The Google File System

柏原秀蔵

<注>

各スライドタイトル末尾の括弧内の番号は論文の

章を示しています

GFSとは

* GFS: Google File System
    * スケーラブルな分散ファイルシステム
    * 可用性・信頼性・効率性を安価なハードウェアで実現
* GFSクラスタ（2003年）
    * 1000ノード
    * ストレージ：300TB
    * クライアント：数百台

Googleが直面した問題

* 故障・障害への対応
* 巨大なファイル数の増大

GFSの設計(2.1)

* 一般的（安価）なハードウェアから構成
* 障害検出、復旧をソフトウェアで実現
* ランダムアクセスより、シーケンシャルアクセスが多い
* 数100MB〜GB単位のファイルを効率的に扱う
* 同一ファイルへの**並列の追記書き込み**に対応
* レスポンスより転送速度を優先

脇道：なぜ追記が多い？

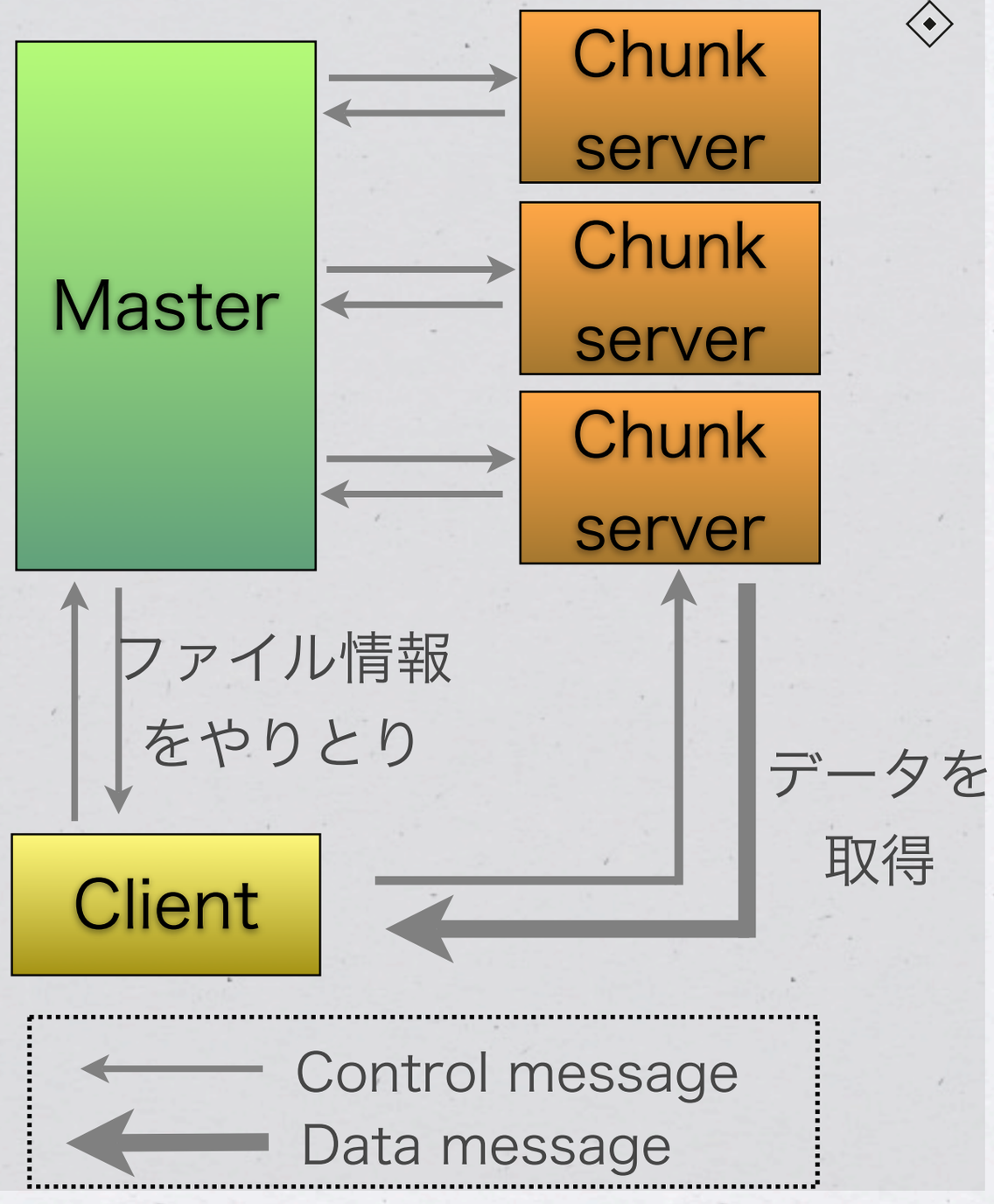
* 論文では、追記が多い理由を記述していない
* ウェブを検索しても論文をまとめた文書ばかり…
* 『Googleを支える技術』西田圭介 著
  * 「膨大なデータの通り道となる」第3章 p64
  * 検索エンジンのクローリング・インデックス作成で生成されたデータがデータ膨大で、追記が多いのではないかと説明

