JÜLICH
FORSCHUNGSZENTRUM

JURECA

Summer School on Fire Dynamics Modeling 2017

Lukas Arnold

Contents:

Motivation for HPC

Many scientific and engineering problems require a large amount of computing time and / or memory. This is generally only available on supercomputers / high performance computing (HPC) systems.

Typical applications are:

- science: particle physics, climate research, molecular dynamics
- engineering: CFD, structural mechanics, computer science

HPC with computing power thousands times larger than personal computer allows to significantly reduce computing time and allows to solve new challenges.

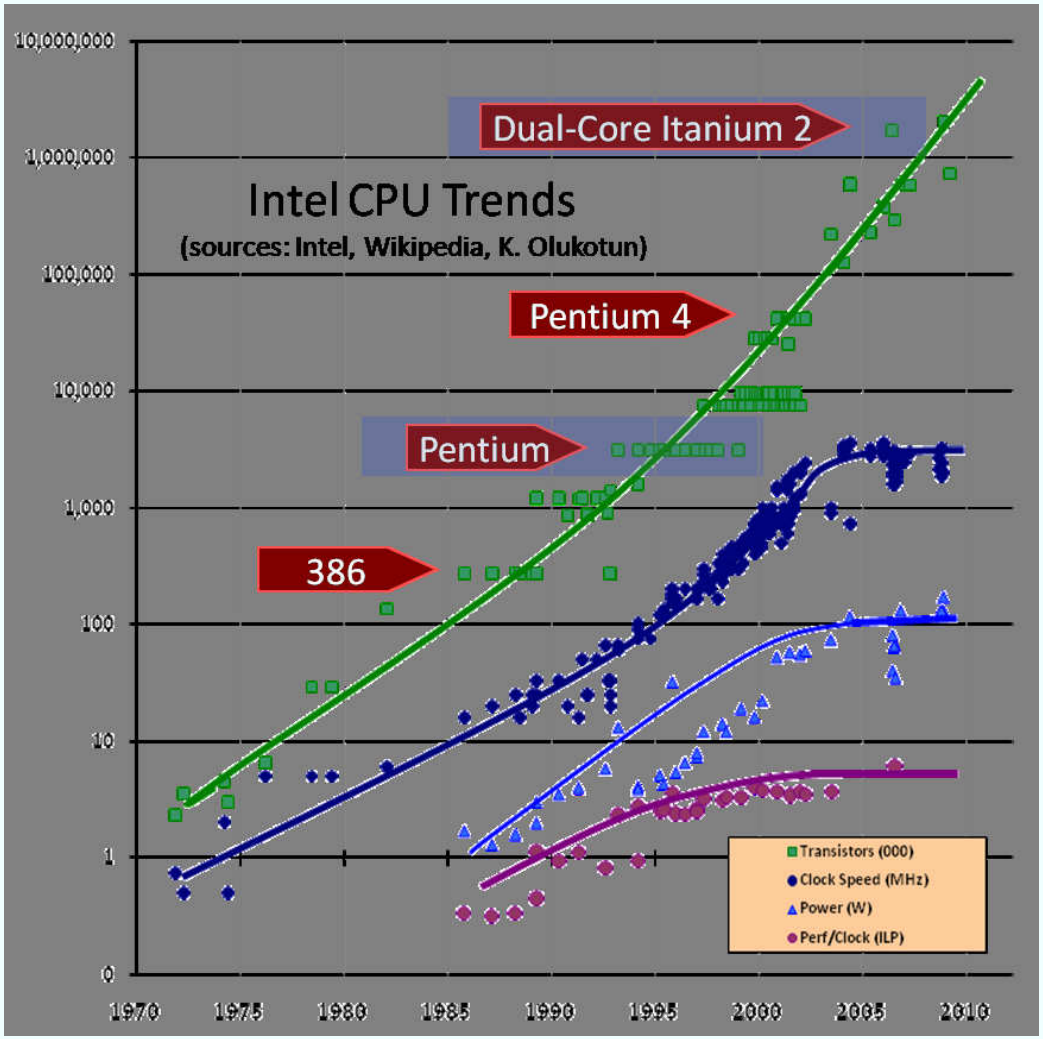
Example: An CFD application run for a week on thousand processors would need 20 years on a personal computer (assuming it provides enough memory).

