OAK RIDGE National Laboratory

Statistical Computing on Large Parallel Architectures

George Ostrouchov
Oak Ridge National Laboratory
and University of Tennessee

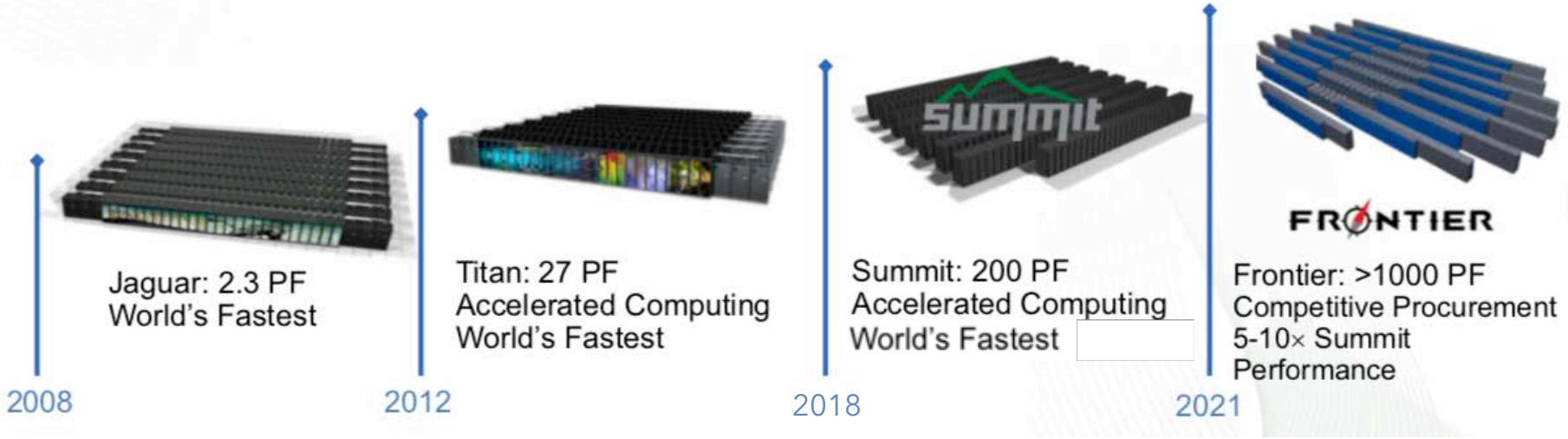
62nd World Statistics Congress
August 18 – 23, 2019, Kuala Lumpur, Malaysia

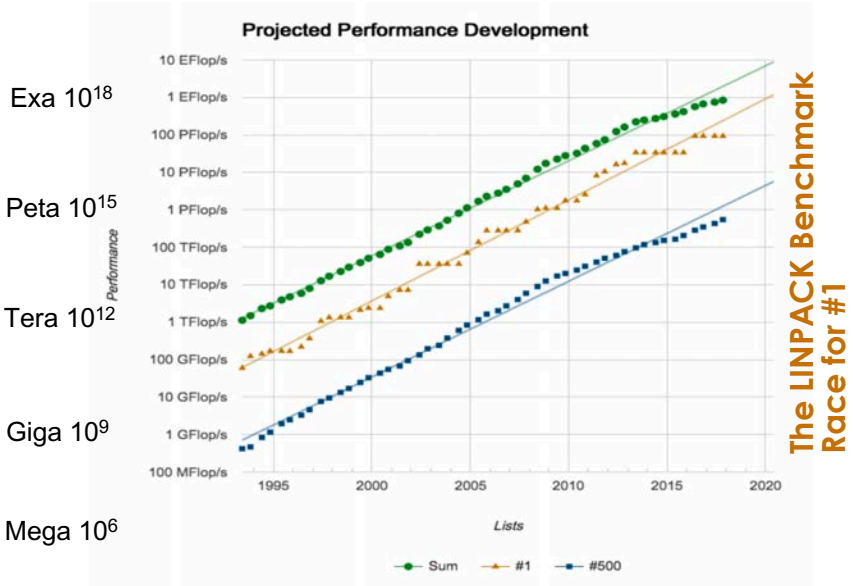
ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

This work used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Oak Ridge Leadership Computing Facility

