



Sistemas Distribuidos I (75.74)

Data Intensive Applications

Particiones y Replicación. Distributed Shared Memory (DSM).
Distributed File Systems (NFS, HDFS). Big Table.

Docentes

- Pablo D. Roca
- Ezequiel Torres Feyuk

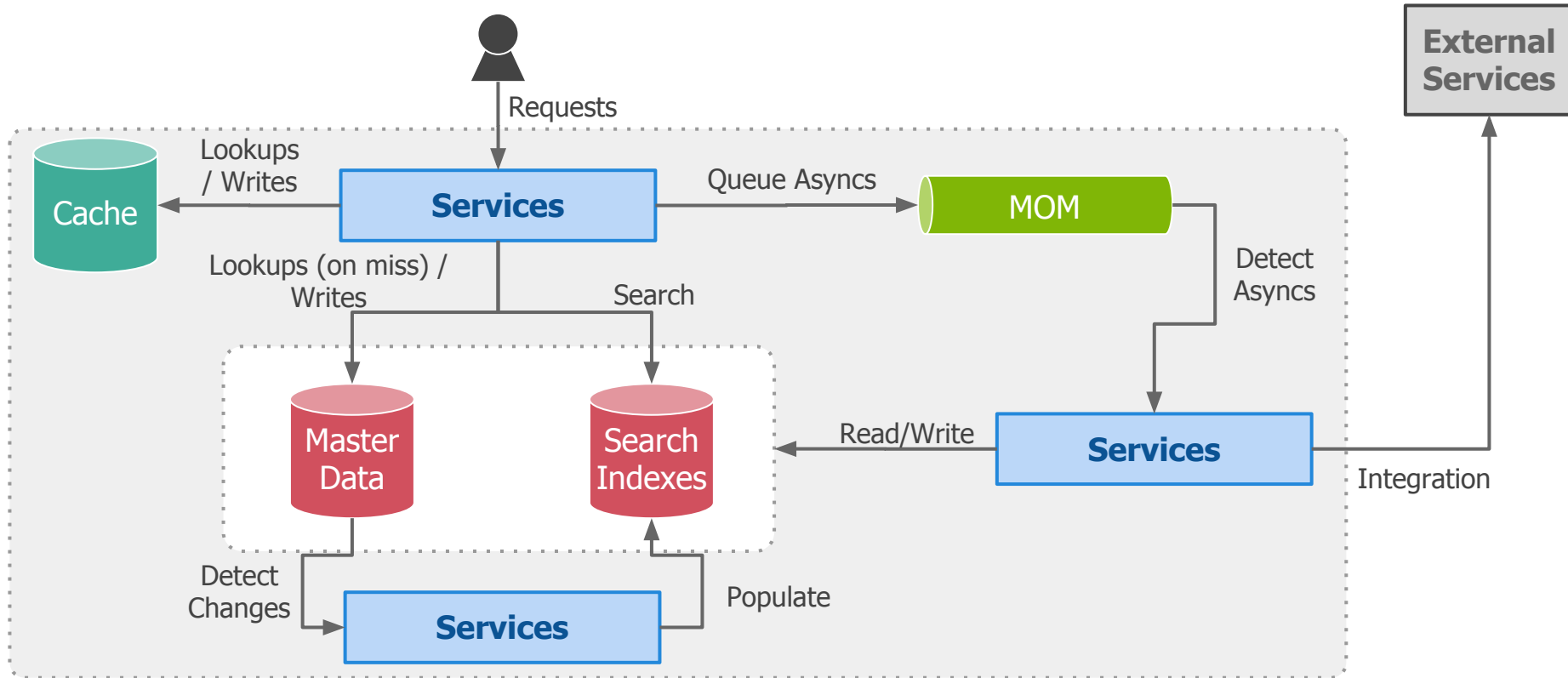


● **Datos en Sistemas de Gran Escala**

- Partición y Replicación
- Distributed Shared Memory (DSM)
- Distributed File Systems (DFS)
- Big Table



Datos en Sistemas de Gran Escala | Flujo de Datos





Master Data | Relacional vs NoSQL

Enfoques que nacen en '60s y '70s: *hierarchical, network & relational model*.

Relacional

- Predomina hasta 2010 con la estandarización de SQL. Almacenamiento en tablas y filas.
- Buen soporte para *joins*, relaciones *many-to-one* o *many-to-many*

No Relacional | Not Only SQL (NoSQL)

- A partir del 2010, la industria imponen otros almacenamientos: clave-valor, documentales, orientadas a grafos, columnares.
- Beneficios claros para modelos afines: sin relaciones, one-to-many (jerárquicos), alta conectividad (grafos).
- Se adapta mejor a modelos con esquemas (*schemas*) cambiantes o no definidos.



Master Data | Transaccional (OLTP) vs *Analytics* (OLAP)

Enfoques que nacen en '60s y '70s: *hierarchical, network & relational model*.

- **Online Transaction Processing (OLTP):** Orientado a unidades lógicas: grupos de *reads/writes* (o transacciones). No necesariamente ACID.
- **Online Analytics Processing (OLAP):** Orientado a analizar el conjunto todo de los datos.

	OLTP	OLAP
Patrón de Read	Pocos registros. Búsqueda por claves.	Agregación de muchos registros.
Patrón de Write	Acceso aleatorio. Registros pequeños.	Importaciones <i>batch</i> (ETLs) o <i>Streams</i> .
Uso Principal	Info maestra. Aplicaciones de usuarios.	Exploración de datos interno. Análisis estadístico.
Datos	Instantánea de los datos en el momento actual. Tamaños de GBs-TBs.	Histórico de los datos. Tamaños de TBs-PBs.



Almacenamiento | Relacional

- Normalmente: un archivo de almacenamiento por tabla.
- Relaciones entre tablas por *foreign-keys*
- Lectura de toda la fila para retornar proyecciones.

Tabla: Sales

date_id	product_id	quantity	price	discount
20100101	100	2	200.00	NULL
20100101	101	1	400.00	50.00
20100102	100	3	300.00	NULL
20100102	103	2	100.00	NULL
20100102	103	10	500.00	NULL

Tabla: Products

product_id	Name	price
100	Prod1	100.00
101	Prod2	450.00
103	Prod3	50.00



Almacenamiento | Columnar

- Normalmente: Un archivos de almacenamiento por columna.
- Lectura de cada columna para retornar proyecciones.
- Grandes beneficios para compresión, lectura y agregaciones.

