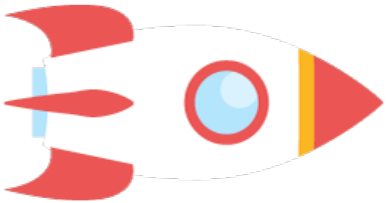
恰如其分的 MySQL 設計技巧

Ant
yftzeng@gmail.com
2016-08-24



Modern Web 2016

Profile

程式開發 X 資訊安全 X 智慧財產權 X 創業

恰 如 其 分

qià rú qí fèn

剛好符合分寸。形容做事、說話十分恰當。

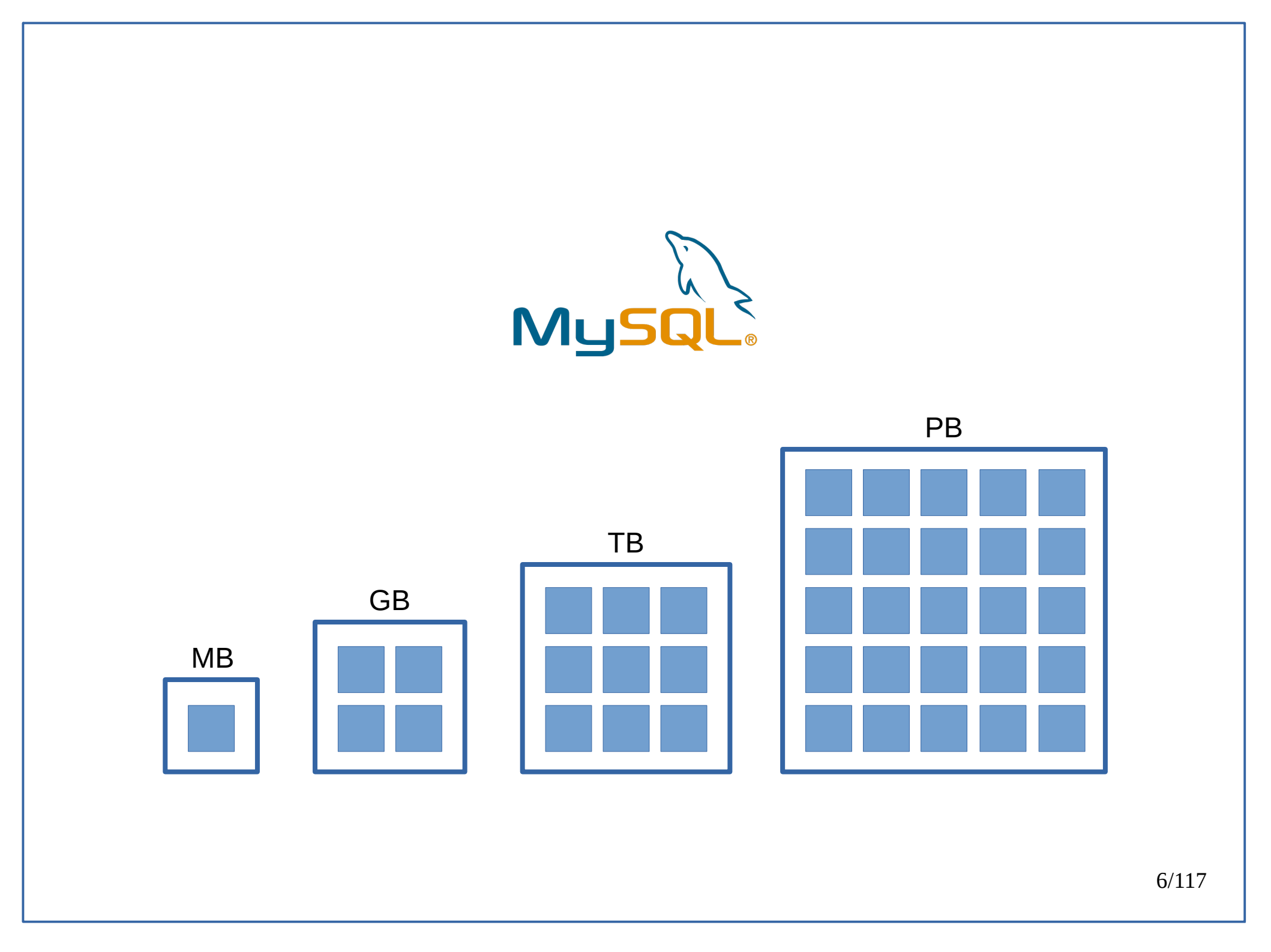
Premature optimization is the root of all evil

過早最佳化是萬惡的根源

- Donald Knuth -

架構是演進的，預想有益身心健康

預先策想


MySQL®
PB
TB
GB
MB

新平台上線

新平台預計下一季上線。

營運預計導入多少流量?
我可以先行準備。

有流量再說，快就好。

那我之後 (有空) 再做。

上線一週後

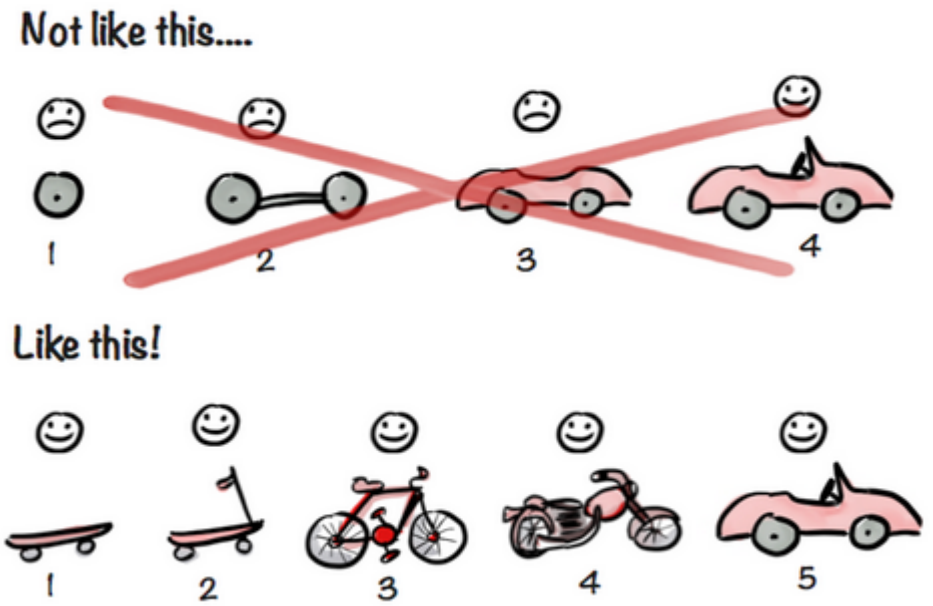
為什麼網站很慢又常當機?

調整架構需要一週。

這麼久? 不能快點嗎?

沒有辦法 ...

Minimum Viable Product



MVP 若沒有控制好，技術債將迅速增長

先前程式常不能用

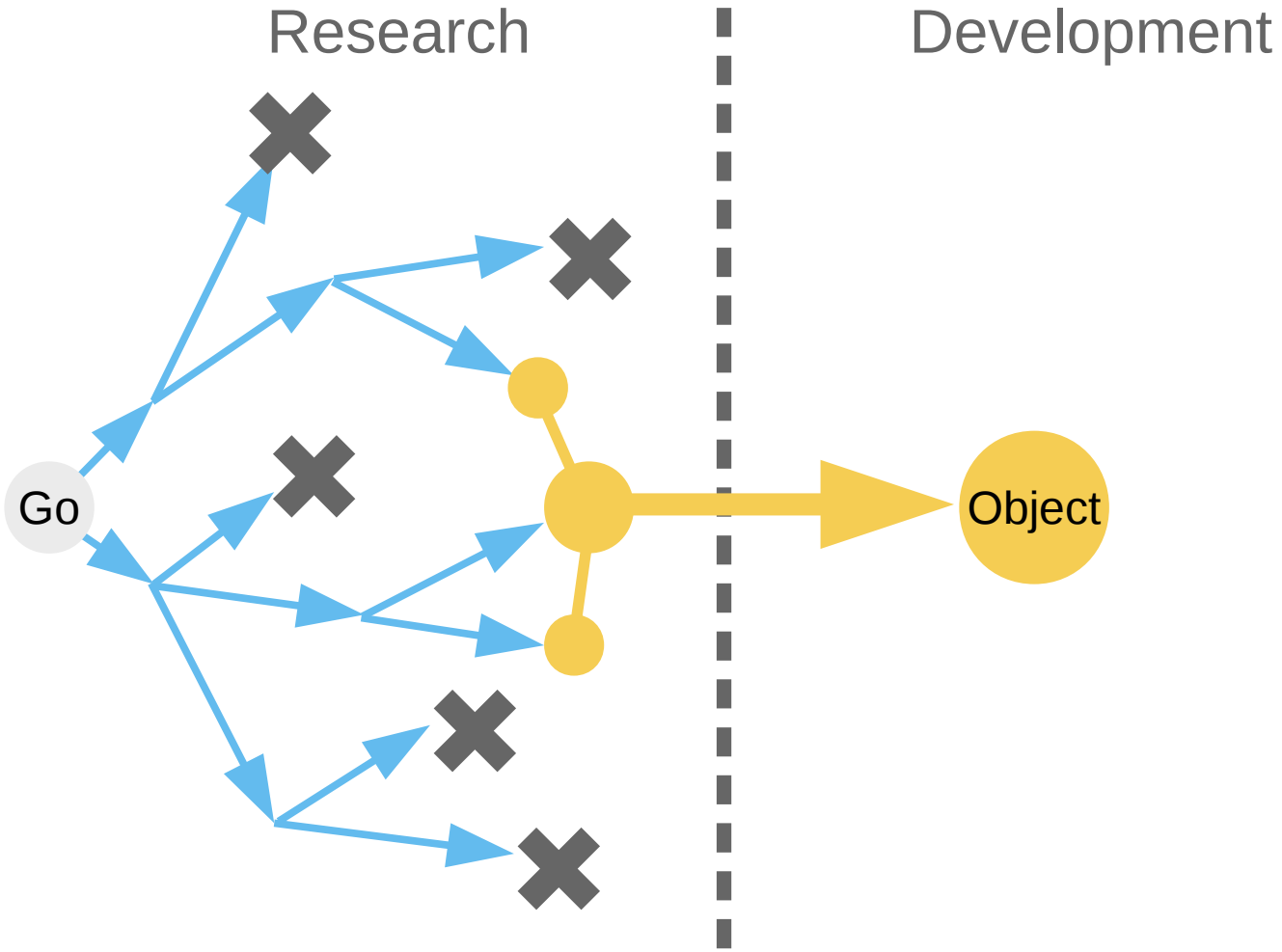
遺舊程式不忍棄之

結構變動反覆無常

只做最低限度產品 (MVP)

最終得到的可能是壞產品

Pivot



❶ 架構先決

無視人員、流程只講技術，是要白痴

架構會影響公司文化、商業擴展；思維更要超越程式碼層次

❷ 沒有完美的架構，只有最適的架構

無視場景只講架構，是耍流氓

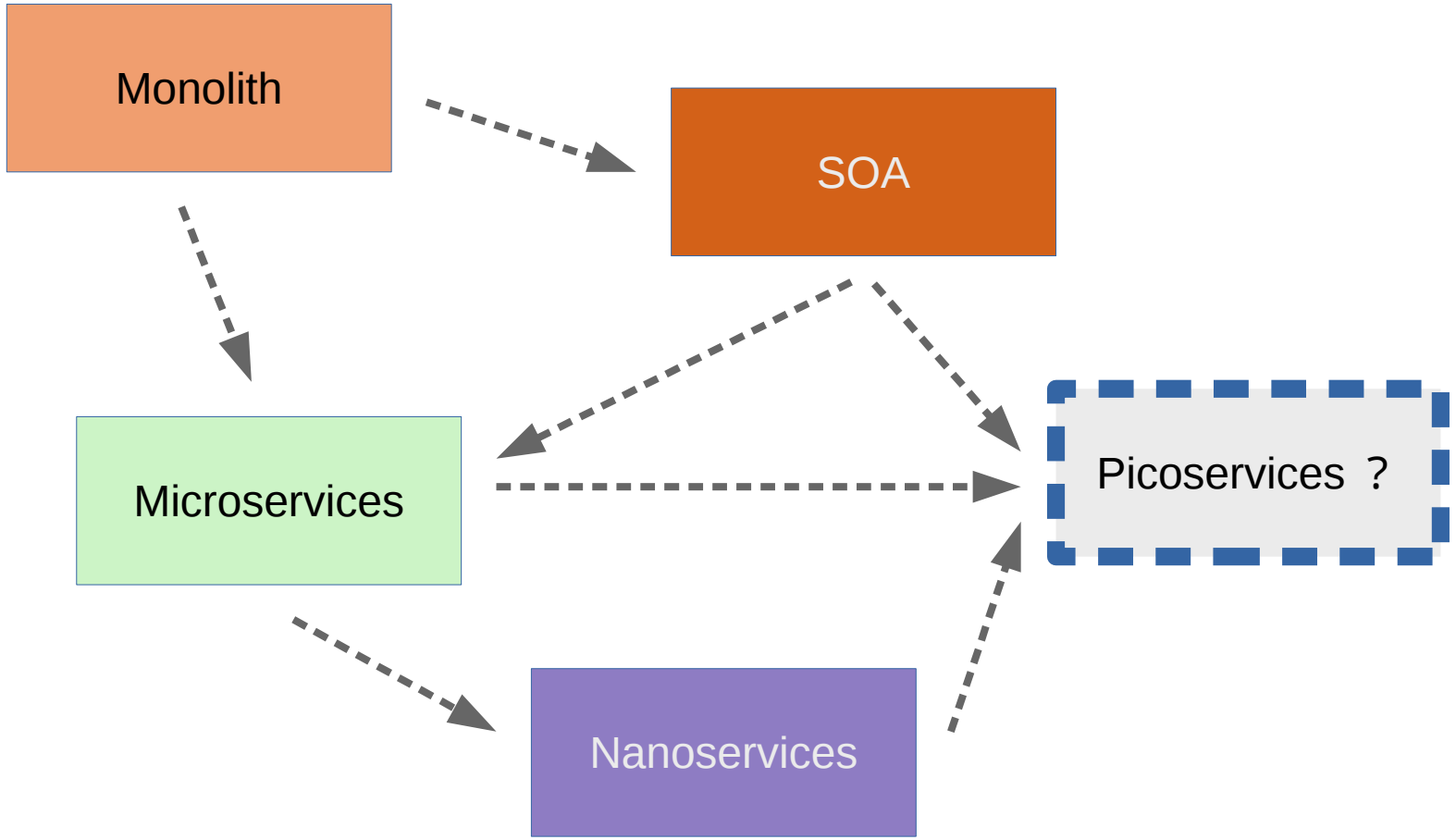
若無法舉出架構的缺陷，代表你還無法掌握

盲目套用別人的架構，並不會讓你變得跟他一樣好

❸ 架構是演進的，預想但不過早調優

無視未來只求現有，是耍自閉

兵馬未動，糧草先行，預想下一步，下下一步，甚至下下下一步 ...


Monolith
SOA
Microservices
Picoservices ?
Nanoservices

Architecture

Business	Technology
License Elastic business Workload	Scale-up Application Connection Database File system OS Kernel Hardware Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE

Architecture

Business	Technology
License	Scale-up Application Connection Database File system OS Kernel Hardware
Elastic business	Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency
Workload	

and more ...

CONSILIENCE

License


MySQL®


PERCONA
Performance Consulting Experts


MariaDB


GPL

License

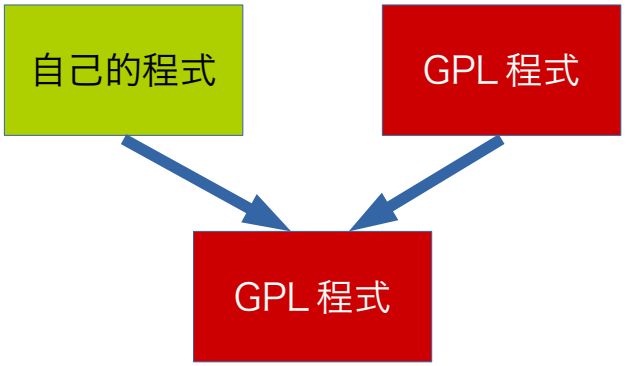
程式原始碼是否在意授權對方修改、散布?

GNU GPL 的互惠性 / 感染性

散布于對方時，強制啟動授權

(接案) 軟體交付時

嵌入式設備

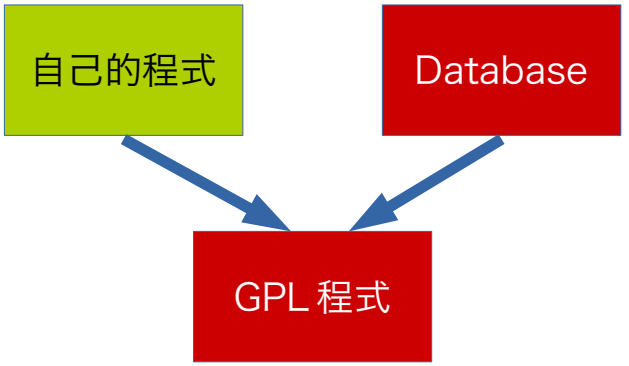


License

C/Java + MySQL/Percona →

PHP/PDO + MySQL/Percona →

C/Java/PHP + MariaDB →



Architecture

Business	Technology
License	Scale-up Application Connection Database File system OS Kernel Hardware
Elastic business	
Workload	Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE

Elastic business

當業務需求變更程式設計時

預想但不過早調優

總工程師不該把每次新需求都認為獨立需求
(多想二分鐘，團隊可以不必自虐)

Elastic business

狀態

原表格設計

id	name	...	is_deleted
1	Apple	...	0
2	Banana	...	1

Elastic business

狀態

新業務需要儲存「鎖定」狀態

id	name	...	is_deleted
1	Apple	...	0
2	Banana	...	1

id	name	...	is_deleted	is_locked
1	Apple	...	0	1
2	Banana	...	1	0

Elastic business

狀態

其實若狀態是互斥的，則可以合併

id	name	...	is_deleted	is_locked
1	Apple	...	0	1
2	Banana	...	1	0

id	name	...	status
1	Apple	...	0
2	Banana	...	1

{ 0: deleted, 1: enabled, 2: locked }

Elastic business

標籤雲

原表格設計

id	name	tag1
1	Apple	admin
2	Banana	reporter
3	Cherry	reporter

```
SELECT * FROM {Table}
WHERE tag1 = ‘admin’
```

Elastic business

標籤雲

新增標籤

id	name	tag1	tag2	tag3
1	Apple	admin	reporter	programmer
2	Banana	reporter	programmer	NULL
3	Cherry	reporter	admin	NULL

ALTER TABLE !!

```
SELECT * FROM {Table}
WHERE (tag1 = ‘admin’ OR tag2 = ‘admin’ OR tag3 = ‘admin’)
  AND (tag1 = ‘reporter’ OR tag2 = ‘reporter’ OR tag3 = ‘reporter’)
```

```
SELECT * FROM {Table}
WHERE ‘admin’ IN (tag1, tag2, tag3)
  AND ‘reporter’ IN (tag1, tag2, tag3)
```

Elastic business

標籤雲

新方法

id	name	X	X	X
1	Apple	X	X	X
2	Banana	X	X	X

Tag

id	tag
1	admin
1	reporter
1	programmer
2	reporter
...	...

```
SELECT * FROM {Table}
  INNER JOIN ‘Tag’ AS t1 USING (id)
  INNER JOIN ‘Tag’ AS t2 USING (id)
WHERE t1.tag = ‘admin’
  AND t2.tag = ‘reporter’
```

Elastic business

標籤雲

或是 M:N

id	name
1	Apple
2	Banana

id	tag_id
1	1
1	2
1	3
2	2
2	3

tag_id	name
1	admin
2	reporter
3	programmer

Elastic business

廣告需求

第一天

營運有新的需求，受眾在一天內不要看到相同廣告。

瞭解，預計一個工作天。

Elastic business

廣告需求

第二天

營運有新的需求，受眾在小時內不要看到相同廣告。

時間粒度不同

瞭解，預計二個工作天。

Elastic business

廣告需求

第三天

營運有新的需求，受眾在一天內不要看到同類廣告。

目標粒度不同

瞭解，預計八個工作天。

不能一次講清楚嗎?

Elastic business

廣告需求

預想

受眾在 M 時間內不要看到 N 廣告

該需求的延伸會是什麼?

M → 年 / 季 / 月 / 週 / 時 / 分 / 秒

N → 相同 / 同類

看到的次數? 1/2/3...

裝置有別?

區域?

廣告主屬性?

