2013年度
実践的並列コンピューティング
第13回

MapReduce Programming その１
2013年7月8日

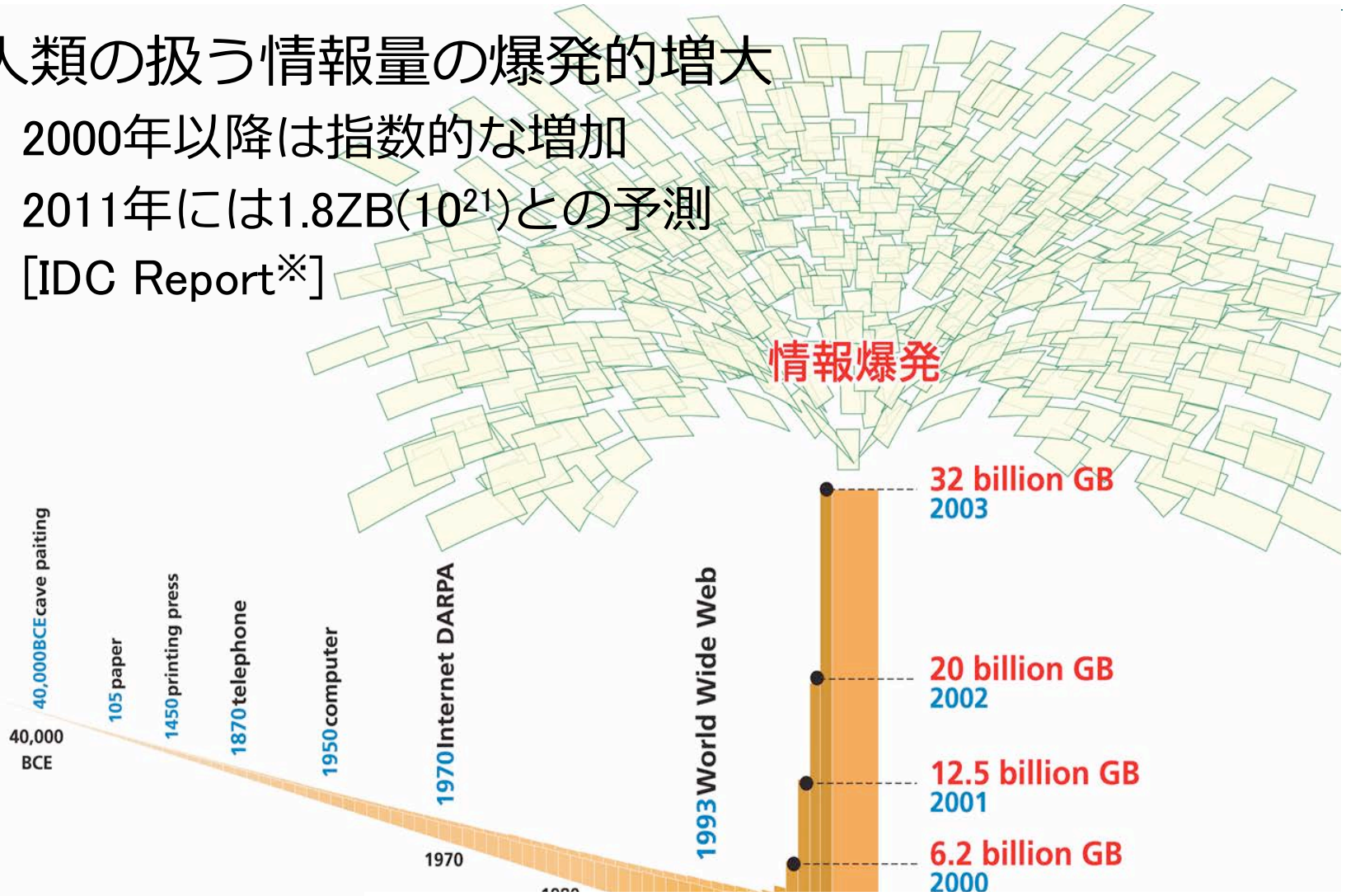
遠藤敏夫（endo@is.titech.ac.jp）
元スライド作成：佐藤仁　学術国際情報センター

授業の目的

- 大規模データ処理に関する最新の話題
- 並列ファイルシステムとは
- 大規模データ処理のためのプログラミングモデルであるMapReduceの概念の理解
- 一般的に使われているMapReduce処理系であるHadoopのプログラミングの理解

情報爆発時代

- 人類の扱う情報量の爆発的増大
  - 2000年以降は指数的な増加
  - 2011年には1.8ZB(10^{21})との予測
    [IDC Report※]

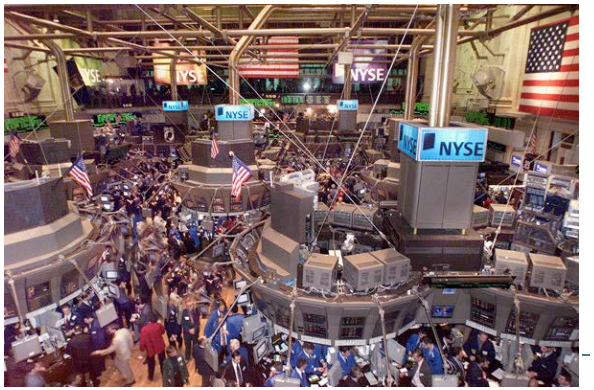


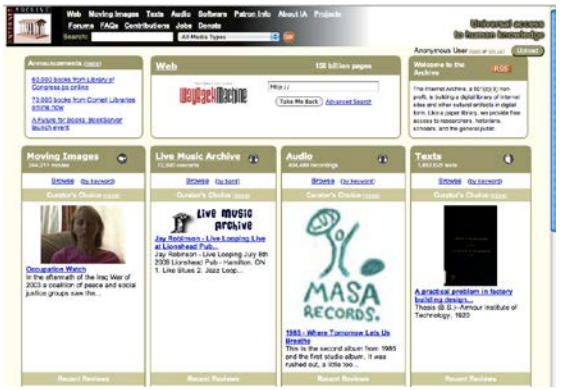
Source: Horison Information Strategies, cited from Storage New Game New Rules. (www.horison.com)

※ Gantz et al., "The Diverse and Exploding Digital Universe," March 2008.

情報爆発の要因の例

- NY証券取引所では1日に1TBの取引データが発生
  - 常に大量のデータがストリームされる
- Facebookでは10億枚の写真データ(1PB)を保持
  - 他のインターネットサービス(Youtubeなど)も同様
- インターネットアーカイブは2PBあり、1ヶ月で20TB増え続けている
  - Web検索(Googleなど)に影響





情報爆発の要因の例（cont’d）

- CERNのLHC実験では１年に15PBのデータが生成
  - 他の科学技術分野(バイオインフォマティクス)なども同様
  - 解析器などのデバイスの性能向上も影響

→・常に大量のデータが生成される
  ・生成された大量のデータに対する解析が必要

例. 天候データの解析

- NCDC天候データ
  - センサーにより地球上の任意の場所の天候データ(気温など)を１時間毎に収集
  - ASCIIファイルとして保存
    - 左の図はある1つのサンプルを表したもの
      - 実際には改行せず複数行ある
    - 日付とセンサーの場所毎にファイルを保存
      - 1901年から毎年分

  → 大量のASCIIファイルを処理しなければならない

```
0057
332130 # USAF weather station identifier
99999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time
4
+51317 # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM-12
+0171 # elevation (meters)
99999
V020
320 # wind direction (degrees)
1 # quality code
N
0072
1
00450 # sky ceiling height (meters)
1 # quality code
CN
010000 # visibility distance (meters)
1 # quality code
N9
-0128 # air temperature (degrees Celsius x 10)
1 # quality code
-0139 # dew point temperature (degrees Celsius x 10)
1 # quality code
10268 # atmospheric pressure (hectopascals x 10)
1 # quality code
```

※　NCDC(National Climatic Data Center)

フォーマットの例

天候データのファイルのサンプル

```
0029029070999991901010106004+64333+023450FM-12+000599999V0202701N015919999999N0000001N9-00781+99999102001ADDGF108991999999999999999
0029029070999991901010113004+64333+023450FM-12+000599999V0202901N008219999999N0000001N9-00721+99999102001ADDGF104991999999999999999
0029029070999991901010120004+64333+023450FM-12+000599999V0209991C000019999999N0000001N9-00941+99999102001ADDGF108991999999999999999
0029029070999991901010206004+64333+023450FM-12+000599999V0201801N008219999999N0000001N9-00611+99999101831ADDGF108991999999999999999
0029029070999991901010213004+64333+023450FM-12+000599999V0201801N009819999999N0000001N9-00561+99999101761ADDGF108991999999999999999
0029029070999991901010220004+64333+023450FM-12+000599999V0201801N009819999999N0000001N9-00281+99999101751ADDGF108991999999999999999
0029029070999991901010306004+64333+023450FM-12+000599999V0202001N009819999999N0000001N9-00671+99999101701ADDGF106991999999999999999
0029029070999991901010313004+64333+023450FM-12+000599999V0202301N011819999999N0000001N9-00331+99999101741ADDGF108991999999999999999
0029029070999991901010320004+64333+023450FM-12+000599999V0202301N011819999999N0000001N9-00281+99999101741ADDGF108991999999999999999
0029029070999991901010406004+64333+023450FM-12+000599999V0209991C000019999999N0000001N9-00331+99999102311ADDGF108991999999999999999
0029029070999991901010413004+64333+023450FM-12+000599999V0202301N008219999999N0000001N9-00441+99999102261ADDGF108991999999999999999
0029029070999991901010420004+64333+023450FM-12+000599999V0202001N011819999999N0000001N9-00391+99999102231ADDGF108991999999999999999
0029029070999991901010506004+64333+023450FM-12+000599999V0202701N004119999999N0000001N9+00001+99999101821ADDGF104991999999999999999
0029029070999991901010513004+64333+023450FM-12+000599999V0202701N002119999999N0000001N9+00061+99999102591ADDGF104991999999999999999
0029029070999991901010520004+64333+023450FM-12+000599999V0202301N004119999999N0000001N9+00001+99999102671ADDGF104991999999999999999
0029029070999991901010606004+64333+023450FM-12+000599999V0202701N006219999999N0000001N9+00061+99999102751ADDGF103991999999999999999
0029029070999991901010613004+64333+023450FM-12+000599999V0202701N006219999999N0000001N9+00061+99999102981ADDGF100991999999999999999
・・・
```

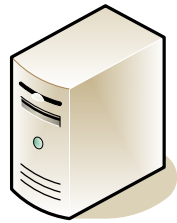
```
$ ls raw/1990 | head

010010-99999-1990.gz
010014-99999-1990.gz
010015-99999-1990.gz
010016-99999-1990.gz
010017-99999-1990.gz
010030-99999-1990.gz
010040-99999-1990.gz
010080-99999-1990.gz
010100-99999-1990.gz
```

