Московский государственный университет им. М.В.Ломоносова
Научно-исследовательский вычислительный центр

Воеводин Вл.В.
Жуматий С.А.

Вычислительное дело и кластерные системы

Советы и рекомендации по планированию, проектированию и использованию кластеров в вычислительной практике

Электронный вариант книги размещен в сети Интернет по адресу
http://ClusterBook.Parallel.ru

Издательство Московского университета
2007

Оглавление

ПРИТЧА
Раз при Горохе, при царе,
его три сына, опоясав сабли,
пошли в поход. Но во дворе
сын старший наступил на грабли.
И средний сын ступил ногой
на грабли, выразившись всяко.
Тут пригорюнился меньшой.
Да, делать нечего, однако…

А.Ману

Нет, нет и еще раз нет. Мы и сами не хотим, и Вам не желаем оказаться на месте младшего брата. Поместив данную притчу в самое начало книги, мы возложили на нее роль, не совсем характерную для эпиграфа. Это не основная мысль данной работы. Это главная проблема, с которой нам приходится сталкиваться на практике и на которой нам хотелось лишний раз заострить внимание. При всей, казалось бы, очевидности предстоящего шага не нужно слепо следовать сложившимся стереотипам действий или же автоматически перенимать уже готовые решения. Быстро – это не всегда хорошо. Особенно в столь доступной, но динамичной, высокотехнологичной и тонкой области, как проектирование и использование кластерных вычислительных систем.

Появление вычислительных кластеров позволило значительно расширить масштаб применения высокопроизводительных компьютеров. Многие инженерные и технологические процессы в нефтегазовом секторе

и машиностроении, фармацевтике и банковском деле, энергетике и телекоме, в биоинженерии и нанотехнологиях теперь существенно опираются на использование кластерных систем. То, что раньше можно было решить только на дорогих, а потому для многих практически недоступных суперкомпьютерах, сегодня можно сделать с помощью недорогих и эффективных кластерных систем. Массовое использование параллельных вычислительных технологий, чему первый основной толчок дали именно кластерные системы, стало характерной чертой нашего времени.

Приведенная в эпиграфе притча очень точно характеризует подход многих организаций к построению высокопроизводительных кластерных систем. Дело новое, но перспективное, опыта не много, но кластерные системы нужны, поэтому копирование архитектуры какого-либо уже существующего кластера рассматривается как вполне допустимый способ развития собственного проекта. Во многих случаях это оправдано, однако столь же часто использование чужого пути приводит к повторению одних и тех же ошибок. При этом все говорят об успехах, но почти никто не рассказывает о неудачах или же оказавшихся ошибочными на практике решениях, хотя именно на них нужно бы обратить внимание новичков кластерного дела, только приступающих к работам в данной области.

В данной книге мы попытались собрать вместе все те вопросы, которые нужно обдумать и решить, начиная с момента появления потребности в кластерной системе для решения конкретных задач, до проектирования, заказа, монтажа, настройки и тестирования кластера, чтобы в процессе будущей эксплуатации **получить действительно эффективное по всем параметрам кластерное решение**. При этом, книга создавалась не для того, чтобы предоставить набор готовых рецептов разрешения сложных ситуаций, хотя во многих случаях делается и это. Ставились иные цели. Прежде всего, мы хотим явно указать на те потенциальные проблемы, которые могут возникнуть на различных этапах работы с кластерными системами. Накопленный за многие годы опыт

подсказывает, что список проблем немалый. Не в каждом проекте все подобные проблемы могут проявиться, у всех коллективов и проекты, и условия их реализации в чем-то различаются, однако “предупрежден – значит вооружен”. Одновременно с этой задачей, мы хотим обратить внимание читателя и на многообразие возможных решений. Технологии быстро развиваются, привычные подходы нужно вовремя модифицировать или же просто переходить на новые методы работы. Кластерные технологии дают в руки мощный инструмент построения вычислительных систем, владение которым и понимание которого приводит к исключительно эффективным решениям.

И, наконец, самое важное. В вычислительном деле с задач все начинается, задачами определяется и по результатам их решения оценивается. Об этом нередко забывают, но это именно та идея, которая должна составлять фундамент любого кластерного проекта, иметь максимальный приоритет при принятии решений на каждом этапе создания кластера. Никто не покупает просто автомашину: разница в назначении представительского седана, карьерного самосвала, минивэна и бюджетного варианта народного автомобиля всем очевидна. Вместе с тем, задачи каждого класса машин полностью определяют особенности их компоновки, проектирования, технологического цикла производства и эксплуатации. В автомире это воспринимается как норма. Так должно быть и в мире кластерных систем. В этом основная цель данной книги.

Следует специально отметить, что, говоря о кластерах, мы имеем в виду высокопроизводительные вычислительные системы. Сам термин “кластер” в последнее время широко используется во многих областях: кластер высокой степени готовности, кластер для web-приложений, однако нам важна суперкомпьютерная, вычислительная сторона дела, которой и будет уделено основное внимание.

Представленный материал отражает как наш собственный опыт выполнения большого числа кластерных проектов, так и опыт, накопленный в кластерной отрасли в целом. Безусловно, много полезного

мы получили от общения с нашими коллегами по лаборатории параллельных информационных технологий Научно-исследовательского вычислительного центра МГУ имени М.В.Ломоносова Антонова Александра и Стефанова Константина, за что выражаем им нашу искреннюю признательность. Мы исключительно благодарны сотрудникам корпорации Intel: Самофалову Виктору, Семину Андрею и Нарайкину Андрею за регулярно предоставляемую возможность работы на новых образцах вычислительной техники, за ценные советы, доброжелательность и внимательное отношение к нашей работе. Мы хотели бы поблагодарить корпорацию Intel, при поддержке которой данная книга была создана и вышла в свет. Наша особая благодарность Валентину Васильевичу Воеводину, учителю в науке и наставнику по жизни, общение и работа с которым всегда подталкивали к исследованию нового, приводили к интересным постановкам задач, к нетривиальным и неожиданным решениям. Спасибо.

Отдельные разделы книги написаны кратко, но дополнительный материал, последние новости и множество технических подробностей доступны на страницах информационно-аналитического центра по параллельным вычислениям в сети Интернет Parallel.ru. Там же по адресу `http://ClusterBook.Parallel.ru` размещен электронный вариант данной книги. В дискуссионном клубе центра можно обсудить возникшие вопросы или высказать предложения по тем направлениям, которые имело бы смысл раскрыть подробнее в последующих редакциях этой работы. Свои комментарии можно направить авторам и напрямую по электронной почте `ClusterBook@Parallel.ru`. Мы будем признательны, если Вы поделитесь с нами своим личным опытом создания и использования кластерных систем, сложными вопросами и нестандартными ситуациями, возникшими по ходу реализации проектов, найденными вариантами их решения.

Успехов Вам и только удачных кластерных решений!

