Using R on HPC Clusters Part 2

George Ostrouchov

Oak Ridge National Laboratory

August 19, 2022

Get this presentation:

```
git clone https://github.com/RBigData/R4HPC.git
```

- Open

  ```
  R4HPC_Part2.html
  ```

  in your web browser (help ? toggle)

Slack workspace link for this workshop was emailed to you.

Many thanks to my colleagues and former colleagues who contributed to the software and ideas presented here. See the RBigData Organization on Github: https://github.com/RBigData. Also, many thanks to all R developers of packages used in this presentation.

Slides are made with the xaringan R package. It is an R Markdown extension based on the JavaScript library remark.js.

Today's Package Installs

See `R4HPC/code_4` R script and shell scripts for your machine

Running MPI on a Laptop

macOS in a Terminal window:

- `brew install openmpi`
- `mpirun -np 4 Rscript your_spmd_code.R`

Windows

- Web Page: https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi
- Download: https://www.microsoft.com/en-us/download/details.aspx?id=100593
- `pbdMPI` has a Windows binary on CRAN

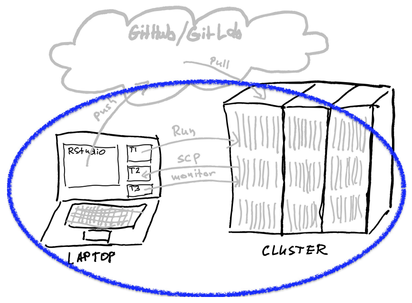
OAK RIDGE National Laboratory

Revisit `hello_balance.R` in `code_1`

```
suppressMessages(library(pbdMPI))
comm.print(sessionInfo())

## get node name
host = system("hostname", intern = TRUE)

mc.function = function(x) {
    Sys.sleep(1) # replace with your function for mclapply cores here
    Sys.getpid() # returns process id
}

## Compute how many cores per R session are on this node
local_ranks_query = "echo $OMPI_COMM_WORLD_LOCAL_SIZE"
ranks_on_my_node = as.numeric(system(local_ranks_query, intern = TRUE))
cores_on_my_node = parallel::detectCores()
cores_per_R = floor(cores_on_my_node/ranks_on_my_node)
cores_total = allreduce(cores_per_R)  # adds up over ranks

## Run mclapply on allocated cores to demonstrate fork pids
my_pids = parallel::mclapply(1:cores_per_R, mc.function, mc.cores = core
my_pids = do.call(paste, my_pids) # combines results from mclapply
##
## Same cores are shared with OpenBLAS (see flexiblas package)
##            or for other OpenMP enabled codes outside mclapply.
```

Working with a remote cluster using R

Running Distributed on a Cluster



7 / 29

Section I: Environment and Workflow

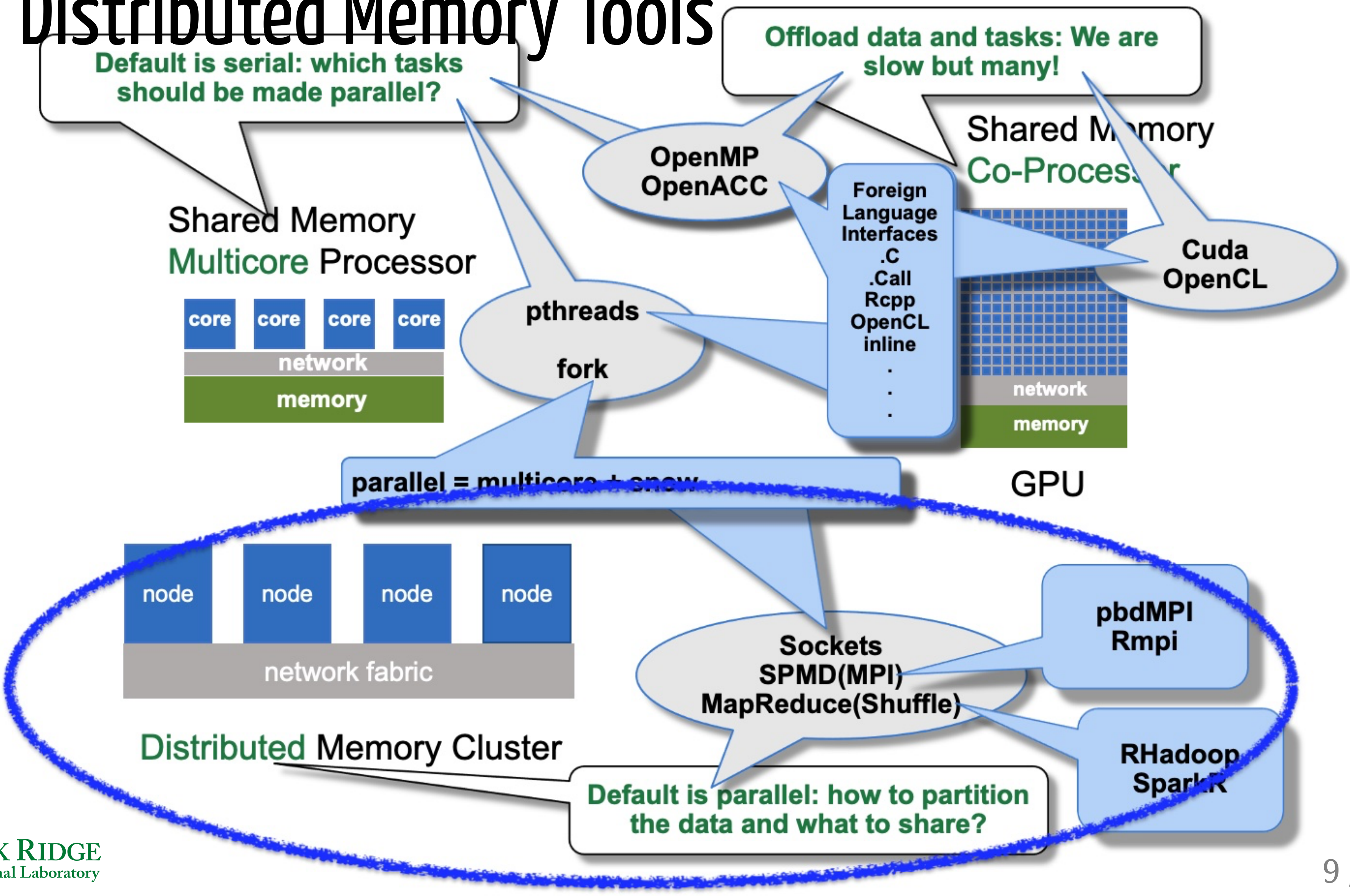
Section II: Parallel Hardware and Software Overview

