ESCOLA DE DOUTORAMENTO INTERNACIONAL EN CIENCIAS E TECNOLOXÍAS DA USC

Feras Mahmoud Naji Awaysheh

Tese de doutoramento

Big Data and large-scale data Analytic: Efficiency of Sustainable Scalability and Security of Centralized Clouds and Edge Deployment Architectures

Santiago de Compostela, 2019

PD en Investigación en Tecnoloxías da Información

CENTRO INTERNACIONAL DE ESTUDOS DE DOUTORAMENTO E AVANZADOS DA USC (CIEDUS)

TESIS DE DOCTORADO

BIG DATA AND LARGE-SCALE ANALYTICS: EFFICIENCY OF SUSTAINABLE SCALABILITY AND SECURITY OF CENTRALIZED CLOUDS AND EDGE DEPLOYMENT ARCHITECTURES

Presentada por:

Feras Mahmoud Naji Awaysheh

Dirigida por:

José Carlos Cabaleiro
Tomás Fernández Pena

ESCUELA DE DOCTORADO INTERNACIONAL PROGRAMA DE DOCTORADO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)

SANTIAGO DE COMPOSTELA
19 de December de 2019

DECLARACIÓN DEL AUTOR DE LA TESIS

Big Data and large-scale Analytics: Efficiency of Sustainable Scalability and Security of Centralized Clouds and Edge Deployment Architectures

Don Feras Mahmoud Naji Awaysheh

Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:

1. *La tesis abarca los resultados de la elaboración de mi trabajo.*
2. *En su caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
3. *La tesis es la versión definitiva presentada para su defensa y coincide con la versión enviada en formato electrónico.*
4. *Confirmo que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*

En Santiago de Compostela, 19 de December de 2019

Fdo. Feras Mahmoud Naji Awaysheh

Autorización del Director/Tutor de la Tesis

Big Data and large-scale Analytics: Efficiency of Sustainable Scalability and Security of Centralized Clouds and Edge Deployment Architectures

Don José Carlos Cabaleiro, Profesor Catedrático da Área de Sistemas Intelixentes da Universidade de Santiago de Compostela

Don Tomás Fernández Pena, Profesor Asociado da Área de Sistemas Intelixentes da Universidade de Santiago de Compostela

INFORMAN:

*Que la presente tesis, corresponde con el trabajo realizado por **Don Feras Mahmoud Naji Awaysheh** bajo nuestra dirección, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores de ésta no incurre en las causas de abstención establecidas en Ley 40/2015.*

En Santiago de Compostela, 19 de December de 2019

Fdo. José Carlos Cabaleiro	Fdo. Tomás Fernández Pena
Director/a tesis	Director/a tesis

Dedication

I am dedicating this thesis to my beloved parents, who have meant and continue to means so much to me, **”Mom and Dad, this is for you.”** To my brothers and sister: **Enas, Anas, and Ahmad** for their love, endless support, and encouragement. And last but not least, the spirit of my life, my daughter **(Noor)**.

Are those who have knowledge equal to those who do not have knowledge

Holy Qura'n, Surah Az-Zumar 39:9

And Allah taught Adam all the names...

Holy Qura'n, Surah Al-baqarah 2:31

When a man dies, his deeds come to an end except for three things: Sadaqah Jariyah (ceaseless charity); a knowledge which is beneficial, or a virtuous descendant who prays for him (for the deceased)

Prophet Mohammad

Acknowledgments

All praise and thanks are due to the Almighty Allah, who always guides me to the right path and has helped me to complete this chapter of my life.

There are many people whom I have to acknowledge for their support and encouragement during the journey of preparing this thesis. So, I will attempt to give them their due here.

First and foremost, I present my sincere gratitude to my parents, my dad Mahmoud (Abo Firas) and my mom Amoun, for their patience and encouragement to achieve my dream throughout studying and researching. I also thank my brothers and sister, Enas (Um Zaid), Anas (Abo Mahmoud), Ahmad (Konz), Anas Abu Rumman, and Diya Salloum for being my friends and soulmates forever. And for the one, I see my self in her eyes, my daughter Noor.

I would like to express my thanks appreciation and gratitude to my supervisors, Assoc. Prof. José Carlos Cabaleiro and Assoc. Prof. Tomás Fernández Pena for the tremendous efforts they have made to accomplish this work and for giving me a free hand to implement all my crazy thoughts.

Heartfelt gratitude to my brothers Eyas Al Awaysheh (Abu Karam; even with his absence), Hussam G. Al Shamat, and Bashar Abo Ruman, and special thanks to Ali Al Rahalah. They have had a great impact by belief in me and provides support and assistance despite the distance between us.

I would like to thank all my colleagues at the Centro de Investigación en Tecnoloxías da Información (CiTIUS) at the Universidade de Santiago de Compostela. Especially, Alberto Suárez Garea, Rubén Arenas Hernán, and all of the support staff (Secretary and technical employees).

Thank to all my Jordanian colleagues Ahmad Abu Roman, Ahmad Abu Shattal, Ali Al Assofi, Ameed Al Momani, Ala'a Al Nuimat, Bilal Al Tawarah, Hatem Al Ameryeen, Ikremah Al Qudah, Khalid Al kharabsheh, Ma'un Al Tawarah, Mutaz Al Mobydeen, Shahed Al Mobydeen, Saadi Al Abadi, Anas Al Rahammneh, Taha Al Marayeh, Abdulatif Al Marayeh, Tareq Al Zubi, Zakria Al Tarawneh, Ziyad Al Hamory, Mohammad Al Adwan, Marwan Al Ajoury, Saqer Abu Shatal, Hiba Al Atyat, Ali Al Wardat, Mohammad Al Da'ajeh, and especially, Sattam Al Matarneh, Mohammad Al Zubi, and Zaid Al Lami (even that he didn't mention me in his thesis).

This work has received financial support from the SDNHPC: Solutions for new challenges in high-performance computing project. Projects of Excellence, State Program for the Promotion of Scientific and Technical Research of Excellence, Ministry of Economy, Industry, and Competitiveness of Spain, TIN2016-76373-P

I would also like to thank the guys at Charles Darwin University, Australia for the great experience in the three months I spent there. I learned a lot from them, especially from the magnificent Prof. Mamoun Alazab who become a tutor for me in the academic field and his family Kate, Amo Ayah and Sarah. Also, I would like to thank guys at the Synchrotron-light for Experimental Science and Applications (SESAME), especially Salman Matalgah and Mostafa Ali Zoubi. I also would like to thank Dr. Alireza Jolfaei from Macquarie University, Australia for his insightful comments and suggestions on my work. And Dr. Rosa Filgueira from EPCC research center at the University of Edinburgh, UK for here support.

Also, I want to thank Dr. Hisham Atyat, Dr. Khalid Abu Amirah, El Sayed Sadek Tohffah, Saman A Ali and all my dear friends in the USA, Australia, and Europe.

I will not forget to thank all of my aunts, uncles, and cousins for their support. Also, in memory of my Grand Mother (Um Mefleh), Uncle Ahmad (Abu Ashraf), Uncle Suliman (Abu Ta'an), and aunt Aminah (Um Khaldun), they based away while I was doing my Ph.D. and they had faith in me. You will always remain in my thoughts, and I hope your souls will find peace in heaven alongside my Grandfathers Naji and Mohammad, and Grandmother Nofah.

Last but not least, to everyone who believes in me and my success, it is an honor to pour out my heart on this stage for you.

19 de December de 2019

Summary

One of the significant shift of next-generation computing technologies will certainly be in the development of Big Data (BD) platforms. This trend results in a wide range of revolutionary and state-of-the-art enhancements within the data science. Apache Hadoop, the BD landmark, evolved as a widely deployed large-scale data operating system. In principle, Hadoop is designed to utilize clusters of commodity machines. Over the years, the model was ported to be compatible with different types of architecture and paradigms. Its new features include federation configuration to provide Hadoop with the maturity to serve different markets.

This dissertation focuses on the architectural elements of the BD processing frameworks and focusing on the scalability aspects to improve its performance and security. We propose a hybrid BD execution environment (called EME) that keeps spawning a containerized DataNodes for the benefit of BD applications. A dynamic provisioning scheduler (called OPERA) is presented to take advantage of the underutilized on-premise and cloud resources. The results demonstrate that OPERA has immense potential as it significantly decreases the time of execution up to 74% for CPU bound jobs (as in the PiEst benchmark), and up to 26% for HDFS bound jobs (as in the wordcount benchmark) compared with a native Hadoop cluster. Also, the privacy and security of BD architectures are discussed as a significant concern among practitioners. A BD federation single-sign-on authentication module and a novel access broker framework are introduced. Experimental results demonstrate the efficiency of the proposed access broker with only 1% impact on the Hadoop performance compared with a non-secure one. Finally, a modern secure case of study regarding data streaming of edge nodes to the clouds in vehicular clouds is explained to validate the thesis findings.

Keywords: Big Data, Cloud Computing, Internet of Things, Resource Management, Security and Privacy, Access Control, Performance Optimization, Docker Containers.

Resumen

Uno de los cambios significativos de las tecnologías de computación de próxima generación será, sin duda, el desarrollo de las plataformas Big Data (BD). Esta tendencia da como resultado una amplia gama de mejoras revolucionarias y vanguardistas dentro de la ciencia de datos. Apache Hadoop, punto de referencia de BD, evolucionó como un sistema operativo de datos a gran escala ampliamente desplegado. Hadoop está diseñado para utilizar clusters de máquinas de propósito general. A lo largo de los años, el modelo fue adaptado para ser compatible con diferentes tipos de arquitectura y paradigmas. Sus nuevas características incluyen la configuración de federación para proporcionarle madurez para servir a diferentes mercados.

Esta tesis se centra en los elementos arquitectónicos de las estructuras de procesamiento de BD y en los aspectos de escalabilidad. Proponemos un entorno híbrido de ejecución de BD (llamado EME) que sigue generando un DataNode contenedorizado para el beneficio de las aplicaciones BD. Se presenta un programador de aprovisionamiento dinámico (llamado OPERA) para aprovechar los recursos infrautilizados en entorno local y en la nube. Los resultados demuestran que OPERA tiene un inmenso potencial, ya que reduce significativamente el tiempo de ejecución hasta un 74% para los trabajos vinculados a la CPU (como en el benchmark PiEst), y hasta un 26% para los trabajos vinculados a HDFS (como en el benchmark Wordcount) comparado con un grupo nativo de Hadoop. Además, la privacidad y la seguridad de las arquitecturas de BD se trata como una preocupación significativa entre los profesionales. Se introduce un módulo de autenticación de inicio de sesión único de la federación BD y un nuevo marco de trabajo de broker de acceso. Los resultados experimentales demuestran la eficiencia del broker de acceso propuesto, con un impacto de sólo el 1% en el rendimiento de Hadoop en comparación con un broker no seguro. Finalmente, se explica un caso de estudio moderno y seguro sobre la transmisión de datos de los nodos de borde a las nubes en nubes vehiculares para validar los resultados de esta tesis.

Hoy en día, la industria y el mundo académico están impulsados por tecnologías como el Big Data (BD), y entre sus usos se incluye la adaptación de aplicaciones de alta productividad para superar los desafíos en múltiples disciplinas. Esta amplia heterogeneidad de aplicaciones depende de los recursos informáticos para procesar sus trabajos y ofrecer el servicio de forma eficiente. Apache Hadoop, es un paradigma prominente que combina nodos de computación a gran escala para construir un entorno analítico, por ejemplo, para el procesamiento y almacenamiento BD, tanto en bases de datos locales como en la nube. Este estudio tiene como objetivo analizar y caracterizar los factores más críticos de la eficiencia de las arquitecturas para el despliegue de entornos BD sostenibles. Concretamente, hemos puesto nuestra atención en evaluar las características de escalabilidad y seguridad de los sistemas BD basados en Hadoop, así como a diseñar soluciones novedosas para sus limitaciones actuales. Las dos preguntas clave que se discuten en esta tesis con respecto a la optimización de la infraestructura de BD (BDI) son (I) Cómo optimizar la escalabilidad de los recursos de BD con el aprovisionamiento dinámico y elástico de recursos de bajo coste para aumentar la capacidad bajo demanda con el objetivo de mejorar el rendimiento general, y (II) Cómo mantener los clústeres de BD y la privacidad de los datos dentro de las arquitecturas de despliegue local y las basadas en la nube, utilizando un agente de control de acceso ligero y distribuido. La Tabla 1 resume los desafíos y problemas que estamos abordando con esta tesis.

Uno de los cambios tecnológicos más importantes de los sistemas informáticos actuales ha surgido con las plataformas Big Data (BD). Esta tendencia ha dado lugar a una amplia gama de mejoras revolucionarias y de última generación dentro de la ciencia de datos que han dado lugar al novedoso concepto de Big Data as a Service (BDaaS). Apache Hadoop, una de las principales aplicaciones BD, ha evolucionado como un sistema operativo de datos a gran escala ampliamente desplegado. Las nuevas características proporcionan a Hadoop 3.x la madurez y aplicabilidad para servir a diferentes mercados. Sin embargo, el rendimiento y la seguridad de estos sistemas siguen siendo las principales preocupaciones de los profesionales. Estos problemas han llamado la atención de los investigadores en ciencias de la información y de los responsables de la toma de decisiones. En esta propuesta de doctorado, pretendemos hacer frente a dos retos críticos abiertos en las actuales arquitecturas de despliegue de Big Data, que son la escalabilidad y la seguridad del sistema. Para ello, hemos empleado la tecnología de contenedorización y diseñado un robusto control de acceso dentro de una arquitectura de recursos híbrida (dedicada y no dedicada), que soporta la última plataforma Hadoop 3.x. El estudio también pretende proponer tecnologías de integración para producir

Table 1: Problemas abordados en este trabajo.

Num.	Problem Description	Type
1	Baja utilización de la mayoría de los computadores y clústeres actuales.	Utilización
2	Particionamiento estático en el espacio físico de una empresa.	Flexibilidad
3	Ajuste estático del tamaño del cluster en el pico de utilización e instalación de nuevos servidores para aumentar la productividad bajo demanda.	Flexibilidad
4	TIncremento del gap entre las capacidades de computación y de E/S en estaciones de trabajo para experimentos científicos.	Rendimiento
5	Dependencia en tenencia múltiple (por ejemplo, cloud computing), lo que provoca muchos problemas como seguridad, movimiento de datos, pérdida de rendimiento, etc.	Escalabilidad
6	Pérdida importante de ciclos de computación para la comprobación de middleware y aplicaciones que no están todavía en producción.	Utilización
7	Necesidad de mejora del rendimiento con un coste mínimo de despliegue.	Rendimiento.
8	Ejecución de un nuevo tipo de experimentos en los mismos recursos con mínima configuración, manteniéndola extensible a cualquier diseño previo.	Flexibilidad
9	Asegurar que usuarios y aplicaciones interactúen de forma segura con funcionalidades principales del sistema y garantizando el acceso a los datos a través del clúster.	Seguridad
10	Controlar la capacidad de una organización, individuo y subcomponentes de un sistema sobre qué datos de un sistema BD pueden ser compartidos con terceras partes.	Privacidad

un entorno híbrido fiable y de bajo coste, que amplíe el planificador de tareas de Hadoop, teniendo en cuenta los requisitos de calidad de servicio y seguridad. Por último, hemos evaluado el diseño de la arquitectura propuesta comparandola con el sistema Hadoop actual mediante la realización de una evaluación exhaustiva de su rendimiento y fiabilidad, tanto en los modelos de implementación centralizada como en los modelos de despliegue en el borde.

Antecedentes:

El Big Data constituye un extenso conjunto de datos (Terabytes o Petabytes) que ha sido creado o recopilado de muchas fuentes, incluyendo datos sociales, datos de máquinas y datos transaccionales (por ejemplo, sensores inteligentes, experimentos científicos, medios sociales, etc.). Se han presentado varias propuestas para gestionar, procesar y mantener estos conjuntos de datos masivos. Uno de los modelos de programación más importante en lo que se ha dado en llamar procesamiento intensivo de datos es el modelo MapReduce. Esta sección presenta información básica sobre computación intensiva en datos, el modelo de programación MapReduce y los sistemas de computación voluntaria, junto con el trabajo relacionado desarrollado en esta tesis.

- Computación intensiva en datos.

  La computación intensiva en datos es una subclase del conjunto de aplicaciones de computación paralela, cuyo objetivo es procesar grandes volúmenes de datos (normalmente terabytes o petabytes). Con el fin de gestionar estas tareas, se utiliza un centro de datos o un sistema distribuido, es decir, un clúster o una red, que utiliza un gestor de recursos para apoyar la homogeneidad o heterogeneidad del entorno, con una variedad de recursos informáticos y de almacenamiento dedicados y no dedicados.

- Plataforma Apache Hadoop.

  Hadoop es la implementación de código abierto líder del paradigma MapReduce. Ha sido desarrollada bajo el paraguas de la Apache Software Foundation y que ha sido ampliamente utilizado en la industria en todo el mundo. Se compone de dos subsistemas: el de gestión de recursos y ejecución de tareas MapReduce y el sistema de archivos distribuidos Hadoop (HDFS), que se encarga de la gestión y acceso de los datos. HDFS utiliza un proceso denominado NameNode que se ejecuta en los nodos maestros y múltiples procesos DataNode que se ejecutan en los esclavos. Además, Hadoop sigue el paradigma maestro/esclavo, en el que un gestor recursos (denominado YARN) se encarga de asignar recursos a los diferentes trabajos mientras que los gestores de tareas (denominado Application Máster) se encarga de iniciar, supervisar y volver a ejecutar las tareas que provoquen fallos en las máquinas esclavas. Además, Hadoop implementa sistemas de colas, como el Hadoop Fair Scheduler, para garantizar un reparto justo de los recursos entre los múltiples usuarios y trabajos.

- Seguridad de Apache Hadoop.

  Muchas de las amenazas de seguridad relacionadas con los frameworks de BD se pueden mitigar utilizando procesos y técnicas de seguridad tradicionales. Sin embargo, algunas amenazas de seguridad requieren soluciones específicas de BD. Los sistemas Hadoop tienen diferentes vulnerabilidades de seguridad y pueden estar expuestos a diversas amenazas. La abstracción por capas de seguridad del sistema BD Hadoop implica varias funciones esenciales. Las tres primeras capas, identificación (ID) y control de acceso, autenticación y autorización se combinan para representar los procesos de gestión de acceso del clúster Hadoop (un acceso de pasarela). Las otras capas de seguridad (gobierno de datos, integridad, confidencialidad y auditoría de seguridad) son requisitos de servicios de seguridad y componentes de la pila de datos de Hadoop.

Objetivos de la investigación.

En este estudio hemos diseñado, construido y evaluado una arquitectura de computación distribuida híbrida (recursos dedicados y no dedicados) efectiva y eficiente dentro de un cluster heterogéneo tanto en potencia de computación como en disponibilidad (Computación Oportunista) para extender el planificador de tareas de Hadoop en una variedad de sistemas operativos. Además, proporcionaremos información sobre los retos de identificación y gestión de acceso (IAM) que imponen las próximas versiones de Hadoop 3.x. Esta tesis trata sobre el control de acceso federado para la pila de BD de Hadoop, que incluye todas las etapas significativas y refleja los detalles específicos del control de acceso dentro de los clústeres Hadoop, tanto de las arquitecturas de despliegue en la nube como de las arquitecturas de despliegue en las instalaciones. Además, se han llevado a cabo experimentos y simulaciones para evaluar el sistema propuesto. Del mismo modo, se han caracterizar todas las mejoras introducidas en el diseño propuesto en base a la escalabilidad y seguridad de Hadoop.

Los objetivos principales de este estudio han sido:

Objetivo 1: Evaluar los actuales entornos de BD y Apache Hadoop mediante un análisis exhaustivo de los elementos arquitectónicos de los frameworks de BD con información sobre los retos abiertos y las tendencias de investigación. Asimismo, identificar los inconvenientes actuales de la plataforma Hadoop en términos de aprovisionamiento elástico de recursos y privacidad de datos, comparando y contrastando las características de estos frameworks.

Objetivo 2: Diseñar e implementar una arquitectura híbrida de recursos de computación distribuidos para mejorar la planificación de tareas BD dentro de las arquitecturas de implementación tanto en la nube como en las instalaciones.

Objetivo 3: Examinar uno de los retos más importantes de las plataformas de BD de próxima generación, es decir, la privacidad de los datos. Además, desarrollar un módulo de autenticación fiable y enchufable para que los clientes externos de BD puedan iniciar sesión en la infraestructura de BD (BDI).

Objetivo 4: Evaluar experimentalmente la usabilidad de los sistemas de seguridad de Apache BD utilizando Hadoop 3.x. El control de acceso a la plataforma (incluyendo Apache

Knox y Apache Ranger) y el despliegue de seguridad de análisis de datos de borde utilizando un control de acceso robusto y ligero.

Objetivo 5: Analizar las soluciones y tecnologías de BD en beneficio de un flujo de datos y aplicaciones de vanguardia, como el Sistema de Transporte Inteligente (ITS), proponiendo un marco de seguridad ITS de varios niveles para validar la usabilidad de la solución.

Objetivo 6: Evaluar exhaustivamente los frameworks propuestos y el rendimiento de los prototipos y caracterizar las mejoras logradas comparables al modelo nativo de Hadoop y su impacto en el futuro del BD infraestructura.

Metodología de estudio

El objetivo principal de nuestro estudio ha sido diseñar un sistema de distribución híbrido eficiente para mejorar la planificación de tareas Hadoop y la seguridad de borde en un entorno de clústeres heterogéneos. Además, hemos formalizado el diseño de soluciones BD seguras dentro de la pila Apache Hadoop, utilizando un gestor de acceso que ayude en la detección de brechas de seguridad y en las investigaciones forenses digitales. La consecución de estos objetivos se ha realizado en las siguientes fases:

En primer lugar, revisar y analizar críticamente la literatura de BDI a través de una serie de análisis teóricos y prácticos de los paradigmas y entornos actuales de la implementación de soluciones de BD para determinar las lagunas en la investigación, junto con las tendencias futuras. En segundo lugar, identificar la degradación del rendimiento del planificador de Hadoop en entornos heterogéneos e identificar las deficiencias actuales en la gestión de recursos para proponer un prototipo de aprovisionamiento de recursos dinámico y elástico para los frameworks de procesamiento de BD. Esta fase incluye la definición y el examen de las herramientas y tecnologías de software necesarias, incluyendo HTCondor, Apache Yarn, MapReduce y tecnologías de contenedorización (por ejemplo, Docker).

Tercero, caracterizar la limitación actual de la plataforma Hadoop 3.x basándose únicamente en sus medidas de control de acceso originales para mantener la privacidad de los datos, en cuanto a la complejidad de la implementación y el consumo de recursos. Esta fase incluye el diseño e implementación de un agente de acceso ligero y distribuido y el examen de las herramientas y tecnologías necesarias, incluyendo Apache Ranger y Knox.

La cuarta fase incluye la implementación de los sistemas propuestos, tanto en clústeres en la nube como en clústeres locales, y demostrar su eficacia. Esta validación incluye la realización de pruebas intensivas de los sistemas propuestos para evaluar su rendimiento y robustez en la práctica. En la quinta fase, las lecciones aprendidas y los mecanismos de verificación han servido para modificar las herramientas seleccionadas, editar el diseño de la arquitectura y, por defecto, actualizar las fases tres y cuatro. A continuación, se han desplegado enfoques de control de acceso y se ha llevado a cabo un análisis exhaustivo de la seguridad de los nodos de borde a gran escala (basados en data streaming).

Posteriormente, en la sexta fase, llevamos a cabo un caso de uso complejo, mediante la implementación de un ejemplo de Internet de las cosas (IoT), concretamente una solución basada en la nube para la gestión de Sistemas Inteligentes de Transporte (ITS), a saber, un sistema de transporte moderno conectado a la red basado en tecnologías BD.

Resultados y publicaciones:

- En nuestro artículo, EME: An Automated, Elastic and Efficient Prototype for Provisioning Hadoop Clusters On-demand, presentado en la International Conference on Cloud Computing and Services Science, CLOSER 2017, propusimos una aproximación que amplía las aplicaciones basadas en MapReduce con un aprovisionamiento dinámico y un aumento de recursos de bajo coste bajo demanda. Este estudio pretende dar soporte a entornos heterogéneos extendiendo la plataforma Apache Hadoop 2.x a un entorno oportunista contenedorizado, que mejora la productividad del sistema añadiendo recursos infrautilizados a un clúster local o basado en la nube. El framework propuesto consta de tres componentes principales. En primer lugar están los usuarios. En segundo lugar, un entorno dedicado, es decir, un clúster típico de Hadoop que sigue el modelo maestro/esclavo, donde Apache YARN es responsable de programar, monitorizar y volver a ejecutar tareas de fallo en máquinas esclavas. En tercer lugar, un entorno no dedicado, que podría controlarse utilizando un middleware voluntario. En este sentido, nuestro enfoque adopta un escalado (scaling-out) elástico y horizontal de recursos en una LAN controlada para ejecutar de forma oportunista tareas MapReduce bajo demanda en conjuntos de recursos distribuidos e infrautilizados. Este entorno de procesamiento gestiona y asigna los nodos no dedicados utilizando HTCondor (para crear una red de equipos de escritorio empresarial dentro de la LAN, es decir, un pool oportunista). Además, se ha implementado un planificador ligero, que sólo considera el despliegue de contenedores dentro de la reserva oportunista.

- En nuestro trabajo A Pluggable Authentication Module for Big Data Federation Architecture presentado en el 24º ACM Symposium on Access Control Models and Technologies. SACMAT, 2019, proponemos un modelo fiable para autenticar usuarios y servicios sobre una arquitectura de despliegue de BD federada. El objetivo principal de este modelo es proporcionar un enfoque de inicio de sesión único (SSO) para la última plataforma Hadoop 3.x. Para ello, se propone un modelo conceptual que combina las primitivas de control de acceso de Hadoop y el framework de Apache Knox. Este estudio presenta los primeros pasos para definir y formalizar una pasarela de servicio para una arquitectura de despliegue de BD federada (BDF). La pregunta clave que nos plantearemos para abordar el reto del control de acceso es cómo diseñar un SSO abstracto que maneje el acceso a datos/servicios en una federación de Hadoop. Abordamos esta cuestión mediante la definición de una arquitectura de referencia de inicio de sesión único (SSORA) para el desarrollo seguro de BDF. El trabajo presentado en este documento puede emplearse en un entorno de Hadoop federado en nubes de tenencia múltiple BDaaS e IaaS, así como en centros de datos locales.

- En el artículo Big Data Security Frameworks Meet the Intelligent Transportation Systems Trust Challenges presentado en el 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, demostramos cómo muchos casos tecnológicos explotan la ciencia de datos y los frameworks de BD en los últimos años. Se propuso un trabajo conceptual que integra los enfoques de seguridad de diferentes disciplinas en una arquitectura de referencia coherente, empleando el Sistema Inteligente de Transporte (ITS) como un caso de uso. Además, se presentó una clasificación de tecnologías, productos y servicios de BD para hacer frente a los retos de confianza de los ITS. En el estudio se proponía un nuevo marco de seguridad de varios niveles para validar la usabilidad del prototipo con el desarrollo de la inteligencia de negocio en el ámbito empresarial.

- En nuestro artículo (en el último estado del proceso de revisión), Next-Generation Big Data Federation Access Control: A Reference Model, enviado a la revista Future Generation Computer Systems, ampliamos la discusión sobre la madurez y aplicabilidad de Hadoop 3.x para servir a diferentes mercados. Además, indicamos ideas sobre el control de acceso y los retos de acceso externo dentro del concepto de Big Data as a Service (BDaaS). Este estudio representa el primer paso en el campo académico hacia el fortalecimiento de las grandes auditorías de seguridad y privacidad de datos a nivel de servicio en arquitecturas federadas de despliegue con un entorno robusto de Gestión de Identificación y Acceso (IAM) para la detección y prevención del acceso no autorizado. También ofrece una visión y una hoja de ruta completa para propagar el control de acceso efectivo a través de los modelos de BD de tenencia múltiple, proporcionando información sobre los últimos desarrollos y los retos abiertos en este campo. Esta investigación destaca, en primer lugar, la necesidad de un gestor de control de acceso robusto que maneje las operaciones de autenticación y autorización dentro de una federación. A continuación, analiza los requisitos actuales de análisis de seguridad y registros de auditoría de acceso dentro de cualquier arquitectura de despliegue. El artículo presenta el modelo FACRM (Federated Access Control Reference Model) que incluye todas las etapas principales y refleja los detalles del control de acceso dentro de los clusters de Hadoop, tanto de los modelos en cloud como de los modelos locales. Validamos la usabilidad de FACRM proponiendo un nuevo gestor BD federado. Nuestros resultados demuestran tanto la efectividad como la eficiencia de nuestro patrón de arquitectura para mantener la privacidad de datos y clústeres y permitir un análisis sofisticado de los registros de acceso. La eficiencia del gestor de acceso propuesto no se ha visto afectada por sobrecargas en el rendimiento. Nuestros resultados experimentales muestran sólo un 1% de incremento de tiempo por cada operación de lectura/escritura de 100MB en un WebHDFS.

- Nuestro artículo en revisión, Big Data, Apache Hadoop Stack, Resource Management, Infrastructure Comparison, enviado a la revista Future Generation Computer Systems (FGCS), presenta un estudio exhaustivo de los elementos arquitectónicos de los diversos entornos de procesamiento de BD y también propone una taxonomía novedosa. Además, se discuten en detalle las características de los distintos modelos de despliegue de los frameworks de procesamiento de BD y de gestión de recursos. Por último, se exponen los retos actuales, así como las orientaciones futuras para mejorar el estado actual de la infraestructura de los frameworks de BD. En este estudio, nos centramos en los elementos arquitectónicos de los frameworks de BD que utilizan diferentes entornos y discutimos sus características más destacadas, centrándonos en los aspectos de gestión de recursos de BD. Proponemos un criterio sobre cómo comparar la correlación/divergencia de estos entornos de ejecución con respecto a la prestación de servicios BD.

- Además, este trabajo tiene por objeto identificar y comprender los conceptos que deben tenerse en cuenta en el desarrollo de los sistemas de DIA (Data-Intensive Application). La revisión introduce las estrategias de optimización descritas en la literatura reciente y discute los temas pendientes y desafíos que surgen al paralelizar el análisis extensivo de datos en estos entornos. Adicionalmente, esboza las tendencias futuras para la gestión de los recursos de DIA y discute a alto nivel los análisis de datos. Presentamos una taxonomía de los requisitos de implementación de los frameworks de procesamiento de BD, que cubre los diferentes objetivos de optimización utilizando un enfoque de clasificación multidimensional. Estas dimensiones representan las bases sobre las que se basan en gran medida el rendimiento de los sistemas de aplicaciones intensivas en datos (DIA), es decir: a) recursos, b) seguridad, c) tareas y d) gestión de datos. En base a la taxonomía propuesta, discutimos los temas pendientes y los desafíos planteados en el análisis de datos utilizando frameworks de procesamiento de BD en diferentes entornos no tradicionales y esbozamos las soluciones correspondientes en la literatura con nuevas direcciones para futuras mejoras. Además, discutimos cómo difieren estos sistemas utilizando dos casos de uso basados en la ubicación del recurso. En particular, presentamos una clasificación de las implementaciones existentes utilizando entornos basados en LAN y WAN, centrándonos en la prestación de servicios de alta calidad (QoS) en cualquier entorno.

Conclusiones

Las arquitecturas Big Data (BD) tienen una larga tradición de aprovechar las tendencias y tecnologías de la comunidad informática en general. En esta tesis, realizamos un análisis en profundidad de dos componentes principales de la pila de Big Data de Apache Hadoop, a saber, el gestor y planificador de recursos Apache YARN y el sistema de archivos distribuidos Hadoop. Este estudio considera tanto la escalabilidad como la posibilidad de mejorar el rendimiento y la productividad general, así como la privacidad de los datos almacenados a gran escala y de los recursos subyacentes como base de su investigación. Esta tesis ha logrado con éxito sus objetivos declarados al demostrar que es factible escalar de forma oportunista bajo demanda los recursos utilizando contenedores. Además, esta tesis ha defendido y probado con éxito la hipótesis de la preservación de la privacidad de BD mediante la aplicación de sistemas distribuidos de un esquema de control de acceso multiagente. La tesis investiga y evalua el rendimiento de los componentes propuestos y los compara con los originales de pila de Apache Hadoop 3.x, optimizándolos sobre clústeres de BD tanto basados en la nube como en instalaciones locales. En general, la investigación presentada en esta tesis responde a las preguntas iniciales del trabajo la investigación y allana el camino para una amplia gama de mejoras revolucionarias y tendencias futuras dentro de la pila Hadoop.

Contents

CHAPTER 1

INTRODUCTION

1.1 Overview: Supporting Concepts and Materials

Nowadays, industry and academia are driven by trends such as BD; their operations include accommodating high-performance and high-throughput applications to overcome challenges in multiple disciplines and areas. This wide range of application heterogeneity depends on computing resources to process tasks and deliver services efficiently.

BD frameworks function over an extensive set of data that have been created or collected from many sources, including social data, machine data, and transactional data (e.g., smart sensors, scientific experiments, and social media. There have been several proposals toward managing, processing, and maintaining these massive datasets in data-intensive processing with the Apache BD stack. This section presents some background information on data-intensive computing, the MapReduce programming model, and High Throughput Computing (HTC) systems alongside containerization technology. These aspects represent the scope and domain of this thesis.

This dissertation also analyzes and characterizes the most critical factors of efficient and sustainable BD deployment architectures. We have enrolled in evaluating Apache Hadoop scalability and security features, as well as novel design solutions for its current limitations.

- Data-Intensive Computing.

  Data-intensive computing is a subclass of parallel computing applications, which process large volumes of data using a dedicated datacenter or distributed system, i.e., a cluster that supports a distributed file system. These systems maintain the homogeneity or heterogeneity of the environment in a variety of dedicated, non-dedicated computational and storage capacities aiming to handle the process of the data-intensive task.

- High Throughput Computing and High Performance Computing.

  High Performance Computing /High Throughput Computing systems (HPC/HTC) are research trends that are intended to run parallel-processing codes, visualization, and scientific applications. They represent two very different computational models, both in implementation as well as in the resources required to run them. In short, HPC brings enormous amounts of computing power to bear over relatively short periods. Meanwhile, HTC systems employ large amounts of computing power for very long periods. This fact is critical, as throughput is the primary limiting factor in many scientific and engineering efforts.

  HPC systems enable users to run a single instance of parallel software over many processors (a single job with many tasks over many processors). It is considered high-performance computing when a large number of high-end resources are required to complete the task in a reasonable amount of time (by high-end resources, we mean many cores per CPU with dozens or hundreds of gigabytes of RAM for each CPU). A HPC cluster is optimized for large parallel scientific applications that spread processing across multiple nodes and require efficient communication between the running of applications.

  However, HTC systems are more suited to run multiple independent instances of the software on multiple processors at the same time (a computational task). An HTC system is used when a task is large and requires a significant number of resources, but those resources can be basic. The developer does not need high-end CPUs. In a way, the HTC design is more tolerant of managing failures and is based on unreliable components. If work is given to every node, the results eventually come back. If some of the nodes fail, the jobs can be restarted on a different system. The key to HTC is the effective management and exploitation of all available computing resources (geographical distance to each other, acting like a grid). Nearly all HTC jobs can be classified as "embarrassingly parallel," which means the workload can be divided up into multiple, autonomous pieces, each of which is capable of being independently executed.

  While both HPC and HTC jobs run parallel computations, HTC requires far less communication and synchronization between nodes. In the meantime, HPC workloads are computation and data-intensive in nature. Furthermore, HTC jobs have, by nature, extremely short runtimes, usually in the millisecond range.

- Volunteer and Opportunistic Computing.

  Volunteer Computing (VoC) is a form of network-based distributed computing, which allows public participants to share their idle computing resources and helps run computationally expensive projects. In general, VoC take advantage of resources donated by participated among spread computer grid or cluster by following the master-worker parallel computing model, which makes computing cycles to form a large-scale virtual supercomputer. The low-cost advantage of VC has made this platform attractive to many research scientists whose projects require considerable computing power, and are used mainly for loosely coupled (independent) processing tasks and committed jobs.

- Containerization Technology.

  Containerization computing, in general, is an OS-level virtualization technique employed to deploy and run distributed applications without launching an entire virtual machine (VM) for each service. The basic idea behind it is that it enables multiple isolated applications to run on a single host and access the same OS kernel with the perception of its own process tree, users, and filesystem. As such, containers have very few overheads compared to VMs, with near-instant startup time and resource impact on the host system. However, unlike VMs, containers are not flexible when it comes to details such as the kernel version and instruction set and inherits those from the host.

  The increase in the popularity of containers can be attributed to their lightweight nature (the time to start a container is on the order of milliseconds). Moreover, containers also incur a lower performance overhead when compared to VMs. To virtualize the OS, a container runtime needs to be able to allocate resources to containers and isolate the view of the system from within the containers: leverage Linux kernel.

1.1.1 Used Frameworks and Technologies

Apache Hadoop

Apache Hadoop is a leading open-source implementation platform of frameworks that has evolved to become the BD defacto stack and ecosystem of technologies. Hadoop was founded and powered by the Apache Software Foundation. It has been widely used in both the industry and academia to store and process massive data sets. When it was first realized, Hadoop only ran the MapReduce framework to process the data stored in the Hadoop Distributed File System (HDFS). This architecture then evolved when Hadoop 2.x came up with a new framework and resource manager, Yet Another Resource Negotiator (YARN), which provides the ability to run non-MapReduce jobs such as Spark, Flink, Giraph, Storm, Hive, and HBase coprocessors.

Apache Hadoop 3.x is the latest version and offers many new features. These features include: (i) Advanced HDFS with erasure encoding, which reduces the storage overhead, maintaining the same level of data recoverability; (ii) support of long-running applications and an enhanced YARN timeline service; (iii) better integration with container-based environments; (iv) the improvement of the horizontal cluster scalability, by natively supporting a federated architecture; and (v) the use of backfilling techniques, i.e., opportunistic containers to improve system utilization. Our goal is to utilize these new features to develop a flexible, dynamic, and adaptive BD runtime environment.

Hadoop abstracts computing resource management, task scheduling, and data management, while maintaining a satisfactory level of security and isolation. The Hadoop stack is typically deployed as part of a large-scale BD platform to support commodity hardware and accommodate different processing frameworks. It is utilized to handle data management and access to the Hadoop ecosystem using a master/slave architecture. The large community behind Hadoop has been working to improve its stack to meet the increasing demands and requirements of BD. The new features shift proves the maturity and applicability of Hadoop 3.x to serve different markets, thus, making it a prominent paradigm that combines large-scale computing nodes to build an analytical environment, e.g., BD processing and storage.

Apache Hadoop Architecture

It is a cluster resource-management platform that is responsible for scheduling users' application tasks, implementing security controls, and managing the computing resources, as well as providing the high availability features of Hadoop. This new architecture removes the old `JobTracker` (in Hadoop v 1) from managing resource usage across applications, which removes the single point of failure and enhances the scalability, thus allowing it to run other applications apart from MR ones. YARN uses a global Resource Manager to receive and run applications on the slaves, and a Job History Server to provide information about completed jobs. On the slave side, the `NodeManagers` (previously known as Task Tracker in Hadoop v1) provides information about the node status and capacity; (iii) An execution framework (e.g., MR), which runs per application `ApplicationMaster` to manage the different jobs. The ApplicationMaster asks for resources from the `ResourceManager`, which, with the information provided by the `NodeManagers`, provides the resources in the form of containers. The `ApplicationMaster` runs the application in those containers, controlling its behavior, and re-launching failed tasks as needed.

HTCondor

HTCondor is an open source that has been empowered by the University of Wisconsin–Madison as a high-throughput computing framework for batch services and coarse-grained distributed parallel computing of intensive tasks. It can be used to manage the workload on a dedicated cluster of computers or to farm out work to idle desktop computers —so-called cycle scavenging. HTCondor can seamlessly integrate both dedicated resources (rack-mounted clusters) and non-dedicated desktop machines (cycle scavenging) of opportunistic computing into one computing environment.

HTCondor supports the standard MPI as well as its master/worker library for extremely parallel tasks on massive collections of distributively owned heterogeneous computing resources. HTCondor architecture is composed of three main components: (i) a central manager runs both a `Collector` and `Negotiator` daemon; (ii) on the execute nodes, one or more machine runs a `Startd` daemon; and (iii) one or more machine runs a `Schedd` daemon on the submit nodes.

Docker

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications that use OS-level virtualization to deliver software in packages called containers. In its core, it has evolved into a comprehensive standard for portable services and applications. Docker relies on core Linux kernel container primitives and abstracts them into a user-friendly API. It, thus, allows the developer to create an isolated container of any service that acts like a lightweight virtual machine, which shares the same kernel as the host operating system (OS). This container can be turned into an image and shared with anyone running Linux using the Docker hub.

The Docker framework consists of the Docker Engine, a portable, lightweight runtime and packaging tool with a workflow for building and containerizing applications, and Docker Hub, a cloud service for sharing applications and automating workflows; Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, on data center VMs, and on any cloud.

Experiment setup

Aiming to implement and validate the thesis proposed frameworks and solutions we enrolled in installing and configuring different Hadoop prototypes and clusters over both virtual and bare-metal infrastructures. The first prototype was a proof-of-concept model over virtual machines, as specified in Table 1.1. The second prototype was a homogeneous cluster (as shown in Table 3.1), which implemented a comprehensive evaluation and series of experiments over an on-premise infrastructure. In the third prototype, a collaboration with the University of Texas at San Antonio, TX (UTSA) took place to validate our access broker performance, as explained in Table 1.3. Recently, we were awarded a ten thousand US dollar grant from Google Cloud to implement a full running cluster to extend our work.

Table 1.1: A homogeneous virtual cluster powered by Oracle VM VirtualBox 5.2 .

Virtual Nodes	Hadoop Daemon	Hardware & Configuration
1	NameNode	CPUs: Intel(R) Core(TM) i7-4770k @ 3.50GHz, RAM: 32 GB RAM DDR5 @ 2666MHz,
3	DataNode	HDD: 2 TB, SSD: 512 GB

Table 1.2: A homogeneous on-premise cluster formed by 5 HP Proliant DL 370 G6 servers.

Node Number	Hadoop Daemon	Hardware & Configuration
1	NameNode	CPUs: 2 x Intel Xeon E5506 @ 2.13GHz, RAM: 12 GB RAM ECC DDR3 @ 800MHz,
4	DataNode	HDD: 2 TB HDD
Common criteria: Hadoop 3.1.1 over 4 x Gigabit Ethernet ports.		

Table 1.3: A heterogeneous OpenStack cloud Hadoop cluster over webHDFS.

Node Number	Hadoop Daemon	Hardware &Configuration
2	NameNode	VCPUs: 8, RAM: 16GB, HDD: 160GB
3	DataNode	VCPUs: 4 RAM: 8GB HDD: 80GB
Common criteria: 1 gbps network connection, Ubuntu 16.04 LTS OS.		

1.2 Motivation: Big Data Scalability and Security

In this study, we are committed to designing, building, and evaluating an effective and efficient hybrid distributed computing (dedicated and non-dedicated resource) architecture within a heterogeneous cluster in both computing power and availability (opportunistic computing) environment to extend Hadoop's Task scheduling applications in a variety of operating systems. Moreover, we will provide insights into the Identification and Access Management (IAM) challenges that are imposed by the upcoming releases on Hadoop 3.x. This thesis discusses the federated access control for the Apache BD stack, which includes all of the significant stages and reflects specifics in access control within Hadoop clusters of both cloud and on-premise deployment architectures. We are also committed to conducting extensive experiments and simulations to evaluate the proposed system. Similarly, we will characterize all improvements by the proposed new design in Hadoop's scalability and security.

The primary objectives of our study are:

- **Objective 1:** To evaluate the current BD and Apache Hadoop framework deployment environments by comprehensively surveying the architectural elements of BD processing frameworks by gaining an insight into the open challenges and research trends; to identify the drawbacks of the current Hadoop platform in terms of elastic resource provisioning and data privacy by comparing and contrasting the features of these frameworks.

- **Objective 2:** To design and implement a hybrid-distributed computing resource architecture to improve BD scheduling within both cloud and on-premise deployment architectures.

- **Objective 3:** To discuss one of the most significant challenges of next-generation BD platforms, namely data privacy; to conduct a trustworthy and pluggable authentication module for BD external clients' logins to the BD Infrastructure (BDI).

- **Objective 4:** To experimentally evaluate Apache BD security frameworks' usability using Hadoop 3.x. platform access control (including Apache Knox and Ranger) and edge data analytics security deployment using a robust light-weight access control framework.

- **Objective 5:** To analyse and classify BD solutions and technologies for the benefit of a leading data streaming and edge application, Intelligent Transportation System (ITS) and propose a multi-tier ITS security framework to validate the usability of the solution.

- **Objective 6:** To extensively evaluate the proposed frameworks and prototype performance and characterize the achieved improvements comparable to the Hadoop native model and its impact on the future of BD infrastructure.

