Protocolo de información sobre la carga computacional de cluster’s

Resumen.

El presente trabajo abarca la creación de un protocolo sobre la carga de recursos computacionales de un conjunto de host (cluster). Este protocolo se encuentra en la capa de aplicación.

Se recorre una breve reseña a la historia de los clusters abordando la problemática, vemos diferentes tipos de cluster, características que en ellos se espera y software middleware que hacen ver el todo como uno. Es decir, el conjunto de host débilmente acoplados como si fuese una sola súper-computadora y como conduce al desarrollo de este trabajo. Yendo a lo primitivo de saber con que recursos cuenta cada nodo y como mantener la información de este conjunto para que las aplicaciones middleware o finales puedan acceder a dicha información para su uso.

Este protocolo llamado PIRLH fue creado en vista de satisfacer las necesidades personales de conocimiento y base para su desarrollo, como se verá, los resultados obtenidos mas acotados a los que se puede ir profundizando la complejidad como conclusión.

Reseña de la historia de los cluster

Comenzaremos a definir que es un clúster; Por este se entiende a un grupo de hosts (computadoras) que se encuentran interconectadas por medio de una red subyacente dando la sensación de estar frente a una sola unidad, su vista al usuario o programa usuario es una visión monolítica o sistema único. Esto es lo más representativo del concepto.Si bien el concepto clúster en informática no es nuevo, data de los años 1950 pensado para el procesamiento en paralelo primero pensado dentro de un solo host, luego en 1967 en los laboratorios IBM, Gene Amdahl publicaba la ley de la aceleración de como un procesamiento en paralelo realizaba una tarea. A finales de los 70 se desarrollo el primer producto de tipo clúster ARCnet. Igualmente la práctica de este concepto no tuvo frutos hasta los años 1990 con el bajo costo de las computadoras personales en contrapartida con los grandes centros de procesamiento. En esa década nace el primer súper-ordenador conformado por múltiples PC's llamado **Beowulf** construido por la NASA (Administración Nacional de Aeronáutica y del Espacio) mas precisamente desarrollado por el CESDIS (Centro de Excelencia en Datos del Espacio y Ciencias de la Información), Sus creadores Thomas Sterling y Don Becker. La premisa era usar hardware y software bien conocido y accesible en cuanto a lo monetario (commodities) para realizar procesamiento en paralelo.

Hoy en día todas las grandes corporaciones tanto gubernamentales y privadas utilizan esta técnica de clustering para su uso comercial o de investigación, sus razones son la prestación que hacen con las siguientes configuraciones: Alto rendimiento, Escalabilidad, Alta disponibilidad y Balanceo de carga.

Los clústers más grandes se publican y entran en un ranking TOP 500 donde están publicados en orden descendente según su potencial; inicialmente se tenía en cuenta para esta medición los FLOPS (Operaciones en Coma Flotante por Segundo) por sus siglas en ingles; Hoy se tienen en cuenta un conjunto de parámetros. Las características de estas supercomputadoras se pueden observar en www.top500.org.

Ya hemos descripto que es y para que sirve; Ahora bien un clúster por si solo no tiene fin específico y las aplicaciones finales que lo utilizan no deberían preocuparse por la conformación y estructura del cluster que esté usando. Existen para ello herramientas software llamados Middleware que actúan entre el sistema operativo y la aplicación para darle la sensación de estar frente a una sola computadora; estos se encargan del mantenimiento y optimización del clúster con las cuatro características (Balanceo de carga, Alto rendimiento, Escalabilidad y Alta disponibilidad). Uno de estos software's es OpenMOSIX.

Motivación

La motivación para el desarrollo de este trabajo nace por medio de la inquietud del autor del presente artículo. En el año 2010 armando un Clúster Mysql a modo de auto aprendizaje y práctica. Surgieron algunas preguntas: ¿Cómo hace el clúster para saber donde llevar la información?, ¿Qué criterios utiliza para indicarlo?, ¿Qué información procesa? ¿Qué datos se tienen en cuenta? ¿Que sucede si un nodo deja de estar operativo?. Estas son algunas de las muchas preguntas que han motivado este trabajo.