Section III: Shared Memory Tools

Section IV: Distributed Memory Tools

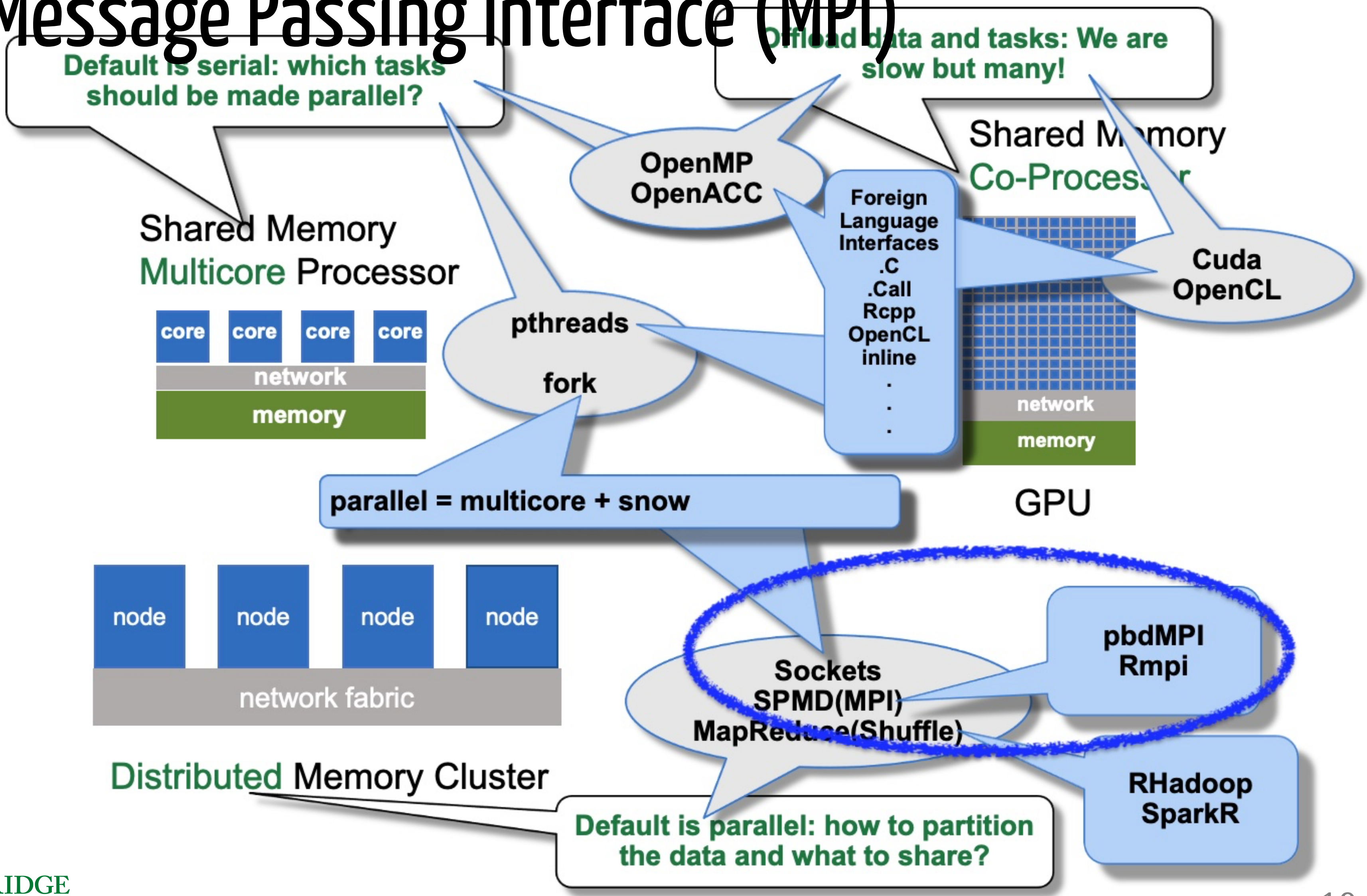
OAK RIDGE National Laboratory

Distributed Memory Tools


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory Co-Processor
OpenMP OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline . . .
Cuda OpenCL
Shared Memory Multicore Processor