1.3 Problem Definition

Nowadays, industry and academia are driven by trends such as Big Data (BD), and their operations include accommodating high-performance and high-throughput applications to overcome challenges in multiple disciplines and areas. This wide range of application heterogeneity depends on computing resources to process tasks and deliver efficient services. Apache Hadoop is a prominent paradigm that combines large-scale computing nodes to build an analytical environment, e.g., BD processing and storage, on both local and cloud bases. This study aims to analyze and characterize the most critical factors of efficiency sustainable BD-deployment architectures. Namely, we enrolled in evaluating BD Hadoop-like scalability and security features, and designed novel solutions for its current limitations.

Table 1.4 summarizes the challenges and issues we addressed through this thesis.

1.3.1 Research Questions

The two key questions that this thesis discusses regarding optimizing BD Infrastructure (BDI) are: (I) How to optimize BD resource scalability with dynamic and elastic resource provisioning of low-cost capacity uplift on-demand to improve the overall performance; and (II) how to maintain BD clusters and data privacy within on-premise and cloud-based deployment architectures using a distributed light-weight access control broker.

Beside the above-mentioned key questions, this thesis discusses the following questions with various insights into the latest ongoing developments and open challenges within this domain:

- What are the main deployment architectures of BD frameworks, and what are their features for improving the state-of-the-art BD processing framework infrastructure?

Table 1.4: The problems we are targeting with this thesis.

Num.	**Problem Description**	**Type**
1	Low utilization of most modern commodity computers and clusters.	Utilization
2	The resources static partitioning within a physical confine of an enterprise between different resource managers.	Flexibility
3	Statically sizing the cluster on peak utilization and in- stalling new servers to enhance the throughput when demand.	Flexibility
4	The increasing gap between computation and I/O capacity on high-end workstations for the scientific experiments and Big Data Analytics.	Performance
5	The dependency on multi-tenant systems (e.g., cloud computing), which raises many issues. For instance, security, data movement cost, and performance degeneration.	Scalability
6	Wasting many computing cycles for testing middleware and applications that are not yet in production.	Utilization
7	The need to enhance performance with minimal cost of deployment.	Performance
8	Running a new type of experiment on the same resources with minimal configuration, keeping it extensible to any prior design.	Flexibility
9	Ensuring users and applications interact securely with system core functionalities and guaranteeing appropriate access to data across the cluster.	Security
10	Controlling the ability of an organization, individual, and system subcomponents to determine which data in a BD system can be shared with third parties.	Privacy

- How can we utilize current Hadoop implementation with cutting edge technologies to improve data-intensive computing performance?

- What is the imposed impact of adding security features (i.e., access control) on Hadoop's overall performance, and how can we unify the efforts towards a unified access model?

- Can the proposed scalability and security features of this work be exploited to data-streaming models of IoT-to-Cloud applications, and how?

1.4 Methodology

The main objective of our study is to design an efficient hybrid distributing system to improve Hadoop task scheduling and edge security in a heterogeneous cluster environment as well as formalizing the design of secure BD solutions within the Apache Hadoop stack using an access broker that aids security breach detection and digital forensic investigations. These goals have been achieved through the following phases.

During the first phase, BDI literature was critically reviewed and analyzed through a series of theoretical and practical analyses of current paradigms and environments for implementing BD solutions to draft research gaps, alongside future trends.

For the second phase, the existing Hadoop scheduler performance degradation in heterogeneous environments was identified along with the current resource management shortcomings to propose a dynamic and elastic resource provisioning prototype for BD processing frameworks . This phase included defining and examining the needed software tools and technologies, including HTCondor, Apache Yarn, MapReduce, and containerization technology (e.g., Docker).

In the third phase, the current Hadoop 3.x platform limitation was characterized, relying solely on its primitive access control measurements for maintaining data privacy, the complexity of deployment, and the consummation of resources. This phase included designing and implementing a light-weight, distributed-access broker, and an examination of the required software tools and technologies, including Apache Ranger and Knox.

During the fourth phase (the proposed systems and frameworks phase) included the implementing of the proposed systems and frameworks over both cloud and on-premise clusters and their effectiveness was demonstrated. This validation included intensively testing the proposed frameworks using a series of evaluations to practically assess their performance and robustness.

During the fifth phase, learned lessons and checking mechanisms took place to modify the selected tools, edit the architecture design, and by default, update phases three and four. Next, access control approaches were deployed, and a thorough security analysis of large-scale edge nodes (data streaming-based) were conducted.

Afterwards, in the sixth phase, we conducted a more challenging use-case by implementing an Internet of Things (IoT) cloud-based solution for the benefit of Intelligent Transportation Systems (ITS), namely, a modern connected vehicles approach, based on cutting-edge BD frameworks and technologies.

During the final phase, we conducted an extensive evaluation of the proposed system and submitted the final architecture design; we then wrote up our thesis and conclusions.

1.5 Thesis Contribution

This thesis has successfully achieved the targeted objectives listed in section 1.2 and has achieved the following contributions:

1. The first contribution of this thesis is the enhancement of BD state-of-the-art resource management by extending the literature review into a comprehensive survey of the architectural elements of various deployment architectures of BD-processing frameworks. A novel taxonomy is proposed to discuss, in detail, the features of the various deployment models of BD frameworks. The BD solution's current challenges are presented along with the future directions for improving future BD-processing frameworks and implementations by comparing and contrasting different solutions.

2. The second contribution is the development of a running prototype that extends the Hadoop YARN resource manager with dynamic provisioning and low-cost resource capacity uplift on-demand. For this, we propose an enhanced MapReduced environment (EME), which supports a heterogeneous infrastructure. The EME architecture extends Apache Hadoop to an opportunistically containerized environment. Doing so enhances the system throughput by adding underused resources to a local or cloud-based cluster. EME's scheduler relies on our novel OPERA framework (OPportunistic and Elastic Resource Allocation), which is a pilot-job paradigm of the Hadoop 3.x framework task execution.

   OPERA continues spawning Hadoop DataNodes, based on Docker containers as worker nodes. BD applications can then use these workers. OPERA is a standalone pluggable architecture that does not require making any modifications to the prior Hadoop design. In OPERA, an opportunistic pool (HTCondor-based pool) and a dedicated YARN cluster can collaborate to improve throughput, with the minimal cost of deployment. In its core, OPERA launches disposable containers among the idle servers (creating an opportunistic container-based cluster) and ensures efficient provisioning for the Hadoop dedicated cluster on-demand.

3. The third contribution proposes the first (to the best of our knowledge) single-sign-on (SSO) approach for the latest Hadoop 3.x platform. This achieved goal discusses one of the most significant challenges of next-generation BD federation platforms, namely Hadoop privacy, with a cutting edge access-control framework. A conceptual model is proposed to achieve this by combining Hadoop access control primitives and the Apache Knox framework. Moreover, we provide a framework called BD Federation Access Broker (BDFAB), which addresses the eight main security limitations of the current Hadoop primitive access control.

   A Federated Access Control Reference Model (FACRM), which aims to formalize the design of secure BD solutions within the Hadoop stack, is introduced. BDFAB is a lightweight framework with minimal performance overheads. Our experiment indicates that our solution will add only 1% for each 100MB read/write operation in a WebHDFS. Overall, our findings in this trend pave the way for a wide range of revolutionary and state-of-the-art enhancements and future trends within the Hadoop stack security.

4. The final contribution of this thesis is the designing of a multi-tier Secure Intelligent Transportation System (SITS) architecture, which comprises the functional components and frameworks of the Apache Hadoop security stack. This model aims to facilitate security pattern design and solution elections by emphasizing the security-by-design concept when constructing ITS systems. In this dissertation, we map the main security features (requirements and threats) of data streaming in a connected vehicle environment to a BD-specific solution domain. Further, we define the security components responsible for delivering reference modules, basic requirements, and central system characteristics of an ITS-to-cloud environment. By doing so, we aim to demonstrate how to use this model in the development of effective BD security solutions and evaluation frameworks of data streaming channels.

1.5.1 Case Studies

In this section, we demonstrate the use-cases and innovative applications of our thesis, which are designed to face specific challenges when dealing with BD scenarios as follows:

1. **Data-Intensive Computing with MapReduce:**

   MapReduce (MR) has been presented as a programming model that relies on two simple primitives in functional programming languages (Map and Reduce) to simplify massive parallel data processing. In particular, Map tasks receive their input as key-value (k1, v1) pairs and return the results as intermediate key-value (k2, v2), which are grouped together by the framework and passed to a Reduce function for each key to generate the output. All input, intermediate, and returned data are maintained in file systems of different implementations (mainly HDFS). The architecture of the Apache Hadoop framework, the original implementation of the MR framework, follows the master-slave model, in which a single master is run to schedule tasks and monitor their progress. This master starts several containers on the slave nodes for executing Map or Reduce tasks (independent chunks in parallel), using large-scale committed nodes (i.e., a dedicated tightly coupled cluster) and data is centrally available (pre-placed).

In our framework, OPportunistic and Elastic Resource Allocation (OPERA), we relied on MR implementation due to its maturity as a BD model, which emphasizes the scalability of the system while maintaining high flexibility. Besides, MR provides a simple model of programming with parallel processing. It further provides dozens of both built-in and external benchmarking and stress testing performance and evaluation scenarios, e.g., TeraSort and TestDFSIO over the Hadoop cluster. In general, OPERA is a pluggable pilot-job abstraction of the Hadoop 3 framework task execution, with a dynamic resource provisioning on-demand. Its novel design supports an event-driven architecture using fine-grained resource allocation. It also enables the coexistence of the YARN cluster and HTCondor, with data nodes running inside the containers.

2. **BD Security with Federation Access Control:**

This case-study discusses one of the most significant challenges of next-generation BD federation—the new Hadoop 3.x feature, namely Hadoop access control. This feature is designed to address the continuous pressure of adding computing nodes to cope with the increasing data sets. A federation-based approach scales a single YARN cluster to tens of thousands of nodes, by federating multiple YARN sub-clusters and proposes the usage of multiple independent NameNodes (in HDFS federation), in which each NameNode is responsible for managing a subset of the whole namespace.

In this case study, we intend to endow software and system engineers with methods, best practices, and tools to systematically apply data protection principles. Hence, this brings the principles of privacy and data protection, alongside security breach detection and digital forensics investigations, into practice. However, rather than creating a set of tools from scratch, herein, we integrate privacy and data protection engineering functionalities into the existent BD security frameworks already in use by the large community behind Hadoop, thus, leveraging the efforts and ensuring a seamless adoption of its results by its intended users (systems and software engineers). More specifically, we introduce data protection functions into a set of open-source tools that are empowered by Apache open-source policies, namely, Apache Ranger and Knox.

3. **Stream Data Privacy with Internet of Vehicles:**

   Modern Intelligent Transportation System (ITS) focus on capturing the value from streaming large-scale data of different objects distributed across the edge of the network. Transferring these BDs towards the cloud has become a preferable computation model, taking advantage of its scalability and pay-as-you-go service, while maintaining beneficial data analytics. This integration of the IoT and connected vehicles is known as the Internet of Vehicles. In this context, security control and level of trust play a decisive role in the sustainable adoption of this trend.

   This thesis studies the feasibility and methodology of integrating Apache BD security frameworks and approaches of different disciplines into one coherent reference architecture. The contribution of this work is a reference architecture of state-of-the-art technologies for ITS security (called SITS). Also, in this case study, a classification of BD technologies, products, and services to address the ITS trust challenges is presented. We also propose a novel multi-tier ITS security framework for validating the usability of SITS with business intelligence development in the enterprise domain.

1.6 Thesis Structure and Outline

In the following chapters, we provide the key articles that encompass the aforementioned main contributions in the previous section and represent the main body of work for this thesis. In Section 1.8, a full compendium of the journal and conference publications related to this thesis is presented. Each section is equipped with its own background, related work, and extensive summary to cope with the vast spectrum of challenges that this thesis carries and to facilitate the reading of this dissertation

Chapter 2 continues this introductory chapter with an exhaustive state-of-art review that covers different BD paradigms from a resource management perspective. This chapter also proposes a BD taxonomy of BD execution environments with different levels of insights. This chapter also recommends a comparison criterion on how to compare and contrast the correlation/divergence of these executive environments, to delivering the Data-intensive applications of MapReduce-based services.

Chapter 3 describes a BD computational model which, based on practical experiences, provisioning on-demand technique, and the idea of elastically utilizing Docker container allow to improve the Hadoop applications performance and throughput. This chapter introduces both an Enhanced Mapreduce Environment (EME) and OPportunistic and Elastic Resource Allocation (OPERA) scheduler for the BD frameworks. Its terminology relies on a container-based cluster, enables the co-existence of Apache Hadoop and HTCondor on the same physical infrastructure. By running Hadoop YARN as HTCondor framework, so they can run side-by-side and dynamically sharing cluster resources. This enabling lightweight environment and performance isolation, fast and flexible deployment, and fine-grained resource sharing,

Chapter 4 and Chapter 5 represent two different applications developed in the context of BD security and privacy-preserving in a Hadoop federation scale. Chapter 4 proposes a trustworthy model for authenticating users and services. The main goal of this model is to provide a Single-Sign-on (SSO) approach for the latest Hadoop 3.x platform. A conceptual model is proposed combining Hadoop access control primitives and the Apache Knox framework to achieve this goal. The chapter provides various insights regarding the latest ongoing developments and open challenges in this domain. In the main time, Chapter 5 provides an in-depth study and comprehensive experiment on the performance impact and overheads associated with the proposed BD Federation Access Broker (BDFAB). Implementing the proposed framework in the BD breach and computing forensics is also discussed by employing a sophisticated audit to the access log management.

In Chapter 6 a secure channel for data streaming of IoT to the cloud case study is described. This case study focused on capturing the value from streaming data of different objects of transport and traffic management in an Intelligent Transportation System (ITS). In this regard, a reference architecture for vehicular clouds security (called SITS) is defined. Moreover, a classification of BD technologies, products, and services to address the ITS trust challenges is proposed. This chapter also introduced a novel multi-tier ITS security framework for validating the usability of SITS with business intelligence development in the enterprise domain.

Chapter 7 summarizes the main contributions of this PhD dissertation retrospectively and provides additional insights and reflections, which open several promising future trends and research lines in the context of BD deployment architectures.

1.7 Collaborations

Some of the work in this thesis was completed through collaborations with others, as well as my magnificent two doctoral advisors, José Carlos Cabaleiro and Tomás Fernández Pena. Here I attempt to attribute these contributions to the correct individuals.

1- Prof. Sherif Sakr, the Head of Data Systems Group at the Institute of Computer Science, University of Tartu, Estonia. Dr. Sherif's experiences in BD analytics, combined with his broad mind, have enabled him to become a first-class data science thinker. He has applied much of his experience to cutting-edge solutions and he has contributed numerous papers and books in this field. He provided inspiration and advice regarding the represented BD framework classifications and taxonomy in Chapter Two.

2- Assoc. Prof. Mamoun Alazab, the College of Engineering, IT & Environment, Charles Darwin University, Australia. His sharp academic mind and experiences have given him remarkable authority to deliver inventions on a platter. In particular, we worked closely to develop the Secure Intelligent Transportation System, a multi-tier architecture and the Hadoop federation access broker. Our collaboration results are represented in Chapters Four and Six.

3- Assoc. Prof. Maanak Goupta, the Department of Computer Science at Tennessee Tech University. Dr. Maanak joined the collaboration for a study of the impact of access control performance over a BD federation cluster of the Apache Hadoop 3.x platform. James Benson from the University of Texas at San Antonio supported this implementation with Dr. Maanak. Our novel implementation and practical investigation are presented in Chapter Five.

4- Assoc. Prof. Blesson Varghese Institute of Electronics, Communications & Information Technology, Queen's University Belfast, UK. Dr. Blesson has vast knowledge and experience in edge- and cloud-computing resource management, in particular, the auto-scaling mechanism of the portable containerized environment in the edge/fog models. He has contributed numerous papers and books in this field. Ongoing work in progress is the application of the thesis-proposed approaches in real-life applications.

5- Ph.D. candidate Pablo V. Caderno the University of Santiago de Compostela. Pablo and I shared an interest in the portable, auto-scale, and containerized processing environments of BD analytics. We both worked in implementing the thesis-proposed OPERA scheduler; many future collaborations are still ongoing.

1.8 List of Publications

Next a list of publications derived in peer reviewed journals and high-impact international conferences from the work developed in this thesis is shown:

Articles in international conferences:

- F. M. Awaysheh, J. C. Cabaleiro, T. F. Pena, and M. Alazab, "Poster: A pluggable authentication module for big data federation architecture," in *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*. ACM, 2019, pp. 223–225
  GGS Rating: Class A-

- F. Awaysheh, J. C. Cabaleiro, T. F. Pena, and M. Alazab, "Big data security frameworks meet the intelligent transportation systems trust challenges," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 807–813
  GGS Rating: Class B

- F. M. Awaysheh, T. F. Pena, and J. C. Cabaleiro, "EME: An automated, elastic and efficient prototype for provisioning hadoop clusters on-demand," in *The 7th International Conference on Cloud Computing and Services Science, CLOSER*, 2017, pp. 709–714
  GGS Rating: Work in Progress

- F. M. Awaysheh, J. C. Cabaleiro, and T. Pena, "OPERA-P: an adaptive scheduler for dynamically provissioning big data frameworks on-demand," in *The third international workshop HTCondor at DESY Hamburg*. HTCondor, 2017

In-press Journal articles:

- Feras M. Awaysheh, Mamoun Alazab, Maanak Guptab, Tomás F. Pena, José C. Cabaleiro "Next Generation Big Data Federation Access Control: A Reference Model" *Future Generation Computer Systems*, ISSN: 0167-739X
  Impact factor (JCR 2018): 5.768. Q1.
  Category: COMPUTER SCIENCE, THEORY & METHODS. Rank: **6/108**. Publisher: ELSEVIER SCIENCE BV.

- Mohammad N. Aladwan, Feras M. Awaysheh, Sadi Alawadi, Mamoun Alazab, Tomás F. Pena, José C. Cabaleiro "TrustE-VC: Trustworthy Evaluation Framework for Industrial Connected Vehicles in the Cloud " *IEEE Transactions on Industrial Informatics*, ISSN: 1551-3203
  Impact factor (JCR 2018): 7.377. Q1. Both first authors contributed equally to this publication.
  Category: AUTOMATION & CONTROL SYSTEMS - SCIE; COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS. Rank: **3/62**. Publisher: IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC.

Under Review Journal articles:

- Feras M. Awaysheh, Tomás F. Pena, José C. Cabaleiro " Security by Design for Big Data Frameworks over Cloud Computing " *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, ISSN: 0018-9391
  Impact factor (JCR 2018): 1.867. Q3.
  Category: ENGINEERING, INDUSTRIAL - SCIE. Rank: **24/46**.

CHAPTER 2

BIG DATA: RESOURCE MANAGEMENT AND DEPLOYMENT ARCHITECTURES

The MapReduce framework has pioneered the new generation of BD processing frameworks. It has been presented as a parallel data processing framework, designed to handle the workloads of Data-Intensive Applications (DIA). Apache Hadoop, the prominent MapReduce implementation, has been originally designed to run on dedicated, homogeneous local-based clusters. In the last years, several deployment environments have been introduced and they become available with various architectures and configurations.

This chapter presents a comprehensive survey, along with a novel taxonomy, for the architectural elements of different deployments of BD processing frameworks. In addition, we discuss in detail and compare these various deployment paradigms.

2.1 Introduction

BD programming models like MapReduce (MR) were initially introduced by Google [5] for processing large data sets as web search. It has emerged as a significant and widely used framework for processing massive volumes of data in parallel and distributed systems, due to its generality, ease of use, scalability, fault tolerance, and flexibility to handle tasks in different environments. The most well-known open-source implementation of the MapReduce framework is Apache Hadoop [6]. In principle, Hadoop is designed to utilize clusters of commodity machines. Thus, many organizations have built their dedicated Hadoop cluster to handle their

BD operations (storage and analysis). Over the years, several approaches have been introduced to improve the efficiency and flexibility of the MR framework. These approaches have been focusing on three main aspects. First, optimizing the scheduling policies and implementing different scheduling techniques, to meet diverse data-intensive MR application requirements. This includes a large body of research, as for, pipelining [7], join processing and handling data skew [8], multi-threading systems [9, 10], iterative operations [11] and make span minimization for both offline and online scheduling [12]. Second, scaling the system resource capability (i.e., scaling horizontally and vertically), and enhancing data distribution mechanisms, like `Gfarm` [13]. Third, searching for new resource providers by adopting specialized hardware accelerators [14] or trying new environments, which represent the research scope of this study.

In this survey, we focus on the architectural elements of the BD processing frameworks using different deployment environments and discuss their salient features by focusing on the resources aspects. We present a taxonomy of the implementation requirements of BD processing frameworks which covers the different optimization objectives using a multidimensional classification approach. These dimensions represent the basics on which the performance of Data Intensive Application (DIA) systems heavily rely, namely: (a) resources, (b) security, (c) tasks, and (d) data management. Based on the proposed taxonomy, we discuss the open issues and challenges raised in data analysis using BD processing frameworks in different non-traditional environments and outline the corresponding solutions in the literature with new directions for future enhancements. In addition, we discuss how these systems differ by using two use cases based on the resource location. In particular, we present a classification of the existing implementations using LAN-based and WAN-based environments, focusing on delivering high Quality of Services (QoS) in any environment.

The remainder of this section is organized as follows. Section 2.2 provides the background of this article. Section 2.3 presents a taxonomy of MR operational environments, from both traditional and non-traditional configurations. A discussion on the convergence of the different deployment architecture is presented in Section 2.4. Section 2.5 highlights the emerging BD architectures and discusses various challenges and future research directions that need to be addressed within the next-generation BD platforms. We finally conclude in Section 2.6.

2.2 Background

In this section, we present a brief overview of the MapReduce framework and highlight the novelty of our survey with respect to other related surveys.

2.2.1 MapReduce framework

MR has been presented as a programming model that relies on two simple primitives in functional programming languages (Map and Reduce) to simplify massive parallel data processing. In particular, `Map` tasks receive their input as key-value (k_1, v_1) pairs and return the results as intermediate key-value (k_2, v_2) that are grouped together by the framework and passed to a `Reduce` function for each key to generate the output. All input, intermediate, and returned data are maintained in file systems of different implementations. The architecture of the Apache Hadoop framework, the original implementation of MR framework, follows the master-slave model, where a single master is run to schedule tasks and to monitor their progress. This master starts several containers on the slave nodes for executing the map or reduce tasks (independent chunks in parallel), using large-scale committed nodes (i.e. dedicated tightly coupled cluster) and data is centrally available (pre-placed).

Figure 3.4 illustrates the different layers and architectural design of some paradigms that implement MR services. It also highlights some of the primary abstractions compared to the native Hadoop deployment architecture. In particular, the Hadoop framework consists of three main components [15]: (i) The Hadoop Distributed File System (HDFS) [16] that handle data management and access, using a `NameNode` process running on the master and multiple DataNode processes running on the worker. HDFS uses data replication to achieve high data availability and reliability. (ii) Apache YARN [17], which is a cluster resource-management platform that is responsible for scheduling user's application tasks, implementing security controls and managing the computing resources, besides providing the high availability features of Hadoop. This new architecture removes the old `JobTracker` (in Hadoop v 1) from managing resource usage across applications, which removes the single point of failure and enhances the scalability, thus allowing it to run other applications apart from MR ones. YARN uses a global Resource Manager to receive and run applications on the slaves, and a Job History Server to provide information about completed jobs. On the slave side, the `NodeManagers` (used to be known as Task Tracker in Hadoop v1) provides information about the node status and capacity. (iii) An execution framework (e.g.,

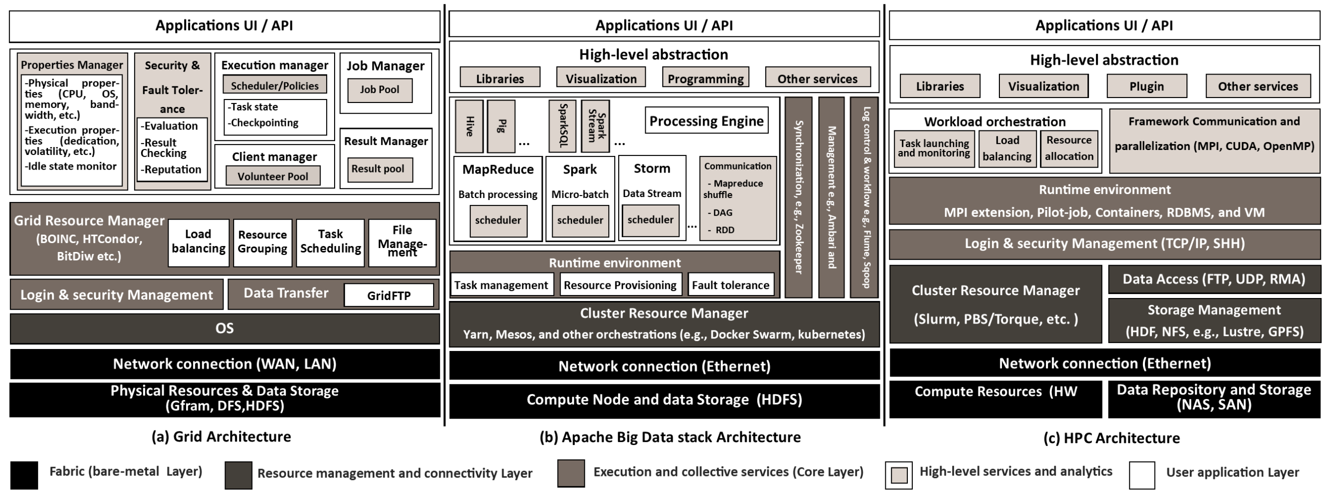

Applications UI / API
Properties Manager
-Physical properties (CPU, OS, memory, bandwidth, etc.)
-Execution properties (dedication, volatility, etc.)
-Idle state monitor
Security & Fault Tolerance
-Evaluation
-Result Checking
-Reputation
Execution manager
Scheduler/Policies
-Task state
-Checkpointing
Client manager
Volunteer Pool
Job Manager
Job Pool
Result Manager
Result pool
Grid Resource Manager (BOINC, HTCondor, BitDiw etc.)
Load balancing
Resource Grouping
Task Scheduling
File Management
Login & security Management
Data Transfer
GridFTP
OS
Network connection (WAN, LAN)
Physical Resources & Data Storage (Gfram, DFS,HDFS)
(a) Grid Architecture
Applications UI / API
High-level abstraction
Libraries
Visualization
Programming
Other services
Hive
Pig
SparkSQL
Spark Stream
Processing Engine
MapReduce
Batch processing
scheduler
Spark
Micro-batch
scheduler
Storm
Data Stream
scheduler
Communication
- Mapreduce shuffle
- DAG
- RDD
Synchronization, e.g., Zookeeper
Management e.g., Ambari and
Log control & workflow e.g., Flume, Sqoop
Runtime environment
Task management
Resource Provisioning
Fault tolerance
Cluster Resource Manager
Yarn, Mesos, and other orchestrations (e.g., Docker Swarm, kubernetes)
Network connection (Ethernet)
Compute Node and data Storage (HDFS)
(b) Apache Big Data stack Architecture
Applications UI / API
High-level abstraction
Libraries
Visualization
Plugin
Other services
Workload orchestration
Task launching and monitoring
Load balancing
Resource allocation
Framework Communication and parallelization (MPI, CUDA, OpenMP)
Runtime environment
MPI extension, Pilot-job, Containers, RDBMS, and VM
Login & security Management (TCP/IP, SHH)
Cluster Resource Manager (Slurm, PBS/Torque, etc.)
Data Access (FTP, UDP, RMA)
Storage Management (HDF, NFS, e.g., Lustre, GPFS)
Network connection (Ethernet)
Compute Resources (HW
Data Repository and Storage (NAS, SAN)
(c) HPC Architecture
Fabric (bare-metal Layer)
Resource management and connectivity Layer
Execution and collective services (Core Layer)
High-level services and analytics
User application Layer

Figure 2.1: Desktop Grid, HPC, and Hadoop architecture and abstractions: Each approach has different resource management and communication capabilities with hitherto distinct resource type.

MR) that runs a per application `ApplicationMaster`, to manage the different jobs. The `ApplicationMaster` asks for resources to the `ResourceManager`, which, with the information provided by the `NodeManagers`, provides the resources in the form of containers. The `ApplicationMaster` runs the application in those containers, controlling its behavior and re-launching failed tasks as needed.

2.2.2 Related work

There is an extensive discussion in the literature of BD processing frameworks. For instance, [18, 19] reviewed the state of the art for big data processing systems. Other studies [20, 21] analyzed the systems from perspective of the performance requirements for deploying data-intensive applications. Khezr et al. [22] discussed the differences among various implementations of MR in cloud environments. Moreover, Li et al. [23] illustrated the differences between MRv1 and MRv2 and addressed the Hadoop 2.0 improvements.

This survey distinguishes itself by providing a comprehensive and thorough coverage for the prominent BD processing paradigms from the resource management perspective. We cover various configurations including a variety of items and techniques to compare, due to the massive difference in the resources type, e.g., dedication, heterogeneity, cost, connectivity, and scalability that has been adapted to handle specific data analytic verticals. To the best of our knowledge, no other study has considered this aspect at the same level of detail.

2.3 Taxonomy of MapReduce environments

In this section, we present a taxonomy of MR implementation environments (MRIE) based on the computing resource types. These deployment environments are from both traditional and non-traditional configurations and paradigms that aims to deliver MR-based services and deal with the challenges of volume, velocity, variety, and veracity inherent in DIA.

Figure 3.6 presents our taxonomy by organizing Big Data processing frameworks into six main categories. These categories are, further, defined and described in Table 1. In our taxonomy, we divide the cluster computing into bare-metal (physical) machines, which consist of either CPU or GPU clusters, and virtual machines (VMs), which in turn are divided into hypervisor-based and container based clusters (CBC). In addition, we categorize the cloud computing implementations separately from the local virtual clusters, even though they rely on the same technology (virtualization). However, by separating the local-based virtual clus-

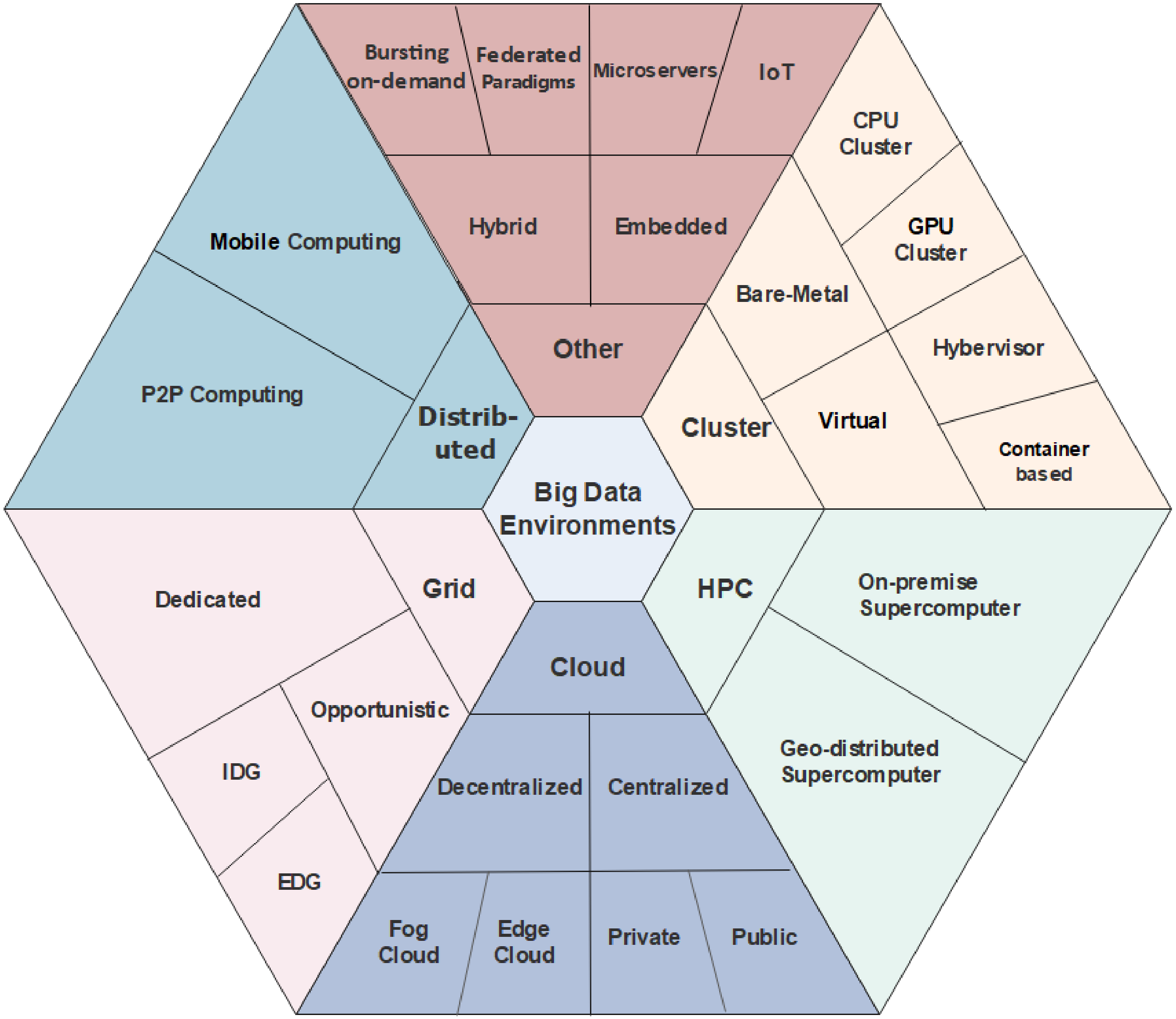


Figure 2.2: Taxonomy of MapReduce Implementation Environments (MRIE).

Table 2.1: Defining various categories of big data taxonomy.

Cluster Computing	A set of a commodity computing nodes connected through fast local area networks as a single system.
Cloud Computing	A shared pool of computing resources which enable remote access and are based on virtualization technologies.
Supercomputing and HPC	A set of high-end computing nodes connected through high-speed local or wide area networks to achieve massive computing power.
Grid Computing	A pool of computing resources from various locations working together to achieve a shared goal, and with non-interactive workloads.
Decentralized Computing	A distributed architecture where each machine is equally capable of servicing requests without the need for central coordination.
Hybrid Computing	An amalgamation of two or more paradigms to exploit the benefits of those paradigms with the aim of better performance or to lower the total cost.

ters from the distance virtual clusters (Cloud-based), we can differentiate the technologies and resources that were used, i.e., computer networks, servers, storage, applications, and services. Thus, we are able to provide the agility required to analyze all MR implementations in this prominent model comprehensively. Besides, grid computing consists of both dedicated and non-dedicated (opportunistic resources) infrastructures. Further, the opportunistic solutions classified into two main categories, namely, Volunteer Computing (VC), also knows as Internet Desktop Grids (IDG), and Enterprise Desktop Grids (EDG). High Performance Computers (HPC) based deployments have been classified, depending on its resource locality, in local and distributed data centers. Meanwhile, decentralized approaches that have been presented in the literature can be categorized into two main pillars: Peer-to-Peer (P2P) computing and mobile computing-based architectures. Finally, we also consider hybrid approaches, which combine different environments in the same deployment, e.g., coupling CPU and GPU clusters.

2.3.1 HPC and Supercomputing

The HPC paradigm is considered an essential environment for scientific Computing Intensive Applications (CIA) and massive-scale distributed tasks systems [24]. Hadoop and HPC paradigms shares some features and components in their design (see Figure 3.4) and might seem similar. However, they are adapted to solve very different problems and fulfill different workloads. Further, the complexity and characteristics of Hadoop framework are reasonably distinct from HPC applications as well. Consequently, both paradigms are architectured differently. In particular, supercomputing platforms typically support separate globally shared file systems such as NFS, SAN or Lustre[1], and do not provide on-node persistent storage. While, on the other hand, Hadoop employs its own distributed file system and processes tasks locally to avoid contention for shared storage resources. HPC uses batch queue systems for the fair sharing of batch resources among different users with parallel distributed file system combined with high-performance interconnection for shared storage. Also, they provide low-level programming APIs that can achieve much higher efficiency. However, they lack fault-tolerant storage support for large-scale data sets. Hence, using HPC resources directly for the benefits of big-data applications will result in certain performance degradation.

In principle, Hadoop has been mainly designed to solve DIA problems, with many loosely coupled tasks, unlike the tightly coupled (usually dependent) tasks that are natural of the supercomputers. In particular, Hadoop brings computation power to data (to minimize data transfer and improve data locality), running the HDFS daemons locally within a significant amount of disk-based space on each node. Thus, data in HDFS is physically distributed and replicated throughout the cluster. In contrast, many HPC clusters rely on diskless nodes and use a separate group of hardware as a data storage to provide file-level access. In its core, Hadoop presents a higher level of abstraction in terms of data management which limits the high latency in data transfer. In contrast to HPC, the task management in Hadoop consists of a single master per application, `ApplicationMaster` (AM), which communicates with many `NodeManagers` on the cluster nodes. The AM supervises all MR workloads. Thus, Hadoop manages failure more efficiently, as if an HDFS node fails, the AM can reassign the task to another node with a redundant copy of those data. Similarly, the failed tasks (Map or Reduce process) may restart on a new node, using task reassignment and resource recovery techniques.

In general, the scale-up capabilities provided by supercomputing platforms attracted atten-

[1]http://lustre.org/

tion to optimize the performance and keep the turnaround time to a minimum. This vertical scaling capabilities for computational power, also, enables using the already installed HPC infrastructures, rather than creating a new and costly Hadoop cluster. Several attempts and techniques have been proposed to merge the technical gap and cope with the inherited limitations between the broader HPC infrastructure and the MR framework. These techniques can be categorized into three main approaches:

1. Running DIA processing frameworks over HPC environments to support applications with both compute and data requirements and enhance supercomputers utilization.
2. Implementing MR extensions using the Message-Passing Interface (MPI) with interfaces and libraries to support key/value communication and in-situ data processing on supercomputing systems.
3. Besides the previous two on-premise categories, utilizing geo-distributed supercomputers was proposed as well.

In this section, we analyze the characteristics and design decisions of deploying big data processing frameworks in HPC environment (see Table 2).

MR over traditional HPC

`myHadoop` [25] has been presented as an MR distribution in traditional HPC resources that allows configuring Hadoop, on-demand, without the need for root-level access to leverage the framework. It is a lightweight distribution. However, it showed much slower performance in executing MR tasks in comparison to the native Hadoop implementation. An extension of this framework has been adopted in `HPCHadoop` [26] that has been designed to run Hadoop on Cray X-series supercomputers. Another example is `MARIANE` [27] that offers ease of programming, synchronization and parallelization abstractions but does not support shared-disk file systems. Supercomputers had also harnessed many-core processors to scale their computing capacities vertically [28]. Sehrish et al. [29] attempted to address the gap among HPC data storage systems and data-intensive systems by implementing a data-centric scheduler that maintains data locality using the integration of data access semantics with the MR programming model. `PortHadoop` [30] proposed a direct data fetching between Hadoop framework and supercomputers via a parallel file system, and `Mrap` [31] extends the MapReduce framework to reduce the number of the pre-processing tasks. Apache Slider [32] deploys long-running distributed applications on a YARN cluster with elastic resource allocation on-demand. `SAGA-Hadoop` [33] has been designed to provide a uniform framework for managing and running a full Hadoop cluster in user space, on top of existing HPC resource management systems (e.g., Fork, SGE, and PBS/Torque).

MR extension to MPI

This approach aims to accommodate fast communication infrastructure and standard interface to facilitate the development of parallel and distributed application. Compared with Hadoop communication primitives, the performance of MPI communication primitives is high [34]. In particular, MR shuffles mechanisms considered as a system bottleneck from the basic communication level of remote interaction. Consequently, several approaches have been proposed relying on MPI techniques for implementing MR frameworks. For example, `SMART` [35] has been designed to process Terabyte-scale data sets on traditional MPI-based supercomputers using a parallel library that contains a lightweight implementation of MR functionality. `Mimir` [36] has been presented as a memory-efficient MR library for supercomputing systems over MPI. Guo et al. [37] presented a study that supports fault tolerance over large HPC clusters with failure detection and recovery plans. Lin et al. [38] presented an approach for integrating MPI into the current BD stack with the aim of addressing the metagenome reads clustering challenge, which holds both data and compute-intensive phases.

Distributed HPC

`G-Hadoop` [39] represents a prominent example that has been designed to support large-scale geographically distributed computing across different clusters. Similarly, Wang and Li. [40] investigated the scheduling of MR jobs over geo-distributed datacenters on heterogeneous networks with an adaptive heartbeat. `GreenMap` [41] introduced an energy-efficient embedding method as a use-case for MR-based virtual networks over heterogeneous datacenter networks. `ChEsS` [42] presented an approach for a dynamic jobs assignment on heterogeneous clusters by exploiting historical data for predicting the processing times with emphasis on data locality restrictions and the cluster processing capacities. Employing MR model for loosely coupled applications (compute-intensive) over distributed cyber-infrastructures using replica exchange was proposed in [43] to provide task-level parallelization and scalability. Further, spawning pilot containers to manage Hadoop MR/Spark clusters on HPC infrastructures on demand was manifested in [44]. In contrast, utilizing parallel storage system (e.g., Lustre)

Table 2.2: Comparison of key attributes of HPC Models

HPC Model	**Resource Allocation**	**Resource Scalability**	**Resource Support**	**Process Migration**	**Security and Fault-Tolerance**	**Overhead Complexity**	**System Functionality**
Traditional HPC	Centralized	Low	Homogeneous	No	High	Low	Support high-level abstractions of MR application
MPI extension	Centralized	Low	Homogeneous	No	High	Medium	Provides basic services for modular deployment of MR application
Distributed HPC	Decentralized	Higher	Heterogeneous	Yes	lower	High	Resources are allocate on-demand and supply on Pre-located bases

for intermediate data has been examined in [45, 46]. In addition, Veiga et al. [47] presented an in-depth analysis and assessment of various HPC-oriented MR platforms in terms of their performance and the efficiency of energy consumption.

2.3.2 Cloud Computing

Cloud computing has become a powerful computing service-provider that shares with both utility computing and grid computing the use of shared virtual machines to deliver computing as a utility and hold large-scale complex computing on-demand [48]. Traditionally, BD processing systems use a shared-nothing architecture, thus, it is considered as costly as it needs to acquire a dedicated physical Hadoop cluster with high installation and maintenance cost. The advent of cloud-based clusters provided an effective solution that not only tackles this challenge but also, enhances the system scalability, provides reduced maintenance cost and increases the efficiency of resource management. Hence, developers of BD applications became among the first adopters of cloud environments for deploying their systems in an elastic and on-demand environment. Over the years, cloud service providers have been offering a wide range of DIA supporting services from storage to process and analyze vast amounts of datasets. Examples include public service providers (e.g., Amazon EMR [49], Microsoft Azure HDInsight [50], or Google Cloud Dataproc [51]) and private vendors (e.g., Cloudera, Hortonworks, or MapR). Table 3 summarizes the reasons that grab much of the attention to

Table 2.3: MapReduce on Cloud computing

MR System requirements	**Cloud computing features**						
	Verity of capacities	**On-demand resources**	**Data/compute-intensive**	**Elastic provisioning**	**Pay-per-use**	**Trustworthy and recovery**	**Auto management**
Load Balancing	✓		✓	✓			
Task Scheduling		✓	✓	✓		✓	
Data Placement	✓	✓		✓			
Fault tolerance						✓	✓
Cost-benefits	✓	✓		✓	✓		✓
Security and Reliability						✓	✓

cloud capabilities and their linkage with the requirements of the MR environments.

In principle, the underlying motivation and advantages of designing and implementing cloud solutions that support BD operations within heterogeneous/homogeneous clusters on-demand lies in the following four criteria:

- *Flexibility*: This includes the flexibility to use the required instance type (high, medium or low capacity) on a scale method (elastic provision to scale up/down). This also includes the capability of handling Petabytes of data on thousands of instances.

- *Usability*: This is related to the ease of launching a cluster and deleting it after finishing the job. The system should be developed as a service-oriented architecture with the aim of allowing users to combine and reuse them on-demand.

- *Cost efficiency*: By eliminating the need for capital expenditure on large private infrastructures and following the pay only for the computational power or data space they use (pay-per-use).

- *Trust*: Clouds are considered as reliable environments, using job recovery techniques and data/task replications (upon policy) to enhance fault tolerance approaches and backup features. Many computing nodes are available in the cluster in failure mode.

