BDTC
2014 中国大数据技术大会
BIG DATA TECHNOLOGY CONFERENCE
暨第二届CCF大数据学术会议

12306：改变传统思路解决问题的NoSQL 实践

人口大迁移

大陆发改委部署2013春运40天 预计发送34亿人次

-2013年春运从1月26日开始至3月6日结束，共计40天

-初步预测，春运期间全国旅客发送量将达到34.07亿人次，比2012年春运实际完成增长8.6%

-预估 铁路约2.25亿人次，增长4.6%
-道路约31.04亿人次，增长9%
-水运约4308万人次，增长1.5%
-民航约3550万人次，增长5.2%

2013年 春运热点

- 直击12306购票网站:5分钟车票抢购一空 - 2013-01-18 19:26 来源：新华网

- 热门车票 **网上1分钟就没了票; 66个电话未买到票; 质疑售票黑箱作业**

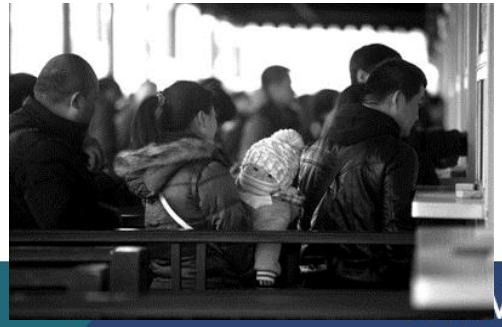
-**“农民工”一票难求，是购票网站12306惹的祸？** 2013-01-24 来源：中华铁道网

-**工信部叫停抢火车票软件：或致12306网站瘫痪**
新闻中心-中国网 news.china.com.cn 时间： 2013-01-19

- 抢票插件横行, 厂商受到铁道部约谈

-**“购票帝” 一票难求不解决, 还将写攻略，三年帮人订20万张票**

- **先抢票, 后退票, 退票率 ： 20%**



铁道部 12306 网上售票 大事记

2012 春运

-用户无法登陆
-无法车次查询
-无法下订单
-吞钱不吐票
-售票系统反应慢
- 余票信息更新慢, 有票买不到票
-售票系统瘫痪

2012 6月 → 启动余票计算和查询项目

2012 10/1 国庆

- 解决余票计算,查询和订单提交
- 订单处理缓慢, 限制在线人数10万人
- 订单提交后, 订单排队处理
- 等待时间 20分钟 – 2小时, 导致民怨

→ 启动订单快速查询项目

2013 1月元旦

- 部署订单快速查询功能，提高订单处理性能
- 限制在线人数 20万人
- 数据中心空调故障
- 12306网站 罢工 1.5天

→ 优化余票和订单查询系统

2013 2月春运

-系统部署成功
- 抢票插件横行,厂商受到铁道部约谈
- 先抢票, 后退票, 退票率： 20%

春运售票 秒殺現象

➢秒殺現象

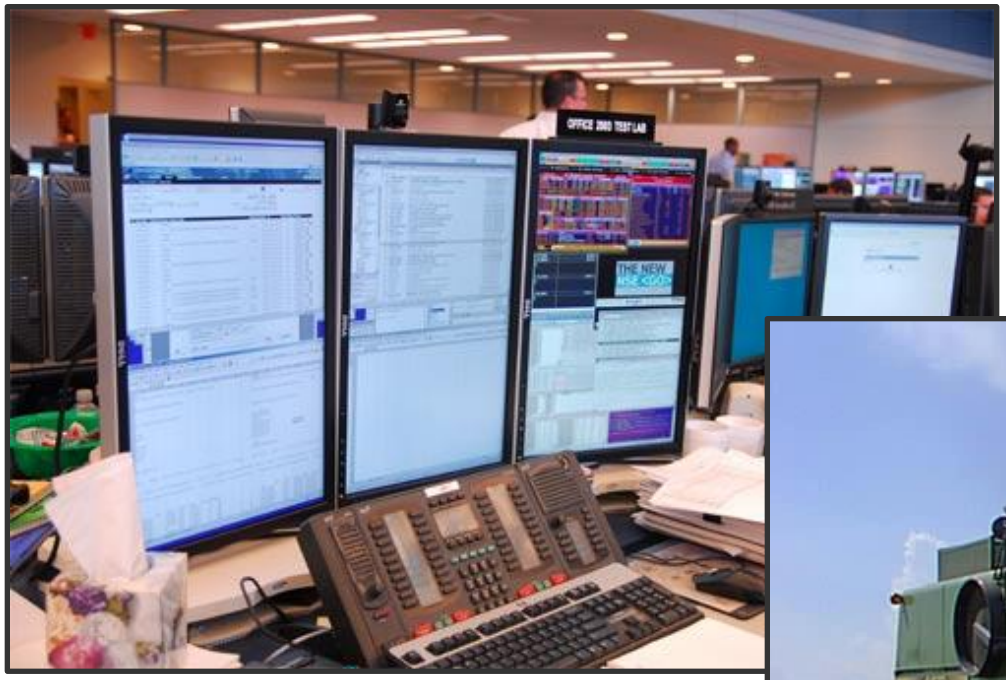
- 高峰售票:800萬張/天，13,000-20,000張票/分鐘
- 網上售票高峰:1700万人登陆，300萬張/天，3,000-5,000張票/分鐘
- 網上高峰點擊：15億次/天，1萬-3萬次/秒
- 23000售票點，1600自動售票機，電話訂票:10萬8千條線
- 網路頻寬：1.5G/秒

售票比例：互联网售票占**35.9%**，电话订票占**7.5%**，人工窗口占**52.1%**，自动售票机占**4.5%**

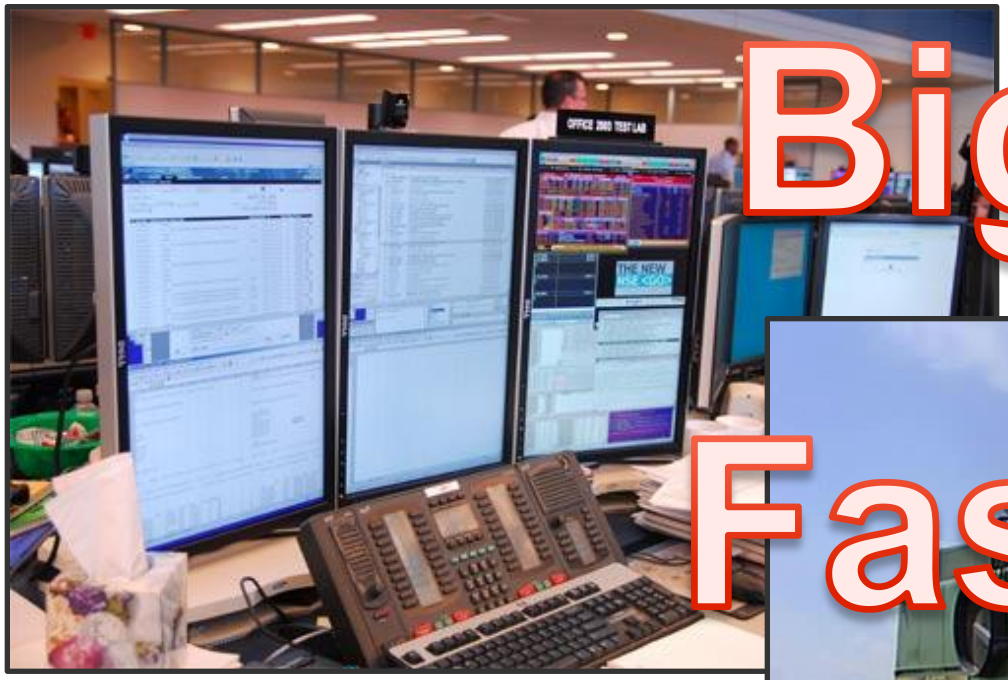
➢衍生問題

- 使用者無法登陸，或登陸後系統反應非常緩慢
- 提交訂單後，系統反應慢或沒有反應
- 吞錢不吐票
- 訂票的餘票查詢 每十分鐘更新一次，在售票高峰期餘票資料沒有參考價值，資訊更新慢，民眾訂不到票，導致民怨。

What's Common?



What's Common?



