Supercomputers?

- Have overclocked high-speed processors? **NO**
- Their CPU runs faster than a desktop PC? **NO**
- Have large internal RAM? **NO (sometimes)**
- Run Excel spreadsheets? **NO**
- Run on MS Windows? **NO**
- Will run my software without changes? **NO (maybe, probably not optimized)**
- Can be used interactively? **NO (partly, to be improved)**
Parallel Programming

• The end of Moore’s law: no longer faster processors, but more of them (and this has limits, too)!
• But beware: 2 x 3 GHz < 6 GHz (cache consistency, multi-threading, etc.)
Supercomputer Scaling

Moore's Law for Supercomputers

- 1 TFlop/s: 6-8 years
- 1 Pflop/s: 8-10 years
- 1 Eflop/s: N=1
- 1 Zflop/s: N=500
- 1 Efin/s: 2021

SUM

Peak Performance at LRZ

10-fold every 3.5 years
Double every 13 Months

SuperMUC
IBM series X iDataplex
+ Linux-Cluster

HLRB2: SGI Altix 4700
+ Linux-Cluster

SuperMUC Phase1+2

SuperMUC-NG

GFlop/s, GByte

1 10 100 1,000 10,000 100,000 1,000,000 10,000,000
SuperMUC Utilization 2017

- Astrophysics/Cosmology: 23.3%
- Physics: High Energy Physics: 13.4%
- Physics - Solid State: 6.0%
- Chemistry: 6.4%
- Biophysics/Biology/Bioinformatics: 11.2%
- Physics - High Energy Physics: 13.4%
- Astrophysics/Cosmology: 23.3%
- Computational Fluid Dynamics: 27.4%
- Material Science: 0.7%
- Environmental Sciences: 0.7%
- Geophysics: 1.0%
- Meteorology: 1.7%
- Plasma Physics: 2.3%
- Crystallography: 2.7%
- Support/Benchmarking: 1.1%
- Engineering - others: 1.0%
- Informatics/Computer Sciences: 0.7%
Supercomputers!

- Have many off-the-shelf CPUs with vector instructions (AVX, AVX2, AVX512)
- The diskless compute nodes (no direct SSH access) are connected by a high-speed internal network (InfiniBand, Omni-Path)
- Different processes on the compute nodes can be synchronized using the Message Passing Interface (MPI)
- All nodes are connected to a parallel file system (GPFS, Lustre) which needs special libraries for full speed (MPI-I/O)
- Programs cannot be run interactively, but have to be submitted to the batch scheduler (SLURM)
- The OS is a version of Linux
Supercomputer aka. HPC Cluster

Switch

Fat Tree

Pruned Tree

Node

Island

Accelerator: GPU, FPGA

Socket

Core
Levels of Parallelization

- Node Level (e.g. SuperMUC Phase 2 has 3072 nodes)
- Accelerator Level (e.g. DGX-1 has 2 CPUs and 8 GPUs)
- Socket Level (e.g. teramem has 4 CPUs with 24 cores)
- Core Level (e.g. CoolMUC-3 has 64 cores with AVX512)
- Vector Level (e.g. AVX512 has 32 vector registers)

SuperMUC Phase 2 Peak Performance:

\[ 3.578 \text{ PFlop/s} = 3072 \text{ Nodes} \times 2 \text{ Sockets} \times 14 \text{ Cores} \times 16 \text{ Vectors} \times 2.6 \text{ GHz} \]
Parallelization Strategy: Shared Memory

- a few processes working closely together (10-100)
- single task list (script/program)
- shared memory
  (cache coherent non-uniform memory architecture aka ccNUMA)
- results are kept in shared memory
Parallelization Strategy: Message Passing

• many (potentially) independent processes (10 - 100,000)
• one task list for all (script/program)
• each process can (in principle) talk to every other process
• private memory
• needs communication strategy in order to scale
• very often: nearest neighbor communication
• beware of deadlocks!
Parallelization Strategy: Embarrassingly Parallel

- many independent processes (10 - 100,000)
- no communication between processes
- individual task list for each process
- private memory for each process
- results are stored separately on a (large) storage medium
Parallelization Strategy: Worker Queue

- many independent processes (10 - 100,000)
- master process (with access to a database/shared file system) coordinates workers
- private memory for each process
- results are sent back to master
- re-scheduling of failed tasks possible
CRAN Task View: High-Performance and Parallel Computing

CRAN Task View: High-Performance and Parallel Computing with R

Maintainer: Dirk Eddelbuettel
Contact: Dirk.Eddelbuettel@R-project.org
Version: 2018-08-27
URL: https://CRAN.R-project.org/view=HighPerformanceComputing

This CRAN task view contains a list of packages, grouped by topic, that are useful for high-performance computing (HPC) with R. In this context, we define 'high-performance computing' rather loosely as just anything related to packing R a little further: using compiled code, parallel computing (in both explicit and implicit modes), working with large objects as well as profiling.

