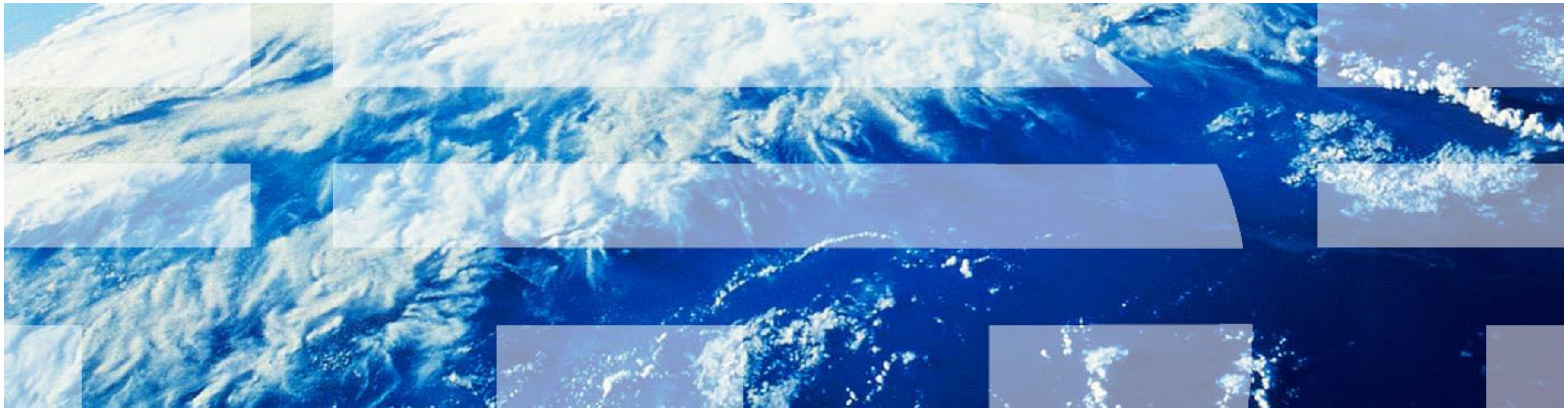
Caching patterns and extending mobile applications with elastic caching (With Demonstration)



The world is changing and each of these technology shifts has potential to make a significant impact.

Big Data

90

percent of the data created in the last two years alone.

Mobile

1

billion (plus) smart devices shipped in 2013 alone.

Social

81

percent of customers depend on social sites for purchasing advice.

Cloud

62

percent of total workloads will be in the cloud by 2016.

Internet of Things

75

billion devices connected to the internet by 2020.

Mobile is driving significant shifts

Mobile has become a primary mode of transaction

35% of the average top 50 retailers' monthly audience comes *exclusively* from mobile platforms

Compatibility is no longer enough; mobile needs to be first

70% or more of mobile leaders say they have been successful in ensuring interoperability with other systems, leveraging **APIs** for external or cloud-provided data services

Becoming mobile first means embedding mobile into the design of business and IT processes.

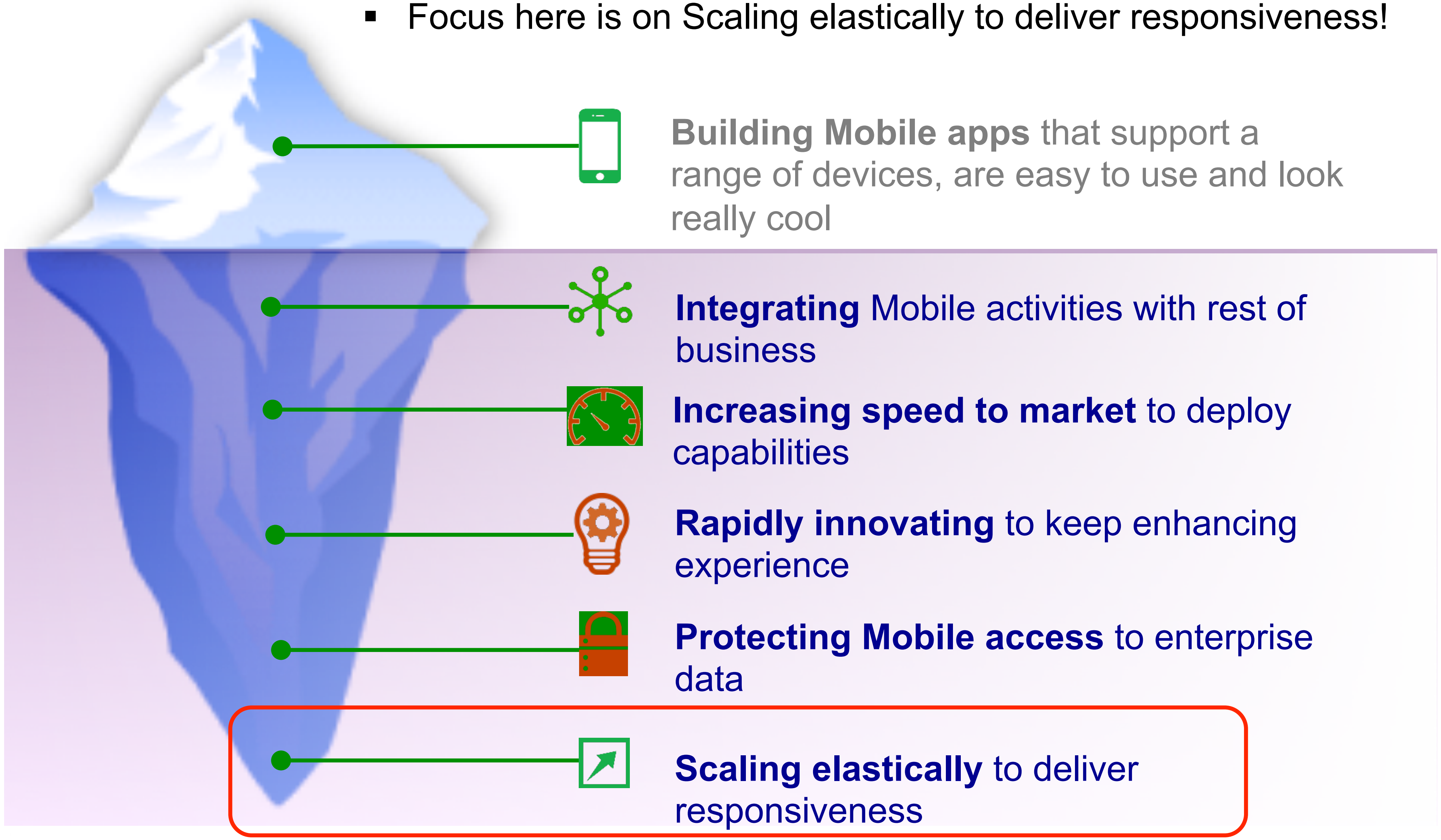
Mobile can capture and deliver insights anytime, anywhere

90% of new passenger cars sold will have some form of in-vehicle platform by 2020, up from 10 percent in 2012.

Billions of connected devices, embedded sensors in roadways, buildings and machinery, and myriad machine-to-machine communication systems are delivering a continuous feed of data.
By applying analytics, organizations can know their customers better and operate more efficiently

Creating a truly engaging Mobile experience involves far more than building great Mobile apps

- Focus here is on Scaling elastically to deliver responsiveness!



Building Mobile apps that support a range of devices, are easy to use and look really cool

Integrating Mobile activities with rest of business

Increasing speed to market to deploy capabilities

Rapidly innovating to keep enhancing experience

Protecting Mobile access to enterprise data

Scaling elastically to deliver responsiveness

What is Unique About Mobile Integration Scenarios?

- **The technologies tend to be a bit different**
  - JSON/REST instead of SOAP/XML,
- **Bandwidth and battery life are at a premium**
  - Need to minimize back and forth communications
- **Security is a paramount concern**
  - Mobility of devices makes them easier to steal or spoof
- **Access patterns and volumes are less predictable**
  - More service calls on mobile devices, potentially more users, anytime
- **Performance matters**
  - Performance expectations are much higher on mobile

How might I improve my mobile application performance?

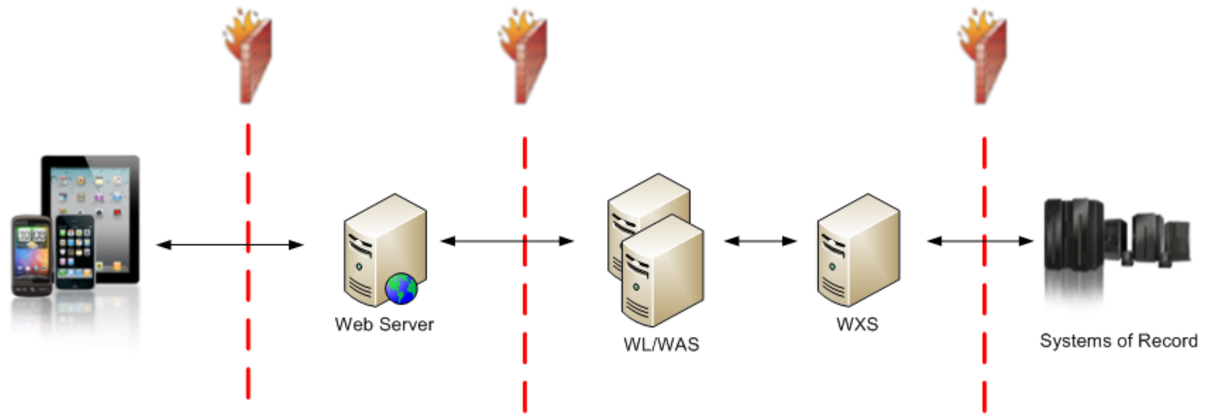
Question:

How can I make sure my mobile solution is responsive regardless of load?

Pattern:
- Worklight
- WebSphere Application Server (WAS/Liberty)
- WebSphere eXtreme Scale (WXS) / XC10

Answer:

Introduce a **caching layer** in the pattern for improved performance and throughput



Apply to:
- Customer has very high volume of mobile traffic
- Customer needs consistent and fast response time for their mobile apps
- Customer wants to reduce hardware spending on backend infrastructure

What is a cache anyway?

