Adventures in Supercomputing with R

Lecture 11: Distributed, I/O, ScaLAPACK

George Ostrouchov

Oak Ridge National Laboratory and University of Tennessee

2022/02/20 (updated: 2022-04-27)

Exercise 8 discussion ...

Knit `RBigData/KPMS-IY4I-EX/exercise/Exercise08.Rmd` and read notes.

See code:

`RBigData/KPMS-IY4I-EX/mpi/mnist_svd_cv_mpi.R`

and

`RBigData/KPMS-IY4I-EX/mpi/mnist_svd_cv_mpi.sh`.

MPI Cooperation

Helper functions coordinate by knowing `comm.rank()` and `comm.size()`

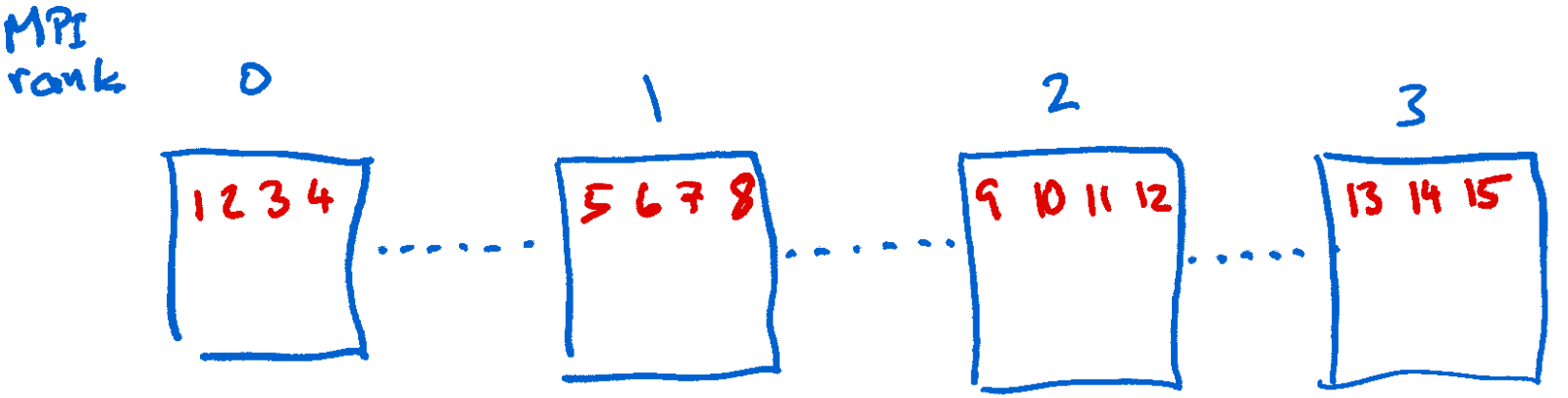
```
pbdIO::comm.chunk(15, form = "vector", all.rank = TRUE, p = 4)
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 5 6 7 8
##
## [[3]]
## [1]  9 10 11 12
##
## [[4]]
## [1] 13 14 15
```

* When running with `mpirun`, omit `p` parameter. Here it is used because my notes run serial R.
** Omitting `all.rank` delivers only the correct vector to each rank.
*** `pbdIO::` is omitted when `library(pbdIO)` was done earlier.

