IBM Sterling Selling And Fulfillment Suite - 9.2.1 - Scalability Capabilities

Eugene Amigud, OM Architect, IBM

Agenda

Database Sharding

Splitting the database into smaller, faster, more easily managed parts

Search Server

A server that provides indexing capabilities to quickly search the order data

Promising and Inv. Hub

A separate component responsible for inv. and availability

Product Principles

Use leading edge, yet proven technology to deploy applications for multi-enterprise commerce applications delivering:

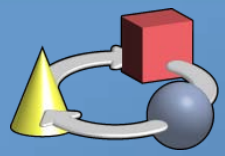
an Architecture for High Availability and Scalability of Mission Critical Solutions

that provide Flexibility and Adaptability to agile business processes, and

are deployable in Complex Multi-Enterprise Organizations

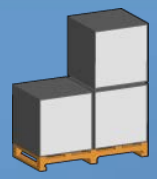
Interoperability

- ✓Native XML
- ✓XML Mapping & Transformation
- ✓Standards such as EDI, RosettaNet, SOAP



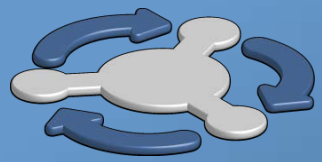
Extensibility

- ✓User Interface
- ✓Business Logic
- ✓Data Model
- ✓Functional Component Independence



Scalability

- ✓Architected for high throughput
- ✓N-tier, component-based architecture
- ✓Leverages leading App Servers



Modularity

- ✓Horizontal and Vertical
- ✓Independently deployable
- ✓Independently upgradable

IBM's Commerce Architecture Vision

Channel Applications


Online


Mobile


Call Center


Store


Kiosks


Field Sales


Email

Cross-channel Commerce Services

Master Data

Item
Customer
User
Store
Supplier
Warehouse

Sales & Marketing

Catalog
Pricing
Contracts
Marketing
Promotion
Coupons
Configuration
Loyalty
Rebates

Transaction Management

Cart
Wishlist
Gift Registry
Quote
Order
Shipment
Delivery
Work Order
Payment
Return
Purchase Order

Promising & Inventory

Inventory
Capacity
Promising
Sourcing
Scheduling
SC Network

Fufillment

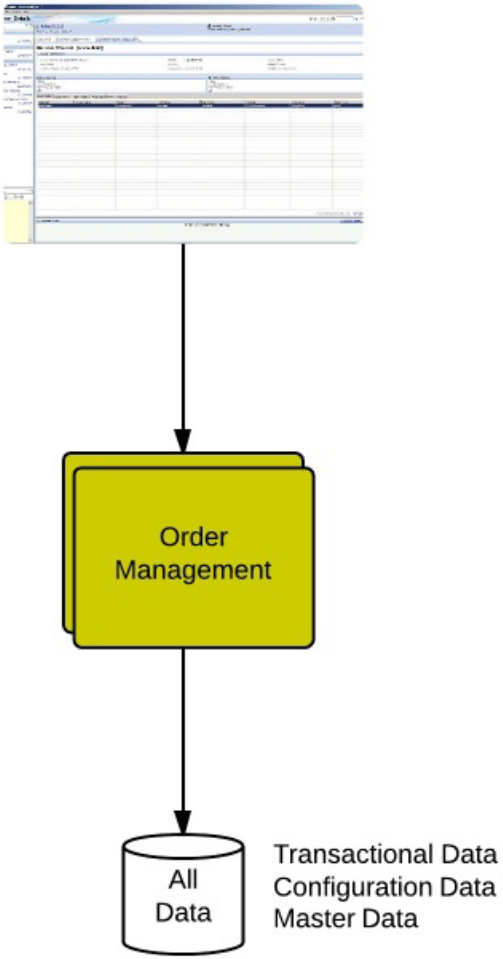
WMS
TMS
Yard
Supplier Portal
Supply Chain
Visibility

Other Services

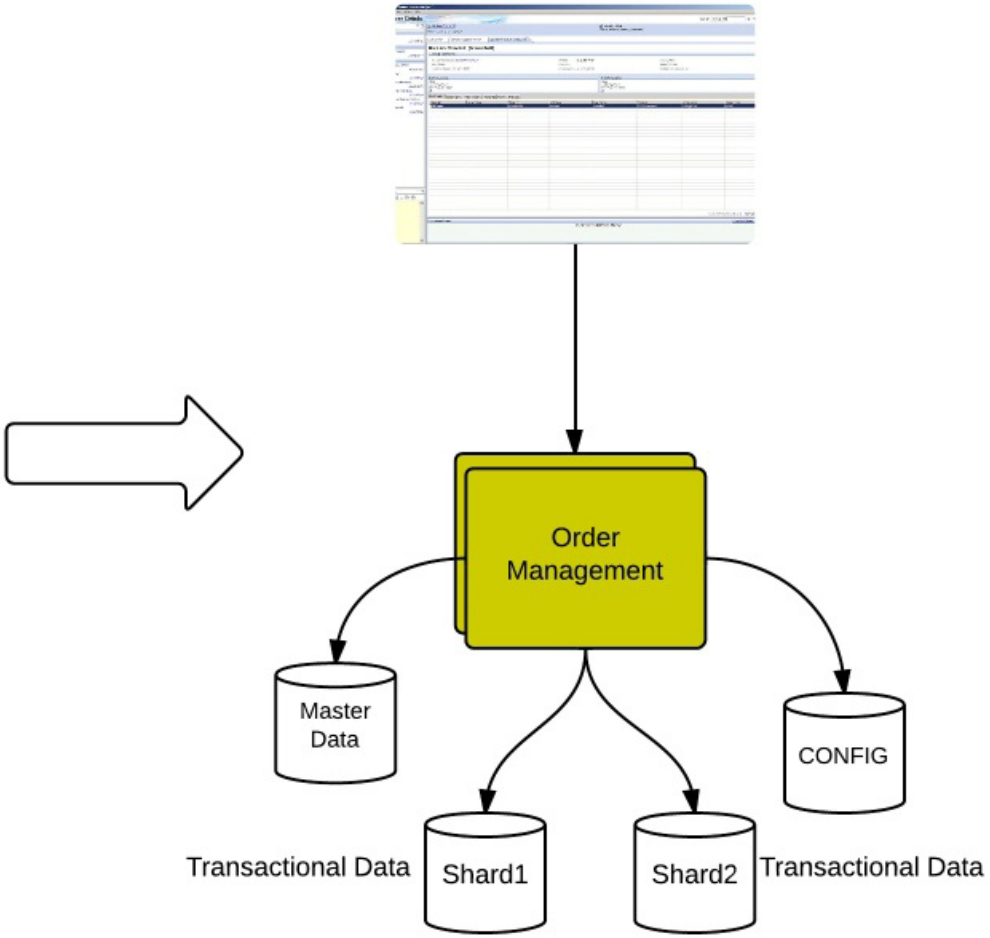
Payment Gateway
Vault
Tax
Fraud Detection
Address Verif
Carrier Execution
Email
SMS
……..
……..