インタフェース(2.2)

* ファイルの名前空間
  * ファイル名・ディレクトリを使った階層型（例：/dir/foo..)
* 提供するファイル操作
  * ファイル作成、オープン、書き込み、読み込み、クローズ等
  * Snapshot: ファイル・ディレクトリを効率的にコピーする
  * Record Append: ファイルへの追記書き込み

アーキテクチャ(2.3)

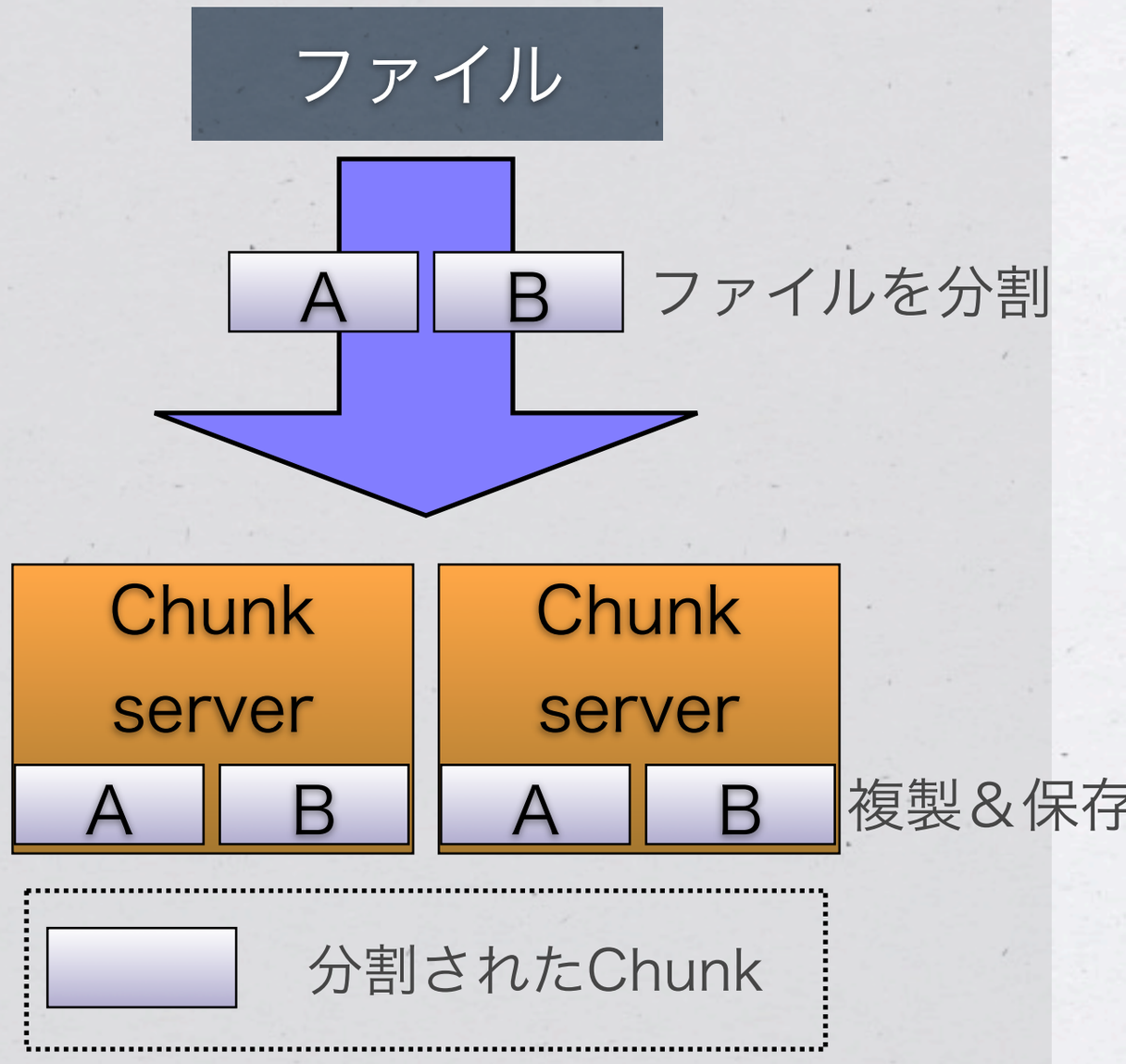
* Master(1台)
* Chunkserver(複数)
* クライアント(複数)
  * GFSを利用するアプリケーション



