



IBM Software Group | Lotus Software

# Lotus® Web Content Management Caching and Performance Tuning

IS26 23.04.08

Filip Zawadiak  
Certified IT Specialist

team best practice lotus quickplace community **ISSL 2.0** deployment consulting portlet development app dev lotus connections **Professional** lotus domain upgrading infrastructure  
basic social software foundation deep dive knowledge market workshop imagination composite application **Symposium** advanced effective thought leader skills development architecture  
project management lecture lotus notes excellence value **20-24 April 2008** design business development networking lotus forms wcm **Porto, Portugal** ibm webphere portal performance tuning lotus sametime enablement



## Agenda

- Introduction
- Tools, Components and Architectures
- Performance Testing of Cached Sites
- Advanced Caching Strategies
  - ▶ Synchronization
  - ▶ Invalidation
  - ▶ Personalization
- Web Content Management Caching, Servlet Caching
- Web Content Management & Servlet Caching Examples
- Assets, References
- Reference materials and examples



## Universal Truths

- 1. Most scalable number is zero**
- 2. Things that do nothing cannot be slow**
- 3. Web Content Management must be cached – it does a lot!**





## Web Content Management Usage Scenarios

- **Static Site – very rare**
  - ▶ Uses prerendering and HTTP server
  - ▶ Caching not needed
- **Web Site – happens**
  - ▶ Content delivered by Web Content Management servlet
  - ▶ Additional functionality using JSP components
- **Portal – usual**
  - ▶ Content delivered by Web Content Management portlets
  - ▶ Additional functionality: JSP components, Portlets



## Performance Tuning – Slow Approach

- Tune lot of parameters: JVM, heaps, GC
- Database connection pool tuning
- Need to check performance impact after each change
- Usually gains are not very big
- The faster things work the bigger probability of faults
- Tests, if done at all use complex user simulation scripts
  
- **Tests are time consuming and hard to setup**
- **Tests are performed by dedicated QA team**
- **As a result, tests are rarely run. And too late!**



## Performance Tuning – Effective Approach

- Basic tuning of JVM
- Servlet caching for Web Content Management and portlets
- Caching proxy for static resources
- Speedup – at least **10 times**
- Lot of things cached – less work for Portal & Web Content Management
- Much lower usage of database connections
- Less opportunities for memory leaks
- **Never forget to test cache expiration scenarios!**
- **Quick, procedural and huge performance improvements**
- **It's still worth doing user simulation test**
- **It's best to let customer QA do those and cross-check**



## Recommended Tools

- **OpenSTA** <http://www.opensta.org>  
Small (8MB), efficient and free load testing tool.
- **IBM® PageDetailer** <http://w3.alphaworks.ibm.com/tech/pagedetailer>  
Makes Gantt-like charts of HTTP requests from browser side.
- **IBM Rational Performance Tester**  
Very powerful, big and complicated product. Much easier to use for user simulation than OpenSTA.
- **WireShark** <http://www.wireshark.org>  
Network protocol analyzer. For sanity checks.
- **Theme&Skin Timers**  
These are described later. Essentially they show rendering time of each portlet on page and time used to completely render whole page.



## Skin Timer

Following code added around portletRender tag in Control.jsp of each skin

```
<div style="border: 1px solid red; padding 2px;">  
<% long start = java.lang.System.currentTimeMillis(); %>
```

```
... <wps:portletRender...> ... </wps:portletRender> ...
```

```
<div style="color: red; padding: 4px; border: solid red; ">  
<%= java.lang.System.currentTimeMillis() - start %>ms  
</div>  
</div>
```



## Theme Timer

Following code added in Default.jsp

```
...  
<body>  
<% long pagestart = java.lang.System.currentTimeMillis();%>  
...  
<div style="color: red; padding: 4px; border: solid red;">  
<%= java.lang.System.currentTimeMillis() - pagestart %>ms  
</div>  
</body>  
...
```

