IBM Power solution for EDB Postgres Advanced Server

A paper on PostgreSQL and EDB Postgres Advanced Server running Linux on IBM Power10 processor-based servers – installation and tuning best practices for IBM Power servers

Overview

Challenge
Customers encounter difficulties when installing EDB and Postgres, configuring the underlying hardware, and tuning the operating system and database parameters to get best throughput from a system.

Solution
This paper describes the methods to install EDB and Postgres and provides the steps to optimize performance by making configuration changes on the IBM Power server using HMC and providing the relevant options to tune the kernel and the operating system.

Table of contents

The primary focus of this paper is on the use, configuration, and optimization of EDB Postgres Advanced Server running on the IBM® Power® servers featuring the new IBM Power10 processor technology.

Note: The scope of this paper is to provide information on how to install and configure EDB Postgres Advanced Server on an IBM Power server for better use. EDB Postgres Advanced Server on IBM Power servers running Linux is based on the open source database PostgreSQL and can handle a wide variety of high-transaction and heavy-reporting workloads.

This paper describes the procedure for general installation and tuning of EDB Postgres Advanced Server database on IBM Power servers running Linux. IBM Power servers offer significant advantages compared to similar configurations of Intel Ice Lake processors

General recommendation

Before installing EDB and PostgreSQL, it is generally recommended to have a root and data disk on separate physical disk or Redundant Array of Independent Disks (RAID).

EDB Postgres Advanced Server on IBM Power servers running Linux

EDB Postgres Advanced Server is built on PostgreSQL, one of the most advanced open source databases and has additional functionality and capabilities in the following areas:

- Performance
- Compatibility
- Security
- Tooling

With the Oracle compatibility features of EDB Postgres Advanced Server, you can use existing Oracle-based applications on a low-cost, high-performance PostgreSQL-based platform, or run adjacent applications that integrate seamlessly with your mission-critical databases without additional Oracle licenses.

Always ensure that EDB Postgres Advanced Server is installed with the latest service pack available from the EDB website at: https://www.enterprisedb.com/software-downloads-postgres

Architecture

Software
- Red Hat Enterprise Linux (RHEL) 8.6

Hardware
- Processor – IBM Power 10
- 16 Core, 1 TB Memory
- System divided into 4 LPARs with each partition allocated 4 cores and 250 GB memory

Prerequisites

Refer to EDB manuals for software prerequisites.

- Confirm that there is adequate paging space and free space on the /tmp and / (root) file systems.
- Verify the ulimits setting for each of the products using the `ulimit –a` command.
- Refer to the EDB tuning guide and best practices guide: https://www.enterprisedb.com/postgres-tutorials/introduction-postgresql-performance-tuning-and-optimization.

EDB Postgres Advanced Server setup requires Java™. Install OpenJDK using the `$ yum install java` command.

Setting up PostgreSQL and EDB Postgres Advanced Server

EDB Postgres Advanced Server can be installed using the rpm, dnf, or yum command. It is recommended to perform the installation using the yum tool.

First, you need to make sure whether a yum repository for RHEL is defined. Also, you can define a yum repository for EDB RPMs for easy one-step installations. Defining a yum repository for the base OS and software installation packages enables the installation of software to automatically fetch the RPM packages (which are the prerequisites).

Configuring the yum repository

Confirm the yum repository defined for the base OS as it helps. A RHEL yum repository can be defined by pointing to a CD/DVD media, a shared network location, or a local directory that contains the RHEL installation media. The test team defined it using a local directory. The yum configuration files are located in the /etc/yum.repos.d/ directory.

