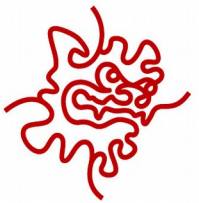
HPC and Scientific Computing at OIST (Basic Training)

Scientific Computing & Data Analysis Section
September 2015



OIST

OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY GRADUATE UNIVERSITY

HPC and Scientific Computing at OIST

- **HPC concepts**
- **Scientific software for HPC**
- **Scientific Programming in HPC**
- **HPC resources and infrastructure at OIST**
- **Getting started with HPC at OIST**

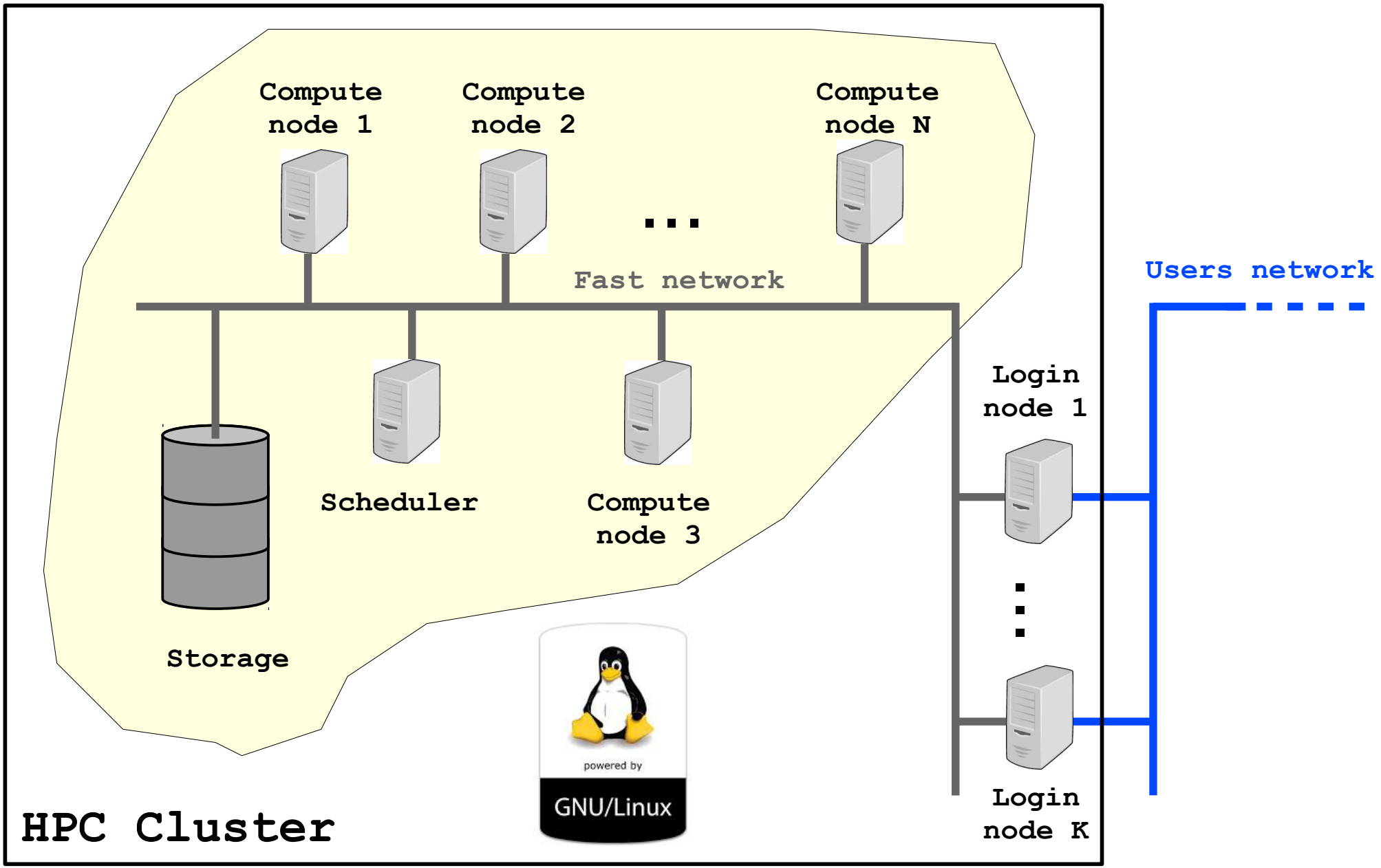
OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY

Part 1

- **HPC concepts**
  - **Node, core, storage, filesystem, scheduler, parallelism**
- **Scientific software for HPC**
- **Scientific Programming in HPC**

HPC concepts

Architectures and operating system



OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY

HPC cluster in Data Center

HPC concepts
components

- Node → core → slot
- Storage
- File-system
- Scheduler
- Parallelism

HPC concepts

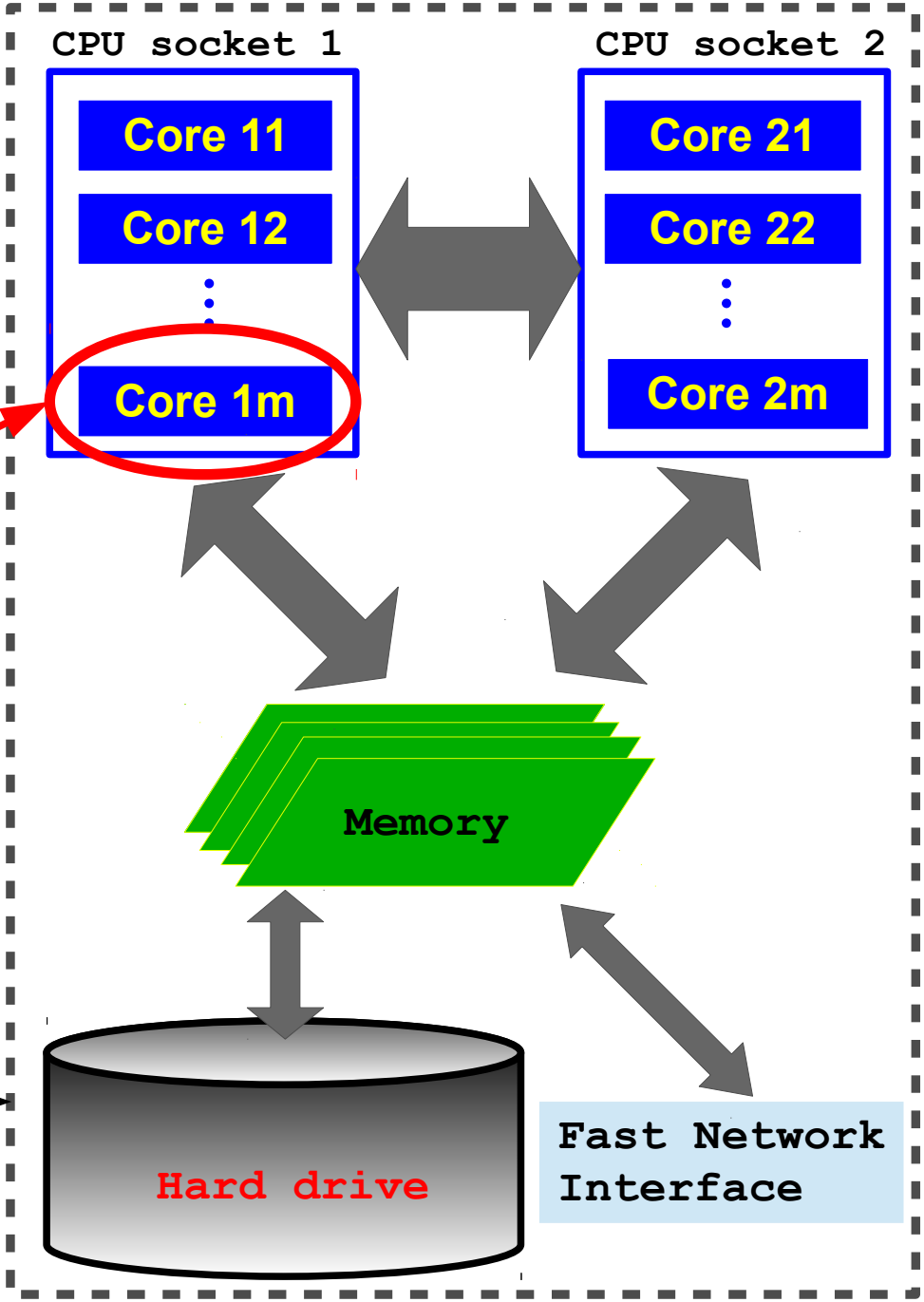
- **Node → core → slot**
- Storage
- File-system
- Scheduler
- Parallelism

