WebSphere software

IBM

WebSphere eXtreme Scale Customer Experiences

Jonathan Marshall
WebSphere Technical Professional
marshalj@uk.ibm.com

WebSphere eXtreme Scale – what's a cache?

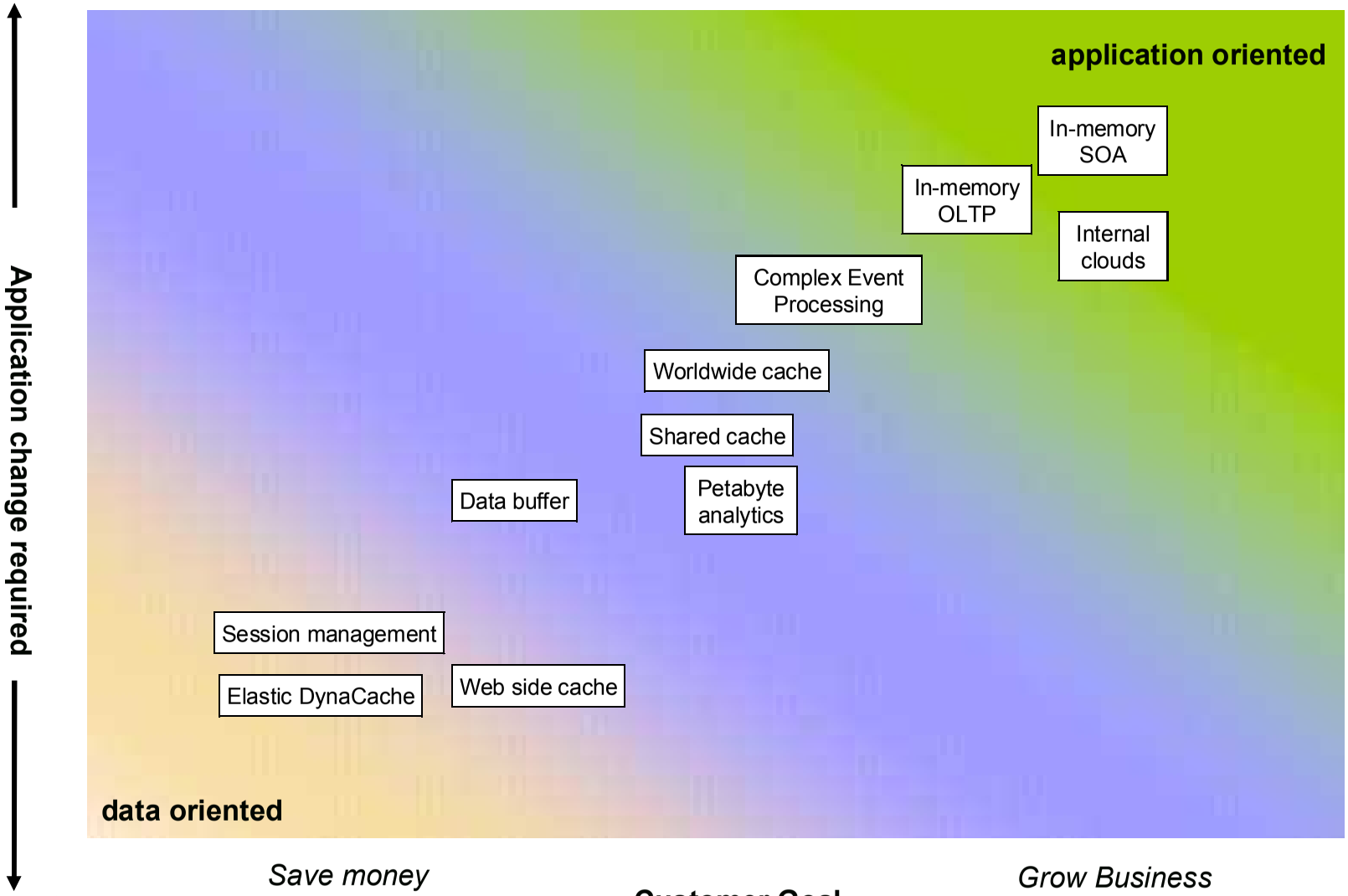
A cache allows you to get stuff faster and helps you avoid doing something over and over again (which may be redundant and may not make sense)

(far away)

(near)

(happy)

What is WebSphere eXtreme Scale?



Application Topologies Today

Web Server Tier **Application Server Tier** **Data sources**

Very large data sets affecting scalability?

Impossible performance targets?

Expensive to scale database?

Shared application state?

Very large or parallel data processing requirement?

