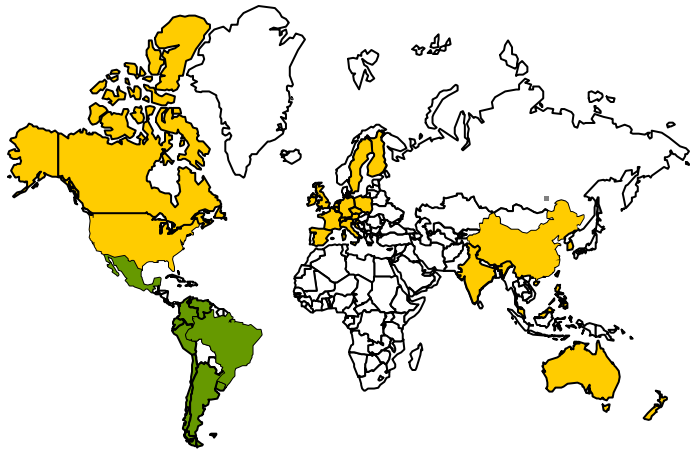
eBay's Challenges and Lessons
from Growing an eCommerce Platform to Planet Scale

Randy Shoup
eBay Distinguished Architect

HPTS 2009
October 27, 2009

Challenges at Internet Scale

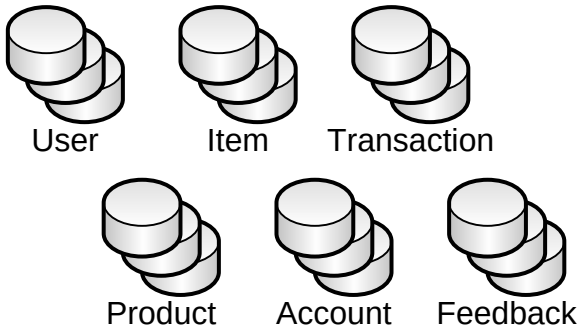
- eBay manages …
  - Over 89 million active users worldwide
  - 190 million items for sale in 50,000 categories
  - Over 8 billion URL requests per day
- … in a dynamic environment
  - Hundreds of new features per quarter
  - Roughly 10% of items are listed or ended every day
- … worldwide
  - In 39 countries and 10 languages
  - 24x7x365



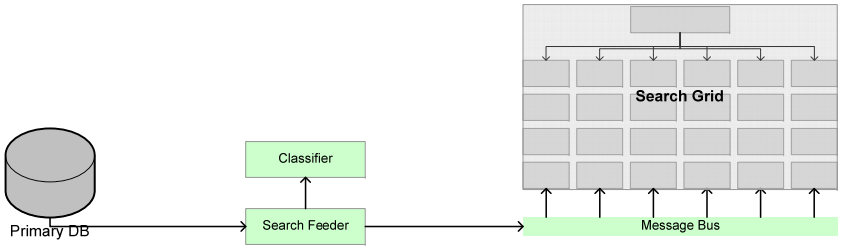
- **>70 billion read / write operations / day**

Architectural Lessons (round 1)

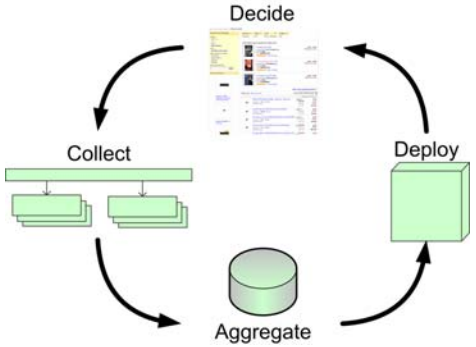
- 1. Partition Everything
  - Functional partitioning for processing (pools) and data (hosts)
  - Horizontal partitioning (“shards”) for data



- 2. Asynchrony Everywhere
  - Event-driven queues and pipelines (at-least-once delivery, order-agnostic)
  - Multicast messaging (SRM-inspired techniques for reliability)



