Universidade Vigo
ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Centros de Datos

T3 - Procesamiento en los Centros de Datos (II)

David Ruano Ordás
Departamento de Informática

drordas@uvigo.es

Despacho 409

Contenidos

- Introducción
  - Definición
  - Tipos
  - Conceptos previos
- Balanceo de Carga
  - Definición
  - Implementaciones
  - Persistencia de Conexiones
- Alta disponibilidad
  - Requisitos
  - Métricas
  - Principios básicos
  - Tipos de clúster
    - Activo↔Activo
    - Activo↔Pasivo
  - Sharding
  - Integridad y consistencia
  - Split-brain
  - Linux-HA
- Computación de altas prestaciones
  - Introducción
  - Programación paralela en HPC
  - Hadoop

Alta disponibilidad

- **Definición:**

Conjunto de nodos que garantizan la disponibilidad de un servicio
- ***Ante** paradas imprevistas (fallos)*
- ***Ante** paradas previstas (mantenimiento)*

Características:
- *Comunicación entre nodos **del** cluster*
- *Disponer **de** nodos redundantes*

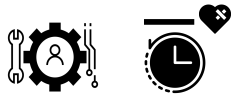
Capaz de:
- *Detección de los fallos hardware/software (en nodos del cluster)*
- *Mantener el servicio operativo (iniciándolo en otro nodo).*
- *Garantizar la integridad de los datos.*

Uso:
- *Soporte a servicios/aplicaciones críticas **para una organización que** no pueden verse interrumpidas → alto coste de downtime*
  - *Bases **de** datos críticas*
  - *Aplicaciones web **de comercio electrónico***
  - *Sistemas **de** ficheros **compartidos***
  - *Servidores **de** correo*

Evitar los puntos únicos de fallo (SPoF)) → Garantizar tolerancia a fallos sin provocar inconsistencias de datos

Alta Disponibilidad

- **Requisitos:**
  - **Fiabilidad**:
    - **Capacidad** del **sistema** de **funcionar** de manera **consistente** y **predecible** a lo largo del tiempo, **sin** experimentar **fallos inesperados o interrupciones.**
    - **Calculado** a **través** de la **métrica MTBF.**
  - **Disponibilidad**:
    - **Capacidad** del cluster **para estar** en **funcionamiento** y **proporcionar servicios** a los usuarios **cuando se** le **necesita** → el **%** de **tiempo** en que el **sistema** está **disponible** para sus **usuarios.**
    - Grados **altos** de **disponibilidad** suponen **contar** con **mecanismos** para que el **sistema continúe operativo** aún **ante** la **presencia** de **fallos** o paradas **imprevistas** (**tolerancia** a fallos/**failover**)
    - **Técnicas**:
      - **Redundancia**.
      - **Detección** y **recuperación** de **fallos automáticos (conmutación** tras **fallo)** → un **nodo asuma** los **servicios** de un **nodo caído** para **minimizar** el **tiempo** de **inactividad**
      - **Balanceo** de la **carga**
  - **Facilidad** de **mantenimiento**:
    - **Capacidad** de **gestionar**, **actualizar** y **mantener** el **cluster** de manera **eficiente** y **con** un **mínimo impacto** en la **disponibilidad** de los **servicios.**
    - **Calculado** a **través** de la **métrica MTTR.**



Alta Disponibilidad

- **Métricas:**

Uptime

Tiempo total en funcionamiento

Downtime

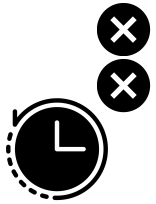
Tiempo total fuera de servicio

TTR
(Time To Repair)

Tiempo transcurrido hasta poner en marcha el sistema

TTF
(Time To Failure)

Tiempo transcurrido hasta que el sistema falla

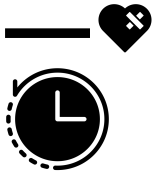


TBF
(Time Between Failures)

Tiempo transcurrido entre fallos

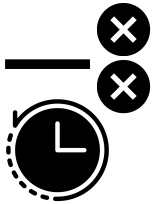
MTTF
(Mean Time To Failure)

$$\frac{\text{Uptime de todos los dispositivos}}{\text{Número de dispositivos}}$$



MTTR
(Mean Time To Recovery)

$$\frac{\text{Tiempo de cada Downtime}}{\text{Número de fallos}}$$



MTBF
(Mean Time Between Failures)

$$\text{MTTF} + \text{MTTR}$$

Availability

$$\frac{\text{Uptime}}{\text{Uptime + Downtime}} \times 100$$

Alta Disponibilidad

- **Métricas:**

Ejercicio:

Una empresa de comercio electrónico gestiona una plataforma crítica de venta online, que cuenta con 20 servidores para soportar tanto el frontend como el backend. Estos servidores deben estar disponibles 24/7, ya que cualquier tiempo de inactividad podría generar pérdidas de ventas. Para analizar la confiabilidad de la infraestructura, se han recopilado los siguientes datos sobre el comportamiento de los servidores a lo largo de un año (8760 horas).