Architecture

Business	Technology
License Elastic business Workload	Scale-up Application Connection Database File system OS Kernel Hardware Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE

Workload

Processing	Intensive	Capacity	Bond
OLTP OLAP Data warehouse	CPU intensive Memory intensive Storage/IO intensive Bandwidth intensive	Throughput Latency Memory footprint	Performance Security Cost restriction

Service-level agreement

Workload

Processing

OLTP (On-Line Transaction Processing)

❶ 應用 : Customer-oriented
❷ 回應時間 (response time) 要求較高。
❸ 併發 (concurrency) 要求較多。
❹ 資料處理量 (volume) 少。
❺ 交易 (transaction) 完整性高。
❻ 安全性 (security) 要求較高。

Workload

Processing

OLAP (On-Line Analytical Processing)

❶ 應用 : Market-oriented
❷ 回應時間 (response time) 要求較低。
❸ 併發 (concurrency) 要求較少。
❹ 資料處理量 (volume) 多。
❺ 交易 (transaction) 完整性低。
❻ 安全性 (security) 要求較低。

Workload

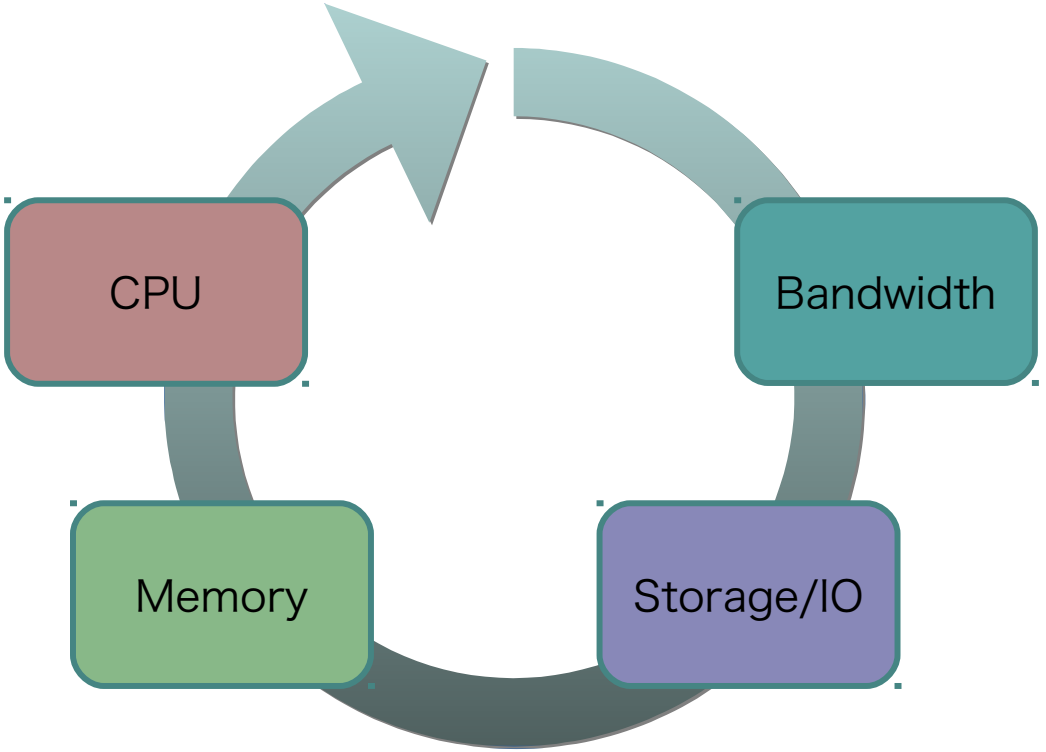
Processing

Data warehouse

❶ 應用 : Subject-oriented
❷ 熱資料 (Hot) 本地、快取、粒度、一致性。
❸ 暖資料 (Warm) 分布、快取、複製。
❹ 冷資料 (Cold) 索引、壓縮、合併、備份。

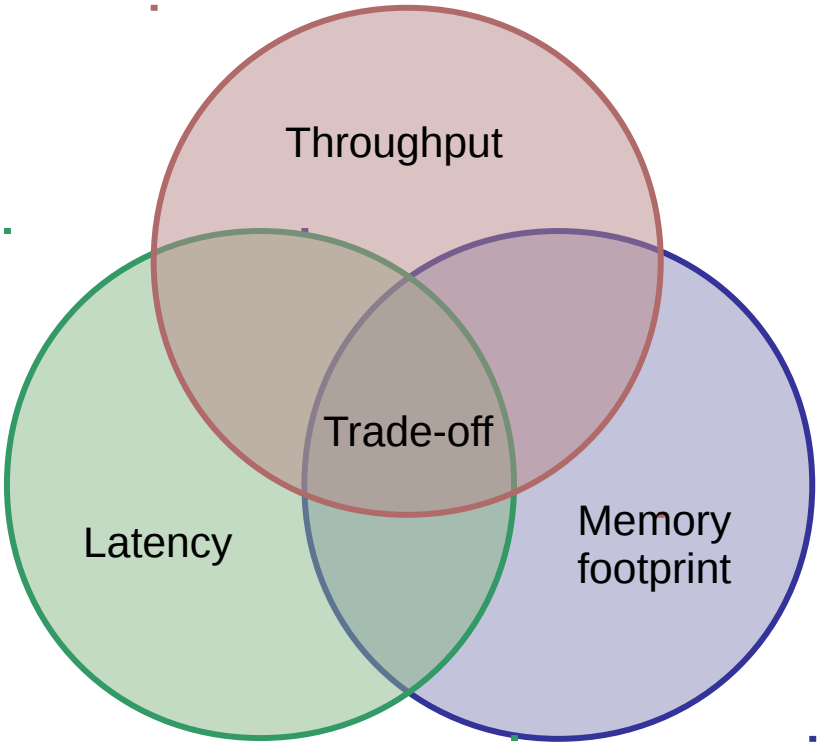
Workload

Intensive



Workload

Capacity



High throughput

Low latency

千萬人同時在線

電子商務、線上媒體

低延遲回應

廣告平台 (30ms)、高頻交易 (0.5~3ms)、醫療等關鍵設備

Workload

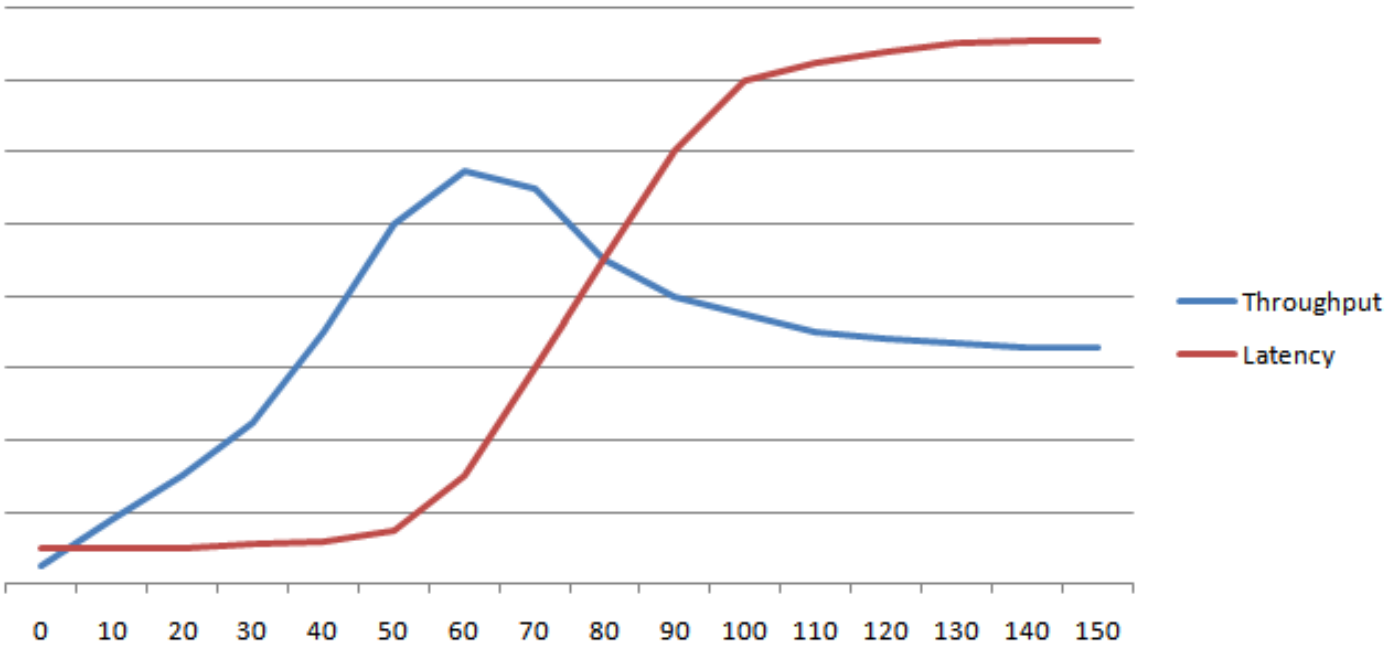
Capacity

機房只能給你 4U 共 2 台機器,
請您務必提供每秒 1 萬次的處理效能。

每次請求包括產生 46 張圖片及 46 次加密演算法

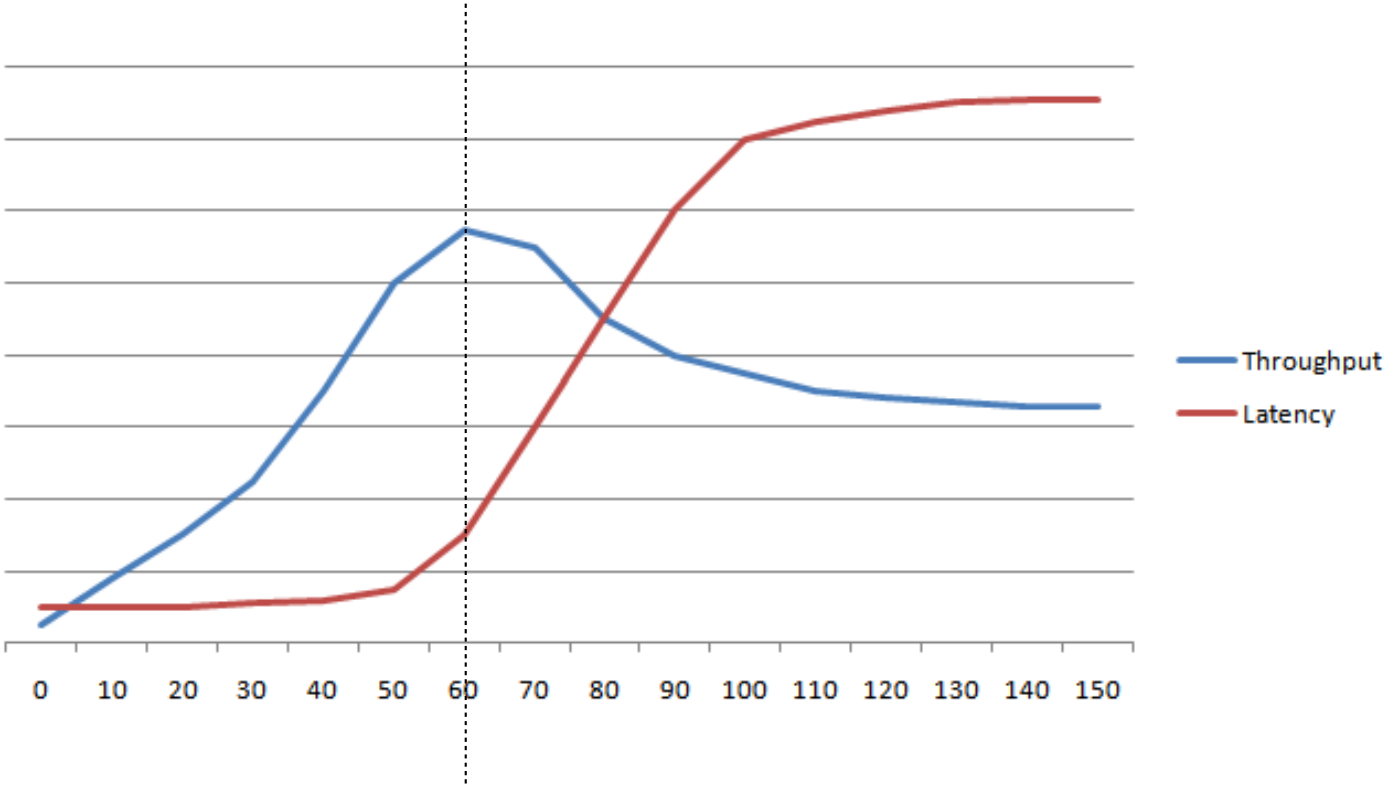
Workload

Capacity



Workload

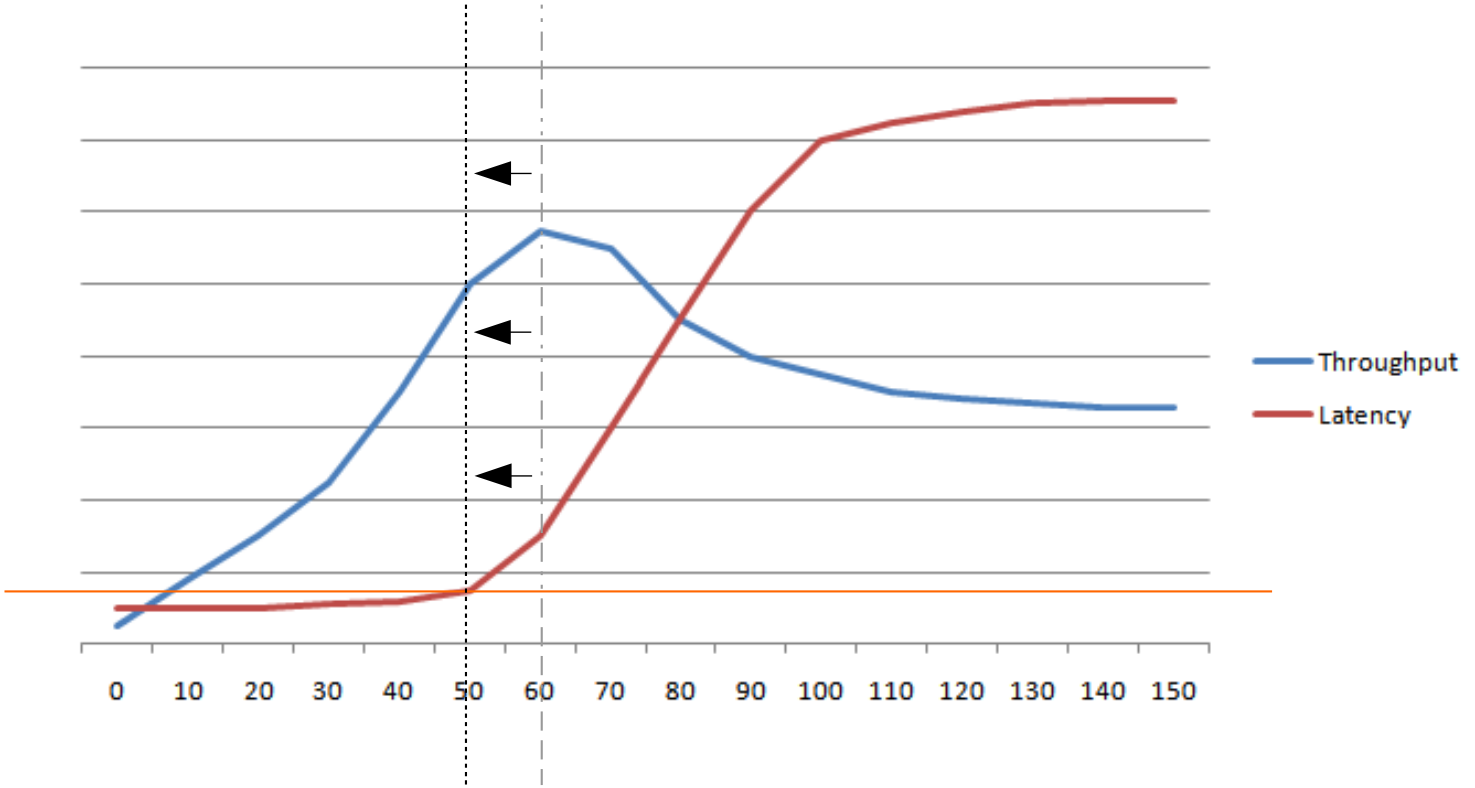
Capacity



Optimal capacity

Workload

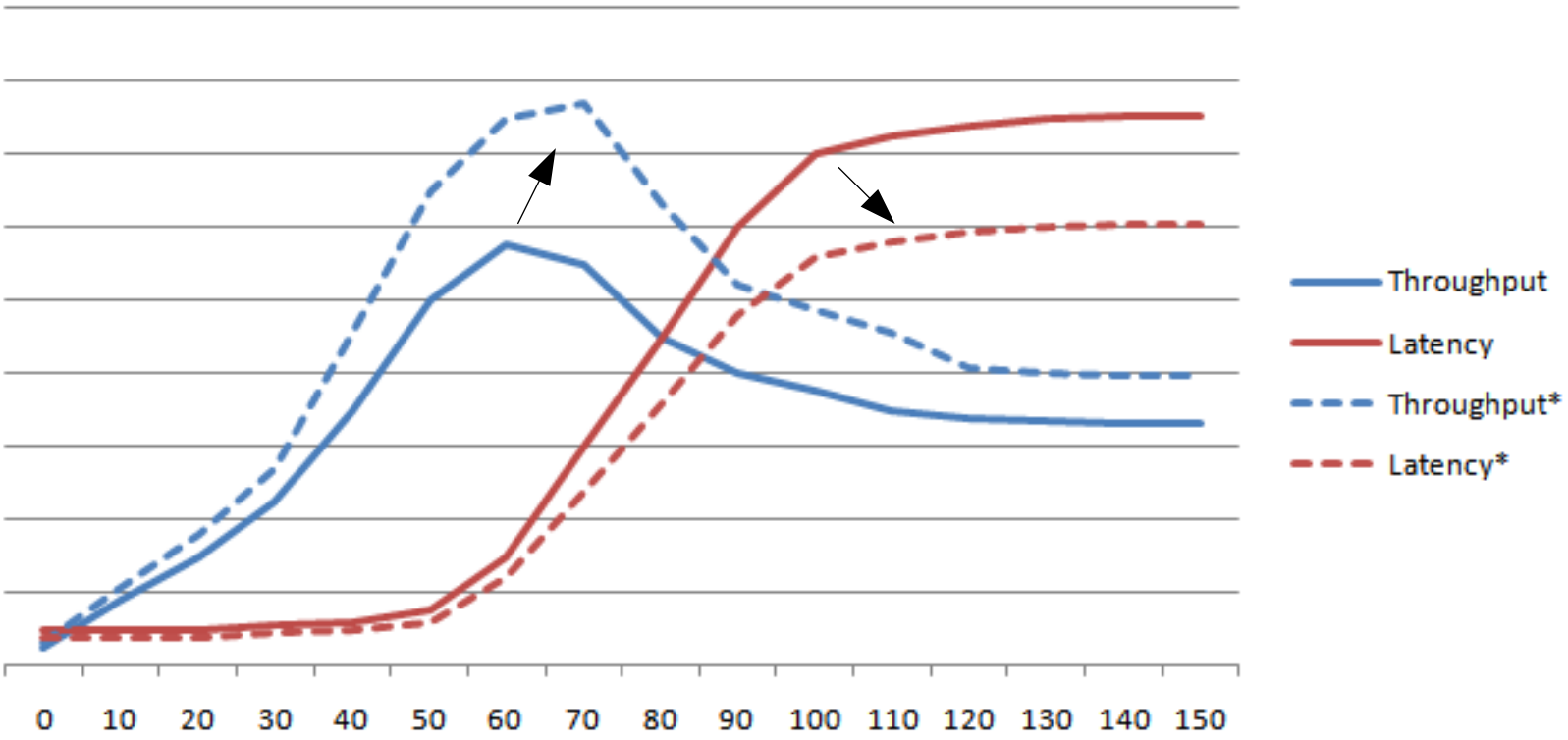
Capacity



Optimal capacity

Workload

Capacity

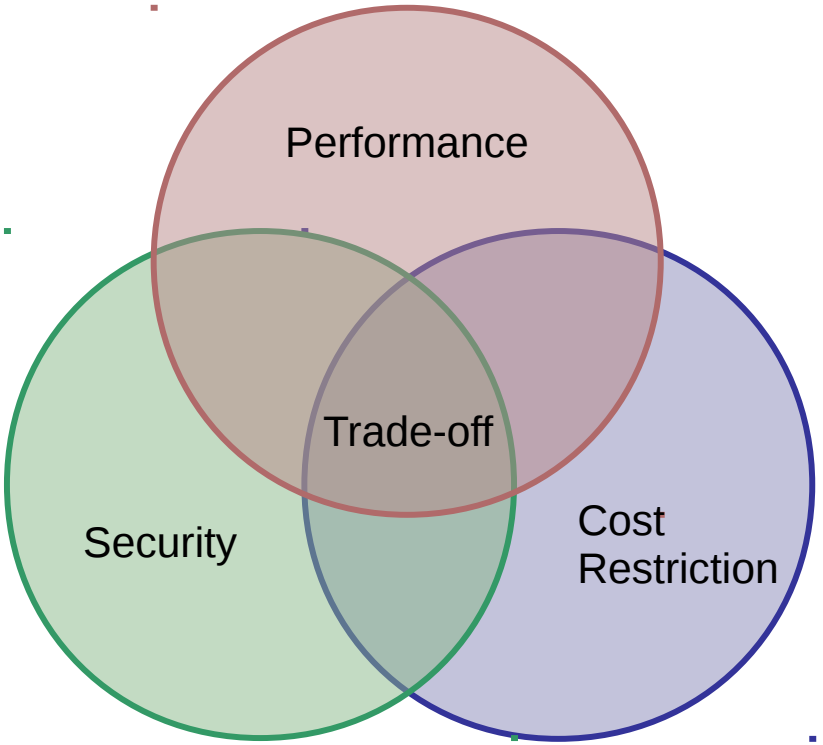


Language / Framework / Algorithm / Hardware

300 Server → Performance +10x → 30 Server
(Price cost reduction)
(Maintenance cost reduction)

Workload

Bond



Architecture

Business	Technology
License Elastic business Workload	Scale-up Application Connection Database File system OS Kernel Hardware Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE

Scale-up

Hardware

CPU

❶ 快取對 InnoDB 影響很大 (CPU Cache) 。
❷ 超執行緒 (Hyper threading) 有助益。
❸ 通常啟用 Node Interleaving ， 可避免 NUMA 問題。
❹ 多核心處理

MySQL 5.5
最佳表現為 16 核心，同時連線數 128 。

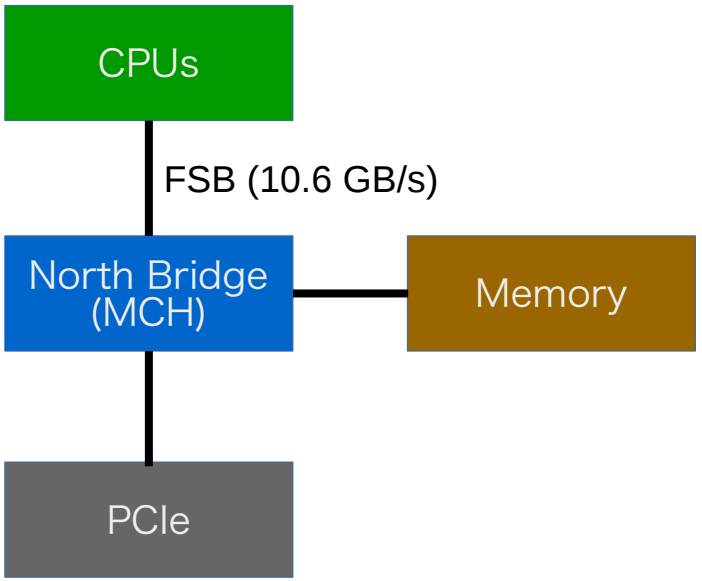
MySQL 5.6
支援至少 64 核心，同時連線數處理不受影響；
但 RW 在同時處理 128 連線數後顯著下降。

MySQL 5.7
支援至少 64 核心，同時連線數處理不受影響；
解決 RW 同時處理連線數下降問題。

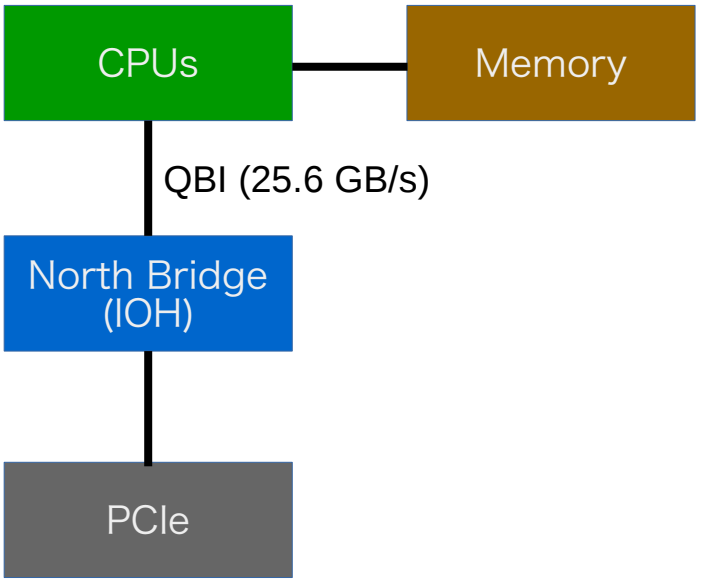
Scale-up

Hardware

Intel Xeon (Older)



Intel Xeon (Newer)



Scale-up

Hardware

Memory

❶ 原則上愈多愈好
❷ 確保能把所需資料表全儲存在記憶體中。

Scale-up

Hardware

Storage

❶ 原則上， PCIe NVMe SSD > SSD > HDD 。

	Capacity HDD	Enterprise HDD	Enterprise SSD
Read Bandwidth	60MB/s	90MB/s	200MB/s
Write Bandwidth	60MB/s	90MB/s	100MB/s
IOPS	70	320	35,000
Latency	15ms	2ms	<0.1ms

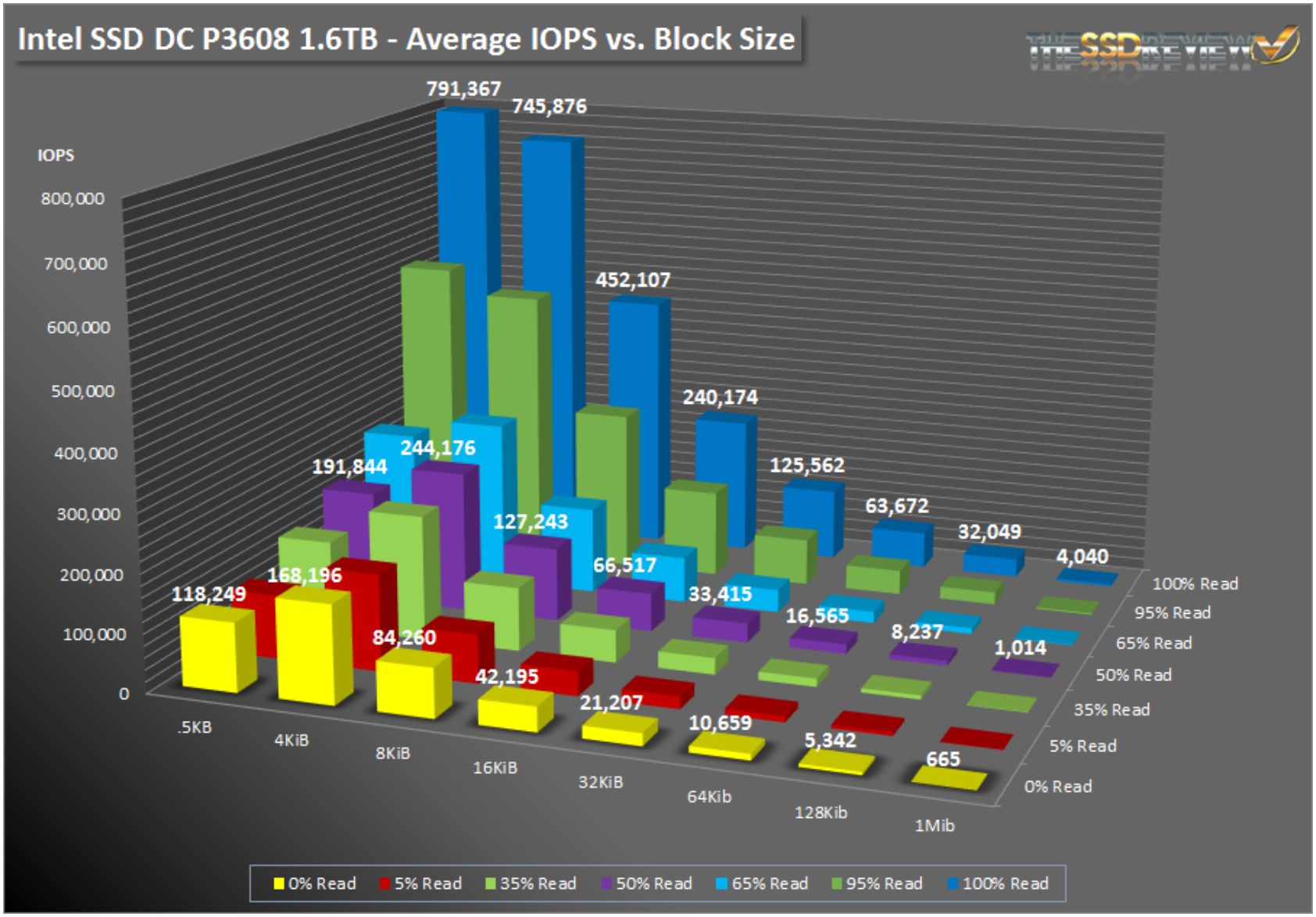
Ref: http://agigatech.com/blog/ssds-some-cold-hard-numbers-to-flavor-your-opinions/

Scale-up

Hardware

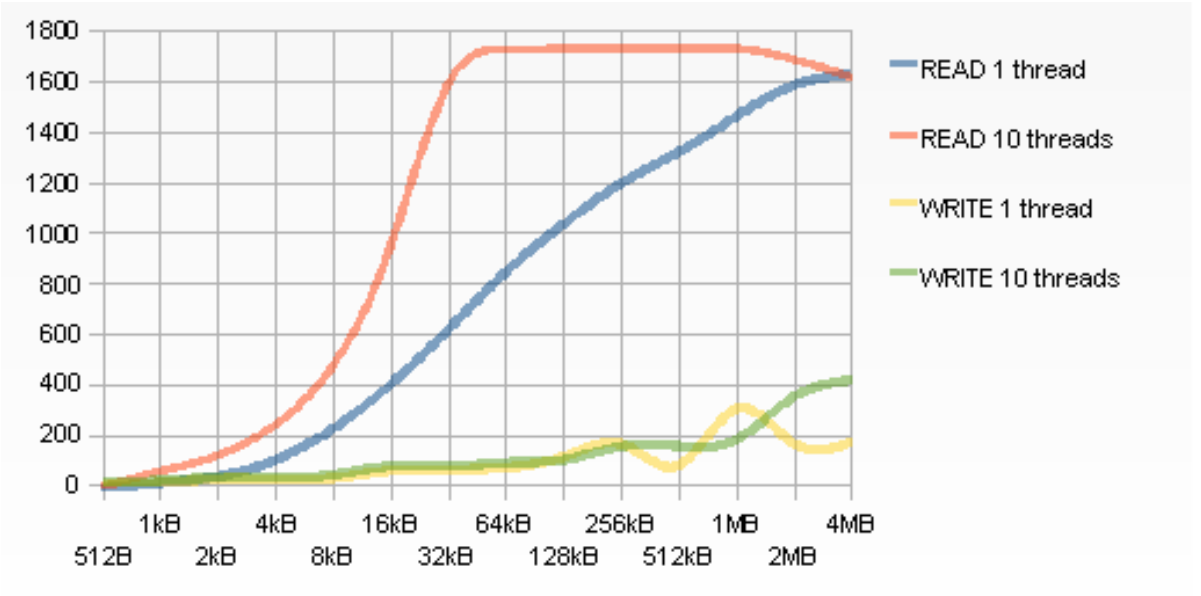
Storage

❶ 原則上，PCIe NVMe SSD > SSD > HDD 。
❷ 區塊大小 (Block size) 對 SSD 很重要。

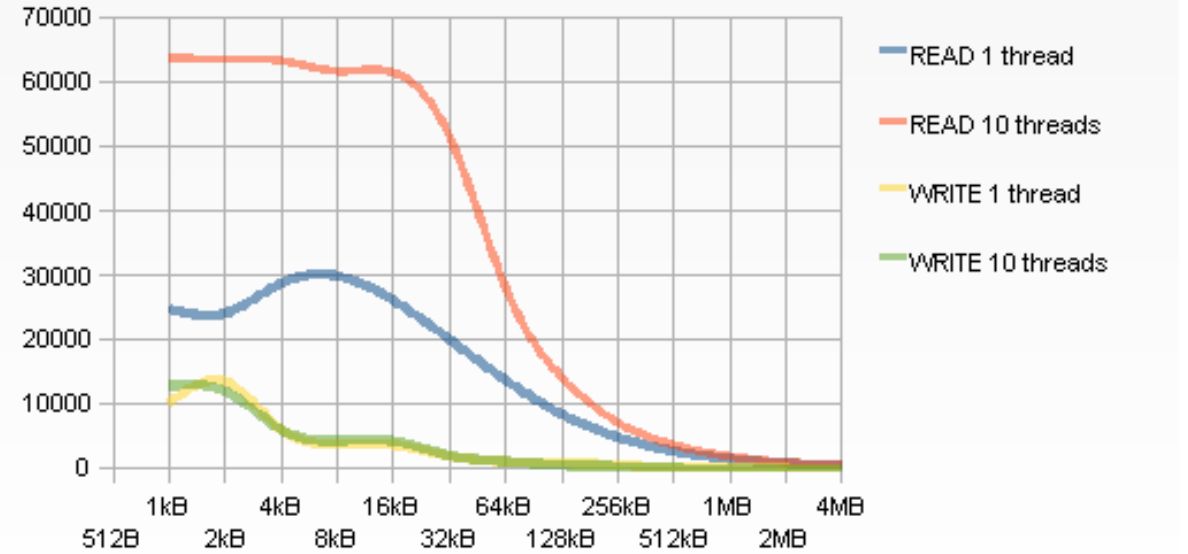


Ref: http://www.thessdreview.com/our-reviews/intel-ssd-dc-p3608-review-1-6tb-over-5gbs-and-850k-iops/2/

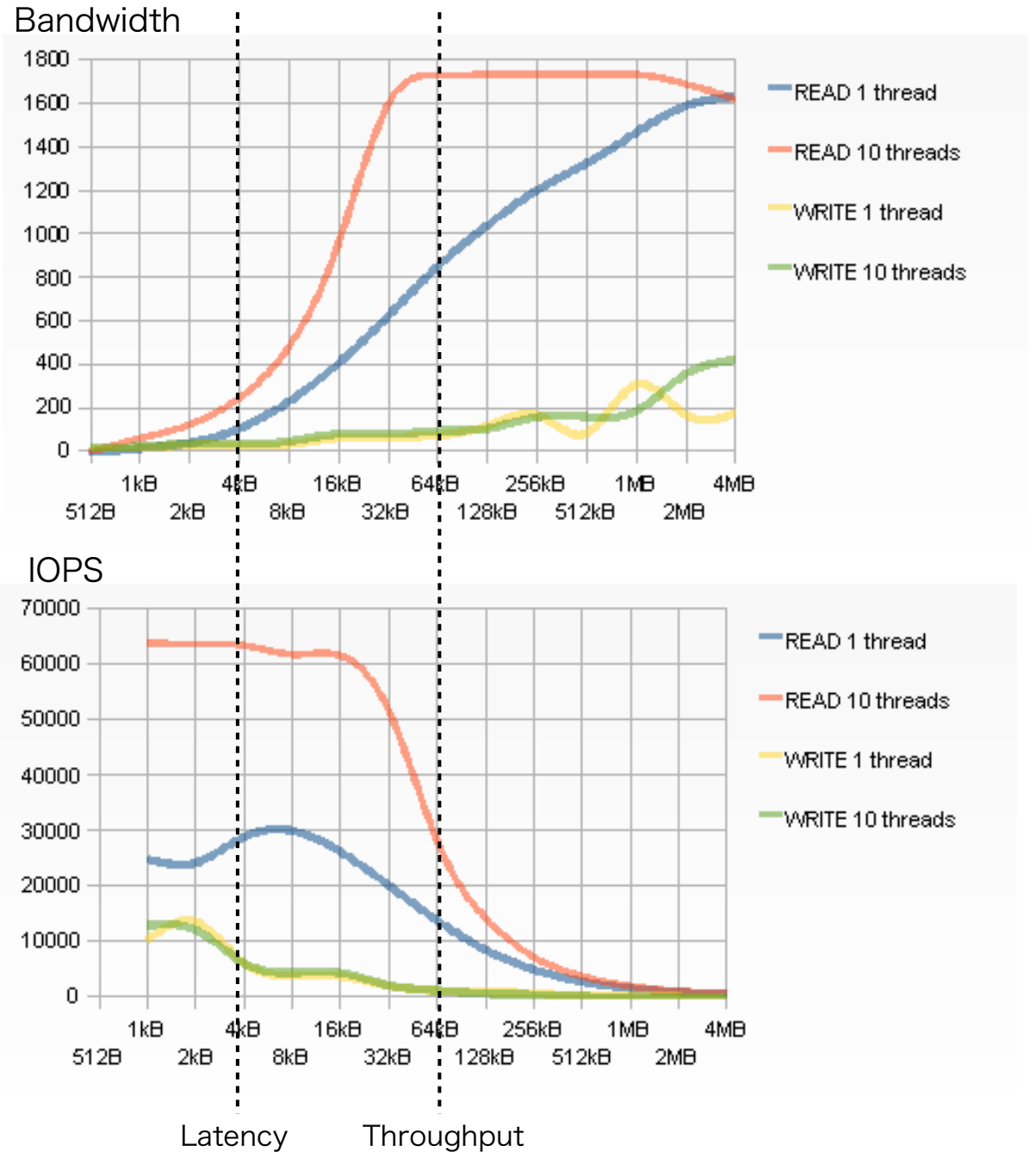
Bandwidth



IOPS



Ref: http://jdevelopment.nl/2009/02/



Ref: http://jdevelopment.nl/2009/02/

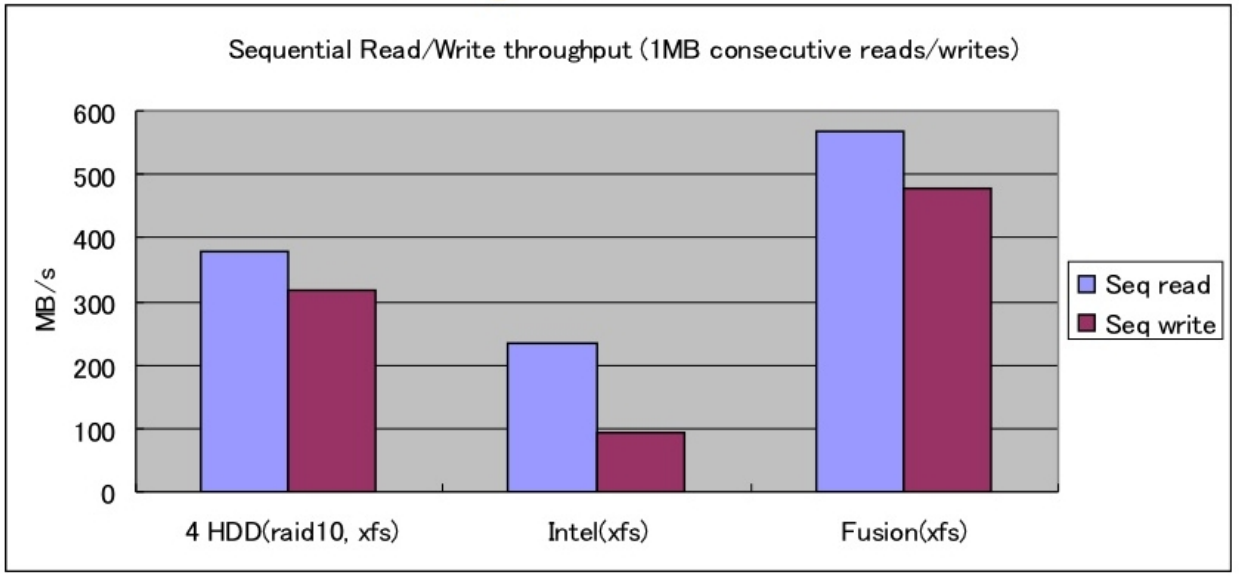
Scale-up

Hardware

Storage

❶ 原則上， PCIe NVMe SSD > SSD > HDD 。
❷ 區塊大小 (Block size) 對 SSD 很重要。
❸ 循序寫 +RAID 的 HDD 不比 SSD 差。

Sequential I/O



RAID-10 > RAID-5 (RAID 控制器很重要)
battery backed up write cache (BBWC)

Ref: SSD Deployment Strategies for MySQL (2010-04-15).pdf (p16)
Ref: http://yoshinorimatsunobu.blogspot.tw/2009/05/tables-on-ssd-redobinlogsystem.html

