Administración de Sistema

Agrupador
Indexador
Buscador

Índice de contenido

Introducción

En este documento se describen los módulos de agrupamiento, indexamiento y motor de búsqueda, así como su instalación y labores sencillas de mantenimiento y ajuste. También se describe el proceso de reinicio del agrupamiento, o agrupamiento inicial de todos los datos de la base de datos.

Módulos

Agrupador

El módulo de agrupamiento es el encargado de realizar la evaluación y clustering de grupos similares para agrupar datos de los contactos utilizando el sistema de Células controladas por un Iniciador. El agrupador reside en el directorio **core_java/** y consta de una serie de clases en java. A continuación describo brevemente los principales archivos de este módulo.

core_java/Initiator.java : Este es el programa principal y lanzador del sistema de células. Recibe como parámetros un archivo de configuración con los datos de conexión a la base de datos, el número de contactos que debe procesar cada célula, el porcentaje mínimo de contactos antiguos (ya procesados) que será incluido en cada célula, el número de células que se lanzaran y un archivo para escribir su log. El iniciador trabaja en iteraciones, lanzando las células para que trabajen en paralelo procesando una cierta cantidad de grupos en cada fase del proceso. Entre iteraciones, el iniciador verifica si se ha dado la orden de parada en la base de datos para no lanzar las células en esa iteración y detener el proceso.

core_java/obs/grouping/ProcessingCell.java : Esta clase representa un thread de procesamiento del algoritmo de agrupamiento, lo que es llamado, Célula de Procesamiento. Cada célula de procesamiento utiliza un objeto de tipo **`Interface`** para comunicarse con la base de datos de forma sencilla.

core_java/obs/comm/Interface.java : Esta clase encapsula varios métodos de comunicación y obtención de datos usados por **`ProcessingCell`**. Pide y recibe datos a un objeto de tipo **`Communication`** que almacena los métodos de comunicación con la base de datos particular. De este modo, al cambiar la base de datos puede adaptarse el modulo de comunicación sin interferir con la lógica de la clase **`Interface`**, ordenamiento, preprocesamiento u otras labores. Uno de los elementos que deben ser notados, es que esta clase entrega a **`ProcessingCell`** las máquinas en las que se encuentran los servicios de búsqueda activos para cada usuario. Esta información podría ser obtenida desde la base de datos, pero en la versión actual, esta información se encuentra directamente en el código. El método **`Interface.getUserMachines(int user_id)`** entrega los pares (IP, puerto) de los servicios activos, y esa información debe ser ajustada si cambian los servicios.

Un ejemplo típico de lanzamiento de este módulo seria el siguiente:

```
> java Initiator /root/datos/pc.config 5000 20 4 /root/logs/log_iniciador.txt
```

Esta línea (normalmente lanzada dentro un screen) inicia el proceso de agrupamiento con los datos de **`/root/datos/pc.config`** para la conexión a la base de datos, cargando 5000 grupos en cada célula de los que un 80% serian contactos nuevos y un 20% de contactos ya procesados en el pasado, y utilizando 4 células (threads) por iteración. El programa escribirá datos sencillos en cada iteración en el archivo **`/root/logs/log_iniciador.txt`**. El formato del archivo de conexión a la base de datos **`/root/datos/pc.config`** es:

```
localhost     ### host de la base de datos
monkeycontact ### base de datos usada
user          ### usuario para la conexión
pass          ### clave para la conexión
```

El módulo de agrupación, en cada paso, escribe en el archivo de salida el número de la iteración, el número total de grupos, el tiempo de la iteración en segundos, y el número de contactos que quedan en la base de datos sin ser procesados (es decir, los que faltan por procesar).

Indexamiento

El módulo de indexamiento es el encargado de procesar el texto de los contactos, construir los índices invertidos por usuario, guardarlos en archivos de texto e informar a los servicios activos para que actualicen los datos de los usuarios modificados. Las clases principales de este módulo son las siguientes.

core_java/CollectionGenerator.java : Este es el componente principal del sistema de indexamiento. Funciona por iteraciones, tomando una cierta cantidad de usuarios que han sido marcados por el agrupador pues sus datos han sido modificados, y por ende deben ser indexados y actualizados en los servicios de búsqueda. Entre iteraciones revisa la marca de para en la base de datos de modo similar al agrupador, y en caso de marca afirmativa, deja de procesar y detiene el proceso retornando en forma segura. Es importante notar que, como en el caso del módulo anterior, las maquinas (pares ip, puerto) en los que reside cada servicio de búsqueda activo, podría ser leído desde la base de datos. Sin embargo, en la implementación actual dicha información esta escrita directamente en el código, por lo que el módulo debe ser ajustado y recompilado cuando cambian los servicios o máquinas de búsqueda. Esta información se encuentra en el método **`generateCollection(...)`** en la fase de generación de la variable servicios_usuarios que guarda en una estructura, los varios puertos asociados a cada ip, para cada usuario que en proceso.

core_java/obs/comm/ConexionMC.java : Este componente tiene todos los métodos de comunicación con la base de datos que utiliza el indexador.

