Neues zur Nutzung + Nutzerumgebung HLRN

Stephen Sachs    sachs@cray.com
Haswell: Advanced Features

- Cores can turbo independently
- Core/uncore can turbo independently
  - core bound apps -> core turbo
  - Memory bound apps -> uncore turbo

<table>
<thead>
<tr>
<th>Processor</th>
<th>Ivybridge</th>
<th>Haswell</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKU</td>
<td>E5-2695 v2</td>
<td>E5-2680 v3</td>
</tr>
<tr>
<td>QPI [GT/s]</td>
<td>8</td>
<td>9.6</td>
</tr>
<tr>
<td>Cache [KB / KB / MB]</td>
<td>32 / 256 / 30</td>
<td>32 / 256 / 30</td>
</tr>
<tr>
<td>Memory BW [GB/s]</td>
<td>59.7</td>
<td>68</td>
</tr>
<tr>
<td>Base freq [GHz]</td>
<td>2.4</td>
<td>2.5</td>
</tr>
<tr>
<td>Max turbo [GHz]</td>
<td>3.2</td>
<td>3.3</td>
</tr>
</tbody>
</table>
Haswell: AVX2 Instructions

Fused Multiply-Add (FMA3)
- Computes \( \pm(a \times b) \pm c \) with only one rounding
  - \( a \times b \) intermediate result is not rounded
  - Increased accuracy compared to MUL & ADD
- Increased performance & accuracy of many FP computations
  - Matrix multiplication, dot product, polynomial evaluation
- 8 single-precision or 4 double-precision FMA
  - FMA is 2 operations – Increases FLOPS potential
  - Haswell: 2xFMA units → Double peak FLOPs

256 Bit SIMD instructions
- Including integer support

Gather instructions
- Load Elements using a vector of indices
- Enables further vectorization capabilities
Haswell: AVX2 Frequency

Amount of turbo frequency achieved depends on:

- Type of workload, number of active cores, estimated current & power consumption, and processor temperature
- Due to workload dependency, separate AVX base & turbo frequencies will be defined for Xeon® processors starting with E5 v3 product family
Haswell: E5-2680 v3 Frequency

Max all turbo (non-AVX)

Expected frequencies for non-AVX workloads

<table>
<thead>
<tr>
<th>Base frequency (non-AVX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected frequencies for workloads with heavy AVX usage</td>
</tr>
</tbody>
</table>

| 2.9 | 2.8 | 2.7 | 2.6 | 2.5 | 2.4 | 2.3 | 2.2 | 2.1 |

AVX Max all turbo

Expected frequencies for most AVX workloads

AVX Base frequency
Haswell: Usage on HLRN

- **Cray compiler (PrgEnv-crty)**
  - when the module “craype-haswell” is loaded, the compiler will try to use AVX2 instructions by default, and FMAs
  - the default floating-point accuracy level (-hfp2)
  - To disable FMA, use -hfp1.

- **Intel compiler (PrgEnv-intel)**
  - will try to use the AVX2 instructions
  - FMA instructions can be disabled with -no-fma.

- **GNU compiler (PrgEnv-gnu, starting at gcc/4.9.0)**
  - add the flags -“march=haswell -mtune=haswell -mavx2 -mfma”
  - With older gcc versions, have “-march=core-avx2 -mtune=core-avx2”
  - FMA instructions can be disabled with -mnofma.

- **Some codes may perform in fact better when not using FMA**
  - due to the Haswell design
  - evaluate the performance of your application after having compiled it with the module “craype-ivybridge” loaded, and omitting the abovementioned FMA-enabling flags.

- **Libraries provided by Cray are already tuned for Haswell when linked to the application.**
Recommended options

- **Cray compiler: -O3 -hfp3**
  - try removing -hfp3 if the precision is not sufficient

- **Intel compiler: -Ofast -unroll-aggressive -fp-model fast=2 -align -fno-alias -fno-falias**
  - try removing the -fp-model fast=2 flag and/or replacing -Ofast with -O3 if having precision issues
  - try removing “-fno-alias -fno-falias” if getting segmentation faults or other code crashes

- **GNU compiler: -Ofast -funroll-all-loops**
  - replace -Ofast with -O3 in case of precision issues
Cray MPICH 7.x.x ABI Compatibility

- Announced at SC’13

- Collaborators of MPICH ABI compatibility
  - MPICH (Starting v3.1, Dec 2013)
  - Intel MPI Library (Starting v5.0, 2014)
  - Cray MPT (Starting v7.0, Jun 2014)
  - IBM MPI (Starting v2.1, Dec 2014)

- Code against cray-mpich/6.x.x has to be recompiled!

