Piz Daint

Cray XC30 @ CSCS

Piz Daint

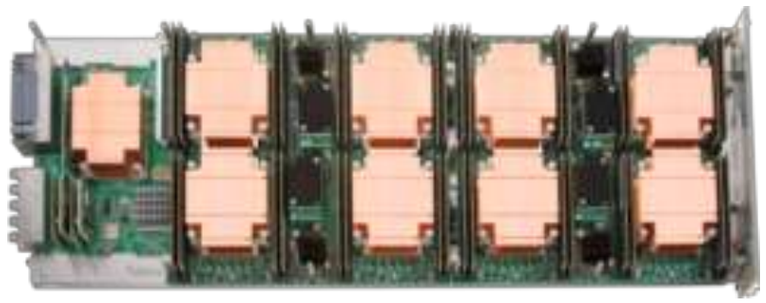
- Hybrid Cray XC30 – First Cray Supercomputer with Intel Sandybridge CPU + NVIDIA TESLA K20X GPU
- **5‘272 nodes**, each with Intel Xeon E5-2670 (8c) + 1 GPU
- **42‘176 cores** (84‘352 in hyperthread) – 7,79 Pflops Peak
- 2.5 PB scratch – based on Lustre parallel file system

Piz Daint - Ranking

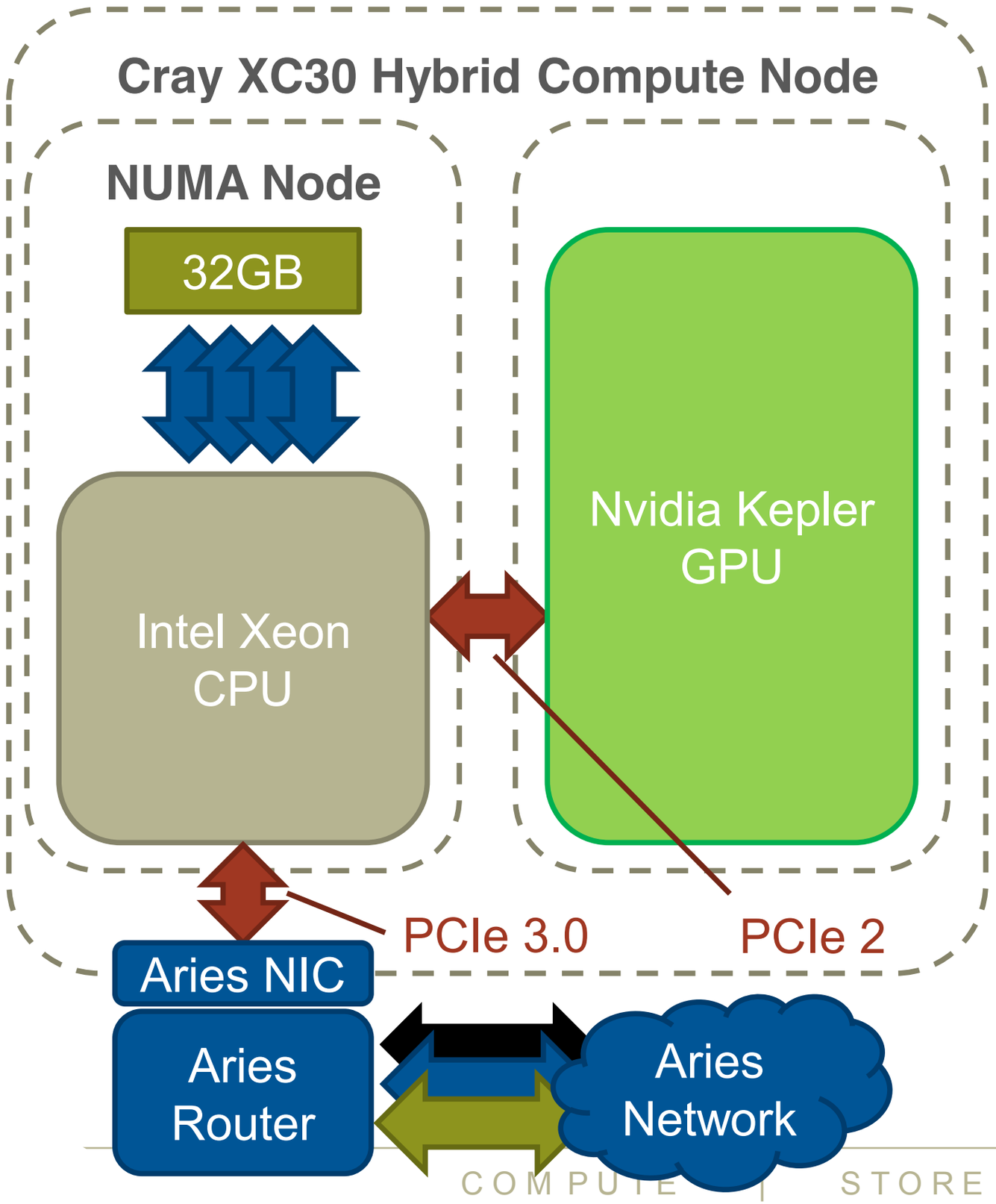
TOP500 The List.
http://top500.org

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ.	**Stampede** - PowerEdge C8220, Xeon	462,462	5,168.1	8,520.1	4,510

Cray XC30 Compute Node Architecture



Cray XC30 Hybrid Compute Node



Node features:

- **1 x Intel Xeon CPU**
  - 8-core Sandybridge
  - a single NUMA node
  - 32 GB memory per node
- **1 x Nvidia GPU**
  - Kepler K20X
  - 6 GB memory per node
- **1 x Aries NIC**
  - Connects to shared Aries router and wider network
  - PCI-e 3.0

Node compute performance

- **Intel Xeon E5-2670 "Sandybridge" CPU**
  - nominal frequency: 2.6 GHz
    - maximum "Turbo Frequency": 3.3 GHz
  - 8 physical cores per CPU
    - each core can run 2 "hyperthread" hardware threads
  - supports 256-bit AVX vector instructions
    - 4 double precision operations per clock cycle
  - Peak DP performance: 2.6x8x2x4 = 166.4 Gflop/s per CPU

- **Nvidia Tesla K20x "Kepler" GPU**
  - nominal frequency: 0.732 GHz
    - maximum boost frequency: 0.784 GHz
  - 896 double precision cores per GPU
    - plus 2688 single precision cores (ratio SP:DP = 3:1)
  - FMA instruction allows 2 DP operation per clock cycle
  - Peak DP performance: 0.732x896x2 = 1311.7 Gflop/s per GPU

