IBM Software

WebSphere eXtreme Scale

Agenda

- IBM Vision
- Introduction and Scaling Challenges
- Caching Scenarios
- WebSphere eXtreme Scale spotlight
- Client Usage
- Summary

Section

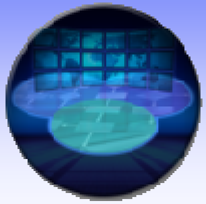
IBM Vision

Application Infrastructure Portfolio

Adoption Patterns

IBM Offerings

Application Foundation



WebSphere Application Server Family
WebSphere sMash
IBM CICS® Transaction Server

Intelligent Management

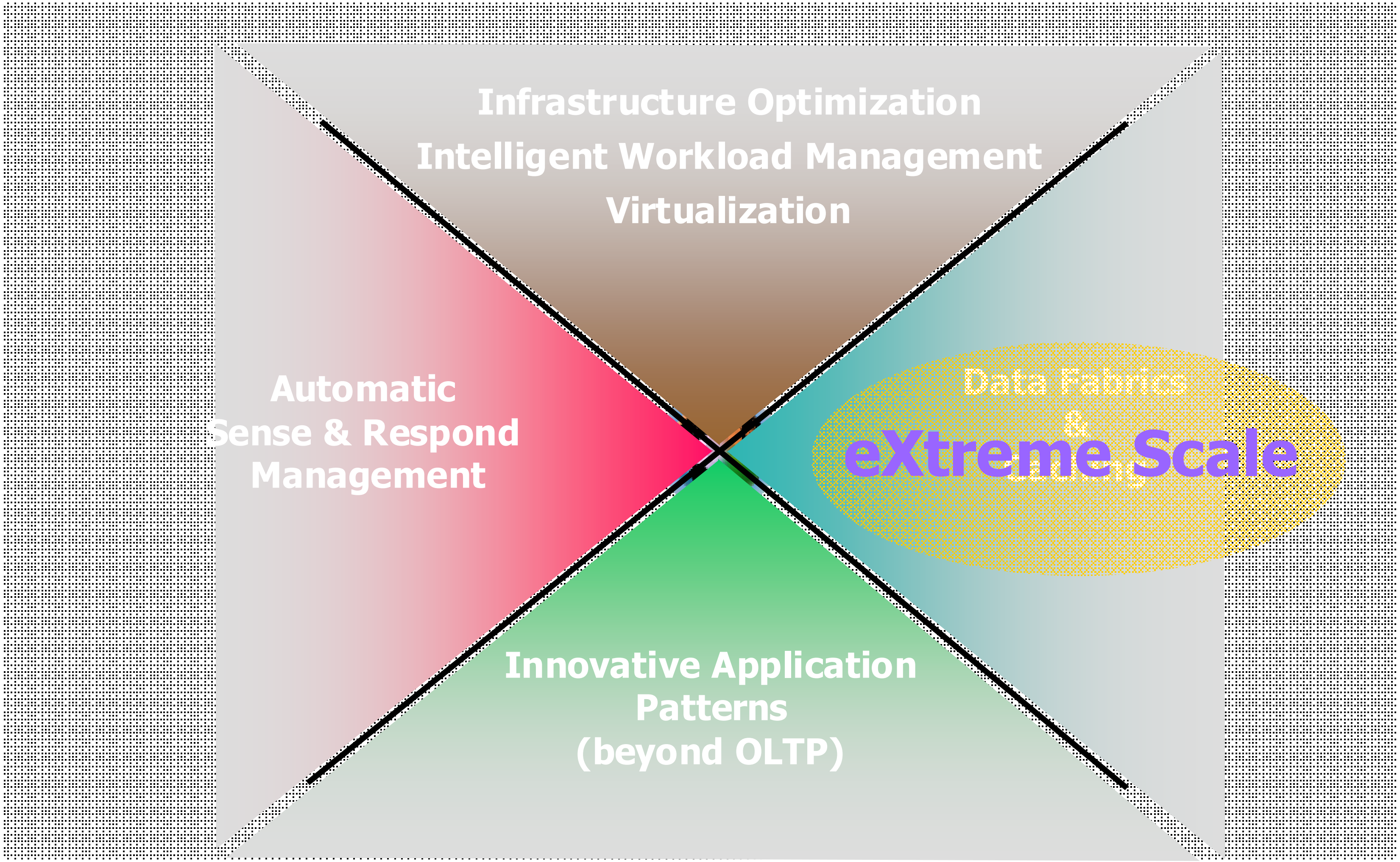
IBM workload Deployer
WebSphere Virtual Enterprise

Extreme Transaction Processing

WebSphere eXtreme Scale
WebSphere Real Time
WebSphere Compute Grid

What is WebSphere XD?

Software to virtualize, control, and turbo-charge your application infrastructure



Infrastructure Optimization
Intelligent Workload Management
Virtualization

Automatic
Sense & Respond
Management

Data Fabrics
&
Caching

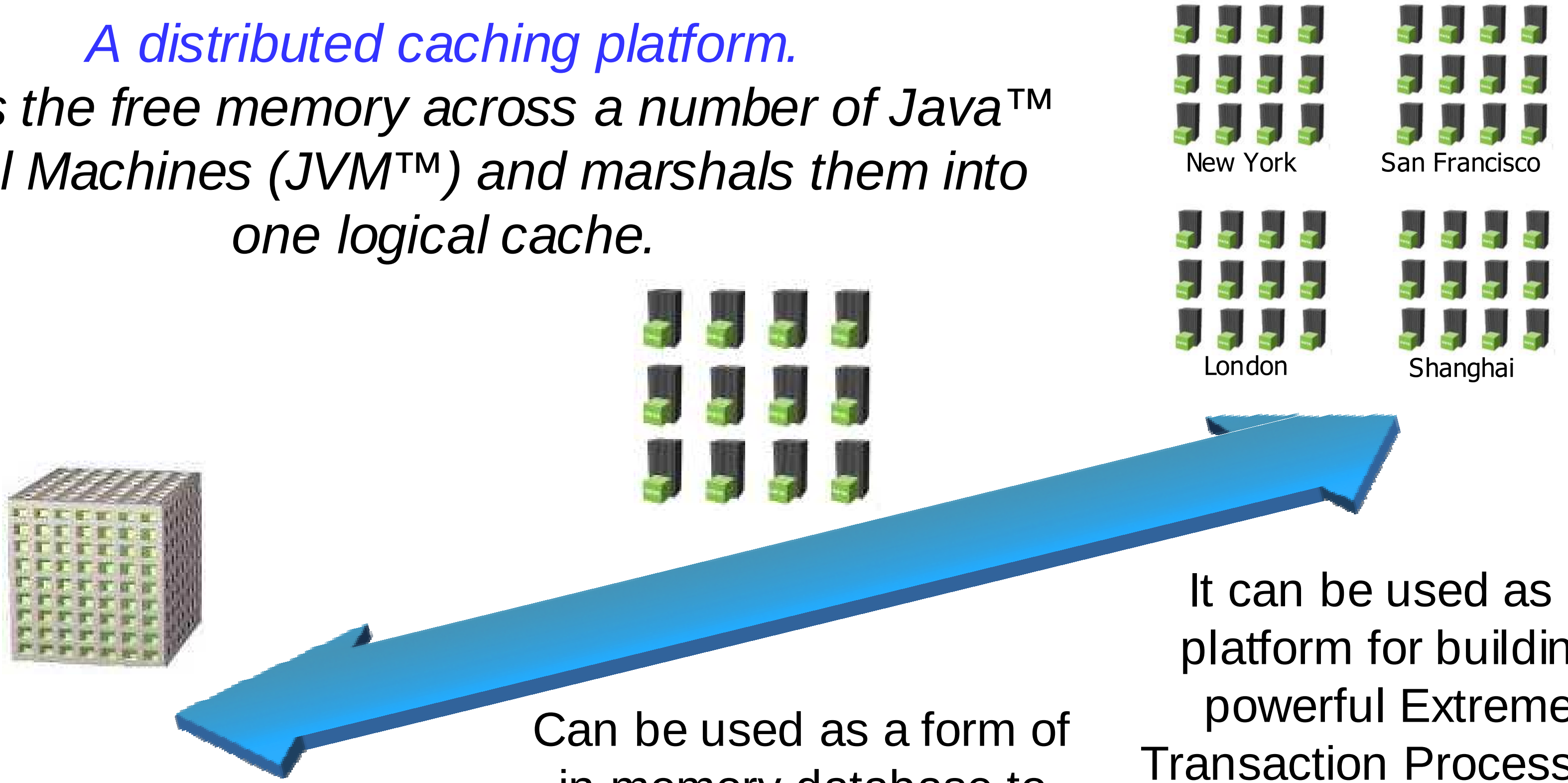
eXtreme Scale

Innovative Application
Patterns
(beyond OLTP)

What is WebSphere eXtreme Scale?

A distributed caching platform.

It takes the free memory across a number of Java™ Virtual Machines (JVM™) and marshals them into one logical cache.



Scales from simple in-process topologies to powerful distributed topologies.

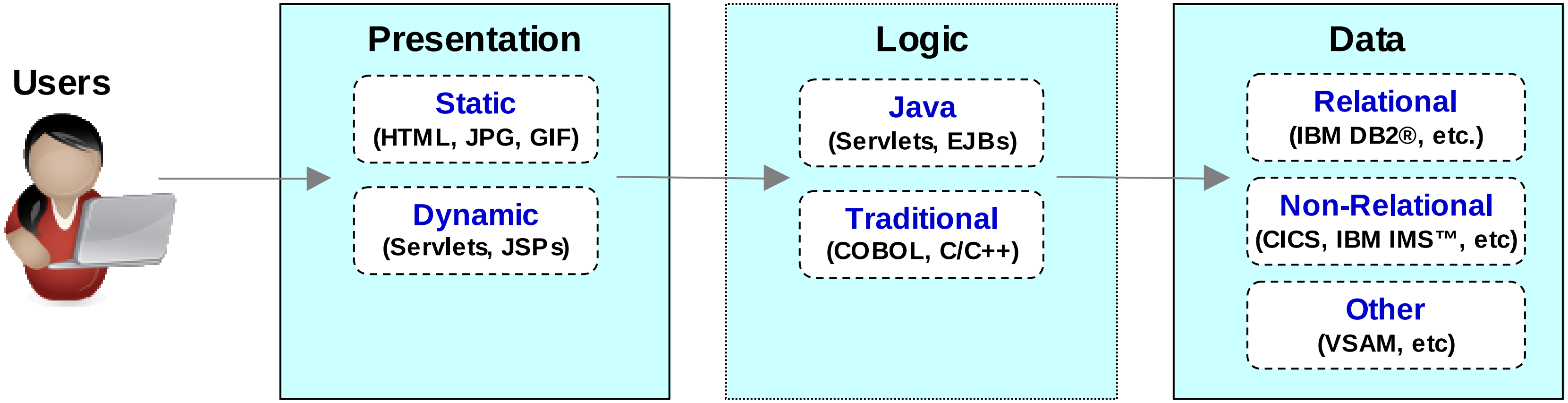
Can be used as a form of in memory database to manage application state.

It can be used as a platform for building powerful Extreme Transaction Processing (XTP) / Data Grid applications.

