Jose Carlos Roncero Blanco

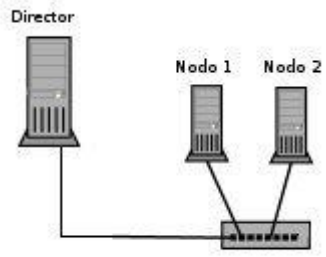
Elabora en un documento PDF como implementa soluciones de alta disponibilidad y balanceo de carga en servidores de bases de datos MySQL.

Suele pasar que nuestra base de datos tiene demasiada carga y te enfrentas a un eterno dilema, actualizar el hardware a un servidor más potente o adquirir hardware más económico y montar un cluster. En ambas situaciones nos enfrentamos a la decisión de tener que hacer una inversión económica y nos surge la duda de elegir la decisión correcta. Cuando hay dinero de por medio, no siempre es fácil para un administrador de sistemas convencer al jefe de cual es la solución “óptima”, a no ser que éste goce de plena confianza por parte de su jefe.

Pues bien, con este artículo intentaré plasmar algunas de las ventajas que tiene la opción dos, el cluster frente a hardware de última generación.

Lo primero que tenemos que plantearnos es lo siguiente, siempre tenemos que preveer que nuestra base de datos puede crecer hasta el punto de que tanto la carga del servidor como el espacio puede llegar a ser un problema. Por eso mismo, una configuración en cluster puede ayudarnos a solventar problemas en un futuro. Otra cosa a tener en cuenta, es la disponibilidad, pues para muchas empresas e instituciones, la caída de la base de datos puede suponer perdidas económicas cuantitativas, o el no poder ofrecer un tipo de servicio específico, estas y muchas otras situaciones problemáticas ya tipificadas que pueden generarnos un gran dolor de cabeza. En cuanto al espacio, dependiendo del crecimiento de la propia base de datos, quizás este problema sea fácilmente solventable añadiendo simplemente discos duros al servidor principal, pero incluso en muchos casos esto no es suficiente, dependiendo como he dicho de la magnitud de la base de datos.

Ahora bien ¿Cómo puede ayudar un cluster en este tipo de situaciones? De forma muy sencilla, por el clásico divide y vencerás. Una configuración en cluster consiste en una serie de servidores que trabajan al unísono como si fueran uno sólo, ofreciendo: alto rendimiento, alta disponibilidad, balance de carga y escalabilidad. Un esquema sencillo de este tipo de configuración usando un mínimo de tres servidores sería el siguiente:



Como podemos apreciar en la imagen, tenemos un servidor principal al que llamamos Director, ¿Cuales son sus funciones? Este servidor se encarga de gestionar el balance de carga entre los nodos y al mismo tiempo de comprobar que ambos nodos están activos (alta disponibilidad).

Normalmente, las aplicaciones que se encargan de comprobar la disponibilidad se les conoce como Heartbeart (latido de corazón), y su función principal es asegurarse de que al menos uno de los nodos esta activo para poder garantizar el servicio. Un ejemplo de software que se encarga de ofrecer este tipo de servicio es UltraMonkey. Hoy día, por suerte, también disponemos de soluciones bastante competentes como MySQL Cluster, que consta de las herramientas necesarias para que podamos configurar nuestro cluster para nuestra base de datos (con sus limitaciones) sin tener muchos problemas. Normalmente, en configuraciones corporativas, el balanceo de carga es llevado a cabo por el front-end del cluster, que son una serie de máquinas que se encargan de repartir las peticiones del cluster principal que se conoce como back-end. En nuestro caso, nuestro front-end sería solo la máquina Director y nuestro back-end los nodos 1 y 2, pero tengamos en cuenta que también podrían ser dos o varias las que se encargaran de este proceso. Un software que por desgracia ha dejado de desarrollarse que se encargaba de esta función de balanceo era OpenMosix.

De este modo, podemos sacar claramente las ventajas que tiene invertir un poco de tiempo y dinero en configurar nuestros servidores usando un modelo de Cluster. Puesto que el fallo de una de las máquinas (Nodos) no supondría el cese del servicio. Además, en cuanto a escalabilidad, simplemente tendríamos que añadir nuevos servidores a la configuración, tanto en la parte de balanceo como en los nodos, incluso, si el espacio empezara a convertirse en un problema, podríamos añadir un servicio de almacenamiento externo (distribuido).

