

CSC Supercomputing Environment

A one-day introduction to CSC's services and their utilization

T. Zwinger, T. Bergman

CSC – Tieteen tietotekniikan keskus Oy
CSC – IT Center for Science Ltd.

Contents

- CSC's server and the Meta-computer environment (*T. Zwinger*)
 - louhi, vuori, murska, hippu
 - Login, software utilization, licenses
 - setup, disk locations, storage strategies
 - Batch jobs (LSF-SLURM, SLURM, PBS)
- Scientists Interface (SI) (*T. Bergman*)
- Compiling and porting (*T. Bergman*)
 - Compiler suites: C, C⁺⁺, F90
 - Some words on libraries

CSC Server



- CSC computing servers
- Cray XT 4/5
louhi.csc.fi
- HP CP4000 ProLiant superclusters
murska.csc.fi and
vuori.csc.fi
- HP DL 785 G5 ProLiant server pair
hippu.csc.fi



CSC Server: purpose



server	purpose
louhi	Large (up to 2048 cores) parallel runs ¹⁾
murska	Smaller (<256 cores) parallel and serial runs ²⁾
vuori	Smaller (<144 cores) parallel and serial runs ²⁾
hippu	Interactive work/runs

¹⁾ needs testing for scalability of the application

²⁾ needs testing of scalability for jobs larger than 128 cores

CSC Server: login

- From **UNIX/Linux/Darwin** systems using ssh command
 - X11 forwarding with option -X
 - Different login ID's with option -l login

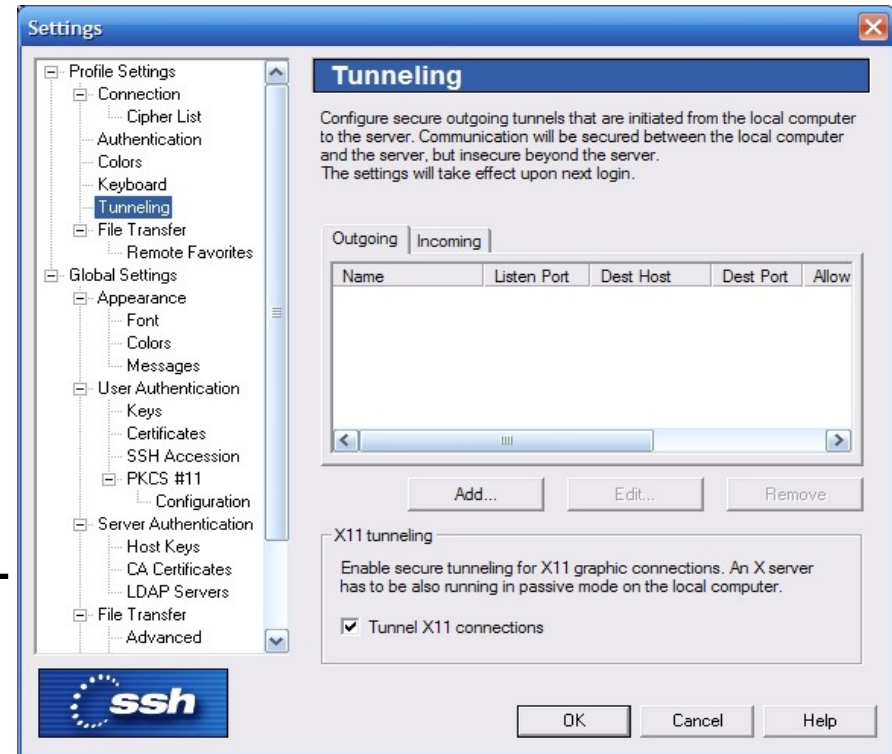
```
> ssh -l yourid -X vuori.csc.fi
```

```
> ssh -X yourid@vuori.csc.fi
```

CSC Server: login



- From **Windows** systems using ssh client program¹⁾
 - Needs separate X11 emulator²⁾
 - Most clients need to explicitly enable X11 tunneling



1) for instance OS solution PuTTY

2) for instance OS solution Xming

CSC Server: file transfer



- From **UNIX/Linux** systems using `scp` command

```
> scp [-r] name  
yourid@vuori.csc.fi: '/path/on/vuori'
```

- From **Windows** systems using a certain program
 - E.g., WinScp
- Via Scientist's Interface (see later)

CSC Server: Using the shell



- By default: home directory
- Listing of contents:

```
> ls [-a -l -t -r]
```

- Changing directory:

```
> cd [../relativepath]  
[ /absolute/path/ ]
```

./ refers to current, ../ to upper level

- Where am I?

```
> pwd
```

CSC Server: Using the shell



- Removing a file¹⁾:

```
> rm [-r -f] filename
```

- Moving a file/directory:

```
> mv [/old/dir/]oldname  
    [/new/dir/]newname
```

- Copying files:

```
> cp [-r] [/old/dir/]oldname  
    [/new/dir/][newname]
```

¹⁾ Files in UNIX file-systems cannot be restored thereafter!

