

This article appeared in the
June/July 2003 issue of

Z/JOURNAL



Subscribe instantly at
www.zjournal.com

- Free in the U.S. and Canada
- \$36 per year outside of the U.S. and Canada



zSeries

Speaks

SCSI

**By Neale
Ferguson**

The alluring promise of Linux on the mainframe is the ability to consolidate the tens, hundreds, or even thousands of discrete servers that have populated many machine rooms onto a single managed platform. While it is one thing to move the server function, the network, and the system management, a key piece is missing from this puzzle. What about all of the data on which these servers operated?

In many instances, this data may have been sitting on Small Computer Systems Interface (SCSI) devices in a Storage Area Network (SAN). However, up until now, it has not been possible for a Linux system running on the mainframe to get its hands on this data. Thus, the Total Cost of Ownership (TCO) equation, a prime driving force for Linux on the mainframe, is negatively affected by the need to acquire extra equipment and migrate data.

This situation has not gone unno-

ticed by IBM. In July 2002, they commenced an early support program that provided SCSI connectivity to its zSeries boxes. Indeed, at Linux World in 2002, IBM demonstrated a prototype of this support when it used its z900 mainframe to burn CDs on a SCSI-attached CD-writer. Practical? Hardly. Cool? Definitely!

This article describes the SCSI support provided to the zSeries architecture and provides a quick look at how it works as well as how well it works.

Software AG participates in many of IBM's early support programs, as it needs to keep on the bleeding edge of technology so that its customers can move to the "latest and greatest" whenever it is generally available. Software AG was also one of the first Independent Software Vendors (ISVs) to announce support for Linux on the mainframe. So, IBM approached us to stress test its SCSI support. For this test, however, we would

not be doing anything as esoteric as CD burning; just regular disk operations using our IBM Shark and EMC Symmetrix subsystems and tape operations using a 3590E.

RATIONALE

What is IBM's rationale for providing SCSI support? According to several IBM presentations, it is to enable Linux for zSeries to do SCSI-based I/O on volumes of arbitrary size and thus be able to access distributed storage existing in open SAN configurations. In addition, Linux for zSeries would have access to low-cost storage controllers and new storage devices not supported in classical zSeries configurations (i.e., optical libraries, CD, and DVD). The goal is to facilitate Linux and Unix server consolidation by hosting multiple Linux systems on a single zSeries server while protecting the investment organizations have made in existing SCSI storage.

```

CHPID PATH=(E0), SHARED,
PARTITION=((LPRB, LPRB), (DAEV, LPRB)), TYPE=FCP
CNTLUNIT CUNUMBR=E0FC, PATH=(E0), UNIT=FCP
IODEVICE ADDRESS=(E000, 016), CUNUMBR=(E0FC), UNIT=FCP
CHPID PATH=(E1), SHARED,
PARTITION=((LPRB, LPRB), (LPRB, LPRB)), TYPE=FCP
CNTLUNIT CUNUMBR=E1FC, PATH=(E1), UNIT=FCP
IODEVICE ADDRESS=(E100, 016), CUNUMBR=(E1FC), UNIT=FCP

