Estudio comparativo de los algoritmos ACO, ABC, FA para el balanceo de carga dinámica en un entorno Cloud Computing

William Alexander Duarte Vargas

2017

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial

Director
Dr. Jonatan Gómez Perdomo Ph.D.

Jurados
Prof. Dr. Oscar Agudelo
Prof. Ms Jesús Tovar

Octubre (2017)
31.10.2017

“Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad”. Albert Einstein

Estudio comparativo de los algoritmos ACO, ABC, FA *para el balanceo de carga dinámico en un entorno Cloud Computing*

William Alexander Duarte Vargas

Tesis de grado para obtener el título de:
Master en Telecomunicaciones

Director:
Jonatan Gómez Perdomo, Ph.D.

Linea de Investigación:
Vida Artificial

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
2017

Agradecimiento

Agradezco a mis padres, que han ofrecido un gran esfuerzo para culminar esta etapa de mi vida, por apoyarme en los momentos de felicidad y tristeza.

Gracias al Doctor Jonatan Gómez, por su paciencia, guía y conocimiento, que me permitieron fortalecer los conocimientos y la culminación de este estudio.

A las voces que alentaron la realización de este trabajo en los momentos de desasosiego y a los que participaron de forma directa e indirecta en su elaboración.

Resumen

La computación en la nube es el paradigma más popular de sistemas distribuidos hoy en día para ofrecer servicios y activos a través de internet. Su popularidad se debe a la promesa de contar con un sistema que puede ser escalado de forma dinámica, de bajo costo y sencillo de administrar, lo que finalmente es traducido en reducción de costos y mejor utilización de los recursos.

Un problema desde el surgimiento de los sistemas distribuidos es la asignación de la carga, debido a su naturaleza dinámica. El problema ha sido abordado desde diversas perspectivas en las cuales se encuentran técnicas evolutivas, teoría de juegos, modelos estocásticos, técnicas de vida artificial o una combinación de estas.

Este estudio compara las técnicas de luciérnagas, abejas y hormigas para evaluar su desempeño en un modelo en el cual se tienen en cuenta los recurso de las máquinas virtuales. Estos recursos son memoria, capacidad de procesamiento y ancho de banda disponible.

Con el propósito de ser objetivos en la evaluación se emplearon técnicas para la selección de los parámetros, con la finalidad de identificar los valores que ofrecieran el mejor desempeño.

Se empleo la herramienta CloudAnalyst para realizar simulaciones en un ambiente de computación la nube.

Palabras clave: computación en la nube, sistemas distribuidos, balance dinámico de carga, inteligencia de enjambre, ACO, ABC, FA.

Abstract

Cloud computing is the most popular of systems distributed today to offer services and assets through Internet paradigm. Its popularity is due to the promise of having a system that can be scaled dynamically, inexpensive and easy to management, which is ultimately translated in costs reduced and better utilization of resources.

One problem since the emergence of distributed systems is the allocation of the resources due to their dynamic nature. The problem has been approached from different perspectives in which evolutionary techniques, game theory, Stochastic models, artificial life techniques or a combination of these are.

This study compares the techniques of fireflies, bees and ants to evaluate how they perform in a model which takes into account the resource of virtual machines. These resources are memory, CPU and bandwidth available.

In order to be objective assessment techniques were used for the selection of the parameters, in order that offered the best performance.

CloudAnalyst has used the tool to perform simulations in a cloud computing environment.

keywords: cloud computing, distributed systems, dynamic load balancing, swarm intelligence, ACO, ABC, FA

Índice general

Índice de cuadros

Índice de figuras

Nomenclatura

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ACO	Ant Colony Optimization
APIs	Application Programming Interface
BLA	Bee Live Algorithm
CERN	Council Européen pour La Recherche Nucléaire
CLUMPs	cluster of miltiprocesors
CoPs	Cluster of Personal Computer
COWs	cluster of workstations
CPU	Central Procsess Unit
FA	FireFly Algorithm
FCFS	Firt Come First Server
IaaS	Infrastructure as a Service
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU	International Telecommunication Union
LAN	Local Area Network
MANET	Mobile ad hoc network
MASS	Multi-agent Spatial Simulation
MIT	Massachusetts Institute of Technology

MMP	Massively Parallel Procesing
MOACO	Multi-Objetive Ant Colony Optimization
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OGF	Open Grid Forum
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
RSM	Response Surface Metodology
SaaS	Software as a Service
SDK	software development kit
SMP	Symetric Multi Procesor
SOAP	Simple Object Access Protocol
SPI	Software, Platform, Infrastructure
SSI	Single System Image
WAN	Wide Area Network
WSDL	Web Services Description Language
XML	eXtensible Markup Language

1. Introducción

El enfoque de sistemas distribuidos con mayor aceptación y popularidad en los últimos años, tanto en ambientes corporativos como académicos, es la computación en la nube (*cloud computing*). Esta popularidad se atribuye a la promesa de contar con un sistema de bajo costo, sencillo de administrar y que puede ser escalado de forma dinámica.

Como consecuencia de su popularidad, vendedores, directores de tecnología y proveedores de servicios, pertenecientes al medio de tecnologías de la información, tienden a distorsionar o confundir la computación en la nube con nociones como la de agrupación de servidores (*cluster*) o de malla computacional (*grid computing*), al asumir que dichos conceptos se refieren a lo mismo.

Los tres conceptos pueden ser identificados como una evolución uno del otro, en donde se identifican tres momentos: se inicia de un conjunto de máquinas físicas que comparten recursos a través de una infraestructura de red, en la que el cliente percibe un único elemento, luego se sigue una distribución geográfica en la que los procesos son divididos en mensajes atómicos que pueden ser procesados en paralelo por un controlador, y finalmente en la utilización de componentes heterogéneos físicos o virtuales con varios controladores para la localización de servicios.

La investigación en computación en la nube trata de aportar soluciones que permitan sobrellevar el aumento en la complejidad de cómputo, las restricciones de tecnología, el alto costo en la administración, con la finalidad de ofrecer una mejor calidad en la prestación de productos y servicios para soportar una o varias unidades de negocio.

Dentro de los enfoques abordados en investigación en el campo de la computación en la nube se encuentran: el aprovisionamiento automático de recursos, la virtualización, la administración energética, y la seguridad, entre otros.

- En el enfoque orientado al aprovisionamiento automático de recursos el objetivo es identificar, asignar y reasignar recursos de forma dinámica y automática con la finalidad de cumplir los niveles de servicios acordados (*SLA, service level agreement* por sus siglas en inglés). Urgaonkar y otros [Urgaonkar et al., 2005] proponen una técnica para manejar las variaciones de carga en internet a través de un modelo de aprovisionamiento en el uso de aplicaciones multicapa que relaciona un modelamiento de colas para localizar los recursos, además, consideran una combinación de métodos de predicción y de reacción para determina el momento

cuando es necesario aprovisionar los recursos. Constantino [Vázquez, 2012] propone una arquitectura de referencia en el aprovisionamiento de recursos aplicable tanto en computación en malla como en computación en la nube.

- En temas de virtualización se han realizado, por un lado, desarrollos orientados a localización de recursos y, por el otro técnicas de migración que permiten distribuir la carga y mejorar la rapidez de repuesta que ofrecen los centros de datos. Se destacan los adelantos realizados por Citryx-Xen y VmWare que recientemente han introducido e implementado la migración sin necesidad de apagar o inhabilitar las máquinas virtuales [Zhang et al., 2010]. También es relevante la iniciativa sobre contenedores, que corresponde a una tecnología de virtualización a nivel de sistema operativo. En Linux se emplean características de aislamiento de recursos del kernel, lo que permite que contenedores independientes se ejecuten dentro de una sola instancia y que se evite la sobrecarga de iniciar y mantener máquinas virtuales.
- Respecto a la administración energética y sostenible se encuentran trabajos orientados a la reducción de emisiones de carbono y a la utilización eficiente de las máquinas empleadas en el centro de datos, para optimizar la capacidad de enfriamiento y el consumo energético. Patel [Patel et al., 2016] realiza una introducción sobre el modelo de "*green cloud computing*" al seleccionar la ubicación de centros de datos que presten el servicio ubicados geográficamente en lugares donde el costo de operación sea menor. Jing [Jing et al., 2013] define un marco de referencia arquitectónico y de principios para alcanzar una eficiencia energética y Liu [Liu et al., 2016] propone un algoritmo para asignación de tareas en paralelo con el propósito de optimizar el consumo energético.
- El campo con mayor atención es el de la seguridad, debido a que en ocasiones el proveedor de servicios no tiene acceso a la seguridad física del centro de datos y es así como la búsqueda de mecanismos que aseguren la confidencialidad adquiere relevancia, tanto en la transmisión como en la recepción de la información para que se puedan articular procesos de auditoría donde se verifique la calidad de los controles aplicados [Zhang et al., 2010]. Hussein [Hussein and Khalid, 2016] realiza una revisión sobre cada una de las capas de la computación en la nube e identifica los riesgos de seguridad y emplea la autenticación y la encripción en los canales de transmisión de datos para mitigar dichos riesgos.
- Otro campo con gran crecimiento es el relacionado con el almacenamiento. Este fue impulsado principalmente para apoyar el paradigma de Big Data empleando MapReduce respecto a la distribución de tareas con un alto consumo de procesamiento. Basha[Basha et al., 2016] plantea el problema de la recuperación de la información almacenada empleando como marco de trabajo Hadoop y MapReduce, a través de un sistema de archivos distribuido. También se encuentran trabajos donde se desarrollan mecanismos para la encriptación de datos.

Este trabajo tiene el propósito de analizar el campo del balanceo de carga, que aunque no es un tema nuevo, ha sido estudiado y sobre el que se han propuesto enfoques que emplean teoría de juegos, computación evolutiva, teoría de colas, procesos ocultos de Markov, entre otros. El campo del balanceo de carga tiene una naturaleza combinatoria enmarcada en los problemas no polinomiales completos (NP-Completo) y aborda tanto a las agrupaciones de servidores, las mallas computacionales y la computación en la nube, por lo que se explora para abordar el problema un enfoque desarrollado desde el campo de inteligencia de enjambres en el cual se aplican las técnicas de colonia de hormigas (Ant Colony Optimization por sus siglas en inglés), Algoritmo de Luciérnagas (FireFly Algorithm por sus siglas en inglés) y Colonia de Abejas Artificiales (Artificial Bee Colony por sus siglas en inglés), con este propósito se realiza una comparación de las técnicas, y se establece a través de de escenarios la simulación cual de las técnicas brinda una mejor solución, identificando así sus fortalezas y sus debilidades.

1.1. OBJETIVOS

Objetivo general

Realizar un estudio comparativo que permita contrastar los técnicas de Colonia de Hormigas (ACO), Colonia de Abejas Artificiales (ABC) y Luciérnagas (FA) para abordar el problema de balanceo de carga dinámico en un entorno de computación en la nube.

Objetivos específicos

- Simular la implementación de los algoritmos ACO, ABC y FA en un entorno de computación en la nube.
- Evaluar el desempeño de los algoritmos propuestos en un entorno de computación en la nube.
- Identificar ventajas y desventajas de cada uno de los algoritmos seleccionados.
- Medir las soluciones obtenidas del problema del balanceo de carga por cada uno de los algoritmos seleccionados.

1.2. Estructura del documento

El trabajo se encuentra organizado de la siguiente forma, el capitulo 2 ofrece una revisión general de sistemas distribuidos, agrupamiento de servidores y mallas computacionales con el propósito de ofrecer una visión general de los conceptos. El capitulo 3 se realiza una revisión de los principales conceptos de computación en la nube. El

capitulo 4 hace una exploración de los conceptos relacionados, se expone el problema de balanceo y las técnicas, el capitulo 5 muestra la implementación, evaluación, comparación y resultados. Finalmente. El capitulo 6 se presentan las conclusiones del presente estudio y se proponen trabajos futuros.

2. Preliminares

En este capítulo se presentan conceptos sobre sistemas distribuidos y una revisión general acerca de agrupaciones de servidores y mallas computacionales, elementos fundamentales en la evolución y conformación de la computación en la nube.

2.1. Sistemas distribuidos

Los sistemas distribuidos han sido estudiados desde mediados de los años sesenta, y producto de los avances, mejoras e innovaciones, en los cuales se ha ampliado y actualizado sus fronteras, razón por la que no existe una única definición.

G. Coulouris, J. Dollimore y Tim Kinberg [G. Coulouris, 2001] definen un sistema distribuido como: "aquel en el que sus componentes localizados en computadores conectados en red se comunican y coordinan sus acciones únicamente mediante el paso de mensajes". Para Tanembaun [G. Tanenmbaum, 2001] corresponde a: "*una colección de computadores independientes que aparecen ante los usuarios como un único computador*".

En la actualidad, la implementación, el diseño y la construcción de sistemas distribuidos no se limita a los grandes centros de datos, dado que pueden ser implementados en cualquier dispositivo que permita compartir sus recursos. Un ejemplo es el trabajo de Zambrano [Vizuete, 2012] Arquitectura e Implementación de un Sistema Distribuido de Detección de Sismos para Alerta Temprana, en el cual realiza la implementación de un sistema en tiempo real de alertas sísmicas empleando teléfonos inteligentes. La forma de representar un sistema distribuido de forma abstracta se presenta en la figura 2.1.1.

En términos generales, un sistema distribuido, es un conjunto de elementos de computo que tiene como objetivo principal compartir recursos para aumentar la capacidad individual obtenida y optimizar su utilización.

2.1.1. Características

- Los elementos de un sistema distribuido son denominados nodos. Estos elementos pueden emplear componentes de diferentes capacidades en hardware, lo cual

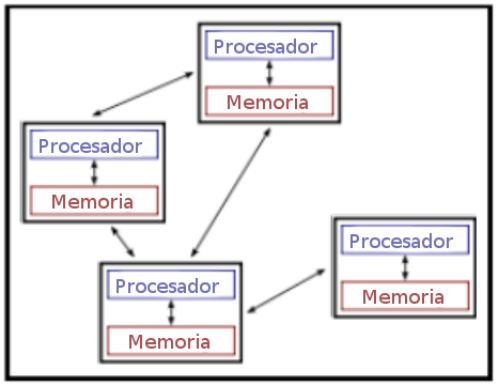


Figura 2.1.1.: Vista Abstracta de Un Sistema Distribuido, Tomada de [Joshi, 2012] capítulo 1 página 3

produce un ambiente heterogéneo en el que se espera, que dichos nodos puedan compartir los recursos disponibles y emplearlos en la realización de las tareas que le han sido asignado. Esta característica permite la utilización de una infraestructura estándar, que opera a un costo inferior al incurrido si se utilizan piezas sofisticadas de hardware.

- Sin importar el número de nodos que pertenecen al sistema distribuido los usuarios finales perciben que solo hay un único elemento realizando la tarea encomendada; está característica es conocida como SSI (*Single System Image* por sus siglas en inglés).
- la capacidad de poder crecer de forma dinámica de acuerdo a las necesidades en un momento determinado. Esta característica puede realizarse de dos formas: una denominada crecimiento vertical donde el aumento de la capacidad de computo o procesamiento se realiza a través de la actualización, la adición o el cambio del hardware por uno de mejores prestaciones, y la otra manera es denominada crecimiento horizontal (también conocida como escalamiento) e implica que se puede adicionar uno o más nodos con características similares o mejores al conjunto existente.
- La tolerancia a fallos como resultado de la noción de grupo o conjunto en los sistemas distribuidos. Si un elemento falla, las tareas asignadas a él pueden ser realizadas por los otros participantes y así evitar que el sistema quede inoperante; es importante reconocer que esto no quiere decir que de esta manera se puedan evitan todas las fallas que pueden dejar inoperante el sistema completo.
- La flexibilidad en los componentes empleados es producto de la utilización de estándares para realizar la comunicación y comprende el principio de que la interacción es únicamente a través del paso de mensajes.

2.1.2. Ventajas

La utilización de componentes heterogéneos conlleva una serie de ventajas, dentro de las cuales se encuentran:

- Aumento en la capacidad de cómputo con un menor costo y aumento de la sinergia producto de la interacción de cada uno de sus componentes.
- El tiempo de respuesta hacia los usuarios o clientes tiende a ser menor al que se obtiene si solo se emplea un único elemento, debido a la ejecución concurrente de procesos o trabajos (los trabajos son distribuidos y ejecutados de forma paralela en varios nodos).
- Tanto la implementación como la creación de nuevas técnicas, en el uso efectivo de los recursos compartidos, son traducidos en eficiencia y flexibilidad.
- La utilización de técnicas de procesamiento distribuido produce un sistema menos propenso a fallas, ya que el falló de uno o varios componentes no afecta a la disponibilidad de todo el servicio. Esto se puede traducir en satisfacción del cliente.
- Inclusión rápida de nuevos recursos, sin afectar los actuales (crecimiento vertical).
- La capacidad de tener más de un elemento permite tener una copia fiel de la información, con lo que se mitiga la perdida de información.

2.1.3. Desventajas

- El principal problema es el software, debido principalmente a su complejidad en el diseño, implantación y uso.
- Requiere un mayor número de controles en el procesamiento, acceso y persistencia.
- La velocidad de propagación de la información puede ser muy lenta. La rapidez de la respuesta no tiene que ver con la carga del sistema o con su velocidad si no con la latencia de la infraestructura de comunicación.
- La operación ya sea de replicación de datos o de servicios presenta posibilidades de fallas e inconsistencias.
- La administración es más compleja y necesita personal calificado.
- Sensible a la pérdidas de mensajes a nivel de redes de comunicación, debido a la saturación en el tráfico, expansiones, etc.

En general, el resultado obtenido es superior en términos de los beneficios, debido a que algunas de las desventajas pueden ser mitigadas dependiendo de la disciplina con la que sea administrada, documentada y diseñada la implementación.

2.2. Agrupación de servidores

También conocida como *clusters*, corresponde a un sistema distribuido diseñado para alcanzar un objetivo específico para un usuario en particular o una tarea, como por ejemplo, el aumento de la capacidad y la disponibilidad en una base de datos, servicios web, que deben: encontrarse bajo un mismo dominio o ubicación lógica y no necesariamente compartir la misma ubicación física pero si pertenecer a la misma red de comunicaciones de área local (*LAN* por sus siglas en inglés *Local Area Network).*

La agrupación de servidores surge como la respuesta a la necesidad de resolver tareas complejas en equipos de computo estándar donde se puede emplear hardware más económico y fácil de adquirir, que el de los denominados super-computadores. Esta agrupación puede ser vista como un sistema de procesamiento masivamente paralelo (MMP *Massively Parallel Procesing* por sus siglas en inglés) pero más económico dado que el costo de componentes especializados es elevado comparado con el costo de los equipos de uso frecuente [Mesa, 2009].

Recientemente, la agrupación de servidores ha obtenido gran popularidad en su implementación, debido al aumento en la capacidad de procesamiento, comunicaciones y servicios ofrecidos a través internet, así como la necesidad de cumplir los acuerdos de servicios y la satisfacción del cliente. Las agrupaciones de servidores son clasificadas de la siguiente forma:

- ***Alto desempeño***: conjunto de nodos cuyo objetivo es la realización de tareas que requieren de una alta capacidad de procesamiento, memoria o de ambas memoria y procesamiento. Su principal objetivo es el número de tareas que puede finalizar correctamente utilizando el menor tiempo posible. Este tipo de agrupación está enfocado en: "resolver cálculos matemáticos, renderizaciones de gráficos, compilación de programas, compresión de datos, descifrado de códigos" [Mesa, 2009]
- ***Alta disponibilidad:*** su objetivo es ofrecer confiabilidad en la disponibilidad de uno o varios recursos presentados al usuario. Su finalidad es aumentar la tolerancia a fallar y la auto-recuperación en caso de presentarse alguna falla. Este tipo de agrupación se enfoca a la necesidad de ofrecer un servicio continuo, un sistema redundante y tolerante a fallos.
- ***Alta eficiencia (throughput):*** su principal objetivo es realizar el mayor número de tareas en el menor tiempo posible.

