DKRZ User Training

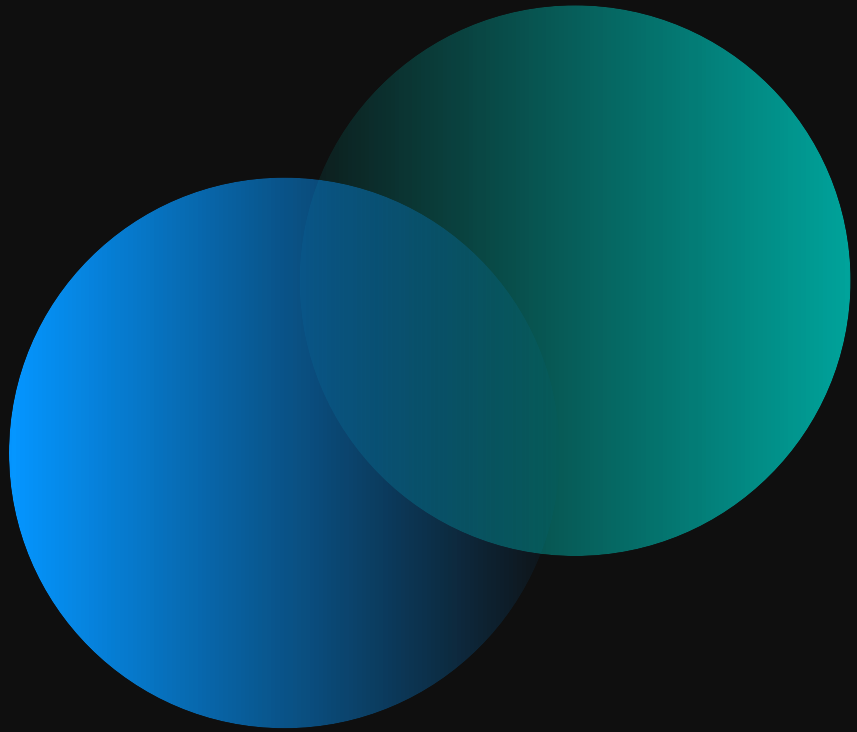
Migration Mistral to Levante

08/06/2022

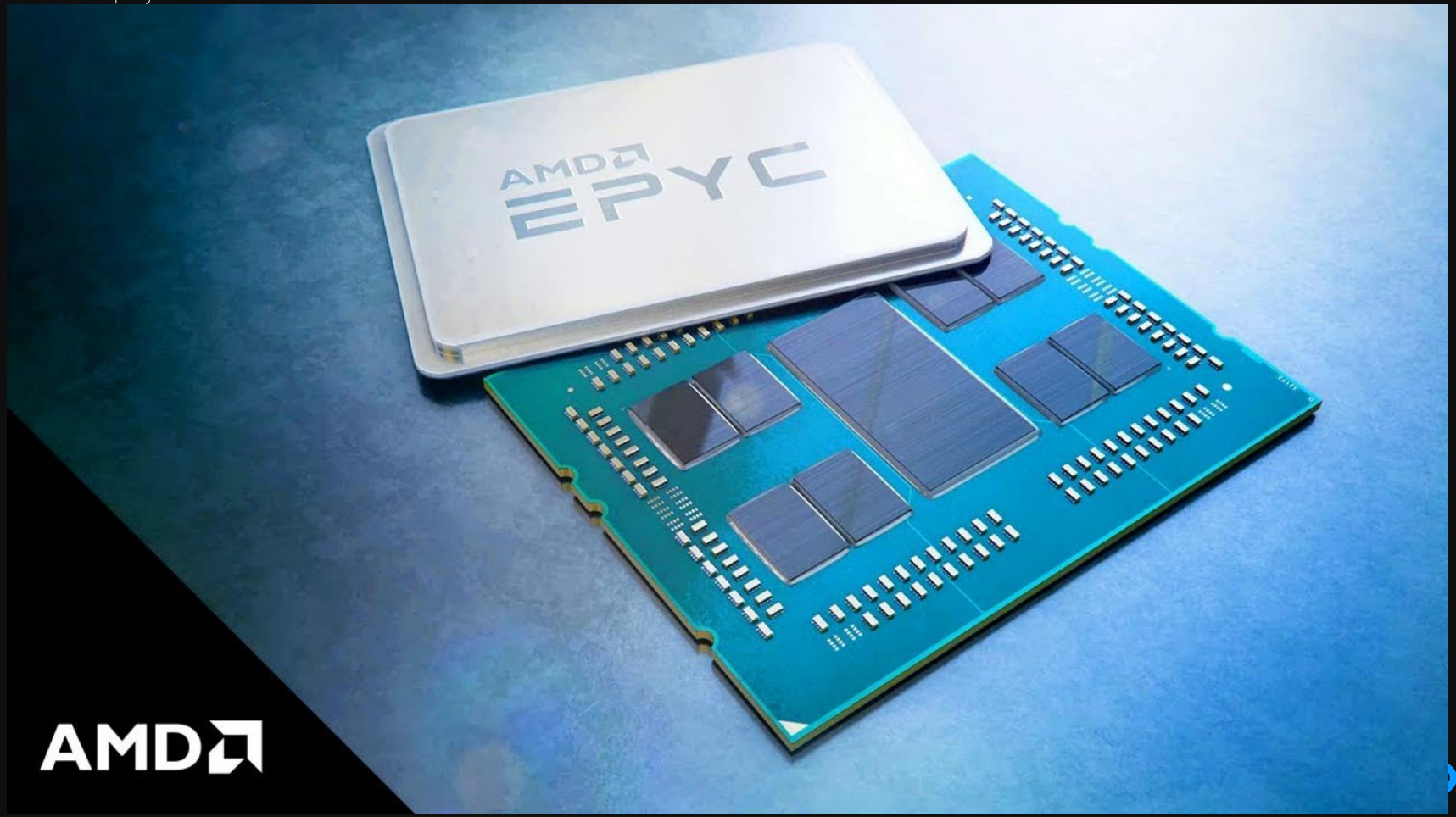
Atos

Content overview

