ARQUITECTURA VS MIDDLEWARE

Freddy L. Abad Leon

—————————— ◆ ——————————

**MONITOREO DE SISTEMAS CON ASTROLABE**

Este es un sistema que puede soportar el monitoreo general de sistemas distribuidos muy grandes. En el contexto de sistemas de autoadministración, Astrolabe es una herramienta general para observar el comportamiento de sistemas. Su salida puede utilizarse para alimentar un componente de análisis que decidirá sobre acciones correctivas.

Astrolabe organiza una gran colección de servidores en cierta jerarquía por zonas. Las zonas

de menor nivel constan de un solo servidor, las cuales posteriormente se agrupan en zonas de mayor tamaño. La zona de mayor nivel comprende todos los servidores. Cada servidor ejecuta un proceso Astrolabe, llamado agente, que colecciona información sobre las zonas en las que se encuentra ese servidor. El agente se comunica también con otros agentes con la intención de esparcir información de la zona a través de todo el sistema.

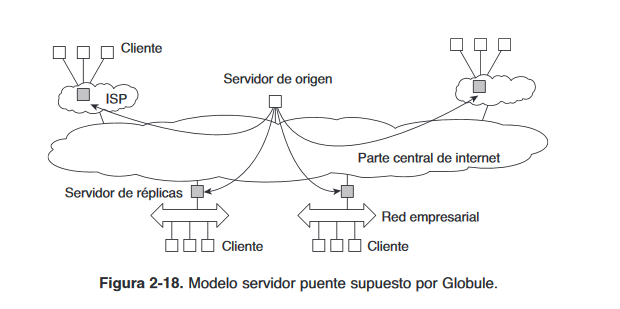
Cada servidor mantiene un conjunto de atributos para recopilar información local. Por ejemplo, un servidor puede rastrear archivos específicos que almacena, utilizar recursos, etc. Sólo aquellos atributos mantenidos directamente por un servidor, es decir, al nivel más bajo de la jerarquía, pueden escribirse. Cada zona puede tener una colección de atributos, pero los valores de éstos se calculan a partir de los valores de las zonas de más bajo nivel.

**DIFERENCIAS CON GLOBULE**

Globule es una red colaboradora de distribución de contenido. Globule se basa en servidores de usuario final colocados en internet, los cuales colaboran para optimizar el rendimiento a través de la replicación de páginas web. Para lograr esto, cada servidor de origen (es decir, el servidor utilizado para manejar las actualizaciones de un sitio web específico) rastrea patrones de acceso por página. Los patrones de acceso se expresan como operaciones de lectura y escritura para una página, y el servidor de origen para esa página registra cada operación y la marca con un tiempo.

En su forma más sencilla, Globule asume que internet puede verse como un sistema servidor

puente. En particular, asume que las solicitudes siempre pueden pasar a través de un servidor puente adecuado.



Este sencillo modelo permite a un servidor de origen ver qué ocurriría si colocara una réplica en un servidor puente específico.

Por una parte, colocar una réplica cerca de los clientes mejoraría la espera percibida por ellos, pero induciría tráfico entre el servidor de origen y el servidor puente para mantener una réplica consistente con la página original

Cuando un servidor de origen recibe una petición para una página, éste registra la dirección IP desde donde se originó la petición y busca al ISP o a la red empresarial asociada con la petición utilizando el servicio de internet WHOIS. El servidor de origen busca después al servidor de réplicas más cercano, que podría actuar como servidor puente para ese cliente,

y posteriormente calcula la espera para ese servidor junto con el ancho de banda máximo. En su configuración más simple, Globule supone que la espera entre el servidor de réplicas y la máquina del usuario solicitante es insignificante, y de igual manera asume que el ancho de banda entre los dos es enorme.

Una vez reunidas suficientes peticiones para una página, el servidor de origen realiza un sencillo “análisis qué sucede si”. Tal análisis se reduce a evaluar varias políticas de replicación, donde una política describe en dónde replicar una página y cómo mantenerla consistente. Cada política de replicación tiene un costo que puede expresarse como una función lineal simple



donde mn es una métrica de rendimiento y wn es una ponderación para indicar qué tan importante es dicha métrica. Las métricas de rendimiento típicas se agregan a los retrasos entre un servidor cliente y un servidor de réplicas cuando devuelven copias de las páginas web, el ancho de banda total consumido entre el servidor de origen y el de réplicas para mantener consistente a una réplica, y el número de copias devueltas a un cliente.

En Globule, un servidor de origen regularmente evalúa unas diez políticas de replicación utilizando simulación basada en rastreo para cada página web. A partir de estas simulaciones se elige la mejor política, y posteriormente se refuerza. Esto puede implicar que se instalen nuevas réplicas en diferentes servidores puente, o que se ha elegido una nueva forma de mantener consistentes a las réplicas. Coleccionar rutas, evaluar políticas de replicación, y el reforzamiento de una política seleccionada se efectúa automáticamente.