A cache allows you to get stuff faster and helps you avoid doing something over and over again (which may be redundant and may not make sense)

(far away)

(near)

(happy)

How to overwhelm an enterprise

Retail, Banking, Finance, Insurance, Telecom, Travel & Transportation …

Mobile Access

Laptops, Ultrabooks, Tablets, Smartphones + 3G 4G Free Wi-Fi = **Transaction Overload**

Social Media

TV, Movie, Sports Personality mentions / endorses product + YouTube twitter facebook = **Transaction Overload**

Targeted Advertising

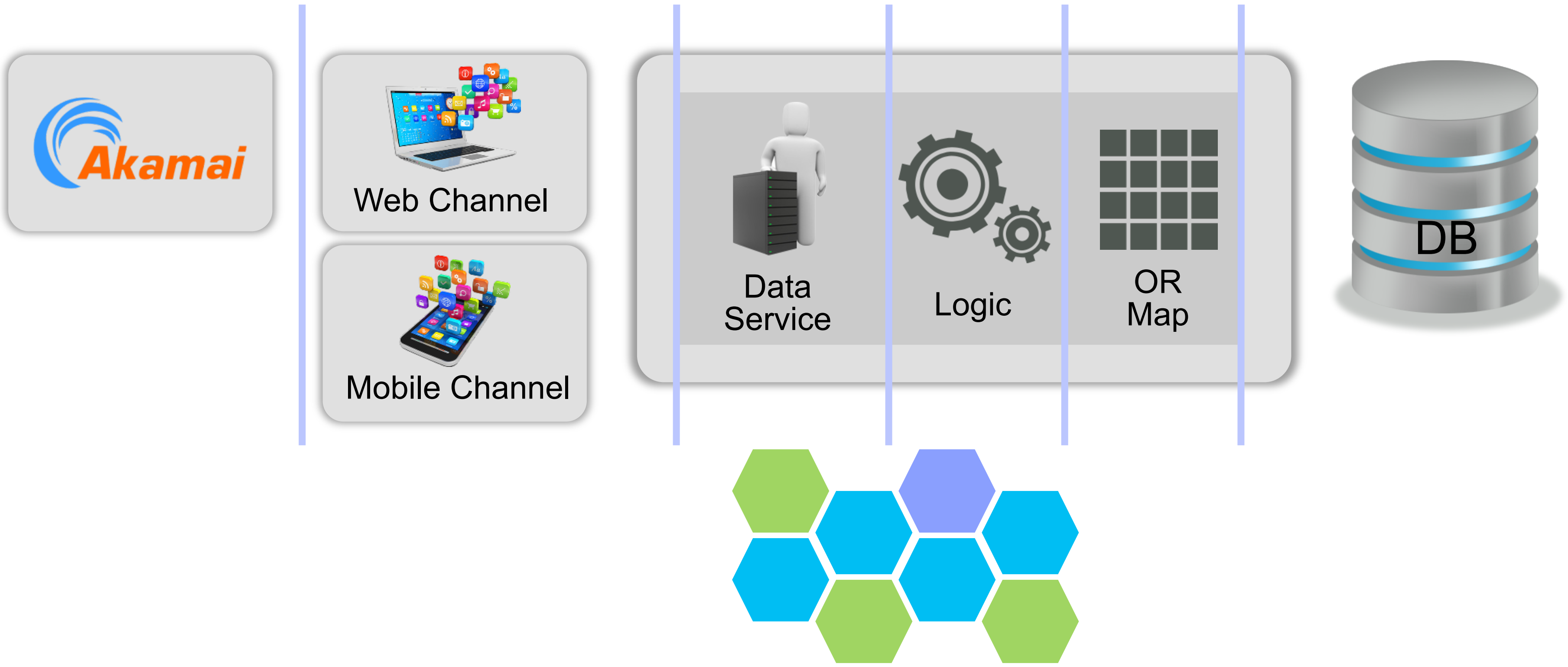
E-mail, SMS, Pop-up, Click-thru Promotions, Web Crawlers + **All of the above** = **Transaction Overload**

Where do we cache?

A database cache? A page fragment cache? A service Cache?

TOO SPECIFIC!

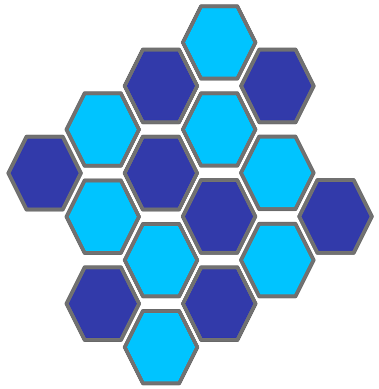
- A cache is a tool for **reducing application path length**
- OR the **distance data has to travel** before it gets to the customer/data sink



Caching requirements and benefits

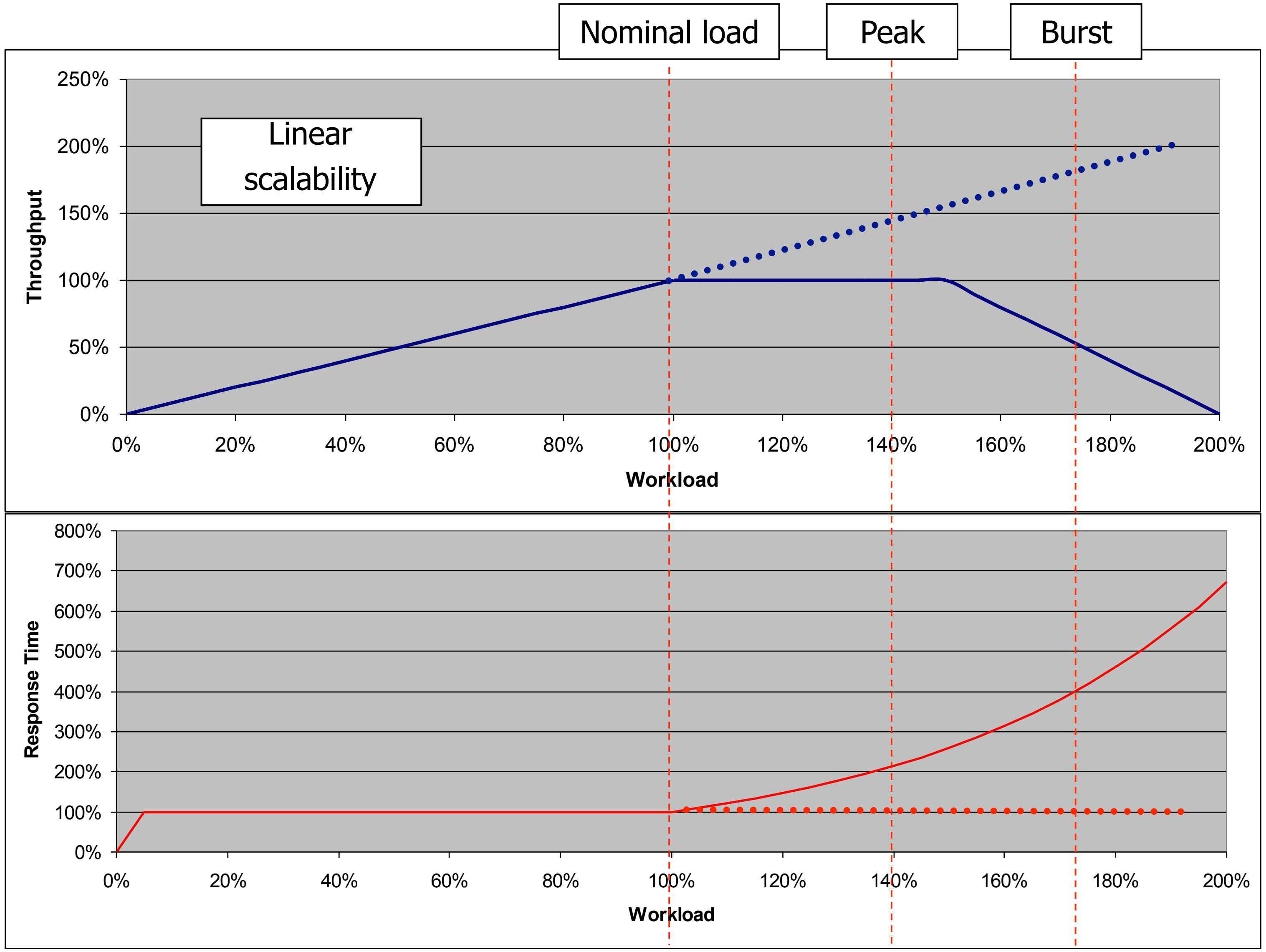
- **Requirements**
  - Read-mostly data access
  - Data change rate << data access rate
  - Same data set accessed across multiple request

- **Expected benefits**
  - Fast data access
    - Low latency
    - Constant millisecond level response time
  - Linear scalability
  - Off-load backends



Throughput & Response Time vs Workload

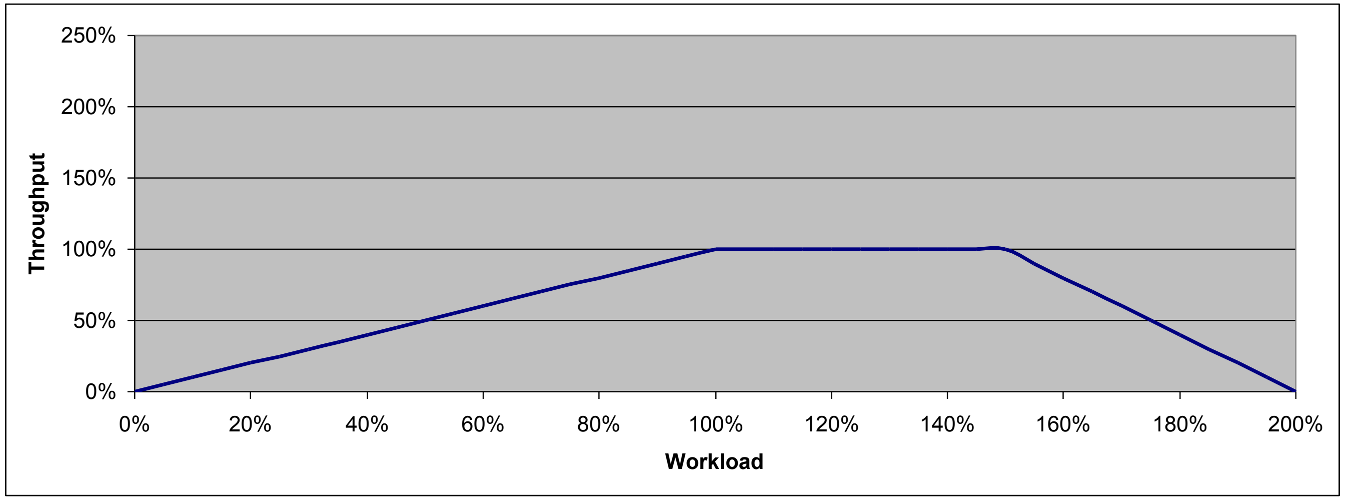
- Caching can provide improved response times and linear scalability of your applications, essential high traffic volumes, peaks, and bursts of transactions.



Gee, I already do caching…

Problems with local caching:

- Local cache doesn't scale
- Local cache is not fault tolerant or highly available
- Need to handle invalidation across a cluster
- Local cache is typically single function or application specific
- Local cache memory requirements could actually degrade performance due to Garbage Collection cycles on large JVM heap sizes
- Resource contention for managing local cache (CPU, memory, I/O)



Why you might need elastic caching

Scalability issues with database servers

- Adding extra hardware is not easy
- Licensing costs

Large volume of data

- Ability to handle volumes of data without slowing down data access
- Handle data surges during product launches and live events

Fault tolerance and self-healing

- Need for automatic mechanisms to avert system failure affecting end-users
- Data integrity

Data redundancy and replication

- Maintain data reliability in case of failover

What is a Data Grid?

