Ground up Introduction to In-Memory Data (Grids)

QCON 2015
NEW YORK, NY

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 ?

hazelcast
git
Viktor
GAMOV
hazelcast
QCon by InfoQ
New York 2015
SPEAKER
BETTER CA

About me

Viktor Gamov

@gamussa

- Solutions Architect at Hazelcast http://hazelcast.com/services/solutions-team/
- International Java Conferences speaker http://lanyrd.com/gamussa/
- Co-author of O’Reilly’s «Enterprise Web Development»
- Co-organizer of User Groups
  - Princeton JUG http://www.meetup.com/NJFlex/
  - Hazelcast User Group NYC http://www.meetup.com/hug-nyc/

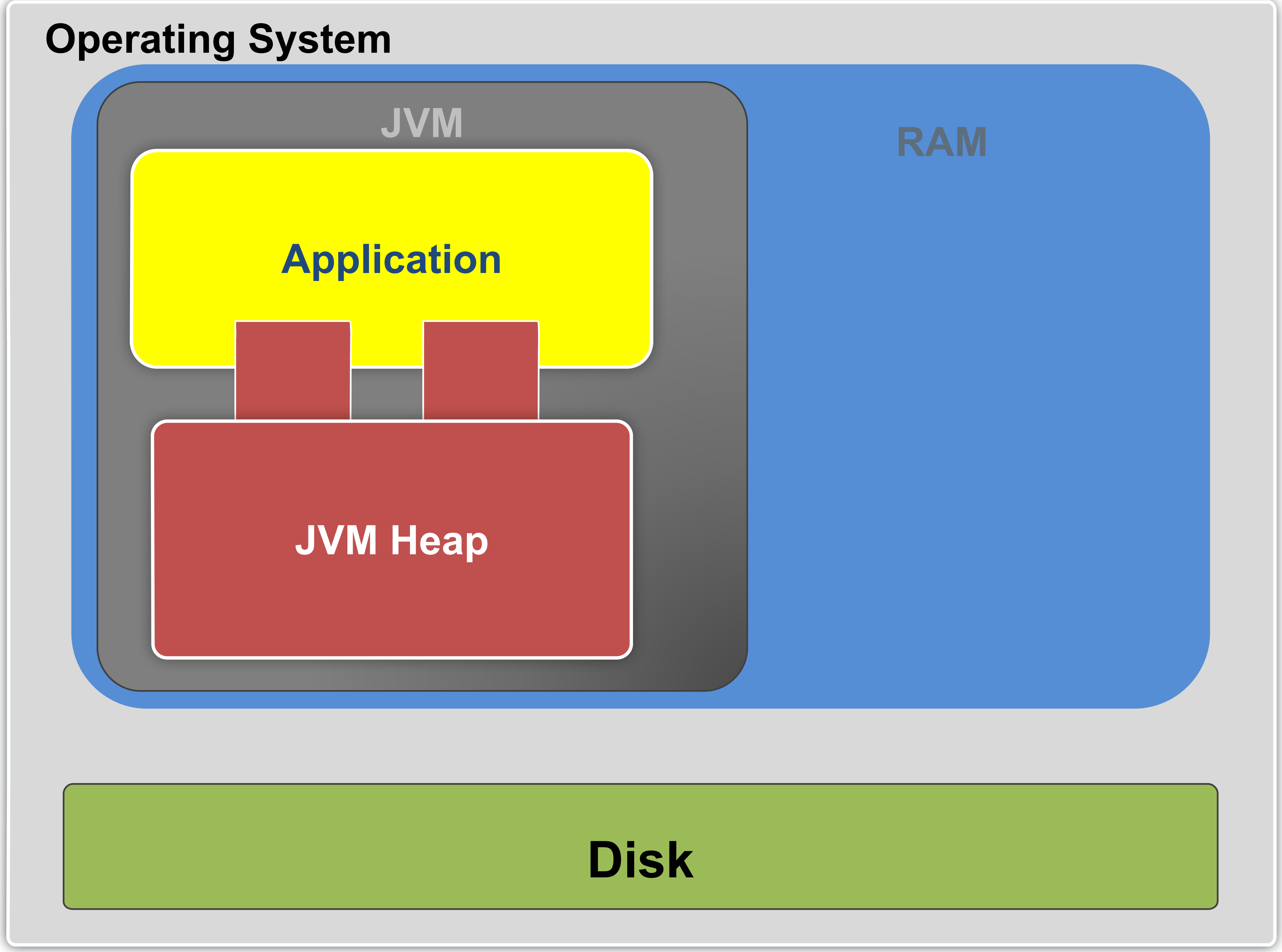
Agenda

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

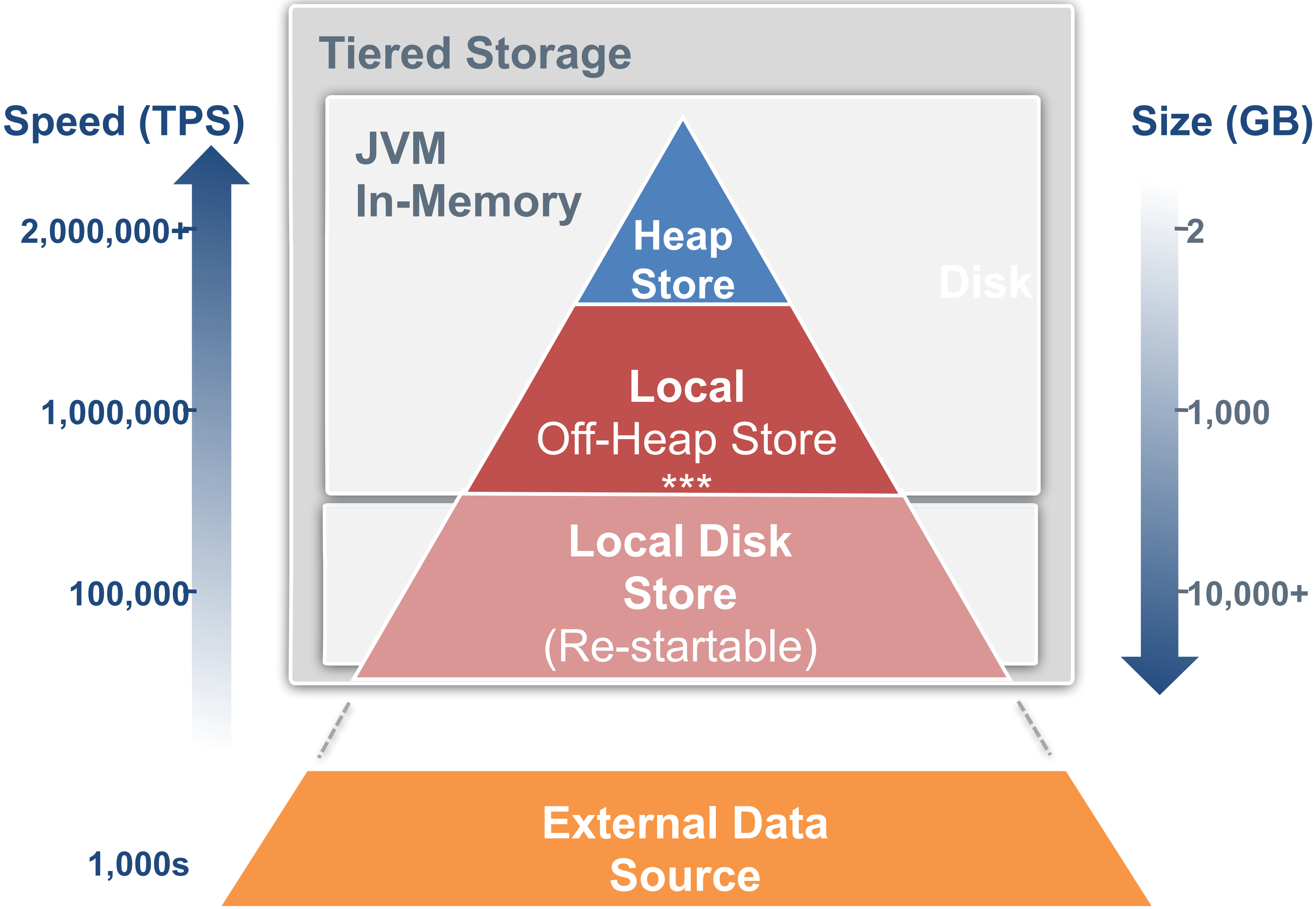
Why
In-Memory Data?

Problem

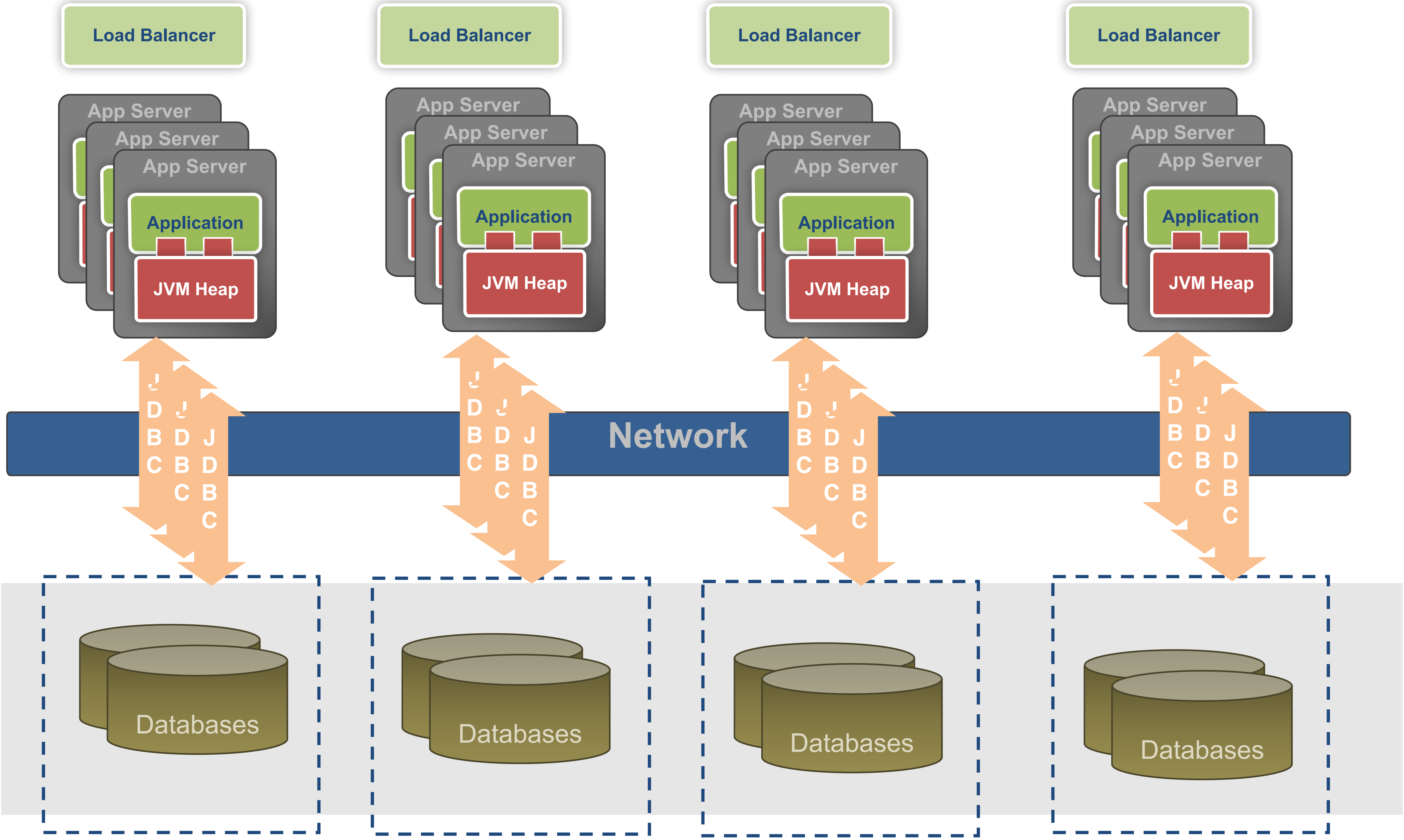
Anatomy of a JAVA Application



Anatomy of an Application: Tiered Data Storage



Common Enterprise Application Landscape



© 2015 Hazelcast, Inc.

