SISTEMAS BASADOS EN GRID

Sistemas Operativos: Grupo 6

Semestre 2021-2

Cabrera Pérez Oswaldo y Hanna Sophia Ceres Martinez

¿Qué es un Grid?

- Un Grid es una infraestructura de recursos que se virtualiza.
- Conecta diversos recursos usando hardware y software que lo coordina y administra.
- Permite la computación distribuida mediante una red de recursos.

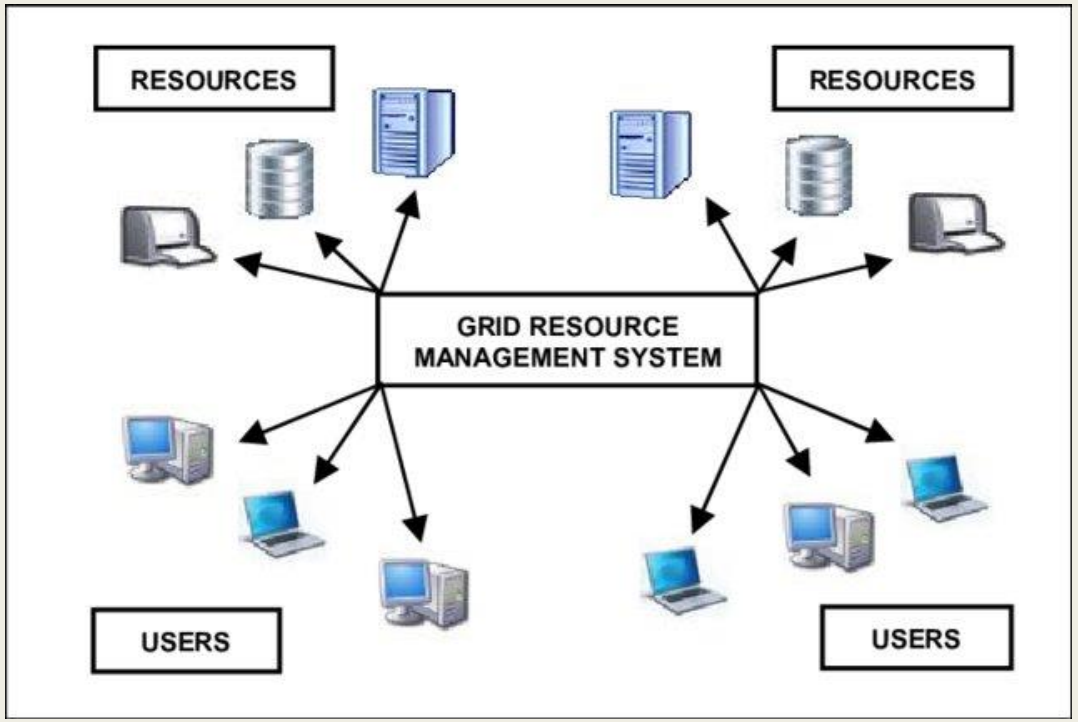


Figura: Imagen de ResearchGate

Características

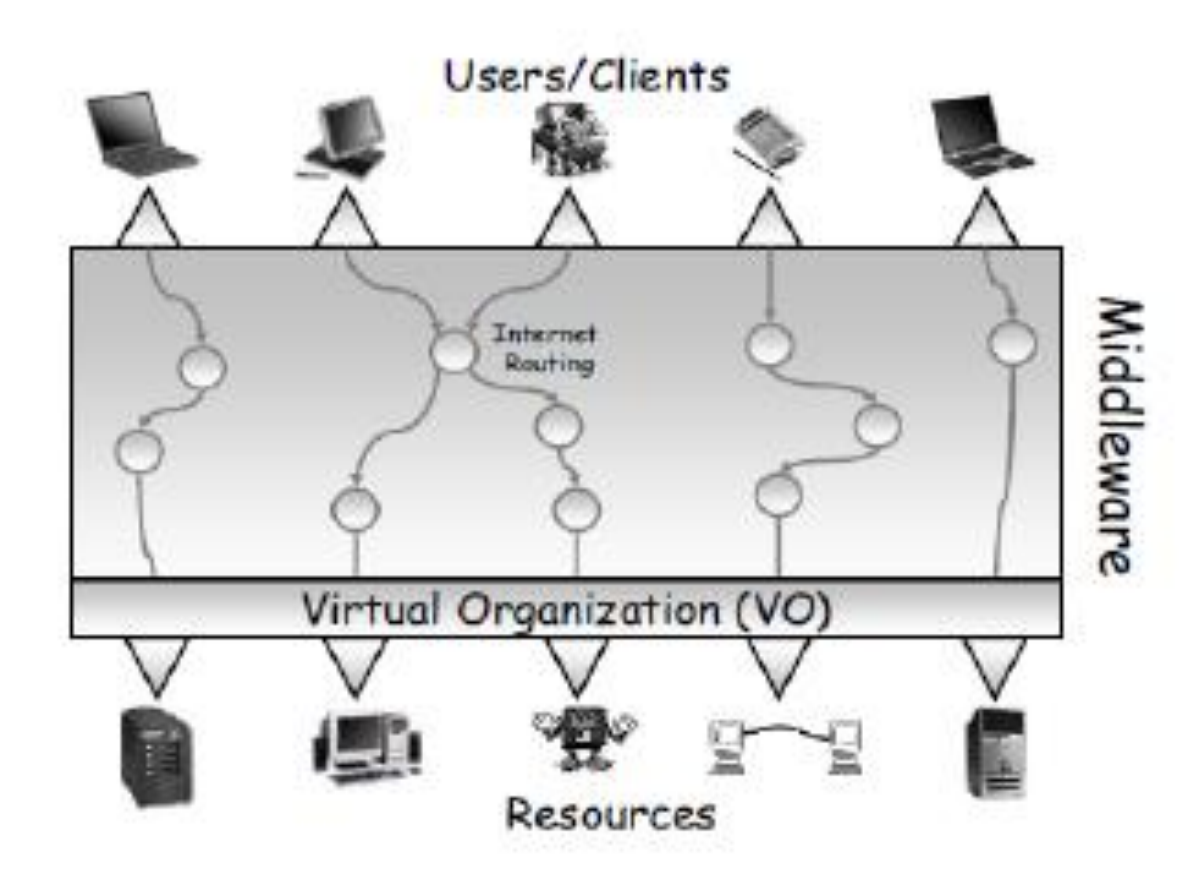
- Los recursos no comparten un reloj físico
- Los recursos no comparten memoria
- Los recursos están geográficamente separados
- Autonomía y heterogeneidad en los recursos
- Coordina recursos que no están sujetos a un control centralizado
- Uso de protocolos e interfaces abiertos, estándares y de uso general
- Calidad del Servicio

Tipos de Grid

- Grid computacional
- Grid de datos
- Grid de Servicios

Arquitectura

- Open Grid Service Architecture (OGSA)



¿CÓMO FUNCIONA? ALGORITMOS Y PROTOCOLOS

Seguridad

- OGSA
- Protocolos Web
- Autoridad de Registro (RA)
- Criptografía
  - *Llave pública y llave privada*