010010-99999-1990.gz

- 実際には大量のファイルに対して解析を行う必要がある

既存のツールを使って解析

- 前提
  - Linuxマシンが1台
  - ディスクに天候データのASCIIファイルを保持
- どうすればよいか？
  - ディスク上のファイルに対してスクリプトを実行

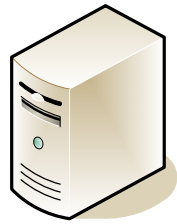


```
#!/usr/bin/env bash
for year in all/*
do
        echo -ne `basename $year .gz`"¥t"
        gunzip -c $year | ¥
        awk '{ temp = substr($0, 88, 5) + 0;
                 q = substr($0, 93, 1);
               if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
               END { print max }'
done
```

max_temperture.sh

既存のツールを使って解析 (cont’d)

- 実行
  - 得たい結果は得られるが。。。



```
$ ./max_temperature.sh
1901 317
1902 244
1903 289
1904 256
1905 283
...
```

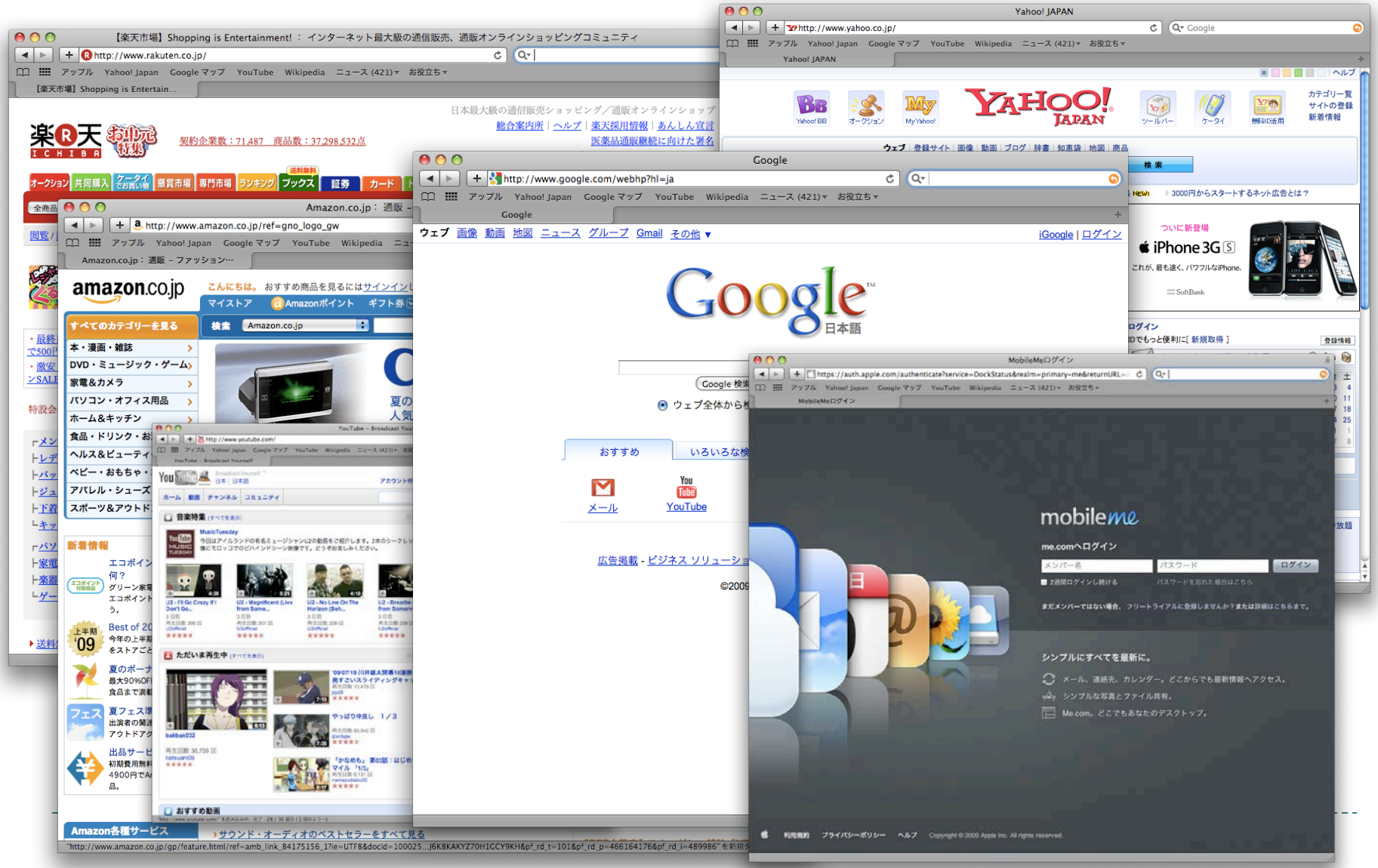
1901年　31.7℃
1902年　24.4℃
1903年　28.9℃
1904年　25.6℃
1905年　28.3℃

問題点

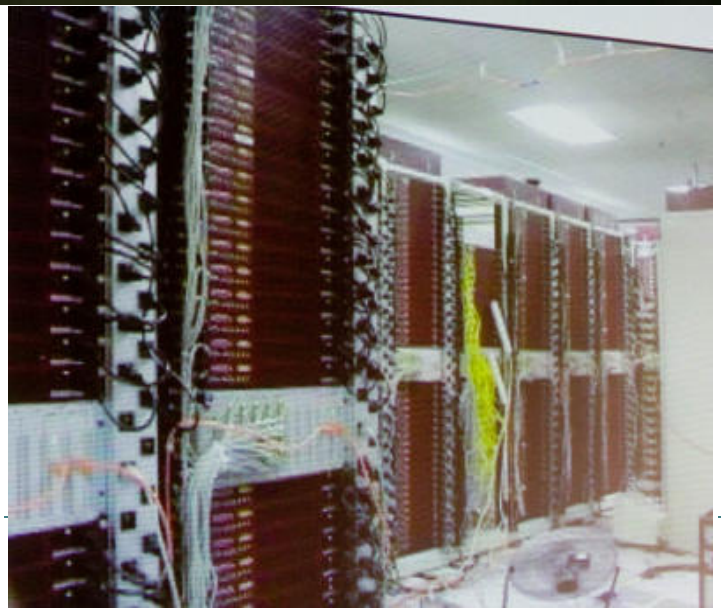
- 解析処理の問題点
  - CPU1コアでの処理には限界
    - 例. 20世紀の全データへの解析は、Amazon EC2の場合で42分
    - CPUのマルチコア化
      - TSUBAME2の場合、1ノード12コア
- I/Oの問題点
  - スループット
    - HDD(SATA2) : 実測150MB/s前後
    - SSD : 実測100～500MB/s前後
  - ディスクあたり容量
    - HDD : 数TB程度
    - SSD : 0.5TB程度

数十TB～数PBを対象にした
データ解析には全く性能が足りない

いろいろなWeb Site



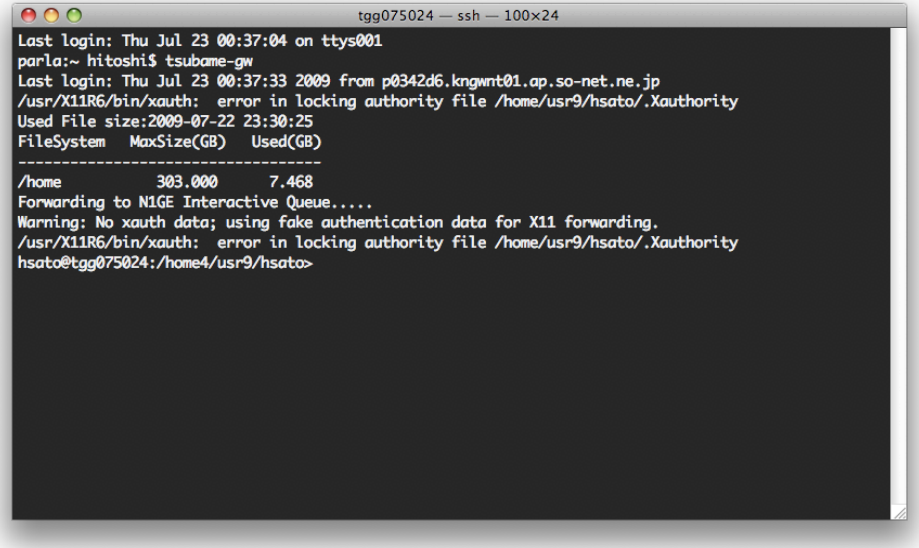
裏側の世界



スーパーコンピューター

- 内部の演算処理速度がその時代の一般的なコンピュータより極めて高速な計算機
- 例: 東工大TSUBAME2.0

大岡山キャンパス　図書館の隣

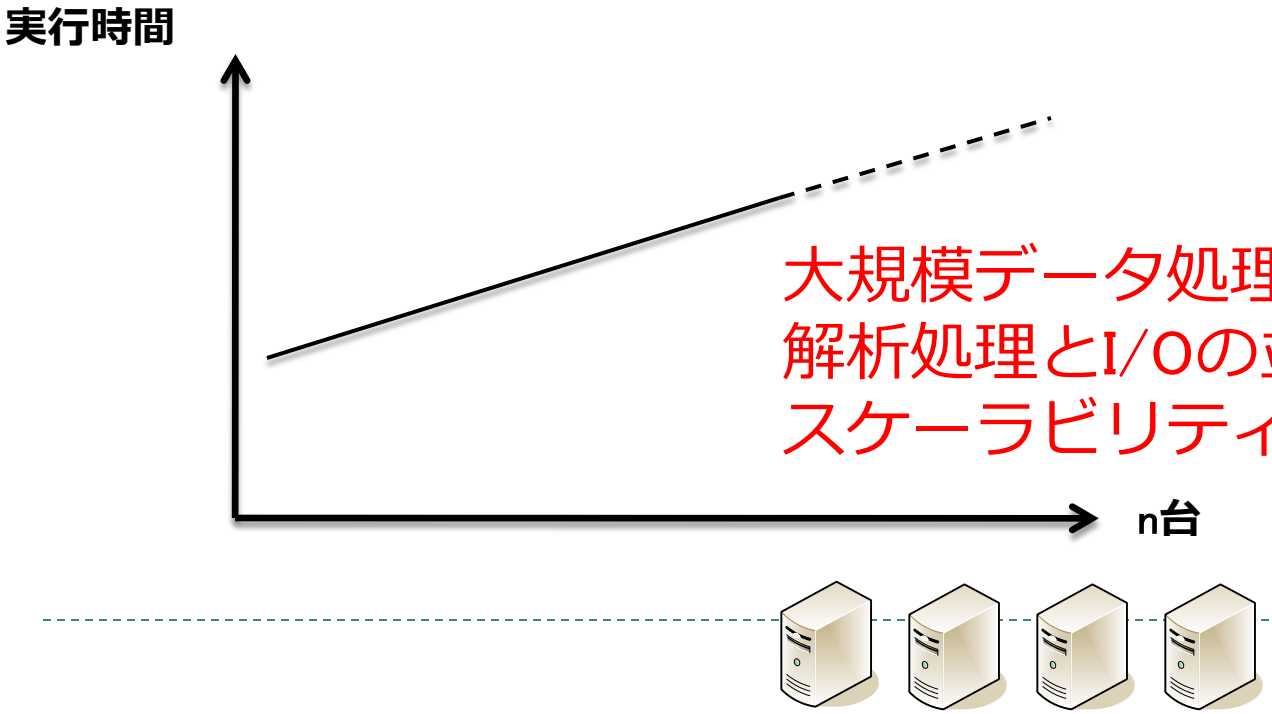


使うときの見た目は普通の端末とあまり変わらず．．


DataDirect

スケーラビリティ

- 仕事の増大に適応する能力･度合い
- 例
  - マシンの台数を増やしたら，，，
    - 解析処理の実行時間が短くなった
    - データ処理の時間が短くなった