Sales

Columna	Contenido
date_id	20100101,20100101, 20100102, 20100102, 20100102
product_id	100, 101, 100, 103, 103
quantity	2, 1, 3, 2, 10
price	200.00, 400.00, 300.00, 100.00, 500.00
discount	NULL, 50.00, NULL, NULL

Products

Columna	Contenido
product_id	100, 101, 103
name	Prod1, Prod2, Prod3
price	100.00, 450.00, 50.00



Almacenamiento | Cubos de Información

- Normalmente: mantienen vistas materializadas con pre-cálculos estadísticos.
- Se crean grillas agrupadas por diferentes dimensiones.
- Operaciones como SUM, COUNT, MAX, MIN, AVG se consultan a estos cubos.

Ej.: Cubo: date_id x product_id, quantity

		quantity				total
		1	2	3	10	
date_id	product_id	100	101	103	total	...
	20100101	200.00	400.00	0.00	600.00	...
	20100102	300.00	0.00	600.00	900.00	...
	total	500.00	400.00	600.00	1500.00	

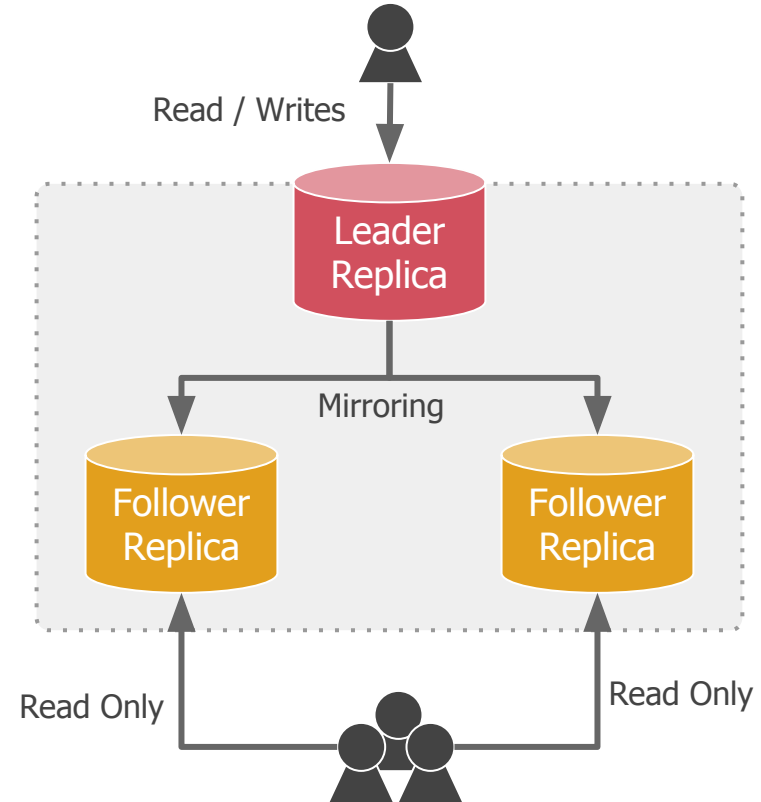


- Datos en Sistemas de Gran Escala
- **Partición y Replicación**
- Distributed Shared Memory (DSM)
- Distributed File Systems (DFS)
- Big Table



Replicación | *Leader based*

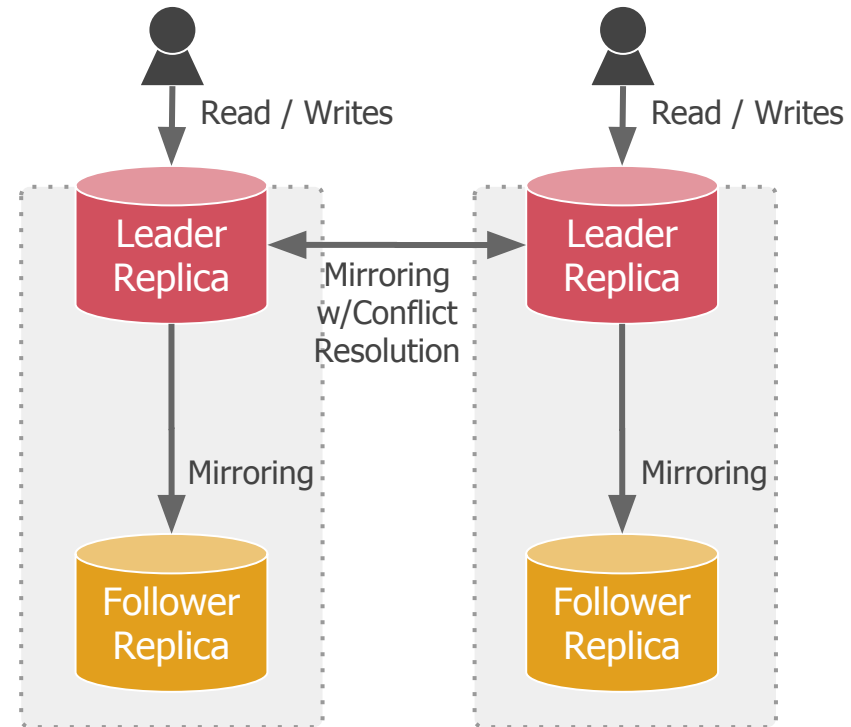
- Una réplica se designa como *master* o *leader*.
- Otras réplicas se designan *mirrors*, *slaves* o *followers*.
- Sólo se aceptan escrituras en el *leader*.
- Tanto *leader* como *followers* aceptan lecturas.
- La replicación puede ser síncrona o asíncrona.
- Problemas de la replicación:
 - *Read your own writes, Monotonic reads*