Extremo del usuario

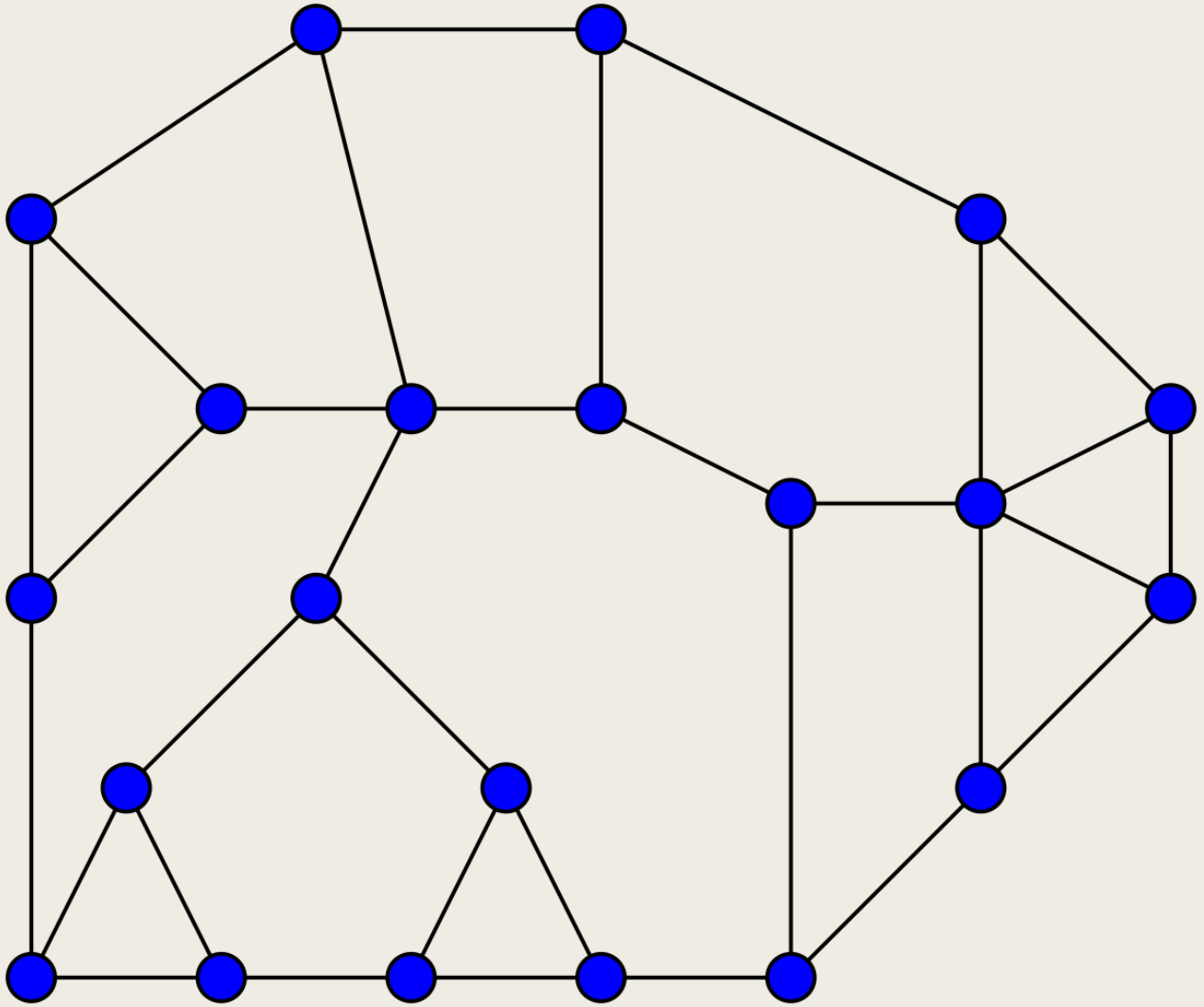
```
1   Proceso ServiceUserEnd
2       GenerarSolicitud() //Generar la solicitud para establecer
3       //el enlace de comunicación con RA vía OGSA
4       ProporcionarCredenciales()
5       respuesta = EsperarRespuestaRA()
6       Si respuesta = "Acceso autorizado" Entonces
7           Escribir "Acceso autorizado"
8           UtilizarGrid()
9       SiNo
10          Escribir "Acceso no autorizado"
11      Fin Si
12  FinProceso
```

Extremo del Proveedor

```
1   Proceso ServiceProviderEnd
2       Si HaySolicitud Entonces
3           IrPaso7
4       SiNo
5           IrPaso2
6       Fin Si
7   Credenciales = RecibirCredenciales //identity cre
8   Si VerificarCredenciales(Credenciales) Entonces
9       Escribir "Acceso Autorizado"
10  SiNo
11      Escribir "Acceso no Autorizado"
12  Fin Si
13  FinProceso
```

Comunicación: Brodcast y Multicast

- Cada recurso/procesador que sea parte del grid se comunica con otros enviando mensajes entre sí.



Protocolo

- Procesador P transmite un mensaje
- -El mensaje del procesador P es recibido sin corromperse por el procesador Q
- -Procesador Q incluye un reconocimiento positivo para los mensajes de P en los siguientes mensajes de Q
- -Procesador R recibe los mensajes de Q y es consciente de que los mensajes de P han sido reconocidos y no hay necesidad de que R los reconozca en el siguiente mensaje, R reconoce el mensaje de Q
- -Si el procesador R no ha recibido el mensaje de P, el mensaje de Q alerta a R de esta pérdida, por lo siguiente R incluye un reconocimiento negativo para los mensajes de P en el siguiente mensaje de R

Exclusión mutua

- Debido a que los recursos se comparten, hay que asegurarnos de algunos aspectos como:
  - *Un recurso no puede usarse al mismo tiempo*
  - *La información puede consultarse al mismo tiempo pero no su edición.*
  - *Un nodo puede realizar una tarea a la vez*

