Introduction to High-Performance Computing at DTU

Outline

- ❑ Why HPC
- ❑ HPC at DTU
  - ❑ Accessing the system
  - ❑ Interactive usage
  - ❑ Batch jobs
- ❑ Parallelization and HPC
  - ❑ The concept of parallelism
  - ❑ Parallel Programming Models
  - ❑ Scalability
  - ❑ Running parallel programs

HPC Concept

- ❑Numerical simulations as a third pillar beside theory and experiments in modern science and technology.
- ❑High Performance Computing: combination of hardware resources, efficient algorithms, and implementations.
- ❑Strictly related to (scientific) numerical modeling.

The PITAC report - 2005

President’s Information Technology Advisory Committee, US

- “*Computational science now constitutes what many call the third pillar of the scientific enterprise, a peer alongside theory and physical experimentation.*”
- “*Computational science is a rapidly growing multidisciplinary field that uses advanced computing capabilities to understand and solve complex problems.*”

Scientific Computing

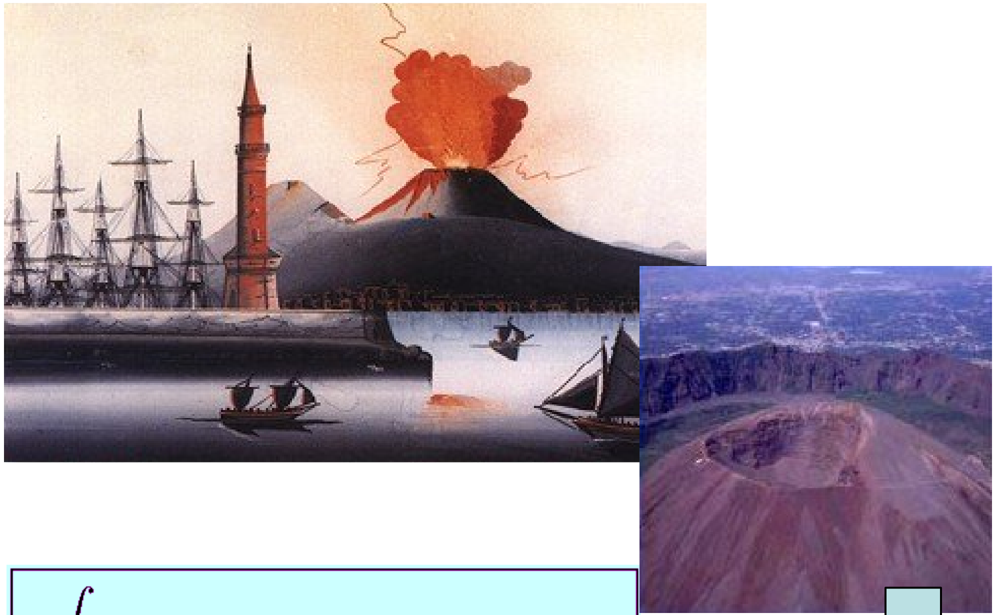
- **Astrophysics**
  - stellar physics
  - galaxy evolution
- **Cryptography**
  - prime numbers
- **Experimental mathematics**
  - fast convergent series
- **Data mining**
  - Google's Page rank
- **Planetary science**
  - geophysics
  - weather forecasts
  - air pollution
  - climate modeling

- **Quantum Physics & Chemistry**
  - superconductivity
  - material science
  - enzymes
- **Bio-informatics**
  - genome research
  - neuroscience
  - heart simulation
- **Engineering design**
  - fluid mechanics, turbulence
  - hydro dynamics
  - structural design
- **Finance**

Computer Simulations

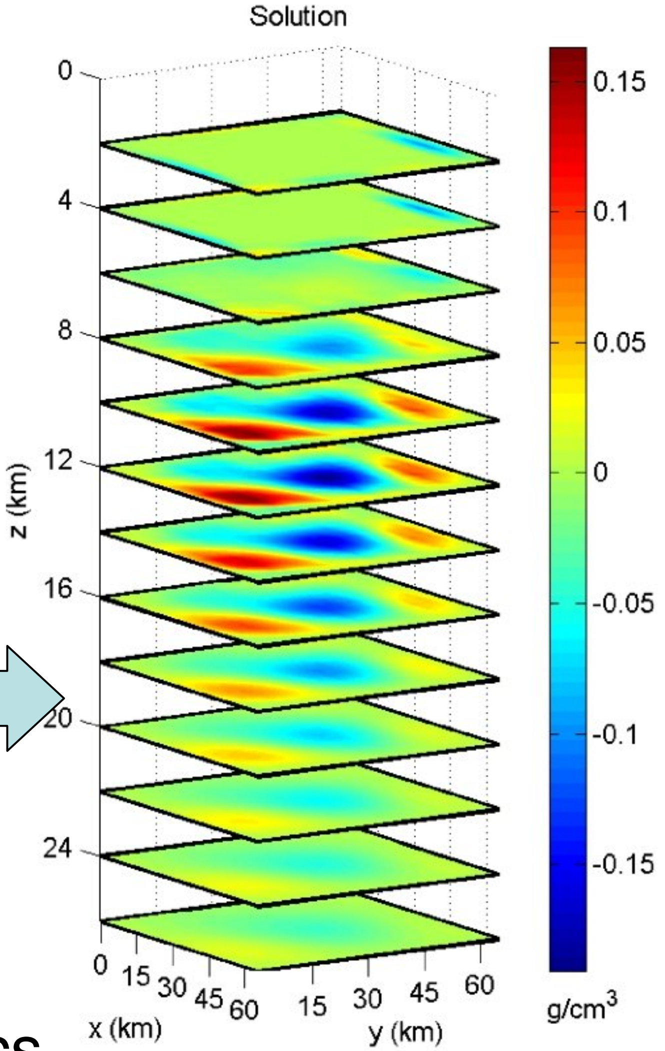
- ❑ Alternative to scale models and lab experiments
  - ❑ faster and cheaper – more flexible
- ❑ Allow a variety of studies
  - ❑ isolated phenomena
  - ❑ change of one parameter at a time
- ❑ Realistic models are large
  - ❑ many model parameters
  - ❑ capture fine details – fine discretization
  - ❑ simulation over a long period of time

Inverse Geomagnetic Problems



$$\boxed{\int_{\Omega} K(\mathbf{s},\mathbf{t})\, f(\mathbf{t})\, d\Omega = g(\mathbf{s})}$$

$$\begin{aligned} f(\mathbf{t}) &= \text{magnetization} \\ g(\mathbf{s}) &= \text{data (anomaly)} \\ K(\mathbf{s},\mathbf{t}) &= \text{magnetic dipole field} \end{aligned}$$



Per Chr. Hansen – DTU Informatics

Wind turbine design - CFD

RISØ DTU – DTU Mechanical Engineering

Topology Optimization

... and Materials:

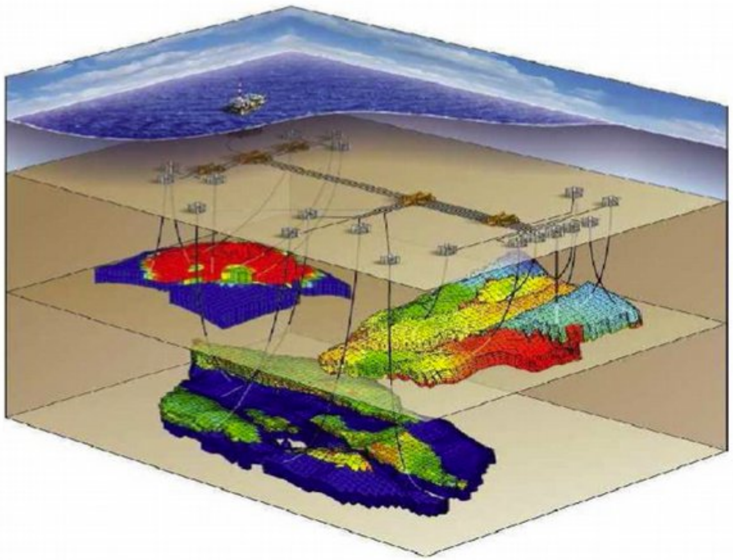
safe and minimum weight structures

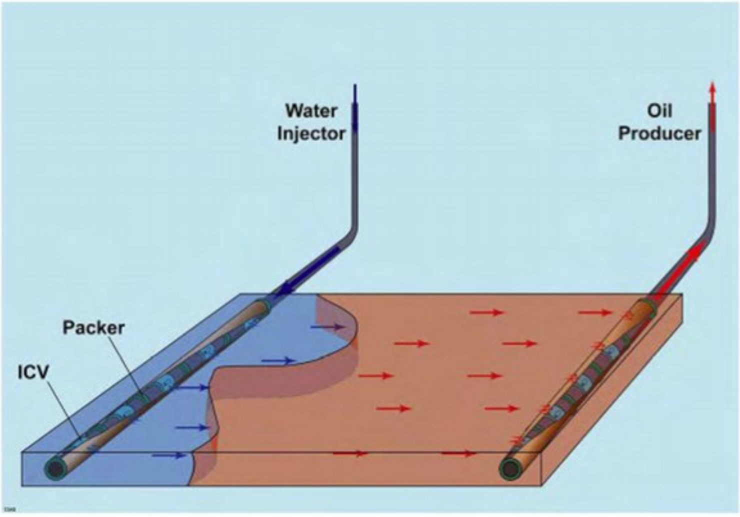
DTU Mathematics –
DTU Mechanical
Engineering

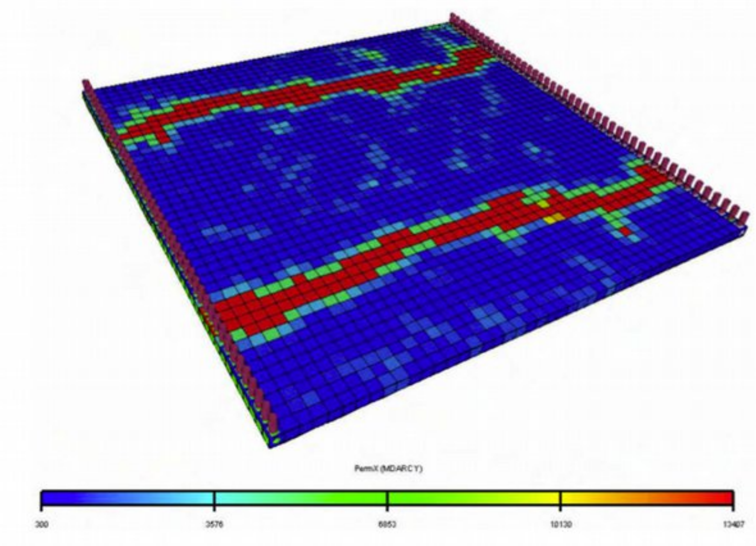
Introduction to HPC

Reservoir Production Optimization

Carsten Völcker,
John Bagterp Jørgensen –
DTU Informatics







HPC Concepts

What all these have in common:

- □An effective algorithm/implementation
- □A suitable hardware infrastructure:
  - □Computational power;
  - □Memory;
  - □Storage (Big data management);
- □Long execution time.

↕

A supercomputer ?

HPC

Two different approaches:

Dedicated architecture/hardware:

- ❑Tuned to the specific problems
- ❑Expensive, not so flexible

Cluster based on commodity hardware

- ❑General purpose
- ❑Flexible
- ❑Not so expensive...

HPC at DTU

HPC Clusters:

- ❑made of ***ordinary*** hardware
- ❑run (also) ***ordinary*** software

Like your personal computer

BUT

it is NOT *personal*

Multi user environment:

Access and resource management policy, in order to satisfy many-user needs.

HPC at DTU

Do you need HPC?

If you have:

- ❑ A simulation that requires a lot of memory
- ❑ A program that could use many cores
- ❑ A program that takes very long to run
- ❑ A special software that is already installed and tuned? (matlab, mathematica, OpenFoam, Gaussian, ...)

Maybe you don't actually ***need*** it but why not?