Slot

Core: cluster computation unit

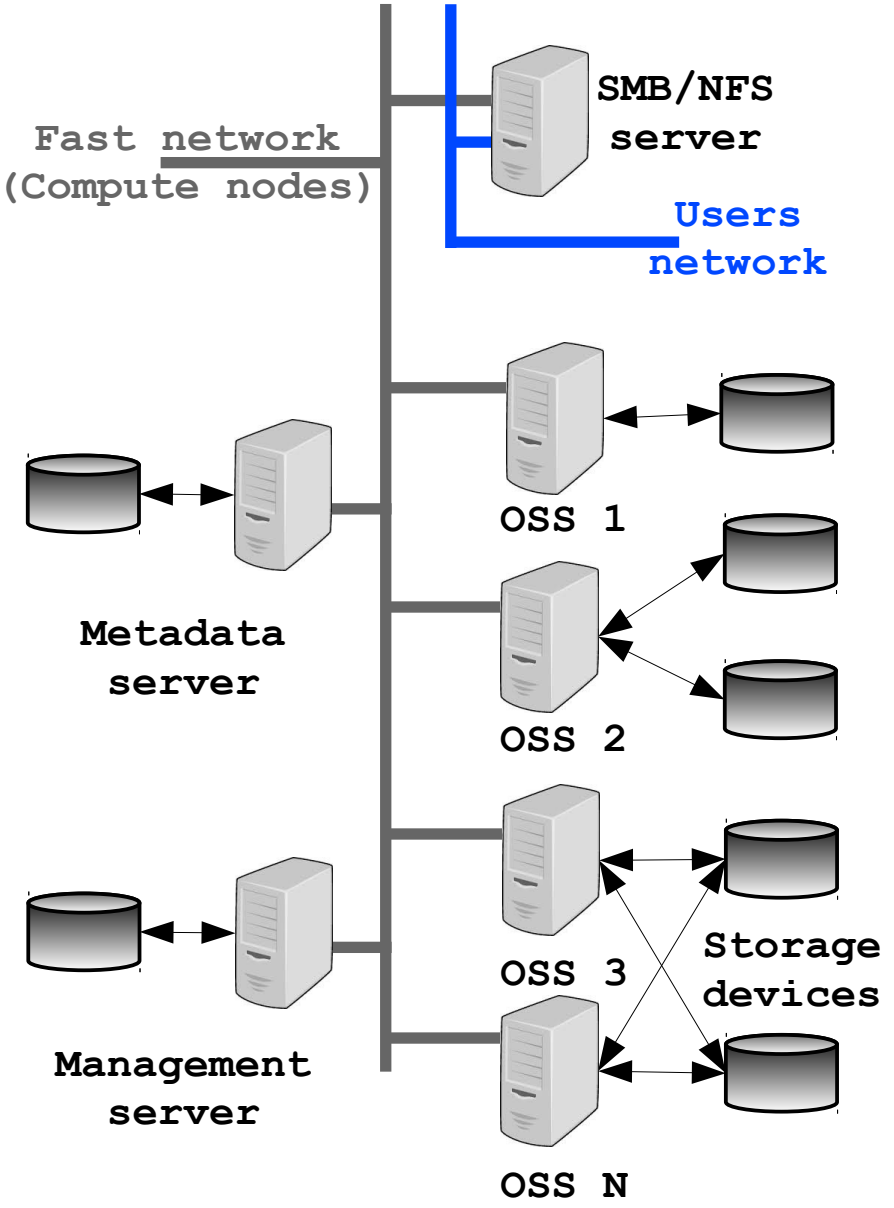
Node

= one high-spec workstation



HPC concepts

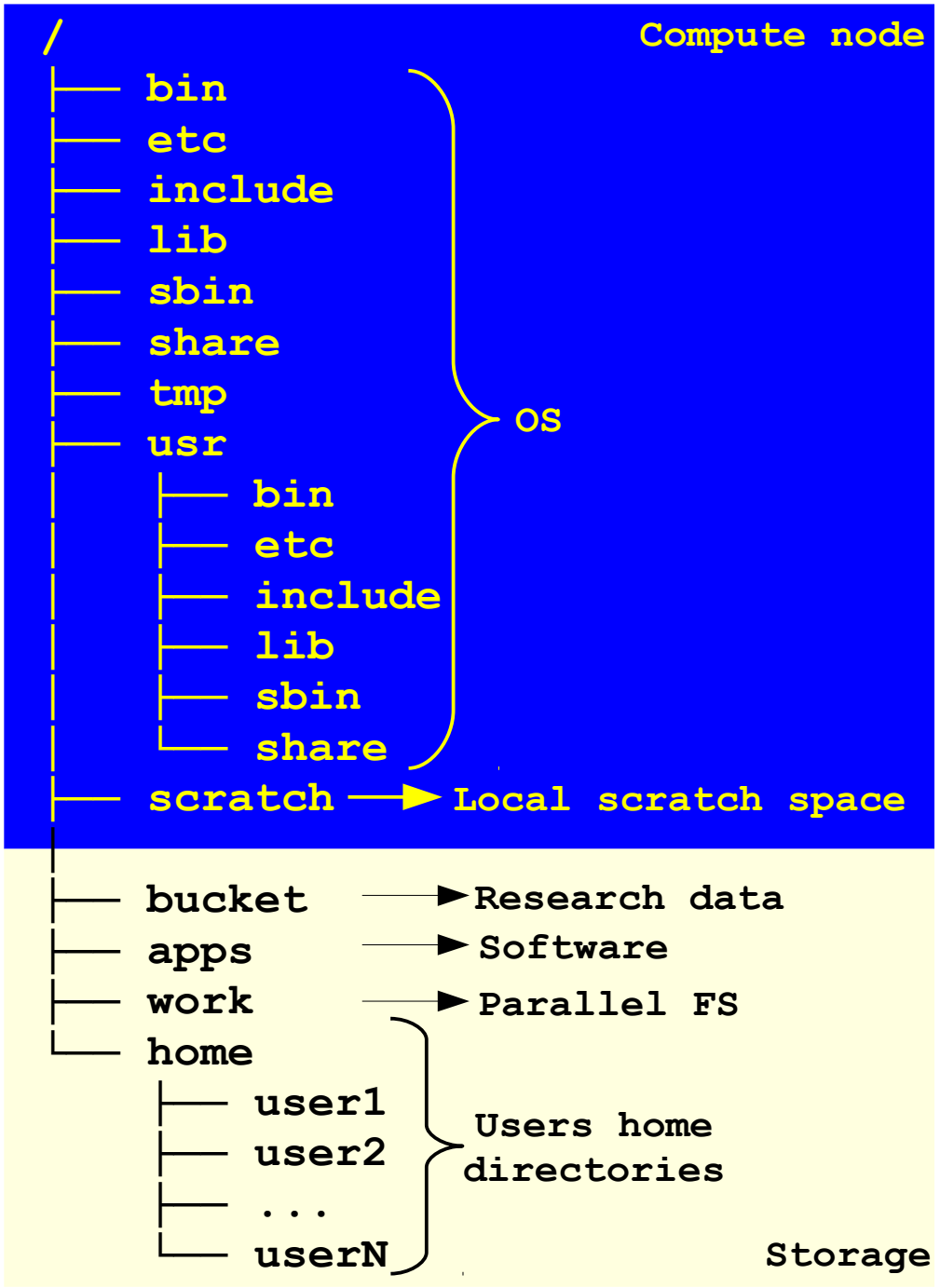
- Node → core → slot
- **Storage**
- File-system
- Scheduler
- Parallelism



HPC concepts

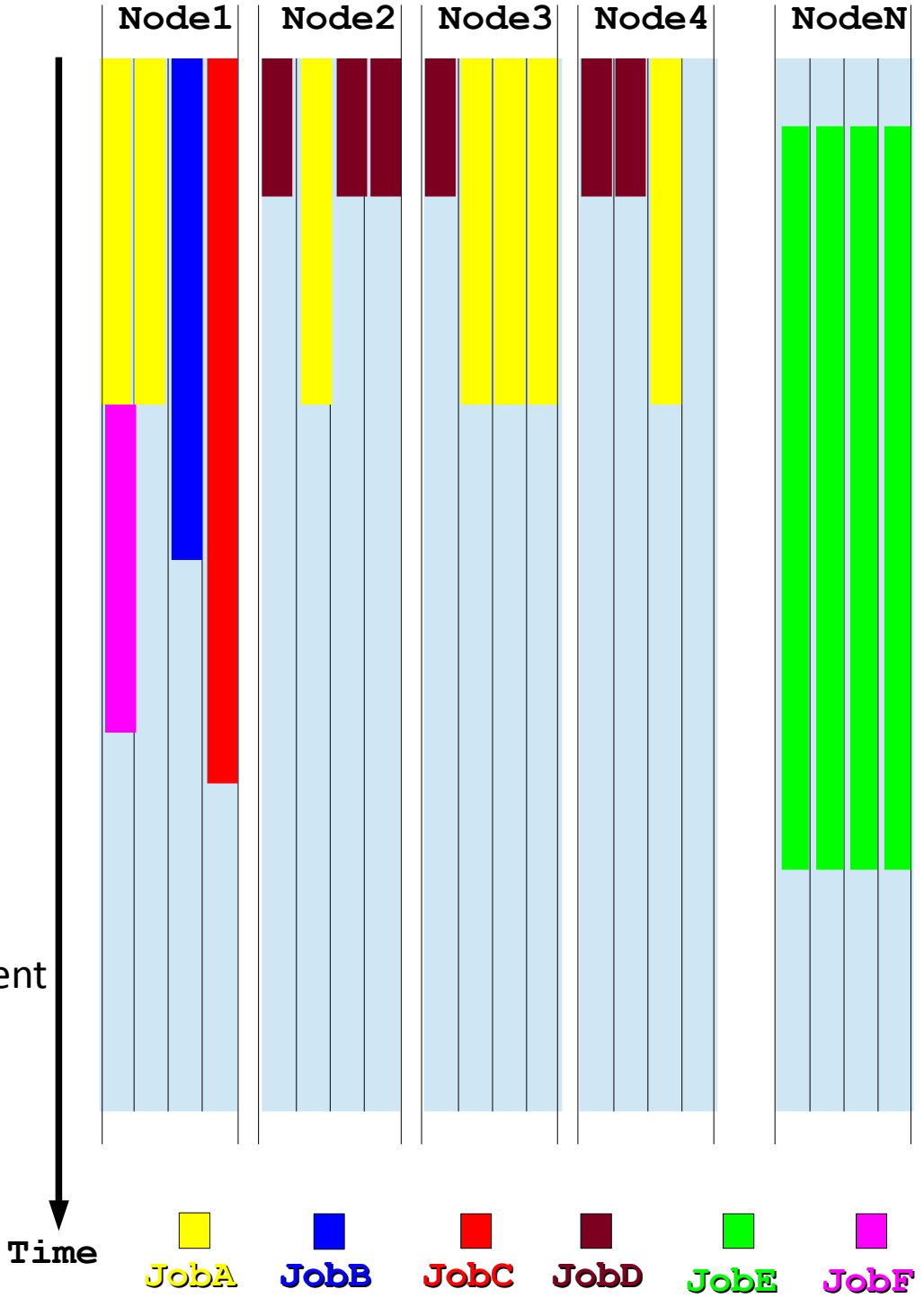
- Node → core → slot
- Storage
- **File-system**
- Scheduler
- Parallelism

File hierarchy



HPC concepts

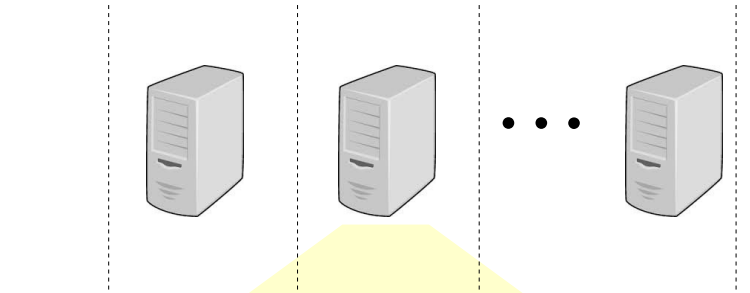
- Node → core → slot
- Storage
- File-system
- **Scheduler**
- Parallelism



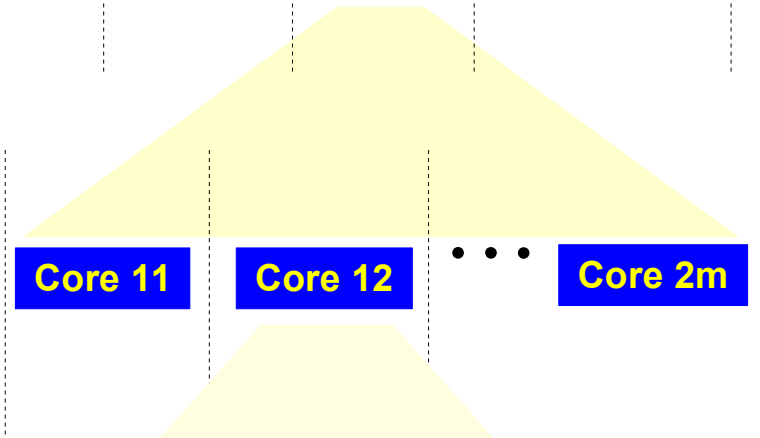
Common Job scheduler:
SLURM: Simple Linux Utility for Resource Management
SGE: Sun Grid Engine (Son of/Univa/Oracle Grid Engine)
PBS: Portable Batch System
LSF: Platform Load Sharing Facility

HPC concepts

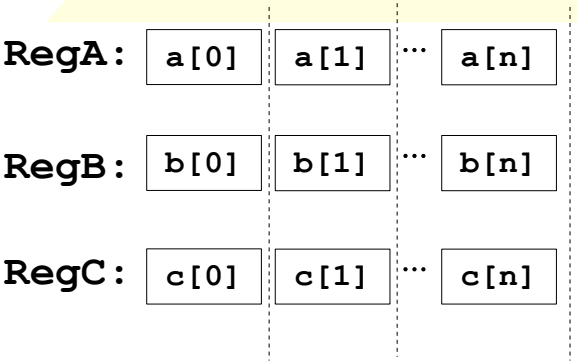
- Node → core → slot
- Storage
- File-system
- Scheduler
- **Parallelism**



Distributed:
MPI
pipeline



Shared:
Threads
openMP



High performance:
Vectorization
Accelerators

```
Ex: RegC = regA * regB
```

Parallelism in HPC

- Different jobs running on different slots (task array, jobs submission script, etc)
  - Embarrassingly parallel problem
  - Computation pipeline
- A job running on several slots (MPI)
  - Distributing computing problem
  - Reduction
  - Embarrassingly parallel problem
- A job running on different slots on a same node (multithread)
  - Shared memory computing problem
  - Reduction