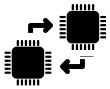
- *Cada servidor funciona en promedio 720 horas antes de fallar*
- *Cuando un servidor falla, el tiempo promedio de reparación es de 4 horas.*
- *Después de la reparación, el servidor funciona en promedio 1440 horas antes de experimentar otro fallo.*
- *En total, cada servidor estuvo fuera de servicio durante 20 horas en todo el año.*

Calcula para un servidor:

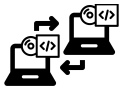
- *Uptime y el Downtime totales en el año.*
- *TTF, TTR y TBF.*
- *Calcula el MTTF (Mean Time To Failure), el MTTR (Mean Time To Repair) y el MTBF (Mean Time Between Failures).*
- *¿Cuál es la disponibilidad (Availability) del sistema en porcentaje, considerando los valores de Uptime y Downtime?*

Alta Disponibilidad

- **Principios básicos:**
  - **Redundancia hardware**: **replicación** de **componentes físicos clave** (servidores, discos duros, fuentes de alimentación y tarjetas de red) con el fin de **evitar puntos únicos** de **fallo**.
    - **Disponibilidad** de **sistemas críticos**
    - **Costes**



  - **Redundancia software**: **replicación** de **aplicaciones** o **servicios en diferentes servidores** o nodos, de modo que **si uno falla, otro asuma** la carga de **trabajo**.
    - **Flexibilidad**
    - Puede **funcionar** en el **hardware existente**
    - **Compleja configuración** y **administración.**

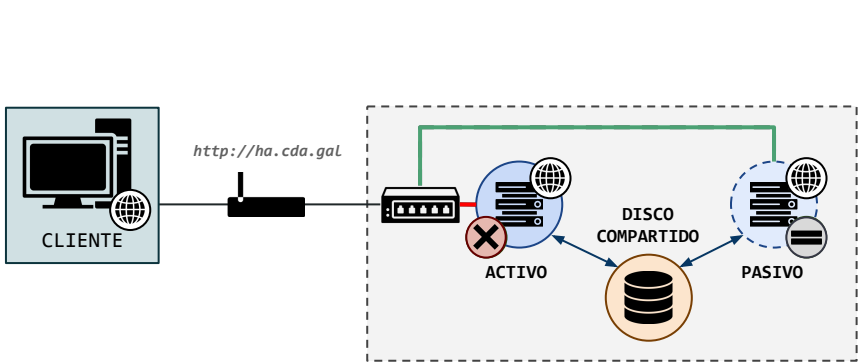


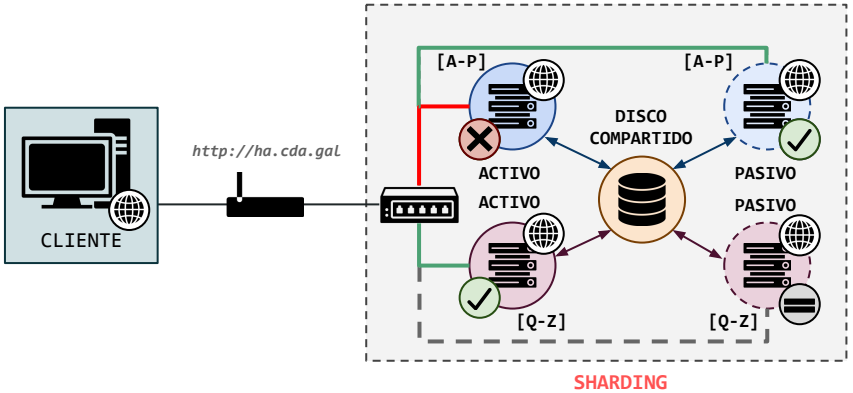
  - **Redundancia** de **datos**: **replicar** los **datos** en **múltiples ubicaciones** o dispositivos → **garantizar** la **disponibilidad** y la **integridad** de los **datos**.
    - **Asegura** la **protección** de **datos**
    - **Costes** de **almacenamiento**
    - **Complejidad** en la **gestión** de los **datos**

  - **Administración/Gestión** de la **redundancia: software** específico para **gestionar** los **componentes redundantes** para:
    - **Asegurar** la **continuidad** y **correcto funcionamiento** en caso de **parada**/caída **imprevista**.
    - **Garantizar** la **consistencia** e **integridad** de los **datos**.