Otra forma de clasificación se despliega a partir de las características propias de los nodos. Las agrupaciones son clasificadas como dedicadas o no dedicadas.

- Según la naturaleza de los elementos utilizados en su implementación, donde se encuentran: agrupaciones de computadores personales (*CoPs cluster of PCs* por sus siglas en inglés), agrupaciones de estaciones de trabajo (*COWs cluster of*

workstations por sus siglas en inglés) agrupaciones de procesadores simétricos (SMP *Symetric Miltiprocesors* por sus siglas en inglés) (*CLUMPs cluster of miltiprocesors* por sus siglas en inglés)[Buyya, 1999].

- Según la disposición dentro de la organización, donde se encuentran: agrupamientos departamentales; agrupamientos interorganizaciones, meta-computadores nacionales y metacomputadores internacionales. Para [Buyya, 1999] el conjunto de varios agrupamientos pueden ser ínter-conectados para construir un gran sistema de agrupamientos, internet puede ser utilizado como un ejemplo.

2.2.1. Arquitectura

Partiendo de la definición de agrupamiento de servidores, cada uno de sus nodos pueden participar con: uno o varios procesadores, su propia memoria, en operaciones de entrada y salida y un sistema operativo. La figura 2.2.1 ilustra la arquitectura de una agrupación de servidores.

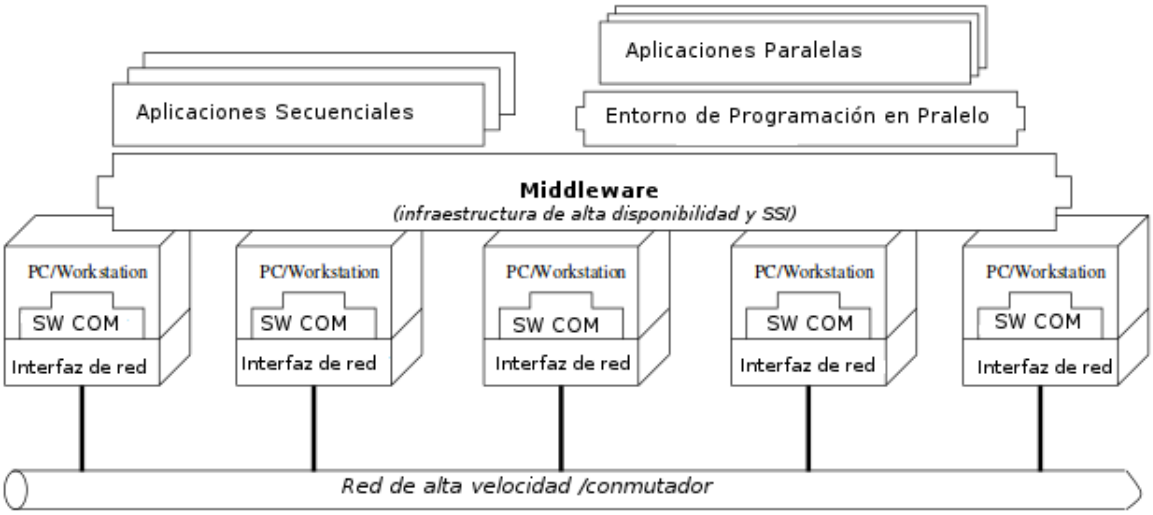


Figura 2.2.1.: Arquitectura general en un agrupamiento de servidores tomado de [Buyya, 1999], capítulo 1 página 10

Los componentes principales de un agrupamiento de servidores son:

- Dos o más nodos que son equipos de cómputo en la mayoría de los casos poseen multiprocesadores simétricos (SMP). Estos equipos tienen su propio sistema operativo y se encuentran conectados a una red, por lo general, de alto rendimiento. La comunicación se realiza a través de protocolos y servicios estándar,

que pueden ser definidos para obtener la finalidad con la que fue diseñada e implementada la agrupación de servidores.

- Una capa intermedia denominada *Middleware*, cuya finalidad es que el usuario perciba un único sistema (*SSI Single System Image* por sus siglas en inglés) y pueda administrar la disponibilidad de la infraestructura del sistema. Esta capa puede ser implementada en hardware especializado o a través de software.
- Una capa de aplicaciones clasificadas como secuenciales y paralelas; estas últimas están soportadas en un conjunto de herramientas como compiladores, máquinas virtuales e interfaces para el intercambio de mensajes.

2.3. Malla computacional

Al igual que una agrupación de servidores, una malla computacional (*grid computing* por sus siglas en inglés), es un sistema distribuido que tiene su origen en 1998, cuando Lan Foster y Carl Kesselman la presentaron como una analogía de la red eléctrica: *"nos podemos conectar a la malla computacional para obtener potencia de cálculo sin preocuparnos de dónde viene al igual que hacemos cuando conectamos un aparato eléctrico"* [Foster and Kesselman, 2003].

A diferencia de las agrupaciones de servidores, los nodos pertenecientes a una malla computacional no se encuentran asociados bajo una única organización o dominio. Su origen es diverso y cada nodo participa con la intensión de compartir recursos sin importar el propietario. El interés en su implementación radica en el uso eficiente de recursos, para ofrecer una calidad en la respuesta óptima, rápida y eficiente.

Las mallas computacionales han tenido gran acogida en los ambientes científicos y académicos. Por ejemplo, la Administración Nacional de la Aeronáutica y del Espacio (*NASA* por sus siglas en inglés), así como el Consejo Europeo para la investigación Nuclear *(CERN Council Européen pour La Recherche Nucléaire* por sus siglas en francés) han participado en proyectos para la implementación de mallas computacionales, principalmente en la ejecución de cálculos complejos o grandes volúmenes de datos. Los entornos corporativos también han desarrollado propuestas en este campo, como las multinacionales Oracle® e IBM® que ofrecen productos que facilitan la implementación de servicios sobre mallas computacionales.

Una ventaja al implementar una malla computacional es la reducción del trabajo empleado en la definición de la capacidad a ser instalada y aprovisionada para soportar los servicios, dado que ya no se encuentra en función de los eventos de sobrecarga, conocidos como picos de tráfico o de usuarios, debido a que el controlador de la malla puede identificar los nodos a utilizar que se encuentren disponibles. A diferencia de las agrupaciones de servidores, que utilizan una red privada o *LAN* para su comuni-

cación, las mallas computacionales pueden utilizar una red privada o internet para la comunicación entre sus nodos.

Su principal inconveniente se encuentra en la administración de los nodos como consecuencia de su alta heterogeneidad, la sincronización de los procesos, el monitoreo de recursos, la asignación de trabajos y la seguridad al emplear canales federados de comunicación.

Las mallas pueden ser clasificadas por sus características, en el trabajo de Maqueira y Bruque [Maqueira and Bruqué, 2011] se propone una clasificación basada en su finalidad y las relaciona de la siguiente forma:

- ***Malla computacional:*** según la potencia resultante de la agregación de un conjunto de sistemas distribuidos que aprovechan la capacidad de proceso no utilizada por los nodos pertenecientes en un momento determinado. Sus recursos son utilizados al momento de quedar ociosos. Esta categoría se subdivide en:
  - ***Mallas domesticas:*** Son mallas que utilizan equipos domésticos o de escritorio.
  - ***Mallas de servidores:*** Son conformados por servidores.
  - ***Mallas de grupos de servidores de alto rendimiento:*** son construidas por un grupos de servidores de alto rendimiento.
- ***Mallas de datos:*** corresponde a una estructura orientada a grandes volúmenes de datos que identifica requerimientos y componentes dentro de cada uno de los nodos.
- ***Mallas colaborativas:*** en ellas participan diferentes agentes dispersos geográficamente que colaboran en un determinado trabajo.
- ***Mallas de servicios:*** supone una plataforma de servicios enfocada a aquellos dispositivos ligeros y de bajo cómputo. En esta clasificación los usuarios pueden ofrecer también servicios de cómputo, teniendo la dualidad de proveedores o clientes. A esta categoría se atribuye la concepción de computación en la nube.

Otra clasificación propuesta también por Maqueira y Bruque [Maqueira and Bruqué, 2011] es a partir del ámbito organizativo de la siguiente forma e ilustrada como en la figura 2.3.1.

- ***Inframalla:*** está construida con un número limitado de nodos altamente homogéneos. Esta clasificación es muy similar a implementar una agrupación de servidores.
- ***Intramalla:*** corresponde a configuraciones donde la ubicación de sus nodos no es la misma geográficamente, pero pertenecen al mismo dominio o empresa.

- ***Extramalla:*** en este modelo los nodos pueden pertenecer a diferentes compañías u organizaciones, donde los recursos son compartidos a través de redes virtuales privadas. Su principal objetivo es colaborar en la solución de proyectos comunes optimizado los recursos existentes.
- ***Intermalla:*** son construidas por diversas organizaciones que tienen por finalidad compartir sus recursos sin existir vínculo alguno entre ellas.

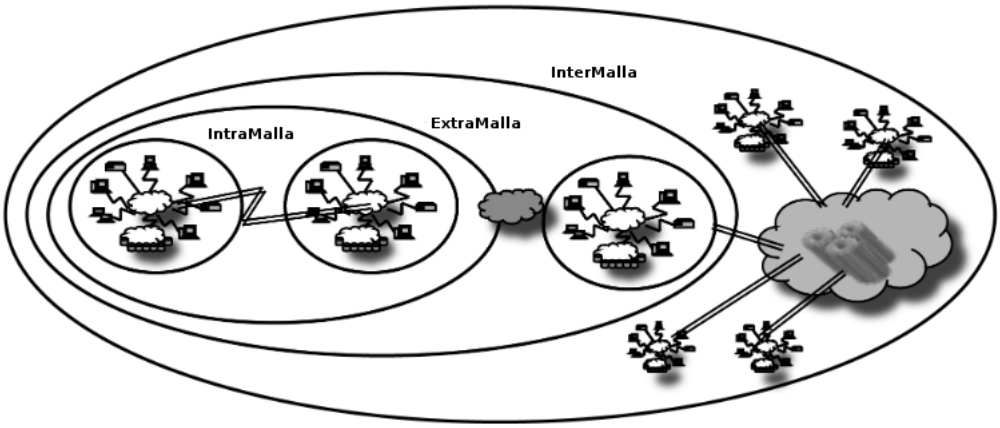


Figura 2.3.1.: Clasificación Mallas tomada de [Maqueira and Bruqué, 2011] página 158.

2.3.1. Arquitectura

La arquitectura para la implementación de una malla computacional está expresada en capas, donde cada una cumple una función determinada, similar al modelo ISO/OSI en la redes de computadores. Las aplicaciones, conjuntos de herramientas, APIs, SDK, etc, deben cumplir una estructura general como la presentada por Ian Foster, Carl Kesselman y Steve Tuecke en su artículo *"Anatomy of the Grid: Enabling Scalable Organizations"* [Berman et al.,]. Esta arquitectura se articula en cinco niveles: la infraestructura, la conectividad, la gestión del recurso, la gestión de recursos o collectiva (collective), y el nivel de aplicación.

Otro enfoque es el propuesto por Millán [Tejedor, 2007] que define que la arquitectura se encuentra distribuida en capas donde entre más alta esté más cerca se encuentra del usuario. Las capas propuestas son las siguientes:

- ***Capa de aplicación:*** está compuesta por las aplicaciones que son expuestas o que son utilizadas por los usuarios.

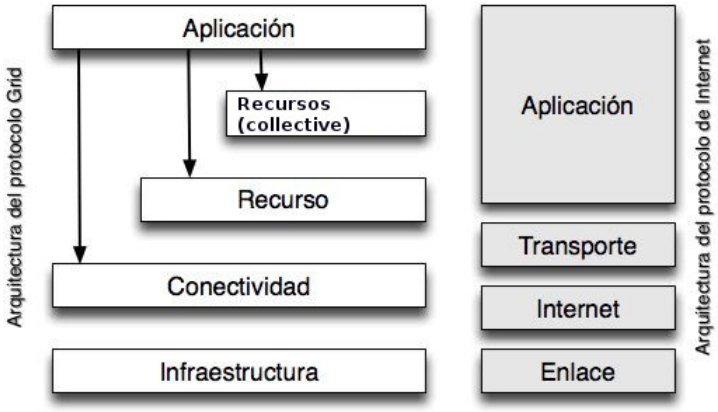


Figura 2.3.2.: Arquitectura Malla Computacional Tomada de [Berman et al.,] página 178.

- ***Capa intermedia o "middleware":*** es la capa que proporciona las herramientas que permite la participación de los diferentes recursos de forma coordinada y segura.
- ***Capa de recursos:*** está compuesta por los recursos compartidos que conforman la malla.
- ***Capa de red:*** es la capa compuesta por las conexiones entre cada uno de los recursos compartidos.

2.3.2. Estandarización

La implementación de un sistema distribuido altamente heterogéneo, como es una malla computacional y cuya comunicación es realizada únicamente a través de mensajes, es solo posible a través de estándares que facilitan la ínter-operabilidad y la portabilidad de sus componentes. De esta manera, la estandarización permite y facilita el proveer servicios.

El adoptar mecanismos y protocolos estándar no asegura que se pueda realizar la integración requerida sin un organismo que asuma las tareas de: discutirlos, probarlos, reglamentarlos, difundirlos y promoverlos. En el caso de las mallas computacionales, esta tarea se encuentra a cargo del *OGF* (*Open Grid Forum* por sus siglas en inglés) [*www.ogf.org*]. A dicha organización pertenecen más de 400 miembros de 50 países y está constituido a través de comunidades encargadas de plantear, discutir y difundir los estándares que van a ser utilizados. Dentro de las tecnologías empleadas para la estandarización se encuentran: XML, SOAP, WSDL, entre otras.

3. Computación en la nube

Aunque el paradigma de computación en la nube en la actualidad se encuentra de moda, el concepto no es nuevo. Para Mariescu [Marinescu, 2013] la idea original fue propuesta en 1961 cuando Jhon Macarthy afirmó en un discurso para celebrar el centenario del MIT: *"la tecnología de tiempo compartido de las computadoras podría conducir a un futuro en el que el poder del cómputo e incluso aplicaciones específicas podrían ser vendidas como un servicio (como el agua o la electricidad)"*.

Los enfoques de tiempo compartido de cómputo estuvieron muy activos hasta finales de los setentas cuando por las restricciones de la época tanto en software, hardware y telecomunicaciones fueron aplazados.

Sultan [Sultan, 2010] indica: *"el término nube fue probablemente inspirado por los textos de tecnologías de la información en donde bajo la ilustración de una nube se representaban entornos remotos, por ejemplo, Internet, con el fin de ocultar la complejidad detrás de ellos. Sin embargo, mediante la comprensión del tipo de servicios que ofrece la computación en nube, uno empieza a entender de lo que este enfoque se trata"*.

3.1. Definición

Una de las constantes en tecnologías de la información y de las comunicaciones es el diverso número de definiciones sobre un mismo concepto o paradigma, la computación en la nube no es ajena a este fenómeno. Armbrust [Armbrust et al., 2010] define la computación en la nube como la referencia tanto a las aplicaciones entregadas como servicios sobre internet como al hardware y sistemas en los centros de datos que proveen dichos servicios.

La norma técnica ISO/IEC 17788:2014[17788, 2014], define la computación en la nube como un paradigma que habilita en red el acceso escalable y elástico de un conjunto de recursos físicos o virtuales bajo demanda para el aprovisionamiento y administración.

Para Wang [Wang et al., 2008] *"la computación en la nube es un conjunto de servicios disponibles, escalables, provistos, con calidad de servicio garantizada, normalizados y personalizados, en plataformas de computación económicas ofrecidas bajo demanda de forma sencilla y generalizada"*. Mientras que para Buyya[Buyya et al., 2009]

"Una nube es un tipo de sistema distribuido que consiste en una colección de recursos ínter-conectados y virtualizados que son dinámicamente aprovisionados como un único recurso de computo basado en acuerdos de servicio producto de la negociación entre los proveedores de servicio y los consumidores".

En 2009 la firma de consultoría McKinsey encontró alrededor de 22 definiciones diferentes. Dado que la mayor definición citada, y por considerarla la más completa, se adopta para este trabajo la definición aportada por el instituto de estándares y tecnología de los estados unidos NIST (*National Institute of Standards and Technology* por sus siglas en inglés), en la cual la computación en la nube es definida como: "un modelo para habilitar ubicua, conveniente, bajo demanda en red un conjunto configurable de recursos (red, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un mínimo de esfuerzo de administración o interacción del proveedor de servicios. Este modelo está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue" [Mell and Grance, 2011]. La definición es ilustrada a en la figura 3.1.1.

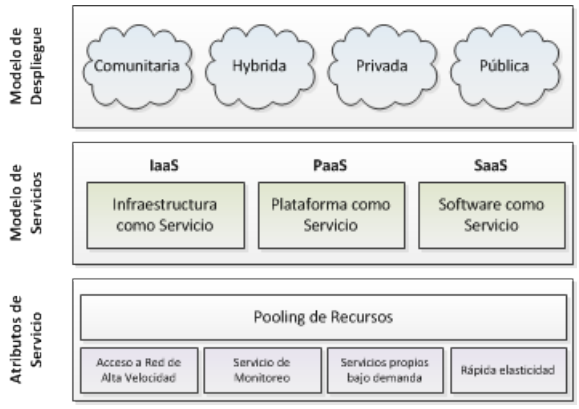


Figura 3.1.1.: Definición de computación en la nube tomado de [Sosinsky, 2011] capítulo 1 página 26.

3.2. Características

Dentro de las principales características, definidas por la Unión internacional de Telecomunicaciones (International Telecommunication UnionITU por sus siglas en inglés) en [Y.3500, 2014], se encuentran:

- ***Acceso amplio de red***: los recursos tanto físicos como virtuales están disponibles sobre una red accedida a través de mecanismos estándar que promueven plataformas heterogéneas. Esta característica se centra en la facilidad de acceso para que los clientes aceden a los recursos desde donde necesiten trabajar.

- ***Medición del servicio***: una característica es la de poder medir los servicios utilizados con el propósito de monitorear, controlar, reportar y facturar los recursos empleados. La finalidad principal de esta característica es que el cliente únicamente pague por los recursos que emplea, es decir le permite pasar de una baja eficiencia en la utilización de activos en el modelo de negocios a uno de alta eficiencia.
- ***Multi-huspedes (Multitenant)***: esta característica permite ofrecer servicios a varios clientes de forma simultanea, aislándolos y haciéndolos inaccesibles con los otros clientes, ya sean en recursos físicos o virtuales (así sea el mismo recurso expuesto a varios clientes estos no pueden acceder a los recursos presentados a otros clientes).
- ***Autoservicio bajo demanda***: permite a un consumidor obtener capacidades de computo, según sus necesidades de forma automática o con un mínimo de interacción del proveedor. El enfoque principal es ofrecer una relativa reducción de costos, tiempo y esfuerzo para realizar cualquier acción, ya que le permite al cliente hacer lo que necesite cuando lo necesite, sin necesidad de interacción humana adicional o sobrecarga del recurso humano.
- ***Rápida elasticidad y escalabilidad***: esta característica permite ajustar los recursos físicos o virtuales de forma rápida y elástica. En algunos casos de forma automática para que rápidamente los recursos sean ampliados o reducidos. El cliente percibe los recursos de forma ilimitada, obteniendo la capacidad de adquirir recursos en cualquier cantidad y momento, dependiendo de los acuerdos de servicio. La finalidad principal de esta característica es que el cliente no se preocupe por planear la capacidad.
- ***Conjunto de recursos***: esta característica permite la adición de recursos físicos y virtuales dependiendo de las necesidades expuestas por los consumidores. Su finalidad principal es soportar la característica de multi-huesped mientras que al mismo tiempo abstrae y enmascara la complejidad subyacente. Desde la perspectiva del consumidor, este es consciente de que tiene servicios expuestos pero desconoce dónde están soportados o su ubicación física, por lo que la carga operativa del mantenimiento y la distribución es responsabilidad del proveedor. Debe señalarse que el cliente puede definir la ubicación donde desea que sus servicios se encuentren.

