A Three-Level Scheduler to Execute Scientific Experiments on Federated Clouds

E. Pacini, C. Mateos and C. G. Garino

***Abstract*— For executing current simulated scientific experiments it is necessary to have huge amounts of computing power. A solution path to this problem is the federated Cloud model, where custom virtual machines (VM) are scheduled in appropriate hosts belonging to different providers to execute such experiments, minimizing response time. In this paper, we study schedulers for federated Clouds. Scheduling is performed at three levels. First, at the broker level, datacenters are selected by their network latencies via three policies –Lowest-Latency-Time-First, First-Latency-Time-First, and Latency-Time-In-Round–. Second, at the infrastructure level, two Cloud VM schedulers based on Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) are implemented. At this level the scheduler is responsible for mapping VMs to datacenter hosts. Finally, at the VM level, jobs are assigned for execution into the pre-allocated VMs. We evaluate, through simulated experiments, how the proposed three-level scheduler performs w.r.t. the response time delivered to the user as the number of Cloud machines increases, a property known as horizontal scalability.**

***Keywords*— Scientific Experiments, Federated Cloud, Scheduling, Ant Colony Optimization, Particle Swarm Optimization.**

I. INTRODUCCIÓN

LA COMPUTACIÓN científica emplea normalmente modelos complejos y simulaciones que requieren de un gran número de recursos de cómputo para producir resultados en plazos de tiempo razonables. Un ejemplo representativo de estos modelos son los experimentos de barrido de parámetros (PSE, por sus siglas en inglés) [1]. Concretamente, ejecutar un PSE implica administrar muchos trabajos independientes, ya que un mismo experimento se ejecuta bajo múltiples configuraciones iniciales. Por otra parte, debido a que los modelos matemáticos detrás de los PSEs poseen una gran complejidad computacional, es necesario realizar una gran cantidad de cálculos para resolver los trabajos asociados. Para ello, un tipo de entorno paralelo que ha cobrado auge para la ejecución de estos experimentos son los Clouds [2], los cuales otorgan recursos de cómputo al usuario a través de Máquinas Virtuales (VM).

La ejecución de PSEs en Clouds no está libre del conocido problema de planificación de trabajos. Por lo tanto, es necesario desarrollar estrategias de planificación eficientes para asignar adecuadamente los trabajos a los recursos de cómputo y reducir el tiempo de cálculo asociado. Además, en Clouds federados [3], resulta aún más necesario el gestionar adecuadamente los recursos físicos, los cuales forman parte de centros de datos que se encuentran geográficamente distribuidos. Por lo tanto, para una adecuada ejecución de trabajos en Clouds federados, la planificación se debe realizar en tres niveles [4]. En primer lugar, a nivel de broker, se utilizan estrategias de planificación para seleccionar los centros de datos. Estas estrategias deben considerar cuestiones tales como las interconexiones de red o en caso de Cloud pagos el costo monetario de las máquinas [5] que componen los diversos centros de datos. En segundo lugar, a nivel de infraestructura, mediante el uso de un planificador específico, las VMs son asignadas a las máquinas físicas disponibles en los centros de datos previamente seleccionados. Por último, a nivel de VM, a través de otro componente de planificación de menor nivel, los trabajos son asignados para su ejecución en las VMs. Sin embargo, independientemente del nivel, la planificación es en general un problema NP-completo [6] y por lo tanto su solución no es trivial desde un punto de vista algorítmico. En este contexto, la necesidad de algoritmos de planificación se extiende hacia los tres niveles de un Cloud federado.

En los últimos años, las metaheurísticas del área de Inteligencia Colectiva (SI) [7] han recibido una gran atención entre los investigadores. Estas técnicas simulan el comportamiento colectivo de enjambres de insectos sociales tales como hormigas y abejas. Inspirados en estas capacidades, se han propuesto algoritmos para resolver problemas de optimización combinatoria, siendo las estrategias de SI más populares la Optimización por Colonias de Hormigas (ACO) y Optimización por Enjambre de Partículas (PSO). Además, debido a que la planificación de trabajos/VMs en Clouds es también un problema de optimización combinatoria, se han propuesto planificadores en esta línea que exploran el uso de SI [8]. Sin embargo, hasta hoy, muy pocos trabajos que aborden el uso de SI en el contexto de Clouds federados han sido encontrados en la literatura.

A diferencia de trabajos previos [9], [10], que aprovechan centros de datos individuales, en este trabajo hemos extendido nuestro planificador para operar en un Cloud federado donde los centros de datos se encuentran geográficamente distribuidos pero conforman un Cloud único. Para ello hemos incluido un nuevo nivel de planificación –el nivel de broker– obteniendo así un nuevo planificador que opera en tres niveles, y no en dos niveles como en Clouds compuestos por un único centro de datos [9], [10]. En primer lugar, por medio de una política que opera a nivel de broker, los centros de

E. Pacini, ITIC Research Institute, Facultad de Ciencias Exactas y Naturales, UNCuyo University & CONICET, Mendoza, Argentina, epacini@uncu.edu.ar

C. Mateos, ISISTAN-CONICET, UNICEN University, Tandil, Buenos Aires, Argentina, cmateos@conicet.gov.ar

C. G. Garino, ITIC Research Institute & Facultad de Ingeniería, UNCuyo University, Mendoza, Argentina, cgarcia@itu.uncu.edu.ar

datos son seleccionados de acuerdo a sus interconexiones de red y sus latencias. De hecho, las latencias de red de los diversos centros de datos pueden contribuir a afectar negativamente el tiempo de respuesta entregado al usuario. Para ello se evalúan tres políticas, Menor-Latencia-Primero (Lowest-Latency-Time-First, LLTF), Primer-Latencia-Primero (First-Latency-Time-First, FLTF), y Latencias-En-Ronda (Latency-Time-In-Round, LTIR). Luego, a nivel de infraestructura, se explora ACO y PSO para la asignación de las VMs en los recursos físicos de un centro de datos. Para asignar las VMs en las máquinas físicas, el planificador debe realizar un número diferente de "consultas" –mensajes a través de la red– a las máquinas para determinar la disponibilidad de sus recursos en cada intento de creación de una VM. Claramente, el número de consultas a realizar por cada algoritmo y las latencias internas de los centros de datos también influyen en el tiempo de respuesta al usuario. Por último, a nivel de VM, los trabajos de los PSEs son asignados a las VMs utilizando FIFO, al igual que en [9]. Sin embargo, es importante mencionar que en este nivel otras políticas podrían implementarse, como por ejemplo la política implementada en [1], que procesa los trabajos en base a prioridades pre-asignadas a los mismos por el usuario. En resumen, en este trabajo incluimos el nivel de bróker y evaluamos cómo las decisiones tomadas tanto a nivel de broker como a nivel de infraestructura influyen en el tiempo de respuesta otorgado al usuario al ejecutar PSEs.

Se realizaron experimentos con datos de trabajos extraídos de un PSE real [11] que involucra un problema viscoplástico. Los resultados sugieren que los planificadores basados en SI a nivel de infraestructura, en combinación con la política LLTF a nivel de broker y FIFO a nivel de VM, ofrecen un buen rendimiento en el tiempo de respuesta con respecto a utilizar FLTF y LTIR a nivel de broker. La razón de basar el planificador general en ACO y PSO a nivel infraestructura es debido a que en publicaciones previas mostramos cómo tales algoritmos en conjunto con FIFO, o bien una política basada en prioridades a nivel VM, ofrecen mejor rendimiento con respecto a algoritmos genéticos, Random y Best Effort para el caso de Clouds no federados (ver [1], [12]). En lo que sigue, nos focalizamos en estudiar cómo las políticas a nivel broker complementan los algoritmos para el caso de Clouds federados. En particular, se evalúa hasta qué punto tales políticas mejoran el tiempo de respuesta otorgado al usuario ante un incremento en el número de recursos disponibles en el Cloud, lo cual también se conoce como escalabilidad horizontal. La escalabilidad horizontal [13] es una de las principales características de los entornos Cloud debido a que permite mejorar el rendimiento de las aplicaciones. Los resultados obtenidos a través de experimentos simulados muestran que el menor tiempo de respuesta se obtiene cuando se combina el uso de LLTF a nivel de broker junto con PSO para asignar las VMs en las máquinas físicas. Los experimentos se realizaron utilizando el difundido simulador CloudSim [14].

