

IBM Commerce

Performance Engineering



Klaus Nossek, Digital Experience Lab

mailto:klaus_nossek@de.ibm.com

Hermann Hübler, Digital Experience Lab Services

<mailto:huebler@de.ibm.com>

Digital Experience Meet the Lab

14.-15. Juni 2016



Agenda

- How Customers benefit from the Performance Work in the Lab
- DX Performance Engineering in the Lab
- Key Performance Driver - Caching
- DX Performance Guidelines
- Performance from the services perspective

How do Customers benefit from the Performance Work in the Lab

The Goals of the Performance Team in Böblingen, Dublin and Raleigh:

- The Performance Team runs performance benchmarks and measurements for portal releases and new features before they ship out to the customers to make sure that they perform well
- The team publishes performance papers that are guidelines for performance planning and tuning
- The team supports customers to prevent and solve performance issues

DX Performance Engineering

Performance Planning

- Planning of performance work items, design reviews, code reviews, performance environment, performance scenarios, performance goals, test execution, documentation

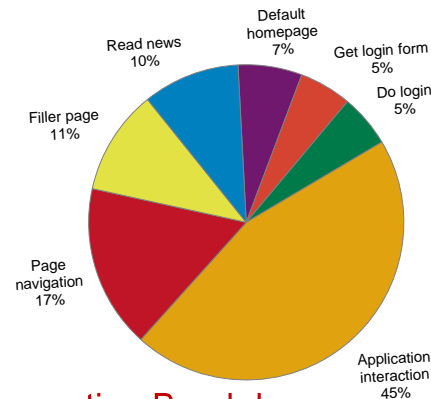
Define Performance Benchmark Scenarios

- Define multiple scenario which are intended to demonstrate the performance characteristics of WebSphere Portal in various use cases
- Design performance scenarios with two main objectives:
 - Demonstrate the maximum load for a system
 - Demonstrate that the maximum load can be sustained

Standard Portal Benchmark Scenarios

A portal benchmark models the viewing of content by users in a corporate intranet environment with:

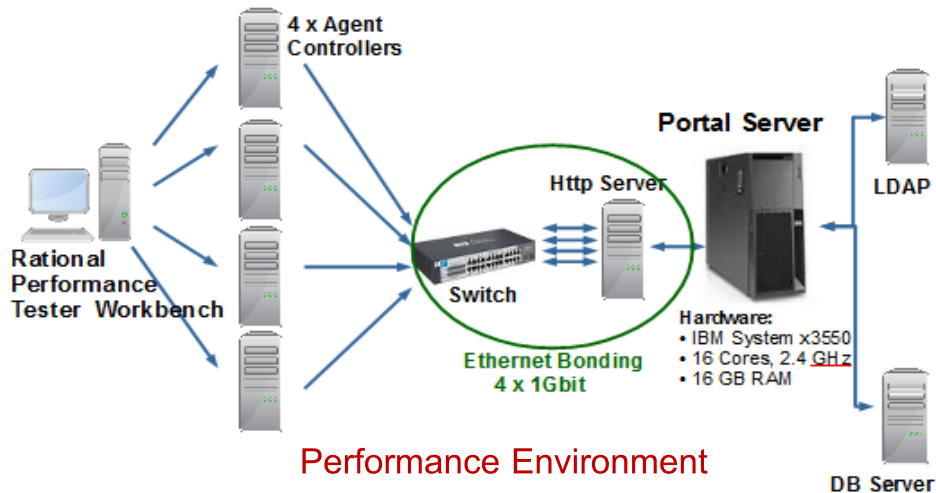
- 100,000 registered user
- 50,000 content items
- 1,000 pages / labels
- 70 portlet applications



Transaction Breakdown

Key Performance Metrics:

- max Throughput / max Concurrent Users
- acceptable Response Time (1sec/5sec/...sec)

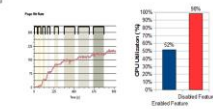
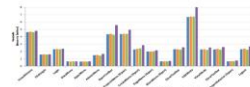
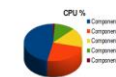
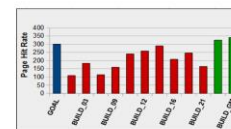
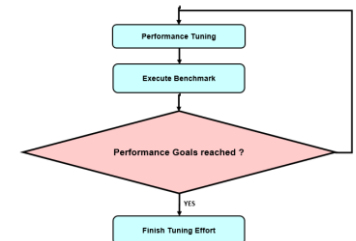


Performance Environment

DX Performance Engineering

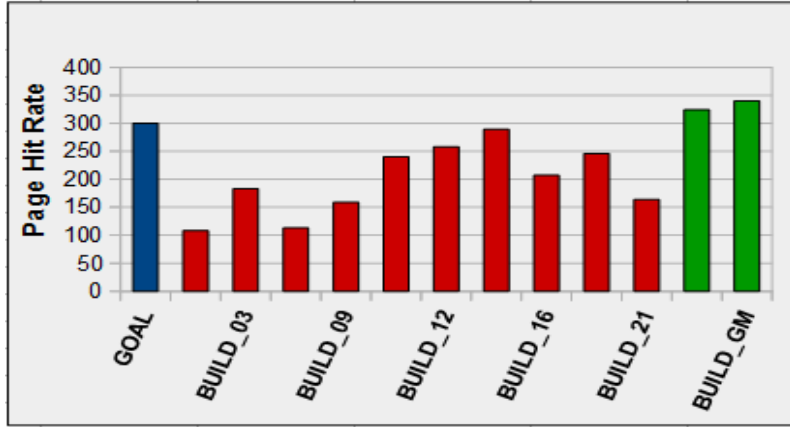
Performance needs to be managed:

- Start performance tests early in the development phase
- Run performance tests continuously
- The elimination of performance bottlenecks is an iterative process
- Automate performance test
- Use dashboards for performance reporting

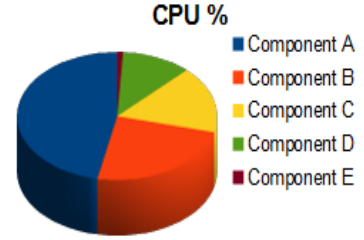
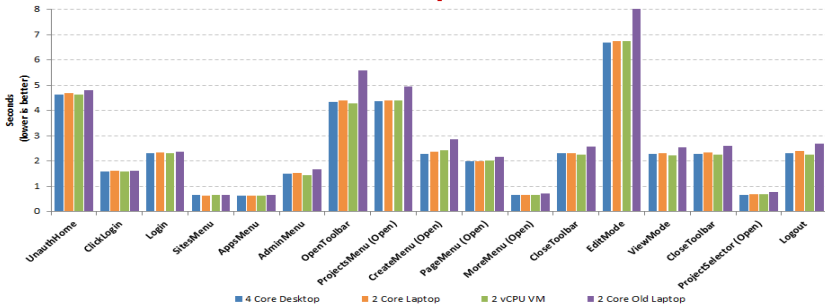