Section

Introduction and Scaling Challenges

Standard Online Transaction Processing Model



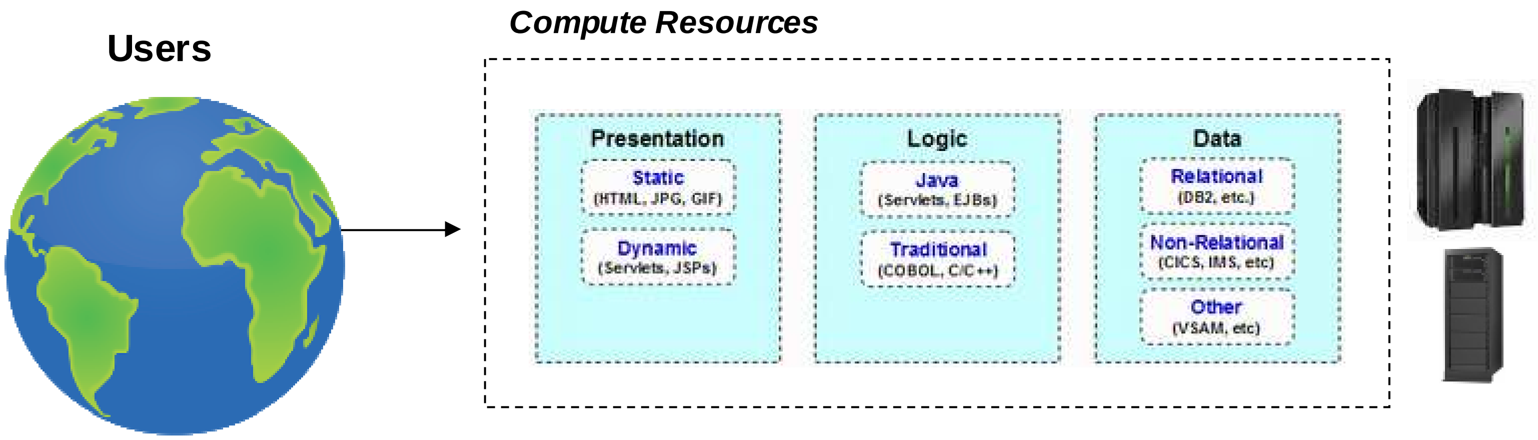
This is the "three tier" model you may have heard about

It may be physically three tier or physically one tier

***Logically* it's three tier … that's the key**

Works and works well as long as the demands placed on it don't outstrip some element of this … which often happens when scaled up

Internet Has Opened Up a World of Users



Users

Compute Resources

Presentation

Static
(HTML, JPG, GIF)

Dynamic
(Servlets, JSPs)

Logic

Java
(Servlets, EJBs)

Traditional
(COBOL, C/C++)

Data

Relational
(DB2, etc.)

Non-Relational
(CICS, IMS, etc)

Other
(VSAM, etc)

Think about some of the *very large* models in today's world:

Online banking

Online retailing

Auction sites

Social networking sites

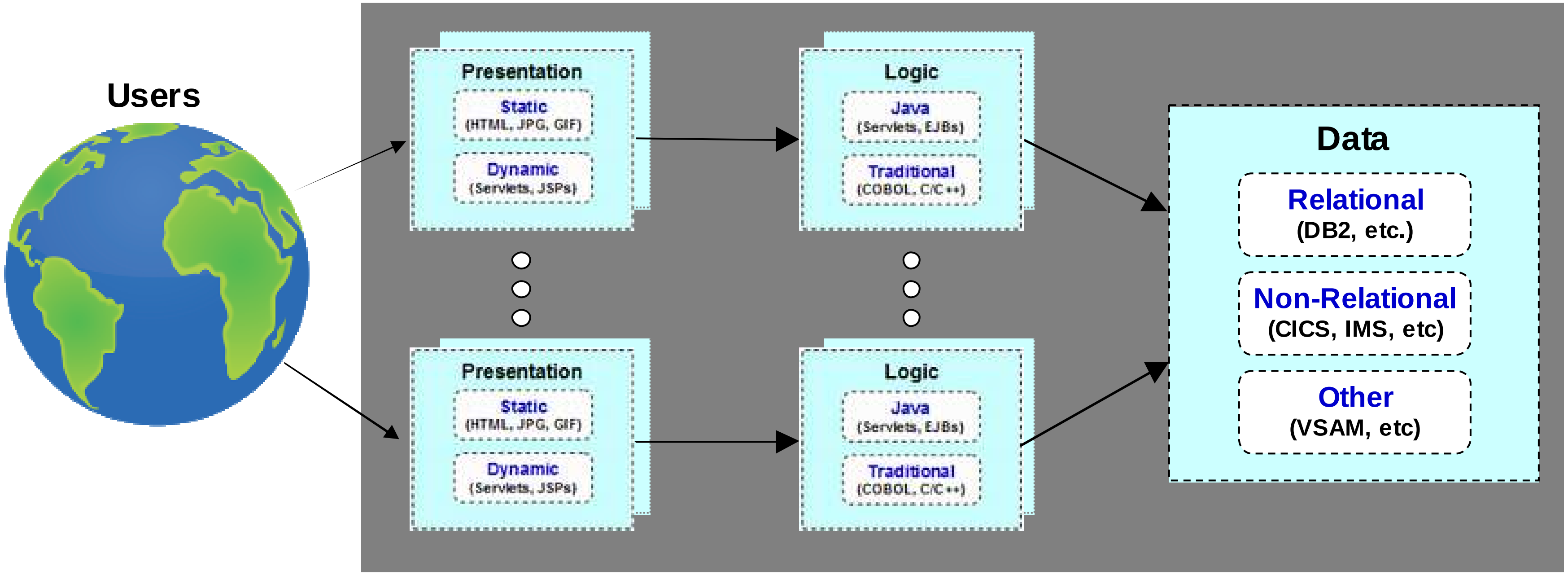
Potentially multi-millions of simultaneous users

These numbers put enormous strain on the resources … need to *scale*

**We're using "very large" here to set the stage to explain why data grids came about.
However, eXtreme Scale can provide cost saving value in *any* sized environment.**

Scaling the Presentation and Logic Layer

Here’s what we often see in response to work demands overwhelming some element of the design:

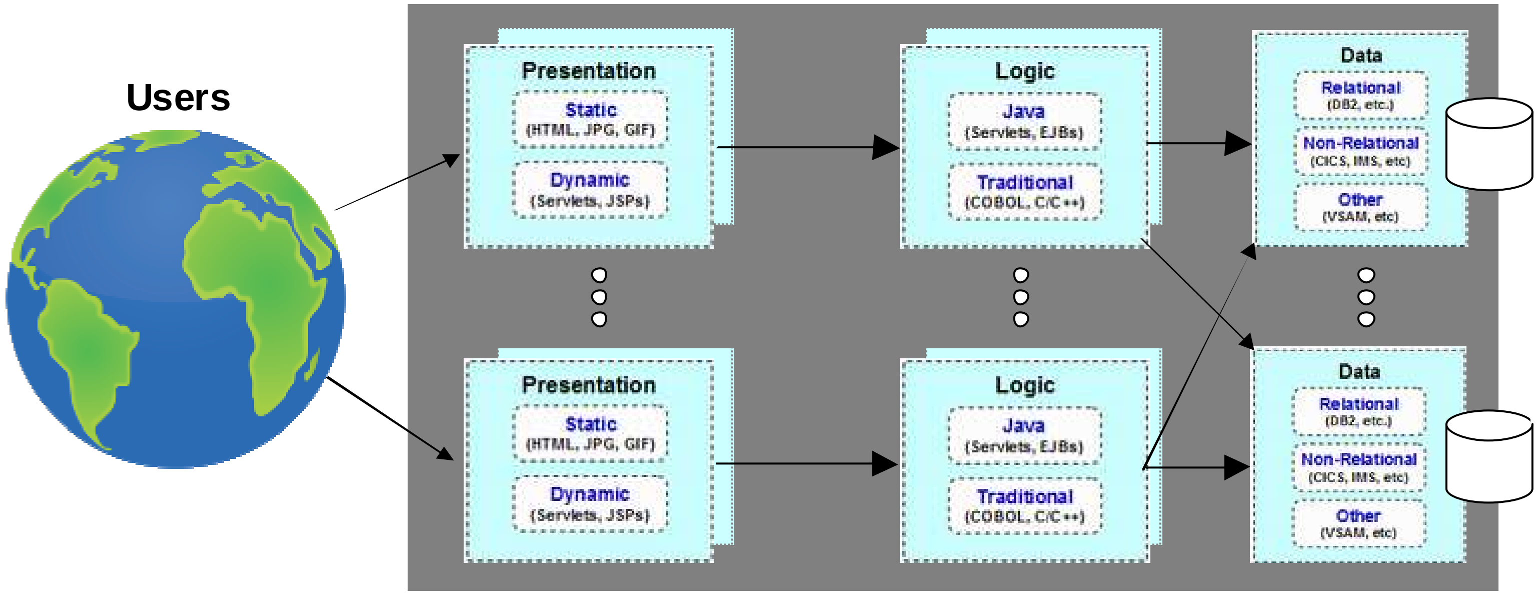


Scaling out the presentation and logic layers spreads the work

The data layer is a another matter … more difficult to “spread out”

Scaling the Data Layer

Logical three tier … physical one or many



See legend below

Share the data -- this is what Parallel Sysplex does

Works well … proven technology. One physical copy shared between data resource engines with Sysplex management of the data locks

Mirror the data -- various data mirroring technologies do this

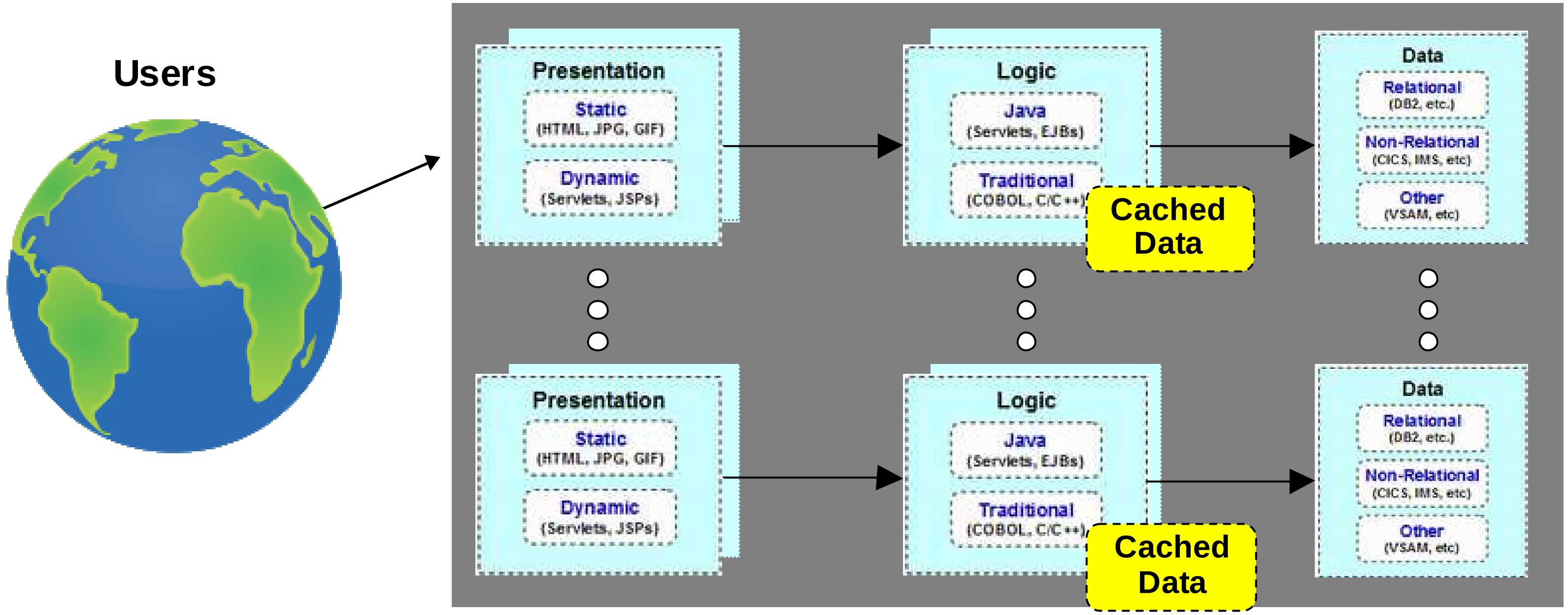
Multiple copies of the data … updates to one reflect over to the other

Partition the data -- split it across databases

Split the data in some fashion and have application go to appropriate data partition

Caching Less-Volatile Data

When scaling limitations are felt, one of the first things pursued is the caching of data that doesn't change much. General principle: avoid data call if possible.



