Linux on z Systems

IBM Enterprise Content Management for Linux on z Systems Scale-Out Case Study (Part 2): Multiple ECM Nodes with IBM WebSphere Application Server Cluster and Spectrum Scale 4.2

Linux on z Systems

IBM Enterprise Content Management for Linux on z Systems Scale-Out Case Study (Part 2): Multiple ECM Nodes with IBM WebSphere Application Server Cluster and Spectrum Scale 4.2

Before using this information and the product it supports, read the information in “Notices” on page 57.

Edition notices

Contents

Figures v

Tables vii

About this publication ix

Notational conventions. ix

Chapter 1. Introduction 1

Objectives of this white paper 1
IBM Enterprise Content Management 1
  IBM FileNet P8 Content Platform Engine 2
  IBM Content Navigator 2
IBM Spectrum Scale Version 4.2 3
WAS clusters 4

Chapter 2. Summary 7

Chapter 3. Description of the SUT used with Part 2 9

Chapter 4. Hardware and software configuration of the SUT 13

IBM z Systems server 13
IBM storage subsystem setup 13
Networking setup 13
Linux virtual machines 14
  IBM ECM servers 14
  Spectrum Scale NSD servers 14
  IBM HTTP Server 15
  IBM DB2 ECM database server 15
  WAS Deployment Manager 16
  IBM Tivoli Directory Server 16
Workload generator for ICN 17

Chapter 5. Setting up and tuning the SUT 19

Managing the WAS cluster workload 20
  Web server request routing 21
ECM WAS cluster with Spectrum Scale 23
  Important Spectrum Scale parameters 24
  Spectrum Scale node quorum 24
  Spectrum Scale node designations 25
  Spectrum Scale cluster configurations. 25
  Spectrum Scale network block I/O 37
  Results of network block I/O tuning 38

Chapter 6. ECM workload. 43

Chapter 7. Scale-out study results . . . 45

ECM user scaling for Spectrum Scale SD 45
  Transaction throughput and average ICN response time. 46
  CPU loads for ECM nodes 47
  Disk I/O for ECM nodes 48
ECM user scaling for Spectrum Scale NSD 50

Chapter 8. Single ECM node with XFS and Spectrum Scale 4.2 (Part 1 white paper) 53

References 55

Notices 57

Trademarks 59
Terms and conditions 59

Figures

1. IBM Spectrum Scale at a glance 4
2. How the ECM system was configured for Spectrum Scale Shared Disk 9
3. How the SUT was configured for Spectrum Scale Network Shared Disk 11
4. Using an IHS to manage Content Navigator WAS cluster workloads 20
5. Load balancing options for Web server request routing 22
6. Load balancing options for ECM node CPU loads for Web server request routing 23
7. ECM cluster with Spectrum Scale SD cluster configuration 26
8. ECM cluster with Spectrum Scale NSD cluster configuration 29
9. Network and disk I/O for NSD servers (unbalanced vs. balanced server list) 32
10. Scaling pagepool sizes for Spectrum Scale NSD servers 33
11. Scaling pagepool sizes for Spectrum Scale NSD clients 34
12. Average ICN response times for Spectrum Scale cluster configurations 35
13. CPU costs for Spectrum Scale cluster configurations. 36
14. ECM node (NSD client) Linux network packet rates for SD and NSD 37
15. Average ICN response time for different MTU sizes 39
16. CPU cost savings (hypervisor view) for Spectrum Scale NSD when using jumbo frames 40
17. Network jumbo frames – network packet rates and sizes on ECM nodes 41
18. Jumbo frames for virtual networking only – average ICN response time 42
19. Virtual user activity flow and workload levels 43
20. ECM user scaling - ICN page throughput and average response times. 46
21. ECM CPU loads: single ECM node versus ECM cluster 48
22. Spectrum Scale disk I/O - single ECM node vs. ECM cluster 49
23. Spectrum Scale cluster configurations SD and NSD 51

Tables

1. Notation conventions ix
2. ECM cluster-member virtual machine resources 14
3. ECM software. 14
4. Lightweight Spectrum Scale NSD virtual machine resources 15
5. Spectrum Scale NSD server software 15
6. IHS virtual machine resources 15
7. IHS software 15
8. ECM database server virtual machine resources 16
9. ECM database server software 16
10. Dmgr virtual machine resources 16
11. Dmgr software 16
12. ITDS virtual machine 16
13. ITDS software. 17
14. x86_64 hardware 17
15. Software running on x86_64 machine 17
16. Setup tuning used for Part 1 and Part 2 white papers 19
17. Important Spectrum Scale parameters used in the Part 2 case study 24
18. Spectrum Scale node designations: Client roles 25
19. Spectrum Scale node designations: Server roles 25
20. SUT load level definitions. 45

About this publication

This white paper is the second of two closely-related white papers that describe *IBM® Enterprise Content Management* (ECM) scale-out case studies for Linux on z Systems™. This white paper examines the setup, tuning, and performance aspects of an ECM system that is implemented as a virtual environment under z/VM®. The ECM system is based on the *IBM FileNet® P8 5.2.1* and *IBM Content Navigator 2.0.3* applications.

The two related white papers can be summarized as follows:

- *Scale-Out Case Study (Part 1) - Single ECM Node with XFS and Spectrum Scale 4.2* (the first white paper), which focuses on a single ECM node with an XFS file system or IBM Spectrum Scale™ as FileNet File Storage Area. The setup, tuning and performance analysis of the ECM components and IBM Spectrum Scale are described for the "system under test" (SUT).
- *Scale-Out Case Study (Part 2) - Multiple ECM Nodes with IBM WebSphere Application Server Cluster and Spectrum Scale 4.2* (this white paper), which focuses on the scale-out to multiple ECM nodes in a WebSphere® Application Server (WAS) Cluster, and the effectiveness of IBM Spectrum Scale as a distributed parallel filesystem. In addition, the Spectrum Scale cluster configurations *Shared Disk (SD)* and *Network Shared Disk (NSD)* are tested.

Authors

IBM Linux end-to-end Performance team: Thomas Weber, Dr. Juergen Doelle

IBM Spectrum Scale team: Tomer Perry

IBM ECM team: Michael V. Bordash, Nhan Hoang, Dave Royer

Remarks

The web-links referred in this paper are up-to-date as of June, 2016.

Notational conventions

The notational conventions used throughout this white paper are described here.

Table 1. Notation conventions

Symbol	Full name	Derivation
KiB	kibibyte	2 ** 10 byte == 1024 byte
MiB	mebibyte	2 ** 20 byte == 1048576 byte
GiB	gibibyte	2 ** 30 byte == 1073741824 byte
KiB/s	kibibyte per second	2 ** 10 byte / second
MiB/s	mebibyte per second	2 ** 20 byte / second
GiB/s	gibibyte per second	2 ** 30 byte / second

Chapter 1. Introduction

Introductory information is provided in this topic about the setup, tuning, and performance aspects of an ECM system scale-out study for Linux on z Systems. The ECM system is implemented as a virtual environment under z/VM and is based on the *IBM FileNet P8 5.2* (FNP8 5.2) and *Content Navigator 2.0* (ICN 2.0) applications.

Note: This introduction for IBM ECM and Spectrum Scale has the same contents that were provided in the Part 1 white paper. This Part 2 white paper first starts to provide new information with the introduction of the WebSphere Application Server cluster.

Objectives of this white paper

IBM ECM solutions use analytics to provide access to content and documents for the right people at the right time, thereby helping customers to make better decisions. An optimized performance of ECM solutions can better handle the vast amounts of content and documents that are typically involved.

The ECM solution implemented for this case study runs in a virtualized environment on the IBM z Systems™ platform with several layers of performance tuning: starting with the z Systems hardware, z/VM hypervisor, Linux operating system, and ending with ECM applications.

In particular, the multi-node ECM solution with a shared FileNet P8 File Storage Area requires a reliable *parallel filesystem* with good performance, such as *IBM Spectrum Scale*.

One of the major objectives for this case study was to demonstrate the significant advantages of running IBM ECM and Spectrum Scale on Linux on z Systems.

IBM Enterprise Content Management

IBM Enterprise Content Management (ECM) high-value solutions can help companies transform the way they do business by enabling companies to put content in motion by capturing, activating, socializing, analyzing, and governing it throughout the entire lifecycle.

ECM solutions help companies and government agencies reduce costs and make quick, smart, and cost-effective decisions. ECM solutions can be organized into these areas:

Essential ECM
Manages essential content anywhere. Production imaging and capture, enterprise report management, office document management (ODM), and standardization and consolidation are content management foundations for reducing costs and improving efficiencies.

Advanced case management
Takes workflow and business process management to the next level. It brings people, processes, and information together to drive better case outcomes.

Information lifecycle governance
Expands the value of repositories. It can reduce costs and risks while improving compliance posture.

Trusted content analytics
Derives unexpected content insights.

ECM solutions take care of content, security, storage, workflows, decision-making, and productivity. They also support rapid application development, using Web 2.0 technology in user-oriented application environments. That accelerates the creation of solutions and reduces dependence on IT resources.

IBM FileNet P8 Content Platform Engine

IBM FileNet P8 (FNP8) *Content Platform Engine* (CPE) is a component that is designed to handle the heavy demands of a large enterprise.

FNP8 CPE can manage enterprise-wide workflow objects, custom objects, and documents by offering powerful and easy-to-use administration tools. Using the tools, an administrator can create and manage the classes, properties, storage, and metadata that form the foundation of an ECM system.

For more information on the Content Platform Engine, proceed to the FileNet P8 Platform 5.2.1 documentation at the IBM Knowledge Center:

`http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8toc.doc/welcome_p8.htm`

Next, select **System overview** > **FileNet P8 architecture** > **Content Platform Engine**.

IBM Content Navigator

IBM Content Navigator (ICN) is a Web client that provides users with a console for working with content from multiple content servers.

ICN also enables users to create custom views of the content on the Web client by creating teamspaces, which provide a focused view of the relevant documents, folders, and searches that a team needs to complete their tasks.

ICN also includes a powerful API toolkit that you can use to extend the Web client and build custom applications. ICN enables users to search for and work with documents that are stored in content servers that are located around the world from a Web browser.

ICN can be used to access multiple content management repositories. You can also configure the same instance of ICN to access multiple types of content management servers, depending on your business needs. You can use ICN to connect to:

- IBM Content Manager Enterprise Edition repositories
- IBM Content Manager OnDemand repositories
- IBM FileNet P8 repositories
- OASIS Content Management Interoperability Services (CMIS) repositories

For more information on the IBM Content Navigator, proceed to the Content Navigator 2.0.3 documentation at the IBM Knowledge Center:

`http://www.ibm.com/support/knowledgecenter/SSEUEX_2.0.3/contentnavigator_2.0.3.htm`

Then select **IBM Content Navigator overview**.

IBM Spectrum Scale Version 4.2

IBM Spectrum Scale Version 4.2 is a proven, scalable, high-performance data and file management solution that is based upon the *IBM General Parallel File System* (GPFS™).

Spectrum Scale provides comprehensive storage management with a high degree of scalability, flash-accelerated performance, and automatic policy-based storage. Furthermore, it can considerably reduce storage costs while improving security and management efficiency in cloud, big data, and analytics environments.

IBM Spectrum Scale 4.2 includes various integrated *Information Lifecycle Management* (ILM) tools that allow you to automatically move data based on *policies*. This can significantly reduce operational costs, since fewer system administrators can manage larger storage infrastructures.

Note: To see which Spectrum Scale functions are supported for your Spectrum Scale version or operating system, refer to the IBM Spectrum Scale FAQs.

Software Defined Storage lets you build your infrastructure your way:
- Easy to scale with relatively inexpensive commodity hardware while maintaining world class storage management capabilities.
- Supports any combination of flash, spinning disk, and tape.
- Supports various cluster configurations that include storage area networks (SANs), Network Shared disk, and Shared Nothing clusters.
- Supports addition of more storage capacity without affecting the application to greatly simplify administration.
- Spectrum Scale is available for IBM z Systems.

Spectrum Scale 4.2:
- Provides global data access across geographic distances and unreliable WAN connections.
- Provides proven reliability with use in the most demanding commercial applications.
- Protects data from most security breaches, unauthorized access, or being lost, stolen, or improperly discarded with native file encryption for data at rest and secure erase.
- Provides multi-site support, connecting local Spectrum Scale cluster to remote clusters to provide Disaster Recovery configurations.

IBM Spectrum Scale provides additional protocol access methods in the Standard and Advanced editions of the product. Providing these additional file and object access methods and integrating them with Spectrum Scale offers several benefits:
- It enables users to consolidate various sources of data efficiently in one global name space.
- It provides a unified data management solution and enables not just efficient space utilization but also avoids having to make unnecessary data moves just because access methods may be different.

The additional protocol access methods integrated with Spectrum Scale are file access using Network File System (NFS) and Server Message Block (SMB) and object access using OpenStack Swift. While each of these server functions (NFS, SMB, and Object) uses open source technologies, this integration adds value by providing the ability to scale and by providing high availability using the clustering technology in Spectrum Scale.

- IBM Spectrum Scale provides policy-driven compression to reduce the size of data at rest.
- IBM Spectrum Scale provides block storage for OpenStack deployments via its cinder driver support.
- IBM Spectrum Scale is a POSIX-compliant file system that offers an enterprise-class alternative to HDFS and is a preferred platform for data analytics where it's Hadoop compatibility extensions help replace HDFS in a Hadoop ecosystem, with no changes required to Hadoop applications.

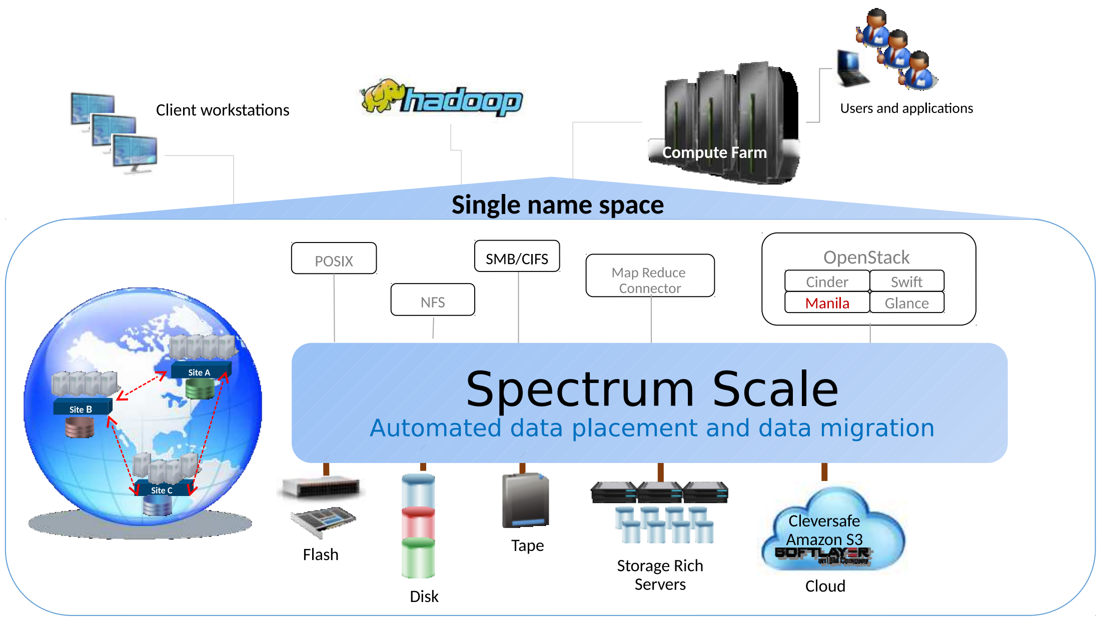


Figure 1. IBM Spectrum Scale at a glance

Spectrum Scale powers many of the world's largest scientific supercomputers and commercial applications that require high-speed access to large volumes of data such as digital media, engineering design, business intelligence, financial analytics, seismic data processing, geographic information systems, and scalable file serving.

WAS clusters

IBM WAS clusters are grouped application servers that are managed together and participate in workload management.

A cluster can contain:

- A single WAS node (for example, a single ECM node with WAS clusters as is described in the White Paper Part 1).
- Multiple nodes.
- Individual application servers.

One or more application servers can run on a single WAS node.

WAS clusters are responsible for workload balancing among application servers. Furthermore, application servers that are a part of a cluster are called *cluster members*.

- Applications installed on cluster level are automatically installed on each cluster member.
- Because each cluster member contains the same applications, weights can be assigned according to each application server's capacity.

Each WAS cluster also provides the *failover* capability:

- If one application server is not available to perform a task, the task can be reassigned to another application server in the cluster.
- Reassigning a task has obvious advantages over running a standalone application server that can become overloaded if too many requests are received.

For more information on IBM WAS clusters, proceed to the WAS documentation at the IBM Knowledge Center:

`http://www.ibm.com/support/knowledgecenter/SSAW57/mapfiles/product_welcome_wasnd.html`

Then, search for the topic **Introduction: Clusters**.

Chapter 2. Summary

This Part 2 white paper shows the advantages of an ECM cluster in a WebSphere Application Server environment for Linux on z Systems. Spectrum Scale was used as the underlying parallel filesystem to enable cluster-wide access to the shared FileNet File Storage Area that was populated with documents. The basic ECM application, WAS, and Spectrum Scale related tuning for a single node ECM cluster is described in the *Scale-Out Case Study (Part 1) - Single ECM Node with XFS and Spectrum Scale 4.2*.

This white paper focuses on the performance of a four-node ECM cluster.