Benchmark Reports

max Throughput with acceptable Response Time

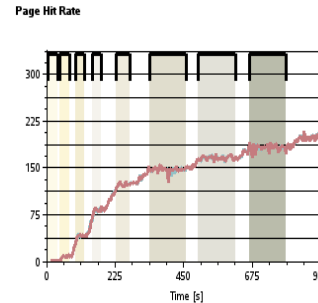


Browser Response Time

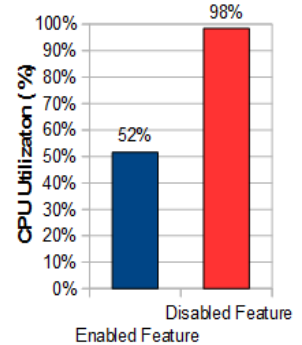


Breakdown by Components

Monitoring



Page Hit Rate



CPU Utilization © 2016 IBM

Troubleshoot Performance Issues

CPU

CPU	Name
2,011,712	org/apache/jsp/_Control_jspService()
1,740,552	org/apache/jsp/_Default_jspService()
1,408,144	com/ibm/wps/composition/model/impl/NodeFactory.getPortletNode()
1,132,200	com/ibm/wps/engine/FixedURL.<init>()
1,063,296	com/ibm/wps/ae/impl/PermissionCollectionImpl.<init>()
1,039,712	com/ibm/wps/state/accessors/collections/MapOnNode.putAll()
720,336	com/ibm/wps/ae/impl/PermissionCollectionImpl.putActions()
486,648	com/ibm/wps/state/utills/CharWriter.toString()
404,08	com/ibm/wps/ae/impl/PermissionCollectionImpl.implies()

Method Profiling:

Lock Contention

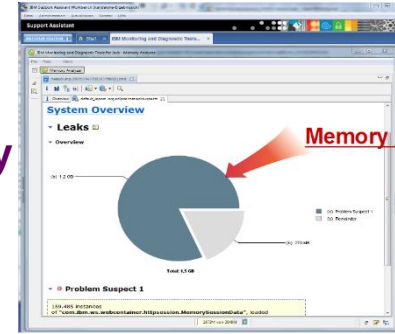
IBM Thread and Monitor Dump Analyzer for Java

Name	State	Method	SL
Thread-48	Parked	sun.misc.Unsafe.park(Native Method)	5
Thread-56	Parked	sun.misc.Unsafe.park(Native Method)	5
Thread-65	Parked	sun.misc.Unsafe.park(Native Method)	5
Non-deferrable Alarm-0	Parked	sun.misc.Unsafe.park(Native Method)	4
Thread-2	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-13	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	30
WebContainer-15	Blocked	org.eclipse.wcm.infocore.impl.PackageRegistryImpl.getPackage(EPackageRegistryImpl)	30
WebContainer-18	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-06	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-12	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	272
WebContainer-43	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	272
WebContainer-36	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	296
WebContainer-42	Blocked	org.eclipse.wcm.infocore.impl.PackageRegistryImpl.getPackage(EPackageRegistryImpl)	144
WebContainer-32	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	296
WebContainer-07	Blocked	org.eclipse.wcm.infocore.impl.PackageRegistryImpl.getPackage(EPackageRegistryImpl)	144
WebContainer-27	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-21	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-11	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	30
WebContainer-49	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-23	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	288
WebContainer-59	Blocked	org.eclipse.wcm.infocore.impl.PackageRegistryImpl.getPackage(EPackageRegistryImpl)	128
WebContainer-33	Blocked	org.eclipse.wcm.infocore.impl.PackageRegistryImpl.getPackage(EPackageRegistryImpl)	300
WebContainer-0	Blocked	java.lang.ClassLoader.loadClass(ClassLoader.java:990(Compiled Code))	0

Thread Dump:

Memory

Memory Dump:




IO

Monitoring: http traffic

Fiddler Web Debugger

#	Result	Protocol	Host	URL
1	200	HTTP	wps188:10039	/wps/portal/
2	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!NA9YQlEmeIodsl8TDA(sp/mashup:ra:collection?offset=0&offset=15&themeID=Z1_0000000000000A0BR2B3C
3	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!NA9YQlEmeIodsl8TDA(sp/mashup:ra:collection?themeID=Z1_0000000000000A0BR2B300QC68loca=en&scale=d
4	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!spPqWEEv-35NnRexJ_xQ/mashup:ra:collection?themeID=Z1_0000000000000A0BR2B300QC68loca=en&scale=c
5	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!GZJ78UJ5gOAgzUJ0M4Lw/mashup:ra:collection?themeID=Z1_0000000000000A0BR2B300QC68loca=en&scale=c
6	200	HTTP	wps188:10039	/wps/PA_Filler_23/Images/ads/ads.js
7	200	HTTP	wps188:10039	/wps/PA_Filler_23/Images/ads/ads2.js
8	200	HTTP	wps188:10039	/wps/PA_Filler_23/Images/ads/ads14.js
9	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!spPqWEEv-35NnRexJ_xQ/dav/fs-type1/themes/Portal8_5/css/images/master.png
1	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!spPqWEEv-35NnRexJ_xQ/dav/fs-type1/themes/Portal8_5/css/images/favicon.ico
1	200	HTTP	wps188:10039	/wps/contenthandler/!ut/p/digest!spPqWEEv-35NnRexJ_xQ/dav/fs-type1/themes/Portal8_5/css/images/loading.gif
1	200	HTTP	wps188:10039	/wps/portal/!ut/p/1/D4_5jPCPKssyOxPLM4M0MAR7j0zJ_S1nH083A28_b39LAWcUJ3DXN0Mgw18gwz0w8EKDHAARwP9K6Lo_AqC5tCg8RUFUHG
1	200	HTTP	wps188:10039	/wps/PA_Filler_12/Images/btn_on.gif
1	200	HTTP	wps188:10039	/wps/PA_DieRoller/Images/Dice.jpg

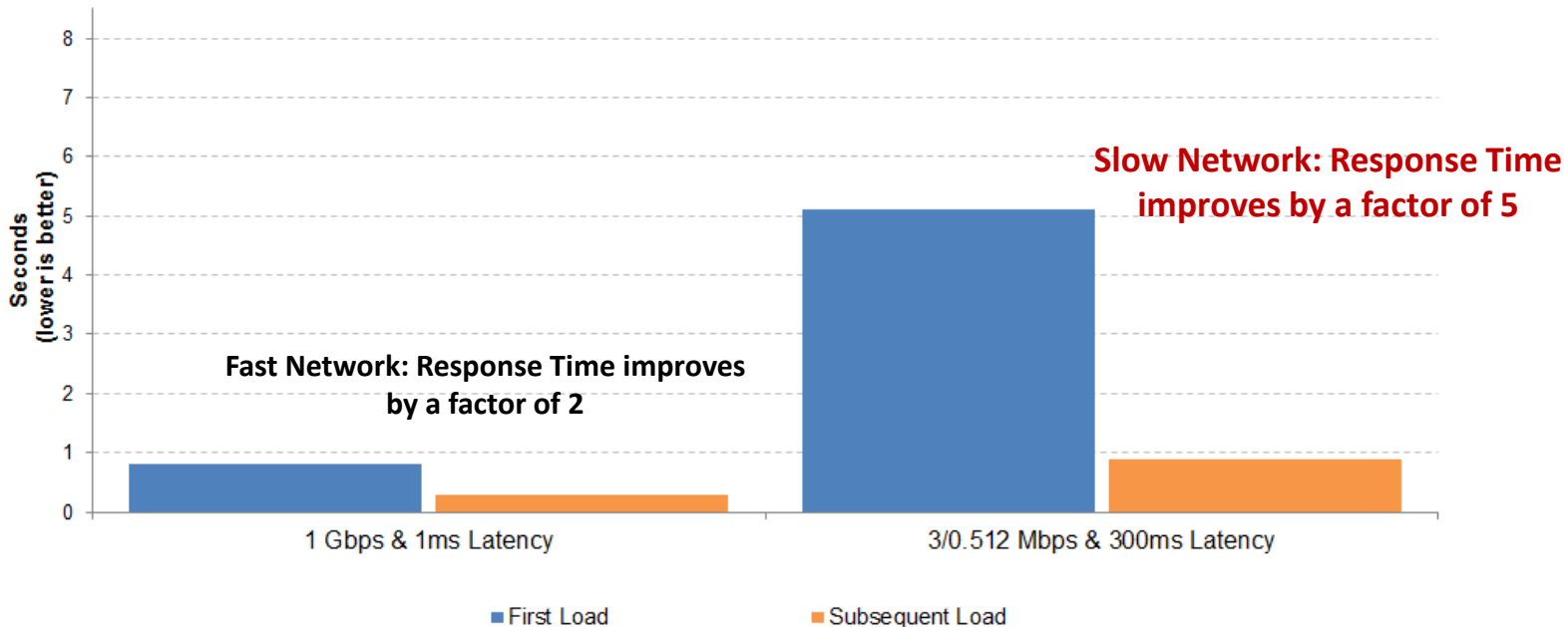
Key Performance Drivers in WebSphere Portal