Chunkとは(2.3)

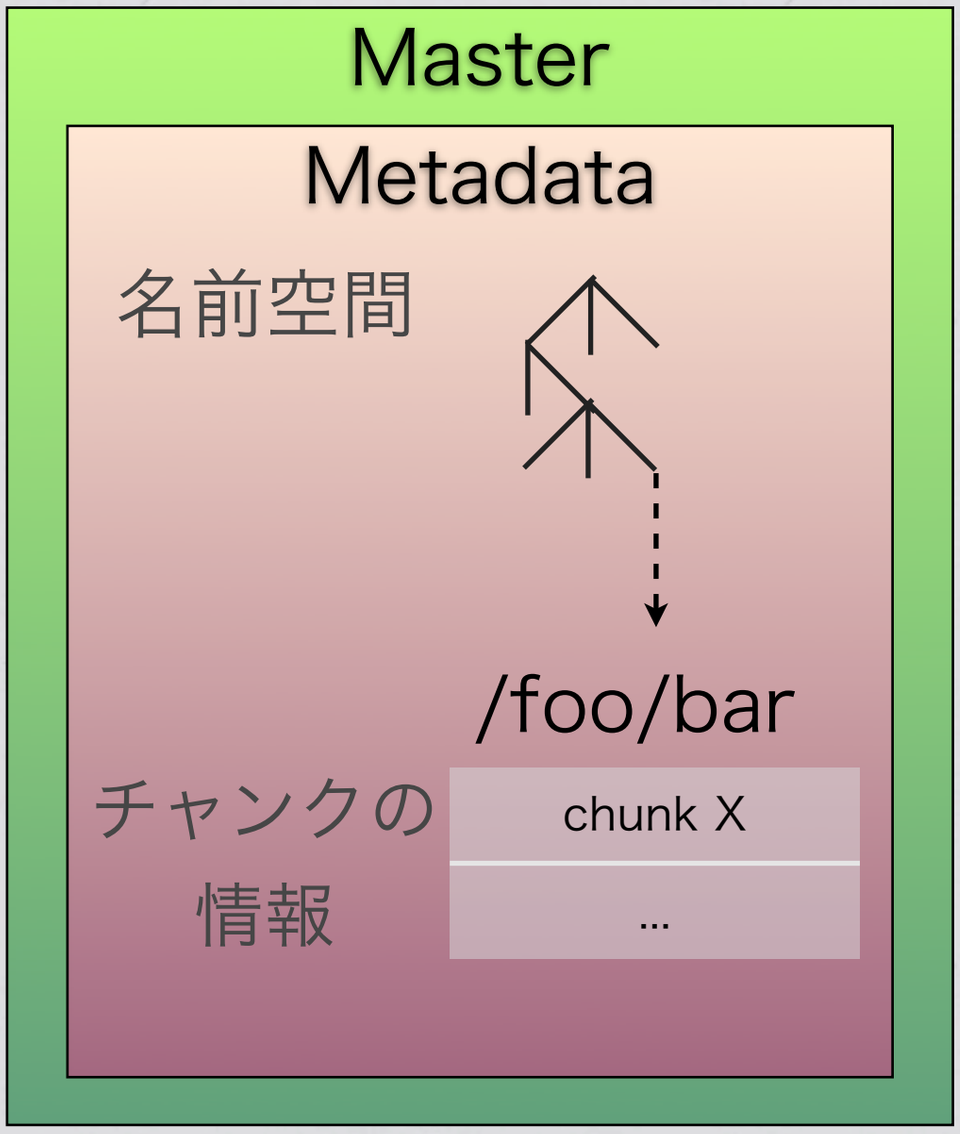
* 特徴
  * ファイルが分割された塊(64MB固定)
  * Chunkserverに保存される（自動で3以上複製される）
  * ファイル情報はMaster側で保持（Chunk識別用のID等）
* 目的：効率性・ボトルネックを減しつつ、信頼性と可用性を！