You can find more information about yum repositories at:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/sec-configuring_yum_and_yum_repositories

Installing EDB Postgres Advanced Server

You can find details about installing EDB Postgres Advanced Server at: https://www.enterprisedb.com/docs/epas/latest/epas_inst_linux/installing_epas_using_edb_repository/ibm_power_ppc64le/epas_rhel8_ppcle/

Installation of EDB Postgres Advanced Server running on IBM Power server on Linux is done using the `yum` or `dnf` repository definition as provided by EDB. Perform the installation of EDB Postgres Advanced Server using a simple install command. For example:

```
$ dnf -y install edb-as<xx>-server
```

Where <xx> is the version of the EDB Postgres Advanced server. This command automatically installs all the required prerequisites, and creates a database administrator user *enterprisedb*, if not already present.

Before proceeding with creating and configuring the DB instance, it is best to apply tunings. In this exercise, the test team performed various system tunings, including kernel tuning.

Building and installing the PostgreSQL Server

Perform the following tasks to build and install the PostgreSQL Server:

1. Get the PostgreSQL source from their official website: https://www.postgresql.org/ftp/source/. After getting the appropriate PostgreSQL version .tar file, extract it.

   ```
   $ tar –zxvf postgresql-X.tar.gz

   $ cd postgresql-X
   ```

   Where X is the version of PostgreSQL. After extracting the content, you can find a directory named, postgresql-X.

2. Configure the source tree.
   This step describes how to configure the source tree for the system and enable users to choose the required compiler options. Ensure that the following packages are installed:

   a. make
   b. readline-devel
   c. zlib-devel

3. Configure PostgreSQL using GCC.

   GCC is part of the OS repository, and therefore, ensure that the latest version of GCC is installed on the server. Run the following command to install GCC and other dependent packages on the system.

   ```
   $ yum install gcc -y
   ```

By default, the configure script compiles using the latest GCC compiler available on the system. In case GCC is not available on the default path or multiple versions of GCC are installed, specify the absolute path.

```
$ ./configure CC=<GCC Directory>/bin/config-file
```

4. After successful completion of configuration, you can now build and install PostgreSQL using the following commands:

```
$ make all
$ make install
```

After successful installation, you can find the binary files in the /usr/local/pgsql/bin/ folder and the required libraries and header files in the /usr/local/pgsql/include/ folder.

Power Systems configuration

The following settings were used for this exercise.

- Divide the whole system into four logical partitions with all cores dedicated. Hardware Management Console (HMC) access is needed to create these LPARs.
- Allocate each partition with four cores and 250 GB of memory each.
- Disable the following power saving options for the Power10 processor-based system.
  - Turn off the **Idle Power Saver** option, this is the default option and it’s not supported on Power10.
  - Set **Power Saver mode** to **Maximum Performance**.

This setting change can be performed only through the Power Saver Management UI console which can be accessed using HMC or directly through a browser using the managed system's IP address.

Open the **Power Management** operation from HMC for the system. The following are the reference screen captures.