3.2.1. Modelos de servicio

Han sido aceptados tres modelos de servicio que son identificados como SPI (software, plataforma e infraestructura). Es muy común, sobre todo en vendedores de servicios de tecnología, hablar de XaaS donde la X es remplazada por Database, Storage, Network u otro término para expresar que el recurso ofrecido puede ser accedido bajo demanda,

a pesar de que pueden ser clasificados en los tres modelos aceptados. La definición de los modelos es tomada de [Marinescu, 2013]:

- **SaaS:** *Software as a Service* o software como servicio es la capacidad ofrecida por el proveedor para el uso de aplicaciones en una infraestructura. Las aplicaciones son accedidas desde varios dispositivos a través de clientes ligeros como, por ejemplo, un navegador web. El usuario no administra o controla la infraestructura con la que están soportadas las aplicaciones, como: la red, los servidores, el sistema operativo, y el almacenamiento. Es así como tiene permitido únicamente limitar el número de usuarios que acceden a la aplicación. Se tiene como ejemplos: Google Docs, el CRM de salesforce.com, servicios geográficos etc. A continuación se ilustra la estructura de un modelo SaaS.

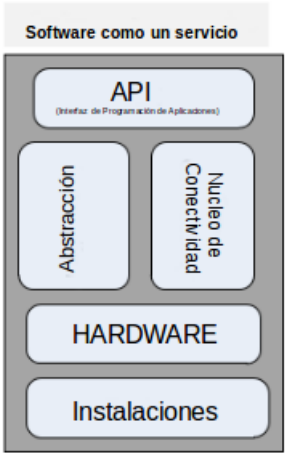


Figura 3.2.1.: Estructura de un modelo SaaS tomado de [Marinescu, 2013] capítulo 1 pagina 12

- **PaaS**: *Plataform as a Service* o plataforma como un servicio es la capacidad ofrecida a un usuario para desplegar aplicaciones adquiridas o implementadas, utilizando lenguajes de programación, sistemas gestores de bases de datos y herramientas soportadas por el proveedor. El usuario solo tiene el control sobre las aplicaciones desplegadas y puede manipular cierta configuración de entorno, como: la administración de sesiones, la integración de dispositivos y la administración de contenidos. Este modelo no es útil cuando la aplicación debe ser potable pues se utilizan lenguajes de programación específicos o propietarios no ofrecidos por el proveedor o cuando se debe especificar hardware o software. Este tipo de modelo se encuentra en: Windows Azure, Google Apps Engine, Amazon EB. A continuación se presenta la estructura de un modelo PaaS:

- **IaaS** *Infraestructure as a Service* o Infraestructura como servicio, es la capacidad

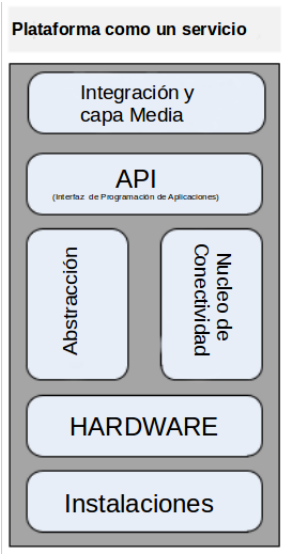


Figura 3.2.2.: Estructura de un modelo PaaS tomado de [Marinescu, 2013] capítulo 1 pagina 12

ofrecida a un usuario donde se provee almacenamiento, red y otros componentes fundamentales. En este modelo el usuario despliega y ejecuta de forma arbitraria software, incluyendo sistemas operativos y aplicaciones que se requieran sin ninguna limitación. El consumidor puede definir el sistema operativo, capacidad de almacenamiento, despliegue de aplicaciones y un control limitado sobre dispositivos de red como cortafuegos. Este modelo es el preferido cuando se desean adquirir servicios de infraestructura. Algunos ejemplos son: RackSpace, Amazon EC2, entre otros. A continuación se expone la estructura de IaaS:

En la figura 3.2.4 se presenta un resumen de lo expuesto en este apartado, identificando los modelos de despliegue, su control y algunos ejemplos.

El control entre proveedor y consumidor de servicios se ilustra en la figura 3.2.5, en el cual dependiendo del modelo seleccionado para suplir las necesidades se adquiere control sobre cada una de las capas de la solución.

3.2.2. Modelos de despliegue

Estos corresponden a una clasificación en la que se define la manera como la nube computacional puede ser organizada y está basada en el control y la manera como se comparten los recursos físicos o virtuales [Y.3500, 2014]. En esta clasificación han sido aceptadas las siguientes categorías:

Figura 3.2.3.: Estructura de un modelo SaaS tomado de [Marinescu, 2013] capítulo 1 pagina 12.

- ***Nubes públicas:*** los servicios se encuentran disponibles para cualquier consumidor, el control de los servicios es responsabilidad solo del proveedor. Una nube pública puede pertenecer, ser administrada y ser operada bajo un modelo de: negocio, academia, entidad gubernamental o una combinación de estas. El consumidor no debe tener ninguna o mínimas restricciones.

- ***Nubes privadas:*** los servicios se encuentran disponibles única y exclusivamente a un consumidor. Una nube privada puede pertenecer, ser administrada y ser operada por una sola organización o por un tercero existiendo ciertos acuerdos de utilización, administración y seguridad través de un acuerdo entre la organización y el operador.

- Nubes comunitarias: los servicios se encuentran disponibles a un conjunto de entidades o consumidores, limitados a una relación de colaboración. Solo pueden acceder los grupos o consumidores que hacen parte de un acuerdo o relación. Una nube comunitaria puede pertenecer, ser administrada y ser operada por

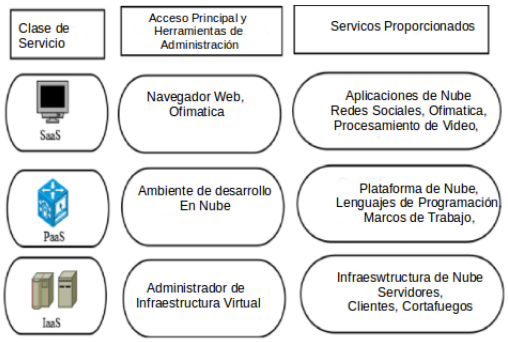


Figura 3.2.4.: Despliegues,control y ejemplos tomado de [Marinescu, 2013] capítulo 1 página 7.

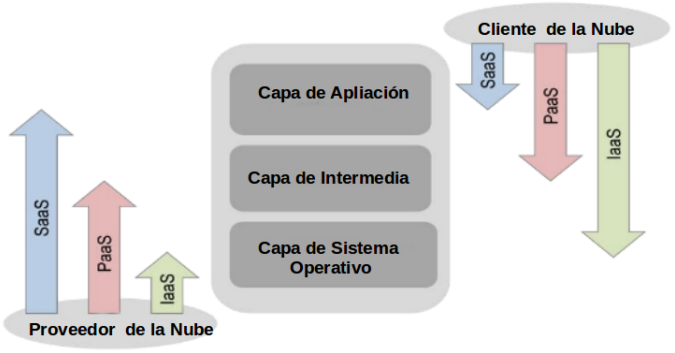


Figura 3.2.5.: Alcance en el control entre proveedor y consumidor de servicios tomado de [Fang Liu and Leaf, 2011] página 9.

uno o más miembros de la comunidad, un tercero o una combinación de estos. En relación a restricciones, pueden existir ciertos acuerdos sobre su uso.

- ***Nubes híbridas:*** los servicios se encuentran disponibles a través de la combinación de dos de los modelos anteriores. Cada modelo seleccionado tiene sus propias restricciones por lo que deben tener en cuenta al momento de realizar la selección.

3.3. Arquitectura

Para mejorar la comprensión de la manera cómo interactúan cada uno de los elementos y los actores, NIST presenta una arquitectura genérica de alto nivel ubicando en forma

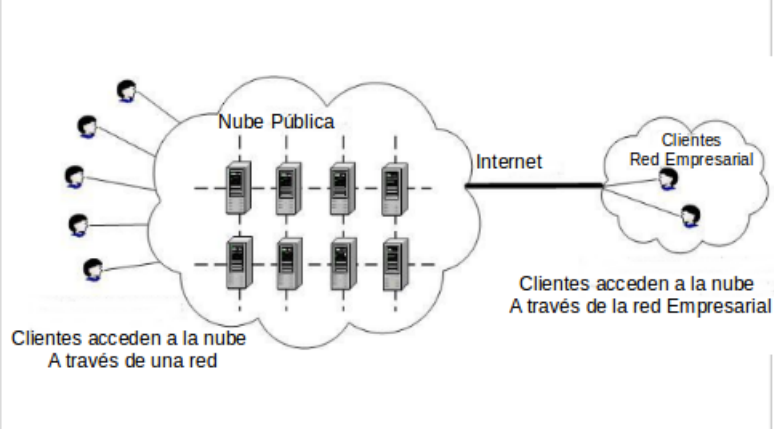


Figura 3.2.6.: Nube pública tomada de [Fang Liu and Leaf, 2011] página 10.

modular: los requerimientos, los usos, las características y los estándares necesarios en la computación en la nube. Dicha arquitectura es presentada en la figura 3.3.1.

De la figura 3.3.1 pueden ser identificados cinco actores: consumidor, proveedor, auditor, portador y negociador. Cada uno de estos actores es una entidad que puede ser una persona o una organización que participa en una transacción o proceso realizando una tarea específica. Las definiciones a continuación son tomadas de [Fang Liu and Leaf, 2011].

3.3.1. El consumidor (*cloud consumer*)

Es el principal interesado en los servicios ofrecidos en la computación en la nube. Representa una persona u organización que tiene una relación de negocios con el proveedor de servicios, para la utilización de los servicios acordados. Los requerimientos técnicos son especificados a través de los acuerdos de niveles de servicio, acuerdos relacionados con: la calidad del servicio, la seguridad, el rendimiento y la solución de fallas, limitaciones y obligaciones. Dependiendo de los servicios contratados, las actividades y los escenarios seleccionados, los consumidores pueden ser ubicados en los modelos de despliegue, como se ilustra en la figura 3.2.8.

3.3.2. El proveedor (*cloud provider*)

Es una persona u organización responsable de hacer accesible los servicios a las partes interesadas. El proveedor es el encargado de administrar la infraestructura necesaria para: proveer los servicios, ejecutar el software que presta los servicios y disponer la entrega de estos a través de una red. Las actividades de un proveedor de servicios son:

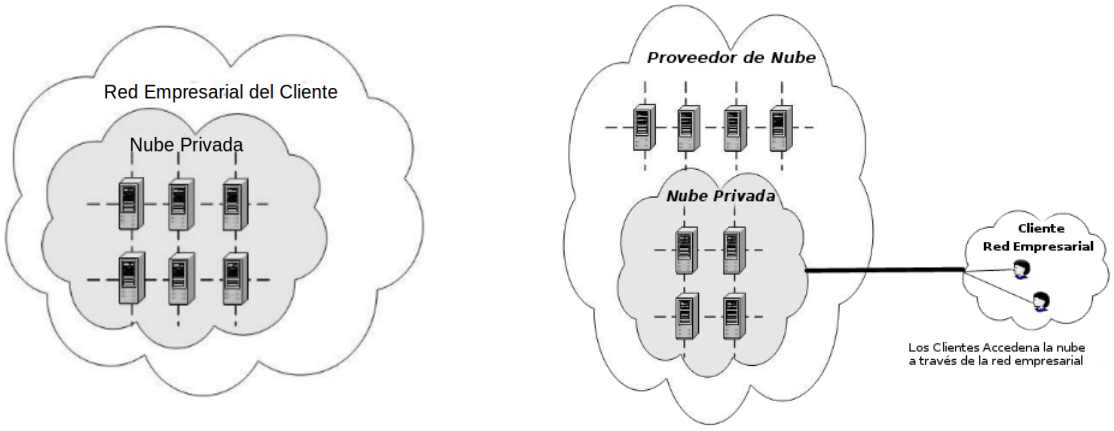


Figura 3.2.7.: Nubes privadas tomada de [Fang Liu and Leaf, 2011] páginas 10 y 11.

publicar, orquestar, administrar los servicios, brindar los mecanismos de seguridad y privacidad. Estos son ilustrados en la figura 3.3.3.

La publicación de los servicios es realizada dentro de los tres modelos aceptados y presentados en la sección anterior. La orquestación hace referencia a la composición de elementos, elementos que permitan ofrecer y soportar las actividades de disposición, coordinación y administración de los recursos que son ofrecidos a los consumidores. Esta actividad emplea un modelo de tres capas en el que se agrupan los elementos necesarios para que el proveedor entregue los servicios.

En la capa superior se encuentran los modelos de despliegue o presentación (SaaS, PaaS, IaaS). En la capa intermedia se realiza una abstracción de los recursos físicos en donde son administrados a través de software. Finalmente, en la capa más baja, se encuentran los recursos de hardware necesarios para soportar la operación. Cada capa depende de la subyacente, pero el consumidor no puede acceder a la capa de recursos físicos ó a la capa de abstracción. El modelo es ilustrado en la figura 3.3.4.

En la actividad de administración de servicios, el proveedor se encarga de realizar todas las tareas relacionadas y requeridas para el suministro y operación, adquiridos por el consumidor. Se pueden distinguir tres grupos: el grupo de mantenimiento del negocio, el grupo de aprovisionamiento y configuración y el grupo de portabilidad/interoperabilidad.

- El grupo de mantenimiento del negocio hace referencia a la negociación realizada entre el consumidor y el proveedor donde: se realiza la definición del soporte; se acuerdan contratos y precios, inventarios de servicios, y se realiza el reporte y la auditoría sobre las operaciones de los usuarios.
- En el grupo de aprovisionamiento y configuración, el proveedor define los niveles

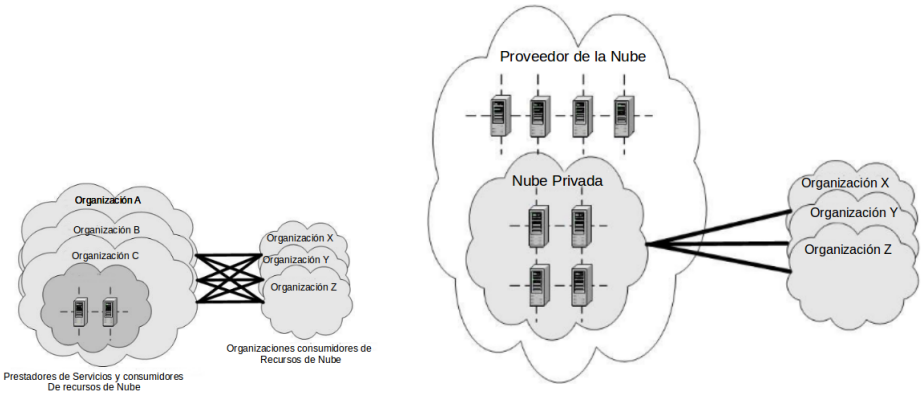


Figura 3.2.8.: Nube pública tomada de [Fang Liu and Leaf, 2011] páginas 11 y 12.

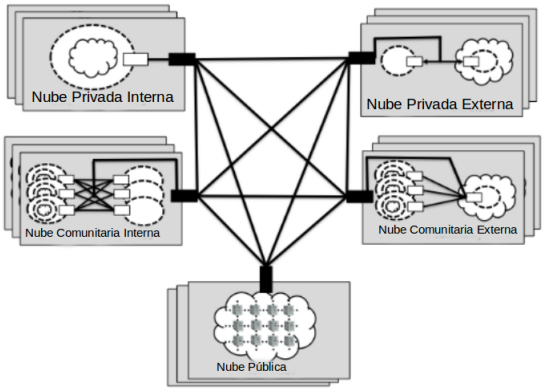


Figura 3.2.9.: Nubes privadas tomada de [Fang Liu and Leaf, 2011] página 12.

de acuerdos de servicios en términos de: calidad, mecanismos de escalamiento, actualización y cambio en los recursos inicialmente contratados.

- En el grupo de portabilidad e ínter-operabilidad, el proveedor de servicios debe facilitar los mecanismos de portabilidad de datos y sistemas, facilitando la copia de datos ya sea interna o externamente, tanto en la generación de copias de seguridad o donde se requiera la migración de los servicios a un nuevo proveedor. En el tema de ínter-operabilidad, se debe asegurar la comunicación de forma transparente entre diversas plataformas y proveedores con una interfaz de administración unificada.

En lo concerniente a la seguridad, las actividades realizadas por el proveedor están centradas en los mecanismos de: autenticación, autorización, disponibilidad, confidencialidad, administración de responsabilidades, integridad, auditoría, monitoreo, acciones

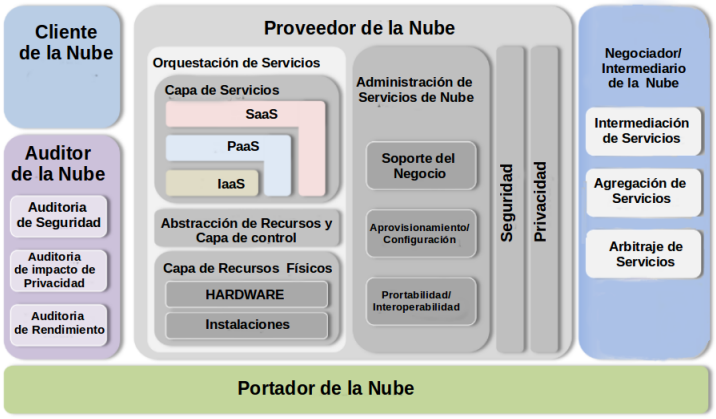


Figura 3.3.1.: Modelo conceptual de referencia tomado de [Fang Liu and Leaf, 2011] página 3.

a incidentes y administración de la seguridad.

En lo concerniente a la privacidad, el proveedor debe: proteger, garantizar y recolectar de forma segura y coherente la información que hace parte de la comunicación. En el caso específico de identificación y datos personales, se deben proveer mecanismos para que la recolección de esta información no sea accedida sin autorización expresa de los involucrados.

3.3.3. El auditor (Cloud Auditor)

Es un tercero que puede realizar una evaluación objetiva e independiente sobre la información, las operaciones, el rendimiento y la seguridad. El auditor puede evaluar los servicios que son proporcionados por el proveedor en términos de controles de seguridad, rendimiento, impacto en la privacidad y que tanto se cumplen los acuerdos entre el proveedor y el consumidor.