Database Sharding

Traditional Deployment



Sharded Deployment

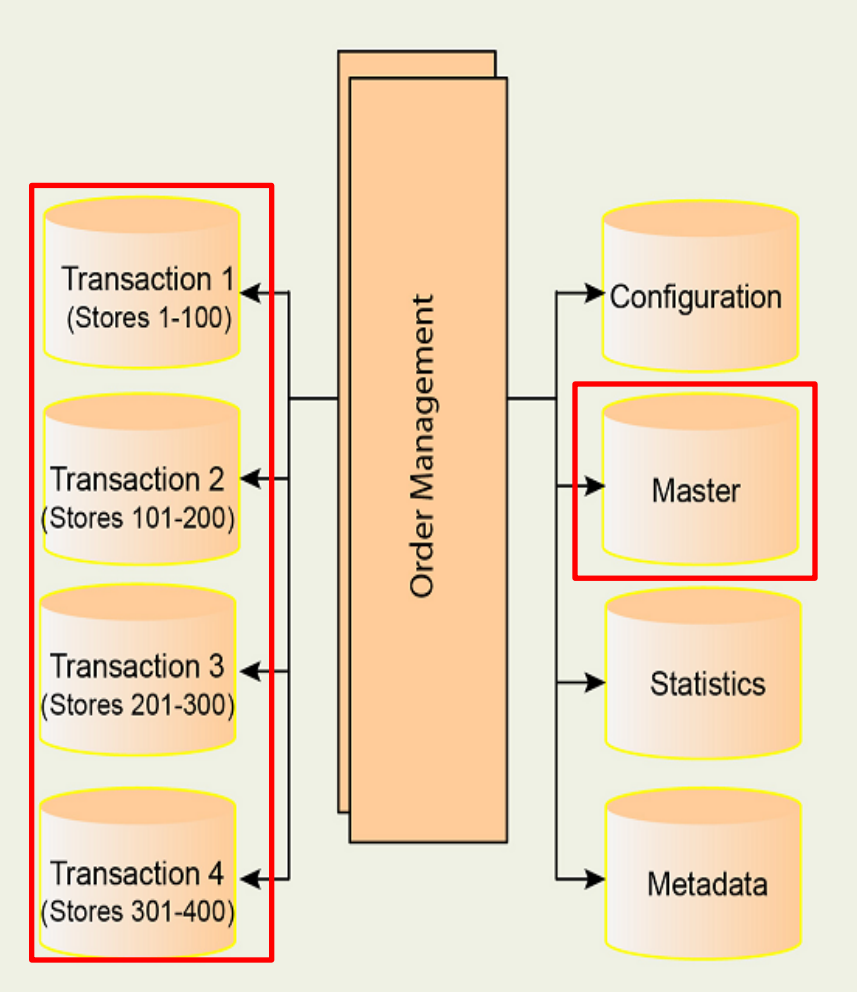


Sharding In IBM Sterling OMS

- Sterling Selling and Fulfillment Foundation supports the following types of single-instance, sharded deployments

  - ✓ Sharding By enterprise
  - ✓ Sharding By Seller
  - ✓ Sharding by custom attributes

Sharding In Action



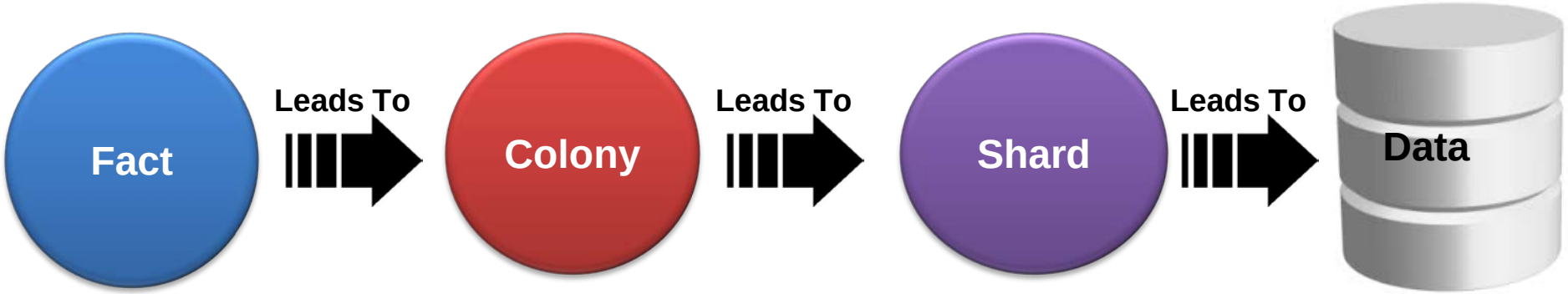
- The Transaction 1 database contains transaction data for stores on the east coast (stores 1-100)