Less I/O activity against the data subsystems

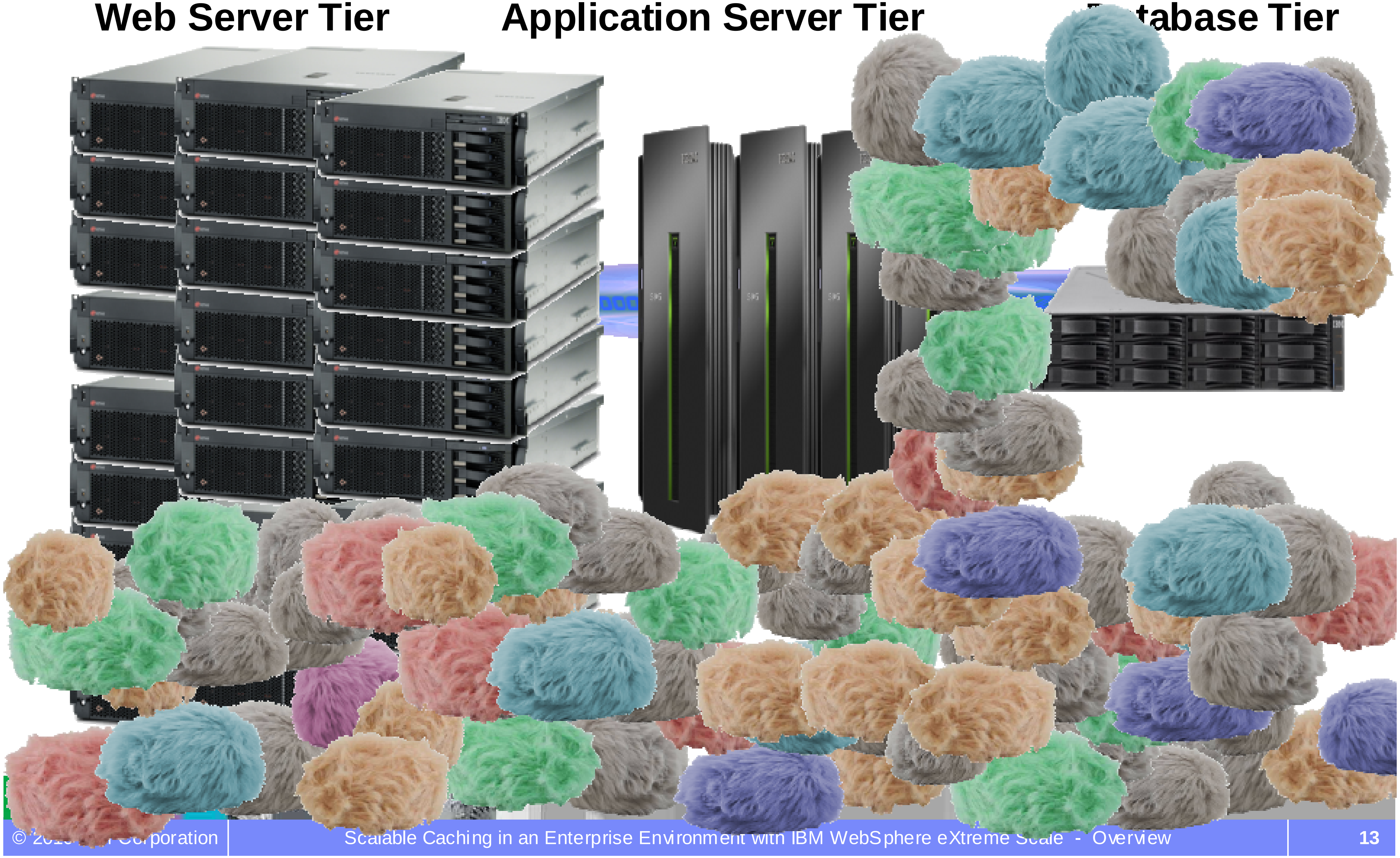
Example: basic user profile data for online banking customers or profile information for social networking site.

Doesn't change much … but it does change *sometimes*

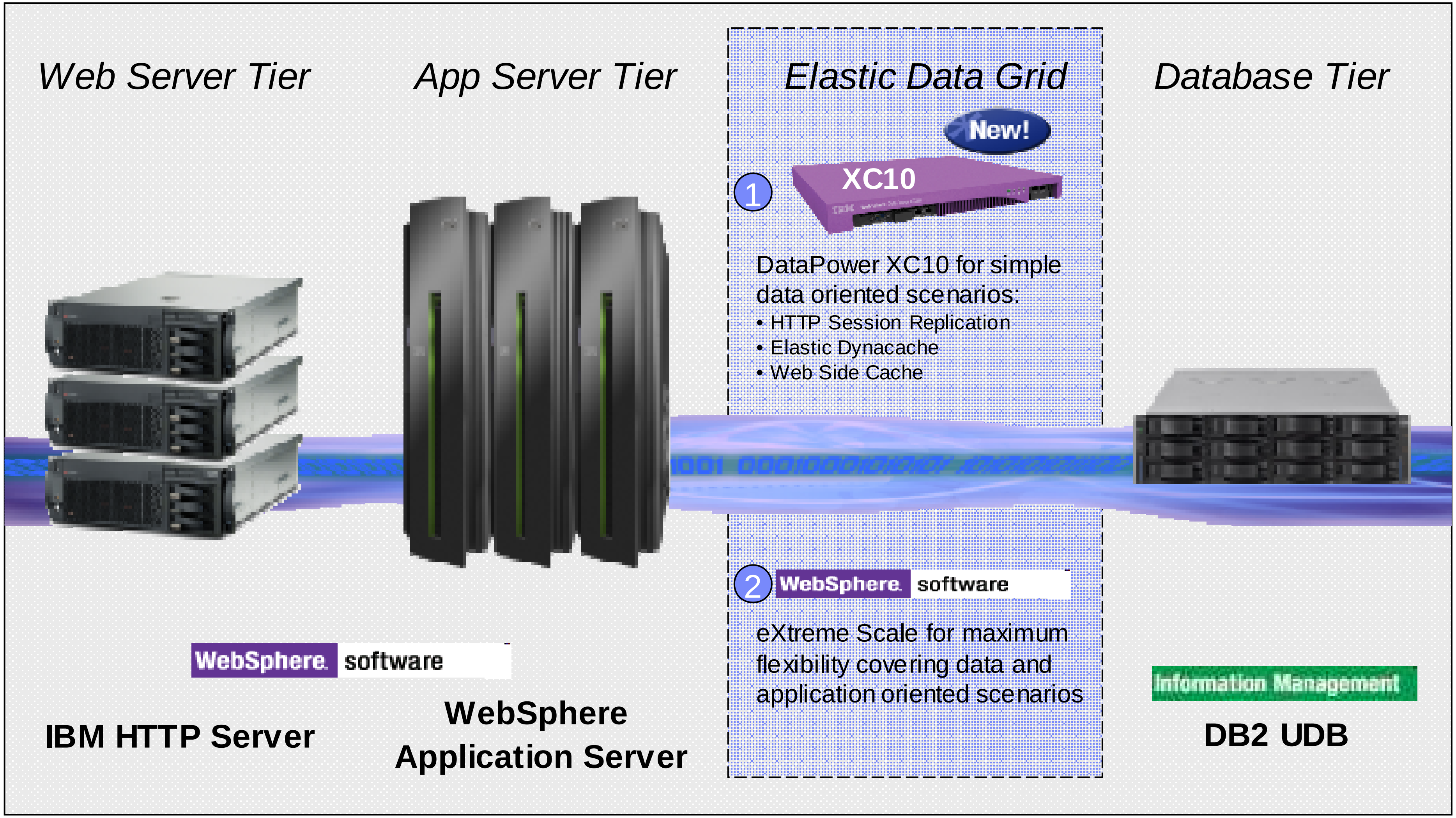
So the invalidation/update model has to be taken into account

Application Topologies Today

Web Server Tier **Application Server Tier** **Database Tier**



Modern Application Infrastructure Topology



Web Server Tier

App Server Tier

Elastic Data Grid

Database Tier

1

DataPower XC10 for simple data oriented scenarios:

- HTTP Session Replication
- Elastic Dynacache
- Web Side Cache

2 WebSphere software

eXtreme Scale for maximum flexibility covering data and application oriented scenarios

WebSphere software

IBM HTTP Server

WebSphere Application Server

Information Management

DB2 UDB

Innovative Elastic Caching Solutions

DataPower XC10 Appliance

- Drop-in cache solution optimized and hardened for data oriented scenarios
- High density, low footprint improves datacenter efficiency

"Data Oriented"

- Session management
- Elastic DynaCache
- Web side cache
- Petabyte analytics
- Data buffer
- Event Processing
- Worldwide cache
- In-memory OLTP
- In-memory SOA

"Application Oriented"

eXtreme Scale

- Ultimate flexibility across a broad range of caching scenarios
- In-memory capabilities for application oriented scenarios

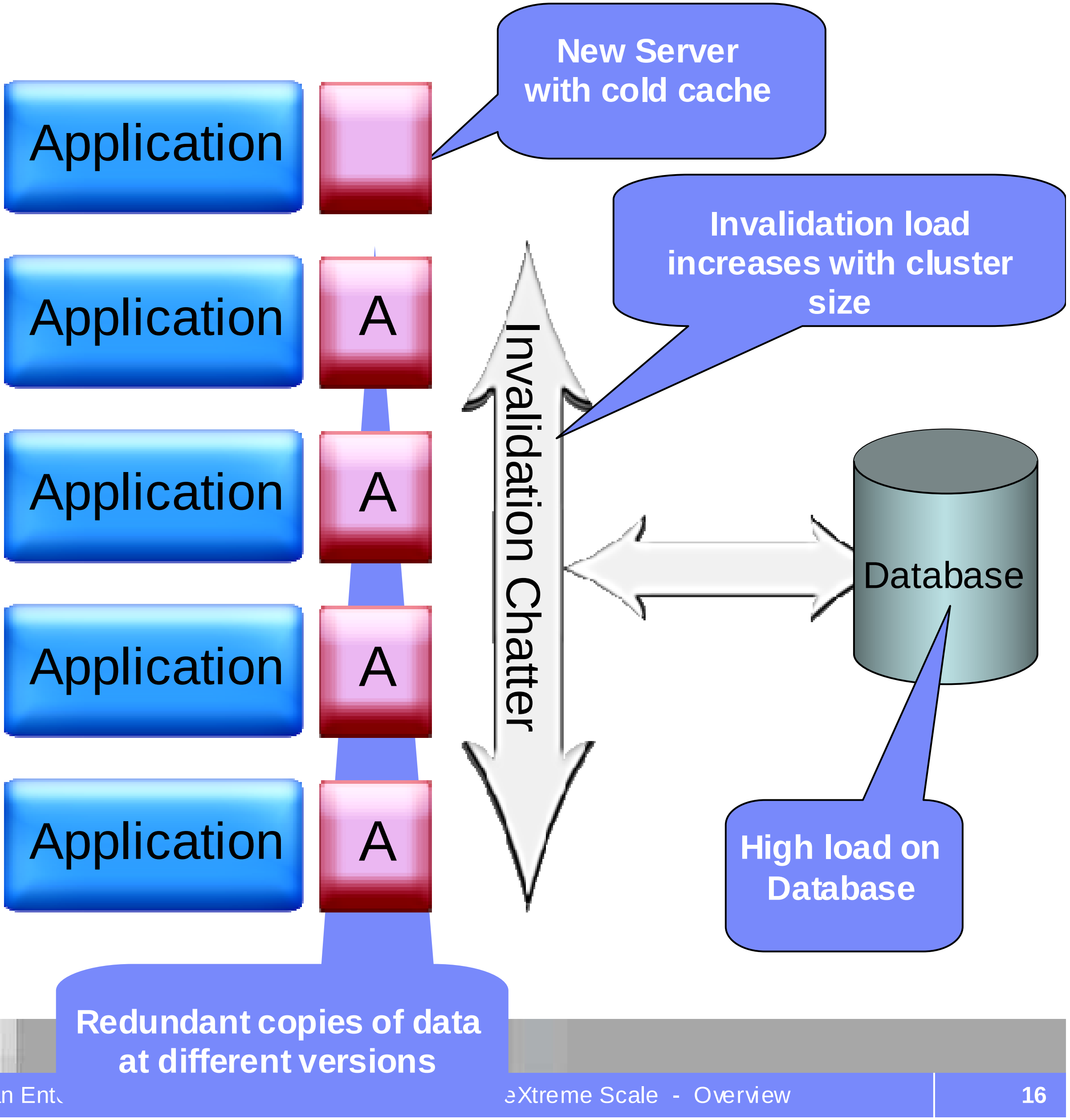
Elastic caching for linear scalability
High availability data replication
Simplified management, monitoring and administration