Distributed in-memory object cache

- Elastic, scalable, coherent in-memory cache
- Dynamically caches, partitions, replicates and manages application data and business logic across multiple servers

Capable of massive volumes of transactions

- Provides qualities of service such as transaction integrity, high availability, and predictable response times

Self-healing, allow scale-out / scale-in

- Automatic failure recovery
- On-the-fly addition / removal of memory capacity

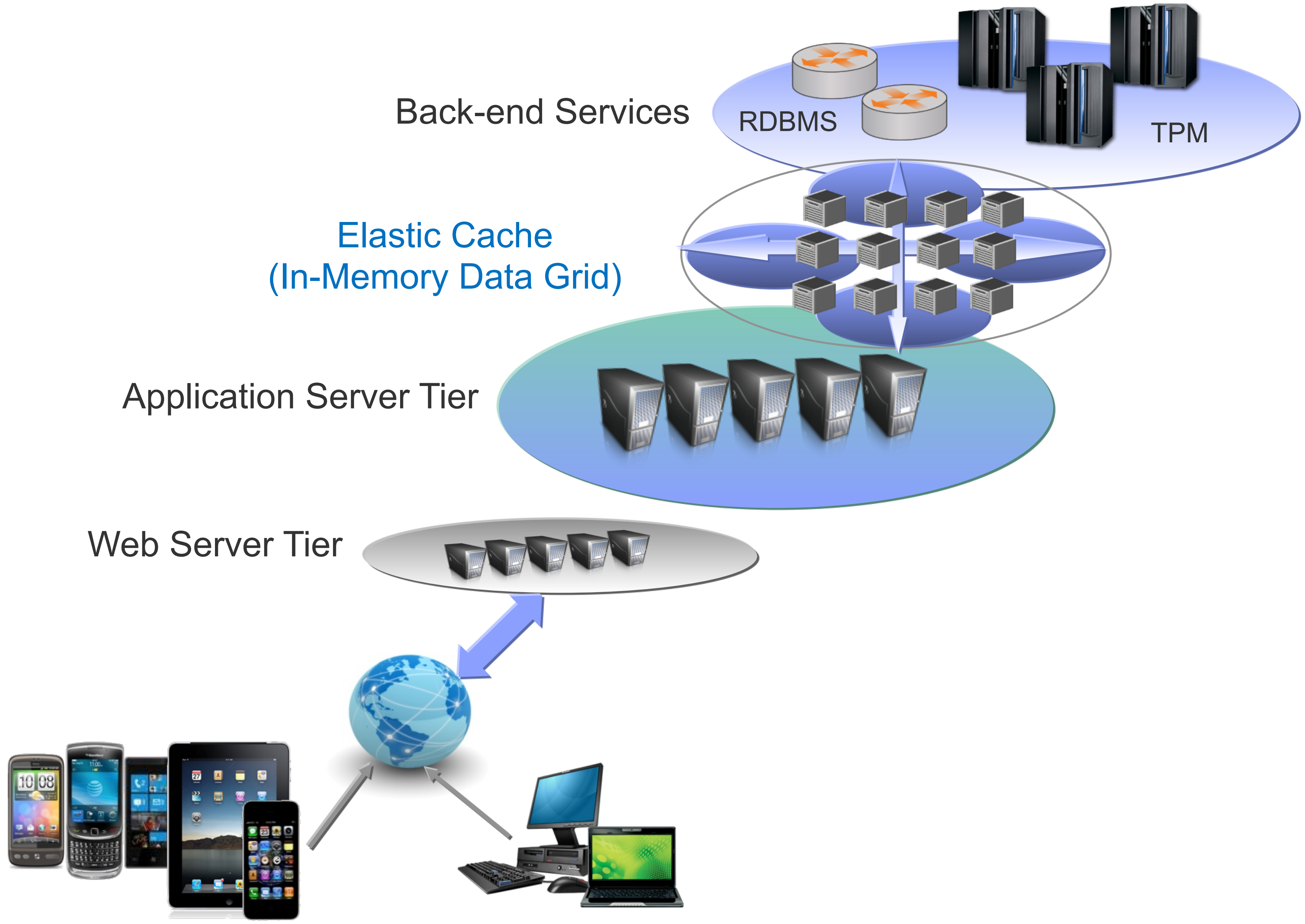
Splits a given dataset into partitions

- Primary and Replica shards (Fault tolerant)

Characteristics of a Data Grid

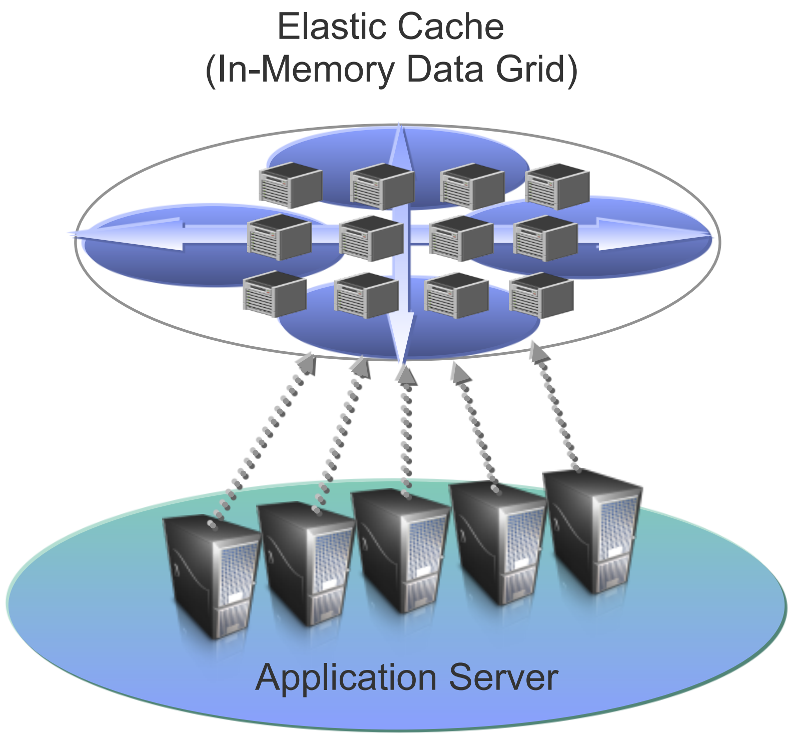
- Data is de-normalized
- Data Grid is transactional
- Simple APIs
  - Get, Insert, Update, Delete
  - SQL-like query language
- Can be horizontally partitioned
  - Linear Scalability
- Often transient or referential data
  - HTTP sessions, user profile, etc
  - Mainframe DBMS / MIPS offloading
  - Read-only or read-mostly
  - Can tolerate some staleness

Enterprise Architecture



1 Application State Store Pattern (like HTTP session data)

Applications use single coherent, highly- available, scalable cache



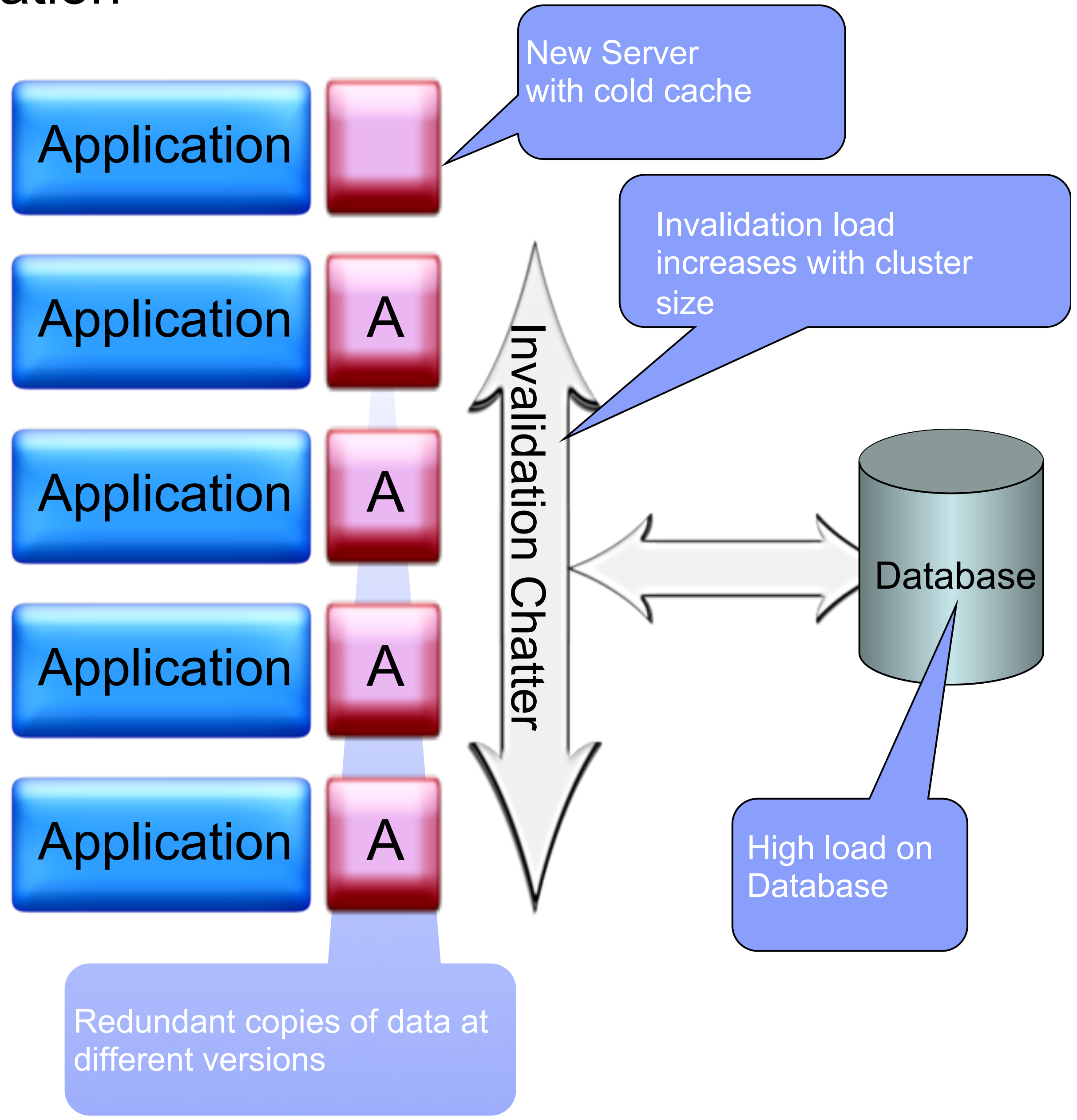
- Single replacement for multiple local caches
  - Increases the size of cache vs replicated caches
- Consistent response times
- Reduces Application Server JVM heap size
  - Improved memory utilization - more memory for applications
  - Faster Application Server start-up
- Removes invalidation chatter of local caches
- Applications move application state to grid
  - Stateless applications scale elastically
- Application state can be shared across data centers for high availability

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

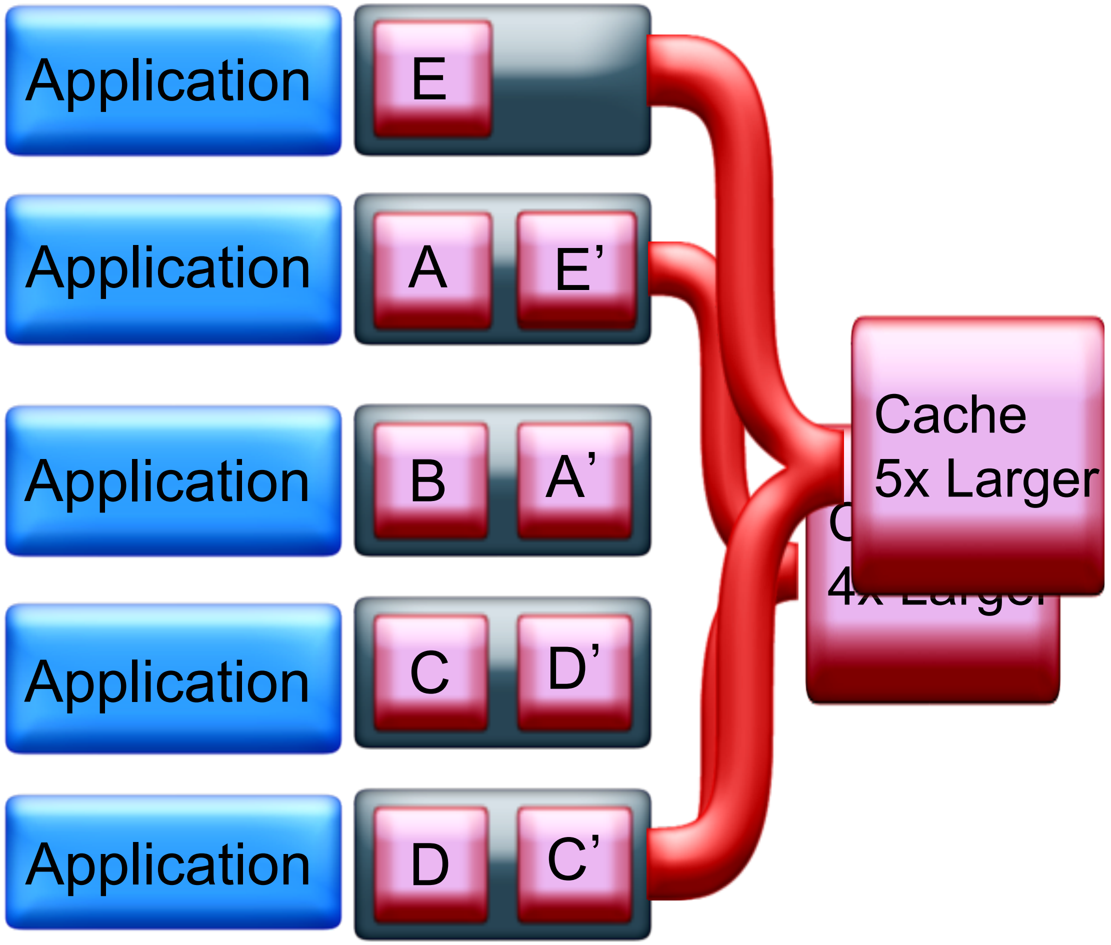
WebSphere eXtreme Scale Cache Operation

- Cache capacity determined by total cluster size
- Size of each cache = M
- # JVMs = N
- Total Cache = M x N

