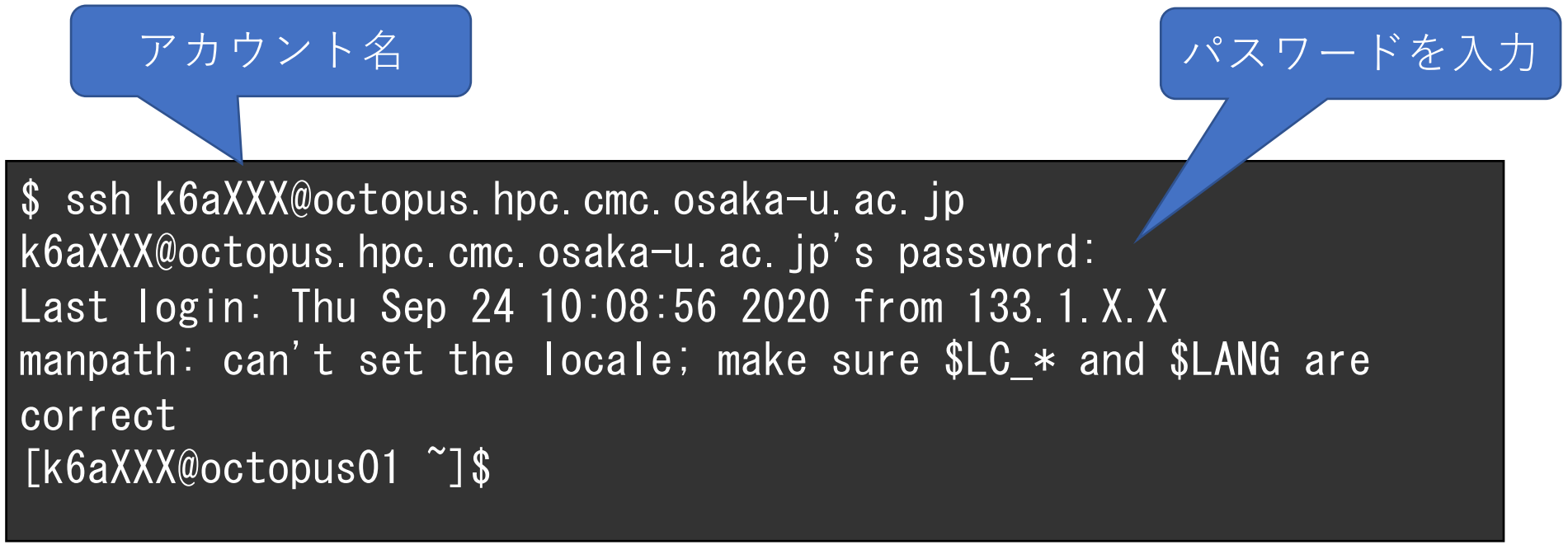
Rハンズオンセミナー
OCTOPUSでRを使おう

大阪大学サイバーメディアセンター
木戸　善之
mailto:kido@cmc.osaka-u.ac.jp
2020/10/02

セミナーを始める前に

- ターミナルソフトを開いて，ログインしてみてください



.R/Makevarsの準備

- ホームディレクトリに.Rディレクトリを作成
- そのあと，以下のサンプルをもとにMakevarsを作成

```
[k6aXXX@octopus01 ~]$ mkdir .R
[k6aXXX@octopus01 ~]$ vi .R/Makevars
```

```
CC=gcc
CXX=g++
CFLAGS=-O3 -fopenmp -std=c11 -L/usr/mpi/gcc/openmpi/lib64
CXXFLAGS=-O3 -fopenmp -L/usr/mpi/gcc/openmpi/lib64
```

Rをビルドしてみよう

```
[k6aXXX@octopus01 ~]$ BASE=/octfs/apl/HPC-X/hpcx-v2.2.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64
[k6aXXX@octopus01 ~]$ source ${BASE}/hpcx-init.sh
[k6aXXX@octopus01 ~]$ hpcx_load
[k6aXXX@octopus01 ~]$ export CC=mpicc
[k6aXXX@octopus01 ~]$ export CXX=mpic++
[k6aXXX@octopus01 ~]$ export FC=mpif90
[k6aXXX@octopus01 ~]$ export LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi/lib64:$LD_LIBRARY_PATH
[k6aXXX@octopus01 ~]$ tar zxvf R-4.0.2.tar.gz
[k6aXXX@octopus01 ~]$ cd R-4.0.2
[k6aXXX@octopus01 ~]$ ./configure
…
[k6aXXX@octopus01 ~]$ make
…
```