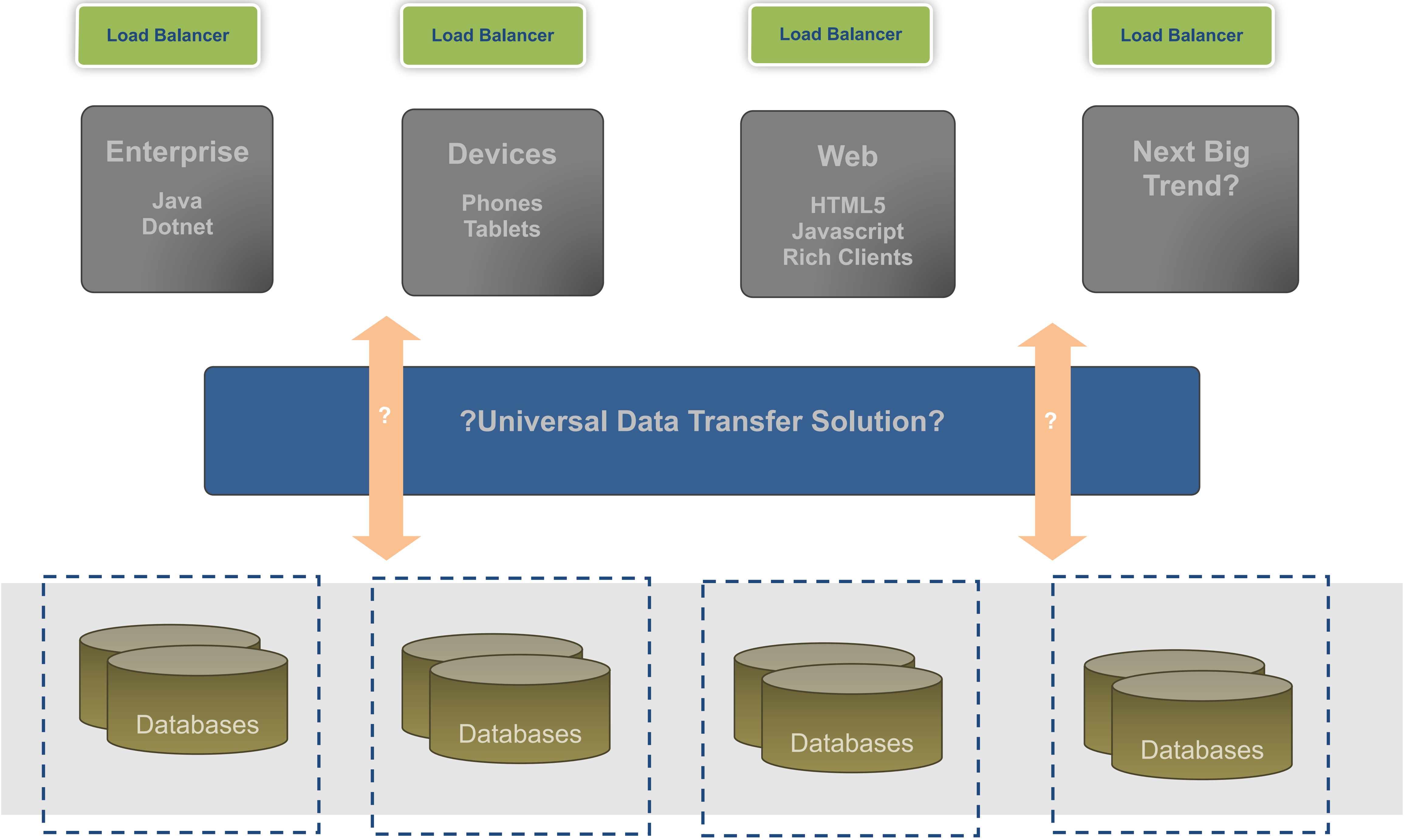
Performance & Scalability Challenges

- 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

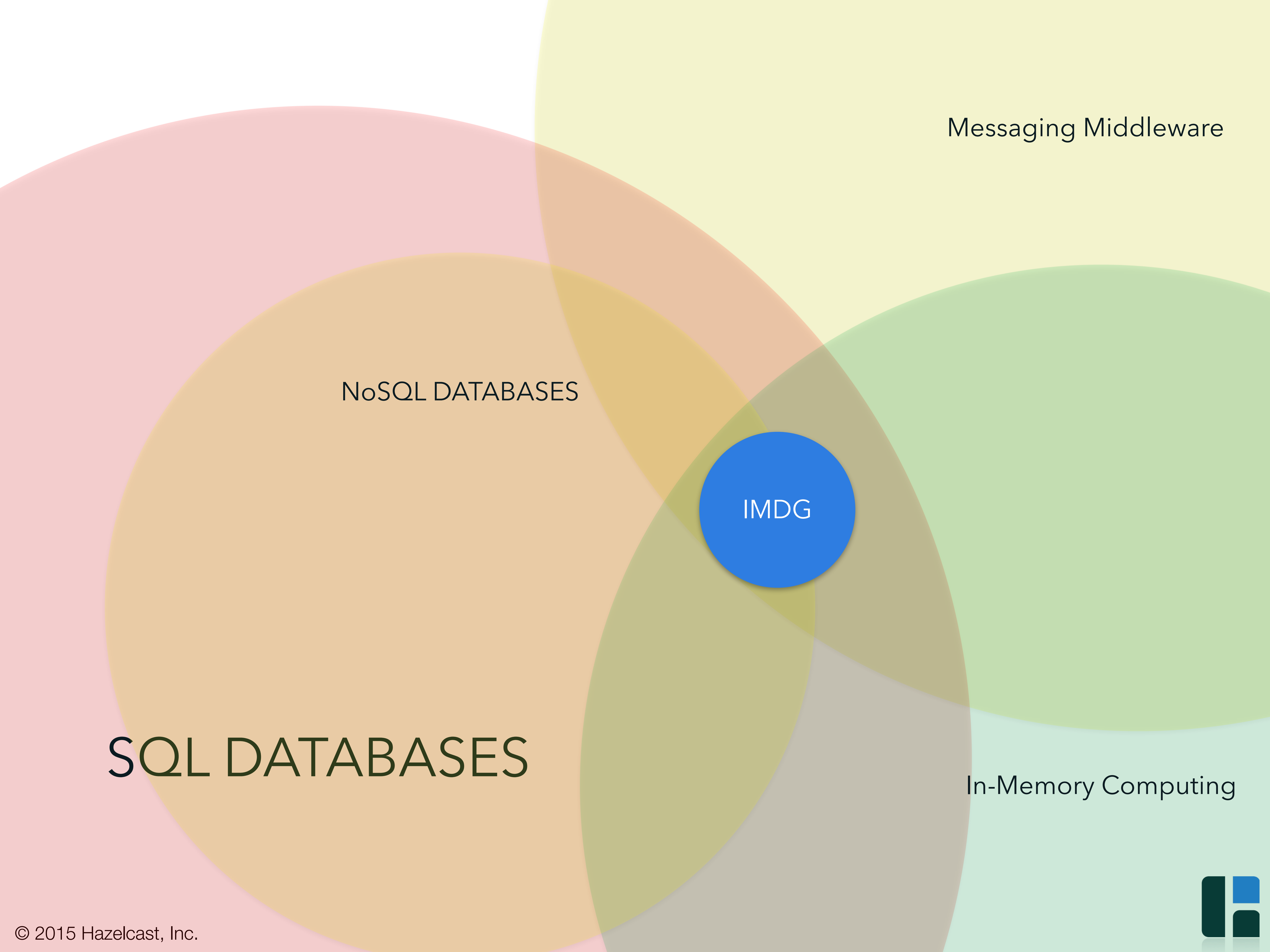
Performance & Scalability Challenges

- 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

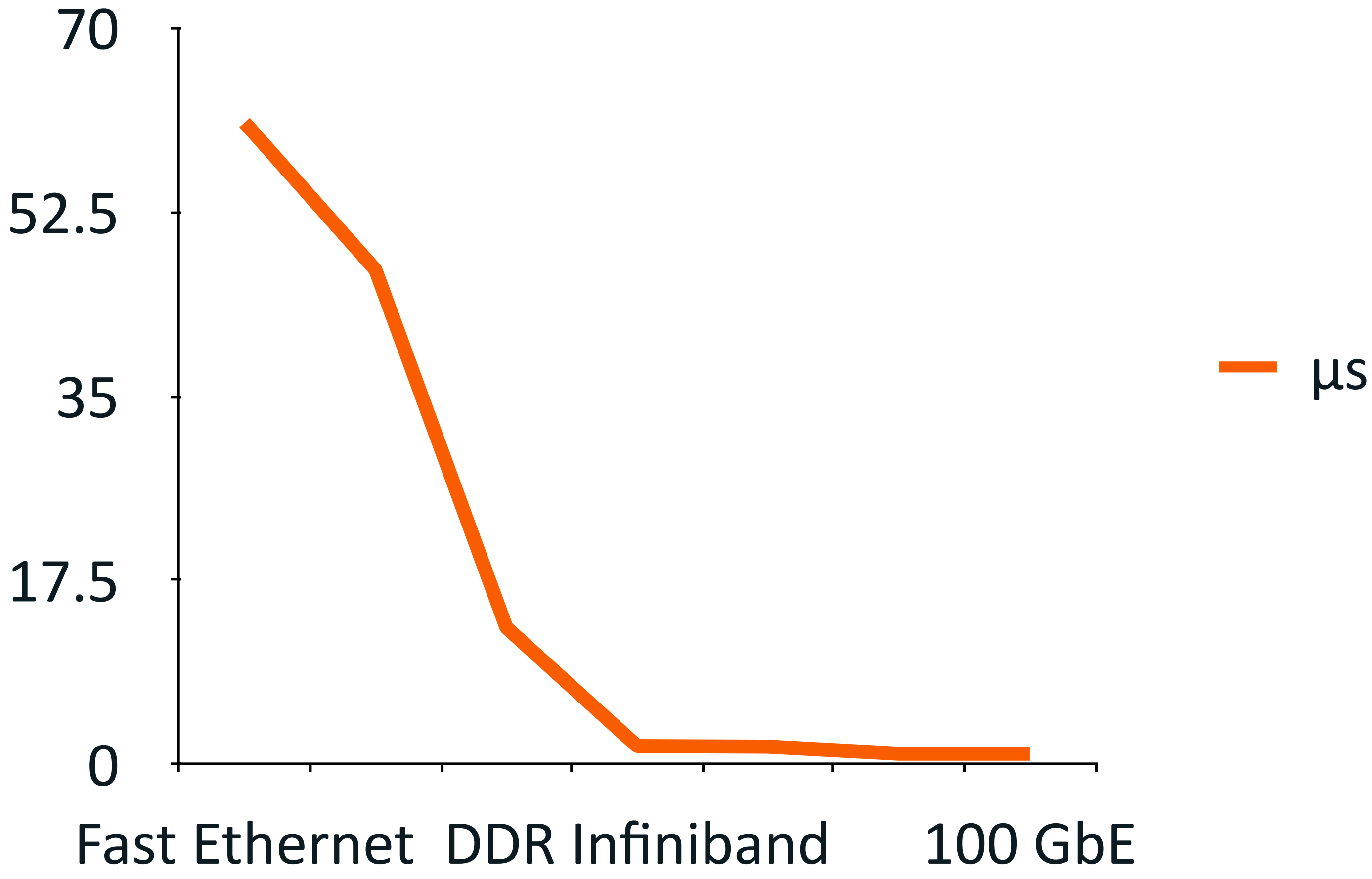


In-Memory Data Lanscape

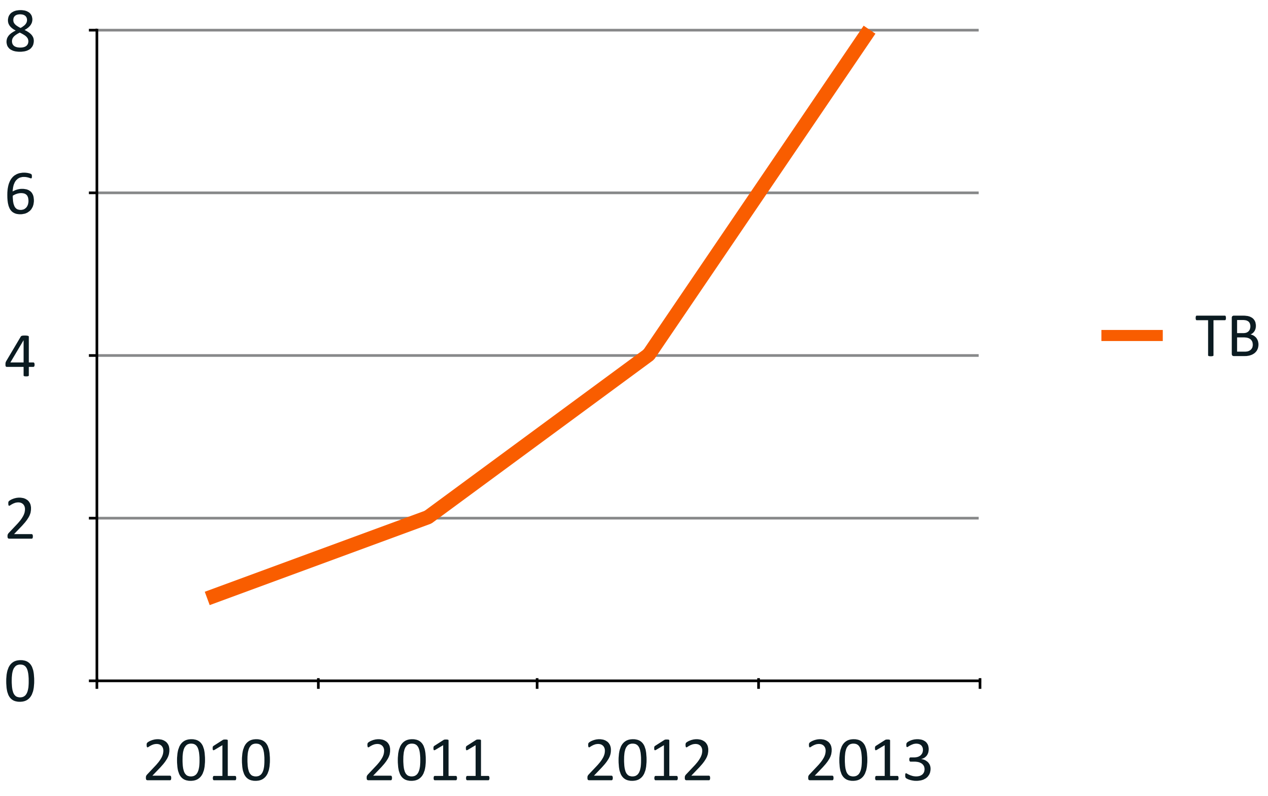

Messaging Middleware
NoSQL DATABASES
IMDG
SQL DATABASES
In-Memory Computing

Why In-Memory Now?