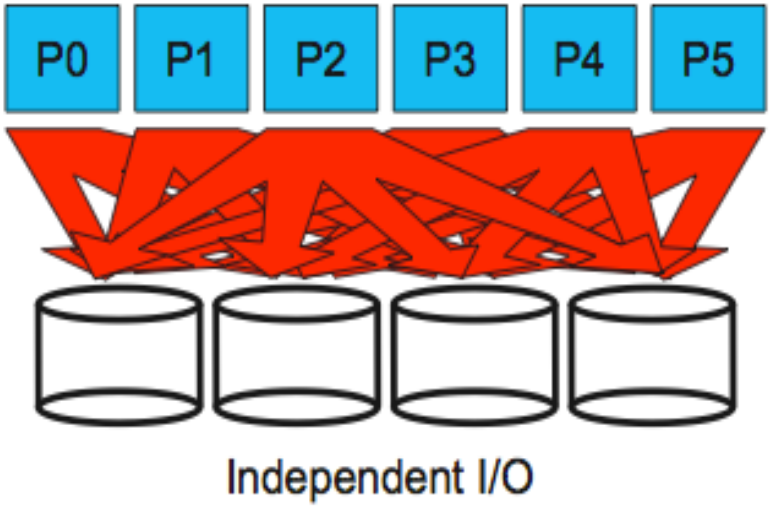


大規模データ処理を行うためには
解析処理とI/Oの並列化による
スケーラビリティの達成が不可欠

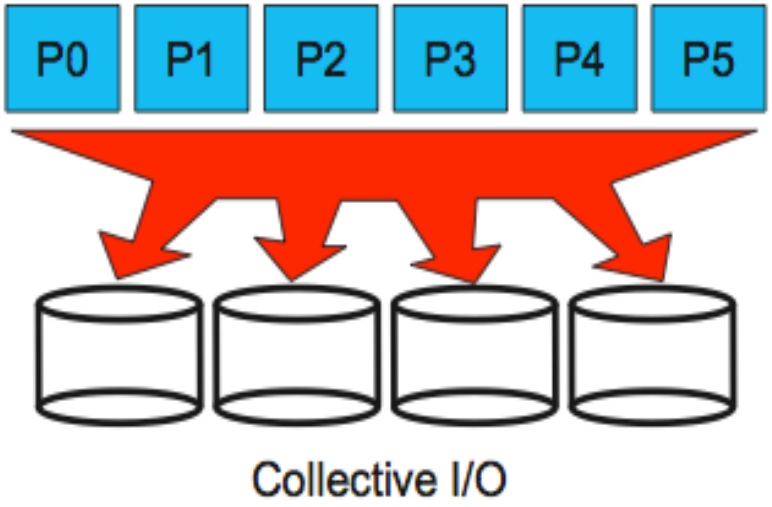
スパコン上でのI/O

- シミュレーション (従来のHPC)
  - 中間データ，結果の出力
    - メモリ内の8GBのデータを200回出力したい → 1.6TB
    - 4(複数)パターン計算したい → 6.4TB
- データ集約的計算 (大規模データ処理)
  - WEB解析
    - 10億ページ規模のHTML(圧縮2TB)に言語解析 → 圧縮20TBの出力
    - 他にもバイオ系(ゲノム), 論文データ，映像などの解析など多数

並列IO (e.x. MPI-IO)



Independent I/O

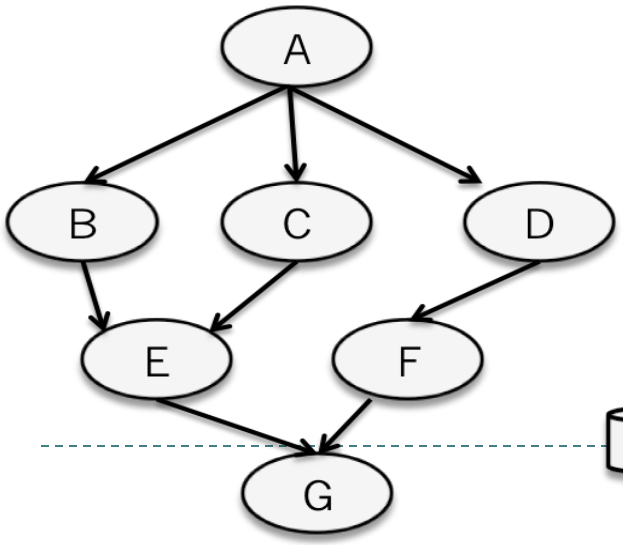


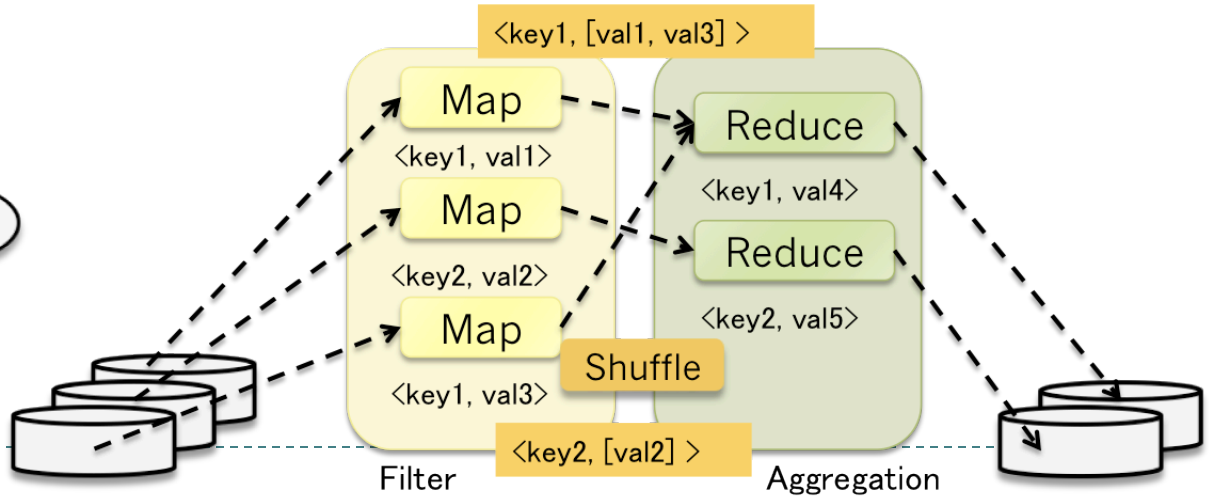
Collective I/O

- 異なるプロセスが並列にディスクへアクセスする
  - Independent I/O
    - 各々のプロセスが各々I/Oを行う
  - Collective I/O
    - プロセス全体が一斉にI/Oを行う
    - チェックポイントなど．．

データ集約的計算

- ワークフロー
  - ジョブのワークフローをDAGとして表現
    - TEXT, XML, Make,
    - 既存アプリを容易に組み合わせられる
  - ex.) DAGMan, Pegasus, Dryad, GXP Make, etc.
- MapReduce
  - ex.) Google MapReduce, Hadoop, etc.

