<https://archive.cloudera.com/cdh5/cdh/5/hbase-0.98.1-cdh5.1.3/book/config.files.html>

<https://docs.cloudera.com/documentation/enterprise/properties/5-12-x/topics/cm_props_cdh5110_hbase.html>

**HBase Performance Tuning | Ways For HBase Optimization**

**1. Mục đích của việc điều chỉnh hiệu năng của HBASE**

Hi cả nhà, hôm nay trong bài viết này mình xin giới thiệu với các bạn về một số cách để tối ưu hóa môi trường HBASE. Chúng ta sẽ cùng nhau tìm hiểu về hiệu chỉnh bộ thu gom rác (garbage collection tuning), nén dữ trong HBASE và các cấu hình trong HBASE. Sau đó mình sẽ trình bày về việc áp dụng thử nghiệm tải cho việc hiệu chỉnh hiệu năng của HBASE.

Như chúng ta đã biết HBASE cũng là một phần của kiến trúc Hadoop và cơ sở dữ liệu phân tán do đó việc tối ưu hiệu năng cho HBASE càng nhiều sẽ càng tốt, nó cũng sẽ giúp chúng ta rút ngắn được các chi phí cho các thao tác dữ liệu trong HBASE cũng như hệ thống của chúng ta hoạt một một cách trơn tru và mượt mà hơn.

Chắc giới thiệu vậy thôi nhỉ? Let’s go ☺

**2. Hiệu chỉnh bộ thu gom rác (Garbage Collection Tuning)**

Tham số thu gom rác - Garbage Collection Parameter là một trong những cài đặt lower-level mà chúng ta cần hiệu chỉnh cho các region server processes. Trong HBASE dữ liệu chắc chắn không đi qua Master và Master cũng không xử lý bất kỳ tải nặng nào vì vậy chúng ta sẽ chỉ hiệu chỉnh các tham số thu gom rác cho các region servers.

Thế tại sao chúng ta lại phải hiệu chỉnh tham số này? HBASE hoạt động dựa trên JRE, các tiến trình sẽ tạo các đối tượng cũng như xin cấp phát bộ nhớ heap để xử lý dữ liệu... có một điều là JRE lại không handle tốt các region servers khi mà dữ liệu được ghi với dung lượng lớn, việc cấp phát bộ nhớ sẽ bị treo và nó sẽ không an toàn cho hệ thống, chúng ta sẽ cần phải hiệu chỉnh các tùy chọn của JRE để điều chỉnh các chiến lược thu gom rác phù hợp với khối lượng công việc.

Đối với các trường hợp ghi dữ liệu với dung lượng lớn, các MemStores tạo và loại bỏ các đối tượng ở nhiều thời điểm khác nhau và ở các kích cỡ khác nhau, Dữ liệu được lưu đệm trên RAM và nó sẽ duy trì ở đó cho đến khi vượt quá kích thước xả tối thiểu (minimum flush size), chúng ta có thể hiệu chỉnh nó với thuộc tính **hbase.hregion.memstore.flush.size** trong file cấu hình **hbase-site.xml**.

Chúng ta có thể cấu hình các tùy chọn cho bộ thu gom rác bằng cách thêm các cấu hình cho biến **HBASE\_OPTS** hoặc **HBASE\_REGIONSERVER\_OPTS** trong file **hbase-env.sh** để hiệu chỉnh cho bộ thu gom rác, cụ thể đó là các tham số sau:

# cấu hình kích thước file khi tạo mới trên heap

-Xmn128m

# cấu hình hiển thị log gc

-verbose:gc

-XX:+PrintGCDetails

-XX:+PrintGCTimeStamps

-Xloggc:$HBASE\_HOME/logs/gc-$(hostname)-hbase.log

# cấu hình sử dụng trình quét đồng thời – CMS với 70% tài nguyên

-XX:+UseConcMarkSweepGC

-XX:CMSInitiatingOccupancyFraction=70

Chú ý rằng: các giá trị có thể tuỳ biến cho phù hợp với các hệ thống và dữ liệu thực tế.

**3. Hiệu chỉnh bộ đệm phân bổ Memstore-Local (Memstore-Local Allocation Buffer)**

Để giảm thiểu vấn đề phân mảnh heap do quá nhiều phiên bản memstore của một HBase Region Server, phiên bản 0.90 của HBase đã giới thiệu một cơ chế nâng cao đó chính là bộ đệm phân bổ Memstore-Local (MSLAB).

Theo như Garbage Collection Tuning chúng ta đã biết rằng khi dữ liệu được flush, nó sẽ tạo ra các vùng trống trong vùng dữ liệu Heap cũ, Khi không còn đủ không gian cấp mới cho các đối tượng do phân mảnh gây ra, JRE sẽ dừng lại và thực hiện thu gom rác, nó ghi lại toàn bộ không gian Heap và gom lại các vùng trống cho các đối tượng hoạt động.

Chìa khóa để giảm việc thu gom tổng hợp này là giảm phân mảnh và MSLAB (**Memstore-Local Allocation Buffer**) được xây dựng để giúp giải quyết vấn đề đó. Ý tưởng ở đây là chỉ các đối tượng có cùng kích thước mới được phân bổ trong Heap do đó khi dữ liệu được flush, các vùng trống được tạo ra là hoàn toàn phù hợp cho các đối tượng mới được phân bổ và JRE sẽ không phải dừng lại để thực hiện việc thu gom rác, hệ thống của chúng ta sẽ hoạt động mượt mà hơn…

Chúng ta có thể cấu hình bật tắt MSLAB bằng cách thay đổi thuộc tính **hbase.hregion.memstore.mslab.enabled** trong file **hbase-default.xml** với giá trị true/false. Tính năng này được bật theo mặc định trong các phiên bản >=0.92 và bị tắt trong phiên bản <=0.90 của HBase.

Chúng ta có thể cấu hình kích thước bộ đệm phân bổ bằng cách thay đổi giá trị của thuộc tính **hbase.hregion.memstore.mslab.chunksize** trong file **hbase-default.xml**, giá trị mặc định của nó là 2MB.

Chú ý rằng: cái gì cũng có 2 mặt của nó, nếu chúng ta bật MSLAB chúng ta phải đánh đổi sự lãng phí bộ nhớ vì với các KeyValue có dung lượng thấp hay không lấp đầy bộ đệm thì sẽ có nhiều bye trống được tạo ra, chúng ta sẽ không tối ưu được bộ nhớ nhưng đổi lại chúng ta sẽ có sự trơn tru trong các hoạt động của HBASE.

**4. Nén trong HBASE (HBase Compression)**

Có một tính năng nữa của HBase, đó là nó hỗ trợ một số thuật toán nén. Về cơ bản, thuật toán nén HBase có thể được kích hoạt ở column family level. Ngoài ra, nén cũng mang lại hiệu năng tốt hơn, đối với mọi trường hợp CPU thực hiện nén và giải nén, chi phí hoạt động sẽ thấp hơn hoạt động đọc thêm dữ liệu từ đĩa.