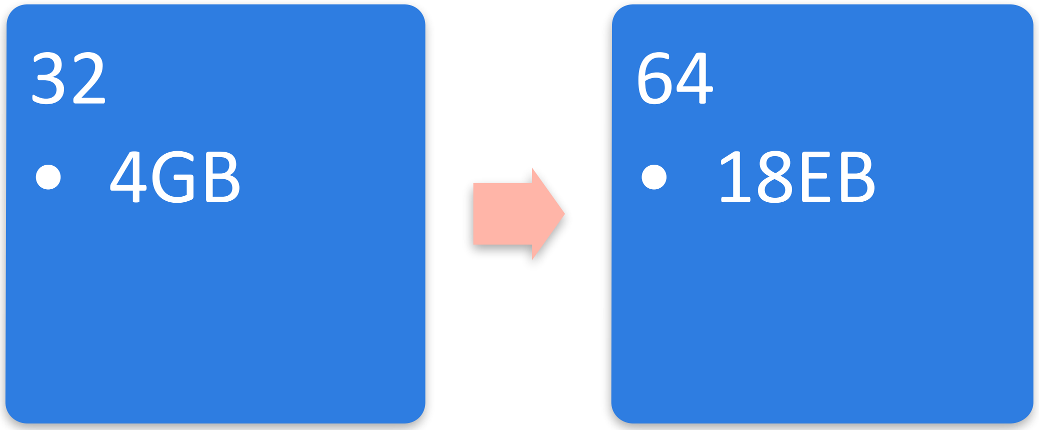
Network



RAM per Commodity Server



Addressable Memory



32

- 4GB

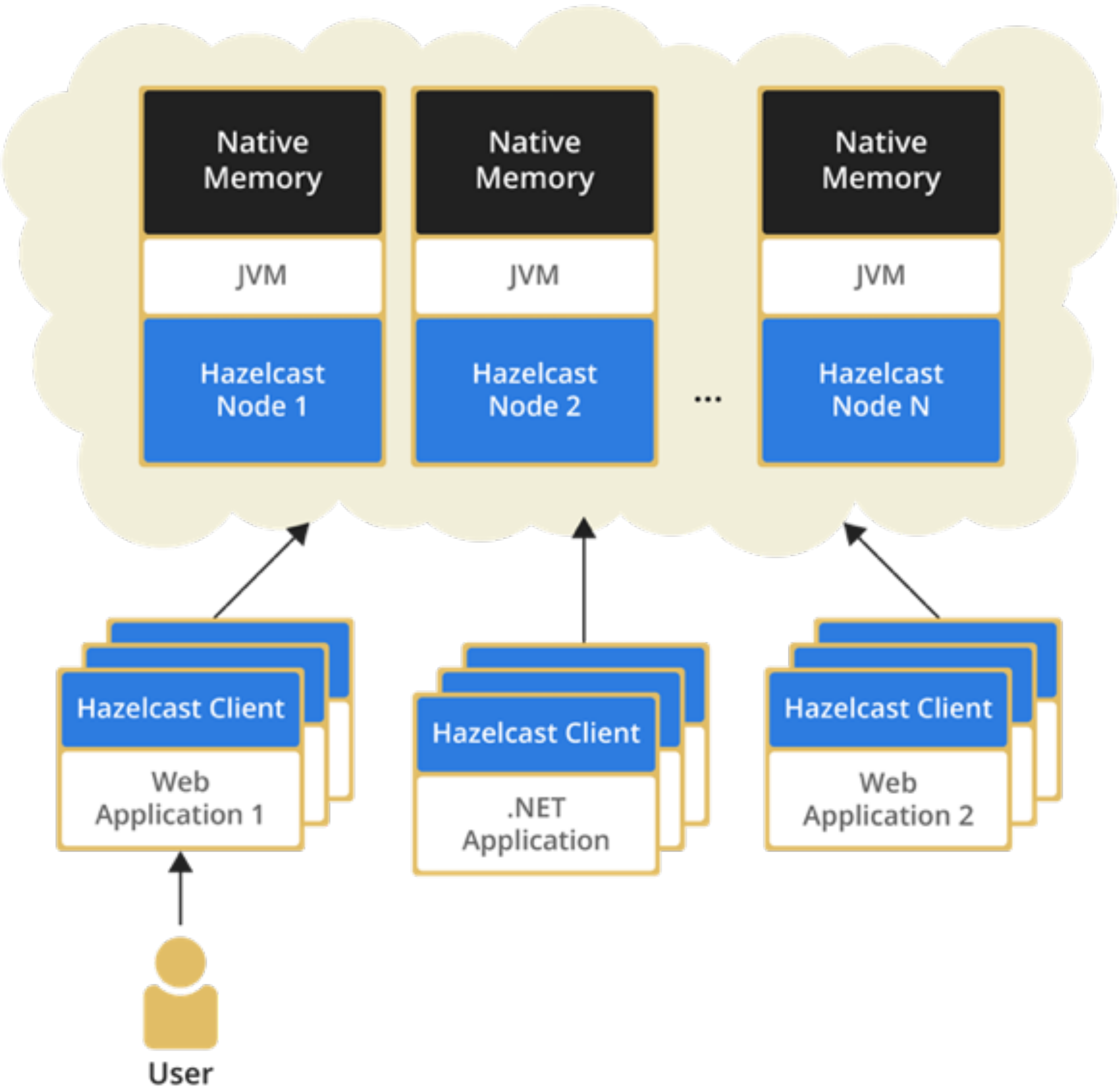
64

- 18EB

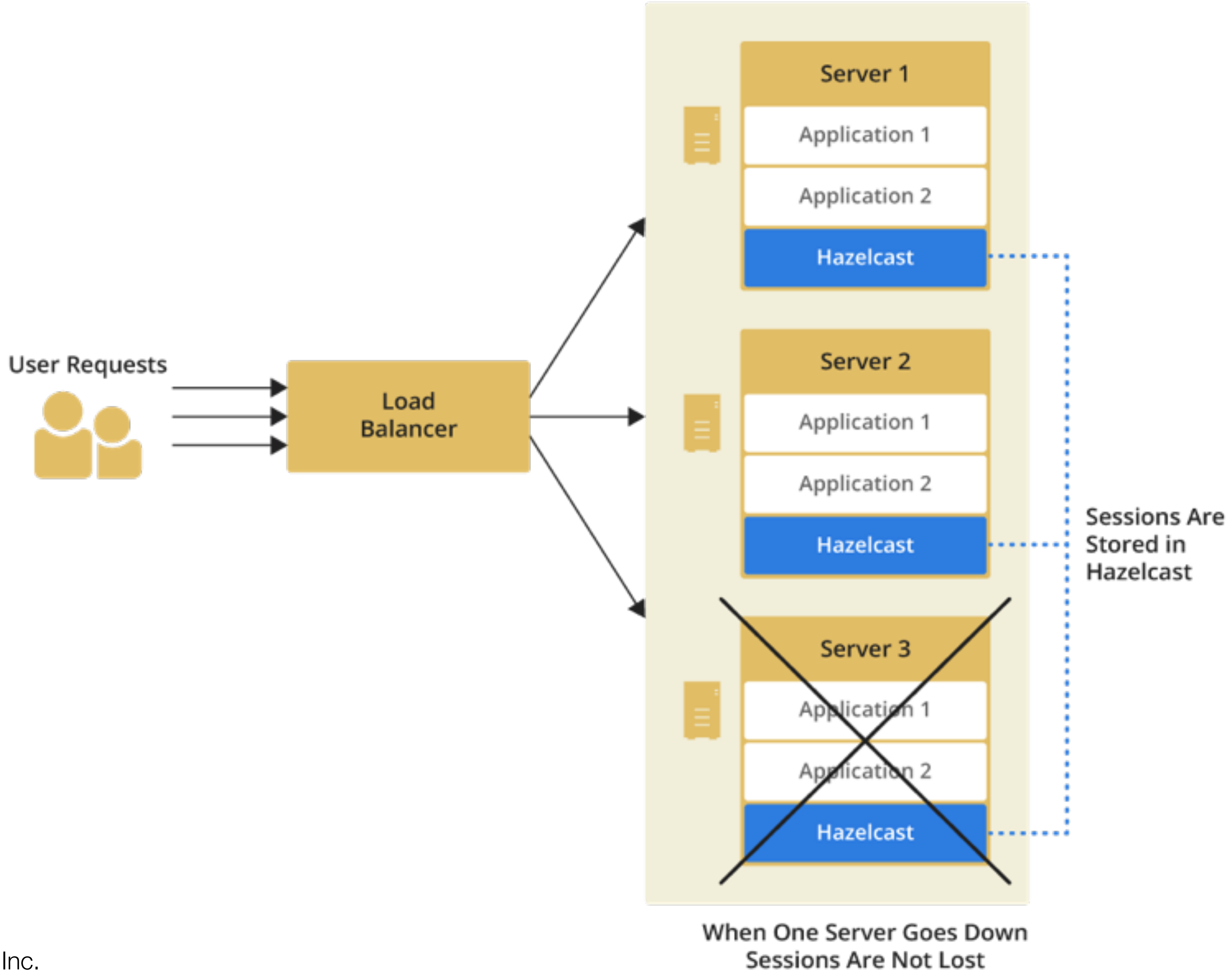
Distributed Data Applications

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

Cache-as-a-Service



Durability of application data

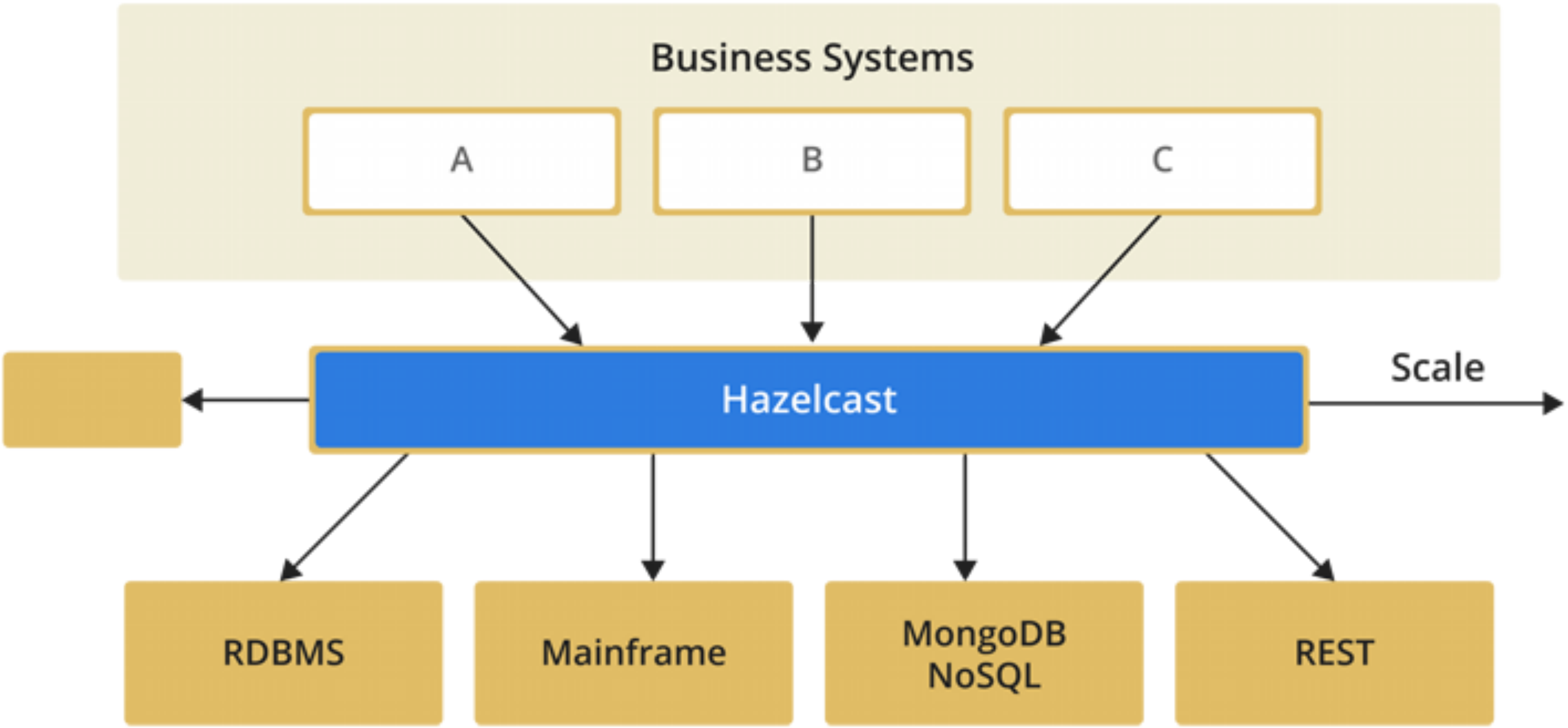


When One Server Goes Down
Sessions Are Not Lost

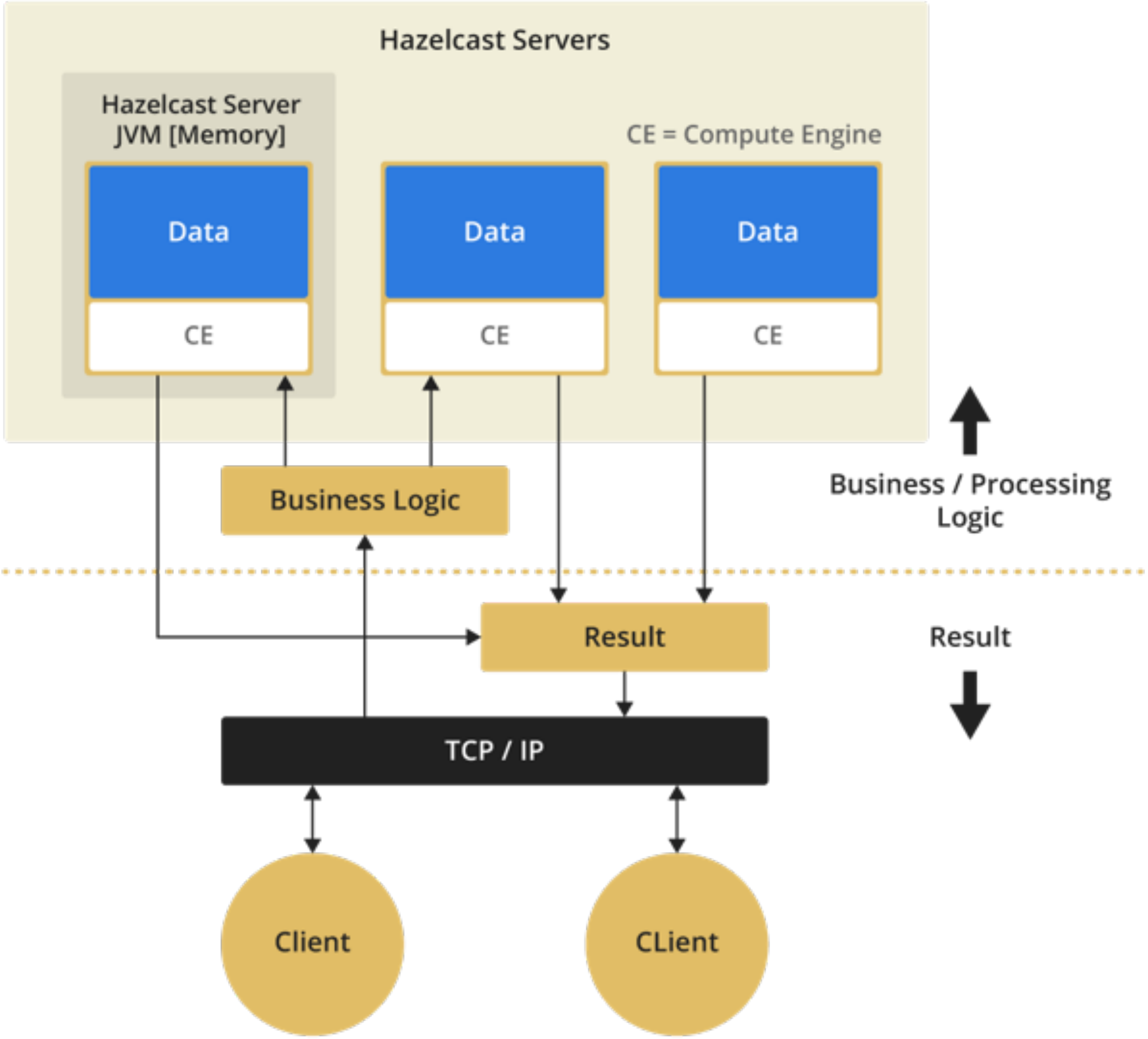
Application Scaling

- Elastic Scalability (just add more servers and they cluster automatically)
- Super Speeds (in-memory transaction speeds)
- High Availability (can deploy in backup pairs or even WAN replicated)
- Fault Tolerance (no single-point-of-failure)
- Cloud Readiness (deploy right into EC2, GCE, Docker)

Database Caching



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

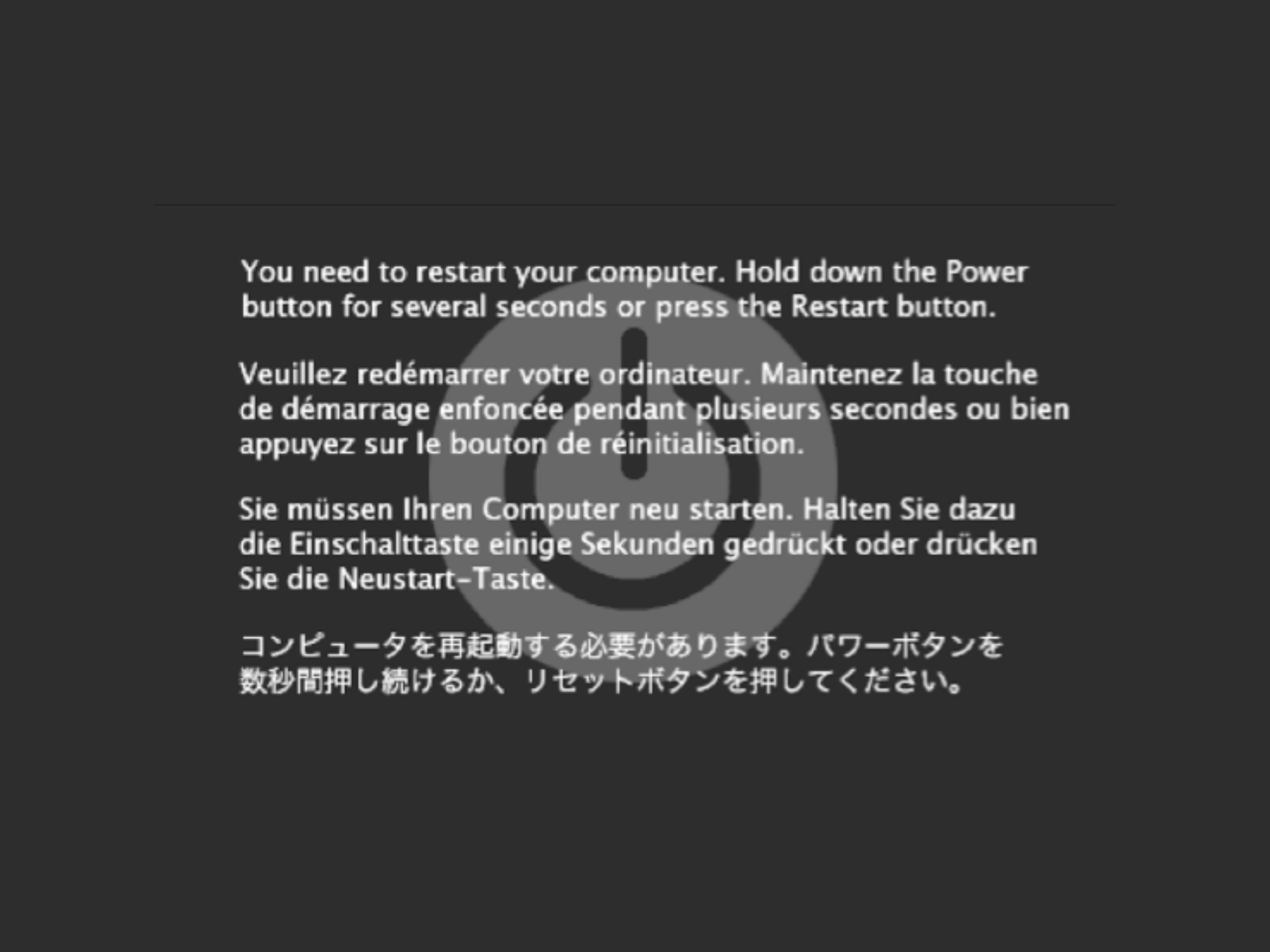
Data Distribution

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

Replication vs. Partitioning

- **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.