- At the start of the case study, the first question was the load balancing algorithm. For this study, the Web server IBM HTTP Server was used to route requests in “Round Robin” mode to the four ECM WAS cluster members. This resulted in a slight improvement in load balancing among the WAS cluster members, compared to the other load balancing option “Random”.
- Another important point to consider was the Spectrum Scale cluster configuration mode. We compared the two basic Spectrum Scale cluster configurations:
  - *Shared Disk (SD)*: The filesystem disks were directly attached to each of the four ECM nodes. As a result, any application data disk I/O was done directly over the SAN.
  - *Network Shared Disk (NSD)*: The filesystem disks were directly attached to dedicated NSD servers that propagate the filesystem via TCP/IP to the four ECM nodes. This Spectrum Scale cluster configuration theoretically allows a very high number of cluster members independently from the scope of the SAN network.

  The outcome for the above question was that a tuned Spectrum Scale NSD setup performed as well as a SD setup for our workload used. Only the initial out-of-the-box NSD configuration shows a performance degradation.
- *Spectrum Scale Network Shared Disk (NSD) tuning*
  - To reach a balanced load for the available NSD servers (where there is more than one NSD server), it is important to vary the order of servers in the list of the NSD servers for the NSD disk definitions. When all NSD disks have the same order of NSD servers, the result is that only the first NSD server in the list is used and receives all NSD disk I/O requests. Therefore, it can be fully loaded while the others are waiting in failover mode.
  - The page pool plays a minor role for the NSD servers and can be even relatively small.
  - The page pools on the NSD clients are more important. This is where the ECM application data processing occurs and cache usage is effective for the NSD cluster configuration.
- *Linux Network MTU sizes*
  - An important NSD tuning parameter is the MTU size.
  - When using the default MTU size of 1492 for the entire SUT, the NSD setup showed a significant degradation compared to the SD setup.
  - When using jumbo frames (MTU 8192) for the entire SUT, the response times for the NSD setup became only slightly worse than the SD setup with jumbo frames.

  - Because it is often difficult to run in an enterprise LAN environment exclusively with jumbo frames, a mixed MTU setup was also tested. In this setup, the remote machines in the LAN used a default MTU of 1492 to access the ECM environment but the virtual network inside the z Systems server used jumbo frames. This is relatively straightforward to implement and provided nearly the same performance compared to an environment that ran exclusively with jumbo frames.
- *Scale-out study with Spectrum Scale SD cluster configuration.* The most relevant performance metrics for this performance test were the response time and the z/VM CPU load (hypervisor load), which reflected the total load caused by all virtual machines for the SUT. To analyze the scaling behavior, the workload was scaled from a lower load level (around 5 IFL processors used) to a high load level (more than 20 IFL processors used).
  - The single-node SD setup with Spectrum Scale was mostly better than XFS, and only at the highest workload level was it worse than XFS, *but still good!* Spectrum Scale for Linux on z Systems demonstrated *excellent performance* for an ECM-like type of workload, even in a non-clustered node configuration.
  - When scaling out to a four-node ECM WAS cluster with Spectrum Scale as underlying parallel filesystem, the response times were *significantly better* across all load levels than the single ECM WAS node performance. The higher the load, the bigger were the differences between response times. Finally at the highest load level, the four-node ECM WAS cluster with Spectrum Scale was *40% faster* than the single ECM WAS node with XFS.
  - The CPU load for the four-node ECM WAS cluster was at the lower load levels higher than for the single ECM WAS node. However, the four-node ECM WAS cluster generally provided *better response times*. The higher the load level, the less was the difference between both setups in terms of CPU load. At the highest load level, the four-node ECM WAS cluster provides *40% better response times* compared to the single ECM WAS node at the same CPU load!

Overall, the four-node ECM WAS cluster together with a disk-I/O-intensive workload and Spectrum Scale as parallel filesystem for the FileNet File Storage Area was a *very successful combination* in regard to overall performance. The CPU load overhead for the clustered setup compared to the single-node environment reduced when the load level increased.

For ECM systems with high workload levels, it is *highly recommended* to consider the use of a clustered ECM setup when transaction response times are important. In addition to the performance wins, a clustered ECM setup provides in any case the high availability feature.

Chapter 3. Description of the SUT used with Part 2

For Part 2, the SUT was based upon *multiple ECM nodes* (whereas for Part 1 it was based upon a single ECM node). The Part 2 SUT consisted of four ECM nodes that ran the ECM application pair IBM FNP8 CPE and ICN for Linux on z Systems on each node. Both applications ran in a clustered WAS environment. More precisely, each application ran in its own WAS cluster.

Furthermore, the SUT included front-end and back-end server components such as:

- IBM IHS as Web server
- IBM ITDS as Directory Service Provider
- IBM DB2® database server

The SUT followed a 3-tier architecture model, enabling client components to interact with data resources and legacy applications, with logical tiers distributed across independent systems.

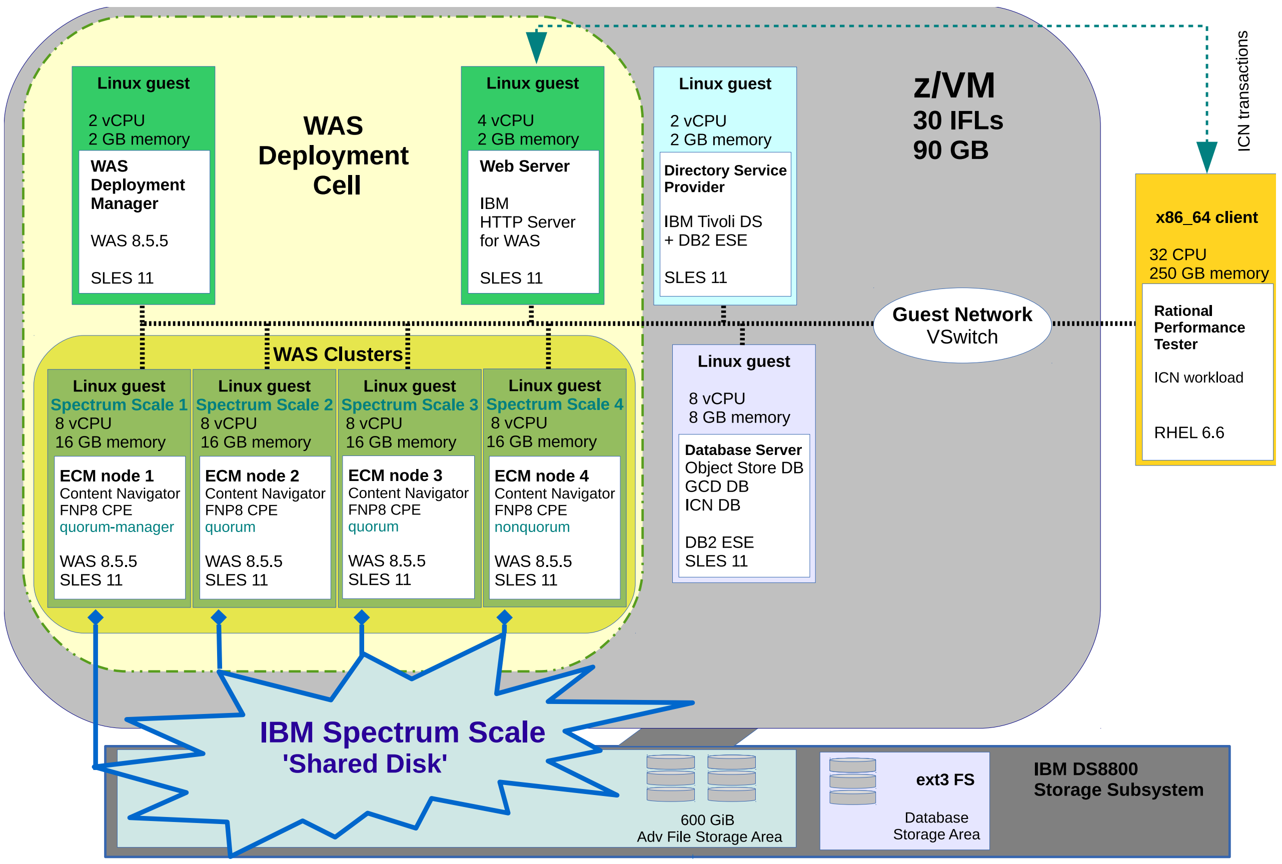


Figure 2. How the ECM system was configured for Spectrum Scale Shared Disk

The SUT was implemented in a single IBM z Systems server Logical Partition (LPAR) that runs a z/VM hypervisor for a virtualized environment.

- ICN and FNP8 CPE were each deployed in their own multiple-node WAS clusters.

- The FileNet Advanced File Storage Area was 600 GiB large and resided on Spectrum Scale as parallel filesystem.
- In the course of the study two Spectrum Scale cluster configurations were considered:
  - Shared Disk
  - Network Shared Disk

The Spectrum Scale cluster configuration SD specifies that SAN filesystem disks were directly attached to all ECM nodes that mount the filesystem. In respect of the SUT, the four ECM nodes were directly attached to the FileNet Advanced Storage Area disks.

Components of the ICN application were mapped to the Web server IHS. IHS was used to manage the HTTP traffic for ICN and did the request balancing for the multiple ECM WAS node setup.

The WAS clusters, the WebSphere Deployment Manager (Dmgr), and IHS together formed a *WebSphere Deployment Cell*.

The DB2 ECM database server maintained three different databases:
- An FNP8 CPE Object Store database.
- An FNP8 Global Configuration Data (GCD) database.
- An ICN database.

The ITDS Directory Service Provider provided authentication methods for usernames and passwords for the ECM system.

For guest-to-guest communication a z/VM Virtual Switch (VSWITCH) was used that also provided LAN access via a 10GbE OSA-Express® network card.

The load generator for ICN was a x86_64 machine running the IBM Rational® Performance Tester (RPT) application with a custom workload. The x86_64 machine was connected over a 10GbE network to the IBM z Systems server.

Figure 3 on page 11 shows a similar SUT layout to that of Figure 2 on page 9. The only difference is the Spectrum Scale cluster configuration. Now the cluster configuration NSD was used, which means that the filesystem disks for the FileNet File Storage Area were connected over the network to the ECM nodes. Three additional lightweight Spectrum Scale nodes were newly introduced which had the Spectrum Scale filesystem disks for the FileNet File Storage Area directly attached. These additional nodes were so called dedicated Spectrum Scale NSD servers that did the disk I/O for the ECM application nodes. The four ECM nodes were considered as NSD clients in the NSD cluster configuration and performed the network block I/O to the NSD servers.

The NSD cluster configuration is also important for widely dispersed Spectrum Scale nodes, which have no direct SAN access. This important Spectrum Scale cluster configuration was also considered in this case study.

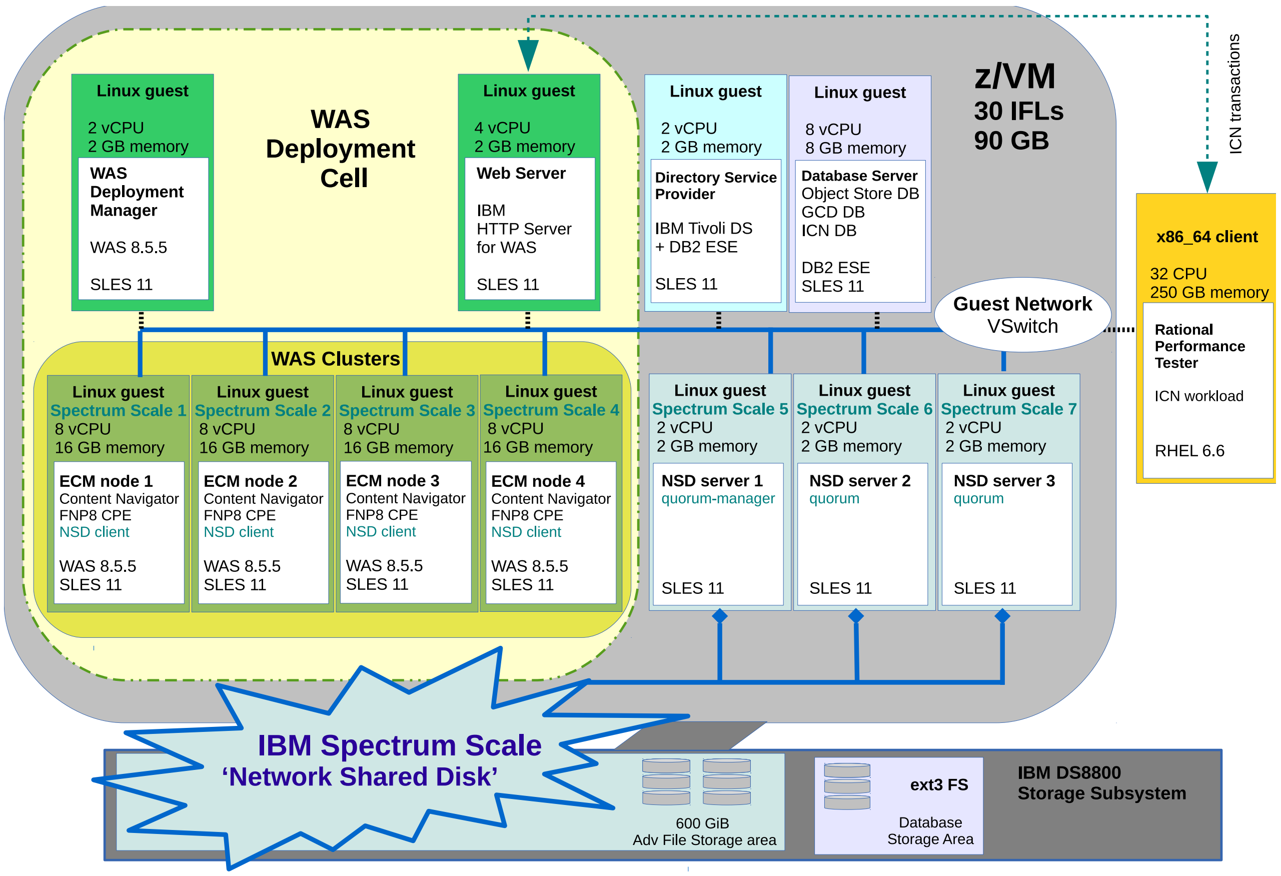


Figure 3. How the SUT was configured for Spectrum Scale Network Shared Disk

Chapter 4. Hardware and software configuration of the SUT

This topic describes the software and hardware that was used for the case study. The SUT formed a basic customer-like ECM system with FNP8 CPE and ICN to integrate the FileNet repositories.

The ECM system was implemented in a z Systems server LPAR running a z/VM 6.3 hypervisor. The storage subsystem was an IBM DS8870 connected via FICON® Express.

IBM z Systems server

The case study involved the configuration of an LPAR in a zEnterprise® EC12 (zEC12) Model 2827.

The LPAR setup consisted of:

- 30 Integrated Facility for Linux (IFL) processors.
- 90 GiB Central Storage.
- 1 10GbE OSA-Express4S card.
- FICON Express8S LX (short wave) cards supporting ECKD™ for DASD device access or FCP devices.

IBM storage subsystem setup

The storage subsystem used in this case study was an IBM System Storage® DS8870 Model 2422.

These are the DASD disk devices that were used as z/VM Linux guest operating systems disks:

- different DASD models (model-27, model-54 and model-128) directly attached to the z/VM guests.
- DASDs selected from 2 ECKD storage pools (storage pool striping enabled) and picked from both internal storage subsystem servers.
- HyperPAV alias devices used for DASDs.

These are the FCP attached SCSI devices that were used as ECM data disks (FileNet File Storage Area and ECM database):

- 100 GiB and 250 GiB SAN-attached SCSI devices.
- SCSI devices are picked from both internal storage subsystem servers.

Networking setup

The network used in this white paper is described in this topic.

- z/VM Virtual Switch (VSWITCH) for all virtual machines (guest-to-guest communication).
- 10 GbE OSA-Express4S card attached to the VSWITCH for external network connectivity.
- no network encryption.

Linux virtual machines

This topic describes the Linux virtual machines that were configured for the SUT. The SUT was implemented as a distributed setup with software components installed in different z/VM Linux virtual machines. These Linux virtual machines were grouped by their application tasks.

IBM ECM servers

The FNP8 CPE and ICN applications represented the ECM system used in the case study.

FNP8 CPE addressed the enterprise content and process management requirements, while ICN is a Web client that provided a console for working with the contents of FileNet repositories.

Both applications were installed in independent WAS clusters with four uniform cluster members.

The FileNet Advanced File Storage Area was shared among all four ECM WAS cluster members either with:

- Directly-attached (Spectrum Scale SD) SAN filesystem disks.
- Network-attached (Spectrum Scale NSD) SAN filesystem disks.

Table 2. ECM cluster-member virtual machine resources

ECM WAS cluster member (maximum 4 x cluster members)
8 virtual CPUs
16 GiB memory
10 GbE LAN via VSWITCH
1 x DASD Model 27 for Linux - 4x HyperPAV aliases directly attached
6 x 100 GiB FCP shared FCP-attached SCSI disks for Advanced File Storage Area
2 x 256 GiB FCP-attached SCSI disks for backup purposes

Table 3. ECM software

Software	**Service level**
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM WebSphere Application Server Network Deployment (64-bit)	v8.5.5 Fix Pack 3 (8.5.5.3)
IBM WebSphere SDK for Java™ Technology Edition 7	v7.0.7.0
IBM FileNet CP Engine	v5.2.1
IBM Content Navigator	v2.0.3
IBM Spectrum Scale	v4.2.0.0

Spectrum Scale NSD servers

This description applies to the NSD configuration only. Three lightweight additional Spectrum Scale NSD servers were added to the SUT for Spectrum Scale NSD.