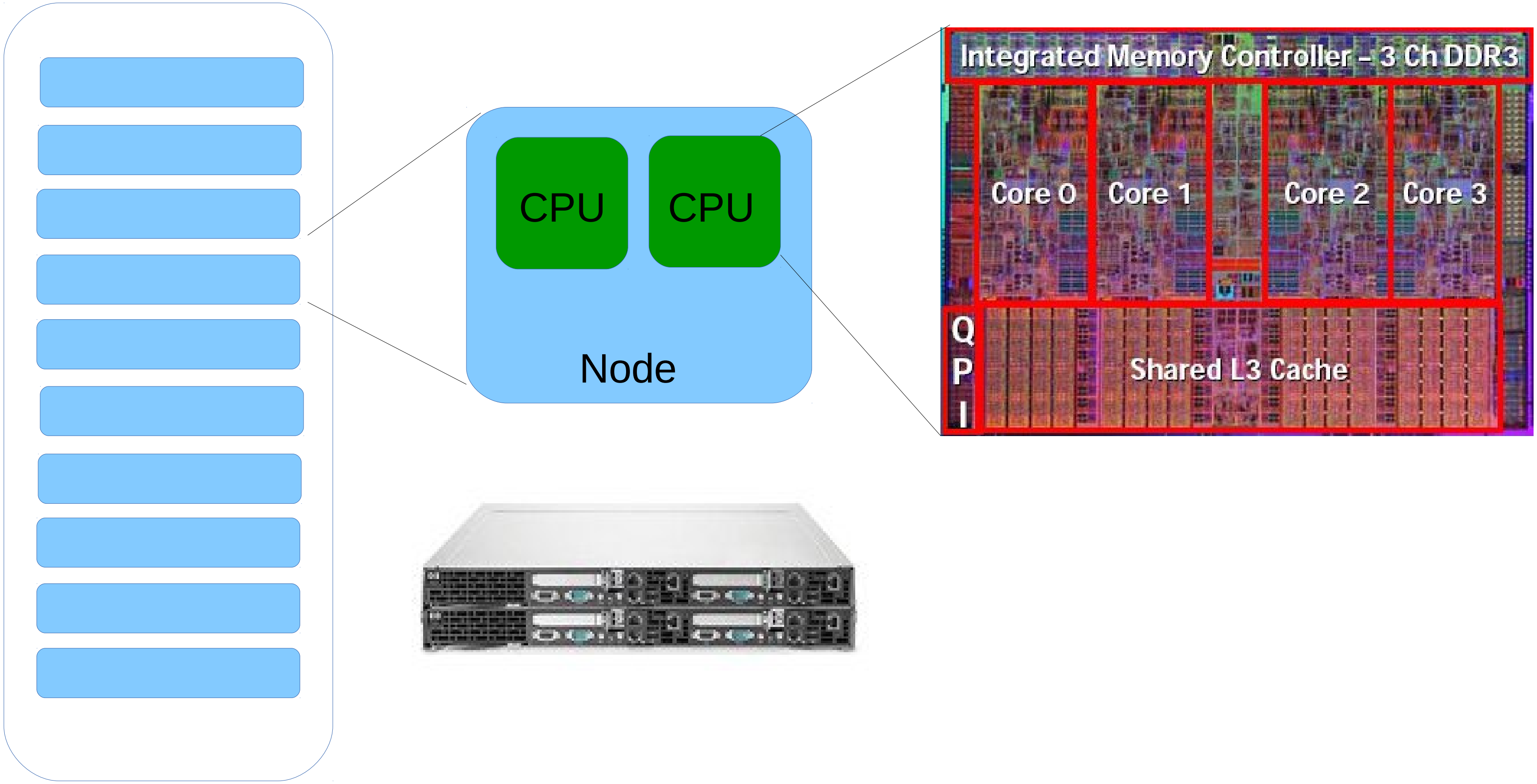
Some recurring terms

Cluster: a set of connected nodes that work together.

Node: a full computer (server) with its own instance of the operating system. It usually has one or more multi-core processors

Core: each independent central processing units in a multi-core processor (+ caches)

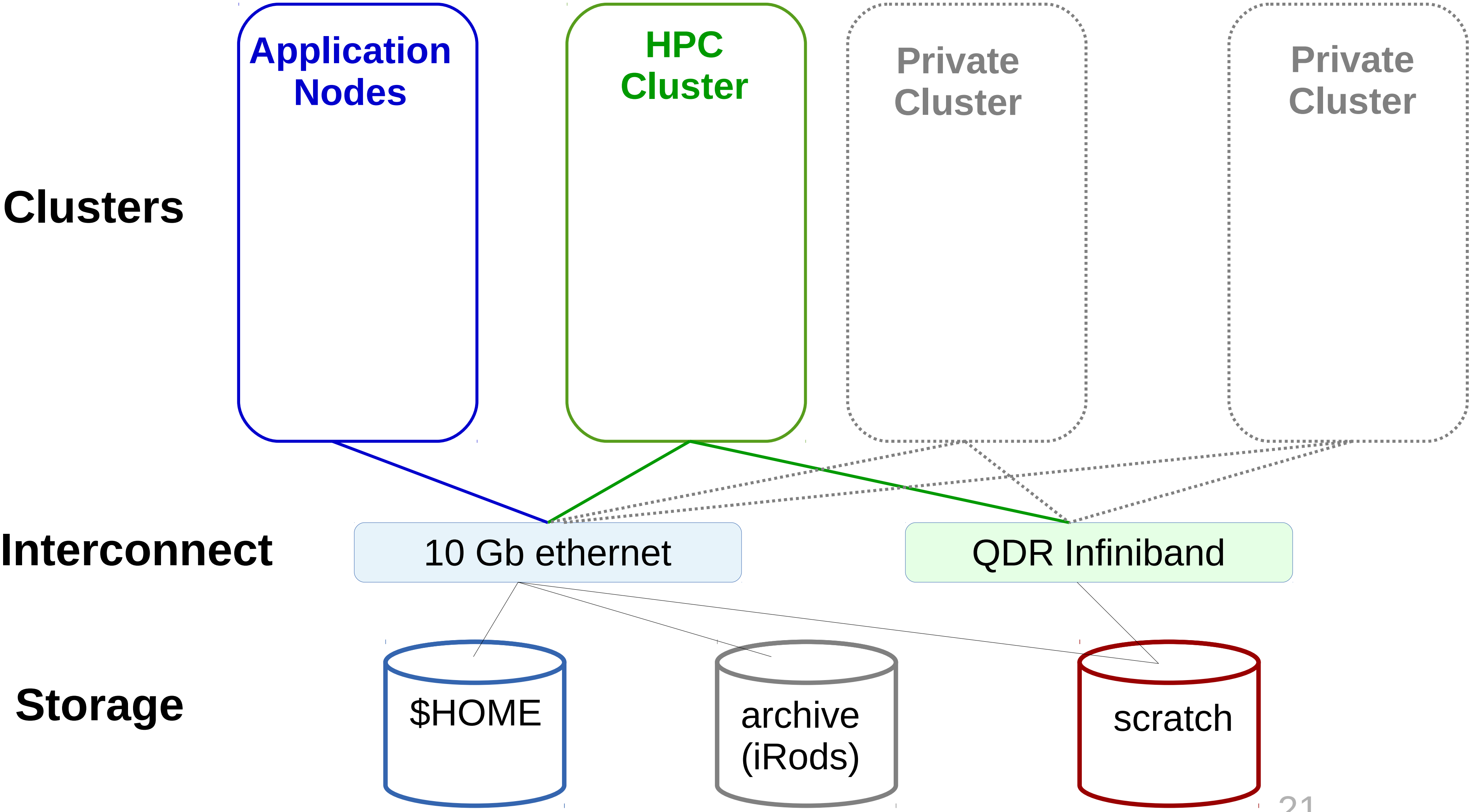
A pictorial view



Cluster/nodes Node/CPUs CPU/cores

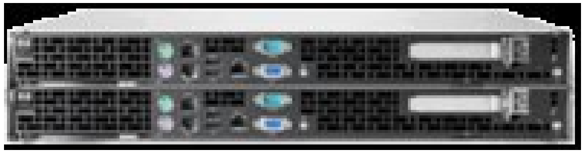
The DTU computer system

System overview

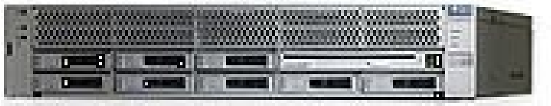


The central DTU UNIX system

- ❑Application servers – x86_64 based:
  - ❑12 HP SL165z G7
  - ❑2x AMD Opteron 6168 (twelve-core, 1.9 GHz, 12 MB L3 cache)
  - ❑ 64 GB memory
  - ❑Scientific Linux 6.4



- ❑Desktop servers (Sun Ray):
  - ❑ 4 Sun SF T5220 (1 US-T2 1166 MHz)
  - ❑Solaris 10



- ❑ 15000+ users (students + employees)

The DTU computer system

- ❑HPC servers:
  - ❑ 64 HP SL2x170z (2x Xeon 5550 2.6 GHz)

    +

    42 IBM NeXtScale nx360 M4 (2x Xeon E5-2680 v2 2.80GHz)
  - ❑+ “private” clusters
  - ❑DTU Compute
  - ❑DTU Nanotech
  - ❑DTU Photonics
  - ❑DTU Chemistry
  - ❑...

The DTU computer system

- ❑HPC servers: 512 Cores, 1.5 TB RAM
  - ❑ 64 HP SL2x170z
    - ❑ 2x Intel Xeon Processor X5550 (quad-core, 2.66 GHz, 8MB L3 Cache)
    - ❑ 24 GB memory
    - ❑ OS: Scientific Linux 6.4
    - ❑ QDR Infiniband interconnect
    - ❑ 500 GB internal SATA (7200 rpm) disk

The DTU computer system

- ❑HPC servers: 840 Cores, 5.25 TB RAM
  - ❑ 42 IBM NeXtScale nx360 M4
    - ❑ 2x Intel Xeon Processor E5-2680v2 (ten-core, 2.80 GHz, 25MB L3 Cache)
    - ❑ 128 GB memory
    - ❑ OS: Scientific Linux 6.4
    - ❑ QDR Infiniband interconnect
    - ❑ 500 GB internal SATA (7200 rpm) disk

Using the HPC at DTU

The DTU computer system

It is a **multi-user** system:

- ❑Need to log in;
- ❑A share of disk space, computational resources;
- ❑(almost) all applications are started by a load-balancing queueing system;

Different:

- ❑CPU types
- ❑clock frequencies
- ❑amounts of RAM
- ❑...

What the user gets

Storage space:

- ❑home directory (30 GB default user quota);
- ❑scratch space (/SCRATCH/$USER upon request);
- ❑/tmp local temporary directory (300 GB)

Your share of resources:

- ❑“flexible” number of nodes/cores/memory;
- ❑Computing time.

Software packages

- ❑Commercial, free, open-source software;
- ❑Compilers, development tools, ...

Accessing the HPC

Students/researchers: DTU userid and password

On Campus:

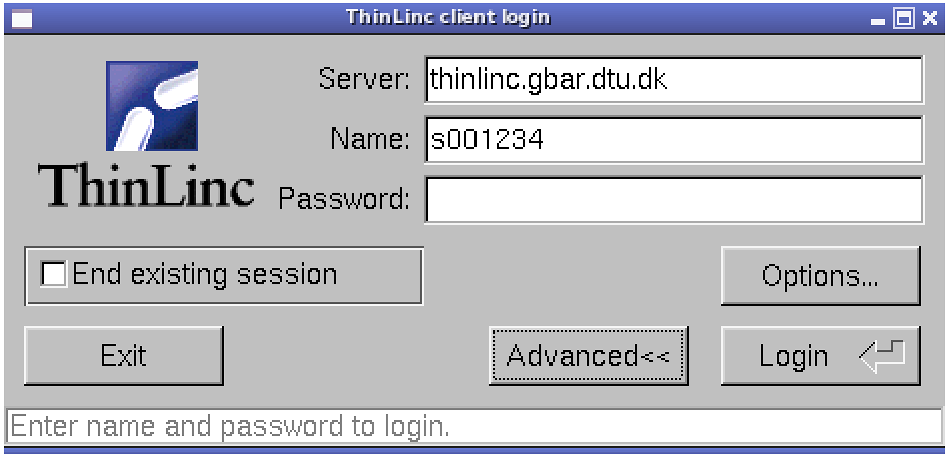
- ❑SunRay terminals at DTU (databars).

Remote access:

- ❑ThinLinc remote desktop session (GUI)
- ❑Secure SHell (ssh) connection (command line)

ThinLinc

A client (Win/Lin/Mac) for graphical login:
(www.thinlinc.com)



ThinLinc client login

Server: thinlinc.gbar.dtu.dk

Name: s001234

Password:

☐ End existing session

Enter name and password to login.

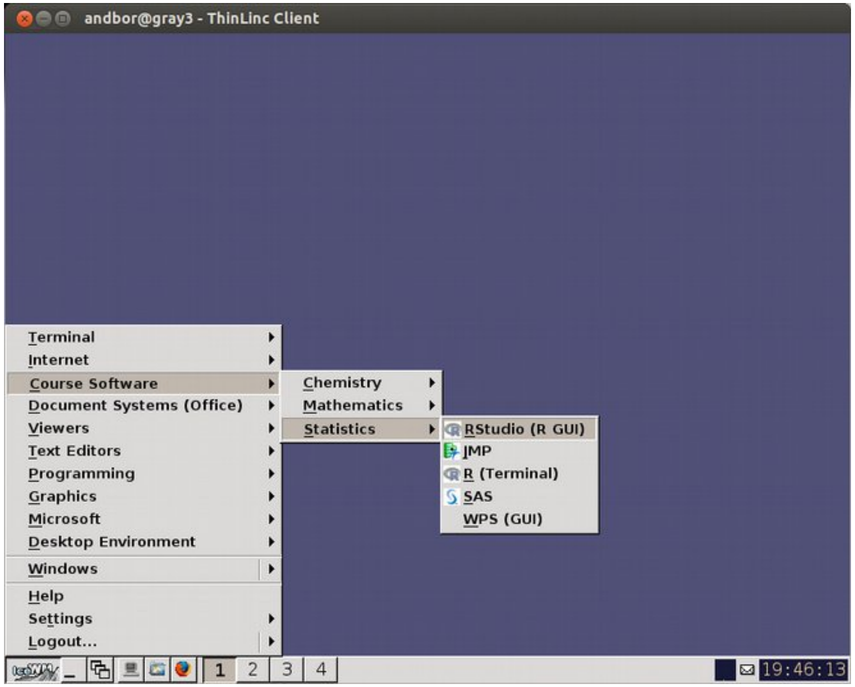
Server:	thinlinc.gbar.dtu.dk
Name:	DTU userid
Password:	DTU password

➡ Login

ThinLinc

You're logged in!