Cray XC30 System Building Blocks

Compute Blade

4 Compute Nodes

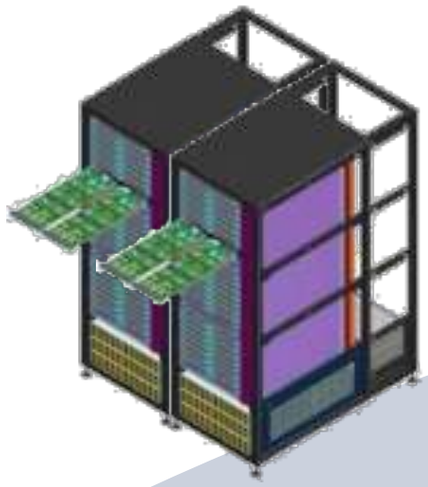
Chassis

Rank 1 Network

16 Compute Blades

No Cables

64 Compute Nodes



Group

Rank 2 Network

Passive Electrical Network

2 Cabinets

6 Chassis

384 Compute Nodes

System

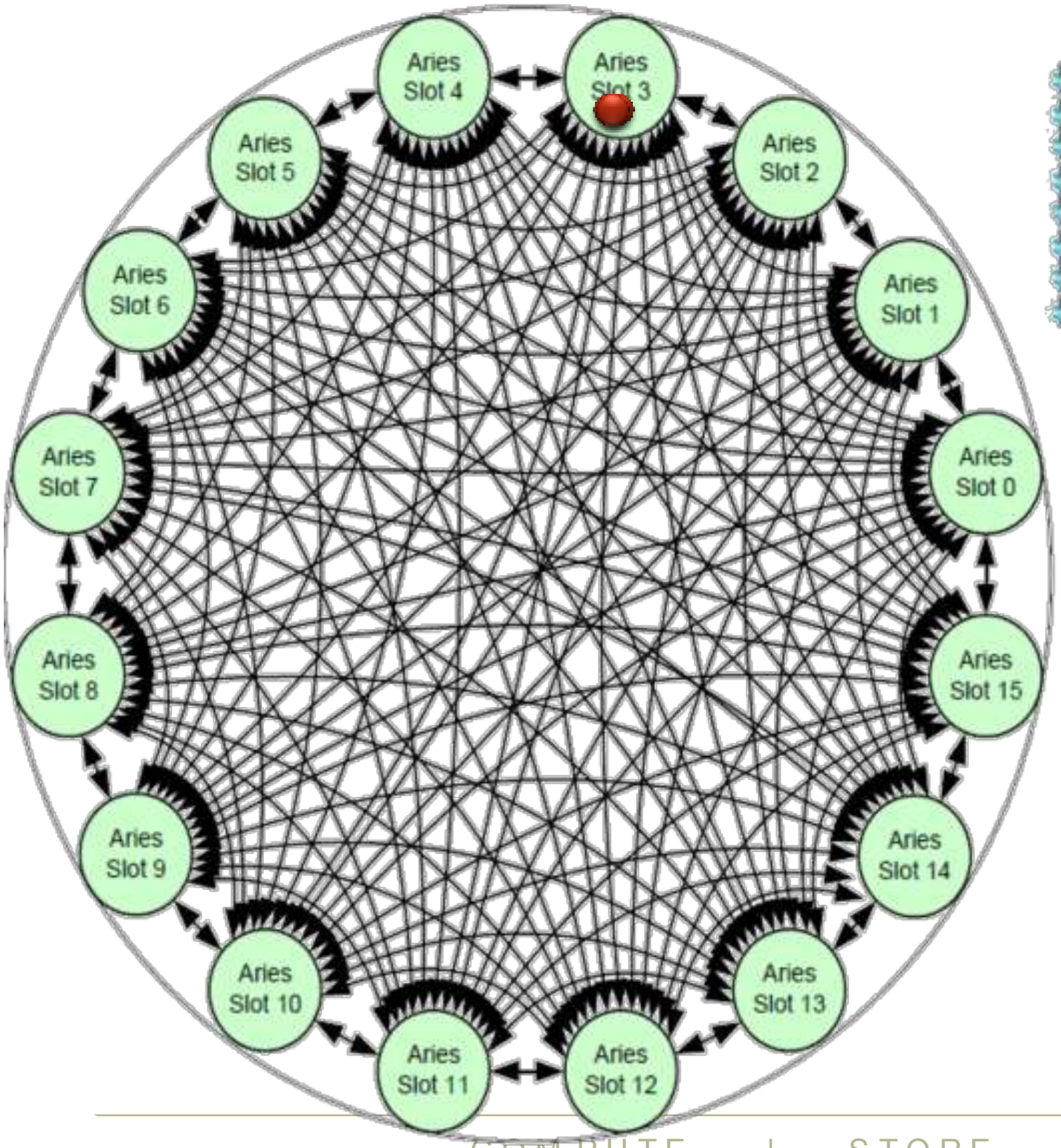
Rank 3 Network

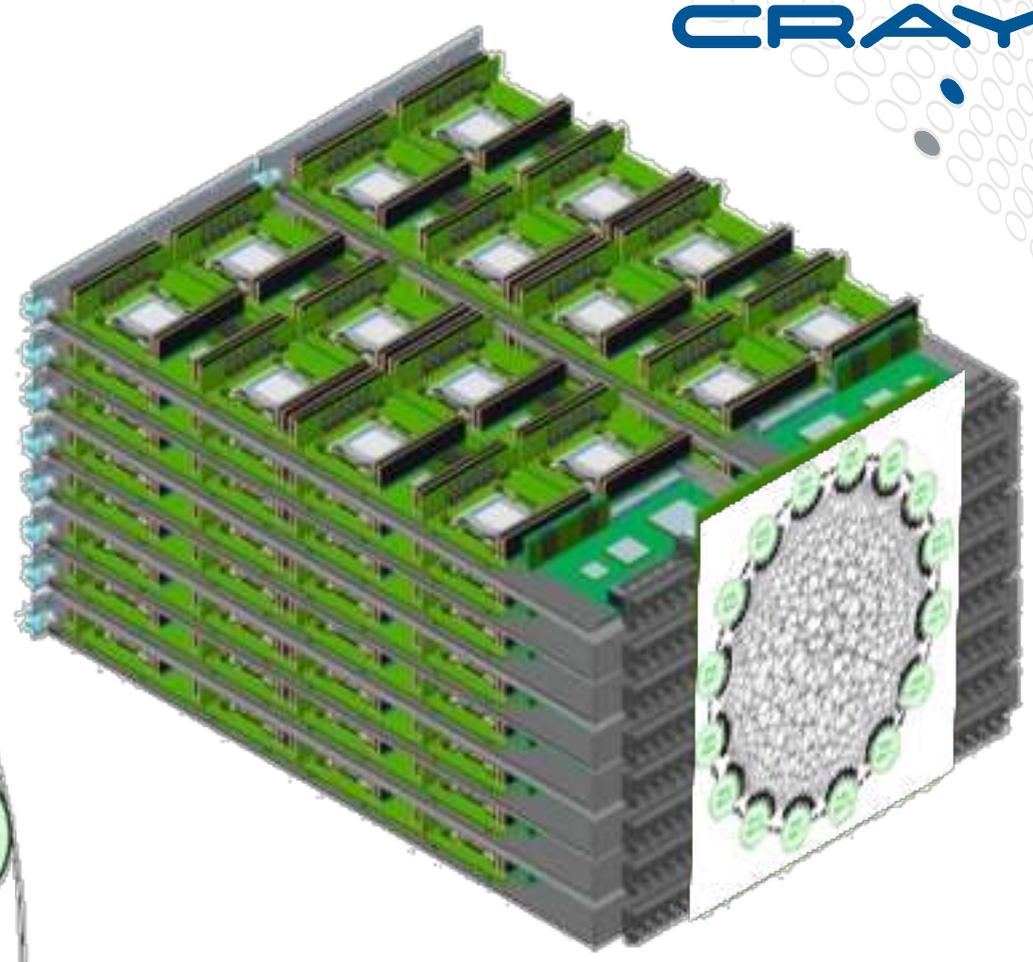
Active Optical Network

Hundreds of Cabinets

Up to tens of thousands of nodes

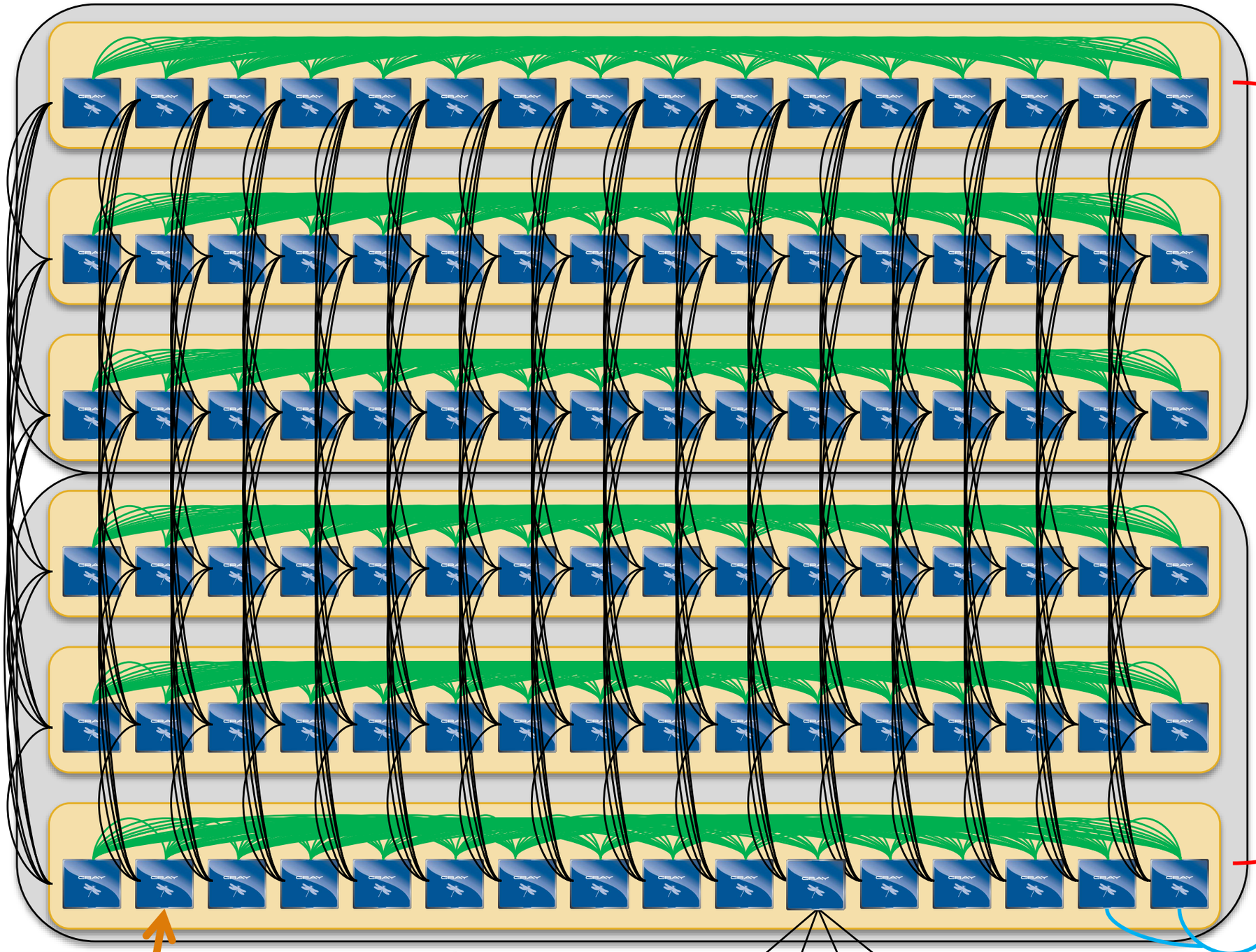
Cray XC30 Rank1 Network



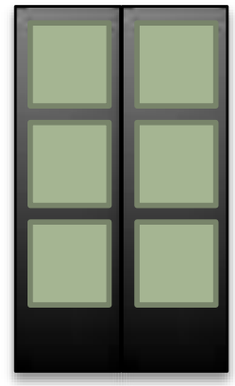


- Chassis with 16 compute blades
- 128 Sockets
- Inter-Aries communication over backplane
- Per-Packet adaptive Routing

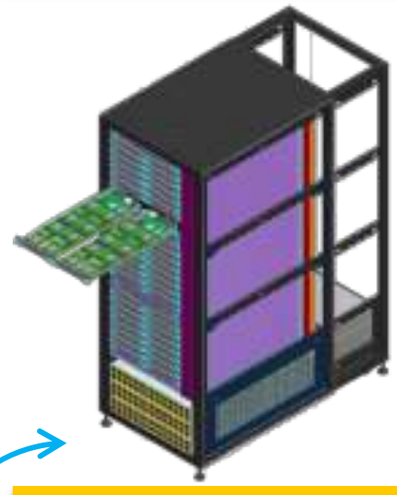
Cray XC30 Rank-2 Copper Network



2 Cabinet Group 768 Sockets



6 backplanes connected with copper cables in a 2-cabinet group: "Black Network"