- The Transaction 2 database contains transaction data for stores on the west coast (stores 101-200)

- The Transaction 3 database contains transaction data for stores in central US (stores 201-300).

- The Transaction 4 database contains transaction data for stores in southern US (stores 301-400).

Data Access in Sharded Environment



createOrder API is invoked in OMS with below input

<Order OrderNo="Order12" EnterpriseCode="S-MART" DocumentType="0001" SellerOrganizationCode="Store123"/>

How to determine the corresponding database shard for this order ?

Comparison

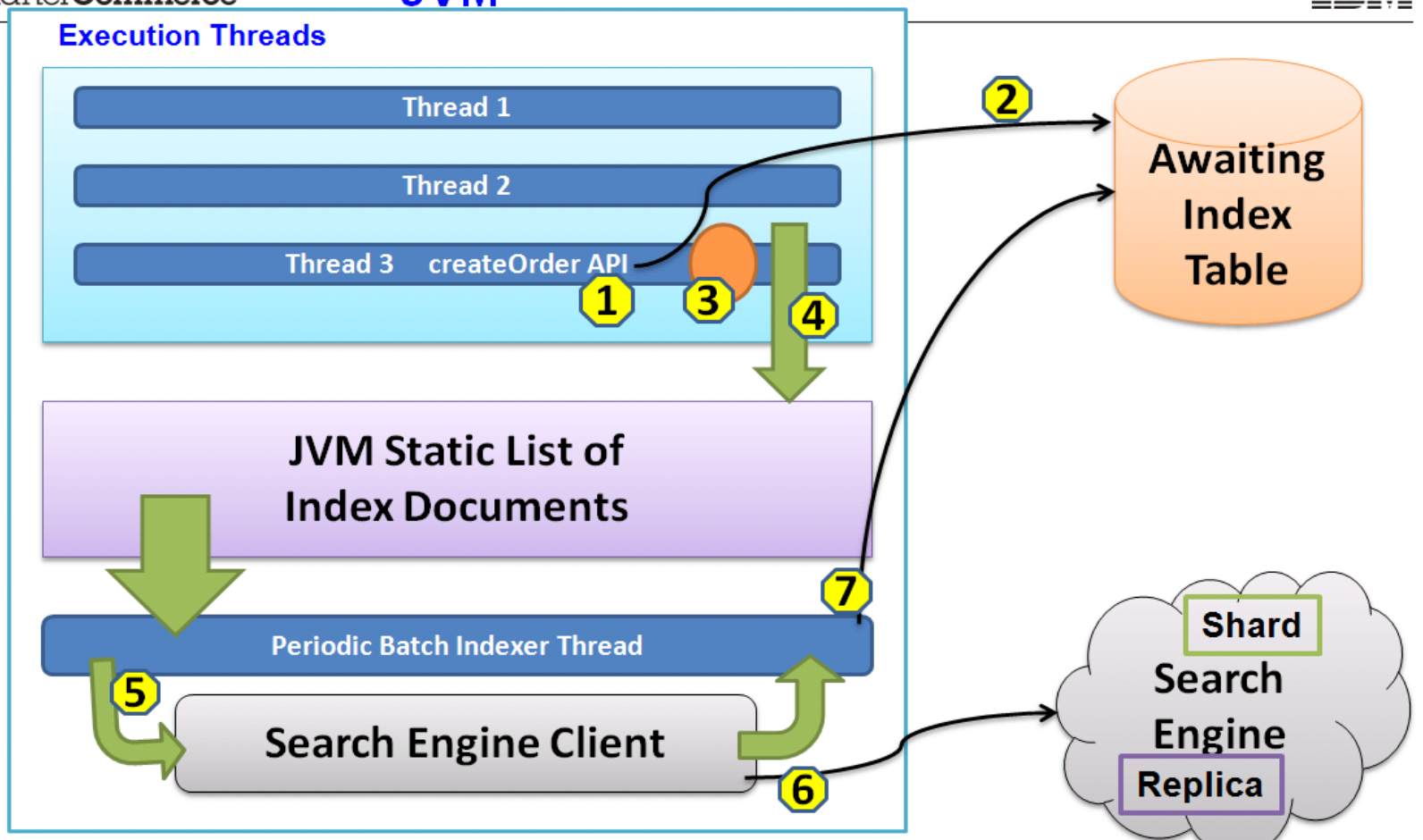
	Traditional	Clustered DB	DB Sharding
Scalability	Limited by one single DB	Potentially limited by clustered DB overheads	Potentially much higher scalability (until one shard becomes bottleneck)
Complexity	Simple, well understood	Medium complexity but reasonably well understood	Higher complexity and less understood though wide acceptance in high volume web sites. Shards need a promising server.
Application	No change	Minor changes	DB sharding implemented in the application layer
Domain level routing or distribution	None	None	Workloads can be distributed using domain attributes