CSC Server: Using the shell



- File permissions:

```
> ls -l
total 6
-rwxrwxrwx 1 zwinger zwinger 487 2010-09-10 09:32 hello.f90
-rwxrwxrwx 2 zwinger zwinger 568 2010-09-10 14:21 hello_world.c
-rwxrwxrwx 2 zwinger zwinger 775 2010-09-10 13:59 hello_world_MPI.c
drwx----- 1 zwinger zwinger  0 2010-09-10 14:23 hippu
drwx----- 1 zwinger zwinger  0 2010-09-10 14:22 louhi
drwx----- 1 zwinger zwinger  0 2010-09-10 14:22 murska
drwx----- 1 zwinger zwinger  0 2010-09-10 14:23 vuori
```

[type][user][group][others]

r=read, w=write, x=execute (access)

CSC Server: Using the shell



- **Guide: Using the data services at CSC**
 - Linux basics for CSC
 - Linux tools for working with data
 - Managing files in CSC environment
 - Moving data between CSC and local environment

CSC Server: software

CSC provides the most extensive selection of software applications in Finland

- 200 different ready-made programs available for research use in Finland

GAMS
FIDAP
Funcs
IMSL
SAS
CHARMM
GCG
ABAQUS/CAE
matlab
MSI

Software licenses for different fields of science, such as:

1. chemistry
2. geosciences
3. biosciences
4. physics
5. statistics
6. CFD
7. structural analysis
8. mathematics
9. scientific visualization

CSC Server: software licenses



- Different types of licenses
(see Software and Databases):
 - A** Only academic use
 - C** Academic and commercial use
 - G** Governmental
 - P** Public domain or free software license
 - R** Academic use and commercial use with a royalty fee

CSC Server: software licenses



license2.csc.fi

license1.csc.fi

license3.csc.fi



- License server**
- 3 redundant (FlexLM) license manager
 - single FlexLm license manager
 - reserved ports for server and daemon



CSC internal network

- Application server**
- software
 - license information in LM_LICENSE_FILE
 - UNIX user groups



vuori.csc.fi



hippu.csc.fi

CSC Server: software licenses



- License policy at CSC:
 - Often user has to sign a general license agreement
 - User has to comply with which she/he signed (e.g., publications, commercial use, ...)
 - Be a nice colleague and free unneeded licenses
 - Often limitations on licenses introduced

CSC Server: software setup

- Setup of software on the fly
- **module** command on all systems
- Listing of loaded modules:

```
> module list
```

- Listing of available modules:

```
> module avail
```

- Loading a module (e.g., Elmer, FEM):

```
> module load elmer/latest
```

CSC Server: software setup



- Unloading a module:

```
> module unload elmer/latest
```

- Switching between mutually excluding modules:

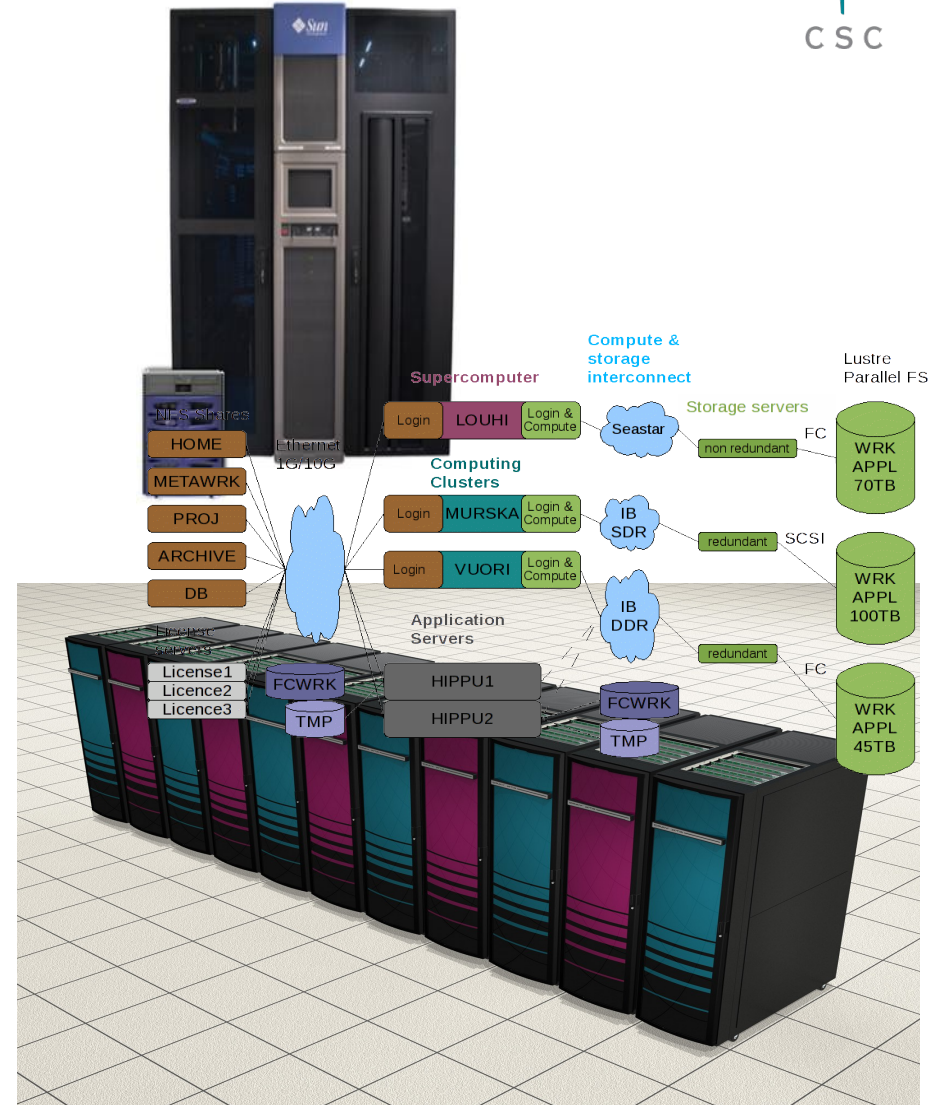
```
> module switch PrgEnv-pgi/10.2-0  
PrgEnv-gnu/4.4.4
```

