Big Servers—2.6 compared to 2.4

Wim A. Coekaerts
Oracle Corporation
`wim.coekaerts@oracle.com`

Abstract

Linux 2.4 has been around in production environments at companies for a few years now, we have been able to gather some good data on how well (or not) things scale up. Number of CPU's, amount of memory, number of processes, IO throughput, etc.

Most of the deployments in production today, are on relatively small systems, 4- to 8-ways, 8–16GB of memory, in a few cases 32GB. The architecture of choice has also been IA32. 64-bit systems are picking up in popularity rapidly, however.

Now with 2.6, a lot of the barriers are supposed to be gone. So, have they really? How much memory can be used now, how is cpu scaling these days, how good is IO throughput with multiple controllers in 2.6.

A lot of people have the assumption that 2.6 resolves all of this. We will go into detail on what we have found out, what we have tested and some of the conclusions on how good the move to 2.6 will really be.

1 Introduction

The comparison between the 2.4 and 2.6 kernel trees are not solely based on performance. A large part of the testsuites are performance benchmarks however, as you will see, they have been used to also measure stability. There are a number of features added which improve stability of the kernel under heavy workloads. The goal of comparing the two kernel releases was more to show how well the 2.6 kernel will be able to hold up in a real world production environment. Many companies which have deployed Linux over the last two years are looking forward to rolling out 2.6 and it is important to show the benefits of doing such a move. It will take a few releases before the required stability is there however it's clear so far that the 2.6 kernel has been remarkably solid, so early on.

Most of the 2.4 based tests have been run on Red Hat Enterprise Linux 3, based on Linux 2.4.21. This is the enterprise release of Red Hat's OS distribution; it contains a large number of patches on top of the Linux 2.4 kernel tree. Some of the tests have been run on the `kernel.org` mainstream 2.4 kernel, to show the benefit of having extra functionality. However it is difficult to even just boot up the mainstream kernel on the test hardware due to lack of support for drivers, or lack of stability to complete the testsuite. The interesting thing to keep in mind is that with the current Linux 2.6 main stream kernel, most of the testsuites ran through completition. A number of test runs on Linux 2.6 have been on Novell/SuSE SLES9 beta release.

2 Test Suites

The test suites used to compare the various kernels are based on an IO simulator for Oracle, called OraSim and a TPC-C like workload generator called OAST.

Oracle Simulator (OraSim) is a stand-alone tool designed to emulate the platform-critical activities of the Oracle database kernel. Oracle designed Oracle Simulator to test and characterize the input and output (I/O) software stack, the storage system, memory management, and cluster management of Oracle single instances and clusters. Oracle Simulator supports both pass-fail testing for validation, and analytical testing for debugging and tuning. It runs multiple processes, with each process representing the parameters of a particular type of system load similar to the Oracle database kernel.

OraSim is a relatively straightforward IO stresstest utility, similar to IOzone or tiobench, however it is built to be very flexible and configurable.

It has its own script language which allows one to build very complex IO patterns. The tool is not released under any open source license today because it has some code linked in which is part of the RDBMS itself. The jobfiles used for the testing are available online `http://oss.oracle.com/external/ols/jobfiles/`.

The advantage of using OraSim over a real database benchmark is mainly the simplicity. It does not require large amounts of memory or large installed software components. There is one executable which is started with the jobfile as a parameter.The jobfiles used can be easily modified to turn on certain filesystem features, such as asynchronous IO.

OraSim jobfiles were created to simulate a relatively small database. 10 files are defined as actual database datafiles and two files are used to simulate database journals.

OAST on the other hand is a complete database stress test kit, based on the TPC-C benchmark workloads. It requires a full installation of the database software and relies on an actual database environment to be created. TPC-C is an on-line transaction workload. The numbers represented during the testruns are not actual TPC-C benchmarks results and cannot or should not be used as a measure of TPC-C performance—they are TPC-C-like; however, not the same.

The database engine which runs the OAST benchmark allocates a large shared memory segment which contains the database caches for SQL and for data blocks (shared pool and buffer cache). Every client connection can run on the same server or the connection can be over TCP. In case of a local connection, for each client, 2 processes are spawned on the system. One process is a dedicated database process and the other is the client code which communicates with the database server process through IPC calls. Test run parameters include run time length in seconds and number of client connections. As you can see in the result pages, both remote and local connections have been tested.

3 Hardware

A number of hardware configurations have been used. We tried to include various CPU architectures as well as local SCSI disk versus network storage (NAS) and fibre channel (SAN).

Configuration 1 consists of an 8-way IA32 Xeon 2 GHz with 32GB RAM attached to an EMC CX300 Clariion array with 30 147GB disks using a QLA2300 fibre channel HBA. The network cards are BCM5701 Broadcom Gigabit Ethernet.

Configuration 2 consists of an 8-way Itanium 2 1.3 GHz with 8GB RAM attached to a JBOD fibre channel array with 8 36GB disks using a QLA2300 fibre channel HBA. The network cards are BCM5701 Broadcom Gigabit Ethernet.

