uc3m | Universidad Carlos III de Madrid

OpenCourseWare

Sistemas Paralelos y Distribuidos

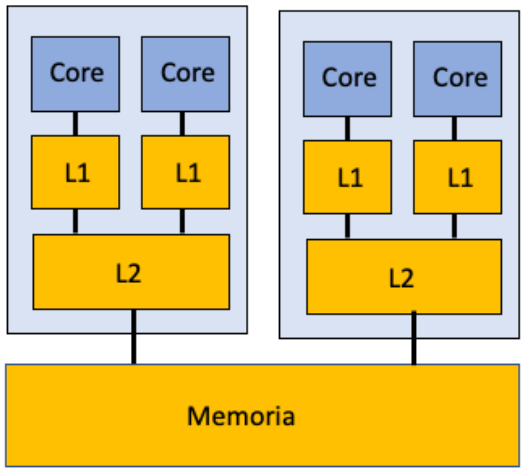
Félix García Carballeira

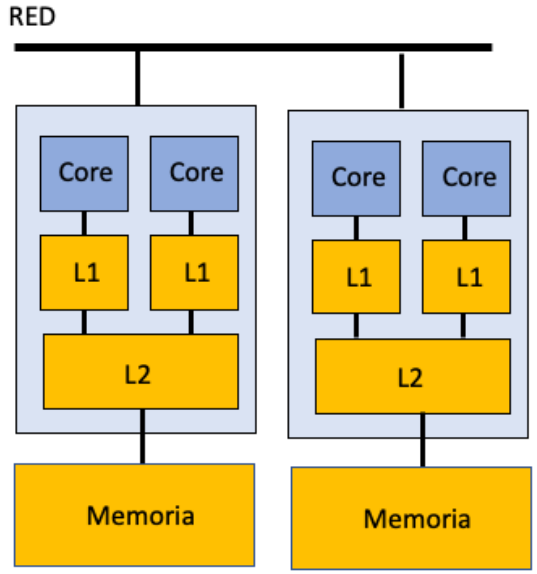
Alejandro Calderón Matos

Tema 1. Introducción

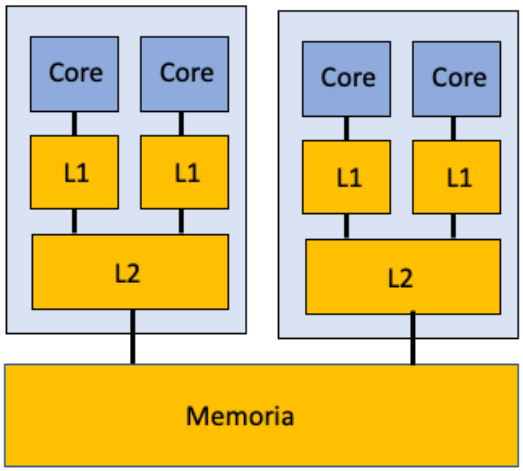
CC BY NC SA

Memoria compartida vs memoria distribuida

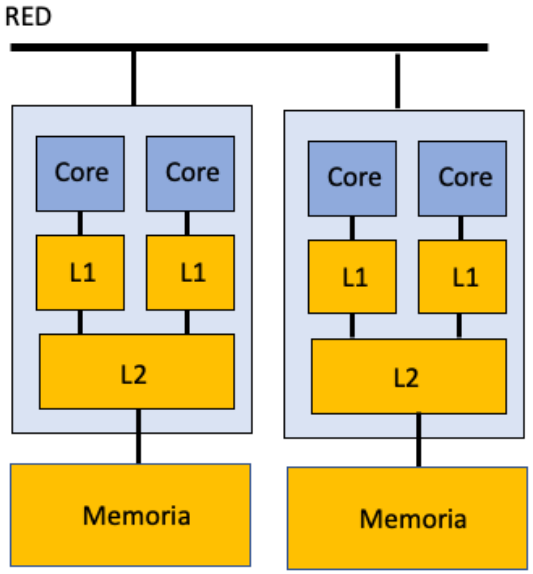




Memoria compartida vs memoria distribuida



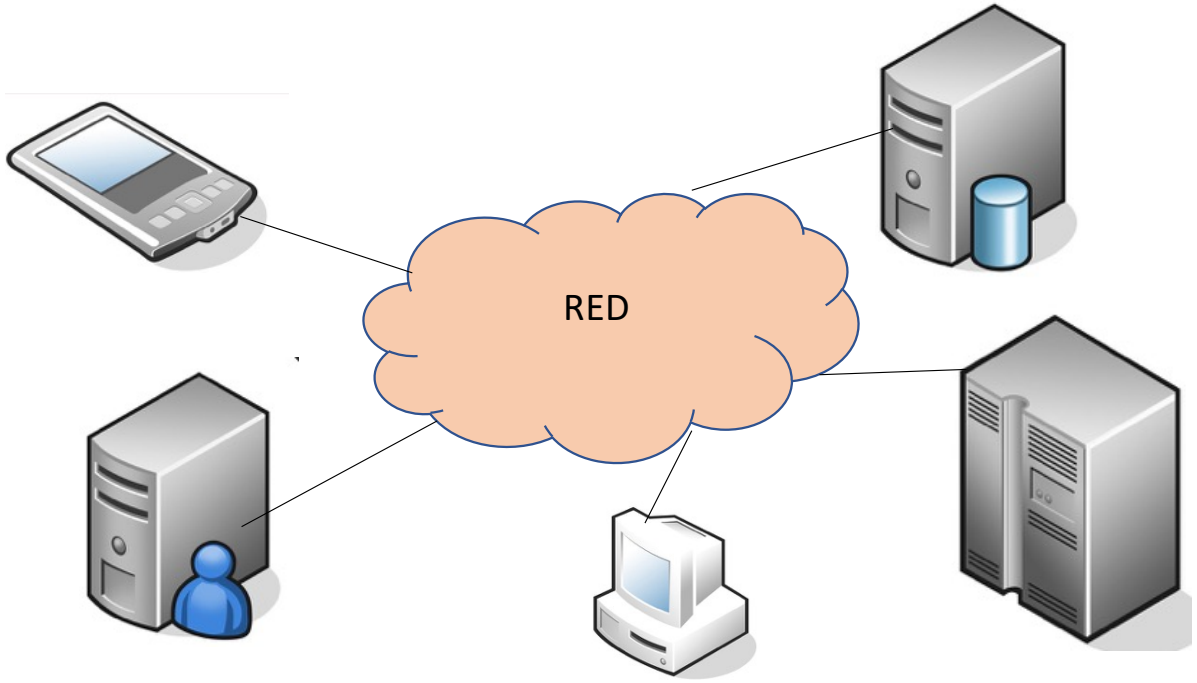
- Comunicación mediante memoria
- Sincronización mediante Locks



- Comunicación mediante paso de mensajes
- Sincronización mediante paso de mensajes

