”Verification of performance benefits of OpenPOWER for EnterpriseDB”

Kenneth Albinsson

Conoa AB

November 2015

Table of Contents

Executive Summary

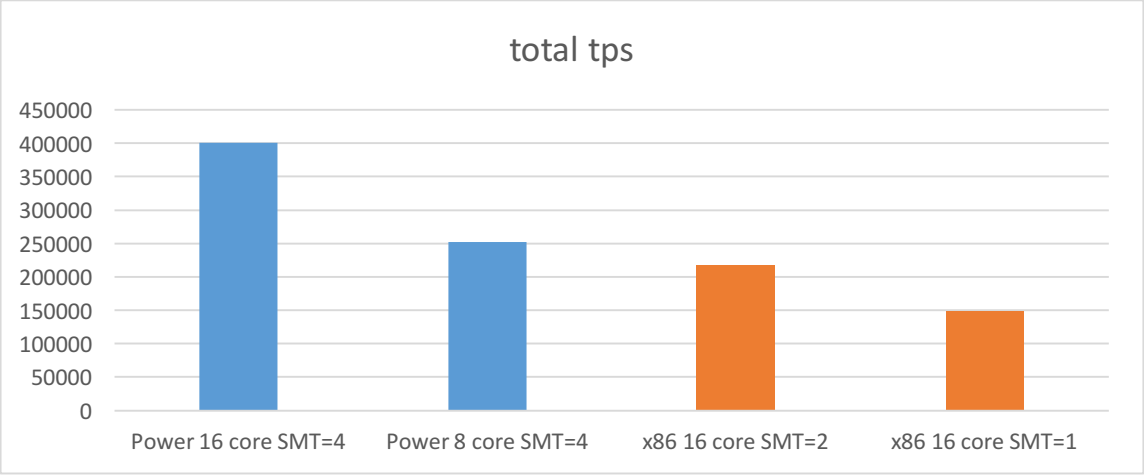
Conoa is a consulting company specializing in Enterprise Open Source. We help some of the largest companies in Sweden with their open source environments. We have partnerships with leading companies within the Open Source Ecosystem including IBM (OpenPOWER), EnterpriseDB & Red Hat.

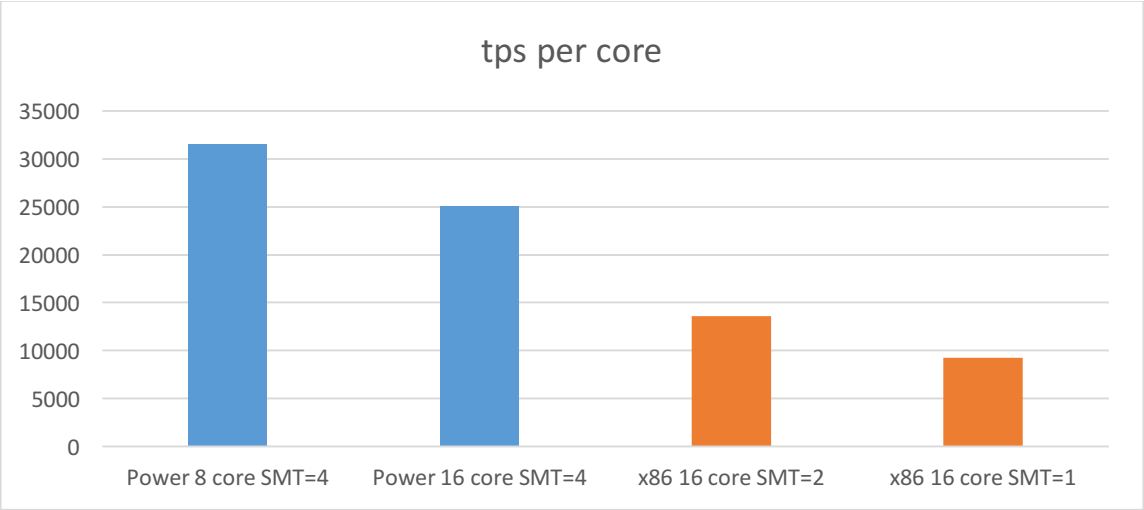
Many of our customers are migrating or planning to migrate their proprietary databases to open source databases. Since OpenPOWER are gaining momentum we wanted to verify the performance benefits of running EntepriseDB on Linux on Power compared to x86 in a virtualized environment such as Red Hat Enterprise Virtualization.

Our own tests in real customer virtualization environments shows that running EnterpriseDB on Linux on Power compared to x86 gives 2X performance benefits.

This means that you can run the same workload with 50% of the CPU resources on Power compared to x86 or that you can run twice the workload on Power compared to x86.