Replicación | *Multi-leader based*

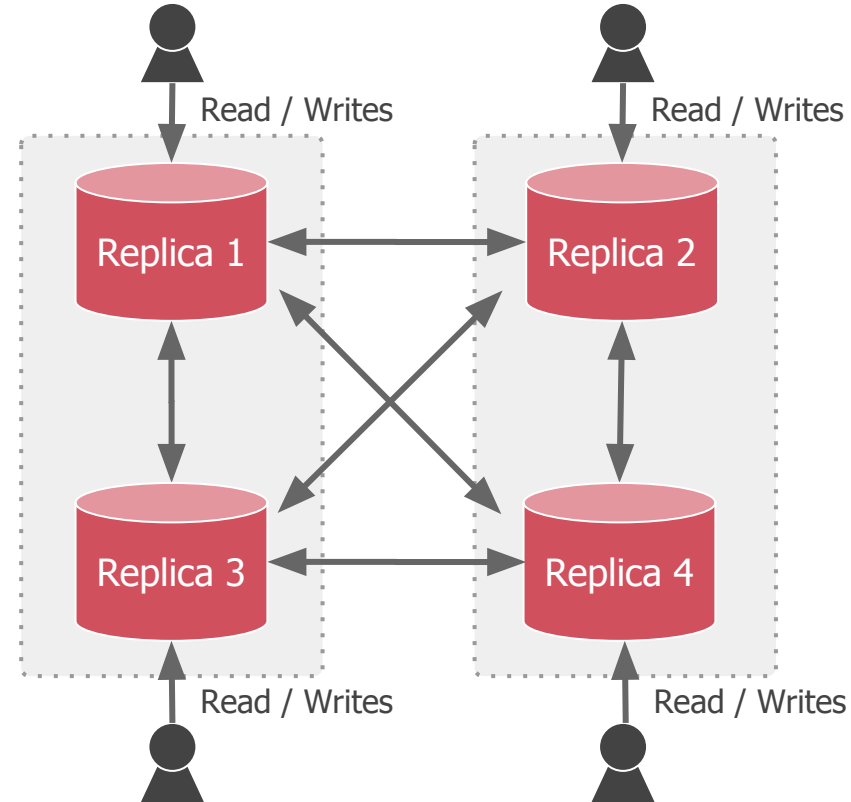
- Modelo normal en escenarios de múltiples data-centers.
- Frente a caídas de un data-center, se puede promover al otro como líder global.
- Problemas de la replicación:
 - Idem a leader based más la posibilidad de conflictos por concurrencia.
- Otros inconvenientes:
 - manejo de triggers, claves incrementales, integridad de relaciones.





Replicación | *Leaderless based*

- Sistema de replicación totalmente distribuido.
- Las réplicas deben sincronizarse mutuamente.
- Se puede definir topologías para la sincronización:
 - anillo, jerárquicas, todos contra todos
- Salvo que la información se encuentre particionada entre las réplicas, los conflictos son muy frecuentes.
- Otra alternativa es conseguir un consenso entre las réplicas para aplicar escrituras

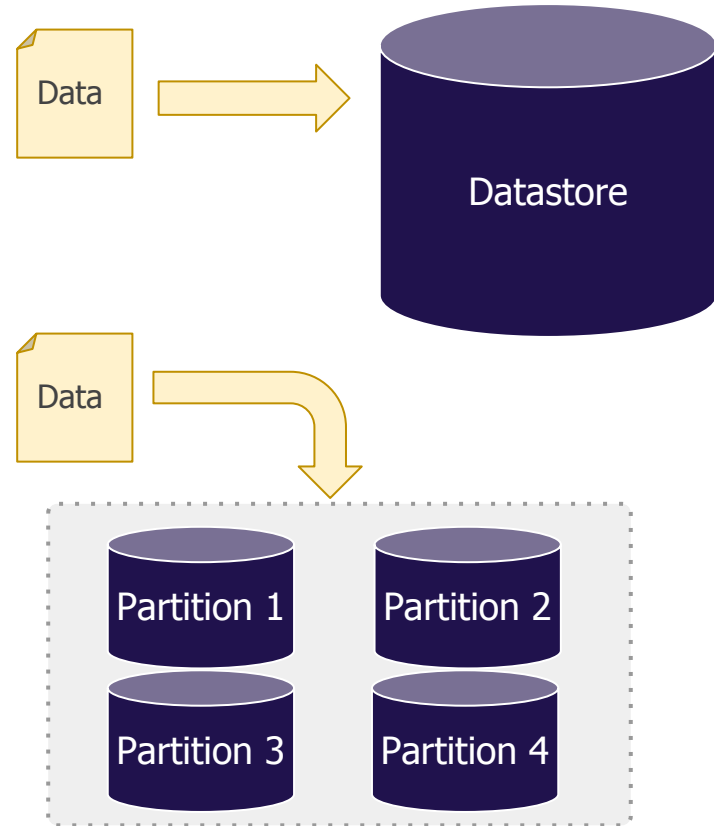




Particionamiento | Motivaciones

Distintos puntos de contacto con la estrategia de **Replicación**:

- *Performance*:
 - Velocidades de escritura
 - Velocidades de lectura
- Conflictos:
 - Evitar colisiones y/o resolución de conflictos
- Redundancia:
 - Permite recuperación frente a fallos





Particionamiento | Horizontal

- La información se segrega de a registros entre cada partición.
- El registro se encuentra en UNA partición a la vez.

date_id	product_id	quantity	price
20100101	100	2	200.00
20100101	101	1	400.00
20100102	100	3	300.00
20100102	103	2	100.00
20100102	103	10	500.00

date_id	product_id	quantity	price
20100101	100	2	200.00
20100101	101	1	400.00

date_id	product_id	quantity	price
20100102	100	3	300.00
20100102	103	2	100.00
20100102	103	10	500.00



Particionamiento | Vertical

- La información se segrega respecto de sus atributos/dimensiones/campos entre cada partición.
- El registro se encuentra en TODAS las particiones.

date_id	product_id	qty	price	disc
20100101	100	2	200.00	NULL
20100101	101	1	400.00	50.00
20100102	100	3	300.00	NULL
20100102	103	2	100.00	NULL
20100102	103	10	500.00	NULL

				date_id	product_id	disc
date_id	product_id	qty	price	100	NULL	
				101	50.00	
20100101	100	2	200.00	100	NULL	
20100101	101	1	400.00	103	NULL	
20100102	100	3	300.00	103	NULL	
20100102	103	2	100.00			
20100102	103	10	500.00			