The SAN disks for the FileNet File Storage Area were directly attached to these three servers only. The NSD clients (ECM nodes) accessed the File Storage Area disks over the network and had no direct SAN access.

Table 4. Lightweight Spectrum Scale NSD virtual machine resources

Spectrum Scale NSD server (maximum 3 x NSD servers)
2 virtual CPUs
2 GiB memory
10 GbE LAN via VSWITCH
1 x DASD Model 27 for Linux - 4 x HyperPAV aliases directly attached
100 GiB shared directly-attached FCP-attached SCSI disks for FileNet File Storage Area

Table 5. Spectrum Scale NSD server software

Software	**Service level**
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM Spectrum Scale	v4.2.0.0

IBM HTTP Server

The Web communication for the ECM system was managed via an IBM HTTP Server (IHS).

The IHS received client requests and forwarded them to the ICN application. Components of ICN were mapped to the IHS to manage the HTTP traffic.

The IHS also act as a load balancer, if there are more than one WAS cluster members available.

Table 6. IHS virtual machine resources

IHS virtual machine
4 virtual CPUs
2 GiB memory
10 GbE LAN via VSWITCH
1 x DASD Model 27 for Linux - 4 x HyperPAV aliases directly attached

Table 7. IHS software

Software	**Service level**
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM HTTP Server	v8.5.5 Fix Pack 3 (8.5.5.3)

IBM DB2 ECM database server

The database server hosted three databases for the ECM system.

- The *FNP8 CPE Object Store* database is a relational database that contains documents, workflows and other objects.
- The *FNP8 Global Configuration Data* (GCD) database stores mostly FileNet configuration metadata.
- Finally, ICN stores configuration data and preferences in its *own* database.

Table 8. ECM database server virtual machine resources

ECM database server virtual machine
8 virtual CPUs
8 GiB memory
10 GbE LAN via VSWITCH
1 x DASD Model 27 for Linux – 4 x HyperPAV aliases directly attached
6 x DASD Model 54 for ICN and GCD database and database log files 24 x HyperPAV aliases directly attached
4 x 256 GiB FCP-attached SCSI disks for FNP8 Object Store and database backups

Table 9. ECM database server software

Software	**Service level**
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM DB2 Enterprise Server Edition	v10.5 Fix pack 4 (10.5.0.4)

WAS Deployment Manager

The WAS Deployment Manager (Dmgr) is the main administration process that manages all other application servers in a WAS deployment cell, including node agents and application server processes.

Dmgr is required only for WAS configuration tasks. Therefore, this virtual machine typically had almost no CPU load when the ECM workload was running.

Table 10. Dmgr virtual machine resources

Dmgr virtual machine
2 virtual CPUs
2 GiB memory
10 GbE LAN via VSWITCH
1 x DASD Model 27 for Linux – 4 x HyperPAV aliases directly attached

Table 11. Dmgr software

Software	**Service level**
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM WebSphere Application Server Network Deployment (64-bit)	v8.5.5 Fix Pack 3 (8.5.5.3)

IBM Tivoli Directory Server

The IBM Tivoli Directory Server (ITDS) provided authentication methods for user names and passwords to the ECM system.

Table 12. ITDS virtual machine

ITDS virtual machine
2 virtual CPUs
2 GiB memory
10 GbE LAN via VSWITCH

Table 12. ITDS virtual machine (continued)

ITDS virtual machine
1 x DASD Model 27 for Linux – 4 x HyperPAV aliases directly attached

Table 13. ITDS software

Software	Service level
SLES11 for z Systems (64-bit)	Service Pack 3 (SP3)
IBM Tivoli Directory Server	v6.3

Workload generator for ICN

The workload generator for ICN was an x86_64 machine that ran the IBM Rational Performance Tester (RPT) with a custom workload.

The workload mix was a typical ECM load with various calls to functions of the ICN user interface with a focus on tasks that access content in the FileNet File Storage Area.

Table 14. x86_64 hardware

Client hardware	Setup
IBM Flex System x86_64	• 16 CPUs (2.9 GHz) • 250 GiB memory • 10 GbE network card

Table 15. Software running on x86_64 machine

Software	Service level
Red Hat Enterprise Linux Server Release	Release 6.6
IBM Rational Performance Tester	v8.5.1 Fix Pack 3

Chapter 5. Setting up and tuning the SUT

This topic describes the *multiple* ECM node clustered setup of the SUT. The Part 1 white paper described a *single* ECM node setup. This Part 2 white paper of the case study describes a clustered setup with WAS and Spectrum Scale.

The setup and tuning applied to the SUT was described in detail in the Part 1 white paper. Since the Part 2 white paper uses the same basic tuning that was used for Part 1, you should refer to the Part 1 white paper *IBM Enterprise Content Management for Linux on z Systems, Scale-Out Case Study (Part 1): Single ECM Node with XFS and IBM Spectrum Scale 4.2* for details.

The setup tuning covered these main programs:
- z/VM
- Linux for z Systems
- WebSphere Application Server
- ECM software components
- DB2
- Spectrum Scale

Table 16 lists the components that were tuned.

Table 16. Setup tuning used for Part 1 and Part 2 white papers

Program	Components that were tuned
z/VM	• Virtual Networking • HyperPAV
Linux on z Systems	• Networking • HyperPAV • FCP Hardware Data Router Support • Device Mapper-Multipathing • Linux kernel parameter
ECM software tuning	• IHS gateway tuning • common WAS tuning • WAS JVM tuning • ECM database server tuning
ECM database server	• workload type • database configuration • database indexes
Spectrum Scale	• `pagepool` size • `maxFilesToCache` • `maxStatCache` • `blocksize` • Inodes

Managing the WAS cluster workload

When clustering WebSphere Application Servers that host Web containers, a plug-in workload management is *automatically* enabled among the cluster members.

Using the HTTP transport protocol, the routing of servlet requests occurs between:

- The Web server plug-in (IHS plug-in for the SUT).
- WAS cluster members.

Furthermore, there is the *failover* aspect of a WAS cluster environment:

- If one server is unavailable or cannot perform the task, tasks can be reassigned to another cluster member.
- Reassigning a task has the obvious advantage over running a standalone application server in that a standalone application server can become overloaded if too many requests are made.

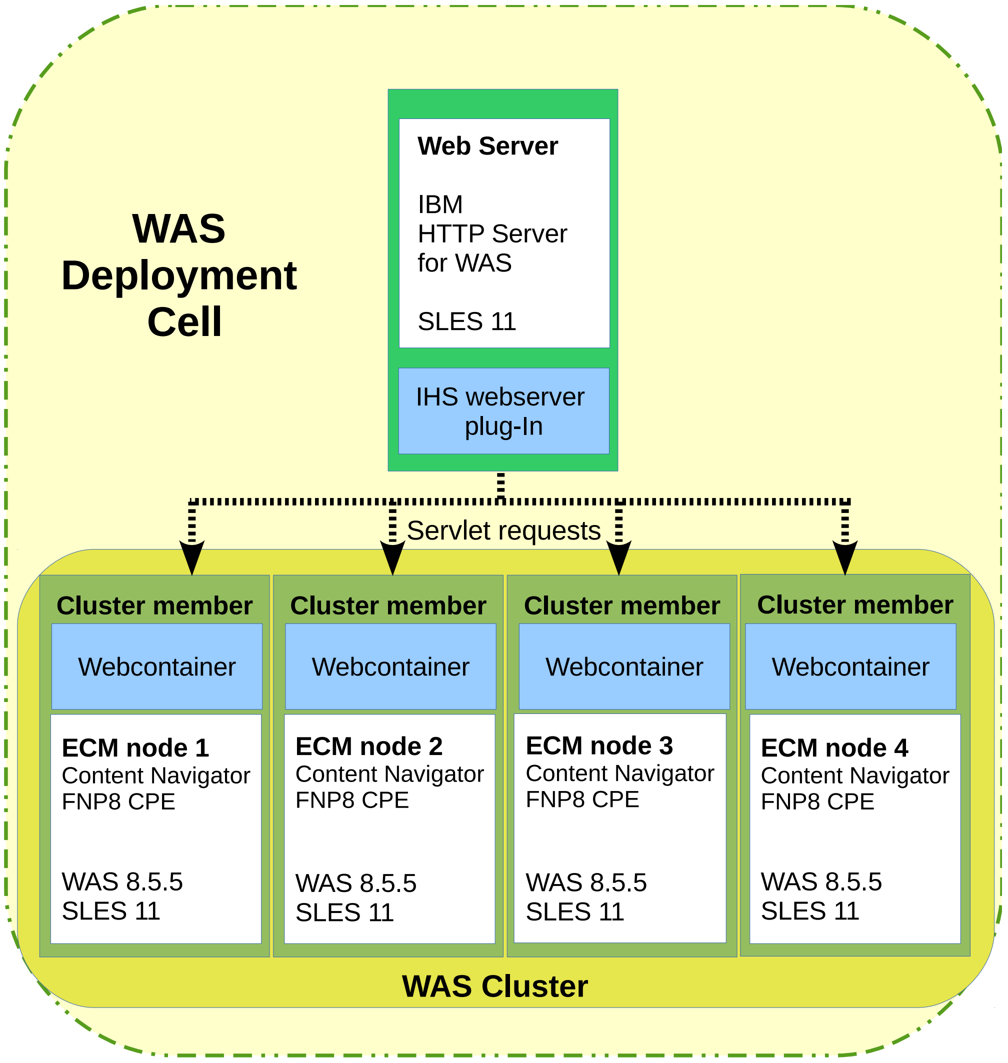


Figure 4. Using an IHS to manage Content Navigator WAS cluster workloads

In a WAS cluster, each member contains the same applications. Therefore, client requests can be distributed across cluster members according to the capacities of the different machines. For that purpose, *weights* can be assigned to each cluster member.

For the SUT, FNP8 CPE and ICN formed an application pair on an ECM node.

- Each of the clustered applications ran in their own WAS cluster but on the *same ECM node*.
- Workload management was only done only at *ICN cluster level* because ICN received client requests and was mapped at cluster level to the IHS Web server.
- Due to the one-to-one relationship between the ICN and FNP8 CPE application on each ECM node, there was also an *inherited* workload management for FNP8 CPE.

The WAS cluster workload management is done by routing servlet requests to all available cluster members according to their weights.

For the SUT: The virtual machine capacities for the ECM nodes were equal. Therefore, each ECM node had the same cluster member weight assigned (the aim was to have equal workload distributions on all four ECM nodes).

Web server request routing

The load balancing option for an Web server plug-in can be configured to specify how the plug-in will route requests to the various application servers associated with that Web server.

The IHS Web server plug-in works with two different load balancing options.

Round robin option (the default)

The “round robin” option uses a random starting point: the first application server is randomly selected. From that point onwards, “round robin” is then used to choose application servers.

The “round robin” option ensures that for multiple-thread based Web servers, all of the threads do not start up by sending the first request to the same application server.

Random option

Using the “random” option, application servers are picked randomly.

If the Web server is using multiple process threads to send requests to the application server and depending on the workload, “random” selection can sometimes yield a more even distribution of work across the cluster.

To set request routing, follow this WAS Administration console path: **Web servers** > ***web_server_name*** > **Plug-in properties** > **Request routing**.

The load balancing options had been tested with the ECM workload used for this study at a high load level.

Figure 5 on page 22 shows the average ICN transaction response times for the two possible load balancing options.

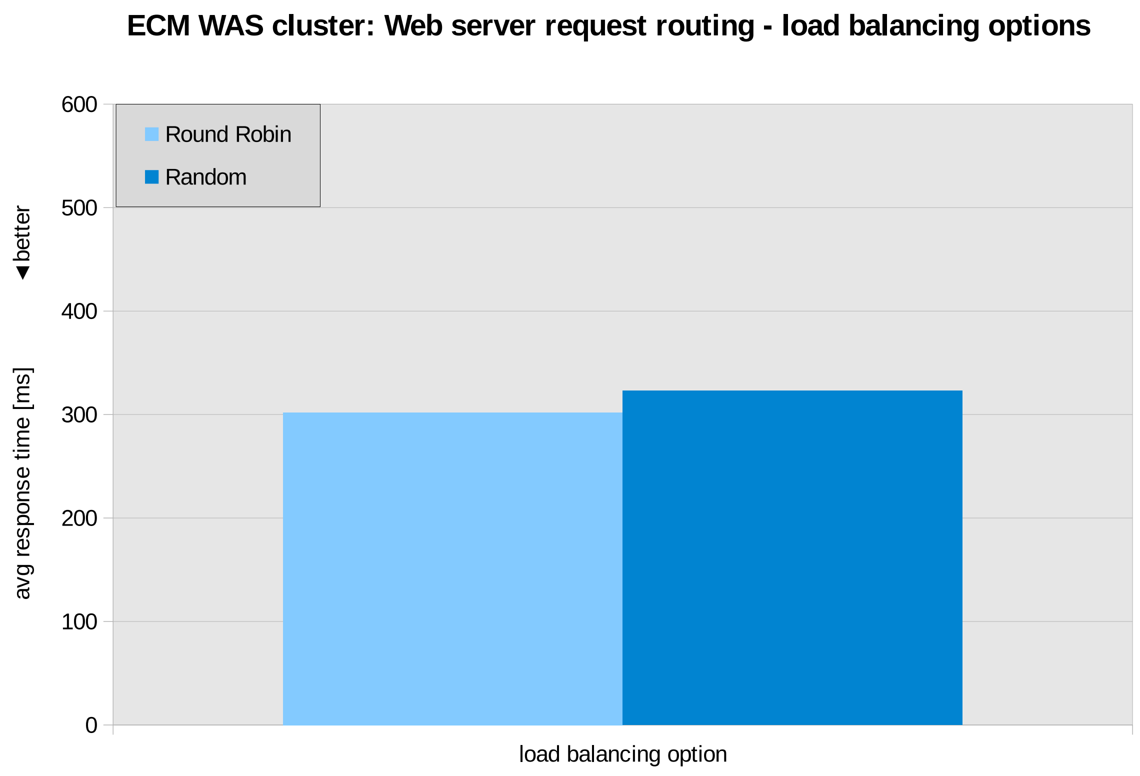


Figure 5. Load balancing options for Web server request routing

For both load balancing options the average ICN transaction response time was similar, however the “round robin” option showed a slightly better response time than the “random” option.

When looking at the CPU loads as an indicator for the load distribution of the ECM nodes, it appeared that the loads were slightly more unequally distributed than was the case using the “random” option. Every ECM node used around 5 IFL processors at the highest ECM load level.

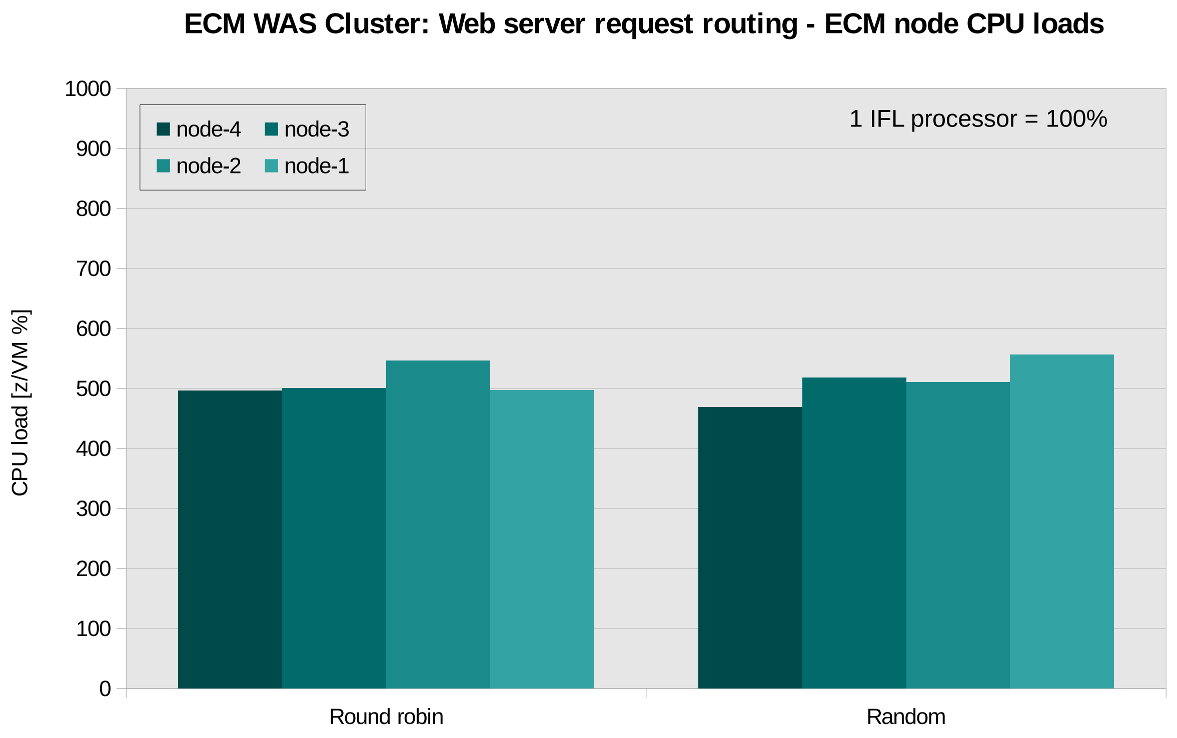


Figure 6. Load balancing options for ECM node CPU loads for Web server request routing

Because the “round robin” option produced a slightly better equal load distribution, it was considered to be more suitable than the “random” option for our type of workload.

For the SUT: The default load balancing option “round robin” was used for the case study measurement series.