In practice, adopting cloud environment does come with its associated new challenges. The main issue concerning BD is related to the high latency of data transfer in/out the cloud environment. Nowadays, data are collected from various sources that do not necessarily available a structured form. Preparing and transforming unstructured data is a challenging step, yet required before storing them in the data warehouse for analysis. On the other hand, the scalable nature of data processing requires efficient and dynamic scheduling algorithms. Furthermore, due to the shared infrastructure in cloud environments, it inherits some existing issues from virtualization technology (see section 2.3.4). Besides, data security on such configurations would be considered as the most challenging affairs. Security threats such as privacy, integrity, confidentiality, and availability of stored data are magnified by the properties of BD (i.e., volume, velocity, and variety). Therefore, cloud vendors must assess all service levels against threats at frequent intervals, even though, it is believed that, among other paradigms, Cloud technology has the highest potentials and is the preferable trend for Data-rich enterprises and communities. Hashem et al. [52] discussed the relationship between the cloud architectures and big data processing in general, besides the current challenges and open issues of this adoption.

Task scheduling in Centralized Cloud

Motivated by the prominent advantages of cloud computing architectures, many small-medium enterprises, and scientific communities chose it to deploy MR solutions on highly configurable and inexpensive resources. For example, `Kahuna` [53] presented an approach for addressing the performance problems of MR cloud-based approaches. It has been implemented on top of Yahoo's M45 Hadoop cluster to simulate a cloud environment under fault-free conditions. Gunarathne et al. [54] presented the case of creating the Azure MapReduce service (using Microsoft Azure HDInsight). To minimize the incurred cost of cloud resource usage, `AROMA` [55] introduced a two-phase machine learning and optimization framework that enables an automated configuration and allocation of heterogeneous instances. `Cloud-MapReduce` [56] implemented the MR programming model over the Amazon cloud OS. `Twister4Azure` [57] presented a distributed decentralized iterative MR runtime that extends MR to Azure Cloud and enables a fault-tolerance execution of broad data intensive applications.

To capitalize on the pay-per-use in the Cloud, a cost-optimized model, `Cura` [58], has been proposed to solve the poor resource utilization and higher cost caused by the complexity of clusters configuration on the cloud. It runs multi-stage interactive jobs workflows and, in general, achieves this in a cost-effective fashion, with lower job response times. Rao and

Reddy [59] have considered various possible scheduler improvements to Hadoop in a Cloud environment. They investigated this issue using a prominent schedules, `DELAY` [60]. Correspondingly, the resource allocation for MR in the cloud was the main focus of `Purlieus` [61]. In particular, this work aims for enhancing the performance of MR jobs using placement methods by optimizing the data placement techniques through both map and reduce phases. This increases locality, processing the intermediate data locally or by machines that are physically close-by. `CMR` [62] suggested two optimizations to their cloud architecture to tackle the batch data processing and spot instances on Amazon EC2, which are called continuous cloud MR and spot cloud MR, respectively. Aiming to alleviate the bottleneck caused by data transfer over WAN network in public clouds, `Ehadoop` [63] proposed a reduced online job profiling for EMR clusters. Motivated by decreasing the disk usage, an event-driven Hadoop MR architecture, `Flame-MR` [64], was proposed and evaluated for both public cloud and HPC clusters.

Cloud computing naturally fits to the requirement of processing large amounts of data in parallel. Studying the impact of virtual network topologies on the MR cloud-based frameworks performance and optimizing these topologies was the main focus of [65] which proposed a mechanism to identify the optimal virtual network topology according to formulating the data processing and data transmission overheads of the MR cluster. `SAMR` [66] introduced a scheduling policy that can precisely identify the straggler tasks and boot their execution. It identifies the straggler tasks by monitoring the progress of all the active tasks and dynamically tunning the weight of each phase based on historical statistics. Ghoshal et al. [67] demonstrated the effectiveness of pipeline provisioning with data placement and partitioning strategies over clouds. The proposed set of strategies does not only improve scalability and performance, but reliability as well.

Additionally, cloud models have attracted serious attention from broad application platforms. For example, `MaMR` [68] proposed adding a new merge phase that can efficiently merge data from the map and reduce modules. It uses an optimized data sharing strategy that employs a hybrid shared-memory and parallel bridges in the programming language. Additionally, other approaches considered accelerating the execution of short jobs [69, 70], hash-partitioning the resource manager [71], and improving the data locality rate [72, 73, 74].

Decentralized Cloud

The success in adopting new paradigms as distributed data-intensive environments have triggered the investigation toward adapting and optimizing more complex computing frameworks in many aspects. Many researchers have proposed and investigate decentralized cloud platforms (e.g., multicloud, edge, and fog computing), as a new distributed MR architecture. This new trend aimed to minimize the data transmission overhead between ubiquitous data-rich sources and cloud datacenters, by deploying computation closer to the edge of the network. Also, it addresses the time-constrained issue of data analytics cloud applications, which is considered as a pressing matter. In addition, the advent of many Internet of Things (IoT) applications, producing large-scale data from a massive number of geo-distributed objects, will imply the need to efficiently store, process, and analyze these data. Therefore fog/edge computing help in fulfilling the computation gap by extending cloud computing to be closer to the things it supports.

Ad hoc data processing, on the contrary, has shown to be a critical aspect for various cloud-based applications [75]. Shi et al. [76] proposed an optimized declarative language for relational databases that targeted the ad hoc data processing. In addition, exploiting adhoc clouds for MR operations had been analyzed in [77], using availability-aware scheduling that dynamically re-adapts task assignments on-demand. A recent study on cloud computing directions [78] predicted further approximation of cloud infrastructure toward decentralized approaches. Thus, new opportunities are found that facilitate processing data-intensive workloads before storing it, closer to where they are generated. `Medusa` [79] has been presented as an approach for scaling MR applications among multiple distributed clouds and maintaining fault tolerance. It uses a proxy for replicating each job in more than one cloud and compares the outputs in two aggregation phases.

Furthermore, splitting the data set and distributing them among multiple federated clouds (geo-distributed clouds) has been labeled as an acceptable scale-out model using external resources and storage on the cloud. This model has emerged due to the growing demands of outsourcing computation to fulfill several users requests, increase the system efficiency and reduce costs. For instance, `FedSCD` [80] utilized idle VMs across cloud federation with a decentralized scheduling algorithm to minimize the cost of MR execution efficiently. `Fed-MR` [81] was designed to enable running MR across geo-distributed clusters by employing two additional phases, proxy-map, and proxy-reduce. Other approaches, like [82, 83], were designed to take into account the nodes heterogeneity in a hierarchical computing model.

`Resilin` [84] allows users to perform MR computations across a broad range of cloud service providers and automate the cluster provisioning on-demand to the best available resources. A comparison of multi-datacenter with single-datacenter Hadoop deployments in a geo-distributed clouds has been presented in [85].

2.3.3 Grid Computing

The term Grid Computing has initially been used to describe technologies that allowed consumers to obtain computing power on demand across multiple administrative domains [86]. Correspondingly, Desktop Grids (DGs, also known as Opportunistic Computing, OC) are considered as a particular case of the Grid. DGs are subdivided into two principal categories, based on resource location:

- The *Internet-based* solutions that are mainly based on volunteer resources participants (e.g. `BOINC` [87], `BitDew` [88]).
- The *Intranet-based* solutions, in which, an organization uses its existing infrastructure (e.g., clusters or desktop machines) to execute its long-running computing tasks. it is distinguished in that it is more trusted than volunteer computing and its deployment is typically automated. Because of technical similarities, it is often known as a cycle-scavenging or cycle stealing technique (e.g. `HTCondor` [89]).

In practice, grid computing capacities are designed to support application-oriented services, in the form of virtual organization environment. Depending on how to employ these capacities, we can subdivide them into distributed supercomputing and DG. Therefore, grid systems can be placed into two categories (see Figure 3.6), as dedicated and opportunistic (or non-dedicated) computing. The opportunistic are, also, categorized as Internet Desktop Grids (IDGs, aka Volunteer Computing VC), and Enterprise Desktop Grids (EDGs). According to Lu et al. [98], adapting this model to the desktop grid enables taking benefit from the large-scale amount of computing resources and distributed storage to perform a new set of applications that can handle massive sizes of data. Table 4 presents a comparison of different MR implementations in the Grid environment. In particular, the advantages of using MR in the Grid can be summarized as follow:

1. Reducing the resources' Total Cost of Ownership (TCO).
2. The adequacy for running embarrassingly parallel workloads (like MR tasks).
3. Increasing the ability to scale the computational capacity as needed.

Table 2.4: A Survey of MR distributions in Grid environment.

Study or Authors Name	Heterogeneity/Dedication	Field of Contribution	Advantages	Limitations
K Lee [90]	Heterogeneous / Opportunistic	Data Allocation based on session up time of the opportunistic resources	-Reduction the service disturbance in Opportunistic environment using the resource session uptime not availability rate - Decrease the number of interrupted tasks	- Single job (Workload) - Doesn't support back up for the Sudden failure -In large VC resource pool it has limitation in monitoring/managing the resources
Yuting Ji [91]	Homogeneous / Opportunistic	-Data Allocation -Task Scheduling	-Fairness scheduling and Multi-job -No third part needed to manage resources	- Do not take into account data locality and data placement -Not applicable for Heterogeneous participation
ADAPT [92]	Heterogeneous / Opportunistic	Data Placement	-MR with low data replication - improve the data locality and reduce the network traffic - End-to-end application performance	- Focused on single job (Workload) -Doesn't support task allocation to improve QoS
Corona [93]	Homogeneous / Dedicated	Data Allocation	- Better scalability and cluster utilization - Lower latency for small jobs - Ability to upgrade without disruption	- Do not take into account data locality and machine capability - Doesn't support undedicated machines
HOD [94]	Heterogeneous / Dedicated	Data Allocation	-Create a temporary Hadoop platform in the Grid - Auto-generates the appropriate configuration files for clients - Has the capability to distribute Hadoop to the nodes in the virtual cluster	- High reconstruction overhead, fixed node number, and randomly chosen head node - Need a 3rd part to do node allocation (Torque resource manager)
HOG [95]	Heterogeneous / Opportunistic	-Data Allocation -Data Placement	- Has a scalable size, and has a static dedicated head Node - All data stored in the grid resources and replication schemes for data availability	- Common lose nodes, and need to rebalance data frequently - Uses plain HTTP for communication between nodes
G-Hadoop [39]	Homogeneous / Dedicated	-Data Allocation -Data Placement	- An improved fault-tolerant data processing environment - Uses Gfram instead to HDFS	- Target multiple distributed High End Computing (HEC) only - Relies on a centralized high performance networks with high bandwidth -Need 3rd part to manage resource (Torque)
MOON [96]	Homogeneous / Hybrid	-Data Allocation -Data Placement	- Replica on dedicated nodes can significantly enhance availability with low cost - long running tasks are executed on dedicated nodes -High QoS on Opportunistic resources using speculative task scheduling	-3rd part needed to manage resource availability -Focused on single job (Workload) - Homogeneous environment -Doesn't demonstrate to work on Internet Desktop Grid
MRA++ [97]	Heterogeneous / Dedicated	-Data Allocation	-Can be implemented in a low network bandwidth (such as Internet) -Different scheduling upon capability -Avoid unworthy task to be remote -Considers the heterogeneity of nodes during data distribution and task scheduling to gain a good balance of tasks	-Not applicable for opportunistic environments -Ignored volatility of Data -Single job scheduling (Workload) -Do not take into account data placement (locality)

4. Gaining better Return on Investment (ROI) in the case of EDG.

5. Improving the ability to handle failures and reduce system management costs.

Several approaches have been focusing on running DIA (MR-based) on Grid and DG. For instance, Lee and Figueiredo [90] suggested a data allocation scheme, based on session uptime of the opportunistic resources, by selecting a subset of the available workers in a volunteer pool of resources that is predicted to be available in the future. A similar approach is followed by `ADAPT` [92] which exploits the availability history to assign an estimated availability rate for each new participant. Then, this rate is used to predict the node future availability and data is sent based on the participant availability. In contrast, Ji et al. [91] proposed a new scheduler, called Opportunistic Fair Scheduler (OFS), that extends the Hadoop Fair Scheduler (HFS) [99]. This approach improves fairness and performance among multiple concurrent jobs in an opportunistic environment by learning node availability statistics and interruption handling within Hadoop. However, the free-to-join mechanism that is natural in the VM environment, causes a significant challenge regarding security to verify that malicious volunteers do not intervene the results of computations. Therefore, a model that characterizes the error rate of a MR application using the majority voting method was proposed by Moca et al. [100]. The proposed system relies on the `BitDew-MR` middleware [101, 102]. Besides, it intends to be appropriate for large scale and loosely coupled IDG. It had further been evaluated and compared with Hadoop [98].

In general, the Grid demands resource-aware schedulers (e.g. Fair Scheduler or Capacity Scheduler) for enhancing its performance. `MRA++` [97] proposed an implementation with low bandwidth networks (such as the Internet) and considers the heterogeneity of nodes during task scheduling and data distribution to gain a fair balance of tasks. `MOON` [96, 103] has been presented as a hybrid configuration in which volunteer computing (non-dedicated) platforms are augmented with a set of dedicated workers to reduce the implementation cost. The main focus of this approach is to allocate a small set of dedicated nodes to volunteer computing systems. `MOON` adopts the LATE algorithm, a scheduler by Zaharia et al [104]. On the contrary, `VMR` [105] proposed an Internet volunteer computing system to execute MR applications as independent tasks. Another example for implementing MR programming over desktop grid distributed on the Internet was proposed by Google [106].

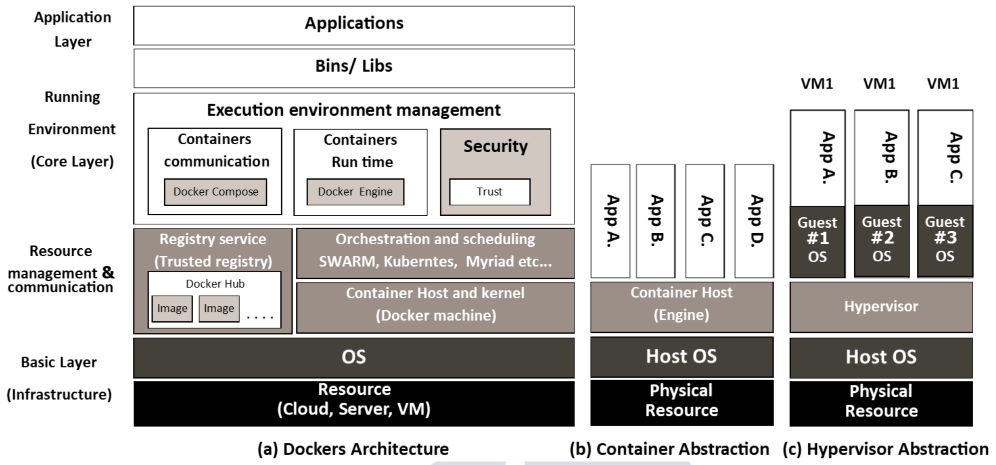

Application Layer
Running Environment (Core Layer)
Resource management & communication
Basic Layer (Infrastructure)
Applications
Bins/ Libs
Execution environment management
Containers communication
Docker Compose
Containers Run time
Docker Engine
Security
Trust
Registry service (Trusted registry)
Docker Hub
Image
Image
Orchestration and scheduling SWARM, Kubernetes, Myriad etc...
Container Host and kernel (Docker machine)
OS
Resource (Cloud, Server, VM)
App A.
App B.
App C.
App D.
Container Host (Engine)
Host OS
Physical Resource
VM1
VM1
VM1
App A.
App B.
App C.
Guest #1 OS
Guest #2 OS
Guest #3 OS
Hypervisor
Host OS
Physical Resource
(a) Dockers Architecture
(b) Container Abstraction
(c) Hypervisor Abstraction

Figure 2.3: Docker layered architecture and hypervisor vs. container virtualization abstraction.

2.3.4 Virtualization Technology

The main focus of virtualization technology (VT) is to emulate the behavior of many execution environments into the same physical implementation using a hypervisor (e.g., VMware, KVM or Xen) [107]. Once a physical machine is virtualized, higher level services can be deployed as a Hadoop cluster. Virtualization techniques provide many advantages including: 1) Reducing operational costs (a single image of a system can be cloned). 2) Allowing the user to set up resources that are elastically sized (expanded or contracted) on-demand. 3) Providing economic cost-benefits by saving power, getting a better return on investments (i.e., physical infrastructure can reuse) and reducing capital expenses on infrastructure (especially in testing cases). 4) Providing an isolated environment for testing MR codes while improving the system's reliability. On the other hand, solutions based on virtualization technology suffer from various performance overhead and limitations [108].

Lightweight virtualization

The use of virtualization technologies has been restricted in HPC and data analytics platforms because of their inherent performance overhead. However, the advent of advanced virtualization solutions, known as container-based virtualization (e.g., Linux Containers, aka LXC), has

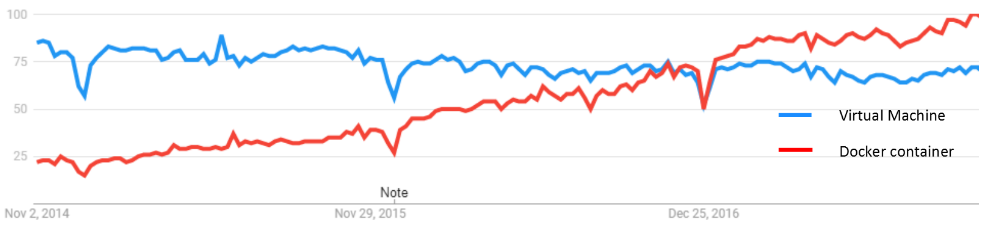


Figure 2.4: Container vs. Virtual machines interest over time.

lead to a growing attention in its ability to hold BD operations [109]. In principle, container virtualization is a kernel-level approach for deploying and running distributed applications in a sandbox environment without launching an entire VM for each application and task. This way, it does not require separate OS instances but multiple isolated environments, called containers, that are deployed on a single control host and access the shared OS kernel. Thus, by eliminating the hypervisor layer, container virtualization guarantees the resource share with higher efficiency and leading to near-native performance [110, 111, 112]. Containers relies on the cgroups (a Linux kernel feature to isolates the resource usage like CPU, memory, etc.) and namespaces functionalities, which are more efficient than VMs with very low overheads. Figure 2.3a shows the architecture of Docker, one of the most popular technologies for containerization, whereas Figures 2.3b and c plots the differences in the layered architecture of both container and hypervisor based virtualization.

In practice, containers are proving to be an extremely valuable technology for service science in delivering reproducibility and continuous integration. Therefore, there is a trend towards using containers in the form of container-based clusters (based, for example, in `Mesos`, `Apache Myriad`, `Docker Swarm`, `Google Kubernetes` or `CoreOS Fleet`). Figure 2.4 shows Google's Web search trends for 'virtual machine' and 'Docker container'. The search for virtual machines and containers have started to go side by side shortly by the middle of 2016 after which Docker containers search trends started to overtake that of virtual machines. This reveals how container-based technologies, e.g., Docker containers, are getting more attention from Internet users, compared to the traditional virtualization technologies.

Some research efforts started to adopt this approach as a promising framework for Hadoop. For example, `XConveryer` [113] has been presented as a Hadoop implementation based on kernel-level virtualization facilities to reduce job competition and eliminate the rescheduling caused by another virtualization. Xavier et al. [111] and Chung et al. [114] studied the performance impact of the isolation of container-based systems for both computer-intensive and data-intensive workloads. Rey et al. [115] proposed a simulator for testing and deploying MR applications using Docker with system fault-tolerance scenarios. In practice, container-based platforms are gaining momentum and have high potential for leveraging the DIA and BD operations. Some studies [116, 117] indicated that it is promising that LXC would be the future cloud and BD analytics further handling increasing memory requirements in data-intensive parallel programs [118]. However, prior to replace conventional hypervisor-based virtual machines with containers, it is crucial to address some concerns. For instance, security characteristics of container implementations that share the OS kernel with a deep level of authorization (usually root access). This could imply much more significant vulnerabilities and increase the potential propagating of malicious activities. Also, containers are stateless; after the container exit or die, the internal state is lost, so launching a new container requires it to be installed from the hub. A solution can be found in implementing a checkpoint mechanism that stores the persistent state of the container in a file system. Furthermore, other solutions include implementing a relaxed policy, by holding the container into a predefined time and waiting time for a process before exiting, in other words, to trigger speculative container.

2.3.5 GPU computing

GPUs (Graphics Processing Unit) have been originally designed for graphics processing with large data sets and complex computational intensive jobs. It has gained much attention due to their ability to speed-up response times when compared to traditional CPUs. Thus, there was increasing interest toward leveraging the General-Purpose GPUs (GPGPU) and GPU-accelerated systems as a scaling-up mechanism to increase computational efficiency, save energy, and exploit data parallelism [119]. On the other hand, GPU programming can lead to unnecessarily highly complex programs that suffer with data-intensive workloads because of the weak memory transfers (in-disk data movement between CPU's main memory and the local memory of the GPU) and the absence of direct access to files on the host OS file system. Additional challenges include the low level nature of their libraries and the synchronization problem of managing GPUs and CPUs. To address these issues, several research efforts suggested different solutions that can be categorized as follows:

- *Minimizing data transfer through the main memory*. Chen and Agrawal [120] proposed an approach for optimizing the shared memory by using a small programmable cache on the GPUs to enhance the system utilization. Other approaches [121, 122] focused on deploying new protocols to reduce data transfer time in the reduce phase by using I/O oriented scheduling to minimize idle I/O time.

- *Hide the programming complexity*. `Mars` [123] presented an approach for hiding the programming complexity of the GPU. `MR-Graph` [124] has been presented as a framework that allows its users to implement their applications more flexibly.
- *Using GPU Clusters*. `GPMR` [125], `MGMR` [126] and `GCMR` [127] are examples of systems that have been proposed to process large-scale datasets on GPU clusters. Moreover, D. Loghin [128] proposed to enable the GPU to process multiple ¡key, value¿ pairs in parallel.
- *Combining the CPU and GPU in a hybrid system.* Chen and Agrawal [129] proposed two approaches: the first one in which the the map processes are shared out between the CPU and the GPU, and a second one that executes the map task on one device and the reduce the other. Jiang, et.al. [130] developed two systems (`MGMR++` and `PMGMR`) to combine CPU and multi-GPU, where if data size is larger than the aggregate momory of the GPUs, CPU memory is exploited to continue the execution of MR processing.

2.3.6 Decentralized Environments

The great success of MR framework attracted both data and distributed system engineers to deploy it in emerging environments such as mobile computing or P2P. In contrast to centralized computing, the typical architecture design for Hadoop, both P2P and mobile computing are decentralized approaches where no single server node is responsible for all data and task allocation. Thus, adopting such environment is challenging, as well as promising.

Peer-to-Peer (P2P) computing

In principle, the design characteristics of the P2P model conflict with a fundamental assumption of the architectural design of the MR framework, which relies on a central management of computing resources using a master/worker strategy. In particular, P2P assumes that all nodes (peers) has equal privilege using a decentralized model. In general, deploying data-intensive applications in the context of P2P environments can introduce several advantages

Table 2.5: MapReduce in P2P environment

A typical P2P environment features are:	Why MR tasks on P2P environment?
• Non-dedicated (Nodes and connection).	• Less manageability and system administration.
• Heterogeneous capabilities (Low-end mostly).	• Low cost (aggregation of peer nodes).
• Mainly used for loosely-coupled CI workloads.	• Easy to install and scale up on-demand.
• Elastic, Auto-scale and fault tolerance (Replication) environment.	• Natural mechanism for fail recovery with high loosely-coupled tasks.

for MR-based systems. Table 5 summarizes some advantages and features of using a P2P environment for MR-based solutions.

Several research efforts have been focusing on achieving a reliable MR-based execution over the P2P environment. For example, the `PER-MARE` project [131] proposed to implement MR over pervasive grids. It relied on porting the MR paradigm over a P2P distributed computing middleware, `CloudFIT` [132], by adapting Hadoop to include context-aware elements to achieve efficient service execution on the target environment. Further, supporting Data-intensive MR-based clusters on pervasive environments was reported in [133] by employing ZooKeeper as a light-weight resource collector. In this approach, context-aware scheduling has been used for dynamically supporting resources availability status in the Hadoop-like systems. Wang et al. [134] proposed to use a queue structure to buffer tasks on P2P networks. This approach eliminates centralization by distributing scheduling responsibility among the participated nodes. Furthermore, it tackles data locality concerns by assigning to local tasks a higher priority when taking executive decisions.

Mobile computing

In the last few years, the processing power of smart phones and small devices have advanced so rapidly. These computing capacities are now more connected than ever. In principle, the ubiquity of mobile phones demands a decentralized system to exploit their potential and maximize its efficiency. In principle, synergising the computing of the massive number of small computing devices represent an emerging and promising new research domain. In particular, the deployment of MR implementations on mobile devices is considered as a special scenario that required the integration between two main technologies, namely, the ubiquitous computing and ad-hoc networks. However, in practice, mobile-based data analytics is facing many inherent resource constraints. For instance, the frequency of network disconnection is much greater than the one in traditional computer networks. In addition, there are still severe power restrictions that limit the life expectancy of the smart phone batteries. Other issues include the asymmetric communications costs, size of storage space and privacy. In principle, this environment can be characterized by the following primary features:

1. The volatility of the components, because participation is non-dedicated (mostly opportunistic) and device memory status fluctuates, which raises the availability issue.

2. The component's heterogeneity, whose capabilities can vary on different characteristics (both software and hardware).

3. The possession of the components where participation is mostly donated (volunteer). Obviously, protecting the privacy of individuals is also a main concern.

4. Connection status, because devices can change their network connection, switching from a fixed network connection to a mobile Internet, e.g., wireless to 4G.

`Misco` [135] has been proposed as a MR polling-based framework for implementing applications on mobile platforms. It comprises a Misco Server that keeps track of applications and input/output data and Misco Workers to process the individual map and reduce tasks returning the results to the server. On the other hand, `Hyrax` [136] is a Hadoop distribution that supports cloud computing on Android smart phones. Finally, George et al. [137] proposed implementing Hadoop MR framework over a mobile distributed file system, while Huang and Wu [138] proposed using heterogeneous mobile devices and a wireless router.

2.3.7 Hybrid environments

Hybrid computing system can be described as a set of heterogeneous computing paradigms that are connected in an amalgamating platform that is, typically (but not necessarily), attached to a unified computational pool of resources. In principle, the main goals of hybrid architectures are: (i) minimizing the capital expenditure on large private infrastructures, (ii) avoiding the static partitioning of computing resources and infrastructure silos, and (iii) reducing operational costs.

Scaling Hadoop MR using cloud bursting by coupling internal cloud to external cloud was the aim of `BStream` [139]. In its approach, BStream integrates stream-processing (using Apache Storm) with batch processing to realize inter-cloud MR by enabling the execution of the reduce operation in the external cloud using pipelined uploading. MOON [140] has been presented as a hybrid resource architecture implementation, in which a set of volunteer (non-dedicated) computing nodes supplements dedicated MR systems on-demand. Kijsipongse et al. [103] discussed the challenges of allocating a small set of dedicated nodes to BOINC volunteer computing system. `Aries` [141] has been presented as an approach for coupling the features of both low-power servers and high-performance ones in order to provide competitive performance and energy efficiency as well. The main target of this approach is to maximize the use of idle CPU, memory and network resources in microservers, using a fine/coarse granularity of resource management abstraction.

In contrast, `Nebula` [142] is a distributed edge cloud infrastructure that has been designed to efficiently exploit the edge resources from both dedicated and non-dedicated servers. `G-MR` [143] has been presented as an optimized framework that executes MR jobs in a hybrid Grid (as arising in cloud-of-clouds scenarios) using geo-distributed clusters. It uses data transformation graphs to find the optimal execution paths among edge nodes. The work by Tang, B., et al., in [144] proposed a hybrid environment that combines EDG and cloud instances. Their system `HybridMR`, employed a hybrid distributed file system and an adaptive scheduler that takes into account the performance differences between the system nodes.

Table 6 presents a comparison of different hybrid architectures and their key attributes. In principle, the key advantage for designing platforms that support resources management in hybrid environments can be summarized as follows:

1. The flexibility to adjust the computation capabilities on-demand and elastically.

2. Self-service functionality to meet the increasing demand for computing environments.

Table 2.6: Comparison of key attributes of Hybrid architecture

Hybrid Technique	**Ease of deployment**	**Flexibility/ Self-service**	**Scalability**	**Security**	**Fault tolerance**	**Data transition**	**Cost efficiency**	**Startup latency**
On-premise with Cloud	Moderate	Static	Scalable	Moderate	High	High	Efficient	High
Centralized cloud with edge-cloud	Complex	Dynamic	High scalability	low	High	High	Efficient	High
On-premise with EDG	Moderate	Static	Limited	Trustworthy	Moderate	Moderate	Efficient	Low
Private with public cloud	Simple	Dynamic	High scalability	Moderate	High	High	Efficient	Moderate
Cloud/local with IDG	Complex	Static	upon participation	Low, vulnerable	Low	Very High	Efficient	High

3. Economic savings for meeting peak demands without increasing the TCO.
4. Improving the overall performance by minimizing the job queuing, enhancing the throughput to the available resource amount and reducing the turn-around time.
5. Strengthening the system scalability by executing beyond the cluster capability with an auto-scale approach.

The main concerns are the high concurrent access to massive data sizes and several security threats. Some of the bottlenecks are due to the limited network connection, processing across different administrative domains, and the variety of computing resources. Also, the high startup latencies, force to wait for the entire system to be allocated in the resource pool.

2.4 Convergence of the paradigms

To analyze and evaluate these paradigms, we grouped them into two use cases, based on the resource location, namely, LAN-based (Intranet) and WAN-based (Internet) target networks. First, LAN-based implementations include: (i) computing clusters (Actual and Virtual), (ii)

Supercomputers (HPC), (iii) General-purpose computing on GPGPUs, and (iv) Enterprise Desktop Grids (EDG). Second, WAN-based target network implementations that include: (i) Cloud Computing, (ii) Peer-to-Peer computing (P2P), (iii) Grid Computing (volunteer DG and dedicated Grids), (iv) distributed High-Performance Computing (HPC) and (v) Mobile devices.

2.4.1 LAN-based Environment

Table 7 compares five different models: Hadoop in actual cluster (`HA`), Hadoop in virtual cluster (`HVi`), Hadoop in supercomputers (`HS`), Hadoop in GPGPUs (`HG`), and Hadoop in EDG (`HE`). This table summarizes the differences among these environments.

- *Resource capacity*: Both Hadoop `HA` and `HS` are built from high-end computing capabilities nodes (usually `HS` employ more powerful computing capabilities than `HA`). Hadoop was built to process large volumes of data, with higher level abstractions of data management (HDFS against MPI). HPC, in contrast, is aimed toward high-end computing capabilities for a wide range of computationally intensive tasks, with fewer volumes of data. Meanwhile, GPU clusters share the aim of processing compute-intensive tasks with the HPC, but for graphics processing with large data sets and weak in-desk data transfers (unlike the Hadoop storing file system). On the other hand, `HE` utilizes low-end desktop machines as a resource pool (e.g., a university laboratory or organization PC's) and `HVi` (depends on the physical capacity) had the lowest computing capacity. Notice that we considered an average VM that mainly deployed on a common server.

- *Connection strength and dedication*: `HA, HS`, and `HG` take advantage of high dedicated connection. `HVi` simulates a communication by providing many modes, e.g., bridge connection, network address translation, and internal networking among VMs. `HE` uses an opportunistic connection, in particular, when the network is idle or lightly used, that usually came with an average (intermediate) bandwidth.

- *Resource dedication patterns*: `HA, HS,` and `HG` intend to be dedicated devices (committed nodes), whereas `HVi` can be deployed and deleted on-demand and `HE` is not dedicated at all (opportunistic).

Table 2.7: MapReduce collaborative implementation comparison in LAN-based (Intranet) environment.

Criteria	LAN-based Implementation				
	Cluster		Supercomputer	GPGPU	Enterprise DG
	Actual	Virtual			
Resource capacity	-High	-Low *(Mainly for testing)	-High	-High	-Low-end *e.g. UV. lab
Network connectivity	-Dedicated -High speed	-Dedicated -Virtual	-Dedicated -High speed	-Dedicated -High speed	-Non-dedcate -low speed
Dedication	-Dedicated	-Dedicated	-Dedicated	-Dedicated	-Opportunistic
Resource support	-Homogeneous	-On-demand	Homogeneous	Homogeneous	Heterogeneous
Trust and Reliability	-High trust -Reliable	-Less than Actual	-Trustworthy -Reliable	-Trustworthy -Reliable	-Low trust -Unreliable
Paradigm Model	-HPC, MTC -DIC	-HPC DIC	-HPC -DIA	-HPC -CIA	-HTC, MTC -CIA
Easy of Deployment	- Moderate	- Easy	- Moderate-to-hard	-Moderate-to-hard	Hard
Scalability and Scaling type	-Scalable -Scale-out -Not-elastic	-Scalable -Scale-up/out -Not-elastic	Low-scalability -Scale-up -Not-elastic	Low-scalability -Scale-up -Not-elastic	-Scalable -Scale-out -Elastic
Deploying cost	- Moderate-to-High	-Low	-High	-High	-Low
Scaling cost-effectiveness	-Moderate -Higher cost	-High -No cost to scale	-Low -High cost	-Low -High cost	-High -No cost to scale

- *Heterogeneity*: All the models seem to be homogeneous (software and hardware), and many optimizations can include heterogeneous capabilities (in both computing and storage capacity). However, `HE` is heterogeneous by nature.

- *Security and reliability*: `HA, HS,` and `HG` are trustworthy environments (with `HA` is the highest) with reliable outputs. `HVi` seemed to be less reliable due to isolation problems with the original OS, which increases the failure ratio in some cases. In contrast, `HE` presents the lowest reliability.

- *Model*: Here we compare both the computing model (High-Performance, High-Throughput, and Many-task Computing) and application model (Data-intensive or Computing-intensive) that the environments originally represent. All `HA, HVi, HS` and `HG` are HPC, while `HE` is a `HTC` environment. `HG` shares with `HE` the application model as a CIA, the rest LAN-based paradigms are DIC environments.

- *Scalability*: The system's ability to scale and increase its capacities, originally `HA` is built to be easy to add new nodes (tens to hundreds). Yet it is not considered as a dynamic (elastic resources) on-demand technology. Also, it is difficult to scale both `HS` and `HG`; they are non-elastic as well. The `HVi` is easy to scale (cloning technique), but it is also not elastic to be scaled on-demand. As such, `HE` seems to excel since it is auto-scale and elastic but it is done opportunistically (non-dedicated).

- *Deploying and scaling cost*: `HVi` has the lowest cost of deploying a framework, and it keeps it in the same way when it is being scaled, i.e., very low cost with almost no HW cost. `HE` also is a cost-effective solution to deploy and scale. The most expensive solutions are the `HS` and `HG` due to hardware installation cost. However, `HA` is considered as being an intermediate solution.

2.4.2 WAN-based Environment

Table 8 compares six different WAN-based models: Cloud computing (HC), P2P (HP), Volunteer computing (HVo), Dedicated Grids (HG), distributed supercomputers (aka, Distributed Data Centers (HDS)) and Mobile devices (HM). Using the same comparison criteria and items in the previous section, some remarks can be made as follows:

Table 2.8: MapReduce collaborative implementation comparison in WAN-based (Internet) environment.

Criteria	WAN-based Implementation					
	Cloud Computing	Peer-to-Peer	Grid Computing		Distributed HPC	Mobile devices
			Volunteer DG	Dedicated Grid		
Resource capacity	-Variety Comp. and storage	-Low-end *Mainly PC's	-Low-end	-Intermediate *e.g.Virtual Org	-High-end *e.g.Virtual DC	-low-end *e.g. S.phones
Network connectivity	-Both patterns Intermed. speed	-Non-dedicated *Usually ad-hoc	-Non-dedicated Poor bandwidth	-Dedicated Intermed. speed	-Dedicated Intermed. speed	-Non-dedicated Ad hoc,3G,4G
Dedication	-On-demand	-Non-dedicated	-Opportunistic -Volatile	-Dedicated	-Dedicated	-Non-dedicated
Resource support	-On-demand *Handle both	-High-Hetro.	-High-Hetro.	-Low-Hetro. *less than VC	Low-Hetro.	-High-Hetro.
Trust and Reliability	Vary(Private/publ -Low-to-Reliable	-Low trust -Low Reliability	-Malicious -Low Reliability	-Trustworthy -Reliable	-Trustworthy -Reliable	-Malicious -Low Reliability
Paradigm Model	-HPC, HTC -DIA, CIA	-P2P -CIA	-HTC -CIA	-HTC, MTC -CIA (Mainly)	-HPC, MTC -DIA	-HTC -CIA
Easy of Deployment	-Easy	-Hard	-Hard	-Moderate	-Moderate	-Hard
Scalability and Scaling type	High Scalability -Scale-up/out -Elastic	-Auto-Scale -Scale-out -Elastic	-Auto-Scale -Scale-out -Elastic	-Scalable -Scale-out -Not elastic	-Low-scalability -Scale-up/out -Not elastic	High Scalability -Scale-out -Elastic
Deploying cost	-Low	-Low	-Low	-Moderate-to-high	-High	-Moderate
Scaling cost-effectiveness	-Efficient -Pay-as-use	-Efficient	-Efficient -Almost no cost	-Intermediate-to-high	-Low -High HW cost	-Moderate

- *Resource capacity*: Both HP and HVo contain low-end nodes mainly from desktop machines with moderate capabilities. HDS intend to own the highest computing capacity followed by HG with intermediate up to high-end nodes (i.e., that it may include average workstations or high-end nodes and clusters) as a virtual organization. Meanwhile, HC is the most flexible resource provider, which provide configurable resources based on the user's requirements and jobs demands, while, HM claimed to be the lowest-end nodes.

- *Connection strength and connectivity dedication*: In general, all of these approaches shared their connection via the Internet (HM uses mobile and ad hoc networks), which is much slower compared to the LAN-based networks. This connectivity results in an intermediate bandwidth upon which most of these approaches depend. In contrast, connection dedication is notably different, where HVo, HM, and HP do not seem to have a constant one. The remaining technologies have more dedicated (stable) connections with better connectivity.

- *Resource dedication patterns*: HDS is a dedicated system, likewise HG. Whereas HVo, HM, and HP intend to have a non-dedicated behavior (e.g., HVo is opportunistic), HC is, again, a flexible solution where resources are dynamically allocated and re-allocated based on user demand.

- *Heterogeneity*: Heterogeneous architectures are designed to reduce overheads among various devices that allow for the integration of these devices in a unified hybrid system. By categorizing the systems into two types: homogeneous and heterogeneous; based on single node and cluster architectures, HP, HVo, and HM are high-level heterogeneous paradigms. On the other hand, HG and HDS are designed, in most cases, by homogeneous model. HC, however, can handle both models on-demand.

- *Security and reliability*: Among all of them (even the LAN-based implementation) HVo and HM systems pose unique challenges from a computer security perspective (high malicious participation risk), and service reliability. The HP is similar in this aspect, but with better manners than the HVo. Other approaches like HC, HG, and HDS have better behavior and more trustworthy environments with failure recovery techniques and security services that prevent serious vulnerabilities. Though, in comparison, HC clusters and HDS are less reliable than the LAN-passed HA and HS, where HG is more reliable than HE.

- *System scalability*: The WAN-based approaches are mainly elastic, except `HDS`. While `HP`, `HM`, and `HVo` are dynamic to be auto-scaled whenever a new resource is available, `HC` provides elasticity by relying on a dynamic provisioning mechanism for allocating computing resources on a fine-grained basis. This feature provides the ability to be scaled up and down depending of the demand of resources. All WAN-based are scale-out approaches, however, `HC` and `HDS` are considered to be a scale-up approach as well.

- *Deploying and scaling cost*: WAN-based implementations, in general, are economical cost solutions (compared to the LAN-based approaches), and achieves economy of scale by reducing operational costs. Also, they represent a low cost of ownership, where scaling is cost-effective as well. However, `HDS` employ high-end servers, which indeed increase the capital expenditure on infrastructure capital, and is not considered as a dynamically auto-scale approach.

2.5 challenges and Directions

Although existing literature addresses many aspects of BD resource management and clustering, there are still some significant requirements and perspectives that need to be considered. This section focuses on how BD deployment architectures have changed over the last decade. In this section, we lay out a roadmap of challenges that would arise when implementing a data-intensive solution within any computing architecture. Also, we discuss the changing BD processing environments and consider the use of emergent paradigms from multiple infrastructures. For this purpose, Figure 2.5 charts out a variety of current challenges, emerging data analytics environments, and BD computing trends that will be offered by future platforms.

2.5.1 Challenges

Herein, we set out several challenges and research gaps that will need to be addressed for developing a BD solution.

Adapting the original paradigm design

To fully benefit from a new BD execution environment, one should adapt its prior design to meet the workloads requirements, as they may have been designed to accommodate another computing model (eg., high-performance, high-throughput, or many-task computing), with a

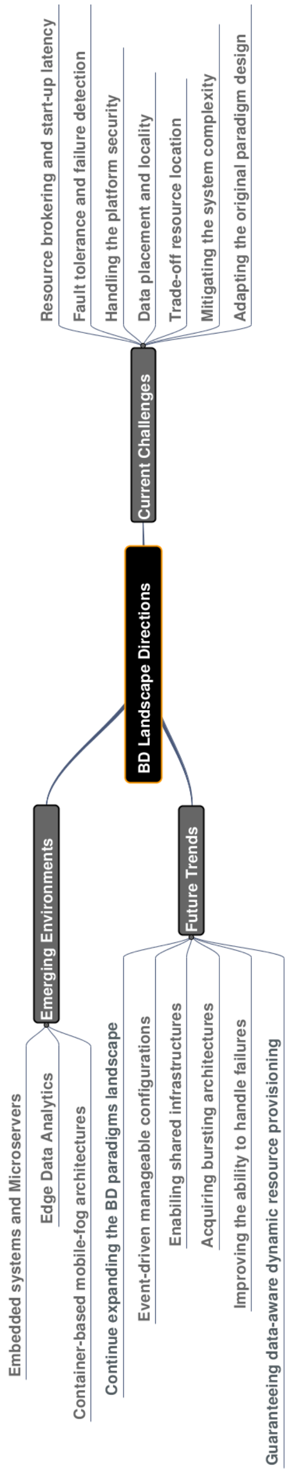

BD Landscape Directions
Current Challenges
Resource brokering and start-up latency
Fault tolerance and failure detection
Handling the platform security
Data placement and locality
Trade-off resource location
Mitigating the system complexity
Adapting the original paradigm design
Emerging Environments
Embedded systems and Microservers
Edge Data Analytics
Container-based mobile-fog architectures
Future Trends
Continue expanding the BD paradigms landscape
Event-driven manageable configurations
Enabling shared infrastructures
Acquiring bursting architectures
Improving the ability to handle failures
Guaranteeing data-aware dynamic resource provisioning

Figure 2.5: Overview of BD deployment architectures challenges and trends.

diversity of underlying resources (hardware, network, etc.). An implementer should consider the abstraction layers contributing to the Big Data stack to facilitate the adaptation of resources (network, storage, servers), data (file systems and databases), runtimes (frameworks), and applications. Changes could be required all over the layers in some scenarios, for instance, in IDG where nodes are highly heterogeneous and non-dedicated over geo-distributed locations using low bandwidth networks. Another example can be found in the distributed paradigms, e.g., P2P and mobile computing with decentralized management and low-end nodes.

Resource brokering and start-up latency

Resource brokering refers to dynamic relocation of available resources within the environment to optimize the cost and performance of an application at any time. This challenge also includes managing the underlying physical resources and monitor the resource (compute and storage) utilization and availability. In volatile and resource shared configurations, predicting the node availability for task allocation can be challenging since it requires sophisticated provisioning mechanisms of both static and dynamic techniques. On the other hand, start-up delay may happen due to two factors. First is data materialization, where the input data must be placed on the computing node before starting the process. This assumption could result in high latency due to the limited capacity of the network link in some architectures. Second, resource materialization, by registering all computing nodes in a single resource pool within a hierarchical clustering. So, the scheduler maps the available computation to applications/task desire to consume them and delay may occur by resource provisioning or processes deployment.

Handling the platform security

Having and maintaining a highly controlled management of the data, nodes, and processes deployed on all the machines is critical in designing a BD system. The fundamental abstraction of traditional BD architectures is directed to on-premise clusters. In such scenarios, faults are less harmful because most of the nodes are trustworthy and can run flawlessly for extended periods. Still, having high-reliability results to avoid potential misuse and malicious corruption is required for other configurations and paradigms as they seem to be more vulnerable. Result verification, as for result certificate, random sampling and majority voting techniques may seem unnecessary in a native Hadoop cluster. However, deploying an MR implementation in an unreliable environment (e.g., volunteer or Edge/Fog mobile computing), demands validation schemes for the results due to these environments vulnerability and volatility [145, 146]. Also, processing results can be secured via encryption.