Your remote desktop



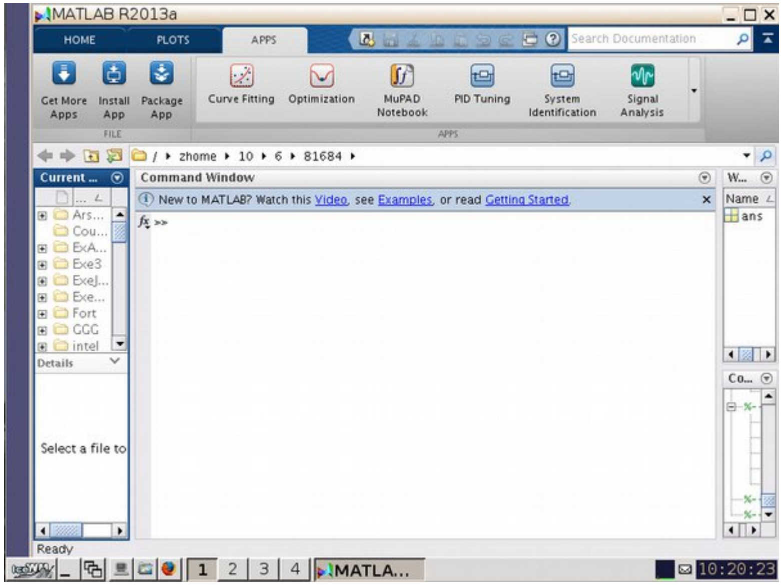
Maybe a bit old-fashioned, but:

- □ you can access your remote files
- □ run all the applications from the menu

Introduction to HPC

ThinLinc

Pick the program from the menu, run it like on a normal computer



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
fx >>
```

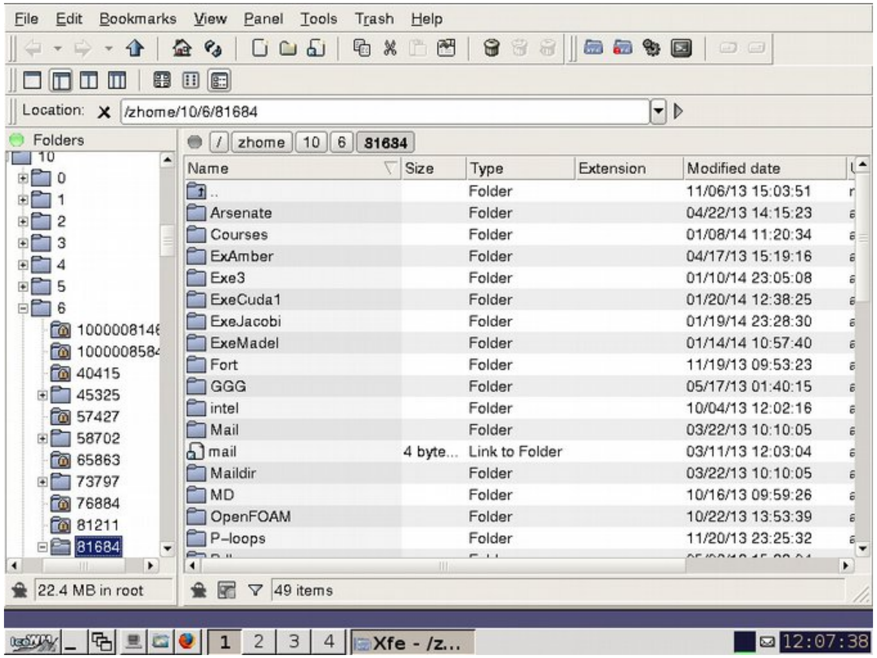
matlab

However:

- ❑ You **cannot access local data**: moving data back and forth;
- ❑ The application are submitted to a queue and and not run directly.

ThinLinc

Your remote data



Location: /zhome/10/6/81684

Name	Size	Type	Extension	Modified date
..		Folder		11/06/13 15:03:51
Arsenate		Folder		04/22/13 14:15:23
Courses		Folder		01/08/14 11:20:34
ExAmber		Folder		04/17/13 15:19:16
Exe3		Folder		01/10/14 23:05:08
ExeCuda1		Folder		01/20/14 12:38:25
ExeJacobi		Folder		01/19/14 23:28:30
ExeMadel		Folder		01/14/14 10:57:40
Fort		Folder		11/19/13 09:53:23
GGG		Folder		05/17/13 01:40:15
intel		Folder		10/04/13 12:02:16
Mail		Folder		03/22/13 10:10:05
mail	4 byte...	Link to Folder		03/11/13 12:03:04
Maildir		Folder		03/22/13 10:10:05
MD		Folder		10/16/13 09:59:26
OpenFOAM		Folder		10/22/13 13:53:39
P–loops		Folder		11/20/13 23:25:32

22.4 MB in root

49 items

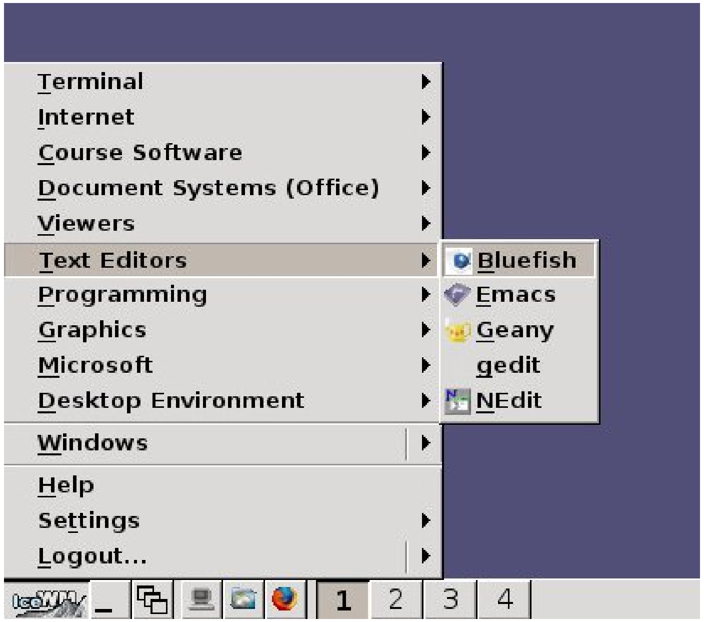
Xfe file manager

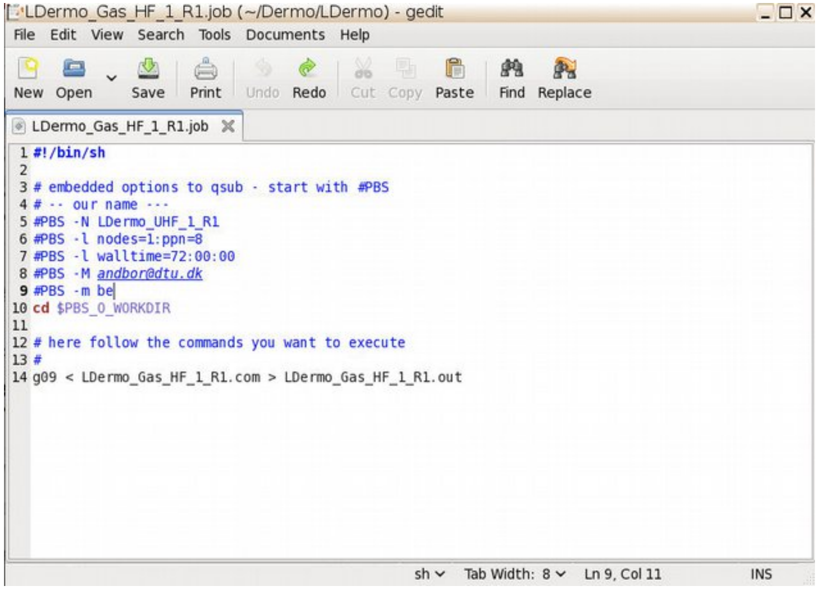
- ❑ Access your data on the remote machine(s);
- ❑ Remember: you have to move data to/from your private machine!

Introduction to HPC

ThinLinc

Text editors: gedit, nedit, emacs



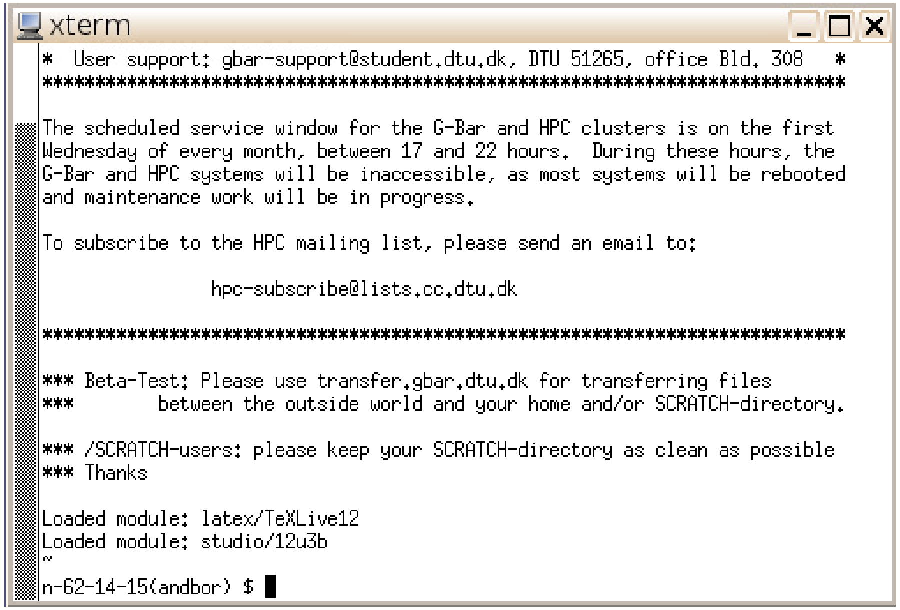


```
#!/bin/sh

# embedded options to qsub - start with #PBS
# -- our name ---
#PBS -N LDermo_UHF_1_R1
#PBS -l nodes=1:ppn=8
#PBS -l walltime=72:00:00
#PBS -M andbor@dtu.dk
#PBS -m be
cd $PBS_O_WORKDIR

# here follow the commands you want to execute
#
g09 < LDermo_Gas_HF_1_R1.com > LDermo_Gas_HF_1_R1.out
```

Use plain text editors

ThinLinc

A special program: the terminal:



```
*  User support: gbar-support@student.dtu.dk, DTU 51265, office Bld. 308    *
****************************************************************************

The scheduled service window for the G-Bar and HPC clusters is on the first
Wednesday of every month, between 17 and 22 hours.  During these hours, the
G-Bar and HPC systems will be inaccessible, as most systems will be rebooted
and maintenance work will be in progress.

To subscribe to the HPC mailing list, please send an email to:

                hpc-subscribe@lists.cc.dtu.dk

****************************************************************************

*** Beta-Test: Please use transfer.gbar.dtu.dk for transferring files
***        between the outside world and your home and/or SCRATCH-directory.