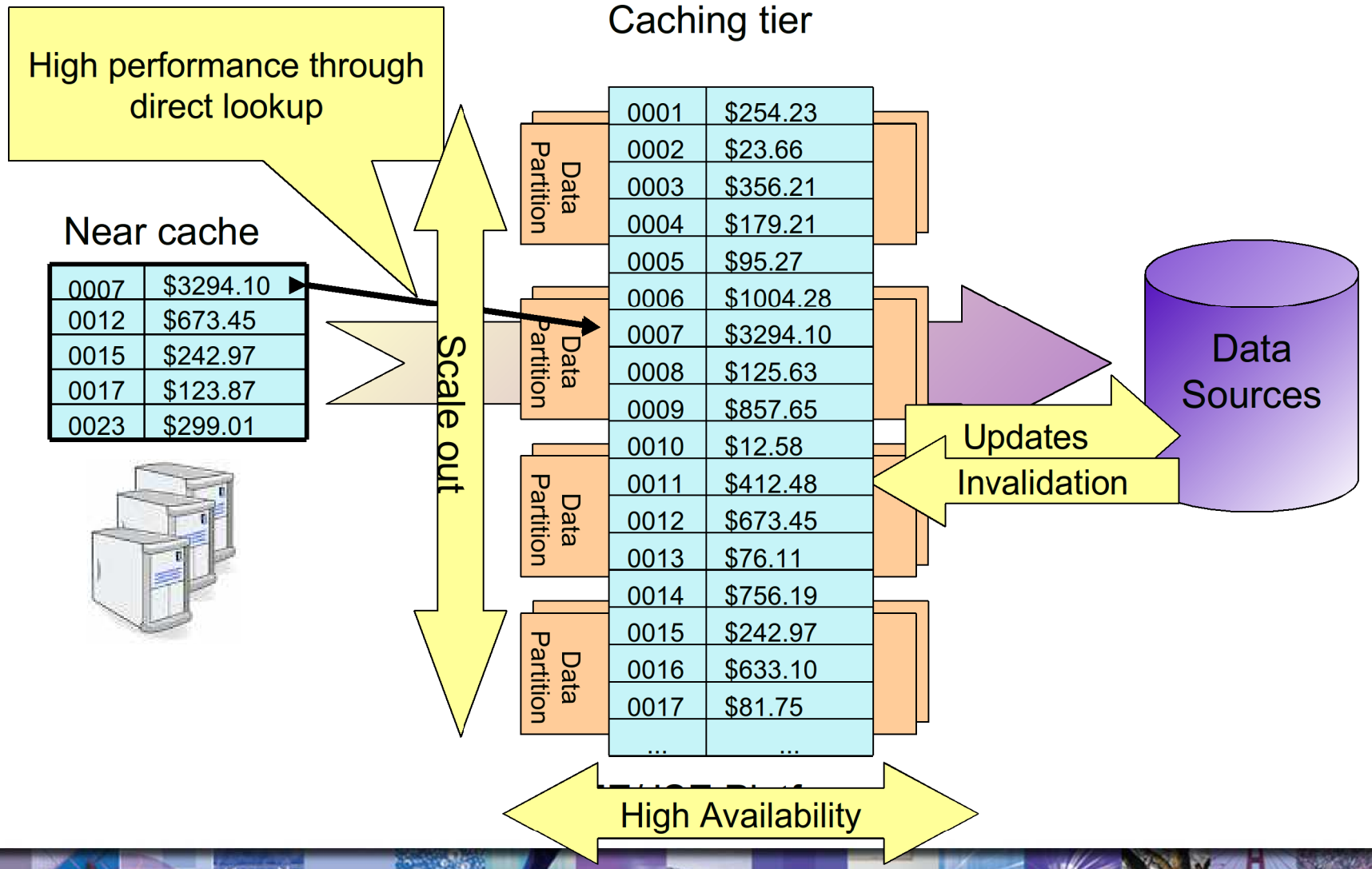
Scale with Simplicity

Web Server Tier | **Application Server Tier** | **Elastic Data Grid** | **Data sources**

Performance | **Scaling** | **Transactional consistency**

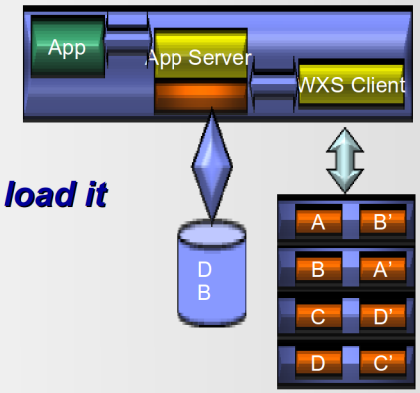
High Availability | **Security**

WebSphere eXtreme Scale – Technical Overview



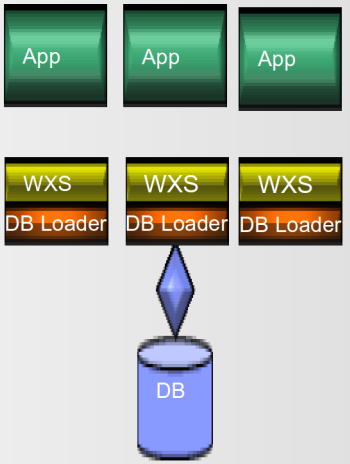
What can you do with eXtreme Scale?

Database Side Cache



Lookup data from cache
If data not found in cache, load it from database to cache

Database Side Cache and Sync Database Loader

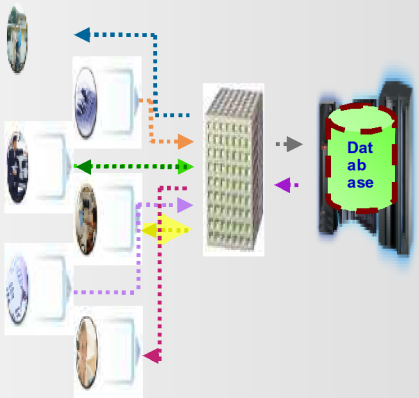


Database loader used to retrieve data from the backend database

Database loader writes back cache changes to the backend database

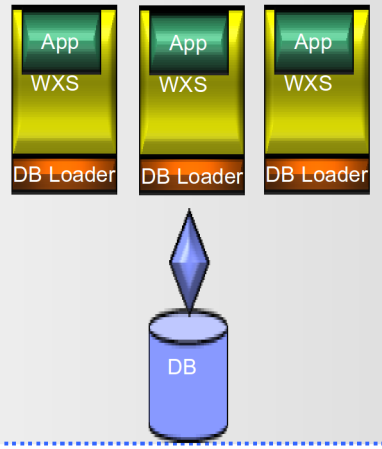
Changes may be written back in batches using write-behind