Big

Fast

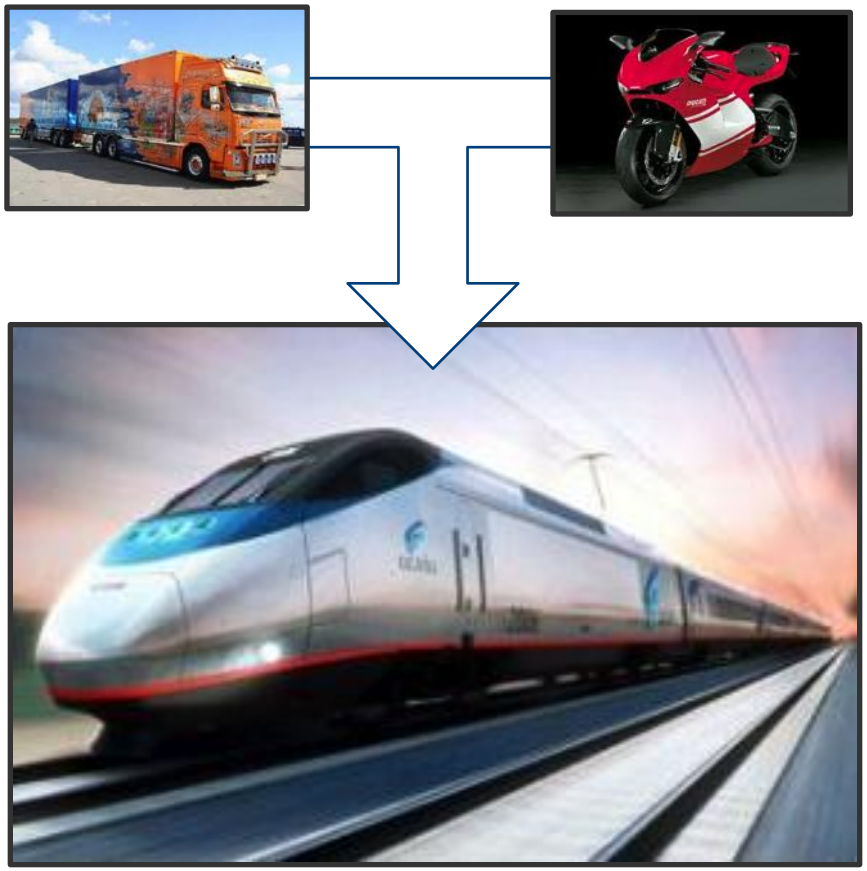
Flexible

Fast Data Meets Big Data

Big Data allows you to find opportunities you didn't know you had

Fast Data allows you to respond to opportunities before they are gone

Fast Data Meets Big Data



Working together they enable entirely new business models

The Database is Being Stretched

Big Data

- Petabytes vs. Gigabytes
- Democratize BI

The Database is Being Stretched

Fast Data

- Low latency expectations
- Horizontal scale

Big Data

- Petabytes vs. Gigabytes
- Democratize BI

The Database is Being Stretched

Fast Data

- Low latency expectations
- Horizontal scale

Big Data

- Petabytes vs. Gigabytes
- Democratize BI

Flexible Data

- Multi-structured data
- Developer productivity

The Database is Being Stretched

Fast Data

- Low latency expectations
- Horizontal scale

Big Data

- Petabytes vs. Gigabytes
- Democratize BI

Flexible Data

- Multi-structured data
- Developer productivity

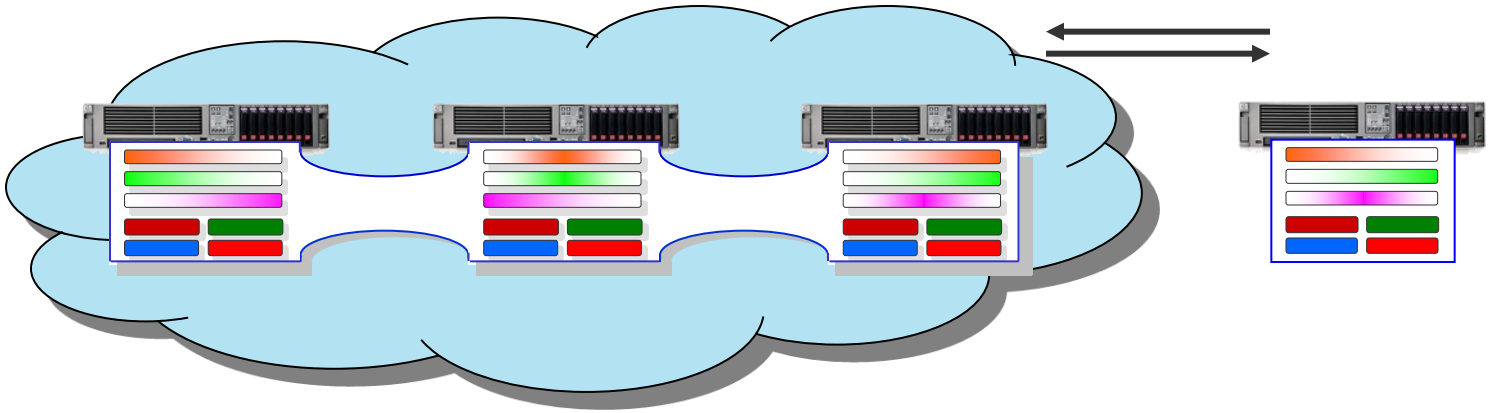
Cloud Delivery

- Virtualized
- Offered “-as-a-Service”