Scientific software for HPC

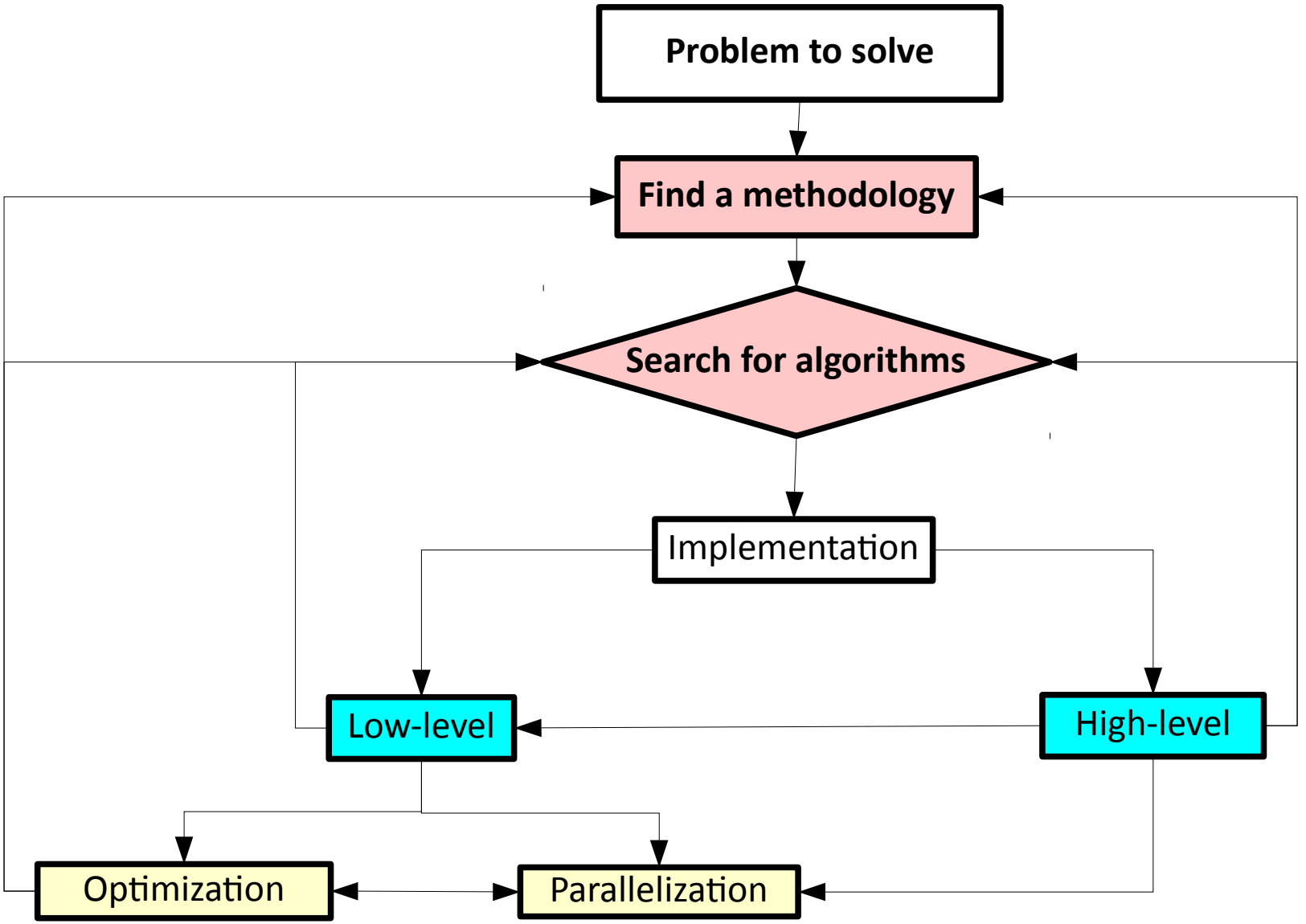
- Software initially developed on PC used directly or after modification on HPC
  - Licensed software (ex: Mathematica, Matlab, COMSOL, etc.)
  - Tools made available from institutes or academia publications
  - Open source software, computation pipelines
  - etc.
- Optimized scientific software initially developed for HPC platforms
  - Life sciences, weather research and forecasting
  - Engineering, geological and energy industries
  - Manufacturing and digital content creation
  - Finance services
  - etc.

OKINAWA INSTITUTE OF SCIENCE AND TECHNOLOGY

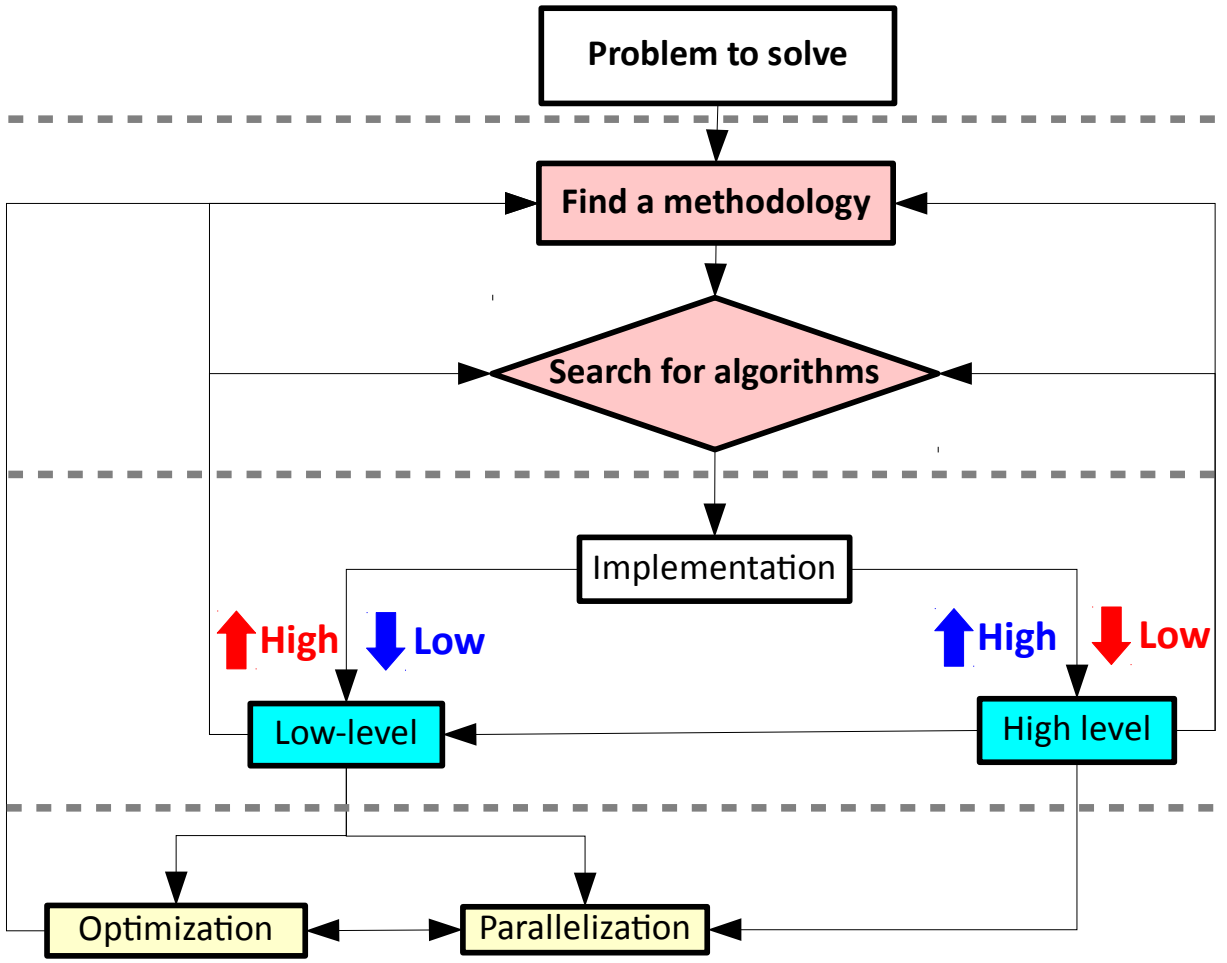
Scientific Programming in HPC

- Scientific programming does not reduce to write code and writing code is never the first step
- Programming languages are tools adapted for specific tasks
- “What is the best programming language for HPC ?” can not be answered without a context
- Expressing computational scientific problem may benefit from a diagram

Scientific computing



Programming in scientific computing



- Mathematical tools, Algorithmic
- Litterature
- CAS: Mathematica, Maple,etc

- High-level : Mathematica, maple, matlab, python, R.
- Low-level : C, C++, Fortran

Computational performance

Human productivity

Parallelization : openMP, MPI, GPGPU, etc.
(Parallelization often impacts on the choice of the methodology)
Optimization : profiling tools

Effective implementation
High-level programming

- Allows more productive scientific programming
- Can benefit from many complex functions already implemented
- Can help investigating methodology using high level abstraction (ex: symbolic calculations)
- Non-computing specialists can benefit from a quicker learning curve
- Typical languages: Mathematica, Matlab, Python, R, Perl, etc.

Efficient implementation
Low-level programming

- Allows deep optimization with efficient utilization of hardware resources (memory, CPU, network bandwidth, accelerators, storage, etc.)
- Works done with high-level languages is **not** lost
- Many functions and scientific libraries (found in high-level language) are also available
- Wildly used in HPC software implementation and supported by practically all HPC system
- Typical languages : C, C++, fortran

Parallelization

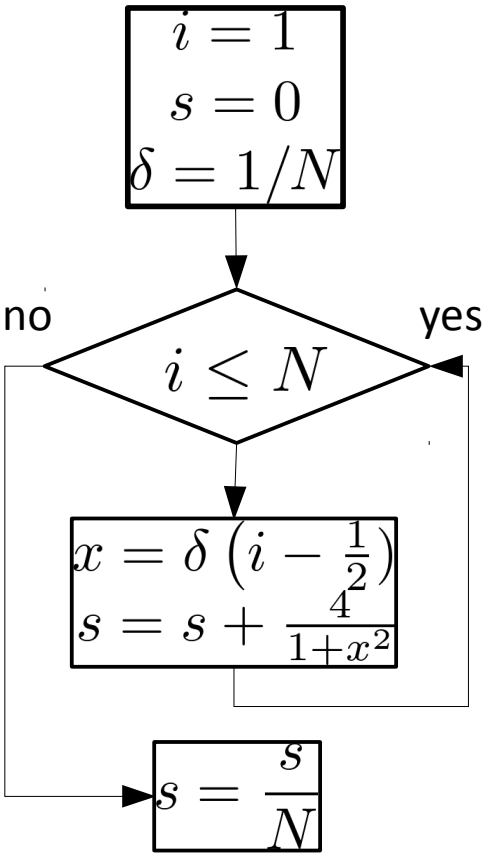
- Questions to ask:
  - Can the problem really be solved in a parallel way ?
  - Is the gain for running in parallel significant ? $\frac{T_{\text{serial}}}{T_{\text{parallel}}} \gg 1$?
- Action to take
  - Choose type of parallelism to use: openMP, GPGPU, MPI, embarrassingly parallel, etc.
  - May need to rethink the methodology
- Indispensable to efficiently and effectively use HPC clusters

Different parallelization methods

- Multithreading approaches
  - Multiple threads having access to the same memory location. Example : openMP, Intel Phi (using **`#pragma`**), CUDA (rewriting of code), etc.
- Vectorization
  - Use vectorization units incorporated in the CPU
  - Can be done by hand (intrinsics) or by the compiler
  - May have to reorganize code
- MPI : inter-communication of independent processes

Example of parallelization
serial version

- Compute the following approximate of π
  - $\pi \approx h \sum_{i=1}^{N} \frac{4}{1+h^2\left(i-\frac{1}{2}\right)^2} \qquad h = \frac{1}{N}$
  - A serial implementation in C language will look like this




```
double x, dx, sum=0.0

dx = 1.0 / (double)N;
for (i = N; i >= 1; i--) {
  x = dx * ((double)i - 0.5);
  sum += 4.0 / (1.0 + x*x);
}
sum = sum * dx;
```

runtime=2s for N=1,000,000,000

Example of parallelization
parallel version using OpenMP (multi-thread)

- Parallelization consists of distributing the for loop over different threads and then doing a final reduction
  - OpenMP uses **`#pragma`** to tell the compiler what and how to parallelize

```
double x, dx, sum=0.0

dx = 1.0 / (double)N;
#pragma omp parallel for reduction(+:sum)
for (i = N; i >= 1; i--) {
  x = dx * ((double)i - 0.5);
  sum += 4.0 / (1.0 + x*x);
}
sum = sum * dx;
```