О прикладных задачах и кластерных системах

Компьютерный мир меняется. Пятидесятилетняя эпоха последовательной организации вычислений сменяется эрой параллелизма, параллельных вычислительных технологий и параллельных вычислительных систем. Сначала появление кластерных систем, а затем переход всех ведущих производителей вычислительной техники на многоядерные процессоры, сделали компьютерный мир параллельным.

Революционность происходящего в наше время сродни изменениям, которые потребовались обществу в середине прошлого века для адекватного восприятия первых компьютеров. Сегодня задача не менее сложная, поскольку необходимо адаптироваться к параллельным компьютерным технологиям. Нужно пересматривать подходы к исследованию свойств и анализу качества алгоритмов, необходимо менять мышление программистов, нужен целый спектр специального инструментария для разработки программного обеспечения нового поколения.

Доступность современных параллельных компьютерных систем создает иллюзию легкости их использования. Многопроцессорные серверы, кластеры, многоядерные процессоры, распределенные вычислительные среды действительно можно использовать для решения любых задач, однако доступно – не значит просто. В параллельных вычислительных системах на передний план выходит понятие, на которое не так сильно обращали внимание в период господства последовательных методов организации вычислительного дела: эффективность использования компьютеров. В самом деле, а почему бы не сохранить традиционные последовательные технологии программирования? Берем существующую и уже работающую программу, из всего множества доступных процессоров, ядер или вычислительных узлов параллельного компьютера задействуем

только один экземпляр, компилируем, запускаем и получаем результат. Ничего сложного, все отработано годами, все привычно и никаких дополнительных проблем или неудобств эти шаги не вызывают. В принципе, все это верно, за исключением одного – забыто главное. Зачем создаются параллельные компьютеры? Для увеличения их производительности. Именно это дает возможность уменьшить время работы программ, перейти к новым размерам и размерностям в постановке задач, повысить точность их решения. И все это останется в стороне, если забыть про параллелизм и по старинке ориентироваться только на последовательные методы работы. Выполнить программу на одном процессоре, безусловно, можно, и человек даже сможет сказать, что он “работал на многопроцессорной системе”. Только какой в этом смысл? Нужно ли было тратить значительные средства на установку дорогостоящей системы, если эффект от нее такой же, как от обычного компьютера? Вряд ли. Целесообразно ли использовать параллельную вычислительную систему в таком режиме? Скорее всего – нет. Инструмент определяет методы и технологии его использования. Если в архитектуре компьютера присутствует параллелизм, значит он в том или ином виде должен быть и в программе. В противном случае, ответ на вопрос: “Зачем использовали параллельный компьютер?”, нужно искать в какой-то совсем иной плоскости.

Все сказанное в полной мере относится и к кластерным системам. Появившись в середине 90-х годов прошлого века как доступная альтернатива традиционным суперкомпьютерам, они показали себя достойным средством решения больших задач. Необычайно быстрый рост популярности кластеров в вычислительном сообществе объясняется двумя причинами: очень высоким показателем отношения пиковой производительности к стоимости и большой свободой при выборе конфигурации кластерной системы. Сегодня за весьма умеренную цену организация может установить систему значительной производительности, причем в той конфигурации, которая лучше всего подходит для решения ее

задач. Подобная ситуация сама по себе не может не радовать, однако нельзя забывать, что кластер – это инструмент, это одно звено в длинной цепочке решения задач на параллельных вычислительных системах. Звено, безусловно, важное, но не единственное. Создавая методы решения задач на параллельных компьютерах, мы обязаны рассматривать всю цепочку в целом: от постановки задачи, описания алгоритма, выбора технологий параллельного программирования, до вопросов организации программно-аппаратной среды самой вычислительной системы и ее инженерной инфраструктуры. Слабость любого одного звена может привести к резкому падению эффективности решения задачи в целом, и только согласованность решений, принятых на всех этапах, даст надежду на действительно хороший результат.

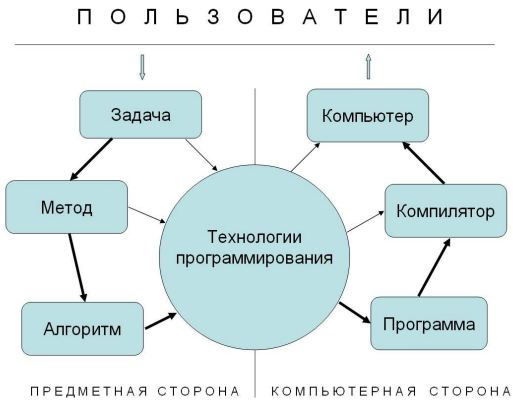

Рис. 1. Этапы решения задач на компьютерных системах

Обратимся к схеме решения задач (рис. 1) с помощью различного рода компьютерных систем. Стартовой точкой является задача – это то, что мы хотим найти, узнать или вычислить с помощью компьютера, это наша цель. Для решения задачи выбирается некоторый метод, т.е. математически обоснованная последовательность действий, следуя которой мы решение

получим. Под действием может пониматься как элементарная арифметическая операция, так и что-то более сложное, но не столь важное с точки зрения самого метода, например, вычисление скалярного произведения или нахождение максимального значения в заданном наборе чисел. Зафиксировав метод, следующим шагом необходимо описать алгоритм. Для этого нужно указать точный набор и порядок выполнения всех операций. Обычно, на данном этапе не очень задерживаются, так как выбранный метод автоматически определяет идею алгоритма. Но только идею. На практике, два алгоритма, построенные по одному и тому же методу, могут обладать принципиально разными свойствами, в частности, один может оказаться чисто последовательным, а другой алгоритм будет обладать значительным ресурсом параллелизма [1]. Двигаясь по цепочке далее, алгоритм записывается в виде программы с помощью той или иной технологии программирования, затем полученная программа обрабатывается компилятором, который уже и генерирует код, пригодный для исполнения компьютером. “Задача”, “Метод” и “Алгоритм” в большей степени относятся к предметной области, “Программа”, “Компилятор” и “Компьютер” составляют основу последующей реализации, а “Технологии программирования” – это посредник между двумя сторонами единого процесса решения задачи.

Такова стандартная цепочка, и на рисунке она отмечена жирными стрелками. Ее много раз проходил каждый программист, как системный, так и прикладной, выбирая оптимальный путь решения своей задачи. За счет использования специальных технологий программирования цепочка в ряде случаев может стать короче, позволяя пользователю найти решение его задачи проще и быстрее. В самом деле, воспользовавшись уже готовым прикладным программным комплексом, все решение можно свести лишь к правильному заданию входных данных, и путь от задачи через такие технологии программирования сразу приведет к компьютеру. Подобных “альтернативных” путей существует немало, и на рисунке они показаны тонкими стрелками.