Alta Disponibilidad

- **Tipos de clúster**:
  - **Activo ⇔ Pasivo**
    - Los **nodos activos manejan** las **solicitudes** de los **clientes**.
    - Los **nodos pasivos** están en **espera** y listos **para asumir** el **control si** un **nodo activo falla**.
      - **Esquema sencillo**.
      - **Configuración simple**.
      - **Uso ineficiente** de **recursos**: los **nodos pasivos** están **sin usarse**.
      - **Costes** de **hardware duplicados**: **nodo pasivo** debe ser **idéntico al** nodo **activo**.
      - **Complejidad** de la **administración**: **sincronización** de datos **entre nodo** activo y pasivo.
    - **Sharding**: **divide** los **datos** en **partes** más **pequeñas (shards)** y los **distribuye** en múltiples **servidores**

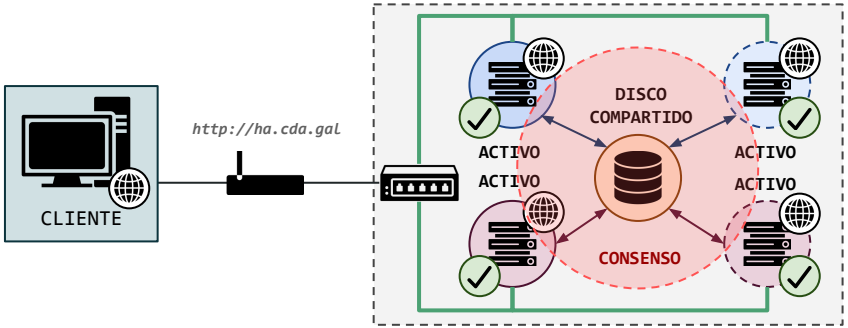


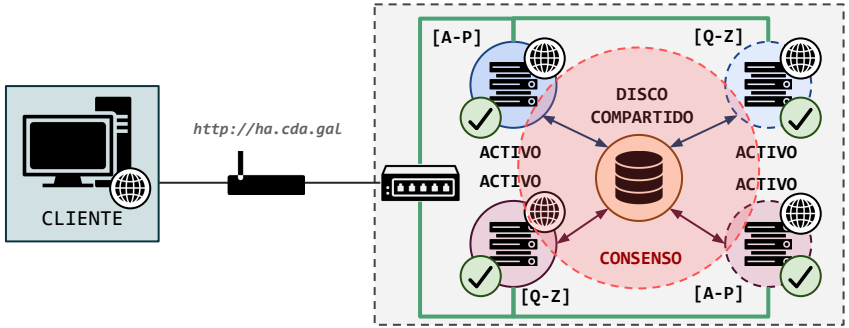


SHARDING

Alta Disponibilidad

- **Tipos de clúster:**
  - **Activo ⇔ Activo**
    - Todos los **nodos manejan** las **solicitudes** de los **clientes** al **mismo tiempo.**
    - **Cualquier nodo** puede servir como **respaldo ante fallos** en los demás nodos
    - Si un **nodo falla**, las **solicitudes** se **redistribuyen automáticamente** a los **nodos restantes**.
      - **Alto rendimiento**: todos los **nodos comparten** la **carga**.
      - **Tolerancia** a **fallos**: si un **nodo falla**, otros **nodos continúan atendiendo** las **solicitudes** de los clientes.
      - **Complejidad** de **configuración** y **administración**: **complicado administrar múltiples nodos** y **gestionar** la **distribución** de la **carga** → **requiere herramientas** de **administración específicas.**
      - **Latencia adicional**: dado que los **nodos comparten** la **carga**, puede haber una ligera **latencia adicional** al **procesar solicitudes** de clientes
    - **Combinable** con **balanceo** de **carga** → repartir la carga entre nodos





SHARDING

Alta Disponibilidad

- **Sharding:**
  - **Divide** los **datos** en **partes (shards)** más **pequeñas** y los distribuye en **múltiples servidores**
  - Tipos de sharding:
    - Sharding **horizontal**: **datos o cargas** de trabajo se **dividen** en **particiones horizontales** (por filas o registros) y se distribuyen entre nodos diferentes en el clúster.
      - Cada nodo en el clúster contiene una fracción de las filas de la base de datos o la carga de trabajo.

DNI,	Nombre,	Apellidos,	Provincia,	Saldo,	Transacciones
12345678A,	Pedro,	Pérez Pérez,	Ourense,	25000,	200
57345679B,	Juan,	Álvarez Conde,	Pontevedra,	10000,	314
89235668C,	María,	García García,	Ourense,	12000,	222
22345678D,	Ana,	Pardo González,	Pontevedra,	50000,	25
52355675E,	Marcos,	Gámez Gálvez,	A Coruña,	125000,	8
72395674F,	Rosa,	Pérez Pérez,	A Coruña,	5200,	87
84967147U,	Rebeca,	López Hijales,	Lugo,	1587,	255
55478965C,	Marta,	Mirás Mujica,	Pontevedra,	14000,	200

sharding horizontal

DNI,	Nombre,	Apellidos,	Provincia,	Saldo,	Transacciones
84967147U,	Rebeca,	López Hijales,	Lugo,	1587,	255

