NetApp
Go further, faster

Technical Report

IBM DB2 9.5 Performance and Scalability on RHEL5 with NFS and FCP Using NetApp FAS or IBM N series Storage System

Bobby Oommen, Jawahar Lal, NetApp

Xin Zhan, John Tran, Sunil Kamath, IBM DB2 Performance
June 2009 | TR-3775

EXECUTIVE SUMMARY

Before you make a business decision to use FCP or NFS for your DB2 database in a NetApp® FAS or IBM N series environment, you need to consider various factors such as performance, scalability, and cost. This paper is based on a joint study between NetApp and IBM and offers tuning recommendations for running DB2 9.5 databases over NFS and FCP on NetApp FAS or IBM N series system storage in a RHEL5 environment. A series of tests using an Online Transaction Processing (OLTP) workload were run to determine how various tunings affect performance, compare the level of performance NFS and FCP protocol can offer, and demonstrate scalability of DB2 as more CPU and memory resources were equipped.

TABLE OF CONTENTS

1 INTRODUCTION

Information technology (IT) has become the key component of growth strategy for enterprises. Business is relying on IT more than ever to deliver business growth to gain competitive advantage in tough economic environment and global expansion. Exponential data growth and increasing demand for information have put a unique challenge for enterprise IT infrastructure. It must not only meet the requirements of handling current diverse workloads, but also scale seamlessly as the business demand grows, aligning the relative cost, complexity, and capabilities of an advanced IT infrastructure with future application needs and vision for the company down the road.

For decades, IBM DB2 database has led the industry in security, availability, performance, and scalability. It is the information backbone for the world's largest enterprises to address their high-performance and high-demand business requirements. IBM DB2 9.5 has extended this lead by introducing capabilities to fully leverage the latest server and storage hardware innovations, simplify manageability and load balancing, and help customers meet the unprecedented demands placed on their IT infrastructure.

In this paper, we discuss the performance and scalability of DB2 9.5 under various OLTP workload scenarios. We provide insights on performance-tuning recommendations for running DB2 database over NFS and FCP on NetApp FAS or IBM N series system storage in a Red Hat Enterprise Linux® (RHEL) 5.1 environment. We also describe best practices to configure and tune data server, operating system parameters, storage system, and network protocols.

In order to take maximum benefit from this paper, the reader should have knowledge of Red Hat Linux operating system administration, DB2 9.5 database administration, network connectivity, and NetApp FAS or IBM N series system storage administration.

2 HARDWARE AND SOFTWARE SETTINGS

The data server ran RHEL 5.1 with 32GB physical RAM and four Intel® dual-core Xeon® 5150 processors. For data storage, a NetApp FAS3070 storage system with two controllers running Data ONTAP® 7.2.3 was used. NetApp FAS series products are rebranded and sold by IBM as N series storage systems. Underlying hardware and software are same on both of the products. NetApp FAS3070 is rebranded as IBM N5600 system storage. We used two Gigabit Ethernet network adapters for Ethernet connectivity, and two 2Gb QLogic® Fibre Channel adapters for Fibre Channel connectivity between database server and storage system. Figure 1 illustrates the high-level architecture of the test environment.

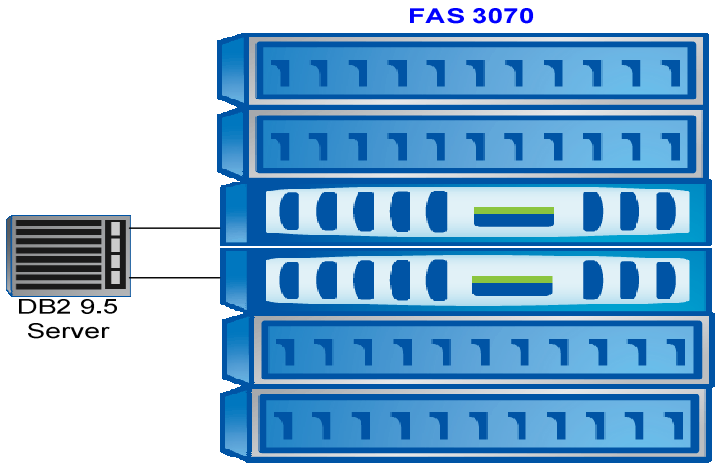

Figure 1) High-level system architecture of test environment.