Test: try to simply load PrgEnv-gnu/4.4.4 instead

The Metacomputer environment



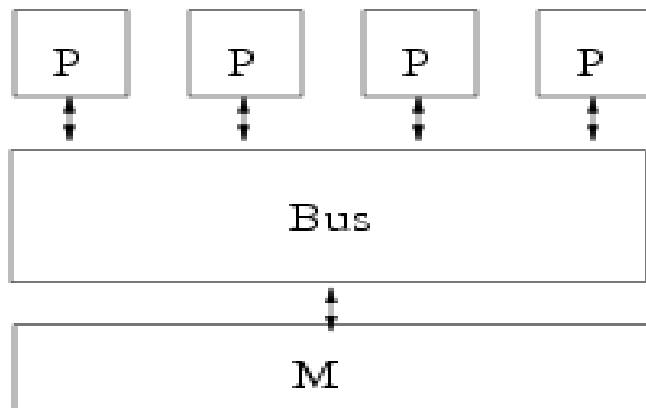
- CSC provides a to all servers common environment
- Helps in moving/storing/handling files and data



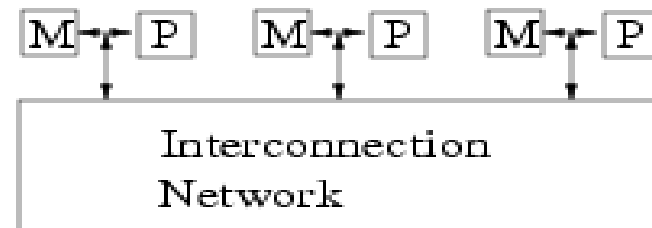
The Metacomputer environment: platforms



- A shared memory machine:
 - All memory accessible (with varying speed)
 - hippu is a SMP



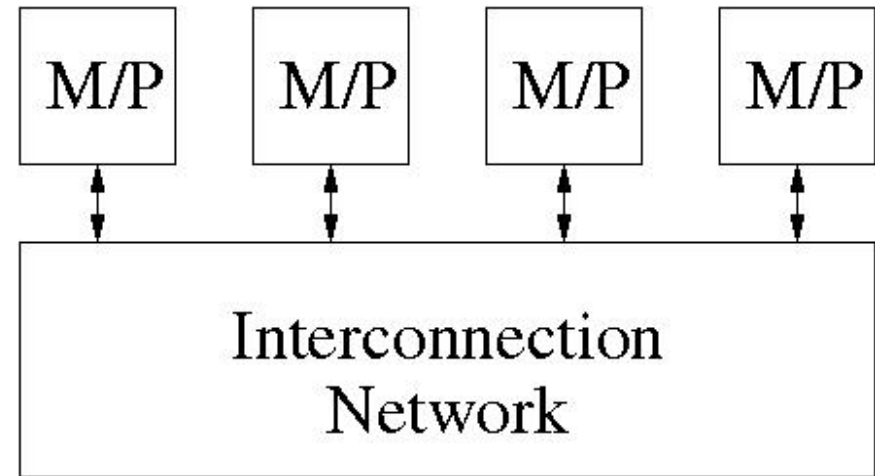
- A distributed memory machine:
 - local memory
 - Real MPP's died with T3e