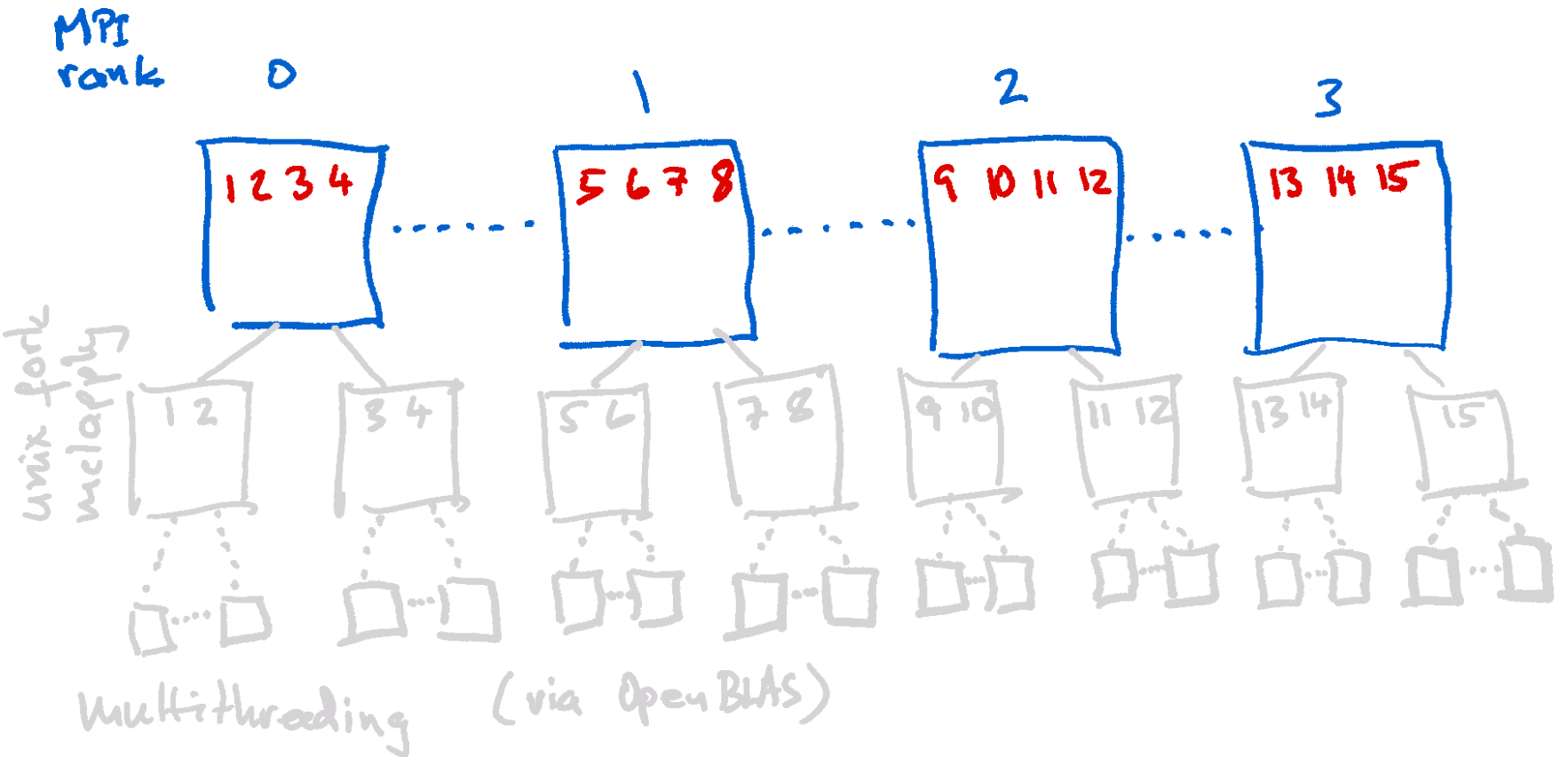
```
mpirun -np 4 Rscript your-code.R
```

```
comm.chunk(15, form = "vector")
```



```
mpirun -np 4 Rscript your-code.R`
```

```
comm.chunk(15, form = "vector")
```



Modes of parallel computing

- Shared memory: multithreading (C, C++, Fortran, etc.)
    - OpenMP Compiler directives: pragma
    - Memory efficient: One copy of memory and program
    - Available via libraries: OpenBLAS
    - Program your own via C, C++, Rcpp, etc.
- Shared memory: unix fork
    - Operating system mediated
    - Memory efficient: copy-on-write
- Distributed memory: MPI
    - Can use memory of multiple nodes
    - Replicates memory unless managed by distributing objects
    - Can operate in shared memory ("push MPI into nodes")

Parallel I/O

- A large object that fits on a node
  - Stay below 1/3 available memory
  - Gain speed by distributing
- Distributing data:
  - Each node (or rank) reads own data
  - Need parallel file system (Lustre or GPFS) for speedup
    - Karolina SCRATCH filesystem: Lustre
    - Accessible via the Infiniband network
    - https://docs.it4i.cz/barbora/storage/#understanding-the-lustre-filesystems
    - Number of file stripes

Parallel I/O with skinny matrix methods

- `rhdf5` and `kazaam` packages
- Reader `KPMS-IT4I-EX/mpi/mnist_read_mpi.R`
- SVD example `KPMS-IT4I-EX/mpi/mnist_kazaam.R`

mnist_read_mpi.R

```
suppressMessages(library(rhdf5))
suppressMessages(library(pbdIO))

filename = "/scratch/project/dd-21-42/data/mnist/train.hdf5"
dataset1  = "image"
dataset2  = "label"

## get and broadcast dimensions to all processors
if (comm.rank() == 0) {
   h5f = H5Fopen(filename, flags="H5F_ACC_RDONLY")
   h5d = H5Dopen(h5f, dataset1)
   h5s = H5Dget_space(h5d)
   dims = H5Sget_simple_extent_dims(h5s)$size
   H5Dclose(h5d)
   H5Fclose(h5f)
} else dims = NA
dims = bcast(dims)

## get my local indices for contiguous data read
nlast = dims[length(dims)] # last dim moves slowest
my_ind = comm.chunk(nlast, form = "vector")

## parallel read of local data
my_train = as.double(h5read(filename, dataset1,
                            index = list(NULL, NULL, my_ind)))
my_train_lab = as.character(h5read(filename, dataset2,
                                   index = list(my_ind)))
H5close()
```