With the recent announcement from IBM regarding their IBM Power Systems LC Server family now is the time to invest in OpenPOWER.





Introduction

This whitepaper describes the hands on testing and verification of the results indicated by IBM in the whitepaper "*IBM Power Systems solution for EnterpriseDB Postgres Plus Advanced Server"* available from IBM that describes the performance benefits of running EnterpriseDB on Linux on Power compared to the x86 platform.

The original whitepaper indicates that you get 2X the number of transactions per second, tps, per core when running EnterpriseDB on Linux on Power compared to x86.

Conoa wanted to see if these performance benefits could be verified outside of an IBM lab environment. We also wanted to see if these benefits were the same when running the OS on a virtual machine instead of running the OS bare metal on the hardware as described in the original whitepaper. We decided to run Red Hat Enterprise Virtualization as our virtualization platform as it is available on both x86 and the OpenPOWER platform.

Preparation

The servers we used for the test were the following:

```
HP ProLiant DL380p
Intel(R) Xeon(R) CPU E5-2690 0 @ 2.9GHz
2 physical CPU, 8-core


IBM Power System S822L
IBM POWER8 @ 4.1 GHz
2 physical CPU, 8-core
```

The latest version of EnterpriseDB available for both Linux on x86 and Power was at the time version 9.3 and is supported on Red Hat 6. The latest version of Red Hat 6 was at the time version 6.7.

EnterpriseDB expects to GA version 9.4 on RHEL 7 for Power in December 2015. The 9.4 version will support Little Endian which will run Intel code natively, and likely provide a performance advantage. Conoa will repeat these tests when 9.4 is GA.

The following versions where used on the Power server:

```
RHEV Manager version 3.5.4.2.
RHEV Hypervisor 3.4.3
Red Hat Enterprise Linux 6.7
```

The following versions where used on the x86 server:

```
RHEV Manager version 3.5.3.1
RHEV Hypervisor - 6.6
Red Hat Enterprise Linux 6.7
```

As noted in the summary of the IBM whitepaper:

***"Note:** The POWER8 processor-based system runs in SMT4 mode (four hardware threads per physical core), whereas, the Intel processor-based system supports only two hardware threads per core on the Xeon E5 processor.*

Additional tuning and optimizations are feasible. RHEL7 supports optimizations and tuning specifically for the new POWER8 processor including SMT8. RHEL7 testing is planned after supported by EnterpriseDB."

In order to achieve optimal performance on the Power system we need to be able to activate multiple threads per core (SMT). After checking the RHEV Manager on both environments we found that there was no way of activating SMT on the virtual machines. After contacting Red Hat support they confirmed that this was true.

The only way to activate multiple threads per core (SMT) was on the hypervisor host itself. The problem is that then there is no way of deciding if there are real physical cores or parts of physical cores (virtual cores) assigned to the virtual machine.

After discussing the matter with Red Hat support they provided Conoa with a vdsm hook that can be installed on the hypervisor host, which allows multiple threads per core (SMT) on the virtual machines from the RHEV Manager to be enabled.

As a result of these discussions with Red Hat support a KB article has been created so that these vdsm hooks can be acquired by request.

https://access.redhat.com/solutions/2024723

According to Red Hat support this feature will be enabled in a future version of RHEV.

At the time it was not possible to install the vdsm hook on the hypervisor running on the x86 server. The vdsm hook was however installed on the hypervisor running on the Power server.

We performed two tests on the x86 server. One test with threads per core set to 1 (SMT = 1) in the hypervisor host and one test with threads per core set to 2 (SMT = 2) in the hypervisor host. We then created one virtual machine at a time with 16 respectively 32 virtual cores both using 16 physical cores.

On the Power server a virtual machine with 8 physical cores and threads per core set to 4 (SMT = 4) which equals to 32 virtual cores was created.

As a result, we were running 16 physical cores on the x86 platform and 8 physical cores on the Power platform.

Setting up the environment

The whitepaper provided by IBM assumes that you start with an already existing Linux environment. As a foundation we installed identical Linux environments on virtual machines.

A fresh install of Red Hat Enterprise Linux 6.7 on both the x86 and Power virtual machines was done. Followed by applying the latest fixes.

```
subscription-manager register
subscription-manager subscribe --auto
yum update
```

On the virtual machine located on the Power server you need to install the the IBM Advance Toolchain for Power to ensure optimum performance.