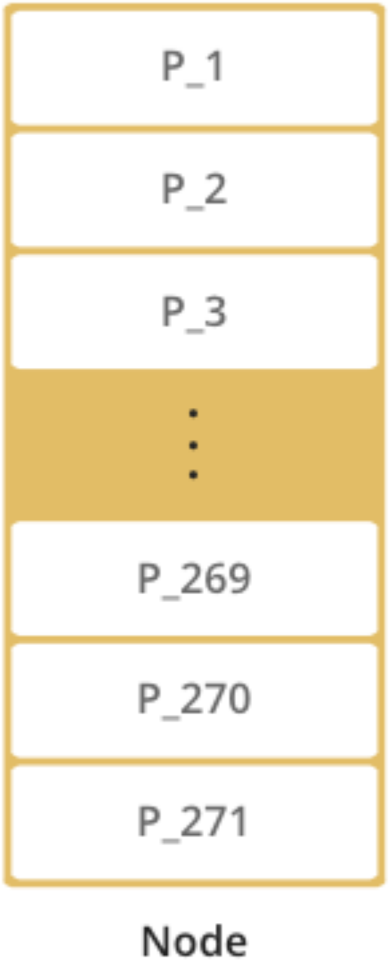
You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

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

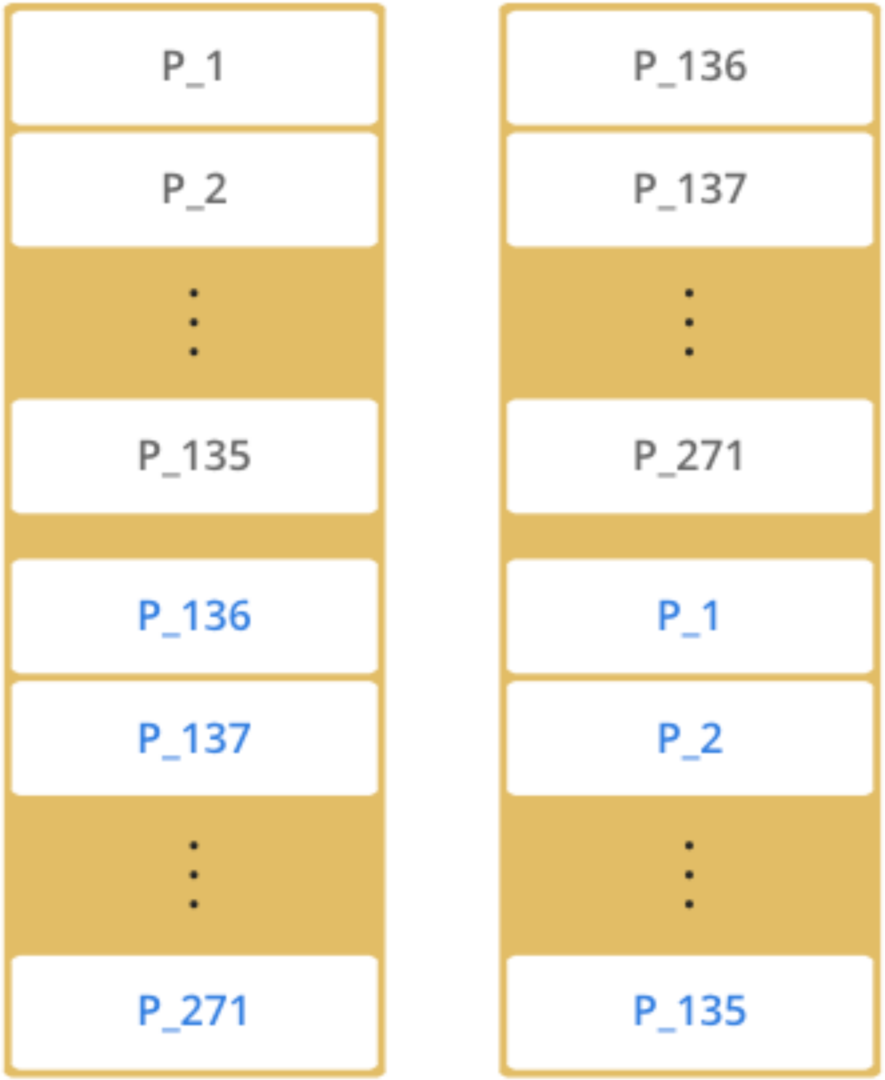
Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

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



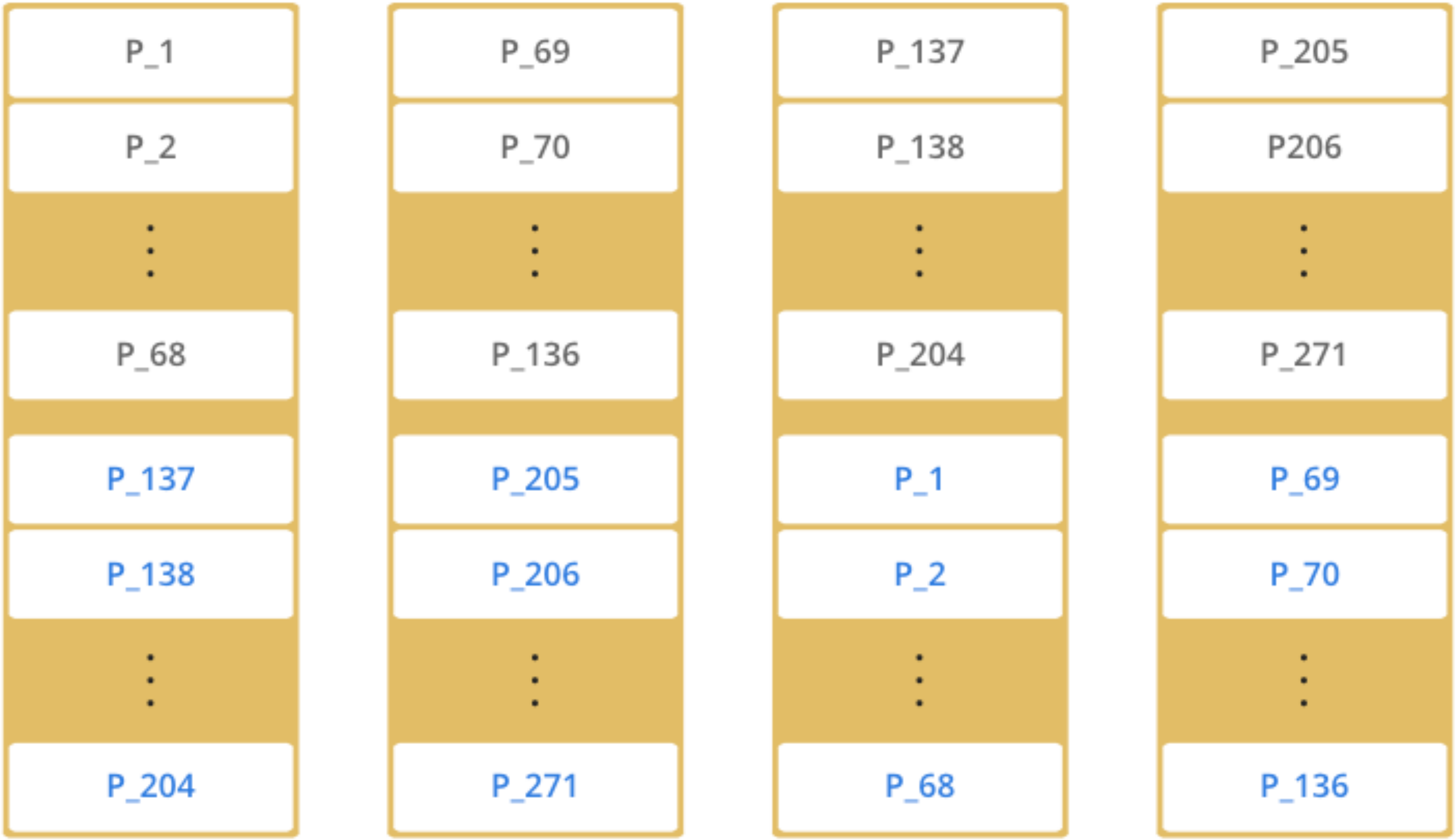
Data Partitioning



P_1
P_2
⋮
P_135
P_136
P_137
⋮
P_271

P_136
P_137
⋮
P_271
P_1
P_2
⋮
P_135

Data Partitioning



P_1	P_69	P_137	P_205
P_2	P_70	P_138	P206
⋮	⋮	⋮	⋮
P_68	P_136	P_204	P_271
P_137	P_205	P_1	P_69
P_138	P_206	P_2	P_70
⋮	⋮	⋮	⋮
P_204	P_271	P_68	P_136

Rebalance Data On New Node

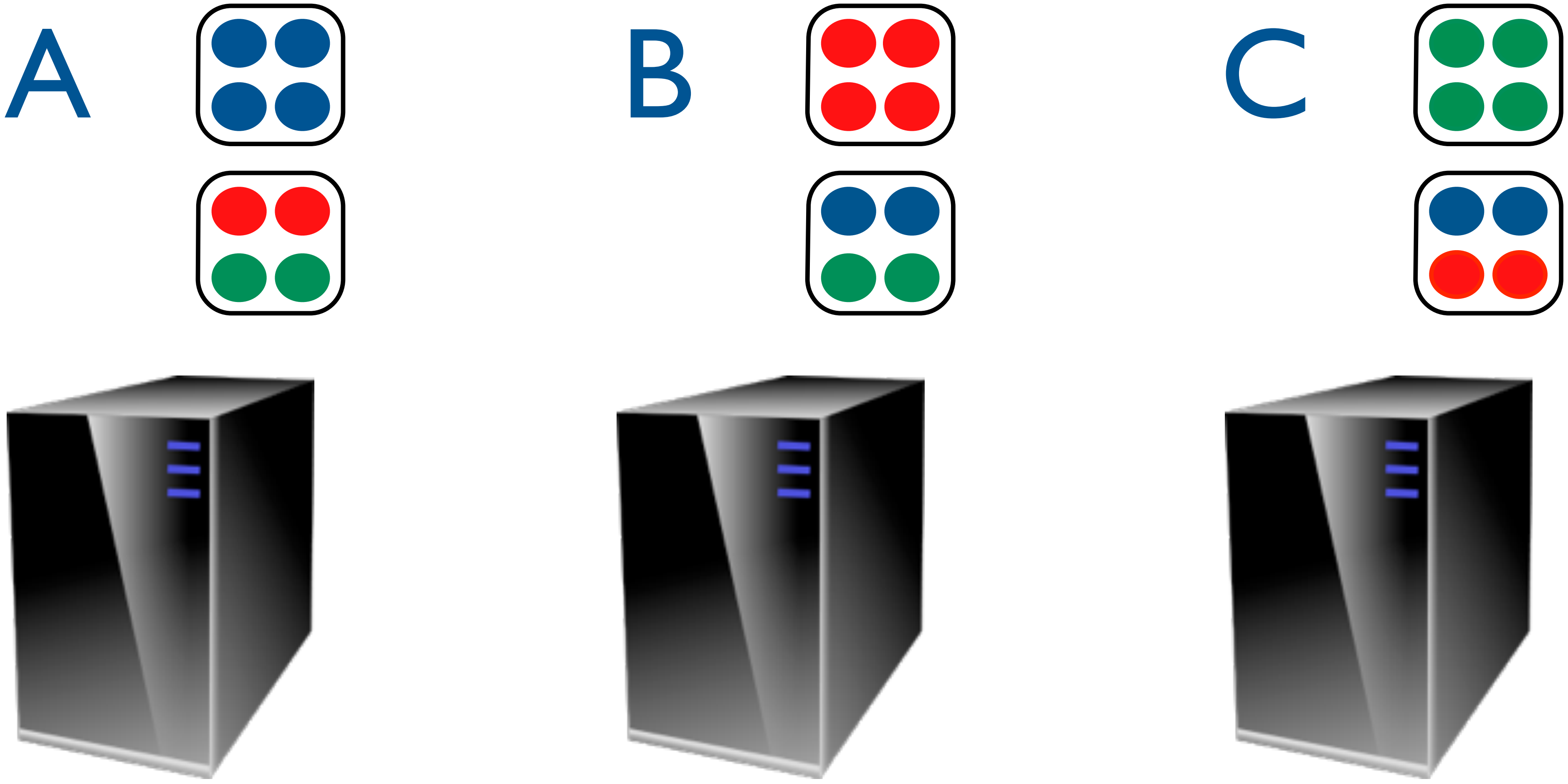
Distributed Maps

Fixed number of partitions (default 271)