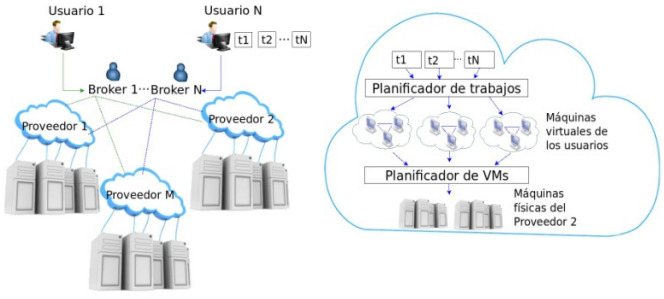
El resto del artículo se ha estructurado de la siguiente manera. La Sección II detalla las bases conceptuales que sustentan nuestro planificador. Luego, la Sección III analiza trabajos relacionados relevantes. La Sección IV describe nuestra propuesta. Posteriormente, en la Sección V se realiza una evaluación experimental de la misma. Finalmente, la Sección VI concluye el documento y discute trabajos futuros.

II. MARCO TEÓRICO

Las infraestructuras Cloud [2], [15] ofrecen a los usuarios finales una variedad de servicios que cubren todas sus necesidades de cómputo y despliegue de aplicaciones. En Clouds, tanto los científicos en general como los usuarios de PSEs en particular, pueden personalizar completamente sus entornos de ejecución realizando la configuración más adecuada para ejecutar sus experimentos. Un Cloud también permite escalar dinámicamente las aplicaciones de los usuarios mediante la provisión de recursos de cómputo a través de VMs. Además, con el objetivo de lograr un buen rendimiento, las VMs tienen que utilizar completamente los recursos físicos con el fin de mejorar su utilidad [16].

Los Clouds federados [16] son infraestructuras que poseen recursos físicos pertenecientes a diferentes proveedores de Cloud, y hacen uso de brokers para satisfacer las necesidades de sus usuarios. Un broker es una entidad que mantiene una cola de solicitudes de un usuario y que deben ser provistas por un centro de datos. En el contexto de este trabajo, donde un usuario ejecuta un PSE –un conjunto de trabajos–, a cada usuario se le asocia un broker.

La Fig. 1 ilustra un Cloud federado donde uno o más usuarios se conectan a través de una red y requieren la creación de un número de VMs para ejecutar sus experimentos (subfigura 1a), es decir, un conjunto de trabajos. Como puede observarse, un broker es creado por cada usuario que se conecta al Cloud. Cada broker, a través de las interconexiones de red, conoce quiénes son los proveedores que forman parte de la federación. La relación de cada broker con los proveedores Cloud está representada con líneas punteadas en color verde y azul. La subfigura 1b ilustra la ejecución de los trabajos enviados por el usuario *User N* al centro de datos del proveedor *Cloud Provider 2*, que involucra los planificadores a nivel de infraestructura y a nivel de VM.



(a) Cloud Federado (b) Planificación intra-centro de datos

Figura 1. Cloud Federado: Vista General.

El valor de utilizar Cloud como una herramienta para la ejecución de aplicaciones científicas complejas [17] ha sido ya reconocido en la comunidad científica. Sin embargo, para

lograr una ejecución eficiente de este tipo de aplicaciones, tanto el problema de planificación de trabajos como el de VMs deben ser abordados.

Por otra parte, las técnicas de SI [7] han demostrado ser útiles en problemas de optimización. La ventaja de estas técnicas se deriva de su capacidad para explorar soluciones eficientemente en grandes espacios de búsqueda. Dentro de las técnicas de SI las más populares son ACO y PSO. ACO [7] surge de la observación de cómo las colonias de hormigas buscan los caminos más cortos para llegar a una fuente de alimento desde su nido. En la naturaleza, las hormigas reales se mueven al azar de un lugar a otro en busca de alimento, y al encontrar comida y regresar a su nido, cada hormiga deja una hormona, llamada feromona, que atrae a otras hormigas a que sigan el mismo camino. A medida que más y más hormigas eligen el mismo camino, el rastro de feromona se va intensificando y así incluso más hormigas sigan ese camino.

PSO [7] es una técnica que busca una solución en un espacio de búsqueda mediante el modelado y predicción de la conducta social de insectos en presencia de objetivos. El término "partículas" se utiliza para representar aves, abejas u otros individuos que exhiben comportamiento social como grupo e interactúan entre sí. Un ejemplo basado en la naturaleza es útil para ilustrar el algoritmo: un grupo de abejas vuela sobre un campo en busca de flores. Su objetivo es encontrar la mayor cantidad posible de flores. Al principio, las abejas no tienen conocimiento del campo y vuelan a lugares aleatorios en busca de flores. Cada abeja puede recordar los lugares donde se vio la mayor cantidad de flores, y, además, conocer los lugares en los que otras abejas han encontrado una alta densidad de flores. Estas dos piezas de información –nostalgia y conocimiento social– modifican continuamente la trayectoria de las abejas. Así, cada abeja altera su camino entre las dos direcciones para volar en algún lugar entre los dos puntos y encontrar una mayor densidad de flores. Ocasionalmente, una abeja puede volar sobre un lugar en el campo con mayor densidad de flores que todos los encontrados hasta el momento por el enjambre, luego de lo cual todo el enjambre es atraído hacia esa nueva dirección.

III. TRABAJOS RELACIONADOS

En los últimos diez años, las técnicas basadas en SI, y más específicamente ACO y PSO [18], [19], han sido el foco de un gran número de estudios de investigación para resolver problemas de optimización combinatoria. Trabajos recientes [20], [21], [22] evidencian que esas técnicas han sido ampliamente aplicadas en problemas de planificación en sistemas distribuidos y por lo tanto son adecuadas para ser usadas en Clouds [22], [23]. En este trabajo abordamos la planificación de aplicaciones científicas en Clouds federados con el fin de minimizar el tiempo de respuesta y considerando la influencia de las interconexiones y latencias de red entre los diferentes centros de datos. En principio, nuestro enfoque difiere de otros encontrados en la literatura donde los autores no han considerado estrategias basadas en SI a nivel de infraestructura ni tampoco los tres niveles de planificación como se propone en este trabajo. En esta línea, en trabajos previos [4], [12] hemos presentado planificadores basados en SI que operan a nivel de infraestructura. Sin embargo, es importante señalar que en esos trabajos los planificadores operan sólo a dos niveles, los que caracterizan a Clouds compuestos por un único centro de datos.

El resto de los trabajos se centran en un solo nivel, léase [24], [25], [26]. En [24] se resumen algunas políticas de asignación de VMs basadas en programación lineal para diferentes arquitecturas de Clouds federados. Luego, en [25], se describen planificadores a nivel de broker tomando en cuenta diferentes criterios de optimización (por ejemplo, costos monetarios o rendimiento) y diferentes restricciones definidas por los usuarios (por ejemplo, presupuesto, rendimiento esperado, tipo de VMs). Por otra parte, en [26], el planificador propuesto considera el despliegue de VMs de acuerdo a algunas restricciones de ubicación (por ejemplo, Clouds en dónde desplegar las VMs) definidas por el usuario.

Dos trabajos que merecen especial atención son [5], [27]. En [5], se utilizaró a nivel de broker un algoritmo Dijkstra para seleccionar el centro de datos con menor costo monetario y a nivel de infraestructura un algoritmo genético para la asignación de VMs. Aunque en ese trabajo se apunta al nivel de broker y nivel de infraestructura, el objetivo fue reducir los costos monetarios sin tener en cuenta el tiempo de respuesta. Cuando se ejecutan aplicaciones científicas, el tiempo de respuesta es muy importante [10]. Por otra parte, en [27] se propone un planificador ACO para realizar balanceo de carga eficiente de los diversos trabajos entre las VMs. El objetivo de ese trabajo consistió en minimizar el tiempo de respuesta y mejorar el balanceo de carga en las VMs. Este es el único trabajo en la literatura donde se ha considerado el uso de SI en Clouds federados. Sin embargo, es importante señalar que ACO fue implementado a nivel de VM y no a nivel de infraestructura. La razón principal por la cual es importante la planificación a nivel de infraestructura es porque la planificación de VMs –en lugar de planificación de trabajos– en un Cloud determina fuertemente el rendimiento general del sistema [26], es decir, cuanto más eficiente la asignación de VMs en las máquinas físicas, mejor rendimiento es obtenido al ejecutar los trabajos en las VMs ya asignadas.