El protocolo se encuentra situado en la parte mas baja de esta arquitectura, precisamente se escribió un protocolo en la capa de aplicación que se ha llamado PIRLH

Estado del arte

En el comienzo de las preguntas y de pensar la idea de como hacer para obtener los recursos de cada host, naturalmente recurrí a buscar estas preguntas en Internet. Como en muchas ocasiones, la idea que había pensado si bien con algunas variantes ya había sido realizada. En primer lugar y como las preguntas nacieron a partir del armado de un clúster Mysql, se consulto en principio a los motores de base de datos como implementaban los protocolos. Mysql en su arquitectura, la información de recursos disponibles al ser un paquete privativo no se encontró el funcionamiento de este protocolo si bien hace balanceo de carga de a pares, con requerimientos bien establecidos, no se sabe si tiene un protocolo de información sobre los recursos disponibles en su totalidad de cada host. Por parte de otro de los motores de base de datos en este caso un producto libre, PostgresSql con su paquete pgpool-II se encarga del balanceo de carga, replicación entre otras tareas, no se encarga directa y únicamente a brindar solo la información de cada host.

En la búsqueda se encontró un protocolo de las cualidades que concordaba con la idea y compartían muchas de las características y otras disimiles. Es el único encontrado en el proceso, si bien no debe ser el único existente, alguna otra universidad/entidad gubernamental/privada habrán desarrollado su propio protocolo, no hay información sobre su existencia. El protocolo encontrado con el nombre de

Ganglia desarrollado en la universidad de Berkeley en modalidad open-source bajo el nombre de proyecto Millenium iniciado en el 2001 por Matt Massie, entre otros. Es un proyecto en su etapa madura utilizado por el gobierno y universidad.

Desarrollo

1. Proceso seguido y descripción para el desarrollo

El proceso seguido para transformar la idea en proyecto; fue el análisis en la etapa temprana construir un protocolo acorde la idea de obtener y procesar la información de los recursos. Luego de depurar los estados mediante una maquina de estados finitos se comenzó a otorgar las responsabilidades a cada parte cliente y servidor, hubo un cambio de paradigma al momento de realizar el trabajo en el tipo de difusión a utilizar (Broadcast / Multicast) esto hizo cambiar parte del desarrollo pero con un bajo costo.

Previo al desarrollo, teniendo en mente que y como presentar los datos, se evaluaron dos situaciones, tanto para la obtención de recursos computacionales de los hosts como para la presentación de los datos en el servidor. Para la parte de obtención de recursos o bien se podría obtener mediante los archivos del sistema dado que el protocolo funciona en ambientes POSIX y todo en estos sistemas es un archivo; esto llevaba a tener que invertir tiempo en crear funciones de parseo para cada uno de los archivos para obtener los datos que se necesitan en contraste la otra opción era utilizar la librería glibtop que utiliza estos archivos para la captura de información y ya tiene una estructura con lo cual el tiempo a invertir en codificar y en aprendizaje era muy dispar; no obstante no hay detrimento en el procesamiento dado que se realiza bajo los mismos archivos y es una librería estable. Para el caso de la presentación de datos tanto en el servidor y el cliente dado que el flujo de información que se envía entre ambos es en formato XML, se estaba en la misma situación o se escribían las rutinas para parsear la información con el costo en tiempo que conlleva o se trataba de invertir tiempo en la búsqueda de una librería para tal propósito, por lo consiguiente se encontró la librería xmlGlib. El aprendizaje en contraposición con el tiempo de invertir en desarrollo era mucho menor, por este motivo se utilizó la librería en cuestión. No obstante para ambos casos hubo que realizar código de parseo pero para cuestiones específicas. Hasta aquí hay dos problemas resueltos y se opta por usar las librerías.

Luego se efectuó todo el análisis del programa cliente y el servidor para cada uno su diagrama de flujo correspondiente. De esta forma ver que parte contendría cada código se siguió con el proceso de modularización con la metodología top-down del código para su reutilización.

