The Google File System

09M37229 松岡研 野村 達雄

開始

出典

- The Google File System
  - S.Ghemawat, H.Gobioff, S.Leung
  - Proceedings of the 19th ACM Symposium on Operating Systems Principles
  - 2003

背景

- Googleの処理すべきデータ量の急激な増大
- 一般的なファイルシステムでは効率が悪い
- Googleの環境に特化したファイルシステムが必要

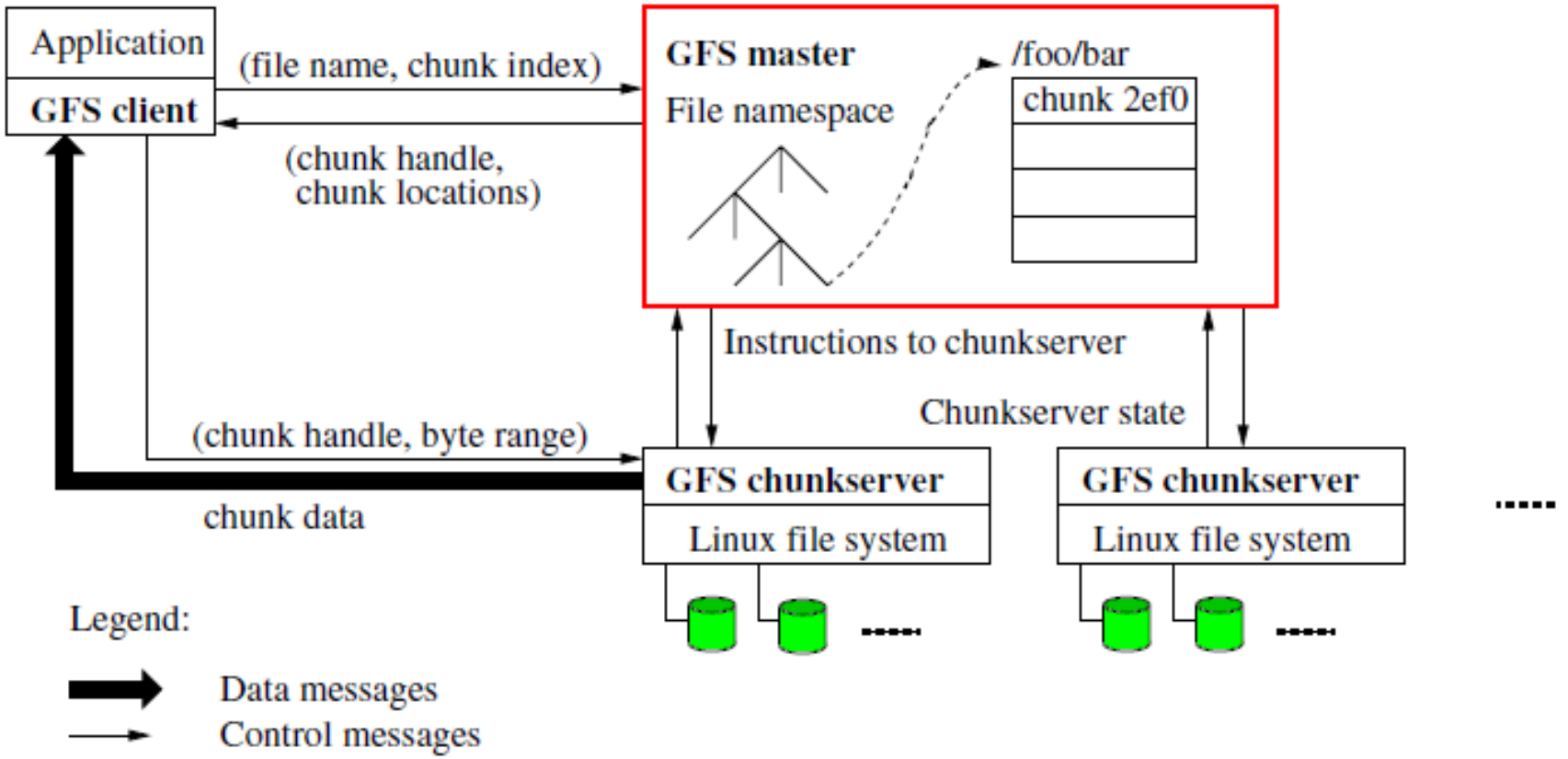
想定される環境

- 多数の安価なマシンで構成されるシステムで、マシンが頻繁に壊れる
- 一つのファイルが数百MB～数GB
- 読み込みは**シーケンシャルアクセスがメイン**
- 書き込みはファイルへの**追記がメイン**
- 一度書かれたファイルはめったに変更されない
- 一つのファイルへ同時に多数の追加書き込みが発生する