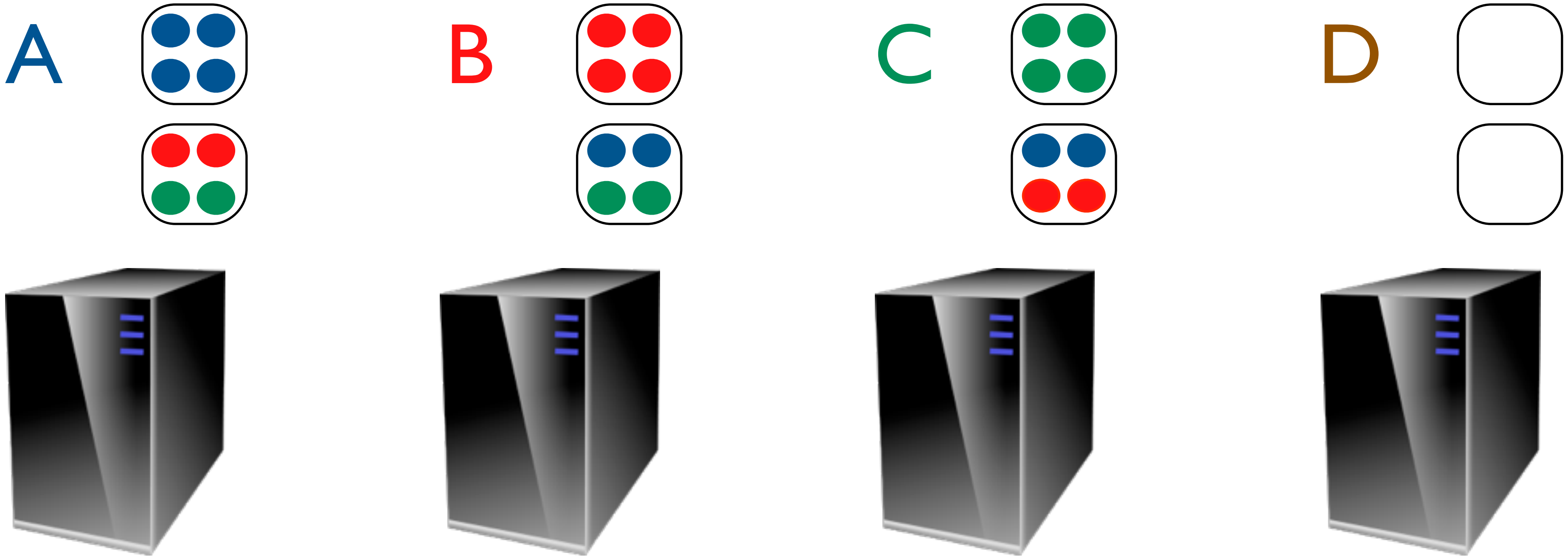
Each key falls into a partition

partitionId = hash(keyData)%PARTITION_COUNT

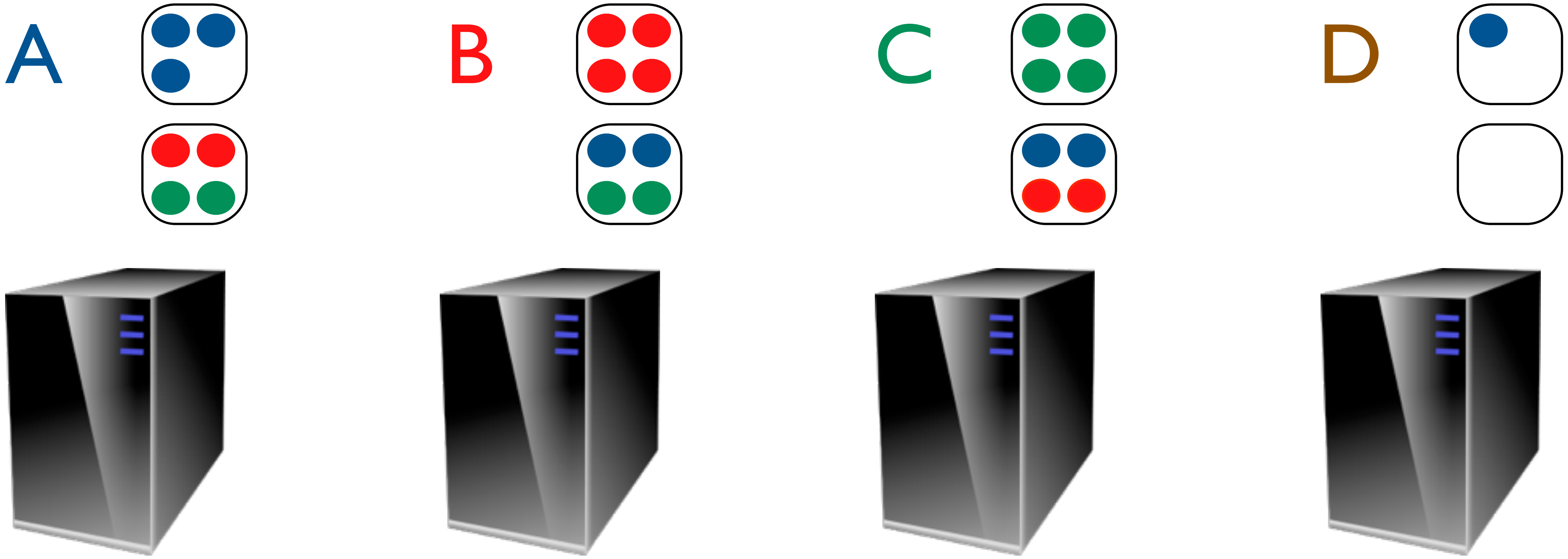
Partition ownerships are reassigned upon membership



