第9回 AICS公開ソフト講習会
K MapReduce ハンズオン

丸山 直也
松田 元彦
滝澤 真一朗

理化学研究所 計算科学研究機構
プログラム構成モデル研究チーム

K computer

目的

- KMRの導入方法を学ぶ
- KMRRUNを用いたMapReduceプログラム実行方法を学ぶ

K computer

ハンズオン環境

- Fujitsu FX10@RIST（高度情報科学技術研究機構）
- 計算資源
  - 12ノード
  - 16Core, 32GB Memory/ノード
  - ジョブ最大実行時間：1時間
- ストレージ資源
  - /home
    - 100GB/ユーザ
    - ステージングなし
  - /data/acource/141111_kmr_handson
    - 本日のプログラム・データ置き場
    - ステージングなし
- ログインノード

```
kuslogin01.kobe.rist.or.jp
```


K computer

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

K computer

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

K computer

KMRのインストール (1/2)

- KMR-1.3.2 Release (2014-11-07) をインストール
- 展開

```
$ cd ~
$ tar zxf \(改行しない)
  /data/acourse/141111_kmr_handson/kmr-1.3.2.tar.gz
$ cd kmr-1.3.2
```

- configure
  - インストールパスを指定

```
$ ./configure --prefix=$HOME/lib/kmr-1.3.2
```

- make & install
  - $HOME/lib/kmr-1.3.2にインストールされる

```
$ make
$ make install
```


K computer

KMRのインストール (2/2)

- 環境変数を設定

```
$ export PATH=~/lib/kmr-1.3.2/bin:$PATH
$ export MANPATH=~/lib/kmr-1.3.2/man:$MANPATH
```

  – シェルの設定ファイル(~/.bashrcなど)に記述しておくと良い

K computer

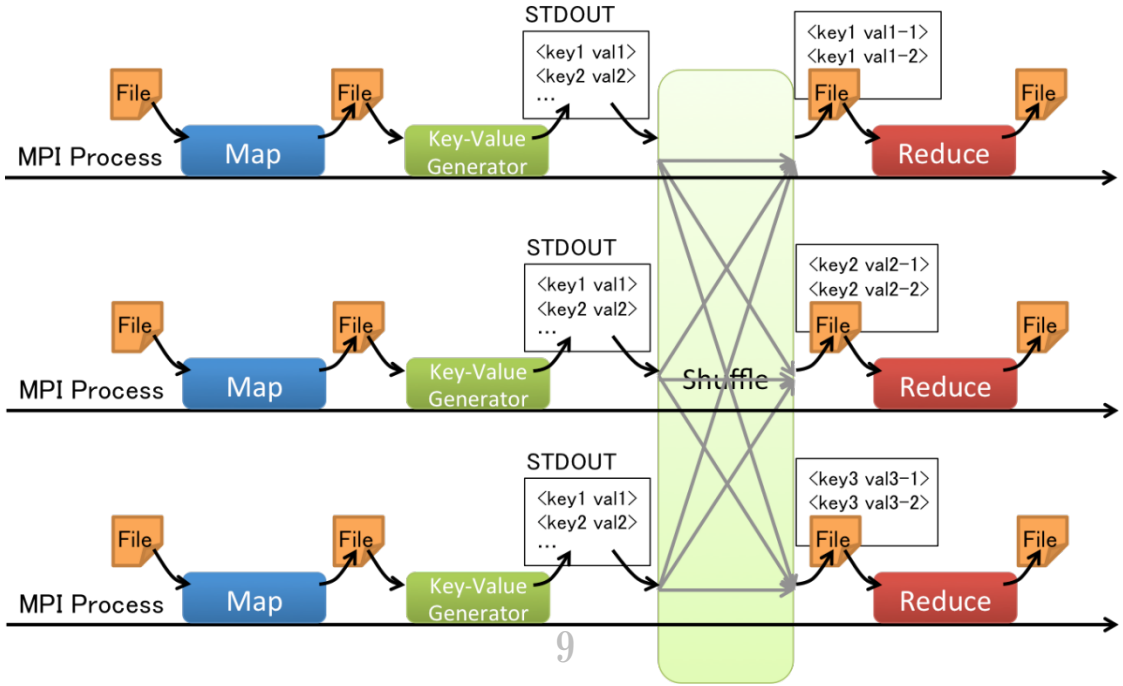
Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

