



Introduction to HPC with practicals on HPCFS

Presenter: Leon Kos, University of Ljubljana

Date: 12 Jun 2022

Why supercomputing?

- **Weather, Climatology, Earth Science**

- degree of warming, scenarios for our future climate.
- understand and predict ocean properties and variations
- weather and flood events

- **Astrophysics, Elementary particle physics, Plasma physics**

- *systems, structures which span a large range of different length and time scales*
- *quantum field theories like QCD, ITER*

- **Material Science, Chemistry, Nanoscience**

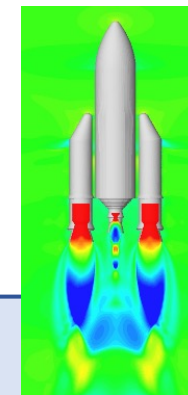
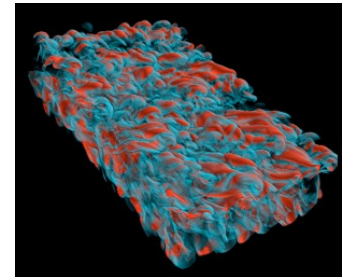
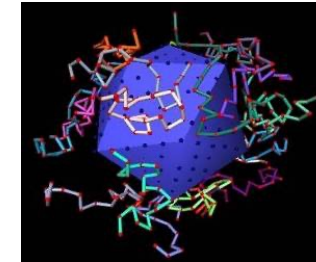
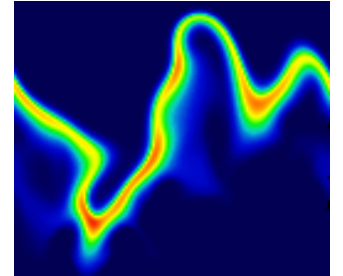
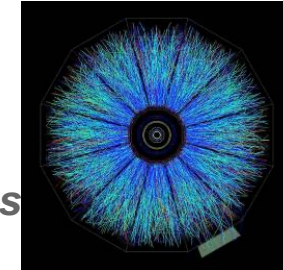
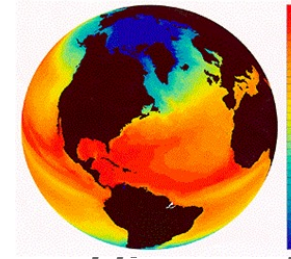
- *understanding complex materials, complex chemistry, nanoscience*
- *the determination of electronic and transport properties*

- **Life Science**

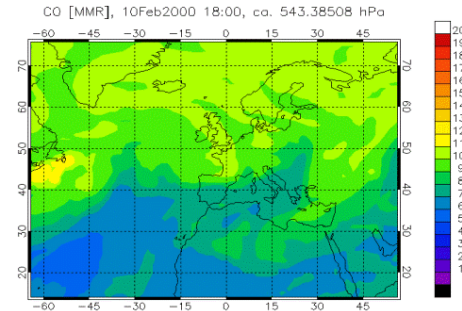
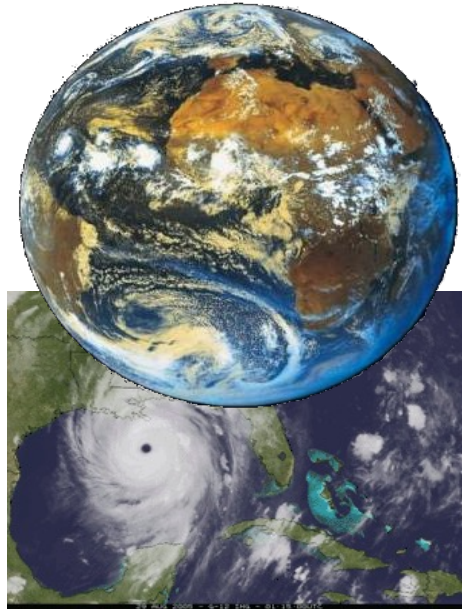
- *system biology, chromatin dynamics, large scale protein dynamics, protein association and aggregation, supramolecular systems, medicine*

- **Engineering**

- *complex helicopter simulation, biomedical flows, gas turbines and internal combustion engines, forest fires, green aircraft,*
- *virtual power plant*



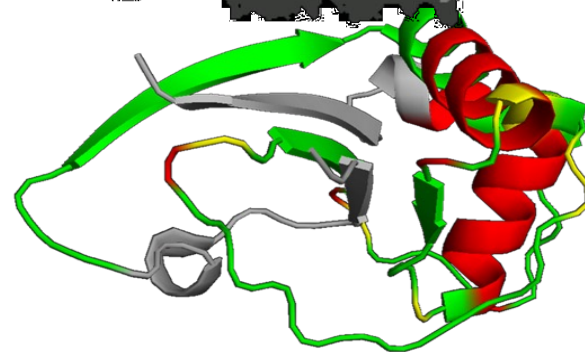
Supercomputing drives science with simulations