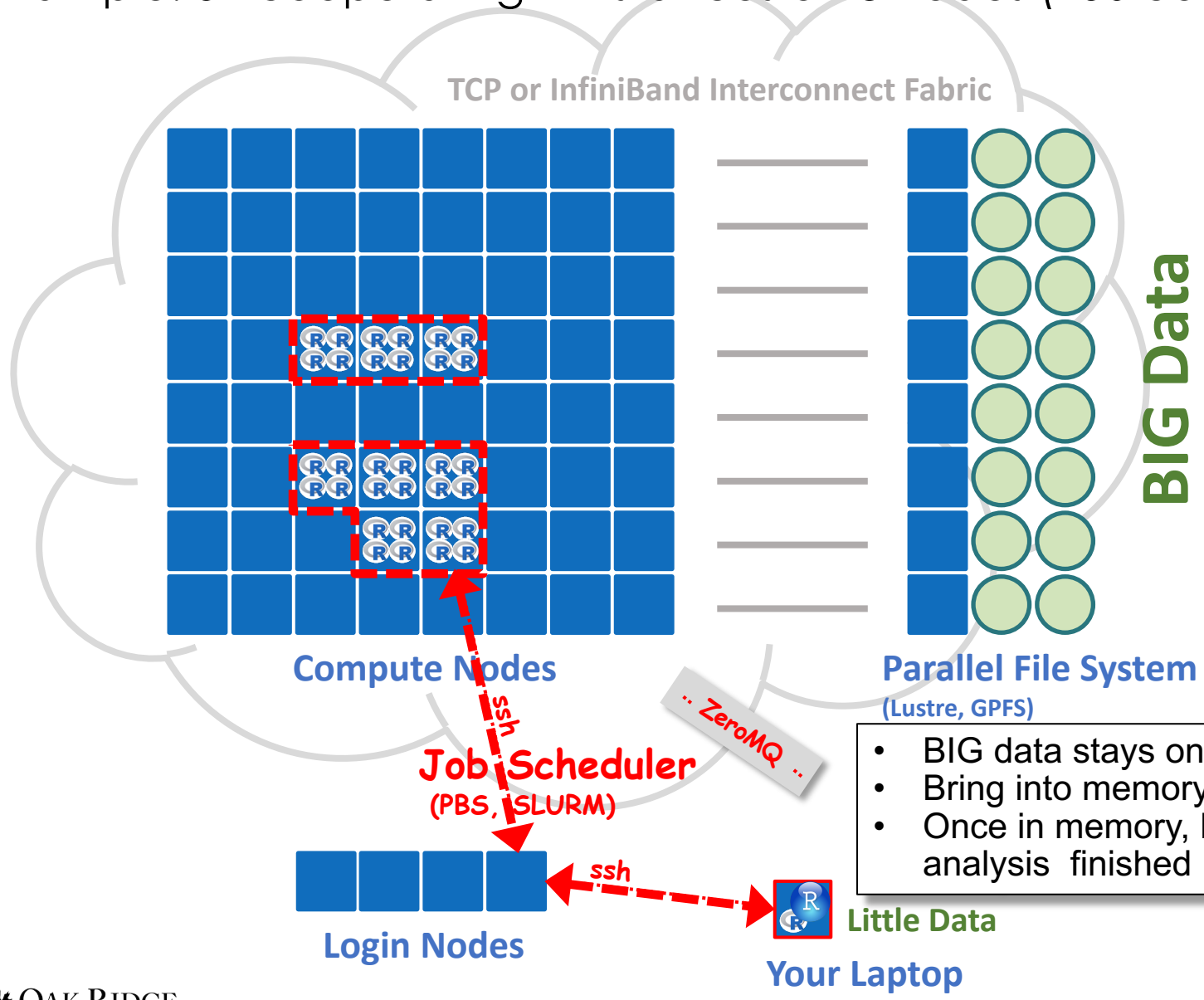




OAK RIDGE National Laboratory

64 node cluster (32 cores per node) = 2,048 cores

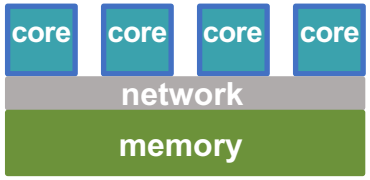
Example: 32 cooperating R instances on 8 nodes (256 cores, 8 per R)



- BIG data stays on big system
- Bring into memory in parallel
- Once in memory, keep it there until analysis finished

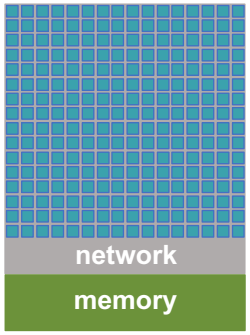
Three Flavors of (Parallel) Architecture