Con respecto a trabajos que abordan el problema de planificación a nivel de infraestructura utilizando estrategias basadas en SI como en este trabajo, se han encontrado pocos esfuerzos a la fecha [8], [22]. Sin embargo, en estos esfuerzos relacionados es importante tener en cuenta que las técnicas de SI se utilizan para resolver el problema de planificación de trabajos propiamente dicho, es decir, determinan cómo se asignan los trabajos en las VMs, más que atacar el problema de planificación de VMs [8]. En conclusión, los esfuerzos existentes apuntan a Clouds federados tomando en cuenta sólo uno de los niveles sin considerar SI para la asignación de VMs, o apuntan a Clouds de un único centro de datos considerando sólo la planificación de trabajos (y no VM).

IV. PLANIFICADORES PROPUESTOS: DESCRIPCIÓN

El objetivo del planificador propuesto consiste en minimizar el tiempo de respuesta de un conjunto de trabajos de un PSE. El tiempo de respuesta es el período de tiempo

entre que un usuario realiza una solicitud al Cloud y obtiene la respuesta. En otras palabras, es el período de tiempo en el que un usuario solicita un número de VMs para ejecutar su PSE y todos los trabajos del PSE terminan su ejecución.

Un PSE se define formalmente como un conjunto de *N=1, 2, ..., n* trabajos independientes, donde cada trabajo corresponde a un valor particular de una variable del modelo estudiado por el PSE. Los trabajos se distribuyen y ejecutan en *v* VMs solicitadas por el usuario. Con el objetivo de minimizar el tiempo de respuesta, es necesario implementar estrategias que permitan seleccionar centros de datos apropiados donde alojar las VMs. Por ejemplo, el centro de datos más adecuado puede ser aquel que proporcione menor latencia de comunicación cuando un broker consulte acerca de la disponibilidad de recursos físicos. Las latencias se producen debido a retrasos de los paquetes que se envían a través de la red entre la máquina del usuario final y los centros de datos geográficamente distribuidos. Una forma de mitigar los efectos de esas latencias es elegir un centro de datos que opere con una red interna rápida y eficiente, y con mucha capacidad de cómputo.

El planificador propuesto opera en tres niveles. En primer lugar, a nivel de broker se selecciona un centro de datos mediante una política que tenga en cuenta las interconexiones de red y/o latencias de red. En segundo lugar, a nivel de infraestructura, por medio de un planificador de VMs, las VMs del usuario se asignan en las máquinas físicas pertenecientes al centro de datos seleccionado a nivel de broker. Cuando no existen más recursos físicos disponibles en el centro de datos seleccionado donde asignar las VMs, se selecciona un nuevo centro de datos a nivel de broker. Por último, a nivel de VM, se utiliza una política de asignación de trabajos mediante la cual los trabajos del usuario son asignados a las VMs. Actualmente utilizamos la política FIFO, aunque como se mencionó anteriormente otras políticas podrían utilizarse.

A. Planificador a nivel de Broker

El planificador a nivel de broker es ejecutado tanto para seleccionar el primer centro de datos donde asignar las VMs, que luego son gestionadas por el planificador implementado a nivel de infraestructura, como así también cada vez que en un centro de datos no hay más capacidad donde realizar la asignación de las VMs. Actualmente, las políticas implementadas en este nivel son las siguientes:

- Menor-Latencia-Primero (LLTF), mantiene una lista de todos los centros de datos interconectados a través de una red y ordenados por sus latencias en orden ascendente. Cada vez que un usuario requiere un número de VMs para ejecutar su PSE, esta política se encarga de seleccionar en primer lugar el centro de datos con la menor latencia. Luego, cada vez que un centro de datos no posee más recursos físicos donde asignar las VMs, el algoritmo selecciona el siguiente centro de datos con menor latencia.
- Primer-Latencia-Primero (FLTF), selecciona el primer centro de datos de una lista conformada aleatoriamente que contiene todos los centros de datos interconectados a través de una red y a los cuales puede acceder un usuario y asignar sus VMs. Cuando el centro de datos seleccionado no posee más recursos físicos disponibles donde asignar las VMs, el algoritmo selecciona el siguiente centro de datos en la lista.
- Latencias-En-Ronda (LTIR), mantiene una lista de todos los centros de datos interconectados a través de una red, ordenados por sus latencias en orden ascendente, y asigna cada VM requerida por el usuario en un centro de datos de la lista en orden circular.

B. Planificador a nivel de Infraestructura

Para implementar las políticas a nivel de infraestructura, se emplean los algoritmos basados en SI previamente propuestos por los autores en [10]. A continuación se describen estos algoritmos.

En el planificador basado en Optimización por Colonia de Hormigas (ACO), cada hormiga trabaja de manera independiente y representa una VM "buscando" la mejor máquina física donde ser asignada. Cuando se crea una VM en un centro de datos, una hormiga es inicializada y comienza a trabajar. Además, el algoritmo inicializa una tabla principal que contiene información de carga de cada máquina del el centro de datos seleccionado. Posteriormente, si ya existe una hormiga asociada a la VM que está ejecutando el algoritmo, esa hormiga se obtiene a través de un pool de hormigas. Si en cambio la VM no existe en el pool de hormigas, entonces se crea una nueva hormiga. Para ello, se obtiene una lista de las máquinas pertenecientes al centro de datos seleccionado y que son adecuados para asignar la VM. Luego, tanto la hormiga como así también su VM asociada se agregan en el pool de hormigas y la lógica específica del algoritmo ACO comienza a operar (véase el Algoritmo 1).

En cada iteración, la hormiga recolecta información de carga de la máquina que está visitando y agrega esa información a su historial de carga privada. Luego, la hormiga actualiza una tabla de información de carga, que se mantiene en cada máquina, con la carga de las máquinas visitadas (localLoadTable.update()). Esta tabla contiene información de carga de la máquina que está visitando la propia hormiga, como así también información de carga de otras máquinas del centro de datos, que se agregaron a la tabla cuando otras hormigas visitaron la máquina. Aquí, carga se refiere a la utilización total de CPU dentro de una máquina y se calcula teniendo en cuenta el número total de VMs que se ejecutan en cada máquina física en un momento dado. La tabla de información de carga actúa como un rastro de feromonas que dejan las hormigas mientras se mueven a través de las máquinas y sirve para guiar a otras hormigas a que elijan mejores caminos en lugar de vagar al azar en el Cloud. Cada entrada de la tabla local son las máquinas que la hormiga ha visitado en su camino para entregar su VM junto con la información de carga de cada máquina.

Cuando una hormiga se mueve de una máquina a otra tiene dos opciones: mover a una máquina aleatoria usando una probabilidad constante o tasa de mutación (*mutationRate*), o utilizando información de la tabla de carga de la máquina actual (chooseNextStep()). La tasa de mutación disminuye progresivamente con un factor denominado tasa de atenuación

(*decayRate*) a medida que pasa el tiempo. Así, la hormiga cada vez es más dependiente de la información de carga en las máquinas que de una elección al azar. Cuando una hormiga lee la información desde una tabla de carga en una máquina, la hormiga elige la máquina con menor carga en la tabla. Si la carga de la máquina visitada es más pequeña que cualquier otra máquina proporcionada en la tabla de información de carga, entonces la hormiga elige la máquina con la carga más pequeña. Ese proceso se repite hasta que se cumpla algún criterio de terminación. El criterio de terminación es igual a un número predefinido de pasos (*maxSteps*). Por último, la hormiga entrega su VM a la máquina actual –el de menor carga– y finaliza su tarea. Dado que cada paso realizado por una hormiga involucra movimientos a través de la red intra-centro de datos, el algoritmo realiza un control para reducir al mínimo el número de pasos que realiza una hormiga: cada vez que una hormiga visita una máquina que aún no tiene asignada ninguna VM, la hormiga asigna directamente su VM asociada a esa máquina sin realizar más pasos. Cada vez que una hormiga envía un mensaje a través de la red intra-centro de datos para obtener información sobre la disponibilidad de las máquinas, se producen latencias. Cuanto menor cantidad de mensajes se envíen a las máquinas a través de la red, menor será el impacto de las latencias en el tiempo de respuesta al usuario.

