Штатные механизмы QNX Neutrino для обеспечения отказоустойчивости вычислительных систем жёсткого реального времени

Сергей Зыль

Надёжность – одна из важнейших характеристик промышленных систем управления. В статье представлен обзор базовых механизмов ОС РВ QNX Neutrino, позволяющих системным интеграторам и разработчикам АСУ ТП создавать распределённые вычислительные комплексы с заданным уровнем отказоустойчивости.

Операционная система жёсткого реального времени QNX Neutrino (известная также как QNX 6) изначально предназначена для управления критичными системами, то есть такими системами, сбой которых может привести к крупному материальному ущербу или даже к человеческим жертвам. Столь ответственная область применения требует не просто высокого качества программирования, но и принятия мер «глубоко эшелонированного» обеспечения надёжности автоматизированных систем (АС). Для поддержки таких мер в QNX Neutrino наряду с микроядерной архитектурой с полной изоляцией модулей в ОЗУ предусмотрен ряд механизмов, основными из которых являются:

- механизм адаптивного квотирования ресурсов ЦПУ и ОЗУ,
- механизм обеспечения «горячей» замены сервисов,
- технология быстрой активизации устройств,
- механизм формирования распределённой вычислительной среды,
- механизм поддержки резервирования физических каналов связи в кластере,
- механизм обеспечения балансировки нагрузки на сервисы,
- технология автоматического восстановления процессов,
- технология автоматизации восстановления логических соединений.

В этой статье мы кратко охарактеризуем каждый из перечисленных механизмов QNX Neutrino.

Микроядерная архитектура с полной изоляцией модулей в ОЗУ

Операционная система QNX Neutrino имеет микроядерную архитектуру. Это значит, что в изолированных адресных пространствах с полной изоляцией от адресных пространств других процессов выполняются не только прикладные процессы, но и большинство системных сервисов, которые в традиционных операционных системах предоставляются непосредственно ядром. Микроядро является всего лишь коммутирующим элементом, позволяющим системным и прикладным процессам общаться друг с другом. Программы, реализующие системные сервисы, с точки зрения микроядра являются обычными прикладными процессами со всеми вытекающими последствиями. Любое добавление или удаление программных компонентов (драйверы устройств, файловые системы, сетевые стеки и т.п.) никоим образом не влияет ни на размер ядра, ни на его контрольную сумму, ни на его работоспособность.

Такая архитектура позволяет разработчику АС самому решить, какие системные сервисы нужны для решения его прикладной задачи, и создать свою собственную конфигурацию операционной системы весьма небольшого размера. По сути дела, разработчик АС может написать и сертифицировать собственный драйвер или системный сервис и включить его в систему без помощи и ведома разработчика ОС, не ухудшив при этом характеристик жёсткого реального времени ОС.

Микроядро QNX Neutrino отвечает за реализацию всех механизмов поддержки жёсткого реального времени ОС РВ QNX Neutrino:

- фиксированные приоритеты потоков (256 уровней) и ISR (пожалуйста, не путайте приоритеты с nice-числами, используемыми в ОС общего назначения, например Linux);
- мгновенное вытеснение задачи с меньшим приоритетом;
- вытесняемые системные вызовы (!) и даже ISR (!!);
- защита от инверсии приоритетов на базе протокола наследования приоритетов (Priority Inheritance Protocol);
- исключение непредвиденных расходов ресурсов (например, на свопинг);
- механизм трассировки ядра, позволяющий узнать точное время каждой операции.

Также микроядро выполняет некоторые другие функции, например, автоматически распределяет задачи по всем доступным ЦПУ или процессорным ядрам в режиме SMP (симметричный многопроцессорный), AMP (асимметричный многопроцессорный) или BMP (можно перевести bound multiprocessing как «исключительная многопроцессорность»). Важно отметить, что механизмы микроядра никоим образом не могут быть нарушены процессами.

Механизм адаптивного квотирования ресурсов ЦПУ и ОЗУ

Механизм квотирования ресурсов был реализован в QNX Neutrino во исполнение требований спецификаций ARINC к авиационным бортовым вычислительным системам. Примечательно, что предложенный в QNX Neutrino адаптивный подход к квотированию ресурсов был удостоен премии Embedded Award 2007.

Необходимость квотирования ресурсов диктуется возможными логическими ошибками при разработке или настройке программного обеспечения. Например, если высокоприоритетный поток выполняет холостой бесконечный цикл, то низкоприоритетные потоки никогда не смогут получить доступ к ресурсам ЭВМ, ведь холостой поток будет иметь право на максимальный доступ к этим ресурсам. Вторая проблема заключается в том, что в современных системах на ЭВМ могут выполняться сотни и даже тысячи конкурирующих за ЦПУ потоков, поэтому чрезвычайно сложно корректно настроить подобную систему, даже располагая такими мощными инструментами, как пакет QNX System Analysis Toolkit и Eclipse-модуль QNX System Profiler.