Need a Horizontally Scalable, Elastic Data Management Solution

Elastic

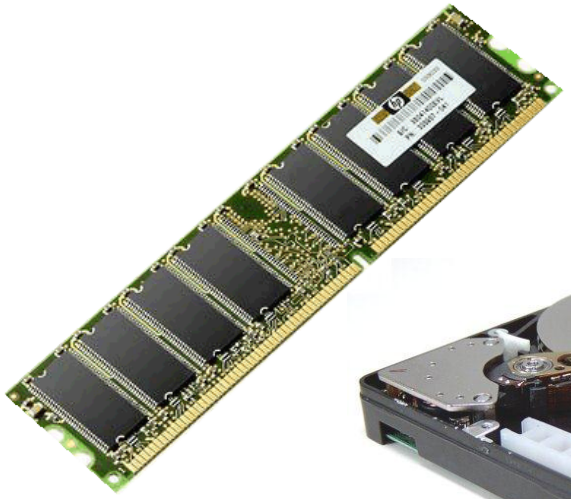
Add/remove data servers dynamically

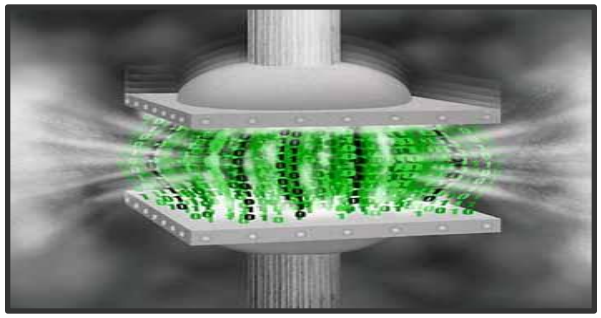


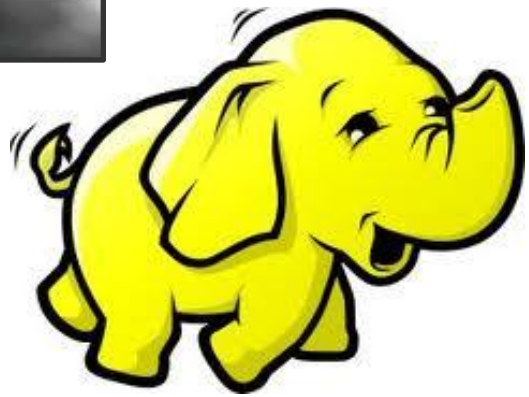
Grow or shrink **dynamically**

with no interruption of service or data loss

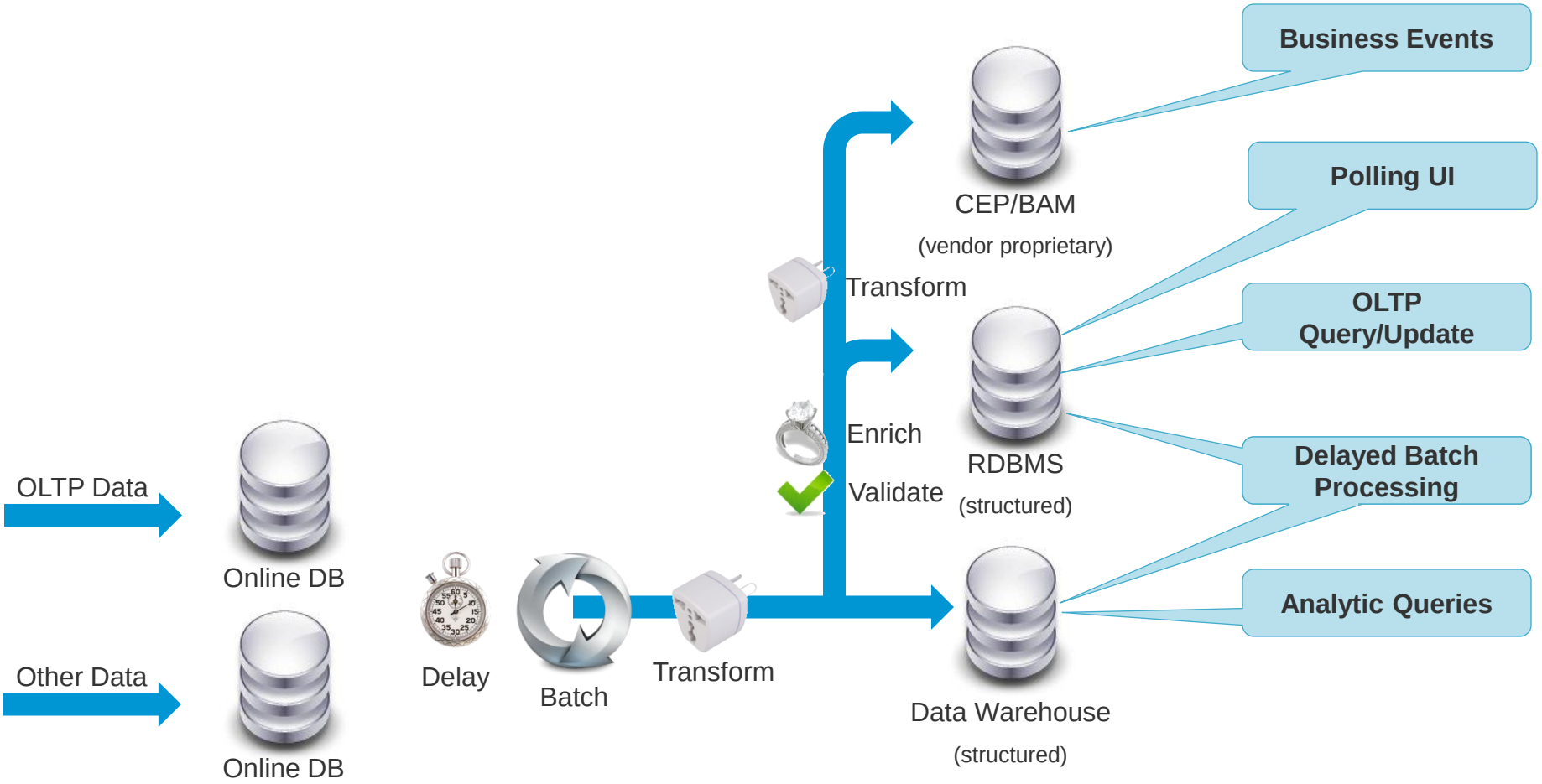
Tiered Data Strategy



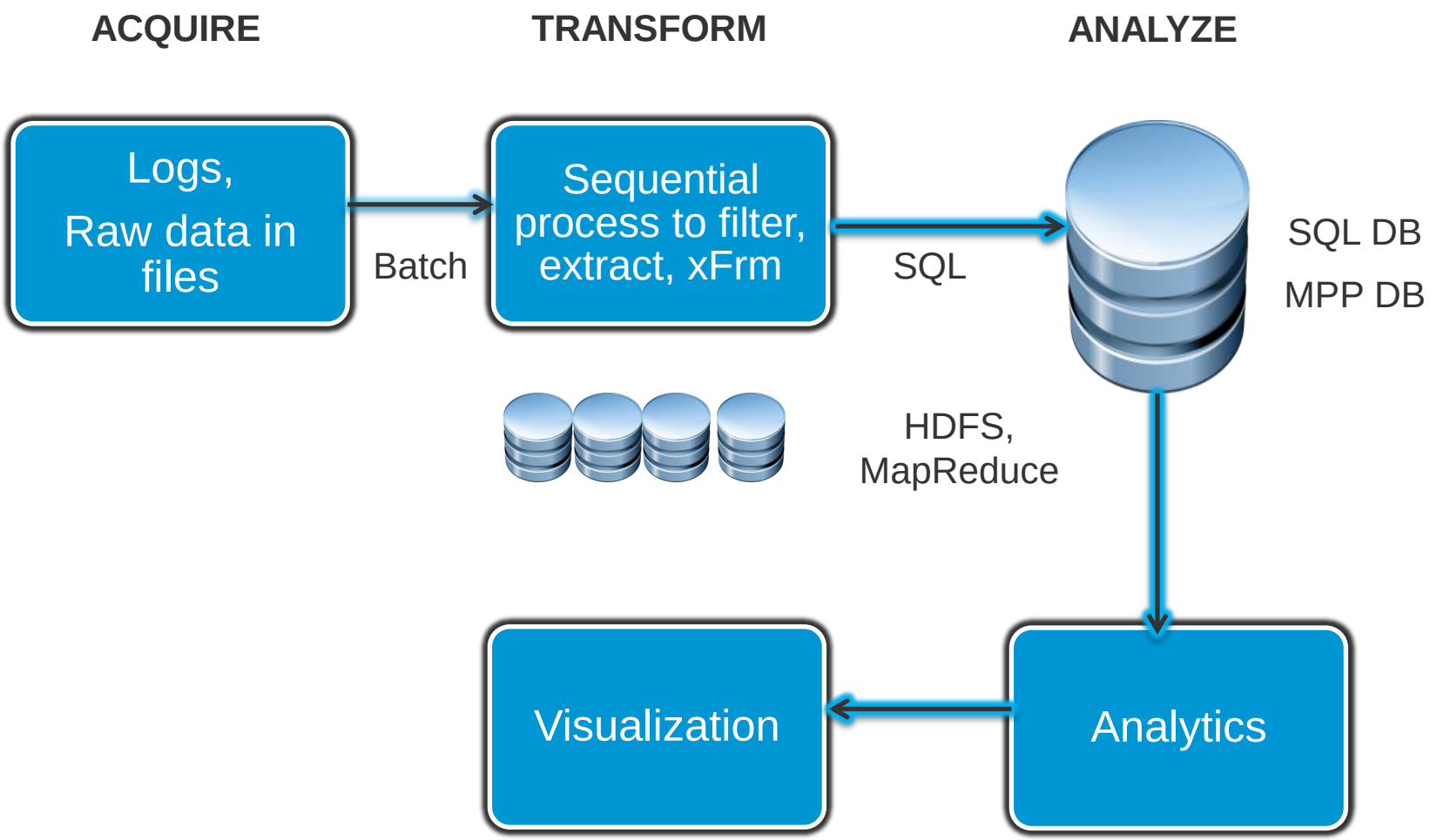




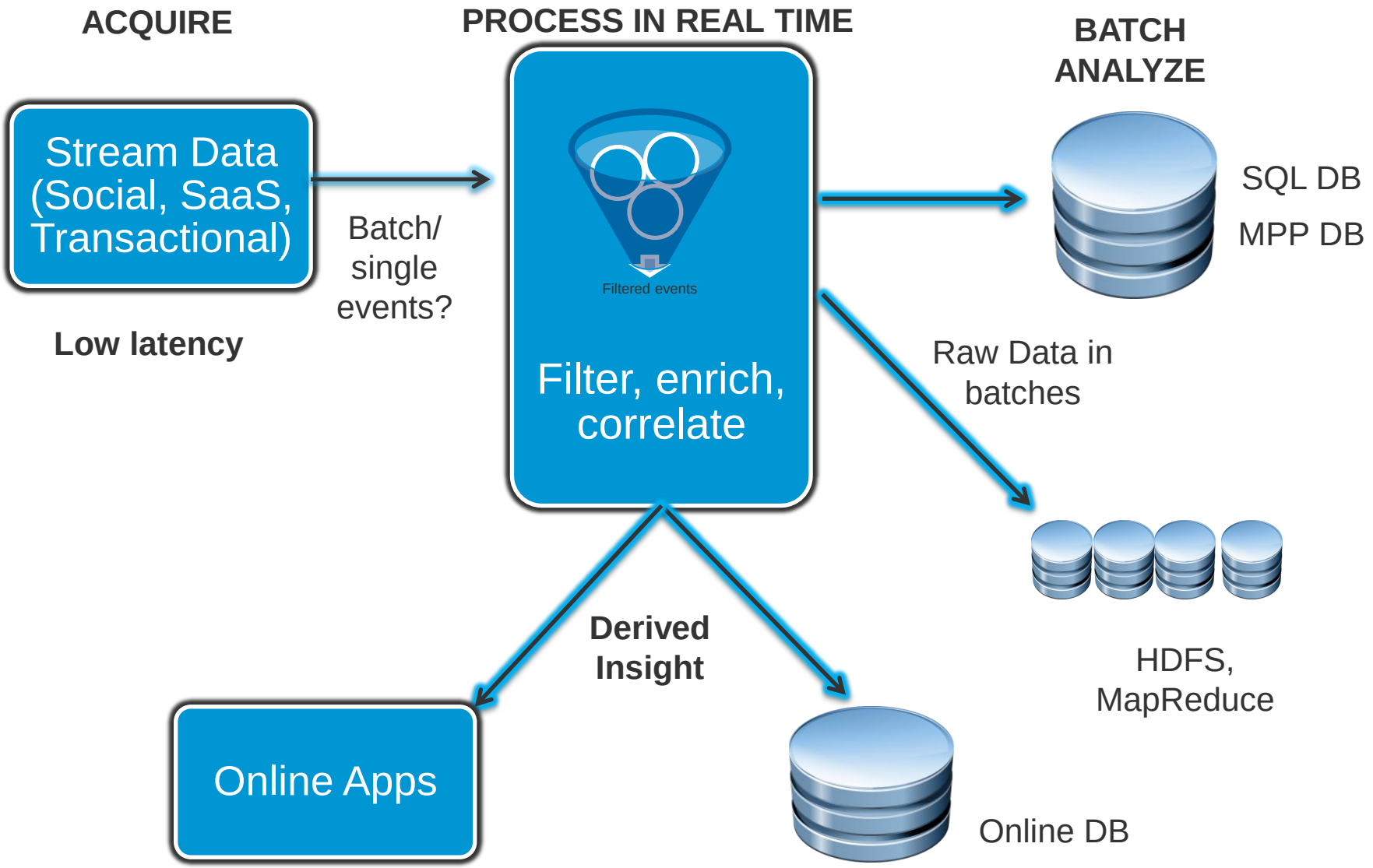
Looking Back – Traditional Batch Analytics



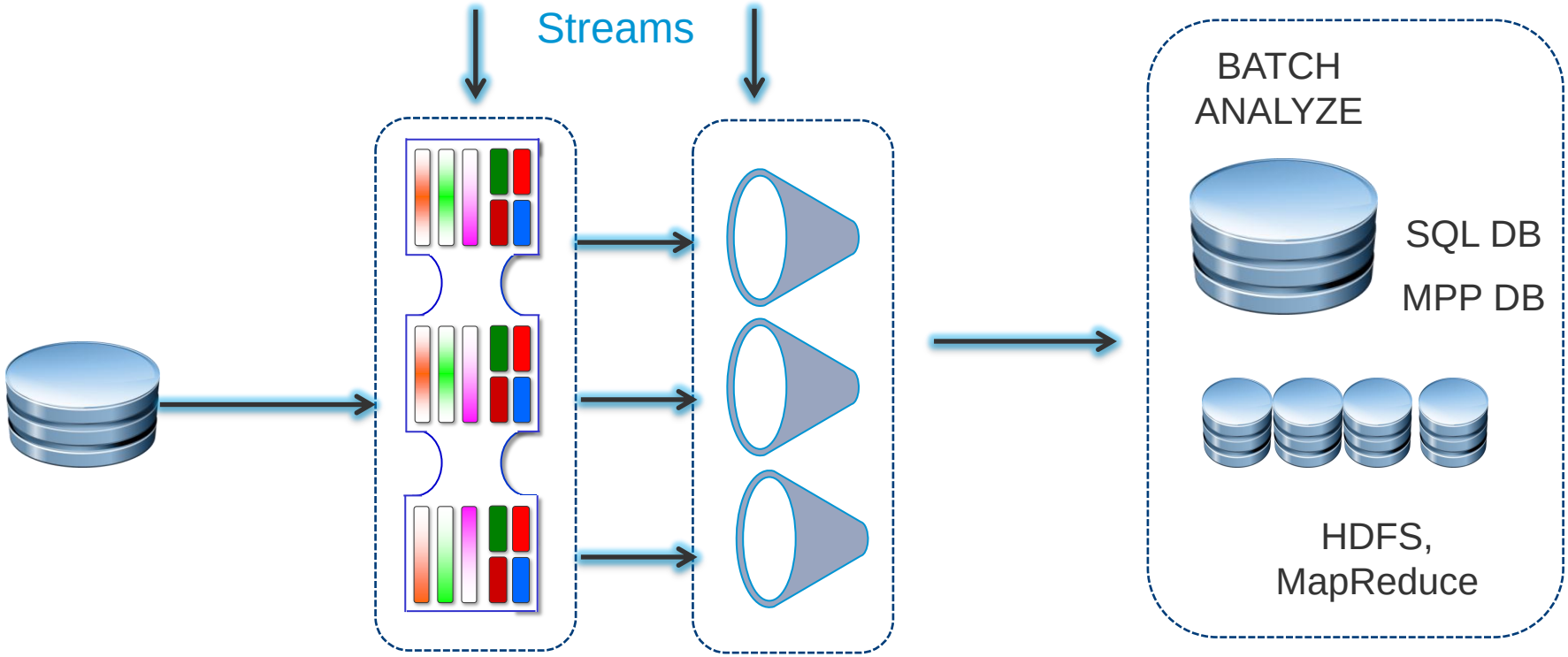
Pipeline with Hadoop (Log Analytics)



The Real-time Pipeline



New Architecture for Real-time (Custom)



In-Memory Data Grids

- Buffer data, process events, In-memory Map-reduce

  (VMWare GemFire, SQLFire, Oracle Coherence, etc.)

Stream Processing

- Derive insight with continuous event processing

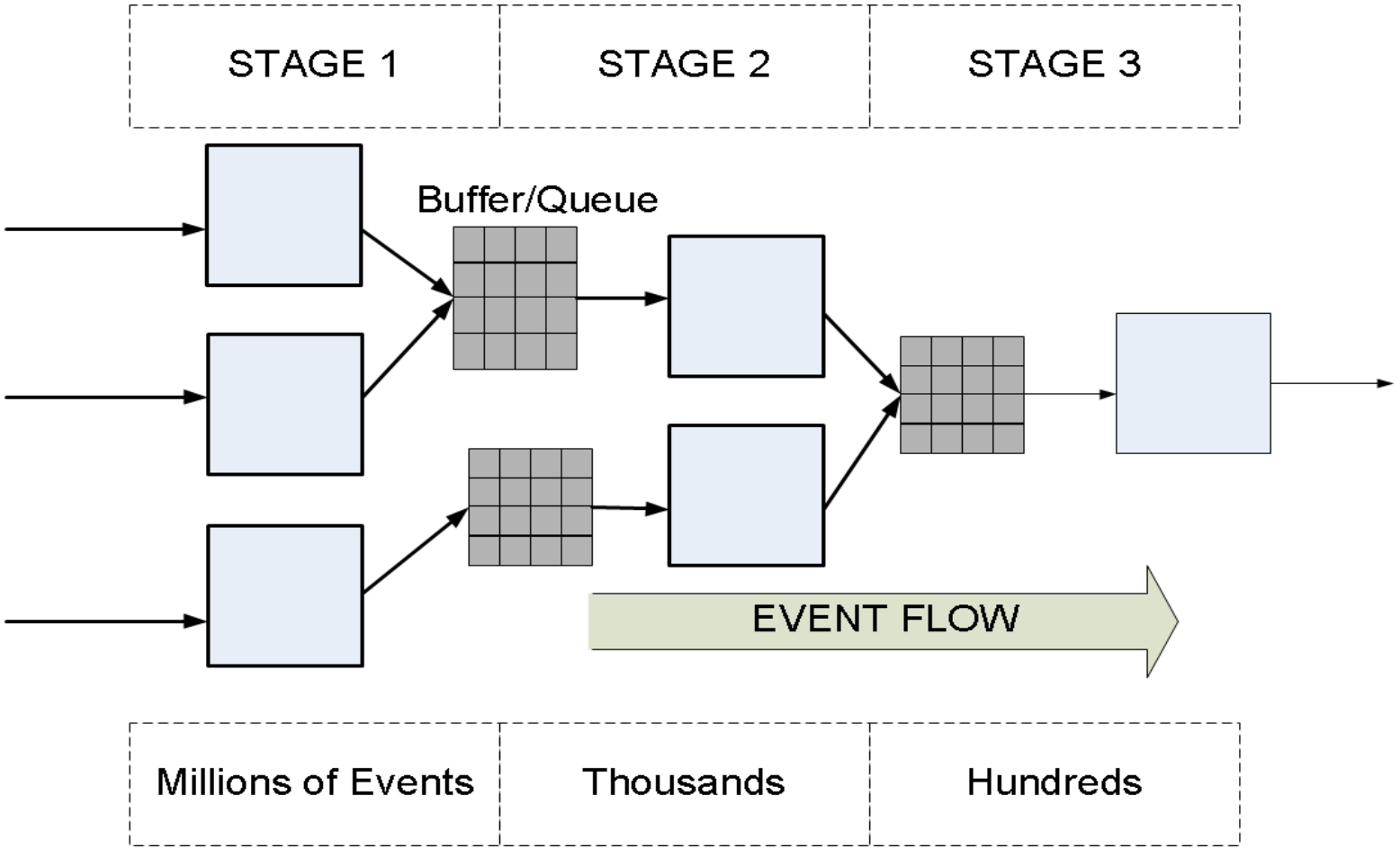
  (Apache S4, STORM, Esper, StreamBase, GemFire)

What Principles Drive This Architecture?