Shared Memory
Processor

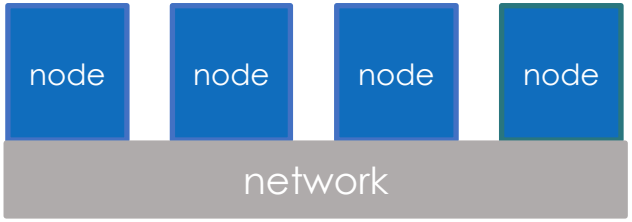


Multicore

Shared Memory
Co-Processor

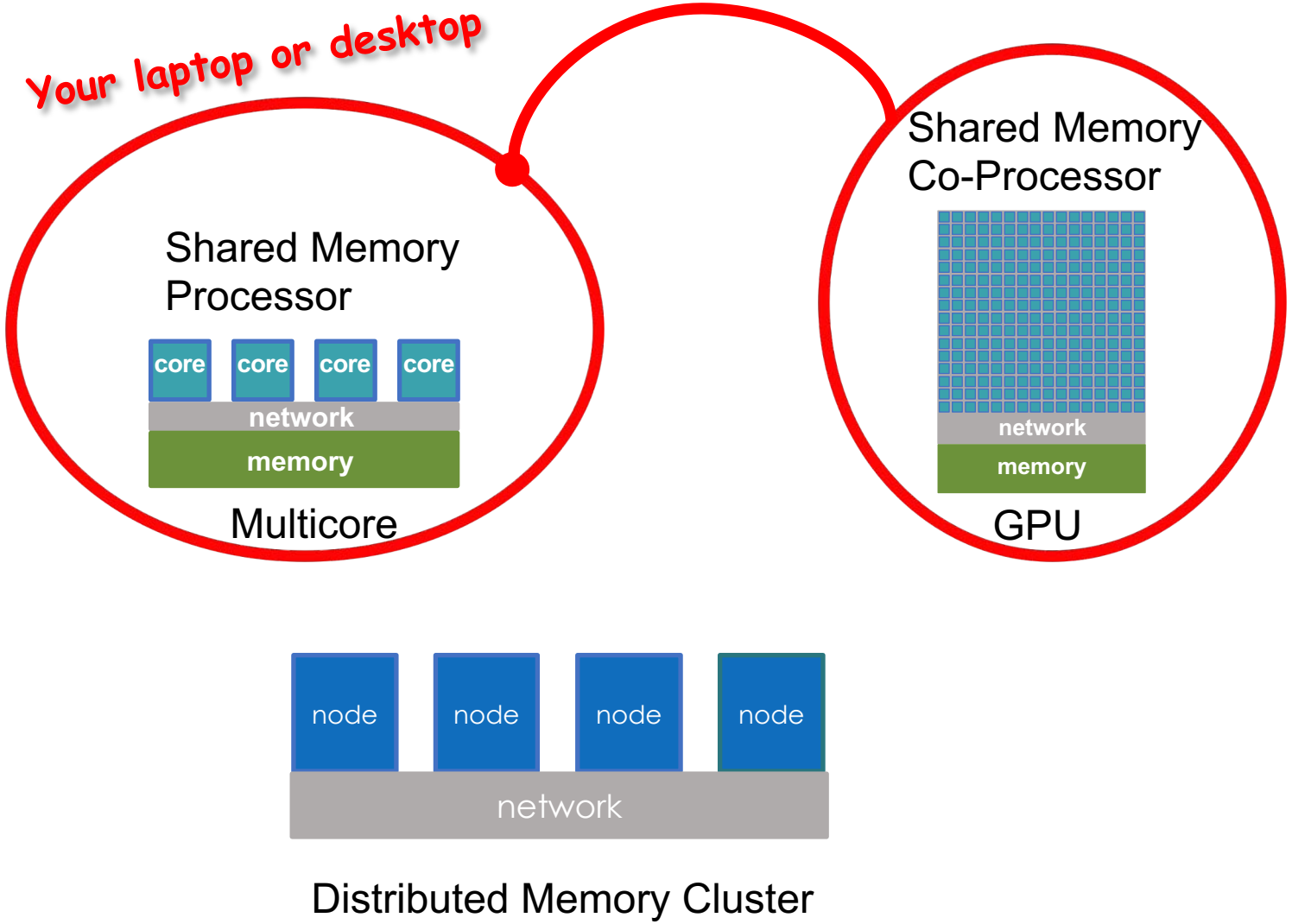


GPU

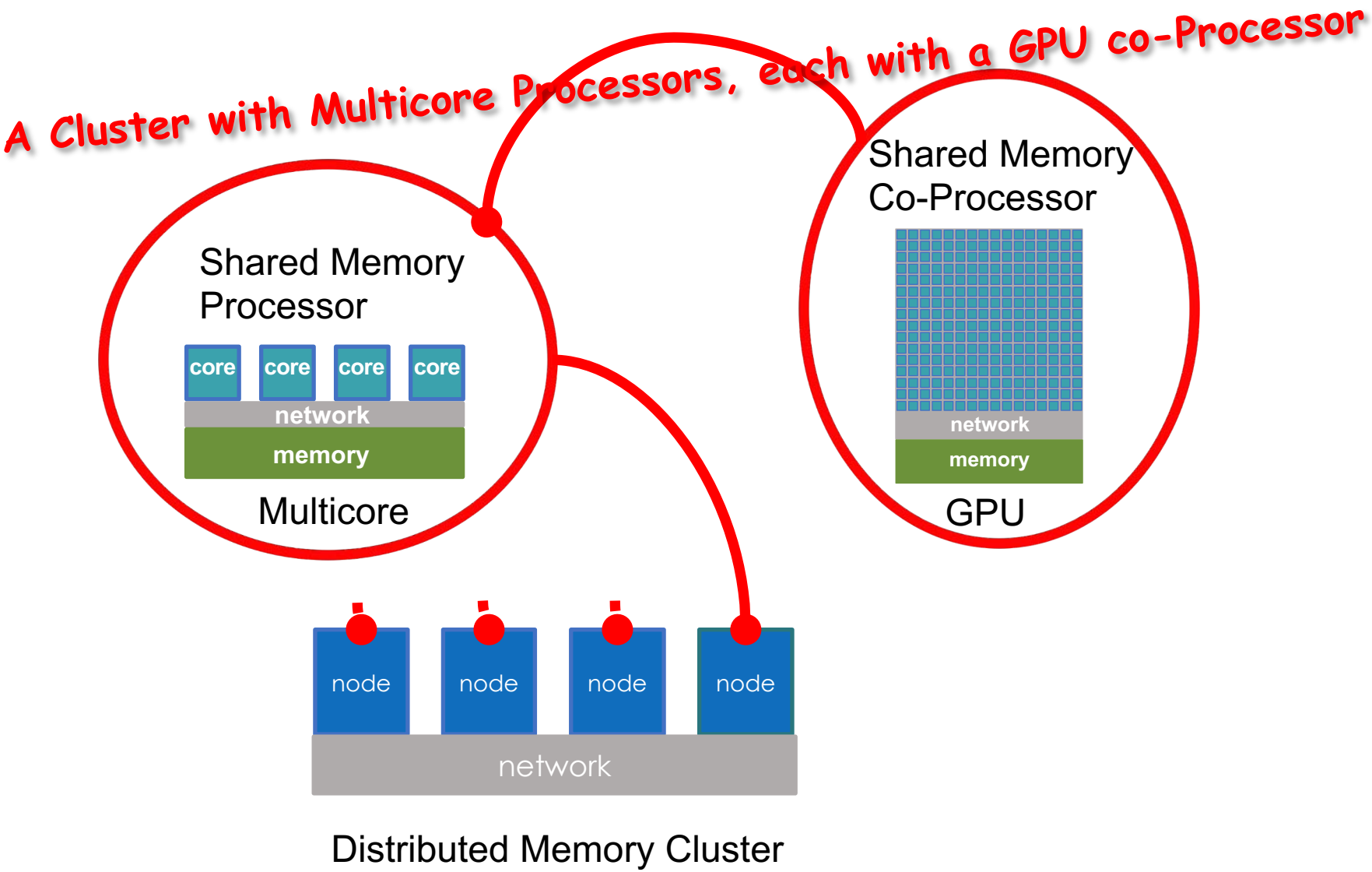


Distributed Memory Cluster

Three Flavors of (Parallel) Architecture



Three Flavors of (Parallel) Architecture

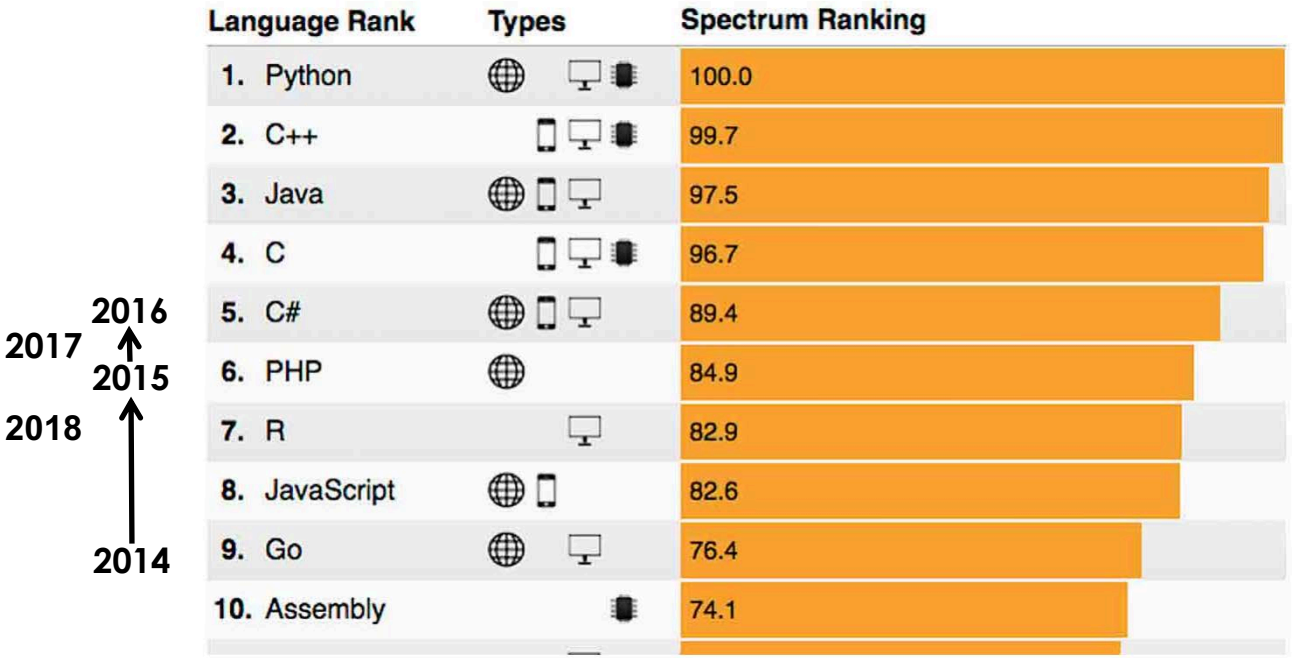


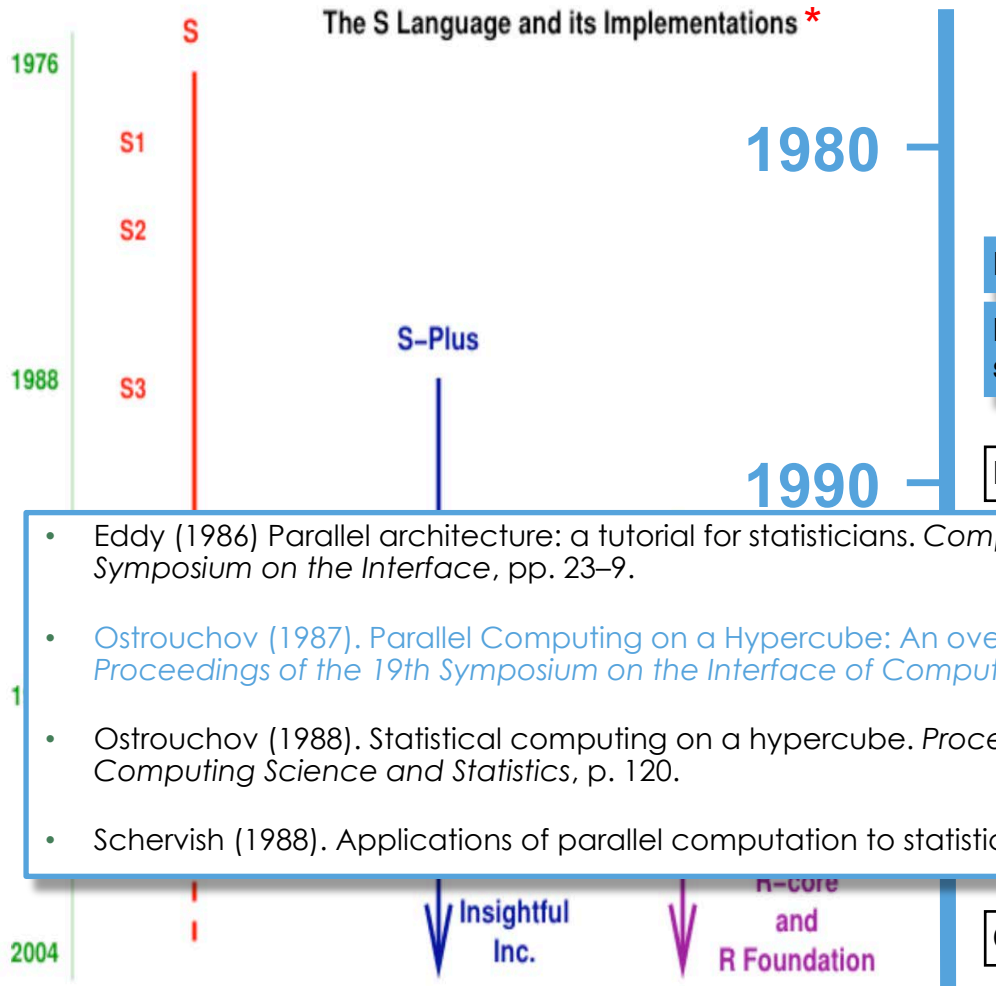
Why R?
Diversity for Data and Depth for Statistics in R

OAK RIDGE National Laboratory

Software for Data Analysis
The R Programing Language

2018 IEEE Spectrum's Ranking of Programming Languages





1980

Early distributed systems produced (iPSC, NCUBE)

Numerical linear algebra embraces parallel computing, some interest in statistics

1990

PVM developed

- Eddy (1986) Parallel architecture: a tutorial for statisticians. *Computer Science and Statistics, Proceedings of the 18th Symposium on the Interface*, pp. 23–9.
- Ostrouchov (1987). Parallel Computing on a Hypercube: An overview of the architecture and some applications. *Proceedings of the 19th Symposium on the Interface of Computer Science and Statistics*, p.27-32.
- Ostrouchov (1988). Statistical computing on a hypercube. *Proceedings of the 20th Symposium on the Interface: Computing Science and Statistics*, p. 120.
- Schervish (1988). Applications of parallel computation to statistical inference. J. Amer. Statist. Assoc., 83:976 – 983.