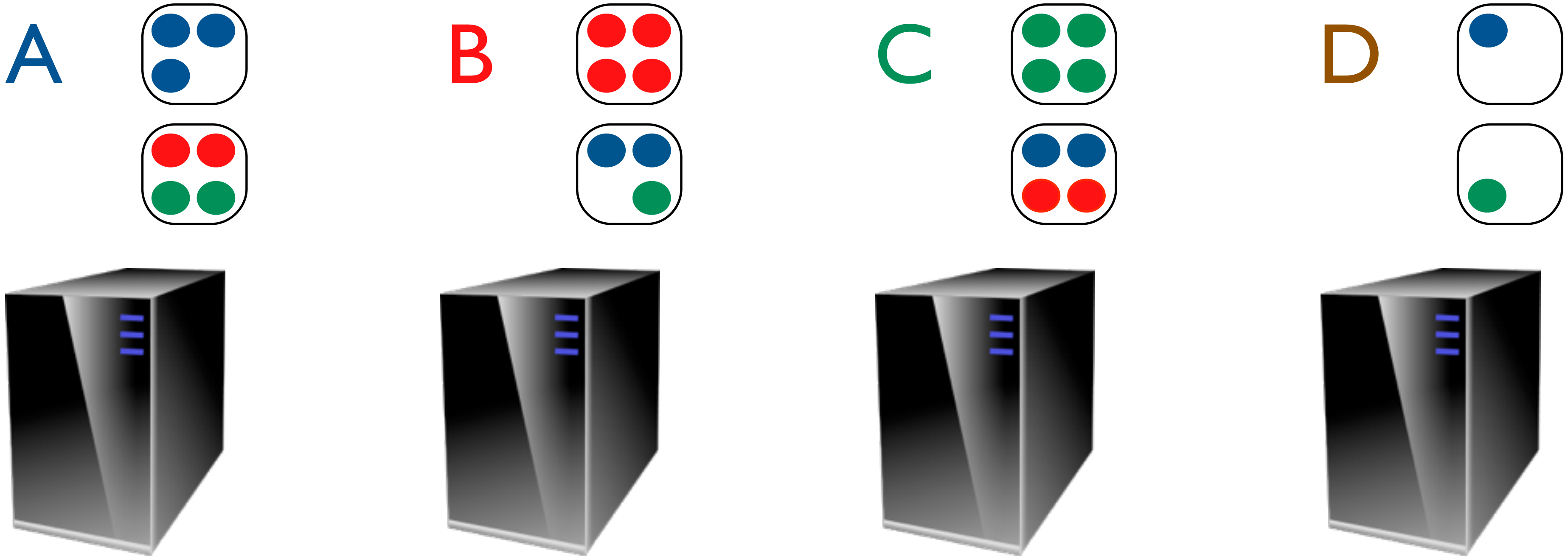
New Node Added



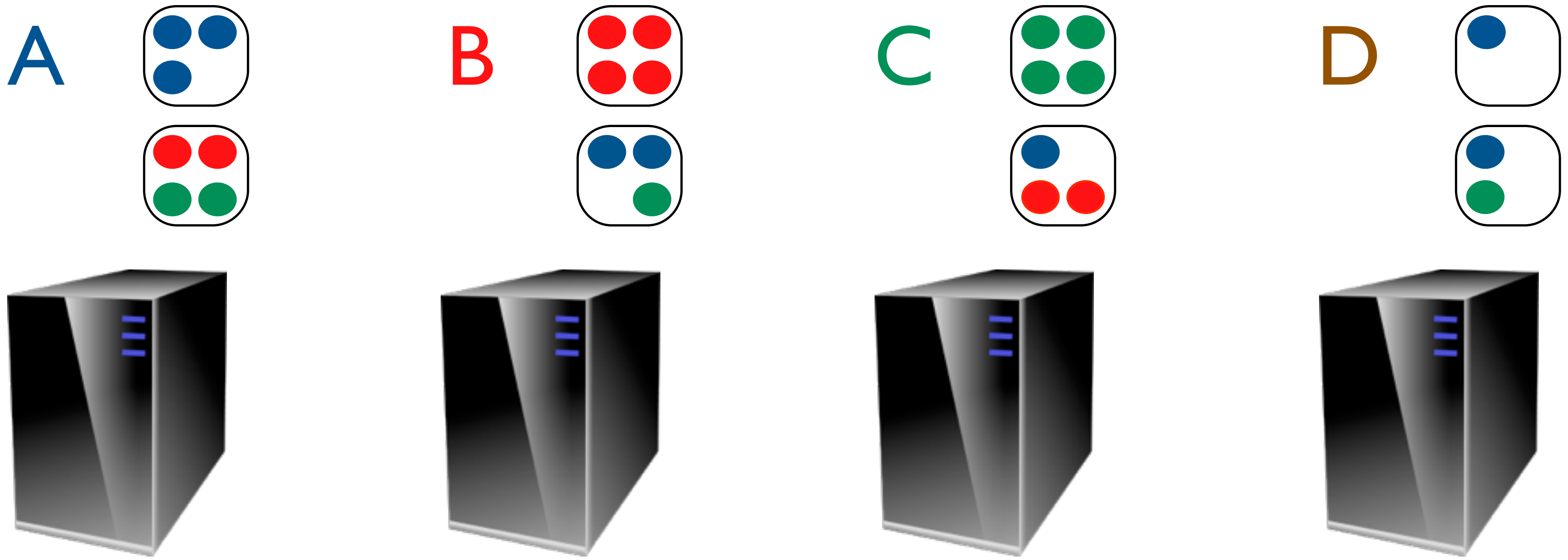
Migration



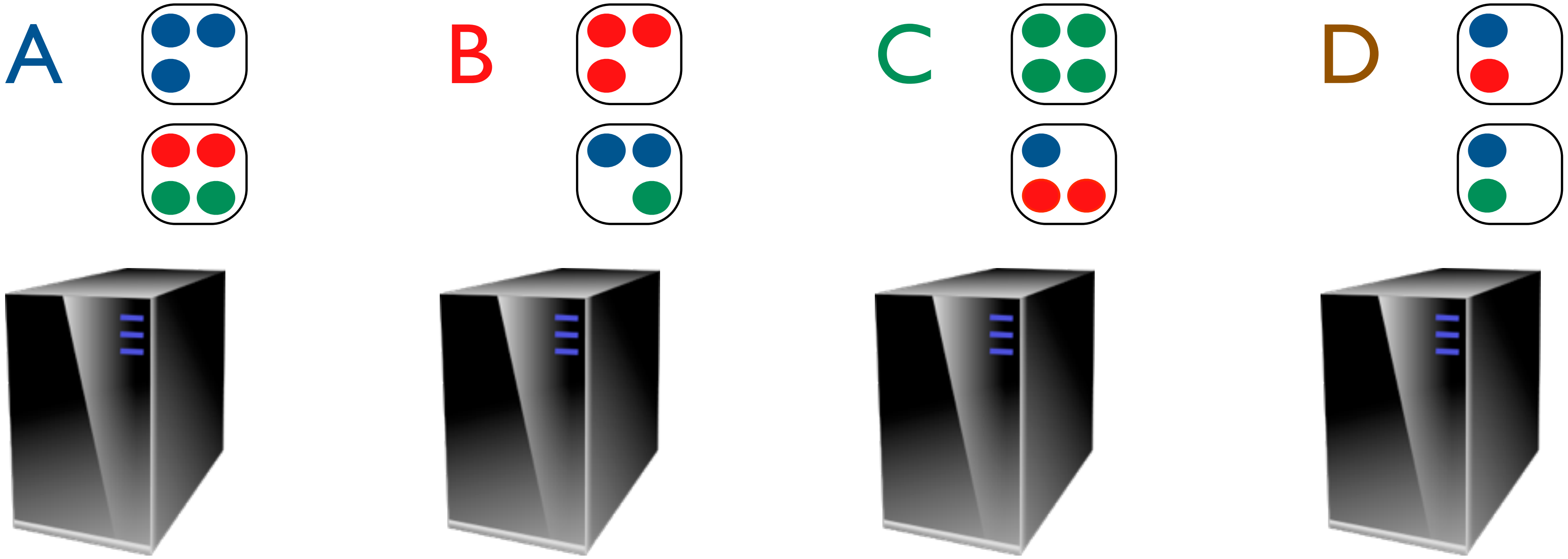
Migration



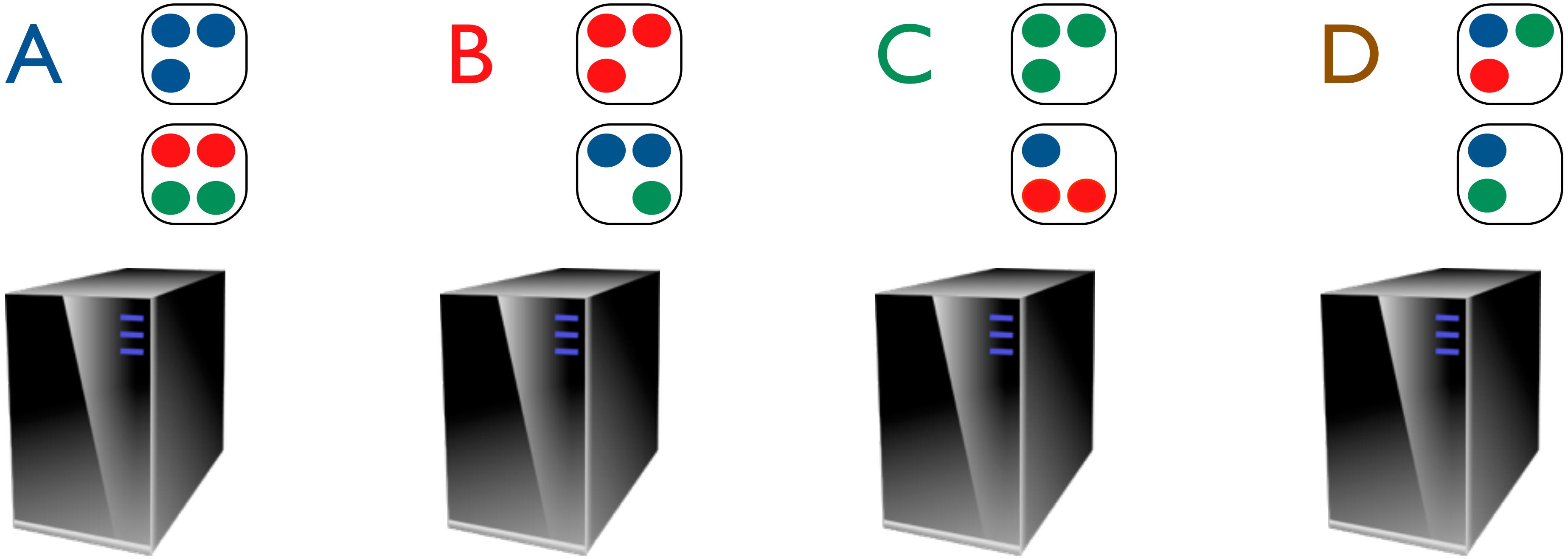
Migration



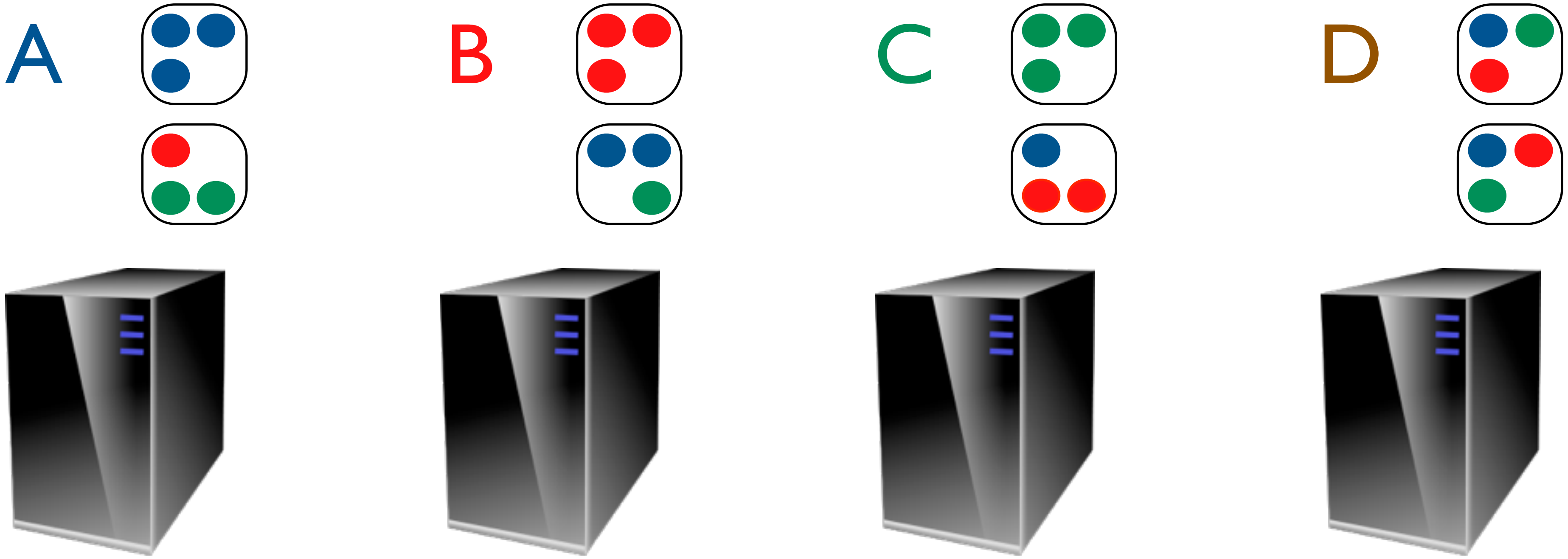
Migration



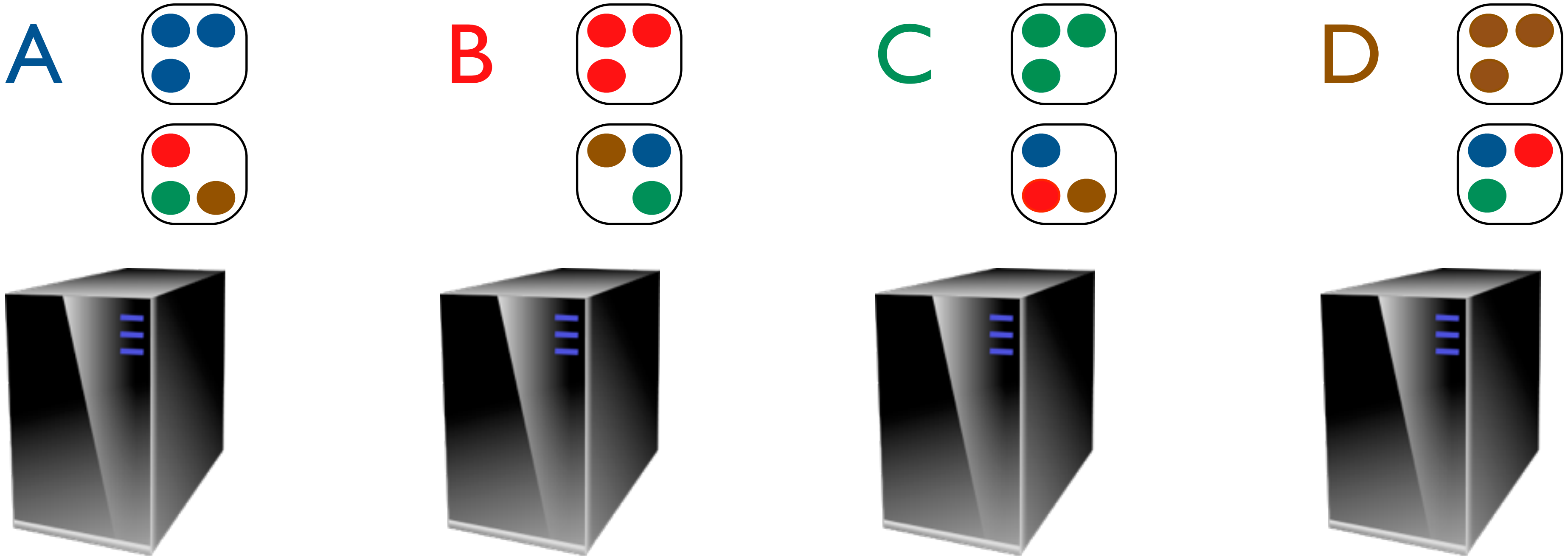
Migration



Migration

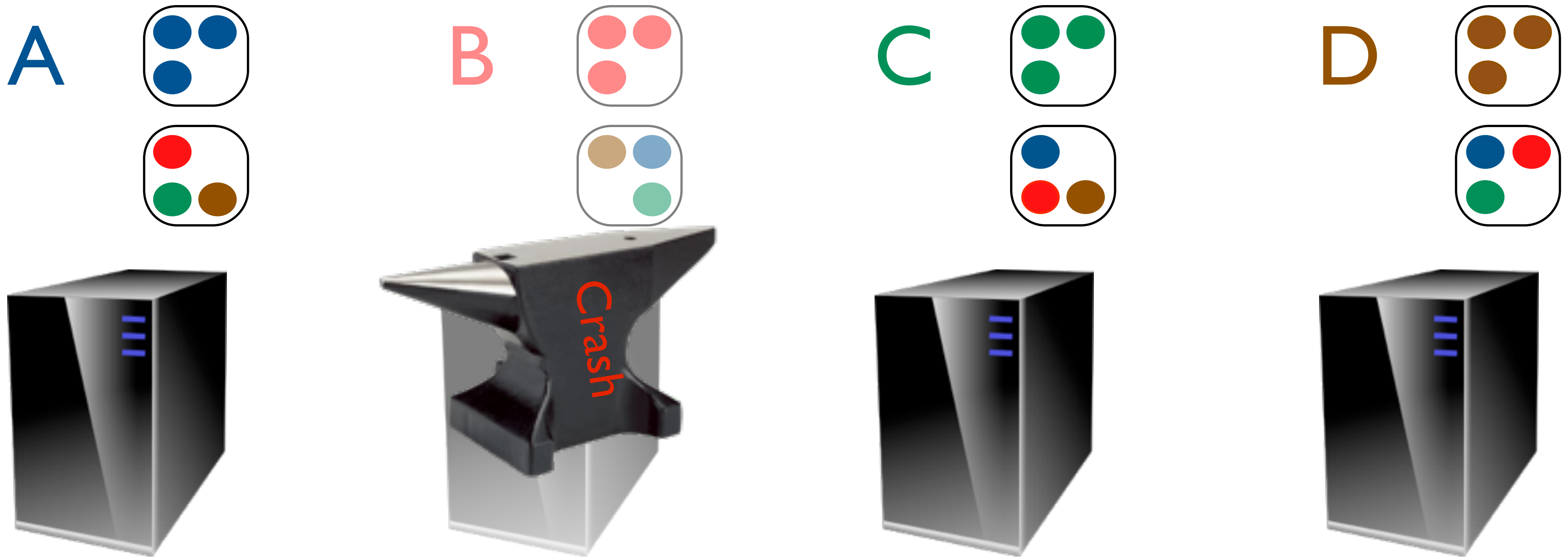


Migration Complete

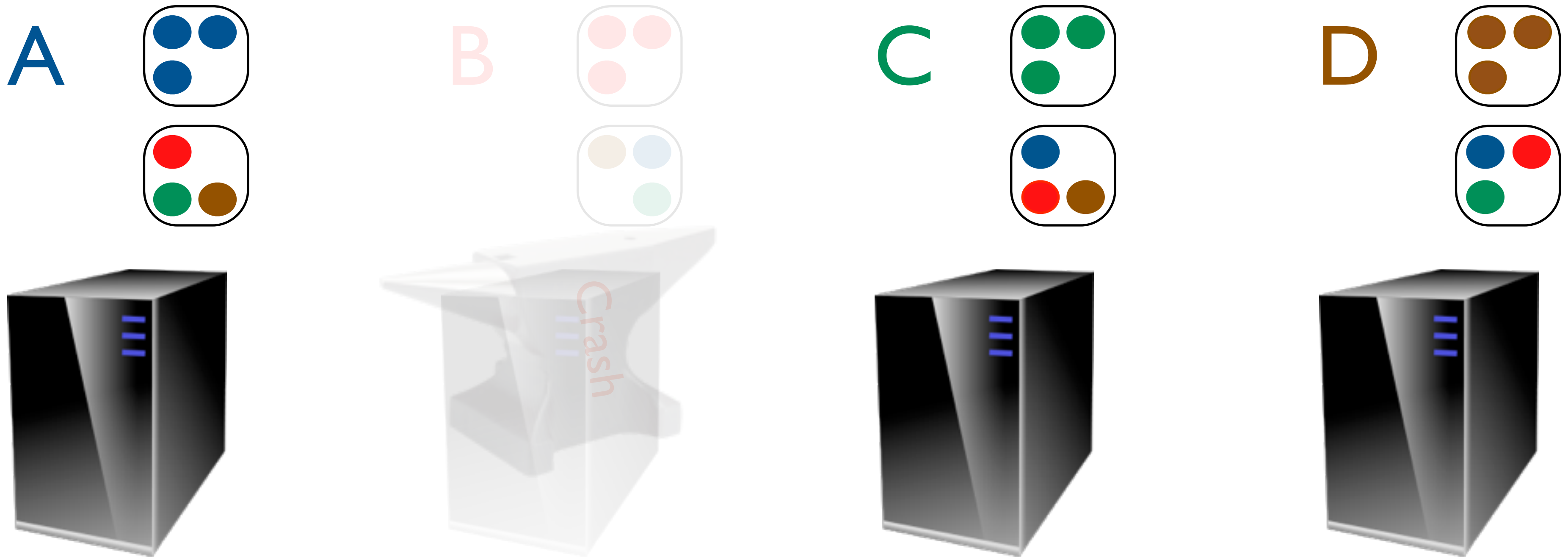


Data Safety When Node Dies

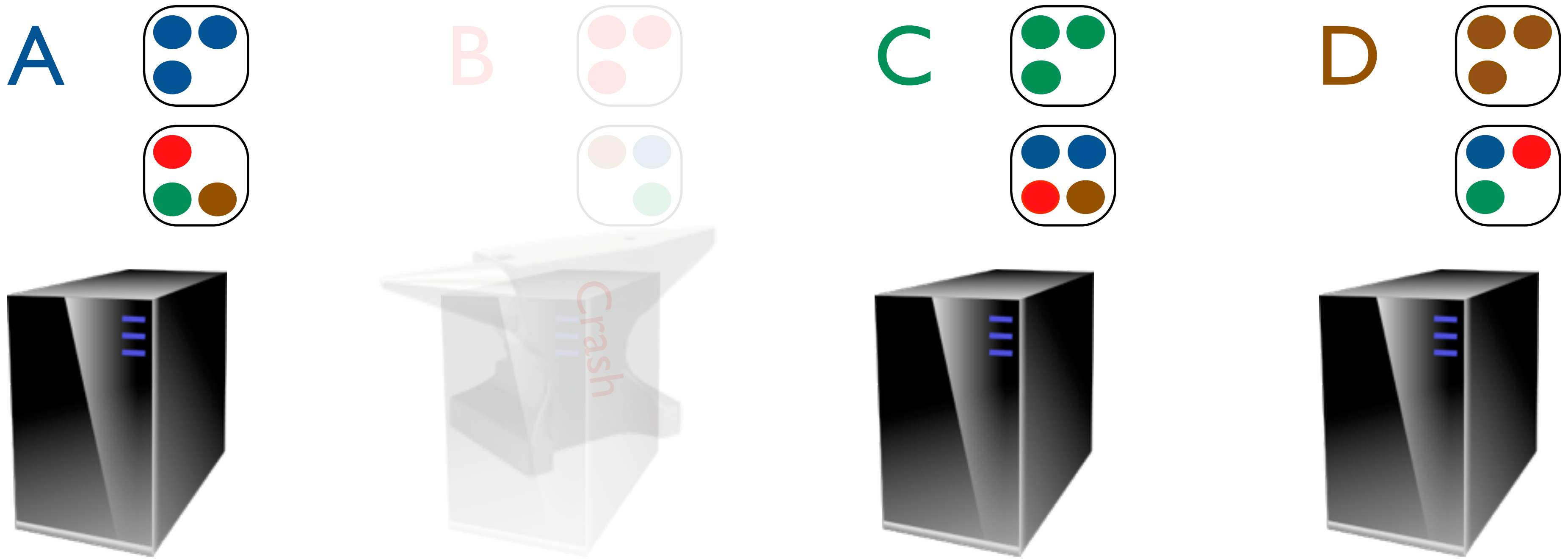
Node Crashes



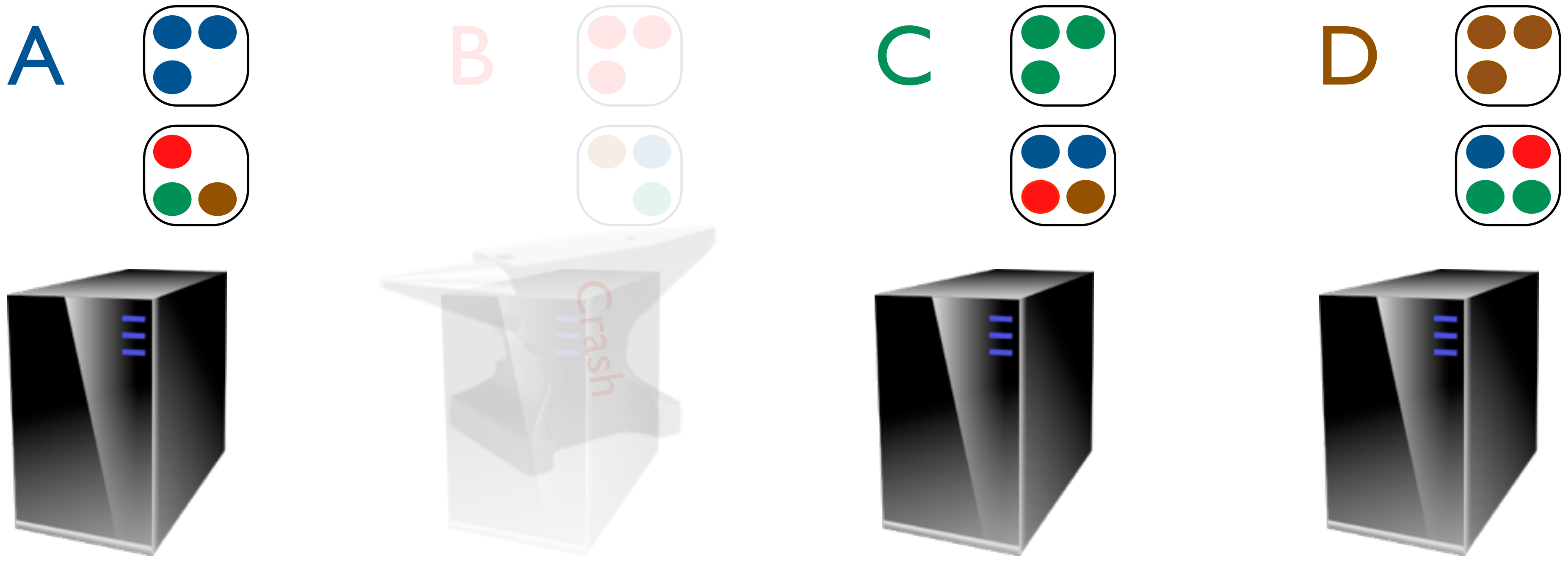
Backups are Restored



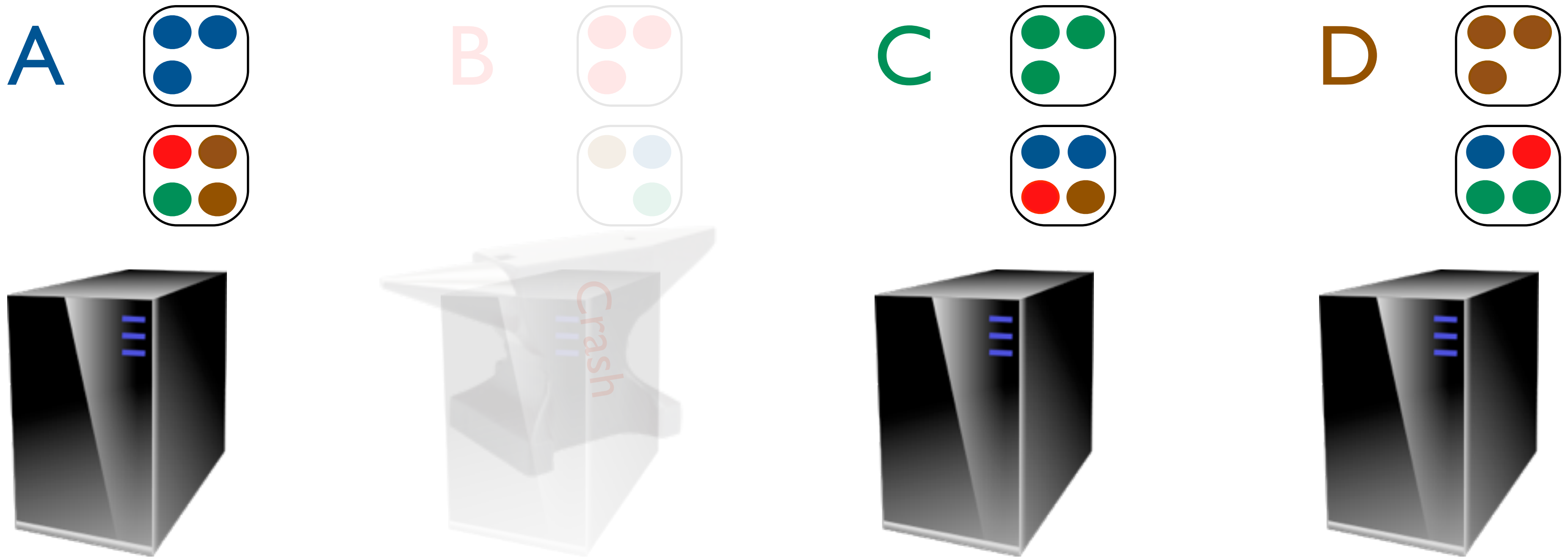
Backups are Restored



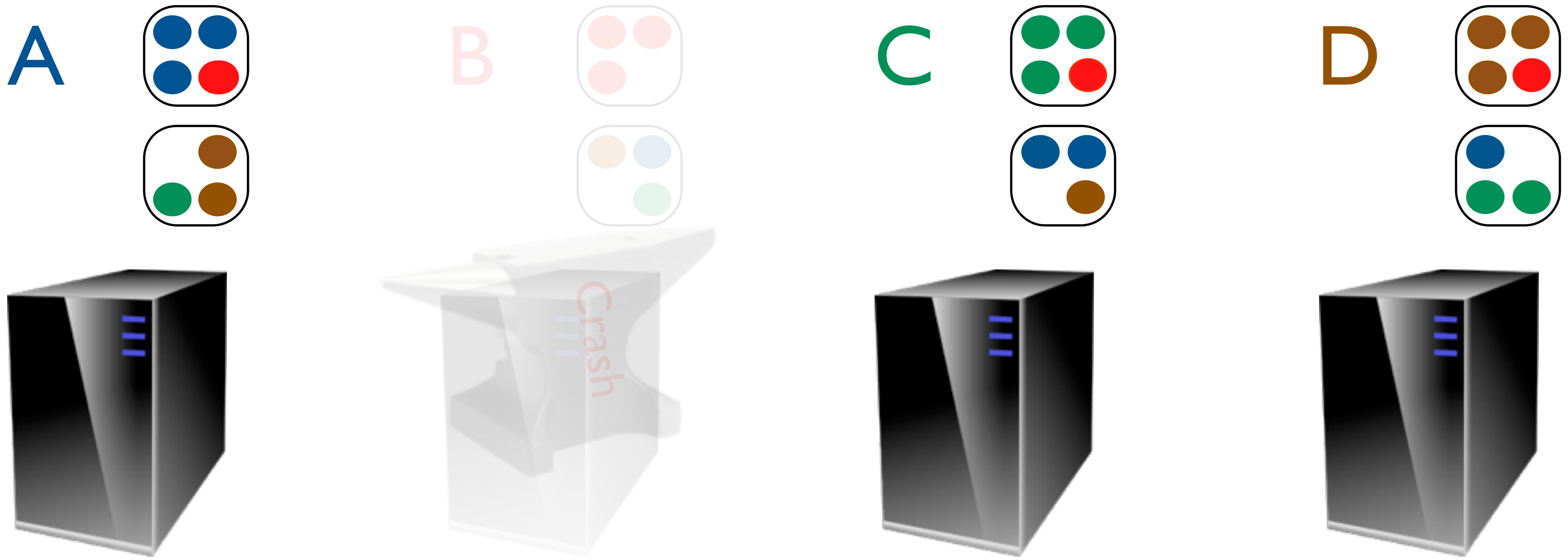
Backups are Restored



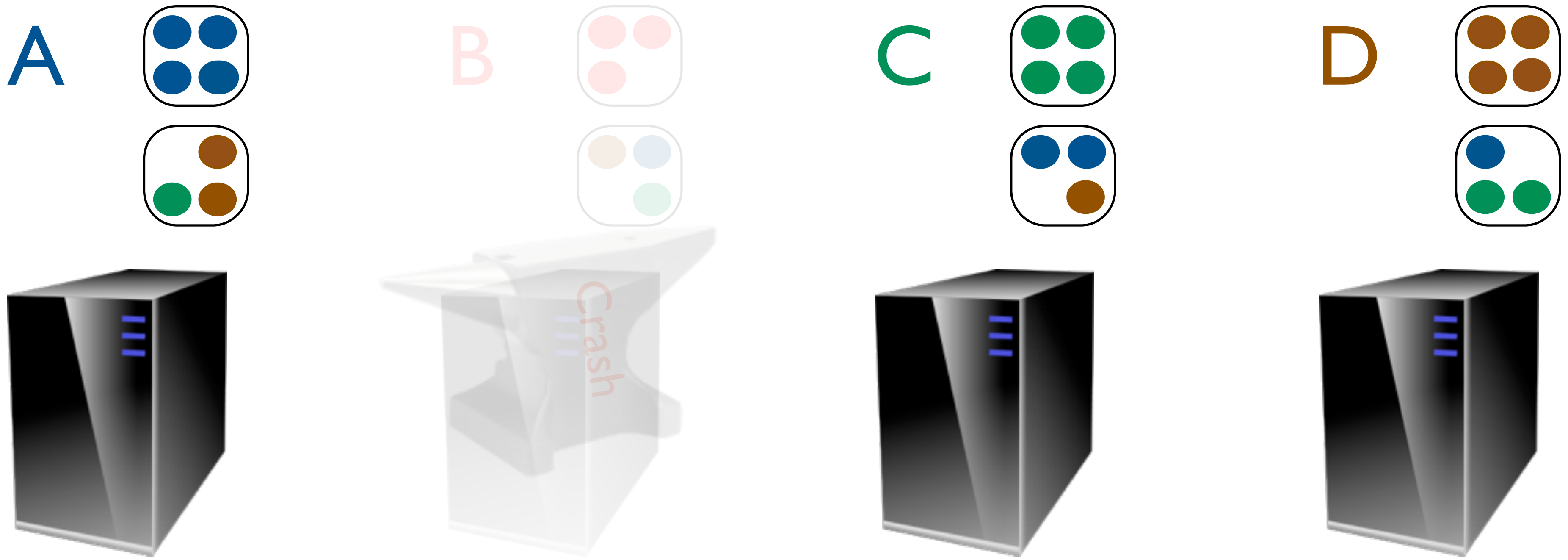
Backups are Restored



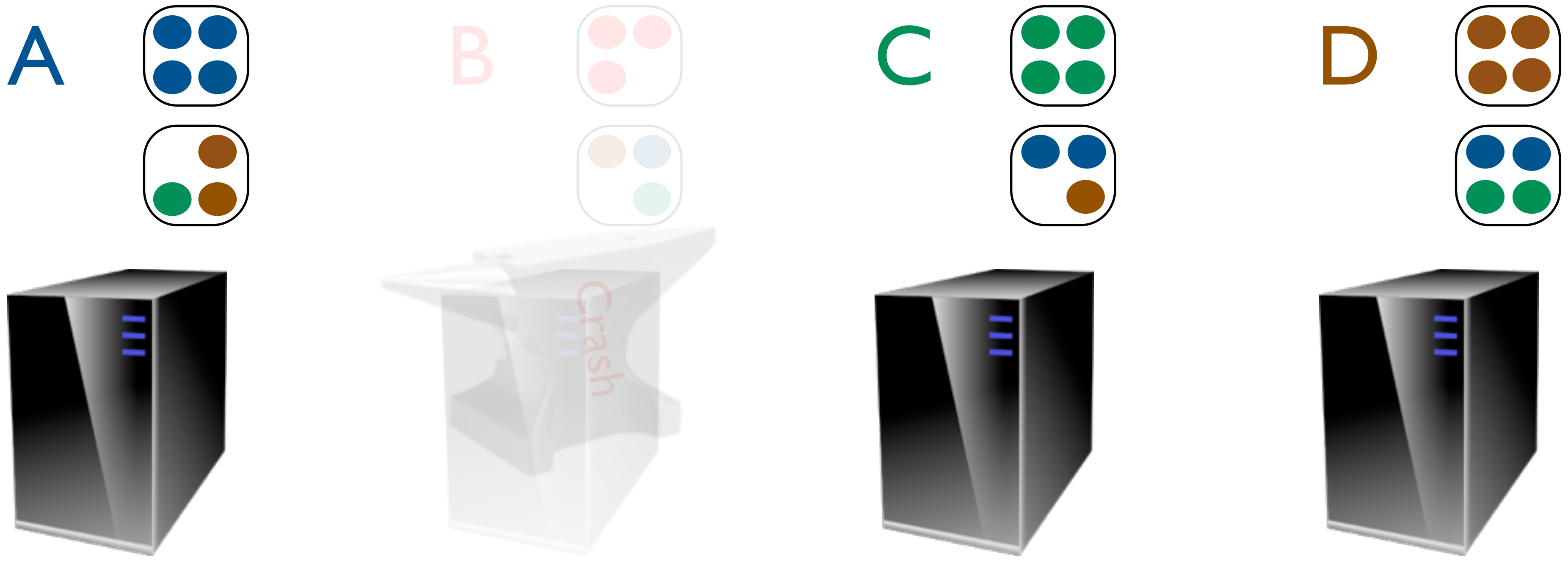
Data is Recovered from backup



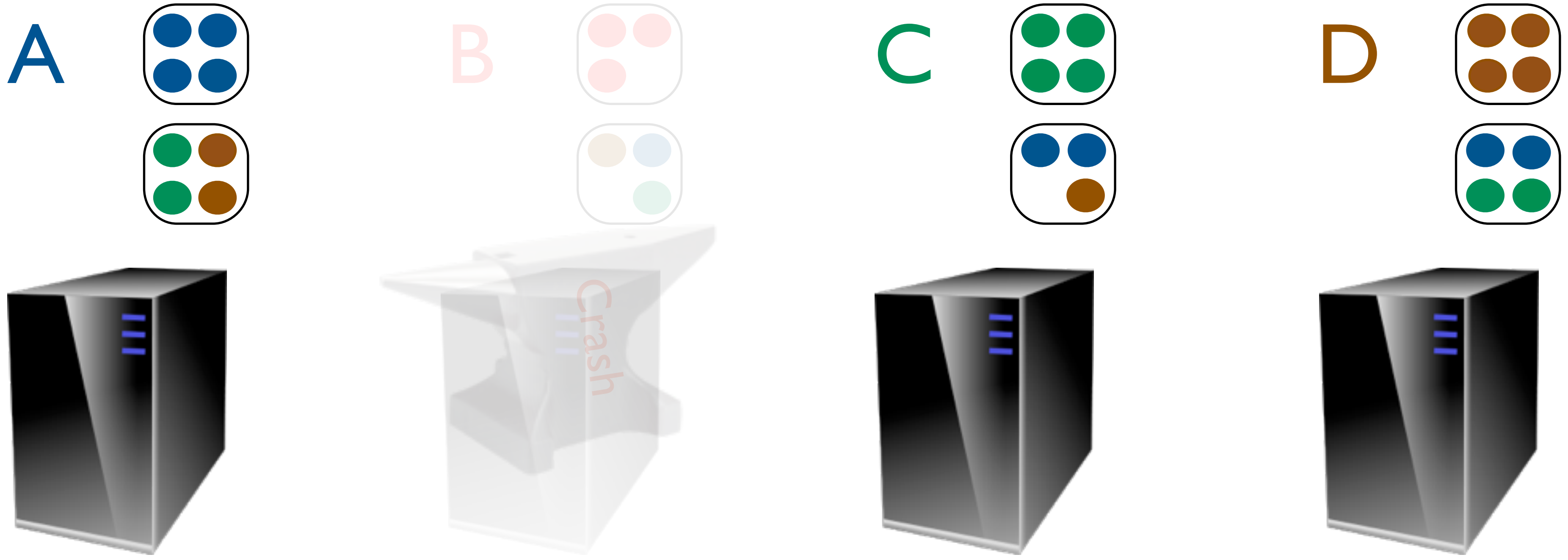
Data is Recovered from backup



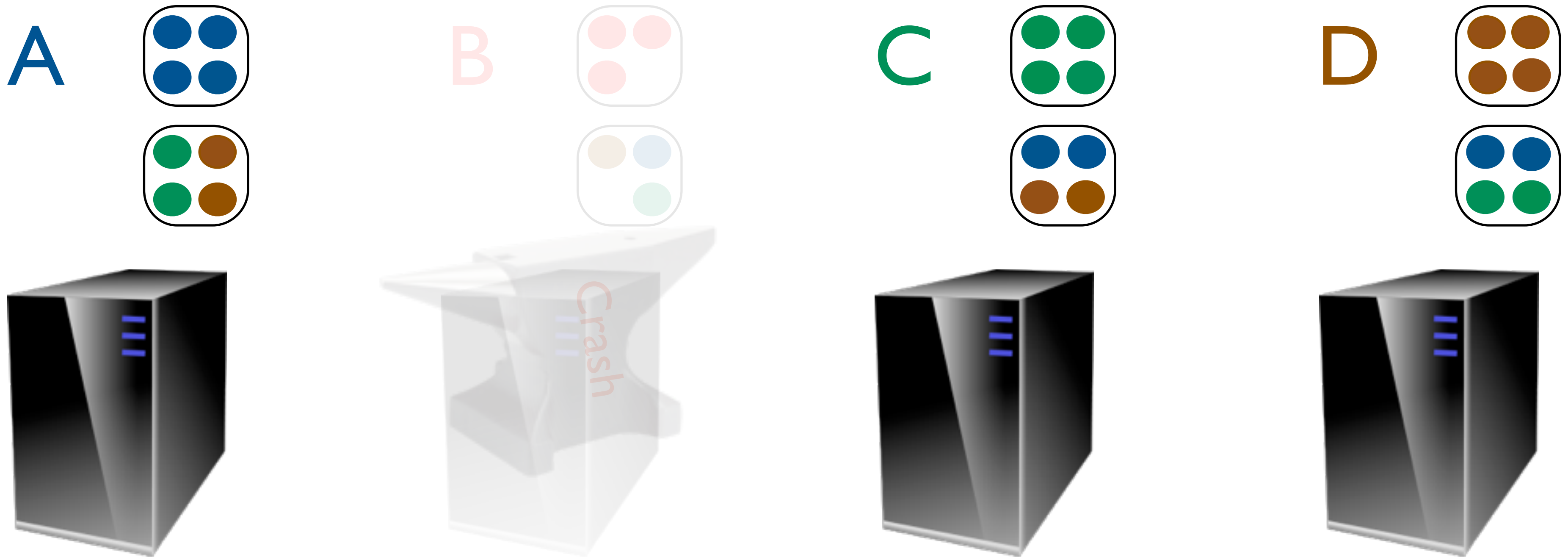
