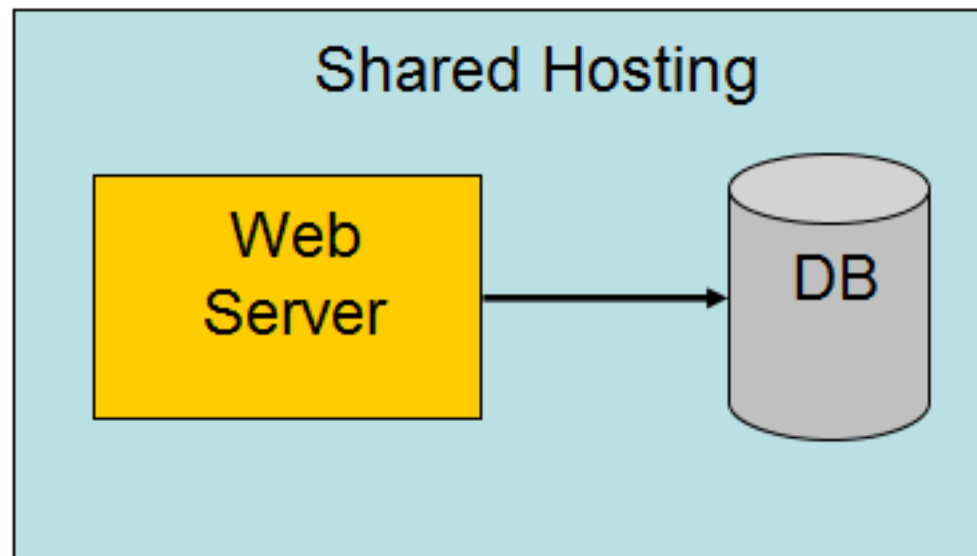


Scalarea Aplicatiilor Web

Andrei Gheorghe

www.idevelop.ro

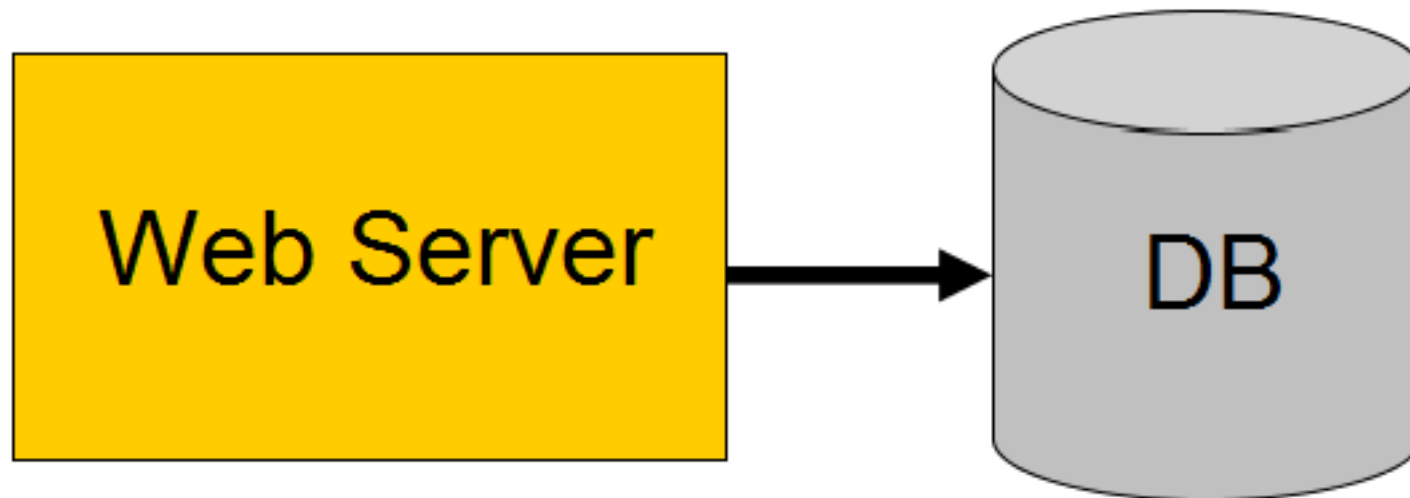
Plecam de jos



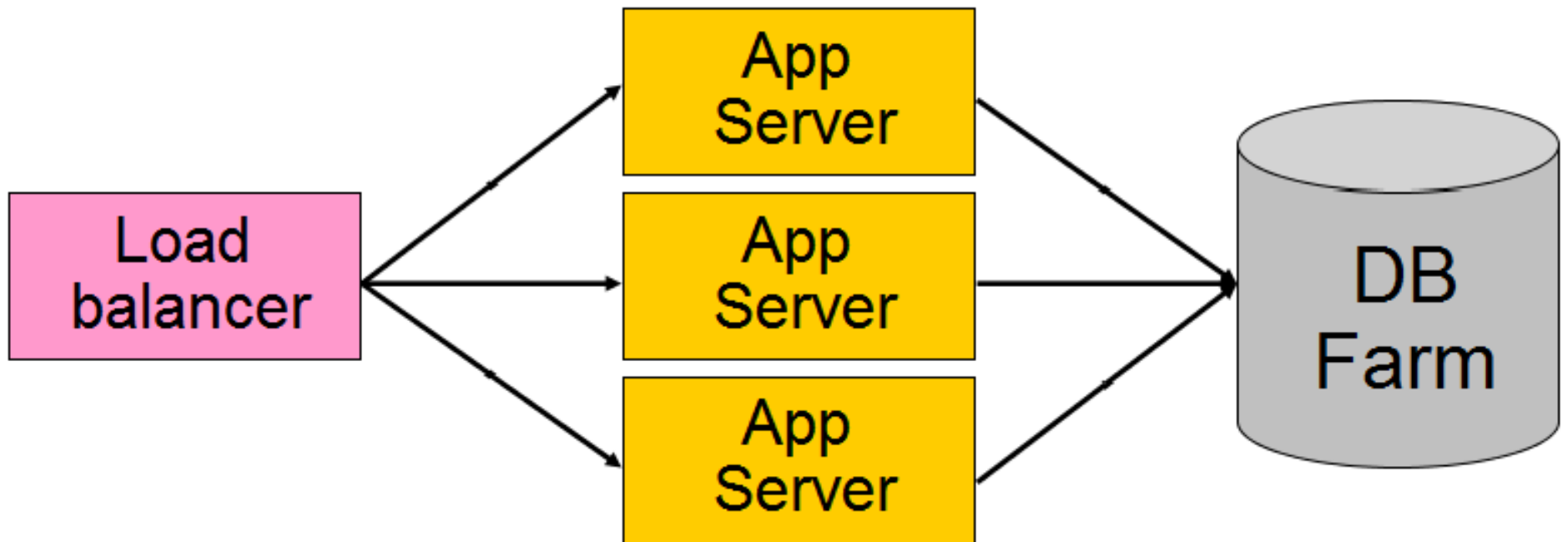
Unde apar probleme

- Puterea de procesare a serverului
CPU, RAM, etc
 - Nici o problema, RAM-ul e ieftin
- Banda, capacitate de stocare
 - Exista solutii relativ simple si pentru asta (NFS, S3)
- Baza de date
 - Mda...