日本のスパコン

名称・愛称	設置者	メーカ	性能	Top 500ランク
富岳	理研	Fujitsu	415.5 PFlops	1 New
AI Bridging Cloud Infrastructure (ABCI)	産総研	Fujitsu	19.9 PFlops	12
Oakforest-PACS	東大・筑波大	Fujitsu	13. 6 PFlops	19
TSUBAME3.0	東工大	HPE	8.1 PFlops	28
-	気象庁	Cray	5.7 PFlops	42, 43

参考性能

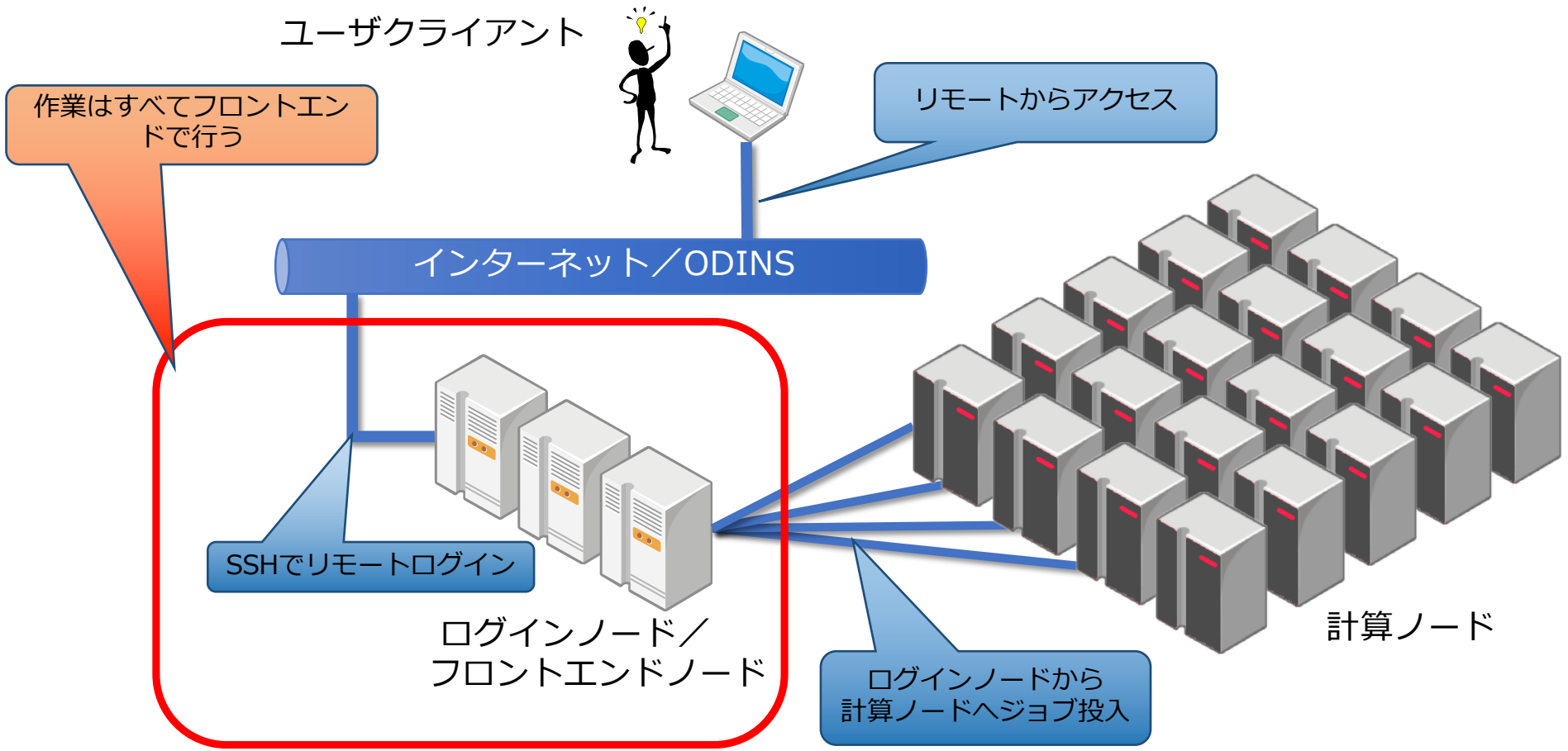
OCTOPUS	阪大	NEC	理論性能 1.4 Pflops	だいたい500位と同じくらいの性能