Runtime=0.58s for N=1,000,000,000 with 4 threads

Example of parallelization
parallel version using MPI

- Parallelization consists of distributing the for loop over different processes and then doing a final reduction

```
double x, dx, psum, sum=0.0
long int kh;

MPI_Comm_size(MPI_COMM_WORLD,&m);
MPI_Comm_rank(MPI_COMM_WORLD,&r);

psum = 0.0;
n = N / m;
kh = (r == m - 1) ? N : (r+1)*n;
dx = 1.0 / (double)N;
for (i=kh; i>=r*n+1; i--) {
  x = dx * ((double)i - 0.5);
  psum += 4.0/(1.0 + x*x);
}

MPI_Reduce(&psum, &sum, 1, MPI_DOUBLE, MPI_SUM,
           0, MPI_COMM_WORLD);
if (r == 0)
  sum = sum * dx;
```

Runtime=0.56s for N=1,000,000,000 with m=4 processes

Optimization

- Last step to tackle
- Time consuming step; can be simplified by usage of optimized libraries
- Benefit from profiling tools, but good grasp of underlying algorithms is more important
- Different algorithms can give the same answer (ex: sorting algorithm)

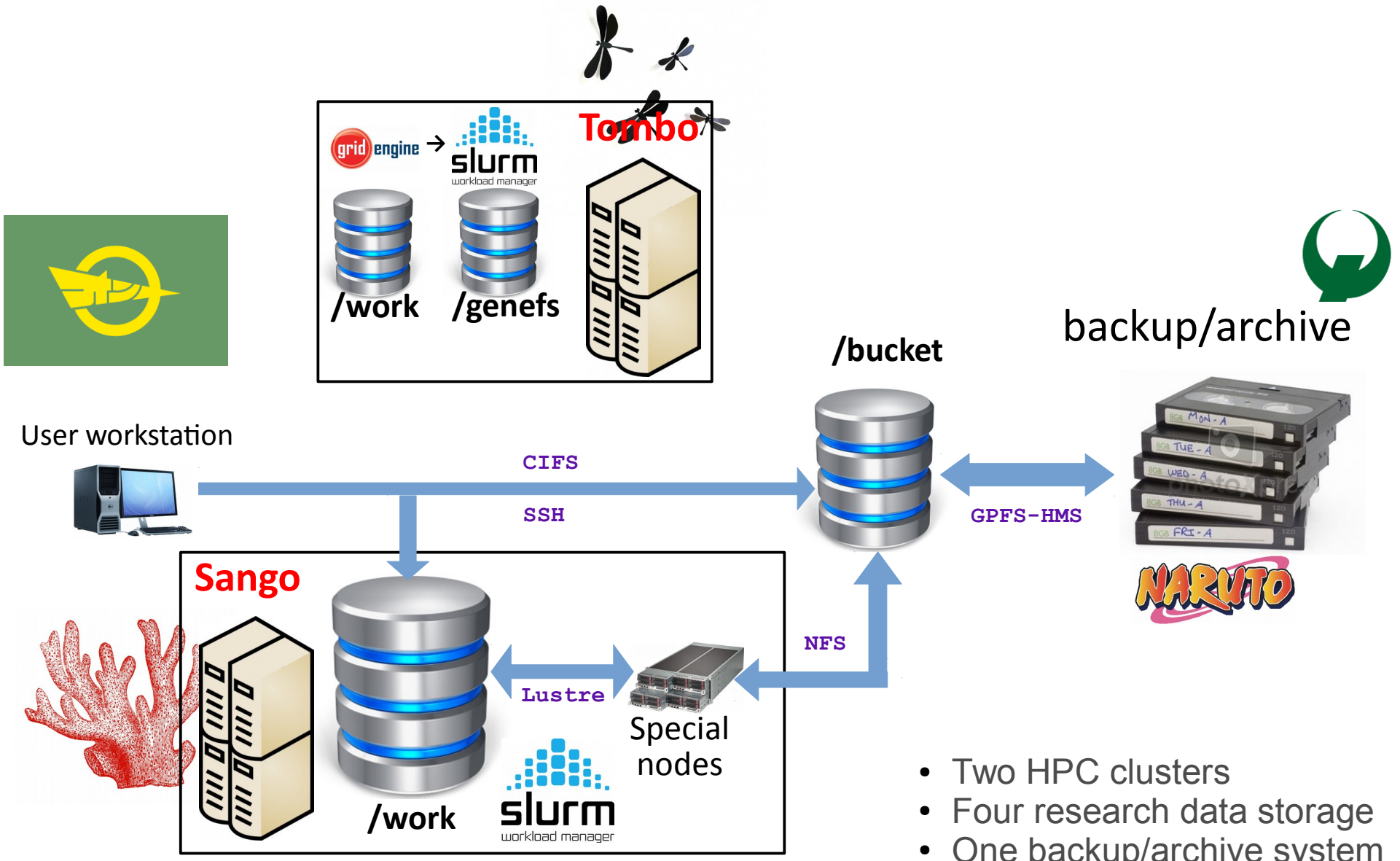
Tools used in HPC implementation

- Compiled languages: C. C++, Fortran
  - Used mainly for developing large programs
  - Compiler available GCC and Intel Compiler Suites
- Script languages: Shell(bash, csh, ksh, etc.), perl, pyhton, R, matlab, mathematica
  - Multipurpose and particularly useful for data pre-/post-proccessing, generating figures, etc.
  - Use to integrate compiled tools into job scripts
  - Generating and compiling code

Part 2

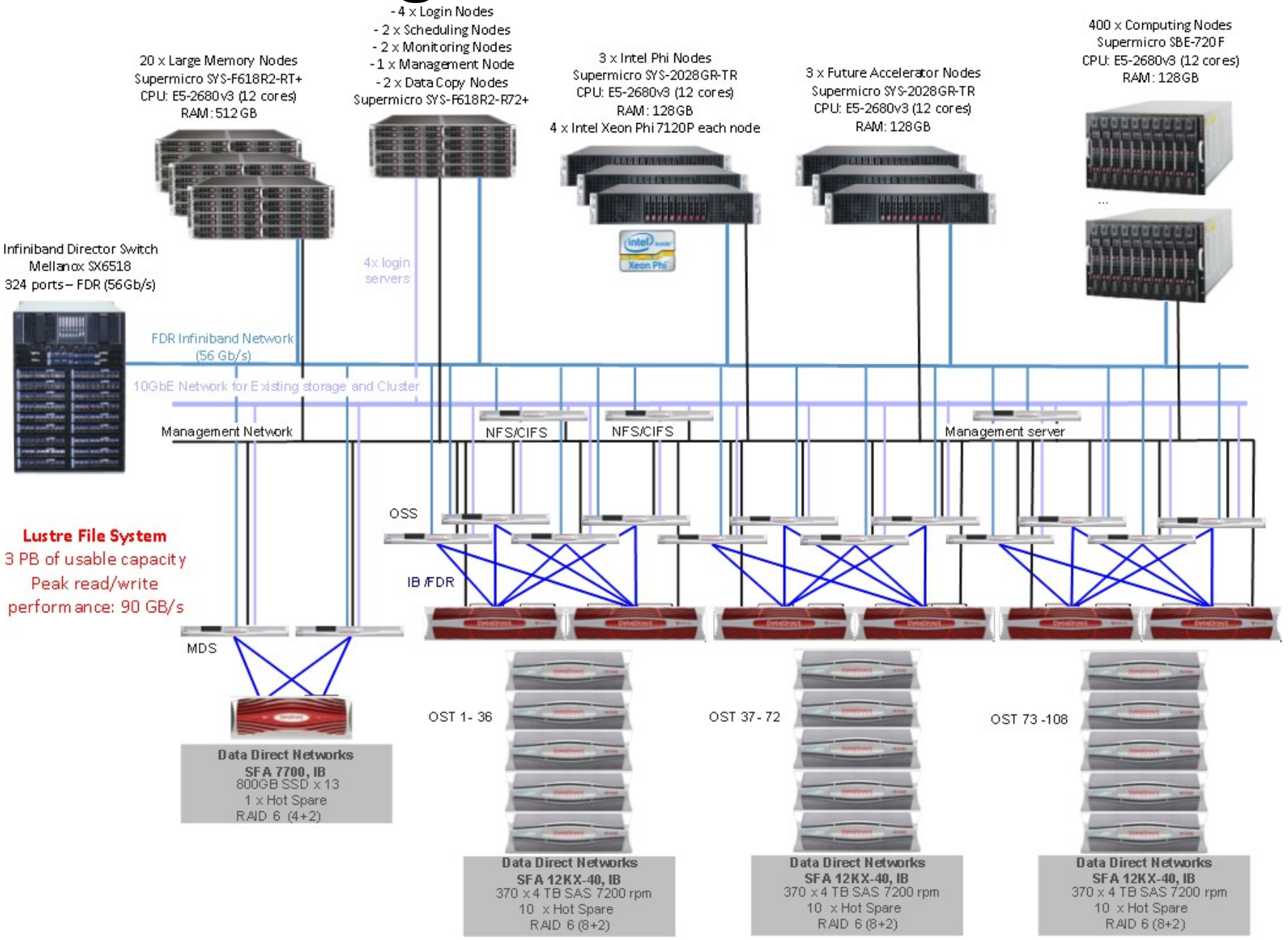
- **HPC resources and infrastructure at OIST**
  - Overview of OIST HPC resources
  - HPC clusters infrastructure
  - SLURM (components, concepts, partition, commands)
- **Getting started with HPC at OIST**
  - Accounts
  - Use the cluster
  - Best Practices

Overview of HPC resources at OIST



- Two HPC clusters
- Four research data storage
- One backup/archive system
- SLURM scheduler

Sango cluster (as of 2015)

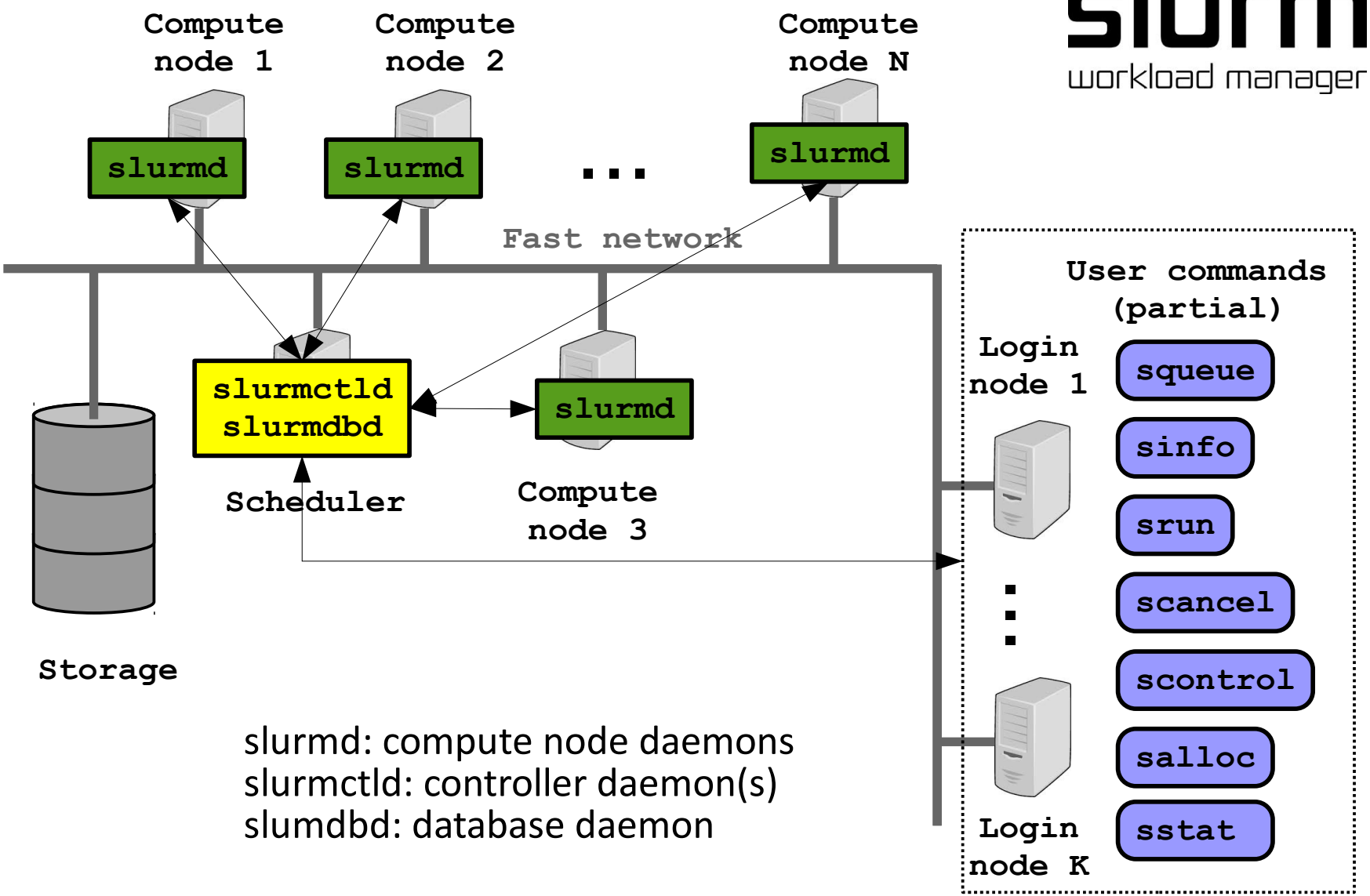


Sango cluster (in pictures)