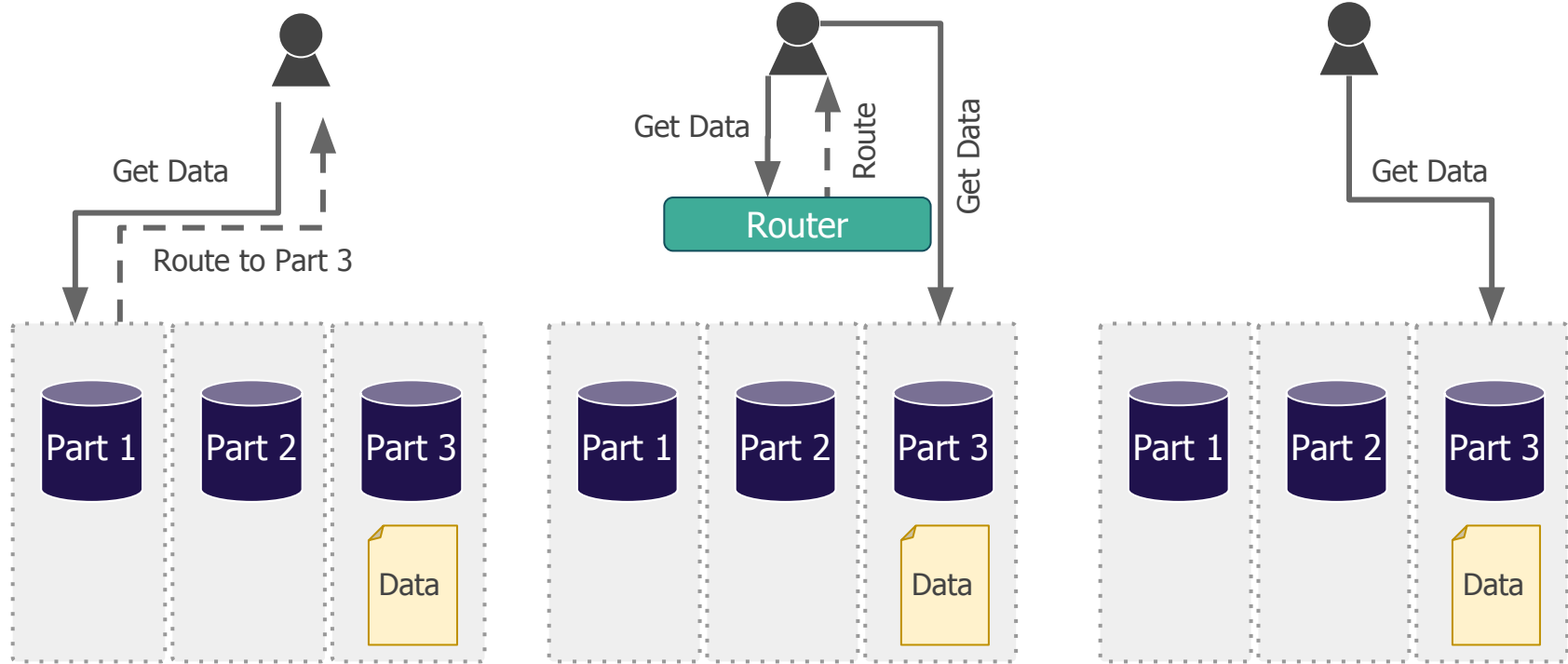


Particionamiento | Función de Partición

- Por *Key-Value*
- Por *Key-Range*
- Por *Hash-of-Key*
- Mixtos:
 - Generar N *shards* por cada *key*
 - Partición por claves secundarias



Particionamiento | Enrutamiento





- ☐ Datos en Sistemas de Gran Escala
- ☐ Partición y Replicación
- ☒ **Distributed Shared Memory (DSM)**
- ☐ Distributed File Systems (DFS)
- ☐ Big Table



Distributed Shared Memory (DSM)

Objetivo

- Brindar la *ilusión* de una memoria compartida centralizada

Ventajas

- Muy intuitivo para el desarrollo de sistemas distribuidos: Los algoritmos no distribuidos pueden ser traducidos fácilmente
- Información compartida entre nodos sin que requieran conocerse

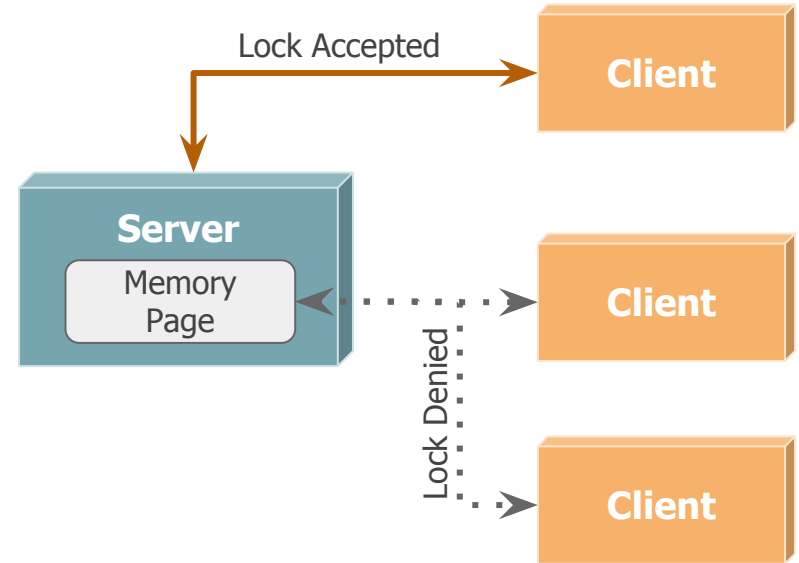
Desventajas

- Desalienta la distribución, genera latencia, cuello de botella y punto único de falla (arquitectura cliente-servidor)



DSM | Enfoque *Naïve*

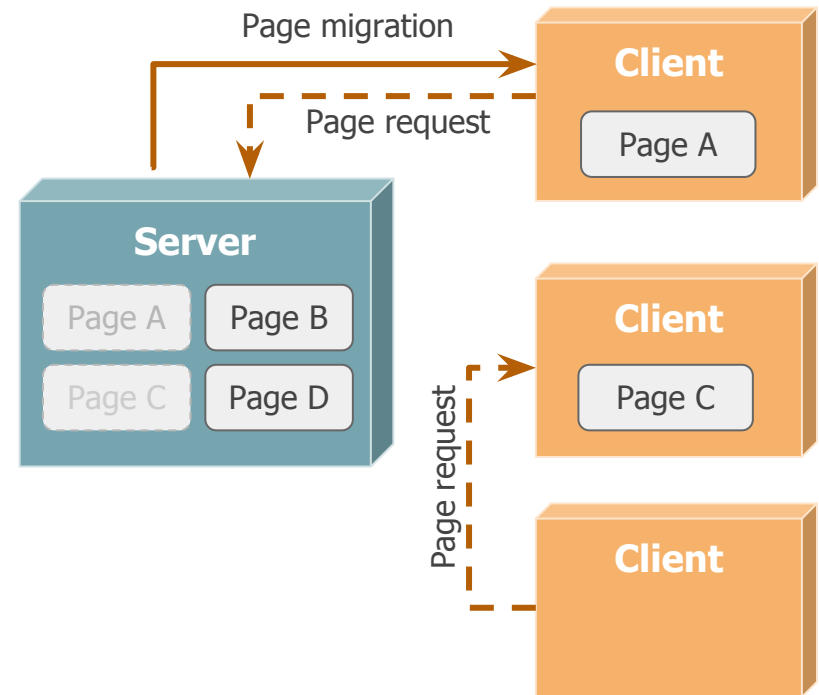
- La información es almacenada en memoria por el servidor
- Los clientes acceden mediante requests a escribir o leer las páginas
- El servidor puede garantizar la consistencia muy fácilmente serializando los requests
- Muy baja *performance* para para las aplicaciones cliente





