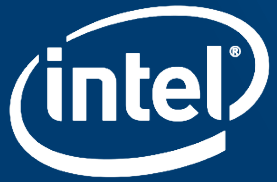


INTEL® OMNI-PATH ARCHITECTURE AND INTEL® MPI

Dr Christopher Dahnken

SSG DRD EMEA HPC

Legal Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Optimization Notice

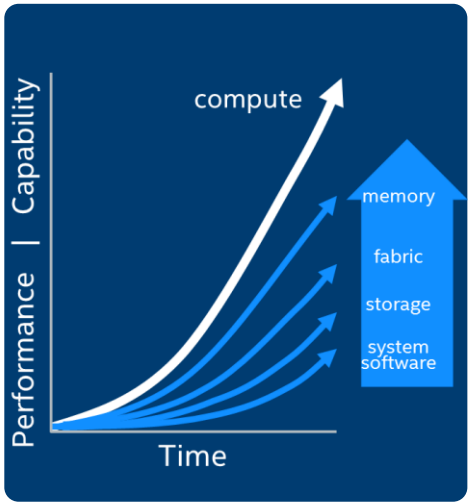
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

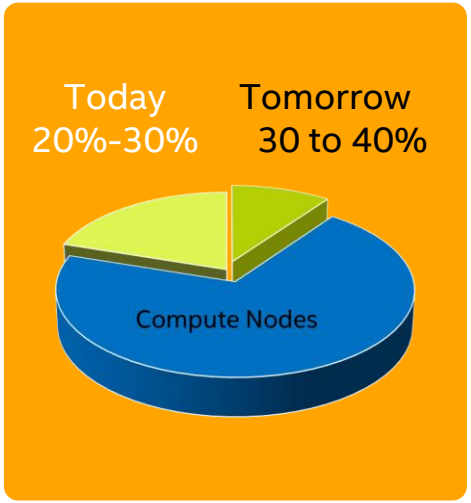
Why Intel® OPA?

Performance



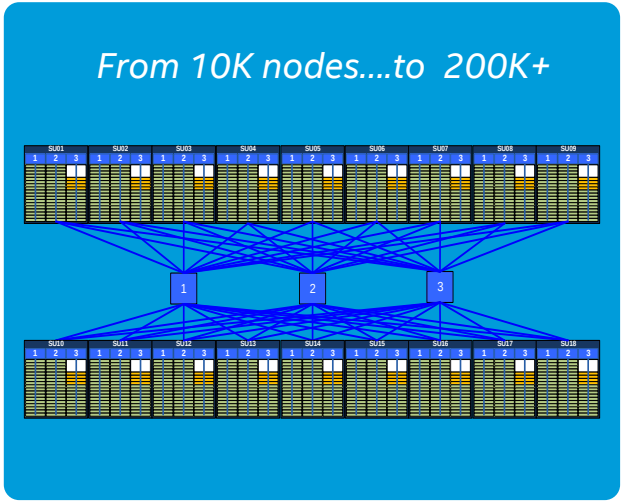
I/O struggling to keep up with CPU innovation

Fabric: Cluster Budget[1]