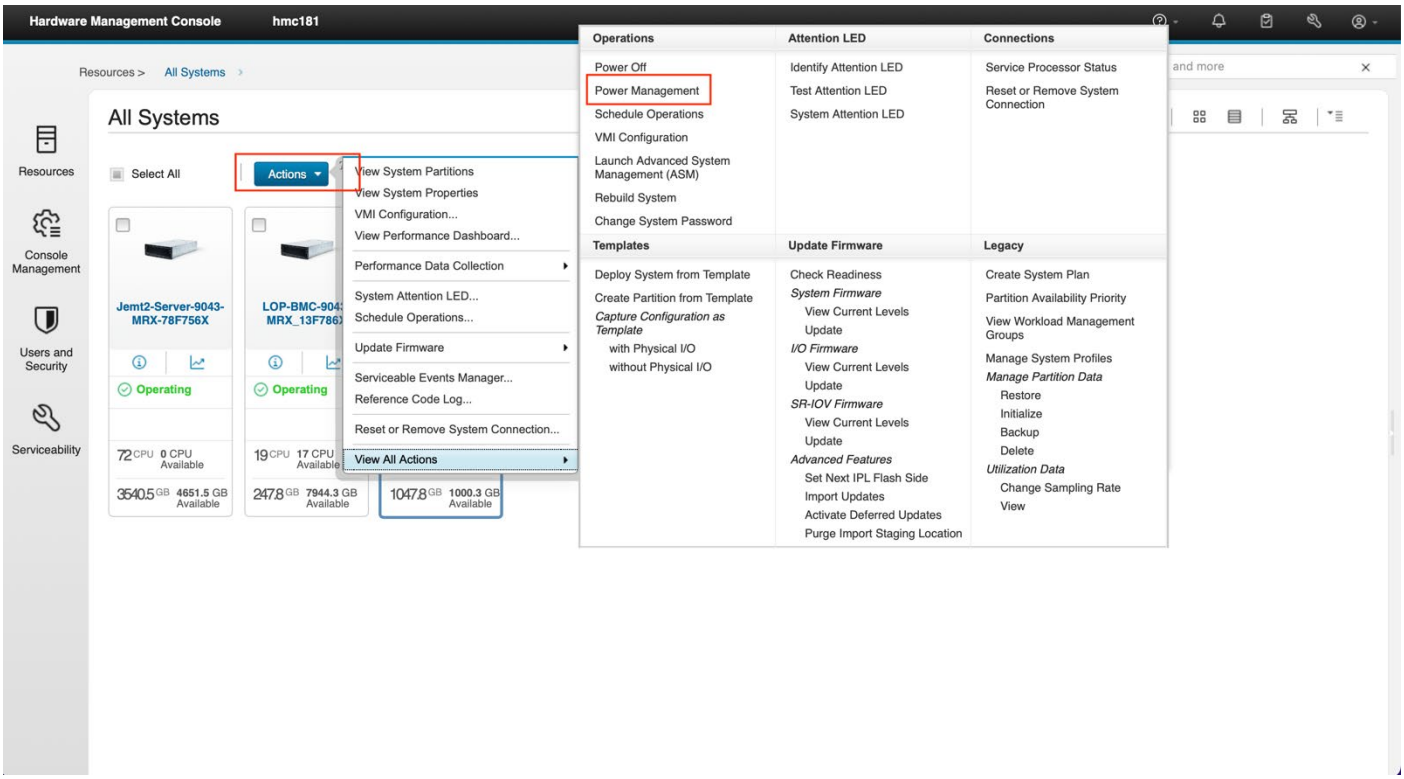

Figure 1. Accessing Power Management from HMC

To view the **Idle Power Saver** option, log in and click **View All Actions** -> **Power Management** -> **Idle Power Saver**.

