Configuring and Implementing Dynamic Caching in WebSphere Application Server 6.1

Geoff Tindall
Agenda

- Dynamic Cache Equals Performance
- Fragment Caching
- Enabling Dynamic Cache
- Caching Rules in CacheSpec.xml
- Monitoring the Cache
- CacheID
- Dependency IDs
- Disk Offload
- Cache Invalidation
- Dynamic Cache Performance Tips
- Troubleshooting
- Further Dynamic Cache Technologies
- Reference Material
Dynamic Cache Equals Performance

- Dynamic Cache is part of the IBM solution for improving the performance of Java 2 Platform, Enterprise Edition (J2EE™) applications running within WebSphere Application Server.

- Dynamic Cache supports caching of Java™ servlets, JavaServer Pages™ (JSP™), WebSphere command objects, Web services objects, and Java objects.

- This presentation describes the features and configuration steps of a dynamic cache environment for servlets and JSPs.
Dynamic Cache Equals Performance

- The concept of caching static information in a browser, proxy or a webserver provides an effective way to reduce network and processing requirements.

- A larger overhead of many web applications is related to the serving, not of static content, but of dynamic content based on user input and retrieval of data from backend resources such as databases.

- IBM’s Dynamic Cache solution allows the customizable caching of dynamic content which can provide a major performance boost for high volume web sites.
Enabling Dynamic Cache

Integrated Solutions Console - Microsoft Internet Explorer

Application servers > server1 > Web container
Click to configure the Web container

Configuration

General Properties

- Default virtual host:
  default_host

- Enable servlet caching

Additional Properties

- Custom Properties
- Web container transport chains
- Session management

Apply OK Reset Cancel
Enabling Dynamic Cache

- Dynamic cache is enabled by default on V6.1 WebSphere AppServers.
- To enable Servlet and JSP caching from the admin console, navigate to the web container configuration panel - Servers – Application Servers - `<select server>` - webcontainer settings – webcontainer
- Check ‘Enable servlet caching’
- Save and restart the Application Server to put the changes into effect.
cachespec.xml

- Now servlet caching is enabled, if is necessary to define which dynamic content will be cached and the rules by which it will be cached or invalidated.

- For this presentation, servlet and JSP caching will be policy based using the cachespec.xml file.

- The preferred location for the cachespec.xml file is within the web applications WEB-INF folder.
cachespec.xml

```xml
<cache>
    <cache-entry>
        <class>servlet</class>
        <name>/Time.jsp</name>
        <cache-id>
            <timeout>30</timeout>
        </cache-id>
    </cache-entry>
</cache>
```
cachespec.xml

- Consider our example of a simple JSP page (Time.jsp) which returns the system time. While the cost in retrieving and displaying the time is minimal, it allows us to demonstrate, in a basic way, the configuration and set up of Dynamic Cache.

- The first request for Time.jsp will result in a cache MISS. Full processing takes place for Time.jsp. The response is returned to the browser but also loaded into cache.

- Subsequent requests for Time.jsp result in a cache HIT until the cached copy of Time.jsp is invalidated.
Monitoring the Cache

- Once Dynamic caching is configured, and dynamic content is cached, the content of the cache can be examined and monitored using the cachemonitor.

- An enterprise application called CacheMonitor is provided with WebSphere Application server.

- CacheMonitor is deployed to WebSphere in the same manner as other enterprise applications. It is available in the folder `<WebSphere Install Root>/installableApps` and called CacheMonitor.ear.
Monitoring the Cache

- Our example, Time.jsp is accessed 3 times.
- From the CacheMonitor statistics we see 1 miss (the first request when content is not yet cached) and 2 hits (content is now pulled from cache).
Monitoring the Cache

- Cache contents can also be displayed.
- In our example, the cacheID is

/referenceweblTime.jsp:requestType=GET
CacheID

- To cache a JSP or element, the application server must know how to generate a unique ID or cacheID for different invocations of the JSP.
- For each cache entry there will be a unique cacheID.
- In our example, the cacheID is
  
  /referenceWEB/Time.jsp:requestType=GET

- This particular cacheID simply comprises of the URI and the request type.
CacheID

```xml
<cache>
  <cache-entry>
    <class>javax.servlet</class>
    <name>/Time.jsp</name>
    <property name="ignore-get-post">true</property>
    <cache-id>
      <component id="format" type="parameter">
        <required>false</required>
      </component>
      <timeout>30</timeout>
    </cache-id>
  </cache-entry>
</cache>
```
Cache ID