Данная схема дает богатую пищу для размышлений и помогает понять причины многих проблем, возникающих при решении задач на компьютерах, особенно на компьютерах с параллельной архитектурой. Тема достойна отдельного обсуждения, однако сейчас сосредоточимся на этапе, который отвечает предмету данной книги – на кластерных системах. Это последнее звено цепочки, обозначенное на рисунке словом “Компьютер”, отвечающее за выполнение программ на кластере. Можно указать по меньшей мере три причины, почему данное звено занимает в цепочке особое положение и заслуживает особого внимания:

- тот факт, что программа будет работать на кластере, должен учитываться на всех предыдущих стадиях: это скажется на выборе метода и алгоритма решения задачи, технологий и систем программирования, т.е. на всех звеньях цепи;
- по мере прохождения этапов накопятся и именно здесь скажутся недостатки всех принятых ранее решений: не лучшие свойства метода, множество последовательных частей в алгоритме, плохо написанная программа, огрехи компилятора, и в результате – низкая эффективность работы параллельной программы;
- наконец, если не обеспечить качественного функционирования кластерной системы, лежащей в основе последнего звена цепочки, то станет просто бессмысленной вся работа, которая была выполнена на всех предыдущих этапах.

Приступая к проектированию кластерной системы, очень важно продумать всю цепочку в целом, чтобы сформировать будущее решение на основе целенаправленного выбора и анализа, а не ощущений или эмоций от рекламных кампаний. Общепринятые стандартные подходы хороши “в среднем”, но для конкретных задач они могут оказаться совершенно непригодными. Если заказчик просит спроектировать кластерную систему, но ничего не говорит о предполагаемом классе задач или же режиме использования системы, то это должно серьезно насторожить. В зависимости от того, создается ли центр коллективного пользования с

большим числом разных задач, учебный центр по параллельным вычислениям или вычислитель для поддержки конкретного проекта, в зависимости от того, насколько критичным является требование надежности и отказоустойчивости работы оборудования, сколь тесно кластер должен быть в будущем интегрирован в информационно-технологическую инфраструктуру предприятия – при каждом сценарии развития кластерного проекта нужно делать свой акцент в архитектуре комплекса. В противном случае, пострадает качество решения задачи, ради которой комплекс и создавался.

Подтверждений этому тезису в нашей практике очень много, и чем раньше удавалось уточнить требования и определить структуру задач, тем лучше получался результат. В одном проекте заказчик сразу же начал с того, что попросил проанализировать эффективность выполнения его программы на различных процессорах. Желание вполне обоснованное, если учесть, что кластерная система проектировалась только под одну конкретную задачу, и об универсальности установки речь не шла. Модельная программа была предоставлена, и нам оставалось лишь исследовать доступные серверные платформы, на основе чего и выбрать вариант построения будущего кластера. Был найден хороший вариант связки платформы и компилятора, дающей наименьшее время выполнения программы на различных наборах входных данных. Вроде бы задача выполнена, и можно было приступать к проектированию кластера в целом, однако настораживало немного необычное поведение модельной программы. Во время испытаний на каждой платформе программа занимала всю доступную оперативную память. Поговорили с заказчиком, и оказалось, что объем оперативной памяти во многом определяет и размер задачи, и время ее решения. Заказчик держал в голове “средние” параметры существующих кластерных систем, и минимизировал время работы программы за счет выбора подходящего процессора. По сравнению с предварительным проектом в окончательном варианте конфигурации кластера число вычислительных узлов было уменьшено в два раза, но зато

объем оперативной памяти на каждом узле был увеличен в четыре раза, что привело к значительному увеличению скорости решения исходной задачи. К слову, в окончательной конфигурации пришлось отказаться и от первоначально найденной “самой лучшей” платформы, которая не поддерживала эффективной работы с большими объемами оперативной памяти.

В другом проекте в научной группе планировался перевод большого программного комплекса, традиционно работавшего в однопроцессорном режиме, на параллельную кластерную систему. Нужно было переходить к новым размерам задач, но поскольку в существующем варианте время расчета было чрезмерно большим, то только переход на параллельную систему в перспективе мог дать необходимое ускорение работы программы. Составили проект, показали нам. Серьезных ошибок технического характера в проекте не было, а сказать что-либо еще без знания структуры будущих задач невозможно. Стали разбираться с программным комплексом и обнаружили потенциально серьезную проблему: интенсивная работа с файлами. При работе комплекса на одном процессоре этот вопрос и не возникал, так как обмен с дисками занимал около 1% всего времени работы программы. Однако переход на кластер из 32 процессоров с перспективой увеличения их числа в будущем до 128 заставлял серьезно задуматься о параметрах и конфигурации подсистем ввода/вывода, поскольку файловые операции все больше и больше выходили на первый план. Кстати, одно из определений суперкомпьютера в точности описывает данный эпизод: “Суперкомпьютер – это вычислительная система, сводящая проблему вычислений к проблеме ввода/вывода”. Немного иронично, но тонко подмечено. Когда вычислительная часть перестает быть узким местом, работу программы начинает сдерживать что-то еще, в данном случае, это работа с устройствами ввода/вывода. Очень жизненная ситуация, характерная для многих проектов.

Теперь рассмотрим типичный пример использования кластерных

систем, когда вся содержательная работа ведется через специализированные прикладные пакеты: ANSYS, GAMESS, ABAQUS, STAR-CD и другие. Нет сомнений, подход правильный и обоснованный: если кто-то уже подумал и о решении задач предметной области, и о распараллеливании, то готовым инструментом нужно воспользоваться. Нужно лишь быть уверенным в качестве двух принципиально важных компонентов. Первый – это сам пакет. Безусловно, он прекрасно справляется с определенными классами задач. Но все реальные задачи разные, пакеты же, как правило, ориентированы на некоторый “усредненный” вариант, и требуется специальная проверка их работоспособности при критических значениях параметров: предельных нагрузках, максимальных скоростях, переходных процессах, приграничных областях. Есть ли уверенность, что планируемый к установке на кластере пакет даст решение задачи? Вопрос непростой, нужна аккуратная оценка математики, методов, точности, диапазона входных данных и допустимых параметров моделей, заложенных в пакете. Второй компонент – это эффективность работы связки пакет-кластер на типичных наборах входных данных. Хорошо ли пакет масштабируется по числу процессоров? Может ли пакет использовать полностью имеющуюся оперативную память? Какие параметры коммуникационной сети для него наиболее критичны? Известно много случаев, например [2], когда конфигурация кластеров менялась после предварительных испытаний различных вычислительных платформ на тестах пакетов с учетом специфики входных данных, характерных для задач пользователей. Эффективно ли пакет адаптирован под архитектуру кластера? Данный вопрос также не следует оставлять без внимания, поскольку разработчики пакета могли использовать не совсем стандартные схемы и модели для реализации вычислительного процесса. Например, известный пакет для квантово-химических расчетов Gaussian для выполнения всех операций взаимодействия параллельных процессов использует систему Linda, которая сама по себе предполагает их общение через общую память. Необходимо убедиться заранее, существуют ли

эффективные реализации данного пакета на системах, аналогичных проектируемому кластеру. Иногда пакеты строятся на основе модели вычислительного процесса типа Мастер/Рабочие. В этом случае вся расчетная часть ложится на процессы-рабочие, а распределение работы между ними и управление процессом вычислений в целом выполняется процессом-мастером. Запуская пакет на четырех процессорах, нужно учитывать, что реальный счет будет идти только на трех, что определит и результирующее ускорение выполнения программы. Более того, в некоторых прикладных пакетах на каждый расчетный процесс порождается по одному вспомогательному, поэтому эффективно использовать 32 процессора реально сможет только программа, состоящая из 64 параллельных процессов.

