Нижегородский государственный университет им. Н.И.Лобачевского

Межфакультетская магистратура по системному и прикладному программированию
для многоядерных компьютерных систем

Учебный курс "Технологии построения и использования кластерных систем"

Раздел "Обзор систем управления кластерами"

Разработчики: К.В. Корняков, А.В. Шишков

Нижний Новгород
2007

Содержание

Обзор систем управления кластерами

1. Введение

Во всю историю вычислительной техники не было момента, чтобы уровня развития вычислительной техники было достаточно для решения всех стоящих перед человечеством задач. Постоянно ставятся новые, все более сложные задачи, требующие все более мощных вычислительных ресурсов для своего решения. И современные технологии создания вычислительной техники подошли к рубежу, когда дальнейшее наращивание скорости работы индивидуальных устройств становится практически невозможным. В связи с этим развитие вычислительной техники пошло по экстенсивному пути, основанному на дублировании вычислительных устройств, которые в параллели могут работать над общей задачей. Вместе с этим родилось параллельное программирование, призванное дать возможность эффективно использовать параллельные архитектуры. И сегодня разработчики программных систем используют параллелизм на всех уровнях, начиная от нескольких конвейеров суперскалярных процессоров, и заканчивая параллельно работающими вычислительными узлами в GRID.

Отдельный класс параллельных архитектур представляют кластерные системы. Кластер – это совокупность вычислительных узлов, объединенных сетью. Параллельное приложение для кластерной системы представляет собой несколько процессов, которые общаются друг с другом по сети. Таким образом, если пользователь сумеет эффективно распределить свою задачу между несколькими процессорами на узлах кластера, то он может получить выигрыш в скорости работы, пропорциональный числу процессоров.

Как правило, кластерные системы крайне интенсивно используются для проведения вычислений. Предприятия и организации чаще всего приобретают кластеры для решения потока задач. И зачастую потребности желающих воспользоваться вычислительными ресурсами превосходят доступный объем ресурсов, поэтому к кластерам можно наблюдать очереди. Ситуация очень похожа на ту, что существовала с мэйнфреймами на заре компьютерной эпохи. В то время для эффективного управления потоками задач создавались так называемые системы пакетной обработки. Пользователи помещали свои задачи в очередь этих систем, а за результатом приходили через нескольких часов, а иногда и дней.

Примерно то же самое происходит сейчас на кластерах, поэтому правильное распределение нагрузки по вычислительным узлам кластера имеет очень большое значение. Этот вопрос приобретает еще большую важность в случае, если кластер имеет неоднородную структуру: различается мощность центральных процессоров, объем оперативной памяти, скорость участков локальной сети. Если не учитывать особенности аппаратуры, то можно наблюдать, как параллельное приложение простаивает, дожидаясь процесса, который был распределен на самый медленный вычислительный узел.

Помимо эффективного планирования запуска задач на кластере, необходимо также автоматизировать процессы приема пользовательских задач, постановки их в очередь, запуска и сбора результатов. Важно обеспечить безопасность использования кластера, его отказоустойчивость, сделав при этом работу с кластером максимально простой, избавляя пользователей от лишних технических подробностей. Все эти факторы приводят к необходимости создания специализированных систем управления кластерами, основная цель которых – предоставить удобные средства эффективного использования кластера.

2. Общая характеристика систем управления кластерами

2.1. Кластерный подход к созданию суперкомпьютеров

Кластеры стали применяться в сфере выскопроизводительных вычислений сравнительно недавно. До конца 80-х практически все суперкомпьютеры представляли собой большой массив соединенных между собой процессоров. Подобные разработки чаще всего были уникальными и имели огромную стоимость не только приобретения, но и поддержки. Поэтому в 90-х годах все более широкое распространение стали получать кластерные системы, которые в качестве основы используют недорогие однотипные вычислительные узлы.

Основными достоинствами кластерного подхода являются именно дешевизна и легкая расширяемость. Цены на системы кластерного типа стремительно падают, а некоторые модели уже сейчас доступны для одиночных исследователей. К недостаткам же можно отнести сложность создания и отладки эффективных параллельных программ для систем с разделенной памятью. Однако в последнее время развиваются инструменты, облегчающие написание параллельных программ, что способствует все большему распространению кластеров.

Как уже было сказано выше, кластеры представляют собой системы с разделенной памятью, поэтому типичное параллельное приложение представляет собой совокупность нескольких процессов, исполняемых

на разных вычислительных узлах и взаимодействующих по сети. В принципе, разработчик может полностью взять на себя программирование распределенного приложения и самостоятельно реализовать общение по сети на основе сокетов, например. Однако в настоящее время существует довольно большое число технологий, упрощающих создание параллельных приложений для кластеров: MPI [18], PVM [19], HPF [20] и другие. Эти технологии существуют уже достаточно продолжительное время, за которое они доказали свою состоятельность и легли в основу огромного числа параллельных приложений.

Кластеры стали фактическим стандартом в области высокопроизводительных вычислений, и можно с большой долей уверенности сказать, что этот подход будет актуален всегда: сколь бы совершенен не был один компьютер, кластер из узлов такого типа справится с любой задачей гораздо быстрее.

2.2. Назначение систем управления кластерами

Во введении мы уже затронули эту тему. Суперкомпьютеры (и кластеры как подкласс) приобретаются для решения вычислительно трудоемких задач: моделирование климата, финансовые расчеты, генная инженерия, моделирование физических процессов, химических реакций и так далее. Сложность экспериментов такова, что могут потребоваться годы расчетов на одном процессоре, поэтому исследователи активно применяют распараллеливание. Как результат, колоссальные по своей сложности задачи могут быть решены «всего» за недели.

Однако при массовом запуске параллельных приложений на кластерах возникают новые сложности, которые связаны с тем, что чаще всего приходится иметь дело с большой серией экспериментов, каждый из которых имеет уникальные потребности относительно аппаратных ресурсов. В такой ситуации нужно очень умело распоряжаться узлами кластера, стараясь распределить нагрузку максимально равномерно. Необходимо находить «зазоры» в расписании между тяжелыми задачами и стараться запускать между ними более легкие, которые должны решаться на оставшихся свободных (или просто наименее загруженных) узлах. Нельзя забывать также, что задачи могут иметь различный приоритет и между запусками приложений могут существовать зависимости, что накладывает определенные ограничения на расписание.

Все вопросы, связанные с составлением расписания запусков пользовательских приложений, ложатся на систему управления кластером. Помимо составления расписания система должна обеспечивать надежность функционирования кластера. Выход одного узла из строя не должен приводить к остановке работы всего кластера. В идеале система должна «на лету» обнаруживать и обрабатывать изменение состава вычислительных узлов. Кроме составления расписания есть еще ряд задач, которые ложатся на систему управления: это предоставление удаленного доступа к вычислительным ресурсам, предоставление средств администрирования и отслеживания состояния кластера и ряд других вопросов.

2.3. Основные требования к системам управления

Главной задачей системы управления кластерами является обеспечение максимально быстрого возврата результатов запуска при существующем аппаратном обеспечении и согласно установленным правилам использования вычислительных ресурсов. Можно даже выдвинуть следующий критерий: хорошая система управления – это такая система, которую пользователь не замечает, имея возможность полностью сконцентрироваться на решаемой прикладной проблеме. Система должна максимально упрощать работу с вычислительным кластером и ускорять получение результатов экспериментов. Таким образом, с нашей точки зрения, можно выделить следующие три основополагающие требования к системам управления кластерами.

- **Производительность (высокая пропускная способность).** Системы управления кластерами фактически представляют собой системы пакетной обработки, и поэтому их основная задача – обеспечить максимальное количество заданий, выполняемых кластером за единицу времени. Чем быстрее получается результат, чем быстрее эта обратная связь для исследователя, тем большее число экспериментов может быть проведено, что позволит глубже проникнуть в исследуемое явление или, например, разработать для коммерческой организации более выигрышную стратегию поведения на рынке.

  Для обеспечения высокой производительности в системах управления используется эффективное априорное планирование и динамическое распределение нагрузки, основанное на миграции процессов.

- **Надежность (отказоустойчивость).** Кластер должен функционировать непрерывно. Все аппаратные и программные сбои должны оперативно обрабатываться, после чего кластер должен продолжить выполнение заданий. Если сбой привел к остановке вычислений, то система должна уметь восстановить их некоторое промежуточное состояние и начать с него, а не с самого начала. Так, если эксперимент продолжался несколько недель, то остановка его при 90%-й готовности может оказаться катастрофой. Поэтому в развитых системах управления периодически создаются контрольные точки исполняющихся процессов, чтобы их при необходимости можно было перезапустить.

- **Удобство использования.** Это достаточно размытое по смыслу требование является, однако, одним из наиболее важных. Система управления кластером должна быть гибкой и хорошо настраиваемой, а также простой в использовании.

  В частности, система должна иметь несколько интерфейсов доступа, для того чтобы пользователь мог выбрать наиболее подходящий для его потребностей. Так, многие системы имеют одновременно GUI-интерфейс, интерфейс командной строки и интерфейс прикладного программирования (API). Администратор же должен иметь широкие возможности выбора алгоритмов и политик планирования, чтобы адаптировать систему к конкретным характеристикам кластера и классам решаемых задач.

2.4. Альтернативный подход к управлению кластерами

Системы управления кластерами решают задачу эффективного управления ресурсами кластера, но это можно делать различными способами, и в настоящее время используются два основных подхода:

- Создание распределенных окружений для кластерных вычислений (Distributed Cluster Computing Environments, DCCE);
- Создание систем управления кластерами (Cluster Management Systems, CMS), или систем управления ресурсами и планирования (Resource Management and Scheduling, RMS).

Идея первого подхода состоит в создании «кластерной» операционной системы, которая бы взяла на себя все вопросы, связанные с эффективным управлением кластером. Так, приложение не обязательно исполняется на том узле, откуда был произведен запуск, а может мигрировать на менее загруженный узел. Также на всем кластере используется общая файловая система, и пользователь может даже не задумываться над тем, на каком именно узле кластера находятся данные. Существует несколько реализаций подобных окружений: OpenMosix [5], Kerrighed [6], OpenSSI [7], Gluster [8] и другие. Эти системы весьма удобны для пользователей, потому как работа в них фактически не отличается от работы в обычной операционной системе. Недостаток подхода состоит в том, что он практически нереализуем без коррекции операционной системы. Именно поэтому все подобные окружения существуют только для UNIX-подобных систем с открытым кодом (Linux, FreeBSD).

Существует альтернативный способ, не связанный с модификацией операционной системы, – создание специализированных систем управления кластером, которые мы рассматриваем в настоящей работе. Как правило, используется следующий порядок работы: пользователь передает системе свое приложение, указывая параметры запуска, а система уже сама определяет в какой момент и на каких узлах будет запущено приложение. То есть системы управления кластерами устанавливаются поверх операционной системы и являются приложениями пользовательского уровня.

Оба изложенных подхода имеют свои достоинства и недостатки. Заметим, что первый из них делает ставку на динамическое планирование и перепланирование (миграция процессов), в то время как в рамках второго подхода используются сложные алгоритмы планирования заданий. Стоит заметить, что в настоящее время более широко используется второй подход. Причина, по всей видимости, кроется в том, что он не требует изменения операционной системы. Распределенные окружения в массе своей представляют экспериментальные образцы, на которых возможны проблемы, связанные с несовместимостью программного обеспечения. В такой ситуации понятен выбор пользователей, отдающих предпочтение системам управления кластерами, работающим на базе обычных операционных систем.

3. Обзор существующих систем управления

В настоящее время все сложнее найти систему управления, которая предлагается как самостоятельный продукт. И это вполне закономерно, поскольку сами по себе эти системы не представляют ценности, так как для использования кластера требуется большой пакет программного обеспечения. Именно поэтому большинство поставщиков кластерных решений включают системы управления в целые комплекты программно-аппаратных средств. Так, Microsoft распространяет свою систему управления вместе с специальной версией операционной системы Windows Server 2003 и реализацией стандарта MPI2.

