eBay Architecture

Scalability with Agility

Tony Ng

Director, Systems Architecture

October 2011

About Me

- eBay – Systems Architecture and Engineering
- Yahoo! – Social, Developer Platforms, YQL
- Sun Microsystems – J2EE, GlassFish, JSRs
- Author of books on J2EE, SOA

eBay Stats

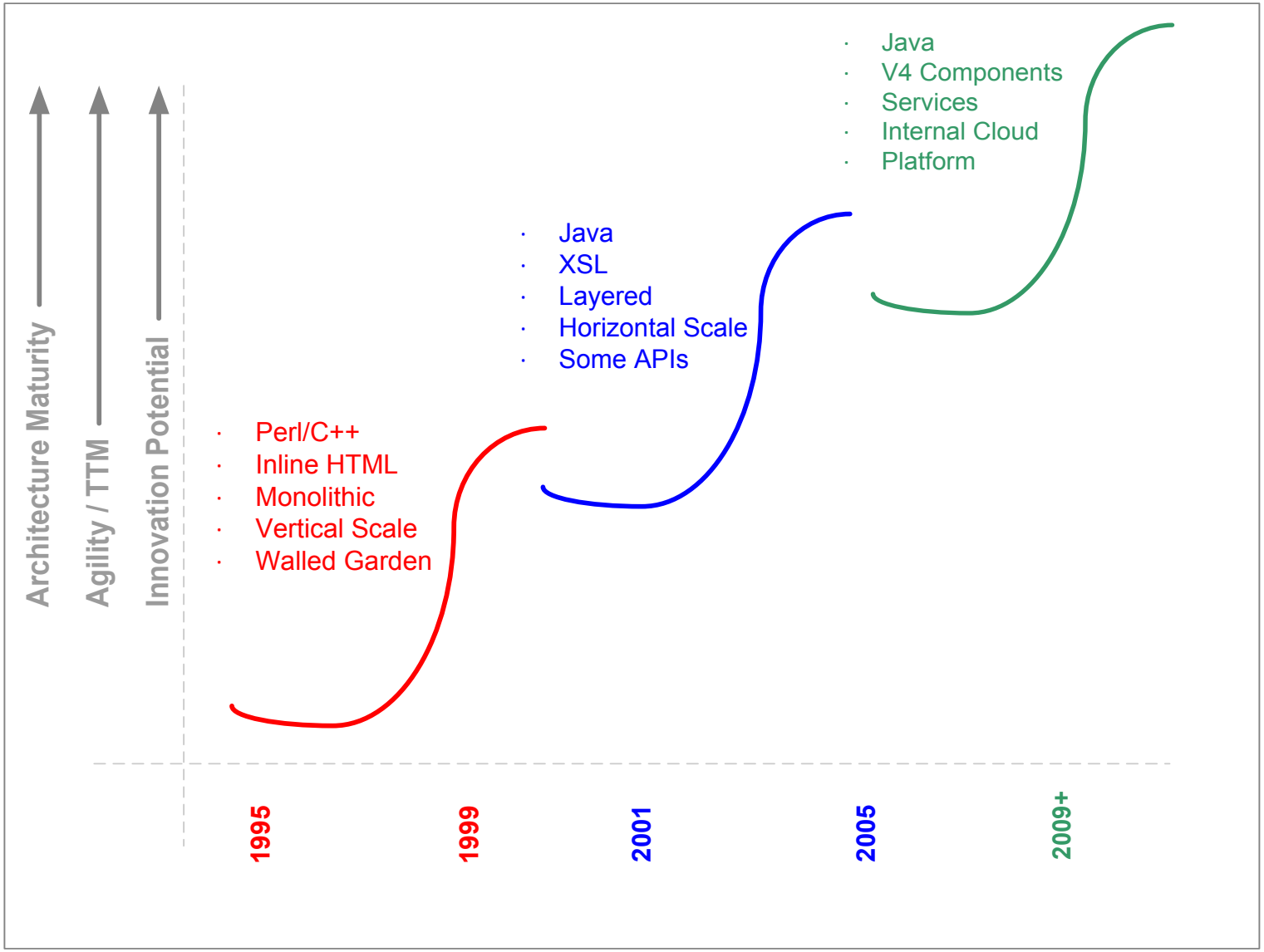
- 97 million active users
- 62B Gross Merchandise Volume in 2010
- 200 million items for sale in 50,000 categories
- A cell phone is sold every 5 seconds in US
- An iPad sold every 2.2 minutes in US
- A pair of shoes sold every 9 seconds in US
- A passenger vehicle sold every 2 minutes
- A motorcycle sold every 6 minutes

http://www.ebayinc.com/factsheets

eBay Scale

- 9 Petabytes of data storage
- 10,000 application servers
- 44 million lines of code
- 2 billion pictures
- A typical day
  - 75B database calls
  - 4B page views
  - 250B search queries
  - Billions of service calls
  - Hundreds of millions of internal asynchronous events

History of Technology



eBay Scalable Architecture

- Partition everything
  - Databases, application tier, search engine
- Stateless preference
  - No session state in app tier
- Asynchronous processing
  - Event streams, batch
- Manage failures
  - Central application logging
  - Mark downs

Next Challenges

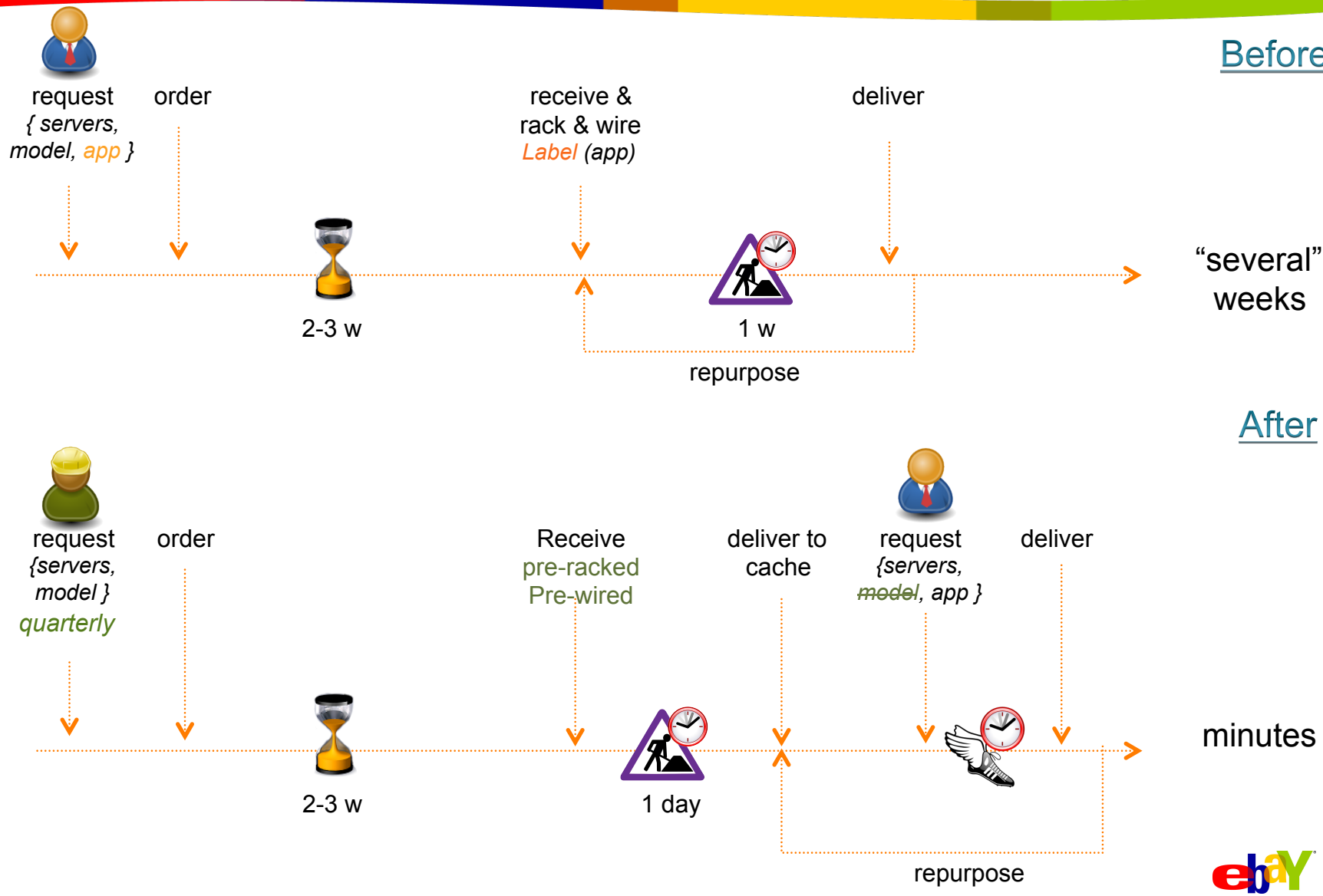
- To stay competitive, we need to deliver quality features and innovations at accelerating paces
- Complexity as our codebase grows
- Improve developer productivity
- Enable faster time-to-market while maintaining site stability