3.3.4. El Intermediario o negociador (Cloud Broker)

Dentro de la arquitectura de referencia propuesta en [Fang Liu and Leaf, 2011], el intermediario o negociador es una entidad que gestiona el uso, el rendimiento y la entrega de los servicios en la nube. Este agente es el encargado de negociar las relaciones entre el proveedor y el consumidor.

A medida que evoluciona la computación en la nube la integración de los servicios se hace más compleja de gestionar para los consumidores; en este caso los servicios

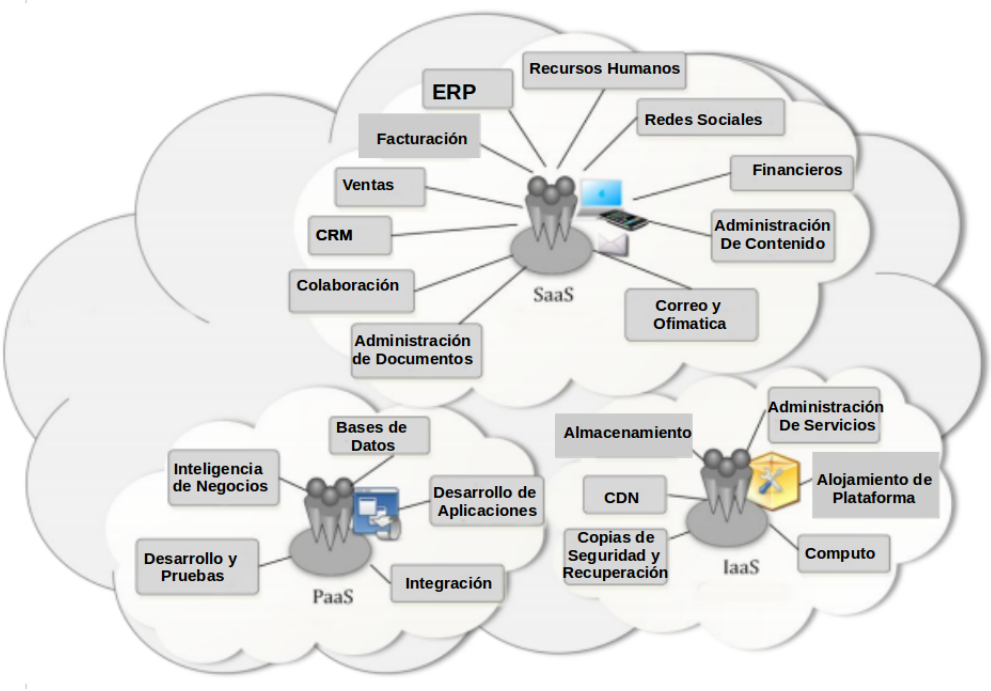


Figura 3.3.2.: Servicios disponibles para un consumidor tomada de [Fang Liu and Leaf, 2011] página 6.

son solicitados al negociador y no directamente al proveedor. Estos agentes proporcionan un único punto de entrada para la gestión de múltiples servicios. La principal diferencia con los proveedores es que el negociador proporciona una única interfaz consistente entre varios proveedores. Los servicios ofrecidos se encuentran clasificados en tres categorías:

- Negociación: el agente mejora la calidad de un servicio determinado, mediante el aumento de características específicas con el propósito de generar valor agregado a los servicios. El negociador puede administrar el acceso a los servicios, la gestión de identidad, generar informes de rendimiento y mejorar la seguridad.
- Agregación: el agente integra y combina múltiples servicios en uno o en varios, proporcionado la integración entre los datos y el servicio, aumentando la seguridad entre el consumidor y múltiples proveedores.
- Arbitraje: esta categoría es similar a la de agregación en el sentido de combinación y/o consolidación, pero los servicios resultantes no son fijos dado que el negociador puede seleccionar los servicios de múltiples proveedores.

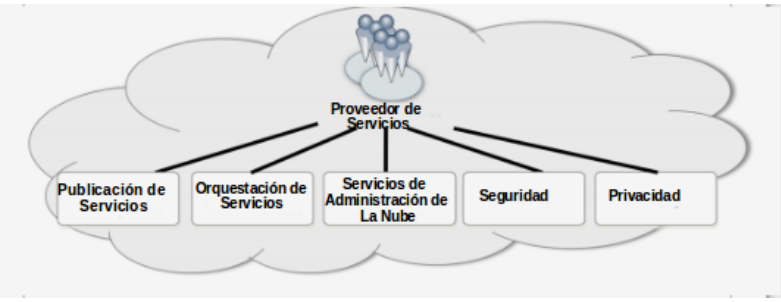


Figura 3.3.3.: Actividades del proveedor de servicios tomado de [Fang Liu and Leaf, 2011] página 7.

Figura 3.3.4.: Actividad de Orquestación tomado de [Fang Liu and Leaf, 2011] página 13

3.3.5. El portador (Cloud Carrier)

El portador actúa como un intermediario que provee el transporte de los servicios entre el proveedor y el consumidor. El portador es el responsable del acceso a través de canales de comunicación, por lo general redes de comunicación como internet. Los canales provistos por el portador pueden ser dedicados y encriptados.

3.4. Comparación entre agrupamientos de servidores, Mallas computacionales y computación en la nube.

Este apartado resume los capítulos dos y tres en el cual se pretende realizar una síntesis de los tres enfoques de sistemas distribuidos aquí presentados. La comparación

se realiza a través del cuadro 3.1, el cual permite identificar las ventajas y limitaciones de cada una.

Agrupaciones de Servidores	**Mallas Computacionales**	**Computación en la Nube**
Sistemas altamente acoplados, percibidos como un único sistema. La asignación y administración de trabajos centralizada.	Sistemas poco acoplados, descentralizados, dinámicos y diversos. La Asignación y administración de trabajos distribuida.	Infraestructura Dinámica,enfocados a la prestación de servicios. Plataforma con una administración o automática. El costo es basado en el consumo.
Principalmente utilizados en recursos educativos, sectores y productos comerciales e investigación.	Principalmente utilizados en : simulación y predicción de modelos, Diseño y automatización, investigación, exploración de recursos energéticos y exploración espacial.	Principalmente utilizados en entidades bancarias, exploración espacial,Investigación y simulación y predicción de modelos.
Los nodos son idénticos o muy similares, comparten el mismo dominio.	Los nodos no comparten la misma ubicación ni el mismo dominio.	Los nodos no comparten la misma ubicación ni el mismo dominio.
Todos los nodos tienen el mismo sistema operativo.	Los nodos no necesariamente tienen el mismo sistema operativo.	los nodos no necesariamente tienen el mismo sistema operativo.
El conjunto de nodos son percibidos como un único sistema (SSI).	Cada nodo actúa como entidad independiente.	Cada nodo actúa como entidad independiente.
Por lo general se encuentran en la misma ubicación geográfica.	Son inherentemente distribuidas a través de una LAN, o WAN.	dependiendo del modelo de despliegue son distribuidas a través de una LAN o una WAN.
El tamaño o escalabilidad es de cientos.	El tamaño o escalabilidad es de miles.	El tamaño o escalabilidad es de cientos a miles.
El sistema operativo es Unix o Windows.	Cualquier sistema operativo, pero dominan los Unix.	Cualquier sistema operativo que soporte Hypervisor, principalmente se utiliza virtualización para los nodos.
Pertenecen a un único dueño.	Múltiples dueños, pero pueden pertenecer a un único dueño.	dependiendo del despliegue puede ser un único dueño o múltiples dueños.
Por lo general son dedicados, con una baja latencia y un alto ancho de banda..	propósito general, tienen una alta latencia y bajo ancho de banda.	dedicados, propósito general, baja latencia, alto ancho de banda.
la negociación de los servicios es limitad.	La negociación de los servicios depende de los SLA.	La negociación de los servicios depende de los SLA.
Administración de usuarios y recursos es centralizada.	Administración de usuarios descentralizada o bajo una organización virtual, la administración de recursos es descentralizada.	Administración de usuarios centralizada o delegada a un tercero, la administración de recursos es descentralizada y/o centralizada.
El acceso se realiza a través de mecanismos tradicionales, privacidad media dependiendo de los privilegios del usuario.	Au.ztenticación en mecanismos de llave pública y asignados a un usuario soporte limitado para privacidad.	Se garantiza alta seguridad (multi-husped), soporta listas de control de acceso a los servicios.
Descubrimiento de servicios solo entre los miembros.	Manejo de un índice centralizado y descubrimiento descentralizado de la información de los servicios.	Descubrimiento de servicios solo entre los miembros.
Capacidad estable y garantizada.	Varia, pero por lo general tiene una alta capacidad.	Se proporcionan bajo demanda
Administración de fallos es limitada, cuando un servicio falla es reiniciado	Administración de fallos es limitada, cuando un servicio falla es reiniciado	Fuerte soporte a fallos y contenido replicado. Si una máquina virtual falla puede ser migrada a otro nodo
Limitados al precio	Precios son dominados por los servicios públicos, predomina el compartir	El precio depende del consumo.
Multi-agrupamiento para el trabajo colaborativo	Limitado, pero bastante explorado en esfuerzos orientados a investigación y académicos	La colaboración puede ser alta al interactuar con varios proveedores

Table 3.1.: Comparación entre agrupaciones de servidores, mallas computacionales y Computación en la nube, tomado de [Kaur and Rai, 2014] y de [Buyya et al., 2009]

Para finalizar esta exploración se presenta la tendencia de cada uno de los paradigmas, donde se puede apreciar como la implementación de agrupaciones se encuentra decayendo mientras que la de las nubes se incrementa. En el caso de las mallas computacionales la tendencia es baja y sigue en decible, debido principalmente a su baja adopción en la industria. La tendencia puede ilustrarse en figura 3.4.1.

En el caso particular de Colombia es más pronunciada la disminución en la imple-

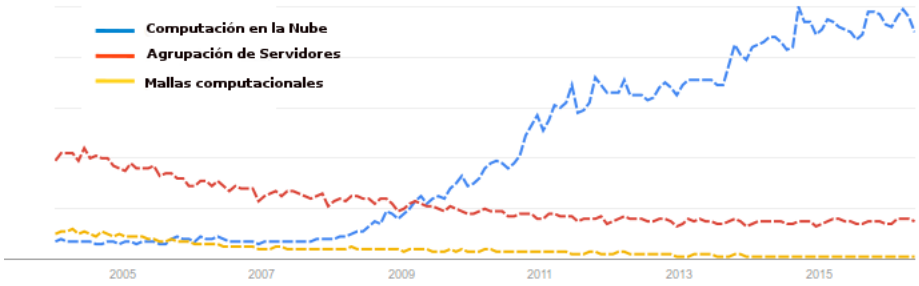


Figura 3.4.1.: Tendencia de las mallas computacionales, agrupación de servidores y computación en la nube tomado de google trends con un periodo comprendido entre 2004 al 2016

mentación de agrupaciones de servidores. Coincide con el crecimiento a nivel global la utilización de la computación en la nube, pero a una menor escala.

En el caso de las mallas la tendencia es muy similar a la presentada a nivel global, aunque con leves crecimientos en 2005. La tendencia puede ilustrarse en la figura 3.4.2.

Figura 3.4.2.: Tendencia para Colombia de las mallas computacionales, agrupación de servidores y computación en la nube tomado de google trends con un periodo comprendido entre 2004 al 2016

4. Balanceo de Carga

En este capítulo se presenta su definición, su clasificación y las técnicas seleccionadas para el desarrollo del presente estudio.

4.1. Definición

Es un mecanismo para distribuir carga o tareas en cada uno de los elementos, con la finalidad de optimizar la utilización de los recursos. El objetivo es maximizar la eficiencia y minimizar el tiempo de la respuesta para evitar la sobrecarga del sistema [Younis and El Halees, 2015], específicamente, respecto a la computación en la nube, el balanceo de carga es empleado para minimizar el tiempo de espera en la localización de los recursos y en la distribución de los trabajos (carga) ya sea en recursos físicos o virtuales.

El mecanismo se enfoca en la evaluación del rendimiento, en cada uno de los elementos que pertenecen al sistema, utilizando herramientas que le permiten seleccionar el recurso que realizara la tarea. La finalidad es asegurar que todos los elementos se encuentren con un nivel de ocupación similar, evitando así cuellos de botella y situaciones en las que algunos nodos se encuentren con una carga alta mientras que otros se encuentren ociosos o con una baja ocupación.

La figura 4.1.1 ilustra la situación donde la asignación de tareas no se encuentra equilibrada, mientras que la figura 4.1.2 se puede observar un sistema en donde la carga se encuentra similar en sus nodos.

Figura 4.1.1.: Balanceo de Carga desequilibrado tomado de [Angonese, 2012] capítulo 1 pagina 28.

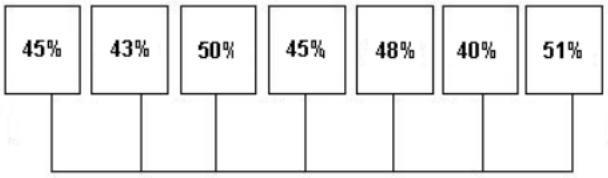


Figura 4.1.2.: Balanceo de Carga equilibrado tomado de [Angonese, 2012] capítulo 1 pagina 28.

El mecanismo también es empleado para el manejo de fallos, identificando cuando se presentan, con el propósito de tomar la alternativa de migrar la máquina virtual o de distribuir los trabajos asignados en los demás nodos. Esto significa que un balanceador de carga puede detectar una falla en uno o más de sus componentes y proporcionar una alternativa en la que las instancias son aprovisionadas o desactivadas sin necesidad de realizar cambios en la configuración de la red. Esta característica es tomada de las mallas computacionales.

Aunque el fundamento del balanceo de carga de la computación en la nube está basado en mecanismos aplicados, tanto a agrupamientos de servidores como a mallas computacionales, este se diferencia principalmente en que el cálculo se basa en la información respecto al uso de los servidores, impulsando nuevas oportunidades y economías de escala [Younis and El Halees, 2015].

De una manera formal, el mecanismo puede ser representado como un conjunto de recursos de computo (máquinas virtuales), de naturaleza heterogénea. Las ecuaciones (4.1.1) a (4.1.7) fueron tomadas de [Younis and El Halees, 2015]. En donde se tiene un conjunto de máquinas vituales o recursos de computo V

$$V = \{v_1, v_2, v_3, \ldots, v_n\} \tag{4.1.1}$$

Al cual se desean asignar un conjunto de cargas o tareas L.

$$L = \{l_1, l_2, l_3, \ldots, l_n\} \tag{4.1.2}$$

Donde la carga actual DL en el centro de datos puede ser expresada cómo:

$$DL = \{vl_1, vl_2, vl_3, \ldots, vl_n\} \tag{4.1.3}$$

Y se requiere encontrar una función $\psi(l_i)$ en donde el conjunto de cargas L puede ser asignado al conjunto de recursos de computo V, haciendo que la carga vl_i en cada una de las unidades de computo v_i sean similares como se muestra en la ecuación (4.1.4).

$$vl_1 \approx vl_2 \approx vl_3 \ldots \approx vl_n \tag{4.1.4}$$

Sea τ_o el tiempo empleado para realizar las tareas de l_0 en la unidad de computo v_i, el tiempo total para ejecutar todas las tareas esta dado por la ecuación (4.1.5).

$$t_i = \sum_{o \epsilon \psi(l_i)(i=1 \ldots n)} \tau_o \tag{4.1.5}$$

En el caso que solo se tenga una unidad de computo (máquina virtual), el tiempo empleado para realizar todas las tareas, es la suma de el tiempo empleado en la ejecución de forma secuencial como lo ilustra la ecuación (4.1.6).

$$t_1 = \sum_{o(o=1 \ldots n)} \tau_o \tag{4.1.6}$$

Si el conjunto de unidades de computo tiene una cardinalidad superior a 1, las tareas son distribuidas en los nodos, para ser ejecutadas en paralelo como lo muestra la ecuación (4.1.7).

$$t_k = max_{i=1,\ldots,n} t_i \tag{4.1.7}$$

Por lo que el objetivo principal del balanceo de carga es encontrar la función $\psi(l_i)$ que minimice el tiempo t_k donde se obtenga el caso expuesto en la ecuación (4.1.4).

La solución de la función $\psi(l_i)$ es expresado como un problema combinatorio.

4.2. Clasificación

El balanceo de carga en computación en la nube puede ser clasificado en dos grandes grupos: estáticos y dinámicos.

4.2.1. Balanceo de carga estático

En esta clasificación la carga es dividida en de forma equivalente entre las unidades de computo. Se dispone un conocimiento previo de la cantidad de solicitudes que realiza el usuario y su tamaño, y por lo general se asume que todas las solicitudes realizadas toman el mismo tiempo en ser ejecutadas. En este enfoque, la carga en los servidores no es evaluado, razón por la que la complejidad en el diseño de los mecanismos es menor, pero es común que se presenten momentos en el que el conjunto se encuentre en un estado de desequilibrio por un largo periodo de tiempo, como el de la figura 4.1.1.

En este enfoque toma una alta relevancia el cálculo de la capacidad a ser aprovisionada, dado que una vez implementada no es posible modificar las características y los procesos son difíciles de migrar[Angonese, 2012]. Los algoritmos más populares que se encuentran en esta clasificación son:

- ***Aleatorio (Random):*** este algoritmo distribuye la carga seleccionado de forma aleatoria un nodo entre los elementos disponibles, su implementación es sencilla, pero tiene el problema de seleccionar nodos que se encuentren con un alta carga mientras hay nodos ociosos o que su estado es no disponible, su tolerancia a fallos es cuestionable dado que la selección no verifica si el nodo se encuentra disponible. La decisión solo depende del generador de números aleatorios implementado.
- ***Asignación Circular o Planificación circular (Round Robin):*** implementa una lista del tipo primero en entrar, primero en salir o primero en llegar, primero en ser atendido (FIFO por sus siglas en inglés First In, First Out), la carga es asignada a cada nodo dependiendo del indice que le corresponda, es decir, distribuye la carga en un nodo a la vez por turno, hasta completar la lista y empezar con el primero nuevamente.
- ***Asignación Circular ponderada o Planificación circular ponderada (Weighted Round Robin):*** al igual que el algoritmo de asignación circular implementa una lista con los nodos disponibles, pero cada nodo recibe una ponderación, mejorando la asignación en ambientes heterogéneos, ya que asigna trabajos dependiendo de la ponderación dada a cada uno de los nodos. La ponderación esta definida por la capacidad de cada elemento.
- ***Más rápido en contestar***: el algoritmo más rápido en contestar (*Fastest*) entrega una nueva asignación al nodo que tenga el menor tiempo de respuesta de todos los nodos disponibles. Este algoritmo puede ser particularmente útil en ambientes donde los servidores están distribuidos en diferentes redes lógicas.
- ***Menos Conexiones***: este algoritmo, asigna al servidor que tenga la menor cantidad de asignaciones en el momento de llegada, el algoritmo no tiene en cuenta las capacidades de los nodos, por lo que funciona mejor en sistemas

homogéneos. Las lista de conexiones, por lo general es almacenada en un nodo central que mantiene un índice con las asignaciones realizadas a cada nodo.

4.2.2. Balanceo de carga dinámico

A diferencia del balanceo de carga estático en el balanceo de carga dinámico toma una alta relevancia el estado actual del sistema, por lo que no es necesario tener un conocimiento previo del número de solicitudes o del tamaño de las mismas, adquiriendo una ventaja sobre al momento de tomar la decisión de la asignación. Los algoritmos de este tipo permiten a los procesos migrar a otros nodos una vez que ellos han iniciado su ejecución.