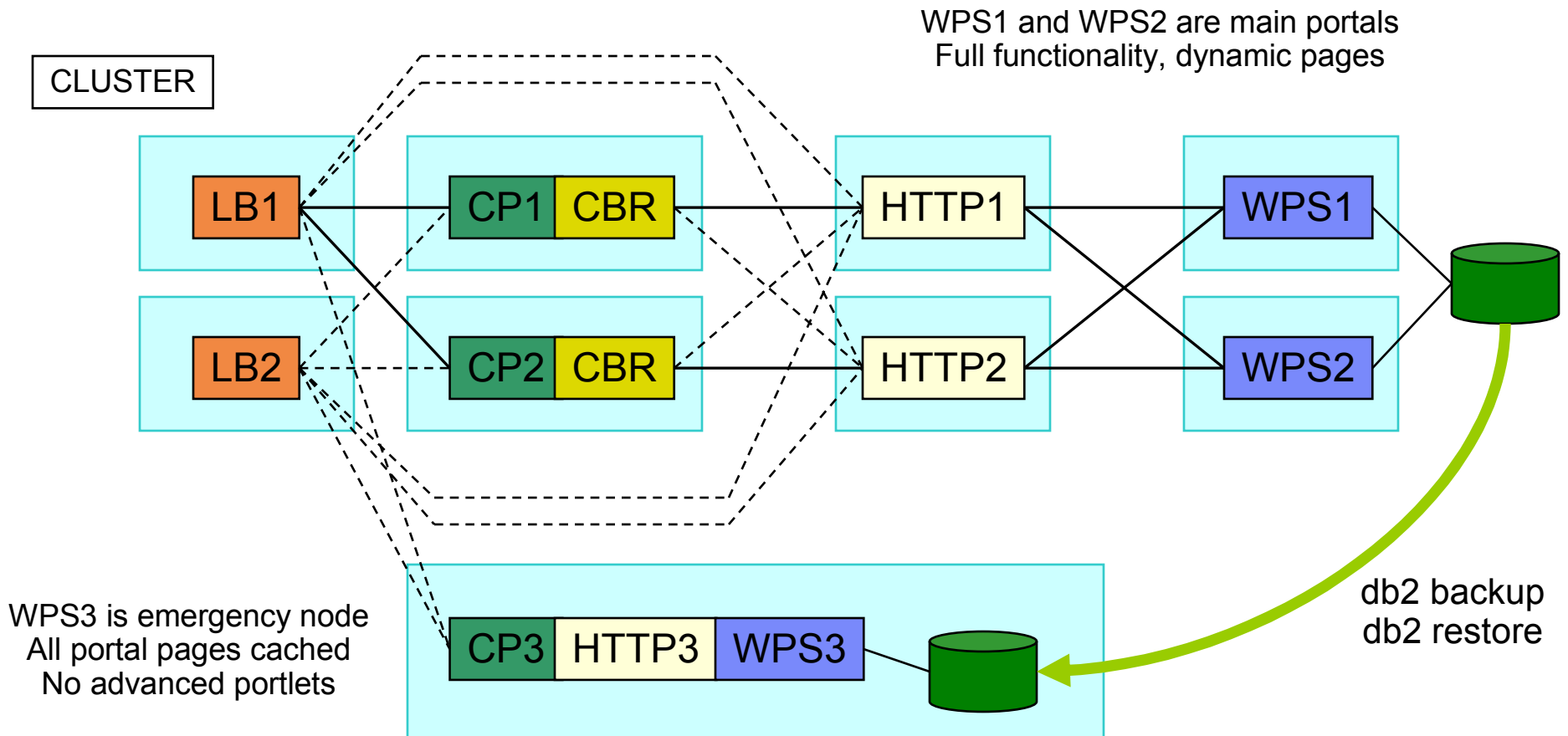


# Edge Components

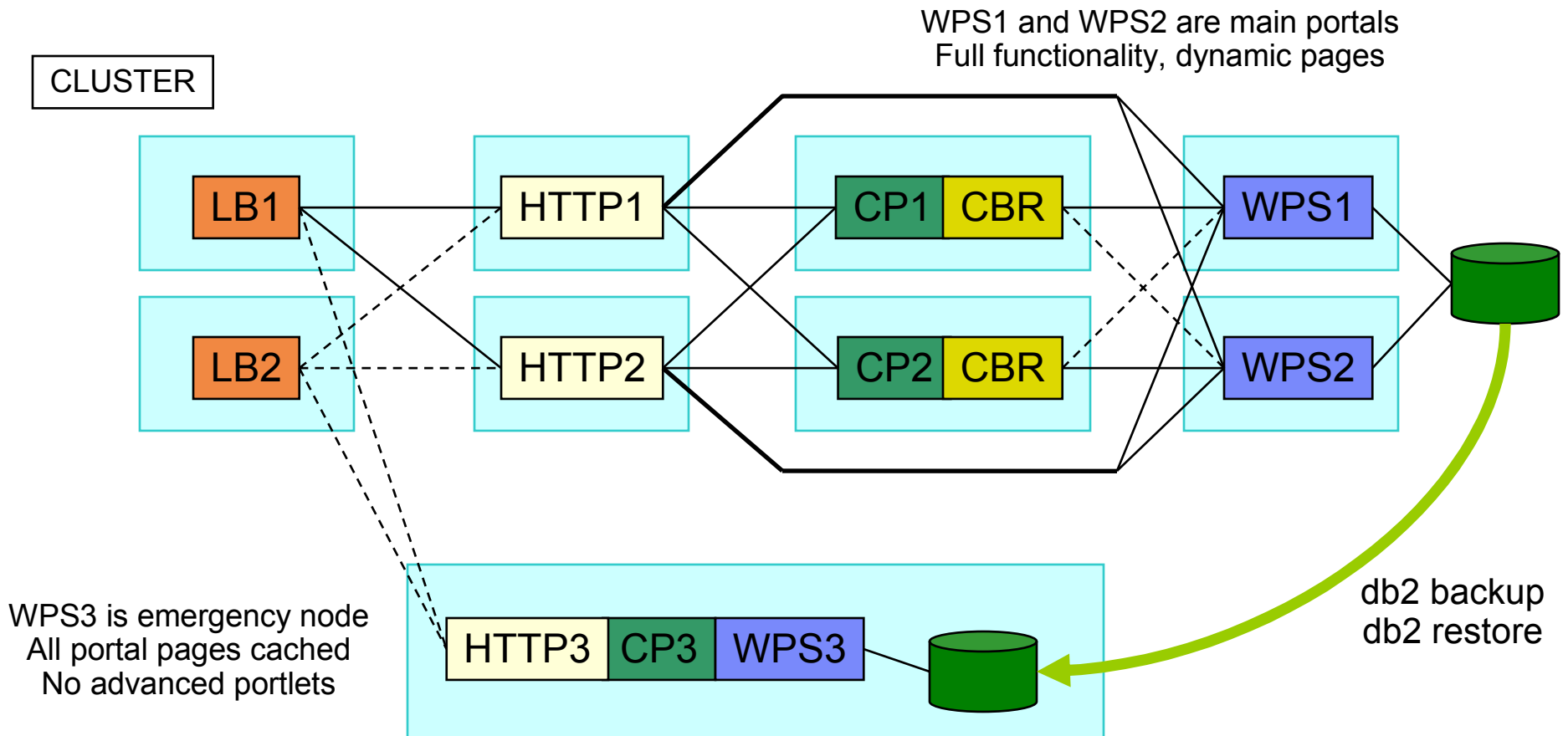
- **Network Dispatcher (ND)**
  - ▶ Load balancer operating on IP level
  - ▶ Supports hot-standby configuration and automatic failover
  - ▶ Uses response times and number of connections to balance
- **Caching Proxy (CP)**
  - ▶ Reverse proxy
  - ▶ No direct support for clustering
  - ▶ It is possible to override caching headers, but better to do it on HTTP server
- **Content Based Router (CBR)**
  - ▶ TCP level equivalent of Network Dispatcher
  - ▶ Plugin to Caching Proxy (CP)
  - ▶ Gives CP ability to switchover between backend servers



# Recommended Architecture – CP at the front



# Recommended Architecture – HTTP at the front



## Proxy Placement Comparison

### ■ Proxy at the front

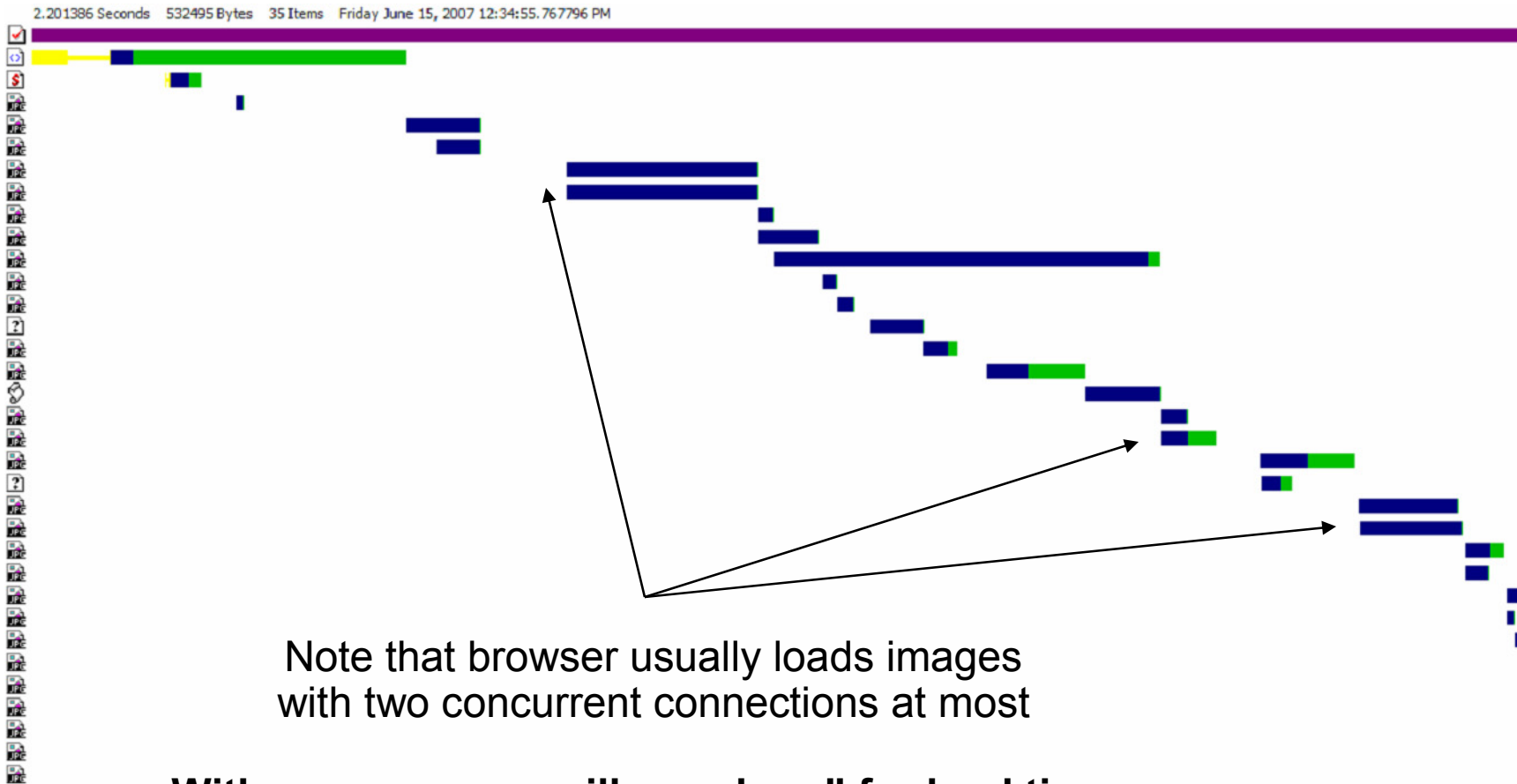
- ▶ Use mod\_headers for Cache-Control override
- ▶ WAS Plugin does all session affinity
- ▶ No authorization for cached resources!
- ▶ Nice reporting with ITCAM for RTT plugin
- ▶ Best performance
- ▶ No NTLM/Kerberos at HTTP possible!

### ■ HTTP at the front

- ▶ Need to use custom servlet filter to override Cache-Control
- ▶ Need to implement additional session affinity rules in CBR
- ▶ You can use native ITCAMforRTT functionality for reporting
- ▶ ITCAMforRTT plugin will supply additional hit ratio data
- ▶ HTTP can do authentication
- ▶ Can use NTLM/Kerberos at HTTP