*** /SCRATCH-users: please keep your SCRATCH-directory as clean as possible
*** Thanks

Loaded module: latex/TeXLive12
Loaded module: studio/12u3b
~
n-62-14-15(andbor) $ 
```

xterm

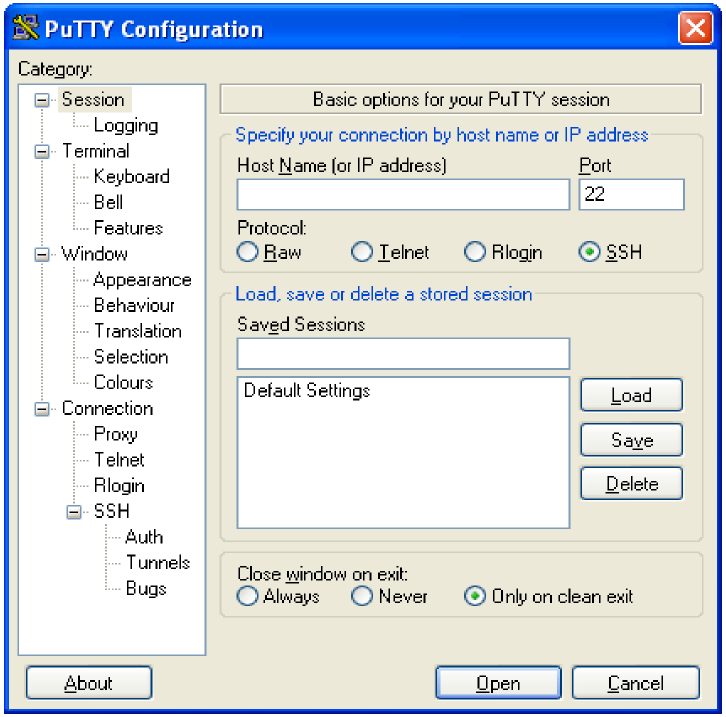
Command line interface:

- ❑ You can do everything from the command line;
- ❑ It is the only way to submit script for batch execution;

SSH access

On Windows use:

PuTTY


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address)

Port: 22

Protocol: ○ Raw ○ Telnet ○ Rlogin ● SSH

Load, save or delete a stored session

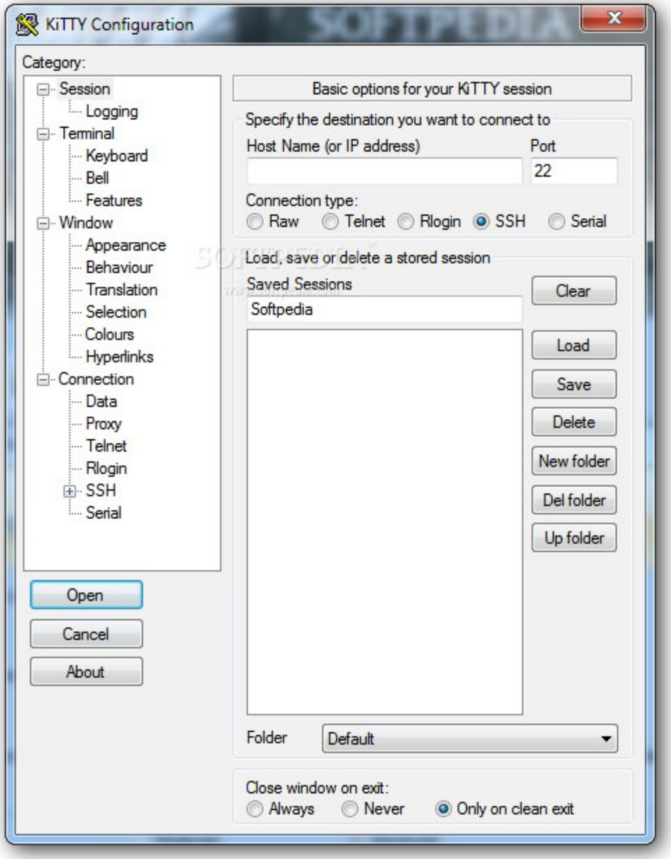
Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

http://www.chiark.greenend.org.uk/~sgtatham/putty/
Help:
http://www.gbar.dtu.dk/index.php/faq/27-putty

KiTTY


KiTTY Configuration

Category:

Basic options for your KiTTY session

Specify the destination you want to connect to

Host Name (or IP address)

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: Softpedia

Folder: Default

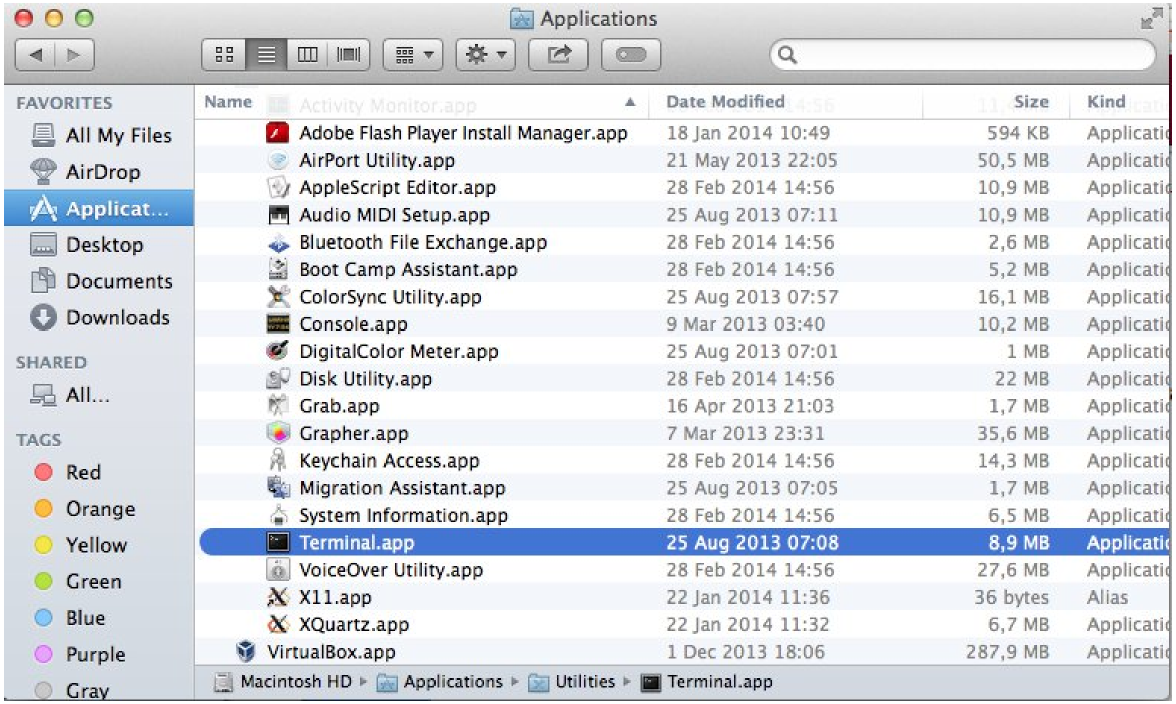
Close window on exit: ○ Always ○ Never ● Only on clean exit

http://www.9bis.net/kitty/

SSH access

You need a terminal (Linux/MacOsX) or an ssh client (Windows)

On Mac:
Application-> Terminal.app



Name	Date Modified	Size	Kind
Adobe Flash Player Install Manager.app	18 Jan 2014 10:49	594 KB	Applicati
AirPort Utility.app	21 May 2013 22:05	50,5 MB	Applicati
AppleScript Editor.app	28 Feb 2014 14:56	10,9 MB	Applicati
Audio MIDI Setup.app	25 Aug 2013 07:11	10,9 MB	Applicati
Bluetooth File Exchange.app	28 Feb 2014 14:56	2,6 MB	Applicati
Boot Camp Assistant.app	28 Feb 2014 14:56	5,2 MB	Applicati
ColorSync Utility.app	25 Aug 2013 07:57	16,1 MB	Applicati
Console.app	9 Mar 2013 03:40	10,2 MB	Applicati
DigitalColor Meter.app	25 Aug 2013 07:01	1 MB	Applicati
Disk Utility.app	28 Feb 2014 14:56	22 MB	Applicati
Grab.app	16 Apr 2013 21:03	1,7 MB	Applicati
Grapher.app	7 Mar 2013 23:31	35,6 MB	Applicati
Keychain Access.app	28 Feb 2014 14:56	14,3 MB	Applicati
Migration Assistant.app	25 Aug 2013 07:05	1,7 MB	Applicati
System Information.app	28 Feb 2014 14:56	6,5 MB	Applicati
Terminal.app	25 Aug 2013 07:08	8,9 MB	Applicati
VoiceOver Utility.app	28 Feb 2014 14:56	27,6 MB	Applicati
X11.app	22 Jan 2014 11:36	36 bytes	Alias
XQuartz.app	22 Jan 2014 11:32	6,7 MB	Applicati
VirtualBox.app	1 Dec 2013 18:06	287,9 MB	Applicati

SSH access

From a terminal on your private machine:

```
comp-nbandbor:~ andbor$
```

ssh to the login node:
hpc-fe.gbar.dtu.dk

Syntax: `ssh userid@hpc-fe.gbar.dtu.dk`

- ❑ Case sensitive!
- ❑ Add the option `-X` for X11 forwarding
- ❑ Full access to machine resources

NEVER run programs on the login node

Command line

If you need to do some work, switch immediately to a work-node:

- ❑ **`qrsh`**
  to access a HPC interactive node: Xeon CPU
  fast connection to /SCRATCH (infiniband).

- ❑ **`linuxsh`**
  to access a Gbar node: Opteron CPU
  slow connection to /SCRATCH (10 Gb Ethernet).

Always add the `-X` option for X11 forwarding

NOTE: /home and /scratch are automatically accessible from any node

Command line

Pros:

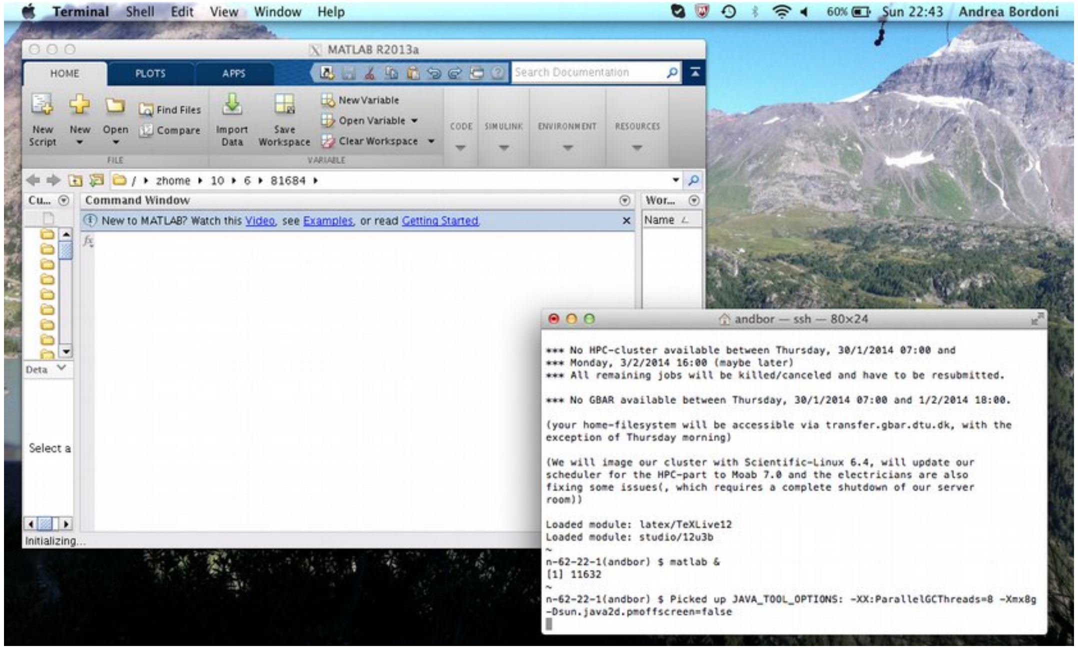
- ❑Very fast (once you know the basic commands)
- ❑Powerful and flexible
- ❑Low bandwidth required (text only)
- ❑Exports graphics (Xwindow)

Cons:

- ❑Not always user-friendly
- ❑A bit of practice is needed

Command line

```
$ ssh -X userid@hpc-fe.gbar.dtu.dk
$ linuxsh -X
$ matlab &
```



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
*** No HPC-cluster available between Thursday, 30/1/2014 07:00 and
*** Monday, 3/2/2014 16:00 (maybe later)
*** All remaining jobs will be killed/canceled and have to be resubmitted.

*** No GBAR available between Thursday, 30/1/2014 07:00 and 1/2/2014 18:00.

(your home-filesystem will be accessible via transfer.gbar.dtu.dk, with the
exception of Thursday morning)

(We will image our cluster with Scientific-Linux 6.4, will update our
scheduler for the HPC-part to Moab 7.0 and the electricians are also
fixing some issues(, which requires a complete shutdown of our server
room))

Loaded module: latex/TeXLive12
Loaded module: studio/12u3b
~
n-62-22-1(andbor) $ matlab &
[1] 11632
~
n-62-22-1(andbor) $ Picked up JAVA_TOOL_OPTIONS: -XX:ParallelGCThreads=8 -Xmx8g
-Dsun.java2d.pmoffscreen=false
```

(Exporting matlab GUI: not recommended)

Command line

Control the system via **text commands**:

- ❑Manage files and directories:

  `pwd, cd, ls, mkdir, cp, mv, rm, ...`

- ❑Start programs (even with GUI);
- ❑Check system status;
- ❑Transfer files over the network:

  `scp, rsync`

All you need is to learn some basic commands

(http://www.gbar.dtu.dk/index.php/faq/48-unix-commands)

Command line

Commands accept arguments and options:

- Arguments: object affected by the command

```
cp source_file destination_file
```

- Options: affect the command behavior:

```
cp -p source_file destination_file
-p  :  preserve file attributes
```