Fault tolerance and failure detection

Traditionally, in Hadoop, detecting failures in machines and processes is based on heartbeats. Each slave node sends a static heartbeat every 3 seconds (this parameter can be configured). The master node performs checks on the expiry time report (timeout) of heartbeats every 200 seconds by default. When no heartbeat is delivered from a slave node for ten minutes (three timeouts), such node gets identified as a failed machine, and the master node triggers the failure handling and recovery processes. However, detecting the failure of sensitive tasks using this mechanism can return with a delay. Replacing heartbeat with an instant messaging mechanism to speed up task scheduling was proposed in [147]. Moreover, checkpointing may be considered an active failure handling technique in which the system is to restart from an early state point in case of failure [148]. Yet system-level checkpointing is considered as a heavy-weight approach for MR applications [37].

Mitigating the system complexity

In theory, enterprises make the choice on how and what resources to use for certain job demands. They may trade-off between using existing infrastructure (adapting a supercomputer for instance) or picking from a new best-in-class solution. However, this raises a compatibility issue, especially across the datacenter. This concern makes it challenging to pick and configure environment components and leverage all its features. Unfortunately, the complexity is not only due to the issue of compatibility, but also other challenges include navigating through logistics behind the system connectivity like long-term maintenance and management. One step towards addressing environment complexity is auto-managed architectures by the orchestration of its component which lowers system administration cost. Another solution would be utilizing containers to provide a compatible runtime environment across any platform.

Trade-off resource location

In the comparison of the two previous cases in section 2.4, LAN-based approaches meet the MR prime assumptions on the characteristics of the environment like dedication, homogeneity, and high connectivity. Meanwhile, WAN-based approaches are more dynamic, scalable and yield cost benefits, with variety in the environmental trust and reliability.

2.5.2 Emerging environments

In practice, BD computing infrastructure is expected to continue growing and evolving to meet the requirements of large-scale data systems and applications. In this section, we provide some suggestions on some computing environments that can play a significant role in future architectures.

Embedded systems and Microservers

The concept of fusing multiple embedded processors environments, including sensors into BD infrastructure, will emerge to accommodate the tightly coupled (usually dependent) data-intensive tasks. This trend is driven by the advent of microservers architectures, which have become increasingly pervasive. Modern microserver technologies, e.g., ARM[2] and Intel Atom[3], continue to improve their computational capabilities steadily, and are gaining momentum on being adopted in BD architectures [141]. Due to this, it is claimed that the next wave in the computing era will go beyond the realm of the conventional servers and desktop computers, gearing towards the use of more collaborative low-end servers, near-user edge devices, which will be responsible for a significant amount of storage and processing [149]. Many recent works have investigated this context over embedded low power processors such as [150]. Hashem and Ibrahim [151] concluded that BD solutions in such deployment architectures are in its infancy, but is becoming a practical field. Still its reduced memory, low I/O and memory bandwidth, and software immaturity may cancel the lower-power advantage of microservers. These are open challenges that needs to be addressed with future implementations.

Edge data analytics

In the last years, the use of computing nodes located at the edge of the networks has accelerated (e.g., fog and mobile edge computing), leading to new forms of heterogeneous computing models. Two factors have influenced this trend. First, the significantly increased number of

[2] `https://www.arm.com/`

[3] `https://www.intel.com/content/www/us/en/products/processors/atom.html`

end-users whom attach their devices to the cloud. Second, data processing is becoming more delegated from the clouds to the edge. These claims are evident with the advent of smart environments that are equipped with distributed sensors, which gather heterogeneous data at the edges of the networks. Exploring data runtime at the edge of the network, near the source of the data, hold a lot of potential. First, decreasing the volumes of data that must be transferred (confine traffic), and, hence, reducing the high latency in transmission costs. Second, enabling the optimization of a wide range of technologies to provide high-level abstractions for BD processing. Third, enhancing the interactive cloud applications that appeal to the end user's devices for closer execution, especially in wireless and mobile ad-hoc network connectivity that tends to be latency-sensitive. However, decentralized cloud models are not appropriate for widely stateless services with limited data transmission, such as the Web. It is also challenging to analyze batch data that originates in the cloud. These issues can be labeled as open challenges to address with future implementations.

Container-based mobile-fog architectures

It is anticipated that running BD tasks using containers on the mobile base stations is an interesting and promising trend. This is especially true for those who are still hesitating to distribute sensitive data over the mobile ad-hoc environment to enable achieving the IoT/BD vision. Also, using decentralized fog computing to work in conjunction with centralized architectures would facilitate more use of distributed computing. There are some general discussions on combining Docker and Android applications since they already run as containers on top of Linux. [152] describes how to run Docker on ARMv7 (the most widely used architecture in mobile devices) based devices. In addition, a recent study, investigated the effects of OS-level virtualization solutions on HPC performance [153]. Besides, aiming at accelerating and securing data exchange among nodes in this environment, specific data exchange channels that established by the nodes wishing to complete their data sets can be created on-demand privately. These private channels can be launched on an end-to-end for massive data transformation as an adapted application socket.

2.6 Summary

Growing interest in applying MapReduce-based implementations in various paradigms and configurations have given rise to numerous adoptions of large-scale data processing environments. These environments include but are not limited to Cloud Computing, Supercomputing, Decentralized Computing and Grid Computing. Thus, they must be considered based on their impact on performance, cost-effectiveness, scalability, resource utilization, overheads and, finally, heterogeneity in both the application and infrastructure. Herein, we have answered three fundamental questions related to the success of porting MapReduce implementations to the broad scope of deploying architectures. First, what are these paradigms and what are the challenges/potentials of their adoption? Second, given the dramatic differences among these environments, what are their essential characteristics and how can we compare and classify them? Finally, what are the current challenges and future directions of BD deployment architectures?

Our contributions are, therefore, three-fold: Firstly, introducing a novel classification taxonomy that groups data processing frameworks based on their environments, with emphasis on the resource perspective. The challenges, central issues, potential effects of adopting these environments, alongside the solutions are then proposed. Secondly, this work has exhaustively analyzed and compared the targeted platforms with Apache Hadoop, which is the leading open source MapReduce implementation. A significant part of this effort is due to missing standards in comparison to large-scale data processing environments, and the wide variety of different technologies of BD domain. The needed tables, figures, terminologies, and factors upon which/how to analyze and compare these paradigms, also, presented with a high-level discussion. Finally, based on the study observations, the current BD computing challenges with several potential future research directions have been suggested. We hope that this article help on guiding further researches to intersect and integrate BD and data-intensive operations with non-traditional solutions. A key goal is to create a convergence of abstractions of different approaches to promote better optimization of resource selection in different environments with reliable, low-cost services. Further, providing guidance on choosing the best platform to optimize the performance of various data-intensive applications in various environments.

CHAPTER 3

BIG DATA SCALABILITY AND PERFORMANCE IMPROVEMENTS

Aiming at enhancing the MapReduce-based application's Quality of Service (QoS), many frameworks suggest a scale-out approach, statically adding new nodes to the cluster. Such frameworks are still expensive to acquire and does not consider the optimal usage of available resources dynamically.

This chapter introduces a novel framework to address this issue by extending Hadoop YARN resource manager with dynamic provisioning and low-cost resources capacity uplift on-demand. We propose an Enhanced Mapreduce Environment (EME), and OPERA scheduler to support heterogeneous environments by extending Apache Hadoop to an opportunistically containerized environment, which enhances system throughput by adding underused resources to a local or cloud-based cluster.

3.1 Introduction

In recent years, under the explosive increase of global data, there has been an increasing emphasis on Big Data, business analytics, and smart living and work environments. Despite the success of large-scale commodity clusters like Apache Hadoop, this continuous data stacks require computational power far beyond the capability of the cluster workstations [154] in the high season intervals (i.e. when massive amount of jobs are submitted to the cluster). Naïve solutions may propose to add new computing nodes and scaling-out the tasks to distributed systems, like in hybrid cloud computing. Hence, two major concerns need to be addressed. First, it could be necessary additional investments on infrastructure (increasing the Total resources Cost of Ownership (TCO)), or paying to rent cloud instances. Second, the cost of data movement to and from the cloud over the Internet, which can be expensive and time-consuming.

As a comprehensive BD analytics platform, Hadoop has become the de-facto technology for storing and processing data-intensive applications (DIA) [52] and large-scale data analytics. These processes mainly use parallel data tasks operating, based in the MapReduce (MR) paradigm [5]. Thus, many frameworks aim at proposing scalable methods to support existent MR-based applications by enhancing the throughput using different techniques, while keeping in mind the QoS and the cost-benefit ratio. For instance, scaling Hadoop capacity horizontally and vertically (scale-up/out techniques), as in [155, 156] or adopting special hardware accelerators, as in [157] and [129].

Opportunistic Computing e.g., Enterprise Desktop Grids (EDG) could provide free compute cycles by harnessing idle CPUs in what is known as cycle scavenging technique [87], enabling the use of all available resources within the enterprise. Using this underused resources to extend the Hadoop cluster with low-cost, efficient and elastic desktop machines will improve throughput, making the most of the IT infrastructure. In exchange, such use would indeed increase the system complexity, so a powerful scheduler and resource manager would be required. Though, we argue that using cutting edge technologies like Linux containers can tackle that issue.

In this chapter of the thesis, we present the EME prototype, short for Enhanced Mapreduce Environment and its novel dynamic and hybrid scheduler, ie., OPERA. EME extends Hadoop with an adaptive hybrid (dedicated and non-dedicated) heterogeneous task allocation and provisioning on-demand. Its architecture combines High Throughput Computing (HTC) and Docker containers with large-scale dedicated clusters, leveraging all available resources. This project aims for a new hybrid environment that can elastically tune the resources participation, optimizing the cluster throughput without the need to add new and costly machines or rent external cloud services.

As a main component of the EME vision, OPERA takes advantage of leveraging the available and unutilized (ideal or lightly-used) capabilities within a physical confine of an enterprise. By scavenging these resources, OPERA keeps spawning Hadoop DataNodes, based on Docker containers as worker nodes. BD applications can then use these workers. OPERA is a standalone pluggable architecture that does not require making any modification of the prior Hadoop design. In OPERA, an opportunistic pool (HTCondor based) and a dedicated Yarn cluster can collaborate to improve throughput, with minimal cost of deployment. OPERA will launch disposable containers among the idle servers (creating an opportunistic container-based cluster) and ensure efficient provisioning for the Hadoop dedicated cluster on-demand. OPERA, hence, provides a seamless bridge from the pool of resources available in HTCondor to the YARN tasks that want those resources. By deploying a pilot (disposable) Container-based Cluster (CBC), i.e., a temporary lightweight layer on tops of HTCondor.

3.2 Background

There have been some proposals going towards managing, processing and maintaining massive datasets in what is called data-intensive processing with MapReduce Applications (MRA). This section presents a brief background on MR programming model and the Volunteer Computing systems alongside with related work and our main motivation to start this project.

1. MapReduce model.

   Ever since its introduction, MR framework has grabbed much attention in its ability to cope with parallel computing applications using a parallel data approach to process large volumes of data (terabytes or petabytes in size), which are typically referred to as Big Data. Apache Hadoop, the MR open source implementation, was naturally designed to be easily scalable, but it was not designed to be auto-scale or elastic. Many studies have proposed a resizable compute capacity in both local clusters [158] and the cloud [62].

2. Volunteer and Opportunistic Computing. Volunteer Computing (VC) is a form of network-based distributed computing, which allows public participants to donate their idle workstations and help to run computationally expensive projects [159]. Opportunistic Computing (OC) is a computer technique that make use not only of the resources available in the local machine or cluster, but can also opportunistically scale-out on other resources of the environment, including those on underused desktop computers, in a reliable and secure way [160]. Two major characters differ these paradigms. First, the resource ownership, i.e., VC take advantage of volunteer resources donated by participants while OC make use of untapped internal resources. Second, geographi-

cal topology, where OC follows an intranet (LAN) topology, while VC mainly follows Internet protocols (WAN). Correspondingly, HTCondor [161] is an open-source cluster resource manager aimed at Distributed High Throughput Computing (HTC) on collections of owned resources (EDG).

3.2.1 Related work

Many related studies in the literature classify the performance of MR data intensive computing into dedicated and non-dedicated resources. For instance, MRA++ [97] is a heterogeneous-dedicated methodology example that proposed a solution by grouping the machines according to their computational capabilities, calculating the execution time, and distributing Map task according to these capabilities. This proposal holds some drawbacks. First, it is not applicable for opportunistic environments, and volatility of data is ignored. Second, it does not take into account data placement (locality). Further examples include homogeneous-non-dedicated methodology [91] and heterogeneous-non-dedicated methodology [92]. Apache Myriad [162] is a case for multi-tenant shared services clusters. Myriad is an open source framework for scaling the YARN cluster into Mesos, which allows to expand or shrink the capacity of the cluster managed by YARN in response to events, as per configured rules and policies. Though, it differs from our proposal in that it works only on dedicated resources, like on a typical Data Center.

Correspondingly, some research using hybrid resource architecture has been carried out, where volunteer computing systems are supplemented by a set of dedicated nodes to reduce the cost. Such research includes MOON [163], which discussed the problem of how to allocate a small set of dedicated nodes to volunteer computing systems to offer reliable MR services on hybrid resource architectures, adopting the LATE algorithm [164]. However, MOON focused on a single job and worked on homogeneous environments only. Also, their solution needs a third part to manage the resource availability. By comparison, our study will follow hybrid resource architecture by adding some dedicated nodes to non-dedicated cluster in not only heterogeneous computing environment but heterogeneous operating systems as well. This will be achieved by using Docker based containers, which will allow us to build and distribute applications on heterogeneous nodes, thereby clearly differentiating our work from others.

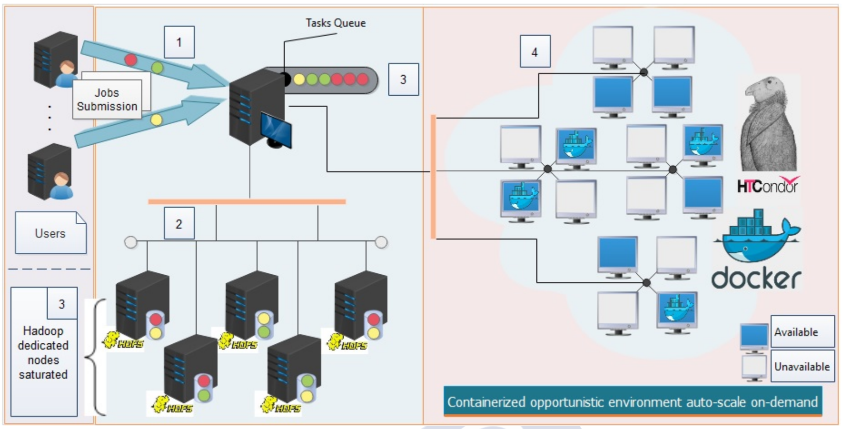

Tasks Queue
1
2
3
4
Jobs Submission
Users
Hadoop dedicated nodes saturated
HDFS
HTCondor
docker
Available
Unavailable
Containerized opportunistic environment auto-scale on-demand

Figure 3.1: EME Proposed Framework Architecture

3.2.2 Motivation

Nowadays, Hadoop resource elasticity and on-demand provisioning have not yet fully exploited by the present state-of-the-art MR execution architecture. We note a lack of studies that examine, from a resource perspective, the effect of resizable compute capacity (i.e., elastic resource provisioning) on MR applications performance. However, in this project we are committed to design, build and evaluate a hybrid distributed computing (dedicated and non-dedicated resources) architecture. We will use heterogeneous clusters, with differences in both computing power and availability (opportunistic computing), to extend Hadoop's MR task scheduling in a variety of environments and operating systems, using containerized based clusters on-demand.

The key challenges we are addressing in our work can be summarized as the following:

- Defining the Hadoop cluster threshold where resources are stretched to the limit (big data workloads start queuing).
- Efficiently execute MR operations on non-dedicated resources (isolated in containers).
- Opportunistically use resources that may be lightly loaded for long periods of time within an enterprise
- Elastically provisioning Hadoop's Yarn on-demand with low-cost nodes.

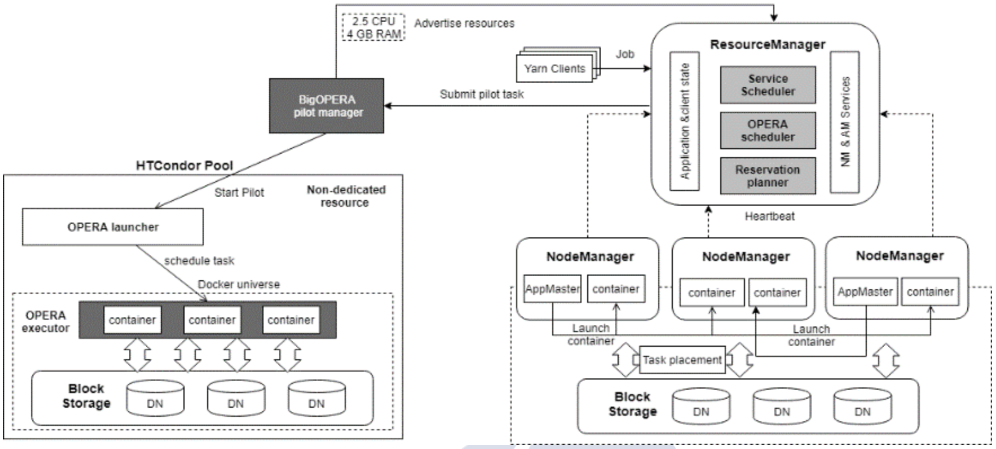


Figure 3.2: OPERA architecture

- Enabling the two environments work harmoniously for the benefit of business and academia
- Making this framework automated (auto-scale), fast and extensible to the prior Hadoop Yarn architecture.

3.3 Methodology

To take advantage of underutilized resources, we deploy YARN containers on top of the HT-Condor pool system. When a job comes to YARN, it places it at a set of dedicated slaves. Under continue job submission, tasks start queuing at the job pool, waiting for a slot. The adapted YARN asks for an offer via the OPERA scheduler (second level scheduler, see Figure 3.3). OPERA will match the request to incoming HTCondor resource offers. HTCondor, in turn, will launch pilot Docker containers among the underutilized nodes. The HTCondor machines will then communicate the request to OPERA executor, which is running DataNode containers. Next, YARN can consume resources as it sees fit.

To enable a universal block storage layer, Hadoop performs a separation of namespace and block storage in a federation configuration of the cluster. A federated BD environment improves the scalability and isolation of Hadoop operations. In our approach, we scale the NameNode horizontally over the HDFS federation using two different data pools (namespaces), one for the dedicated Hadoop cluster and another for the HTCondor containers (see Figure 3.2). This approach enables the aggregation of a hybrid BD infrastructure (dedicated and non-dedicated). This shared environment facilitates data placement among the containers while supporting pilot job abstraction. When Hadoop YARN reaches a peak utilization, the AplicationMaster asks OPERA to launch a new container over the idle HTCondor resources.

3.4 EME and OPERA: Framework architecture

Prototyping a distributed application like MR is considered a hard and time-consuming task. In this section, we will introduce EME's design, goals and architecture overview.

To define the Hadoop cluster threshold and whether the dedicated DataNodes are saturated, OPERA used Algorithm 1. OPERA thus monitors the I/O bandwidth and utilized capacities at each dedicated DataNode and sends this information to the NameNode. The HTCondor manager, in return, deploys a DataNodes using its Docker universe over any idle or lightly used nodes in its pool. These opportunistic DataNodes assigned under the control of the NameNode (the ApplicationMaster) to meet the peak utilization intervals. The container is deleted once the job is finished (reported by the ApplicationMaster). The container is also deleted once the resource is needed again by the HTCondor tasks.

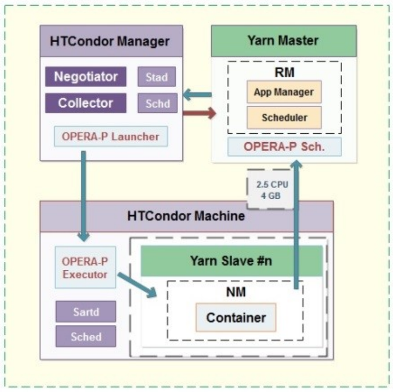


Figure 3.3: OPERA resource allocation demonstration.

Algorithm 1: Defining dedicated DataNodes threshold

```
1  Let $\widehat{\phi_i}$, be the volatile DataNodes in the cluster.
2  Let $T_s h$, be the defined threshold of the system.
3  Let $J_z$ be the measured job size at time-step $q$.
4  Input: current dedicated cluster size $Cz_i$
5  Output: setting throttling state of the dedicated node
6  $avg\_Jz = \frac{(\sum_{j=i-w}^{i-1} Cz_i)}{\widehat{\phi_i}}$
7  if $Cz_i > avg\_Jz$ then
8      if $(state == unthrottled)$ and $(Cz_i < avg\_Jz * (1 + T_s h))$ then
9          $state = throttled$
10     end
11 end
12 if $Cz_i < avg\_Jz$ then
13     if $(state == unthrottled)$ and $(Cz_i < avg\_Jz * (1 - T_s h))$ then
14         $state = throttled$
15     end
16 end
```

3.4.1 Design overview

The proposed framework presented in Figure 3.1 consists of three major components. First are the users, EME's framework targeting a composite (multi-users/multi-jobs) architecture. Second, a dedicated environment, i.e., a typical Hadoop cluster that follows master/slave technique, where Yarn is responsible for scheduling, monitoring and re-executing failure tasks on slave machines. Third, a non-dedicated environment, which could be controlled using VC middleware like BOINC [165]. However, our approach adopts an elastically and horizontally scaling (scaling-out) of resources in a controlled LAN to opportunistically execute MR tasks on collections of distributed, underused and owned resources on-demand. So, to manage and allocate the non-dedicated nodes, HTCondor (to create an EDG within LAN, i.e., an HTCondor pool) can be used in EME. Thus, the proposed integrated architecture will reduce the system complexity and avoid system degeneration due to overloaded Hadoop master. Anyway, a lightweight scheduler has to be implemented in HTCondor, which only consider container deployment inside the opportunistic pool.

Figure 3.4 presents a layered architecture and abstractions of both environments. For the opportunistic environment, an operating system-level virtualization environment (Docker

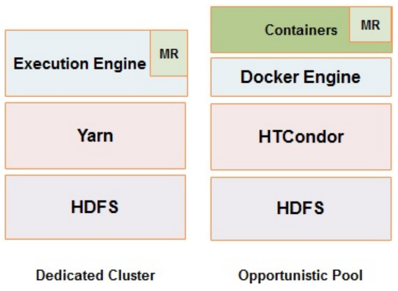


Figure 3.4: EME architecture layers and abstractions

containers) will be implemented, with HTCondor as its execution engine. This will provide not only an efficient use of the opportunistic resources, but also an isolated fault tolerance environment with high scalability potentials.

With this additional nodes, the throughput will be significantly improved. So, with a minimal cost of deployment, EME will improve scalability and optimize idle resources utilization, which would imply, as another direct impact, a greater return on infrastructure investments.

The proposed framework, presented in Figure 3.2, consists of three components. At first, an HTCondor manager, which is responsible for monitoring and re-executing failure tasks on slave machines. It also contains a container launcher that keeps spawns YARN Node Managers (using local repository Docker containers) among the dedicated or non-dedicated nodes (see Figure 3.3). Second, an adapted HTCondor machines, coupled with OPERA executor that launches the containers on the HTCondor pool, and ensures running the YARN tasks in a sandbox environment. Finally, a typical YARN cluster with an integrated OPERA scheduler at the master workstation that runs as a second level scheduling service. OPERA scheduler keeps contact with the process status on the HTCondor pool —an example of how the OPERA workflow run is illustrated in Figure 3.5.

3.4.2 Fault tolerance

The basic abstraction of Hadoop architecture is directed to cluster and supercomputing paradigms. In these environments, faults are less harming as most nodes are reliable and will operate flawlessly for an extended period. Though, the failure of a node is still a concern. Some techniques like task replication, speculative execution of tasks, and heartbeats may limit the threat on most scenarios. However, the enterprise grid (non-dedicated) environment may seem more

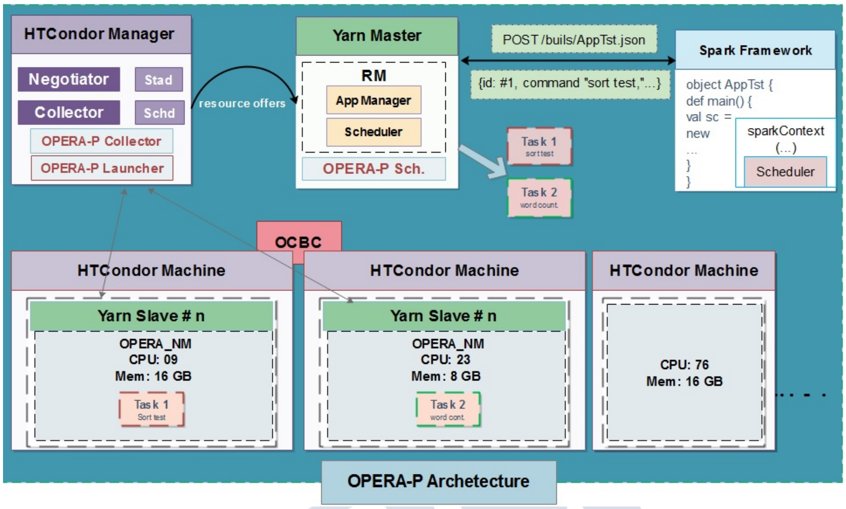


Figure 3.5: OPERA workflow example.

vulnerable. Accordingly, an advanced technique that supports fault-tolerance, while minimizing its impact on performance is a must.

In our work, we will concentrate on the enhancement of fault-tolerance on the opportunistic pool. Keeping in mind that no single point of failure is predicted at that environment, we propose two approaches for solve this problem. On the one hand, tuning the heartbeat may enable the NodeManager in the opportunistic resources to register dynamically on the dedicated ResourceManager (see Figure 3.6). Additionally, the nodes history (availability) will be added as an additional policy to determine the global resources availability.

On the other hand, the results obtained in the non-dedicated environment will be checked using the majority voting technique [100]. This will add two main advantages to this scenario. At first, majority voting will require a minimum of three containers task replication to be implemented, which directly impact the fault tolerance by enhancing the task failure recovery. Second, leaving the result submission (the work done flag) until we collect two out of three results for each work from different workers, will provide high reliable results.

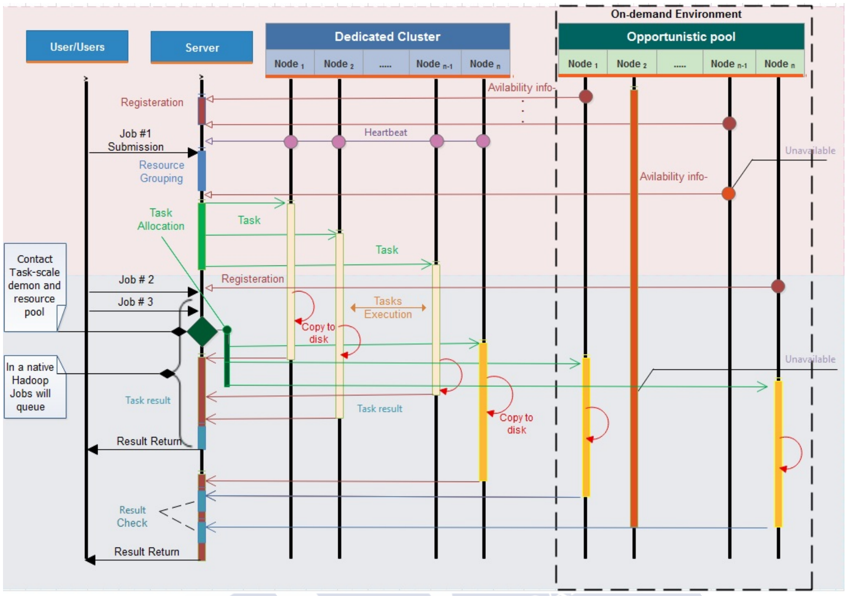


Figure 3.6: Framework sequence diagram

3.5 Experiment Results

Docker provides a systematic way to automate the faster deployment of variety applications and libraries inside portable containers (isolated environment in operating systems kernel). Also, Container as a Service (CaaS) is considered as a particular case of IaaS without the hypervisor layer for better performance. Hence, EME proposes idle nodes as a viable way to deploy an opportunistic environment using disposable containers (deleted after executing the task) for provisioning the Hadoop cluster.

On the other hand, HTCondor will provide a consistent environment that continuously schedules new containers when it is possible and quickly gives resources back to the user when required. In our proposed architecture, Docker will provide a runtime service that ensures three aspects. First, isolation between the MR-tasks and other resources operations with minimal configuration on the HTCondor pool. Second, secure containers deployment with lightweight virtualization that runs on near bare-metal performance. Finally, an extensible interface that copes with Hadoop Yarn, where no rewrite for the MR applications is required.

For evaluating OPERA performance enhancement and overall throughput growth, we

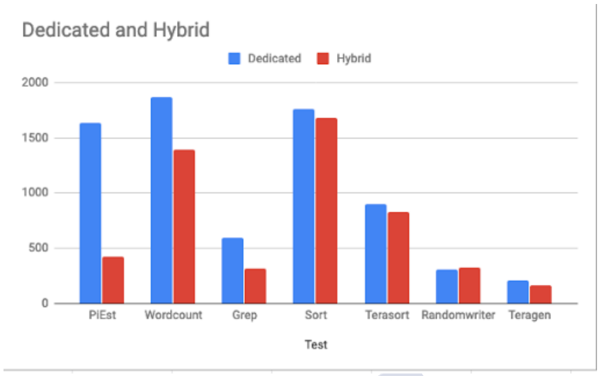


Figure 3.7: OPERA performance against native Apache Hadoop.

Table 3.1: Hadoop cluster with two dedicated DataNodes and two opportunistic DataNodes

Node Number	Hadoop Daemon	Hardware & Configuration
1	NameNode	CPUs: 2 x Intel Xeon E5506 @ 2.13GHz, RAM: 12 GB RAM ECC DDR3 @ 800MHz, HDD: 2 TB HDD
4	DataNode	
Common criteria: Hadoop 3.1.1 over 4 x Gigabit Ethernet ports.		

compared the performance of MapReduce benchmarks on native Hadoop environments against the proposed one. In Figure 3.7, the X-axis represents the benchmark and stress testing of different tools. Y-axis represents the execution time (job completion time) over 40GB of data. The testbed was deployed over a physical cluster, as shown in Table 3.1 of Hadoop 3.1.1 platform.

We can observe that OPERA achieved definite improvement with up to 74% for CPU bound jobs, as in the PiEst and up to 26% (with an average of 26%) regarding HDFS bound jobs, as in the Wordcount. The improvement varies within different benchmarks like Grep, Sort, Terasort, Randomwrite, and Teragen. We also observe that the performance gap widens positive linearly with the increase of the tested data set.

To investigate this behavior, we benchmark the performance in terms of read and write I/O throughput on the OPERA environment normalized to the corresponding native Hadoop dedicated cluster. We use the Hadoop TestDFSIO benchmark, and the results are shown in Figure 3.8. The evaluation shows a performance improvement with an overall average of 22% (26% in the best scenario and 20% in the worse scenario). Moreover, for all benchmarks, as the size of input data increases, the performance gap widens in favor of OPERA.

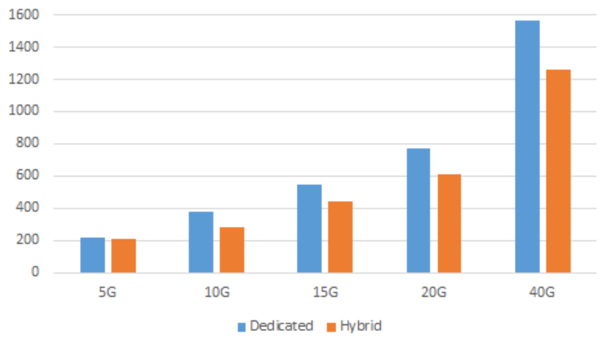


Figure 3.8: OPERA Result.

3.6 Discussion

As Figure 3.1 illustrate, the framework can be divided into four main phases that can be classified as follows:

1. Job submission: The job/jobs are submitted to the Hadoop master by the users via a job submission manager. The job manager at the master will place these jobs into a global job pool (task queue), where tasks wait to be batched.

2. Task allocation: Yarn manages task allocation, status monitoring, and reallocation failure tasks at the dedicated environment. Each dedicated slave shall run a NodeManager and a DataNode daemon. Yarn will start an ApplicationMaster per job, which asks the master (ResourceManager) for resources. The ResourceManager will contact the NodeManagers, and will provide the required resources in the form of containers. The ApplicationMaster will run the jobs in the allocated containers till the cluster reaches a predefined threshold where resources are fully utilized, and no more tasks can be processes.

3. Cluster saturation: Under this circumstance, tasks start queuing at the job pool. The Yarn decision maker will use the global ResourceManager and will ask for an offer from the available (registered) resources at the opportunistic environment (i.e., in a pull approach).

4. Opportunistic resources Provisioning: At this point, the ResourceManager asks a special daemon (ContainerLauncher) to start containers (Docker) at the available opportunistic resources. These containers will run a NodeManager daemon. The new containers will refresh the ResourceManager offering these resources for the ApplicationMaster, which only exist in the dedicated environment.

Hence, the container launcher will create several Docker containers on the idle nodes for executing the independent task chunks in parallel. The ApplicationMaster will run these jobs in the allocated opportunistic containers, and the results will be sent back to the dedicated cluster via a result manager, which is in charge of result checking. Figure 3.6 shows a sequence diagram that explains the task execution phases in a sequence action timeline.

OPERA is thus a second-level scheduler by separating concerns of resource allocation and task placement by employing a pilot-job abstraction. In definition, pilot-job is the concept of multi-level scheduling as manifested in the decoupling of workload assignment from resource management using the concept of intermediate container jobs. In this context, a pilot-job is the ability to utilize a client's job as a container for a dynamically determined set of computing tasks. In particular, Figures 3.7 and 3.8 demonstrates its usability over data-intensive job execution with up to 74% performance enhancement compared to a native Hadoop cluster.

3.7 Summary

Deploying a variety of computing frameworks and clusters on the same resources infrastructure was the main objective of many paradigms. Virtualization technology, computer utility, and cloud computing, to mention some. Those paradigms aimed at maximizing the efficient use of resources and minimize the associated deploying and operational costs. Sharing resources between deferent resource managers and service provisioning on-demand approaches was the subject of many other studies in the literature. So doing was motivated by improving the overall cluster utilization and avoiding statically partitioning resources and infrastructure silos among separate clusters and resource schedulers.

Big Data analytics and data-intensive applications industry witness many enhancements regarding large-scale processing. Apache Hadoop, with its ecosystem, is the dominant platform and become a comprehensive OS-like platform. However, current implementations and resource managers are unable to employ lightly loaded resources within their controlled network, which is not yet fully exploited in the MapReduce state-of-the-art architectures. The research work described in this chapter fills a vital gap in understanding statically partitioning the infrastructures among different resource managers.

Here we introduced the OPERA and EME, which aims at filling the increasing gap between computation and I/O capacity on high-end workstations. OPERA will help developers, resources providers, e-infrastructures, and scientific communities to overcome current challenges in the large-scale data analytics from both PaaS and IaaS levels. OPERA allows the coexistence of different resource managers and sharing computation among different resource managers. Also, it enables automatically adapting the cluster size to the resources demand. This hybrid architecture intends to (i) minimize the capital expenditure on large private infrastructures, (ii) removing the statically partitioning of computing resources (iii) improving DIA performance and (iv) to reduce operational costs.

On the other hand, EME, the prototype blueprint, aims at improving data-intensive computing performance in both local and cloud-based clusters. EME offers a hybrid resource architecture (dedicated and non-dedicated) that supports heterogeneous capabilities by extending Hadoop's cluster beyond its dedicated nodes, with elastic and low-cost resources on-demand. In particular, employing an opportunistically container-based cluster with auto-scale capabilities. EME will allow to automatically adapting the cluster size to the resources demand. This hybrid architecture aims to minimize the capital expenditure on large private infrastructures and to reduce operational costs. Additionally, it improves the MapReduce infrastructure throughput, performance, and QoS, helping to meet job deadlines.

Even that our work is being developed with Hadoop Yarn as the supporting resource manager, but we believe that the ideas presented in this research can easily be adapted to other resource management frameworks, for instance, Apache Mesos and Docker Swarm. As it is usual in the literature, we will start prototyping EME in a virtualized cluster, and next, we test it in a bare-metal environment.

The work presented in this chapter contributes to the state-of-art practice of Hadoop by guiding the decision on the enormous technological advancements of Hadoop federation use cases. We believe that both the findings and the approach presented in this chapter will appeal to BD practitioners and researchers from different disciplines. Besides, it will act as a catalyst in the initial study for all those researchers who are implementing BD solutions within any runtime environment.

CHAPTER 4

BIG DATA FEDERATION AUTHENTICATION AND AUTHORIZATION

Privacy and security in a federation scale remain a significant concern among practitioners. Current Hadoop's primitive access control has security concerns and limitations, such as the complexity of deployment and the consummation of resources. This significant challenge can be tackled using Access Control techniques.

This chapter intends to propose a trustworthy model for authenticating users and services over a Big Data Federation deployment architecture. The main goal of this model is to provide a Single-Sign-on (SSO) approach for the latest Hadoop 3.x platform. To achieve this, a conceptual model is proposed combining Hadoop access control primitives and the Apache Knox framework. The paper provides various insights regarding the latest ongoing developments and open challenges in this domain.

4.1 Introduction

Apache Hadoop, the prominent Big Data (BD) paradigm, has become a large-scale data analytic operating system. The large community behind Hadoop have been working to improve its ecosystem to meet the security increasing demands and requirements. The new features shift proves the Hadoop 3.x maturity and applicability to serve different markets. Enterprises across all major industries have adopted Hadoop for its capability to store and process an abundance of new types of data and leverage modern data architecture. With a broad spectrum of both structured and unstructured workloads, Hadoop abstracts the computing resource management, task scheduling, and data management, while maintaining a satisfying level of security and isolation.

The Hadoop Distributed File System (HDFS) is typically deployed as part of a large-scale Hadoop platform, to supports low-cost commodity hardware and accommodate different processing frameworks. It is utilized to handle data management and access to the Apache Hadoop ecosystem, using a master/slave architecture. The latest advancement in BD platforms leads to support of a clear separation of data management and its physical storage. Using multi namespaces in the form of federation architecture, Hadoop improves its scalability and isolation.

In this section, we propose a mechanism to integrate Hadoop 3.x authentication schemes into Apache Knox [166] in a high-level abstraction. This presents a first work to define and formalize a service gateway for a BD federation (BDF) deployment architecture. The key question we will be asking for addressing the access control challenge is “How to design SSO abstract that handles Data/Service access in a Hadoop federation?” We address this question by defining a SSO reference architecture (SSORA) for the secure development of BDF. The work presented in this section may employ in a Hadoop federation environment across multi-tenant BDaaS and IaaS clouds, as well as on-premises datacenters.

4.2 HDFS federation

HDFS [167] is composed of two primary daemons, (i) a single NameNode (NN) that is deployed at the cluster master node, and (ii) several DataNodes (DNs) running at the cluster slaves (usually one per-node). The NN runs the namespace process, which manages the file system information and regulates access to files using a traditional hierarchical organization.

To enable universal block storage layer, Hadoop performed separation of namespace and blocked storage [168]. A federation BD environment, through multi-independent namespaces for block management and a shared block pool for data storage, improves scalability and isolation of Hadoop operations. So, by losing the tightly-coupled block storage and namespace, each DN registers with all the NNs in the cluster (raises authentication requirement). This allows to scale the NN horizontally and enable the aggregation of geo-distributed Hadoop clusters. This feature directly enhances throughput by adding more access enforcers (typically NNs in HDFS architecture), which improve read/write operations.

Table 1. Hadoop Federation SSORA Model Definitions.

Basic Semantic

- U, R, P, A (finite set of users, roles, permissions, and actions respectively)
- HS, HC, HCC (finite set of Hadoop service, cluster, and cluster components apiece)
- AR_{re}, AR_{wr}, AR_{ed}, AR_{conf} (finite set of Authorization Roles: read, write, edit, and configuration)

Functions

- Direct UR: $U \rightarrow R$, mapping each user to a role $\in$ (user, superuser, owner, admin)
- Direct RP: $R \rightarrow P$, mapping each role to a set of permissions $\in$ (read, write, edit, and configuration)
- Direct HSA: set relation between the Hadoop service and role actions
  - HC→A, mapping roles to Hadoop cluster authorization level
  - -HCC→A mapping roles to Hadoop cluster compnents authorization level

Actions Assignment

- $AA_{HS} \subseteq (U \cup R) \times HC \rightarrow A \supset HCC \rightarrow A$, mapping Hadoop cluster permission level (represented by each NameNode) and then the permission level of its components (represented by DataNodes in HDFS architecture).

SSORA Authentication Decision

- A user $u \in U$ is assigned to a Role $r \in R$ to operate the Permissions $p \in P$. The Hadoop service authorization level is tagged to Hadoop cluster (e.g., NN1) and then Hadoop cluster component (e.g., DN2).
- $APA_{HS} \rightarrow \text{Permissions}_{(p)}$, defined as operation $(x) = \{P \mid (x, p) \in AA_{HS} \times (U \cup R \cup A)\}$
  - e.g., APA_{HS} := {John, Admin, Hadoop cluster 1 && 2}.

4.3 SSORA access pattern

There is a necessity to create a common language tailored to build an efficient, elastic and autonomous access control reference model for BDF. This reference model will not only define patterns that can result in the dynamic optimization of accelerating access enforcement, but it will keep access discussions to a minimum as well. Here, we present a HDFS federation SSO Reference Architecture (SSORA), a sequential pattern-based access control for a federated SSO model. This will aid in clarifying the required tools/mechanisms to implement a SSO architecture for Hadoop 3.x clusters.

SSORA affords a binding among the external clients and the Hadoop services in a federated Hadoop platform. It enables the internal entities (e.g., NN and DN daemons) to verify these bindings through the use of a central policy authorization and several distributed policy agents. We will now discuss the formal definitions of SSORA model as shown in Figure 5.6 and specified in Table 1.

`SSORA Pluggable Module` enables fast integration with architecture (i.e., allows to plug functionality without modification), it does not require making any modification of the

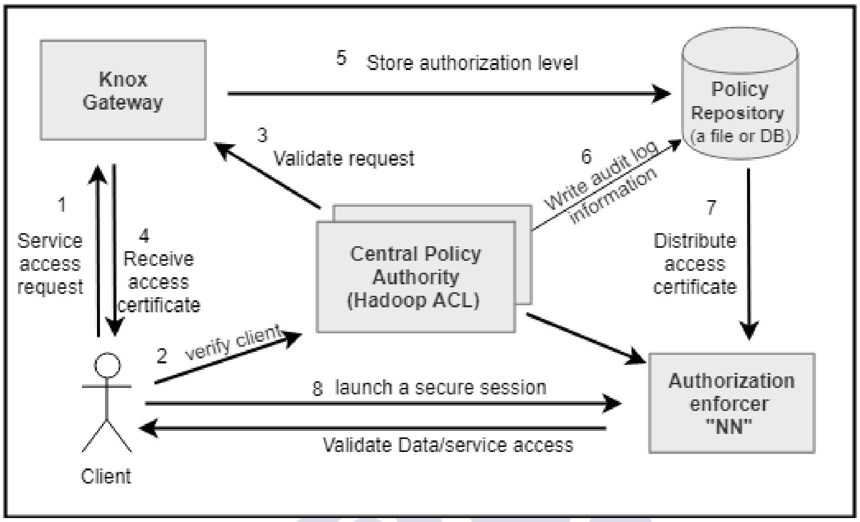

Knox Gateway
5 Store authorization level
Policy Repository (a file or DB)
3 Validate request
6 Write audit log information
1 Service access request
4 Receive access certificate
Central Policy Authority (Hadoop ACL)
7 Distribute access certificate
2 verify client
8 launch a secure session
Authorization enforcer "NN"
Validate Data/service access
Client

Figure 4.1: SSO Reference Architecture (SSORA) and federation access pattern.

prior cluster design. Before addressing the SSORA and federation access pattern, there are few definitions to guide and facilitate the discussion:

- Apache Knox service is utilized as a unified access point of the federation service.
- Hadoop daemons and services (internal communication) are authenticated using Hadoop core capabilities, i.e., Kerberos principals and their access granularity is managed using POSIX ACLs.
- Knox authenticates clients (admins may set an LDAP group), and Apache Ranger [169] applies their access granularity.

In a federation environment, it is often the case that a client of a particular HDFS cluster (e.g., NN1) wishes to utilize services offered by another administrative domain (e.g., NN2). In such cases, the user needs to be authorized by the central policy authority (CPA) to reach the remote domain. The necessary steps required for a given BDF to access the provided SSO federation are described subsequently (see Figure 5.6):

1. The Client asks to access the Hadoop cluster, interface with the cluster gateway (Knox) using his identity. If the external client wish to upload new data to the Hadoop data

lake, the gateway may ask to secure the connection via SSL/TLS communications over the public network.

2. If Knox cannot find a valid session for the access request (for first-time registration) it start a secure session with the CPA (ACL in Hadoop) on clients' behalf.