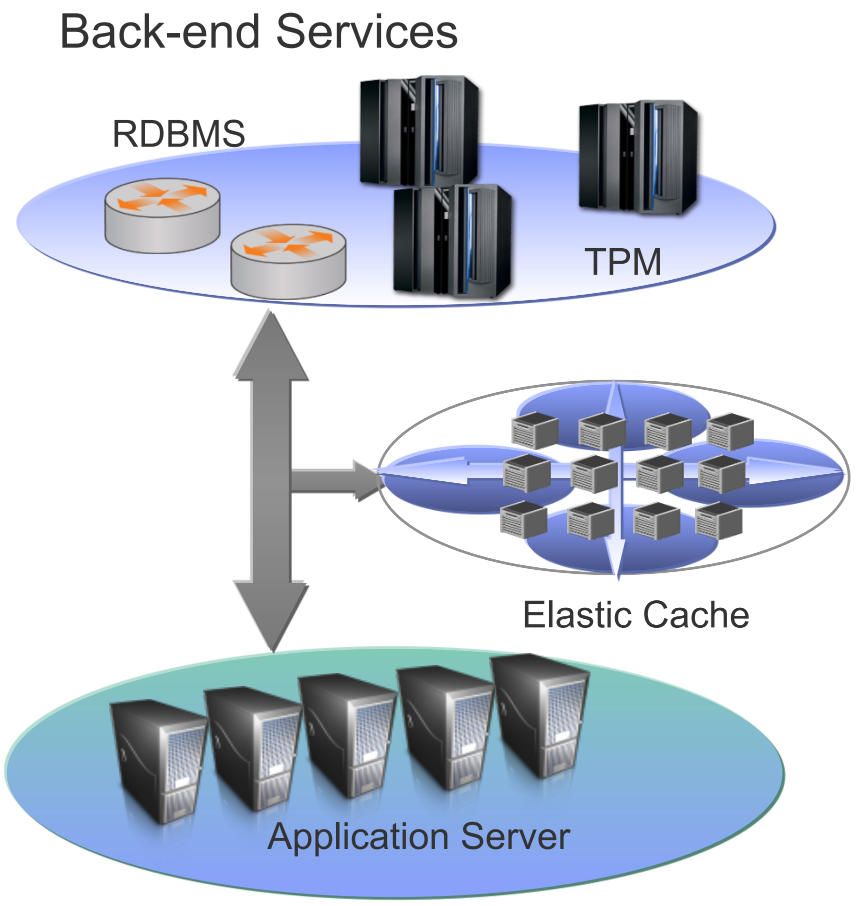
- No invalidation chatter

- Linearly scalable

- Less load on database and no cold start spikes

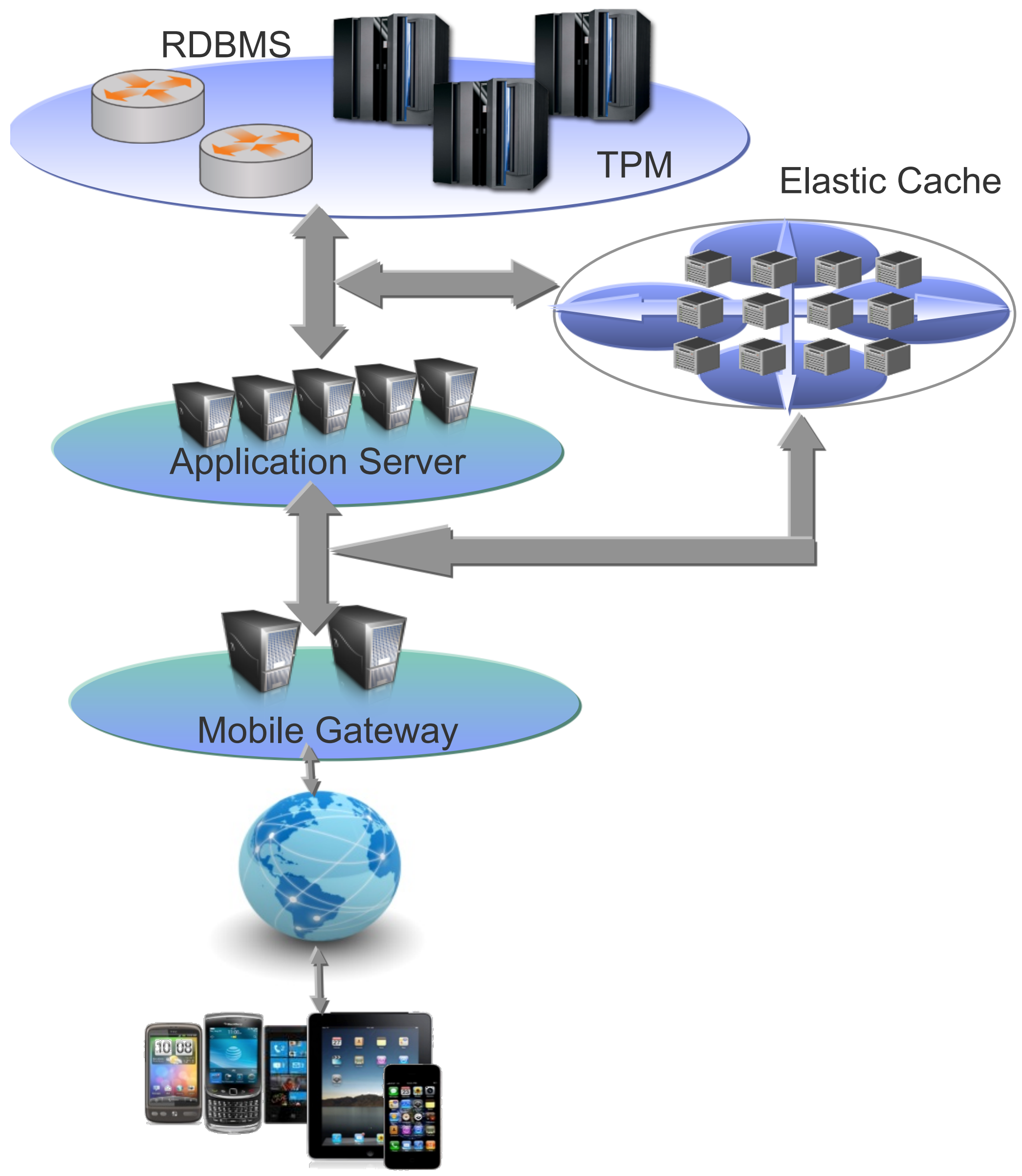


2 Side Cache Pattern



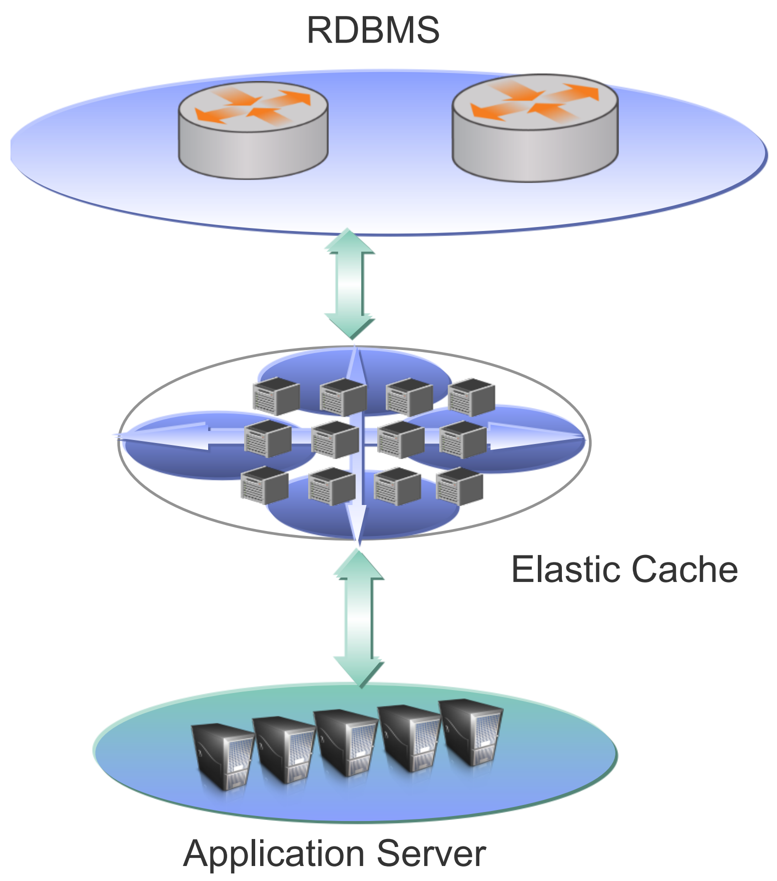
- Client first checks the grid before using the data access layer to connect to a back end data store.
- If an object is not returned from the grid (a cache “miss”), the client uses the data access layer as usual to retrieve the data.
- The result is put into the grid to enable faster access the next time.
- The back end remains the system of record, and usually only a small amount of the data is cached in the grid.
- An object is stored only once in the cache, even if multiple clients use it. Thus, more memory is available for caching, more data can be cached, which increases the cache hit rate.
- Improve performance and offload unnecessary workload on backend systems.