The Metacomputer environment: platforms



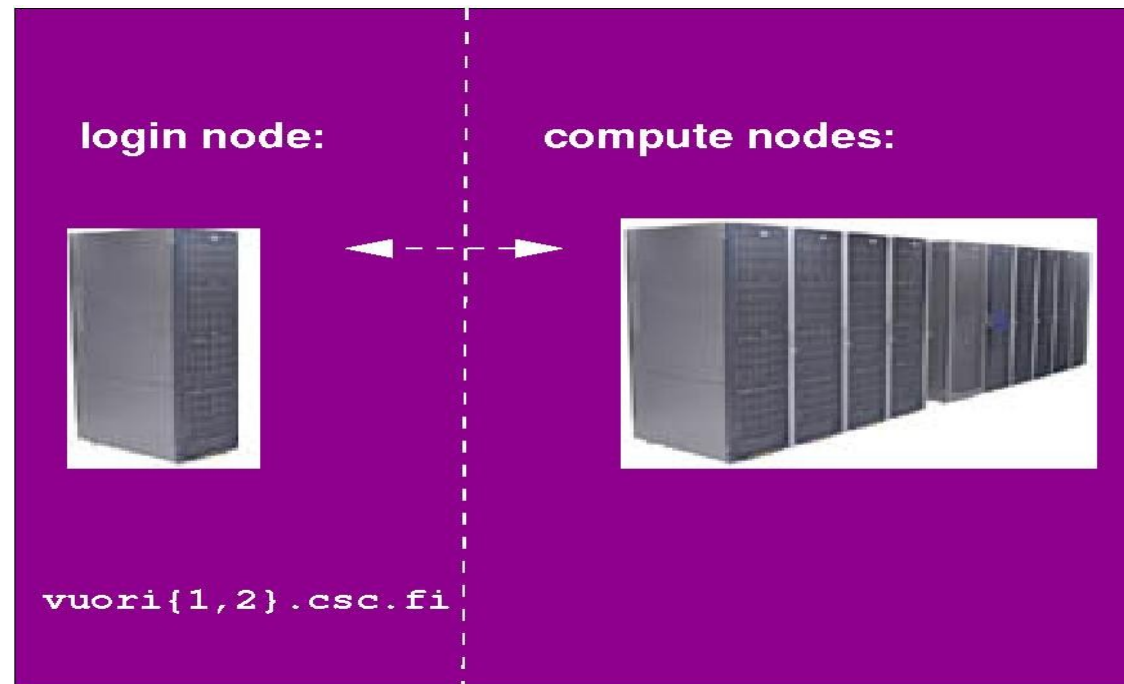
- A cluster is a connection of separate units (nodes) via a fast network
 - All larger CSC platforms (louhi, murska, vuori) are clusters in a general sense