- Environment: HW, OS, Network, DB, LDAP, Http Server, Backend
- Application Complexity: Pages, Theme, Portlets
- Caching 
- Memory
- JVM Tuning & WebSphere Tuning

Benefit of Caching – Example from the Lab: Slow vs Fast Network

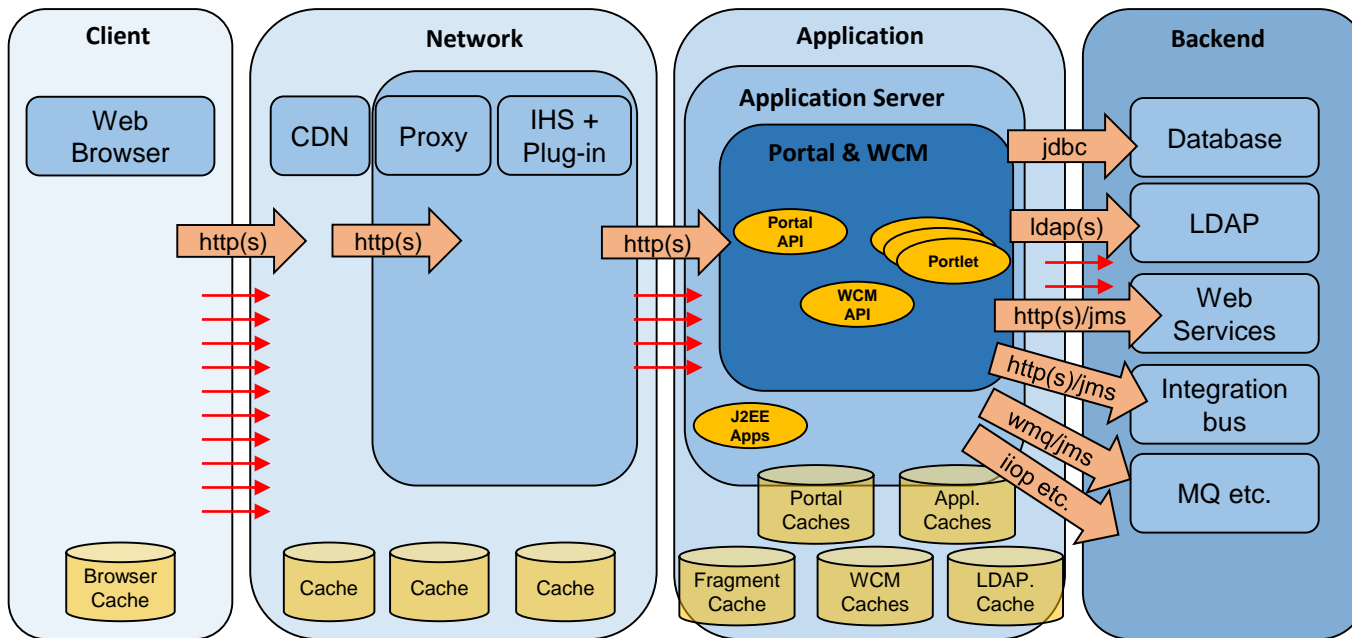
Portal 8.5, Welcome Page with Firefox 24

Single User Browser Response Time



First Load without Caching – Subsequent Load with Caching

Caching Architecture



HTTP responses

- Images
- CSS
- Java Script etc.

Private & Public

HTTP responses

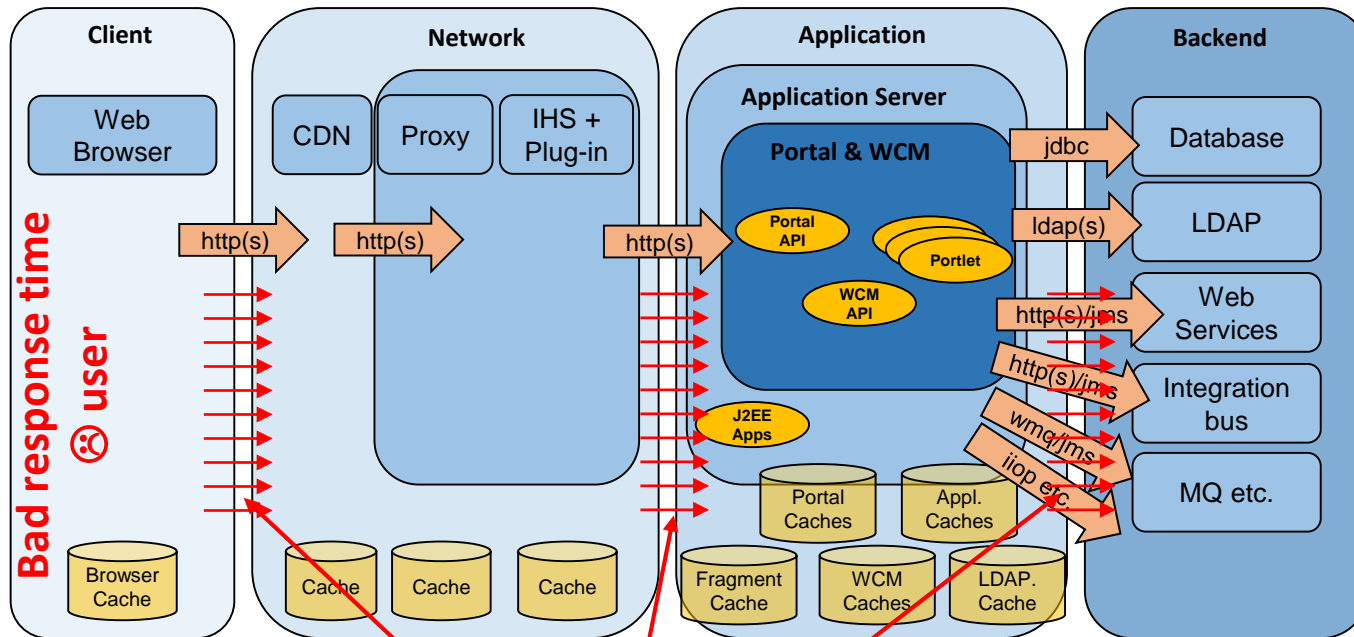
- Images
- CSS
- Java Script etc.

