2012 年 5 月 21 日 情報システム基盤学合同輪講発表資料

Cloud MapReduce:a MapReduce Implementation on top of a Cloud Operating System

著者： Huan Liu and Dan Orban
出典： *11th IEEE/ACM International Symposium on Cluster,Cloud and Grid Computing,PP.464-474,2011.*
発表者： 本多・近藤研究室 1253011 沈　峻

1　概要

伝統的なオペレーティングシステム（OS）と同様に、クラウド OS は、低レベルのクラウドリソースを管理し、インフラストラクチャの詳細情報を隠すために、アプリケーションプログラマーに高レベルのインタフェースを提供する。しかし、従来の OS とは異なり、クラウド OS は、そのサービスをスケーラブルにするために複雑である、

本研究では、Amazon のクラウド OS の上に MapReduce のプログラミングモデル CloudMapReduce（CMR）実装して、評価する。CMR は、クラウド OS の上に構築することにより、システムの設計と実装を簡約化することが可能であることを示し、horizontal scaling と eventual consistency[2] といったクラウド OS のリミットを克服する方法を発見するのを目的としている。

2　クラウド OS

クラウド OS が提供しているサービスの視点から見ると、伝統的な OS と同じである。

表 1: 伝統的な OS と Cloud　OS の対比

伝統的な OS（windows os）	Amazon Cloud OS
Compute services	Virtual Machines(Amazon EC2)
Storage services	Amazon S3
Communication services	Amazon ’ s Simple Queue Service
Persistent storage services	Amazon ’ s SimpleDB

サーバ OS に比べて、クラウド OS の主要な違いは、スケーラブルであるということである。

3　従来の MapReduce

MapReduce は、巨大なデータセットを持つ高度に並列可能な問題に対して、並列処理させるためのフレームワークである。そのアーキテクチャを図 1 に示す.

Map ステップ - マスターノードは、入力データを受け取り、それをより細かい単位に分割し、複数のワーカーノードに配置する。

Reduce ステップ - 続いて、マスターノードが、Map ステップでの処理結果を集約し、目的としていた問題に対する答え（結果）を何らかの方法によって出力する。

図 1: MapReduce の実行の概要 [1]

4　Cloud MapReduce

Amazon Cloud OS を用いて CMR を提案した。
そのアーキテクチャを図 2 に示す.

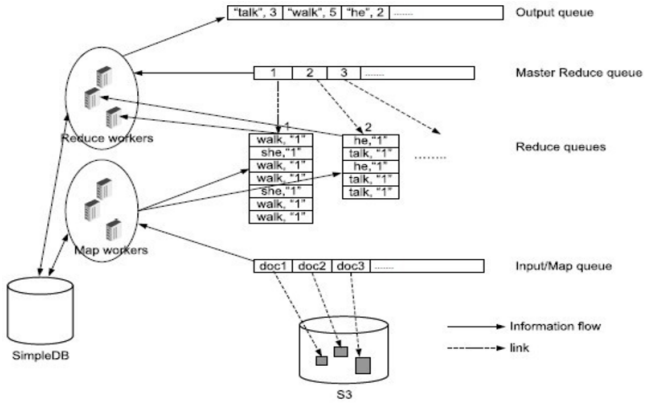

図 2: Cloud MapReduce のアーキテクチャ

Input queue の中にある key-value の数 S と Master Reduce queue の中にある Reduce queue の数 Q は、ユーザが設定する。

4.1　CMR のメリット

Cloud MapReduce は、他の高度にスケーラブルなシステムと同様ないくつかの望ましい性質を持っている。

Incremental scalability: Cloud MapReduce は、

コンピューティングノードの数を拡張することができる。

Symmetry and Decentralization: Cloud MapReduce 内のすべてのコンピューティングノードが自立的である。

Heterogeneity: 各計算ノードは、計算能力が多様であっても構わない。

Simplicity:CMR の Java コード数は 3,000 行しかない。

4.2 Cloud での課題と解決方法

前述したとおり MapReduce を Cloud OS 上で実装するのはいくつのメリットがあるが、クラウドにおけるいくつかの制限を克服必要がある。

Horizontal scaling:クラウドではシステムにノードを増やすことで、性能向上をはからなくてはならない。

Long latency: Amazon のサービスはネットワークを通じてアクセスされるので、遅延が大きくなってる。

解決方法:メッセージを集めて、キューにリクエストするスレッドプールを使って reduce キューとのデータ転送をする。

Indeterministic eventual consistency windows: データを更新した後一定期間、変更が反映されない。

解決方法:ワーカーが SimpleDB にステータスを知らせる。

Potential node failure:ワーカーノードが Map と Reduce タスクを処理している間に、失敗するかもしれない。

解決方法:タスクがもう一回現れるのため、他のワーカーが処理できる。

Duplicate message. キュー内の同じメッセージが複数回で処理される可能性がある。

解決方法: Reduce キューの冗長なマップの出力をフィルタする。Master reduce queue に対しては、conflict resolutoin というアルゴリズムがある。

5 Cloud Mapreduce の欠点

現在の CMR の実装の 1 つの欠点は、ローカリティの最適化を採用していないため、コンピューティングノードとクラウドサービス間のネットワークリンクが飽和している時、最終的にはネットワークのボトルネックが発生する。

6 実験評価

Wikipedia の記事のテキストファイルの約 13GB のデータを用いて、100 m1.small EC2 インスタンスのクラスタ上で、Hadoop0.21.0 と CMR の間のパフォーマンスを比較する。

MapReduce のジョブの実行時間を表 2 に示す。

表 2: 100 ノード上の 13GB データの処理時間（秒）

	combiner	no combiner
Hadoop	264	537
Cloud MapReduce	104	372
Cloud MapReduce w/sort	221	463

両方のケースでは、クラウド MapReduce が、Hadoop の約 2 倍の速さである。

図 3 は、 1 個の m1.small EC2 インスタンスが２００ MB データを処理してるの CPU とネットワークの利用率を示している。

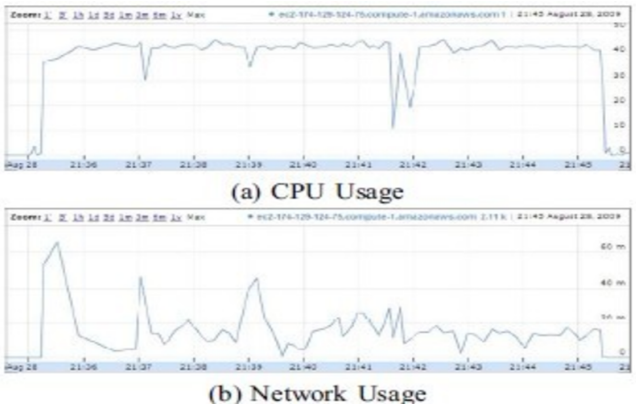

図 3: 1 個の m1.small EC2 インスタンスの CPU とネットワークの利用率

CPU は、ほとんどピーク使用率（40 % m1.small の最高率です）でジョブを実行している。ネットワークアクセスが広がっているため、ネットワーク帯域幅の需要は 60Mbps である。

7 まとめ

性能低下なしに、クラウドリミットを克服する方法を示した。このテクニックはクラウド OS 上の他のアプリケーションも利用できる。完全分散な MapReduce プログラミングモデルを提案した。

参考文献

[1] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in OSDI ’04: Sixth Symposium on Operating System Design and Implementation, December 2004.

[2] E. A. Brewer, “Towards robust distributed systems (abstract),” in PODC ’00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing. New York, NY, USA: ACM, 2000, p. 7.