2.1 SERVER

Server configuration details are shown in Table 1.

Table 1) Server configuration.

Component	Details
Operating system	• Red Hat Enterprise Linux 5.1, kernel 2.6.18-53.el5
• Database server	• DB2 9.5 Enterprise Server Edition, 64 bit
• Total physical RAM	• 32GB
• Processor	• 4 x Intel dual-core Xeon 5150, 2.66GHz, 4MB cache
• Ethernet connection	• 2 x 1Gb Ethernet
• Fibre Channel connection	• 2 x 2Gb dual-channel QLogic 2432 (2462)

2.2 STORAGE

IBM system storage N5600 offers efficient connectivity to data server by supporting various transport protocols such as FCP, iSCSI, and NFS. The Data ONTAP operating system provides a virtualized data environment by creating two storage layers: physical layer, called aggregate, a pool of physical disk spindles protected by the same RAID layout; logical layer, known as flexible volume or FlexVol® volume, created within an aggregate, can grow and shrink as needed and spans all the disk spindles available to the underlying aggregate.

For our test, each storage controller is attached to two shelves for a total 26 disks. Following NetApp best practices, we used RAID-DP®, which allows greater data protection in the event of double disk failures,[1] and created an aggregate with 26 disks on each controller. Disks are grouped into RAID groups of 16: 14 data disks plus two parity disks. IBM N5600 configuration is shown in Table 2.

Table 2) Storage system configurations.

Component	Details
Operating system	Data ONTAP Release 7.2.3
Disks	52 disks, 134GB, 15K RPM
Storage controller	FAS3070 cluster

Storage layout is outlined below as well as recommendations on optimizing the storage system for OLTP workloads.

2.2.1 Storage Layout

In our environment, we created two aggregates, each consisting of 26 disks, one on each node of the clustered storage system. The aggregates were created using the following command:

```
Create aggr0 –r 16 26@134g –t RAID_DP
```

In this example, we created an aggregate named aggr0 over 26 disks; each disk was 134GB in size. Available capacity of this aggregate was 2.44TB in total. Following best practices from NetApp we used RAID-DP or double parity, with RAID group size of 16 disks. Figure 2 illustrates the resulting aggregate on the storage system. An aggregate is a virtual pool of disks and offers segregation between disk spindles and storage

[1] www.redbooks.ibm.com/redpapers/pdfs/redp4169.pdf.

volumes. An aggregate can have one or more logical storage layers within it called flexible volumes. Flexible volumes can offer greater flexibility and better performance than traditional storage volumes. All the volumes in the aggregate use the disk spindles available in the aggregate, which results in better performance.

To store database data we created flexible volumes by executing the following command on the storage system:

vol create [*VolName*] [*AggrName*] [*VolSize*]

Where:

- `VolName` indicates the name of the volume
- `AggrName` indicates the name of the aggregate that contains the volume
- `VolSize` indicates the size of volume in KB, MB, or GB

For example, the command below creates a volume of size 864GB named dbstorage_data1 within an aggregate named aggr0:

```
vol create dbstorage_data1 aggr0 864G
```

For better performance and manageability, it is recommended that transaction logs and database data table space containers reside on separate volumes. For both NFS and FCP tests, we created two volumes to house table space containers, one on each aggregate. An additional volume was created for transaction logs.

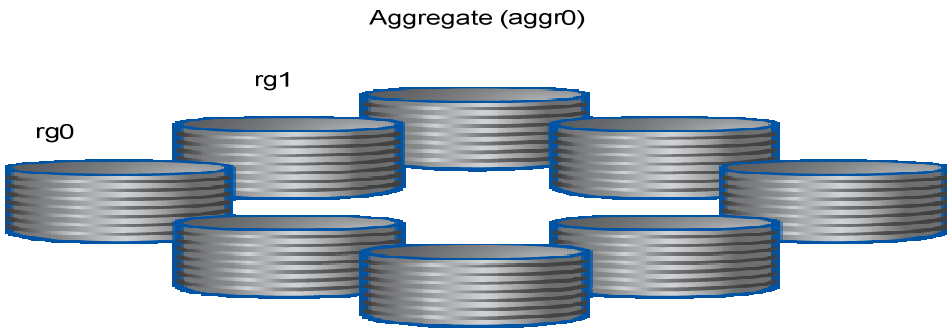

Figure 2) Raid group aggregate layout on a FAS 3070 system.