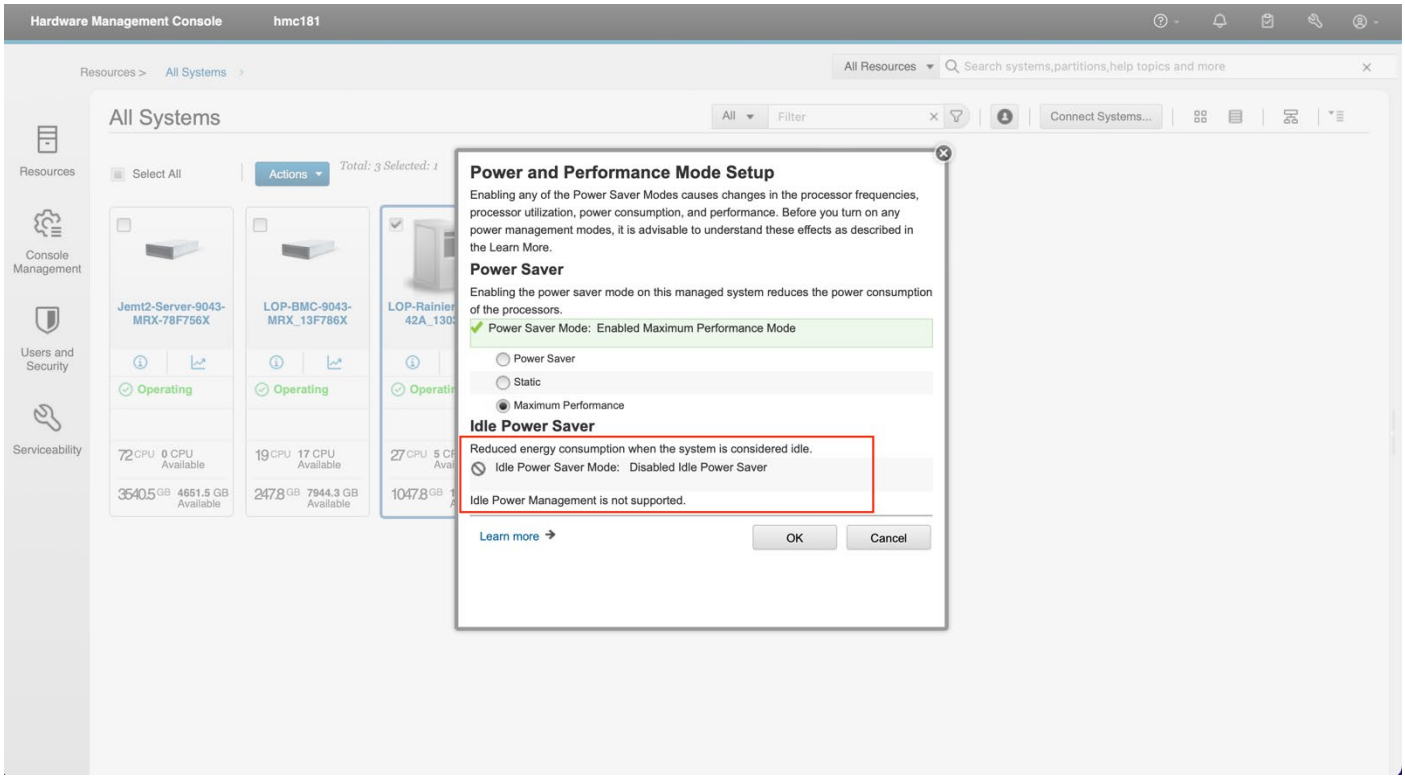

Figure 2. Power Management – Idle Power Saver page

To enable the power saver mode, perform the following steps:

1. Click **Power Management** and in the Power Saver section, select the **Maximum Performance** option.
2. Click **OK**.