Search Server

IBM Sterling OMS Search Server

- Central location maintains transactional entities
  - Orders
  - Shipments
- Accepts documents, returns Shard #s
- Uses Elastic search
  - Distributed restful search engine
  - Apache Lucene based
  - Indexing capabilities for transactional data

JVM



1. Order updated, index dirty
2. Insert temp. record, create document
3. Commit
4. Submit document for indexing
5. Periodically submit batch to search engine client
6. Update index
7. Upon successful indexing, remove temp record

Search Server – Retrieving Order

Get all orders of customer with Phone # 8885555555

getOrderList

Sterling Order Management

Query the Search server with phone number

Search server returns all the colonies where order(s) with phone number exists

Retrieve the orders from those colonies, aggregate them and return it back to the customer

Search Server

Colony-Web

Web-Shard

Colony- Mobile

Mobile-Shard

Colony- POS

POS-Shard

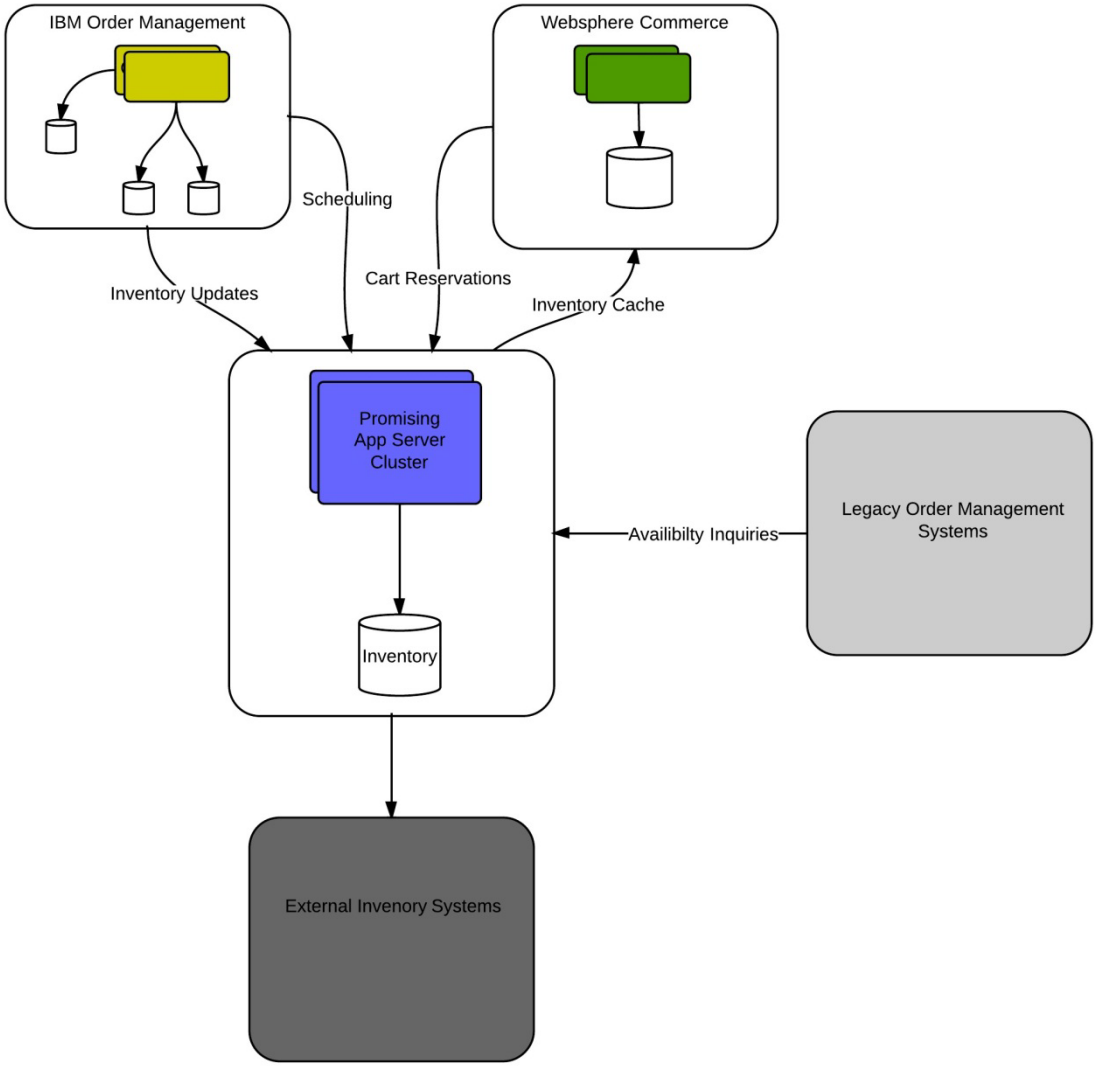
Parallel processing

- Search can return multiple shards
- APIs will be called in parallel to retrieve data
- Parallel processing in Sterling OMS uses EJB 3.1 async beans, Internal thread pools
- Parallel processing is supported only for the List API
- The application server must be upgraded to the version that supports EJB 3.1