Terminology

A list of common terms used in HPC

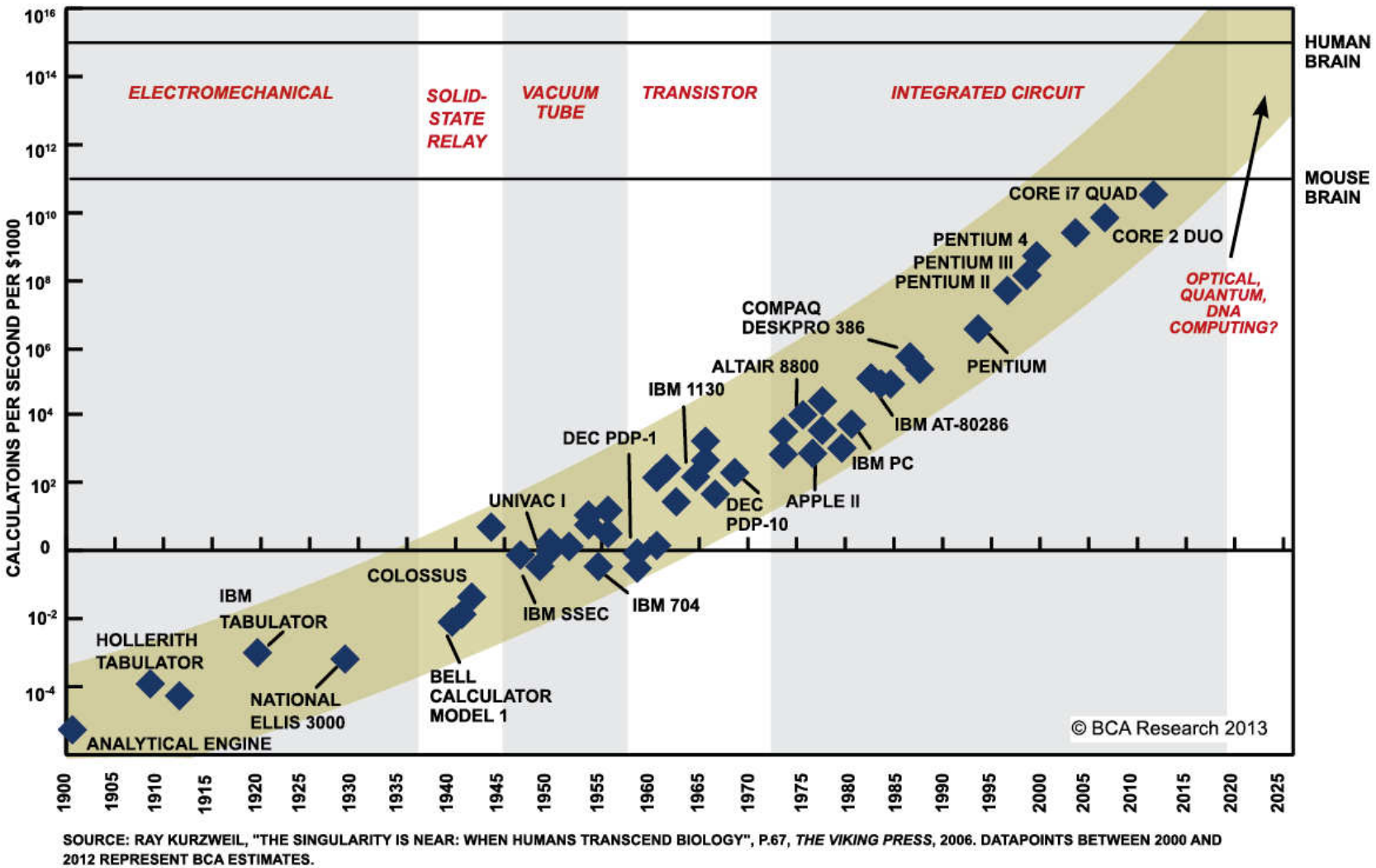
- CPU, GPGPU, SIMD, SMP, NUMA, SMT
- compute node: nodes that are used only for computation, no direct user access
- front/login node: users login here, do compilation and job submission
- batch/submission system: system to allocate resources to a scheduled job
- interconnect: network between components or computing nodes (e.g. Infiniband, 10GE)
- FLOP/s: floating point operations per second
- peak performance: theoretical number of FLOP per second
- linpack performance: floating point performance in the linpack test
- top500 list: supercomputer ranking based on the linpack performance