mnist_kazaam.R

```
suppressMessages(library(kazaam))
source("mnist_read_mpi.R") # reads blocks of rows

## create shaq class distributed matrix from local pieces
sq_train = shaq(my_train)

## svd (shaq class: tall-skinny matrix)
train_svd = svd(sq_train, nu = 0, nv = 5)
comm.cat("kazaam top(5) singular values:", train_svd$d[1:5], "\n")

cp_train = allreduce(crossprod(my_train))
train_svd2 = eigen(cp_train)
comm.cat("direct top(5) singular values:", sqrt(train_svd2$values[1:5

finalize()
```

ScaLAPACK Processor Grid and Block Cyclic Layout

RBigData/pbdDMAT package on GitHub

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

(a) 1×6 (b) 2×3 (c) 3×2 (d) 6×1

Table: Processor Grid Shapes with 6 Processors

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} & x_{28} & x_{29} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} & x_{38} & x_{39} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} & x_{48} & x_{49} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} & x_{58} & x_{59} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} & x_{68} & x_{69} \\ x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} & x_{78} & x_{79} \\ x_{81} & x_{82} & x_{83} & x_{84} & x_{85} & x_{86} & x_{87} & x_{88} & x_{89} \\ x_{91} & x_{92} & x_{93} & x_{94} & x_{95} & x_{96} & x_{97} & x_{98} & x_{99} \end{bmatrix}_{9\times 9}$$

$$\begin{bmatrix} x_{11} & x_{12} & x_{17} & x_{18} \\ x_{21} & x_{22} & x_{27} & x_{28} \\ x_{51} & x_{52} & x_{57} & x_{58} \\ x_{61} & x_{62} & x_{67} & x_{68} \\ x_{91} & x_{92} & x_{97} & x_{98} \end{bmatrix}_{5\times 4} \begin{bmatrix} x_{13} & x_{14} & x_{19} \\ x_{23} & x_{24} & x_{29} \\ x_{53} & x_{54} & x_{59} \\ x_{63} & x_{64} & x_{69} \\ x_{93} & x_{94} & x_{99} \end{bmatrix}_{5\times 3} \begin{bmatrix} x_{15} & x_{16} \\ x_{25} & x_{26} \\ x_{55} & x_{56} \\ x_{65} & x_{66} \\ x_{95} & x_{96} \end{bmatrix}_{5\times 2}$$

$$\begin{bmatrix} x_{31} & x_{32} & x_{37} & x_{38} \\ x_{41} & x_{42} & x_{47} & x_{48} \\ x_{71} & x_{72} & x_{77} & x_{78} \\ x_{81} & x_{82} & x_{87} & x_{88} \end{bmatrix}_{4\times 4} \begin{bmatrix} x_{33} & x_{34} & x_{39} \\ x_{43} & x_{44} & x_{49} \\ x_{73} & x_{74} & x_{79} \\ x_{83} & x_{84} & x_{89} \end{bmatrix}_{4\times 3} \begin{bmatrix} x_{35} & x_{36} \\ x_{45} & x_{46} \\ x_{75} & x_{76} \\ x_{85} & x_{86} \end{bmatrix}_{4\times 2}$$

```
> x <- as.rowblock( x )
> x <- as.blockcyclic( x )
> x <- redistribute( x, bldim=c(8, 8), ICTXT = 0 )
```

ScaLAPACK Complex Communication

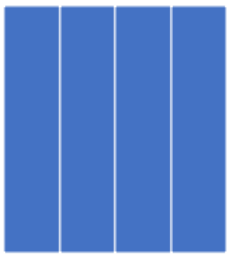
- Several communicators (global, row wise, column wise)

Converters for Bolck-Cyclic Layout

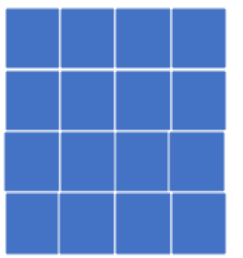
```
as.matrix(X)
```

```
as.rowblock(X)
```

```
as.colblock(X)
```



```
as.blockcyclic(X)
```



ddmatrix Examples

`KPMS-IT4I-EX/dmat`