  - Note:
    - Options starts usually with a dash symbol -
    - Order in the sequence of arguments/options

Moving files

From private machine to cluster:

```
 scp  file userid@transfer.gbar.dtu.dk:directory/
```

From cluster to private machine:

```
scp  userid@transfer.gbar.dtu.dk:directory/file .
```

`transfer.gbar.dtu.dk` is a server dedicated to file transfer

If you need to transfer very large amount of data/file:

rsync (http://en.wikipedia.org/wiki/Rsync)

Windows: use Winscp (http://winscp.net/eng/index.php)

Instructions: (http://www.gbar.dtu.dk/faq/25-winscp)

Command line tips

Man pages: type **man** command to access the manual pages:

```
$ man scp
```

```
SCP(1)                BSD General Commands Manual
      SCP(1)

NAME
     scp - secure copy (remote file copy program)

SYNOPSIS
     scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i
identity_file]
         [-l limit] [-o ssh_option] [-P port] [-S program]
         [[user@]host1:]file1 ... [[user@]host2:]file2
```

Command line tips

Auto-completion:

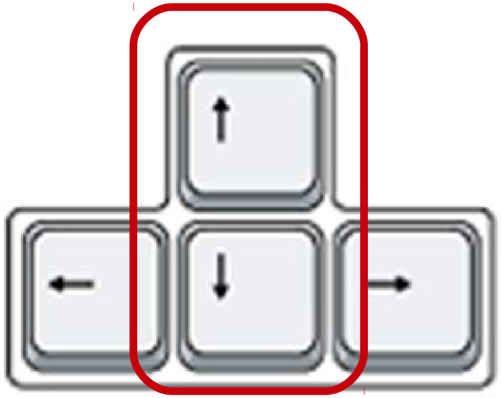
press TAB when writing a command

```
hpc-fe1(andbor) $ mat
match_parens   math80          matlab       atlab73     matlab800
matextract     mathematica     matlab710    matlab75    matlab810
matgen         mathematica60   matlab711     matlab76   matlab820
math           mathematica70   matlab712     matlab77   matmerge
math60         mathematica80   matlab713     matlab78   matmul
math70         mathspic        matlab714    matlab79    mattrib
```

Command line tips

Command history

Scroll the previous commands with up and down arrows:



- ❑Correct a command failed for typos;
- ❑Recall a command you used last time you logged in;

Command line tips

Scripting

Basic:

- ❑combine many text command in a text file (batch file)

```
#!/bin/bash
date > List
ls -l >> List
echo List Done
```

← Test.sh

- ❑Make the file executable;

```
$ chmod +x Test.sh
```

- ❑Run it

```
$ ./Test.sh
```

Command line tips

Scripting

The shell commands is a real language:

a script can be a complex program.

Modules

Introduction to HPC

Modules

General purpose HPC:

- ❑Large software stack installed
- ❑Different users may need different tools, or different versions of the same software;
- ❑Potential conflicts between the different programs

Modular approach:

- ❑Software packages/tools are available as modules
- ❑Environment variables are set accordingly
- ❑Conflicts are avoided

http://www.gbar.dtu.dk/index.php/faq/83-modules

Modules

The user loads and unloads modules.

Command line: **`$ module [option]`**

`list:` Shows the loaded modules

```
$ module list
Currently Loaded Modulefiles:
1) latex/TeXLive12    2) studio/12u3b
```

`avail`: To have a list of the available modules (long list)

```
$ module avail
… … … … … …
OpenFoam/2.2.2/gcc-4.7.2-openmpi mpi/gcc-4.7.2-openmpi-1.6.3

… … … … … …
```

Modules

`load:` loads module (remember autocompletion!)

```
$ module load OpenFoam/2.2.2/gcc-4.7.2-openmpi
Loaded module: OpenFoam/2.2.2/gcc-4.7.2-openmpi
```

Other useful options:

- **`help`**: all possible options
- **`whatis`**: short description of module
- **`show`**: prints the modification the module does
- **`unload`**: unload the module

HPC at DTU: Batch jobs

Batch jobs

Typical HPC usage: unattended execution

- ❑ prepare the program so that it can run without any intervention;
- ❑ (try to) estimate the computational resources and time needed;
- ❑ submit the job to the system.

The Resource Manager (RM):

- ❑ checks the resources available;
- ❑ schedules the execution of the jobs of the users, in a queue, in order to optimize the cluster usage.

Resource Manager

The user specifies the resources needed, e.g.

- ❑ # of nodes/cores
- ❑ amount of memory
- ❑ expected run time (wall-clock time)
- ❑ CPU-type
- ❑ other resources, like disk space, GPUs, etc

This is done in a **job script**

The Resource Manager (MOAB (scheduler) + Torque (resource manager) relies on these information, so

BE ACCURATE: a misuse of the resources affect all the users of the system (and your jobs!).

Introduction to HPC

Your program

Program has to be run without intervention:

□ No GUI;

□ No waiting for user input:

□ Login to one of the linux nodes: `qrsh`
(hpc_interactive queue, same environment as the production environment)

□ Test the correct command(s) for your job execution. Better: prepare a shell script.

Introduction to HPC

Resources

Decide:

- □ How many cores/processors;
- □ How much memory
- □ Disk space
- □ Execution time

⇨ Write your job script

Summary

Prepare:

- ❑Prepare all the program needs for the execution;
- ❑Estimate the resources you need
- ❑Prepare the job script asking for the resources AND for running your program

Run:

- ❑Submit your job script
- ❑Check your job status

And then wait patiently for the results

Job Script

The job script is a simple text file.

Two sections:

First: all the information for the scheduler (#PBS)

Last: your command(s) for the actual execution.

NOTES:

- ❑ only the options for the scheduler that appear BEFORE the first command will be considered. The others will be simply ignored
- ❑ Do not use special characters or spaces in batch filenames or arguments

Submit

From the directory where you have your files, type

```
$ qsub submit.sh
```

Where **`submit.sh`** is your job file.

The scheduler reads its options (#PBS), and assigns the job to the right queue.

NOTE: your program will start when there are available resources!

Learn by examples!

Batch Jobs

The simplest job script:

```
#!/bin/sh
sleep 60
```

submit.sh

```
$ qsub submit.sh
611064.hpc-fe1
```

```
$ qstat

                                                  Req'd   Elap
Job ID          Username Queue    Jobname         Time  S Time
--------------- -------- -------------- ----- - -----
611064.hpc-fe1  gbarbd   hpc      submit.sh         --  R   --
```

```
$ ls -g
total 3
-rw-r--r-- 1 gbar 19 Jan  3 17:21 submit.sh
-rw------- 1 gbar  0 Jan  3 17:21 submit.sh.e611064
-rw------- 1 gbar  0 Jan  3 17:21 submit.sh.o611064
```

Batch Jobs

A less simple job script:

```
#!/bin/sh
#PBS -N sleeper
#PBS -q hpc
#PBS -l walltime=2:00
cd $PBS_O_WORKDIR

sleep 60
```

Overrides the default values

```
$ qsub submit.sh
611070.hpc-fe1
```

```
$ ls -g
total 3
-rw-r--r-- 1 gbar 19 Jan  3 17:31 submit.sh
-rw------- 1 gbar  0 Jan  3 17:34 sleeper.e611070
-rw------- 1 gbar  0 Jan  3 17:34 sleeper.o611070
```

Introduction to HPC

Batch Jobs

Another simple job script:

```
#!/bin/sh
#PBS -N sleeper
#PBS -o $PBS_JOBNAME.$PBS_JOBID.out
#PBS -e $PBS_JOBNAME.$PBS_JOBID.err
cd $PBS_O_WORKDIR

echo "Just a minute ..."
Sleep 60
```

```
$ qsub submit.sh
611075.hpc-fe1
```

Job ID

```
$ ls -g
total 3
-rw-r--r-- 1 19 Jan  3 17:41 submit.sh
-rw------- 1  0 Jan  3 17:45 sleeper.611075.hpc-fe1.err
-rw------- 1 18 Jan  3 17:45 sleeper.611075.hpc-fe1.out
```

Batch Jobs

A *test* job script:

```
#!/bin/sh
#PBS -N Test_W2
#PBS -q hpc
#PBS -l walltime=5:00
# -- number of processors/cores/nodes --
#PBS -l nodes=1:ppn=2
#PBS -M s012345@dtu.dk
#PBS -m abe

cd $PBS_O_WORKDIR
sleep 120
pwd > Out_Test.txt
printenv | grep PBS >> Out_Test.txt
```

Time Format:
DD:HH:MM:SS

Prints the environment variables

```
$ ls -g
total 3
-rw-r--r-- 1 19 Jan  3 17:41 submit.sh
-rw------- 1  0 Jan  3 17:45 sleeper.611075.hpc-fe1.err
-rw------- 1 18 Jan  3 17:45 sleeper.611075.hpc-fe1.out
```

Modules and batch jobs

When running a batch job, only the default modules are loaded.

If you need a specific modules, **add the corresponding load command** in your job file:

```
# -- run in the current working (submission) directory --
cd $PBS_O_WORKDIR
# – here load modules you need
module load mpi/intel
# here follow the commands you want to execute
myapplication.x < input.in > output.out
```

Managing jobs

Commands to access/retrieve infos on the jobs:

- ❑ qsub batch_file: submit job
- ❑ qstat: show status of batch jobs
- ❑ showstart: show expected start/end date/time
- ❑ checkjob <jobid>: display job status and more
- ❑ qdel <jobid>: delete your own job from the queue
- ❑ showq: display general informations about “all” the jobs

As normal shell commands, they accept many options:

- ❑ Use the man <command> to find out!
- ❑ see http://www.cc.dtu.dk/ under HPC

Checking system status

A couple of useful commands:

- ❑classstat (queue): summary of the queue status
- ❑nodestat (queue): show details on the status of single nodes in the queue

NOTE: these two commands only work on the front-end

qstat

```
$ qstat
```

```
Job ID                    Name             User       Time Use S Queue

-------------      ---------------- ----------- -------- - -----

3597252.hpc-fe1    ...-COOETC_R9-16    s012345   3401:00: R hpc
3640759.hpc-fe1    xterm-linux         s012345          0 R app
```

showstart

```
$ showstart <jobid>
```

```
job <jobid> requires 8 procs for 3:00:00:00

Estimated Rsv based start in           -00:47:16 on Fri Apr 18 11:50:00

Estimated Rsv based completion in     2:23:12:44 on Mon Apr 21 11:50:00

Best Partition: hpc-fe1
```

checkjob

```
$ checkjob <jobid>
```

```
Job <jobid>
AName: Ldermo_UHF_1
State: Running
Creds:  user:userid  group:fys  class:hpc  qos:hpc_longhours
WallTime:   00:50:33 of 3:00:00:00
SubmitTime: Fri Apr 18 11:49:59
(Time Queued  Total: 00:00:01  Eligible: 00:00:01)
StartTime: Fri Apr 18 11:50:00
Total Requested Tasks: 8
Req[0]  TaskCount: 8  Partition: hpc-fe1
Dedicated Resources Per Task: PROCS: 1  MEM: 2048M
NodeSet=ONEOF:FEATURE:hpc_node:fullhpc_node:ibm_fullhpc_node
Allocated Nodes:
[n-62-13-2:8]
Notification Events: JobStart,JobEnd  Notification Address:
s123456@xxx.dk
IWD:            /zhome/10/6/81684/Dermo/Ldermo
StartCount:     1
Flags:          RESTARTABLE
Attr:           checkpoint
StartPriority:  120
Reservation '4242173' (-00:50:55 -> 2:23:09:05  Duration: 3:00:00:00)
```

classstat

```
hpc-fe1(userid)$ classstat
```

```
queue                    total  used avail
----------------------------------------
hpc                       1000   315   685
fotonano                   636   450   186
mek                        120    64    56
topopt                     300     0   300
dyna                       144   128    16
hpc_interactive             64     3    61
cmp_interactive             16     0    16
k40_interactive             12     0    12
course_02614               176     0   176
compile                     16     0    16
compute                    384   159   225
computebigmem               96     0    96
visual                      64     0    64
ibm                        200     0   200
ibmtest                     80     0    80
~
```

nodestat

```
hpc-fe1(userid)$ nodestat compute
```

```
Name                    State    Procs     Load
n-62-18-20               Idle    16:16     0.24
n-62-18-21               Idle    16:16     0.45
n-62-18-22               Idle    16:16     0.38
n-62-18-23               Idle    16:16     0.40
...
n-62-18-30               Idle    14:16     2.14
n-62-18-31               Idle     9:16     7.29
n-62-18-32            Running     7:16     8.90
n-62-18-33            Running     6:16    10.31
n-62-18-34            Running     2:16    14.41
n-62-18-36               Busy     0:16    12.45
n-62-18-37               Busy     0:16    12.78
n-62-18-38               Busy     0:16    13.18
n-62-18-39               Busy     0:16    13.52
n-62-18-40               Busy     0:16    12.88
n-62-18-41            Running     1:16    15.19
n-62-18-42               Busy     0:16    16.38
~
```

Parallelization and HPC

Outline

- ❑Parallelism ...
- ❑Parallelization and HPC
  - ❑Main concepts
  - ❑Resources for parallel computing
  - ❑Programming models
  - ❑Parallel costs
  - ❑Scalability and metrics
  - ❑Running parallel programs

Introduction to HPC

What is Parallelization?