Unless otherwise mentioned, all packages presented with highlights are available from CRAN, the Comprehensive R Archive Network.

Server of the area discussed in this Task View is undergoing recent change. Please send suggestions for additions and extensions for this task view to the task view maintainers.

Suggestions and corrections by Adnan Zahir, Markus Schindler, Martin Morgan, Max Kuhn, Tamas Rudasitsch, Johan Kjeus, Tobias Verbeke, Hao Yu, David Rosenberg, Marco Bello, Joo Myeol, Jay Emerson, Wei-Chen Chen, Bill Cleveland, Ross Beyer, Ramon Diaz-Uriarte, Mark Zeilinger, Kevin Utwel, Christian Pfaff, Will Landau, Tim Hines, Ravi Mohammed, Ralf Stuehr, and Bob Jensen (as well as others I may have forgotten to mention) are gratefully acknowledged.

Contributions are always welcome and encouraged. Since the start of this CRAN task view in October 2001, most contributions have arrived as email suggestions. The source file for this particular task view file now also resides in a GitHub repository (see below) so pull requests are also possible.

The CRAN package supports these Task Views. Its function install.views and update.views allow, respectively, installations or updates of packages from a given Task View; the option available can execute operations to packages listed as core below.

Direct support in R started with release 2.14.0 which includes a new package parallel incorporating (slightly revised) copies of packages multicore and snow. Some types of clusters are not handled directly by the base package 'parallel'. However, and as explained in the package

Parallel computing: Explicit parallelism

- Several packages provide the communications layer required for parallel computing. The free package in this area was open by Li and Rosenzweig which uses the PVM (Parallel Virtual Machine) standard and is open; no longer actively maintained, but available from the CRAN archive directory.
- In recent years, the alternative MPI (Message Passing Interface) standard has become the de facto standard in parallel computing. It is supported in R via the Rmpi by Yu, Rmpi package is mature yet actively maintained and offers access to numerous functions from the MPI API, as well as number of R-specific extensions. Rmpi can be used with the LAM/MPI, MPICH / MPICH2, Open MPI, and Fawkes MPI implementations. It should be noted that LAM/MPI is now in maintenance mode, and no development is focused on Open MPI.
- The package provides 54 classes to directly interface MPI in order to support the Single Program/Multiple Data (SPMD) parallel programming style which is particularly useful for both parallel execution. The phnlAPL builds on this and uses scalable linear algebra packages (namely BLAS, PLAPACK, and ScaLAPACK) in parallel versions based on ScaLAPACK version 3.5. The phnlAPL builds on these and provides the core classes and tools for distributed data types such as the phnlDDE, which provides distributed data matrices for 'Programming with Big Data'. The phnlDDE package provides multiple processing modes to write to the same file (without manual synchronization) and supports locally-sized files. The phnlDDE package provides examples for these packages, and a detailed vignette. The phnlPROF provides profiling functions for SPMD code via MPI profiling libraries, such as smppl, mppl, or DAIU.
- An alternative is provided by the axe (NoWorkflowSpace) packages from Revolution Computing. It is the successor to the earlier LooSpace approach to parallel computing, and is implemented on top of the Twisted networking toolkit for Python.
- The open (Single Program/Workstation) packages by Tye et al. can use PVM, MPI, NFS as well as direct networking sockets. It provides an abstraction layer for linking the communications details. The openCT package provides fault-tolerance extensions to open.
- The snail package allows general iterations over elements in a collection without the use of an explicit loop construct. Using loops without side effects also facilitates executing the loops in parallel which is possible via the dask (using parallel/cluster on simple vectorized, daskDANCE (using dask), or parallel/cluster on simple vectorized). daskDANCE (using dask), or parallel/cluster on simple vectorized).
- The future package allows for asynchronous (sequential) and asynchronous (parallel) evaluations via delegation of futures, either via function calls or using delayed execution. Global variables are automatically identified. Iteration over elements in a collection is supported.
- The Rcpp parallel provides OpenMP parallelism in a cooperative-progressive model. It is part of the Rcpp package which provides efficient use of multicore hardware in managing data and is describing splitting criteria, both of which are performance bottlenecks in the algorithm.
- The irlba package connects to the irlba open source machine learning environment which has scalable implementations of random forests, GBM, and glm (with elastic net regularization), and deep learning.
- The rlearn implementation can be used with HPC as well as MPI for random forest experiments suitable for survival analysis, computing risks analysis, classification as well as regression.
- The nmpc package can perform simulations using one or multiple cores, both locally and on HPC clusters.
- The openmp package can submit commands to run on remote clusters.

