Red Hat Reference Architecture Series

Comparing BenchmarkSQL Performance on Red Hat® Enterprise Linux 5 to Windows Server Enterprise

BenchmarkSQL 2.3.2		BenchmarkSQL 2.3.2
Postgres Plus 8.3.8		SQL Server 2008 R2
Red Hat® Enterprise Linux 5	Compared to:	Windows Server Enterprise
HP ProLiant DL370 G6 (Intel Xeon W5580 - Nehalem)		HP ProLiant DL370 G6 (Intel Xeon W5580 - Nehalem)

Version 1.1
March 2010

redhat

EnterpriseDB®
The Enterprise Postgres Company

Comparing BenchmarkSQL Performance on Red Hat® Enterprise Linux 5 to Windows Server Enterprise

1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

1 Executive Summary

This paper compares the performance of an Online Transaction Processing (OLTP) based workload executed on a PostgreSQL database running on a Red Hat Enterprise Linux 5.4 operating system to that of the same workload executed on a SQL Server database running on Windows Server 2008 R2 Enterprise.

For this effort, Red Hat partnered with EnterpriseDB, a leader in products and services based on PostgreSQL, the world's most advanced open source database. Their Postgres Plus product is ideally suited for transaction-intensive applications requiring superior performance, massive scalability, and compatibility with proprietary database products. Additionally, Postgres Plus provides an economical open source alternative or complement to proprietary databases without sacrificing features or quality.

2 Test Configuration

2.1 Hardware

Database 2 x HP ProLiant DL370 G6	Dual Socket, Quad Core (Total of 8 cores) Intel® Xeon® CPU W5580 @ 3.20GHz
	48 GB RAM
Driver 1 x HP ProLiant DL580 G5	Quad Socket, Quad Core (Total of 16 cores) Intel® Xeon® CPU X7350 @ 2.93GHz
	64 GB RAM

Table 1: Hardware

2.2 Software

	Linux	**Windows**
OS	Red Hat Enterprise Linux 5.4 (2.6.18-164.el5)	Windows Server 2008 R2 Enterprise
Database	Postgres Plus 8.3.8	SQL Server 2008 R2
Workload	BenchmarkSQL 2.3.2	BenchmarkSQL 2.3.2
JDBC Driver	2.0	2.0

Table 2: Software

2.3 SAN

Both the Linux and Windows Server systems utilized two MSA2324fc fibre channel storage arrays for this testing, used to store workload data and logs. Additional details regarding the Storage Area Network (SAN) hardware are in **Table 3**.

(2) HP StorageWorks MSA2324fc Fibre Channel Storage Array	Storage Controller: Code Version: M100R18 Loader Code Version: 19.006 Memory Controller: Code Version: F300R22 Management Controller Code Version: W440R20 Loader Code Version: 12.015 Expander Controller: Code Version: 1036 CPLD Code Version: 8 Hardware Version: 56
(1) HP StorageWorks 4/16 SAN Switch	Firmware: v5.3.0
(1) HP StorageWorks 8/40 SAN Switch	Firmware: v6.1.0a

Table 3: Storage Area Network

3 Test Configuration

3.1 Workload

An Oracle OLTP workload was chosen as it represents a common database implementation exercising server memory and I/O sub-systems.

Characterizing database performance is difficult due to the seemingly endless combinations of tuning attributes available to each database. Performance results can vary a great degree depending on variables such as the application used and database architecture.

For this cross platform database comparison, a vendor and platform neutral driver application was selected. Based on JTPCC and modeled after an industry benchmarking association scenario, BenchmarkSQL is an open source and easy to use JDBC benchmark application closely resembling the TPC-C standard for OLTP. As such, BenchmarkSQL can be pointed at many different databases. As a Java application, it is OS and platform unaware using database neutral drivers for database communication, effectively eliminating outside factors such as proprietary interfaces that could influence performance in one direction or another. The end result is a comparison focused on the core SQL processing and transaction handling abilities of the database.

The Java Database Connectivity (JDBC) driver was used for database access in all testing. The open source BenchmarkSQL project is available at http://sourceforge.net/projects/benchmarksql/. The JTPCC benchmark is also available on SourceForge at http://sourceforge.net/projects/jtpcc.