Public

- HTTP Responses from a portlet
- Internal Portal & WCM data
- Backend query results
- Custom application data

Caching Architecture

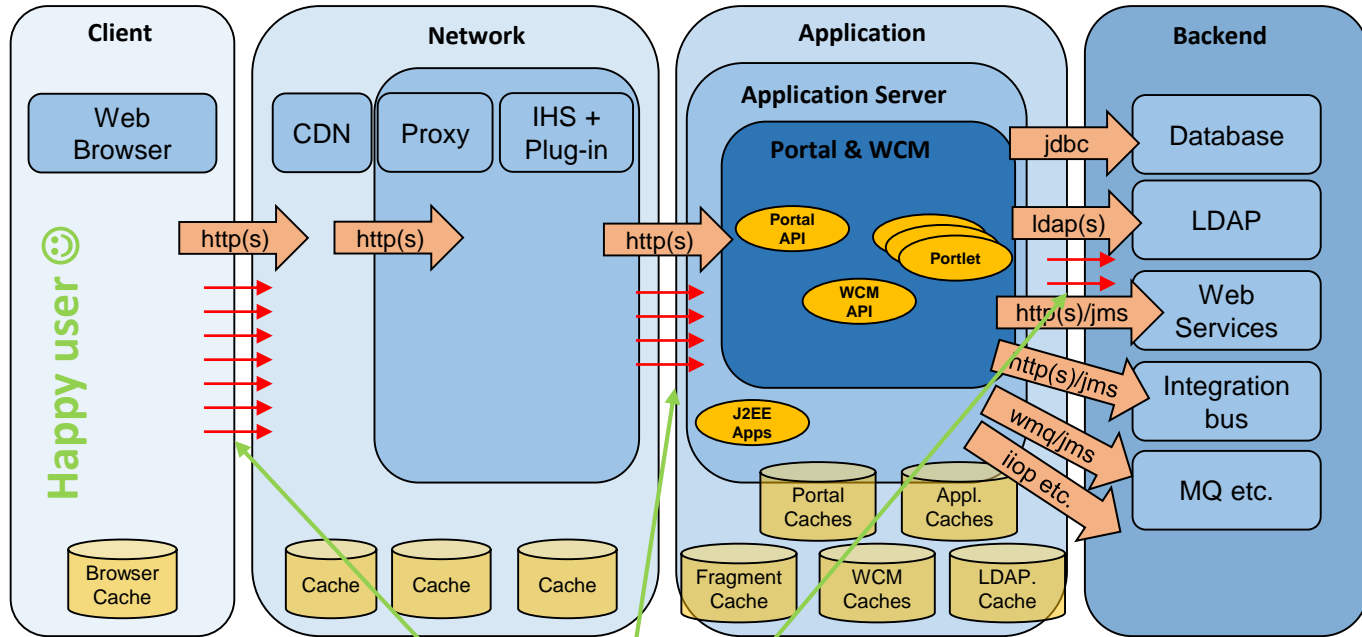
No Caching



High traffic

Caching Architecture

Using Caching

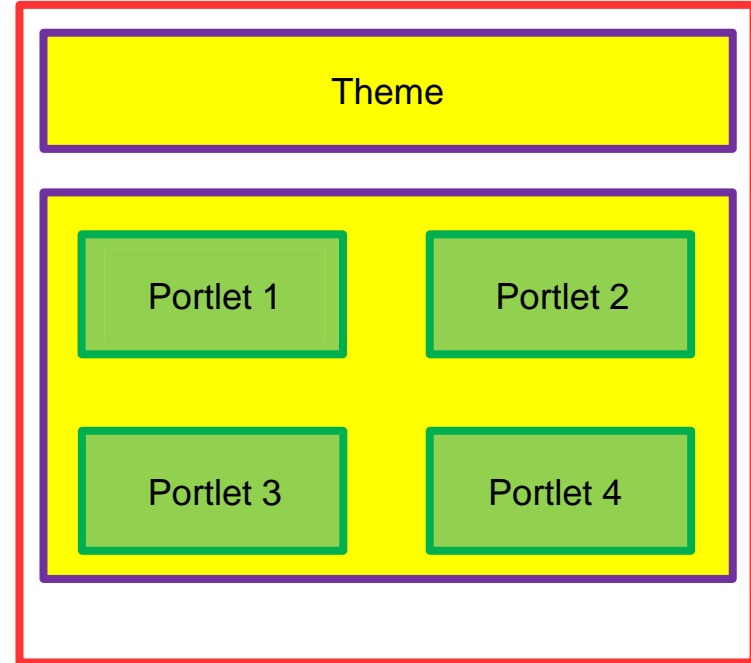


Less traffic

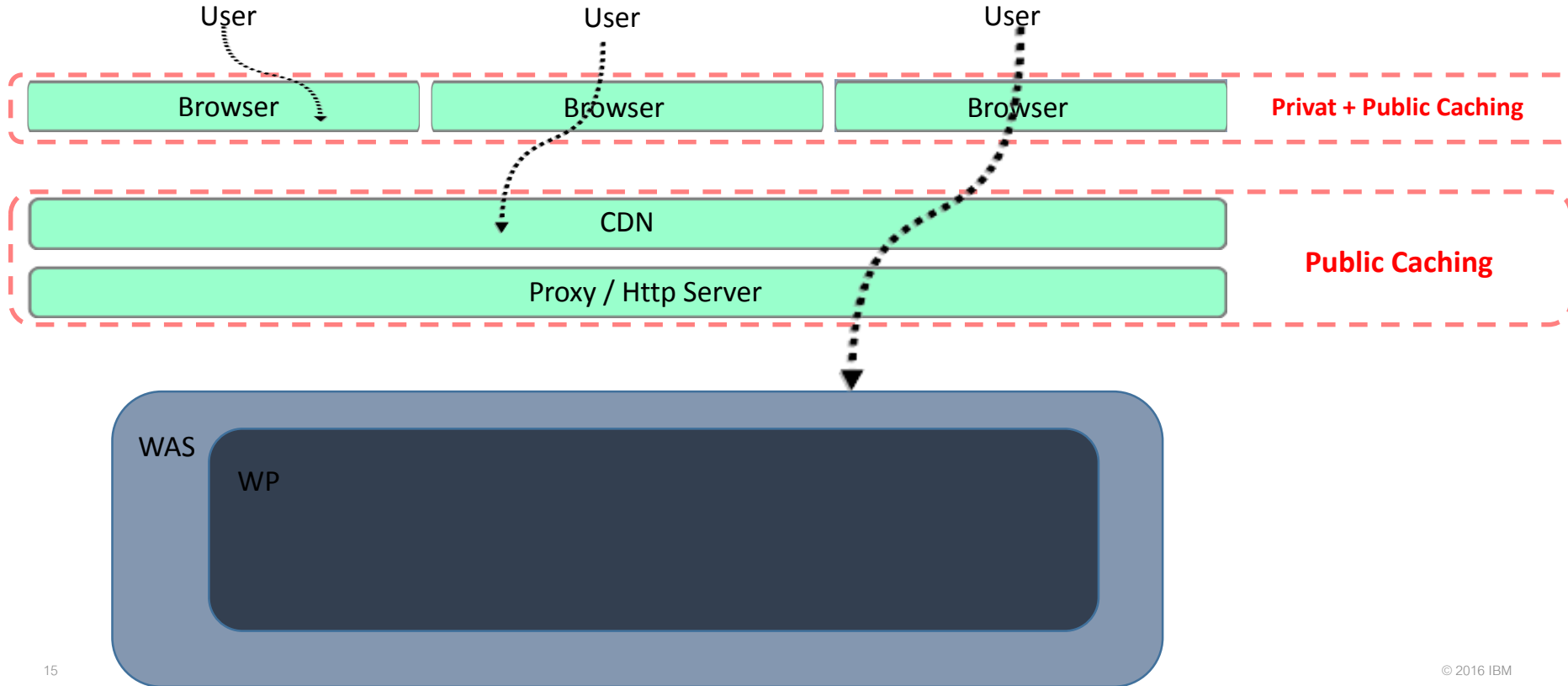
Page Caching -> Browser and Network Caches

