COMPARACIÓN DE CÓMPUTO EN LA NUBE Y EN LA NIEBLA MEDIANTE SIMULACIÓN DE ESCENARIOS DE CONTROL DE SEMÁFOROS

A COMPARISON BETWEEN CLOUD AND FOG COMPUTING BY SIMULATION OF TRAFFIC LIGHT CONTROL SCENARIOS

Grecia Medina Rioja
Instituto Tecnológico Autónomo de México, México
gmedinar@itam.mx

José Alberto Incera Diéguez
Instituto Tecnológico Autónomo de México, México
jincera@itam.mx

Recepción: 7/noviembre/2021 **Aceptación:** 21/diciembre/2021

Resumen

Los beneficios del Cómputo en la Nube han hecho que sea la tecnología de facto para la provisión de servicios informáticos. Sin embargo, con la masificación de la Internet de las Cosas y el surgimiento de entornos inteligentes, esta tecnología muestra limitantes, como saturación (de corto plazo) de recursos de cómputo, ancho de banda o tiempos de respuesta largos. Por ello nacen nuevos paradigmas como el Cómputo en la Niebla. En este artículo se simulan arquitecturas de Cómputo en la Nube y en la Niebla para un escenario hipotético de monitoreo de servicios en una Ciudad Inteligente. Las variables de interés son capacidad de procesamiento, tiempo de respuesta y nivel de utilización de los enlaces de red. Los resultados muestran que Fog Computing tiene un enorme potencial para ofrecer servicios con mejor calidad en los entornos que caracterizarán a las Sociedades del Siglo XXI.

Palabras Clave: Cómputo en la niebla, cómputo en la nube, cómputo en la orilla, entornos inteligentes, internet de las cosas.

Abstract

Cloud Computing's proven benefits have made this the de facto technology for the provision of computing services. Nonetheless, with the massification of the so-

called Internet of Things and the emergence of smart environments, Cloud Computing architectures begin to show some limitations such as possible short term saturation of computing resources, or long response times. This has led to the emergence of new paradigms, such as Fog Computing. In this article, we make a comparison between Cloud and Fog Computing architectures for a hypothetical monitoring service scenario in a Smart City. The variables of interest are the processing capacity, the response time and the usage level of the network links. The results show that Fog Computing has enormous potential to offer services with higher quality of service in the environments that will characterize 21st Century Societies.

Keywords**:** *Cloud computing, edge computing, fog computing, internet of things, smart cities.*

1. Introducción

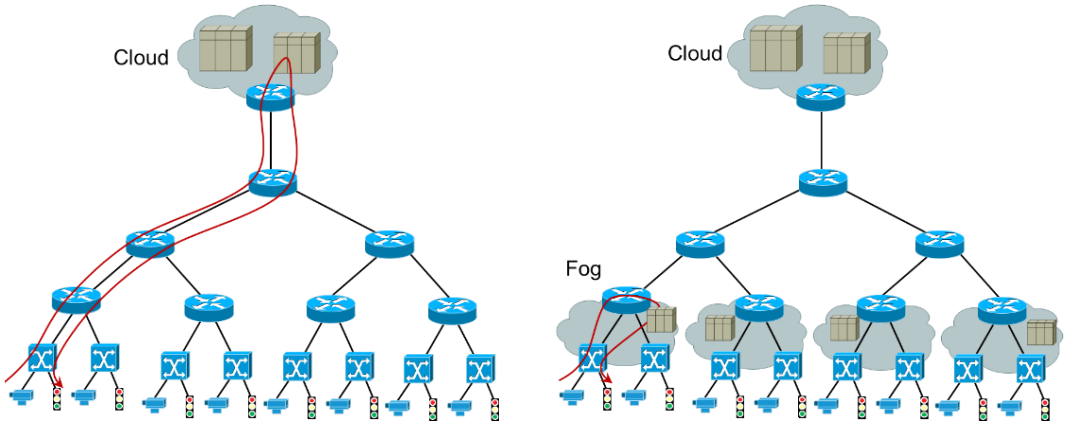
Con su capacidad de ofrecer recursos de cómputo (como aplicaciones, procesamiento y almacenamiento) bajo demanda entre otras ventajas, el Cómputo en la Nube (CC, *Cloud Computing*[1]) ha experimentado una acelerada absorción. Se estima que para 2022, el 60% de las organizaciones estará contratando algún tipo de provisión de servicios en la nube [Costello, 2019]. Sin embargo, CC tiene limitaciones. Por razones de espacio, no se discuten los muchos retos asociados a la contratación de servicios y migración de a la nube; el artículo presenta algunos de los retos técnicos que dan lugar al surgimiento del paradigma de Cómputo en la Niebla (FC, *Fog Computing*) [Bonomi, 2012].

En el modelo de Infraestructura como Servicio de CC, se arrendan servidores que se encuentran en algún lugar en la Internet. En estos servidores se procesan los datos que fueron generados desde alguna fuente y el resultado del procesamiento es eventualmente enviado a un destinatario [Gessert, 2020]. En la Internet de las Cosas (IoT, Internet of Things) las fuentes de datos pueden ser miles de dispositivos

[1] *En este documento utilizaremos los términos Cloud Computing, Edge Computing y Fog Computing en inglés en vez de sus contrapartes en español pues así es como frecuentemente se denotan estas tecnologías aún en la literatura en español.*

y el procesamiento de los datos generados por ellos, puede exceder las capacidades de CC. Si bien se CC adapta dinámicamente a las demandas de procesamiento, este ajuste no ocurre de manera instantánea y un servidor puede saturarse antes de aprovisionar otros más.