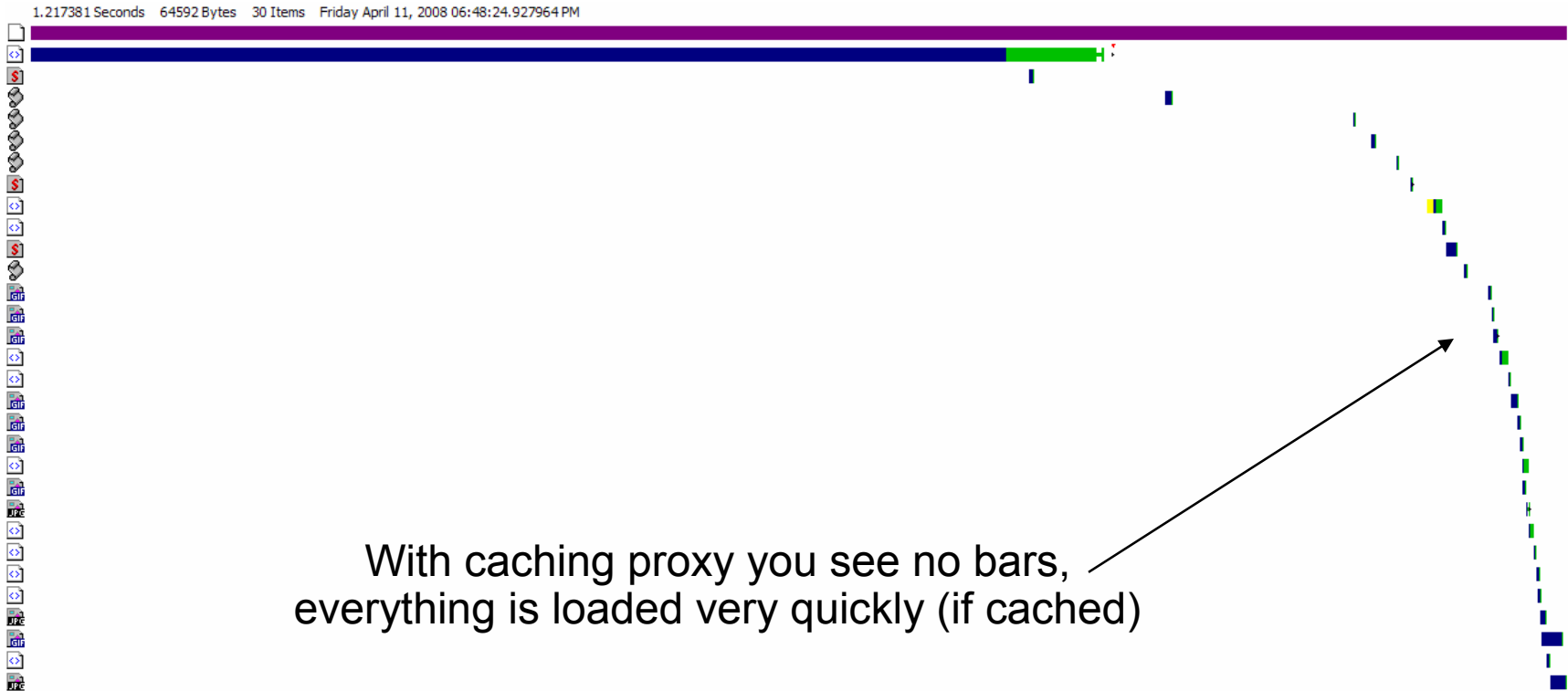
# Page Detailer – without proxy



**With no proxy you will see „bars” for load times**



# Page Detailer – with proxy



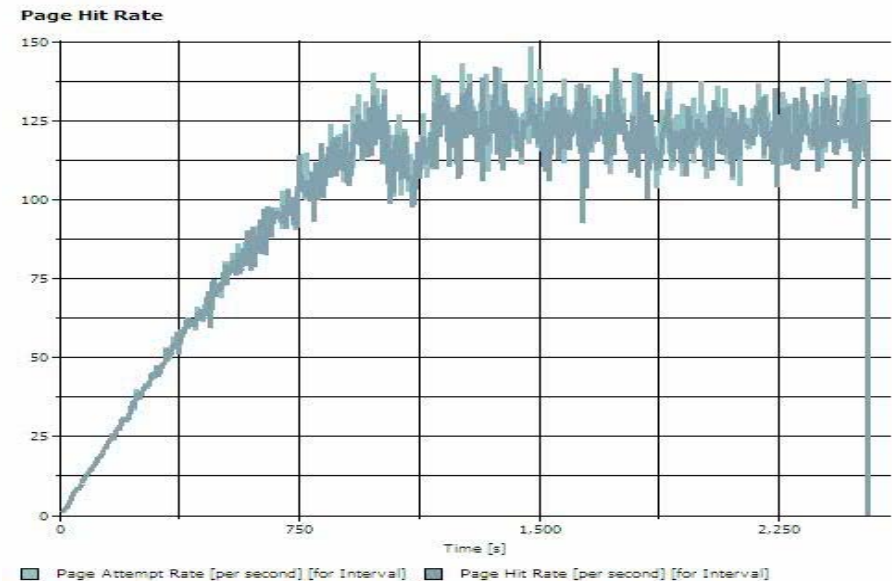
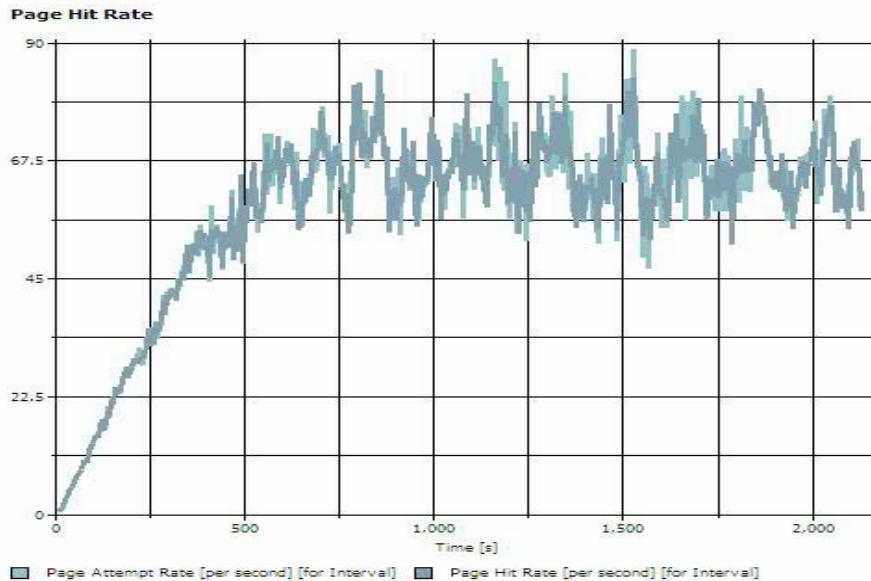
# Do you really need caching proxy?

## Without caching proxy:

- Page views per second: **67**
- Requests per second: **600**
- Avg page resp. time: **1700ms**
- Avg resource resp. time: **165ms**

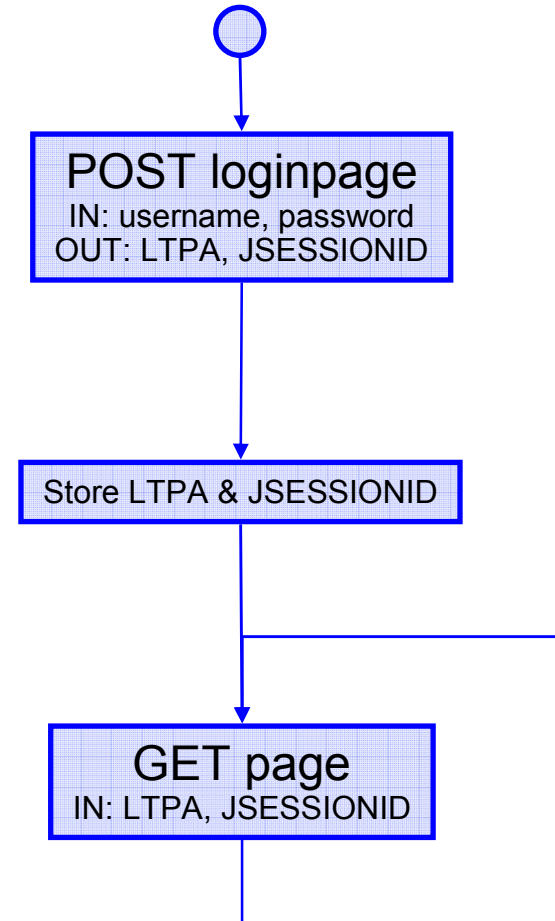
## With caching proxy:

- Page views per second: **125**
- Requests per second: **1400**
- Avg page resp. time: **662ms**
- Avg resource resp. time: **56ms**



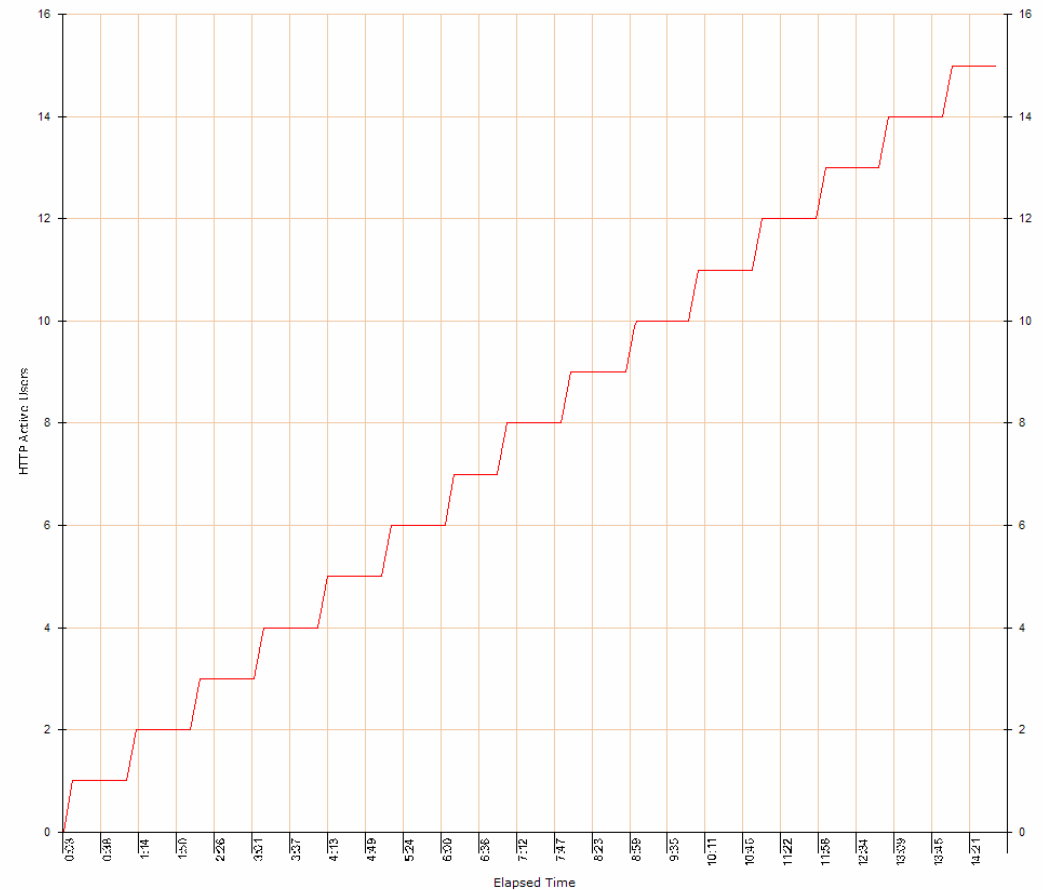
# Caching Performance Test Script

- No need to decide on arbitrary parameters
  - ▶ Think times
  - ▶ User tasks sequence
  - ▶ Creation of test users
- You get three important numbers
  - ▶ **Maximum responses per second**
  - ▶ **Response time at that time**
  - ▶ **CPU usage at that time**
- CAUTION!
  - ▶ This test runs very fast
  - ▶ Reuse LTPA and sessions
  - ▶ Tune TCP/IP configuration at tester machine
  - ▶ Monitor loading machine CPU usage!



