ОПЫТ РЕАЛИЗАЦИИ КЛАСТЕРА НА ОСРВ

В.В. Макаров
Институт проблем управления РАН им. В.А. Трапезникова
Россия, 117997, Москва, ул. Профсоюзная, 65
E-mail: makarov@ipu.ru

Демонстрируется целесообразность создания кластеров высокой доступности на базе ОСРВ. Совокупная стоимость владения такой системы на порядки ниже аналогичных систем высокой доступности с Windows- и Unix-систем. На базе гетерогенного QNX-кластера решалась тестовая задача, как на вычислительном кластере, по подбору криптографического ключа шифрования. В этом случае мы получили рост производительности системы (гетерогенного кластера) в сравнении с одиночным ПК, при этом стоимость владения у нас не изменилась (использовались обычные ПК из сети лаборатории).

EXPERIENCE OF REALIZATION CLUSTER ON OSRT. / V.V. Makarov (Institute of Control Sciences RAS, Profsoyuznaya str., 65, Moscow, 117997, Russia). The expediency of creation High Availability Clusters on the basis of Real Time Operating System is shown. TCO (Total Cost of Ownership) of such system on usages below similar High Availability Clusters with Windows- or Unix-systems. On the basis of heterogeneous QNX-cluster, as on High Performance Cluster, the test problem on search of a cryptographic key was accomplished. In this case we have received growth of system throughput (heterogeneous cluster) in comparison with the standalone personal computer, thus TCO has not changed (usual personal computers from a laboratory network were used).

Введение

Современные объекты управления предъявляют очень высокие требования к безотказной работе управляющего компьютера. Это медицинские компьютерные системы, управляющие аппаратурой жизнеобеспечения пациента, системы управления опасным производством и ряд других систем, которые объединяются понятием критические системы. Фактически все системы жесткого реального времени предполагают безотказную работу управляющего компьютера. Аналогичные требования к безотказной работе файл-сервера предъявляются в настоящее время в ИТ-системах: биллинговые, банковские системы, электронная коммерция, системы бронирования ж/д и авиабилетов, системы поиска и заказа литературы в библиотечных системах и т.п.

Одним из эффективных способов реализации высокой надежности управляющего компьютера является построение кластера, объединяющего ряд компьютеров в единый управляющий комплекс, видимый извне в сети, как один компьютер. Такого рода кластер называют кластером высокой готовности (High Availability – HA) [8].

Известный опыт построения кластеров для управления и решения сложных вычислительных задач, поддающихся распараллеливанию, говорит о том, что это однородные вычислители (в качестве узлов используют «машины-близнецы»), работающие либо на специализированной ОС, либо ОС общего назначения, как правило, одной из реализаций Unix-систем. Стоимость такой системы определяется количеством узлов, объединенных в кластер, обеспечивающей аппаратурой и ПО. Стоимость готовых кластерных решений порой очень высока – от нескольких десятков до нескольких сотен тысяч долларов, что исключает возможность построения по-настоящему эффективных кластерных решений с невысокой стоимостью.

В работе рассматривается опыт построения эффективной кластерной системы высокой готовности. Такая реализация оказалась возможной благодаря выбору в качестве базовой ОС операционной системы реального времени (ОСРВ) с уникальными на данный момент времени характеристиками. Сам выбор ОСРВ для кластера НА, кажется вполне закономерным. Однако проведенный библиографический поиск не выявил примеров

реализации такого рода кластеров. Кроме того, мы поставили цель реализовать файл-сервер на предельно дешевой аппаратуре: на обычных ПК, причем, на существенно различных по производительности. Последнее требование, кластер должен функционировать в сети с машинами под ОС Win dows, в частности, Windows XP (рис.1).

Рис. 1. Архитектура сети с кластером.

Операционная система НА-кластера – ОСРВ QNX/Neutrino

Выбор ОСРВ был сделан для QNX/Neutrino построенной по микромодульной архитектуре [1-3]. QNX – POSIX-совместимая операционная система реального времени, предназначенная преимущественно для встраиваемых систем. Считается одной из лучших реализаций концепции микроядерных операционных систем.

Микроядерная операционная система QNX (рис.2) позволяет улучшить общее быстродействие системы (небольшое микроядро может уместиться в кэше процессора). Основное достоинство микроядерной архитектуры — высокая степень модульности ядра операционной системы. Это существенно упрощает добавление в него новых компонентов. В микроядерной операционной системе можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д. Существенно упрощается процесс отладки компонентов ядра, так как новая версия драйвера может загружаться без перезапуска всей операционной системы [4].