Por otra parte, el tiempo de respuesta (definido aquí como latencia) puede resultar prohibitivo en aplicaciones sensibles al retardo (aplicaciones en tiempo real) si los servidores están ubicados en un sitio remoto. Este tipo de aplicaciones es común en entornos inteligentes como Smart Cities y Smart Factories. Además, la cantidad de datos que se generan en los dispositivos IoT, se va agrupando conforme se aproximan a los servidores en la nube, ver figura 1. Lo cual puede exceder la capacidad de los enlaces que conforman la red de comunicaciones [Chaudhary, 2017].



a) Cloud computing. b) Fog computing.

Figura 1 Cloud computing y fog computing.

Fog Computing busca resolver estos retos con una idea muy sencilla: En vez de enviar todos los datos a servidores remotos, se brindan servidores tan cerca como sea posible de las fuentes y destinatarios de datos, como se aprecia en la figura 1b. Dado que puede haber muchos de estos nodos intermedios, se crea un poderoso sistema distribuido que aumenta sustancialmente la capacidad de procesamiento. Puesto que los nodos están cerca de los equipos terminales, la latencia disminuye drásticamente, ver trayectorias rojas en la figura 1 y también disminuye la probabilidad de congestión de los enlaces.

Marco contextual

El Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST, por sus siglas en inglés) define CC como un modelo con base en la demanda de recursos computacionales compartidos. Es posible reconfigurar dinámicamente esos recursos para una carga de trabajo escalable [Mell, 2011]. El documento hace referencia a tres modelos de servicio:

- Software as a Service (SaaS). Los usuarios acceden a aplicaciones que están hospedadas en servidores en la nube, en principio, “en cualquier momento, desde cualquier lugar y con cualquier dispositivo que tenga conexión a Internet”.
- Platform as a Service (PaaS). Se proporciona un entorno para desarrollar aplicaciones (infraestructura, ambientes de desarrollo). El cliente de PaaS ofrece estas aplicaciones a usuarios finales.
- Infrastructure as a Service (IaaS). Este modelo es el que más interesa a los fines del presente trabajo. Lo que se ofrece es la provisión de recursos de cómputo que el cliente pueda desarrollar cualquier tipo de aplicación como lo haría en sus propios servidores.

El éxito de CC es incuestionable. Es una unidad de negocios muy importante para las grandes firmas de TI como Google (Google Cloud), Amazon (AWS Elastic Cloud Computing), IBM (IBM Cloud) y Microsoft (Azure Services). Sin embargo, este paradigma presenta varios inconvenientes para ciertos contextos de IoT y entornos inteligentes que requieren de tiempos de respuesta rápidos (aplicaciones en tiempo real), que pueden saturar los recursos de la red o de la capacidad de procesamiento contratada en la nube.

Para estos contextos, se han propuesto nuevos modelos que buscan optimizar el uso de los recursos computacionales que ya existen en las redes, creando entornos distribuidos. Algunos de estos paradigmas son:

- **Edge Computing**: El procesamiento se realiza en el borde de la red, en (pequeños) centros de datos cercanos a los usuarios [Open Edge Computing, 2021].

- **Mist Computing**: El procesamiento (al menos parte de él) se realiza con los recursos de los mismos dispositivos IoT [Yogi, 2017].
- **Cloudlets**: También conocido como micro nube móvil, consiste en un pequeño centro de datos a la orilla de la red muy orientado a aplicaciones móviles interactivas que requieren de cómputo intensivo y baja latencia [Satyanarayanan, 2009].
- **Fog Computing**: La "niebla" está conformada por los dispositivos, nodos en el borde, las redes que los interconectan (específicas a IoT, como ZigBee o redes LAN como WiFi) y quizás otros nodos en la ruta hacia la nube. En todos ellos se puede hacer procesamiento, almacenamiento y gestión [Bonomi, 2012].

No existe en la actualidad un consenso para la definición de estos paradigmas y, de hecho, las definiciones se modifican con el progreso tecnológico, pero todos tienen un término en común: el borde o frontera (*Edge*) entendida como la demarcación entre los dispositivos y equipos terminales, y el punto de contacto a Internet.

Existen varios organismos y grupos de investigación que trabajan en la definición de FC y la implementación de su arquitectura. Entre los más sobresalientes se encuentran:

- DECENTER es un proyecto de investigación e innovación que está desarrollando una plataforma robusta de FC, que cruza el continuo Cloud-to-Things. Su plataforma busca proporcionar orquestación y provisión de recursos con reconocimiento de aplicaciones, impulsado por métodos de inteligencia artificial [DECENTER, 2020].
- Open Fog Consortium es un consorcio de empresas de la industria de alta tecnología e instituciones académicas de todo el mundo cuyo objetivo es la estandarización y promoción de FC en diversas capacidades y campos. Recientemente este consorcio se fusionó con el Consorcio de Internet Industrial [IIC, 2019].