The BenchmarkSQL OLTP scenario models a wholesale supplier managing orders. The test is designed to impose a transaction load on a database and track the amount of new orders placed and completed under this load. In addition to transaction processing, the suite strings together operations into large transactions. Transactional and referential integrity is ensured throughout the duration of the test by comparing transaction history with actual results. Non-transactional database engines fail this verification.

3.1.1 Test Parameters

The test was driven by BenchmarkSQL with a setting of 32 warehouses and models a set of five transactions driven by a group of simulated operators. The transactions modeled are:

- New-Order
- Payment
- Order Status
- Delivery
- Stock Level

The data set exercised emulates the structural data requirements of a real business. In this example, company X has multiple warehouses, each consisting of ten districts.

Each district has 3000 customers and its own sequential system for numbering order transactions. Additionally, each district as its own operator who creates new orders, books payments, checks the status of existing orders, issues delivery tickets, and checks stock level.

Each warehouse has inventory from a list of 100,000 parts. Therefore a stock-level of as much as 100,000 parts must be maintained per warehouse. For every warehouse added, the amount of information grows rapidly. For instance, in a test executed on 100 warehouses there will be 1000 districts, each with an operator (meaning there will be 1000 terminal connections pushing transactions) and 3000 customers for a total of 3,000,000 customers to track along with the status of any orders generated for each customer.

100 warehouses * 10 districts = 1000 districts

1000 districts * 3000 customers = 3,000,000 customers

As a result, a benchmark simulating 10,000 warehouses requires significantly more underlying hardware than a test simulating 100 warehouses.

The test is designed to measure not just the raw throughput of a database, but the throughput of New-Order transactions while under a heavy load from the other four transactions; Payment, Order Status, Delivery and Stock Level. These transactions not only generate load, but also exercise the ability of the database to effectively and efficiently maintain the integrity of information as it is being accessed and changed from multiple points. The database is responsible for processing concurrent transactions on the same information and giving results that are accurate for the specific point in time in which they are relevant. For example, checking the status of an order tests the multi-version concurrency control of a database (i.e., the value returned from an order status query should reflect the state at the exact time of the request). This is true even if an update is performed that would change the state of that order milliseconds after the query was issued.

A minimum ratio of the other four transaction types is maintained to ensure a healthy load is applied to the database at the same time New- Order transactions are being processed. This ratio is based on a goal of a minimum of one Payment transaction for each New-Order and a minimum of one Order-Status, Delivery and Stock-Level transaction for every ten New-Orders. This order is maintained by the testing application.

Transaction Type	Mix Percentage
New-Order	Up to 45.0
Payment	43.0 Minimum
Order-Status	4.0 Minimum
Delivery	4.0 Minimum
Stock-Level	4.0 Minimum

Table 4: Test Transaction Ratio

Ramp-up (in this case 10 minutes) is the amount of time a test is running only to allow the database to reach a steady transaction rate, where the caches are filled and prioritized. The database is allocated time to adjust to the load produced by the test.

After the ramp-up time the measurement interval was 60 minutes, the period during in which transactions per minute are tracked.

BenchmarkSQL was configured to generate as many transactions as possible at the tested databases. Further, this scenario was configured to skip the wait times of a standard benchmarking association-style test in order to create the heaviest load and update contention possible. The effective rate of New-Order transactions is approximately equivalent to the maximum throughput of 300 warehouses and 3000 terminals.

Parameter	Value
Warehouses	100
Districts[1]	1,000
Customers[2]	3,000,000
Test Ramp-Up	10 Minutes
Test Duration	60 Minutes

[1] Ratio of Warehouses to Districts is 1:10
[2] Ratio of Districts to Customers is 1:3000

Table 5: BenchmarkSQL Test Settings

3.2 Profiling

CPU cycles were closely monitored throughout all testing using *Oprofile*, a system-wide profiler for Linux systems consisting of a kernel driver, a daemon for collecting sample data, and several post-profiling tools for parsing data. It leverages the hardware performance counters of the CPU which can be used for basic time-spent profiling. All code is profiled including hardware and software interrupt handlers, kernel modules, the kernel, shared libraries, and applications. Oprofile was used to collect performance statistics from both the PostgreSQL database server and the BenchmarkSQL test driver. The Windows utility *Perfmon* was used to collect similar statistics (CPU usage, data/log I/O, etc.) on the SQL Server database server.

