Migración de aplicaciones Monolíticas a entornos distribuidos Serverless

Nelson Rodríguez[1], María Murazzo[1], Diego Medel[1], Daniel Arias Figueroa[2], Lorena Parra[1], Ana Laura Molina[1], Adriana Martin[1], Hernán Atencio[3], Martín Gómez[3] Guillermo Casasola[3]

[1] Departamento e Instituto de Informática - F.C.E.F. y N. - U.N.S.J.

[2] Consejo de Investigación, FCE, Universidad Nacional De Salta

[3] Alumno Avanzado Licenciatura en Sistemas de Información y Cs. de la Computación - F.C.E.F. y N. - U.N.S.J.

Complejo Islas Malvinas. Cereceto y Meglioli. 5400. Rivadavia. San Juan, 0264 4234129

nelson@iinfo.unsj.edu.ar, marite@unsj-cuim.edu.ar, vdiego.unsj@hotmail.com, daaf@cidia.unsa.edu.ar, lorenaparra152@yahoo.com.ar, almm95@gmail.com, fgsanchez@unsj-cuim.edu.ar, adrianamartin1@gmail.com, hernan.atencio.98@gmail.com, martinsj0811@gmail.com, guillecasasola@gmail.com

Resumen

Serverless Computing es una arquitectura o modelo de ejecución en el cloud, alternativo al modelo tradicional. Ofrece numerosas ventajas sobre una arquitectura monolítica, como aportar agilidad, innovación, un mejor escalado automático, flexibilidad en el desarrollo y una mejor evaluación y control de los costos. Surgió como una evolución de microservicios corriendo en contenedores e implementando funciones, por lo cual a veces se lo denomina función como servicio. Las ventajas que presenta este modelo de computación, promueven a que sea conveniente migrar aplicaciones montadas sobre el cloud tradicional a una arquitectura serverless. Estas aplicaciones se deben llevar a una arquitectura de microservicios, donde se ejecutan funciones, cada una de las cuales de forma independiente y conducida por eventos. Si bien existen varias estrategias y propuestas metodológicas para llevar a cabo la migración, surgen numerosos desafíos y problemas a resolver, debido a que el desarrollo es completamente diferente, además se deben aplicar técnicas de observación y monitorear el progreso de la migración , por lo tanto toda la problemática expresada es el motivo de la presente línea de investigación.

Palabras clave: *Serverless Computing, Distributed Computing, Software Architecture, Cloud Computing*

Contexto

El presente trabajo se encuadra dentro del área de I/D Procesamiento Distribuido y Paralelo y es una de las líneas de investigación internas, del proyecto: Computación Serverless para el tratamiento de datos provenientes de dispositivos de IoT, cuya propuesta ha sido aprobada y está en desarrollo para el período 2020-2021, y se ha extendido un año más. Asimismo el grupo de investigadores viene trabajando en proyectos relacionados con la computación distribuida y de alta performance desde hace más de 21 años. Como continuación del proyecto anterior: Orquestación de Servicios para la Continuidad Edge al Cloud, se continúa el trabajo con investigadores de otras universidades, lo cual favorece notablemente a todos las instituciones participantes.

Introducción

Cloud Computing es una arquitectura o modelo de provisión de servicios bastante adoptado y aceptado, sin embargo administrar

estos servicios no es una tarea fácil en absoluto. Los autores de [1] han abordado varios desafíos al administrar un entorno de nube por parte de un usuario, como la disponibilidad, el equilibrio de carga, el escalado automático, la seguridad, la supervisión, etc.

Estos desafíos han llevado a introducir otro modelo informático en el Cloud, que se denomina computación Serverless.

Tiene como objetivo fundamental aliviar varios problemas que tienen los desarrolladores en el Cloud tradicional, permitiéndoles concentrarse solo en la funcionalidad principal de su producto [2].

La computación Serverless es una tecnología con un impacto creciente en nuestra sociedad y una mayor adopción tanto por parte de la academia como de la industria [3]. Es un paradigma en el que las aplicaciones de software se descomponen en múltiples funciones independientes sin estado [4] [5]. Las funciones se ejecutan en contenedores en respuesta a acciones desencadenantes (como interacciones de usuario, eventos de mensajería o cambios en la base de datos), se pueden escalar de forma independiente y pueden ser efímeras.

En este enfoque, casi todas las preocupaciones operativas son abstraídas lejos de los desarrolladores. Los cuales en principio simplemente escriben código e implementan sus funciones en una plataforma sin servidor. La plataforma se encarga de la ejecución de la función, el almacenamiento y la infraestructura de contenedor, redes y tolerancia a fallas. Adicionalmente, también se encarga de escalar las funciones según la demanda real.