DSM | Migración de *Memory Pages*

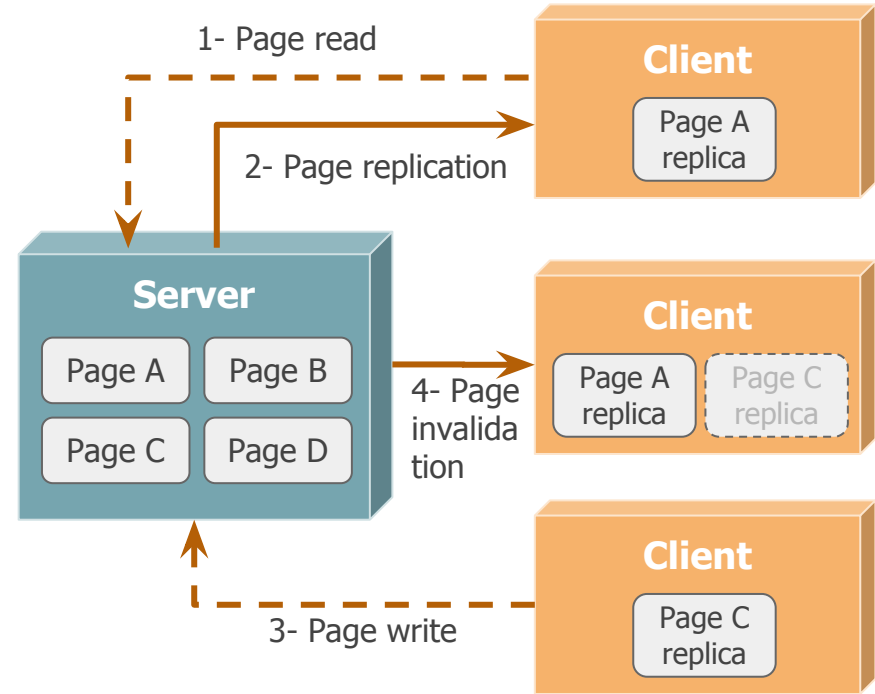
- La información es almacenada en memoria por el servidor y delegada en los clientes
- Los clientes pueden optimizar la localidad de acceso pidiendo una *memory page* prestada
- Otros clientes puede pedir la misma página y quedar bloqueados, salvo que se permita una sub-delegación
- Garantiza consistencia. No se accede concurrentemente a las páginas





DSM | Replicación de *Memory Pages* (solo lectura)

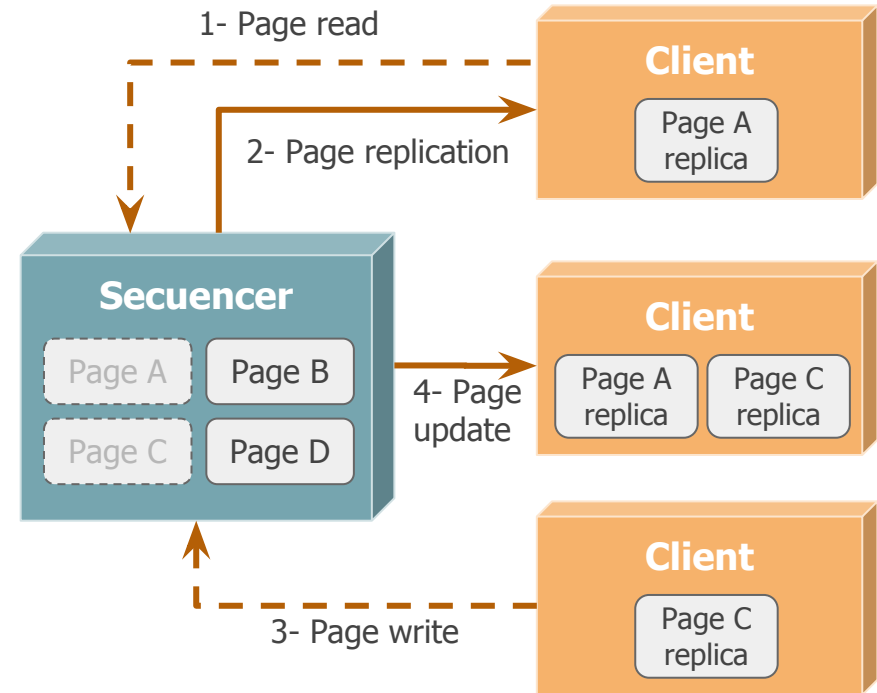
- Favorece escenarios con muchas lecturas y pocas escrituras
- Las escrituras son coordinadas por el servidor
- Las lecturas implican una replicación de la página en modo *read-only*
- El servidor invalida las réplicas frente a cambios





DSM | Replicación de *Memory Pages* (lectura-escritura)

- El servidor mantiene las páginas de memoria hasta que los clientes las requieren
- Los clientes toman control total de las réplicas
- El servidor se transforma en un secuenciador de las operaciones
- El servidor aplica también los cambios ante caídas de los clientes





- Datos en Sistemas de Gran Escala
- Partición y Replicación
- Distributed Shared Memory (DSM)
- **Distributed File Systems (DFS)**
- Big Table



Distributed File Systems (DFS) | Motivaciones

- Compartir archivos en redes locales e intranets
- Poseer un esquema centralizado de información persistente
 - Control de Backups
 - Control de acceso y monitoreo
- Optimización de recursos por la concentración:
 - Discos de mayor capacidad permitían economizar
 - El costo de administración se reduce