- Consider **Page**, **Theme**, **Portlet** and Global Settings for Page Caching
- Since portlets default to no caching, pages will not be cached by just changing the page cache settings
- What happens if a page has a portlet with conflicting configuration ?
 - Lowest common denominator wins
 - Everything on page must be SHARED for public
 - Minimum value of all expirations will be used for max-age

Page

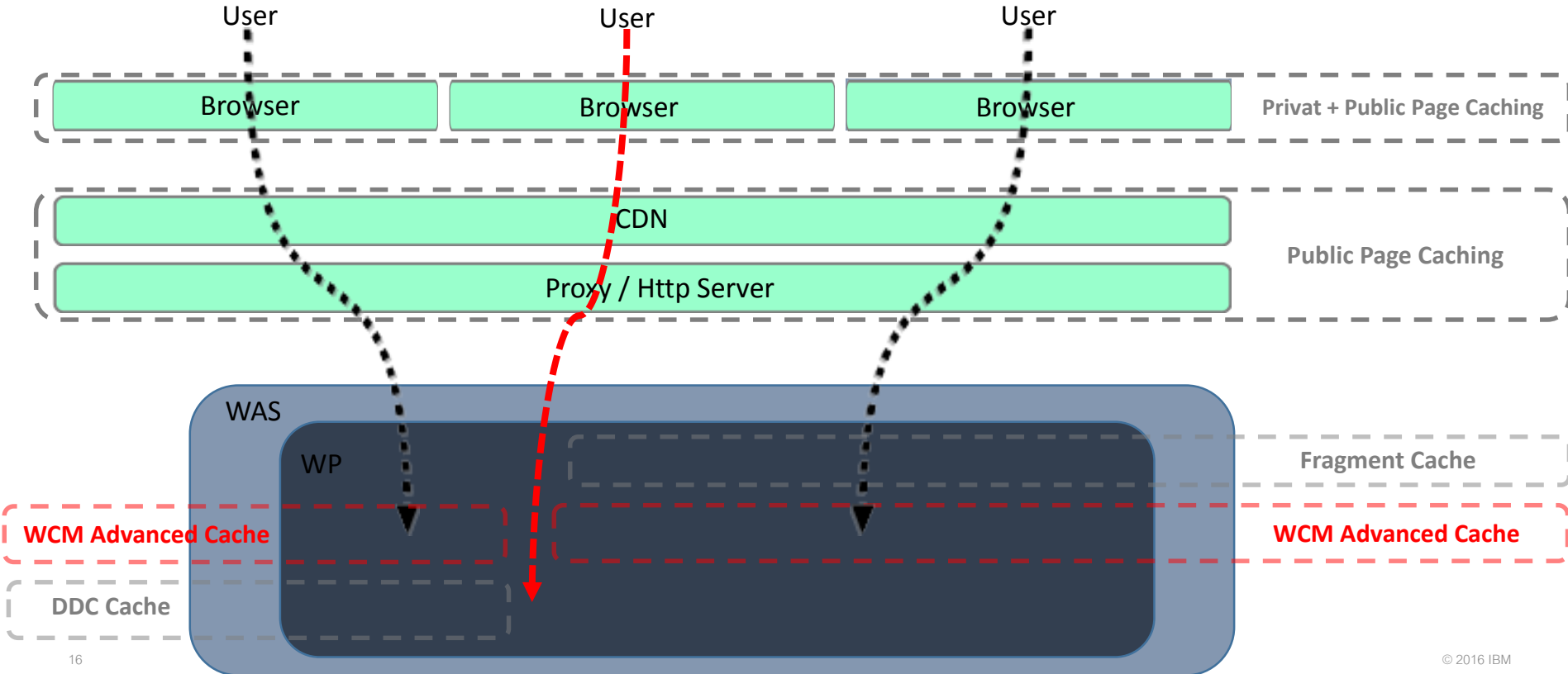


Page and Static Content Caching - in remote - Browser and Network Caches



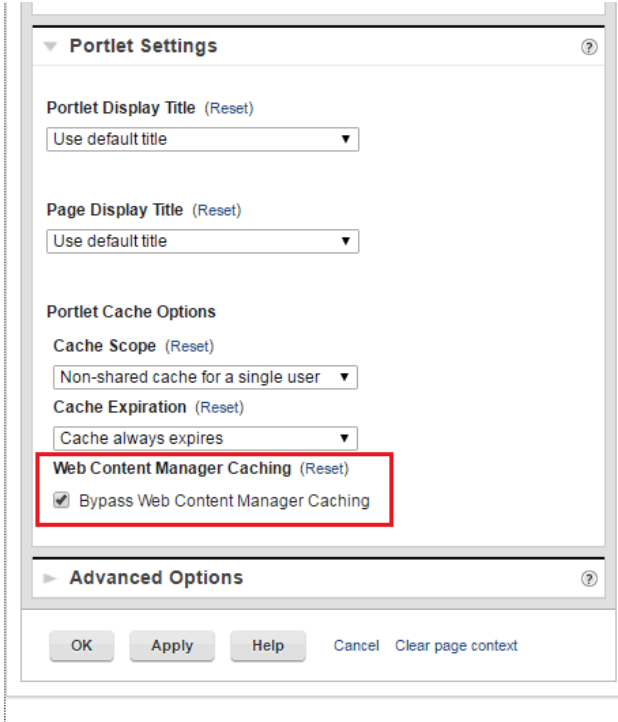
WCM Rendering - **NEW WP 8.5 CF11**: If WCM Advanced Cache is enabled for all Portlets

=> bypass WCM Advanced Cache for dynamic Portlets



WCM Rendering - **NEW WP 8.5 CF11**: => bypass WCM Advanced Cache for dynamic Portlets

There is a new flag in the Configure and Edit Shared Settings mode of the Web Content Viewer portlet