Scale-up

OS Kernel

❶ vm.swappiness = 1

Scale-up

OS Kernel

❶ vm.swappiness = 1
❷ vm.dirty_background_ratio / vm.dirty_ratio

Scale-up

OS Kernel

❶ vm.swappiness = 1
❷ vm.dirty_background_ratio / vm.dirty_ratio
❸ IO scheduler (DEADLINE or NOOP)

Scale-up

File system

OLTP


```
Ext4 / XFS
Journal / O_DIRECT
COMPRESSION
...
```

Scale-up

Database

Design

```
MT-malloc: jemalloc / tcmalloc / etc.
DB Engine: InnoDB / TokuDB / RocksDB
Schema design / ID
Index
Partitions
```

Configuration

```
default_time_zone = '+00:00'
max_connections
sort_buffer_size
join_buffer_size
read_buffer_size
innodb_use_native_aio = 1
innodb_file_per_table = 1
innodb_buffer_pool_size = { 65~80% of Mem }
innodb_thread_concurrency = { 2xCPUs }
innodb_read_io_threads = { CPUs }
innodb_write_io_threads = { CPUs }
```

Scale-up

Database

OLTP


```
innodb_flush_method
innodb_max_dirty_pages_pct
innodb_page_size
innodb_io_capacity
innodb_flush_neighbors
innodb_random_read_ahead
innodb_read_ahead_threshold
...
```

Scale-up

Connection

Connection pool (Client/Application)

❶ 不是所有應用程式都支援。
❷ 無法得知伺服器的狀態及承載。
❸ 遭遇錯誤時，必須執行完整的資源清理。
❺ 會保持連線，佔用伺服器連線數及線程快取。

Scale-up

Connection

架構問題

Application

最大連線數 100

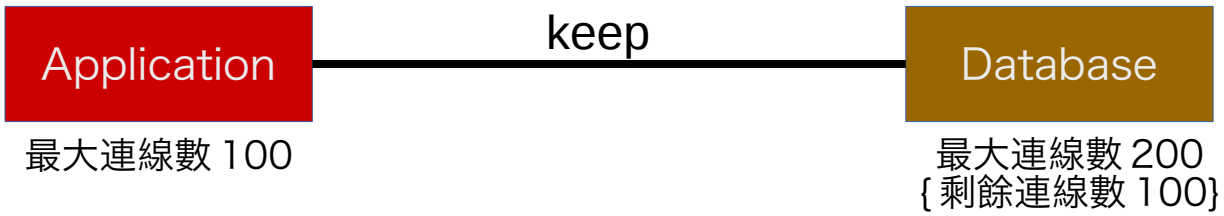
Database

最大連線數 200

Scale-up

Connection

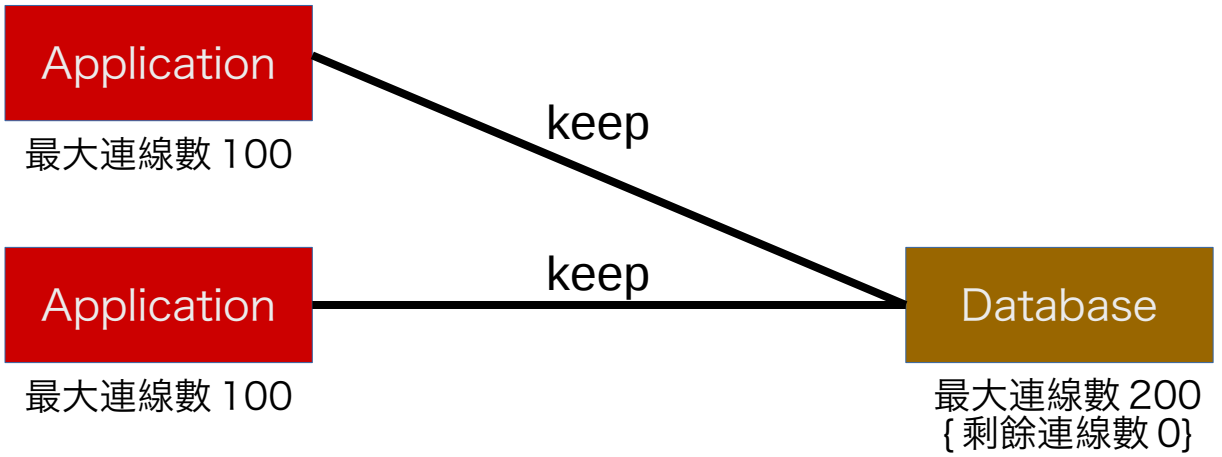
架構問題



Scale-up

Connection

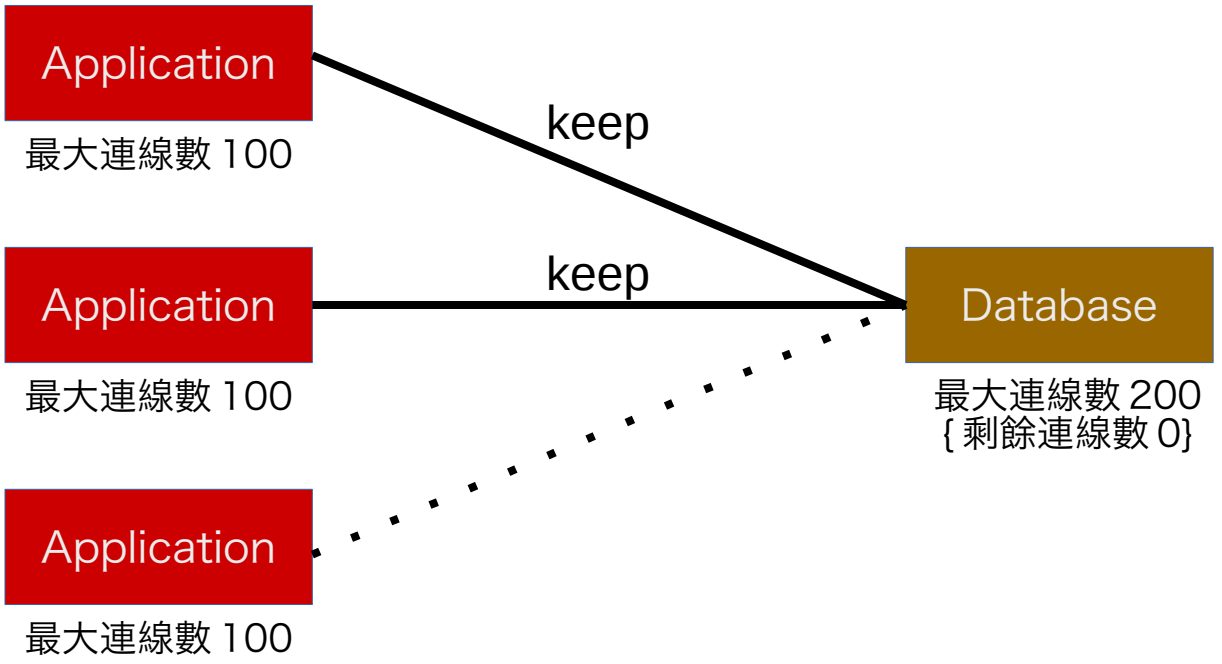
架構問題



Scale-up

Connection

架構問題



Scale-up

Connection

架構問題

MySQL

❶ 線程模式。
❷ 不需 Connection pool 就可以支持高併發。
❸ 支持短連接使用資料庫。
❹ 新版 5.7 建立連線的開銷更少。
　　5.6 版的 62.5% ;
　　5.5 版的 40% 。

Scale-up

Application

效能通常有 99% 的問題在於 Application

❶ N+1 queries / ORM
❷ Bad SQL
❸ Bad Schema Design
❹ Big SQL
❺ Big Transaction
❻ Big Batch

Scale-up

Application

效能通常有 99% 的問題在於 Application

❶ N+1 queries / ORM
❷ Bad SQL
❸ Bad Schema Design
❹ Big SQL
❺ Big Transaction
❻ Big Batch

Scale-up

Application

(MySQL) CHAR vs. VARCHAR

① 如果更新頻繁且長度不一， CHAR 通常比較快。

② 在 MySQL 5.7.7 之後， CHAR 通常比 VARCHAR 快。

Scale-up

Application

(MySQL) VARCHAR vs. VARCHAR

① 某些編碼下， VARCHAR(760) 與 VARCHAR(770) 快得多。

② 某些編碼下， VARCHAR(190) 比 VARCHAR(200) 快得多。

③ 不過在 MySQL 5.7.7 之後，前兩者幾乎沒什麼差別。

Scale-up

Application

INDEX

① Primary Index 對 MySQL 很重要，循序式比亂序式快。

② Index 愈多不一定愈好。

③ Composite Index 需善用。

Architecture

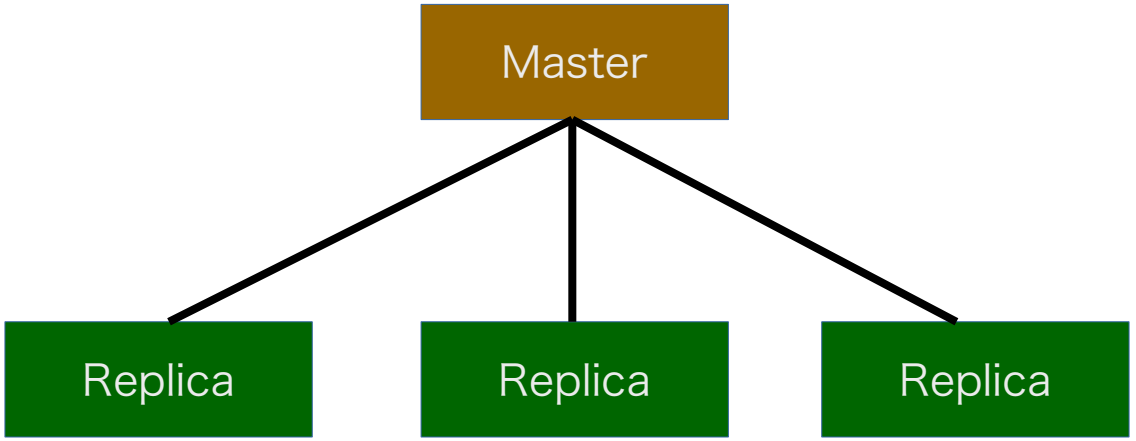
Business	Technology
License Elastic business Workload	Scale-up Application Connection Database File system OS Kernel Hardware Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE

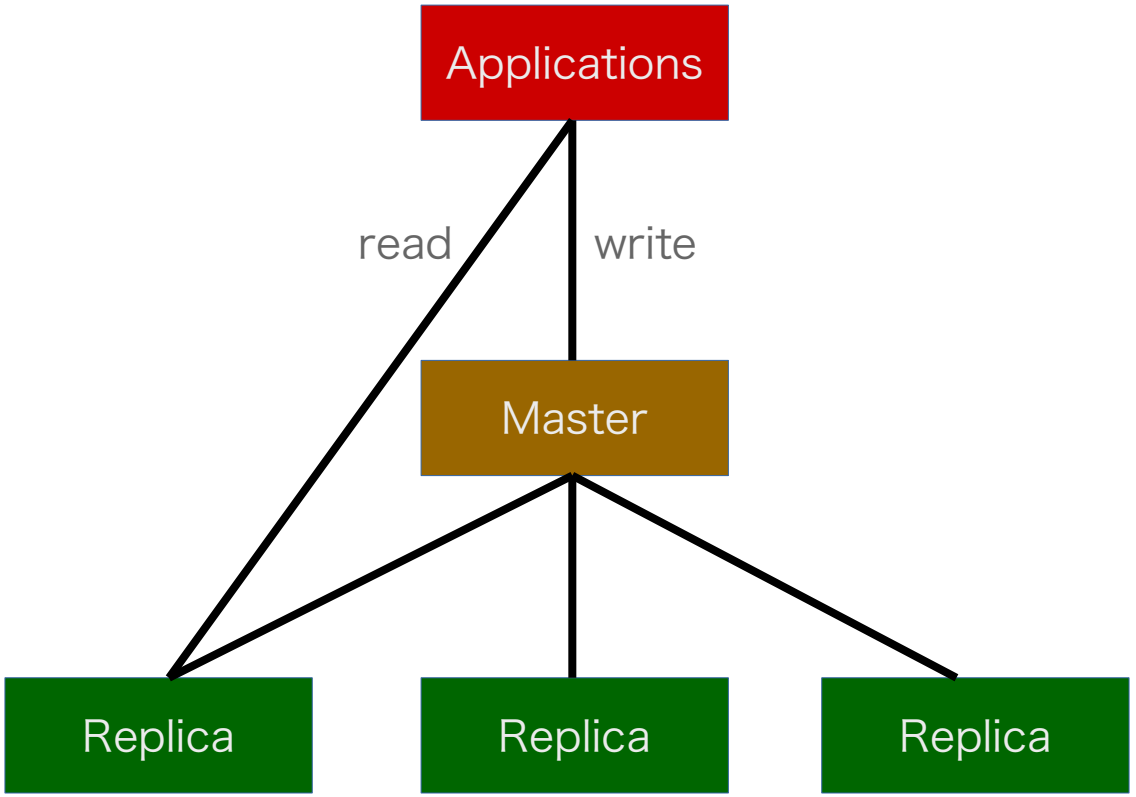
Scale-out

Replication



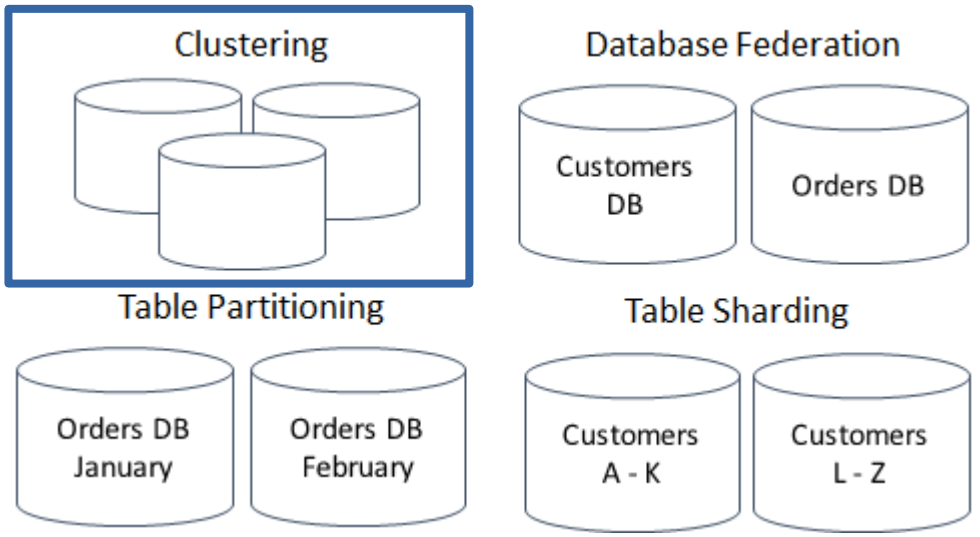
Scale-out

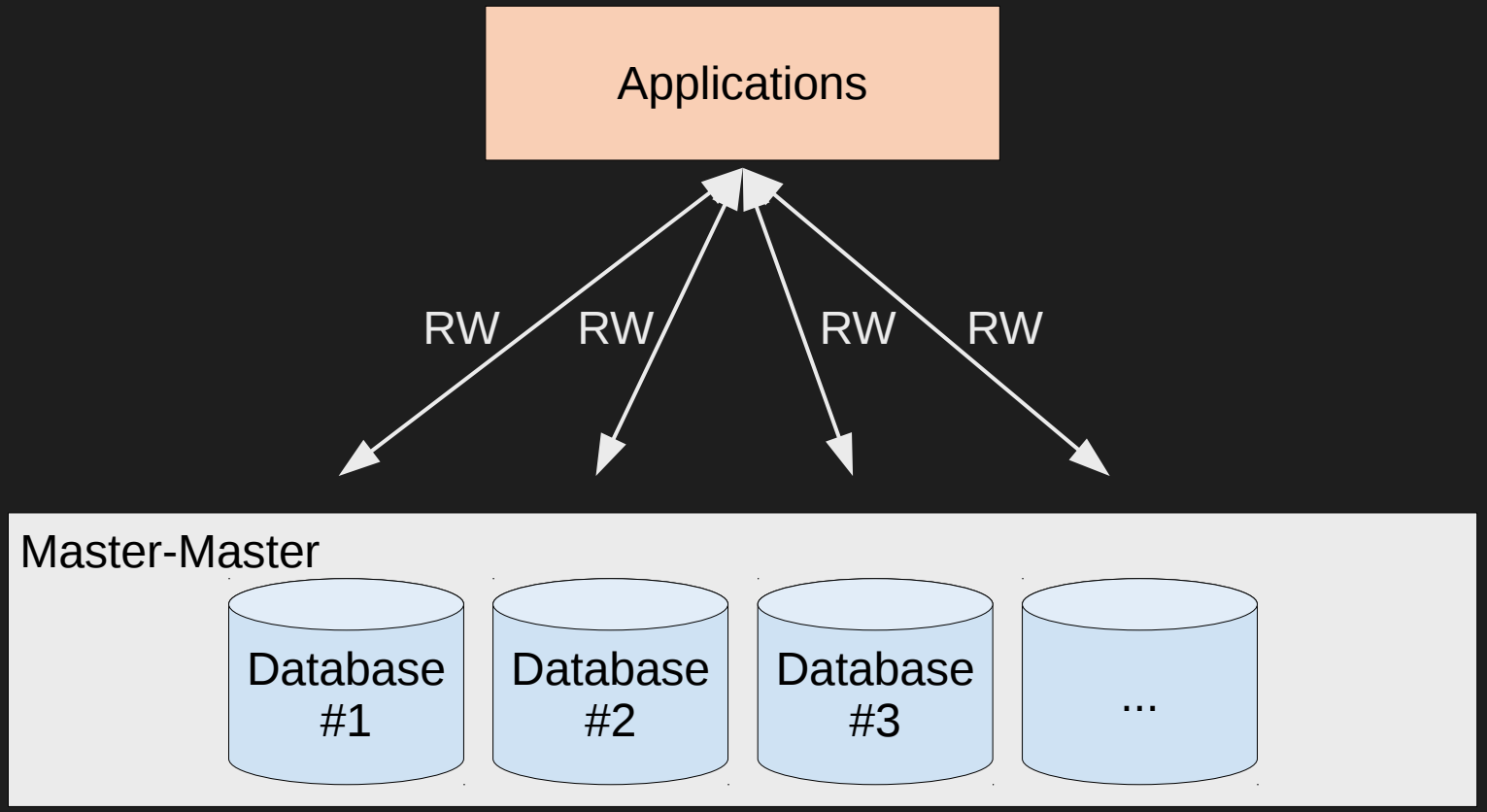
Replication

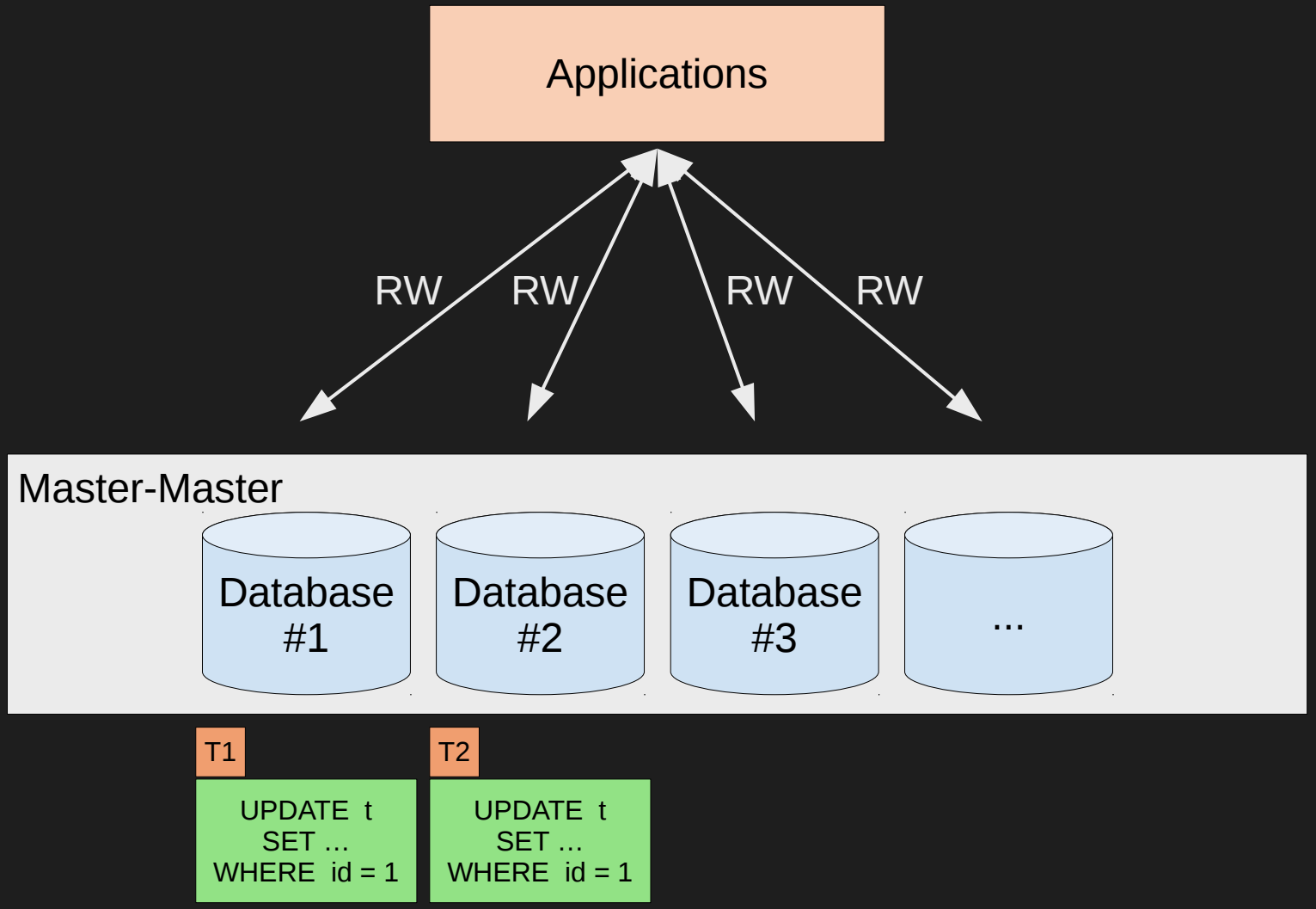


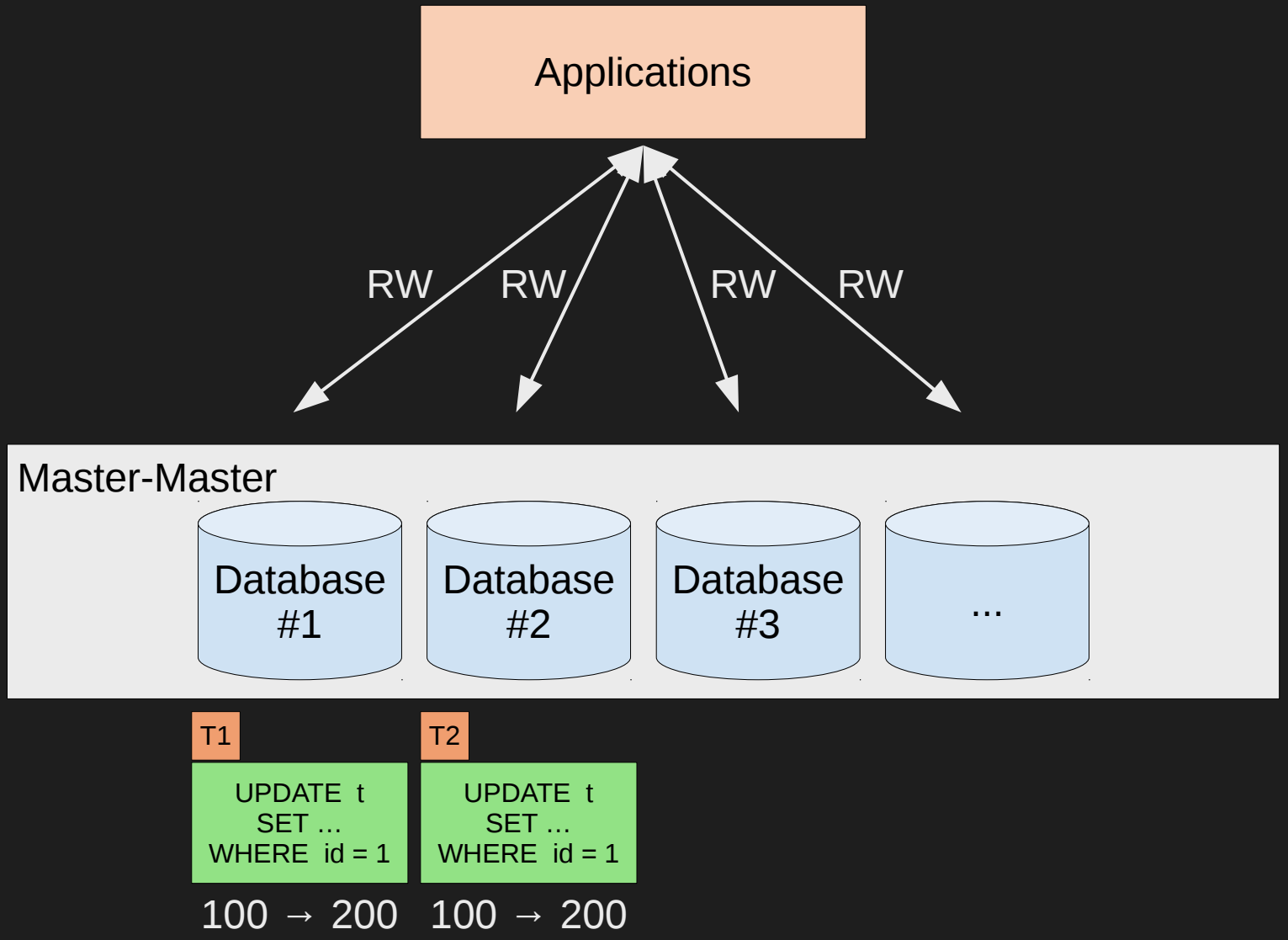
Scale-out

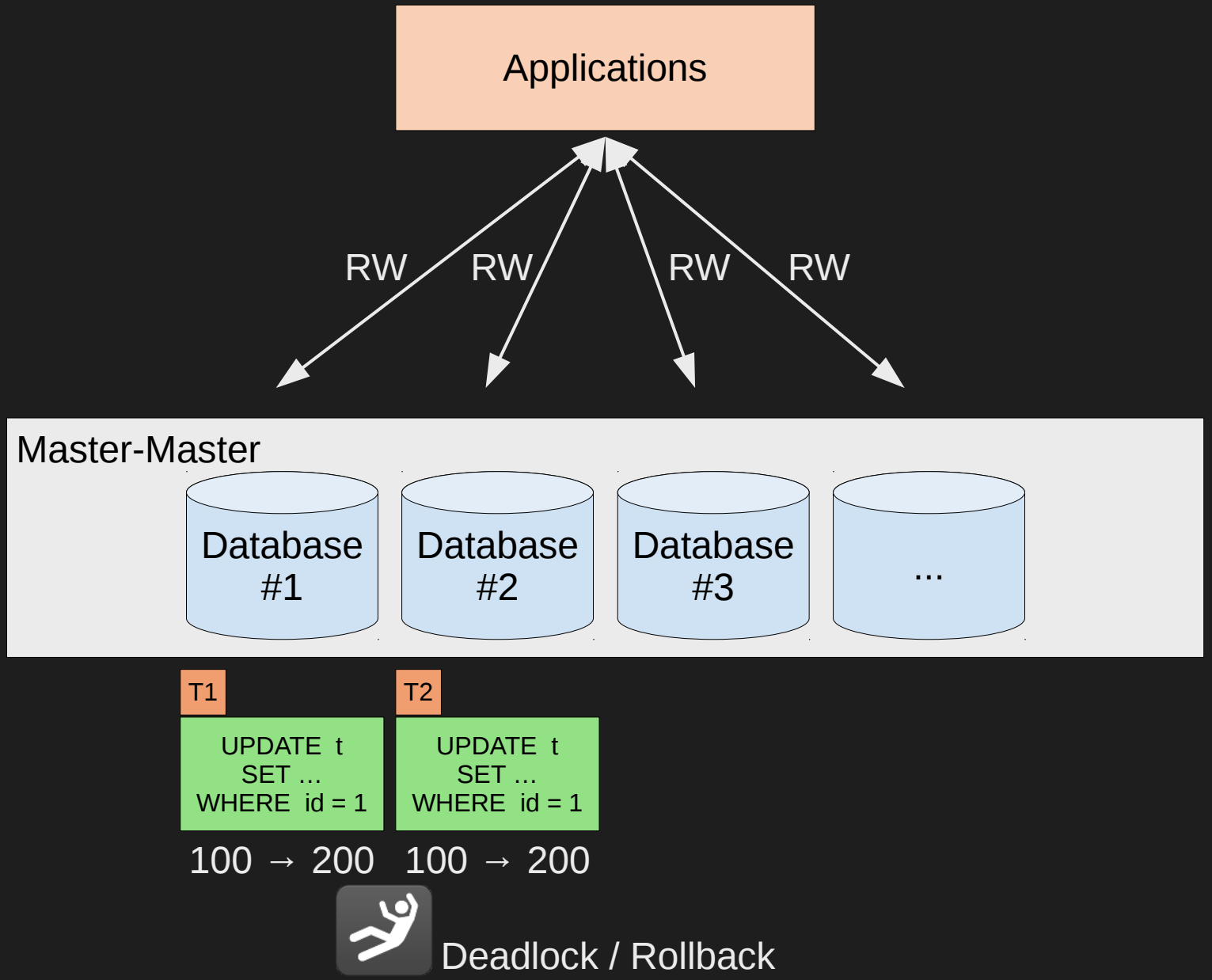
Clustering

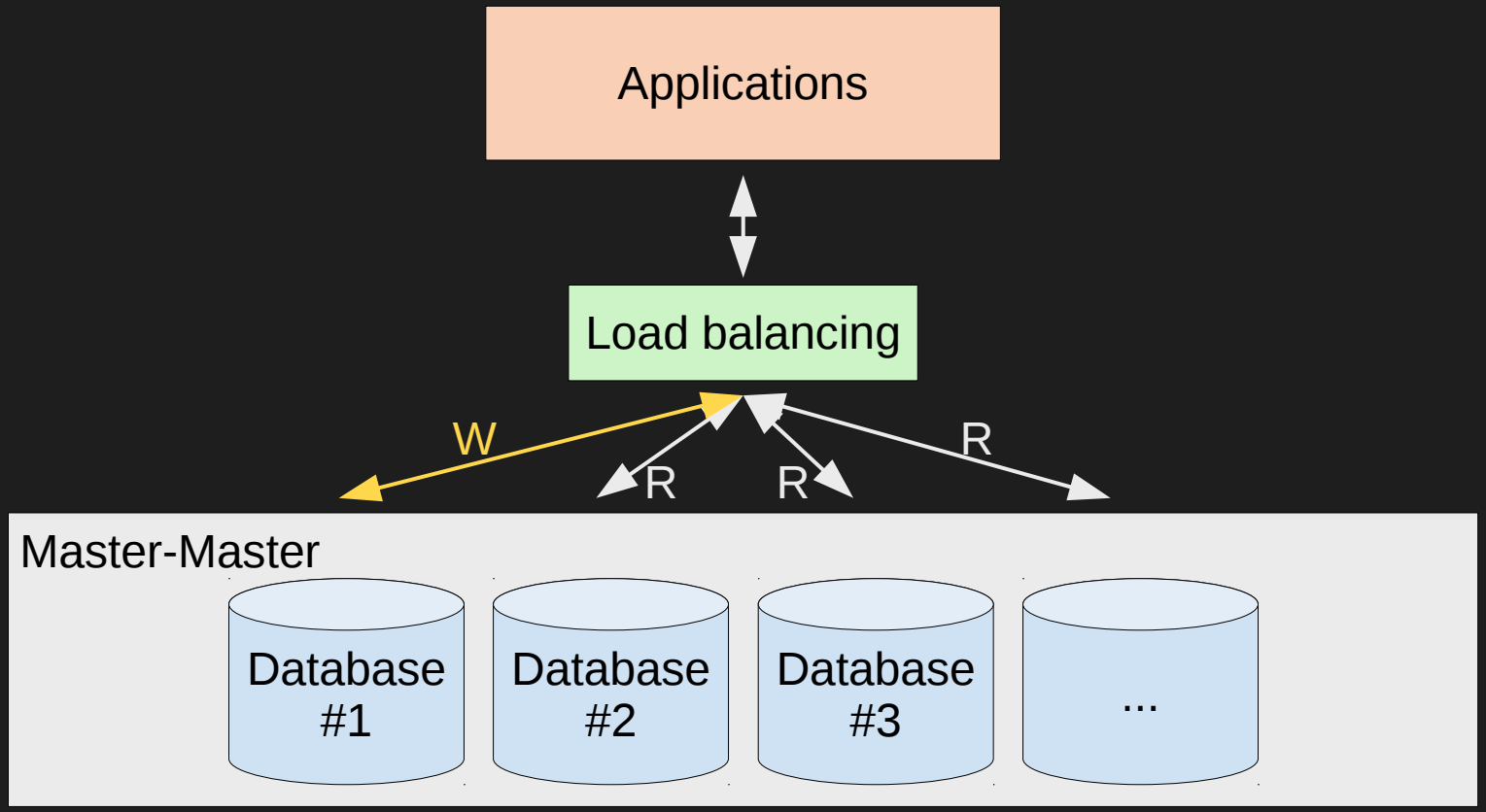


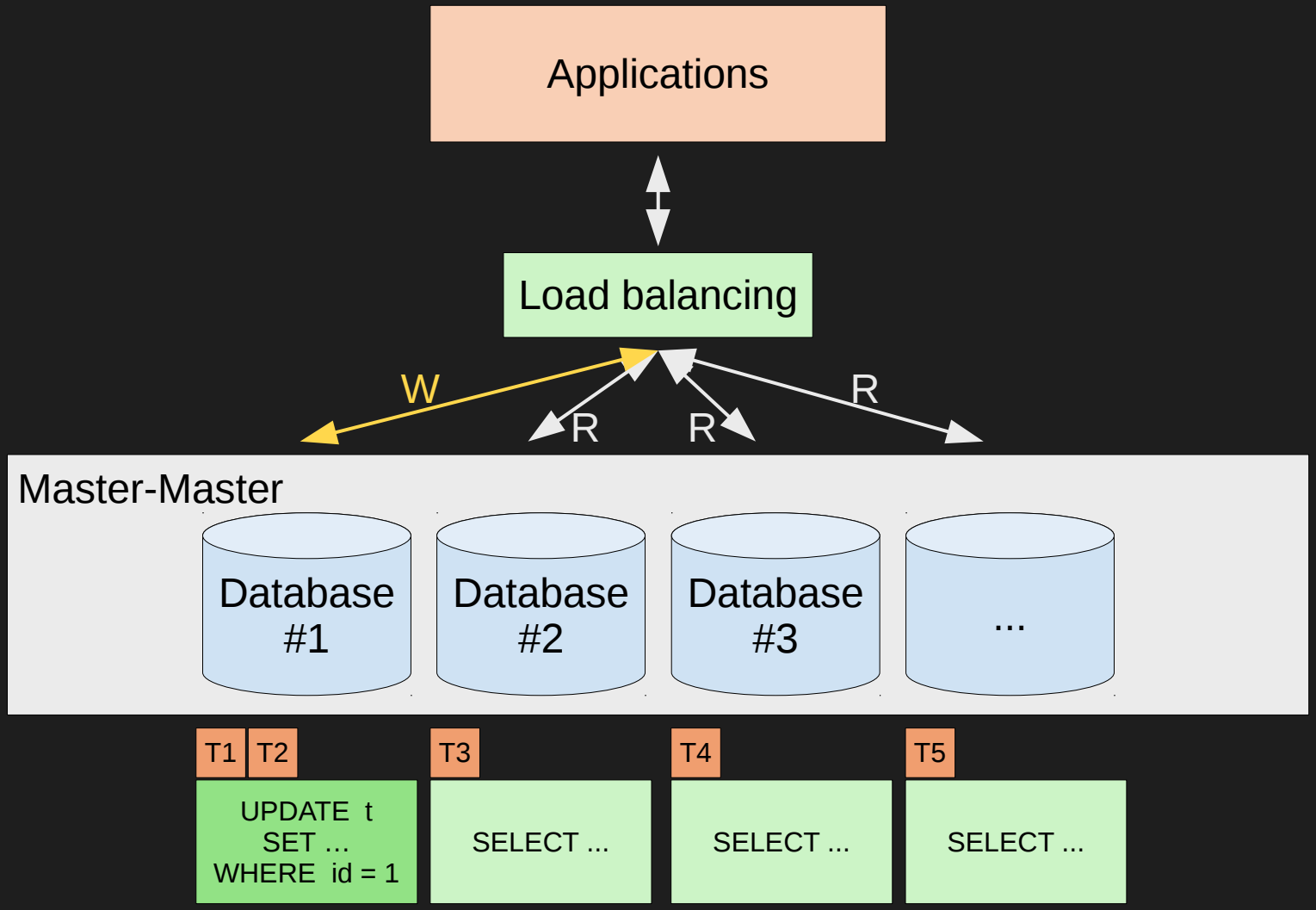

Applications
RW
RW
RW
RW
Master-Master
Database #1
Database #2
Database #3
...

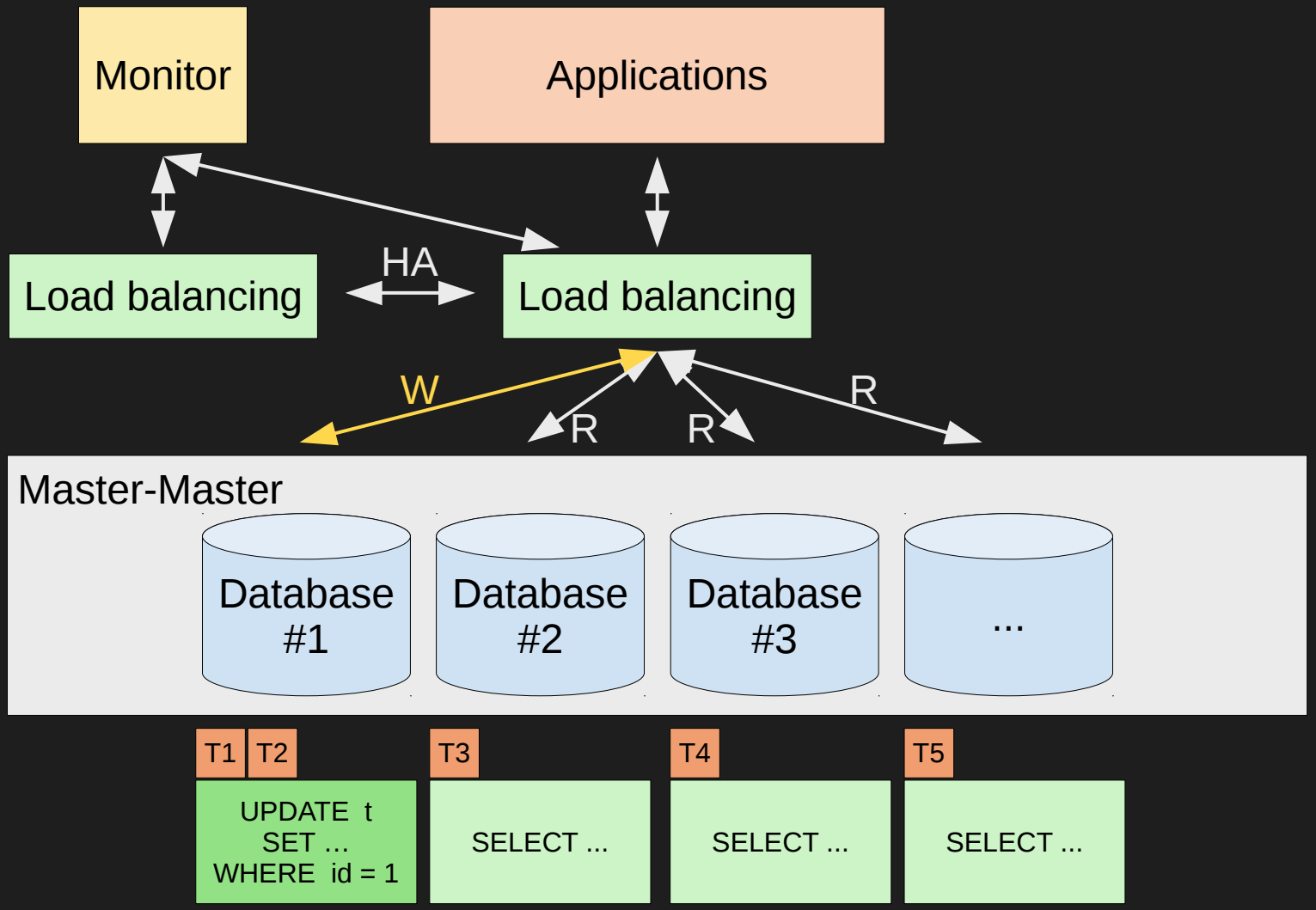

Applications
RW
RW
RW
RW
Master-Master
Database #1
Database #2
Database #3
...
T1
UPDATE t
SET …
WHERE id = 1
T2
UPDATE t
SET …
WHERE id = 1


Applications
RW
RW
RW
RW
Master-Master
Database #1
Database #2
Database #3
...
T1
UPDATE t
SET …
WHERE id = 1
T2
UPDATE t
SET …
WHERE id = 1
100 → 200
100 → 200


Applications
RW
RW
RW
RW
Master-Master
Database #1
Database #2
Database #3
...
T1
UPDATE t
SET …
WHERE id = 1
T2
UPDATE t
SET …
WHERE id = 1
100 → 200
100 → 200
Deadlock / Rollback


Applications
Load balancing
W
R
R
R
Master-Master
Database #1
Database #2
Database #3
...