Fabric an increasing % of HPC hardware costs

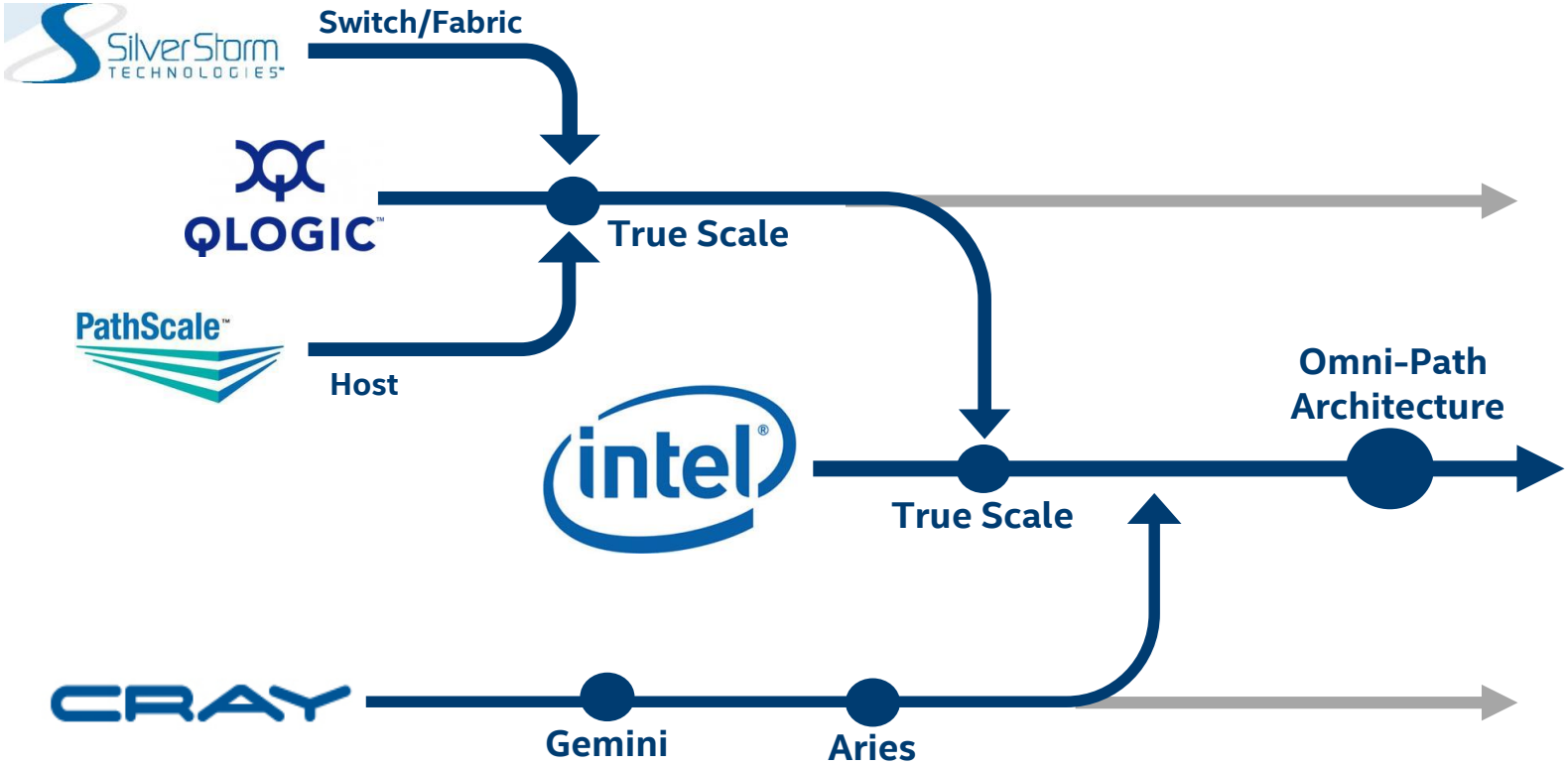
Increasing Scale



Existing solutions reaching limits

1 Source: Internal analysis based on a 256-node to 2048-node clusters configured with Mellanox FDR and EDR InfiniBand products. Mellanox component pricing from www.kernelsoftware.com Prices as of November 3, 2015. Compute node pricing based on Dell PowerEdge R730 server from www.dell.com. Prices as of May 26, 2015. Intel® OPA (x8) utilizes a 2-1 over-subscribed Fabric. Intel® OPA pricing based on estimated reseller pricing using projected Intel MSRP pricing on day of launch.

How we got here…



* Other names and brands may be claimed as the property of others

Intel® Omni-Path Host Fabric Interface

100 Series Single Port[1]

Low Profile PCIe Card

- 2.71”x 6.6“ max. Spec compliant.
- Standard and low profile brackets

Wolf River (WFR-B) HFI ASIC

PCIe Gen3

Single 100 Gb/s Intel® OPA port

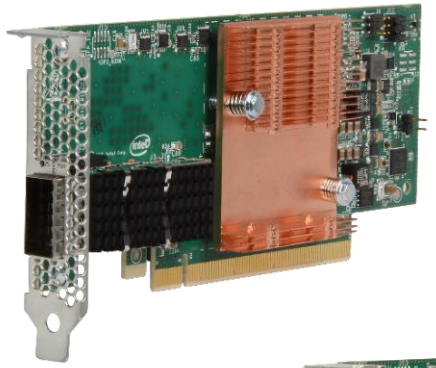
- QSFP28 Form Factor
- Supports multiple optical transceivers
- Single Link status LED (Green)

Power

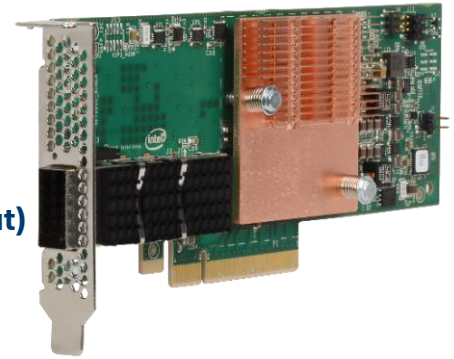
	Copper		Optical (3W QSFP)	
Model	Typical	Maximum	Typical	Maximum
X16 HFI	7.4W	11.7W	10.6W	14.9W
X8 HFI	6.3W	8.3W	9.5W	11.5W

Thermal

- Passive thermal - QSFP Port Heatsink
- Standard 55C, 200lfm environment



x16 HFI
(100Gb Throughput)



x8 HFI
(~58Gb Throughput)
PCIe Limited

[1]Specifications contained in public Product Briefs.