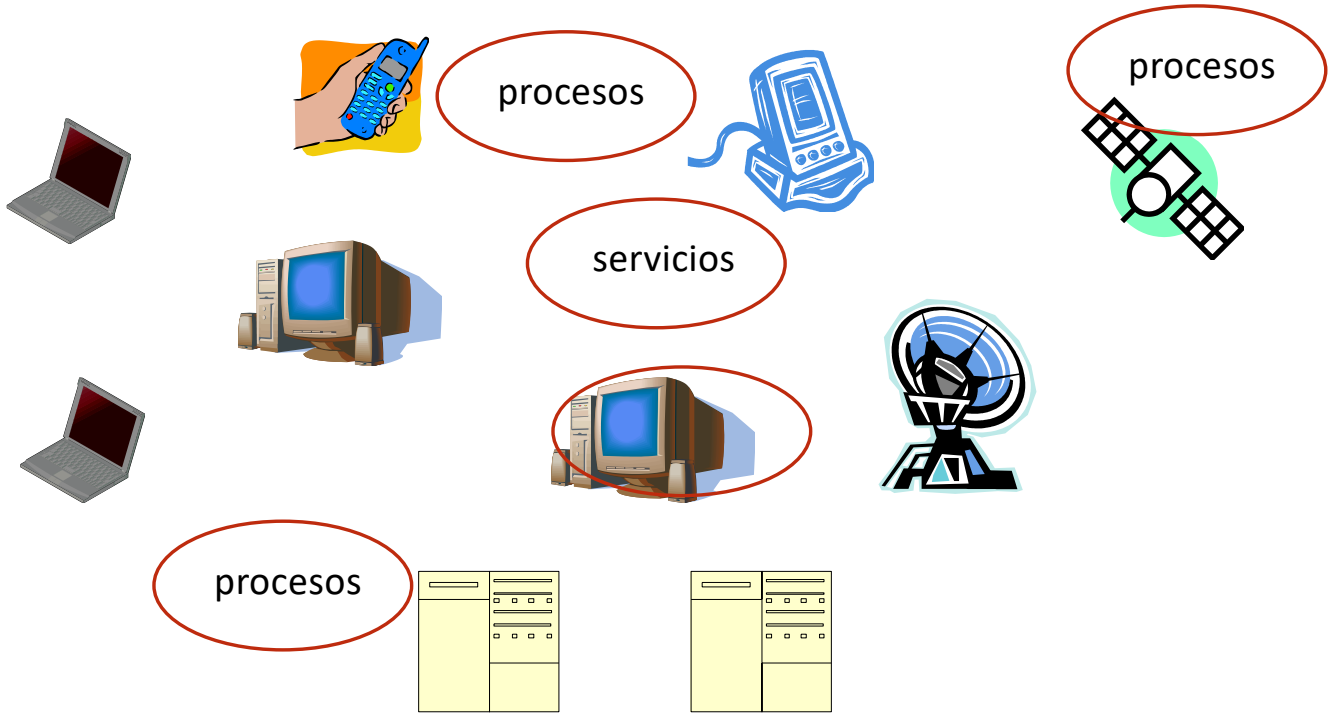
Sistema distribuido

- Sistema compuesto por recursos de computación físicamente distribuidos conectados a través de una red que se comunican y coordinan entre sí



Computación distribuida

- Computación que se realiza en un sistema distribuido



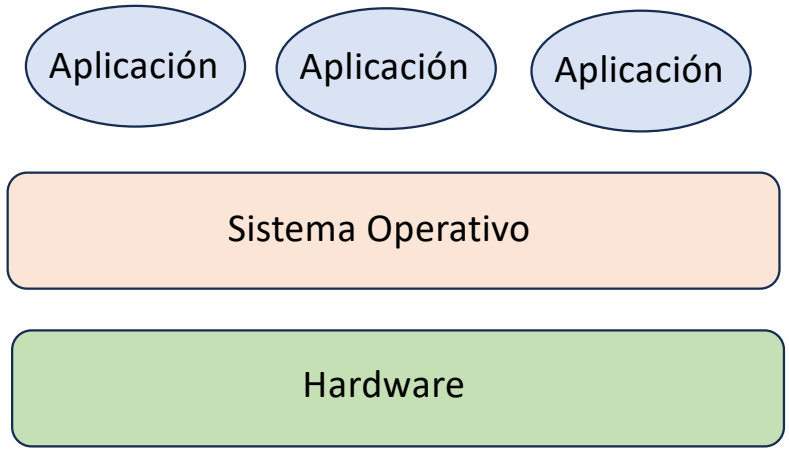
Sistemas distribuidos y paralelos

- Sistemas distribuidos
  - Objetivo: compartir recursos y colaborar
  - Redes de computadores
- Sistemas paralelos
  - Objetivo:
    - Alto rendimiento (*speedup*)
    - Alta productividad
  - Máquinas paralelas (arquitecturas dedicadas)
    - Multiprocesadores
    - Multicomputadores
  - Entornos distribuidos:
    - Clusters
    - Peer-to.peer
    - Cloud
    - Edge computing

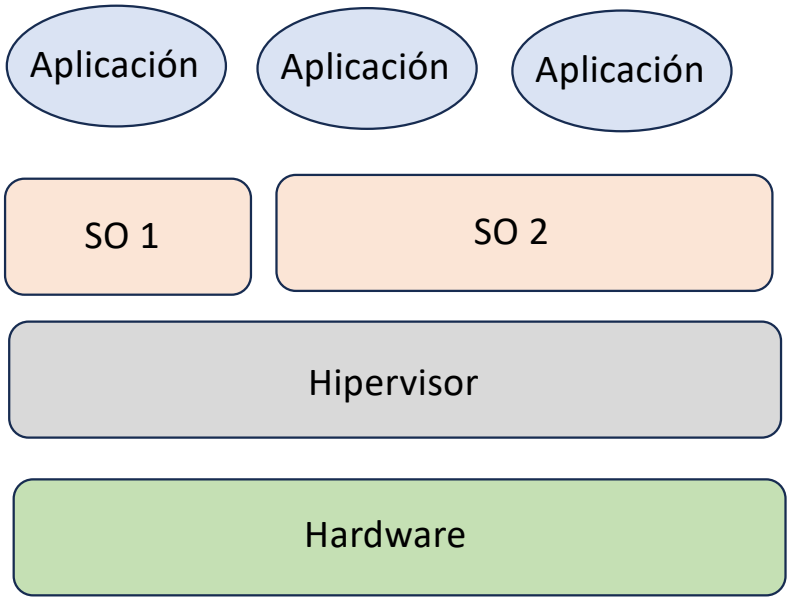
Sistemas distribuidos y paralelos

- Avances en sistemas distribuidos:
  - Mejoras en los procesadores (arquitecturas multicore)
  - Mejoras en las memorias y tecnologías de almacenamiento
  - Mejoras en las técnicas de virtualización
  - Mejoras en el Desarrollo de aplicaciones

Virtualización



Sistema tradicional



virtualización

- Posibilidad de ejecutar en un computador (*host*) un programa que crea un computador virtual (*guest*) sobre el que ejecutar cualquier entorno.

Características de un sistema distribuido

- Múltiples componentes autónomos
- Ausencia de un reloj global
- Todos los recursos pueden no ser accesibles a la vez
- El software ejecuta en procesos concurrentes sobre máquinas diferentes
- Software más complejo
- Múltiples puntos de fallo
- Comunicación a través de redes

Ventajas que ofrecen los sistemas distribuidos

- Compartir recursos (HW, SW, datos).
- Ofrecen una buena relación coste/rendimiento
- Capacidad de crecimiento
- Tolerancia a fallos, disponibilidad, replicación
- Distribución de la carga
- Concurrencia. Servicio a múltiples usuarios
- Velocidad. Capacidad global de procesamiento disponible para ejecución paralela de una aplicación

Aspectos de diseño

- Heterogeneidad de los componentes
- Comunicación y sincronización
- Capacidad de crecimiento, escalabilidad
- Estructura de software
- Tolerancia a fallos
- Calidad de servicio (QoS)
- Transparencia

Heterogeneidad

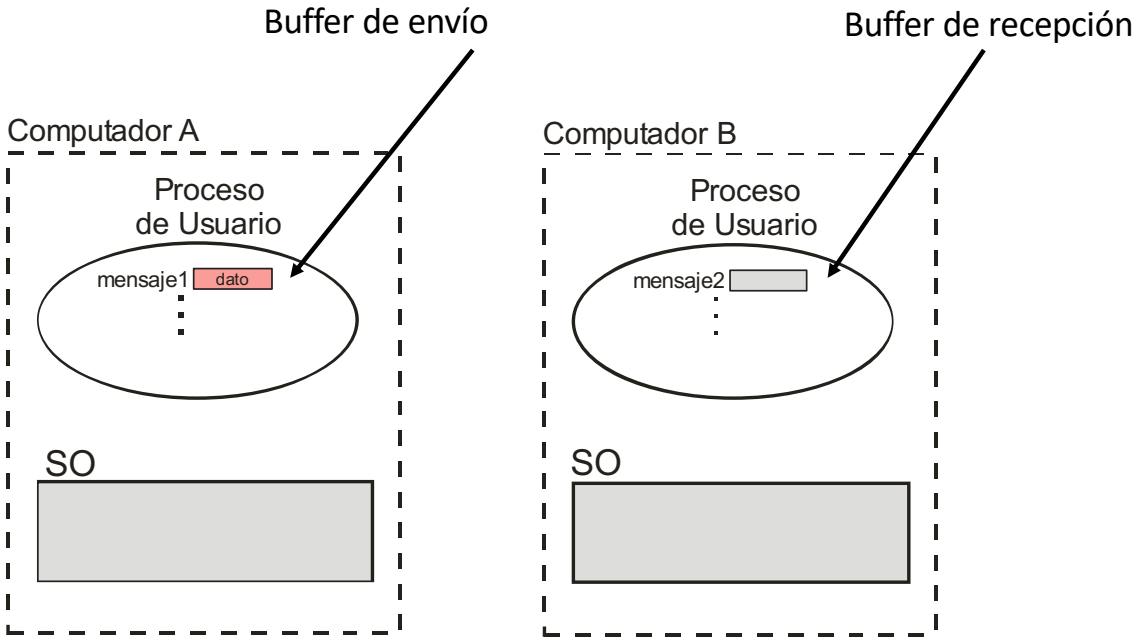
- Heterogeneidad de los componentes:
  - Es la variedad y diferencia de los siguientes componentes:
    - Redes
    - HW de computadores
    - Sistemas operativos
    - Lenguajes de programación
    - Aplicaciones

¿Cómo resolver la heterogeneidad?