ファイルを分割するイメージ



Master(2.3, 2.4)

* Master
  * 1台, Metadataをメモリ上に保持
* Metadata
  * ファイル・名前空間の情報
    （ファイルシステム）
  * ファイルとChunkのマッピング
  * Chunkの複製がある
    Chunkserverの情報



Chunkの場所情報(2.6.2)

* Chunkの場所情報
  * MasterのMetadata（メモリ上）で保持
    * Masterのディスクには保存されない（Chunkserverで保存
* Chunkserverが起動時・起動中にMasterへ場所を通知
  * Masterは最新のChunkserverの場所が分かる
  * Chunkserverのアドレスが変わっても、Masterの情報は更新されていく仕組み

Operation Log(2.6.3)

* GFSのジャーナリング機構
    * Masterのファイル情報の破損を防ぐ（例：ファイルを操作している途中でMasterがクラッシュ）
    * GFS(ファイルシステム)を操作した履歴の差分を全て保存
* 最適化
    * ログのサイズを小さくするため、定期的にチェックポイントという区切りを作成する
    * Metadataの復元に必要な時間を減らす工夫

書き込みのふるまい(3.1)

* クライアントは1台にだけデータを送信
* データはChunkserverをパイプライン方式で転送される
* Primaryと呼ばれるChunkserverが、書き込みを指示する
* 複数クライアントからの同時書き込みの安全性は保証されない

