EURO

SLING

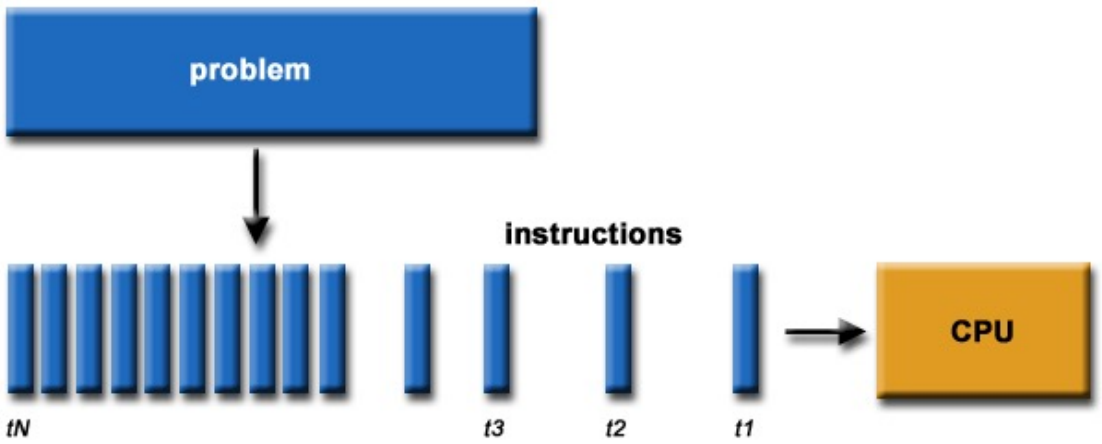
Introduction to HPC and HPCFS cluster usage