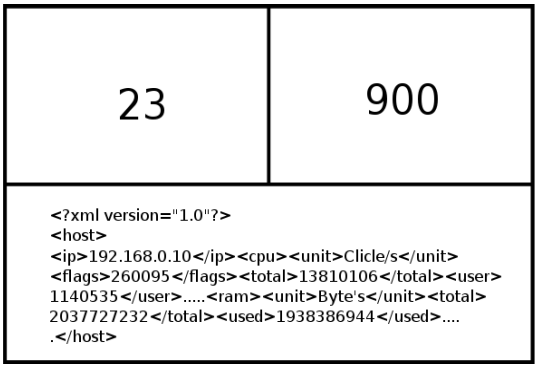
2. Protocolo y maquina de estado finito

Estructura del protocolo

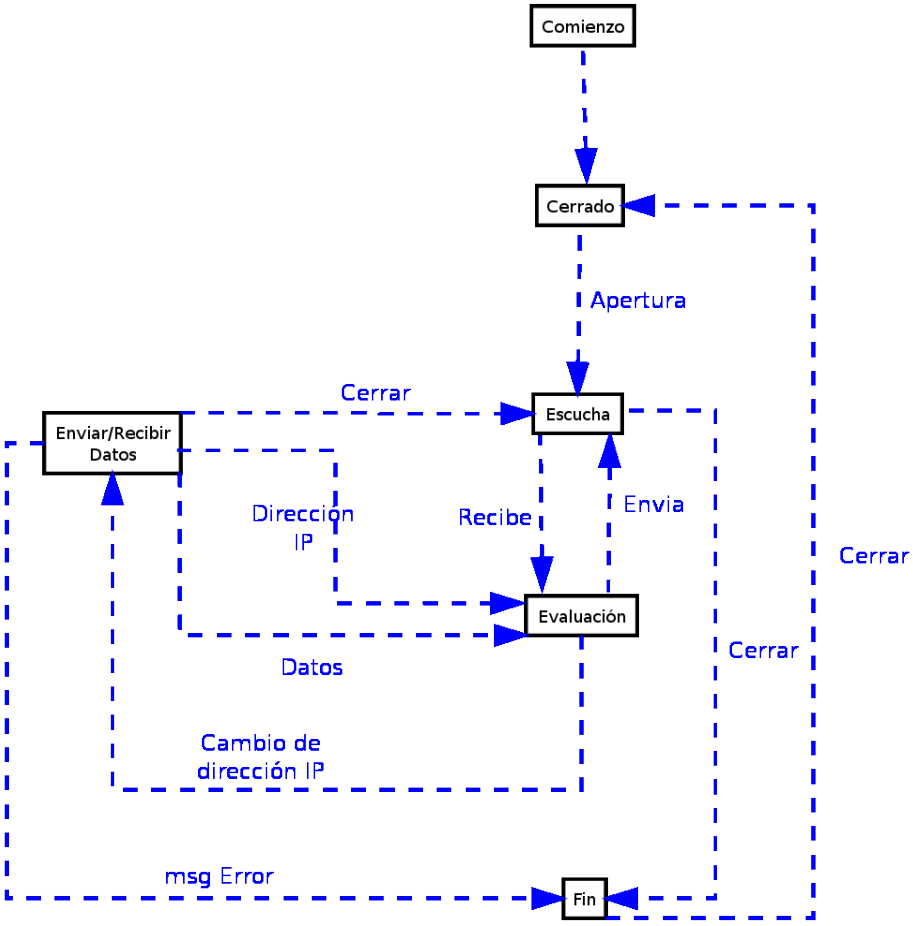
```
struct pirlh {
  char operation;
  int16 leng;
  char msg[2000];
}
```

Mensaje

Ejemplo



Maquina de estado finito



En cuento a la estructura del protocolo, esta estructura se empaqueta en un datagrama UDP lo cual comunican al cliente y servidor, este paradigma se elije para no causar mas trafico que el necesario en el cluster para la mensura de los recursos. Como tal datagrama no es orientado a la conexión y no es de entrega confiable, una de las actividades del servidor es enviar un eco para saber si un nodo esta activo. Ambos extremos están diseñados con tiempos de tolerancia, cada host tiene un tiempo de envió dentro del marco temporal (ventana) que el servidor espera los datos, por ejemplo, vamos a suponer que el servidor espera paquetes por 4 segundos, y la configuración de tiempo de envió de los host (esto es igual para todos en principio) es de un segundo; En este caso es muy probable que haya colisiones mientras mas host

haya en el cluster. Por esta cuestión se genera para cada host un delta tiempo así se disminuye la tasa de colisiones; otra de las razones es que un host al enviar sus recursos puede enviar otro paquete en ese tiempo, esto esta limitado por un temporizador, pero nada nos asegura que no vuelva a enviar un paquete en la misma ventana de tiempo, para ello lo que hace el servidor al encontrar otro dato de ese mismo host es desestimar uno de ellos. El restante tiempo se deja un delta tiempo al servidor para procesar todos los documentos requeridos tanto el maestro (que es temporal para poder generar los de recursos) como los separados por recurso.