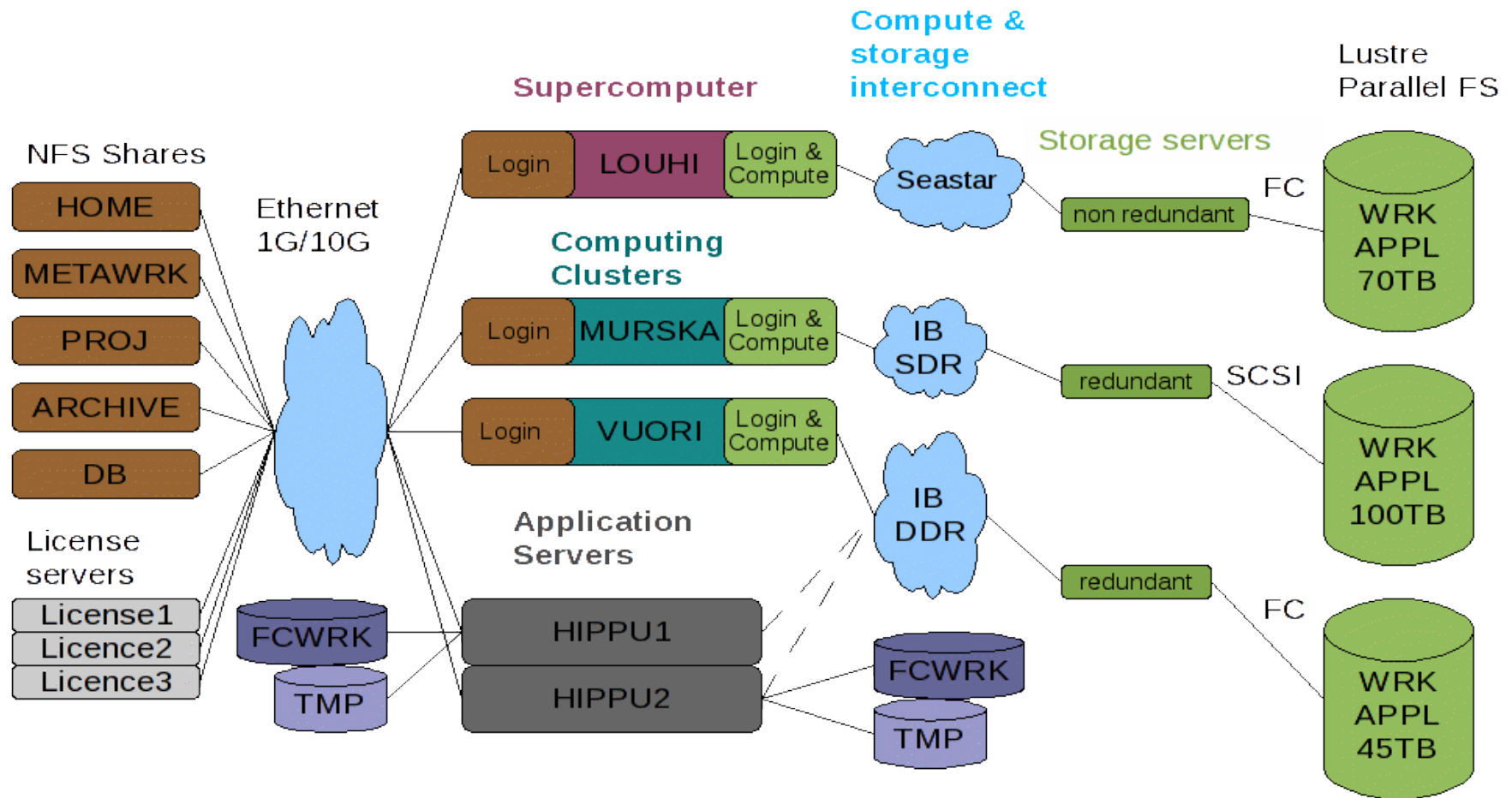
The Metacomputer environment: platforms



- CSC's clusters:
 - login nodes:
 - setup and submission of runs
 - very small interactive work
 - compute nodes:
 - number-crunching
 - batch jobs



The Metacomputer environment: directories



Graphics by T. Tervo

The Metacomputer environment: directories



- **Home directory:** `$HOME`
 - for storage of most important files
 - permanent, quotas (1GB)
 - backup
 - common visibility except compute nodes
- **User application directory:** `$USERAPPL`
 - server specific directory for user's own software installation
 - permanent
 - backup
 - visibility server specific (login & compute)

The Metacomputer environment: directories



- **Metawork directory**¹⁾: `$METAWRK`
 - project and input files (between servers)
 - quota (200 GB), storage time 30 days
 - no backup
 - common visibility except compute-nodes
- **Work directories**¹⁾: `$WRKDIR`
 - temporary data files
 - no quota, storage time 7 days
 - no backup
 - visibility server specific (login & compute)

¹⁾ on hippu: `$METAWRK` and `$WRKDIR` are the same

The Metacomputer environment: directories



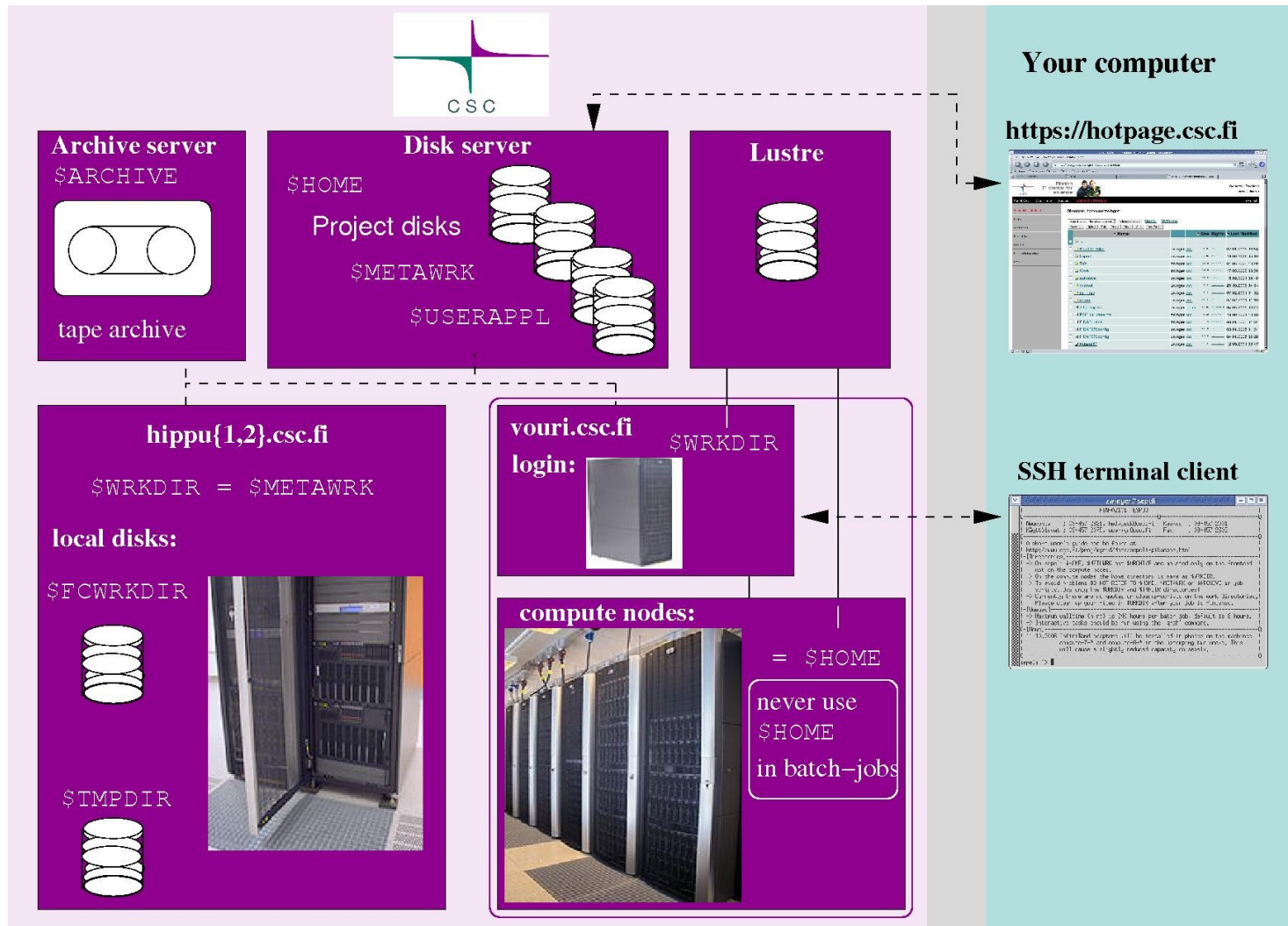
- **Temporary directories:** `$TMPDIR`
 - run-time storage
 - fast local disks to
 - no quota, storage time 24 hours
 - Visibility server specific (login- and compute)
- **Hippu work directories:** `$FCWRKDIR`
 - fast local workdirectory specific to hippu{1,2}
 - quota (150 GB), storage time 30 days
 - cross mounted on both machines

