# Consultas en grandes volúmenes de datos

Facultad de Ingeniería, Universidad de la República

Computación de alta performance 2013

Luciana Canales
Montevideo, Uruguay
lucanales@gmail.com

Gabriel Centurión
Montevideo, Uruguay
gccomputos@gmail.com

***Resumen* - En el marco del proyecto de computación de alta performance el presente trabajo se orienta a evaluar las herramientas hadoop y hive para comprobar las capacidades de disponibilidad, escalabilidad y paralelización en el procesamiento de grandes volúmenes de datos**

**Dicha evaluación permitirá conocer los pasos a seguir para realizar el procesamiento masivo de logs, utilizando las herramientas mencionadas en un ambiente distribuido. La misma se realizará tanto desde las perspectivas de las consultas como de la paralelización, escalabilidad y rendimiento en un ambiente heterogéneo.**

**Para ello se diseñaran pruebas que serán ejecutadas en arquitecturas centralizadas como distribuidas y se realizarán las correspondientes mediciones, estableciendo los porcentajes de mejoras para cada caso.**

***Palabras claves - Big data, Hadoop, Hive, BGP, Sistema Autonomo (AS), Prefijo, AS_PATH.***

## I. INTRODUCCIÓN

El procesamiento de grandes cantidades de datos (Big Data) es una necesidad para empresas que manejan una cantidad considerable de transacciones. Ya en etapas maduras cuando los procesos obtienen el grado de maduración aceptable, aún es posible mejorar los mismos y tomar decisiones estratégicas basándose en la experiencia propia de la empresa y de su realidad.

Gran parte de la historia de las ejecuciones y flujos de las mismas quedan en los logs generados por los procesos. Allí se pueden identificar desde caídas del sistemas, cantidad de errores en un periodo de tiempo, entre otros datos.

## II. JUSTIFICACIÓN DE UTILIZAR PROCESAMIENTO PARALALEO

El procesamiento masivo de logs de gran tamaño requiere de una capacidad computacional importante. Esto se ve incrementado cuando las consultas tienen cierta complejidad y no solo son de sumarización o de contabilización, sino que además se pretende reconstruir un comportamiento basándose en los datos recabados, de forma que hasta podrían ser necesarias estructuras intermedias para realizar su cálculo.

La incorporación del procesamiento paralelo a este problema pretende satisfacer unos de los requisitos del poder de computo: grandes volúmenes de datos. [1]

## III. ESTRATEGIA DE RESOLUCIÓN

En la realización del proyecto propuesto se decide trabajar sobre un caso de estudio en particular, logs BGP.

BGP (Border Gateway Protocol) es un protocolo mediante el cual se intercambia información de encaminamiento o ruteo entre sistemas autónomos [3]. Es un protocolo absolutamente critico en Internet [4] y por tal motivo resulta interesante realizar estudios estadísticos sobre sus logs.

Dichos estudios permitirán obtener información sobre la diversidad de conectividad del AS y aproximaciones para realizar estudios sobre el diámetro de Internet. Para ello se realizaran las siguientes estadísticas sobre los logs:

### ESTADISTICA 1

Búsqueda de la existencia de múltiples rutas para un prefijo. Esto da la idea de la diversidad de conectividad del AS origen del prefijo.

### ESTADISTICA 2

Estadísticas sobre el length del AS_PATH. Esto permite aproximarse a estudiar el diámetro de internet.

Los pasos a realizar en la resolución del proyecto son la familiarización de los datos que nos proporcionan los logs, la transformación de los mismos a un formato aceptable para ser procesados por Hive [2] y luego la realización de las consultas necesarias para resolver las estadísticas anteriormente descriptas.

Las diferentes pruebas que se realizaran sobre las consultas permitirán obtener información relevante al proceso de paralelización como ser, speedup, escalabilidad, entre otros.

### A. *Balanceo de cargas*

Hadoop promete un balance de carga automatizado. Se comprobará mediante la experimentación el funcionamiento del mismo.

### B. *Tolerancia a fallos*

Hadoop provee de tolerancia a fallos. Se evaluará el funcionamiento.

## IV. EVALUACIÓN EXPERIMENTAL

Para realizar una evaluación de la solución se propone desarrollar un algoritmo serial de referencia y medir los tiempos de ejecución. Luego desarrollar un algoritmo paralelo con Hive, medir los tiempos, calcular el speedup y evaluar su escalabilidad.

Los pasos requeridos para la evaluacion son los siguientes:

### OBTENCION DE LOGS

Los logs se obtienen mediante la herramienta wget la cual permite descargar contenido desde servidores web de una forma simple [5].

El proceso se descarga se realiza mediante un script (descarga.sh) que obtiene de forma serial los logs desde http://data.ris.ripe.net/rrc00/$anio.$mes/updates.2013$mes$dia.$hora$minuto.gz para todo el año 2013 y uno cada 5 minutos.