Mobile Gateway Acceleration



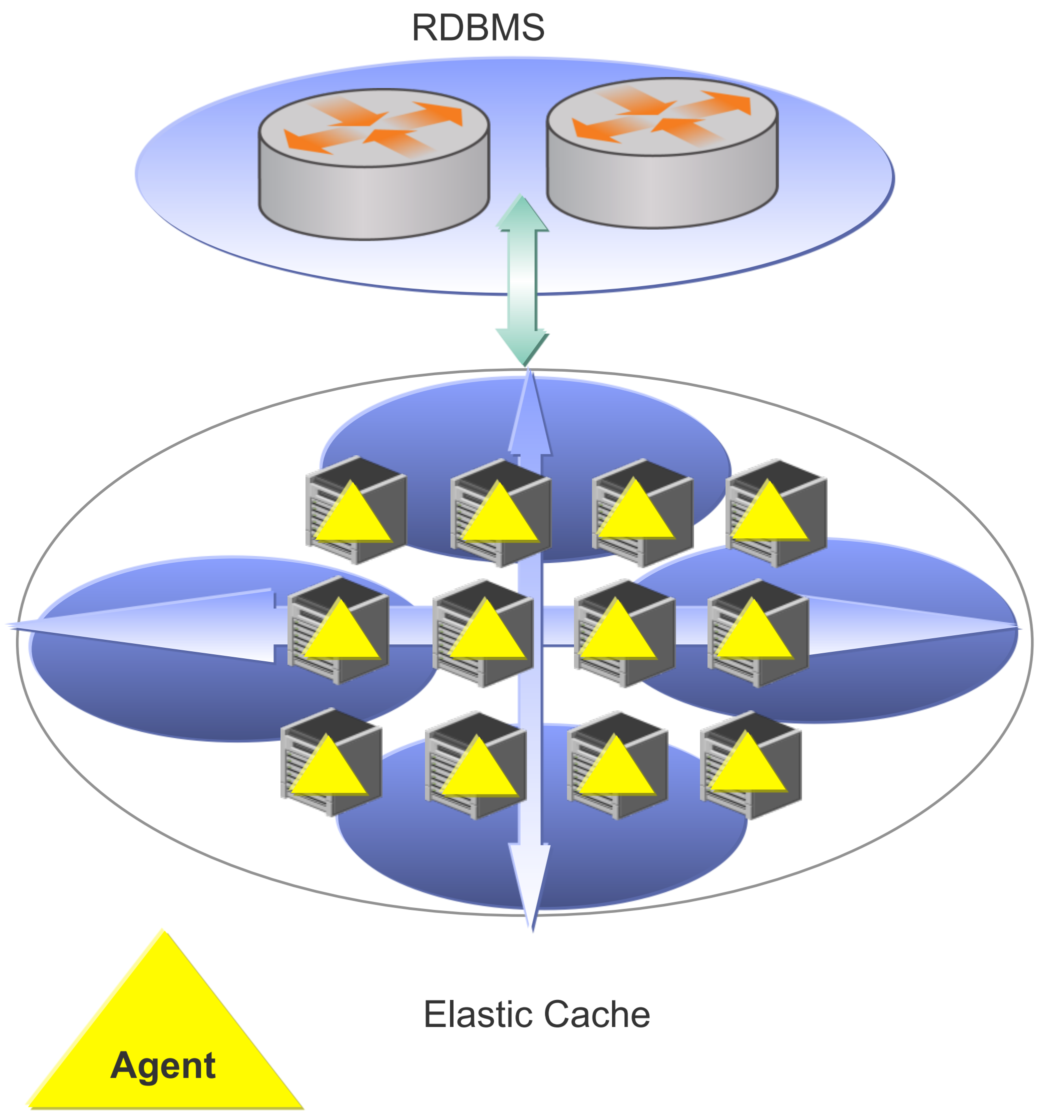
- By integrating Elastic Cache with Mobile Gateway, users can see improved performance without the penalty of having to scale to a large cluster of Mobile Gateways
- Use Side cache to cache XSLT transforms
- Directly access the Elastic Cache to retrieve cached objects
- Use Elastic Cache to provide session state for stateless communication

3 In-line cache – Database shock absorber



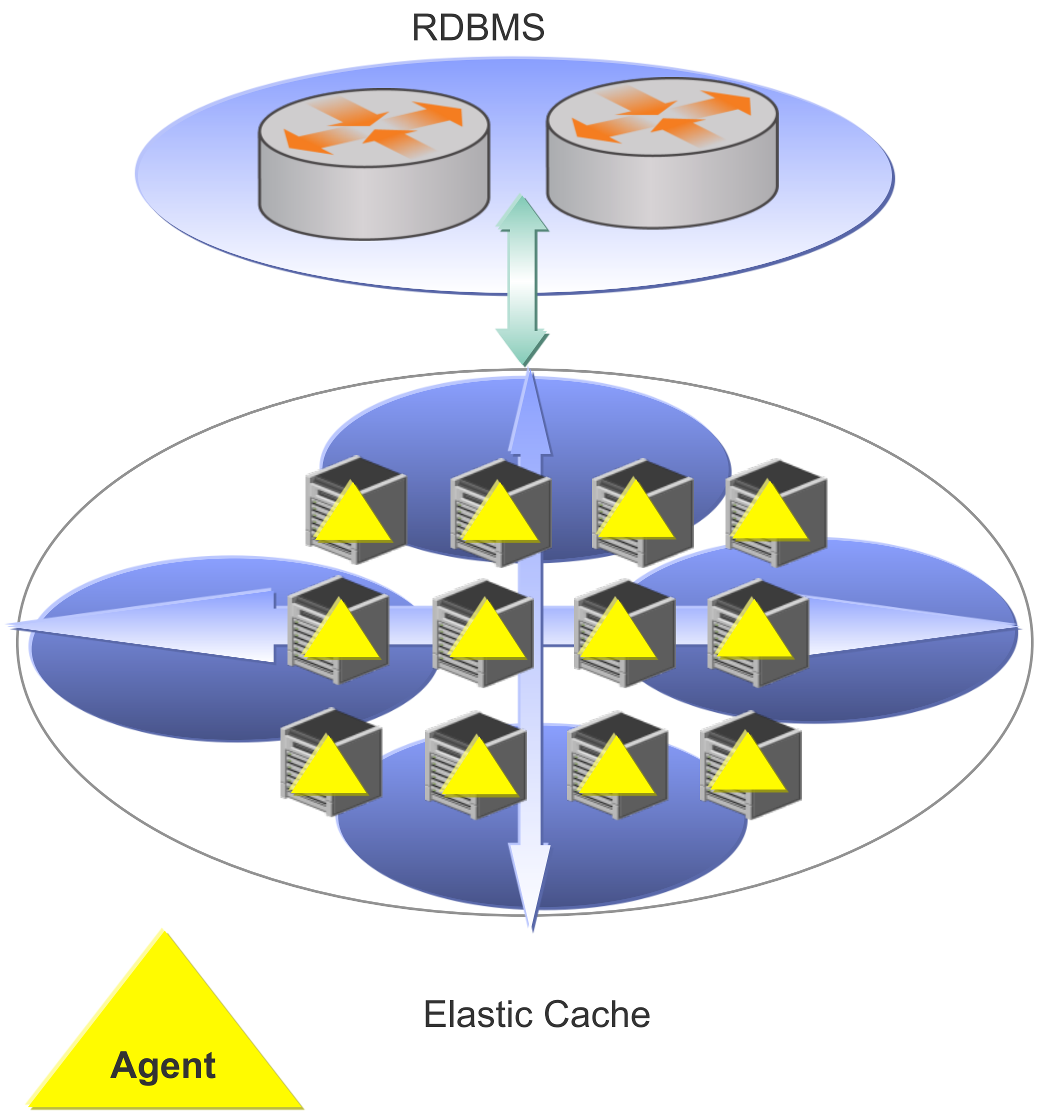
- The grid can be used as a special data access layer where it is configured to use a loader to get data from the back-end system.
  - Read through cache
  - Write through cache (Synchronous writes)
  - Write-behind cache (Asynchronous writes)
- System of Record Data Store
  - Cache is used as the system of record
  - Write behind technology pushes changes asynchronously to the backend.
    - Changes batched
    - Only last change written
  - Runs through backend outages!
- Benefits
  - Writes faster (memory vs. disk speed)
  - Backend load reduced, throughput improved
  - Increased availability and scalability

4 eXtreme Transaction Processing



- Lowest possible latency
- Application code (Agent) runs in the grid itself
  - Map/Reduce API supported
  - Results sent back to client
- Events routed to correct partitions for processing
- Databases relegated to durable log and reports

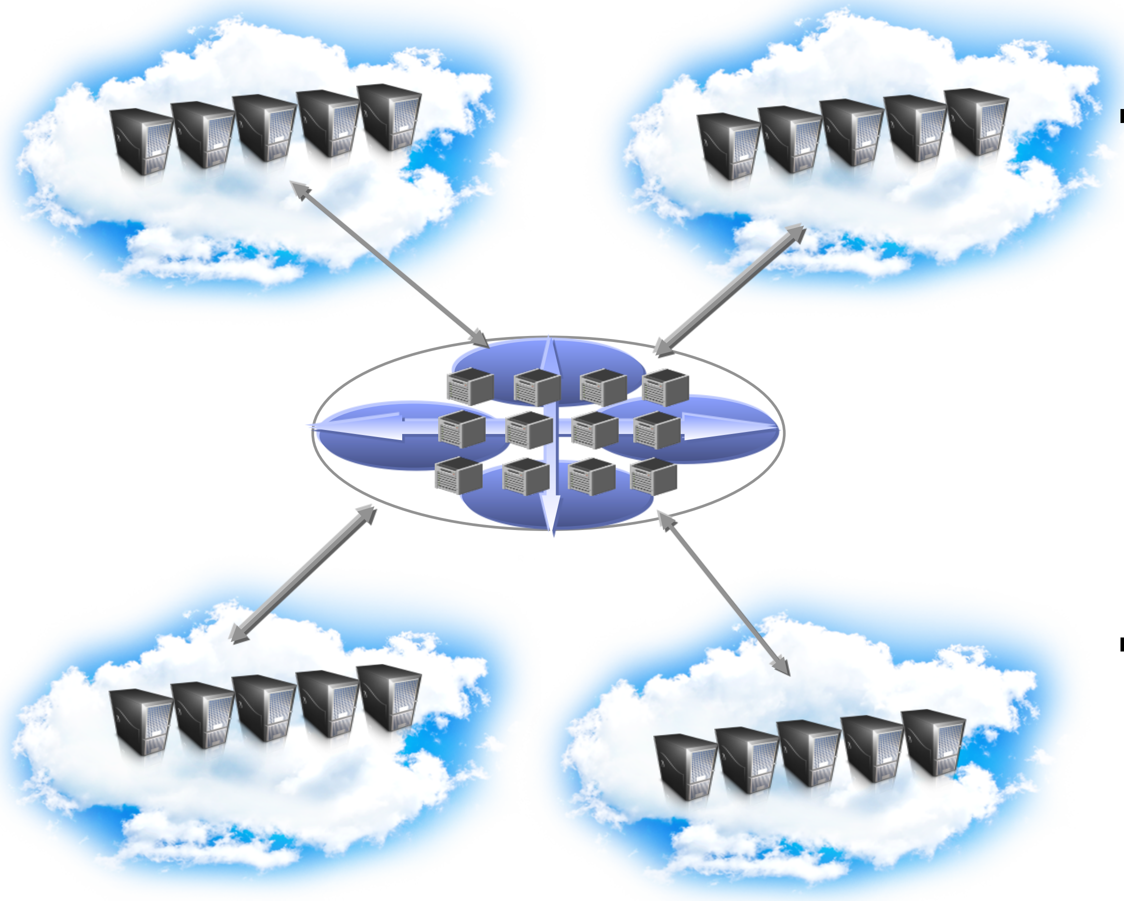
Map Reduce Parallel Processing



- Parallel Map
  - Allows the entries for a set of Entities or Objects to be processed and returns a result for each entry processed
- Parallel Reduction
  - Processes a subset of the entries and calculates a single result for the group of entries
- Since the Elastic Cache is the system of record, there is little to no load on the back-end data stores