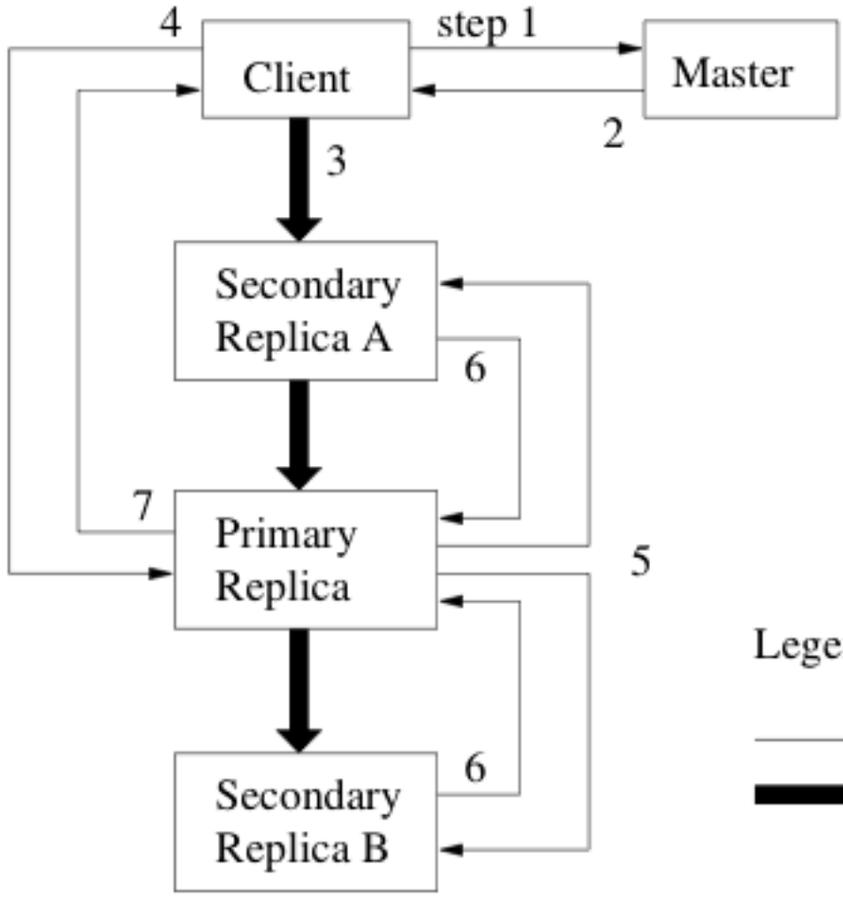
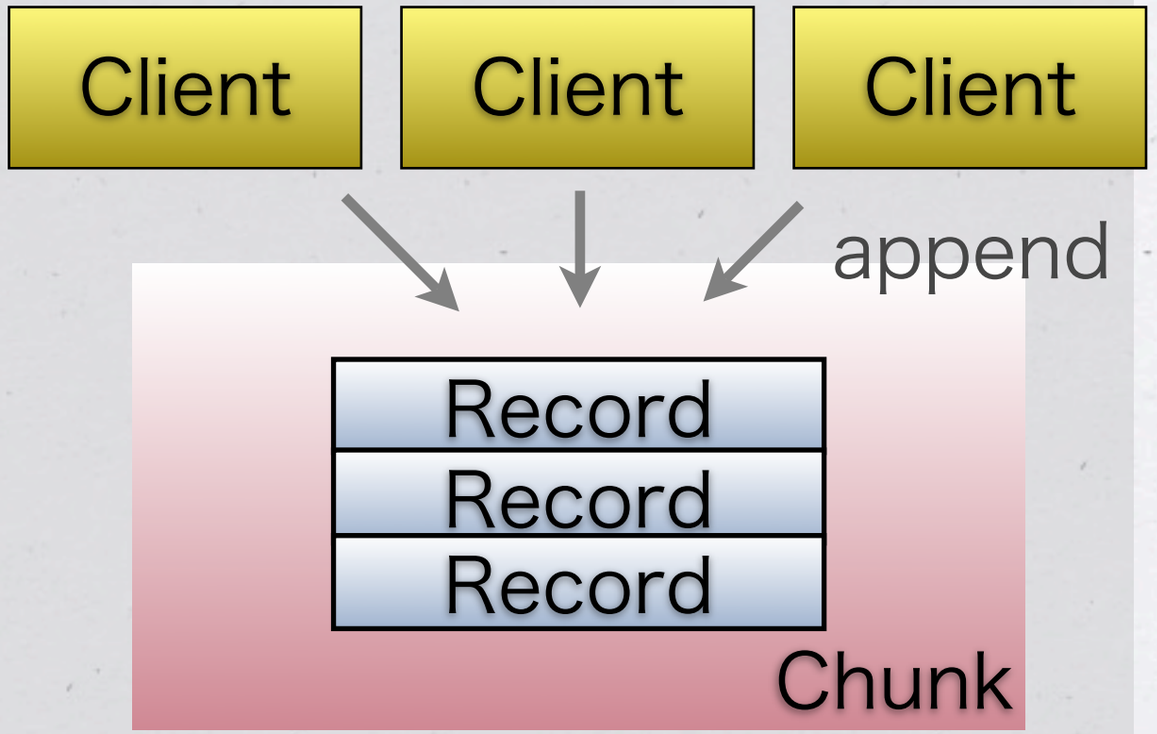