core core core core
network
memory
pthreads fork
network
memory
GPU
parallel = multicore + snow
node node node node
network fabric
Sockets SPMD(MPI) MapReduce(Shuffle)
pbdMPI Rmpi
RHadoop SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Oak Ridge National Laboratory

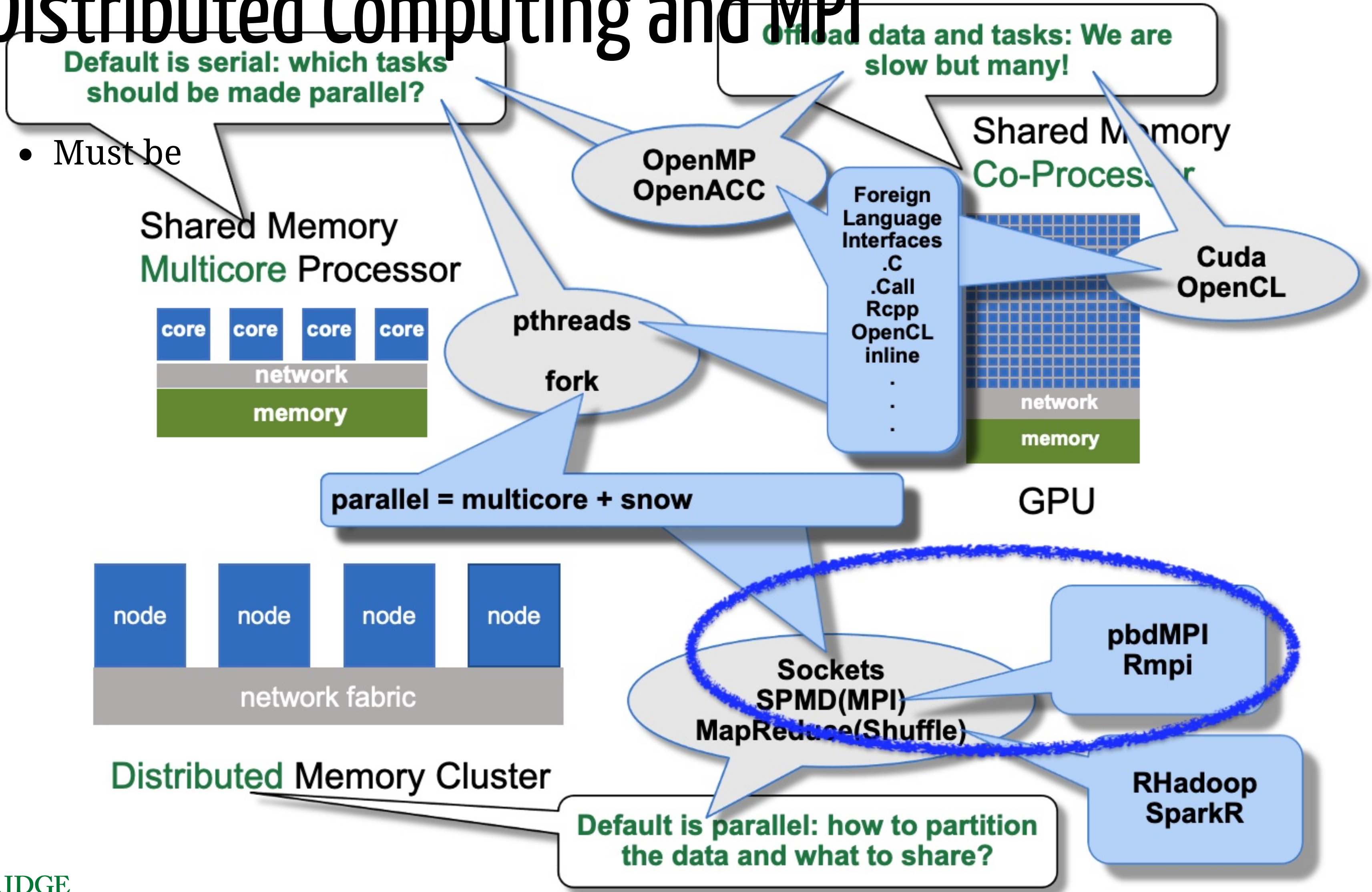
Message Passing Interface (MPI)