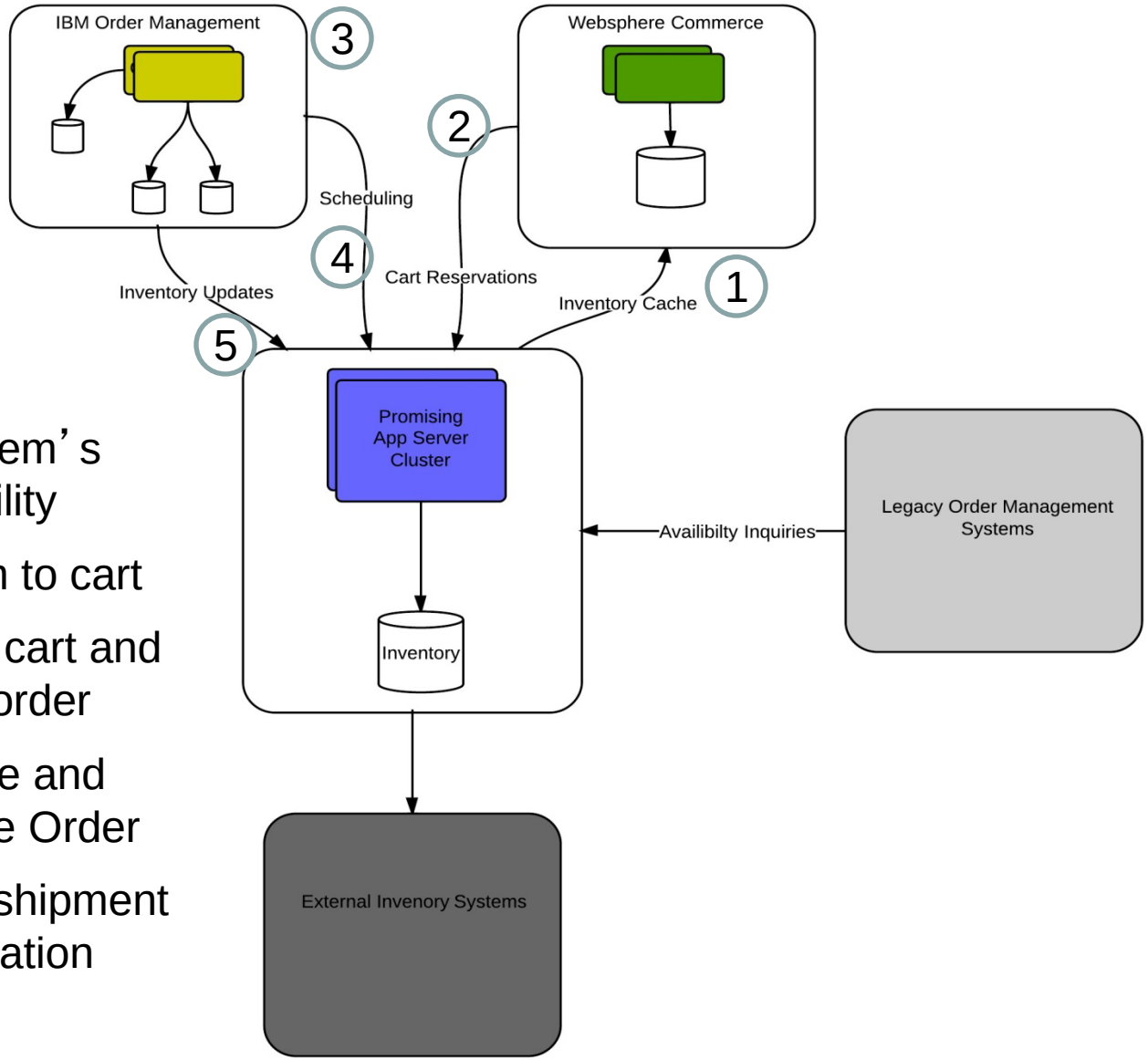
Promising Server

Promising and Inventory Hub Architecture

- Centralized supply and demand
  - Supports external inventory
- Configurations
  - Sourcing Rules
  - Scheduling Rules
  - Inventory Rules
- Demands on the system are modeled as reservations
- Provides various ATP services
- Real Time Availability Monitor