Figure 2: Write Control and Data Flow

追記書き込み(3.3)

* 一般的な実装：ファイルをロックして...
* GFS：Queueを使う
  * 複数Clientから同時に要求を受けることができる
  * 書き込み自体はAtomic
* Record: 書き込むデータ
* 追記はAtomicな操作
  * 同一ファイルへの追記は、同時に複数行われることはない



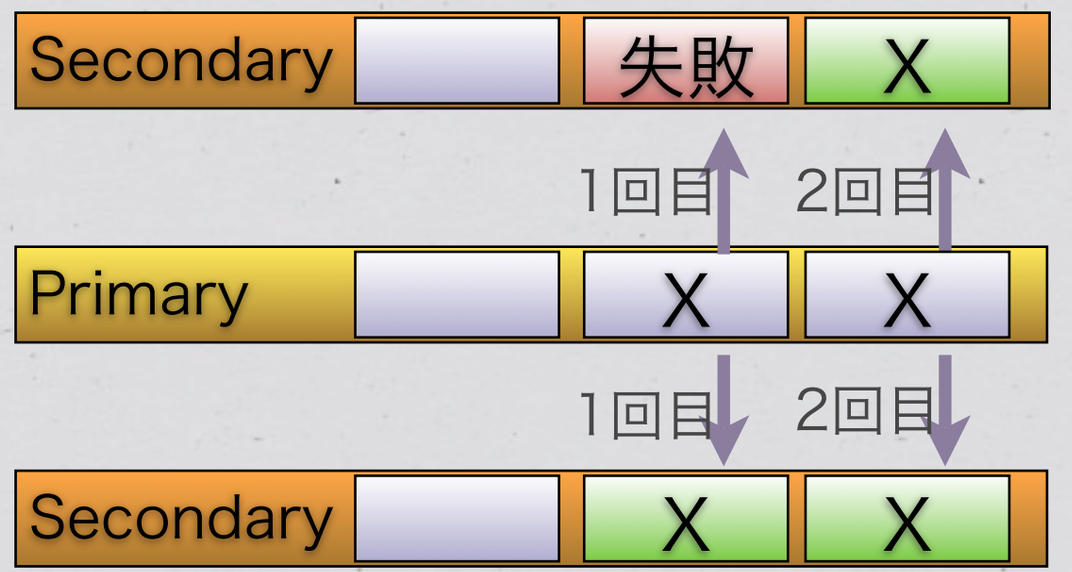
追記書き込み -続き-(3.3)

* 追記（Append）の特徴
    * 成功時：1回以上書き込まれることを保証
    * 失敗時：再度試す（あるサーバのChunkには追記完了していたとしても、再度繰り返すので複数書き込まれる事がある）
* 一貫性を保証しない
    * Chunkの複製はすべて同一であること