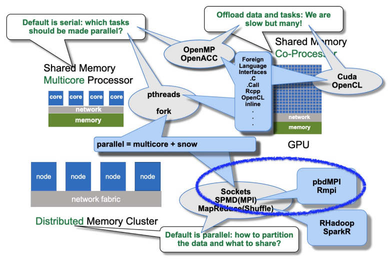
OAK RIDGE National Laboratory

Distributed Computing and MPI

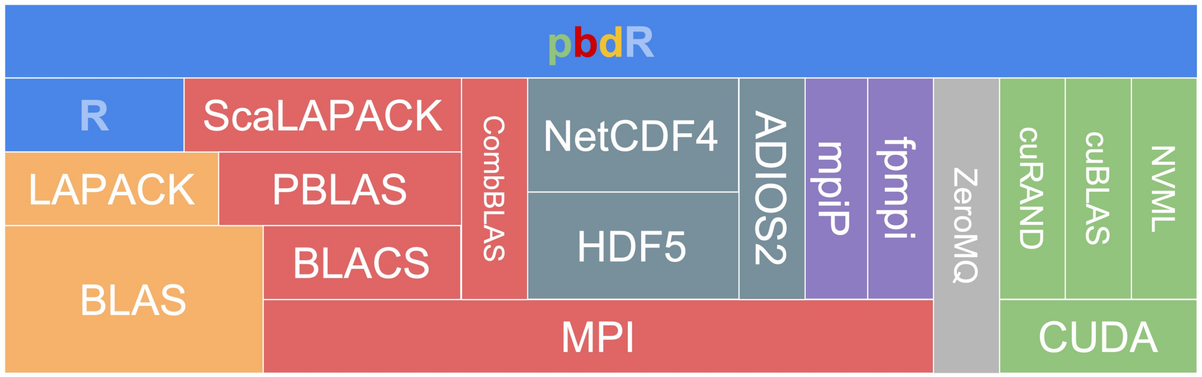
- Must be


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory Multicore Processor
core
core
core
core
network
memory
OpenMP
OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline . . .
Shared Memory Co-Processor
Cuda
OpenCL
network
memory
GPU
pthreads
fork
parallel = multicore + snow
node
node
node
node
network fabric
Distributed Memory Cluster
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Default is parallel: how to partition the data and what to share?

OAK RIDGE National Laboratory

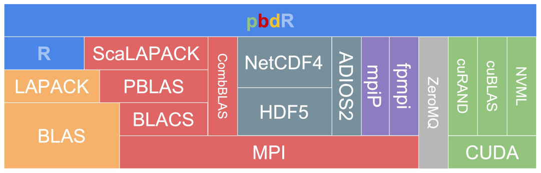


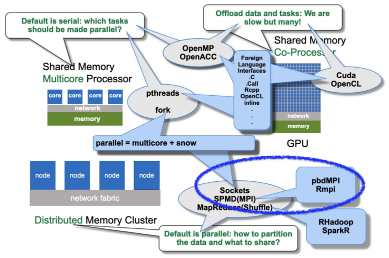
pbdR Project



- Bridge HPC with high-productivity of R: Expressive for data and modern statistics
- 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

OAK RIDGE National Laboratory

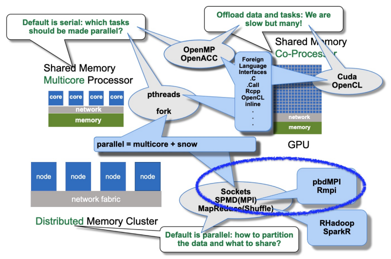



Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory Multicore Processor
Shared Memory Co-Processor
OpenMP OpenACC
Cuda OpenCL
pthreads
fork
parallel = multicore + snow
GPU
network fabric
Distributed Memory Cluster
Sockets SPMD(MPI) MapReduce(Shuffle)
pbdMPI Rmpi
RHadoop SparkR
Default is parallel: how to partition the data and what to share?