DNI,	Nombre,	Apellidos,	Provincia,	Saldo,	Transacciones
57345679B,	Juan,	Álvarez Conde,	Pontevedra,	10000,	314
22345678D,	Ana,	Pardo González,	Pontevedra,	50000,	25
55478965C,	Marta,	Mirás Mujica,	Pontevedra,	14000,	200

DNI,	Nombre,	Apellidos,	Provincia,	Saldo,	Transacciones
52355675E,	Marcos,	Gámez Gálvez,	A Coruña,	125000,	8
72395674F,	Rosa,	Pérez Pérez,	A Coruña,	5200,	87

DNI,	Nombre,	Apellidos,	Provincia,	Saldo,	Transacciones
12345678A,	Pedro,	Pérez Pérez,	Ourense,	25000,	200
89235668C,	María,	García García,	Ourense,	12000,	222

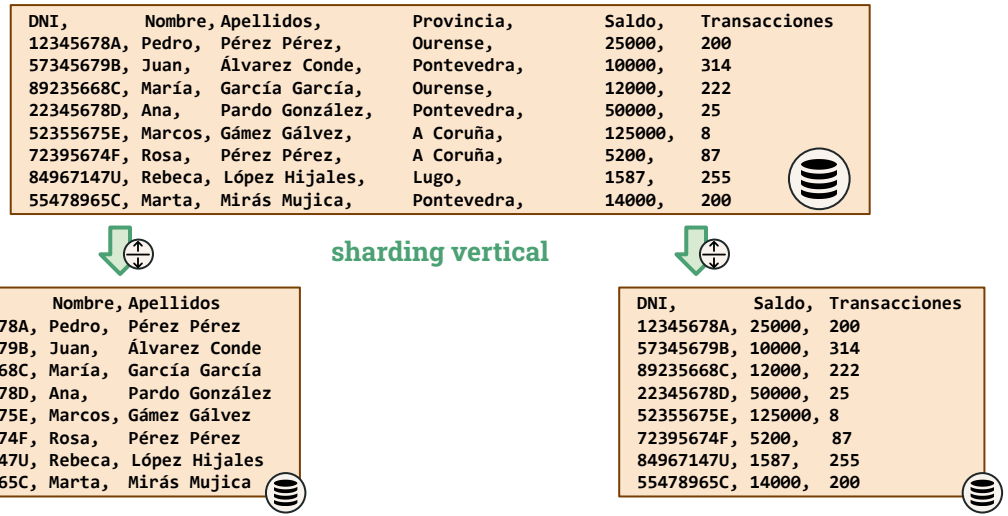
⏫ **Escalabilidad: más shards → más nodos**

⏫ **Carga distribuida: repartir → reducir** la **carga** de los **nodos**

⏬ **Consultas lentas: abarcan múltiples shards (totales)**

Alta Disponibilidad

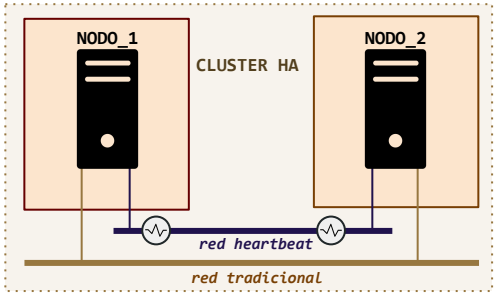
- **Sharding:**
  - **Divide** los **datos** en **partes (shards)** más **pequeñas** y los distribuye en **múltiples servidores**
  - Tipos de sharding:
    - Sharding **vertical**: **datos o cargas** de trabajo se **dividen** por **columnas** o funciones y se distribuyen en diferentes nodos del clúster



- ⏫ **Optimización: más shards → más nodos**
- ⏫ **Seguridad: repartir → reducir** la **carga** de los **nodos**
- ⏬ **Consultas lentas: múltiples columnas** de **distintos shards**
- ⏬ **Escalabilidad: limitada** por las **columnas**

Alta Disponibilidad

- **Consideraciones:**
  - El **fallo** de un **nodo** supone que **otro nodo**/s pasa a **hacerse cargo** del **recurso**
    - ***Failover***: **capacidad** de **recuperarse automáticamente** de un fallo en un nodo **desplegando** el **recurso** en **otro nodo.**
  - **Recuperación automática** de **servicios**/**aplicaciones** debe **garantizar** la **integridad** (consistencia) de los **datos**.:
    - **Heartbeat (latido)**: **mecanismo** de **sondeo** del **estado** de los **nodos.**
      - Se encarga de que los **nodos** del **cluster** se **comunican periódicamente** para **notificar su estado.**
      - **Si** un **nodo activo no emite** su **"latido"**, un **nodo** de **respaldo** (activo o pasivo) **lo reemplaza** y pasará a ocuparse del servicio/aplicación/recurso
      - **Implementación mediante conexiones** de **red dedicadas** (**red heartbeat**).
        - **Aislamiento** del **tráfico** → **evitar** que el **tráfico** de **red** y **heartbeat compitan** en la red.
        - **Evitar falsos positivos** (de fallos) → **evitar errores** de **nodos caídos causados** por **saturación**.
      - **Replicación** de **redes heartbeat** → **incrementar** la **tolerancia** a **fallos**.