2.2.2 Enabling Jumbo Frames

In order to achieve better performance, jumbo frames were enabled on both the data server and the storage system. If your environment connection topology involves a switch, you must make sure that the switch supports jumbo frames, and they should be enabled on the switch.

Jumbo frames can carry up to 9,000 bytes per payload. In contrast to a conventional 1,500-byte payload, it significantly reduces the number of Ethernet packet headers required by similar-sized messages, thus reducing CPU overhead and improving data transfer rate. On a storage system , jumbo frames can be enabled by executing the following command:

```
ifconfig e10 mtusize 9000 up
```

To make the above setting persistent, add the following line into /etc/rc file on the storage system:

```
ifconfig e10 mtusize 9000 up
```

2.2.3 Modifying Volume Options

To achieve good performance on the storage system when running database applications, we changed the following options in our environment:

- **Disabled automatic Snapshot™ copies.** A default Snapshot schedule is set for a volume when it is created. Normally a database is backed up based on a user-defined schedule; therefore we disabled automatic Snapshot copies using the following command:

```
vol options [VolName] nosnap on
```

- **Disabled the Snapshot directory display.** Snapshot directory is visible to users by default at the client mountpoints. We disabled the display of the Snapshot directory in our environment using the following command:

```
vol options [VolName] nosnapdir on
```

- **Set nvfail option on.** If this option is on, additional status checking is performed at boot time to verify that nonvolatile random access memory (NVRAM) of the storage system is in a valid state. This option is useful for databases, because if any problems with NVRAM are found, database instances are shut down, and an error message is sent to the console to alert database administrators. The following command was used to set nvfail option for each volume:

```
vol options [VolName] nvfail on
```

- **Set the Snapshot reserve to zero.** When a new volume is created, Data ONTAP reserves 20% of space for Snapshot copies by default, which is unavailable to end users. In order to better utilize the storage space, we opted to set Snapshot reserve to zero by executing the following command:

```
snap reserve –V [VolName] 0
```

- **Updated the access time of all files.** If this option is on, it prevents the update of access time on an inode when a file is read; each file is associated with an inode that stores its basic information. Since DBMS manages the access time on inodes, we enabled this option by executing the following command on each volume used by databases:

```
vol options [VolName] no_atime_update on
```

To change all of the options mentioned above for a volume named dbstorage_data2, one would execute the following set of commands:

```
vol options dbstorage_data2 nosnap on
vol options dbstorage_data2 no_atime_update on
vol options dbstorage_data2 nosnapdir on
vol options dbstorage_data2 nvfail on
snap reserve –V dbstorage_data2 0
```

For other configuration and suggestions on the storage system's physical design, refer to the technical report "DB2 9 for UNIX: Integrating with NetApp Storage Server."[2]

2.3 LINUX KERNEL PARAMETER

We altered the default shared memory limits on Linux to make them sufficient for running a DB2 database system. The following kernel parameters were configured on the data server:

- `kernel.shmmax = 30923763712`
- `kernel.shmall = 7549747`

---

[2] http://media.netapp.com/documents/tr-3531.pdf.

`Kernel.shmmax` sets the maximum size of a shared memory segment in bytes; kernel.shmall determines available memory for shared memory in 4K pages. The settings above allow shared memory to take 28.8GB memory, approximately 90% of the physical memory, and a shared memory segment to have the maximum size of 28.8GB.

3 CONFIGURATION AND TUNING

3.1 LINUX NFS CONFIGURATION AND TUNING

For better NFS performance and reliability, we recommend several configurations and tuning on Linux data server as follows.