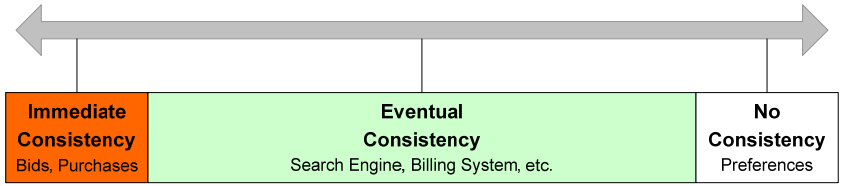
- 3. Automate Everything
  - Adaptive configuration of components
  - Feedback loops and machine learning



Architectural Lessons (round 1)

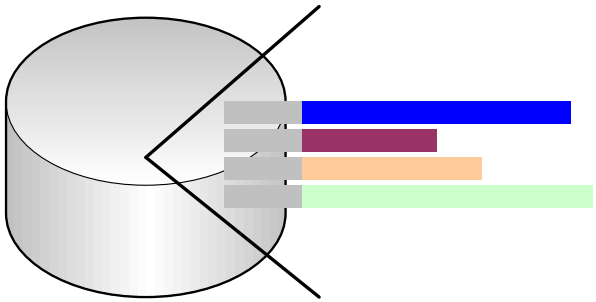
- 4. Remember Everything Fails
  - Extensive telemetry for failure detection
  - Graceful degradation of functionality

- 5. Embrace Inconsistency
  - Consistency is a spectrum
  - Each usecase trades off CAP properties
  - No distributed transactions
  - Minimize inconsistency through state machines and careful ordering of operations
  - Eventual consistency through asynchronous recovery or reconciliation

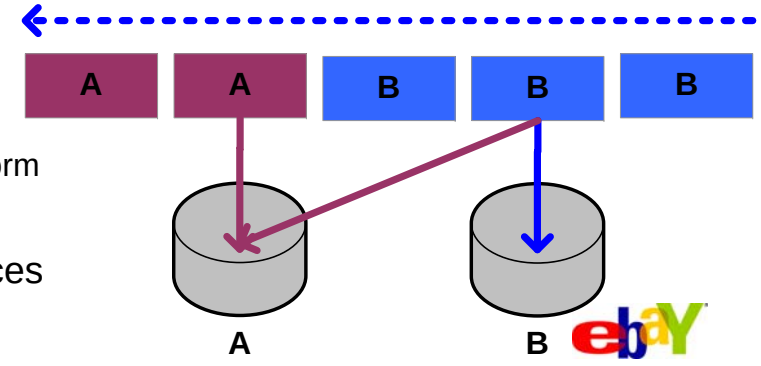


Lesson 6: Expect (R)evolution

- **Change is the Only Constant**
  - New entities and data elements
  - Constant infrastructure evolution
  - Regular data repartitioning and service migration
  - Periodic large-scale architectural revolution
- **Design for Extensibility**
  - Flexible schemas
    - Extensible interfaces (attributes, k-v pairs)
    - Heterogeneous object storage
  - Pluggable processing
    - Disparate systems communicate via events
    - Within system, processing pipeline controlled by configuration

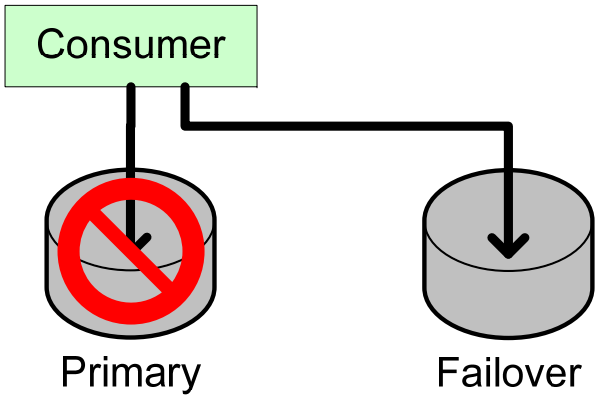


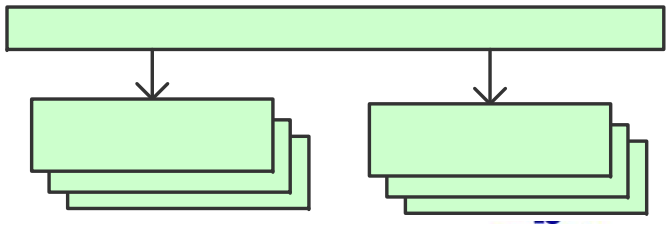
- **Incremental System Change**
  - Decompose every system change into incremental steps
  - Multiple versions and systems coexist
    - Every change is a rolling upgrade; transitional states are the norm
    - Version A -> A|B -> B|A -> Version B
  - Strict forward / backward compatibility for data and interfaces
  - Dual data processing and storage (“dual writes”)