Alta Disponibilidad

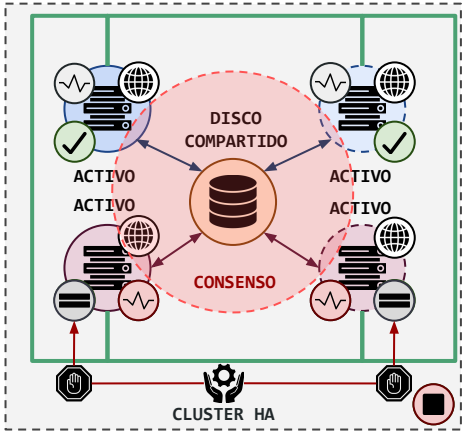
- **Integridad y consistencia:**
  - En **configuraciones Activo ⇔ Activo**:
    - Problemas:
      - **Múltiples nodos acceden simultáneamente** a los mismos **datos** →**riesgo** de **corrupción** de datos.
    - **Soluciones**:
      - Mecanismos de **control** de **concurrencia** (bloqueos/versiones).
      - Mecanismos de **consenso distribuido**.
      - **Transacciones**: agrupar **operaciones** para **garantizar** la **atomicidad**.
  - En **recuperaciones** ante **fallos**:
    - ***Split-brain***
      - Un **nodo** **activo no responde (emite pulsos)** a los **heartbeats**.
        - **Fallo** de **comunicación** de la red **heartbeat**
        - **Puede darse** en **más** de dos **nodos** de la **red**.
      - **Sistema asume** que ha **fallado** y **otro nodo asume** su **tarea** → **ambos nodos** están activos **realizando** la **misma tarea** simultáneamente.
      - **Mayor nodos** en cluster → **mayor posibilidad** de **split-brain**
      - Si se **usan datos compartidos** → **pueden corromperse** por el uso simultáneo.
        - Mecanismos de *fencing* (protección de acceso a datos compartidos):
          - ***Quórum***: **garantizar** que **solo** la **mayoría** de los **nodos activos controlan** los **recursos compartidos**.
          - ***STONITH***: **asegurar** que **un nodo defectuoso** no **acceda** ni corrompa los **recursos compartidos**.

Alta Disponibilidad

- **Integridad y consistencia:**

Quorum

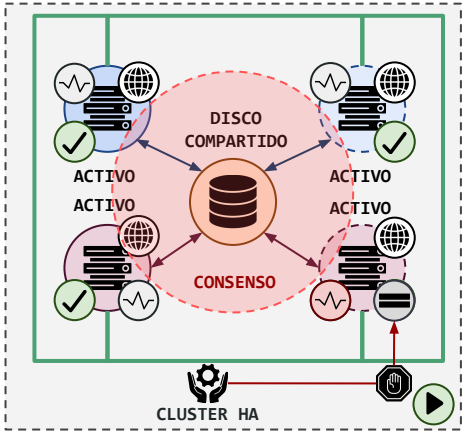
- Comunicación constante entre nodos
- Sistema de votación
- 1 voto por nodo
- Nodos que alcanzan el quórum (más del 50%) acceden a los recursos compartidos.