- **Very low latency ingest, high scalable (write scalability)**
  - Support structured as well as unstructured
- **Real-time processing cannot throttle incoming stream(s)**
  - Highly parallelizable with minimum IO (network and disk)
  - Be elastic
- **Cannot lose events else derived value is questionable**
- **Post processing Raw, derived events (batch analytics)**

What Principles Drive This Architecture?

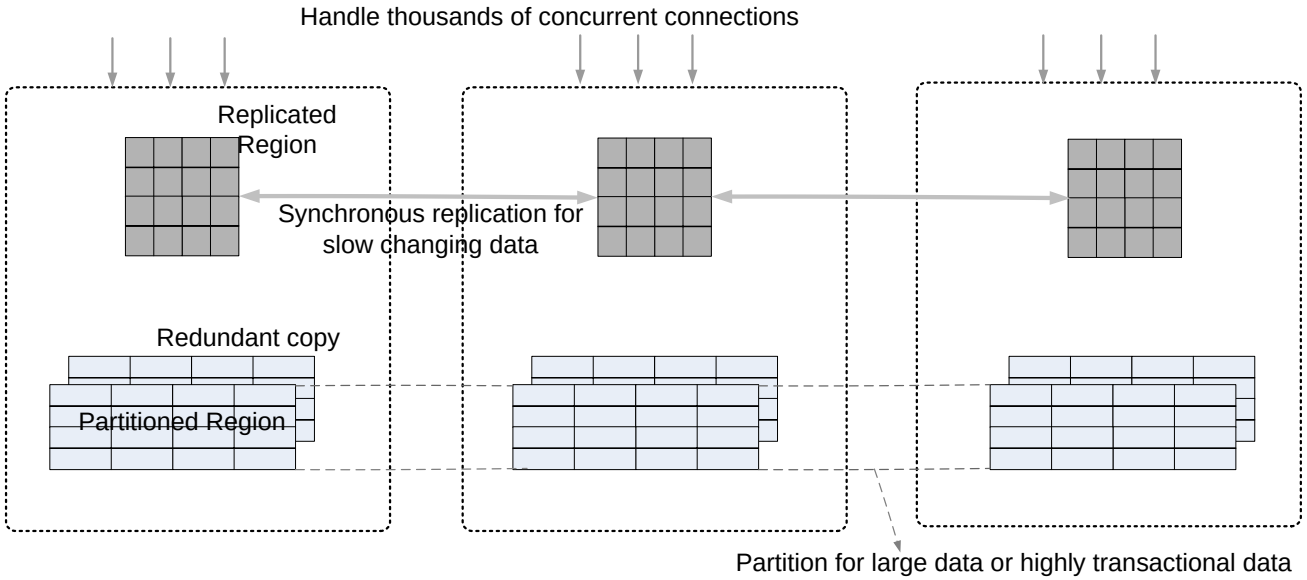
- **STAGE to SCALE**
  - Staged Events Driven Architecture



Acquire, Transform, Filter (Fast Ingest)

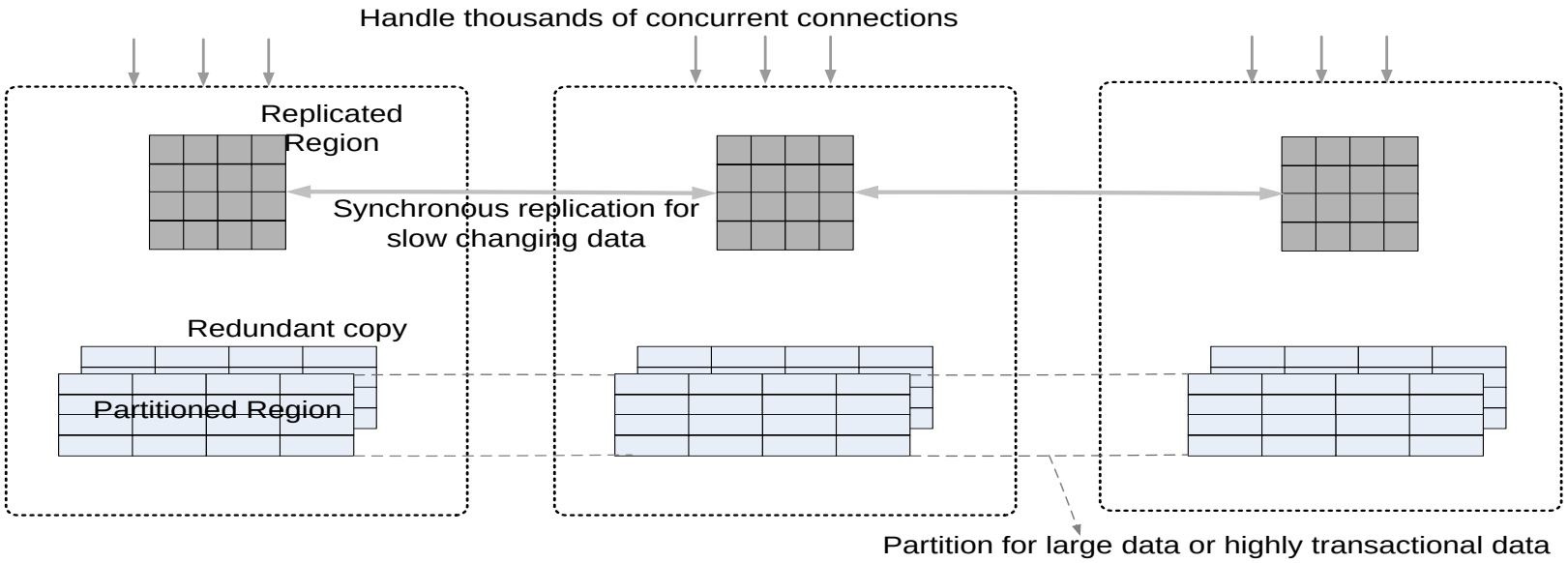
In-memory Data Grid Concepts

- **Distributed memory oriented key-value store**
  - Queriable, Indexable and transactional
- **Distributed namespace of Maps (key-value)**
  - Called “Regions” (GemFire, Hibernate), “Cache”(Oracle), etc.
  - 2 key storage models: Replication, Partitioning



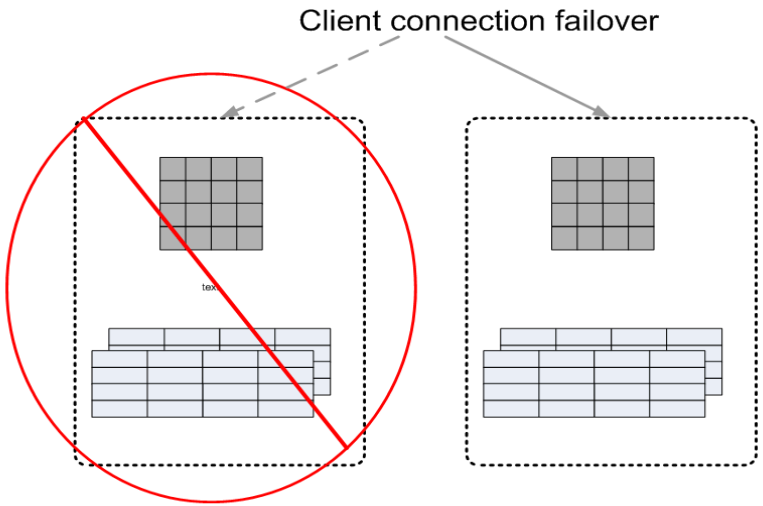
Acquire, Cache, Transform, etc.

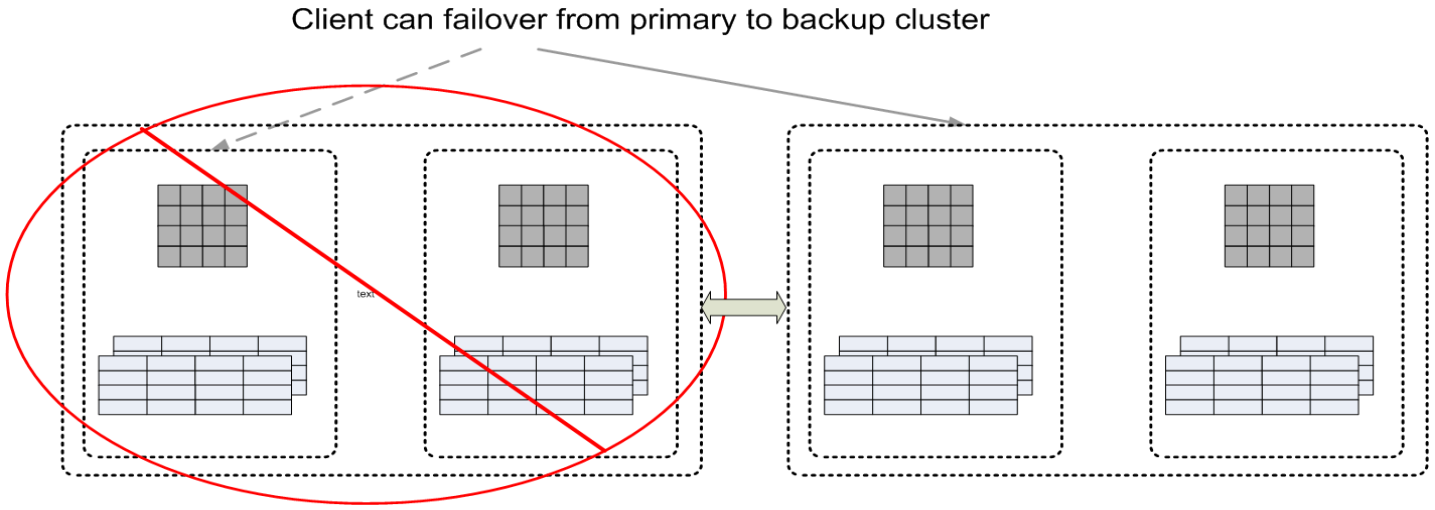
- **High ingest partitioned buffering**
  - Expiry based on TTL, idleTime
  - Windows – count, heap size, LRU eviction
- **Works with rigid or flexible Schema (JSON, Objects, SQL)**
- **Cache frequently used DB data for transform, massaging**
- **Partitioned listeners for filtering, event transform, etc.**