First, NFS mount options *rsize* and *wsize* determine the block size NFS uses to read files from or write files to an NFS server. Set options *rsize* and *wsize* to 32768, because the optimal transmission speed was achieved with 32K block size in our test environment. Edit /etc/fstab to incorporate them into the mount options for NFS file systems:

```
bg,rw,hard,nointr,tcp,vers=3,rsize=32768,wsize=32768,timeo=600
```

Second, the NFS module tunable parameter *sunrpc.tcp_slot_table_entries* controls the maximum number of remote procedure call (RPC) commands that can be sent in flight over TCP. Default value is 16. We set it to the maximum value of 128 by changing the system-wide limit in */etc/sysctl.conf*:

```
sunrpc.tcp_slot_table_entries = 128
```

Note that it is ignored by the kernel if simply modified in sysctl.conf. Add the following line as the first line in /etc/init.d/netfs, which guarantees sysctl.conf is executed before mounting NFS file systems:

```
/sbin/sysctl -p
```

Third, enable jumbo frames with maximum transmission unit (MTU) size of 9,000 on Ethernet adapters in both data server and storage system. Experiment results in section 4.1 show the impact of these tunings on NFS performance.

3.2 LINUX FCP CONFIGURATION AND TUNING

Logical unit numbers (LUNs) represent a logical abstraction between the physical disk devices and the applications. A flexible volume is mapped to a LUN in our FCP environment. The queue_depth setting specifies the number of outstanding requests per LUN. Default value is 32. We changed it to 128. In addition, the QLogic Fibre Channel adapter card has a parameter setting for ql2xmaxqdepth, which defines the maximum queue depth reported to SCSI midlevel per device. Default value is 32. We changed it to 256 in the */etc/modprobe.conf* file:

```
options qla2xxx ql2xmaxqdepth=256
```

Effects of such tunings are shown in section 6.1.

3.3 DB2 CONFIGURATION

Large or regular database managed table spaces (DMS) were created with the NO FILE SYSTEM CACHING clause. To enable parallelism, two containers were allocated for each table space, each container placed on an aggregate in the storage system.

The DB2 direct I/O feature is enabled by using the NO FILE SYSTEM CACHING clause in a CREATE TABLESPACE or ALTER TABLESPACE statement. It is available on a table space-by-table space basis, effectively allowing some table spaces in DB2 to use file system caching and others to bypass it. DB2 registry variable DB2_LOGGER_NON_BUFFERED_IO is set to enable direct I/O for transaction logs.

```
db2set DB2_LOGGER_NON_BUFFERED_IO=ON
```

4 DB2 PERFORMANCE AND SCALABILITY OVER NFS AND FCP

Our test environment was designed to stress most components of data server with the goal of utilizing all CPU resources for each test. The database used for test has a physical size of approximately 300GB. During the test, we generated an OLTP workload that simulates the inventory management system of a wholesale supplier, with a population of users executing transactions against the database. These transactions include order entry and delivery, payment entry, order status queries, and stock-level queries. The workload was run in client-server mode; the client was hosted on a separate Linux box. In terms of measuring database throughput, the metric of interest was defined as the number of transactions per second (TPS).

4.1 LINUX NFS AND FCP TUNING EFFECTS

We began our test with four CPUs and 16GB physical memory, running our workload with 150 concurrent users. Figure 3 compares normalized database throughput over FCP and NFS, before and after tuning. It shows approximately 237% improvement of TPS after NFS tuning and 144% improvement of TPS after FCP tuning.