Подобных нюансов на практике очень много. Завершая вводную часть книги, хочется еще раз подчеркнуть следующее. Главное в вычислительном деле – это задачи, в конечном итоге именно они определяют все. И чтобы перенести усилия пользователей в сторону предметной, содержательной и наиболее важной части их деятельности, нужно быть абсолютно уверенным в качестве используемых ими инструментов, в эффективности и согласованности всех этапов, через которые пользователи должны будут пройти на пути от постановки задачи до получения ее решения. Все этапы важны, ни один нельзя оставлять без внимания, каждый должен быть аккуратно исследован. Данная книга – о базовом уровне вычислительной практики, о кластерных системах, о тонких местах их проектирования, использования, сопровождения, о важных деталях и одновременно обо всех тех незаметных мелочах, которые скрасят вычислительные будни положительными эмоциями от красиво реализованных решений.

Глава 1.
Нужен кластер. С чего начать?

Итак, будем предполагать, что к настоящему моменту появились две главные составляющие: есть задачи и желание использовать собственную кластерную вычислительную систему. Сформировалось четкое представление, для чего строится кластер, на основе какого бюджета, каких целей мы хотим достичь, какие стратегические задачи предприятия с его помощью нужно решить, какой класс вычислительных задач будет доминировать. Есть понимание того, планируется ли развитие кластерного проекта в будущем или же речь идет о разовой акции. Все эти вопросы важны и чем точнее удастся сформулировать ответы, тем эффективнее будет реализован проект в целом. Безусловно, поскольку Вы уже задумались о кластере, то какой-то вариант ответов наверняка есть, но важно от "какого-то" варианта перейти к четким формулировкам и попытаться отразить его на бумаге. Поверьте, нюансов здесь много, и стоит потратить день на размышления, чтобы потом иметь годы спокойной работы.

Поскольку речь пойдет о выборе системы с необходимым уровнем производительности, сразу начнем с неожиданного вопроса: "А нужен ли кластер в классическом виде вообще?". В настоящее время начинают выпускаться компактные, но исключительно производительные системы, сопровождение которых не сильно отличается от администрирования обычных систем. В конце 2006 года корпорация Tyan Computer (`http://www.tyan.com/`) объявила о выпуске "персонального суперкомпьютера" Tyan Typhoon T600, построенного на основе четырехядерных процессоров Intel Xeon серии 5300. Параметры системы: пиковая производительность 256 Гфлопс, энергопотребление 1400 Вт, низкий уровень шума, размеры обычной рабочей станции, и все это при

вполне доступной цене – есть над чем задуматься… Будем предполагать, что такие альтернативы уже рассматривались, но по некоторым причинам были отклонены и чаша весов склонилась в пользу кластерных решений.

Определившись с общими целями, выберите для себя оптимальный способ размещения и использования будущего кластера. Казалось бы, о чем тут размышлять: купил кластер, поставил и используй. Тем не менее, уже на этом этапе можно предложить несколько различных вариантов реализации проекта.

Установка кластера на своей территории. В настоящее время это самый распространенный вариант работы с кластерными системами. Его основное преимущество состоит в том, что кластер будет находиться полностью под Вашим контролем и управлением. Это важный аргумент, который часто перевешивает все остальные доводы. По такому пути, в частности, в 2005 году пошло научно-производственное объединение “Сатурн” (г. Рыбинск), разместившее у себя мощный кластер IBM из 128 процессоров Intel Xeon EM64T 3,6 ГГц вместе со всей связанной с ним информационно-вычислительной инфраструктурой.

Принимая решение разместить кластер у себя, продумайте, как будет решаться весь спектр смежных вопросов. Необходимость выделения специального помещения, установка климатического и защитного оборудования, дополнительная подводка электропитания, дополнительные затраты на электричество и инженерное обслуживание инфраструктуры, высококвалифицированные специалисты – эти и многие другие вопросы полностью лягут на Вас. Есть ли в Вашем офисном здании резерв по электропитанию 50 кВт, необходимый для работы кластера? Может ли компания расширить свой штатный состав, включив дополнительный персонал для сопровождения и администрирования новых систем? Если ответы не кажутся столь очевидными, подумайте об альтернативных способах размещения.

Установка кластера на чужой площадке предполагает, что принадлежащий Вам кластер будет физически располагаться на территории

другой организации. Вся работа с кластером будет проходить в удаленном режиме через сеть. Если внешняя сеть быстрая, то пользователи никакой разницы между этим и предыдущим вариантом не заметят. Преимущество второго подхода заключается в том, что нет необходимости в подготовке и обслуживании помещения собственными силами. Дополнительными аргументами такого варианта размещения часто служат более низкие тарифы на электроэнергию и ставки аренды помещений. Именно в таком режиме используется 152-процессорный кластер компании Paradigm Geophysical, расположенный на площадях центра кластерных технологий компании "Т-Платформы" (`http://www.t-platforms.ru/`).

Выбранная площадка, конечно же, должна удовлетворять всем необходимым требованиям. Она должна соответствующим образом быть подготовлена и оборудована, должен присутствовать подготовленный персонал как для обслуживания собственно кластерных систем, так и для поддержания нормальной работы климатических установок, систем электропитания, мониторинга, безопасности.

На выбранную площадку должен быть обеспечен доступ сотрудников для проведения на кластере профилактических и других регламентных работ. Практически все действия, связанные с установкой системного и прикладного программного обеспечения, современные технологии позволяют выполнять в удаленном режиме без необходимости явного присутствия рядом с компьютером. Однако следует иметь в виду, что при возникновении сложных проблем с аппаратурой, время их решения неизбежно возрастет в связи с необходимостью выезда на удаленную площадку.

Свои специалисты для сопровождения кластерных систем в удаленном режиме все равно потребуются, но степень их квалификации теперь будет напрямую зависеть от объема обязательств, взятых на себя специалистами удаленной площадки.

Еще одним распространенным способом организации работ на кластерных системах является **аренда вычислительных мощностей**

кластера другой организации. Этот вариант полностью избавляет от затрат не только на приобретение, но и на его обслуживание и администрирование. Недостаток этого подхода состоит в том, что становится сложнее контролировать и планировать процессорное время, а, следовательно, и время получения результатов.

По большому счету, весь спектр проблем решается в переговорах с владельцами арендуемой системы. Во многих случаях можно договориться о предоставлении “выделенного” времени, когда кластер или его часть отдаются в монопольное использование. Это стоит дороже и нужно отметить, что не всегда удается рационально использовать это время. Чаще всего договариваются о выделении некоторой квоты на суммарное процессорное время, которое могут использовать приложения арендатора.

Еще один существенный минус третьего варианта – ограниченность прикладного и системного программного обеспечения. Если для расчетов необходим специальный программный пакет или библиотека, то не факт, что он будет присутствовать на удаленном кластере, и далеко не всегда есть возможность его туда установить.