Traditional Cache Operation

- Cache capacity determined by individual JVM Size.
  - Size of each cache = M
  - # JVMs = N
  - **Total cache = M**

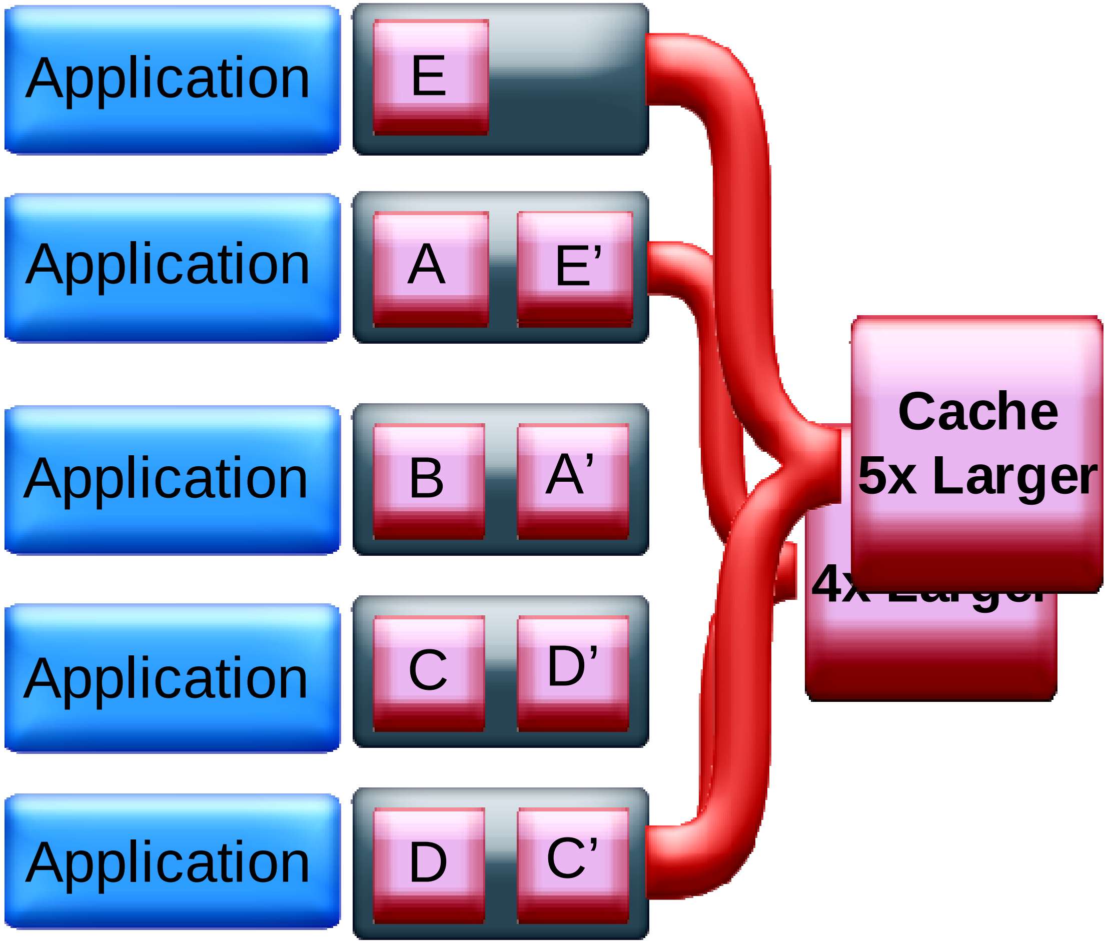
- Invalidation load per server increases as cluster grows.

- Cold start servers hit the database.

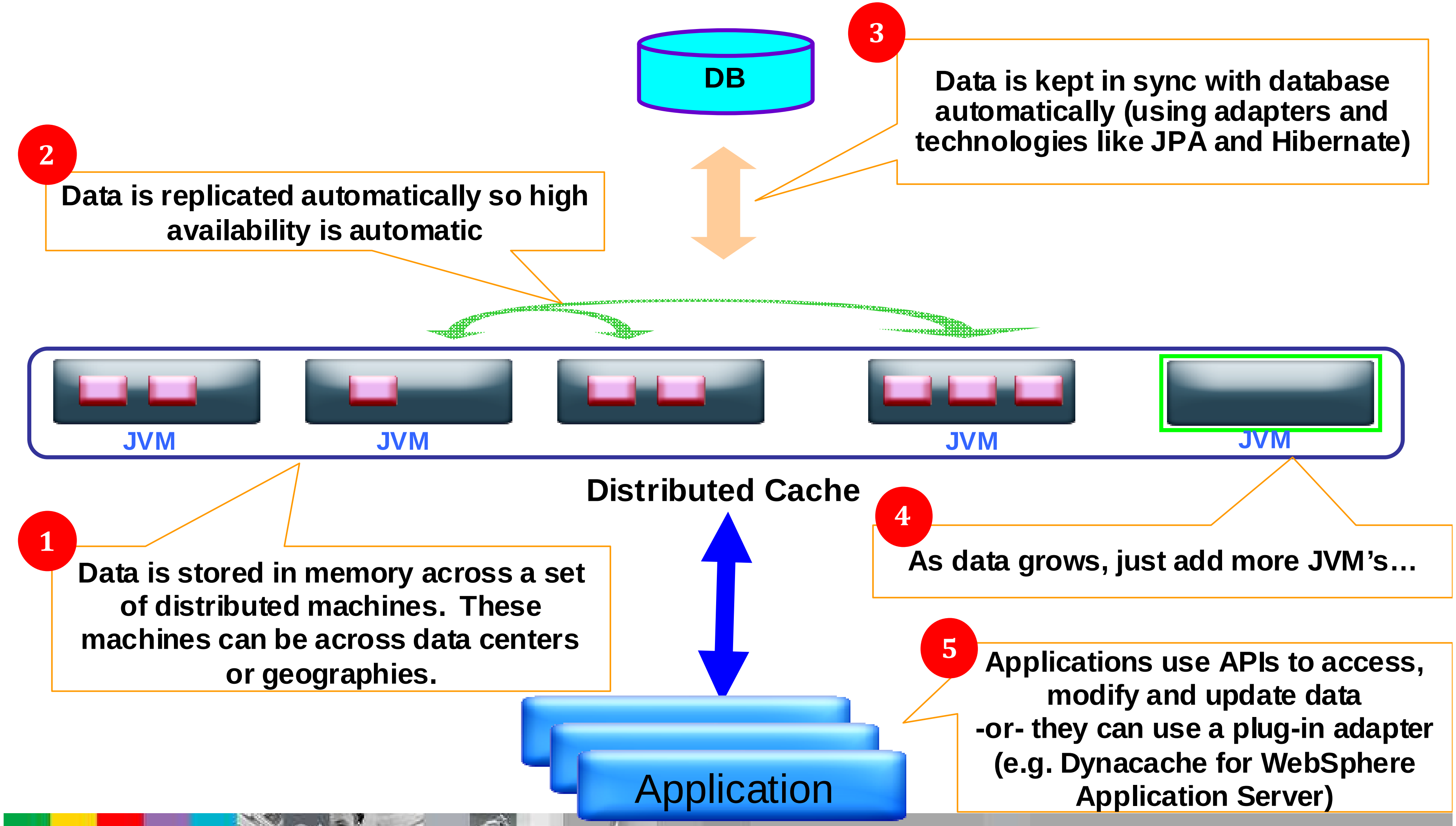

New Server with cold cache
Invalidation load increases with cluster size
Application
Application
A
Application
A
Application
A
Application
A
Invalidation Chatter
Database
High load on Database
Redundant copies of data at different versions

WebSphere eXtreme Scale Cache Operation

- Cache capacity determined by total cluster size
  - Size of each cache = M
  - # JVMs = N
  - Total Cache = **M x N**
- No invalidation chatter
- Linearly scalable
- Less load on database and no cold start spikes



WebSphere eXtreme Scale and Distributed Caching: A Primer



1 Data is stored in memory across a set of distributed machines. These machines can be across data centers or geographies.

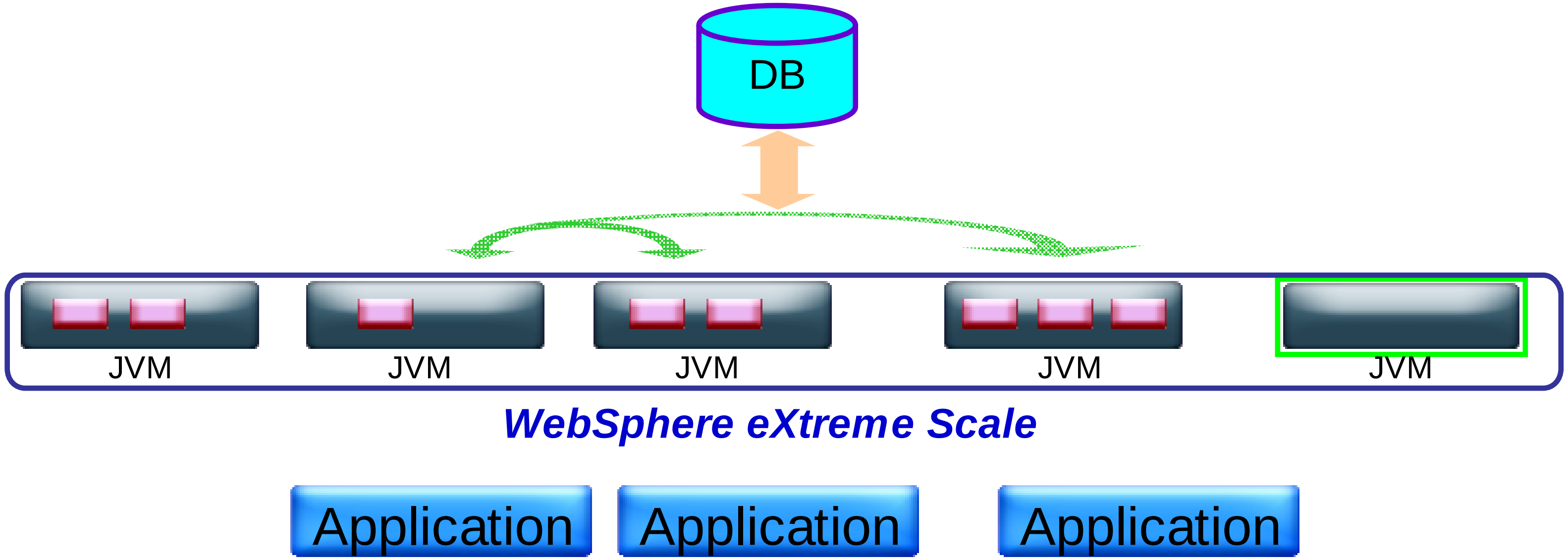
2 Data is replicated automatically so high availability is automatic

3 Data is kept in sync with database automatically (using adapters and technologies like JPA and Hibernate)

4 As data grows, just add more JVM's…

5 Applications use APIs to access, modify and update data -or- they can use a plug-in adapter (e.g. Dynacache for WebSphere Application Server)

Tell Me What It Really Means … To Applications!