El objetivo de migrar procesos o tareas es identificar quien puede realizar de forma mas efectiva la ejecución de la asignación. Las decisiones son hechas usando la información sobre el estado del sistema actual, esto es, cada nodo debe conocer el estado de los otros nodos. Tomar las decisiones de forma dinámica es mucho más complicado que en forma estática, debido principalmente a la necesidad de recolectar y mantener la información del estado de los nodos[Kameda et al., 1997].

Kameda y otros en[Kameda et al., 1997] define que algoritmos dinámicos de balanceo de carga están compuestos por tres políticas básicas:

- ***Política de Información:*** responsable de recolectar el estado del sistema. La información es recolectada de dos formas la primera denominada bajo demanda, en la cual el estado de cada uno de los nodos es solicitado por un nodo central o por un nodo participante y la segunda en la que cada nodo informa a los demás del estado ya sea periódicamente o al finalizar una asignación.
- ***Política de Migración:*** responsable de determinar si un nodo se encuentra en la capacidad de participar en la ejecución de una tarea o por el contrario si es mejor transferir la asignación a un nodo con mejor estado. Por lo general la política se basa en un umbral de ocupación, para definir si es apto o no. Tales umbrales son expresados en unidades de carga, y la migración es iniciada cuando se detecta que la carga entre los nodos no se encuentra balanceada.
- ***Política de localización:*** responsable de determinar cual de los nodos participantes es el más apto para ejecutar la asignación. La política puede ser, sin estado de información, en la cual no se emplea información del estado de los otros nodos, y con estado de información, en la cual se realiza un análisis de estado de un nodo y se verifica si en el nodo se puede realizar la asignación.

En el campo del balanceo dinámico los algoritmos más populares son:

- ***Cola centralizada:*** existe un nodo denominado el gestor de la cola, el cual es encargado de almacenar las tareas y de solicitarlas las actividades dependiendo de un umbral definido en cada uno de los nodos.

- ***Menor Número de Conexiones:*** recolecta la información de los nodos y selecciona el que menor número de conexiones activas tiene, con el propósito de garantiza el equilibrio de la carga. Este algoritmo es uno de los mas populares
- ***Min-min*** : en este algoritmo se calcula el tiempo que tarda una asignación en ser ejecutada, para seleccionar el nodo que realiza dicha ejecución en el menor tiempo posible. En el caso de Min-Min el algoritmo funciona en dos fases, en la primera se calcula el tiempo mínimo de espera de cada asignación y el tiempo total de ejecución de las asignaciones, en la segunda se selecciona la tarea con menor tiempo.
- ***Max-min:*** el algoritmo calcula el tiempo máximo de ejecución de las tareas, y trabaja en dos fases en la primera se calcula el tiempo total para la ejecución de las tareas en cada una de los nodos, y en la segunda fase se selecciona el nodo con un mayor tiempo de espera.

4.3. Trabajos relacionados

Esta sección se presenta una conjunto de trabajos y estudios relacionados con el tema de balanceo de carga dinámica que fueron revisados dentro de la literatura para este estudio y que fueron considerados relevantes.

El estudio realizado por Cybenko en [Cybenko, 1989] presenta un modelo de balanceo de carga dinámica para la asignación en el paso de mensajes en un entorno con varios procesadores en red. Modela el mecanismo a través matrices donde se almacenan las asignaciones de las tareas. El objetivo del modelo es distribuir de forma equitativa la carga en los nodos, formulando el problema como un hipercubo. Construye una matriz que representa un grafo, en el cual los vértices son cada uno de los nodos y las aristas son cada una de las rutas. La distribución de la carga es descrita como un vector en el cual se almacenan el número de tareas que posee un procesador en un tiempo t.

Se emplean modelos de cadenas de Markov, asumiendo que la carga tiene una distribución de probabilidad inicial y que la asignación del trabajo es matemáticamente idéntica a la evaluación de la probabilidad.

Subrata y colegas en [Subrata et al., 2007] realizan un estudio en el cual abordan el problema de balanceo de carga en malla computacional con algoritmos genéticos y búsqueda tabú (*tabú search*), la implementaciones son comparadas con *Best-fit*, Aleatorio, Min-min, *Max-min*, y *Sufferage*. Presentan como resultado que la implementaciones propuestas poseen menor tiempo de repuesta que los algoritmos tradicionales, pero incurren en un mayor almacenamiento y procesamiento en cada uno de los nodos.

Mientras que Sethi y colegas en [Sethi et al., 2012] presentan la utilización de lógica difusa para abordar el problema de balanceo de carga modificando el algoritmo de

Asignación Circular, este enfoque pretende simular la toma de decisiones realizada por los humanos, basado en reglas de control difuso y parámetros lingüísticos.

En relación a estudios comparativos Ray y De Serkar [Ray and De Sarkar, 2012], realiza una comparación de algoritmos *Enrutamiento por Testigo (Token Routing*), *Asignación Circular (Round Robin), Cola Central, (Central Queuing) y Menor número de Conexiones (Least Connection).* Realizan la implementación de los algoritmos en la herramienta *cloud-sim*, comparando el desempeño en el tiempo de respuesta y la utilización en cada una de las máquinas virtuales. Mientras que Supreeth y Biradar en [Supreeth and Biradar, 2013] realizan un análisis comparativo de diferentes algoritmos *voraces (Gready)* y *Asignación Circular* para el balanceo de carga en maquinas virtuales, identificando sus ventajas y desventajas. Proponen una modificación del algoritmo de *Asignación Circular* en donde se adicionan valores de peso a cada una de las maquinas para realizar la distribución de los trabajos, y comparan los resultados entre el algoritmo propuesto y el algoritmo tradicional de *Asignación Circular*.

Chen y colegas en [Chen et al., 2013], proponen una estrategia basada en Optimización por Colonia de Hormigas (*ACO Ant Colony Optimization* por sus siglas en ingles) para la asignación de la carga. El objetivo principal es identificar una solución óptima global al problema de localización de recursos en un entorno de computación en la nube, inspirado en el comportamiento social de estos insectos, comparando esta técnica con algoritmos tradicionales.

Otro trabajo relacionado con hormigas es es el presentado por Tawfeek, Medhat y colegas en [Tawfeek et al., 2013] donde implementa el algoritmo de colonias de hormigas multi-objetivo (*MOACO Multi-Objetive Ant Colony Optimization* por su siglas en inglés). Para proponer una solución al el problemas de asignación, es utilizando el mismo criterio de auto-adaptación del algoritmo MOACO tradicional. El enfoque propuesto se encuentra orientado a la optimización multi-objetivo, asumiendo que las hormigas viajan a través de los nodos para construir la mejor solución. El propósito es identificar los nodos en estado de espera o con menor carga. La implementación y experimentación se realiza a través de simulaciones donde se se definen como variables de estudio: la longitud de la tarea y la capacidad de procesamiento de cada una de los nodos. Los resultados son comparados con un algoritmo *voraz,* y con el algoritmo Primero en llegar primero en ser servido (FCFC *Firt Come First Server* por sus siglas en inglés).

Sun, Hong y colegas en [Sun et al., 2013] presentan la utilización de diversas técnicas entre las cuales se encuentran colonias de hormigas, *map-reduce*, asignación basa en agentes, redes de *petri* y basado en costo estándar. Realizan la comparación a través de la simulación con la cual presentan resultados de dicha comparación. Mientras que el trabajo presentado por Song y colegas en [Song et al., 2014], presentan un enfoque en el cual utilizan teoría de juegos con la finalidad de modelar el problema de balanceo de carga a través de juegos cooperativos, y no cooperativos, la comparación de la

propuesta la realizan a través de simulación de las dos estrategias

Otro enfoque es el presentado por Katyal y Mishra en [Katyal and Mishra, 2014] presentan un completo compendio de algoritmos de balanceo de carga de diversas técnicas y realizan una comparación de su aplicabilidad en cada una de los escenarios que pueden ser planificados o implementados para ofrecer una solución de computación en la nube. Raghava y Singh en [Raghava and Singh, 2014] también presentan un estudio comparativo de diversos algoritmos de balanceo de carga, pero analizan aspectos como el rendimiento, sobrecarga, tolerancia a fallos, tiempo de migración de tareas, tiempo de respuesta, tiempo de migración de tareas, utilización de recursos, y la complejidad de cada uno de los algoritmos, los resultados son obtenidos a través de la revisión de literatura realizada por los autores, no se identifica si realizan la implementación de los algoritmos presentados.

Bilgaiyan y colegas en [Bilgaiyan et al., 2015], presentan una recopilación de los algoritmos propuestos para abordar el problema del balanceo de cargas a través de técnicas evolutivas y de enjambre, en las que se encuentran algoritmos genéticos, técnicas basadas en partículas (PSO Particle Swarm Optimization por sus siglas en ingles), Colonias de hormigas y de Abejas, el estudio presenta los objetivos, la finalidad y las ventajas.

Por otra parte Nipane y colegas en [Nipane and Dhande, 2014] plantean la utilización de colonias artificiales de abejas, para abordar problemas como el aprovisionamiento automático de servicios, migración de maquinas virtuales, administración de energía, y almacenamiento de datos, otro enfoque relacionado con estos insectos es el propuesto por Bitam Salim en [Bitam, 2012] quien presenta la utilización del algoritmo de vida de las abejas Abejas (*BLA) (Bee Live Algorithm* por sus siglas en inglés), abordando el problema de asignación de tareas.

En el estudio definen un operador binario y dos puntos de cruce aleatorios con una probabilidad de 0.95, en la mutación utilizan un operador unitario de acuerdo a una probabilidad de 0.01. En la búsqueda del alimento proponen enfoque voraz *(gready)* para seleccionar el mejor individuo de la vecindad, el tamaño de la poblaciones es de 4 zánganos y 5 trabajadoras, definen 3 regiones por iteración y 15 abejas reclutadas, los resultados son comparados con el enfoque de un algoritmo genético.

El trabajo presentado por Mistry Bhargav en [Mistry, 2013] realiza simulación de balanceo de carga utilizando la biblioteca (MASS) (*Multi-agent Spatial Simulation* por sus siglas en inglés), implementando tres algoritmos: basados en el histórico de tiempo, una ventana de tiempo reciente y en el crecimiento del uso de la CPU. En los resultados se presenta la comparación entre los tres enfoques seleccionados.

El estudio realizado por Florence y Shanti en [Florence and Shanthi, 2014] propone la utilización de un algoritmo basado en el comportamiento de luciérnagas (*FIREFLY*), definen la utilización de una lista que almacena un índice con cada una de las solicitudes realizadas (cola de asignaciones). En base a los índices se formula la localización de los

recursos para realizar asignación de tareas. Los criterios de evaluación son la longitud de las colas, la taza de utilización de CPU y la cantidad de memoria empleada.

Kumar y Shobana [Kumar and Shobana,], presentan un enfoque en el cual implementan el algoritmo del descenso de agua (*Water Drops*), comparándolo con los un algoritmos basados en Colonias de Hormigas y de Partículas. El documento revisado no presentan resultados de experimentación o hacen mención a la implementación.

Desde el enfoque energético, Kansal y colegas [Kansal et al., 2010] realizan un estudio comparativo entre diferentes algoritmos de balanceo con el propósito de evitar las situaciones desequilibrada, ya que hay una relación directa entre la distribución los trabajos con el consumo de energía. Mientras que Adnan y colegas en [Adnan et al., 2012] proponen un algoritmo cuya finalidad es encontrar un distribución eficiente en términos energéticos dada una distribución geográfica de los centros de datos. El objetivo principal es optimizar la utilización y disminución de la latencia de las solicitudes a ser asignadas obteniendo colas de trabajo mas pequeñas evitando la sobrecarga de las máquinas.

Otro estudio perteneciente a las iniciativas de computación amigable con el ambiente (*green computing*), es el enfoque propuesto por Shu, Wanneng y colegas en [Shu et al., 2014] en el que se emplean algoritmos basados en sistemas inmunes implementando el algoritmo de selección clonal para la localización eficiente de recursos.

En la literatura también fueron identificados trabajos relacionados con MANET (*Mobile ad hoc network* por sus siglas en inglés), abordando el problema de balanceo de carga en la ubicación de recursos.

4.4. Algoritmos *de Colonia de Hormigas Artificiales (ACO), Colonia de Abejas Artificiales(ABC) y Algoritmo de Luciérnagas (FA)*

4.4.1. Algoritmo de Optimización de Colonia de Hormigas

Uno de los primeros comportamientos estudiados por la entomología es la habilidad de las hormigas para encontrar la ruta más corta desde el nido a la fuente de alimento. Estos estudios y observaciones inspiraron a Marco Doriogo en su disertación doctoral en 1992 a formular los primeros modelos basados en este comportamiento [Engelbrecht, 2007].

La hipótesis que inspiro estos trabajos está relacionada con la pregunta: ¿Cómo las hormigas buscan el alimento sin ningún mecanismo visible, central, o de coordinación?.

Estudios del comportamiento de especies de hormigas reales, han concluido que al inicio la actividad tiene un patrón caótico y altamente aleatorio, pero tan pronto

es localizada la ruta más cercana se puede diferenciar un patrón más organizado y observándose un aumento de hormigas siguiendo la ruta mas corta.

El comportamiento emergente es el resultado de un mecanismo de reclutamiento donde las hormigas que han hallado alimento influencian a otras hacia la fuente seleccionada. Este reclutamiento es realizado a través de una comunicación indirecta empleando una sustancia química denominada feromona. Cuando una hormiga encuentra una fuente de alimento lleva el alimento al nido y deposita una cantidad de feromona en el camino. Únicamente las hormigas exploradoras deciden que ruta seleccionar dependiendo de la cantidad de feromona depositada. Las rutas con alta concentración de feromona tienen una alta probabilidad de ser seleccionada.

Entre más hormigas seleccionen una ruta mayor cantidad de la sustancia es depositada, atrayendo a un mayor número de individuos que seguirán dicho camino. En este estudio cuando una solicitud de un usuario (*cloudlet*) es recibida, el negociador (*cloud broker*) solicita la disponibilidad de los recursos de cada uno de los nodos. Los recursos son: cantidad de memoria disponible, el ancho de banda por consumir y la utilización de la CPU.

Tomando estas variables el valor de feromona inicial en el el nodo i está dado por:

$$\tau_i(0) = (c + m + b) \tag{4.4.1}$$

Donde c representa el porcentaje de CPU disponible, m la memoria disponible y b el ancho de banda disponible para consumir. Si se toma el tiempo cero (0) entonces el valor es uno (1), ya que el nodo tiene disponible toda la capacidad dispuesta a ser utilizada.

La intensidad del rastro en el tiempo t esta definida por (4.4.2).

$$\tau_i(t) = \rho * \tau_i(t-1) + \Delta\tau_i(t-1, t) \tag{4.4.2}$$

donde $\Delta\tau_i(t-1, t)$, es la variación de la cantidad del rastro depositada en el recurso i entre el tiempo *t-1* y t; ρ es la permanencia de feromona la cual esta en el rango $(0 < \rho < 1)$; por lo que $(1 - \rho)$ es la evaporación de la sustancia.

Cuando al nodo (i) se le asigna una tarea para ser ejecutada es actualizado $\Delta\tau_i(t-1, t)$.

la probabilidad que una área se asignada al cada nodo esta dad por la ecuación (4.4.3)

$$\rho = \frac{\tau_i(t)^\alpha \eta(t)^\beta}{\sum \tau_i(t)^\alpha \eta(t)^\beta} \tag{4.4.3}$$

Donde el $\eta(t)^{\beta}$ es denominado el factor de visibilidad, es decir este factor representa la cantidad de recursos disponibles para realizar la tarea en el tiempo (t).

La asignación de las tareas a los nodos es basado en la cantidad de feromona calculada, la cual es generada por el nivel de ocupación actual en cada uno de los nodos. El nodo con mayor cantidad de recursos es que tiene mejor probabilidad de ser seleccionado.

Algoritmo 4.1 Algoritmo de optimización de Colonia de Hormigas *(ACO)*

```
t=0, inicializar parámetros α, β, ρ,
inicializar la población de hormigas
para cada hormiga seleccione un nodo de forma aleatoria
repita
    para cada hormiga i = 1...n haga
        repita
            de forma aleatoria visite los nodos
            construya los enlaces (i,j)
        hasta nodos de destino sean rechazados(ya han sido visitados)
    fin para
        para cada enlace (i,j) haga
            reduzca la feromona
        fin para
        para cada i = 1...n haga
            para cada enlace(i,j) haga
                calcule Δt
                calcule ρ
            fin para
        fin para
        t=t+1
hasta condición de parada
retorne nodo con la menor ocupación
```

4.4.2. Algoritmo de Colonia de Abejas Artificiales

Está técnica es inspirada en el comportamiento de búsqueda de fuentes de alimento de las abejas, estos insectos viven como una familia, en donde todos los miembros son encargados de realizar un labor compleja. Cada abeja tiene un comportamiento (social) individual y colectivo que es utilizado para: la comunicación, construcción y el cumplimiento de responsabilidades. Fue propuesta por Karaboga en el 2005 [Karaboga and Basturk, 2007].

En el algoritmo se pueden reconocer tres tipos de abejas: espectadoras, empleadas y exploradoras. Para cada fuente de alimento existe únicamente una abeja empleada y

cuando se agota el alimento se convierte en una abeja exploradora. En el algoritmo la posición de una fuente de alimento representa una solución candidata y la cantidad del néctar corresponde a la aptitud (fitnes). El número de abejas empleadas es igual al número de soluciones.

Las abejas exploradoras se mueven de forma aleatoria en un área alrededor de la colmena evaluando las de las fuentes de alimento. Cuando retorna al panal la exploradora deposita el néctar recolectado y se dirige a una zona conocida como la "pista de baile", para llevar acabo un ritual denominado la danza de la abeja.

A través de la danza, la exploradora comunica el descubrimiento de las fuentes de alimento a las abejas espectadoras, las cuales se unen en la exploración del alimento. La duración del ritual es proporcional a la calificación dada por la exploradora. Al finalizar la danza, la exploradora regresa a la fuente de alimento para seguir recolectando.

En el algoritmo la posición de una fuente de alimento representa una solución candidata y la cantidad del néctar corresponde a la aptitud (*fitnes*). El número de abejas empleadas es igual al número de soluciones.

Si se toman los recursos como:

$$r_i = (c + m + a) \tag{4.4.4}$$

Donde c representa el porcentaje de disponible de CPU, m la memoria sin utilizar y b el ancho de banda disponible para consumir. En el tiempo cero el valor es 1, ya que el nodo tiene todo los recursos disponibles para ser utilizados.

La cantidad de néctar de la región para ejecutar la tarea está dada por (4.4.5)

$$R_i(t) = \frac{r_i\,(t)}{\sum r_i\,(t)} \tag{4.4.5}$$

Donde $R_i(t)$, es el porcentaje de recursos disponibles por nodo y $r_i(t)$ son los recursos disponibles del nodo.

La asignación de las tareas es basado en el porcentaje de recursos disponibles, el cual es definido por el nivel de ocupación actual. El nodo con mayor cantidad de recursos es el seleccionado para ejecutar los trabajos.

4.4.3. Algoritmo de Luciérnagas Artificiales

Esta técnica se encuentra inspirada en los patrones de destello luminiscente y de comportamiento de las luciérnagas. En esencia cada luciérnaga es atraída por el brillo de otra, mientras que al mismo tiempo explora y busca presas de forma aleatoria. El algoritmo fue propuesto por Xin-She Yand en el 2007 [Yang, 2013].