- Commonly, a cacheID will also contain a parameter.
- Our Time.jsp has the ability to show times in different formats based on a query string parameter called format.
Cache ID

```xml
<cache-entry>
  <class>servlet</class>
  <name>/Time.jsp</name>
  <property name="ignore-get-post">true</property>
  <cache-id>
    <timeout>90</timeout>
    <component id="jsessionid" type="sessionID">
      <required>true</required>
    </component>
  </cache-id>
</cache-entry>
```
Cache ID

- A cacheID can be comprised of the following
  - Request parameters and attributes
  - The URI used to invoke the servlet or JSP
  - Session information
  - Cookies
  - Pathinfo and servlet path
  - Http header and Request method
  - Servlet/JSP result caching can be used to cache both whole pages or fragments.
Fragment Caching

The content of A.jsp is composed of its own content plus the content of the 3 other included jsp\(s\) (fragments) B.jsp, C.jsp and D.jsp
Fragment Caching

- Most dynamic Web pages are far more complex than our simple Time.jsp and consist of multiple smaller and simpler page fragments. Some fragments are static (such as headers and footers), while others are dynamic (such as fragments containing stock quotes or sport scores).

- Breaking a page into fragments or components makes effective caching possible for any page, even a highly dynamic page.

- For our example application, Time.jsp now becomes a fragment of its parent JSP, Display.jsp
Caching Rules in cachespec.xml

- Consider the simple JSP, Display.jsp.
- The request to Display.jsp returns itself, as the parent, plus the included fragment Time.jsp.
Caching Rules in cachespec.xml

consume-subfragments

- The consume-subfragments property tells the cache not to stop saving content when it includes a child servlet. The parent entry will include all the content from all fragments in its cache entry, resulting in one big cache entry.

- Use the <exclude> element to tell the cache to stop consuming for the excluded fragment and instead, create a placeholder for the include or forward. For example, exclude Time.jsp from the consume-subfragment, as follows...
Caching Rules in cachespec.xml

```xml
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/Display.jsp</name>
    <property name="consume-subfragments">true</property>
    <exclude>/Time.jsp</exclude>
  </cache-entry>
  <cache-id>
    <timeout>30</timeout>
  </cache-id>
</cache>
```
Caching Rules in cachespec.xml

As discussed, the consume-subfragments property tells the cache to save all content including child fragments.

A fragment can be excluded from the rules of the parent by using the do-not-consume property.

In this cachespec, Time.jsp is labeled do-not-consume and can specify its own timeout or other rules.

```xml
<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/Display.jsp</name>
    <property name="consume-subfragments">true</property>
    <cache-id>
      <timeout>30</timeout>
    </cache-id>
  </cache-entry>
  <cache-entry>
    <class>servlet</class>
    <name>/Time.jsp</name>
    <property name="do-not-consume">true</property>
    <cache-id>
      <timeout>10</timeout>
    </cache-id>
  </cache-entry>
</cache>
```
Caching Rules in cachespec.xml

Inactivity.
- While the timeout directive dictates how long content can remain in cache before being refreshed, the Inactivity directive can cause a refresh prior to the timeout when a page is not used frequently.

```xml
<cache-id>
  <timeout>600</timeout>
  <inactivity>30</inactivity>
</cache-id>
```
Dependency IDs

- Dynamic Cache provides a group-based invalidation mechanism through dependency IDs.

- A dependency ID identifies a cache entries dependency based on certain factors, such that when those factors occur they trigger an invalidation of all the cache entries that share this dependency.

- An example of such a dependency could be the invalidation of a page which lists customer names. (Customer.jsp) Cached entries for this list should be invalidated when a customer is added to or removed from the list.
Dependency IDs

- SetTime.jsp has a dependency declared called settime.
- Time.jsp references the same dependency.
- The cachespec entry for both jsps contains:
  
  `<dependency-id>settime</dependency-id>`

- The cachespec entry for SetTime.jsp also contains the invalidation element
  
  `<invalidation>settime</invalidation>`

- Now, when SetTime.jsp is called, Time.jsp and any other cache entry with a dependency-id of settime will be invalidated.
Disk Offload

- By default, when the number of cache entries reaches the configured limit for a given application server, cache entries are removed from the memory cache, allowing newer entries to be stored in the cache. Use disk offload to copy the cache entries that are being removed from the memory cache to disk for potential future access.