- Consistent application and transaction response times … even as data grows !
  - This is referred to as “linearly scalable”
  - No need to continually re-architect as data and transaction volumes continue to grow and grow
- Much faster access to data than a traditional database (just a hop over the network)
- Most of the management of data is automatically handled by WebSphere eXtreme Scale

Caching Scenarios

WebSphere eXtreme Scale overview

- WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers
- Performs massive volumes of transaction processing with high efficiency and linear scalability
- Provides qualities of service such as
  - Transactional integrity
  - High availability
  - Predictable response times

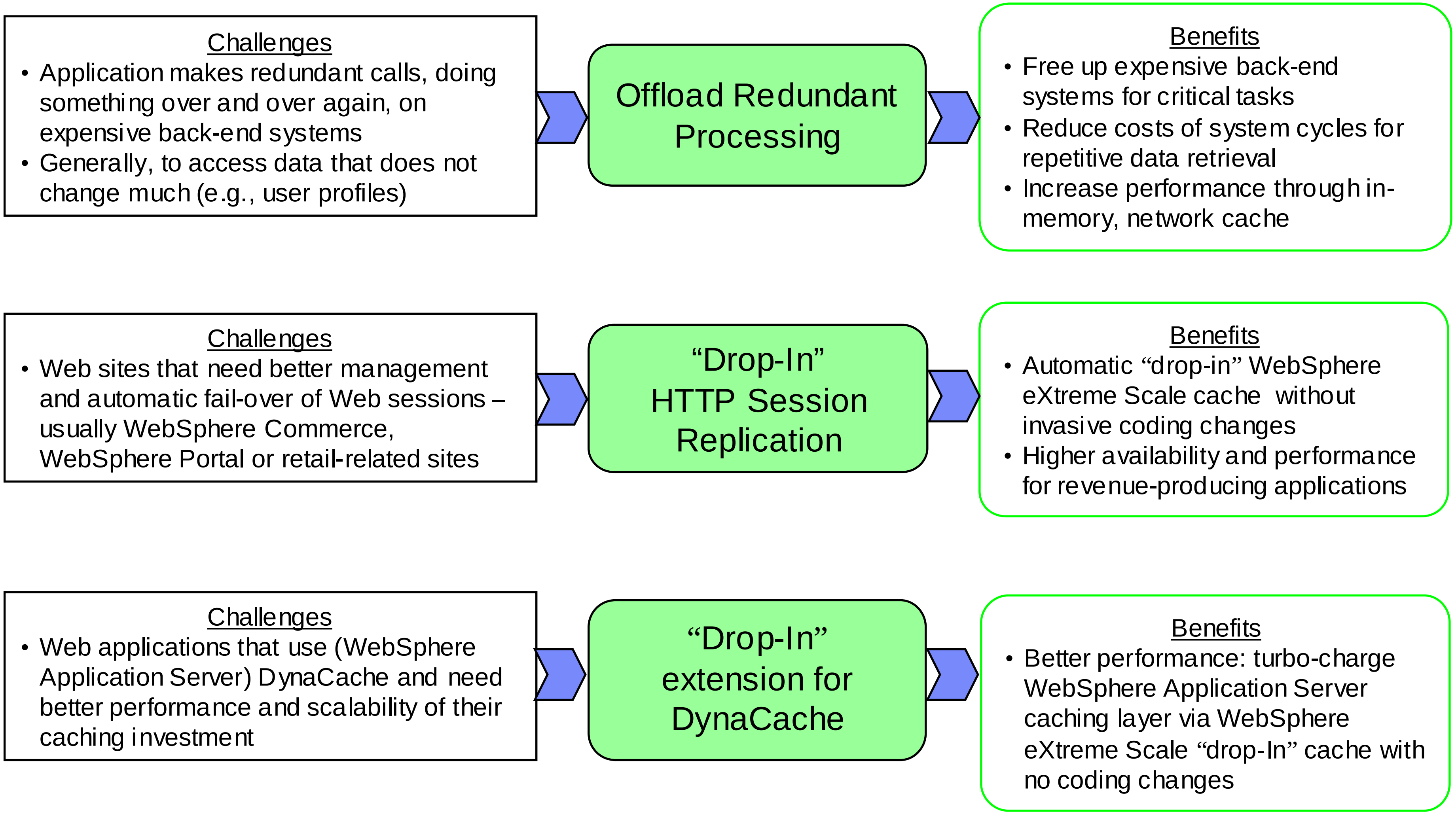
WebSphere eXtreme Scale overview

- Can be used as a powerful distributed cache to speed application access to data
- Can be backed by hardened storage
  - Cache loader interface enables automatic storage and retrieval of data according to user preferences to the technology of your choice (like a database)
- Can automatically replicate data either synchronously or asynchronously to ensure data availability and lower management burden
- Securable using Java Authentication and Authorization Service (JAAS) API

Scalability

- eXtreme Scale can grow to hold hundreds of JVMs of data for very large data sets
  - Scales nearly linearly with additional hardware
  - Supports thousands of concurrent clients
  - Supports 32-bit or 64-bit JVMs
  - eXtreme Scale clusters can be replicated and partitioned for fault-tolerance and high performance
- Cluster members communicate with each other using an ultra-high speed pub sub system
  - High performance and low processor overhead

Simple Caching Scenarios



Challenges

- Application makes redundant calls, doing something over and over again, on expensive back-end systems
- Generally, to access data that does not change much (e.g., user profiles)

Offload Redundant Processing

Benefits

- Free up expensive back-end systems for critical tasks
- Reduce costs of system cycles for repetitive data retrieval
- Increase performance through in-memory, network cache

Challenges

- Web sites that need better management and automatic fail-over of Web sessions – usually WebSphere Commerce, WebSphere Portal or retail-related sites

“Drop-In” HTTP Session Replication

Benefits

- Automatic “drop-in” WebSphere eXtreme Scale cache without invasive coding changes
- Higher availability and performance for revenue-producing applications

Challenges

- Web applications that use (WebSphere Application Server) DynaCache and need better performance and scalability of their caching investment

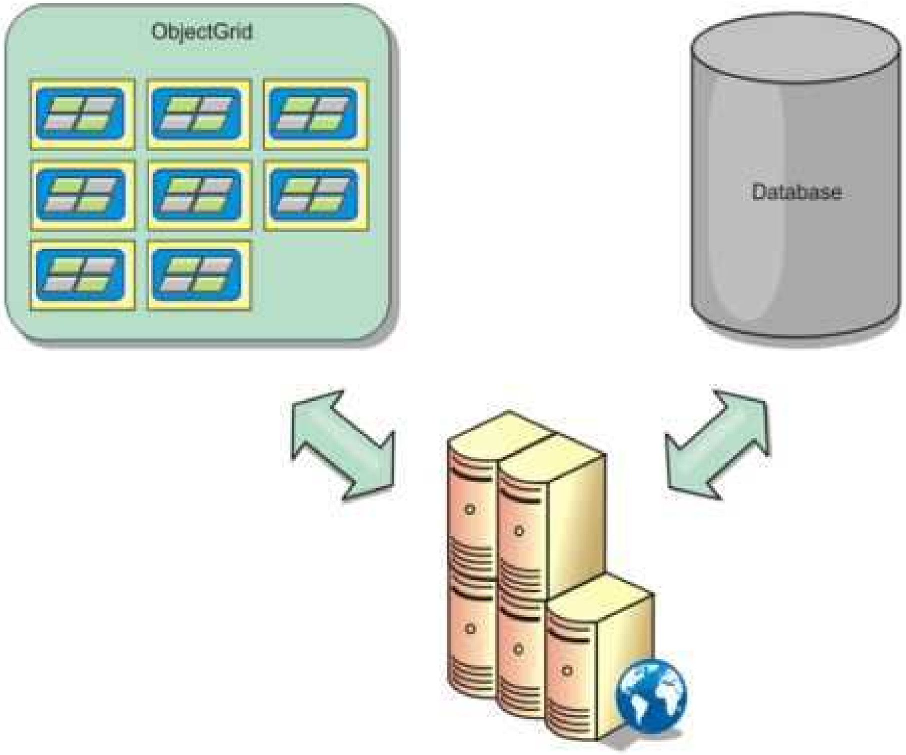
“Drop-In” extension for DynaCache

Benefits

- Better performance: turbo-charge WebSphere Application Server caching layer via WebSphere eXtreme Scale “drop-In” cache with no coding changes

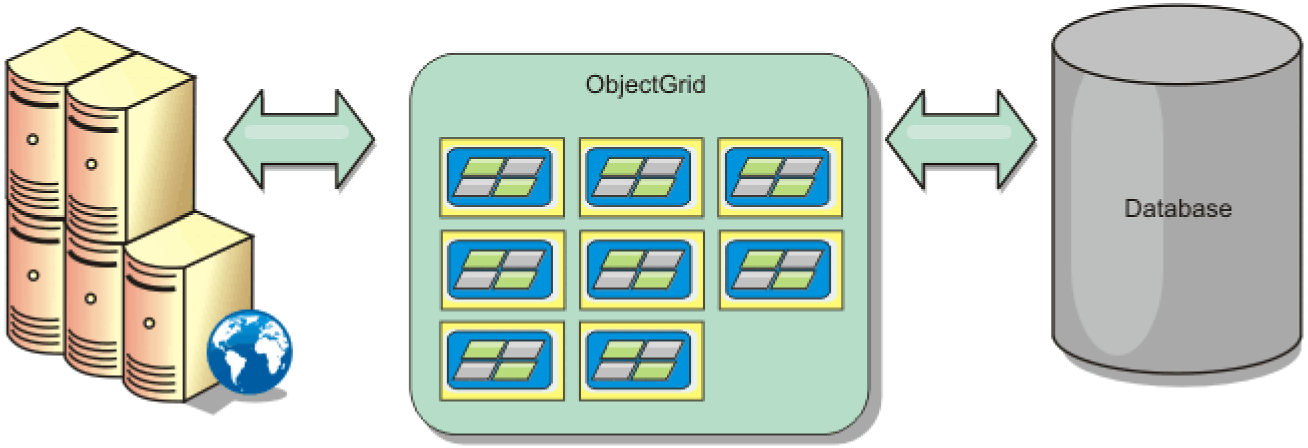
Offload Redundant Processing : Side Cache

- WebSphere eXtreme Scale is used to temporarily store objects that would normally be retrieved from a back-end database.
- Applications check to see if WebSphere eXtreme Scale contains the desired data.
- If the data is there, the data is returned to the caller. If the data is not there, the data is retrieved from the back-end and inserted into WebSphere eXtreme Scale so that the next request can use the cached copy.



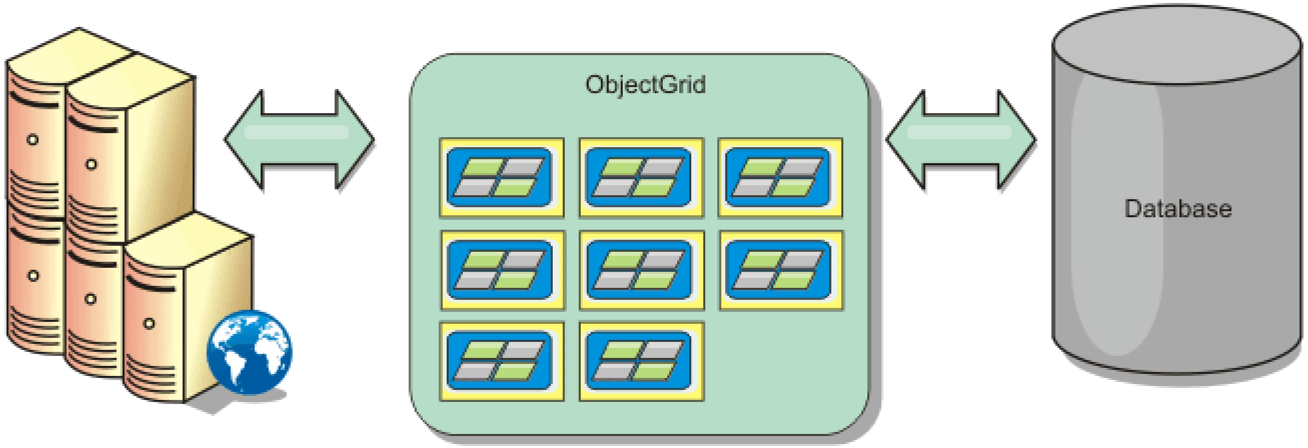
Offload Redundant Processing : In-line cache

- Applications check to see if WebSphere eXtreme Scale contains the desired data.
- If the data is there, the data is returned to the caller. If the data is not there, the data is retrieved from the back-end by WebSphere eXtreme Scale so that the next request can use the cached copy.
- Changes are written to the cache and back-end synchronously. A *write-through* cache.