インターフェイス

- *create delete open close read write*
- 一般的なファイルシステムと似ているが、POSIX標準のAPIは提供しない
- *snapshot*と*append*コマンドを提供（後のスライドで説明）

アーキテクチャ


Application
GFS client
(file name, chunk index)
(chunk handle,
chunk locations)
GFS master
File namespace
/foo/bar
chunk 2ef0
Instructions to chunkserver
Chunkserver state
(chunk handle, byte range)
chunk data
GFS chunkserver
Linux file system
GFS chunkserver
Linux file system
Legend:
Data messages
Control messages

マスターサーバ

- マスターは1台のみである
  - デザインがシンプルである
  - 複雑なチャンク配置が可能である

- ファイルのメタデータのみを保持
- ファイルの実データは直接チャンクサーバとやり取りする

チャンクサーバ

- ファイルを構成するチャンクを保持
- チャンクはLazy space allocationで確保
- 1つのチャンクは64MBのLinuxの通常ファイル
- 64MBより小さいファイルの場合はそれがHot spotになり得るが、実用上は問題になっていない

メタデータ

- マスターは主に3種類のメタデータをメモリ上に保持している
  1. ファイルとチャンクのネームスペース
  2. ファイルからチャンクへのマッピング
  3. チャンクのレプリカの位置
- マスターは1と2を永続的な形で保持している。
- 3はマスターの起動時やチャンクサーバが新たに加わったとき等にチャンクサーバに問い合わせる

メタデータ

- メモリ上に保持する利点
  - マスターの動作が速い
  - バックグラウンドでメタデータをスキャンするような操作が容易にできる（GCやRe-replicationなど）
  - 扱えるファイル数を増やすには単純にマスターのメモリを増設するだけでよい
- チャンク毎メタデータは64byte以下であり、ネームスペースはプレフィックス圧縮をするためさらにサイズが小さい

チャンクの場所

- マスターはチャンクのレプリカの位置を永続的には保持せずに、起動時に**チャンクサーバに問い合わせる**。
- マスターは定期的にチャンクサーバとメッセージをやりとりし、チャンクサーバを監視する
- 最終的にどのチャンクがディスク上に存在しているかを知っているのはチャンクサーバであるため、一貫性をほじするためにもマスターは永続的なデータを保持しない

操作ログ

- いわゆるジャーナルのこと
- チャンクへの操作を論理時間順に記録
- ログがあるサイズに達したらマスターはチェックポイントを**ローカルディスクとリモートディスクに保存**
- チェックポイントは時間が掛かるため、別スレッドで実行する
- マスターはチェックポイントをとっておき、障害に備える

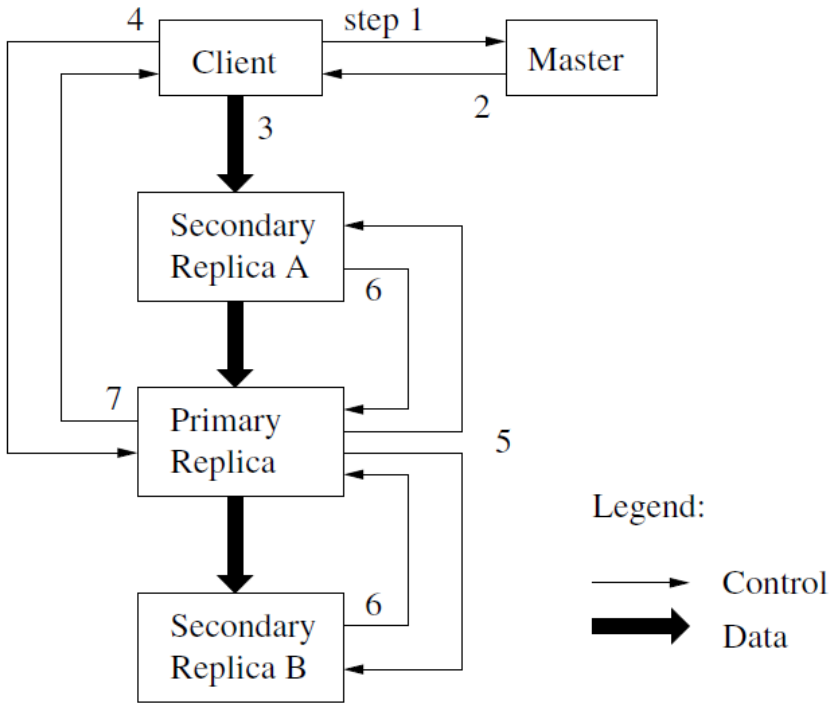
一貫性モデル

- GFSは緩い一貫性しか提供しない
- ***consistent***: レプリカによらず、クライアントは常に同じデータが見える
- ***defined***: consistentであり、ファイル操作後にその操作の結果が反映されている