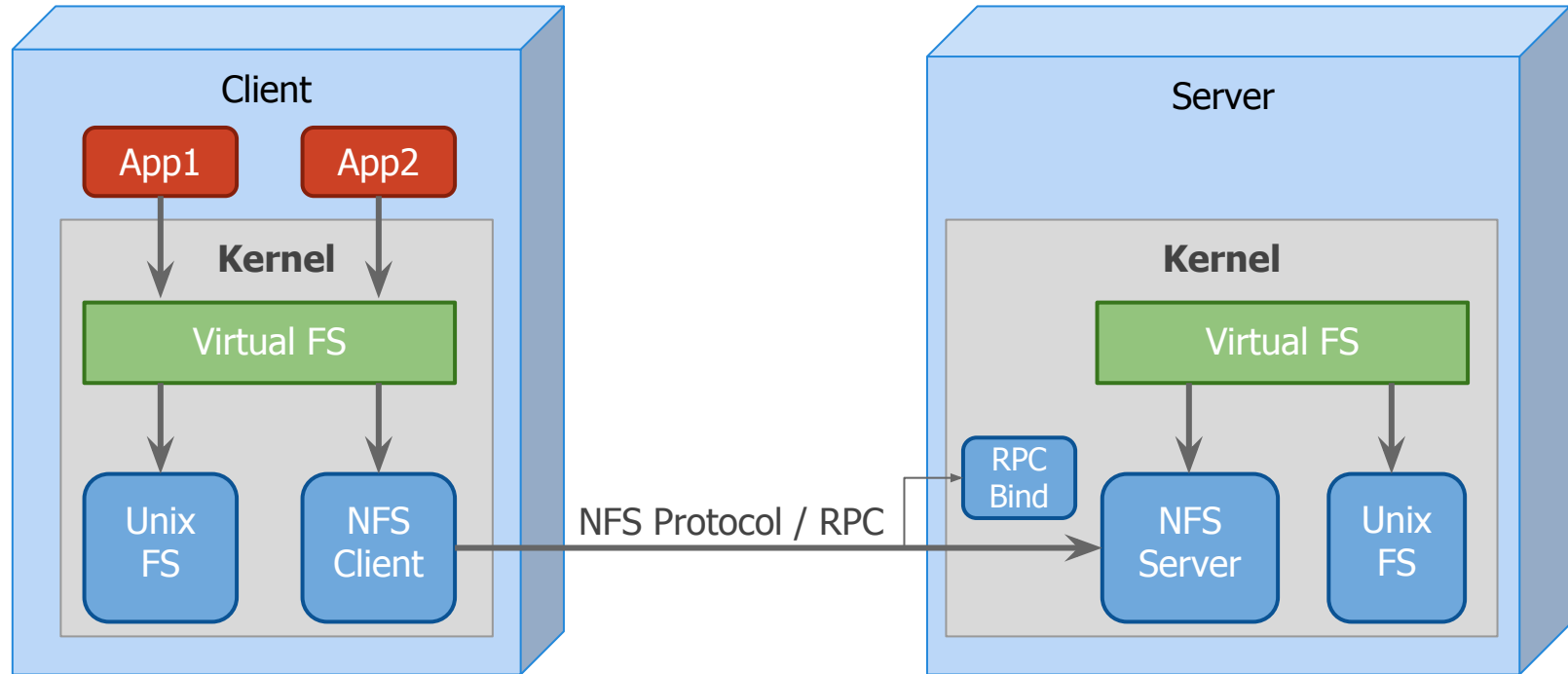


- **Transparencia a los clientes:**
 - Acceso: obtención de los recursos con credenciales usuales
 - Localización: operación de los archivos como si fueran locales
 - Movilidad: el movimiento interno de archivos no debe ser percibido
 - Performance y Escala: las optimizaciones no deben afectar al cliente
- **Concurrencia:** el acceso concurrente no debe requerir operaciones particulares al cliente
- **Heterogeneidad de Hardware:** adaptación automática a diferentes HWs
- **Tolerancia a Fallos:** capacidad de ocultar o minimizar los fallos (permitir operaciones como *at-least-once* o *at-most-once*)



Caso de Estudio | Network File System (NFS)

- Diseñado para ser independiente de plataformas (pero desarrollado sobre UNIX)
- Primera versión en 1984 por Sun Microsystems
- Requiere una nueva abstracción en el kernel: Virtual File System
- Arquitectura de cliente-servidor utilizando RPC sobre TCP o UDP
- Las aplicaciones utilizan el VFS para acceder a los archivos, lo que requiere una invocación remota
- Los servidores proveen operaciones idénticas a las requeridas por Posix:
 - Soporta ser montado como una unidad virtual





NFS | Instalación Servidor-Cliente

Server

```
apt-get install nfs-kernel-server
vim /etc/idmapd.conf
...
[Translation]
Method = nsswitch
...
vim /etc/exports
/export 192.168.1.0/24 (...)
```

Client

```
apt-get install nfs-common
mount -t nfs -o proto=tcp,port=2049
<nfs-server-IP>:/ /mnt
```



Caso de Estudio | Hadoop DFS (HDFS)

- Sistema de archivos distribuido diseñado para utilizar hardware de bajo costo
- Implementación de Apache basada en el diseño de Google File System (GFS)
- No soporta POSIX por lo que se lo considera un Data Storage en lugar de FS
- Base del ecosistema de tecnologías Hadoop para procesamiento distribuido.





HDFS | Factores de Diseño

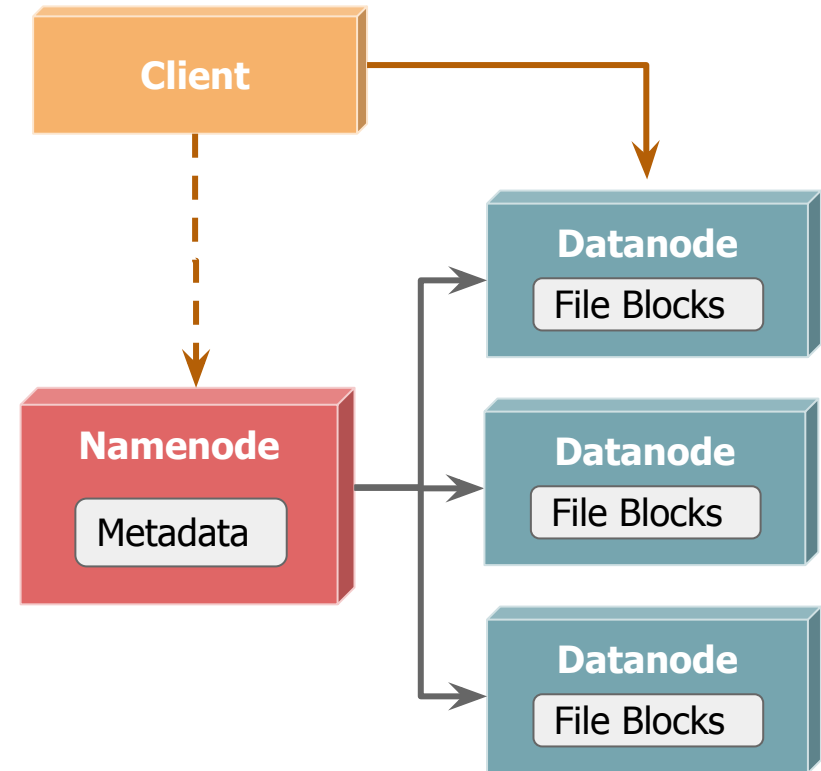
- **Tolerancia a Fallos:** Los fallos de HW son normales. Es más económico adaptarse que defenderse
- **Volumen y Latencia:** Favorece las operaciones de *streaming* y los archivos volumétricos frente operaciones de usuarios finales de baja latencia
- **Portabilidad:** Preparado para ser utilizado en hardware de bajo costo. Utiliza TCP entre servidores y RPC con clientes
- **Performance:** Favorece operaciones de lectura. Política de *write-once-read-many*

