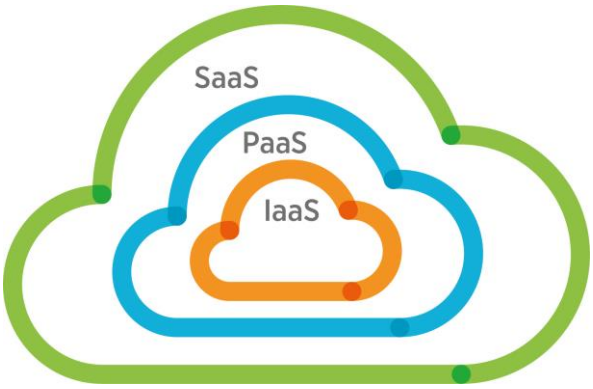
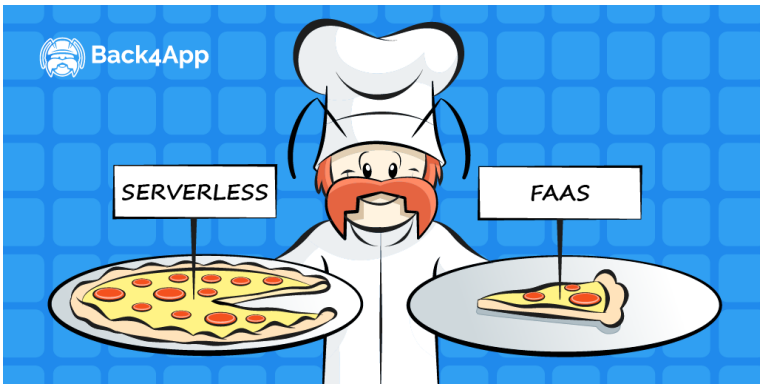
CLOUD COMPUTING





FAAS

Maestrante:

CAIZA CAIZABUANO JOSE RUBEN

ENERO 2020

Investigación preliminar
Faas

Jose Ruben Caiza Caizabuano
jrcaiza@espe.edu.ec, Universidad de las Fuerzas Armadas ESPE

El modelo de servicio de la función como servicio (FaaS) ha ido ganando aceptación a un ritmo rápido. En este modelo de servicio, las aplicaciones en la nube se estructuran como módulos de código autocontenidos llamados funciones que se instancian bajo demanda, y la facturación se basa en el número de invocaciones de la función y en el tiempo de ejecución de la misma. Los desarrolladores se sienten atraídos por FaaS porque promete eliminar dos inconvenientes de los modelos de servicio tradicionales de IaaS y PaaS, la necesidad de proveer y administrar la infraestructura y la necesidad de pagar por los recursos no utilizados. En la práctica, sin embargo, las cosas son un poco menos halagadoras: los desarrolladores todavía tienen que elegir la cantidad de memoria asignada a las funciones, y los costes son menos predecibles, especialmente porque están ligados al rendimiento de las funciones.

Una tendencia creciente en el cloud computing ha sido la adopción del modelo de servicio Function-as- a-Service (FaaS), en el que las aplicaciones se construyen como conjuntos de funciones que se ejecutan en instancias (contenedores o máquinas virtuales) activadas bajo demanda en respuesta a eventos . Estas instancias son apátridas y temporales, posiblemente permanecen activas durante la ejecución de una sola invocación, y son completamente gestionadas por el proveedor, que es responsable de asegurar su escalabilidad y disponibilidad. El cliente se cobra por cada invitación, sin tener que pagar por los recursos ociosos. Por lo tanto, el modelo FaaS puede brindar otro significado al ahorrar costes en comparación con los modelos más tradicionales de infraestructura/plataforma-como-servicio (IaaS/PaaS), mientras que al mismo tiempo libera a los desarrolladores de tener que preocuparse por la gestión de la infraestructura . Varios estudios de casos en la literatura ya han demostrado que se puede lograr una reducción significativa de los costos: Adzic y Chatley informan de un ahorro del 66% en una migración a PaaS→FaaS y del 95% en una migración de en una migración a IaaS→FaaS, Villamizar y otros muestran que un despliegue FaaS

Un área de la computación en nube que se expande dinámicamente es la de Func- tion as a Service (FaaS). FaaS permite a los clientes desarrollar, ejecutar y gestionar las funcionalidades de las aplicaciones en la infraestructura de la nube sin la carga de construir y gestionar una infraestructura virtual. La gran mayoría de los servicios de FaaS utilizados en producción son proporcionados por operadores de nube públicos, pero un número creciente de marcos de trabajo de FaaS de código abierto, es una alternativa de despliegue de servicios de FaaS en las instalaciones. Los marcos FaaS soportan el puerto diferente de los tiempos de ejecución del lenguaje de programación. El rendimiento de estos sistemas depende de estos tiempos de ejecución del lenguaje. Nuestro objetivo es mostrar y analizar esta dependencia, y proporcionar información sobre el aspecto importante cuando el rendimiento es esencial. Este documento proporciona una evaluación basada en la medición de las capacidades de los tiempos de ejecución de los lenguajes en los marcos FaaS. Evaluamos tres cargas de trabajo diferentes (eco, de comunicaciones y de datos) en los tiempos de ejecución seleccionados.