Backup for Recovered Data



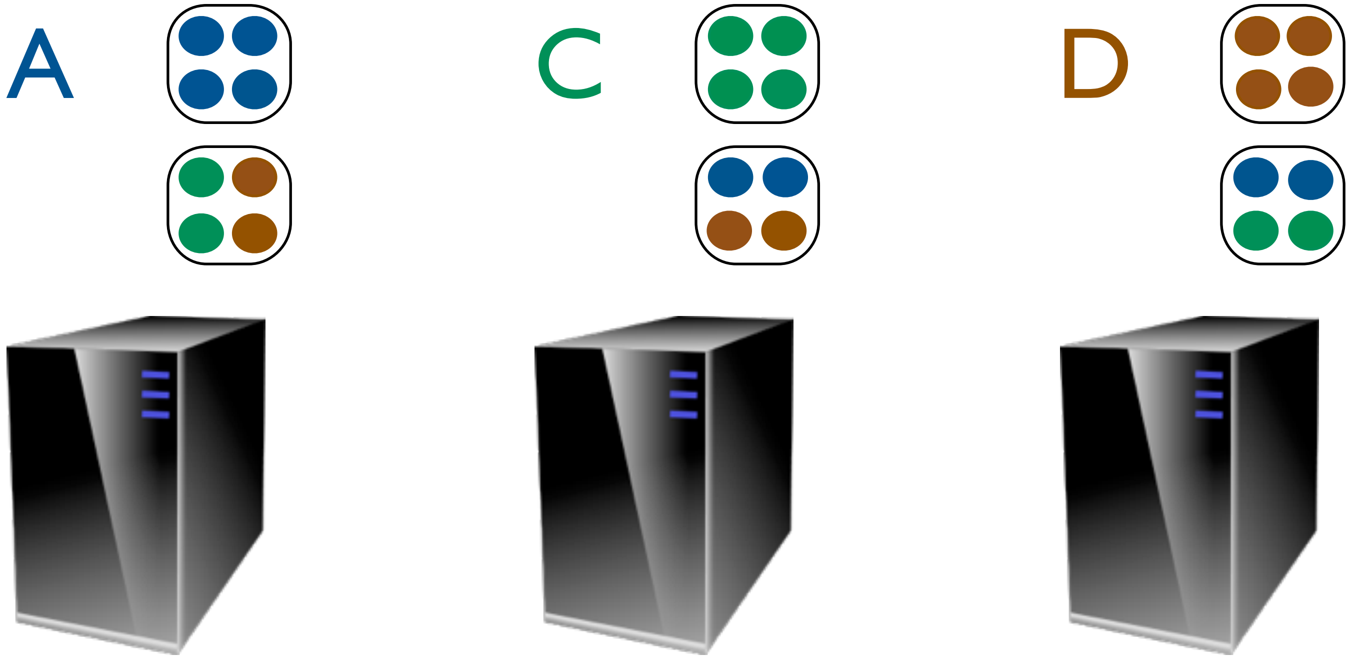
Backup for Recovered Data



Backup for Recovered Data



All Safe



Scalability: Up or Out?

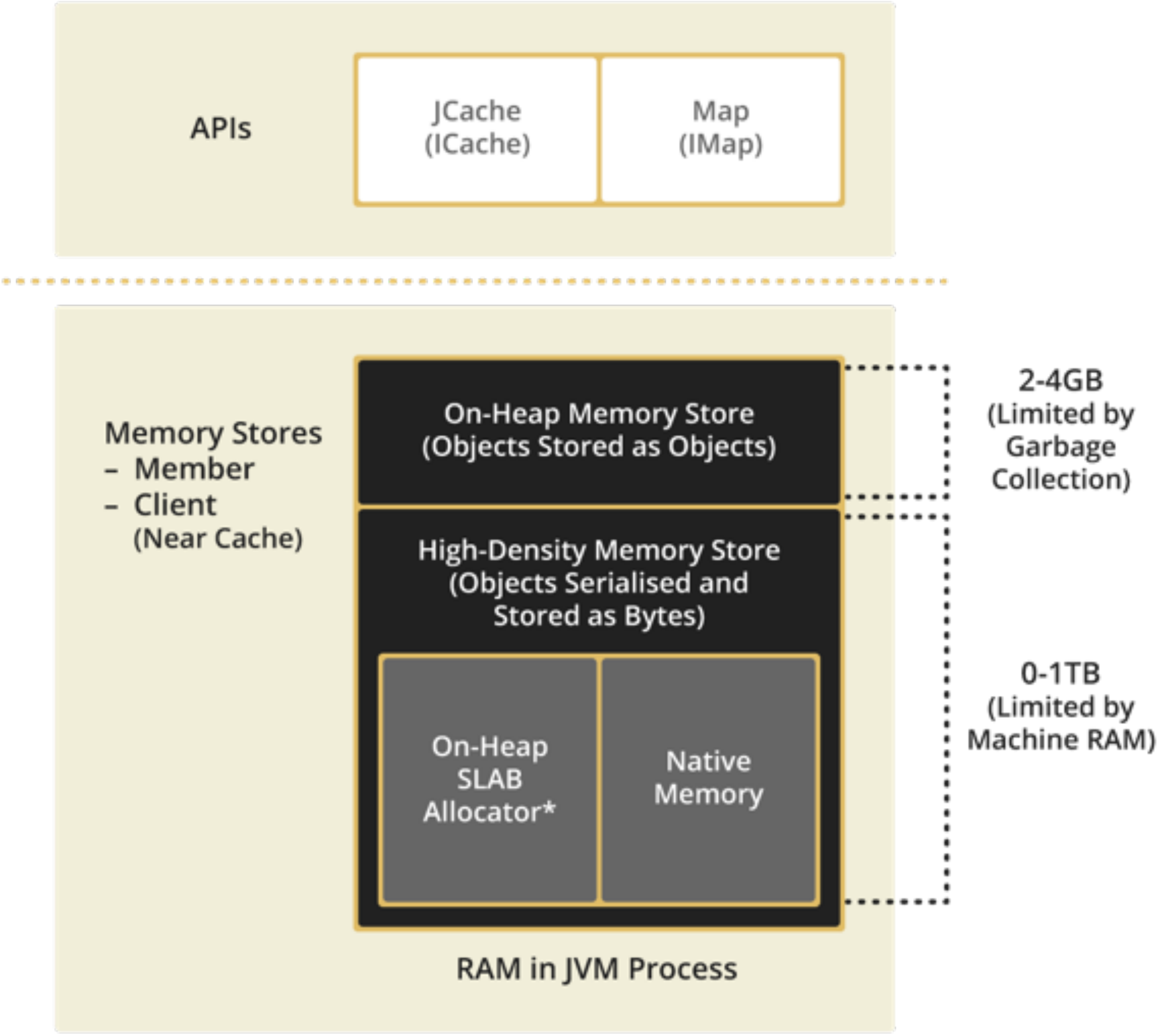
Scale Out

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

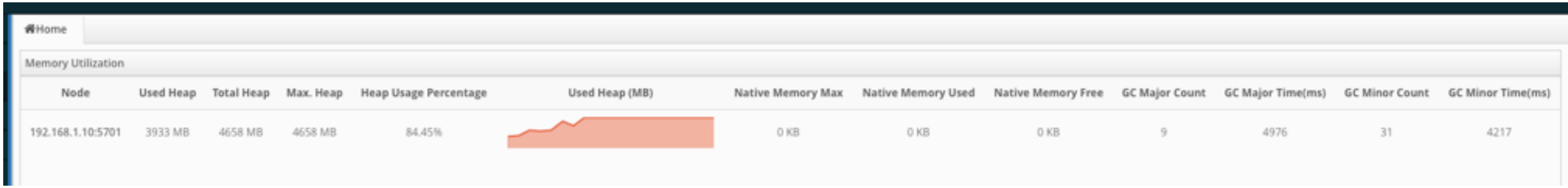
High Density Caching



* coming in 3.6

HD vs Heap

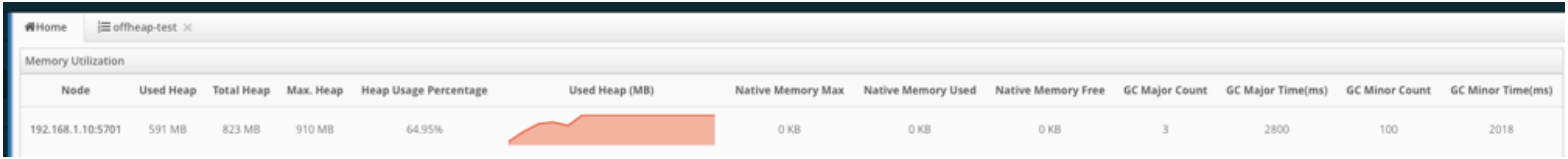
Heap: 0 MB Native, 4000 MB Heap, 9 Major GC (4900 ms), 31 Minor GC (4217 ms)



Memory Utilization

Node	Used Heap	Total Heap	Max. Heap	Heap Usage Percentage	Used Heap (MB)	Native Memory Max	Native Memory Used	Native Memory Free	GC Major Count	GC Major Time(ms)	GC Minor Count	GC Minor Time(ms)
192.168.1.10:5701	3933 MB	4658 MB	4658 MB	84.45%		0 KB	0 KB	0 KB	9	4976	31	4217

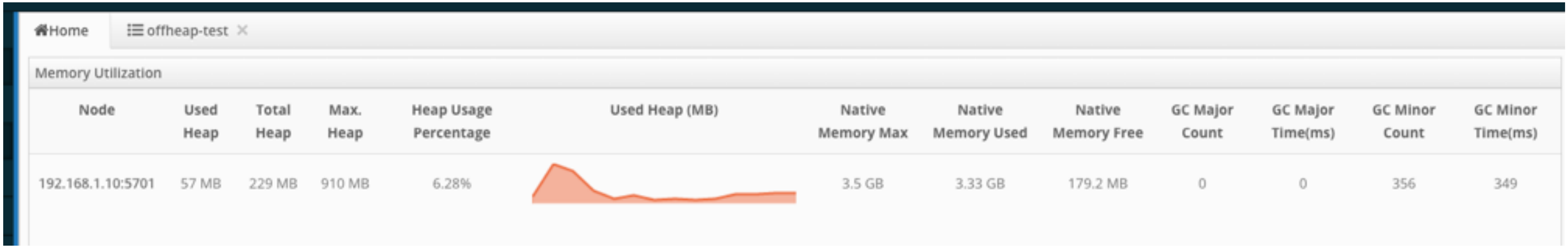
HD Memory v1:3000 MB Native, 600 MB Heap, 3 Major GC (2800 ms), 100 Minor GC (2018 ms)



Memory Utilization