В настоящее время в мире существует достаточно большое число систем управления кластерами. Вот далеко не полный список наиболее популярных систем управления: Microsoft Compute Cluster Server 2003 [10], Condor [3], PBSPro [9], LSF, Sun Grid Engine. С появлением технологии GRID многие системы пошли в этом направлении и приобрели ряд новых полезных возможностей, как то работа в существенно гетерогенных и распределенных средах. Мы, однако, сконцентрируемся на тех возможностях, которые имеют отношение к исходному назначению систем управления – массовому запуску вычислительно трудоемких заданий.

Мы рассмотрим подробно две популярные системы управления кластерами: Microsoft Compute Cluster Server 2003 и Condor. Это рассмотрение интересно прежде всего тем, что две упомянутые системы имеют

совершенно различные истории и внутреннее устройство, поэтому анализируя их можно сделать выводы об общих тенденциях развития систем управления кластерами.

4. Система управления Microsoft Compute Cluster Server 2003

4.1. История

Долгое время среди операционных систем для кластеров лидировали UNIX-подобные системы. Прежде всего, это объясняется тем, что кластеры функционировали в основном в исследовательских организациях, большинство из которых по историческим причинам ориентированы на использование UNIX. Еще одним сдерживающим фактором распространения Windows на кластерах было отсутствие хорошей системы управления. Впоследствии, однако, многие системы управления были перенесены с UNIX на Windows (Condor, PBS), но при этом остались трудны в эксплуатации для рядового пользователя Windows.

Отсутствие надежной и удобной системы управления причиняло многочисленные неудобства пользователям Windows-кластеров. В настоящее время многие организации для ведения своего бизнеса приобретают огромные парки вычислительных машин, и довольно часто выбор операционной системы решается в пользу Windows из соображений простоты и удобства использования. Если этой же организации необходимо было проводить массовые параллельные вычисления, то еще совсем недавно у нее практически не оставалось иного выбора, как приобретать Linux-кластер, поскольку стабильного и эффективного решения для Windows не существовало. Отсюда возникали различные сложности, связанные с интеграцией Linux-кластера в Windows-окружение, разработкой программного обеспечения под Linux и так далее. Фактически информационная инфраструктура организаций разделялась на Windows и Linux составляющие, поэтому много усилий уходило на то, чтобы организовать взаимодействие между ними.

9 июня 2006 года Microsoft объявила о выходе собственной системы управления Microsoft Compute Cluster Server 2003 (CCS). Система получила хорошую оценку и была признана очень удобной для небольших и средних Windows-кластеров. Крупные поставщики аппаратного обеспечения, такие как Hewlett Packard и IBM, практически сразу после появления CCS стали предлагать на ее основе интегрированные решения для HPC.

Появление CCS позволило снять целый ряд проблем. Существенно упростились процедуры разворачивания вычислительного кластера, его настройки и включения в информационную инфраструктуру организаций. Что особенно важно, вместе с CCS поставляется набор утилит для разработки параллельных приложений (реализация стандарта MPI2). Разработчики получили возможность писать и отлаживать параллельные программы в интегрированной среде Microsoft Visual Studio, а затем отправлять свои задания на кластер через дружественный графический интерфейс. Тем самым Microsoft обеспечил максимально простой переход разработчиков от последовательного программирования к параллельному, что раньше требовало приобретения массы дополнительных знаний, в том числе и об устройстве операционной системы Linux. Администратор Windows, даже без опыта в HPC, в состоянии развернуть CCS-кластер буквально за пару часов. Кластеры же на основе Linux при настройке обычно требуют значительных усилий и существенных знаний операционной системы Linux.

Многие представители университетов заявили, что намерены вести обучение параллельному программированию на основе CCS, поскольку это позволяет студентам использовать имеющиеся знания о Windows и избавляет их от необходимости изучать тонкости работы с Linux, такие как компиляция и конфигурирование приложений, написание сценариев и другие.

При этом можно заметить, что в списке Top500 (список самых высокопроизводительных кластеров) присутствуют буквально единичные кластеры на основе CCS. Некоторые склонны считать, что причина этого в том, что Windows плохо приспособлен для работы на огромном числе машин, становясь очень неудобным в администрировании, в то время как Linux был специально спроектирован для эффективного функционирования в распределенных окружениях с большим числом узлов. Однако это мнение опровергается имеющимся опытом успешных инсталляций. Так, на основе CCS был развернут кластер IBM BladeCenter HS21, включающий 1792 процессора и имеющий производительность 6.52 терафлопс. Также был развернут кластер на основе Dell PowerEdge 1955 из 256 вычислительных узлов с процессорами Xeon 5300 quad-core и имеющий производительность 8.99 терафлопс.

Объективная причина отсутствия CCS-кластеров в Top500 состоит в том, что они изначально ориентированы на несколько других потребителей. Microsoft позиционирует свою систему как массовый продукт для небольших и средних кластеров [4], поскольку именно в этом секторе наблюдается взрывной рост. Высокопроизводительные системы более не сосредоточены в крупных исследовательских и промышленных центрах, находя потребителей среди малого бизнеса и даже индивидуальных пользователей. CCS является идеальным решением для небольших и средних организаций, основной операционной системой которых является Windows.

Несмотря на то, что официальный выпуск системы состоялся в июне 2006, реальное использование системы в различных организациях началось еще раньше. Многие известные и крупные организации приняли участие в бета-тестировании системы. Следует заметить, однако, что начало бета-тестирования фактически означало начало промышленного использования, потому как уже первые версии работали вполне стабильно. Нижегородский государственный университет также начал использование системы практически с самого момента ее появления. CCS была установлена на вычислительный кластер из 12 узлов и использовалась как индивидуальными исследователями, так и в различных проектах. В качестве примеров можно привести проект по интеграции CCS с системой ParaLab, предназначенной для изучения параллельного программирования, проект по сравнению производительности MS MPI и других реализаций MPI и проект «Метакластер», в рамках которого была осуществлена интеграция CCS-кластера в единую вычислительную инфраструктуру Нижегородского университета.

4.2. Общая характеристика

4.2.1. Комплект поставки Microsoft Compute Cluster Server 2003

Microsoft Compute Cluster Server 2003 представляет собой не просто систему управления кластером, а целую интегрированную платформу для поддержки высокопроизводительных вычислений на кластерных системах. CCS поставляется на двух компакт-дисках, на первом их которых находится операционная система Windows Server 2003 Compute Cluster Edition (CCE), а на втором – пакет Microsoft Compute Cluster Pack (CCP). CCE – это специализированная 64-битная операционная система, построенная на базе Windows Server 2003 и ориентированная на использование в области высокопроизводительных вычислений. В частности это означает, что CCE настроена для эффективного функционирования в сетевых конфигурациях с большим числом узлов, а вся функциональность, которая не нужна на вычислительном кластере, из операционной системы исключена. Так, например, CCE-узлы не могут исполнять роль файловых серверов или серверов печати.

CCP – это набор приложений, интерфейсов и утилит, предназначенных для управления кластером и разработки приложений для него. Сюда входят планировщик заданий, интерфейсы администратора и пользователя, утилиты для управления вычислительными узлами и мониторинга их состояния. В CCP входят одновременно несколько интерфейсов доступа к планировщику заданий: графический, командный и интерфейс прикладного программирования (API). Также в комплект поставки CCP входит MS MPI – реализация стандарта MPI2 от Microsoft. Вообще говоря, CCS допускает выполнение MPI приложений, созданных на основе других реализаций MPI (например, MPICH2), но из соображений производительности и совместимости рекомендуется использовать MS MPI. Для наилучшей производительности и эффективного использования центрального процессора MS MPI использует WinSock Direct протокол. MS MPI поддерживает языки программирования C/C++, Fortran77 и Fortran90.

Кроме того, к Microsoft Compute Cluster Server 2003 логически примыкает среда Microsoft Visual Studio 2005, являющаяся интегрированной средой разработки параллельных программ, содержащая компилятор и отладчик MPI и OpenMP программ. При помощи отладчика разработчики могут локально запустить свое MPI приложение, после чего можно приостанавливать его выполнение и просматривать значение переменных в каждом процессе отдельно. Таким образом, разработчики, знакомые со средой Visual Studio имеют возможность начать разработку параллельных приложений для CCS, затратив минимальные усилия.

4.2.2. Системные требования Microsoft Compute Cluster Server 2003

В качестве вычислительных узлов кластера могут быть использованы 64-битные процессоры семейства x86 с, как минимум, 512 Мб оперативной памяти и 4 Гб свободного дискового пространства.

На вычислительных узлах кластера должна быть установлена операционная система Microsoft Windows Server 2003 Compute Cluster Edition. Вместо нее могут быть использованы 64-битные версии Standard Edition или Enterprise Edition.

Вычислительные узлы кластера необязательно должны иметь одинаковую операционную систему или аппаратные характеристики. Однако в целях облегчения развертывания, администрирования и управления ресурсами рекомендуется выделять в кластер узлы с идентичными характеристиками. Наличие узлов с разной конфигурацией может негативно сказаться на производительности кластера, поскольку скорость выполняющихся в параллельном режиме заданий, распределенных по узлам с разными параметрами, будет определяться скоростью самого медленного процессора из числа установленных на этих узлах.

4.2.3. Основные возможности Microsoft Compute Cluster Server 2003

Цель, которую преследовала корпорация Microsoft при создании CCS, – это предоставить пользователям возможность решать большое количество вычислительно трудоемких задач на

распределенных компьютерных ресурсах, работающих под управлением операционной системы Windows. Решение этой задачи неразрывно связано с предоставление удобных средств администрирования подобных вычислительных сетей, поскольку конфигурирование кластера и контроль безопасности при его использовании, как правило, очень сложны.

И необходимо сказать, что корпорация Microsoft в полной мере справилась с поставленной задачей, предоставив комплексное решение, включающее в себя планировщик заданий и несколько интерфейсов доступа к нему, развитые средства администрирования, имеющие привычный интерфейс, и инструменты для разработки параллельных приложений. Таким образом, появление Microsoft Compute Cluster Server 2003 сделало распределенные компьютерные ресурсы проще в развертывании, использовании и управлении.

Рассмотрим, какие возможности предоставляет CCS. Центральным компонентом системы является планировщик заданий, который позволяет отправлять одиночные и пакетные задания в очередь, назначать запуски на определенное время, принудительно снимать задачи с выполнения, просматривать состояния всех запущенных задач, собирать статистику и прочее. При этом пользователи имеют возможность взаимодействовать с планировщиком через графический интерфейс, командную строку или посредством интерфейса прикладного программирования. Первый из них очень прост в освоении и предлагается к использованию по умолчанию. Командный интерфейс может быть удобен пользователям, которые часто выполняют похожие операции. Кроме того, этот интерфейс особенно удобен при администрировании кластера и пакетном запуске заданий, поскольку имеется возможность написания сценариев. Интерфейс прикладного программирования может быть использован для организации взаимодействия каких-либо пользовательских приложений с CCS, например, администратор может реализовать специальную утилиту для отслеживания и анализа отдельных показателей работы кластера.

Кроме того, в CCS по умолчанию включены удобные средства администрирования кластера. Они доступны как через стандартный графический Windows-интерфейс, так и через командную строку, что позволяет создавать специальные сценарии. Также администратор имеет возможность влиять на работу планировщика, создавая специальные фильтры, которые могут использоваться для проверки пользовательских заданий на соответствие различным критериям (например, ограничениям на использование ресурсов кластера).

CCS поддерживает пять различных сетевых топологий, при этом каждый узел может иметь от одной до трех сетевых карт. Правильный выбор используемой топологии необходим для оптимального функционирования вычислительного кластера.

