# IBM WebSphere eXtreme Scale

## (formerly WebSphere Extended Deployment Data Grid)

### An Introduction and Overview

Branham
BRANHAM GROUP INC.

## Contents

# Overview

## Introduction

Over the past few years, companies have seen a significant increase in transaction processing style applications to enable reliable, concurrent access to shared data servers. Simultaneously, organizations have also seen a significant increase in data growth, with an additional fivefold increase expected over the next three years. While some of this data may be used passively, rarely being accessed, the majority of corporate data is accessed through multiple channels. As customer demand for access to, and manipulation of, information stored on corporate servers increases, the demand to implement scalable and highly available extreme transaction processing (XTP) and data server solutions to meet these demands simultaneously increases.

Regardless of the channel, today's multi-tiered business critical applications pull data from some form of back-end servers (RDBMS, SOA data or application service, etc.), allowing interaction with the data, committing any changes. However, many of these applications need to convert data from back-end systems to an object form appropriate to the application. The resource cost associated with continually converting between back-end formats and these usable object formats is expensive from a CPU perspective. Additionally, many user sessions exhibit fetches to the same data resulting in redundant load on the back-end servers.

Client application interactions can involve multiple transactions in a single client session, operating on the same dataset. As the number of simultaneous client requests to a given server increases, the server can become severely overloaded. The movement towards Service Oriented Architectures (SOA) and Web 2.0 applications puts further demand on back-end software. This increase in processing requirements may become a cost issue through inopportune server upgrades and excessive hardware sprawl to meet demands but preemptively eliminate customer distain associated with increasing response times.

Consider the remarkable scalability challenges faced by financial services and travel reservation sites. Data volumes, and the need for consistent response times and transaction throughput, can bring the hardiest data server to its knees. A solution that helps alleviate load from these critical resources, while simultaneously deferring upgrade requirements, provides significant value to customers. IBM WebSphere eXtreme Scale is designed to address these key concerns.

## WebSphere eXtreme Scale v6.1

WebSphere eXtreme Scale (formerly Extended Deployment Data Grid) is available separately, or as part of the entire WebSphere Extended Deployment offering. It is an add on to existing application servers including WebSphere Application Server Network Deployment, WebSphere Application Server for z/OS, BEA WebLogic, JBoss, and more. It extends these environments to provide on demand dynamic responsiveness, high performance computing enhancements, simplified administration and management of complex distributed environments and a flexible approach to support both transactional and long running workloads. As a simple delta to existing application server installations, WebSphere eXtreme Scale does not require migration or restructuring of the existing environment and is integrated into the administration environment. WebSphere eXtreme Scale extends capabilities to dynamically accommodate variable business demands, support mixed workloads (transactional and long running) and mixed server environments, and reduce the complexity and cost of managing complex distributed environments.

WebSphere eXtreme Scale itself provides application acceleration and transaction processing for objects obtained from back-end systems or stored within the WebSphere eXtreme Scale. WebSphere eXtreme Scale is a software layer that sits behind an application providing the application its data from WebSphere eXtreme Scale itself or from back-end systems. The layer can be embedded in the same Java Virtual Machine as the application or can be a distributed layer where it makes sense for the application. WebSphere eXtreme Scale also provides a foundation for hosting XTP applications, distributed transaction processing applications that must provide high performance, availability and scalability requirements, ensuring fault tolerance and data integrity.

WebSphere eXtreme Scale provides the facilities to:

- Reduce database contention through caching database objects for multiple clients, removing the requirement for direct access.
- Decrease client/server response times by caching data subsets on clients.
- Integrate with event-driven Service Oriented Architectures (SOA) via Enterprise Service Bus solutions through a peer to peer / shared infrastructure.
- Support real-time decision making via real time data and event mining thanks to continuous queries on data.
- Create extremely scalable data grids and co-locate applications with in-memory data. Agents can be executed in parallel on all machines, aggregating the results.