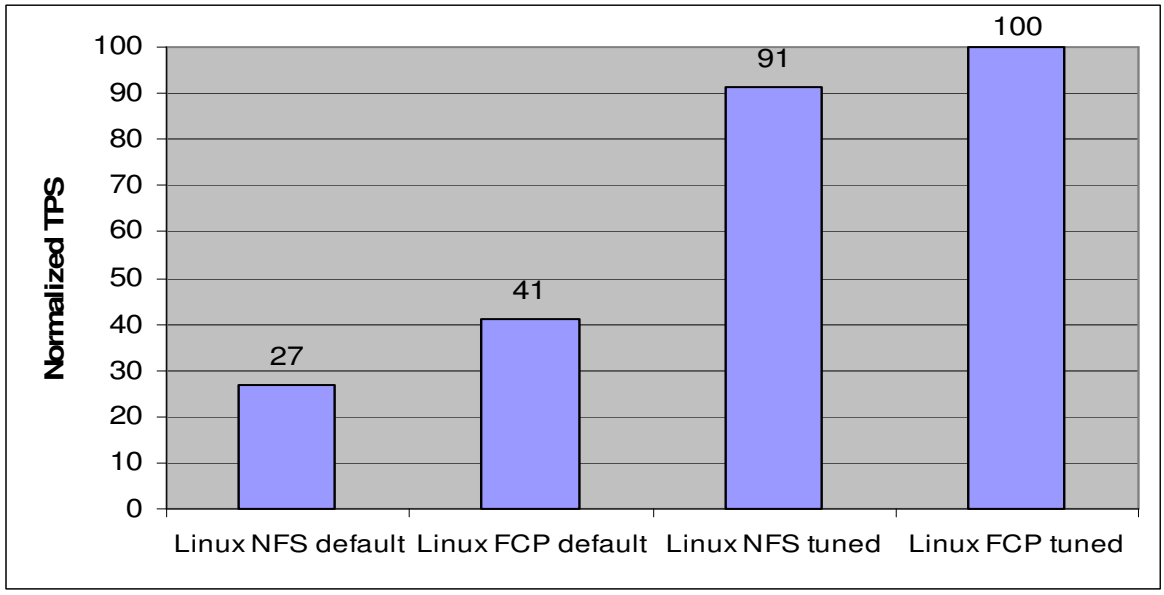

Figure 3) NFS and FCP performance (four way).

To illustrate the impact of various tunings on DB2 performance, Figure 4 shows normalized database throughput when each of the three NFS knobs were tuned independently and all together. Parameter sunrpc.tcp_slot_table_entries played a leading role in performance improvement, followed by mount options *rsize wsize* and jumbo frames. Figure 5 shows normalized database throughput as queue_depth varies from default value 32 to 256 for FCP. Queue_depth of 128 provided the most benefits.

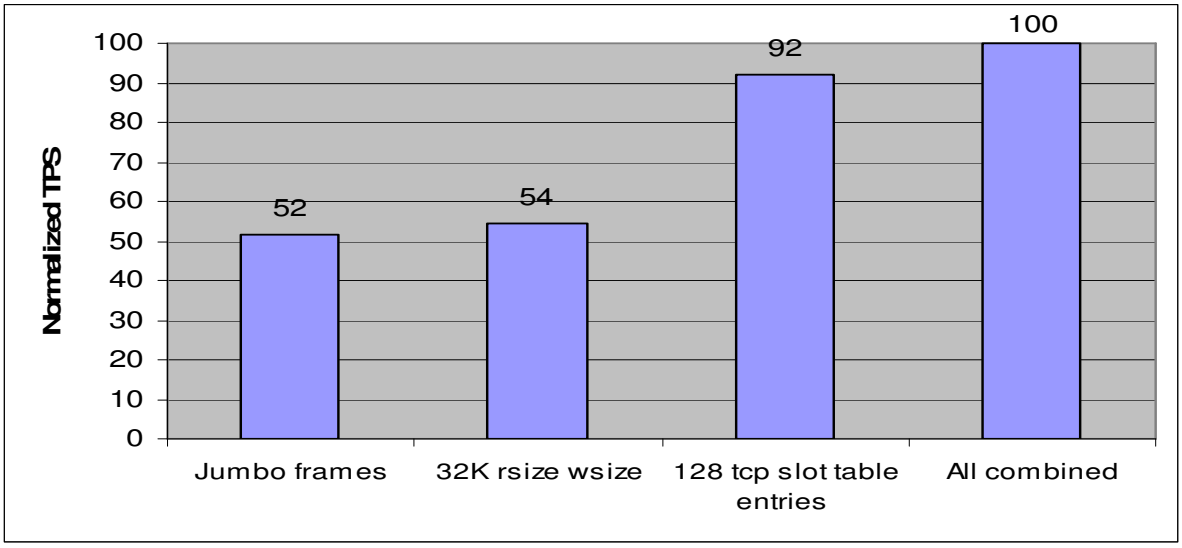

Figure 4) NFS tuning effect (four way).