Moore's law (I)



Moore's law: the number of transistors in integrated circuits double every two years (pure observation)

Moore's law (II)



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Top500 list (I)

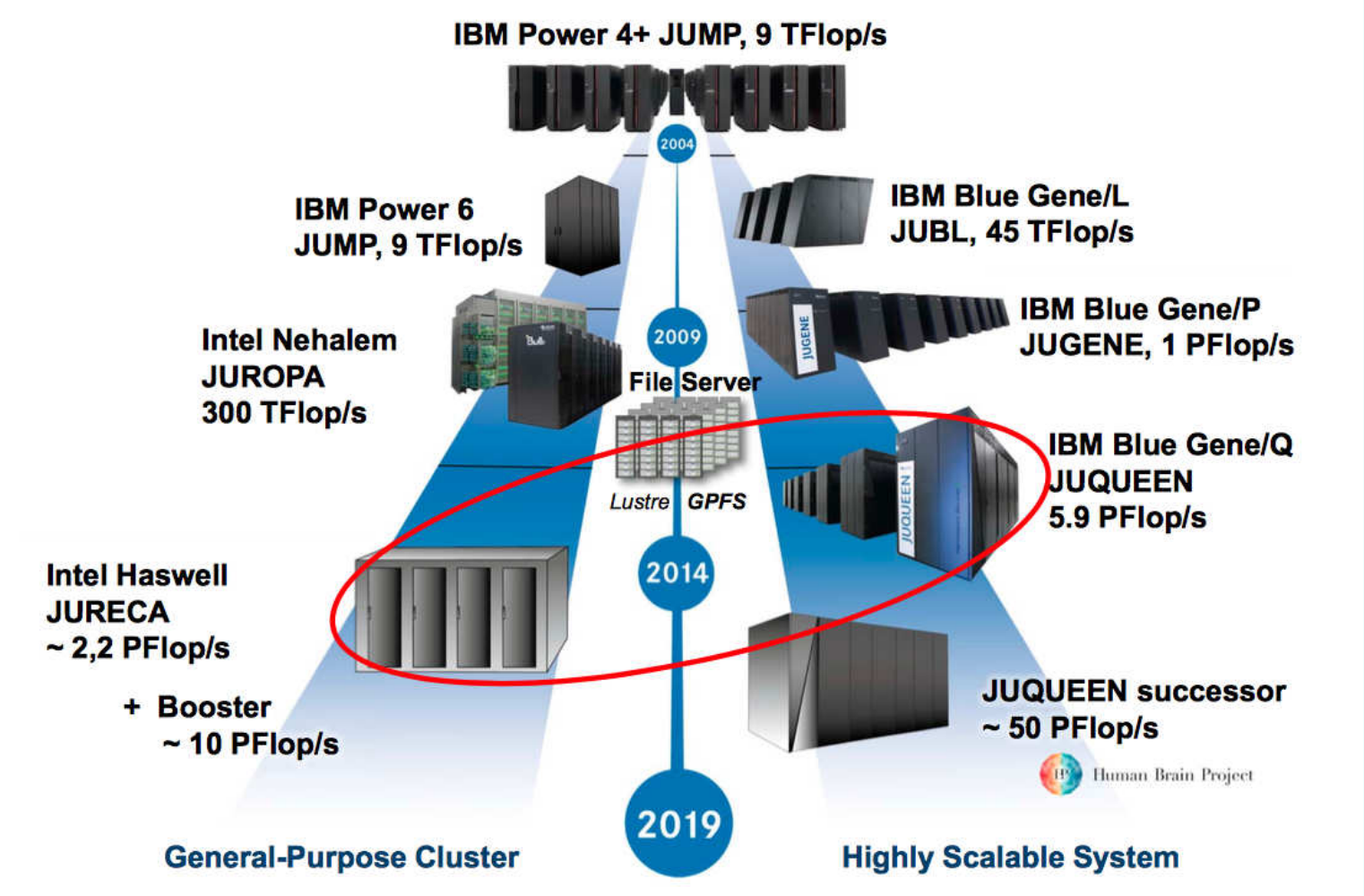
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - 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
3	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
4	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
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
7	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
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
9	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
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233