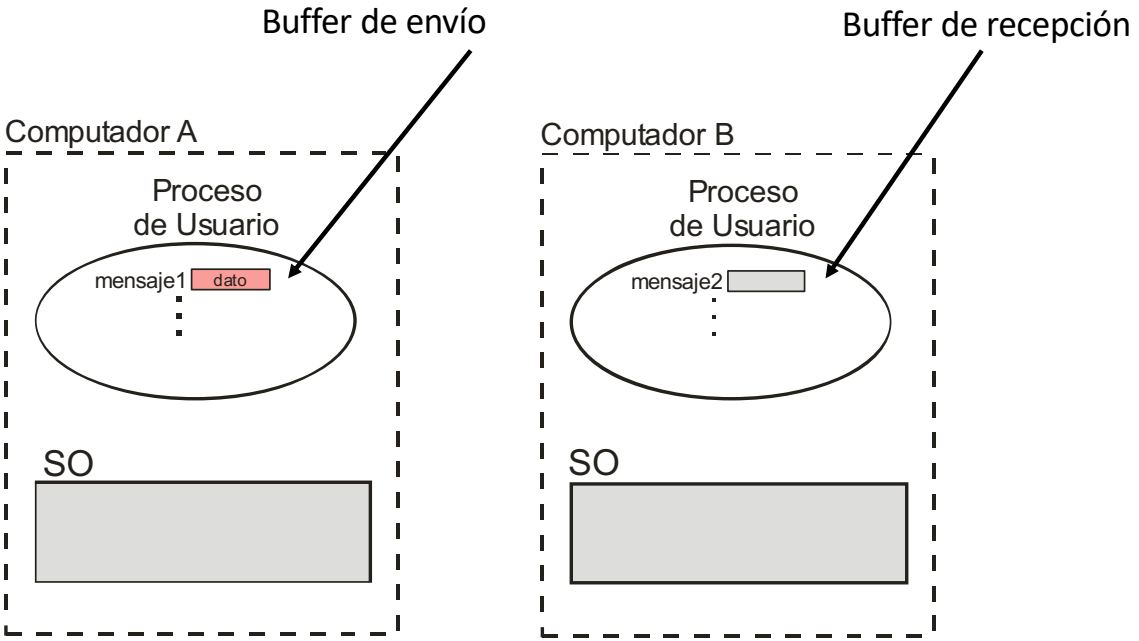
- Empleo de sistemas abiertos y estándares (es la característica del sistema que determina si el sistema puede ser extendido y re-implementado)
  - Especificaciones e interfaces de acceso públicas (ej. RFCs)
  - Mecanismos de comunicación uniformes
  - Se pueden construir sobre SW y HW heterogéneo
- Ejemplos de sistemas abiertos:
  - TCP/IP
  - NFS
  - CORBA (www.omg.org)
  - HTTP
  - XML

Comunicación y sincronización

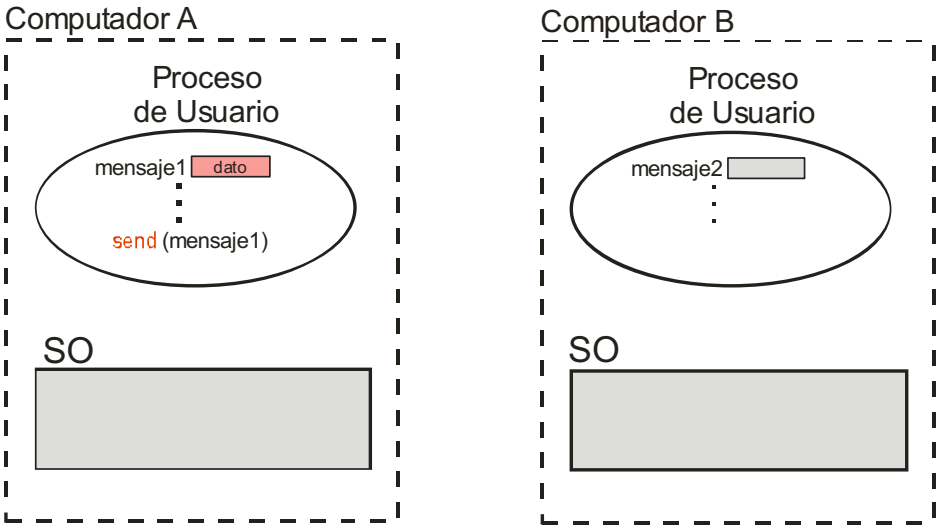
- Forma básica de C y S: paso de mensajes