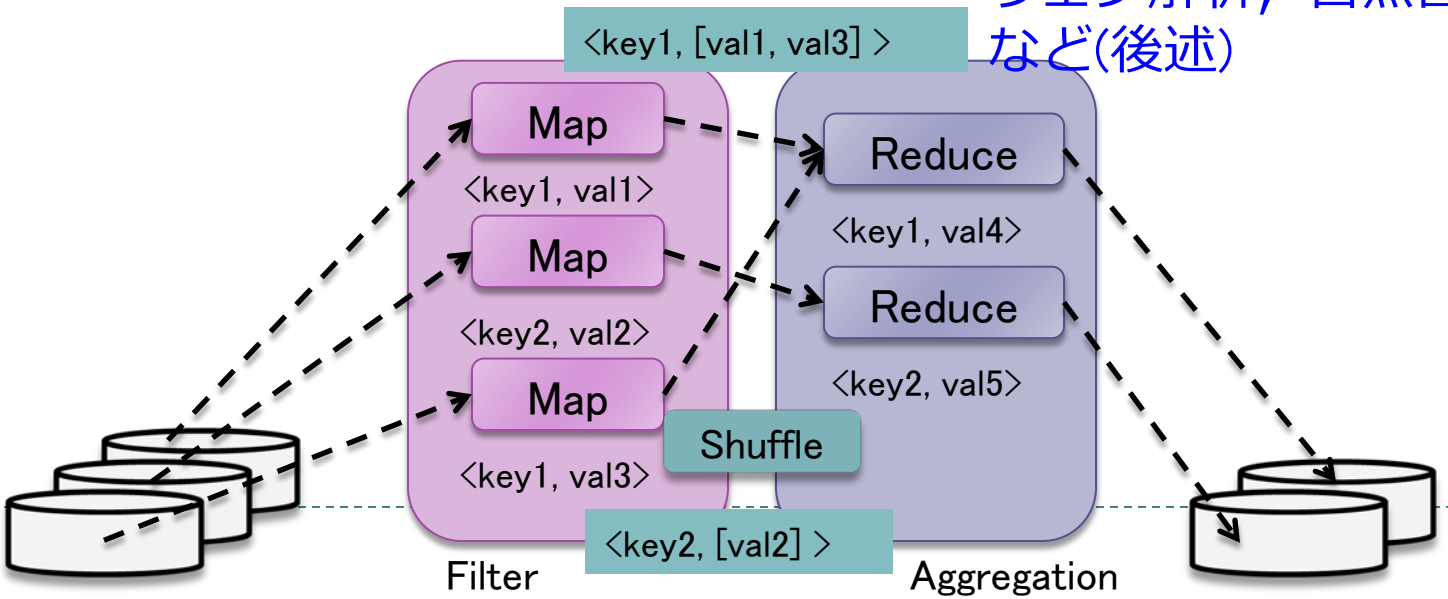




MapReduce

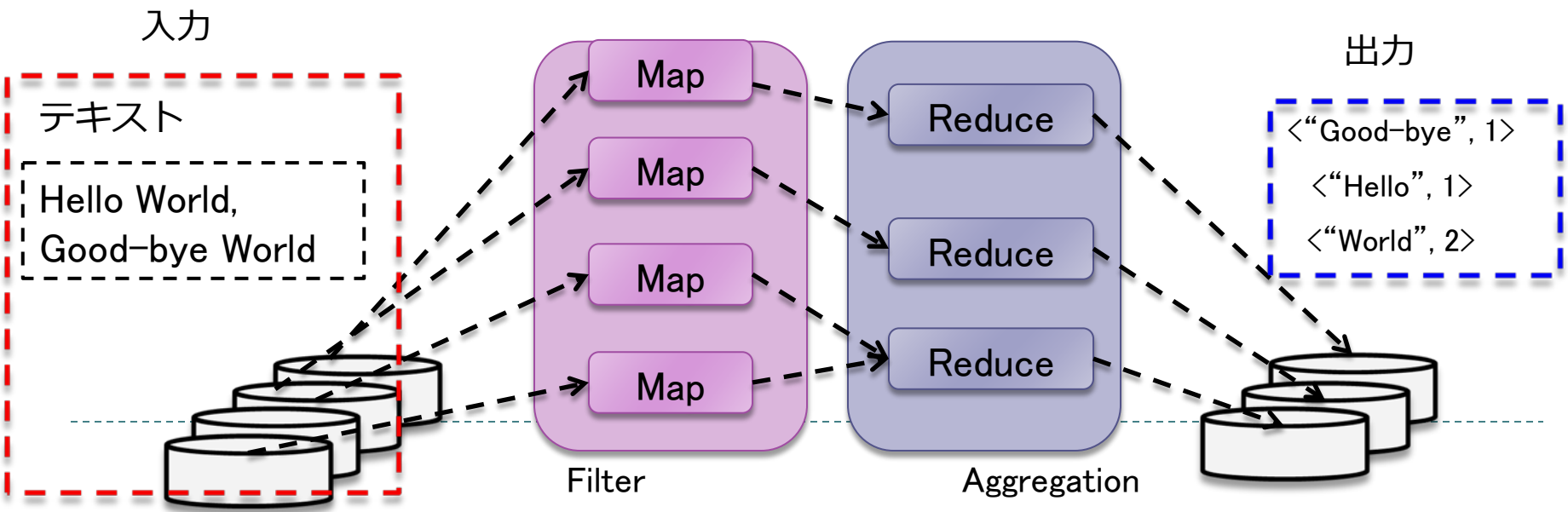
- 2004年にGoogle社により提案された大量のデータを並列に処理するためのプログラミングモデル（とその実装）
  - Mapフェーズ
    - keyとvalueのペアから中間データとなるkeyとvalueのペアを生成する
  - Reduceフェーズ
    - 中間データから同じkeyに関連づけられたvalueを集めて処理し，結果を出力する

ウェブ解析，自然言語処理など(後述)



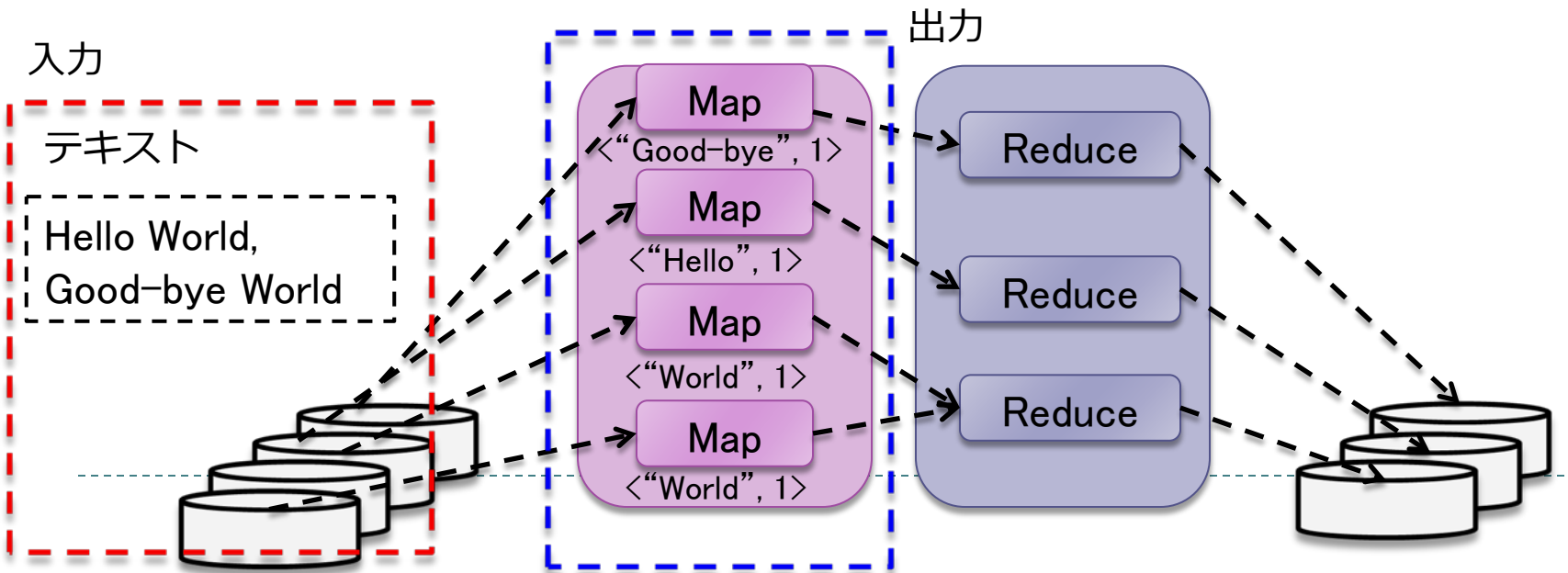
例. WordCount (単語数え)

- 目的
  - (複数)ファイル中の単語の数を数えたい
    - 入力
      - Key : ドキュメントのID(ファイル名) と Value : テキスト
    - 出力
      - Key : 単語 と Value : ファイル中に出現する数



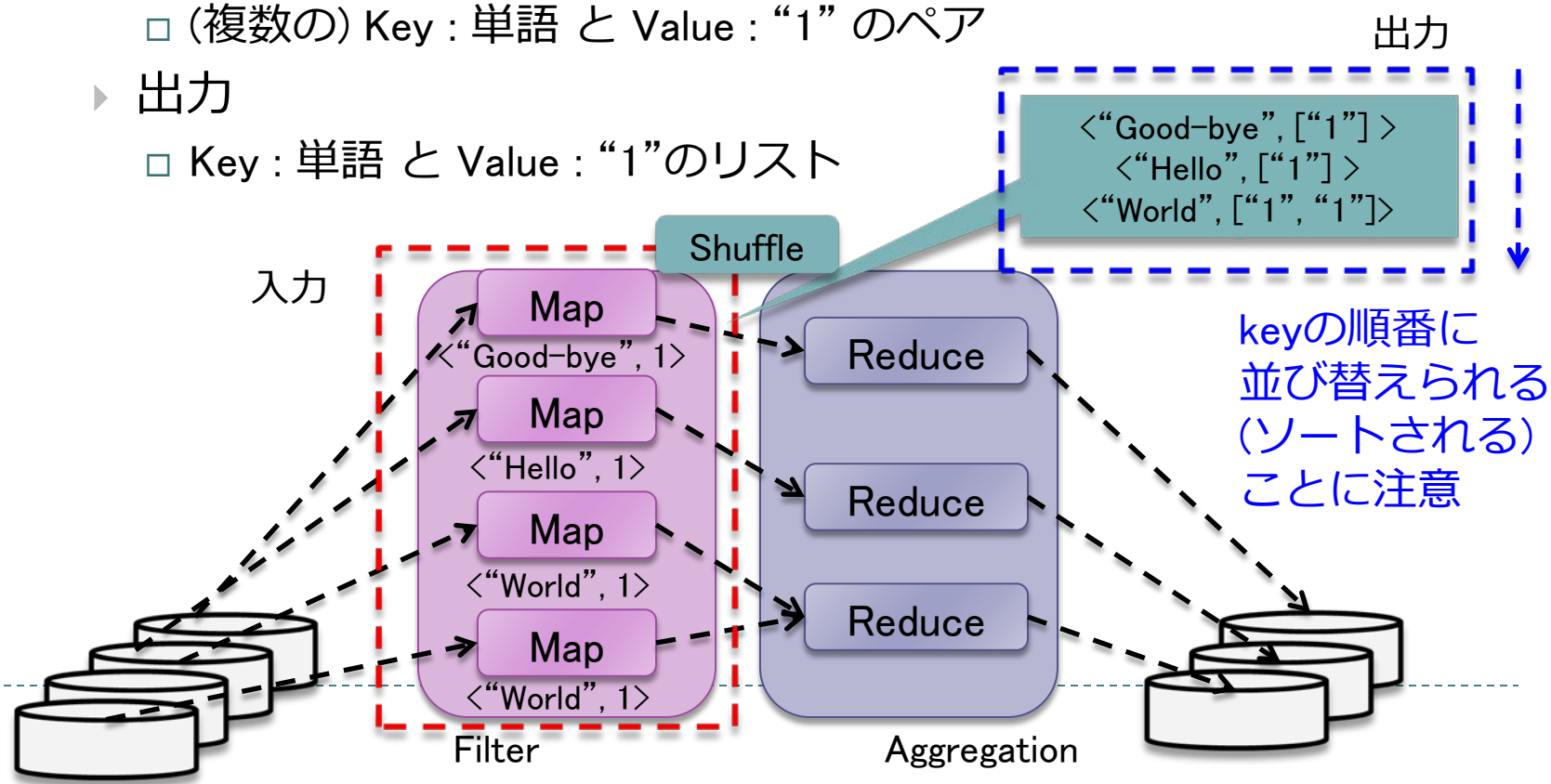
例. WordCount (単語数え) : Mapフェーズ

- Map処理
  - テキストから単語(key)とその存在を表す”1”(value)からなるkeyとvalueのペアを生成
    - 入力
      - Key : ドキュメントのID(ファイル名) と Value : テキスト
    - 出力
      - Key : 単語 と Value : “1”