Reference **Appendix A** for example CPU profiles of the BenchmarkSQL driver system as well as that of the Postgres Plus server.

3.3 Tuning & Optimizations

3.3.1 Operating System

To minimize latency for I/O requests, the deadline kernel I/O scheduling algorithm (elevator=deadline) was used. This scheduler provides near real-time behavior and uses a round robin policy to attempt fairness among multiple I/O requests and to avoid process starvation. Using five I/O queues, this scheduler aggressively re-orders requests to improve I/O performance.

Hyperthreading technology was not engaged during testing.

HugePages were not configured.

Several processes deemed unnecessary for the purpose of this testing were disabled using the `chkconfig` command on both the client and server systems.

auditd
avahi-daemon
bluetooth
cmirror
cpuspeed
cups
gpm
haldaemon
hidd
hplip
ip6tables
iptables
iscsi
iscsid
isdn
kdump
libvirtd
mcstrans
mdmonitor
modclusterd
pcscd
restorecond
rhnsd
ricci
rpcgssd
rpcidmapd
rpcsvcgssd
saslauthd
sendmail
setroubleshoot
smartd
xend
xendomains
xfs
xinetd
yum-updatesd

Security Enhanced Linux (SELinux) was also disabled.

3.3.2 Storage

Each of the HP MSA storage arrays were divided into four 12-disk RAID0 vdisks. On each array, four 20GB LUNs were created for use by PostgreSQL and the same for SQL Server.

To guarantee separate spindles for data and logging (to avoid mixing random and sequential I/O), the database servers each used Logical Volume Management (LVM) to stripe eight LUNs presented from storage into two 80GB volumes, one for data files and the other for logging.

The volume for the data files was formatted with a file system of type ext3 while the log volume was configured to use ext2.

Device-mapper multipathing was used to manage multiple paths to each LUN.

3.3.3 Database

Minimal custom tuning was implemented to the PostgreSQL database for all performance measurements. The parameters modified are listed below.

Parameter	Default	Modified
autovacuum	true	false
checkpoint_timeout	5min	1h
effective_cache_size	3041205	4050045
max_connections	100	400
max_fsm_pages	7522049	10044817
max_fsm_relations	470128	627801
shared_buffers	988820	2640950
work_mem	261288	348211

Table 6: PostgreSQL Tuning

Already optimized for peak performance out of the box, no specific SQL Server tuning was performed.

3.3.4 Driver

The BenchmarkSQL script *runBemchmark.sh* was modified on the driver system to increase java memory from the default 128MB to 4GB (e.g., java -Xms4000m -Xmx4000m -Xmn3600m).

Additionally, the BenchmarkSQL application was modified to properly handle row locking because it did not support the “FOR UPDATE” syntax outside of a cursor. The change allowed BenchmarkSQL to lock rows in SQL Server in the same manner it uses for PostgreSQL and Oracle. This modification was posted to SourceForge.

4 Test Results

Figure 1 graphs the results of increasing the number of terminals, where each terminal represents 3000 users, on a two-socket, quad-core HP ProLiant DL370 G6. The throughput demonstrates scaling only to the point where the network becomes the bottleneck for the client-server remote connections.

