October 2004

DB2 UDB Version 8.2 Enhancements for Linux

Authors:
Rav Ahuja, Dan Behman, John Keenleyside
IBM Software Group

Document Version 2; Last Updated October 20, 2004

CONTENTS

1. OVERVIEW

IBM® DB2® Universal Database™ (DB2 UDB) Version 8.2 (previously codenamed DB2 Stinger) includes exciting new features to improve application programmer productivity and lower total cost of ownership for our customers so that they can gain competitive advantage in managing their information assets. These innovative features deliver improvements in areas such as automating various database management tasks, ease of use, application programmer productivity, reliability, security, scalability, and performance.

IBM was an early supporter of Linux and continues to demonstrate technology and market leadership on this platform. Enhancing DB2 for Linux environments and exploiting the Linux kernel has been an important part of the continuing evolution of DB2 UDB.

Version has numerous enhancements that can deliver benefits in Linux environments. These include:

- *Autonomic features for enhanced manageability and ease of use*
- *Support for new Linux platforms*
- *Transparent optimization and exploitation of Linux environments*
- *Exploitation of 2.6 kernel features*
- *Utilization of asynchronous I/O, direct I/O and vectored I/O for enhancing I/O performance*
- *Support for features such as large pages for memory usage enhancements*
- *Processor affinity to achieve improved scalability and parallelism*
- *Design Advisor for manageability of cluster deployments*
- *Integrated high availability and automated failover capabilities*
- *Support for COBOL programming language*
- *DB2 UDB plug-ins for Eclipse-based integrated development environments*

This paper discusses some of the features of DB2 UDB v8.2 that are unique on Linux platforms. There are many other Version 8.2 features that are common across platforms and are also available on Linux. Information about these non-platform specific features can be found at:

www.**ibm.com**/db2/udb/v82

2 NEW ENHANCEMENTS AND FEATURES

2.1 New Linux Platforms

IBM offers the flexibility and choice to deploy DB2 products on a wide variety of hardware platforms and operating systems. The availability of 64-bit computing platforms presents new possibilities for increased performance of database servers as well as database applications. 32-bit platforms have an inherent address space limitation of 4 gigabytes (GB). Removal of this 4 GB limit on the address space of database servers allows for the creation of larger buffer pools, sort heap, package caches, and other resources that can consume large amounts of memory. A 64-bit environment, and the ability to address more than 4GB of memory, can greatly enhance the scalability and performance of databases.

In this release, IBM has delivered native 64-bit editions of DB2 UDB optimized for all major Linux platforms. The Linux platforms supported by the new 64-bit editions of DB2 UDB are:

- *POWER™ (IBM eServer™ iSeries™, pSeries® and OpenPower systems)*
- *zSeries® (IBM eServer zSeries systems)*
- *Intel® EM64T*

This new 64-bit platform support is in addition to the following Linux platforms for which DB2 UDB Version 8.1 is already available:

- *x86 (32-bit edition for Intel Pentium®, Xeon and AMD Athlon based systems)*
- *AMD64 (64-bit edition for AMD Opteron and Athlon64 based systems)*
- *IA64 (64-bit edition for Intel Itanium based systems)*
- *POWER (32-bit edition for IBM eServer iSeries and pSeries systems)*
- *S/390® (31-bit edition for IBM eServer zSeries systems)*

2.2 Support for 2.6 Kernel

The Linux 2.6 kernel boasts many improvements for running Linux in enterprise environments and datacenters. The DB2 team has been

working with 2.5 and 2.6 kernels for quite some time, enhancing DB2 UDB to exploit new features in these kernels.