3. The CPA validates the proof of authentication and issues a new security certificate to initialize his permissions (get the authorization level access). The certificate is forwarded to the gateway interface and performs a login if needed.

4. Access certificate containing the ClientID is issued for valid users and is signed with the CPA private key. The client receives the certificate and adds it as an embedded proof of access. This certificate provides an abstraction for combining any number of authentication and authorization systems. Also, it accelerates authentication to a vast number of participants (under the federation umbrella).

5. The policy repository is updated, and the authorization level is stored. The CPA can always issue new security session without the need for storing the client identities, permissions, and logs locally. Policy agent will keep listening to the new authorization session.

6. For auditing purposes and security analysis, the CPA manage user activities (at the service level) by writing audit logs in a common repository.

7. The CPA centralizes the access control and shares the allowed user/calls via a push mechanism to the Authorization Enforcer (AE), which is, typically, the NN in the architecture.

8. The trust relationship between the client and AE is established, and his requests are authorized to proceed. Other security layers may apply (e.g., an internal TLS and files encryption).

Figure 4.2 shows a sub-method of the admin permission in SSORA as an XML fragment policy that performs two operations (delete and intra-node balancing) over BDF.

```
<method-permission>
  <Role-name>  Admin </role-name>
  <HS-name>
       <HC-name> nameNode1 </HC-name>
       <HCC-name> datanode2 </HCC-name>
       <method-name> delete [file_path] </method-name>
  </HS-name>
  <HS-name>
        <HC-name> nameNode1 </HC-name>
       <HCC-name> datanode2 </HCC-name>
       <method-name> intra-node disk balancing </method-name>
</HS-name>
```

Figure 4.2: A representative SSORA XML example.

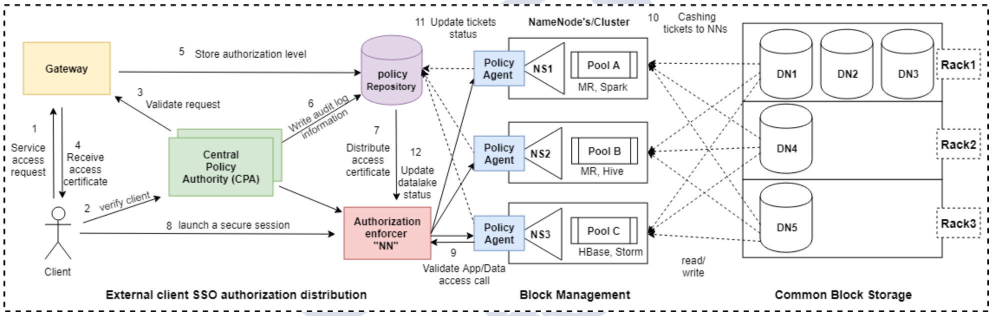

Gateway
5 Store authorization level
3 Validate request
1 Service access request
4 Receive access certificate
Central Policy Authority (CPA)
2 verify client
8 launch a secure session
Client
6 Write audit log information
policy Repository
11 Update tickets status
7 Distribute access certificate
12 Update datalake status
Authorization enforcer "NN"
9 Validate App/Data access call
NameNode's/Cluster
Policy Agent
NS1
Pool A
MR, Spark
NS2
Pool B
MR, Hive
NS3
Pool C
HBase, Storm
10 Cashing tickets to NNs
read/ write
DN1
DN2
DN3
DN4
DN5
Rack1
Rack2
Rack3
External client SSO authorization distribution
Block Management
Common Block Storage

Figure 4.3: SSO Reference Architecture (SSORA) and federation access pattern.

4.4 Implementations and use cases

BD federation, in general, provides a systematic way for dynamically provisioning the user demands with elastic resources and services over shared data blocks. Implementing a federated authorization and delegation to enhance BD client access decisions is vital for the success of adopting these Hadoop 3.x new features. This aim translates the IAM within the Hadoop core mechanisms and the Apache BD security-oriented frameworks. Here in, we argue that FACRM is essential for the concept of unifying the IAM regarding BD federation infrastructures and the data that they stock. We demonstrate how the proposed components of BD federation deployments can be brought to a centralized SSO-based fabric and Multi-tier Hadoop architectures.

4.4.1 BDF SSO access model

As it should be clear by the previous discussion, there is a necessity to create a common language that is tailored to build efficient, elastic and autonomous access control RM for BDF platforms. This RM will not only define patterns that can result in the dynamic optimization of accelerating access enforcement, but it keeps the IAM discussions to a minimum as well. Next, we present HDFS federation SSO Reference Architecture (SSORA), a sequential pattern-based access control for a federated SSO model. This will aid in clarifying the RM and drop the reader's attention to the required tools/mechanisms to implement a SSO architecture for Hadoop 3.x clusters. The SSORA is derived from the core concept of cutting-edge access control frameworks and Hadoop best practices.

SSORA affords a binding among the external clients and the Hadoop services in a federation Hadoop platform. It enables the internal entities (e.g., NN and DN daemons) to verify these bindings through the use of a central policy authorization and several distributed policy agents. The proposed architecture arrangement will enable users of distributed clusters (represented by various NNs) to be authenticated to interact with each other in a secure mode.

Before addressing the SSORA and federation access pattern in Figure 5.6, there are few definitions to guide and facilitate the discussion:

- Apache Knox is utilized as a unified access point of the federation service.
- Hadoop daemons and services (internal communication) are authenticated using Hadoop core capabilities, i.e., Kerberos principals and their access granularity is managed using the POSIX ACL. For simplicity sake, these two processes shall be called tickets now on in the discussion.
- Knox authenticates clients (admins may set an LDAP group), and Apache Ranger applies their access granularity (alternatively may employ Apache Sentry). WebClient secure service calls and data access will be called sessions. Finally, the granted access control (authentication and authorization permissions) shall be called certificates.

In a federation environment, it is often the case that a client of a particular HDFS cluster (e.g., NN1) wishes to utilize services offered by another administrative domain (e.g., NN2). In such cases, the user needs to be authorized by the CPA to reach the remote domain. The necessary steps required for a given BDI to access the provided SSO federation are described subsequently (see 5.6):

1. The Client asks to access the Hadoop cluster, interface with the cluster gateway (Knox) using his identity, i.e., username/password (which initialized by the admin). If the external client (WebClient as in a cloud model) wish to upload new data to the Hadoop data lake, the gateway may ask to secure the connection via TLS communications over the public network.

2. Knox cannot find a valid session for the access request (for first time registration) and start establishing a secure session with Ranger server as the Central Policy Authority CPA (could be an ACL in a simple Hadoop architecture within the NN daemon) on clients behalf, thus it requests the CPA to authorize based on the chosen identity.

3. The CPA validates the proof of authentication and issues a new security certificate to initialize his permissions (get the authorization level access). The certificate is forwarded to the gateway interface and performs a login if needed.

4. Access certificate containing the ClientID is issued for valid users and is signed with the CPA private key. The client receives the certificate and adds it to his job/operation submissions as an embedded proof of access. This certificate provides an abstraction for combining any number of authentication and authorization systems. Also, it accelerates scaling the IAM to a vast number of participants running on different clusters (under the federation umbrella) with various Hadoop services.

5. The policy repository is updated after starting the new session, and the authorization level access is stored. Thus, the CPA can always issue new security session without the need for storing the client identities, permissions, and logs locally. Policy agent will keep listening to new authorization session.

6. For auditing purposes and security analysis, the CPA manage user activities (at the service level) by writing audit logs in a common repository. These log information files are monitor based on either users and group activities or based on timeline structure (for each entity) via TimelineClient on the timeline server.

7. The CPA centralizes the access control and shares the allowed user/calls via a push mechanism to the Authorization Enforecore (AE), which is, typically, the NN in the HDFS architecture. Hence, eliminating the need for policy creation via SLA and ACL

on each AE that could be dozens (even hundreds) in a federation infrastructure. This approach, indeed, minimizes the access discussion and communications to the minimum within a large Hadoop 3.x environments and Multi-tenant BDaaS clouds.

8. The trust relationship between the client and AE is established, and his requests are authorized to proceed. Other security layers may apply (e.g., an internal TLS and files encryption).

9. The AE forwards the client request to the framework endpoint (DNs), embedding the security certificate obtained from the CPA. The policy agent periodically caches each user authorization level of access and permissions to the common blocks storage at the DNs local disks. Even that data is stored within a unified data storage, this approach enforces access discussion for each BD pool (represented by NNs), before accessing the data.

Steps from 10 to 12 refer to Hadoop local entities (daemons and processes) access control. This process includes registering DN by forwarding access tickets, and internal communications among containers (as for task status and heartbeats). DNs required proof of authentication using Kerberos principal and keytab file location. Meanwhile, authorization access enforced using Hadoop core access control, i.e., ACL. Fault tolerance is managed using erasure coding, and other security layers may implement (e.g., an internal TLS and encrypted files) as well. The policy agent caches each DN ticket information to the unified repository to update the NN cluster status.

4.4.2 Multi-tier access model

The multi-tier BD federation is a four-tier architecture as illustrated in Figure 4.4. However, access hierarchy in general has at least three access layers. A first-tier; system gateway with a plurality of HDFS clusters (represented by NNs) that connected to a central HDFS federation setting. Second-tier; access logic that connects the clients' requests to the DNs. It acts as a policy enforcer that handle the allow/denial discussions and establishing the sessions. The data access is bound to the third-tier (low-level data access) via data buses involving the predefined access policies.

A high-level data access layer may be connected to the tertiary data layer in which clients calls requires authorized session. This session connected to a particular policy agent that provided access control with means for detecting unauthorized operations. The data and calls

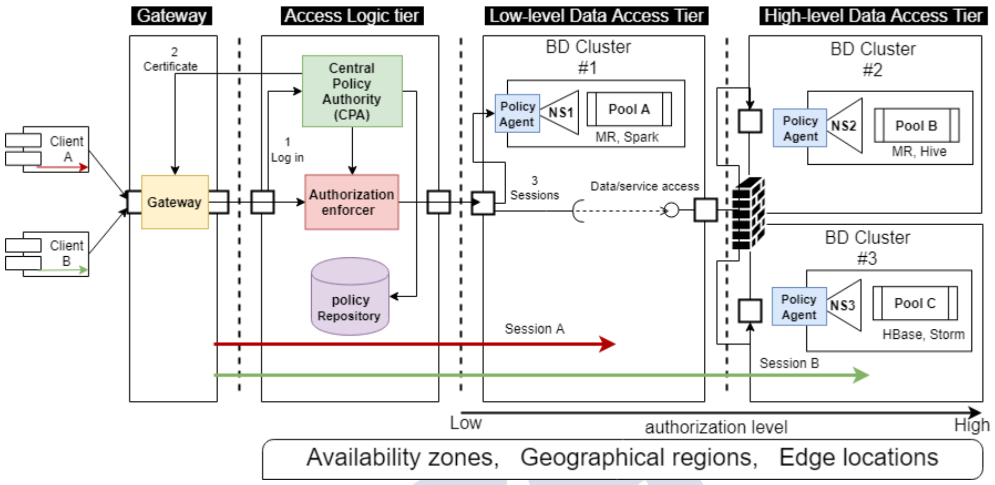


Figure 4.4: Multi-tier BD federation architecture component diagram.

traffic may remain behind a firewall in the fourth-tier. This way connects the DNs to several NNs buss as a transceiver that provides an additional layer of control security. This hierarchical access model, indeed, acts as a multi-tier authorization approach that assists in applying role-based access control [170] mechanisms within both BD federations and multi-tenant deployment environments. In particular:

Tier-One, gateway access: external client identification and cluster access;

Tier-Two, access logic discussion (assign authorization): ClientID passed to the CPA for validating to contact NNs and session issued. If successful, the CPA responds with a certificate for the NN server and establish a session. This certificate contains the clientID and a copy of the session key, which used to communicate with the authorized cluster entities.

Tier-Three, low-level data access specifies clients (could be all clients that pass the gateway) authorization level access and permissions for each NN within this layer (defined by the admin) and reaching the specific stored data in the DNs which belongs to that NN. This layer includes submitting jobs and querying status, etc., upon receiving certifications;

Tier-Four, high-level data access: designate the authorized clients (could be a private list of the previous tier) for the NNs within this layer and establish the session to enable the certified client calls.

In the component diagram in Fig. 6, the gateway is normalizing the authentication process by abstracting the ticket exchange of different frameworks. This capability is resulting in a universal authentication platform that scales the access control process in a multi-tier architecture. Both Ranger and Knox daemons may represent the IAM. The gateway contacts CPA to validate the client session tasks and to set up his level access. A different access level may be attached to different users. In the previous example, client A has access at the low-level data access tier only while client 2 yield high-level data access. This BDI imposes fine-grained data access among the clients. In the earlier example in section 2.1, a client could have access to the archival data that allocated in NS1 (tier-3).

Meanwhile, the current data manage its metadata at NS3 in tier-4, which required high-level data access session. Without the access certificates, each entity would need to authenticate and authorize with other entities on its own. On the contrary, by employing the FACRM components, it only needs to combine a central authority that issues an access certificate (based on admin policies) that applies to every entity within variant access tiers.

4.5 Authorization Level Classification

HDFS has supported a permission model that control distinct user classes with a restricted list of control roles in the ecosystem services. This way, security is sufficient with different levels of authorization access for each user-class that allow/deny the service-call based on the predefined levels of access. HDFS, therefore, enforces permissions during user connection to DNs (e.g., access file system) by the user class and then assign roles to secure files and directories. Additionally, HDFS operations rely on checking the user permission of all components of a path, which requires traversal access. Each user service-call that reaches HDFS has a two-part identity the user name and groups list.

To allow a service-call, it demands that the owner has the required permissions, which granted by file/job ownership, group membership or other authorities (e.g., ACL). For instance, a file owner (creator by default) service-call will grant an inclusive authorization level of access (read, write, execute) if his UserID matches the owner permissions. Else if the user is not the owner but a member of the groups' list, HDFS checks the authorities of that group as a non-owner class with a limited level of access (could be read-only). If a permissions check fails, the client operation fails as well.

As missioned earlier, authorization control applies at the whole path components. Hadoop checks permissions hierarchically in HDFS files-level. Further, the path components can describe as (i) Parent: the parent directory of the requested path. (ii) Ancestor: the last existing element of the inquired path. (iii) Final: the final component of the requested path [171]. Meanwhile, directories paths are defined as sub-tree (directory and child sub-directories) path recursively. Changing the data owner requires administration authorization level access. This process can also set to a superuser upon a policy. However, a user may change the group and, still, be the owner of the data, if he is a member of the specified group.

Next, we Define the different user-classes and their roles in a Hadoop cluster:

- `User`: Regular system client with limited access level, only submit jobs and query results of his data access over a predefined availability zone (NN). The user may belong to one or more groups.

- `Superuser/owner`: The user who creates the file (upload data) in the cluster and user with an authorization level of access to these files. The superuser may act on behalf of another user by impersonating (based on Kerberos credentials) their ID [172]. In general, many scenarios can be applied in this category. First, Hadoop cluster with a simple authentication policy (without secure mode), in this case, any user logs as a superuser (no IAM required) with a complete access level of authorization. Second, the user registers with the Kerberos authentication protocol in a secure mode. Data owners (the user who submitted the job) hold superuser privileges over their tasks within the cluster (e.g., he may cancel the job). However, the system admins may implement ACL at the authorization level to mitigate the security risks using the POSIX ACL model over both the files and directory using a single API gateway-access for all ecosystem components [171]. Even though, superuser does not control the underlying infrastructure but has limited control (based on the SLA policy) of select networking components (e.g., virtual networks) and configurations.

- `Administrator`: The one who sets the policy for other users of the system. With administrative privileges, the admin has absolute power over the use of the services and components. The Hadoop administrator daily tasks include correctly setting the configuration files, implementing secure ACL and SLA policies, and managing the underlying resources. These underlying capabilities include hardware, network, and runtime environment with arbitrary software that includes operating systems (OS), virtual machines (VM) and containers. Delegations allow the administrator to give another user the ability to perform actions on their behalf.

Table 4.1: Examples of authorization level classification.

Role	Example

READ	`HDFS fs -ls <args>`
	`HDFS fs -cat/-get <file>`
	`HDFS dfs -count [-q] <paths>`

WRITE	`HDFS fs -put <dir>`
	`dfs -copyFromLocal <localsrc>`
	`intra-node disk balancing`

EDIT	`hadoop namenode -format`
	`Chmod/Chmown <file>, setPermission <path>`
	`delete <Path>, removeDefaultAcl.`

CONFIG	`Start/stop-dfs.sh`
	`hdfs dfs -setReplication [-R] [-w]`
	`setAcl, <numReplicas> <path>`

INTERNAL	Heart beat, ApplicationMaster to NodeManager, NM to container. Data pipelining with Hive, HBase, etc.
UNDERLY	administrative commands edit etc/hosts, directory foo machines on/off, etc.

- `Daemon`: The processes that started under Hadoop and run in the background to be functional. The main Hadoop Yarn and HDFS daemons are ResourceManager (RM), NodeManager (NM), NameNode (NN), and DataNode (DN). Besides, there can be secondary NameNode, standby NameNode, Job HistoryServer, etc.

Each Hadoop command (operation) can belong to one role of the authorization levels listed in Table 4.1. For instance, when a user runs `hadoop job -list` it authorizes the connection requests at the READ authorization level. Meanwhile, an attempt with an unauthorized access level, e.g., `hadoop job -kill jobid` that does not belong to job owner (nor he is an admin) will be denied. A complete list of HDFS methods that may categorized based on our classification can be found in [173]. Besides, the relation between different levels of access and user-class in Apache BD cluster is illustrated in Table 4.2. It worth to mention that those roles are not alternatives for the general Hadoop file permission values (read, write, execute). On the contrary, those roles represent a categorization of the HDFS operations by aggregating them based on their role. This categorization aims at setting each operation/command to proper access level state. In an operational HDFS environment, any command may categorize to:

- **READ**; permission is required to read or list the contents of DN, querying information, and check job status and results.
- **WRITE**: permission is required to add data or append it to DN, submitting jobs to NN, and updating information in HDFS.
- **EDIT**: permission is required to access data blocks, delete data, cancel a job, formate the NN, and perform data balance.
- **CONFIGURATION (CONFIG)**: permission is required to access runtime variables, paths, add/remove Hadoop features, change the scheduler (e.g., Fair), etc.
- **INTERNAL COMMUNICATION (INTERNAL)**: permission is required to access daemon to daemon communication, as for task updates and cluster resource status, etc.
- **UNDERLYING Capabilities**: permission is required to access infrastructure variables, add nodes, OS, VM, Network, Hardware.

We characterize the conceptual classes of Hadoop Yarn cluster users and the HDFS operation roles in a federation architecture. From these specifications, we try to characterize each service-call with constraints into five main classes to refer as authorization level classification. This categorization deliberates, indeed, as the foundation for future secure API design and standardization work. It also worth to mention that Hadoop 3.x advanced with intra-queue preemption that priorities applications and users limits on a queue (within the capacity scheduler [174]).

4.6 Related work

Security of BD deployment architectures and Hadoop service trust have always labeled as a research concern. Several research papers discuss the Hadoop ecosystem privacy and access management [175, 176, 177, 178]. Although the access control requirements and privacy

Table 4.2: The relation between levels of access and user-class

	Read	Write	Edit	Configuration	Internal	Underlying
User	√	√				
Superuser/Owner	√	√	√			
Administrator	√	√	√	√		√
Daemon	√	√			√	

analysis of Hadoop frameworks have addressed in the literature [179, 180]. In [175] and [176] the authors formalize the access control features of Hadoop core capabilities, as well as other associated frameworks. They extend their work with HeAC [177], a multi-layer access control model that represents a role-based abstraction of both Apache Ranger and Sentry access control. A framework for data-intensive application fine-grained policy enforcement proposed in [181].

Standardizing the security development schema of secure systems was first reported in [182]. This unifies practices and languages for modeling security and access control among different implementers. A reference model for developing cloud applications was reported [183], besides, a survey that supports federated access control [184]. Barker, S. propose a meta-model and best-practices of access control modeling [185]. Both Barkers' work and this research aim at unifying access control models by drafting a reference model. However, herein we address the modern Hadoop 3.x status given the broad diversity of existent access control uniformity associated with a federation environment. Also, this work is distinguished by addressing the HDFS federation access control challenges and maps cutting-edge frameworks to that problem domain.

4.7 Summary and Open challenges

In this section, we present a BD Federation-oriented reference model for the secure development of access control solutions within Hadoop clusters. The proposed model is a generic high-level conceptual metamodel (called FACRM), which aims at formalizing Hadoop 3.x core access controls capabilities and highlighting some of Hadoop's top-level security projects. It, also, aims to successfully implementing a unified and secure BD solution over on-premises and cloud-based deployment architectures in favor of BD applications and frameworks. The

FACRM supports myriad access control options across a variety of policies, users, and frameworks in pursuit of modern BD infrastructure. This study also represents an in-depth analysis of Hadoop IAM ecosystem and draft current access control limitations within a federation deployment architecture. We validate the use of our proposed RM using two use cases; SSO federation architecture and multi-tier access control architecture for the security of BD deployment. Finally, it categorizes the authorization levels that afford comprehensive coverage of Apache BD IAM and related manipulation languages.

Leverage recent proposed independent management mechanisms of policies enforcement [186] to combine cloud and edge federations with role-based access control could label as future research. It is, also, expected to adopt the FACRM use cases to be implemented within our BD opportunistic and elastic resource allocation (BigOPERA) platform. BigOPERA architecture (its prototype proposed at [3]) combines the computing power of available (non-dedicated) high-throughput resources to Hadoop 3 dedicated cluster using Docker containers as worker nodes. Those non-dedicated containers tend to be more vulnerable, which require robust IAM approach to maintain security threats to a minimum.

This work also, intends to propose a trustworthy and Federation-oriented model for authenticating users and services over a Hadoop 3.x. platform. The main goal of this model is to provide a SSO gateway based on Hadoop access control primitives and Apache Knox. A SSO architecture can be managed using Kerberos with Lightweight Directory Access Protocol (LDAP), which is enabled by setting `Hadoop.security.group.mapping.ldap.url` to true. Alternatively, implementing a single gateway that handles client access management behind a firewall can be achieved by employing Apache Knox. Knox will allow/deny users to access the ecosystem services before interacting with the Hadoop cluster. Following to user verification using Knox gateway, Knox will use its Kerberos principals to confirm, securely, with other Hadoop services and daemons.

It is expected to adopt the SSORA within our BD opportunistic and elastic resource allocation (BigOPERA) platform. BigOPERA architecture (its prototype proposed at [3]) combines the computing power of available (non-dedicated) high-throughput resources to Hadoop 3.x dedicated cluster using Docker containers as worker nodes.

Examples of different open challenges and research directions include:

Scalable Authorization Despite the availability of several adapted security mechanisms and techniques that have been developed for BD security, more research activities are still needed to establish scalable security models and paradigms that must be driven by BD specifications and requirements. Also, new security abstractions for BDF are still needed to simplify the task of identifying the unimproved gaps. Also, leverage recent mechanisms of policies enforcement [186] to combine BDF with ABAC [177] could label as future research.

Fine-grained Authentication Management The need to develop active and dynamic models (policies and standards) for BDF is inevitable. The challenge here is to extend the previous approaches with new policy editing and presentation tools for flexible and extensible data access, i.e., policies that extend ABAC with stateful access sessions and mutable attributes (i.e., characteristics that change dynamically during the session). This extension is mainly invaluable to advance the authorization mechanisms of the multi-tenant data lake architecture as well as the latest BDF model. Tackling this issue can be achieved by deploying policy templates that extend the primary ACL (or any access decision enforcer) as a specialization of the XACML V3.0 standard.

Data-centric security This requires restricting the BDF access to only those authorized by succinct gateway. However, the current state-of-the-art BD-based encryption technologies are around transparent data encryption. This takes place at the file-level (for data at rest) and does not protect data-in-transit or data-in-process (not even the metadata). Attribute-based Encryption (ABE) in conjunction with usage control approach and ABAC can be labeled as a future direction for this issue. It could provide a complete end to end encryption, i.e., including data traffic between the store and application.

Providing new solutions in all those research directions will promote innovative BDF solutions with advanced security features.

CHAPTER 5

A UNIFIED BIG DATA FEDERATION ACCESS CONTROL AND AUDITING

[5]

5.1 Introduction

Apache Hadoop [6], the BD landmark, has become a large-scale data analytics operating system. The large community behind Hadoop has been working to improve its stack to meet the increasing demands and requirements of BD. Enterprises across all major industries have adopted Hadoop due to its capability to store and process an abundance of new types of data and leverage modern data architecture. With a broad spectrum of both structured and unstructured workloads, Hadoop abstracts the computing resource management, task scheduling, and data management, while maintaining a satisfactory level of security and isolation.

The Hadoop distributed file system (HDFS) [187] is typically deployed as part of a large-scale Hadoop platform to support commodity hardware and accommodate different processing frameworks. It is utilized to handle data management and access to the Hadoop ecosystem using a master/slave architecture. It is also successfully employed by several distributed systems and can be used by different resource schedulers as a data storage system, e.g., HTCondor [188] and Spark [189]. IAM in the HDFS ecosystem can be defined as the set of tools and mechanisms that enables end users and applications to interact securely with system core functionalities, thus ensuring appropriate access to data across the cluster. This security discipline can be separated into three abstraction layers: identification and access control, authentication, and authorization.

As more services and users have joined the Hadoop federation portfolio in pursuit of a scalable BD hub, access control has become increasingly critical. One of the main obstacles in the development of an adaptive access control solution for BD platforms is the lack of a standard model to which access control rules and the associated enforcement monitor can be bound. A recent study [179] indicates that BD, as an emerging research trend, lacks standardized models for unifying user/application access to resources, services, and data. Moreover, Hadoop's primitive configuration lacks any classification mechanism that improves metadata governance or facilitates auditing procedures. Therefore, formalizing the IAM features of Hadoop 3.x, in addition to emphasizing the complexity of securely utilizing their core functionality, is becoming a research interest. This trend serves as a form of knowledge capture by mapping current technologies of concern and formulating the latest Hadoop capabilities related to access control frameworks and audit log management.

This paper highlights the need for robust access control that handles the authentication and authorization operations within a federation scale. The paper defines the associated requirements of secure BD runtime and management of the Hadoop HDFS federation. It therefore proposes a reference model (RM) that provides a basis for building an interoperable data federation scheme that includes all of the major stages and reflects specifics in access control within Hadoop clusters using modern open-source technologies. Furthermore, this paper explains how the proposed models can be implemented using modern BD infrastructure (BDI) to meet the increasing demand for scalable access controls and digital forensics (using access log analysis). The key question we will be asking to address the IAM challenge is as follows: "How can we create a dynamic reference model that is compatible with the foreseen BDI scenarios in a federated setting?" We address this question by employing the proposed RM in a novel access broker that provides a single sign-on environment. The proposed BD Federation (BDF) broker is an access control logic component to securely connect the external users with the Hadoop cluster gateway. The work presented in this paper may be employed in a Hadoop federation environment across multi-tenant BD clouds, as well as within on-premise data centres.

An essential application of the proposed solution deals with security auditing and analysis of audit logs of BD operations. These log files contain detailed information about all access call activities regarding BD processes and infrastructure to be protected by centralizing the access control of the data lake, BD services, and underlying resources in a unified access broker pattern. This broker abstracts the identification, authentication, and authorization discussion within large-scale data analytics. This layer may also afford granular insights into pieces of information by performing security and risk assessment, tracking data pipeline audit logs, and examining behavioural analytics to meet their compliance and governance demands within the Hadoop 3.x platform.

The rest of this paper is organized as follows. Section 5.2 provides an overview of the Hadoop federation and its role in supporting BD operations. It also highlights complementary security measurements for BD frameworks and discusses the related work. The HDFS access control primitives and federation access provisioning limitations on which this study is based are investigated in Section 5.3. Section 5.4 describes the logic language for defining the RM, while we implement the proposed model in a novel access broker proof-of-concept framework in Section 5.5. Section 5.6 demonstrates the advantages of the proposed centralized audit log management, and we ultimately provide the conclusion in Section 5.7.

5.2 Hadoop Federation

As the size of a Hadoop cluster increases, the pressure on the NameNode and the ResourceManager increases. To alleviate this problem, Hadoop has introduced both HDFS Federation [190] and YARN Federation [191]. HDFS Federation consists of the usage of multiple independent NameNodes, where each NameNode is responsible for managing a subset of the whole namespace. To understand how this federated architecture works, we need to explain how the single HDFS is designed. HDFS is composed of two main layers:

- The namespace layer, which runs in the NameNode and is in charge of storing all information related to directories, files, and blocks (creation, deletion, modification, listing).
- The block storage service, which supports low level block-related operations and contains two parts:
  - Block management, which runs in the NameNode and is responsible for block creation, deletion, modification, replication, and location.
  - Physical storage, which runs in the DataNodes and is responsible for storing the data blocks and providing read/write access to them.

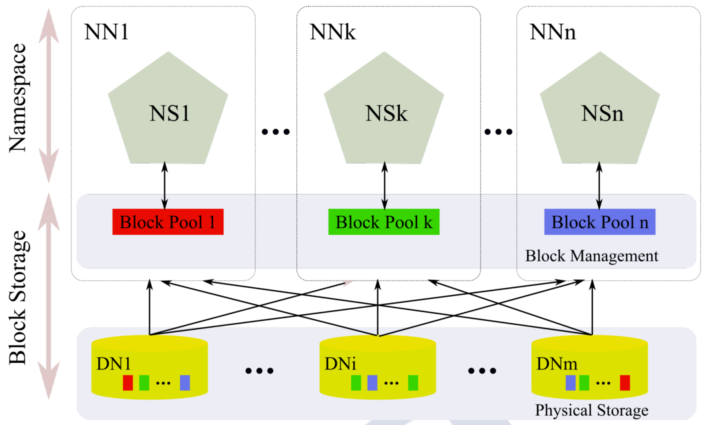


Figure 5.1: HDFS Federation Architecture with n NameNodes (NN) and Namespaces (NS) and m DataNodes (DN)

Whereas physical storage can be easily scaled horizontally, simply by adding more DataNodes, previous versions of HDFS only allowed the namespace layer to be scaled vertically, as it runs on a single NameNode. The HDFS federation removes that limitation, allowing the namespace layer to be scaled horizontally. It uses a federation of multiple NameNodes that are independent; i.e., they do not require coordination with each other. Each NameNode is assigned a set of blocks, which is called a block pool. Different blocks in a Block Pool can live in any DataNode, and so each DataNode must register with all NameNodes in the cluster, sending them periodic heartbeats and reports about block status. A schema of this architecture is shown in Figure 6.1.

However, on Jan 16, 2019, Apache Hadoop released its 3.2.0 stable platform, while the first stable 3.x line was released on Apr 6, 2018. This release incorporates several significant enhancements over the previous primary release line. A federated architecture has also been proposed for Apache YARN. This approach allows YARN to scale to tens of thousands of nodes. In this architecture, a large YARN cluster is split into a set of sub-clusters. Each sub-cluster has its own ResourceManager and compute nodes. From the point of view of applications, the federated cluster is still seen as a single huge YARN cluster, and tasks can be scheduled on any node of the cluster. The federation system is responsible for negotiating with the resource managers of the sub-clusters and providing resources to each application. YARN Federation functionality relies on reliable connectivity across sub-clusters to deploy atop the HDFS Federation. A federation architecture across data centres among several physical confines requires further investigation.

5.2.1 HDFS federation

In this section, we present a brief overview of HDFS federation architecture and highlight the Apache Hadoop security features.

HDFS architecture

HDFS [167] is a distributed and open-source file system designed to meet the rapidly growing demands of large-scale data management and access. The HDFS is composed of two primary daemons: (i) a single NameNode (NN) that is deployed at the cluster master node and (ii) several DataNodes (DNs) running at the cluster slaves (usually one per node). The NN runs the namespace process, which manages the file system information and regulates access to files using a conventional hierarchical organization. In addition, it tracks block locations and numbers and writes log information files for auditing purposes. Internally, a file is partitioned into multiple data blocks that are placed into various DNs to be stored in local disks (block storage). These blocks are replicated for fault tolerance over several DNs from different racks (if possible). The NN makes all decisions regarding replicas and periodically receives a static heartbeat; the default period is every 3 seconds. The NN checks the expiry time report of a heartbeat every 200 seconds (as a default timeout). When a new file is updated, the NN places replicas of the file blocks in different nodes and racks (if available).

Conventionally, the Hadoop cluster runs several DNs, but it has only one NN (and one namespace) for all DNs (HDFS High-Availability allows running two NNs when working in active/standby mode). The DNs can be scaled both vertically (by adding more resources to the nodes) or horizontally (by adding more nodes to the cluster). The NN, however, may only scale vertically, which means that one needs to add more resources (CPUs and RAM) to that NN server to serve more DN metadata. Notice that metadata are preserved in memory for minimizing latency and enabling faster retrieval. This approach causes a single point of failure and, hence, limits the number of blocks, files, and directories maintained on the file system. The HDFS Federation has been introduced to cope with this issue.

To enable a universal block storage layer, Hadoop performed separation of namespace and blocked storage [168]. A federation BD environment, through multi-independent namespaces for block management and a common block pool for data storage, improves scalability and isolation of Hadoop operations. Therefore, by loosening the tightly coupled block storage and namespace, each DN registers with all the NNs in the cluster (this increases the authentication requirement). These DNs send periodic heartbeats, block reports and handle commands from the NNs. This allows horizontal scaling of the NN and enables the aggregation of geo-distributed Hadoop clusters. This feature directly enhances throughput by adding more access enforcers (typically, NNs in HDFS architecture), which improves read/write operations.

Moreover, the HDFS federation services high-intensity BD applications that block vast resources on the NN by distributing them among different namespaces. However, this imposes authentication and authorization challenges, as well as security concerns, which we address in the next section. For instance, a federation cluster may improve the query performance in a Hive framework – as Apache Hive manages data in partitions within different tables and locations. This setting can store various tables in separate namespaces, or even save the table partitions in different namespaces. In principle, this optimizes load balancing across multiple namespaces (e.g., one for archival data and another for current data), which reduces each namespace load and improves the application isolation.

5.2.2 Complementary security measures for HDFS federation

Figure 6.2 presents a security layered abstraction of the engineering BD system, which involves several essential functions. The first three layers, identification (ID) and access control, authentication, and authorization, are combined, representing the IAM processes of the Apache Hadoop cluster and the objective of this paper. The other security layers (data governance, integrity, confidentiality, and security auditing) are briefly presented in this section to perform a comprehensive security discussion of a BD environment.

- **Access control** is a service gateway to securely and efficiently communicate with the BD federation. This layer verifies the external client's access to the system using the user identification (ID) and passwords. Every client username and IP address must be in the host file or in a DNS table, and must match the client-given password. This process may also include Apache Knox [192], a unified gateway framework for Hadoop services and ecosystem that can be utilized as a single-sign-on (SSO) gateway.

- **Authentication** is the act of confirming authentication access to the Hadoop services and HDFS data (after user log-on to the cluster), i.e., the process of actually determining the client identity. In a non-secure Hadoop mode, this layer is disabled and internal

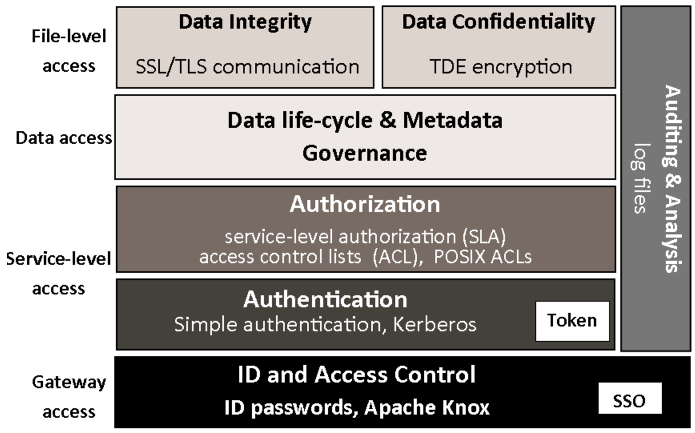


Figure 5.2: Apache Hadoop security layer abstraction.

entities, i.e., clients (confirmed by the host OS), application, and ecosystem, interact directly with the Hadoop services. However, Hadoop secure mode [193] enables an authentication mechanism to verify that an entity is what it claims to be using the Kerberos protocol (authentication based on tokens). Each Hadoop service and user must be authenticated by the Kerberos keytab file (binary containing the information needed to log) using Hadoop tokens to initialize trust between a client/application and the HDFS (more details in Section 5.3.3). Authentication for access to the Hadoop services web console requires enabling the HTTP SPNEGO protocol as a backend for Kerberos credentials [194]. Thus, preventing the stored data in HDFS from unauthorized access is applied both to all clients accessing the cluster and to any service claimed to be part of the cluster.

- **Authorization** is the process of defining access rights which an entity (service, daemon, or client) can perform to the given service. It manages access in the context of a specific service, resource, and data functionality provided by the cluster. Hadoop service level authorization (SLA) manages the fundamental set of permissions, such as defining the users and groups who are authorized to make service calls (e.g., data access) to that service. The call will pass the authorization check only if the user making the call belongs to an authorized service entity. It also provides fine-grain access control (table, column, and file levels) by enforcing the access control list (ACL), i.e., consistent policy administration across all Hadoop ecosystems [195]. ACL combines three elements: effect (allow or deny), action (e.g., data access or execution), and resources (e.g., NameNode1, Hive table). In principle, authorization is considered to be the final IAM layer in a Hadoop security abstraction, which means that no additional security mechanism is required as an IAM intention for an authorized client. Nevertheless, data encryption both at rest and in transition is still required, in addition to security analysis, auditing, and metadata governance, which represent high-level security services when combined as in Figure 6.2.

- **Data Governance** is a capability that ensures adequate manageability of data through the complete lifecycle (i.e., data at rest, movement, and processing). This layer includes several dimensions, including classification (labeling and description), source tracking, and quality across data sources. Providing a typical store for exchanging metadata tags and attributes among the Hadoop stack can be achieved using Apache Atlas [196] and Apache Solr [197] for defining data types and fields using full-text indexing and querying techniques.

- **Data Integrity and Confidentiality** is a capability to ensure adequate consistency and accuracy of data-at-rest as well as in-transit. This security layer includes validity and recoverability approaches, as Hadoop 3.x utilizes erasure coding for fault tolerance [**?**]. However, aiming for confidentiality, HDFS implements end-to-end encryption with so-called transparent data encryption [198]. This HDFS encryption occurs at the file level of on-disk data and is stored as NN metadata. On the other hand, Hadoop wire-security (such as for data transfer between Web-console and client) is managed via SSL/TLS for HTTP communications.

- **Security auditing and analysis:** The aggregation of log files and reports provides a robust audit capability within different components of the Hadoop ecosystem. This layer may also afford granular insights into pieces of information by performing security and risk assessment, tracking data pipeline audit logs, and examining behavioural analytics to meet their compliance demands within Hadoop.

5.2.3 Related work

The security of BD deployment architectures and Hadoop service trust have always been labelled as research concerns. Several research papers have discussed the Hadoop ecosystem privacy and access management [175, 176, 177, 178, 199, 200, 201, 202, 203, 204]. The access control requirements and privacy analysis of Hadoop frameworks have been addressed in the literature [179, 180]. Recently, Gupta et al. [175] and [176] formalized the access control features of Hadoop core capabilities, as well as other security associated frameworks including Apache Ranger and Sentry. They extend their work with the HeABAC [177] model, a multi-layer attribute-based access control model [205, 206] that provides fine-grained authorization policies for Hadoop. Ulusoy et al. proposed an approach for fine-grained access control authorization for MapReduce systems in [181, 207]. Big data privacy issues are also well addressed, and novel solutions have been proposed in [180, 208, 209, 210], in addition to an SSO framework for Hadoop services in [1]. Colombo et al. conducted a comprehensive study of big data technologies, including access control requirements, state-of-the-art and future trends, in [211, 179, 178]. One of the earliest works by Kulkarni [212] targets wide-column NoSQL databases which support content and context-based access control policies at different levels of the data model, such as row or column. Another work on Cassandra datastore [213] presents a cryptographic enforcement of RBAC policies. Predicate [214] and second level encryption [215] have been used for defining an efficient RBAC enforcement scheme operating within the Cassandra distributed architecture.

Research has also been presented in the field of Data Stream Management Systems (DSMSs) [216, 217, 218], wherein a framework, called FENCE, has been proposed in [216], which supports continuous access control enforcement. Complex event processing (CEP) systems represent the evolution of DSMSs [219, 167], and have been used in contemporary applications including IoT and Smart Cyber-physical systems [220]. Researchers from the National Institute of Standards and Technology (NIST) [221] have presented a general access control model for BD processing frameworks which introduces the novel notion of chain of trust among entities to authorize access requests.

Access control of data and resources from multiple sources within centralized computing systems has been a subject of intensive studies in the literature. Context-awareness using fuzzy logic conditions as access control approach has been proposed in [222] to address the dynamic outsourcing environment on the edge of the network. Aiming to protect the redundant data stored over the cloud, an approach by Zhou, Yukun et al. [223] supported flexible access control with revocation. Adopting a proxy re-encryption policy to update the process to the outsourced cloud was reported for BD deployments [224] to support ciphertext re-encryption. Controlling access to sensitive BD, e.g., healthcare records, using the quantum mechanical equivalent of digital signature was introduced [225], along with an unaddressed challenge of the data transfer process. Other than the implementations mentioned above, the BDs of time series (e.g., time-series databases) are required to be associated with an encrypted timestamp, and Noury, Amir, and Morteza Amini in [226] proposed an access and inference control model to enforce time and value-based constraints over the hierarchical time series data.

Standardizing the security development schema of secure systems was first reported in [182]. This standardization unifies practices and languages for modelling security and access control among different implementers. A reference model for developing cloud applications was reported [183], along with a survey that supports federated access control [184]. Barker, S. proposed meta-model and best-practices of access control modelling [185]. Both the work of Barker and this research aim at unifying access control models by drafting a reference model. However, herein, we address the modern Hadoop 3.x status given the broad diversity of existent access control uniformity associated with a federation environment. Additionally, this paper distinguishes itself by addressing the HDFS federation access control challenges and maps cutting-edge frameworks to that problem domain.

5.3 Access control in HDFS

A client needs to perform tasks through the NN as a central policy enforcer for the HDFS. The NN receives the client call and allows it to reach data files which are stored in local disks via the DN pool. HDFS applications need a write-once-read-many access model for files. Once a file is created and written, it cannot be modified without an authorization access level policy. A secure IAM model must ensure that entities (clients, processes, and daemons) are who they claim to be (authentication) and that they have permissions to perform the requested operation (authorization).

5.3.1 HDFS replica selection

It is important to direct the reader's attention to "how it works" before presenting the access challenges within the federation settings.

The HDFS cluster responds to client calls via the closest block replica to the reader node, aiming to keep data movement and read latency to a minimum. Thus, the preferred replica is the one stored locally (if available) to the reader node or the replicas of the same rack to

Table 5.1: HDFS data access priority scheme.

Cluster Number	Rack Number	Node Number	Path	Priority
Cluster 1	Rack 1	Node 1	Local	First
		Node 2	Node 2>Rack 1	Second
	Rack 2	Node 3	Node 3>Rack 2>Cluster 1	Third

reduce bandwidth consumption (see Table 5.1). If the HDFS spans multiple rack clusters, then a replica placed in the local rack is preferred over any remote replica [227]. The NN manages this fault tolerance operation by implementing a rack-aware policy (e.g., the first replica is placed on the local node, the second is located in a different rack, and the third one is stored on another node within the local rack). By analysing the DN length to the parent, the NN selects the replica placement path. Consider the example in Table 5.1 that assists in the further discussion concerning HDFS access control and HDFS Federation (section 5). A Hadoop application will use blocks locally within its current node. However, if the block has to be copied from a remote DN, it will calculate and pass the block to the clients in the order of proximity.

Figure 5.3 demonstrates the HDFS authorization model pattern with an HDFS service call scenario. At first, Apache Yarn client make an HDFS call through the NN cluster as the service level access control. The NN checks the ACL, which had been set by the administrator, and the service call is either granted authorization to proceed with the operation or to deny it. Finally, the NN writes audit log information to a local file. System administrators must set this security mechanism for each Hadoop component and execution engine (Spark, Hive, HBase, etc.). Every ACL has one NN (and only one), while the NN could or could not have an ACL. In the last case, any service call made by the user that reaches the NN will gain access to DNs (the user still needs the ID and password for the external log).

5.3.2 Apache Hadoop federation access management

A cluster with high-level data analysis, i.e., sequel operations, and different execution engines will contact its security module (local policy authority or access enforcer) to validate access control rules. Figure 5.4 illustrates a Hadoop Yarn cluster with several data access

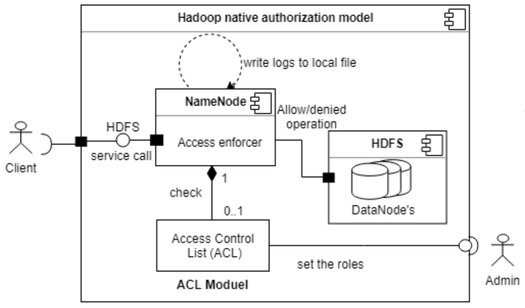


Figure 5.3: Authorization access pattern in a native Hadoop Yarn cluster.