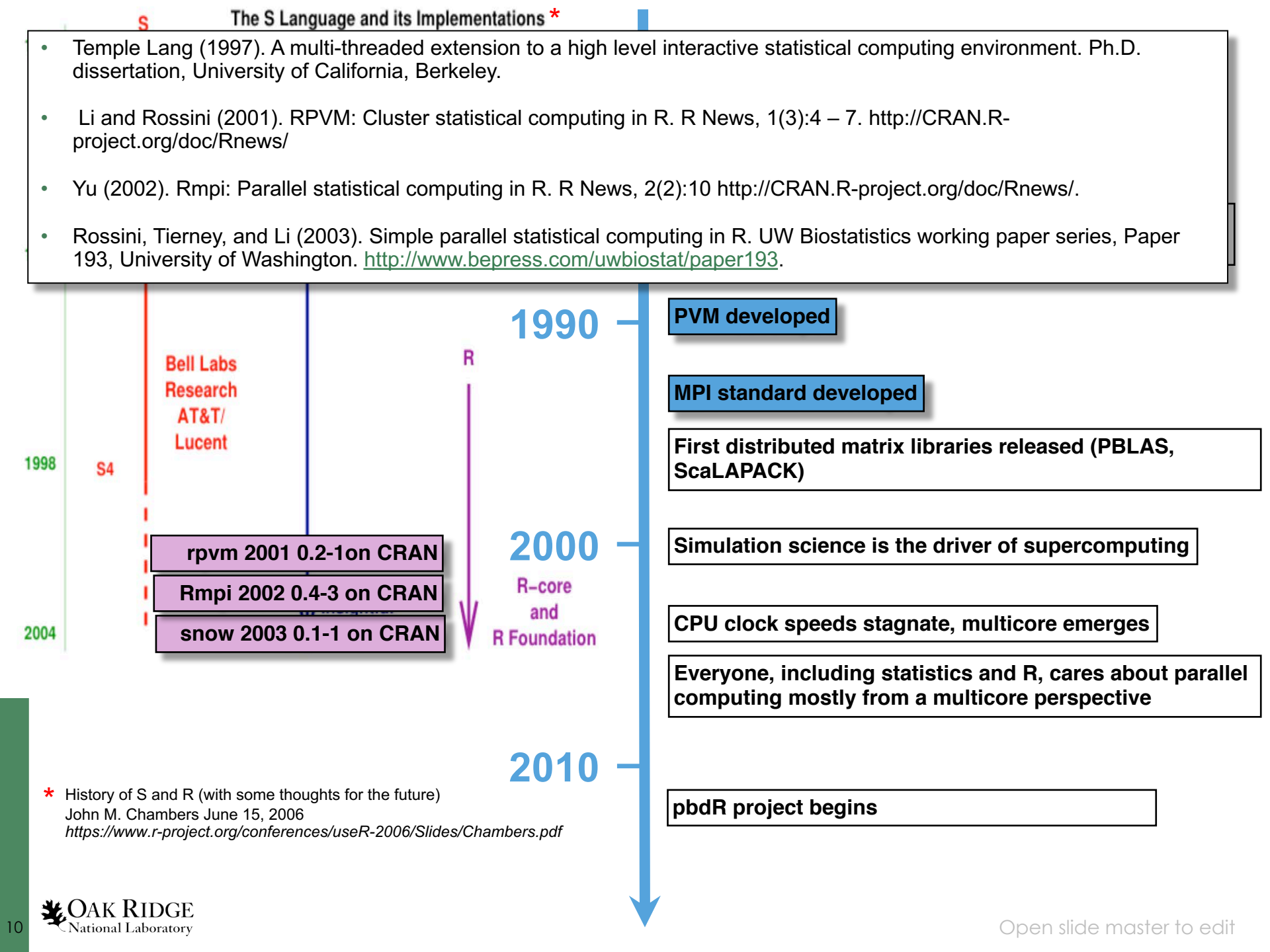
CPU clock speeds stagnate, multicore emerges

Everyone, including statistics and R, cares about parallel computing mostly from a multicore perspective

2010

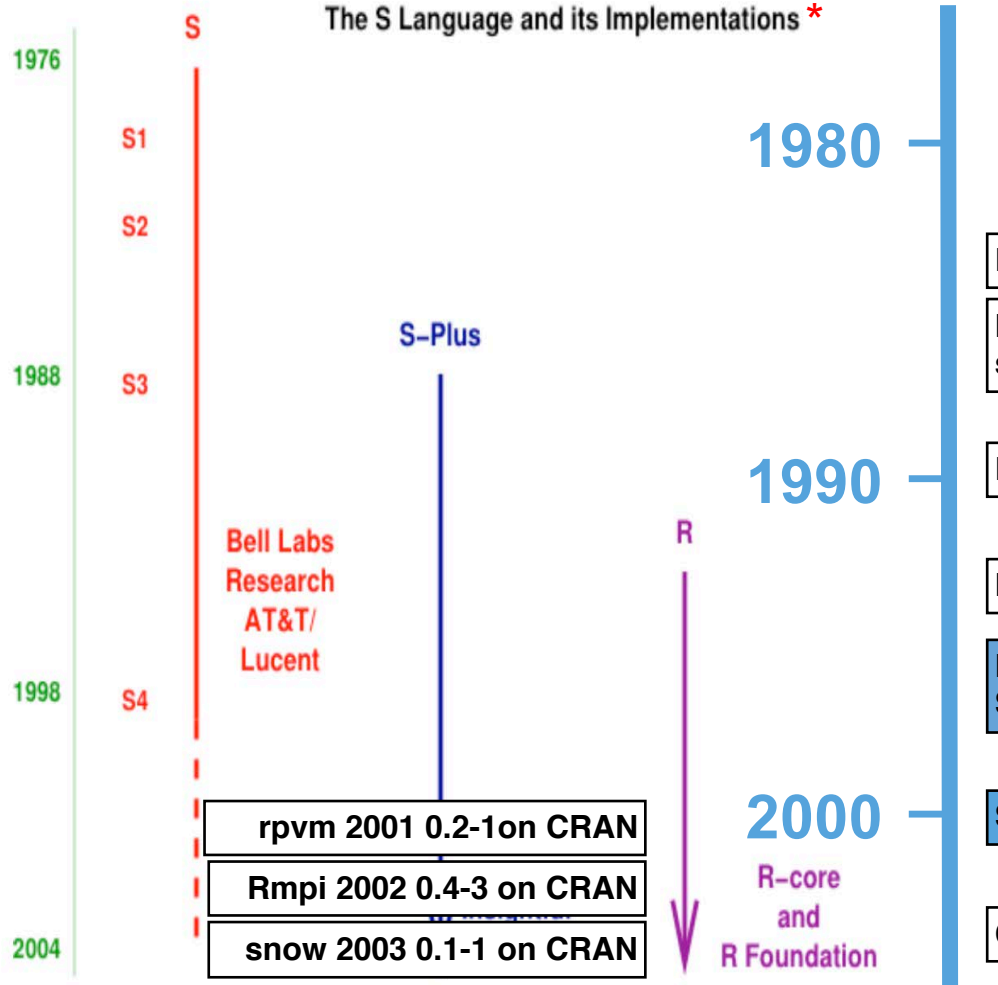
pbdR project begins

* 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



- Temple Lang (1997). A multi-threaded extension to a high level interactive statistical computing environment. Ph.D. dissertation, University of California, Berkeley.
- Li and Rossini (2001). RPVM: Cluster statistical computing in R. R News, 1(3):4 – 7. http://CRAN.R-project.org/doc/Rnews/
- Yu (2002). Rmpi: Parallel statistical computing in R. R News, 2(2):10 http://CRAN.R-project.org/doc/Rnews/.
- Rossini, Tierney, and Li (2003). Simple parallel statistical computing in R. UW Biostatistics working paper series, Paper 193, University of Washington. http://www.bepress.com/uwbiostat/paper193.