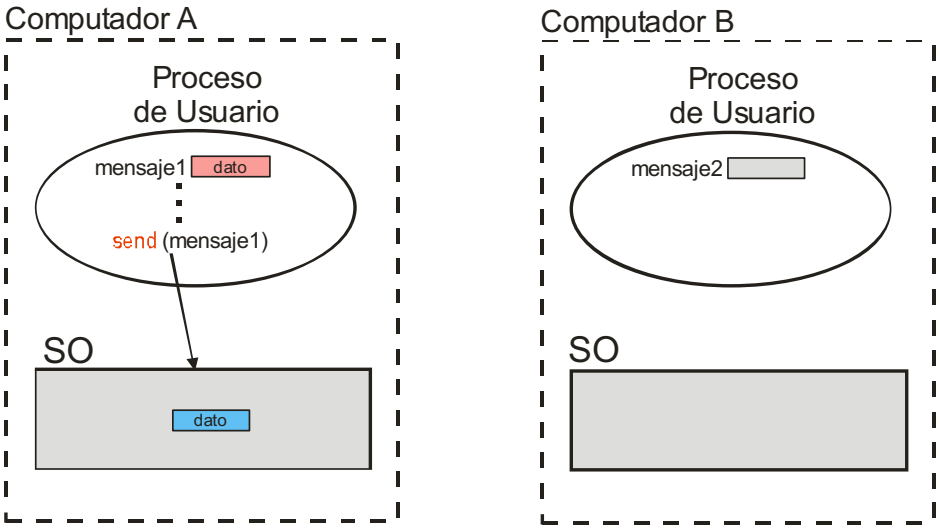
Paso de mensajes



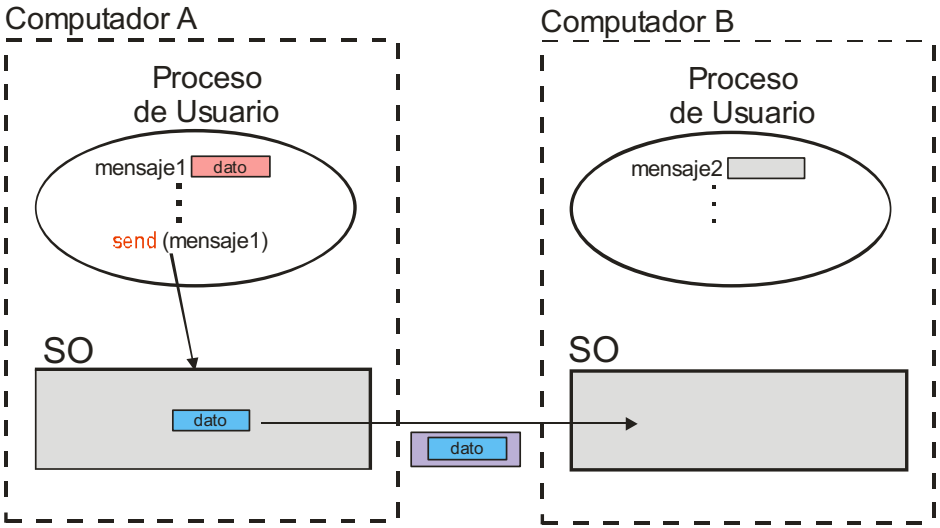
Paso de mensajes



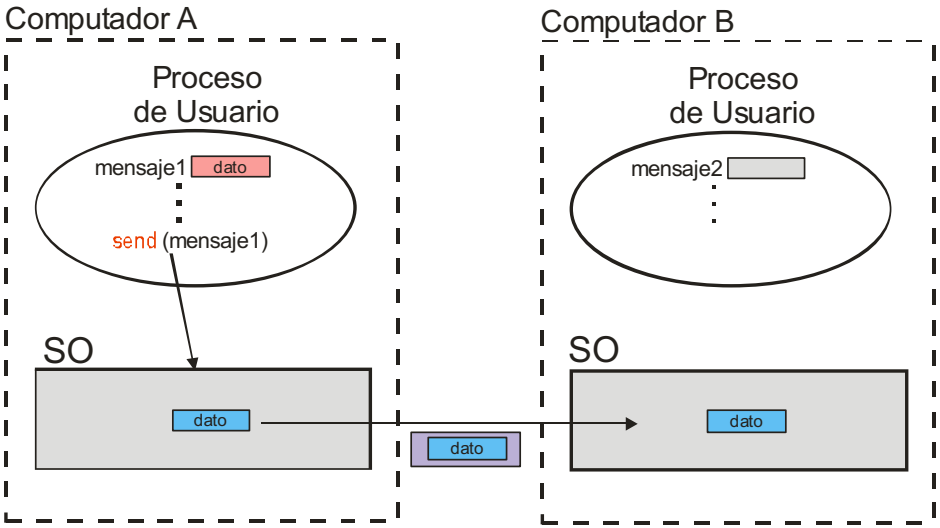
Paso de mensajes



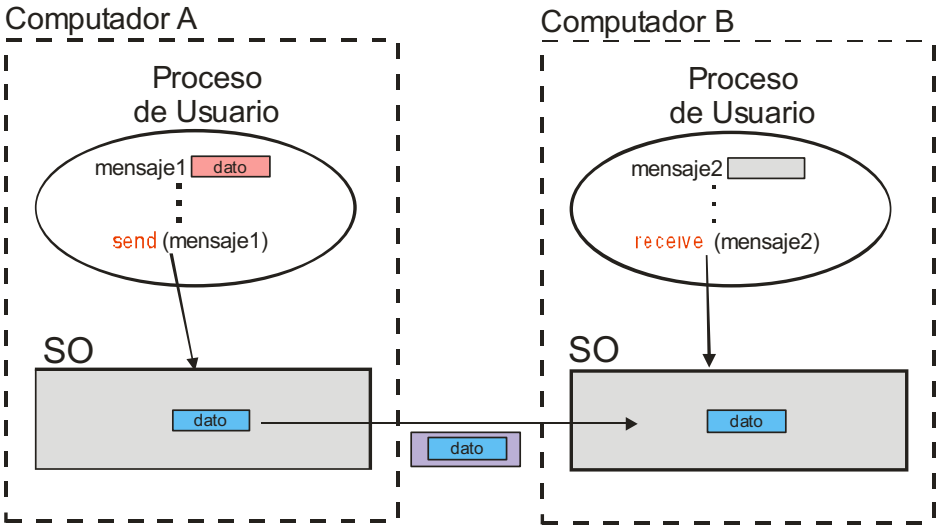
Paso de mensajes



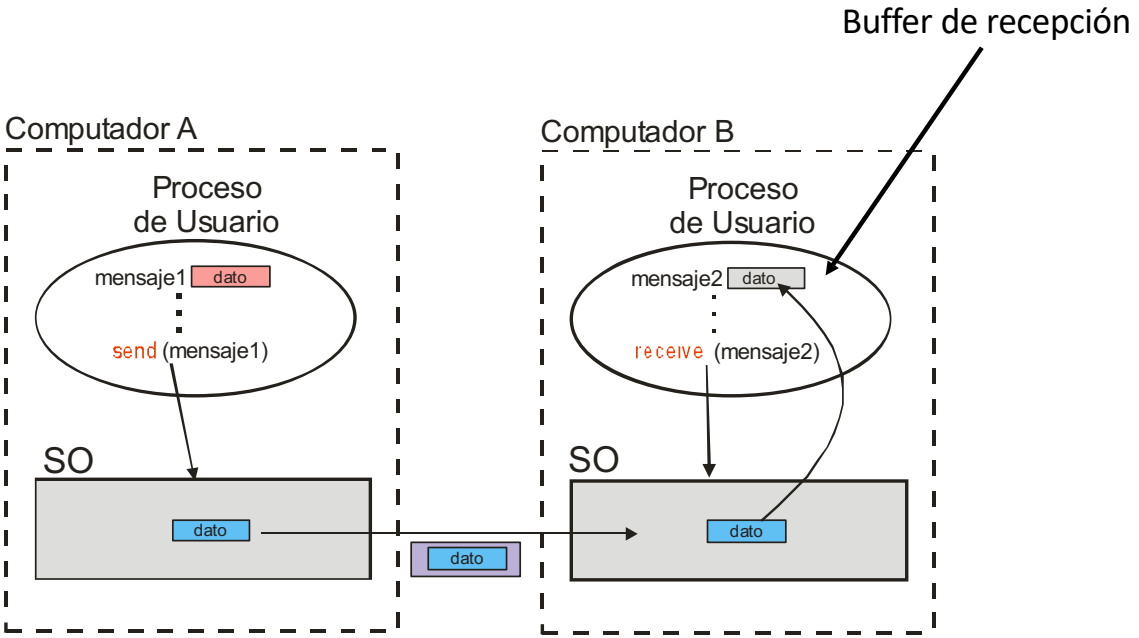
Paso de mensajes



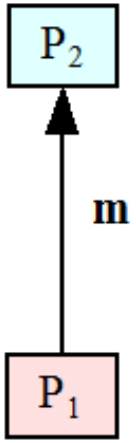
Paso de mensajes



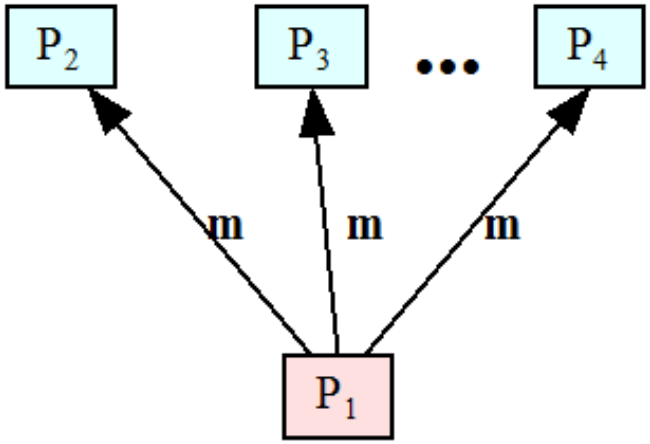
Paso de mensajes



Comunicación basada en mensajes



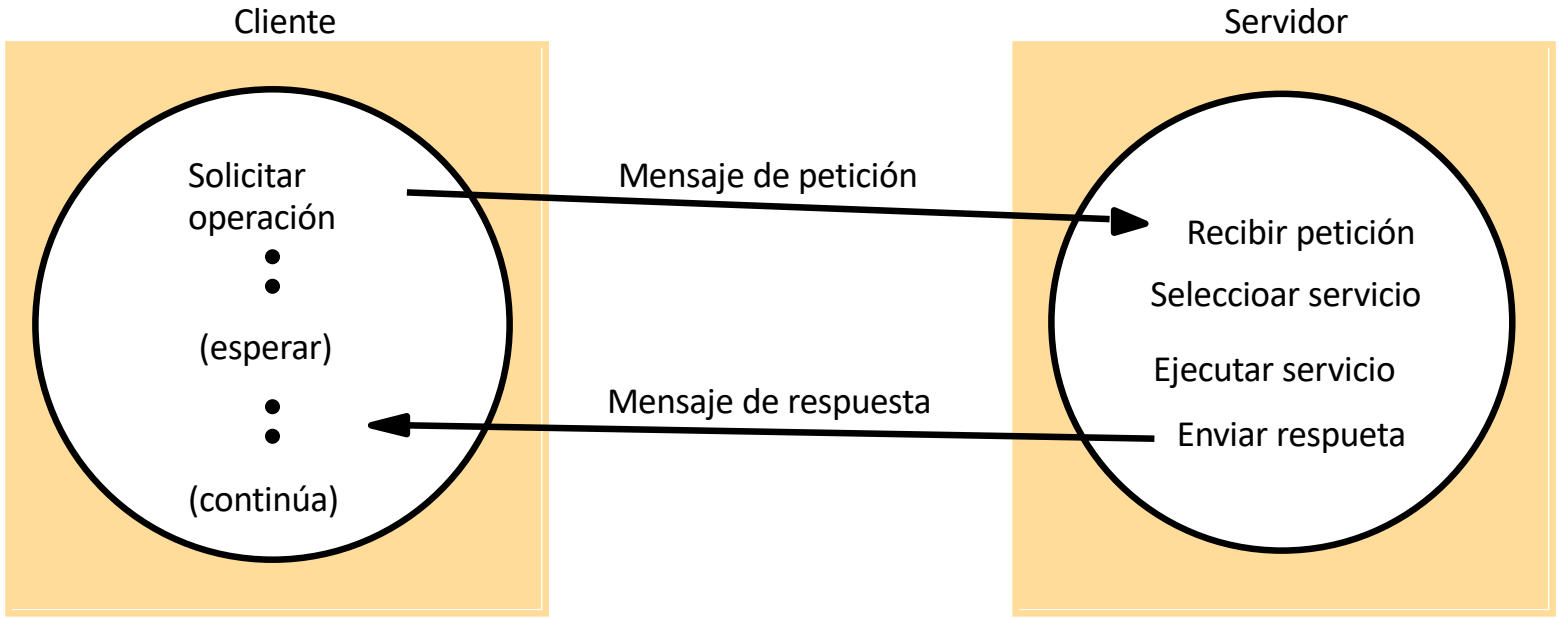