La descripción de la estructura del protocolo es la siguiente:

Hay un campo *operation* de 1 Byte que indica que tipo de operación se debe hacer con ese mensaje que se encuentra encapsulado en el campo *msg[2000]* y que tiene una longitud de *leng* medida en 2 Bytes. Ahora explicada la estructura el protocolo seria el siguiente:

Se inicia cerrado, se genera la *apertura* y *escucha* en el servidor, este espera la llegada de los datagramas en el estado *escucha* Al recibir un datagrama realiza una *evaluación* si existe un nodo con la misma dirección ip y diferente MAC Address se envía una señal de desconexión al cliente con el mensaje solicitando el cambio de dirección IP con un código de operación 23, en este caso pasa al estado *fin* y consecuente estado *cerrado.* Si por el contrario no se encuentra la dirección IP y su código de operación es 01 entonces el servidor envía los parámetros de tiempo y direcciones actualizados al cliente con el código 20 para que guarde en su archivo de configuración y los utilice en la sesión actual, si el código de operación que llega al servidor es 03 entonces se procesa los datos de recursos contenido en el capo de mensaje el cual el servidor guardara temporalmente en el documento maestro para luego separarlo por recurso .

Luego del tiempo de ventana de recepción de datos, si la lista de nodos activos y los que no enviaron datos en ese período no coincide el servidor envía un eco código 02 con lo cual el cliente responde al mismo con el código 22.

Este ciclo se mantiene mientras este funcionando. Cabe destacar que en un principio el primer datagrama enviado por el cliente es por multicast, luego de recibir la respuesta de quien es el servidor ya se conecta directamente a él mediante su dirección IP sin anunciarse a todos y optimizando el trafico de la red de cluster.

