卒論の研究内容について

奥寺昇平

2010 年 10 月 22 日

1　研究目的

長尾さんが卒論で考案した、経路表 (データのキーとその担当ノードの一覧表) 管理アルゴリズムの FRT の効果を測定することが研究目的である。前期に取り組んだ課題に引き続き、Cassnadra という分散システムに FRT を導入し、その効果を測定する。想定する環境としては、ノードの数、ノードの出し入れの頻度をパラメータにする。

2　目標

1. オリジナルの Cassandra と同等の機能をもつ、Cassandra を実装する。
   -前期の課題の段階では、データの複製戦略や、一貫性レベルを調整する機能が使えなかった。
   -データの複製戦略の実装の工夫が必要である。

2. オリジナルの Cassandra をベースにして、新たに作成した Cassandra を性能比較する。
   -ノードの数、ノードの出し入れの頻度をパラメータにする。特に、ノードの数が多く、ノードの出し入れの頻度が高い条件の時、性能が出ることを予想する。

3　Cassandra とは

米 Facebook 社が自社の SNS のインフラとして開発した、大量の商用サーバーに広がる大量の構造化データを扱う分散ストレージシステムである。

2008 年にソースコードがオープンソースとして公開されています。オーバレイネットワーク上に連想配列 (key-value) を構築する技術である DHT を用いて、データを複数ノードに分散して配置する。ノードと、データのキーには、リング上の ID 空間の座標 Token が一つ割り当てられていて、あるデータの担当ノードは、「前のノードの Token ＜ データの Token ＜＝ ノードの Token」を満たすノードに決定されます。このノードがデータの主担当で、さらに複製として、ID が大きくなる方向に主担当から設定された分の数のノードが同じデータを保存します。これがデータの複製方法です。

次に、Cassandra の (データの) ルーティングの方法について説明する。Cassandra クラスタを構成するノード群は、Gossip と呼ばれるプロトコルを利用して随時ノード情報を取得し、ノード情報を最新の状態に保っている。read/write 等のリクエストをクライアントから受けると、そのノードが担当するノードを決定し、リクエストを転送する仕組みになっている。

4 前期で取り組んだこと

前期の課題は、長尾さんが卒論で考案した、柔軟に経路表を管理するアルゴリズム FRT を Cassandra に導入することが目的であった。

FRT では、設定数以上のノードを経路表に登録しようとしたとき、適切な経路情報だけを残して経路表から削除してしまうので、各ノードが持つ経路情報は、完全なものではなくなる。そこで、シングルホップから、リクエストがノード間を経由して最終的に担当ノードにたどり着くマルチホップシステムを採用し、データを取得できるようにした。前期の段階では、データの複製については考慮していなかった。そこで、この複製についてこれから特に取り組んでいく。

5 想定仕様 (データ複製の方法)

ここではデータ複製を考慮した read 時のオペレーションについて、そのアイディアを説明する。

- 現状の Cassandra
  - 【条件】
    ・クラスターを構成するあらゆるノードが、完璧な経路表情報 (データのキーとその担当ノードの一覧表) を持っている。
  - 【手順】
    1.Proxy(クライアントからリクエストを受けたノード) が担当ノードを指定する。
    2.proxy からリクエストをもらった担当ノードは、データの要約を返す。IP アドレス的に近い担当ノードからは、同時にデータも取得する。
    3.proxy は担当ノード群からもらったデータの要約を照らし合わせ、一貫性がとれているならば、そのデータを返す。一貫性がとれていなかったり、データが古かったら、最新のデータがクライアントに渡せるように、一貫性の調整を行い、データを再取得し、クライアントにデータを渡す。
- 新しく作る Cassandra
  - 【条件】
    ・どのノードも、完璧な経路表情報 (データのキーとその担当ノードの一覧表) を持ってない。
    しかし、自らのノードの (リング上の ID 的に) 近傍のノード情報については完全に知っている。

(そのように設定しておく)

– 【手順】

1. マルチホップで主担当ノードへ到達する。

2. 主担当ノードは、近傍のノード情報については完全に知っているので、キーに対するレプリカの位置情報を取得でき、これを Proxy に送ります。

3.Proxy は、その位置情報を元に、リクエストを投げ、データの要約を受け取る。

4. データの要約を照らし合わせ、一貫性がとれているならば、IP アドレス的に近い担当ノードからデータを要求し、クライアントにデータを渡す。一貫性がとれていない場合、最新のデータがクライアントに渡すと同時に、一貫性の調整を行います。一貫性の調整と実データ取得は別プロセスで行います。

- メリット、デメリット
  – 【メリット】

・クライアントにデータを返す時間が短い。

新しく作る Cassandra では、担当ノード群すべてからデータの要約のみをもらうので、担当ノードにアクセスしてから一貫性チェックを開始するまでの時間がはやい。結果として、クライアントにデータを返す時間が短縮できる機会が増える。

つまり、オリジナルの Cassandra のやり方では、データの要約をリクエストすると同時に、必ず一つのノードからは、データ本体を取得しないといけない。そのデータ量が多いと、転送時間に時間がかかるのである。

オリジナルの Cassandra で取得したデータが最新のものであれば、すぐクライアントにデータを渡せるので、新しく作る Cassandra と同等だが、データが最新ではなかった場合は、再度最新のデータを持っているノードにデータを要求するため、さらに時間がかかるのである。その分、新しい Cassandra では、データ本体を転送するのは必ず 1 回だけなので、クライアントにデータを帰す時間は短い。

・ルーティング時のホップ数が平均 O(LogN) で押さえられる。(全体のノード数を N とする。)

– 【デメリット】

・ルーティング時のホップ数が O(1) から平均 O(LogN) に上昇する。(全体のノード数を N とする。)