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W51a7ffcf4dfd_4b40_9d82_446ebc23c550/page/IBM%20Advance%20Toolchain%20for%20PowerLinux%20Documentation?section=distros

- For RHEL6 use advance-toolchain-at7.0
- For RHEL7 use advance-toolchain-at8.0

The gpg public keys `gpg-pubkey-6976a827-5164221b` (for **RHEL6/7**, **SLES 12** and **Ubuntu**) and `gpg-pubkey-3052930d-5175955a` (for **RHEL6, RHEL5, SLES 10 and SLES 11**) are provided in the repositories. These public keys can be used to verify the authenticity of both the Advance Toolchain packages and the repository contents. Download the gpg public key for your Linux distribution and import it using the following commands:

```
wget ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/redhat/RHEL6/gpg-pubkey-6976a827-5164221b

rpm --import gpg-pubkey-6976a827-5164221b

wget ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/redhat/RHEL6/gpg-pubkey-3052930d-5175955a

rpm --import gpg-pubkey-3052930d-5175955a
```

Create the following file using a text editor of your choice:

```
vi /etc/yum.repos.d/at7.0.repo
```

and add the following content:

```
[at7.0]
name=Advance Toolchain Unicamp FTP
baseurl=ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/redhat/RHEL6
gpgkey=ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/redhat/RHEL6/gpg-pubkey-6976a827-5164221b
       ftp://ftp.unicamp.br/pub/linuxpatch/toolchain/at/redhat/RHEL6/gpg-pubkey-3052930d-5175955a
failovermethod=priority
enabled=1
gpgcheck=
```

Install the advance-toolchain packages:

```
yum install advance-toolchain-atX.X-runtime \
            advance-toolchain-atX.X-devel \
            advance-toolchain-atX.X-perf \
            advance-toolchain-atX.X-mcore-libs
```

Run ldconfig from the advance toolchain:

```
/opt/at7.0/sbin/ldconfig
```

From here on you can follow the instructions in the whitepaper provided by IBM but for your convenience a short summary is included.

The EnterpriseDB binaries can easily be downloaded from the yum repository provided by EnterpriseDB. If you don't have access to the yum repository you need to contact sales@enterprisedb.com who will then provide you with login information.

Create configuration files for the EnterpriseDB repositories. Notice the difference depending on if you are downloading the x86_64 or ppc64 (Power) binaries. Replace the username:password with your login credentials provided by EnterpriseDB.

x86

```
vi /etc/yum.repos.d/ppas.repo
[ppas93]
name=PPAS
baseurl=http://username:password@yum.enterprisedb.com/9.3/redhat/rhel-latest-x86_64
enabled=1
gpgcheck=0
```

Power

```
vi /etc/yum.repos.d/ppas.repo
[ppas93]
name=PPAS
baseurl=http://username:password@yum.enterprisedb.com/9.3/redhat/rhel-latest-ppc64
enabled=1
gpgcheck=0
```

Install EnterpriseDB by running the following command:

```
yum install ppas93-server
```

We followed the instructions in the whitepaper and did the following:

Enabled Dynamic Power Saver (favor performance) mode option in the ASMI interface.

Executed the kernel tunings and OS tunings described in the original whitepaper

Created a 20 GB RAM disk.

Initialized the DB cluster instance on the RAM disk.

Tuned the EnterpriseDB Postgres Plus Advanced Server parameters in postgresql.conf (We then copied this configuration file to a persistent filesystem to be able to easily recall the parameters whenever we needed to recreate the RAM disk)

Started the DB cluster and created a DB

Initialized the database

Ran the benchmarking tool

At the end of each run, this utility displays the results as transactions per second (TPS) including and excluding connections.

Test performed

We created a text file so that we could easily copy & paste the commands to be able to conveniently redo the test multiple times. This can easily be done using a script as well.

List of commands to run (as described in the original whitepaper)

```
# As root

sysctl -w fs.file-max=65535
sysctl -w vm.dirty_background_bytes=37108864
sysctl -w vm.dirty_ratio=40
sysctl -w vm.dirty_background_ratio=40
sysctl -w vm.hugetlb_shm_group=`id -g enterprisedb`
sysctl -w vm.dirty_bytes=296870912
sysctl -w vm.nr_hugepages=2000
sysctl -w vm.swappiness=0
sysctl -w vm.hugepages_treat_as_movable=0
sysctl -w vm.nr_overcommit_hugepages=512
sysctl -w vm.zone_reclaim_mode=0
sysctl -w vm.drop_caches=3
sysctl -w kernel.sched_migration_cost=500000
sysctl -w kernel.sched_autogroup_enabled=1

ppc64_cpu --smt-snooze-delay=16777215

mount -t debugfs debugfs /sys/kernel/debug
echo NO_WAKEUP_PREEMPT > /sys/kernel/debug/sched_features

ppc64_cpu --dscr=1

mkdir -p /media/tmp
mount -t tmpfs -o size=20G tmpfs /media/tmp # Mounts the RAM FS

# Switch to enterprisedb user
su - enterprisedb


initdb -D /media/tmp/data

# Copy the tuned postgresql.conf file back to RAM FS
cp /home/edb/postgresql.conf /media/tmp/data

pg_ctl -D /media/tmp/data -l /home/edb/logfile start

createdb pgbench

pgbench -i -s 1000 pgbench

pgbench -n -S -T 300 -c 100 -j 100 pgbench
```

Results

Screendumps from the test is showed here followed by a summary.