Un ejemplo típico de comando para activar este módulo es el siguiente.

```
> java -Xmx4096m CollectionGenerator /root/datos/pc.config
/root/datos/indices_usuarios/indice /root/datos/indices_usuarios/grupos
```

Este comando (normalmente lanzado desde un screen) activa el indexador configurando su base de datos con el archivo **`/root/datos/pc.config`**, generara los archivos de índices invertidos en las rutas **`/root/datos/indices_usuarios/indice_{userid}.ii.txt`** (agregando el userid de cada usuario) y su correspondiente archivo con información de grupos en **`/root/datos/indices_usuarios/grupos_{userid}.txt`**. El archivo de configuración de la base de datos es idéntico al usado por el Agrupador.

Motor de Búsqueda

El módulo del motor de búsqueda es el sistema de respuesta para las consultas usando índices invertidos por usuario. Además el sistema funciona con un conjunto de servicios de búsqueda como procesos en paralelo. Estos servicios pueden ser de tipo Vocabulario, de enriquecimiento de consulta, o de Índice, que responde consultas ya enriquecidas.

La interfaz de consulta (probablemente un PHP) escoge al azar un par de servicios activos, siempre uno de Vocabulario y otro de Índice, para realizar el proceso de consulta. Por simplicidad, se han escogido puertos particulares para los servicios (por ejemplo, 30001, 30002, 30003, y similares para los servicios de índice, y 31001, 31002, 31003 y similares para los de vocabulario), pero los puertos usados no tienen importancia para el sistema siempre que todos los módulos involucrados los conozcan. Es importante notar que tanto el Agrupador como el Indexador deben tener la información de qué máquinas tienen servicios activos y en que puertos para funcionar adecuadamente.

Los dos binarios que componen este modulo son activar_servidor_vocabulario y activar_servidor_consultas. Ambos permanecen activos escuchando consultas (tanto queries normales de su tipo para responder, como ordenes de actualizar datos). Los comandos típicos para lanzarlos son lo siguientes.

```
> /root/buscador_cpp/bin/activar_servidor_vocabulario 31001
/root/datos/indices_usuarios/indice_0.ii.txt 0
```

Con este comando se lanza un servicio de Vocabulario que escuchara en el puerto 31001, y carga inicialmente el índice **/root/datos/indices_usuarios/indice_0.ii.txt**. Puede recibir cualquier índice válidamente construido para la carga inicial, los demás índices deben ser enviados uno a uno con el mensaje de actualización del Indexador u otro programa adecuado. El 0 al final es una marca que indica que el índice que se le está entregando para la carga no es binario, sino en texto.

La función de carga de índices binarios está desactivada en la versión actual, pues los archivos que genera el indexador en esta versión son en texto, pero la marca se conserva en las llamadas pues esa función podría ser activada sin grandes dificultades.

```
> /root/buscador_cpp/bin/activar_servidor_consultas 30001
/root/datos/indices_usuarios/indice_0.ii.txt
/root/datos/indices_usuarios/grupos_0.txt 0
```

De forma similar, este comando activa un servicio de Índice que escuchara en el puerto 30001. Carga los datos de un usuario inicial por medio de los archivos de índice **/root/datos/indices_usuarios/indice_0.ii.txt** y con información de grupos en el archivo **/root/datos/indices_usuarios/grupos_0.txt**, ambos en texto (como indica la marca 0 final del comando). Como en el caso del servicio de vocabulario, la información de los demás usuarios debe ser entregada uno a uno a este servicio por el indexador u otro programa adecuado.