```

Figure 1 — A New CHPID Type

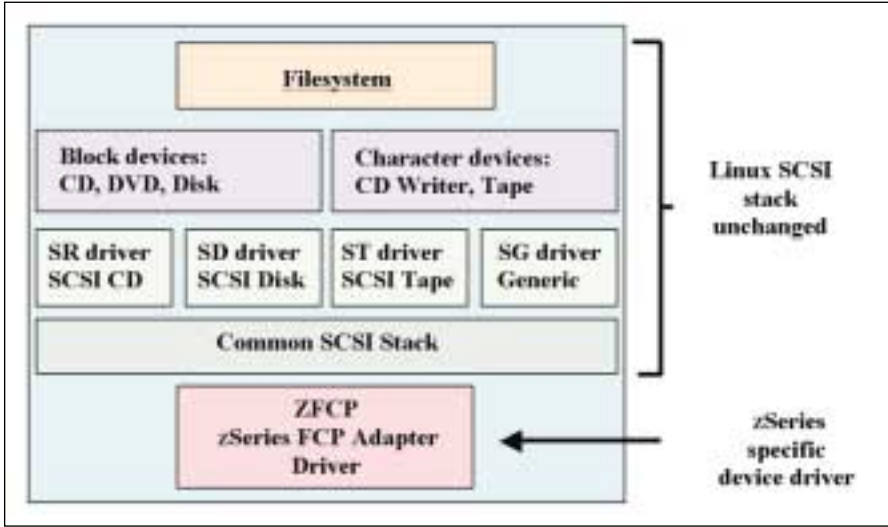


Figure 2 — Linux SCSI Stack Schematic

Component	Model/Version
IBM z900	2064-103
McData SAN Switch	Connectrix ED-64M
Shark	2105-F20
EMC	8730
IBM Magstar	3590-E
Linux for S/390	SLES7++/SLES8
Linux for zSeries	SLES7++/SLES8
z/VM	4.3
ESM Manager	2.0

Figure 3 — Configuration for the SCSI Tests

A NEW CHPID

IBM has used its existing Fibre CONNECTIONS (FICON) technology as the basis of its SCSI regime. The FICON card runs new microcode that implements the Fibre Channel Protocol (FCP). FCP support for zSeries has been implemented by providing a new Channel Path Identifier (CHPID). Figure 1 shows an example of its definition within an IOCDs. This channel uses either the two-port fibre channel cards FICON or FICON Express. Currently, these cards operate at 1 Gbit per second, but 2Gbit per second support has been announced. The loading of different firmware into the card activates the FCP support. The QDIO

protocol is used for communicating between processor, memory, and channel. This is the same protocol used by the OSA express card. This protocol uses continuously running channel programs to reduce I/O path lengths and the number of interrupts.

LINUX SUPPORT

The Linux team at IBM's lab in Boeblingen, Germany, has written a device driver that enables the existing Linux SCSI subsystem to access the FCP hardware. The zfc driver is a low-level or host-bus adapter driver that supplements the Linux SCSI I/O subsystem (SCSI stack). Thus, Linux for zSeries and S/390 can use all SCSI device types currently supported by Linux on other platforms, including SCSI disks, tapes, CD-ROMs, and DVDs. Both 32-bit and 64-bit Linux systems are supported (see Figure 2).

SAN TOPOLOGIES

The SCSI support is based around a SAN. There are three different ways to configure a SAN:

- Point-to-point is the simplest topology to configure. A point-to-point configu-

ration is a direct connection between two end-points. Typically, it consists of a host, a device (such as a disk controller), and a dedicated fibre link.

- Arbitrated loop is a ring topology that shares the fibre channel bandwidth among multiple end-points. The loop is implemented within a hub that interconnects the end-points. An arbitrated scheme is used to determine which end-point gets control of the loop. The maximum number of ports is 127.
- The switched fabric topology provides the most flexibility and makes the best use of the aggregated bandwidth by using switched connections between end-points. One or more switches are interconnected to create a fabric to which the end-points are connected.

The switched fabric topology is the only one supported in the zSeries environment.

OUR CONFIGURATION

The Software AG configuration used for the SCSI tests consisted of the components presented in Figure 3. The configuration is shown in Figure 4. Configuring SCSI is similar to configuring a network. You need to talk to different groups within your organization about the parameters you require (such things as World Wide Port Numbers [WWPNs] or Logical Unit Numbers [LUNs]). In addition, recall when subsystems such as the Shark and Symmetrix were first introduced, systems programmers were told they no longer had to worry about where data was actually being stored: An S/390 device didn't necessarily equate to a physical device on the subsystem. Now, however, if you want to maximize your data transfer rate, you need to understand how to carve up the box so that you can reduce any potential bottlenecks.

One of the key differences between

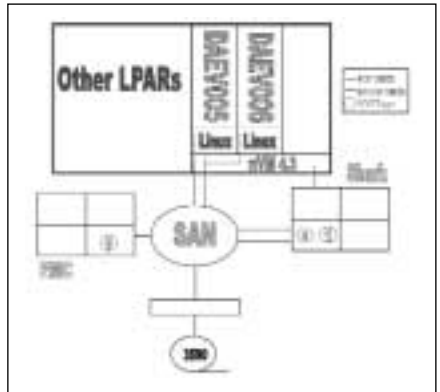


Figure 4 — Hardware Configuration

Component	Traditional	SCSI
CHPID	Path(s) from CPU to device	Path from CPU to SAN
Control Unit	Entity at other end of channel	N/A
Unit Address	Target Device	QDIO queue
WWPN	N/A	Controller port or SCSI bridge
LUN	N/A	Target device