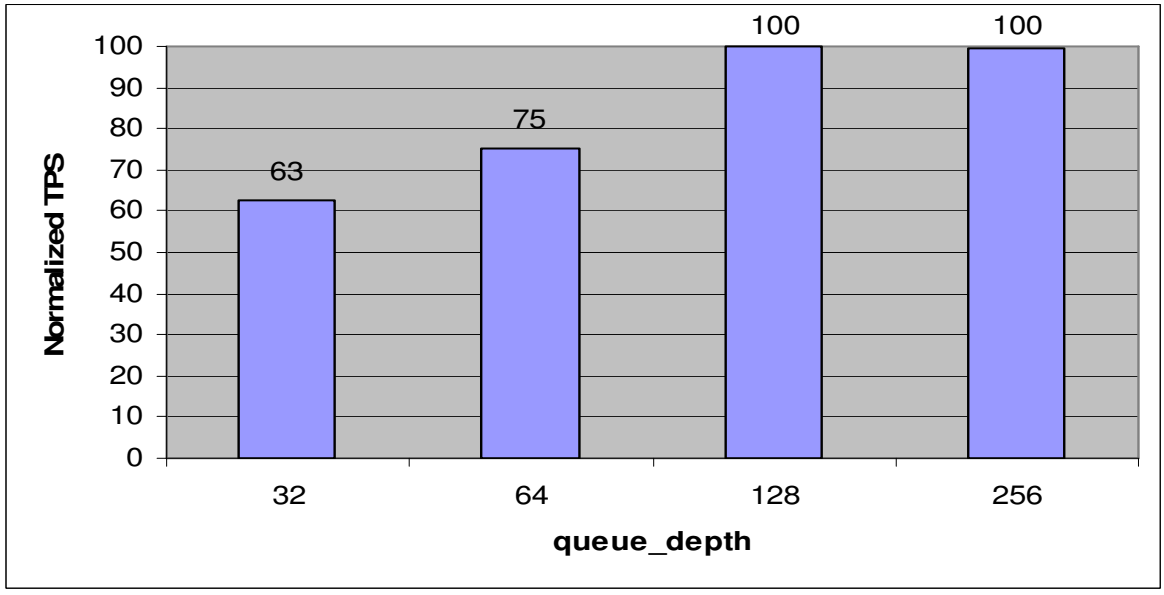

Figure 5) FCP tuning effect (four way).

4.2 LINUX NFS AND FCP PERFORMANCE COMPARISON

As Figure 3 shows, tuned NFS was able to achieve comparable database throughput with FCP, in line with their CPU utilization as shown in Figure 6. These results indicate that NFS after tuning is able to achieve significant performance improvements. Given that it requires much less cost to deploy a database over NFS, it offers an economical alternative for DB2 to connect to a data storage system with minimum performance loss.

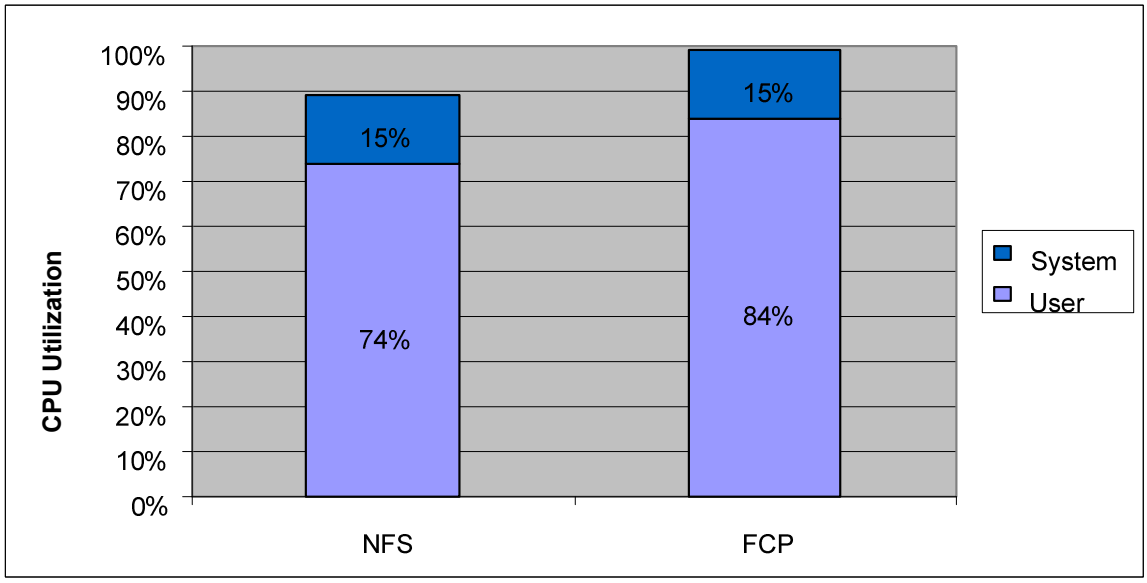

Figure 6) CPU utilization comparison between NFS and FCP (four way).