4.2.4. Архитектура Microsoft Compute Cluster Server 2003

На **Ошибка! Источник ссылки не найден.** условно представлена архитектура вычислительного кластера. Далее рассмотрим основные компоненты, входящие в состав Microsoft Compute Cluster Server 2003, охарактеризуем роль каждого из них и обсудим их расположение на узлах кластера:

Рис. 1. Архитектура кластера

- *Вычислительные узлы*. Компьютеры, объединенные сетью в вычислительный кластер, на которые установлено специальное программное обеспечение (будет рассмотрено ниже), позволяющее пользователям запускать свои задания и контролировать их состояние. Среди узлов кластера выделяется ведущий, на который устанавливается управляющее программное обеспечение, которое с одной стороны предоставляет пользователям интерфейс доступа к кластеру, а с другой стороны осуществляет управление вычислительными узлами, посылая на них команды и запросы;
- *Администратор вычислительного кластера и диспетчер заданий*. Это приложения, предназначенные для администрирования кластера и управления пользовательскими заданиями соответственно. Администратор вычислительного кластера служит для конфигурации кластера, а также

мониторинга состояния вычислительных узлов и кластера в целом. Посредством диспетчера заданий пользователи кластера осуществляют создание, отправку и отслеживание состояния своих заданий;

- *Инфраструктура управления*. Представляет собой набор служебных приложений, позволяющих развертывать вычислительные узлы и осуществлять управление заданиями на них. Инфраструктура состоит из специальных служб, выполняющихся на ведущем и каждом из вычислительных узлов и предназначенных для администрирования кластера и управления очередью заданий;
- *Планировщик заданий*. Координирующий центр системы управления, который управляет очередью заданий, распределением ресурсов и выполнением заданий. Располагается на ведущем узле и, взаимодействуя со службой диспетчера узла, получает и анализирует информацию о загрузке и состоянии запущенных заданий;
- *Различные интерфейсы доступа*. В состав пакета Compute Cluster Pack входит три различных интерфейса для управления узлами и заданиями: графический, командный и интерфейс прикладного программирования;
- *MS-MPI*. Окружение, необходимое для выполнения параллельных MPI-программ. Может использовать любой канал связи Ethernet, поддерживаемый Windows Server 2003, а также каналы связи с небольшими задержками и высокой пропускной способностью, например InfiniBand и Myrinet.

4.2.5. Архитектура планировщика заданий и алгоритм его работы

Планировщик заданий является центральным компонентом системы управления, поэтому рассмотрим его более подробно. Планировщик заданий контролирует запуск заданий, распределение ресурсов между заданиями, отслеживает состояние заданий и вычислительных узлов, а также обеспечивает восстановление после сбоев.

На **Ошибка! Источник ссылки не найден.** представлена схема взаимодействия планировщика с другими компонентами CCS.

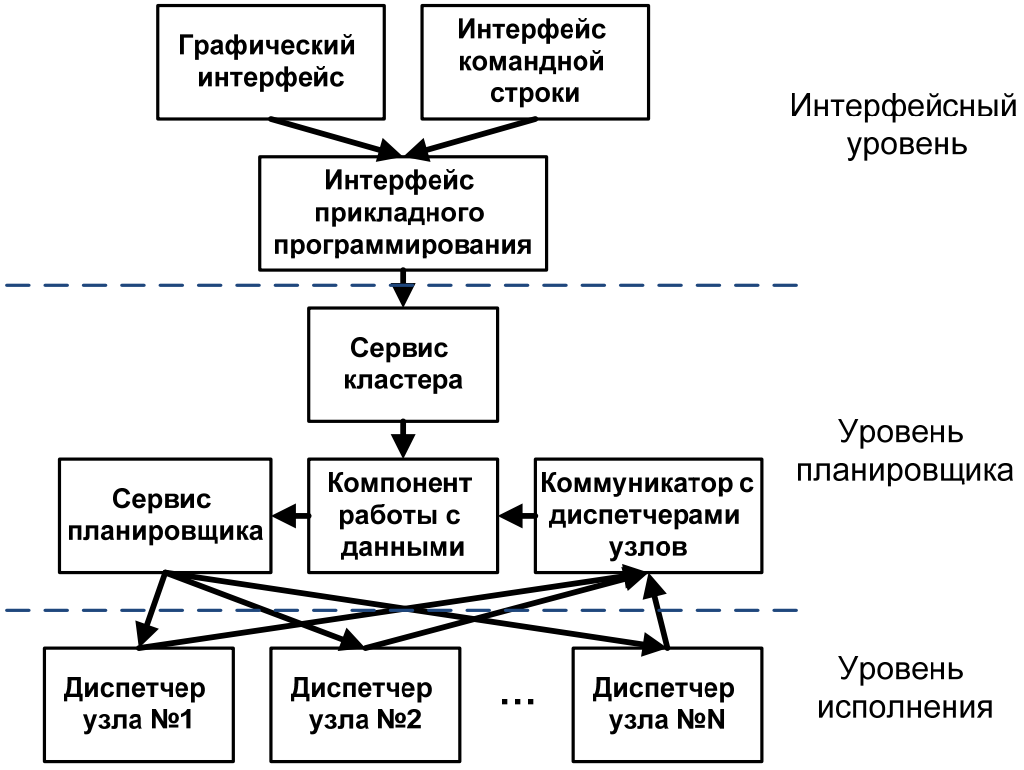


Рис. 2. Схема взаимодействия планировщика с другими компонентами Microsoft Compute Cluster Server 2003

Здесь присутствуют три уровня, рассмотрим назначение каждого из них:

- Верхний уровень – интерфейсный. Именно на этом уровне пользователи и их приложения взаимодействуют с планировщиком;
- Следующий уровень – уровень самого планировщика. На этом уровне принимаются решения о запуске заданий. Также здесь находится очередь заданий и хранится статистика об использовании кластера;

- Нижний уровень – уровень исполнения. На этом уровне создается и контролируется среда выполнения пользовательских заданий. Среда выполнения позволяет настраивать конфигурацию рабочей области, включая переменные среды, параметры рабочих дисков, контекст безопасности, целостность выполнения, специальные механизмы запуска приложений и восстановление после системных прерываний.

Графический интерфейс планировщика заданий и утилиты командной строки получают доступ к функциональности планировщика через интерфейс прикладного программирования, доступ к которому можно получить через библиотеку ccpapi.dll, входящую в CCP SDK.

Коммуникатор с диспетчерами вычислительных узлов используются для сигнализации планировщику об изменении состояния пользовательских заданий. Компонент работы с данными отвечает за хранение всей информации о функционировании кластера. Он хранит информацию об очереди заданий и информацию об использовании ресурсов вычислительных узлов. Сервис планировщика заданий – центральный сервис, непосредственно осуществляющий планирование. Сервис кластера – это удаленный сервис .NET. Он позволяет получать доступ к настройкам всего кластера, осуществлять операции, связанные с вычислительными узлами, заданиями, а также получать информацию об использовании ресурсов. Этот сервис представляет собой точку входа планировщика.

На уровне исполнения функционирует диспетчер вычислительного узла, контролирующий работу узлов и выполняющихся на них заданий. Диспетчер осуществляет фактический запуск пользовательских заданий и отслеживает загрузку вычислительного узла.

Ресурсы кластера распределяются согласно приоритету заданий. Если приоритеты заданий совпадают, то сначала будет запущено то приложение, которое было поставлено в очередь первым. Планировщик для запуска выбирает наиболее подходящие среди всех доступных узлов. Также пользователь имеет возможность указать, какие из узлов может использовать его задание, тогда планировщик будет выбирать узлы только из списка допустимых узлов.

Планировщик поддерживает возможность уплотнения расписания, то есть, например, он может назначить запуск низкоприоритетной задачи вперед высокоприоритетной в том случае, если последняя ожидает освобождения большого количества ресурсов для своего запуска. При этом гарантируется, что запуск низкоприоритетной задачи не приведет к увеличению времени ожидания высокоприоритетного задания.

Существует полезная возможность расширения планировщика путем использования фильтров. Администратор может управлять заданиями с помощью фильтров, которые перед началом выполнения заданий проверяют их на соответствие определенным критериям.

Ниже приведены некоторые из возможных критериев.

- Проверка проекта. Проверяется, что имя проекта принадлежит действительному проекту, для которого разрешена работа на кластере, а пользователь, запустивший задание, является членом данного проекта.
- Обязательная политика. Проверяется, чтобы не была настроена бесконечная продолжительность выполнения (это может привести к перегрузке кластера), и чтобы задание не использовало ресурсы сверх выделенного пользователю лимита.
- Время использования. Осуществляется контроль за соблюдением выделенного пользователю лимита времени. В отличие от обязательной политики ограничением выступает общее время, доступное пользователю для всех заданий.

Кроме того, фильтры могут перед выполнением задания проверить его на соответствие условиям лицензирования. Фильтры являются мощным средством управления заданиями и должны использоваться на каждом кластере. Планировщик вызывает фильтры в процессе разбора файла задания, содержащего свойства задания. Дальнейшие действия планировщика заданий зависят от кода завершения программы фильтра. Существуют следующие варианты.

- 0: задание можно отправлять с неизменными параметрами;
- 1: задание можно отправлять после изменения параметров;
- При появлении другого значения задание отклоняется.

4.3. Работа с системой

4.3.1. Установка и настройка Microsoft Compute Cluster Server 2003

CCS поддерживает кластеры, состоящие из одного ведущего узла и одного или нескольких вычислительных узлов. При этом ведущий узел управляет доступом ко всем ресурсам и представляет собой

единую точку для управления и развертывания ресурсов, исполнения заданий на вычислительном кластере. CCS может использовать существующую инфраструктуру Active Directory для обеспечения безопасности и управления учетными записями, а также может быть интегрирован с такими средствами управления, как MOM (Microsoft Operations Manager) 2005, SMS (Systems Management Server) 2003, MMC (Microsoft Management Console) 3.0.

Для установки Compute Cluster Server необходимо выполнить следующие действия:

1. Убедитесь, что компьютеры, которые вы хотите сконфигурировать в качестве узлов кластера, удовлетворяют аппаратным требованиям и требованиям к программному обеспечению;
2. Проверьте, что CCE поддерживает сетевую топологию вашего кластера;
3. Настройте ведущий узел кластера, выполнив следующие шаги:
    - Установите на него поддерживаемую операционную систему (например, CCE);
    - Добавьте компьютер в состав какого-либо существующего домена Active Directory, либо создайте новый домен и назначьте ведущий узел контроллером домена;
    - Установите Compute Cluster Pack;
    - Если развертывание вычислительных узлов будет производиться с помощью Remote Installation Services (RIS), то необходимо создать новый том на жестком диске и установить RIS.

По окончании установки пакета Compute Cluster Pack на ведущем узле, на его экране появится окно **To Do List**, в котором будет показано какие шаги необходимо выполнить для завершения конфигурирования вычислительного кластера.

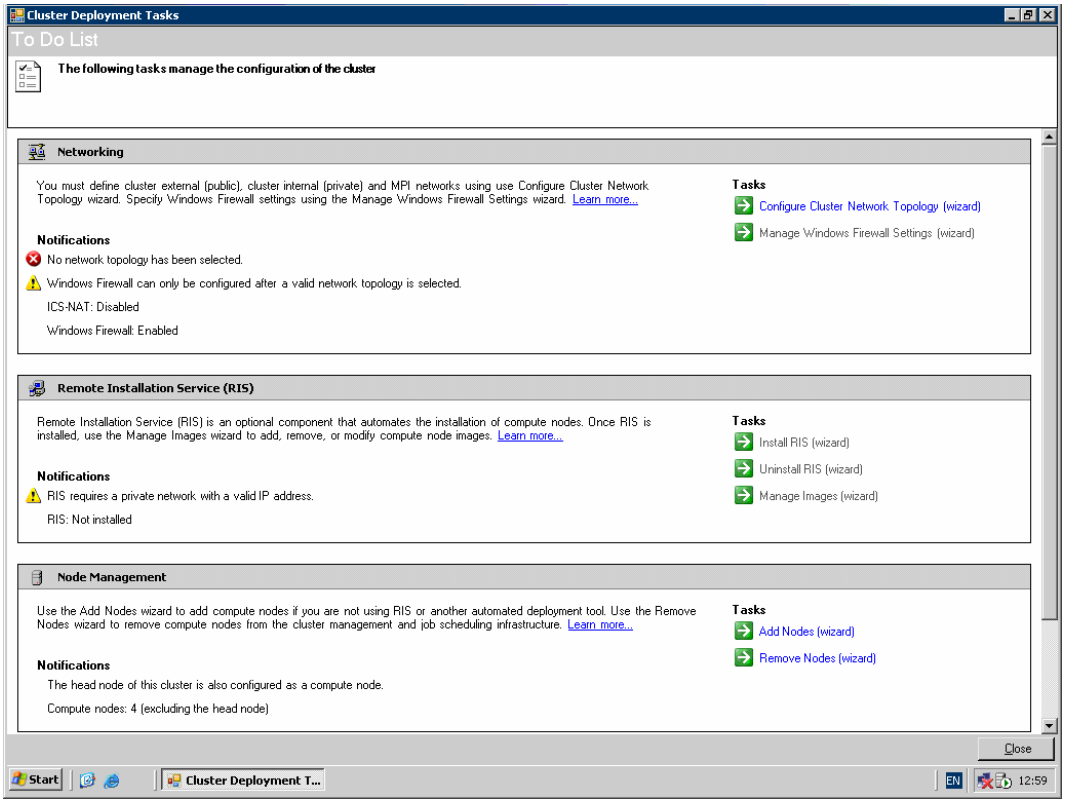


Рис. 3. Окно с настройками конфигурации кластера

Необходимо сделать следующее:

- Сконфигурируйте сетевые интерфейсы на ведущем узле в зависимости от топологии сети;
- Настройте Windows Firewall;
- Если планируется использование RIS, то необходимо создать установочный образ;
- В случае ручной установки необходимо установить Compute Cluster Pack на все вычислительные узлы;
- Добавьте пользователей и администраторов кластера.