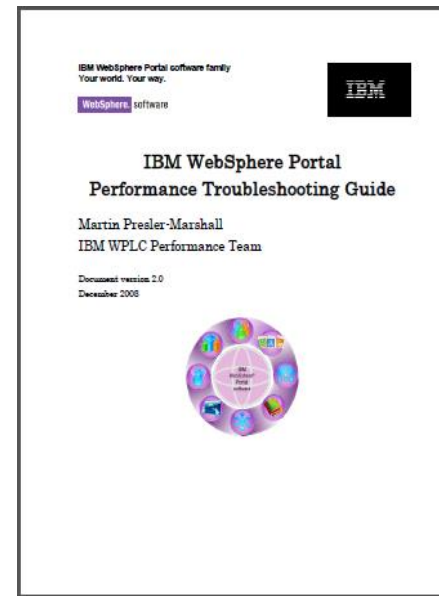
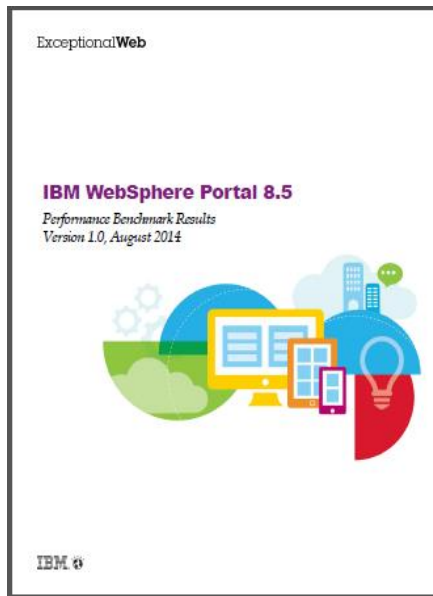
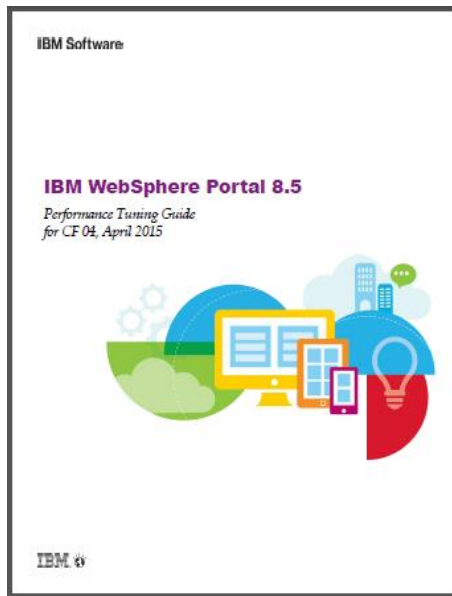


The screenshot shows the 'Portlet Settings' dialog box for the Web Content Viewer portlet. The settings are organized into sections:

- Portlet Settings** (with a help icon):
 - Portlet Display Title** (Reset): Use default title (dropdown menu)
 - Page Display Title** (Reset): Use default title (dropdown menu)
 - Portlet Cache Options**
 - Cache Scope** (Reset): Non-shared cache for a single user (dropdown menu)
 - Cache Expiration** (Reset): Cache always expires (dropdown menu)
 - Web Content Manager Caching** (Reset): Bypass Web Content Manager Caching (highlighted with a red box)
- Advanced Options** (with a help icon):

At the bottom of the dialog, there are buttons for **OK**, **Apply**, **Help**, **Cancel**, and **Clear page context**.

DX Performance Guidelines



NEW: Performance Tuning Guide for WP 8.5 CF 10 Mai 2016

Performance from the services perspective

- Performance in the field
- The performance-map
- Performance tuning
- Load testing
- Operational aspects of performance

Principle for complex (technical) systems

There are known knowns; there are things we know we know.

We also know there are known unknowns; that is to say we know there are some things we do not know.

But there are also unknown unknowns – there are things we do not know we don't know

U.S. Secretary of Defense Donald Rumsfeld, 2002/02/12

Performance I

- Needs to be managed
 - Iterative process
 - Start early in the development/implementation phase
- Depends on every component involved in performing work
 - Principle of the weakest link
- Better performance means
 - Supporting more users
 - Lower hardware and maintenance cost
 - Increased customer satisfaction

Performance II

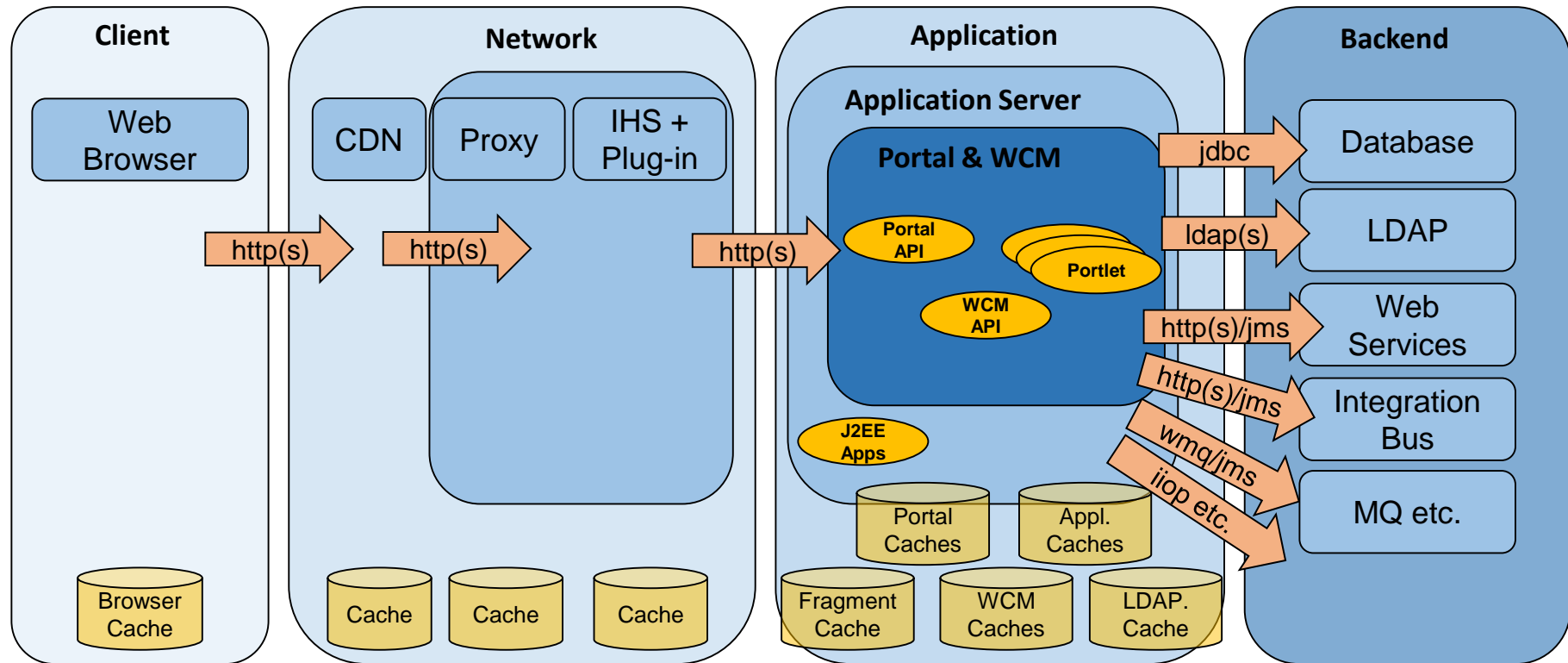
- Two main areas
 - Application performance
 - Middleware and infrastructure performance
 - ➔ Focus of this presentation in is this area