追記で一貫性を保証できない理由

* 成功した時
  * Chunkへの追記がすべて成功(一貫性OK)
* 失敗した時（2通り）
  * 複製の全てへ追記失敗(一貫性OK)
  * **複製の一部へ追記成功、一部は失敗 (一貫性に問題有)**

1回目：複製の一部へ追記成功、一部は失敗
2回目：追記成功



GFSの一貫性(2.7)

consistent	どの複製を読むかに関わらず、全てのクライアントから同じデータが見える状態のこと。一貫性がある
defined	ファイルの書き込み後、consistentな状態かつ変更した箇所のデータが反映されている状態
inconsistent	読み出すChunkの場所（Chunkserver）によって、見えるデータが異なることがある状態
undefined	書き込みに成功していても、変更がChunkに反映されていない（書いたはずのデータが無い！）

操作とその一貫性(2.7)

	Write	Record Append
直列 成功	defined	defined 部分的に inconsistent(になる可能性)
並列 成功	consistent undefined	
失敗	inconsistent	

2.7 Consistency Model: Table 1より

Masterの役割(4)

* 名前空間の管理(4.1) <-- 次のスライドで詳細に説明
* 複製の配置(4.2)
* ファイル作成、再複製、再バランシング(4.3)
  * Chunkの複製の数が足りなかったときの対応
  * ディスク容量に空きのあるChunkserverへChunkを自動的に移動
* ガベージコレクション(4.4)
  * 参照されなかったり、古くなってしまったChunkを自動的に削除

名前空間の管理とロック(4.1)

* 特徴（Master内部）
  * ディレクトリごとに構造体を持っているわけではない
  * ハードリンク・シンボリックリンクは非対応
* ロック(read-write lock)
  * アクセスの時：ディレクトリを read lock + フルパスでlock(r/w)

データ構造（表に変換）

/dir/foo	chunk X
/dir/foo2	
...	....

ロック

ディレクトリごと削除されるのを防ぐ

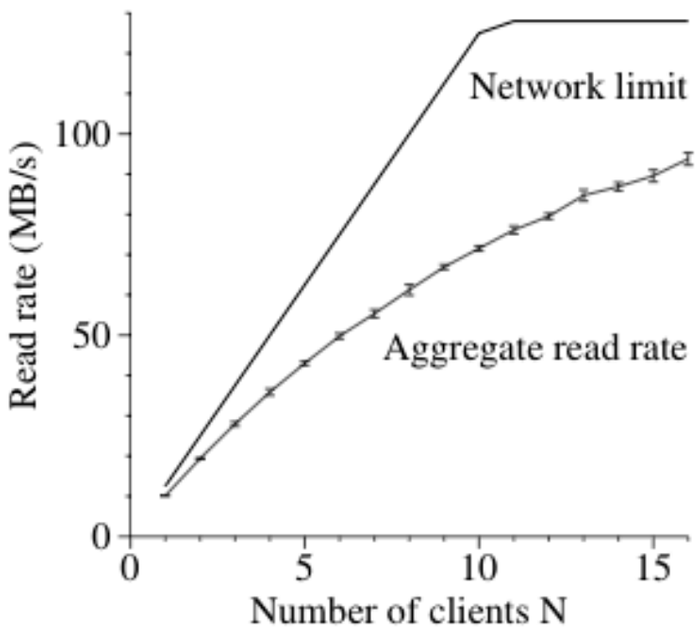
耐障害性(5.1)

* 短時間の復旧(5.1.1)
* Chunkの複製 (5.1.2)
* Masterの複製 (5.1.3)
    * オペレーションログ, チェックポイント
* チェックサムによるデータ破損の確認(5.2)
* デバッグ・パフォーマンス診断用にログを保存(5.3)