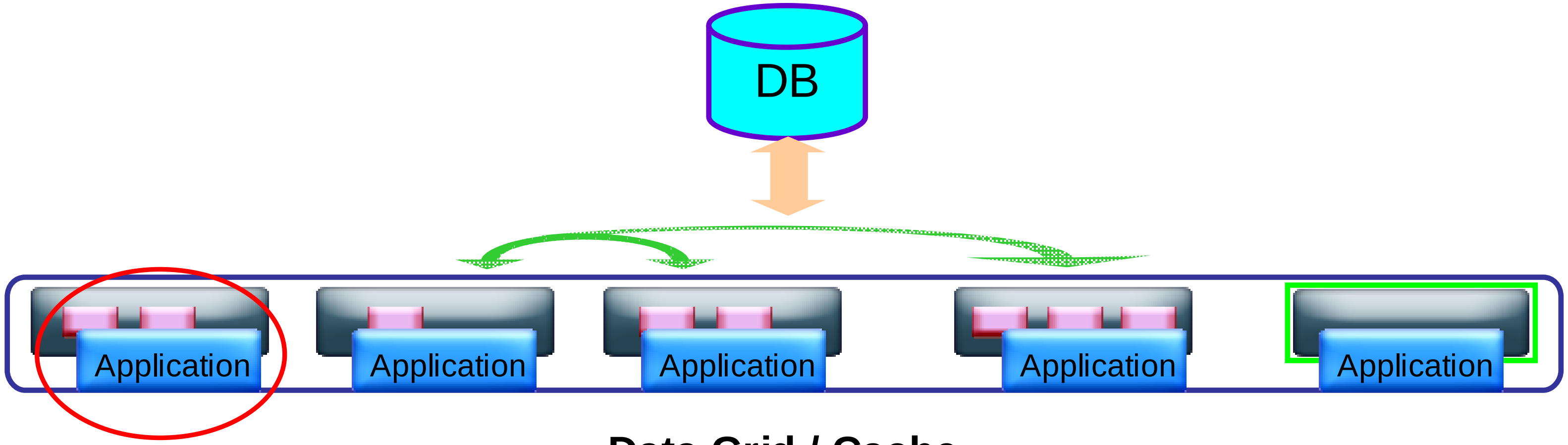
Offload Redundant Processing : In-line cache with Write-Behind

- Variation of previous scenario. Changes are written to the back-end asynchronously. A *write-behind* cache.
- Back-end load is significantly reduced as there are fewer but larger transactions
- Back-end availability has no impact on application availability.



Offload Redundant Processing : Co-located application

- Business logic runs in parallel with where the data is stored.
- This is called *co-location*. The application and data are together.
- Results in very good application and transaction performance because the application lives with the data, which is in-memory.



Data Grid / Cache

“Drop-In” HTTP Session Replication

- HTTP sessions can be replicated across servers using eXtreme Scale
  - A servlet filter that enables session replication can be inserted into any Web application
  - Provides a session persistence approach that is independent of the WebSphere cell infrastructure
- WebSphere products can use WebSphere eXtreme Scale as an upgraded session persistence mechanism
- Non-WebSphere servers (such as Geronimo or JBoss) can also use this servlet filter

“Drop-In” Dynamic cache service support

- Allows applications using the WebSphere dynamic cache service to leverage the advanced features and performance improvements of eXtreme Scale
- Supports WebSphere Application Server V6.1 or V7.0
- Dynamic cache evictors, dependency-based invalidation functions, and event listeners can be used on the WebSphere eXtreme Scale cache
- Dynamic cache can keep statistics for each grid instance

Section

WebSphere eXtreme Scale Spotlight

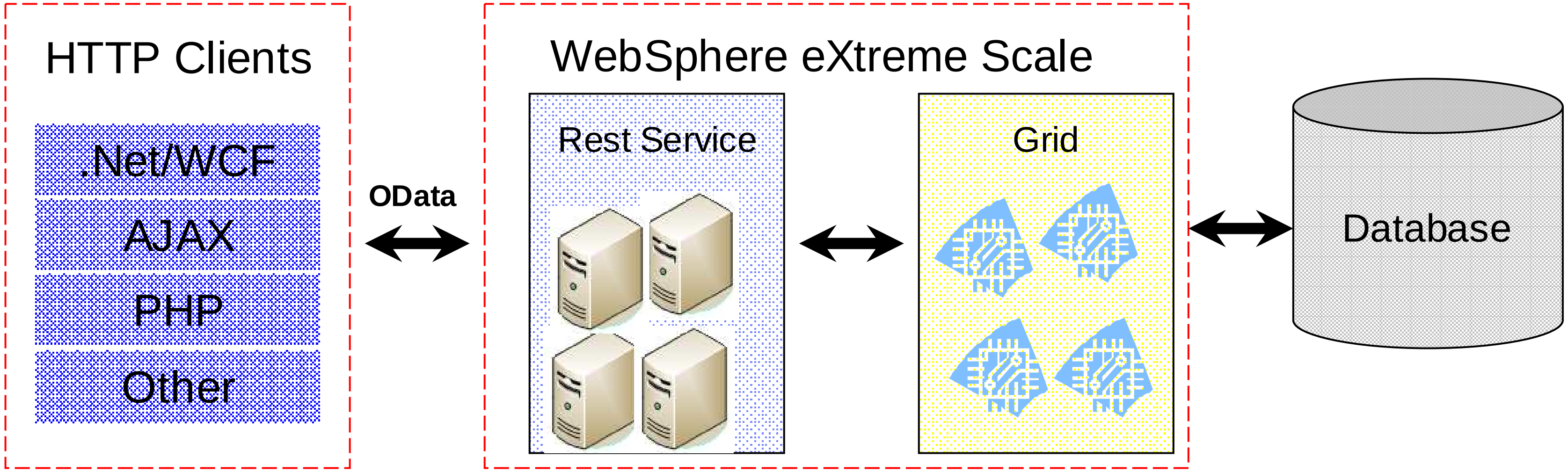
Non-invasive middleware

WebSphere. eXtreme Scale

- Lightweight
  - Single JAR, 15MB
- Integrates with WebSphere Application Server and stack products
  - WebSphere Business Events
  - WebSphere Commerce
  - IBM ILOG® JRules
  - … And more!
- Also supports Non-WebSphere servers
  - Geronimo, JBoss, and others
- Enhances existing application infrastructure with little to no application code change

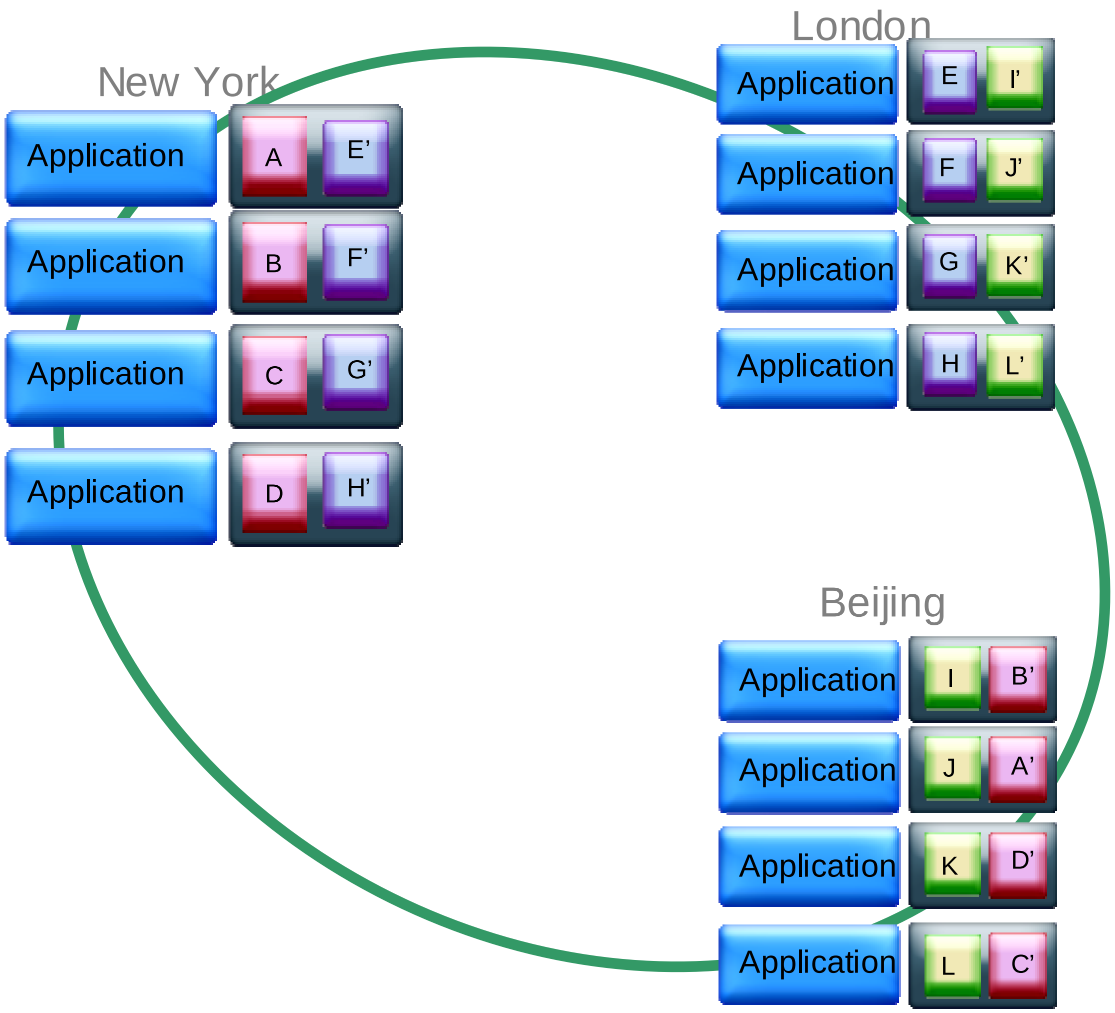
It’s Not Just for Java

- Rest Data Service
  - Allows HTTP clients, including .NET and AJAX clients, to use and exploit WebSphere eXtreme Scale.
  - Rest Service is hosted on application server such as WebSphere Application Server or Tomcat.



Replication and Availability

- Granular control over where information is placed and replicated
- Route client transactions to information based on your specifications
- Flexible definition of *zone*
  - Across data centers
  - Specific chassis
  - Rooms in a building



Working with WebSphere eXtreme Scale data

- An ObjectGrid contains one or more ObjectMaps
  - Interface similar to java.util.Map
  - Supports all of the expected Map methods
    - put(), get()
    - Additional methods are also provided, such as insert(), update(), and invalidate()
- Objects are stored as key/value pairs
  - Can be entered into the Map by the application
  - Can be loaded from an external source using custom loader objects
- Can be indexed on key or attribute values

Access map and data

- Access ObjectMap through Session
  - ObjectMap is local container for BackingMap

```
ObjectMap myEmpMap = mySession.getMap("employees");
```

- Map is defined in ObjectGrid configuration xml file
  - BackingMap entry
  - Defines map configuration
- Now access data
  - Data copied between BackingMap and ObjectMap

```
myEmpMap.get("000300"); // get employee with key “000300”
```

Entity manager

- Alternative mechanism for interacting with distributed cache
- Uses JPA-like APIs (Entities)
- Persist POJO’s, along with any relationships it has, in a transactional manner
- Retrieve/find POJO’s using the EntityManager APIs
- Keep track of changes in any attributes
- Loose coupling of data and metadata to allow heterogeneous server and client application versions

EntityManager annotation example

- Employee entity, using Java SE Development Kit (JDK™) 5 annotations
  - Declare an Entity called Emp with a key and four attributes:

```
import com.ibm.websphere.projector.annotations.*;

// Identify an Employee entity, named Emp
//  o The default name is the short class name. The name is used
//    to identify the entity using queries and to assign an
//    ObjectGrid BackingMap.
// The entity is persisted using field access. To persist using properties,
// annotate the get/is methods instead of the field.

@Entity(name="Emp“,accessType=AccessType.FIELD)
public class Employee {
    // Fields should not normally be public.
    @Id String SSN;                // Key
    @Basic String firstName;    // Persistent Attribute (@Basic is optional)
    @Basic String surname;      // Persistent Attribute
    @Basic Double salary;       // Persistent Attribute
    @ManyToOne Manager manager; // Persistent entity reference
    public Employee(){ … };     // Default no-arg, public constructor
                                //  is required.
}
```