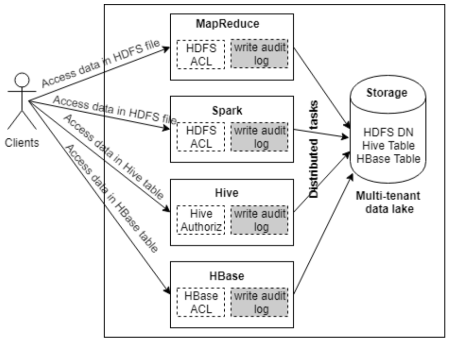


Figure 5.4: Access pattern in a multi execution-engine Hadoop cluster with no central policy service.

requests managed by different BD execution frameworks. The cluster administrators need to separately create the ACLs by specifying authorization access based on the local policy authority (e.g., HDFS ACL, Hive ACL, etc.) for each framework [195]. The NN will perform a permission check before issuing any operation; those ACLs are located and defined in the `$HADOOP_CONF_DIR/hadoop-policy.xml` file. In this case, the request will reach the local authorization module that allows/denies the operation and writes its own logs to a local file. These individual access enforcers (see Figure 5.4), coupled to file permissions and different Hadoop framework orchestration, make it challenging and time-consuming to set a secure BD environment, not to mention the new BD federation features and multi-tenant BDaaS cloud deployment architecture. A solution relies on a cross-service authorization framework that provides a centralized policy authority to store and manage security policies for multiple ecosystem components.

5.3.3 Limitations of BD Federation access control

The broad participation nature of both clients and execution frameworks that are associated with the Hadoop federation warns against security concerns. In addition to the complexity of settings, dynamically and elastically scaling client authorization and policy provisioning are challenging tasks. A central policy enforcer to manage the access discussions seems mandatory. This section outlines the inherited limitations of implementing HDFS access control in a federation setting.

Restricting access to the data stored in HDFS requires ensuring an accurate level of access by both clients and applications to the ecosystem components. In this way, by enabling Hadoop secure mode, every client, service, daemon, and process (which we refer to as entities) running within Hadoop must authenticate its membership of the cluster. The Hadoop underlying authentication service is based on Kerberos encapsulation, a leading local area network certificate-based protocol. Kerberos is a three-way protocol that requires every task to be authenticated before accessing the data. Therefore, only verified HDFS DN could register with the NN, and every task must be authenticated via the three-way handshake process before resource acquisition. This approach can cause performance degeneration in distributed clusters (e.g., think of a MapReduce application with thousands of tasks).

One solution to this problem is to complement Kerberos with Delegation Tokens (DT), a lightweight authentication process. With DT, a client or application is first authenticated with Kerberos, and DT is then applied for subsequent and iterative service calls. Thus, the cluster services and daemons are only authenticated with Kerberos, while the client's subsequent service calls and job tasks will use the DT credentials to interact with the HDFS. The use of Kerberos and DT in secure HDFS federation authentication is described in Figure 5.5:

1. Clients (and every entity) are initially confirmed to contact the NN using Kerberos authentication, and each client receives a DT certificate from the NN server to access the HDFS DNs using block tokens (access tickets) for every block with which they must communicate.

2. The client uses the access ticket for subsequent service calls, instead of using Kerberos. Furthermore, the distributed tasks use that DT certificate on behalf of clients to securely reach the NN at runtime.

3. The client and the distributed tasks can access the HDFS DN using the granted access tickets, which declare that the caller has the stated access rights to the data blocks within the job submission.

A single-sign on (SSO) architecture can be managed using Kerberos with the lightweight directory access protocol (LDAP), which is enabled by setting `Hadoop.security.group.mapping.`
to true. Alternatively, implementing a single gateway that handles client access management behind a firewall can be achieved by employing Apache Knox. Knox will allow/deny users to access the ecosystem services before interacting with the Hadoop cluster. Following user

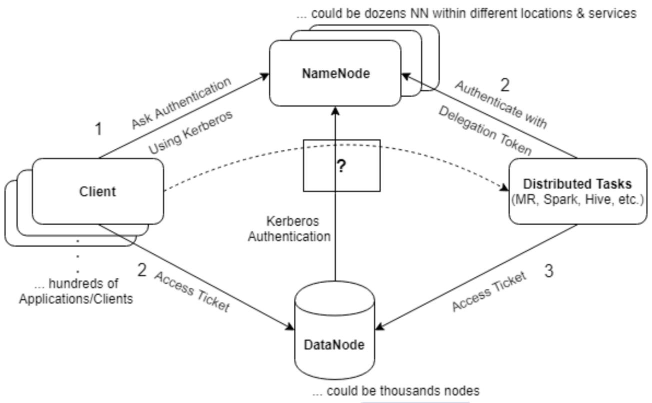


Figure 5.5: A simplified diagram of the HDFS federation authentication mechanisms.

verification using the Knox gateway, Knox will use its Kerberos principals to securely confirm access with other Hadoop services and daemons.

5.3.4 Problem definition

The previous example demonstrates a single client contacting the NN, while several authentication mechanisms must be sequentially acquired for each Hadoop entity (daemon, client, and task). The DT ticketing process considerably decreases the amount of authentication requests; however, we argue that a federation Hadoop cluster can process the login requests of thousands of entities within a short period. A federation operation may require authorization checks for different components of the path: not only the final entity (as for a multi-tier federation architecture) This serialization of authentication before issuing the task will degenerate the system throughput in a large-scale data cluster and promptly become a bottleneck. This bottleneck may involuntarily cause a distributed denial of service attack within a federation environment. In addition, information logs are stored and managed locally (by NNs) without any classification mechanism that improves metadata governance or facilitation of auditing procedures.

Relying on Hadoop IAM as demonstrated in figures 5.4 and 5.5 could present the following underlying limitations:

1. Access control complexity: Hadoop core IAM requires a client to go through multi-permission stages for each service that it is willing to contact. This results in additional

time and bandwidth consumption in a federation setting. Additionally, the fact that Kerberos/DT tickets are time-limited means that we need to re-authenticate the entities based on the given maximum lifetime. A client must request a new DT and pass it to the running process.

2. Long-running services: Due to token expiration time, supporting long-running jobs beyond the token maximum lifetime is a difficult task (e.g., aggregating logs) [228]. This involves a continual reauthentication process that requires cancelling all running tasks in order to start a new session. The new features of Hadoop 3.x need to include an adaptive mechanism for each expired token to continue to support the long-lived YARN applications service [229].

3. Communication orchestration: There could be thousands of node-to-node communications for a job, resulting in the same magnitude traffic without a centralized (third-part) orchestration service that enforces authorization policies in a fine-grain model to guarantee data security across the Hadoop cluster.

4. Data integrity: The communication channels between services still need to be secure; there is no end-to-end data encryption. This issue indicates that tokens need to be encrypted over the network (SSL/TLS may be utilized). Sniffing (sniffing attack) tokens are sufficient to incorporate an authorized entity, thus prompting a vulnerability to threaten the HDFS data.

5. Node orchestration: Time needs to be roughly uniform across cluster nodes; else, the time-limited tokens will not work. Additionally, any compromised entity may manipulate the local time to infinitely extend the ticket lifetime.

6. Audit log management: It is difficult to analyse or test against the security configurations of each entity within a federation cluster. Different NNs separate log categories from regular processing logs and store them in separate locations with varying policies of persistence.

7. Security auditing: It fails to cope with modern auditing demands of large-scale distributed clusters, as in a federation. Any secure IAM model should not only secure the access control but should be able to perform security auditing at the service level: for instance, incident reporting, behavioural analytics, and regular risk assessment of the cluster entities.

8. Access control scalability: Adding new security features, cluster entities or clients, as well as updating policies, could require a complicated (and time-consuming) process that must be configured carefully. This model might be acceptable for a relatively small setup, but not for a federation infrastructure.

5.4 Federation access control reference model

A service-oriented metamodel for access control support in BD federation platforms is presented in Figure 5.6. The Federated Access Control Reference Model (FACRM) is a set of architecture components that are associated with the security development of BD environments. FACRM also addresses many technical aspects that optimize context-specific design, which need to be clarified at different granularity levels. The FACRM is characterized by (i) ease of distributing secure access at scale, (ii) accelerated policy spawning and provisioning, (iii) providing ease-of-use horizontal scalability, and (v) flexibility for adapting to the service provider requirements and demands. Some highlighted elements will be discussed in Section 5.7.

Next, we describe the FACRM components as listed below:

IdentificationAccessManagement(IAM) Maintains the access control status of system entities within three layers of defense: ID gateway (allowed to reach the cluster), authentication (identify verification; allowed to contact the underlying services), and authorization (which access rights and permissions are held).

a- GatewayManager: a registration authority for external login and all REST APIs. It represents the first IAM stage for verifying client service calls. It could employ either simple user name/password or a third-party framework, namely Apache Knox. By leveraging Knox as the cluster access point and URL gateway, it enforces access discussions behind the infrastructure firewall. Hence, failed requests neither reach the Hadoop daemon nor operate on its entities (e.g., NN and DN). In this scenario, Knox is composed of three main components to validate the access of several connections:

i- UserID: client identification obtained via profile (name/password) and Hadoop can set the group mapping service [230]. However, resolving URL access via groups is managed via LDAP with simple password authentication using JNDI API.

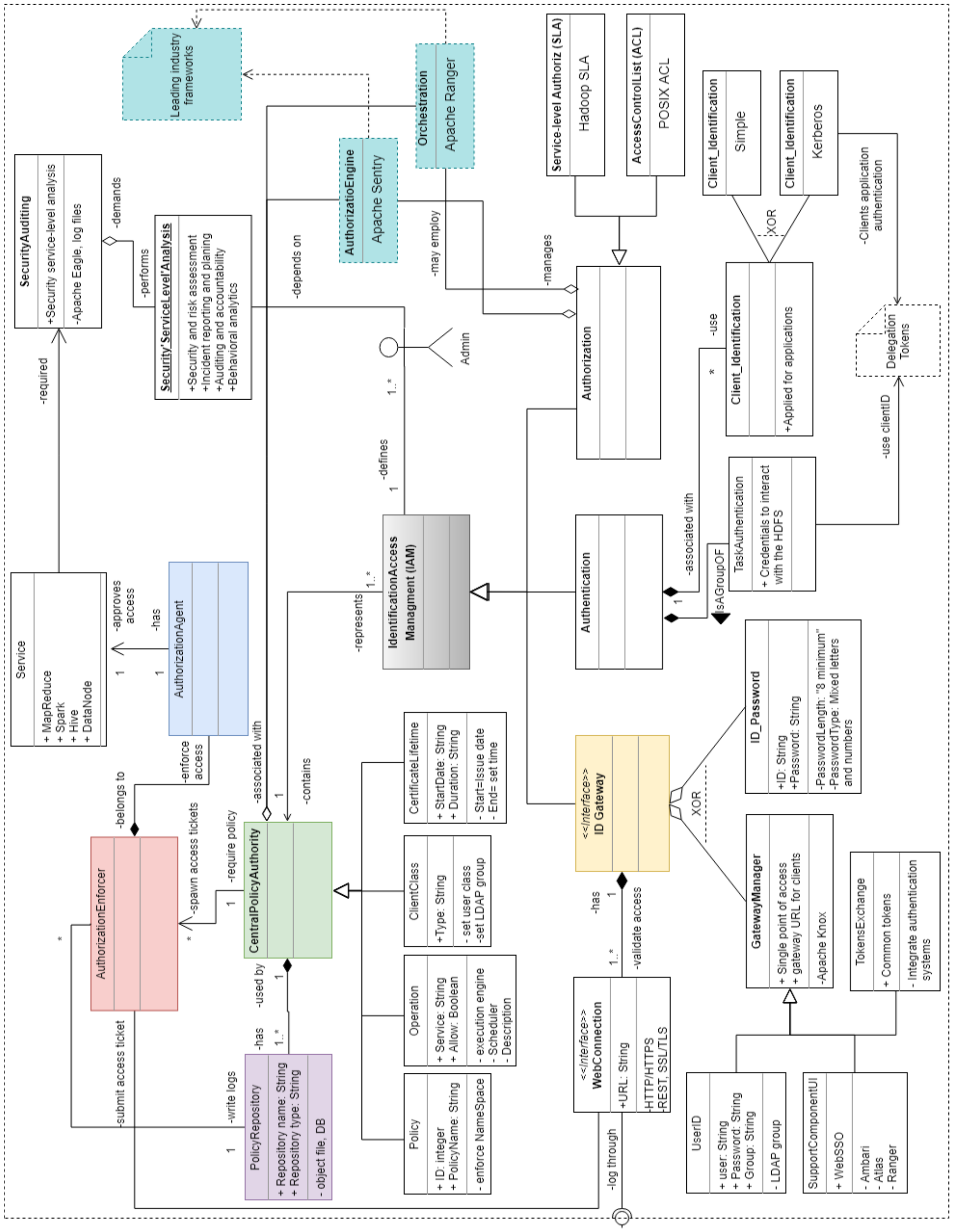


Figure 5.6: Federated Access Control Reference Model (FACRM) for Apache BD stack.

ii- SupportComponentUI: Knox provides WebSSO capabilities to the Hadoop cluster. It is therefore essential to support different system component UIs and to link them to the end user. Examples include Apache Ambari UI for cluster development and orchestration and the Ranger admin console for setting/modifying policies.

iii- TokensExchange: maintains a secure interaction among different frameworks and clients by providing a universal authentication platform that abstracts token exchange within federated Hadoop clients.

b- Authentication: the second stage of access control, which uses the client identification to authenticate the job service calls (after validating the client access to the cluster in the previous stage). The process takes place by specifying the core-site.xml in the Hadoop configuration files through enabling the hadoop.security.authentication feature. This configuration can be either simple or Kerberos authentication. In a multi NNs environment, like a federation environment, Kerberos manages the DNs authentications by defining the entities which are allowed to communicate with the HDFS manager.

c- Authorization: the final stage of access control that sets each client permissions within the cluster entities. By default, this service is disabled; admins, however, can enable it using the Hadoop core capability in hadoop.security.authorization within the core-site.xml file. This includes the basic set of entity permissions using the SLA. An additional level of access control granularity can be acquired using HDFS POSIX ACL. For instance, ACL supports entity authorizations such as file-permission (read, write, execute). When the client creates a new file or sub-directory, it will automatically inherit the ACL permissions of the parent directory. However, clients are asked to gain separate authorization for each service. Correspondingly, client authorization (fine-grain access control) may be managed separately via different access control vendors, such as Apache Ranger and Sentry. Ranger provides dynamic data masking (in movement) for several frameworks of the Hadoop stack (e.g., HBase, Storm, Knox, Solr, Kafka, and YARN), while Sentry supports the Impala SQL query engine. The next section will demonstrate how ACL and Ranger/Sentry roles may be used interchangeably in a Hadoop cluster.

WebConnection: every client log has only one gateway, while the gateway manages several log connections.

CentralPolicyAuthority (CPA): a policy-based authority that keeps spawning access certification (tickets) for IAM and monitoring user access. It represents the central process of the IAM that releases access tickets and auditing functionalities. These tickets map each

user/group with its granted permission and enforce service access discussions. It also sets the maximum lifetime and the start date of each policy, as well as the user class authorization intervals. The CPA validates the client login and creates an access certificate by checking both the Kerberos authentication list and Ranger authorization level access. Hadoop administrators may manage repository policies by setting up both Ranger and ACL policies (optional). In this case, Ranger will verify the fine-grain client access control, i.e., which HBase/Hive DB and table columns they have access to, Kafka queues, and HDFS level of access. Meanwhile, the ACL will verify the access control of remaining entities. However, Ranger policies will take priority over those of ACL. If a Ranger policy does not exist, then local ACL will take effect. Hadoop daemon authentications and internal communication (such as task status) will primarily rely on using the Kerberos principal and keytab file locations and are enforced using Hadoop core access control, i.e., ACL.
i- Policy: as a unique name for every policy, a policyID ensures that policies are not duplicated in the cluster.

ii- Operation: defines the operation type (its authorization level value) and passes a Boolean that enables/disables the authorization level for every user/group for each service. Policies are enabled by default unless the admin restricts them.

iii- ClientClass: assigns individual user permissions or sets group permissions for each policy, and defines each client class according to that policy.

v- CertificateLifetime: a validation interval of the certificate for the client session on the system.

PolicyRepository: a file or DB that stores the certifications, as well as the auditing logs information. The repository functions may regularly include cache policies and track ticket updates (status saving). Numerous AuthorizationEnforcer daemons will write auditing logs to the PolicyRepository (as a unified auditing store). These policies can be classification-based, prohibition-based, time-based, and location-based.

AuthorizationEnforcer: a process that enforces access decisions based on the CPA policies before allowing communication with its underlying services. It represents the NN daemon (see Figure 5.3) in a native HDFS access control pattern. To generalize the IAM processes within different methodologies (i.e., only utilizing the Hadoop core capabilities or employing a third party), the presented metamodel separates this functionality. Every service call thus must pass through a predefined AuthorizationEnforcer before reaching the service.

AuthorizationAgents: a distributed representative component that belongs to the AuthorizationEnforcer. Each service daemon (DN, HBase table) has an AuthorizationAgent that approves the access call. The AuthorizationAgent could be an ACL or a Ranger policy agent.

Service: the system processing framework, execution engines, and HDFS DNs. These services may include, but are not limited to: MapReduce for batch querying, Apache Spark for micro-batches, and Apache Storm for real-time processing. A comprehensive security ecosystem will require performing auditing measurements at the service level of BD applications and frameworks.

SecurityAuditing: tracks the service calls and status of requests, in addition to monitoring client activities. This service may include identifying security and performance issues within the local log files or employing the Apache Eagle framework. These services require security auditing to improve the overall security measurements.

SecurityAnalysis: carrying out a security assessment based on client behavioural analysis, incident reports, and audit pieces of information.

Admin: the one who sets the policy for other users of the system. The admin depends on the SecurityAnalysis and client demands to define the IAM policies. Admin is also responsible for securely installing/configuring the cluster entities, maintaining a secure operational environment, performing security patching policy, and scaling computing resources based on peak utilization.

5.5 Implementation and Validation

In general, BDF provides a systematic way to dynamically provision the user demands with elastic resources and services over shared data blocks. Implementing federated authorization and delegation to enhance BD client access decisions is vital for the success of adopting these new features of Hadoop 3.x. In this paper, we argue that the FACRM is essential for the concept of unifying the IAM in a BDF infrastructure and the data that they stock. We demonstrate how the proposed components of BDF deployments can be brought to a secure access broker architectural pattern. This BDF access broker abstracts the identification, authentication, and authorization discussion within the large-scale data analytics of the Hadoop 3.x platform. Furthermore, we explain the required steps to utilize the FACRM as a multi-tier Hadoop architecture within any deployment architecture.

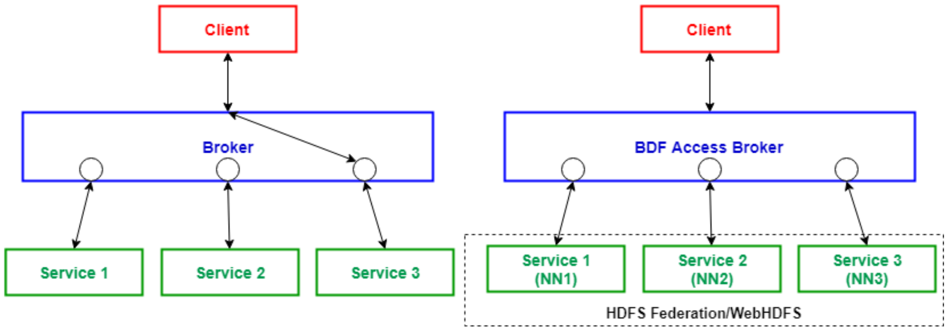


Figure 5.7: A high-level abstraction of general access broker pattern vs. the proposed BDF access broker

5.5.1 Federation Access Broker

Conventionally, system architectures employ an intermediate layer, i.e., intermediate brokers that orchestrate the underlying service's communications with external entities. The most common implementation of such architectures relies on forwarding all of the client's traffic into a proxy server or gateway for authentication. Such applications are called access brokers, which provide perimeter security by adding an authorization layer based on policies. An access security broker is thus an on-premise or cloud-based security policy enforcement point. It is placed between the external client and the underlying services provided by the system.

In general, the broker architectural pattern can be used to structure distributed systems with decoupled components that interact through remote service invocations. In this study, we present and demonstrate a pattern for this type of system, and we highlight its importance on the federation scale. Figure 5.7 shows a conventional access broker (on the left) and an abstracted architecture of the proposed BDF access broker. The proposed BDF access broker aims to abstract the management of the remote access calls to Hadoop federation services and data without user intervention. Thus, it is responsible for coordinating security policies (authentication and authorization) as BDF assets are accessed (data, services, and infrastructure). It also enables centralized control through the utilization of a unified pattern, which facilitates auditing and analysis.

The proposed architecture broker pattern in Figure 5.8 is composed of (i) authentication (credentials and passwords) using Apache Knox, (ii) authorization policy enforcement using Apache Ranger, and (iii) security analysis/auditing capabilities using a centralized access

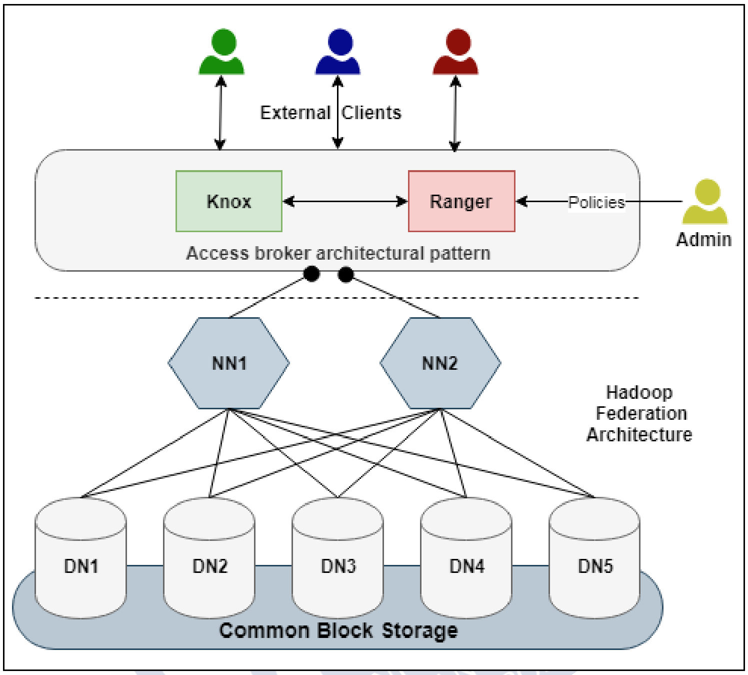


Figure 5.8: BDF access broker architecture pattern

audit log repository and Apache Hadoop 3.x. In particular:

1. Knox gateway access: employed for external client identification and cluster access.

2. Ranger authorization assignment: the clientID is passed to Ranger to validate the access call to contact NNs, and a secure session is established. If successful, Ranger responds with a certificate for authorization level access and assigns the permissions of each NN (defined by the admin) to reach the stored data in the DNs which belong to that specific NN.

3. Security log: access audit logs are created with each attempt to reach the data and are stored in unified HDFS storage (see Section 5.6).

Table 5.2: Federation Supported Hadoop Access Control Model

Basic Sets and Functions
– U, G and S (finite set of users, groups and subjects, respectively)
– HS, OP_{HS} (finite set of Hadoop services and operations, respectively)
– directUG : $U \to 2^{G}$, mapping each user to a set of groups, equivalently $UGA \subseteq U \times G$
– $\text{HS-PRMS} = 2^{HS \times OP_{HS}}$, set of Hadoop service permissions

Permission Assignments
– $PA_{HS} \subseteq (U \cup G) \times \text{HS-PRMS}$, mapping entities to Hadoop service permissions.
Alternatively,
$hs_{prms} : (x) \to 2^{\text{HS-PRMS}}$, defined as $hs_{prms}(x) = \{p \mid (x,p) \in PA_{HS}, x \in (U \cup G)\}$

Effective User Permissions
- $\text{effectiveHS}_{prms} : U \to 2^{\text{HS-PRMS}}$, defined as
  $\text{effectiveHS}_{prms}(u) = hs_{prms}(u) \cup \bigcup_{g \in \{\text{directUG}(u)\}} hs_{prms}(g)$

User Subject
- userSub : $S \to U$, mapping each subject to its creator user, where the subject acquires some or all of the permissions of the creator user.

Hadoop Service Access Operation Decision
A subject $s \in S$ is allowed to perform an operation $op \in OP_{HS}$ on a service $hs \in HS$
if the effective permissions of userSub(s) include permission assignments for $hs \in HS$.
Formally, $(hs,op) \in \text{effectiveHS}_{prms}\,(\text{userSub}(s))$

5.5.2 Formal Access Control Model Components

We have formally defined the federation supported access control model in Table 5.2. This model has been adapted from the object tagged RBAC model [176] for Hadoop, and introduces required components to demonstrate the federation.

The basic components for the model include a finite set of Users (U), Groups (U), and subjects (S), which are created by the user and run on its behalf. A user can belong to groups, and permission can be assigned to groups which trickle to the member users. Hadoop Services (HS) are the required daemon background processes, including NameNode, DataNode, YARN ResourceManager, etc., to which the user must be allowed access. In the case of federated Hadoop 3.0, because we have multiple NNs in the system, the user must have access to only specific ones. OP_{HS} are the actions of Hadoop services. ACLs have been used in native Hadoop service authorization capabilities to restrict access to users.

Table 5.3: BDF access Broker proof of concept validating environment.

Node Number	Hadoop Daemon	Hardware &Configuration
2	NameNode	VCPUs: 8, RAM: 16GB, HDD: 160GB
3	DataNode	VCPUs: 4 RAM: 8GB HDD: 80GB
Common criteria: 1 Gbps network connection, Ubuntu 16.04 LTS OS.		

As shown in the table, the Hadoop service permissions (HS-PRMS) are the power set of the cross products of HS and OP_{HS}. A many to many relation PA_{HS} specifies the assignment of Hadoop service permissions to the users or groups. In this way, a user can be assigned permission to access Namenodes in the system, either directly or through group membership (using the sufficient permissions $effectiveHS_{prms}$). For example, a user u1 may only be allowed to access NameNode1, and then the permission assignment must be only for that tuple (u1,NameNode1) in the PA_{HS} relation.

A user creates a subject stated by the userSubfunction. The subject obtains either some or all of the permissions of its creator user. A subject is allowed to access a Hadoop service if its creator user has the permissions assigned to it by the security administrator.

5.5.3 Experiment and results

Aiming to validate the BDF access broker, we implement an OpenStack [231] based Hadoop cluster that is configured as one rack of five nodes using two different node types, as detailed in Table 5.3. The bar chart in Figure 5.9 illustrates a performance comparison of the WebHDFS read operation in two use cases: the first without utilizing any security measurements (native WebHDFS access), and the second by employing the proposed BDF access broker architecture pattern. The prime motivation for this performance evaluation is to gain an understanding of how the performance will be impacted with respect to Knox and Ranger. Overall, it can be observed that the performance effect, i.e., the time needed to process different file sizes, seems to be minimal. A more specific study on performance degradation is eventually needed, but for this proof-of-concept, it is correct to state that the performance has not been significantly impacted.

To define the relationship between the basic WebHDFS configuration and secure WebHDFS configuration with the proposed broker, we compare the speeds of the two methods. We fitted two linear regression models (see Appendix A) that can be found in Table 5.4 and 5.5. In the first case, the slope is approximately 0.24, and 0.25 in the second. These results show that the time increases by only 1% for each MB using the secure approach. Figure 5.9 also shows the time needed to perform WebHDFS calls (read/write) over different data points (data chunks between 100 MB and 500 MB).

Table 5.4: Access Control impact on Read performance with Sample Standard Deviation (SSD) and Transfer Speed (TS)

	100MB	200MB	300MB	400MB	500MB	TS	Mean	Median	SSD*
Native WebHDFS	23.64	47.45	69.63	94.23	120.62	25 MB/s	71.11	69.63	38.08
BDF Access Broker	25.07	49.36	72.05	98.1	127.85	26 MB/s	74.49	72.05	40.26

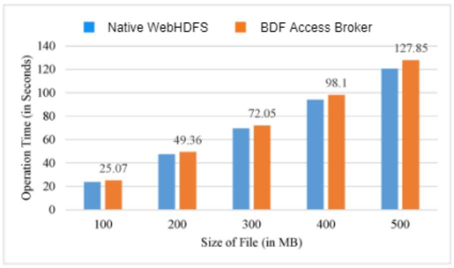
Table 5.5: Access Control impact on Write performance with Sample Standard Deviation (SSD) and Transfer Speed (TS)

	100MB	200MB	300MB	400MB	500MB	TS	Mean	Median	SSD*
Native WebHDFS	16.55	33.9	48.7	65.9	84.43	42 MB/s	49.9	48.7	26.54
BDF Access Broker	21	43	61.8	83.69	107.2	-	63.34	61.8	33.71

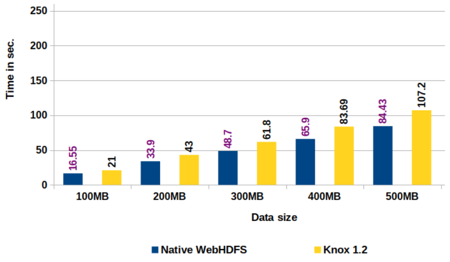
A box and whisker chart in Figure 5.10 shows the distribution of data into quartiles, highlighting the mean and outliers. Our experiment shows the box and whisker plot medians, interquartile ranges, and ranges of the secure and non-secure approaches. This experiment aimed to identify covariates that could influence the length of time taken to perform a read/write call by using the box and whisker plots to allow visual comparison of the median and spread. Areas of potential interest are highlighted for future prospective work. The whiskers extend from the box to the highest and lowest values, and our experiment shows that no outliers exist. Also, Figure 5.11, shows that the relationship between the two variables is positive linear.

5.6 Access Audit Log Management and Analysis

NN, as a data lake gateway, writes access audit log information to local files after each successful/failed access call. These access log files generally contain information regarding the user and the data the user accessed. Typically, Hadoop creates logs after each user read/write call using Apache Log4j files [232], i.e., a Java library. This detailed log information is managed locally by each Hadoop component and provides a useful accounting process (e.g., timestamp and IP address) that logs all operations. The log files may serve multiple purposes, such as debugging operational issues and regulatory compliance. They also provide security

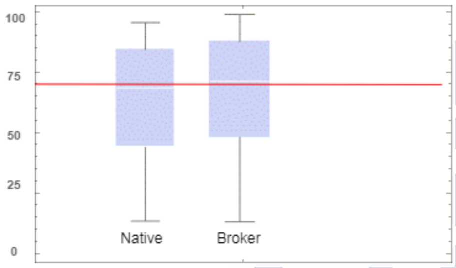


((a)) WebHDFS Read

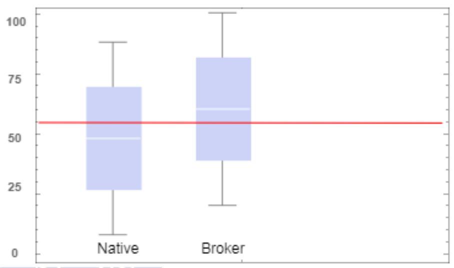


((b)) WebHDFS Write

Figure 5.9: Performance analysis of WebHDFS operation within different file sizes

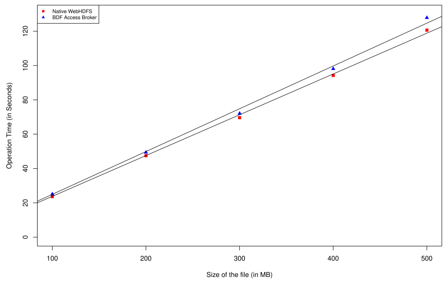


((a)) WebHDFS Read)

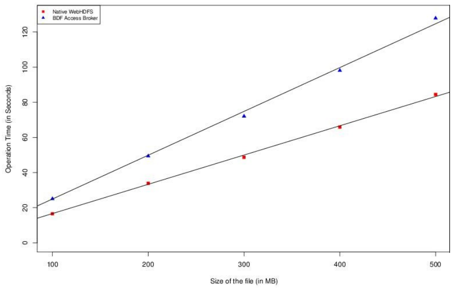


((b)) WebHDFS Write

Figure 5.10: A box and whisker plot displays a performance comparison of BDF broker against native (non-secure) Hadoop implementation



((a)) WebHDFS Read



((b)) WebHDFS Write

Figure 5.11: Graph of the relationship between secure and non-secure WebHDFS rates

capabilities to execute forensics from an access audit perspective (e.g., user logs over HTTP). In particular, access audit logs are valuable pieces of information for conducting security auditing and analysis of the Hadoop cluster with high operational resolution.

However, analysing the collected YARN and HDFS logs generated by applications of many distributed components (locally at each daemon) is a difficult and insufficient task. Each Hadoop daemon creates its own output log messages and manages them in a local directory (on the node on which it is running). Hadoop Archives may be employed to reduce the number of small files and thus the stress on the NN [233]. However, it is still difficult to perform security analysis or digital forensics over a large-scale Hadoop federation model that could include dozens of NNs and hundreds of DNs with broad user participation. Another way to reduce the complexity of the distributed log file analysis is by utilizing log aggregation: for instance, YARN log aggregation, a management process that fuses different log files from various sources in a centralized repository to facilitate the analysis of the collected logs. This feature is disabled by default; by enabling it, the log files of different daemons will be allocated to a centralized directory (e.g., HDFS). However, even using this feature, the large amount of aggregated data in Hadoop federation architecture will lack governance requirements, like file tagging, labeling, and classification. It will be stressful for investigators to examine such pieces of information, and it can easily frustrate them.

The proposed access broker not only provides fine-grained access control but also a centralized auditing approach (see Figure 5.12). By employing the Ranger Audit Server, the BDF access broker will aggregate all access logs into a centralized repository (RDBMS, HDFS, or Log4j). The log aggregation service will regularly check the running tasks and start to aggregate and transfer the logs to a centralized repository (typically HDFS). Subsequently, access audit logs can be allocated by other digital forensics tools for any visualization or search tool for further investigation. Moreover, server audit logs provide valuable information to identify and repair system vulnerabilities.

Consider the following scenario that illustrates how to investigate a security breach using the access audit logs by storing and analysing the access log data.

A Hadoop system administrator notices a tenfold spike in the number of support tickets. Because the system utilization is high, due to the unusual ticket requests, clients start facing problems logging into the VPN. The admin must analyse the log data, since he suspects that he may be experiencing a distributed denial of service (DDoS) attack. The questions that he must answer are: “how many connection requests have been delivered in the past few

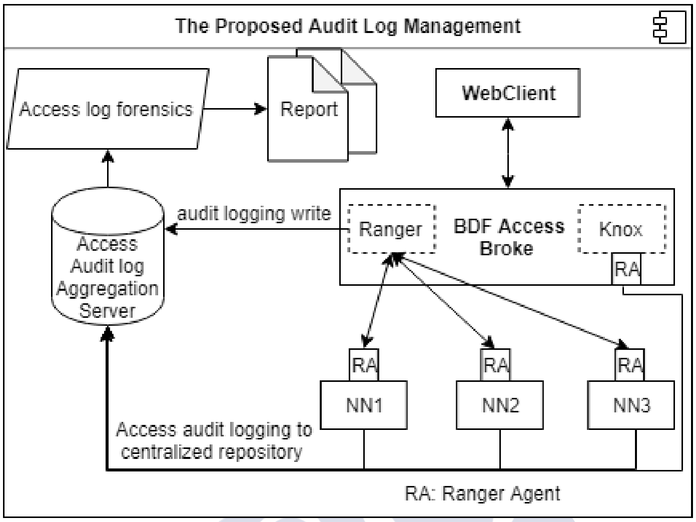

The Proposed Audit Log Management
Access log forensics
Report
WebClient
audit logging write
Access Audit log Aggregation Server
Ranger
BDF Access Broke
Knox
RA
RA
RA
RA
NN1
NN2
NN3
Access audit logging to centralized repository
RA: Ranger Agent

Figure 5.12: A conceptual BDF access broker log management model

hours?", "where are the requests coming from?", and "what types of connections are they (authorized or not)?" The analysis includes investigating how and when the network traffic changed. Doing so requires engaging in three steps to transform the logs into useful figures, namely, loading, refining, and visualizing the data, in the search for useful information, as follows:

- Loading log data: Utilizes Apache Flume [234] to stream large amounts of log data from the VPN server into the HDFS. The log records contain a timestamp, IP address, country, and indicator about whether the connection was successful or not.

- Refining log data: A storage management layer must be employed to enable a relational view of data without affecting the data format. This layer aids in browsing the log data (making sure it has been loaded correctly). Apache HCatalog, i.e., Hadoop catalogue, may be utilized to provide a metadata layer and rest interface for Hadoop data. Next, the admin needs to write them onto any data analytics framework for review (visualization tool). An Apache Pig script may be used to push the latest log data to the visualization tool.

- Visualizing log data: The admin interfaces the visualization tools, like ElasticSearch (a real-time search engine and analytics tool), and Microsoft Excel (using open database connectivity). ElasticSearch provides high-level visualization to identify the moment when the network traffic increased, where it came from, and whether or not it was authorized in order to confirm his hypothesis about the DDoS attack. More detailed maps may be created using the Excel Power View functionality by linking the Apache Hive (the base of HCatalog) and Excel to determine whether there is cause for concern.

The forensic report and the process output data are used to update the firewall to deny requests from the unauthorized IP addresses. To automate this update, the admin can use Apache Ozzie to schedule a job to automatically update his firewall every hour. This report can also be used for other forensics to respond to different security threats, prepare for compliance audits, and prepare behavioural analytics (for security patching).

5.7 Summary and Future directions

Next-generation access control systems aim to become more scalable with even finer-grained authentication and authorization management [1]. However, given the emerging trends of the distributed, heterogeneous, and federated infrastructure associated with Hadoop 3.x architectures, the need for a lightweight security framework is inevitable. In this paper, we present a BD Federation-oriented reference model for the secure development of access control solutions within Hadoop clusters. The proposed model is a generic high-level conceptual metamodel (called the FACRM) that aims to formalize Hadoop 3.x core access control capabilities and highlight some of Hadoop's top-level security projects. It also aims to successfully implement a unified and secure BD solution over on-premise and cloud-based deployment architectures in favour of BD applications and frameworks. The FACRM supports myriad access control options across a variety of policies, users, and frameworks in pursuit of modern BD infrastructure. Furthermore, this study presents an in-depth investigation of the Hadoop IAM ecosystem and draft current access control limitations within a federation deployment architecture. We validate the utilization of our proposed RM by implementing a proof-of-concept architecture for the security and privacy of BDF deployment. Our test conclusively confirms the effectiveness of the proposed access pattern performance and security.

The general aim of this study is to develop an RM using open-source frameworks (e.g., Knox, Ranger) to promote the adoption of reliable access control mechanisms within the BD deployment architectures. It also aims to facilitate the security design of all clients, resources, applications, and networks in Hadoop-based BD processing environments. In this context, the proposed federation BD access control meta-model is a service-oriented architecture (SOA) that conforms to the best practices of independent vendors, products, and technologies. This SOA makes FACRM frameworks (a framework design based on FACRM) into SOA frameworks. However, FACRM frameworks are not necessary for BD frameworks only; it could be any framework or application that runs over the HDFS system, which is a large-scale distributed environment (platform). This environment is, thus, characterized by horizontal scalability, rapid provisioning of access control, ease of access, and security. Additionally, Our BDF access broker provides an attractive solution to the audit log problem that effectively addresses compliance requirements. To map the current Hadoop limitations (presented in Section 5.3.3) to the solution domain of this study, Table 5.6 represents these limitations as threats and links them with solutions.

Table 5.6: Linking current Hadoop federation limitations to the proposed solutions

Current limitations	**Proposed solution**	**Description**
Access control complexity	FACRM propose that policies are written once and applied many times.	In Figure 6, the CentralPolicyAuthority (in green) allows the admin to set up a policy and distribute it to all nodes (using the Ranger agents), which are represented by the AuthorizationAgent (in light blue) attached to all of the services.
Long-running services	FACRM propose flexible time-based policies.	The CertificateLifetime in the proposed reference model corresponds to the long-running services. These certificates are associated with the CentralPolicyAuthority and saved at the PolicyRepository to be updated automatically upon demand.
Communication orchestration	FACRM propose a third-party centralized access control.	Security verification for all of the entities (users, nodes, and daemons) inside the Hadoop cluster and external clients. In Figures 7 and 8, it can be observed that the broker transfers the access discussion responsibility from the NN to the broker.
Data integrity	access control at the file level and service level (access call).	A federation configuration requires authorization verification of the entire file path to create secure multi-tier federation architecture. Additionally, multi-layer control (authentication and authorization) occur at the beginning of operation (reading and writing to HDFS).
Node orchestration	FACRM provide access control using any approach.	FACRM and BDF broker provide the base to support the multipart access control framework and technologies, e.g., RBAC, ABAC, etc.
Audit log management	utilization log aggregation using centralized auditing approach.	Figure 12 illustrates how to employ the proposed broker with Ranger Audit Server for accessing broker log management.
Security auditing	proposal of an architecture pattern for BD governance.	In Section 7, we proposed a scenario that transforms the logs into useful figures by loading, refining, and visualizing the data into incident and forensic reports using open-source frameworks. This section shows how to investigate a security breach using Apache Flume, Apache HCatalog, and ElasticSearch, among other tools.
Access control scalability	proposal of a novel BDF access broker.	The main components of our proposed broker provide high scalability. In particular, Ranger provides centralized Hadoop security administration and management, while Knox streamlines security for services and users who access the cluster data and execute jobs.

Overall, the primary benefits of the BDF broker architectural pattern can be summarized in four different categories:

Enhanced security: A method to ensure that data is stored securely in BDF and BDaaS clouds using a robust access control mechanism. This involves using (i) Knox to expose REST and HTTP services without revealing the details of the underlying Hadoop cluster, and (ii) Ranger for fine-grain authorization over federation NNs.

Centralized control: Using a single gateway with a distributed agent to prevent unauthorized access to the modern data lake. This also allows IT departments to set and enforce security policies regarding data usage, access calls, and resource utilization over secure channels.

Facilitated auditing and analysis: A centralized repository for all access logs with a sophisticated governance and accountability approach (reporting and regulatory compliance). This approach enables threat prevention methods, e.g., behavioural analytics and threat intelligence.

Consolidated governance pattern: A way to ensure and prove compliance with granular visibility and control to meet regulatory requirements, and to guide the efforts of admins to ensure that the system complies with all relevant regulations and standards (such as data residency) when using Hadoop 3.x.

Blockchain technology can serve as a revolutionary solution, addressing current BD privacy concerns such as wire (end-to-end) encryption and decentralization data-driven cryptography. As an enabling technology, blockchain provides a sequence of block cryptography that stores all transactions, which has been established as a solution for the large-scale distributed agents of Bitcoin system security [235]. The core functionality of blockchain technology relies on providing robust cryptographic (each agent is assigned a private key, whereas a public key is shared with all other agents) proof for data authentication and integrity by providing a list of all transactions and a hash to the previous block. This list is verified by majority agreement of nodes (P2P network where the sender broadcasts it to all of the other nodes) that are actively involved in verifying and validating transactions [236].

The current Hadoop setup does not enforce access control by the DN to access its data blocks. Another issue remains in that both NN and DN generate a block access token using the same key. If an attacker leaked the key, he could generate a token and access any data blocks in the HDFS. Also, the leaked DT can be used to steal a large amount of data from HDFS, since it has the privilege to behave as the Hadoop client and to access all content which he is allowed to access. Any future BD Hadoop solution should consider the previous limitations on a federation scale. To this end, FACRM addresses all concerns mentioned above by defining a reference model for Hadoop federation; more particularly, it presents a meta-model that includes the main access control vocabulary and design elements, the set of configuration rules, and the semantic interpretation. Leveraging recently published management mechanisms of policy enforcement, e.g. [186], to combine cloud and edge federations with role-based and attribute-based access control [175, 177] could be considered in future research. Examples of different research directions include:

- **Hybrid access control solutions:** The development in usage control for data privacy and security leads to integrating conventional access control approaches, e.g., ABAC, with other context-based encryption technologies. This integration could include functional encryption or other generalizations of identity-based encryption and attribute-based encryption for protecting big data in the presence of mutable attributes, which may lead to general advancement in other fields, such as IoT.

