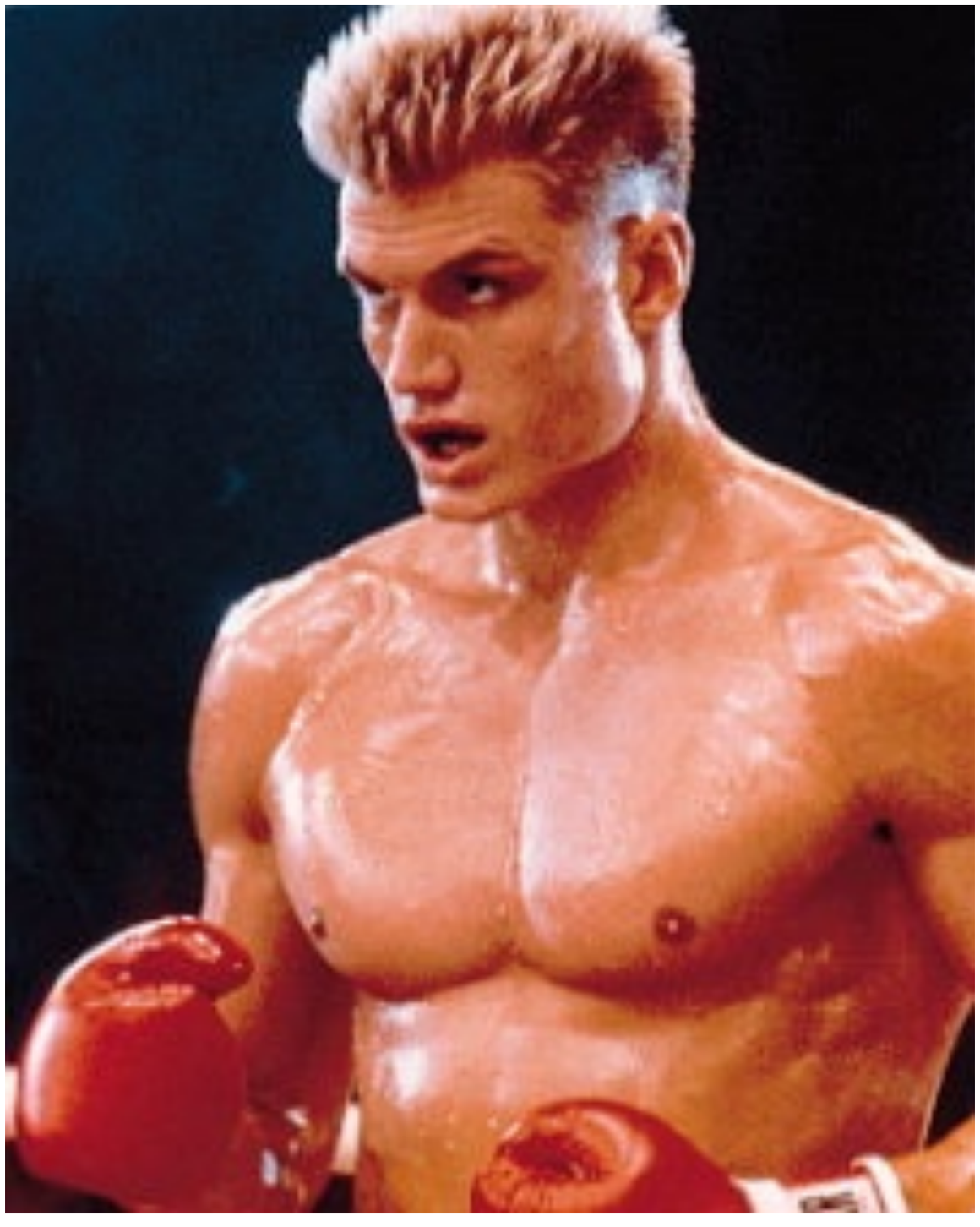
Ground up Introduction to In-memory Data (Grids)

Tweet about this
#hazelcast #imdg #devnexus

Why you here?

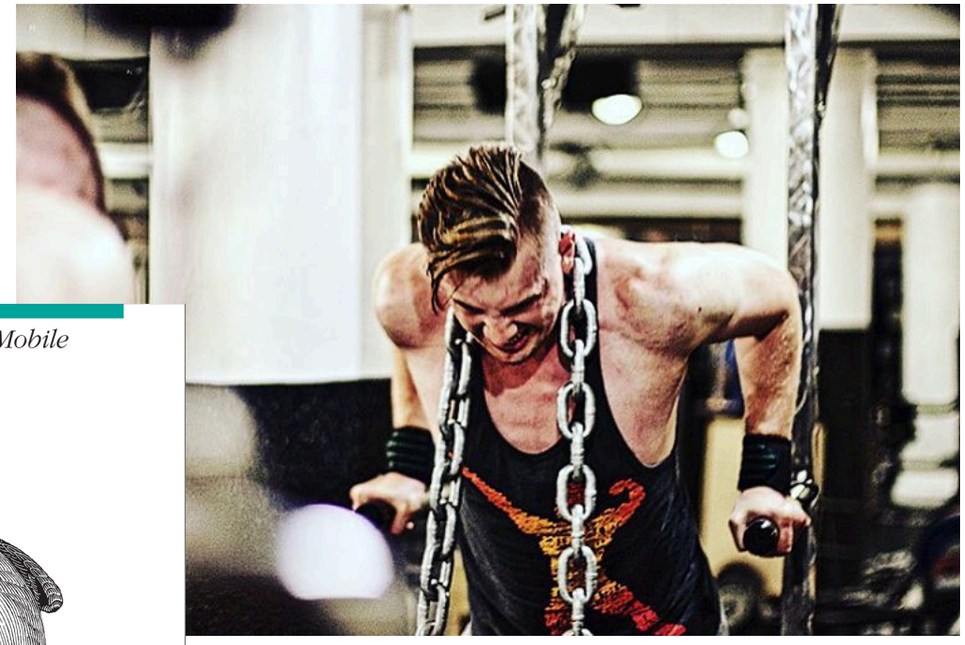

Java Developer on a quest for scalability frameworks
Architect on low-latency projects
NoSQL practitioner
IMDG newbie
Data enthusiast

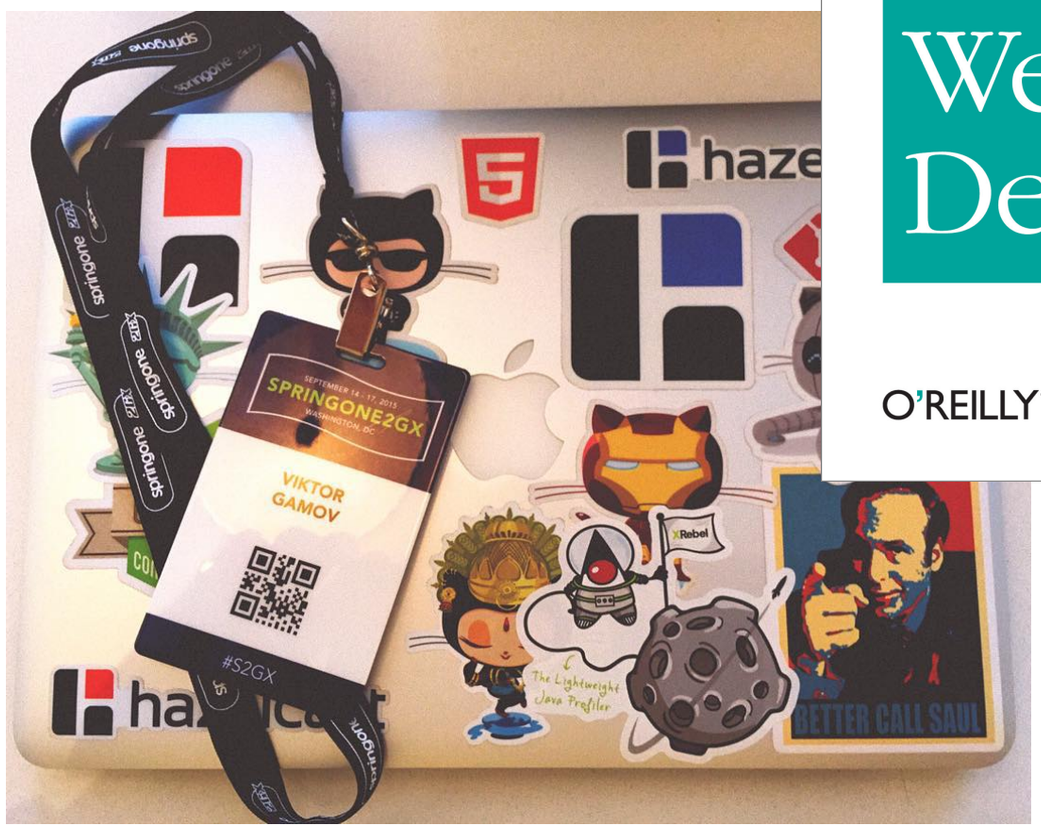
Who am I ?



Viktor Gamov

@gamussa





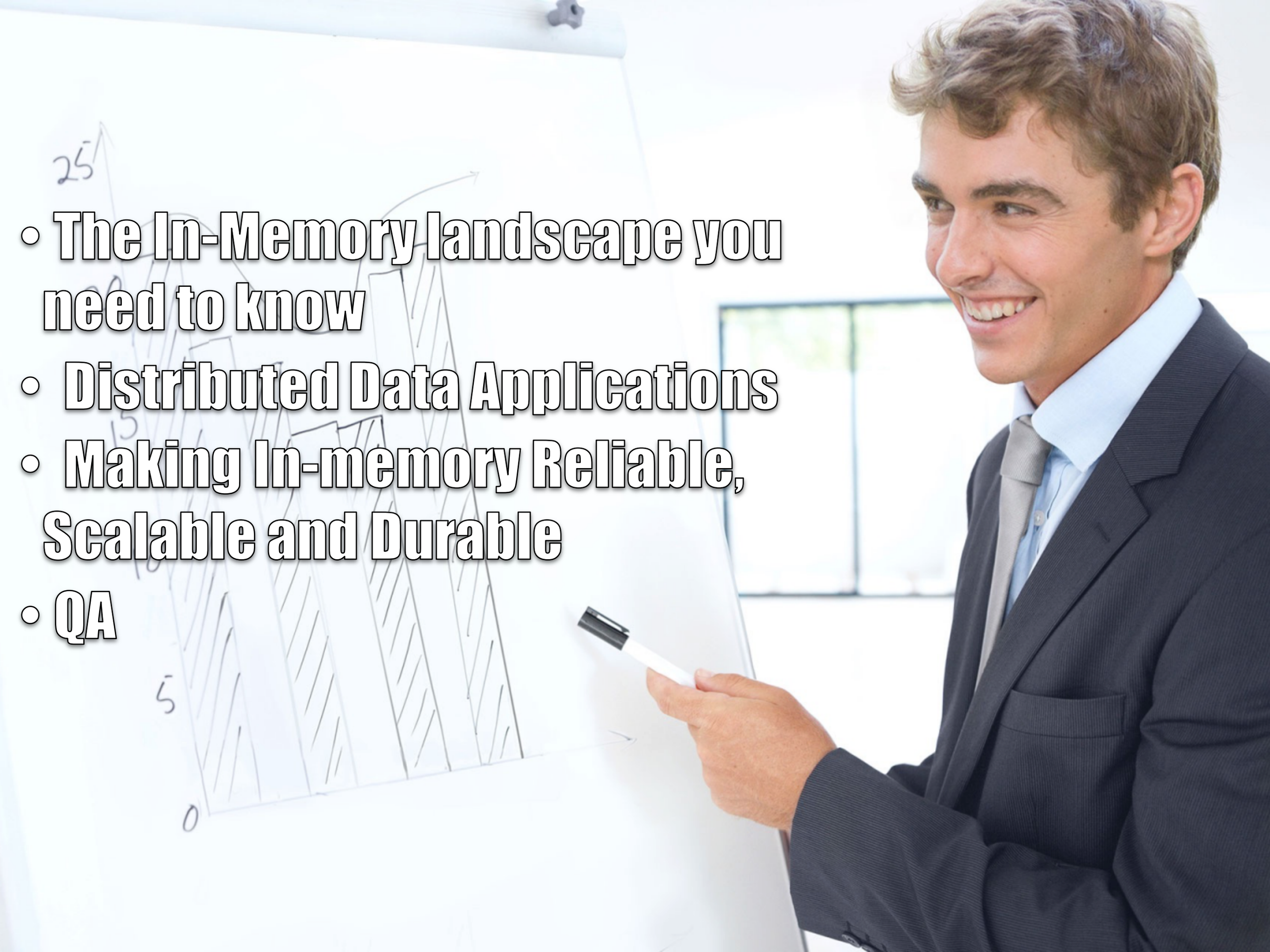
hitect at Hazelcast

nal Conferences

peaker

- **Co-author of O'Reilly's «Enterprise Web Development»**

Agenda

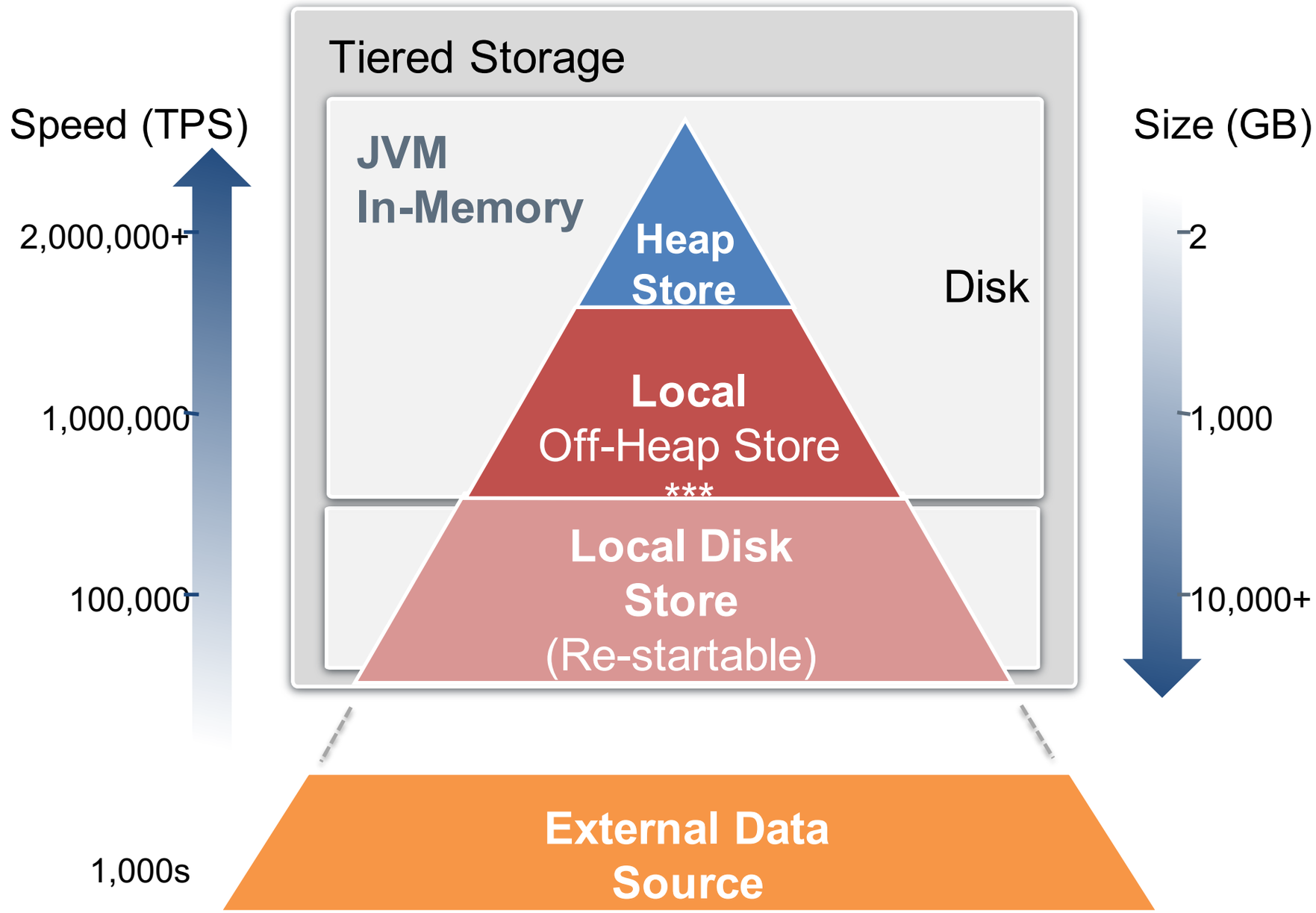


- The In-Memory landscape you need to know
- Distributed Data Applications
- Making In-memory Reliable, Scalable and Durable
- QA