source: top500.org

Top500 list (II)

source: top500.org

JSC's systems – overview

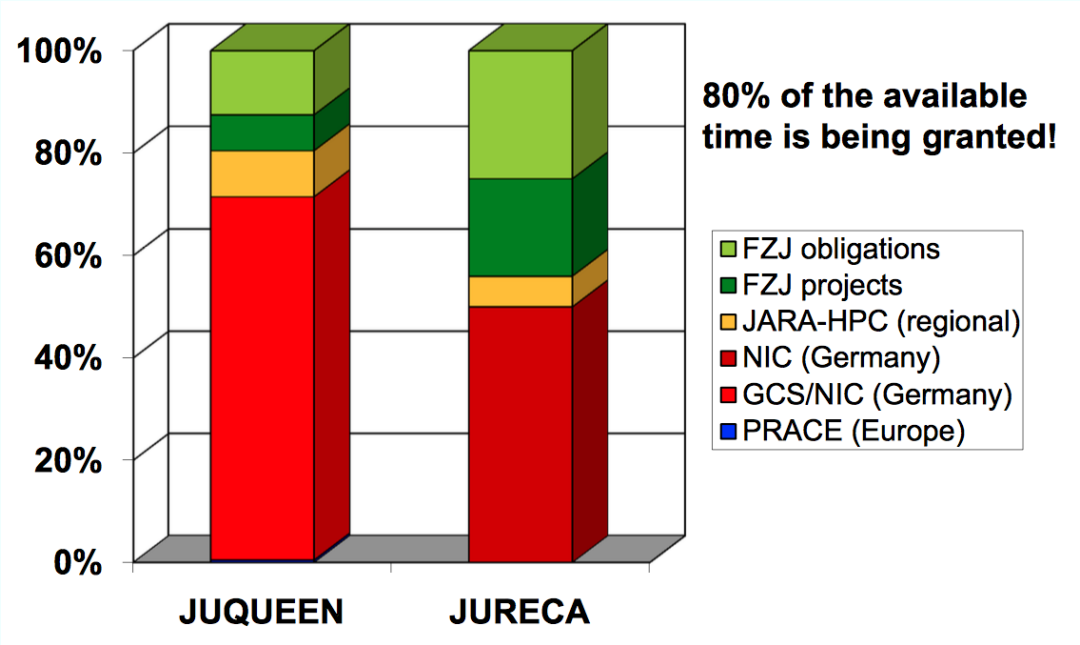


JSC's systems – JURECA

JURECA (installed in 2015)

- two Intel Xeon E5-2680 v3 Haswell, 12 cores
- 128 / 256 / 512 / 1024 GB per node
- 45,216 cores
- 1.8 Peta FLOP/s peak
- equipped with NVIDIA GPGPUs (K40, K80)
- EDR InfiniBand network (100 Gbit/s)

PRACE, NIC, VSR



The computing time on JSC's supercomputers is granted via three structures:

- PRACE, for European scientist
- NIC, for German scientist
- VSR, for FZJ members

There are two deadlines for computing time proposals.

ssh connection

To connect to modern supercomputer, you have to use `ssh` with a public-private key encryption.