Mellanox® Infiniband Director Switch

DDN® 3 Petabyte Storage

SLURM on Sango components



slurmd: compute node daemons
slurmctld: controller daemon(s)
slumdbd: database daemon

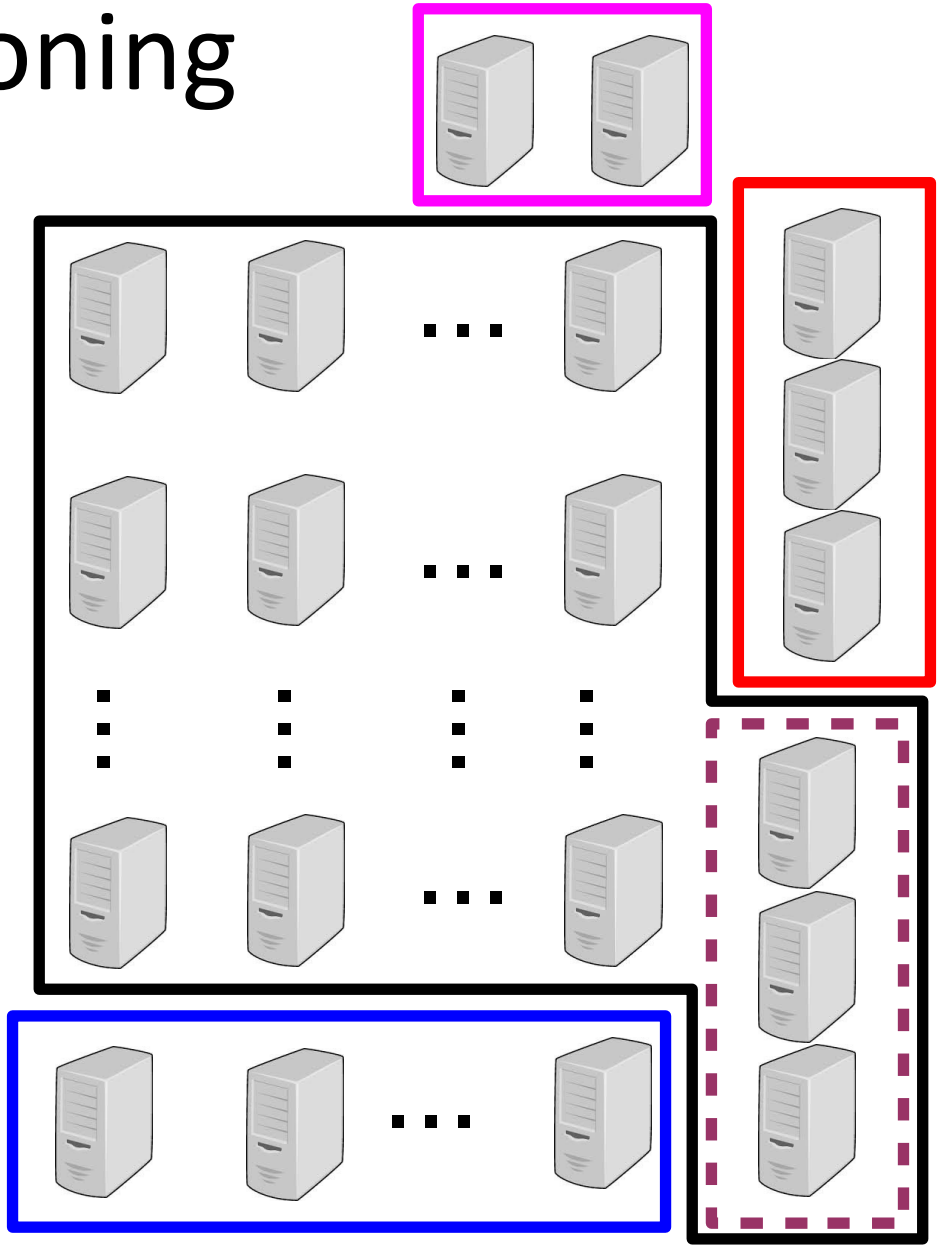
SLURM at OIST concepts

- Manage and schedule the allocation of a partitioned cluster resources
- Group node into sets called **partitions**; Jobs are submitted to a partition to run (similar to “queue” in SGE)
- Allocate consumable resources:
  - **CPU**: smallest consumable resources; **core** for multi-core machine; thread for hyper-threaded machine (equivalent to SGE slot)
  - **TASK**: synonym of process; for MPI it is a process or rank
  - **MEMORY**: amount of physical memory
- Allow charging resources consumption on a project basis using ACCOUNT

SLURM on Sango partitioning

- Partitions:
  - Compute nodes (**compute**)
    - ✔ 400 x 128GiB + 4 x 128GiB
  - Intel Xeon Phi nodes (**phi**)
    - ✔ 3 x 4 Xeon Phi
  - Large memory nodes (**largemem**)
    - ✔ 20 x 512 GiB
  - Special nodes (**datacp**)
    - ✔ 2 x 128GiB
  - nVidia GPGPU nodes (**gpu**)
    - ✔ 3 x 4 dual-gpu K80

Default partition: **compute**



Sango cluster nodes

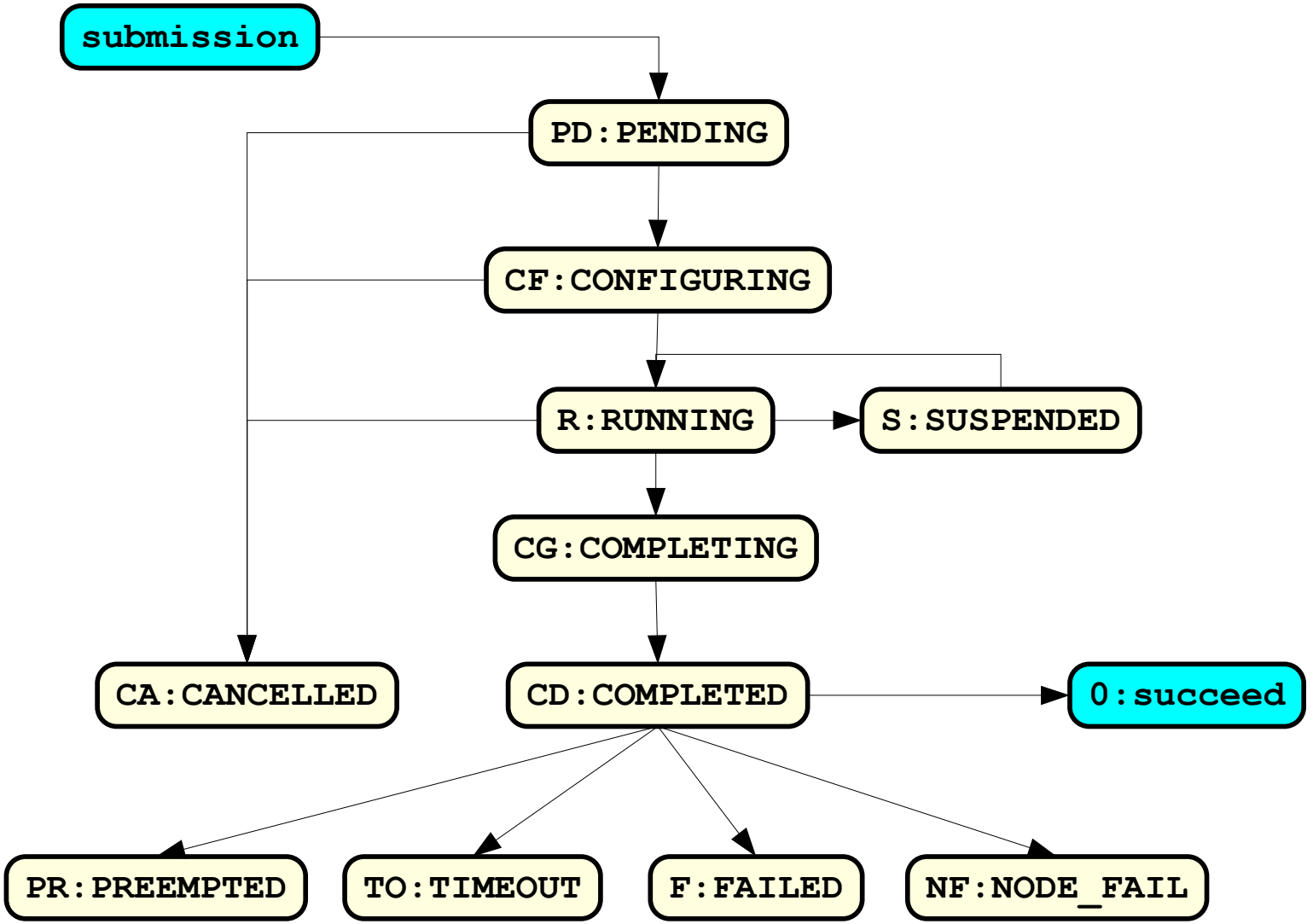
SLURM partition on Sango

partition name	maximum memory	number of nodes	number of cores (CPUs)	Job runtime (*)		Availability
				default	maximum	
compute	128 GiB	400	24	**8 hours**	**7 days**	unrestricted
largemem	512 GiB	20	24	**7 weeks**	**∞**	restricted (**)
phi	128 GiB (16GiB/cd.)	3 (4 cards/nd.)	24 (4 x 61)	2 days	2 days	restricted
datacp	64 GiB	2	24	8 hours	8 hours	unrestricted
gpu	128 GiB (2x12GiB/cd.)	3 (4 cards/nd.)	24 (4 x 2 x 2496)	8 hours	2 days	restricted

(*) jobs that do not specify their walltime using the --time= option will have the default value as their walltime. The walltime value can be set up to the maximum value

(**) priority should be given to genomics computation with high memory and very long computation time, and to jobs requiring more than 128 GiB of memory

SLURM at OIST
Job state simplified diagram



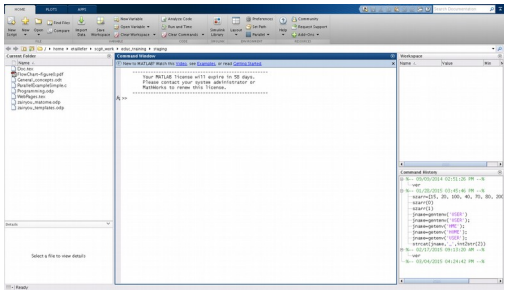
SLURM at OIST commands

- **srun**
  - submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements, including: minimum and maximum node count, processor count, specific nodes to use or not use, and specific node characteristics (so much memory, disk space, certain required features, etc.). A job can contain multiple job steps executing sequentially or in parallel on independent or shared nodes within the job's node allocation.

```
john-doe@sango-login1 ~ $ srun -c 1 --mem=10m date; sleep 10; date
Wed Mar  4 14:41:19 JST 2015
Wed Mar  4 14:41:29 JST 2015
john-doe@sango-login1 ~ $
```