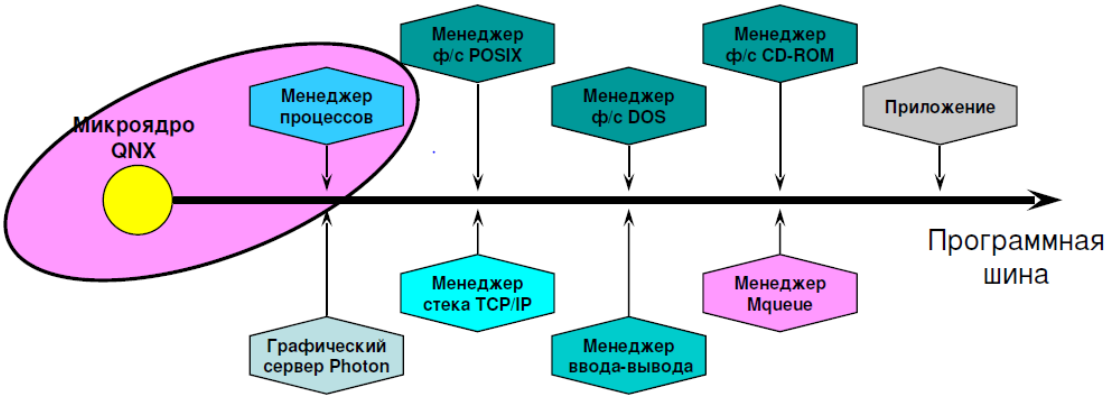

Рис. 2. Архитектура QNX

Компоненты ядра операционной системы ничем принципиально не отличаются от пользовательских программ, поэтому для их отладки можно применять обычные средства. Микроядерная архитектура повышает надежность системы, поскольку ошибка на уровне непривилегированной программы менее опасна, чем отказ на уровне режима ядра.

В то же время микроядерная архитектура операционной системы вносит дополнительные накладные расходы, связанные с передачей сообщений, что отрицательно влияет на производительность. Для того чтобы микроядерная операционная система по скорости не уступала операционным системам на базе монолитного ядра, требуется очень аккуратно проектировать разбиение системы на компоненты, стараясь минимизировать взаимодействие между ними.

Только процесс микроядра (менее 32 KB) выполняется в режиме супервизора. Все остальные (включая менеджер процессов, драйверы устройств, файловых систем и т. д.) – как ординарные процессы пользовательского уровня.

Благодаря этому драйверы устройств (и, что особенно важно, псевдоустройств, которые могут быть всем, чем угодно, например, оконной подсистемой со своим специфическим поведением) подгружаются и выгружаются динамически, гарантированно не нарушая работоспособности ядра (любой, кому доводилось перекомпилировать ядро Linux для добавления нового драйвера, оценит это по достоинству).

Менеджер процессов обеспечивает работу каждого из них в отдельном, полностью защищенном адресном пространстве. Все пользовательские приложения используют третье кольцо защиты процессора, драйверы – второе и (изредка, в этом практически нет необходимости) первое, и только микроядро безраздельно владеет нулевым. Это – одно из основных отличий QNX от других ОС данного класса. Все вызовы, для выполнения которых необходима работа в нулевом кольце, реализуются системными сообщениями, что обеспечивает дополнительную стабильность [10].

Вполне допустимо организовать такую конфигурацию, в которой вообще не будет менеджера процессов – одно микроядро и потоки, выполняющие конкретную задачу. Модульность подразумевает, что вы можете укомплектовать систему под определенное целевое применение – от наручных часов до высокопроизводительного кластера с несколькими процессорами на каждом узле. Кроме того, это означает возможность реализации максимально "щадящего" режима использования аппаратных ресурсов.

Любые параметры ОС (сетевые адреса, протоколы, файловые системы) конфигурируются "на лету" без необходимости перезагрузки. Полная остановка компьютера может понадобиться только в случае сборки нового ядра (точнее, перекомпоновки его образа, что не требует перекомпиляции) [5].

Система удовлетворяет требованиям "жесткого" реального времени.

QNX способна мирно сосуществовать на диске компьютера практически со всеми современными ОС. Один из предлагаемых способов установки - прямо "поверх" файловой системы FAT32, вся инсталляция занимает не более 10 минут.