For information about IBM WAS cluster workload management, proceed to the WebSphere Application Server documentation at the IBM Knowledge Center:
`http://www.ibm.com/support/knowledgecenter/SSAW57/mapfiles/product_welcome_wasnd.html`

Then search for the topic: **Balancing workloads**.

ECM WAS cluster with Spectrum Scale

IBM Spectrum Scale is a scalable, high-performance data and file management solution (based upon IBM General Parallel File System or GPFS).

The Part 1 white paper (the single ECM node study) proved that Spectrum Scale can compete with a high performance filesystem such as XFS.

When shared amongst all nodes in the ECM WAS cluster, the FileNet Advanced File Storage Area had to reside on a parallel filesystem. For this reason, the File Storage Area resided on the Spectrum Scale filesystem configured as either SD or NSD.

This topic discusses the Spectrum Scale setup in more detail in relation to the ECM WAS cluster.

This example shows the mounted FileNet File Storage Area on a Spectrum Scale filesystem (type `gpfs`). The File Storage Area was 600 GiB large and was already populated with some ECM content data.

```
# df -hT
Filesystem                        Type   Size  Used Avail Use% Mounted on
...
/dev/FSData                       gpfs   600G  337G  264G  57% /data/FSData
```

Important Spectrum Scale parameters

The Spectrum Scale parameters commonly considered for performance tuning related to ECM workloads are described in detail in the Part 1 white paper that preceded this white paper. Most of the important parameters are related to Spectrum Scale *cache usage*. The table provided here lists the parameters used in the Part 2 case study, together with their default and tuned values .

Table 17. Important Spectrum Scale parameters used in the Part 2 case study

Parameter	Description	Value used in Part 2 case study
Pagepool size	Used for caching user data and file system metadata (default 1GiB).	• **SD:** set to 2 GiB on ECM nodes • **NSD:** 2 GiB on ECM nodes (NSD clients) and 1 GiB for NSD servers
`maxFilesToCache`	Total number of files that can be cached at one time in the pagepool (default 4,000).	• Set to `20,000` (20k). This value proved as good for this particular ECM workload
`maxStatCache`	Not effective on the Linux platform (default 1,000).	• Set to a smaller value (`512`).
`blocksize`	The blocksize of the Spectrum Scale filesystem.	• Default (`256 KiB`) was used.
Inode limit	The default Inode limit value can become a threshold for the file systems.	• Was increased to `15M` inodes.

For other details, refer to the chapter "IBM Spectrum Scale tuning parameters" in the Part 1 white paper.

Spectrum Scale node quorum

Spectrum Scale uses a cluster mechanism called *quorum* to maintain data consistency in the event of a node failure.

Quorum operates on the principle of majority rule. This means that a majority of the nodes in the cluster must be successfully communicating before any node can mount and access a filesystem. This keeps any nodes that are cut off from the cluster (for example due to a network failure) from writing data to the filesystem. Maintaining quorum in a Spectrum Scale cluster means that a majority of the nodes designated as quorum nodes are able to successfully communicate.

Node quorum is the default quorum algorithm for Spectrum Scale. Node quorum is defined as one plus half of the explicitly defined quorum nodes in a cluster. This is considered as the node pool from which node quorum is derived.

There are no default quorum nodes. Nodes with a quorum role must be *explicitly* specified.

For the SUT: The SUT had four ECM nodes in total. A reasonable number of quorum nodes would be three. The general recommendation is to choose an odd number of quorum nodes. In a three node quorum configuration two nodes have to be communicating for cluster operations to continue.

Note: Another possibility would be a node quorum with tiebreaker disks. The cluster can remain online with only one surviving node. Typically tiebreaker disks are only used for two node clusters.

For more information on Spectrum Scale Quorum, proceed to the IBM Spectrum Scale documentation at the IBM Knowledge Center:

```
http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/ibmspectrumscale42_welcome.html
```

Then, search for the topic **Node Failure** and select **Quorum**.

Spectrum Scale node designations

Spectrum Scale nodes can have certain roles. There must be at least one node with a manager role, and a reasonable number of nodes with quorum roles.

For details of quorum roles, see "Spectrum Scale node quorum" on page 24. The node designations are divided into *client roles* and *server roles* which require appropriate Spectrum Scale licenses.

Table 18. Spectrum Scale node designations: Client roles

Node designation	Description
client	(default) not in the pool of manager nodes
nonquorum	(default) not in the pool of quorum nodes

Table 19. Spectrum Scale node designations: Server roles

Node designation	Description
manager	part of the pool of filesystem and token manager nodes
quorum	part of the node pool from which quorum is derived

Note: A Spectrum Scale node can have more than one role assigned, but only client or server roles can be combined (for example, quorum-manager).

Spectrum Scale cluster configurations

IBM Spectrum Scale supports various cluster configurations that include SAN Shared Disk (SD), Network Shared Disk (NSD), and Shared Nothing (SN) clusters.

The cluster configurations *SD* and *NSD* were considered during this case study with the ECM WAS cluster.

While SD was the preferred cluster configuration for a system like the SUT, NSD was included into this case study too.

Note: The SD cluster configuration is the preferred model for smaller clusters (for example, less than 50 servers) and provides the highest performance for cluster wide data access. The NSD configuration provides access to a Spectrum File filesystem via a network connection by doing network block I/O. Therefore, NSD clients do not need to be connected to the SAN. This enables distant remote systems to mount shared filesystems including the high availability features provided by Spectrum Scale.

Shared Disk (SD)

For the SD cluster configuration, the filesystem disks are directly attached to the cluster nodes. As a result, any user data flows over the SAN, and Spectrum Scale control information over a TCP/IP network (LAN).

With regard to the SUT, this means that the FileNet File Storage Area was shared among the ECM nodes and the SAN disks for the File Storage Area were directly attached to each of the four ECM nodes. Every ECM node triggered its own disk I/O directly to the SAN and caused at the same time a control information flow over the LAN to maintain data consistency.

Figure 7 shows the four ECM cluster nodes sharing the FileNet Advanced File Storage Area using the Spectrum Scale cluster configuration SD.

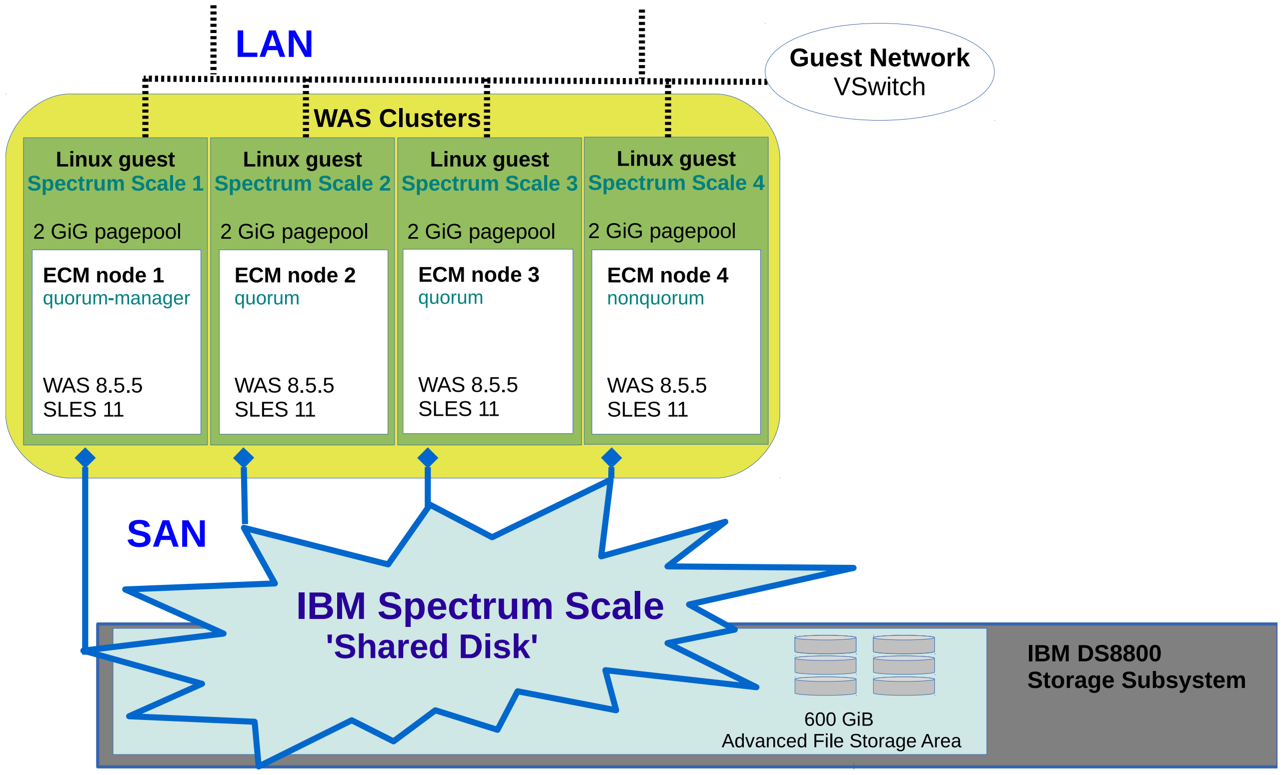


Figure 7. ECM cluster with Spectrum Scale SD cluster configuration

The following example lists the Spectrum Scale cluster configuration SD for the SUT with its node designations for the ECM nodes (cluster members).

```
# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         ECM_4_node
  GPFS cluster id:           714383681xxxxxxxxx
  GPFS UID domain:           ECM_4_node
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
---------------------------------------------------------------------
   1   ECMnod1           10.xxx.xx.xxx  ECMnod1          quorum-manager
   2   ECMnod2           10.xxx.xx.xxx  ECMnod2          quorum
   3   ECMnod3           10.xxx.xx.xxx  ECMnod3          quorum
   4   ECMnod4           10.xxx.xx.xxx  ECMnod4
```

The Spectrum Scale cluster had four member nodes (`ECMnod[1-4]`). `ECMnod1` had the node designation quorum-manager, which means it acted as the filesystem manager for the cluster and was also in the node pool from which quorum was derived.

- `ECMnod2` and `ECMnod3` complemented the node quorum and were both in the pool of quorum nodes.
- `ECMnod4` had the node designation *nonquorum-client*, which was not explicitly listed in the designation column for the mmlscluster command.

Therefore, `ECMnod[1-3]` had server roles and `ECMnod4` a client role.

The **`mmlsnsd`** command displays network shared disk information for a Spectrum Scale cluster. In the case of SD, “directly attached” is usually reported in the NSD servers column.

Here is an example of SAN disks for the FileNet File Storage Area that are *directly* attached to a node:

```
# mmlsnsd

File system     Disk name      NSD servers
---------------------------------------------------------------------------
FSData          nsd14cb        (directly attached)
FSData          nsd15cb        (directly attached)
FSData          nsd14cc        (directly attached)
FSData          nsd15cc        (directly attached)
FSData          nsd14cd        (directly attached)
FSData          nsd15cd        (directly attached)
```

In the above example, the SAN disks for the filesystem `FSData` (which was used as FileNet File Storage Area) were directly attached to each ECM node. The **`mmlsnsd`** command would give the same output on all four ECM nodes.

The next example shows the filesystem `FSdata` mounted on all four ECM nodes.

```
# mmlsmount FSData -L
File system FSData is mounted on 4 nodes:
  10.xxx.xx.xxx   ECMnod1
  10.xxx.xx.xxx   ECMnod2
  10.xxx.xx.xxx   ECMnod3
  10.xxx.xx.xxx   ECMnod4
```

In the above example, Spectrum Scale allows the filesystem FSDdata to be mounted on all four ECM nodes so that FileNet File Storage Area can be shared.

In the following example, the Spectrum Scale pagepool sizes for ECM nodes are listed:

```
# mmlsconfig pagepool
pagepool 2G
```

Every node in a Spectrum Scale cluster has its own pagepool. The size of the pagepool can vary according to the intended role or task of the node. The default pagepool size for Spectrum Scale version 4.2 is 1 GiB.

For the SUT: For the SD cluster configuration, the pagepool size was set to 2 GiB on all four ECM nodes. This pagepool size was chosen by taking into account:
- The available memory on the ECM nodes.
- The used ECM workload.

Network Shared Disk (NSD)

If all servers in a cluster cannot be attached directly to the SAN, Spectrum Scale provides a protocol that implements a block-level interface over the network, called an NSD.

For larger Spectrum Scale clusters, the complexity of SAN configuration can become increasingly difficult. In addition, the costs and management of the SANs increase as the cluster size increases.

Such setups can benefit from *NSD*, with its network block I/O feature. All cluster members must be able to communicate via a network to each other. This also enables the possibility to add far distant cluster members.

In an NSD configuration there are typically only a few servers attached to the SAN with user data. These servers are called NSD servers and provide access to the user data for the servers that are not connected to the SAN. These are called NSD clients and must have LAN access in order to do network block I/O to the NSD servers. For the NSD clients user data and Spectrum Scale control information flows over the TCP/IP network first. Physically reading or writing user data to the SAN disks is done on behalf of the NSD servers that trigger the SAN disk operations.

For the SUT: When the SUT (shown in Figure 8 on page 29) was set up for NSD, the four ECM nodes played the role of NSD clients and had no access to the SAN disks. Three additional new nodes were added to the SUT as NSD servers that had SAN access.

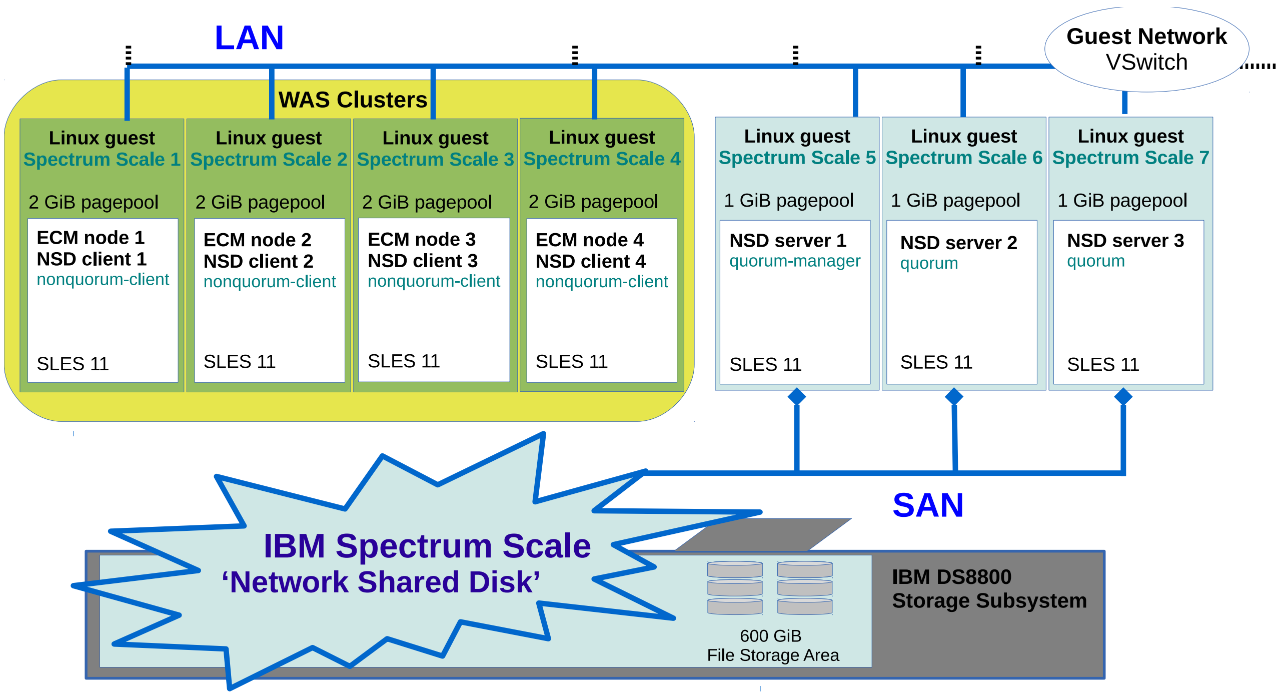


Figure 8. ECM cluster with Spectrum Scale NSD cluster configuration

Note: The network that is used to transfer data and control information in case of NSD does not need to be dedicated to Spectrum Scale. However, the network bandwidth must be sufficient to meet the goals of Spectrum Scale and any other application sharing the same network.

The following Spectrum Scale list cluster command output shows the layout of the NSD cluster for the SUT with the node designations for the ECM nodes (cluster members). At the first sight one cannot directly see that this is an NSD cluster configuration apart from the node naming scheme.

`mmlscluster` command output for the SUT Spectrum Scale NSD cluster NSD clients (ECM nodes) and NSD servers:

```
# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         ECM_4_node
  GPFS cluster id:           714383681xxxxxxxxx
  GPFS UID domain:           ECM_4node
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
---------------------------------------------------------------------
   1   ECMnod1          10.xxx.xx.xx  ECMnod1
   2   ECMnod2          10.xxx.xx.xx  ECMnod2
   3   ECMnod3          10.xxx.xx.xx  ECMnod3
   4   ECMnod4          10.xxx.xx.xx  ECMnod4
   5   NSDsrv1          10.xxx.xx.xx  NSDsrv1         quorum-manager
   6   NSDsrv2          10.xxx.xx.xx  NSDsrv2         quorum
   7   NSDsrv3          10.xxx.xx.xx  NSDsrv3         quorum
```

The Spectrum Scale cluster had now seven nodes in total (`ECMnod[1-4]` and `NSDsrv[1-3]`).

