Adventures in Supercomputing with R

Lecture 1: Introduction

George Ostrouchov

Oak Ridge National Laboratory and University of Tennessee

2021/12/31 (updated: 2022-02-16)

Lecture Today

- Some history for the big picture
- Parallel programming models and libraries
- Typical workflow and layout on a cluster
- Getting started: access, login, accounts, tools

Exercise Today

- Access, login, accounts, at IT4I, GitHub
- Basic unix, ssh, git

The Big Picture

via history of practical parallel computing ...

Early Practical Parallel Computing

A personal perspective (1985: iPSC/1)

1980's & 1990's - numerical mathematics

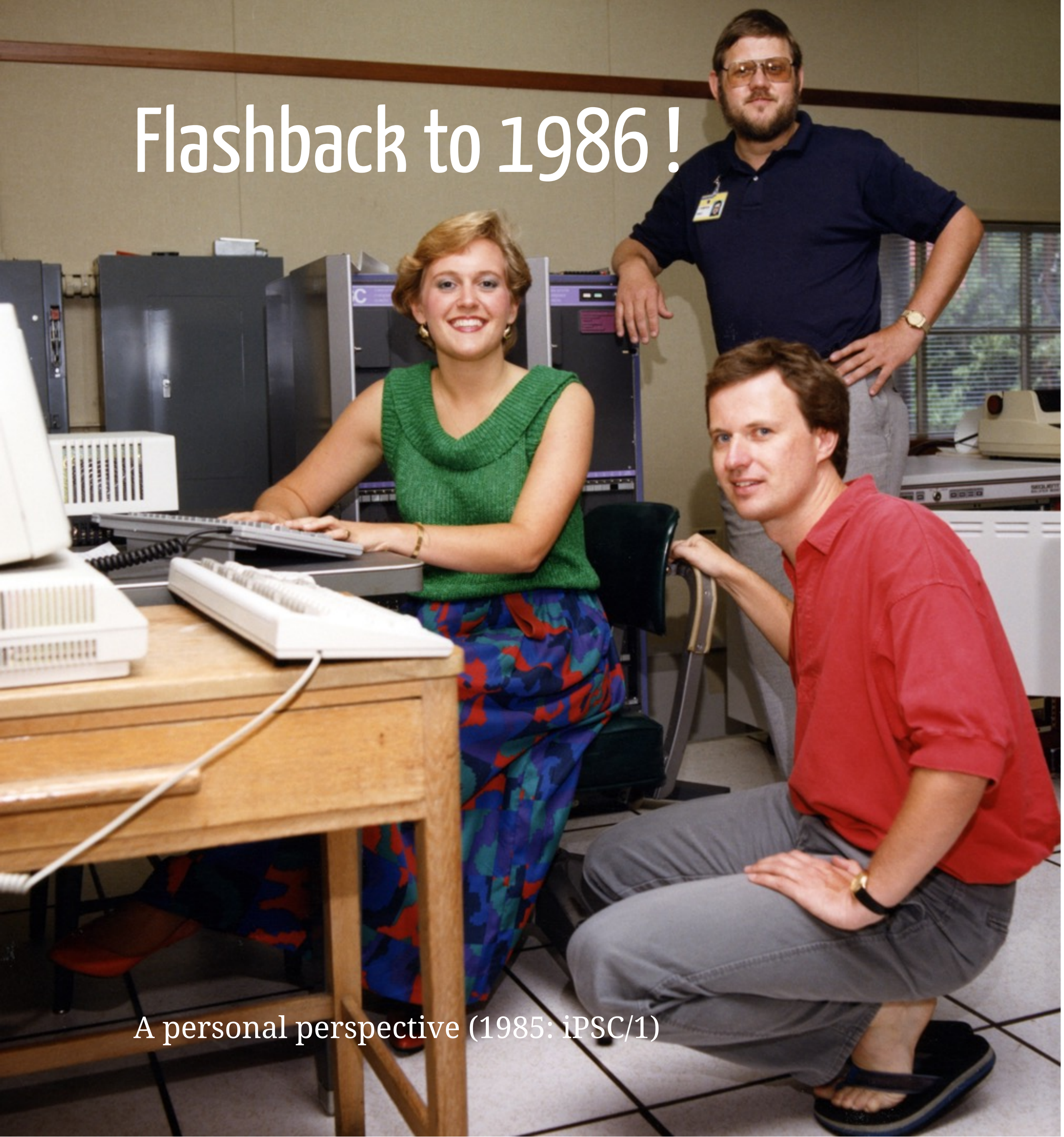
- Fast numerical computation, mostly matrix algebra
- Parallel numerical algebra (PVM, MPI, ScaLAPACK, PETSc, Trilinos, etc.)
- Simulation science - **supercomputing**

1990's & 2000's database computer science

- Data storage, retrieval, and search services
- Parallel data processing (Hadoop/MapReduce, Spark, etc.)
- Web search and business data - **massive data centers**

*iPSC/1 Intel Personal Super Computer: Computer History Museum

Flashback to 1986 !



