

非集中型クラウドストレージのスケラビリティ評価

氏名 奥寺 昇平 学籍番号 07_0615_4

卒業年度 2010 年度 指導教員 首藤一幸

1. はじめに

Amazon Dynamo[1]、Cassandra[2]をはじめとした単一故障点がなく、負荷が自動的に分散される非集中型のクラウドストレージが普及しつつある。このような非集中なクラウドストレージにおいて、任意のノードからデータを保持する担当ノードにリクエストを到達させるためには、各ノードが他のノードを把握する必要がある。特に、クライアントが接続したノードから担当ノードに直接リクエストを送るクラウドストレージでは、全ノードがシステム全体の最新の状態を保持する必要があり、システムの整合性を保つことが難しい。

そこで、gossip プロトコルをベースとしたメンバーシップ管理を行うプロトコルが取り入れられ、効率よく通信を行うことが可能である。

一方、このようなメンバーシップ管理もスケラビリティを制約する要因の一つとなりうる。この管理方法では、すべてのノードで定期的に通信が発生するので、ノード数が増えるにつれ、総通信量が増えるのである。よって、フロントエンドの処理(データの読み書き処理)効率、つまりアベイラビリティを下げると考えられる。

しかしながら、非集中型クラウドストレージにおいて、この管理を行う処理がどれくらいの通信負荷をもたらすのかといったことは知られていない。

そこで本研究では、gossip プロトコルを用いる Cassandra を対象として、ノード数に応じてシステム全体の通信負荷がどのように変化するのかを計測・考察する。

2. gossip プロトコル

gossip プロトコルはソーシャルネットワークで見られる噂(ゴシップ)の伝搬をモデルとしたアルゴリズムである。例えば、以下のような手順で繰り返し行われる。

1. ノード P のデータが更新されたとき、ランダムに他のノード Q を選択して更新情報をノード Q に反映させる。
2. ノード Q がすでに更新済みであったときは、ノード P は他のノードに更新情報を伝えるのをやめる。

gossip プロトコルに代表される伝染性のアルゴリズムを利用したプロトコルの長所は、プロセスが通信する回数が他の伝搬手法と比べて比較的少ないために、ノード数がスケールしやすいところにある。

3. Cassandra

本節では、実験に用いた Cassandra について説明する。Cassandra は、Facebook 社が開発し、Apache Project としてオープンソース化した非集中型クラウドストレージである。Cassandra では、gossip プロトコルをベースとしたメンバーシップ管理アルゴリズム[3]を利用している。このアルゴリズムでは、毎秒各マシンでランダムに他のノードと経路情報を交換しあい、システム全体のノード情報の整合性を保っている。

4. 測定手法

実験では、1 台あたり複数の Cassandra ノードを起動し、マシン間で発生するトラフィックを解析した。tcpdump を使用して、トラフィックを計測した。また、Cassandra ノードを多数立ち上げる際に、メモリー使用量を節約するために、Cassandra のデータ保存部分のプログラムを改変し、メモリー使用量を削減した。

以下に実験環境を示す。

- Cassandra 0.6.6
- OS: Linux 2.6.35.10-74.fc14.x86_64
- Java 仮想マシン: Java SE 6 Update 21
- CPU: 2.40 GHz Xeon E5620×2
- メモリー: 32GB RAM
- ネットワーク: 1000BASE-T

実験にはマシンを 10 台使用した。

実験シナリオについて説明する。30 秒ごとに、1 台あたり 10 ノードの Cassandra を一度に起動し、これを目指す台数に達成するまで続ける。最初の Cassandra ノードを起動した瞬間から 10 分間の通信量を計測した。

5. 実験・評価

Figure 1 は、10 秒あたりの Cassandra ノー

ドで発生する総通信量の平均の時間変化をノード数別に表したグラフである。(ただし、 $1M=10^6$, $1K=10^3$ とする。)