2.2.1 DB2 Support on 2.6.x Kernels

Stable versions of the Linux 2.6 kernel have been publicly available since late 2003. IBM teams have been working closely with the major Linux distributors to optimize DB2 UDB for enterprise-class distributions based on 2.6 kernel. SUSE Linux Enterprise Server 9 (SLES 9) from Novell became the first major Linux distribution based on the 2.6 kernel to be validated for DB2 UDB. IBM also intends to support Red Hat Enterprise Linux 4 (RHEL 4) once it becomes generally available. Most Linux distributors have also back-ported some of the features in 2.6 kernel to their 2.4.x kernel based enterprise-class distributions. The most up to date list of Linux distributions supported for DB2 UDB can be found at:

www.**ibm.com**/db2/linux/validate

2.3 Asynchronous I/O

Asynchronous I/O (AIO) allows applications to overlap processing with I/O operations for improved CPU and device utilization. With AIO, processes do not need to wait for I/O requests to complete; they can continue with other processing while the I/O operations are completed. DB2 UDB uses the Linux-AIO interface within its page cleaners to write data to disk more efficiently.

2.3.1 Operating System Requirements

Native AIO support is included in the mainline 2.6 kernel and in newer 2.4 kernel based distributions such as Red Hat Enterprise Linux 3[2] and SUSE Linux Enterprise Server 8. Note that the **libaio.so** library is required, regardless of the distribution or kernel level. At the time of writing, DB2 UDB requires a minimum of libaio 0.3.96 for the x86, x86-64, and PPC platforms and a minimum of libaio 0.3.98-2.1 for the IA64 and zSeries platforms. If the version of libaio installed on your computer

[2] Red Hat Enterprise Linux 3 with a minimum level of **Update 2** is required for running DB2 UDB V8.2 on this Linux distribution

does not meet these minimum requirements, then you may download the latest source RPMs from:

http://download.fedora.redhat.com/pub/fedora/linux/core/development/SRPMS/

Before building and installing the new version of libaio, the old packages (libaio and libaio-devel if applicable) should be removed.

2.3.2 DB2 UDB Configuration

AIO is not enabled by default in DB2 UDB because libaio is not available by default on all main enterprise Linux distributions. To enable the use of AIO in DB2 UDB, run the following command before doing a db2start:

```
db2set DB2LINUXAIO=true
```

Once AIO is enabled, DB2 UDB will attempt to dynamically load libaio. If this shared library cannot be loaded then AIO will be disabled and a message will be logged to db2diag.log. If AIO is still enabled but the kernel returns ENOSYS during an AIO operation, then the Linux kernel does not support native AIO, and a message will be logged to db2diag.log. DB2 UDB will not fall back to POSIX AIO, which was the previously supported version of AIO on Linux, partly because of code complexity, and partly because POSIX AIO tended to decrease performance.

2.3.2 Performance Benefits

The main advantage of AIO is that the DB2 UDB page cleaners can submit a set of pages to be written to disk and then continue scanning for other buffer pool pages to be written out. Since the page cleaners can work more efficiently with AIO, it is possible that a fewer page cleaners will be required. This in turn will free up system resources which can enhance DB2 UDB performance even further.

In preliminary testing with OLTP workloads, performance improvements of up to 14% have been observed when using AIO on file systems, and up to 5% when using AIO on raw devices. When using AIO on file systems it is recommended that you enable direct I/O for table spaces that do not contain LOB or LONG VARCHAR data.

2.4 Direct I/O

Direct I/O (DIO) is an alternative caching policy that reduces CPU utilization for reads and writes by eliminating the copy from file cache to user buffer. A read/write against a file opened with the O_DIRECT flag causes data to be transferred directly between the user buffer and the disk.

When using the file system caching policy, which is the default policy for DB2 UDB, all I/O operations are performed in buffered mode. While this caching policy is extremely effective when the cache hit ratio is high, it has an overhead of making an extra copy of the buffer from the disk to file cache (in the case of read) or from file cache to disk (in the case of write). Since the buffer is already cached in the DB2 buffer pool layer, this dual level of caching proves to be unnecessary in situations where the file system cache hit ratio is low and many I/O operations are performed.