A personal perspective (1985: iPSC/1)

Early Practical Parallel Computing

A personal perspective (1985: iPSC/1)

1980's & 1990's - numerical mathematics

- Fast numerical computation, mostly matrix algebra
- Parallel numerical algebra (PVM, MPI, ScaLAPACK, PETSc, Trilinos, etc.)
- Simulation science - **supercomputing centers**

1990's & 2000's database computer science

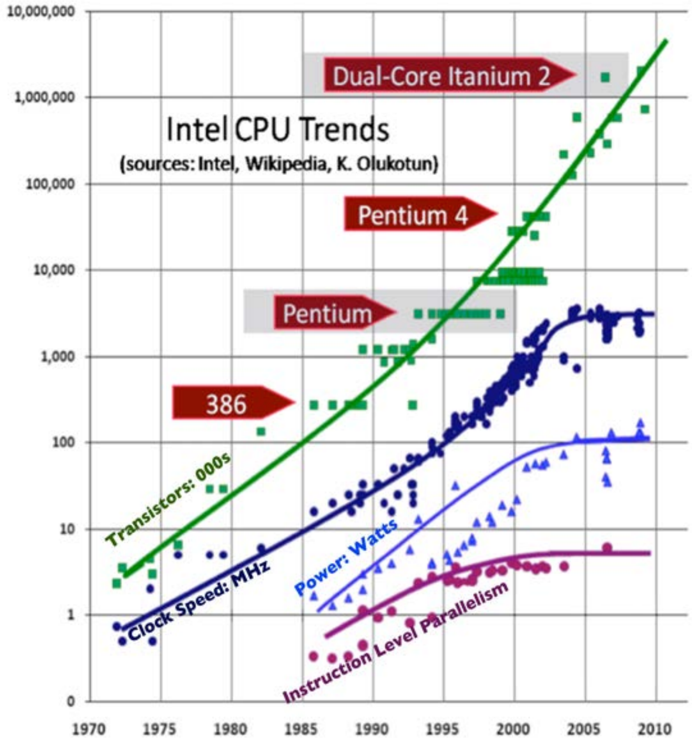
- Data storage, retrieval, and search services
- Parallel data processing (Hadoop/MapReduce, Spark, etc.)
- Web search and business data - **massive data centers**

*iPSC/1 Intel Personal Super Computer: Computer History Museum

Multicore Arrives around 2004

The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*

- Processor clock speeds stop increasing ~3 GHz
  - Power heat wall
- Parallel computing enters all software development
- Who understands statistical computing?
  - **Massive data centers?**
  - **Supercomputing centers?**



* Herb Sutter (2004)

1980's S Language for Ineractive Statistics

- Mainframe computers - time sharing or batch
- File-backed objects

1990's R (a dialect of S)

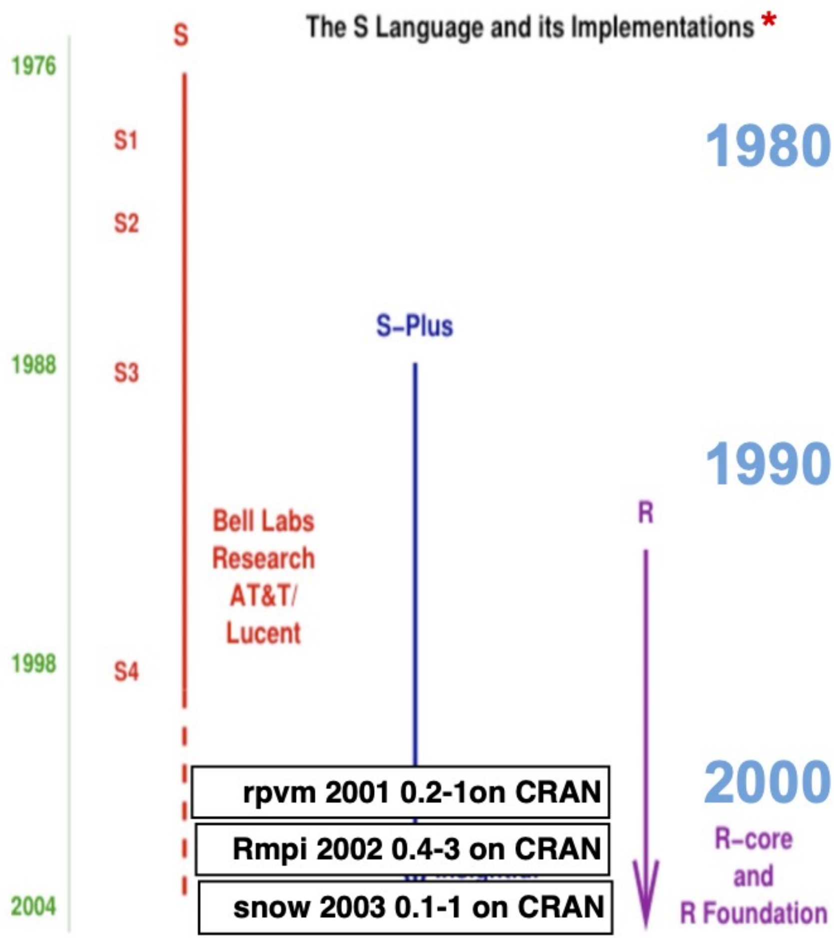
- Personal workstations, personal computers, laptops
- In-memory objects

2000's Parallel computing with R

- Small network of workstations (`snow` package)
- Multiple cores via `unix fork` (`multicore` package)
- Now both combined in `parallel` package

2010's Supercomputing with R

- Back to "mainframes" and batch!
- pbdR - programming with "big" data in R



1980

- Early distributed systems produced (iPSC, NCUBE)
- Numerical linear algebra embraces parallel computing, some interest in statistics

1990

- PVM developed
- MPI standard developed
- First distributed matrix libraries released (PBLAS, ScaLAPACK)

2000

- rpvm 2001 0.2-1on CRAN
- Rmpi 2002 0.4-3 on CRAN
- snow 2003 0.1-1 on CRAN
- Simulation science is the driver of supercomputing
- CPU clock speeds stagnate, multicore emerges
- Everyone, including statistics and R, cares about parallel computing mostly from a multicore perspective
- multicore 2009 0.1-0 on CRAN