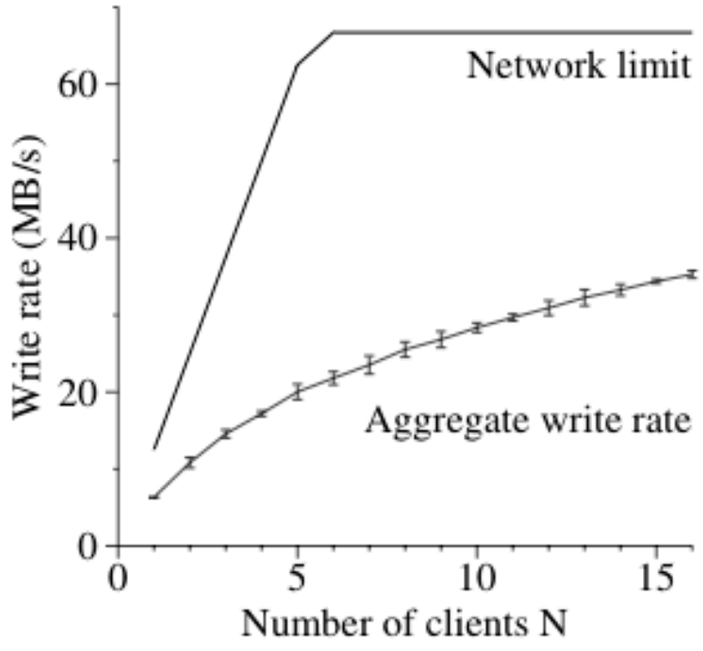
測定(6)

* マイクロベンチマーク(6.1)
  * Pen III 1.2GHz, RAM 2GB, HDD 80GB(2つ)
  * Master 1台 (+複製2台)
  * クライアント/Chunkserver： 16台
* 実クラスタ(6.2)
  * ClusterA: 数百のエンジニアで、研究・開発に使用
  * ClusterB: Google製品のデータ処理

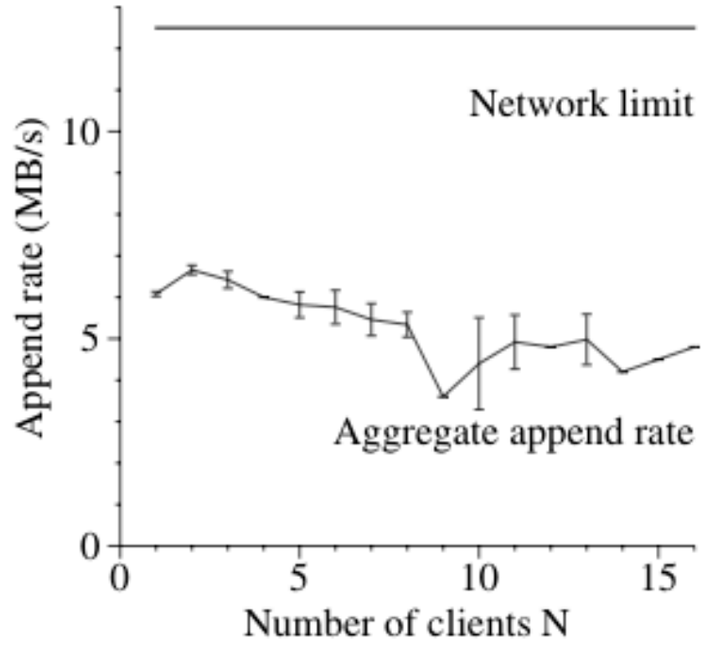
ベンチマーク(6.1)



(a) Reads



(b) Writes



(c) Record appends

実クラスタでの結果(6.3)

* ストレージ
  * TB単位のディスク容量にも対応
* Metadata
  * サイズも十分小さかった：平均1ファイルあたり約100バイト
  * 短時間（数秒のディスク読み込み）で復元可能だった
* Masterの負荷
  * 200〜500 operations/secでボトルネックにならず

まとめ(6, 9)

* Googleの欲していたシステムの実現
    * 安価なハードウェア
    * ディスク容量のスケール
    * 障害の自動検出・耐障害性
    * データ処理向けの効率性