**UG3.1.2: MARCONI - A2 (KNL) details**

*Starting from January, 2020 the activity on Marconi-A2 has been stopped.*

- Production environment
  - Batch
  - Submitting Batch jobs for A2 partition
  - Summary
- Programming environment

---

**hostname:** login.marconi.cineca.it

**start of production**

(A2 - Knights Landing): 04/01/2017

---

- A2: a new section has been added at the start of 2017, equipped with the next-generation of the Intel Xeon Phi product family (Knights Landing), based on a many-core architecture, enabling an overall configuration of about 250 thousand cores with expected additional computational power of approximately 11 Pflop/s.

---

**System A2 (Knights Landing) - out of production since January, 2020**

**Model:** Lenovo Adam Pass

- **Racks:** 50
- **Nodes:** 3,600
- **Processors:** 1 x 68-cores Intel Xeon Phi 7250 CPU (Knights Landing) at 1.40 GHz
- **Cores:** 68 cores/node (272 with HyperThreading), 244,800 cores in total
- **RAM:** 16 GB/node of MCDRAM and 96 GB/node of DDR4

**Peak Performance:** 11 Pflop/s
KNL is the evolution of Knights Corner (KNC), available on GALILEO until January 2018. The main differences between KNC and KNL are:

- KNLs are standalone, self-bootable processors, unlike KNCs, which are treated as accelerators;
- a better power in performance;
- a faster internal network.

Applications compiled for KNL are binary compatible with regular computing nodes.

KNL supports Intel AVX-512 instruction set extensions. The same three login nodes serve the Marconi-Broadwell (Marconi-A1) and the Marconi-KNL (Marconi-A2) partitions and queueing systems.

Storage devices are in common between the two partitions.

### Many Trailblazing Improvements in KNL

<table>
<thead>
<tr>
<th>Improvements</th>
<th>What/Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Boot Processor</td>
<td>No PCIe bottleneck</td>
</tr>
<tr>
<td>Binary Compatibility with Xeon</td>
<td>Runs all legacy software. No recompilation.</td>
</tr>
<tr>
<td>New Core: SLM based</td>
<td>~3x higher ST performance over KNC</td>
</tr>
<tr>
<td>Improved Vector density</td>
<td>3+ TFLOPS (DP) peak per chip</td>
</tr>
<tr>
<td>AVX 512 ISA</td>
<td>New 512-bit Vector ISA with Masks</td>
</tr>
<tr>
<td>Scatter/Gather Engine</td>
<td>Hardware support for gather and scatter</td>
</tr>
<tr>
<td>New memory technology:</td>
<td>Large High Bandwidth Memory (\rightarrow) MCDRAM</td>
</tr>
<tr>
<td>MCDRAM + DDR</td>
<td>Huge bulk memory (\rightarrow) DDR</td>
</tr>
<tr>
<td>New on-die interconnect:</td>
<td>High BW connection between cores and memory</td>
</tr>
<tr>
<td>Mesh</td>
<td></td>
</tr>
</tbody>
</table>
Production environment

Since MARCONI is a general purpose system and it is used by several users at the same time, long production jobs must be submitted using a queuing system. This guarantees that the access to the resources is as fair as possible. Roughly speaking, there are two different modes to use an HPC system: Interactive and Batch. For a general discussion see the section Producti on Environment and Tools.

Batch

Submitting Batch jobs for A2 partition

On MARCONI it is possible to submit jobs requiring different resources by specifying the corresponding partition. If you do not specify the partition, your jobs will run on the bdw_all_serial partition.

If you do not specify the walltime (by means of the #SBATCH --time directive), a default value of 30 minutes will be assumed.

With respect to the previous configuration, submission process results simplified. You do not more need to load the specific "env-knl" module to submit jobs on partitions based on Knights Landing processors. Instead, you simply need to specify the correct partition using the "SBATCH -p directive. Choosing a knl_***_*** partition, you will be sure to work on KNL nodes.

Each KNL node exposes itself to SLURM as having 68 cores (corresponding to the physical cores of the KNL processor). Jobs should request the entire node (hence, #SBATCH --n 68), and the KNL SLURM server is configured so that to assign the KNL nodes in an exclusive way (even if less ncpus are asked). Hyper-threading is enabled, hence you can run up to 272 processes/threads on each assigned node.

The configuration of the Marconi-A2 partition allowed to require different HBM modes (on-package high-bandwidth memory based on the multi-channel dynamic random access memory (MCDRAM) technology) and clustering modes of cache operations:

#SBATCH --constraint=flat/cache

Please refer to the official Intel documentation for a description of the different modes.

For the queues serving the Marconi FUSION partition, please refer to the dedicated document.