**Application Platform Agnostic**

WebSphere eXtreme Scale, while included with WebSphere Extended Deployment, is designed as a standalone product that requires only J2SE 1.4 or above, and can easily be integrated with any Java application platform with full functionality regardless of the application platform(s) in use. Java EE applications can make use of WebSphere eXtreme Scale simply by including XML configuration files in a folder within a WAR or EJB module. WebSphere eXtreme Scale, on detecting the XML files, will automatically start the services for that module and stop them when the module is stopped. This makes it very easy for WebSphere or third party application servers, or Java SE complaint Java Virtual Machines to use WebSphere eXtreme Scale within their clusters.

**Cost Benefits**

IBM WebSphere eXtreme Scale is specifically designed to help applications lower the load they place on back-end servers, increase availability and scalability, and provide the facilities for extreme transaction processing. A software layer between an application and its back-end resources, WebSphere eXtreme Scale accelerates applications by caching objects obtained from the back-end, providing a customizable, pluggable data fabric framework that allows applications to share data across a wide range of application scenarios.

A WebSphere eXtreme Scale implementation can provide significant cost benefits including:

- Delaying or entirely eliminating the need to upgrade existing back-end systems (potentially saving significant dollars).
- Increasing transaction throughput and performance without expensive scaling of traditional transaction processing systems.
- Reducing back-end loads while simultaneously allowing support for larger application loads than previously anticipated for existing assets.
- Facilitating the means for real-time analysis such as fraud detection and risk management on more transactions than those created by traditional transaction processing systems.
- Improved client response times thus eliminating the customer and user frustration associated with timeouts while waiting for data from overloaded back-ends, during peak periods.
- Platform agnostic eliminates the potential costs associated with migration requirements to a specific platform from different commercial vendor or open source platforms.
- The ability to use commodity hardware.
- Simple implementation and maintenance requirements.

WebSphere eXtreme Scale can cache data that is accessed by an application session. It can also cache reference data commonly used by multiple sessions. This means that once an application obtains the data from a back-end initially, the application can retrieve the data from the WebSphere eXtreme Scale rather than the back-end on subsequent requests. As WebSphere eXtreme Scale caches the object representation of the data, it also significantly lowers the CPU cost of converting this data into object form once it's cached. Similarly, as demands on application architectures and technology infrastructures increase in terms of scalability and performance, XTP can provide significant benefits by reducing the costs associated with scaling traditional transaction systems and supporting the ability for continuous queries to support real-time decision making.

### High Availability and Scalability

Today's business-critical environments must deliver high availability to provide uninterruptible service to the enterprise. A failure that directly affects business-critical customers or business partners is unacceptable. Similarly, solutions must provide the necessary scalability to meet the most demanding transaction volumes during peak periods, while still providing adequate response times.

WebSphere eXtreme Scale provides high availability through a peered architecture that provides self-repair capabilities. The core group members monitor the health of other members of the group. As servers leave the grid due to failure or otherwise, WebSphere eXtreme Scale will recreate its data on surviving servers automatically.

Similarly, through its peering configuration and deployment model, WebSphere eXtreme Scale provides extreme linear scalability, potentially spanning thousands of Java Virtual Machines and extremely large data sets. Data cached in WebSphere eXtreme Scale can be striped across hundreds of nodes, and workloads from application Java Virtual Machines can be spread across WebSphere eXtreme Scale clusters aggregating results.

## Implementation

Customers are not required to conform to a WebSphere eXtreme Scale solution, but instead can implement WebSphere eXtreme Scale in a manner that best suites their IT infrastructure with minimal disruption. For example, WebSphere eXtreme Scale can be deployed as a local cache within the same Java Virtual Machine as an application, or can be part of a multi-tiered cache topology. It can be configured to conform to a rigid or fixed topology, or as a "plug-and-play" style dynamic topology.

### Multi-tier Caching

WebSphere eXtreme Scale supports being implemented as a local cache or through a multi-tiered cache topology. While implementing WebSphere eXtreme Scale as a local cache within the same Java Virtual Machine as an application provides the benefit of a very simple setup, it does inherit certain limitations including:

- ⇨ The size of the cache is constrained by the amount of memory in the application Java Virtual Machine.
- ⇨ A large cache can increase the garbage collector pauses experienced by the application.
- ⇨ Application start results in an empty cache which increases the load on back-end systems until the cache "warms up."