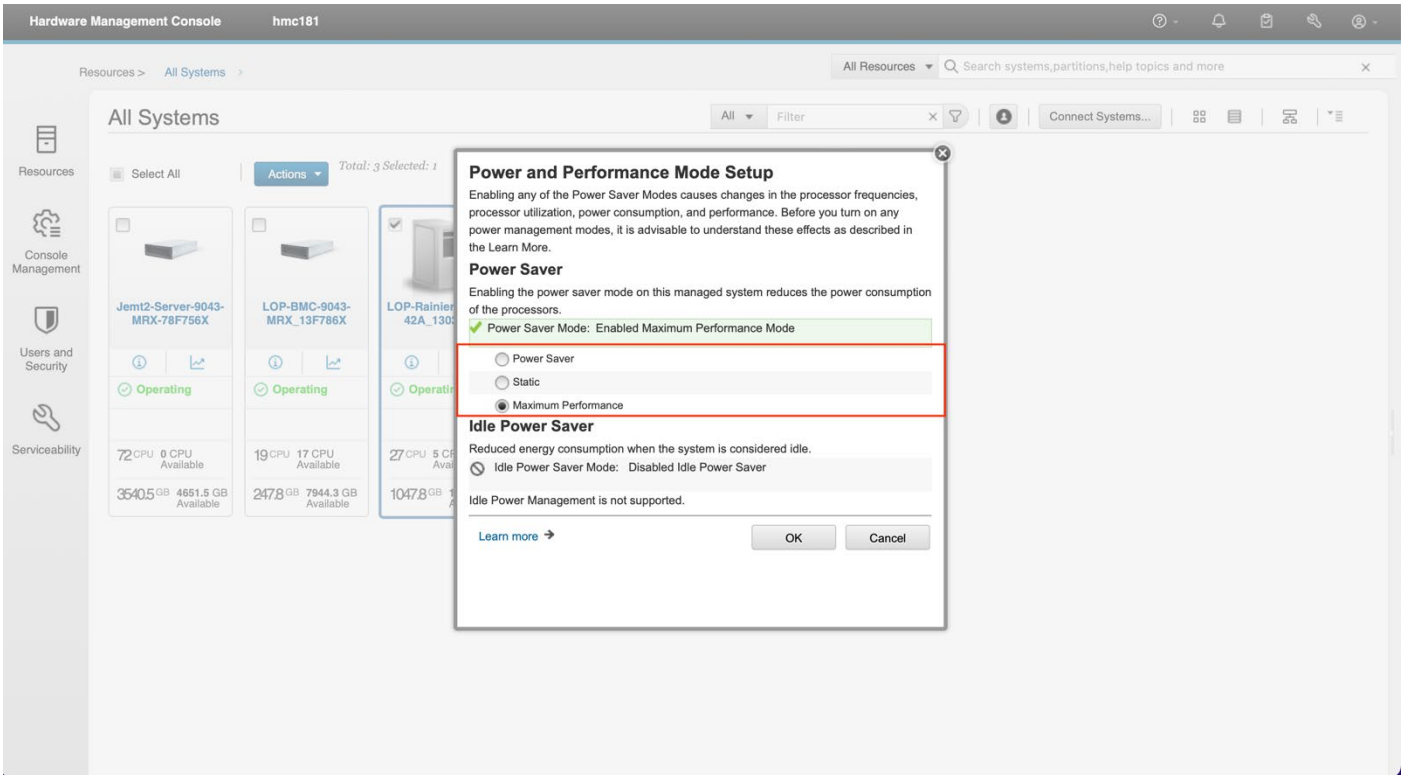

Figure 3. Power Management – Power Saver

These settings help in increasing the processor speed. This can be verified at the shell by using the $ `ppc64_cpu –frequency` command. This operation does not require the system to be shut down or restarted. It takes effect immediately.

Note: The Power Management options might have moved around during future updates and releases of firmware. Just look around and find the required configuration.

Note: The Advanced System Management Interface (ASMI) options might not be available if a firmware other than IBM PowerVM, such as OPAL, is set. In such cases, set the processor mode through the Linux command line using the **cpupower** utility with the **governor** option set to **performance mode**. This setting must be applied to all the cores on the system.

For example: `$ cpupower frequency-set --governor performance`

You can find more information about power management settings at:

https://access.redhat.com/documentation/fr-fr/red_hat_enterprise_linux/8/html/monitoring_and_managing_system_status_and_performance/tuning-cpu-frequency-to-optimize-energy-consumption_monitoring-and-managing-system-status-and-performance

Kernel tunings and OS tunings

The following kernel parameters have shown favorable results in internal tests. These kernel parameter settings are to be applied as a root user. Perform the activity with caution and set the values according to the system's hardware specifications and operating system implementation.