Se desarrolla una primera versión del script en bash 4.0 el cual permitía la utilización de rangos en el wget, por ejemplo wget data.ris.ripe.net/rrc00/20{13..14}.{01..06}/updates.20{13..14}{01..12}{00..31}.{00..23}{00..55..5}.gz, pero luego debió desarrollarse en bash 3.x (instalado en el ambiente de ejecución cluster fing [10]) y reprogramarse todo con for ya que dicha versión no soportaba rangos.

### CONVERSION

Los archivos que se obtuvieron con el proceso detallado en el paso anterior eran en formato binario por ello debieron ser convertidos a un formato legible para poder ser procesados.

La conversión de los datos se realizaron con la herramienta bgpdump.

### PRUEBAS SERIAL

El objetivo de las pruebas seriales es obtener tiempos de referencias para luego realizar comparaciones con las pruebas en paralelo y ver si realmente se logra obtener mejorías. Para ello se desarrollo un modulo java.

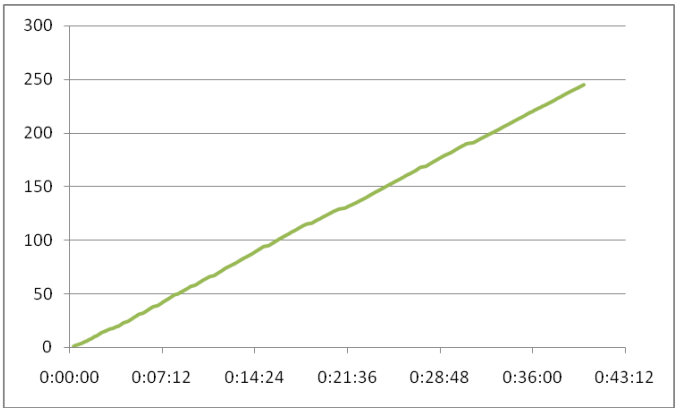
*Prueba 1 - 1 archivo de 31 gb, mes de abril del 2003*

La misma no finalizó debido a errores de heap space.

*Prueba 2 - 1 archivo de 31 gb, mes de abril del 2003*

Se realiza lo mismo que en la prueba anterior pero con un heap space de 40gb.

La misma finalizó correctamente y en 40 min 4 seg sobre un total de 245078506 líneas.

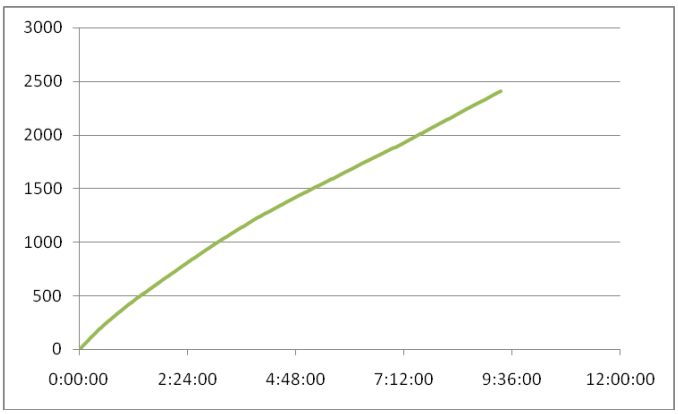



#líneas archivo (en millones) vs tiempo

*Prueba 3 - 6 archivos que suman 303 gb en total, primer semestre del 2013*

Se asigna un heap space de 60gb y se procesan 6 archivos de gran tamaño, 31gb, 32gb, 34gb, 59gb, 64gb y 83gb.

La misma finalizó correctamente y en 9 hr 21 min 38 seg sobre un total de 2410707908 líneas.

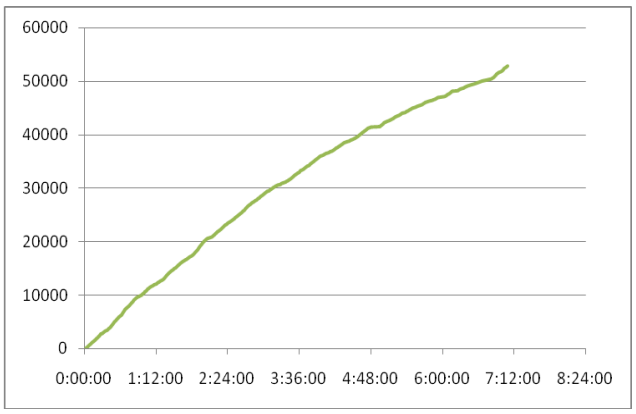



#líneas archivos (en millones) vs tiempo

*Prueba 4 - 52848 archivos que suman 263 gb en total, segundo semestre del 2013*

Se asigna un heap space de 60gb y se procesan 52848 archivos de tamaño pequeño.

La misma finalizó correctamente y en 7 hr 6 min 6 seg.




#archivos vs tiempo

## PRUEBAS EN PARALELO

Hadoop permite la ejecución en un único nodo en un modo pseudo-distribuido donde cada daemon Hadoop ejecuta en un proceso independiente de Java [6].