En la mayoría de los casos, se puede escribir funciones en el lenguaje favorito del programador (Node.js, Python, Go, Java y más) y utilizar herramientas de contenedor y serverlessr, como AWS SAM o la CLI de Docker, para compilar, probar e implementar las funciones.

La computación serverless ha sido identificada como un enfoque prometedor para varias aplicaciones, como el análisis de datos en el edge [6]. En consecuencia, una plataforma maneja el ciclo de vida, la ejecución y escalada de las funciones reales; estas necesitan correr solo cuando son invocadas o activadas. Por lo tanto, el mayor beneficio son las pocas preocupaciones operativas y de gestión y la utilización eficiente de los recursos [7].

Un modelo basado en funciones es particularmente adecuado para ráfagas, uso de CPU intensivo, cargas de trabajo granulares. Actualmente, los casos de uso de FaaS varían ampliamente, incluido el procesamiento de datos, el procesamiento de flujo, la computación de borde (IoT) y la computación científica [8] [9], y es probable que otros casos de uso surjan en un futuro.

El término "sin servidor" es confuso ya que hay hardware y procesos de servidor que se ejecutan, pero la diferencia en comparación con los enfoques tradicionales es que la organización que construye y admite una aplicación "sin servidor" no se ocupa de ese hardware o esos procesos, debido a que están subcontratando esta responsabilidad.

El término comenzó a utilizarse enn 2012 [10]. El término se hizo más popular en 2015, luego del lanzamiento de AWS Lambda,

Existen varias definiciones, se ha tomado como referencia la elaborada por el grupo de investigación The SPEC Cloud, que la describe de la siguiente manera:

La computación serverless es una forma de computación en el cloud que permite a los usuarios ejecutar eventos y aplicaciones facturadas de forma granular, sin tener que abordar la lógica operativa [11].

Los desarrolladores se centran en abstracciones de alto nivel (por ejemplo, funciones, consultas y eventos) y en crear aplicaciones que se asignan a recursos concretos y servicios de soporte. Esto permite que los desarrolladores se enfoquen en la lógica empresarial y en las formas de interconectar elementos de la lógica empresarial en flujos de trabajo complejos. Mientras tanto, los proveedores de servicios se aseguran de que las aplicaciones están alojadas en contenedores, desplegadas, aprovisionadas y disponibles bajo demanda, mientras se factura al usuario solo por los recursos utilizados [12].

La computación serverless se puede identificar como resultado de la unión de Cloud y Microservices Architecture. Pero la evolución a Serverless ha pasado por varias etapas y está en permanente crecimiento.

Migración de aplicaciones Monolíticas a Serverless Computing

El término arquitectura monolítica se deriva de la era histórica cuando los edificios antiguos fueron tallados, fundidos o excavados a partir de una sola pieza de material. Por ejemplo, las iglesias monolíticas de Etiopía y Pancha Rathas en la India fueron talladas en una sola pieza de roca. La mayor desventaja era que si se realizaban pequeños trabajos de reparación afectaba a toda la arquitectura.

Del mismo modo, en el contexto de desarrollo de software, los componentes de la arquitectura monolítica están fuertemente acoplados. Por lo tanto, si se planea escalar, eliminar errores o realizar ligeros cambios en cualquier componente, toda la aplicación se verá afectada. Eso significa que, en tales casos, se debe reescribir la aplicación desde cero.

Serverless presenta muchas ventajas sobre la arquitectura monolítica, como son: recuperación más rápida de fallos, reducción del mantenimiento del servidor, reutilización de código y libertad para codificar en diferentes lenguajes. Por lo tanto es conveniente realizar la migración a este entorno.

La migración seguramente implicará una vasta reescritura del código existente, a diferencia de, por ejemplo, si está migrando una aplicación tradicional basada en servidor para que se ejecute dentro de contenedores donde los cambios generalmente se limitan al nivel de infraestructura. Se debe realizar un análisis de costo-beneficio antes de continuar.

Otro aspecto muy importante es la migración de la base de datos, por ejemplo, si se tiene la aplicación en MongoDB, surgen interrogantes sobre cómo realizar la migración, por ejemplo a DynamoDB (si es usa AWS Lambda) y todo lo que ello implica: sincronización de datos, autenticación, tiempo de transición en que ambos están en uso, etc.

En el inicio de la migración, se debe desacoplar el sistema monolítico, dividiéndolo en una serie de servicios bien definidos y poco acoplados [13].