Environment

Weather/ Climatology

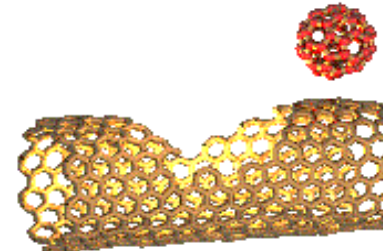
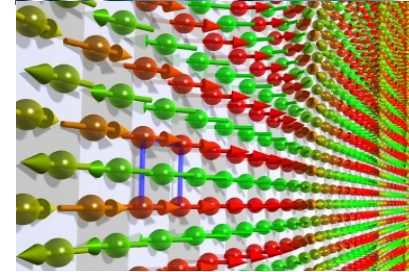
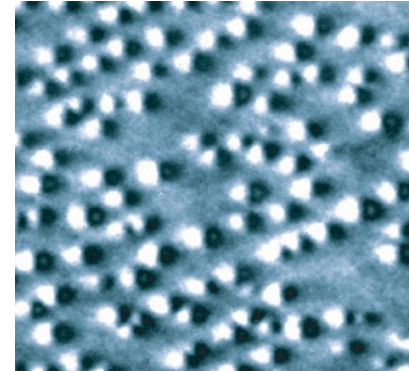
Pollution / Ozone Hole



Aging Society

Medicine

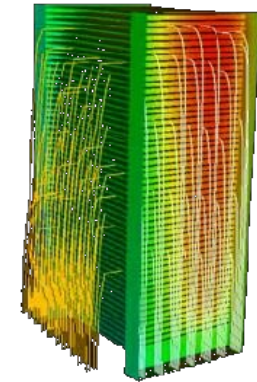
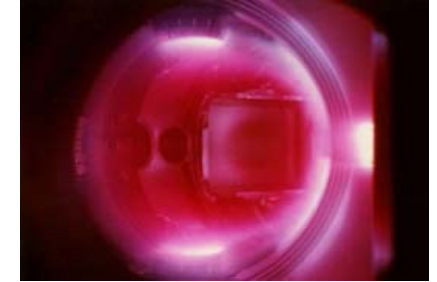
Biology



Materials/ Inf. Tech

Spintronics

Nano-science



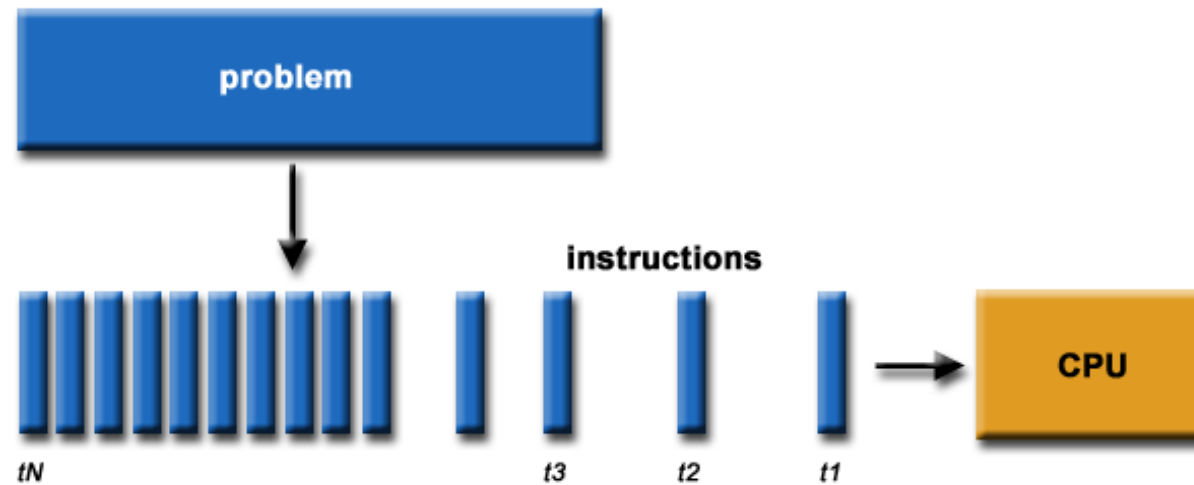
Energy

Plasma Physics

Fuel Cells

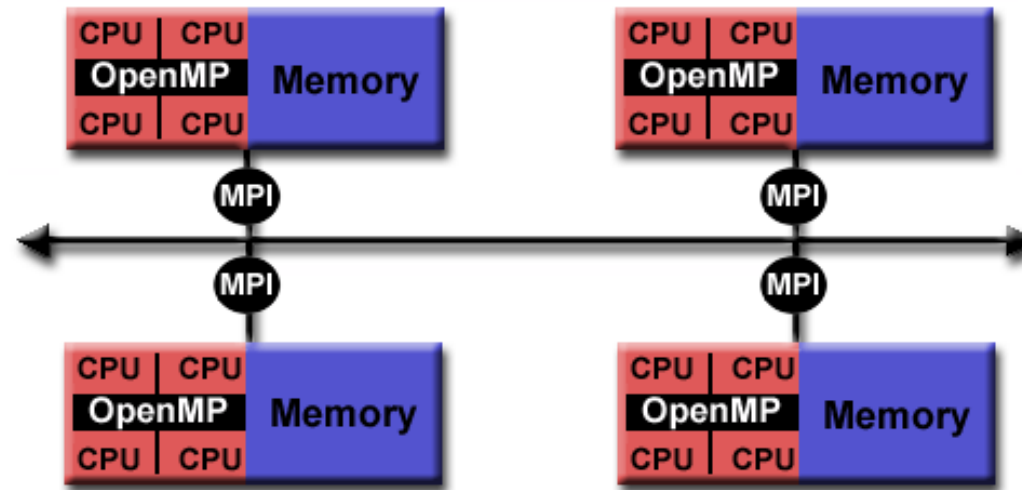
Introduction to parallel computing

- ▶ Usually is the program written for serial execution on one processor
- ▶ We divide the problem into series of commands that can be executed in parallel
- ▶ Only one command at a time can be executed on one CPU