https://cran.r-project.org/web/views/HighPerformanceComputing.html

CRAN Task View: High-Performance and Parallel Computing

Parallel computing: Explicit parallelism

- Several packages provide the communications layer required for parallel computing. The free package in this area was open by Li and Rosenzweig which uses the PVM (Parallel Virtual Machine) standard and is open; no longer actively maintained, but available from the CRAN archive directory.
- In recent years, the alternative MPI (Message Passing Interface) standard has become the de facto standard in parallel computing. It is supported in R via the Rmpi by Yu, Rmpi package is mature yet actively maintained and offers access to numerous functions from the MPI API, as well as number of R-specific extensions. Rmpi can be used with the LAM/MPI, MPICH / MPICH2, Open MPI, and Fawkes MPI implementations. It should be noted that LAM/MPI is now in maintenance mode, and no development is focused on Open MPI.
- The package provides 54 classes to directly interface MPI in order to support the Single Program/Multiple Data (SPMD) parallel programming style which is particularly useful for both parallel execution. The phnlAPL builds on this and uses scalable linear algebra packages (namely BLAS, PLAPACK, and ScaLAPACK) in parallel versions based on ScaLAPACK version 3.5. The phnlAPL builds on these and provides the core classes and tools for distributed data types such as the phnlDDE, which provides distributed data matrices for 'Programming with Big Data'. The phnlDDE package provides multiple processing modes to write to the same file (without manual synchronization) and supports locally-sized files. The phnlDDE package provides examples for these packages, and a detailed vignette. The phnlPROF provides profiling functions for SPMD code via MPI profiling libraries, such as smppl, mppl, or DAIU.
- An alternative is provided by the axe (NoWorkflowSpace) packages from Revolution Computing. It is the successor to the earlier LooSpace approach to parallel computing, and is implemented on top of the Twisted networking toolkit for Python.
- The open (Single Program/Workstation) packages by Tye et al. can use PVM, MPI, NFS as well as direct networking sockets. It provides an abstraction layer for linking the communications details. The openCT package provides fault-tolerance extensions to open.
- The snail package allows general iterations over elements in a collection without the use of an explicit loop construct. Using loops without side effects also facilitates executing the loops in parallel which is possible via the dask (using parallel/cluster on simple vectorized, daskDANCE (using dask), or parallel/cluster on simple vectorized). daskDANCE (using dask), or parallel/cluster on simple vectorized).
- The future package allows for asynchronous (sequential) and asynchronous (parallel) evaluations via delegation of futures, either via function calls or using delayed execution. Global variables are automatically identified. Iteration over elements in a collection is supported.
- The Rcpp parallel provides OpenMP parallelism in a cooperative-progressive model. It is part of the Rcpp package which provides efficient use of multicore hardware in managing data and is describing splitting criteria, both of which are performance bottlenecks in the algorithm.
- The irlba package connects to the irlba open source machine learning environment which has scalable implementations of random forests, GBM, and glm (with elastic net regularization), and deep learning.
- The rlearn implementation can be used with HPC as well as MPI for random forest experiments suitable for survival analysis, computing risks analysis, classification as well as regression.
- The nmpc package can perform simulations using one or multiple cores, both locally and on HPC clusters.
- The openmp package can submit commands to run on remote clusters.
LRZ Machine Learning Infrastructure

- User PC
- Firewall
- $WORK
- $SCRATCH
- $HOME
- Linux Cluster
- Data Science Storage (DSS)
- GPU Cloud
- DGX-1
- Tape-Archive
- OpenNebula Cloud
- lxlogin[5-8;10].lrz.de
- https://www.rstudio.lrz.de
- https://datalab.srv.lrz.de
- https://www.cloud.mwn.de
LRZ Machine Learning Infrastructure