Algoritmo 1 Planificador ACO a nivel de infraestructura basado

```
Procedure AntAlgorithm ()
Begin
    Step = 1
    initialize ()
    While ( step < maxSteps) do
        currentLoad = getHostLoadInformation ()
        AntHistory.add (currentLoad)
        localLoadTable.update ()
        if (currentLoad = 0.0)
            break
        else if (random () < mutationRate)  then
            nextHost = randomlyChooseNextStep ()
        else
            nextHost = chooseNextStep ()
        end if
        mutationRate = mutationRate−decayRate
        step = step + 1
        moveTo (nextHost)
    end while
    deliverVMtoHost ()
End
```

La Fig. 2 ilustra un ejemplo en el cual diferentes hormigas asignan sus VM asociadas en un centro de datos compuesto por cinco máquinas: M0, M1, M2, M3, M4.

Cada hormiga realiza un máximo de cuatro pasos para encontrar la mejor máquina donde asignar la VM. Como puede observarse, algunas hormigas ya fueron inicializadas y agregadas al pool, y además ya asignaron sus VMs en algunas de las máquinas del centro de datos. Luego, es el turno de la hormiga *ant_n*, la cual es inicializada en la máquina M3 y también se agregan en el pool. Dado que la carga en la máquina inicial M3 no es cero, la hormiga realiza cuatro pasos para encontrar la máquina con menor carga donde asignar su VM. Antes de realizar el primer paso, la hormiga *ant_n* almacena la información de carga de M3 –carga igual a 0.25– en la tabla de carga privada perteneciente a M3. Los tres primeros pasos realizados por la hormiga son seleccionados de manera aleatoria e ilustrados de color azul en la Fig. 2. Cuando la hormiga realiza el primer paso y visita la máquina M4, la hormiga actualiza la tabla local de M4 con su propia carga, es decir con carga igual a 0.50, y además, la tabla local de M4 contiene información de carga de las máquinas previamente visitadas, es decir, la carga de la máquina M3. De la misma manera, en los pasos siguientes, se actualizan las tablas locales pertenecientes a las máquinas M1 y M2. Finalmente, la hormiga realiza el último paso utilizando la información de carga de su tabla de carga local (método elegirProximoPaso() ilustrado de color rojo en la Fig. 2) de la máquina actual –M2– y decide moverse a M1 (máquina con menor carga) donde la hormiga *ant_n* asigna la VM y finaliza.

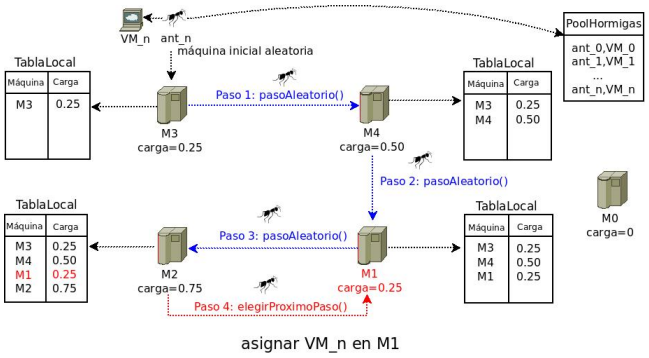


Figura 2. Ejemplo de los pasos llevados a cabo por una hormiga.

Por otra parte, en el planificador basado en Optimización por Enjambre de Partículas (PSO), cada partícula trabaja de manera independiente y representa una VM en busca de la mejor máquina –en el centro de datos seleccionado a nivel de broker– donde ser asignada. Siguiendo la analogía con el ejemplo de las abejas en la sección II, cada VM es considerada una abeja y cada máquina representa ubicaciones en el campo con diferente densidad de flores. Cuando se crea una VM, una partícula es inicializada en una máquina al azar. La densidad de las flores de cada máquina está determinada por su carga.

Durante la búsqueda en el campo de flores, cuanto menor sea la carga en una máquina, mejor será la concentración de flores. Esto significa que la máquina tiene más recursos disponibles para asignar una VM. En el algoritmo (véase el Algoritmo 2), cada vez que un usuario requiere una VM, una partícula es inicializada en una máquina al azar del centro de datos seleccionado (getInitialHost()). Cada partícula en el espacio de búsqueda toma una posición de acuerdo a la carga de la máquina en la cual fue inicializada mediante el método calculateTotalLoad(hostId). Carga se calcula del mismo modo que en ACO. El vecindario de cada partícula está compuesto por las máquinas restantes del centro de datos excluyendo aquella en la cual la partícula fue inicializada. El vecindario de una partícula se obtiene a través del método getNeighbors(hostId,neighborSize). Cada uno de los vecinos –máquinas– que componen el vecindario se seleccionan también al azar. Además, el tamaño de la vecindad de una partícula (*neighborhood*) es un parámetro dado por el usuario.

Algoritmo 2 Planificador PSO a nivel de infraestructura basado

```
Procedure PSOallocationPolicy(vm , hostList)
    Begin
    particle = newParticle(vm , hostList)
    initialHostId = particle . getInitialHost()
    currentPositionLoad = particle.calculateTotalLoad(initialHostId )
    neighbors = particle.getNeighbors(initialHostId, neighborSize )
    While (I < neighbors.size ()) do
        neighborId = neighbors.get (i)
        destPositionLoad = particle.calculateTotalLoad (neighborId )
        if (destPositionLoad = = 0)
            currentPositionLoad = destPositionLoad
            destHostId = neighbors.get(i)
            i=neighbors.size()
        end if
        if (currentPositionLoad − destPositionLoad > velocity)
            velocity = currentPositionLoad − destPositionLoad
            currentPositionLoad = destPositionLoad
            destHostId = neighbors.get(i)
        end if
        i = i +1
    end while
    allocatedHost = hostList.get(destHostId)
    if (!allocatedHost.allocateVM(vm)
        PSOallocationPolicy(vm , hostList)
End
```

En cada iteración del algoritmo, la partícula se mueve a las máquinas vecinos, desde su máquina actual, en busca de una máquina con menor carga. La velocidad de cada partícula está definida por la diferencia de carga entre la máquina en la cual la partícula ha sido asignada con respecto a sus máquinas vecinas. Si alguna de las máquinas en el vecindario tiene una carga menor que la máquina original, entonces la partícula se mueve a la máquina vecina con una velocidad mayor. Teniendo en cuenta que las partículas se mueven a través de las máquinas de su vecindario –en un centro de datos– en busca de una máquina con menor carga, el algoritmo llega a un óptimo local rápidamente. Por lo tanto, cada partícula hace un movimiento desde su máquina asociada hacia uno de sus vecinos que tenga la carga mínima entre todos. Si todos sus vecinos están más ocupados que la propia máquina asociada, la partícula no se mueve de la máquina actual. Por último, la partícula asigna su VM a la máquina con la carga más baja entre sus vecinos y termina su tarea.

Similar a ACO, cada movimiento de una partícula implica el envío de mensajes a través de la red intra-centro de datos. Para reducir el número de movimientos de una partícula, cada vez que una partícula se mueve desde una máquina a un vecino que no posee asignada ninguna VM, la partícula asigna su VM asociada a esa máquina inmediatamente y no realiza más movimientos a las restantes máquinas vecinas. Cuanto menor el número de mensajes enviados por una partícula a las máquinas a través de la red, menor será el impacto de la latencia en el tiempo de respuesta al usuario.

La Fig. 3 ilustra un ejemplo en el cual diferentes abejas (partículas) asignan sus VMs asociadas en un centro de datos compuesto por cinco máquinas: M0, M1, M2, M3 y M4. El tamaño del vecindario explorado por cada abeja es igual a 3 y la densidad de las flores indica la carga de cada máquina. Como puede observarse, algunas abejas ya asignaron sus VMs en las máquinas que componen el centro de datos. Cuando llega el momento de la abeja *bee_n* (inicializada en máquina M1), su vecindario está compuesto por las máquinas M0, M2 y M4. Dado que la máquina inicial M1 posee una carga mayor a cero, es decir que M1 ya tiene algunas VMs ejecutándose, la abeja *bee_n* comienza a explorar su vecindario para buscar una máquina con mejor concentración de flores, es decir, una máquina con menor carga. Para ello, la abeja *bee_n* se traslada a cada una de las máquinas que componen su vecindario en busca de aquel vecino al cual el movimiento se realice con mayor velocidad. Finalmente, la abeja *bee_n* ofrece su VM asociada a la máquina M0 –cuya velocidad V es igual a 0.50– y finaliza su tarea.