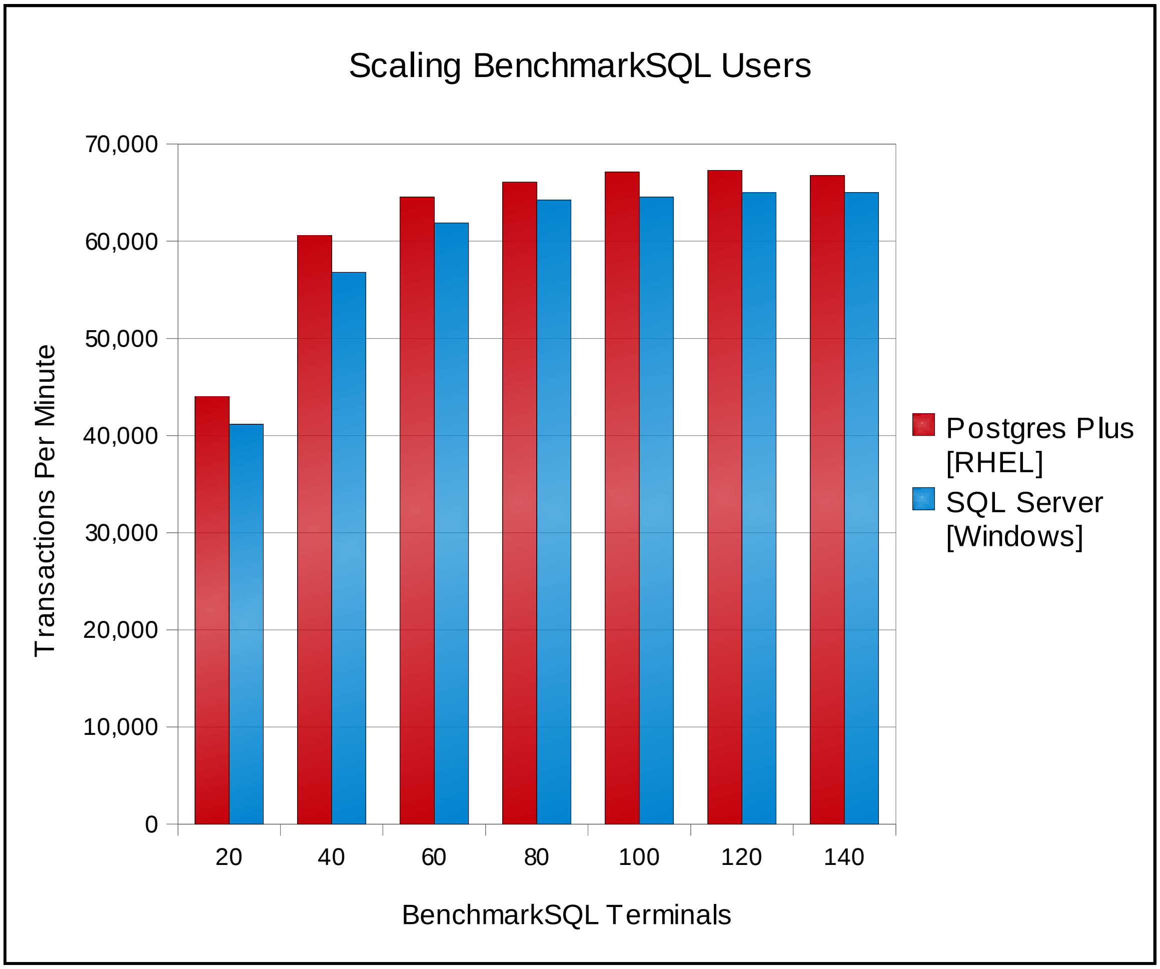

Figure 1: Results of Scaling BenchmarkSQL Users

While it is possible to simply change the network card to eliminate the bottleneck, this may not be an option in many deployment scenarios. This is especially true in public cloud infrastructures. The network connectivity in public cloud deployments can be very unpredictable and the more efficient use of network resources by RHEL and PostgreSQL shown below demonstrates that it is a better choice for these environments.

5 Conclusions

This paper compares the performance and scaling of the BenchmarkSQL workload running on Red Hat Enterprise Linux 5.4 with that of the same workload on Windows Server 2008 R2 Enterprise. The database servers used were HP ProLiant DL370 G6 servers equipped with 48 GB of RAM and comprised of dual sockets, each with a 3.2 GHz Intel Xeon W5580 Nehalem processor (totaling 8 cores).

The data presented in this paper establishes that a common OLTP workload on PostgreSQL can contend with SQL Server and with minimal tuning, is capable of outperforming SQL Server using the same load in an enterprise environment.