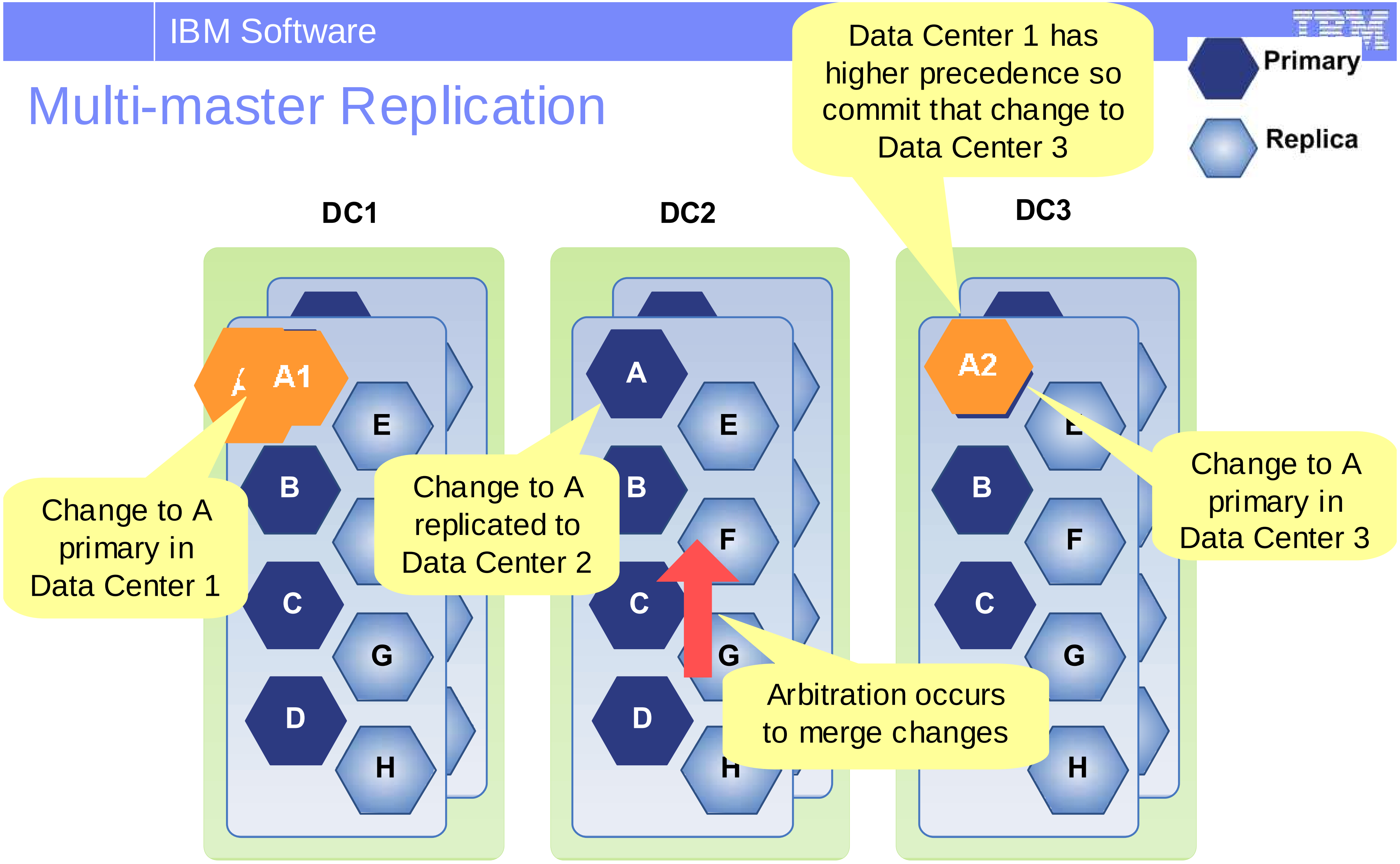
Locking strategies

- Optimistic locking
  - Locks are only acquired during the actual update action
    - Exception if two threads try to update the same data simultaneously
  - Most useful for “read mostly” Maps
- Pessimistic locking
  - Data is locked when a transaction “gets” data
  - Best used when optimistic locking results in frequent collisions
- None
  - ObjectGrid does not manage concurrency
  - Relies on EJB persistence manager or concurrency provided by a Loader

Evicting objects from the map

- Cache size is controlled by evicting objects when space is needed
- An Evictor is an extensible object type for creating custom eviction schemes
- Some Evictors are provided
  - TTL (time to live) eviction is built into ObjectGrid
    - Can be based on creation time or last used time
  - Plug-in Evictors:
    - LRU (least recently used)
    - LFU (least frequently used)

Multi-master Replication



Data Center 1 has higher precedence so commit that change to Data Center 3

Change to A primary in Data Center 1

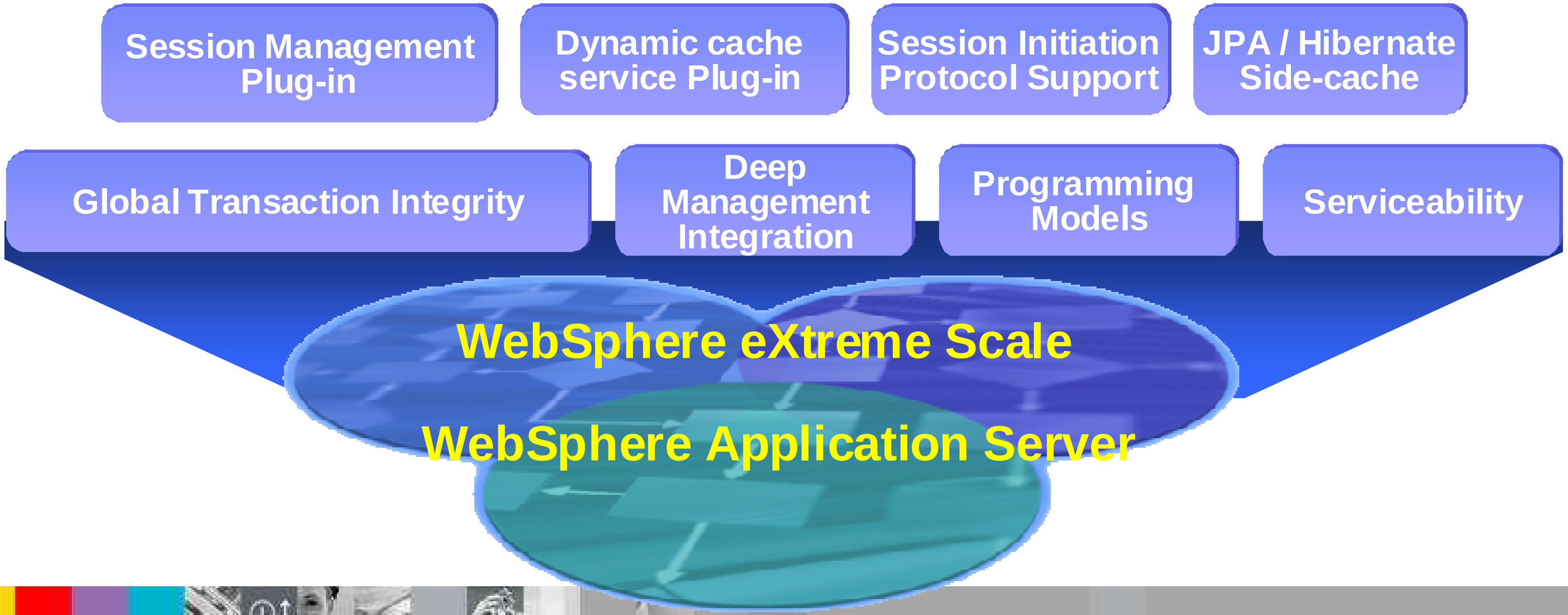
Change to A replicated to Data Center 2

Arbitration occurs to merge changes

Change to A primary in Data Center 3

Native Integration with WebSphere Application Server

- WebSphere eXtreme Scale works with all WebSphere Application Server 6.x and 7.x versions
- WebSphere eXtreme Scale extends the value of WebSphere Application Server deployments by providing:
  - Session management plug-in for multi data center support
  - Dynamic cache plug-in to turbo-charge existing caching environments
  - JPA / Hibernate side-cache to accelerate existing database queries