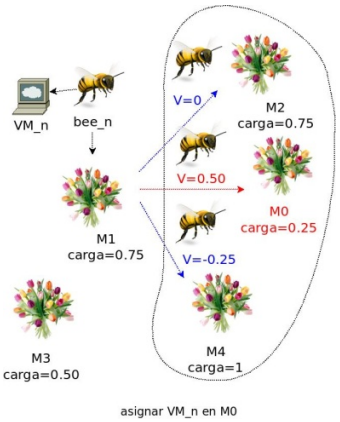


Figura 3. Ejemplo de los pasos llevados a cabo por una partícula.

C. *Planificador a nivel de VM*

Una vez que las VMs se han asignado a los recursos físicos a nivel de infraestructura, el planificador de trabajos procede a asignar los trabajos a las VMs. Este sub-algoritmo utiliza dos listas, una lista con los trabajos que han sido enviados por el usuario, es decir, un PSE, y otra lista con las VMs de los usuarios que ya han sido asignadas a los recursos físicos. A este nivel pueden implementarse diversas políticas clásicas tales como FIFO, o basadas en información de los trabajos como tamaño estimado (que mitiga el conocido "efecto convoy") o prioridad relativa entre trabajos de un PSE [1].

A efectos de nuestra experimentación, el algoritmo itera la lista de los trabajos y los envía a ejecutar en las VMs utilizando una política FIFO. Cada vez que se obtiene un trabajo de la lista se envía a ejecutar a una VM mediante round robin. Internamente, el algoritmo mantiene una cola para cada VM que a su vez contiene una lista de trabajos a ejecutar. El procedimiento se repite hasta que se hayan enviado todos los trabajos para su ejecución. Debido a sus altos requerimientos de CPU, y el hecho de que cada VM requiere sólo un CPU, se asumió un modelo de ejecución de trabajo-VM de 1-1. Esto implica que los trabajos en la cola esperando a ser ejecutados en una VM se ejecutan uno a la vez compitiendo por tiempo de CPU con otros trabajos de otras VMs alojadas en la misma máquina. Así, se utilizó una política de tiempo compartido para la planificación de CPUs para ejecutar trabajos intensivos en uso de CPU de manera equitativa.

V. EVALUACIÓN Y RESULTADOS

Para evaluar la efectividad de los algoritmos hemos procesado un la solución de un PSE real descrito en [11]. Los

detalles sobre la metodología experimental se proporcionan en la subsección V-A. En la subsección V-B se evalúa la performance y escalabilidad de las técnicas a cada nivel del planificador y se presentan los resultados en términos del tiempo de respuesta. Por otro lado, cuanto mayor es el número de máquinas a explorar para consultar su disponibilidad, mayor la cantidad de mensajes a enviar y por lo tanto mayor influencia de las latencias. Por ese motivo es importante evaluar hasta qué cantidad de máquinas es conveniente escalar sin afectar el tiempo de respuesta al usuario.

A. Metodología Experimental

El caso de estudio consiste en un PSE de una placa de deformación plana con un agujero circular central [11]. Las dimensiones de la placa son 18 x 10 m, con $R = 5$ m. Para realizar el estudio paramétrico, se utilizan dos mallas de elementos finitos, una malla 2D y otra 3D, de 1,152 elementos cada una. Para generar los trabajos del PSE, se seleccionó como parámetro de variación la viscosidad del material (η). Luego, se consideraron 25 valores diferentes para η, a saber $x.10^y$ Mpa, con x = 1, 2, 3, 4, 5, 7 e y = 4, 5, 6, 7, más 1.10^8 Mpa. Detalles preliminares sobre la teoría viscoplástica e implementación numérica se pueden encontrar en [11].

Luego, hemos utilizado una máquina real para ejecutar el PSE mediante la variación de η con los valores indicados, y midiendo el tiempo de ejecución de los 25 experimentos diferentes, lo que resultó en 25 archivos de entrada con diferentes configuraciones de entrada y 25 archivos de salida por cada una de las mallas –2D y 3D–. Las pruebas se realizaron usando el software de elementos finitos SOGDE [28]. Una vez que se obtuvieron los tiempos de ejecución en la máquina real, aproximamos para cada experimento el número de instrucciones ejecutadas mediante la siguiente fórmula $NI_i = mipsCPU * T_i$, donde NI_i es el número de millones de instrucciones a ser ejecutadas por el trabajo i, $mipsCPU$ es el poder de procesamiento de la CPU (en MIPS) de nuestra máquina real, y T_i es el tiempo que tomó ejecutar el trabajo i en la máquina real. Por ejemplo, para un trabajo que tomó 539 segundos en ejecutar en una máquina de un poder de procesamiento de 4,008.64 MIPS, el número aproximado de instrucciones fue 2,160,657 MI (Millones de Instrucciones).

Utilizando los datos de los trabajos generados instanciamos CloudSim [14]. El escenario experimental consiste de un Cloud compuesto por 5 centros de datos. La topología de la red utilizada fue generada utilizando el generador de topología de red de la Universidad de Boston dominada topología BRITE [29]. Este generador produce un archivo BRITE que es utilizado por CloudSim y en el cual se definen los diferentes nodos que componen una federación (por ejemplo, centros de datos, brokers) y las interconexiones de red entre ellos. Este archivo se utiliza luego para calcular las latencias producidas por el tráfico de red. Por otra parte, cada centro de datos está compuesto por 10 máquinas físicas. Cada máquina posee las siguientes características: 4,008 MIPS (procesamiento), 4 GBytes (RAM), 400 GBytes (almacenamiento), 100 Mbps (ancho de banda), y 4 CPUs. Además, cada centro de datos posee una latencia externa asociada de 0.8, 1.5, 0.5, 0.15, y 2.8 segundos, respectivamente. Estas latencias han sido asignados tomando en consideración trabajos del área [30], [31].

El objetivo de los experimentos consiste evaluar el efecto en el tiempo de respuesta al aumentar el número de máquinas de cada centro de datos y al enviar a ejecutar un número fijo de trabajos de un PSE. Esta variación en la cantidad de máquinas también se conoce como escalabilidad horizontal. Particularmente, nosotros evaluamos el rendimiento de los algoritmos propuestos variando el número de máquinas por centro de datos en *50*i* máquinas, donde *i = 1, 2, 3, 4, 5, 6.* Además, la cantidad de VMs para ejecutar los experimentos aumenta proporcionalmente al número de máquinas, es decir, la cantidad de VMs varía en *100*i*. Cada VM posee una CPU virtual de 4,008 MIPS, 512 Mbytes de memoria RAM, una imagen 100 Gbytes y un ancho de banda de 25 Mbps. Por otra parte, el conjunto base de trabajos comprendido por los 25 trabajos que se obtuvieron al variar el valor de η, fue clonado para obtener un conjunto más grande de 10,000 trabajos. Cada trabajo se determinó mediante el número de instrucciones a ser ejecutadas, el cual varió entre 244,527 y 469,011 MI para la malla 2D y entre 1,362,938 y 2,160,657 MI para la malla 3D. Además, otro parámetro configurado en CloudSim es el número de elementos de procesamiento (PE) necesarios para procesar cada trabajo. Cada trabajo requiere sólo un PE y además los trabajos son secuenciales (no multi-hilo). Por último, los experimentos tienen archivos de entrada de 93,082 bytes y 291,738 bytes y archivos de salida de 2,202,010 bytes y 5,662,310 bytes para las mallas 2D y 3D, respectivamente.

B. Experimentos Realizados

En esta subsección mostramos los resultados obtenidos al ejecutar el PSE mediante el planificador de tres niveles propuesto y las técnicas asociadas. En los experimentos hemos establecido los parámetros específicos de ACO como: mutationRate = 0.6, decayRate = 0.1 y maxSteps = 8, y el parámetro específico de PSO, es decir neighborhood = 8. Dado que el número de máquinas que componen cada centro de datos es igual a 10 (en el primer conjunto de experimentos), el valor de los parámetros específicos (es decir, maxSteps en ACO y neighborhood en PSO) es igual a 8. Esto significa explorar un porcentaje del 80% del número de máquinas en cada centro de datos. Luego, cuando el tamaño de cada centro de datos se incrementa como *50*i* máquinas, donde *i = 1, 2, 3, 4, 5, 6*, el valor de los parámetros específicos de cada algoritmo también aumenta proporcionalmente de modo de explorar siempre, a medida que aumenta el tamaño del Cloud, el 80% de cada centro de datos.