01. AMD 7763 architecture

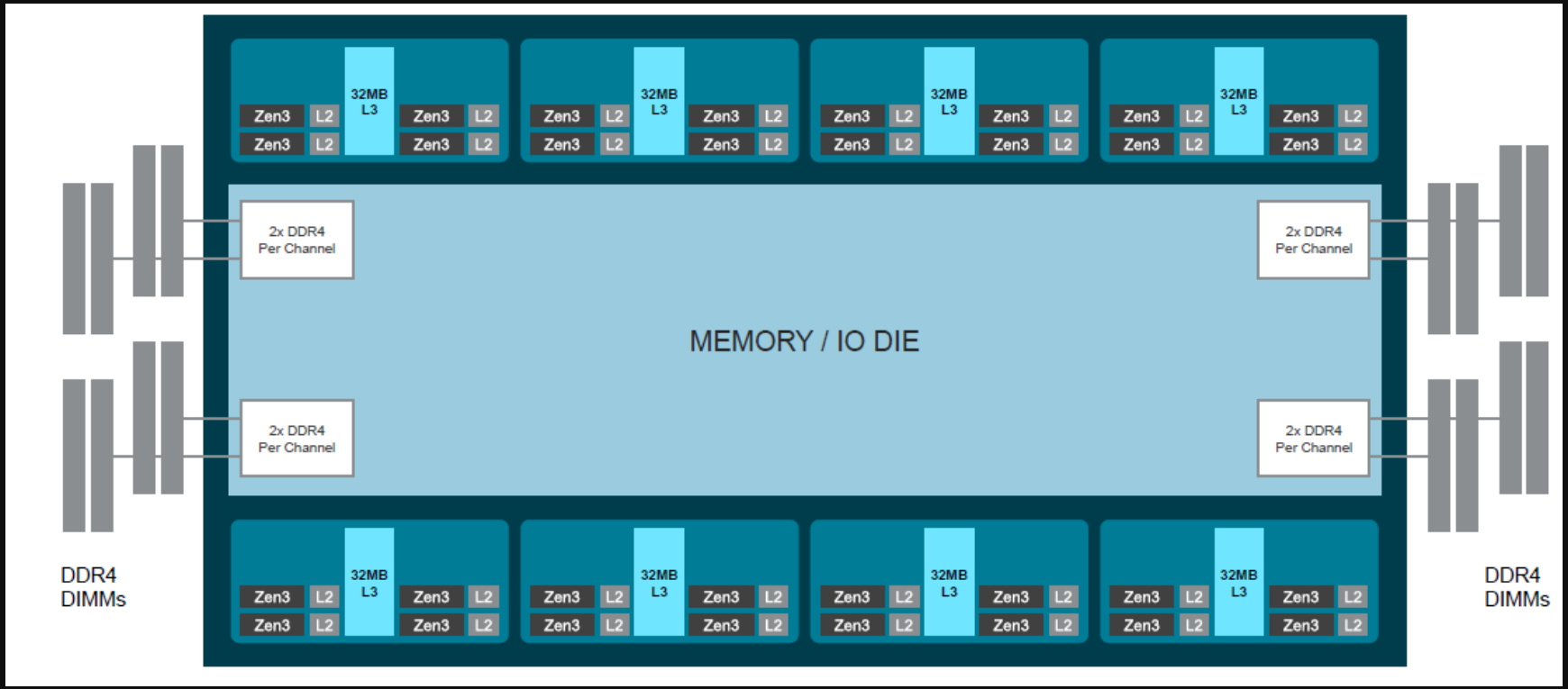


AMD Epyc



AMD Epyc

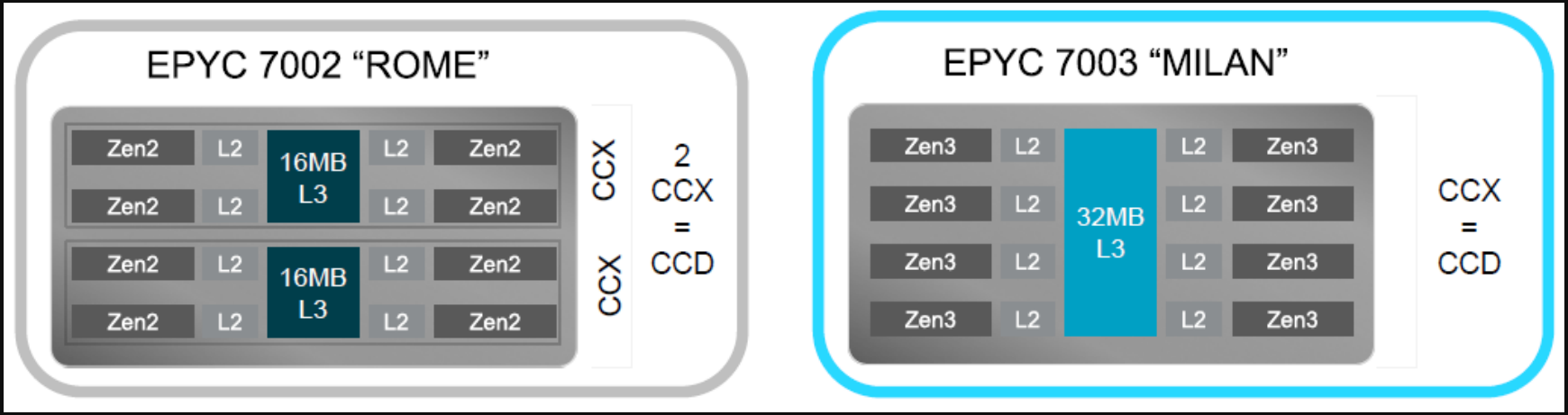
Milan: Multi-Chip (in) Package (MCP)