OCTOPUSの性能

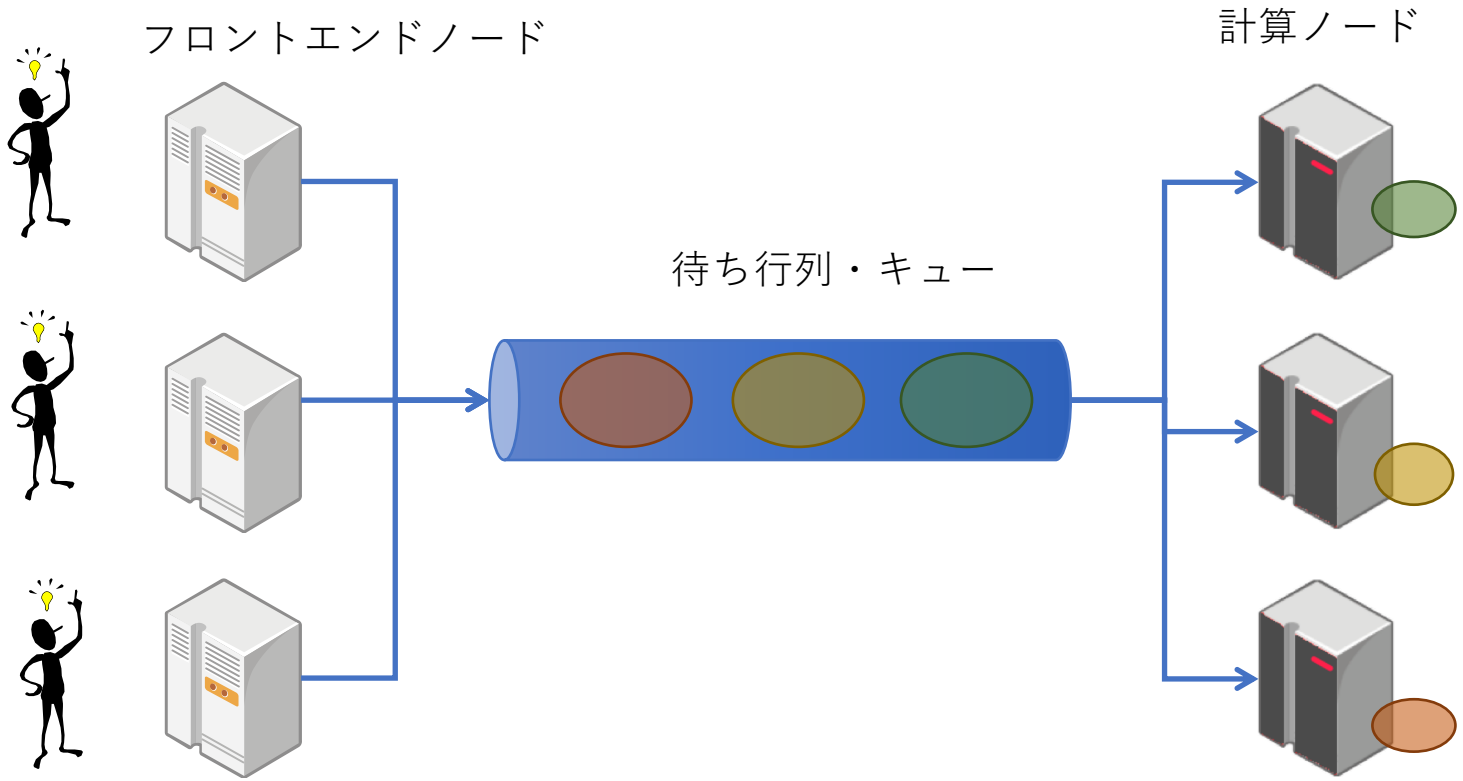
総演算性能	1.463 PFLOPS	
ノード構成	**汎用CPUノード 236ノード**	**Intel Xeon Gold 6126 (12コア) × 2基 メモリ：192 GB**
	GPUノード 37ノード	Intel Xeon Gold 6126 (12コア) × 2基 メモリ：192 GB GPU：NVIDIA Tesla P100×4基
	Xeon Phiノード 44ノード	Intel Xeon Phi 7210 (64コア) × 1基 メモリ：192 GB
	大容量主記憶搭載ノード 2ノード	Intel Xeon Platinum 8153 (16コア) × 1基 メモリ：6 TB