Немаловажным нюансом являются и те гарантии безопасности, которые дают хозяева предоставляемых ресурсов. Можно ли рассчитывать на сохранение коммерческой тайны при выполнении расчетов? Могут ли хозяева системы, работающей в режиме коллективного пользования, гарантировать невозможность доступа для всех других пользователей к размещаемым на кластере данным? Все эти вопросы нужно обязательно обговорить с владельцами кластерной системы заранее. Отчасти подобный спектр проблем касается и второго варианта размещения, поскольку доступ в помещение, где установлена кластерная система, не всегда может полностью контролироваться. Подумайте о дополнительных функциях систем круглосуточного автоматического мониторинга с возможностью разного рода оперативного оповещения и трансляции необходимых данных, в частности видеокартинки помещения, по сети.

Заметим, что последние два варианта использования кластерных

систем: размещение на чужой площадке и аренда времени (ресурсов) кластеров, в настоящее время становятся все более и более популярными. Во многих случаях для компаний оказывается экономически выгодным вывести непрофильные информационно-вычислительные подразделения из своего состава, воспользовавшись услугами сторонних организаций.

Проектируя кластерную систему, ни в коем случае нельзя замыкаться только на аппаратуре, иначе с бюджетом проекта могут быть большие проблемы. Обязательно продумайте, какое программное обеспечение будет использоваться на кластере для выполнения поставленных задач. А еще лучше с этого и начинать проект. Стоимость многих прикладных пакетов настолько значительна, что может легко превысить стоимость собственно кластерной системы. Существуют разные типы программного обеспечения, различные виды лицензий, разумный выбор которых позволит весьма аккуратно вписаться в имеющийся бюджет. Вопрос очень важный, и мы вернемся к его обсуждению позднее.

Правильный подход к делу предполагает отдельный **бюджет на обслуживание и содержание кластера**. По статистике в среднем около 5-7% стоимости кластера в год будет уходить на техническое обслуживание, проведение регламентных работ и замену неисправного оборудования. Плюс регулярное обновление лицензий на программное обеспечение. Плюс содержание обслуживающего персонала.

Не забывайте про расход электроэнергии, который для кластера может быть весьма существенным. При этом энергопотребление кластера будет заведомо выше потребления электроэнергии таким же числом обычных персональных компьютеров: вычислительные нагрузки выше, мощность используемого серверного оборудования почти всегда больше мощности персональных машин, кластерные системы почти всегда работают в круглосуточном режиме. Рассмотрим простой пример. Пусть кластер состоит из 10 узлов IBM eServer x306m, потребляющих 150 Вт каждый, такого же управляющего узла и коммутатора Cisco Catalyst 2960G-24TC (мощность 75 Вт). Суммарная потребляемая мощность составит:

150*10+150+75=1725 Вт. За 30 суток работы потребление составит 24*30*1,725=1242 кВт·ч. Нужно учесть, что кроме собственно кластера, электроэнергия будет потребляться климатическими системами и любым другим элементом кластерной инфраструктуры. Набегает немало. Энергопотребление кластера СКИФ Cyberia составляет 90 кВт, а всего суперкомпьютерного комплекса в целом – 115 кВт. Впечатляет, но не удивляет: все параметры рекордных по производительности систем намного выше среднего, а СКИФ Cyberia в начале 2007 года занял первую строчку в списке Top50 самых мощных компьютеров СНГ.

Взвесив все организационно-административные и технологические решения, **оцените стоимость владения кластером** и, возможно, скорректируйте параметры проекта. Поскольку стоимость владения является серьезной, весьма непростой и неоднозначной характеристикой, в процессе дальнейшего изложения мы еще будем к ней возвращаться.

Понимаем, что немного странно напоминать про очевидные факты, однако жизнь показывает, что напоминать нужно: **не откладывайте составление проекта кластерной системы и его реализацию на последний момент**, на конец финансового или календарного года. Это только кажется, что впереди еще много времени и все можно сделать завтра. Поверьте, непредвиденных задержек будет очень много. Неожиданно появятся другие неотложные дела, поставку компьютеров для предварительного тестирования задержат, оценить качество новых решений или проверить эффективность необходимого прикладного пакета для выбранной платформы не успеете, сам кластер, как обычно, потребуется уже завтра, а подобного рода системы у поставщиков редко бывают на складе готовыми к немедленной доставке и т.п. Цепочка потянется дальше, и в результате будет принято далеко не лучшее компромиссное решение между тем, что доступно, и тем, что на самом деле необходимо. Определитесь заранее хотя бы с основными опорными точками проекта, тогда последующая реализация пройдет без особых осложнений.

И последнее. При всей внешней доступности нельзя забывать, что кластер – это сложная, высокотехнологичная вычислительная система, требующая правильного сопровождения и грамотного использования. **Кадры, кадры, кадры…** Исключительно серьезный вопрос, и об этой проблеме нужно задуматься чуть ли не в первую очередь. Недостающее оборудование можно быстро купить, а квалифицированного специалиста нужно готовить в течение длительного времени. Дополнительная сложность заключается в том, что параллельные вычислительные технологии долго оставались в тени ставших за 50 лет привычными последовательных методов работы с компьютерами. Ситуация меняется, но пройдут годы, пока создание параллельных программ станет нормой программистского мира. А пока как администраторов, так и прикладных специалистов нужно готовить к новой технике, к новым методам программирования, к новым понятиям и технологиям. И их нужно подготовить до того, как кластер будет поставлен.

Глава 2.
Базовая инфраструктура будущего проекта

Если принято решение размещать кластер на чужой площадке или воспользоваться уже работающей кластерной системой, то этот раздел можно пропустить. В данном разделе мы обсудим основные требования к инженерной инфраструктуре кластера: помещению и его оборудованию, системам охлаждения, электропитания и другим не менее важным ее составляющим.

Итак, решено установить кластер на своей территории. В этом случае потребуется продумать и составить систему подготовительных мероприятий, а также выполнить расчеты необходимых параметров инфраструктуры. Если нужных данных в распоряжении не оказалось, от расчетов не отказывайтесь, а просто отложите их на некоторое время. Не исключено, что к расчетам придется вернуться не один раз, если не удастся сразу, например, подобрать помещение с достаточной площадью или же обеспечить подводку электропитания нужной мощности.

Настоятельно рекомендуем **выделить для кластера отдельное помещение**, изолированное от рабочих мест пользователей. В нашей практике не было ни одного случая, когда размещение кластера в одной комнате с людьми было бы действительно обоснованным и оправданным. Причин этому несколько.

- Даже кластер начального уровня из 2-3 узлов может производить шум, при котором трудно разговаривать, не повышая голоса. Например, уровень шума сервера HP DL145 (1U, 2 процессора AMD Opteron) составляет 63,2 децибел, что сравнимо с шумом хорошо заполненного ресторана. Если поставить 10 таких серверов, то уровень шума будет около 70 децибел, что сопоставимо с нахождением в пяти метрах от оживленной автотрассы.