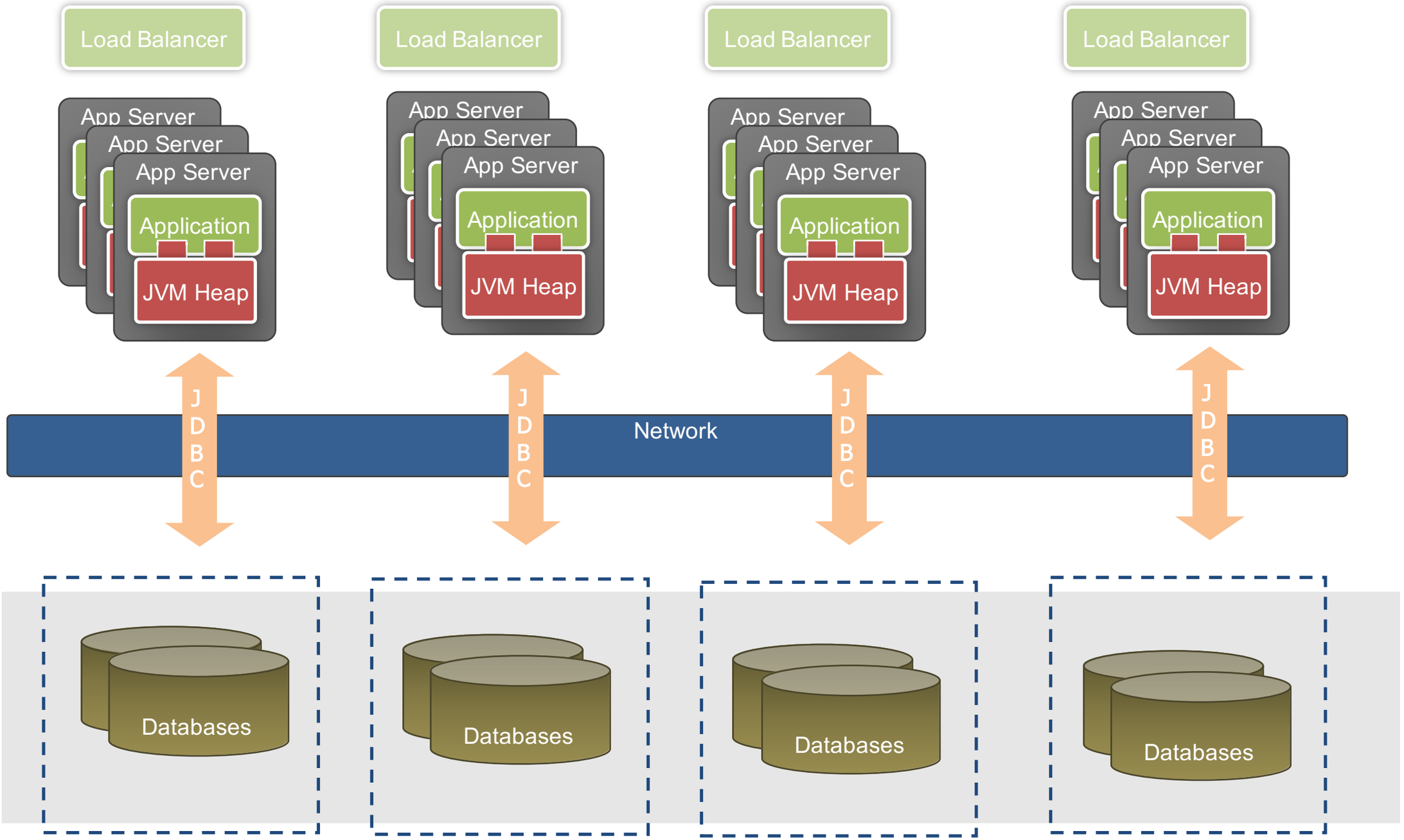
Why
In-Memory Data?

Problem

Anatomy of an Application: Tiered Data Storage



Common Enterprise Application Landscape



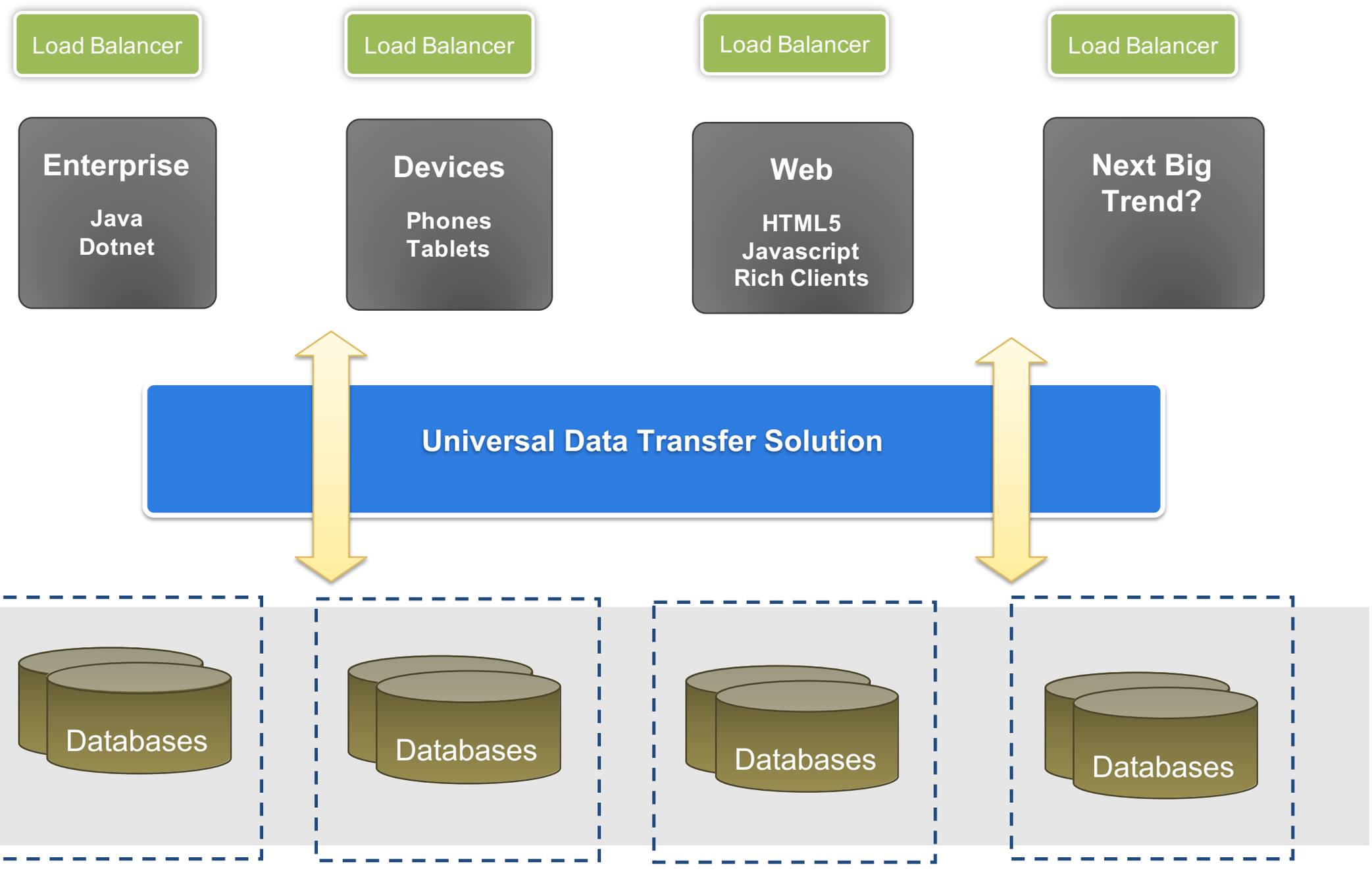
Performance & Scalability Challenges

- **JVM memory constraints**
  - Heap size limitations
  - GC pauses and tuning
  - Handling peak loads
  - Swapping
- **CPU limitations**
  - Limited processor power
  - Many applications sharing CPU cycles
  - CPU intensive processes

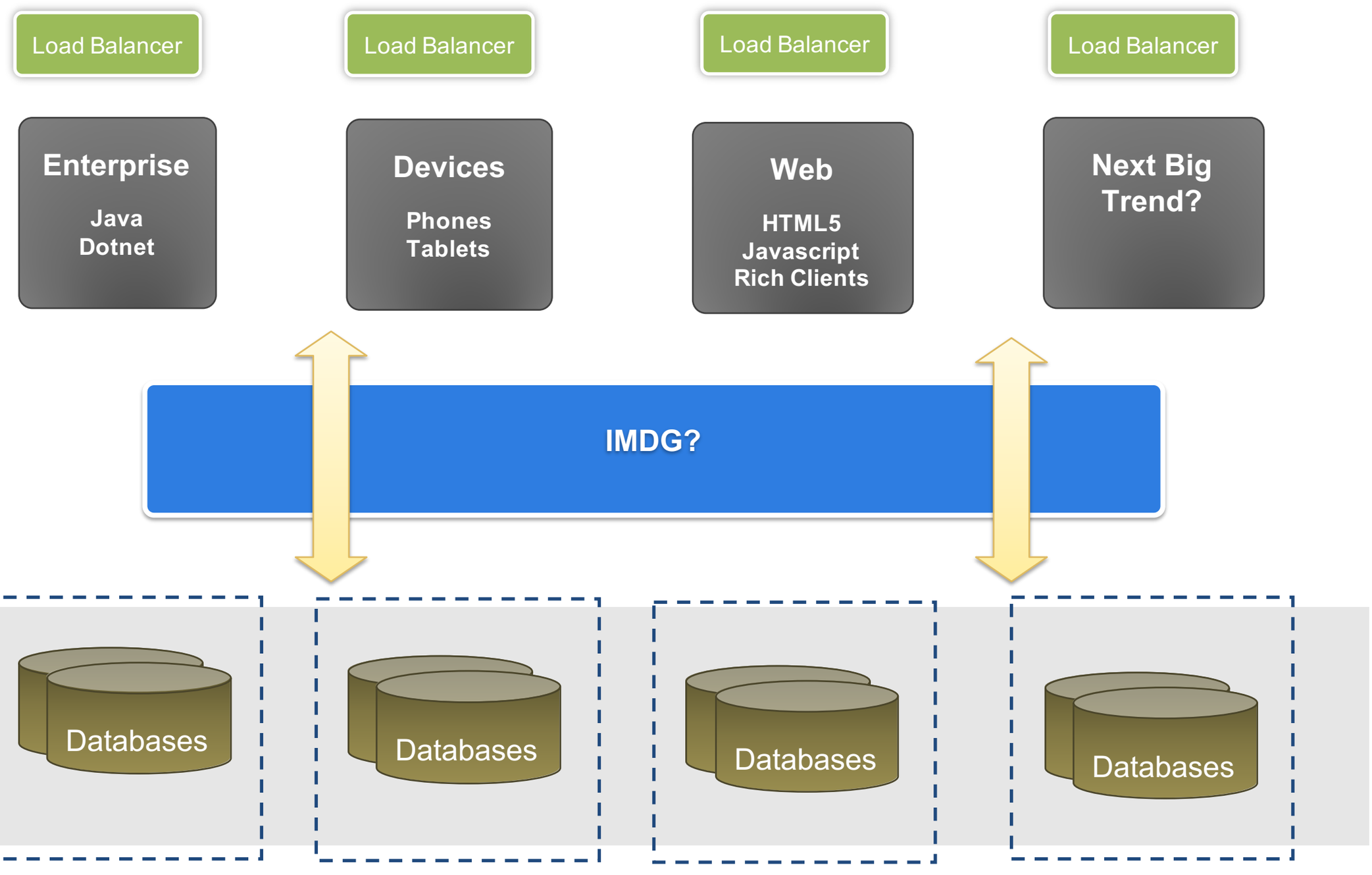
Performance & Scalability Challenges

- **Reliance on the database**
  - All transactions go through the database
  - Expensive and complex to scale
  - Reliance on DBA(s) to maintain and tune
- **Network I/O**
  - Bandwidth limitations
  - Latency
- **State-aware cluster**
  - Session replication

