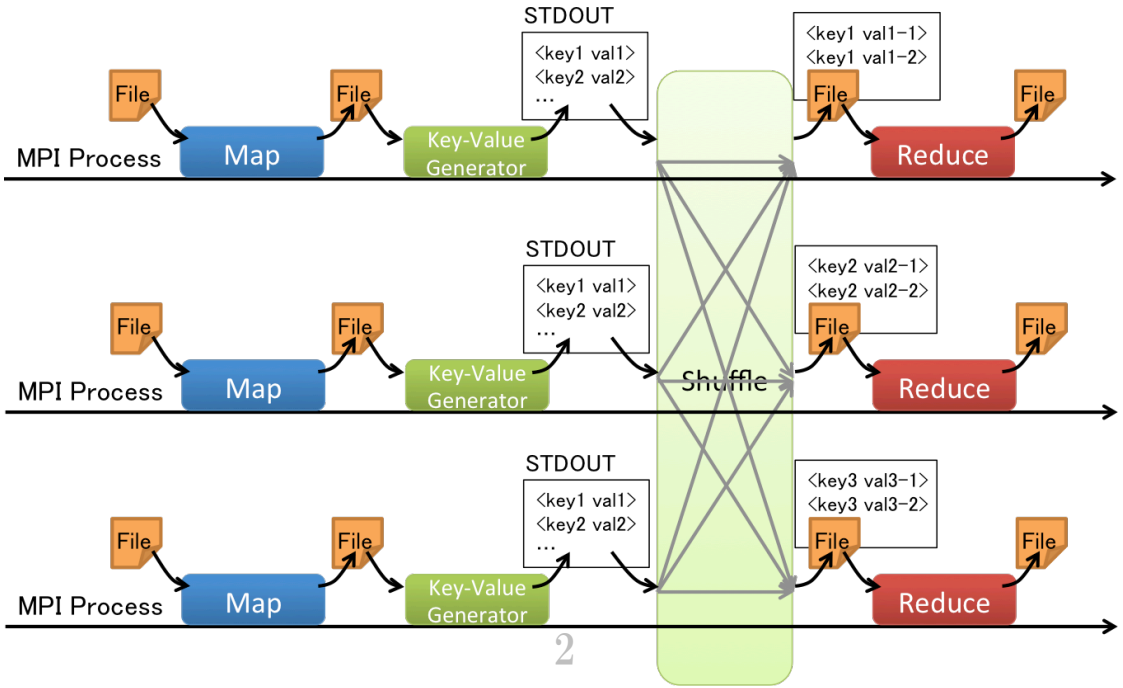
第2回 AICS公開ソフト講習会
K MapReduce 補足資料

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

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

	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 • ファイル名はコマンドの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コマンドの最後の引数として渡される • Mapperの出力を読み込む場合は、ファイル名を類推して作成	• ファイル読み込み • ファイル名はコマンドの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

PI計算サンプルプログラム

- モンテカルロ法による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_SRC/kmrrun
$ make pi.mapper pi.reducer
$ make mpi_pi.mapper mpi_pi.reducer
```

PI計算の実装 (1/3)

- Mapper
  - 点の数が書かれた入力ファイルを受け取る
  - 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
  - 「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 ./work/000
10000
$ ./pi.mapper ./work/000
$ ls ./work
000    000.out
$ cat ./work/000.out
7829/10000
```

PI計算の実装 (2/3)

- KV Generator
  - Mapperの入力ファイルを受け取る
  - Mapperの出力ファイルパスを作成
  - 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
  - Mapperの出力ファイルを削除

実行イメージ

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

PI計算の実装 (3/3)

- Reducer
  - 1行に1KVが書かれた入力ファイルを受け取る
  - 点の数よりPIを計算
  - 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
```

KMRRUNコマンド

- 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	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。

PI計算の実行:インタラクティブ実行 (1/2)

1. ワーキングディレクトリに実行ファイルをコピー

```
$ cp KMR_INST/lib/kmrrun .
$ cp KMR_SRC/kmrrun/pi.mapper .
$ cp KMR_SRC/kmrrun/pi.reducer .
$ cp KMR_SRC/kmrrun/pi.kvgen.sh .
```

2. 入力ファイルの用意

```
$ mkdir ./inp
$ echo 10000 > ./inp/000; echo 10000 > ./inp/001
$ echo 10000 > ./inp/002; echo 10000 > ./inp/003
```

3. MPIマシンファイルの用意

```
$ echo SERVER1 > machines;  echo SERVER2 >> machines
$ echo SERVER3 >> machines; echo SERVER4 >> machines
$ echo SERVER5 >> machines; echo SERVER6 >> machines
```

PI計算の実行：インタラクティブ実行 (2/2)

4. ファイル確認

```
$ ls
inp/   kmrrun   machines   pi.kvgen.sh   pi.mapper   pi.reducer
```

5. 実行