Scalability with Agility

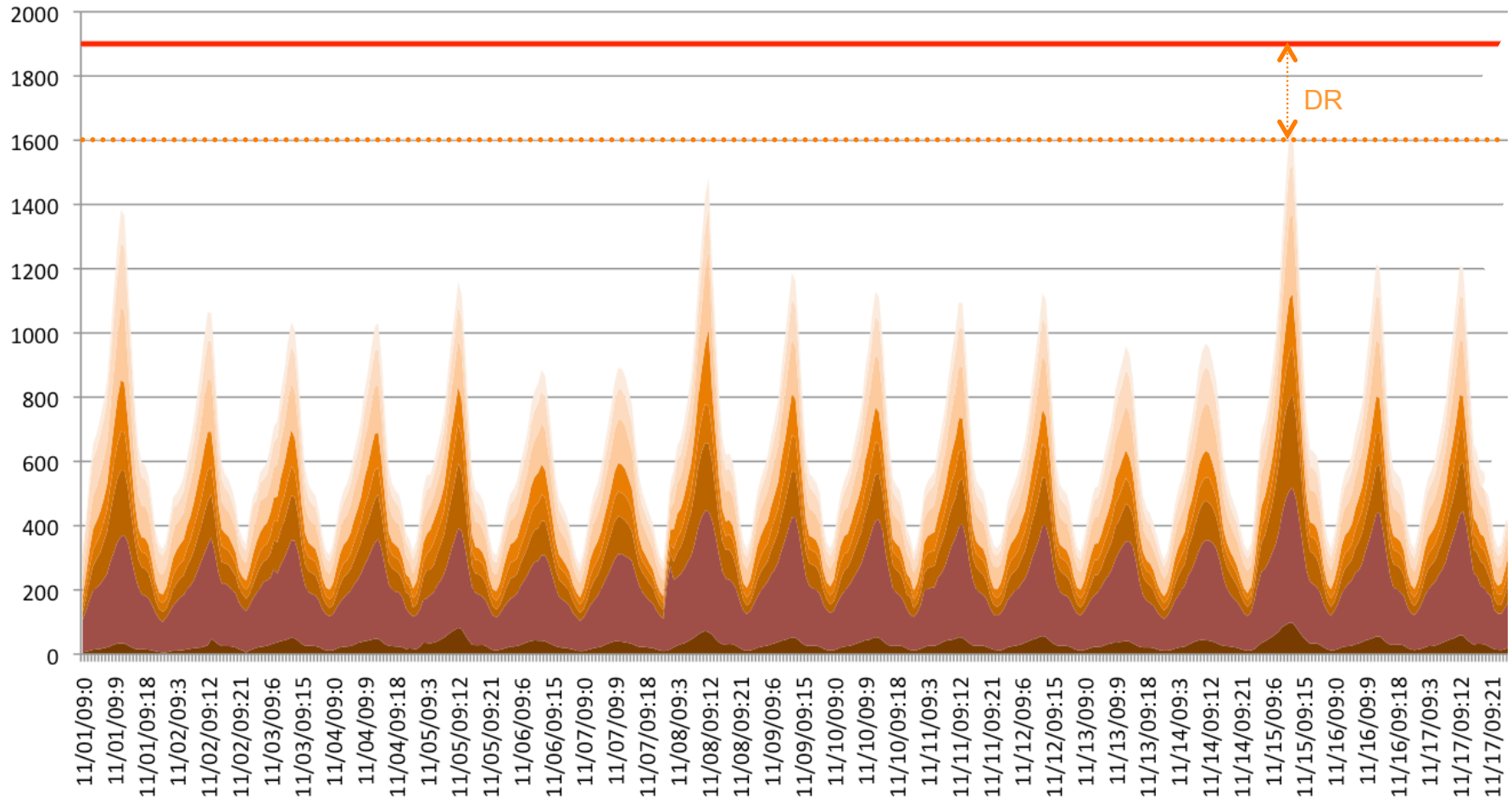
- Strategy 1: Automation with Cloud
- Strategy 2: Next Gen Service Orientation
- Strategy 3: Modularity

Automation with Cloud

Hardware Acquisition

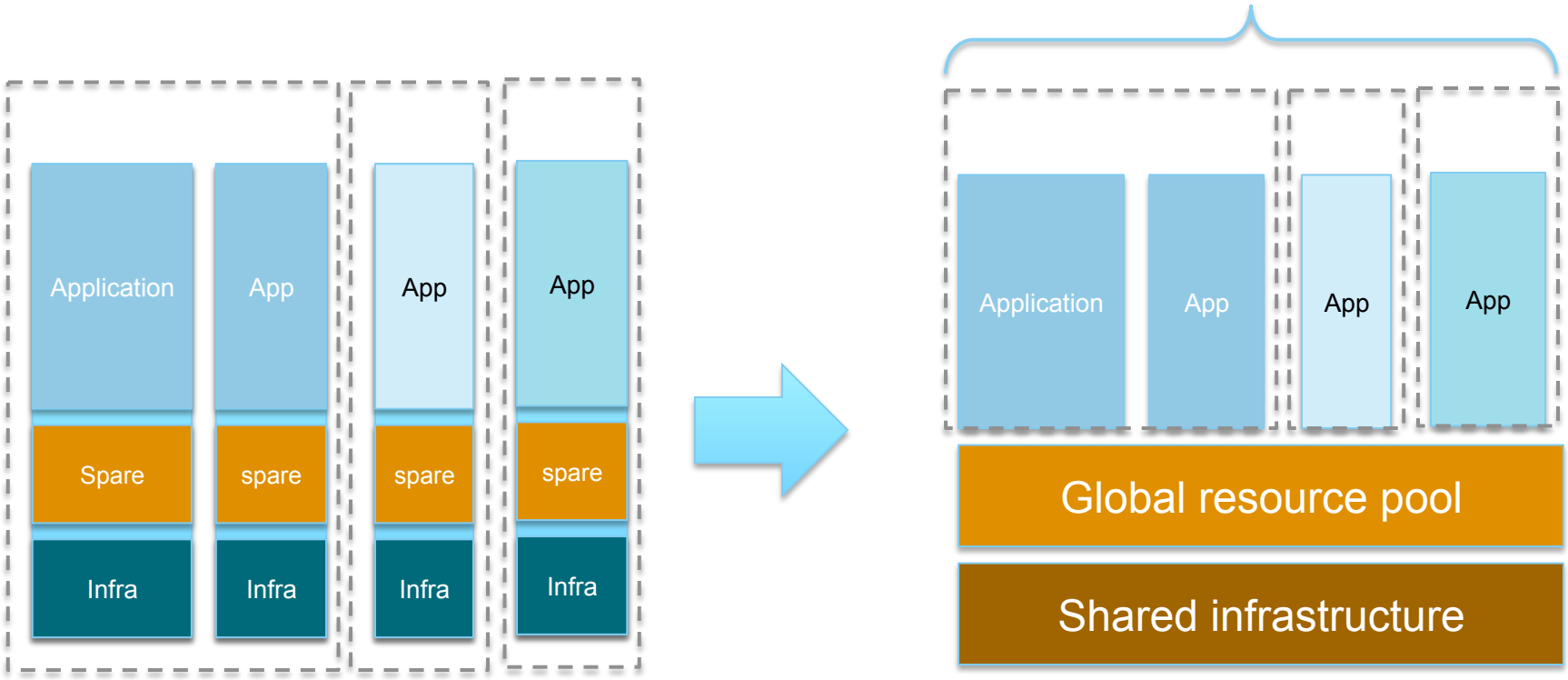


Improving Utilization



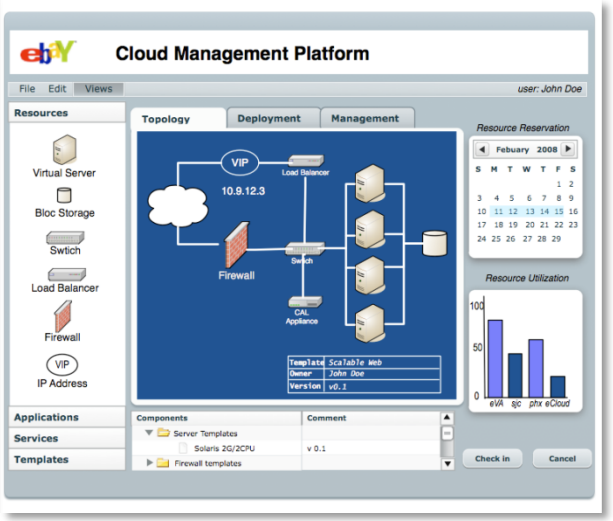
Number of servers required based on utilization for 8 pools