Package `pbdMPI`

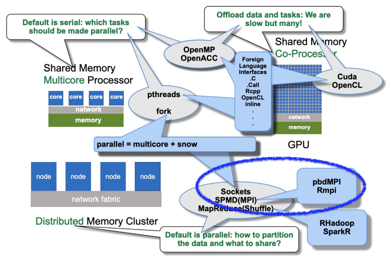
- Specializes in SPMD programming for HPC clusters
    - Manages printing from ranks
    - Provides chunking options
    - Provides communicator splits for multilevel parallelism
    - In situ capability to process data from other MPI codes without copy
- A derivation and rethinking of the `Rmpi` package aimed at HPC clusters
    - Simplified interface with fewer parameters (using R's S4 methods)
    - Faster for matrix and array data - no serialization

13 / 29


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
core
core
core
core
network
memory
OpenMP
OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline . . .
Shared Memory
Co-Processor
Cuda
OpenCL
pthreads
fork
network
memory
GPU
parallel = multicore + snow
node
node
node
node
network fabric
Distributed Memory Cluster
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Default is parallel: how to partition the data and what to share?

Single Program Multiple Data (SPMD)

- One code and a parallel mindset
- A generalization of a serial code
- Many rank-aware operations are automated
- Collective operations are high level and easy to learn
- Explicit point-to-point communications are an advanced topic
- No manager, it is all cooperation


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
core
core
core
core
network
memory
OpenMP
OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline . . .
Shared Memory
Co-Processor
Cuda
OpenCL
pthreads
fork
network
memory
GPU
parallel = multicore + snow
node
node
node
node
network fabric
Distributed Memory Cluster
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Default is parallel: how to partition the data and what to share?

High-level Collective Communications

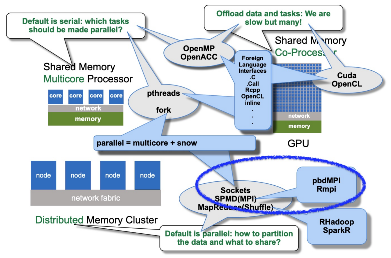
$$\mathbf{A} = \sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{X}_\mathbf{i}$$