The connection syntax is as follows

```
ssh user@server
```

The configuration files and public-private keys are stored in `~/.ssh`

To ease a frequently used connection, the `config` file may by customised, e.g.

```
1  Host jureca
2  Hostname jureca.fz-juelich.de
3  User train115
4  IdentityFile ~/.ssh/id_train115
```

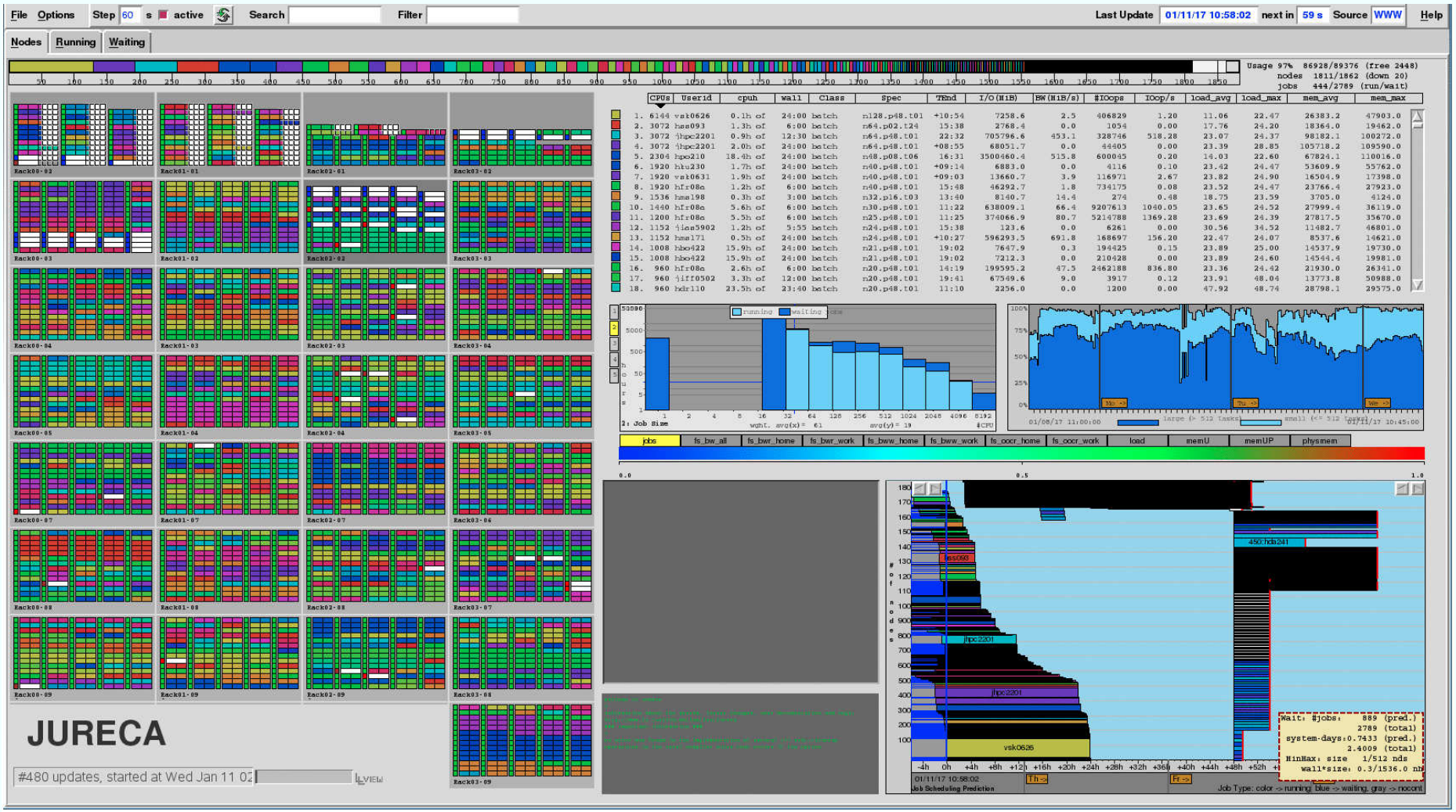
Remote file copy (in both directions) is done with `scp`:

```
scp file.dat user@server:
```

Or use a sftp client. Enter the following line to the address line of the file browser in the VM:

```
sftp://jureca/homea/hpclab/train115
```

Status of JURECA



Module environment

On modern (HPC) computer systems with a multitude of users, multiple versions of software must be provided.

