MapReduce プログラミングモデル

- p.4 入力データはテキストのみか、バイナリも OK か？
  - 処理系依存。バイナリを読み込み、テキスト形式にして Map の入力とするものもある。
- p.4 KV の Key と Value はテキストかバイナリか
  - 何方でも可。
- p.11 塩基配列パターンの長さ（contig のこと）は？
  - 具体的には覚えていないが、日本人男性 1 人ゲノム解析で用いられたデータを使っており、公開されている。
- p.13 MapReduce で実行することにより、並列化できない部分、例えば初期化処理のオーバーヘッドが大きく現れることは無いか
  - ある。並列化できる部分の割合による。

K MapReduce

- p.20 オンメモリで shuffle を行うということは、通信を行わないのか
  - 表現が悪かった。ファイル IO を行わないという意図である。
- p.22 MPI / OpenMP で自動並列するということは、利用者が実装する Mapper/Reducer においては OpenMP 並列化も不要か
  - Key-Value 単位で MPI / OpenMP 並列実行されるので、その粒度での並列実行で十分であれば不要である。
- p.23 MPIIO を使っているのか
  - 使っていない。
- p.23 この手法を用いた場合、1 つの大きいファイルを読み込むのが良いのか、それとも予めある程度のサイズに分割したファイルを読み込むのが良いのか、何方が性能的に優れているか。
  - 利用者レベルでファイルを分割していても、結局はファイルシステムへのアクセスはストライプ分割された、複数 IO ノードへのアクセスとなるため、ストライプカウントを適切に設定すれば、どちらも性能は変わらないと思われる。

KMR 利用方法

- p.41 Map により生成された Key の個数が、（Reduce 実行する）ノード数

よりも多くなった場合にはどうなるか。

- ハッシュの性能に依存するが、おおよそ均等になるように、Key がノードに分散される。

- p.50 KVS はメモリ上に確保されるとのことであるが、メモリ量を超えた場合はどうなるか。
  - アボートする。そのような場合には、実行ノード数を増やし、総メモリ量を増やすしか無い。1 ノードに 16GB を超える KV が集まった場合には、対処方法が無い。京コンピュータで動作する、KV をファイルに保存する MapReduce 処理系として MR-MPI というものがある。

KMR 利用事例

- p.70 REMD の KMR 実装は公開されているのか。読みやすいか。
  - 公開はしていないが、参考にした REIN が GPL で公開されているので、同じ GPL として公開可能である。REIN よりも 500 行ほど行数が増えていること、Fortran に不慣れな人間が実装しているので、読みやすくはないかもしれない。
- p.74 REMD の結果にて、MapReduce のほうが性能よく見えるがなぜか。
  - 行っている計算は全て同じであり、IO、通信量も少ない。タスク管理が KMR の方が優れていると思われる。