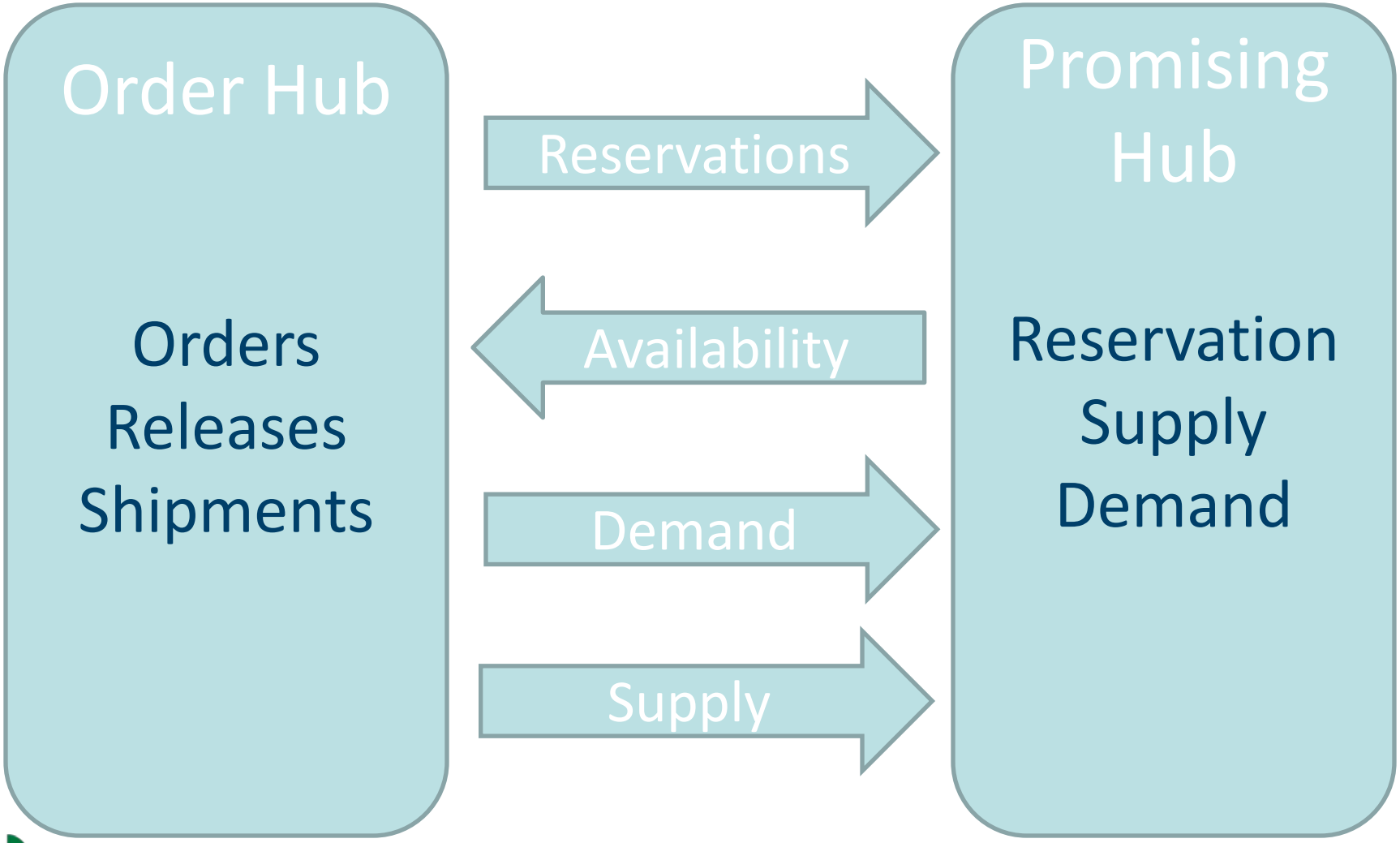


Promising and Inventory Hub Architecture



1. Check item's availability
2. Add item to cart
3. Confirm cart and create order
4. Schedule and Release Order
5. Record shipment confirmation

Order and Promising Hub Interaction



Promising and Inventory Hub

- Advantages
  - Consistent Promise
  - Enables order sharding
  - Integration with legacy OM systems
  - Runs as an standalone component
  - Out of the box integration
- Limitations
  - Another instance to maintain
  - Item Based Allocation Agent is not supported
  - Provided/Delivery services are not supported

Scalability vs Consistency

CAP (Brewer's) theorem

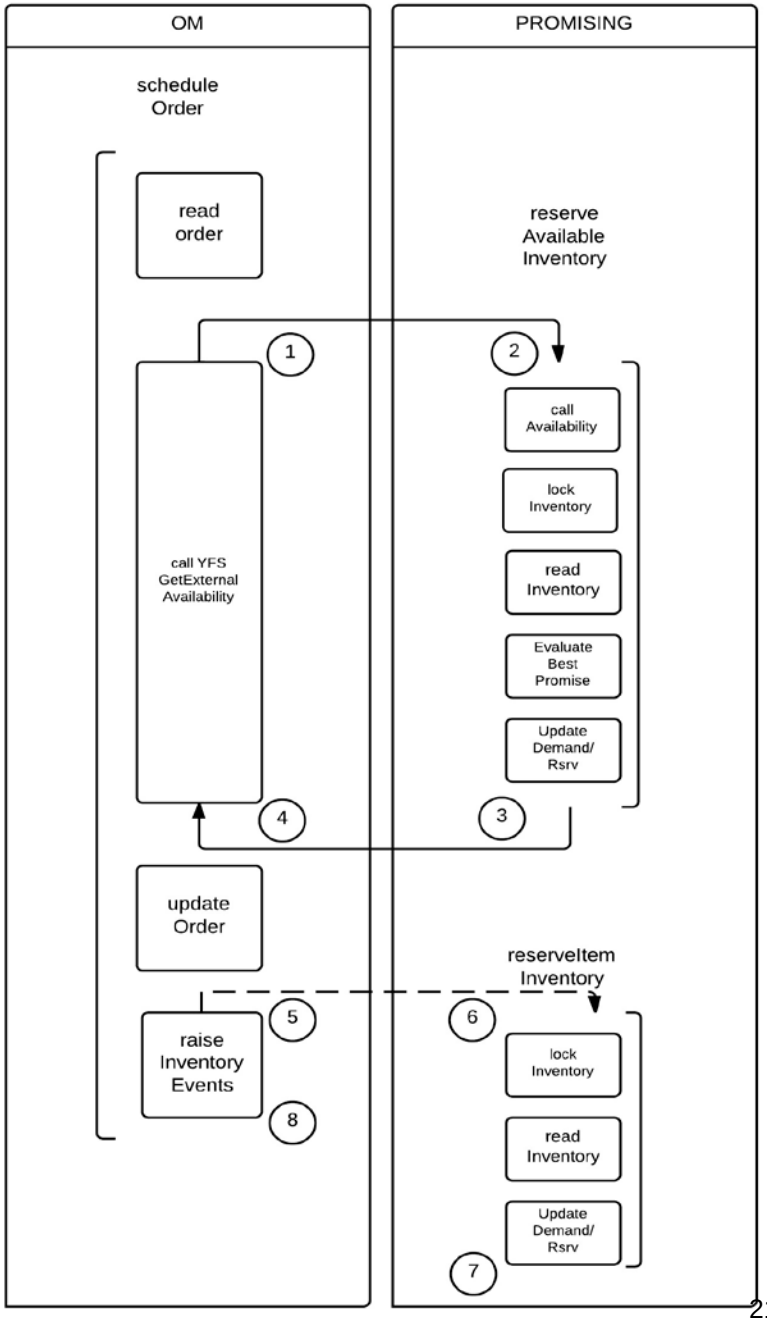
Impossible for a distributed computer system to simultaneously guarantee:

- *Consistency*
- *Availability*
- *Partition tolerance*

Use of BASE

Basically **A**vailable **S**oft-state services with **E**ventual-consistency

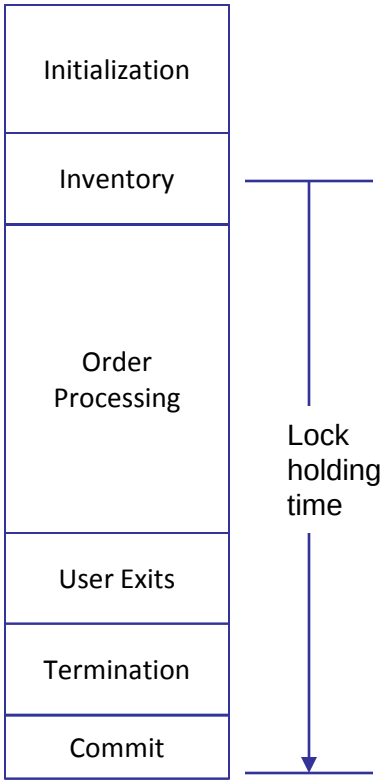
- Reservations are created during inquiry to guarantee stock (1,2)
- Reservations expire if hand-shake does not occur (failure between 4,5)
- Inventory Events are self correcting in case some messages are lost (5,6)



Reduction in Lock Contention

Order Transaction in a Single OMS Instance

OMS Instance



Probability of Inventory Locking is a factor of

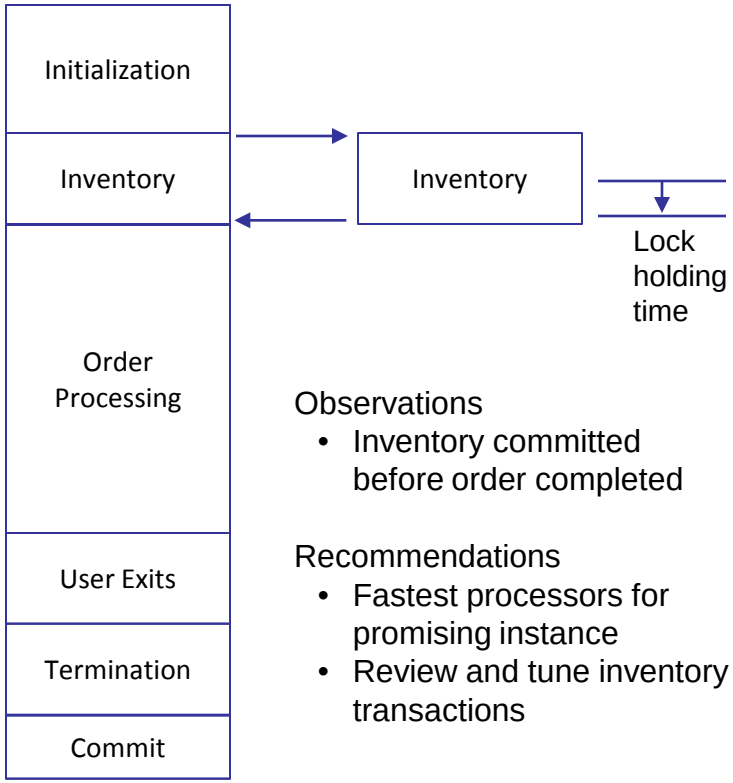
- Concurrency Levels
- Presence of Common Items
- **Lock Holding Times**

Lock Holding Times is a factor of how fast transactions take to process

- Slow processors
- Virtualization
- Slow User Exits – e.g., Long calls to external systems
- Size of order
- Record locking

Order Transaction in an OMS Instance with a Promising Server

OMS Instance **Promising Instance**



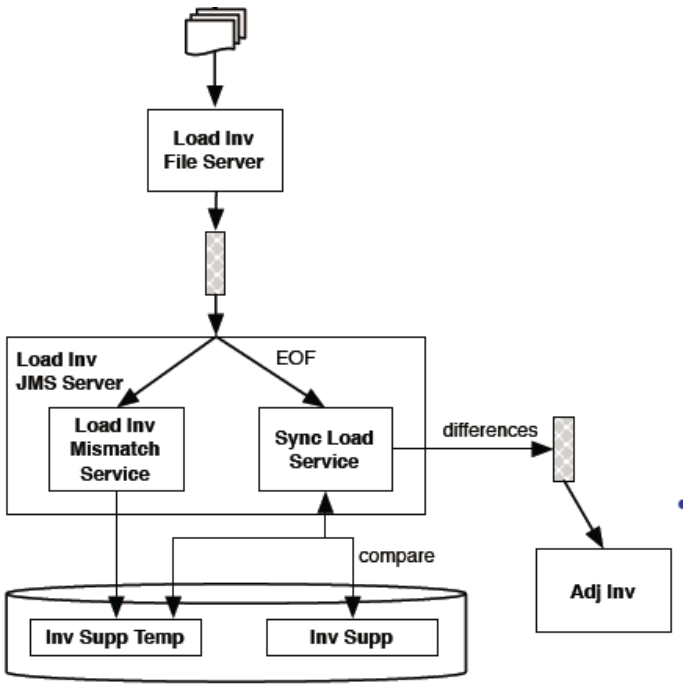
Observations

- Inventory committed before order completed

Recommendations

- Fastest processors for promising instance
- Review and tune inventory transactions

