AMD High Performance Linpack Benchmark

This document describes building the components to run the High Performance Linpack (HPL) benchmark using the AMD xhpl binary, HPL.dat and script files.

HPL Implementation:
The open source BLIS library (BLAS-like Linear algebra Instantiation Library) is used for DGEMM, which performs the majority of the computations for HPL. This library can be optionally configured with threading support (POSIX threads or OpenMP). The library comes with several pre-built optimized kernels including an optimized AVX2 kernel for AMD Zen.

The ideal scalable implementation for an EPYC system is a hybrid approach where OpenMPI is used at the top level(s) of the hierarchy and a multi-threaded (OpenMP) client is used for each OpenMPI rank targeting a single shared L3 cache instance within the EPYC architecture. Within a cache coherent domain (for example, within a server), the zero copy KNEM shared memory module is used for OpenMPI communication among clients. Outside of this domain (for example from server to server), RDMA or Ethernet would be used.

To take maximum advantage of the Zen architecture, the OpenMPI rank running the multi-threaded client is bound to an L3 cache domain, with one computational child thread per physical core sharing the cache (SMT disabled or unused). The AVX2 DGEMM computation is SIMD bound so an additional active thread per core hinders rather than helps performance. If SMT is enabled, each child thread is bound to only the lower numbered of the two logical CPUs per core. The second CPU is left idle. Note the script has dependency on logical CPU numbering as specified by the ACPI MADT supplied by UEFI Firmware. Dell 14G PowerEdge systems use different CPU numbering compared to other systems in the market and will require updates to the script to accommodate.

The Hardware Locality library (hwloc) is used to allow OpenMPI to automatically extract platform topology information for process and memory binding. This feature can be used with a single-threaded BLIS implementation where all inter-process communication is through OpenMPI, although this feature is currently unused.

Note for 24/16/8 core EPYC SKUs (7451, 7401, 7401P, 7351, 7351P, 7301, 7251):
The example in this ap-note and the sample scripts provided by AMD are based on a 32 core device. The HPL.dat file discussed later in this document and the CPU binding will need to be modified for 24 and 16 core machines.

Also, Linux kernels prior to 4.14 exhibit a bug whereby the kernel generated map of which logical CPUs share which L3 cache is incorrectly generated for the 24/16/8 core SKUs. This bug will cause libhwloc to generate a warning when started. Provided all rank to L3 and CPU binding within a rank steps are followed, this warning can be ignored. A patch is available on most common kernel versions (3.10, 4.4, 4.8, 4.10, 4.13) to address this bug. An example of the error message is below:

```
**************************************************************************
* hwloc 1.11.2 has encountered what looks like an error from the operating system.
*
* L3 (cpuset 0x60000060) intersects with NUMANode (P#0 cpuset 0x3f00003f) without
inclusion!
* Error occurred in topology.c line 1046
*
* The following FAQ entry in the hwloc documentation may help:
*   What should I do when hwloc reports "operating system" warnings?
* Otherwise please report this error message to the hwloc user's mailing list,
* along with the output+tarball generated by the hwloc-gather-topology script.
**************************************************************************
```

Build dependencies:
g++ (gcc-c++ on RHEL)
Kernel headers for the running kernel (to build KNEM): linux-headers on Ubuntu, kernel-devel on RHEL
libnuma-dev (Ubuntu - for building OpenMPI)
hwloc
libhwloc-dev (Ubuntu) or hwloc-devel (RHEL) (for building OpenMPI and KNEM)

Additional packages sources:
OpenMPI: https://www.open-mpi.org/software/ompi/v3.0
BLIS: https://github.com/flame/blis/tree/amd
KNEM: http://knem.gforge.inria.fr/download

System configuration steps:
On Red Hat Enterprise Linux / CentOS, if running ktune or tuned, the recommended tuned profile for HPL is throughput-performance.

On Ubuntu, insure that the ondemand or performance cpufreq governor is used.

AMD recommends disabling SMT in the UEFI Firmware for this benchmark, setting the determinism slider to Power Determinism mode, enabling manual cTDP override and setting the cTDP to 200 watts. Different platform vendors may use different nomenclature for these settings.

Memory speed should be limited to 2400 MHz to provide more power headroom for the CPUs.