Database Cache as a System of Record

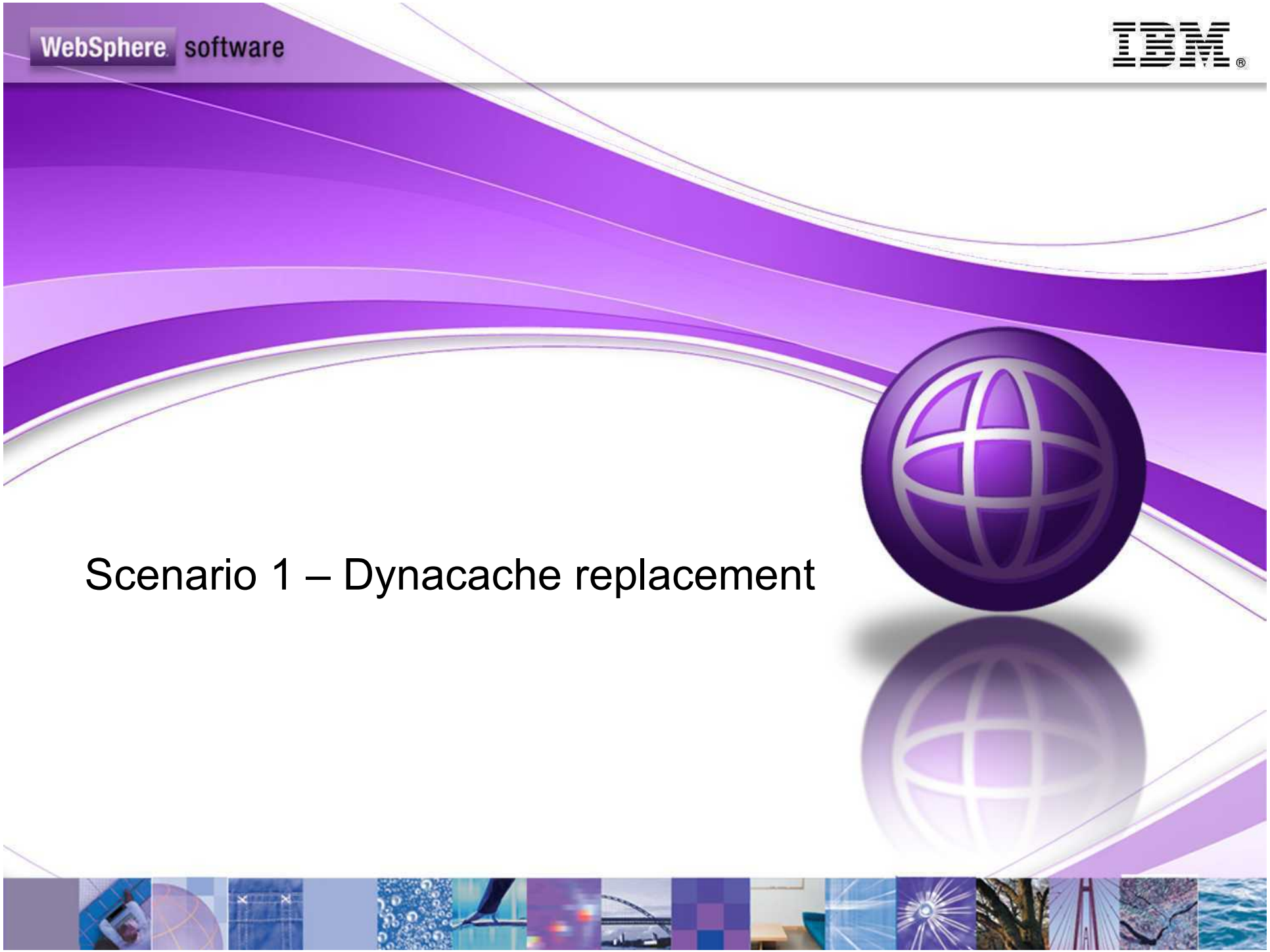


Load cache at startup and make updates to cache only
Batch updates to backend database using write behind
Use for high volume websites with exploding volumes of data

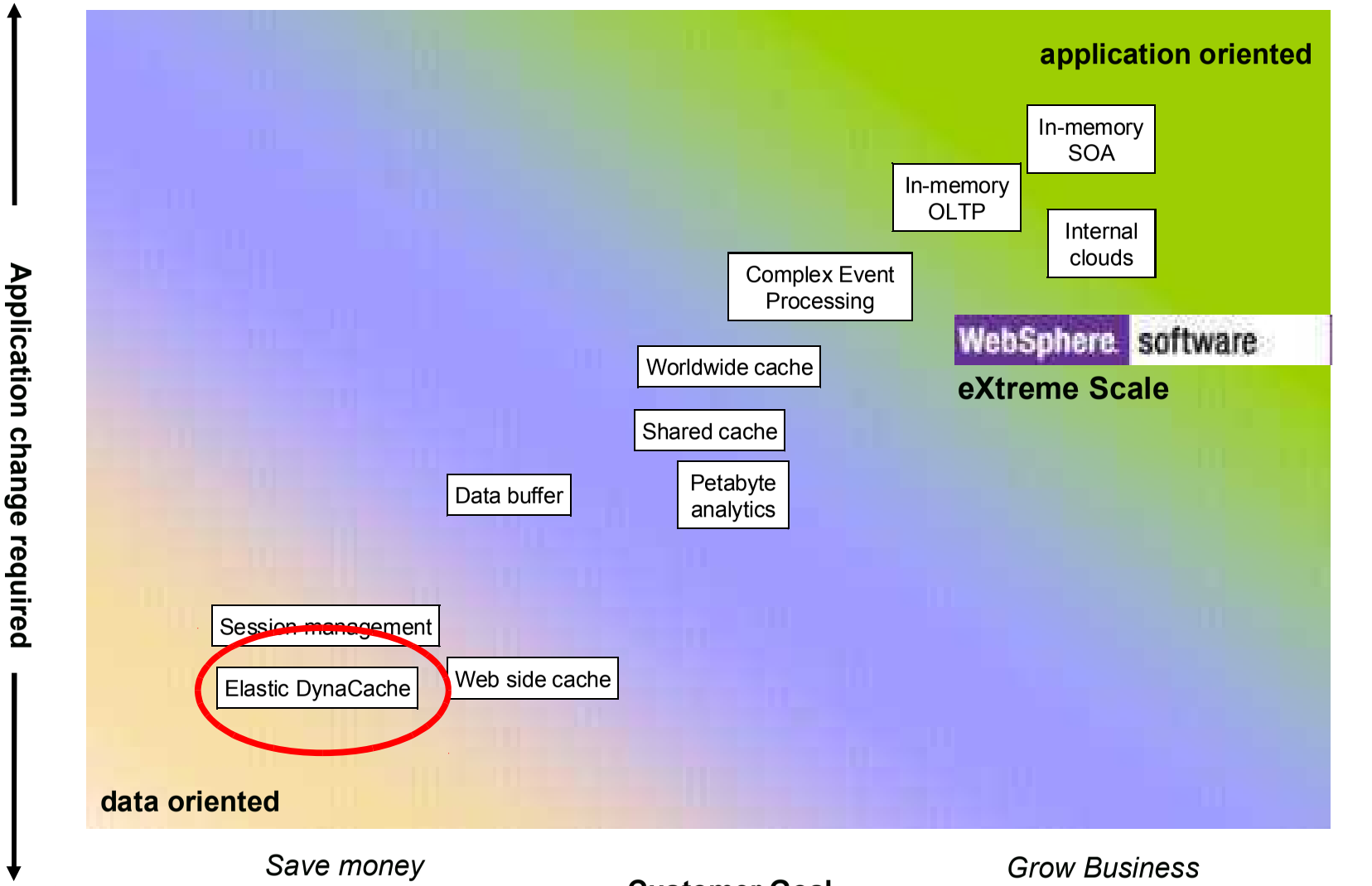
Database Cache Collocated with the Application



Eliminates network delay in accessing the cache
Optimal performance


WebSphere software
IBM
Scenario 1 – Dynacache replacement

What is WebSphere eXtreme Scale?



Dynacache - What's the challenge?