QNX, как ОСРВ имеет ряд штатных механизмов повышения надежности, поэтому не требует специального ПО, повышающего надежность фунционирования ОС [10]:

- полная защита памяти процессов
- возможность динамической реконфигурации системы без перезагрузки
- программная поддержка сетевых кластеров Qnet
- специализированные инструменты диагностики и анализа (например, SAT, служащий для трассировки ядра)
- High Availability Toolkit – специализированный пакет, состоящий из администратора высокой доступности HAM, библиотека функций взаимодействий с HAM (HAM API), клиентская библиотека восстановления (Client Recovery Library) [8]
- технология Adaptive Partitioning для организации процессами гарантированного доступа к ресурсам [6].

QNX может работать на машинах разной архитектуры и производительности, что снижает затраты на организацию HA-кластера.

QNX позволяет обеспечить программную поддержку кластеров при помощи технологии QNET. QNET – сетевая архитектура на основе обмена сообщениями, предоставляет возможность прозрачного доступа к любому ресурсу, где бы он ни находился

[7].Отличительными особенностями QNET являются высокая надежность и производительность, динамическая балансировка нагрузки, расширяемость и прозрачность распределенного доступа.

Построение сети QNET по сути своей уникально: вместо передачи условных, определенных протоколом, данных, по сети передаются системные сообщения, которые, после “сборки” менеджером сети поступают непосредственно в ядро. Передача сообщения по сети происходит, используя механизм, в точности аналогичный тому, который используется и при выполнении запросов системного API (например, open()) на локальном компьютере. QNET обеспечивает обмен сообщениями между процессами на удаленных узлах прозрачно, как если бы они находились на одном узле сети: вся сеть превращается в единый компьютер, с общими ресурсами и задачами.

За счет такой архитектуры все, что работает на локальном процессоре, работает и на удаленном (рис.3). Несколько компьютеров, соединенных в QNX-сеть, могут быть задействованы для выполнения единого вычислительного процесса (в смысле "процесса операционной системы") [7]. Ни в одной ОС мы не встречали такой степени "сетевой прозрачности". В частности, таким образом обеспечивается доступ к любой системной информации на соседних узлах (вплоть до контекста каждого процесса), прямой доступ к удаленным устройствам (начиная с накопителей и заканчивая COM-портами, PCI-шиной и оперативной памятью).

Стандартный вариант QNX, что важно, стандартный по стоимости, позволяет строить кластерную систему (в отличии, например, от Windows 2000, для которой существует специальный кластерный вариант поставки – Server Data Center, и соответственно, своя стоимость лицензии).

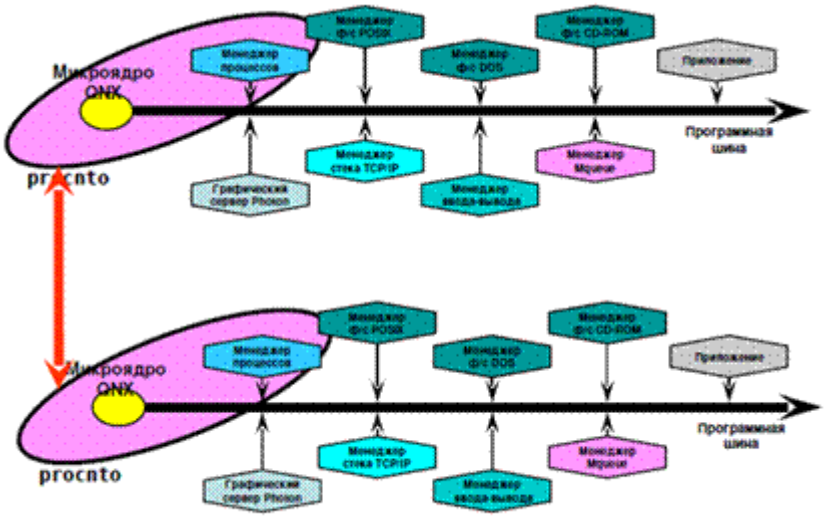

Рис. 3. Узлы обмениваются сообщениями микроядра.

Для построения сервера была выбрана классическая кластерная схема «активный-пассивный»: мастер-сервер и сервер-дублер (рис.4).