	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

Interactions

- GFSはなるべくマスターと他のやり取りが少なくなるよう設計されている
- データはA->B, B->Cのようなチェーンで近いサーバを伝って転送される
- データ転送はパイプライン方式で行われる



追記とスナップショット

- 追記では1回以上データが書き込まれることを保障
- 失敗したデータは取り除かずにさらに追記する
- データへの追記はキューを経由して行われる
- データのconsistencyはアプリケーション側で担保
- スナップショットはCopy on writeで高速にとれる

ロック機構

- GFSはディレクトリ構造をもたずに、フルパスをキーにメタデータをルックアップする
- /d1/d2/…/dn/**leaf** へのロックを取得するには
  1. /d1, /d1/d2, /d1/…/dnへのリードロックを取得する
  2. /d1/d2/…/dn/leafへの読み込み/書き込みロックを取得する
- 同じディレクトリに対して同時に複数のファイルを作成できる（ディレクトリへの読み込みロックと、ファイルへの書き込みロック）
- ロックを辞書順に取得するようにすることでデッドロックを回避する

レプリケーション

- レプリカは物理的に離れたラックのチャンクサーバに保存される
- マスターはレプリカの数が足りなくなる（チャンクサーバのダウンなどで）とすぐにレプリカを新たに他につくる
- マスターは定期的にサーバ間のバランスを取るためにレプリカを再配置する

Garbage Collection

- GFSはファイルの削除が実行されてもデータはすぐには消されず、定期的にGCが動いてディスクスペースを空ける
- 削除が実行されると、実際にはファイルはタイムスタンプを含む別名にリネームされる
- マスターの定期スキャン時に削除済みファイルが3日以上経過していたらメタデータとチャンクを削除する
- マスターとチャンクサーバは定期的に通信し、孤立したメタデータから参照されないチャンクを削除する

対故障性

- グーグルのシステムは多数の安価なマシンで構成されているため、マシンの故障が頻繁におこる
- マスターとチャンクサーバはどんな理由で停止したかに関わらず、数秒以内に起動できる
- チャンクのレプリケーション
- マスターのレプリケーション
- シャドウマスター（読み込み専用）