The number of actual users and throughput supported in any specific customer situation would naturally depend on the specifics of the application used and the degree of user activity.

Appendix A: Linux Profiles

This appendix contains CPU profiles for the Linux systems captured during a test run of 40 BenchmarkSQL terminals. Note how java itself is the major consumer of CPU cycles on the driver system. The driver load was moved to a remote connection because java's CPU footprint made it difficult to determine exactly how many cycles were being allocated to the database load. Also note the heavy footprint of the e1000 (1GB interconnect used for client/server traffic) as the I/O bottleneck is identified.

Driver CPU Profile:

```
CPU: Core 2, speed 2933.43 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
CPU_CLK_UNHALT...|
  samples|      %|
------------------
 87764690 62.5641 java
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
         79134323 90.1665 anon (tgid:20786 range:0x2aaaab525000-0x2aaaabb75000)
          6641092  7.5669 libjvm.so
          1185918  1.3512 libpthread-2.5.so
           345514  0.3937 libc-2.5.so
           234838  0.2676 libnet.so
            96470  0.1099 libjava.so
            74092  0.0844 libmawt.so
            30290  0.0345 libX11.so.6.2.0
            16018  0.0183 libfontmanager.so
             6129  0.0070 libawt.so
                6 6.8e-06 librt-2.5.so
 42302592 30.1559 vmlinux
  4101457  2.9238 ip_conntrack
  2912736  2.0764 e1000e
   709744  0.5059 oprofiled
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
           702806 99.0225 oprofiled
             6938  0.9775 libc-2.5.so
   503225  0.3587 python
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
```

```
          402738 80.0314 libpython2.4.so.1.0
           33826  6.7218 libc-2.5.so
           26601  5.2861 libpthread-2.5.so
           21837  4.3394 libm-2.5.so
           12180  2.4204 timemodule.so
            2227  0.4425 libglib-2.0.so.0.1200.3
            1302  0.2587 ld-2.5.so
CPU: Core 2, speed 2933.43 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
samples  %        image name               app name                 symbol name
79134323 56.4118  anon (tgid:20786 range:0x2aaaab525000-0x2aaaabb75000) java                   anon
(tgid:20786 range:0x2aaaab525000-0x2aaaabb75000)
6641092   4.7342  libjvm.so                java                     /usr/lib/jvm/java-1.6.0-openjdk-
1.6.0.0.x86_64/jre/lib/amd64/server/libjvm.so
4101457   2.9238  ip_conntrack             ip_conntrack             /ip_conntrack
3101706   2.2111  vmlinux                  vmlinux                  schedule
2912736   2.0764  e1000e                   e1000e                   /e1000e
1974773   1.4077  vmlinux                  vmlinux                  lock_timer_base
1655304   1.1800  vmlinux                  vmlinux                  tcp_v4_rcv
1635908   1.1662  vmlinux                  vmlinux                  .text.show_schedstat
1597135   1.1385  vmlinux                  vmlinux                  __mod_timer
1051217   0.7494  vmlinux                  vmlinux                  nr_context_switches
1004201   0.7159  vmlinux                  vmlinux                  dev_queue_xmit
905060    0.6452  vmlinux                  vmlinux                  copy_user_generic_unrolled
861823    0.6144  vmlinux                  vmlinux                  pskb_copy
834649    0.5950  vmlinux                  vmlinux                  skb_append_datato_frags
819039    0.5839  vmlinux                  vmlinux                  .text.cpu_to_phys_group
702806    0.5010  oprofiled                oprofiled                /usr/bin/oprofiled
697813    0.4974  vmlinux                  vmlinux                  kfree
668870    0.4768  vmlinux                  vmlinux                  mwait_idle
631582    0.4502  vmlinux                  vmlinux                  tcp_ack
625379    0.4458  vmlinux                  vmlinux                  __write_lock_failed
602601    0.4296  vmlinux                  vmlinux                  thread_return
592207    0.4222  vmlinux                  vmlinux                  system_call
586281    0.4179  vmlinux                  vmlinux                  __alloc_skb
571547    0.4074  vmlinux                  vmlinux                  ip_output
566669    0.4040  vmlinux                  vmlinux                  kmem_cache_free
525573    0.3747  vmlinux                  vmlinux                  __wake_up
515956    0.3678  vmlinux                  vmlinux                  tcp_sendmsg
507699    0.3619  vmlinux                  vmlinux                  ip_route_input
501778    0.3577  vmlinux                  vmlinux                  tcp_rcv_established
490709    0.3498  vmlinux                  vmlinux                  tcp_sendpage
473004    0.3372  libpthread-2.5.so        java                     pthread_mutex_lock
467205    0.3331  vmlinux                  vmlinux                  init_idle