K computer

KMRRUN

- MapReduceワークフローを実行
- Mapper/Reducerとして、MPIプログラム、任意の言語で実装された逐次プログラム(ノード内並列対応)を実行可能
  - Mapperの出力からKVを生成し、標準出力に書き出す Key-Value Generator プログラムも必要



K computer

Mapper/Reducer/KV Generator仕様

講義資料
再掲

	Mapper	KV Generator	Reducer
実装言語	任意	任意	任意
並列実行	MPI/OpenMP	OpenMP	MPI/OpenMP
入力	• ファイル読み込み • ファイル名は最後の引数として渡される	• ファイル読み込み • Mapperの入力ファイル名が最後の引数として渡される • Mapperの出力を読み込む場合は、ファイル名を推測して作成	• ファイル読み込み • ファイル名は最後の引数として渡される • 1 KV/行フォーマット • KeyとValueはスペース1つで区切られる
出力	• ファイル書き出し • ファイル名は、入力ファイル名から類推できる名前とする	• 標準出力に出力 • 1 KV/行フォーマット • KeyとValueはスペース1つで区切られる	• ファイル書き出し

K computer

PI計算サンプルプログラム

- PI計算のサンプルプログラムを2種類用意
  - 逐次プログラム版：KMR_SRC/kmrrun/
    - Mapper：pi.mapper.c
    - KV Generator：pi.kvgen.sh
    - Reducer：pi.reducer.c
  - MPIプログラム版：KMR_SRC/kmrrun/
    - Mapper：mpi_pi.mapper.c
    - KV Generator：mpi_pi.kvgen.sh
    - Reducer：mpi_pi.reducer.c
- コンパイル方法

```
$ cd ~/kmr-1.3.2/kmrrun
$ make pi.mapper pi.reducer
$ make mpi_pi.mapper mpi_pi.reducer
```


K computer

PI計算の実装 (1/3)

- Mapper
  - 入力
    - プロットする点の数が書かれた入力ファイル
  - 処理
    - 指定された数分の点をランダムに生成し、半径1の円に入る点の数を数え上げる
  - 出力
    - 「半径1に入る点の数/点の総数」を「入力ファイル名.out」ファイルに書き出す

実行イメージ

```
$ cat ./input/000
10000
$ ./pi.mapper ./input/000
$ ls ./input
000    000.out
$ cat ./input/000.out
7829/10000
```


K computer

PI計算の実装 (2/3)

- KV Generator
  - 入力
    - Mapperの入力ファイル
  - 処理
    - Mapperの出力ファイルパスを推測して作成
    - Mapperの出力ファイルを読み込む
    - Mapperの出力ファイルを削除
  - 出力
    - Key-Value Pair <0, ファイルコンテンツ> を標準出力に出力
      - KMRRUN実行時には、Shuffle後、ファイル名「0」ファイルに全てのKVが保存される
      - Keyが1種類なので、Reducerは1ノードでしか実行されない

実行イメージ

```
$ ls ./work
000    000.out
$ ./pi.kvgen.sh ./work/000
0 7829/10000
$ ls ./work
000
```


K computer

PI計算の実装 (3/3)

- Reducer
  - 入力
    - 1行に1KVが書かれた入力ファイル
  - 処理
    - 点の数よりPIを計算
  - 出力
    - 計算結果を「入力ファイル名.out」ファイルに書き出す

実行イメージ

```
$ ls ./
0                    pi.mapper
pi.keygen.sh         pi.reducer
$ cat 0
0 7829/10000
0 7830/10000
$ ./pi.reducer ./0
3.131800
$ ls ./
0                    pi.mapper
0.out                pi.reducer
pi.keygen.sh
$ cat 0.out
3.131800
```


K computer

KMRRUNコマンド

講義資料
再掲

- 実行コマンド

```
$ mpirun MPIOPT ./kmrrun -n procs -m mapper ¥
-k kvgen -r reducer ./input
```

- コマンドの意味（kmrrunのmanpageも参考）

kmrrun	KMRRUNプログラム本体。 KMR_INST/lib/kmrrunにインストールされている。
-n procs	1回のMapper/Reducer実行で使用するプロセス数を指定。 「m_procs:r_procs」フォーマットで指定すれば、Mapper/Reducerで異なるプロセス数で実行可能。デフォルトは1。 [省略可能]
-m mapper	Mapperプログラム
-k kvgen	KV Generatorプログラム [省略可能]
-r reducer	Reducerプログラム [省略可能]
./input	Mapperの入力ファイル、またはディレクトリ。 全MPIプロセスからアクセスできる、共有ディレクトリ上におくこと。