Existen unas cuantas cuestiones sutiles con las que aún tenemos que tratar. Hasta cierto punto, no queda claro cuántas peticiones tienen que recopilarse antes de que pueda realizarse la evaluación de la política en curso. Para explicarlo, suponga que en el tiempo Ti el servidor de origen selecciona la política para el siguiente periodo hasta Ti+1. Esta selección ocurre de acuerdo con una serie de peticiones previas que sucedieron entre Ti+1 y Ti. En retrospectiva al tiempo Ti+1, el servidor puede concluir que debe seleccionar la política p\* dadas las peticiones reales que sucedieron entre Ti y Ti+1. Si p\* es diferente de p, entonces la selección de p en Ti estuvo mal.

Como es evidente, el porcentaje de pronósticos erróneos depende de la longitud de las series

de peticiones (conocida como longitud de ruta) que se utilizan para pronosticar y seleccionar la siguiente política.

**ADMINISTRACIÓN DE LA REPLICACIÓN AUTOMÁTICA DE COMPONENTE EN JADE**

Cuando se da mantenimiento a clusters de computadoras, cada una compuesta por sofisticados servidores, resulta importante mitigar los problemas de administración. Un método que puede aplicarse a servidores construidos bajo métodos basados en componentes es detectar las fallas latentes en ellos y reemplazarlos automáticamente. El sistema Jade sigue este enfoque. Jade está construido sobre el modelo de componentes Fractal, una implementación de Java de un Framework que permite agregar y eliminar componentes al tiempo de ejecución. Un componente en Fractal puede tener dos tipos de interfaces. Una

interfaz de servidor utilizada para llamar métodos que se implementan mediante ese componente; y una interfaz de cliente que utiliza un componente para llamar a otros componentes. Los componentes se conectan uno con otro mediante interfaces de

vinculación. Una primitiva de vinculación significa que una llamada a una interfaz de cliente resulta directamente en llamar a la interfaz de servidor vinculada. En el caso de vinculación compuesta, la llamada puede proceder a través de uno o más componentes, por ejemplo, debido a que la interfaz de cliente y la interfaz de servidor no coincidieron y es necesario entonces implementar algún tipo de conversión. Otra razón puede ser que los componentes conectados se encuentren en máquinas diferentes.

Jade utiliza la idea de un dominio de administración de reparaciones. Tal dominio consiste

en cierto número de nodos, donde cada nodo representa un servidor junto con los componentes que éste ejecuta. Aparte existe un nodo administrador responsable de agregar y eliminar nodos del dominio. El nodo administrador puede replicarse para garantizar una alta disponibilidad.

Cada nodo está equipado con detectores de fallas, los cuales monitorean el bienestar de un nodo o de uno de sus componentes e informan sobre cualquier falla al nodo administrador. En general, estos detectores consideran cambios excepcionales en el estado de los componentes, el manejo de recursos, y la falla real de un componente. Esto puede significar, en realidad, que una máquina se ha estropeado.

Cuando una falla es detectada, se inicia un procedimiento de reparación. Tal procedimiento es dirigido por una política de reparación parcialmente ejecutada por el nodo administrador. Las políticas se establecen explícitamente y se llevan a cabo de acuerdo con la falla detectada. Por ejemplo, supongamos que se detectó la falla de un nodo. En ese caso, la política de reparación puede indicar que se realicen los siguientes pasos:

1. Finalizar toda vinculación entre un componente en un nodo sin problemas y un componente en el nodo que ha fallado.
2. Solicitar al nodo administrador que inicie y agregue un nuevo nodo al dominio.
3. Configurar el nuevo nodo exactamente con los mismos componentes que se encontraban en el nodo que falló.
4. Restablecer todos los vínculos que finalizaron anteriormente.

En este ejemplo, la política de reparación es sencilla y sólo funcionará cuando no se hayan perdido datos cruciales (se dice que los componentes arruinados son desplazados).

El método seguido por Jade es un ejemplo de autoadministración: conforme se detecta una falla, se ejecuta automáticamente una política de reparación para llevar al sistema como un todo al estado en el que se encontraba antes del problema. Al ser un sistema basado en componentes, esta reparación automática requiere de un soporte específico para permitir que se agreguen y eliminen componentes al tiempo de ejecución. En general, no es posible cambiar aplicaciones preexistentes en sistemas de autoadministración.

**BIBLIOGRAFIA**

1. Tanenbaum A.S., Van Steen Maarten , “Sistemas Distribuidos: Principios y Paradigmas”, <http://climate.cs.buap.mx/CondeJC/cursos/Material/Distribuido_/Lecturas/300_ARQ_MIDDLEWARE_AUTOADMIN.pdf>