441834    0.3150  vmlinux                  vmlinux                  try_to_wake_up
431200    0.3074  vmlinux                  vmlinux                  tcp_recvmsg
```

```
428224   0.3053  vmlinux              vmlinux              .text.cpu_attach_domain
408606   0.2913  vmlinux              vmlinux              lock_sock
406349   0.2897  vmlinux              vmlinux              ip_queue_xmit
CPU: Core 2, speed 2933.43 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
samples  %        symbol name
3101706   7.3322  schedule
1974773   4.6682  lock_timer_base
1655304   3.9130  tcp_v4_rcv
1635908   3.8672  .text.show_schedstat
1597135   3.7755  __mod_timer
1051217   2.4850  nr_context_switches
1004201   2.3739  dev_queue_xmit
905060   2.1395  copy_user_generic_unrolled
861823   2.0373  pskb_copy
834649   1.9730  skb_append_datato_frags
819039   1.9361  .text.cpu_to_phys_group
697813   1.6496  kfree
668870   1.5812  mwait_idle
631582   1.4930  tcp_ack
625379   1.4783  __write_lock_failed
602601   1.4245  thread_return
592207   1.3999  system_call
586281   1.3859  __alloc_skb
571547   1.3511  ip_output
566669   1.3396  kmem_cache_free
525573   1.2424  __wake_up
515956   1.2197  tcp_sendmsg
507699   1.2002  ip_route_input
501778   1.1862  tcp_rcv_established
490709   1.1600  tcp_sendpage
467205   1.1044  init_idle
441834   1.0445  try_to_wake_up
431200   1.0193  tcp_recvmsg
428224   1.0123  .text.cpu_attach_domain
408606   0.9659  lock_sock
406349   0.9606  ip_queue_xmit
387542   0.9161  tcp_transmit_skb
376864   0.8909  sock_wfree
374083   0.8843  avc_audit
372781   0.8812  pfifo_fast_enqueue
371685   0.8786  .text.domain_distance
357001   0.8439  __read_lock_failed
CPU: Core 2, speed 2933.43 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
```

```
samples  %       image name               app name                 symbol name
79134323 56.4118  anon (tgid:20786 range:0x2aaaab525000-0x2aaaabb75000) java               anon
(tgid:20786 range:0x2aaaab525000-0x2aaaabb75000)
6641092   4.7342  libjvm.so               java                /usr/lib/jvm/java-1.6.0-openjdk-
1.6.0.0.x86_64/jre/lib/amd64/server/libjvm.so
3101706   2.2111  vmlinux                 vmlinux                  schedule
1974773   1.4077  vmlinux                 vmlinux                  lock_timer_base
1655304   1.1800  vmlinux                 vmlinux                  tcp_v4_rcv
1635908   1.1662  vmlinux                 vmlinux                  .text.show_schedstat
1610639   1.1482  ip_conntrack.ko         ip_conntrack.ko          tcp_packet
1597135   1.1385  vmlinux                 vmlinux                  __mod_timer
1051217   0.7494  vmlinux                 vmlinux                  nr_context_switches
1004201   0.7159  vmlinux                 vmlinux                  dev_queue_xmit
917511    0.6541  e1000e.ko               e1000e.ko                e1000_xmit_frame
905060    0.6452  vmlinux                 vmlinux                  copy_user_generic_unrolled
901553    0.6427  ip_conntrack.ko         ip_conntrack.ko          ip_conntrack_find_get
861823    0.6144  vmlinux                 vmlinux                  pskb_copy
834649    0.5950  vmlinux                 vmlinux                  skb_append_datato_frags
819039    0.5839  vmlinux                 vmlinux                  .text.cpu_to_phys_group
702806    0.5010  oprofiled               oprofiled                /usr/bin/oprofiled
697813    0.4974  vmlinux                 vmlinux                  kfree
668870    0.4768  vmlinux                 vmlinux                  mwait_idle
657744    0.4689  e1000e.ko               e1000e.ko                e1000_clean_tx_irq
631582    0.4502  vmlinux                 vmlinux                  tcp_ack
625379    0.4458  vmlinux                 vmlinux                  __write_lock_failed
602601    0.4296  vmlinux                 vmlinux                  thread_return
601138    0.4285  ip_conntrack.ko         ip_conntrack.ko          __ip_ct_refresh_acct
592207    0.4222  vmlinux                 vmlinux                  system_call
586281    0.4179  vmlinux                 vmlinux                  __alloc_skb
583925    0.4163  e1000e.ko               e1000e.ko                e1000_clean_rx_irq
571547    0.4074  vmlinux                 vmlinux                  ip_output
566669    0.4040  vmlinux                 vmlinux                  kmem_cache_free
525573    0.3747  vmlinux                 vmlinux                  __wake_up
515956    0.3678  vmlinux                 vmlinux                  tcp_sendmsg
507699    0.3619  vmlinux                 vmlinux                  ip_route_input
501778    0.3577  vmlinux                 vmlinux                  tcp_rcv_established
490709    0.3498  vmlinux                 vmlinux                  tcp_sendpage
478056    0.3408  ip_conntrack.ko         ip_conntrack.ko          ip_conntrack_in
473004    0.3372  libpthread-2.5.so       java                pthread_mutex_lock
467205    0.3331  vmlinux                 vmlinux                  init_idle
```