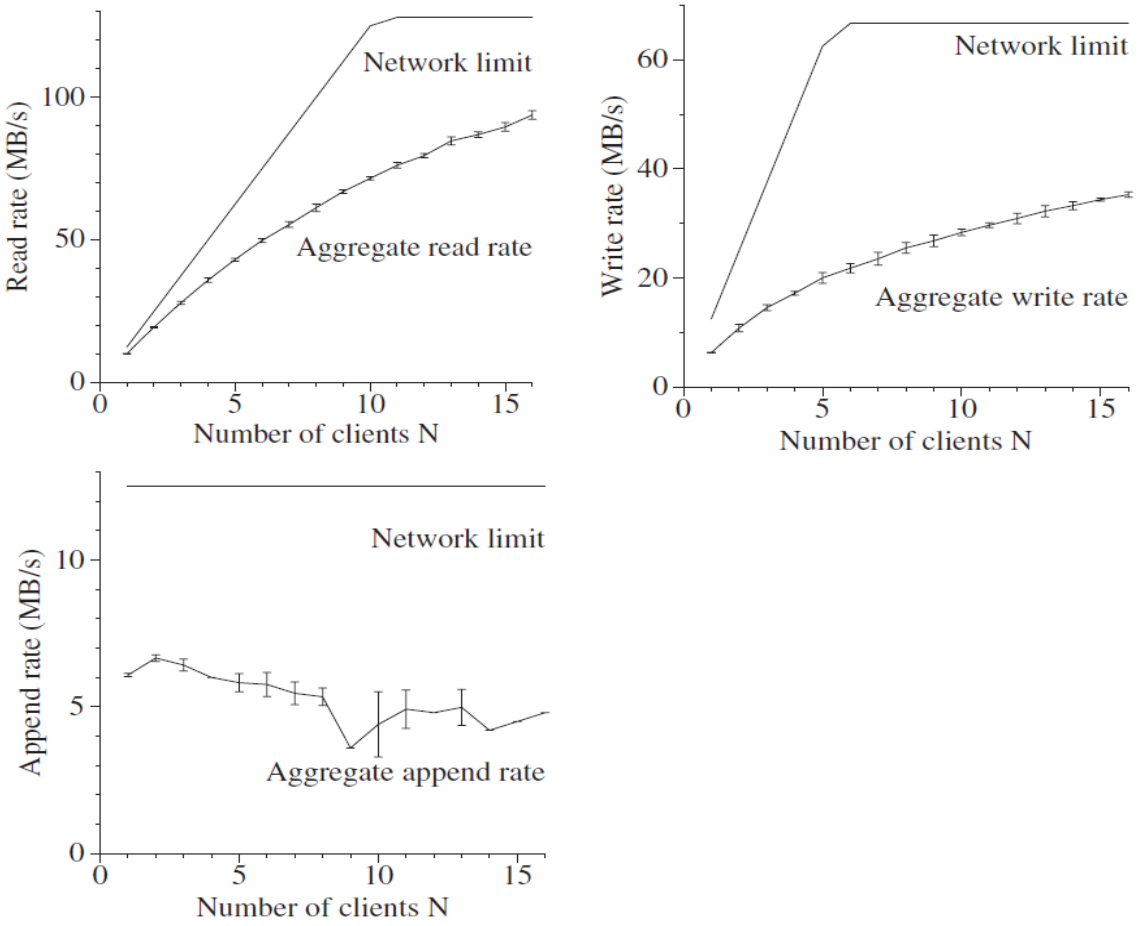
対故障性

- チャンクのチェックサム
  - 1つのチャンクは64KBのブロックからなる
  - 各ブロックには32bitのチェックサムが付いている
  - チェックサムはメモリ上（と操作ログ）に保持されている
  - チャンクサーバはデータを返す前にチェックサムを検査する
- チャンクサーバはアイドル中にチャンクをスキャンする
- GFSは常にあらゆる操作に対してログをとっている

マイクロベンチマーク

- 測定環境
  - マスター　　x1
  - マスターレプリカ　　x2
  - チャンクサーバ　　x16
  - クライアント　　x16
  - CPU dual 1.4GHz PIII processors
  - Memory 2GB
  - 80GB 5400rpm HDD x2
  - クライアントとサーバはそれぞれ100Mbpsでスイッチに接続
  - スイッチ間は1Gbpsで接続

マイクロベンチマーク



(c) Record appends

Real World Clusters

- **A**: 開発者が日常的に使用するクラスタ
- **B**: プロダクションが稼動しているクラスタ
- 稼動開始から一週間後のデータ

Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

Workload Breakdown

- **X**: 開発者が日常的に使用するクラスタ
- **Y**: プロダクションが稼動しいているクラスタ

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
0K	0.4	2.6	0	0	0	0
1B..1K	0.1	4.1	6.6	4.9	0.2	9.2
1K..8K	65.2	38.5	0.4	1.0	18.9	15.2
8K..64K	29.9	45.1	17.8	43.0	78.0	2.8
64K..128K	0.1	0.7	2.3	1.9	< .1	4.3
128K..256K	0.2	0.3	31.6	0.4	< .1	10.6
256K..512K	0.1	0.1	4.2	7.7	< .1	31.2
512K..1M	3.9	6.9	35.5	28.7	2.2	25.5
1M..inf	0.1	1.8	1.5	12.3	0.7	2.2

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
1B..1K	< .1	< .1	< .1	< .1	< .1	< .1
1K..8K	13.8	3.9	< .1	< .1	< .1	0.1
8K..64K	11.4	9.3	2.4	5.9	2.3	0.3
64K..128K	0.3	0.7	0.3	0.3	22.7	1.2
128K..256K	0.8	0.6	16.5	0.2	< .1	5.8
256K..512K	1.4	0.3	3.4	7.7	< .1	38.4
512K..1M	65.9	55.1	74.1	58.0	.1	46.8
1M..inf	6.4	30.1	3.3	28.0	53.9	7.4

Cluster	X	Y
Open	26.1	16.3
Delete	0.7	1.5
FindLocation	64.3	65.8
FindLeaseHolder	7.8	13.4
FindMatchingFiles	0.6	2.2
All other combined	0.5	0.8

結論

- 安価なマシンを大量に使った分散システム用のファイルシステムを開発した
- Googleの環境に適したファイルシステム
- 高い分散性と対故障性を実現した

- 用途をよく考えており、複雑な部分を極力省いたエレガントな設計
- 現在のGFSがどうなっているかが気になる