Infrastructure Virtualization



eBay Cloud

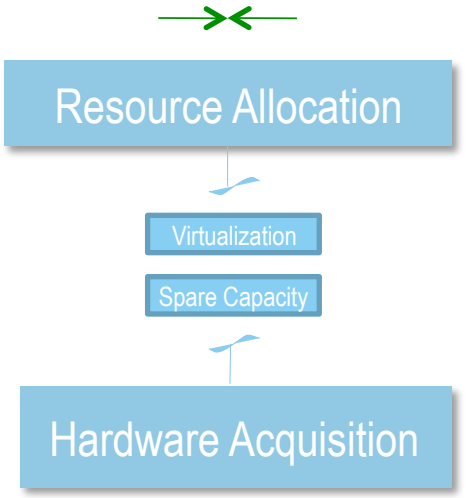
Self Service Portal



Automation

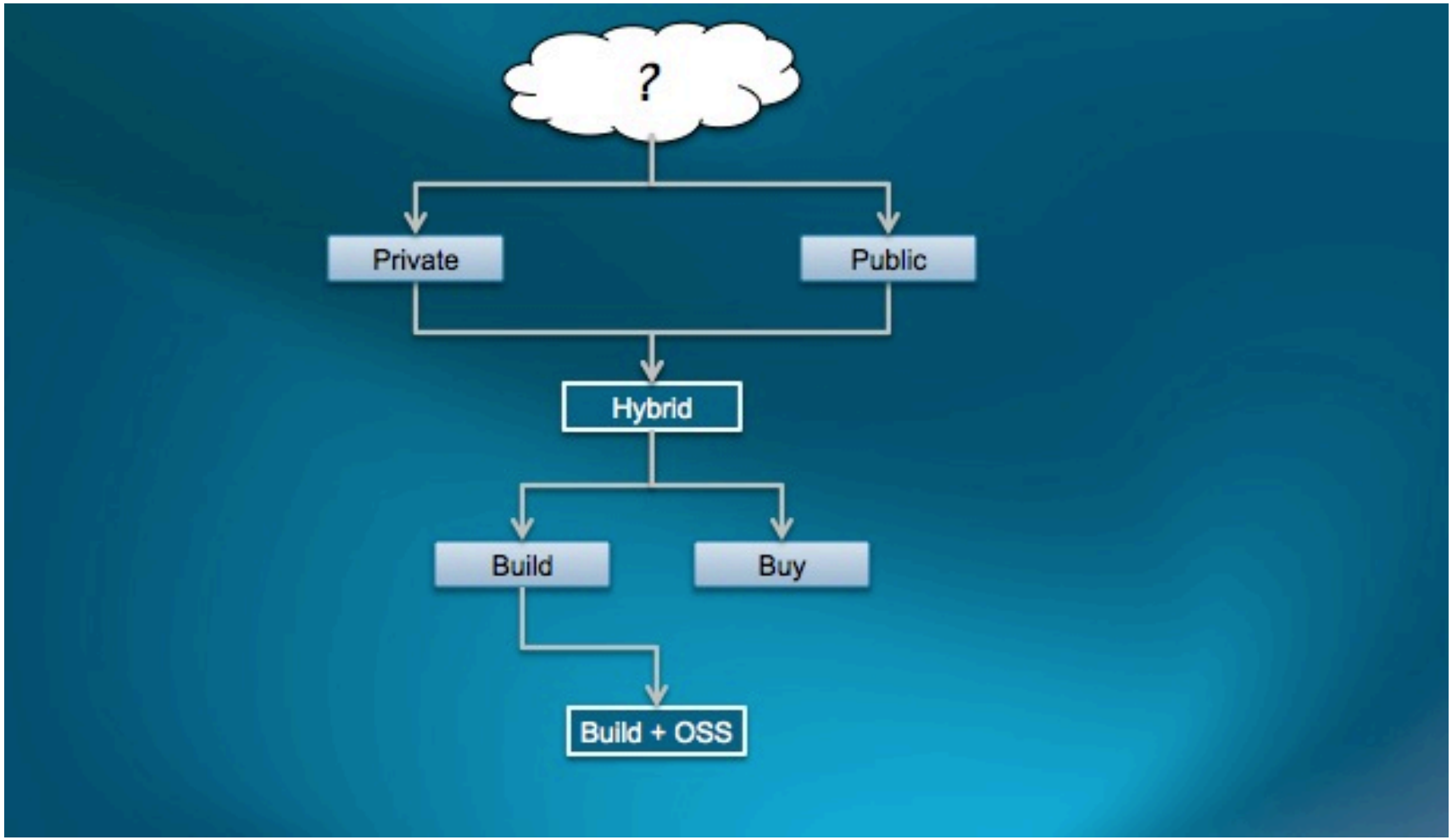
pool provisioning in minutes

Capacity Management

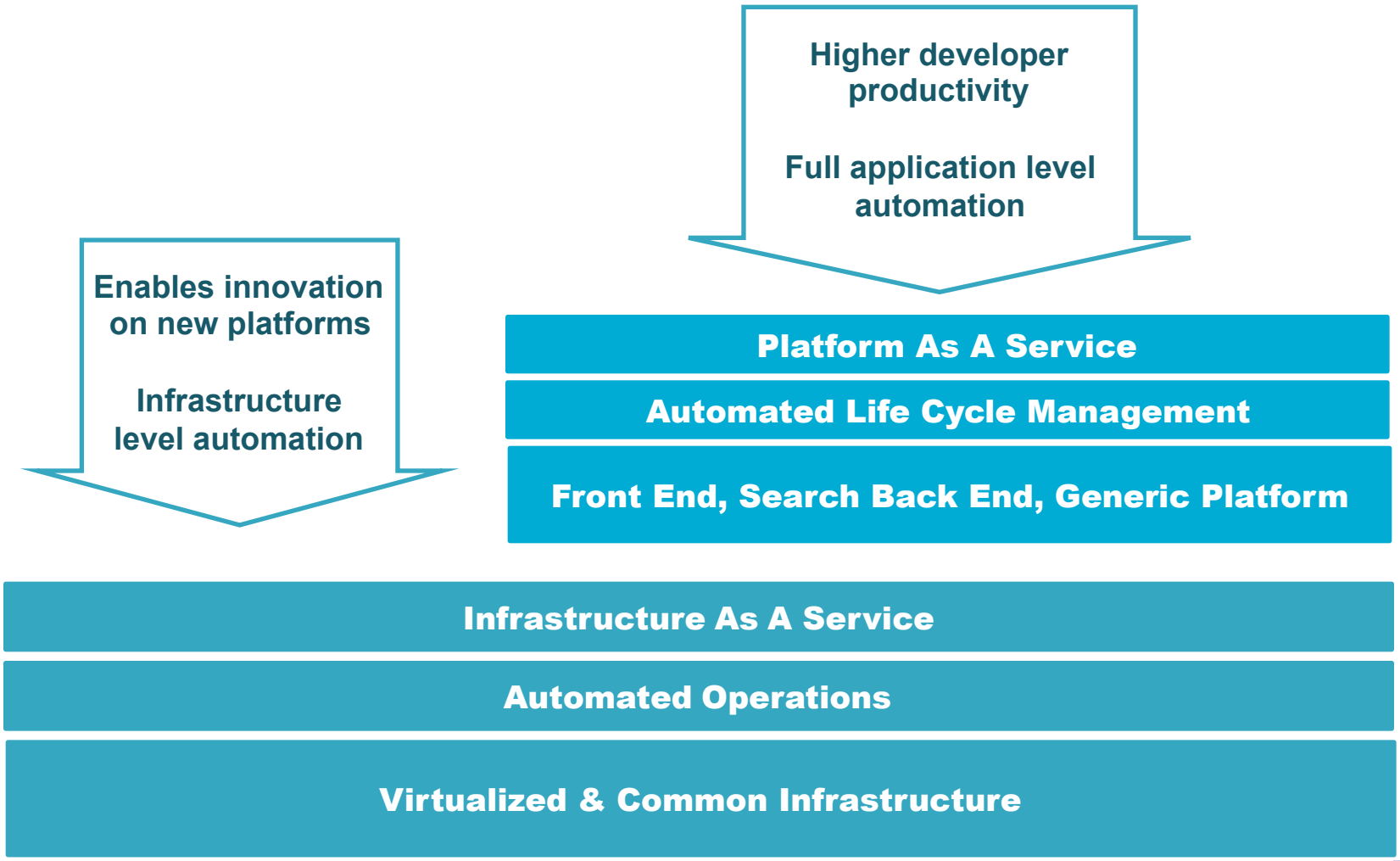


= Improved Time to Market

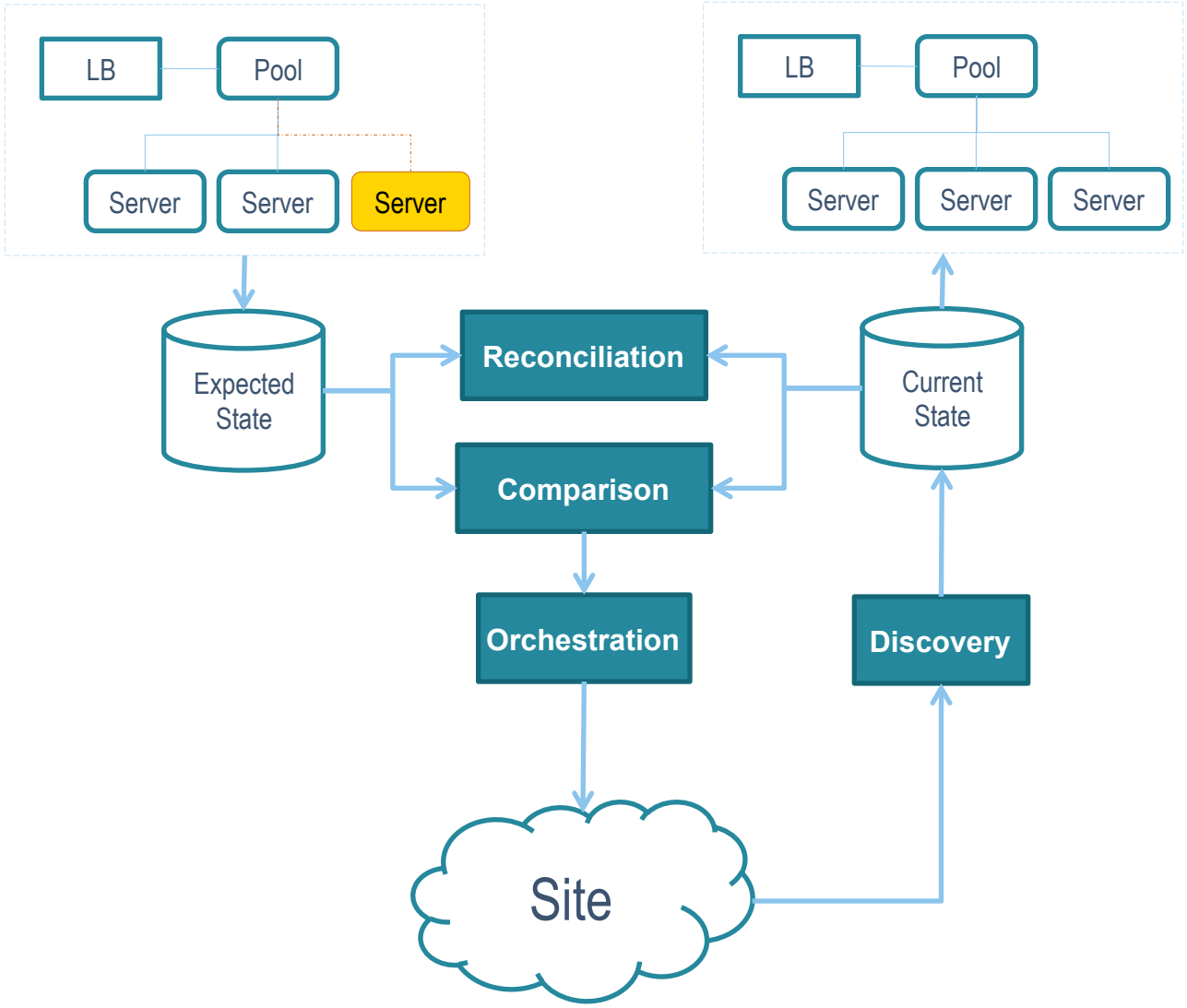
Architecture Decision



Infrastructure & Platform as a service

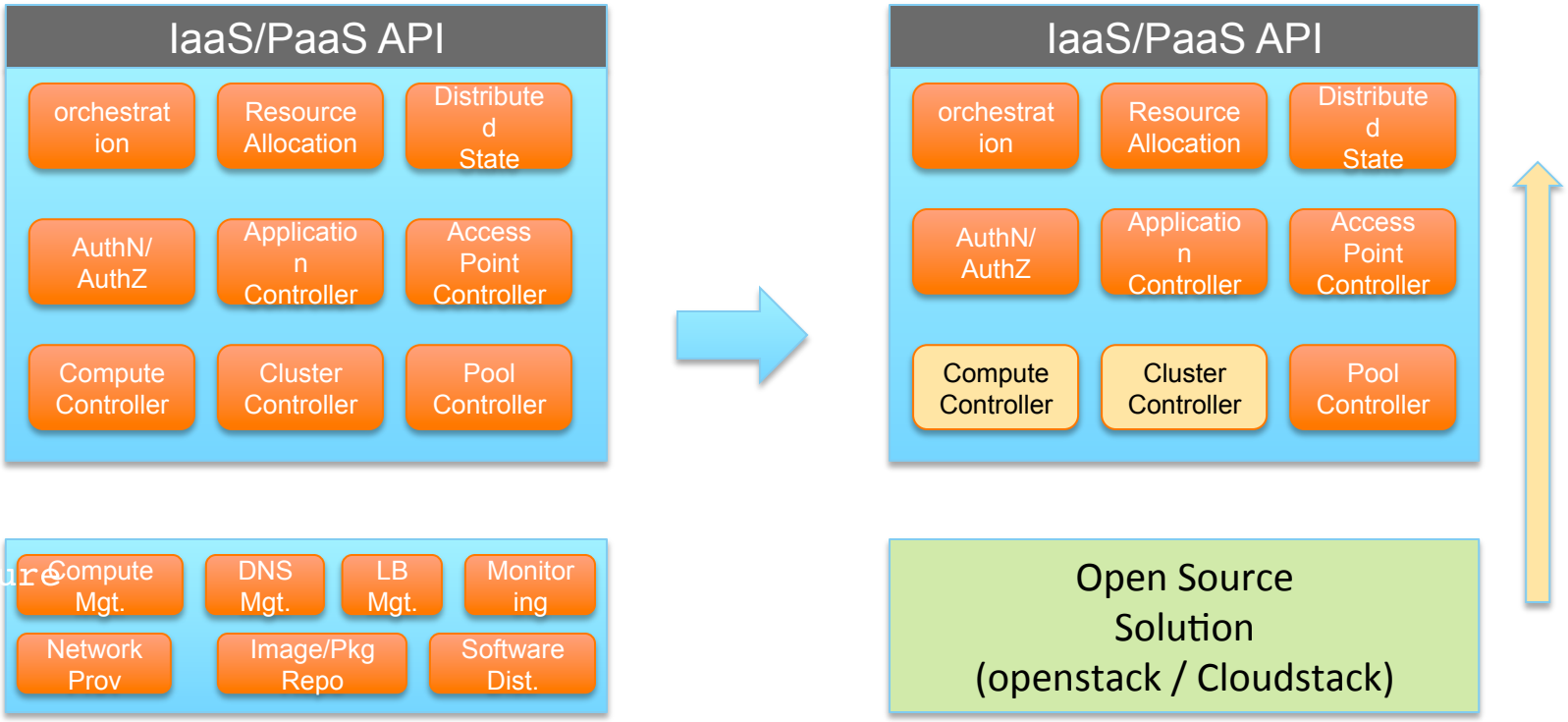


Model Driven Deployment Automation

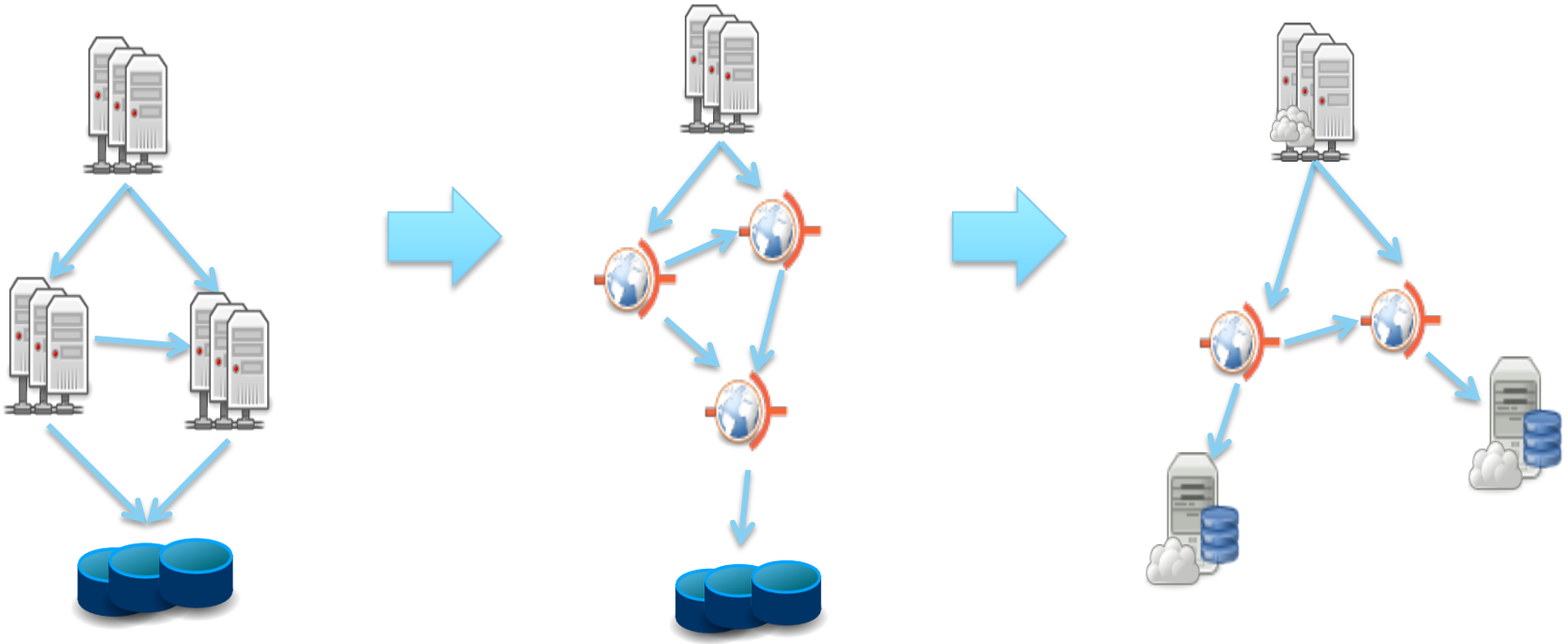