AMD Milan
Zen3

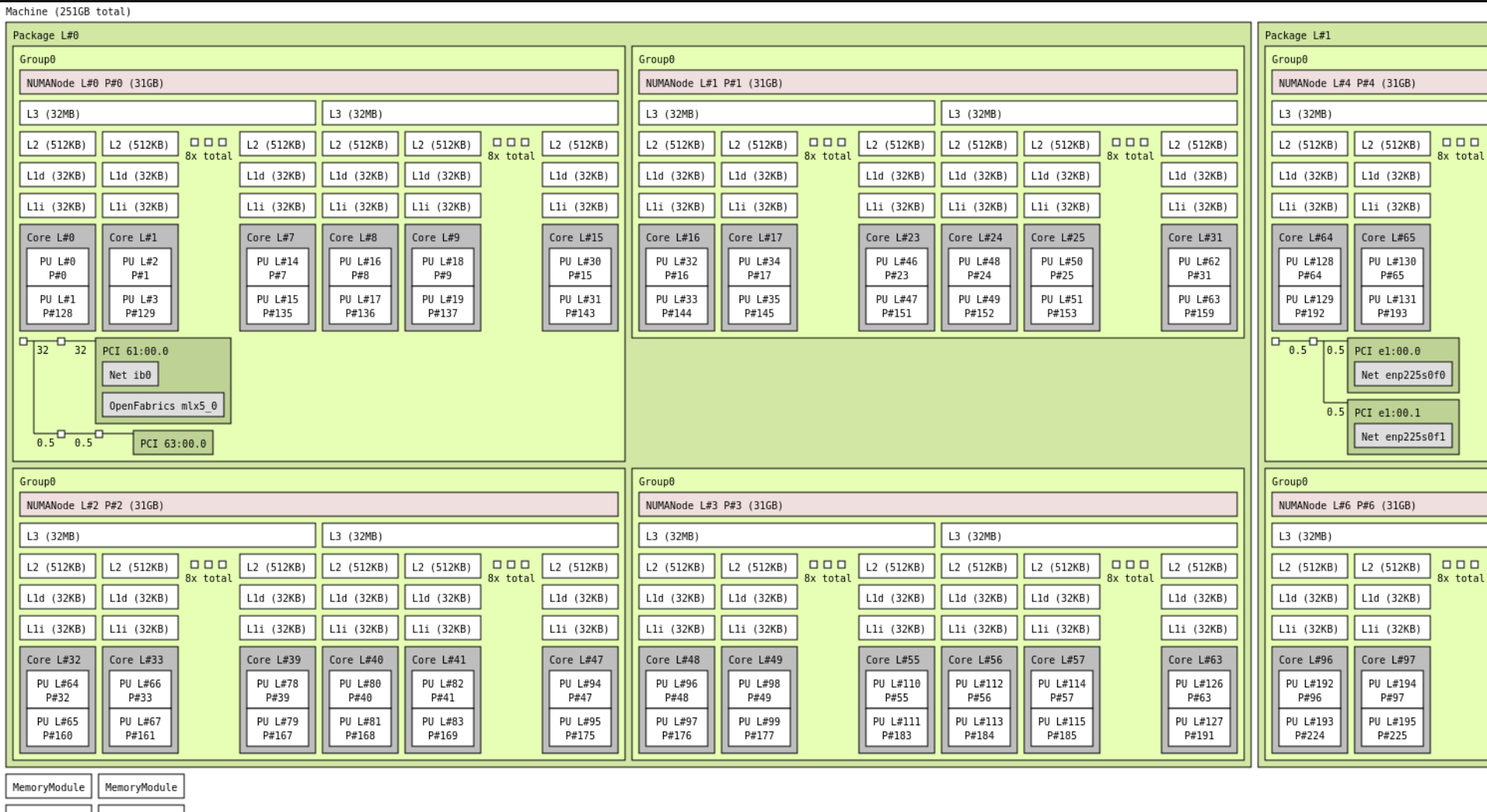
32MB Unified L3 CACHE Benefits

- 2×L3 cache directly accessible per core (if you don't use all core)
- All 32MB can be allocated to a single active core



Rome (Zen2) vs Milan (Zen3)

Atos


Machine (251GB total)
Package L#0
Group0
NUMANode L#0 P#0 (31GB)
L3 (32MB)
L3 (32MB)
L2 (512KB)
L1d (32KB)
L1i (32KB)
8x total
Core L#0
PU L#0 P#0
PU L#1 P#128
Core L#1
PU L#2 P#1
PU L#3 P#129
Core L#7
PU L#14 P#7
PU L#15 P#135
Core L#8
PU L#16 P#8
PU L#17 P#136
Core L#9
PU L#18 P#9
PU L#19 P#137
Core L#15
PU L#30 P#15
PU L#31 P#143
32
32
PCI 61:00.0
Net ib0
OpenFabrics mlx5_0
0.5
0.5
PCI 63:00.0
Group0
NUMANode L#1 P#1 (31GB)
Core L#16
PU L#32 P#16
PU L#33 P#144
Core L#17
PU L#34 P#17
PU L#35 P#145
Core L#23
PU L#46 P#23
PU L#47 P#151
Core L#24
PU L#48 P#24
PU L#49 P#152
Core L#25
PU L#50 P#25
PU L#51 P#153
Core L#31
PU L#62 P#31
PU L#63 P#159
Package L#1
Group0
NUMANode L#4 P#4 (31GB)
Core L#64
PU L#128 P#64
PU L#129 P#192
Core L#65
PU L#130 P#65
PU L#131 P#193
0.5
0.5
PCI e1:00.0
Net enp225s0f0
0.5
PCI e1:00.1
Net enp225s0f1
Group0
NUMANode L#2 P#2 (31GB)
Core L#32
PU L#64 P#32
PU L#65 P#160
Core L#33
PU L#66 P#33
PU L#67 P#161
Core L#39
PU L#78 P#39
PU L#79 P#167
Core L#40
PU L#80 P#40
PU L#81 P#168
Core L#41
PU L#82 P#41
PU L#83 P#169
Core L#47
PU L#94 P#47
PU L#95 P#175
Group0
NUMANode L#3 P#3 (31GB)
Core L#48
PU L#96 P#48
PU L#97 P#176
Core L#49
PU L#98 P#49
PU L#99 P#177
Core L#55
PU L#110 P#55
PU L#111 P#183
Core L#56
PU L#112 P#56
PU L#113 P#184
Core L#57
PU L#114 P#57
PU L#115 P#185
Core L#63
PU L#126 P#63
PU L#127 P#191
Group0
NUMANode L#6 P#6 (31GB)
Core L#96
PU L#192 P#96
PU L#193 P#224
Core L#97
PU L#194 P#97
PU L#195 P#225
MemoryModule
MemoryModule