Even More Common Enterprise Application Landscape



Even More Common Enterprise Application Landscape



In-Memory Data Lanscape

Messaging Middleware

NoSQL DATABASES

IMDG

SQL DATABASES

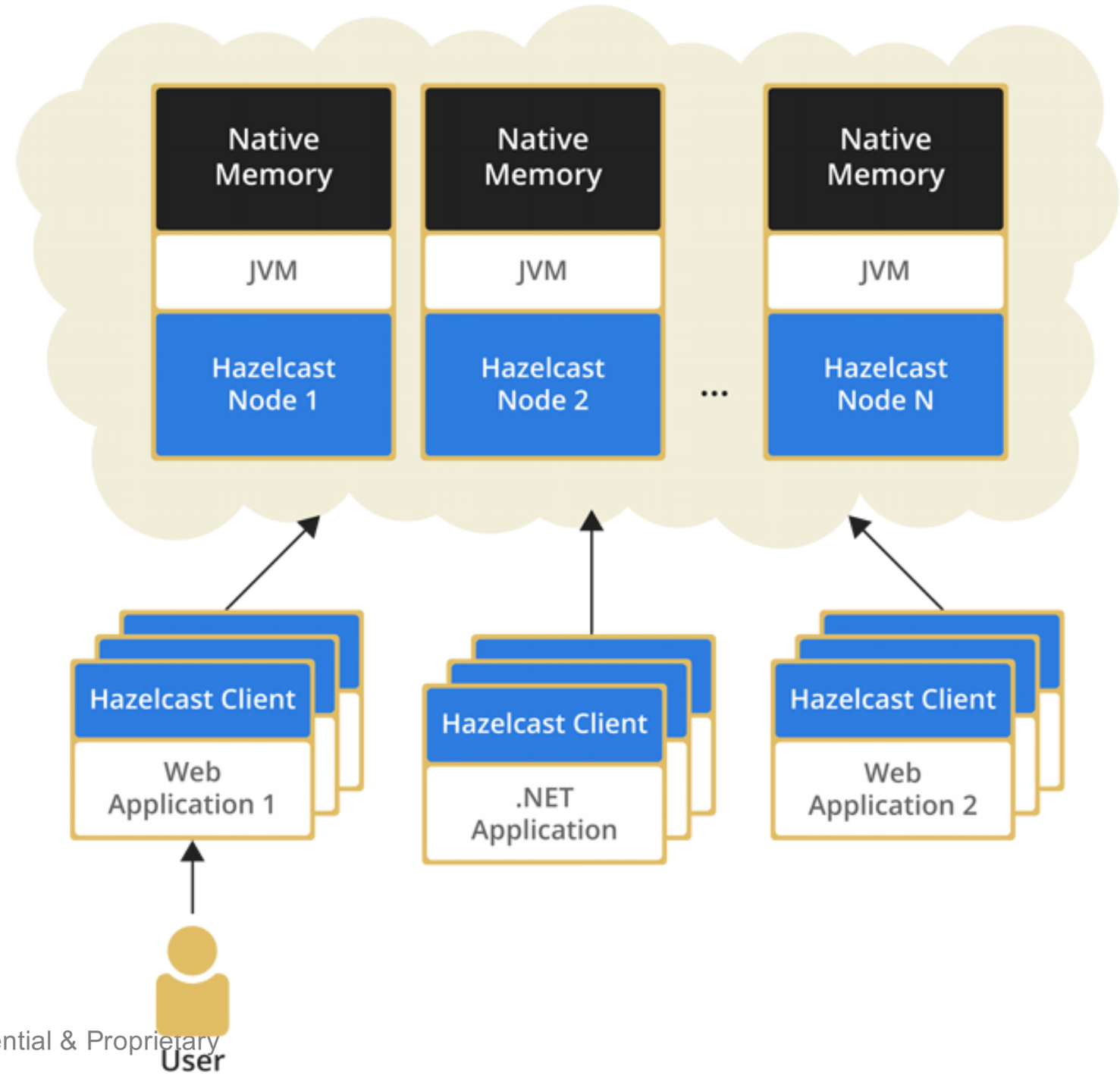
In-Memory Computing

Distributed Data Applications

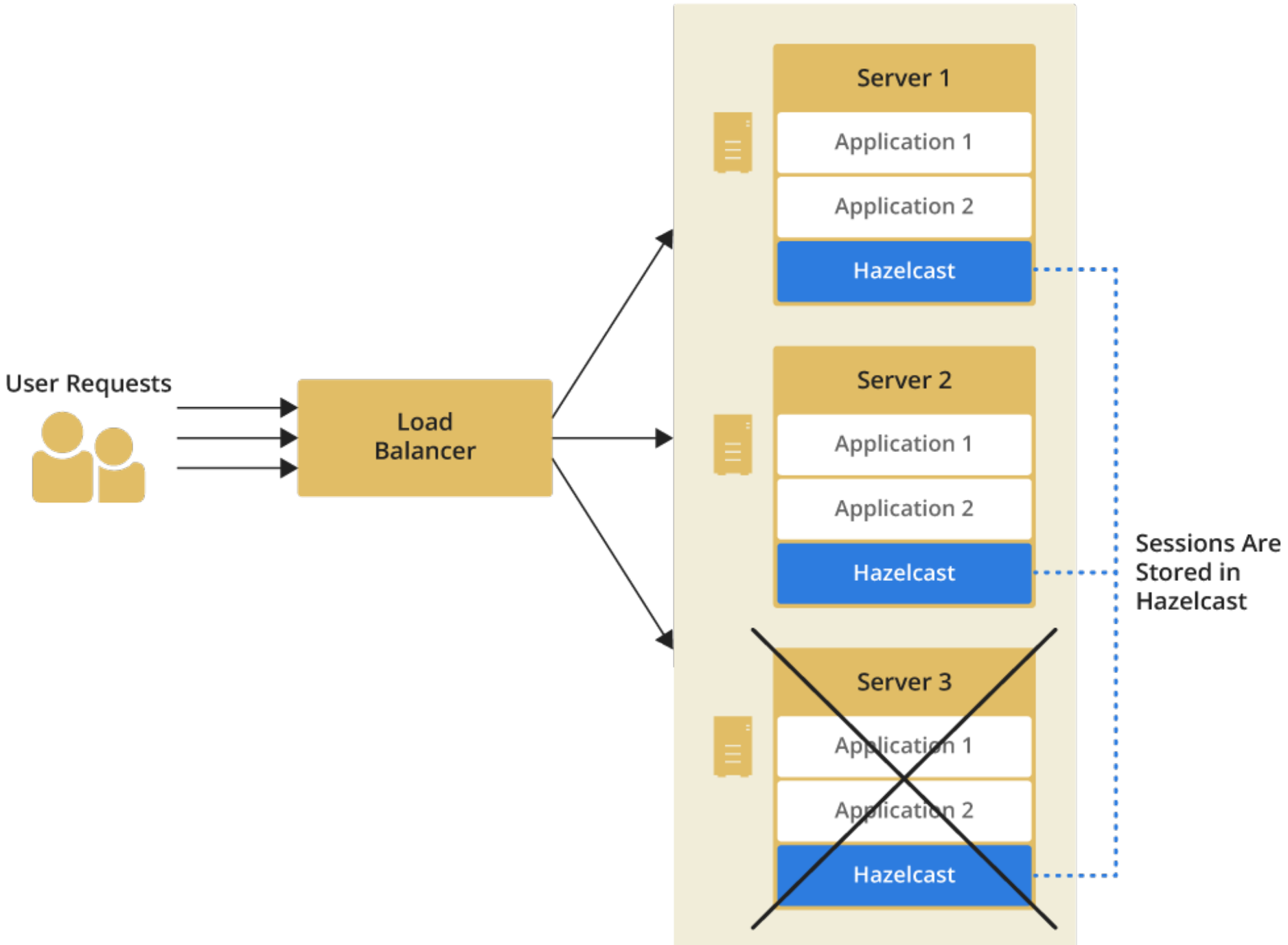
Distributed Data Applications

- Cache-as-a-Service
- Application Scaling
- Database Caching
- Distributed Computing
- Reactive / Smart Clients

Cache-as-a-Service



Durability of application data

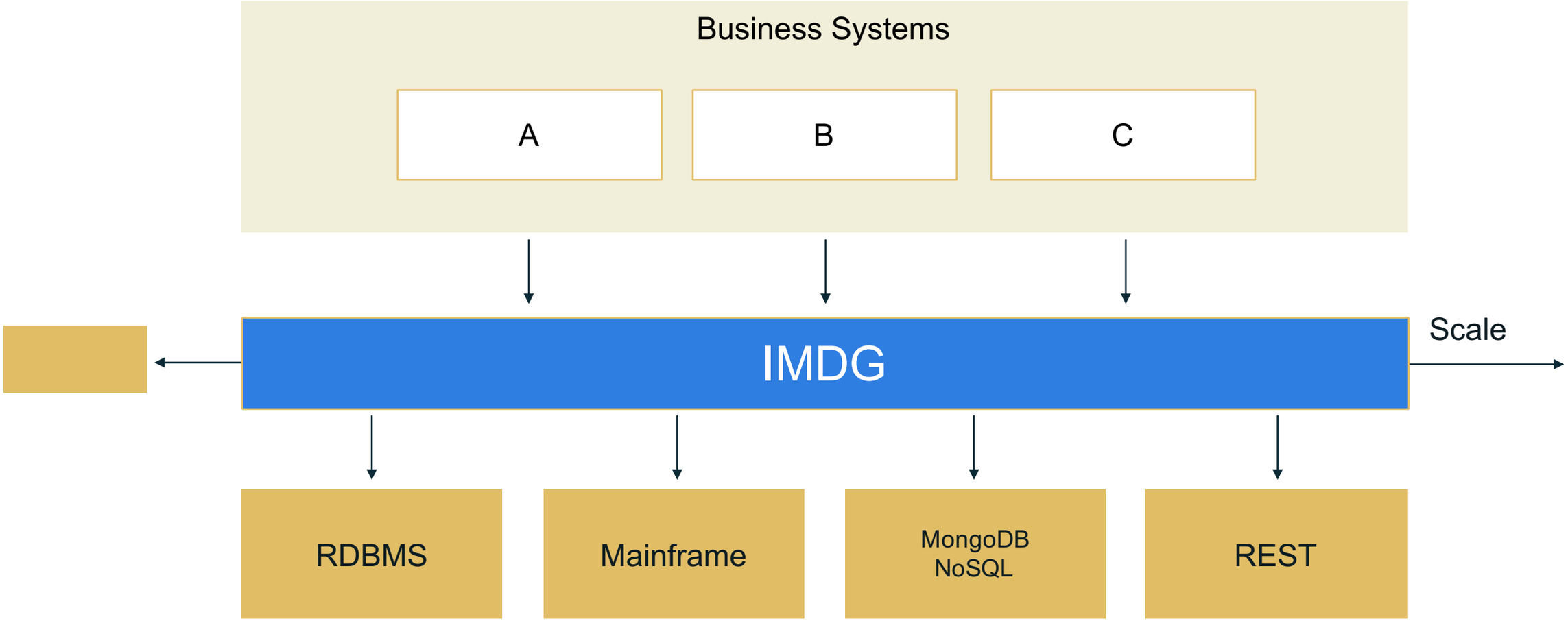


When One Server Goes Down Sessions Are Not Lost

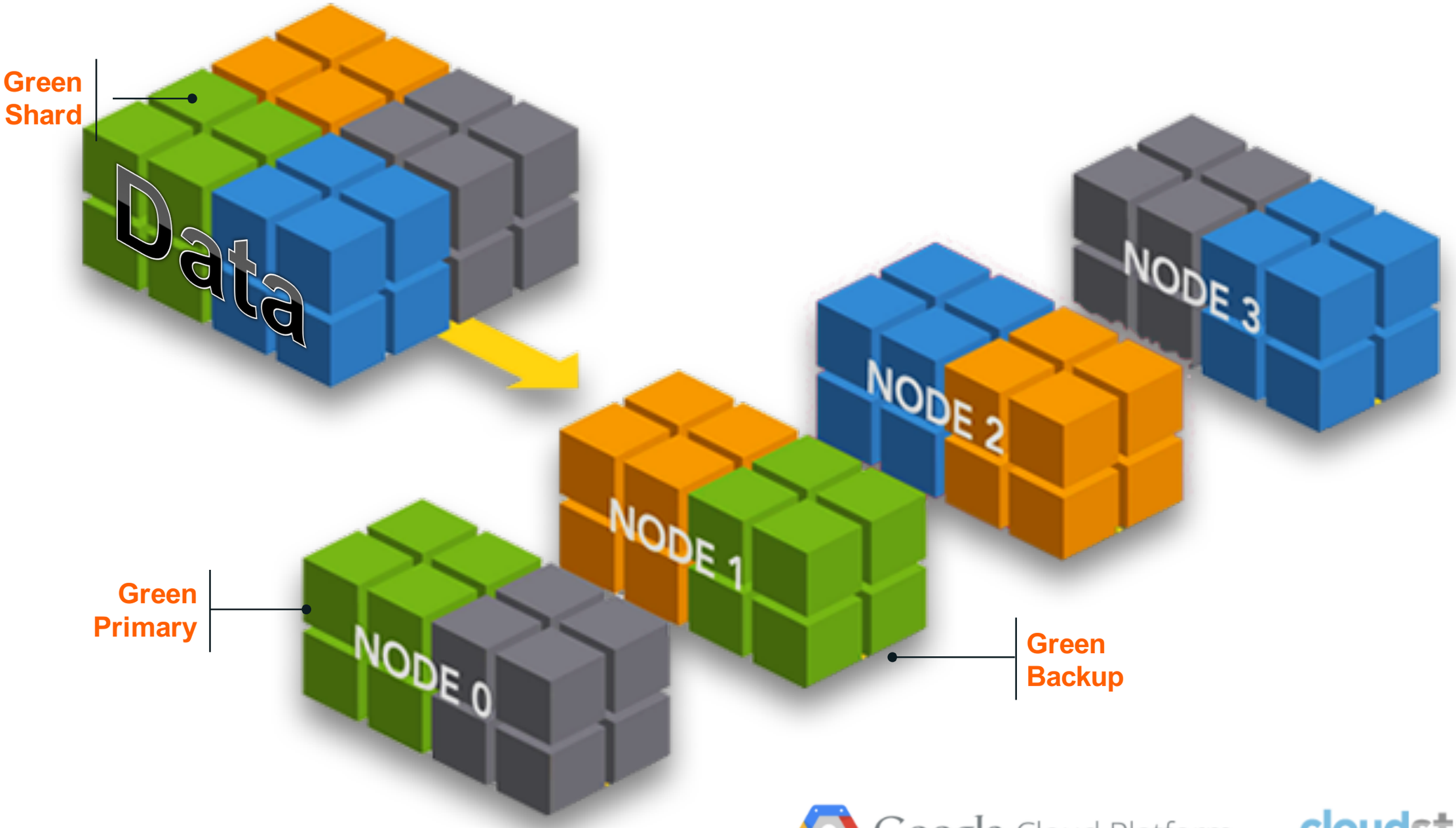
Application Scaling

- Elastic Scalability
- Super Speeds
- High Availability
- Fault Tolerance
- Cloud Readiness
  - EC2, GCE, Docker deployment

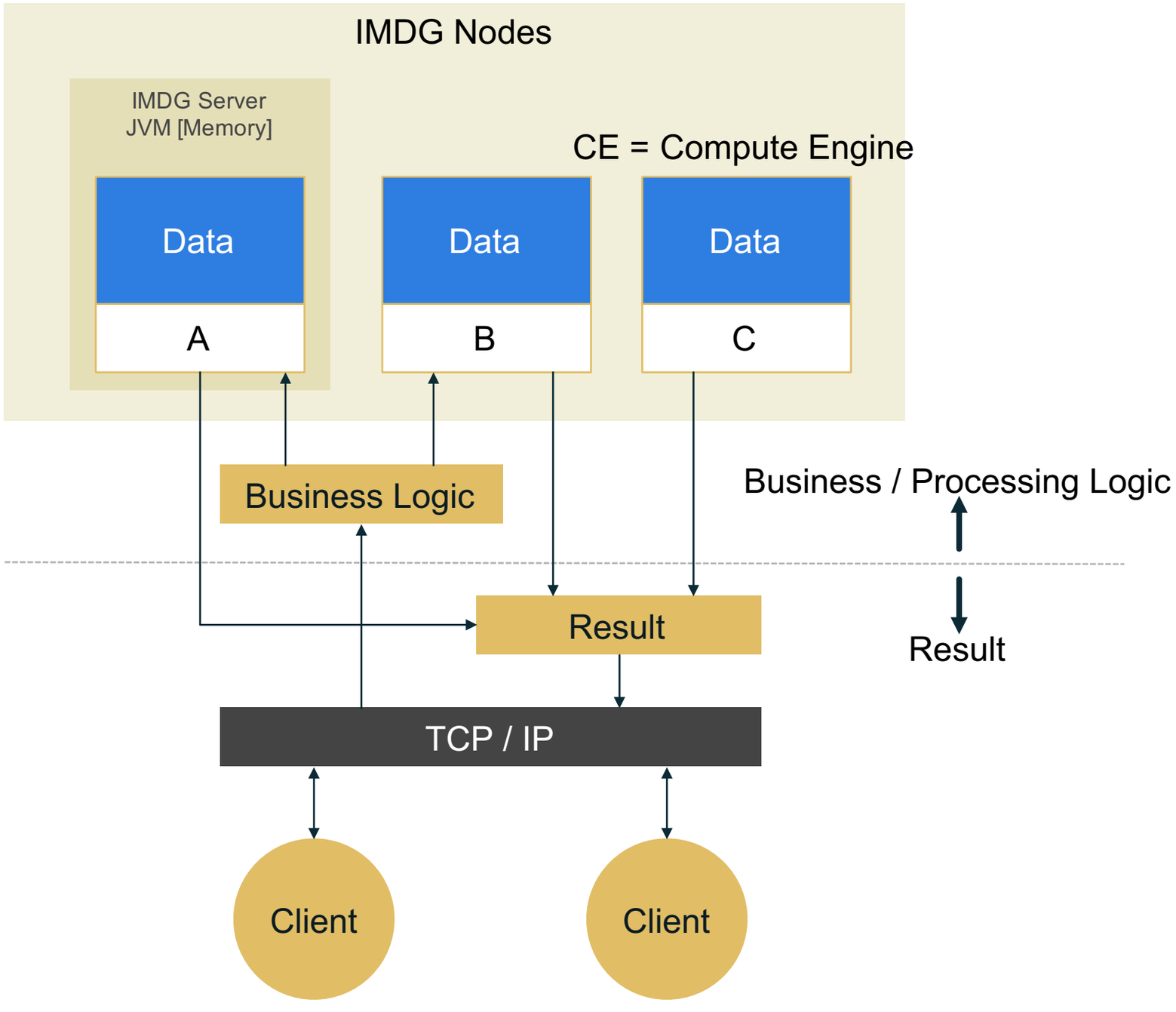
Unified Data Access / Database Caching



Elastic Scalability



Distributed Computing



Reactive Clients

The clients (part of an application) enable you to do the operations without being a member of the grid. Clients can be:

- smart: this means that they immediately can send an operation like *map.get(key)* to the member that owns that specific key.
- dumb: it will connect to a random member in the cluster and send requests to this member. This member then needs to send the request to the correct member.

High availability

When the connected cluster member dies, client will automatically switch to another live member.

Making In-Memory Reliable, Scalable and Durable

Data Distribution

http://www.ministryoftofu.com/2012/01/photos-upcoming-chinese-new-year-creates-backlogs-for-courier-companies/

Data Distribution

Replication - Copying an entire dataset onto multiple servers. Used for improving speed of access to reference records such as master data.

Partitioning - Splitting up a large monolithic dataset into multiple smaller sets based on data cohesion.