Dynacache brings a number of challenges to a large or growing deployment

- Each JVM has it's own cache store
- Each JVM has it's own disk offload

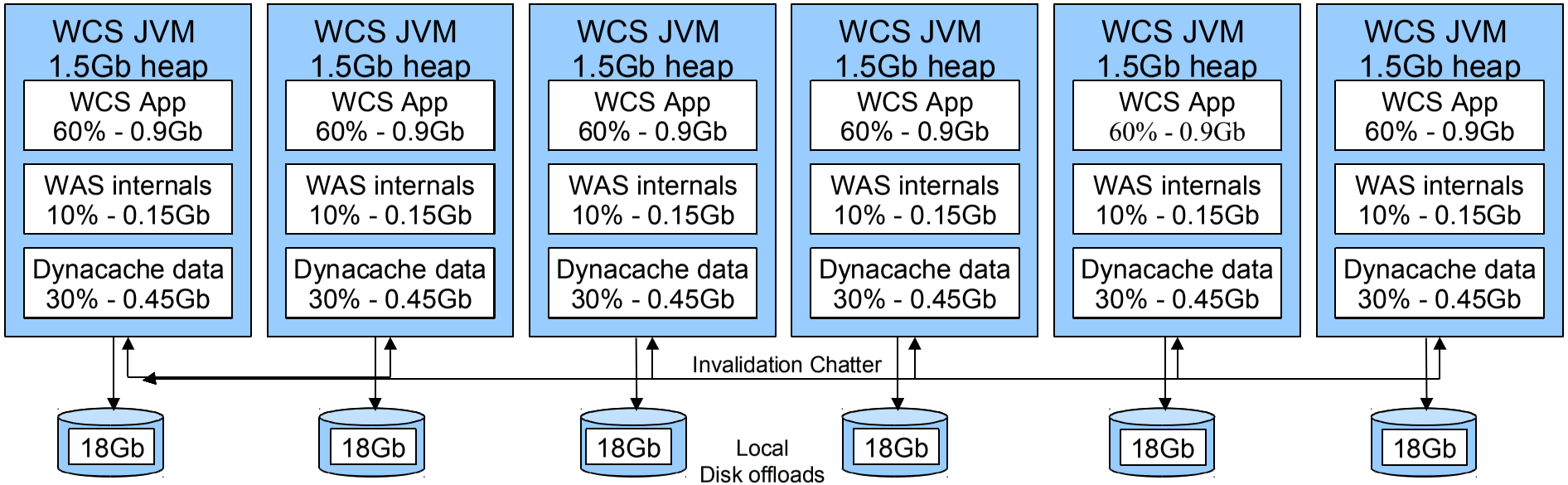
Cache is private to a JVM costing CPU, memory, network and disk

- Each cache entry is duplicated n times for n JVMs, wasting memory
- Cache change requires n invalidations wasting CPU and network bandwidth
- n copies of the cache entry means there is potential for a stale cache hit
- Disk offload for each JVM can get expensive and require high-powered hardware

Cache needs “warming up” on JVM (re)start

- Upon restart will experience slow responses
- Restart causes heavy disk I/O and heavy CPU for invalidating stale data on disk

Example - WebSphere Commerce with dynacache



Dynacache memory use example

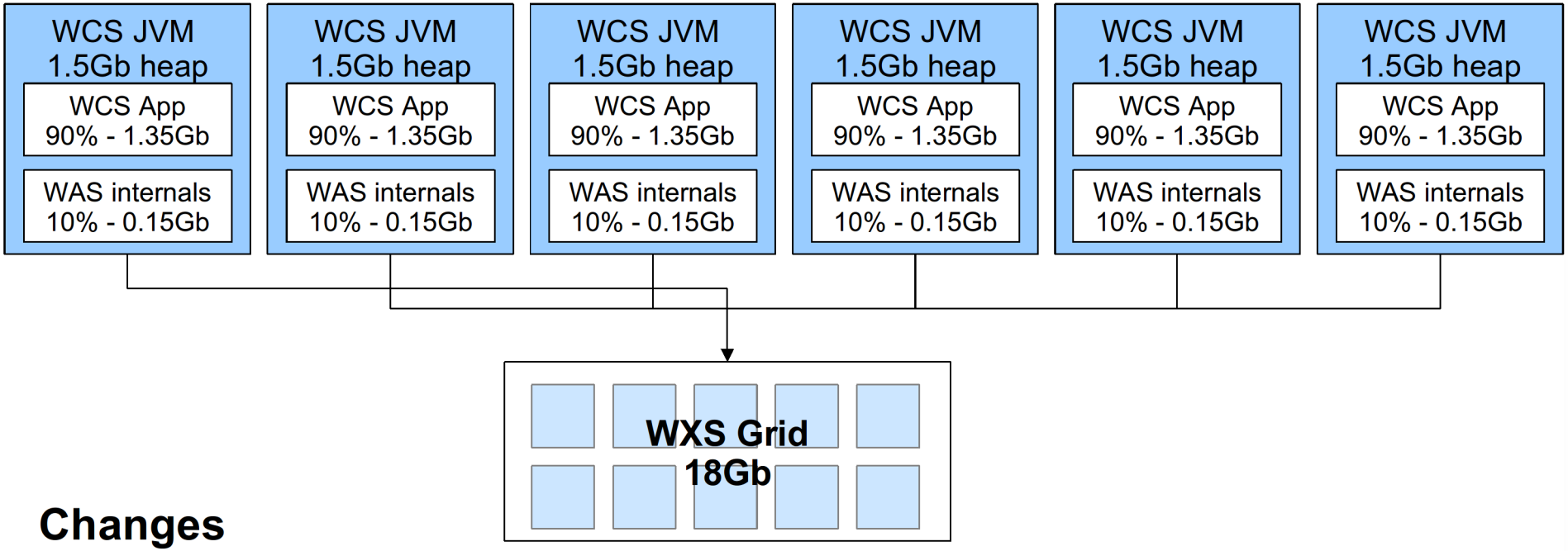
- Assume takes 30% of 1.5Gb heap = 450Mb cached data – this is the same data in each JVM
- For 6 JVMs, this is 2.7Gb of data needed just to represent 450Mb of cache
- Disk offload is 6 x 18Gb = 108Gb (*every instance has it's own disk offload*)
- Costs in performance
  - Garbage collection
  - Disk I/O
  - Dynacache invalidation

Dynacache replaced with WebSphere eXtreme Scale

- It is a shared cache stored independently from the application server
  - The application can run more efficiently; faster response times and greater throughput
  - No duplicate of cache in-memory reducing memory waste
  - No invalidation between caches reducing CPU and network of JVM “chatter”
  - When a cache entry is invalidated, it only needs recreating once and all JVMs can see it. This is a significant CPU saving on a large estate
  - No stale cache hits
  - No “warm up” needed or performance hit on JVM restart
- The cache can be very large; 100s of GBs without requiring disk
  - Saving money on disk hardware
  - Scale cache more easily (just add more JVMs)
  - Improve performance through storing data in memory

No code change needed, just plug eXtreme Scale in as dynacache provider

Example – WebSphere Commerce with eXtreme Scale



Changes

With WXS, we offload the dynacache data store to WXS "grid"

- WCS estate needs 25% less memory for dynacache - potentially reduce WCS estate
- Performance improvement from not needing disk I/O - around 25%
- Disk not needed – cost savings
- WCS throughput improvement through reduced "chatter" between JVMs and less GC overhead
- WXS can now provide much larger in-memory cache if desired by adding more JVMs (disk often constrained by size and contention limits
- Replica WXS JVM gives the cache resilience

What's the catch?