- **Fine-grained Access Control:** The need to develop active and fine-grained approaches (policies and standards) for BDF is inevitable. The challenge here is to extend the previous approaches with new policy editing and presentation tools (classification-based, prohibition-based, time-based, and location-based policies) for flexible and extensible data access. In particular, this includes policies that extend ABAC with stateful access sessions and mutable attributes (i.e., characteristics that dynamically change during the session). This extension is invaluable to advance the authorization mechanisms of the multi-tenant data lake architecture, as well as the latest BDF model. Tackling this issue can be achieved by deploying policy templates that extend the primary ACL (or any access decision enforcer) as a specialization of the XACML V3.0 standard.

- **Dynamic Access Control:** More than ever, scalability and dynamicity of the BD deployment architecture in terms of user and node participation are vital for the secure delivery of security provisioning in a federation environment. There have been different adapted security mechanisms and techniques proposed to cope with this concern. However, more research activities are still needed to establish scalable security models and paradigms that must be driven by BD specifications and requirements. Also, new security abstractions for BDF are still needed to simplify the task of identifying the unimproved gaps. For instance, leveraging new mechanisms of policy enforcement [186] to combine BDF with ABAC [177] could be pursued in future research.

- **Stateful Access Sessions:** This requires restricting the BDF access to only those authorized by the succinct gateway. However, the current state-of-the-art BD-based encryption technologies involve transparent data encryption. Federation environments track the participation states and characteristics of the clients and networks traversing them with mutable attributes (i.e., the values of the attributes change over time during an access session). This operation requires formulating policy templates on top of standards such as XACML V3.0 for dynamic authorization with stateful access sessions. This takes place at the service level (e.g., data access calls) and does not protect data-in-transit or data-in-process (not even the metadata). Attribute-based encryption in conjunction with the usage control approach and ABAC can be labeled as a future direction for this issue.

Providing new solutions for all of these research directions will promote innovative BDF solutions with advanced security features. It is also expected that adoption of the FACRM use cases will be implemented within our BD opportunistic and elastic resource allocation (OPERA) platform. OPERA architecture (prototype proposed in [3]) combines the computing power of high-throughput resources available (non-dedicated) to the Hadoop 3 dedicated cluster using Docker containers as worker nodes. Those non-dedicated containers tend to be more vulnerable, which requires a robust IAM approach to minimize security threats.

CHAPTER 6

USE CASE: A SECURE VEHICULAR DATA STREAMING TO THE CLOUD

Many technological cases exploiting data science have been realized in recent years; machine learning, Internet of Things, and stream data processing are examples of this trend. Other advanced applications have focused on capturing the value from streaming data of different objects of transport and traffic management in an Intelligent Transportation System (ITS). In this context, security control and trust level play a decisive role in the sustainable adoption of this trend.

This chapter contribute in a conceptual methodology that integrats the security approaches of different disciplines into one coherent reference architecture. In addition, a classification of BD technologies, products, and services to address the ITS trust challenges is presented. We also proposed a novel multi-tier ITS security framework for validating the usability of SITS with business intelligence development in the enterprise domain.

6.1 Introduction

Recently, we have been observing significant advancements in the scale of generated and collected data using multiple technologies, real-time analytics, commodity sensors, and embedded systems to refer as the Internet of Things (IoT) paradigm. The ability to employ these modern technologies to process, analyze, and understand data and services securely is vital to advance in the use of transport networks. The intersection of these trends is what is, nowadays, called the Intelligent Transportation System (ITS). ITS, however, requires secure architectures for advancing traffic management with an adequate level of trust. In context, the ITS trust indicates that the system components meet the security requirements, —hence, clients satisfaction of availability, reliability, and governance mechanisms of the security measurements.

Advanced with a large community of open-source frameworks, the Big Data (BD) security stack (i.e., platforms and technologies) represents a practical and cost-effective solution for supporting this trend. In this work, we analyze the large body of work related to data science security frameworks and technologies. More specifically, the building blocks of the security stack for supporting IoT as a secure service for ITS scientists. Doing so aims to demystify the unique security challenges introduced in ITS environment and clarify issues from a security perspective. Also, we investigate the state-of-the-art of BD security, most of them developed under the umbrella of the Apache Software Foundation.

A secure ITS (SITS) reference architecture for facilitating security pattern design and solution election when constructing ITS systems is presented in this work. The proposed multi-tier security solution leverages Apache Hadoop frameworks from the security burden. We also present various insights into the latest ongoing developments and open challenges in this domain. The research methodology adopted towards achieving this goal uses software engineering and information systems design approaches. The necessary steps for designing the system architecture include the collection of requirements and the analysis of abstract functional specifications.

The rest of this part is organized as follows. Section 6.2 provides an overview of ITS security characteristics and discusses the main requirements of deploying ITS trusted solutions. Section 6.3 analyzes the state-of-the-art of Big Data systems based on their security supported service model. Design and construction of the multi-tier reference architecture are presented in Section 6.4. Classification of big data technologies and commercial products/services are highlighted in Section 6.5. Finally, we draw our conclusions in Section 6.6.

6.2 ITS main security requirements

It has been well-recognized that IoT effectively supports a predominant phenomenon in every sphere of smart life. This vision including vehicular connection to computation, as for vehicular clouds [237, 238], and vehicular IoT [239, 240]. The connection involves data exchange between vehicles, traffic components, and other subsystems, which impose security concerns. In general, the security of ITS may be characterized by:

- Ease of distribution of secure access;
- Accelerate authentication spawning and provisioning;
- Provide easy-of-use mechanisms at scale;
- Flexible for adapting to client requirements and demands.

To achieve the goals mentioned above, we envision a list of main requirements that should be met by ITS security solutions. In particular, from the infrastructure point of view, the following requirements should be met:

- **Standardized Access Management**: The framework architecture should expose its resources using standard technologies (e.g., Web services, service-oriented architectures, microservices, etc.) making them usable as building blocks, e.g., directed acyclic graph for real-time computation and services. This also includes identification and access management, as in robust authentication of these resources.
- **Efficient Distributed Mechanisms**: The framework architecture should be able to cope with very large and broad nodes participation that are geographically distributed across many sites.
- **Scalability**: The framework infrastructure should be able to handle elastically growing participation (deriving from vast sensors and end-nodes) by dynamically allocating the needed resources (processors, storage, network). Moreover, as soon as the resources are allocated, the security architecture should implement the required security management, but with minimal permission stages (to improve the overall performance).
- **Efficiency**: The framework architecture should minimize the security discussion for a given task execution (process and all related communications). In the case of large-scale distributed and parallel execution, efficient allocation of security measurements should be assured.
- **Data Availability**: The framework architecture should achieve high availability with robust fault tolerance and recoverability techniques. In practice, IoT-based ITS lives and

dies by the shared data. It is crucially required to maintain the availability of massive datasets, i.e., the more data that are available, the more productive discussion and results that ITS can produce. Hence, any solution attempted to incorporate strong security has to consider recoverability and fault tolerance approaches to deliver data availability on a real-time standard.

- **System Interoperability**: The framework infrastructure should be designed as a set of network-enabled security components (services) implementing the different stages of the overall security (e.g., integrity, confidentiality, governance, etc.) to facilitate their effective reuse, composition, and interoperability. Standardization can enable interoperability and ease interaction among the various security frameworks and mechanisms. A service-oriented model can present a vital role in supporting the integration of various components of different security layers.
- **Model Usability**: The framework process should be usable by any ITS system teams of both experts and non-security experts. The design should easily apply security to their problems, achieve a high-trust environment, and deploy solutions that can be used in the modern ITS applications.

  Among the requirements at the architectural level, the main two are:

  - **Service Performance**: In the realm of ITS, applications deliver their services in a continuous data stream pattern within a small throughput time (from receiving the data to process it). The applications detection period varies from a few milliseconds to seconds. As so, security must deliver its functionalities in a real-time fashion.
  - **Openness and Extensibility**: ITS security architecture should be open to the integration of new frameworks and mechanisms (with even already installed services). Moreover, existing services should be open for extension but closed for modification. By which means easiness of extending new approaches to tackle new threats and vulnerabilities while reserving its core functionalities maintained.

6.3 Big Data Security Frameworks

In the realm of intelligent life of broad applications and services, the explosive growth of data traffic will drive a variety of domains including IoT-based systems and ITS. This phenomenon

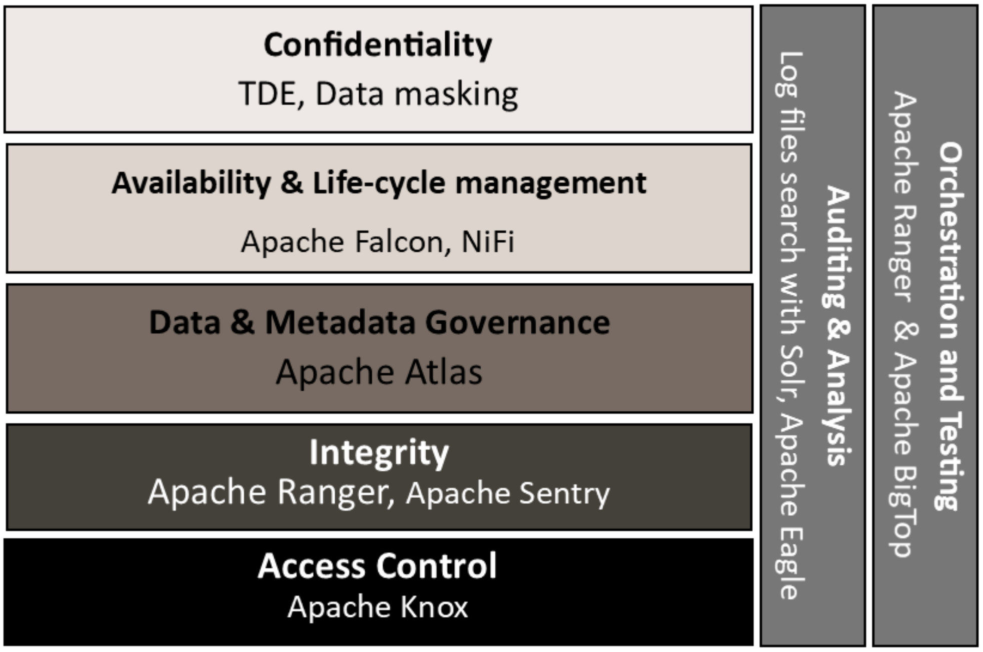

Confidentiality
TDE, Data masking
Availability & Life-cycle management
Apache Falcon, NiFi
Data & Metadata Governance
Apache Atlas
Integrity
Apache Ranger, Apache Sentry
Access Control
Apache Knox
Auditing & Analysis
Log files search with Solr, Apache Eagle
Orchestration and Testing
Apache Ranger & Apache BigTop

Figure 6.1: Intelligent Transportation Systems Trust layers abstraction.

has created ever-increasing pressure for large-scale data analytics security solutions. Several data security management systems have been recently implemented. In particular, they are addressing the CIA triad, i.e., Confidentiality, Integrity, and Availability.

The increasing data security requirements of almost all application domains have also created a crucial need for designing and building a new generation of service-oriented frameworks. These frameworks can efficiently and effectively secure data at its three main status, i.e., data-at-rest, data-at-transit, and data-at-process. Data life cycle security within ITS environment is illustrated in Figure 6.1. This abstraction summarizes the basic trust criteria for security evaluation and selection based on BD frameworks as a service for ITS scientists. In the following sub-sections, we survey the Apache BD security stack according to these layers.

6.3.1 Access Control

An ITS infrastructure could represent a virtual net of various independent domains. In such infrastructure, the relationship between edge nodes is more ad hoc and dynamic; resources are not in the same security domain. Sensors and edge IT may be identified by their characteristics or attributes, besides predefined identities. The traditional access control models fail to cope with these modern requirements. Attribute certificates can address this issue by assigning a value pair and the principal to whom it applies. This contains an attribute to make access decisions based on the attributes of requestors, resources, and the environment. Attribute authorities sign them, which provide the needed flexibility and scalability that are essential to large-scale distributed systems such as the IoT.

Here in, we subdivided the access control into three primitive layers. Namely, a service gateway, authentication, and authorization. They are discussed further as follow:

Service Gateway connects ITS different components securely and efficiently. This layer verifies the edge access to the system using their ID and security domain. The scope of a gateway within an ITS is to provide scalable security services based on standards across different domains and geographical areas. Introducing a trusted gateway can specifically address the loss of the traditional security boundary by producing trusted security domains and enabling cooperation between these. This process may include Apache Knox [192], a unified gateway framework for big data services and ecosystems that can be utilized as a SSO gateway. Its architecture is formed as a reverse proxy that can be configured to enforce SSL/TLS communication.

Authentication is the act of confirming authentication access to the network, i.e., the process of actually determining the identity. Any secure model should enable an authentication mechanism to verify that an entity is what it claims to be, e.g., using the Kerberos protocol (authentication based on tokens). In this case each service and user must be authenticated by Kerberos keytab file (binary containing the information needed to log) to initialize trust between ITS edge and applications. This prevents an attacker from compromising an identity and hence, send malicious traffic to victim endpoints or other internal services (edge IT or cloud resources). Besides, it also prevents from unauthorized access to any object inside the cluster and any service claimed to be part of the cluster. Apache Sentry [241] provide a role engineering approach that enforce fine-grained levels of authentication discussion to data and metadata. Its architecture covers variety of data pipelines, task execution, and storage systems (including Apache Hive, Solr, Impala, HDFS, etc).

Authorization is the process of defining access rights (privileges) that an entity (service, daemon, or device) can perform to the given service. It manages access in the context of a specific service, resource, and data functionality provided by the cluster. Service level authorization deals with the fundamental set of permissions; like defining the users and groups who are authorized to make service calls (e.g., data access) to that service. Only if the object making a call belongs to an authorized service entity, the call will pass the authorization check. Apache Ranger [?] is a centralized administration that enables write-once-apply-many policies (using distributed agents across the infrastructure). Ranger affords to monitor and manage comprehensive data security over large-scale data analytics. It also provides fine-grained access control (table, column, and file levels) by enforcing an access control list, i.e., consistent policy administration across all ecosystem. Access control list, in return, combines three elements: Effect (allow or deny), action (e.g., data access or execute), and resources (e.g., datacenter, IaaS cloud, cloud storage). In principle, authorization is the final access control layer in a security abstraction, which means that no additional security mechanism is required as an identity access management intention for an authorized entity. Nevertheless, data encryption both at rest and in transit is still required, besides security analysis, auditing, and metadata governance, which combined came as high-level security services.

6.3.2 Integrity

ITS applications deliver their service by transferring data (structured and unstructured) through several communication technologies (Bluetooth, WiFi, etc.). Assuring that their assets can be modified only by authorized agents or in authorized ways is critical. In this context, data integrity refers to protecting data from unauthorized manipulation or abuse, including altered by an attacker. In other words, maintaining the accuracy and consistency of data over their entire life-cycle. By implementing end-to-end security and blockchain technology [242], ITS applications can achieve sufficient confidence in system overall integrity.

However, a breach can occur due to faults in the edge nodes, network faults, or buggy software, besides abuse or attack. Several approaches are implemented to tackle this issue, as implementing checksum checking on the transit data. Wire encryption, as for data transfer between the edge and datacenter/cloud may be managed via IPSec or Transport Layer Security (TLS) [243] (and its predecessor Secure Sockets Layer, SSL) cryptographic protocols over HTTP communications. So that all communication between every component of the system, as well as with the client traffic is encrypted.

6.3.3 Governance

Data governance is a capability that ensures efficient manageability and data quality throughout the complete life-cycle. This feature is leveraging information to help the implementers gain insight and build confidence in business decisions and operations. Also, it delivers detailed information regarding data objects, location, characteristics, usage, and many other properties. With governance capabilities, the system can achieve higher accountabilities and data quality, for the benefits of ITS analysts and practitioners.

This layer includes several dimensions including classification (labeling and description), source tracking, and quality across data sources. Providing a typical store for exchanging metadata tags and attributes can be achieved using Apache Atlas [196] and Apache Solr [197] for defining data types and fields using full-text indexing and querying techniques. Atlas, in return, associate attribute values with resources in the ecosystem based on predefined policies (e.g., content, expiration time, or sensitivity of resources). This data tagging (metadata information) can be employed as a tag-based access policy by utilizing Ranger service [175]. This combination delivers deep audit visibility with an examination of the information captured within different components of ITS. The aggregated audit information also enhances the productivity of data life-cycle security administrators with granular insights through reporting capability.

6.3.4 Life cycle management and Availability

The IoT-based ITS distributed nature poses unique security challenges of different levels. ITS trust practitioners need to guarantee that data transition and processing stages are available to applications on-demand. In general, availability refers to the quality of service being accessible and functional upon demand by the ecosystem components. This feature includes systems ability to carry on operations even under system failures, entities misbehave, and security breach. Here in, we will address the data availability concerns by highlighting specific system requirements. So goal aims at designing a solution that assures these necessities.

In particular, the data and metadata pipelines, queuing, and life-cycle management —as ITS demands to generate a heavy reliance on the ubiquitous network's availability. Apache Falcon [244] simplifies the development of data processing solution by providing data life-cycle management and back up data services. Falcon is meant to enable administrators to facilitate data replication for fault tolerance and disaster recovery retention. Finally, Apache

NiFi [245] provides an integrated data logistics system for connecting edge data source and devices in an IoT use case. It provides real-time control and offers security using TLS encryption. Its features include the ability to operate within centralized clusters, as well as decentralized clouds (IoT-based).

6.3.5 Confidentiality

Encryption technologies mainly protect the confidentiality of data. Employing a mixture of symmetric and asymmetric cryptography can offer the efficiency of symmetric cryptography while maintaining the security of asymmetric cryptography. Traditionally, BD platforms usually utilize Transparent Data Encryption (TDE) [198]. This encryption technology sets at the file-level (data-at-rest) and does not provide the needed data masking. Analyzing how other encryption technologies such as functional encryption or other implementation of Identity-based and attribute-based encryption and signature can address the previous issue [246]. It is also recommended to integrate it with attribute-based access control, including how they perform for protecting big data and in the presence of multidimensional data attributes.

The analysis of identity-based encryption and attribute-based encryption can lead to robust end-to-end security. This ambitious goal can be achieved by integrating with attribute-based access control. Also, attribute-based encryption for stream data protection, including network functions, virtualization infrastructures, and mobile 5G [247], can be labeled as a future direction. Additionally, secure key management is considered as a critical technique for heterogeneous network security, and dynamic blockchain approaches may apply [248].

6.3.6 Auditing and Analysis

Any secure model should not only secure its entities but should be able to perform security auditing and structural analytic regularly at the service level. ITS has unique attributes that require risk assessment in areas such as physical and network security, access management, usage monitoring, and behavioral analysis among many more. This layer performs the aggregation of log files and reports to provide a robust audit capability within different components. This layer may also afford granular insights into pieces of information by tracking data pipeline audit logs and examining behavioral analytics to meet their compliance demands within IoT. Usually, such environments generate a massive amount of operational logs in real-time. The need for an analytics solution that analyzes data activities and identifying security

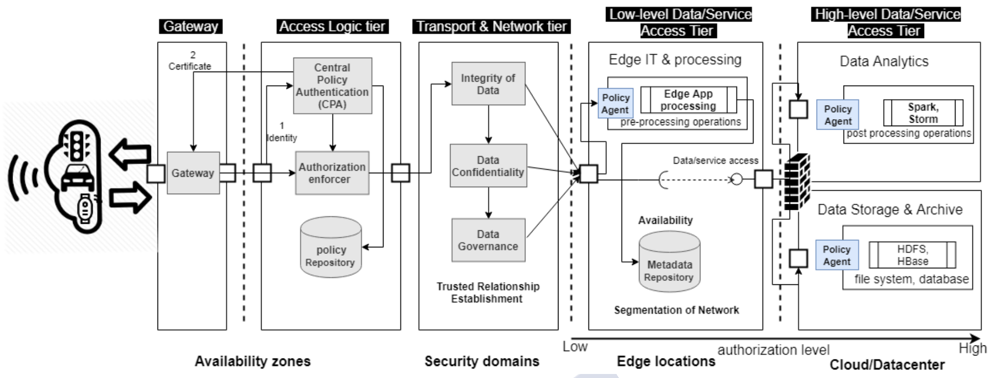


Figure 6.2: A Secure Intelligent Transportation Systems (SITS) Reference Architecture.

issues is imperative. Apache Eagle [249] normalizes and evaluates the stream log activities (including but not limited to audit logs) and alerting immediately even under huge traffic. Its agile engine allows comprehensive policies including filter and pattern matching.

6.3.7 Orchestration and Testing

Coordination and management of different security frameworks/technologies among several layers required an ecosystem orchestration. This process brought together various technologies, so they work in harmony for the benefit of ITS. It is crucially required standardizing and modeling security to enable interoperability among the various security subcomponents and products —this aids in supporting the heterogeneity of security deployment over ITS using various security layers and tools. By bringing together security components consistency, it improves efficiency and effectiveness of security management and the processes surrounding them.

Apache Ranger can effectively orchestrate classification-based security policies and allows administrators to define these policies (Geo-based, Time-based, Prohibition-based, etc.) using Apache Atlas metadata tags and attributes. It then applies this policy in a real-time approach to the entire hierarchy of the infrastructure. Also, Ranger affords central management on access audit history, including the capability to filter based on different parameters. Furthermore, Apache BigTop [250] is a unified initiative for testing various levels of data systems configuration and components (packaging, platform, runtime, etc.). Its architecture aims to provide the needed interfaces for continuous interoperability testing of integration of individual components.

6.4 Multi-tier access model

The multi-tier ITS is a five-tiered security reference architecture to improve overall trust, as illustrated in Figure 6.2, where data flow from left to right of the tiered spectrum. In general, a typical data hierarchy workflow within ITS has at least four access layers, namely, networked things layer (i) that are comprised of embedded systems, sensors and actuators. Data acquisition layer (ii) manages data load and aggregation from the network things. Edge IT layer (iii) processes the enormous amount of information prepared on the previous stage. The Edge IT also provides some pre-processing features as data cleaning, tagging, and carry data to the cloud. Finally, the cloud layer where data are materialized in the system to execute a more in-depth analysis (iv).

SITS came from the need to addresses the trust concern within the ITS platform architecture. As so, its components are specifically designed to achieve high security, availability, and reliability. The first two tiers of SITS architecture comprises the availability zones, that consist of smart and less smart things. ITS regions are geographically separated, where each area has multiple, isolated locations known as availability zones. The availability zones are structured through a set of correlated domains abstracted from the IoT scenario, but independent of any particular technology or implementation. In terms of the security level of abstraction, the zones are classified into several domains realizing a hierarchical structure. A network segmented with a different security domain meant to keep separation (using encryption and access control) of functionalities. This separation improves the isolation of edge operations, maintain integrity, confidentiality, and governance as well. Besides, helping with advanced audibility capabilities. The network tier function includes providing secure paths to transmit data between multiple security domains (sub-networks) and edge IT. Next to that is the low-level data access tier, which is located at the edge of the core network. After establishing the primary security features, it is time to allocate fine-grained authorizations.

The system administrators can enforce security discussions by employing a distributed policy agent. These security representative entities (like access control lists and service level authorization) apply several policies. For instance, classification-based policy, time-based policy, location-based policy and so on. Lastly, a centralized computation and storage capacities take place at the high-level data access tier. A single-sign-on may be implemented for

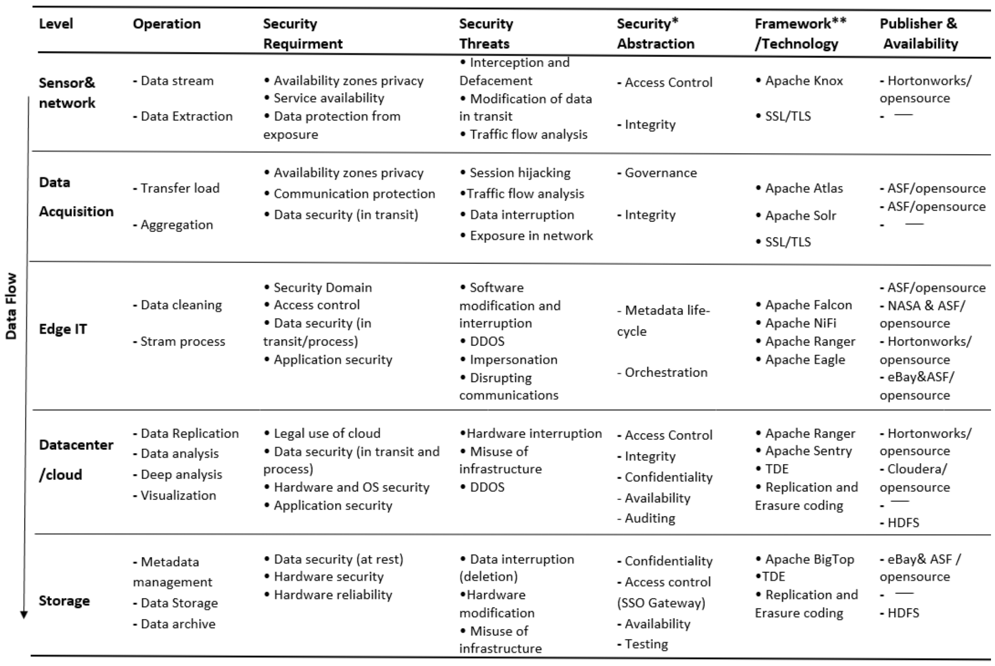


Table 1. Summary and categorization of ITS main security features (requirements and threats) mapped to BD-specific domain of solution.

Level	Operation	Security Requirment	Security Threats	Security* Abstraction	Framework** /Technology	Publisher & Availability
Sensor& network	- Data stream - Data Extraction	• Availability zones privacy • Service availability • Data protection from exposure	• Interception and Defacement • Modification of data in transit • Traffic flow analysis	- Access Control - Integrity	• Apache Knox • SSL/TLS	- Hortonworks/ opensource - —
Data Acquisition	- Transfer load - Aggregation	• Availability zones privacy • Communication protection • Data security (in transit)	• Session hijacking •Traffic flow analysis • Data interruption • Exposure in network	- Governance - Integrity	• Apache Atlas • Apache Solr • SSL/TLS	- ASF/opensource - ASF/opensource - —
Edge IT	- Data cleaning - Stram process	• Security Domain • Access control • Data security (in transit/process) • Application security	• Software modification and interruption • DDOS • Impersonation • Disrupting communications	- Metadata life-cycle - Orchestration	• Apache Falcon • Apache NiFi • Apache Ranger • Apache Eagle	- ASF/opensource - NASA & ASF/ opensource - Hortonworks/ opensource - eBay&ASF/ opensource
Datacenter /cloud	- Data Replication - Data analysis - Deep analysis - Visualization	• Legal use of cloud • Data security (in transit and process) • Hardware and OS security • Application security	•Hardware interruption • Misuse of infrastructure • DDOS	- Access Control - Integrity - Confidentiality - Availability - Auditing	• Apache Ranger • Apache Sentry • TDE • Replication and Erasure coding	- Hortonworks/ opensource - Cloudera/ opensource - — - HDFS
Storage	- Metadata management - Data Storage - Data archive	• Data security (at rest) • Hardware security • Hardware reliability	• Data interruption (deletion) •Hardware modification • Misuse of infrastructure	- Confidentiality - Access control (SSO Gateway) - Availability - Testing	• Apache BigTop •TDE • Replication and Erasure coding	- eBay& ASF / opensource - — - HDFS

Apache Software Foundation (ASF). SSL/TLS technology is the dominant wired encryption technology in BD.
* Most important securiy layer abstraction. ** Not limited to the listed items in the table.

high authorization and data confidentiality measurement. The function of this tier includes hosting ITS applications that are critical in providing end-to-end services. It is essential that security services in the datacenter/cloud internal communication are significant in ensuring that the ITS security as a whole has been hardened to protect against threats. In particular:

Tier-One. Gateway access deals with the basic set of external entities (physical smart devices like sensors) identification and network access. These devices can communicate with each other and to other devices using various communication technologies, e.g., Bluetooth or WiFi.

Tier-Two. Access logic discussion (assign authorization): The entitytID is passed to the policy authentication, exposes policies and validates to contact internal services, and issue a session. If successful, it responds with a certificate and establishes a session. This certificate contains the entityID and a copy of the session key, which is then used to communicate with the authorized cluster components.

Tier-Three. Trust relationship is the core of security establishment. This requires a robust model that unifies the security mechanism of different dimensions, i.e., the security trio of integrity, confidentiality, and governance. A high-level data access layer may be connected to that trio, in which access calls require an authorized session. This session is connected to a particular policy agent that provides access control with a means for detecting unauthorized operations. The data and call traffic may remain behind a firewall in the fifth-tier. This way allows the edge to connect to several buses as a transceiver that provides an additional layer of security.

Tier-Four. Low-level data access specifies roles for each entity within this tier (it is associated with the previous stage). This phase includes granting access (e.g., submitting jobs or querying status) with fine grain control, i.e., which database, columns, queues, etc., over the specific data blocks within the shared data repository. In the building of an IoT architecture, the location of edge IT systems is close to the sensors and actuators, end-to-end wire encryption is crucial.

Tier-Five. High-level data access gives the final level of granularity by designating the authorized entity (could be a private list of the previous tier) to contact the cloud services and cloud storage. This tier establishes the session to enable private services or classified data.

In the component diagram in Figure 6.2, the gateway is normalizing the authentication process by abstracting the ticket (access certification) exchange of different components. This capability results in a universal authentication platform that scales the access control process in a multi-tier architecture. Both Ranger and Knox daemons may represent access control. The gateway contacts the Central Policy Authentication (CPA) to validate the session and set up their level of access. A different access level may attached to different data pipelines.

Meanwhile, the metadata manage their data in tier-4, which requires a high-level data access session. Without the access certificates, each entity would need to authenticate and authorize with other entities on its own. In contrast, by employing the SITS components, it only needs to combine a central authority that issues an access certificate (based on admin policies) that applies to every entity within variant access tiers.

6.5 Discussion

The security of IoT deployment architectures and ITS service trusts have always been labeled as a research concern. Several research papers discuss the privacy issues of the IoT-based cloud systems and ITS [251, 177, 178, 252]. A multi-tier fog computing model was reported [253], along with a traveling plan-aware scheduling scheme that support electric vehicles [254]. However, herein we address the current ITS security status given the broad diversity of existing solutions uniformity associated with the trust concern. This work distinguishes itself by addressing the ITS security challenges and maps cutting-edge BD frameworks to that particular problem domain. Next, we identify a set of main security requirements for effectively achieving the vision of providing a secure ITS service. The security threats concerning each stage are defined accordingly.

To complete our discussion and offer a look of comparison, Table 1 summarizes the main features of ITS mapped to the related frameworks we presented in Section 6.3. Vendors have been categorized in terms of security abstraction, supported ITS operation, security requirements and threats, and their availability type. At first, general ITS stages (levels of service and architectural layers) are represented to refer as one of (i) the sensors and embedded actuators network, (ii) data acquisition systems, (iii) edge IT and processing, (iv) data analytics (cloud and on-premise), and finally (v) data storage representative stage (as it could be implemented within the previous stage). As shown in the table, the data flow from the embedded systems, sensors and actuators to the storage stage through edge IT and datacenter/cloud environment.

Second is the ITS related operation, where transfer load and data preparation, stream processing, data analysis, in-depth analysis, and storage and archive operations take place. The capability of embedded and distributed intelligence in the network is a core architectural component of the IoT-based ITS. So, we aggregate the domain concepts based on grouping the related functionalities into requirements. This in returns, facilitates the design of security solution by addressing the security requirement and threats. The security role that assures the authenticity of the data life-cycle within each stage (the most relevant) is presented as a security abstraction. Finally, we map the BD security frameworks or technology respectively to their security abstraction.

6.6 Summary

Currently, from various considerations, the vendors of ITS appliances prefer to develop proprietary security that reflects their interests. Various communication protocols and standards are typically deployed within each platform. Hence, interconnecting heterogeneous security services provided by different vendors, and providing seamless interactions/interoperations across the available platforms remain a challenging task. A high-level discussion has been proposed to address this issue and solve better the heterogeneity in the security layered. Also, it aids to provide the interoperability for intelligent transportation devices and services from different vendors of technology independent reference architecture.

This study concentrated on security classifications for modern ITS development based on analysis of published big data frameworks and use cases. The work aims to facilitate architecture design and technology selection in the construction of secure ITS applications. As so, a reference architecture was proposed, which consists of functional components of the Apache Hadoop security stack. Also, we implement a mapping between the specification domain requirements and the related frameworks and use cases. Finally, an adapted model that presents a horizontal level of multi-layer security service that realizes a security mesh through ITS was presented.

CHAPTER 7

CONCLUSIONS

One of the significant shifts of next-generation computing technologies will certainly be in the development of Big Data (BD) deployment architectures. Apache Hadoop, the BD landmark, evolved as a widely deployed BD operating system. Its new features include federation structure and its many associated frameworks, which provide Hadoop 3.x with the maturity to serve different markets. This dissertation addresses two leading issues involved in exploiting BD and the large-scale data analytics realm, using the Hadoop platform: (i) Scalability that directly affects the system performance and overall throughput using portable Docker containers; and (ii) Security that spreads the adoption of data protection practices among practitioners using access controls. An Enhanced MapReduce Environment (EME), OPportunistic and Elastic Resource Allocation (OPERA) scheduler, BD Federation Access Broker (BD-FAB), and a Secure Intelligent Transportation System (SITS) of multi-tier architecture for data streaming to cloud computing are the main contribution of this thesis study.

This PhD dissertation has resulted in improving the BD processing environment from two different perspectives.

- **First, from a theoretical perspective,** by providing insights into different BD deployment architectures and potential relationships among them, and by covering a potential data streaming reference model for a revolutionary and trustworthy edge data analytics architecture.

Chapter Two provides a solid background regarding BD resource management, and how we can compare and contrast different BD environments. A significant part of this effort is due to missing standards in comparison to large-scale data processing environments and the wide variety of different technologies of the BD domain.

Chapter Five studies the feasibility and methodology of employing state-of-the-art Hadoop frameworks for the edge data-intensive community, i.e., IoT-to-cloud models. Hence, developing practical applications that make use of BD techniques and deliver trustworthy solutions for the broader data community, such as vehicle clouds.

- **Second, from a practical perspective,** two applied frameworks address the scalability on the base of elastic resource provisioning of containerized resources and security based on access management was conceived during the scope of this PhD thesis.

Chapter Three introduced the OPERA project, which aimed to fill the increasing gap between computation and I/O capacity on high-end workstations. OPERA allows the coexistence of different resource managers (i.e., HTCondor and Hadoop YARN) and shares computation over the same infrastructure. It also enables the automatic adaptation of the cluster size to the resource demand. This hybrid architecture intends to: (i) minimize the capital expenditure on large private infrastructures; (ii) remove the statical partitioning of computing resources; (iii) improve BD performance; and (iv) reduce operational costs.

Chapter Four presents the first study of the data preservation approach on a BD federation scale using a novel access broker for external Hadoop clients. The proposed access broker not only provides fine-grained access control, but also a centralized auditing facility. This feature also provides BD practitioners with an approach toward security breach detection and digital forensic investigations.

7.1 Future Work

Herein, we present a list of different future research directions that can be carried out to continue the work started in this thesis, including:

- **Hybrid access control solutions:** Development in usage control for data privacy and security leads to integrating traditional access control approaches, e.g., ABAC to other context-based encryption technologies. This integration could include functional encryption or other generalizations of identity-based encryption and attribute-based encryption to protect BD in the presence of mutable attributes, which may lead to general advancement in other fields such as IoT.

- **Fine-grained Access Control:** The need to develop active and fine-grained approaches (policies and standards) for BDF is inevitable. The challenge here is to extend previous approaches with new policy editing and presentation tools (classification-based, prohibition-based, time-based, and location-based policies) for flexible and extensible data access. In particular, policies that extend ABAC with stateful access sessions and mutable attributes (i.e., characteristics that change dynamically during the session). This extension is mainly invaluable to advance the authorization mechanisms of the multi-tenant data lake architecture as well as the latest BDF model. Tackling this issue can be achieved by deploying policy templates that extend the primary ACL (or any access decision enforcer) as a specialization of the XACML V3.0 standard.

- **Dynamic Access Control:** More than ever, the scalability and dynamicity of the BD deployment architecture in terms of user and nodes participation are vital for the secure delivery of security provisioning in a federation environment. Different adapted security mechanisms and techniques have been proposed to cope with this concern. However, more research activities are still needed to establish scalable security models and paradigms that must be driven by BD specifications and requirements. Also, new security abstractions for BDF are still required to simplify the task of identifying unfilled gaps. For instance, leveraging new mechanisms of policy enforcement [186] to combine BDF with ABAC [177] could be labeled as future research.

- **Stateful Access Sessions:** This requires restricting BD federation access to those authorized by the succinct gateway. However, current state-of-the-art BD-based encryption technologies are around transparent data encryption. A federation environment tracks the participation state and characteristics of clients and networks, traversing them with mutable attributes (i.e., the values of the attributes change overtime during an access session). This operation requires formulating policy templates on top of standards, such as XACML V3.0 for dynamic authorization with stateful access sessions. This takes place at service level (e.g., data access calls) and does not protect data-in-transit or data-in-process (not even metadata). Attribute-based encryption, in conjunction with the usage control approach and ABAC can be labeled as a future direction for this issue.

Providing new solutions within all of these research directions will promote innovative BDF solutions with advanced security features. It is also expected that the adoption of FACRM use cases will be implemented within our BD opportunistic and elastic resource allocation (OPERA) platform. OPERA architecture (its prototype proposed in [3]) combines the computing power of available (non-dedicated) high-throughput resources to the Hadoop 3 dedicated cluster using Docker containers as worker nodes. Those non-dedicated containers tend to be more vulnerable and requires a robust IAM approach to minimize security threats.

7.2 Broader Impact

Although clear contributions are given in this dissertation regarding BD optimization, this PhD thesis is, by no means, conclusive *per se*, as its findings pave the way for a wide range of revolutionary and state-of-the-art enhancements and future trends within the Hadoop stack and, in a more general scope, the BD realm. Nevertheless, besides the above-mentioned future work, other future trends and research directions that need to be tackled to improve state-of-the-art BD deployment architectures include:

- *Continue expanding the BD paradigms landscape*

  BD platforms are evolving rapidly and require new computing models to satisfy large-scale data applications. BD architectures will continue leveraging current trends and technologies from the broader computing community. This trend not only yields more significant computational power to accommodate the massive volumes of generated data, but also achieve a financial cut in infrastructure expenditure. Hence, expressing distributed data analytics over new computing models will continue to be an important research direction.

- *Enabling shared infrastructures*

  In prevailing data centers, it is common to create infrastructures silos to accommodate different clusters and afford various frameworks. This approach includes statically partitioning the resources among different resource managers within the physical confines of an enterprise. Modern platforms should overcome this fragmentation by supporting the dynamic sharing of computation and removing these barriers. Therefore, pilot job abstractions and bridge platforms can be employed to enable the co-existence of several schedulers on the same physical infrastructure. This direction provides a unified data center (distributed model) for an enterprise that serves various frameworks over the same bare-metal hardware.

- *Acquiring bursting architectures*

  The search for a cost-effective way to manage the increasing complexity of large-scale data processing could lead, during peak utilization, to the lease of temporary off-premise resources to boost the overall capacity. This trend is particularly promising within cloud implementations (in both centralized and decentralized forms) offering both computing and storage of highly reliable, scalable, and flexible resources. This mainstream approach has gained momentum in data-intensive computing [255, 256].

- *Achieving data-aware dynamic resource provisioning*

  Future platforms should address the requirements to efficiently attend to the fluctuating peak demands of BD workloads and tasks. Dynamic provisioning (aka, provisioning on-demand (POD), defines the ability of the cluster to automatically allocate the needed resources to the processes (dynamically and elastically) at runtime. It also describes the capability to deploy a solution by optimizing the infrastructure in responding to the rapid change in job conditions. POD is considered as a remarkable trend in current solutions with many potentials for future enhancements. Thus, dynamic resource provisioning should be the determinant for scheduler responsivity, variability, resilience, and reliability. The provisioning decisions should also consider the data-locality and prevent over-provisioning. In this regard, serverless computing platforms may be labeled as a potential execution model for future BD architectures.

- *Improving the ability to handle failures and security*

  The ability to work under unfavorable conditions and handle such situations is critical for future data-intensive platforms. Hadoop MR failures are traditionally handled through the re-execution (reassignment) of failed tasks . Many additional measurements can be integrated to ensure a reliable QoS in other deployment architectures. For instance, the use of supernodes [257] in decentralized environments, or optimized speculation mechanisms [258] for short turnaround jobs. Further work includes leveraging the checkpoint to start speculative task execution [259]. Besides, aiming to accelerate and secure data exchange among nodes in these environments, a specific data-exchange channel that is established by nodes wishing to complete their data sets can be created on-demand privately. These private channels can be launched end-to-end for massive data transformation as an adapted application socket [1]. To illustrate, Oracle defines application sockets1 as ”one endpoint of a two-way communication link between two programs running on the network.” As sockets are bound to a port number, the communication protocols (mainly TCP/IP) can identify the direction of the sent data and secure it (e.g., encryption). Thus, application sockets enhance data privacy in a secure channel and speed up data transfer, without interfacing to the primary service messaging (traffic) of the local host.

- *Event-driven manageable configurations*

  Deploying a BD cluster includes setting up and tuning autoscaling policies and computing nodes by application developers and infrastructure operators. However, there can be large variations among the implemented BD frameworks/workloads, requiring the continuous effort of configuring and managing runtime of the the application’s runtime. Sizing, provi- sioning, and scaling the resources, besidesas well as, managing the updates, security patches, and monitoring the whole infrastructure performance and availability, are all a downstream effect. Addressing these issues by abstracting away the cluster management will beis essential in when implementing novel platforms. This direction, mainly, aims to shrink the amount of time it takes to provision provide computation. Utilizing functions as a service and service-oriented architectures by launching servers that work almost identically with predictable configurations, could be the key in to spinning up configurable instances.

[1]`https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html`

APPENDIX A

APPENDIX

Algorithm 2: Two methods performance relationship of read benchmark with R tool

```
initialization;
while
$m1= c(23.64,47.45,69.63,94.23,120.62)$
$m2= c(25.07,49.36,72.05,98.1,127.85)$
$m= c(m1,m2)$
 do
    x= seq(100,500,100) x1= c(x,x) grupo= as.factor(c(rep(”A”,5),rep(”B”,5)))
      grupo= relevel(grupo,ref=”A”)
    postscript(”graficoferas.eps”) plot(x,m1,col=”red”,ylim= c(0,130),xlim=
      c(100,500),xlab= ”Size of the file (in MB)”, ylab= ”Operation Time (in
      Seconds)”,pch=15) points(x,m2,col=”blue”,pch=17)
    legend(”topleft”,legend= c(”Native WebHDFS”,”BDF Access
      Broker”),col=c(”Red”,”Blue”),lwd=1, lty=c(0,0), pch=c(15,17),cex = 0.75)
    abline(lm(m2 -1+x)) abline(lm(m1 -1+x))
    dev.off()
    mod1= lm(m1 -1+x) mod2= lm(m2 -1+x)
    mod= lm(m -1+x1+grupo)
end
```

A.0.1 Appendix B

Algorithm 3: Two methods performance relationship of write benchmark with R tool

```
initialization;
while
m1= c(16.55, 33.9, 48.7, 65.9, 84.43)
m2= c(21, 43, 61.8, 83.69, 107.2)
m= c(m1,m2)
do
    x= seq(100,500,100) x1= c(x,x) grupo= as.factor(c(rep(”A”,5),rep(”B”,5)))
      grupo= relevel(grupo,ref=”A”)
    postscript(”graficoferas.eps”) plot(x,m1,col=”red”,ylim= c(0,130),xlim=
      c(100,500),xlab= ”Size of the file (in MB)”, ylab= ”Operation Time (in
      Seconds)”,pch=15) points(x,m2,col=”blue”,pch=17)
    legend(”topleft”,legend= c(”Native WebHDFS”,”BDF Access
      Broker”),col=c(”Red”,”Blue”),lwd=1, lty=c(0,0), pch=c(15,17),cex = 0.75)
    abline(lm(m2 -1+x)) abline(lm(m1 -1+x))
    dev.off()
    mod1= lm(m1 -1+x) mod2= lm(m2 -1+x)
    mod= lm(m -1+x1+grupo)
end
```

A.0.2 Appendix C

Algorithm 4: Multidecision approach for weighting security criteria in the VC framework

```
1  Require: Criterion functions $f(y)_i$
2  Determine best $\phi_i^*$ and worst $\widehat{\phi_i}$ values $\forall \quad f(y)_i$;
3  if $i^{th}$ function represents a benefit then
4      $\phi^* = \max(\phi_j)$, $\widehat{\phi} = \min(\phi_j)$
5  else
6      $\phi^* = \min(\phi_j)$, $\widehat{\phi} = \max(\phi_j)$
7  end
8  Compute both values $S_j$ and $R_j$.
9  Calculate the comprehensive sorting index $I_j$
10 Rank the fuzzy values $R$, $S$ and $I$ in ascending order.
11 Select the $\min(I_j)$ that represent the best ranked; the alternative $A^{(1)}$ is proposed as a
     compromise solution.
12 if $C1$ is Acceptable advantage then
13     $I(A^{(2)}–I(A^{(1)}) \geq \frac{1}{m-1}$, where $A^{(2)}$ is the alternative with the second position in the
         ranking list by $I$.
14 end
15 if $C2$ is Acceptable stability in decision making then
16     The alternative $A^{(1)}$ must also be the best ranked by $T$ or/and $S$.
17 end
```

Bibliography

[1] F. M. Awaysheh, J. C. Cabaleiro, T. F. Pena, and M. Alazab, "Poster: A pluggable authentication module for big data federation architecture," in *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*. ACM, 2019, pp. 223–225.