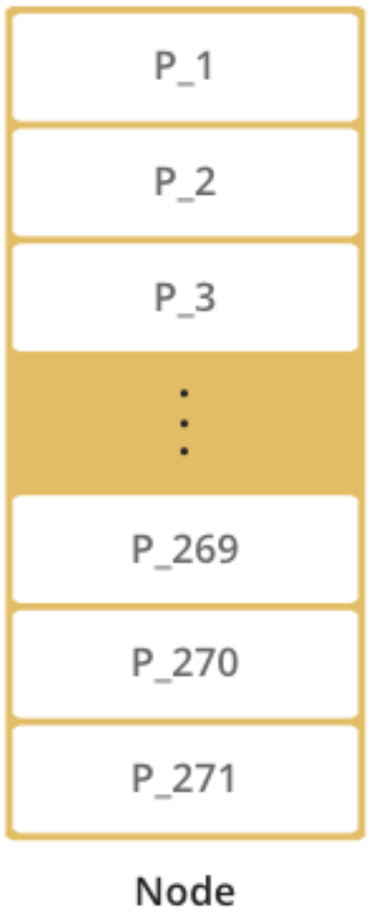
You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

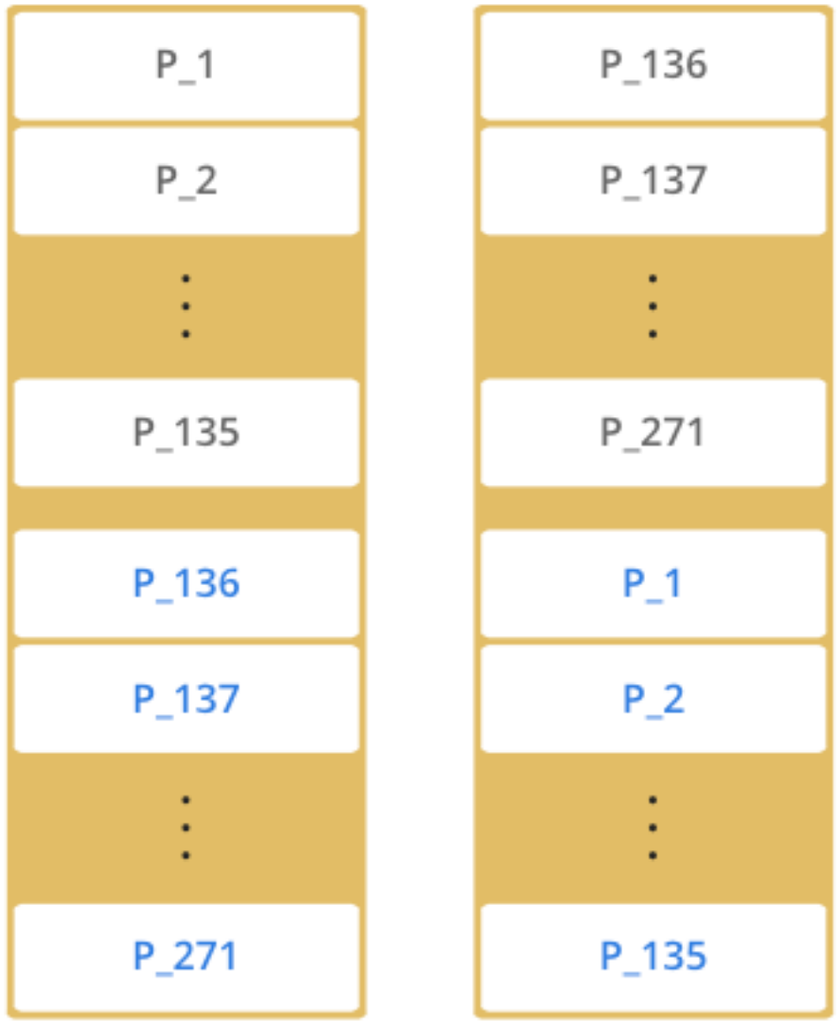
Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

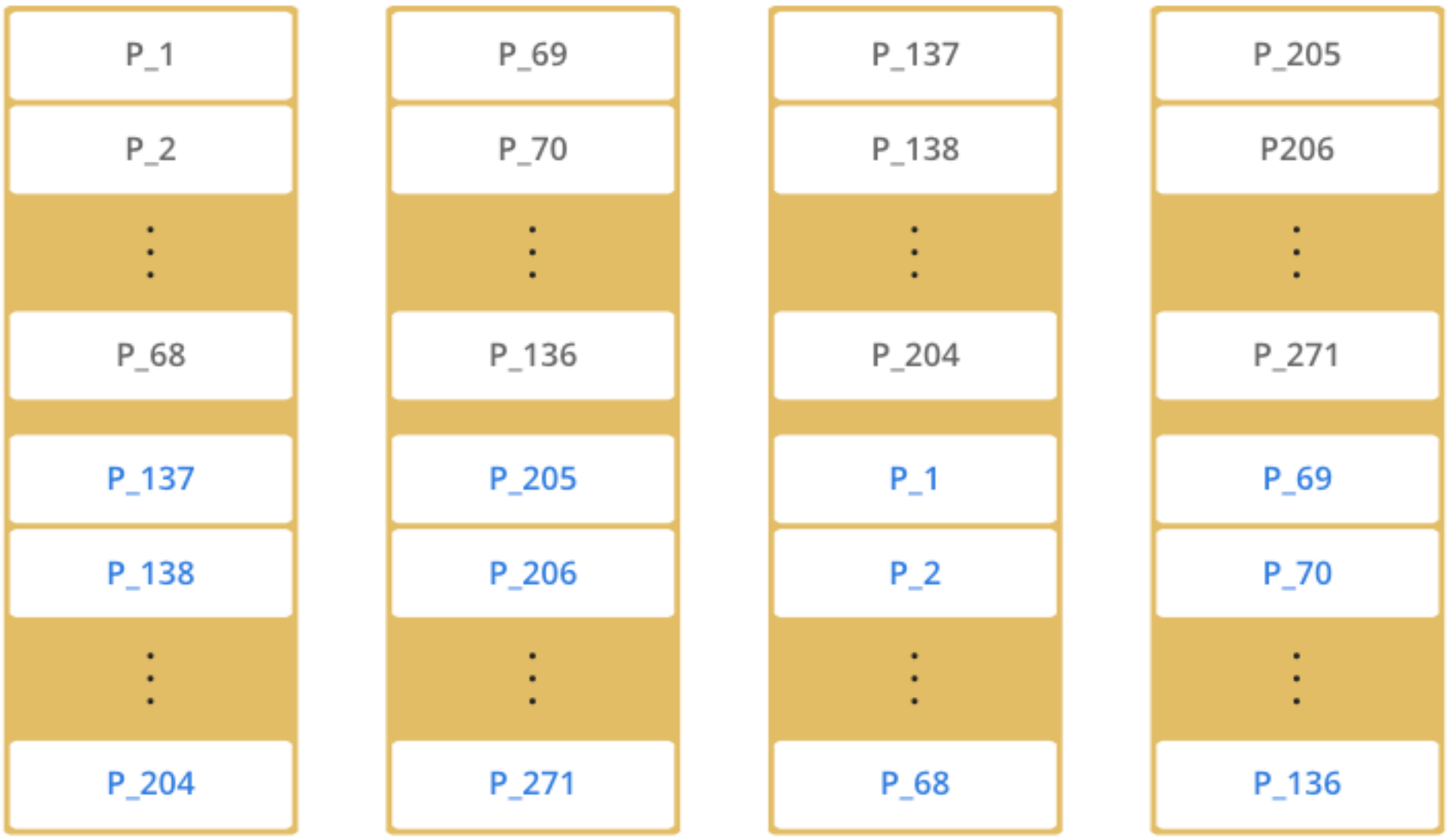
Data Partitioning



Data Partitioning

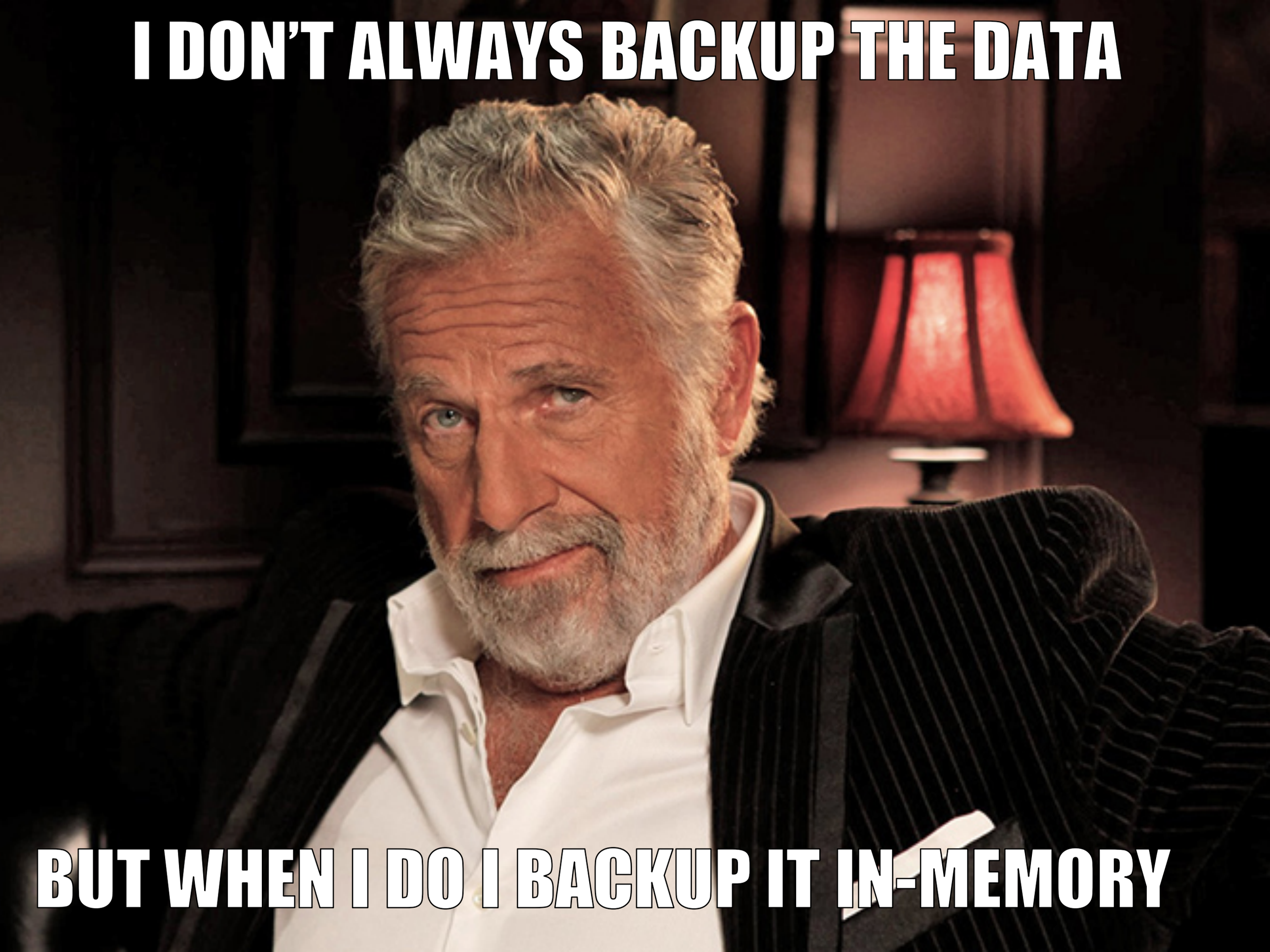


Data Partitioning



Rebalance Data On New Node

I DON'T ALWAYS BACKUP THE DATA



BUT WHEN I DO I BACKUP IT IN-MEMORY

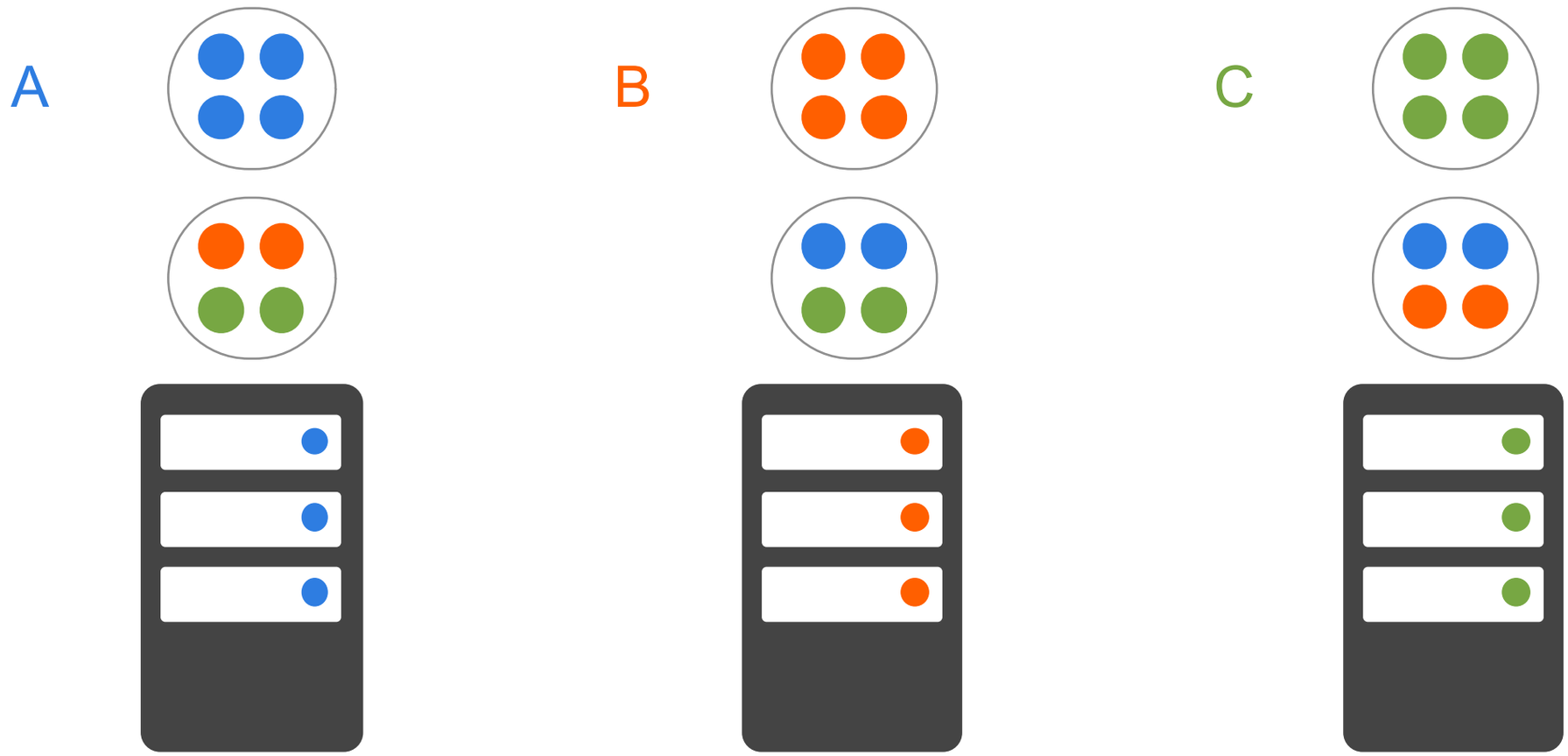
Consistent Hashing

Fixed number of partitions (default 271)

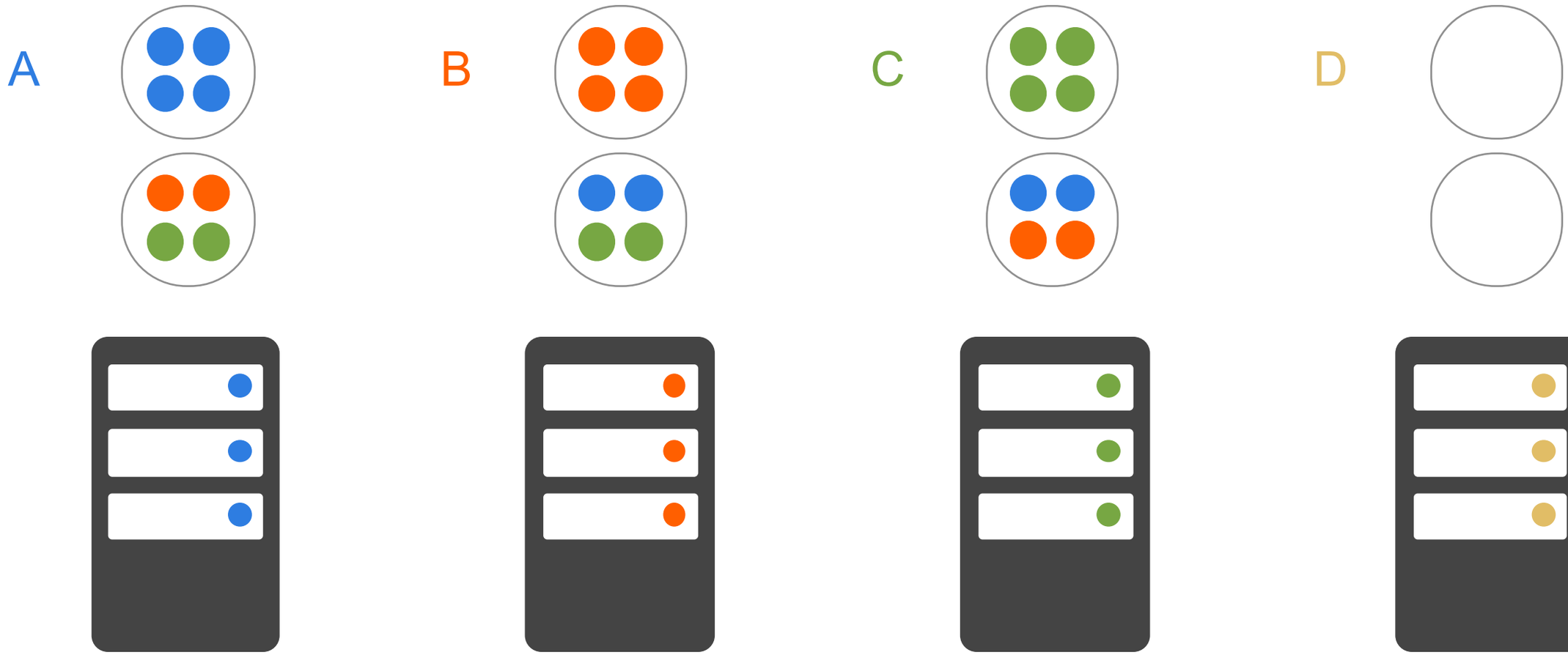
Each key falls into a partition

```
partitionId = hash(keyData)%PARTITION_COUNT
```