De acuerdo con las últimas tendencias, las tecnologías de virtualización son cada vez más granulares. Las robustas máquinas virtuales (VM) han sido reemplazadas por tecnologías basadas en contenedores ligeros como Docker o LXC o micro VMs, por ejemplo Fire- cracker, que permiten ejecutar microservicios en entornos de nube más fácilmente. En los últimos años, ha empezado a surgir la función como servicio (FaaS). En el caso de FaaS, en lugar de proporcionar el entorno de tiempo de ejecución completo, el usuario sólo registra las funciones y declara cuándo deben activarse estas funciones. El precio de las FaaS en las nubes públicas es más granular, ya que a los usuarios sólo se les factura por los recursos de computación utilizados durante el tiempo de ejecución de sus funciones [26]. Las funciones que se ejecutan en un sistema FaaS pueden ser bloques de construcción ligeros de un servicio de nivel superior, que benefit de utilizar la virtualización basada en contenedores o micro VMs. Las instancias de las funciones se crean dinámicamente tras la invocación de la función y están disponibles durante algún tiempo, defined por parte del proveedor de servicios en la nube. Después de ese tiempo, son desalojados y los recursos asociados liberado hasta la próxima invocación. FaaS es una tecnología emergente; varios proveedores de servicios en la nube, como Amazon Web Services, Google Cloud y Microsoft Azure, tienen soluciones FaaS. Por otro lado, FaaS ha sido adoptado por la comunidad de código abierto; numerosos entornos FaaS están disponibles en GitHub, como

OpenFaaS, Kubeless, Nuclio o Fission. Según las estadísticas de GitHub, la iniciativa FaaS de código abierto que se mantiene con mayor frecuencia es Open- FaaS. El comportamiento de los entornos de tiempo de ejecución de funciones seleccionadas. Seleccionamos y examinamos los tiempos de ejecución, que son comunes en el código abierto y también en los proveedores públicos de FaaS, como Amazon Web Services (AWS), Microsoft Azure y Google Cloud. Investigue los tiempos de ejecución basados en Python(v2.7), Node.js, y Go.

La investigación se basa en los tiempos de ejecución en cada una de las plataformas FaaS, ya que cuando se llama a una función, al final de la cadena es el runtime el que ejecuta el código de la función. Por lo tanto, el rendimiento del tiempo de ejecución es crucial para el rendimiento general de la FaaS.

El objetivo es mostrar las limitaciones, y los mecanismos que influye en el rendimiento y la escalabilidad de cada tiempo de ejecución, y no la comparación del rendimiento de los tiempos de ejecución. Las tres funcionalidades son: una función que implementa un servicio de eco, una función de cálculo intensivo y una función de datos intensivos. Los dos primeros no utilizan servicios externos, mientras que en la función de datos intensivos, el código de función interactúa con una base de datos externa. Este tipo de funciones se encuentran en las aplicaciones web modernas que generalmente se ejecutan en entornos de nube.

El trabajo se basa casi exclusivamente en la base de código de los tiempos de ejecución de las funciones de OpenFaaS, con sólo una sutil modificación . OpenFaaSes un framework desarrollado en Go para construir funciones sin servidor con Dockery Kubernetes. Basamos nuestro trabajo en OpenFaaS porque es una plataforma muy popular según las estrellas de GitHub. La arquitectura de OpenFaaS es relativamente simple, y en muchos aspectos, es similar a las otras plataformas FaaS de código abierto, por ejemplo, ha sido escrita en Go, y como la mayoría de las soluciones, utiliza contenedores Docker encima de Kubernetes.

FaaS Arquitectura

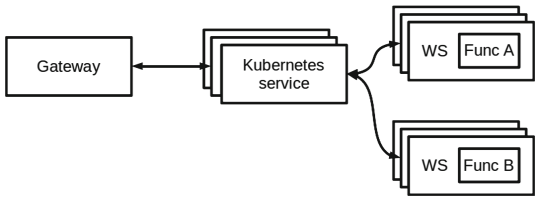
Los entornos FaaS se implementan sobre una arquitectura distribuida. Los servicios que se ejecutan en estas arquitecturas utilizan en su mayoría entornos virtualizados ligeros como contenedores o micro VMs.