Performance in a Digital Experience (DX) environment

- DX environments can become quite complex
 - Typically consist of multiple systems such as reverse proxy servers, HTTP servers, WebSphere Application Servers, database servers etc.
 - Mostly clustered or farmed
 - Different types require different tuning
 - Authoring and rendering systems
- Need to track performance on every system
 - Client, Network, Operating system, JVM, Database, Caching, etc. etc.
- Use the [WebSphere Portal Tuning Guide](#) as a starting point
- ConfigEngine task: [tune-initial-performance](#)

Need to balance all these components
for optimal performance

The performance map



Performance tuning I

■ Two main areas

- Middleware and infrastructure performance tuning
 - Method of changing parameter settings to achieve optimal performance
 - Remember the weakest link principle → need to balance the resources in the environment for optimal results

→ Focus of this presentation in is this area

■ Application tuning

- Developers need to provide efficient code - of course they always do ;)
- Application complexity: Pages, Theme, Portlets, WCM templates etc.

■ You can't tune out of a poorly written application

- Efficient application code is a prerequisite for good end user performance

Performance tuning II

■ Before you start

- Make sure you understand the environment/architecture
 - To avoid unknown unknowns in the architectural level
 - Including systems your DX environment connects to

Architectural and system context diagrams are helpful (if available)

- Make sure you know the most critical transactions/pages
- Be aware that tuning is application/environment specific
- Make sure that you have monitoring/logging in place
 - to reduce the known unknowns and be prepared to find unknown unknowns

Performance tuning III

- Disable what you don't need
 - For example session persistence
- Define performance goals
 - Throughput
 - Response time
 - Etc.

Make sure these goals are clear, meaningful and measurable

Performance tuning – The big plan

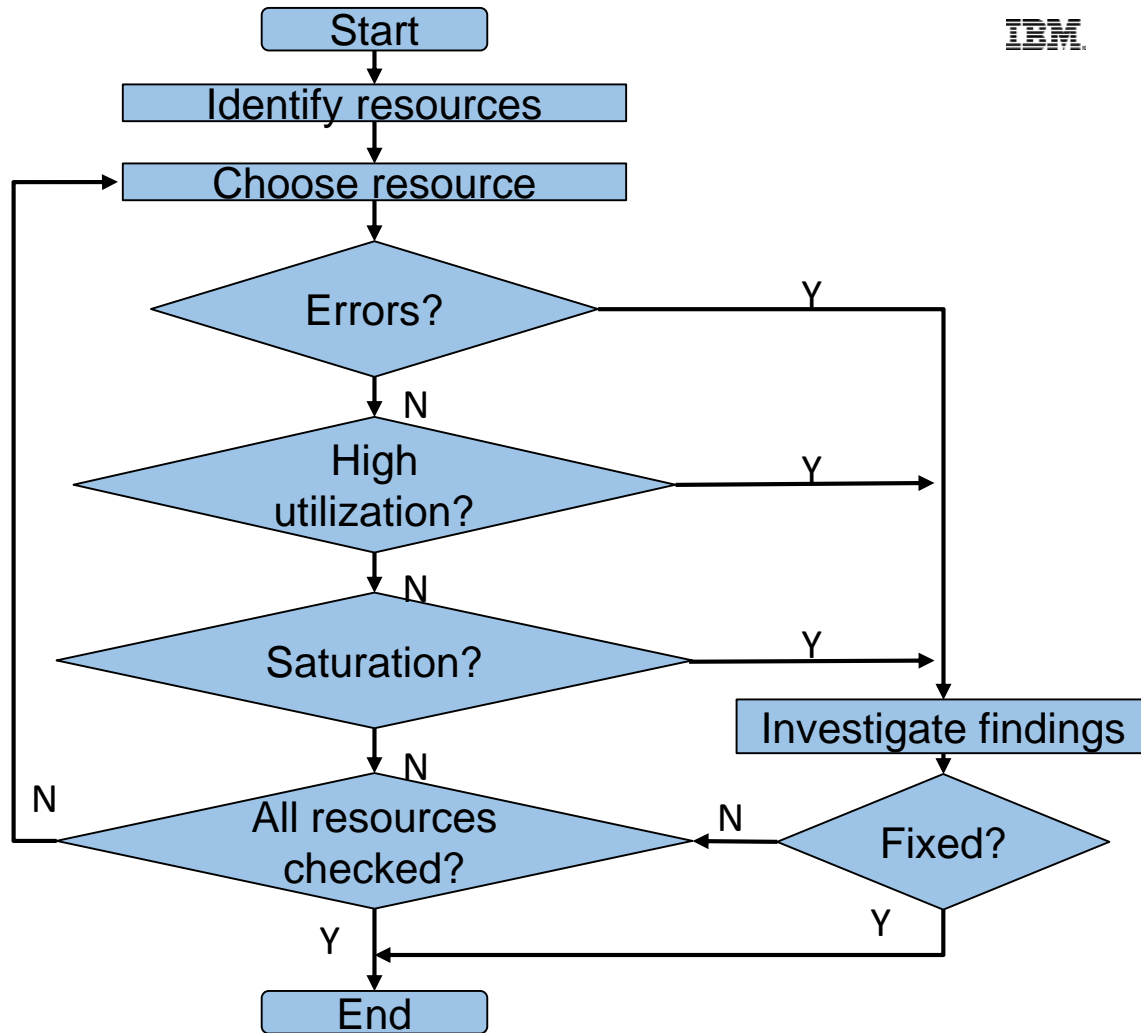
1. Fully understand the environment, the architecture and the data flow
2. Ensure you have sufficient hardware resources
3. Ensure your hardware resources are configured efficiently
4. Ensure your OS resources are configured efficiently
5. Ensure your network resources are configured efficiently
6. Use offloading and accelerators whenever possible/available
7. Stay current with fixes and maintenance packages
8. Tune your JVM
9. Tune WebSphere's queuing framework (HTTP server, plug-in, thread pools, connection pools etc.)
10. Tune caches – Browser, Proxy, HTTP Server, WAS, Portal, WCM and LDAP
11. Tune your databases
12. Use monitoring tools to track resource utilization and to detect trends
13. Set up proper logging across the request chain
14. Use tracing, sampling and profiling tools to identify bottlenecks

USE method I

- Utilization, Saturation and Errors are checked for every resource
 - Resource – all server functional components. Includes hardware and software
 - Utilization – percentage of time the resource is busy providing service work
 - Saturation – degree to which the resource has work to do but can't service the work → waiting on a queue
 - Errors – the number of errors logged
- Should be used early in the performance investigation
- Focuses on resources – not on tools
 - Supports getting a full list of resources and what to monitor → avoids unknown unknowns

USE method II

- Iterates over all resources and checks each resource for USE
 - Checks for errors first (easy to find and fixed) – expressed in numbers
 - Check for utilization – expressed in percentage
 - Check for saturation – queue length



Load testing - overview

- Load testing is part of capacity planning
- Need to test code to ensure that
 - It runs without errors under load (race-conditions, concurrent modifications etc.)
 - It does not cause leaks of any kind
 - It meets performance criteria

Load test types

- Load test types
 - Perform Load Test
 - Shows that the application can support expected transaction volumes and still meets expected response times
 - Stress Test
 - Analyze the breaking point of the system with the expected response times
 - Endurance Test
 - To demonstrate that the code runs for a long period without breaking (memory leaks, file system fill-ups etc.)

