University of Bologna

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

Engineering Bologna Campus

Class of

Computer Networks M

EBAY

Antonio Corradi

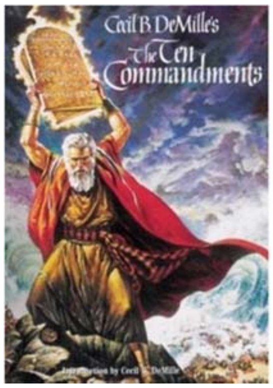
Academic year 2015/2016

MORE ALONG THAT LINE

Randy Shoup described **eBay Five Commandments for their system organization**

Thou shalt…

1. Partition Everything
2. Use Asynchrony Everywhere
3. Automate Everything
4. Remember: Everything Fails
5. Embrace Inconsistency



Challenges at Internet Scale

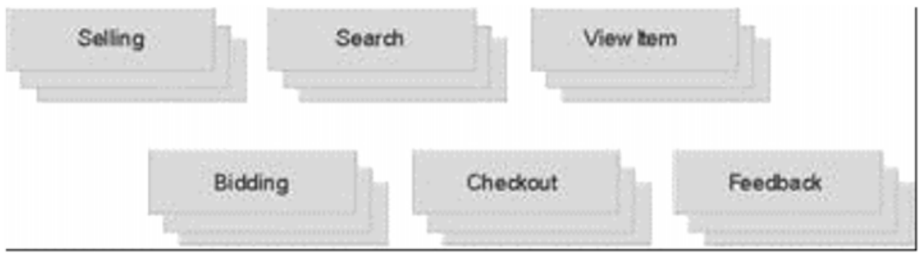
- eBay manages …
  - Over 276,000,000 registered users
  - Over 2 Billion photos
  - eBay users trade $2040 in goods every second -- $60 billion per year
  - eBay averages over 2 billion page views per day
  - eBay has roughly 120 million items for sale in over 50,000 categories
  - eBay site stores over 2 Petabytes of data
  - eBay Data Warehouse processes 25 Petabytes of data per day
- In a dynamic environment
  - 300+ features per quarter
  - We roll 100,000+ lines of code every two weeks
- In 39 countries, in 8 languages, 24x7x365

>48 Billion SQL executions/day!

1 - Partition Everything

Pattern: Functional Segmentation

- Segment processing into pools, services, and stages
- Segment data along usage boundaries



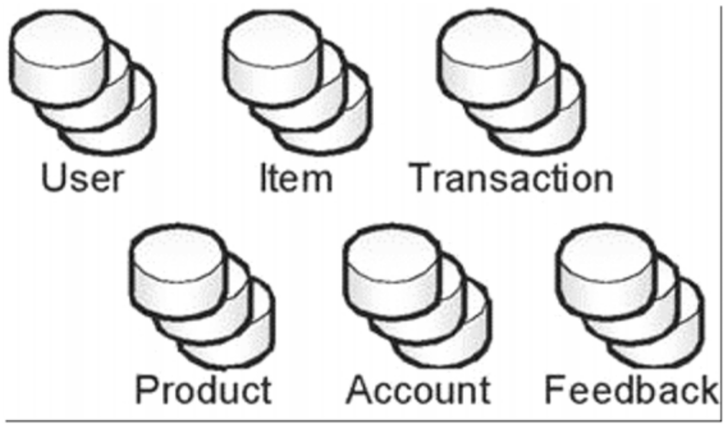
You should split anything you can in separated localities

No **large components** (to be kept consistent)

1 - Partition Everything

Pattern: Horizontal Split

- Load-balance processing
  all servers are created **equal** within a pool
- Split (or "shard") data along primary access path
  partition by range, modulo of a key, lookup, etc.



1 - Partition Everything

The principle suggests to simplify the management

Corollary: No Session State

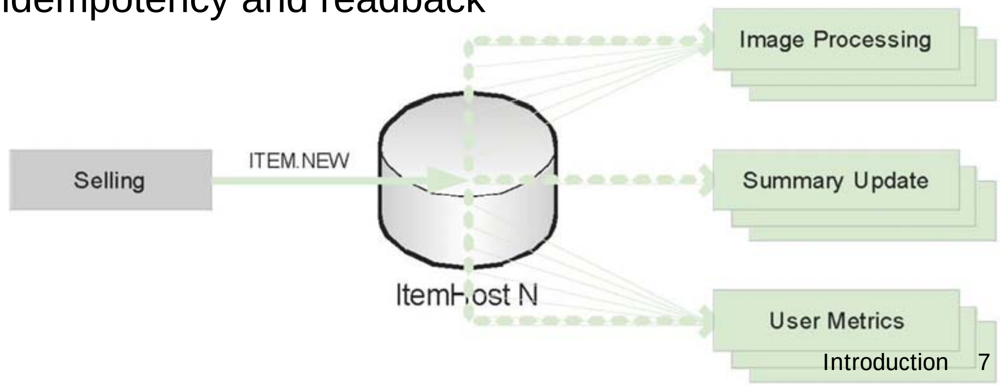
- User session flow moves through multiple application pools
- Absolutely no session state in application tier