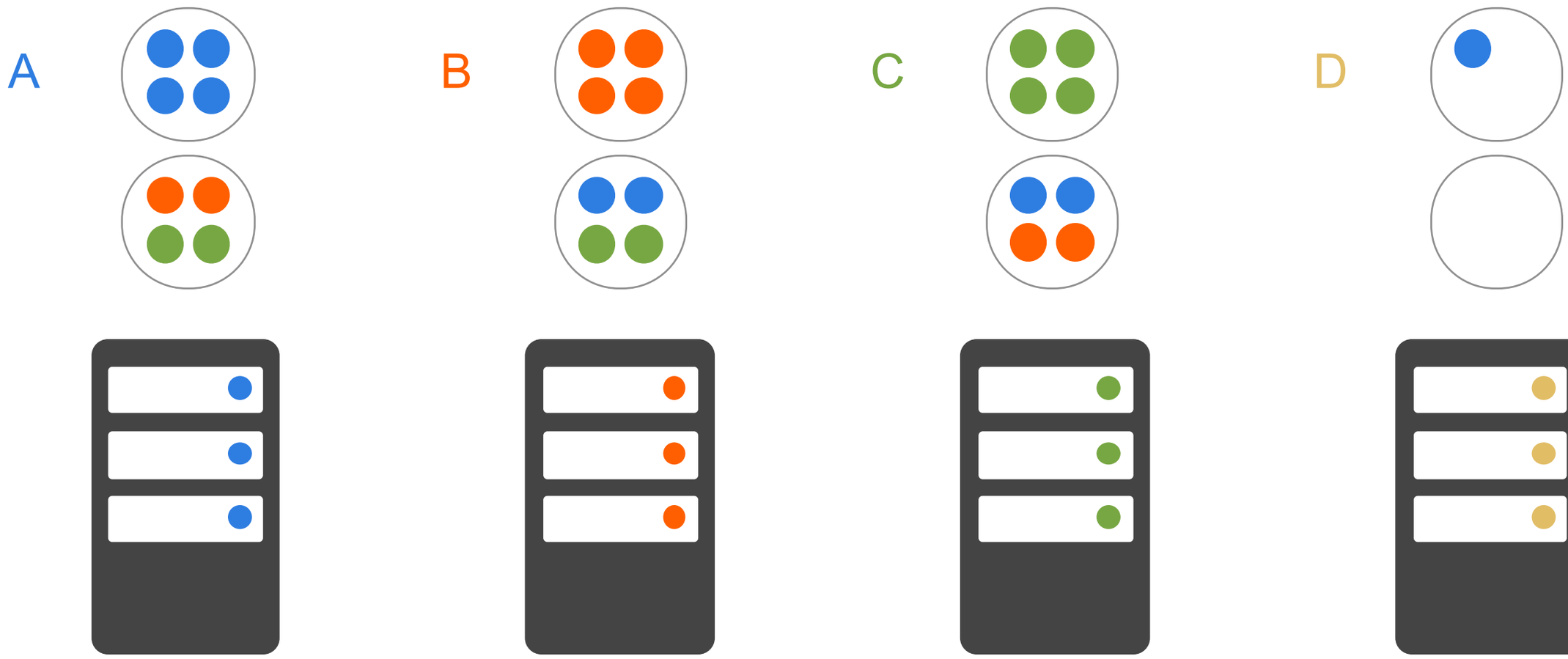
Partition ownerships are reassigned upon membership change