FC es un paradigma muy reciente y en constante evolución, por lo que resulta inviable implementar pruebas de concepto en redes de producción o aún en

maquetas de experimentación. Al mismo tiempo, las promesas de este nuevo modelo invitan a conocerlo con mayor detalle y evaluar su potencial. Para ello, se han desarrollado distintas herramientas dentro del campo de la simulación por computadora, por ejemplo:

- iFogSim: es un kit de herramientas de simulación de FC, en el cual se pueden modelar infraestructuras y aplicaciones para medir el rendimiento en términos de latencia, consumo de energía y uso de la red [Gupta, 2017].
- FogTorchII: Permite evaluar implementaciones de infraestructura de FC, modela capacidades de software, capacidades de hardware y atributos de QoS, incluida la latencia y el ancho de banda [Brogi, 2017].

2. Métodos

El estudio exploratorio presentado en este trabajo se realizó evaluando distintos escenarios en el ambiente de simulación iFogSim (iFS). Se eligió esta herramienta debido a su versatilidad y facilidad para modelar arquitecturas FC con un nivel de detalle razonable, y con la capacidad de capturar las métricas de desempeño de interés, como tasa de procesamiento y latencia. En esta sección se introducen las principales características de iFS así como el ambiente hipotético en el que se basa el estudio exploratorio y los escenarios de evaluación.

Arquitectura general

El modelado de un ambiente de FC en iFogSim, sigue una arquitectura de capas [Gupta, 2017]:

- **Sensores y Actuadores**. Modelan los dispositivos IoT que interactúan con el mundo real. Los sensores son la fuente de datos de una Aplicación en FC. Los actuadores modelan la acción sobre el entorno que debe realizarse como respuesta al procesamiento de los datos generados por los sensores. Los sensores generan tuplas de datos parametrizables que definen tres características clave: El volumen de datos (que incide en la utilización de los enlaces de red), la cantidad de procesamiento requerido (que incide en la utilización de los procesadores) y la tasa de emisión, que puede ser

constante (determinista) o con base en una función de distribución estadística.

- **Fog Devices**. Son los nodos que conforman el ambiente de FC. Tienen capacidades de procesamiento y almacenamiento, por lo que pueden ejecutar aplicaciones enteras o partes de una aplicación (por ejemplo, una función), llamadas Application Modules, AppMod.
- **IoT Data Streams**. Modelan el flujo de información que viaja a través de los nodos y que es procesada por los AppMod. Estos flujos inician con las tuplas emitidas por los sensores y están dirigidas a un AppMod, que las procesa y genera nuevas tuplas dirigidas a otro AppMod o, quizás, a un actuador.
- **Monitoreo de infraestructura**. Es la capa encargada de estimar datos relacionados con la utilización de los recursos, consumo de energía, disponibilidad de los elementos de la topología simulada, entre otros.
- **Gestión de Recursos**. Monitorea la disponibilidad de recursos en los Fog Devices, y con base en ello asigna y despacha cargas de trabajo.
- **Modelos de Aplicación**. El despliegue de una aplicación se modela como un flujo de datos distribuido. La aplicación está compuesta por una serie de módulos (los AppMod). La salida del módulo i puede ser la entrada de un módulo j, generando una dependencia de datos representada por un grafo dirigido en la que los vértices son los AppMod y los arcos son el flujo de información caracterizados por las tuplas de los IoT Data Streams. Un ejemplo se muestra en la figura 2.

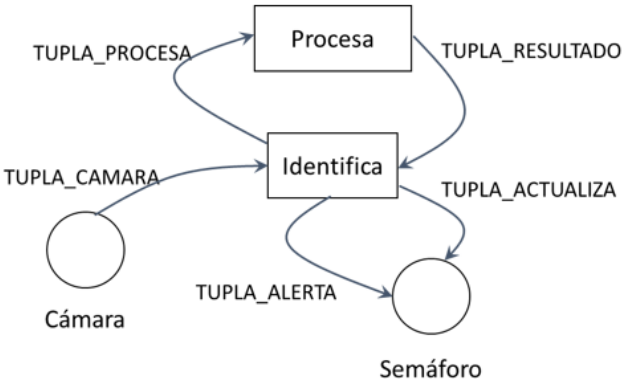


Figura 2 Diagrama de flujo de datos entre los módulos de la aplicación.

Componentes

Los componentes que conforman las capas anteriores y otros complementarios, están implementados como clases de Java en iFS. Simular un ambiente consiste en instanciar las clases requeridas con los parámetros apropiados:

- **FogDevice**. Permite modelar los nodos del sistema. Sus principales atributos son la tasa de ejecución del procesador, la cantidad de memoria RAM y de almacenamiento, la capacidad de los enlaces ascendente y descendente y la latencia del enlace ascendente.
- **Sensor**. Modela los dispositivos sensores en la arquitectura a simular. Sus atributos permiten especificar las características estadísticas para la generación de tuplas, el **FogDevice** al que está conectado y la latencia del enlace de esta conexión.
- ***Actuador***. Permite modelar un actuador en la arquitectura simulada. Los atributos relevantes son el nodo al que está conectado y la latencia de este enlace.
- **Tupla**. Representa el flujo de datos entre los AppMod, sensores y actuadores. Se caracteriza por su tipo (arbitrario, es un identificador) y los AppMod origen y destino. Los otros dos argumentos especifican “la cantidad de trabajo” para procesarla en el nodo receptor y el volumen de datos transmitidos.
- **Aplicación**. La aplicación que se desea modelar se especifica como una gráfica acíclica dirigida (DAG, *Directed Acyclic Graph*) entre los módulos que conforman la aplicación. Para ello se definen las clases AppModule, que representan los AppMod como vértices de la DAG, y AppEdge, que representan los arcos dirigidos de la DAG y por tanto definen el flujo de información. AppLoop es una clase adicional que permite medir la latencia de los flujos de interés.