Implementing WebSphere eXtreme Scale in a multi-tiered cache topology uses a separate set of Java Virtual Machines, referred to as an WebSphere eXtreme Scale cluster, providing cache services to a large number of application Java Virtual Machines. WebSphere eXtreme Scale, when asked by applications for objects, will check if the object is in a local cache, if enabled, and then check the WebSphere eXtreme Scale cluster for the data. If the data isn't in either then the object can be pulled automatically from the back-end on behalf of the application.

A remote WebSphere eXtreme Scale provides many advantages over a simple local cache including:

- ⇨ The amount of objects that can be cached is potentially unlimited as an WebSphere eXtreme Scale cluster supports automatically spreading the data over the set of Java Virtual Machines in the cluster.

- ⇨ If configured, a local cache can be tuned to a much smaller size since the remote WebSphere eXtreme Scale provides caching for the objects not cached locally. This can dramatically reduce garbage collector pauses seen by applications without increasing load on the back-end systems.
- ⇨ Once an object is fetched by any application Java Virtual Machine, it becomes available to every other application Java Virtual Machine. A local cache does not exhibit this behavior.
- ⇨ If application data was previously placed in the remote WebSphere eXtreme Scale, load spikes on back-end systems are reduced as application Java Virtual Machines start up and local caches "warm up."
- ⇨ Replication provides WebSphere eXtreme Scale failover. Specifically, if a Java Virtual Machine in the remote WebSphere eXtreme Scale cluster fails, the data cached in the failed node has been replicated to another Java Virtual Machine within the cluster, circumventing unpredicted load spikes on back-end systems that might occur in a non replicated environment.
- ⇨ More modern Java Virtual Machines can be used for the remote cache (64-bit J2SE 5, IBM J9). A local cache is limited to the Java Virtual Machine used by the application platform. This is usually fixed and cannot be changed easily. This allows an WebSphere eXtreme Scale cluster to leverage the performance or address space improvements offered by the latest Java Virtual Machine technology, without forcing application platforms to be migrated.
- ⇨ An WebSphere eXtreme Scale cluster can run on lower cost commodity platforms. For example, an application running on a mainframe could run against an WebSphere eXtreme Scale cluster running on zLinux LPARs (simultaneously capitalizing on the cost savings of IFL and zAAP processors) or Intel blades.
- ⇨ Easier horizontal scaling for remote WebSphere eXtreme Scale servers. The workloads from the application Java Virtual Machines can be spread over replicas in the WebSphere eXtreme Scale cluster and aggregate the results. This allows the load supported by the WebSphere eXtreme Scale to be linearly scalable, supporting hundreds of Java Virtual Machines and extremely large data sets.

## Static or Dynamic

New for WebSphere eXtreme Scale v6.1 is support for a "plug-and-play" type of dynamic configuration. This new dynamic deployment topology provides flexibility in adding resources to an WebSphere eXtreme Scale implementation without tearing down the entire cache. This is accomplished by using a Catalog Service which automatically manages the assignment of WebSphere eXtreme Scale servers. The catalog service provides a peering configuration and deployment model that allows WebSphere eXtreme Scale to perform extreme scalability, potentially spanning thousands of Java Virtual Machines and allowing for extremely large datasets.

WebSphere eXtreme Scale's catalog service is a replicated service running on at least two machines. It acts as a fault tolerant controller for the grid. It lays out data (primaries and replicas) across the set of online servers in a balanced fashion. If servers are started or stopped, then it automatically reacts to this by dynamically moving data across the grid to ensure that failed replica servers are replaced, and to spread the data out across all servers in the grid for maximum performance.

The catalog service uses well known endpoints so that clients and new Java Virtual Machines can find it. These endpoints are the only static end points in the component alleviating a requirement for multicast. All Java Virtual Machines holding data automatically find unused ports for their endpoints and then the catalog service allows peers/clients to find each others ports. This avoids the need to pre-configure individual Java Virtual Machines with ports, and allows multiple Java Virtual Machines to be easily started on a single machine without worrying about port conflicts. This capability can provide significant total cost of ownership savings by eliminating configuration requirements for each Java Virtual Machine and through the automatic organization of large grids even on complex topologies.