2010

- parallel = multicore + snow 2011 0.1-1 on CRAN
- first pbdR project release 2012 on CRAN

- Matloff (2015). Parallel Computing for Data Science: With Examples in R, C++ and CUDA. Chapman & Hall/CRC

* History of S and R (with some thoughts for the future)
John M. Chambers June 15, 2006
https://www.r-project.org/conferences/useR-2006/Slides/Chambers.pdf

Summary

- Two parallel computing communities
  - Parallel numerical math (generating data) - supercomputing
  - Processing massive data (storing data) - data centers
- Statistical computing left batch environment to be interactive
  - Developed S and then R
  - High-level, extensible, and interactive
  - Use numerical libraries
- Supercomputing takes R back to batch
  - High-level, extensible, but batch
  - Use scalable numerical libraries

Questions?

Parallel Programming Models

Interactive vs batch, shared or distributed memory, libraries ...

Interactive vs Batch

- Data analysis:
  - Discovery process
  - Many iterations with a human in the loop
  - Each iteration brings new things to try
- High-level language programming
  - Manipulate more complex objects
  - Complex objects require probing (e.g. gam model)
  - Objects change during execution (e.g. matrix dimensions)
- Cluster computers are batch
  - Batch runs
  - Batch debugging
  - Minimize frustration with:
    - Edit code with familiar laptop tools
    - Send code to batch system for execution

Parallel Computing Models

Shared memory parallel computing

- Processors have access to all memory
  - Locking mechanism
- Kinds: unix fork, pthreads, OpenMP, OpenACC
- Libraries: OpenBLAS, MKL, FlexiBLAS, PLASMA, MAGMA, etc.

Distributed memory parallel computing

- Processors have only local memory
  - Communication mechanism
- Kinds: MPI, MapReduce, DataFlow
- Libraries: OpenMPI, ScaLAPACK, PETSci, Trillinos, etc.

Parallel Hardware and Programming Models

- Flynn's taxonomy* for hardware
    - *Single Instruction, Single Data (SISD):* scalar processor, serial program
    - *Single Instruction, Multiple Data (SIMD):* vector processor,
    - *Multiple Instruction, Multiple Data (MIMD):* multiple cores in a single processor, multiple processors in a single computer, and multiple computers in a cluster.
    - *Multiple Instruction, Single Data (MISD):* is not used much
- Software programming models
    - manager-workers: most common in simple cases
    - Single program, multiple data (SPMD): most common and most scalable on clusters
    - MapReduce: common in data processing
    - Dataflow: dependency graph directed, still evolving

*Flynn (1972)