Configuration 3 consists of a 2-way AMD64 2 GHz (Opteron 246) with 6GB RAM attached to local SCSI disk (LSI Logic 53c1030).

4 Operating System

The Linux 2.4 test cases were created using Red Hat Enterprise Linux 3 on all architectures. Linux 2.6 was done with SuSE SLES9 on all architectures; however, in a number of tests the kernel was replaced by the 2.6 mainstream kernel for comparison.

The test suites and benchmarks did not have to be recompiled to run on either RHEL3 or SLES9. Of course different executables were used on the three CPU architectures.

5 Test Results

At the time of writing a lot of changes were still happening on the 2.6 kernel. As such, the actual spreadsheets with benchmark data has been published on a website, the data is up-to-date with the current kernel tree and can be found here: `http://oss.oracle.com/external/ols/results/`

5.1 IO

If you want to build a huge database server, which can handle thousands of users, it is important to be able to attach a large number of disks. A very big shortcoming in Linux 2.4 was the fact that it could only handle 128 or 256.

With some patches SuSE got to around 3700 disks in SLES8, however that meant stealing major numbers from other components. Really large database setups which also require very high IO throughput, usually have disks attached ranging from a few hundred to a few thousand.

With the 64-bit `dev_t` in 2.6, it's now possible to attach plenty of disk. Without modifications it can easily handle tens of thousands of devices attached. This opens the world to really large scale datawarehouses, tens of terabytes of storage.

Another important change is the block IO layer, the BIO code is much more efficient when it comes to large IOs being submitted down from the running application. In 2.4, every IO got broken down into small chunks, sometimes causing bottlenecks on allocating accounting structures. Some of the tests compared 1MB `read()` and `write()` calls in 2.4 and 2.6.

5.2 Asynchronous IO and DirectIO

If there is one feature that has always been on top of the Must Have list for large database vendors, it must be async IO. Asynchronous IO allows processes to submit batches of IO operations and continue on doing different tasks in the meantime. It improves CPU utilization and can keep devices more busy. The Enterprise distributions based on Linux 2.4 all ship with the async IO patch applied on top of the mainline kernel.

Linux 2.6 has async IO out of the box. It is implemented a little different from Linux 2.4 however combined with support for direct IO it is very performant. Direct IO is very useful as it eliminates copying the userspace buffers into kernel space. On systems that are constantly overloaded, there is a nice performance improvement to be gained doing direct IO. Linux 2.4 did not have direct IO and async IO combined. As you can see in the performance graph on AIO+DIO, it provides a significant reduction in CPU utilization.

5.3 Virtual Memory

There has been another major VM overhaul in Linux 2.6, in fact, even after 2.6.0 was released a large portion has been re-written. This was due to large scale testing showing weaknesses as it relates to number of users that could be handled on a system. As you can see in the test results, we were able to go from around 3000 users to over 7000 users. In particular on 32-bit systems, the VM has been pretty much a disaster when it comes to deploying a system with more than 16GB of RAM. With the latest VM changes it is now possible to push a 32GB even up to 48GB system pretty reliably.

Support for large pages has also been a big winner. `HUGETLBFS` reduces TLB misses by a decent percentage. In some of the tests it provides up to a 3% performance gain. In our tests `HUGETLBFS` would be used to allocate the shared memory segment.

5.4 NUMA

Linux 2.6 is the first Linux kernel with real NUMA support. As we see high-end customers looking at deploying large SMP boxes running Linux, this became a real requirement. In fact even with the AMD64 design, NUMA support becomes important for performance even when looking at just a dual-CPU system.

NUMA support has two components; however, one is the fact that the kernel VM allocates memory for processes in a more efficient way. On the other hand, it is possible for applications to use the NUMA API and tell the OS where memory should be allocated and how. Oracle has an extention for Itanium2 to support the libnuma API from Andi Kleen. Making use of this extention showed a significant improvement, up to about 20%. It allows the database engine to be smart about memory allocations resulting in a significant performance gain.

6 Conclusion

It is very clear that many of the features that were requested by the larger corporations providing enterprise applications actually help a huge amount. The advantage of having Asynchronous IO or NUMA support in the mainstream kernel is obvious. It takes a lot of effort for distribution vendors to maintain patches on top of the mainline kernel and when functionality makes sense it helps to have it be included in mainline. Micro-optimizations are still being done and in particular the VM subsystem can improve quite a bit. Most of the stability issues are around 32-bit, where the LowMem versus HighMem split wreaks havoc quite frequently. At least with some of the features now in the 2.6 kernel it is possible to run servers with more than 16GB of memory and scale up.

The biggest surprise was the stability. It was very nice to see a new stable tree be so solid out of the box, this in contrast to earlier stable kernel trees where it took quite a few iterations to get to the same point.

The major benefit of 2.6 is being able to run on really large SMP boxes: 32-way Itanium2 or Power4 systems with large amounts of memory. This was the last stronghold of the traditional Unices and now Linux can play alongside with them even there. Very exciting times.

Proceedings of the Linux Symposium

Volume One

July 21st–24th, 2004
Ottawa, Ontario
Canada