Punto a punto



multicast

- Sockets

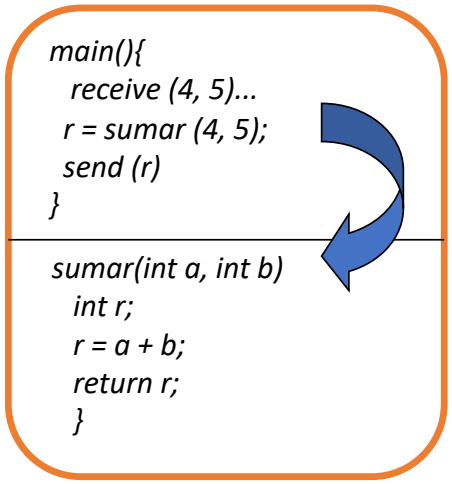
Comunicación cliente-servidor



- Ejemplos: Mail, web,
- Mecanismos: Sockets, RPC, RMI, Servicios Web

Llamadas a procedimientos remotos

Aplicación convencional



Llamadas a procedimientos remotos

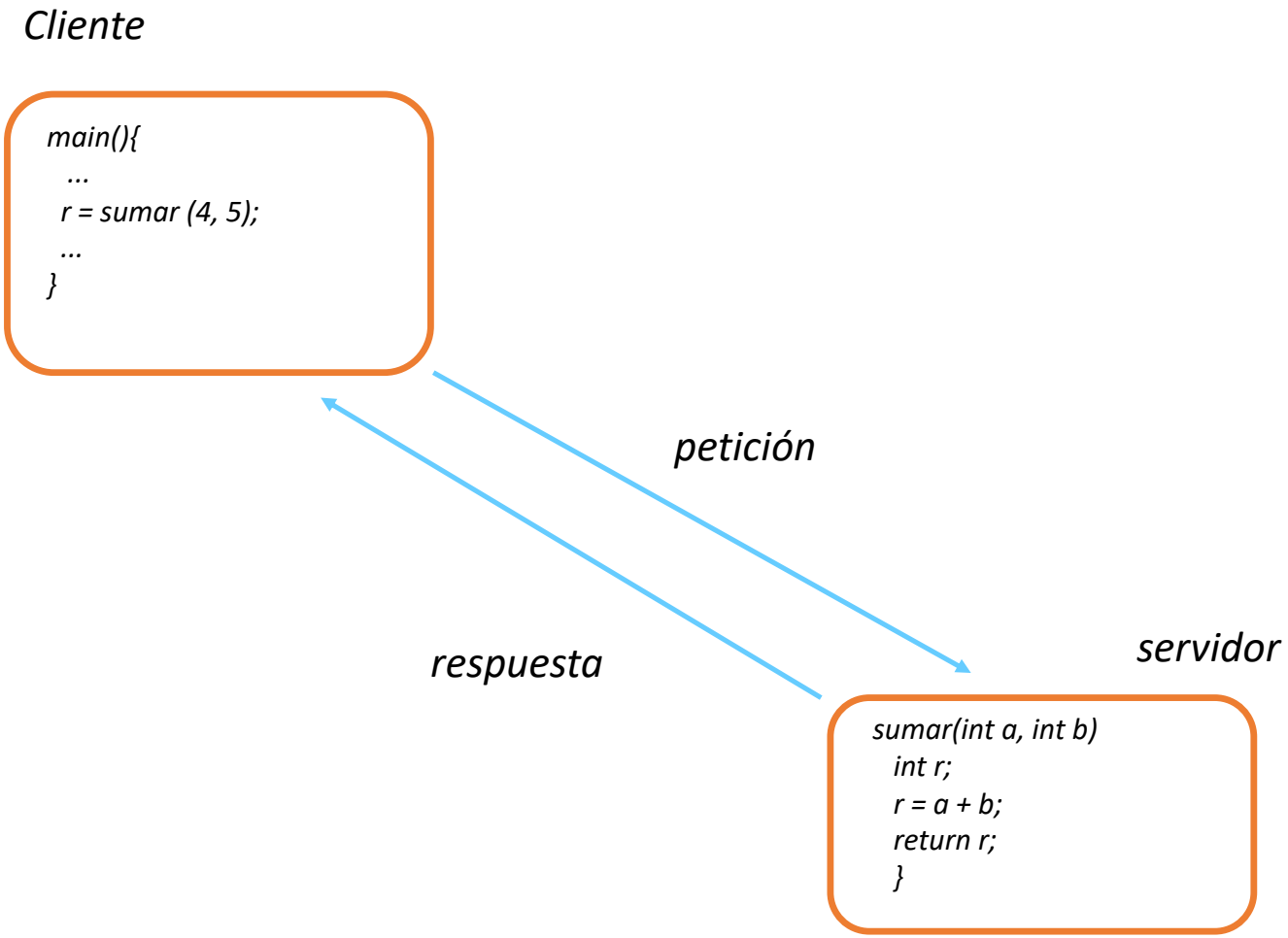
Cliente

```
main(){
  ...
  r = sumar (4, 5);
  ...
}
```

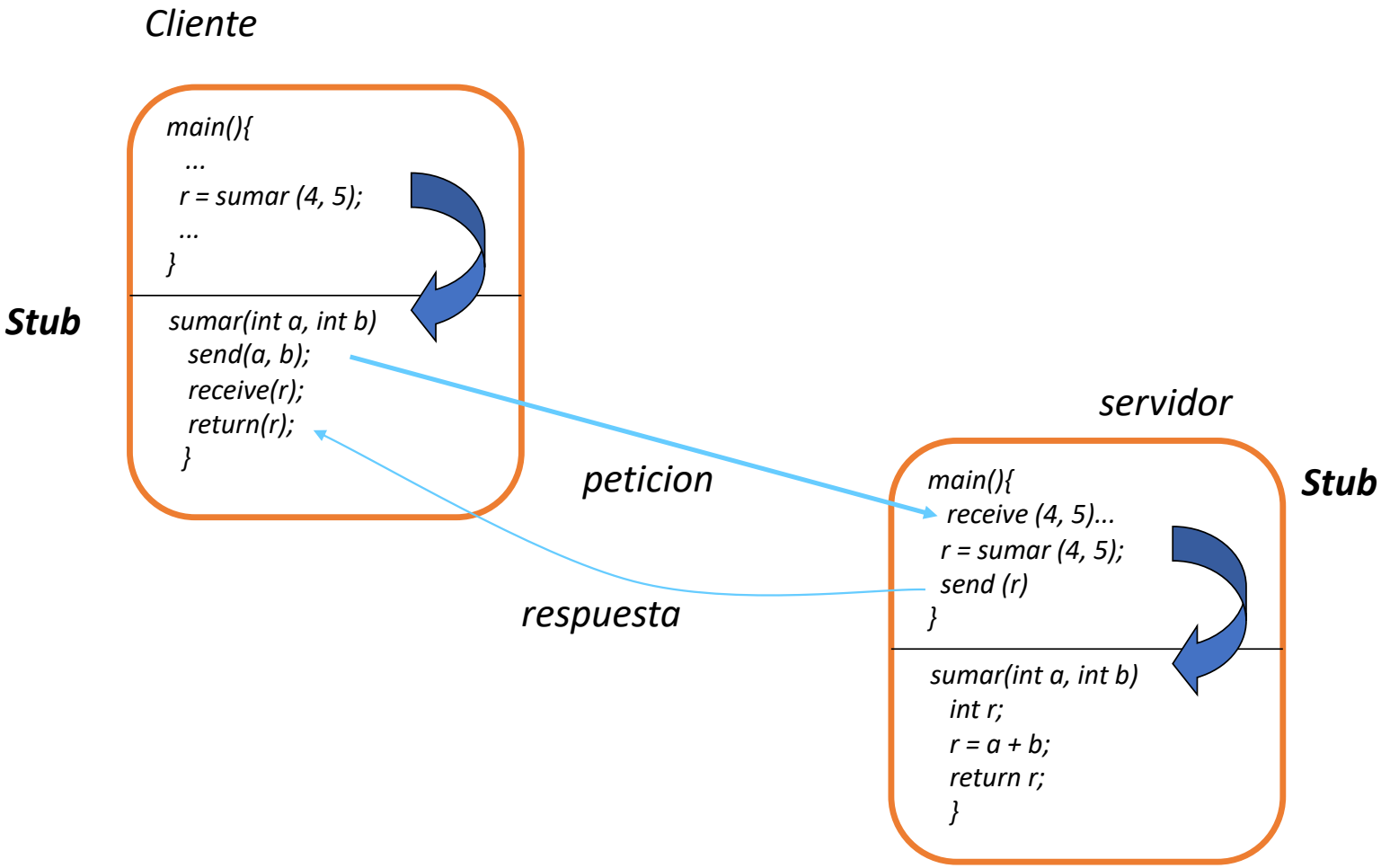
servidor

```
sumar(int a, int b)
  int r;
  r = a + b;
  return r;
  }
```

