Introducción

Este trabajo consiste en el diseño, desarrollo y testeo de un monitor de mensajes de log para servidores Apache. Como base se tienen una serie de requerimientos que se deben cumplir:

* Requerimientos Funcionales:
  + Se debe monitorear un archivo de log de servidores Apache redirigido a la entrada estándar de la aplicación.
  + Es necesario relevar las líneas de log de forma constante para identificar estadísticas con: cantidad de requests por segundo, requests por cliente, cantidad de errores, 10 recursos más pedidos.
  + Se deben identificar los mensajes de error con más repeticiones.
  + La aplicación se debe detener si se detecta un cierre del archivo de log o la señal de interrupción control+C.
* Requerimientos No Funcionales:
  + La aplicación debe priorizar la recepción de la entrada de logs en todo momento y ejecutar el análisis en paralelo.
  + Utilizar cantidades controladas de hilos (revisar necesidad de *Thread Pools*)
  + Se deben mostrar las estadísticas por pantalla con una frecuencia de 1 minuto.
  + Se debe realizar una descarga de todas las entradas de log recibidas a un archivo.
  + Se debe realizar una descarga de los mensajes de error en otro archivo de texto, incluyendo únicamente fecha y mensaje.
  + El almacenamiento de datos y entidades del sistema se puede realizar utilizando algún esquema simple de serialización (no es foco del TP el uso eficiente de la persistencia).

Suposiciones

Se relevaron los requerimientos mencionados para anotar aquellas áreas bien definidas que tendrán un impacto directo sobre el diseño e implementación, y otras más ambiguas sobre las que fue necesario definir una estrategia a seguir.

Suposición 1 – Existe un formato conocido que tendrán los mensajes de log recibidos

La aplicación se centra en el monitoreo de mensajes de log de servidores Apache. Estos mensajes son altamente configurables, por lo que se definió una estructura para los mensajes de la forma: “client datetime [log\_type] “METHOD request http\_ver” status\_code status\_msg”. Por ejemplo:

* 192.168.0.1 28/03/2018:16:05:33 [error] “GET /home/test/file HTTP/1.1” 200 Ok
* 168.14.22.1 23/06/1952:13:23:42 [log] “POST /home/asd/1 HTTP/1.1” 404 File not found

Suposición 2 – La ventana para la visualización de estadísticas temporales no es estricta

Uno de los requerimientos es la visualización de estadísticas calculadas en una ventana de tiempo. Dado que el servidor probablemente este encendido todo el tiempo, estará enviando mensajes de log constantemente a la aplicación. Estas estadísticas entonces serán informativas del estado general del servidor en un momento dado, pero dicho momento no necesita estar definido rigurosamente. Por esta razón no considero necesario hacer un corte fuerte para el cálculo de cada estadística. Es suficiente con realizar un cálculo aproximado aún si se procesa algún mensaje extra durante el cálculo de ellas.

Suposición 3 – La cantidad de estadísticas a acumular no es muy elevada

Existe una distinción entre estadísticas y los valores utilizados para calcularlas. Por ejemplo, para el cálculo de “cantidad de requests por cliente” es necesario conocer la cantidad de requests y la cantidad de clientes. Estos valores podrían acumularse en un contador, pero si en un futuro se quisiera agregar la estadística “los 10 clientes con más requests realizadas”, sería necesario agregar dicha estadística. Para evitar esto, propongo separar por un lado el almacenamiento de datos estadísticos, y por otro el procesamiento de ellos para obtener ciertas estadísticas deseadas. Esto implica que como máximo será necesario acumular apariciones para cada uno de los campos definidos en la Suposición 1.

Suposición 4 – Existe un rango horario de baja solicitud del servidor

Entre los requerimientos se menciona un “identificar los mensajes de error con más repeticiones”. Dado que el servidor estará prendido siempre, esta información puede crecer enormemente con el paso del tiempo. En primer lugar, esto implica que se almacenen los datos de manera permanente ya que resultaría imposible mantenerlos en memoria. Y dado que se pide mostrar la información con un cierto orden, será necesario procesar esta información periódicamente. Dado que esta operación podría tardar un tiempo con volúmenes elevados de datos, se propone la definición de un rango horario conocido donde el servidor no recibirá muchos pedidos. Durante este rango horario se actualizara este listado con la información acumulada desde la última actualización.

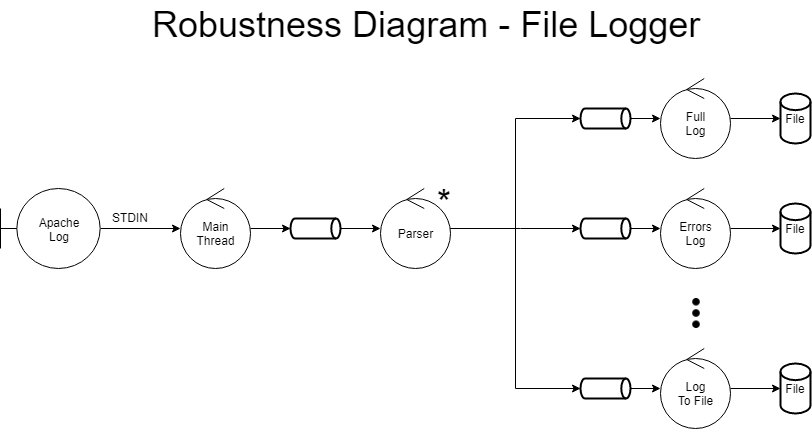
Suposición 5 – Es despreciable la perdida de mensajes aun en memoria al cerrar la aplicación

En caso de tener que apagar la aplicación por cualquier razón, se considera que no es necesario hacer una descarga de los mensajes recibidos pero aun no procesados o persistidos. Ya que el volumen de información crece con el paso del tiempo, la perdida de estos pocos datos no resulta significativa para el análisis del funcionamiento del servidor.

Diseño

A partir de los requisitos y las suposiciones planteadas se armó el diseño de la aplicación que se explicará a través de algunos diagramas.

En primer lugar se puede separar la estructura de la aplicación en una parte de recibido y extracción de campos de líneas de log, y una segunda que consiste en realizar alguna acción con los campos extraídos. Según los requerimientos existen tres posibles acciones que son: la actualización de estadísticas temporales, el logeo a archivos de log totales o de errores, la actualización de un ranking permanente de errores.



Este diagrama muestra el camino