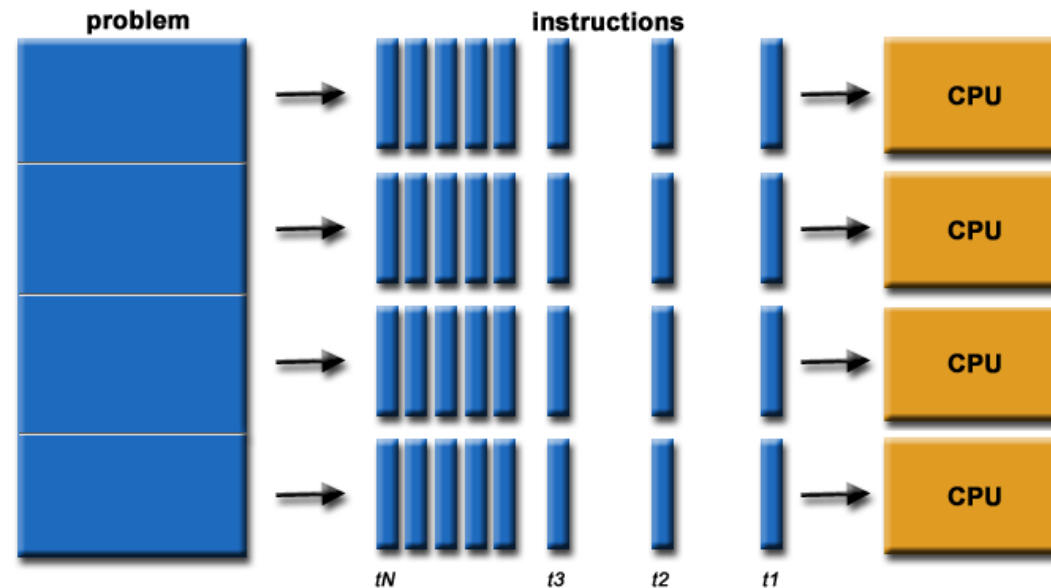
Parallel programming models

- ▶ Threading
- ▶ **OpenMP** – *automatic parallelization*
- ▶ Distributed memory model = **Message Passing Interface (MPI)** – *manual parallelization needed*
- ▶ **Hybrid model OpenMP/MPI**



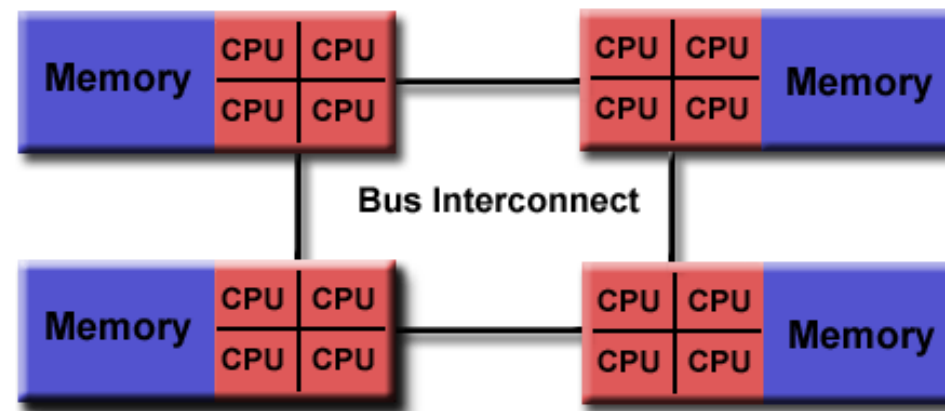
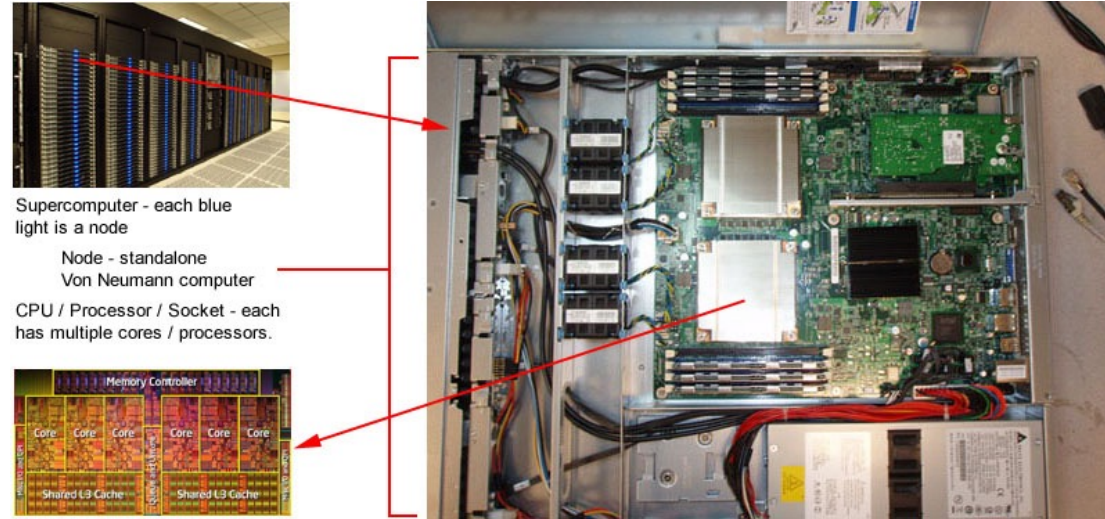
Embarrassingly simple parallel processing