Clients connect to a catalog service, retrieve a description of the WebSphere eXtreme Scale server topology, and then interact with each WebSphere eXtreme Scale server directly as needed. As such, the catalog service is not a factor in the runtime performance of the grid and should not be considered a bottleneck. When the server topology changes due to the addition of new servers, or due to the failure of others, the client is automatically routed to the appropriate server that is hosting the data.

If a Java Virtual Machine fails, then replicas on survivors are promoted to be primaries and any lost replicas are repaired by adding new ones if possible on surviving Java Virtual Machines. This means scaling out an WebSphere eXtreme Scale is simply a matter of starting additional Java Virtual Machines and data is automatically migrated to them once they register with the catalog service.

Despite the obvious benefits of a dynamic implementation, static deployment is still available for those customers who prefer this option. Although a static deployment topology is fixed, it does not require a catalog service and allows more controlled placement of the WebSphere eXtreme Scale data. The location of each primary, replica and partition is explicitly defined in the cluster deployment XML. However, the dynamic deployment topology will continue to be enhanced to support a variety of deployment scenarios and should be strongly considered over the static deployment topology.

### Disaster Recovery

In today's enterprises, high availability is a requirement. WebSphere eXtreme Scale provides the facilities to control how fault tolerance and disaster recovery should be addressed through Policy Based replication. If a primary server fails, then a replica can be promoted to the primary position and additional replicas will automatically be created according to quality of service specifications. Specifically, customers can control how replication is performed. For example, each primary node can have synchronous and/or asynchronous replicas. They can be configured for a single synchronous replica, a single asynchronous replica, or one of each at the same time. Similarly, a topology aware replication can be configured to avoid placing primaries and replicas on machines within common failure nodes.

What this allows customers to do is achieve high availability configurations for local failover and larger scale disaster recovery. For example, a large datacenter can perform synchronous replication locally and asynchronous replication to a remote location. This provides failover for local servers while also allowing the remote data center to take over if connectivity to the primary facilities experiences a large scale failure.

## Security

WebSphere eXtreme Scale provides distributed caching of data, much of which is often sensitive information that must be protected at all times. Therefore, it is necessary to provide security mechanisms for authentication and authorization to cached data. WebSphere eXtreme Scale relies on the environment, either the application server's or application's, for authentication. For example, when WebSphere eXtreme Scale is used in a WebSphere Application Server or WebSphere WebSphere eXtreme Scale environment, the applications can use the WebSphere Application Server security authentication mechanism. When an WebSphere eXtreme Scale is running on a Java Platform, Standard Edition (Java SE) environment, the application can manage authentications by itself using JAAS authentication or other authentication mechanisms. For improved performance, WebSphere eXtreme Scale also supports caching the permission checking results.

### Integration

Through an extensible architecture, WebSphere eXtreme Scale can be integrated into any security environment. This integration allows WebSphere eXtreme Scale to use a customer's authentication and authorization infrastructure to control access to data held within the WebSphere eXtreme Scale. WebSphere eXtreme Scale has been successfully integrated with authentication and authorization infrastructures including:

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- WebSphere Application Server Security
- Tivoli Access Manager
- Java Authentication and Authorization Service (JAAS)

WebSphere eXtreme Scale uses the security provider for authenticating clients to servers and for authorizing clients to access certain WebSphere eXtreme Scale artifacts or to specify what can be done with WebSphere eXtreme Scale artifacts. WebSphere eXtreme Scale includes the following types of authorizations:

- ⇨ Map authorization: Clients or groups can be authorized to perform insert, read, update, evict or delete operations on Maps.
- ⇨ WebSphere eXtreme Scale authorization: Clients or groups can be authorized to perform object or entity queries and stream queries on object grids.
- ⇨ DataGrid agent authorization: Clients or groups can be authorized to allow DataGrid agents to be deployed to an WebSphere eXtreme Scale.
- ⇨ Server side map authorization: Clients or groups can be authorized to replicate a server map to client side or create a dynamic index to the server map.
- ⇨ Administration authorization: Clients or groups can be authorized to perform administration tasks.