- ECMnod[1-4] were the NSD clients and the newly added NSDsrv[1-3] virtual machines were the NSD servers.
- NSDsrv1 had the node designation quorum-manager and acted as the filesystem manager for the cluster and was also in the node pool from which quorum was derived.
- NSDsrv2 and NSDsrv3 complemented the node quorum and were both in the pool of quorum nodes.
- ECMnod[1-4] had the node designation nonquorum-client which was listed as an empty string in the designation column for the mmlscluster command.

Therefore, ECMnod[1-4] had a client role and NSDnod[1-3] server roles.

The **mmlsnsd** command displays Network Shared Disk information for a Spectrum Scale cluster. The NSD servers column lists the NSD servers which provide SAN access.

The following example shows the FileNet File Storage Area NSD disks with their NSD servers (an “unbalanced server list”):

```
# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 FSData        nsd14cb      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd15cb      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd14cc      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd15cc      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd14cd      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd15cd      NSDsrv1,NSDsrv2,NSDsrv3
```

In the above example, the NSD disks for the filesystem FSData (used as the FileNet File Storage Area) were SAN-attached to the NSD servers (NSDsrv[1-3]).

However, the NSD setup shown above has a certain disadvantage. All six NSD disks had NSDsrv1 as the first NSD server in the list. NSDsrv[2,3] were only used when the precursor server in the list was not available. So the example above showed a failover configuration with an *unbalanced* NSD server list.

Exploring some tuning options

Another way to define the NSD disks is to mix the order of the NSD servers in the list across the available NSD servers. In doing so, this allows a simple but effective I/O striping across the NSD servers in the case when all servers are available at the same time (see the output from **mmlsnsd** in the example that follows). This setup is called a *balanced NSD server list*.

This example shows NSD disks for the File Storage Area network that are attached to the ECM nodes (a “balanced NSD server list”):

```
# mmlsnsd

File system   Disk name    NSD servers
---------------------------------------------------------------------------
 FSData        nsd14cb      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd15cb      NSDsrv3,NSDsrv1,NSDsrv2
 FSData        nsd14cc      NSDsrv2,NSDsrv3,NSDsrv1
 FSData        nsd15cc      NSDsrv1,NSDsrv2,NSDsrv3
 FSData        nsd14cd      NSDsrv3,NSDsrv1,NSDsrv2
 FSData        nsd15cd      NSDsrv2,NSDsrv3,NSDsrv1
```

In this example, the order of the servers in the NSD servers list was different to that of the unbalanced NSD server list.

- The six NSD disks had different first NSD servers in the list. As a result, each NSD server now served two NSD disks out of six, and disk I/O was done in parallel.
- In the unbalanced example, a single NSD server did all the disk I/O to the six NSD disks.

The following example shows the creation of NSD disks with balanced NSD servers via **mmcrnsd** using a stanza file:

```
# cat /data/FNP8_GPFS/nsd-disks.txt
%nsd:
  device=/dev/mapper/scsi14cb
  servers=NSDsrv1,NSDsrv2,NSDsrv3
  nsd=nsd14cb
  usage=dataAndMetadata
%nsd:
  device=/dev/mapper/scsi15cb
  servers=NSDsrv3,NSDsrv1,NSDsrv2
  nsd=nsd15cb
  usage=dataAndMetadata
...

# mmcrnsd -F /data/FNP8_GPFS/nsd-disks.txt
mmcrnsd: Processing disk mapper/scsi14cb
mmcrnsd: Processing disk mapper/scsi15cb
...
```

Figure 9 on page 32 shows the outbound I/O traffic (disk read and network transmitted) for the NSD servers during the execution of the ECM workload. For the unbalanced NSD servers, a single NSD server performed all of the disk and network I/O. In contrast, in the case of the balanced NSD servers the disk and network I/O was equally distributed across the three servers.

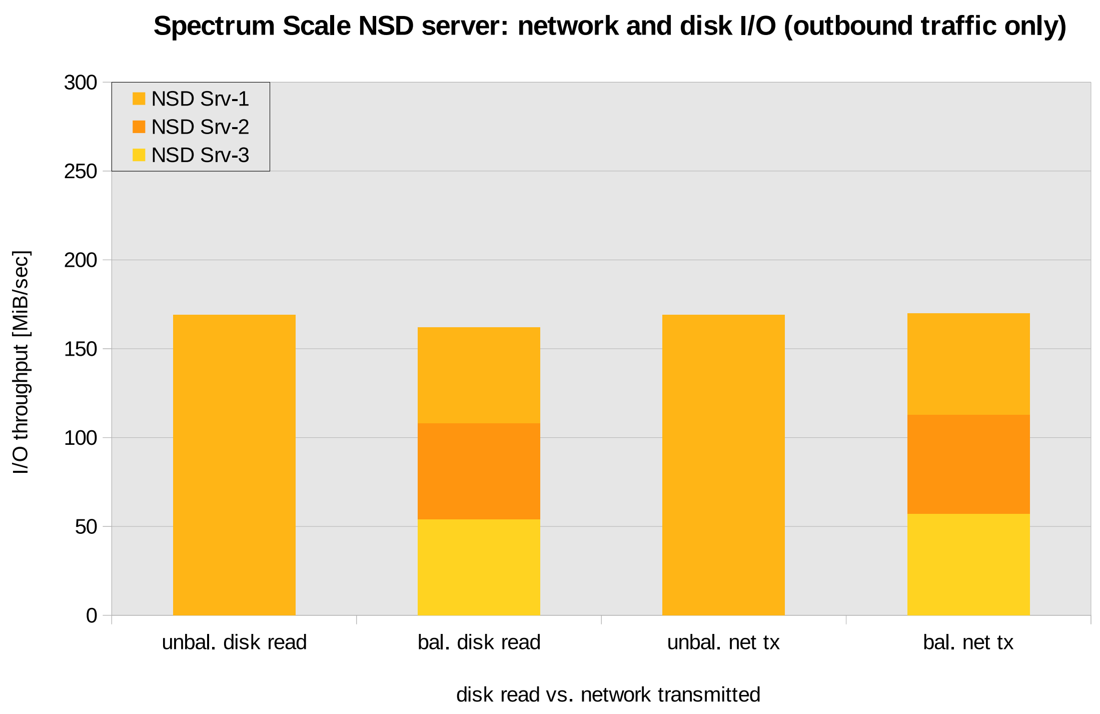


Figure 9. Network and disk I/O for NSD servers (unbalanced vs. balanced server list)

In addition, Figure 9 shows that the disk and network I/O rates were almost equal on the NSD servers. This implies that a dedicated NSD server simply forwards both the data that is read from the SAN and the data that originates from the NSD clients. Therefore, this behavior was also true for the inbound I/O traffic (for the disk write and network received).

The **mmlsmount** command lists the cluster members that have mounted a particular filesystem. In this command example the filesystem FSData was mounted on all seven cluster members (NSD servers and NSD clients). However, the filesystem does not necessarily have to be mounted on the NSD servers.

```
# mmlsmount FSData -L

File system FSData is mounted on 7 nodes:
  10.xxx.xx.xxx   NSDsrv1
  10.xxx.xx.xxx   NSDsrv2
  10.xxx.xx.xxx   NSDsrv3
  10.xxx.xx.xxx   ECMnod1
  10.xxx.xx.xxx   ECMnod2
  10.xxx.xx.xxx   ECMnod3
  10.xxx.xx.xxx   ECMnod4
```

NSD configuration study of pagepools in an NSD cluster

Every node in a Spectrum Scale cluster has its own pagepool. The pagepool can vary in size for each node according to the intended role or task of the node.

The following short side study analyzes the pagepool efficiency in a NSD cluster. For the cluster configuration SD pagepooling is easier to understand, because the relation of directly attached SAN disks and pagepools on the same node is more obvious.

However in a NSD cluster there are two types of roles in a cluster, NSD servers and NSD clients each having a pagepool. The following pagepool measurement series was done to investigate the pagepool effects on the server and on the client nodes in a Spectrum Scale NSD cluster.

The NSD disk read rate on the NSD servers was used as a metric to qualify the efficiency of the pagepools.

Measurement series 1: The pagepools were scaled from 256 MiB to 2 GiB at the same time on all three NSD servers, whilst the pagepools on the four ECM nodes (NSD clients) were kept at a fixed size of 2 GiB on each ECM node.

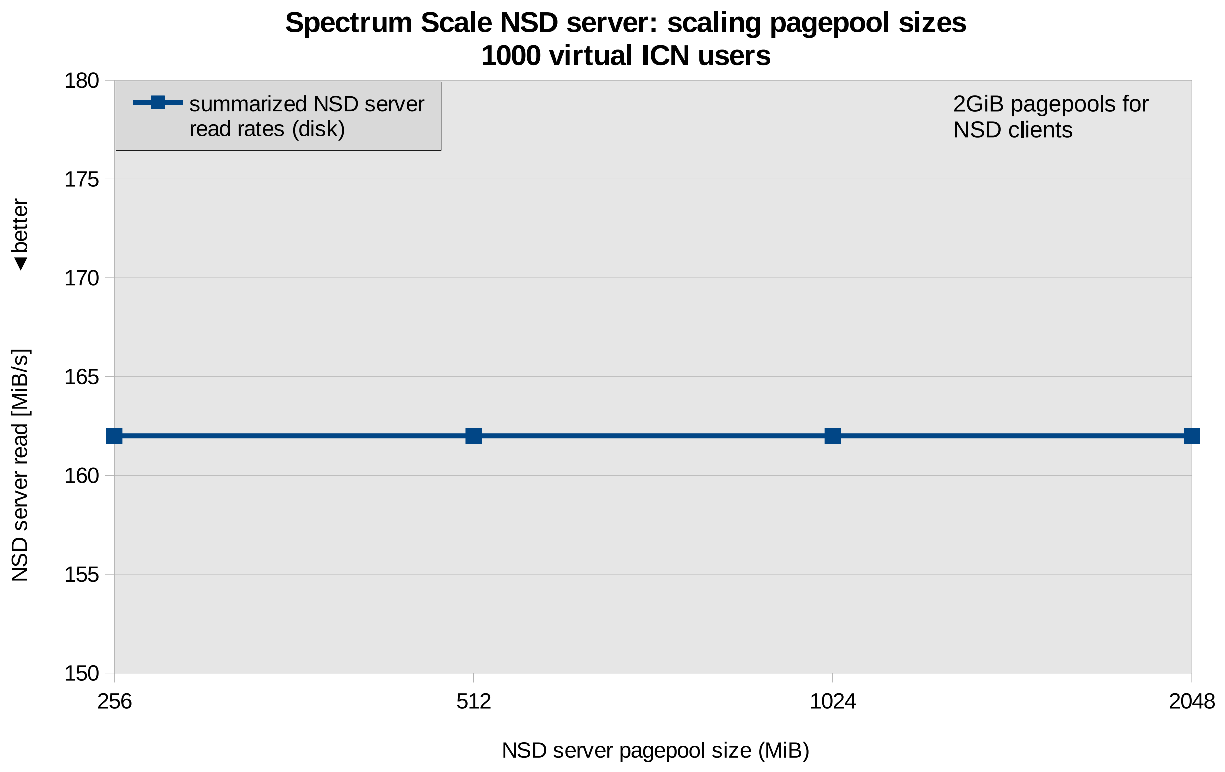


Figure 10. Scaling pagepool sizes for Spectrum Scale NSD servers

Figure 10 shows the read disk I/O rates in total (summarized for all three NSD servers) for different pagepool sizes on the NSD servers. The disk I/O read rates on the NSD servers remained at a constant level independent of the pagepool size on the NSD servers.

Measurement series 2: The pagepools were scaled from 256 MiB to 2 GiB on the four ECM nodes (NSD clients), while the pagepools on the NSD servers were kept at a fixed size of 2 GiB.

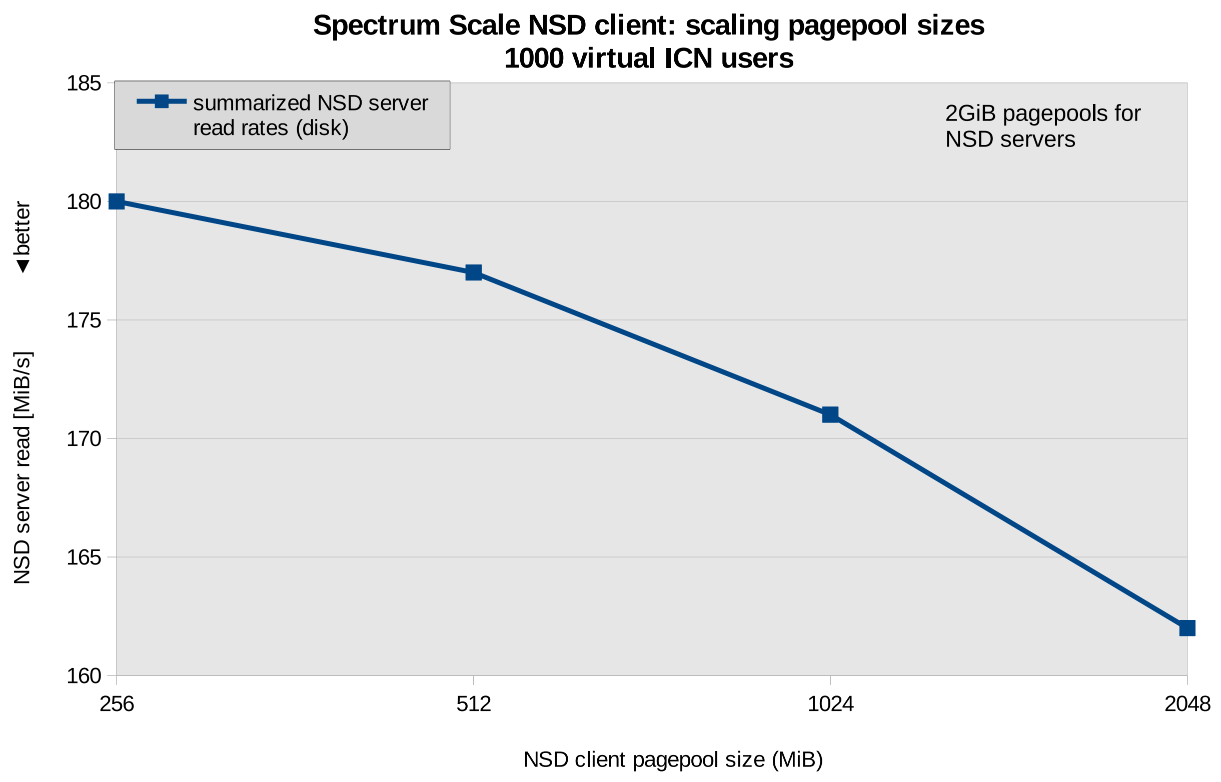


Figure 11. Scaling pagepool sizes for Spectrum Scale NSD clients

Figure 11 shows the size scaling for the NSD client pagepools in a second measurement series. The NSD clients do network block I/O and no disk I/O for Spectrum Scale filesystem. Therefore, the metric to look at is still the disk I/O read rate for the NSD servers.

In contrast to the NSD server pagepool size scaling, there was now a dependency of the NSD client pagepool sizes and the NSD server disk I/O read rates:

- The NSD server read rates started at 180 MiB/sec for 256 MiB and gradually decreased to 162 MiB for a 2 GiB pagepool.

Conclusion

Spectrum Scale cache usage is most effective on the nodes where the data processing occur. Typically, this is on the NSD clients when the NSD servers are used for I/O processing only. In that case larger pagepools are less important for NSD servers. The reason for this behaviour is that pagepools on NSD servers are not used for any application data caching but for I/O buffering. The larger the NSD client pagepool, the less disk I/O was done by the NSD servers.

For the SUT: The pagepool size was set to 2 GiB on all four ECM nodes (NSD clients). This was the same pagepool size as for the ECM nodes in the SD cluster configuration. The pagepool size was set to 1 GiB (default value) for the NSD servers. These pagepool sizes were used for the scale-out study for the NSD cluster configuration.

The **mmlsconfig** command can be used to query the pagepool sizes in a Spectrum Scale cluster. Note that now *two* different sizes were reported for NSD servers and clients:

```
# mmlsconfig pagepool
pagepool 2G
pagepool 1G [NSDsrv1,NSDsrv2,NSDsrv3]
```

Comparison of SD and NSD

The first figure shows the average ICN transaction response times in milliseconds at a high load level (1000 virtual ICN users) for the SD and NSD cluster configurations.