- ▶ Parallel processing of the same subproblems on multiple processors
- ▶ No communication is needed between processes



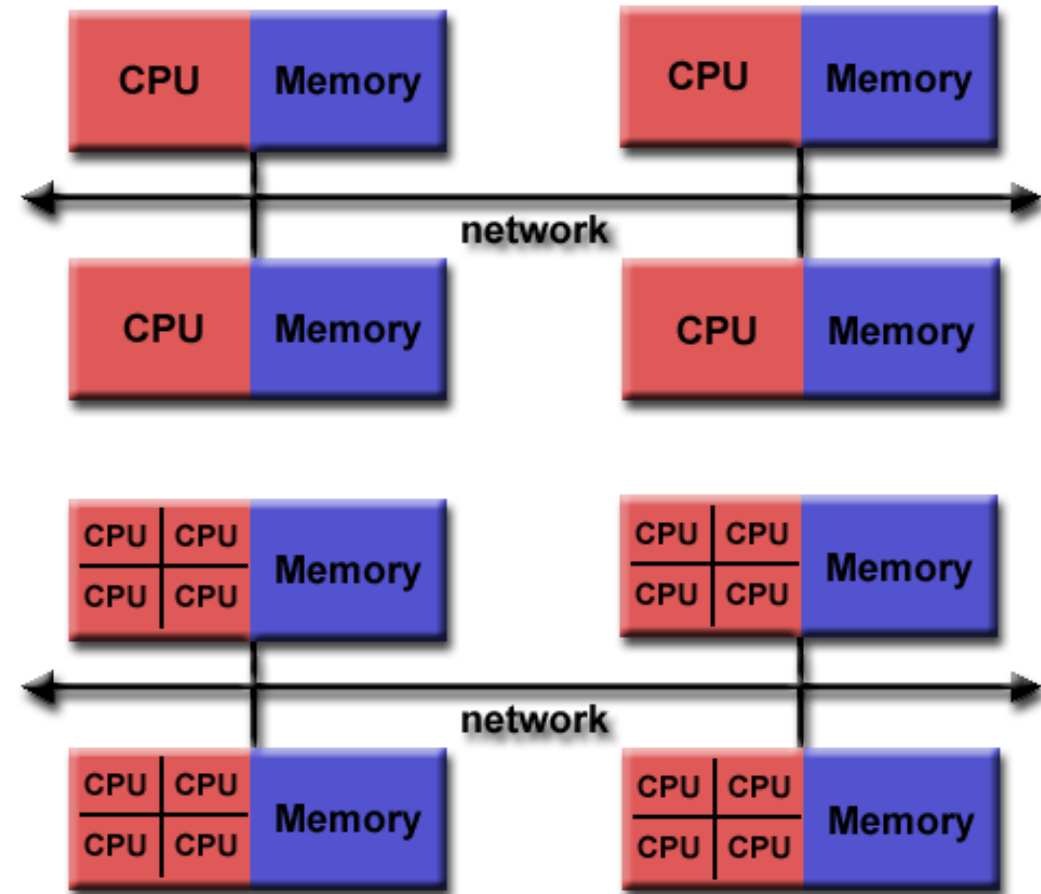
Logical view of a computing node

- ▶ Need to know computer architecture
- ▶ ***Interconnect bus for sharing memory between processors (NUMA interconnect)***



Nodes **interconnect**

- ▶ Distributed computing
 - ▶ Many nodes exchange messages on
 - ▶ high speed,
 - ▶ low latency interconnect
- such as **Infiniband**