### Policies

Similar to many other systems, WebSphere eXtreme Scale adopts a permission-based authorization mechanism. WebSphere eXtreme Scale has different permission categories represented by different permission classes including read, write, insert, remove, invalidate, and all permissions. Security policies can be created for use with WebSphere eXtreme Scale allowing CIOs to create policies without relying on programmatic API implementations (although they are still available). This means that security policies can easily be changed without a requirement to alter applications, significantly reducing complexity for application lifecycle management.

## Development

At its most basic definition, WebSphere eXtreme Scale is a cache and an in-memory repository for objects. However, WebSphere eXtreme Scale provides an extensible, transactional object caching framework. In its simplest form, customers may simply look to WebSphere eXtreme Scale as a means to improve data performance and throughput for their applications. This is the traditional data caching scenario. WebSphere eXtreme Scale however, enhances even this scenario by offering rollback capabilities. The system programming model provides several additional features and extension points for the WebSphere eXtreme Scale.

### Invasive vs. Non-invasive

Developers can integrate caching into their applications as they are being developed. This invasive method of implementing caching for applications can provide some significant performance benefits but cannot easily be removed or modified. Many customers may not have expert developers who can build applications with built-in caching. As an alternative, WebSphere eXtreme Scale provides a more non-invasive approach providing the ability to cache calls to the application or uses a mechanism to plug caches into applications that may have already been developed and deployed.

Service oriented Architectures (SOA) and technologies such as J2EE/CORBA allow remote calls to data services. WebSphere eXtreme Scale can be attached as a filter to the client portion of these technologies and provide simple caching for the results of the remote calls. This is a very cost effective way of reducing the load on an application/system that provides data services to an organization and can be applied even to legacy applications by integrating WebSphere eXtreme Scale with the standard clients used by Java applications. This is currently the most common scenario used by IBM customers and is a very cost effective way to introduce caching to an organization.

### Java

Using the industry standard Java platform, a WebSphere eXtreme Scale application can be configured to run in either a Java Platform Standard Edition (Java SE) or a Java Platform Enterprise Edition (Java EE) environment. WebSphere eXtreme Scale allows customers to use the Java programming environment that best suits their needs from the extensible, Eclipse open-source based development framework, to the command line Java environment. As customers become more familiar with WebSphere eXtreme Scale, they can also incorporate WebSphere eXtreme Scale into enterprise beans and servlets.

IBM also includes Java EE best practices including JAR/WAR-based deployment schemes, JCache-compliant pluggable mapping concepts, and XML-based configuration which allows for extensive customization at deploy-time. WebSphere eXtreme Scale is also designed to be configured via JavaBeans to make it easy to integrate into Spring-based applications, the leading Java/J2EE application framework, further reducing development effort and costs while improving test coverage and quality.

### POJO simplicity

Many cache products use Map-based APIs to store data in key/value pairs. Although this approach is successful, it does provide limitations including:

- ⇨ The cache must use reflection to extract data from the objects in the cache, which has performance implications.
- ⇨ Two applications cannot share a cache if they use different objects for the same data.
- ⇨ Data evolution is not possible (i.e. an attribute cannot be added to a cached Java object without significant effort).
- ⇨ Working with graphs of objects is cumbersome. The application must store artificial references between objects and manually join them together.

WebSphere eXtreme Scale does continue to offer Map-based APIs for its programming model. However, WebSphere eXtreme Scale also provides a more streamlined, lightweight approach, through its EntityManager to significantly simplify interaction with the WebSphere eXtreme Scale cache. This set of APIs follows the Plain Old Java Object (POJO) style of programming that is adopted by most frameworks. Specifically, EntityManager uses the existing Map-based infrastructure, automatically decomposing the entity relationships to the appropriate key references when storing the entities in the WebSphere eXtreme Scale. It converts entity objects to and from tuples (an array of primitive attributes) before storing or reading them from the Map. An entity object is transformed into a key tuple and a value tuple, which are then stored as key/value pairs. When an application is finished with the objects, changes are detected and written back to the grid.