Replicación en MYSQL

MySQL 5.1, la última versión del software al momento de este escrito, provee un nuevo soporte a una replicación basada en hilera (row). Las versiones previas de MySQL (5.0 e inferiores) apoyaron la replicación basada en declaración (statement).

Además, una combinación de la replicación basada en declaración y la basada en hilera pueden ser utilizadas en MySQL 5.1.

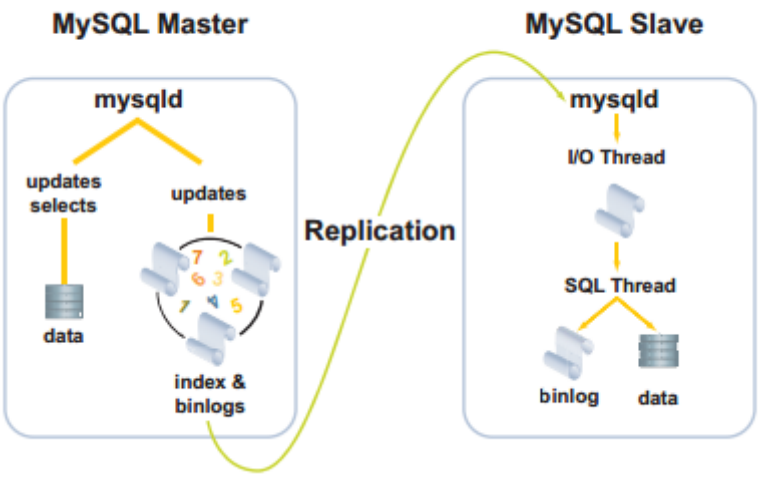
La principal diferencia entre una declaración y una hilera para propósitos de replicación es que una declaración es generada a través de un lenguaje de búsqueda o de query estructurado, mientras que en la replicación basada en hilera está basada en los datos no procesados en cada hilera.

Antes de la introducción del soporte MySQL 5.1 para la replicación basada en hilera, se debía tener cuidado al usar la replicación basada en la declaración para optimizar totalmente la declaración y evitar el desempeño de los cuellos de botella. Tales problemas de desempeño podían ser generados a través de una estructura de búsqueda ineficiente que solicitaba múltiples datos basados en pequeños resultados, con una búsqueda extendida compleja. En esa etapa, ese tipo de desempeño para regresar una búsqueda con dichas características era inevitable.

Replicación por Declaración versus Replicación por Línea

Iniciando en MySQL 5.1, los DBAs tienen la elección de utilizar la replicación basada en la declaración o basada en hileras, o una combinación de las dos. La replicación basada en la declaración es mejor para las aplicaciones que no hacen un uso pesado de las funciones no-determinantes o lo que el sistema llama tal como usuarios SELECTS(). La replicación MySQL basada en la declaración produce pequeños archivos de registro binario y un registro binario puede ser utilizado para auditar la base de datos. Debido al registro binario más eficiente, la replicación basada en declaración puede procesar más transacciones por segundo en muchos casos.

Con una replicación basada en hilera, todo (es decir, una hilera completa) puede ser replicado. Muy pocos candados son usados en muchas declaraciones DML tanto en los servidores maestros como en los servidores esclavos utilizando la replicación basada en hilera. Además, la replicación basada en hilera puede resultar en una aplicación más rápida de cambio de datos en los servidores esclavos, especialmente en objetos con llaves primarias.



Topologías de Replicación MySQL

Múltiples topologías reciben soporte de la replicación MySQL, como se ha visto en la Figura 3. La replicación de topología más simple es una sóla configuración maestro/esclavo. Las configuraciones con un maestro y múltiples esclavos también reciben soporte. Topologías más complejas incluyen configuraciones multi-maestras, con característica de dos o más servidores maestros. Mientras estas configuraciones están soportadas, se debe tener cuidado de asegurarse que los espacios de datos no están compartidos entre los servidores maestros. Si el maestro es situado incorrectamente en una configuración multi-maestra, puede ocurrir una sobre-escritura de los datos.

Una topología que no está soportada es una configuración multi-fuente, con múltiples maestros actualizando un sólo servidor esclavo.