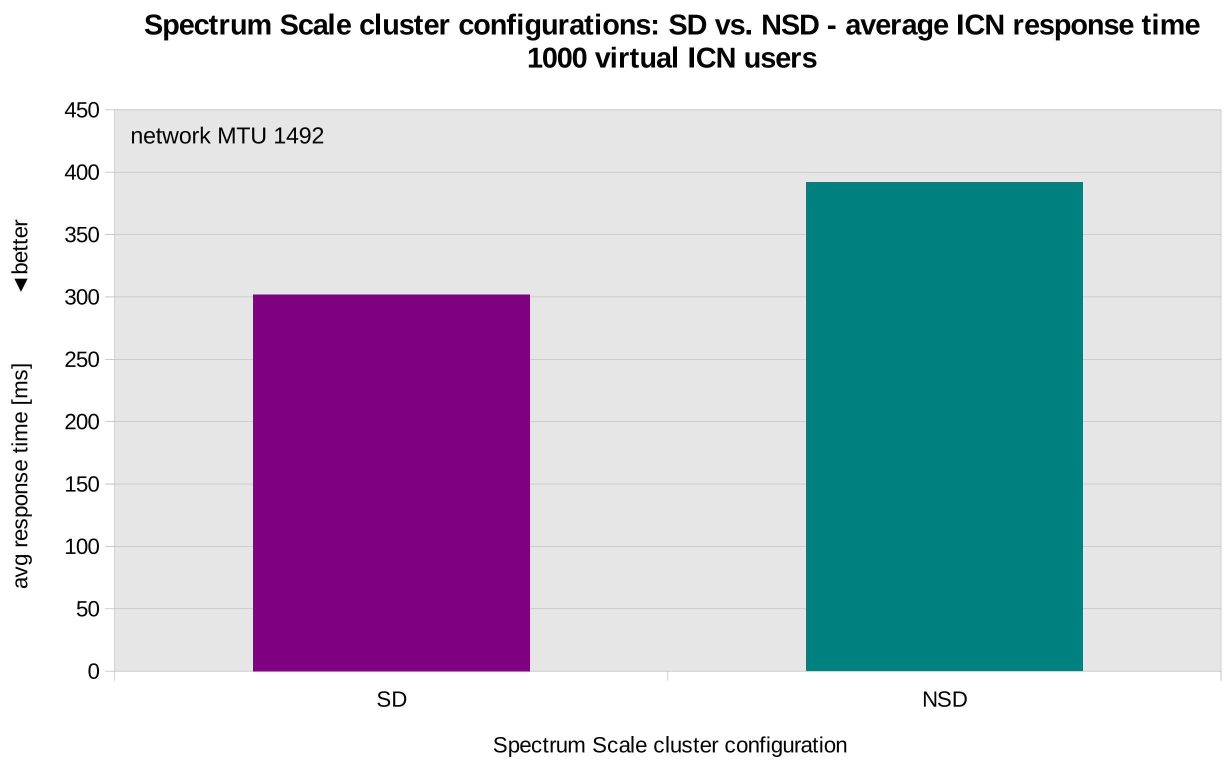


Figure 12. Average ICN response times for Spectrum Scale cluster configurations

As shown in Figure 12:

- The ICN transaction response time was around 30% higher for NSD compared to SD. These findings support similar findings that the SD cluster configuration provides good performance for smaller Spectrum Scale clusters.
- When looking at the hypervisor CPU load (z/VM total load), at the highest workload level (with 1000 virtual ICN users) the additional overhead for NSD showed an increase of 1.7 IFL processors for the complete SUT.
- For SD, the complete SUT consumed around 24 IFL processors.
- For NSD, the complete SUT consumed around 25.7 IFL processors.

Figure 13 shows how the CPU costs varied for the cluster configurations.

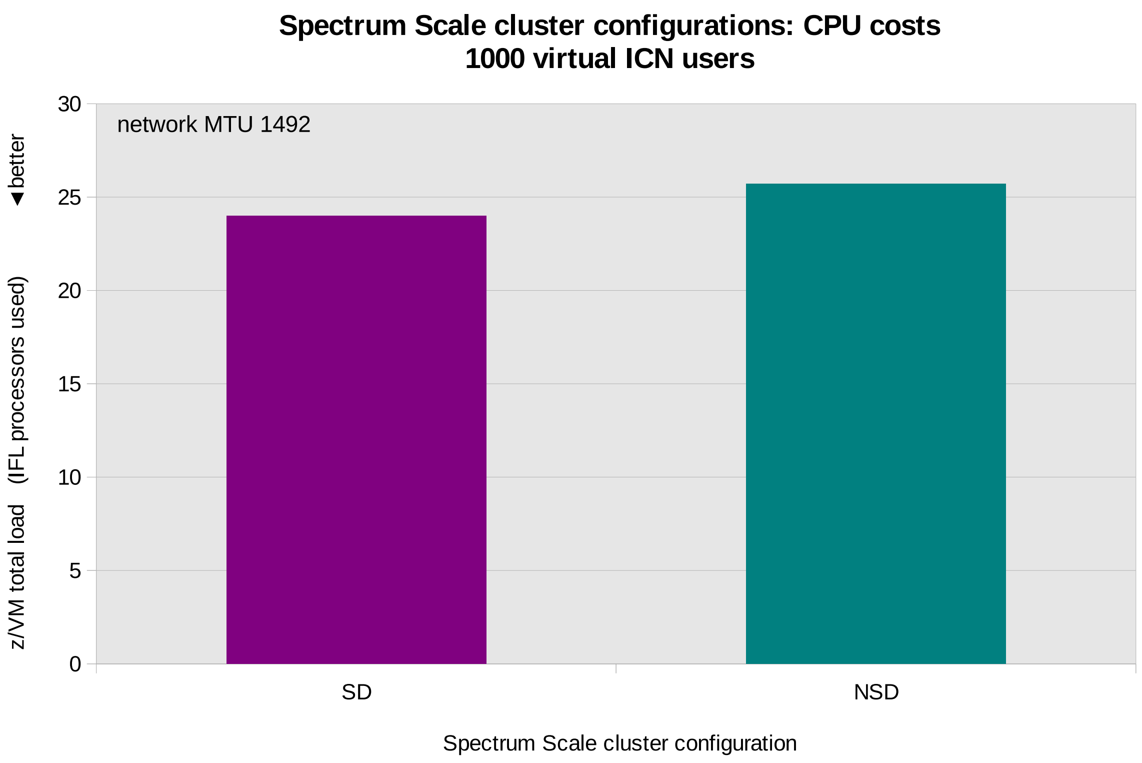


Figure 13. CPU costs for Spectrum Scale cluster configurations

As shown in Figure 13, the NSD cluster configuration introduced some overhead due to network block I/O and additional control information flow between the NSD clients and servers. When implementing NSD in the SUT, there were three additional z/VM virtual machines added as NSD servers.

Figure 14 on page 37 shows how the network block I/O overhead for NSD became apparent when looking at the Linux network packet rate metrics. It shows the stacked Linux network packet rates (received [rx] + transmitted [tx]) for a single ECM node (NSD client) out of four. The other three ECM nodes showed the same network packet rates.

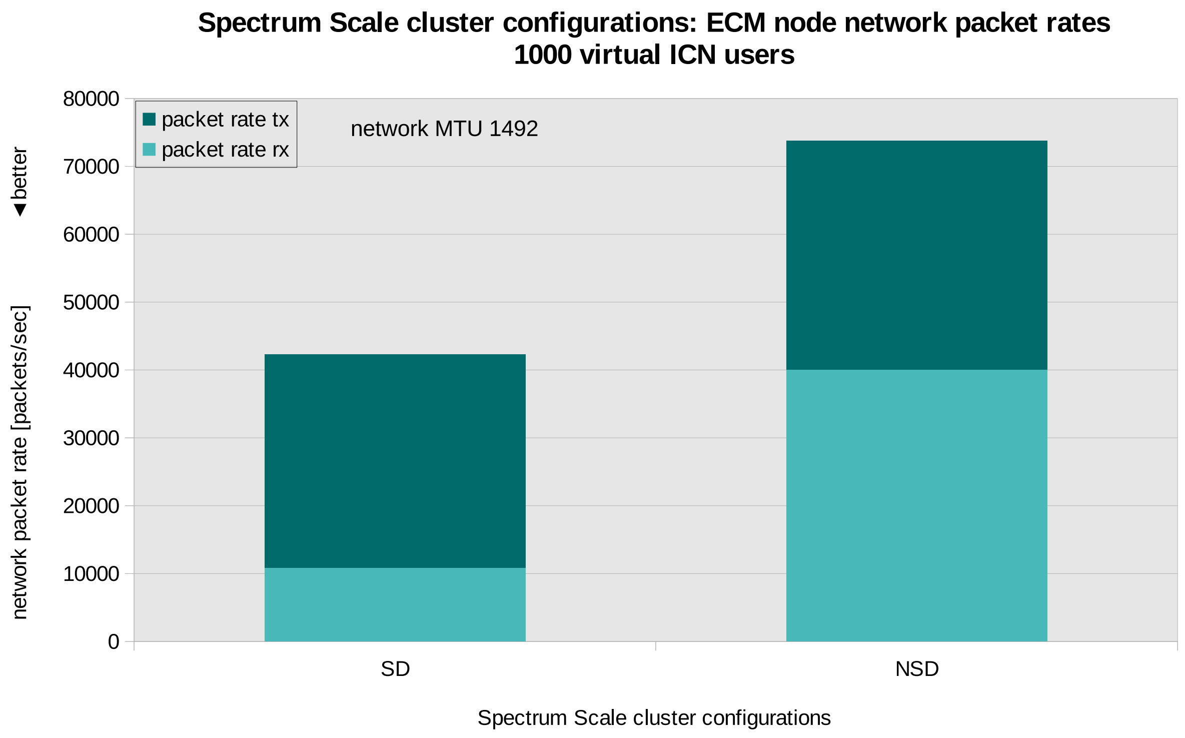


Figure 14. ECM node (NSD client) Linux network packet rates for SD and NSD

Specifically in Figure 14:

- For NSD there was a major increase of Linux network packets from 40,000 packets/sec to over 70,000 packets/sec. This was an increase of 74% for NSD.
- The rx network packet rate (inbound traffic) was nearly 4 times higher compared to the SD rate.
- This increase of incoming network packets can be explained with NSD network block I/O done for reading ECM content data from the NSD servers.

Tuning considerations:

- Because of the increase of network I/O for NSD, it was worth looking at other possibilities to tune the network.
- The measurement series for the above Spectrum Scale cluster configuration comparison was done with the default network maximum transmission unit (MTU) of 1492 bytes.
- If the bandwidth for the Spectrum Scale network is not sufficient, it can quickly become a bottleneck for a NSD cluster. Therefore, a common Spectrum Scale recommendation is to consider a larger MTU (jumbo frames) for the Spectrum Scale network. This consideration is discussed in more detail in the next chapter.

Spectrum Scale network block I/O

The comparison of the SD and NSD Spectrum Scale cluster configurations, that was described in the previous main topic, showed an additional overhead for NSD.

The major cause of the overhead for NSD was introduced with the *network block I/O layer*. This layer was used for exchanging data between the NSD client and NSD server nodes.

A well known tuning recommendation for NSD with its network block I/O was the usage of a larger MTU (*jumbo frames*) for the Spectrum Scale network.

Jumbo frames

Jumbo frames are Ethernet frames with a payload greater than the standard maximum transmission unit (MTU) of 1500 bytes.

The standard MTU must be set to 1492 bytes when using *Point To Point Protocol over Ethernet* (PPPoE) connectivity.

To enable jumbo frames for testing with the ECM workload, the MTU was set to 8192 for the complete SUT. This included the workload driver machine running on x86_64.

Jumbo frames can be easily enabled in Linux by setting a larger MTU in the network configuration file for a network interface. The network interface needs to be restarted after a MTU change to become effective.

Note: Best network performance can be achieved when the entire network and also the physical network switches support jumbo frames.

The following examples show the jumbo frame MTU for a network interface:

```
# cat ifcfg-eth1
...
MTU='8192'
...

# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 02:8C:44:00:00:06
          inet addr:10.196.39.122  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::28c:4400:100:6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:8192  Metric:1
          RX packets:701 errors:0 dropped:0 overruns:0 frame:0
          TX packets:925 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:63956 (62.4 Kb)  TX bytes:93435 (91.2 Kb)
```

For information about additional TCP/IP tunings that might be effective for very highly utilized network interfaces, refer to the article “Linux on System z® – Tuning hints and tips” at this IBM developerWorks® URL:

`http://www.ibm.com/developerworks/linux/linux390/perf/tuning_networking.html`

Results of network block I/O tuning

The series of graphics provided in this topic show the results of network block I/O tuning for Spectrum Scale. In particular, the graphics provide a comparison of different MTU sizes.

Figure 15 on page 39 shows a comparison of using the default MTU size (1492 bytes) and jumbo frames (8192 bytes) either SD or NSD. The performance metrics were the average ICN transaction response time at the highest load level.

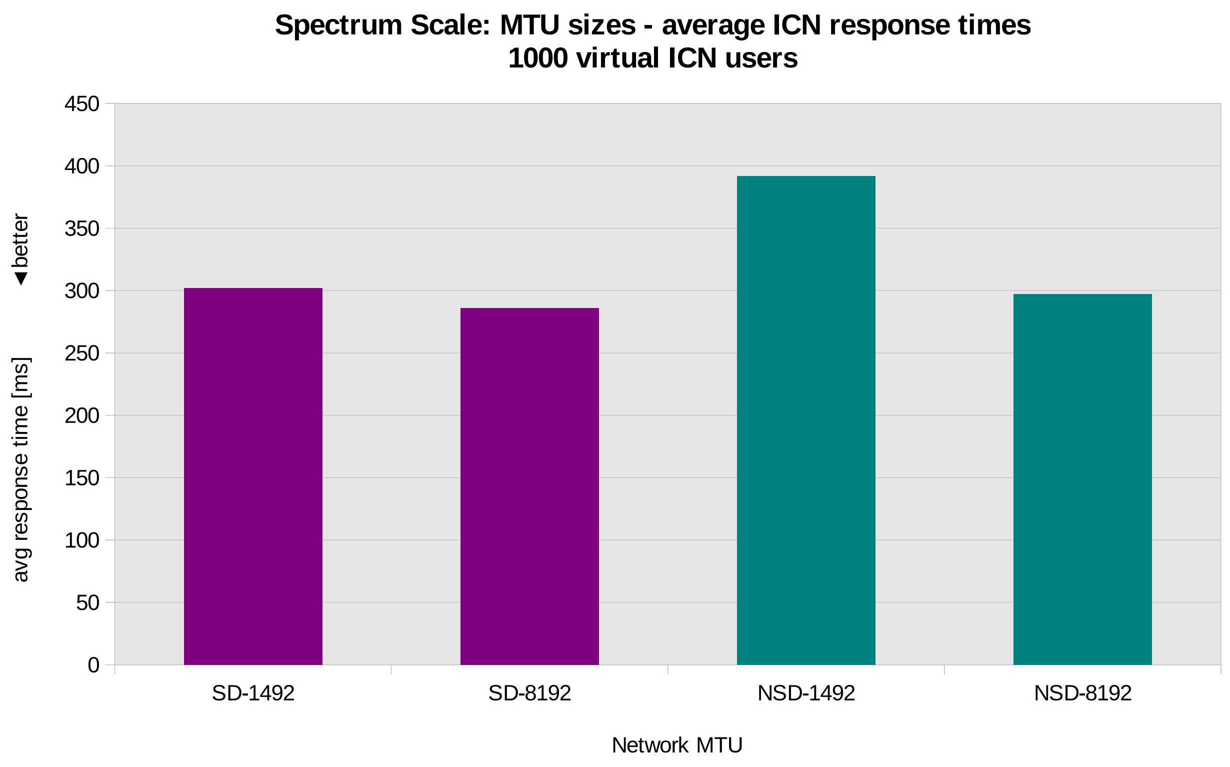


Figure 15. Average ICN response time for different MTU sizes

As shown in Figure 15, the Spectrum Scale cluster configuration SD did not significantly benefit from the use of jumbo frames.

- The average transaction response time dropped to below 300 ms with jumbo frames. For NSD, the improvement with jumbo frames was much larger.
- The average transaction response time dropped to 300 ms, and was then at the same level as Spectrum Scale SD with default MTU.
- When looking at the hypervisor CPU load (z/VM total load), there was also a CPU cost savings with the usage of jumbo frames.

Figure 16 on page 40 shows the CPU loads for the z/VM hypervisor.

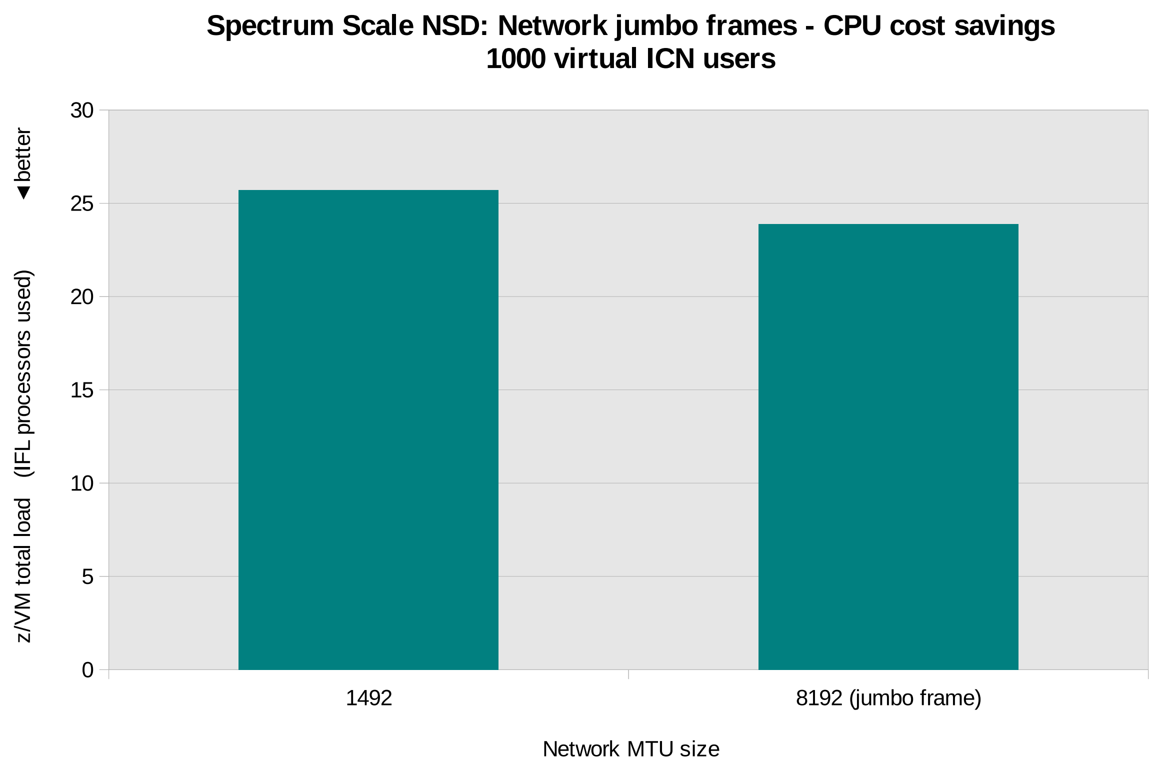


Figure 16. CPU cost savings (hypervisor view) for Spectrum Scale NSD when using jumbo frames

As shown in Figure 16:

- By using jumbo frames for the complete SUT, the hypervisor load (total z/VM load) was reduced by 1.7 IFL processors.
- The CPU cost savings was the result of less networking overhead with the usage of jumbo frames.
- The processing for the z/VM virtual network (VSWITCH based) was done by IFL processors.
- When using jumbo frames, additional payload data fits into a single network packet. As a result, the Linux network packet rate dramatically went down.