On the PostgreSQL server, note that postgres itself and components such as hash searching and lock acquisition now occupy the top CPU cycles as expected.

PostgreSQL Server CPU Profile:

```
CPU: Core 2, speed 3199.19 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
CPU_CLK_UNHALT...|
  samples|      %|
------------------
 10274094 82.5856 postgres
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
          8703603 84.7141 postgres
          1564679 15.2294 libc-2.5.so
             5812  0.0566 libm-2.5.so
  1825849 14.6766 vmlinux
   174818  1.4052 e1000e
    39240  0.3154 oprofiled
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
            39060 99.5413 oprofiled
              180  0.4587 libc-2.5.so
    31963  0.2569 perl
        CPU_CLK_UNHALT...|
          samples|      %|
        ------------------
            29269 91.5715 libperl.so
             1356  4.2424 libc-2.5.so
              606  1.8959 libz.so.1.2.3
              461  1.4423 Zlib.so
              264  0.8260 libpthread-2.5.so
                3  0.0094 ld-2.5.so
                3  0.0094 libm-2.5.so
                1  0.0031 HiRes.so
    22642  0.1820 oprofile
    15574  0.1252 bridge
    14687  0.1181 qla2xxx
    12916  0.1038 dm_mod
     5939  0.0477 scsi_mod
     5165  0.0415 jbd
     4936  0.0397 db2fm
        CPU_CLK_UNHALT...|
CPU: Core 2, speed 3199.19 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
samples  %        image name               app name                 symbol name
414702    3.3335  postgres                 postgres                 hash_search_with_hash_value
407760    3.2777  postgres                 postgres                 LWLockAcquire
```

```
382291   3.0729  postgres              postgres              index_getnext
295996   2.3793  postgres              postgres              _bt_compare
293843   2.3620  postgres              postgres              AllocSetAlloc
230794   1.8552  libc-2.5.so           postgres              memcpy
229094   1.8415  postgres              postgres              GetSnapshotData
223350   1.7953  postgres              postgres              PinBuffer
199799   1.6060  postgres              postgres              _bt_checkkeys
181103   1.4557  postgres              postgres              XLogInsert
178660   1.4361  libc-2.5.so           postgres              vfprintf
177767   1.4289  postgres              postgres              ExecInitExpr
174818   1.4052  e1000e                e1000e                /e1000e
168057   1.3509  postgres              postgres              LWLockRelease
135002   1.0852  postgres              postgres              SearchCatCache
134273   1.0793  libc-2.5.so           postgres              strncpy
128127   1.0299  vmlinux               vmlinux                schedule
122210   0.9824  postgres              postgres              hash_any
121631   0.9777  postgres              postgres              slot_deform_tuple
115940   0.9320  postgres              postgres              PostgresMain
108476   0.8720  postgres              postgres              fmgr_info_cxt_security
107241   0.8620  postgres              postgres              MemoryContextAllocZeroAligned
103539   0.8323  libc-2.5.so           postgres              strlen
97895    0.7869  postgres              postgres              FunctionCall2
94390    0.7587  postgres              postgres              AllocSetFree
83720    0.6730  postgres              postgres              heap_page_prune_opt
77368    0.6219  libc-2.5.so           postgres              _int_malloc
