**Desarrollo y Análisis de Resultados**

Para el desarrollo y análisis de los resultados de la investigación, se utilizó la versión 7.0 del CloudSim, en el que se simularon varios entornos de ejemplos de código para un proveedor de infraestructura como servicio.

Entre los ejemplos que se utilizaron, me gustaría destacar 2, que a mi parecer son los más interesantes ya que poseen estas características propias de un IaaS.

**Los Ejemplos son:**

**Ejemplo 1:**

Se implementa una simulación con características propias de una infraestructura como servicio (IaaS), ya que modela recursos físicos (hosts, almacenamiento, ancho de banda) y su uso por máquinas virtuales (VMs) y tareas (Cloudlets).

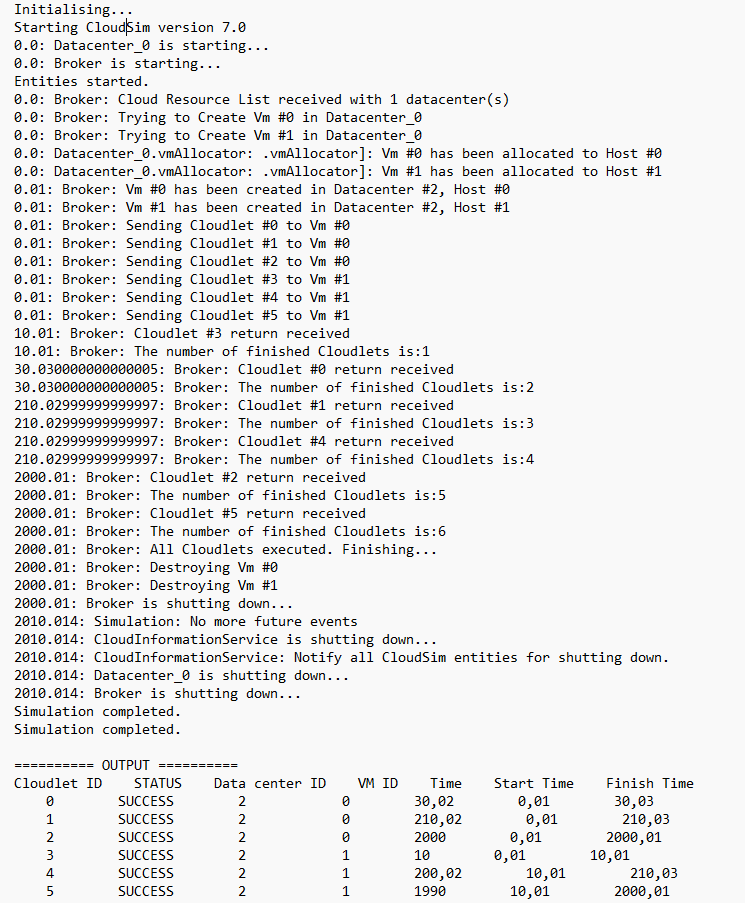
En particular, tiene los siguientes elementos clave de un modelo IaaS:

* **Host físico:** Se definen hosts con características específicas como CPU, RAM, almacenamiento, y ancho de banda.
* **Máquinas virtuales (VMs):** Estas son instancias virtualizadas que los hosts alojan y utilizan para ejecutar tareas.
* **Cloudlets (tareas):** Representan las aplicaciones que se ejecutan en las VMs.

Además, integra el concepto de red mediante la clase NetworkDatacenter, y la configuración de switches, añadiendo funcionalidad para manejar la transferencia de datos entre VMs.

* **Broker:** Actúa como intermediario entre los usuarios (simulados) y el datacenter, gestionando la asignación de tareas a las VMs.

A continuación, la respuesta de la consola al ejecutar el ejemplo:

****

**Inicialización:**

Por lo que se puede ver en este ejemplo, el CloudSim y el broker se inicializan. Se crea un datacenter (Datacenter\_0) con múltiples hosts (máquinas físicas), que son los proveedores de recursos.

En el ejemplo, el datacenter tiene dos hosts, cada uno asignado a una VM.

Se crean dos máquinas virtuales (VM #0 y VM #1) con configuraciones específicas:

VM #0 utiliza un planificador de tareas en tiempo compartido (time-shared).