http://www.hpc.cmc.osaka-u.ac.jp/octopus/

供用コンピュータ・クラスタ



ジョブ投入
バッチキューシステム



qsubでジョブ投入

今回のセミナー用環境変数

計算機環境の指定

ジョブキューに登録

```
#!/bin/csh
#PBS -q LECTURE
#PBS -y 336
#PBS -l elapstim_req=1:00:00,memsz_job=60GB
#PBS -v F_RSVTASK=4
setenv F_PROGINF DETAIL
cd $PBS_O_WORKDIR
./a.out
```

```
$ qsub a_batch.sh
Request 88156.cmc submitted to queue: ACE.
$
```

R関連のファイル／ディレクトリの説明

- 実行ファイル：R-4.0.2/bin/Rscript
  - バッチジョブ形式で投入するので，インタラクティブではなくRスクリプトを書いて，ジョブを投入するから
- ユーザライブラリディレクトリ：R-4.0.2/library
  - 様々なパッケージ（今回だとrbenchmark, ff, bigmemory, Rmpiなど）をインストールするディレクトリ
  - 環境変数での設定は R_LIBS=/octfs/home/k6aXXX/R-4.0.2/library
- パッケージをMakeする時の環境設定：.R/Makevars
  - パッケージをインストールする際，C言語やC++で書かれたファイルもダウンロードし，オンデマンドでMakeする．だからこのファイルが必要

OCTOPUS固有のディレクトリの説明

- ホームディレクトリ：/octfs/home/k6aXXX
  - フロントエンド，計算ノードからアクセス可能.
- ワークディレクトリ：/octfs/work/kosyuXXX
  - フロントエンド，計算ノードからアクセス可能．計算ノードからのファイルアクセスはこちらを推奨

セミナー用ファイル

```
/octfs/apl/kosyu/R
+- data
|  +- airlinemid.csv : 航空データ (3.9GB)
|  +- airlinenew.csv : オリジナル航空データ (68GB)
+- for_R_build.txt : Rビルド用のメモファイル
+- Makevars : パッケージビルド用環境変数ファイル
+- R-4.0.2.tar.gz : Rソースファイル
|
|
+- R4_FF
|  +- airlineBM.R : bigmemory用サンプルスクリプト
|  +- airlineFF.R : ff用サンプルスクリプト
|  +- airline.R : 通常読み込み用サンプルスクリプト
|  +- r4BM.sh : bigmemory用ジョブスクリプト
|  +- r4FF.sh : ff用ジョブスクリプト
|  +- r4NO.R : 通常読み込み用ジョブスクリプト
+- R4_Rmpi
   +- hello.R : Rmpiを用いたHello Worldスクリプト
   +- r4hello.sh : Hello World用ジョブスクリプト
```

Rをフロントエンドで動かしてパッケージをインストールしてみよう

```
[k6aXXX@octopus01 ~]$ R-4.0.2/bin/R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

...

> install.packages("rbenchmark", repos="http://cran.ism.ac.jp")
> install.packages("ff", repos="http://cran.ism.ac.jp")
...
> install.packages("Rcpp", repos="http://cran.ism.ac.jp")
> install.packages("bigmemory", repos="http://cran.ism.ac.jp")
...
```