Some FAQs

But isn't a disk offload more resilient?

WebSphere eXtreme Scale can efficiently copy cache data to a configured number of replicas to provide in-memory availability

Can a WebSphere eXtreme Scale cache be big enough?

WebSphere eXtreme Scale would partition the cache data set, which is proven to scale to 1000s of JVMs with consistent and predictable response times. To increase cache size, we simply need to add JVMs. No extra configuration is needed

What's the catch?

Very little – the primary difference will be the network bandwidth from the Commerce tier to the WXS tier. But this is mitigated by the WXS compression of cache data over the wire (between 2.5 or 3:1)

Dynacache Replacement - Summary

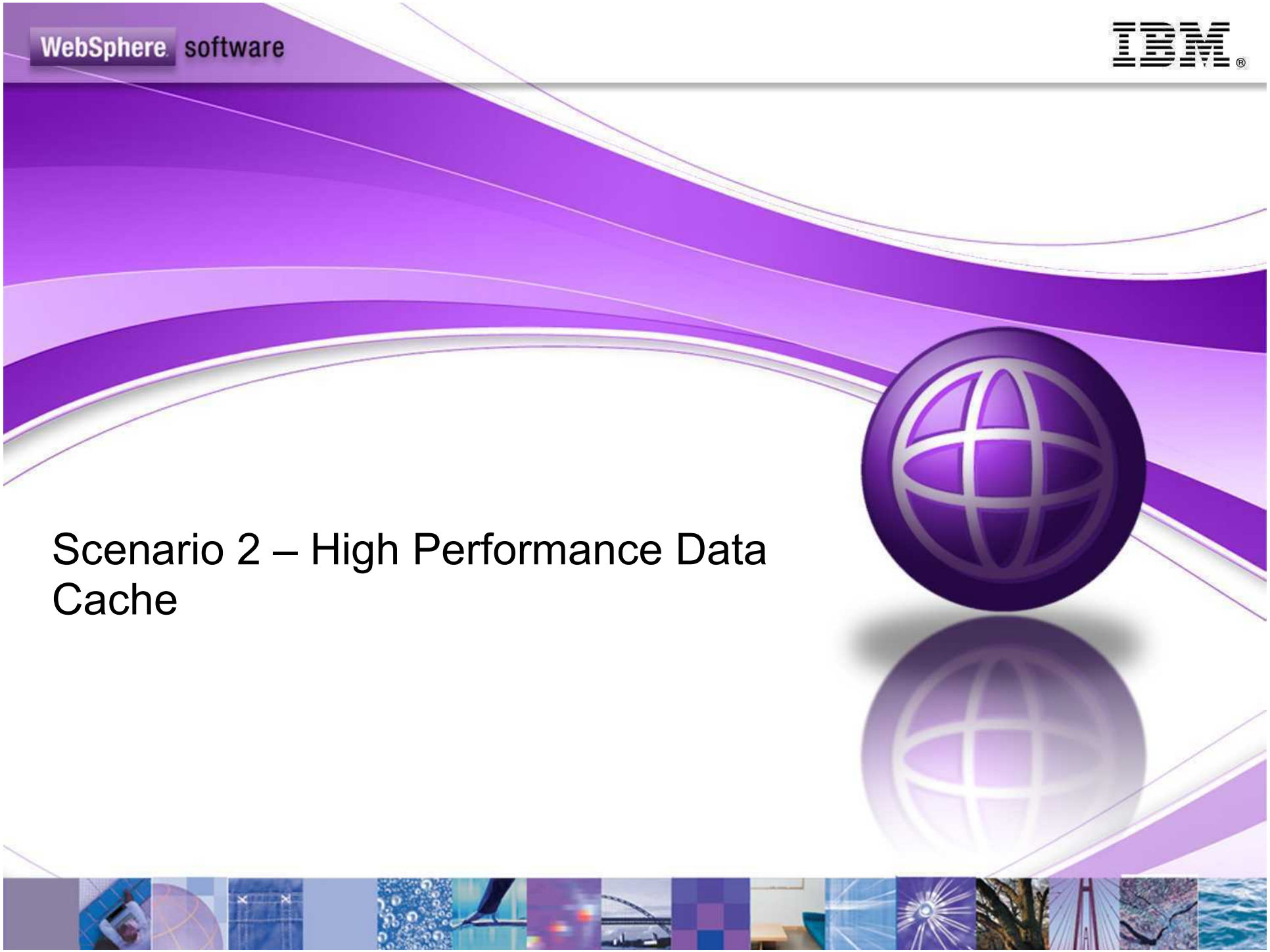
WebSphere eXtreme Scale brings elastic data grid technology to dynacache primarily bringing benefits on

- Application performance and scalability
- Disk hardware cost
- Reduction in Stack-product server estate (e.g. Commerce, Portal)