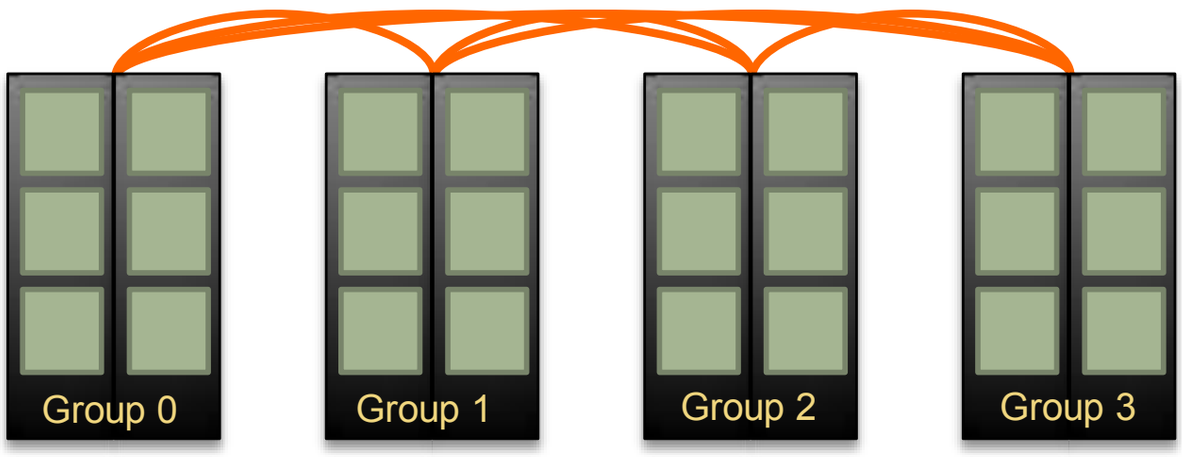
Active optical cables interconnect groups "Blue Network"

16 Aries connected by backplane "Green Network"

4 nodes connect to a single Aries

Cray XC30 Network Overview – Rank-3 Network

- An all-to-all pattern is wired between the groups using optical cables (blue network)
- Up to 240 ports are available per 2-cabinet group
- The global bandwidth can be tuned by varying the number of optical cables in the group-to-group connections



Example: An 4-group system is interconnected with 6 optical "bundles". The "bundles" can be configured between 20 and 80 cables wide

Login

- Piz Daint only accessible inside CSCS network:

1. ssh -Y studXX@ela.cscs.ch

2. ssh -Y daint

Modules

- Module system similar to Euler

  - `module list`
    `show loaded modules`

  - module avail
    list all available modules

  - module load XYZ
    load module XYZ

  - module switch X1 X2
    unload module X1 and load X2

Programming Environment

```
stud01@daint104:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.7                          14) gni-headers/3.0-1.0501.8317.12.1.ari
  2) eswrap/1.1.0-1.010400.915.0              15) xpmem/0.1-2.0501.48424.3.3.ari
  3) switch/1.0-1.0501.47124.1.93.ari         16) job/1.5.5-0.1_2.0501.48066.2.43.ari
  4) craype-network-aries                     17) csa/3.0.0-1_2.0501.47112.1.91.ari
  5) craype/2.2.1                             18) dvs/2.4_0.9.0-1.0501.1672.2.122.ari
  6) cce/8.3.4                                19) alps/5.1.1-2.0501.8713.1.1.ari
  7) totalview-support/1.1.4                  20) rca/1.0.0-2.0501.48090.7.46.ari
  8) totalview/8.11.0                         21) atp/1.7.5
  9) cray-libsci/13.0.1                       22) PrgEnv-cray/5.1.29
 10) udreg/2.3.2-1.0501.7914.1.13.ari         23) craype-sandybridge
 11) ugni/5.0-1.0501.8253.10.22.ari           24) slurm
 12) pmi/5.0.5-1.0000.10300.134.8.ari         25) cray-mpich/7.0.4
 13) dmapp/7.0.1-1.0501.8315.8.4.ari          26) ddt/4.3rc7
```

