eBay -
Very Large Distributed Systems (a.k.a. Grids) @ Work

Paul Strong
Distinguished Engineer, eBay Research Labs
Acting Chair, Open Grid Forum (OGF)
© 2008 eBay Inc. All rights reserved.

- No part of these materials may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording, or otherwise) without the prior permission of eBay Inc.
- eBay and the eBay logo are registered trademarks of eBay Inc.
- PayPal and the PayPal logo are registered trademarks of PayPal, Inc.
- Other trademarks and brands are the property of their respective owners.
- Please do not take our picture or record the class/session without asking permission.
eBay – The 30 Second Introduction!

eBay users trade about $2,039 worth of goods on the site every second

On an average day on eBay...

A vehicle sells every minute
A motors part or accessory sells every second
Diamond jewelry sells every 2 minutes

1.3m people make all or part of their living selling on eBay

©2008, eBay Inc.

*ACNielsen International Research, June 2006
PayPal – The Even Shorter Summary!

- 141 million accounts (57 m active*)
- 17 currencies
- 2008 - $47 billion TPV
- Q4 2008 –
  - $14 billion
  - $1806 TPV/sec
  - 12% US e-Commerce, 8 global e-Commerce
  - #2 e-Commerce payment mechanism in US (#1 VISA)
  - #1 e-Commerce payment mechanism in UK, Australia

* At least 1 transaction in last 12 months
Why eBay Is A Useful Example

New Challenges
Extreme Engineering

The Bleeding Edge

Technology trickle down/transfer

Everyday use

©2008, eBay Inc.
eBay’s Drivers

• Extreme Scale
  276m Registers Users, 113m+ Items, 6m+ New Items Per Day

• Extreme Growth
  Near exponential growth in listings for most of history – 12 years

• Extreme Agility
  Roll code to the site every 2 weeks

• Constant, predictable presence
  Must be 24x7x365

• Efficiency

Failure To Keep Up Is Not An Option!

©2008, eBay Inc.
Challenges

- Scaling The Database
- Scaling Services
- Managing At Scale
- Better Management Through Semantics
Grid – eBay’s Perspective?

• Inevitable consequence of trends
  • Network of servers ➔ Fabric of resources
  • Server centric apps ➔ Network distributed services

• Network distributed services + platform
  • Scales (performance, throughput) with network
  • Inherent resilience
  • Flexibility (if loosely coupled)

• Middle-ware
  • Meta-OS maps workload onto resources based on policy
  • Incomplete today

• General purpose platform
  • Compute intensive
  • Data intensive
  • Transaction intensive
  • Hybrid

©2008, eBay Inc.
Grids @ eBay

- Build and release – “Traditional” Grid
  - 300+ servers
- Search – Scatter/gather transactional
  - 3000-4000 servers
- Auction Platform – Transactional
  - 8000+ Blades
- Virtualized Database – Data Grid
  - 630+ Database Instances
  - Extensive caching and distribution
eBay Example #1
Making The Database Scale

- Second Database for failover
- CGI pools, Listings, Pages, and Search continued to scale horizontally

However ...

By November 1999, the database servers approached their limits of physical growth.

1999
eBay Example #1
Making The Database Scale

- Database "split" technology.
- Logically partition database into separate instances.
- Horizontal scalability through 2000, but not beyond.

2000

4 RDBMS
4 UNIX
bull.ebay.com bear.ebay.com chard.ebay.com cab/bongo.ebay.com
eBay Example #1
Virtualizing the Database

- Separate Application notion of a database from physical implementation
- Databases may be combined and separated with no code changes
- Reduce cost of creating multiple environments (Dev, QA, …)
- Application can continue to function without non-critical data (markdown)

©2008, eBay Inc.
eBay Example #1
Virtualizing & Scaling the Database

November, 1999
eBay Example #1
Virtualizing & Scaling the Database

December, 2002
SAN
eBay Example #1
Virtualizing & Scaling the Database

• Scales Out
  276 million registered users
  113 million Items
  6+ million new items per day
  34 billion SQL transactions per day
  600+ production database instances (inc replicas)
  100+ clusters

• Cheaper
  Smaller, potentially commodity, servers

• Highly Resilient
  2-4 copies of everything
  Minimized impact of outage to [relatively] small sub-set of data

• Flexible/Agile
  Easy to change – database, schemas, partitioning etc.
  Minimal impact on architecture or code
Stateless is good but how do you scale things that share dependencies?
eBay Example #2
Scaling Services

• Partition code into functional areas
  – Application is specific to a single area (Buying, Selling etc.)
  – Domain contains common business logic across applications

• Restrict inter-dependencies
  – Applications depend on Domains, not on other applications
  – No dependencies among shared domains
eBay Example #2
Scaling The Application

• Segment functions into separate application pools
  – Minimizes/isolates DB dependencies
  – Allows for parallel development, deployment and monitoring

ViewItem Pool
http://cgiX.ebay.com...

SYI Pool
http://cgiY.ebay.com...