The maximum memory which can be requested is 86000MB for cache/flat nodes:

#SBATCH --mem=86000

(the default measurement unity, if not specified, is MB)

For flat nodes the jobs can require the KNL high bandwidth memory (HBM) using Slurm's Generic RESource (GRES) options:

#SBATCH --constraint=flat
#SBATCH --mem=86000

For example, to request a single KNL node in a production queue the following SLURM job script can be used:

#!/bin/bash
#SBATCH -N 1
#SBATCH -A <account_name>
#SBATCH --mem=86000
#SBATCH --constraint=knl_usr_prod
#SBATCH --time 00:05:00
#SBATCH --job-name=KNL_batch_job
#SBATCH --mail-type=ALL
#SBATCH --mail-user=<user_email>

srun ./myexecutable

where <account_name> is the project account user has to specify to pay for the job, and <user_email> is a valid email address user can specify to receive real-time information about his job status changes.

Summary
In the following table you can find all the main features and limits imposed on the queues of the shared A1 and A2 partitions. For Marconi-FUSION dedicated queues please refer to the dedicated document.

<table>
<thead>
<tr>
<th>MARCONI Partition</th>
<th>SLURM partition</th>
<th>QOS</th>
<th># cores per job</th>
<th>max walltime</th>
<th>max running jobs per user</th>
<th>max memory per node (MB)</th>
<th>priority</th>
<th>HBM /clustering mode</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>knl_usr_dbg</td>
<td>no QOS</td>
<td>min = 1 node</td>
<td>00:30:00</td>
<td>5/5</td>
<td>86000 (cache)</td>
<td>40</td>
<td></td>
<td>runs on 144 dedicated nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>max = 2 nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>knl_usr_prod</td>
<td>no QOS</td>
<td>min = 1 node</td>
<td>24:00:00</td>
<td>1000 nodes</td>
<td>86000 (cache)</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>max = 195 nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>knl_qos_bprod</td>
<td></td>
<td>min = 196 nodes</td>
<td>24:00:00</td>
<td>1/1000</td>
<td>86000 (cache)</td>
<td>85</td>
<td></td>
<td>#SBATCH --p knl_usr_prod</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>max = 1024 nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#SBATCH --qos=knl_qos_bprod</td>
</tr>
<tr>
<td></td>
<td>qos_special</td>
<td>&gt;1024</td>
<td>&gt;24:00:00 (max = 195 nodes for user)</td>
<td>00:30:00</td>
<td>860</td>
<td>00 (cache)</td>
<td>40</td>
<td></td>
<td>#SBATCH --qos=qos_special request to <a href="mailto:superc@cineca.it">superc@cineca.it</a></td>
</tr>
</tbody>
</table>

**Programming environment**

**Compiling for KNL and SKL**

Since KNL and SKL nodes are binary compatible with legacy x86 instruction set, any code compiled for BDW nodes will run on these nodes. However, specific compiler option is needed to generate AVX-512 instructions to derive better performance from these nodes.

Version 15.0 and newer of the Intel compilers can generate these instructions if you specify for **KNL nodes** the "-xMIC-AVX512" flag (which generates specific AVX512 instructions, hence the binary will not work on the Broadwell partition) or the -axMIC-AVX512 flag (which generates optimized executables for both AVX2 and AVX512 ISA):

```
module load intel
icc -axMIC-AVX512 -O3 -o executable source.c
icpc -axMIC-AVX512 -O3 -o executable source.cc
ifort -axMIC-AVX512 -O3 -o executable source.f
```

Differently for **SKL nodes** you have to specify the "-xCORE-AVX512" flag in order to generate AVX-512 instructions. When using this option, Intel compilers default to using AVX512 “low”, i.e., a 256-bit version of AVX512 through AVX512-VL (see also compiler documentation for -qopt-zmm-usage=low).

This means that by default, the compiler generates instructions which operate only on 256 bits of the 512 bit registers, but benefit from things like masking and doubled register set size. The most obvious benefit from this approach is that the frequency drop equals only that of AVX2 code.

If you have a code which vectorizes well, you can try experimenting with "-xCORE-AVX512 -qopt-zmm-usage=high" to make the compiler generate AVX-512 code with full 512 bit vectors in use. Doing this will cause the clock frequency of the CPU to drop rather significantly which in turn often causes the code either to run slower or at equal speed compared to for instance AVX2."

There are certain considerations to be taken into account before running legacy codes on KNL and SKL nodes. Primarily, the effective use of vector instructions is critical to achieve good performance on the cores. For guideline on how to get vectorization information and improve code vectorization, refer to

How to Improve Code Vectorization