24

Continuously Available

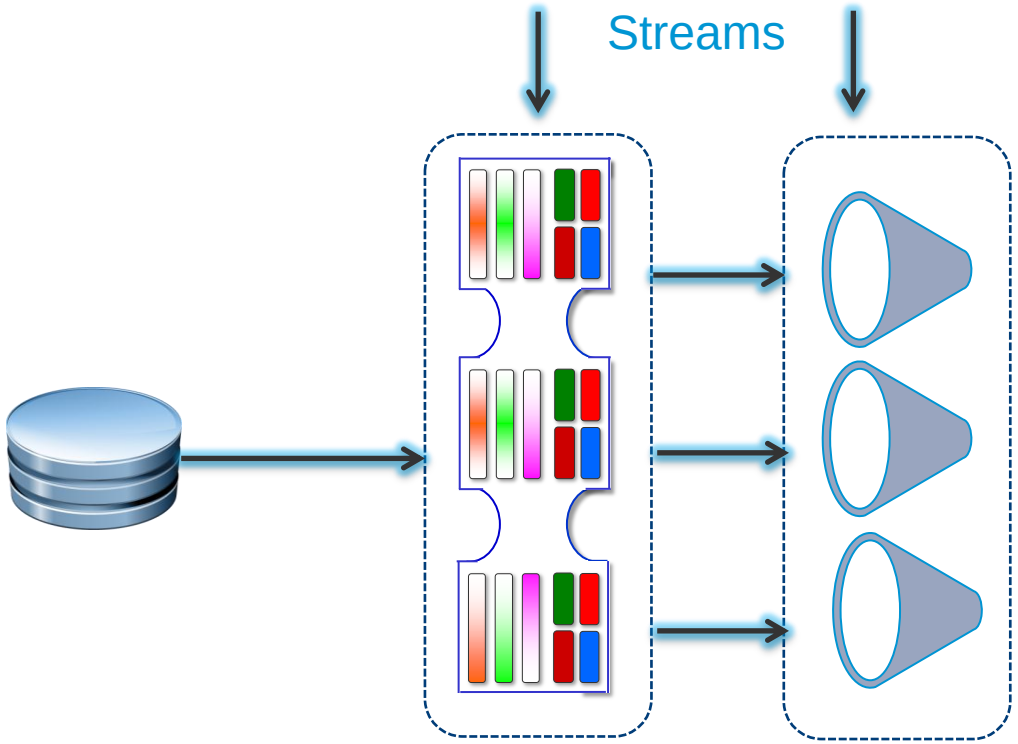




25

Complex Stream Processing

Continuous Filtering Using CQs



Apps subscribe to streams using Queries:

Select * from Tweets where tweetCount > 10 and userId in (X,Y,Z)

Distributed Processing

When data changes, subscribers are pushed Async events reliably

-All related data is accessible at memory speeds

CQ: Continuous Queries

27

Using CEP, S4, STORM

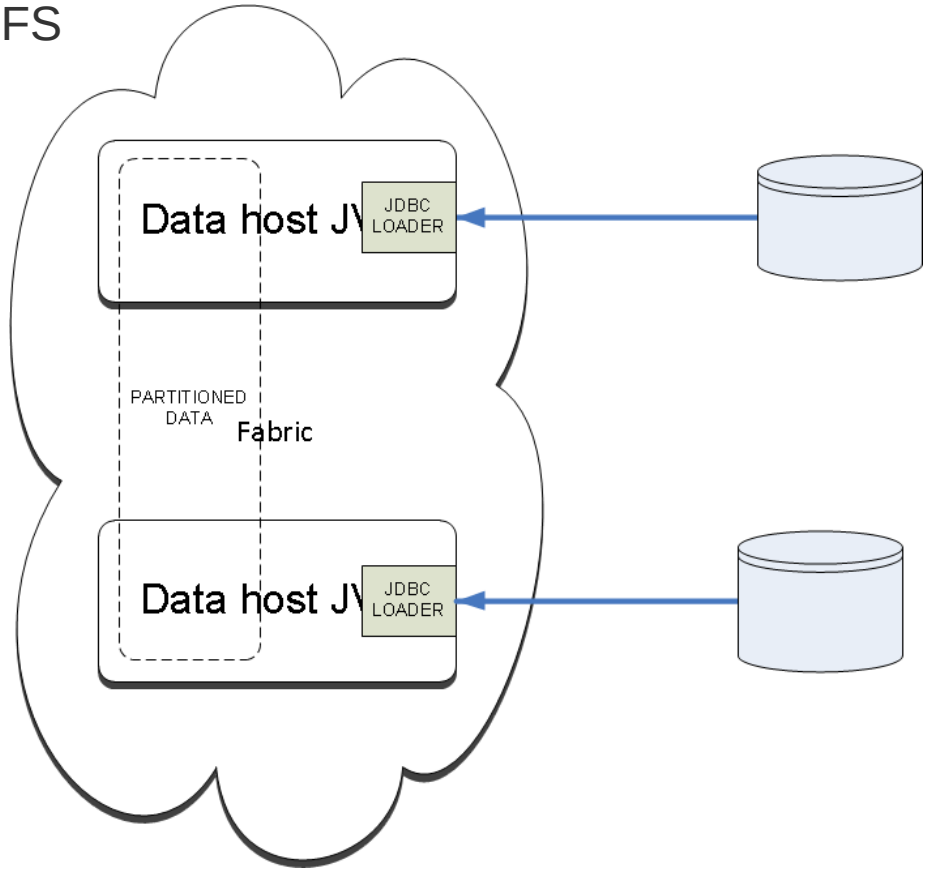
- Distributed framework for unbounded streams
- Custom App processing code that filters, routes, joins multiple streams
- Main proposition is horizontally, elastically scalable processors
- Simple config model to create processing pipelines

S4 distributed stream computing platform

Correlate, Joins with Data

Accessing Historical Data in Real-time

- **Correlations/Joins with History**
  - Option 1) Keep history in memory
  - Option 2) Keep in MPP DB (Greenplum DB)
  - Option 3) In HDFS

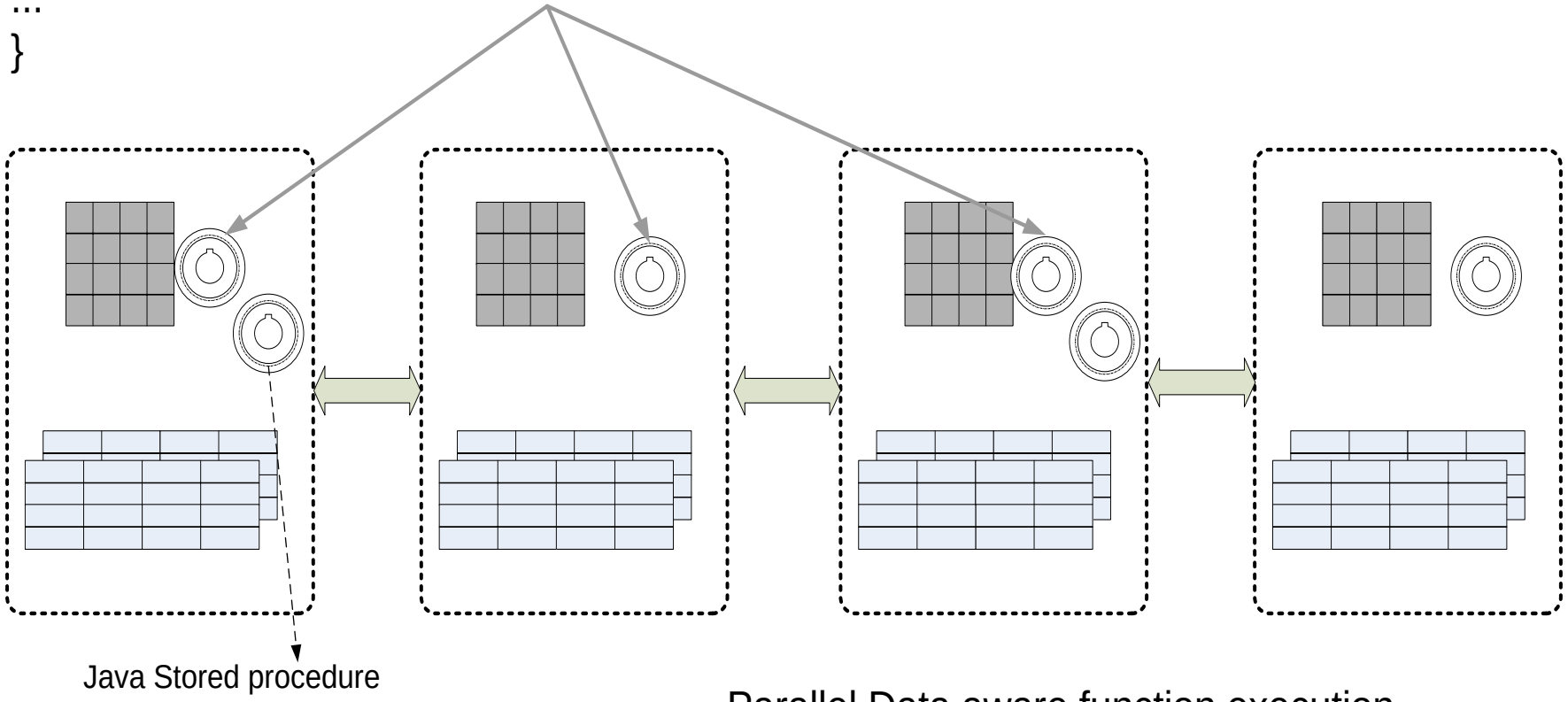


Real-time Aggregations with Parallel Processing

Map-reduce for Real Time

Hadoop M/R is for sequential batch processing and not for real time

```
@DistributedFunction(regionName="trades")
public List AnalyzeTrades(@FilterKey Set<String> months, String portfolio) { …
...
}
```