逐次版PI計算の実行 (1/3)

1. 入力ファイル群を作業ディレクトリにコピー

```
$ mkdir ~/work
$ cp -r /data/acourse/141111_kmr_handson/samples/pi ~/work
```

2. プログラム群を作業ディレクトリにコピー

```
$ cd ~/work/pi
$ cp ~/lib/kmr-1.3.2/lib/kmrrun .
$ cp ~/kmr-1.3.2/kmrrun/pi.mapper .
$ cp ~/kmr-1.3.2/kmrrun/pi.kvgen.sh .
$ cp ~/kmr-1.3.2/kmrrun/pi.reducer .
```

3. ジョブスクリプト作成プログラムを実行

```
$ cd ~/work/pi
$ kmrrungenscript.py -t 00:10:00 -e 4 -s 1 -p 8 \
-d ./input -n 1 -m ./pi.mapper -k ./pi.kvgen.sh \
-r ./pi.reducer -w job.sh
```


K computer

KMRRUNヘルパープログラム

- kmrrungenscript.py
  - 京コンピュータ用のジョブスクリプトを生成
    - ハンズオンでの利用には一部を修正（次スライド）
  - KMRインストールディレクトリ下のbin/に存在
  - 使い方の詳細はコマンドのManpageを参照
- 先の実行例

実行時間10分　4ノード使用　8プロセス / ノード

```
$ cd ~/work/pi
$ kmrrungenscript.py -t 00:10:00 -e 4 -s 1 -p 8 \
-d ./input -n 1 -m ./pi.mapper -k ./pi.kvgen.sh \
-r ./pi.reducer -w job.sh
```

Mapper/Reducerは
1ノードで実行

K computer

逐次版PI計算の実行 (2/3)

4. ジョブスクリプトを修正

```
$ vi job.sh
```

ジョブスクリプト修正例

```
#!/bin/bash -x
#
#PJM --rsc-list "elapse=00:10:00"
#PJM --rsc-list "node=4"
#PJM --mpi "shape=1"
#PJM --mpi "proc=8"
#PJM --rsc-list "proc-core=unlimited"
#PJM --stg-transfiles "all"
#PJM --stgin "/volume3/home/acourse/aics01/lib/kmr-1.3.2/lib/kmrrun ./kmrrun"
#PJM --stgin "./pi.mapper ./pi.mapper"
#PJM --stgin "./pi.kvgen.sh ./pi.kvgen.sh"
#PJM --stgin "./pi.reducer ./pi.reducer"
#PJM --stgin "./input/* ./input/"
#PJM --stgout "./output ./"
#PJM --stgout "./core* ./"
#PJM -S

. /work/system/Env_base_K-1.2.0-15

mpirun -np 8 ./kmrrun -n 1 -m "./pi.mapper" -k "./pi.kvgen.sh" -r "./pi.reducer" ./input
```

FX10@RISTではステージング不要なため削除
逆に京の場合、kmrrunは作業ディレクトリに自動でコピーされる

FX10@RISTの最新の言語環境では不具合が出ているため、京互換バージョンを使用

K computer

逐次版PI計算の実行 (3/3)

5. ジョブ実行

```
$ pjsub job.sh
```

– pjstat コマンドでジョブ実行状況を確認

6. 出力を確認

```
$ ls
0.out    job.sh.e13203  job.sh.o13203
job.sh   job.sh.i13203
```

0.out	最終出力ファイル
job.sh.i13203	ジョブ統計情報
job.sh.o13203	標準出力
job.sh.e13203	標準エラー出力

K computer

MPI版PI計算の実行

1. プログラムを作業ディレクトリにコピー

```
$ cd ~/work/pi
$ cp ~/kmr-1.3.2/kmrrun/mpi_pi.mapper .
$ cp ~/kmr-1.3.2/kmrrun/mpi_pi.kvgen.sh .
$ cp ~/kmr-1.3.2/kmrrun/mpi_pi.reducer .
```

- ジョブスクリプト作成プログラムを実行
  - ステージング指示文を削除、言語環境を修正

```
$ cd ~/work/pi
$ kmrrungenscript.py -t 00:10:00 -e 4 -s 1 -p 8 \
-d ./input -n 2 -m ./mpi_pi.mapper -k ./mpi_pi.kvgen.sh \
-r ./mpi_pi.reducer -w job.sh
```

- ジョブ実行

```
$ pjsub job.sh
```

MPIプログラムでは必ず2以上を指定

K computer

[補足] ノード・プロセス数指定時の注意 (1/2)

- KMRRUNではMPIの動的プロセス生成の仕組みを用いて、Mapper/Reducerを実行しています
- Mapper/Reducerの同時実行数は、生成できる動的プロセス数に依存

最大動的プロセス数 = 全MPIプロセス数 – 静的プロセス数

- 全MPIプロセス数(MPI_UNIVERSE_SIZE)
  - MPIプログラム実行時に決定
    - » インタラクティブ実行の場合、Machinefileで指定するノード数
    - » 京、FX10の場合の場合、ジョブスクリプトのnode, shape, procで決定(次ページに式)
- 静的プロセス数
  - mpirun、mpiexecの「-np」オプションで指定

K computer

[補足] ノード・プロセス数指定時の注意 (2/2)

- 京、FX10でのMPI_UNIVERSE_SIZEの算出式

MPI_UNIVERSE_SIZE = １ノードあたりのプロセス数 x node

１ノードあたりのプロセス数 = proc / shape （小数点以下切り上げ）

- 京、FX10での動的プロセス使用時の特殊な仕様
  1. 1ノードに、静的なプロセスと動的なプロセスは同居できない
  2. 1ノードに、複数の異なるmpi_comm_spawn（動的プロセス生成API）によって起動されたプロセスは同居できない
- 特殊な仕様による弊害
  - 計算リソースの無駄
    - 1により、ノード内コア数分静的プロセスを使わない限り、無駄が生じる
    - 2により、Mapper/Reducerはノード内コア数分プロセスを使わない限り、無駄が生じる
  - KMRRUNにてプロセス起動エラーが発生する場合がある
    - MPI_UNIVERSE_SIZEは上記の式通りにセットされるが、実際に起動できるMapper/Reducer数は上記仕様により、MPI_UNIVERSE_SIZEよりも小さい
    - KMRRUNでは純粋にMPI_UNIVERSE_SIZEまでプロセス起動するので、動的プロセス起動に失敗してエラー終了する場合がある

K computer

Agenda

- KMRのインストール
- KMRRUNによるMapReduce実行
  - PI計算を例に、逐次プログラム、MPIプログラムをMapper/Reducerとして実行
- KMRRUNによる複数計算の一括実行
  - ゲノム解析プログラムを例に、従来なら複数ジョブとして実行する計算を1ジョブにまとめて実行

K computer

ゲノム解析プログラム

- ゲノムシークエンサーからの大量の出力データを解析
  - 出力は一定長の塩基配列の羅列
  - 既知のゲノム配列のどの部分に該当するか、全配列に対して処理する（アライメント処理）

```
ATTTTGCAGCT
ATGGCGAAGTC
ATCGATGGCGA
...
```

解析対象ゲノム

参照ゲノム

```
    GATCGCG
     ATGGCGAA
ATCGATGGCGAACTTAC...
```

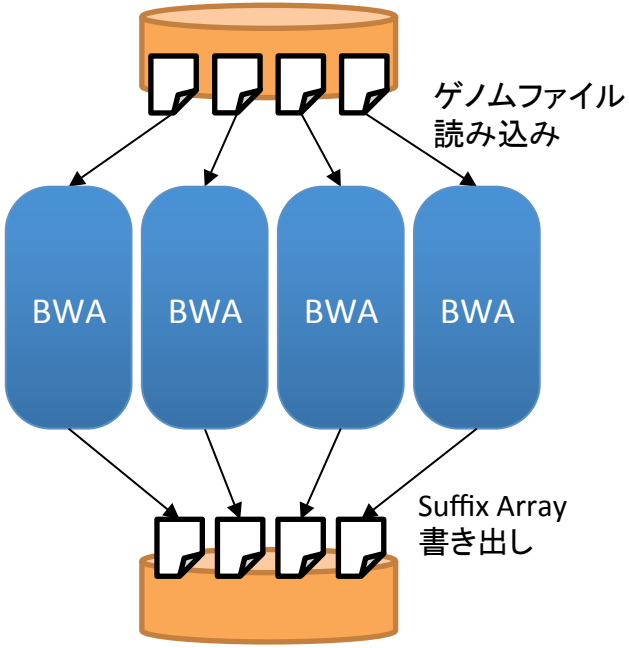
- アライメントツール BWA を、多数に分割されたゲノムデータに対して繰り返し実行
  - ハンズオンでは、入力ゲノムファイルに対してSuffix Arrayを作る処理のみを行う