NUMA topology NPS4

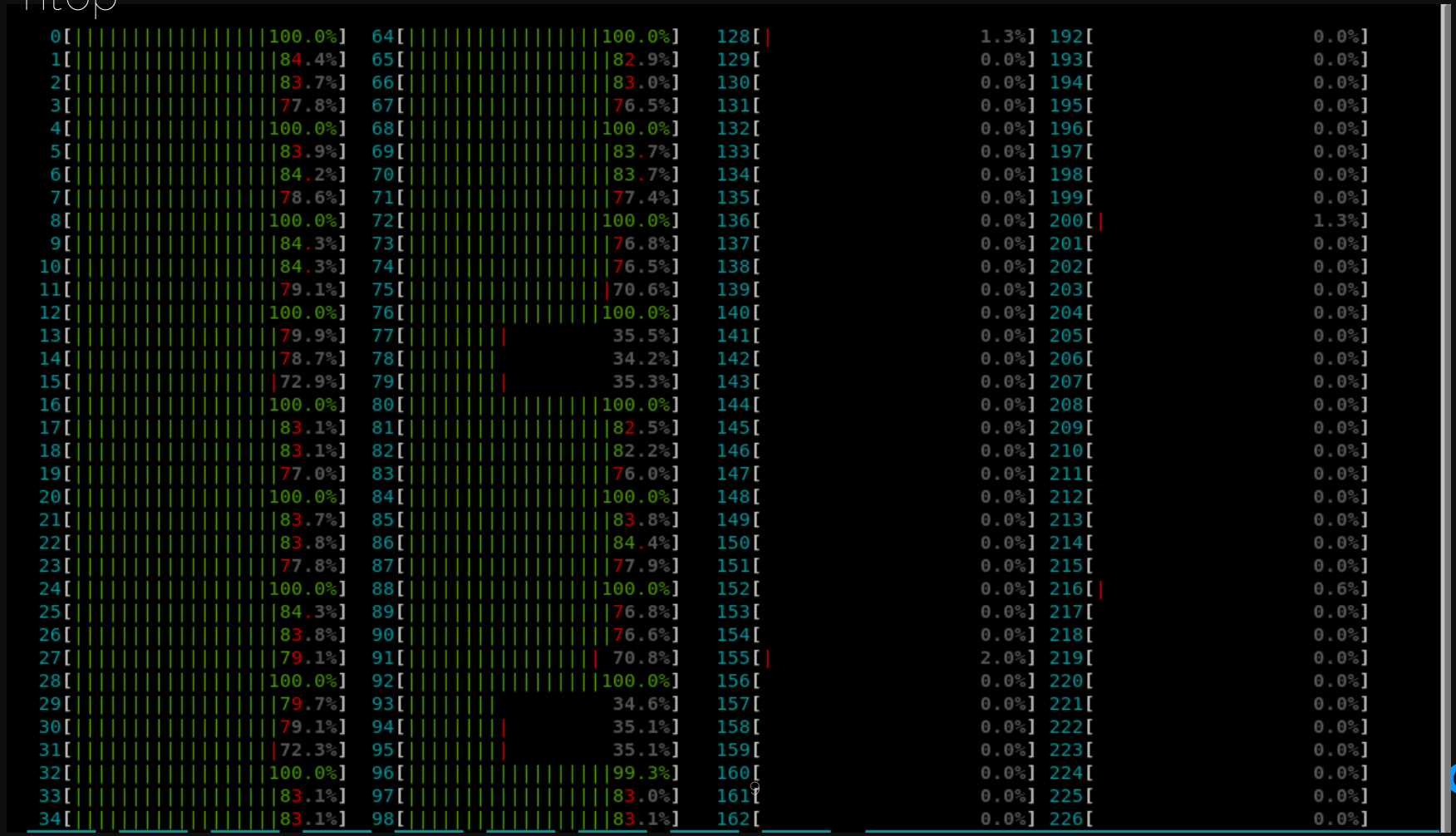
- Local node = 10
- Near-Hop Distance (nodes on same socket) = 12
- Far-Hop Distance (nodes on different socket) = 32

- `numactl -H`

```
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  12  12  12  32  32  32  32
  1:  12  10  12  12  32  32  32  32
  2:  12  12  10  12  32  32  32  32
  3:  12  12  12  10  32  32  32  32
  4:  32  32  32  32  10  12  12  12
  5:  32  32  32  32  12  10  12  12
  6:  32  32  32  32  12  12  10  12
  7:  32  32  32  32  12  12  12  10
```


Atos

htop


0[||||||||||||||||100.0%] 64[||||||||||||||||100.0%] 128[| 1.3%] 192[0.0%]
1[||||||||||||||||84.4%] 65[||||||||||||||||82.9%] 129[0.0%] 193[0.0%]
2[||||||||||||||||83.7%] 66[||||||||||||||||83.0%] 130[0.0%] 194[0.0%]
3[||||||||||||||||77.8%] 67[||||||||||||||||76.5%] 131[0.0%] 195[0.0%]
4[||||||||||||||||100.0%] 68[||||||||||||||||100.0%] 132[0.0%] 196[0.0%]
5[||||||||||||||||83.9%] 69[||||||||||||||||83.7%] 133[0.0%] 197[0.0%]
6[||||||||||||||||84.2%] 70[||||||||||||||||83.7%] 134[0.0%] 198[0.0%]
7[||||||||||||||||78.6%] 71[||||||||||||||||77.4%] 135[0.0%] 199[0.0%]
8[||||||||||||||||100.0%] 72[||||||||||||||||100.0%] 136[0.0%] 200[| 1.3%]
9[||||||||||||||||84.3%] 73[||||||||||||||||76.8%] 137[0.0%] 201[0.0%]
10[||||||||||||||||84.3%] 74[||||||||||||||||76.5%] 138[0.0%] 202[0.0%]
11[||||||||||||||||79.1%] 75[||||||||||||||||70.6%] 139[0.0%] 203[0.0%]
12[||||||||||||||||100.0%] 76[||||||||||||||||100.0%] 140[0.0%] 204[0.0%]
13[||||||||||||||||79.9%] 77[||||||||| 35.5%] 141[0.0%] 205[0.0%]
14[||||||||||||||||78.7%] 78[||||||||| 34.2%] 142[0.0%] 206[0.0%]
15[||||||||||||||||72.9%] 79[||||||||| 35.3%] 143[0.0%] 207[0.0%]
16[||||||||||||||||100.0%] 80[||||||||||||||||100.0%] 144[0.0%] 208[0.0%]
17[||||||||||||||||83.1%] 81[||||||||||||||||82.5%] 145[0.0%] 209[0.0%]
18[||||||||||||||||83.1%] 82[||||||||||||||||82.2%] 146[0.0%] 210[0.0%]
19[||||||||||||||||77.0%] 83[||||||||||||||||76.0%] 147[0.0%] 211[0.0%]
20[||||||||||||||||100.0%] 84[||||||||||||||||100.0%] 148[0.0%] 212[0.0%]
21[||||||||||||||||83.7%] 85[||||||||||||||||83.8%] 149[0.0%] 213[0.0%]
22[||||||||||||||||83.8%] 86[||||||||||||||||84.4%] 150[0.0%] 214[0.0%]
23[||||||||||||||||77.8%] 87[||||||||||||||||77.9%] 151[0.0%] 215[0.0%]
24[||||||||||||||||100.0%] 88[||||||||||||||||100.0%] 152[0.0%] 216[| 0.6%]
25[||||||||||||||||84.3%] 89[||||||||||||||||76.8%] 153[0.0%] 217[0.0%]
26[||||||||||||||||83.8%] 90[||||||||||||||||76.6%] 154[0.0%] 218[0.0%]
27[||||||||||||||||79.1%] 91[||||||||||||||| 70.8%] 155[| 2.0%] 219[0.0%]
28[||||||||||||||||100.0%] 92[||||||||||||||||100.0%] 156[0.0%] 220[0.0%]
29[||||||||||||||||79.7%] 93[||||||||| 34.6%] 157[0.0%] 221[0.0%]
30[||||||||||||||||79.1%] 94[||||||||| 35.1%] 158[0.0%] 222[0.0%]
31[||||||||||||||||72.3%] 95[||||||||| 35.1%] 159[0.0%] 223[0.0%]
32[||||||||||||||||100.0%] 96[||||||||||||||||99.3%] 160[0.0%] 224[0.0%]
33[||||||||||||||||83.1%] 97[||||||||||||||||83.0%] 161[0.0%] 225[0.0%]
34[||||||||||||||||83.1%] 98[||||||||||||||||83.1%] 162[0.0%] 226[0.0%]