Result on x86

16 physical core SMT = 2 (enabled in host)

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    1
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Stepping:              1
CPU MHz:               2892.748
BogoMIPS:              5785.49
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):     0-31

# pgbench -n -S -T 300 -c 100 -j 100 pgbench
transaction type: SELECT only
scaling factor: 1000
query mode: simple
number of clients: 100
number of threads: 100
duration: 300 s
number of transactions actually processed: 65199131
tps = 217307.943121 (including connections establishing)
tps = 217409.898803 (excluding connections establishing)
```

tps / core = 217.307 / 16 = 13.581 tps / core

Out of curiosity we ran the same test again after disabling SMT on the x86 platform as we have clients that has this disabled in their RHEV environments with the assumption that the performance can degrade if it is enabled in the hosts. In our extreme example running 1 virtual machine with all 16 physical cores with either SMT = 2 enabled or disabled we actually got 40-45% better performance with SMT = 2 enabled.

16 physical core SMT = 1 (disabled in host)

```
# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    1
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Stepping:              1
CPU MHz:               2892.748
BogoMIPS:              5785.49
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):     0-15

# pgbench -n -S -T 300 -c 100 -j 100 pgbench
transaction type: SELECT only
scaling factor: 1000
query mode: simple
number of clients: 100
number of threads: 100
duration: 300 s
number of transactions actually processed: 44565909
tps = 148537.972211 (including connections establishing)
tps = 148561.290478 (excluding connections establishing)
```

tps / core = 148.537 / 16 = 9.283 tps / core

Results on Power

8 physical core SMT = 4 (enabled in guest)

```
# lscpu
Architecture:          ppc64
Byte Order:            Big Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    4
Core(s) per socket:    1
Socket(s):             8
NUMA node(s):          1
Model:                 IBM pSeries (emulated by qemu)
L1d cache:             64K
L1i cache:             32K
NUMA node0 CPU(s):     0-31

# pgbench -n -S -T 300 -c 100 -j 100 pgbench
transaction type: SELECT only
scaling factor: 1000
query mode: simple
number of clients: 100
number of threads: 100
duration: 300 s
number of transactions actually processed: 75626752
tps = 251960.194070 (including connections establishing)
tps = 252020.423657 (excluding connections establishing)
```

tps / core = 251.960 / 8 = 31.495 tps / core

We also tried to maxout the virtual machine by enabling all 16 physical cores using SMT=4.