Development of parallel codes

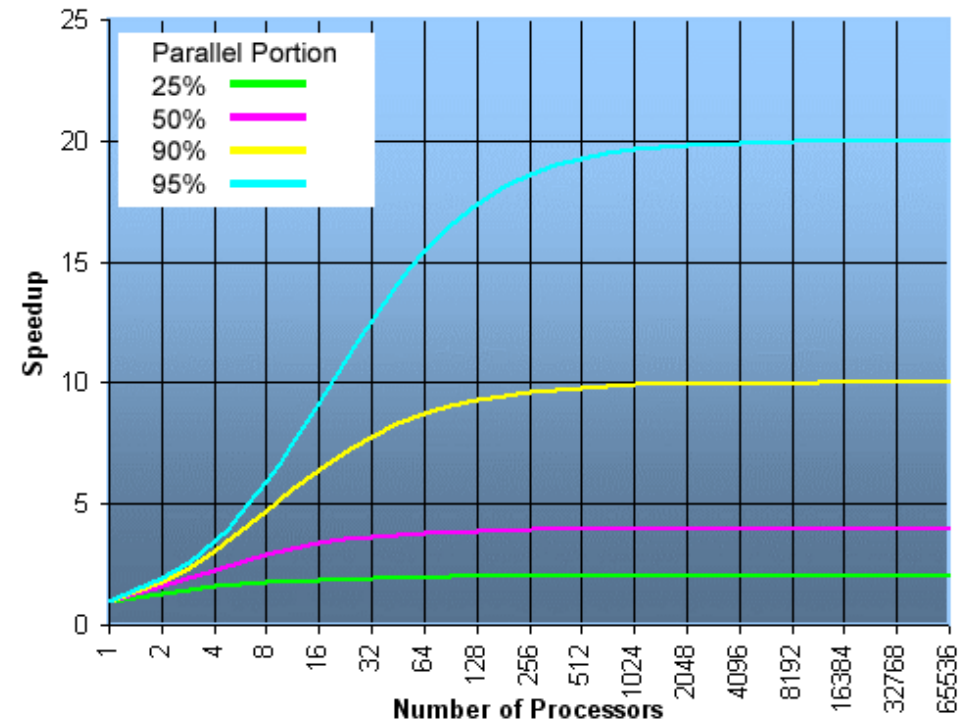
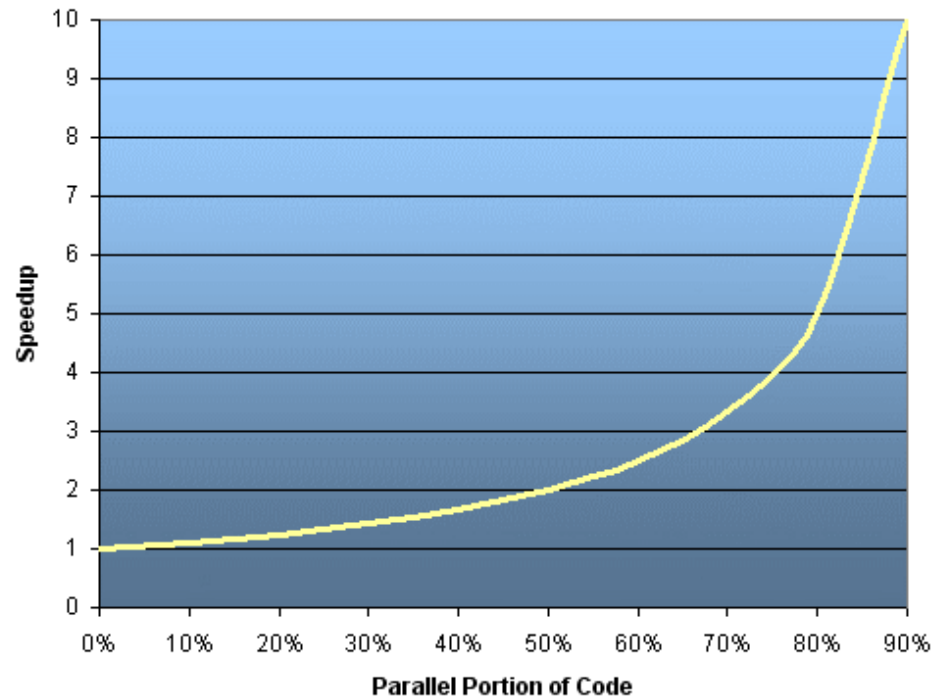
- ▶ Good understanding of the problem being solved in parallel
- ▶ How much of the problem can be run in parallel
- ▶ Bottleneck analysis and profiling gives good picture on scalability of the problem
- ▶ We optimize and parallelize parts that consume most of the computing time
- ▶ Problem needs to be dissected into parts functionally and logically

Interprocess communications

- ▶ Having little and infrequent communication between processes is the best
- ▶ Determining the largest block of code that can run in parallel and still provides scalability
- ▶ Basic properties
 - ▶ *response time*
 - ▶ *transfer speed - bandwidth*
 - ▶ *interconnect capabilities*

Parallel portion of the code determines code **scalability**

► Amdahl's law: ***Speedup*** = $1/(1-p)$



Direct Solver or Iterative Solver?

- We are solving a set of matrix equations of the form $[K]\{u\} = \{f\}$. Here $[K]$ is referred to as the stiffness matrix; $\{f\}$ as the force vector and $\{u\}$ as the set of unknowns.
 - Several millions of unknowns
 - Lot of zeros in K
- Direct solvers: Multfront, MUMPS, and LDLT, Pardiso, ...
- Iterative solvers: PETSc and GCPC, ...

Computer Aided Engineering

open source tools

- CAD/CAM: [Salome](#), [Freecad](#), OpenSCAD, LibreCad, Pycam, Camotics, dxf2gcode & Cura
- FEA, CFD & multiphysic simulation: [Salome-Meca](#) / Code-Aster, SalomeCFD/Code-Saturne, HelyxOs/OpenFOAM, Elmer FEM, [Calculix](#) with Launcher & CAE GUI, Impact FEM, MBDyn, [FreeFEM](#), [MFEM](#), [Sparselizard](#)
- Meshing, pre-post, & visualization: [Salome](#), [Paraview](#), Helyx-OS, Elmer GUI, VoxelMesher, Tetgen, CGX, GMSH