Intel® OPA Software Stack

Performance Scaled Messaging designed for HPC

Carefully selected division of responsibility

- MPI handles higher level capabilities (includes a wide array of MPI and user-facing functions)
- PSM focuses on HPC interconnect (optimized data movement, MPI performance, QoS, dispersive routing, resiliency)

Binary Compatible

- Common base architecture – PSM2 is a superset of PSM
- Applications built and tuned for PSM will just work

Connectionless with minimal on-adapter state

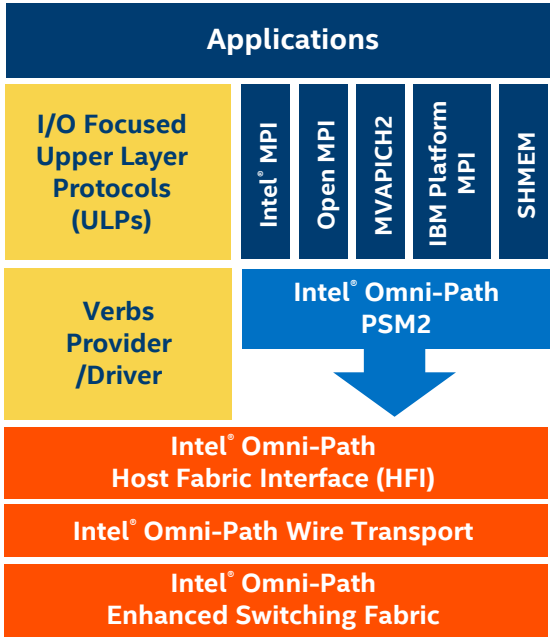
- No cache misses as the fabric scales

High MPI message rate – short message efficiency

Designed to scale with today's servers

- Dense multi-core/multi-socket CPUs
- Fast processors, high memory bandwidth, more cores

Designed for Performance at Extreme Scale



Intel® Omni-Path Software Strategy

- Leverage OpenFabrics Alliance (OFA) interfaces so InfiniBand applications “just work”
- Open source **all** host components in a timely manner
  - Changes pushed up stream in conjunction with Delta Package release
- “Inbox” with future Linux OS releases
  - RHEL, SLES and OFED (standalone distribution from OFA)
- Deliver delta package that layers on top of the OS
  - Updates before they are available inbox
  - Only change what’s necessary. This isn’t a complete distribution!
  - Delta packages will support N and N-1 versions of RHEL and SLES
  - Delta Packages available on Intel® Download Center
- Note: Intel-OFED only layers necessary changes **on top** of existing installations to reduce risk of compatibility issues with other interconnects.

CPU-Fabric Integration

with the Intel® Omni-Path Architecture

KEY VALUE VECTORS

- ✓ Performance
- ✓ Density
- ✓ Cost
- ✓ Power
- ✓ Reliability

Tighter Integration

Multi-chip Package Integration

Intel® OPA HFI Card

Future Generations
Additional integration, improvements, and features

- Future Intel® Xeon® Phi™ processor
- Future Intel® Xeon® processor (14nm)
- Intel® Xeon Phi™ processor x200 family (KNL)
- Intel® Xeon® processor E5-2600 v4 (BDW)
- Intel® Xeon® processor E5-2600 v3 (HSW)