Leon Kos

Date: 9 Feb 2022

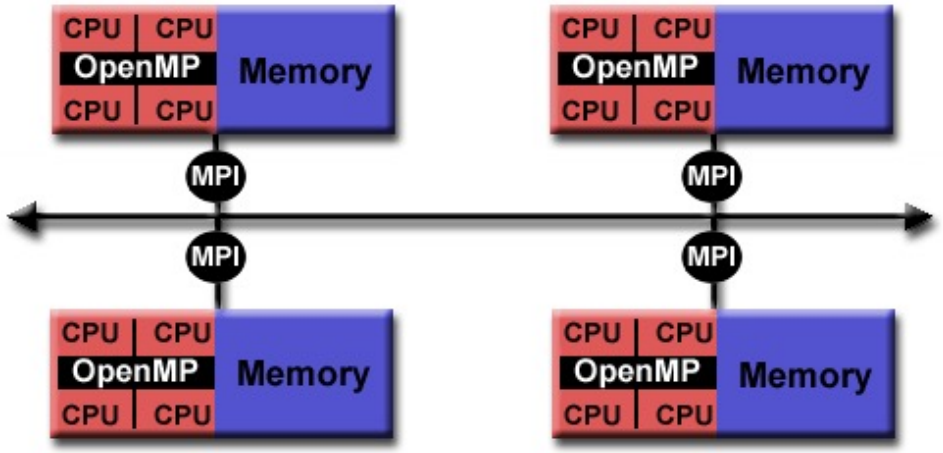
Introduction to parallel computing

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



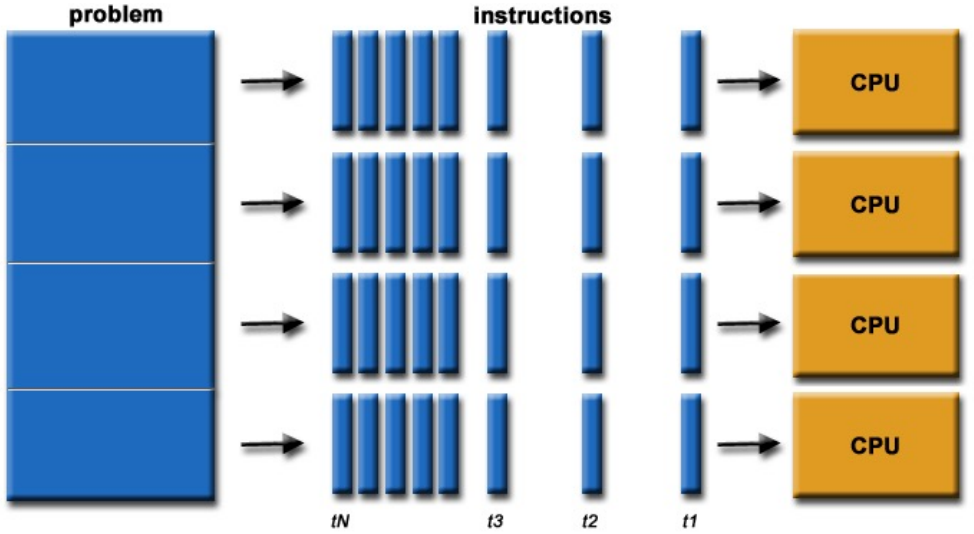
Parallel programming models

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



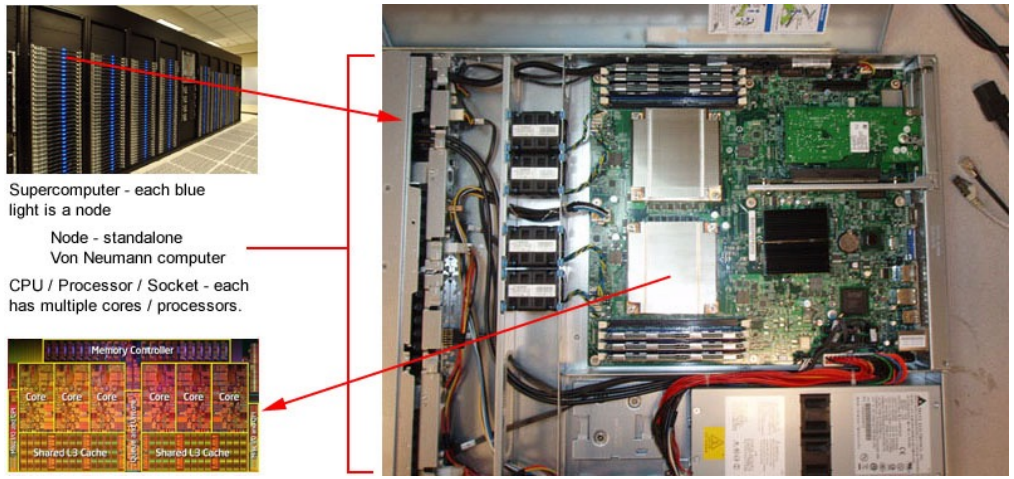
Embarrasingly simple parallel processing

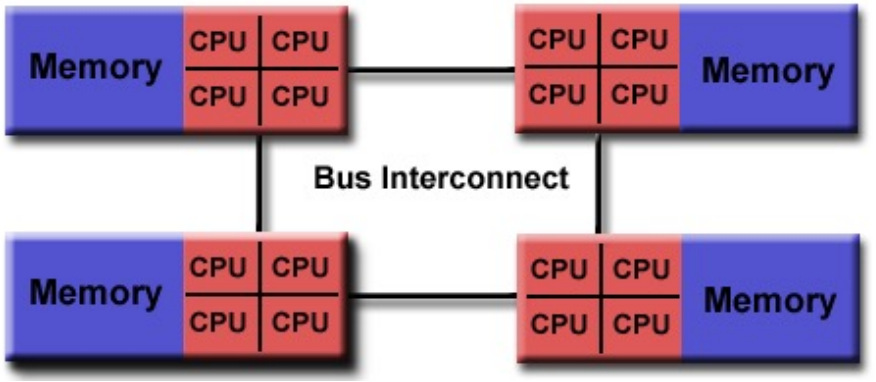
- Parallel processing of the same subproblems on multiple prooocessors
- No communication is needed between processes