- Скорее всего, для кластера потребуется кондиционирование, а его правильно организовать и контролировать можно только в отдельном помещении.
- Для своего охлаждения вычислительные узлы протягивают сквозь корпус внушительный поток воздуха, а вместе с ним – пыль и грязь. Отдельное помещение, где не будет открываться окно, а доступ посторонних будет ограничен, гарантирует, что количество пыли, попадающее в узлы, будет минимальным.
- Физический доступ к кластеру желательно ограничить небольшим кругом лиц, подготовленных и сертифицированных соответствующим образом. Имея отдельное помещение, сделать это будет намного проще.

Заранее подумайте о том, **какое место в помещении займет кластер**. Не забудьте о расстояниях до стен и вентиляции, возможности подойти к стойкам со всех сторон, открыть их двери, о рациональном подводе электропитания.

Чем больше **объем помещения**, тем лучше условия для нормального функционирования кластера, поскольку риск резкого повышения температуры из-за отключения систем кондиционирования меньше. Если есть мониторинг температуры, то в случае возникновения аварийной ситуации будет достаточно времени для автоматического выключения кластера до того момента, пока температура в помещении поднимется выше критической отметки. С этой точки зрения, при прочих равных условиях под кластер лучше выделить комнату с высокими потолками.

Для кластера потребуется оперативный набор запасных частей; к узлам, к системному и прикладному программному обеспечению будут прилагаться многочисленные диски, документация и гарантийные листы – все это необходимо где-то хранить. Ящика стола и даже одной полки в шкафу, скорее всего, будет мало, поэтому желательно предусмотреть **специальное место для хранения вспомогательных материалов**.

Как и любой электроприбор, кластер и связанная с ним аппаратура выделяют тепло, поэтому проведите расчет мощности необходимой **системы охлаждения воздуха**. Не стоит опираться на ощущения, прикидывать “на глаз” или надеяться, что солидного по внешнему виду кондиционера хватит на создаваемый комплекс. Неверный расчет параметров температурного режима в помещении может доставить большие неприятности в ходе эксплуатации кластерной системы.

Посмотрите на web-сайте фирмы-производителя характеристики вычислительных узлов, которые планируется установить. Найдите мощность тепловыделения. Если не удалось найти эту характеристику, найдите мощность блока питания в ваттах. Если в ваттах нет, то возьмите мощность в вольт-амперах. Полученное число можно принять за тепловую мощность (реально она будет чуть ниже). Ту же процедуру проведите со всей периферией кластера: управляющим узлом, файл-сервером, сетевым оборудованием и другими компонентами.

Рассмотрим пример. Потребляемая мощность упоминавшегося выше сервера HP DL145 составляет 500 Вт. Не забудьте о сетевых коммутаторах, поскольку они потребляют не так уж мало, как может показаться на первый взгляд. Тепловыделение коммутаторов 3Com SuperStack 3 Switch серии 4400 может составлять от 100 до 275 Вт. В результате, кластер из десяти серверов DL145 и двух коммутаторов 3Com потенциально может выделять до 5,55 кВт тепла.

Потребуется кондиционер, мощность охлаждения которого (но не потребляемая мощность, не путайте эти два понятия!) будет не меньше суммарной мощности тепловыделения всех узлов и всего набора периферии кластера. Обратите внимание, что сам кондиционер тоже может выделять тепло.

Кроме оборудования, необходимого для работы кластера, помните о других источниках тепла. Первый – это солнце. Если в помещении окна выходят на южную сторону, и на них нет ни отражающей пленки, ни штор или жалюзи, то комната будет нагреваться одним только солнцем до

внушительной температуры.

Второй источник тепла – это центральное отопление. В некоторых случаях проще вовсе убрать из комнаты батареи. Это не всегда реально, поэтому имеет смысл предусмотреть возможность явного отключения батарей в комнате. Помните, что при отключении батарей нужно предусмотреть слив оставшейся в них воды. Еще одна потенциальная опасность центрального отопления состоит в губительном для аппаратуры **увеличении влажности** при возникновении протечек соединений или разрывов труб. Установите **систему мониторинга с набором датчиков**, которая будет следить за температурой и влажностью в помещении. В случае необходимости она оповестит по электронной почте или мобильной связи о возникновении нештатной ситуации либо автоматически выключит оборудование.

Предупреждая почти уже традиционное предложение, хотим подчеркнуть сразу: **охлаждение "открытым окном" равносильно медленной казни кластерного оборудования**. Этого нельзя делать по многим причинам. Во-первых, как правило, этого явно недостаточно и не стоит сравнивать кластер с рабочими машинами в офисе, поскольку они работают в абсолютно разных режимах. Во-вторых, контроль за температурой в этом режиме просто невозможен. В-третьих, пыль и грязь с улицы за короткий промежуток времени приведут вычислительные узлы в полную негодность.

Немаловажен **выбор места установки кондиционера**. Ни в коем случае нельзя ставить кондиционер в непосредственной близости от стойки кластера. Лучшее место – на потолке или у потолка на расстоянии 3-4 или более метров от стоек (рис. 2.1).

Рис. 2.1. Пример установки кондиционеров

Заметим, что эффективность воздушного охлаждения сильно зависит от высоты помещения над уровнем моря. Например, национальная лаборатория в Лос Аламосе (США) расположена на высоте более двух километров над уровнем моря, и эффективность охлаждения воздухом там примерно в 2 раза ниже по сравнению с охлаждением на нулевой отметке.

Не забывайте о том, что любой кондиционер выделяет конденсат, который нужно будет куда-то отводить. Если внешний блок кондиционера установлен на первом этаже во дворе здания, то это не проблема. А если на 5-м этаже и над парадным входом, то придется обдумать этот вопрос специально.

Предмет особого внимания – это работа кондиционера зимой. Если мы сами, как правило, зимой им не пользуемся, то на кластер смена времен года, в этом смысле, почти никак не влияет. Основное требование состоит в том, чтобы кондиционер был рассчитан на работу зимой. Учтите, что на холоде выделяемый конденсат может замерзнуть, делая кондиционер неработоспособным, поэтому отвод должен быть спроектирован так, чтобы подобная ситуация не возникала.

Кроме традиционных кондиционеров можно воспользоваться еще

одним решением – водяным стоечным охлаждением. Такое решение предлагает, например, компания APC (http://www.apc.com/): система охлаждения ARAC15000U. Это полностью готовая стойка с комплектом водяного охлаждения мощностью до 18 кВт, которую нужно только правильно подключить.

Подчеркнем еще раз, что правильно организованное охлаждение крайне важно. За считанные минуты температура в помещении с работающим кластером даже менее чем из десятка узлов может подняться до 50-60 градусов, при этом температура внутри системных блоков будет 90 градусов и выше. В этих условиях оборудование может попросту перегореть, и по гарантии его в таком случае не заменят.