2.4.1 Operating System Requirements

DB2 UDB V8.2 supports direct I/O on Linux with the 2.6 kernel. Refer to section ***2.2.2 DB2 Support on 2.6.x Kernels*** for more information.

2.4.2 DB2 UDB Configuration

Direct I/O is enabled in DB2 UDB by specifying the NO FILE SYSTEM CACHING parameter for the CREATE/ALTER TABLESPACE or CREATE DATABASE commands. This will cause DB2 to open all the containers in the table space with the O_DIRECT flag which instructs the Linux kernel not to use the file system cache for I/O operations. The following example shows how to create a table space with direct I/O enabled.

```
CONNECT TO SAMPLE;

CREATE REGULAR TABLESPACE DIRECTTS
   MANAGED BY DATABASE
   USING ( FILE '/data/dbinst1/container1' 500 )
   NO FILE SYSTEM CACHING;

CONNECT RESET;

TERMINATE;
```

2.4.3 Performance Benefits

Direct I/O avoids the buffering of data in the file system cache in addition to the DB2 UDB buffer pools. This may free up memory that can be used by DB2 UDB for larger buffer pools, sort heaps, or utility heaps. It should also be noted that DIO can be combined with AIO to provide even greater benefits.

File system caching is beneficial for table spaces that contain LOB or LONG VARCHAR data. For system-managed table spaces, DB2 will not use direct I/O for LOBs and LONG VARCHAR data. For database-managed table spaces it is recommended to separate LOB/LONG VARCHAR data from other data by using a large table space.

During a recent internal test, a 12% improvement in throughput was measured on an OLTP workload with direct I/O enabled.

2.5 Vector I/O

Vector (or scatter/gather) I/O allows a vector of I/O buffers to be used instead of one contiguous buffer. For input, the readv() function works like read(), except multiple buffers are filled. For workloads involving table scans, the DB2 prefetchers read multiple extents into the buffer pool cache. Using read(), the prefetchers must read data into a temporary buffer and then copy it to the individual bufferpool pages. DB2 UDB (since Version 8.1.4) can exploit readv() to read data directly into the bufferpool pages.

DB2 UDB does not use writev(). Instead, the page cleaners exploit the AIO interfaces which allow multiple I/O requests to be submitted with one system call.

2.5.1 Operating System Requirements

Vector I/O support is included in the 2.4 and 2.6 kernels. Red Hat Advanced Server 2.1, Red Hat Enterprise Linux 3, and SUSE Linux Enterprise Server 8 all support vector I/O. To obtain the maximum benefit from this feature, you should apply recent service packs or updates made available by the distribution vendors be applied.

2.5.2 DB2 UDB Configuration

Support for readv() can be enabled by issuing the following command before starting DB2 UDB:

```
db2set DB2_SCATTERED_IO=ON
```

2.5.3 Performance Benefits

The main benefit of vector I/O is that DB2 UDB can avoid expensive memory copies. During a recent internal test, query execution times improved by more than 15% with vector I/O enabled.

2.6 Large Pages

Maximizing memory usage is a very important task when tuning DB2 for optimal performance. The Linux 2.6 kernel and 2.4 kernels from some distributions include the option to force the kernel to use large memory pages. DB2 UDB takes full advantage of this option. The kernel handles everything related to memory in terms of pages. Each page can be thought of as a real page in a book. The page can be described in various ways: the text on the page, the number of the page, and the chapter to which the page belongs. Imagine changing the physical size of the pages in a book from a pocket book size to a large atlas size. If the text size remains the same, the obvious difference is that a great deal more information can be held on each page, thus reducing the total number of pages required. The same thing applies to memory pages on Linux. The huge advantage is that the overhead used in identifying and

tracking each and every page is greatly reduced, thus freeing up more memory for more pages to be used by DB2 UDB!