VM #1 utiliza un planificador en espacio compartido (space-shared).

**Creación y asignación de Cloudlets:**

Se crean seis tareas (Cloudlets), cada una con diferentes requisitos de procesamiento (longitud). Estas tareas se asignan a las VMs:

Los Cloudlets #0, #1, y #2 se asignan a VM #0.

Los Cloudlets #3, #4, y #5 se asignan a VM #1.

**Ejecución:**

Se inician las VMs en los hosts correspondientes, y los Cloudlets son enviados a cada VM.

El planificador de cada VM define cómo se ejecutan los Cloudlets:

En tiempo compartido, los recursos se dividen entre las tareas, permitiendo que varias tareas progresen simultáneamente.

En espacio compartido, cada tarea usa todos los recursos hasta que termina, ejecutándose una por una.

**Resultados:**

**Cloudlets 0-2 (en VM #0):**

Se ejecutan de forma simultánea gracias al planificador time-shared, pero su tiempo de finalización varía dependiendo de su longitud (10,000; 100,000; 1,000,000 instrucciones).

**Cloudlets 3-5 (en VM #1):**

Se ejecutan uno por uno debido al planificador space-shared. Por ejemplo:

El Cloudlet #3 termina en 10.01, y el Cloudlet #4 comienza justo después.

La diferencia en los tiempos de ejecución (30, 210, 2000, etc.) refleja los diferentes tamaños y prioridades de los Cloudlets, así como las políticas de planificación de las VMs.

**Ejemplo 2:**

**Inicialización de CloudSim**: Se crea un objeto CloudSim que sirve como núcleo para manejar los eventos y la simulación.

**Creación del Datacenter**:

Un datacenter representa la infraestructura física que ofrece recursos. Aquí se crean dos hosts (servidores físicos), cada uno con:

* + - 8 CPUs.
    - 16 GB de RAM.
    - 1 millón de unidades de almacenamiento.
    - También se especifican costos asociados a la utilización de los recursos del datacenter (CPU, memoria, almacenamiento, y ancho de banda).

**Creación del Broker**:

El broker actúa como intermediario entre los usuarios (que envían tareas) y el datacenter. Se encarga de asignar tareas a las máquinas virtuales (VMs) y comunicarse con el datacenter.

**Creación de Máquinas Virtuales (VMs)**:

Se crean 4 VMs, cada una con:

* + - 2 CPUs.
    - 4 GB de RAM.
    - 10 GB de almacenamiento.
    - Estas VMs representan los recursos virtualizados que el datacenter asigna a los usuarios.

**Creación de Tareas (Cloudlets)**:

Se crean 6 tareas (cloudlets) que necesitan ser procesadas. Cada tarea tiene:

* + - 10,000 instrucciones.
    - Un archivo de entrada y salida de 300 unidades de tamaño.
    - Las tareas representan aplicaciones o cargas de trabajo que los usuarios desean ejecutar.

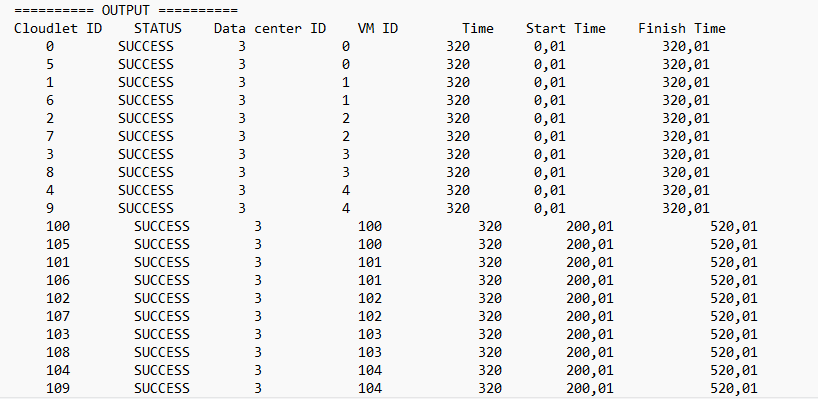
**Ejecución de la Simulación**:

Se inicia la simulación, permitiendo que el broker asigne las tareas (cloudlets) a las máquinas virtuales y coordine su ejecución.