Server Web + Server DB



Load Balancing



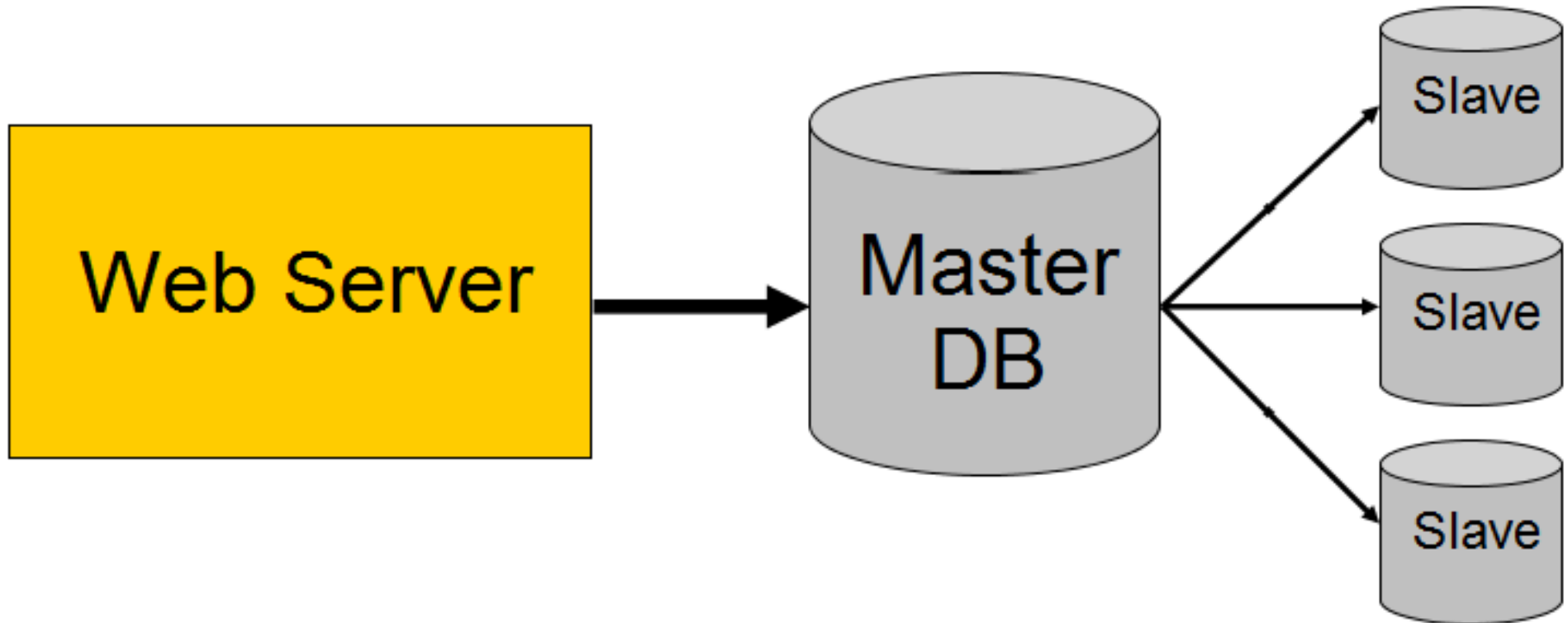
Load Balancing

- **Hardware**
 - Balancingul se face la nivel de transport pachete
 - Scump
- **DNS Load Distribution ("Round Robin")**
 - Statistic, distribuie traficul uniform
 - Nu stie nimic despre disponibilitatea serverelor
 - Pot aparea probleme de dns caching
- **Reverse Proxy - nginx, squid, lighttpd**

Relational Databases

tabele, coloane, joinuri

MySQL Replication



Normalizare / Denormalizare

- **USERS**

user_id, user_name, user_password

- **POSTS**

post_id, post_blog_id

- **COMMENTS**

c_id, c_post_id, c_text

Normalizare / Denormalizare

- **USERS**

user_id, user_name, user_password

- **POSTS**

post_id, post_blog_id, post_user_name

- **COMMENTS**

c_id, c_post_id, c_text

Normalizare / Denormalizare

- **USERS**

user_id, user_name, user_password

- **POSTS**

**post_id, post_blog_id, post_user_name,
post_comment_count**

- **COMMENTS**

c_id, c_post_id, c_text

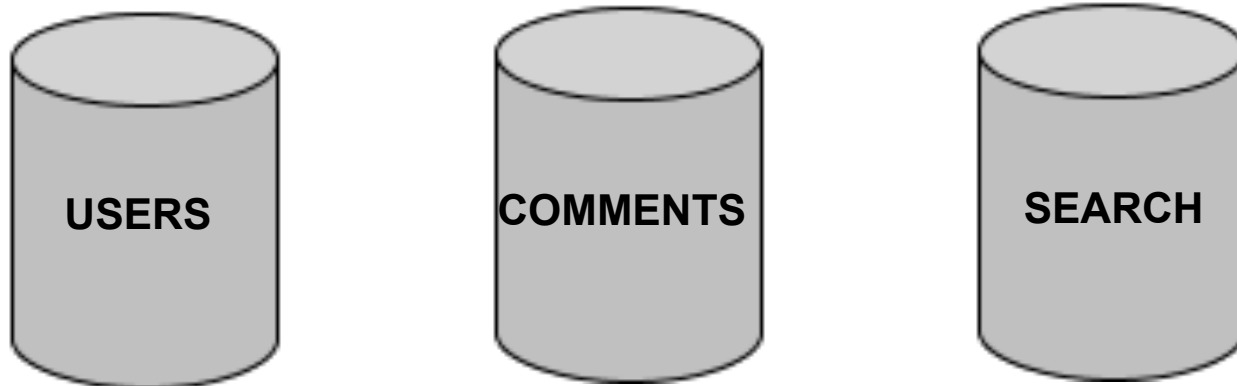
Key → Value Databases

- Distributed, persistent hash tables
 - "Eventual consistency"
- Permit SELECT-uri cu conditii
- Necesita o doza de denormalizare a datelor
 - Tratarea manuala a inconsistentelor, propagarea datelor corecte
- MemcacheDB, CouchDB, Amazon SimpleDB, Hypertable, Google BigTable

Sharding

Vertical Sharding

- Un server pentru useri, un server pentru search, etc
- JOIN-urile intre tabele se fac manual
 - Denormalizarea DB reduce nevoia de JOIN-uri