* 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



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

Simulation science is the driver of supercomputing

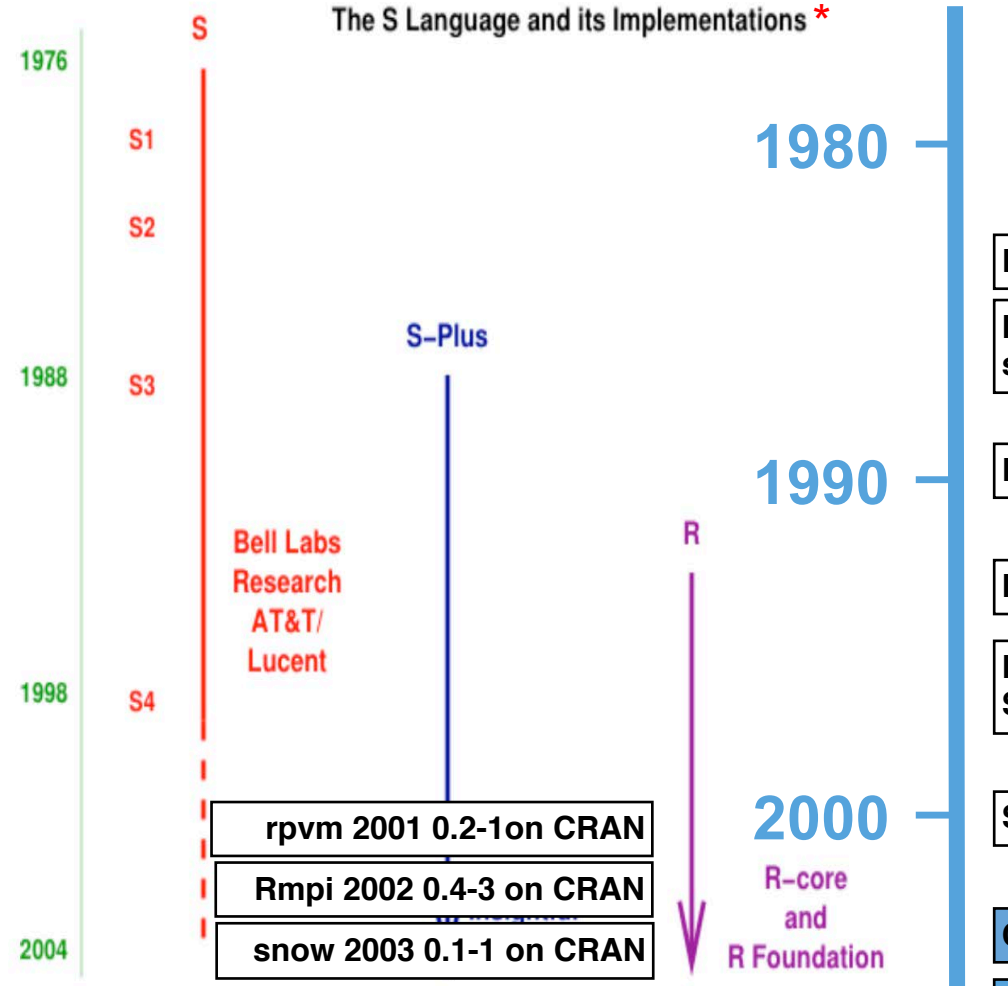
CPU clock speeds stagnate, multicore emerges

Everyone, including statistics and R, cares about parallel computing mostly from a multicore perspective

2010

pbdR project begins

* 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



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

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

parallel = multicore + snow 2011 0.1-1 on CRAN

2010

first pbdR project release 2012 on CRAN

pbdR project begins

* 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

pbdR Project to Bring Statistical Science and R to HPC

Core Team (leadership and lead developers)

Wei-Chen Chen, US FDA

George Ostrouchov, ORNL & UTK

Drew Schmidt, ORNL

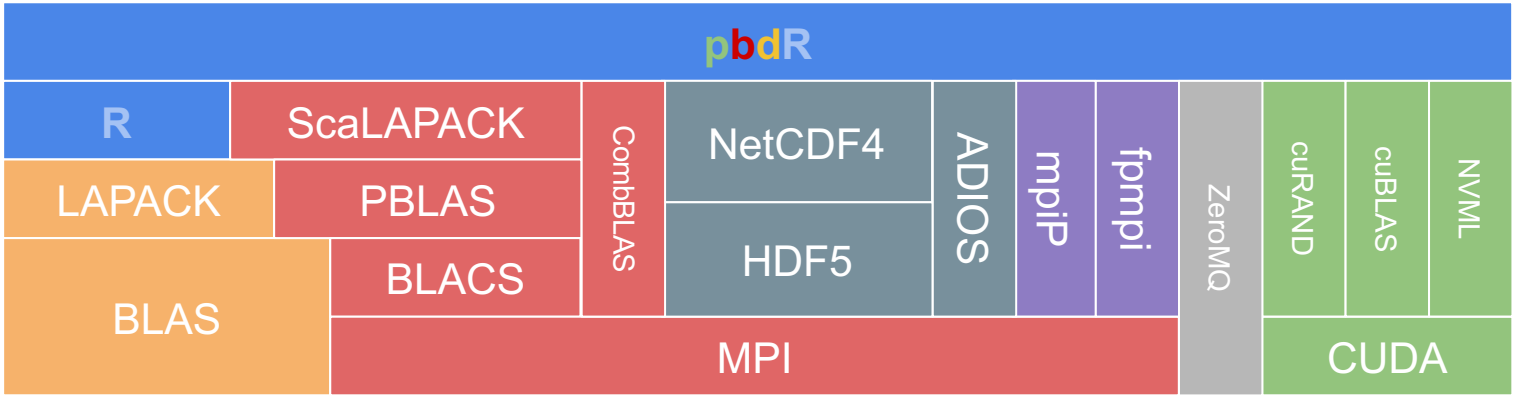
pbdR
Programming with Big Data in R

Developers

Christian Heckendorf, Yuping Lu, Michael Matheson, Pragneshkumar Patel, Gaurav Sehrawat , Amil Williamson