Serverless, son aplicaciones sin estado, sin embargo las aplicaciones monolíticas, por lo general tienen estado. En este caso se deben implementar mecanismos para proveer de estado a la aplicación.

El arranque en frío, es una característica de Srverless, dado que la función a ejecutar debe cargarse en el contenedor y esto lleva una demora que puede afectar la performance, en el caso de que exista algún requerimiento temporal, debe ser tenido en cuenta.

El diseño de aplicaciones distribuidas (en este caso rediseño), requiere que los programadores tomen en cuenta la totalidad de los aspectos operativos incluyendo confiabilidad, mantenimiento y sobreprovisión de recursos, estos deben ser conocidos en profundidad.

Al ser código heredado de una aplicación monolítica, se debe considerar si se va a realizar esta migración de forma gradual, o se va a implementar todo el sistema. En el caso de una migración gradual: Los componentes de la aplicación se reemplazan por versiones serverless de ellos, una a la vez, mientras los usuarios utilizan esta versión híbrida.

Un plan de migración está determinado por:

El estado actual de su organización, el estado actual de la aplicación y el estado deseado.

Existen tres estrategias generales de migración para crear una aplicación sin servidor:

- Leapfrog
- Organic
- Strangler

La estrategia Leapfrog se salta las fases intermedias y se mueve directamente a una arquitectura de Cloud Serverless.

Con la estrategia Orgánica, se "trasladan" los programas locales al Cloud. Las aplicaciones actuales se mantienen ejecutándose en instancias Amazon EC2 o Amazon ECS (por ejemplo si la migración fuera en AWS).

En la estrategia s**trangler, se** descompone los programas monolíticos mediante el establecimiento de API y componentes basados en eventos. Combina la interfaz de usuario y el código de acceso a los datos y

reemplazan lentamente los componentes heredados.Permite un desarrollo más rápido de nuevas características con menos riesgo que la estrategia Leapfrog. Strangler es la estrategia más común [14].

Líneas de Investigación, Desarrollo e Innovación

La investigación sobre esta línea de trabajo ha comenzado hace un año aproximadamente a partir de un análisis documental sobre revisiones sistemáticas [15], surveys [16] [17] [18] y otros trabajos, que permitió profundizar sobre el estado del arte en la migración de aplicaciones monolíticas a Serverless. Esto posibilitó encontrar cuales son los problemas científicos, consideraciones y desafíos que se van a tratar de solucionar.

Posteriormente la investigación se conducirá de forma hipotética mixta (experimental y deductiva). Esta forma de trabajo permitirá analizar la estrategia en diferentes tipos de migraciones según el tipo de aplicación, enfrentar problemas y desafíos y construir un conjunto de buenas prácticas.. El método a utilizar será empírico analítico, pero puede variar en función de cada problema en particular.

Resultados y Objetivos

Resultados Obtenidos

Durante los últimos catorce años se trabajó en el área de Computación de Altas Prestaciones y distribuidas, en particular sobre análisis de diversas arquitecturas paralelas y distribuidas, tales como: Cloud Computing, Cluster de commodity, arquitecturas distribuidas y paralelas de bajo costo y fog computing. El proyecto marco de esta línea de investigación, se inició hace dos años, el mismo tiene como temática principal a Srverless Computing. A partir de distintos análisis y debates en el grupo, se inició la línea de investigación del presente trabajo. El grupo ha realizado varias publicaciones en esta área: ocho trabajos de investigación en Congresos y Jornadas, se realizaron tres publicaciones en revistas científicas y se transfirieron los resultados mediante conferencias en eventos científicos.

Se han aprobado dos tesinas de grado, se incorporó un becario de investigación categoría alumno y otra beca está en evaluación.

Objetivos

Los objetivos del grupo de investigación en esta línea de conocimiento son los siguientes:

-Realizar aportes desde la academia a la temática de la presente propuesta, debido a que existen pocos trabajos científicos publicados. Por otro lado, si bien la industria ha realizado su aporte (aunque son también pocas publicaciones) son menos rigurosos en sus conclusiones, procedimientos y métodos.

-Desarrollar un conjunto coherente de buenas prácticas, que permitan ofrecer una guía o soporte metodológico para que el equipo de desarrollo lleve adelante sus tareas.

-Profundizar en el análisis y tratamiento de problemas abiertos o aún no resueltos, para lograr una efectiva migración.