Эта схема применяется в тех случаях, когда необходимо использовать отказоустойчивый кластер для поддержки приложений, для кластерной архитектуры не предназначенных. При такой схеме выполнением задач занят только один из узлов кластера. На втором узле поддерживается полная копия данных первого и при сбое главного узла второй берет на себя его функции. Схема обеспечивает высокий уровень надежности.

На каждой машине установлено по две сетевые карты – один сетевой адаптер для сети QNET, другой – для сети TCP/IP. Для общения с внешней сетью использовалась программа/утилита **Samba**, которая позволяет обращаться к сетевым дискам на различных операционных системах по протоколу SMB/CIFS [9].

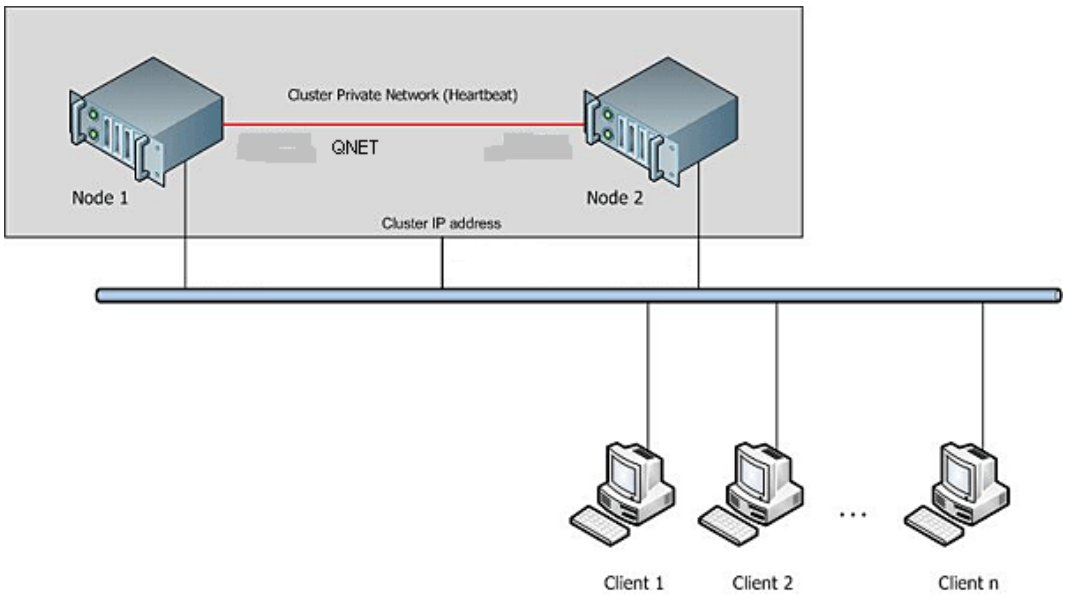

Рис. 4. Схема active-passive

Для зеркалирования использовалась программа **rsync**, выполняющая синхронизацию файлов и каталогов в двух местах с минимизированием трафика. Важным отличием **rsync** от многих других программ является то, что зеркалирование осуществляется одним потоком в каждом направлении, а не по одному или несколько потоков на каждый файл. Работает **rsync** в связке с клок-сервером **cron**. **cron** – демон-планировщик задач, использующийся для периодического выполнения заданий в заданное время. Задавая время синхронизации в специальном файле заданий – **crontab**, на сервере-дублере, мы можем иметь актуальную копию мастер-сервера. Для тестов мы задавали время синхронизации одну минуту. Симметрично, на мастер-сервере проводим аналогичные установки для зеркалирования на случай, если потребуется поменять серверы местами.

Для обеспечения гарантированной работы критических для функционирования системы процессов запускаем специальную программу, входящую в состав ОСРВ – администратор высокой доступности/надежности (High Availability Manager – HAM) [8]. HAM регистрирует зону ответственности и создает свою точную копию – Guardian. Если убить Guardian, то HAM сразу же создаст нового. Если убить HAM, то Guardian создаст свою копию и встанет на место HAM. То есть эти процессы не могут умереть. Корректно их завершить можно только специальной командой. Любой процесс для системы под управлением QNX может контролироваться с помощью HAM и оперативно подвергаться нужному воздействию в случае появления заданного состояния, что особенно важно для критических процессов, от нормальной работы которых зависит не только QNX, но и весь кластер в целом.

После того, как обеспечена стабильная работа нужных приложений на каждой из машин, необходимо приложение, которое будет обеспечивать контроль наличия «соседа» по QNET и требуемых для нормальной работы высоконадежного сервера приложений. В случае «смерти» мастер-сервера со стороны дублера должны быть произведены следующие действия: запуск у себя настроек мастер-сервера и информирование о «смерти» мастер-сервера. В случае «смерти» мастер-сервера со стороны сервера-дублера должны быть произведено информирование о «смерти» дублера (рис.5).

В штатном режиме на мастер-сервере работает суперсервер inetd, подключенный как объект HAM, и осуществляет контроль наличия дублера, а на сервере дублере работает клок-сервер cron со сконфигурированной для инкрементального копирования с мастер-сервера сетевых ресурсов таблицей crontab, и осуществляет контроль наличия мастер-сервера.

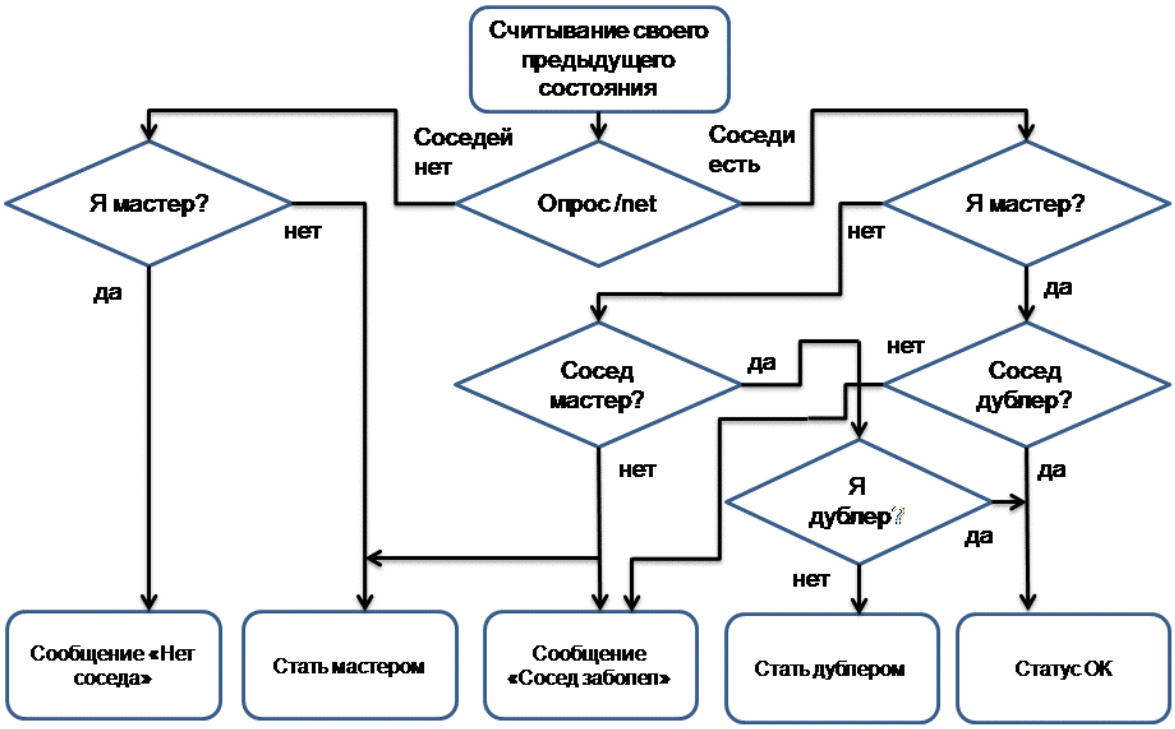

Рис. 5. Алгоритм работы приложения мониторинга.

Кроме того, не нужно забывать и о свойствах QNX как ОСРВ – обеспечивать требуемый уровень сервиса в заданное время, что даёт право говорить о большей эффективности работы через inetd, чем это было бы в операционной системе общего назначения, задача которой – распределить ресурсы между приложениями.

Тестирование компонентов кластера

Протестируем, как обеспечивается надежная работа критических для мастер-сервера приложений. Смотрим на консоли мастер-сервера, есть ли процесс inetd. inetd не запущен, поэтому, естественно, такого процесса нет. Далее идёт попытка запустить программу без предварительного запуска HAM – присоединить inetd к HAM не выходит, т.к. HAM не запущено, ошибка. После старта HAM программа запускается и отрабатывает так, как это и требовалось: HAM запускает inetd, соответствующий процесс отображается при получении списка процессов.

inetd останавливается командой slay. Если снова получить сведения о процессе inetd, то этот процесс будет виден на консоли, но уже с другим номером. Это означает, что HAM, увидев смерть inetd, перезапустил его. Далее HAM останавливается, также останавливается процесс inetd, снова идёт получение сведений о процессе inetd – процесс не отображается, что и должно быть. Таким образом обеспечена бесперебойная работа inetd. Используя штатные средства QNX, гарантирована непрерывная работа критического для функционирования системы процесса.