- Notice that srun will not return the command prompt until the command execution has finished
- This is the recommended way to run a command from a login node

```
john-doe@jdws ~ $ ssh -X sango
john-doe@sango-login1 ~ $ module load matlab
john-doe@sango-login1 ~ $ srun -c 4 --mem=8g --x11=last matlab
```



SLURM commands

- **salloc**

allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. **The shell is then used to execute srun commands to launch parallel tasks.**

- **scontrol**

administrative tool used to view and/or modify Slurm state. Note that many scontrol commands can only be executed as user root.

```
john-doe@sango-login1 ~ $ salloc -n 48 --mem=32g
salloc: Granted job allocation 653
john-doe@sango0189 ~ $ scontrol show job 653
JobId=653 Name=bash
   UserId=john-doe(516) GroupId=oist(500)
   Priority=4294901703 Nice=0 Account=john-doe QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 ExitCode=0:0
   RunTime=00:00:04 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2015-03-04T12:51:45 EligibleTime=2015-03-04T12:51:45
   StartTime=2015-03-04T12:51:45 EndTime=Unknown
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=compute AllocNode:Sid=sango:16440
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=sango[0189,0191]
   BatchHost=sango0189
   NumNodes=2 NumCPUs=24 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=0
   MinCPUsNode=1 MinMemoryNode=8G MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=0 Contiguous=0 Licenses=(null) Network=(null)
   Command=(null)
   WorkDir=/home/j/john-doe

john-doe@sango0189 ~ $ exit
exit
salloc: Relinquishing job allocation 653
salloc: Job allocation 653 has been revoked.
john-doe@sango-login1 ~ $
```

SLURM commands

- **sbatch**
  - submit a job script for execution.

```
#!/bin/bash

#SBATCH --job-name=job_script
#SBATCH --partition=compute
#SBATCH --mem-per-cpu=1G
#SBATCH --ntasks=1

${HOME}/bin/myprog
```

```
john-doe@sango-login1 ~ $ sbatch job_script.slurm
Submitted batch job 650
```

SLURM commands

- **squeue**
  - report the state of partitions and nodes managed by SLURM. It has a wide variety of filtering, sorting, and formatting options.

```
john-doe@sango-login1 ~ $ squeue -p partition
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  618   compute     test john-doe  R       0:24      1 sango0111
```

Customized output for personalized reports:

```
john-doe@tombo-login1:~> squeue -u ${USER} -o "%A %P %u %t %M %D %C %N %j"
JOBID PARTITION USER ST TIME NODES CPUS NODELIST NAME
1093010 largemem john-doe R 8-21:18:26 1 1 sango40208 JB1
1093011 largemem john-doe R 8-21:18:26 1 1 sango40301 JB2
1093017 largemem john-doe R 8-21:18:05 1 1 sango40302 JB4
1093018 largemem john-doe R 8-20:24:03 1 1 sango40308 JB5
1097189 largemem john-doe R 8-01:31:47 1 1 sango40207 JB3
```

SLURM commands

- **scancel**
  - cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

```
john-doe@sango-login1 ~ $ scancel 661
john-doe@sango-login1 ~ $
```

Cancel ranges of job tasks from job array

```
john-doe@tombo-login1:~> scancel 867_[4-34]
john-doe has registered the job 867_[4-34] for deletion
```

SLURM commands

- **sinfo**
  - report the state of partitions and nodes managed by SLURM. It has a wide variety of filtering, sorting, and formatting options.

```
john-doe@sango-login1 ~ $ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
largemem     up   infinite     16    mix sango[40102-40103,...]
largemem     up   infinite      2  alloc sango[40101,40201]
largemem     up   infinite      2   idle sango[40104,40304]
phi          up 2-00:00:00      3   idle sango[20101,20201,20301]
gpu          up 2-00:00:00      1    mix sango30201
gpu          up 2-00:00:00      2   idle sango[30101,30301]
datacp       up    8:00:00      2   idle sango-mover[1-2]
compute*     up 7-00:00:00      1 drain* sango10112
compute*     up 7-00:00:00      2   drng sango[10102,10802]
compute*     up 7-00:00:00    239    mix sango[10101,10103-10105,...]
compute*     up 7-00:00:00    143  alloc sango[10114,10205,...]
compute*     up 7-00:00:00     15   idle sango[10106-10107,...]
longrun      up   infinite      2   idle sango-monitor[1-2]
```

SLURM commands

- **sstat**
  - report job or job step accounting information about user's active jobs.

```
john-doe@sango-login1 ~ $ sstat -e
AveCPU             AveCPUFreq         AveDiskRead        AveDiskWrite
AvePages           AveRSS             AveVMSize          ConsumedEnergy
ConsumedEnergyRaw  JobID              MaxDiskRead        MaxDiskReadNode
MaxDiskReadTask    MaxDiskWrite       MaxDiskWriteNode   MaxDiskWriteTask
MaxPages           MaxPagesNode       MaxPagesTask       MaxRSS
MaxRSSNode         MaxRSSTask         MaxVMSize          MaxVMSizeNode
MaxVMSizeTask      MinCPU             MinCPUNode         MinCPUTask
Nodelist           NTasks             Pids               ReqCPUFreq

john-doe@sango-login1 ~ $ sstat -j 1242684 --format=MaxRSS,MaxRSSNode,MinCPUNode
    MaxRSS MaxRSSNode MinCPUNode
---------- ---------- ----------
     5036K sango11402 sango11402
```

Tombo as a HPC test environment during the software week

- Use Tombo environment with your OIST account (same credentials as for tida) for the training hands-ons:

tombo.oist.jp

```
john@johnws ~ $ ssh john-doe@tombo.oist.jp
jhon-doe@tombo.oist.jp's password:
jhon-doe@tombo-login2 ~ $ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
compute*     up 7-00:00:00     96   idle tombo[20101-20116,
20201-20216,20301-20316,20401-20416,20501-20516,20601-20616]
```

Getting started HPC at OIST

- Accounts
  - Connect to the cluster
- Use the clusters
  - Transfer files
  - Software
  - Run Job
  - Manage Job
- Best practice
  - Computing work-flow
  - General rules

Accounts

Connect to the cluster

- Any OIST member having a user account may use the HPC resources
  - Check gosa.oist.jp
- Access Sango using SSH from a terminal
  - Linux terminal, OSX Terminal, or Windows Putty/TeraTerm
  - GUI enable (Linux terminal)
  - GUI enable (OSX Terminal)

```
john-doe@jdws ~ $ ssh sango
john-doe@sango-login3 ~ $
```

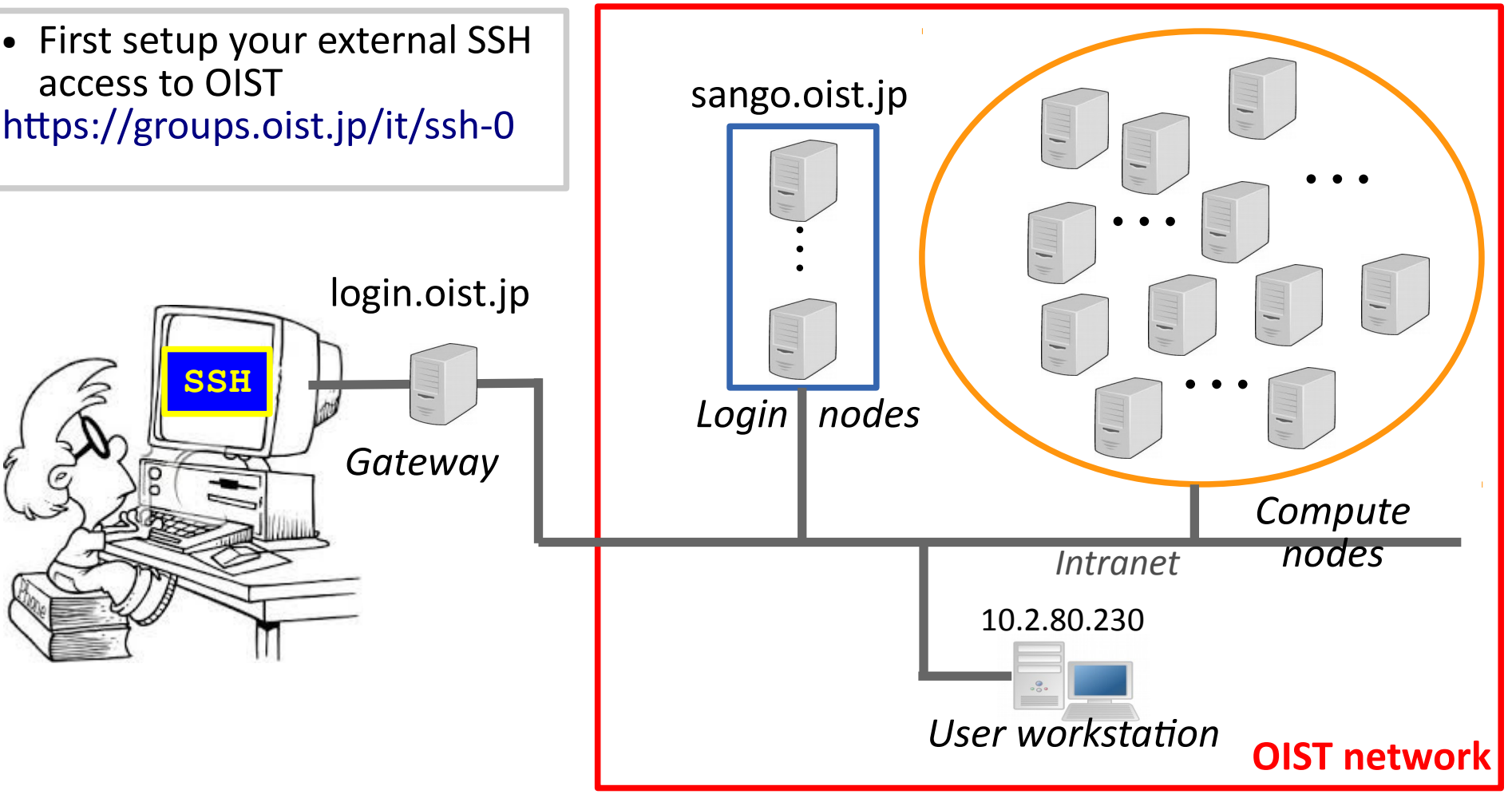
```
john-doe@jdws ~ $ ssh -X sango
```

```
john-doe@jdws ~ $ ssh -Y sango
```