Figure 5 — The Way a Device Number Is Interpreted

```

SCSI disk error : host 0 channel 0 id 1 lun 0 return code = 8000002
Current sd08:00: sns = 70 7
ASC=27 ASCQ= 0
Raw sense data:0x70 0x00 0x07 0x00 0x00 0x00 0x00 0x0a 0x00 0x00
0x00 0x00 0x27 0x00 0x00 0x00 0x00 0x00
I/O error: dev 08:00, sector 0

```

Figure 6 — SCSI Error Accessing EMC

traditional zSeries devices and the new SCSI devices is centered on device addressing. Traditionally, a device number would map to an entry in the Input/Output Configuration Dataset (IOCDs) that described the CHPID(s) that connected to the device, a control unit that managed the device, and a unit address that identified the discrete device. In the zSeries SCSI world, a device number will map the CHPID that connects to the device, and the unit address identifies a distinct QDIO queue. The SCSI subsystem requires additional information such as WWPN and LUN to allow access to a specific device within a SAN. The table in Figure 5 illustrates the way a device number is interpreted in the traditional and SCSI environments.

Configuration of a Linux system is relatively straightforward, especially with SuSE's SLES8 distribution. The configuration activities are similar (and as awkward) as those undertaken on other Linux platforms. (This awkwardness is eliminated within the 2.6 kernel and its device support restructuring.)

As a systems administrator, you need to talk closely with and clearly to those who are responsible for the SAN. This point was brought home to me when we first attempted to use the EMC devices. Our SAN people told me that the LUN I needed to use was 1. I simply assumed that this was 0x01,

and plugged this value into the SCSI mapping parameters. It was accepted and the device was accessed. However, when I attempted to use the device, I received the messages shown in Figure 6.

The error tells me that the device is read-only. After clarifying that with the SAN administrators, I learned that the LUN value was actually: 0x0000100000000000, and the device I was accessing was the configuration database EMC used to keep track of the subsystem.

Note that you can use the normal Linux tools such as fdisk (as opposed to dasdfmt for Extended Count Key Data [ECKD] devices). You can partition your SCSI disks as you would on the Intel platform. You can also access data existing on those SCSI disks without waving any magic wands.

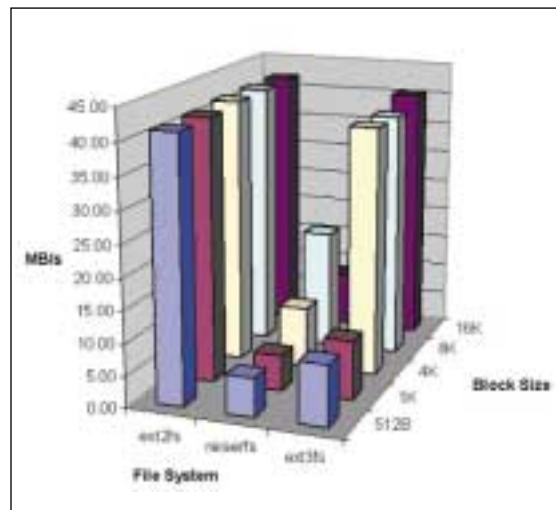


Figure 7 — ESS Write Performance

While it is one thing to move the server function, the network, and the system management, a key piece is missing from this puzzle. What about all of the data on which these servers operated?