This allows customers to define a schema for what is stored in the grid and then different applications using the same or different Java objects can share the data in the grid. While this may be somewhat slower than using the Map APIs directly, it provides for a significantly simplified programming model when dealing with graphs of objects as compared to older JCache or Map-based APIs. Annotations or XML files can then be used to specify how the mapping of data to POJOs is completed.

## Administration

WebSphere eXtreme Scale provides a system management infrastructure with which customers can monitor and administer static WebSphere eXtreme Scale topology deployment environments. WebSphere eXtreme Scale can integrate with a WebSphere Extended Deployment console or any industry standard Java Management Extensions (JMX) console. This allows dashboards to be built to monitor how WebSphere eXtreme Scale is performing.

### Management

WebSphere eXtreme Scale provides Managed Beans (MBeans) allowing customers to monitor and administer a static topology. The system management architecture is a three tiered approach where a user client connects to the management gateway server, which makes a WebSphere eXtreme Scale client connection to a WebSphere eXtreme Scale cluster.

There are five types of MBeans that exist in a WebSphere eXtreme Scale environment. Each MBean refers to a specific entity, such as a map, object grid, server, replication group, or replication group member.

- ⇨ The MapMBean MBean is used to monitor the statistics of each map that is defined for the cluster such as count, hit rate, and batch update time.
- ⇨ The WebSphere eXtreme ScaleMBean MBean is used to monitor the statistics for all the maps in each WebSphere eXtreme Scale that is defined for the cluster.
- ⇨ The ServerMBean MBean is used to perform operations on servers in the cluster.
- ⇨ The ReplicationGroupMBean MBean is used to monitor the status for all the replication group members that are associated with a specific replication group, including the primary server and up to ten replica servers.
- ⇨ The ReplicationGroupMemberMBean MBean is used to monitor the status of a replication group member (primary or replica members), and the replica weight ratio - a quantification of how close the replica maps are to being synchronized with the primary maps. The higher the ratio, the closer a replica is to having the current primary information.

The WebSphere eXtreme Scale JMX and MBean administration model was created to take advantage of the various JMX consoles that are available for administering JMX environments. Customers put together dashboards using the JMX console of choice. Consoles can be attached to the MBeans running on the management gateway Java virtual machine and dashboards can be assembled using these MBeans. Consoles offer graphical histories or charts of numerical and string values. The management gateway is not needed for dynamic WebSphere eXtreme Scale deployment topologies.

## Summary

For most customers, back-end systems represent a significant investment, and they are continually looking for opportunities to leverage them. This increase in utilization however, creates new challenges related to increasing load on these systems. WebSphere eXtreme Scale provides technology to significantly lower the load on existing back-end systems through transaction and caching capabilities, and provides this advantage even when application servers are started, which would normally empty local caches.

A WebSphere eXtreme Scale implementation provides caching services for applications requesting data from back-end servers and provides facilities for XTP. This significantly reduces their loads and simultaneously improves application performance. WebSphere eXtreme Scale can be used for Enterprise Service Bus products that offer access to the back-end through a Java API, or used to cache previous Service Oriented Architecture (SOA) calls eliminating SOA calls where they may be a duplicate of a previous SOA call (same service with same arguments).

Through WebSphere eXtreme Scale, customers can postpone or virtually eliminate costs associated with upgrading more expensive, heavily loaded back-end database and transactional systems, while meeting the high availability and scalability requirements for today’s environments. WebSphere eXtreme Scale provides low maintenance and implementation costs through a dynamic “plug-and-play” topology, while providing extreme, yet linear scalability on commodity hardware. WebSphere eXtreme Scale also helps ease the development process through non-invasive methods, POJO programming styles, and increasing integration with the Spring Framework. WebSphere eXtreme Scale provides enterprise customers with significant value and low total cost of ownership.