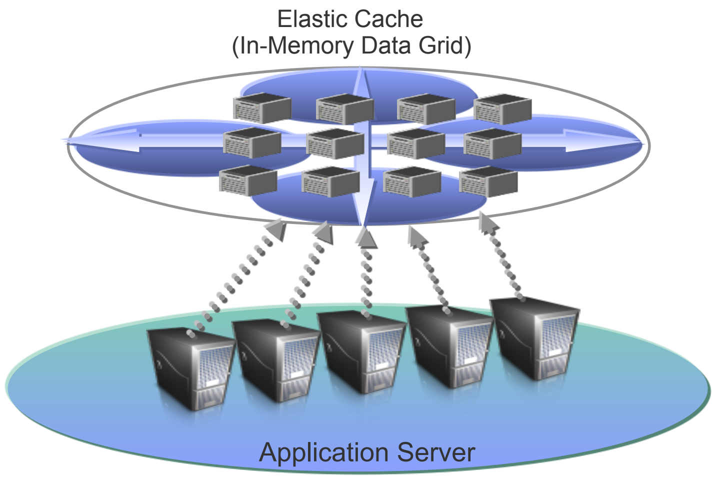
Elastic Cache Shared Service

PureApplication System, PureApplication Service, Bluemix

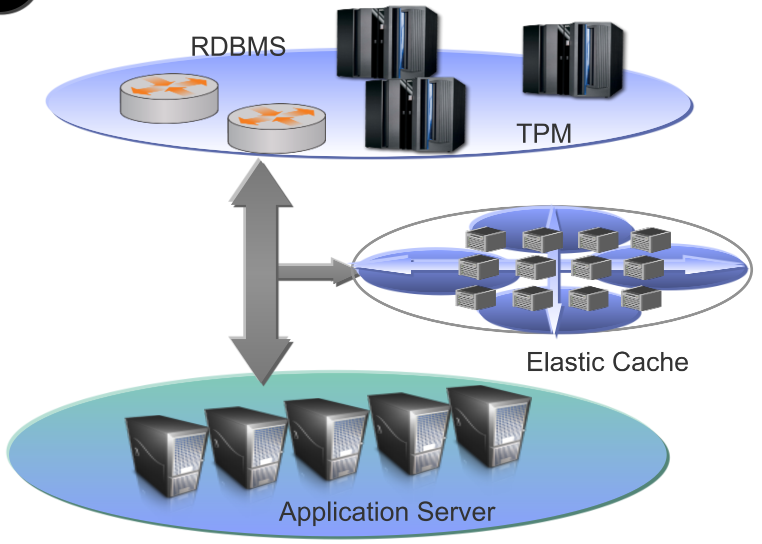


- Provides Elastic Caching resource for cloud based architectures
- Elastic Cache service is multi-tenant
  - Support grid capping
  - Individual maps per cloud group
  - Authentication / Authorization per map/ grid
- Used for
  - Simple Cache
  - HTTP session distribution
  - Dynamic Cache provider

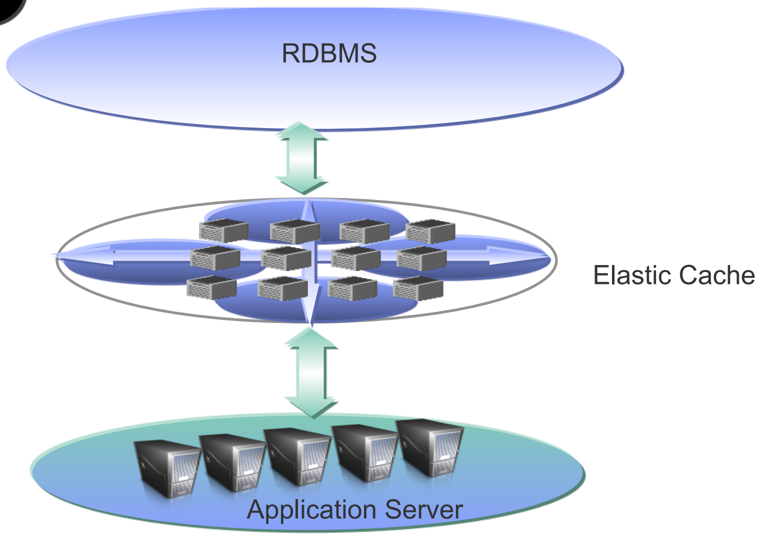
1 Application State Store Pattern



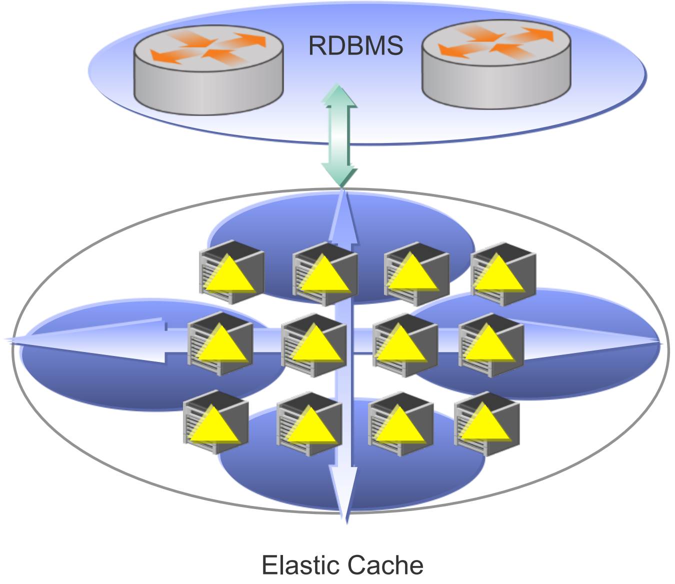
2 Side Cache Pattern



3 In-line cache



4 eXtreme Transaction Processing



IBM Elastic Caching Delivers

Consistent Response Times, High Availability of Data & Linear Scalability for Enterprise-wide Data Grids

WebSphere eXtreme Scale V8.6

A powerful, scalable, elastic in-memory grid for your business-critical applications

- Java and .NET applications can now interact natively with the same data in the same data grid, leading the way toward a true enterprise-wide data grid.
- A new REST Gateway provides simple access from other languages.
- WXS 8.6 delivers a faster, more compact serialization format called eXtreme Data Format (XDF), which is neutral to programming languages.
- A new transport mechanism, eXtreme IO (XIO) removes the dependency on the IBM ORB, enabling easier integration with existing environments.
- Built in pub/sub capabilities enable WXS 8.6 to update client "near caches" whenever data is updated, deleted, or invalidated on the server side.
- API enhancements enable continuous query or data that is inserted and updated in the grid.

DataPower XC10 Appliance V2.5

Rapid, "drop-in" use with a broad range of Java and non-Java application environments

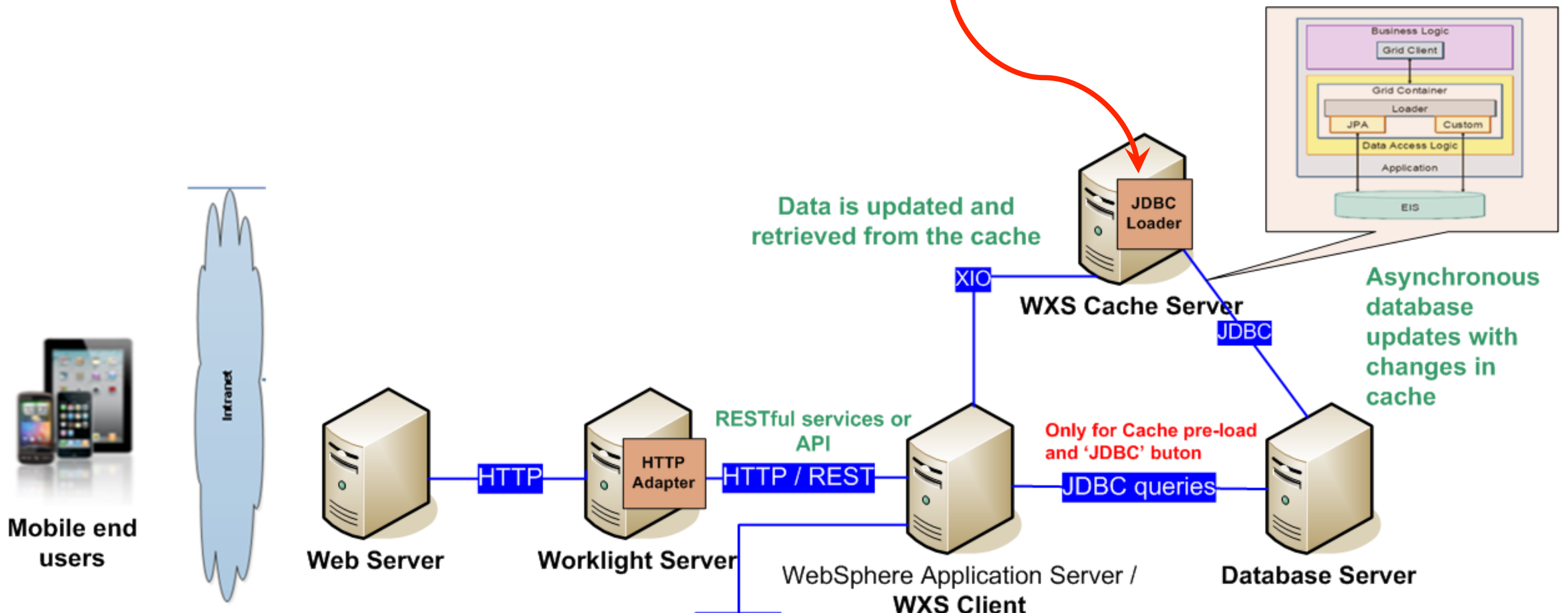
- Rapid drop-in use across a broad range of application server technologies and programming languages
- New data format (eXtreme Data Format –XDF) improves performance and allows data to be shared natively between Java & .NET applications
- Built in notification infrastructure allows for client-side event notification, continuous query cache and near-cache invalidation
- Improved usability, serviceability
- Supports FIPS security protocol for government and financial sector compliance
- Improved performance
- Improved monitoring and administration capabilities

Mobile / Caching DEMO integration: Goals

- **Goals:**
  - Reduce response time for REST Service (or API) providing the offers and incentive information to IBM bank customers
  - Provide increased throughput of the IBM bank services
  - Ensure that increased demand of the mobile bank features do not overload the back-end services
  - Reduce the overall load to the back-end resource intensive resources such as the databases
  - Updates to the offerings and details (directly in the cache) are immediately available to mobile banking customers

Mobile / Caching DEMO integration: Goals

- **How:**
  - Implement an elastic (In-Line) caching layer between the Application servers and the backend database
  - Provide cache preloading to ensure high cache hit ratio from the very first transaction
  - Provide an asynchronous loader to keep the back-end database in synch with changes to the cache content. Loaders run inside the Grid Container.



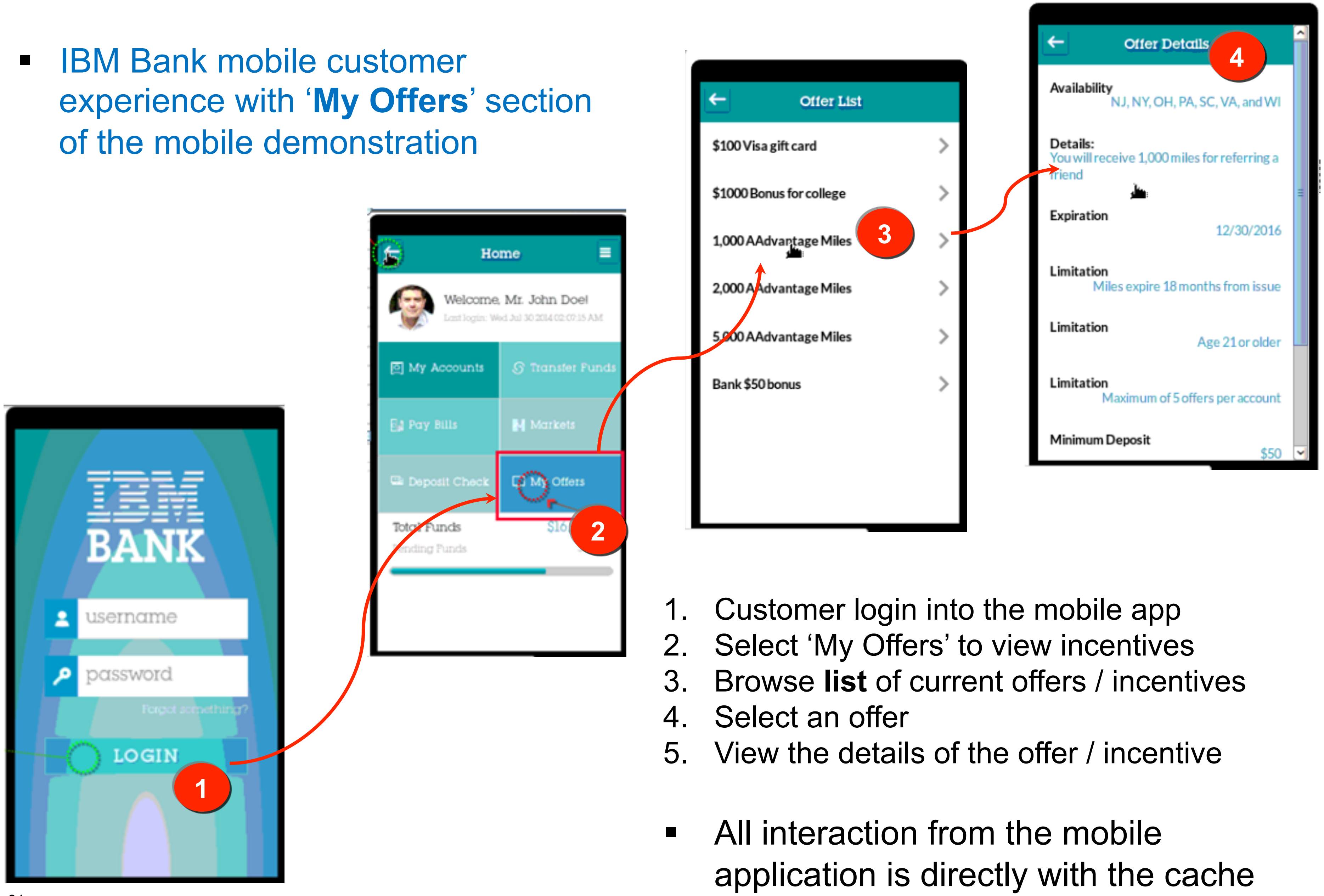
1. Load cache using JDBC client loader
2. Retrieve and update data in cache
3. The asynchronous JDBC loader updates the DB based on changes to the cache

The WXC Client makes direct connect to JDBC when the 'JDBC' button is clicked to retrieve data. This is for demo purposes.