Questions and practicals on the HPCFS cluster

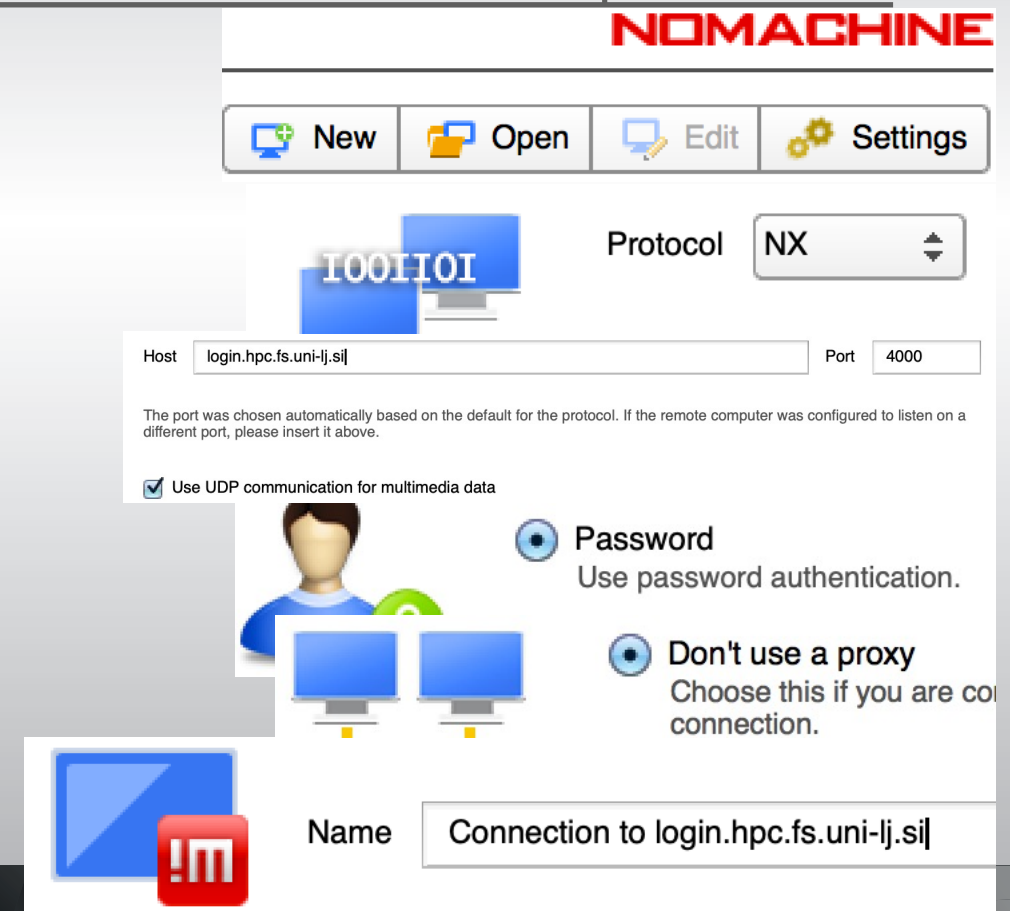
- ▶ Demonstration of the work on the cluster by repeating
- ▶ Access with NX client
- ▶ Learning basic Linux commands
- ▶ SLURM scheduler commands
- ▶ Modules
- ▶ Development with OpenMP and OpenMPI parallel paradigms
- ▶ Exercises and extensions of basic ideas
- ▶ Instructions available at <http://hpc.fs.uni-lj.si/>