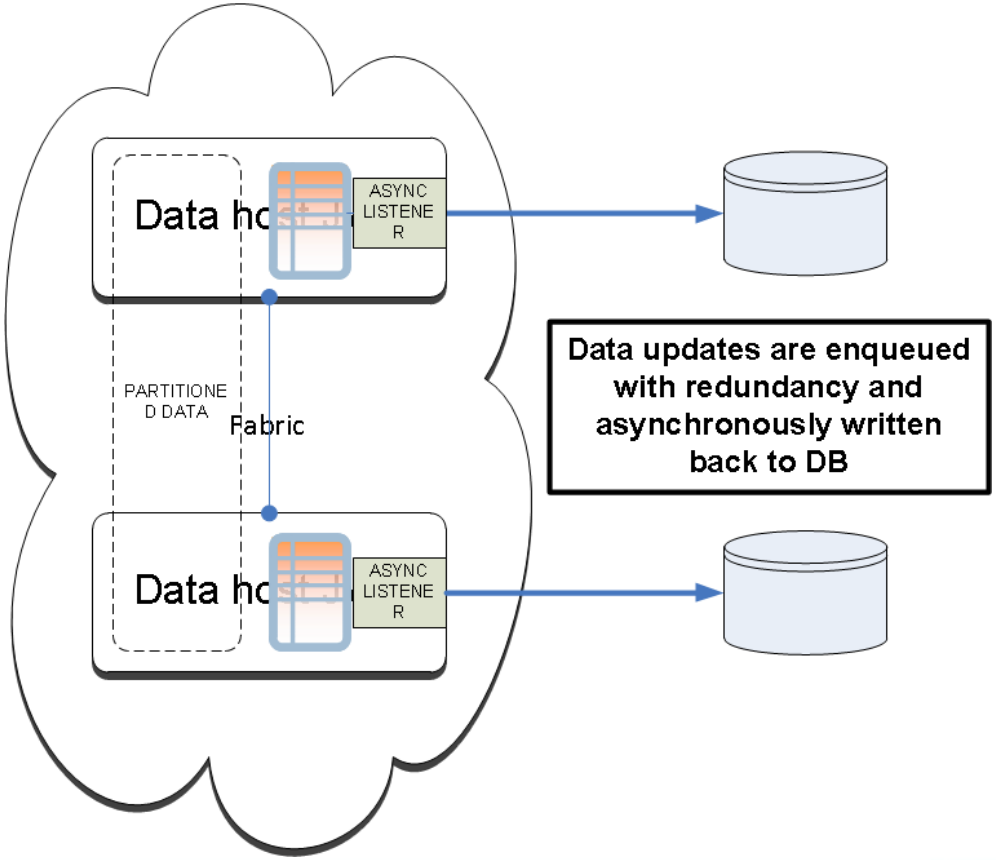


Parallel Data aware function execution

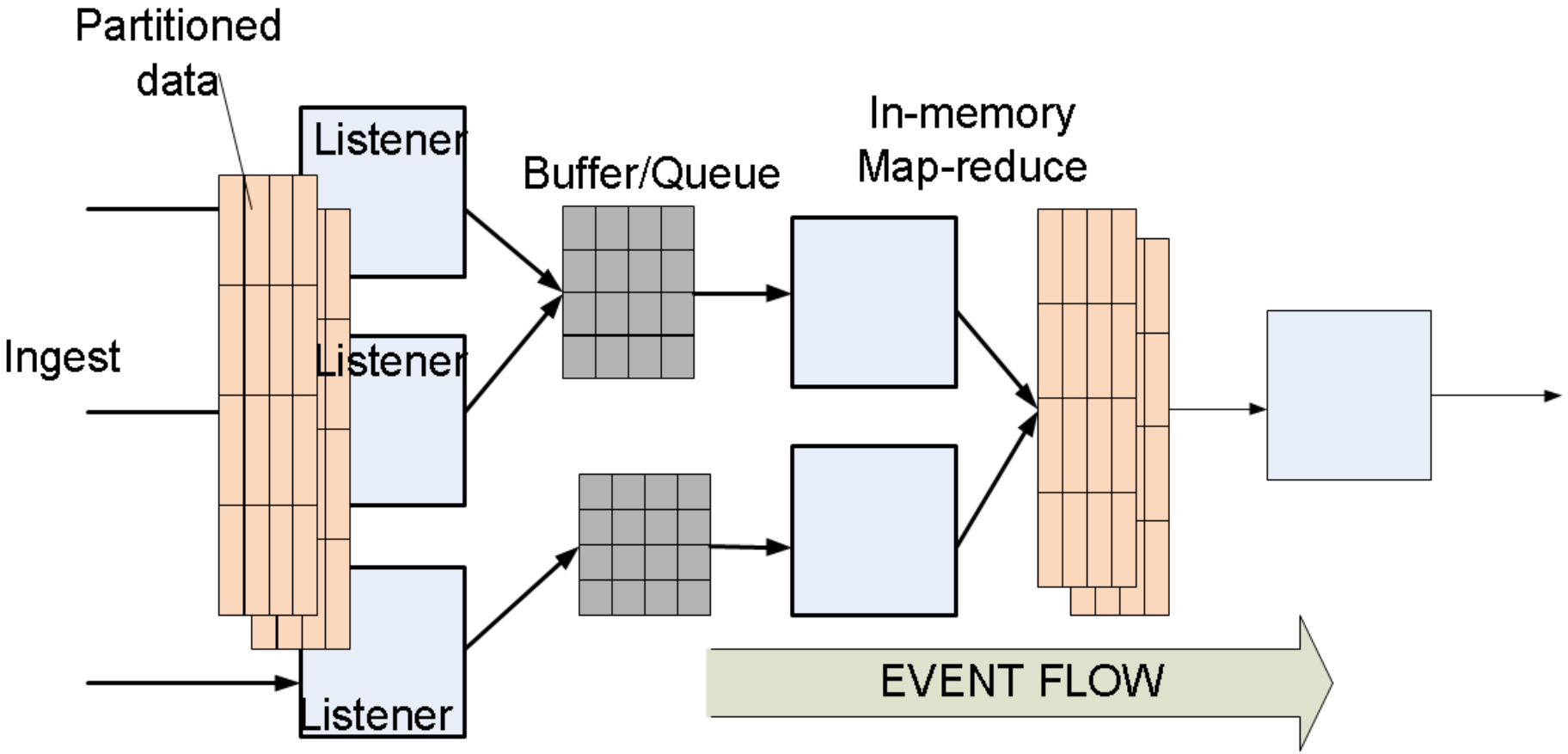
Ingestion to Hadoop, MPP DB

Enabling Batch Analytics with “Write Behind”

- **Store all raw, derived data in Hadoop**
- **Async, parallel write behind from data grid into HDFS**
  - Each partition writes batches in parallel for max throughput



Staged Pipeline – bringing it all together



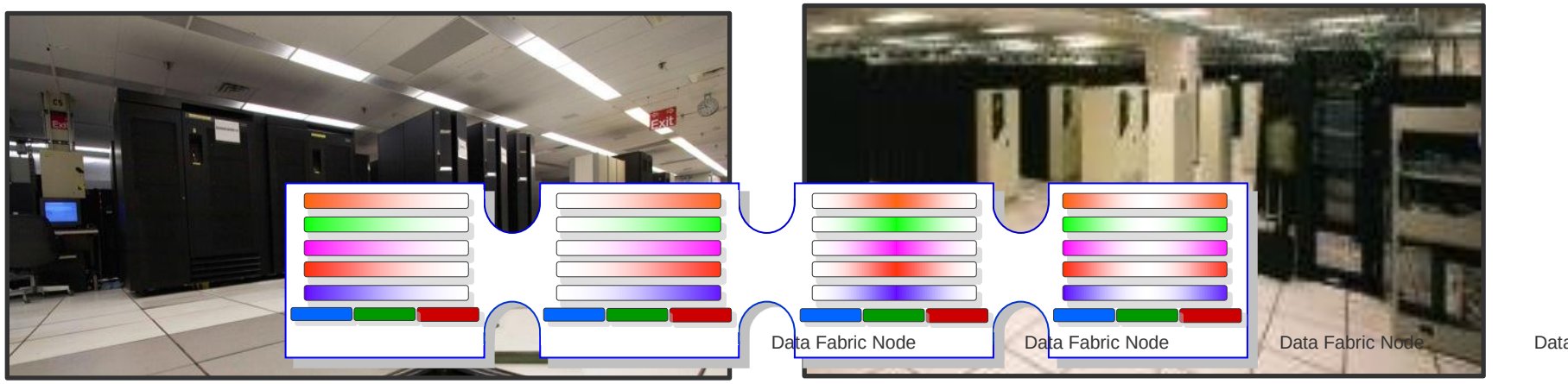
Use Spring Integration to orchestrate the pipeline

– Patterns: Pub-sub, splits, routers, Xfrm, etc.