Во время этой процедуры на ведущем узле кластера устанавливаются следующие сервисы:

- Management Service – управление кластером и обнаружение узлов;

- Scheduler Service – постановка задач в очередь, планирование их запуска, распределение ресурсов, исполнение задач;
- SDM Store Service – операции чтения/записи в хранилище данных System Definition Model (SDM), управление целостностью;
- MPI Service – реализация MPI на основе MPICH2;
- Node Manager Service – устанавливается на ведущем узле только в том случае, когда он сконфигурирован также и в качестве вычислительного узла, этот сервис взаимодействует с Scheduler Service и отвечает за исполнение заданий;
- SQL Service – Microsoft SQL Server 2000 Desktop Engine (MSDE).

На вычислительных узлах кластера устанавливаются следующие сервисы:

- Management Service – взаимодействует с одноименным сервисом на ведущем узле, отвечает за поддержку управления кластером и обнаружения узлов со стороны вычислительного узла;
- MPI Service – реализация MPI на основе MPICH2;
- Node Manager Service – взаимодействует с Scheduler Service на ведущем узле и отвечает за исполнение заданий.

После установки программного обеспечения на ведущий и вычислительные узлы необходимо разрешить использование вычислительных узлов. Для того чтобы это сделать, необходимо зайти под пользователем с административными правами на ведущий узел CCS и открыть консоль администратора кластера (Compute Cluster Administrator) и выбрать пункт Node Management. В списке вычислительных узлов все установленные узлы будут иметь состояние Pending Approval (Ожидающий подтверждения). Щелкните на узле правой кнопкой мыши и выберите в открывшемся контекстном меню пункт Approve (Подтвердить). Еще раз щелкните правой кнопкой мыши на узле и выберите пункт Resume (Продолжить работу) для начала работы узла. Для приостановки использования узла выберите пункт Pause (Пауза), а чтобы удалить его из списка – Remove (Удалить).

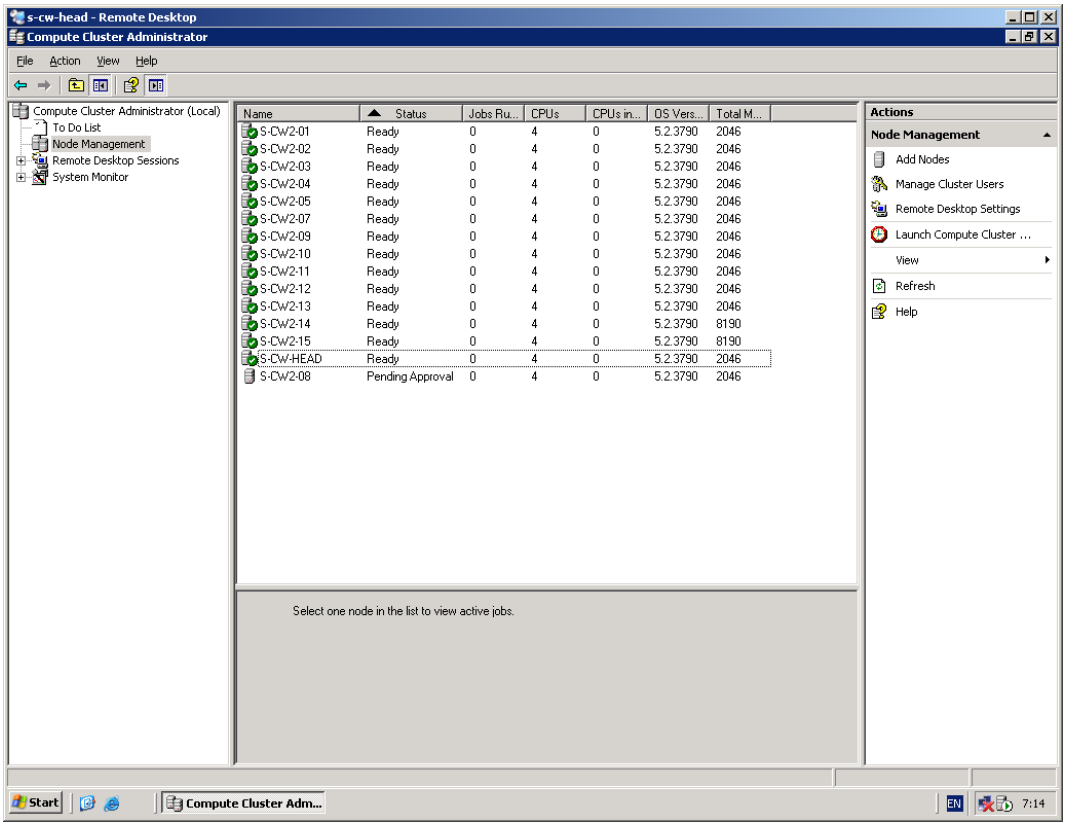


Рис. 4. Управление вычислительными узлами

CCS постоянно отслеживает состояние узлов, и если произошел аппаратный сбой или отключение узла, на котором выполнялась задача, произойдет перезапуск этого задания. Кроме того Compute Cluster Service осуществляет мониторинг некоторых характеристик производительности узлов. Кроме того можно графически отобразить изменение этих характеристик во времени. Для этого обратимся к вкладке System Monitor (Системный монитор). Чтобы открыть системный монитор для конкретного вычислительного узла, выделите нужный узел на странице Node Management, щелкнув по нему правой кнопкой мыши, и выберите

команду Start Perfmon Task (Запустить системный монитор). Все сеансы системного монитора отображаются на панели переходов под заголовком System Monitor (Системный монитор), как показано на **Ошибка! Источник ссылки не найден.**.

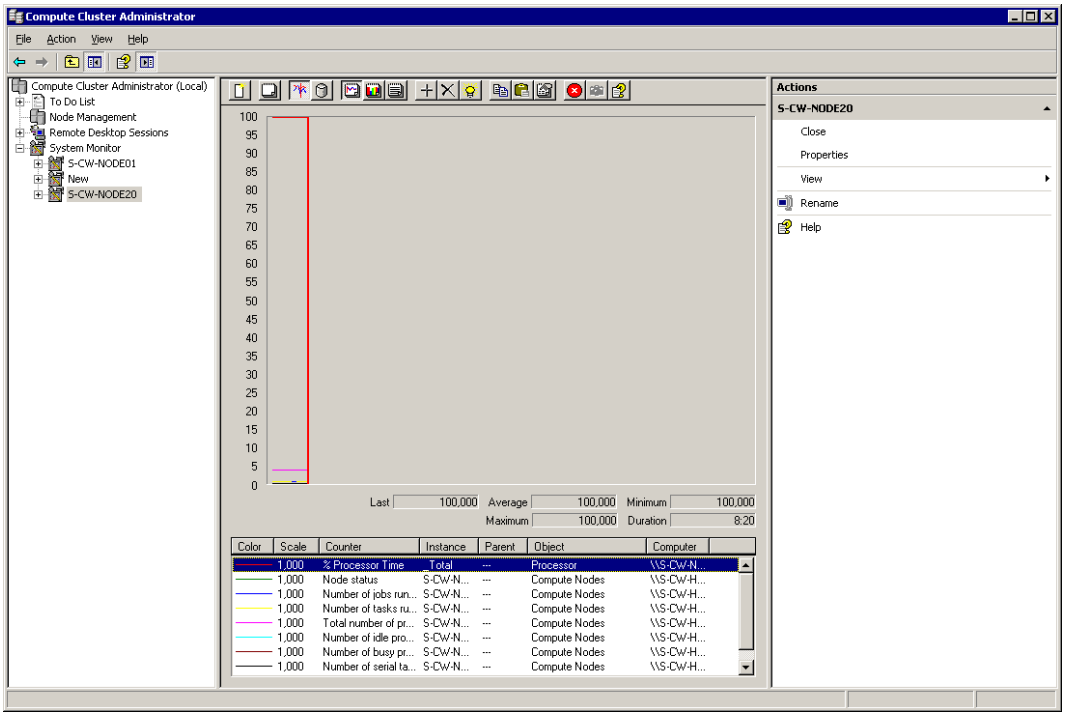


Рис. 5. Системный монитор

4.3.2. Работа с заданиями

Для эффективного использования вычислительного ресурса кластера необходимо обеспечить не только непосредственный механизм запуска заданий на выполнение, но и предоставить среду управления ходом выполнения заданий, решающую, в том числе, задачу эффективного распределения ресурсов. Эти задачи эффективно решаются с использованием встроенных в CCS средств.

Дадим определение важнейшим понятиям, используемым в CCS:

- *Задание (job)* – запрос на выделение вычислительных ресурсов кластера для выполнения задач. Каждое задание может содержать одну или несколько задач;
- *Задача (task)* – команда или программа (в том числе, параллельная), которая должна быть выполнена на кластере. Задача не может существовать вне некоторого задания, при этом задание может содержать как несколько задач, так и одну;
- *Планировщик заданий (job scheduler)* – сервис, отвечающий за поддержание очереди заданий, выделение системных ресурсов, постановку задач на выполнение, отслеживание состояния запущенных задач;
- *Процессор (processor)* – один из, возможно, нескольких вычислительных устройств узла;
- *Очередь (queue)* – список заданий, отправленных планировщику для выполнения на кластере. Порядок выполнения заданий определяется принятой на кластере политикой планирования;
- *Список задач (task list)* – эквивалент очереди заданий для задач каждого конкретного задания. Порядок запуска задач определяется правилом FCFS (First come – first served, первыми будут выполнены задачи, добавленные в список первыми), если пользователь специально не задал иной порядок.

Планировщик заданий CCS работает как с последовательными, так и с параллельными задачами. Последовательной называется задача, которая использует ресурсы только 1 процессора. Параллельной же называется задача, состоящая из нескольких процессов (или потоков), взаимодействующих друг с другом для решения одной задачи. Как правило, параллельным задачам для эффективной работы требуется сразу несколько процессоров. При этом, в случае использования MPI в качестве интерфейса передачи сообщений, процессы параллельной программы могут выполняться на различных узлах кластера.

Как уже было сказано выше, CCS включает собственную реализацию стандарта MPI2: библиотеку Microsoft MPI (MS MPI). В случае использования MS MPI в качестве интерфейса передачи сообщений

необходимо запускать параллельные задачи с использованием специальной утилиты mpiexec.exe, осуществляющей одновременный запуск нескольких экземпляров параллельной программы на выбранных узлах кластера. Кроме того для компиляции приложения, использующего MS MPI, необходимо установить Microsoft Compute Cluster Pack SDK.