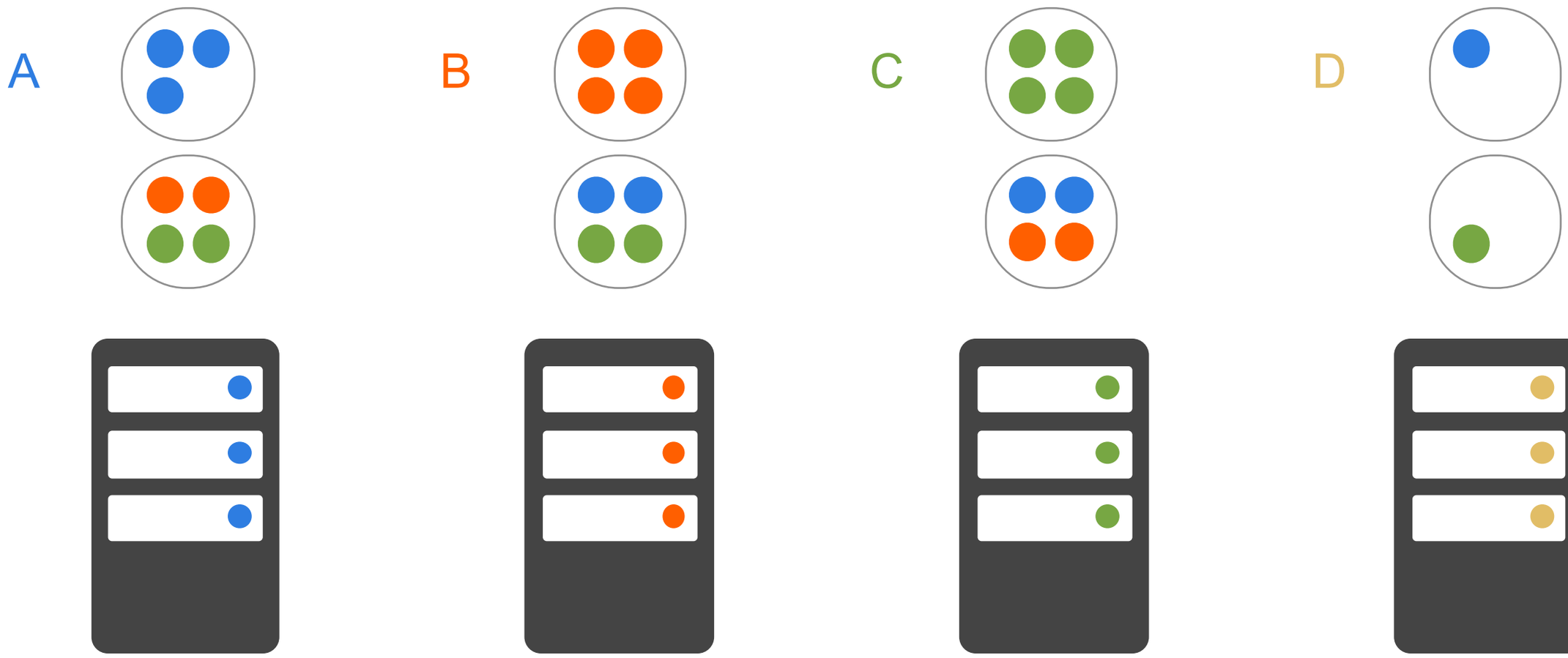
New Node Added



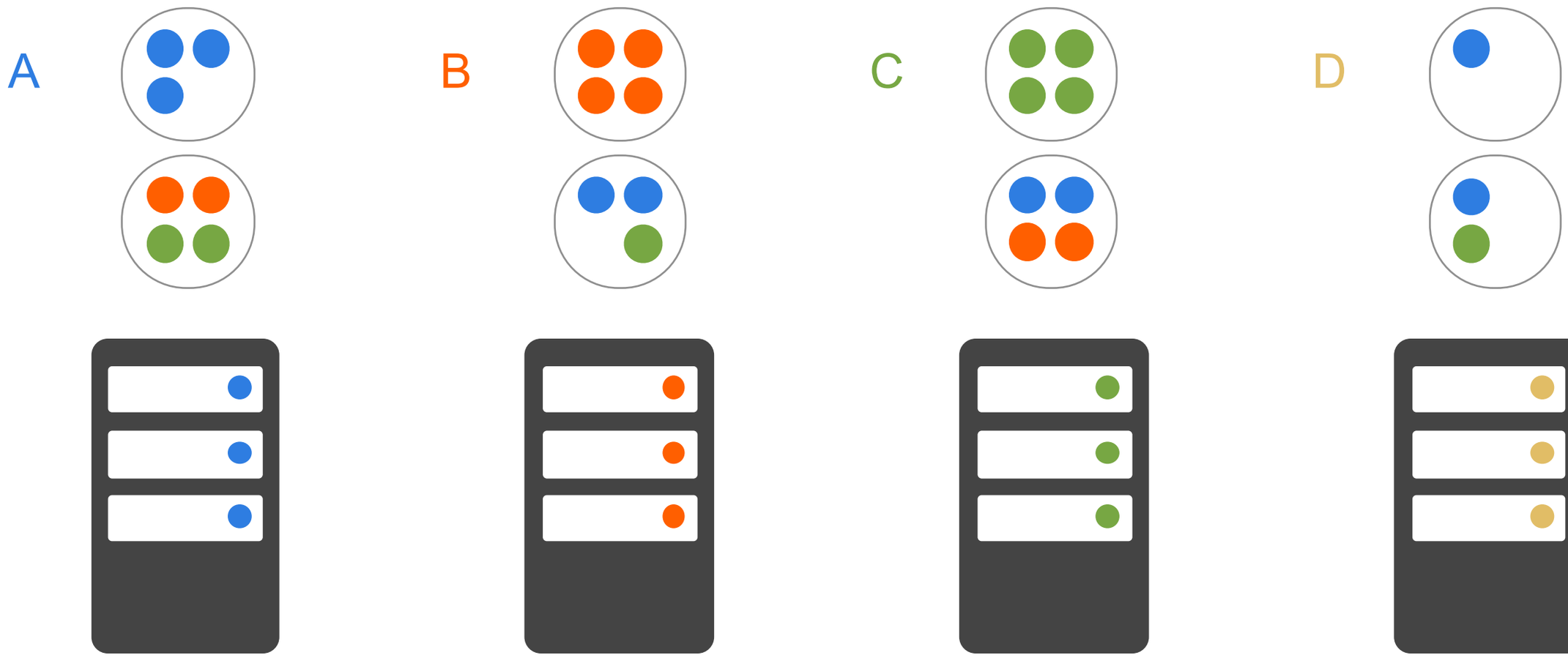
Migration



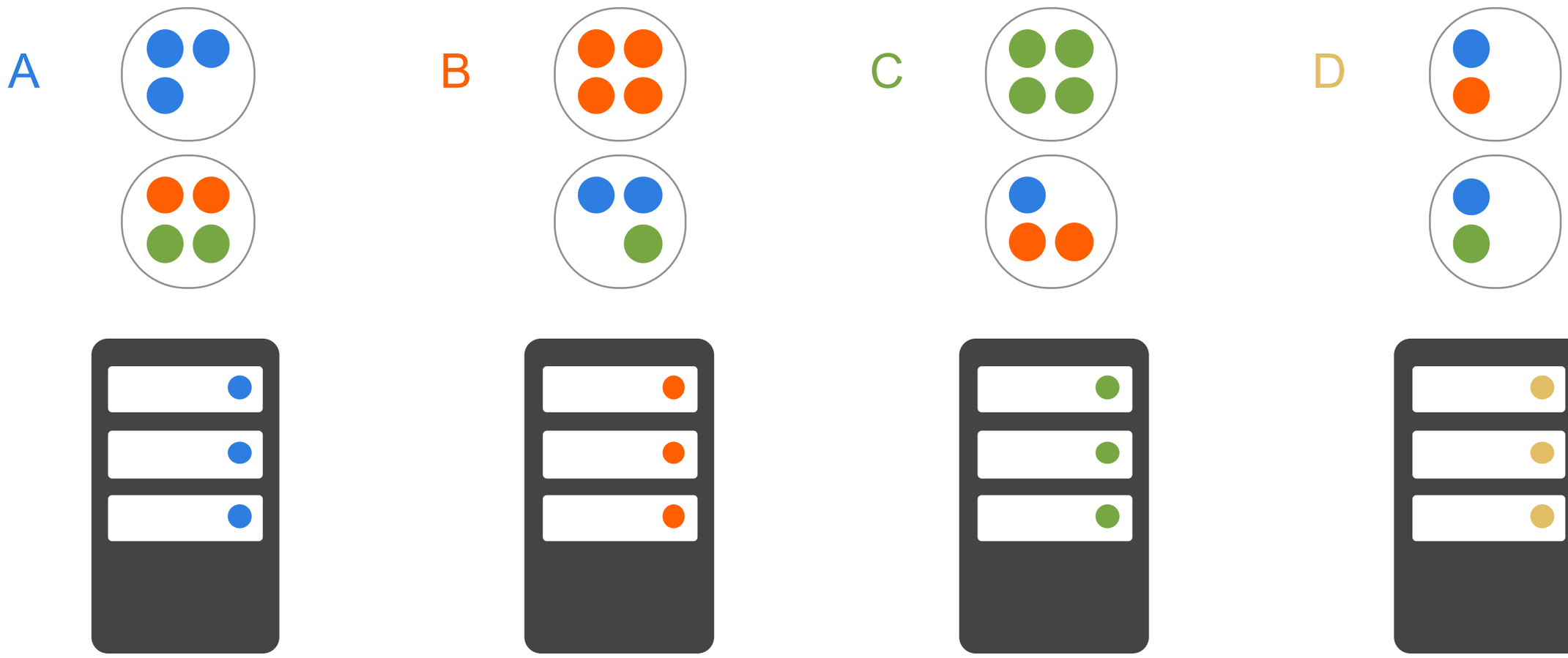
Migration



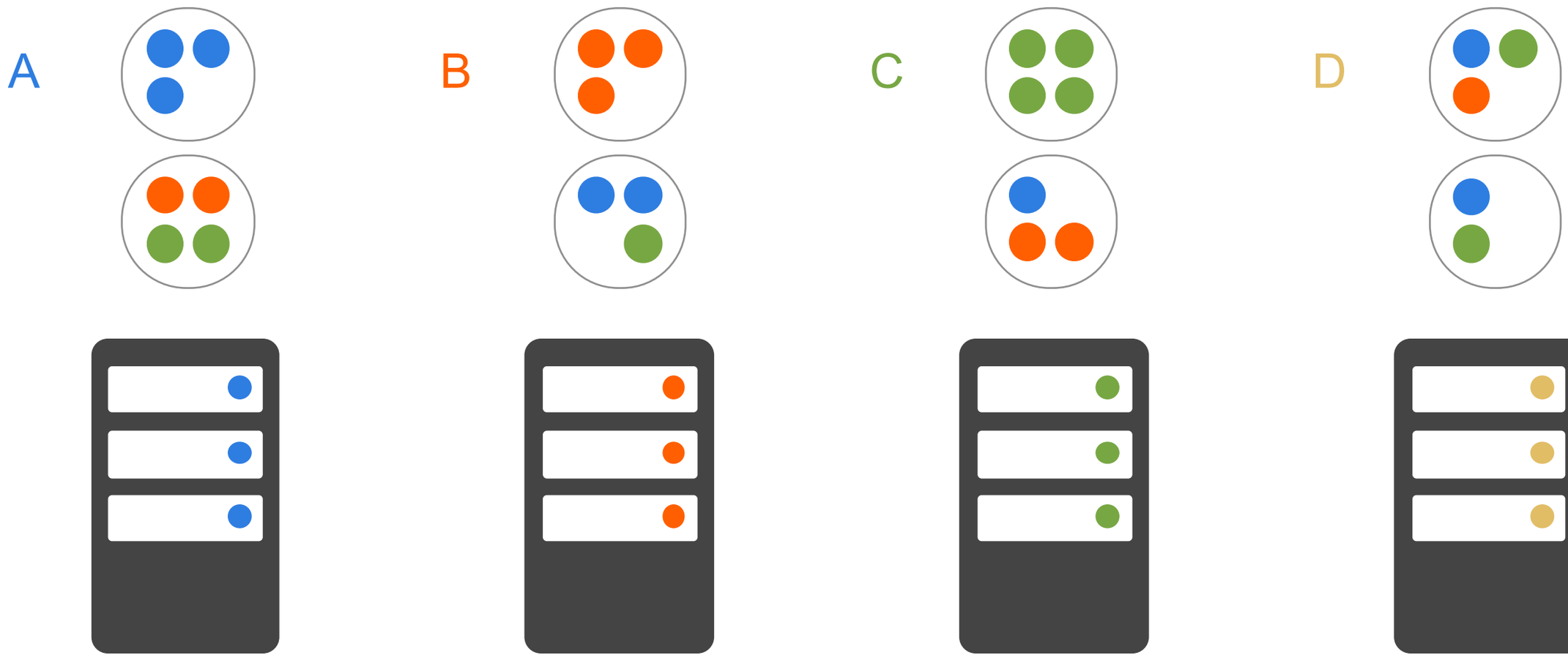
Migration



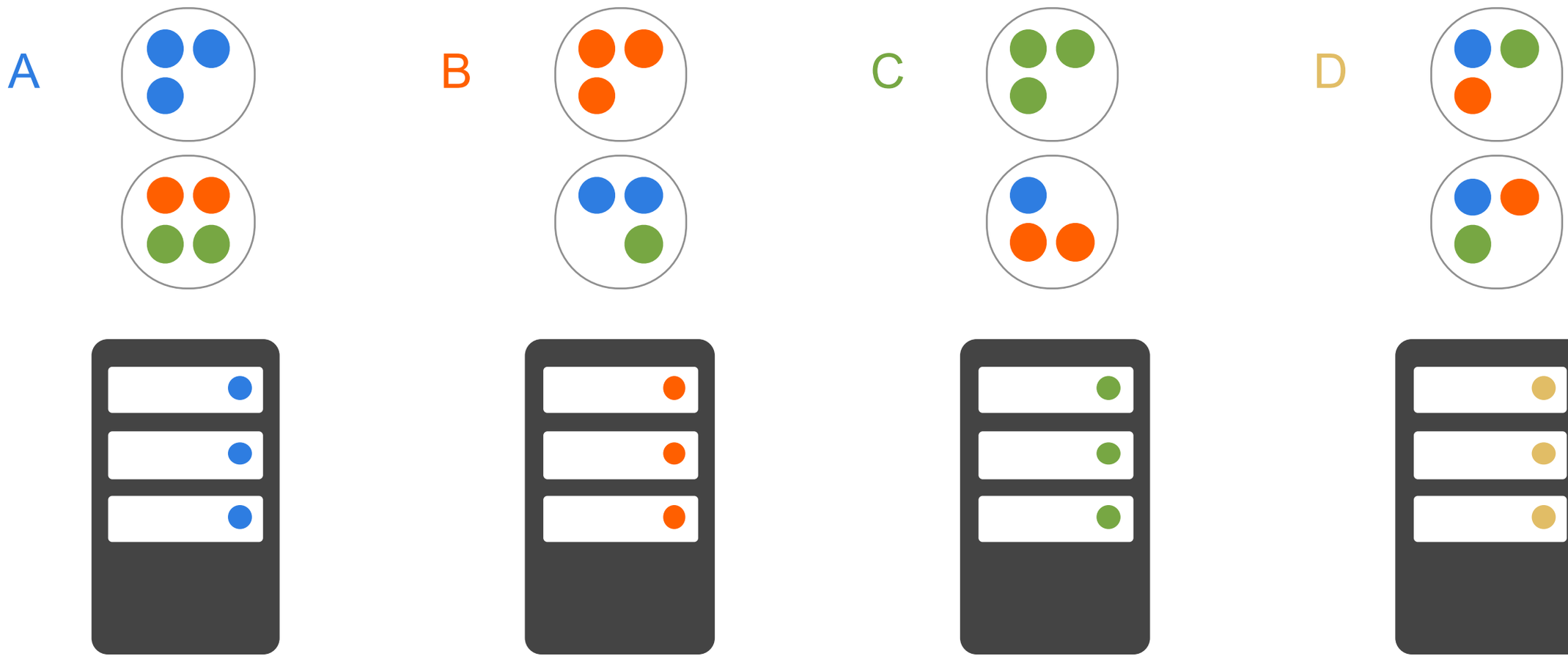
Migration



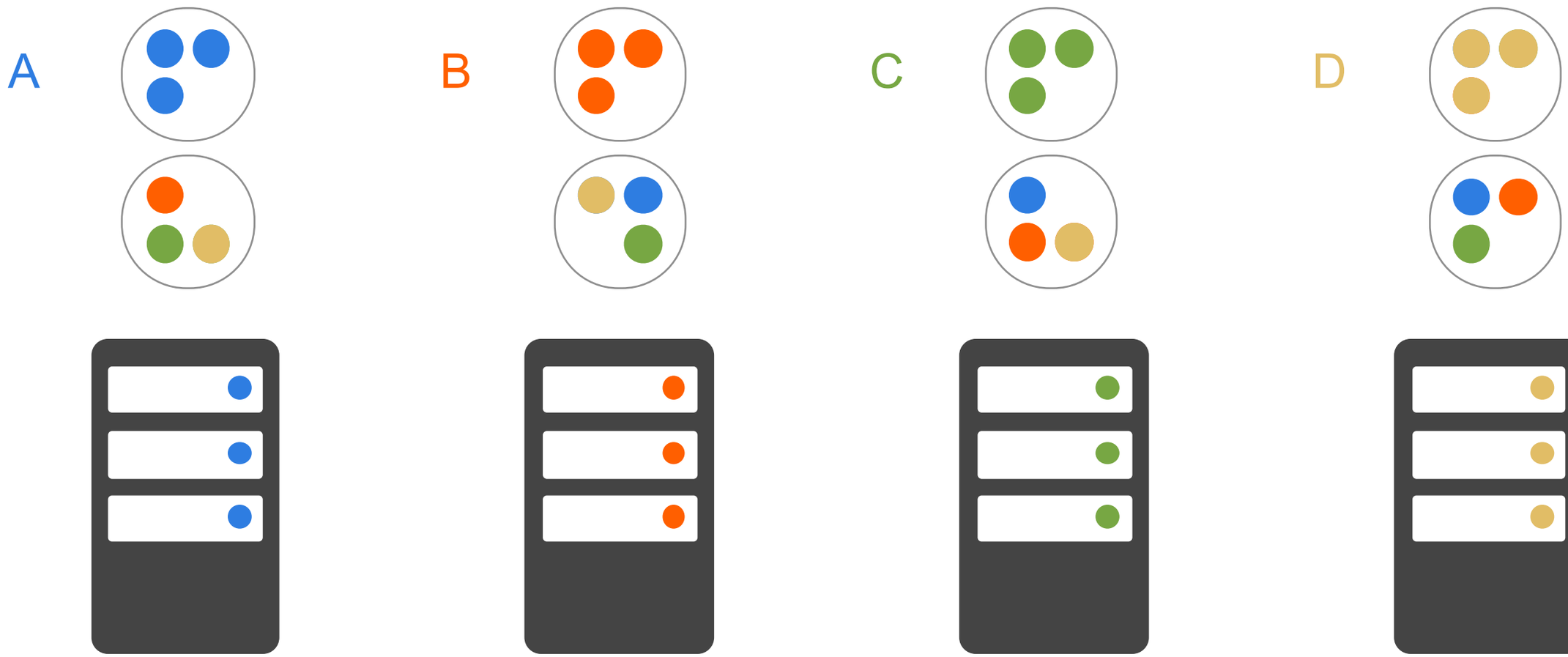
Migration



Migration

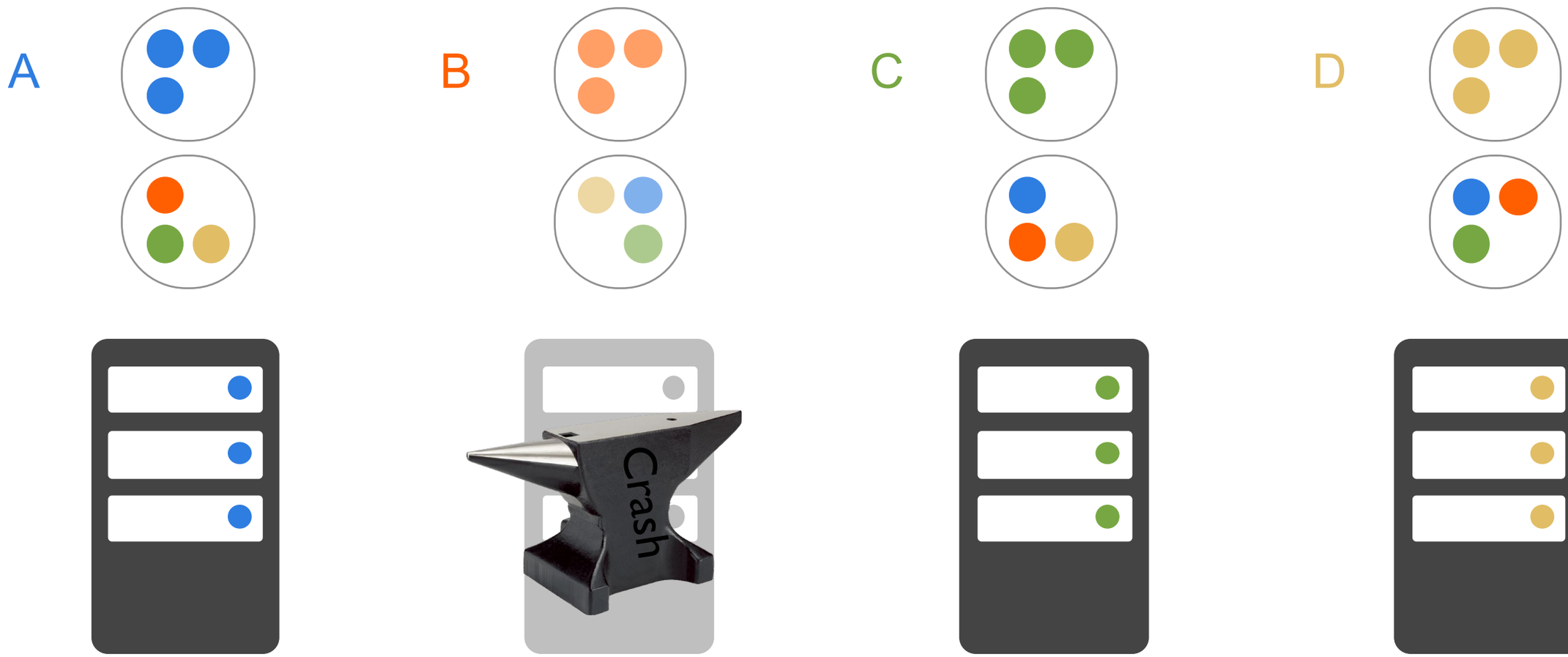


Migration Complete

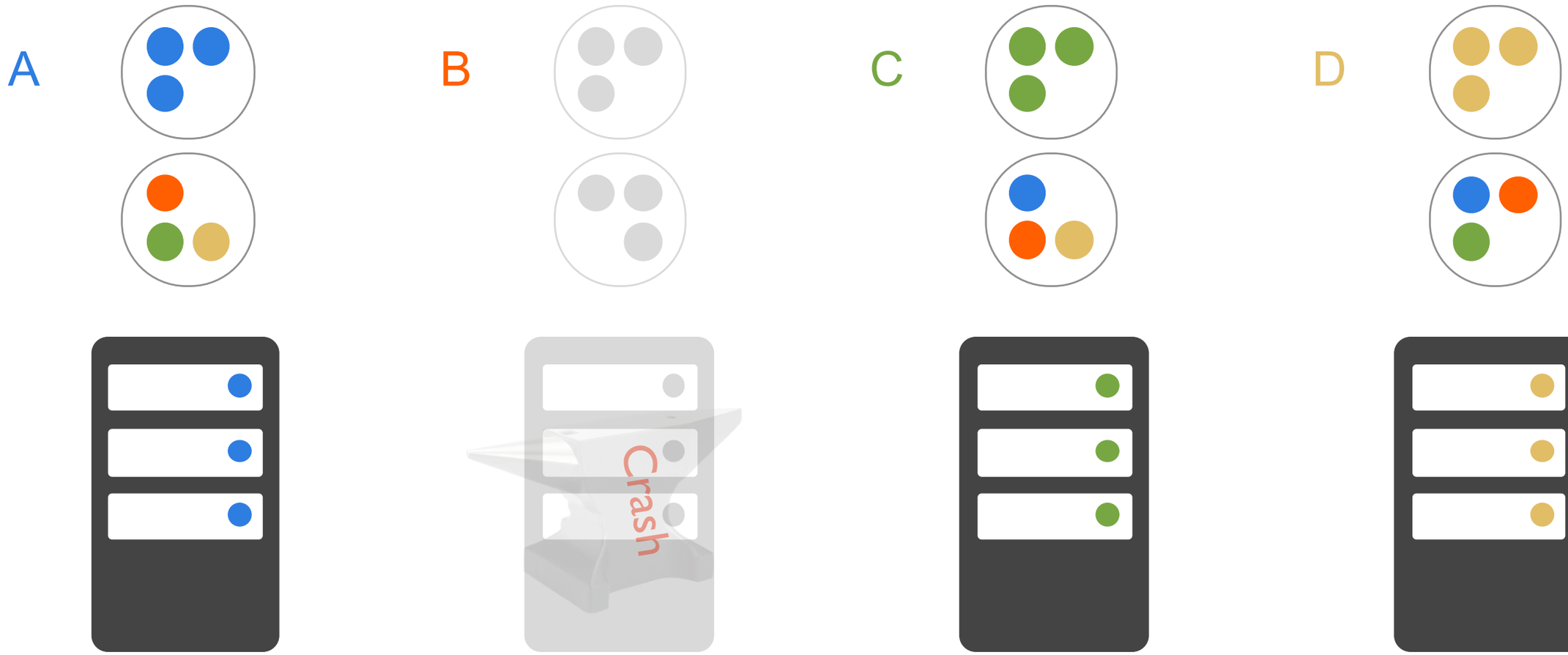


Data Safety when Node Dies

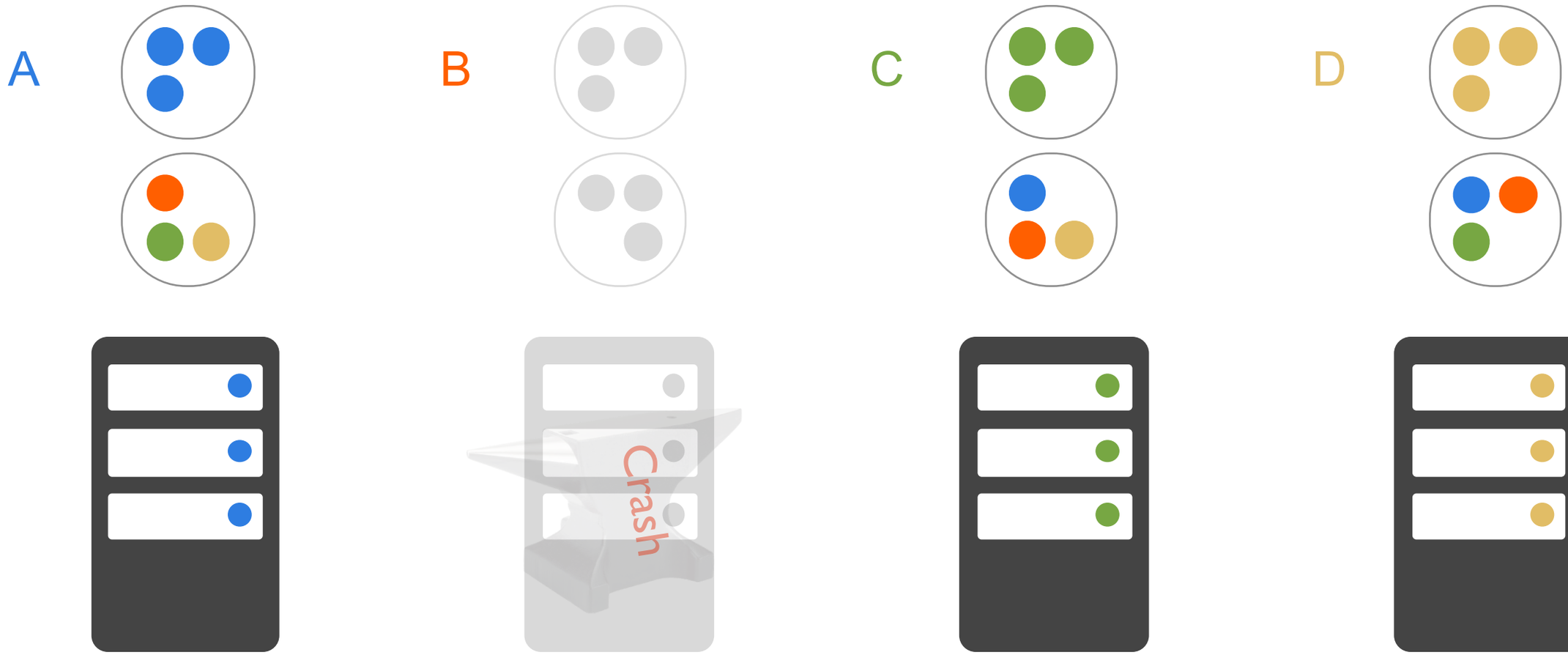
Node Crashes



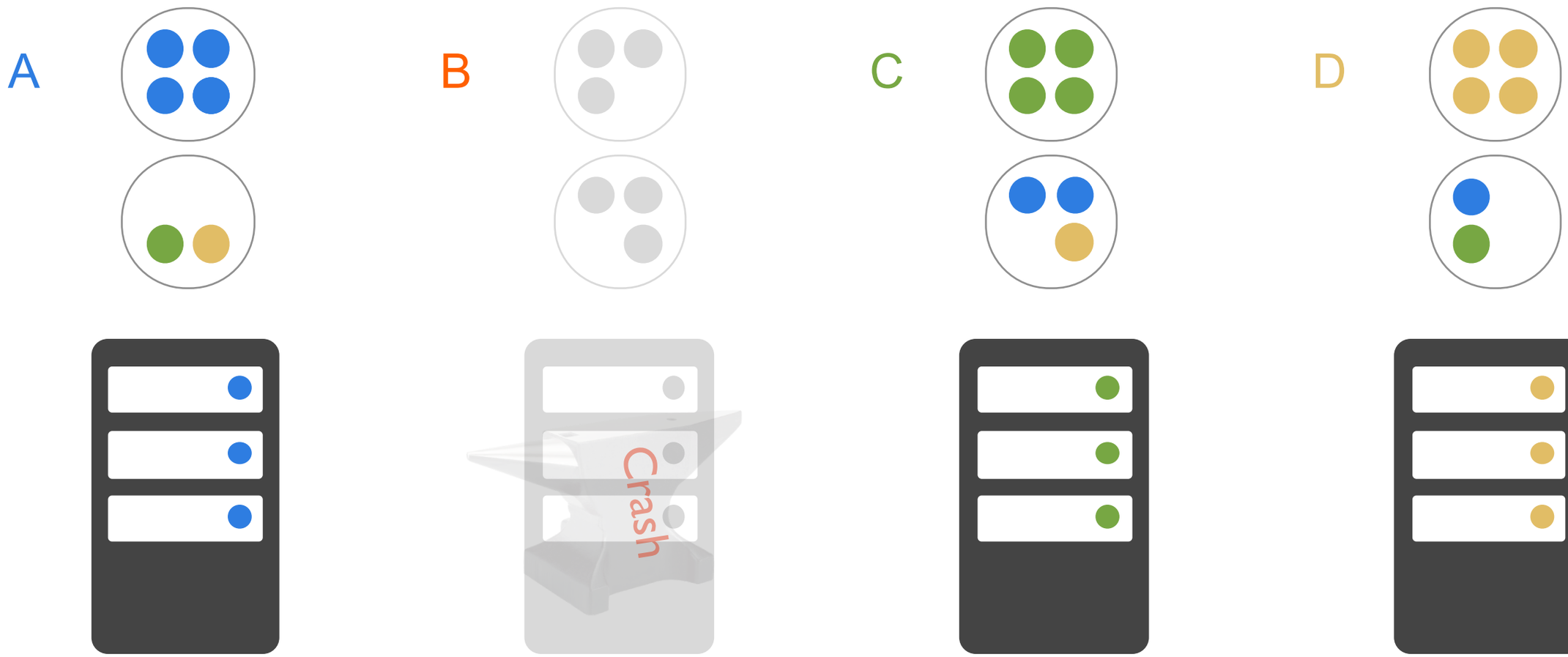
Backups Are Restored



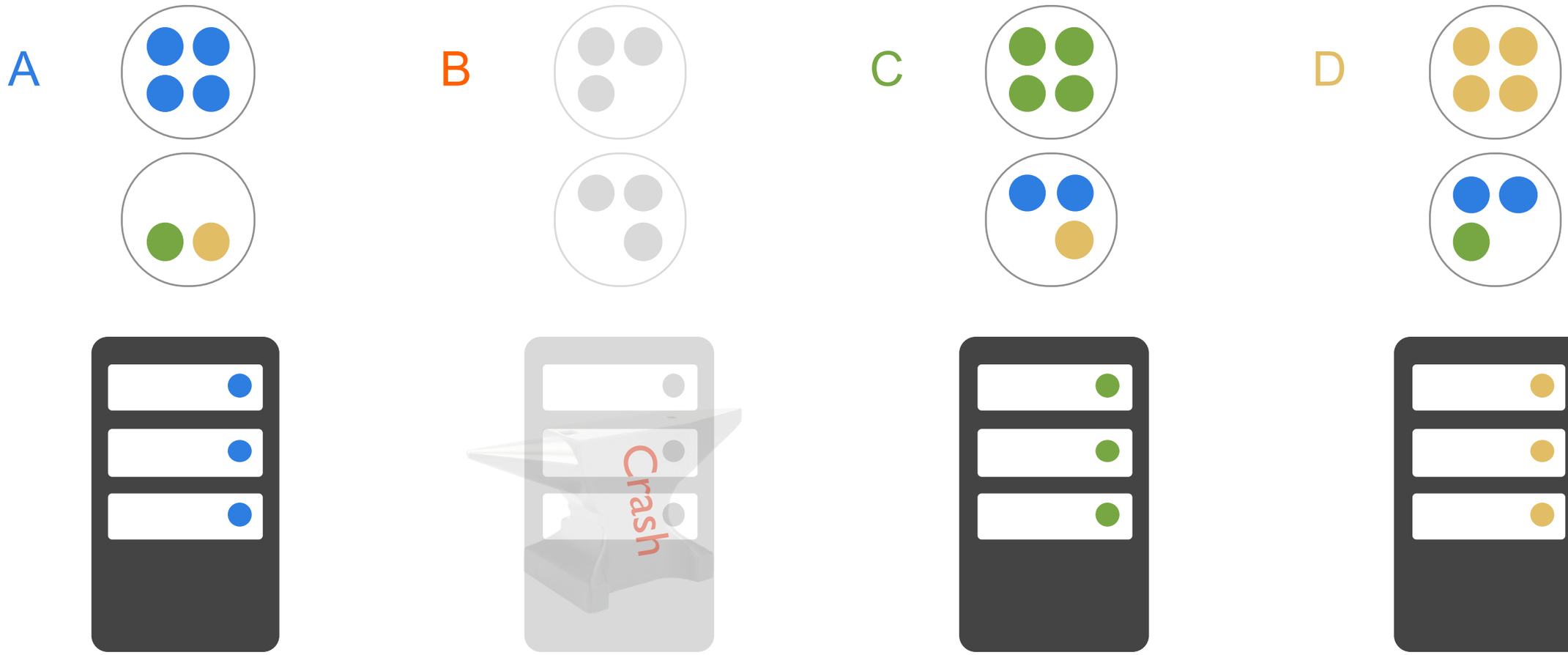
Backups Are Restored