The WXS client also makes direct JDBC queries to the database when pre-loading the cache.

Mobile and Caching integration demonstration

- IBM Bank mobile customer experience with '**My Offers**' section of the mobile demonstration



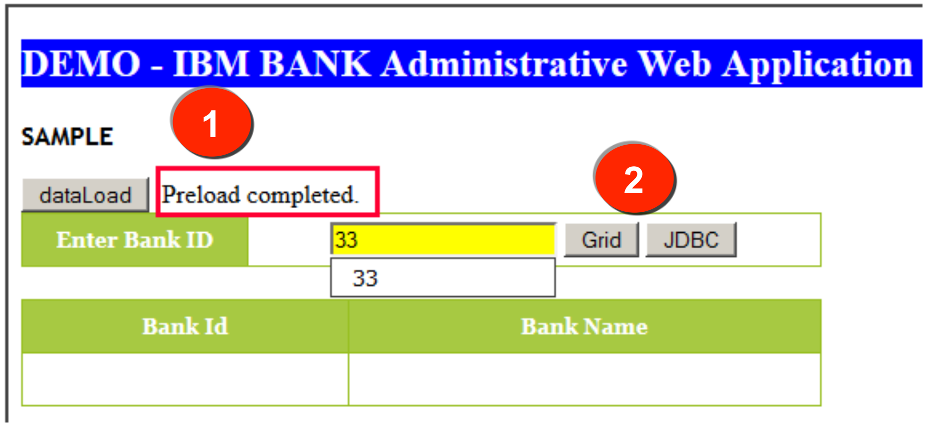
$100 Visa gift card
$1000 Bonus for college
1,000 AAdvantage Miles
2,000 AAdvantage Miles
5,000 AAdvantage Miles
Bank $50 bonus

Availability	NJ, NY, OH, PA, SC, VA, and WI
Details:	You will receive 1,000 miles for referring a friend
Expiration	12/30/2016
Limitation	Miles expire 18 months from issue
Limitation	Age 21 or older
Limitation	Maximum of 5 offers per account
Minimum Deposit	$50

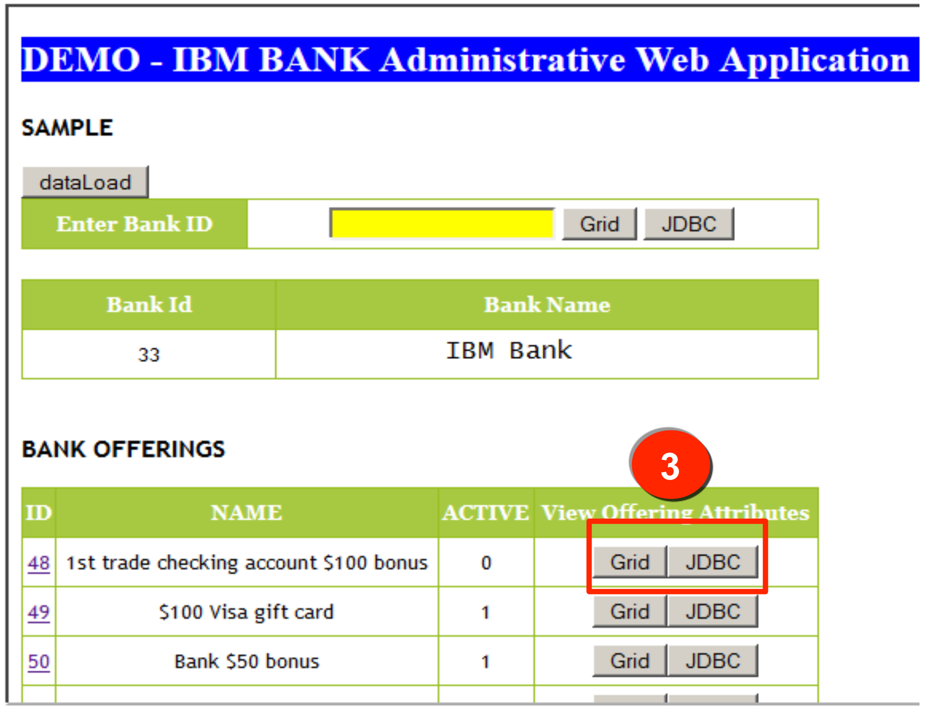
1. Customer login into the mobile app
2. Select 'My Offers' to view incentives
3. Browse **list** of current offers / incentives
4. Select an offer
5. View the details of the offer / incentive

- All interaction from the mobile application is directly with the cache

Administrative views in Mobile and Caching demo



Bank Id	Bank Name

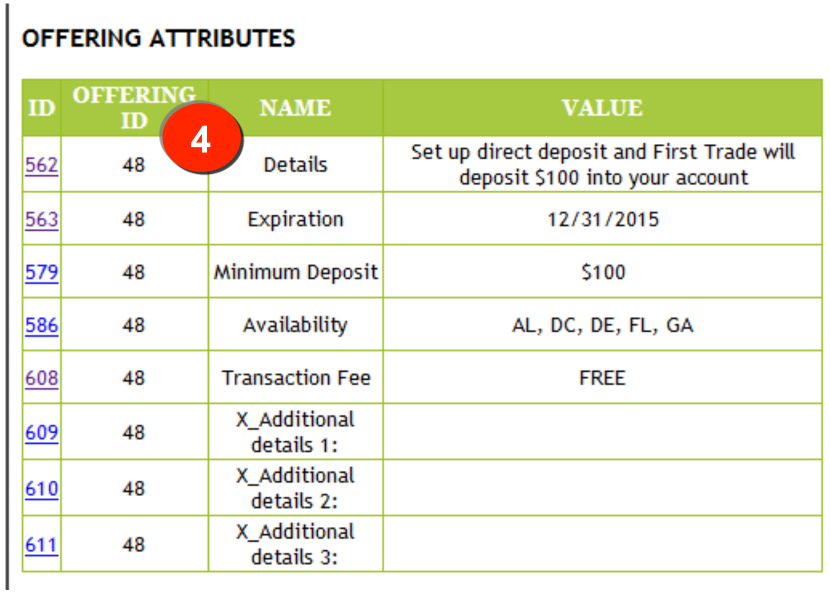


Bank Id	Bank Name
33	IBM Bank

BANK OFFERINGS

ID	NAME	ACTIVE	View Offering Attributes
48	1st trade checking account $100 bonus	0	Grid JDBC
49	$100 Visa gift card	1	Grid JDBC
50	Bank $50 bonus	1	Grid JDBC

- IBM Bank Administrator experience
  - View offers from cache and/or database



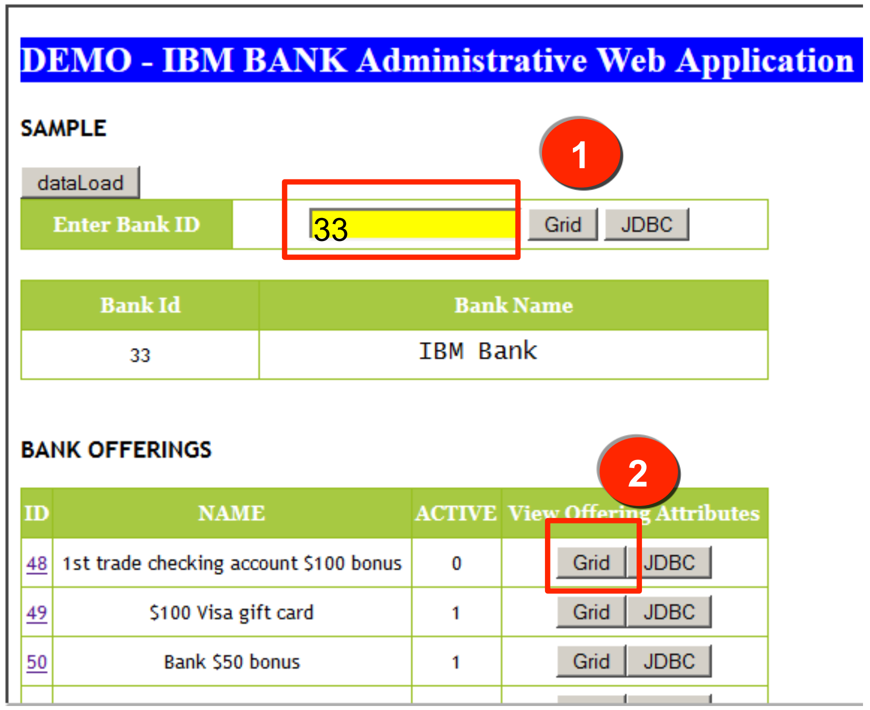
OFFERING ATTRIBUTES

ID	OFFERING ID	NAME	VALUE
562	48	Details	Set up direct deposit and First Trade will deposit $100 into your account
563	48	Expiration	12/31/2015
579	48	Minimum Deposit	$100
586	48	Availability	AL, DC, DE, FL, GA
608	48	Transaction Fee	FREE
609	48	X_Additional details 1:	
610	48	X_Additional details 2:	
611	48	X_Additional details 3:	

1. Admin pre-loads the cache from the database
2. Select ‘IBM Bank (ID=33)’ to view offers from the cache or database
3. Select an offer from the list to view the details page of the specific offering
4. Work with specific details of the offering