Важно отметить, что непосредственным запуском задач занимается планировщик, а пользователь может лишь добавить задачу в очередь, так как время ее запуска выбирается системой автоматически в зависимости от того, какие вычислительные ресурсы свободны и какие задания ожидают в очереди выделения им ресурсов. Таким образом, для исполнения программы в CCS необходимо выполнить следующие действия:

- Создать задание с описанием вычислительных ресурсов, необходимых для его выполнения;
- Создать задачу. Задача определяется при помощи той или иной команды, выполнение которой приводит к запуску на кластере последовательных или параллельных программ. Например, параллельная задача описывается при помощи команды mpiexec.exe с соответствующими параметрами (список узлов для ее запуска, имя параллельной программы, аргументы командной строки программы и др.);
- Добавить задачу к созданному ранее заданию.

Для запуска параллельного приложения через CCS с помощью графического интерфейса откройте **Computer Cluster Job Manager** (Менеджер заданий) для запуска программы на кластере. Менеджер заданий может быть установлен и исполняться на машине, не принадлежащей вычислительному кластеру, что дает возможность использовать кластер удаленно. Далее необходимо выполнить следующие шаги:

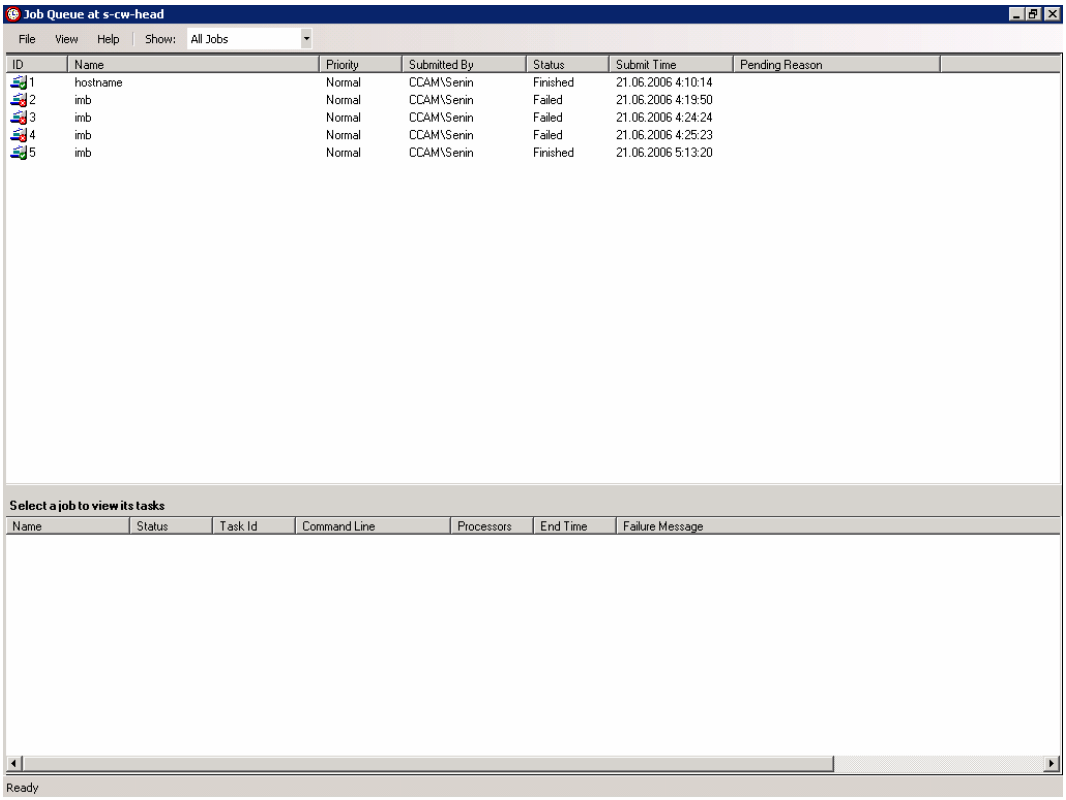


Рис. 6. Менеджер заданий

- В открывшемся окне менеджера заданий выберите пункт меню **File>Submit Job** для постановки нового задания в очередь;
- В окне постановки задания в очередь введите имя задания (поле **Job Name**), при необходимости измените приоритет задания (пользовательские задания с большим приоритетом будут выполнены раньше заданий с меньшим приоритетом). Перейдите на вкладку **Processors**;
- На вкладке **Processors** введите минимальное и максимальное число процессоров, необходимых для выполнения. Под максимальным здесь понимается оптимальное для задания число процессоров (именно столько будет выделено в случае низкой загрузки кластера). Гарантируется, что задание не начнет выполняться, если на кластере доступно менее минимального числа процессоров. Дополнительно можно ввести предполагаемое время работы задания (это поможет планировщику эффективнее распределить системные ресурсы) – панель **Estimate run time for this job**. Если Вы

хотите, чтобы для задания вычислительные ресурсы оставались зарезервированными в течение указанного времени, даже после того, как все задачи задания завершили свою работы, то поставьте галочку около пункта **Run job until end of run time or until canceled**. Таким образом, Вы сможете запускать новые задачи в рамках задания даже после того, как все задачи, указанные первоначально, выполнены. Перейдите на вкладку **Tasks** для добавления в задание новых задач;

- Введите имя задачи (поле «**Task Name**») и команду, которую необходимо выполнить (поле «**Command Line**»). Запуск задач, разработанных для MS MPI, необходимо осуществлять с использованием специальной утилиты **mpiexec.exe**, которая принимает в качестве параметров имя параллельной программы, список узлов, на которых произойдет запуск, и параметры запускаемой программы. Список узлов задается параметром «**-hosts**». При этом в случае, если узлы были выделены планировщиком автоматически, список узлов будет содержать переменная окружения **CCP_NODES**, значение которой и следует передавать в качестве параметра утилите. Добавленная задача появится в списке задач текущего задания (список **This job contains the following tasks**). Выделите ее в списке и нажмите кнопку **Edit** для редактирования дополнительных параметров задачи;
- В открывшемся окне введите файл, в который будет перенаправлен стандартный поток вывода консольного приложения (поле **Standard Output**). Кроме того, можно указать файл стандартного потока ввода (поле **Standard Input**), файл стандартного потока ошибок (поле **Standard Error**), рабочую директорию запускаемой программы (поле **Work Directory**) и ограничение по времени на продолжительность выполнения задачи (суммарное время выполнения задач не должно превышать оценки времени выполнения задания) – поле **Limit task run time to**. Выберите вкладку **Processors**;
- На вкладке **Processors** в верхнем списке (**Select task to view settings**) выделите задачу, настройки которой Вы хотите изменить, и укажите минимальное и максимальное число процессоров для выбранной задачи (поля **Min. required** и **Max. required**) в случае, если Вы хотите, чтобы планировщик выбрал узлы для запуска автоматически (**Use any available processors on any nodes**). Если Вы хотите вручную указать узлы для запуска, выберите пункт **Select nodes required for this task** и поставьте флажки около требуемых узлов в нижнем списке. Нажмите **OK** для сохранения внесенных изменений и возвращения к настройкам задания;
- Перейдите на вкладку **Advanced** и выберите пункт **Use any available nodes** для автоматического выбора узлов для задания. Если Вы хотите вручную указать узлы, на которых будет выполняться задание, выберите пункт **Use only these nodes**. Учтите, что в том случае, если Вы вручную указали узлы для запуска задачи, то они также должны быть выбраны для всего задания (в случае неавтоматического распределения). Установите флаг **Use the allocated nodes exclusively for this job** для того, чтобы запретить запуск нескольких заданий на одном узле. Нажмите кнопку **Submit** для добавления задания в очередь;
- Введите имя и пароль пользователя, имеющего право запуска задач на кластере, и нажмите **OK**;
- Задание появится в очереди. По окончании выполнения его состояние изменится на **Finished**. В файле, указанном в настройках задачи для перенаправления стандартного потока вывода, содержится результат работы программы.

Для того чтобы остановить ваше задание, щелкните по нему правой кнопкой мыши и выберите в появившемся меню пункт **Cancel your job**.

Также CCS поддерживает два особых вида заданий:

- **Параметрическое множество задач (parametric sweep)** – одна и та же программа (последовательная или параллельная), несколько экземпляров которой запускается (возможно, одновременно) с разными входными параметрами и разными файлами вывода;
- **Поток задач (task flow)** – несколько задач (возможно, одна и та же программа с разными входными параметрами) запускаются в определенной последовательности. Последовательность запуска объясняется, например, зависимостью некоторых задач последовательности от результатов вычислений предыдущих.

4.3.3. Командный интерфейс

CCP предоставляет командный интерфейс для управления заданиями и вычислительными узлами. Эти команды позволяют пользователям создавать, отправлять и контролировать задания, а администраторам – управлять самим кластером. Также командный интерфейс может быть использован для написания сценариев, так, например, администраторы могут автоматизировать некоторые операции по работе с очередью заданий или с вычислительными узлами.

Собственно сам интерфейс состоит из пяти ключевых команд: **`job`**, **`task`**, **`cluscfg`**, **`clusrun`** и **`node`**. Эти команды, вызываемые с различными аргументами, дают доступ практически ко всей

функциональности планировщика заданий. Для удобства пользователей вся необходимая им функциональность доступна через команды **job** и **task**. Фактически эти две команды полностью покрывают всю функциональность графического менеджера заданий.

Сочетание, например, следующих трех команд: **job new, job add, job submit** и **job view**, с соответствующими аргументами, позволяет создать новое задание, поставить его в очередь, подтвердить необходимость запуска, а затем узнать его состояние. Для быстрого запуска задания можно воспользоваться, например, следующей строкой:

```
job submit /numprocessors:4 /stdin:infile /stdout:outfile mpiexec myapp.exe
```

В результате на кластере будет запущено параллельное приложение `myapp` из 4 процессов, которое в качестве стандартного потока ввода использует файл с именем **infile**, а в качестве стандартного потока вывода – **outfile**.

Командный интерфейс достаточно хорошо документирован [5] и достаточно прост в освоении. Кроме того, Microsoft предлагает широкий выбор [6] уже готовых сценариев для работы с CCS, которые будут полезны каждому администратору.

4.3.4. Интерфейс прикладного программирования

Если обратить внимание на схему взаимодействия планировщика с другими компонентами Microsoft Compute Cluster Server (рис. 2), то можно заметить, что все интерфейсы доступа реализованы путем использования интерфейса прикладного программирования. Этот интерфейс предоставлен в виде динамически загружаемой библиотеки **ccpapi.dll**, которая входит в состав CCP SDK. Эта библиотека содержит COM-интерфейсы, необходимые для взаимодействия с планировщиком заданий. Основные их них – это интерфейсы **ICluster, INode, IJob** и **ITask**, являющиеся абстракциями кластера, вычислительного узла, пользовательского задания и задачи соответственно.

Предоставленный Microsoft программный интерфейс работы с планировщиком может быть полезен в ряде случаев. Администратор на его основе может создавать специализированные утилиты для работы с кластером, например, осуществлять подробное протоколирование событий или сбор, анализ и представление в графическом виде статистики об использовании кластера. Для пользователей может быть создан некий особый графический интерфейс доступа, который может быть ориентирован на использование при решении какого-либо специфического класса задач. Например, если необходимо провести множество экспериментов, между которыми существуют некоторые зависимости, и расписание последующих запусков необходимо корректировать в соответствии с результатами прошедших тестов.

Интерфейс прикладного программирования имеет подробную документацию [7], включающую в себя примеры программного кода для работы с CCS. Мы не станем подробно описывать все интерфейсы, а лишь приведем их полный список и охарактеризуем каждый их них. Всего **ccpapi.dll** включает 11 интерфейсов.