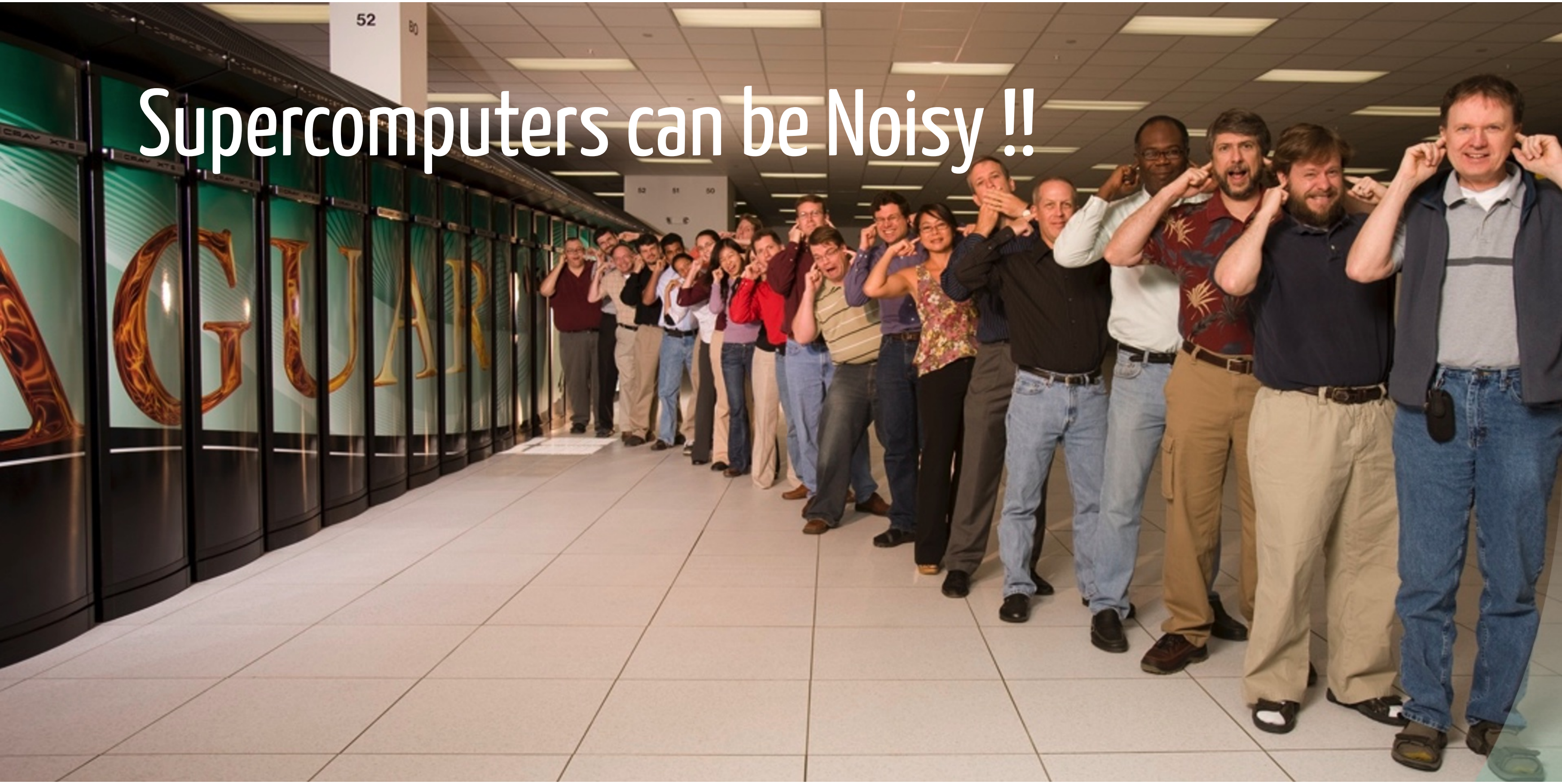
Faster Serial Code

- Almost any R code can be made faster
- Profile, profile, profile
- Fast libraries: OpenBLAS or MKL
- C/C++ access

Multicore Shared Memory Approaches

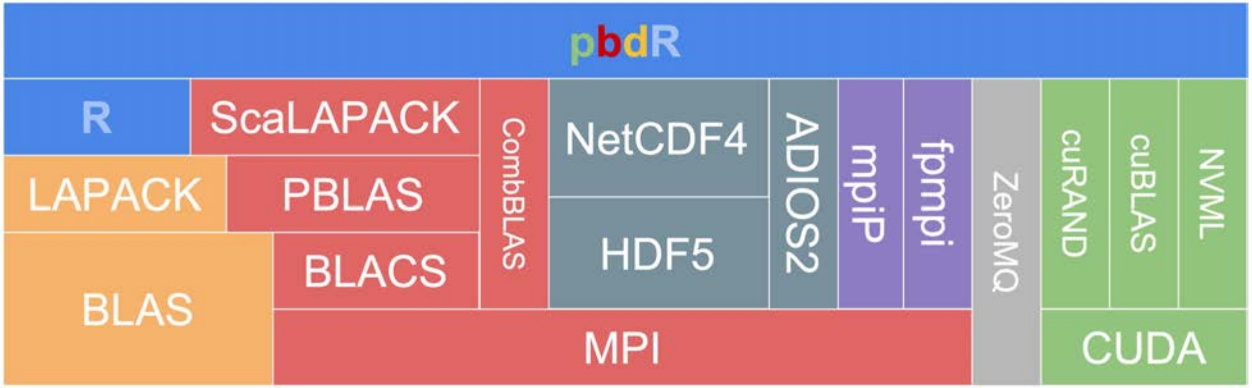
- Fast multithreaded libraries: OpenBLAS or MKL
- Unix fork via mclapply, et. al
- OpenMP via C/C++ access
- Cuda, OenCL on GPU

Flashback to 2011 !



ORNL Jaguar: 16,688 nodes with 224,256 cores, a Cray XT5 system

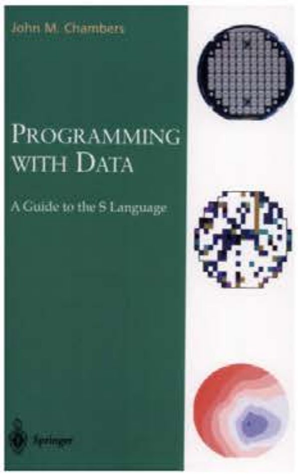
R Supercomputing via Scalable Libraries*



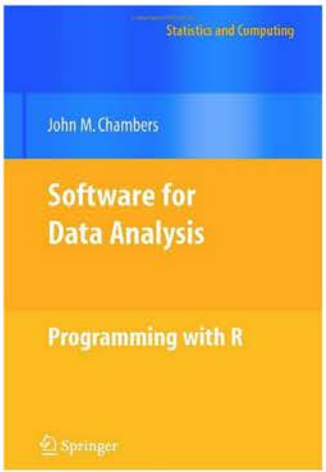
- Bridge high-performance computing with high-productivity of R language
- Keep syntax identical to R, when possible.
- Software reuse philosophy:
  - Don't reinvent the wheel when possible
  - Introduce HPC standards with R flavor
  - Use scalable HPC libraries with R convenience
- Simplify and use R intelligence where possible

*pbdR packages to access these libraries are at https://github.com/RBigData

Language for Programming with Data: R



1998 S language

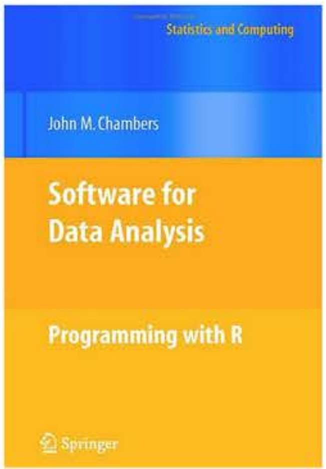


2008 R language

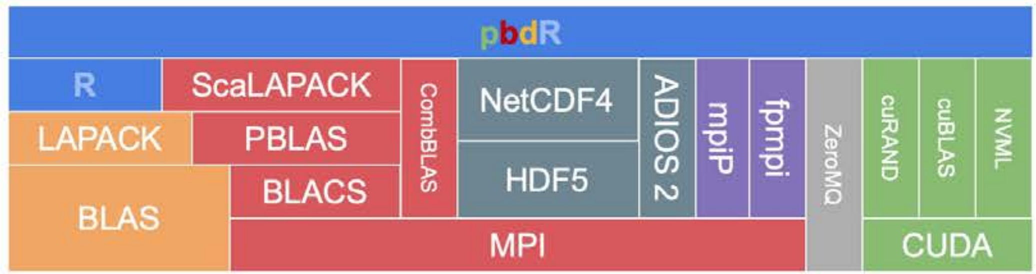
First S, then R:
Same language, expressive for data, different engine