PERFORMANCE

TIME

INTEL® MPI LIBRARY OVERVIEW

Intel® MPI Library Overview

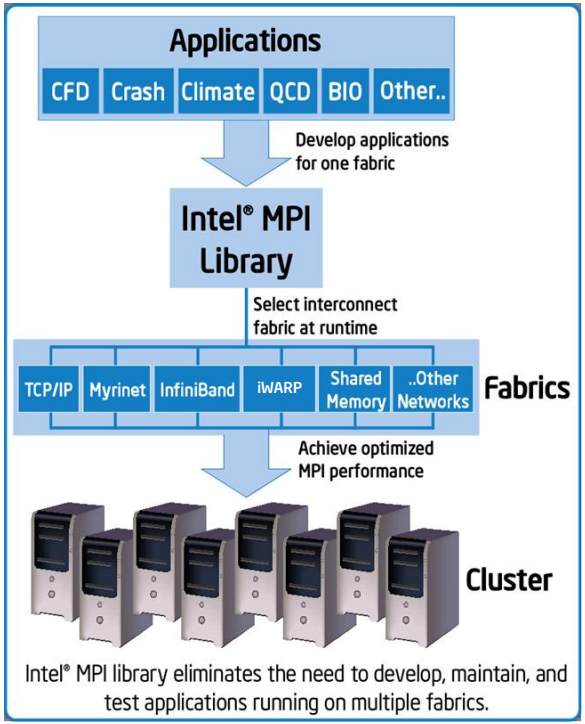
Optimized MPI application performance
- Application-specific tuning
- Automatic tuning

Lower latency and multi-vendor interoperability
- Industry leading latency
- Performance optimized support for the latest OFED capabilities through DAPL 2.x

Faster MPI communication
- Optimized collectives

Sustainable scalability beyond 262K cores
- Native InfiniBand* interface support allows for lower latencies, higher bandwidth, and reduced memory requirements

More robust MPI applications
- Seamless interoperability with Intel® Trace Analyzer and Collector



Intel® MPI library eliminates the need to develop, maintain, and test applications running on multiple fabrics.

Intel® MPI Library Overview

Streamlined product setup

- Install as root, or as standard user
- Environment variable script mpivars.(c)sh sets paths

Compilation scripts to handle details

- One set to use Intel compilers, one set for user-specified compilers

Environment variables for runtime control

- I_MPI_* variables control many factors at runtime
  - Process pinning, collective algorithms, device protocols, and more

Compiling MPI Programs

Compilation Scripts

- Automatically adds necessary links to MPI libraries and passes options to underlying compiler
- Use ***mpiifort*, *mpiicpc*, or *mpiicc*** to force usage of the associated Intel compiler
- Use ***mpif77*, *mpicxx*, *mpicc***, or others to allow user to specify compiler (I_MPI_F77, … or –f77=, -cxx=, …)
  - Useful for makefiles portable between MPI implementations
- All compilers are found via PATH

MPI Launcher

Robust launch command

$mpirun <mpi args> executable <program args>

Options available for:

- Rank distribution and pinning
- Fabric selection and control
- Environment propagation
- Debugging and profiling
- And more

Machine file and number of processes

The scheduler normally provides a an environment variable with the name of the hosts that it provides for you job. (e.g. $PBS_NODEFILE in my case).

You may either use this variable directly

$mpirun –machinefile $PBS_NODEFILE

Or cat this to a local file

$cat $PBS_NODEFILE > hostfile

$mpirun –machinefile hostfile

(please adapt the environment variable to your scheduler)

```
[cdahnken@eln4 PSIWAT]$ less hostfile
ewb133
ewb134
ewb135
ewb136
ewb138
ewb139
ewb311
ewb313
....
```

Process Placement

Default placement puts one rank per core on each node provided in the machine file. If you ask for more ranks than machines, the processes will be placed round-robin

Use **–ppn <n>** or **–perhost <n>** to control processes per node. This places <n> consecutive processes on each host.

$mpirun –n 64 –perhost 8

places 8 processes on 8 nodes (which are provided in the machine file)

If you have complex setting, use a configuration files for precise control for complex jobs

$mpirun –configfile configfile.txt

One can also get very specific

Mpirun also lets you specify per host how many processes you want to have and which environment variables to set, e.g.

$mpirun –genv WORKDIR=/tmp/ –np 8 –host <hostname1> -env OMP_NUM_THREADS=2 -np 4 –host <hostname2> -env OMP_NUM_THREADS=4

Here, **-genv** is a global variable (communicated to all hosts), and **–env** is a local environment variable communicated only to the hostname(s) mentioned after host.

Fabric Support

Fabric selection is through **I_MPI_FABRICS=<intranode>:<internode> | <fabric>**

- shm – Shared memory
- dapl – Direct Access Provider Layer*
- tcp – TCP/IP
- tmi – Tag Matching Interface
- ofa – OFED* verbs
- ofi – OpenFabrics Interfaces

OmniPath Example

```
I_MPI_FALLBACK=0                #if the fabric provided is not found, fail
I_MPI_SHM_LMT=shm               #large messages to use shared memory
I_MPI_FABRICS=shm:tmi           #for intranode coms use share memory,
                                #TMI between nodes
I_MPI_TMI_PROVIDER=psm2         #use the Omnipath driver for the TMI
```

Process pinning

I_MPI_PIN=1 enables process pinning

I_MPI_PIN_DOMAIN will split the logical processors to non-overlapping domains

The process will affinitized to this domain.