- **ICluster**. Является абстракцией кластера, предоставляет методы работы с кластером, заданиями и задачами;
- **IClusterCounter**. Служебный класс, предоставляющий статистическую информацию о кластере, такую как общее число задач, число занятых узлов и так далее;
- **IClusterEnumerable**. Представляет коллекцию объектов. Многие методы интерфейсов возвращают коллекцию, например **ICluster::ListJobs** возвращает коллекцию всех задач на кластере;
- **IExecutionResult**. Хранит результат выполнения операции;
- **IJob**. Интерфейс для работы с заданиями, позволяет добавлять в него задачи;
- **IJobCounter**. Предназначен для получения статистической информации о задачах в одном задании: число запущенных, отработавших и так далее;
- **INameValue**. Служебный интерфейс, представляет пару имя/значение;
- **INameValueCollection**. Предназначен для хранения коллекции интерфейсов INameValue;
- **INode**. Абстракция вычислительного узла, хранит их аппаратные характеристики;
- **IResourceUsage**. Интерфейс для хранения информации об использовании ресурсов заданием или задачей;
- **ITask**. Интерфейс, представляющий пользовательское приложение.

Кроме интерфесов, в ccpapi.dll включены несколько перечислимых типов:

- **JobPriority**. Приоритет задания;
- **JobStatus**. Статус задания;

- **`NodeStatus`**. Статус вычислительного узла;
- **`TaskStatus`**. Статус задачи.

Рассмотрим основные принципы работы с интерфейсом прикладного программирования. Прежде чем работать с COM-интерфейсами из **`ccpapi.dll`** необходимо объявить их в своем приложении. Это делается по-разному для языков C++ и C#, для простоты изложения рассмотрим этот процесс на примере языка C#. Для этого в Microsoft Visual Studio необходимо открыть окно с настройками проекта и, выбрав слева пункт **Common Properties>References**, необходимо добавить ссылку на **`ccpapi.dll`**. После этого в своем приложении можно создавать экземпляры интерфейсов, предварительно объявив пространство имен, в котором они находятся: **`Microsoft.ComputeCluster`**.

Ниже приведен код приложения, которое отправляет в очередь одно пользовательское задание. Мы опустили код обработки ошибок и другие несущественные для понимания подробности.

```
using System;
using System.Text;
using System.Collections.Generic;
using Microsoft.ComputeCluster;

namespace CCPAPI_Using_Example
{
  class CCPAPI_Program
  {
    static void Main(string[] args)
    {
      //объявление всех необходимых переменных
      ICluster cluster = new Cluster();
      IJob job = null;
      ITask task = null;

      // установление соединения с ведущим узлом CCS
      cluster.Connect("ccs_head_node");

      // создание задания и установка его параметров
      job = cluster.CreateJob();
      job.Name = "First Job";

      // создание задачи и установка ее параметров
      task = cluster.CreateTask();
      task.Name = "First Task";
      task.CommandLine = "mpiexec -n 4 parallel_application.exe";
      task.Stdout = "outfile.txt";

      // добавление задачи в задание
      job.AddTask(task);
      // постановка задания в очередь
      cluster.QueueJob(job, null, null, true, 0);

      // закрытие соединения с кластером
      cluster.Dispose();
    }
  }
}
```

Итак, прежде чем работать с планировщиком, необходимо установить соединение с кластером, а именно с головным узлом CCS. После завершения работы с планировщиком нужно закрыть соединение. Работа с заданиями и задачами ничем не отличается от работы с простыми классами языка C#. Необходимо лишь понимание основных принципов работы с интерфейсами CCS. Задание есть пакет задач, поэтому для каждой задачи пользователь должен задать все необходимые параметры (устанавливая соответствующие атрибуты интерфейса) и поместить их в задание, после чего нужно отправить задание в очередь на выполнение.

5. Система управления Condor

5.1. История

Система управления кластером Condor разрабатывается в университете Висконсин-Мэдисон (University of Wisconsin-Madison) с 1988 года. Это одна из широко известных и часто используемых систем управления кластерами. Широкому распространению системы способствует возможность ее бесплатного

использования. Любой желающий может скачать дистрибутив с сайта разработчиков и развернуть вычислительный кластер под управлением этой системы. Круг пользователей этой системы чрезвычайно широк. С одной стороны ее используют частные лица и представители малого бизнеса в виду того, что она бесплатна и поддерживает самые разнообразные аппаратные архитектуры и операционные системы. С другой стороны, Condor используется и для управлениями огромными кластерами, так как это одна из самых надежных и проверенных временем систем. В настоящее время под управлением этой системы находится около ста тысяч компьютеров по всему миру.

С самого начала развития проекта был взят курс на создание системы поддержки вычислений на распределенных рабочих станциях. Разработчики системы верили в то, что будущее принадлежит не мэйнфреймам, а персональным компьютерам. Их прогноз оправдался, сегодня в мире насчитывается около миллиарда компьютеров, и большинство из них именно персональные. Несмотря на то, что Condor способен эффективно управлять однородными вычислительными кластерами, это не является его единственным предназначением, он имеет еще одно важное преимущество перед другими системами. Оно состоит в том, что он позволяет распределять задачи по существующей в организации сети рабочих станций, заставляя компьютеры работать в свободное время (то есть в то время, когда они бы простаивали). Именно поэтому эту систему еще называют «охотником за свободными станциями». Таким образом, Condor превращает набор независимых компьютеров и кластеров в единую высокоэффективную вычислительную инфраструктуру.

При этом пользователь вместо того, чтобы запускать задание на своем компьютере, обращается к системе Condor, которая ищет незанятые машины и осуществляет на них запуск пользовательского задания. Когда машина, на которой происходят вычисления, перестает быть свободной (например, пользователь вошел в систему), Condor приостанавливает выполнение и осуществляет миграцию задания на свободную машину. При этом выполнение задания продолжается не с самого начала, а с момента прерывания. Если же свободных машин в наличии нет, то задание будет ожидать освобождения ресурсов. Подобная технология позволила решить вторую важную задачу, которую ставили перед собой разработчики системы – обеспечить равномерную нагрузку на узлы кластера путем миграции процессов.

Идеи, заложенные в основу этой системы, во многом перекликаются с технологией GRID. Интеграция Condor с инструментарием Globus позволила решить некоторые очень трудоемкие задачи, например, проблему «Nug30» [23]. По условиям этой задачи предполагается, что имеется множество точек, куда должно быть доставлено какое-либо воздействие, и требуется определить стоимость передачи этих воздействий для каждой пары точек. Эта задача была решена за неделю. Над ее решение в каждый момент времени в среднем работало 650 процессоров (максимальное количество 1009), установленных на компьютерах пяти различных платформ, физически расположенных в нескольких штатах США и в Италии.

При рассмотрении функциональности системы Condor мы ограничимся изучением варианта системы, работающим под управлением ОС Windows (основные возможности Condor являются общими для версий для ОС Windows и Unix).

5.2. Общая характеристика

5.2.1. Комплект поставки

Система Condor распространяется бесплатно и имеет достаточно либеральную лицензию [22]. Последнюю версию системы всегда можно загрузить с официального сайта проекта [3], для чего достаточно лишь указать свое имя, адрес электронной почты и название организации. Condor способен функционировать на большом количестве платформ. Под платформой подразумевается сочетание аппаратной архитектуры и операционной системы. Для каждой из платформ на сайте доступен дистрибутив в виде либо архива, либо самоустанавливающегося модуля (MSI на Windows или RPM на UNIX-подобных системах).

В отличие от Microsoft Compute Cluster 2003, Condor представляет собой именно систему управления и не включает в себя никаких других компонент. При этом, однако, Condor способен взаимодействовать с множеством свободно распространяемых библиотек и приложений, позволяя владельцам кластеров решать практически любые задачи. Так, параллельные приложения легко могут быть запущены посредством свободно распространяемого пакета MPICH или PVM. Также для Condor существует набор администраторских утилит, написанных сторонними разработчиками.

После инсталляции и конфигурации системы на вычислительных узлах кластера запустятся специальные сервисы, обеспечивающие функционирование системы. После этого Condor готов к использованию.

5.2.2. Системные требования

Как уже было сказано выше, Condor способен функционировать на большом многообразии платформ, и имеет при этом очень скромные системные требования. Так, при инсталляции на Windows предъявляются следующие требования:

- Windows 2000 (или выше), или Windows XP.
- 300МБ свободного дискового пространства.
- Файловая система NTFS или FAT. Из соображений безопасности следует отдавать предпочтение NTFS.

Собственно требования самого Condor исчерпываются этим небольшим списком. При этом нельзя путать требования системы управления и требования пользовательских приложений. Так, возможно, что на узлах необходимо будет установить некоторое дополнительное программное обеспечение, например MPICH, который может потребоваться пользовательским приложениям.

Рассмотрим теперь на каких платформах способен функционировать Condor. Поддерживаются следующие архитектуры процессоров:

- Hewlett Packard PA-RISC;
- Sun SPARC Sun4m, Sun4c, Sun UltraSPARC;
- Intel x86;
- Alpha;
- PowerPC;
- Itanuim I64;
- Opteron x86_64.

Далее приведем список поддерживаемых операционных систем:

- Solaris;
- Red Hat Linux;
- Fedora Core;
- SuSE Linux;
- HP-UX;
- AIX;
- Mac OS X;
- Windows 2000, XP, Vista.

5.2.3. Основные возможности

Создатели системы Condor заявляют, что она предназначена не для HPC, а для HTC, что расшифровывается как High-Throughput Computing – вычисления с высокой пропускной способностью. Отличие от HPC состоит в том, что максимизируется не скорость работы индивидуальной задачи, а общее число задач, выполненных за длительный срок. Другими словами, идея состоит в том, чтобы использовать все имеющиеся ресурсы, пусть даже некоторые компьютеры имеют низкую производительность и доступны не 24 часа в сутки. Собственно эта идеология и определила многие черты системы Condor: возможность работы на существенно гетерогенных вычислительных ресурсах, поддержка широкого спектра платформ, упор на динамическое управление ресурсами, а не интеллектуальные алгоритмы априорного планирования.

Таким образом, Condor стремится по максимуму использовать вычислительную мощность компьютеров, объединенных в сеть, а значит применим и для управления вычислительными кластерами. При этом было бы интересно провести сравнение с другими системами, идет ли подобная универсальность системы Condor в ущерб производительности на гомогенном кластере.

Рассмотрим типичный сценарий работы с Condor. Пользователь отправляет задание, после чего Condor находит подходящий свободный узел и запускает на нем задание. При этом, если запуск произошел на рабочей станции, которую впоследствии занимает пользователь, то Condor приостанавливает исполнение приложения, создает контрольную точку и перенаправляет задание на другой подходящий узел. Под контрольной точкой понимается образ процесса операционной системы, который может быть перенесен на другую машину и возобновлен на ней.

Condor может быть очень полезен в случае, если задание должно быть запущено большое число раз (сотни и более), например на различных наборах входных данных. Пользователь создает простой

конфигурационный файл и всего одной командой подтверждает запуск задания. После этого все множество заданий может быть запущено и работать в параллели на десятках машин.

При этом не требуется наличие администраторских прав на машине, на которой происходит исполнение приложения, поскольку Condor имеет механизм перенаправления системных вызовов. Так, например, файловый ввод-вывод пользовательского приложения транслируется по сети и исполняется на том самом узле, откуда был произведен запуск приложения.

Condor имеет развитые средства управления вычислительными ресурсами, которые основаны на технологии ClassAds. Далее мы рассмотрим эту технологию подробнее, а сейчас скажем лишь, что она основывается на поиске совпадений между пользовательскими запросами и списком доступных ресурсов. Данный подход упрощает работу как пользователей, так и администраторов, поскольку в других системах управления для разделения задач на классы требуется создавать несколько очередей заданий, каждую из которых необходимо настраивать и постоянно поддерживать.

Оригинально Condor разрабатывался под UNIX-подобные окружения, поэтому именно на них доступны все функциональные возможности системы. Версия же для Windows имеет некоторые ограничения, например недоступны режимы Standard и PVM. Далее перечислим основные возможности Windows-версии Condor.