An attempt of a definition:

> “*Something*” is parallel, if there is a certain level of independence in the order of operations

“*Something*” can be:

- ► A collection of program statements
- ► An algorithm
- ► A part of your program
- ► The problem you are trying to solve

granularity

Introduction to HPC

Parallelism in the problem

Problems have certain amount of potential parallelism.

How much parallelism is present in the solution?

Depends on:

- □the available resources
- □the algorithm
- □the tools

Introduction to HPC

Example: Cooking

Prepare three different first courses.

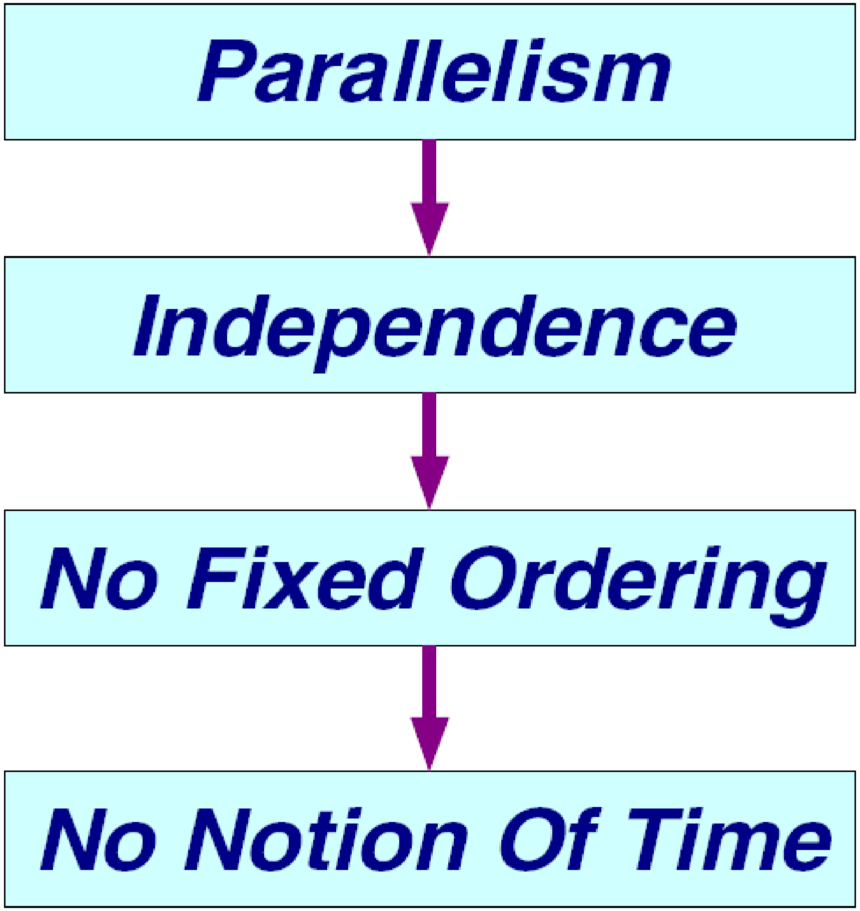
Independent tasks, but:

- □ how may people are working?
- □ are there enough kitchen tools for all?
- □ do the preparation require the same amount of time?

⬇

The solution is not independent from the available resources

Parallelism – when?



Parallelism

Independence

No Fixed Ordering

No Notion Of Time

Something that does not follow this rule is not parallel !!!

From problem to solution

Analyze the problem:

- Split problem in independent tasks
- Find solutions for each task

Resource limitations:

- (Partial) sequentialization:
 impose an artificial order to independent tasks

Limiting cases

Sequential:

- No parallelism
- Order

OR

- No parallel resources (Single core, limited memory)

Embarassingly parallel:

- Completely independent tasks
- No order

AND

- Enough parallel resources (Multi/many cores, enough memory)

Sequential solution

Parallel solution

Real world

The problem can be split in a set of tasks, **but** they are **not all independent**

AND/OR

There are resources, but are limited

- Multi/many cores machine;
- Limited memory;
- Limited communication bandwidth.

Look for a good compromise

HPC cluster and parallelism

Certain amount of resources:

- Multi core nodes with shared memory;
- Cluster of nodes with fast interconnect;
- Accelerators (Nvidia, XeonPhi).

⬇

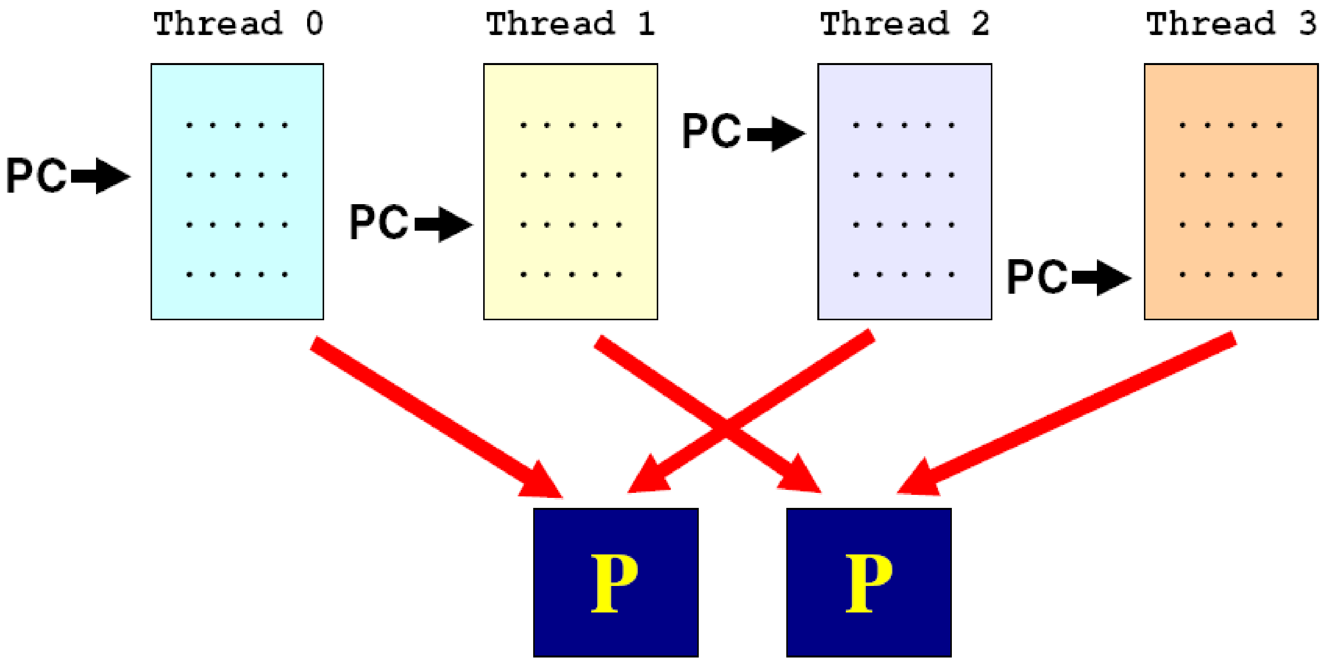
Potential for exploiting parallelism:

- Within a single node (shared memory)
- Between nodes (explicit communication)
- Hybrid

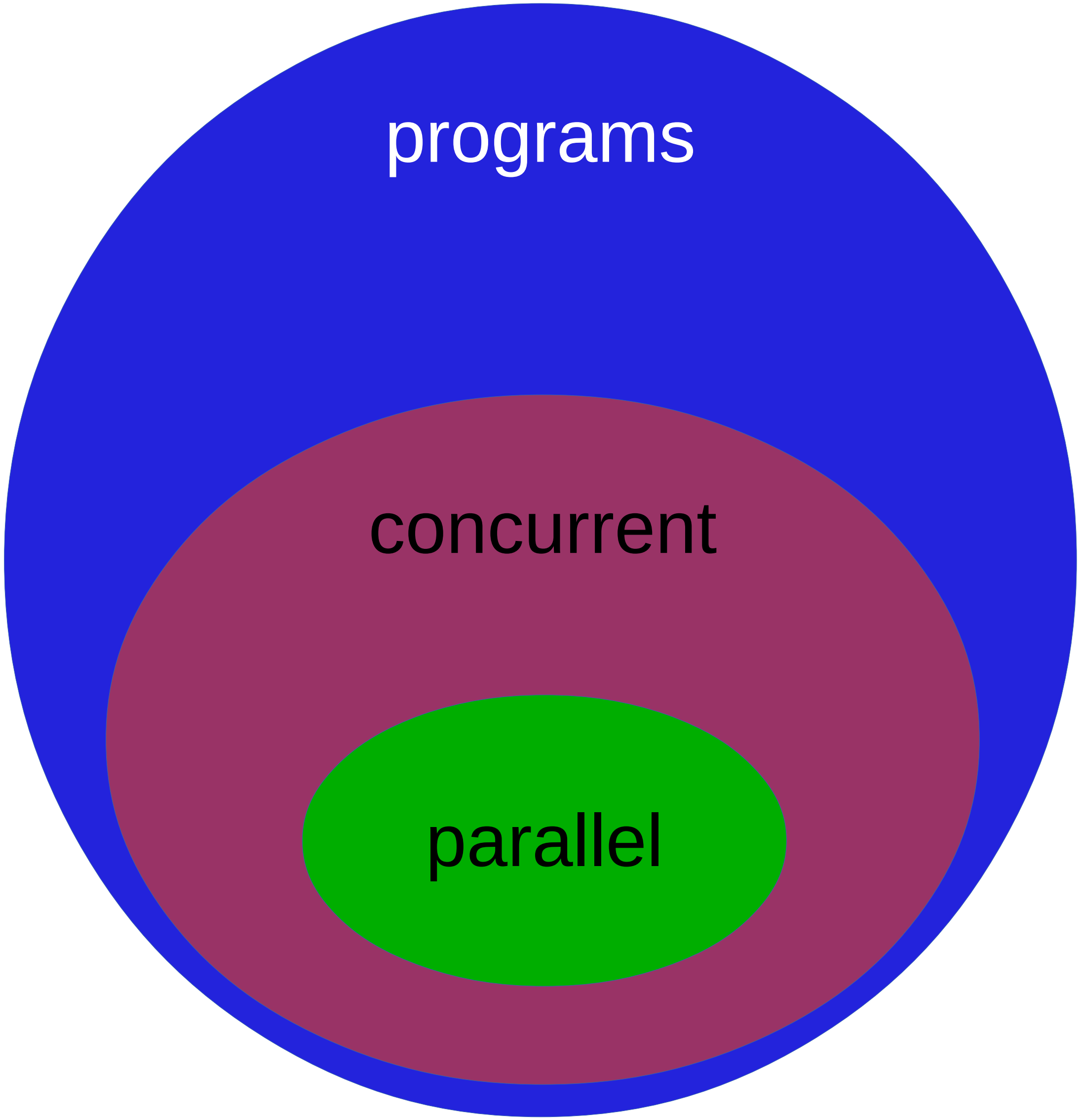
Some more details

Thread

- ❑ Loosely said, a thread consists of a series of instructions with it's own program counter ("PC") and state
- ❑ **A parallel program will execute threads in parallel**
- ❑ These threads are then scheduled onto processors

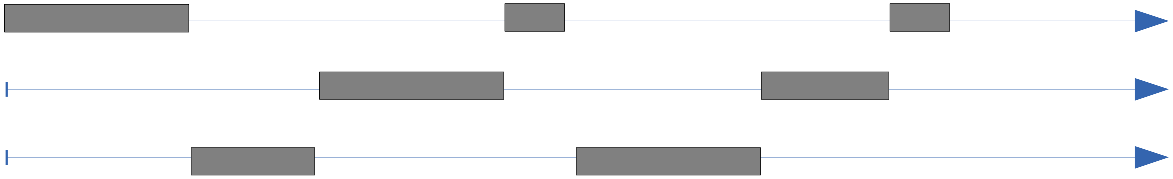


Parallelism vs Concurrency



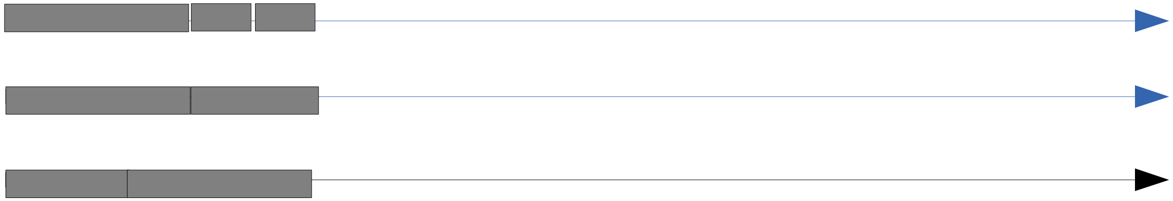
Parallelism vs Concurrency

Concurrent, non-parallel execution:



e.g. multiple threads on a single core CPU

Concurrent, and parallel execution



Parallel Programming models

Parallel Programming Models

Two “classic” parallel programming models:

- □ Distributed memory
  - □ MPI (de-facto standard, widely used)

    http://mpi-forum.org or http://open-mpi.org/

Cluster, SMPs

- □ Shared memory
  - □ OpenMP (de-facto standard)

    http://openmp.org/

SMP only

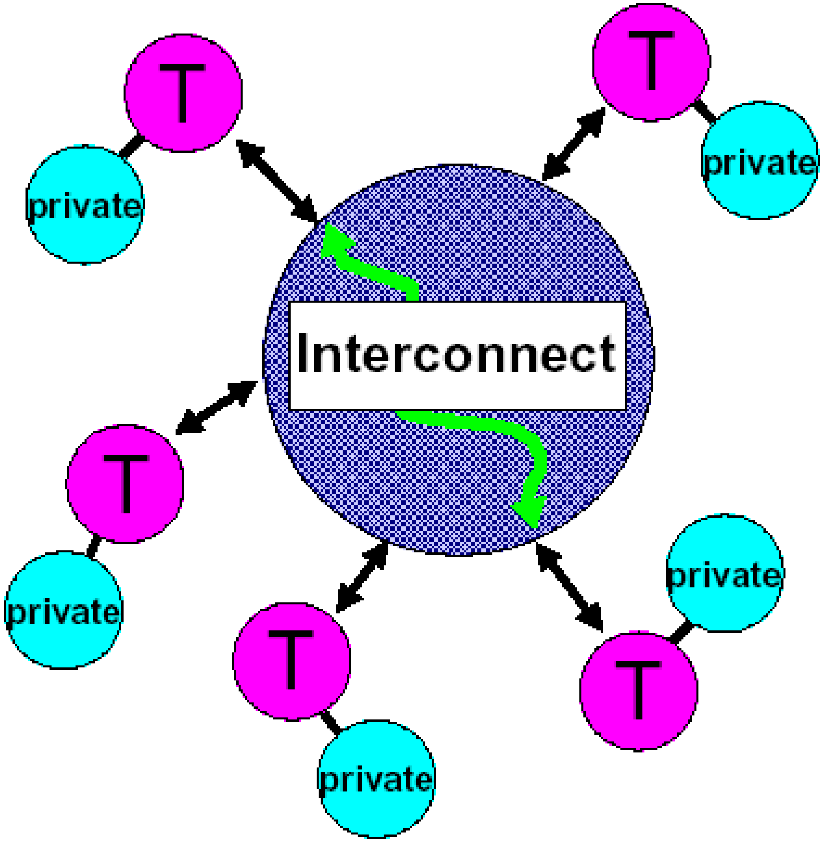
Distributed memory

Distributed memory programming model, e.g. **Message Passing Interface**:

❑ all data is private to the threads

❑ data is shared by exchanging buffers

❑ explicit synchronization



Distributed Memory

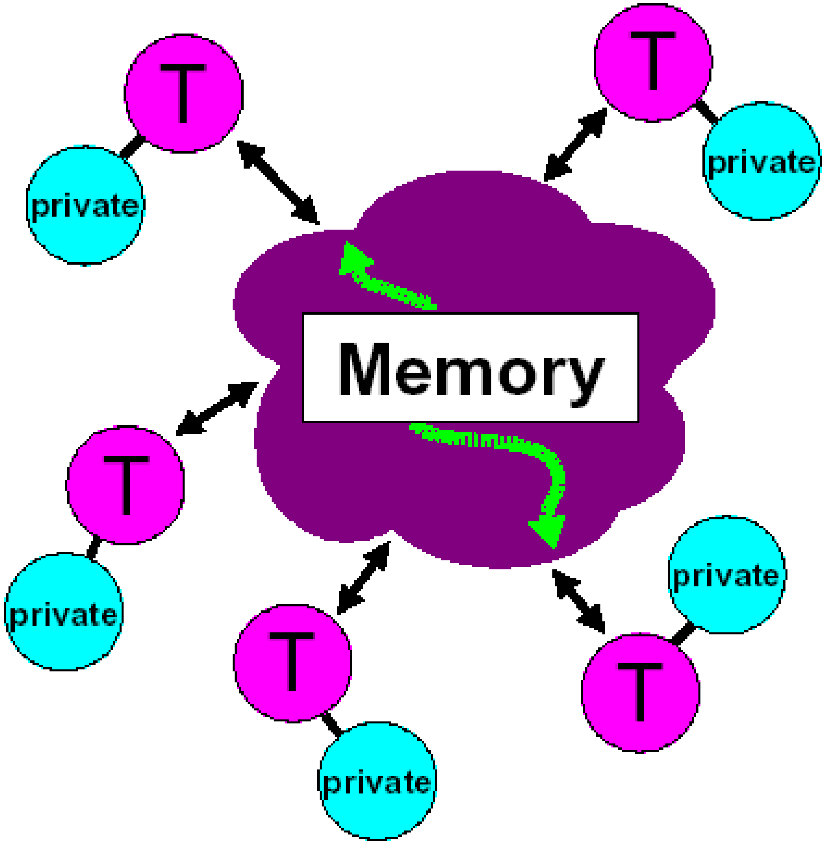
Message Passing Interface:

- ❑An MPI application is a set of independent processes (threads)
  - ❑on different machines
  - ❑on the same machine
- ❑communication over the interconnect
  - ❑network (network of workstations, cluster, grid)
  - ❑memory (SMP)
- ❑communication is under control of the programmer

Shared memory

Shared memory model, e.g. **OpenMP**:

- ❑ all threads have access to the same global memory
- ❑ data transfer is transparent to the programmer
- ❑ synchronization is (mostly) implicit
- ❑ there are private data as well



Shared Memory

OpenMP:

- ❑ needs an SMP (Symmetric MultiProcessing)
- ❑ but ... with newer CPU designs, there is an SMP in (almost) every computer
  - ❑ multi-core CPUs (CMP)
  - ❑ chip multi-threading (CMT)
  - ❑ or a combination of both, e.g the Sun UltraSPARC-T series
  - ❑ or ... (whatever we'll see in the future)

Parallel Overhead

Parallelization comes with some **costs**:

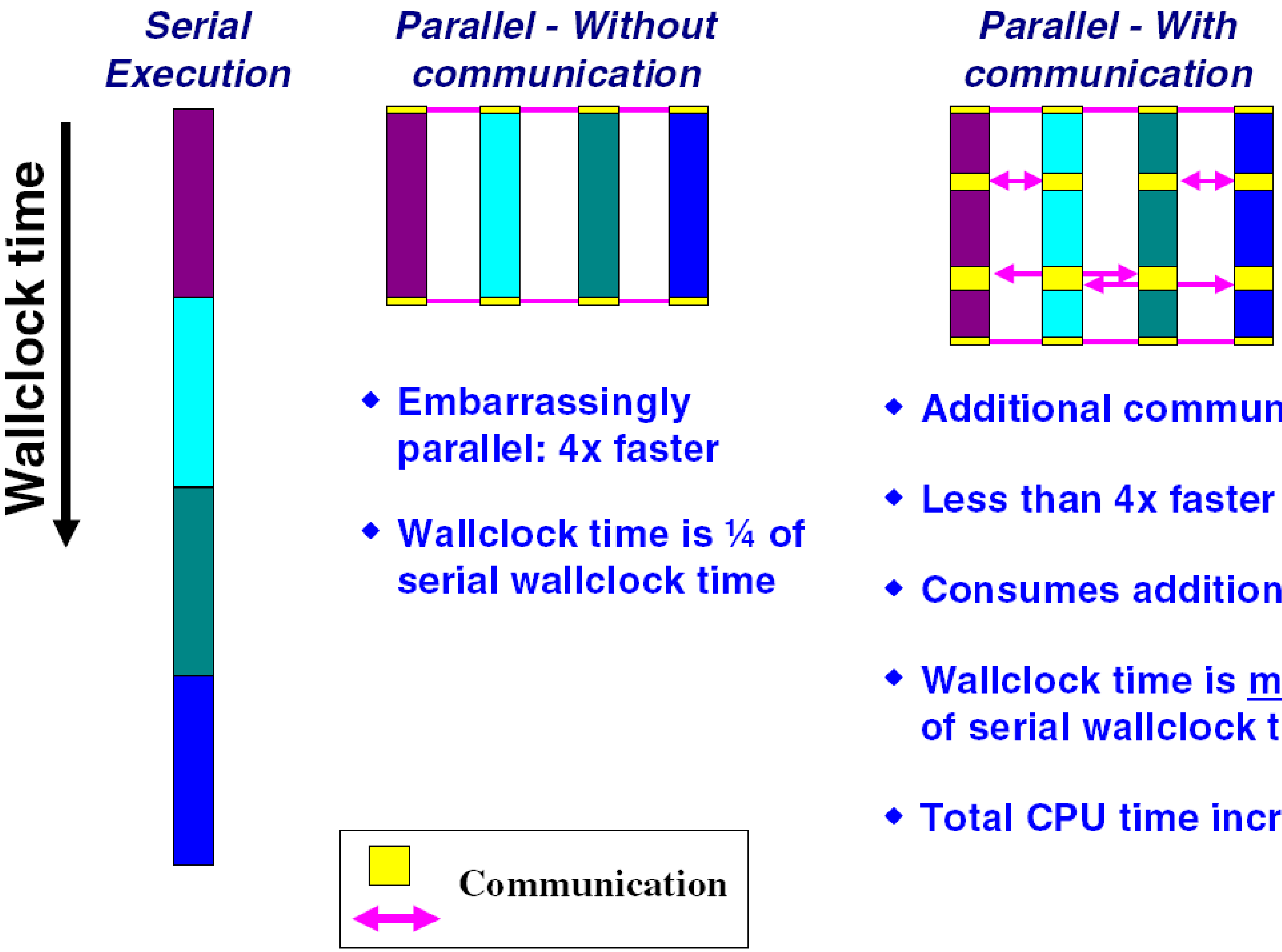
- ❑ Communication among the different parallel units (threads)
- ❑ Synchronization (to avoid conflicts)
- ❑ Management of the resources

Parallelization does not always pay:

- ❑ Not enough workload;
- ❑ Not enough granularity;

Efficient parallelization is about minimizing the communication overhead

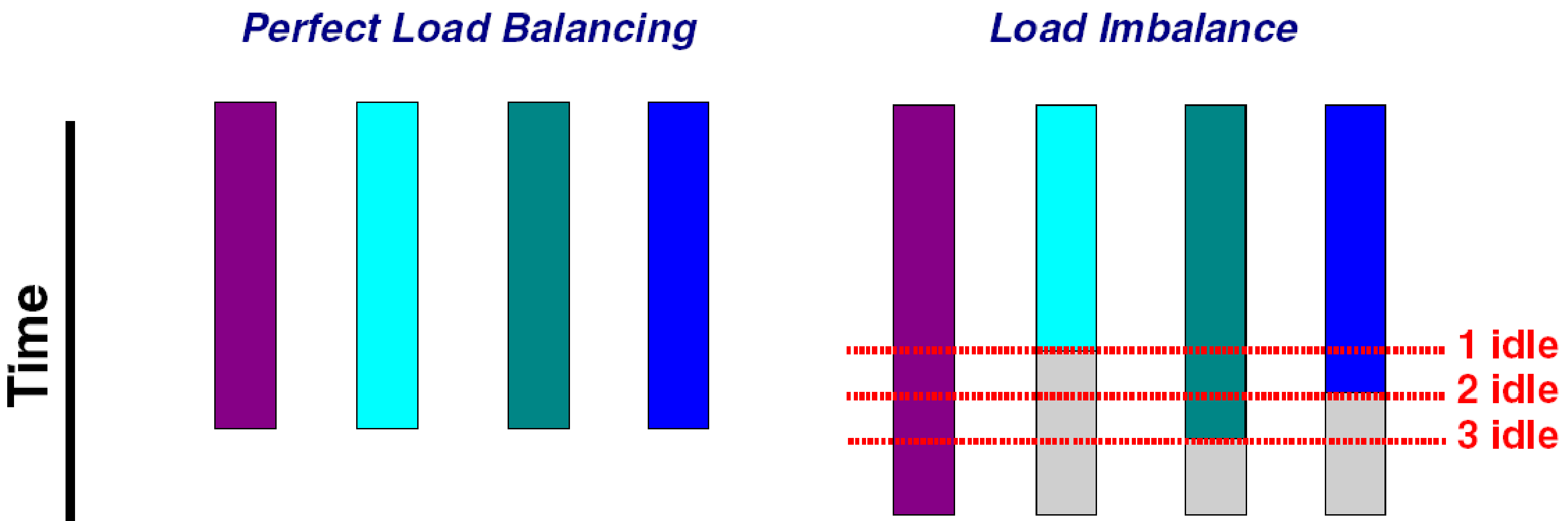
Communication


Serial Execution
Parallel - Without communication
Parallel - With communication
Wallclock time
Communication