pbdMPI: **A = reduce(X)** **A = allreduce(X)**

$$\mathbf{A} = [\mathbf{X}_1|\mathbf{X}_2|\cdots|\mathbf{X}_\mathbf{n}]$$

pbdMPI: **A = gather(X)** **A = allgather(X)**

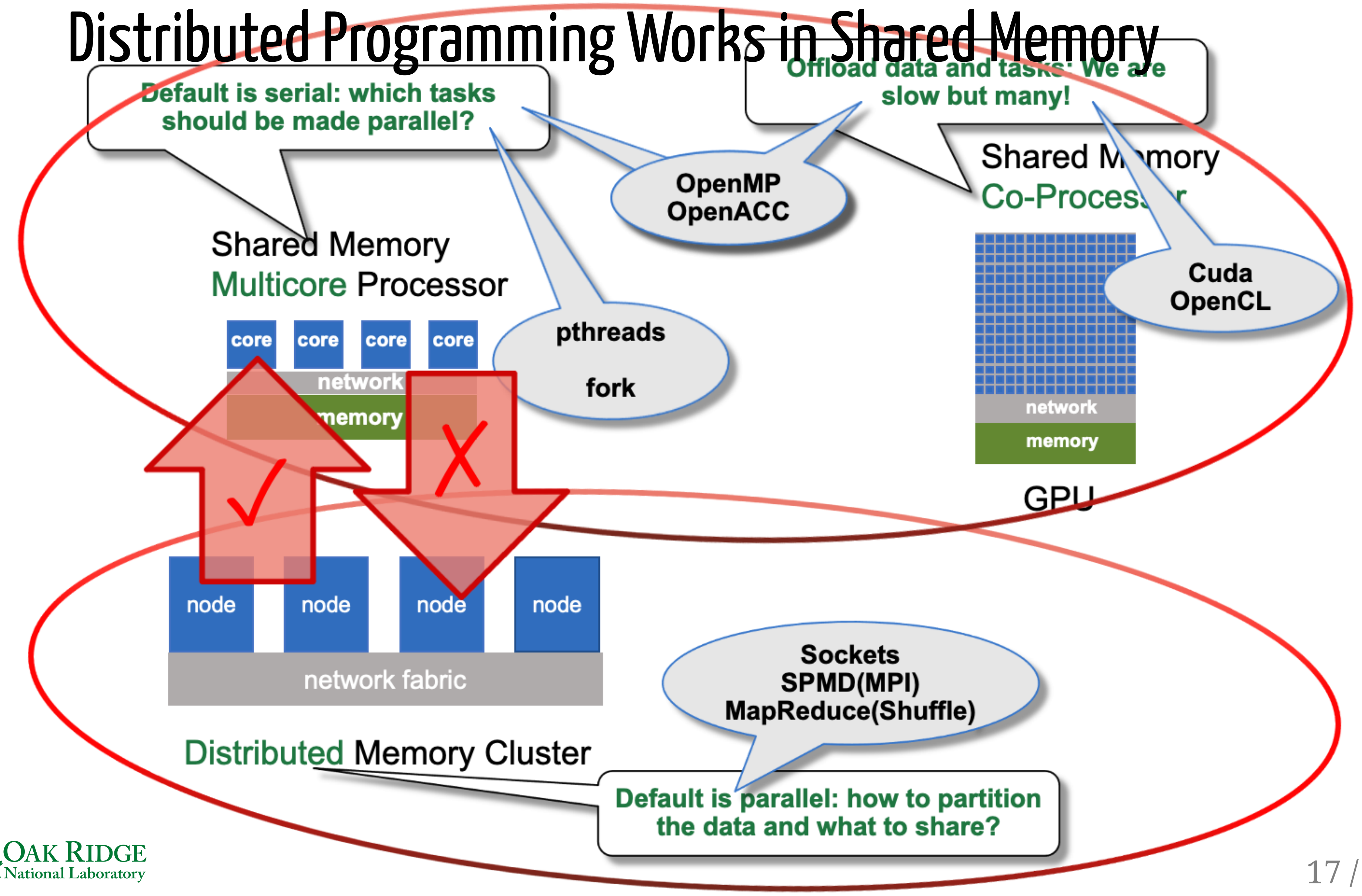
OAK RIDGE National Laboratory


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
core
core
core
core
network
memory
OpenMP
OpenACC
Shared Memory
Co-Processor
Foreign Language Interfaces .C .Call Rcpp OpenCL inline
Cuda
OpenCL
pthreads
fork
network
memory
parallel = multicore + snow
GPU
node
node
node
node
network fabric
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

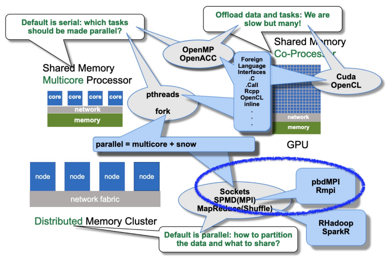
Functions to Facilitate SPMD Programming

- **comm.chunk()** splits a number into chunks in various ways and various formats. Tailored for SPMD programming, returning correct (usually different) results to each rank.
- **comm.set.seed()** sets the seed of a parallel RNG. If diff = FALSE, then all ranks generate the same stream. Otherwise, ranks generate different streams.
- **comm.print()** and **comm.cat()** print by default from rank 0 only, with options to print from any or all ranks.

Distributed Programming Works in Shared Memory



OAK RIDGE National Laboratory

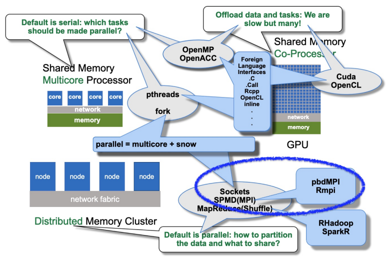

Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
core
core
core
core
network
memory
OpenMP
OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline
Shared Memory
Co-Processor
Cuda
OpenCL
pthreads
fork
network
memory
parallel = multicore + snow
GPU
node
node
node
node
network fabric
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Hands on Session 5: Hello MPI Ranks

`code_5/hello_world.R`

```r
suppressMessages(library(pbdMPI))

my_rank = comm.rank()
nranks = comm.size()
msg = paste0("Hello World! My name is Rank", my_rank,
             ". We are ", nranks, " identical siblings.")
cat(msg, "\n")

finalize()
```

Rank distinguishes the parallel copies of the same code


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
core
network
memory
OpenMP
OpenACC
Foreign Language Interfaces .C .Call Rcpp OpenCL inline
Shared Memory
Co-Processor
Cuda
OpenCL
pthreads
fork
network
memory
GPU
parallel = multicore + snow
node
network fabric
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Hands on Session 5: Random Forest with MPI

`code_5/rf_mpi.R`

```
suppressPackageStartupMessages(library(randomForest))
data(LetterRecognition, package = "mlbench")
library(pbdMPI, quiet = TRUE)
comm.set.seed(seed = 7654321, diff = FALSE)

n = nrow(LetterRecognition)
n_test = floor(0.2 * n)
i_test = sample.int(n, n_test)
train = LetterRecognition[-i_test, ]
test = LetterRecognition[i_test, ][comm.chunk(n_test, form = "vector"),

comm.set.seed(seed  = 1234, diff = TRUE)
my.rf = randomForest(lettr ~ ., train, ntree = comm.chunk(500), norm.vot
rf.all = allgather(my.rf)
rf.all = do.call(combine, rf.all)
pred = as.vector(predict(rf.all, test))

correct = allreduce(sum(pred == test$lettr))
comm.cat("Proportion Correct:", correct/(n_test), "\n")

finalize()
```

Hands on Session 5: `comm.chunk()`

`mpi_shorts/chunk.r`

```r
library( pbdMPI, quiet = TRUE )

my.rank = comm.rank( )

k = comm.chunk( 10 )
comm.cat( my.rank, ":", k, "\n", all.rank = TRUE, quiet = TRUE)

k = comm.chunk( 10 , form = "vector")
comm.cat( my.rank, ":", k, "\n", all.rank = TRUE, quiet = TRUE)

k = comm.chunk( 10 , form = "vector", type = "equal")
comm.cat( my.rank, ":", k, "\n", all.rank = TRUE, quiet = TRUE)

finalize( )
```