Contributors

Moustaffa Ahmed, Whit Armstrong, Ewan Higgs, Sébastien Lamy de la Chapelle, Michael Lawrence, Andrey Paul, A. Philipp, David Pierce, Brian Ripley, Elliott Sales, Aiello Spencer, ZhaoKang Wang, Hao Yu



OAK RIDGE National Laboratory

Open slide master to edit

"… pbdR … outperformed all the other systems in almost all cases on dense data."*

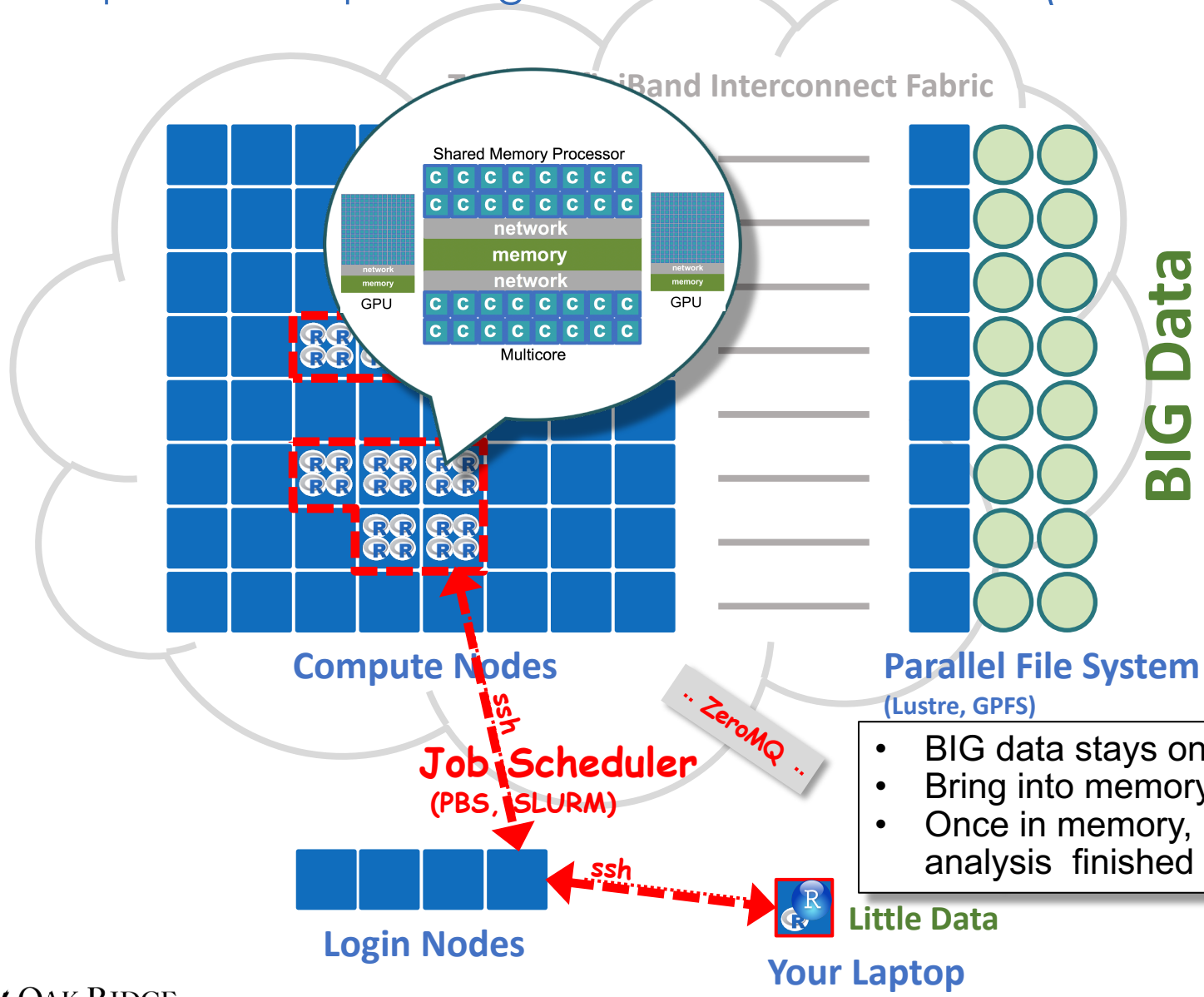
- An independent comparison, on small distributed systems with up to 8 nodes (224 cores)
- Systems considered were ***MADlib***, ***MLlib***, ***SystemML***, ***TensorFlow***, *and* ***pbdR***

* Anthony Thomas, Arun Kumar: **A Comparative Evaluation of Systems for Scalable Linear Algebra-based Analytics**. *Proceedings of the VLDB Endowment,* Volume **11,** No. **13,** September 2018, p. 2168-2182.

pbdR
Programming with Big Data in R

64 node cluster (32 cores per node) = 2,048 cores

Example: 32 cooperating R instances on 8 nodes (256 cores, 8 per R)



- BIG data stays on big system
- Bring into memory in parallel
- Once in memory, keep it there until analysis finished

Combining MPI and mclapply()

https://github.com/RBigData/mpi_balance

Example: 16 MPI ranks, each using available cores

hello_balance.pbs

```
[ost@or-condo-login02 mpi_balance]$ cat hello_balance.pbs
#!/bin/bash
#PBS -N balance
#PBS -A ccsd
#PBS -l qos=std,nodes=4:ppn=32,walltime=00:02:00
#PBS -q batch
#PBS -W group_list=cades-ccsd
#PBS -e balance.e
#PBS -o balance.o

cd ~/mpi_balance
pwd

## module names can vary on different platforms
module load R
echo "loaded R"
module list

## prevent warning when fork is used with MPI
export OMPI_MCA_mpi_warn_on_fork=0

mpirun -np 16 --map-by ppr:4:node Rscript hello_balance.R
```