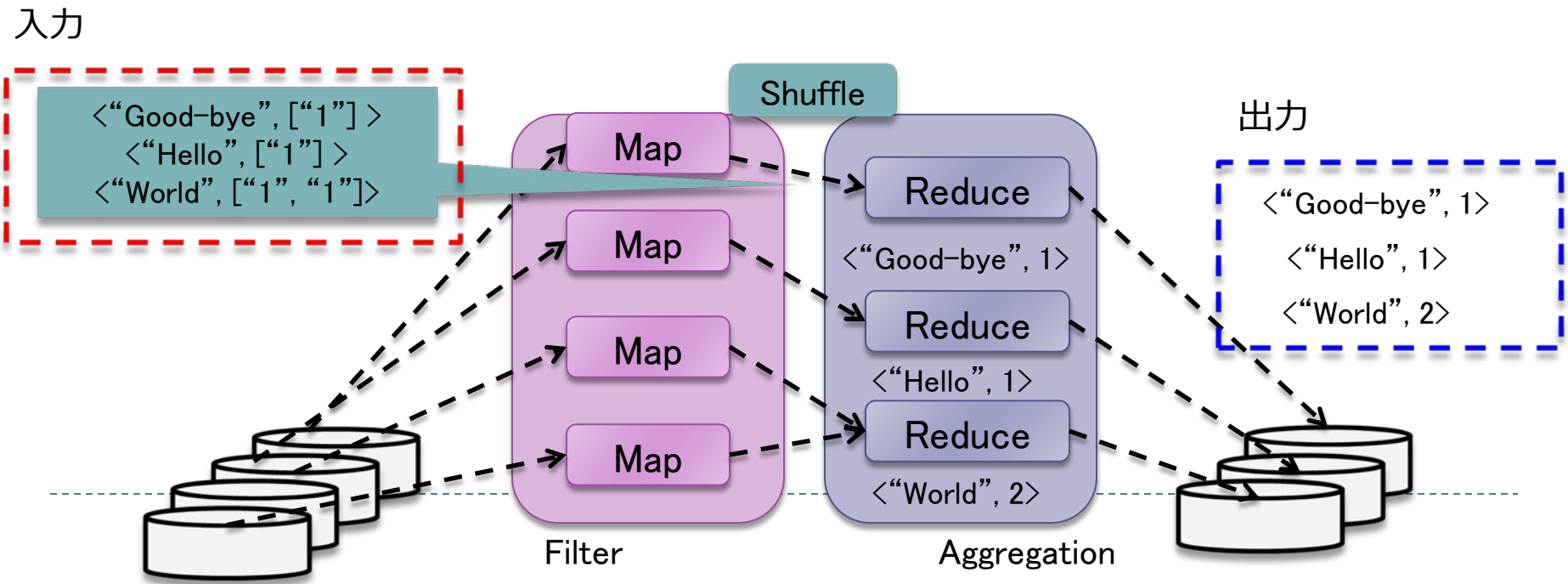
例. WordCount (単語数え) : Shuffleフェーズ

- Shuffle処理
  - Map処理の中間データを(keyとvalueのペア)を整理し、同じkeyのvalueを集める
    - 入力
      - (複数の) Key : 単語 と Value : “1” のペア
    - 出力
      - Key : 単語 と Value : “1”のリスト



例. WordCount (単語数え) : Reduceフェーズ

- Reduce処理
  - 同じ単語(key)に対する”1”の数を足し合わせる
    - 入力
      - Key : 単語 と Value : “1”のリスト
    - 出力
      - Key : 単語 と Value : ファイル中に出現する単語の数



MapReduceの典型的な処理

- カウンタ (既出)
  - 入力ファイルから条件に合うデータの数を数える
    - Map
      - Key : 数える対象(文字などなんでも)、Value : “1” として出力
    - Reduce
      - Keyについて “1”の数を数え上げる
- 分散Grep
  - ファイル(複数)から特定の文字列を含んだ行を見つける
    - Map
      - 目的の文字列を探し、見つかった時だけ、Valueとして出力
    - Reduce
      - 特に何もしないでValueを出力

MapReduceの典型的な処理 (cont'd)

- 分散Sort
  - 入力データを順番に並び替える
    - Map
      - 並べ替えたいものをkeyとして出力する
      - keyがShuffle処理で並べ替えられる
    - Reduce
      - そのまま出力すると、並べ替えた状態の出力が得られる
- 逆リンクリスト
  - Webページからリンク情報を抜き出す
    - Map
      - Key : WebページのURL、Value : HTML を入力
      - Key : 自分のURL、Value : リンク先のURL を出力
    - Reduce
      - Key : リンク先のURL、Value : 自分のURL として出力

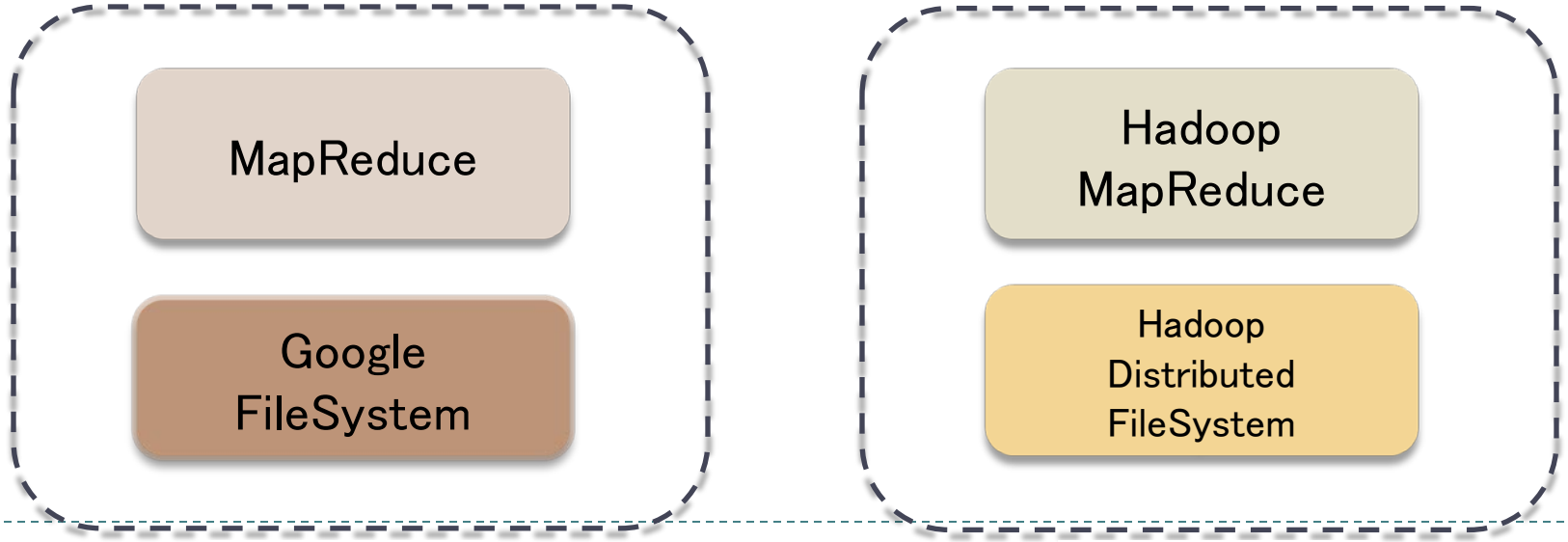
MapReduceの処理系

- Google MapReduce
  - 2004年の論文のベースになった実装
  - http://labs.google.com/papers/mapreduce.html
- Apache Hadoop
  - Javaで実装され最もポピュラー
  - http://hadoop.apache.org/
  - http://www.cloudera.com
- Phoenix
  - マルチコア、共有メモリ環境に特化
  - http://mapreduce.stanford.edu
- Sector/Sphere
  - C++で実装されHadoopより高速なのが売り
  - 広域環境も考慮？
  - http://sector.sourceforge.net
- Mars
  - GPU上でMapReduceを行う
  - http://www.cse.ust.hk/gpuqp/Mars.html

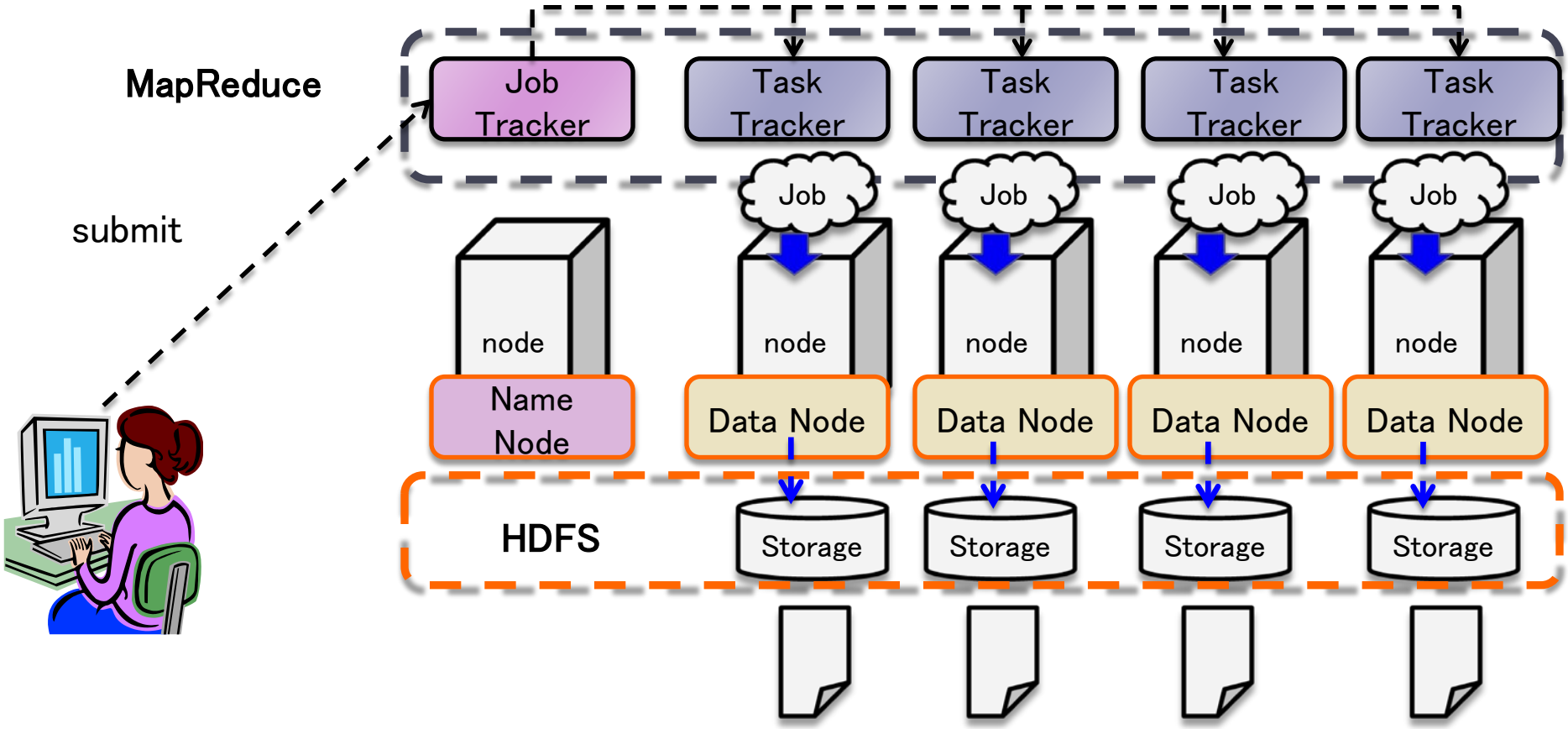
他にもたくさん。。。

Hadoop

- Apacheプロジェクトによる**OSS**
- グーグル社にインスパイアされて開発されている**MapReduce**の実装
  - MapReduce処理系
    - MapReduce vs Hadoop MapReduce
  - ファイルシステム
    - GFS vs Hadoop Distributed Filesystem (HDFS)