Hands on Session 5: other short MPI codes

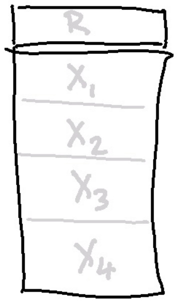
bcast.r chunk.r comm_split.R cov.r gather-named.r gather.r gather-unequal.r hello-p.r hello.r map-reduce.r mcsim.r ols.r qr-cop.r rank.r reduce-mat.r timer.r

- These short codes only use `pbdMPI` and can run on a laptop in a terminal window if you installed OpenMPI
- On the clusters these can run on a login node with a small * number of ranks
- Wile in the `mpi_shorts` directory, run the following
  - `source ../code_4/modules_MACHINE.sh`
  - `mpirun -np 4 Rscript your_script.r`

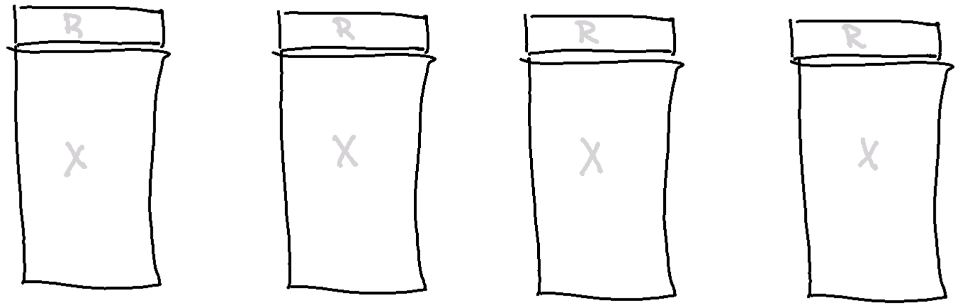
* Note that running long or large jobs on login nodes is strongly discouraged

Shared Memory - MPI or fork?

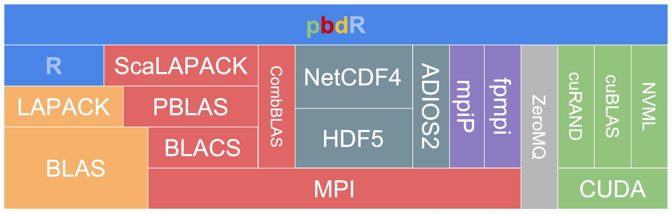
- fork via `mclapply()` + `do.call()` combine

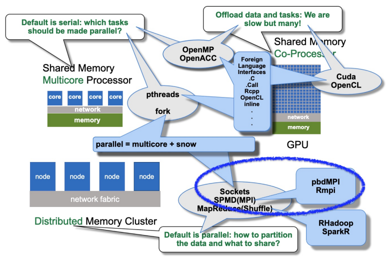


- MPI replicated data + `allreduce()`



- MPI chunked data + `allreduce()`

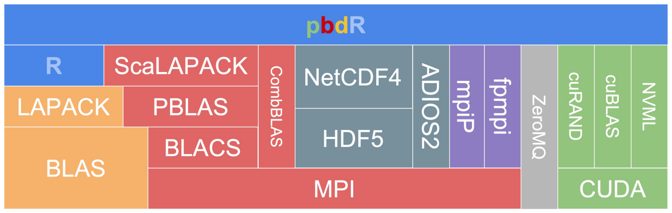


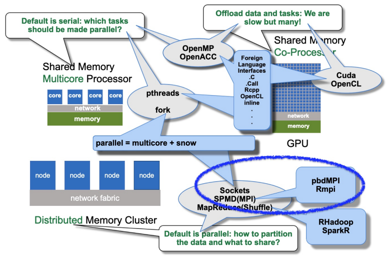

Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
Shared Memory
Co-Processor
OpenMP
OpenACC
pthreads
fork
Cuda
OpenCL
parallel = multicore + snow
GPU
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Package pbdDMAT

- ScaLAPACK: Scalable LAPACK - Distributed version of LAPACK (uses PBLAS/BLAS but not LAPACK)
  - 2d Block-Cyclic data layout - mostly automated in `pbdDMAT` package
  - BLACS: Communication collectives for distributed matrix computation
  - PBLAS: BLAS - distributed BLAS (uses shared memory BLAS within blocks)
- Most matrix operations in R code are identical to serial through overloading operators and `ddmatrix` class

OAK RIDGE National Laboratory




Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
OpenMP
OpenACC
Shared Memory
Co-Processor
Cuda
OpenCL
pthreads
fork
parallel = multicore + snow
GPU
Sockets
SPMD(MPI)
MapReduce(Shuffle)
pbdMPI
Rmpi
RHadoop
SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Package pbdML

- A demonstration of `pbdDMAT` package capabilities
- Includes
    - Randomized SVD
    - Randomized principal components analysis
    - Robust Principal Component Analysis?" from https://arxiv.org/pdf/0912.3599.pdf

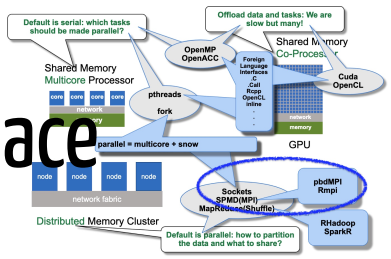
Hands on Session rsvd: Randomized sketching algorithms

Fast new alternatives to classical numerical linear algebra computations.