Llamadas a procedimientos remotos



Llamadas a procedimientos remotos



RPC en Pyhton. Servcio calculadora

```
import rpyc
from rpyc.utils.server import ThreadedServer
class CalculatorService(rpyc.Service):
    def exposed_add(self, a, b):
        return a + b
    def exposed_sub(self, a, b):
        return a - b
    def exposed_mul(self, a, b):
        return a * b
    def exposed_div(self, a, b):
        return a / b

if __name__ == "__main__":
    server = ThreadedServer(CalculatorService, port = 12345)
    server.start()
```

Cliente del servicio calculadora

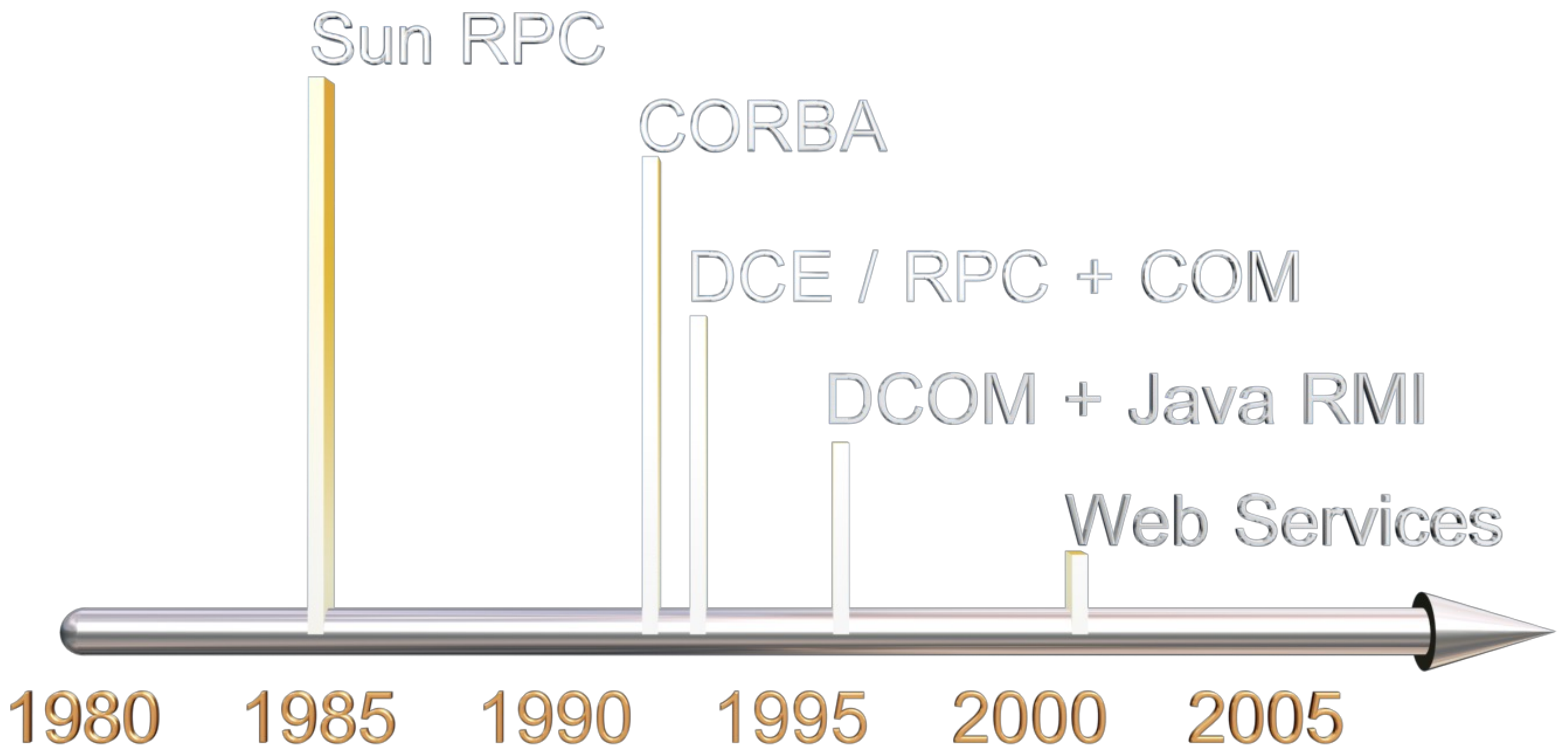
```
import rpyc

conn = rpyc.connect("localhost", 12345)

x = conn.root.add(4,7)
print(x)

x = conn.root.sub(4,7)
print(x)
```

Evolución de las RPC



Functions as a Services (FaaS)

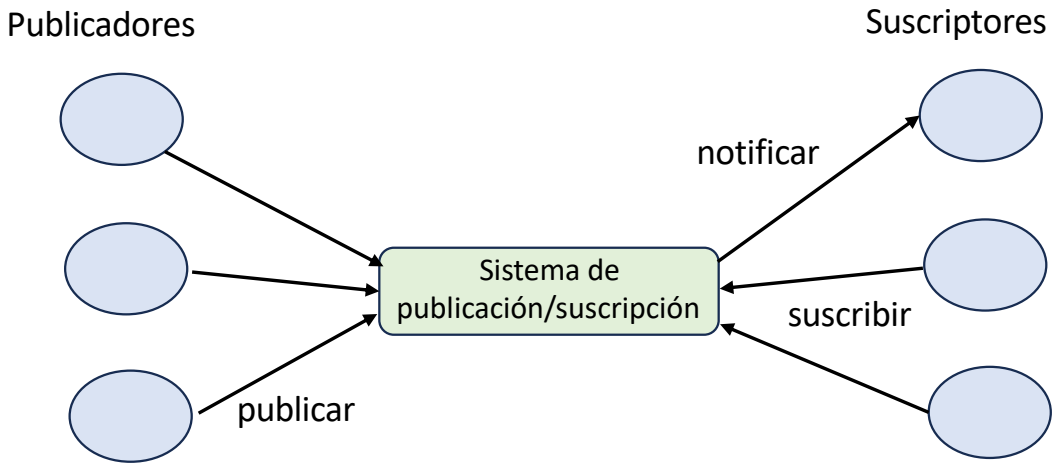
- Modelo de computación en cloud que permite desarrollar aplicaciones sin necesidad de servidor
- Aplicaciones pueden ejecutar funciones o partes de una aplicación sin necesidad de un servidor
- Mayor escalabilidad
- Ejemplos:
  - AWS Lambda
  - Google Cloud Functions
  - Microsoft Azure Functions
  - IBM/Apache's OpenWhisk