Load test planning

- Key factors
 - Number of users and transaction rate
 - Transaction mix
 - Dataset size and contents
 - Think time
 - Session handling
- Results must be
 - Repeatable and reliable (ignore warm-up data)

Load test verification

- Results need to be checked against production data
 - Transaction mix
 - Resource utilization
 - Performance results (response time, throughput)
- Multiple tools available
 - Rational Performance Tester
 - Apache Jmeter
 - Etc.

References I

- WebSphere Application Server performance tuning community
 - <https://w3-connections.ibm.com/communities/service/html/communityview?communityUuid=6e0cd8fa-e743-4540-bf5e-a182a1aa5f6a&successMessage=label.action.confirm.community.join>
- WebSphere Application Server performance cookbook
 - Public version: <https://publib.boulder.ibm.com/htpserv/cookbook/>
 - Internal version: https://w3-connections.ibm.com/wikis/home?lang=en-us#!/wiki/W821910dd75b9_434e_8b82_477706b8118d
- WebSphere Application Server Forum
 - <https://w3-connections.ibm.com/forums/html/forum?id=11111111-0000-0000-0000-000000001245>
- WebSphere Application Server – recommended fixes
 - <http://www-01.ibm.com/support/docview.wss?uid=swg27004980>
- Health Center in headless node
 - https://www.ibm.com/developerworks/community/blogs/kevgrig/entry/running_ibm_java_healthcenter_in_headless_mode48?lang=en
- IHS mod_mpmstats
 - https://publib.boulder.ibm.com/htpserv/manual70/mod/mod_mpmstats.html

References II

- WebSphere Portal Tuning guides
 - <https://www-10.lotus.com/ldd/portalwiki.nsf/xpViewCategories.xsp?lookupName=Performance%20Tuning%20Guides>
- Caching theme objects
 - http://www-01.ibm.com/support/knowledgecenter/SSHRKX_8.5.0/mp/dev-theme/themeopt_mod_adminmod.dita?lang=en
 - http://www-01.ibm.com/support/knowledgecenter/SSHRKX_8.5.0/mp/admin-system/mash_webdav_store.dita?lang=en
- WebSphere Portal and Web Content Management – recommended fixes
 - <http://www-01.ibm.com/support/docview.wss?uid=swg27007603>

References III

- Sven Sterblings blog posts on developer.ibm.com
 - <https://developer.ibm.com/digexp/blog/2016/06/03/a-little-lesson-on-why-you-should-care-about-proper-performance-testing/>
 - <https://developer.ibm.com/digexp/docs/docs/customization-administration/genperfguide/>
 - <https://developer.ibm.com/digexp/docs/docs/customization-administration/dxperftestsubpatterns/>

References IV

The experiences our team has gathered is published in the performance guidelines:

- Websphere Portal Performance Benchmark Results

<https://w3-03.ibm.com/tools/cm/iram/oslc/assets/E4A38100-7B02-F39C-A3D3-3870A7E26BB7/1.0>

- Websphere Portal Performance Tuning Guide

http://www-10.lotus.com/ldd/portalwiki.nsf/dx/IBM_WebSphere_Portal_V_8.5_Performance_Tuning_Guide

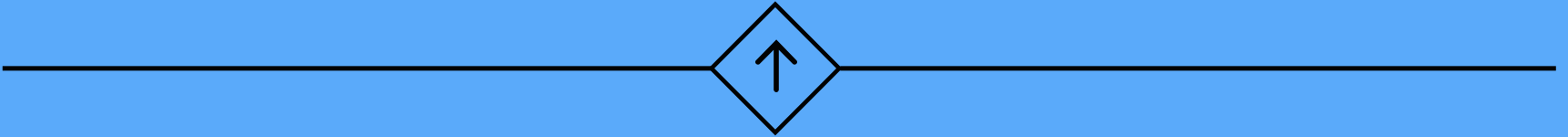
- Websphere Portal Performance Troubleshooting Guide

<http://www-10.lotus.com/ldd/portalwiki.nsf/dx/02092009093345AMWEBK46.htm>

- Websphere Portal Performance Wiki

<http://www-10.lotus.com/ldd/portalwiki.nsf/xpViewCategories.xsp?lookupName=Performance>

Backup



Operational aspects I

- Be prepared that things will sometimes go wrong over time
- Need problem determination data to isolate the root cause
 - Choose carefully what you are logging
- Need to prepare for problem data (which most likely requires OS configuration like file systems etc.)
- Maintain your log files
 - Rotate all logs
 - Via WebSphere Application Server tools
 - Via rotatelog daemon
 - Via HTTP Server rotatelog tool
 - Custom scripts
- Cleanup/archive of old log files should be automated
 - In line with customer's retention policy

Operational aspects II

- Collection of debug and trace data
 - Prepare file systems
 - Dumps and traces can be quite big
 - Cleanup/archive of old debug/trace data should be automated

Tools I

- Large set of tools available in support of different problem areas
 - From IBM or provides by 3rd party
- Choose tools based on the resource to investigate
 - Don't investigate only areas your tools support
- Each tool is targeting a specific area – JVM, Caches etc.
- [IBM Support assistant](#) is a suite of the most relevant tools
 - Current version v5 widely uses java web-start technology
 - You might need to edit your java.policy file to allow web-start
- You might need different tools depending on the JVM being used

Tools II

- Most commonly tools used on WebSphere Application Server
 - Admin Console
 - Tivoli performance viewer
 - Request metrics – in combination with an ARM server
 - GCMV – Garbage collection and memory visualizer to analyze verbose:gc logs
 - TMDA – Thread and Monitor Dump Analyzer to process javacore files
 - WebSphere Application Server Performance Tuning Toolkit (PTT) to monitor performance based on PMI
 - Health Center – Provides an indepth view into the JVM (profiling etc.)
 - Memory Analyzer – to analyze heap- and system dumps

Tools III

- Tools used in other areas
 - FireBug, Fiddler, HttpWatch, etc. to analyze browser performance, headers etc.
 - IBM Http Server – access.log to analyze response times at the HTTP server, Caching at the HTTP server
 - Cache Viewer Portlet/Extended Cache Monitor – to analyze WebSphere Application Server and Portal Cache
 - Specific tooling for backend systems

Example: Customer complained about Portal performance

- Analyze and understand the environment
- Test the environment to evaluate the problem statement
- Analyze the performance data from the OS perspective
 - Found high paging activity
- Analyzed verbose:gc output of the server
 - Confirmed lacking memory (high mark times)
- ➔ requested more memory ➔ immediate improvement
- Done? Nope – did not understand why system was paging
- Further investigation revealed that
 - Customer configured OS for large pages (i.e. Memory was reserved)
 - Customer did NOT configure JVM to use the large pages

Vielen Dank