The Metacomputer environment: directories



- **Project directory:**
 - disk space assigned to special project
 - quota, storage time and location on request
 - backup
 - common visibility except compute-nodes
- **Archive directory:** `$ARCHIVE`
 - long time tape storage (very, very slow!)
 - quota (550 GB/10k files)
 - two tapes for redundancy
 - common visibility except compute-nodes

The Metacomputer environment: directories



The Metacomputer environment: resources



- **Disk quotas:**
 - Command `quota` gives the used and available disk space

```
zwinger@vuori2:~> quota
[_$ARCHIVE_usage_____]
      Online      Offline
      Used  Avail      Used  Avail
Files   122 10000     122   10000
Blocks  0.00G  0.45T     33.29G  0.45T
Grace period 1w
[_Projects_usage_____]
Size Used Avail on
 25G 24G 1.3G /fs/proj1/elmer
 10G 1.0G 9.0G /fs/proj1/ice2sea
[_$METAWRK_usage_____]
Size Used Avail on
200.0G 17713M 182.7G /fs/metawrk/zwinger
[_$HOME_usage_____]
Size Used Avail on
1024M 305M 719M /home/csc/zwinger
```

The Metacomputer environment: resources



- **CPU time:**
saldo
command
- `-p prjctID`
gives
statistics
for
particular
project
- Similar
feature in
Scientist's I

```
zwinger@vuori2:~> saldo
```

```
-----  
Saldo for year 2010 month 9
```

```
Report updated 9.9.2010 12:15  
-----
```

```
Project tlp0060 Ice2sea (EU project) Thomas Zwinger
```

```
start 04.12.2009      end: 04.12.2011      budget: 30.07.2010
```

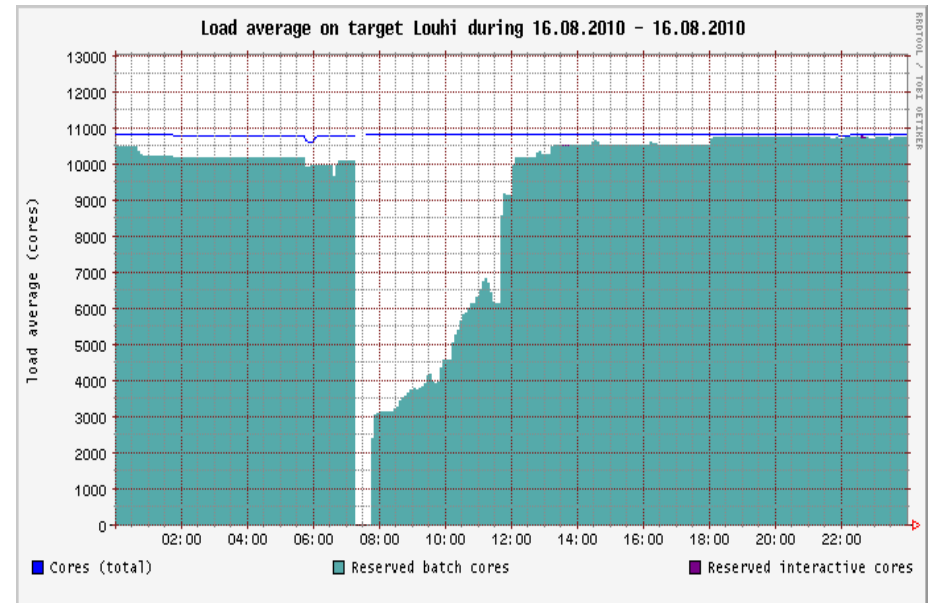
```
CSC budget: 80000    used: 31243.91      remain: 48756.09  
-----
```