The `module` environment is able to provide an easy way to have parallel installations without interaction.

Some common `module` commands are:

- `avail`: list all available modules
- `load`: load a module
- `unload` / `purge`: unload target module / all modules

On JURECA, you have first to load a compiler and MPI combination to see the available software packages.

Overview batch system

The batch system manages the user requests and the available resources.

A typical procedure is as follows:

1. the user prepares the environment for a job
2. the user defines the resources needed to run a job (wall-clock time, number of nodes)
3. the user submits the job definition
4. the batch system checks continuously if the requested resources are free
5. if the requested resources are available, the batch system executes the job according the the job definition

The JURECA system

The JURECA system is divided into various partitions and is managed by the `slurm` batch system.

partition	max. nodes	max. WCT [h]	info
devel	8	2	for code development
develgpus	2	2	allocates GPUs (K40)
batch	256	24	default queue
mem512	32	24	nodes with 512 GB memory
mem1024	2	24	nodes with 1024 GB memory
gpus	32	24	allocates GPUs (K80)
vis	4	24	visualisation nodes

Job Scripts

A typical structure of a job definition file is:

slurm.job

```
1  #!/bin/bash -x
2  #SBATCH --job-name slurm-test
3  #SBATCH --nodes=8
4  #SBATCH --cpus-per-task=4
5  #SBATCH --time=00:30:00
6  ### start of jobscript
7
8  export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
9
10 srun ./mpi_prog
```

Useful Commands

To submit and monitor jobs the following commands are used:

- `sbatch`: submit a job
- `scancel`: chancel a job
- `squeue`: show the job queues

In addition, it is possible to define job chains, i.e. dependant jobs which will run in sequence.