Language for Programming with big Data: R + pbdR

1998 S language



2008 R language



Harnessing the engines of supercomputing with R + pbdR for statistical computing

First S, then R:
Same language, expressive for data, different engine

R-LAPACK-BLAS

- BLAS: Basic Linear Algebra Subroutines - A matrix multiplication library
  - vector-vector (Level-1), matrix-vector (Level-2), matrix-matrix (Level-3)
- LAPACK: dense and banded matrix decompositions and more
  - $LU \quad LL^T \quad QR \quad UDV^T \quad VD^2V^T \quad \|\cdot\|_p$
- Implementations: OpenBLAS, Intel MKL, Nvidia nvBLAS, Apple vecLib, AMD BLIS, Arm Performance Libraries

pbdR-ScaLAPACK-PBLAS-BLACS-MPI

- 2d Block-Cyclic data layout - mostly automated in pbdR
- MPI: Message Passing Interface - *de facto* standard for distributed communication in supercomputing
- BLACS: Basic Lineaar Algebra Communication Subroutines - special communication collectives for distributed matrix computation in PBLAS and ScaLAPACK
- PBLAS: Parallel BLAS - distributed BLAS (uses BLAS within blocks)
- ScaLAPACK: Scalable LAPACK - Distributed version of LAPACK (uses PBLAS/BLAS but not LAPACK)
- Implementations: HPE Cray LibSci, many vendors use reference version with custom compiler and custom BLAS and MPI implementations

pbdR - NetCDF4, HDF5, ADIOS2 - I/O Libraries

- NetCDF4: binary I/O (uses HDF5). Common in climate simulations.
- HDF5: binary I/O. Hierarchical directory structure.

- ADIOS2: binary I/O library. Common in largest time-stepping simulations. Has own .bp format, can work with HDF5 format

- pbdIO: R package (not pictured) CSV reader with parallel file system support (uses data.table's fread)

pbdR - fpmpi, mpiP - MPI

- fpmpi: Fast profiling of MPI communication patterns library
- mpiP: A light-weight MPI profiler

pbdR -ZeroMQ

- ZeroMQ: lightweight messaging library via sockets
  - pbdZMQ - used for interactive SPMD - ZMQ ships R code

pbdR - cuRAND, cuBLAS, NVML - CUDA

- CUDA: Compute Unified Device Architecture - NVIDIA's API for GPU parallel computing
- cuRAND: CUDA random number generation library
- cuBLAS: BLAS for NVIDIA GPUs
- NVML: NVIDIA Management Library - Monitoring and managing NVIDIA GPU devices.

Summary

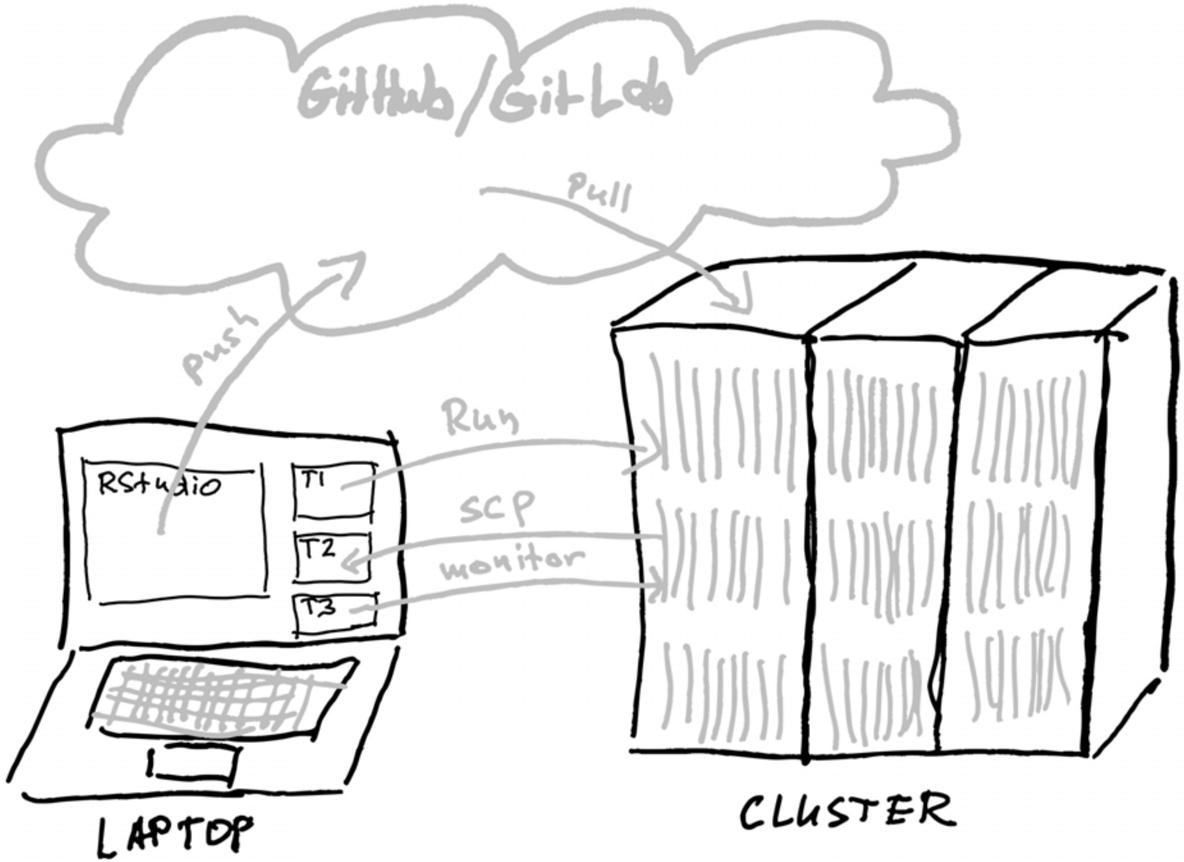
- Speed up serial code before considering parallel
- Consider computing needs in terms of standard libraries before implementing custom
- Serial libraries have scalable distributed equivalents
- Programming models: shared memory, distributed memory

Questions?