NoMachine client



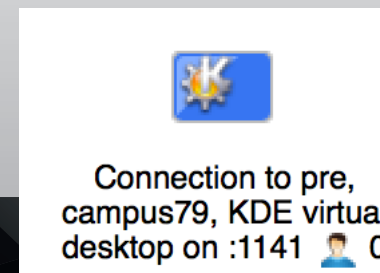
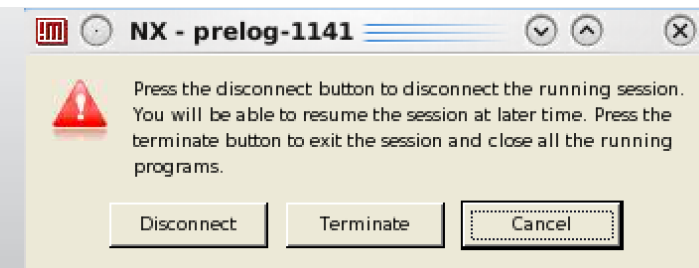
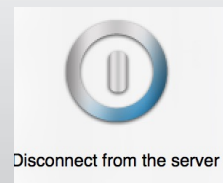
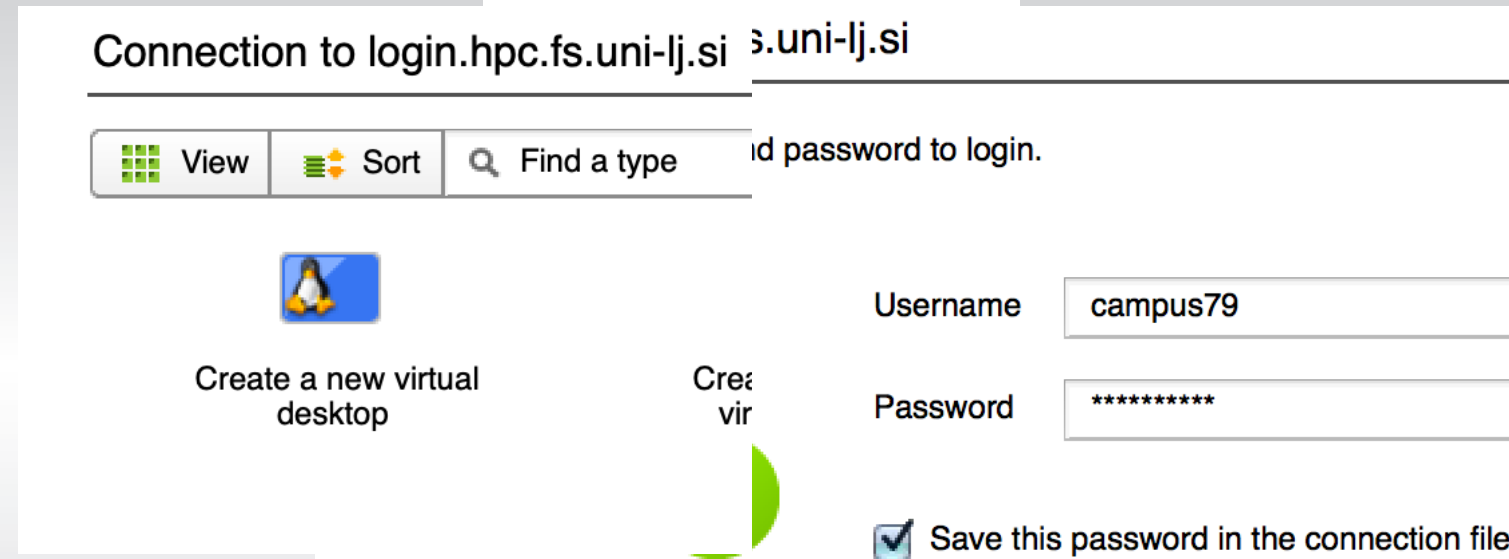
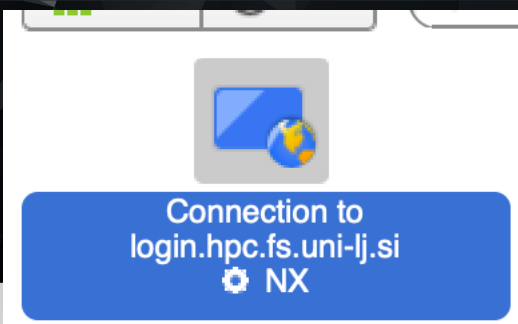
- **Setting up** the NoMachine client version 6 available for installation at page <https://www.nomachine.com/download-enterprise#NoMachine-Enterprise-Client> or <https://bit.ly/1fal6ac>

1. Select New
2. Protocol NX
3. Host: login.hpc.fs.uni-lj.si Port: 4000
4. Use Password authentication
5. Don't use proxy
6. Done with Connection to login.hpc.fs.uni-lj.si