Escenario de evaluación

El escenario que se propone en este trabajo para comparar los ambientes CC y FC es un sistema de control de semáforos en una Ciudad Inteligente. En cada

crucero con semáforo, se tienen N ≥ 2 cámaras, cada una capturando imágenes en la dirección del flujo vehicular. Si el crucero tiene sólo dos calles de un solo sentido, se requiere de dos cámaras; un crucero donde convergen cuatro calles bidireccionales requerirá de ocho cámaras.

Las cámaras monitorean el nivel de congestión vehicular en una calle. Si se empieza a saturar, la luz del semáforo correspondiente debería pasar de verde a rojo. En un sistema de tránsito inteligente, este tipo de decisión no puede tomarse de manera local; debe analizarse qué está ocurriendo con el flujo calles abajo, arriba, en calles perpendiculares y en adyacentes. Más aún, se esperaría poder identificar de inmediato las causas de una eventual congestión (por ejemplo, una colisión, un objeto obstruyendo la calzada) lo cual justifica el uso de cámaras para el monitoreo del tráfico. Una versión simplificada de la aplicación representada bajo el modelo DDF, se muestra en la figura 2. Las cámaras son los *Sensores.* Capturan imágenes periódicamente y las envían a un AppMod *Identifica* que se encarga de analizar la imagen recibida para estimar el nivel de saturación de la calle y condiciones de alerta como la presencia de un objeto o una colisión. El resultado de este módulo se envía a *Procesa* en el que se correlaciona la información de varios módulos *Identifica*. La mayoría de las veces, *Procesa* envía una tupla de actualización a los semáforos, que son los *Actuadores*. Sin embargo, si el módulo *Identifica* detecta que hay una saturación en la ocupación de la calle, enviará de inmediato una notificación al semáforo para que se ponga en rojo en el sentido de la congestión. La topología del escenario se muestra en la figura 3.

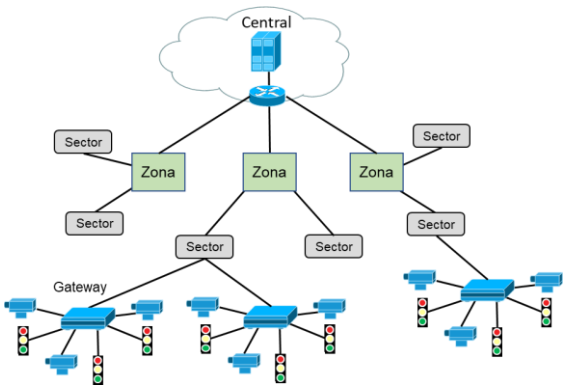


Figura 3 Topología del escenario de estudio.

Una serie de cámaras y semáforos conectadas a un nodo *Gateway* corresponderían a un crucero. Varios *Gateways* se conectan a un nodo *Sector* que representa un conjunto de cruceros en un barrio. Varios nodos *Sector* se conectan a un nodo *Zona* y varios nodos *Zona* se conectan al nodo *Central* donde se estaría monitoreando todo el sistema de semaforización de la ciudad. El nodo *Central* se hospeda en la nube.

Por razones de espacio, en este trabajo el estudio se realiza en un subconjunto de la topología anterior, el cual es suficiente para el análisis al comparar CC y FC. Tiene una cámara, un semáforo, un Gateway y un nodo Central en la nube, como se muestra en la figura 4, junto con las características de los enlaces.

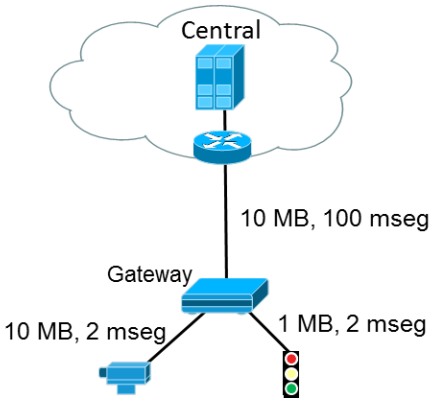


Figura 4 Topología simplificada para evaluación de los escenarios.

Los resultados que se presentan en la siguiente sección corresponden a la siguiente batería de experimentos:

- Capacidad de procesamiento

  Escenario 1. Procesamiento en Cloud: Los AppMod Identifica y Procesa se ejecutan en el nodo Central, en la nube. El nodo tiene una capacidad de procesamiento limitada y la “cantidad de trabajo” emitida por la cámara (la TUPLA_CAMARA), que debe ser procesada por el AppMod Identifica, se incrementa gradualmente. La cantidad de trabajo del AppMod Identifica a Procesa, (la TUPLA_PROCESA) se mantiene constante, al igual que la cantidad de trabajo emitida en las tuplas TUPLA_RESULTADO y TUPLA_ACTUALIZA.

Escenario 2. Procesamiento Distribuido: El módulo Identifica se ejecuta en el nodo Gateway y el módulo Procesa en el nodo Central. La cantidad de trabajo de las tuplas TUPLA_CAMARA y TUPLA_PROCESA es la misma y se incrementa gradualmente.

- Ancho de banda

  Escenario 3. Se utiliza la configuración del escenario 2 y se incrementa gradualmente la cantidad de datos transportados por TUPLA_PROCESA, que viaja por la conexión entre Gateway y Central. Se analiza el efecto en el ancho de banda de los enlaces.