- `$ fs.file-max=65535`
- `$ vm.zone_reclaim_mode=0`
- `$ vm.drop_caches=3`
- `$ vm.dirty_ratio=60`
- `$ vm.dirty_background_ratio=60`
- `$ vm.dirty_background_bytes=37108864`
- `$ vm.dirty_bytes=296870912`
- `$ vm.hugetlb_shm_group=`id -g <group_name>``
- `$ vm.hugepages_treat_as_movable=0`
- `$ vm.nr_hugepages=2000`
- `$ vm.nr_overcommit_hugepages=512`
- `$ kernel.sched_autogroup_enabled=1`
- `$ vm.swappiness=0`

Set the following parameters based on the memory available on your system:

- `$ shmall shmall=$(( `grep MemTotal /proc/meminfo |awk '{print $2}'` * 1024 * 9 / (`getconf PAGE_SIZE` * 10)))`
- `$ sudo sysctl -w kernel.shmall=$shmall`
- `$ shmmax shmmax=$(( `grep MemTotal /proc/meminfo |awk '{print $2}'` * 1024 * 8 / 10 ))`
  `$ sudo sysctl -w kernel.shmmax=$shmmax`

Apply the following OS changes. These tuning changes in the number of TPS on pgbench testing is shown to be beneficial.

- Set the processor's simultaneous multithreading (SMT) snooze delay to zero using the following command. `$ ppc64_cpu --smt-snooze-delay=0`
- Run the following command on your Linux system to disable hardware data prefetch (usually to improve Postgresql performances). `$ ppc64_cpu -dscr=1`
- Stop and disable firewall.
  - `$ systemctl stop firewalld`
  - `$ systemctl disable firewalld`

EDB/PostgreSQL database setup

This section explains how to initialize a database cluster instance, apply the DB cluster parameter tunings, and create a database.

EDB

- Configure Repository: `$ dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm`
- Configure yum with EDB repo: `$ sed -I“"s@<username>:<passwor”" /etc/yum.repos.d/edb.repo`
- Configure EPEL repository: `$ dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm`
- Register System: `$ subscription-manager repos --enable "codeready-builder-for-rhel-8-${ARCH}-rpms"`
- Disable built-in PostgreSQL module: `$ dnf -qy module disable postgresql`
- Install EDB server: `$ dnf -y install edb-as14-server`
- Initialise initdb options: `$ PGSETUP_INITDB_OPTIONS="-E UTF-8" /usr/edb/as14/bin/edb-as-14-setup initdb`
- Start database cluster: `$ systemctl start edb-as-14`

PostgreSQL

- Create a directory to stage the PostgreSQL database files. Create a user (for example, enterprisedb) and set the home directory to the user.
  - `$ useradd enterprisedb`
  - `$ mkdir /dbdirectory/`
  - `$ usermod -d /dbdirectory/ enterprisedb`
  - `$ chown enterprisedb:enterprisedb /dbdirectory/`
  - `$ chmod -R 755 /dbdirectory/`
- Switch to the enterprisedb user and create a new PosgreSQL database instance under enterprisedb’s home directory using the `initdb` command.
  - `$ initdb -D /dbdirectory/data/`
- Start the PostgreSQL server using the `pgctl` command.
  - `$ pg_ctl -D /dbdirectory/ logfile start`

Tune the DB server

Find the default DB cluster parameter in the postgresql.conf file. For this exercise, change the following parameters to the recommended values based on the selected workload (pgbench). Parameters might differ according to the specific workload being run.

- `shared_buffers = 60 GB` (Tune this based on the memory allocated for the LPAR. The recommendation is that this cannot exceed 1/4th of the LPAR memory.)
- `checkpoint_completion_target = 0.9`
- `work_mem = 1 GB`
- `wal_buffers = 32 MB`

Note: Some of the parameter tunings might become invalid in the future release of the product. If any parameter fails or displays an error, comment it out. As a general recommendation, if a predefined parameter named in the list provided in this section is not found, skip it.

Start the DB cluster and create a DB

- Start the DB cluster using the `pg_ctl` utility that comes with the server binaries. Redirect the output to a file using the –l option
  - `$ /usr/edb/as14/bin/pg_ctl start -D /var/lib/edb/as14/data -l logfile start`
- Create a database using the `createdb` command-line utility. Here, `pgbench` is chosen as the database name.
  - `$ createdb pgbench`

PostgreSQL and EDB Postgres Advanced Server database benchmarking

This section discusses the benchmark test that the test team ran on the database after initializing the workload. Reinitializing the DB is not needed for every run, but is recommended as it refreshes the DB.