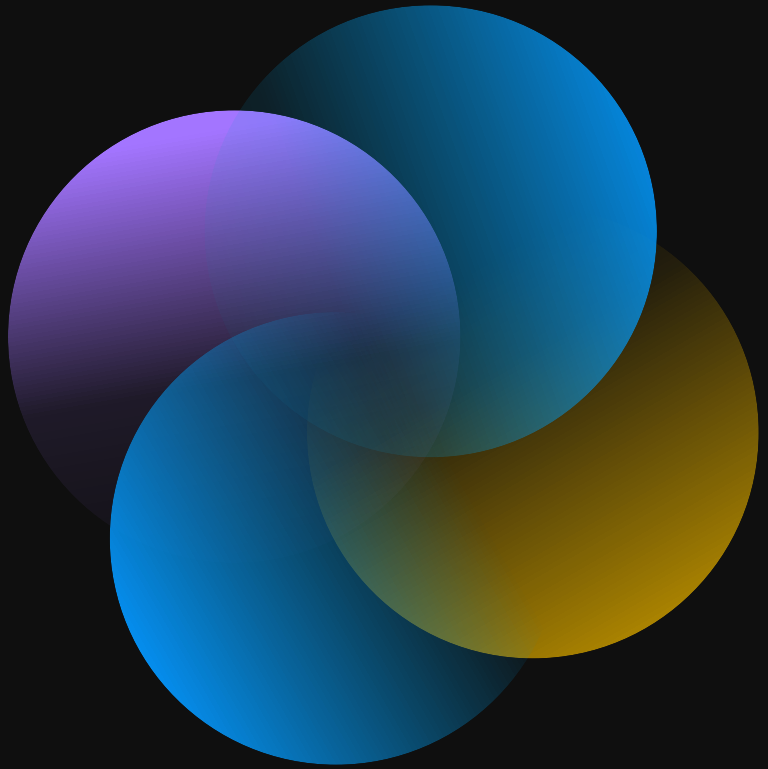
CPU Frequency, Bandwidths, SMT

- Nominal: 2.45 GHz
- Turbo frequency: 3.5 GHz

- Memory bandwidth: 350 GB/s
- Bandwidth between CPUs: 188 GB/s

- SMT
  - hyperthreads are available, logical CPUs 128-255
  - The use of hyperthreads can give an improvement, but it depends
  - Proof of the pudding is in the eating

03. Cluster



Topography: scontrol show topo & sbatch --switches

1 hop:

SwitchName=isw40600 Level=0 LinkSpeed=1 Nodes=l406[00-47]

SwitchName=isw40601 Level=0 LinkSpeed=1 Nodes=l406[48-83,87-95]

3 hops:

SwitchName=isw10005 Level=1 LinkSpeed=1 Nodes=l[10000-10083,10090-10095,10100-10183,10190-10195,10200-10283,10290-10295],levante[0-1]
Switches=isw10000,isw10001,isw10100,isw10101,isw10200,isw10201

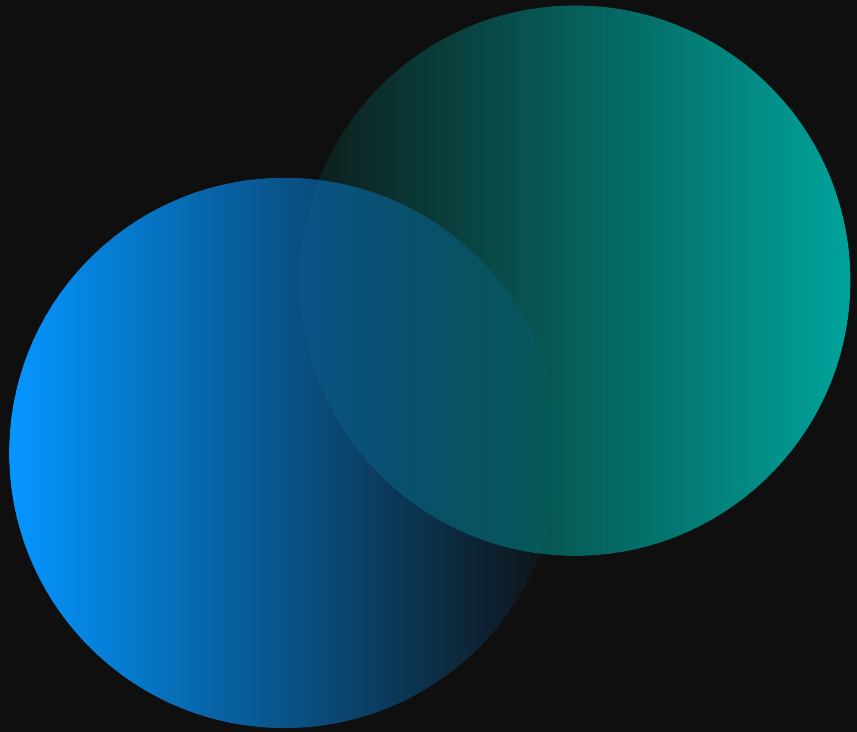
5 hops:

SwitchName=isw10008 Level=2 LinkSpeed=1 Nodes=l[10000-10083,…

- `sbatch --switches=1@1:00:00   # wait up to 1 hour for a job over max 1 switch`
- `sbatch --constraint=="[cell01|cell02|cell03|cell04|cell05|cell06|cell07|cell08|cell09|cell10|cell11]"`
  `-> restrict to 3 hops (270 nodes at max)`

Atos

03. Compiler



Atos

Important! Compiler flags

- AMD Milan (aka Zen3) supports AVX2, but it does NOT support AVX-512!

- Use the `-march` flag to specify the CPU:
  - `-march=core-avx2` flag for the classic Intel compiler (icc, ifort)
  - `-march=znver3` flag for the next-gen Intel compiler, AOCC, LLVM, GCC (core-avx2 would also work)
  - `-march=zen3` flag for the Nvidia HPC SDK compiler (nvcc, nvc++, nvfortran)

- DON'T use the `-xCORE-AVX2` flag for the Intel compiler! (It requires an Intel CPU...)
- DON'T use the `-xHOST` flag for the Intel compiler!