# Caching Performance Graphs – User Rampup

- Every user runs the same script
- New user added every 60 seconds
- Two important graphs collected
  - ▶ Number of requests per second
  - ▶ Response time
- **Remember to monitor loading machine CPU and network!**
- If 80% is reached you need to organize second or faster machine
- Fast Ethernet is not THAT fast – very easy to saturate



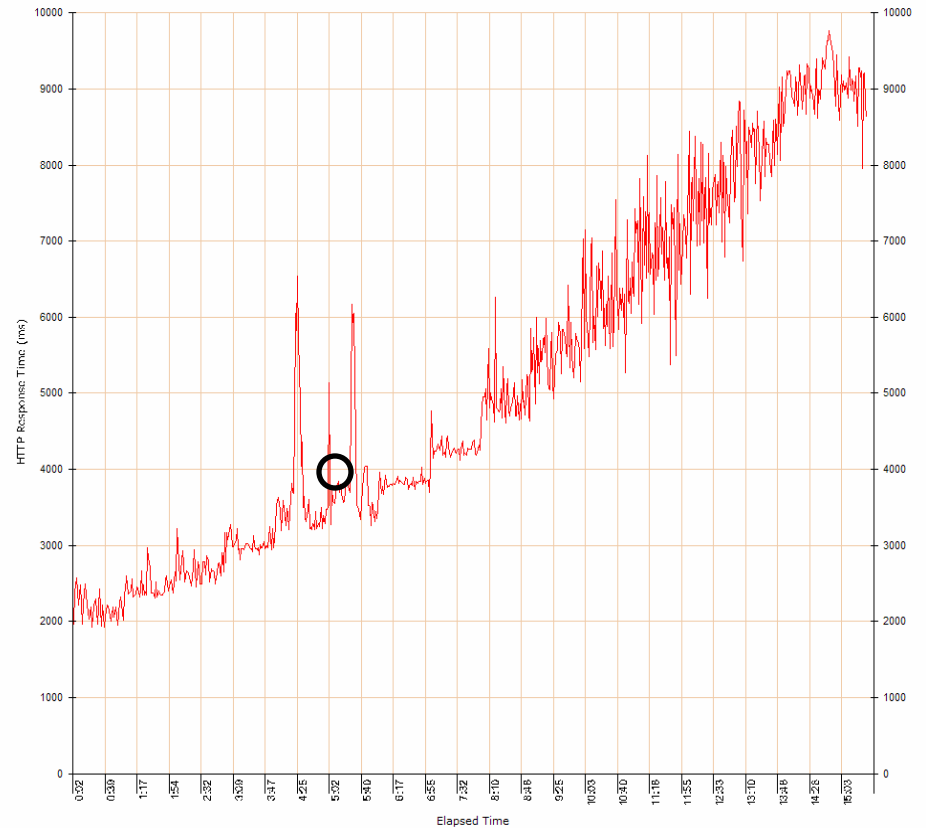
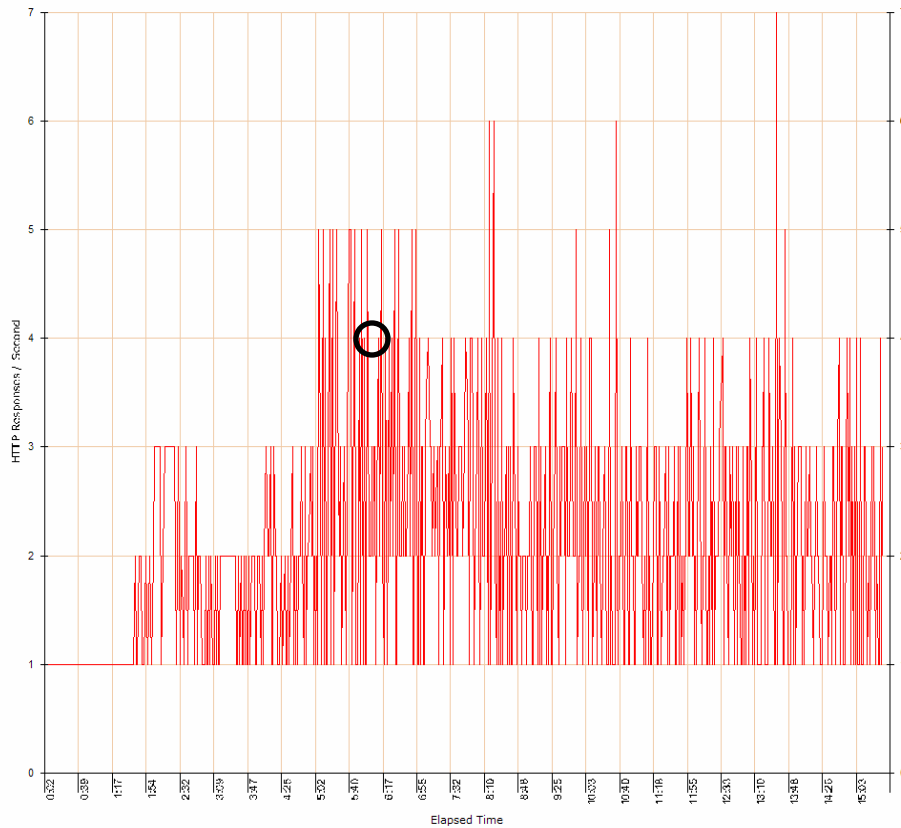
## Caching Performance Numbers and Reality

- Customers usually want to know how many users site will handle
- Easy to answer
  - ▶ Pick your think time
  - ▶ Multiply by max. responses per second
  - ▶ EQ: **60s \* 75rps = 4500** users
  - ▶ **Assuming 100% cache hit ratio**
- How to increase hit ratio
  - ▶ Increase cache size
  - ▶ Increase expiration timeouts
  - ▶ Use same cache keys for the same content
- How to decrease publishing time
  - ▶ Automatic invalidation
- **Hopefully, it is possible to reduce impact of cache miss**



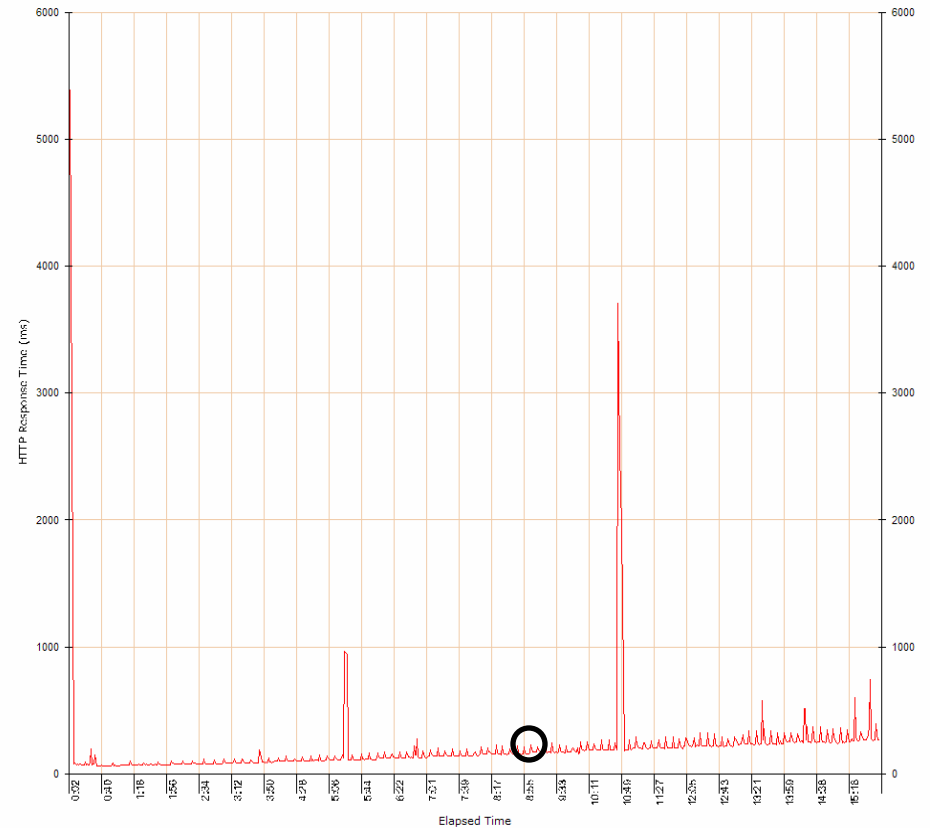
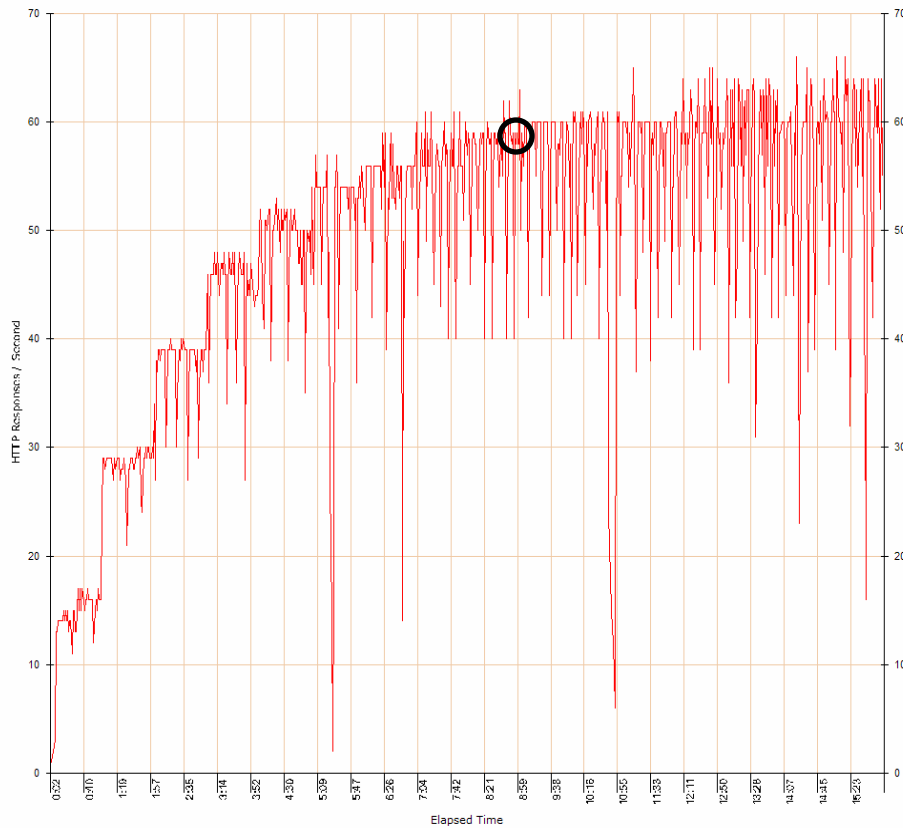
# Graphs – Without Caching

Peak performance: **4rps**, response time **3500ms**. **100%** CPU utilization.