Аналогично, должна быть обеспечена и бесперебойная работа клок-сервера cron на сервере-дублёре. Таким образом обеспечивается работа нужных приложений.

Тестирование кластера

Рассмотрим пример тестирования системы. Задача состоит в запуске гетерогенного QNX кластера и проверки доступности файлов, хранящихся на кластере, из внешней Windows-сети в случае «выхода из строя» одной из машин кластера.

Запускаем QNX кластер на следующем оборудовании: IBM PC Intel Core 2 Duo + IBM PC AMD Semphron x86 (Рис.1, 4). На консолях машин убеждаемся в желаемом развитии запуска критических процессов. На машинах получаем «зеркальный» контент. Одну из папок мастер-сервера предоставляем в общее пользование для клиентского компьютера с Windows XP. Для этого потребуется создать на обоих серверах пользователя с логином и паролем, аналогичным паролю этого пользователя при входе в Windows на своей машине. Пользователю автоматически будет создан каталог с его именем в /home. Туда помещаются несколько файлов. Подключенная на Windows-хосте папка видна как сетевой диск X:\. Убеждаемся в доступности файлов этой папки из Windows-клиента. Далее имитируем выход из строя мастер-сервера. (можно просто его выключить, а можно остановить менеджер сети – для системы в целом разницы никакой, главное, что мастер-сервер больше недоступен). На консоли видим, что сервер-дублёр стал мастер-сервером, что и должно было произойти. Теперь нужно с машины пользователя посмотреть, что данными. Вход на диск X:\ из Windows-хоста. Вход на сетевой диск происходит нормально – следовательно, поставленная задача выполнена.

Выше был рассмотрен пример построения и тестирования высоконадёжного файл-сервера на базе QNX-кластера. Требуемые свойства системы были достигнуты за счёт использования штатных средств повышения надёжности и приложения собственной разработки, которое использовало в работе уникальное свойство QNET – прозрачный доступ ко всем ресурсам соседа.

Тестирование кластера на задаче с параллельным выполнеием

На самом деле, использование прозрачного доступа к ресурсам соседей по QNET – это только «верхушка айсберга». Используя механизм межпроцессного взаимодействия QNX с помощью обмена сообщениями микроядра или более понятной и высокоуровневой технологии написания менеджера ресурса, мы получаем возможность создания гораздо более глубокого взаимодействия между машинами, чем просто «заглядывание» друг другу в папки. Протестируем, возможно ли использование данного кластера для задач, требующих более высокой производительности, чем единичный (standalone) компьютер.

Возьмём довольно известный проект Олега Цирюлика «Симметричный кластер», в котором производится поиск ключа шифрования для сообщения, зашифрованного при помощи алгоритма XOR-свертки [5]. Основывается проект на «врождённых» кластерных свойствах QNX и идеологии использования менеджера ресурса.

Исходя из того, что QNX хосты, объединённые QNET сетью, сами по себе уже являются полноценным многомашинным кластером, появляется возможно организовать систему для применения в целом ряде кластерных задач.

Действительно:

- Каждый QNX-хост в сети обладает собственным микроядром OS;
- Микроядро QNX с одинаковой лёгкостью обменивается сообщениями уровня микроядра (по схеме Send – Receive - Reply) как с процессами на своём собственном хосте, так и с процессами на удалённых хостах.

Стоит «нагрузить» сообщения уровня микроядра целевой информацией для взаимодействия разнесённых частей распределённого приложения – кластер готов.

В предложенной к тестированию задаче поиска ключа показана возможная реализация кластерных вычислений в системе из нескольких универсальных компьютеров, работающих под управлением OS QNX и объединённых сетью QNET. Отдельные части

единой задачи разбросаем (например, по диапазонам начальных условий) между идентичными процессами на хостах.