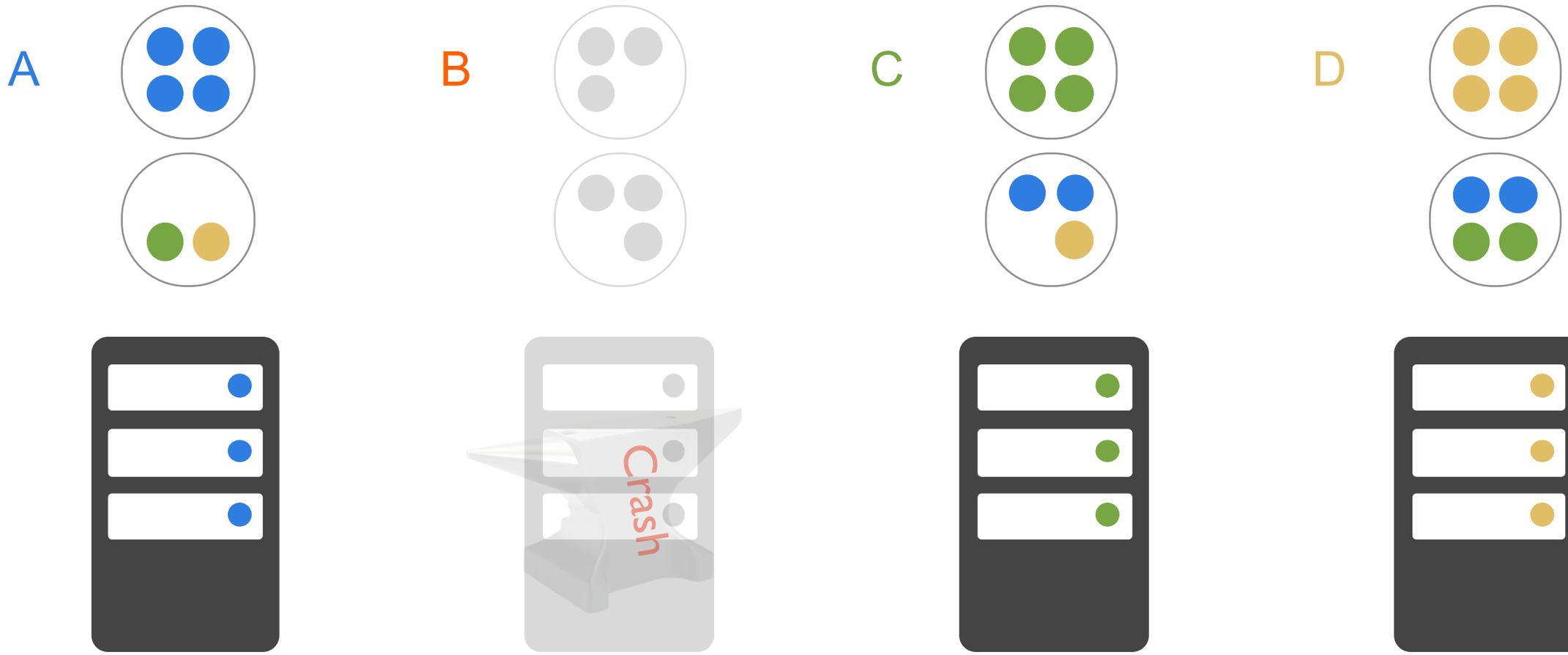
Backups Are Restored



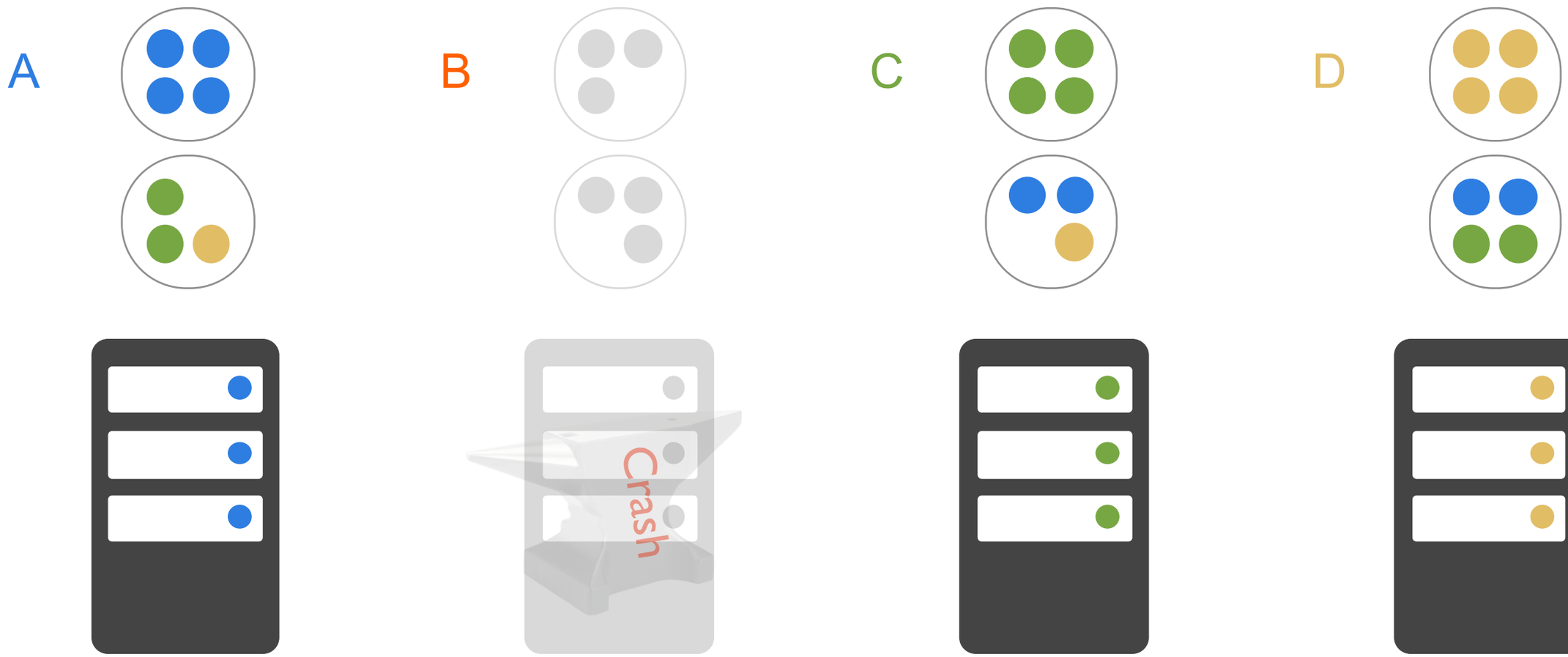
Backups Are Restored



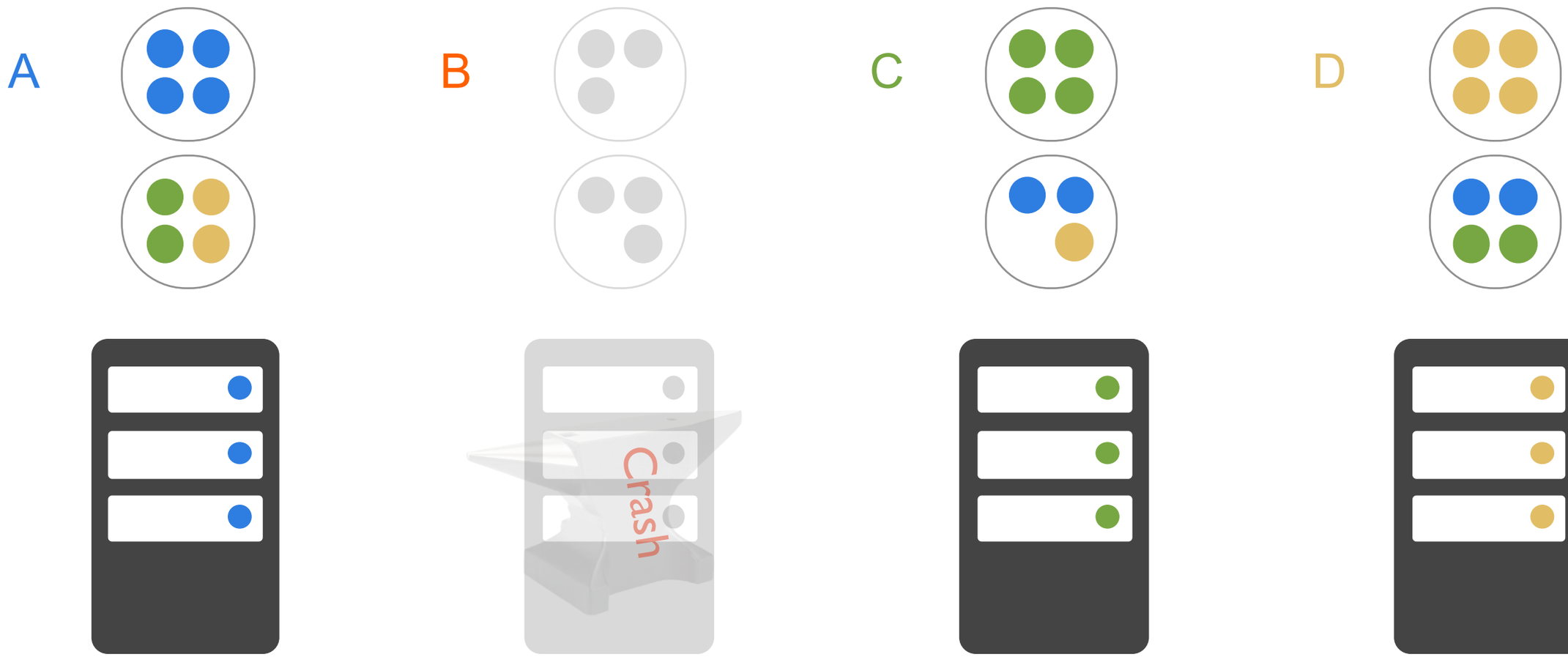
Backups Are Restored



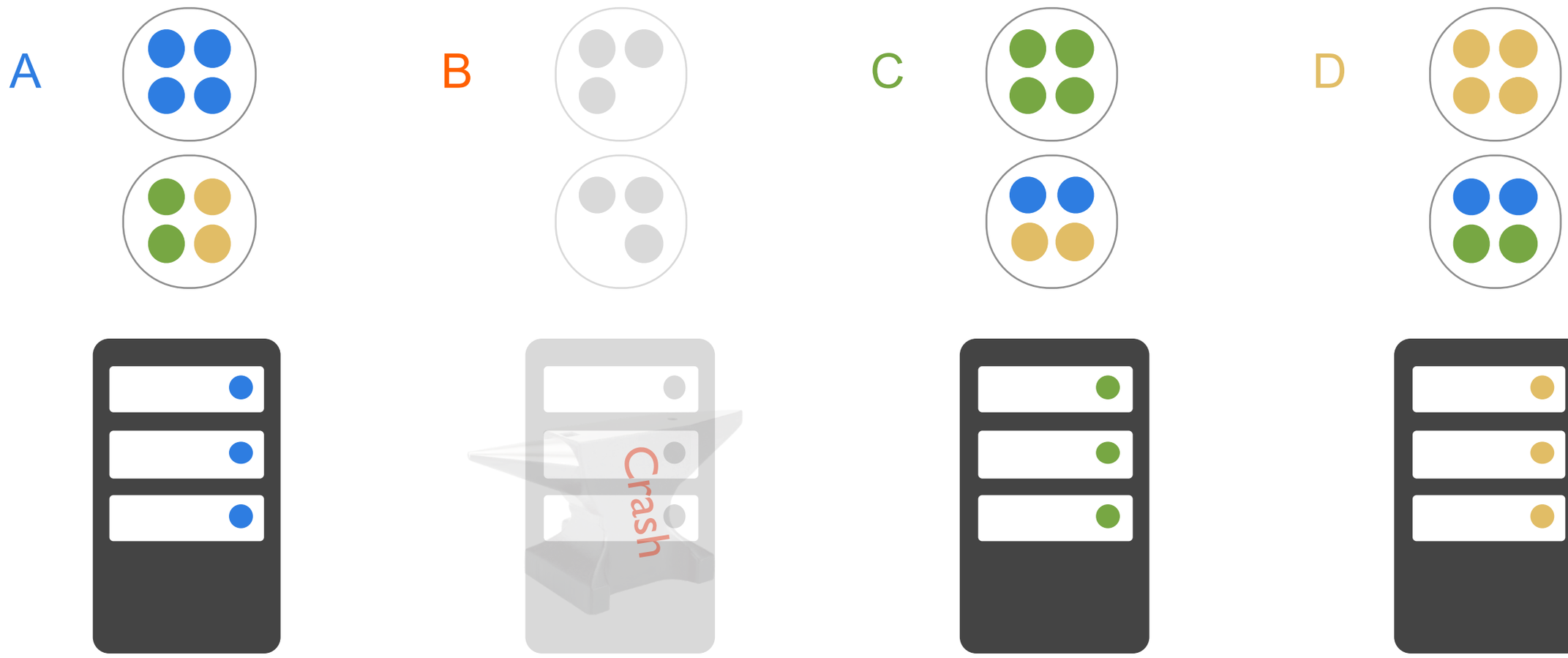
Backups Are Restored



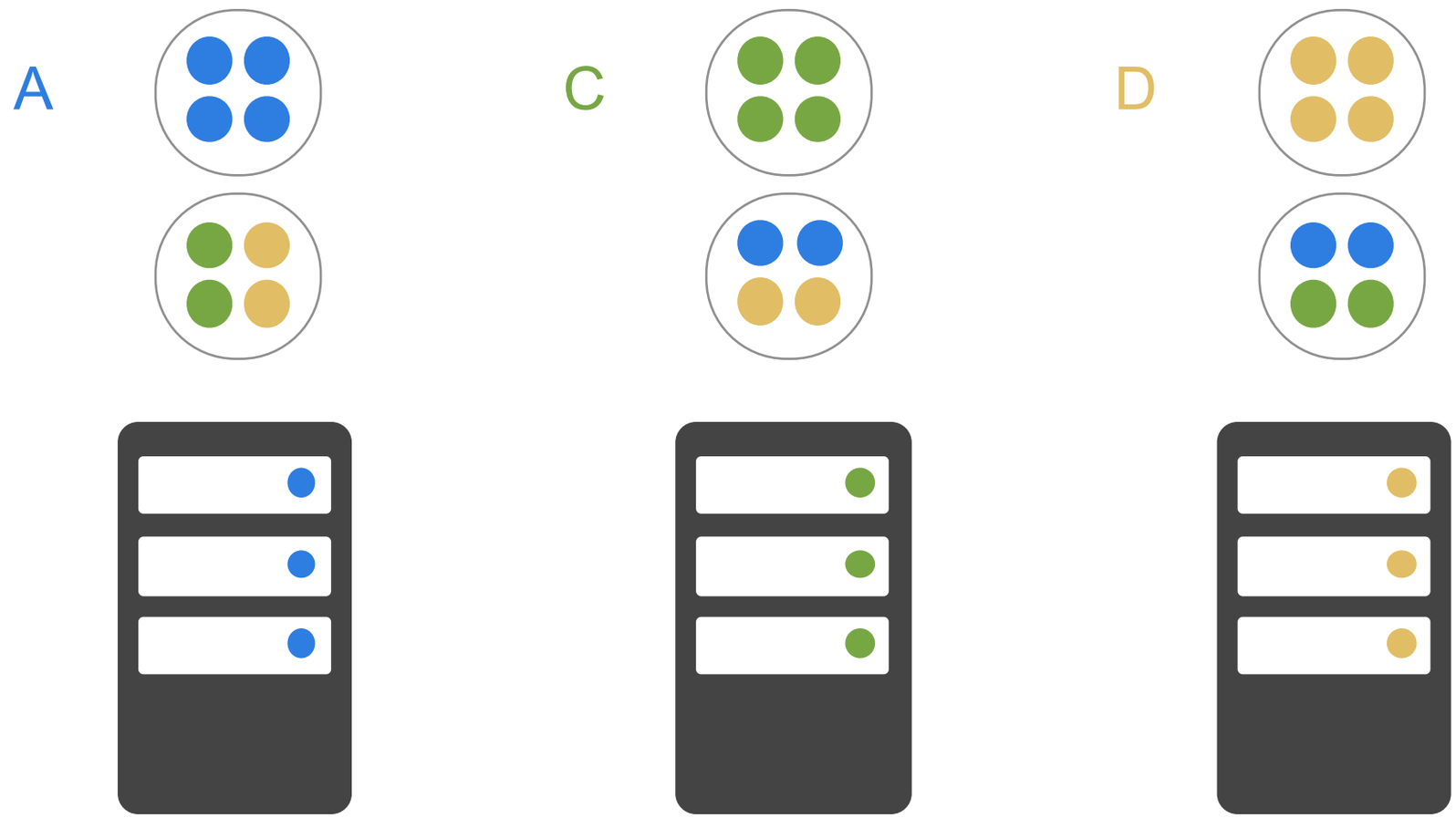
Backups Are Restored



Backups Are Restored



Recovery Is Complete



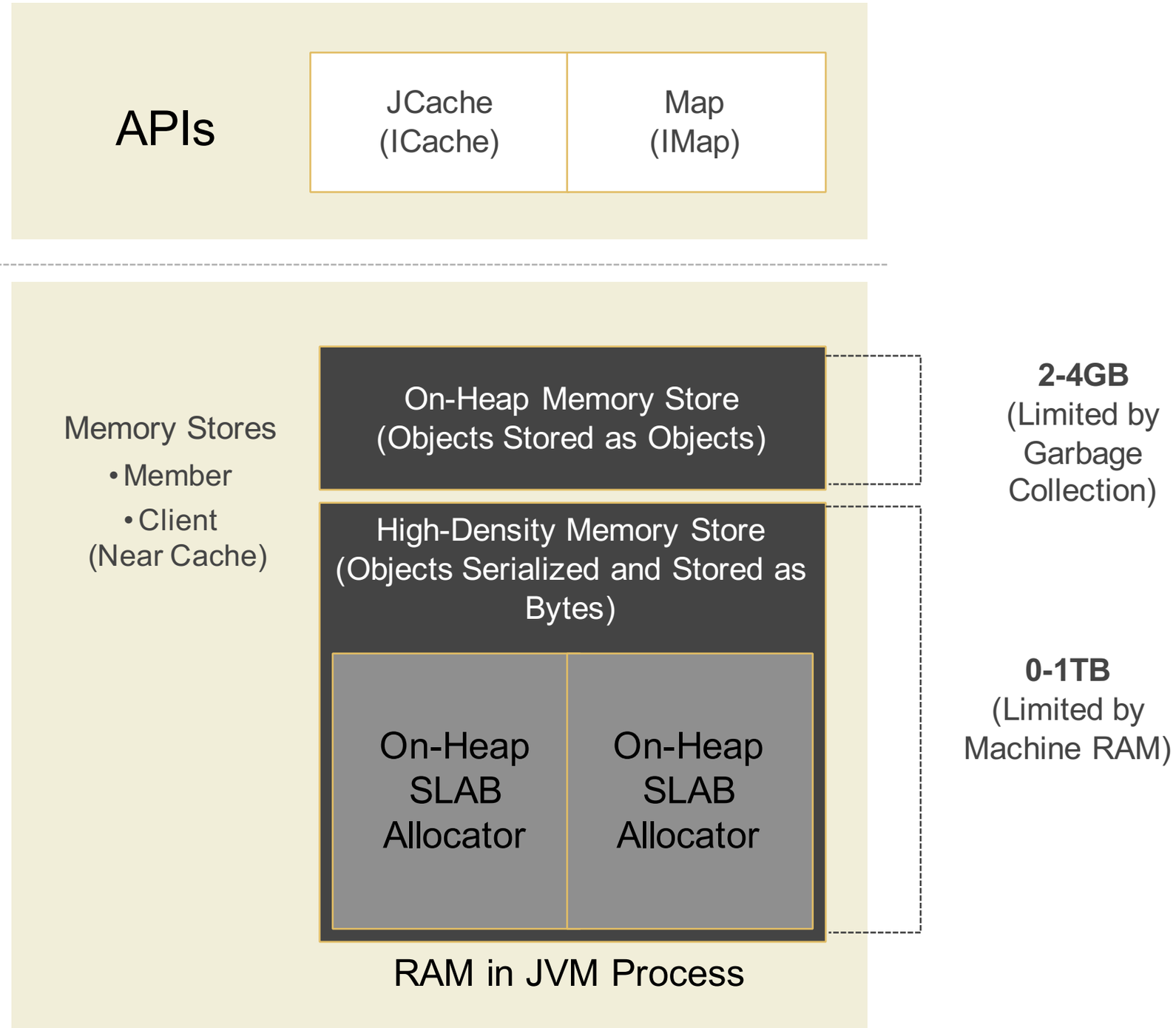
Scalability: Up or Out?

http://fc07.deviantart.net/fs71/i/2012/156/3/b/bruce_banner_by_lightning_stroke-d52erxe.png

Durability

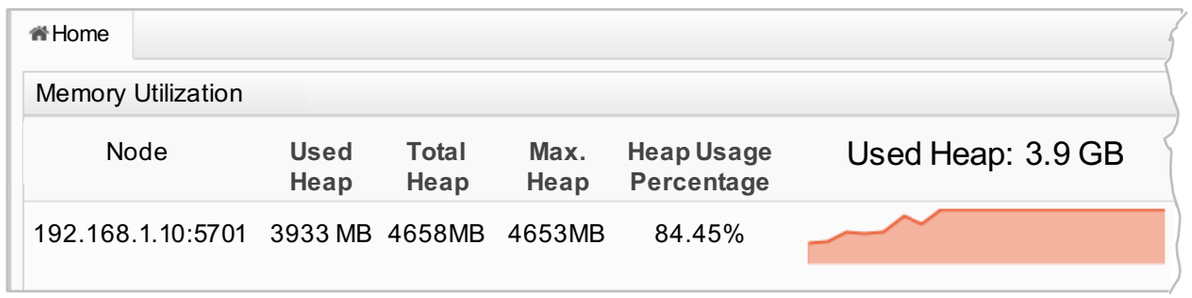
- Persist to or read from any external data store (SQL database, NoSQL data stores, REST service, EIS, file, etc.)