Se toma como referencia para el diseño de las pruebas a ejecutar las pruebas realizadas en modo serial. Las mismas se realizan en un ambiente pseudo-distribuido.

Para el proceso de evaluación de la herramienta hadoop cada prueba será ejecutada reiteradas veces pero en cada caso variando propiedades tales como la cantidad de mappers y reducers, control del números mappers y reducers que se ejecutan en paralelo y control del número de procesos con el file size y el block size del DFS.

*Prueba 1 - 1 archivo de 31 gb, mes de abril del 2003*

Las variaciones de las propiedades descriptas anteriormente arrojan resultados fallidos.

No se logra controlar la cantidad de procesos debido a que hadoop decide el número de procesos que él considera conveniente a la hora de ejecutar. Tampoco se logra controlar la cantidad de mappers y reducers a utilizar en cada consulta. El trabajar de forma local presenta limitantes con los reducers según especificación [7].

En el único caso que se logra visualizar valores mínimamente significativos variando el tamaño en bytes por reducers mediante el comando set hive.exec.reducers.bytes.per.reducer = nro bytes.

La siguiente tabla muestra los resultados obtenidos:

| Tamaño en gb | Tiempo de ejecución |
|---|---|
| 0.5 | 1:32:00 |
| 0.75 | 1:10:00 |
| 1 | 0:55:00 |
| 2 | 0:53:00 |
| 3 | 0:51:00 |
| 31 | 0:51:00 |

Tabla 1

La elección de los diferentes tamaños se hace en base a el tamaño por defecto que es de 1gb [7] y se prueban valores inferiores y valores superiores.

En la tabla 1 se puede observar como el tiempo de ejecución va disminuyendo hasta 1gb y luego se mantiene prácticamente sin variaciones. Esto da una pauta de que por más tamaño que se asigne el resultado no va a sufrir grandes cambios, aún cuando se pruebe con un tamaño igual al del archivo a evaluar (31 gb en este caso).

*Prueba 2 - 6 archivos que suman 303 gb en total, primer semestre del 2013*

Pese a varios intentos no se logra concluir dicha prueba debido a los errores suscitados en la importación de todos los archivos al dfs.

Se probaron diferentes opciones tales como las que se indican más abajo, pero en todos los casos la importación del primer archivo funcionaba correctamente pero el segundo fallaba.

hadoop fs -put <localsrc> <dst>

hadoop dfs -put <localsrc> <dst>

hadoop fs -copyFromLocal <localsrc> URI

También se trabajó con diferentes opciones publicadas en los foros ([8] y [9] entre otros) pero no se obtuvieron resultados; el problema persistió.

## V. CONCLUSION

En la evaluación de los tiempos de ejecución obtenidos tanto serial como paralelo se puede apreciar que para obtener resultados en momentos precisos para que las organizaciones puedan tomar sus decisiones de negocio, se necesitan de herramientas que trabajen de manera adecuada con grandes volúmenes de datos.

El presente trabajo muestra como hadoop y hive no están pensados para trabajar con poca información y en ambientes no distribuidos ya que claramente se puede apreciar que el tiempo de respuesta serial fue inferior al paralelo.

Como conclusión final cabe destacar que la experiencia adquirida en herramientas de gran porte fue muy enriquecedora.

## VI. TRABAJO A FUTURO

Como trabajos a futuros se realizaran dos estadisticas que permitirán obtener información sobre el funcionamiento del AS y detectar ASes que tienen problemas de configuración del routing interno.

### ESTADISTICA 1

Búsqueda de updates (o "announcements") duplicados. Esto puede dar indicio de anomalías en el funcionamiento del AS que produce los duplicados.

### ESTADISTICA 2

Esta estadística busca obtener que ASes generan más prefijos y/o son más "habladores" por unidad de tiempo correlacionando updates y withrawals para detectar ASes que tienen problemas de configuración del routing interno que se reflejan en el BGP.

Para mejorar el proceso completo del caso de estudio se realizará la paralelización de la descarga y conversión de lo logs mediante uso de hilos. El criterio para la generación de los mismos será por mes, quincenales, o según estudio previo para realizar un correcto balance de carga.

## REFERENCIAS


[1] http://www.fing.edu.uy/inco/cursos/hpc/material/clases/Tema1-2013.pdf

[2] http://help.papertrailapp.com/kb/analytics/log-analytics-with-hadoop-and-hive

[3] http://es.wikipedia.org/wiki/Border_Gateway_Protocol

[4] Redes de computadoras, Quinta Edición, de James F. Kurose y Keith W. Ross

[5] http://es.wikipedia.org/wiki/GNU_Wget

[6] http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html

[7] https://svn.apache.org/repos/asf/hive/tags/release-0.11.0/conf/hive-default.xml.template

[8] http://stackoverflow.com/questions/5293446/hdfs-error-could-only-be-replicated-to-0-nodes-instead-of-1

[9] http://stackoverflow.com/questions/21581448/hadoop-file-could-only-be-replicated-to-0-nodes-instead-of-1

[10] http://www.fing.edu.uy/cluster/index.php