Our Narrow Path to Supercomputing

What is our workflow and how does it work ...

Typical Workflow from Laptop to Cluster



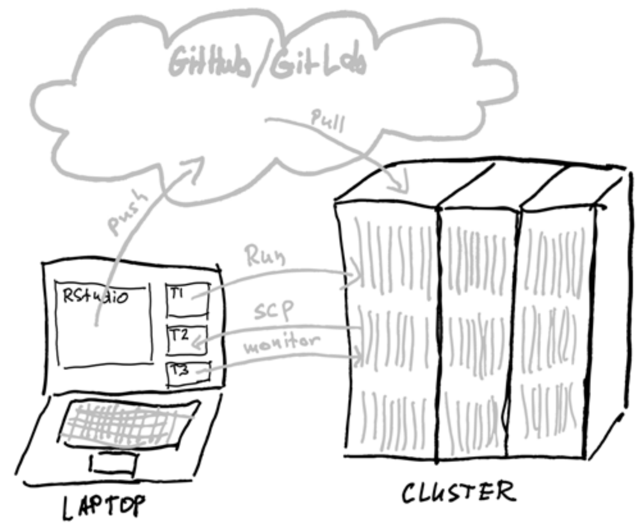
Why?

Laptop RStudio

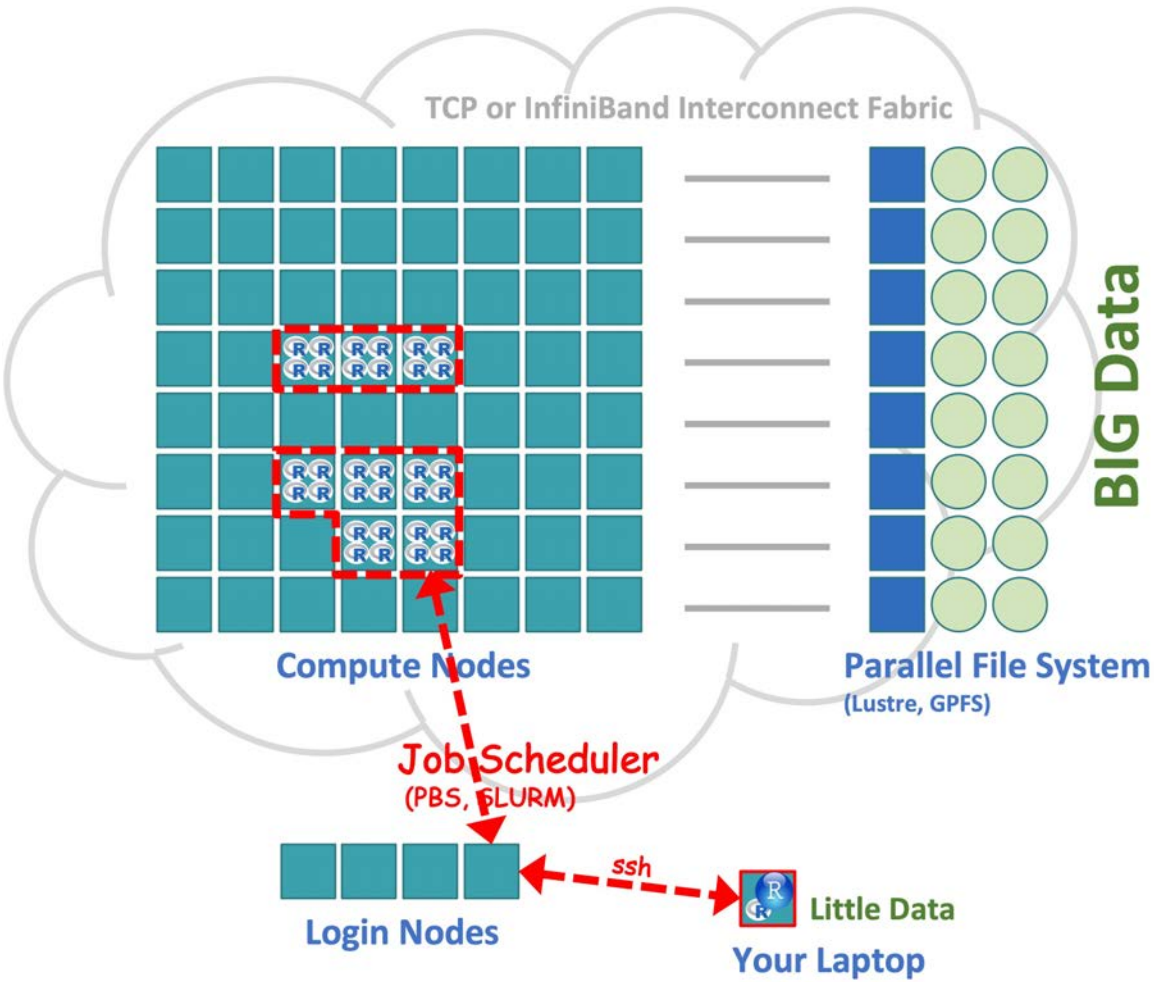
- Familiar custom editing environment
- Syntax checking