Algoritmo de la Panadería (Leslie Lamport)

```
1  Proceso BakeryAlgorithm
2      Dimension  Seleccionando[100]
3      Dimension Ticket[100]
4      Ticket[i] = 0
5      Seleccionando[i] = Falso
6
7      //Entry section
8      Seleccionando[i] = Verdadero
9      Ticket[i] = Maximo(Tickets) + 1
10     Seleccionando[i] = Falso
11     Para j=0 Hasta n-1 Hacer //n = total de procesadores excepto el que esté ejecutando las instrucciones
12         Mientras !Seleccionando[j] = Falso Hacer
13         FinMientras
14         Mientras Ticket[j]=0 o dupla(Ticket[j] ,j) > dupla(Ticket[i],i) Hacer
15         Fin Mientras
16     Fin Para
17     //Se entra a la Sección Crítica
18     //Se sale de la zona Crítica
19     Ticket[i]=0
20 FinProceso
```

Código para el procesador p_i, para $0 < i < n-1$, n=total de procesadores

Sincronización de relojes

* Cada recurso tiene un reloj interno físico propio.

* La falta de un reloj global, el hecho de que cada reloj da marcas de tiempo a diferentes velocidad e incluso causas físicas puede ocasionar problemas como que los mensajes que se intercambian parezcan que vienen del pasado o futuro.

* Hay que sincronizar usando una fuente externa.

Algoritmo

```
1   Proceso ClackSynchronization
2       Dimension diff[i]=0
3
4       EnviarHC
5
6       Mientras MensajeRecibido Hacer
7           diff[j] = T + d - u/2 - HC
8           Si MensajesCompletos Entonces
9               adj = Promedio_diff
10          Fin Si
11      Fin Mientras
12  FinProceso
```

Código para el procesador p_i, para $0 < i < n-1$, n=total de procesadores

Selección

Cuando se trabaja en equipo o en conjunto para resolver algún problema en particular, uno de los aspectos clave para lograrlo es la elección de un líder para poder coordinar y asignar las tareas a los demás participantes.

Algoritmo

```
1   Funcion eleccionDeLider
2       nodoLider=enviarMensajeEleccion(nodosVecinos)
3       Broadcast(nodoLider)
4   Fin Funcion
5
6   Algoritmo seleccionLider
7
8       participacion = Falso
9       mensaje = Falso
10      Si RecibiMensaje y !participacion y !mensaje Entonces
11          mensaje = Verdadero
12          participacion=Verdadero
13          nodoPadre = mensajeTrasmisor
14          mejorNodo = enviarMensajeEleccion(nodosVecinos - nodoPadre)
15      SiNo
16          confirmarRecepcion
17          Mientras RecibirConfirmaciones Hacer
18          Fin Mientras
19          enviarConfirmacionPadre
20          enviarInfoRecursos
21          enviarMejorNodo
22      Fin Si
23
24  FinAlgoritmo
```

Consenso

El problema del consenso en un sistema consiste en que todos los procesadores coincidan en un mismo valor o en una decisión.

Problemas de los Generales Bizantino

Existe un grupo de generales de un ejército bizantino que rodean una ciudad enemiga, cada general está al cargo de una tropa existen solo dos órdenes: atacar o retirarse. Los comandantes se comunican mediante mensajes, pero existe la sospecha de que algunos de ellos sean traidores, alguno podría enviar un mensaje diciendo que su tropa atacará cuando realmente se retirará, poniendo en desventaja a las demás tropas y aumentando sus posibilidades de fracaso.

Fallas Bizantinas

- Fallar al momento de realizar una tarea y dejar de trabajar
- Fallar al momento de mandar un resultado
- Responder con un resultado incorrecto
- Responder con un resultado incorrecto a propósito

El algoritmo de Tolerancia práctica a fallas bizantinas (pBFT por sus siglas en inglés)

- El cliente envía su petición al nodo líder
- El nodo líder hace un broadcast de la petición a los nodos secundarios (o de respaldo)
- Todos los nodos (líder y de respaldo) realizan la tarea o servicio y envían su resultado al cliente
- El servicio es realizado exitosamente cuando el cliente recibe ‘m + 1’ respuestas de diferentes nodos con el mismo resultado, donde m es igual al número máximo de nodos que pueden fallar.