When there are fewer pages in a book, the table of contents may be the same size since it's not the content that is being reduced. However, the number of unique pages that are referenced will be less. In other words, more topics will fit on each page in a larger book. This concept translates directly to how Linux handles the mapping of memory pages to memory in a process' address space. The act of converting these values is called *address translation* and as you can imagine it is something that happens very frequently. To make this task as fast as possible, a buffer of recently translated addresses exists called the *Translation Lookaside Buffer (TLB)*. When an address translation is requested, the TLB is first checked to see if some time can be saved by reusing work that has already been done. If the address isn't found in the buffer, we have a *TLB miss* and the kernel must spend more time completing the actual translation. With large page support there are fewer pages and thus fewer TLB misses and ultimately improved performance.

2.6.1 Operating System Requirements

Large page support is included in the mainline 2.6 kernel but does not exist in the mainline 2.4 kernel. However, certain distributions have back-ported it to their 2.4 based kernels such as Red Hat Advanced Server 2.1, Red Hat Enterprise Linux 3, and SUSE Linux Enterprise Server 8 (Service Pack 3 or later). Note that the **libcap.so** library is required, regardless of distribution/kernel. This library is part of the libcap RPM which is usually included with Linux distributions.

2.6.2 Operating System Configuration

Since large page support is not enabled, or provides insufficient support for DB2 UDB requirements by default, special operating system configuration must be done first. The steps for doing this are specific to the kernel/distribution.

2.6.2.1 SUSE Linux Enterprise Server 8 with Service Pack 3(all platforms except IA64) and Red Hat Advanced Server 2.1

The following two tasks must be performed to enable full large page support.

1) Boot[3] the kernel with the parameter "bigpages=xxxM", where xxx is the desired amount of large pages in Megabytes.

2) As root, run "echo 1 > /proc/sys/kernel/shm-use-bigpages" to dynamically enable large page support in the kernel.

The amount of "bigpages" to use will depend on the size of your buffer pools, and an additional small amount for overhead. It is recommended that you monitor the /proc/meminfo output, paying special attention to the "BigFree" entry to monitor the amount of free large pages remaining.

2.6.2.2 SUSE Linux Enterprise Server 8 with Service Pack 3 (IA64 only)

IA64 adopts the same method that is used in the mainline 2.6.x kernel. Please see section 2.6.2.4 for further instructions.

2.6.2.3 Red Hat Enterprise Linux 3

The amount of large pages (also known as "huge pages" for this distrobution) available to applications is controlled by the /proc/sys/vm/hugetlb_pool kernel parameter. The value it contains is a size in Megabytes and changing it is simply a matter of running a command such as "echo 500 > /proc/sys/vm/hugetlb_pool" as root.

Note that the size must be available in a contiguous block of memory. If it is not, the value is automatically reduced to the largest contiguous block that is available. Because system memory gets fragmented throughout regular system operation, it might be necessary to reboot the computer and set hugetlb_pool as soon as possible.

To monitor the large page usage, examine the /proc/meminfo output and look for the following lines:

[3] You can do this by adding "bigpages=xxxM" parameter to your boot manager configuration file e.g. menu.lst for grub

```
HugePages_Total:   250
HugePages_Free:    250
Hugepagesize:     2048 KB
```

The values will vary depending on the settings on your computer.

2.6.2.4 Distributions based on mainline 2.6.x Kernel

Large page support in the mainline 2.6.x kernel, as obtained direct from kernel.org, is very similar to that of Red Hat Enterprise Linux 3. The only difference is that the kernel parameter is called nr_hugepages instead of hugetlb_pool, and it is sized in hugepage units instead of Megabytes.

2.6.3 DB2 UDB Configuration

Once Linux is configured properly, the next step is to tell DB2 to take advantage of the large pages. This is done by running the following command before doing a db2start:

```
db2set DB2_LGPAGE_BP=yes
```

DB2 UDB will now allocate all memory used for buffer pools and other control segments from the pool of large pages[4].