Diagrama de flujo Servidor

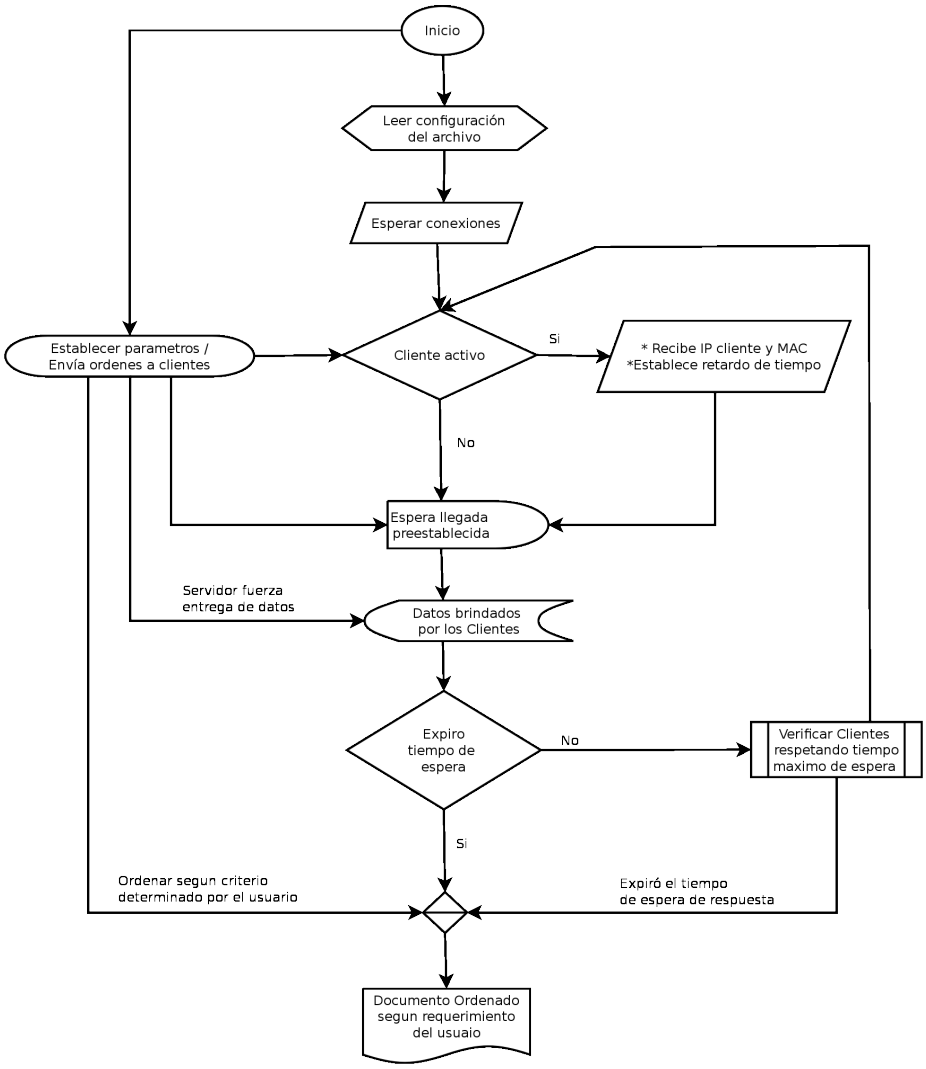
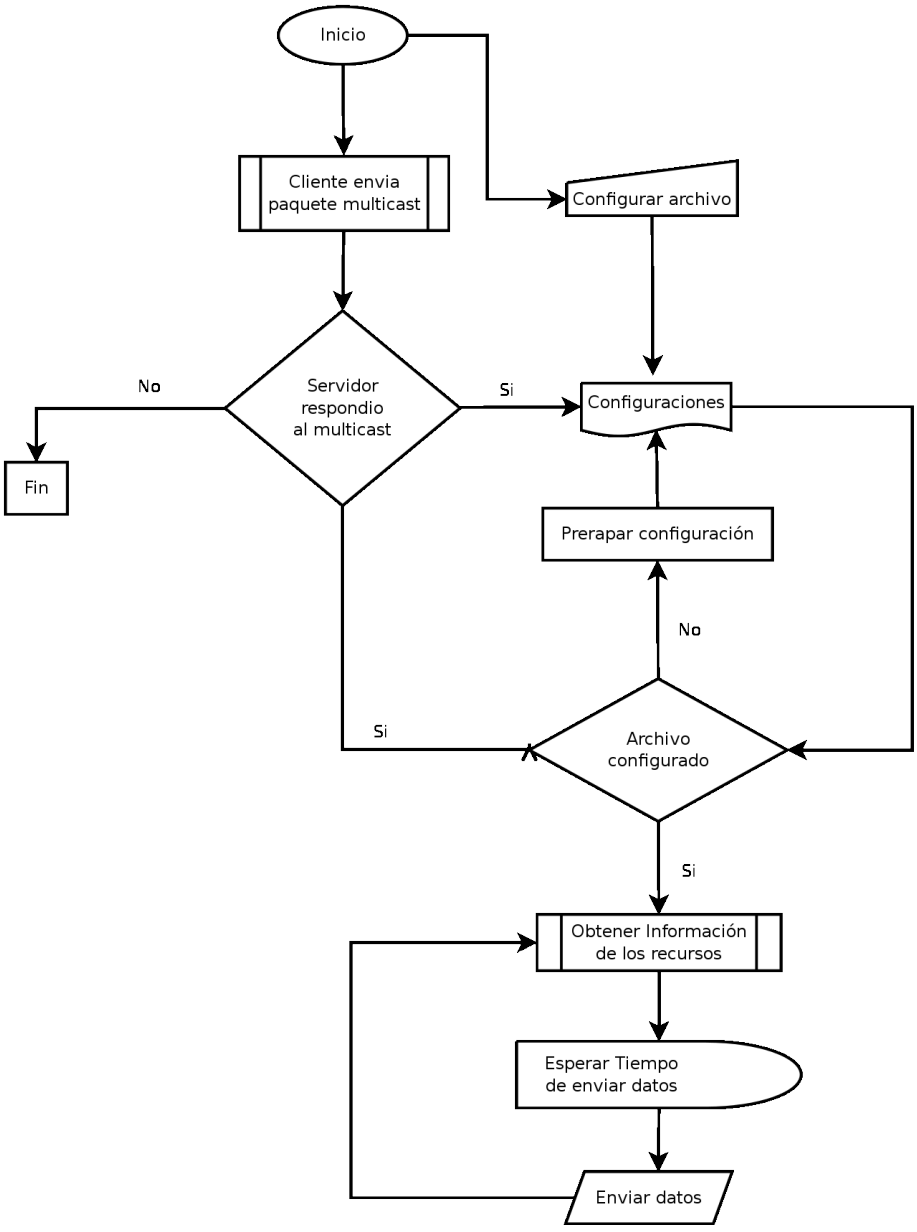


Diagrama de flujo Cliente



A grandes rasgos se separan en los siguientes archivos más representativos:

estructuras.h

1. Este archivo contiene definiciones y estructuras que utilizan ambas partes, tanto cliente como servidor. Esta contiene la estructura en la que se basa el trabajo, el protocolo PIRLH (este se describirá luego)

listas.h, listas.c

2. Estos archivos contienen los módulos/funciones de procesamiento de las distintas listas que disponen tanto los clientes como el servidor. Almacenamiento de nodos para la verificación de nodos únicos, lista con los datos de cada host y recursos, etc.