**Visualización de Resultados**:

Al finalizar la simulación, se muestran los resultados en una tabla, incluyendo métricas como:

* + - El tiempo de inicio y finalización de las tareas.
    - El estado de las tareas (éxito o falla).
    - El VM en el que se ejecutó cada tarea.



Se mostrará solo la tabla final debido a que la ejecución de este ejemplo es muy extensa, pero con la explicación previa y los resultados de la imagen queda claro el objetivo del código.

**Conclusión**

Para concluir con una parte más generalizada de los resultados tras el análisis de los ejemplos, se puede decir que, en el caso del ejemplo 1, el código simula el comportamiento de una infraestructura en la nube, donde se ejecutan cloudlets (trabajos) sobre máquinas virtuales (VMs) que están alojadas en un centro de datos.El código crea una serie de cloudlets y asigna su ejecución a las VMs de acuerdo con estos modelos de asignación.

En caso del ejemplo 2, es mas desarrollado, por eso precisamente escogí estos 2 ejemplos comparándolos, en el ejemplo 1, se utilizaba un DatacenterBroker directamente en la simulación, lo que es más común y directo. En este ejemplo, se introduce un nuevo concepto: el GlobalBroker. Este actúa como un intermediario entre el sistema principal de simulación y el broker que gestiona las máquinas virtuales (VMs) y los cloudlets. La creación de este GlobalBroker se maneja como una entidad separada y sus eventos se gestionan mediante el método processEvent.

Esto marca una diferencia porque el flujo de trabajo está más descentralizado y el GlobalBroker coordina la creación de un broker normal y la asignación de recursos. Este enfoque puede ser útil en simulaciones más complejas donde se necesitan intermediarios o gestores globales.

En el código anterior, las máquinas virtuales (VMs) y los cloudlets se creaban directamente en el método main y se enviaban al broker inmediatamente después de su creación.

En este código, el proceso de creación de VMs y cloudlets se realiza mediante el GlobalBroker, que es quien organiza y delega la creación de estas entidades. El broker es creado de forma dinámica dentro del GlobalBroker, lo que da un enfoque más flexible y modular.

Este cambio introduce una abstracción adicional, lo que facilita la adaptación del sistema a situaciones más complejas donde varios brokers pueden estar involucrados o gestionados por una capa de administración global. En este otro código, el GlobalBroker gestiona eventos. El método processEvent maneja el evento de creación del broker, donde se crean las VMs y los cloudlets y se envían al broker, y se reanuda la simulación, este patrón permite una mayor flexibilidad y control sobre el ciclo de vida de la simulación, permitiendo la inyección de otros eventos en el futuro (si se desea, como la reconfiguración del sistema, la creación de más recursos, etc.).

**Estructura del Broker:**

El GlobalBroker actúa como una capa adicional sobre el DatacenterBroker, lo que permite la creación dinámica del broker y su integración con el sistema. Además, el GlobalBroker extiende SimEntity, lo que le permite interactuar con el sistema de simulación como una entidad propia, gestionando eventos específicos y la creación de brokers.

Esto da como resultado una simulación que puede ser más fácilmente escalable o modular, permitiendo la incorporación de nuevos brokers en tiempo de ejecución.

La creación de cloudlets y VMs es bastante similar en ambos códigos, pero en este ejemplo, la cantidad de VMs y cloudlets se pasa como parámetros más dinámicos a través del GlobalBroker en lugar de estar fijada de manera más estática.

**Estructura del Datacenter:**

En ambos ejemplos, se crea un Datacenter con un conjunto de Hosts y PEs (procesadores). Sin embargo, este código tiene dos centros de datos (Datacenter\_0 y Datacenter\_1), lo que agrega más flexibilidad al simular una configuración de nube distribuida o multi-datacenter.

Además, se mantiene la estructura detallada de cómo se crean los PEs (procesadores) y los Hosts, pero con la capacidad de usar múltiples datacenters, lo que permite una simulación más rica de la distribución y gestión de recursos, se especifican propiedades adicionales como costos de memoria, almacenamiento y ancho de banda en la creación del Datacenter, lo que permite a la simulación gestionar estos aspectos económicos en su flujo de trabajo.