Figure 17 on page 41 further shows the results of using jumbo frames. It shows Linux network packets rates (packets/sec) and average packet size (bytes/packet). The listed rates and sizes were for *one ECM node* in the ECM cluster. However, the reported metrics would quadruple if the entire ECM cluster was considered.

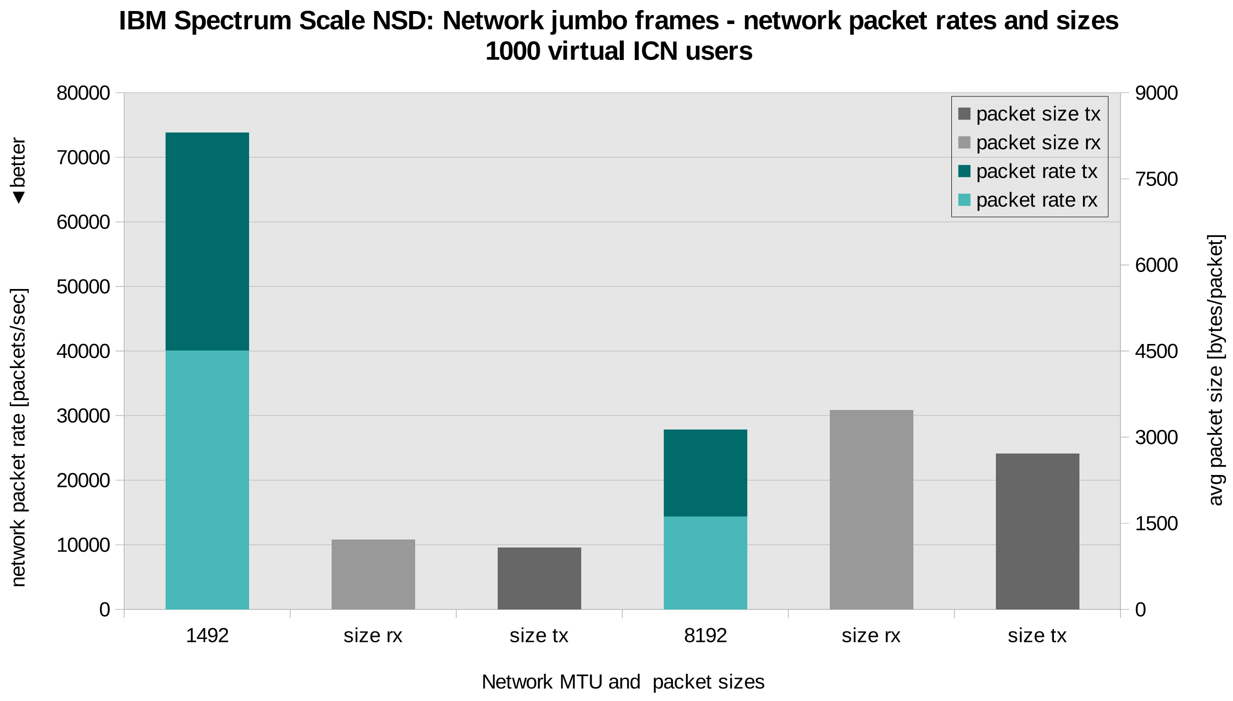


Figure 17. Network jumbo frames – network packet rates and sizes on ECM nodes

As shown in Figure 17:

- The packet rates and sizes were shown for the default MTU size (1492) and for jumbo frames (8192). Furthermore, the network packet rate was a stacked number for the received (rx) and transmitted (tx) packets displayed as a single bar.
- The packet sizes for the rx and tx packets are shown as two grey bars next to the packet rate bars.
- With jumbo frames enabled, the Linux network packet rate was reduced to 30,000 packets/sec compared to more than 70,000 packets/sec when the default MTU was used. This corresponded to a packet rate reduction of *more than 60%*.
- If the received and transmitted packet size metrics for the default MTU are examined, the network packets were almost completely using the available payload (packet size vs. MTU size). With jumbo frames, the network packet sizes were *2.5 times to 3 times larger*.
- The payload data for a single network packet was larger. As a result, less network packets were needed to transport the same amount of user data.

Only enabling jumbo frames for z/VM virtual networking

Jumbo frames are sometimes not available throughput an entire LAN infrastructure. Virtual networking for the z/VM hypervisor was provided with a Virtual Switch (VSWITCH) for the SUT. Therefore, network traffic between the z/VM virtual machines was inside the scope of the hypervisor.

Since all Spectrum Scale network block I/O was between virtual machines of the same z/VM, enabling jumbo frames for these virtual machines did not cause any difficulties. The workload driver machine running on Linux on x86_64 was kept at the default MTU (1500) for this side study.

Therefore, any network traffic between the virtual machines of the z Systems server and the external workload machine could not use jumbo frames.

In Figure 18, three measurements are compared:

- The left bar shows the average ICN transaction response time when jumbo frames were enabled for the complete SUT (including the x86_64 workload driver machine).
- The bar in the middle shows the response time when jumbo frames were enabled only inside z/VM.
- The rightmost bar shows the default networking MTU (1492) throughout the SUT.

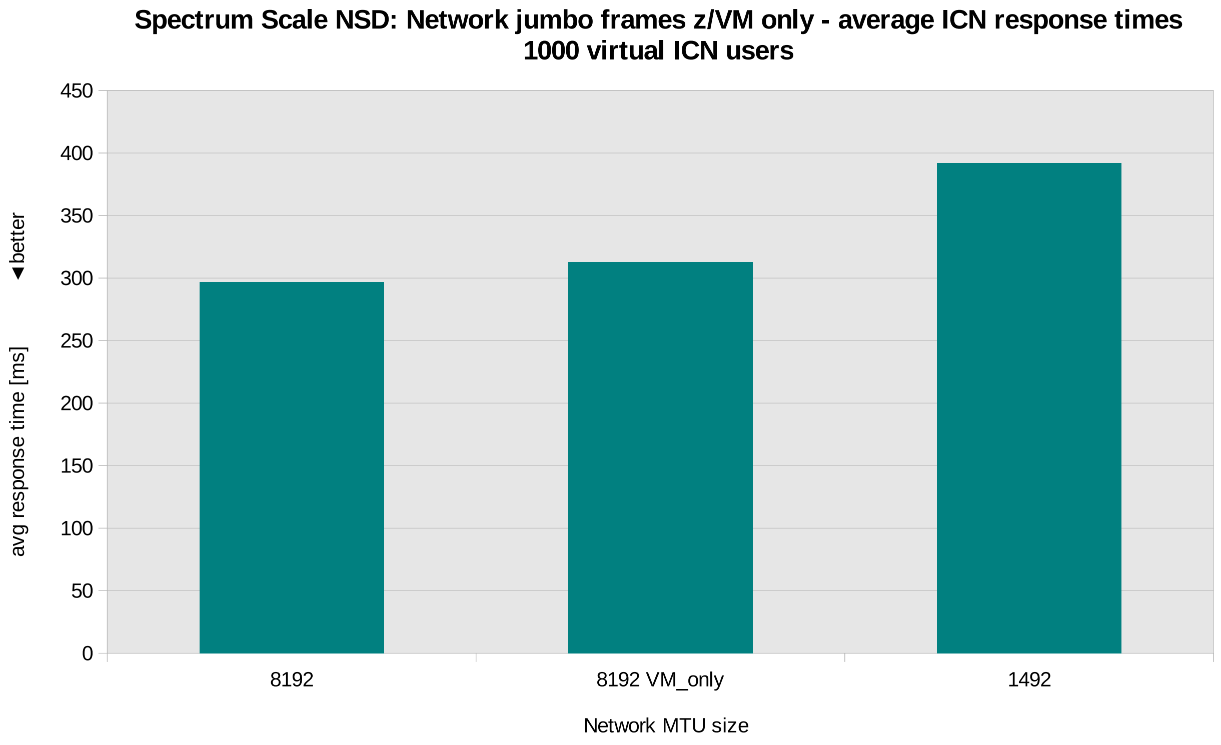


Figure 18. Jumbo frames for virtual networking only – average ICN response time

There was only a *5% increase* of the response time when jumbo frames were enabled for z/VM only compared to jumbo frames for the complete SUT. This was still a significant improvement in comparison to the response time with the default network MTU.

Conclusion

Even if the entire network infrastructure (that is, network switches or other servers) does not support jumbo frames, it *can still be beneficial to enable jumbo frames for z/VM virtual networking only.*

Jumbo frames *can help to improve throughput or response times when enabled for Spectrum Scale network block I/O.*

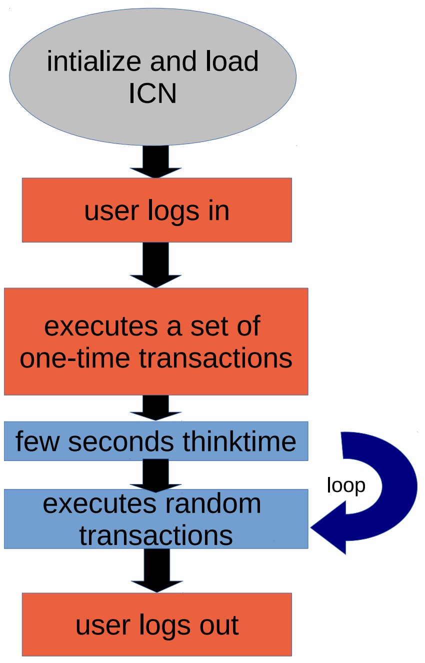
Chapter 6. ECM workload

This topic contains a summary of the terms and definitions that are used in the next main topic.

You should also refer to the Part 1 White Paper for a detailed description of the workload used in this case study.

The IBM Rational Performance Tester (RPT) product was used to emulate the virtual users. The *think time* is the time during which the virtual user was idling for a few seconds before starting the next transaction. Because of the short think times in the RPT scripts that were used, each virtual user generated the load-equivalent of many human end-users who would use ICN to accomplish real work tasks.

The stable workload phase was reached when all virtual users were logged in and were executing their transaction loop. As shown in Figure 19, between the transaction execution was a think time. The metrics transaction throughput and average response time were taken from this stable workload phase.



SUT load level z/VM total load (IFL processors)	Virtual ICN users
low < 10 IFLs used	250
medium 10-15 IFLs used	500
medium-to-high 15-20 IFLs used	750
high > 20 IFLs used	1000

Figure 19. Virtual user activity flow and workload levels

The more virtual users were logged in parallel during the stable workload phase. The smaller the think time, the higher was the workload level for the SUT.

The z/VM hypervisor had 30 IFL processors in total and for the case study four load levels were defined. Starting from low to medium, medium-to-high and high (shown in Figure 19). These load levels were used throughout the scale-out study.

Chapter 7. Scale-out study results

For the measurement series discussed in this topic, all of the described setup and tuning was applied to the SUT. To summarize, in this topic the measurement series scaled the number of virtual ICN users from 250 to 1000 for the four-node ECM WAS cluster setup.

This topic uses the ECM workload (with its different SUT load levels) that was described in Chapter 6, “ECM workload,” on page 43.

One of the major objectives for the scale-out case study was the performance of a *clustered multiple ECM node environment* (that used WAS and Spectrum Scale clusters) versus a *single ECM node setup*. For this purpose, the ECM clusters were scaled-out from a single node to a four-node cluster during this study. Spectrum Scale as parallel filesystem played a major role in this type of clustered multiple ECM node setup.

The ICN transaction throughput depended upon the number of virtual ICN users and the think time between transactions. The ECM cluster had four nodes with 8 virtual CPUs per node (that is, 32 virtual CPUs for the entire ECM cluster).

Note: For the scale-out case study, the average ICN transaction response time over all ICN transactions was considered as performance indicator.

The ratio of CPU overcommitment was:
- **1:1.6** (30 IFL processors versus 48 virtual CPUs) with Spectrum Scale SD.
- **1:1.8** (30 IFL processors versus 54 virtual CPUs) with Spectrum Scale NSD.

Note that the SUT z/VM hypervisor had 30 IFL processors in total.

Table 20. SUT load level definitions

SUT load level	# of virtual ICN users	z/VM total load (IFL processors used)
low	250	< 10 IFLs (0-33%)
medium	500	10 - 15 IFLs (33-50%)
medium-to-high	750	15-20 IFLs (50-66%)
high	1000	> 20 IFLs (66-100%) - see **Note** below

Note: The highest z/VM IFL processor usage was 80% (24 out of 30 IFL processors used) for the SUT with 1000 virtual ICN users and Spectrum Scale NSD.

ECM user scaling for Spectrum Scale SD

For the Spectrum Scale measurement series, the shared FileNet Advanced File Storage Area resided on a Spectrum Scale filesystem.
- The four ECM nodes belong to a WAS cluster and a Spectrum Scale cluster configured as SD.
- The SAN disks for the File Storage Area were directly attached to the ECM nodes.
- Each ECM node ran ICN and FNP8 CPE in two separate WAS clusters.

- Both applications formed an application pair on each ECM node.

For a graphical display of this aspect of the SUT, see Figure 2 on page 9.

Transaction throughput and average ICN response time

This topic shows the normalized ICN page throughput rate for different SUT load levels. The ICN page throughput rates increased as the number of virtual ICN users increased. The ICN transaction response time was used as an indicator of performance.

The blue bars shown in Figure 20 represented the average ICN transaction response times for the following configurations:

- Single ECM node with File Storage Area on XFS.
- Single ECM node with File Storage Area on Spectrum Scale (SD).
- Scale-out to four ECM nodes with File Storage Area on Spectrum Scale (SD).

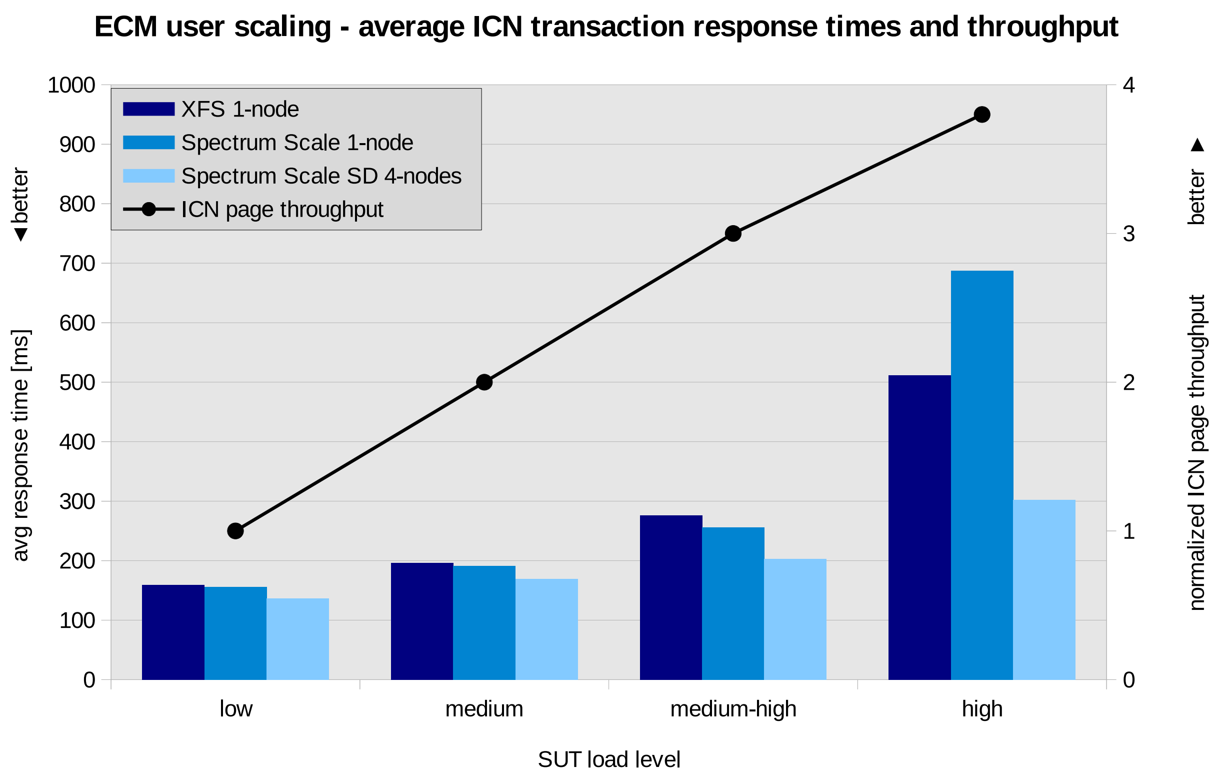


Figure 20. ECM user scaling - ICN page throughput and average response times

Observations

The throughput rate showed an almost linear characteristics correspondent to the increasing SUT load level (number of virtual ICN users), starting at a low load level (250 users) and increasing up to a high load level (1000 users).