Further information

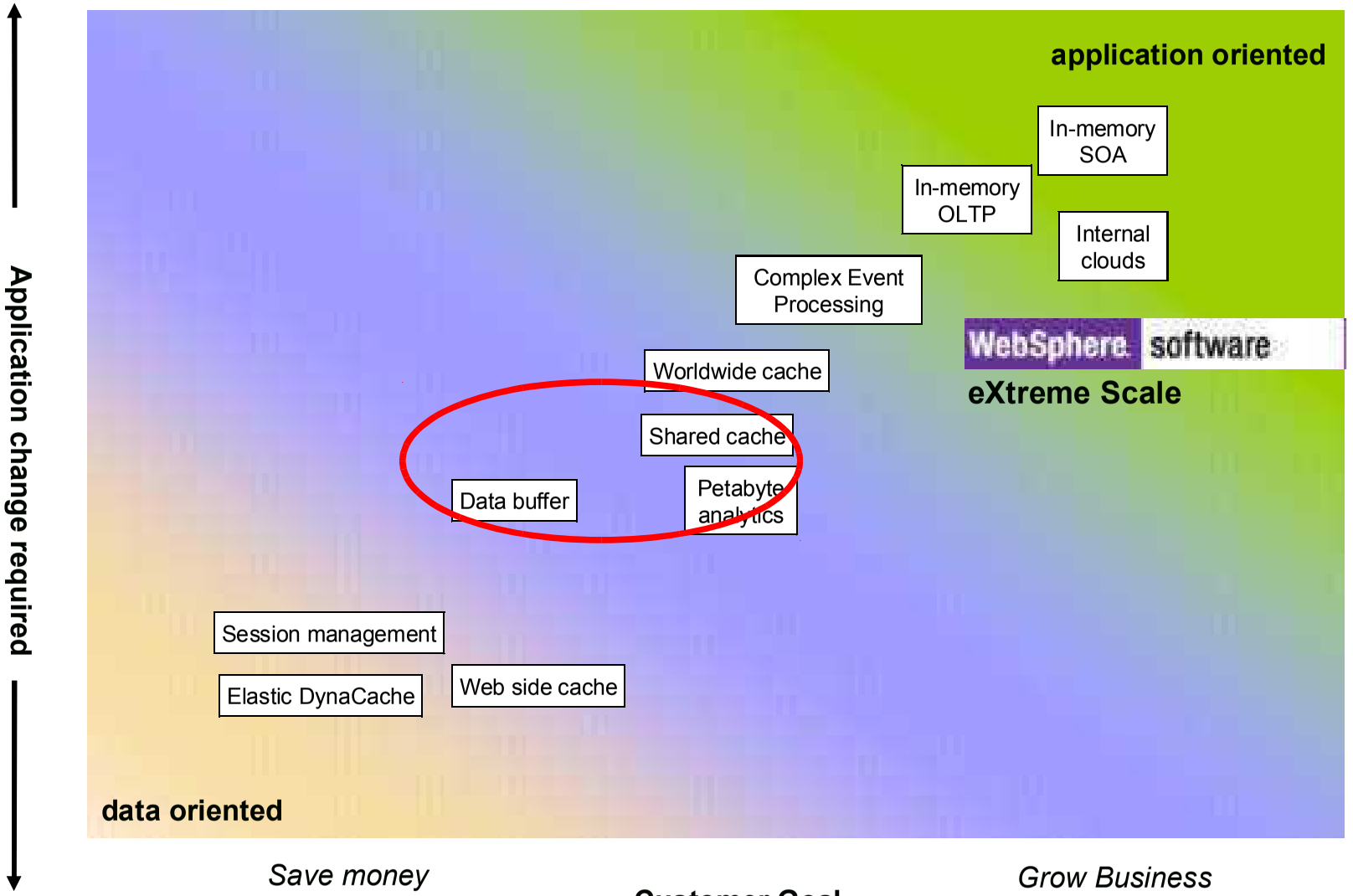
- These highlights are drawn from Billy Newport's Blog
  http://www.devwebsphere.com/devwebsphere/2010/05/websphere-commerce-server-now-supports-using-websphere-extreme-scale-for-page-fragment-caching.html

  http://www.devwebsphere.com/devwebsphere/2010/05/some-reasons-why-websphere-extreme-scale-lowers-commerce-server-cpu-requirements.html
- A load test comparison of dynacache with and without WebSphere eXtreme Scale
  http://www.devwebsphere.com/websphere_extreme_scale/2009/09/replacing-dynacache-disk-offload-with-dynacache-using-ibm-websphere-extreme-scale.html

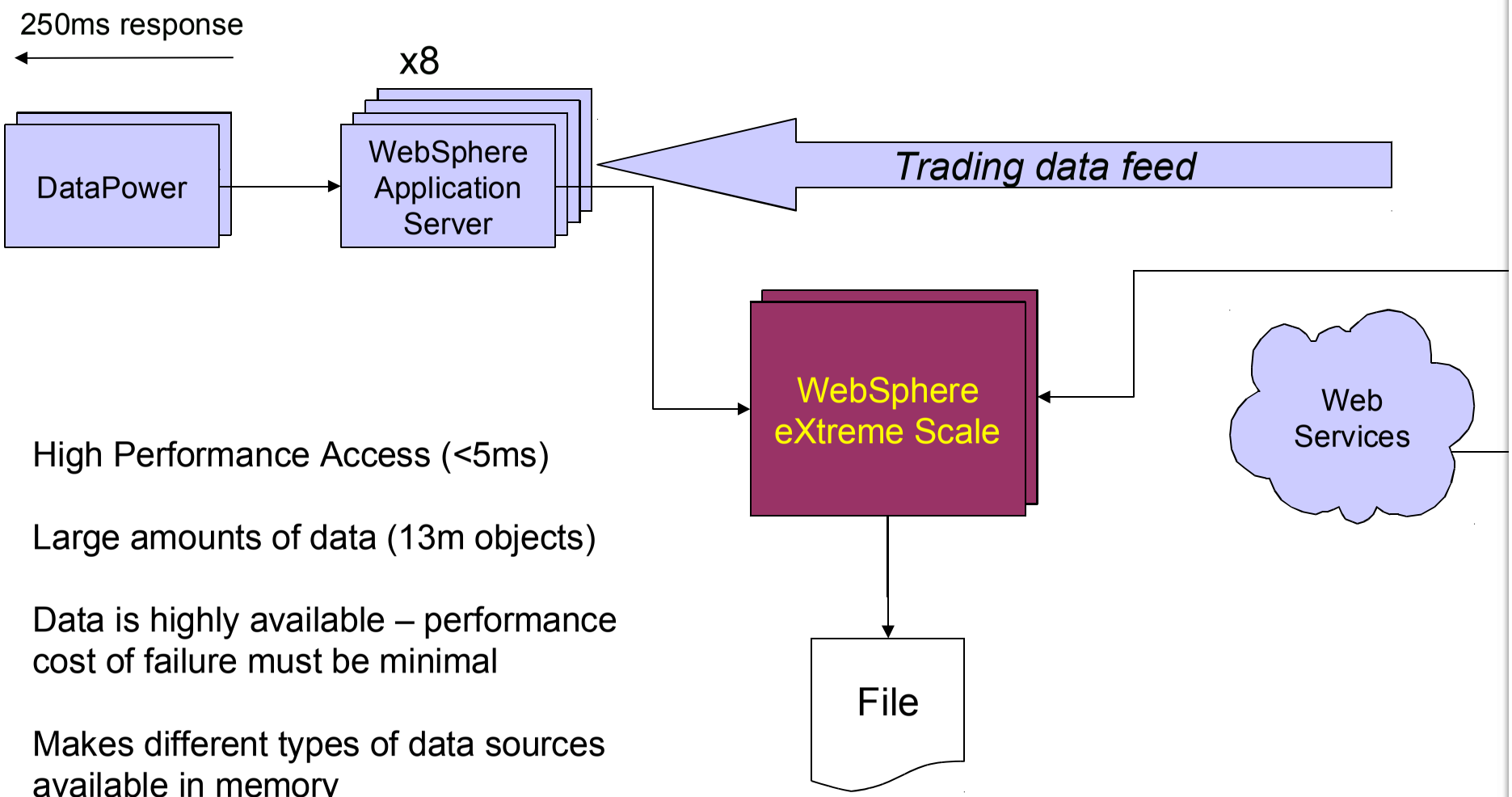


Scenario 2 – High Performance Data Cache

What is WebSphere eXtreme Scale?