Lesson 7: Dependencies Matter

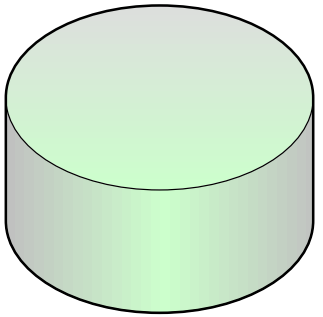
- **Minimize and Control Dependencies**
  - Service topology constrained by dependencies
    - Data center moves change latency characteristics (!)
  - Depend only on abstract interface and virtualized endpoint
  - Make QoS parameters (latency, throughput) explicit in SLA
- **Consumer Responsibility**
  - It is fundamentally the consumer's responsibility to manage unavailability and SLA violations
  - (Un)availability is an inherently *Leaky Abstraction*
    - 1st Fallacy of Distributed Computing: "The network is reliable"
  - Recovery is typically use-case-specific
    - Driven by criticality of the operation and the strength of dependency
  - Can abstract with standard patterns
    - Sync or async failover, degraded function, sync or async error
- **Monitor Dependencies Ruthlessly**
  - Registries provide WISB but only monitoring provides WIRI
  - Invaluable for problem diagnosis and capacity provisioning



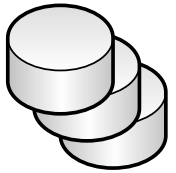


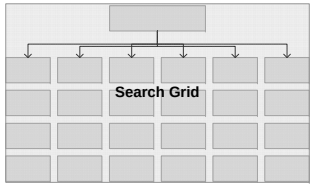
Lesson 8: Be Authoritative

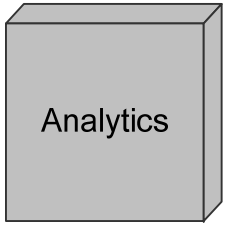
- **Authoritative Source ("System of Record")**
  - At any given time, every piece of (mission-critical) data has a System of Record
  - Authority can be explicitly transferred (failure, migration)
  - Typically transactional database



- **Non-authoritative Sources**
  - Every other copy is derived / cached / replicated from System of Record
    - Remote disaster replicas
    - Search engine
    - Analytics
    - Secondary keys
  - Relaxed consistency guarantees with respect to System of Record
  - Optimized for alternate access paths or QoS properties
  - Perfectly acceptable for most use-cases