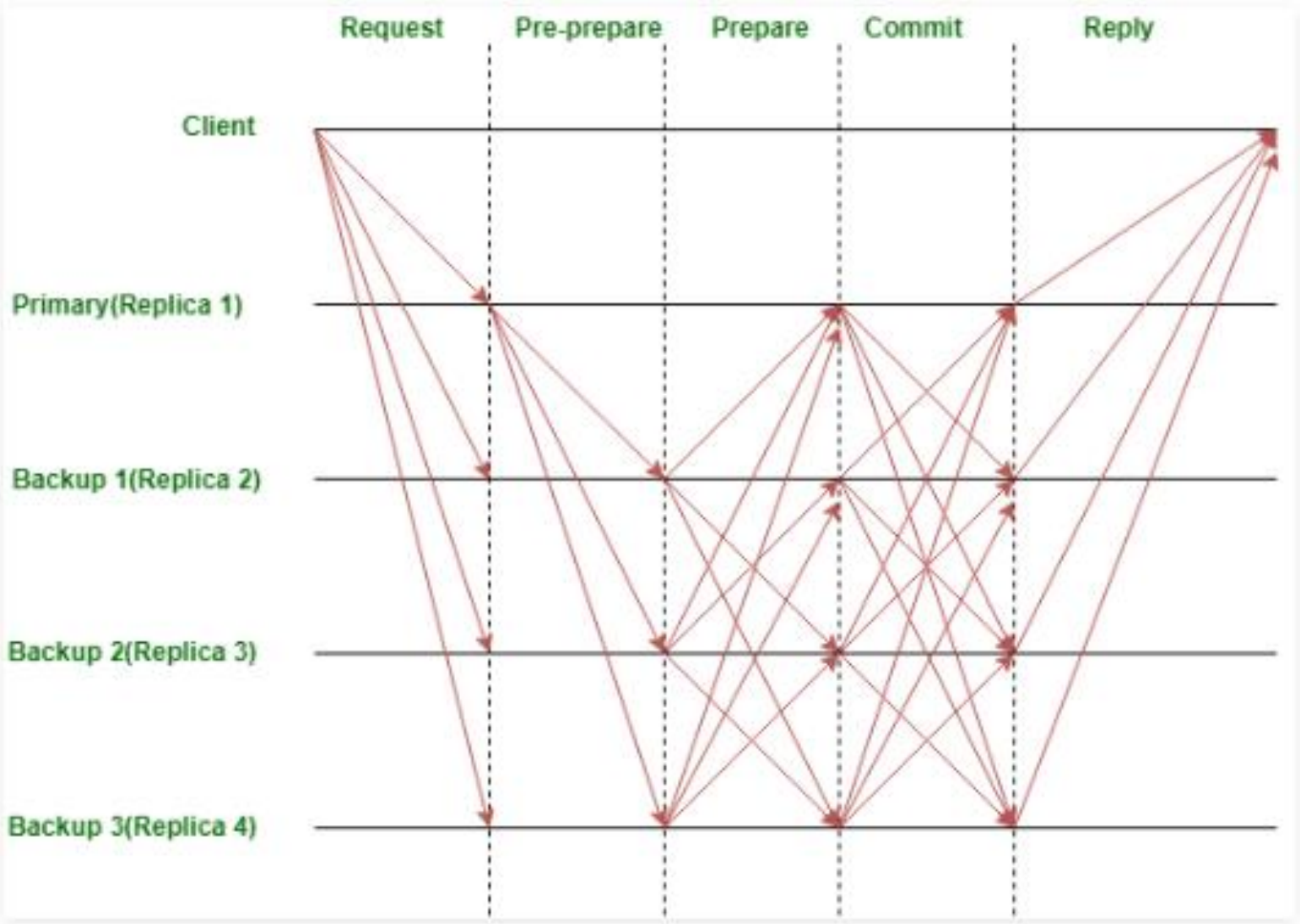


Imagen tomada de GeeksforGeeks

Conclusión

La meta del Grid Computing es reunir la mayor cantidad de recursos computacionales para ponerlos a disposición de quien los necesite a un bajo coste. Las ventajas que ofrece el Grid han hecho que su popularidad aumente y sea implementado en proyectos y áreas de todo tipo.

World Community Grid
(imagen tomada de Wikipedia)

European Grid Infrastructure
(imagen tomada de Wikipedia)