©2008, eBay Inc.
eBay Example #2
Scaling The Application

• Everything behaves as loosely coupled services
• Minimize inter-dependencies
• Infrastructure is like a giant FPGA
  – Potential to re-program by re-routing traffic
• Scales
  – Scale out means scaled throughput and resilience
  – 16000+ concurrent instances
  – 8000+ servers (mainly blades)
• Efficiency
  – Run traffic from different time zones on the same server but different instances
Scaling Search – Voyager

- **Real-time feeder infrastructure**
  Reliable multi-cast from primary database to search nodes

- **Real-time indexing**
  Search nodes update index in real time from messages

- **In memory search index**

- **Horizontal segmentation (scatter, gather)**
  Search index divided into N slices (“columns”)
  Each slice replicated to M instances (“rows”)
  Aggregator parallelizes query over all N slices, load balances over M instances

- **Caching**
  Cache results for highly expensive and frequently used queries
Architectural Lessons Learnt

• Scale Out, And Enable Scaling Up Too
  Horizontal scaling at every tier plus multi-threading too
  Enable deployment time choice
  Functional decomposition

• Prefer Asynchronous Integration
  Minimize availability coupling
  Improve scaling options

• Virtualize Components
  Reduce physical dependencies
  Improve deployment flexibility

• Design For Failure
  Automated failure detection and notification
  “Limp mode” operation of business features
Management complexity scales with this
Understanding Relationships

Service A is composed of
Persistence Sub-Service B
Business Logic Sub-Service C
Presentation Sub-Service D

©2008, eBay Inc.
Business Logic Sub-Service C is composed of

- A Load Balancing Service
- Several Application Instances
Understanding Relationships

The Application Instances are *hosted on* Operating System Instances

The Load Balancing Service is *hosted on* A Load Balancer Operating System
Understanding Relationships

The Operating System Instances are *hosted* on
Servers or Virtual Servers, which are in turn hosted on servers

The Load Balancer OS is *hosted* on
A Physical Load Balancer
Categorizing The Components
Interaction/Traffic Relationships Starting To Look Complicated!
Relationships Are Everything!

• Everything is interconnected
• Changing one thing causes ripples
• How you connect things together determines business functionality and business value
• Agility is the ability to change these relationships dynamically (easier with loosely coupled services)
• Virtualization is about standardizing a relationships and interposing/isolating one end from the other
• Understanding these relationships allows you to
  Tie business processes to the infrastructure they run on
  Map value to cost
  Understand and manage traffic flow
  Understand and manage provisioning etc.
• It’s all about managing relationships, not things!

©2008, eBay Inc.
Managing Complexity Using Patterns Example - Storage

- **Minimize Set Of Components**
  - Products - HBAs, Arrays, Switches
  - Vendors

- **Small Set Of Storage Classes**
  - eBay has 5 or 6 patterns
  - Each reflects an SLO in terms of
    - Performance
    - Availability
    - Cost
Typical eBay Storage Pattern

- Stripe Volume Across LUNs from Different Loops
- Dual SANs
- 4 Paths Per Node
- Partition Stripes - LUNs
- Stripe Across Mirrors
- Mirror Whole Disks
- Disks On Dual Loops
Patterns Are Successful...

• Scale
  – 15 SANS / 245 switches / 7,800 ports
  – 78 arrays / 3.5 PB’s / 56,000 luns
  – 180 Clusters / 890 servers / 650 databases

• Agility
  – 11 TB/Week Provisioned
  – 85 New Data Volumes Per Week
  – 10 Database Moves/Week

Managed by 11 People
But…

- Patterns constrain agility in other dimensions
  - Adoption of new vendors
  - Adoption of new products
  - Adoption of new application or architectural patterns
- Especially if patterns and tools are heavily automated
The Future...

- Datacenter is becoming non-deterministic or chaotic
- Emergent behavior of services
The Future…
Better Management Through Semantics

• Capture relationships in Semantic Query Service (in memory, custom OWL/RDF based Ontology)
  – Extensible
  – Patterns in data and not in code!
• Feed from CMDB and from Run Time Telemetry
• Query-able by other management services
• Visualized as graphs (DAGs) through UI
Of Grids & Clouds

- Simplify building applications
- Eliminate management of unnecessary infrastructure
- Grids can supply services accessed via clouds
- Grids can run in/on clouds
Future Platforms & Business Paradigms

• Clouds & Grids ride the wave of application, and by thus by inference, business process disaggregation
• Eliminate/out-source non-core business process
  • Access via clouds as commodity services?
• Opportunity to mash up new process
  • Program via BPEL or something simpler?
• Opportunity for new platforms
Conclusions

• Large scale, distributed systems (aka Grids) are the platform of the present and future
• If you can’t or don’t want to manage one, you will use someone else’s in the cloud
• Disaggregation will lead to biz process re-factoring
• Opportunities
  • New services via mashed up biz processes
  • New platforms to support ubiquitous biz process (extending SaaS models, eBay, Google, Amazon)
  • Businesses with no infrastructure, just good ideas, smart people, minimal technical knowledge and access to the cloud!
Thank You

Paul Strong
Paul.Strong@eBay.com
Distinguished Engineer
eBay Research Labs,
eBay Inc.