Comunicación de grupos

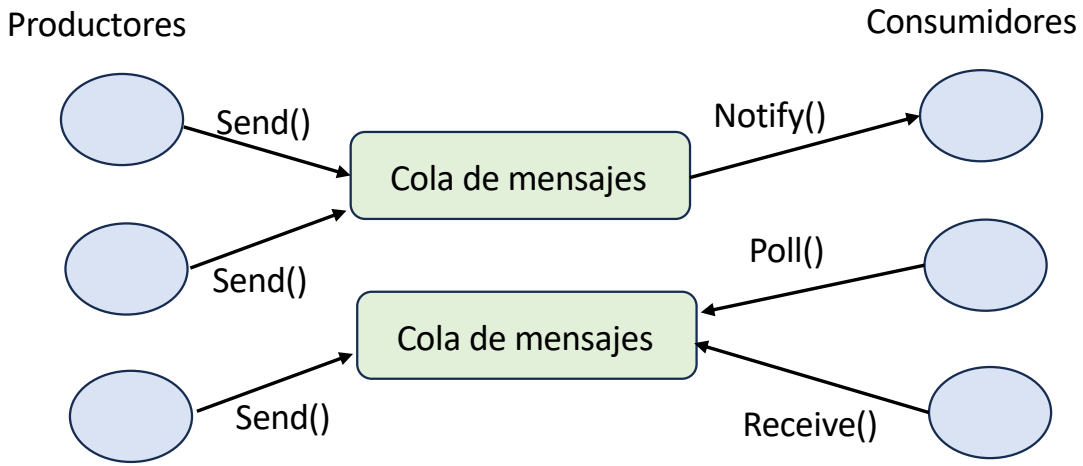
- Servicio en el que un mensaje en enviado a un grupo y, posteriormente, el mensaje es entregado a todos los miembros del grupo
  - El emisor no conoce las identidades de los receptores
- Utilidad:
  - Envío fiable de información a múltiples clientes (operaciones multicast)
  - Aplicaciones colaborativas (juegos,…)
  - Soporte para técnicas de tolerancia a fallos
  - Sistemas monitorización y gestión
- La comunicación de grupos implementa una comunicación *multicast y* operaciones para unirse o dejar un grupo

Sistemas de publicación/suscripción

- Sistemas distribuidos basados en eventos
  - Sistemas financieros
  - Aplicaciones de tiempo real
  - Aplicaciones colaborativas
  - Aplicaciones de monitorización



Colas de mensajes

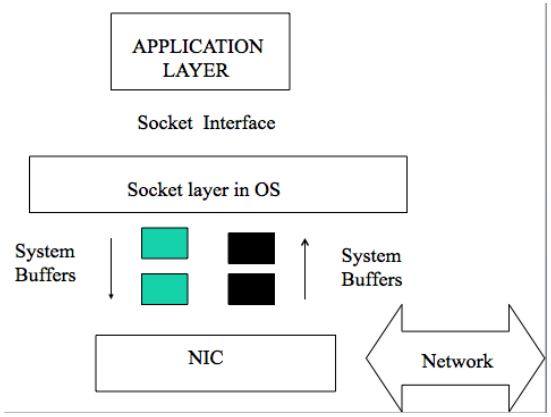


- IBM’s WebSphre MQ, Microsoft’s MSMG, Oracle’s Streams Advanced Queuing (AQ), JMS

Papel del sistema operativo en las comunicaciones

- Los sistemas operativos proporcionan implementaciones de los protocolos de comunicaciones

- El SW de comunicación de un sistema operativo se organiza como un conjunto de componentes con tareas concretas
  - Subsistema de almacenamiento: *buffers* donde almacenar los paquetes que llegan y se envían (limitado)



Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en la misma máquina. Intercambio de 300000 datagramas.

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en la misma máquina. Intercambio de 300000 datagramas.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	0.17	177
128	0.09	274
256	0.034	102
512	0.12	373
1024	0.32	959
2048	0.16	482
4096	0.18	534
8192	0.41	64

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos máquinas. Intercambio de 300000 datagramas.

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos máquinas. Intercambio de 300000 datagramas.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	0	
128	0	
256	0.08	17
512	0	
1024	0.004	14
2048	0	
4096	0.0036	5
8192	0.0073	21

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en una instancia de máquina virtual de Amazon.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	1,95	1951
128	2,87	1972
256	3,63	2030
512	2,29	2072
1024	4,22	2095
2048	2,06	2060
4096	2	2049
8192	1,79	1712

Problemas de fiabilidad

- Pérdida de datagramas entre dos procesos que ejecutan en dos instancias de máquinas virtuales en Amazon.

Mensaje (bytes)	% datagramas perdidos	longitud de la ráfaga máxima
64	3,45	175
128	0,25	119
256	1,11	209
512	2,83	307
1024	1,97	341
2048	3,56	263
4096	7,73	99
8192	5,35	60

Funciones del SO en las comunicaciones

- El SW de comunicación de un sistema operativo se organiza como un conjunto de componentes con tareas concretas
  - Subsistema de almacenamiento: *buffers* donde almacenar los paquetes que llegan y se envían (limitado)
- En implementaciones UNIX típicas
  - TCP reserva para cada puerto (socket) un buffer de 8 KB y UDP 2 buffers de 8KB. El tamaño se puede incrementar hasta 64 KB.
  - Los mensajes a enviar se copian a estos buffers
  - El contenido de estos buffers se fragmenta y se copian a nuevos bloques de memoria a utilizar por IP
  - IP envía finalmente los paquetes por la interfaz de red correspondiente

Sobrecarga introducida por el SO

- Procesamiento de TCP/IP
- Cambios de contexto
- Copias de datos en buffers intermedios

Transparencia

- Heterogeneidad: acceso a recursos heterogéneos de forma idéntica
- Acceso: acceso a recursos locales y remotos de forma idéntica
- Ubicación: Acceso a recursos sin conocimiento de su posición física en la red
- Concurrencia: ejecución concurrente de procesos sin interferencias
- Replicación: uso de múltiples réplicas de recursos sin conocimiento de su existencia
- Fallos: permitir la ejecución aun en presencia de fallos
- Movilidad: permitir el movimiento de los recursos y clientes sin afectar a su funcionamiento
- Crecimiento: permitir la capacidad de crecimiento sin afectar al sistema y aplicaciones
- Rendimiento: rendimiento similar en recursos locales y remotos

Escalabilidad

- Un sistema distribuido es escalable si su capacidad de procesamiento puede crecer al añadir más usuarios
  - Aumenta el rendimiento al aumentar el número de nodos
  - El tiempo de respuesta no aumenta
  - La fiabilidad no se degrada
- Dimensiones:
  - Numérica: usuarios, objetos, servicios
  - Geográfica: distancia
  - Administrativa: número de organización involucradas

Técnicas de escalabilidad

- Escalabilidad horizontal
  - Incrementar el número de nodos
- Escalabilidad vertical
  - Incrementar la potencia de los nodos existentes
- Replicación
  - Datos: múltiples copias del mismo dato
  - Procesos: múltiples ejecuciones de la misma computación
- Caching
  - Consistencia
- Distribución
  - Distribución de carga en múltiples nodos
    - DNS, DHT
- Servicios sin estado

Consistencia

- El problema de la consistencia (coherencia) surge cuando varios procesos acceden y actualizan datos de forma concurrente
  - Coherencia de las actualizaciones
  - Coherencia de la replicación
  - Coherencia de caches
  - Coherencia ante fallos
  - Relojes consistentes

Modelos de consistencia

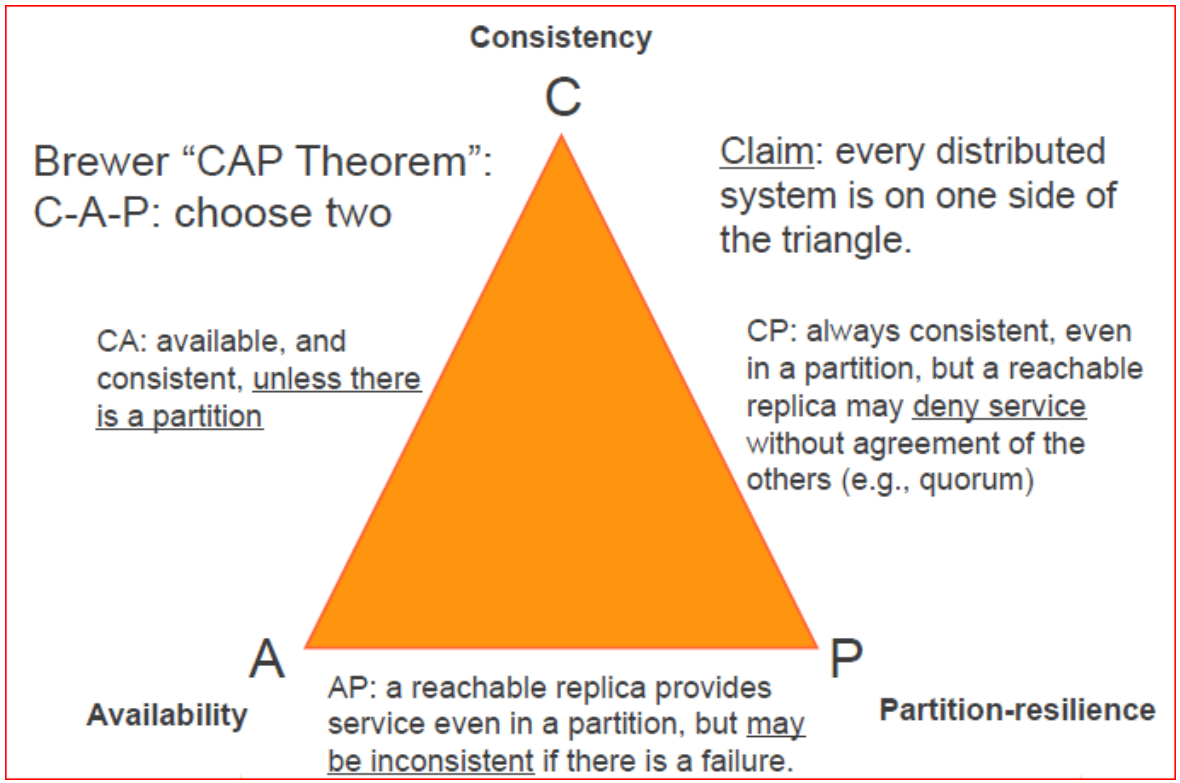
- Consistencia estricta
  - Las actualizaciones se hacen visibles a todos los nodos de forma inmediata
- Consistencia eventual
  - Modelo de consistencia débil, las actualizaciones no se hacen visibles de forma inmediata a todos los nodos, pero una cierta operación deberá ser finalmente visible para todos los sistemas
- Consistencia causal
  - Las operaciones sigues relaciones de orden causal
    - A→B ==> la operación A se recibe antes que B en todos los nodos