- Latencia

  Escenario 4. Bucle largo: El flujo "regular": Cámara → Identifica → Procesa → Identifica → Semáforo y Cámara → Identifica → Semáforo de la figura 2, tiene que recorrer toda la red, ocasionando tiempos de respuesta largos. Se analiza si en el tiempo de respuesta también influye el tiempo de procesamiento al incrementar la carga de trabajo en las tuplas.

  Escenario 5. Bucle corto: Se analiza la latencia del flujo "de alerta": Sensor → Identifica → Semáforo en la configuración del escenario 2 y se evita el recorrido en el tramo de latencia grande hacia el nodo Central.

3. Resultados

Capacidad de procesamiento

En el **escenario 1** se fija la capacidad de procesamiento del nodo *Central* en 44,800 MIPS (este es el valor por defecto en iFS) y la cantidad de trabajo de las tuplas TUPLA_PROCESA, TUPLA_RESULTADO y TUPLA_ACTUALIZA en 1,000 MIPS. La TUPLA_CAMARA incrementa su cantidad de trabajo de 10,000 MIPS a 39,600 MIPS y se estudian los tiempos de ejecución de las tuplas. La cámara genera tuplas a una tasa constante de 1.0 tuplas por segundo. Los resultados se muestran en la tabla 1. Conforme aumenta la cantidad de trabajo que debe procesarse en el nodo *Central,* el tiempo de ejecución de todas las tuplas aumenta pues los recursos de cómputo disponibles en el nodo en la nube se van agotando. El tiempo de ejecución de la TUPLA_CAMARA se ve mucho más afectado que el de las otras

dos tuplas: El tiempo de ejecución de TUPLA_PROCESA el incremento es marginal y el de TUPLA_ACTUALIZA se incrementa casi 30 veces, pero el tiempo de ejecución de TUPLA_CAMARA simplemente se dispara, como se muestra en la figura 4.

Tabla 1 Tiempos de ejecución al incrementar la carga de trabajo, escenario 1 (nube).

TUP_CAMARA	Tiempo de ejecución				
Cantidad de trabajo	TUP_CAMARA	TUP_PROCESA	TUP_RESULTADOS	Utilización	Latencia
10,000	0.323	0.122	0.122	0.290	204.968
15,000	1.000	0.144	0.144	0.402	205.687
20,000	1.000	0.145	0.145	0.513	205.689
25,000	1.000	0.145	0.145	0.625	205.689
30,000	1.000	0.145	0.145	0.737	205.689
35,000	2.000	0.145	0.167	0.848	206.711
37,000	3.000	0.145	0.189	0.893	207.733
37,500	7.063	0.153	0.283	0.904	211.962
38,000	9.333	0.189	0.346	0.915	214.435
38,500	12.333	0.189	0.412	0.926	217.431
39,000	17.291	0.184	0.521	0.938	222.461
39,350	80.041	0.150	2.463	0.945	282.549
39,500	99.333	0.189	2.610	0.949	299.501
39,600	139.000	0.145	3.655	0.951	332.661

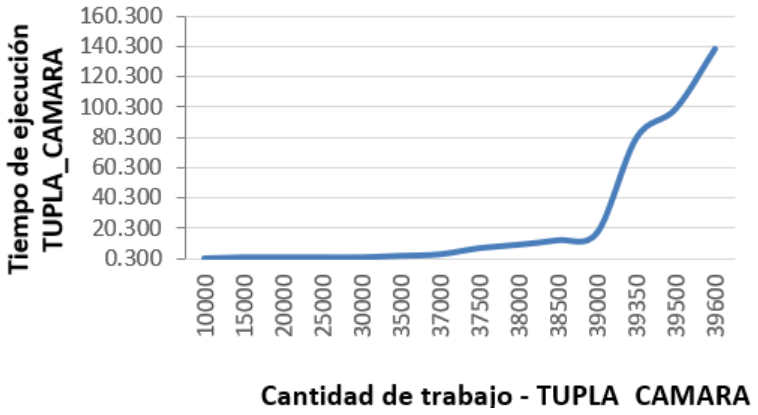


Figura 4 Tiempo de ejecución de TUPLA_CAMARA en el escenario 1.

En el **escenario 2** se distribuye el procesamiento. En el nodo *Gateway* se procesan TUPLA_CAMARA y TUPLA_RESULTADO y en el nodo *Central* se procesa TUPLA_PROCESA. Los dos nodos tienen la misma configuración; en particular,

cada nodo tiene una capacidad de procesamiento de 44,800 MIPS[2]. Las tuplas TUPLA_CAMARA y TUPLA_PROCESA incrementan su cantidad de trabajo de 5,000 (10,000 sumadas) a 43,000 (86,000 sumadas). Los resultados se presentan en la tabla 2. El comportamiento general es similar al del primer escenario: Los tiempos de ejecución son relativamente estables hasta un cierto punto (el llamado *knee point*) en el que los tiempos de ejecución crecen excesivamente. En la figura 5 se muestran los resultados para la TUPLA_PROCESA, que es en la que el incremento en los tiempos de ejecución es más evidente.

Tabla 2 Tiempos de ejecución al incrementar la carga de trabajo, escenario distribuido.