- Desired configuration is specified in the expected state and persisted in CMS
- Upon approval, the orchestration will configure the site to reflect the desired configuration.
- Updated site configuration is discovered based on detection of configuration events
- Reconciliation between the expected and current state allows to verify the proper configuration.
- On going validation allows the detection of out of band changes.

Open Source Integration



Application Architecture



Before

Ongoing
“Cloud
Friendly”

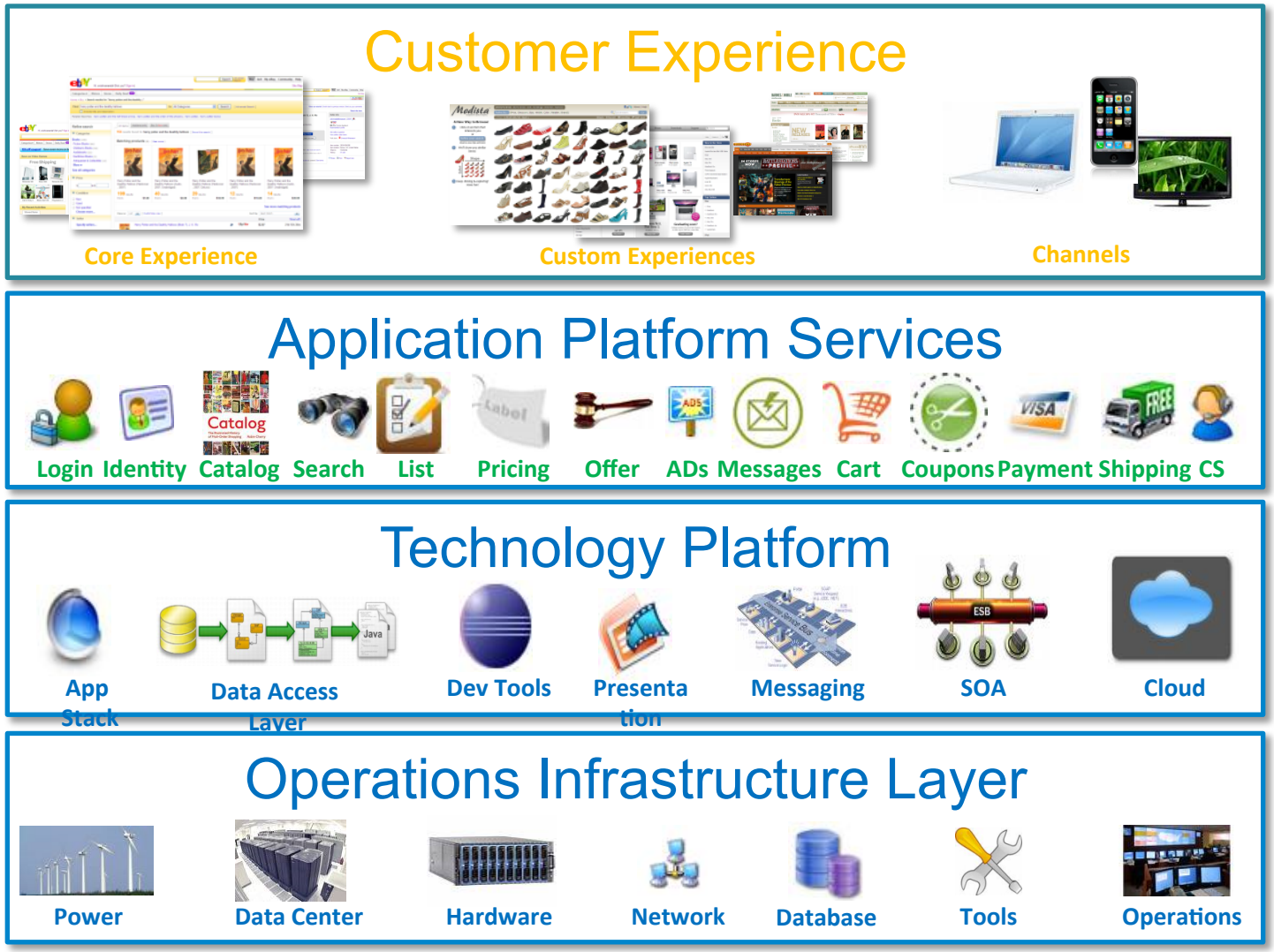
Future
‘Cloud
ready’

Next Gen Service Orientation

Services @ eBay

- It's a journey !
- History
  - One of the first to expose APIs /Services
  - In early 2007, embarked on service orienting our entire ecommerce platform, whether the functionality is internal or external
  - Support REST style as well as SOA style
  - Have close to 300 services now and more on the way
  - Early adopter of SOA governance automation (Discovery vs. control)