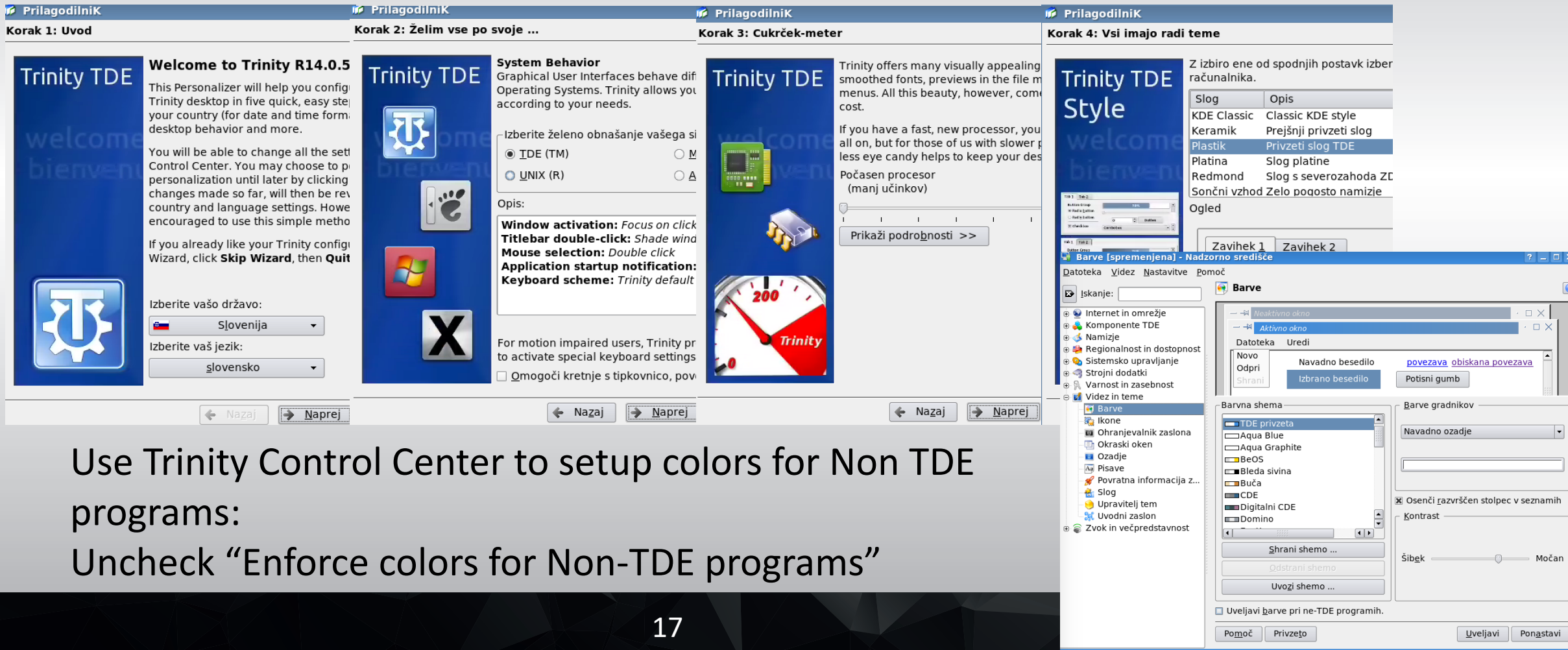


Connecting to HPCFS

1. Select and Connect
2. Use **your** account credentials
3. Create **New** desktop **once**
4. Use the Trinity (KDE) desktop
5. To **Disconnect** press Ctrl+Alt+T
6. To **Reconnect** select previous virtual desktop



Tuning desktop with KPersonalizer for remote speed (use less effects=slower processor)



The image displays four sequential screenshots of the KPersonalizer application interface, which is used for configuring the Trinity desktop environment.

- Korak 1: Uvod (Welcome to Trinity R14.0.5):** This screen provides a welcome message and instructions on how to use the personalizer. It includes a "Skip Wizard" button and a "Quit" button.
- Korak 2: Želim vse po svoje ... (System Behavior):** This screen allows users to configure system behavior. It includes options for "Izberite želeno obnašanje vašega sistema" (Choose the desired system behavior) with radio buttons for "TDE (TM)" and "UNIX (R)". It also includes a section for "Window activation" and "Keyboard scheme".
- Korak 3: Cukrček-meter (Trinity TDE Style):** This screen displays a "Trinity TDE Style" preview and a "Cukrček-meter" (sugar meter) graphic. It includes a "Prikaži podrobnosti" (Show details) button.
- Korak 4: Vsi imajo radi teme (Barve [spremenjena] - Nadzorno središče):** This screen shows the "Barve" (Colors) configuration window. It includes a "Barvna shema" (Color scheme) section with a list of themes (e.g., Aqua Blue, Aqua Graphite, BeOS, Bleda sivina, Buča, CDE, Digitalni CDE, Domino) and a "Barve gradnikov" (Colors of components) section with a "Navadno ozadje" (Default background) dropdown and a "Kontrast" (Contrast) slider.

Use Trinity Control Center to setup colors for Non TDE programs:
Uncheck "Enforce colors for Non-TDE programs"

- Setting GNOME or KDE desktop locale preferences for keyboard, LANG environment
- Using NX client (Disconnect, Terminate, Logout)
- Console commands in Linux
- Editors for programming (emacs, gedit, kate, eclipse, vi, pico, ...) on login only!

Modules (LUA):

- module avail
- module help/info
- module show
- module load/unload
- module list
- module purge

SLURM batch scheduler

Compiled-in OpenMPI support

- `srun --nodes=N --ntasks=n cmd`
- `sbatch script.sh`
- `sinfo`
- `squeue`
- Alias for interactive usage of nodes:

```
alias node='srun -N1 --time=1:00:00 --pty bash -i'
```


Using SLURM (interactively) and Message Passing Interface (MPI)



```
[leon@viz mpi]$ module purge && module load foss/2019a
[leon@viz mpi]$ cat hello.f90
program hello
  use mpi
  integer rank, size, ierror, strlen,
    status(MPI_STATUS_SIZE)
  character(len=MPI_MAX_PROCESSOR_NAME) :: hostname

  call MPI_INIT(ierror)
  call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
  call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
  call MPI_GET_PROCESSOR_NAME(hostname, strlen, ierror)
  print*, trim(hostname), rank, size
  call MPI_FINALIZE(ierror)
end
```

```
[leon@viz mpi]$ mpif90 hello.f90
[leon@viz mpi]$ LD_PRELOAD= srun -
n 4 --tasks-per-node=2 --kill-on-
bad-exit --partition=haswell
./a.out
```

cn80	2	4
cn79	0	4
cn80	3	4
cn79	1	4

```
#include <stdio.h>
#include <math.h>
#define N 1000000
int main()
{
    double area = 0.0;
    #pragma omp parallel for reduction(+:area)
    for(int i = 0; i < N; i++)
    {
        double x = (i+0.5)/N;
        area += sqrt(1.0 - x*x);
    }
    printf("Površina : %14lf\n", 4.0*area/N);
    return 0;
}
```

```
[leon@cn36 pi]$ module purge && module
load foss/2019a

[leon@cn36 pi]$ gcc -fopenmp pi-
openmp.c -lm -o pi-openmp

[leon@cn36 pi]$ OMP_NUM_THREADS=4 ./pi-
openmp
```



Thanks!



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, United Kingdom, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, Republic of North Macedonia, Iceland, Montenegro



EuroHPC
Joint Undertaking