BWA : http://bio-bwa.sourceforge.net/

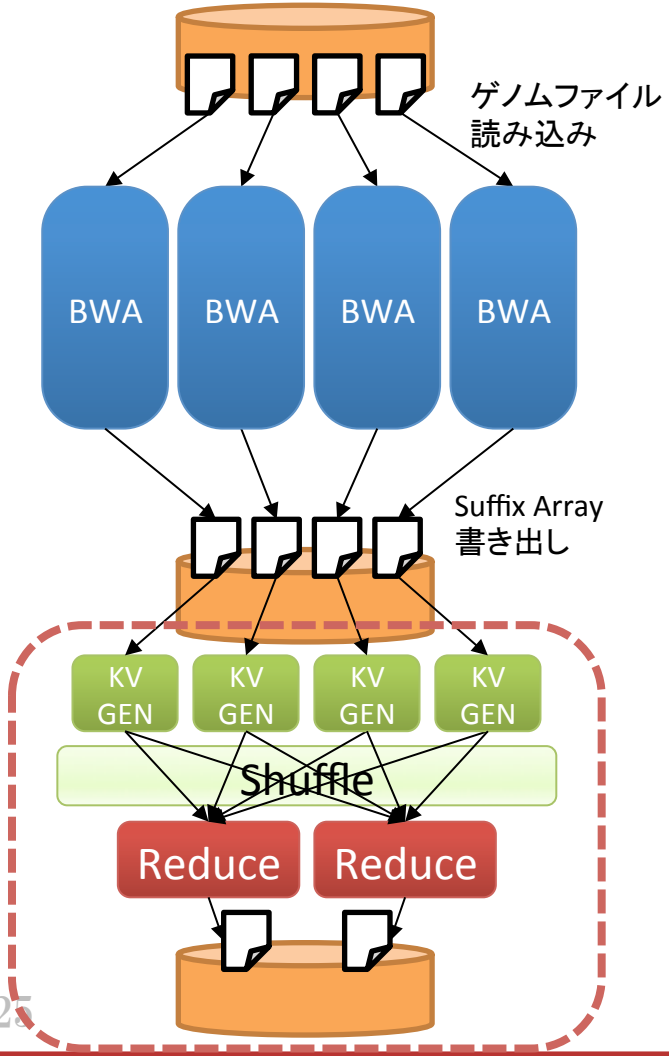
K computer

ゲノム解析プログラム実行イメージ

BWAの実行パターン



KMRRUNへのマッピング



Reduceを行わない
KMRRUNを実行

K computer

BWA実行の実装

- bwa コマンドを実行するラッパープログラムとしてシェルスクリプトで実装
- 入力
  - 参照ゲノム
  - 解析対象ゲノムファイル
- 出力
  - Suffix Arrayファイル
- 処理
  - bwa コマンド(逐次プログラム)のパラメータを設定して実行

実行イメージ

```
$ ls ./
bwa        bwa_task.sh
bwa_db/    input/
$ ls bwa_db
reference.fa.bwt
reference.fa.rbwt
$ ls input
part_1.000    part_1.001
$ ./bwa_task.sh \
./bwa_db/reference.fa \
./input/part_1.000
...(エラー出力を表示)
$ ls ./
bwa            input/
bwa_db/        output/
bwa_task.sh
$ ls ./output
part_1.000.sai
```


K computer

BWAの実行 (1/2)

1. 入力ファイル群・KMRRUNを作業ディレクトリにコピー

```
$ cp -r /data/acourse/141111_kmr_handson/samples/genome \
~/work
$ cp ~/lib/kmr-1.3.2/lib/kmrrun ~/work/genome
```

2. ジョブスクリプト作成プログラムを実行
   – ステージング指示文を削除、言語環境を修正

```
$ cd ~/work/genome
$ kmrrungenscript.py -t 00:10:00 -e 4 -s 1 -p 8 \
-d ./input -n 1 \
-m "./bwa_task.sh ./bwa_db/reference.fa" -w job.sh
```

クオーテーションでくくり、引数に
参照ゲノムファイルを指定

3. ジョブ実行

```
$ pjsub job.sh
```


K computer

BWAの実行 (2/2)

5. 出力を確認

```
$ ls
job.sh          job.sh.i13296  output/
job.sh.e13296   job.sh.o13296
$ ls output
part_1.000.sai  part_1.001.sai
```


K computer

おわり

ご質問・お問い合わせ

丸山: nmaruyama
松田: m-matsuda
滝澤: shinichiro.takizawa @riken.jp

K computer