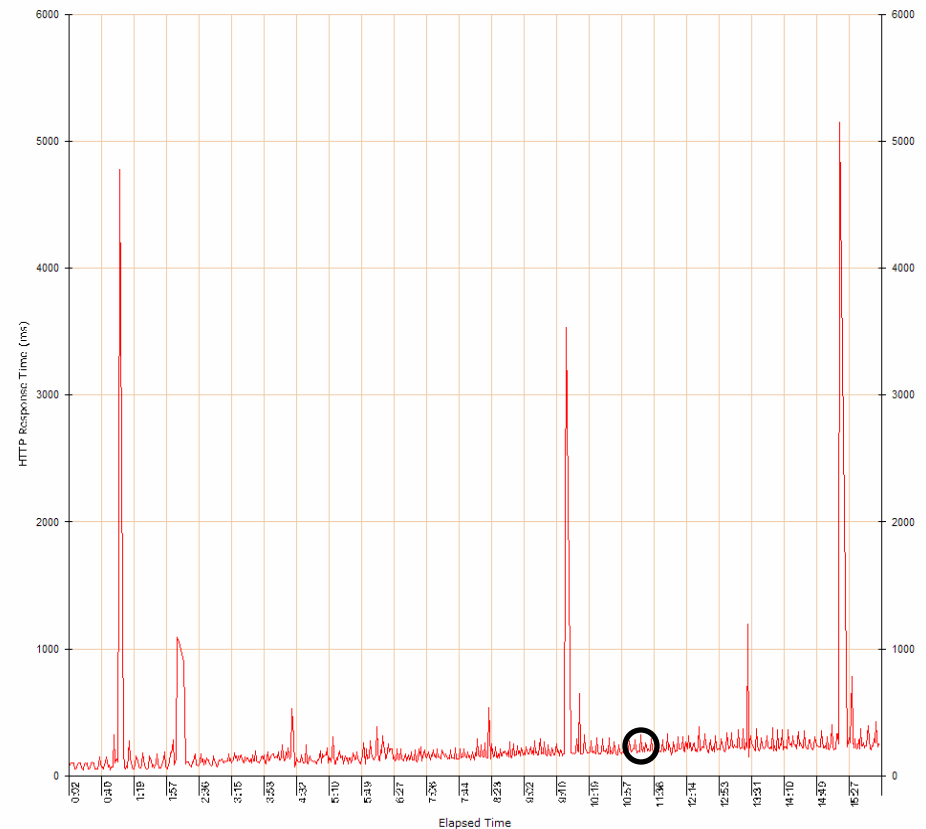
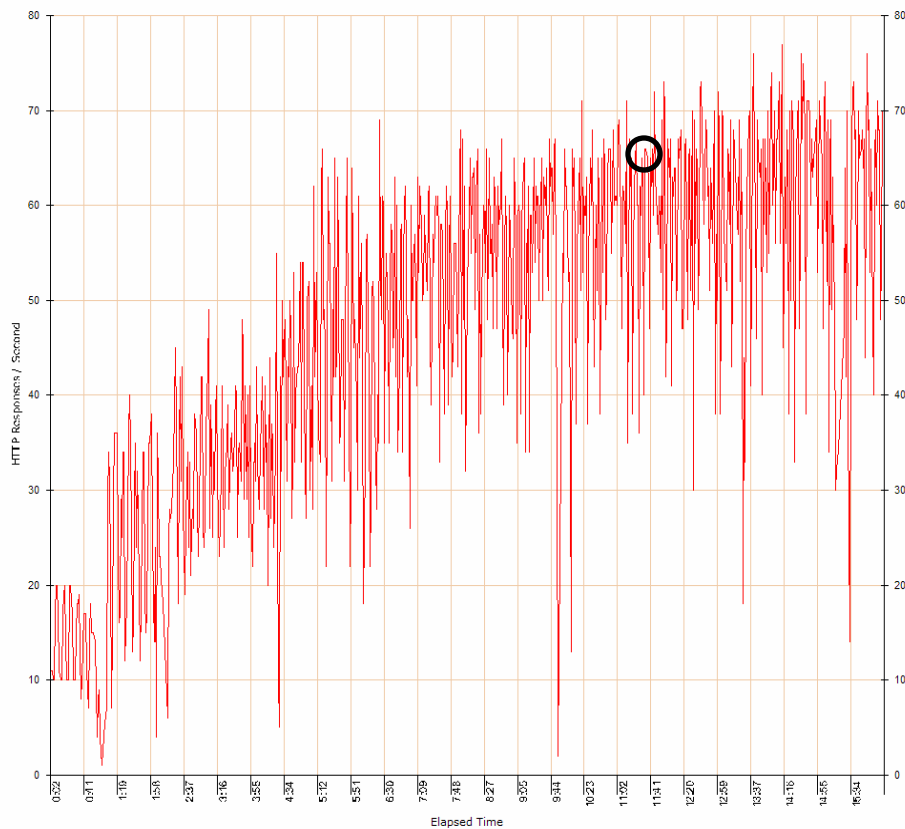
# Graphs – With Web Content Management SITE Caching

Peak performance: **60rps**, response time **250ms**. **75%** CPU utilization.



# Graphs – With Servlet Caching

Peak performance: **65rps**, response time **200ms**. **20%** CPU utilization.



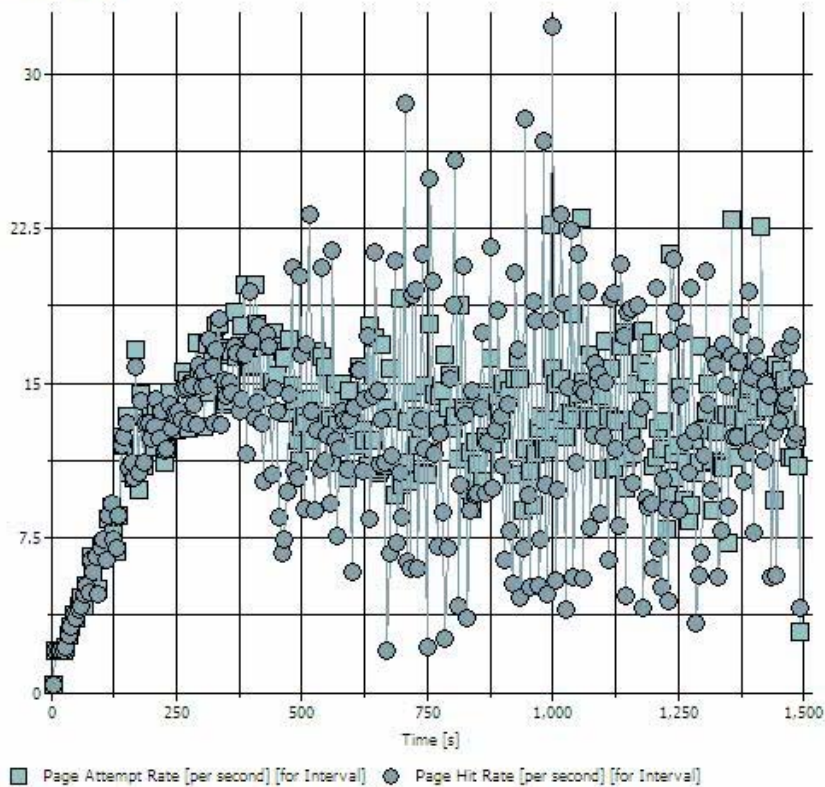
## Cache Expiry Test Scenarios

- Results look great with 100% cache hit ratio
- Scenarios that have to be tested:
  - ▶ Rendering cache miss
  - ▶ Rendering cache miss after simulated syndication/edit event
- Why this is important
  - ▶ Need to make sure system will cope with it under load
  - ▶ Syndication and content edits **clear ALL items** in Web Content Management **navigation** and **menu** caches. Huge performance hit!
- Requires Extended Cache Monitor