Luego de lanzar ambos servicios (y de hecho todos los que se desea activar) es conveniente utilizar un programa para cargar los datos de todos los usuarios. Desde que los servicios son lanzados (e incluso durante la duración de la carga de datos) los servicios pueden responder consultas, sin embargo, responderán sin resultados para aquellos usuarios que aún no han sido cargados. Hay un programa construido para esta carga masiva de datos en **core_java/UpdateServices.java**. Este programa puede realizar la carga de datos para un usuario particular o para todos los usuarios del sistema. Sin embargo, este programa está pensado para actualizar datos de un nuevo servicio habiendo otros activos, o para levantar los servicios luego de una caída o actualización, no está pensado para un lanzamiento inicial de los servicios por lo que no crea los archivos de índices invertidos. Para que se puedan actualizar los datos de los usuarios, dichos índices deben haber sido previamente creados por el indexador. Si esos archivos no existen, entonces es necesario indexar todo el texto de la base de datos y el método para hacer eso es hacer correr el Indexador con todos los usuarios del sistema, módulo que automáticamente se comunica con los servicios para actualizar los datos cada vez que crea un nuevo índice.

El programa UpdateServices.java se utiliza de la siguiente manera.

```
> java UpdateServices /root/datos/pc.config /root/datos/indices_usuarios/indice
/root/datos/indices_usuarios/grupos 0 localhost
```

Donde se le pasa un archivo de configuración para la conexión a la base de datos, tal como en el caso del Agrupador e Indexador, se le pasa la ruta base a los archivos de índice y de grupos (a los que agregara el userid y extensión, como en casos anteriores), el 0 en este caso indica que se procesen todos los usuarios (un número mayor que cero será interpretado como userid de un único usuario para actualizar sus datos) y la máquina que aloja los servicios para la comunicación. Es importante notar que este programa no recibe los puertos como parámetro, están en el código mismo del programa (por lo que debe ser ajustado y recompilado si hay cambios en los servicios activos). Los métodos que deben ser ajustados para incluir los puertos de los servicios activos son **`updateForSingleUser(...)`** y **`updateForUsers(...)`**. El programa es bastante simple y puede ser modificado para que reciba los puertos y varias máquinas como parámetro o leyendo un archivo de configuración adicional.

Detención de Módulos

Los servicios del Buscador pueden ser detenidos sin problemas entrando al screen que los contiene y terminando su proceso. Sin embargo, tan pronto como los servicios se detienen, los clientes que intenten comunicarse con ellos fallarán, por lo que es recomendable siempre ajustar los clientes PHP para que utilicen un nuevo conjunto de servicios antes de detener los servicios que se desea bajar.

Los módulos de Agrupamiento e Indexamiento, por otra parte, deben ser detenidos por medio de una orden en la base de datos. Para ello, basta con definir en 1 el valor del campo **`stop`** en las filas correspondientes de la tabla **`data_control`**. Por ejemplo, la orden sql:

```
> update data_control set stop = 1 where module = “ProcessingCell”
```

Indica al Agrupador que se detenga, mientras que **`module = “CollectionGenerator”`** dará la orden al Indexador de parar. Luego de que los campos de la base de datos tengan la orden de parada, debe esperarse a que la iteración actual de cada módulo termine para que revisen sus correspondientes órdenes de detención y paren. Esta es la forma correcta de detener a dichos módulos. Si los programas son terminados en forma abrupta, podrían dejar datos inconsistentes en la base de datos o en los archivos que generan. Revisando la salida de ambos programas en sus correspondientes screen luego de dar la orden de detención, puede verificarse que han terminado en forma segura.

Agrupamiento Inicial

Este proceso puede ser utilizado cuando se desea el reprocesamiento o procesamiento inicial de todos los contactos de la base de datos. El proceso destruye todos los datos de grupos creados y se encarga de preparar la base de datos, realizar las labores iniciales de agrupamiento, procesar todo el texto para la generación de índices, y deja el sistema listo para luego activar los servicios del Buscador, el Agrupador, y el Indexador de forma normal.

Para realizar este proceso debe detenerse tanto los módulos de Agrupación como de Indexamiento. Existe un script para realizar esta labor en el directorio **`agrupacion_inicial`**, llamado **`s_agrupacion_inicial.sh`** que automatiza el proceso. En ese script, sin embargo, debe definirse el id inicial que se usará para grupos, reseteando el valor de **`AUTO_INCREMENT`** de la tabla **`as_groups`** (en la versión actual del script, se utilizó el valor 20000000 para diferenciar los grupos nuevos de potenciales grupos previos al ultimo uso de este método).

Instalación del sistema

Para la instalación del sistema, basta con copiar el código de los diferentes módulos a la o las máquinas objetivo, ajustar los parámetros que dependen del número de servicios de búsqueda que se usarán y sus pares (IP, puerto), compilarlos y lanzar cada proceso por separado. Los servicios de búsqueda siempre deberían activarse antes que los demás, pues el Indexador se comunica con ellos para actualizar sus datos entre iteraciones. Si los servicios que se han configurado aún no están activos, el Indexador fallara al intentar comunicarse con ellos para actualizarlos (lo que, si bien no provoca la falla del módulo de indexamiento, arroja varios mensajes de advertencia y deja usuarios con datos sin actualizar).

