mroongaを使った、高速な対訳検索＆ダウンロードシステムについて

～BI的使い方の場合の高速化～

目次

- システム概要
- C#プログラマが構築した第1システム
- 第1システムの特性
- ユーザーからの要望
- 遅い原因の推測
- DBエンジニアが構築した第2システム
- 第2システムの特性
- 第2システムによる高速化結果
- まとめ

システム概要

- 3千5百万件の日英対訳データ
  （データ量は16GB弱）
- Mroongaで全文検索DBを構築
- 単語 or フレーズで検索
- 検索結果をファイルとしてダウンロード

C#プログラマが構築した第1システム

- SENNAでの知見を活かし、5百万件程度にテーブルを分割
- 分割後に別データを追加したので件数が不揃いに

  （第1~第4テーブルは5百万件、

  　第5は374万件、第6は千百万件）
- C#のプログラムが、エンドユーザーのPCに
- DB自体は同じフロアの別セグメントのマシンルーム内

第1システムの特性

- 20スレッドで検索（SENNAでの知見による）
- 1回のSQLで、limit　1万を指定
- ループでフェッチ
- 1万件を超えた場合は、limit 1万 offset 1万で再検索
- 単語によっては、検索開始からダウンロード終了まで5分!!
- Xeon L5320(1.8GHz)4コア、8GBのRAM

ユーザーからの要望

- 最低でも10秒以内にしてください
- 出来れば、5秒以内にしてください
- つまり、現状300秒の処理を、30倍～60倍高速化する必要がある!!

遅い原因の推測

- 何件マッチしたかの select count(1)の結果は1秒以内に返って来る
- RAMが足りなく、Swapが2GBも使われていた
- フェッチでDBから大量にデータを取得するのに時間が掛かっていると思われる

```
[root@DB28 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          7981       7937         44          0        156       6731
-/+ buffers/cache:       1049       6932
Swap:         9983       2055       7928
[root@DB28 ~]#
```

DBエンジニアが構築した第2システム

- キューブPCを使用(32GBのRAM、Core i7 3.5 GHz 仮想8コア。マシンルーム内）
- /dev/shm （RAM Disk）上にmroongaを構築
- UIをPHPに
- PHPがシェルを呼び出し、シェルがmysqlを呼び、SQLの結果をファイルに出力
- CPUが8個なので、テーブルを8分割し、1プロセスが1テーブルを検索するようにした

第2システムの特性

- 各プロセスは、それぞれ用のテーブルを見る
- 8プロセスが、バックグラウンドモードで起動され、検索結果を専用のファイルに書く
- 全プロセス終了を待機(bashのwaitを使用）
- 全プロセス終了後、各ファイルをcatで繋げて出力
- 上記を1つのbashコマンドとして実行

第2システムによる高速化結果

- なんと、5分掛かっていた検索が2秒以下に。
- 150倍の高速化。

実際のbashスクリプトは、こんな感じ。

mysql $c < sql1 > out1 & mysql $c < sql2 > out2 & … wait; cat out1 out2 …

まとめ

- クライアント＆サーバー型で、1レコード毎にデータを大量にフェッチすると遅い
- SWAPが起きる状況だとmroongaといえど遅い
- RAM Disk上にデータを置くと速い
- 結果をファイルとして得るなら、可能な場合はmysqlを使って1処理で取得した方が速い
- 複数テーブルに分けて検索して結果をマージしたほうが速いかは未検証