VOTOS=4
QUORUM=4*0.5=2 → >2
FALLOS POSIBLES=1

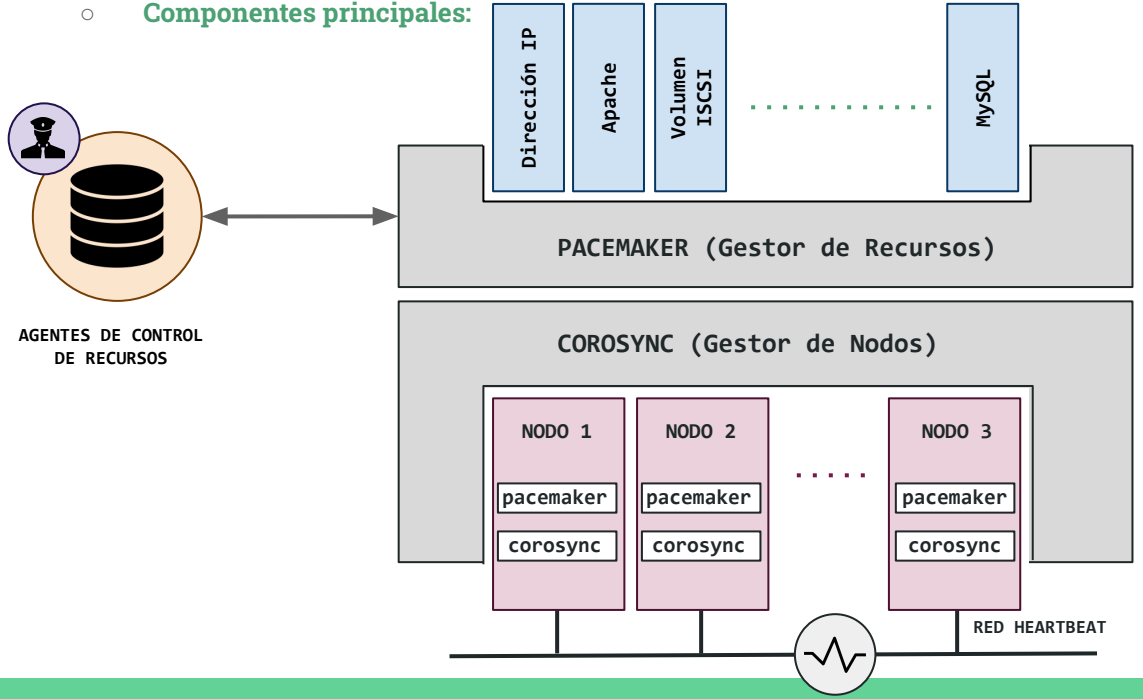
STONITH

- Aísla los nodos fallidos
- Monitorización constante de nodos
- Fuerza apagado/reinicio del nodo (hardware, software)



Alta Disponibilidad

- **LinuxHA:**
  - **Proyecto** de **código abierto** que provee la **infraestructura** para el **despliegue** de **clusters HA**
  - Disponible para **sistemas basados** en **UNIX** (GNU/Linux, FreeBSD, OpenBSD, Solaris y MacOS).
  - **Componentes principales:**



+ Responsable de gestión de las máquinas y servicios del cluster.
+ Inicio, parada y reinicio de componentes del cluster
+ Gestiona los mecanismos de quorum/stonith.

+ Encargado de la detección de nodos caídos mediante latidos (heartbeats).
+ Da soporte al intercambio de información en procesos del cluster

Computación de altas prestaciones

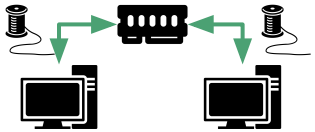
- **Introducción:**
  - Un **clúster HPC** es un **conjunto** de **computadoras interconectadas** (nodos) que **trabajan juntas** como una **única unidad** para **resolver** un **problema** de computación **intensiva**.
  - **Componentes** clave:
    - **Nodos** de **cálculo**: **computadoras** o servidores **individuales** que **ejecutan partes** del **cálculo**.
    - **Interconexión**: **redes** de **alta velocidad** y **baja latencia** (**InfiniBand**)
    - **Almacenamiento compartido**: **almacenamiento** centralizado para **manejar grandes volúmenes** de **datos**.
  - **Usos**:
    - Simulaciones físicas
    - **Bioinformática**
    - Predicción climática
    - **Inteligencia Artificial**
    - **Análisis** de **datos masivos** (Big Data).

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	

Computación de altas prestaciones

- **Programación Paralela en HPC:**
  - **Dividir tareas computacionales** en **subtareas más pequeñas** que **pueden ejecutarse** en **paralelo en** diferentes **nodos** o **procesadores**.
  - **Modelos** de **programación**:
    - **Memoria compartida**: todos los **procesadores acceden** a la **misma memoria** (OpenMP, MPICH)
      - **OpenMP**
        - **API** que permite **paralelizar código** en sistemas de memoria **compartida**

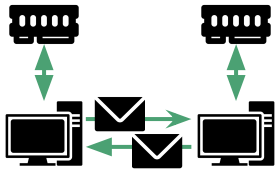
```
#pragma omp parallel for
for (int i = 0; i < 100; ++i) {
    // Código a paralelizar
    array[i] = i * i;
}
```



        - Aplicación:
          - Cálculos científicos que requieren una comunicación compleja y personalizada.
          - Simulaciones numéricas.
          - Aplicaciones donde se necesita un control preciso sobre la ejecución de los procesos.

Computación de altas prestaciones

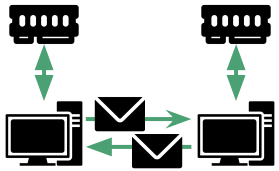
- **Programación Paralela en HPC:**
  - **Dividir tareas computacionales** en **subtareas más pequeñas** que **pueden ejecutarse** en **paralelo en** diferentes **nodos** o **procesadores**.
  - **Modelos** de **programación**:
    - **Memoria distribuida**: cada **nodo tiene** su **propia memoria** y se **comunican** entre **sí** (MPI, MapReduce).