Atos

Recommended application compiler flags (Intel classic compiler)

- ICON: `-march=core-avx2 -mtune=core-avx2 -fma -g -gdwarf-4 -ip -pc64 -fpp -assume realloc_lhs -align array64byte -qopenmp -O2 -fp-model source -qoverride-limits`
- FESOM2: `-O3 -fp-model fast=2 -march=core-avx2 -ip -ipo -assume buffered_stdout -r8 -implicitnone`
- EMAC/MeSSy: `-O2 -march=core-avx2 -fp-model=source -static-intel -mkl=sequential`
- MITRAS: `-fopenmp -O2 -r8 -ftz -no-fma -align all -warn unused \ -fp-stack-check -traceback -list-line-len=120 \ -assume buffered_io -assume norealloc_lhs`

Atos

Overview of different compilers

Currently installed on Levante

	C/C++	Fortran	Version
Intel*	Yes	Yes	2022.0.1
GNU	Yes	Yes	11.2.0
AMD Optimizing C/C++ and Fortran Compilers	Yes	Yes	3.2.0
Nvidia HPC	Yes	Yes	21.11

*Focus on Intel compiler

Atos

Intel Compiler

General overview – Intel

- OneAPI toolkits introduced in 2020
- Classical compiler C/C++ Fortran (icc/icpc/ifort)
- New Data Parallel C++ (DPC++) compiler
  - DPC++ is C++ with SYCL
  - Implementation based on Clang + LLVM + Compute runtimes
  - Open Community project
  - Compatible with C, Fortran, OpenMP, MPI, Python
  - High-level language designed to target heterogenous architectures and take advantage of data parallelism
  - Reuse code across CPUs and accelerators while permitting custom tuning
- Added new compilers based on LLVM framework
  - Drivers: icx, icpx, ifx, and dpcpp
- Compatible with GNU products

Atos

Intel Compiler
Fortran Compiler

- IFX Fortran language
  - Everything implemented up to Fortran 2008 except Coarrays
  - Some features of Fortran 2018 implemented
- IFX OpenMP support
  - CPU OpenMP 4.5 clauses mostly work, except `nowait`, `depend`, `hint` clauses, complex type for `reduction` and `depend` clause modifiers.
  - OFFLOAD: some limited offload – simple arrays, simple OpenMP `TARGET MAP` directives
  - Much work needed in 2022 and beyond to implement OpenMP offload (only on Intel GPU)

  - Check https://www.intel.com/content/www/us/en/developer/articles/technical/fortran-language-and-openmp-features-in-ifx.html for up-to-date information

Atos

Intel Compiler
How to port

- For all Intel compilers
  - `-g -debug=full` all generates complete debugging information
  - `-fmath-errno` Tells the compiler to assume that the program tests errno after calls to math library functions
    - Default: `-fno-math-errno`
    - Floating-point code that does not rely on errno to detect errors can safely use this option to improve performance.
  - `-fp-model <keyword>` Controls the semantics of floating-point calculations
    - Default: `-fp-model fast=1` The compiler uses more aggressive optimizations on floating-point calculations.
    - `-fp-model` precise Disables optimizations that are not value-safe on floating-point data and rounds intermediate results to source-defined precision.
  - `-no-vec -no-simd` To disable all compiler vectorization
  - `-dryrun` Tells the driver that tool commands should be shown but not executed.
- Only with classical compilers
  - -mp1 Improves floating-point precision and consistency (this option disables fewer optimizations and has less impact on performance than option fltconsistency)

Atos

Intel Compiler
How to port (Fortran)

- `-O0` Disables all optimizations
- `-warn all` Enables all warning messages.
- `-warn errors` Force warnings to be reported as errors.
- `-check all` Enables all check options (runtime)
- `-init=snan` Determines whether the compiler initializes to signaling NaN all uninitialized variables of intrinsic type REAL or COMPLEX that are
- `-init=huge` or `minus_huge`
  - whether the compiler initializes to the largest representable positive or negative real value all uninitialized variables of intrinsic type REAL or COMPLEX that are saved, local, automatic, or allocated variables
  - whether the compiler initializes to the largest representable positive or negative integer value all uninitialized variables of intrinsic type INTEGER that are saved, local, automatic, or allocated variables saved, local, automatic, or allocated variables. (runtime)
- `-fpe0` This setting provides full IEEE support (runtime).
  - `-fpe3` All floating-point exceptions are disabled (default)
- `-fltconsistency` Enables improved floating-point consistency (Default: OFF)

Atos

Intel Compiler
How to port (C/C++)

- C/C++, icc/icpc/icx
  - `-O0` Disables all optimizations
  - `-Wall` Display remarks, warnings, and errors
  - `-Werror` Force warnings to be reported as errors.
  - `-ftrapuv` This option initializes stack local variables to an unusual value

- C/C++, icc/icpc only
  - `-fp-trap=all` This setting provides full IEEE support (runtime)
    - `-fp-trap=none` No traps are enabled when a program starts.

Atos

03. OpenMP & MPI Runtime environment

OpenMP
Recommendations

use Intel OpenMP library (even if compiling with other compilers)
  e.g. using "-liomp5 -L<path-to-library>" when linking...

for hybrid (MPI/OpenMP) applications:
OMP_NUM_THREADS=4 or 8 (same NUMA-domain, shared L3-cache)
OMP_STACKSIZE=128M (adjust stack-size limit for OMP-threads to your needs; default = 32M)

IOMP settings:

- KMP_LIBRARY=turnaround (keep idle threads spinning)
  - in some cases of MPI_THREAD_MULTIPLE applications, maybe KMP_LIBRARY=throughput (default) is better
- KMP_AFFINITY=granularity=fine,scatter (pin threads, no migration between SMT-pairs)

OpenMPI
Essential / Recommended Environment settings

OMPI settings

- OMPI_MCA_pml=ucx
- OMPI_MCA_btl=self (shortcut; use UCX to communicate to other tasks)
- OMPI_MCA_osc=ucx (pt2pt is more robust against slight "code-mishaps")
- OMPI_MCA_coll=^hcoll,ml (disabling HCOLL is beneficial for most applications tested)
- OMPI_MCA_io=romio321

UCX settings

- UCX_TLS=self,sm,dc_mlx5,dc_x small jobs (up to 128 nodes may use "self,sm,rc_mlx5,rc_x" instead
- UCX_UNIFIED_MODE=y (disable heterogenous support)
- UCX_IB_SL=auto plus UCX_IB_AR_ENABLE=yes (use job-aware adaptive routing)

for a full list of runtime parameters try these:

```
ompi_info --param all all -l 9
ucx_info -fa
```