The average ICN transaction response time started at 150 ms for the low load level, and increased up to 700 ms for the high load level for a single ECM node with Spectrum Scale. Up to the medium-high load level the setups with Spectrum Scale showed better response times than a single ECM node with XFS.

However, for *all load levels* the ECM cluster showed the *best average response times*.

- At the highest load level, the average response was still at 300 ms compared to the single ECM node setups that showed response times from 500 up to 700 ms.
- The response-time improvements with a ECM cluster varied from *40% to 56%*, compared to the single ECM node setups.

Conclusion

For this particular ECM workload, an average response time below 1 second was considered as good. For all the setups that were tested, the average response times remained *significantly below that requirement*. This proves conclusively how good a z Systems environment is for handling such workloads.

The scale-out solution to a multiple ECM cluster using Spectrum Scale showed the *best performance* of all tested setups. Especially at the higher load levels, the performance of the ECM cluster solution greatly benefited from the request balancing for the ECM nodes. Spectrum Scale *scaled perfectly* when used with such a clustered environment.

CPU loads for ECM nodes

The CPU loads for a single ECM node versus an ECM cluster were compared across the SUT load levels.

Figure 21 on page 48 shows the CPU loads for the individual ECM nodes. For the ECM cluster, the CPU loads were displayed as “stacked”.

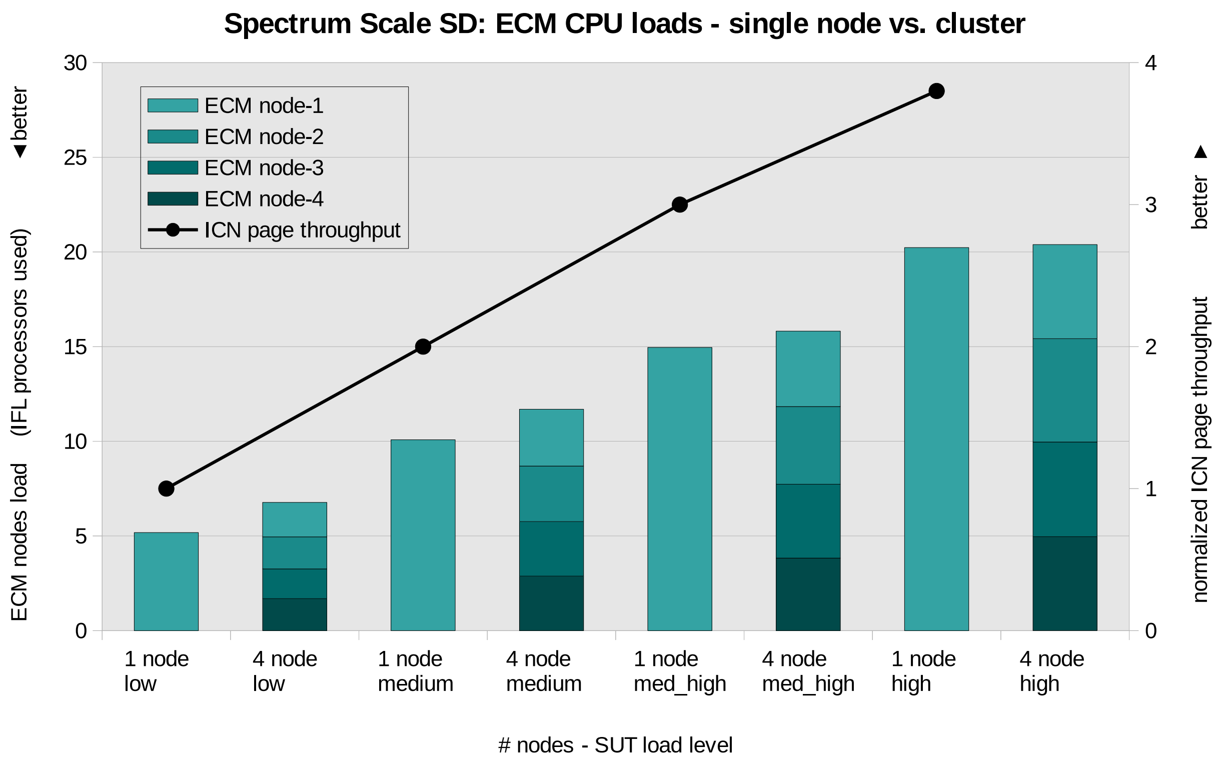


Figure 21. ECM CPU loads: single ECM node versus ECM cluster

Observations

- At the low load level, the CPU load used 5.2 IFL processors for the single ECM node and 6.7 IFL processors for the ECM cluster.
- At the high load level, the CPU load was around 20 used IFL processors for both node setups.
- The differences between the CPU load became less at the higher ECM node load levels.
- At the highest load level, the CPU load was equal for the single ECM node and for the ECM cluster.

Conclusion

At the lower load levels, the ECM cluster appeared to consume *slightly more CPU* compared with the single ECM node. However, the additional CPU usage became insignificant when we looked at the *average ICN transaction response times*:

- The average ICN response times were *much better* for the ECM cluster across the *complete load level bandwidth*.
- The CPU usage was equal at the highest load level for both setups, but the ECM cluster response time *was outstanding* compared to the single ECM node.
- At the higher load level, using the ECM cluster was shown to be highly advantageous in transaction performance at the same CPU cost.

Disk I/O for ECM nodes

In this topic, the disk I/O rates for the single ECM node versus the ECM cluster are compared at different load levels.

Figure 22 shows the disk I/O rates (read and write) for the individual ECM nodes. The disk I/O rates are displayed as "stacked" for the individual nodes. The read/write rates are also displayed as "stacked".

Therefore, for the ECM cluster there are:

- four values for read that are added together in a stack.
- four values for write that are added together in a stack.

Spectrum Scale SD was used for both setups, and the SAN disks were directly attached to the ECM nodes.

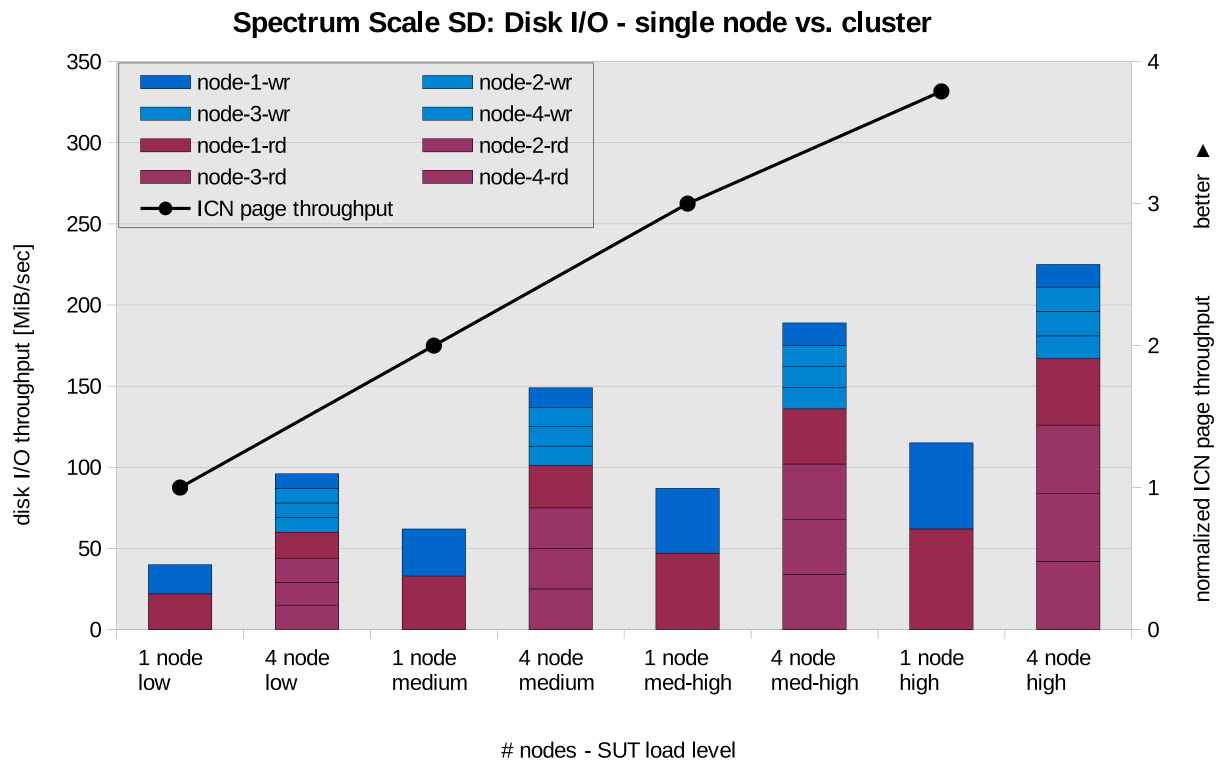


Figure 22. Spectrum Scale disk I/O - single ECM node vs. ECM cluster

Observations

- For the low load level, the disk I/O rates (read+write) were less than 50 MiB/sec for the single ECM node, and around 100 MiB/sec for all nodes of the ECM cluster.
- At the high load level, the disk I/O rates was around 115 MiB/sec for the single ECM node, and 225 MiB/sec for all nodes of the ECM cluster.
- The ECM cluster had almost double the disk I/O rates compared to the single ECM node.
- The write I/O (shown as blue bars) was almost the same level for the single ECM node as for the ECM cluster over the complete load level bandwidth.
- The read I/O (shown as red bars) was almost 3 times higher for the ECM cluster compared to the single ECM node.

Conclusion

The amount of updated or new ECM content *was similar*, independently of whether a single-node or multiple-node ECM system was being used. Therefore, the write I/O rates were similar for both setups.

However, the read I/O rates were significantly higher when using an ECM cluster.

- In a Spectrum Scale cluster with multiple nodes every node had its own pagepool, whereas there was only one unique pagepool for the single ECM node.
- In a Spectrum Scale multiple-node cluster, the effort required to synchronize the pagepools was higher. In addition, user data had to be re-read from the NSD disks more often.
- Therefore, the overall resulting read I/O was higher in the case of a ECM cluster.

ECM user scaling for Spectrum Scale NSD

In this topic, the average ICN transaction response times are compared for the ECM cluster for both Spectrum Scale cluster configurations SD and NSD.

In this measurement series, for the Spectrum Scale virtual network (z/VM VSWITCH):

- The default MTU (1492 bytes) was used for SD.
- Jumbo frames (8192 bytes) were used for NSD.

For reference purposes, the response times for the single ECM node with XFS were also listed.

In Figure 23 on page 51, the hypervisor CPU (total z/VM CPU load) for the entire SUT are shown as bars for both Spectrum Scale cluster configurations. The cluster configurations were compared for all SUT load levels, starting from low (8 IFL processors used), up to high (24 IFL processors used).

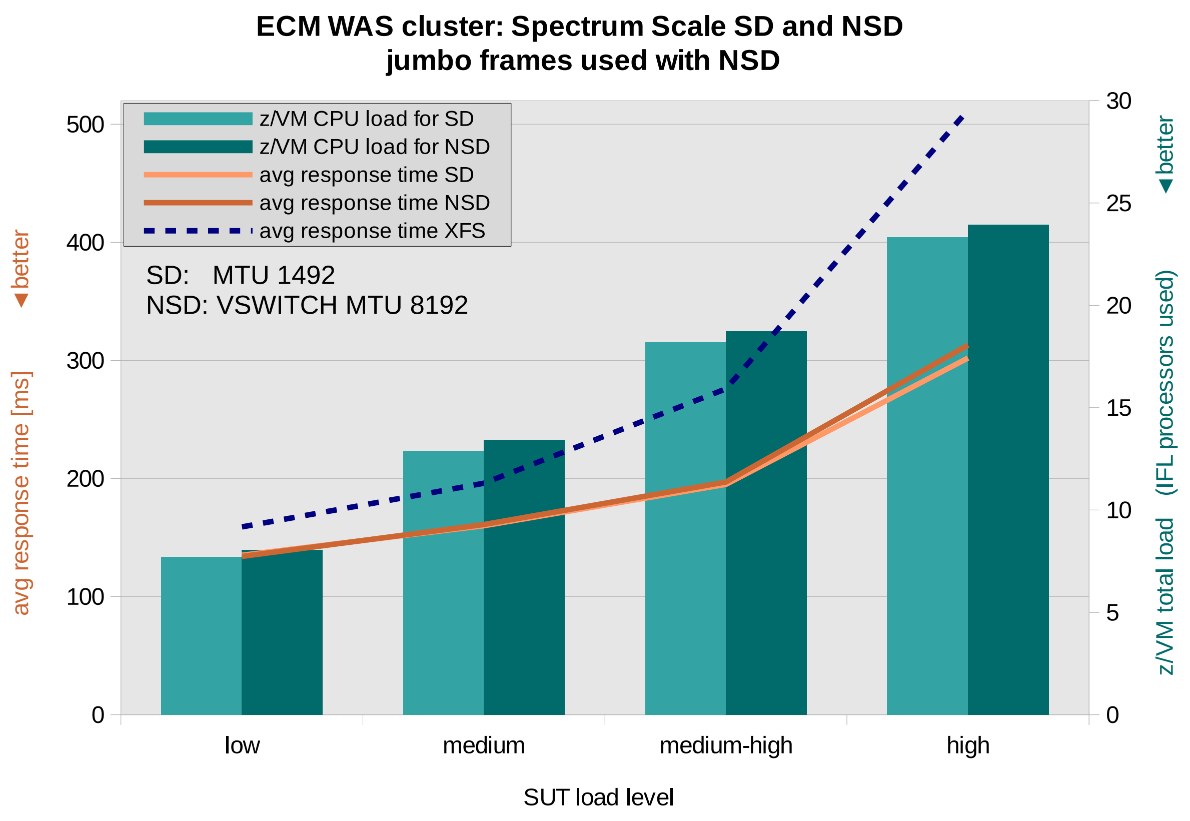


Figure 23. Spectrum Scale cluster configurations SD and NSD

Observations

The average ICN transaction response times are shown in Figure 23 as red, orange and blue dashed lines.

- For both Spectrum Scale cluster configurations (SD and NSD), the response times were *almost equal* from the low to the medium-high load levels.
- For the highest load level, the response time was *only slightly higher* for NSD but still remained around 300 ms like for SD. However, this was still *significantly lower* than for the single ECM node with XFS that showed a response time around 500 ms.

The bars in Figure 23 for the total z/VM hypervisor load were almost the same for both Spectrum Scale cluster configurations. At the high load level, the Spectrum Scale NSD setup consumed around 0.6 IFL processors more than SD.

Note: Compared to SD, the NSD cluster configuration also had three additional virtual servers and performed additional network block I/O.

Conclusion

For the ECM SUT, Spectrum Scale *worked excellently* together with the four-node ECM cluster.

- The Spectrum Scale cluster configurations SD and NSD performed at *almost the same level* when network block I/O tuning was applied for NSD.

- Especially with z/VM virtual networking, the use of network jumbo frames is beneficial for the NSD setup (see also “Spectrum Scale network block I/O” on page 37).

The additional CPU costs for NSD were moderate (less than 1 IFL in our setup).

The *best overall performance* was provided by the *four-node ECM cluster with Spectrum Scale Shared Disk* (SD). At least for this relatively small cluster, Spectrum Scale SD required less resources (since there are no additional NSD servers) and was less complicated to set up.

Once the overhead and manageability for directly attaching a SAN to an increasing number of cluster members becomes significant, *the NSD model is a good alternative*.

Chapter 8. Single ECM node with XFS and Spectrum Scale 4.2 (Part 1 white paper)

The Part 1 white paper in this series discusses the single ECM node setup, Spectrum Scale tuning parameters, and includes a comparison of IBM Spectrum Scale with XFS.

The title of the white paper is:

IBM Enterprise Content Management for Linux on z Systems, Scale-Out Case Study (Part 1):

Single ECM Node with XFS and IBM Spectrum Scale 4.2

References

View a list of documents referenced in this white paper.

IBM FileNet P8 Platform V5.2.1 documentation (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1`

IBM Content Navigator V2.0.3 documentation (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/SSEUEX_2.0.3`

IBM WebSphere Application Server Network Deployment V8.5.5 documentation (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5`

IBM Spectrum Scale V4.2.0 documentation (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0`

IBM Spectrum Scale Wiki (IBM developerWorks):

`https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General %20Parallel%20File%20System%20%28GPFS%29`

IBM Spectrum Scale and ECM FileNet Content Manager Are a Winning Combination: Deployment Variations and Value-added Features (IBM Redpaper):

`http://www.redbooks.ibm.com/abstracts/redp5239.html?Open`

IBM DB2 10.5 for Linux, Unix and Windows documentation (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0`

Linux on z Systems - Device Drivers, Features, and Commands, SC33-8411-nn:

`http://www.ibm.com/developerworks/linux/linux390/development_documentation.html`

Details about Linux on z Systems network tuning (IBM developerWorks):

`http://www.ibm.com/developerworks/linux/linux390/perf/tuning_networking.html`

How to Improve Performance with PAV, SC33-8414

Learn how to configure, operate, and troubleshoot Linux on z Systems attached to a SAN environment:

`https://www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_ts.html`

z/VM V6R3 Library (IBM Knowledge Center):

`http://www.ibm.com/support/knowledgecenter/SSB27U_6.3.0`

Notices

Trademarks

Terms and conditions

Printed in USA