2.6.4 Performance Benefits

Some of the benefits of using large pages have already been discussed in this section but it is important to stress the main advantage. Because there are fewer pages, less overhead is required which in itself consumes memory. With less overhead, more memory then becomes available to be used in bufferpools, which are extremely important to performance.

Another positive property of large pages is that they will not be swapped out by the operating system and are therefore effectively locked in memory.

In preliminary testing with large database workloads, performance improvements of up to 9% have been observed when using large pages.

---

[4] If you enable large page support in the operating system but neglect to enable it in DB2 UDB, the memory allocations within DB2 UDB will not come from the pool of large pages. This can result in an out-of-memory condition and cause DB2 UDB to operate incorrectly.

2.7 Processor Affinity

Processor affinity refers to binding a process or a set of processes to a specific CPU or a set of CPUs. The advantage of doing this is to override the system's built-in scheduler to force a process to only run on specified CPUs. This can provide some performance gains in SMP and NUMA environments because it's much more likely that the processor's cache will contain cached data for the process bound to that processor. When a process gets scheduled onto different processors, there is little chance for cache hits and can even result in performance degradation. Caution must be exercised with this feature however, as incorrect use could easily result in a negative performance impact. Overriding what the kernel thinks is best for the process can be tricky, therefore caution must be exercised with this feature. Obtaining a significant performance improvement using processor affinity involves some experimentation because every workload is different and exercises the kernel and I/O scheduler differently. It is recommended to focus on defining the affinity for db2loggr (log reader) and db2loggw (log writer) processes first.

2.7.1 Operating System Requirements

CPU affinity is a 2.6 kernel feature that has also been back ported. For DB2 UDB to take advantage of this feature on a 2.4 kernel, one of the following distributions is required:

- *Red Hat Enterprise Linux 3*
- *SUSE Linux Enterprise Server 8*

2.7.2 DB2 UDB Configuration

Which process(es) get bound to which CPU(s) is defined in the resource affinity configuration file. It is a text file consisting of XML-based tags that define the desired resource affinities. You must set the DB2_RESOURCE_POLICY registry variable to the full pathname of your resource affinity configuration file. For example, if /home/db2inst1/resource.conf contains the resource affinity settings, you must issue the following command prior to db2start:

```
db2set DB2_RESOURCE_POLICY=/home/db2inst1/resource.conf
```

The format of the configuration file is a pseudo-XML tag based format[6]. This format provides a high level of future flexibility and customization, but for the purposes of binding processes to a CPU or set of CPUs, the format is straightforward.

2.7.3 Performance Benefits

The benefits of binding processes to a processor are less cache misses and a reduction in inter-node data transfers in a NUMA system. If configured correctly, a performance increase of 1% to 2% could be observed.

2.8 High Availability & Failover

DB2 UDB provides integrated high availability and failover capabilities by including IBM Tivoli® System Automation for Linux (ITSAL). ITSAL offers mainframe-like high availability for Linux applications through policy-based self-healing. ITSAL is included with DB2 UDB V8.2 at no additional cost for usage by DB2 products only in a 2-node cluster configuration. Further information about ITSAL can be found at:

http://www.ibm.com/software/tivoli/products/sys-auto-linux/

DB2 UDB also includes the High Availability Disaster Recovery (HADR) feature. HADR (which is not unique to DB2 UDB on Linux) is useful for recovering from site failures and protects against data loss by replicating changes to a standby server. ITSAL compliments HADR in a Linux environment. It allows for automated failover/takeover and resource management in a disaster recovery situation.

[6] Configuration file samples will be made available in a subsequent version of this document.

3. RESOURCES

This document will likely be updated with new information about DB2 UDB V8.2 for Linux. Please check the site where you downloaded this version for updates in a few weeks.

To learn more about DB2 UDB Version 8.2, visit:

www.**ibm.com**/db2/udb/v82

For further information about DB2 on Linux visit:

www.**ibm.com**/db2/linux

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada