

OpenMosix

Кластером называется совокупность вычислительных машин, объединенная для решения неких специфических задач программно – техническими средствами, связанными между собой.

Кластеры с защитой от сбоев состоят из двух или более компьютеров, соединённых сетью и отдельным синхронизирующим соединением, которое используется для мониторинга того, что все службы задействованы. В случае если одна из служб даёт сбой, другой узел перенимает её выполнение.

Для кластеров с балансировкой нагрузки концептуальным является то, что когда узел становится доступным, кластер проверяет, какой из узлов является наименее загруженным, и посылает запрос к этому узлу. Фактически, большую часть времени кластер с балансировкой нагрузки действует как кластер с защитой от сбоев, но с дополнительной балансирующей функциональностью, и, как правило, содержит меньшее число узлов.

Высокопроизводительные вычислительные кластеры, конфигурируются специально для того, чтобы обеспечивать наивысшую производительность. Этот тип кластеров также обладает некоторыми свойствами балансировки нагрузки, так как кластер пытается распределить процессы по узлам для увеличения производительности. И главным следствием из этого является то, что в случае когда процесс распараллеливается, то подпроцессы могут быть распределены по различным узлам, вместо того, чтобы выполняться один за другим.

Существует большое количество кластерных проектов. Наиболее известными являются **MOSIX** и её клоны, **Beowulf**, **X-Com**, **LVS**, **HeartBeat**. Все они имеют свои применения, преимущества и недостатки.

В данном случае будет рассмотрено применение системы **OpenMosix** для увеличения производительности сервера баз данных под управлением **Firebird**. Выбор на **OpenMosix** пал потому, что применение этой системы не требует модификации исходного кода приложения.

OpenMosix расшифровывается как **Open Multicomputer Operating System for UNIX**. Ядром **OpenMosix** являются адаптивные интерактивные алгоритмы, балансирующие нагрузку, управляющие памятью и файловым вводом – выводом.

Особенностью **OpenMosix** является то, что пользователь только запускает программу, а кластер решает, где её выполнить. После создания нового процесса, **OpenMosix** пытается назначить его лучшему доступному узлу на этот момент. **OpenMosix** контролирует все процессы, и, в случае необходимости, мигрирует процессы между узлами, чтобы максимизировать общую эффективность кластера.

Для пользователя этот механизм миграции скрыт. Он продолжает видеть (и контролировать) все свои процессы, как будто они выполняются на узле, с которого он их и запустил. Для того, чтобы посмотреть реальную картину существует утилита **OpenMosixView**, с помощью которой можно управлять кластером.

Системная модель образа **OpenMosix** основана на модели "домашнего узла", в которой кажется, что все пользовательские процессы выполняются на узле, с которого пользователь вошёл в систему. Каждый новый процесс создаётся на том же самом узле, что и его родительский процесс. Процессы, которые мигрировали, взаимодействуют с пользовательской средой окружения через домашний узел пользователя, но там, где возможно, используют локальные ресурсы. Пока нагрузка узла остаётся ниже порогового значения, все пользовательские процессы ограничены этим узлом. Однако когда нагрузка начинает превышать пороговое значение, некоторые процессы могут мигрировать на другие узлы кластера.

Для миграции процесс разбивается на две части: пользовательскую и системную. Пользовательская часть переносится на удалённый узел, в то время как системная остаётся на домашнем узле, уникальном для всего кластера. Все узлы кластера имеют уникальные идентификационные номера. Системную часть называют представительской (*deputy*), поскольку

она отвечает за разрешение большинства системных вызовов (syscalls). **OpenMosix** берёт на себя задачу обеспечения взаимодействия между этими двумя частями.

Основным отличием **OpenMosix** является доставка процесса к данным, а не наоборот, как это принято в большинстве других систем. Такой подход позволяет экономить системные ресурсы на процедурах ввода – вывода, поскольку "таскать" большой файл данных с узла на узел – занятие ресурсоёмкое. Отсюда вытекают требования к файловой системе кластера. Она должна обеспечивать непротиворечивость и согласованность своего состояния.

Для поддержки файловых операций применяется **oMFS**. Может применяться как с поддержкой файловой системы прямого доступа **DFSA**, так и без. С **DFSA** скорость процедур ввода – вывода значительно выше, что повышает общее быстродействие кластера.

DFSA работает поверх любых подходящих общедоступных всему кластеру файловых систем, которые соответствуют следующим требованиям:

- Файловая система смонтирована в той же точке монтирования всеми узлами;
- Идентификаторы пользователей / групп являются идентичными на всех узлах кластера.

Таким образом **oMFS** и **DFSA** делают все каталоги и обычные файлы доступными для всех узлов по всему кластеру, обеспечивая согласованность во время просмотра файлов с различных узлов, как если бы весь доступ к файлу был произведён с того же самого узла, на котором этот файл и находится.

Таким образом, кластер на **OpenMosix** являет собой систему, очень близко напоминающую многопроцессорный сервер. Всё было бы очень хорошо, если бы не "но". Всё дело в том, что **Firebird** производит доступ к файлу базы данных, как к разделяемой памяти (**SHM**). И это препятствует миграции процессов **Firebird** на другие узлы. В настоящее время выпущен патч **MigSHM**, который снимает эту проблему.

Теперь об установке.

Для установки применялся дистрибутив **ALTLinux Master-2.4**. В его комплекте есть всё необходимое, за исключением утилиты **OpenMosixView** и ядра с патчем **MigSHM**.

Последовательность:

1. Устанавливаем на узлах кластера пользователей, от имени которых будут запускаться процессы. Особо следим за **UID** и **GID**, которые должны быть идентичными на всех узлах.
2. Устанавливаем R-утилиты (**rsh** и пр.). В каталогах пользователей прописываем **.rhosts** и **/etc/hosts.equiv**. Вносим туда пользователей и узлы кластера для беспарольного доступа.
3. Проверяем, пускает ли **rsh nodeIP**. Если пускает уа все ноды, то можно идти дальше.
4. Устанавливаем ядро **apt-get install kernel-image-om-smp**
5. Устанавливаем **apt-get install openmosix-tools**
6. Создаём каталог **/mfs** на каждом узле
7. В **/etc/fstab** прописываем **mosix /mfs mfs dfsa=1 0 0**

На сём перегружаем узлы на новое ядро и радуемся. Если нужна миграция приложений типа **Firebird**, то устанавливаем ядра с патчем **MigSHM**. Дальнейшие тонкие настройки производятся с помощью **openmosix-tools** и **OpenMosixView**.