- If you need to use old binaries shift the environment by
  - module load cray-mpich-compat/v6
  - module load cray-mpich-compat/v7
Energy Counters

Intel RAPL (Running Average Power Level) and Cray Power Management (PM) performance counters

- /sys/cray/pm_counters/*

- **RAPL:**
  
<table>
<thead>
<tr>
<th>PAPI Event</th>
<th>Name</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>THERMAL_SPEC</td>
<td>Watts</td>
<td>Thermal specification</td>
</tr>
<tr>
<td></td>
<td>MINIMUM_POWER</td>
<td>Watts</td>
<td>Minimum power</td>
</tr>
<tr>
<td></td>
<td>MAXIMUMPOWER</td>
<td>Watts</td>
<td>Maximum power</td>
</tr>
<tr>
<td></td>
<td>MAXIMUM_TIME_WI</td>
<td>Seconds</td>
<td>Maximum time window</td>
</tr>
<tr>
<td></td>
<td>PACKAGE_ENERGY</td>
<td>Joules*</td>
<td>Energy used by chip package</td>
</tr>
<tr>
<td></td>
<td>PP0_ENERGY</td>
<td>Joules*</td>
<td>Energy used by all cores in package</td>
</tr>
<tr>
<td></td>
<td>DRAM_ENERGY</td>
<td>Joules*</td>
<td>Energy used by DRAM in package</td>
</tr>
</tbody>
</table>

- see “man papi”, “man pm” and “man rapl”
Intel RAPL (Running Average Power Level)
How to Use

● Instrument application for sampling or tracing

● Enable RAPL or PM counter collection at runtime:

   export PAT_RT_PERFCTR=RAPL

   Or

   export PAT_RT_PERFCTR=PM

   Or

   export PAT_RT_PERFCTR=event1, event2, ...

● Run program
  ● For RAPL: use **aprun –cc cpu** to bind MPI ranks to specific sockets

● Generate and view profile along with counter data
Resource Utilization Reporting (RUR)

- Produces a log-file containing energy consumption and further information about your run.

- Deactivated by default at HLRN upon request
  - To activate your output create the following file
    $HOME/.rur/user_output_optin

- The default output will be in your home directory under:
  - $HOME/rur.<MOAB batch ID>
    - Set different output file or directory by supplying the path in
      $HOME/.rur/user_output_redirect

- Default output is one file per submitted job.
  - You can change this setting by putting 'apid', 'jobid', or 'single' in the first line of $HOME/.rur/user_output_report_type

- For further information refer to [http://docs.cray.com/](http://docs.cray.com/)
RUR Example Output

HPL:
uid: 2135, apid: 2247507, jobid: , cmdname: ./xhpl_ivb_hybrid:/xhpl_haswell_mpi, plugin: taskstats
[ 'uname', 720708068000, 'stime', 76991328000, 'max_rss', 82396, 'rchar', 4242838045, 'wchar', 6202000, 'exitcode:signal', ['0:0'], 'core', 0
uid: 2135, apid: 2247507, jobid: , cmdname: ./xhpl_ivb_hybrid:/xhpl_haswell_mpi, plugin: energy
[ 'energy_used', 12238954]

SLEEP:
uid: 31418, apid: 1354315, jobid: 1408.sdb, cmdname: /bin/sleep, plugin: taskstats
[ 'utime', 12000, 'stime', 448000, 'max_rss', 572, 'rchar', 124744, 'wchar', 4974, 'exitcode:signal', ['0:0'], 'core', 0
uid: 31418, apid: 1354315, jobid: 1408.sdb, cmdname: /bin/sleep, plugin: energy, energy_used: 2204, nodes: 4

MD5SUM:
uid: 31418, apid: 1354317, jobid: 1408.sdb, cmdname: /usr/bin/md5sum, plugin: taskstats
[ 'utime', 152000, 'stime', 40000, 'max_rss', 612, 'rchar', 73191352, 'wchar', 218, 'exitcode:signal', ['0:0', '0:2'], 'core', 0
uid: 31418, apid: 1354317, jobid: 1408.sdb, cmdname: /usr/bin/md5sum, plugin: energy, energy_used: 875, nodes: 1
RUR Example HPL

- E5-2690 v3 2.60 GHz, 10752 CPUs
- Large run, 80 % of memory on the node used
- 10 minutes wall time