Accounts
Access from outside

- First setup your external SSH access to OIST

https://groups.oist.jp/it/ssh-0



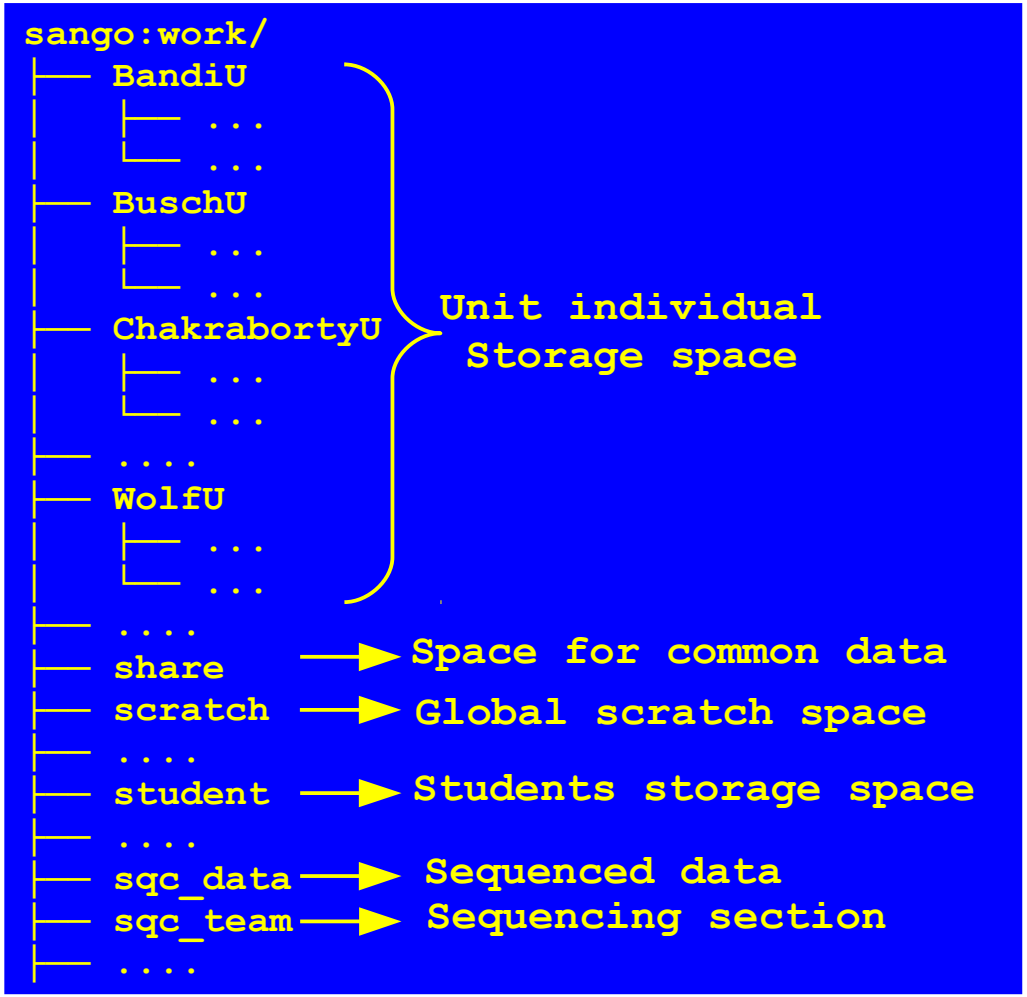

```
john-doe@housews ~ $ ssh john-doe@login.oist.jp
john-doe@loginc02 ~ $ ssh sango
john-doe@sango-login3 ~ $
```

Use the cluster
Transfer data

- Use SMB protocol to connect to **`sango.oist.jp:/bucket`** and **`sango.oist:/work`** from Linux, OSX, or Windows

  https://groups.oist.jp/it/research-storage

- Use **scp** command from a terminal to transfer file from/to your workstation, **`sango:/bucket`**, **`sango:/home`**, or **`sango:/work`**

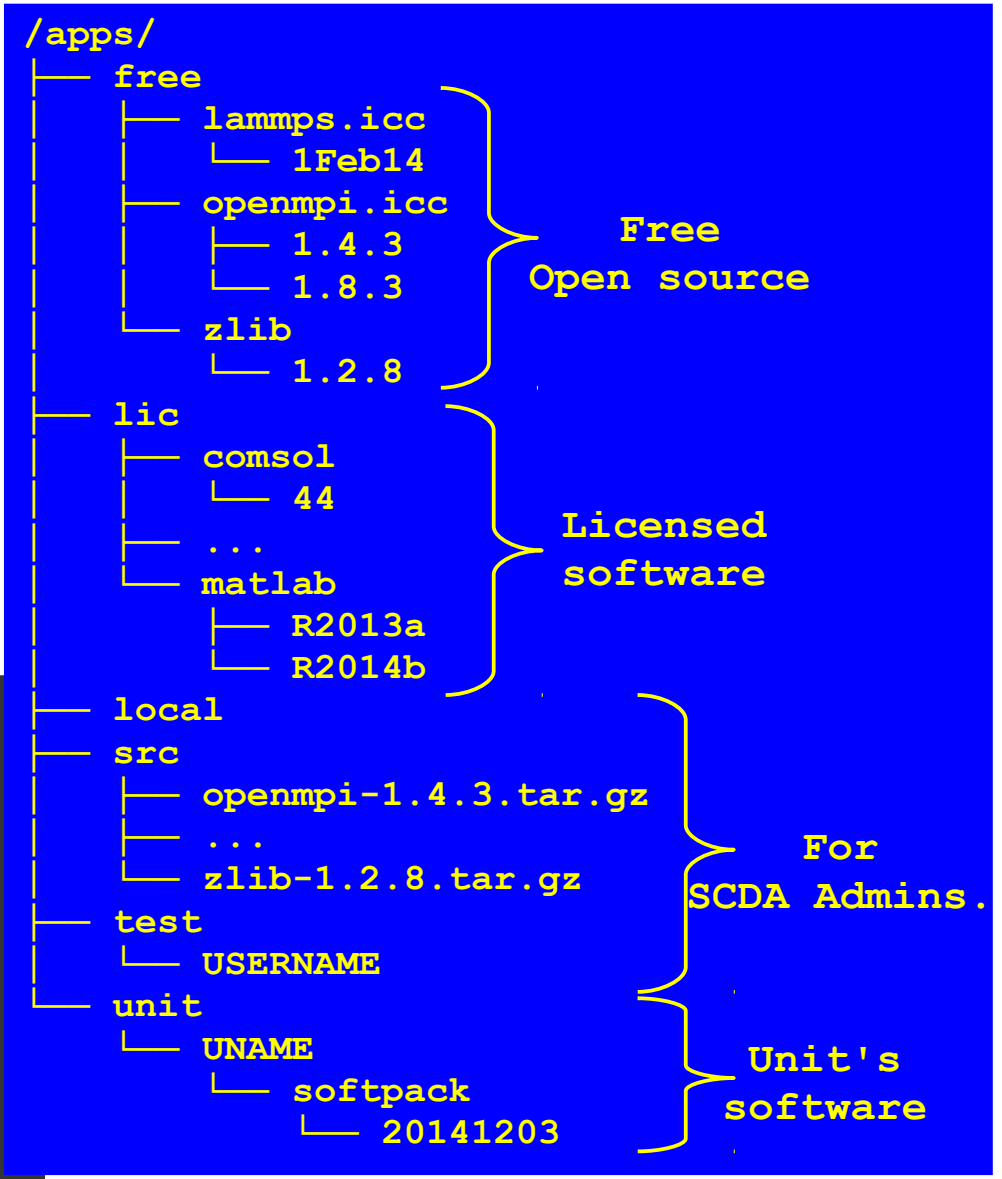



```
john-doe@jdws ~ $ scp big_data.tar.gz john-doe@sango.oist.jp:/work/DoeU/john-doe
```

Use the cluster
Software available

- Software not available on the cluster OS are provided through the Environment modules
- And, are deployed in the /apps directory
- Licensed software at OIST

  https://groups.oist.jp/it/oist-software




```
john-doe@sango-login1 ~ $ module available
----------- /etc/modulefiles ---------------
....
R/2.15.3                    comsol/43b    ....
R/3.1.0                     comsol/44
R/3.1.1                     cpptest/1.1.2 ....
SuiteSparse/4.4.2           cuda/6.0.37   ....
....
john-doe@sango-login1 ~ $ module load SuiteSpars
john-doe@sango-login1 ~ $ module list
Currently Loaded Modulefiles:
  1) intel/2015            2) SuiteSparse/4.4.2
john-doe@sango-login1 ~ $ module purge
```

Use the cluster
Run simple computation

- Submit job scripts using **`sbatch`**

test_ucsc.slurm

```
#!/bin/bash

#SBATCH --job-name=test_ucsc
#SBATCH --partition=compute
#SBATCH -time=00:10:00
#SBATCH --mem-per-cpu=1G
#SBATCH --ntasks=1
#SBATCH --mail-user=john-doe@oist.jp
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --output=test_ucsc.o%j
#SBATCH --input=none
#SBATCH --error=test_ucsc.e%j

module load ucsc_utils/9999

goldenpath="/genefs/Gene-Mirrors/UCSC/WEBROOT/goldenPath"
gver="hg18"
for chrnum in $(seq 1 22) X Y; do
  echo "${gver} chr-${chrnum}"
  faCount ${goldenpath}/${gver}/chromosomes/chr${chrnum}.fa.gz
done
```

- Best way to run computation on the cluster
- Computation is queued into the scheduler, so it will run even after logout or termination of the SSH session

```
john-doe@sango-login3 ~ $ sbatch test_ucsc.slurm
```

Use the cluster
Run simple computation

- Open interactive session using **`srun`**

```
john-doe@sango-login1 ~ $ srun --partition=compute --mem-per-cpu=1G --ntasks=1 --pty bash
john-doe@sango10101 ~ $ module load ucsc_utils/9999
john-doe@sango10101 ~ $ export GENE_DIR="/genefs/Gene-Mirrors/UCSC/WEBROOT/goldenPath"
john-doe@sango10101 ~ $ faCount ${GENE_DIR}/hg18/chromosomes/chr1.fa.gz
#seq len     A    C    G    T    N    cpg
...
john-doe@sango10101 ~ $ exit
exit
john-doe@sango-login1 ~ $
```

Typical use cases:

- Develop, compile, test programs, batch scripts or computation pipelines
- Run interactive software

Drawback

- The computation is lost if you lose the SSH connection

Use the cluster
Run simple computation

- Submit computation from the command line using **`srun`**

```
john-doe@sango-login2 ~ $ module load ucsc_utils/9999
john-doe@sango-login2 ~ $ export GENE_DIR="/genefs/Gene-Mirrors/UCSC/WEBROOT/goldenPath"
john-doe@sango-login2 ~ $ srun --partition=compute --mem-per-cpu=1G -n 1 \
   --output="job%j.out" --input=none --error="job%j.err" \
   "faCount ${GENE_DIR}/hg18/chromosomes/chr1.fa.gz" &
```

Don't forget to put in background

Typical use cases:

- Quickly test a series of programs or scripts
- Run very short computation

Drawbacks

- The computation is lost if you log out or lose the SSH connection
- Can be used only with independent computation

Use the cluster
Run parallel computation

- Main types of parallel jobs
  - Single node
    - ✔ Open MP
    - ✔ Multi thread
  - Multiple nodes
    - ✔ Job array
    - ✔ MPI
    - ✔ Hybrid MPI+multithread
  - GPGPU

Use the cluster
Run parallel computation

- OpenMP and Multi-thread jobs (using **`sbatch`**)

job_script_openmp.slurm

```
#!/bin/bash

#SBATCH --job-name=omp_job
#SBATCH --mail-user=john-doe@oist.jp
#SBATCH --partition=compute
#SBATCH --mem=22G
#SBATCH --cpus-per-task=22
#SBATCH --ntasks=1

${USER}/bin/openmp_program
```

Note that **`OMP_NUM_THREADS`** is set to **`$SLURM_CPUS_PER_TASK`** before the computation code block

job_script_mp.slurm

```
#!/bin/bash

#SBATCH --job-name=mp_job
#SBATCH --mail-user=john-doe@oist.jp
#SBATCH --partition=compute
#SBATCH --mem=22G
#SBATCH --cpus-per-task=22
#SBATCH --ntasks=1

${USER}/bin/mp_program \
        -n $SLURM_CPUS_PER_TASK
```

```
john-doe@sango-login3 ~ $ sbatch job_script_openmp.slurm
Submitted batch job 850
john-doe@sango-login3 ~ $ sbatch job_script_mp.slurm
Submitted batch job 851
```

Use the cluster
Run parallel computation

- Job array (1. using **`sbatch`** and **`runarray`**)

job_script.slurm

```
#!/bin/bash

#SBATCH --job-name=Montecarlo
#SBATCH --partition=compute
#SBATCH --mem-per-cpu=200M

module purge
module load gsl intel

runarray -r 1-100 Montecarlo.sh
```

Montecarlo.sh

```
#!/bin/bash

#SBATCH --time=1:0:0
#SBATCH --mem-per-cpu=1G
#SBATCH -partition=compute
#SBATCH --output=Montecarlo-%A_%a.out
#SBATCH --error=Montecarlo-%A_%a.err

module purge

tid=$SLURM_ARRAY_TASK_ID

DATASET=dataset.${tid}
OUTFILE=result.${tid}

Montecarlo ${DATASET} > ${OUTFILE}
```

```
john-doe@sango-login3 ~ $ sbatch job_script.slurm
```