Se asume:

Algorithm 4.2 Algoritmo de Colonia de Abejas Artificiales

inicializar parámetros trabajadoras,exploradoras ,empleadas
inicializar la población de forma aleatoria
repita
  para cada abeja $i = 1 \dots n$
    evaluar la aptitud de la abeja
  fin para
  seleccionar los m regiones para la búsqueda de las soluciones
  determinar el tamaño de la región
  reclutar abejas para las regiones seleccionadas
  seleccione la abeja con mejor aptitud del área
  memorizar la mejor fuente de alimento
hasta condición de parada

- Son asexuales, es decir una luciérnaga es atraída por otra sin importar el sexo.
- La atracción es proporcional al brillo despedido. Una luciérnaga es atraída a otra que posee una mayor intensidad de brillo, este decrece en ambas a medida que la distancia se incrementa. Si la luciérnaga no encuentra una fuente de brillo se mueve de forma aleatoria por el espacio de búsqueda.
- El brillo esta determinado por la región de la función objetivo.

El brillo de de la luciérnaga en una localización x esta dada por $I(x)$. La atracción β es relativa, variando con respecto a la distancia r_{ij} entre la luciérnaga i y la j

En este estudio la intensidad del brillo inicial esta dada por la ecuación (4.4.6)

$$I(0) = c + m + b \tag{4.4.6}$$

La intensidad del brillo decrece con respecto a la distancia por lo que la atracción puede variar dependiendo del coeficiente de absorción. Dado un coeficiente de absorción fijo γ la intensidad I varia con la distancia r de la siguiente la ecuación (4.4.7).

$$I = I_0 \epsilon^{-\gamma r} \tag{4.4.7}$$

La atracción es proporcional a la intensidad de la luz vista por las luciérnagas adyacentes a esta, es definida como se presenta en la ecuación (4.4.8):

$$\beta = \beta_0 \epsilon^{\gamma r^2} \tag{4.4.8}$$

Donde β_0 es la atracción en $r = 0$ y r es definida por la ecuación (4.4.9)

$$\|x_i - x_j\| = \sqrt{(x_i - x_j)^2} \tag{4.4.9}$$

En el caso del balanceo de carga a la atracción esta definida por la cantidad de recursos de cada nodo.

El movimiento de la luciérnaga es gradualmente reducido como lo propone la técnica de templado simulado de la siguiente forma:

$$\alpha = \alpha_0 \delta \qquad donde \qquad 0 < \delta < 1$$

Algorithm 4.3 Algoritmo de Luciérnagas Artificiales

inicializar parámetros α, γ,
inicializar la población de luciérnagas
para cada luciérnaga seleccione un nodo de forma aleatoria
repita
  para cada luciérnaga $i = 1 \ldots n$ *haga*
    formule la Intensidad del brillo asociada a la función
  fin para
    defina coeficiente de absorción
  repita
    para cada luciérnaga $i = 1 \ldots n$ *haga*
      para cada cada luciérnaga $j = 1 \ldots n$ *haga*
        si (Intensidad de brillo(j) >Intensidad de brillo(i))
          mueva (i) hacia(j)
          varíe la atracción con distancia r vías $\epsilon^{-\gamma r}$
          evalué las solución y actualice la intensidad
        fin si
      fin para
    fin para
    ordene las luciérnagas y seleccione la mejor solución
hasta condición de parada
retorne nodo con la menor ocupación

4.5. Metodología de superficies de respuesta (RSM)

La metodología de superficies de respuesta (RSM *Response Surface Metodology* por sus siglas en inglés), es un conjunto de herramientas matemáticas y estadísticas utilizadas en el tratamiento de problemas, en los cuales una respuesta de interés está influenciada por varios factores. La técnica fue introducida por Box y Wilson en los años cincuenta [Oscar Orlanos Melo, 2007], con un amplia utilización en procesos industriales, principalmente en procesos químicos.

El propósito inicial es diseñar un experimento que proporcione valores razonables de la variable de repuesta, con el propósito determinar el modelo matemático que mejor se ajusta a los datos obtenidos, para así poder determinar los valores de los factores que mejor optimizan la variable de observación [Cepillo, 2011].

Cuando se expresa el valor esperado η, que toma la respuesta, este valor se encuentra influenciado por los niveles de los k factores cuantitativos $x_1, x_2, \ldots, x_k$, en donde existe una función de $f(x_1, x_2, \ldots, x_k)$ continua que proporciona el valor η para alguna combinación de los factores [Cepillo, 2011].

$$\eta = f(x_1, x_2, \ldots, x_k) \tag{4.5.1}$$

de tal forma que la respuesta está dada por la ecuación 4.5.2

$$y = \eta + \varepsilon = f(x_1, x_2, \ldots, x_k) + \varepsilon \tag{4.5.2}$$

Donde ε, es el error de la respuesta. La relación $\eta = f(x_1, x_2, \ldots, x_k)$ y los factores puede ser expresada como una hiper-superficie o subconjunto de un espacio Euclideano de $k+1$ dimensiones, denominada superficie de respuesta. La hiper-superficie se puede representar como una ecuación polinomial por lo general de de primer o segundo grado.

El empleo de la metodología en este trabajo tiene dos etapas distintas, modelamiento y desplazamiento. Las etapas fueron repetidas hasta alcanzar una región, en la cual los parámetros de los algoritmos propuestos generan el mejor comportamiento. En la etapa de modelamiento, se emplean modelos simples lineales y/o cuadráticos para ajustar la respuesta obtenida.

La etapa de desplazamiento se da a lo largo del camino con una máxima inclinación, que es una trayectoria en la cual la respuesta varía de forma más pronunciada.

5. Simulación y Evaluación Propuesta.

En este capítulo se presentan los escenarios definidos para observar el comportamiento de las técnicas seleccionadas y se presenta la metodología empleada para la selección de los parámetros de cada una de las técnicas.

La construcción de la superficie se realizó con una configuración diferente de valores a los de los escenarios, con el propósito de poder evaluar como se comportan las técnicas en diferentes escenarios. El equipo en el cual fueron realizados cada uno de los experimentos cuenta con un procesador intel core i7® de cuarta generación con una velocidad de 2.6GHZ, 8 GB de memoria RAM y un disco duro de 1TB.

5.1. Escenarios de Simulación

Se han definido tres escenarios, en los cuales se configura el tamaño y número de centros de datos. El centro de datos está constituido por una red de comunicaciones donde se hospeda el conjunto de máquinas físicas y virtuales, y a la cual pueden acceder los clientes desde diferentes regiones geográficas.

Como herramienta para realizar la simulación se emplea *Cloud-Analyst*, la cual es una herramienta de código abierto, derivada de *Cloud Sim*. Esta herramienta proporciona una interfaz gráfica para la configuración de los parámetros a ser simulados, lo cual facilita la definición y ejecución de los experimentos.

En todos los escenarios propuestos, se selecciona como tiempo a simular un valor de 120 horas, es decir cinco (5) días de trabajo, y picos de carga de los usuarios en diferentes horarios como se ilustra en la figura 5.1.1. Cada conjunto de usuarios simula la solicitud de peticiones (cloudlets), con un promedio de 1000 usuarios. Estos parámetros de carga y tiempo son iguales en todos los escenarios.

5.1.1. Escenario pequeño

Este escenario pretende simular centros de datos pequeños, en donde se ha configurado pocas prestaciones de computo. La configuración evaluá el comportamiento de las técnicas donde los recursos son escasos. La configuración es ilustrada por la figura 5.1.2

Name	Region	Requests ... User per Hr	Data Size per Reque... (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Pe... Users
UB1	1	60	100	9	12	1000	100
UB2	2	60	100	8	12	1000	100
UB3	3	60	100	14	18	1000	100
UB4	4	60	100	15	22	1000	100

Figura 5.1.1.: Definición de los usuarios y sus características

Service Broker Poli... Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	4	100000	5242880	1000

Figura 5.1.2.: Centro de Datos para el escenario pequeño

5.1.2. Escenario mediano

Este escenario pretende simular un centro de datos mediano. Se simulan dos centros de datos distribuidos geográficamente, cada uno con diez (10) máquinas virtuales. La configuración pretende evaluar las técnicas en un entorno con prestaciones moderadas. La configuración es ilustrada por la figura 5.1.3.

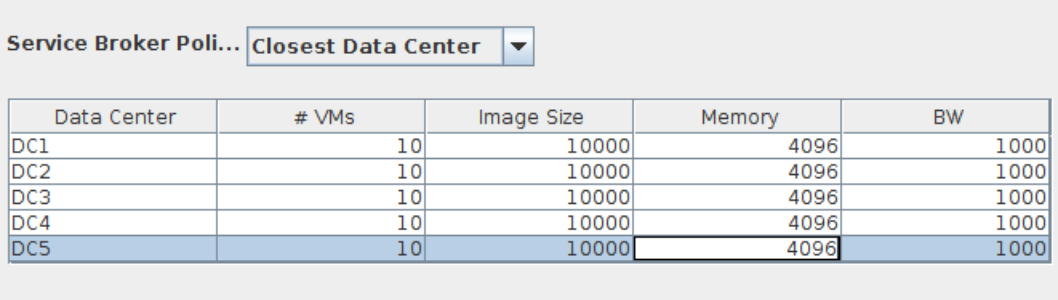
Service Broker Poli... Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	10	100000	5242880	1000
DC2	10	100000	5242880	1000

Figura 5.1.3.: Configuración Centros de Datos Escenario Mediano

5.1.3. Escenario grande

Este escenario pretende simular un escenario con prestaciones no tan moderadas el cual se posee cinco (5) centros de datos distribuidos geográficamente cada uno con diez (10) máquinas virtuales asignadas, pretende evaluar las técnicas en donde se tiene acceso a un conjunto amplio de recursos. La configuración se muestra en la figura 5.1.4.



Data Center	# VMs	Image Size	Memory	BW
DC1	10	10000	4096	1000
DC2	10	10000	4096	1000
DC3	10	10000	4096	1000
DC4	10	10000	4096	1000
DC5	10	10000	4096	1000

Figura 5.1.4.: Configuración centros de datos Escenario de gran tamaño

5.2. Selección de parámetros

Teniendo presente que la selección de los parámetros de las técnicas impacta los resultados, y con el propósito de asegurar que la elección de los parámetros sea la mejor posible, se construyó una superficie de respuesta para cada una de las técnicas seleccionadas con el propósito de obtener los parámetros óptimos.

Para la selección de las corridas iniciales de cada algoritmo se ejecutaron 2^n, experimentos factoriales para alimentar la superficie de respuesta inicial, donde n es el número de parámetros empleados por la técnica.

Los parámetros de la simulación para la construcción de la respuesta es el mismo para las tres técnicas y la configuración cuenta con dos (2) centros de datos cada uno con diez (10) máquinas virtuales cada uno y cinco (5) regiones de usuarios distribuidas geográficamente, los cuales generan peticiones presentando picos de sobrecarga a diferentes horas por región.

Por cada nuevo valor de la superficie se realizó la simulación treinta (30) veces y se cálculo el promedio de la respuesta para alimentar el nuevo valor y ajustar el modelo de la superficie para la etapa de desplazamiento.

5.2.1. Superficie para la selección de parámetros para la técnica *de Colonia de Hormigas Artificiales (ACO)*

para la técnica de colonia de hormigas se construyó una superficie para la selección de los parámetros α, β, ρ. El cuadro 5.1 presenta los valores generados en la construcción de la superficie de respuesta y la figura 5.1.1, la gráfica de la superficie generada y sus respectivos contornos.

El modelo que mejor respuesta genero fue un modelo de primer orden con dos interacciones entre los parámetros α, β y dos interacciones entre los parámetros β, ρ, y términos cuadráticos de los factores α, β, la formula en el paquete estadístico R (r-cran) fue la siguiente: $FO(\alpha, \beta, \rho) + TWI(\alpha, \beta) + TWI(\beta, \rho) + PQ(\alpha, \beta)$, En está técnica se alcanzó el punto estacionario en veinte (20) iteraciones y la mejor respuesta fue generada en la iteración dieciocho (18).

Los valores de la iteración dieciocho son los seleccionados realizar la simulación en los escenarios de prueba, ya que según la el cuadro de la superficie son los valores con el menor tiempo de respuesta obtenidos.

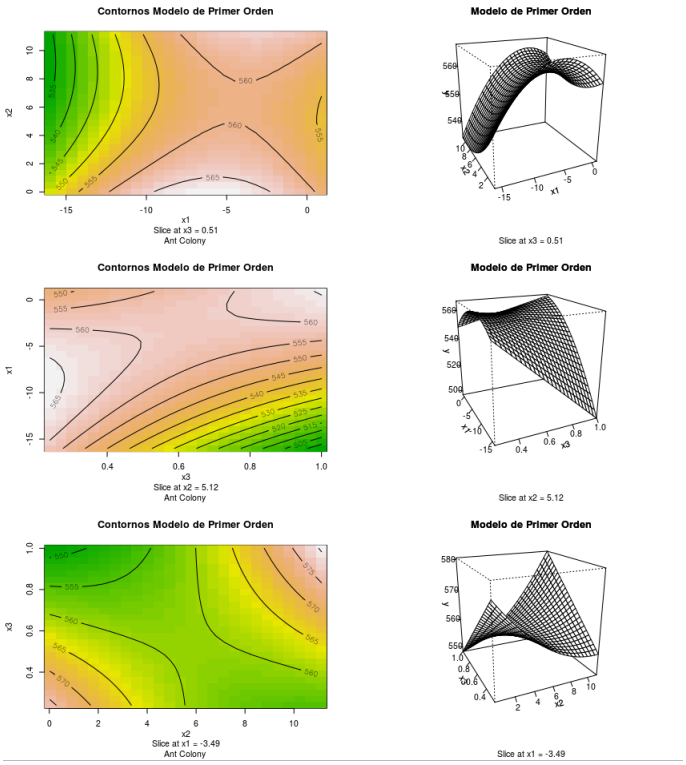


Figura 5.2.1.: Superficie para el algoritmo *ACO fuente* fuente elaboración propia.

Iteración RSM	α	β	ρ	y
1	0.7	4.889534	0.8	558.97
2	0.8	5	1	567.545
3	0.9	3.41	1	558.34
4	0.5	2	0.5	558.35
5	0.15	9	0.5	558.35
6	0.6	0.03	0.7	558.89
7	-0.15	0.96	0.47	558.27
8	-0.17	2.15	0.35	558.27
9	-016	2.075	0.351	558.31
10	0.18	1.5	0.526	558.30
11	-0.16	2.324	0.35879	558.37
12	-0.1667	2.3218	0.35852	558.37
13	-0.166662	2.2830	0.3577	558.31
14	-8.0699	10.6383197	0.25555	558.32
15	-8.07551	10.849739	0.24	558.34
16	-8.2853776	11.1236	0.50833	558.26
17	-8.27045	9.59541	0.5066	558.31
18	***-8.1025341***	***6.62593***	***0.4411***	***557.90***
19	-7.9983139	8.0244452	0.45127	558.31
20	-7.9995262	7.686657	0.4486705	558.33

Cuadro 5.1.: Valores de la superficie de respuesta para el algoritmo *ACO*.

5.2.2. Superficie para la selección de parámetros para la técnica *de Colonia de Abejas Artificiales (ABC)*

Para la técnica de colonia de abejas artificiales se construyó una superficie para la selección de los porcentajes de abejas exploradoras, trabajadoras y espectadoras, que son los parámetros de los cuales depende el algoritmo.

El cuadro 5.2, y la figura figura 5.2.1 muestran la superficie generada.El modelo que mejor respuesta generó fue un modelo de primer orden con dos interacciones y un término cuadrático, el modelo en R fue el siguiente: $FO(trabajadoras, expectadoras, exploradoras)+$ $TWI(trabajadoras, expectadoras, exploradoras)+TWI(trabajadoras, expectadoras)+$ $TWI(trabajadoras, expectadoras) + PQ(expectadoras$en está técnica el menor valor fue encontrado en la segunda iteración. El punto estacionario se encontró con 18 iteraciones.

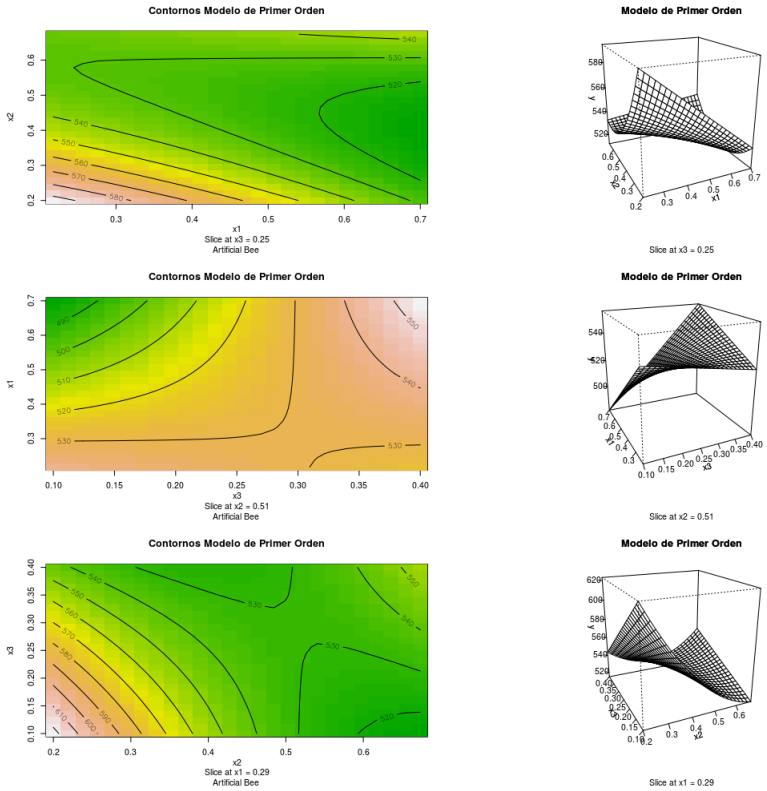


Figura 5.2.2.: Superficie para el algoritmo *ABC fuente* fuente elaboración propia

Iteración RSM	Trabajadoras	Expectadoras	Exploradoras	y
1	.5	.25	.25	490.24
2	***.8***	***.1***	***.1***	***490.14***
3	.8	.0	.2	490.58
4	.7	.0	.3	490.38
5	.7	.2	.1	490.91
6	.5	.3	.2	490.69
7	.3	.3	.4	490.54
8	.3 .	3	.4	490.36
9	.3	.5	.2	490.81 .
10	1 .5	.3	.2	490.23
11	.3	.5	.2	491.04
12	26	.41	.34	490.18
13	.18	.5	.37	490.41
14	0.2733997	0.3991648	0.3404078	490.23
15	0.2659883	0.4045610	0.3427149	490.61
16	0.2971891	0.3798508	0.3325279	491.13
17	0.3515216	0.3468346	0.3177336	490.58
18	0.3618620	0.3407488	0.3150958	490.48

Cuadro 5.2.: Valores de la superficie de respuesta para el algoritmo *ABC* fuente elaboración propia.

5.2.3. Superficie para la selección de parámetros para la técnica *Luciérnagas Artificiales (AF)*

En lo que corresponde a la la técnica de luciérnagas artificiales se construyó una superficie para la selección de los parámetros α, γ, α_0, en cada iteración se ejecutó treinta (30) veces. El cuadro 5.2, y la figura figura 5.2.1 muestran la superficie generada.

El modelo que mejor respuesta generó fue un modelo de segundo orden con términos cuadráticos y dos interacciones $SO(\alpha, \gamma, \alpha_0) + FO(\alpha, \gamma, \alpha_0) + TWI(\alpha, \gamma, \alpha_0) + PQ(\alpha, \gamma, \alpha_0)$, la superficie fue construida con 33 iteraciones, obteniendo mejor valor de salida en la sexta iteración.