Figure1:ノード台数別通信量の時間変化

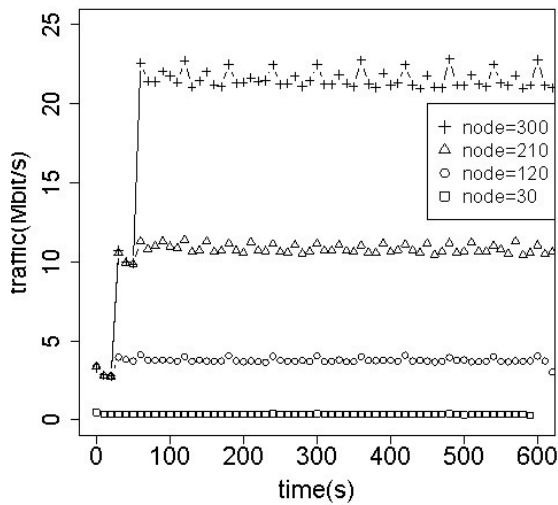
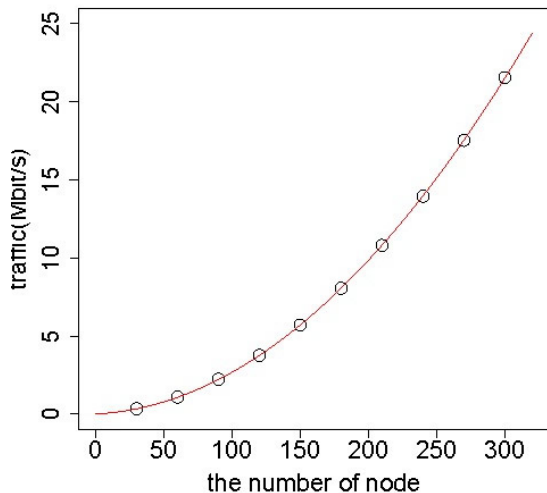


Figure2:ノード台数と通信量の変化



ノード数によらず、100 秒以降は通信量が安定していることがわかる。

Figure 2 は、ノード数と通信量が安定している時の(ここでは、実験開始から 200-300 秒後とした)1 秒あたりの通信量の平均をプロットしたものである。図中の曲線は、プロットした点から二次関数でフィッティングしたものである。 n をノード数として得られた関数は、

[通信量(bit)] = $224.6 \times n^2 + 4314.8 \times n$ である。つまり、通信量は $O(n^2)$ でスケールすることがわかった。また、上の式を n で割った 1 ノードあたり通信量は $O(n)$ で増加することもわかった。

これらの関数から、ノード数をパラメータとして gossip プロトコルで発生しうる全体の通信量を推測することができる。例えば、 $n = 1000$ のとき、[通信量] = 229Mbps となる。このように、この関数を使って総通信量が見積もることができる。また、クラスタの設計時にも活かすことができる。ケースとして、複数のデータセンターをまたぐクラスタを構成する時を

考える。この時、データセンター間のリンクは $O(n^2)$ で通信量が増加する。つまり、ノード数が増加したときにこのリンク部分の通信が圧迫される可能性がある。このようなリンク部分を考慮した gossip プロトコルの改善が望まれる。

6. まとめ

非集中型クラウドストレージにおけるメンバーシップ管理の通信負荷を推測するために、gossip プロトコルを用いた Cassandra を対象として、ノード数に応じてシステム全体の通信負荷がどのように変化するのを実験・評価した。

この実験と評価により、定量的にクラウドストレージの gossip ベースのメンバーシップに要する通信量を計測することができ、またそれは $O(n^2)$ で増加することがわかった。さらに複数のデータセンターをまたぐクラスタを構成するとき、データセンター間をまたぐリンク部分は $O(n^2)$ で通信量が増加し、通信が圧迫される可能性がある。このようなリンク部分を考慮した gossip プロトコルの改善が望まれる。

7. 今後の研究

今後の研究としては、大きく二つを考えている。一つ目は、データセンター間のリンクを意識した gossip プロトコルの提案である。本研究により、データセンター間のリンクは $O(n^2)$ で通信量が増加し、通信が圧迫される可能性があることがわかった。そこでこのようなリンク部分を考慮した gossip プロトコルの改善が望まれる。二つ目としては、gossip プロトコルを通信量以外の切り口でスケーラビリティ評価することである。CPU 占有率や故障の伝搬スピード等、スケーラビリティの制約になる可能性のある指標はまだ残されている。

参考文献

- [1]Avinash Lakshman and Prashant Malik, *Cassandra – A Decentralized Structured Storage System*, In Proc. LADIS '09, 2009.
- [2] Giuseppe de Candia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Voss, and Werner Vogels, *Dynamo: Amazon's Highly Available Key-value Store*. In Proc. SOSP '07, 2007.
- [3] Robbert van Renesse, Dan Dumitriu, Valient Gough, Chris Thomas and Amazon.com, Seattle. *Efficient Reconciliation and Flow Control for Anti-Entropy Protocols*. In Proc LADIS '08, 2008.