- Disk offload is configured via the adminconsole either globally or for a specific cache instance.
Disk Offload

Cache Instances

- For this presentation, we can create a new servlet cache instance to demonstrate disk offload.

- Since different webapplications or pages may have different caching requirements, unique cache instances can be created for each. This allows applications to have greater flexibility and better tuning of cache resources.

- The characteristics of each servlet cache instance are a unique jndi name, disk offload policies, cache configuration and performance parameters.
Disk Offload

- Configure a new cache instance
- Resources – Cache Instances – Servlet Cache Instance – new
- A jndi name must be given to the cache instance which will be used to refer back to the instance in cachespec.xml.
- Check ‘Enable disk offload’ and specify a disk off load location, cache size and cache entry limits.
Disk Offload

```xml
<cache>
  <cache-instance name="offloadtest">
    <cache-entry>
      <class>servlet</class>
      <name>/Display.jsp</name>
      <property name="consume-subfragments">true
        <exclude>/Time.jsp</exclude>
      </property>
    </cache-entry>
    <cache-id>
      <component id="name" type="parameter">
        <required>true</required>
        <timeout>180</timeout>
      </component>
      <timeout>180</timeout>
    </cache-id>
  </cache-instance>
</cache>
```
Disk Offload

### Cache Instance

<table>
<thead>
<tr>
<th>Cache Instance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Size</td>
<td>100</td>
</tr>
<tr>
<td>Used Entries</td>
<td>100</td>
</tr>
<tr>
<td>Cache Hits</td>
<td>11</td>
</tr>
<tr>
<td>Cache Misses</td>
<td>125</td>
</tr>
<tr>
<td>LRU Evictions</td>
<td>25</td>
</tr>
<tr>
<td>Exploit Removals</td>
<td>0</td>
</tr>
<tr>
<td>Default Priority</td>
<td>1</td>
</tr>
<tr>
<td>Servlet Caching Enabled</td>
<td>Yes</td>
</tr>
<tr>
<td>Disk Offload Enabled</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Disk Cache Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used Entries</td>
<td>25</td>
</tr>
<tr>
<td>Used Data Size in MB</td>
<td>3,7205772</td>
</tr>
<tr>
<td>Disk Cache Garbage Collector Invalidations</td>
<td>0</td>
</tr>
<tr>
<td>Explicit Invalidations</td>
<td>0</td>
</tr>
<tr>
<td>Timeout Invalidations</td>
<td>0</td>
</tr>
<tr>
<td>Overflow Invalidations</td>
<td>0</td>
</tr>
</tbody>
</table>
Cache Invalidation

- It is essential that timely invalidations of cached content take place for the integrity of the website.

- Mechanisms for invalidation are:
  - timeout or inactivity directives within cachespec.xml
  - group-based invalidation mechanism through dependency IDs.
  - Programmatic invalidation via the cache API 'com.ibm.websphere.cache.*'
  - The CacheMonitor
**Dynamic Cache Performance Tips**

- WebSphere Application Server uses JVM memory to store cached objects. Therefore, it is important to know how much memory can be allocated for the cache and based on this information you can set the cache size to the proper value.

- Increase the priority of cache entries that are expensive to regenerate.

- Modify timeout of entries so that they stay in memory as long as they are valid.

- If the estimated total size of all cached objects is bigger than the available memory, you can enable the disk offload option.

- Increase the cache size if memory allows.
Troubleshooting Dynamic Cache

- The following troubleshooting tools are available for Dynamic Cache.
  - Cache Monitor
  - SystemOut log
  - Dynamic Cache Trace
  - Performance Monitoring Infrastructure (PMI)
  - Cache Statistics Visualization MBeans
  - Review the troubleshooting tips page of the WebSphere Information Center

Troubleshooting Dynamic Cache

Reviewing the SystemOut log.

- Positive messages
  - ResourceMgrIm I WSVR0049I: Binding services/cache/diskoffload as services/cache/diskoffload
  - CacheOnDisk I DYNA0053I: Offload to disk is enabled
  - CacheServiceI I DYNA0048I: WebSphere Dynamic Cache initialized successfully.
  - ConfigManager I DYNA0062I: Successfully loaded cache-instances from configuration file

- Problematic messages
  - ConfigErrorHa E DYNA0045E: XML parsing error: The markup in the document following the root element must be well-formed.
  - SystemOut O SAXParseException loading cache entries
  - RuleHandler E DYNA0037E: Unrecognized element property while processing element <property>.
Troubleshooting Dynamic Cache

- **Dynamic Cache Trace**
  - Use the WebSphere trace facility to review key trace points and verify expected caching behavior.