- Embarrassingly parallel: 4x faster
- Wallclock time is ¼ of serial wallclock time

- Additional communication
- Less than 4x faster
- Consumes additional resources
- Wallclock time is more than ¼ of serial wallclock time
- Total CPU time increases

Load Balancing



- All threads finish in the same amount of time
- No threads is idle

- Different threads need a different amount of time to finish their task
- Total wall clock time increases
- Program will not scale well

□ Thread is idle

What to expect: metrics

Parallelization as an optimization technique:

Use the existing resource better to:

- ❑ Get results faster
- ❑ Deal with bigger problems

This leads to metrics and realistic expectations

Scaling: strong vs. weak

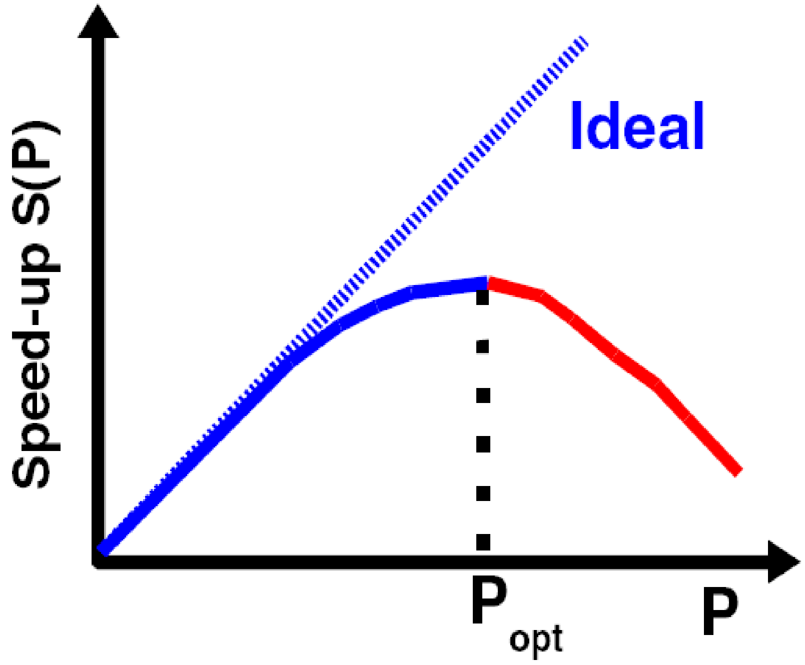
How does the execution time go down for a fixed problem size by increasing the number of Processing Units?

- ❑ Amdahl's law ⇒ speed-up, i.e. reduce time
- ❑ also known as “strong scaling”

How much can we increase the problem size by adding more Processing Units, keeping the execution time approx. constant?

- ❑ Gustafson's law ⇒ scale-up, i.e. increase work
- ❑ also known as “weak scaling”

Scalability – speed-up & efficiency



In some cases, S(P) will exceed P

This is called "superlinear" behaviour

Don't count on this to happen though

- *Define the speed-up S(P) as* *S(P) := T(1)/T(P)*
- *The efficiency E(P) is defined as* *E(P) := S(P)/P*
- *In the ideal case,* *S(P)=P* *and* *E(P)=P/P=1=100%*
- *Unless the application is embarrassingly parallel, S(P) will start to deviate from the ideal curve*
- *Past this point* P_{opt}*, the application will get less and less benefit from adding processors*
- *Note that both metrics give no information on the actual run-time*
- *As such, they can be dangerous to use*

Amdahl's Law

Assume our program has a parallel fraction "f"

This implies the execution time $T(1) := f*T(1) + (1-f)*T(1)$

On P processors: $T(P) = (f/P)*T(1) + (1-f)*T(1)$

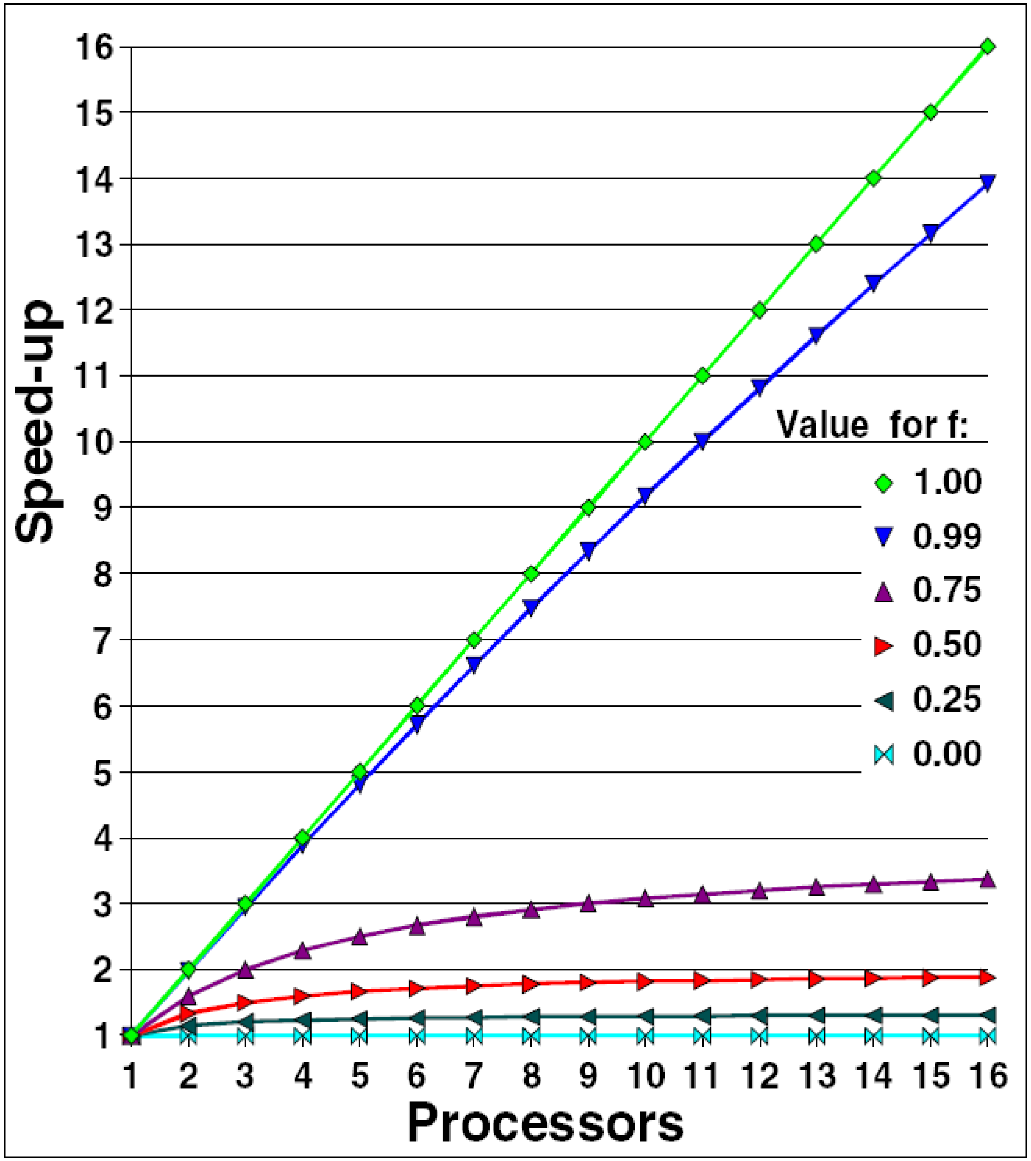
Amdahl's law:

$$\boxed{S(P) := T(1) / T(P) = 1 / (f/P + 1-f)}$$

Comments:

- ☞ *This "law' describes the effect that the non-parallelizable part of a program has on scalability*
- ☞ *Note that the additional overhead caused by parallelization and speed-up because of cache effects are not taken into account*

Amdahl's Law



- *It is easy to scale on a small number of processors*
- *Scalable performance however requires a high degree of parallelization i.e. f is very close to 1*
- *This implies that you need to parallelize that part of the code where the majority of the time is spent*
- *Use the performance analyzer to find these parts*

Gustafson's Law

- Effect of multiple processing units on the execution time – with a fixed amount of parallel work per PU:
  - $T(p) := t((1 - \pi) + \pi) \quad \Rightarrow T(1) = t((1 - \pi) + p\pi)$
- max speed-up:

$$S(p) = T(1)/T(p) <= \pi(p - 1) + 1$$

- **π** is the fraction where the parallel work per PU is fixed – different from the f in Amdahl's law above!
- **p** is the number of PUs (processors, cores, ...)

Amdahl's vs Gustafson's Law

- **Amdahl**'s law
  - Theoretical performance of an application with a ***fixed amount of parallel work*** given a particular number of PU (PUs)
- **Gustafson**’s Law:
  - Theoretical performance of an application with a ***fixed amount of parallel work per PU*** given a particular number of PUs

Code scalability in practice – I

- ❑Although Amdahl and Gustafson provide theoretical upper bounds, eventually real data are necessary for analysis
- ❑Inconsistencies in performance – especially on shared systems – often appear in singular runs
- ❑Best practice: Monitor codes several times and average the results to filter out periods of heavy usage due to other users

Code scalability in practice – II

- ❑Ideally, HPC codes would be able to scale to the theoretical limit, but ...
  - ❑Never the case in reality
  - ❑All codes eventually reach a real upper limit on speedup
  - ❑At some point codes become “bound” to one or more limiting hardware factors (memory, network, I/O)

Running Parallel Software

Parallel Software at DTU

- Available cores, memory, disk space.
- Many compilers and tools are ready to use
- Commonly used parallel softwares are already installed and set-up
- If something doesn't work, there is the technical support

HPC cluster is the right place to start playing with all this

Parallel Software

Your software:

Parallel programming, and getting performance out of parallel code, needs **compilers and toolboxes**.

Scientific software:

Most of the current scientific software comes now in a parallel version (MPI and/or openMP (and or Cuda/OpenCL)):

Not so easy to set-up

Parallel Software

Special care with parallel software:

- ❑ Scheduler is integrated with the parallel environment
- ❑ Take care of asking resources correctly
- ❑ Use the website for instructions and help
- ❑ Look a the batch file examples
- ❑ Remember: many software are organized in modules!

Batch Jobs

Nodes, cores, memory: when running a parallel job, you need to specify the number of cores you need.

Remember:

- ❑ A shared memory program (e.g. openMP) requires a SMP, so cores on one single node!
- ❑ A distributed memory program (e.g. MPI) *can* run on multiple nodes.
- ❑ Usually you have to pre-load the correct *environment*, before the real call to the program

Introduction to HPC

Batch Jobs

PBS options for nodes/cores:

```
#PBS -l procs=4
```

Ask for 4 cores, no matter where

```
#PBS -l nodes=1:ppn=4
```

Ask for 4 cores, on 1 node! Both openMP and MPI
Total # cores: ppn*nodes=4

```
#PBS -l nodes=2:ppn=4
```

Ask for 8 cores, 4 on each of 2 nodes! MPI only
Total # cores: ppn*nodes=8

Batch Jobs

Implemented PBS options for memory:

```
#PBS -l vmem=6gb
```

Maximum amount of job's virtual memory

```
#PBS -l pvmem=512mb
```

Maximum amount of virtual memory per process

Batch Jobs

Shared Memory example

```
#!/bin/sh
#PBS -N openMP_job
#PBS -q hpc
#PBS -l walltime=12:00:00
# -- number of processors/cores/nodes --
#PBS -l nodes=1:ppn=8
#PBS -M s012345@dtu.dk
#PBS -m abe

cd $PBS_O_WORKDIR
#Load needed modules
. . . . . .
#set and export Env VARIABLE for openMP
OMP_NUM_THREADS=$PBS_NUM_PPN
export OMP_NUM_THREADS
#eventually other openMP options
. . .
# Call user program
./myprogram.x [options]
```

Better use PBS environment variable

Batch Jobs

MPI example

```
#!/bin/sh
#PBS -N MPI_job
#PBS -q hpc
#PBS -l walltime=12:00:00
# -- number of processors/cores/nodes --
#PBS -l nodes=3:ppn=8
#PBS -M s012345@dtu.dk
#PBS -m abe

cd $PBS_O_WORKDIR
#Load mpi module
module load mpi
#load other modules
  . . . . . .
# Call user program
mpirun ./myprogram.x [options]
```

Installed OpenMPI version is tightly integrated with MOAB/Torque
no need to specify the number of processor in the mpirun call

Looking for help

HPC info:	http://www.cc.dtu.dk/
Gbar webpage:	www.gbar.dtu.dk
Ask for help:	support@hpc.dtu.dk

- REMEMBER:
  - Write the job-id, and all your job infos
  - Write back, especially when the problem is solved!