Node	Used Heap	Total Heap	Max. Heap	Heap Usage Percentage	Used Heap (MB)	Native Memory Max	Native Memory Used	Native Memory Free	GC Major Count	GC Major Time(ms)	GC Minor Count	GC Minor Time(ms)
192.168.1.10:5701	591 MB	823 MB	910 MB	64.95%		0 KB	0 KB	0 KB	3	2800	100	2018

HD Memory v2:3300 MB Native, 57 MB Heap, 0 Major GC (0 ms), 356 Minor GC (349 ms)



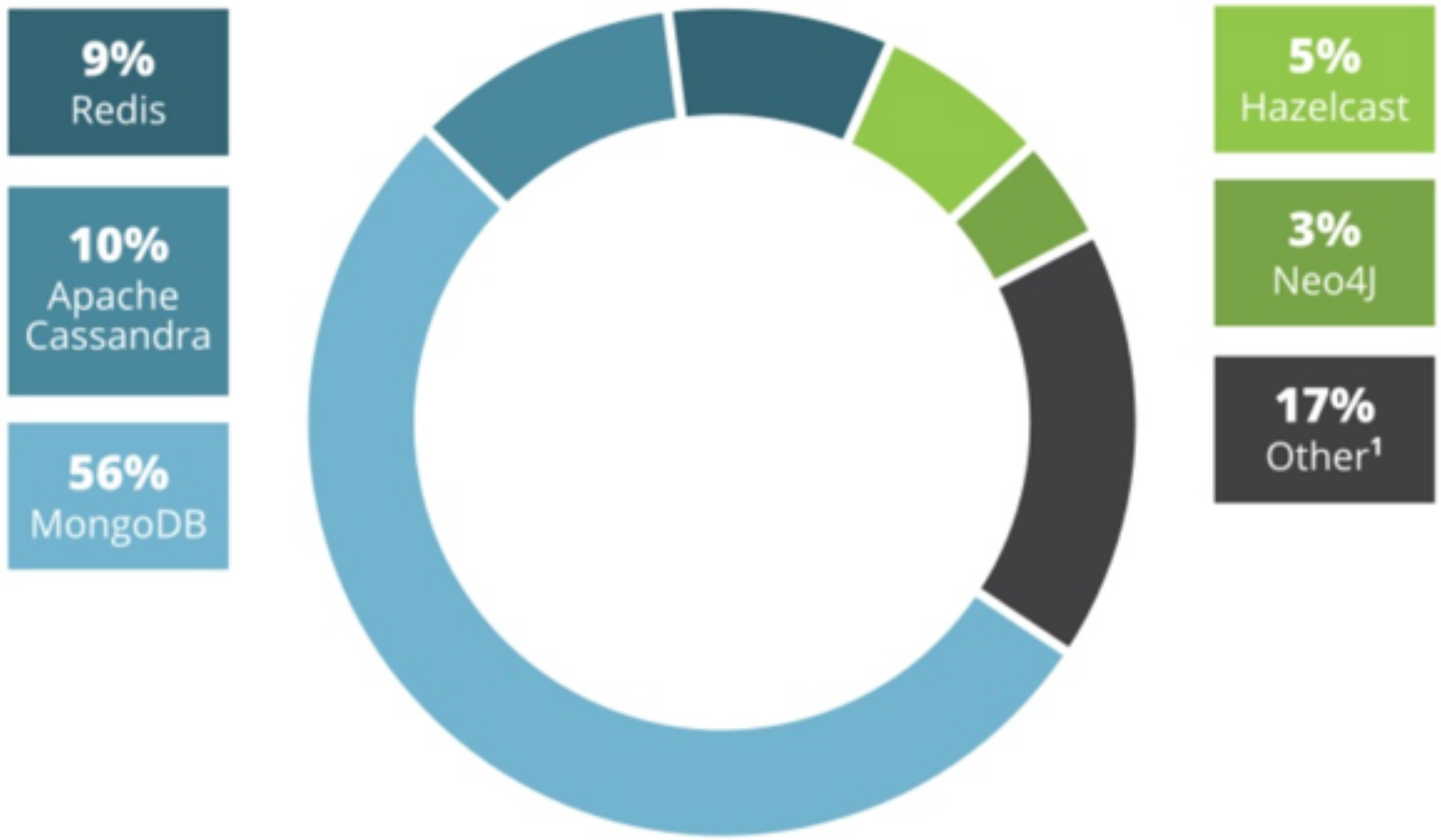
Memory Utilization

Node	Used Heap	Total Heap	Max. Heap	Heap Usage Percentage	Used Heap (MB)	Native Memory Max	Native Memory Used	Native Memory Free	GC Major Count	GC Major Time(ms)	GC Minor Count	GC Minor Time(ms)
192.168.1.10:5701	57 MB	229 MB	910 MB	6.28%		3.5 GB	3.33 GB	179.2 MB	0	0	356	349

Do I really need NoSQL?

NoSQL Landscape

Primary NoSQL technology*



REBELLABS

The results were normalized to exclude non-users

[1] *Including Memcache, Riak and a dozen others*

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

QA

Bonus!

Go to

http://hazelcast.org/t-shirts/

enter **hazelcastic**

and get your t-shirt!

Question, please?