Keep it simple (and short in time)

2 - Asynchrony Everywhere

Pattern: **Event Queue**

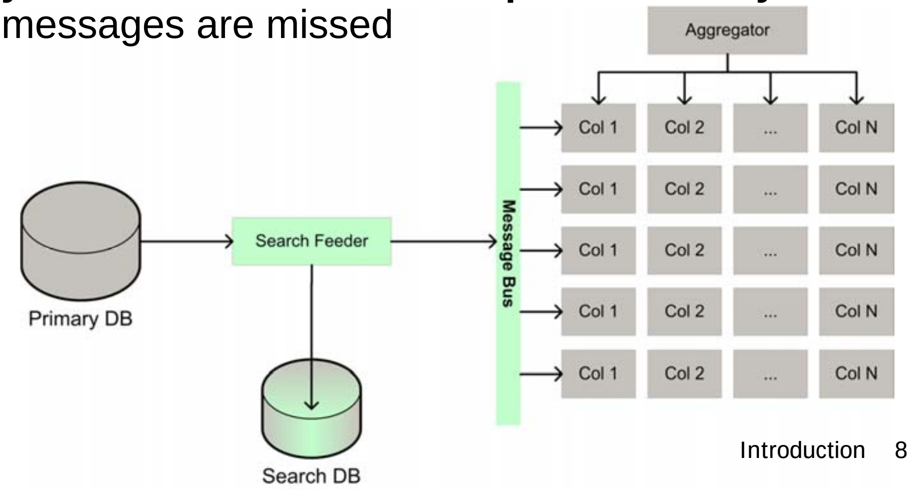
- Primary use-case **produces event transactionally** such as
  Create event (ITEM.NEW, ITEM.SOLD)
  with primary insert/update
- Consumers subscribe to event
  At least once delivery, No guaranteed order
  with idempotency and readback



2 - Asynchrony Everywhere

Pattern: **Message Multicast**

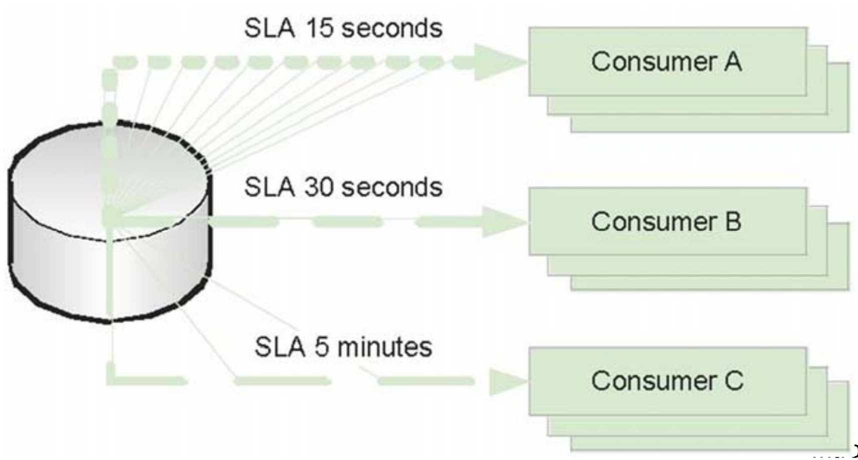
- **Search Feeder** publishes item updates, by reading item updates from primary database, and it publishes sequenced updates via Scalable Reliable Multicast-inspired protocol
- **Nodes listen** to assigned subset of messages, by the update of **in-memory index in real time** and **request recovery** (NAK) when messages are missed



3 - Automate Everything

Pattern: ***Adaptive Configuration***

- define **SLA for a given logical** consumer
  such as 99% of events processed in 15 seconds
- **dynamically adjust configuration** to meet defined SLA