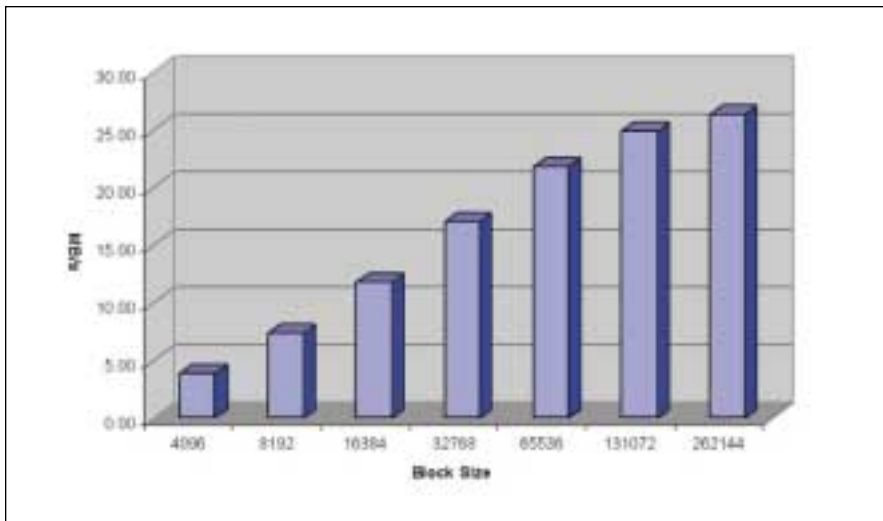


Figure 8 — Streaming Tape Data Rate

```
Attached devices:
Host: scsi0 Channel: 00 Id: 01 Lun: 00
Vendor: IBM      Model: 2105F20      Rev: .107
Type: Direct-Access      ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 01 Lun: 01
Vendor: IBM      Model: 2105F20      Rev: .107
Type: Direct-Access      ANSI SCSI revision: 03
```

Figure 9 — /proc/scsi/scsi

```
0xe000 0x00000001:0x5005076300c38c6d 0x00000000:0x5200000000000000
0xe000 0x00000001:0x5005076300c38c6d 0x00000001:0x5201000000000000
```

Figure 10 — /proc/scsi/map

EXERCISING THE DEVICES

To exercise the disk support, I created a development environment for our Adabas and Adabas SQL Access (AQA) products, and proceeded to build them. The build involved significant amounts of compilation and linking that would allow me to really work the disks. We did not encounter any problems during the build processes.

I created two Adabas databases on the Linux guest DAEV005 using the SCSI disks. One was for the normal installation verification databases and one was for use by AQA. I built and installed AQA, built our internal test suite, and ran the various test scripts. We did not experience any problems. In fact, since we commenced testing in July 2002, we have not had a single outage or loss of data.

In addition, I performed some very simplistic tests to see how fast I could

drive the device writing to a single large file, using different block sizes on different file systems. The test is heavily biased against the Reiser file system, as it is best at handling many small files rather than one large file. These results are shown in Figure 7.

At SHARE in February 2003, Klaus Bergmann of IBM reported that using Logical Volume Manager (LVM) and striping he was able to obtain aggregate data rates of more than 100MB/s on a Reiser file system.

Testing of the tape consisted of exercising the driver, dumping and restoring files, and running more simplistic benchmarks to examine the data transfer rates achievable. The latter's results are shown in Figure 8.

The SCSI subsystem provides a great deal of information within the /proc file system to allow those interested to inspect, monitor, and change the config-

uration and operation of the SCSI subsystem. For example, you can display the devices attached to the system by viewing /proc/scsi/scsi, as shown in Figure 9. Similarly, the parameters used to map the SCSI devices are available in /proc/scsi/map, as shown in Figure 10.

Since the completion of the testing phase we have continued to use the SCSI disks as standard devices in our zSeries environment. We have one application using our Tamino XML database that retrieves a news-feed from Reuters. It is currently chewing its way through 192GB of space as well as backing up to another 100GB partition.

SUMMARY

So, what do I like about the SCSI support? This boils down to two issues:

- The ability to carve out large amounts of disk space rather than using LVM to aggregate the (in comparison) anorexic 3390 family
- Speed, which is of great importance for a database company.

What's missing? The ability to boot from a SCSI disk is an important item. You still need a channel-attached ECKD or FBA device in order to use Linux on the mainframe. IBM understands this deficiency and, hopefully, it won't be too long before it addresses this issue. Also, because the zfcpl device driver uses QDIO to perform its I/O, there is an overhead on z/VM systems due to the maintenance of shadow tables. This overhead is quite minimal but not insignificant. IBM has already reduced some of this overhead by using "thin interrupts" that do not cause the processor to drop out of SIE.

Knowledge of device geometry has become an important skill again. To achieve the throughput results promised, you need to understand what's in that "black box." So, welcome back to the future. The world of SCSI is now open to you, but make sure you relearn the lessons of disk placement that we all forgot so many years ago. **Z**

About the Author

Neale Ferguson is an R&D Fellow with Software AG. With an extensive background in VM, he participated in the first public port of Linux to the mainframe before jumping in with both feet to the IBM-sponsored effort. Originally from Sydney, Australia, he now resides in Northern Virginia. e-Mail: Neale.Ferguson@SoftwareAG-USA.com