Atos

Intel MPI

Essential / Recommended Environment settings

essential:

- I_MPI_PMI=pmi2
- I_MPI_PMI_LIBRARY=/usr/lib64/libpmi2.so
- srun --mpi=pmi2 <other srun args>

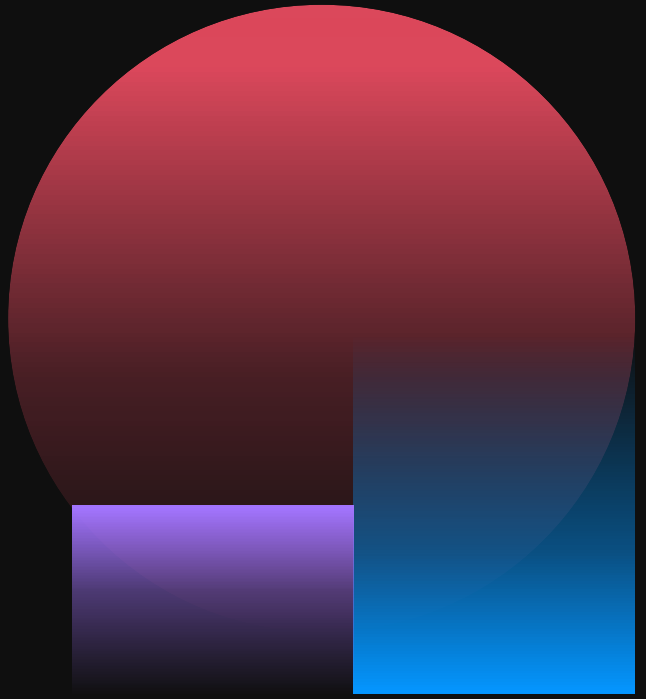
recommended:

- I_MPI_FABRICS=shm:ofi
- I_MPI_OFI_PROVIDER=mlx and/or FI_PROVIDER=mlx
- FI_MLX_TLS=self,shm,dc_x,dc
- UCX_... environment variables work, too, if using mlx-provider (see OpenMPI)

(impi_info)

Atos

03. SLURM examples



pure OpenMP / serial job
SLURM batch script

```
# https://slurm.schedmd.com/sbatch.html

#!/bin/bash
#SBATCH --job-name=<name>
#SBATCH --account=<SLURM-account> # usually your project name
#SBATCH --partition=compute
#SBATCH --chdir=<workdir> # script execution starts in this pwd (must exist)
#SBATCH --nodes=<nodes to use for this job># here "1" obviously
#SBATCH --output=<your-log-file> --error =<your-log-file>  # directory must exist
#SBATCH --time=<estimated runtime of job> # (max. 08:00:00)
# the above SBATCH-head is common for all "scripts"
ulimit -s 204800 # limit stack size (adjust to your needs)
ulimit -c 0 # disable generation of core-files (set to ‚unlimited' for debugging purposes only)

... stuff ...
... more stuff ...
... and yet more stuff ...
```


Atos

pure MPI
SLURM batch script

```
# SBATCH header

... stuff ... (e.g. OMPI environment)

#launch application
# https://slurm.schedmd.com/srun.html

# use all cores of a node
srun -l --hint=nomultithread --cpu-bind=v --tasks-per-node=128 --distribution=block:cyclic -- application



#use only a fraction of cores (but utilize all memory-bandwidth)
srun -l --hint=nomultithread --cpu-bind=v --tasks-per-node=32 --cpus-per-task=4 --distribution=block:cyclic --
application

#in case of IntelMPI provide "--mpi=pmi2" on the srun command line
```


Atos

MPMD (pure MPI)
SLURM batch script

```
# SBATCH header

... stuff ... (e.g. OMPI environment)

#launch application

cat  > ./mpmd.conf << EOF
#
0,42,23 applicationI
45-255 applicationII
* applicationIII
#
EOF
# applications I-III may also actually the very same application but with different command-line args or wrapper scripts around

srun -l --hint=nomultithread --cpu-bind=v --tasks-per-node=32 --cpus-per-task=4 --distribution=block:cyclic:block --multi-prog mpmd.conf
```


Atos

hybrid MPI/OpenMP
SLURM batch script

```
# SBATCH header

... stuff ... (e.g. OMPI environment)

#launch application

# use all cores of a node
export OMP_NUM_THREADS=4
srun -l --hint=nomultithread --cpu-bind=v --tasks-per-node=32 \
                --cpus-per-task=4 --distribution=block:cyclic:block -- application

#use only a fraction of cores (but utilize all memory-bandwidth)
export OMP_NUM_THREADS=4
srun -l --hint=nomultithread --cpu-bind=v --tasks-per-node=16 \
                --cpus-per-task=8 --distribution=block:cyclic:block -- application
```


Atos

hetjob (heterogeneous step)
SLURM batch script

alike "simple" MPMD, but allows to provide different resource specifications to different ranks within an otherwise homogeneous allocation

e.g. on the first 20 nodes 32 tasks of applicationI with 4 cores each are launched
while the remaining nodes are filled with 8 tasks of applicationII per node (16 cores each)

```
# SBATCH header

... stuff ... (e.g. OMPI environment)

#launch application

srun \
    -N 20 -l --hint=nomultithread --cpu-bind=v --tasks-per-node=32 -c 4 -m block:cyclic:block -- applicationI  : \
     -l --hint=nomultithread --cpu-bind=v --tasks-per-node=8 -c 16 -m block:cyclic:block -- applicationII
```


Atos

hetjob (heterogeneous allocation)

SLURM batch script (Header part)

https://slurm.schedmd.com/heterogeneous_jobs.html

```
#!/bin/bash
#SBATCH --job-name=<name>
#SBATCH --account=<SLURM-account> # usually your project name
#SBATCH --chdir=<workdir> # script execution starts in this pwd (must exist)
#SBATCH --output=<your-log-file> --error =<your-log-file>  # directory must exist
#SBATCH --time=<estimated runtime of job> # (max. 08:00:00)
#SBATCH --partition=compute
#SBATCH --nodes=2 # or other sbatch-args
#SBATCH hetjob # seperates het-group specifications
#SBATCH --partition=compute # could be a different one
#SBATCH --nodes=2 # -> so total request is for an allocation of 4 nodes
#SBATCH --constraint=512G # or more/other sbatch-args
#SBATCH hetjob
#SBATCH --nodes=3 --account=<SLURM-account> # or more
# defititions of more het-groups could be added here ...
```