Las Fig. 4 y 5 muestran los resultados obtenidos por el planificador de tres niveles propuesto al ejecutar un PSE (conjunto de 10,000 trabajos) con las mallas 2D y 3D, respectivamente, a medida que aumenta el tamaño del Cloud. En ambas figuras, cada subfigura representa la combinación de las diferentes políticas consideradas a nivel de broker (LLTF, FLTF, LTIR) con las políticas implementadas a nivel de infraestructura (ACO y PSO).

De las Fig. 4 y Fig. 5, independientemente de la política utilizada a nivel de broker, se observa que PSO es el algoritmo

que produce el menor tiempo de respuesta al usuario con respecto a ACO. Además, dado que ambos algoritmos deben enviar una cierta cantidad de mensajes a través de la red (ver subsección IV-B) para consultar la disponibilidad de las máquinas cada vez que se requiere la asignación de una VM, la cantidad de mensajes afecta directamente el tiempo de respuesta al usuario. Esto se debe a que por cada mensaje enviado a las máquinas, se producen latencias internas en los centros de datos. Por ejemplo, el número de mensajes a enviar por ACO depende del número máximo de pasos –maxSteps– que lleva a cabo una hormiga para asignar su VM asociada. Cuando el número máximo de pasos es igual *p*, ACO envía un máximo de *p* mensajes por asignación de VM. Además, cuando ACO encuentra una máquina sin ninguna carga, asigna la VM y no realiza más pasos. Esto reduce el número total de mensajes de red enviados por ACO. Por otro lado, el número de mensajes de red a enviar por PSO depende del tamaño del vecindario, es decir de la variable neighborhood. Cuando el tamaño del vecindario es igual a v, PSO envía un máximo de v mensajes por asignación de VM. Cuando PSO encuentra una máquina sin carga, asigna la VM directamente. Esto también reduce el número total de mensajes de red enviados.

La razón por la cual, independientemente de la política utilizada a nivel de broker, PSO otorga el menor tiempo de respuesta, es porque se observa que este algoritmo no repite las máquinas visitadas en cada asignación de una VM. Como se explicó en la subsección IV-B2, cada partícula visita cada una de las máquinas de su vecindario, las cuales son diferentes entre sí, en busca de aquella que posea menor carga. Esto incrementa las chances a PSO de encontrar una máquina con carga igual a cero, reduciendo así la cantidad de movimientos totales a realizar. Por otra parte, bajo ACO existe la posibilidad de que una hormiga visite alguna máquina más de una vez, reduciendo así el total de máquinas exploradas. Esto se debe a que ACO utiliza una función aleatoria en sus primeros pasos para elegir la máquina hacia la cual realizará el movimiento, pudiendo repetirse una máquina ya visitada al pasar de una máquina a otra. En el ejemplo de la Fig. 2 (ver subsección IV-B1), cuando la hormiga está en la máquina M4 y va a realizar el *Paso 2*, el cual se realiza aleatoriamente, la hormiga podría volver a visitar la máquina M3 en lugar de una máquina diferente –M1 como ilustra en la Fig. 2–. Además, otra situación en la cual una hormiga puede repetir una máquina ya visitada es cuando las máquinas entre las cuales debe decidir dar un paso poseen cargas iguales. En estos casos la hormiga se mueve hacia alguna de las máquinas de igual carga de manera aleatoria.

Como se puede observar en todas las subfiguras de la Fig.4 y como se resume en la Tabla I, cuando se ejecuta el PSE con la malla 2D y la cantidad de máquinas del Cloud aumenta de 50 a 100 el tiempo de respuesta disminuye. Las ganancias obtenidas por ACO al aumentar la cantidad de máquinas de 50 a 100 son del 36.94%, 21.49% y 16.21%, y las ganancias obtenidas por PSO son del 38.55%, 22.21% y 20.00% cuando se utiliza a nivel de broker las políticas LLTF, FLTF y LTIR, respectivamente. Sin embargo, como se observa en la Fig. 4, cuando la cantidad de máquinas aumenta a más de 100, el tiempo de respuesta comienza a incrementarse cada vez más. El motivo de este incremento en el tiempo de respuesta se debe a que, cuanto mayor es la cantidad de máquinas por centro de datos, mayor es también la cantidad de mensajes a enviar a las máquinas por los algoritmos, y mayor es a su vez la influencia negativa de las latencias internas.

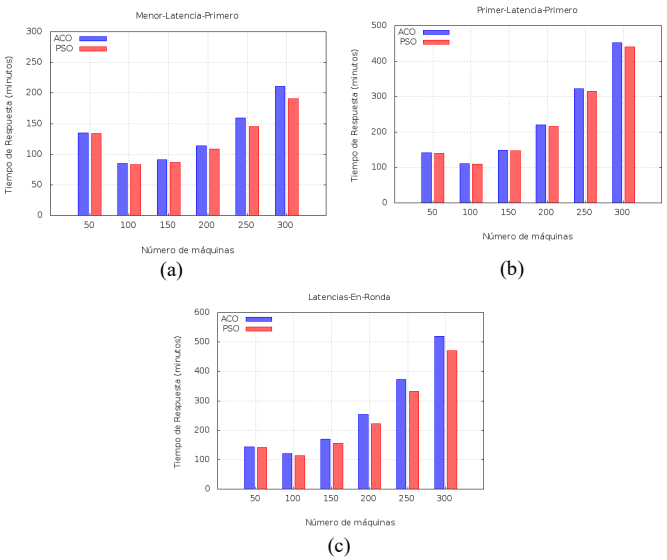


Figura 4. Malla 2D: Tiempo de respuesta a medida que aumenta la cantidad de máquinas.

TABLA I. MALLA 2D: GANANCIAS OBTENIDAS POR ACO Y PSO AL AUMENTAR LA CANTIDAD DE MÁQUINAS.

	LLTF		FLTF		LTIR	
Nº de máquinas	ACO (min)	PSO (min)	ACO (min)	PSO (min)	ACO (min)	PSO (min)
50	134.68	131.89	140.93	138.11	143.49	141.57
100	84.93	81.04	110.64	107.42	120.23	113.26
%Ganancia	36.94	38.55	21.49	22.21	16.21	20.00

Cuando se ejecuta el PSE utilizando la malla 3D (ver subfiguras en la Fig. 5 y Tabla II), el tiempo de respuesta disminuye hasta un incremento en el número de máquinas igual a 200 cuando se utiliza LLTF a nivel de broker (ver subtabla IIa) y hasta 150 cuando se utilizan FLTF y LTIR (ver subtabla IIb). Cuando la cantidad de máquinas aumenta de 50 a 200 y se utiliza LLTF a nivel de broker las ganancias de ACO y PSO son del 59.15% y 59.61%. Por otra parte, las ganancias obtenidas por ACO al incrementar la cantidad de máquinas de 50 a 150 son del 51.03% y 47.96%, y la ganancias obtenidas por PSO son 51.57% y 49.75% al utilizar a nivel de broker las políticas FLTF y LTIR, respectivamente. Al igual que en los experimentos con la malla 2D, cuando el número de máquinas es mayor a 200 máquinas cuando se utiliza LLTF a nivel de broker y mayor a 150 al utilizar FLTF y LTIR, el tiempo de respuesta comienza a incrementarse dado que el mismo está más influenciado por la cantidad de mensajes a través de la red enviado por ACO y PSO.

Como se puede observar también en las Fig. 4 y 5, el menor tiempo de respuesta con ambas mallas (2D y 3D) se obtiene cuando se seleccionan los centros de datos por medio de la política LLTF. Las subtablas IIIa y IVa resumen el menor

tiempo de respuesta alcanzado por cada uno de los algoritmos. Las ganancias obtenidas al ejecutar el PSE con la malla 2D y al utilizar la combinación LLTF–ACO para el nivel de broker e infraestructura (ver subtabla IIIb) fueron del 23.24% y 29.36% con respecto a utilizar FLTF–ACO y LTIR–ACO, respectivamente. Las ganancias de LLTF–PSO fueron del 24.56% y 28.44% con respecto a FLTF–PSO y LTIR–PSO. Por otra parte, cuando se ejecutaron los experimentos con la malla 3D (ver subtabla IVb), las ganancias obtenidas por LLTF–ACO fueron del 24.99% Y 29.63% con respecto a utilizar FLTF–ACO y LTIR–ACO, y las ganancias de LLTF–PSO fueron del 25.29% y 28.29% con respecto a FLTF–PSO y LTIR–PSO. El motivo por el cual la política LLTF a nivel de broker produce los menores tiempos de respuesta se debe a que la mayor cantidad de VMs son asignadas en los centros de datos de menor latencia, las cuales generan un menor impacto en el tiempo de respuesta final.