	Cpu usage	Cpu secs	Bu	
csc001	zwinger	3324	0.92	hippu
	zwinger	0	0.00	hotpage
	zwinger	1	0.00	murska
	zwinger	600442	166.79	vuori
	zwinger	Tot	603768	167.7

CSC Server: Batch jobs



- Why batch job systems?
 - Usually demand higher than supply
 - Optimal load of machines
 - Optimal experience for user

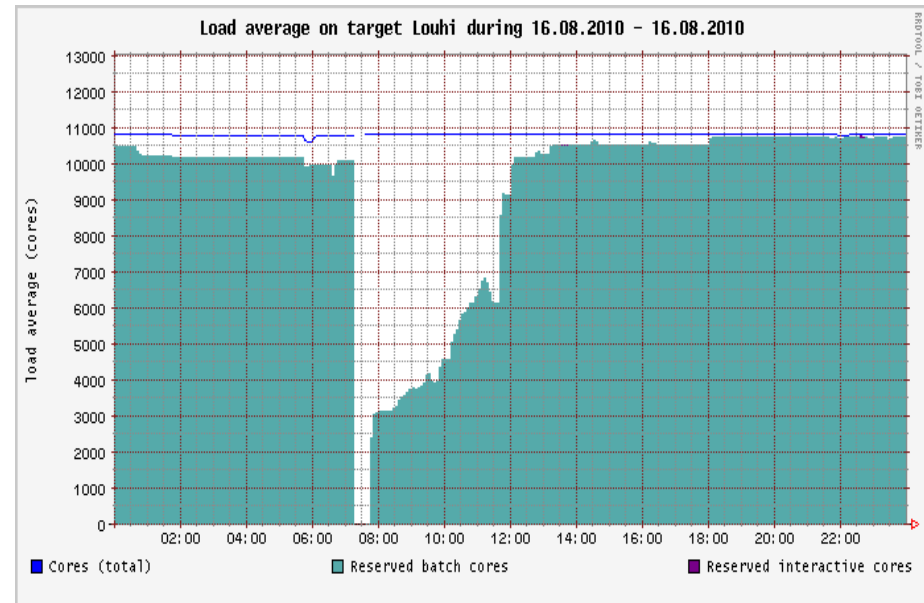


A batch job system always is a compromise between the points above

CSC Server: Batch jobs



- How do resource managers work?
 - Different *queues* (long, short, large, interactive jobs)
 - Optimizes the way free *slots* are filled with requests
 - Tools for communication (monitoring, output files, job-status manipulation)



CSC Server: vuori - SLURM



- Simple Linux Utility for Resource Management
- Open Source solution for larger Linux clusters provided by LLNL
- Two types of jobs: Interactive and batch jobs



[vuori user guide](#)

CSC Server: vuori – Interactive jobs



- Allocate the serial job:

```
> salloc -p interactive -n 1 -t  
02:00:00
```

- Allocates 1 processor for 2 hours.
- Wait until resources get allocated:

```
salloc: Granted job allocation jobid
```

- Run the job:

```
> srun ./my_serial_executable
```

CSC Server: vuori – Interactive jobs



- Parallel interactive job:
 - Simply replace the `-n 1` with `-n N`, where N is the number of cores, e.g.,

```
> salloc -p interactive -n 6 -t  
02:00:00 --mem-per-cpu=1000
```

- Run the job the same way as before

- Alternatively – all-in-one (e.g., serial job):

```
> salloc -p interactive -n 1 -t  
02:00:00 srun ./my_serial_executable
```

CSC Server: vuori – serial batch jobs



- All directives for SLURM start with #SBATCH
- -J job name
- -e stderr
- -o stdout
- %j adds the job-id

```
#!/bin/csh
#SBATCH -J my_jobname
#SBATCH -e my_output_err_%j
#SBATCH -o my_output_%j
#SBATCH --mem-per-cpu=1000
#SBATCH -t 01:01:00
#SBATCH -n 1
srun ./my_serial_program
```

CSC Server: vuori – parallel batch jobs



```
#!/bin/csh
#SBATCH -J my_jobname
#SBATCH -e my_output_err_%j
#SBATCH -o my_output_%j
#SBATCH --mem-per-cpu=1000
#SBATCH -t 11:01:00
#SBATCH -n 24
#SBATCH --ntasks-per-node=12
#SBATCH -p parallel
srun ./my_mpi_program
```