76237    0.6128  libc-2.5.so           postgres              _itoa_word
72867    0.5857  postgres              postgres              internal_putbytes
71588    0.5754  postgres              postgres              pg_mblen
71536    0.5750  postgres              postgres              ExecProject
70928    0.5701  postgres              postgres              pfree
69779    0.5609  postgres              postgres              HeapTupleSatisfiesVacuum
63584    0.5111  postgres              postgres              MemoryContextAlloc
63090    0.5071  postgres              postgres              pg_mbcliplen
62862    0.5053  postgres              postgres              ReadBuffer_common
62052    0.4988  postgres              postgres              appendBinaryStringInfo
CPU: Core 2, speed 3199.19 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00
(Unhalted core cycles) count 100000
samples  %       image name            app name               symbol name
414702   3.3335  postgres              postgres              hash_search_with_hash_value
407760   3.2777  postgres              postgres              LWLockAcquire
382291   3.0729  postgres              postgres              index_getnext
295996   2.3793  postgres              postgres              _bt_compare
293843   2.3620  postgres              postgres              AllocSetAlloc
230794   1.8552  libc-2.5.so           postgres              memcpy
229094   1.8415  postgres              postgres              GetSnapshotData
223350   1.7953  postgres              postgres              PinBuffer
```

```
199799   1.6060  postgres          postgres          _bt_checkkeys
181103   1.4557  postgres          postgres          XLogInsert
178660   1.4361  libc-2.5.so        postgres          vfprintf
177767   1.4289  postgres          postgres          ExecInitExpr
168057   1.3509  postgres          postgres          LWLockRelease
135002   1.0852  postgres          postgres          SearchCatCache
134273   1.0793  libc-2.5.so        postgres          strncpy
128127   1.0299  vmlinux            vmlinux            schedule
122210   0.9824  postgres          postgres          hash_any
121631   0.9777  postgres          postgres          slot_deform_tuple
115940   0.9320  postgres          postgres          PostgresMain
108476   0.8720  postgres          postgres          fmgr_info_cxt_security
107241   0.8620  postgres          postgres          MemoryContextAllocZeroAligned
103539   0.8323  libc-2.5.so        postgres          strlen
97895    0.7869  postgres          postgres          FunctionCall2
94390    0.7587  postgres          postgres          AllocSetFree
83720    0.6730  postgres          postgres          heap_page_prune_opt
77368    0.6219  libc-2.5.so        postgres          _int_malloc
76237    0.6128  libc-2.5.so        postgres          _itoa_word
72867    0.5857  postgres          postgres          internal_putbytes
71588    0.5754  postgres          postgres          pg_mblen
71536    0.5750  postgres          postgres          ExecProject
70928    0.5701  postgres          postgres          pfree
69779    0.5609  postgres          postgres          HeapTupleSatisfiesVacuum
63584    0.5111  postgres          postgres          MemoryContextAlloc
63090    0.5071  postgres          postgres          pg_mbcliplen
62862    0.5053  postgres          postgres          ReadBuffer_common
62052    0.4988  postgres          postgres          appendBinaryStringInfo
61117    0.4913  postgres          postgres          hash_seq_search
```