```
$ mpirun -machinefile machines -np 2 ./kmrrun ¥
-m ./pi.mapper -k ./pi.kvgen.sh -r ./pi.reducer ./inp
3.135600
$ ls
0.out    kmrrun      pi.kvgen.sh    pi.reducer
inp/     machines    pi.mapper
$ cat 0.out
3.135600
```

マシンファイルでは6ノード指定し、-npオプションでは2ノードを指定

⇒ kmrrun, KV Generatorは2ノードで動作
⇒ Mapper, Reducerは残りの4ノードで動作
　Mapperの入力ファイルは4個あるので、4 Mapperが同時実行可能

K computer

PI計算の実行：京で実行 (1/2)

1. ワーキングディレクトリに実行ファイルをコピー

```
$ cp KMR_SRC/kmrrun/pi.mapper .
$ cp KMR_SRC/kmrrun/pi.reducer .
$ cp KMR_SRC/kmrrun/pi.kvgen.sh .
```

2. 入力ファイルの用意

```
$ mkdir ./inp
$ echo 10000 > ./inp/000; echo 10000 > ./inp/001
$ echo 10000 > ./inp/002; echo 10000 > ./inp/003
```

3. ジョブスクリプトを作成

```
$ KMR_INST/bin/kmrrungenscript.py -e 6 -s 2 -p 2 ¥
-m pi.mapper -k pi.kvgen.sh -r pi.reducer -d ./inp ¥
-o 0.out -w job.sh
```

4. 実行

```
$ pjsub job.sh
```

PI計算の実行：京で実行 (2/2)

- 生成されるジョブスクリプト

```
#!/bin/bash -x
#
#PJM --rsc-list "elapse=00:10:00"
#PJM --rsc-list "node=6"
#PJM --mpi "shape=2"
#PJM --mpi "proc=2"
#PJM --rsc-list "proc-core=unlimited"
#PJM --stg-transfiles "all"
#PJM --stgin "KMR_INST/lib/kmr-1.2/lib/kmrrun ./kmrrun"
#PJM --stgin "./pi.mapper ./"
#PJM --stgin "./pi.kvgen.sh ./"
#PJM --stgin "./pi.reducer ./"
#PJM --stgin "./inp/* ./work/"
#PJM --stgout "./0.out ././/"
#PJM --stgout "./core* ./"
#PJM -S

. /work/system/Env_base

mpirun -np 2 ./kmrrun -n 1 -m ./pi.mapper -k "./pi.kvgen.sh" -r ./pi.reducer ./work
```

KMRShell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
  - 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、または「-h」をつけて実行したヘルプを参照

KMRRUN コツと注意点 (1/3)

- Mapperだけを実行したい
  - 結果をReduceする必要なく、異なるパラメータで複数の計算を実行したい場合など、Reducer実行が不要な場合があります。その時にはKV GeneratorやReducerは省略できます。

```
$ mpirun -np 4 ./kmrrun -n 8 -m ./mapper ./input
```

複数の計算を1つのジョブとしてまとめて実行したいときに有効

KMRRUN コツと注意点 (2/3)

- 障害等で実行が中断した時に、実行を再開したい
  - KMRRUNでは実行状態の保存と再開を実現する機能 Checkpoint/Restart を提供します。再開時にはノード数を減らしての再開もサポートします。

```
$ mpirun -np 4 ./kmrrun -n 8 -m mapper -k kvgen.sh ¥
-r reducer --ckpt ./input
```

  - 中断時には、カレントディレクトリにチェックポイントファイル（ckptdirXXXXX, XXXXXはランク）を生成
  - チェックポイントファイルが存在する時に --ckpt オプションを指定すると、保存された状態から処理を再開

KMRRUN コツと注意点 (3/3)

- オプション「-n」(1回のMapper/Reducer実行時の使用プロセス数)の指定方法
  - Mapper/Reducerが逐次プログラムの場合 => 1
    - 1が指定された場合は、逐次プログラムと自動判定します
  - Mapper/ReducerがMPIプログラムの場合 => 2以上
    - MPIプログラムは並列度1では実行できません

- Reduce結果はソートされない
  - HadoopではReduce結果はソートされて出力されるが、KMRではソートしません

- MPIで動作しているので、1プロセスで障害が起こると、全体の実行が中断する

KMRRUNとKMRShellの違い

	KMRRUN	KMRShell
Mapper/Reducer 入力	• ファイル名がMapper/Reducerに渡される • 入力ファイルは共有ファルシステム上に格納	• ファイルコンテンツがMapper/Reducerの標準入力に渡される • 入力ファイルはプロセスローカルなファイルシステム上に格納
Mapper/Reducer 出力	ファイルとして出力	標準出力に出力
Mapper/Reducer プログラムの種類	• 逐次・ノード内並列プログラム • MPIプログラム	• 逐次プログラム
KVの生成方法	KV Generatorプログラムにて、Mapper出力を読み込み、標準出力に出力	Mapperが標準出力にKVを出力
Checkpoint/Restart	対応	未対応

- KMRShellは今後機能拡張の予定はありません

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