Rmpiをインストールしてみよう

- まずMakevarsを編集

```
CC=mpicc
CXX=mpic++
FC=mpif90
CFLAGS=-O3 -fopenmp -std=c99 -L/usr/mpi/gcc/openmpi/lib64
CXXFLAGS=-O3 -fopenmp -std=c99 -L/usr/mpi/gcc/openmpi/lib64
```

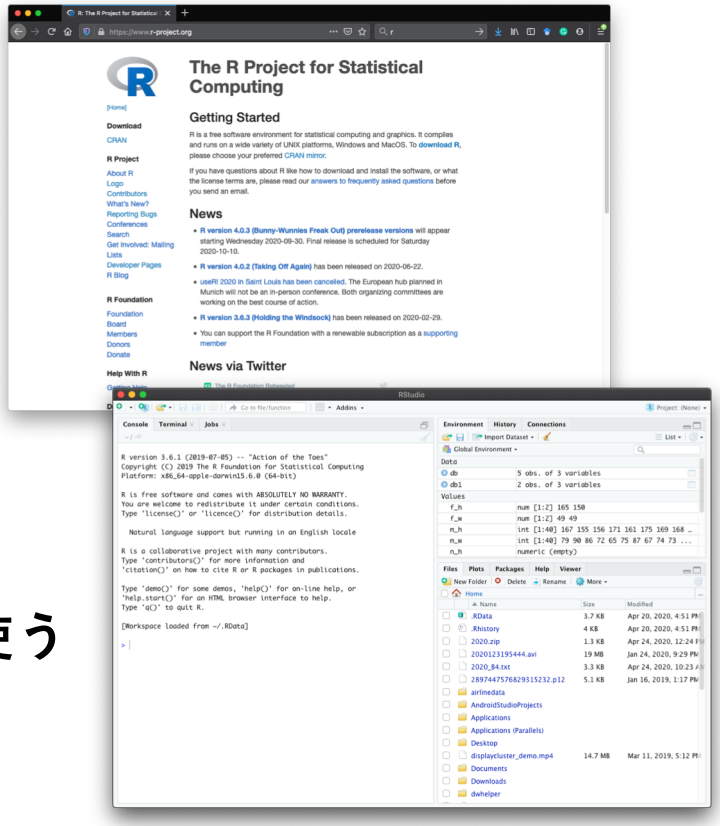
- その後，Rでインストール

```
[k6aXXX@octopus01 ~]$ R-4.0.2/bin/R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

...

> install.packages("Rmpi", repos="http://cran.ism.ac.jp", configure.args = paste("--with-mpi=/usr/mpi/gcc/openmpi
--with-Rmpi-type=OPENMPI"))
```

Rの特徴（え？このタイミング？）

- 統計のソフトウェアスイート
- 対話環境，IDEなどがあって使いやすい
- スクリプト実行もできちゃう
- フリーソフト（Sとは違う）

- 欠点
  - **オブジェクトは値渡しなのでメモリを大量に使う**
  - **並列環境が苦手（できなくはない**

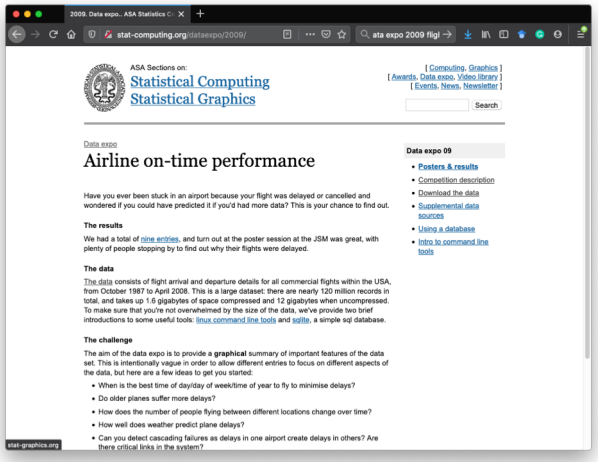


大容量ファイルを扱うパッケージ

- ff : ディスク上の大容量データのメモリ効率の高い格納と高速アクセス機能
  - 大容量のファイルを開いた場合，Rのメモリ領域に展開せずに，カラム数に応じた一時ファイルを作る
  - メインメモリ以上のデータを扱うことができる
  - データのアクセスごとに一時ファイルを参照しに行く（ただし参照方法よってはオンメモリにする場合もある
- bigmemory : 共有メモリとマップファイルを用いた大規模マトリックス管理パッケージ
  - 大容量のファイルを開いた場合，backing fileという一時ファイルを作る
  - あとdescriptor fileというマップファイルを作る
  - メインメモリ以上のデータを扱うことができる
  - ただファイルの読み込みはread.table()と大して変わらない