Formación de Recursos Humanos

El equipo de trabajo de esta línea de investigación está compuesto de ocho investigadores que figuran en este trabajo de las universidades Nacional de San Juan y Nacional de Salta y dos alumnos de grado. Además, el proyecto marco donde se está desarrollando esta propuesta incluye a tres investigadores más de la Nacional de San Luis, de la Universidad Champagnat y de la Universidad Nacional de San Juan y tres alumnos de grado.

Se está desarrollando una tesis doctoral sobre paralelismo híbrido y Big Data, una tesis de maestría en áreas afines y seis tesinas de grado en el área de Serverless computing, Concurrencia y Computación distribuida, en particular dos sobre migración de aplicaciones monolíticas. Además se espera aumentar el número de publicaciones. Por otro lado también se prevé la divulgación de varios temas investigados por medio de cursos de postgrado y actualización o publicaciones de

divulgación y asesoramiento a empresas y otros organismos del estado.

Referencias

[1] Jonas E, Schleier-Smith J, Sreekanti V, Tsai C-C, Khandelwal A, Pu Q, Shankar V, Carreira J, Krauth K, Yadwadkar N, Gonzalez JE, Popa RA, Stoica I, Patterson DA (2019) Cloud Programming Simplified: A Berkeley View on Serverless Computing. http://arxiv.org/abs/1902.03383. Accessed 6 Jan 2021

[2] S. Eismann et al., 'Serverless Applications: Why, When, and How?' (2021), IEEE Software, vol. 38, no. 1, pp. 32–39, Jan. 2021, doi: 10.1109/MS.2020.3023302.

[3] IDC, FutureScape: Worldwide IT Industry 2019 Predictions," https://www.idc. com/getdoc.jsp?containerId=US44403818, (2018).

[4] Adzic, G., & Chatley, R. (2017). Serverless computing: economic and architectural impact. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (pp. 884-889). ACM.

[5] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V. & Suter, P. (2017). Serverless computing: Current trends and open problems. In Research Advances in Cloud Computing (pp. 1-20). Springer, Singapore.

[6] Nastic, S., Rausch, T., Scekic, O., Dustdar, S., Gusev, M., Koteska, B. & Prodan, R. (2017). A serverless real-time data analytics platform for edge computing. IEEE Internet Computing, 21(4), 64-71.

[7] Mohanty, S. K., Premsankar, G., & Di Francesco, M. (2018). An Evaluation of Open Source Serverless Computing Frameworks. In CloudCom (pp. 115-120).

[8] Gottlieb, N. (2016). State of the Serverless Community Survey Results. https: //serverless.com/blog/state-of-serverless-community/.

[9] Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., & Recht, B. (2017). Occupy the cloud: Distributed computing for the 99%. In Proceedings of the 2017 Symposium on Cloud Computing (pp. 445-451). ACM.

[10] Fromm, K. (2012). https://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless/

[11] Van Eyk, E., Iosup, A., Seif, S., & Thömmes, M. (2017). The SPEC cloud group's research vision on FaaS and serverless architectures. In Proceedings of the 2nd International Workshop on Serverless Computing (pp. 1-4). ACM.

[12] Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uță, A., Iosup, A. (2018). Serverless is more: From paas to present cloud computing. IEEE Internet Computing, 22(5), 8-17.

[13] Alireza Goli, Omid Hajihassani, Hamzeh Khazaei*, Omid Ardakanian, Moe Rashidi and Tyler Dauphinee. Migrating from Monolithic to Serverless: A FinTech Case Study. In ACM/SPEC International Conference on Performance Engineering Companion (ICPE '20 Companion), April 20–24, 2020, Edmonton, AB, Canada. ACM, New York, NY, USA, 6 pages.

[14] Mustafa Osama. Migrating to serverless. Migrating to Serverless. (2021). Technology Geek (osamaoracle.com)

[15] Kjorveziroski, V.; Filiposka, S.; Trajkovik, V. IoT Serverless Computing at the Edge: A Systematic Mapping Review. Computers (2021), https://doi.org/10.3390/computers10100130

[16] Hassan et al. Survey on serverless computing Journal of Cloud Computing: Advances, Systems and Applications (2021) 10:39 https://doi.org/10.1186/s13677-021-00253-7

[17] Zijun Li, Linsong Guo, Jiagan Cheng, and Quan Chen, The Serverless Computing Survey: A Technical Primer for Design Architecture. (2022) ACM Computers Survey. https://doi.org/10.1145/3508360

[18] Zhe Li et al. A Survey of Cost Optimization in Serverless Cloud Computing (2021) Journal Phys.: Conf. Ser. 1802 032070