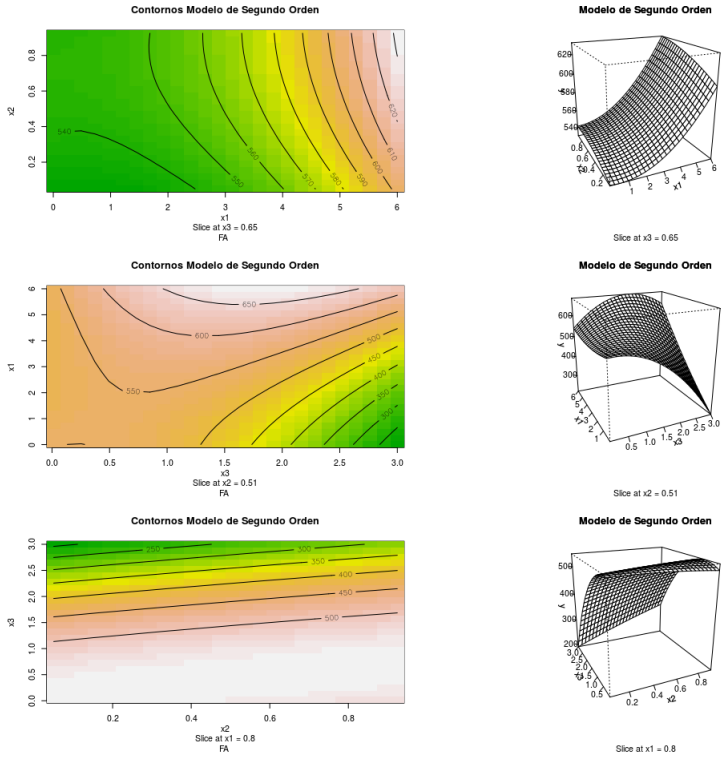


Figura 5.2.3.: Superficie para el algoritmo *FA* fuente elaboración propia

Iteración RSM	α	α_0	γ	y
1	1	0.9	0.7	544.28
2	0.9	0.5	0.8	544.54
3	0.5	0.3	0.5	544.94
4	0.67597549	0.72589368	0.01472459	532.55
5	0.4400869	0.2620797	0.8172006	530.30
6	***0.4321494***	***0.2561135***	***0.8621533***	***529.10***
7	0.4374362	0.2555387	0.7757953	532.30
8	0.4261746	0.2428275	0.7954456	529.29
9	0.4323387	0.2487501	0.7767674	529.33
10	0.4353511	0.2522269	0.7720699	529.32
11	3	0.07	0.9	545.15
12	6	0.05	3	544.27
13	0.3060497	0.9246206	0.2267383	542.78
14	0.4323387	0.2522269	0.4700293	545.11
15	0.7599573	0.6448757	0.4628823	545.9
16	0.4225819	0.7073014	0.5004606	544.52
17	0.009854902	0.782867209	0.577468997	544.05
18	0.03276754	0.79029018	0.58642481	546.19
19	.03934364	0.77947715	0.57135866	544.25
20	0.02164604	0.78305166	0.57501384	544.03
21	0.007036518	0.785635477	0.578046985	545.14
22	0.3937061	0.6982434	0.5259661	545.13
23	0.3920555	0.7009852	0.5302704	545.5
24	0.3897900	0.7044769	0.5359166	544.27
25	0.3908546	0.7031143	0.5335893	547.36
26	0.3850845	0.7117487	0.5478692	543.89
27	0.3874488	0.7099783	0.5441348	542.33
28	0.3928599	0.7054800	0.5353219	544.27
29	0.3935933	0.7046366	0.5339079	542.96
30	0.67597549	0.72589368	0.01472459	532.55
31	0.4400869	0.2620797	0.8172006	530.30
32	0.4446448	0.2644144	0.7676077	532.63
33	0.4374362	0.2555387	0.7757953	532.30

Cuadro 5.3.: Valores de la superficie de respuesta para el algoritmo *AF*

5.3. Resultados

5.3.1. Escenario pequeño

Para el escenario en donde los recursos de computo son escasos los resultados son presentados en el cuadro 5.7 muestra los resultados para los tres algoritmos.

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*455.18*	*68.29*	*3072.68*	*135.209*	*428.21*	*59.8*	*3023.51*	*135.46*	*443.74*	*61.05*	*3072.68*	*135.26*
Tiempo de Procesamiento en el centro de datos	*219.35*	*12.58*	*2717.47*	*27.69*	*196.97*	*12.55*	*2500.8*	*23.92*	*210.36*	*12.58*	*2500.81*	*27.04*

Table 5.4.: Resultados para la simulación en un ambiente con recursos escasos en un centro de datos con configuración estática

En este escenario la técnica que mejores resultados presento fue la técnica de las abejas, con un promedio de 428.21(ms), seguida por las luciérnagas con un 443.74 (ms), la técnica con mayor tiempo de respuesta fue la de hormigas con un promedio de 455.18 (ms).

Con respeto al tiempo máximo de respuesta en el centro de datos se presento un empate entre las técnicas de abejas y luciérnagas, mientras que las hormigas presentaron un valor un más elevado. Como valor mínimo presentaron los mismos valores.

En el caso que el centro de datos se re-configure de forma dinámica los tiempos máximos y mínimos observados fueron muy similares para las tres técnicas, en particular para las luciérnagas y las hormigas. El mejor tiempo de respuesta fue obtenido por las colonia de abejas artificiales.

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*2743.40*	*163.26*	*30709.88*	*135.47*	*2676.13*	*163.26*	*300507.96*	*135.20*	*2716.92*	*163.26*	*30709.88*	*135.26*
Tiempo de Procesamiento en el centro de datos	*2362.89*	*106.26*	*30531.14*	*263.98*	*2300.91*	*105.01*	*30355.00*	*263.25*	*2338.40*	*106.26*	*30530.51*	*263.32*

Table 5.5.: Resultados para la simulación en un ambiente con recursos escasos en un centro de datos con re-configuración dinámica

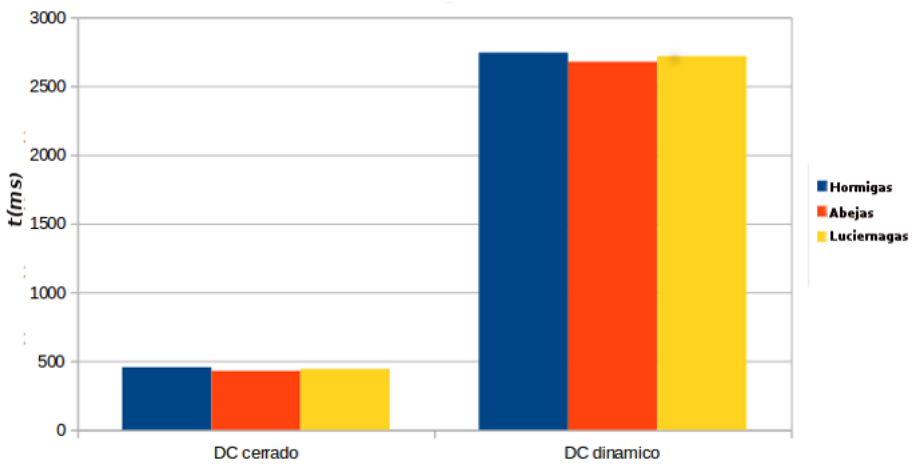


Figura 5.3.1.: Comparación de las técnicas seleccionadas en con configuración dinámica y cerrada fuente elaboración propia

Para comparar los resultados obtenidos se realizó un análisis de varianza, empleando el test ANOVA, y se verifica los residuos utilizando los métodos gráficos qq-plot e histograma. Tanto el diagrama de QQ-plot como el de histograma permiten observar los residuos del anova no se distribuyen de forma normal, por lo que para verificar si existen diferencias significativas se emplea el test de Mann-Whitney-Wilcoxon, el cual arroja como resultado V = 101480, p-value $<$ 2.2e-16 con lo cual se rechaza la hipótesis nula, la cual corresponde a que las medias de las tres técnicas es igual.

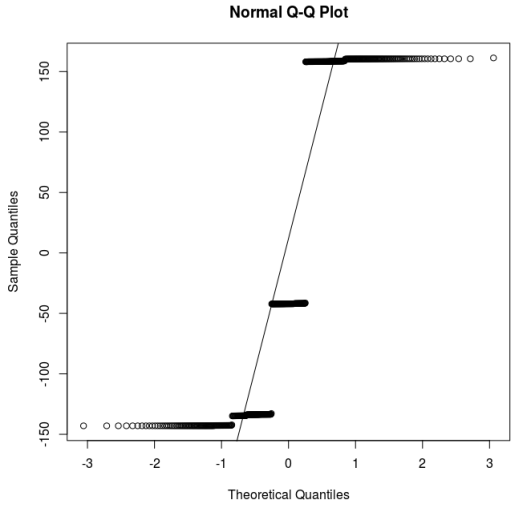


Figura 5.3.2.: Diagrama QQ-Plot de los residuos de las muestras resultantes para el escenario pequeño fuente elaboración propia

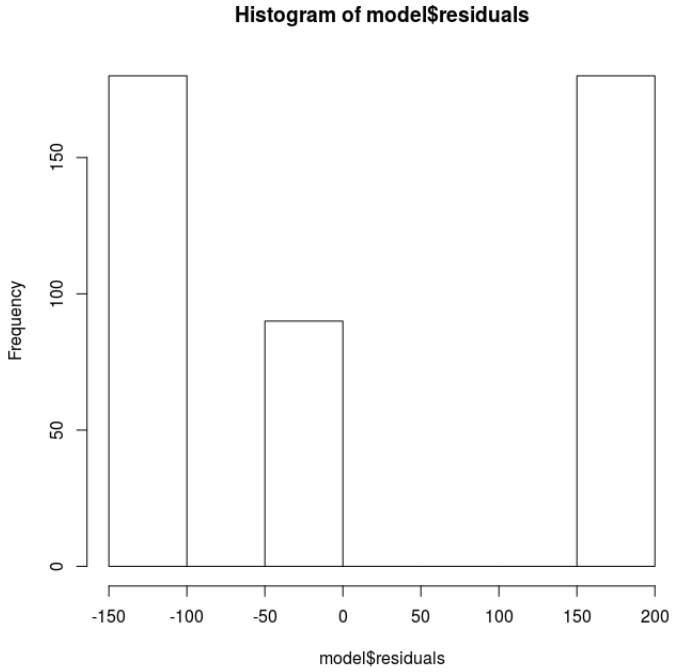


Figura 5.3.3.: Histograma de los residuos de las muestras resultantes del escenario pequeño fuente elaboración propia

Para confirmar que los datos no tienen una distribución normal, analíticamente se realizo el test de Shapiro-Wilk arrojando que las poblaciones no se distribuyen de forma normal. Los valores arrojados por el test se pueden observar en el cuadro 5.6

ACO		ABC		FA	
W	p-value	W	p-value	W	p-value
0.74041	*5.544e-15*	*0.7392*	*5.103e-15*	*0.74025*	*5.482e-15*

Table 5.6.: Resultados test Shapiro-Wilk para el escenario pequeño

5.3.2. Resultados en el escenario mediano

El cuadro 5.7 muestra los resultados para los tres algoritmos en el escenario mediano. En este escenario la técnica con un menor tiempo de respuesta fue la colonia de abejas con 145.88, seguida por las luciérnagas con 150.48, y por último la técnica de colonia de hormigas con 151.57. El menor tiempo correspondiente al máximo tiempo de respuesta correspondió a 54047.01 obtenido por la técnica de abejas, en el caso de las la otras dos técnicas el tiempo fue de 58953.25

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*151.78*	*35.66*	*58953.25*	*135.21*	*145.88*	*35.66*	*54047.01*	*135.47*	*148.10*	*35.66*	*58953.25*	*135.26*
Tiempo de Procesamiento en el centro de datos	*7.07*	*0*	*58899.50*	*6.31*	*1.06*	*0*	*53995.75*	*6.33*	*3.39*	*0*	*58899.50*	*6.28*

Table 5.7.: Resultados para la simulación en un ambiente mediano en recursos en un un centro de datos con configuración estática

En el caso que el centro de datos se re-configure de forma dinámica los el tiempo no disminuye en forma significativa, y el menor tiempo sigue siendo obtenido por la técnica de Abejas, aunque las hormigas disminuyen su diferencia con respecto a las luciérnagas.

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*151.57*	*35.66*	*58953.25*	*134.26*	*145.88*	*35.66*	*54047.01*	*133.56*	*150.48*	*35.66*	*58953.25*	*134.18*
Tiempo de Procesamiento en el centro de datos	*6.86*	*0.0*	*58899.50*	*6.31*	*1.17*	*0.0*	*53995.75*	*6.28*	*5.77*	*0.0*	*58899.50*	*6.30*

Table 5.8.: Resultados para la simulación en un ambiente mediano en recursos en un centro de datos con re-configuración dinámica

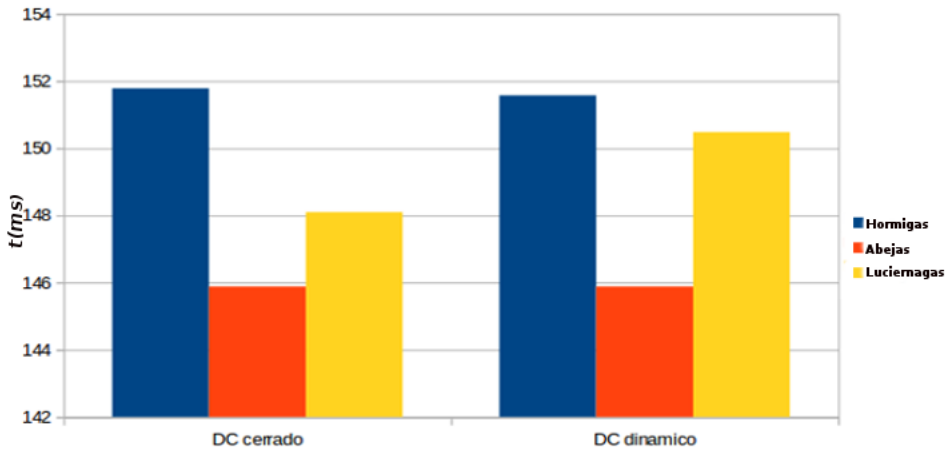


Figura 5.3.4.: Comparación de las técnicas seleccionadas en con configuración dinámica y cerrada

Para comparar los resultados obtenidos se realizó un análisis de varianza, empleando el test ANOVA, y se verifica los residuos utilizando los métodos gráficos qq-plot, histograma. Tanto el diagrama de QQ-plot como el de histograma representan que no se distribuyen de forma normal, por lo que para verificar si existen diferencias significativas se emplea el test de Mann-Whitney-Wilcoxon, el cual arroja como resultado V = 101480, p-value $<$ 2.2e-16 con lo cual se rechaza la hipótesis nula, la cual corresponde a que los valores tiene una media igual.

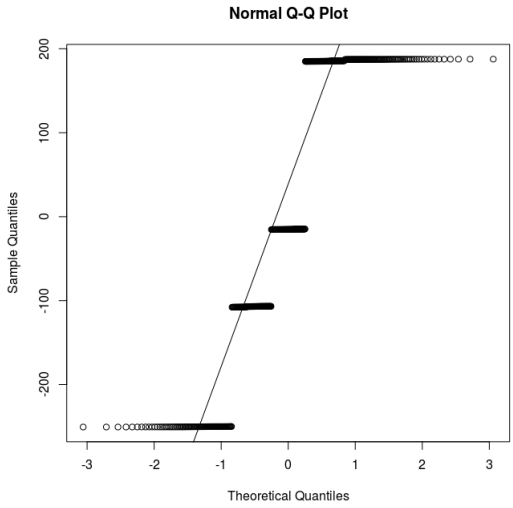


Figura 5.3.5.: Diagrama de QQ-Plot de los residuos de las muestras resultantes para el escenario pequeño fuente elaboración propia

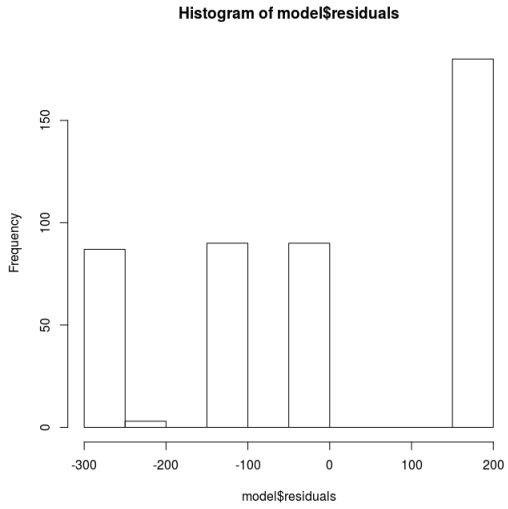


Figura 5.3.6.: Histograma de los residuos de las muestras resultantes para el escenario pequeño fuente elaboración propia

Para confirmar que los datos no tienen una distribución normal, analíticamente se realizó el test de Shapiro-Wilk arrojando que las poblaciones no se distribuyen de forma normal. Los valores arrojados por el tes se pueden observar en el cuadro 5.9

ACO		ABC		FA	
W	p-value	W	p-value	W	p-value
0.82066	*2.835e-12*	*0.82046*	*2.785e-12*	*0.82075*	*2.859e-12*

Table 5.9.: Resultados test Shapiro-Wilk para el escenario mediano

5.3.3. Resultado escenario grande

El cuadro 5.10 muestra los resultados para los tres algoritmos en el escenario mediano. En este escenario la técnica con un menor tiempo de respuesta fue la colonia de abejas con 80.44, seguida por las luciérnagas con 84.73, y por último la técnica de colonia de hormigas con 91.63. El menor tiempo fue igual para las tres técnicas el cual fue de 35.66, le menor de los tiempos máximos fue obtenido por la técnica de las abejas con un valor de 70245.27.

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*91.63*	*35.66*	*75647.26*	*69.58*	*80.44*	*35.66*	*70245.27*	*69.35*	*84.73*	*35.66*	*75647.26*	*69.49*
Tiempo de Procesamiento en el centro de datos	*14.62*	*0.11*	*75594.50*	*6.59*	*1.06*	*0.11*	*70192.51*	*6.28*	*7.72*	*0.11*	*75594.50*	*6.47*

Table 5.10.: Resultados para la simulación en un ambiente amplio en recursos en un centro de datos con configuración estática

Métrica	ACO				ABC				FA			
	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar	Promedio (ms)	Mínimo (ms)	Máximo (ms)	D. estándar
Tiempo de Respuesta	*78.63*	*35.66*	*75647.26*	*62.53*	*78.44*	*35.66*	*70245.27*	*62.29*	*78.53*	*35.66*	*75647.26*	*62.49*
Tiempo de Procesamiento en el centro de datos	*14.62*	*0*	*75594.50*	*6.45*	*1.06*	*0*	*70192.51*	*6.17*	*7.72*	*0*	*75594.50*	*6.25*

Table 5.11.: Resultados para la simulación en un ambiente amplio en recursos en un centro de datos con re-configuración dinámica

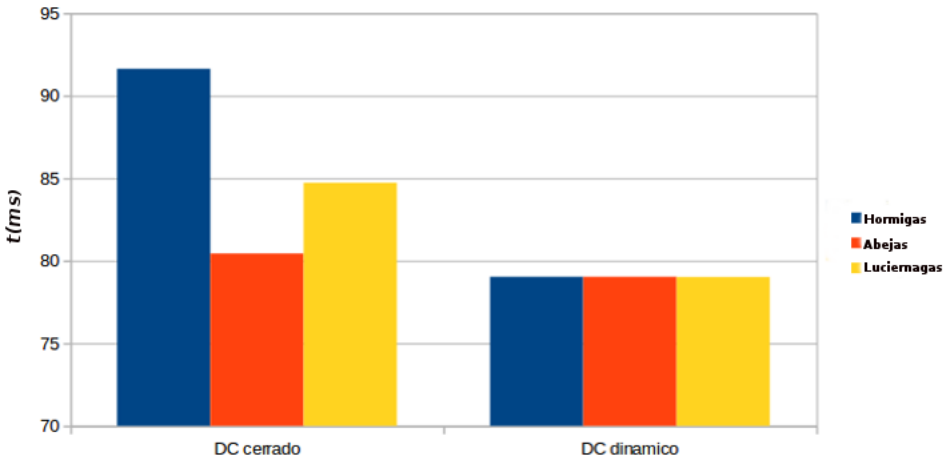


Figura 5.3.7.: Comparación de las técnicas seleccionadas en con configuración dinámica y cerrada.