4.3 SCALABILITY TEST

Additional tests were run in order to determine the impact of scaling from four processors to eight and increasing memory from 16GB to 32GB. We ran the same workload, but with 400 concurrent users against the database.

As Figure 7 shows, the number of I/O operations per second (IOPS) doubled, indicating that I/O performance scales linearly with CPU and memory resources over both NFS and FCP. I/O latency, however, increased by 69% over NFS and 18% over FCP, which is due to saturation of Gigabit Ethernet cards. With approximate network transfer speed of 200MB per second, Gigabit Ethernet network reached saturation.

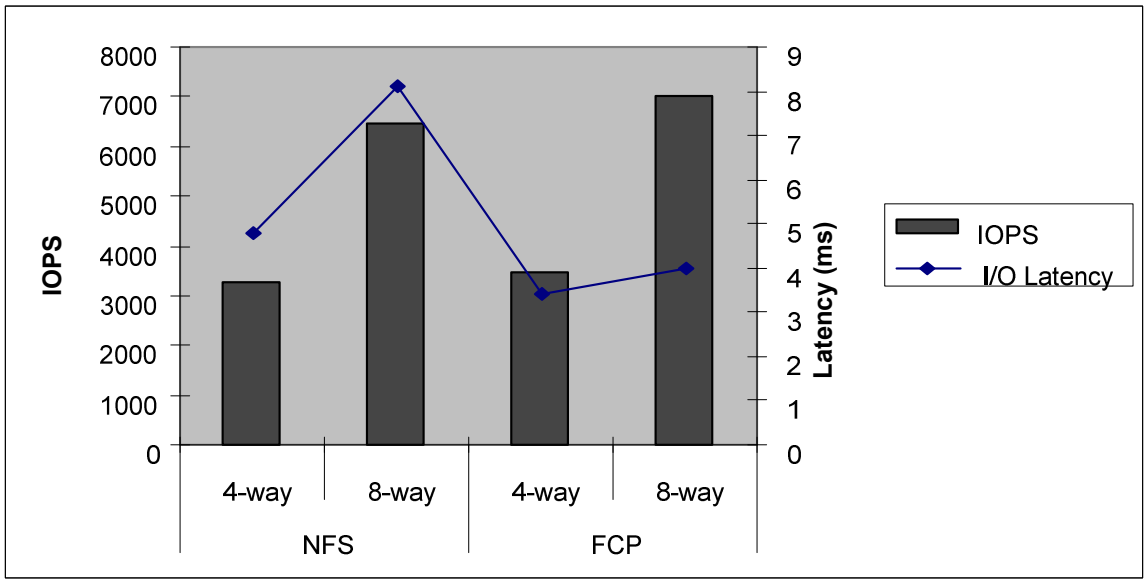

Figure 7) IOPS and I/O latency over NFS and FCP (four way compared to eight way).

5 CONCLUSION

To meet the growing demands for enterprise-wide business, high-performance, low-latency data access to storage is critical for organizations of all types and sizes. DB2, coupled with IBM system storage N5600, is empowering organizations with a flexible, reliable, scalable, cost-effective architecture for their growing business demand.

We provide performance-tuning recommendations for running DB2 database over NFS and FCP on a storage system in an RHEL 5.1 environment. Test results reveal that storage optimization over NFS offers more economical connectivity to data servers with minimum performance loss. Our best practices achieve high performance gains that translate directly into improved business value for customers. They also close the disparity gap between NFS and FCP solutions for DB2, opening the door to a wider range of storage connectivity options for today's forward-thinking companies.

6 ACKNOWLEDGEMENTS

Special thanks for the following people for their contributions and support:

Frank Filz, Krishna Kumar, Steve Pratt IBM Systems and Technology Group.

www.netapp.com