Distribution of Analytic Results

Multi-Site Capability

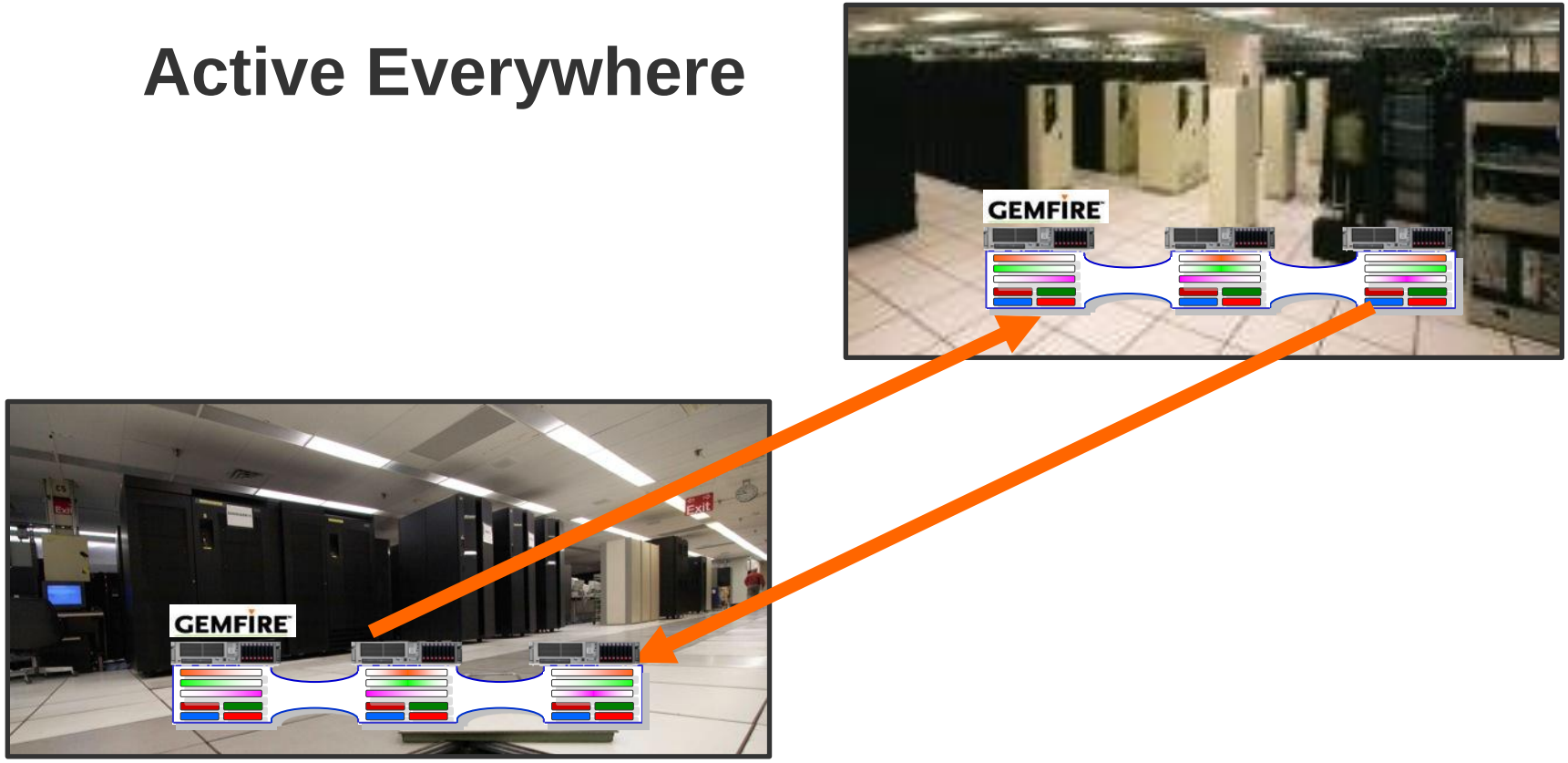
Active Everywhere



Single Cluster Spanning Data Centers

Multi-Site Capability

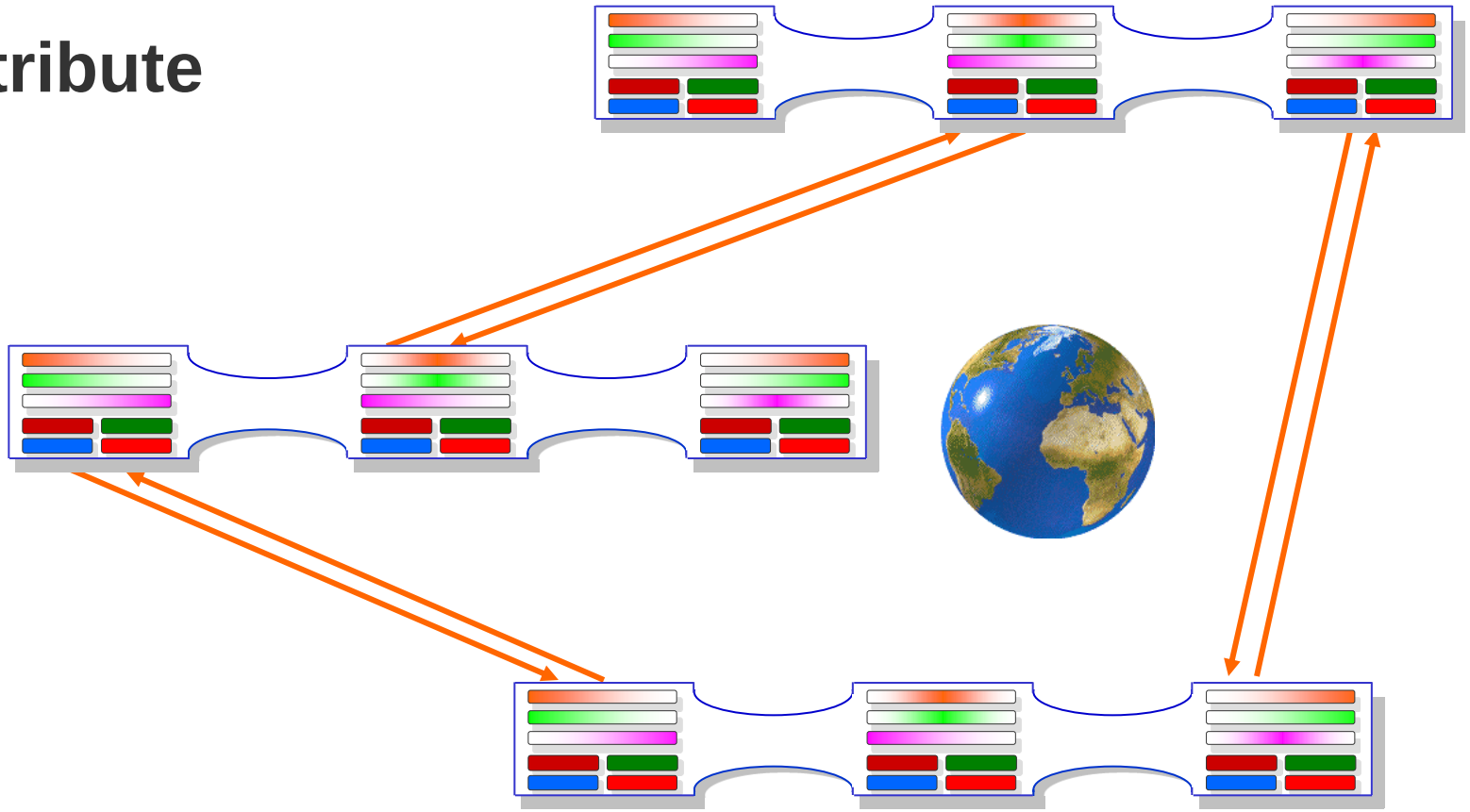
Active Everywhere



Asynchronous, Fault Tolerant, Bi-Directional WAN Gateway

Global Data Distribution

Distribute



GemFire can keep clusters that are distributed around the world synchronized in real-time and can operate reliably in Disconnected, Intermittent and Low-Bandwidth network environments.

Bringing It All Together…What Would It Look Like?

Existing technologies …
…working together

客戶面临的難點

1. 原有系統運行在xxxxxx数据庫的存儲過程上(Store Procedure)。經過數年優化已達最佳情況

==》**优化“传统设计的系统架构”已无法解决所面临的问题**

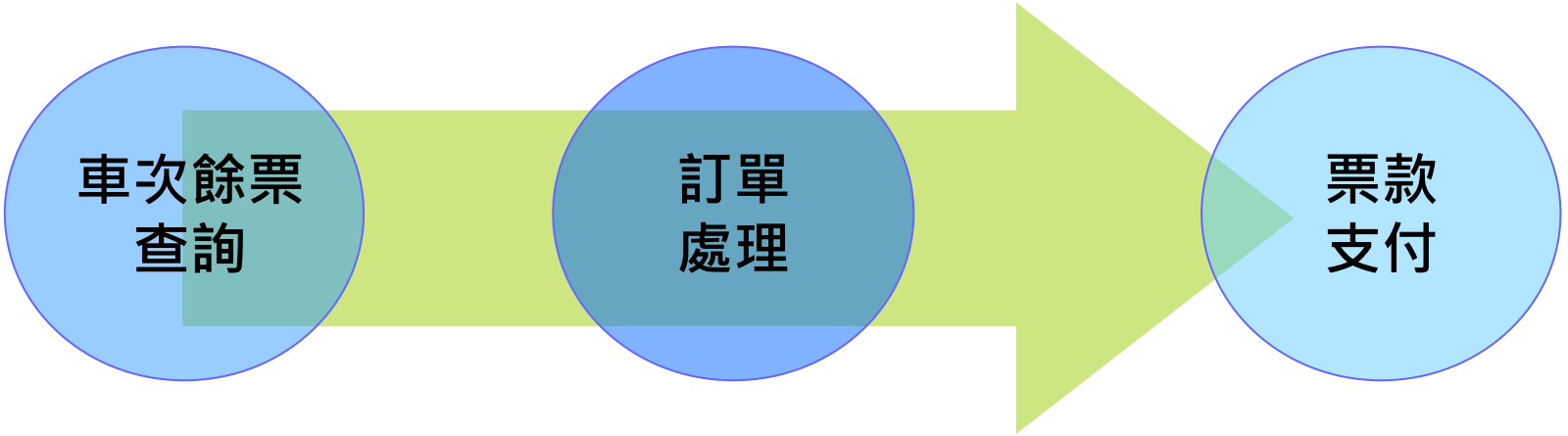
2. 票量餘額計算邏輯非常複雜，任何票數變化會導致票量餘額計算匹配規則變化。一張票的變化會導致上千種匹配規則

==》**业务逻辑计算复杂，无法提供快速的反应时间**

3. 餘票數量變化非常頻繁，目前使用xxx数据庫的同步伺服器，同步分支單位的“即時資料”到總部”中心数据庫”，速度可以達到“秒”級的同步效率。

==》**数据库读/写频繁, 在海量并发请求下导致系统崩溃.**

訂票系統 3大瓶頸



查詢合適的車次和席位

訂單處理和訂單查詢

信用卡，銀行卡支付

客戶面臨的挑戰

挑戰 1：在原有系統架構，採用雲應用虛擬化技術改造

- 在不改變訂票系統原有設計架構，採用“雲應用虛擬化”軟體技術。系統改造後必需具有“簡單配置”管理的水準擴展能力，按需隨時可以彈性調整伺服器集群數量， 滿足動態需求

挑戰 2：系統性能

- 採用“雲應用虛擬化”軟體技術，整體系統效率提高10倍以上
- 併發查詢：訂票的餘票查詢性能要達到處理每秒10,000次以上的併發查詢
- 余票更新：余票更新頻率能在1分鐘之內

挑戰 3：系統安全和穩定性

- 改造後的查詢結果應保證與原有系統一致的查詢準確度
- 提供HA（熱備份）和資料安全存儲

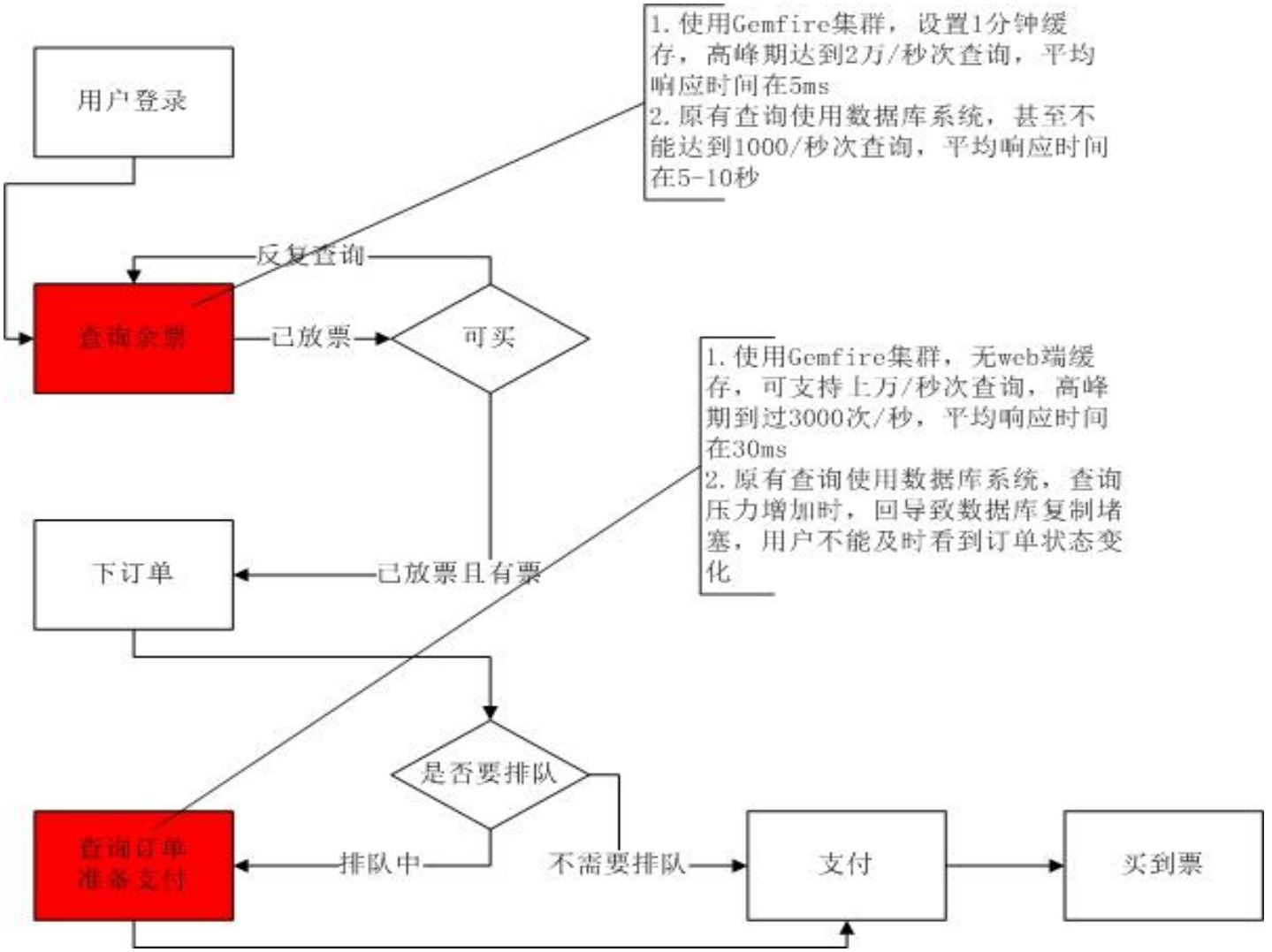
挑戰 4：性價比

- 使用x86伺服器+VMware在提供相同性能的基礎上降低成本

網上訂票系統架構改造方案

數據分流(Data Off-loading)-雲應用虛擬化技術（Unix to Linux）

網上訂票流程


用户登录
1. 使用Gemfire集群，设置1分钟缓存，高峰期达到2万/秒次查询，平均响应时间在5ms
2. 原有查询使用数据库系统，甚至不能达到1000/秒次查询，平均响应时间在5-10秒
反复查询
查询余票
已放票
可买
1. 使用Gemfire集群，无web端缓存，可支持上万/秒次查询，高峰期到过3000次/秒，平均响应时间在30ms
2. 原有查询使用数据库系统，查询压力增加时，回导致数据库复制堵塞，用户不能及时看到订单状态变化
下订单
已放票且有票
是否要排队
查询订单
准备支付
排队中
不需要排队
支付
买到票