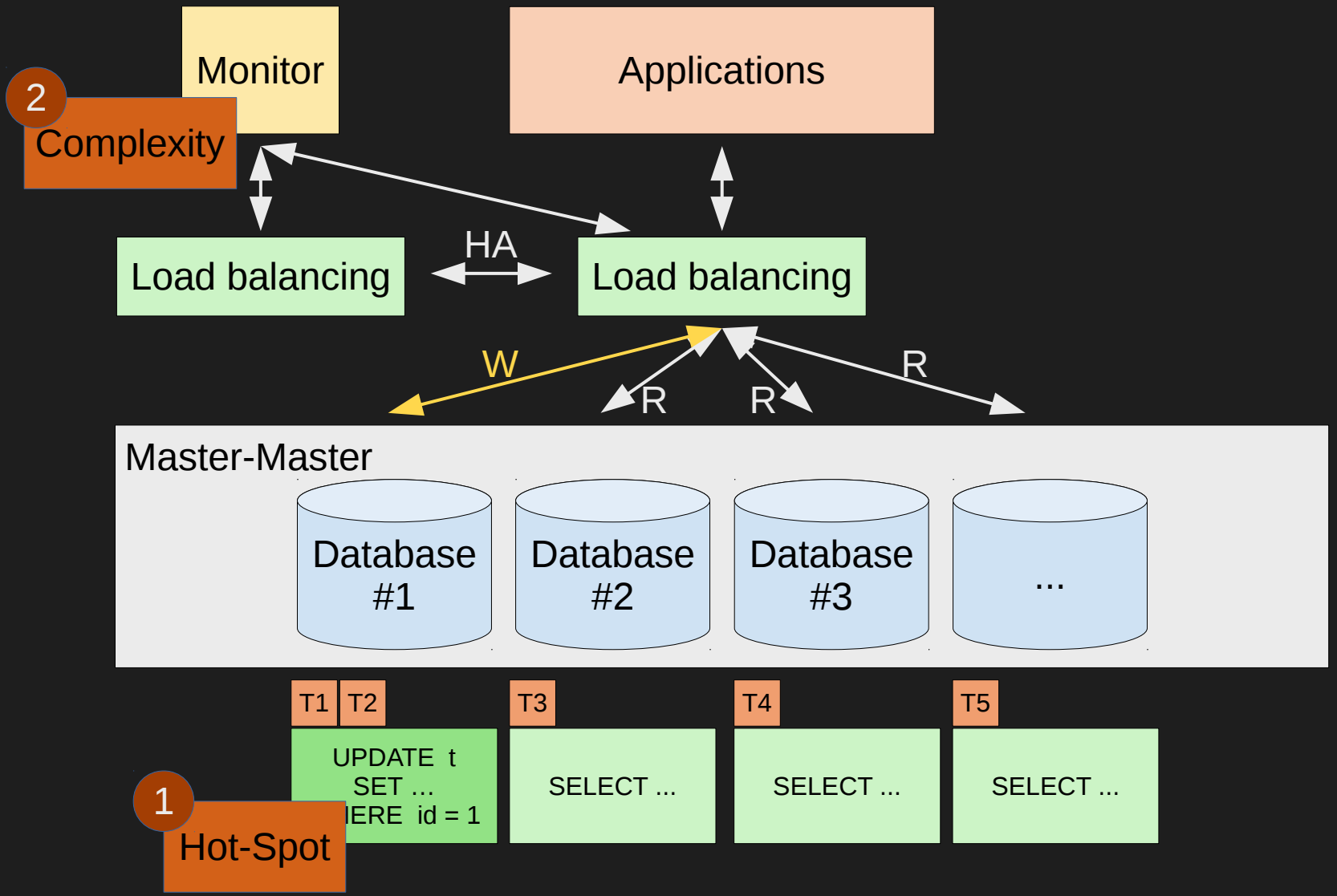

Applications
Load balancing
W
R
R
R
Master-Master
Database #1
Database #2
Database #3
...
T1
T2
T3
T4
T5
UPDATE t
SET …
WHERE id = 1
SELECT ...
SELECT ...
SELECT ...

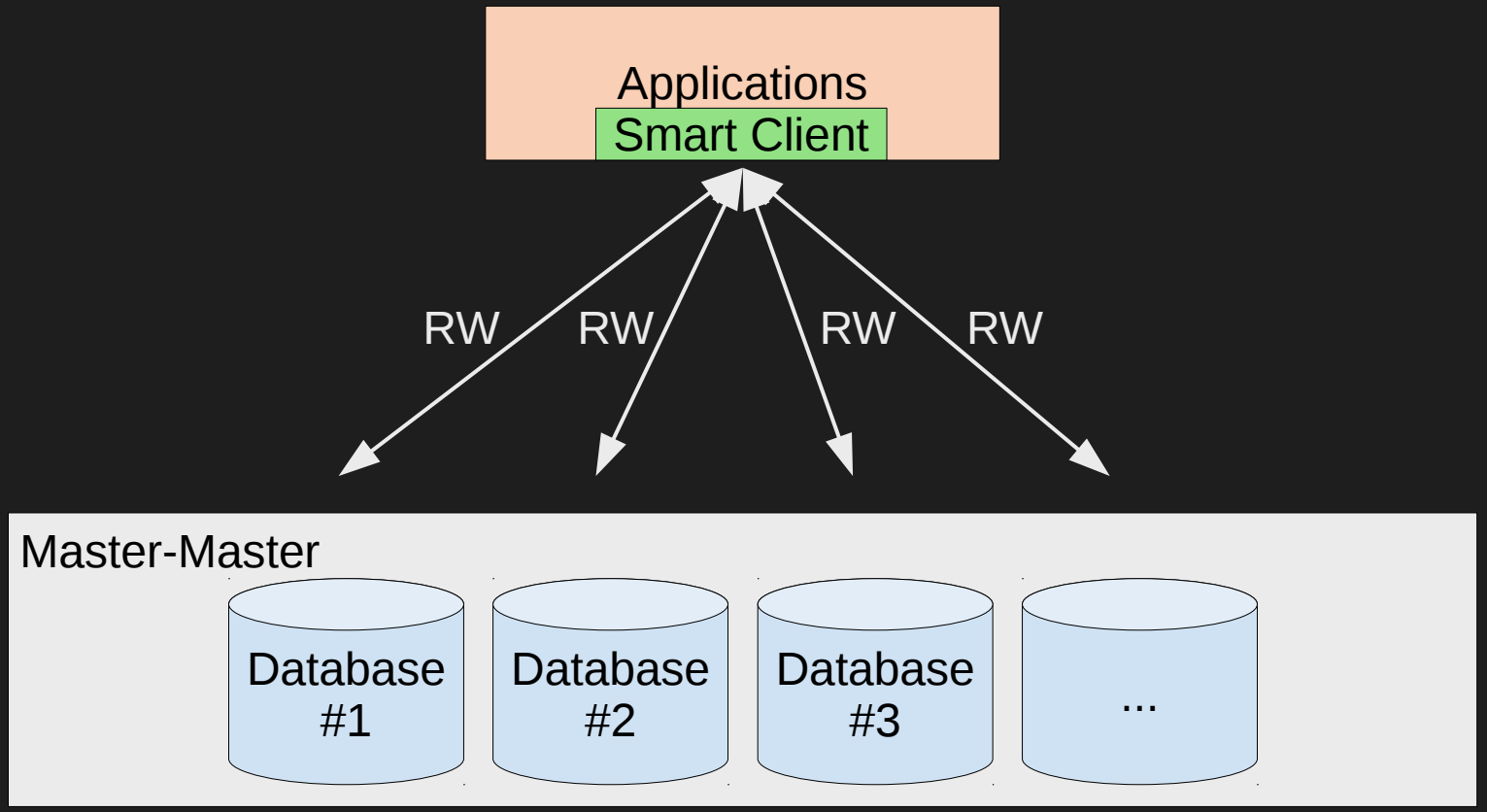

Monitor
Applications
Load balancing
HA
Load balancing
W
R
R
R
Master-Master
Database #1
Database #2
Database #3
...
T1
T2
T3
T4
T5
UPDATE t SET … WHERE id = 1
SELECT ...
SELECT ...
SELECT ...

Percona XtraDB Cluster: Multi-node writing and Unexpected deadlocks
2012-08-17
https://www.percona.com/blog/2012/08/17/percona-xtradb-cluster-multi-node-writing-and-unexpected-deadlocks/

Avoiding Deadlocks in Galera - Set up HAProxy for single-node writes and multi-node reads
2013-09-17
http://www.severalnines.com/blog/avoiding-deadlocks-galera-set-haproxy-single-node-writes-and-multi-node-reads

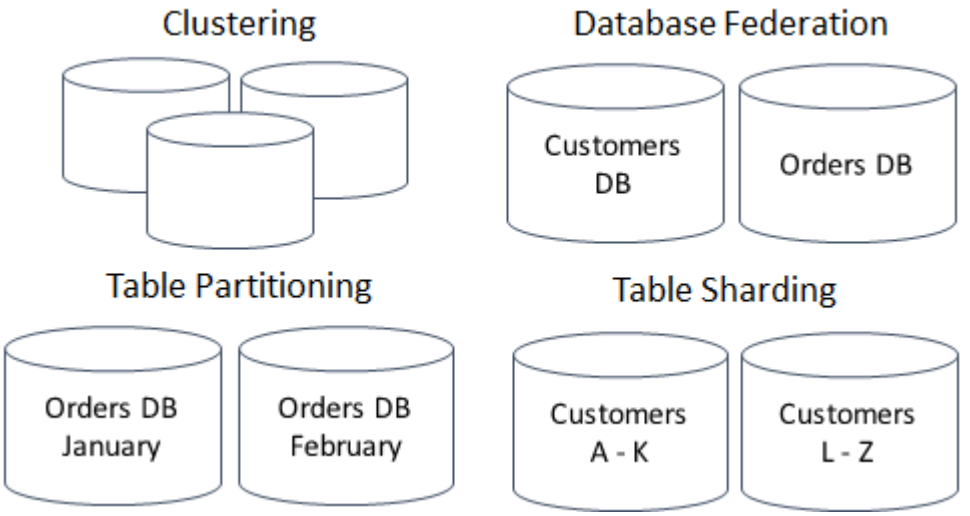
Optimizing Percona XtraDB Cluster for write hotspots
2015-06-03
https://www.percona.com/blog/2015/06/03/optimizing-percona-xtradb-cluster-write-hotspots/


Monitor
Applications
2
Complexity
Load balancing
HA
Load balancing
W
R
R
R
Master-Master
Database #1
Database #2
Database #3
...
T1
T2
T3
T4
T5
UPDATE t
SET …
WHERE id = 1
SELECT ...
SELECT ...
SELECT ...
1
Hot-Spot


Applications
Smart Client
RW
RW
RW
RW
Master-Master
Database #1
Database #2
Database #3
...