- Firewall
- User PC
- lxlogin[5-7].lrz.de
- lxlogin8.lrz.de
- lxlogin10.lrz.de
- Data Science Storage (DSS)
- CoolMUC-2 (mpp2)
- CoolMUC-3 (mpp3)
- mxlogin
- mpp2_inter
- mpp3_inter
- teramem
- $HOME
- $WORK
- $SCRATCH

Disk Resources

- $HOME (home directory)
  - 100GB per project (!)
  - Automatic backup and snapshots
    (there is a truly hidden “$HOME/.snapshot” subdirectory)
  - All your important files/anything you invested a lot of work into should be here
- $WORK/$PROJECT (project file system)
  - Initially 1TB per project, can be extended to 5TB (or more) if requested
  - No (automatic) backup by LRZ (!)
  - Use this for e.g. large raw data with a backup somewhere else
    (either at your institution or the LRZ tape archive)
Disk Resources

- **$SCRATCH** (scratch file system, “temporary file system”)
  - Several thousand TBs, i.e. PBs
  - No backup (!) and sliding window file deletion, i.e. old files will eventually be deleted (!!!) – a data retention time of approx. 30 days may be assumed, but is not guaranteed
  - This is the place for e.g. very large, temporary files or intermediate results, directly feeding into additional analyses
  - Do not save any important data exclusively on this file system! Seriously, don’t do it!

- Tape archive: visualize industrial robots juggling tape drives. If you need this, take a look at the documentation (and talk to us).

- Data Science Storage (DSS): novel approach at LRZ to meet demands and requirements of data intensive science, not generally available yet (if you have access, you probably know, e.g. some TUM projects)
R Modules

- R is not accessible on the Linux Cluster by default (try: `$ which R`)
- Environment modules allow for the dynamic modification of environment variables
- A (minimal) set of default modules is active after login:
  `$ module list`
- Use the module system to search for different R versions:
  `$ module available r` (or `module av r`)
R Modules

• (The default version of) R can be loaded using
  $ module load r

• If you need a different version, you have to specify the full name of the module

```bash
di36pez@mpp2-login5:~$ module load R
di36pez@mpp2-login5:~$ which R
/lrz/sys/applications/R/3.3.1/bin/R
```
R Package Management

• All R packages are installed into libraries – these are just directories in the file system with subdirectories for each installed package
• The default installation of R comes with a single library R_HOME/library usually containing the standard and recommended packages (in RStudio, this is called the System Library)
• Regular users may not add/install packages directly into this library (but administrators can)
R Package Management

- Individual users can have (one or more) additional, personal libraries (called User Library in RStudio)
- The path for this library directory can be specified by the environment variable $R_LIBS_USER
- If this is not set, R will ask you to create a personal package library when installing packages for the first time
- Per default, the suggested path is specific to the (minor) version R in use!
- You can use the .libPaths() function within R to check the current library directories
Slurm Workload Manager

• Job scheduler
  • Allocates access to resources (time, memory, nodes/cores)
  • Provides framework for starting, executing, and monitoring work
  • Manages queue of pending jobs (enforcing “fair share” policy)
• Use the sinfo command to get information about the available clusters
  $ sinfo --clusters=all or, shortened:
  $ sinfo -M all
• Get information about a specific cluster segment, e.g.
  $ sinfo -M mpp2 or
  $ sinfo -M inter
Job Processing

• For production jobs, you want to prepare and submit batch scripts
• They tell Slurm about the resources you need and the scripts/programs you want to run…
This is an example job script for a shared memory job…

… on a single node (with 28 CPUs)…

… of the CoolMUC-2 main cluster mpp2

Submit the job script to the queue:

```bash
$ sbatch myjob.sh
```
Job Management and Accounting

- Submit a job:
  
  \$ sbatch myjob.sh

- Query status of your jobs:
  
  \$ squeue -M mpp2 -u <user>

- Approximate start time of pending jobs:
  
  \$ squeue -M mpp2 -u <user> --start

- Abort a job:
  
  \$ scancel -M mpp2 <jobid>

- Get accounting data for (past) jobs:
  
  \$ sacct -X -M mpp2 [-S <YYYY-MM-DD>] -u <user>