Para comparar los resultados obtenidos se realizó un análisis de varianza, empleando el test ANOVA, y se verifica los residuos utilizando los métodos gráficos qq-plot, histograma. Tanto el diagrama de QQ-plot como el de histograma representan que no se distribuyen de forma normal, por lo que para verificar si existen diferencias significativas se emplea el test de Mann-Whitney-Wilcoxon, el cual arroja como resultado V = 101480, p-value $<$ 2.2e-16 con lo cual se rechaza la hipótesis nula, la cual corresponde a que los valores tiene una media igual.

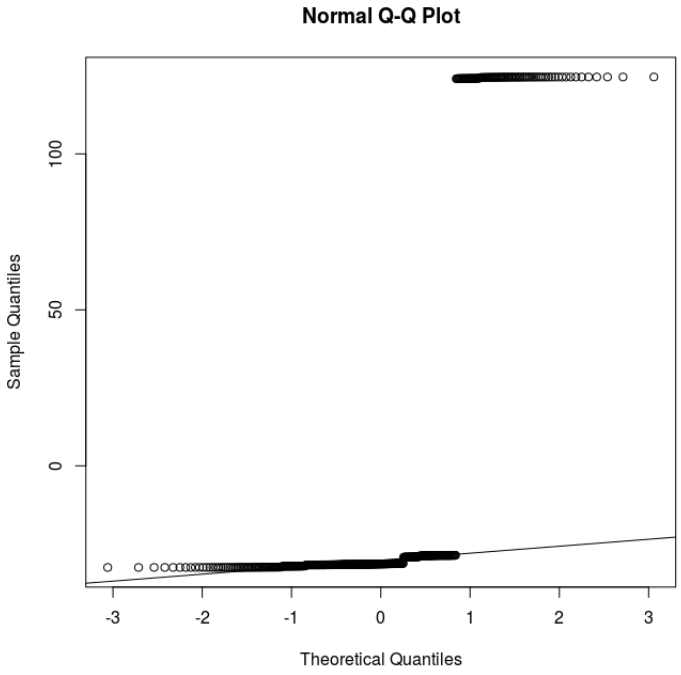


Figura 5.3.8.: Diagrama de QQ-Plot de los residuos de las muestras resultantes para el escenario pequeño fuente elaboración propia

Para confirmar que los datos no tienen una distribución normal, analíticamente se realizó el test de Shapiro-Wilk arrojando que las poblaciones no se distribuyen de forma normal. Los valores arrojados por el tes se pueden observar en el cuadro 5.12

ACO		ABC		FA	
W	p-value	W	p-value	W	p-value
0.50563	*2.2e-16*	*0.50218*	*2.2e-16*	*0.50524*	*2.2e-16*

Table 5.12.: Resultados test Shapiro-Wilk para el escenario grande

6. Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones del presente trabajo, como también las plantean posibles mejoras al presente estudio o enfoques que surgieron durante la elaboración del presente documento.

6.1. Conclusiones

La promesa de pagar únicamente por los recursos empleados ha impulsado la computación en la nube, sobre todo en empresas pequeñas al facilitar ofrecer servicios sin incurrir en costos de infraestructura. En el caso de empresas grandes permite optimizar la inversión en infraestructura dado que no se incurre en temas de depreciación tecnológica, administración y/o fallos,con la posibilidad de un escalamiento dinámico.

El balanceo de carga toma una alta relevancia en un entorno de computación la nube, debido principalmente a la naturaleza dinámica de las tareas, del tráfico y de la capacidad de re-configurase de la nube, aumentando o disminuyendo elementos de computo sin intervención humana.

En el proceso de recolección de la información del estado de los elementos del sistema distribuido, se debe ser muy cauteloso, no recolectar la suficiente información o no tener información reciente puede generar malas decisiones en la localización de los recursos, pero realizar esta actividad muy seguido puede saturar los canales de comunicación.

La política de localización de recursos y migración de tareas, permiten mejorar la distribución de carga en el conjunto de nodos, pero por si sola no soluciona el problema de balanceo de carga.

A medida que aumenta el número de recursos disponibles las diferencias entre las técnicas seleccionadas se reducen, y mucho mas equivalentes si el se trata de un entorno con re-configuración dinámica.

El algoritmo con mejor desempeño fue el de Abejas Artificiales, dado que en la simulaciones siempre fue el que presento menor tiempo de respuesta. La técnica con menor desempeño fue la de la colonia de hormigas.

La fortaleza de las técnicas esta basada en la en las decisiones basadas en la información del sistema. En el caso de las abejas el hecho de explotar al máximo los recursos

disponibles antes de seleccionar una nueva fuente facilita la identificación de las maquinas con sobrecarga, pero a la vez en el caso de tener un número mayor de recursos al de las hormigas trabajadoras puede dejar buenas soluciones sin ser explotadas.

Las tres técnicas presentan unas buenas soluciones al problema de balanceo de carga sobre todo en entornos dinámicos, pero a un mayor número de operaciones para la asignación de los trabajos, lo que implica un mayor procesamiento.

La herramienta empleada para ejecutar los experimentos, permite a los interesados en implementar un proyecto de computación en la nube planear la capacidad requerida según sus necesidades, evaluar los costos y dimensionar la capacidad requerida de instalación.

La técnica de las Luciérnagas fue la que mas iteraciones necesito para encontrar el punto estacionario, pero fue la única que se construyo con un modelo de segundo orden.

6.2. Trabajo Futuro

En este estudio se planeo minimizar el tiempo de respuesta, pero se plantea involucrar variables que permitan minimizar el costo de utilización, el ancho de banda, el costo por migración de procesos, la tasa de solicitudes rechazadas, y reducir la migración de maquinas virtuales, con las versiones multi-objetivo de las técnicas expuestas.

Se propone explorar los algoritmos descentralizados como el propuesto por [Angonese, 2012] para evaluar su comportamiento y compararlo con enfoques inspirados en teoría de juegos. para explorar problemas como el del nodo egoísta en un ambiente de computación en la nube.

Se propone explorar algoritmos híbridos y modificaciones de estas como el algoritmo *Adaptative FireFly*, *Fast Aco* con el propósito de mejorar los tiempos de respuesta.

Implementar una política de migración de tareas que permita mejorar el balanceo de carga, que complemente las técnicas seleccionadas.

Implementar un enfoque para la re-configuración dinámica empleando el cambio de roles de la técnica de abejas artificiales.

Implementar técnicas en la política de migración que complemente la política de localización.

A. Anexos

A.1. *Cloud Analyst*

Cloud Analys es un derivado de CloudSim, herramienta desarrollada por la universidad de Melbourne, escrita completamente en Java®, bajo una licencia Apache Lincence 2.0[Wickremasinghe et al., 2010][Computing and Distributed Systems (CLOUDS) Laboratory, 2009]

Uno de los objetivos es separar el ejercicio de simulación con el de programación, centrándose y facilitando la construcción de escenarios y la definición parámetros, automatizando aspectos técnicos al utilizar directamente el conjunto de herramientas proporcionadas por CloudSim.

Una característica de la herramienta, es contar con una una interfaz gráfica de usuario la cual permite configurar el o los experimentos con un alto grado de flexibilidad. La interfaz ayuda a definir y asignar parámetros de elementos como: centros de datos, maquinas virtuales, dispositivos de almacenamiento, memoria, ancho de banda, latencias de red, políticas de asignación de recursos, usuarios y su distribución, políticas de negociación, costos estimados por la utilización de la plataforma entre otras dinámicas que pueden estar presentes en la prestación de servicios por internet. El tiempo a simular en los experimentos puede ser proporcionado en minutos, horas o días[Wickremasinghe et al., 2010].

Con el propósito que los experimentos puedan ser repetidos, la herramienta permite almacenar los parámetros definidos en un formato basado en xml, de igual forma los resultados pueden ser almacenados en formato pdf . Los resultados son resúmenes de las estadísticas recolectadas expresados en gráficas y tablas con el objetivo de poder identificar los patrones importantes.

Con respecto a la distribución geográfica la versión utilizada permite definir un máximo de cinco regiones diferentes con sus respectivos parámetros asignados.

Bibliografía

[17788, 2014] 17788 (2014). Information technology – Cloud computing – Overview and vocabulary.

[Adnan et al., 2012] Adnan, M. A., Sugihara, R., and Gupta, R. K. (2012). Energy efficient geographical load balancing via dynamic deferral of workload. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 188–195. IEEE.

[Angonese, 2012] Angonese, C. (2012). Balanceamento de carga de trabalho em computação em nuvem baseado em redes magnéticas vituais. Master's thesis, Pontifícia Universidade Católica do Paraná.

[Armbrust et al., 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

[Basha et al., 2016] Basha, S. J., Kumar, P. A., and Babu, S. G. (2016). Storage and processing speed for knowledge from enhanced cloud computing with hadoop frame work: A survey.

[Berman et al.,] Berman, F., Fox, G., and Hey, A. J. G. In Hey, T., editor, *Grid Computing: Making the Global Infrastructure a Reality*.

[Bilgaiyan et al., 2015] Bilgaiyan, S., Sagnika, S., Mishra, S., and Das, M. (2015). Study of task scheduling in cloud computing environment using soft computing algorithms. *International Journal of Modern Education and Computer Science (IJMECS)*, 7(3):32.

[Bitam, 2012] Bitam, S. (2012). Bees life algorithm for job scheduling in cloud computing. In *Proceedings of The Third International Conference on Communications and Information Technology*, pages 186–191.

[Buyya, 1999] Buyya, R. (1999). *High Performance Cluster Computing: Programming and Applications*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[Buyya et al., 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging {IT} platforms: Vision, hype, and reality

for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616.

[Cepillo, 2011] Cepillo, D. I. S. (2011). Diseño óptimo de laminados en materiales compuestos. aplicación del mef y el método de las superficies de respuesta. Master's thesis, Escuela Superior de Ingenieros - Universidad de Sevilla.

[Chen et al., 2013] Chen, H., Xiong, L., and Wang, C. (2013). Cloud task scheduling simulation via improved ant colony optimization algorithm. *Journal of Convergence Information Technology*, 8(7).

[Computing and Distributed Systems (CLOUDS) Laboratory, 2009] Computing, T. C. and Distributed Systems (CLOUDS) Laboratory, U. o. M. (2009). CloudSim a framework for modeling and simulation of cloud computing infrastructures and services.

[Cybenko, 1989] Cybenko, G. (1989). Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279 – 301.

[Engelbrecht, 2007] Engelbrecht, A. (2007). *Computational Intelligence An introduction*. addisosn Wessley, 2nd edition.

[Fang Liu and Leaf, 2011] Fang Liu, Jin Tong, J. M. R. B. J. M. L. B. and Leaf, D. (2011). Sp 500-292. nist cloud computing reference architecture. Technical report, Gaithersburg, MD, United States.

[Florence and Shanthi, 2014] Florence, A. P. and Shanthi, V. (2014). A load balancing model using firefly algorithm in cloud computing. *Journal of Computer Science*, 10(7):1156.

[Foster and Kesselman, 2003] Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. The Elsevier Series in Grid Computing. Elsevier Science.

[G. Coulouris, 2001] G. Coulouris, J. Deollimore, T. K. (2001). *Sistemas Distribuidos Conceptos Y Diseño*. addisosn Wessley, 3nd edition.

[G. Tanenmbaum, 2001] G. Tanenmbaum, J. Deollimore, T. K. (2001). *Sistemas Distribuidos Conceptos Y Diseño*. addisosn Wessley, 3nd edition.

[Hussein and Khalid, 2016] Hussein, N. H. and Khalid, A. (2016). A survey of cloud computing security challenges and solutions. *International Journal of Computer Science and Information Security*, 14(1):52.

[Jing et al., 2013] Jing, S.-Y., Ali, S., She, K., and Zhong, Y. (2013). State-of-the-art research study for green cloud computing. *The Journal of Supercomputing*, 65(1):445–468.

[Joshi, 2012] Joshi, N. A. (2012). *IDevelopment of Algorithms for Optimized Process Migration for Load Balancing in Distributed Systems.* PhD thesis, SardarPatelUniversity- VallabhVidyanagar Gujarat.

[Kameda et al., 1997] Kameda, H., Li, J., Kim, C., and Zhang, Y. (1997). *A Comparison of Static and Dynamic Load Balancing*, pages 225–240. Springer London, London.

[Kansal et al., 2010] Kansal, A., Zhao, F., Liu, J., Kothari, N., and Bhattacharya, A. A. (2010). Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 39–50. ACM.

[Karaboga and Basturk, 2007] Karaboga, D. and Basturk, B. (2007). Artificial bee colony (abc) optimization algorithm for solving constrained optimization. In *Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, Springer-Verlag, IFSA(2007*, pages 789–798.

[Katyal and Mishra, 2014] Katyal, M. and Mishra, A. (2014). A comparative study of load balancing algorithms in cloud computing environment. *arXiv preprint arXiv:1403.6918*.

[Kaur and Rai, 2014] Kaur, K. and Rai, A. K. (2014). A comparative analysis: Grid, cluster and cloud computing. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(3):5730–5734.

[Kumar and Shobana,] Kumar, P. and Shobana, P. A survey on workflow task scheduling using intelligent water droplets in cloud computing.

[Liu et al., 2016] Liu, Y., Shu, W., and Zhang, C. (2016). A parallel task scheduling optimization algorithm based on clonal operator in green cloud computing. *Journal of Communications*, 11(2).

[Maqueira and Bruqué, 2011] Maqueira, J. M. and Bruqué, S. (2011). Las tecnologías grid de la información como nueva herramienta empresarial: definición, taxonomía y niveles de adopción. *Revista de Economía Industrial*, (380):153 – 162.

[Marinescu, 2013] Marinescu, D. C. (2013). *Cloud computing: theory and practice.* Morgan Kaufmann.

[Mell and Grance, 2011] Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States.

[Mesa, 2009] Mesa, A. (2009). Método para el manejo del balanceo de carga en sistemas de cómputo de alto desempeño. Master's thesis, Universidad Nacional de Colombia- Medellín.

[Mistry, 2013] Mistry, B. (2013). Dynamic load balancing in mass. Master's thesis, University of Washington.

[Nipane and Dhande, 2014] Nipane, M. N. S. and Dhande, N. M. (2014). Abc-load balancing technique-in cloud computing. *International Journal of Innovative Research in Advanced Engineering*.

[Oscar Orlanos Melo, 2007] Oscar Orlanos Melo, Luis Alberto López, S. E. M. (2007). *Diseño de Experimentos*.

[Patel et al., 2016] Patel, N., Sahu, D., Patel, P., and Singh, P. K. (2016). A survey on introduction of green computing. *International Journal of Research*, 3(5):390–400.

[Raghava and Singh, 2014] Raghava, N. and Singh, D. (2014). Comparative study on load balancing techniques in cloud computing. *Open Journal of Mobile Computing and Cloud Computing*, 1(1).

[Ray and De Sarkar, 2012] Ray, S. and De Sarkar, A. (2012). Execution analysis of load balancing algorithms in cloud computing environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 2(5):1–13.

[Sethi et al., 2012] Sethi, S., Sahu, A., and Jena, S. K. (2012). Efficient load balancing in cloud computing using fuzzy logic. *IOSR Journal of Engineering*, 2(7):65–71.

[Shu et al., 2014] Shu, W., Wang, W., and Wang, Y. (2014). A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–9.

[Song et al., 2014] Song, S., Lv, T., and Chen, X. (2014). Load balancing for future internet: an approach based on game theory. *Journal of Applied Mathematics*, 2014.

[Sosinsky, 2011] Sosinsky, B. (2011). *Cloud Computing Bible*. Wiley Publishing, 1st edition.

[Subrata et al., 2007] Subrata, R., Zomaya, A. Y., and Landfeldt, B. (2007). Artificial life techniques for load balancing in computational grids. *Journal of Computer and System Sciences*, 73(8):1176–1190.

[Sultan, 2010] Sultan, N. (2010). Cloud computing for education: A new dawn? *International Journal of Information Management*, 30(2):109–116.

[Sun et al., 2013] Sun, H., Chen, S.-p., Jin, C., and Guo, K. (2013). Research and simulation of task scheduling algorithm in cloud computing. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 11(11):6664–6672.

[Supreeth and Biradar, 2013] Supreeth, S. and Biradar, S. (2013). Scheduling virtual machines for load balancing in cloud computing platform. *International Journal of Science and Research (IJSR), India Online ISSN*, pages 2319–7064.

[Tawfeek et al., 2013] Tawfeek, M. A., El-Sisi, A., Keshk, A. E., and Torkey, F. A. (2013). Cloud task scheduling based on ant colony optimization. In *Computer Engineering & Systems (ICCES), 2013 8th International Conference on*, pages 64–69. IEEE.

[Tejedor, 2007] Tejedor, R. J. M. (2007). Grid computing. *Manual Formativo*, (43):17 – 22.

[Urgaonkar et al., 2005] Urgaonkar, B., Shenoy, P., Chandra, A., and Goyal, P. (2005). Dynamic provisioning of multi-tier internet applications. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 217–228.

[Vázquez, 2012] Vázquez, C. (2012). *Arquitectura para el aprovisionamiento dinámico de recursos computacionales*. PhD thesis, Universidad Complutense de Madrid - Facultad de Informática.

[Vizuete, 2012] Vizuete, A. M. Z. (2012). Arquitectura e implementación de un sistema distribuido de detección de sismos para alerta temprana. Master's thesis, Universidad Politectica de Valencia- Valencia.

[Wang et al., 2008] Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., and Karl, W. (2008). Scientific cloud computing: Early definition and experience. In *HPCC*, volume 8, pages 825–830.

[Wickremasinghe et al., 2010] Wickremasinghe, B., Calheiros, R. N., and Buyya, R. (2010). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 446–452.

[Y.3500, 2014] Y.3500 (2014). Series Y: Global Information Infraestructure, Internet protocol aspects and next-generation networks Cloud Computing.

[Yang, 2013] Yang, X.-S. (2013). *Artificial Intelligence, Evolutionary Computing and Metaheuristics: In the Footsteps of Alan Turing*, chapter Metaheuristic Optimization: Nature-Inspired Algorithms and Applications, pages 405–420. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Younis and El Halees, 2015] Younis, H. J. and El Halees, A. (2015). Efficient load balancing algorithm in cloud computing. Master's thesis, Islamic University - Gaza.

[Zhang et al., 2010] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.