Logical view of a computing node

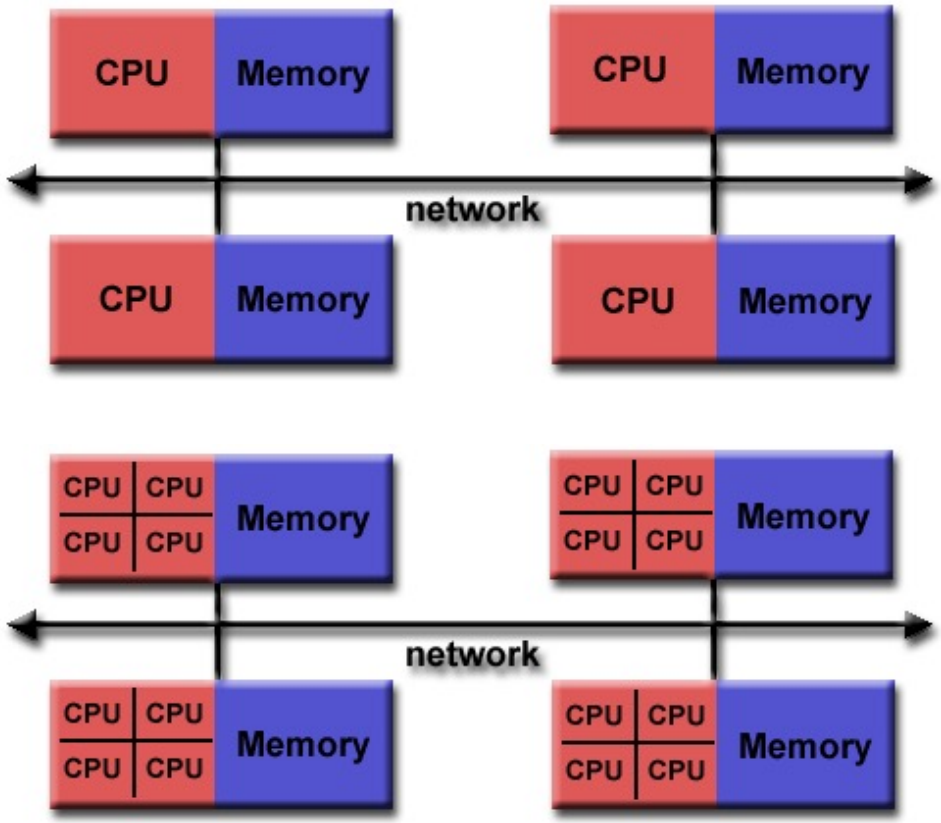
- Need to know computer architecture
- ***Interconnect bus for sharing memory between processors (NUMA interconnect)***