今回使うデータ

- フリーで利用できる定番，航空データを利用
  - 1987年-2015年の全米のフライトデータ
  - http://stat-computing.org/dataexpo/2009/
  - 全データで68GB
  - これを使おうと思ったんだけど，セミナーの時間内に終わらないので，短くしたものを用意しました．
    - 全データから10,000,000行抜き出したデータ3.9GB



普通に読み込んでみる

airline.R

```
library(rbenchmark)
start_time <- Sys.time()
air <- read.table("/octfs/work/kosyuXXX/airline/airlinemid.csv", header=T, sep=",")
Sys.time() - start_time

test.fx <- function() {
  print(mean(air$Distance))
}
rbenchmark::benchmark(test.fx())
end_time <- Sys.time()
end_time - start_time
```

ジョブスクリプト

r4NO.sh

```
#!/bin/bash
#PBS -q LECTURE
#PBS -y 336
#PBS -l elapstim_req=0:10:00,cpunum_job=24
#PBS -M kido@cmc.osaka-u.ac.jp
#PBS -m b
#PBS -b 1
#PBS -v R_LIBS=/octfs/home/$USER/R-4.0.2/library
cd $PBS_O_WORKDIR
/octfs/home/$USER/R-4.0.2/bin/Rscript /octfs/home/$USER/R4_FF/airline.R
```

```
[w6XXXX@octopus01 ~]$ qsub r4NO.sh
Request 878315.oct submitted to queue: DBG.
[w6XXXX@octopus01 ~]$ qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
878315.oct      r4NO.sh  w6XXXX   DBG-C       0 QUE -    0.00B     0.00        0 Y Y Y    1
```

bigmemoryを使ってみる

airlineBM.R

```
library(bigmemory)
library(rbenchmark)
start_time <- Sys.time()
air <- read.big.matrix("/octfs/work/kosyuXXX/airline/airlinemid.csv", header=T,
backingfile="air.bm", type="integer", descriptorfile="air.bm.desc",
backingpath="/octfs/work/kosyuXXX/airline/")
Sys.time() - start_time



test.fx <- function() {
  print(mean(air[,"Distance"]))
}
rbenchmark::benchmark(test.fx())
end_time <- Sys.time()
end_time - start_time
```