For additional OpenMP threading, one needs to take care of this with

KMP_AFFINITY or OMP_PLACES (to be dealt with in the OMP section)

Process pinning – multicore shape

I_MPI_PIN_DOMAIN=**<setting>**

core: Each domain consists of the logical processors that share a particular core.

socket: Each domain consists of the logical processors that share a particular socket

numa: Each domain consists of the logical processors that share a particular NUMA node

See also: **cache1, cache2, cache3** and **cache**

Process pinning – explicit shape

I_MPI_PIN_DOMAIN=**<size>:<layout>**

<size>=

omp: The domain size is equal to the OMP_NUM_THREADS environment variable value.

auto: The domain size is defined by the formula size=#cpu/#proc, where #cpu is the number of logical processors on a node, and #proc is the number of the MPI processes started on a node. This is the default

n: The domain size is defined by a positive decimal number <n>

See also: **cache1, cache2, cache3** and **cache**

Process pinning – explicit shape

I_MPI_PIN_DOMAIN=**<size>:<layout>**

<layout>=

platform: Domain members are ordered according to their BIOS numbering

compact: Domain members are located as close to each other as possible in terms of common resources (cores, caches, sockets, and so on). This is the default value

scatter: Domain members are located as far away from each other as possible in terms of common resources (cores, caches, sockets, and so on)

Lightweight statistics

- Set I_MPI_STATS to a non-zero integer to gather MPI communication statistics (max. 10)
- Change scope with I_MPI_STATS_SCOPE
- Example here:

  Gromacs rank 0 with

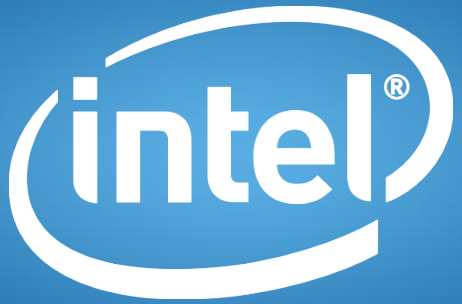
  I_MPI_STATS=3

  I_MPI_STATS_SCOPE=coll

```
Communication Activity by actual args
Collectives
Operation       Context  Algo  Comm size    Message size       Calls  Cost(%)
-----------------------------------------------------------------------------
Allreduce
1                    58     1          4              24           1    0.00
2                    58     1          4               4           8    0.00
3                    58     1          4               8          12    0.03
4                    58     1          4            1376         181    0.04
5                    58     1          4            1344          19    0.01
6                    58     1          4            1216           1    0.00
7                    58     1          4             224           1    0.00
8                     0     5        192               8           2    0.00
9                     0     5        192             968           1    0.00
10                    0     5        192             288           2    0.01
11                    0     5        192             768           2    0.00
Barrier
1                    62     5        160               0           1    0.00
2                     0     5        192               0           1    0.00
Bcast
...
Gather
1                    52     3          5              32          25    0.01
2                    54     3          4              36          25    0.00
3                    56     3          8              28          25    0.01
Reduce
1                    60     1         40              24           1    0.00
2                    60     1         40               4           8    0.00
3                    60     1         40               8          12    0.01
4                    60     1         40            1376         181    0.21
5                    60     1         40            1344          19    0.03
6                    60     1         40            1216           1    0.00
7                    60     1         40             224           1    0.00
Scatter
1                    62     1        160               8           1    0.00
Scatterv
1                    62     1        160          315840           2    0.03
2                    62     1        160           52640           1    0.08
=============================================================================
```

Example – not for the faint hearted

```
mpirun -bootstrap ssh -genv OMP_NUM_THREADS=2 -genv KMP_NUM_THREADS=2 -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[1-40:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[41-80:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[81-120:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[121-160:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[161-200:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=0 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[201-240:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 0-11,24-35 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[1-40:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[41-80:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[81-120:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-5.0.2-
noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[121-160:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-
5.0.2-noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[161-200:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-
5.0.2-noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in : -np 1 -host esg348 -env QE_MIC_DEVICE=1 -env
PHI_KMP_AFFINITY=explicit,granularity=fine,proclist=[201-240:1] -env PHI_OMP_NUM_THREADS=40 taskset -c 12-23,36-47 /home/cdahnken/espresso-
5.0.2-noscalapack/espresso-5.0.2/bin/pw.x -npool 2 < ausurf.in
```


intel®