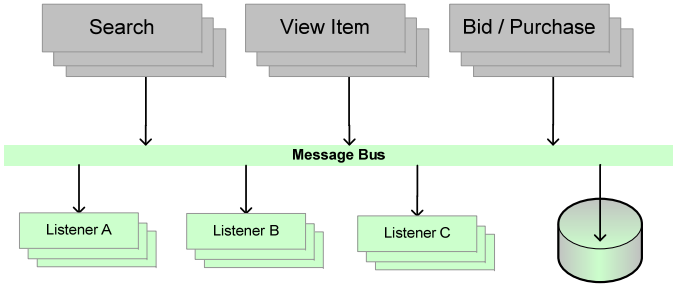




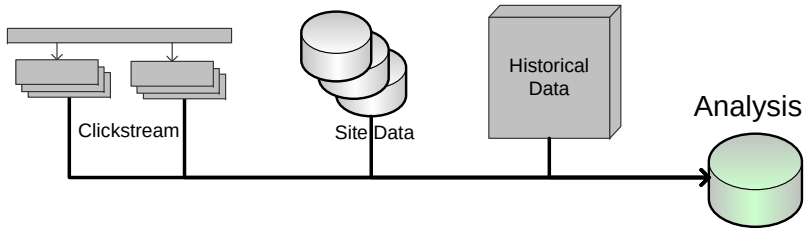


Lesson 9: Never Enough Data

- **Collect Everything**
  - eBay processes 50TB of new, incremental data per day
  - eBay analyzes 50PB of data per day
  - Every historical item and purchase is online or nearline
  - Requires large-scale distributed storage
- **Example: System Monitoring**
  - Failures at scale are difficult to diagnose and near-impossible to replicate
    - Requires granular instrumentation of every operation
  - Stream processing for pattern detection and failure prediction
  - Historical data to identify optimization opportunities and inform capacity provisioning

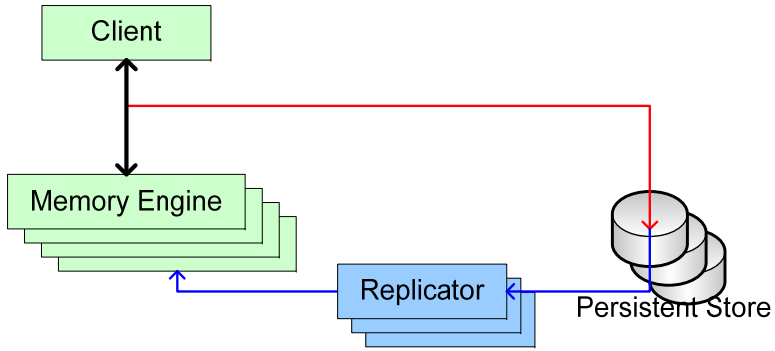


- **Example: Recommendations and Ranking**
  - Collect user behavior in the clickstream
    - Collect -> filter -> enrich -> aggregate -> store
  - Drive purchase recommendations
  - Drive models that predict value of page view, module impression, pixel allocation
  - Predictions in the long tail require massive data

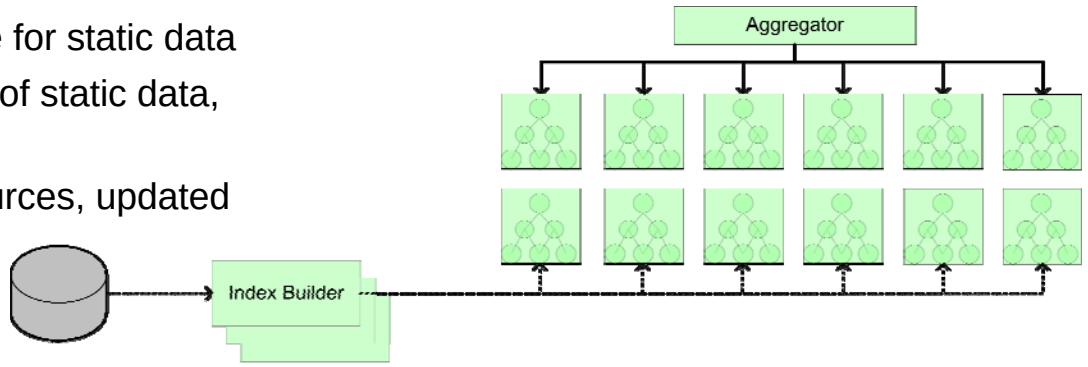


Lesson 10: Custom Infrastructure

- **Right Tool for the Right Job**
  - Need to maximize utilization of every resource
    - Data (memory), processing (CPU), clock time (latency), power (!)
  - One size rarely fits all, particularly at scale
  - Compose from orthogonal, commodity components
- **Example: Session and Personalization Cache**
  - In-memory volatile KVSS on partitioned MySql Memory Engine
  - Async replication to partitioned backing store (Oracle)
  - State redistributed on node failure
  - Versioning, optimistic concurrency, and resolver pattern for conflicts



- **Example: Metric Server**
  - In-memory hierarchical lookup structure for static data
  - Shared infrastructure for multiple types of static data, partitioned horizontally
  - Index built offline from multiple data sources, updated periodically



Questions?

- Randy Shoup, eBay Distinguished Architect (rshoup@ebay.com)