Project Architecture Overview



High Performance Access (<5ms)

Large amounts of data (13m objects)

Data is highly available – performance cost of failure must be minimal

Makes different types of data sources available in memory

Why WebSphere eXtreme Scale?

- File loading – it's 200Mb, couldn't it be loaded everywhere? (With WXS it takes approx 3Gb!)
  - Only loaded once
  - Requirement to manage reloads
  - No “warm-up” time for new JVMs
  - Maintains very fast access (no txn mgmt for read-only)
  - Availability critical – Cache outage = site outage
- Web Services – just normal cache operation?
  - Stateless WS, so no affinity to WAS => cache duplication (similar scenario to dynacache)
  - Fine-grained invalidation control essential for permissions

1. Data partitioning is abstracted but not hidden

Principle: WXS does not support 2-phase commit (it is 7x slower). No transaction can interact with multiple partitions. Design accordingly

1 - Think carefully about queries

- WXS is highly optimised for direct access – map.get(key)
- Generalised queries need to scan entire grid – can perform well as all parallelised, but are a scalability limit
- Grid agents can run queries on relevant partitions

2 - Loading complexities

- Run on 1 partition or every partition? No choice with file
- Batching updates within transaction
- There are now great samples available (see notes)

2. Object map design

Principle: Store data “as close to the edge” as possible

Guidance

- Optimise for application use – how is it going to be used?
- Duplicate, don't normalise – use large entities with optional fields
- Optimise for queries - the fastest queries are cached queries
  - E.g. Full-text search on github
- Store sub-entities in same partition as primary
- Optimise for remote access
- Don't want to have to recycle grid, so schema design important
  - Generic “helper” agent interface can add flexibility

3. To near cache or not near cache

Principle: WXS allows for a subset of the grid to be stored in the client in a near cache for absolute maximum performance

Surely near cache is super fast = good? Super fast is µs instead of ms. Are ms good enough?

- Downside is that you *will* have stale data,

Guidance

1) Use a eviction policy on the near cache to cache. The trade off is better performance in exchange for data being potentially stale.

2) Rely on optimistic locking. Stops writes of stale data but can still have stale reads. An eviction policy can reduce the frequency of optimistic rollbacks.

3) Use JMS to push stale event notifications from the remote grid to clients. You still need to do (2) but the stale window is very short now

4) Don't use a near cache if 1,2 or 3 are not acceptable.

Or if you have to...

- No near cache, and evaluate NO_COPY option to remove txn semantics ← risky, only for read-only!

4. Sizing the grid

Principle: Don't assume that the amount of data to store = the required Java heap

- Use real data to size the grid
- Based on how many objects would fill 60% of the heap of a JVM
- Be aware of object overhead size
  - 13m x 200bytes = really quite alot!
- Replication - How many synchronous replicas and asynchronous replicas?
- Number of partitions
- Aim for 10 primaries per JVM – even distribution and low impact of failure
- Number of partitions recommended to be a prime number ← don't know why
- Set NumInitialContainers to be the total number of JVMs in the initial grid
- It can destroy start-up performance otherwise
- Remember to factor in the Loader as can take up significant space.

Methodology outlined in Billy's video - http://www.devwebsphere.com/devwebsphere/2009/02/websphere-extreme-scale-sizingconfiguration-presentation.html

Memory overhead - http://www.devwebsphere.com/devwebsphere/2009/10/memory-usage-in-ibm-websphere-extreme-scale.html