hello_balance.R

```
[ost@or-condo-login02 mpi_balance]$ cat hello_balance.R
suppressMessages(library(pbdMPI))
suppressMessages(library(parallel))

host = system("hostname", intern = TRUE)

mc.function = function(x) {
    ## Put code for mclapply cores here
    Sys.getpid() # returns process id
}

## Compute how many cores per R session are on this node
ranks_per_node = as.numeric(system("echo $PBS_NUM_NODES", intern = TRUE))
cores_on_my_node = detectCores()
cores_total = allreduce(cores_on_my_node)
cores_per_R = floor(cores_on_my_node/ranks_per_node)

## Run lapply on allocated cores to demonstrate fork pids
my_pids = mclapply(1:cores_per_R, mc.function, mc.cores = cores_per_R)
my_pids = do.call(paste, my_pids) # combines results from mclapply

## Same cores available for OpenBLAS (see openblasctl package)
##            or for other OpenMP enabled codes outside mclapply.
## If BLAS functions are called inside mclapply, they compete for the
##            same cores: avoid or manage appropriately.

## Now report what happened and where
msg = paste0("Hello World from rank ", comm.rank(), " on host ", host, "\n",
             "      with ", cores_per_R, " cores allocated (", ranks_per_node,
             " R sessions sharing ", cores_on_my_node, " cores).\n",
             "      pid: ", my_pids, "\n")
comm.cat(msg, quiet = TRUE, all.rank = TRUE)

comm.cat("Total R sessions:", comm.size(), "Total cores:", cores_total, "\n",
         quiet = TRUE)

finalize()
```

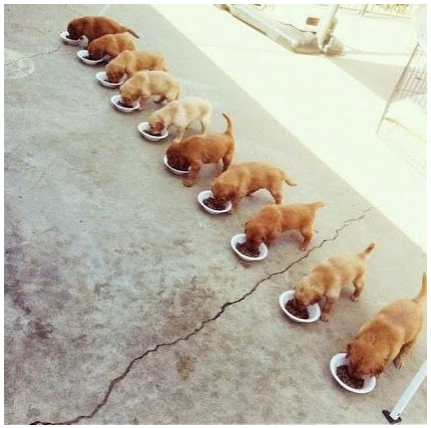
On Windows, use Docker for mclapply()
https://hub.docker.com/r/rbigdata/mpi

RBigData/mpi_balance

```
[ost@or-condo-login02 mpi_balance]$ cat hello_balance.pbs
#!/bin/bash
#PBS -N balance
#PBS -A ccsd
#PBS -l qos=std,nodes=4:ppn=32,walltime=00:02:00
#PBS -q batch
#PBS -W group_list=cades-ccsd
#PBS -e balance.e
#PBS -o balance.o

cd ~/mpi_balance
pwd

## module names can vary on different platforms
module load R
echo "loaded R"
module list

## prevent warning when fork is used with MPI
export OMPI_MCA_mpi_warn_on_fork=0

mpirun -np 16 --map-by ppr:4:node Rscript hello_balance.R
```

```
[ost@or-condo-login02 mpi_balance]$ cat balance.o
/home/ost/mpi_balance
loaded R
Hello World from rank 0 on host or-condo-c193.ornl.gov
      with 9 cores allocated (4 R sessions sharing 36 cores).
      pid: 169183 169188 169192 169196 169200 169204 169208 169212 169216
Hello World from rank 1 on host or-condo-c193.ornl.gov
      with 9 cores allocated (4 R sessions sharing 36 cores).
      pid: 169181 169185 169189 169193 169198 169201 169206 169210 169214
Hello World from rank 2 on host or-condo-c193.ornl.gov
      with 9 cores allocated (4 R sessions sharing 36 cores).
      pid: 169184 169187 169191 169195 169199 169203 169207 169211 169215
Hello World from rank 3 on host or-condo-c193.ornl.gov
      with 9 cores allocated (4 R sessions sharing 36 cores).
      pid: 169182 169186 169190 169194 169197 169202 169205 169209 169213
Hello World from rank 4 on host or-condo-c54.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 39877 39881 39885 39889 39894 39897 39901 39905
Hello World from rank 5 on host or-condo-c54.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 39879 39883 39887 39891 39893 39898 39902 39906
Hello World from rank 6 on host or-condo-c54.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 39878 39882 39886 39890 39895 39899 39903 39907
Hello World from rank 7 on host or-condo-c54.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 39876 39880 39884 39888 39892 39896 39900 39904
Hello World from rank 8 on host or-condo-c151.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 70877 70881 70885 70889 70893 70897 70901 70905
Hello World from rank 9 on host or-condo-c151.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 70879 70883 70887 70891 70895 70899 70903 70907
Hello World from rank 10 on host or-condo-c151.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 70876 70880 70884 70888 70892 70896 70900 70904
Hello World from rank 11 on host or-condo-c151.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 70878 70882 70886 70890 70894 70898 70902 70906
Hello World from rank 12 on host or-condo-c124.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 16014 16018 16022 16026 16030 16034 16038 16042
Hello World from rank 13 on host or-condo-c124.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 16012 16016 16020 16024 16028 16032 16036 16040
Hello World from rank 14 on host or-condo-c124.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 16011 16015 16019 16023 16027 16031 16035 16039
Hello World from rank 15 on host or-condo-c124.ornl.gov
      with 8 cores allocated (4 R sessions sharing 32 cores).
      pid: 16013 16017 16021 16025 16029 16033 16037 16041
Total R sessions: 16 Total cores: 528
```

Multithreading Theory and Practice



https://www.reddit.com/r/aww/comments/2oagj8/multithreaded_programming_theory_and_practice/

Statisticians are Needed in BIG Data Analysis

Where to Get a Cluster Allocation:

- **Your university or company cluster computer,** or another source, including:
- USA: XSEDE http://xsede.org (NSF funded, open to US institutions), INCITE http://doeleadershipcomputing.org (DOE funded, open competition)
- EU: PRACE http://www.prace-ri.eu/ (Partnership for Advanced Computing in Europe)
- UK: NERC HPC https://nerc.ukri.org/research/sites/facilities/hpc/applying/
- Canada: Compute Canada account http://www.computecanada.ca and an allocation from SciNet http://www.scinethpc.ca
- Australia: National Computational Merit Allocation Scheme (NCMAS): https://ncmas.nci.org.au
- New Zealand eScience Infrastructure: https://www.nesi.org.nz/
- Amazon EC2: https://aws.amazon.com/hpc/efa/ Elastic Fabric Adapter (EFA)