Nodes interconnect

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



Development of parallel codes

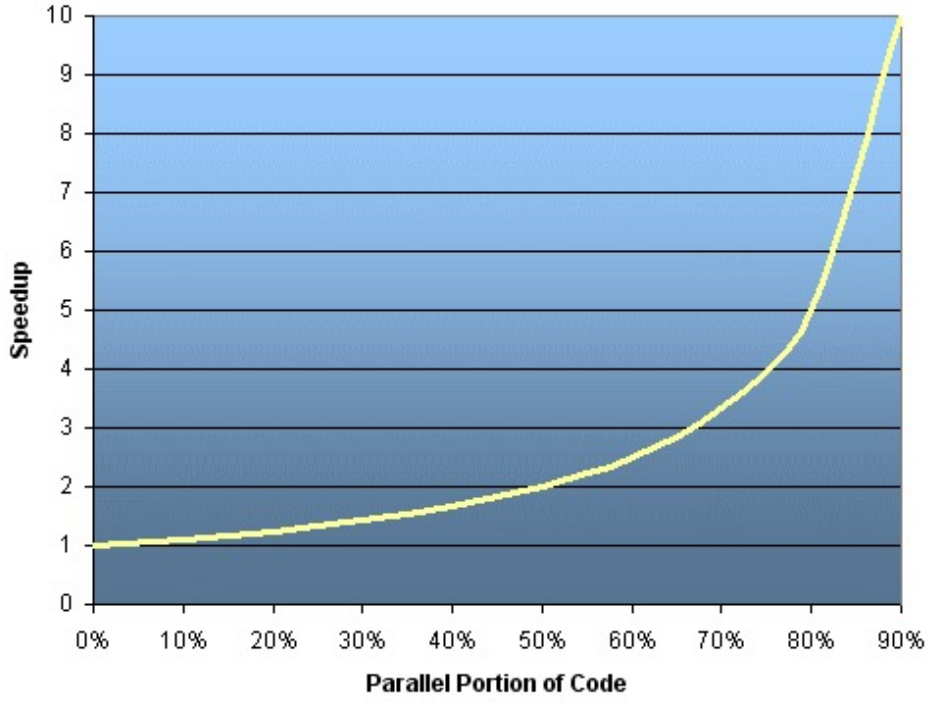
- Good understanding of the problem being solved in parallel
- How much of the problem can be run in parallel
- Bottleneck analysys and profiling gives good picture on scalability of the problem
- We optimize and parallelize parts that consume most of the computing time
- Problem needs to be disected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

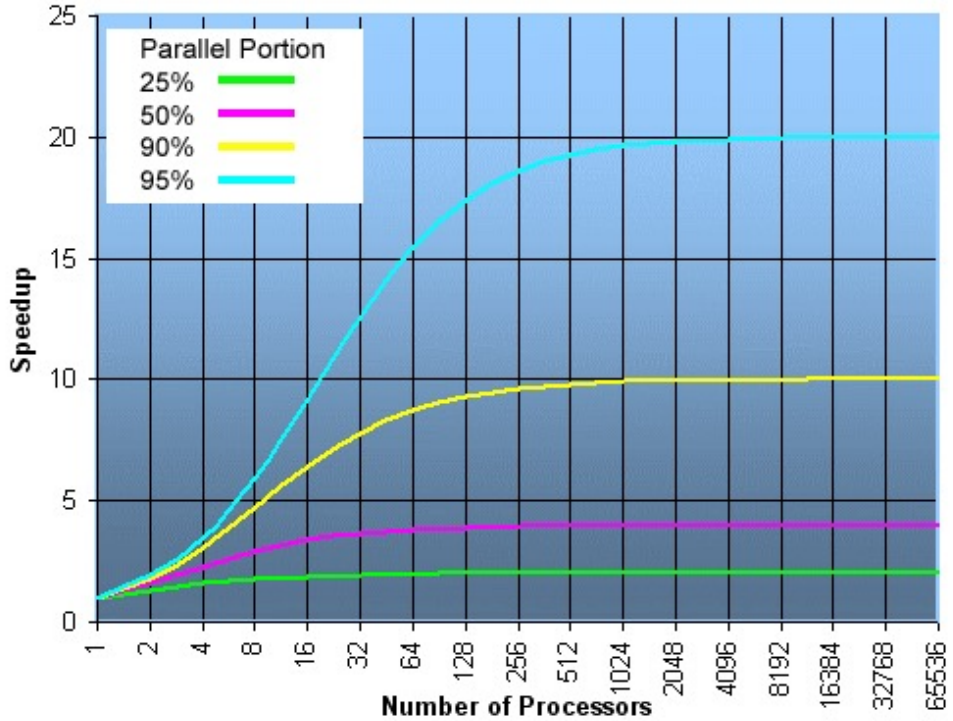
Interprocess communications

- Having little an 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

- Amdahlov 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 milions 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

- 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
- Excercises and extensions of basic ideas
- Instructions available at http://hpc.fs.uni-lj.si/

Basic HPCFS cluster usage

- 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 (interactivelly) 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
```

OpenMP

```
#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
```

Hvala za pozornost!

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