Guarantees are given with probability statements instead of classical error analysis.

Martinsson, P., & Tropp, J. (2020). Randomized numerical linear algebra: Foundations and algorithms. Acta Numerica, 29, 403-572. https://doi.org/10.48550/arXiv.2002.01387

OAK RIDGE National Laboratory


Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory Multicore Processor
core
core
core
core
network
memory
OpenMP OpenACC
Shared Memory Co-Processor
Foreign Language Interfaces .C .Call Rcpp OpenCL inline . . .
Cuda OpenCL
pthreads fork
network
memory
GPU
parallel = multicore + snow
node
node
node
node
network fabric
Sockets SPMD(MPI) MapReduce(Shuffle)
pbdMPI Rmpi
RHadoop SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Hands on Session rsvd: Randomized SVD via subspace embedding

Given an $n \times p$ matrix X and $k = r + 10$, where r is the *effective rank* of X:

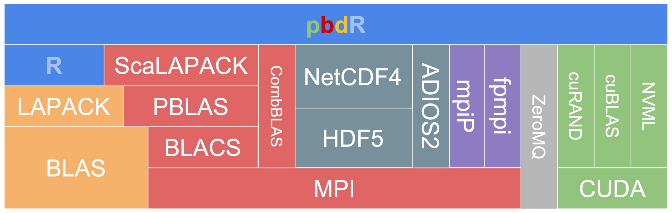
1. Construct a $p \times k$ random matrix Ω
2. Form $Y = X\Omega$
3. Decompose $Y = QR$

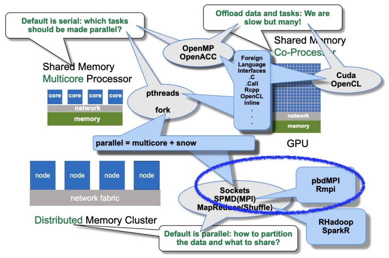
Q is an orthogonal basis for the columnspace of Y, which with high probability is the columnspace of X. To get the SVD of X:

1. Compute $C = Q^T X$
2. Decompose $C = \hat{U}\Sigma V^T$
3. Compute $U = Q\hat{U}$
4. Truncate factorization to r columns

mnist_rsvd.R

```r
source("mnist_read_mpi.R") # reads blocks of rows
suppressMessages(library(pbdDMAT))
suppressMessages(library(pbdML))
init.grid()

## construct block-cyclic ddmatrix
bldim = c(allreduce(nrow(my_train), op = "max"), ncol(my_train))
gdim = c(allreduce(nrow(my_train), op = "sum"), ncol(my_train))
dmat_train = new("ddmatrix", Data = my_train, dim = gdim,
                 ldim = dim(my_train), bldim = bldim, ICTXT = 2)
cyclic_train = as.blockcyclic(dmat_train)

comm.print(comm.size())
t1 = as.numeric(Sys.time())
rsvd_train = rsvd(cyclic_train, k = 10, q = 3, retu = FALSE, retv = FALS
t2 = as.numeric(Sys.time())
t1 = allreduce(t1, op = "min")
t2 = allreduce(t2, op = "max")
comm.cat("Time:", t2 - t1, "seconds\n")

comm.cat("rsvd top 10 singular values:", rsvd_train$d, "\n")

finalize()
```




Default is serial: which tasks should be made parallel?
Offload data and tasks: We are slow but many!
Shared Memory
Multicore Processor
Shared Memory Co-Processor
OpenMP OpenACC
pthreads
fork
Cuda OpenCL
GPU
parallel = multicore + snow
Sockets SPMD(MPI) MapReduce(Shuffle)
pbdMPI Rmpi
RHadoop SparkR
Distributed Memory Cluster
Default is parallel: how to partition the data and what to share?

Package kazaam

- Distributed methods for tall matrices (and some for wide matrices) that exploit the short dimension for speed and long dimension for parallelism
- Tall matrices, `shaq` class, are chunked by blocks of rows adn wide matrices, `tshaq` class, by blocks of columns
- Much like `pbdDMAT`, most matrix operations in R code are identical to serial through overloading operators and `shaq` `S4` class

Naming is a "tongue-in-cheek" play on 'Shaquille' 'ONeal' ('Shaq') and the film 'Kazaam'

Oak Ridge National Laboratory

Hands on Session `kazaam`

To be added