```
pgbench -n -S -T 300 -c 100 -j 100 pgbench
transaction type: SELECT only
scaling factor: 1000
query mode: simple
number of clients: 100
number of threads: 100
duration: 300 s
number of transactions actually processed: 120189480
tps = 400625.047110 (including connections establishing)
tps = 400783.259258 (excluding connections establishing)
```

tps / core = 400.625 / 16 = 25.039 tps / core

Summary of test results

The results were on par with the results presented in the original whitepaper.

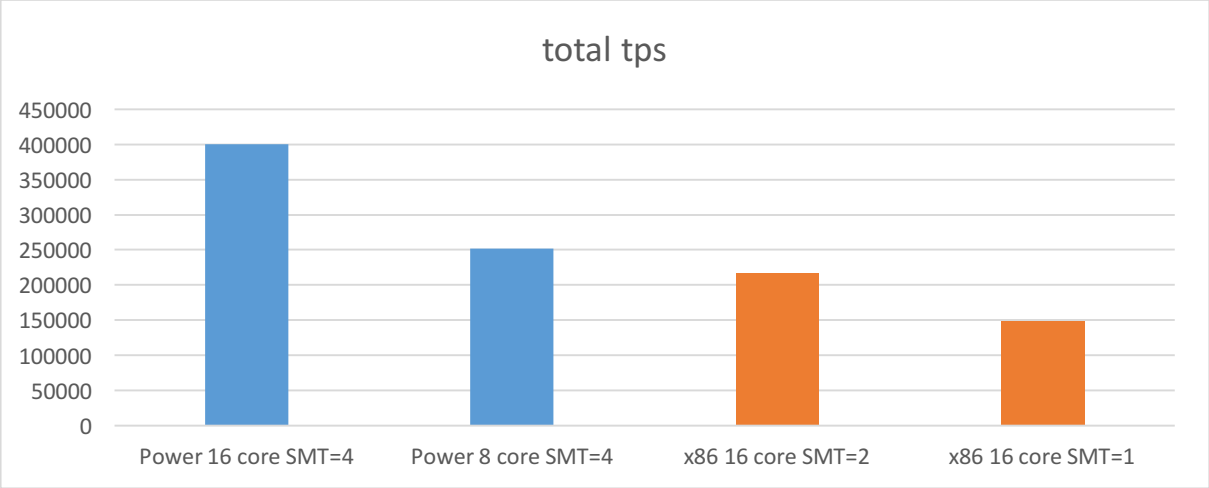
Using the numbers from above the tps / core ratio between POWER8 & x86 would give us a ratio between 1.8 – 2.3X

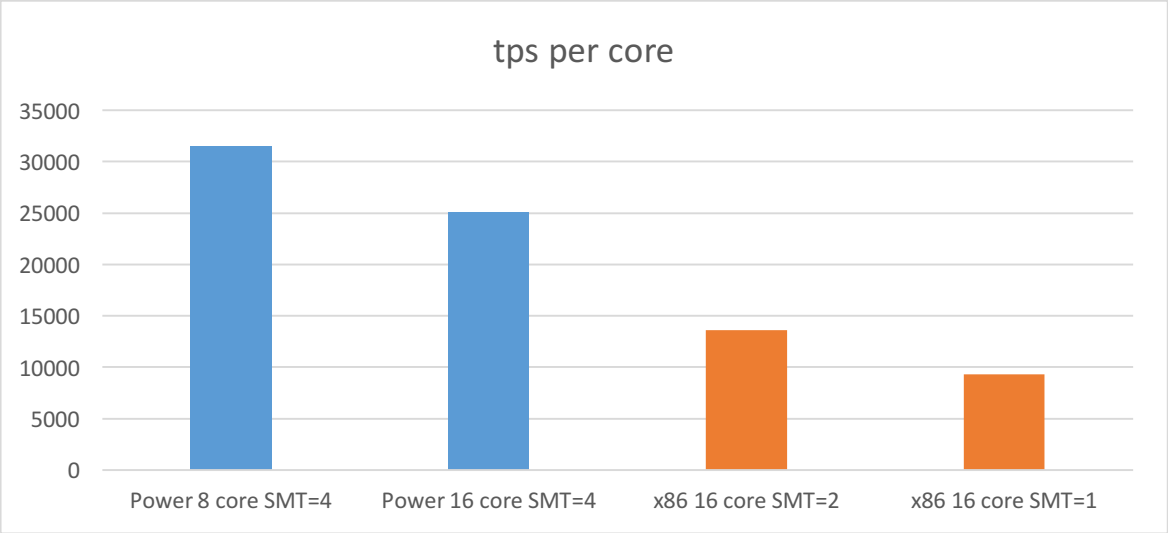
25.039 / 13 581 = 1.8 X
31 495 / 13 581 = 2.3 X

The differences in our results compared to the original whitepaper is probably a combination of us using different hardware as well as later Red Hat versions compared to the original whitepaper.

Even though we only present a single screenshot from the command we executed them several times with several reboots of the virtual machine between the tests to verify consistency with the results.

To get the most of the Power server we recommend finding the optimum balance between number of CPUs allocated to each virtual machine and number of virtual machines per physical server. In our scenario it would be more beneficial running 2 virtual machines with 8 physical cores per virtual machine instead of 1 virtual machines with 16 physical cores.





Try it yourself

These tests were done with the database located on a RAM FS to reduce I/O bottlenecks and to focus on CPU performance alone.

Pgbench which comes bundled with EnterpriseDB is a great tool to evaluate database performance between different hardware platforms, OS platforms, virtualization platforms as well as different storage solutions.

So don’t take our word for it. You can easily go ahead and verify these performance benefits yourself.

Kenneth Albinsson
Conoa AB
Enterprise Open Source Experts

About Conoa

Conoa specializes in Enterprise Linux and Open Source solutions. Conoa analyzes, design, implement, maintain and support IT infrastructure based on Enterprise Linux and Open Source software. Our consultants are experts with extensive experience and knowledge from large corporate environments. We also have strong partnerships with leading companies within the Open Source Ecosystem including IBM (Open POWER), EnterpriseDB and Red Hat.

Please visit us at www.conoa.se