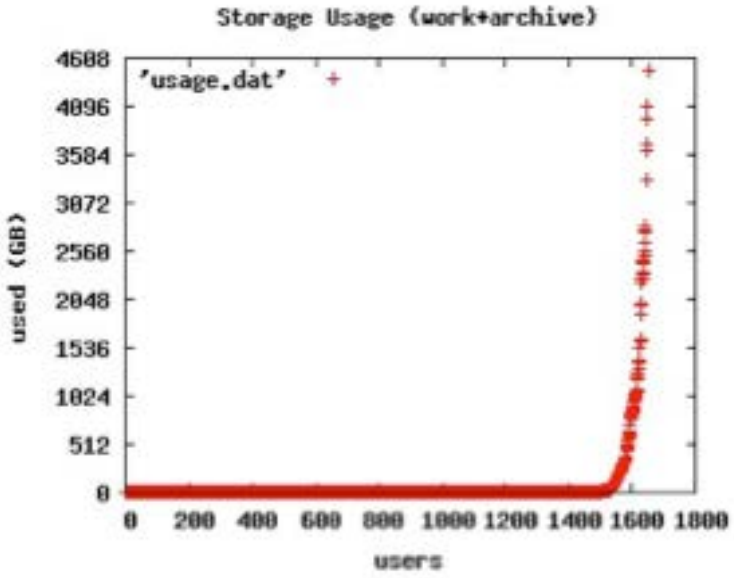


Hadoopの構成


MapReduce
submit
Job Tracker
Task Tracker
Task Tracker
Task Tracker
Task Tracker
Job
Job
Job
Job
node
node
node
node
node
Name Node
Data Node
Data Node
Data Node
Data Node
HDFS
Storage
Storage
Storage
Storage

スパコンのストレージの抱える問題

- 容量
  - 慢性的に不足傾向
    - 例. TSUBAME1
      - 0.2%のユーザが1TB以上を消費
      - データのライフサイクルを考慮する必要
- 性能
  - 様々なI/Oワークロードをサポート
    - コンカレントな並列I/O (c.f. MPI-IO)
    - チェックポイント，一時ファイル
    - Data-Intensive I/O



スパコンのストレージの抱える問題 (cont'd)

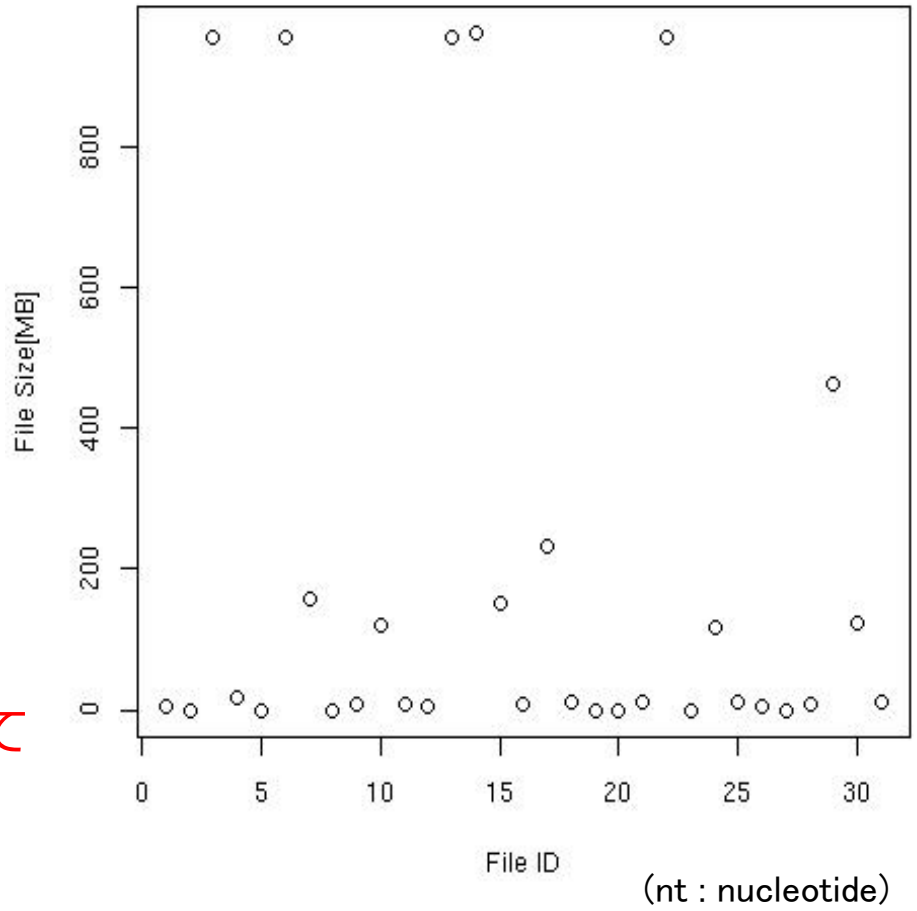
- 利便性
  - スパコンへのシームレスなデータアクセスの実現
    - PCや研究室のクラスタとのFederation
    - 学内へのファイルストレージサービス
  - 大規模データによるユーザロック
    - スパコン上へのデータのステージイン・アウト
    - 例. TSUBAME1上でのWebデータ解析
      - NICT (大阪) → TokyoTech@東京 : 初期データ2TBのステージイン
      - TokyoTech → NICT : 結果60TBのステージアウト
        - インターネット転送　8日間
        - Fedex

典型的なData-Intensive Applicationの Workload

- 例. BLAST(相同性検索)
  - 入力 → 計算 → 出力
  - ファイル全体のreadを複数に対して行う
  - 莫大な数の繰り返し
    - パラメタサーベイのために様々なクエリを試す
    - 異なるデータセットにも。。

同じプログラムを
異なるデータ及びパラメタに対して
繰り返し行う

→ Write once, Read mostlyな workloadといわれる所以

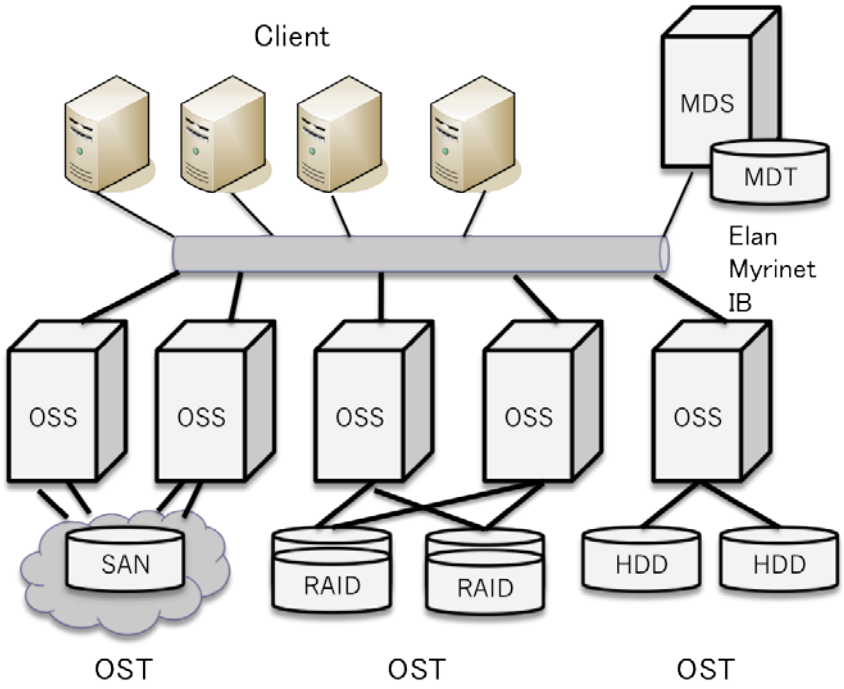


並列ファイルシステム

- Parallel File System
  - コンカレントな並列I/Oのスループット・レイテンシを重視
    - マスタ・スレーブモデルで、ファイルをストライピング(1MB程度)して格納
    - 計算ノードとストレージノードを高速なインターコネクトで接続
    - 大規模シミュレーション(MPIなど)に多いワークロード
    - 例. Lustre, GPFS, pNFS, PVFS
- Distributed File System
  - EP(Embarresing Parallel)なI/Oのスループットを重視
    - マスタ・スレーブモデルで、ファイルをストライピング(64MB程度)して格納
    - 計算ノードとローカルストレージを結合させた構造
    - 大規模データ処理に多いワークロード
    - 例. HDFS, GoogleFS, Gfarm

Parallel File System
例. Lustre (TSUBAME2の/workで利用)

- 構成
  - MDS
    - ファイルのメタデータを管理
    - MDTに実際のメタデータを格納
  - OSS
    - ClientからのI/O要求を処理
    - OST毎にストライプされる
    - OSTのストレージ構成は自由
  - インターコネクト
    - TSUBAME2の場合InfiniBand
- 特徴
  - ロック機構
    - MDS(MDT) : メタデータのロック
    - OST : オブジェクト(ファイル)のロック
  - 複製作成機能はない