Следующим необходимым для аккуратного расчета параметром является **электропитание**. Подсчитайте мощность, потребляемую всеми узлами кластера и сопутствующими компонентами проекта (источниками бесперебойного питания, файловым хранилищем, коммутаторами и т.п.). Проконсультируйтесь с главным инженером, как обеспечить соответствующее электропитание размещаемого оборудования. Скорее всего, придется сделать дополнительную проводку. Если будете устанавливать мощные источники бесперебойного питания, то для них потребуется специальная проводка, так как в обычные розетки они не включаются. Проверяйте характеристики устанавливаемого оборудования: оборудование ставится мощное, стандартные и привычные решения могут уже и не работать. Например, обычная электрическая розетка рассчитана примерно на 16 ампер, т.е. примерно на 3,5 кВт, больше оборудования на нее вешать нельзя.

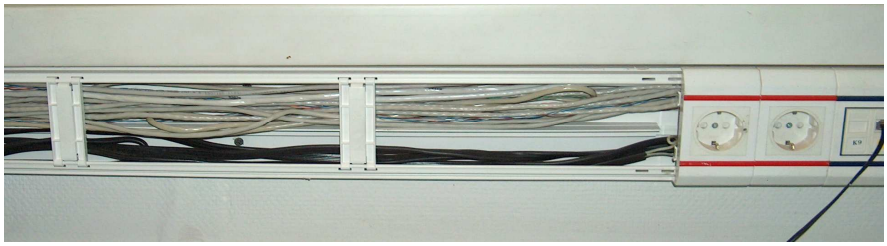

Рис. 2.2. Короб для укладки кабелей с открытой крышкой

Обратите внимание, что, меняя компоновку вычислительных узлов, их архитектуру, состав, степень отказоустойчивости, производителей, можно изменить энергопотребление всей кластерной установки в несколько раз. Стоит обратить особое внимание на новые модели процессорных платформ, которые при большей производительности по сравнению с существующими вариантами, как правило, имеют пониженное энергопотребление. Например, стоечный сервер C1000 от компании Rackable Systems, представленный в формате 1 U на базе двух четырехядерных процессоров Intel Xeon E5345 с частотой 2,33 ГГц, потребляет всего 290 Вт при пиковой производительности 74,5 Гфлопс. В последнее время характеристика “производительность на ватт” многими принимается в расчет, и свобода выбора существует большая.

Важна не только подводимая мощность электропитания, но также его надежность и гарантированное качество, особенно для систем, работающих в круглосуточном режиме. Во многих случаях все диктуется существующими возможностями здания и его инженерных коммуникаций. Очень хороший вариант – это обеспечить поступление питания к кластеру от разных электроподстанций. Установите силовую автоматику, которая обеспечит автоматическое переключение с одной подстанции на другую при возникновении аварийной ситуации. Подключите кластер через источники бесперебойного питания (UPS). Они сгладят незначительные

скачки напряжения, обеспечат стабильную работу вычислительной системы при кратковременных перебоях питания и позволят кластеру нормально завершить работу при отсутствии питания в течение длительного времени.

Ставить или нет источники бесперебойного питания, каждый решает для себя сам. Это сильно зависит от места установки, конкретных условий эксплуатации кластера, предъявляемых к нему требований. Расскажем лишь один пример из нашей практики. Организация поставила у себя вычислительную систему, но от UPS отказались, аргументируя высокой стабильностью питания. Какое-то время работали без проблем, но через некоторое время система стала останавливаться по непонятным причинам. Начали разбираться и сразу обратили внимание на то, что аварийные остановки происходят приблизительно в одно и тоже время. Выяснили причину остановок – сильные скачки напряжения. Провели расследование происходящего и поразились нелепости возникшей ситуации. В организации открыли столовую, в которой мощные электроплиты для приготовления пищи оказались в той же сети, что и вычислительная система. Утром плиты включали, а вечером выключали приблизительно в одно и тоже время, отсюда и регулярность бросков напряжения. Пока выясняли источник проблем, несколько недель использовать систему было невозможно.

Тщательно продумайте способ прокладки кабелей и их подведения к розеткам, UPS и другому оборудованию (рис. 2.2). Как правило, обычного розеточного удлинителя для питания кластера недостаточно. Пучки подводимых кабелей будут весьма внушительными, без должной укладки провода будут путаться, спотыкаться об них будет и неприятно, и опасно. Вопрос внешнего вида этого клубка оставим в стороне и обсуждать не будем. Обратите внимание на сечение используемой проводки, площадь которого зависит от мощности проектируемой кластерной системы.

Если помещение позволяет, то **рекомендуем использовать фальшпол** для подведения электропитания. Кроме фальшпола есть

вариант использования навесных корзин для проводки, но выглядят они не так аккуратно. Если ставите фальшполы, посоветуйтесь с инженерами относительно их прочности, поскольку мощные кластерные системы могут иметь значительный вес. В ноябре 2006 года для проектирования болидов Формулы-1 компания BMW установила кластер из 10 стоек стандартного формата, вес которого с полным комплектом источников бесперебойного питания составил 21 тонну. Вес кластерной системы СКИФ Cyberia со всем набором необходимого силового электрооборудования и устройств климат-контроля составляет 16 тонн.

Учтите, что **не стоит прокладывать рядом коммуникационную и силовую проводку**, поскольку циркулирующие по силовым кабелям токи могут создавать значительные наводки, приводя к ошибкам при передаче данных по сети кластера.

Вне зависимости от того, используются источники бесперебойного питания или нет, кластерную стойку и ее электропитание **необходимо заземлить**.

Уделите особое внимание покрытию пола в помещении. Это должен быть специальный **антистатический линолеум**, паркет или плитка (последние варианты нежелательны, так как могут просто не выдержать веса стойки, а паркет еще и пожароопасен). Не допускайте даже временной установки кластера в помещении с обычным линолеумом или ковровым покрытием.

Желательно иметь возможность обесточить все оборудование в помещении из одной точки, лучше всего – у входа. Например, с единого электрического щитка, расположенного в этом же помещении. Это очень важно для обеспечения безопасности, прежде всего, в случае возникновения экстренных ситуаций.

Необходимость специальных мер для повышения **вибрационной устойчивости** возникает не часто, однако задуматься об этой потенциальной проблеме стоит. Если рядом проходит железная дорога или трамвайные пути, то проходящие составы вполне могут служить причиной

неустойчивой работы отдельных компонент кластера, среди которых наиболее чувствительными к подобного рода помехам являются диски.

Кроме обязательной защиты кластера от хакеров, надо позаботиться и о его **физической защите** и ограничении доступа в помещение. Безусловно, помещение должно быть надежно укреплено, и стоимость соответствующей подготовки нужно включить заранее в проект. Ключи от помещения должны быть только у ответственных лиц. Возможно, стоит поставить сигнализацию или организовать систему видеонаблюдения с возможностью видеоконтроля за помещением через Интернет.

Требования к **пожарной безопасности** в случае с круглосуточно работающим кластером выше требований, предъявляемым к местам, где на компьютерах работают лишь в рабочие часы, следовательно, может потребоваться соответствующая доработка помещения. Как минимум, там должен быть порошковый огнетушитель и пожарная сигнализация, для серьезных систем лучше поставить автоматическую систему пожаротушения.