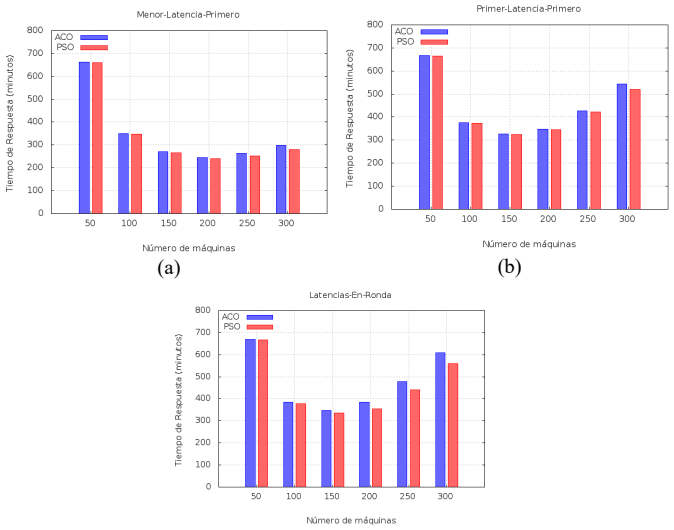


Figura 5. Malla 3D: Tiempo de respuesta a medida que aumenta la cantidad de máquinas.

TABLA II. MALLA 3D: GANANCIAS OBTENIDAS POR ACO Y PSO AL AUMENTAR LA CANTIDAD DE MÁQUINAS.

	LLTF	
Nº de máquinas	ACO (min)	PSO (min)
50	661.26	659.37
200	245.38	240.71
%Ganancia	59.15	59.61

(a) Ganancias obtenidas por LLTF

	FLTF		LTIR	
Nº de máquinas	ACO (min)	PSO (min)	ACO (min)	PSO (min)
50	667.95	665.37	669.97	668.05
150	327.12	322.21	348.68	335.67
%Ganancia	51.03	51.57	47.96	49.75

(b) Ganancias obtenidas por FLTF y LTIR

Podemos concluir que cuando se ejecutan experimentos que requieren de mayor cantidad de cálculo y se hace mayor uso de los procesadores, como los experimentos con la malla 3D, el uso de la red impacta menos en el tiempo de respuesta al usuario que cuando se utilizan trabajos más pequeños (malla 2D). Como se puede observar en los resultados presentados al utilizar la malla 3D, el Cloud puede escalar a un mayor número de máquinas sin afectar el tiempo de respuesta.

TABLA III. MALLA 2D: GANANCIAS AL UTILIZAR LLTF.

	ACO	PSO
FLTF	110.64	107.42
LLTF	84.93	81.04
LTIR	120.23	113.26

(a) Menor tiempo de respuesta obtenido (min)

	LLTF	
	ACO	PSO
FLTF	23.24	24.56
LTIR	29.36	28.44

(b) Ganancia de LLTF con respecto a FLTF y LTIR

TABLA IV. MALLA 3D: GANANCIAS AL UTILIZAR LLTF.

	ACO	PSO
FLTF	327.12	322.21
LLTF	245.38	240.71
LTIR	348.68	335.67

(a) Menor tiempo de respuesta obtenido (min)

	LLTF	
	ACO	PSO
FLTF	24.99	25.29
LTIR	29.63	28.29

(b) Ganancia de LLTF con respecto a FLTF y LTIR

VI. CONCLUSIONES Y TRABAJO FUTURO

Un tipo de experimento científico muy popular son los PSEs, los cuales involucran ejecutar muchos trabajos intensivos en uso CPU de manera independiente. Estos trabajos deben ser planificados de manera eficiente en diferentes recursos de cómputo de un entorno distribuido, tales como los proporcionados por un Cloud. Particularmente, los Clouds federados ofrecen potencialmente una gran cantidad de recursos a los usuarios, especialmente cuando el número de VMs requerida por un usuario excede el máximo que puede ser proporcionado por un único proveedor o centro de datos. Luego, la planificación de trabajos/VMs juega un papel fundamental en Clouds federados.

Recientemente, los algoritmos inspirados en SI han recibido una creciente atención en la comunidad de investigación en Cloud para hacer frente a la planificación de trabajos y VMs. Particularmente, en este trabajo hemos descrito dos algoritmos basados en ACO y PSO para la asignación eficiente de las VMs en las máquinas físicas, en combinación con tres estrategias –LLTF, FLTF y LTIR– que consideran información de red para la selección de los centros de datos. Luego, para mapear los trabajos del PSE en las VM se implementa una política FIFO. Estas técnicas se enmarcan en un planificador genérico de trabajos para Cloud federados de tres niveles. Los experimentos simulados realizados con CloudSim y datos de trabajos de un PSE real sugieren que el ensamble de las estrategias propuestas, en un planificador de tres niveles, mejoran el tiempo de respuesta al usuario al incrementar el tamaño del Cloud. Sin embargo, cuando se consideran latencias de red es necesario realizar un análisis de hasta dónde es conveniente incrementar la cantidad de máquinas sin afectar el tiempo de respuesta. En los

experimentos realizados se evidencia que cuando PSO y ACO se combinan con LLTF a nivel de broker, el tiempo de respuesta es el más bajo con respecto a FLTF y LTIR, siendo LTIR el más influyente en el tiempo de respuesta.

El presente trabajo se está extendiendo en varias direcciones. En primer lugar se enriquecerá al planificador propuesto mediante nuevas estrategias que operen en los diferentes niveles. A nivel de broker, por ejemplo, puede ser deseable que las estrategias obtengan información de cuál es el centro de datos que cuenta con una mayor cantidad de máquinas disponibles, posibilitando la creación de la mayor cantidad de VMs de un usuario posible en un mismo centro de datos. Esto permitirá disminuir así las comunicaciones de red externas entre las diversas VMs. Otra característica a considerar en este nivel es cuando los usuarios están más interesados en reducir los costos monetarios para el caso de Cloud pagos. A nivel de infraestructura, planeamos considerar otras técnicas bio-inspiradas, tales como Colonias de Abejas Artificiales (ABC), las cuales son también ampliamente utilizadas para resolver problemas de optimización combinatoria. Además, otra cuestión que merece atención es considerar otros escenarios Cloud [16] con recursos físicos heterogéneos que pertenezcan a diferentes proveedores Cloud. Debido a la existencia de múltiples usuarios, en Clouds es necesario establecer mecanismos distribuidos de asignación de recursos a trabajos/VMs de usuarios independientes, maximizando la equitatividad. Sin embargo, los planificadores actuales pueden dar un tiempo global de respuesta promedio alto pero ciertos desvíos que benefician más a un usuario que a otro. Para ello, se pondrá en práctica un planificador Cloud basado en técnicas de SI con el objetivo de realizar una planificación justa de los trabajos/VMs de los usuarios.

Finalmente, otro aspecto que merece especial atención es direccionar la manera en que los trabajos se asignan en las VMs. Actualmente, los trabajos se asignan a las VMs en base a políticas que asumen trabajos independientes sin plazos de ejecución. Sin embargo, para obtener una mayor eficiencia es necesario tomar en consideración características específicas de la aplicación científica que se va a ejecutar. Un ejemplo son aplicaciones de bolsas de tareas –bag-of-task– con restricciones de plazos [32] ó aplicaciones científicas [33] en áreas como la bioinformática y la astronomía, que requieren procesamiento de workflows en los cuales los trabajos se ejecutan basados en sus dependencias.

AGRADECIMIENTOS

Los autores agradecemos el soporte financiero proporcionado por la ANPCyT a través de los proyectos PICT-2012-0045, PICT-2012-2731 y PICT-2014-1430, y a la Universidad Nacional de Cuyo a través del proyecto 06B/308.

REFERENCIAS

[1] C. Mateos, E. Pacini, and C. García Garino, “An ACO-inspired Algorithm for Minimizing Weighted Flowtime in Cloud-based Parameter Sweep Experiments,” Advances in Engineering Software, vol. 56, pp. 38–50, 2013.

[2] V. Mauch, M. Kunze, and M. Hillenbrand, “High performance Cloud computing,” Future Generation Computer Systems, vol. 29, no. 6, pp. 1408 – 1416, 2013.