funciones.c, funciones.h

3. Estos archivos tienen las funciones/módulos en común para la utilización de las diferentes funciones para la presentación de los datos y librerías ya mencionadas. Conjuntamente con las listas y las librerías de XMLLIB y LIBTOP se generan todos los documentos a presentar.

recursos.c, recursos.h

4. estos archivos tienen las funciones para la obtención de los recursos del sistema utilizados solamente por los clientes, es un archivo muy sencillo.

conf_cliente.c/h,conf_servidor.c/h

5. Estos dos archivos son para los archivos de configuración de clientes y servidores, se llaman mediante el paso de parámetros en las aplicaciones tanto cliente como servidor. Mediate la consola, aquí se especifican datos como la direción ip, multicast, puerto, directorio, entre otros.

3. Herramientas utilizadas

Las herramientas utilizadas para el desarrollo se describen a continuación:

Librería LIBGTOP

Descripción: librería que contiene el paquete top del entorno Gnome .

Es utilizada para obtener la información y los datos de rendimiento del sistema. Si bien es posible obtenerlos por otros medios, este en cierta manera facilita el acceso.

Librería LIBXML2

Descripción: Esta librería nos da las herramientas para procesar documentos XML en su segunda versión de una forma ágil a tener que hacer las propias lecturas y parseos de datos.

Editor de texto

Como editor de texto el trabajo se realizo íntegramente con gedit, editor Gnome proporcionado por este paquete que viene por defecto en Ubuntu.

Compiladores

En cuanto al lenguaje el código esta enteramente escrito en ANSI C, la herramienta de compilación es el compilador GCC (GNU C Compiler) de la FSF (Fundación de Software Libre) cuenta con una variedad de lenguajes que puede compilar.

Gráficos

Para la realización de documentos de desarrollo y etapa previa se ha usado el entorno DIA tanto para los gráficos de diagramas de flujo de los programas como para el gráfico del protocolo.

Documentación

La documentación se ha realizado en el editor de texto latexila y exportados a formato PDF

Resultados

A lo largo de toda la etapa se ha afrontado al aprendizaje de un nuevo lenguaje y entornos de desarrollo desconocidos para uno. En vista a este esfuerzo entregado se consiguió el objetivo propuesto, es tener en funcionamiento un protocolo que informe los recursos de un cluster con el fin de facilitar y optimizar la toma de decisión de los sistemas de software que requieran uno o mas recursos en particular. Contestando los interrogantes que que dieron inicio a esta investigación. Se ha probado en un número reducido de nodos en mi hogar. Se resolvieron y depuraron fugas de memoria verificado en las etapas de test correspondientes.

Conclusiones

En vista que no existen tanto en el ámbito académico como comercial una gran variedad de protocolos específicamente abordando esta problemática, muchos de los mencionados en este documento utilizan variantes sobre el protocolo SNMP. Resulta de interés esta propuesta. Un ejemplo de un protocolo dedicado a esta problemática es el ya mencionado Ganglia. Resulta por un lado gratificante adentrarse en una temática poco estudiada o profundizada, por otro lado un cierto asombro dado que este

concepto de clúster se originó hacia fines de los 50’ ya han pasado seis décadas y no se observan la variedad que se esperaba encontrar de un protocolo de este tipo que no sean con tecnologías bien conocidas y sobre protocolos de administración.

Por parte de este protocolo PIRLH queda en una etapa temprana, sin embargo su estudio y visión seguirá desarrollándose. En base a lo que respecta al protocolo en concepción, quedan temáticas a abarcar que fueron base de la idea original como conceptos de alta disponibilidad dejando un clúster no dependiente de un solo servidor sino que pueda establecerse una dinámica en la automatización para escoger su propio nodo servidor en un tiempo dado, métricas a partir de su uso, como así también combinar varios clúster como visión de uno solo, temáticas que escapa al presente desarrollo por su envergadura del proyecto resultante.

4. Referencias

5. Concepto de clustering http://www.dei.uc.edu.py/tai2003-2/clustering/index.html
6. Mysql para desarrolladores http://dev.mysql.com/
7. PostgresSql para desarrolladores http://www.postgresql.org/developer/
8. Ganglia protocolo equivalente http://ganglia.info/
9. Librería glibtop para desarrolladores http://developer.gnome.org/libgtop/stable/
10. Librería xmllib para desarrolladores http://xmlsoft.org/
11. Oracle basado en SNMP http://docs.oracle.com