      - **MPI** (Message Passing Interface)
        - **Intercambio** de **mensajes** → mecanismo para **coordinación** entre los **nodos** del clúster
        - **Alto nivel**: ofrece una **colección** de **primitivas** que **ocultan** los **detalles** (software y hardware).
      - **MapReduce**
        - **Modelo** de **programación paralela** orientado a **procesamiento** de **grandes volúmenes** de **datos**.
        - **Divide** un **problema complejo** en **tareas** más **pequeñas** y **sencillas**, que se **ejecutan** en **paralelo**.
        - **Procesamiento** de los datos **centrado** en dos **operaciones básicas**: ***map*** y ***reduce***

Computación de altas prestaciones

- **Programación Paralela en HPC:**
  - **Dividir tareas computacionales** en **subtareas más pequeñas** que **pueden ejecutarse** en **paralelo en** diferentes **nodos** o **procesadores**.
  - **Modelos** de **programación**:
    - **Memoria distribuida**: cada **nodo tiene** su **propia memoria** y se **comunican** entre **sí** (MPI, MapReduce).



      - **MPI** (Message Passing Interface)
        - **Intercambio** de **mensajes** → mecanismo para **coordinación** entre los **nodos** del clúster
        - **Alto nivel**: ofrece una **colección** de **primitivas** que **ocultan** los **detalles** (software y hardware).
      - **MapReduce**
        - **Modelo** de **programación paralela** orientado a **procesamiento** de **grandes volúmenes** de **datos**.
        - **Divide** un **problema complejo** en **tareas** más **pequeñas** y **sencillas**, que se **ejecutan** en **paralelo**.
        - **Procesamiento** de los datos **centrado** en dos **operaciones básicas**: ***map*** y ***reduce***

Computación de altas prestaciones

- **Programación Paralela en HPC:**
  - **Dividir tareas computacionales** en **subtareas más pequeñas** que **pueden ejecutarse** en **paralelo en** diferentes **nodos** o **procesadores**.
  - **Modelos** de **programación**:
    - **Memoria distribuida**: cada **nodo tiene** su **propia memoria** y se **comunican** entre **sí** (MPI, MapReduce).
      - **MapReduce**
        - **Fases:**

`Map`

- Transforma el input en pares clave-valor.
- Dependiente del input.
- Conceptualmente ≈ ConcurrentHashMap()<>

`Shuffle & Sort`

- Agrupa y ordena los pares por clave
- Valores asociados a una misma clave se incluyen en una lista

`Reduce`

- Procesa y combina los valores para producir el resultado final
- Personalizado en función del output que se necesite

Computación de altas prestaciones

- **Programación Paralela en HPC:**
  - **Dividir tareas computacionales** en **subtareas más pequeñas** que **pueden ejecutarse** en **paralelo en** diferentes **nodos** o **procesadores**.
  - **Modelos** de **programación**:
    - **Memoria distribuida**: cada **nodo tiene** su **propia memoria** y se **comunican** entre **sí** (MPI, MapReduce).
      - **MapReduce**
        - **Ejemplo:**

> ***Calcular el salario promedio de los empleados en cada departamento de una empresa.***

ID	Nombre	Departamento	Salario
1	Juan	Ventas	2500
2	Ana	TI	3000
3	Luis	TI	3500
4	María	Ventas	2800
5	Carlos	RRHH	2000
6	Rosa	RRHH	2200

Map

(Ventas, 2500)
(TI, 3000)
(TI,3500)
(Ventas, 2800)
(RRHH, 2000)
(RRHH, 2200)

Shuffle & Sort

Ventas → [2500,2800]
TI → [3000, 3500]
RRHH → [2000, 2200]