Tolerancia a fallos

- Un sistema distribuido es inherentemente más propenso a errores
- Un sistema es tolerante a fallos si el sistema cumple sus especificaciones a pesar de la presencia de fallos
- Se debe asegurar:
  - Disponibilidad: los recursos son disponibles a pesar de que haya fallos.
  - Atomicidad: la consistencia de los recursos se debe asegurar a pesar de fallos

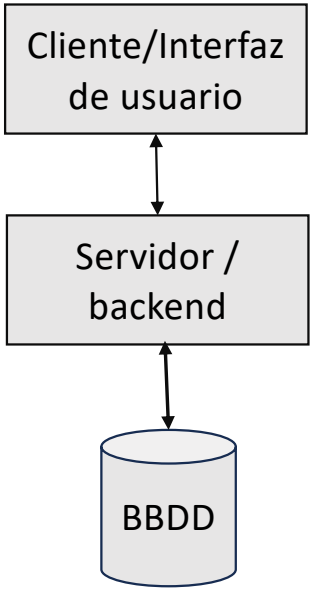
El teorema CAP

Brewer, PODC 2000



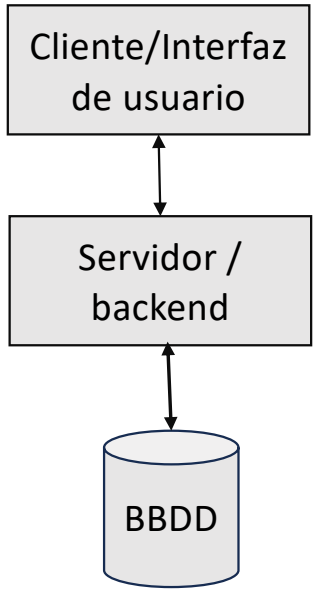
Arquitecturas basadas en microservicios

- Arquitectura de una aplicación distribuida monolítica



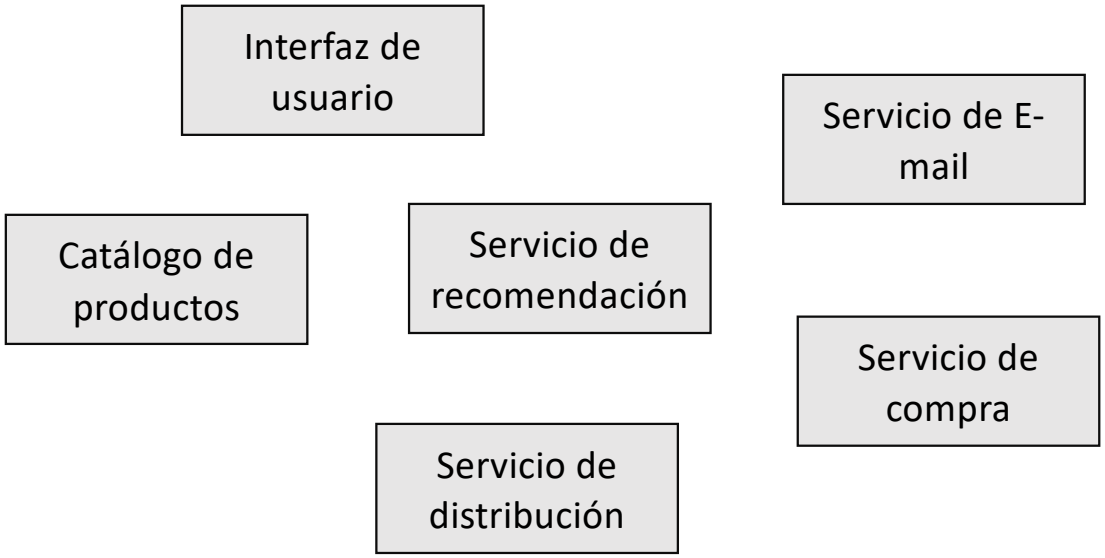
Arquitecturas basadas en microservicios

- Desventajas de una distribuida monolítica
  - Dificultad para gestionar la escalabilidad
  - Dificultad para integrar tecnologías diferentes



Arquitectura basada en microservicios

- Aplicación basada en servicios independientes que se comunican entre sí



Ventajas de las arquitecturas basadas en microservicios

- Servicios (procesos) independientes
  - Contenedores
- Despliegues independientes
- Equipos de desarrollo independientes
- Mejora la tolerancia a fallos
- Mejora la escalabilidad

API Gateways

- Servicio para la creación, publicación, el mantenimiento y monitorización de las API utilizadas para la comunicación entre servicios
  - Servicios REST

Ejemplos de sistemas distribuidos

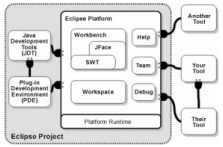
- Sistemas Cliente-servidor
- Clusters /entornos HPC
- Grid Computing
- Sistemas peer to peer
- Computación voluntaria
- Cloud computing
- Fog Computing
- Edge computing (IoT)
- Mobile cloud computing
- Sistemas procesamiento big data

Computación de altas prestaciones

- Mejores algoritmos

- Mejores procesadores

- Paralelismo

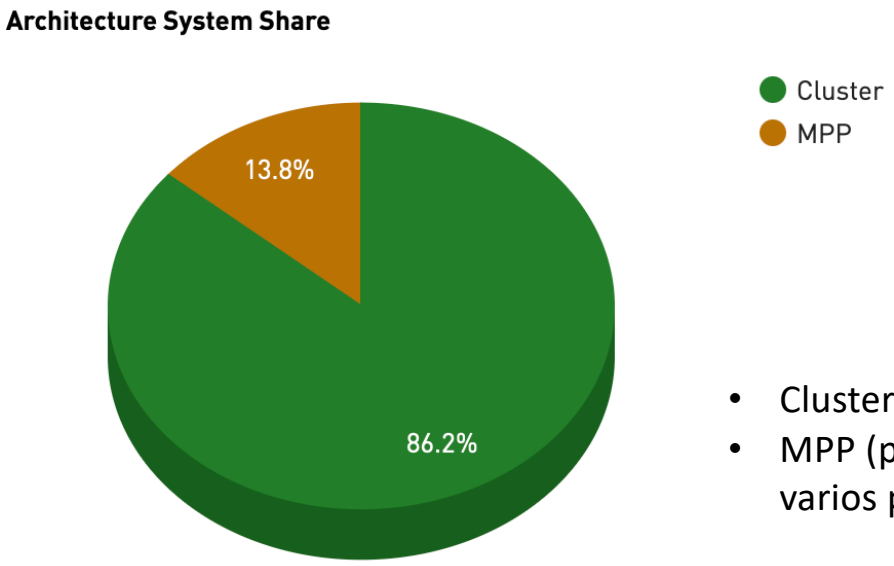


software

+

hardware

Top 500



- Cluster: máquinas independientes conectadas por red
- MPP (procesador paralelo masivo): un solo equipo con varios procesadores conectados en red.