Atos

hetjob (heterogeneous allocation)
SLURM batch script (Body part)

```
# execute on het-groups 0 and 1 (only)
srun --het-group=0-1 <other srun-args> applicationI # execute on het-groups 0 and 1

# execute on het-groups 0 and 1 but use separate srun aruments and applications
srun --het-group=0 <other srun-args grp1> applicationII : --het-group=1  <other srun-args grp2> applicationIII
```

- limitation: het-groups consist of sets of nodes

  so no intermixing of e.g. 4 tasks using 16 cores each plus 8 tasks using 8 cores each on one node

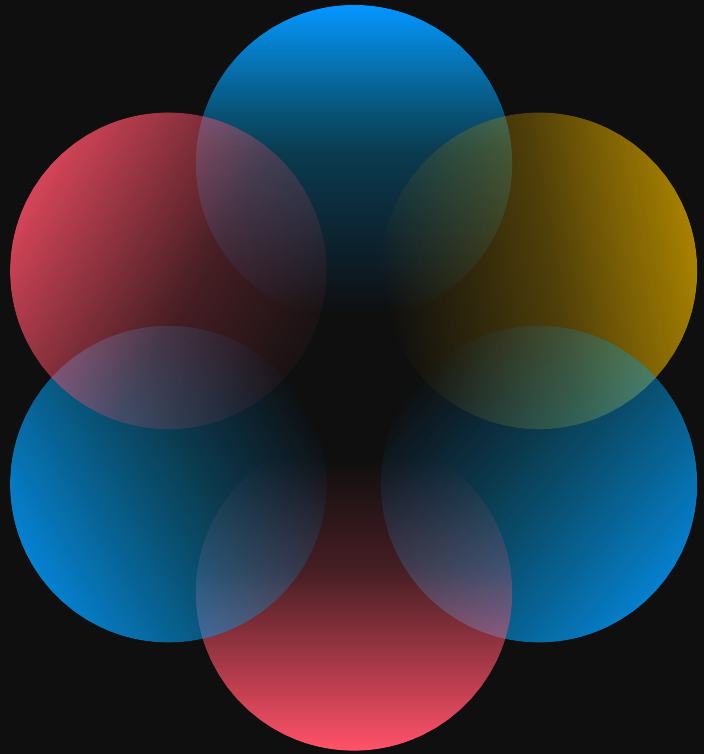
  this is a more complex thing to do

  -> a combination of:

  1. (het-job/het-step)
  2. cpu-masks
  3. arbitrary distribution
  4. MPMD

Atos

Known issues



Atos

SIGBUS error

- Applications fail with a SIGBUS during I/O on a lustre filesystem

- This problem was fixed with a lustre client update
- Problem should not occur since 2 weeks

- Are you still seeing problems? Please report to the helpdesk

Atos

Endpoint timeout

- Application fails with different error messages:

```
ucp_ep.c:1096 UCX ERROR ep 0x7ffbdc0661c0: error 'Endpoint timeout' on
dc_mlx5/mlx5_0:1 will not be handled since no error callback is installed
```

```
236: [l20122:1058445:0:1058445] ib_mlx5_log.c:162 Transport retry count
exceeded on mlx5_0:1/IB (synd 0x15 vend 0x81 hw_synd 0/0)
```

- InfiniBand card switches off for a short while
- Unfortunately, the exact reason is still unknown

- Turbo is disabled to reduce the problem. Performance impact ~10%.

Permission denied on Lustre

- Unfortunately, there is no reproducer for this problem yet.

- Observations so far:
  - This happens sporadically
  - It seems to happen especially for pool-data
  - MeSSy is (mainly?) impacted by this bug
  - The use of ACLs could be a factor

- Please inform the helpdesk, if:
  - You can consistently reproduce this issue
  - it occurs for data not in a pool
  - It occurs with applications other than MeSSy
  - It occurs without using ACLs

srun --distribution=XXXX:cyclic does not work exacly as expected

- for processors with more than one NUMA-domain per socket (like AMD EPYC) SLURM should treat the NUMA-domians as "sockets" when it comes to cyclic distributions, but uses the actual sockets instead)

  -> will be fixed in a future SLURM version

- in the meantime use cpu-masks or cpu-lists in combination to reproduce the adverted pacement and binding (see Elfenstaub)

Atos

Questions?

Thank you!

For more information please contact:
harald.braun@atos.net
john.donners@atos.net

Elfenstaub I :: hybrid MPI/OpenMP

SLURM batch script (cpu-masks)

```
# SBATCH header plus stuff ... (e.g. OMPI environment)

#launch application
export OMP_NUM_THREADS=4
mask4=0xf,0xf0000,0xf00000000,0xf0000000000,0xf0000000000000,0xf0000000000000000,0xf00000000000000000,0xf0000000000000000000000000,0xf0,0xf00000,0xf00000000,0xf0000000000,0xf0000000000000000,0xf0000000000000000000,0xf0000000000000000000000000,0xf0000000000000000000000000000,0xf00,0xf000000,0xf0000000000,0xf0000000000000,0xf0000000000000000,0xf0000000000000000000,0xf0000000000000000000000000,0xf000,0xf0000000,0xf00000000000,0xf000000000000000,0xf0000000000000000000,0xf0000000000000000000000000000000,0xf0000000000000000000000000000000
srun -l  --cpu-bind=v,cpu_mask=$mask4 --distribution=block:cyclic:block -- application

mask8=0xff,0xff0000,0xff00000000,0xff0000000000000,0xff0000000000000000,0xff0000000000000000000,0xff00000000000000000000000,0xff000000000000000000000000000,0xff00,0xff000000,0xff0000000000,0xff00000000000000,0xff0000000000000000000,0xff00000000000000000000000000,0xff000000000000000000000000000000
```


Atos

Elfenstaub II :: hybrid MPI/OpenMP

SLURM batch script (arbitrary distribution)

...same, but:

```
export SLURM_HOSTFILE=<my-directory-of-choice>/hfile
ppi=1; ppii=0
nl=$(nodeset -e $SLURM_NODELIST)
anode=$(echo $nl | cut -d ' ' -f 1)
for pp in $(seq 1 $ntask); do
    ppii=$((ppii + 1))
    echo $anode >> $SLURM_HOSTFILE
    if [ $ppii -eq 8 ] ; then
      ppi=$(($ppi + 1))
      if [ $ppi -gt $SLURM_JOB_NUM_NODES ] ; then ppi=1; fi
      anode=$(echo $nl | cut -d ' ' -f $ppi)
      ppii=0
    fi
done

srun -l  --cpu-bind=v,cpu_mask=$mask4 --distribution=arbitrary -- application
```


Atos