- Возможность добавлять и запускать задания, управлять очередями заданий на кластере Windows-машин.
- В дистрибутив включены все основные утилиты: condor_q, condor_status, condor_userprio. Не включена только condor_compile.
- Возможность настраивать политику запуска через ClassAds. Сохраняется вся та же информация об использовании ресурсов, что и на UNIX: средняя загрузка, размер свободного объема оперативной памяти, число целочисленных операций и операций с плавающей запятой в единицу времени, время, прошедшее с последнего сигнала клавиатуры и мыши, и так далее.
- Механизмы безопасности.
- Поддержка SMP машин (многопроцессорные или многоядерные).
- Condor способен запускать задачи с самым низким приоритетом операционной системы, что позволяет оказывать минимальное воздействие на вычислительные узлы, когда это необходимо. Также задания могут быть приостановлены, завершены путем посылки сообщения WM_CLOSE или автоматически завершены при нарушении политик использования вычислительных ресурсов. Condor может приостанавливать исполнение в случае, если на узле наблюдается активность пользователя, и восстанавливать работу задания, если пользователь не работает больше определенного интервала времени.
- Condor корректно управляет заданиями, которые порождают многочисленные процессы. Если, например, требуется снять задание с выполнения, то завершается не только процесс самого задания, но и все его дочерние процессы.
- Пользователи и администраторы могут получать информацию от Condor посредством электронной почты или файлов с протоколами.
- Condor включает приложение с дружественным графическим интерфейсом, которое позволяет проводить полную инсталляцию и деинсталляцию системы управления. Информация, указанная пользователем, заносится в реестр. Программа установки может обновить версию приложения с минимальными усилиями.

5.2.4. Архитектура

Архитектура Condor имеет две составляющие: аппаратную и программную. Первая описывает то, какие роли могут выполнять вычислительные узлы, а вторая описывает, какие процессы участвуют в управлении кластером, и как они взаимодействуют друг с другом. Condor разделяет всю совокупность узлов на несколько типов:

- *Центральный менеджер*. Этот узел является ядром системы, собирая информацию о загрузке узлов кластера и получая запросы пользователей, он затем определяет время и место запуска. Все запросы обрабатываются на этой машине. Остановка менеджера приведет к остановке всей системы, поэтому на эту роль должен выбираться узел с хорошими аппаратными характеристиками и надежным сетевым соединением, поскольку менеджер общается со всеми компонентами системы.
- *Выполняющие машины*. Это узлы, на которых собственно происходит исполнение задания. Роль выполняющей машины может играть любой узел кластера, в том числе и узел с центральным менеджером. Вообще говоря, желательно, чтобы выполняющие машины имели высокую

производительность, поскольку это в конечном итоге определяет, когда пользователь получит результат. Однако это требование вовсе не является обязательным, Condor создавался с намерением использовать все доступные ресурсы, в том числе и низкопроизводительные, которые, однако, также способны решать некоторые задачи.

- *Запускающие машины*. Это узлы, с которых производится запуск пользовательских заданий. Во время работы задания на запускающей машине будут выполняться некоторые служебные процессы, поэтому желательно наличие достаточного количества ресурсов.
- *Сервер контрольных точек*. Эта роль является опциональной и при использовании Condor на Windows отсутствует. Назначение этого узла состоит в хранении всех контрольных точек пользовательских заданий, поэтому основными требованиями к этому узлу являются большой объем памяти и надежное сетевое соединение.

Каждый узел может выполнять несколько ролей одновременно. Перечислим далее основные сервисы (демоны) системы Condor. Стоит обратить внимание, что разработчики системы Condor тщательно продумали вопросы, связанные с надежностью системы, и включили несколько сервисов, позволяющих оперативно обрабатывать возникающие аппаратные и программные сбои.

- *condor_master*. Это ведущий сервис системы Condor, который ответственен за поддержку работы сервисов на всех остальных узлах. Он запускает другие процессы, устанавливает их обновления, перезапускает их при сбоях. Кроме того, он предоставляет некоторые возможности по удаленному управлению сервисами.
- *condor_startd*. Этот сервис представляет абстракцию вычислительного узла. Он отслеживает состояние узла и перед запуском заданий проверяет выполнение необходимых условий, например отсутствие пользователя.
- *condor_starter*. Этот процесс порождается condor_startd и предназначен для непосредственного запуска приложения. Кроме этого, этот процесс ответственен за установку окружения для пользовательского задания и контроля его выполнения.
- *condor_schedd*. Этот сервис хранит запросы ресурсов к пулу системы Condor. Запросы пользователей помещаются в очередь этого сервиса и остаются в ней до тех пор, пока необходимые ресурсы не освободятся. При освобождении ресурсов происходит порождение процесса condor_shadow.
- *condor_shadow*. Этот процесс является менеджером ресурсов пользовательского задания. Он существует на машине, с которой был произведен запуск, все время работы приложения. Так, если необходимо произвести удаленный системный вызов (файловый ввод-вывод, например), то его выполняет именно этот сервис.
- *condor_collector*. Этот сервис собирает и хранит всю информацию о состоянии узлов в пуле системы управления Condor.
- *condor_negotiator*. Этот сервис предназначен для поиска соответствий между пользовательскими запросами ресурсов и наличием таковых в пуле Condor. Периодически запускается акт «переговоров», во время которого сервис пытается найти соответствия, имеющиеся между пользовательскими запросами в condor_schedd и свободными узлами, информация о которых находится в condor_collector.
- *condor_had*. Это служебный сервис системы Condor, который контролирует правильность работы всей системы, отслеживая общение основных сервисов системы. В случае, если центральный менеджер по каким-то причинам прекращает работу, то сервис проверяет успешность его возобновления на другой машине.

5.2.5. Архитектура планировщика

За планирование запуска в Condor ответственны одновременно несколько сервисов: condor_schedd, condor_collector и condor_negotiator. Первый из них хранит очередь пользовательских заданий и ожидает освобождения ресурсов, чтобы запустить сервис condor_shadow. Второй хранит актуальную информацию о состоянии вычислительных узлов в пуле Condor. Собственно планирование осуществляет сервис condor_negotiator.

Рассмотрим подробнее алгоритм планирования. Создатели Condor при описании своего алгоритма используют метафору доски объявлений. Так, пользователи размещают на этой доске объявления о своих потребностях в ресурсах, а вычислительные узлы, в свою очередь, размещают информацию о доступных ресурсах. При этом все, что должен сделать планировщик – это найти удовлетворяющие друг друга задания и вычислительные узлы.

Все узлы периодически обновляют информацию о своем статусе на сервисе condor_collector. В этот статус входит такая информация, как объем свободного дискового пространства, объем свободной части оперативной памяти, загрузка центрального процессора и другие характеристики. Кроме динамических,

отсылаются также и статические показатели, такие как архитектура процессора, их число, операционная система и так далее. Совокупность этих характеристик формирует «предложение» пользовательским заданиям. Запросы пользователей, в свою очередь, попадают к condor_schedd, указывая свои потребности: архитектура, операционная система, потребность в оперативной памяти, ожидаемое время работы и много другое. Детальность этой информации задается пользователем, и чем меньше требований он выдвинет, тем большее число узлов смогут начать выполнение его задания.

5.3. Работа с системой

5.3.1. Установка и настройка

Condor поддерживает несколько различных операционных систем. Рассмотрим процесс установки и настройки этой системы управления кластером для операционной системы Windows. Как осуществляется установка системы на других платформах, можно узнать из документации на официальном сайте Condor [3].

Condor, так же как и CCS, поддерживает кластеры, состоящие из одного ведущего узла и одного или нескольких вычислительных узлов. Поэтому первое, что необходимо сделать – это выбрать ведущий узел, в терминологии Condor он называется центральным менеджером. К этому выбору следует отнестись очень ответственно, так как сбой в работе этого узла повлечет сбой в работе всей системы. Уже запущенные задачи при этом продолжат выполняться, однако запустить новое задание будет невозможно.

В первую очередь необходимо установить сервисы Condor на ведущий узел, а уже после этого на все остальные машины, входящие в пул. Для этого необходимо на каждом компьютере запустить инсталлятор, входящий в комплект поставки. При этом необходимо будет указать следующую информацию:

- Информация о пуле машин. В этом пункте необходимо выбрать, будет ли этот узел ведущим, либо он войдет в уже существующий пул. При создании нового пула указывается его имя, имя ведущего узла, размер пула.
- Информация о роли машины. Существует три роли: только добавление заданий в очередь, только выполнение заданий, их комбинация.
- Имя папки, в которую будет установлена система.
- Путь до Java Virtual Machine.
- Политика запуска задач. Имеется несколько вариантов: выполнять задание независимо от состояния машины; только если пользователь не работает за консолью (клавиатура или мышь) в течение последних 15 минут и загрузка CPU низкая (менее 30%); выполнять независимо от загрузки, но после 15 минут неактивности с консоли.
- Политика обработки задания после активации консоли. При активации консоли задание выгружается и через 5 минут продолжения активности Condor решает, что делать с частично выполненным заданием. Можно либо завершить выполнение задания, либо приостановить, не выгружая из памяти.
- Права доступа. По правам доступа все машины в пуле делятся на три категории:
- *Права на чтение (Read)*. При этом машине разрешается получать информацию о состоянии машин в пуле и очередей задач. Все машины в пуле должны обладать такими правами.
- *Права на запись (Write)*. Компьютер получает право посылать информацию сервисам Condor.
- *Права администратора (Administrator)* позволяют запускать/останавливать сервисы Condor. Такими правами должен обладать ведущий узел.

После установки необходимо на каждом узле запустить сервис Condor. Сделать это можно либо через окно «Сервисы», либо набрав в командной строке *net start condor*. После этого должны запуститься следующие сервисы:

- condor_master.exe
- condor_negotiator.exe (если это ведущий узел).
- condor_collector.exe (если это ведущий узел).
- condor_startd.exe (если Вы выбрали, что роль этого узла позволяет запускать задачи).
- condor_schedd.exe (если Вы выбрали, что роль этого узла позволяет добавлять задачи в очередь).

5.3.2. Работа с заданиями

При подготовке задания необходимо учесть, что программа будет выполняться в автоматическом режиме без вмешательства пользователя. Поэтому задание не должно содержать графический интерфейс и

интерактивный ввод/вывод. Кроме того, в некоторых случаях, которые мы опишем ниже, необходимо перекомпилировать пользовательское приложение с библиотеками Condor с помощью специальной команды компиляции. Кроме того, при подготовке нужно учесть, что Condor позволяет перенаправлять стандартные потоки ввода/вывода (stdin, stdout, stderr) в заданные файлы.

Процесс подготовки задания предусматривает описание системного окружения (Universe) системы. При этом можно использовать десять режимов работы: «Standard», «Vanilla», «PVM», «MPI», «Grid», «Java», «Scheduler», «Local», «Parallel» и «VM» (заметим, что режимы Standard и PVM недоступны для операционной системы Windows). Рассмотрим подробнее эти режимы:

Режим **«Standard»** обеспечивает максимально возможные средства миграции и надежности выполнения заданий (за счет использования механизма контрольных точек). Однако, при его использовании имеются некоторые ограничения на тип запускаемых программ (например, нельзя чтобы задание состояло из нескольких процессов, запрещены некоторые системные вызовы, нельзя открывать файлы одновременно на чтение и запись). Для использования этого режима необходимо пересобрать задание с помощью утилиты condor_compile (ее параметры аналогичны gcc, например, condor_compile cc main.o tools.o -o example).

«Vanilla». Данный режим предназначен для случаев, когда нет возможности пересобрать приложение с библиотеками Condor (например, по причине отсутствия исходных кодов). При этом использование контрольных точек и удаленных системных вызовов становится невозможным. В случае сбоя или выключения выполняющей машины задание будет либо ожидать возможности продолжения выполнения, либо будет перезапущено на другом компьютере. Для передачи входных и выходных файлов необходима одна из файловых систем: NFS или AFS.

Режим **«PVM»** предназначен для программ, использующих интерфейс Parallel Virtual Machine, а «MPI» нужен для программ, работающих с использованием библиотеки MPICH. В последних версиях Condor на смену **«MPI»** приходит новый режим **«Parallel»**, обеспечивающий более удобную работу с заданиями, использующими MPI.