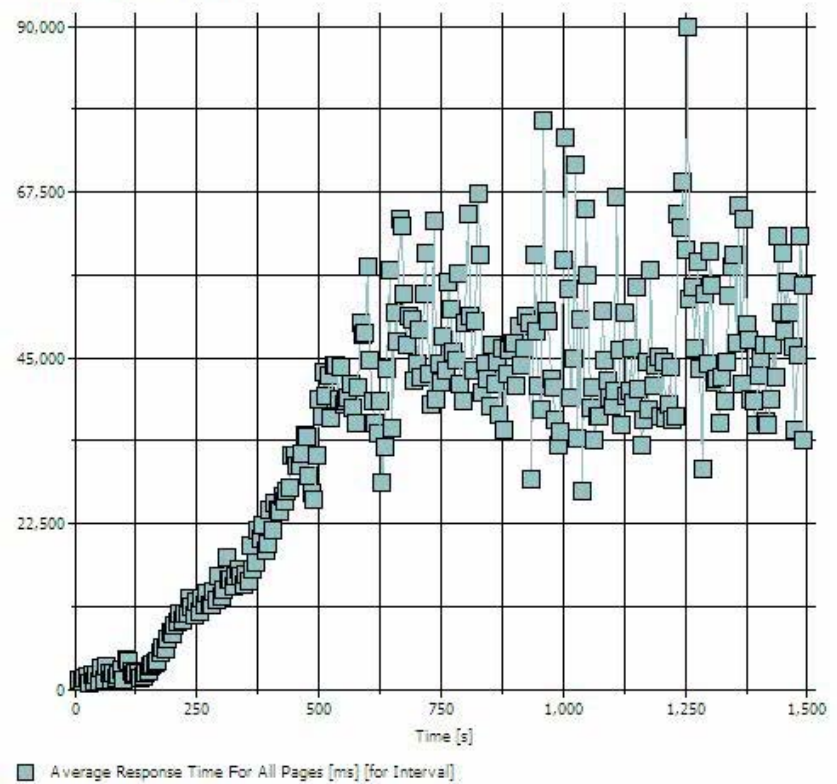


# Performance – Before Tuning

Page Hit Rate



Page Response vs. Time

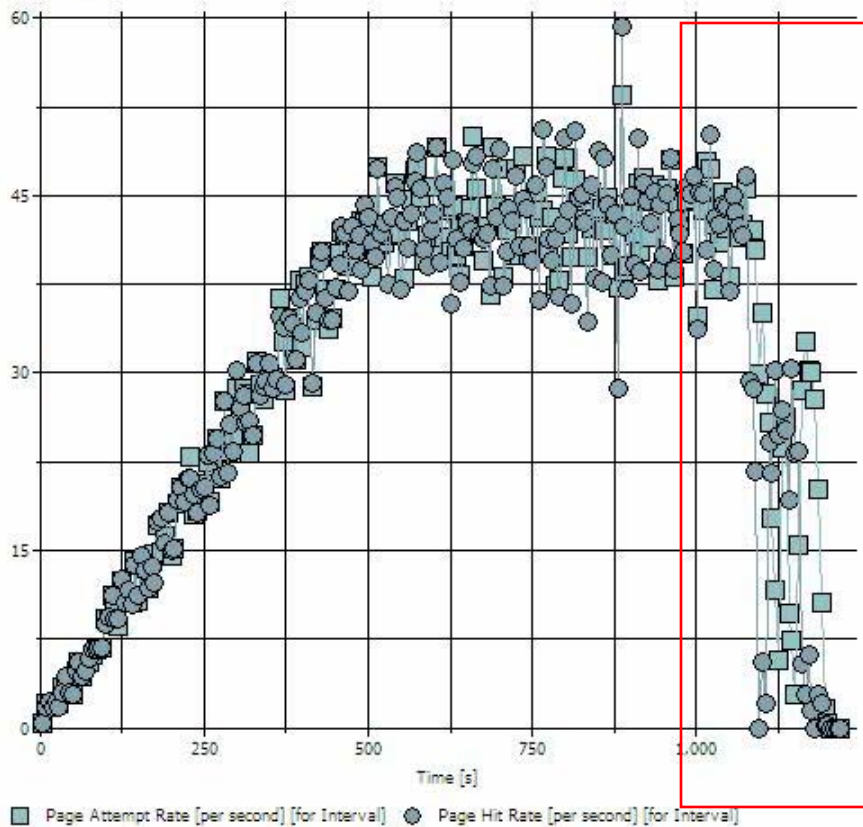


Average response time for HTML: 37'102ms  
 Average response time for images: 1'919ms  
**Page loading time:  $37102 + 1919 \cdot 34/2 = 69'725\text{ms}$**

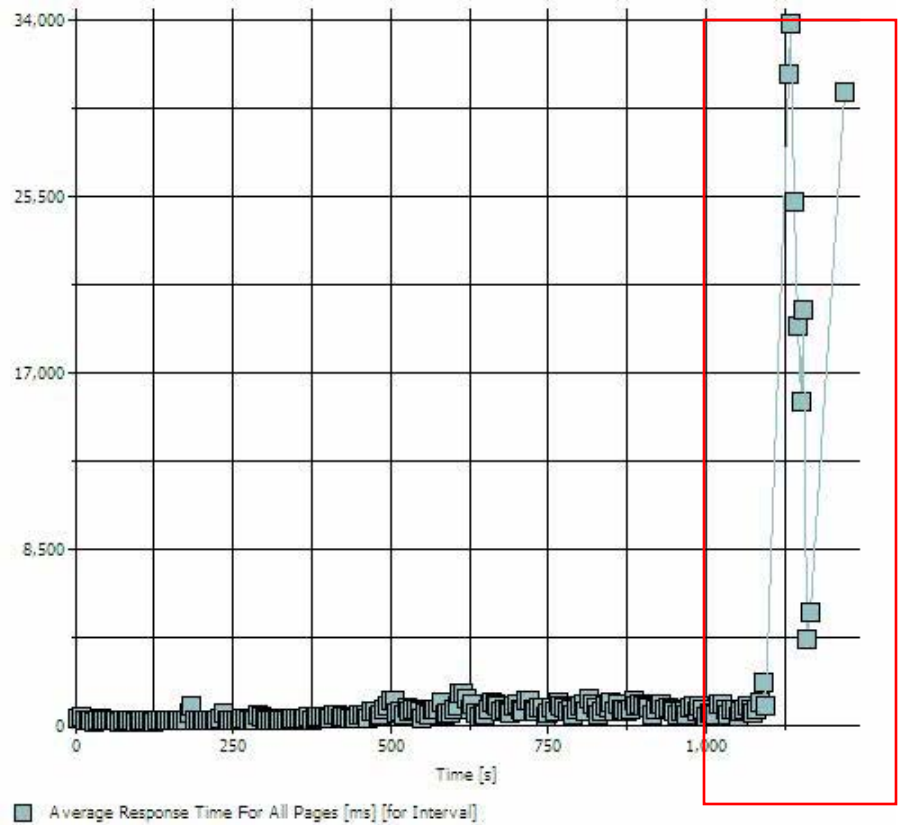


# Performance – Great until Cache Expiration

Page Hit Rate



Page Response vs. Time

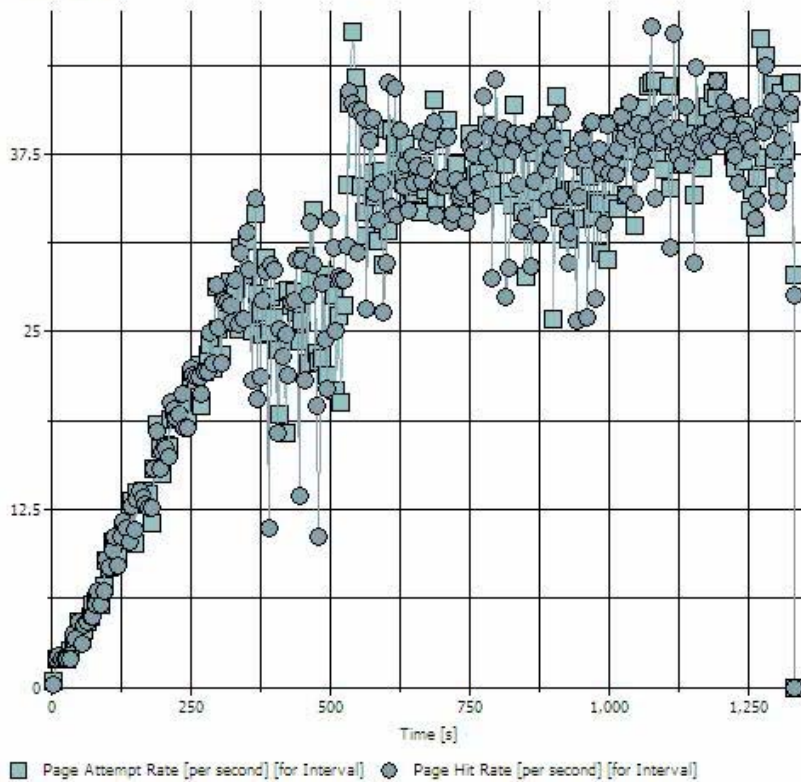


All portlets expired after 1200 seconds. Test could not be continued, no responses. Random caching times need to be introduced.