- Durability of application data (user sessions, shopping carts, master data, configs)
- Write through and write behind
- Hot Restart

High Density Caching

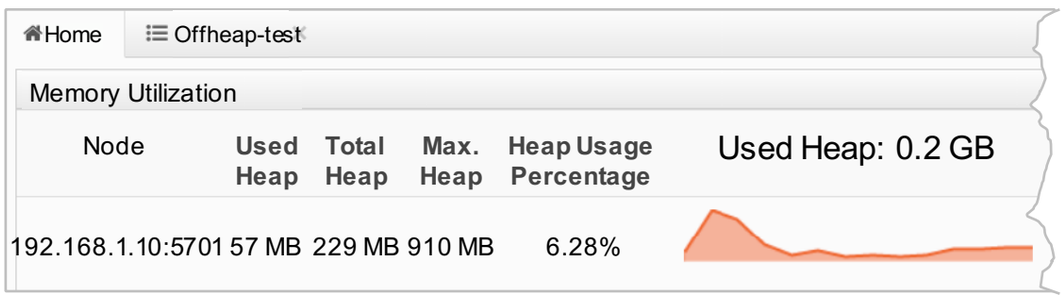


On Heap Vs. High-Density Memory Management

On Heap Memory		HD Memory v2
0 MB	*Native*	*3.3 GB*
3.9 GB	*Heap Storage*	*0.6 GB*
9 (4900 ms)	*Major GC*	*0 (0 ms)*
31 (4200 ms)	*Minor GC*	*356 (349 ms)*



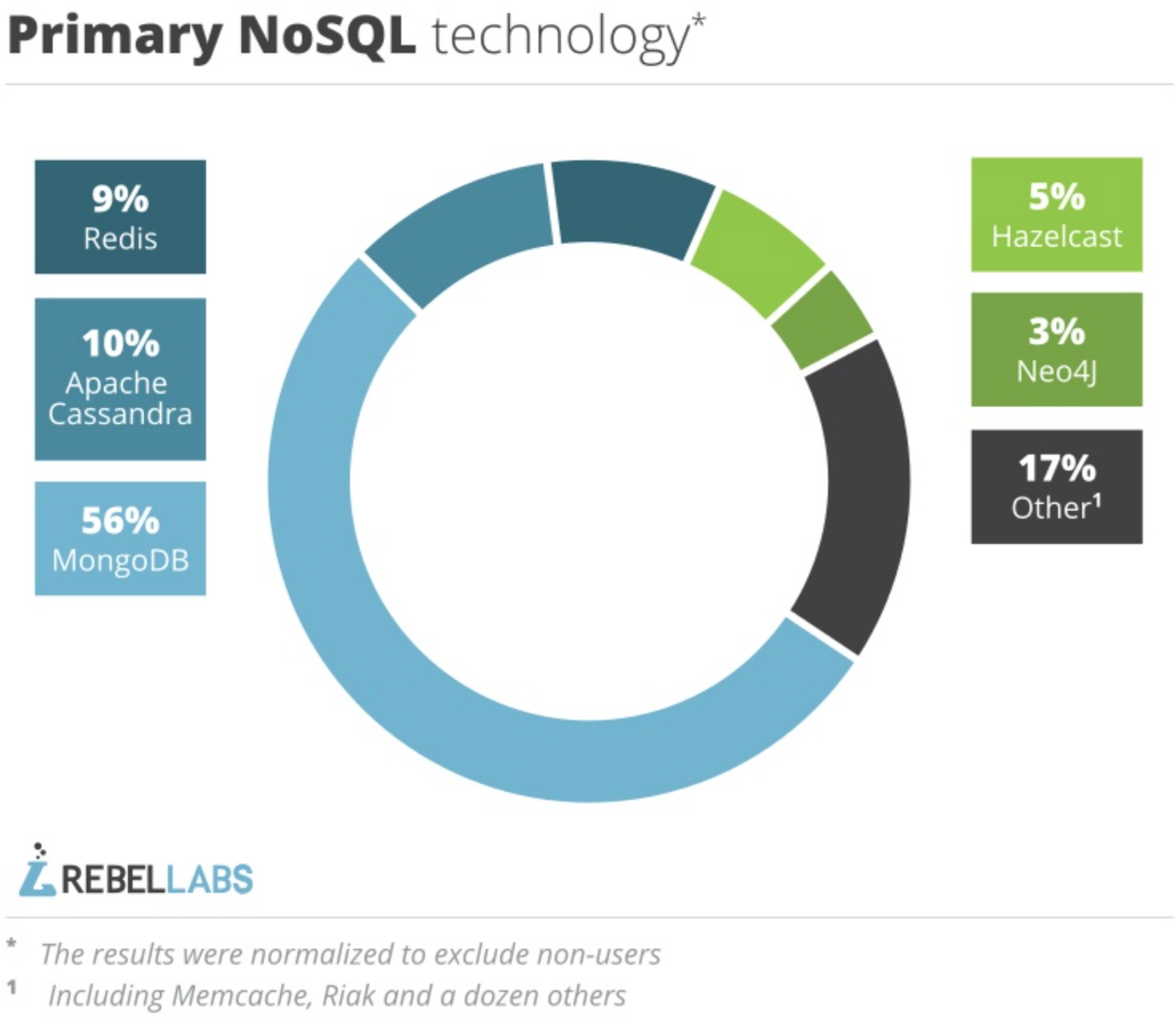
Example: On Heap Memory



Example: HD Memory v2

Do I really need NoSQL?

NoSQL Landscape



http://hazelcast.com/use-cases/nosql/nosql-data-store/

NOW YOU KNOW IMDG!

QA

Question, please?