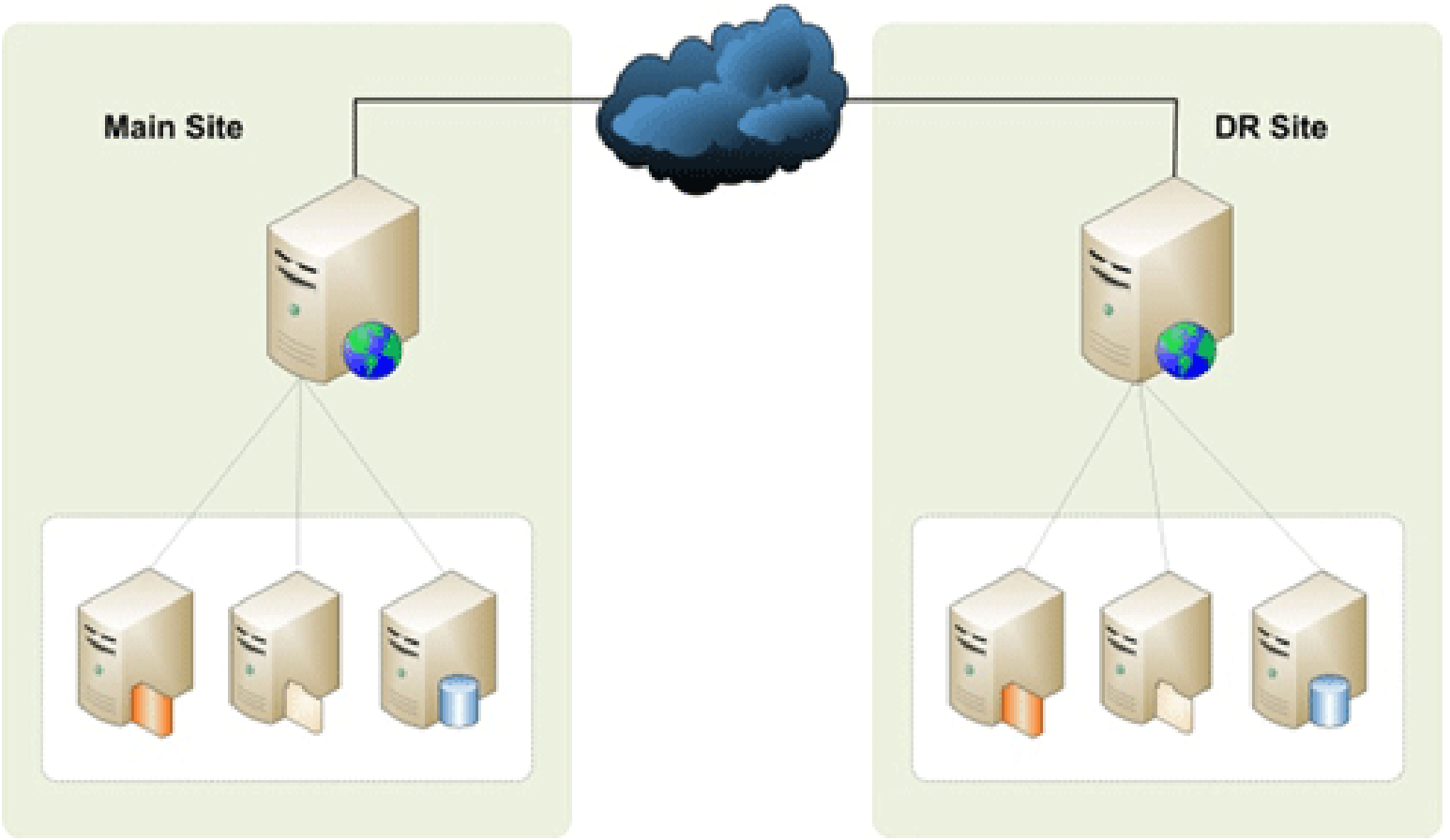
Scale-out

Sharding



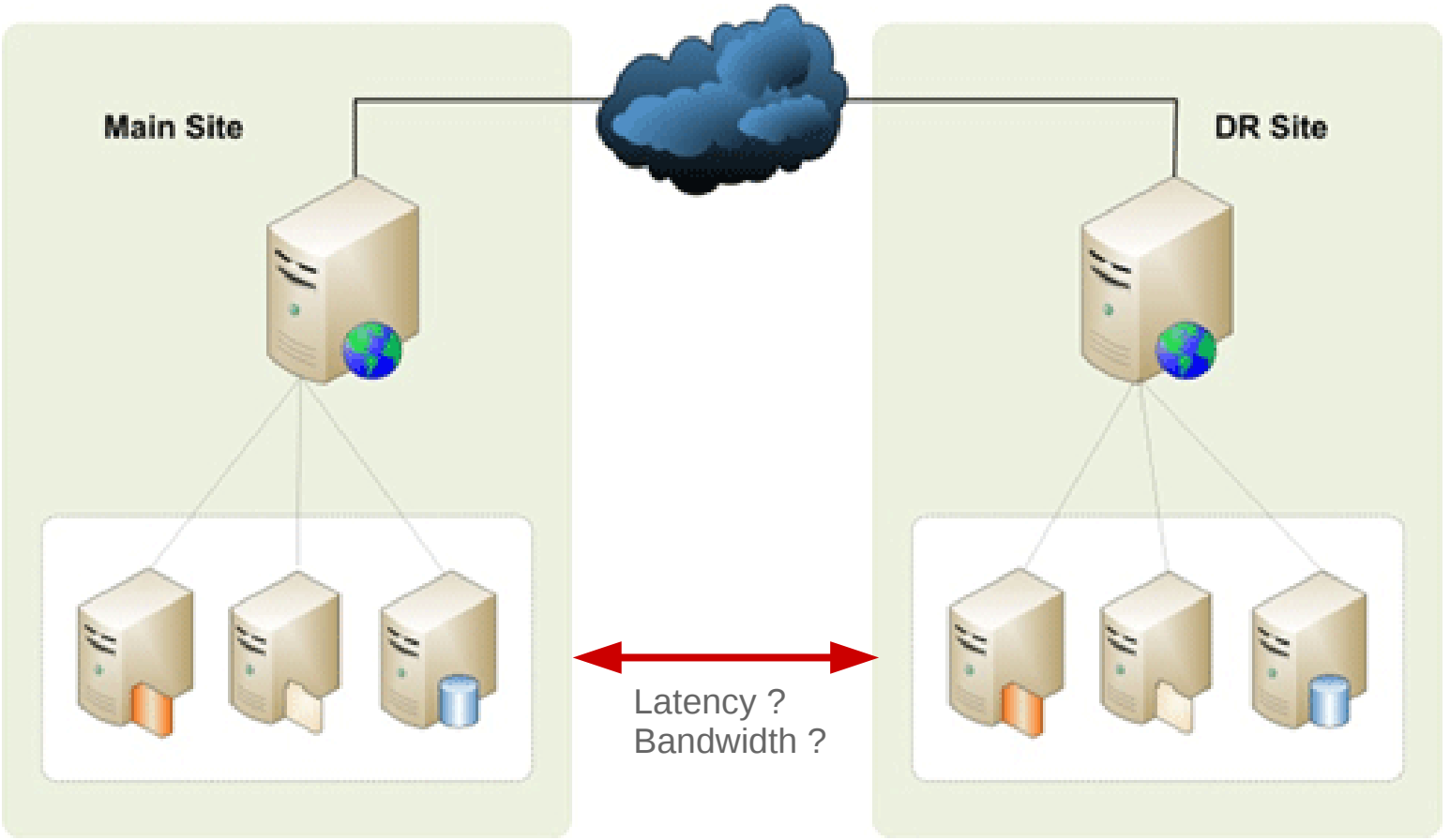
Scale-out

Disaster Recovery



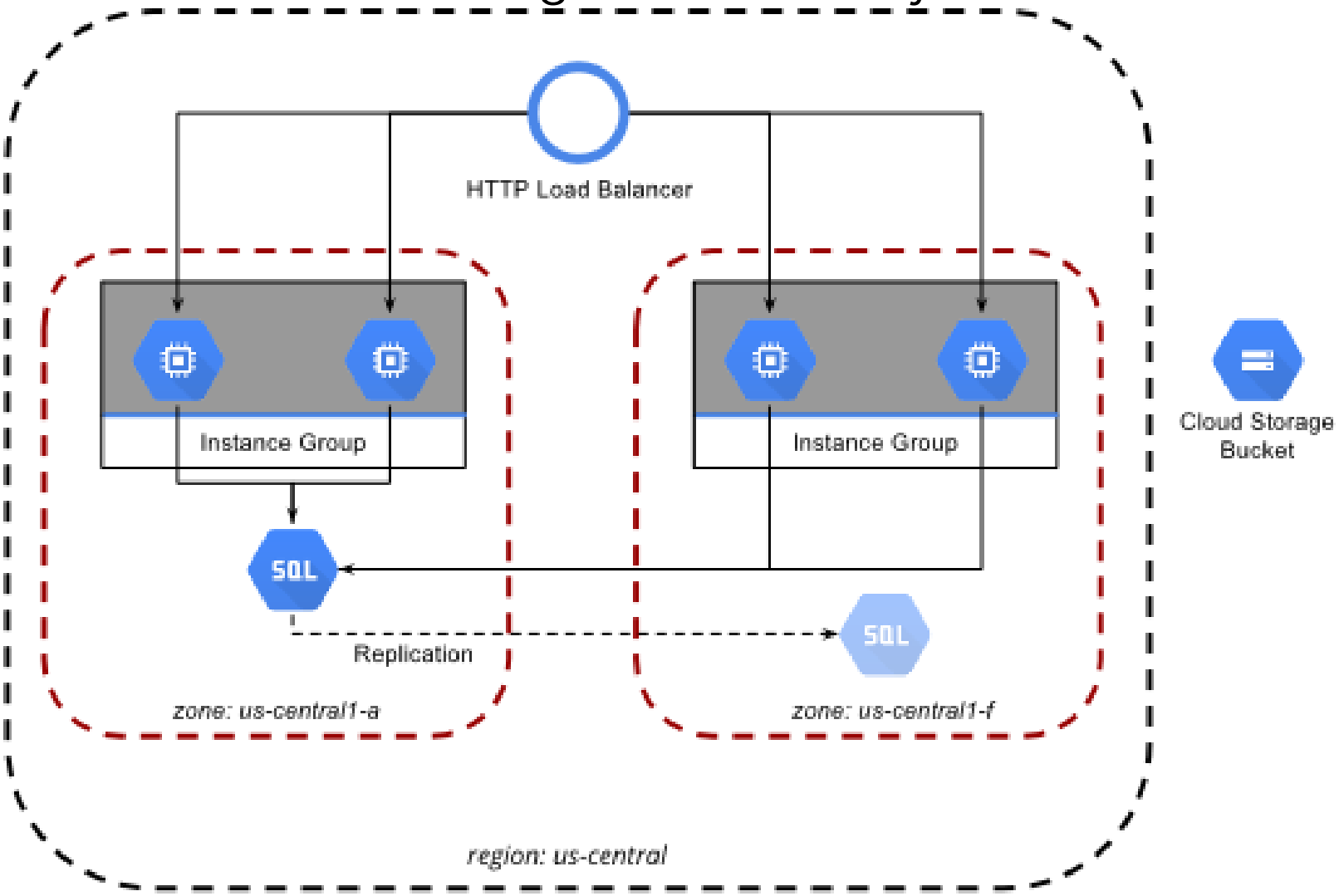
Scale-out

Disaster Recovery



Scale-out

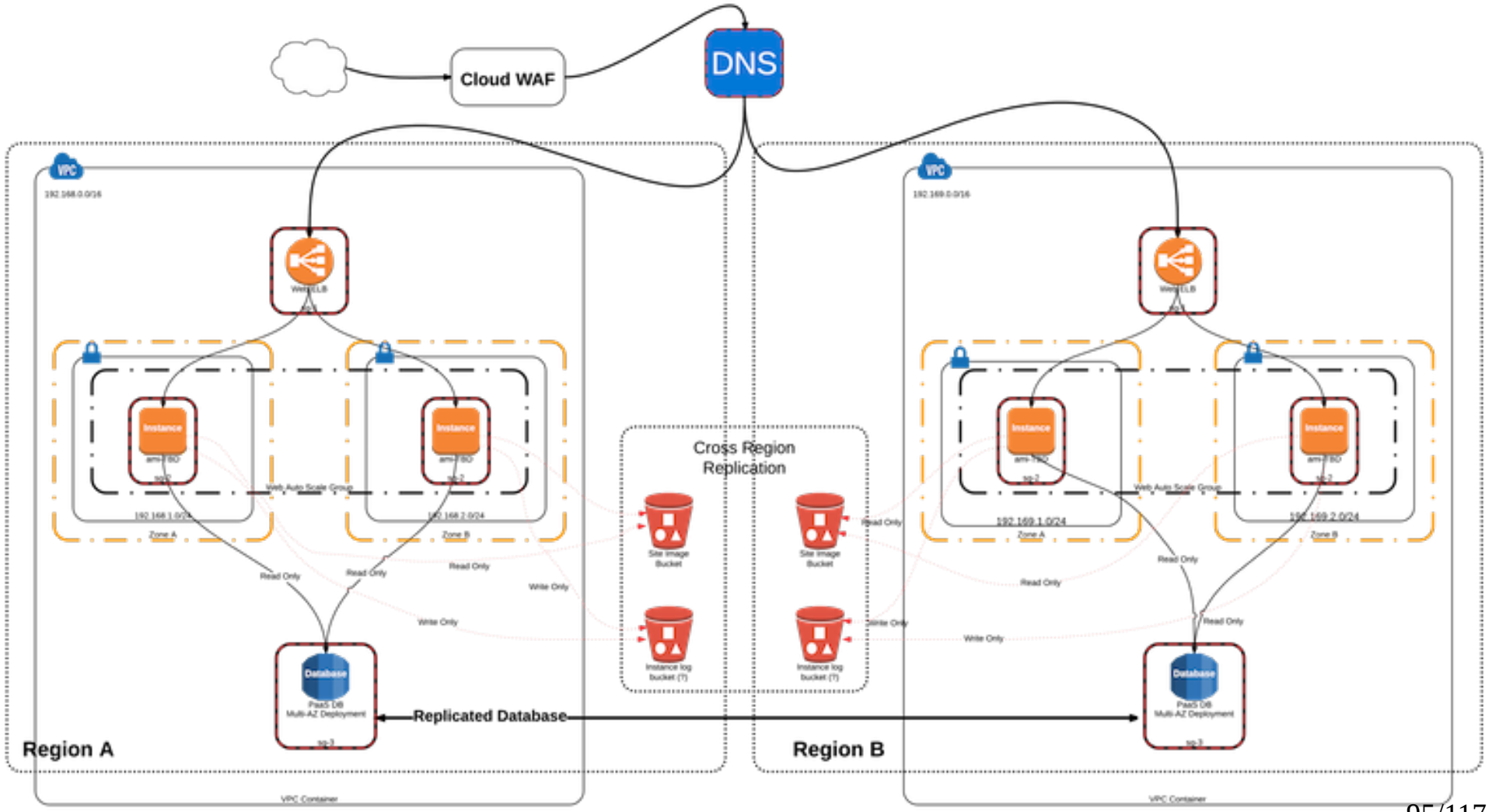
Multi Regional Resiliency



Ref: https://cloud.google.com/solutions/scalable-and-resilient-apps

Scale-out

Multi Regional Resiliency



Ref: https://securosis.com/blog/resilient-cloud-network-architectures-design-patterns

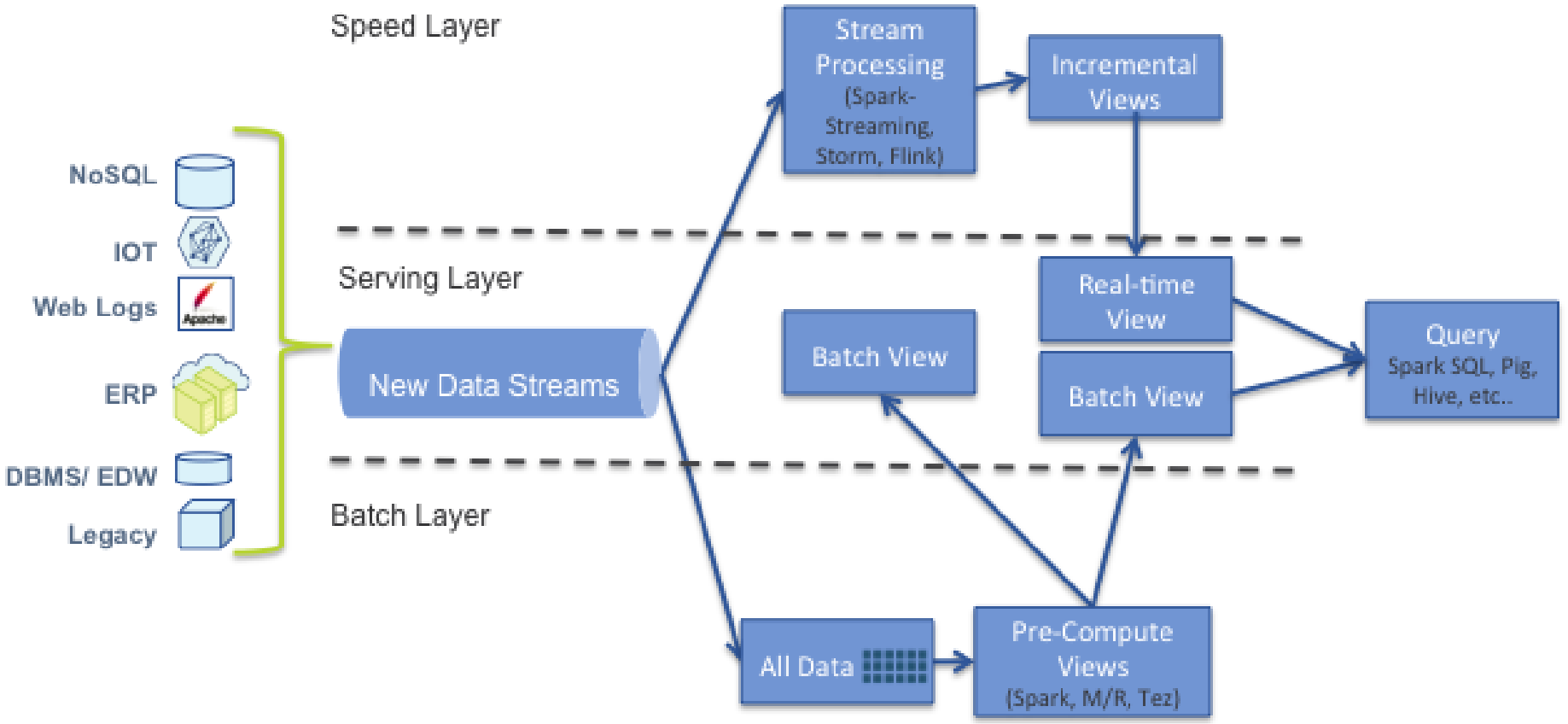
Architecture

Business	Technology
License Elastic business Workload	Scale-up Application Connection Database File system OS Kernel Hardware Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency

and more ...

CONSILIENCE


大數據
BigData



Ref: https://www.talend.com/blog/2015/07/15/hadoop-summit-2015-takeaway-the-lambda-architecture

SMACK Stack

- **Spark** - fast and general engine for distributed, large-scale data processing

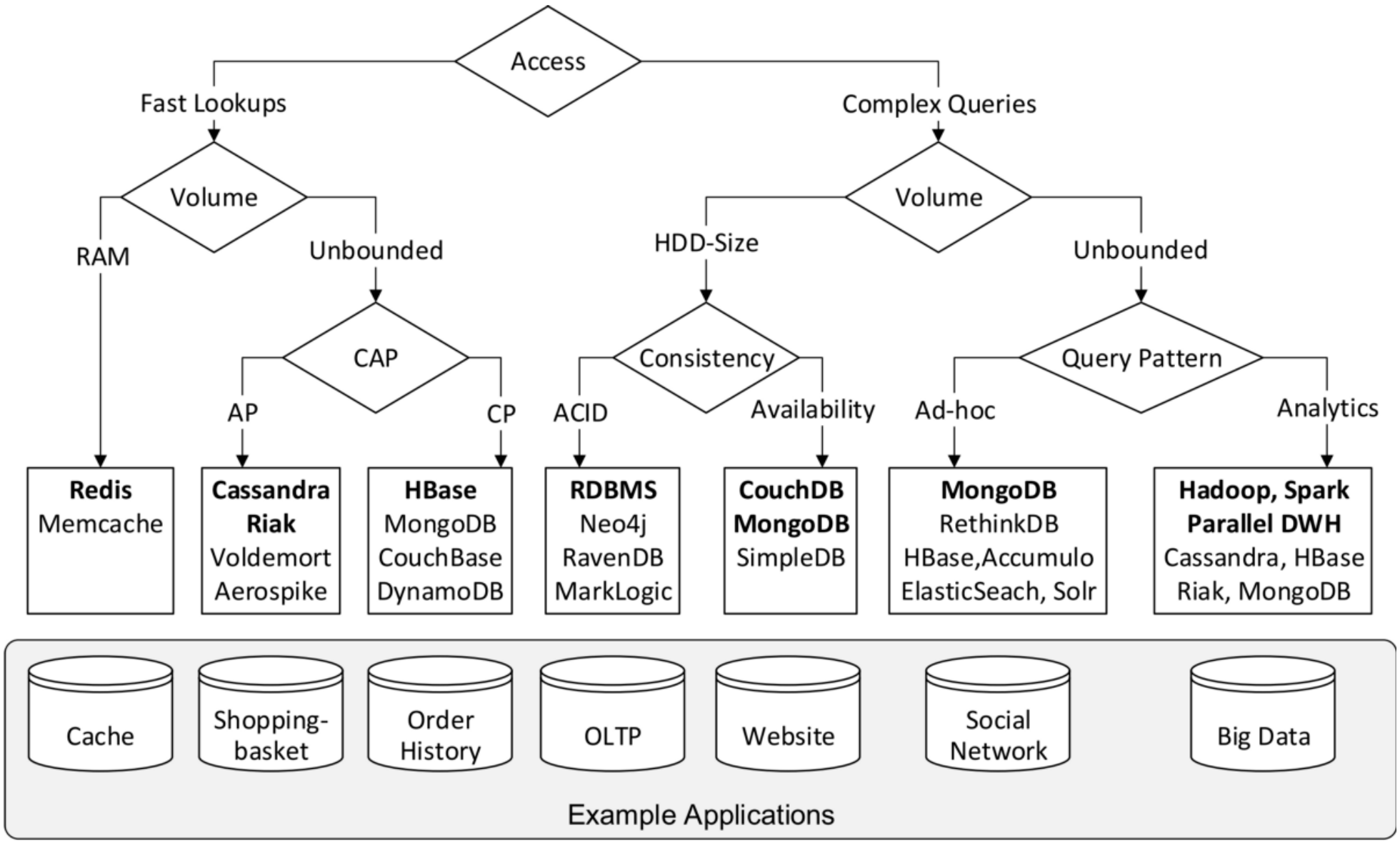
- **Mesos** - cluster resource management system that provides efficient resource isolation and sharing across distributed applications

- **Akka** - a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM

- **Cassandra** - distributed, highly available database designed to handle large amounts of data across multiple datacenters

- **Kafka** - a high-throughput, low-latency distributed messaging system designed for handling real-time data feeds

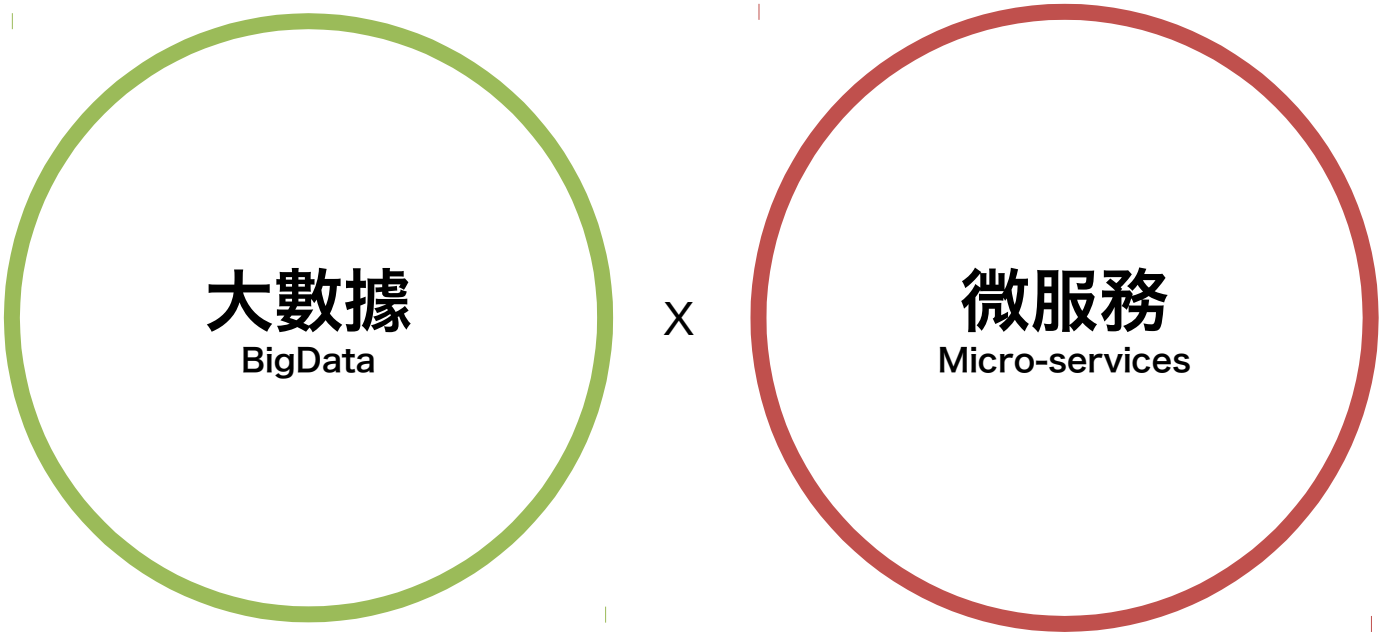
Ref: http://www.slideshare.net/akirillov/data-processing-platforms-architectures-with-spark-mesos-akka-cassandra-and-kafka#p4



Ref: https://medium.baqend.com/nosql-databases-a-survey-and-decision-guidance-ea7823a822d

我們很少在大數據架構中見到 RDBMS 的蹤影

但 Google/Facebook/Twitter/Uber/Alibaba ...


大數據
BigData
X
微服務
Micro-services

Hadoop (Java)

Spark (Java/Scala)

Cassandra (Java)

Kafka (Java/Scala)

Pig (Java)

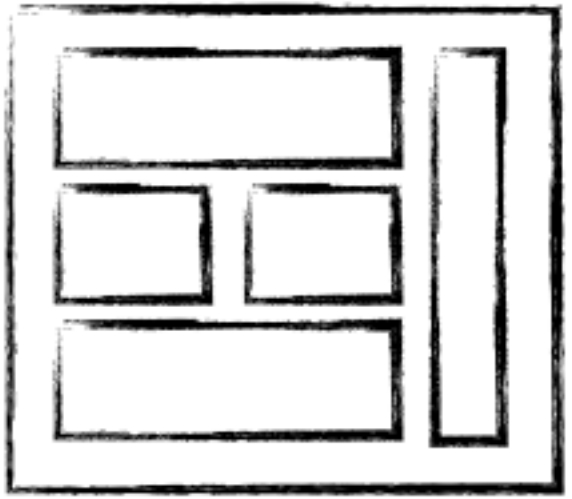
Hive (Java)

HBase (Java)

Flink (Java)

ElasticSearch (Java)