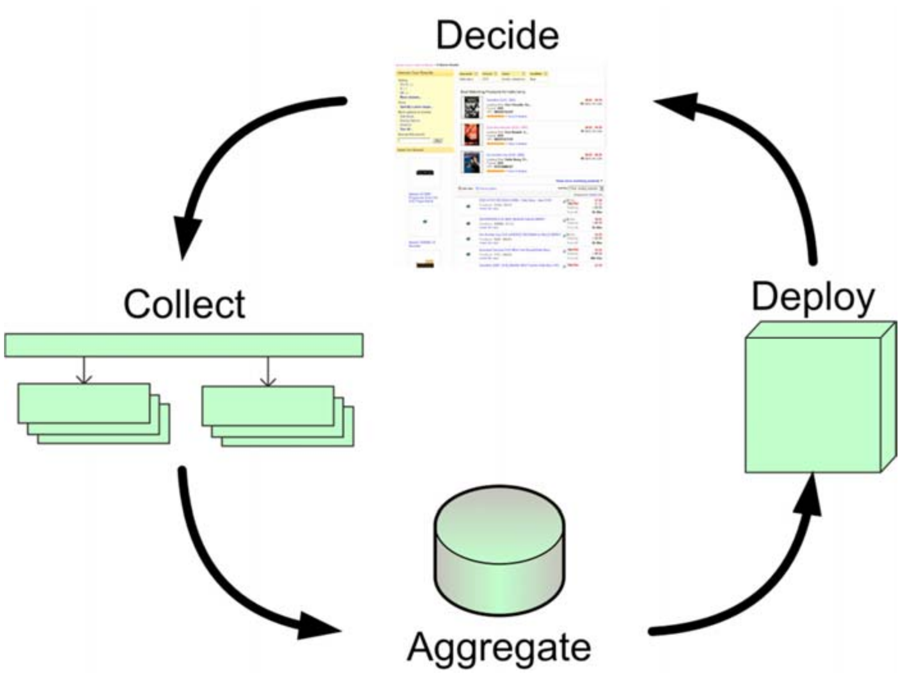


3 - Automate Everything

Pattern: **Machine Learning**

- **Dynamically adapt search** experience
  Determine best inventory and assemble optimal page for that user and context
- **Feedback loop** enables system to learn and improve over time
  Collect user behavior
  Aggregate and analyze offline
  Deploy updated metadata
  Decide and serve appropriate experience
- **Perturbation and dampening**

3 - Automate Everything



4 - Everything Fails

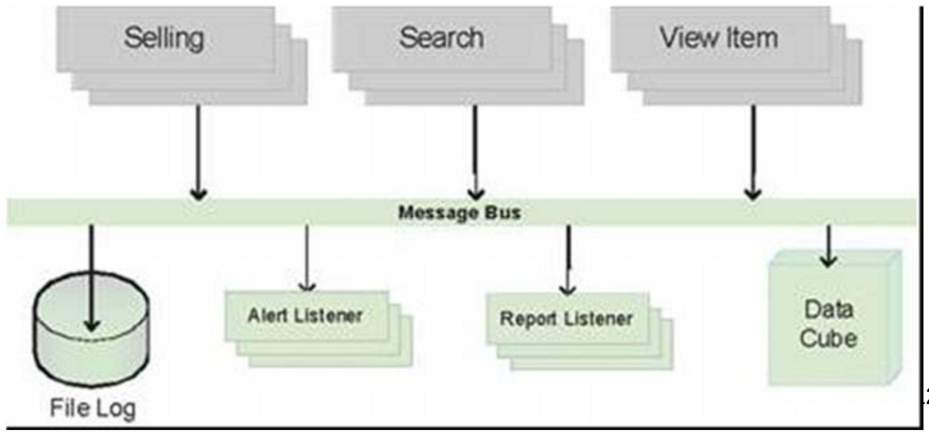
Pattern: **Failure Detection**

• Servers **log all requests**

Log all application activity, database and service calls on multicast message bus

Over 2TB of log messages per day

• Listeners **automate failure detection and notification**



4 - Everything Fails

Pattern: **Rollback**

- **Absolutely no changes** to the site that **cannot be undone** (!)

  The system does take any action in case irreversible actions are to be taken
- Every feature has **on / off state driven** by central configuration

  Feature can be immediately turned off for operational or business reasons

  Features can be deployed "**wired-off**" to unroll dependencies

4 - Everything Fails

Pattern: **Graceful Degradation**

- Application "marks down" an **unavailable or distressed resource**

  Those resources are dealt with specifically
- **Non-critical functionality is removed or ignored**

  All unneeded functions are neither considered nor generally supported
- **Critical functionality is retried or deferred**

  All critical points are dealt with specifically and in case of success no problem, in case of a failure, retried until completed

5 - Embrace Inconsistency

• Choose **Appropriate Consistency Guarantees**

According with Brewer's CAP Theorem prefer **eventual consistency** to **immediate consistency**

To guarantee availability and partition-tolerance, we trade off immediate consistency

Avoid Distributed Transactions

- eBay does absolutely **no distributed transactions – no two-phase commit**
- minimize inconsistency through **state machines and careful ordering of operations**
- **eventual consistency** through **asynchronous event or reconciliation batch**