Horizontal Sharding

- Impartirea inregistrarilor dintr-un tabel intre mai multe servere
- Algoritmul de impartire este foarte important
 - in functie de algoritmul ales, reechilibrarea datelor in cazul modificarii topologiei poate fi dificila
- Se poate folosi un dictionar central
 - algoritm transparent
 - mai usor de reechilibrat
 - poate crea SPF



Avantajele sharding-ului

- High availability.
 - Daca un server crapa, aplicatia continua sa functioneze
- Faster queries.
 - Query-urile fiind pe bucati mai mici de date se executa mai repede
- More write bandwidth.
 - Scrierile se executa mai repede deoarece, neavand un server central, se executa in paralel

Cache

memcached

```
memcached -d -u www -m 2048 -l 10.0.0.8 -p 11211
```

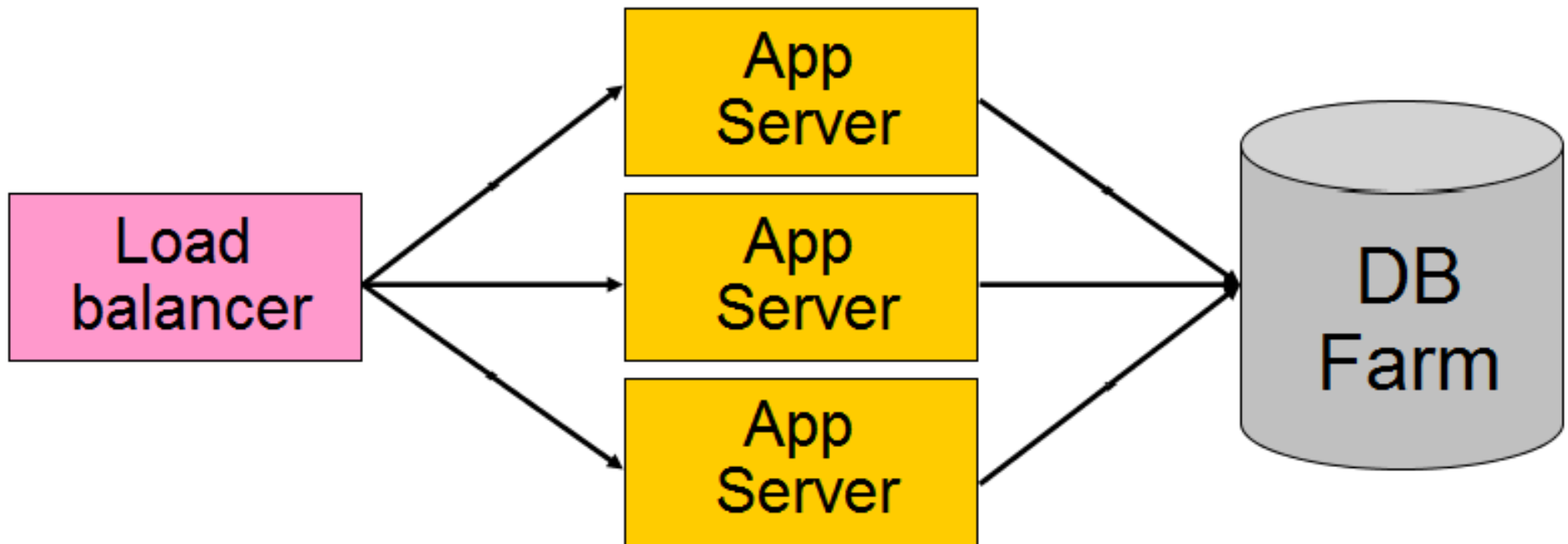
- Hash table distribuit, pastrat in RAM
 - `set(key,value)`
 - `get(key)`
 - `delete(key)`
- **value** este de obicei un intreg obiect serializat
Ex: articol+comentarii+informatii autor

memcached

- "Least Recently Used"
- Intr-o retea cu mai multe servere, instantele de memcached pot fi legate intre ele pentru a forma un cluster memcache in care cache-ul este replicat pe mai multe noduri
- memcached ruleaza pe Linux, Windows, poate fi pornit oriunde exista RAM liber

Session Clustering

Load Balancing Revisited



Session Clustering

- **Store in common filesystem**
 - Not useful in multi-server environments
 - NFS will cache pages
- **Store in database**
 - Very fast because you are only ever looking up primary keys
 - Make sure the DB has row locking (InnoDB), not table locking.
- **Store in memcached**
 - Stored across several machines rather than just one.
 - A total machine failure now affects only a percentage of users rather than everyone.

Amazon Web Services

- Simple Storage Service (S3)
- Elastic Compute Cloud (EC2)
- SimpleDB
- Simple Queue Service (SQS)

thank you, come again