Кроме того, представлена задача **single** – однопроцессорный вариант того, что предполагается далее разложить на узлы кластера: поиск приемлемых ключей дешифрования. Для поиска ключа декодирования используется простой линейный перебор всех возможных значений ключа заданной длины. Программа выполняется с двумя аргументами: имя исходного (дешифрируемого) файла и длина ключа (точно в том же формате будет запускаться и её многопроцессорный аналог).

Эта программа будет нужна для сравнения результатов работы (они даже имеют аналогичный по форме вывод) со своим многопроцессорным аналогом **master**. Но самое главное, почему эта программа просто необходима – это для сравнения временных характеристик однопроцессорного и многопроцессорного исполнения:

#time single d0-2.txt 2

#time master d0-2.txt 2

В связи с определёнными лимитами на публикацию не представляется возможным детально рассказать об однопроцессорном и многопроцессорном вариантах программы дешифрования, поэтому остаётся отослать заинтересованного читателя к публикации [5]

Запустим сначала на 2-байтовом ключе. Поиск ключа происходит быстрее при работе программы на одной машине, чем при работе на двух. Это связано с различными накладными расходами при передаче информации по сети и т.д. При увеличении же длины ключа до 3 байт видно увеличение быстроты кластерного перебора по сравнению с одиночным более чем в полтора раза! Если реализовать то же самое на кластере из большего числа машин, то и скорость увеличится.

Заключение

В работе была продемонстрирована возможность и целесообразность создания кластеров высокой доступности на базе ОСРВ QNX Neutrino. Имеющиеся механизмы ОСРВ для обеспечения надежности функционирования системы во многом превосходят возможности ОС общего назначения, таких например, как Linux и т.п. При этом стоимость владения – TCO (Total Cost of Ownership), может оказаться на порядки ниже аналогичных систем высокой доступности с Windows- и Unix-систем. Кроме того, на базе гетерогенного QNX-кластера решалась тестовая задача, как на вычислительном кластере (High Performance Cluster) по подбору криптографического ключа шифрования. В этом случае мы получили рост производительности системы (гетерогенного кластера) в сравнении с одиночным ПК при этом TCO не изменилось (использовались обычные ПК из сети лаборатории).

Список литературы

1. Зыль С.Н. Операционная система реального времени QNX Neutrino: от теории к практике / С.Н. Зыль – Изд. 2-е - СПб.: БХВ-Петербург, 2004. – 192 с.
2. Зыль С.Н. QNX Momentics. Основы применения / С.Н. Зыль - СПб.: БХВ-Петербург, 2004. – 256 с.
3. Операционная система реального времени QNX Neutrino 6.3. Руководство пользователя: Пер. с англ. – СПб.: БХВ-Петербург, 2009. – 480 с.:ил.
4. Операционная система реального времени QNX Neutrino 6.3. Системная архитектура: Пер. с англ. – СПб.: БХВ-Петербург, 2006. – 336 с.:ил.
5. Практика работы с QNX / [Д. Алексеев и др] – М.: Издательский дом «Комбук», 2004. – 432 с.: ил.
6. QNX Neutrino RTOS. Adaptive Partitioning User's Guide [Electronic resource] // QNX Software Systems International Corporation. – Electronic Data. – Kanata, Ontario, cop. 2009. – 109 p. – Mode access: http://www.qnx.com/download/download/19496/adaptive_partitioning.pdf

7. QNX Neutrino RTOS. Core Networking with io-pkt User's Guide [Electronic resource] // QNX Software Systems International Corporation. – Electronic Data. – Kanata, Ontario, cop. 2009. – 105 p. – Mode access: http://www.qnx.com/download/download/19512/core_networking.pdf
8. QNX Neutrino RTOS. High Availability Framework User's Guide [Electronic resource] // QNX Software Systems International Corporation. – Electronic Data. – Kanata, Ontario, cop. 2009. – 206 p. – Mode access: http://www.qnx.com/download/download/19517/high_availability.pdf
9. QNX Neutrino RTOS. QNX Neutrino Utilities Reference [Electronic resource] // QNX Software Systems International Corporation. – Electronic Data. – Kanata, Ontario, cop. 2009. – 2092 p. – Mode access: http://www.qnx.com/download/download/19526/utilities_reference.pdf
10. QNX Neutrino RTOS. QNX Neutrino System Architecture [Electronic resource] // QNX Software Systems International Corporation. – Electronic Data. – Kanata, Ontario, cop. 2009. – 334 p. – Mode access: http://www.qnx.com/download/download/19524/sys_arch.pdf