Architecture Vision



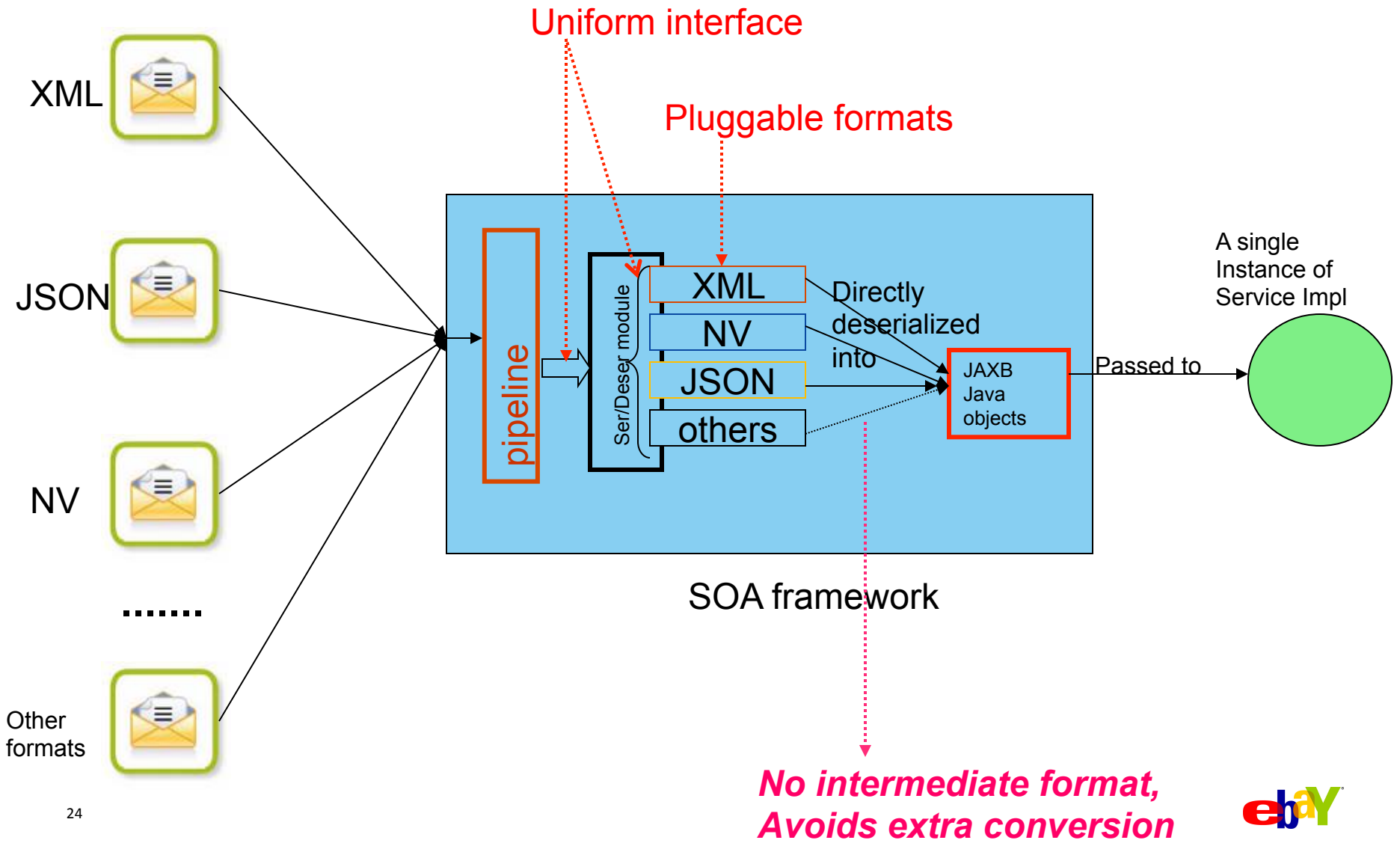
Challenges

- Multiple data formats
- Latency
- Service consumer productivity

Challenge 1: Multiple Data Formats

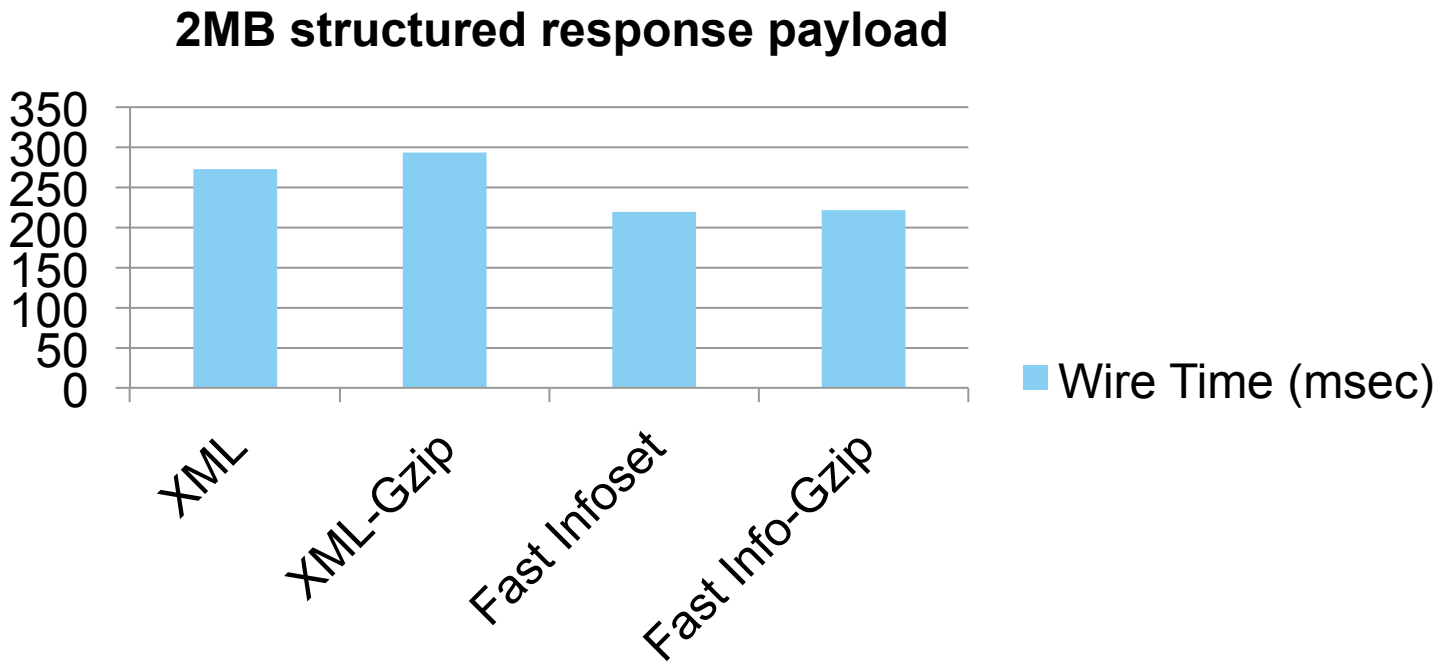
- Mix of user preferences
  - SOAP
  - XML / HTTP
  - JSON
  - Name-Value Pair (NV)
- Service developers **don't want to write extra code to do conversions**; too much maintenance impact
- Key observations:
  - Users ask for whatever data format they want.
  - Anything you can express in XML, you can express in other formats
  - Complete mapping from XML structures to NV and JSON

Solution: Pluggable Data Formats Using JAXB



Challenge 2: Latency

- For large datasets, there can be nasty latencies.
  - **Not** fixed by compressing or using Fast Infoset



Solution: Binary Formats

- Evaluated binary formats:
  - **Google Protocol Buffers, Avro, Thrift**
- Numbers look promising (serialization, deserialization)
- New challenges with these:
  - Each has its own **schema** (type definition language) to model types and messages
  - Each has its own **code generation** for language bindings
    - NOT directly compatible with JAXB beans
  - eBay SOA platform uses **WSDL/XML Schema (XSD)** data modeling, and **JAXB** language bindings