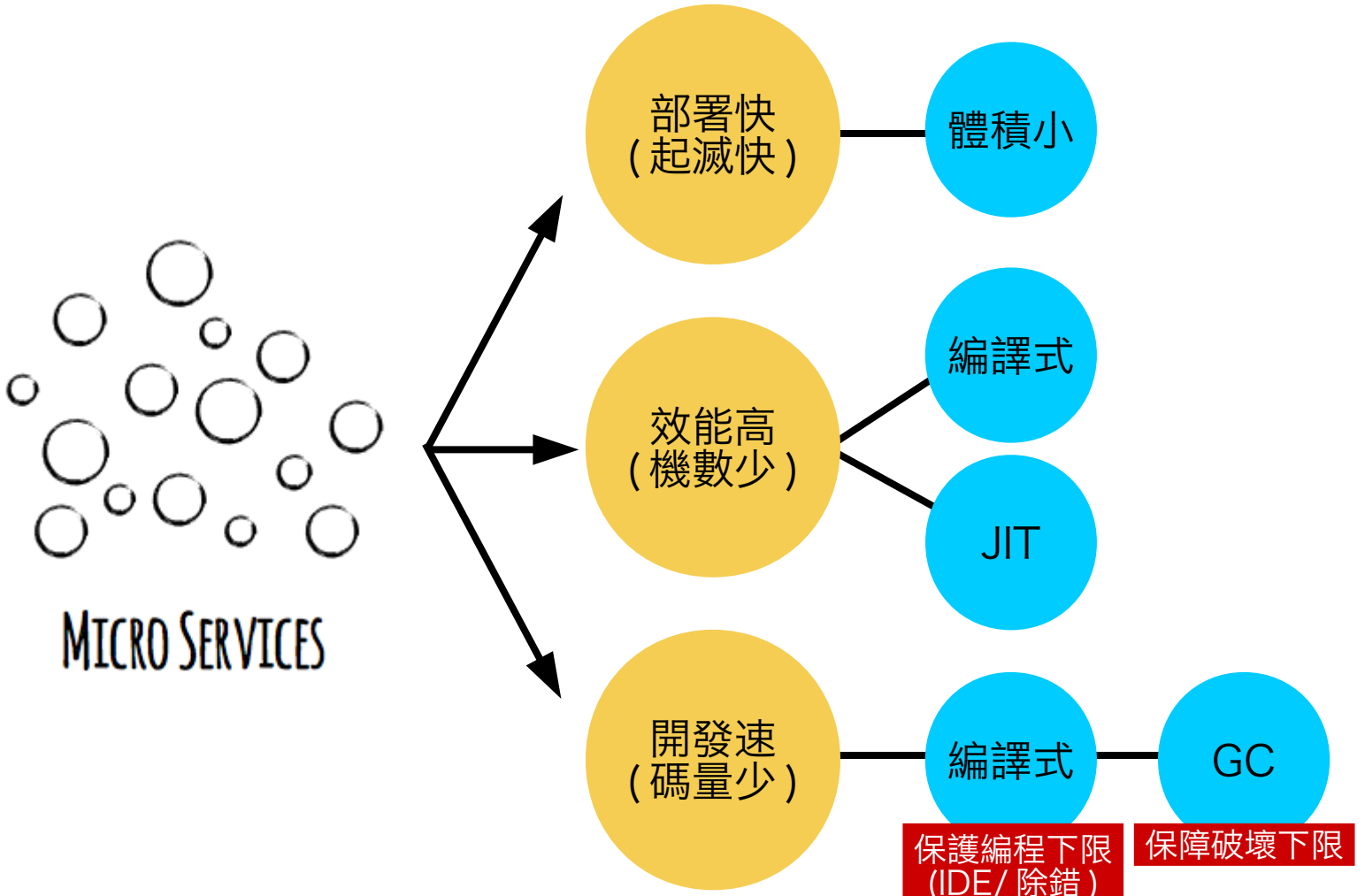
JVM 的天下

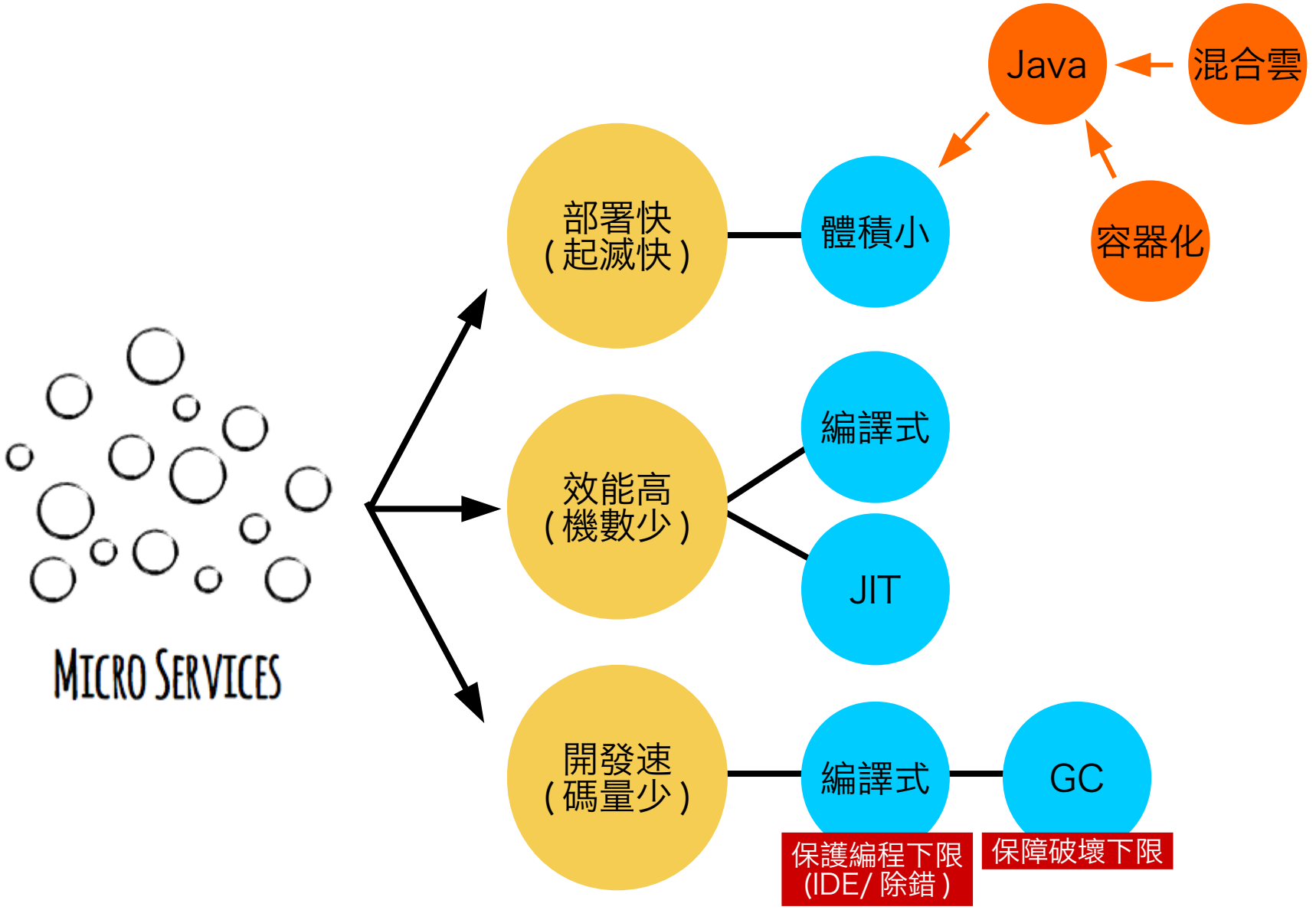


MONOLITHIC/LAYERED

MICRO SERVICES

Ref: http://eugenedvorkin.com/seven-micro-services-architecture-advantages/


部署快
(起滅快)
體積小
效能高
(機數少)
編譯式
JIT
開發速
(碼量少)
編譯式
GC
保護編程下限
(IDE/ 除錯)
保障破壞下限
MICRO SERVICES


Java
混合雲
容器化
部署快
(起滅快)
體積小
效能高
(機數少)
編譯式
JIT
開發速
(碼量少)
編譯式
GC
保護編程下限
(IDE/ 除錯)
保障破壞下限
MICRO SERVICES

Java

Container Size

Oracle JDK (~350 MB)

Oracle JRE (~70 MB)

Oracle Server JRE (~70 MB)

只是 JRE 本身

Alpine Java Docker Container ?

Java

Container Size

docker-alpine-java

<snip>

curl -jksSLH "Cookie: oraclelicense=accept-securebackup-cookie" ...

```
rm -rf /opt/jdk/*src.zip \
       /opt/jdk/lib/missioncontrol \
       /opt/jdk/lib/visualvm \
       /opt/jdk/lib/*javafx* \
       /opt/jdk/jre/plugin \
       /opt/jdk/jre/bin/javaws \
       ...
```

</snip>

Ref: https://github.com/anapsix/docker-alpine-java/blob/master/8/92b14/jdk/standard/Dockerfile

License

Business	Technology
License	Scale-up Application Connection Database File system OS Kernel Hardware
Elastic business	Scale-out Replication Clustering Sharding Disaster Recovery Multi Regional Resiliency
Workload	

and more ...

CONSILIENCE

Java

License

▌問題一：

Oracle/Java 安裝前需人工同意 Oracle/Java 授權。

▌問題二

Oracle/Java 為整體不可分割之授權。

<snip>

(i) you distribute the Redistributables complete and unmodified, and only bundled as part of Programs,

</snip>

Oracle Java 部署時，不得刪減任何程式及文件

Ref: http://www.oracle.com/technetwork/java/javase/terms/license/index.html

Java

License

▌問題一：

OpenJDK
(openjdk-7-jre-headless)

有些人不敢用

Oracle/Java 為整體下的分割之授權。

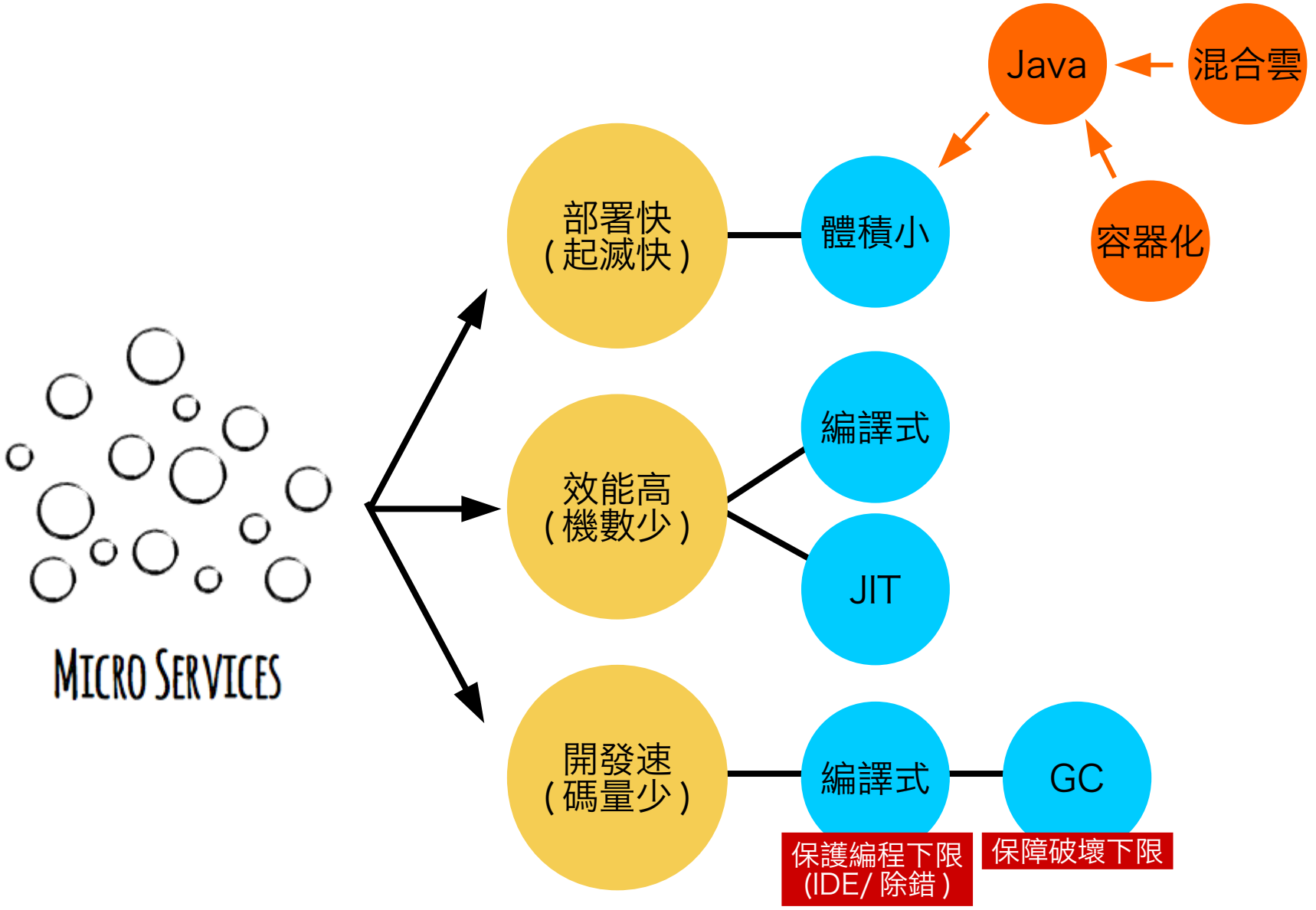
<snip>

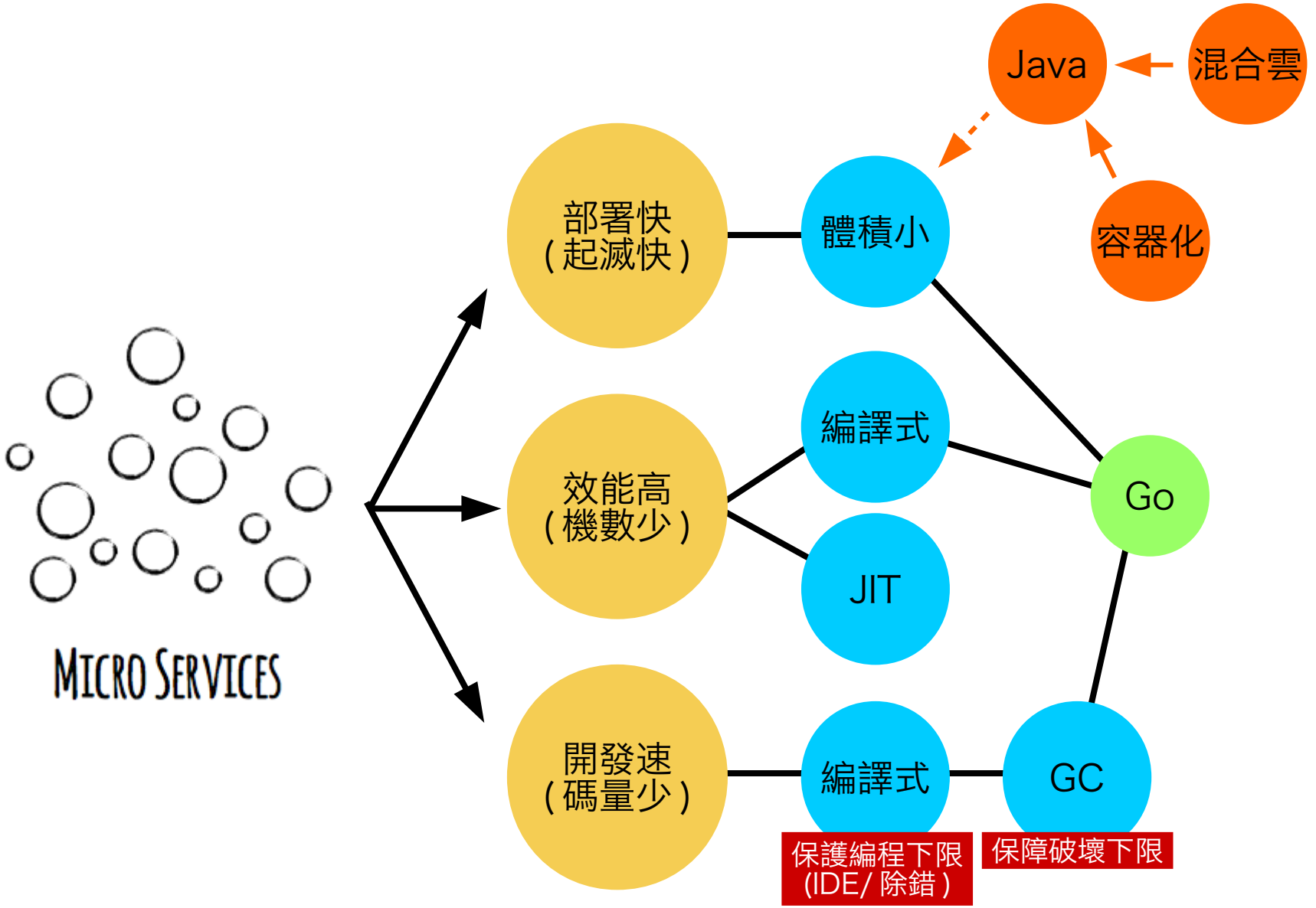
(i) you distribute the Redistributables complete and unmodified, and only bundled as part of Programs,

</snip>

Oracle Java 部署時，不得刪減任何程式及文件

Ref: http://www.oracle.com/technetwork/java/javase/terms/license/index.html


Java
混合雲
容器化
部署快
(起滅快)
體積小
MICRO SERVICES
效能高
(機數少)
編譯式
JIT
開發速
(碼量少)
編譯式
GC
保護編程下限
(IDE/ 除錯)
保障破壞下限

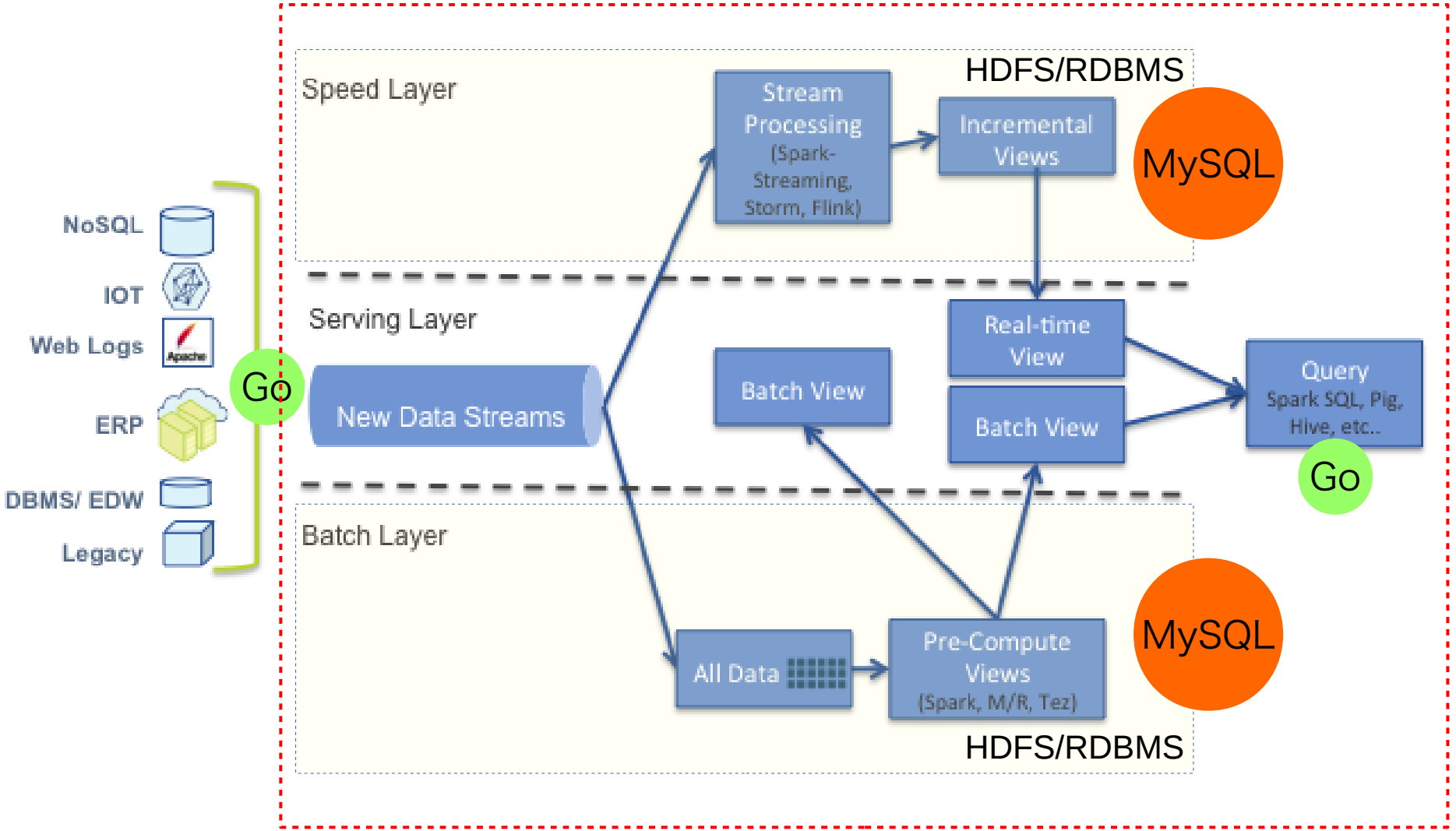

Java
混合雲
容器化
部署快
(起滅快)
體積小
編譯式
效能高
(機數少)
JIT
Go
開發速
(碼量少)
編譯式
GC
保護編程下限
(IDE/ 除錯)
保障破壞下限
MICRO SERVICES

小結

- 未捨棄既有累積的知識 (RDBMS)
- 降低開發與維運人員的焦慮感 (專注，技術深化)
- 系統異質性低 (出錯少，除錯易)
- 依然相容現行大數據架構
- 支持容器化、混合雲、私有雲 (顧問性質)
- 大數據核心與對外服務層權責分離 (兼具安全性)
- 其實大多時候我們不需要大數據解決方案

資料多≠需要大數據解決方案，或許只是垃圾數據多

處理慢≠需要大數據解決方案，大多都是程式差，架構弱


NoSQL
IOT
Web Logs
ERP
DBMS/ EDW
Legacy
Go
Speed Layer
HDFS/RDBMS
Stream Processing (Spark-Streaming, Storm, Flink)
Incremental Views
MySQL
Serving Layer
New Data Streams
Batch View
Real-time View
Batch View
Query
Spark SQL, Pig, Hive, etc..
Go
Batch Layer
All Data
Pre-Compute Views
(Spark, M/R, Tez)
MySQL
HDFS/RDBMS

1. 架構先決
2. 沒有完美的架構，只有最適的架構
3. 架構是演進的，預想但不過早調優

Contact

yftzeng@gmail.com

https://www.facebook.com/yftzeng.tw