Суть механизма адаптивного квотирования иллюстрирует рис. 1. В некоторой вычислительной системе выполняется шесть потоков: A, B, C, D, E и F. Каждый из потоков имеет своё значение приоритета: максимальный приоритет у потока A, минимальный – у потока F. Для квотирования ресурсов процессора созданы три логических раздела по 30%, 30% и 40% соответственно от всего процессорного времени. Поток A запускается в разделе 1, потоки B, C и D – в разделе 2, остальные потоки – в разделе 3. На рисунке показано, что потоки ведут себя стандартным для системы реального времени образом, но только в пределах своих разделов. Если же какие-то потоки не полностью используют свой раздел (в нашем примере – раздел 3), то «лишнее» процессорное время распределяется между готовыми к исполнению потоками всех разделов в соответствии с их приоритетами. Кстати, именно поэтому механизм квотирования называется адаптивным.

Что же даёт механизм адаптивного квотирования ресурсов ЭВМ:

- повышение защищённости и коэффициента готовности АС за счёт невозможности монополизации ресурсов какой-либо программой (при DoS-атаках и некорректном коде);
- сокращение трудозатрат на сопровождение АС на 25-30% (по оценкам экспертов компании QNX Software Systems) за счёт локализации аномального поведения.

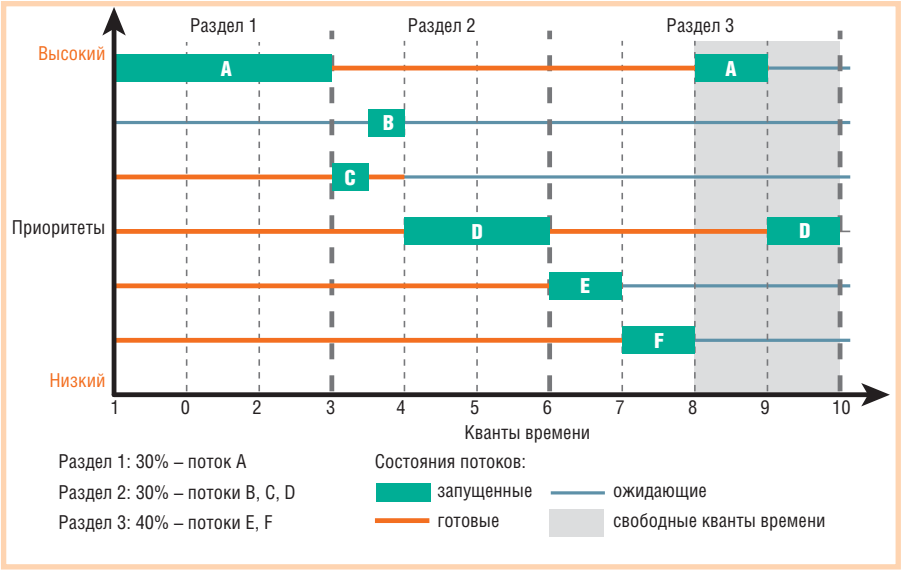

Рис. 1. Механизм адаптивного квотирования позволяет защититься от несанкционированного перерасхода вычислительных ресурсов при логических ошибках в программе и DoS-атаках

Следует сказать, что защита от монополизации ресурсов позволяет запускать в критичной системе даже потенциально ненадёжные программы: за счёт полной защиты памяти они не смогут повредить ядру и ответственным сервисам, а за счёт квотирования они, даже имея высокий приоритет, не смогут воспрепятствовать выполнению важных задач.

Механизм обеспечения «горячей» замены сервисов

Этот механизм основан на двух особенностях QNX Neutrino: микроядерной архитектуре и возможности перегрузки префиксов менеджера процессов.

Обычно встречается два варианта использования механизма «горячей» замены.

Первый вариант – запуск «теневого» сервера – используют для повышения времени наработки на отказ за счёт дублирования приложений, обеспечивающих критичные сервисы. В этом случае при запуске приложения-сервера сразу же запускается «теневой» сервер – его двойник, регистрирующий такое же, как у основного приложения, путевое имя (префикс). Разумеется, можно запускать несколько «теневых» серверов, причём для каждого сервиса, для которого это необходимо. В случае сбоя основного сервера новые запросы будут автоматически поступать на «теневой» сервер.