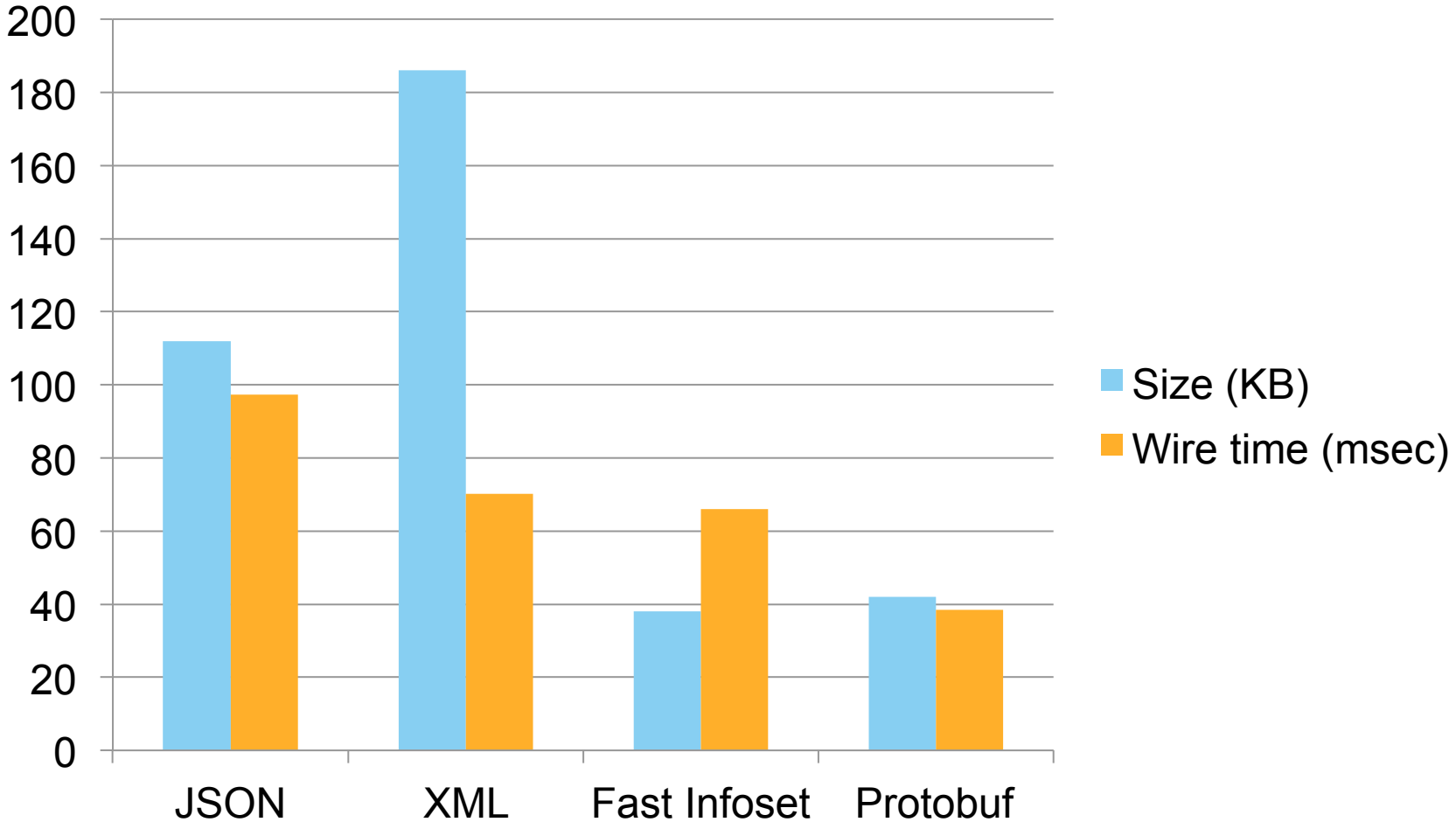
Compare Popular Binary Formats

Protobuf	Avro	Thrift
• Own IDL/schema • **Sequence numbers** for each element • Compact **binary representation** on the wire • Most XML schema elements are **mappable** to equivalents, except polymorphic constructs • Versioning is similar to XML, a bit more complex in implementing due to sequence numbers	• **JSON based** Schema • Schema prepended to the message on the wire • Compact **binary representation** on the wire • Most XML schema elements are **mappable** to equivalent, except polymorphic constructs • Versioning is easier	• Own IDL/schema • **Sequence numbers** for each element • Compact **binary representation** on the wire • Most XML schema elements are **mappable** to equivalents, except polymorphic constructs • Versioning is similar to XML, a bit more complex in implementing due to sequence numbers

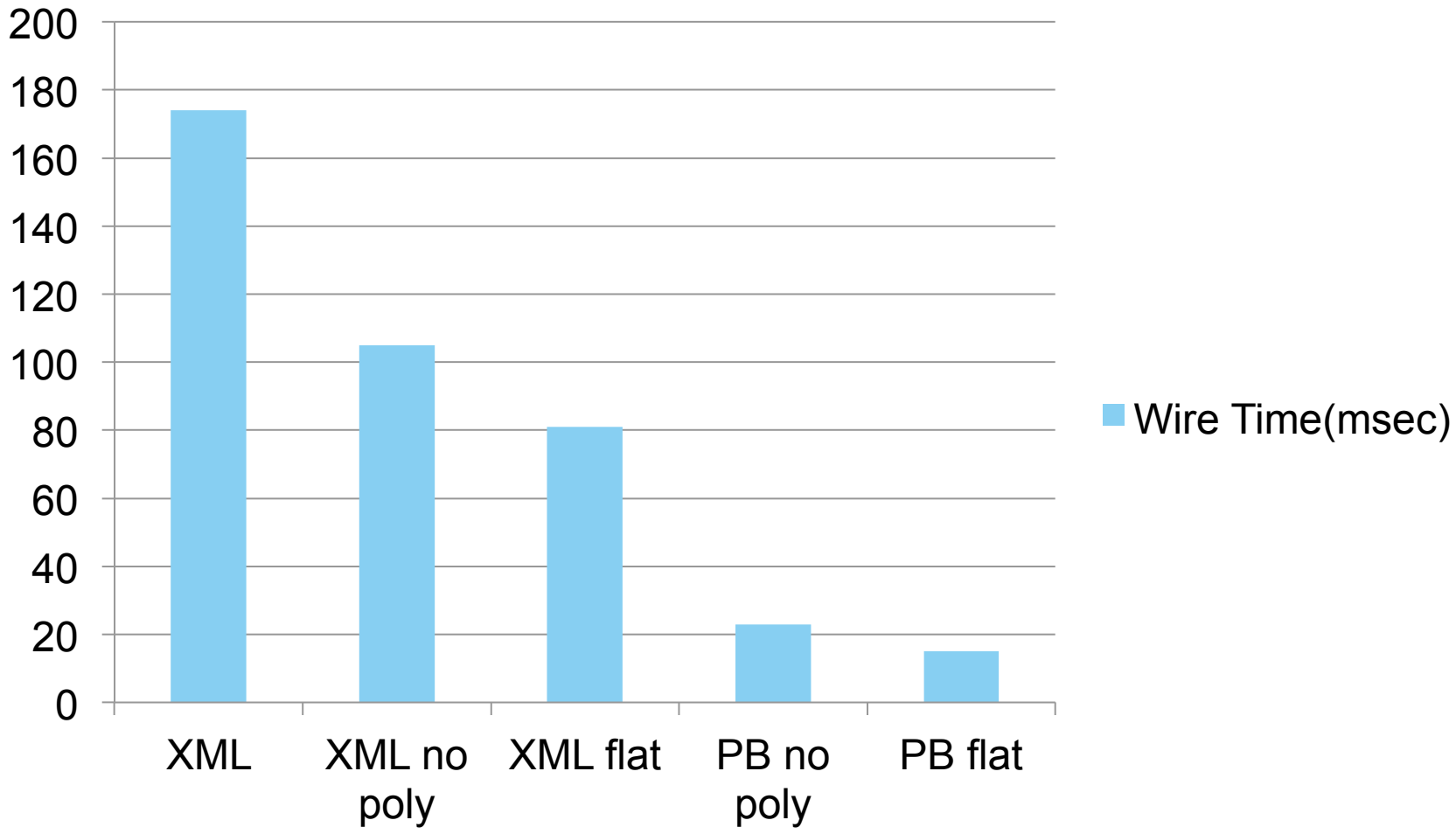
	Complex Types	Unions (Choice Type)	Self-References (Trees)	Enums	Inheritance / Polymorphism	Inline Attachment
Protobuf	Yes	No	Yes	Yes	No	No
Avro	Yes	Yes	Yes (with workaround)	Yes	No	No
Thrift	Yes	No	No	No	No	No
XML	Yes	Yes	Yes	Yes	Yes	Yes (MIME-TYPE)

Comparison of Data Formats

Response data: 50 items x 75 fields (about 8000 objects)



Latency Improvements



Challenge 3: Service Consumer Productivity

- Large, complex requests and responses
- Get exactly what they want in data returned from services
- Lack of consistency in service interface conventions and data access patterns
- Real client applications make calls to multiple services at a time
  - Serial calls increase latency. Managing parallel calls is complex
- Impedance mismatch between service interface and client needs
  - Too much data is returned
  - 1 + n calls to get detailed data

Sneak Preview: ql.io

- New technology from eBay
- Plan to open source soon
- SQL + JSON based scripting language for aggregation and orchestration of service calls
- Filtering and projections of responses
- Async orchestration engine
  - Automatic parallelization, fork / join

What ql.io Enables

- **Create consumer-controlled interfaces**
  - fix/patch APIs on the fly
- **Filter and project responses**
  - use a declarative language
- **Bring in consistency**
  - offer RESTful shims with simpler syntax
- **Aggregate multiple APIs**
  - such as batching
- **Orchestrate requests**
  - without worrying about async forks and joins

ql.io Examples

- Simple Select
  - select * from ebay.finding.items where keywords='ipad'
- Field Projections
  - select title, itemId from ebay.finding.items where keywords='ipad'
- Sub-Select
  - select e.Title, e.ItemID from ebay.item.details as e where e.itemId in (select itemId from ebay.finding.items where keywords = 'ipad')

ql.io Batch Example

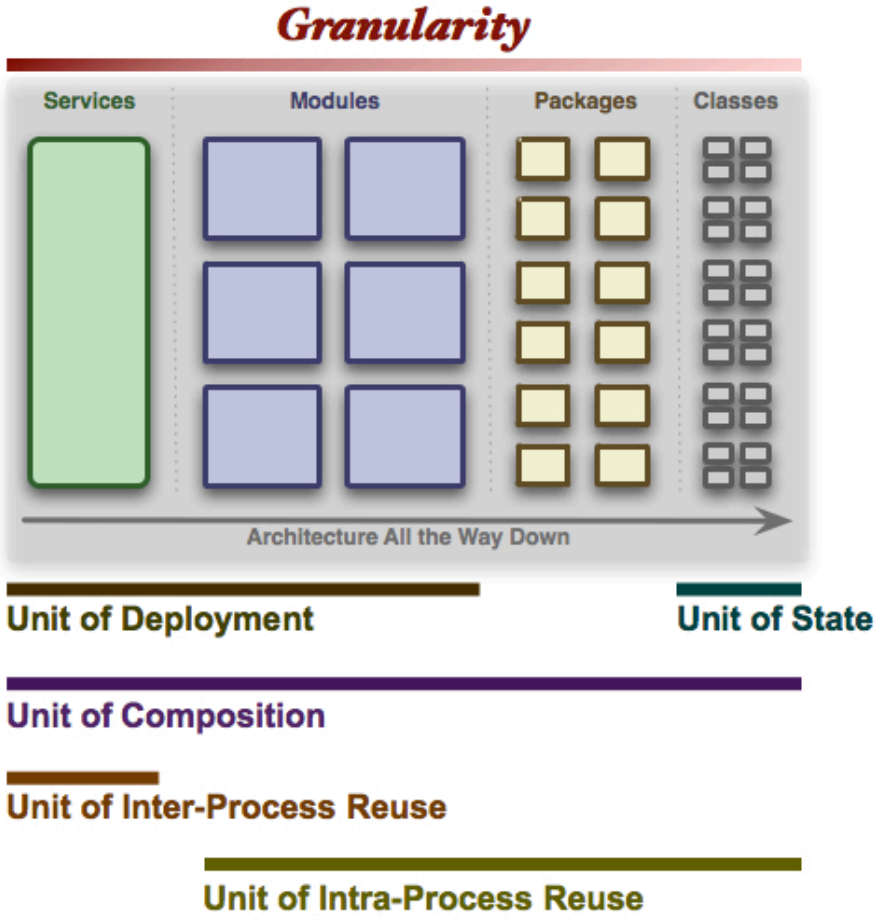
```
itemId = select itemId from ebay.finding.items where keywords = 'ferrari' limit 1;
item = select * from ebay.shopping.singleitem where itemId = '{itemId}';
user = select * from ebay.shopping.userprofile where userId = 'sallamar';
tradingItem = select * from ebay.trading.getitem where itemId = '{itemId}';
bestOffers = select * from ebay.trading.bestoffers where itemId = '{itemId}';
bidders = select * from ebay.trading.getallbidders where itemId = '{itemId}';
return {
  "user" : "{user}",
  "item" : "{item}”,
  "tradingItem" : "{tradingItem}",
  "bidders" : "{bidders}",
  "bestOffers" : "{bestOffers}"
};
```