- Administrator can also Update offers in the cache / database

Administrative update of Offering details / attributes



DEMO - IBM BANK Administrative Web Application

SAMPLE

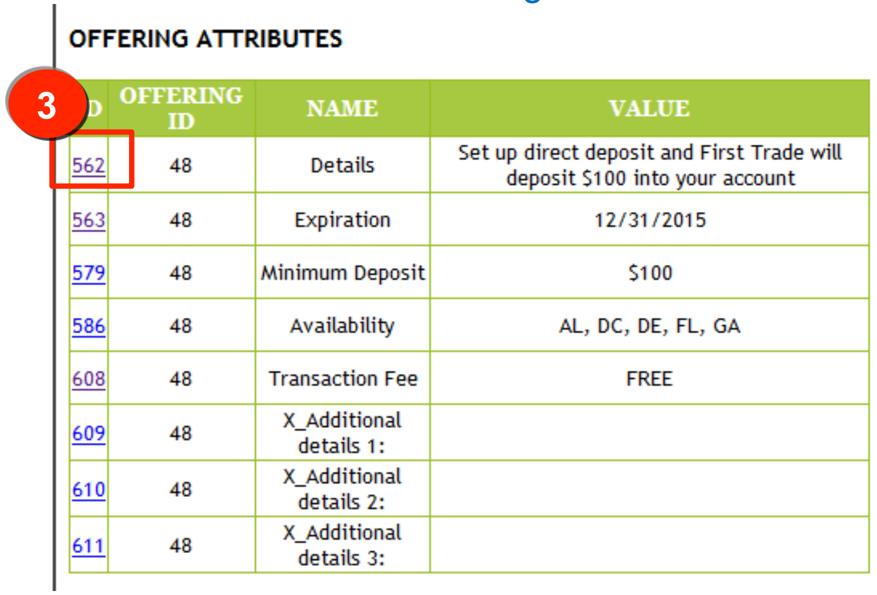
Enter Bank ID	33

Bank Id	Bank Name
33	IBM Bank

BANK OFFERINGS

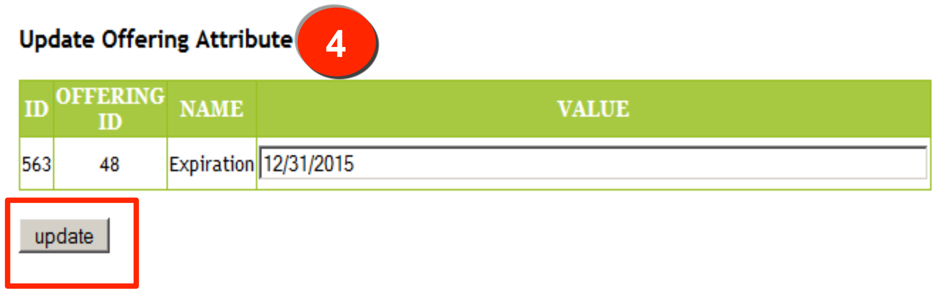
ID	NAME	ACTIVE	View Offering Attributes
48	1st trade checking account $100 bonus	0	Grid JDBC
49	$100 Visa gift card	1	Grid JDBC
50	Bank $50 bonus	1	Grid JDBC

- IBM Bank Administrator experience
  - Update offering details
    - Set an appropriate expiration date for an offering



OFFERING ATTRIBUTES

D	OFFERING ID	NAME	VALUE
562	48	Details	Set up direct deposit and First Trade will deposit $100 into your account
563	48	Expiration	12/31/2015
579	48	Minimum Deposit	$100
586	48	Availability	AL, DC, DE, FL, GA
608	48	Transaction Fee	FREE
609	48	X_Additional details 1:	
610	48	X_Additional details 2:	
611	48	X_Additional details 3:	



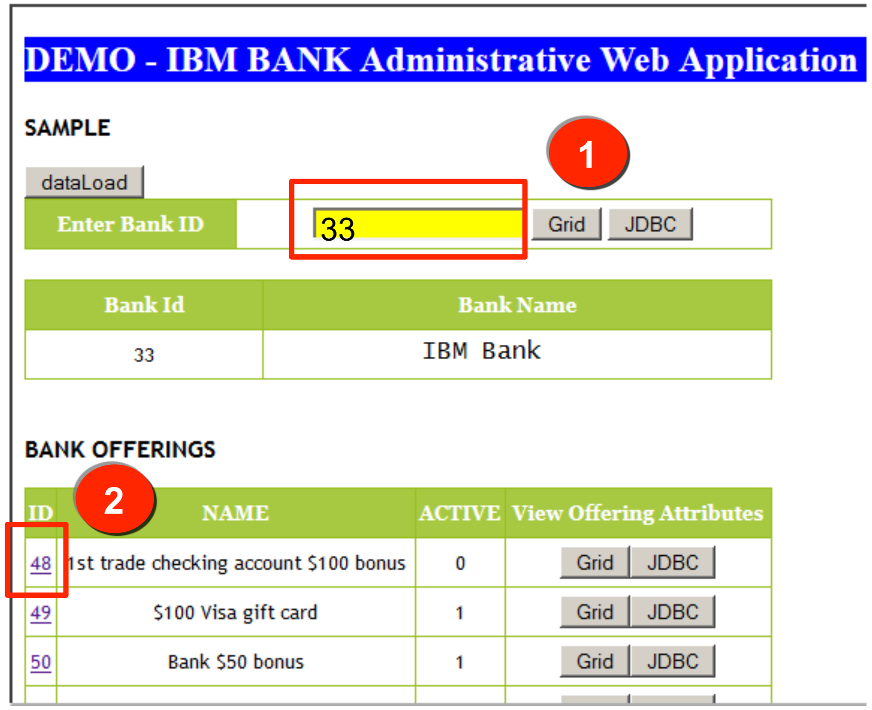
Update Offering Attribute

ID	OFFERING ID	NAME	VALUE
563	48	Expiration	12/31/2015

1. Admin selects IBM bank from Grid
2. Admin Selects an offering from the Grid
3. Admin Clicks one of the detailed attributes to update (Expiration date)

- All updates are to the grid.
- WXS loader is used to synchronize the grid changes to the database, asynchronously
  - Write-behind loader

Administrative updates to Offering in cache / database



DEMO - IBM BANK Administrative Web Application

SAMPLE

Enter Bank ID	33

Bank Id	Bank Name
33	IBM Bank

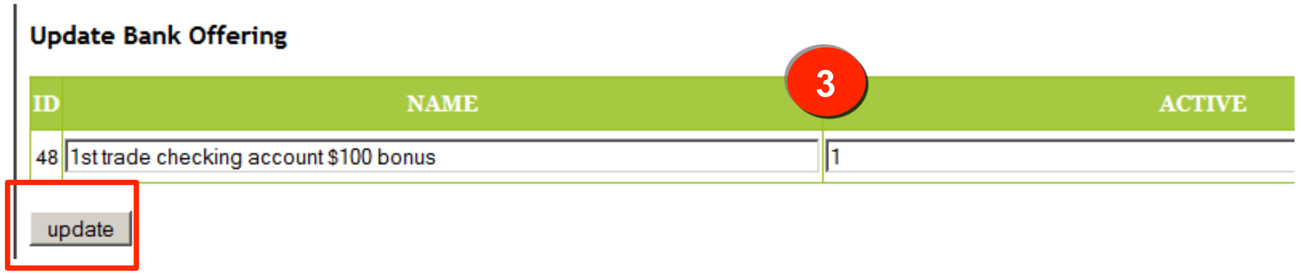
BANK OFFERINGS

ID	NAME	ACTIVE	View Offering Attributes
48	1st trade checking account $100 bonus	0	Grid JDBC
49	$100 Visa gift card	1	Grid JDBC
50	Bank $50 bonus	1	Grid JDBC

- IBM Bank Administrator experience
  - Update offering information
  - Activate an offering for mobile customers

1. Admin selects IBM bank from Grid
2. Admin Clicks one of the offerings to be updated (Activated status)
3. Admin updates the 'Active' status to activate the offering

- All updates are to the grid
- WXS loader is used to synchronize the grid changes to the database, asynchronously
  - Write-behind loader

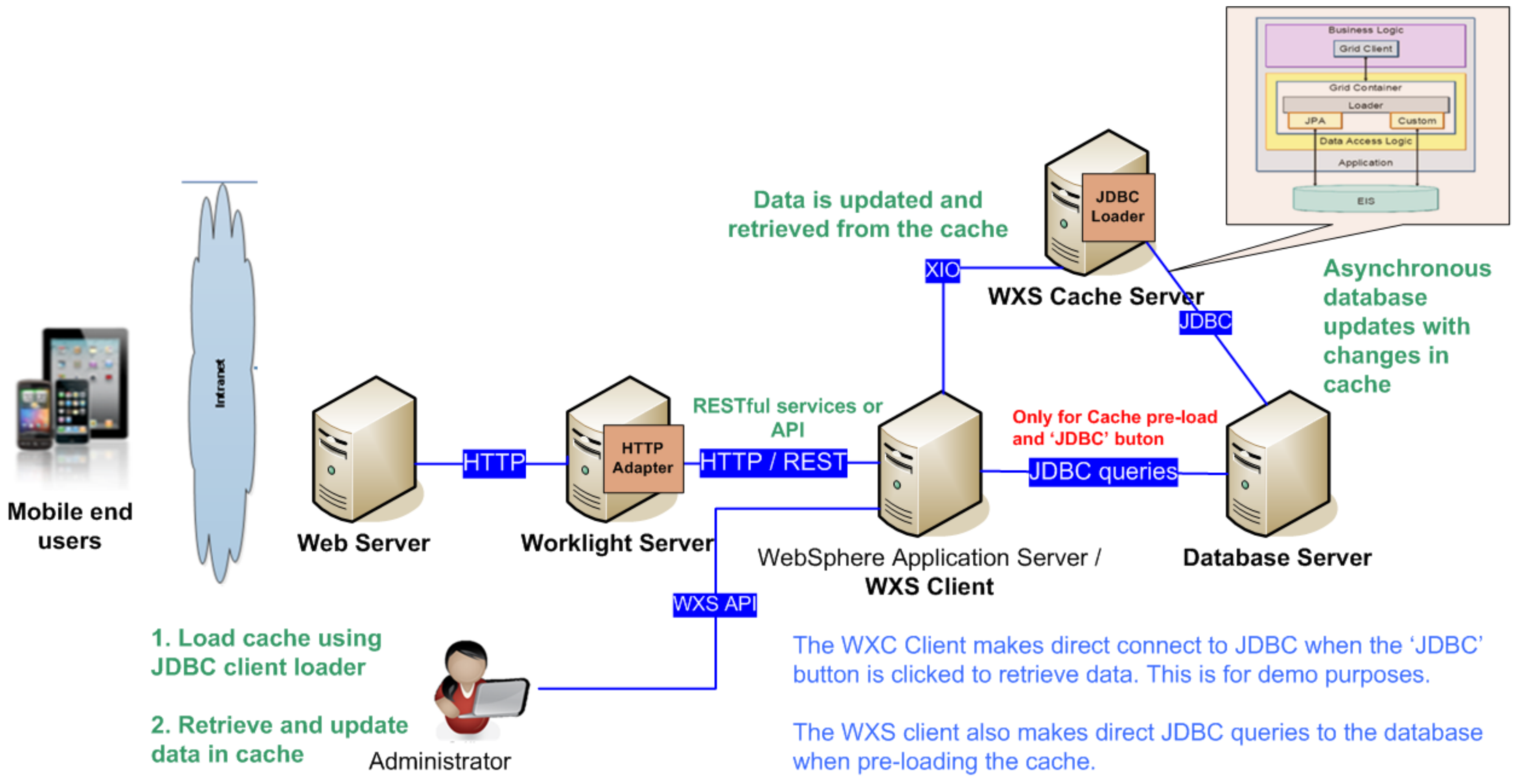


Update Bank Offering

ID	NAME	ACTIVE
48	1st trade checking account $100 bonus	1

Demonstration – IBM Bank Mobile application with elastic caching

1. Pre-load the cache with IBM Bank Offerings and Incentives
2. View ‘Active’ offerings using mobile application. All data retrieved directly from cache via REST Service or API
3. Administrator updates and activates a new offering in the cache.
   - In-line cache updates the back-end database via asynchronous ‘write-behind’ loader
4. From mobile application, view updated list of offerings in cache



1. Load cache using JDBC client loader

2. Retrieve and update data in cache

3. The asynchronous JDBC loader updates the DB based on changes to the cache

The WXC Client makes direct connect to JDBC when the ‘JDBC’ button is clicked to retrieve data. This is for demo purposes.

The WXS client also makes direct JDBC queries to the database when pre-loading the cache.