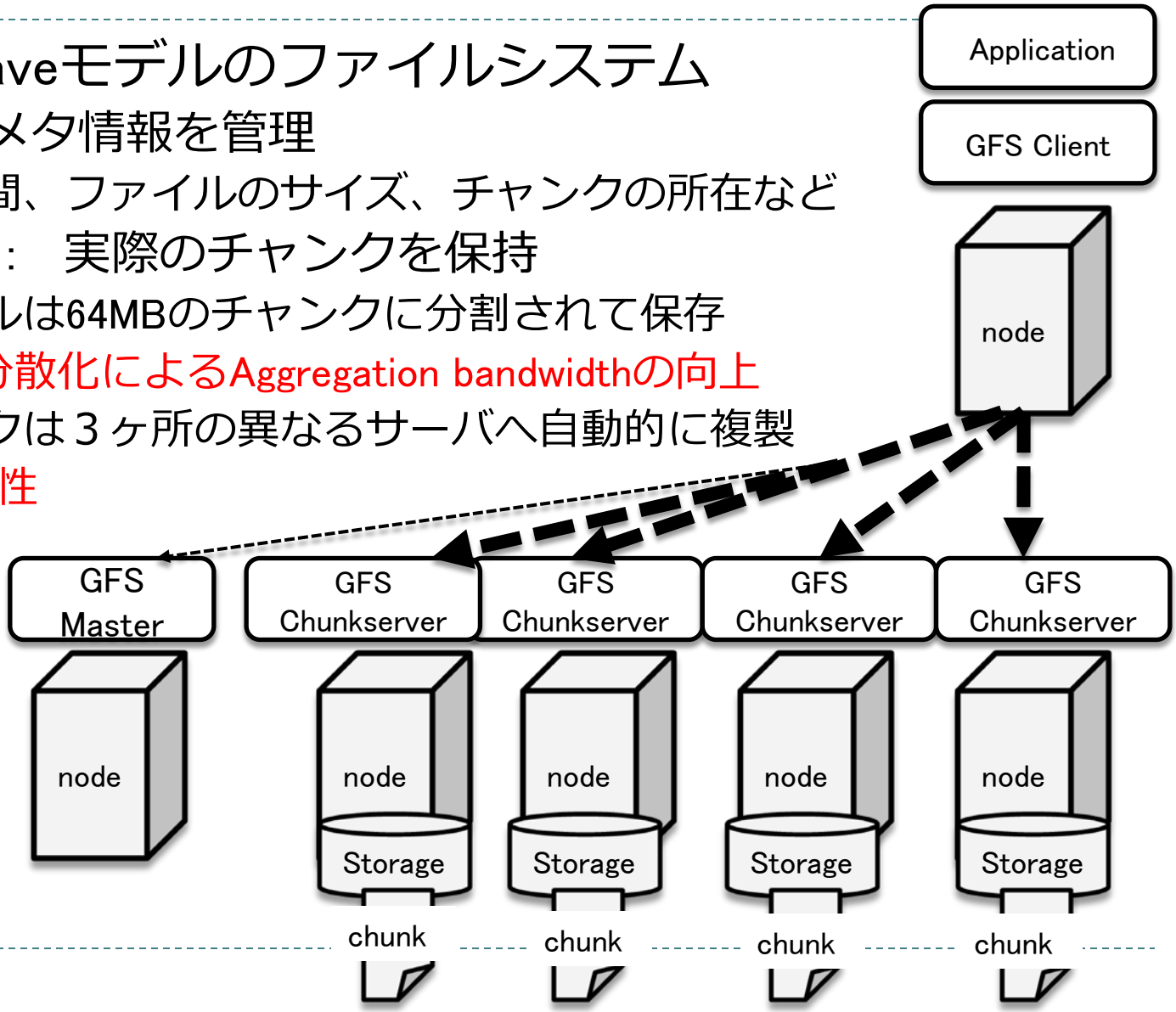


Distributed File System
例. Google File System (GFS)

- Master-Slaveモデルのファイルシステム
  - マスタ：メタ情報を管理
    - 名前空間、ファイルのサイズ、チャンクの所在など
  - スレーブ：　実際のチャンクを保持
    - ファイルは64MBのチャンクに分割されて保存

    → I/Oの分散化によるAggregation bandwidthの向上
    - チャンクは３ヶ所の異なるサーバへ自動的に複製

    → 耐故障性



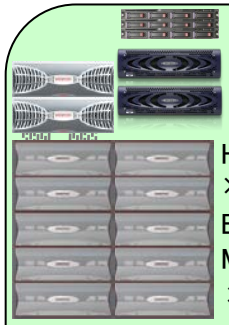
Distributed File System
例. Google File System (GFS) (cont'd)

- Write once, Read mostlyなワークロードに特化
  - 専用のファイル操作インターフェース (非POSIX)
    - Create, Delete, Open, Close, Read, Write,
      Snapshot, Record Append (非POSIX)
  - チャンクの読み出しはネットワーク的に近いサーバから行う
    - 同じマシン→同じスイッチ内のマシン など
    - I/Oの局所化
  - 書き込みはファイルの末尾へのappendのみサポート
    - 複数クライアントのファイルへの同時書き込みの非サポート
      - ロック機構がないので書き込む順番を保証できない
      - ファイルの末尾にレコードを1回以上書き込むことを保証

例．TSUBAME2.0のストレージ

TSUBAME2.0 Storage 11PB（7PB HDD, 4PB Tape）

Home



NFS, CIFS, iSCSI

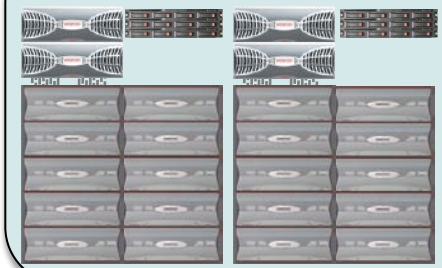
HP DL380 G6 × 4

BlueArc Mercury 100 × 2

1.2PB

DDN SFA 10K
2TB SATA × 600 disks

Phome



GPFS（GridScaler）

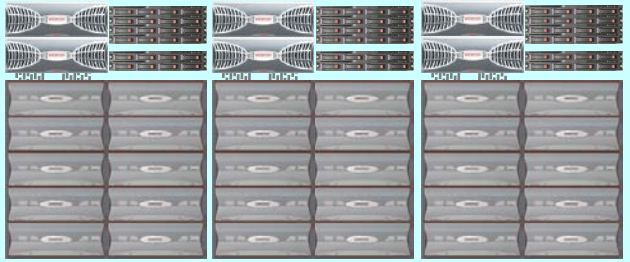
HP DL380 G6 × 4
HP DL360 G6 × 4

DDN SFA 10K × 2
2.4 PB
2TB SATA
× 2400 disks

HSM
with GPFS, Tivoli Storage Manager
2.4 PB HDD + ～4PB Tape

SL8500 × 2
4PB LTO4 5000 roles

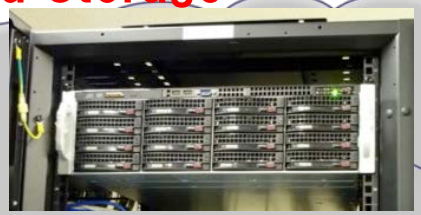
Work



Lustre
MDS : HP DL360 G6 × 6
OSS : HP DL360 G6 × 20
DDN SFA 10K × 3,
2TB SATA × 3550 disks, 600GB SAS × 50 disks

3.6 PB

Grid Storage



RENKEI-PoP

130 TB～

Scratch

Thin node

190 TB

SSD
60GB × 2 (120GB/node)
120GB × 2 (240GB/node)
RD 460MB/s WR 720MB/s

Medium and Fat node

SSD
120GB × 4 (480 GB/node)
RD 920 MB/s
WR 720MB/s

TSUBAME2.0 Storage Overview

TSUBAME2.0 Storage 11PB (7PB HDD, 4PB Tape)

Home

- Home for compute nodes
- Storage services for campus

NFS, CIFS, iSCSI

HP DL380 G6 × 4

BlueArc Mercury 100 × 2

DDN SFA 10K
2TB SATA × 600 disks

1.2PB

Phome

GPFS (GridScaler)

HP DL380 G6 × 4
HP DL360 G6 × 4
DDN SFA 10K × 2

2.4 PB
2TB SATA
× 2400 disks

Work

Concurrent Parallel I/O (e.g. MPI-IO)

Lustre
MDS : HP DL360 G6 × 6
OSS : HP DL360 G6 × 20
DDN SFA 10K × 3,
2TB SATA × 3550 disks, 600GB SAS × 50 disks

3.6 PB

Read mostly I/O (data-intensive apps, parallel workflow, parameter survey)

GPFS with GPFS, Tivoli Storage Manager

2.4 PB HDD + ～4PB Tape

Scratch

Thin node

SSD
60GB × 2 (120GB/node)
RD 460MB/s WR 720MB/s

190 TB

Medium and Fat node

SSD
120GB × 4 (480 GB/node)
RD 920MB/s WR 720MB/s

Fine-grain R/W I/O (check point, temporal files)

Grid Storage

Data transfer service between SCs/CCs

RENKEI-PoP

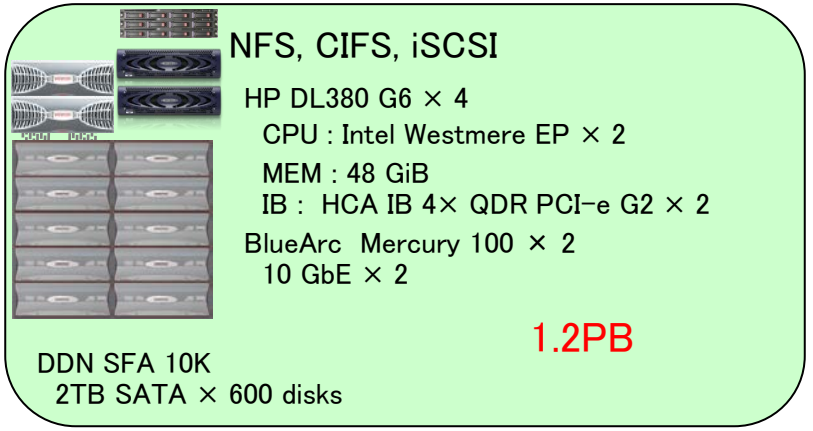
Backup

SL8500 × 2
4PB LTO4 5000 roles

30 TB～

Home Volumes

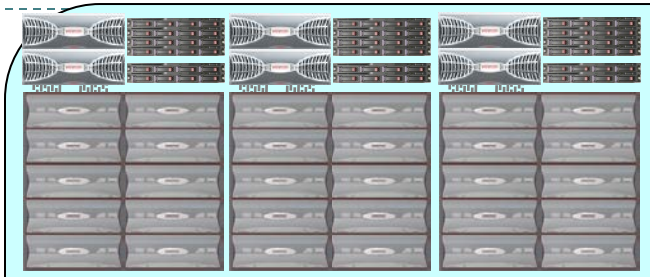
- Spec.
  - HP DL 380 G6 × 4 (GPFS)
  - BlueARC Mercury 100 × 2 (NFS, CIFS, iSCSI)
  - DDN SFA 10K
    - 2TB SATA × 600 disks (1.2 PB)
- Usage
  - User Home directories for compute nodes (cNFS over GPFS)
  - Campus storage services (CIFS)
  - Campus VM hosting services (iSCSI)



Parallel File System (Lustre)

Work

- Spec.
  - MDS : HP DL 360 G6 × 6
  - OSS : HP DL 360 G6 × 20
  - DDN SFA 10K
    - 2TB SATA × 3550 disks +
      600 GB SAS × 50 disks (3.6 PB)