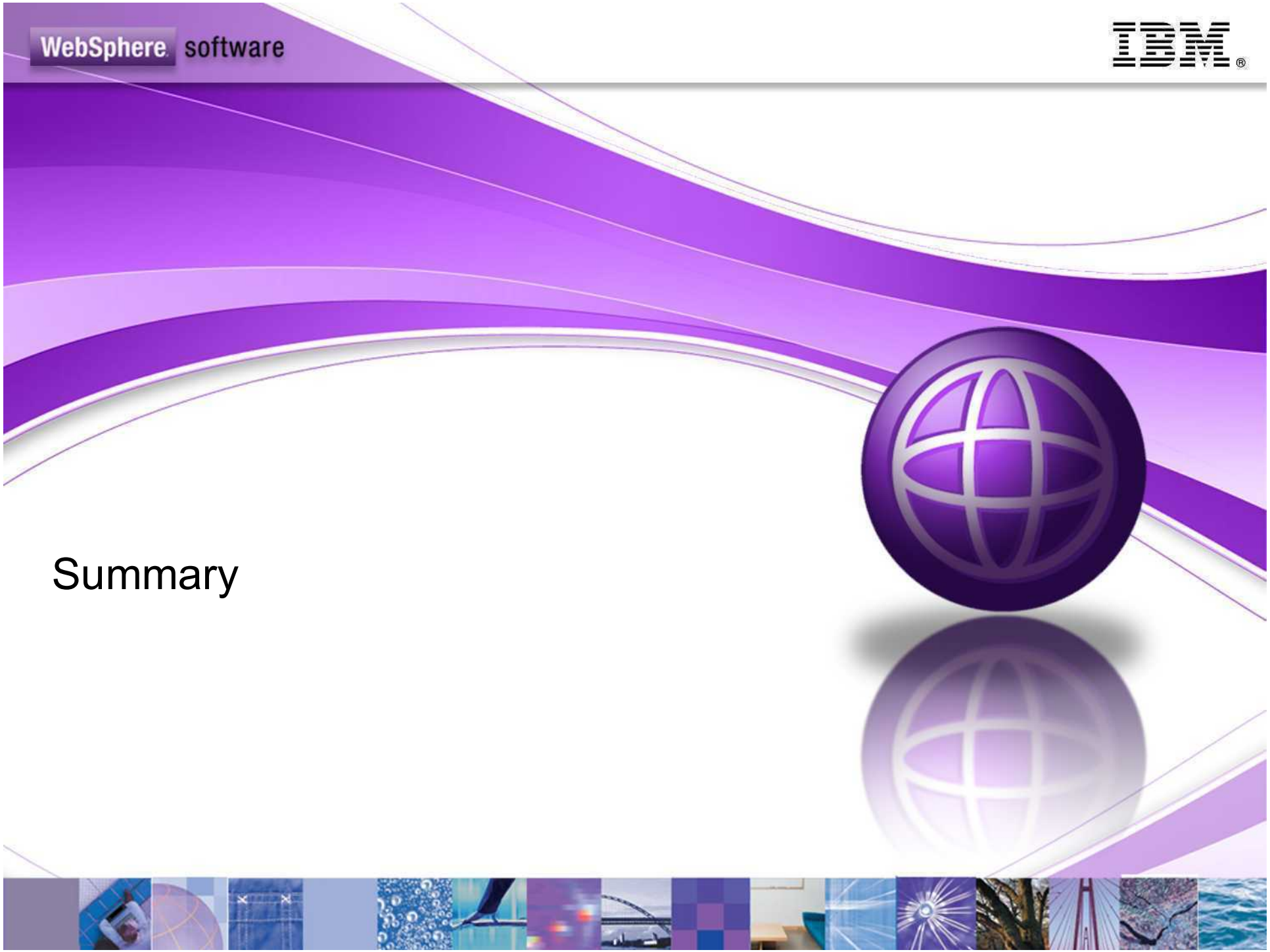
5. Managing the grid

Principle: WebSphere eXtreme Scale can run in WAS or as a standalone JVM

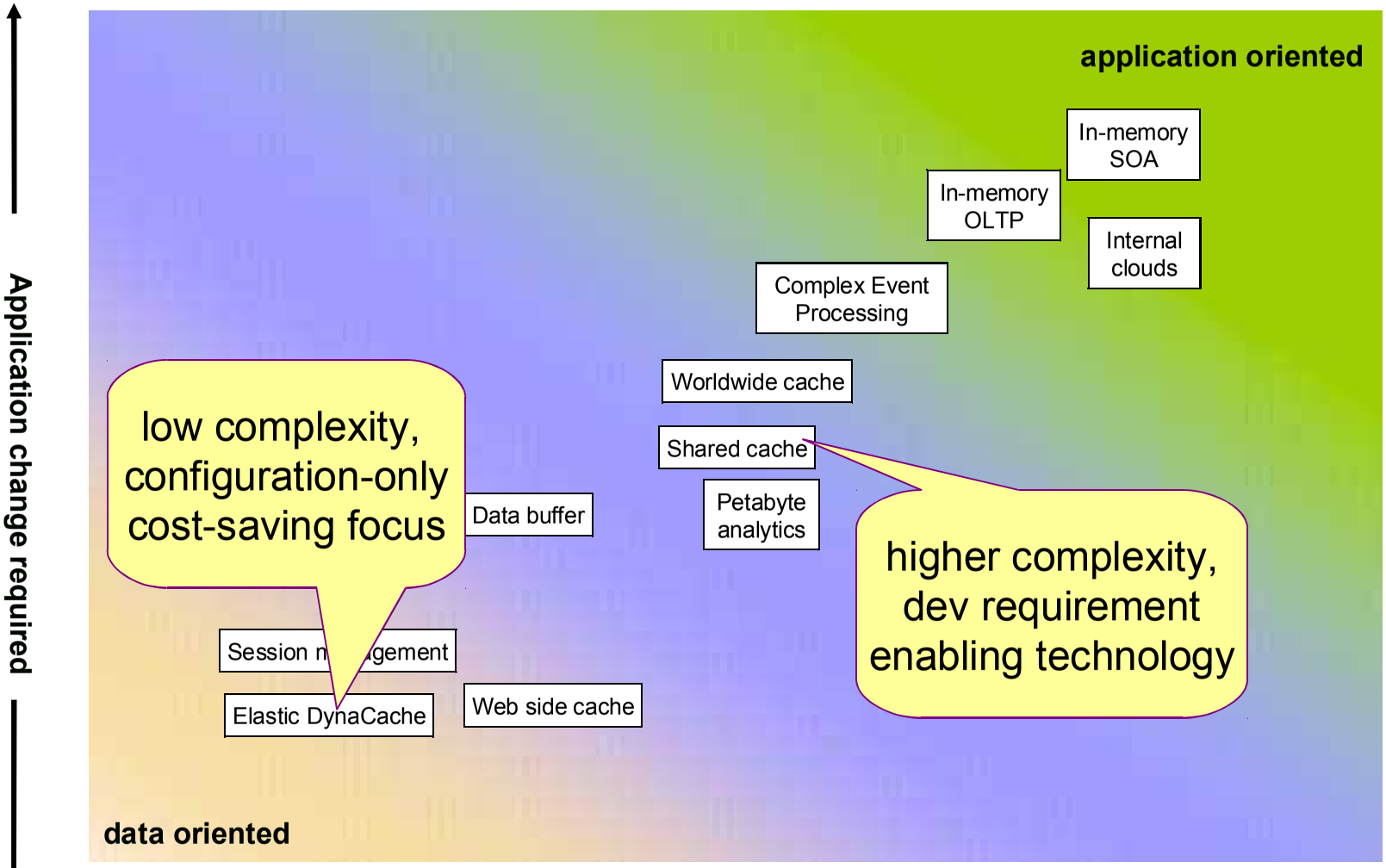
Standalone
- Cheaper license!
- Implication is manual management; sizing, operational control, manual administration
- Xsadmin tool is provided, but is described as a sample

Within WebSphere Application Server
- WXS administration is trivial – catalog servers and routing automatically provided
- Common administration infrastructure

Should the grid collocate with the application or not?
- Often start with collocation for reasons of “performance”. Probably won't materialise
- Performance hit of network is low and can be optimised
- Separate grid gives flexibility of management:
  - Application updates are separate from grid
  - Application and grid tiers can be sized very differently


WebSphere software
IBM
Summary

WebSphere eXtreme Scale – Experiences Summary



Summary

WebSphere eXtreme Scale is immensely powerful and supports a diverse range of deployment scenarios

- Dynacache – low complexity, cost-saving focus
- Application caching – higher complexity, enabling technology

Next steps: How do I get started?

Download WebSphere eXtreme Scale for free and build a trial application
http://www.ibm.com/developerworks/downloads/ws/wsdg/learn.html

Check out WebSphere eXtreme Scale in Amazon EC2
http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2721

Join the WXS community forum
http://www.ibm.com/developerworks/forums/forum.jspa?forumID=778&start=0

Getting started tutorials on developerWorks
http://www.ibm.com/developerworks/websphere/techjournal/0711_chambers/0711_chambers.html
http://www.ibm.com/developerworks/websphere/techjournal/0712_marshall/0712_marshall.html

Additional resources

Engage with the XTP Community: forums, samples, videos
http://www.ibm.com/developerworks/spaces/xtp

Read Billy Newport's blog on XTP
http://www.devwebsphere.com/

Watch Billy Newport answer your questions on YouTube
http://www.youtube.com/ibmextremescale

Redbook: WXS Users Guide
http://www.redbooks.ibm.com/redpieces/abstracts/sg247683.html?Open

WXS Version 7 InfoCenter
http://publib.boulder.ibm.com/infocenter/wxsinfo/v7r0/index.jsp