Cluster unix

- Same environment for all



Running Distributed on a Cluster



Software on Laptop/Desktop

RStudio: RStudio Desktop Free

git: RStudio with Git

ssh: Wikipedia ssh

- IT4I access, IT4I ssh
- IT4I Windows

Software on Cluster

Basic unix (Linux) commands

- ssh, ls, mkdir, rm, rmdir, less (and / search), mv, cp, scp, cat, echo, export, etc.

shell scripts

- A script of unix commands
- Example: orchestrate environment and job submission

PBS or SLURM

- Job scheduling and resource manager
- IT4I uses PBS, many other systems use SLURM

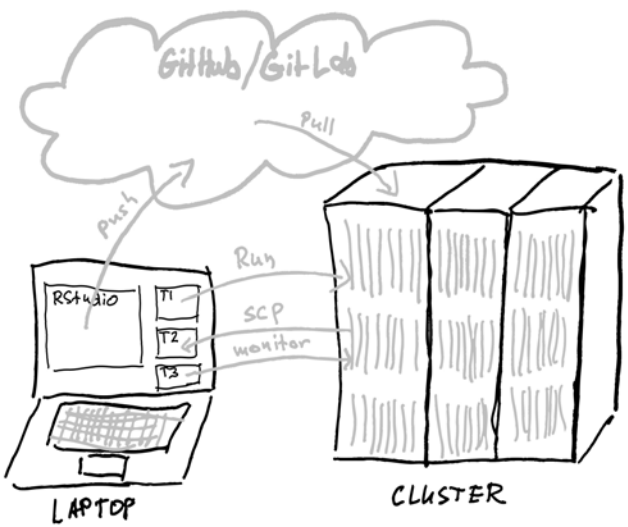
Software modules

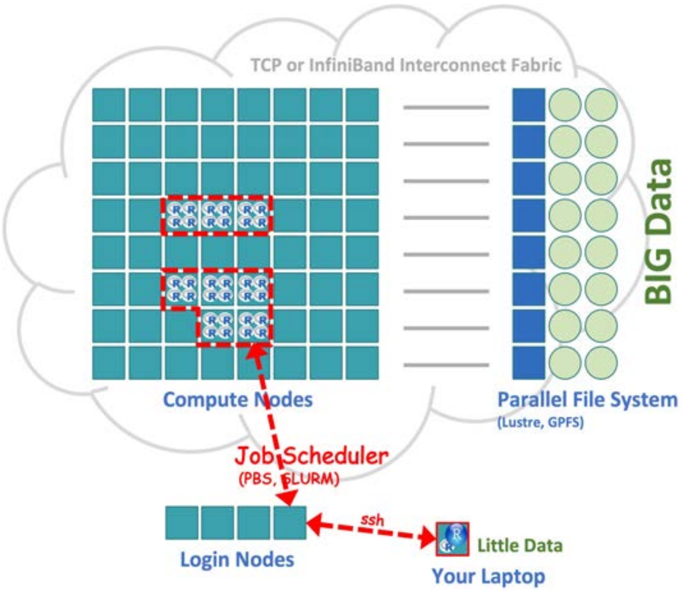
- Sets environment varialbes
- Loads software paths
- Example: `module load R`

Summary

- Edit code on familiar laptop
- Push code to cloud, pull code to cluster
- Run batch on cluster
- Clusters are unix systems

Questions?





Getting Started ...

- IT4I Account and connect to project allocation
- GitHub account (free personal)
- RStudio setup
    - Generate GitHub PAT, store in password manager
    - Use PAT - this stores it in RStudio
    - Happy Git with R

IT4I Systems

Barbora

- Installed 2019
- 0.849 PFlop/s Peak
- 192 Nodes, each with 36 cores
- 8 Nodes, each with 24 cores and 4 GPU accelerators
    - Reduced precision AI Peak at 4 PFlop/s

IT4I Systems

Karolina

- Installed 2021 (#69 on Top500[1], #8 on Green500[2])
- 15.7 PFlop/s Peak
- 720 Nodes, each with 128 cores (universal, 3.8 peak)
- 72 Nodes, each with 128 cores and 8 GPU accelerators (AI, 11.6 peak)
- 36 Nodes, each with 128 cores (cloud, 0.192 peak)

[1] https://www.top500.org/lists/top500
[2] https://www.top500.org/lists/green500

Class Project Allocation: 100,000 core hours

- Easy to spend allocation:
  - Karolina: 1 node (128 cores each) for 32.5 days
  - Barbora: 4 nodes (36 cores each) for 29 days

Use tight time limits on batch jobs!