3.6 PB

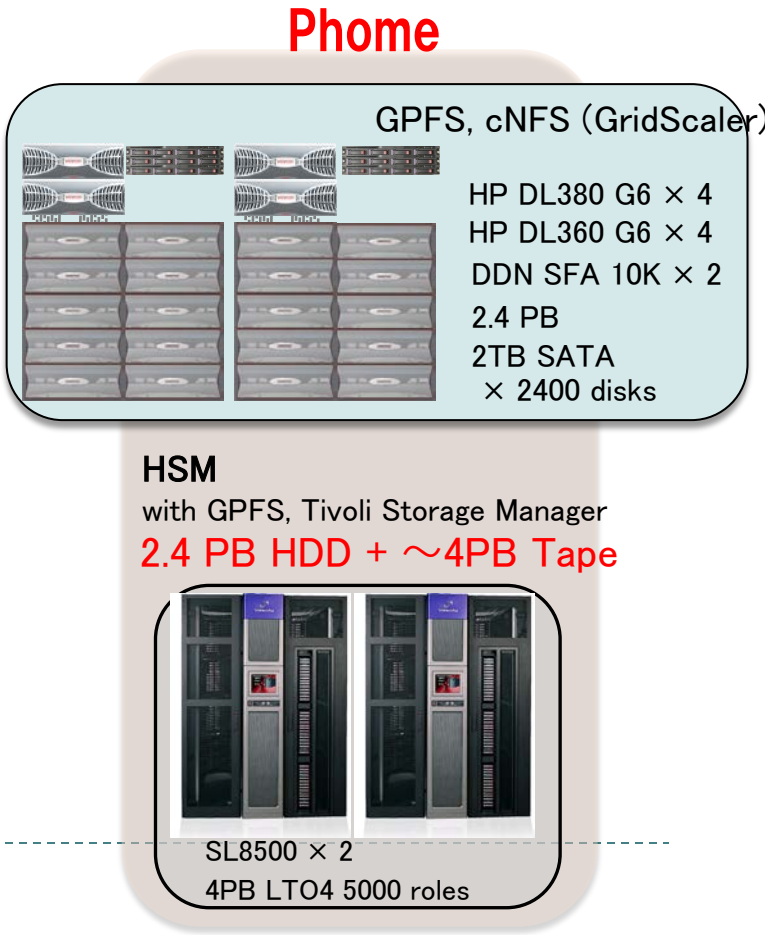
Lustre

MDS/OSS : HP DL360 G6 × 6
CPU : Intel Westmere EP × 2 (12 cores)
MEM : 48 GiB (MDS), 24 GiB (OSS)
IB : HCA IB 4× QDR PCI-e G2 × 1 (MDS), × 2 (OSS)
DDN SFA 10K × 3,
2TB SATA × 3550 disks, 600GB SAS × 50 disks

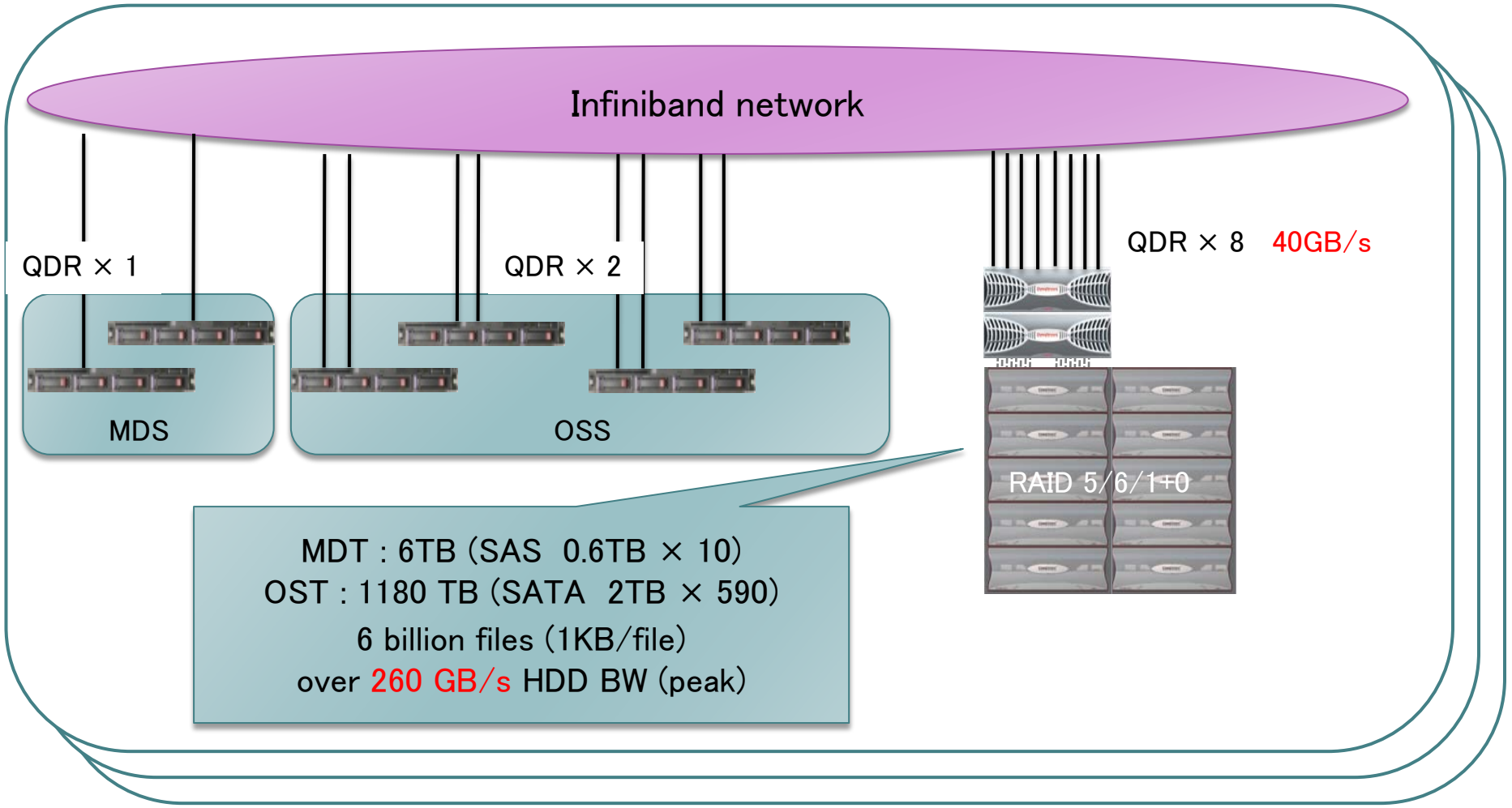
- Usage
  - Concurrent Parallel R/W I/O
    - e.g. MPI-IO

Parallel File System (GPFS)

- Spec.
  - Servers : HP DL 380 G6 × 4, HP DL 360 G6 × 4
  - DDN SFA 10K
    - 2TB SATA × 2400 disks (3.6 PB)
  - StorageTek SL8500 × 2
    - 4PB LTO5 × 5000 roles
    - HSM
      - with GPFS ILM, Tivoli Storage Manager
- Usage
  - READ mostly I/O
    - Data-Intensive apps,
      parallel workflow, parameter survey

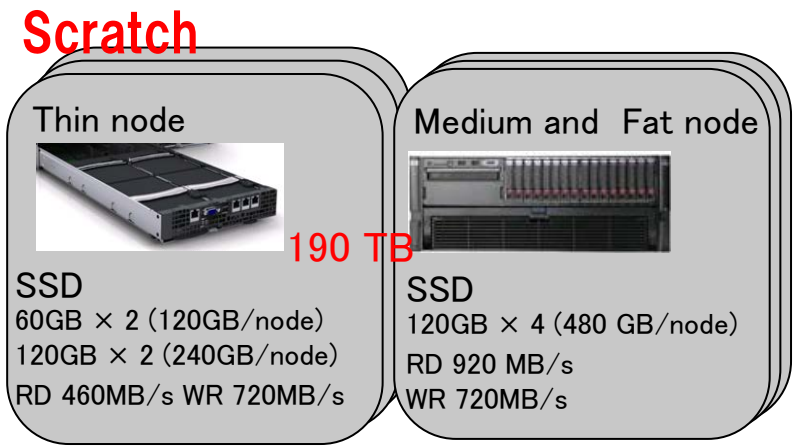


Parallel File System Configuration (Lustre)



SSDs

- Spec.
  - Thin node
    - 60/120 GB × 2 (120/240 GB/node)
    - Read 460 MB/s, Write 720 MB/s
  - Medium/Fat node
    - 120 GB × 4 (480 GB/node)
    - Read 920 MB/s, Write 720 MB/s
- Usage
  - Fine-grain R/W I/O
    - e.g.) check point, temporal files
  - READ mostly I/O
    - e.g.) MapReduce

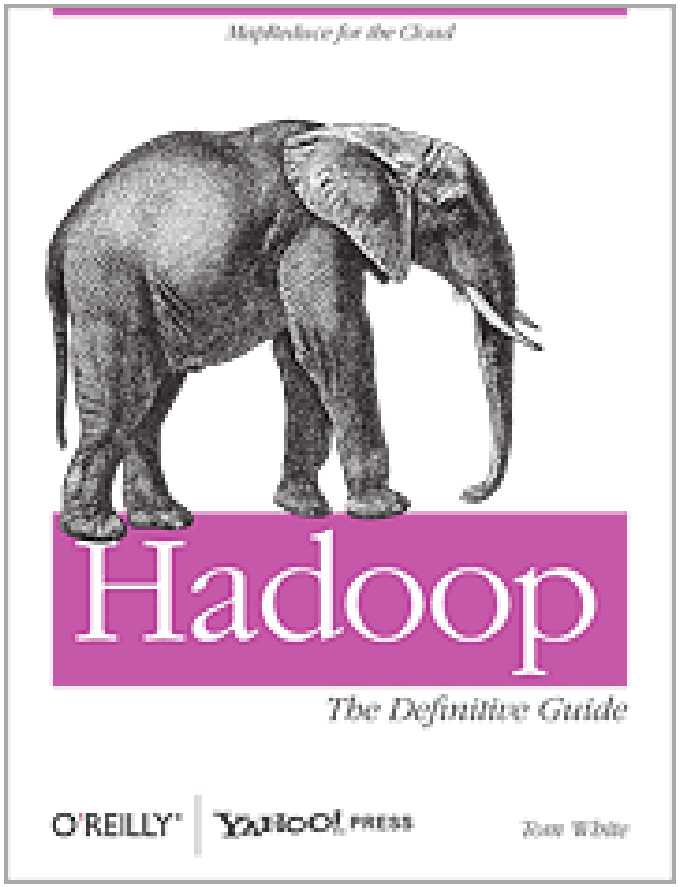


参考文献

- MapReduce: Simplified Data Processing on Large Clusters
  - Jeffrey Dean et al.
  - http://labs.google.com/papers/mapreduce.html
- The Google File System
  - Ghemawat et al.
  - http://labs.google.com/papers/gfs.html

参考文献 (cont’d)

- タイトル　Hadoop: The Definitive Guide
  著者 Tom Whilte
  出版元 O’Reilly Media



次回以降の予定

- 7/15(月・祝)
  - 特別授業・課題なし
- 7/22(月)
  - Map-Reduceプログラミングその2
    - Hadoopのソフトウェアの構造
    - Hadoopを用いたMapReduceプログラミング
    - Map-Reduce編課題説明　(8/8〆切予定)