[3] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, “Virtual machine provisioning through satellite communications in federated Cloud environments,” Future Generation Computer Systems, vol. 28, no. 1, pp. 85–93, 2012.

[4] E. Pacini, C. Mateos, and C. García Garino, “SI-based Scheduling of Parameter Sweep Experiments on Federated Clouds,” in First HPCLATAM-CLCAR Joint Conference Latin American High Performance Computing Conference (CARLA), vol. 845 of High Performance Computing. Communications in Computer and Information Science, pp. 28–42, Springer, 2014.

[5] L. Agostinho, G. Feliciano, L. Olivi, E. Cardozo, and E. Guimaraes, “A Bio-inspired Approach to Provisioning of Virtual Resources in Federated Clouds,” in Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), DASC 11, (Washington, DC, USA), pp. 598–604, IEEE Computer Socienty, 12-14, 2011.

[6] G. Woeginger, “Exact algorithms for NP-Hard problems: A survey,” in Combinatorial Optimization - Eureka, You Shrink! (M. Junger, G. Reinelt, and G. Rinaldi, eds.), vol. 2570 of Lecture Notes in Computer Science, pp. 185–207, Springer, 2003.

[7] J. Kennedy, “Swarm Intelligence,” in Handbook of Nature-Inspired and Innovative Computing (A. Zomaya, ed.), pp. 187–219, Springer, 2006.

[8] E. Pacini, C. Mateos, and C. García Garino, “Distributed job scheduling based on Swarm Intelligence: A survey,” Computers & Electrical Engineering, vol. 40, no. 1, pp. 252–269, 2014. 40th-year commemorative issue.

[9] E. Pacini, C. Mateos, and C. García Garino, “Multi-objective Swarm Intelligence schedulers for online scientific Clouds,” Special Issue on Cloud Computing. Computing, pp. 1–28, 2014.

[10] E. Pacini, C. Mateos, and C. García Garino, “Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization,” Advances in Engineering Software, vol. 84, pp. 31–47, 2015.

[11] C. García Garino, M. Ribero Vairo, S. Andía Fagés, A. Mirasso, and J.-P. Ponthot, “Numerical simulation of finite strain viscoplastic problems,” Journal of Computational and Applied Mathematics, vol. 246, pp. 174–184, July 2013.

[12] E. Pacini, C. Mateos, and C. García Garino, “Dynamic scheduling of scientific experiments on clouds via ant colony optimization,” in International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG 2013) (B. H. V. Topping and P. Iványi, eds.), (Pécs, Hungary), pp. 1–21, Civil-Comp Press, 2013.

[13] R. Costa, F. Brasileiro, G. Lemos, and D. Sousa, “Analyzing the impact of elasticity on the profit of cloud computing providers,” Future Generation Computer Systems, vol. 29, no. 7, pp. 1777–1785, 2013.

[14] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, “Cloudsim: A toolkit for modeling and simulation of Cloud Computing environments and evaluation of resource provisioning algorithms,” Software: Practice & Experience, vol. 41, no. 1, pp. 23–50, 2011.

[15] D. Marinescu, “Cloud computing,” in Cloud Computing, pp. 1–396, Boston: Morgan Kaufmann, 2013.

[16] R. Moreno Vozmediano, R. Montero, and I. Llorente, “IaaS Cloud architecture: FromvVirtualized datacenters to federated Cloud infrastructures,” IEEE Computer, vol. 45, no. 12, pp. 65–72, 2012.

[17] L. Wang, M. Kunze, J. Tao, and G. von Laszewski, “Towards building a Cloud for scientific applications,” Advances in Engineering Software, vol. 42, no. 9, pp. 714–722, 2011.

[18] R. Tavares Neto and M. Godinho Filho, “Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research,” Engineering Applications of Artificial Intelligence, vol. 26, no. 1, pp. 150–161, 2013.

[19] E. Mahdiyeh, S. Hussain, K. Mohammad, and M. Azah, “A survey of the state of the art in Particle Swarm Optimization,” Research journal of Applied Sciences, Engineering and Technology, vol. 4, no. 9, pp. 1181–1197, 2012.

[20] U. Singha and S. Jain, “An analysis of swarm intelligence based load balancing algorithms in a cloud computing environment,” International Journal of Hybrid Information Technology, vol. 8, no. 1, pp. 249–256, 2015.

[21] A. Beegom and M. Rajasree, “A particle swarm optimization based pareto optimal task scheduling in cloud computing,” in Advances in Swarm Intelligence, vol. 8795 of Lecture Notes in Computer Science, pp. 79–86, Springer International Publishing, 2014.

[22] Z. Zhan, X. Liu, Y. Gong, J. Zhang, H. Chung, and Y. Li, “Cloud computing resource scheduling and a survey of its evolutionary approaches,” ACM Computing Surveys, vol. 47, no. 4, pp. 63:1–63:33, 2015.

[23] S.J.Mohana, M.Saroja, and M.Venkatachalam, “Comparative analysis of swarm intelligence optimization techniques for cloud scheduling,”

International Journal of Innovative Science, Engineering & Technology, vol. 1, no. 10, pp. 15–19, 2014.
[24] M. Gahlawat and P. Sharma, “Survey of virtual machine placement in federated Clouds,” in International Advance Computing Conference (IACC), pp. 735–738, IEEE, 2014.
[25] J. Lucas-Simarro, R. Moreno-Vozmediano, R. Montero, and I. Llorente, “Scheduling strategies for optimal service deployment across multiple clouds,” Future Generation Computer Systems, vol. 29, no. 6, pp. 1431–1441, 2013. Including Special sections: High Performance Computing in the Cloud & Resource Discovery Mechanisms for P2P Systems.
[26] J. Tordsson, R. Montero, R. Moreno Vozmediano, and I. Llorente, “Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers,” Future Generation Computer Systems, vol. 28, no. 2, pp. 358 – 367, 2012.
[27] G. de Oliveira, E. Ribeiro, D. Ferreira, A. Araújo, M. Holanda, and M. Walter, “ACOsched: a scheduling algorithm in a federated Cloud infrastructure for bioinformatics applications,” in International Conference on Bioinformatics and Biomedicine, pp. 8–14, IEEE, 2013.
[28] C. García Garino, F. Gabaldón, and J. M. Goicolea, “Finite element simulation of the simple tension test in metals,” Finite Elements in Analysis and Design, vol. 42, no. 13, pp. 1187–1197, 2006.
[29] J. Jung, S. Jung, T. Kim, and T. Chung, “A study on the Cloud simulation with a network topology generator,” World Academy of Science, Engineering & Technology, vol. 6, no. 11, pp. 303–306, 2012.
[30] S. Malik, F. Huet, and D. Caromel, “Latency based group discovery algorithm for network aware Cloud scheduling,” Future Generation Computer Systems, vol. 31, pp. 28 – 39, 2014.
[31] T. Somasundaram and K. Govindarajan, “CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science Cloud,” Future Generation Computer Systems, vol. 34, pp. 47 – 65, 2014.
[32] K. Kim, R. Buyya, and J. Kim, “Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters,” in Seventh IEEE International Symposium on Cluster Computing and the Grid, pp. 541–548, IEEE Computer Society, May 2007.
[33] E. Deelman, D. Gannon, M. Shields, and I. Taylor, “Workflows and e-Science: An overview of workflow system features and capabilities,” Future Generation Computer Systems, vol. 25, no. 5, pp. 528–540, 2009.

Elina Pacini received her Ph.D. degree in Computer Science from the UNICEN, in 2014, and a BSc. in Information Systems Engineering from the UTN-FRM, in 2005. She is a Teacher Assistant at the UNCuyo and member of the ITIC and the CONICET. She is interested in job and VM scheduling, Cloud Computing and scientific applications.

Cristian Mateos received his Ph.D. degree in Computer Science from the UNICEN, in 2008, and his M.Sc. in Systems Engineering in 2005. He is a full time Teacher Assistant at the UNICEN and member of the ISISTAN and the CONICET. He is interested in parallel/distributed programming, Grid middlewares, Service-oriented and Mobile Computing.

Carlos García Garino received his Ph.D. degree from Universidad Politécnica de Cataluña, Spain, in 1993, and the Civil Engineering degree from UBA, Argentina, in 1978. He is a full Professor at the UNCuyo, and director of the ITIC–UNCuyo. He is interested in Computer Networks, Distributed Computing and Computational Mechanics.