Los sistemas FaaS de código abierto se implementan casi exclusivamente sobre los Kubernetes. Kubernetes implementa una capa de orquestación para poder gestionar los contenedores Docker de forma unified. Las instancias de funciones se ejecutan en pods de Kubernetes, encapsulados en contenedores Docker. Para operaciones fiables, Kubernetes puede configurarse como un clúster de alta disponibilidad.

El acceso a las funciones requiere una interfaz común, que en el caso de las soluciones FaaS de código abierto se implementa en la mayoría de los casos mediante un servidor HTTP, que convierte las peticiones para que sean procesadas por las funciones y mantiene las tareas de control de salud. En el caso de los sistemas de código abierto, las funciones de usuario se insertan en una envoltura por función soportada por el sistema FaaS. El acceso a las funciones se realiza a través de HTTP, por lo que los empaquetadores de funciones implementan un servidor web ligero que reenvía las consultas a la función del usuario. Por lo tanto, el servidor web juega un papel muy importante en la cadena de ejecución de funciones. Por lo tanto, centramos nuestra evaluación en las implementaciones de servidores web de los tiempos de ejecución del lenguaje diferente en lo siguiente.

Además de las instancias funcionales, el otro componente fundamental de una infraestructura FaaS es el Gateway, que garantiza un punto de acceso a las consultas de los usuarios del sistema y de los proxies a las instancias funcionales. La pasarela se implementa normalmente como un proxy frontal, que enruta las peticiones a las funciones apropiadas. La comunicación entre la pasarela y las funciones se implementa a través de una instancia de servicio de Kubernetes que oculta las vainas de la función detrás de una dirección IP común y equilibra la petición entre las réplicas de las instancias de la función, tal y como se muestra en la Fig. 1.

Fig. 1. Arquitectura general de FaaS



Nodo.js

Originalmente Node.js soporta la implementación de servidores web de un solo hilo.

Las implementaciones del servidor web Single threaded Node.js muestran un comportamiento de cola similar al de la Flasa Python de un solo hilo. Por otro lado, la plataforma soporta la ejecución de hilos de trabajo desde la versión v10.5.0, permitiendo que cada petición entrante sea servida por un trabajador dedicado. Sin embargo, en el caso de utilizar la nueva implementación basada en hilos de trabajo, se observar un comportamiento similar al de la implementación basada en hilos de Python Flask. La inialización de cada hilo de trabajo requiere abrir un archivo en la que se implemente la tarea, además se inicia un nuevo bucle de eventos,un motor JavaScript y una instancia Node.js para el nuevo hilo. La plataforma también soporta la creación de procesos hijos mediante el uso de la biblioteca de clusters. En este caso, cada uno de los nodos tiene un intérprete individual, porlo que no se puede producir ninguna condición de raza.

Similitudes entre Python y Node.js

Hay similitudes entre Python y Node.js. Ambos idiomas son interpretados, por lo que podemos ver patrones de comportamiento similares. La librería de hilos en Python puede corresponderse con los nuevos hilos de trabajo en Node.js; por otro lado, la implementación de multiprocesamiento de Flask es similar a la solución basada en cluster en Node.js.

Discusión

Los tiempos de ejecución de las funciones se encuentran al final de la cadena de llamadas y son responsables de la ejecución de las funciones del usuario. Dado que estas cadenas de llamadas en los sistemas FaaS de código abierto se implementan a través de HTTP, los tiempos de ejecución implementan un servidor HTTP. Investigamos las características de rendimiento de estos tiempos de ejecución en caso de un eco, la funcionalidad de cálculo intensivo y de datos intensivos, utilizando las opciones de servidor HTTP Python,Node.js y Go.

En las plataformas de código abierto FaaS los parámetros de tiempo de ejecución,por ejemplo, levels de concurrencia, manejo de hilos, etc. pueden ser ajustados para el rendimiento, la invetigacion demuestra que las funciones Go superaron todas las variantes en el caso de los valores de latencia mediana. Python mostró la más pobre performance en caso de funciones de cálculo intensivo y de eco. Aunque las funciones Go mostraron los mejores valores de latencia media, los valores de latencia de cola larga de las funciones intensivas de datos basadas en Go mostraron valores atípicos espectacularmente altos. Por otro lado, deacuerdo con las mediciones realizadas con Node.js las funciones intensivas de datos mostraron menos valores atípicos que las relacionadas con Go o Python. Según las mediciones, el tiempo de ejecución de Python puede ser optimizado aún más, sin embargo, incluso el tiempo de ejecución de Python optimizado no puede superar los otros dos tiempos de ejecución.