TUP_CAMARA + TUP_PROCESA	Tiempo de ejecución				
Cantidad de trabajo	TUP_CAMARA	TUP_PROCESA	TUP_ACTUALIZA	Utilización	Latencia
10,000	0.212	0.212	0.122	0.134	204.966
15,000	1.000	0.323	0.122	0.190	205.714
30,000	0.435	0.470	0.530	0.357	205.807
40,000	0.547	1.000	0.145	0.469	205.934
50,000	0.658	0.679	0.145	0.580	205.961
60,000	0.770	1.000	0.145	0.692	206.314
70,000	0.881	1.000	0.145	0.804	206.425
80,000	0.993	1.000	0.145	0.915	206.537
85,000	1.000	3.000	0.189	0.971	208.534
85,500	2.000	6.000	0.256	0.977	212.299
86,000	6.942	23.874	0.658	0.982	232.664

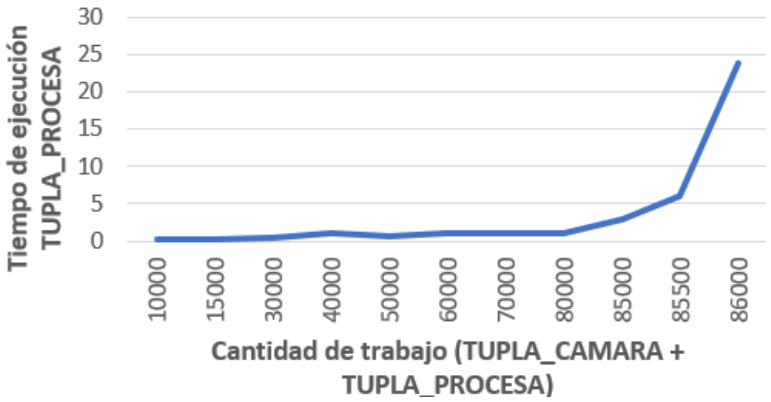


Figura 5 Tiempo de ejecución de TUPLA_PROCESA en el escenario 2.

[2] *En realidad, es de esperar que los elementos de procesamiento en el borde, como el Gateway, tenga una capacidad de procesamiento mucho menor que los elementos en la nube.*

Ancho de banda

En el **escenario 3** se mantiene la configuración de procesamiento distribuido con una carga de trabajo de TUPLA_CAMARA y TUPLA_PROCESA en 10,000 cada una, para que no afecte el tiempo de ejecución. La cantidad de datos a transmitir en TUPLA_PROCESA se incrementa de 1000 hasta 13,000 bytes, lo que excede en 30% la capacidad del enlace que es el ancho de banda configurado en los enlaces. Los resultados se muestran en la tabla 3. iFogSim reporta la cantidad de bytes que viajaron por los enlaces del escenario de simulación. Esa es la penúltima columna de la tabla 3, misma que se presenta gráficamente en la figura 6.

Tabla 3 Uso de la red al incrementar la cantidad de datos en tránsito.

TAM_DATOS	Tiempo de ejecución				
TUP_PROCESA	TUP_PROCESA	TUP_SENSOR	TUP_RES	Uso de Red	Latencia
1,000	0.212	0.212	0.122	149,455	205.016
3,000	0.212	0.323	0.332	349,390	205.465
5,000	0.212	0.131	0.869	549,330	205.907
7,500	0.212	0.212	0.122	799,255	205.688
9,000	1	0.212	0.122	949,205	206.583
9,500	0.212	0.212	0.122	999,195	206.496
10,000	1	0.212	0.122	1,049,175	206.683
11,000	0.212	0.293	0.074	1,044,660	651.083
12,000	0.212	0.213	0.128	1,040,950	1,022.1
13,000	0.212	0.51	0.55	1,037,760	1,336.198
14,000	0.212	0.213	0.128	1,035,080	1,605.1

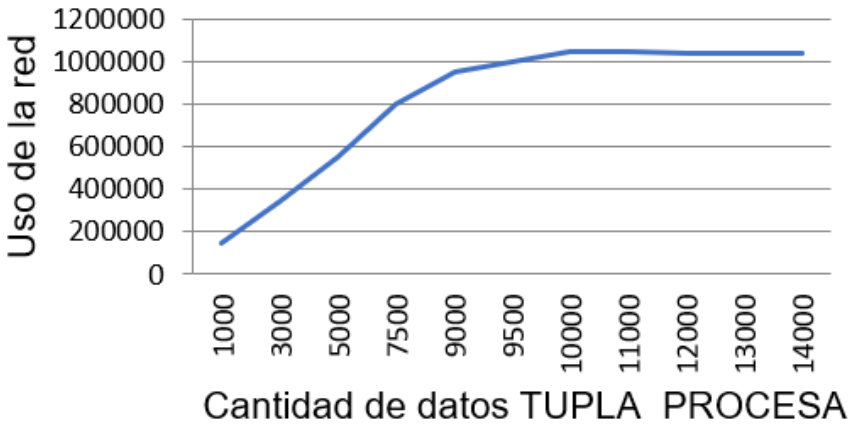
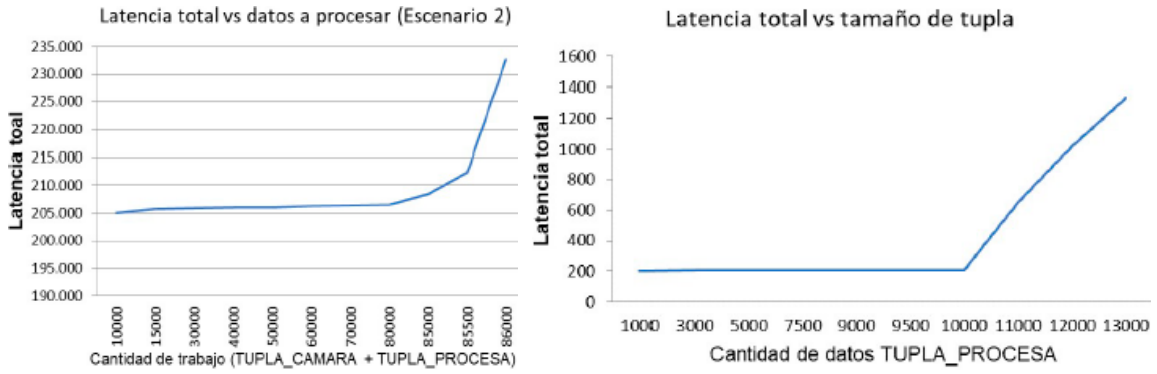