Session Management Plug-in

Dynamic cache service Plug-in

Session Initiation Protocol Support

JPA / Hibernate Side-cache

Global Transaction Integrity

Deep Management Integration

Programming Models

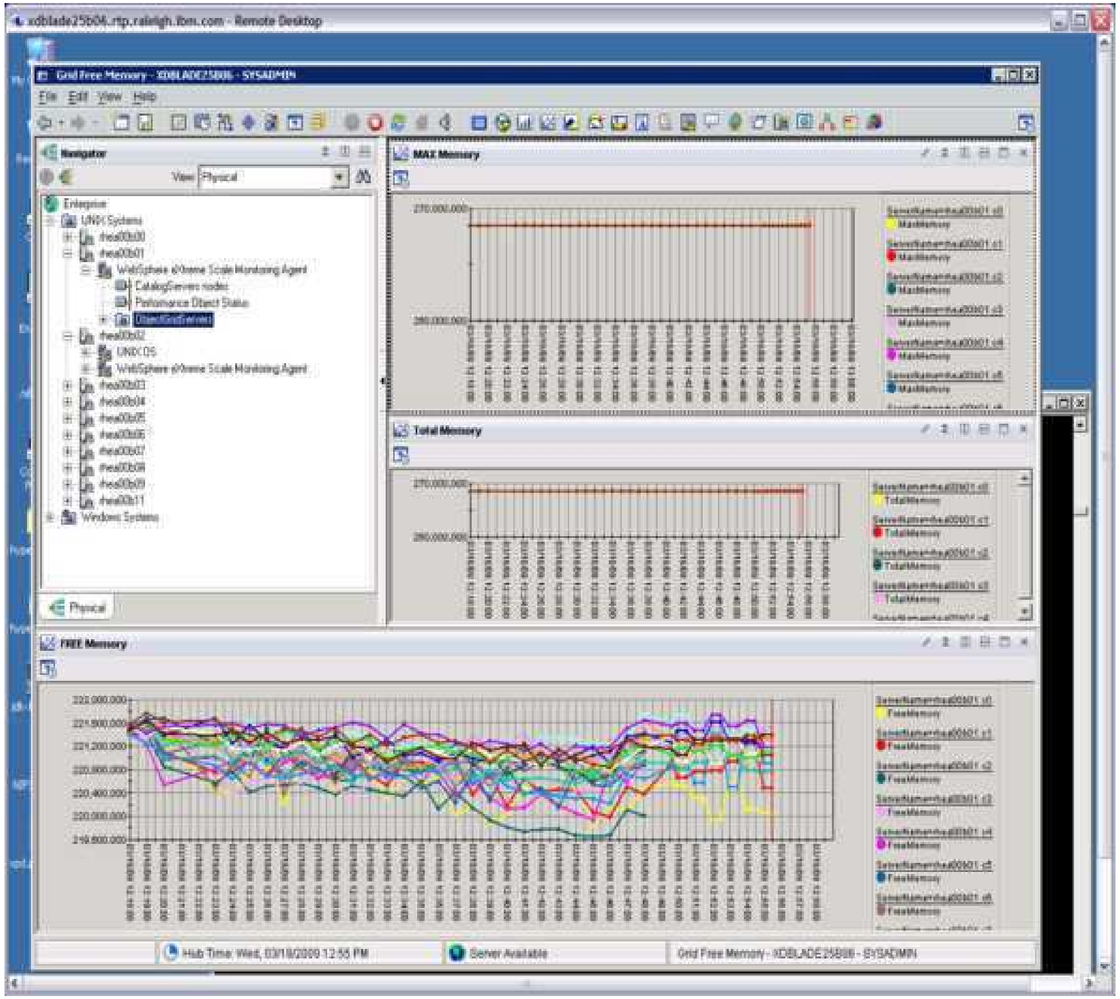
Serviceability

WebSphere eXtreme Scale

WebSphere Application Server

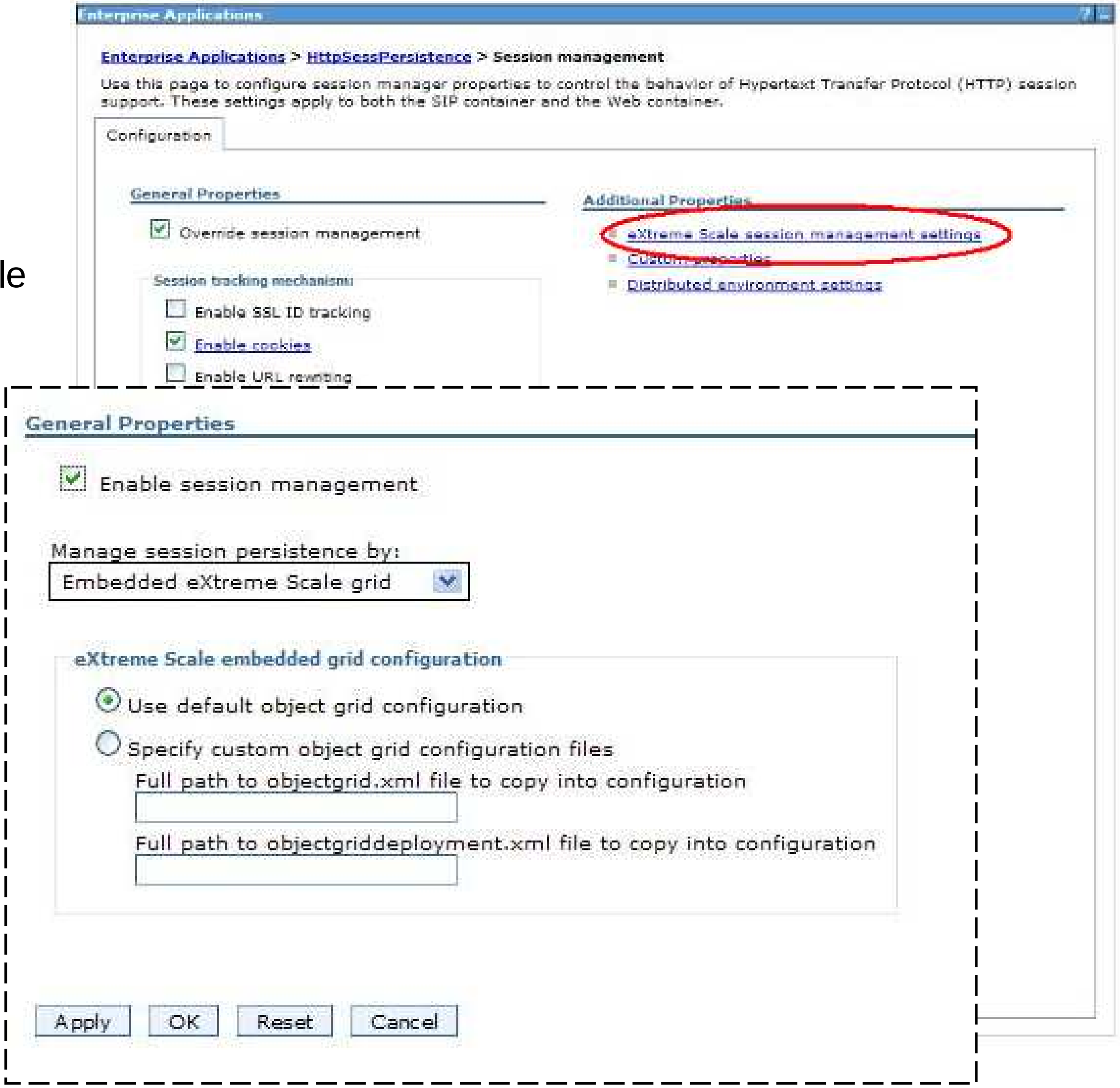
Monitoring Tools

- Support for leading external monitoring products
  - IBM Tivoli® Monitoring
  - CA Wily Introscope
  - Hyperic HQ
- Built-in monitoring with WebSphere Integrated Solutions Console and xsAdmin command-line utility


xdblade25b04.rtp.raleigh.ibm.com - Remote Desktop
Grid Free Memory - XDBLADE25B06 - SYSADMIN
File
Edit
View
Help
Navigator
View: Physical
Enterprise
UNIX Systems
WebSphere eXtreme Scale Monitoring Agent
CatalogServers nodes
Performance Object Status
Windows Systems
Physical
MAX Memory
270,000,000
Total Memory
270,000,000
FREE Memory
MaxMemory
TotalMemory
FreeMemory
Hub Time: Wed, 03/18/2009 12:55 PM
Server Available
Grid Free Memory - XDBLADE25B06 - SYSADMIN

WebSphere Application Server Console extensions

- Administrators and configurators can create/register catalog services
- Administrators and deployers can enable eXtreme Scale session persistence



Enterprise Applications

Enterprise Applications > HttpSessPersistence > Session management

Use this page to configure session manager properties to control the behavior of Hypertext Transfer Protocol (HTTP) session support. These settings apply to both the SIP container and the Web container.

Configuration

General Properties

☑ Override session management

Session tracking mechanism:

☐ Enable SSL ID tracking

☑ Enable cookies

☐ Enable URL rewriting

Additional Properties

- eXtreme Scale session management settings
- Custom properties
- Distributed environment settings

General Properties

☑ Enable session management

Manage session persistence by:

Embedded eXtreme Scale grid

eXtreme Scale embedded grid configuration

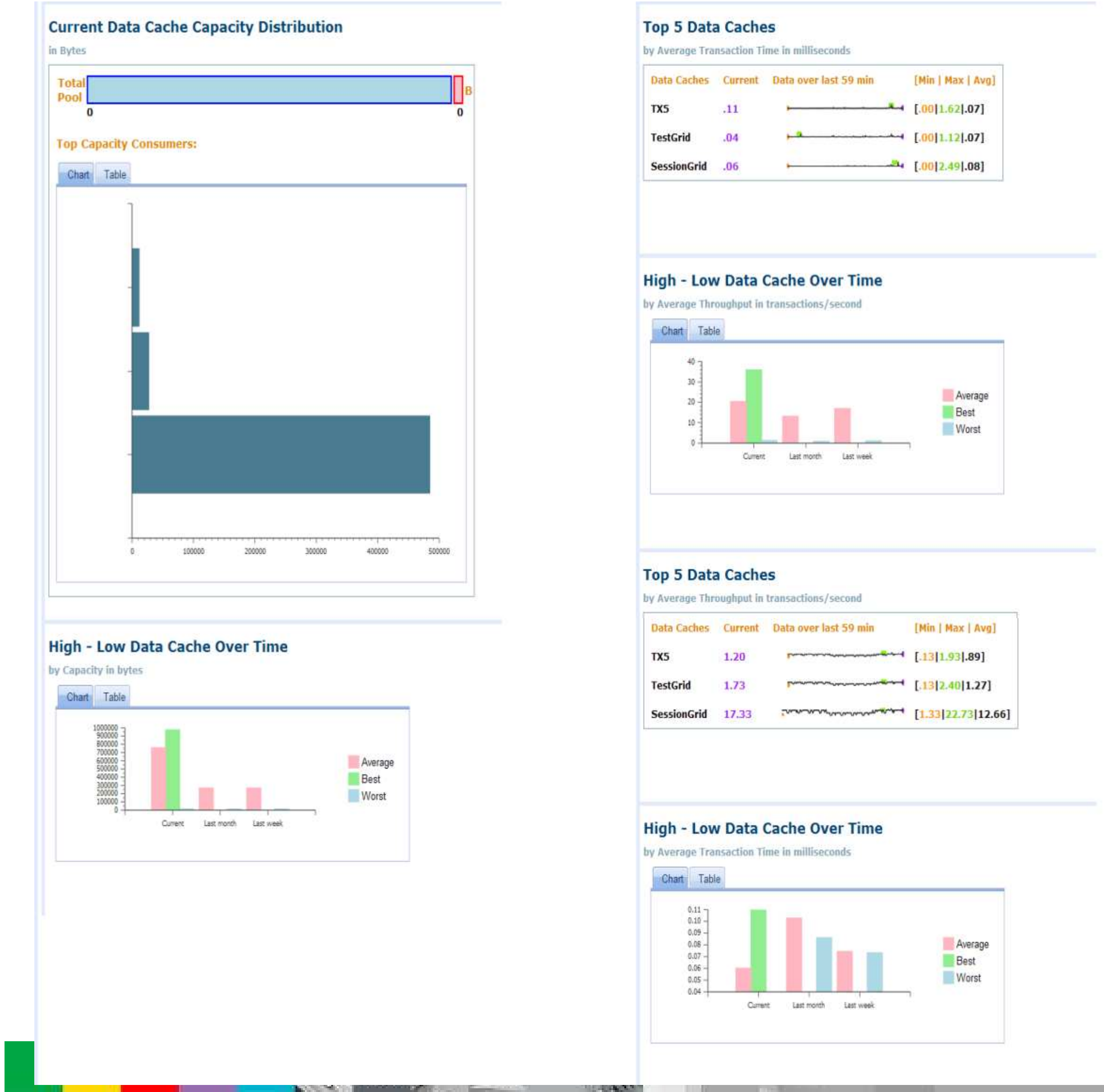
● Use default object grid configuration

○ Specify custom object grid configuration files

Full path to objectgrid.xml file to copy into configuration

Full path to objectgriddeployment.xml file to copy into configuration

New monitoring console - All Data Caches overview



Current Data Cache Capacity Distribution

in Bytes

Top Capacity Consumers:

High - Low Data Cache Over Time

by Capacity in bytes

Top 5 Data Caches

by Average Transaction Time in milliseconds

Data Caches	Current	Data over last 59 min	[Min | Max | Avg]
TX5	.11		[.00|1.62|.07]
TestGrid	.04		[.00|1.12|.07]
SessionGrid	.06		[.00|2.49|.08]

High - Low Data Cache Over Time

by Average Throughput in transactions/second

Top 5 Data Caches

by Average Throughput in transactions/second

Data Caches	Current	Data over last 59 min	[Min | Max | Avg]
TX5	1.20		[.13|1.93|.89]
TestGrid	1.73		[.13|2.40|1.27]
SessionGrid	17.33		[1.33|22.73|12.66]

High - Low Data Cache Over Time

by Average Transaction Time in milliseconds

- The most broad of the overviews
- Various charts on capacity, throughput, and transaction time

Note: Final view may change

Section

Client Usage

Client Usage: Worldwide Fantasy Sports Web Site

Entertainment

5 Billion
requests per day

10x
reduced
response
time

Fantasy Sports Web Infrastructure

- **Before:** 60ms response time against database
- **After:** WebSphere eXtreme Scale improved to 6ms response time
- 450k concurrent users
- 80k requests per second up to 1M in 2011
- 6 weeks from concept to production

Support transaction-intensive services

Deliver consistent & predictable response times

Take action on growing volumes of business events

Scale with simplicity and lower cost

Client Usage: On-Line Banking

Retail Banking & Investments

22 Million
online banking users

35x
reduced response times

US$500k
reduced costs per month

20x
reduction in “FCIs”

Next-generation Online Banking

- **Before:** 700ms to login with 2 backend calls
- **After:** 20ms to login with profile cache access
- US$6M cost savings in Millions of Instructions Per Second (MIPS) reduction
- 700k transactions per hour across 3 data centers
- 8Gb of data transfer per hour between data centers

Provide seamless cache infrastructure across applications

Deliver high performance & consistent response times

Ensure high availability of critical online applications

Scale with simplicity and lower total cost of ownership (TCO)

Section

Summary

Summary

- WebSphere eXtreme Scale provides a high performance, scalable cache system capable of performing massive volumes of transaction processing
- WebSphere eXtreme Scale integrates with existing environments to save money while improving response time and scalability
- WebSphere eXtreme Scale is a versatile product that has been deployed in a variety of enterprise situations

Reference materials

- IBM Redbooks® - User's Guide to WebSphere eXtreme Scale
  - http://www.redbooks.ibm.com/abstracts/sg247683.html
- WebSphere eXtreme Scale Trial Download
  - http://www.ibm.com/developerworks/downloads/ws/wsdg/learn.html
- WebSphere eXtreme Scale Videos
  - http://www.youtube.com/ibmextremescale

Questions