Los directorios que contienen el código necesario son:
core_java/ : que contiene los módulos de Agrupamiento, Indexamiento y algunas aplicaciones adicionales como UpdateServices.
buscador_cpp/ : que contiene el Buscador, incluyendo los servicios de Vocabulario y de Índice.
agrupacion_inicial/ : que contiene algunas aplicaciones para el proceso de Agrupación Inicial.

Debe definirse que máquinas contendrán servicios de búsqueda y los puertos que serán usados. Esta información debe ser agregada al Agrupador y al Indexador modificando los siguientes archivos.

core_java/obs/comm/Interface.java : Este módulo del Agrupador, encargado de la lógica relacionada a la toma de datos, define las máquinas que contienen servicios y que deben ser actualizados cuando cada iteración de agrupación termina. El método **`public List<Machine> getUserMachines(int user_id)`** debe ser ajustado para incluir los pares (máquina, ip) de cada servicio que será utilizado.

core_java/CollectionGenerator.java : El indexador también debe ser ajustado para que se comunique con los servicios que se planea mantener activos de modo de actualizar su información cada vez que se procese a un usuario. En dicho programa, debe ajustarse el método **`CollectionGenerator.generateCollection(...)`** en la fase de generación de la variable servicios_usuarios que define los pares ip, puerto de los servicios usados.

Con el Agrupador e Indexador ya modificados para conectarse con los servicios que serán usados, deben prepararse y lanzarse los servicios de búsqueda. Para ello, deben compilarse los servicios en el directorio buscador_cpp/ del siguiente modo:

```
> make activar_servidor_vocabulario activar_servidor_consultas
```

Esto creará ambos binarios. Para lanzarlos, es recomendable crear un screen para cada servicio de similar a:

```
> screen -S serv_voc_1
```

Esto creará un screen de nombre serv_voc_1 (para facilitar su identificación), y desde éste puede lanzarse el correspondiente servicio con un comando como:

```
> /root/buscador_cpp/bin/activar_servidor_vocabulario 31001
/root/datos/indices_usuarios/indice_0.ii.txt 0
```

Esto activara al servicio de vocabulario y lo dejara corriendo con un indice inicial, y en espera de la carga de datos adicionales o enriquecer consultas. Para el servicio de índice, podrían usarse los comandos:

```
> screen -S serv_index_1
> /root/buscador_cpp/bin/activar_servidor_consultas 30001
/root/datos/indices_usuarios/indice_0.ii.txt
/root/datos/indices_usuarios/grupos_0.txt 0
```

Teniendo todos los servicios que se desea usar activos y en espera, podría realizarse la carga masiva de datos de usuarios, si sus archivos ya existen. Para eso, como se explicó antes, puede usarse el programa UpdateServices.

Por último, teniendo ya ajustados tanto el Agrupador como el Indexador, se pueden compilar con:

```
> javac Initiator.java
> javac CollectionGenerator.java
```

Con los módulos ajustados y compilados, y con los servicios activos, pueden activarse del modo usual con comandos como:

```
> screen -S ProcessingCell
> java Initiator /root/datos/pc.config 5000 20 4 /root/logs/log_iniciador.txt
```

Para activar el agrupador y:

```
> screen -S CollectionGenerator
> java -Xmx4096m CollectionGenerator /root/datos/pc.config
/root/datos/indices_usuarios/indice /root/datos/indices_usuarios/grupos
```

Lo que deja corriendo el Indexador.

Como un paso final, puede dejarse activo algún sistema de verificación de servicios que, cada cierto, revise si los sistemas están activos y, en caso de contrario, relanzar los servicios caídos. Un programa que cumple ese papel es el script **probar_servicios_v2.php** del directorio **buscador_cpp/**. Ese programa, que puede activarse cada lapsos regulares por medio del crontab, revisa que los programas del buscador estén activos en la maquina en la que se aloja. Si encuentra programas inactivos, los relanza y luego de esperar unos segundos a que se activen, lanza el programa UpdateServices para actualizar sus datos. Es importante notar que este programa también debe ser ajustado acorde con los servicios activos.

Un ejemplo de linea de activación del programa de prueba de servicios en el crontab es:

```
### CHECK DE SERVICIOS
0-59/5 * * * * /usr/bin/php /root/buscador_cpp/probar_servicios_v2.php >>
/root/logs/check_services.log
```