ffを使ってみる

airlineFF.R

```
library(ff)

library(rbenchmark)

start_time <- Sys.time()

air <- read.csv.ffdf(file="/octfs/work/kosyuXXX/airline/airlinemid.csv", header=TRUE,
colClasses=c(Year="integer",Quarter="integer",Month="integer",DayofMonth="integer",DayOfWeek="integer",FlightDate="factor",UniqueCarrier="factor",AirlineID="factor",Carrier="factor",TailNum="factor",FlightNum="factor",OriginAirportID="factor",OrjginAirportSeqID="factor",OriginCityMarketID="factor",Origin="factor",OriginCityName="factor",OriginState="factor",OriginStateFips="factor",OriginStateName="factor",OriginWac="factor",DestAirportID="factor",DestAirportSeqID="factor",DestCityMarketID="factor",Dest="factor",DestCityName="factor",DestState="factor",DestStateFips="factor",DestStateName="factor",DestWac="factor",CRSDepTime="factor",DepTime="factor",DepDelay="factor",DepDelayMinutes="factor",DepDel15="factor",DepartureDelayGroups="factor",DepTimeBlk="factor",TaxiOut="factor",WheelsOff="factor",WheelsOn="factor",TaxiIn="factor",CRSArrTime="factor",ArrTime="factor",ArrDelay="factor",ArrDelayMinutes="factor",ArrDel15="factor",ArrivalDelayGroups="factor",ArrTimeBlk="factor",Cancelled="factor",CancellationCode="factor",Diverted="factor",CRSElapsedTime="factor",ActualElapsedTime="factor",AirTime="factor",Flights="factor",Distance="double",DistanceGroup="factor",CarrierDelay="factor",WeatherDelay="factor",NASDelay="factor",SecurityDelay="factor",LateAircraftDelay="factor",FirstDepTime="factor",TotalAddGTime="factor",LongestAddGTime="factor",DivAirportLandings="factor",DivReachedDest="factor",DivActualElapsedTime="factor",DivArrDelay="factor",DivDistance="factor",Div1Airport="factor",Div1AirportID="factor",Div1AirportSeqID="factor",Div1WheelsOn="factor",Div1TotalGTime="factor",Div1LongestGTime="factor",Div1WheelsOff="factor",Div1TailNum="factor",Div2Airport="factor",Div2AirportID="factor",Div2AirportSeqID="factor",Div2WheelsOn="factor",Div2TotalGTime="factor",Div2LongestGTime="factor",Div2WheelsOff="factor",Div2TailNum="factor",Div3Airport="factor",Div3AirportID="factor",Div3AirportSeqID="factor",Div3WheelsOn="factor",Div3TotalGTime="factor",Div3LongestGTime="factor",Div3WheelsOff="factor",Div3TailNum="factor",Div4Airport="factor",Div4AirportID="factor",Div4AirportSeqID="factor",Div4WheelsOn="factor",Div4TotalGTime="factor",Div4LongestGTime="factor",Div4WheelsOff="factor",Div4TailNum="factor",Div5Airport="factor",Div5AirportID="factor",Div5AirportSeqID="factor",Div5WheelsOn="factor",Div5TotalGTime="factor",Div5LongestGTime="factor",Div5WheelsOff="factor",Div5TailNum="factor"))

Sys.time() - start_time


test.fx <- function() {

  print(mean(air[,"Distance"]))

}

rbenchmark::benchmark(test.fx())

end_time <- Sys.time()

end_time - start_time
```

比較

- read.table
  - ファイルオープン： 3.054893 mins
  - データアクセス（１列の読み）：1.922 mins
- bigmemory
  - ファイルオープン： 7.149671 mins
  - データアクセス（１列の読み）： 1.741 mins
- ff
  - ファイルオープン： 4.171675 mins
  - データアクセス（１列の読み）：6.906 mins

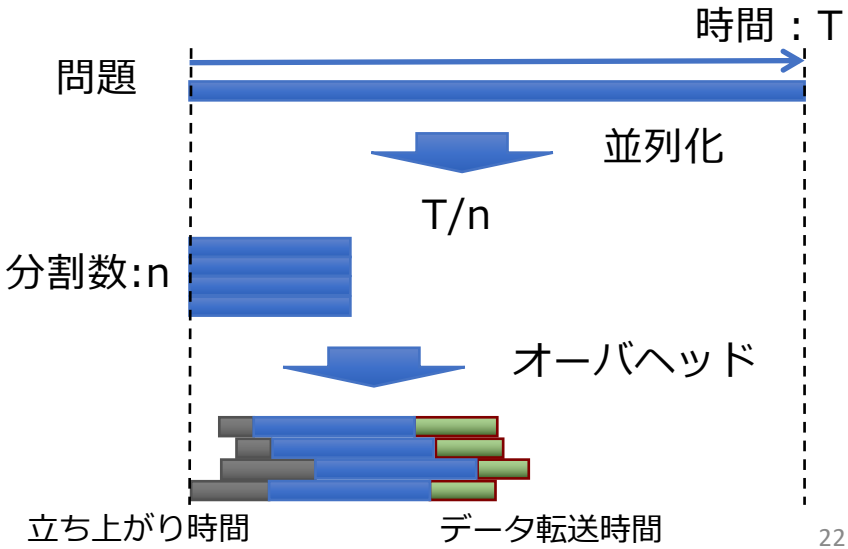
- airlinemid.csv
- 3.9 GB
- 行：10,000,000