Inventory Agents

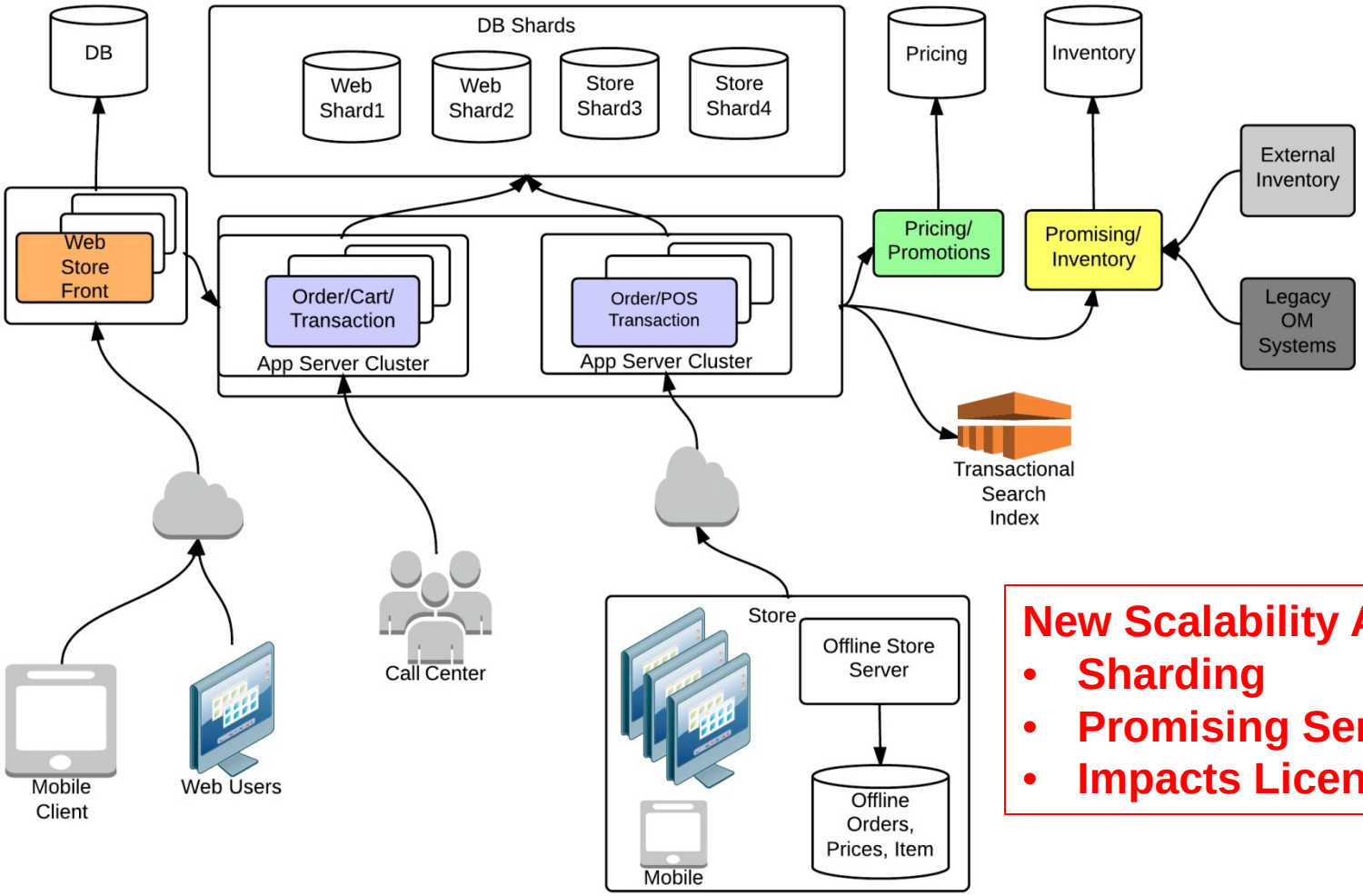


- Load Inventory
  - JDBC Batch Inserts (Temp table)
  - Event contains multiple messages
  - Adjust inventory can process batches of changes
- RTAM
  - Optimized getter query
  - Eliminated duplicate triggers

Benchmark Scenarios

	Peak Hour	Uniform Sustained	Night Batch
Duration	1	5*	4
OLTP Workloads			
• Orders per Hour	1.5M orders/hr	420K orders/hr	-
• Order Lines per Hour	2.7M lines/hr	756K lines/hr	-
• Shipments per Hour	-	630K shipments/hr	-
Inventory Updates			
• Inventory Updates	-	7.25M inv upd/hr	27.2M inv upd/hr
• Inventory Adjustments	-	7.25M inv adj/hr	5.4M inv adj/hr
Web Store			
• findInventory per Hour	3M calls/hr	840K calls/hr	0
• reserveAvailableInventory per Hour	300K calls/hr	84K calls/hr	0

Scalable Deployment 9.2.1



New Scalability Add-On

- **Sharding**
- **Promising Server**
- **Impacts Licensing**