Reduce

Ventas

Suma: 2500+2800 = 5300
Promedio: 5300 / 2 = 2650

TI

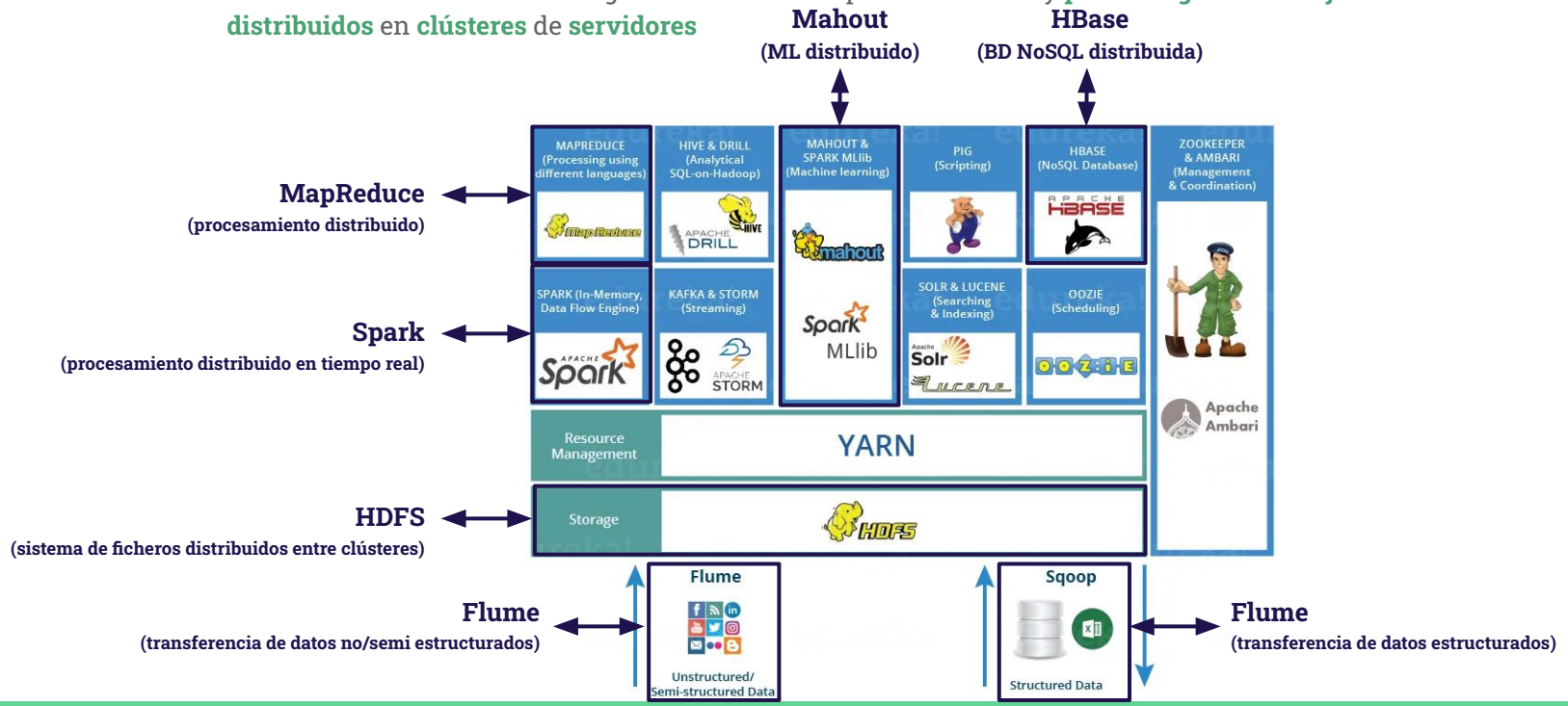
Suma: 3000+3500 = 6500
Promedio: 6500 / 2 = 3250

RRHH

Suma: 2000+2200 = 4200
Promedio: 4200 / 2 = 2100

Computación de altas prestaciones

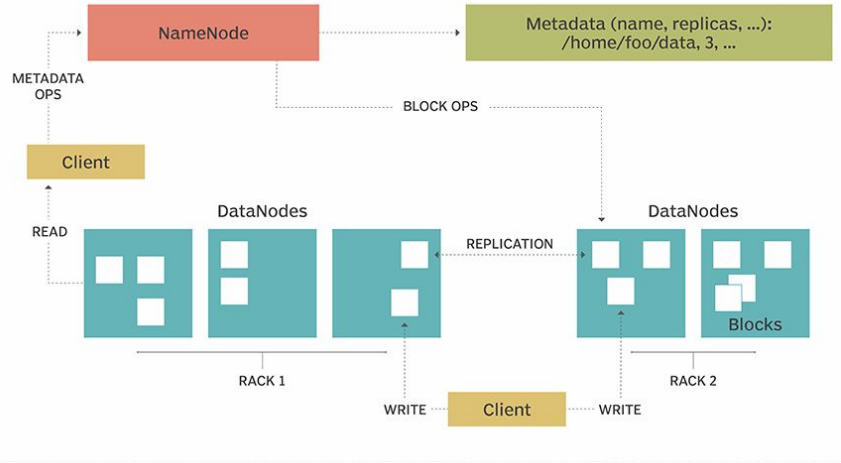
- **Hadoop**
  - **Plataforma** de software de código abierto **diseñada** para **almacenar** y **procesar grandes conjuntos** de **datos distribuidos** en **clústeres** de **servidores**



Computación de altas prestaciones

- **Hadoop**
  - **Arquitectura HDFS:**
    - Proporciona soporte para **almacenamiento distribuido**.
      - **Bloques grandes**, 64MB
      - **Múltiples réplicas** en **distintos nodos** del cluster
    - **NameNode** (uno por clúster):
      - Almacena **metadatos** de ficheros y de la distribución de los bloques.
    - **DataNodes** (en cada nodo del clúster):
      - Almacena los **bloques asignados**

Arquitectura HDFS



Universidade Vigo
ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Centros de Datos

T3 - Procesamiento en los Centros de Datos (II)

David Ruano Ordás
Departamento de Informática

drordas@uvigo.es

Despacho 409