«Теневой» сервер может обрабатывать даже те запросы, которые поступили на основной сервер до сбоя. Для этого информация о таких запросах и о состоянии их обработки может храниться сервером, например, в регионе разделяемой памяти, доступном его «теневым» серверам. Такой подход реализован в мониторе ключевых процессов для самовосстановления (подробнее о мониторе см. далее в разделе «Технология автоматического восстановления процессов»), доступном в исходных текстах.

Второй вариант – обновление сервера – используют, когда необходимо обновить программу-сервер без перерыва в обработке клиентских запросов. В этом случае при запуске новой версии приложения-сервера он регистрирует такое же, как у старого приложения, путевое имя, но делает это так, чтобы последующие запросы клиентов направлялись в первую очередь к новому серверу. Другими словами, после запуска новой версии сервера все новые запросы автоматически начнут поступать к нему, а старый сервер может закончить обработку старых запросов и прекратить работу.

Технология быстрой активизации устройств

Эта технология предназначена для повышения коэффициента готовности АС путём сокращения времени восстановления работоспособного состояния. Она используется во время начальной загрузки QNX Neutrino на процессорной плате, и её часто называют «технологией мини-драйверов». Технология быстрой активизации устройств позволяет ещё до загрузки и в процессе загрузки ОС обес-

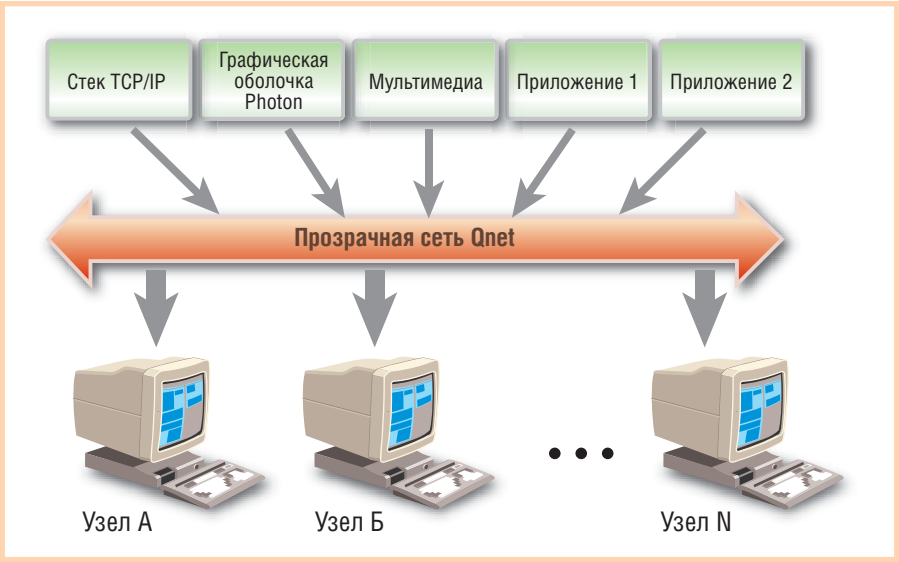

Рис. 2. Qnet (протокол 4-го уровня модели ISO OSI) объединяет ресурсы локальной сети в единую виртуальную ЭВМ

печивать отклик на внешние события, например команды системы управления электропитанием или сигналы, передаваемые по шине CAN.

Конечно, до того как загрузилась ОС, у программиста нет тех богатых функциональных возможностей и механизмов, которые эта ОС предоставляет. Тем не менее, некоторые операции, критичные ко времени отклика, можно осуществлять и даже получать при этом доступ к оборудованию. Речь идёт о возможности приступить к обработке критичных сигналов через десятки миллисекунд после подачи питания на ЦПУ.

Важная особенность технологии заключается в том, что после окончания загрузки ОС обработку запросов может продолжить полноценный драйвер без задержек времени или потерь данных.

Механизм формирования распределённой вычислительной среды

Это традиционный и чрезвычайно эффективный механизм ОС РВ семейства QNX. В QNX Neutrino он основан на протоколе четвёртого уровня модели ISO OSI, получившем название Qnet.

Уникальность этого механизма заключается в том, что Qnet, по сути дела, связывает в единую сетевую инфраструктуру непосредственно ядра копий QNX Neutrino, выполняющихся на разных ЭВМ сети. Это означает, что с точки зрения приложений и системных сервисов АС выполнение происходит на одной ЭВМ, как это схематично показано на рис. 2. То есть при старте Qnet программные компоненты АС автоматически становятся сетевыми приложениями без добавления какого-либо специального сетевого программного кода. ЭВМ под управлением QNX Neutrino, объединённые Qnet, фактически представляют собой псевдоединую виртуальную суперЭВМ. Почему «псевдо»? Потому что ОЗУ каждого из узлов сети Qnet сохраняет независимость, то есть оперативная память узлов является некогерентной.