Figura 6 Utilización de la red al aumentar la carga en TUPLA_PROCESA.

Latencia

La latencia en este estudio es el tiempo que transcurre desde que la cámara envía una imagen, hasta que el semáforo recibe una instrucción. Todas las tuplas emitidas por la cámara generan una instrucción en el semáforo. La latencia se ve afectada por muchos factores; los tres más importantes son el tiempo de propagación, los tiempos de procesamiento y los tiempos de transmisión.

Para el bucle largo del **escenario 4**, el efecto de los tiempos de procesamiento en la latencia se muestra en la última columna de las tablas 1 (Procesamiento en la nube) y tabla 2 (Procesamiento distribuido), y el efecto de los tiempos de transmisión, en la última columna de la tabla 3. Estos resultados se grafican en la figura 7.



a) Tiempo de procesamiento *b) Tiempo de transmisión*

Figura 7 Efecto de los retardos de procesamiento y transmisión en la latencia total.

En el **escenario 5** de bucle corto, el AppMod *Identifica* emite una tupla TUPLA_ALERTA que va del *Gateway* al semáforo. No interviene el retardo de transmisión de la red hacia el nodo *Central*, que es lo que contabiliza iFS. La latencia total obtenida para este escenario fue de 4.854 milisegundos.

4. Discusión

Capacidad de procesamiento

El crecimiento exponencial en el tiempo de ejecución para TUPLA_CAMARA tras el *knee point* de la figura 4 no es tan sorprendente si se analiza desde la óptica de la teoría de colas. La capacidad del nodo Central puede ser considerada la tasa de

servicio, μ, y la cantidad de trabajo que se recibe de las tuplas, sería la tasa de llegada, λ. Así, la utilización del nodo Central puede representarse como $\rho = \lambda/\mu$. Esta utilización se muestra en la penúltima columna de la tabla 2. Por ejemplo, para el caso en que la TUPLA_CAMARA requiere de 37,000 MIPS de procesamiento, el cálculo de ρ es $37{,}000 + 3 * 1{,}000)/44{,}800 = 0.893$. Como lo predice la teoría de colas, conforme ρ se acerca a 1, los parámetros de desempeño, como el tiempo de ejecución, se degradan exponencialmente.

Para el modelo de Fog Computing (escenario 2), lo que más llama la atención, es la cantidad de trabajo *total* que se puede ejecutar. Mientras que en el primer escenario, en el que todas las tuplas se procesan en la nube el *knee point* se alcanza con una carga de trabajo de aproximadamente 43,000 MIPS (39,000 de la TUPLA_CAMARA y 3,000 de las otras tres), en este escenario el punto se alcanza con el doble de carga: 86,000 MIPS. Es lógico, pues en esencia, la arquitectura ofrece el doble de capacidad de procesamiento (distribuido). Lo importante aquí es que, si la carga de trabajo requerida se acerca a la capacidad del nodo en la nube, con la arquitectura FC un factor esencial de la calidad de servicio, que es el tiempo de ejecución, se ve sumamente beneficiado. Queda demostrado el primer beneficio de este paradigma: El modelo de cómputo distribuido de FC permite ejecutar más trabajo (total) con tiempos de ejecución menores.

Ancho de banda

En este experimento se está aumentando el tamaño de la tupla que viaja del nodo *Gateway* al *Central* en un enlace de 10 MBytes/s[3]. Se aumenta el uso de la red hasta agotar los recursos del enlace. iFS reporta la cantidad de tráfico en todos los enlaces, excepto los que conectan sensores y actuadores con su Gateway. En el escenario estudiado, viajan dos tuplas: TUPLA_PROCESA y TUPLA_RESULTADO; esta última con un tamaño fijo de 500 Bytes. Para TUPLA_PROCESA de 1000 Bytes (renglón 1, tabla 3), se reporta un uso de 149,455, Bytes; se concluye que se registraron prácticamente 100 tuplas de cada

[3] *La velocidad de los enlaces se debe especificar en bits por segundo, pero por comodidad en iFogSim se especifica en Bytes por segundo.*

tipo. Las corridas se simulan durante 100 segundos y TUPLA_CAMARA se emite cada segundo, disparando la generación de las demás tuplas a la misma tasa.
Esta cadencia de 100 tuplas de cada tipo se mantiene hasta el punto de saturación del enlace (10 MBytes/s). A partir de ese momento, se observa una disminución en el desempeño como se observa en los últimos tres renglones de la tabla 3. Las tuplas se están generando a una tasa mayor a la que pueden ser transmitidas.
El aumento del tamaño de la TUPLA_PROCESA es un artefacto para ejemplificar la posibilidad de congestión en una topología más realista, como la de la figura 3. Una forma de evitar ese punto de congestión es precisamente procesando las tuplas en los servidores en la niebla y sólo enviar hacia la nube datos agregados o aquellos que requieren de un análisis más holístico. Con esto se demuestra el riesgo de utilizar CC en un entorno de IoT donde el número de datos a procesar es enorme.