FDS job template

fds.jobtemplate

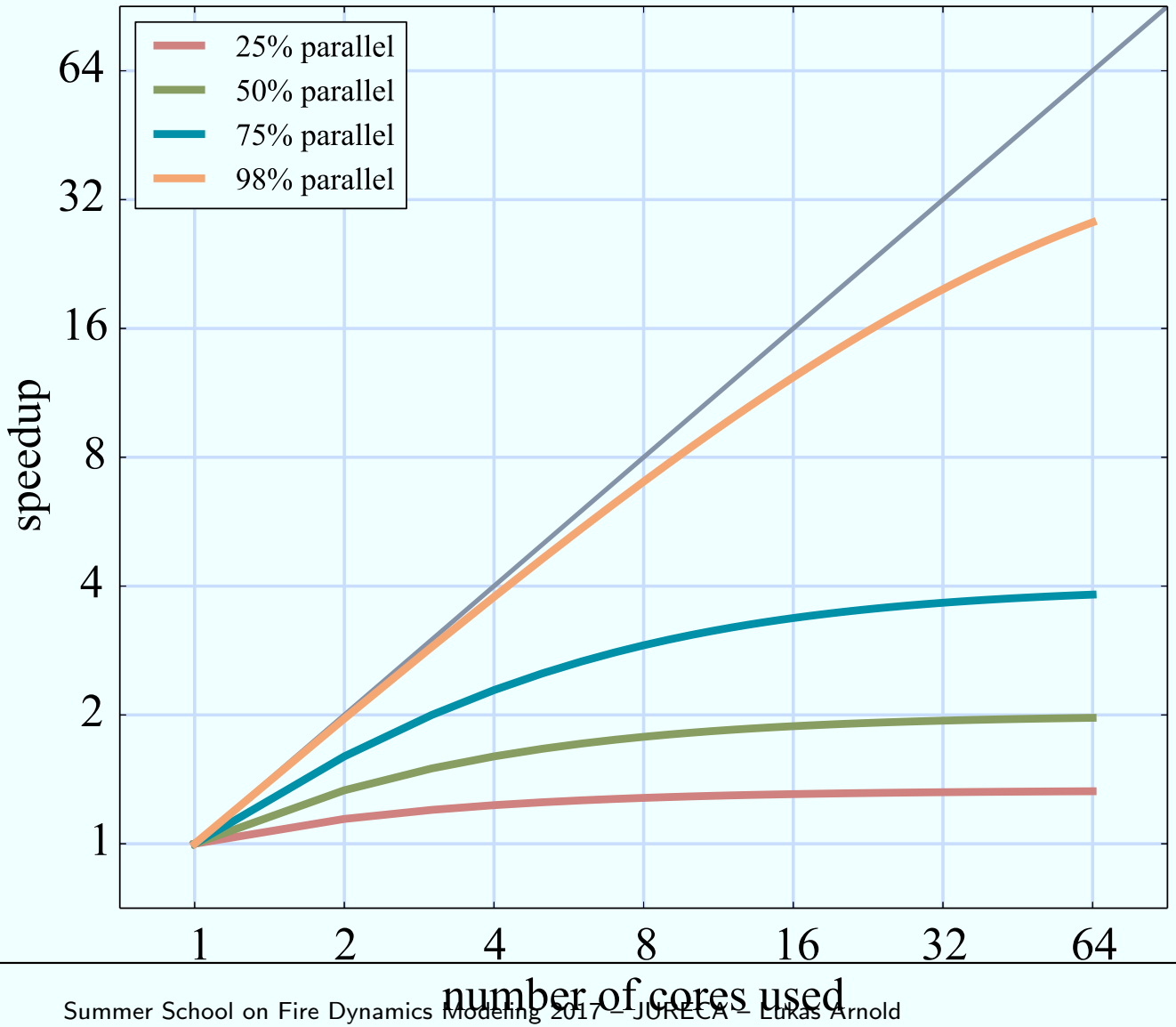
```
 8  ## INFO: compute nodes of JURECA have 24 (physical) cores
 9  ## how many MPI tasks?
10  #SBATCH --ntasks XX
11
12  ## how many OMP threads?
13  #SBATCH --cpus-per-task XX
14
15  ## how long shall the job run? Format: HH:MM:SS
16  #SBATCH --time=00:XX:00
17
18  ## load FDS module
19  module use --append ~larnold/modules_fire
20  module load FDS/6.5.3-intel2017-intel2017
21
22  ## set number of OMP threads based on SLURM setting
23  export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
24
25  ## run FDS with <INPUTFILE> as input file
26  srun fds <INPUTFILE>
```

Tuesday reservation

We have a reservation for Wednesday. To submit your job into it, use the following extended `sbatch` command:

```
sbatch -p batch --reservation=fire-sim <job file>
```

Amdahl's law


25% parallel
50% parallel
75% parallel
98% parallel
speedup
64
32
16
8
4
2
1
1
2
4
8
16
32
64
number of cores used

FDS scaling (I)

The `exercises/scaling` directory contains a JURECA job file `fds_scaling.job` to run a series of FDS simulations:

fds_scaling.job

```
18  ## load FDS module
19  module use --append ~larnold/modules_fire
20  module load FDS/6.5.3-intel2017-intel2017
21
22  ## set number of OMP threads based on SLURM setting
23  export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
24
25  ## run FDS with as input file
26  for n in 01 02 04 08; do
27      echo "running scaling test with $n MPI tasks"
28      echo "start time: `date`"
29      mkdir -p rundir_n$n
30      cd rundir_n$n
31      srun -n $n fds ../scaling_m$n.fds
32      echo "stop time: `date`"
33      cd -
34  done
```

FDS scaling (II)

1. Copy the timing data (`*_cpu.scv`) into the directory `plot_cpu_timings`.
2. Execute the `plot_scaling.py` script.