Build steps:

The following steps will install the extra packages to /opt – you can change this as needed with `--prefix={/path you want}` on the `./configure` commands:

1) Build and install KNEM
    a) Extract the source tarball and cd into the created directory
    b) `CFLAGS="-Ofast -march=native" ./configure --prefix=/opt/knem`
    c) `make`
    d) `sudo make install`
    e) `sudo /opt/knem/sbin/knem_local_install`

2) Build BLIS
    a) Extract the source tarball and cd into the created directory
    b) `./configure --prefix=/opt/blis --enable-threading=openmp zen`
    c) `make –j 32`
    d) `sudo make install`

3) Build OpenMPI
    a) Extract the source tarball and cd into the created directory
    b) `CFLAGS="-Ofast -march=native" ./configure --prefix=/opt/openmpi --with-knem=/opt/knem`
    c) `make –j 32`
    d) `sudo make install`
    e) `sudo ln –s /opt/openmpi/lib/libmpi.so.40.0.0 /opt/openmpi/lib/libmpi.so.20`

Step e above is only necessary if building OpenMPI 3.x – it is not required if building OpenMPI 2.x

Running on a 2P EPYC system:

1) `./run_hpl_ccx.sh` (The script will automatically load the knem module if not already loaded

The benchmark will take about an hour to run on a 2P machine with 256GB.

Running on a 1P EPYC system:

1) Edit the HPL.dat file to change the Ps line from 4 to 2. This will change the number of process grids from 16 (4 x 4) to 8 (2 x 4). You may also need to adjust the problem size N to account for less total system memory. See below for more details.
2) `./run_hpl_ccx_1p.sh`

Configuring HPL.dat

HPL.dat defines the problem (matrix size) and some parameters of how the solution is computed for the benchmark. Some parameters you can adjust:

Number of Process Grids (P, Q):

The number of expected process grids (P x Q) must match the number of xhpl clients that will be started to complete the calculations. P and Q should be as close to each other as possible. If the numbers are not equal, Q should be larger. For this hybrid OpenMPI + OpenMP implementation on EPYC, the number of process grids should match the total number L3 caches in the system. For a 2P system there are 16 L3 caches thus P = Q = 4 (P x Q = 16). For a 1P system use P = 2, Q = 4 (P x Q = 8).

Block size (NB):

HPL uses the block size NB for the data distribution as well as for the computational granularity. A block size of 232 or 240 is optimal for EPYC. It should not normally be necessary to adjust this parameter.

Problem size (N):

HPL performs calculations on an N x N array of double precision elements. Each double precision element requires 8 bytes. Thus the memory consumed for a problem size of N is $8N^2$. Assuming K machines with equal memory J, the total available memory M is K x J. The problem size should be set to a multiple of the block size (above) times the total number of cores on all machines, and should be the largest size possible without exceeding 90% of the available memory (N <= sqrt(M/8)). For example, for a single machine with 256GB of memory and a block size of 240, N should be 168,960 elements.

Expected results:

Maximum performance for Zen AVX2 is 16 DP flops per two core per core clock cycles. Assuming a relatively cool environment (25 deg. C) and an effective cooling solution, the EPYC CPU should be able to operate at or near the all cores boost frequency even with all cores running AVX2. Increasing the cTDP will increase the headroom for all cores to achieve maximum boost. A 2P EPYC 7601 system (2.7 GHz all cores boost) has a theoretical maximum performance of 64 x 1/2 x 16 x 2.7M or 1382 DP GFLOPS at cTDP = 200W.

The expected efficiency (calculated as actual GFLOPS / theoretical GFLOPS) will be in the 84% - 86% range for a 2P system and 85% - 87% range for a 1P system (both with SMT disabled and 2400 MHz memory). SMT enabled will reduce the efficiency slightly.

Expected results for a 2P system are 1160 – 1190 GFLOPS, and for a 1P system 580 – 600 GFLOPS.

Below are example test results for a 2P system and 1P system on Ubuntu 16.04 with SMT enabled:

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR12R2R4      168960   232     4     4            2785.72              1.154e+03
HPL_pdgesv() start time Mon Sep 11 07:59:28 2017

HPL_pdgesv() end time   Mon Sep 11 08:45:53 2017

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0028913 ...... PASSED
================================================================================
```

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR12R2R4       84480   232     2     4             742.86              5.411e+02
HPL_pdgesv() start time Mon Sep 11 08:09:53 2017

HPL_pdgesv() end time   Mon Sep 11 08:22:16 2017

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0029770 ...... PASSED
================================================================================
```

Multi-threaded (OpenMP) DGEMM parallelization:

When configured with multi-threading enabled, the BLIS library is controlled by the following environment variables, set in the run script. The DGEMM parallelization at each shared L3 cache is configured as follows:

BLIS_JC_NT=1 (No outer loop parallelization)
BLIS_IC_NT=4 (4 2nd level threads – one per core in the shared L3 cache domain)
BLIS_JR_NT=1 (No 4th level threads)
BLIS_IR_NT=1 (No 5th level threads)

For more information, check here: https://github.com/flame/blis/wiki/Multithreading