Режим **«Grid»** позволяет перемещать задания из очереди задач одного компьютера в очередь задач другого, возможно удаленного. С помощью режима **«Java»** можно запускать Java-приложения, при этом Condor берет на себя такие действия, как поиск бинарных файлов JVM и установка путей до пакетов.

Режим **«Scheduler»** позволяет добавить небольшое задание в очередь. При этом оно будет немедленно запущено на исполнение. Подобную задачу нельзя запустить с удаленной машины, и она не может быть вытеснена другими задачами. Похожий режим **«Local»** позволяет немедленно запустить задание на машине, где оно было добавлено в очередь. При этом требования задачи к аппаратному обеспечению игнорируются. Наконец, последний режим **«VM»** позволяет запускать задания на виртуальных машинах VMware и Xen.

5.3.3. Командный интерфейс

Для запуска задания необходимо создать текстовый файл определенного формата, содержащий информацию о входных и выходных файлах, аргументах командной строки, требованиях задачи к аппаратному и программному обеспечению. Рассмотрим пример такого файла:

```
# comments
universe   = vanilla
executable = example.exe
output     = output.txt
queue
```

Строки, начинающиеся с символа ‘#’, являются комментариями. В нашем примере, мы хотим запустить приложение с именем **example.exe** в режиме «Vanilla», перенаправив стандартный вывод в файл **output.txt.** Чтобы добавить наше задание в очередь, необходимо выполнить команду **`condor_submit`**, единственным аргументом которой является имя конфигурационного файла. В процессе выполнения задания пользователь может наблюдать за ходом работы задания с помощью команд **`condor_q`** и **`condor_status`**. Кроме того, Condor может информировать обо всех событиях, произошедших с задачей, по электронной почте, либо записывать все события в файл в определенном формате. Также Condor поддерживает возможность наглядного графического отображения процесса выполнения задания с помощью компоненты Condor JobMonitor Viewer.

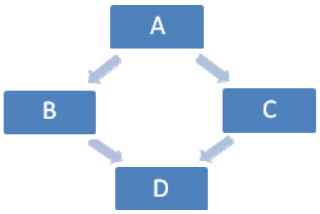
Condor поддерживает возможность запуска параметрического множества задач. Для этого используются параметризованные значения в конфигурационном файле:

```
Universe   = vanilla
Executable = example.exe
Input      = input.txt
Output     = output.txt
Error      = error.txt
```

```
Log        = example.log
InitialDir = run_$(Process)
Queue 600
```

Данный файл описывает задачу, состоящую в многократном запуске приложения **example.exe**. При этом стандартные потоки ввода/вывода перенаправляются в соответствующие файлы, все события, произошедшие с заданием, записываются в файл **example.log**. Подразумевается, что в текущем каталоге существуют подкаталоги с именами «run_1», «run_2», …, «run_600», в каждом из которых лежит файл **input.txt**, с соответствующими входными данными. Для удаления запущенной задачи необходимо воспользоваться командой **`condor_rm`**.

Кроме того, в системе Condor можно указать зависимости между заданиями. Для этого используются ациклические графы. При этом каждое задание представляется вершиной графа, а направленное ребро из вершины *A* в вершину *B* означает следующую зависимость между этими вершинами: «Не запускать задачу *B*, пока не завершится задача *A*». Каждый такой граф задается конфигурационным файлом (.dag). Рассмотрим эту возможность на примере. Пусть граф зависимостей имеет следующую структуру:



Тогда конфигурационный файл будет иметь следующий вид:

```
# example.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```

При этом каждой задаче соответствует свой конфигурационный файл (например, задаче A соответствует файл конфигурации **a.sub**).

Наконец, рассмотрим, как выглядит конфигурационный файл для параллельной задачи. Для этого воспользуемся режимом «MPI»:

```
universe = MPI
executable = mpi_program
machine_count = 4
queue
```

В простейшем случае необходимо лишь указать количество процессов (или количество машин, подразумевается, что на каждой машине запускается один процесс).

5.3.4. Интерфейс прикладного программирования

Condor поддерживает несколько различных интерфейсов прикладного программирования:

- Condor's Web Service (WS) API – интерфейс основанный на обмене сообщениями на основе протокола SOAP;
- DRMAA API (Distributed Resource Management Application API) – интерфейс, разработанный рабочей группой GGF (Global Grid Forum);
- GAHP – низкоуровневый протокол, основанный на передаче ASCII сообщений через потоки стандартного ввода и вывода;
- Condor Perl Module – библиотека языка Perl, содержащая необходимые функции для работы с Condor;
- Checkpoint Library API – библиотека, позволяющая использовать реализацию механизма контрольных точек вне системы Condor.

Condor's WS API является наиболее универсальным способом использования возможностей системы управления. При этом выбор веб-сервиса в качестве механизма взаимодействия обеспечивает большую свободу выбора языка программирования и платформы, на которой будет вестись разработка.

В рамках этого подхода задачи объединяются в кластеры, поэтому типичная последовательность вызова функций имеет следующий вид: создание кластера, создание задания, помещение задачи в очередь.

В этом интерфейсе в настоящее время доступно двадцать методов, которые можно разделить на четыре части:

- Служебные;
- Постановка задачи в очередь на выполнение;
- Управление заданиями;
- Работа с файлами.

Несмотря на их, казалось бы, небольшое количество, эти методы позволяют реализовать большую часть необходимых действий по управлению кластером, а наличие других программных интерфейсов покрывает всю оставшуюся функциональность.

6. Заключение

Системы управления кластером берут свое начало от систем пакетной обработки, появившихся в конце семидесятых годов. Главной целью и критерием для подобных систем является пропускная способность, то есть количество выполненных задач в единицу времени. При этом большое внимание уделяется равномерности загрузки вычислительных узлов. Системам управления кластерами, как их наследникам, присущи те же черты. Их основная задача – интеграция вычислительных ресурсов, объединенных локальной сетью, в единый многопроцессорный комплекс, который используется для пакетной обработки вычислительно трудоемких задач. При этом пользователь имеет возможность помещать свои задачи в общую очередь, использовать единый интерфейс для запуска, остановки, изменения и получения информации о текущем состоянии заданий. Система автоматически осуществляет планирование и запуск заданий, отслеживает их состояние и после завершения выполнения задачи информирует об этом пользователя и возвращает ему результаты.

Можно выделить типичный набор возможностей, который присутствует практически во всех системах управления: механизмы мониторинга загрузки вычислительных ресурсов, планирования запуска пользовательских задач, создание контрольных точек исполняющихся процессов, перезапуск и миграция заданий. Условно все системы управления распределенными вычислительными ресурсами можно разделить на два типа: классические системы управления и метакомпьютинг-системы. Представители первого класса предназначены для управления кластерами, представляющих собой объединенные в массивы специализированные высокопроизводительные сервера, единственным назначением которых является решение вычислительно трудоемких задач. Системы же второго типа строятся на основе рабочих станций «с хозяином», то есть вычислительных узлов, основная задача которых отлична от работы в режиме пакетной обработки заданий. Это могут быть домашние компьютеры, либо рабочие станции в терминал-классах образовательных учреждений.

Широко известными представителями классических систем являются Portable Batch System, Microsoft Compute Cluster Server 2003 и Sun Grid Engine. Примером метакомпьютинг-системы может служить система Condor. Метакомпьютинг-системы являются промежуточным звеном между системами пакетной обработки и GRID-системами, позволяя, например, выполнять задания лишь в то время, когда компьютер бы простаивал.

Бурное развитие технологии GRID привело к тому, что очень многие системы управления кластерами приобрели интерфейс взаимодействия с GRID-системами (например, с помощью Globus Toolkit). Метакомпьютинг и грид-системы обычно организуют на основе компонент, которыми являются как простейшие персональные компьютеры, так и мощные суперкомпьютерные системы, объединенные посредством локальных или глобальных сетей. При этом кластеры и суперкомпьютеры, входящие в такие системы, работают под управлением одной из систем управления кластерами. Таким образом, последние помимо задач владельцев кластера могут принимать задачи от пользователей со всего мира.

Итак, системы управления кластерами решают очень важную задачу – планирование и осуществление массовых запусков вычислительно трудоемких задач. От эффективности подобных систем часто зависит эффективность работы целых организаций и промышленных предприятий. То, как быстро происходит решение вычислительных задач, с какой скоростью протекают информационные процессы предприятия в целом, определяет, насколько успешным оно будет, насколько быстро оно сможет отвечать потребностям времени.

В этой связи особое внимание следует обратить на правильный выбор системы управления, благо сейчас есть из чего выбирать: от бесплатных систем с открытым исходным кодом до дорогостоящих коммерческих реализаций, обладающий широкой функциональностью и отличной технической поддержкой.

Как мы уже заметили выше, набор возможностей систем управления кластерами, представленных сейчас на рынке, примерно одинаков. При этом свободно распространяемые системы иногда могут даже

опережать по своим возможностям коммерческие аналоги. И в настоящее время ситуация такова, что на основе свободно распространяемого программного обеспечения можно сконструировать собственную высокоэффективную систему управления кластером, развернув и настроив все необходимое для параллельных вычислений окружение. Совершенно свободно доступны, например, компоненты мониторинга производительности вычислительных ресурсов (Ganglia, Mon, MRTG, Big Brother), средства и утилиты администрирования кластера (C3, WebMin), планировщики (Maui scheduler). Существуют также бесплатные реализации стандартов параллельного программирования, такие как MPICH. Часто путь самостоятельной разработки избирают команды исследователей в области параллельных вычислений, индивидуальные исследователи или небольшие коммерческие организации.

В свою очередь коммерческие реализации систем управления часто распространяются вместе с дополнительным программным обеспечением, позволяющим ускорить и облегчить подготовку заданий и их запуск на кластере. Это могут быть: среда разработки параллельных приложений, оптимизирующий компилятор, параллельный отладчик, собственная реализация стандарта параллельного программирования и так далее. Что немаловажно, компании-производители коммерческих систем предоставляют услуги по технической поддержке, что сильно облегчает установку, настройку и использование кластера. Таким образом, если кластер приобретается для обеспечения каких-либо бизнес-процессов, то коммерческие системы управления – это оптимальный выбор.

7. Литература

1. T. Sterling. Beowulf Cluster Computing with Windows // MIT Press, Cambridge, MA,2001.
2. Д. Владимиров. Кластерная система Condor. // Открытые системы, 07-08, 2000.
3. www.cs.wisc.edu/condor
4. M.A. Baker, G.C. Fox, and H.W. Yau. Review of Cluster Management Software. NHSE Review, May 1996. http://www.nhse.org/NHSEreview/CMS/.
5. http://openmosix.sourceforge.net/
6. http://www.kerrighed.org/
7. http://openssi.org/
8. http://www.gluster.org/
9. http://www.openpbs.org/
10. http://www.microsoft.com/windowsserver2003/ccs/default.aspx
11. Чарли Рассел «Обзор Microsoft Compute Cluster Server 2003». https://msdb.ru/Downloads/WindowsServer2003/CCS/CCS2003_Overview_Rus.pdf
12. «Microsoft Compute Cluster Server 2003 руководство рецензента». https://msdb.ru/Downloads/WindowsServer2003/CCS/CCS2003_Guide_Rus.pdf
13. http://msdn2.microsoft.com/en-us/library/aa155256.aspx
14. http://www.microsoft.com/rus/windowsserver2003/ccs/faq.mspx
15. http://technet2.microsoft.com/windowsserver/en/library/e544d9e2-e1db-447c-8a31-6bd0c30d809c1033.mspx?mfr=true
16. http://www.microsoft.com/technet/scriptcenter/hubs/ccs.mspx
17. http://msdn2.microsoft.com/en-us/library/bb540429.aspx
18. http://www-unix.mcs.anl.gov/mpi/
19. http://www.csm.ornl.gov/pvm/
20. http://www.vcpc.univie.ac.at/information/mirror/HPFF/
21. http://www.platform.com/Products/Platform.LSF.Family/
22. http://www.cs.wisc.edu/condor/condor-public-license
23. http://www-unix.mcs.anl.gov/metaneos/nug30/run.html