Сетевые возможности обычных (несетевых) приложений в сети Qnet хорошо иллюстрирует такой пример: мы можем, работая за терминалом узла А, запустить на ЦПУ узла Б программу, которая хранится в файловой системе узла В, и при этом указать данной программе, чтобы она в качестве своей «родной» файловой системы использовала файловую систему узла Г, исходную информацию считывала из устройства ввода-вывода на узле Д, а в качестве управляющей консоли использовала одну из виртуальных консолей узла Е.

Кстати, применяемая в QNX Neutrino графическая оболочка Photon microGUI – это набор взаимодействующих программ, предоставляющих различные сервисы (управление видеоадаптером, управление устройствами ввода, управление сервером шрифтов, управление рабочим столом и т.д.). При использовании Qnet можно создавать весьма интересные сетевые конфигурации графических средств.

И напоследок следует сказать, что существует несколько способов защиты информации в сети Qnet. К ним относятся отображения идентификаторов пользователей, защита сетевых сообщений от несанкционированной модификации, использование различных методов взаимной идентификации узлов и т.д.

Механизм поддержки резервирования физических каналов связи в кластере

Надёжность функционирования Qnet обеспечивается с помощью резервирования физических линий связи. Для *каждого* логического соединения процесса-клиента с процессом-сервером может быть определён один из трёх вариантов выбора физического канала [1]:

1. **С автоматической балансировкой нагрузки (loadbalance)**. В этом случае Qnet самостоятельно принимает решение, по какой из доступных физических линий передавать пакеты. Для выбора адаптера Qnet анализирует время отклика удалённого узла на запросы, посланные по разным физическим линиям, и использует ту линию, которая является наиболее быстрой в данный момент времени. Этот способ используется Qnet по умолчанию.
2. **С предпочтением (preferred)**. В этом случае задаётся, какой из сетевых адаптеров предпочтителен для передачи информации. Если предпочтительный адаптер недоступен, то Qnet будет использовать остальные адаптеры, автоматически балансируя нагрузку между ними.
3. **Эксклюзивный (exclusive)**. Позволяет задавать передачу данных строго через определённый адаптер. Если адаптер недоступен, Qnet не будет передавать данные вообще, даже если с удалённым узлом можно связаться через другие адаптеры.

Вариант выбора физической линии для того или иного логического соединения можно задавать на разных этапах жизненного цикла АС: при разработке, при установке, при эксплуатации. Изменение варианта не требует перекомпиляции программ, что особенно важно для сертифицированного программного обеспечения.

Механизм обеспечения балансировки нагрузки на сервисы

Механизмы Qnet позволяют элегантно решать проблему нехватки вычислительных ресурсов в случаях, когда в процессе эксплуатации АС реальная нагрузка превышает пределы, предусмотренные при проектировании АС.

В таких случаях в сетевую АС, объединённую Qnet, могут быть добавлены дополнительные ЭВМ, на которые пере-

несено выполнение тех или иных вычислительных задач. Для обеспечения такой возможности QNX Neutrino поддерживает механизм, который позволяет распределять запросы клиентов между процессами-серверами, выполняющимися на разных узлах сети Qnet.

Суть механизма заключается в том, что при получении от клиента запроса на установление соединения сервер может вместо возврата клиенту идентификатора соединения перенаправить клиента к другому серверу, имеющему другое путевое имя.

Решение о перенаправлении может задаваться разработчиком АС на основе различных исходных данных, например с учётом загрузки физических каналов связи или текущей загрузки серверов, реально обрабатывающих запросы клиентов. Кроме того, этот механизм может быть использован в интересах информационной безопасности.

Технология автоматического восстановления процессов

Автоматическое восстановление процессов позволяет повышать коэффициент готовности АС путём сокращения времени восстановления работоспособного состояния.

В основе этой технологии лежит уже упоминавшийся монитор ключевых процессов. Основное назначение монитора – максимально быстрое определение факта сбоя серверного приложения и принятие мер для восстановления нормальной работы сервиса.

Чтобы выполнять свои задачи, монитор для начала «клонирует» самого себя, то есть порождает дублирующий процесс, которому, используя механизм разделяемой памяти, предоставляет полную информацию о мониторинге АС [2]. Монитор и его дублёр строго следят друг за другом, и при сбое одного из них второй процесс немедленно порождает новый процесс-дублёр. В качестве объекта мониторинга могут быть заданы любые (серверные или клиентские) процессы, выполняющиеся как на той же ЭВМ, что и монитор, так и на других узлах Qnet. Для каждого из объектов мониторинга указывается перечень событий, на которые следует реагировать монитору. Например, событиями являются завершение процесса-объекта или его перезапуск. И наконец, для каждого из событий определяется реакция – действие или перечень действий.