- PBS job submission:
  - https://docs.it4i.cz/general/job-submission-and-execution/
  - Option: `-l select=x:ncpus=y,walltime=[[hh:]mm:]ss[.ms]`
  - 1 minute should be enough for most: `walltime=00:01:00`

IT4I Systems Documentation

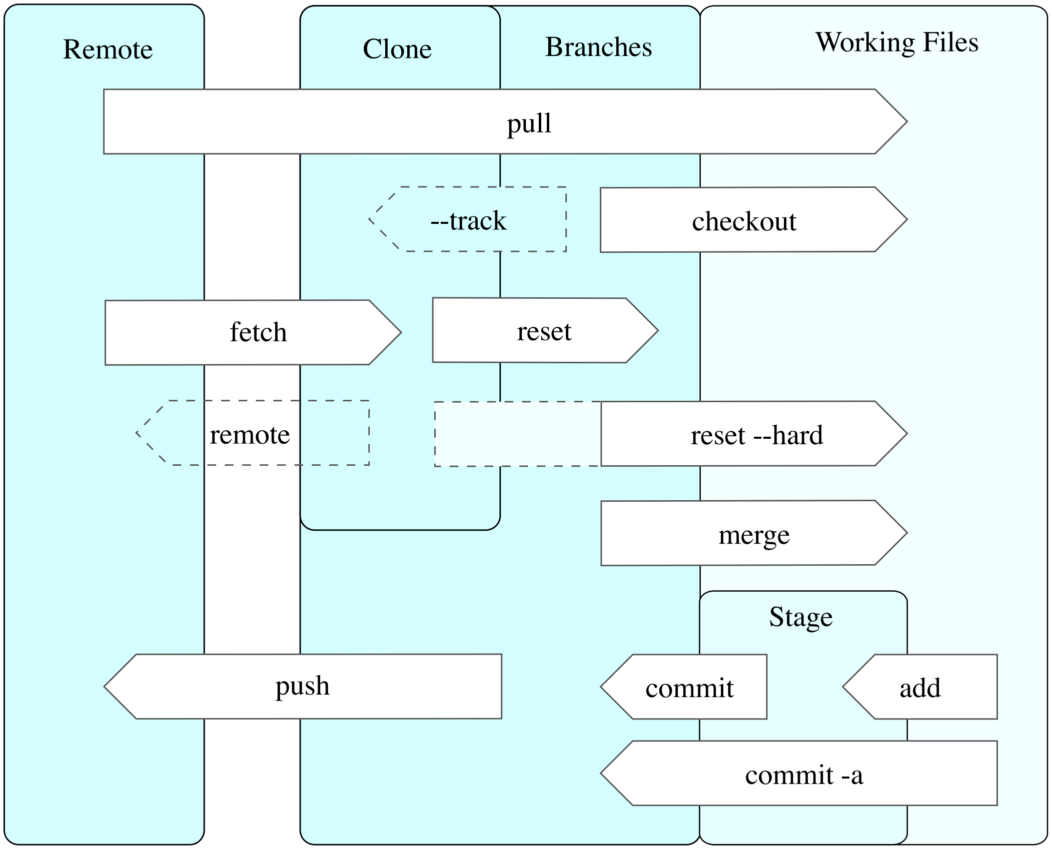
https://docs.it4i.cz/

- "Basic proficiency in Linux environments is required."
    - Introduction to Linux
        - History of Unix and Linux
- "Learn how to parallelize your code."

IT4I Login and Cluster Access

- Read link to obtain login
  - *e-INFRA CZ Account* for those affiliated with a Czech academic institutiion
  - *IT4I Account* for others
- Then, folow instructions to get connected to class project DD-21-42

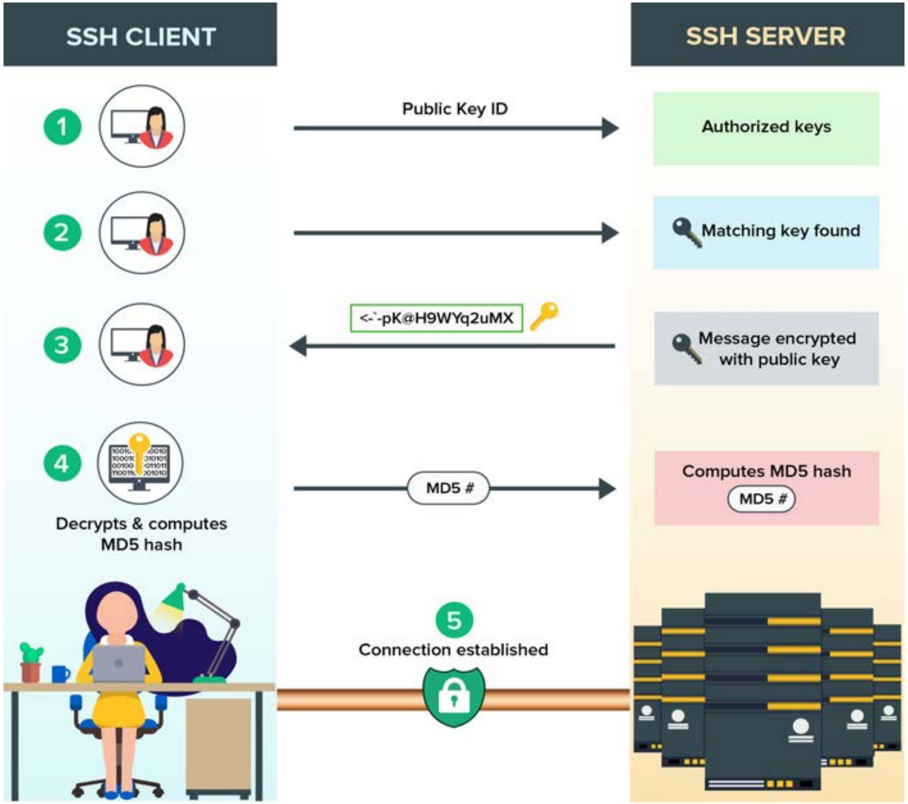
GitHub and git



*By Daniel Kinzler - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=25223536

ssh keys

A message encrypted by public key can be decrypted by private key



Graphic from https://www.manageengine.com/key-manager/information-center/what-is-ssh-key-management.html

Exercise & Discussion Follows

Slides created via the R package **xaringan**
and converted to pdf with Google Chrome