Benchmarking tool (pgbench)

`pgbench` is a simple utility to run the benchmark tests on EDB Postgres Advanced Server or PostgreSQL. Find more information about this utility at: https://www.postgresql.org/docs/devel/static/pgbench.html

`pgbench` offers various read-only (select only query) and read/write (`select`, `update`, and `insert`) query modes. For this exercise, the `select only` option of pgbench was used. This utility can be run on the same system for PostgreSQL or EDB Postgres Advanced Server. You can also run this utility on a separate computer over the network.

Initialize the database

First, the database must be initialized. Invoke the `pgbench` utility using the `-I` option and specify a scaling factor. In this exercise, the test team used a scaling factor of 1000.

```
$ pgbench -i -s 1000 pgbench
```

Initialization takes some time as it populates the DB. Scaling of 1000 typically consumes around 16 GB of memory.

Run the benchmarking tool

Use the following command to run the benchmarking tool:

```
$ pgbench -n -S -c 32 -j 32 pgbench -T 600
```

- -T specifies the duration for which the tool runs.
- -S enables *select only* loads.
- -c is the number of clients.
- -j is the number of worker threads.

This run is performed for various client and thread counts. For this test, the client and thread resided on the same core. The pgbench tool is run for 5 minutes (300 seconds). For stable results, running pgbench for 5 minutes or longer is recommended.

Figure 4. Sample result output from pgbench tool

```
 This is a Read-ONLY Test
/usr/edb/as14/bin/pgbench -n -S -T 600 -c 32 -j 32 pgbench
pgbench (14.4.0, server 14.4.0)
transaction type: <builtin: select only>
scaling factor: 1000
query mode: simple
number of clients: 32
number of threads: 32
duration: 600 s
number of transactions actually processed: 193995697
latency average = 0.099 ms
initial connection time = 6.806 ms
tps = 323329.258622 (without initial connection time)
waiting for server to shut down.... done
server stopped
./auto-run-test.sh: line 433: kill: (2912517) - No such process
Number of cores online = 4
CPU(s):              32
SMT=8
MemTotal:       257595968 kB
Client    TPS
------    ---
32      324233.743775
```

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

Summary

In this paper, we described how to set up EDB and PostgreSQL on IBM Power, and also how to optimize and tune them. The sample benchmark results from pg_bench demonstrated how you can achieve the best throughput on IBM Power by adhering to suggested power setting modifications using HMC and applying proper configuration and tuning options based on the type of application that is deployed.

About the authors

Calvin Sze is a performance optimization engineer with 18 years of performance experience. Currently, Calvin is focusing on open source database and AI performance optimization for ISVs on IBM Power platform. Before that, Calvin was a performance analyst for various software stacks on IBM AIX® and Linux®. You can reach Calvin Sze at calvins@us.ibm.com

Kemparaju is a technical specialist and is part of the ISV team in IBM India Systems development Lab. He has experience in MongoDB, EDB, web development, and distributed systems. He is currently helping clients on digital and cloud transformation with open source software porting and benchmarking. You can reach Kemparaju at kemparaju.kemparaju@ibm.com

Mithun H R is a technical specialist and is part of ISV team in IBM India Systems development Lab. He has been a developer, a consultant, and a research associate over the years and has experience in retail, defense, and aerospace functional domains. He is currently helping clients on digital and cloud transformation with open source software porting and benchmarking. His has experience in MongoDB, EDB, and RedHat OpenShift on the IBM Power platform. You can reach Mithun at mithunhr@in.ibm.com

IBM Systems
3039 Cornwallis Road
RTP, NC 27709

Produced in the United States of America