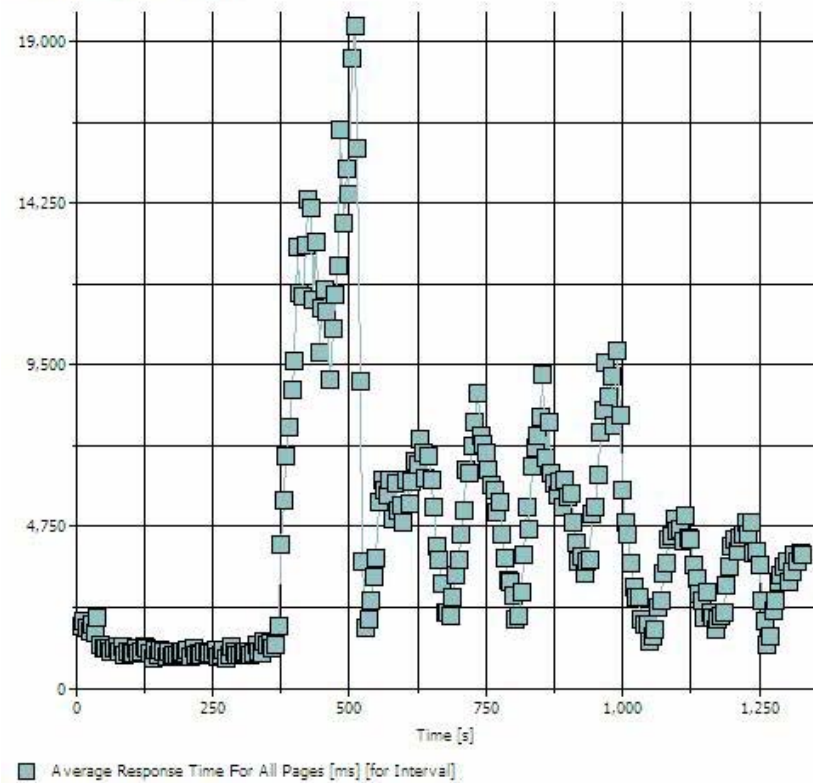


# Performance – Multigeneration Cache

Page Hit Rate



Page Response vs. Time



Eight cache instances used, selected randomly. Timeout  $400+n*25$   
 Average response time for HTML: 4'786ms (775% better)  
 Average response time for images: 1'45ms (1323% better)  
**Page loading time:  $4'786 + 145*34/2 = 7'251ms$  (962% better)**



## Cache Expiration Tests

- Rendering cache miss
  1. Quickly start all users
  2. Wait 60 seconds
  3. Clear rendering cache
  4. Wait 60 seconds
- Rendering cache miss after simulated synd/edit event
  1. Quickly start all users
  2. Wait 60 seconds
  3. Clear menu cache
  4. Clear navigation cache
  5. Clear rendering cache
  6. Wait 60 seconds



## Cache Background Refresh

- Simple version
  - ▶ Do automatic page hit before cache expiry
  - ▶ Essentially, cached entries never expire
  - ▶ And are frequently refreshed
- Complex version
  - ▶ When user triggers cache refresh, return cached version
  - ▶ Submit background work for refresh
  - ▶ Implement WorkManager to manage those
  - ▶ Very complicated to do with Web Content Management, as it requires request and response objects to work. Requires custom implementation.
  - ▶ Other solution – use HTTP requests to fetch content from Web Content Management servlet.



## Automatic Invalidation: Dependency IDs

- Dependency ID is configured like cache components
- Examples of dependency IDs
  - ▶ Portal Page Title
  - ▶ Portal Path
  - ▶ Web Content Management Site Area
- To force refresh of number of portlets
  - ▶ All „Sports” category
  - ▶ All information about this department page
  - ▶ All portlets using this Web Content Management context
- To make finding specific content in Cache Monitor faster







## Web Content Management Rendering Caches

- Web Content Management Cache settings apply to all content
- Basic
  - ▶ Low memory usage
  - ▶ Same content for every user
  - ▶ Better to use Caching Proxy in typical scenario
- Advanced
  - ▶ Increased memory usage
  - ▶ Supports access control and personalization
  - ▶ Less configurable than servlet caching for Portal scenario
  - ▶ Types: SITE, SECURED, PERSONALIZED, USER, SESSION



## Servlet Caching

- Servlet Caching caches requests at the WebSphere® layer
- Faster than Web Content Management Content Caching
- Extremely flexible
- Servlet Caching can cache
  - ▶ Web Content Management Rendering Portlets
  - ▶ Web Content Management Servlet
    - Requires specific configuration to keep syndication working properly
- Monitoring and management via the Cache Monitor web application
- Uses Dynacache
- Cache Instances Configurable with WAS Console



## Servlet Caching: Configuration

- Dynamic Cache Service enabled by default
- Servlet Caching disabled by default
  - ▶ Change in Application Server/Web Container configuration
- Cache Monitor used for management
  - ▶ Install from WASROOT/installableApps/cachemonitor.ear
  - ▶ Extended Cache Monitor patch from DeveloperWorks
  - ▶ Use my patch (small extension to above) to have
    - Cache Hit Ratios
    - Ability to preview Object Cache items
- Specific Cache Behaviour defined by configuration file
  - ▶ WEBAPP/WEB-INF/cachespec.xml



# Servlet Caching: cachespec.xml example

```

<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-instance name="jndi/cache-instance-name">
    <cache-entry>
      <class>servlet</class>
      <name>/jsp/html/view-original.jsp</name>
      <sharing-policy>not-shared</sharing-policy>
      <property name="save-attributes">false</property>
      <property name="ignore-get-post">true</property>
      <property name="consume-subfragments">true</property>
      <cache-id>
        <component id="UNIQUE_ID" type="attribute">
          <required>true</required>
        </component>
        <component id="WCM_GLOBAL_CONTEXT" type="parameter">
          <required>false</required>
        </component>
        <component id="PAGE" type="attribute">
          <required>false</required>
        </component>
        <timeout>150</timeout>
        <metadatagenerator>
          com.ibm.pl.fz.randomcacher.RandomCacher
        </metadatagenerator>
      </cache-id>
    </cache-entry>
  </cache-instance>
</cache>

```







## Servlet Caching: cachespec.xml (3)

- `<component>`  
Defines specific part of cache ID
  - ▶ `parameter` – will use value of named GET/POST parameter
  - ▶ `attribute` – will use value of named request attribute
  - ▶ `session` – will use value of named session attribute
- `<required>`
  - ▶ If component is required, but value is null request will not be cached
- `<value>`
  - ▶ Defines specific value, that is cacheable
- `<not-value>`
  - ▶ Defines specific value, that is not cacheable



## Servlet Caching: Generic Workflow

- Create basic version that will cache page
- Check that contents of page will change when performing actions
- Add components to make actions change content of page
- cachespec.xml automatically reloads after changes
- Modified JSPs should reload automatically
- Repeat
  
- **It is tricky!**



## Servlet Caching: Web site specific

- Advanced caching logic requires populating request attributes before calling Web Content Management servlet
- Can be achieved using
  - ▶ servlet filter
  - ▶ custom metadata generator
- Both require Java class creation
- Servlet filter requires changing of web.xml
- Custom metadata generator need to be put into WEB-INF/classes folder
- Slower development cycle, requires application restart



## Servlet Caching: Use with Web Content Management Servlet (1)

- Name of Web Content Management servlet is **com.presence.connect.ConnectServlet.class**
- Some requests can never be cached, include following in <cache-id> section

```
<component id="MOD" type="parameter">  
  <required>false</required>  
  <not-value>Subs</not-value>  
  <not-value>Synd</not-value>  
  <not-value>ItemDispatcher</not-value>  
  <not-value>Syndication</not-value>  
  <not-value>MemberFixer</not-value>  
  <not-value>VersioningEnablement</not-value>  
  <not-value>WorkflowEnablement</not-value>  
  <not-value>PlutoUploadFile</not-value>  
  <not-value>PlutoDownloadFile</not-value>  
  <not-value>AJPECatSelect</not-value>  
  <not-value>RefreshAllItems</not-value>  
  <not-value>Template</not-value>  
</component>
```



## Servlet Caching: Use with Web Content Management Servlet (2)

- Additional components to include in the cache-id section
- All should be marked as not required
- What they are used for:
  - ▶ **id** To choose between Draft and Published versions
  - ▶ **pagedesign** To choose between alternate page designs
  - ▶ **version** To choose between different versions
  - ▶ **source** To render components directly
  - ▶ **cmpntname** To render components directly
  - ▶ **cmpntid** To render components directly





## Web Content Management Servlet Caching – view.jsp

```
<%@ page session="true" contentType="text/html" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<%
portletRequest.setAttribute("UNIQUE_ID",
    portletResponse.encodeNamespace(portletRequest.getUser() == null ? "" : „anonymous"));
java.lang.StringBuffer sb = new java.lang.StringBuffer();

// Build paging list from all WCM_Page.<MenuID>=<PageNum> parameters
java.util.Map params = portletRequest.getParameterMap();
for(java.util.Iterator i = params.keySet().iterator();i.hasNext();) {
    String key = (String) i.next();
    int k = key.indexOf("WCM_Page.");
    if(k>=0) {
        // Store just menu IDs and page numbers
        sb.append(key.substring(k+9));
        sb.append("=");
        sb.append(params.get(key));
        sb.append(";");
    }
}
// If we have any pagers, add them to cache id
if(sb.length() > 0)
    portletRequest.setAttribute("PAGE",sb.toString());
%>
<jsp:include page="view-original.jsp"/>
```



# Web Content Management Servlet Caching – cachespec.xml

```

<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-instance name="jndi/portlet-dynacache">
    <cache-entry>
      <class>servlet</class>
      <name>/jsp/html/view-original.jsp</name>
      <sharing-policy>shared-push</sharing-policy>
      <property name="save-attributes">>false</property>
      <property name="ignore-get-post">>true</property>
      <property name="consume-subfragments">>true</property>
      <cache-id>
        <component id="UNIQUE_ID" type="attribute">
          <required>true</required>
        </component>
        <component id="WCM_GLOBAL_CONTEXT" type="parameter">
          <required>>false</required>
        </component>
        <component id="PAGE" type="attribute">
          <required>>false</required>
        </component>
        <timeout>150</timeout>
        <metadatagenerator>
          com.ibm.pl.fz.randomcacher.RandomCacher
        </metadatagenerator>
      </cache-id>
    </cache-entry>
  </cache-instance>
</cache>

```





## Multiple Caching Policies

- „Max 3 Web Content Management portlets per page rule” – **ignore!**
- **It is worth having multiple Web Content Management portlets on pages**
  - ▶ Different caching policies
  - ▶ Some elements are static, rarely change
  - ▶ Some change frequently
- Need to balance number of views with expiration times
- Can be accomplished in two ways:
  - ▶ Use individual UNIQUE\_ID values for portlets in cachespec.xml
  - ▶ Use something else: categories