"Moving Computation is Cheaper than Moving Data"



HDFS | Arquitectura

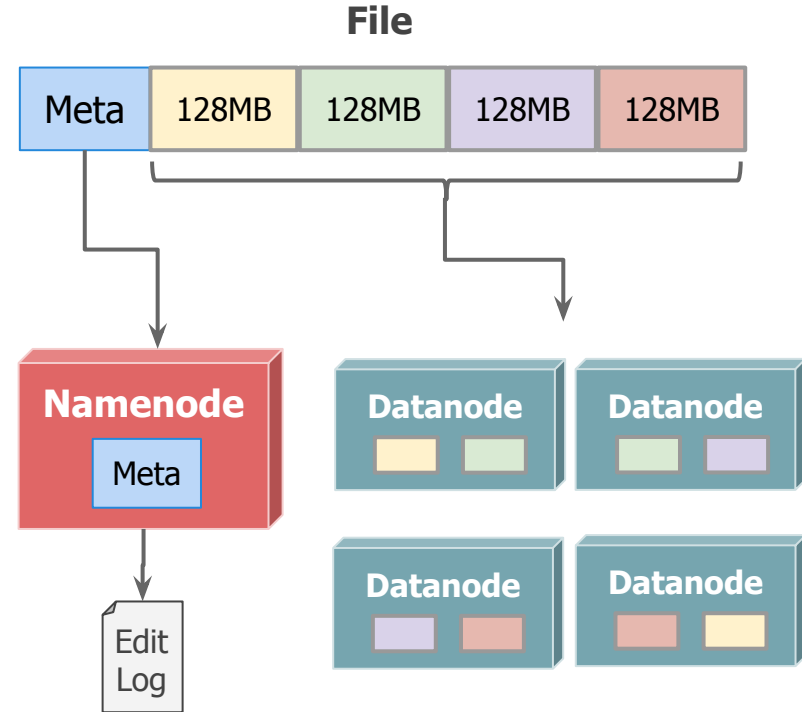
- Arquitectura maestro-esclavo
- **Namenode:** Contiene la información de metadata de archivo. Coordina a los Datanodes
- **Datanodes:** Almacena los datos de archivo.
- Los clientes consultan al Namenode por el 'File system' y ubicación de los datos.
- Los clientes se comunican luego con Datanodes para obtener la información





HDFS | Almacenamiento de Datos

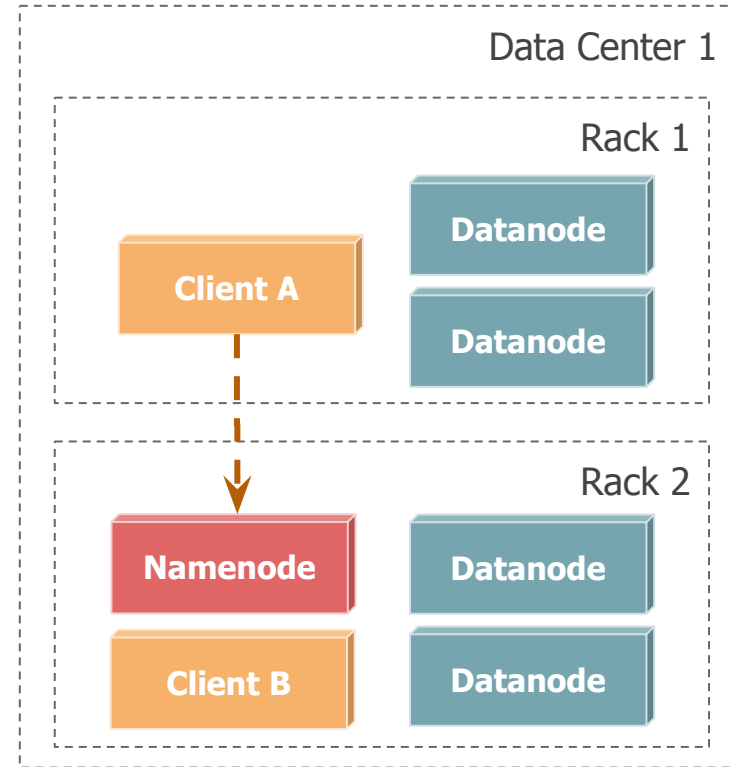
- Los archivos se particionan en bloques de 128MB
- Los bloques son replicados en distintos Datanodes
- El Namenode mantiene el listado de Datanodes para un archivo
- La metadata es mantenida en memoria para optimizar acceso, con un log de transacciones
- El cluster de Datanodes permite re-balanceo de bloques





HDFS | Acceso a Datos

- El Namenode favorece el principio de localidad de datos para el cliente
- El cliente recibe listado de Datanodes para cada bloque y sus réplicas
- Se intenta obtener los bloques desde el mismo rack.
- Si no es posible, desde el mismo datacenter



Agenda



- Datos en Sistemas de Gran Escala
- Partición y Replicación
- Distributed Shared Memory (DSM)
- Distributed File Systems (DFS)
- **Big Table**



Claves, datos y columnas

- Solo almacena pares clave-datos.
- Los datos son en realidad un conjunto de valores (o columnas).
- Al tratarse de conjuntos dispersos (*sparse*), los valores no se almacenan en un orden definido sino que conocen su 'column family'.

Tablets

- Conjunto de filas consecutivas de acuerdo a la clave.
- Unidad de balanceo de BigTable. Permite escalar el sistema.