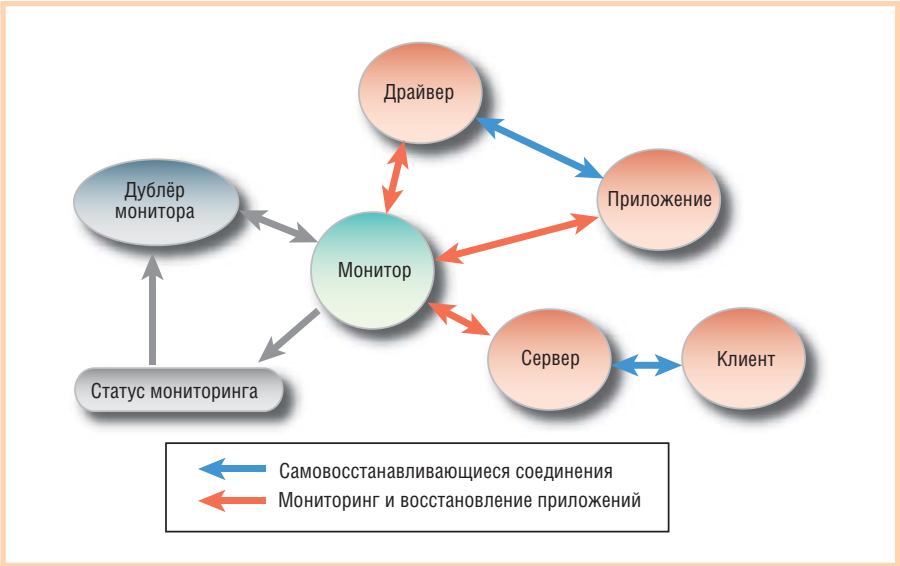

Рис. 3. Схема использования технологии автоматического восстановления процессов и соединений

Технология автоматического восстановления процессов позволяет создавать сценарии многоступенчатого восстановления достаточно сложных распределённых АС. На одной ЭВМ может быть запущен один тандем монитора ключевых процессов и его дублёра. Мониторы могут быть запущены на любом количестве узлов Qnet. И при этом каждый из мониторов сети Qnet может контролировать объекты, работающие на любом из доступных узлов Qnet.

Технология автоматизации восстановления логических соединений

Эта технология предназначена для сохранения логических соединений между клиентами и серверами в случаях временных отказов физических линий связи.

Суть решаемой проблемы заключается в следующем. При отсутствии физического соединения системные средства поддержки сетевых протоколов пытаются в течение определённого времени (тайм-аута) «достучаться» до удалённой ЭВМ. Если по истечении тайм-аута соединение не восстановилось, функция в приложении, инициировавшая передачу данных, завершится с ошибкой, имеющей определённый код. Следовательно, для обеспечения восстановления работоспособности приложения при сбоях физических линий связи программист должен предусмотреть в клиентской программе обработку кодов ошибок, возникающих при разрывах соединений, и принять меры к восстановлению соединений. Это существенно усложняет программирование.

Для того чтобы разделить логику выполнения прикладной задачи и логику обработки отказов линий связи, а также для автоматизации идентификации отказа в приложении существует технология восстановления логических соединений, основанная на использовании библиотеки восстановления клиента (Client Recovery Library). Обычно её сочетают с технологией автоматического восстановления процессов, как это показано на рис. 3.

Заключение

Думаю, что сказанного достаточно для иллюстрации уникальных возможностей микроядерной архитектуры QNX Neutrino, которые в сочетании с различными штатными средствами позволяют создавать АС жёсткого реального времени высокой надёжности. Следует помнить, что справочная служба QNX Neutrino содержит беспрецедентно обширную документацию, и данное обстоятельство в сочетании с доступностью исходных текстов этой популярной ОС РВ открывает перед разработчиками чрезвычайно богатые возможности. ●

Литература

1. Операционная система реального времени QNX Neutrino 6.3. Руководство пользователя: пер. с англ. – СПб. : БХВ-Петербург, 2009. – 480 с.: ил.
2. Зыль С.Н. QNX Momentics. Основы применения. – СПб.: БХВ-Петербург, 2004. – 149 с.: ил.

Автор – сотрудник
ООО «СВД Встраиваемые системы»
E-mail: s.zyl@kpda.ru