# Multiple Caching Policies – cachespec.xml

```

<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-instance name="jndi/portlet-dynacache">
    <cache-entry>
      <class>servlet</class>
      <name>/jsp/html/view-original.jsp</name>
      <sharing-policy>shared-push</sharing-policy>
      <property name="save-attributes">false</property>
      <property name="ignore-get-post">true</property>
      <property name="consume-subfragments">true</property>
      <cache-id>
        <component id="UNIQUE_ID" type="attribute">
          <required>true</required>
        </component>
        <component id="CACHE_POLICY" type="attribute">
          <required>true</required>
          <value>DEFAULT</value>
        </component>
        <timeout>150</timeout>
      </cache-id>
      <cache-id>
        <component id="UNIQUE_ID" type="attribute">
          <required>true</required>
        </component>
        <component id="CACHE_POLICY" type="attribute">
          <required>true</required>
          <value>5876533A56C765EA6</value>
        </component>
        <timeout>3000</timeout>
      </cache-id>
    </cache-entry>
  </cache-instance>
</cache>

```



## ARM Reporting Plugin for Caching Proxy

- **Ready to use**
- Reports all requests from Caching Proxy to ITCAM for RTT
- Nested Transaction Correlation
- Reports for each hit:
  - ▶ Cache Hit/Miss
  - ▶ URL
  - ▶ Backend response time
- Configurable Transactions
  - ▶ URL Patterns
  - ▶ Content Types
- Defines Boundary Transactions
- Can replace WAS Plug-in (CBR reqd, no ESI)

Topology (2 of 2 nodes displayed)

0.009
1
0.006
2008-01-20 02:54:49
200

Inspector Transaction Stack

EdgeProxy/URI=//ServerPort=8090	
Property	Value
User Name	
ARM Return Code	Success
Name	EdgeProxy/URI=//ServerPort=8090
Average Duration	0.009
CACHE_HIT	1
Duration	0.006
Start Time	2008-01-20 02:54:49
HTTP_STATUS	200





## Summary

- Use tools
  - ▶ PageDetailer to see what's happening
  - ▶ OpenSTA to see performance impact
  - ▶ Timers to see where to concentrate your work
  - ▶ Rational Performance Tester for final results
- Edge Components are always needed
- Portal by itself is very fast
- Web Content Management without caching is very slow
- Caching Proxy is blazingly fast
- **Web Content Management project without caching strategy = trouble**





# Metadata Generator: Random Caching Time

```
package com.ibm.pl.fz.randomcacher;

import javax.servlet.http.HttpServletResponse;
import com.ibm.websphere.servlet.cache.CacheConfig;
import com.ibm.websphere.servlet.cache.FragmentInfo;
import com.ibm.websphere.servlet.cache.MetaDataGenerator;
import com.ibm.websphere.servlet.cache.ServletCacheRequest;
public class RandomCacher implements MetaDataGenerator {

    public void initialize(CacheConfig cc) {
    }

    public void setMetaData(ServletCacheRequest scr, HttpServletResponse hsr)
    {
        FragmentInfo fi = scr.getFragmentInfo();
        int timeout = fi.getTimeLimit();
        int rtimeout = timeout+(int)Math.round(timeout*Math.random());
        fi.setTimeLimit(rtimeout);
    }
}
```



# Handling of Web Content Management GLOBAL\_CONTEXT – view.jsp

```
// Get listensTo attribute from WCM config
String listensTo = (String) portletRequest.getData().getAttribute("WCM_LISTENS_TO");
if(listensTo==null)
    listensTo = (String) portletRequest.getPortletSettings().getAttribute("WCM_LISTENS_TO");

String contextPath = null;
String portlet = portletRequest.getParameter("WCM_PORTLET");
if(listensTo.equals("WCM_LINKING_SELF") && portlet != null &&
    portlet.equals(portletResponse.encodeNamespace("WCM"))) {
    contextPath = (String)
    portletRequest.getParameter(portletResponse.encodeNamespace("WCM_CONTEXT"));
}

if(contextPath==null && !listensTo.equals("WCM_LINKING_NONE")){
    contextPath = (String) portletRequest.getParameter("WCM_GLOBAL_CONTEXT");
    if(contextPath!=null)
        portletRequest.getSession().setAttribute("WCM_CONTENT_CONTEXT", contextPath);
}
if(contextPath==null
    contextPath = (String) portletRequest.getSession().getAttribute("WCM_CONTENT_CONTEXT");
if(contextPath==null)
    contextPath = (String) portletRequest.getData().getAttribute("WCM_CONTENT_CONTEXT");
if(contextPath==null)
    contextPath = (String) portletRequest.getPortletSettings().getAttribute("WCM_CONTENT_CONTEXT");
portletRequest.setAttribute("WCM_GLOBAL_CONTEXT", contextPath);
```



## Getting current page ID – view.jsp

```
String customID = (String)
    portletRequest.getPortletSettings().getAttribute("CID");
if(customID==null || customID.equals(""))
    customID = (String) portletRequest.getData().getAttribute("CID");
if(customID==null || customID.equals("")){
    portletRequest.setAttribute("UID", portletResponse.encodeNamespace(""));
    ModelUtil util = com.ibm.wps.model.ModelUtil.from(request);
    NavigationNode node =
        (NavigationNode)util.getNavigationSelectionModel().getSelectedNode();
    MetaData iMetaData = ((MetaDataProvider)node).getMetaData();
    Object internalPageName = iMetaData.getValue("InternalPage");
    if (internalPageName != null)
        portletRequest.setAttribute("PORTAL_PAGE", internalPageName);
} else {
    portletRequest.setAttribute("CID", customID);
}
```



# Cache Dependencies

```
<cache-id>
  <component id="UID" type="attribute"><required>true</required></component>
  <component id="ANON" type="attribute"><required>true</required></component>
  <component id="WCM_GLOBAL_CONTEXT" type="attribute"><required>true</required></component>
  <component id="PAGE" type="attribute"><required>false</required></component>
  <timeout>86400</timeout>
  <metadatagenerator>com.ibm.pl.fz.randomcacher.RandomCacher</metadatagenerator>
</cache-id>
<cache-id>
  <component id="CID" type="attribute"><required>true</required></component>
  <component id="ANON" type="attribute"><required>true</required></component>
  <timeout>864000</timeout>
  <metadatagenerator>com.ibm.pl.fz.randomcacher.RandomCacher</metadatagenerator>
</cache-id>
<dependency-id>CID
  <component id="CID" type="attribute"><required>true</required></component>
</dependency-id>
<dependency-id>CTX
  <component id="WCM_GLOBAL_CONTEXT" type="attribute"><required>true</required></component>
</dependency-id>
<dependency-id>PP
  <component id="PORTAL_PAGE" type="attribute"><required>true</required></component>
</dependency-id>
```



# Network Dispatcher Configuration

```
dscontrol executor start
dscontrol executor set clientgateway 192.168.1.1
dscontrol executor set staletimeout 30
dscontrol executor set fintimeout 30
dscontrol highavailability heartbeat add lb01 lb02
dscontrol highavailability backup add primary manual 3333
dscontrol cluster add www.cluster.com
dscontrol port add www.cluster.com:80 method mac porttype tcp reset yes
dscontrol server add www.cluster.com:80:cp01
dscontrol server add www.cluster.com:80:cp02
dscontrol server add www.cluster.com:80:http01
dscontrol server add www.cluster.com:80:http02
dscontrol rule add www.cluster.com:80:primary type true priority 1
dscontrol rule add www.cluster.com:80:backup type true priority 2
dscontrol rule useserver www.cluster.com:80:primary cp01
dscontrol rule useserver www.cluster.com:80:primary cp02
dscontrol rule useserver www.cluster.com:80:backup http01
dscontrol rule useserver www.cluster.com:80:backup http02
dscontrol executor configure www.cluster.com
dscontrol manager start
dscontrol manager smoothing 10
dscontrol advisor start http www.cluster.com:80
dscontrol advisor interval http www.cluster.com:80 3
```



# Content Based Router Configuration

```
cbrcontrol executor start
cbrcontrol cluster add www.cluster.com
cbrcontrol port add www.cluster.com:80
cbrcontrol server add www.cluster.com:80:http01 \
  advisorrequest "GET /wps/portal HTTP/1.1\r\nHost: www.cluster.com" \
  advisorresponse "HTTP/1.1 200 OK"
cbrcontrol server add www.cluster.com:80:http02 \
  advisorrequest "GET /wps/portal HTTP/1.1\r\nHost: www.cluster.com" \
  advisorresponse "HTTP/1.1 200 OK"
cbrcontrol rule add www.cluster.com:80:http1 type true priority 1
cbrcontrol rule add www.cluster.com:80:http2 type true priority 2
cbrcontrol rule useserver www.cluster.com:80:http1 http01
cbrcontrol rule useserver www.cluster.com:80:http2 http02
cbrcontrol manager start
cbrcontrol manager smoothing 10
cbrcontrol advisor start http www.cluster.com:80
cbrcontrol advisor interval http www.cluster.com:80 3
```



# Caching Proxy Configuration

```
Proxy /wps/* http://www.cluster.com/wps/* :80
Redirect / http://www.cluster.com/wps/portal :80
CacheMinHold http://* 30 minutes
AggressiveCaching http://*
CacheQueries Always http://* 30 minutes
NoCaching http://*/wps/myportal*
NoCaching http://*/wps/portal*
CacheExpiryCheck off
ServerConnPool on
ProxyPersistence on
CacheMemory 512 M
MaxContentLengthBuffer 500 K
KeepExpired on
```

Caching Proxy configuration contains lots of comments. You need to locate proper places for above configuration directives.

Remember to remove comments from CBR plugin initialization lines!



# HTTP Server Configuration

```
LoadModule headers_module modules/mod_headers.so  
Header set Cache-Control public
```

This will make all items cacheable.

Remember to put **NoCaching** directives to caching proxy configuration for dynamic pages!