Programming Environment

	PrgEnv	Fortran	C	C++	w/ OpenMP	w/ OpenAC
CRAY	PrgEnv-cray	ftn	cc	CC	by default	-h acc
INTEL	PrgEnv-intel	ftn	cc	CC	-openmp	N/A
GNU	PrgEnv-gnu	ftn	cc	CC	-fopenmp	(future)
PGI	PrgEnv-pgi	ftn	cc	CC	-mp	-acc -ta=kepler

- Compile programs with the provided compiler wrapper:

```
> CC main.cpp -o prog
```

Storage

- **Home filesystem $HOME=/users/$USER**
  - quota of 10 Gbytes per user and backed up
  - not to be be used for simulation I/O, usually for keeping source code/binaries.
- **Scratch filesystem $SCRATCH=/scratch/daint/$USER**
  - to be used for I/O during a simulation
  - no quota but no backup as well: temporary storage only!
  - data subject to a cleaning policy: see details on CSCS User Portal

Types of nodes

- Login node
  - compile, data transfer
- Service node
  - runs the job script
- Compute node
  - runs the parallel jobs

Submitting a job

1. Prepare a job sbatch script

```
#!/bin/bash -l
#SBATCH --ntasks=48
#SBATCH --time=00:30:00
aprun -B ./test
```

2. Submit the job

```
> sbatch job.sbatch
```

Example sbatch script

```
#!/bin/bash -l
#SBATCH --job-name=mytest
#SBATCH --time=00:05:00
#SBATCH --ntasks=48
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH --output=out.log.j%j
#SBATCH --error=out.log.j%j

echo "The current job ID is $SLURM_JOB_ID"
echo "Running on $SLURM_JOB_NUM_NODES nodes: $SLURM_JOB_NODELIST"
echo "Using $SLURM_NTASKS_PER_NODE tasks per node"
echo "A total of $SLURM_NTASKS tasks is used"

export OMP_NUM_THREADS=2
echo "Running on $HOSTNAME"

echo "Launching parallel job."
aprun -B ./test
echo "Parallel execution finished."
```

sbatch options

useful debugging output

your commands

Sbatch script components

- Sbatch commands
  `#SBATCH` ***`options`***
- Service commands
  - executed on service nodes!
- Parallel job commands
  - dispatched to compute nodes
  - launched with **`aprun`**
  - **NO** mpirun

Useful Slurm / aprun options

Request	Sbatch	Aprun
#Processes	--ntasks	-n
#Threads per process	--cpus-per-task	-d
#Processes per node	--ntasks-per-node	-N

shortcut:
aprun –n X –N Y –j Z – d W == aprun -B

- more sbatch options: **`man sbatch`**
- more aprun options: **`man aprun`**

Interacting job

- ONLY FOR SHORT TIMES

1. `salloc –N <number of nodes>`
   options similar to sbatch

2. `aprun <options> <myexecutable>`

Interacting GPU profiling

- **nvvp** (visual profiler) is available via the Cluster Compatibility Mode (CCM).
  - (X11 forwarding must be enabled)
  - `module load cudatoolkit`
  - `salloc –N1 -p ccm`
  - `module load ccm`
  - `export PBS_JOBID=$SLURM_JOBID`
  - `ccmlogin –V`
  - `nvvp`

Queuing system

- `squeue`
  list all jobs in the queue

- **`squeue -u $USER`**
  list only your jobs

References

- More info
  - http://www.cscs.ch/computers/piz_daint_piz_dora/index.html
  - http://user.cscs.ch/get_started/run_batch_jobs/piz_daint_and_piz_daint_extension/index.html
  - video tutorial: http://user.cscs.ch/get_started/run_batch_jobs/tutorial/index.html
- DO NOT write to help@cscs.ch