分支機構

原有余票计和算查询 系統結構

即時資料流程

即時資料複製

即時資料複製

中央資料庫小型機

資料庫小型機 N > 5

資料庫小型機 M > 50

Web層 伺服器

應用層 伺服器

Web & App 小型機 N > 100

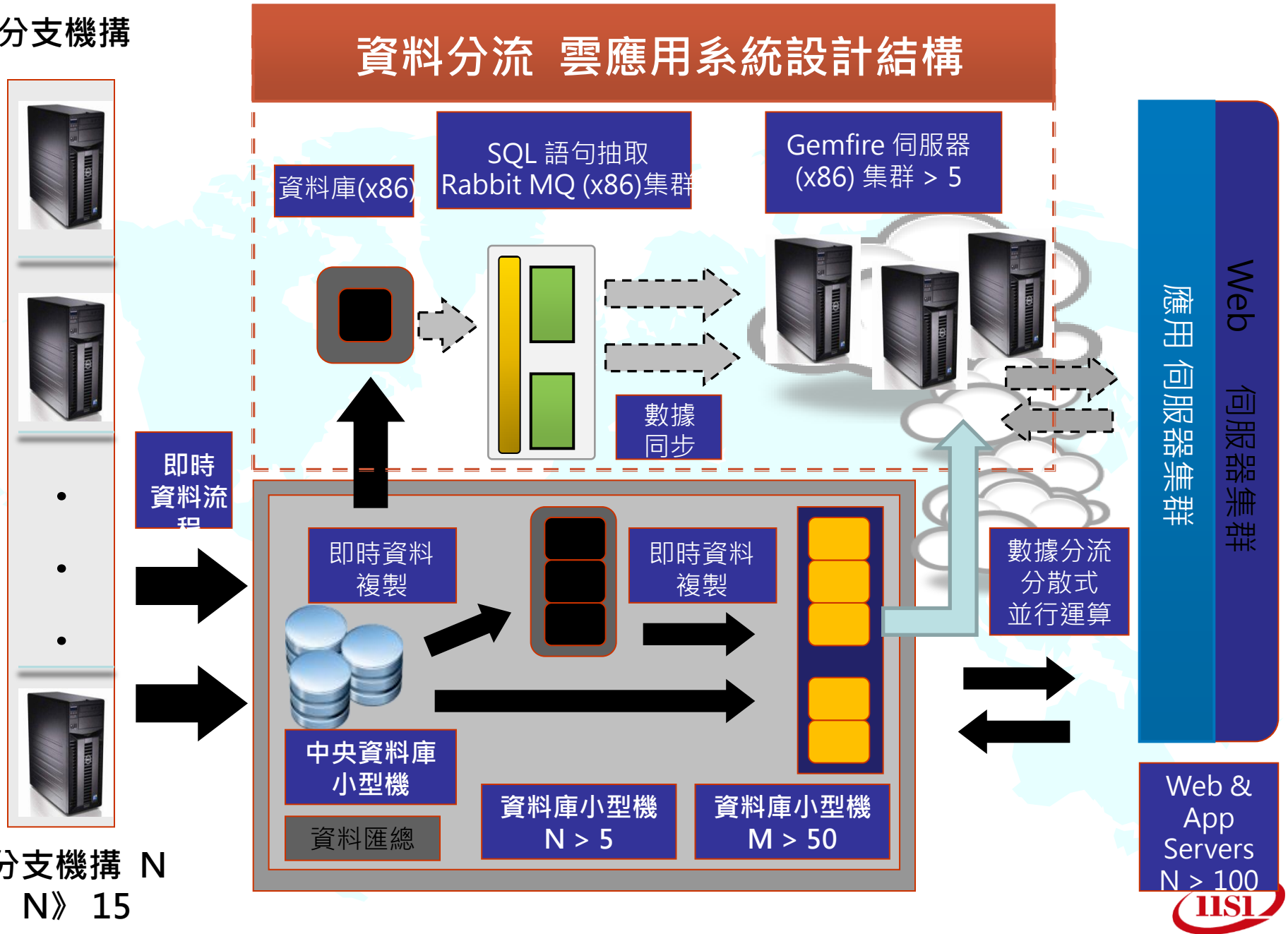
各地分支機構“資料即時複製”傳輸並匯總到總部中央資料庫

1.資料 “即時複製”到所有 “資料分庫”
2. 所有資料分庫資料統一一致，承擔部分預處理運算

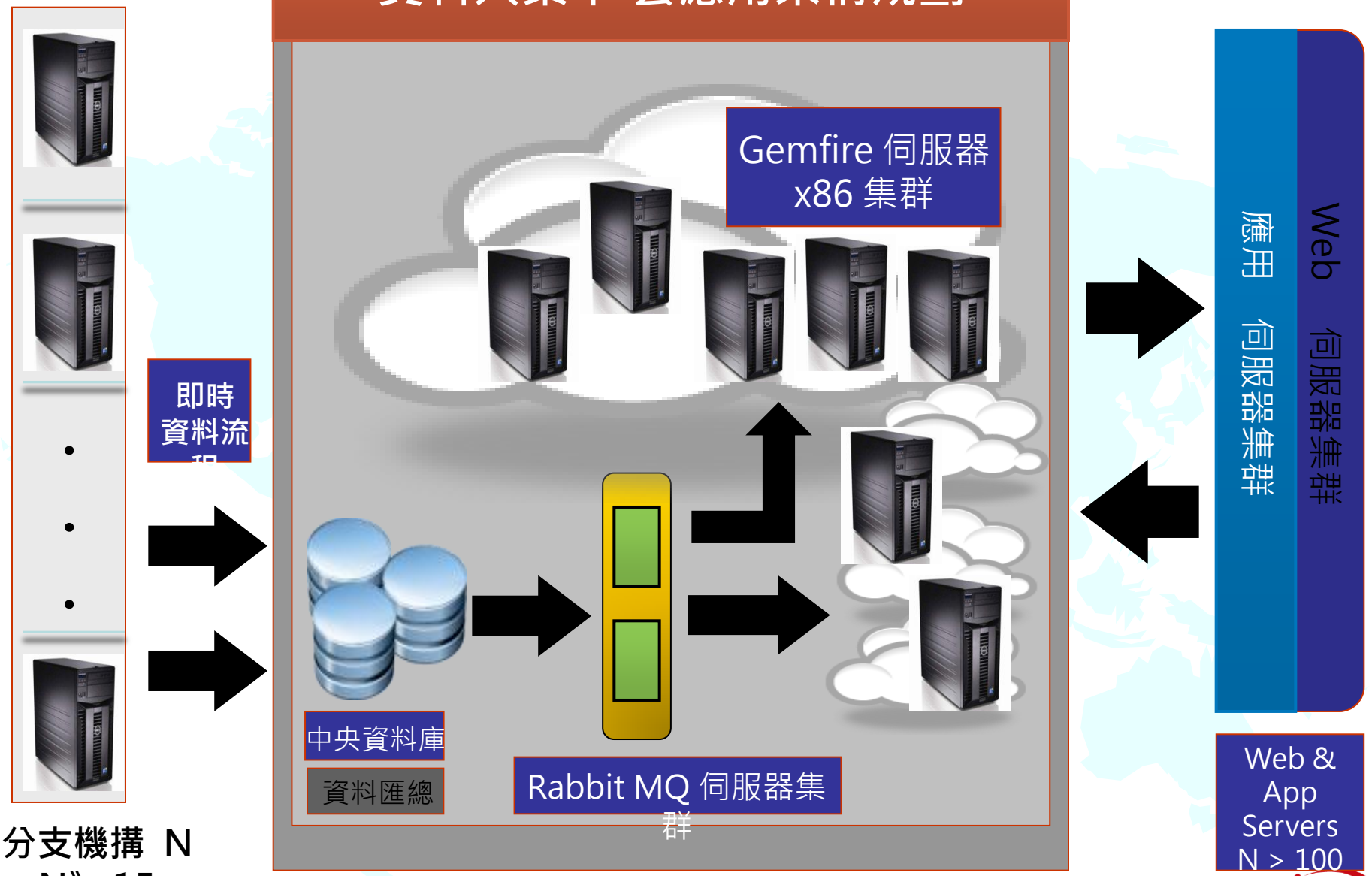
1.深度處理資料統一複製到所有 “資料分庫” 2. 資料分庫獨立運行，運算負荷高，DB IO 瓶頸

分支機構 N
N》 15

資料分流 雲應用系統設計結構



資料大集中 雲應用架構規劃

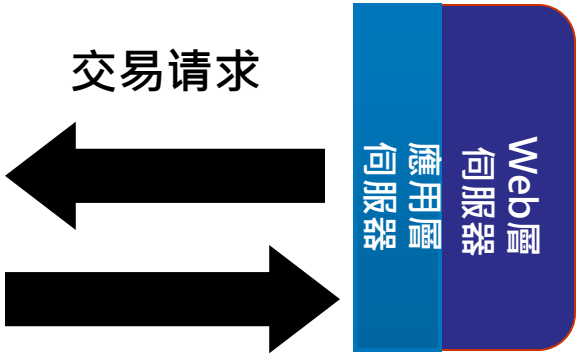


原有订单处理系統結構

交易 UNIX Server cluster：
最大处理能力： 200笔/秒

交易 web Server cluster

交易请求



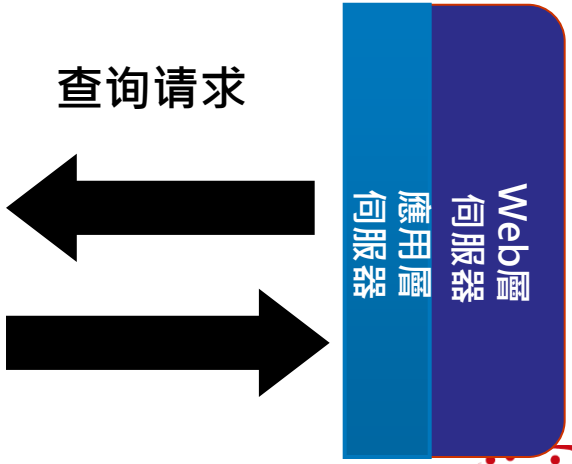
数据复制

- 复制数据到“查询Unix Server”数据库
- 海量并发查询, 影响数据复制效率
或造成复制过程中的数据堵塞

查询 UNIX Server cluster：
最大处理能力： 200-300笔/秒

查询 web Server cluster

查询请求



改造后订单处理系統結構

交易 UNIX Server cluster：
最大处理能力： 500笔/秒

交易 web Server cluster

交易请求

Web層伺服器
應用層伺服器

数据复制到内存

查询 web Server cluster

查询请求

Web層伺服器
應用層伺服器

Gemfire Server x86 cluster ：
最大处理能力：》 10000笔/秒
Average response time: 20-30 ms

x86 cluster

網上訂票-余票查询系統實際運行資料

IISI&VMware採用“資料分流”雲應用虛擬化技術方案對比

採用資料分流和雲應用虛擬化技術

- 單次訂票余票查詢最長耗時150-200毫秒
- 單次查詢最短耗時1-2毫秒
- 同步即時變化的資料耗時秒級
- 支持每秒上萬次的併發查詢，按需彈性動態擴展
- 運行在Linux X86伺服器VMware集群

性能平均提升百倍

原有技術設計框架

- 單次查詢耗時15秒左右
- 無法支持高流量併發查詢，只能通過分庫來實現
- 在極端高流量併發情況，系統無法支撐
- 運行在 UNIX小型機

客戶收益

- **可靠性**

  系統已經部署在生產環境中，一切正常穩定運行，準確度與原系統一致。

- **性能**

  原來訂票查詢需要15-30秒，系統改造上線後最长反應時間需要150-200毫秒，性能提升百倍。

- **性價比**

  使用基於Linux x86平臺的VMware解決方案，成本比原有Unix小型機節約幾十倍。

- **擴展性**

  使用VMware雲應用虛擬化解決方案，業務資料節點和x86伺服器VMware集群可彈性動態水準擴展，輕鬆應對高流量併發查詢。而原有基於xxx資料庫系統，遇到資料庫 I/O瓶頸，無法提高性能。