[2] F. Awaysheh, J. C. Cabaleiro, T. F. Pena, and M. Alazab, "Big data security frameworks meet the intelligent transportation systems trust challenges," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 807–813.

[3] F. M. Awaysheh, T. F. Pena, and J. C. Cabaleiro, "EME: An automated, elastic and efficient prototype for provisioning hadoop clusters on-demand," in *The 7th International Conference on Cloud Computing and Services Science, CLOSER*, 2017, pp. 709–714.

[4] F. M. Awaysheh, J. C. Cabaleiro, and T. Pena, "OPERA-P: an adaptive scheduler for dynamically provissioning big data frameworks on-demand," in *The third international workshop HTCondor at DESY Hamburg*. HTCondor, 2017.

[5] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[6] D. Cutting, M. Cafarella, "Apache Hadoop," http://hadoop.apache.org, 2006, accessed: 2019-06-10.

[7] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online." in *Nsdi*, vol. 10, 2010, p. 20.

[8] J. Myung, J. Shim, J. Yeon, and S.-g. Lee, “Handling data skew in join algorithms using MapReduce,” *Expert Systems with Applications*, vol. 51, pp. 286–299, 2016.

[9] Q. Althebyan, O. ALQudah, Y. Jararweh, and Q. Yaseen, “Multi-threading based Map Reduce tasks scheduling,” in *Information and Communication Systems (ICICS), 2014 5th International Conference on*. IEEE, 2014, pp. 1–6.

[10] J. Urbani, A. Margara, C. Jacobs, S. Voulgaris, and H. Bal, “Ajira: a lightweight distributed middleware for MapReduce and stream processing,” in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 545–554.

[11] X. Hua, H. Wu, Z. Li, and S. Ren, “Enhancing throughput of the Hadoop distributed file system for interaction-intensive tasks,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 8, pp. 2770–2779, 2014.

[12] Y. Jiang, Y. Zhu, W. Wu, and D. Li, “Makespan minimization for MapReduce systems with different servers,” *Future Generation Computer Systems*, vol. 67, pp. 13–21, 2017.

[13] O. Tatebe, K. Hiraga, and N. Soda, “Gfarm grid file system,” *New Generation Computing*, vol. 28, no. 3, pp. 257–275, 2010.

[14] I. El-Helw, R. Hofman, and H. E. Bal, “Glasswing: accelerating MapReduce on multicore and many-core clusters,” in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*. ACM, 2014, pp. 295–298.

[15] T. White, *Hadoop: The Definitive Guide*, 4th ed. O’Reilly Media, Inc., 2015.

[16] D. Borthakur, “HDFS architecture guide,” *Hadoop Apache Project*, p. 39, 2008.

[17] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, “Apache Hadoop YARN: Yet another resource negotiator,” in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 5.

[18] S. Sakr, *Big Data 2.0 Processing Systems: A Survey*. Springer, 2016.

[19] M. Chen, S. Mao, and Y. Liu, “Big data: A survey,” *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[20] C. P. Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” *Information Sciences*, vol. 275, pp. 314–347, 2014.

[21] S. Sakr, A. Liu, and A. G. Fayoumi, “The family of MapReduce and large-scale data processing systems,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 11, 2013.

[22] S. N. Khezr and N. J. Navimipour, “MapReduce and its applications, challenges, and architecture: a comprehensive review and directions for future research,” *Journal of Grid Computing*, vol. 15, no. 3, pp. 295–321, 2017.

[23] R. Li, H. Hu, H. Li, Y. Wu, and J. Yang, “MapReduce parallel programming model: a state-of-the-art survey,” *International Journal of Parallel Programming*, vol. 44, no. 4, pp. 832–866, 2016.

[24] W. J. Kaufmann and L. L. Smarr, *Supercomputing and the Transformation of Science*. WH Freeman & Co., 1992.

[25] S. Krishnan, M. Tatineni, and C. Baru, “myHadoop: Hadoop-on-demand on traditional HPC resources,” *San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego*, 2011.

[26] S. Michael, A. Thota, and R. Henschel, “HPCHadoop: A framework to run Hadoop on Cray X-series supercomputers,” *Cray USer Group (CUG)*, 2014.

[27] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan, “Mariane: MapReduce implementation adapted for HPC environments,” in *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*. IEEE, 2011, pp. 82–89.

[28] M. Lu, Y. Liang, H. P. Huynh, Z. Ong, B. He, and R. S. M. Goh, “Mrphi: An optimized MapReduce framework on Intel Xeon Phi coprocessors,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 3066–3078, 2015.

[29] S. Sehrish, G. Mackey, P. Shang, J. Wang, and J. Bent, “Supporting HPC analytics applications with access patterns using data restructuring and data-centric scheduling techniques in MapReduce,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 158–169, 2013.

[30] X. Yang, N. Liu, B. Feng, X.-H. Sun, and S. Zhou, "PortHadoop: Support direct HPC data processing in Hadoop," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 223–232.

[31] S. Sehrish, G. Mackey, J. Wang, and J. Bent, "Mrap: a novel MapReduce-based framework to support HPC analytics applications with access patterns," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 107–118.

[32] a. Steve Loughran, et, "Apache Slider," https://slider.incubator.apache.org/, 2014, accessed: 2018-05-02.

[33] A. Luckow, "SAGA-Hadoop ," https://github.com/drelu/saga-hadoop, 2013, accessed: 2018-05-02.

[34] X. Lu, F. Liang, B. Wang, L. Zha, and Z. Xu, "DataMPI: extending MPI to Hadoop-like big data computing," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 829–838.

[35] Y. Wang, G. Agrawal, T. Bicer, and W. Jiang, "Smart: A MapReduce-like framework for in-situ scientific analytics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 51.

[36] T. Gao, Y. Guo, B. Zhang, P. Cicotti, Y. Lu, P. Balaji, and M. Taufer, "Mimir: Memory-efficient and scalable MapReduce for large supercomputing systems," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 1098–1108.

[37] Y. Guo, W. Bland, P. Balaji, and X. Zhou, "Fault tolerant MapReduce-MPI for HPC clusters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 34.

[38] H. Lin, Z. Su, X. Meng, X. Jin, Z. Wang, W. Han, H. An, M. Chi, and Z. Wu, "Combining Hadoop with MPI to solve metagenomics problems that are both data-and compute-intensive," *International Journal of Parallel Programming*, pp. 1–14, 2017.

[39] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, "G-Hadoop: MapReduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 739–750, 2013.

[40] J. Wang and X. Li, “Task scheduling for MapReduce in heterogeneous networks,” *Cluster Computing*, vol. 19, no. 1, pp. 197–210, 2016.

[41] E. Ghazisaeedi and C. Huang, “GreenMap: Green mapping of MapReduce-based virtual networks onto a data center network and managing incast queueing delay,” *Computer Networks*, vol. 112, pp. 345–359, 2017.

[42] N. Zacheilas and V. Kalogeraki, “Chess: Cost-effective scheduling across multiple heterogeneous MapReduce clusters,” in *Autonomic Computing (ICAC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 65–74.

[43] R. Platania, S. Shams, C.-H. Chiu, N. Kim, J. Kim, and S.-J. Park, “Hadoop-based replica exchange over heterogeneous distributed cyberinfrastructures,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 4, 2017.

[44] A. Luckow, I. Paraskevakos, G. Chantzialexiou, and S. Jha, “Hadoop on HPC: integrating Hadoop and pilot-based dynamic resource management,” in *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, 2016, pp. 1607–1616.

[45] M. Wasi-ur Rahman, X. Lu, N. S. Islam, R. Rajachandrasekar, and D. K. Panda, “High-performance design of YARN MapReduce on modern HPC clusters with Lustre and RDMA,” in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*. IEEE, 2015, pp. 291–300.

[46] M. Wasi-ur Rahman, N. S. Islam, X. Lu, and D. K. D. Panda, “A comprehensive study of MapReduce over Lustre for intermediate data placement and shuffle strategies on HPC clusters,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 633–646, 2017.

[47] J. Veiga, R. R. Expósito, G. L. Taboada, and J. Touriño, “Analysis and evaluation of MapReduce solutions on an HPC cluster,” *Computers & Electrical Engineering*, vol. 50, pp. 200–216, 2016.

[48] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica *et al.*, “Above the clouds: A Berkeley view of cloud computing,” Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[49] "Amazon MapReduce website:," https://aws.amazon.com/emr/, accessed: 2018-05-02.

[50] "Microsoft Azure HDInsight website," https://azure.microsoft.com/en-us/services/hdinsight/, accessed: 2018-05-02.

[51] "Google Cloud Dataproc website," https://cloud.google.com/dataproc/, accessed: 2018-05-02.

[52] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of Big Data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[53] J. Tan, X. Pan, E. Marinelli, S. Kavulya, R. Gandhi, and P. Narasimhan, "Kahuna: Problem diagnosis for MapReduce-based cloud computing environments," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*. IEEE, 2010, pp. 112–119.

[54] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "MapReduce in the clouds for science," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 565–572.

[55] P. Lama and X. Zhou, "Aroma: Automated resource allocation and configuration of MapReduce environment in the cloud," in *Proc. of the 9th Int. Conf. on Autonomic Computing*. ACM, 2012, pp. 63–72.

[56] H. Liu and D. Orban, "Cloud MapReduce: A MapReduce implementation on top of a cloud operating system," in *Proc. of the 11th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid computing*. IEEE Computer Society, 2011, pp. 464–474.

[57] T. Gunarathne, B. Zhang, T.-L. Wu, and J. Qiu, "Scalable parallel computing on clouds using Twister4Azure iterative MapReduce," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1035–1048, 2013.

[58] B. Palanisamy, A. Singh, L. Liu, and B. Langston, "Cura: A cost-optimized model for MapReduce in a cloud," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*. IEEE, 2013, pp. 1275–1286.

[59] B. T. Rao and L. S. S. Reddy, "Survey on improved scheduling in Hadoop MapReduce in cloud environments," *arXiv preprint arXiv:1207.0780*, 2012.

[60] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 265–278.

[61] B. Palanisamy, A. Singh, L. Liu, and B. Jain, “Purlieus: locality-aware resource allocation for MapReduce in a cloud,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 58.

[62] D. Dahiphale, R. Karve, A. V. Vasilakos, H. Liu, Z. Yu, A. Chhajer, J. Wang, and C. Wang, “An advanced MapReduce: cloud MapReduce, enhancements and applications,” *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 101–115, 2014.

[63] L. Yazdanov, M. Gorbunov, and C. Fetzer, “EHadoop: network I/O aware scheduler for elastic MapReduce cluster,” in *Cloud Computing (CLOUD), 2015 IEEE 8th Int. Conf. on*. IEEE, 2015, pp. 821–828.

[64] J. Veiga, R. R. Expósito, G. L. Taboada, and J. Touriño, “Flame-mr: An event-driven architecture for MapReduce applications,” *Future Generation Computer Systems*, vol. 65, pp. 46–56, 2016.

[65] C. Xu, J. Yang, K. Yin, and H. Yu, “Optimal construction of virtual networks for cloud-based MapReduce workflows,” *Computer Networks*, vol. 112, pp. 194–207, 2017.

[66] Q. Chen and M. Guo, “MapReduce for cloud computing,” in *Task Scheduling for Multi-core and Parallel Architectures*. Springer, 2017, pp. 173–198.

[67] D. Ghoshal and L. Ramakrishnan, “Provisioning, placement and pipelining strategies for data-intensive applications in cloud environments,” in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 325–330.

[68] W. Jing, D. Tong, Y. Wang, J. Wang, Y. Liu, and P. Zhao, “MaMR: High-performance MapReduce programming model for material cloud applications,” *Computer Physics Communications*, vol. 211, pp. 79–87, 2017.

[69] H. Zhang, H. Huang, and L. Wang, “MRapid: An efficient short job optimizer on Hadoop,” in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 459–468.

[70] Y. Shi, K. Zhang, L. Cui, L. Liu, Y. Zheng, S. Zhang, and H. Yu, “MapReduce short jobs optimization based on resource reuse,” *Microprocessors and Microsystems*, vol. 47, pp. 178–187, 2016.

[71] W. Lu, L. Chen, H. Yuan, W. Xing, L. Wang, and Y. Yang, “Improving MapReduce performance by using a new partitioner in YARN,” in *The 23rd International Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems*, 2017, pp. 24–33.

[72] P. Wangsom, K. Lavangnananda, and P. Bouvry, “Measuring data locality ratio in virtual MapReduce cluster using WorkflowSim,” in *Computer Science and Software Engineering (JCSSE), 2017 14th International Joint Conference on*. IEEE, 2017, pp. 1–6.

[73] X. Ma, X. Fan, J. Liu, H. Jiang, and K. Peng, “vLocality: Revisiting data locality for MapReduce in virtualized clouds,” *IEEE Network*, vol. 31, no. 1, pp. 28–35, 2017.

[74] L. Yang, Y. Dai, and B. Zhang, “Optimized speculative execution to improve performance of MapReduce jobs on virtualized computing environment,” *Mathematical Problems in Engineering*, vol. 2017, 2017.

[75] D. Logothetis and K. Yocum, “Ad-hoc data processing in the cloud,” *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1472–1475, 2008.

[76] X. Shi, B. Cui, G. Dobbie, and B. C. Ooi, “UniAD: A unified ad hoc data processing system,” *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 1, p. 6, 2017.

[77] M. Hamdaqa, M. M. Sabri, A. Singh, and L. Tahvildari, “Adoop: MapReduce for ad-hoc cloud computing,” in *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2015, pp. 26–34.

[78] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[79] P. A. Costa, X. Bai, F. M. Ramos, and M. Correia, “Medusa: An efficient cloud fault-tolerant MapReduce,” in *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE, 2016, pp. 443–452.

[80] T. Gouasmi, W. Louati, and A. H. Kacem, “Cost-efficient distributed MapReduce job scheduling across cloud federation,” in *Services Computing (SCC), 2017 IEEE Int. Conf. on*. IEEE, 2017, pp. 289–296.

[81] C.-Y. Wang, T.-L. Tai, S. Jui-Shing, C. Jyh-Biau, and S. Ce-Kuen, “Federated MapReduce to transparently run applications on multicluster environment,” in *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, pp. 296–303.

[82] D. Jayalakshmi and R. Srinivasan, “Simulation of MapReduce across geographically distributed datacentres using cloudsim,” in *International Conference on Distributed Computing and Internet Technology*. Springer, 2017, pp. 70–81.

[83] O. Tomarchio, G. Di Modica, M. Cavallo, and C. Polito, “A hierarchical Hadoop framework to handle big data in geo-distributed computing environments,” *International Journal of Information Technologies and Systems Approach (IJITSA)*, vol. 11, no. 1, pp. 16–47, 2018.

[84] A. Iordache, C. Morin, N. Parlavantzas, E. Feller, and P. Riteau, “Resilin: Elastic MapReduce over multiple clouds,” in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 261–268.

[85] Q. Zhang, L. Liu, K. Lee, Y. Zhou, A. Singh, N. Mandagere, S. Gopisetty, and G. Alatorre, “Improving Hadoop service provisioning in a geographically distributed cloud,” in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 432–439.

[86] F. Berman, G. Fox, and A. J. Hey, *Grid computing: making the global infrastructure a reality*. John Wiley and sons, 2003, vol. 2.

[87] D. P. Anderson, “Boinc: A system for public-resource computing and storage,” in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. IEEE, 2004, pp. 4–10.

[88] G. Fedak, H. He, and F. Cappello, “BitDew: a programmable environment for large-scale data management and distribution,” in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*. IEEE, 2008, pp. 1–12.

[89] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[90] K. Lee and R. Figueiredo, "MapReduce on opportunistic resources leveraging resource availability," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 435–442.

[91] Y. Ji, L. Tong, T. He, J. Tan, K.-w. Lee, and L. Zhang, "Improving multi-job MapReduce scheduling in an opportunistic environment," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 9–16.

[92] H. Jin, X. Yang, X.-H. Sun, and I. Raicu, "Adapt: Availability-aware MapReduce data placement for non-dedicated distributed computing," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 516–525.

[93] "Facebook, under the hood: Scheduling MapReduce jobs more efficiently with Corona," https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920, 2012.

[94] "Hadoop on-demand (HOD) scheduler by Apache Hadoop website:," https://hadoop.apache.org/docs/r1.2.1/hod_scheduler.html, accessed: 2018-05-02.

[95] C. He, D. Weitzel, D. Swanson, and Y. Lu, "Hog: Distributed Hadoop MapReduce on the grid," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*. IEEE, 2012, pp. 1276–1283.

[96] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, "Moon: Mapreduce on opportunistic environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 95–106.

[97] J. C. Anjos, I. Carrera, W. Kolberg, A. L. Tibola, L. B. Arantes, and C. R. Geyer, "MRA++: Scheduling and data placement on MapReduce for heterogeneous environments," *Future Generation Computer Systems*, vol. 42, pp. 22–35, 2015.

[98] L. Lu, H. Jin, X. Shi, and G. Fedak, "Assessing MapReduce for Internet computing: a comparison of Hadoop and BitDew-MapReduce," in *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE Computer Society, 2012, pp. 76–84.

[99] M. Zaharia, "The Hadoop fair scheduler," 2010.

[100] M. Moca, G. C. Silaghi, and G. Fedak, "Distributed results checking for MapReduce in volunteer computing," in *Parallel and distributed processing workshops and Phd Forum (IPDPSW), 2011 IEEE international symposium on*. IEEE, 2011, pp. 1847–1854.

[101] B. Tang, M. Moca, S. Chevalier, H. He, and G. Fedak, "Towards MapReduce for desktop grid computing," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*. IEEE, 2010, pp. 193–200.

[102] B. Tang, M. Tang, G. Fedak, and H. He, "Availability/network-aware MapReduce over the Internet," *Information Sciences*, vol. 379, pp. 94–111, 2017.

[103] E. Kijsipongse, U. Suriya *et al.*, "Scaling Hadoop clusters with virtualized volunteer computing environment," in *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*. IEEE, 2014, pp. 146–151.

[104] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments." in *OSDI*, vol. 8, 2008, p. 7.

[105] F. Costa, L. Veiga, and P. Ferreira, "Vmr: volunteer MapReduce over the large scale Internet," in *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM, 2012, p. 1.

[106] "MapReduce-BitDew website:," http://freecode.com/projects/mapreduce-bitdew, accessed: 2018-05-02.

[107] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[108] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating MapReduce on virtual machines: The Hadoop case," in *Cloud Computing*. Springer, 2009, pp. 519–528.

[109] Y.-W. Jung, J.-Y. Lee, and K.-D. Jung, “Map Reduce-based P2P DBaaS hub system,” *International journal of advanced smart convergence*, vol. 5, no. 1, pp. 16–22, 2016.

[110] J. Bhimani, Z. Yang, M. Leeser, and N. Mi, “Accelerating big data applications using lightweight virtualization framework on enterprise cloud,” in *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*. IEEE, 2017, pp. 1–7.

[111] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, “Performance evaluation of container-based virtualization for high performance computing environments,” in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013, pp. 233–240.

[112] R. Morabito, J. Kjällman, and M. Komu, “Hypervisors vs. lightweight virtualization: a performance comparison,” in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 386–393.

[113] A. Qin, D. Tu, C. Shu, and C. Gao, “XConveryer: Guarantee Hadoop throughput via lightweight OS-level virtualization,” in *Grid and Cooperative Computing, 2009. GCC’09. Eighth International Conference on*. IEEE, 2009, pp. 299–304.

[114] M. T. Chung, N. Quang-Hung, M.-T. Nguyen, and N. Thoai, “Using Docker in high performance computing applications,” in *Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on*. IEEE, 2016, pp. 52–57.

[115] J. Rey, M. Cogorno, S. Nesmachnow, and L. A. Steffenel, “Efficient prototyping of fault tolerant MapReduce applications with Docker-Hadoop,” in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 369–376.

[116] R. Rosen, “Linux containers and the future cloud,” *Linux J*, vol. 2014, no. 240, 2014.

[117] H. Chung and Y. Nah, “Performance comparison of distributed processing of large volume of data on top of Xen and Docker-based virtual clusters,” in *International Conference on Database Systems for Advanced Applications*. Springer, 2017, pp. 103–113.

[118] W. Chen, J. Rao, and X. Zhou, “Addressing memory pressure in data-intensive parallel programs via container based virtualization,” in *Autonomic Computing (ICAC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 197–202.

[119] C. Nvidia, “Nvidia CUDA programming guide,” *Nvidia Corporation*, vol. 120, no. 18, 2015.

[120] L. Chen and G. Agrawal, “Optimizing MapReduce for GPUs with effective shared memory usage,” in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*. ACM, 2012, pp. 199–210.

[121] C.-H. Wang, C.-K. Yang, W.-C. Liao, R.-I. Chang, and T.-T. Wei, “Coupling GPU and MPTCP to improve Hadoop/MapReduce performance,” in *Intelligent Green Building and Smart Grid (IGBSG), 2016 2nd International Conference on*. IEEE, 2016, pp. 1–6.

[122] Y. Liu, H.-W. Tseng, and S. Swanson, “Spmario: Scale up MapReduce with I/O-oriented scheduling for the GPU,” in *Computer Design (ICCD), 2016 IEEE 34th Int. Conf. on*. IEEE, 2016, pp. 384–387.

[123] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, “Mars: a MapReduce framework on graphics processors,” in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 260–269.

[124] Z. Qiao, S. Liang, H. Jiang, and S. Fu, “A customizable MapReduce framework for complex data-intensive workflows on GPUs,” in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.

[125] J. Stuart, J. D. Owens *et al.*, “Multi-GPU MapReduce on GPU clusters,” in *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*. IEEE, 2011, pp. 1068–1079.

[126] Y. Chen, Z. Qiao, H. Jiang, K.-C. Li, and W. W. Ro, “Mgmr: Multi-GPU based MapReduce,” in *International Conference on Grid and Pervasive Computing*. Springer, 2013, pp. 433–442.

[127] Y. Guo, W. Liu, B. Gong, G. Voss, and W. Muller-Wittig, “GCMR: A GPU cluster-based MapReduce framework for large-scale data processing,” in *High Performance Computing and Communications (HPCC_EUC), IEEE 10th International Conference on*. IEEE, 2013, pp. 580–586.

[128] D. Loghin, “Efficient time-energy execution of data-parallel applications on heterogeneous systems with GPU,” Ph.D. dissertation, 2017.

[129] L. Chen, X. Huo, and G. Agrawal, “Accelerating MapReduce on a coupled CPU-GPU architecture,” in *Proc. of the Int. Conf. on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 25.

[130] H. Jiang, Y. Chen, Z. Qiao, T.-H. Weng, and K.-C. Li, “Scaling up MapReduce-based big data processing on multi-GPU systems,” *Cluster Computing*, vol. 18, no. 1, pp. 369–383, 2015.

[131] L. A. Steffenel, O. Flauzac, A. Schwertner Charao, P. Pitthan Barcelos, B. Stein, S. Nesmachnow, M. Kirsch Pinheiro, and D. Diaz, “Per-mare: Adaptive deployment of MapReduce over pervasive grids,” in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*. IEEE, 2013, pp. 17–24.

[132] L. A. Steffenel and M. K. Pinheiro, “Leveraging data-intensive applications on a pervasive computing platform: The case of MapReduce,” *Procedia Computer Science*, vol. 52, pp. 1034–1039, 2015.

[133] G. W. Cassales, A. S. Charão, M. Kirsch-Pinheiro, C. Souveyet, and L.-A. Steffenel, “Improving the performance of Apache Hadoop on pervasive environments through context-aware scheduling,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, no. 3, pp. 333–345, 2016.

[134] W. Wang, M. Barnard, and L. Ying, “Decentralized scheduling with data locality for data-parallel computation on peer-to-peer networks,” in *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, 2015, pp. 337–344.

[135] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos, “Misco: a MapReduce framework for mobile systems,” in *Proceedings of the 3rd international conference on pervasive technologies related to assistive environments*. ACM, 2010, p. 32.

[136] E. E. Marinelli, “Hyrax: Cloud computing on mobile devices using MapReduce,” DTIC Document, Tech. Rep., 2009.

[137] J. George, C.-A. Chen, R. Stoleru, and G. Xie, "Hadoop MapReduce for mobile clouds," *IEEE Transactions on Cloud Computing*, 2017.

[138] R.-J. Huang and C.-H. Wu, "A MapReduce framework for heterogeneous mobile devices," in *Intelligent Green Building and Smart Grid (IGBSG), 2014 International Conference on*. IEEE, 2014, pp. 1–5.

[139] S. Kailasam, P. Dhawalia, S. Balaji, G. Iyer, and J. Dharanipragada, "Extending MapReduce across clouds with BStream," *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 362–376, 2014.

[140] H. Lin, X. Ma, and W.-c. Feng, "Reliable MapReduce computing on opportunistic resources," *Cluster Computing*, vol. 15, no. 2, pp. 145–161, 2012.

[141] H. Zhang, C. Hao, Y. Wu, and M. Li, "Towards a scalable and energy-efficient resource manager for coupling cluster computing with distributed embedded computing," *Cluster Computing*, pp. 1–14, 2017.

[142] A. Jonathan, M. Ryden, K. Oh, A. Chandra, and J. Weissman, "Nebula: Distributed edge cloud for data intensive computing," *IEEE Tran. on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3229–3242, 2017.

[143] C. Jayalath, J. Stephen, and P. Eugster, "From the cloud to the atmosphere: Running MapReduce across data centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 74–87, 2014.

[144] B. Tang, H. He, and G. Fedak, "HybridMR: a new approach for hybrid MapReduce combining desktop grid and cloud infrastructures," *Concurrency and computation: Practice and experience*, vol. 27, no. 16, pp. 4140–4155, 2015.

[145] Y. Wang, J. Wei, and Y. Duan, "Securing MapReduce result integrity via verification-based integrity assurance framework," *International Journal of Grid and Distributed Computing (IJGDC)*, vol. 7, no. 6, pp. 53–70, 2014.

[146] G. Pareek, C. Goyal, and M. Nayal, "A result verification scheme for MapReduce having untrusted participants," in *Intelligent Distributed Computing*. Springer, 2015, pp. 11–19.

[147] R. Gu, X. Yang, J. Yan, Y. Sun, B. Wang, C. Yuan, and Y. Huang, “SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters,” *Journal of parallel and distributed computing*, vol. 74, no. 3, pp. 2166–2179, 2014.

[148] Y. Liu, W. Wei, and Y. Zhang, “Checkpoint and replication oriented fault tolerant mechanism for MapReduce framework,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 2, pp. 1029–1036, 2014.

[149] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[150] M. Malik, K. Neshatpour, S. Rafatirad, and H. Homayoun, “Hadoop workloads characterization for performance and energy efficiency optimizations on microservers,” *IEEE Transactions on Multi-Scale Computing Systems*, 2017.

[151] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, “The role of big data in smart city,” *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, 2016.

[152] “Installing, running, using Docker on mobile devices, Github website,” https://github.com/umiddelb/armhf/wiki/Installing,-running,-using-docker-on-armhf-(ARMv7)-devices, 2015.

[153] D. Beserra, M. K. Pinheiro, C. Souveyet, L. A. Steffenel, and E. D. Moreno, “Performance evaluation of OS-level virtualization solutions for HPC purposes on SoC-based systems,” in *Advanced Information Networking and Applications (AINA), 2017 IEEE 31st Int. Conf. on*. IEEE, 2017, pp. 363–370.

[154] H. Herodotou, F. Dong, and S. Babu, “No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics,” in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 18.

[155] K. Chen, J. Powers, S. Guo, and F. Tian, “Cresp: Towards optimal resource provisioning for MapReduce computing in public clouds,” *IEEE Trans. on Par. and Distributed Systems*, vol. 25, no. 6, pp. 1403–1412, 2014.

[156] P. P. Nghiem and S. M. Figueira, “Towards efficient resource provisioning in MapReduce,” *Journal of Parallel and Distributed Computing*, vol. 95, pp. 29–41, 2016.

[157] T. Honjo and K. Oikawa, “Hardware acceleration of Hadoop MapReduce,” in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 118–124.

[158] G. Ananthanarayanan, C. Douglas, R. Ramakrishnan, S. Rao, and I. Stoica, “True elasticity in multi-tenant data-intensive compute clusters,” in *Proc. 3rd ACM Symposium on Cloud Computing*, 2012, p. 24.

[159] M. N. Durrani and J. A. Shamsi, “Volunteer computing: requirements, challenges, and solutions,” *Journal of Network and Computer Applications*, vol. 39, pp. 369–380, 2014.

[160] M. Conti, S. Giordano, M. May, and A. Passarella, “From opportunistic networks to opportunistic computing,” *IEEE Communications Magazine*, vol. 48, no. 9, 2010.

[161] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: the Condor experience,” *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[162] Apache Software, “Apache Myriad,” http://myriad.apache.org/, 2017, accessed: 2017-01-30.

[163] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, “Moon: MapReduce on opportunistic environments,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 95–106.

[164] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, “Improving MapReduce performance in heterogeneous environments,” in *Osdi*, vol. 8, no. 4, 2008, p. 7.

[165] I. Kurochkin and A. Saevskiy, “Boinc forks, issues and directions of development,” *Procedia Computer Science*, vol. 101, pp. 369–378, 2016.

[166] Apache Knox, “REST API and Application Gateway for the Apache Hadoop Ecosystem,” https://knox.apache.org/, 2019, accessed: 2019-02-10.

[167] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. Ieee, 2010, pp. 1–10.

[168] Apache Hadoop, “Apache Hadoop 3.x HDFS Federation Features,” http://hadoop.apache.org/docs/r3.2.0/hadoop-project-dist/hadoop-hdfs/Federation.html, 2019, accessed: 2019-06-10.

[169] Apache Ranger, “Monitor and manage comprehensive data security,” https://ranger.apache.org/, 2019, accessed: 2019-02-10.

[170] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[171] Apache Hadoop, “HDFS Permissions Guide, Hadoop 3.x,” https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html, 2019, accessed: 2019-06-10.

[172] ——, “Superuser Authorization for Hadoop 3.x cluster,” https:https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/Superusers.html, 2019, accessed: 2019-06-10.

[173] ——, “HDFS operations and their methods,” https://hadoop.apache.org/docs/r3.0.0/api/org/apache/hadoop/fs/FileSystem.html, 2019, accessed: 2019-06-10.

[174] ——, “Hadoop 3.x Capacity Scheduler,intra-queue preemption,” https://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html, 2019, accessed: 2019-06-10.

[175] M. Gupta, F. Patwa, J. Benson, and R. Sandhu, “Multi-layer authorization framework for a representative Hadoop ecosystem deployment,” in *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. ACM, 2017, pp. 183–190.

[176] M. Gupta, F. Patwa, and R. Sandhu, “Object-tagged RBAC model for the Hadoop ecosystem,” in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2017, pp. 63–81.

[177] ——, “An attribute-based access control model for secure big data processing in Hadoop ecosystem,” in *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*. ACM, 2018, pp. 13–24.

[178] P. Colombo and E. Ferrari, “Enhancing MongoDB with purpose-based access control,” *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 591–604, 2017.

[179] ——, “Access control in the era of big data: State of the art and research directions,” in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. ACM, 2018, pp. 185–192.

[180] ——, “Privacy aware access control for big data: a research roadmap,” *Big Data Research*, vol. 2, no. 4, pp. 145–154, 2015.

[181] H. Ulusoy, M. Kantarcioglu, E. Pattuk, and K. Hamlen, “Vigiles: Fine-grained access control for MapReduce systems,” in *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, pp. 40–47.

[182] D. Basin, J. Doser, and T. Lodderstedt, “Model driven security: From UML models to access control infrastructures,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 15, no. 1, pp. 39–91, 2006.

[183] M. Hamdaqa, T. Livogiannis, and L. Tahvildari, “A reference model for developing cloud applications,” in *CLOSER*, 2011, pp. 98–103.

[184] W. Jie, J. Arshad, R. Sinnott, P. Townend, and Z. Lei, “A review of grid authentication and authorization technologies and support for federated access control,” *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, p. 12, 2011.

[185] S. Barker, “The next 700 access control models or a unifying meta-model?” in *Proceedings of the 14th ACM symposium on Access control models and technologies*. ACM, 2009, pp. 187–196.

[186] T. Ahmad, U. Morelli, S. Ranise, and N. Zannone, “A lazy approach to access control as a service (ACaaS) for IoT: An AWS case study,” in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. ACM, 2018, pp. 235–246.

[187] D. Borthakur, “The Hadoop distributed file system: Architecture and design,” *Hadoop Project Website*, vol. 11, no. 2007, p. 21, 2007.

[188] HTCondor, “Deploying High-throughput cluster using HTCondor over HDFS, HTCondor Manual,” http://research.cs.wisc.edu/htcondor/manual/v8.8/index.html, 2019, accessed: 2019-06-10.

[189] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[190] Apache Hadoop Project, “HDFS federation,” https://hadoop.apache.org/docs/current3/hadoop-project-dist/hadoop-hdfs/Federation.html, accessed: 2019-06-13.

[191] ——, “YARN federation,” https://hadoop.apache.org/docs/current3/hadoop-yarn/hadoop-yarn-site/Federation.html, accessed: 2019-06-13.

[192] Apache Hadoop, “REST API and Application Gateway for the Hadoop Ecosystem,” https://knox.apache.org/, 2019, accessed: 2019-06-10.

[193] ——, “Deploying Hadoop 3.x secure mode,” https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/SecureMode.html, 2019, accessed: 2019-06-10.

[194] ——, “Authentication for Hadoop 3.x HTTP Web Consoles,” https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/HttpAuthentication.html, 2019, accessed: 2019-06-10.

[195] ——, “Configure and manage Service Level Authorization for Hadoop 3.x,” https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/ServiceLevelAuth.html, 2019, accessed: 2019-06-10.

[196] Apache Atlas, “Data governance and metadata framework for Hadoop,” https://atlas.apache.org, 2019, accessed: 2019-06-10.

[197] Apache Solr, “Full-text search and real-time indexing,” http://lucene.apache.org/solr/, 2019, accessed: 2019-06-10.

[198] Apache Hadoop, “Transparent data encryption in HDFS,” https://hadoop.apache.org/docs/r3.0.2/hadoop-project-dist/hadoop-hdfs/TransparentEncryption.html, 2019, accessed: 2019-06-10.

[199] Intel, “Big Data: Securing Intel IT’s Apache Hadoop Platform,” http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/

big-data-securing-intel-it-apache-hadoop-platform-paper.pdf, 2016, accessed: 2019-06-10.

[200] Securosis, “Securing Hadoop: Security recommendations for Hadoop environments,” https://securosis.com/assets/library/reports/Securing_Hadoop_Final_V2.pdf, 2016.

[201] D. Das, O. O’Malley, S. Radia, and K. Zhang, “Adding security to Apache Hadoop,” *Hortonworks, IBM*, 2011.

[202] M. Gupta, F. Patwa, and R. Sandhu, “POSTER: Access Control Model for the Hadoop Ecosystem,” in *Proc. of ACM SACMAT . 3 Pages.*, 2017.

[203] O. O’Malley, K. Zhang, S. Radia, R. Marti, and C. Harrell, “Hadoop security design,” *Yahoo, Inc., Tech. Rep*, 2009.

[204] P. P. Sharma and C. P. Navdeti, “Securing big data Hadoop: a review of security issues, threats and solution,” *IJCSIT*, vol. 5, 2014.

[205] M. Gupta and R. Sandhu, “The GURA$_G$ administrative model for user and group attribute assignment,” in *Proc. of NSS*. Springer, 2016, pp. 318–332.

[206] X. Jin, R. Krishnan, and R. Sandhu, “A unified attribute-based access control model covering DAC, MAC and RBAC,” in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2012, pp. 41–55.

[207] H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, and E. Pattuk, “GuardMR: Fine-grained security policy enforcement for MapReduce systems,” in *Proc. of ACM ASIACCS*, 2015, pp. 285–296.

[208] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, “Toward efficient and privacy-preserving computing in big data era,” *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.

[209] J. Soria-Comas and J. Domingo-Ferrer, “Big data privacy: challenges to privacy principles and models,” *Data Science and Engineering*, vol. 1, no. 1, pp. 21–28, 2016.

[210] O. Tene and J. Polonetsky, “Big data for all: Privacy and user control in the age of analytics,” *Nw. J. Tech. & Intell. Prop.*, vol. 11, p. xxvii, 2012.

[211] P. Colombo and E. Ferrari, “Access control technologies for big data management systems: literature review and future trends,” *Cybersecurity*, vol. 2, no. 1, p. 3, 2019.

[212] D. Kulkarni, “A fine-grained access control model for key-value systems,” in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 161–164.

[213] Y. Shalabi and E. Gudes, “Cryptographically enforced role-based access control for NoSQL distributed databases,” in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2017, pp. 3–19.

[214] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” *Journal of cryptology*, vol. 26, no. 2, pp. 191–224, 2013.

[215] M. Nabeel and E. Bertino, “Privacy preserving delegated access control in public clouds,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2268–2280, 2013.

[216] R. V. Nehme, H.-S. Lim, and E. Bertino, “Fence: Continuous access control enforcement in dynamic data stream environments,” in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 243–254.

[217] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, “DPBSV–an efficient and secure scheme for big sensing data stream,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 246–253.

[218] B. Carminati, E. Ferrari, J. Cao, and K. L. Tan, “A framework to enforce access control over data streams,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 3, p. 28, 2010.

[219] P. Leitner, C. Inzinger, W. Hummer, B. Satzger, and S. Dustdar, “Application-level performance monitoring of cloud services based on the complex event processing paradigm,” in *2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 2012, pp. 1–8.

[220] M. Dayarathna and S. Perera, “Recent advancements in event processing,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 33, 2018.

[221] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, “An access control scheme for big data processing,” in *Proc. of IEEE CollaborateCom*, 2014, pp. 1–7.

[222] A. Kayes, W. Rahayu, T. Dillon, E. Chang, and J. Han, “Context-aware access control with imprecise context characterization for cloud-based data resources,” *Future Generation Computer Systems*, vol. 93, pp. 237–255, 2019.

[223] Y. Zhou, D. Feng, Y. Hua, W. Xia, M. Fu, F. Huang, and Y. Zhang, “A similarity-aware encrypted deduplication scheme with flexible access control in the cloud,” *Future Generation Computer Systems*, vol. 84, pp. 177–189, 2018.

[224] S. Fugkeaw and H. Sato, “Scalable and secure access control policy update for outsourced big data,” *Future Generation Computer Systems*, vol. 79, pp. 364–373, 2018.

[225] L. Qiu, F. Cai, and G. Xu, “Quantum digital signature for the access control of sensitive data in the big data era,” *Future Generation Computer Systems*, vol. 86, pp. 372–379, 2018.

[226] A. Noury and M. Amini, “An access and inference control model for time series databases,” *Future Generation Computer Systems*, vol. 92, pp. 93–108, 2019.

[227] J. Shafer, S. Rixner, and A. L. Cox, “The Hadoop distributed filesystem: Balancing portability and performance,” in *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*. IEEE, 2010.

[228] H. Li and R. Rao, “Accommodate Apache YARN to long-lived services,” in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1. IEEE, 2015, pp. 262–265.

[229] Apache Hadoop, “Apache Yarn long running services,” https://hadoop.apache.org/docs/r3.1.0/hadoop-yarn/hadoop-yarn-site/yarn-service/Overview.html, 2019, accessed: 2019-06-10.

[230] ——, “Apache Hadoop Groups Mapping,” https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/GroupsMapping.html, 2019, accessed: 2019-02-10.

[231] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “OpenStack: toward an open-source solution for cloud computing,” *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[232] Apache Log4j, “A Java-based logging utility,” https://logging.apache.org/log4j/2.x/, 2019, accessed: 2019-06-10.

[233] Apache Hadoop, “Apache Hadoop archive logs,” https://hadoop.apache.org/docs/r3.2.0/hadoop-archive-logs/HadoopArchiveLogs.html, 2019, accessed: 2019-06-01.

[234] S. Hoffman, *Apache Flume: Distributed log collection for Hadoop*. Packt Publishing Ltd, 2015.

[235] M. A. Khan and K. Salah, “IoT security: Review, blockchain solutions, and open challenges,” *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[236] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-ng: A scalable blockchain protocol,” in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 45–59.

[237] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, “Internet of vehicles: architecture, protocols, and security,” *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[238] E.-K. Lee, M. Gerla, G. Pau, U. Lee, and J.-H. Lim, “Internet of vehicles: From intelligent grid to autonomous cars and vehicular fogs,” *International Journal of Distributed Sensor Networks*, vol. 12, no. 9, p. 1550147716665500, 2016.

[239] M. Eltoweissy, S. Olariu, and M. Younis, “Towards autonomous vehicular clouds,” in *International Conference on Ad Hoc Networks*. Springer, 2010, pp. 1–16.

[240] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, “A survey on vehicular cloud computing,” *Journal of Network and Computer applications*, vol. 40, pp. 325–344, 2014.

[241] Apache Sentry, “Role engineering for data access,” https://sentry.apache.org/, 2019, accessed: 2019-02-10.

[242] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, “Blockchain based data integrity service framework for IoT data,” in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 468–475.

[243] A. Akhunzada and M. K. Khan, “Toward secure software defined vehicular networks: Taxonomy, requirements, and open issues,” *IEEE Communications Magazine*, vol. 55, no. 7, pp. 110–118, 2017.

[244] Apache Falcon, “Feed processing and feed management system,” https://falcon.apache.org/, 2015, accessed: 2019-02-10.

[245] Apache NiFi, “Big data workflow for IoT,” https://nifi.apache.org, accessed: 2019-02-10.

[246] X. Liu, Z. Shan, L. Zhang, W. Ye, and R. Yan, “An efficient message access quality model in vehicular communication networks,” *Signal Processing*, vol. 120, pp. 682–690, 2016.

[247] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[248] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, “Blockchain-based dynamic key management for heterogeneous intelligent transportation systems,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.

[249] Apache Eagle, “Data activity monitoring,” http://eagle.apache.org/docs/, 2015, accessed: 2019-02-10.

[250] Apache BigTop, “Packaging and interoperability testing,” http://bigtop.apache.org/, 2015, accessed: 2019-02-10.

[251] M. Tao, J. Zuo, Z. Liu, A. Castiglione, and F. Palmieri, “Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes,” *Future Generation Computer Systems*, vol. 78, pp. 1040–1051, 2018.

[252] V. Sucasas, G. Mantas, F. B. Saghezchi, A. Radwan, and J. Rodriguez, “An autonomous privacy-preserving authentication scheme for intelligent transportation systems,” *Computers & Security*, vol. 60, pp. 193–205, 2016.

[253] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, “Multitier fog computing with large-scale IoT data analytics for smart cities,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 677–686, 2018.

[254] M. Zeng, S. Leng, Y. Zhang, and J. He, “Qoe-aware power management in vehicle-to-grid networks: a matching-theoretic approach,” *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2468–2477, 2018.

[255] F. J. Clemente-Castello, B. Nicolae, R. Mayo, and J. C. Fernández, “Performance model of MapReduce iterative applications for hybrid cloud bursting,” *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[256] A. N. Toosi, R. O. Sinnott, and R. Buyya, “Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka,” *Future Gen. Comp. Sys.*, vol. 79, pp. 765–775, 2018.

[257] M. Cogorno, J. Rey, and S. Nesmachnow, “Fault tolerance in Hadoop MapReduce implementation,” 2013.

[258] H. Fu, H. Chen, Y. Zhu, and W. Yu, “FARMS: Efficient MapReduce speculation for failure recovery in short jobs,” *Parallel Computing*, vol. 61, pp. 68–82, 2017.

[259] Y. Wang, W. Lu, R. Lou, and B. Wei, “Improving MapReduce performance with partial speculative execution,” *Journal of Grid Computing*, vol. 13, no. 4, pp. 587–604, 2015.

List of Figures

List of Tables

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CAMPUS VIDA
CAMPUS DE EXCELENCIA INTERNACIONAL

One of the significant shifts of the next-generation computing technologies will certainly be in the development of Big Data (BD) deployment architectures. Apache Hadoop, the BD landmark, evolved as a widely deployed BD operating system. This dissertation addresses two leading issues involved in exploiting BD and large-scale data analytics using the Hadoop platform. (i) Scalability that directly affects the system performance using portable Docker containers. (ii) Security that spread the adoption of data protection among practitioners using federation access controls. This thesis has successfully studied the feasibility of applying state-of-the-art solutions to address previous issues in both cloud computing and on-premise models.