- **Enabling Trace**
  - Dynamic Cache issues can be traced using the following trace specification:
    - Dynacache replication is disabled
      - *=info:com.ibm.ws.cache.*=all
    - Dynacache replication is enabled

- For information regarding trace setting please refer to the WebSphere information center or see this link:
Troubleshooting Dynamic Cache

Binding to cache instance
ResourceMgrIm WSR0049I: Binding services/cache/diskoffload as services/cache/diskoffload

First request
CacheHook 3 handleServlet: absoluteUri = /referenceWEB/Display.jsp
EntryInfo 3 set id=/referenceWEB/Display.jsp:name=Fred:requestType=GET
FragmentCompo 3 setConsumeSubfragments: /Display.jsp consumeSubfragments=true
FragmentCompo 3 setDoNotConsume: /Display.jsp doNotConsume=false
CacheStatisti 3 CACHE: Cache Miss: /referenceWEB/Display.jsp:name=Fred:requestType=GET
CacheHook 3 CACHE MISS id: /referenceWEB/Display.jsp:name=Fred:requestType=GET
CacheHook 3 servicing /referenceWEB/Display.jsp:name=Fred:requestType=GET
FragmentCompo 3 saveCachedData uri=/Display.jsp

Next request:
CacheStatisti 3 CACHE: Local Cache Hit: /referenceWEB/Display.jsp:name=Fred:requestType=GET
CacheHook 3 CACHE HIT id:/referenceWEB/Display.jsp:name=Fred:requestType=GET
Invalidation due to cache timeout
Cache > internalInvalidateById() cacheName=baseCache id=/referenceWEB/Display.jsp:name=Fred:requestType=GET Entry
Cache < internalInvalidateById() id=/referenceWEB/Display.jsp:name=Fred:requestType=GET Exit

- If a problem is to be submitted to IBM Support, please collect trace and configuration by following this Mustgather document
Additional Dynamic Cache technologies

- Dynamic Cache API
- Cache replication
- ESI Caching
- Command Caching
- Object caching
- WebServices Caching
Additional Dynamic Cache technologies

- Cache Replication

- Servers > Application servers > server_name > Container services > Dynamic cache service.
- select Enable cache replication.
- Global policies for sharing can be configured through the admin console and can be overridden within cachespec.xml.
- Sharing policies can be Both push and pull, push only or not shared.
- See the following link for more replicator configuration information
Additional Dynamic Cache technologies

- ESI Caching
  - A facility exists within the EdgeSideInclude ESI cache of the Websphere plugin to cache fragments.
  - Fragments which are edge cacheable should have the following property set within cachespec.xml.
    ```xml
    <property name="EdgeCacheable">true</property>
    ```
  - Details and examples are available in the IBM Redbook WebSphere Application Server V6 Scalability and Performance Handbook or at the following WebSphere Information Center page:
Additional Dynamic Cache technologies

Object caching

- Using the DistributedMap and DistributedObjectCache interfaces allows applications to cache and share Java objects by storing a reference to the object in the cache.

- An example of object caching could be the loading of a pricelist or catalog data object.

- For more information, see the API documentation for the com.ibm.websphere.cache package or follow this Information center link:
  
Useful links and Resources

- A technology preview of the Extended Cache Monitor for WebSphere is currently available at:

- The Dynamic Cache Collector and Visualizer for IBM WebSphere Application Server enables a Dynamic Cache user to collect and visualize the cache statistics exposed by the DynaCache Mbean
Acknowledgements

- IBM Redbooks
  - WebSphere Application Server V6 Scalability and Performance Handbook
    http://www.redbooks.ibm.com/abstracts/SG246392.html
  - Mastering DynaCache in WebSphere Commerce
    http://www.redbooks.ibm.com/abstracts/sq247393.html

- WebSphere Dynamic Cache: Improving J2EE application performance
Additional WebSphere Product Resources

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/

- Learn about other upcoming webcasts, conferences and events: http://www.ibm.com/software/websphere/events_1.html

- Join the Global WebSphere User Group Community: http://www.websphere.org

- Access key product show-me demos and tutorials by visiting IBM Education Assistant: http://www.ibm.com/software/info/education/assistant


- Sign up to receive weekly technical My support emails: http://www.ibm.com/software/support/einfo.html
Questions and Answers