Caso practico

SUNFISH , que significa "SecUre iNFormatIon Shaaring en nubes privadas heterogéneas federadas", comenzó como una propuesta de Investigación e Innovación presentada en la primera convocatoria Horizon 2020 dedicada a las TIC, particularmente a Infraestructura y servicios avanzados de la nube.

Actualmente, los jugadores del sector público europeo carecen de la infraestructura y la tecnología necesarias para poder integrar sus nubes informáticas. Además, las barreras legislativas a menudo dificultan el uso de soluciones tecnológicas comerciales disponibles.

El propósito del proyecto SUNFISH, y de la Plataforma desarrollada, es facilitar la formación de federaciones seguras de diversas implementaciones en la nube del Sector Público , para poder compartir de manera segura datos y servicios . Esto hace posible el intercambio de datos entre funciones y la implementación transparente de servicios entre funciones.

Las federaciones en la nube permiten el acceso a Software como servicio (SaaS), Infraestructura como servicio (IaaS) y Plataforma como servicio (PaaS) desde diferentes nubes públicas o privadas a través de Internet, introduciendo el concepto de Federación como servicio (FaaS) en Un entorno de nube híbrida. Dicha solución FaaS se puede adaptar de manera flexible a las necesidades de las nubes y socios participantes en la federación mediante el uso de componentes de gestión de identidad existentes . Esta flexibilidad permite a las administraciones públicas y gobiernos colaborar y compartir de manera segura sus recursos de nube privada. Por lo tanto, una Administración pública puede decidir qué datos permanecen en su nube privada y qué servicios pueden migrarse a otra nube administrada por un proveedor de nube diferente.

A través de la plataforma, es posible reutilizar las soluciones existentes al integrarlas en la federación, lo que resulta en una mejor utilización de los recursos de las infraestructuras en la nube del sector público, así como en una implementación más rápida y económica de servicios públicos interoperables y escalables.

La plataforma SUNFISH, por lo tanto, se enfoca en permitir el intercambio de datos entre entidades potencialmente no confiables mientras protege los datos confidenciales de cada entidad. Esto se logra mediante varios componentes para el intercambio controlado de datos entre servicios proporcionados por diferentes nubes privadas, que se invocarán cuando el mecanismo que brinden sea el más eficiente.

La federación en la nube está diseñada para permitir la administración y la optimización de los recursos informáticos de manera segura y compatible en toda la federación en la nube para ofrecer servicios comerciales a los usuarios finales. Una infraestructura distribuida permite la asignación inteligente y dinámica de recursos de procesamiento a tareas o cargas de trabajo, en función de criterios tales como prioridades de procesos de negocios, disponibilidad de recursos, protocolos de seguridad y programación de eventos.

Con el fin de garantizar que todos los componentes de la infraestructura federada funcionen bien individualmente, e interoperen colectivamente como un todo cohesivo, los diferentes subsistemas SUNFISH se sometieron a un proceso de prueba adaptado para determinar si el rendimiento del sistema cumple con las especificaciones requeridas y los atributos de calidad. validarlos contra un conjunto de casos de uso predeterminados que abordan los desafíos específicos que enfrentan el Ministerio de Economía y Finanzas de Italia , el Ministerio de Finanzas de Malta , así como la Unidad Regional de Delitos Cibernéticos del Sudeste de Reino Unido .

Caso de uso 1 : Servicios en línea para administrar cuentas de sueldos del personal.

Caso de uso 2 : Uso de Public Cloud PaaS para alojar SaaS y / o para integrarse con soluciones comerciales de SaaS y la propia nube privada de una entidad para proporcionar datos a la entidad, mientras se garantiza la confidencialidad, integridad y disponibilidad de los datos.

Caso de uso 3 : sistema basado en la federación en la nube que permite la búsqueda y el intercambio de evidencias de delitos cibernéticos al mismo tiempo que impone requisitos de cómputo y de intercambio de datos para la gestión de datos sensibles

Referencias

Tuning Runtimes in Open Source FaaS, David Balla, Markosz Maliosz, Csaba Simon and Daniel Gehberger,2020

https://github.com/openfaas-incubator/golang-http-template

https://github.com/fission/fission

https://github.com/pallets/flask

https://github.com/rakyll/hey