- Each node has 2 6-core CPU's
- Use multiple of 6 for parallel runs
- `-ntasks-per-node` influences the number of used nodes (communication)

CSC Server: vuori - batch job handling



- Status:

```
> sinfo -all
```

- Submitting jobs (graphically xsub):

```
> sbatch batchjobscript.sh
```

- Monitoring jobs (displays *JOBID*):

```
> squeue [-u userid]
```

- Deleting jobs:

```
> scancel JOBID
```

CSC Server: murska - LSF-SLURM



- Jobs submitted to LSF, which is a layer above SLURM
- Scheduling to LSF
- Resources managed by SLURM
- Two types of jobs: Interactive and batch jobs



[murska user guide](#)

CSC Server: murska - Interactive jobs



- Allocate the serial job:

```
> bsub -n 1 -M 1048576 -W 00:30  
-Ip $SHELL -i
```

- Allocates 1 processor for 30 minutes and 1GB/process.
- Wait until resources get allocated:
<<Starting on lsfhost.localdomain>>

- Run the job:

```
> srun ./my_serial_executable
```

- Exit after job:

```
> exit
```

CSC Server: murska - Interactive jobs



- Parallel interactive job:

```
> bsub -n 4 -M 1048576 -W 00:30  
-Ip $SHELL -i
```

- Allocates 1 processor for 30 minutes.
- 1 GB max per process, hence 4 GB in total

- Run the job:

```
> mpirun -srun ./my_MPI_executable
```

CSC Server: murska - Interactive jobs



- All in one (e.g., parallel):

```
> bsub -n 4 -M 1048576 -W 00:30  
-Ip srun -mpirun  
./my_MPI_executable
```

- In an extra xterm:

```
> xterm -T "$HOST 4p 20min" -e  
bsub -n 4 -W 00:20 -Ip $SHELL &
```

or

```
> bsub -n 4 -W 00:20 -I  
/usr/bin/xterm &
```

CSC Server: murska - serial batch jobs



- All LSF directives start with #BSUB
- -J job name
- -e stderr
- -o stdout
- %j adds the job-id
- -M memory in kB
- -W wall clock time
- -N email notification
- bjobs job summary

```
#!/bin/csh
#BSUB -L /bin/csh
#BSUB -J my_jobname%J
#BSUB -e my_output_err_%J
#BSUB -o my_output_%J
#BSUB -N
#BSUB -M 524288
#BSUB -W 01:01
#BSUB -n 1
srun my_serial_program
bjobs -l $LSB_JOBID
```

CSC Server: murska - parallel batch jobs



```
#!/bin/csh
#BSUB -L /bin/csh
#BSUB -J my_jobname%J
#BSUB -e my_output_err_%J
#BSUB -o my_output_%J
#BSUB -N
#BSUB -M 524288
#BSUB -W 01:01
#BSUB -n 4

/opt/hpmpi/bin/mpirun -srun my_MPI_program
bjobs -l $LSB_JOBID
```

CSC Server: murska - batch jobs



- Advanced option to bsub (also in script):

option	purpose
-q queue	choose queue
-u email@address	Specify mail address (in combination with -N option)
-ext "SLURM[options]"	External SLURM option (as would be given to srun command)
-ext "SLURM[nodes=N]"	Distribute run over N nodes (in order to control the cores/node)
-ext "SLURM[constrain=smallmem bigmem hugemem]"	Use up to 8GB 16GB 32GB per node

CSC Server: murska - batch job handling



- Available queues:

```
> bqueues
```

- Submitting jobs (graphically xsub):

```
> bsub < batchjobscript.sh
```

- Monitoring jobs (displays *JOBID*):

```
> bjobs [-u all]
```

- Deleting jobs:

```
> bkill JOBID
```

CSC Server: louhi - PBS

- Jobs submitted to PBS
- Only one type of job:
 - parallel batch jobs
 - Interactive only for debugger session
- CNL (Compute Node Linux) on compute nodes → cross-platform-compilation
- All PBS commands:

#PBS



[louhi user guide](#)

CSC Server: louhi – parallel batch jobs



```
#!/bin/sh
#PBS -N my_job
#PBS -j oe
#PBS -l walltime=1:00:00
#PBS -l mppwidth=256
#PBS -m e
#PBS -M user1@univ2.fi
#PBS -r n
cd $PBS_O_WORKDIR
aprun -n 256 programextbl
```

-N *name* of job

-j oe combined output

-l resource option

-n option for aprun corresponds to mppwidth

-m e mail notification

-r n cannot be rerun

\$PBD_O_WORKDIR
job-submission directory

CSC Server: louhi - batch job handling



- Queue status:

```
> qstat -Q
```

- Submitting jobs (-h displays *jobid*):

```
> qsub [-h] batchjobscript.sh
```

- Monitoring jobs:

```
> qstat [-u user] [-a jobid]
```

- Deleting jobs:

```
> qdel jobid
```