Latencia

En el modelo simplificado que se ha estudiado en este trabajo, figura 4, el bucle largo tendría un retardo de propagación de 204 milisegundos: 2 de la cámara a Gateway, 100 de este nodo a Central, 100 en el sentido inverso y 2 de Gateway al semáforo. Bajo condiciones de poca carga y poca congestión, este es aproximadamente el valor reportado en los escenarios explorados: Son los primeros renglones de la columna Latencia en todas las tablas. La diferencia se debe precisamente a los otros dos elementos que no se suelen tomar en cuenta: retardo de procesamiento y retardo de transmisión. En condiciones de poca carga y poca congestión, la diferencia puede ser despreciable.
En el escenario de bucle corto, la latencia esperada sería de 4 milisegundos, muy cercano a lo reportado por el último experimento. Esto demuestra el tercer, y quizás más citado beneficio de Fog Computing: La capacidad de proveer de servicios con latencias apropiadas para las aplicaciones en tiempo real.
Estos experimentos ponen en evidencia que, en condiciones de alta demanda como las que pueden ocurrir en IoT y entornos inteligentes, la latencia puede verse severamente afectada por saturación de los recursos de cómputo o por congestión en la red, ver figura 7. Así, el procesamiento distribuido y la dispersión de carga en

distintos enlaces de la red característicos del paradigma Fog Computing contribuyen a garantizar la calidad de servicio que requieren estas aplicaciones.

5. Conclusiones

En este artículo se ha analizado el impacto en dos métricas clave en la calidad de servicio: Latencia y tiempo de ejecución en una aplicación que genera grandes volúmenes de datos, a gran velocidad y que requiere de tiempos de respuesta muy cortos. Estos requerimientos son característicos de IoT y los entornos inteligentes. El caso de uso elegido simula el control de semáforos en una Ciudad Inteligente.
En la actualidad se privilegia un modelo de Cloud Computing para desplegar este tipo de aplicaciones. En los experimentos realizados por medio de simulación, se ha demostrado que este modelo puede afectar seriamente las dos métricas mencionadas, haciendo inviable el despliegue de la aplicación. Para estos entornos, se ha demostrado que el paradigma de Fog Computing es una muy atractiva alternativa. Sus capacidades de procesamiento distribuido van haciendo una realidad la visión de cómputo ubicuo. Al quitar estrés a la nube, a los enlaces que conducen a ella y al reducir el bucle de aplicación, tanto la latencia como el tiempo de ejecución pueden cumplir con rigurosos requerimientos de calidad de servicio.

Agradecimientos

Este trabajo ha sido realizado con el apoyo de la Asociación Mexicana de Cultura, AC y del Consejo Nacional de Ciencia y Tecnología.

6. Bibliografía y Referencias

[1] Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, y Sateesh Addepalli, Fog computing and its role in the Internet of Things. MCC Workshop on Mobile Cloud Computing. New York: ACM, 2012.

[2] Brogi, A., Forti S., y Ibrahim A., How to best deploy your fog applications, probably. IEEE 1st International Conference. IEEE, 2017.

[3] DECENTER, (2020). Decentralised technologies for orchestrated Cloud-to-Edge intelligence. 2020. https://www.decenter-project.eu/.

[4] Costello, Katie, y Meghan Rimol, Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17% in 2020. Stamford, Connecticut: Gartner Newsroom Press Release, 2019.

[5] Chaudhary, R., Kumar, N. y Zeadally, S., Network Service Chaining in Fog and Cloud Computing for the 5G Environment: Data Management and Security Challenges, IEEE Communications Magazine, 55(11), 2017.

[6] Gessert F., Wingerath W., Ritter N., (2020). Latency in Cloud-Based Applications. En: Fast and Scalable Cloud Data Management. Springer, Cham. https://doi.org/10.1007/978-3-030-43506-6_2.

[7] Gupta, H., Dastjerdi A., Ghosh S., y Buyya R., iFogSim: A toolkit for modeling and simulation of resource. Software Practice and Experience, John Wiley & Sons, 2017.

[8] IIC, (2019). The Industrial Internet Consortium And Openfog Consortium Unite. 2019. https://www.iiconsortium.org/press-room/01-31-19.htm.

[9] Mell, Peter, y Timothy Grance, The NIST Definition of Cloud Computing. NIST Special Publication 800-145. Gaithersburg, MD.: NIST, 2011.

[10] Open Edge Computing, (2021). Open Edge Computing Initiative. s.f. https://www.openedgecomputing.org/.

[11] Regalado, Antonio, Who Coined "Cloud Computing"? MIT, Cambridge: MIT Technology Review, 2011.

[12] Satyanarayanan, M., Bahl M., Caceres R. y Davies N., The Case for VM-Based Cloudlets in Mobile Computing. IEEE Pervasive Computing 8(4), 2009.

[13] Yogi, M., Chandrasekhar K., y Vijay Kumar G., Mist Computing: Principles, Trends and Future Direction. SSRG International Journal of Computer Science and Engineering 4(7), 2017.