Use the cluster
Run parallel computation

- Job array (2. using **`#SBATCH --array=`** or **`sbatch --array=`**)

job_script.slurm

```
#!/bin/bash

#SBATCH --job-name=Montecarlo
#SBATCH --partition=compute
#SBATCH --array=1-100%4
#SBATCH --mem-per-cpu=1G
#SBATCH --output=Montecarlo-%A_%a.out
#SBATCH --error=Montecarlo-%A_%a.err

module purge
module load gsl intel

tid=$SLURM_ARRAY_TASK_ID
DATASET=dataset.${tid}
OUTFILE=result.${tid}

Montecarlo ${DATASET} > ${OUTFILE}
```

%4: Number of tasks simultaneously running (**MANDATORY**)

```
john-doe@sango-login3 ~ $ sbatch job_script.slurm
john-doe@sango-login3 ~ $ scancel <JOBID>_[50-100]
```

Use the cluster
Run parallel computation

- MPI job (using **`sbatch`**)

OpenMPI: job_script_ompi.slurm

```
#!/bin/bash

#SBATCH --job-name=ompi_job
#SBATCH --mail-user=john-doe@oist.jp
#SBATCH --partition=compute
#SBATCH --mem-per-cpu=2G
#SBATCH --ntasks=128

module purge
module load openmpi.gcc/1.8.6

srun --mpi=pmi2 myprogram
```

Intel MPI: job_script_impi.slurm

```
#!/bin/bash

#SBATCH --job-name=impi_job
#SBATCH --mail-user=john-doe@oist.jp
#SBATCH --partition=compute
#SBATCH --mem-per-cpu=2G
#SBATCH --ntasks=128

module purge
module load intel.mpi/5.0.3.048

mpiexec.hydra -bootstrap slurm -n ${SLURM_NTASKS} myprogram
```

Use the cluster
Run parallel computation

- Hybrid MPI+openmp job (using **`sbatch`**)

Hybrid hello world example in C
`mixed_hello.c`

```c
#include <stdio.h>
#include <mpi.h>
#include <omp.h>

int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int iam = 0, np = 1;

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  #pragma omp parallel default(shared) private(iam, np)
  {
    np = omp_get_num_threads();
    iam = omp_get_thread_num();
    printf("Hello from thread %d out of %d from process %d out of %d on %s\n",
           iam, np, rank, numprocs, processor_name);
  }

  MPI_Finalize();
}
```

Use the cluster
Run parallel computation

- Hybrid MPI+mp job (using **`sbatch`**)

Compilation

```
$ module load openmpi.gcc/1.8.6
$ mpicc -fopenmp mixed_hello.c -o mixed_hello
```

SLURM script

```
#!/bin/bash

#SBATCH --job-name=calpi_MPI
#SBATCH --mail-user=%u@oist.jp
#SBATCH --partition=compute
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=200m

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
module load openmpi.gcc/1.8.6

srun --mpi=pmi2 ./mixed_hello
```

Execution output

```
 $ sbatch job_MPI_hybrid.slurm
Submitted batch job 545323

 $ cat slurm-545323.out
Hello from thread 0 out of 2 from process 5 out of 8 on sango11019
Hello from thread 1 out of 2 from process 5 out of 8 on sango11019
Hello from thread 1 out of 2 from process 3 out of 8 on sango11018
Hello from thread 1 out of 2 from process 2 out of 8 on sango11017
Hello from thread 1 out of 2 from process 4 out of 8 on sango11019
Hello from thread 0 out of 2 from process 1 out of 8 on sango11016
Hello from thread 1 out of 2 from process 1 out of 8 on sango11016
Hello from thread 1 out of 2 from process 6 out of 8 on sango11020
Hello from thread 0 out of 2 from process 6 out of 8 on sango11020
Hello from thread 0 out of 2 from process 7 out of 8 on sango11020
Hello from thread 1 out of 2 from process 7 out of 8 on sango11020
Hello from thread 0 out of 2 from process 3 out of 8 on sango11018
Hello from thread 0 out of 2 from process 2 out of 8 on sango11017
Hello from thread 1 out of 2 from process 0 out of 8 on sango11016
Hello from thread 0 out of 2 from process 0 out of 8 on sango11016
Hello from thread 0 out of 2 from process 4 out of 8 on sango11019
```

Use the cluster
GPGPU computing

```
john-doe@sango-login3:~> srun --partition=gpu --gres=gpu:1 nvidia-smi
Wed Sep  9 11:29:12 2015
+------------------------------------------------------+
| NVIDIA-SMI 352.21     Driver Version: 352.21         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:04:00.0     Off |                    0 |
| N/A   55C    P0   145W / 149W |    120MiB / 11519MiB |     92%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:05:00.0     Off |                    0 |
| N/A   42C    P0   147W / 149W |    120MiB / 11519MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:84:00.0     Off |                    0 |
| N/A   57C    P0   146W / 149W |    120MiB / 11519MiB |     93%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 0000:85:00.0     Off |                    0 |
| N/A   44C    P0   148W / 149W |    120MiB / 11519MiB |     97%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla K80           On   | 0000:8A:00.0     Off |                    0 |
| N/A   26C    P8    26W / 149W |     22MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla K80           On   | 0000:8B:00.0     Off |                    0 |
| N/A   34C    P8    30W / 149W |     22MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla K80           On   | 0000:8E:00.0     Off |                    0 |
| N/A   26C    P8    27W / 149W |     22MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla K80           On   | 0000:8F:00.0     Off |                    0 |
| N/A   30C    P8    30W / 149W |     22MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     12676    C   ...aillefumier/codes/McCuda/KagomeCudaDouble    96MiB |
|    1     12676    C   ...aillefumier/codes/McCuda/KagomeCudaDouble    96MiB |
|    2     12676    C   ...aillefumier/codes/McCuda/KagomeCudaDouble    96MiB |
|    3     12676    C   ...aillefumier/codes/McCuda/KagomeCudaDouble    96MiB |
+-----------------------------------------------------------------------------+
```

- 3 nodes (24-cores, 128GiB) equipped with 4 nVidia Tesla K80 (dual GPU) are available ← **`nvidia-smi`** output from one node
- Request is needed to use GPU computing: https://groups.oist.jp/scs/request-accessing-sango-restricted-shared-resources
- Requires **`--partition=gpu`** and **`--gres=gpu:1`** SLURM parameters
- GPU-ready software available on Sango, see: **`module avail`**

Use the cluster

GPGPU computing (example)

https://groups.oist.jp/scs/gpgpu-computation-using-matlab

Matlab script **`matlab_prog_gpu`** making use of gpu functions(*)

```
gpuDeviceCount
gpuDevice

A = ones(10, 'single', 'gpuArray');
B = 5 .* eye(10, 'single', 'gpuArray');
C = A * B;
C_host = gather(C);

C_host
```

Slurm script to run the matlab code

```
#!/bin/bash

#SBATCH --job-name=matlab-gpu
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH --time=00:10:00
#SBATCH --input=none
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err

module load matlab/R2015a
module load cuda/7.0.28

matlab_exe="matlab -nosplash -nodisplay -nojvm -nodesktop"

${matlab_exe} -r "matlab_prog_gpu, exit"
```

(*) List of matlab gpu functions:
http://www.mathworks.com/help/distcomp/graphics-processing-unit-gpu-computing.html

Use the cluster

GPGPU computing (typical dev. workflow)

- Reserve a node using **`srun`**
- Load cuda module
- Compile with **`nvcc`**
- Run some tests

- Write SLURM script
- Submit to scheduler

job_script_gpu.slurm

```
#!/bin/bash

#SBATCH --job-name=matrixMul_gpu
## 5min walltime
#SBATCH --time=00:05:00
## use Sango gpu partition
#SBATCH --partition=gpu
## use one GPU
#SBATCH --gres=gpu:1
## job input/output
#SBATCH --input=none
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err

module load cuda/7.0.28
/apps/local/training/samples/matrixMul.gpu
```

```
sango-login3$ srun --partition=gpu --gres=gpu:1 --pty /bin/bash
sango30201$ module load cuda/7.0.28
sango30201$ export TSMPLDIR="/apps/local/training/samples"
sango30201$ nvcc --include-path ${TSMPLDIR} \
                 -o matrixMul.gpu ${TSMPLDIR}/matrixMul.cu
sango30201$ ls -lh matrixMul.gpu
-rwxr-xr-x 1 john-doe oist 521K Aug  1 09:26 matrixMul.gpu
```

```
sango30201$ ./matrixMul.gpu
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Tesla K80" with compute capability 3.7

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 229.09 GFlop/s, Time= 0.572 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

Use the cluster

Manage submitted jobs

https://groups.oist.jp/scs/getting-started#gs_use5

- List running jobs

```
john-doe@sango-login4 ~ $ squeue -u ${USER}
```

- Check status of a job

```
john-doe@sango-login4 ~ $ squeue <Job-ID>
```

- Cancel a jobs

```
john-doe@sango-login4 ~ $ scancel <JID> ... <JID>
```

- Reason of waiting

```
john-doe@sango-login4 ~ $ squeue -o '%r %R' <Job-ID>
```

- Cancel all pending jobs in compute

```
john-doe@sango-login4 ~ $ scancel -t PENDING \
  -u ${USER} -p compute
```

- Sango usage summary

```
john-doe@sango-login4 ~ $ sango_usage
```

- Check memory, CPU, disk usage of a running job

```
john-doe@sango-login4 ~ $ sstat -j <Job-ID>
```

Best Practices
computing work-flow

- Persistent/permanent data are placed in **`/bucket/<unitU>`**
- Configuration files, scripts and programs placed in **`/home`**
- Computing data and and program in **`/work/<unitU>`** and **`/apps/unit/<unitU>`**, resp.

Preliminary step if you have data on bucket:

(0) if you have computing data on **`/bucket/<unitU>`** copy them to using **`srun`** or a batch script (https://groups.oist.jp/scs/getting-started#gs_use4)

Inside your SLURM batch script:

(1) Create (**`mkdir`**) **`$WORK_SCRATCH`** and copy data from **`/work/<unitU>`** and **`$HOME`** to **`$WORK_SCRATCH`**

(2) Perform computation under **`$WORK_SCRATCH`**

(3) Copy computation results into **`/work/<unitU>`**
(then to **`/bucket/<unitU>`** if backup or archiving is needed)

(4) Delete **`$WORK_SCRATCH`**

`$WORK_SCRATCH` is set to **`/work/scratch/$USER/$SLURM_JOB_ID`** upon job starting

Best Practices
general rules

- Do not run compute jobs or heavy applications (MATLAB, etc.) on the login nodes; always use SLURM commands srun, salloc, or sbatch
- Always specify the amount of memory and time that would be used by your job in the **`--mem=`** (openMP/multithread or serial jobs), **`--mem-per-cpu=`** (MPI jobs), and **`--time=`** parameters of the SLURM commands
- Remove temporary data or intermediate files generated by your computation. Use global scratch space for temporary data **`/work/scratch/${USER}/<job-id>`** (available through the variable **`WORK_SCRATCH`**)
- Do not submit several thousand or hundred of thousands of long running jobs at the same time because it will prevent other users from being able to use the cluster. If you really need to run so many jobs, please contact us first (it-help@oist.jp or during Open Hours).
- Use **`${HOME}/slurm_<job-id>.log`** generated after job completion to obtain effective time and memory resource consumption of the job, and then use those information to optimize the **`--mem=`**, **`--mem-per-cpu=`** and **`--time=`** parameters
(more information on: https://groups.oist.jp/scs/getting-started#gs_use6)

Any question ?

Thank you
for your time

SCDA Members:
Francesca Tartaglione
Mathieu Taillefumier
Eddy Taillefer
Namiko Nagao
Tim Dyce

http://groups.oist.jp/scs

it-help@oist.jp

SCDA Open Hours

- Purpose:
  - Time slot allocated for HPC and scientific computing support
  - Help HPC users to effectively use the cluster for their scientific computing and research activity
  - Consulting about HPC scientific computing and data analysis
- When:
  - Every weekday, 15h30 to 17h30
- Place:
  - Lab2, B648