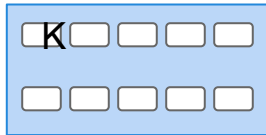


Key: 20100101, [Column: Product_id, Data: 100, Column: Quantity, Data: 2, Column: price: 200.00]

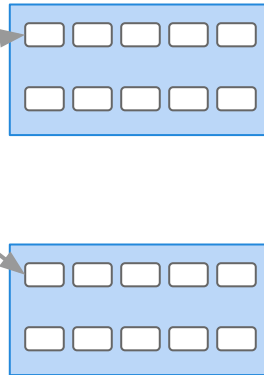
date_id	product_id	quantity	price
20100101	100	2	200.00
20100101	101	1	400.00
20100102	100	3	300.00
20100102	103	2	100.00
20100102	103	10	500.00



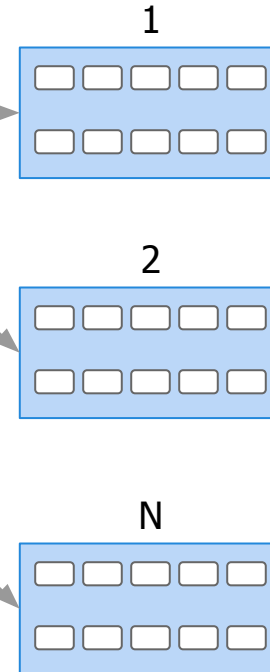
**Root Metadata
(1st level tablets)**



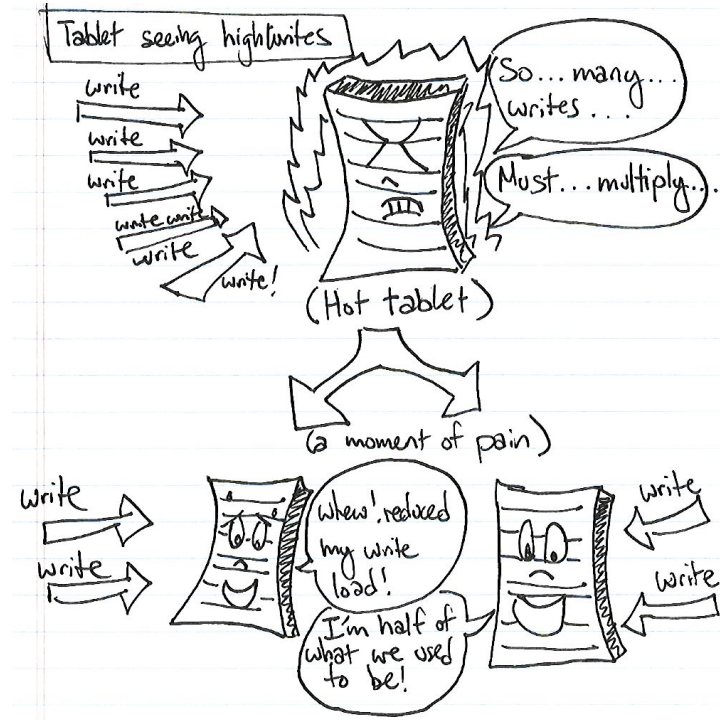
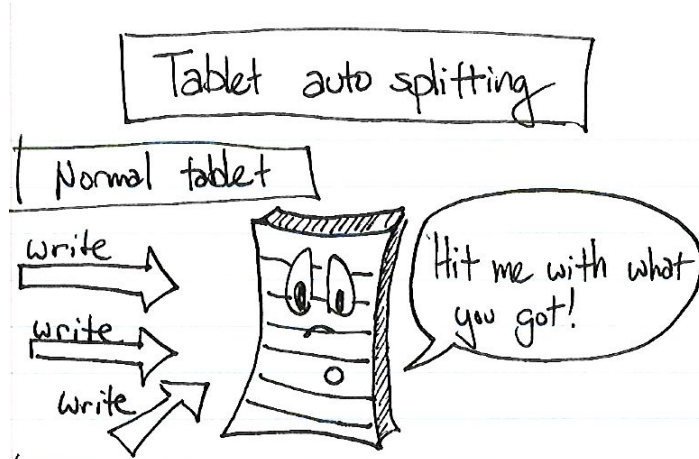
(2nd level tablets)



**User Level Table
(3er level tablets)**



Solo 3 niveles. Similar a Árboles B+.



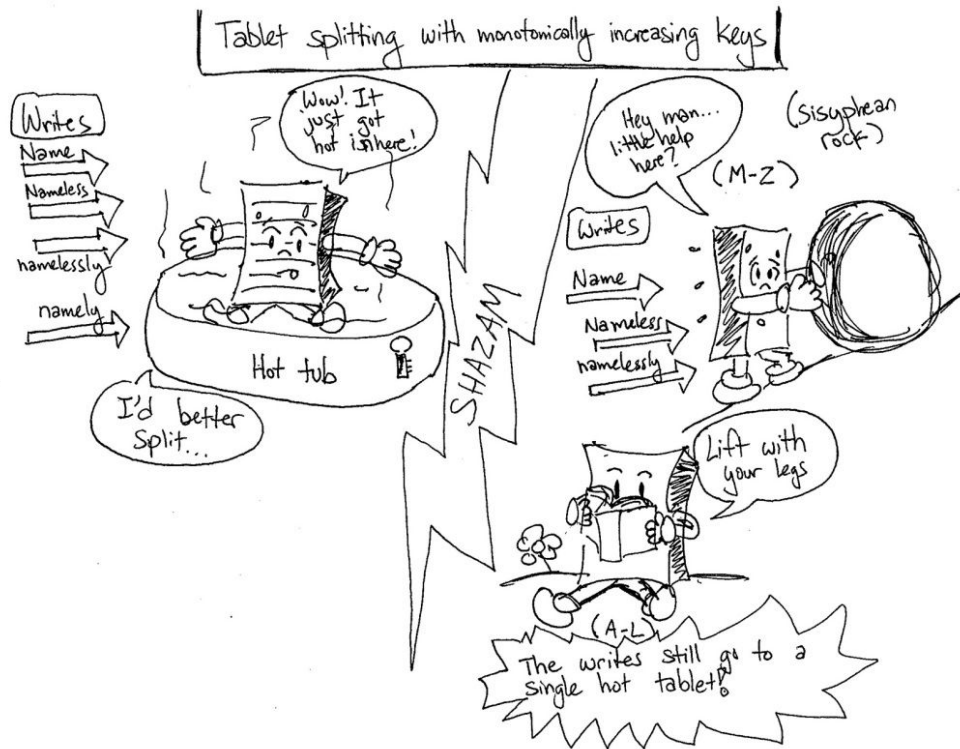
Source:
<https://ikaisays.com/2011/01/25/app-engine-database-tip-monotonically-increasing-values-are-bad/>, Ikai Lan

BigTable | División de Tablets Exitoso



Source:
<https://ikaisays.com/2011/01/25/app-engine-datastore-tip-monotonically-increasing-values-are-bad/>, [Ikalai Lan](#)

BigTable | División de Tablets No Exitoso



Source:
<https://ikaisays.com/2011/01/25/app-engine-database-tip-monotonically-increasing-values-are-bad/>, Ikai Lan