一桁GBであればOCTOPUSでは普通に使える
より大規模になるとffがいいかも

並列プログラミングとは

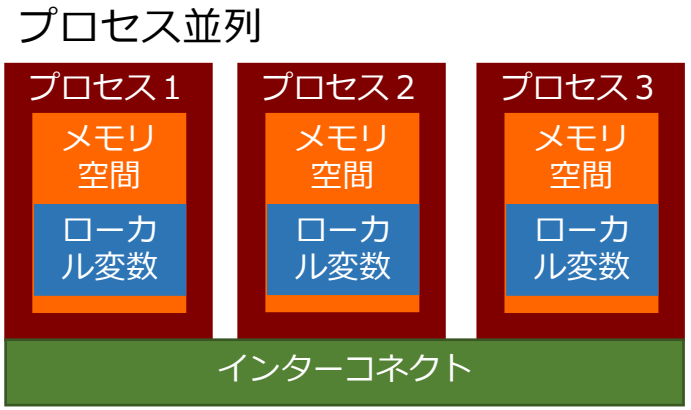
- 逐次処理の問題、プログラム（実行時間：T）をnに分割し、 n 台のプロセッサ（or 計算機）でT/n時間にする
- プロセッサに割り当てられるタスクは独立
- 並列化できる問題はデータに依存性のない問題のみに限られる

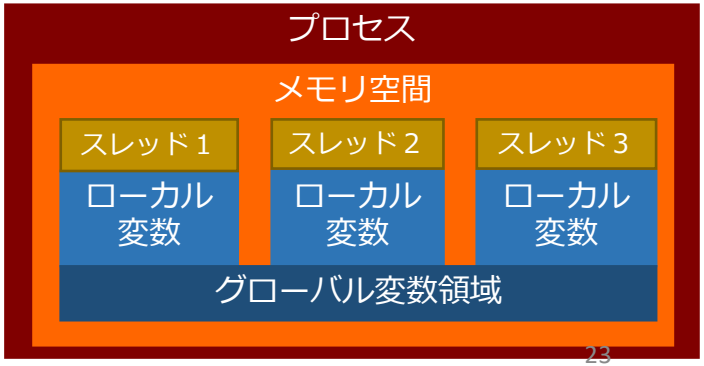
- 通信によるオーバヘッド
  - 立ち上がり時間
  - データ転送時間



プロセス並列とスレッド並列

- プロセス並列
  - メモリ空間は独立
  - 並列タスク間でのデータ通信は必要に応じて
  - 例：Message Passing Interface (MPI)
  - RだとRmpi
- スレッド並列
  - メモリ空間を共有
  - データ通信の必要性なし
  - 例：OpenMP





Rで並列処理 in OCTOPUS

hello.R

```
library(Rmpi)

id <- mpi.comm.rank(comm = 0)
np <- mpi.comm.size(comm = 0)
hostname <- mpi.get.processor.name()

msg <- sprintf("Hello world from process %03d of %03d, on host %s¥n",id, np, hostname)
cat(msg)

mpi.barrier(comm = 0)
mpi.finalize()
```

MPIを使う場合のジョブスクリプト

r4mpi.sh

```
#!/bin/bash
#PBS -q LECTURE
#PBS -y 336
#PBS -l elapstim_req=0:10:00,cpunum_job=24
#PBS -M kido@cmc.osaka-u.ac.jp
#PBS -m b
#PBS -b 2
#PBS -T openmpi
#PBS -v R_LIBS="/octfs/home/k6aXXX/R-4.0.2/library"
#PBS -v LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/mpi/gcc/openmpi/lib64"
BASE=/octfs/apl/HPC-X/hpcx-v2.2.0-gcc-MLNX_OFED_LINUX-4.2-1.2.0.0-redhat7.3-x86_64
source $BASE/hpcx-init.sh
hpcx_load
cd $PBS_O_WORKDIR

mpirun $NQSII_MPIOPTS -np 48 /octfs/home/k6aXXX/R-4.0.2/bin/Rscript /octfs/home/k6aXXX/R4_Rmpi/hello.R
```

まとめ

- OCTOPUSでRは使える
- 大規模データもあつかえる
  - ff, bigmemoryパッケージを使えばより大規模なデータを使える

- 並列処理
  - 今の所バルクジョブは可能