ql.io Demo

Modularity

Key modularity concepts for software

- Building blocks
- Re-use
- Granularity
- Dependencies
- Encapsulation
- Composition
- Versioning



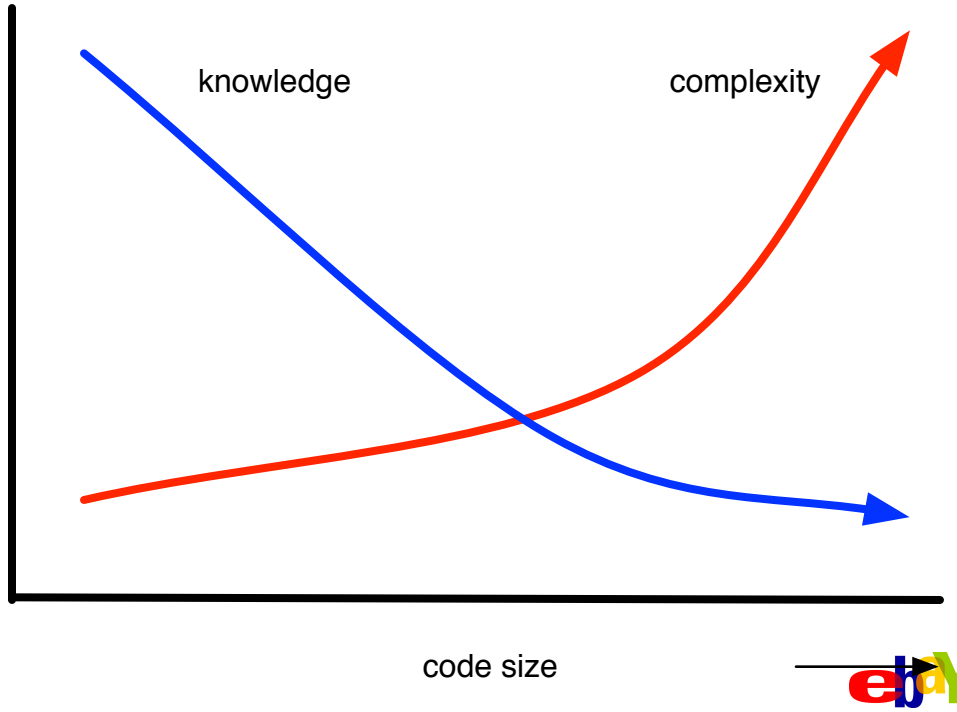
Source: http://techdistrict.kirkk.com/2010/04/22/granularity-architectures-nemesis/
Author: Kirk Knoernschild

Challenges for Large Enterprises

- Some stats on the eBay code base
  - ~ 44 million of lines of code and growing
  - Hundreds of thousands of classes
  - Tens of thousands of packages
  - ~ 4,000+ jars
- We have too many dependencies and tight coupling in our code
  - Everyone sees everyone else
  - Everyone affects everyone else

Challenges for Large Enterprises

- Developer productivity/agility suffers as the knowledge goes down
  - Changes ripple throughout the system
  - Fallouts from changes/features are difficult to resolve
  - Developers slow down and become risk averse



Our Goals with Modularity Efforts

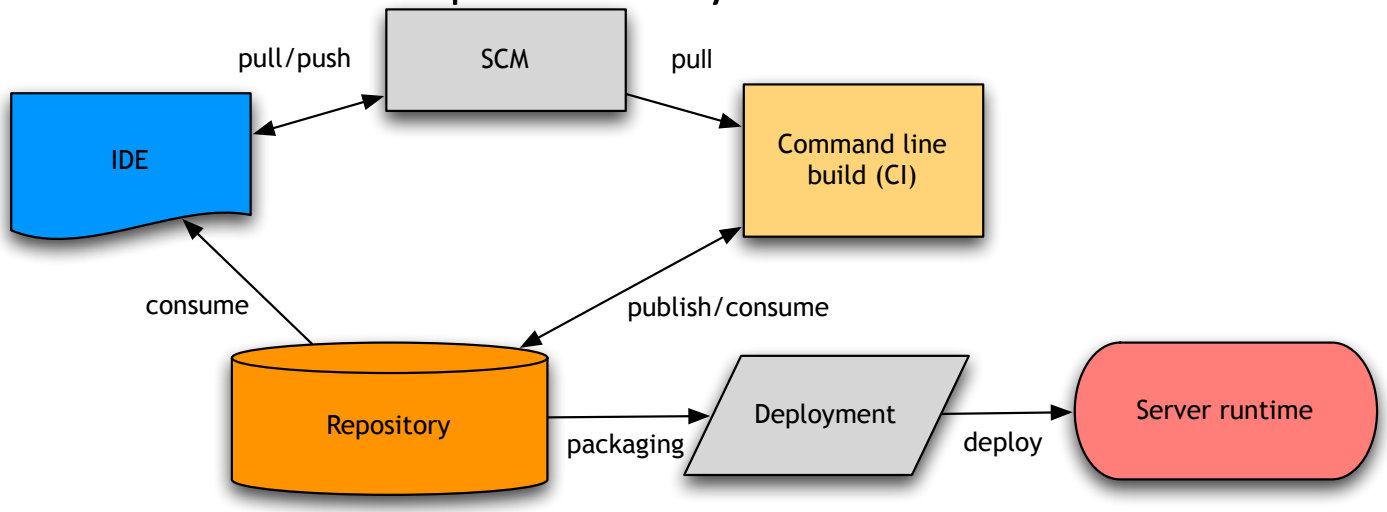
- Tame complexity
- Organize our code base in loose coupling fashion
  - Coarse-grained modules: number matters!
  - Declarative coupling contract
  - Ability to hide internals
- Establish clear code ownership, boundaries and dependencies
- Allow different components (and teams) evolve at different speeds
- Increase development agility

Modularity Solutions Evaluation

- Evaluated OSGi, Maven, Jigsaw and JBoss Module
- Criteria include:
  - Modularity enforcement
  - End-to-end development
  - Migration concerns
  - Adoption
  - Maturity
- Selected OSGi

OSGi @ eBay

- Modularize platform into OSGi bundles with well-defined imports and exports
- Challenges: split packages, Classloader contructs
- Source to binary dependencies
- Refresh **end-to-end** development life cycle



Lessons Learned

- OSGi learning curve is still fairly steep
  - large group of developers with varying skill levels
- End-to-end development lifecycle
  - Tools may not work well together. Leverage OSGi tools like bnd
- Conversion/migration of existing code base
  - Not starting from vacuum
  - Cost to rewrite / refactor code
  - We cannot afford disruption to business meanwhile: “change parts while the car is running”
- Semantic versioning adoption is important

Overall Summary

- Strategies
  - Deployment Agility: Automation with Cloud
  - Development Agility: Next gen Service Orientation
  - Taming complexity: Modularity
- Systems quality & scalable architecture as key foundation
- Complexity management and developer productivity becomes increasingly important
- Strike balance between agility and stability

eBay Open Source

- eBay has been a strong supporter of Open Source model and community

- Check out http://eBayOpenSource.org
  - Mission is to open source some of the best of breed technologies that were developed originally within eBay Inc.
  - Under a liberal open source license.
  - These projects are generic technology projects and several years of development effort has gone into them to mature them.

(Coming soon)


ebaY