Будет ли кластер расширяться или как-то модифицироваться в будущем? Очень желательно решить этот вопрос сразу. Сейчас помещение подготовлено под текущую конфигурацию кластера и никаких проблем с обеспечением его работы не ожидается. А что будет при добавлении стоек, новых кластеров, расширении дисковых подсистем? Если планы развития вычислительной базы вполне реальны, то нужно сразу оценить имеющиеся возможности по его администрированию и сопровождению, площади для размещения нового оборудования, резервы по электропитанию, по мощности систем климат-контроля. По-хорошему, нужно еще раз аккуратно пройтись по всем вопросам данного раздела, сопоставив масштабность будущих изменений с теми последствиями и затратами, которые они вызовут.

Глава 3.
Проектирование архитектуры кластерной системы

Если говорить совсем кратко, то вычислительный кластер – это совокупность компьютеров, объединенных в рамках некоторой сети для решения одной задачи. В качестве вычислительных узлов обычно используются доступные на рынке однопроцессорные компьютеры, двух или четырехпроцессорные SMP-серверы. Каждый узел работает под управлением своей копии операционной системы, в качестве которой чаще всего используются варианты стандартных ОС: Linux, Windows, Solaris и другие. Состав и мощность узлов могут меняться даже в рамках одного кластера, что дает возможность создавать неоднородные системы. Выбор конкретной коммуникационной среды определяется многими факторами: особенностями класса решаемых задач, доступным финансированием, необходимостью последующего расширения кластера и т. п. Часто включают в конфигурацию кластера специализированные компьютеры, например, файл-сервер. Как правило, предоставляется возможность удаленного доступа на кластер через Интернет.

Ясно, что простор для творчества при проектировании кластеров огромен. Узлы могут не содержать локальных дисков, коммуникационная среда может одновременно использовать различные сетевые технологии, узлы не обязаны быть одинаковыми и масса прочих нюансов. Рассматривая крайние точки, кластером можно считать как пару ПК, связанных локальной сетью Fast Ethernet, так и рекордные вычислительные системы из первых строк списка 500 самых мощных систем мира, объединяющих тысячи процессоров.

Рассмотрим аппаратную сторону будущей кластерной системы. Возможны различные варианты критериев, с позиций которых будут оцениваться принимаемые в дальнейшем решения. Однозначно

посоветовать что-либо здесь просто невозможно, все определяется задачами каждого конкретного проекта. Часто при заданном бюджете требуется получить максимальную производительность системы на решении некоторого класса задач. Другой вариант – минимизировать стоимость проекта при заданной производительности. Для кого-то важна компактность системы, для других ее отказоустойчивость и высокая степень доступности. В последнее время очень важной характеристикой становится энергопотребление, которую рассматривают либо в абсолютном исчислении, либо принимают в расчет отношение производительности системы к ее энергопотреблению. Вариантов оценки и выбора существует много, поэтому, начиная проектирование кластера, определитесь, что первично, а чем в какой-то степени можно и пожертвовать.

Рассмотрим базовые компоненты и их характеристики, которые должны учитываться при построении кластерных систем:

- размещение и компоновка кластера,
- вычислительные узлы,
- управляющий узел,
- файл-сервер и хранилище данных,
- сетевая инфраструктура,
- источники бесперебойного питания.

В процессе проектирования важно осознать, что сейчас для конструирования кластерной системы в руки дается мощный конструктор, с помощью которого можно собрать именно ту систему, которая в наибольшей степени будет соответствовать решаемым задачам и бюджету проекта. Нужно правильно сформулировать свои пожелания и вовремя высказать их фирме-поставщику.

Начнем с того, что для эффективного обслуживания кластера не последнюю роль будет играть его компоновка. Лучшее решение – это **расположение кластера в стойке**. Даже для небольшого кластера из 4-6 узлов стойка уже уместна (рис. 3.1), при этом увеличение стоимости решения будет не столь существенным.

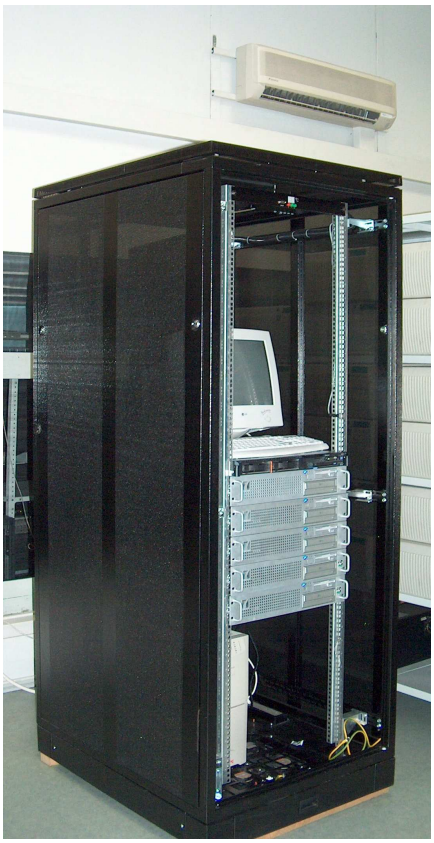

Рис. 3.1. Пример стойки с пятью узлами и монитором

Не стоит сильно экономить на собственно стойке, так как недоработки ее конструкции могут в последствии превратиться в неожиданные проблемы. При выборе стойки следует обратить внимание на следующие нюансы.

- Соответствие формата стойки формату узлов кластера (обычно это стандарт 19-дюймовой стойки).
- Наличие в комплектах узлов, головного узла, файл-сервера и сетевых коммутаторов рельсов (rail kit) для крепления в стойку.
- Соответствие крепежа рельсов и стойки. На практике используются два основных стандарта, которые называют Compaq и HP. В первом случае отверстия для крепления в профилях стойки будут круглыми, во втором – квадратными. Переходников с одного стандарта на другой не существует.
- Наличие кабельных органайзеров. К каждому вычислительному узлу кластера будут приходить как минимум питание и сеть, поэтому представьте, что будет твориться в местах скопления этих проводов.
- Перфорированная лицевая дверь стойки. Если поставить сплошную дверь, например, стеклянную, то обеспечить полноценный доступ

Рис. 3.2. Пример стойки с 30 узлами формата 1 U

холодного воздуха к узлам будет невозможно, однако почти все современные стоечные серверы захватывают воздух для охлаждения только спереди.

Расположить стойку в помещении нужно так, чтобы был удобный доступ к узлам и спереди, и сзади. Спереди должно быть достаточно места для того, чтобы поместился узел из стойки и человек. Сзади будет выходить тепло, поэтому расстояние до стены должно быть достаточным для его рассеивания.

Обратите внимание, что стоечный крепеж обычно поставляется отдельно, поэтому, приобретая стойку, купите там же соответствующие винты и гайки. Стоит закупить их с запасом, чтобы сберечь свое время в будущем.

Не менее важным, чем выбор стойки, является **выбор форм-фактора вычислительных узлов**. Устоявшихся решений в настоящее время существует несколько.

Стоечное решение с серверами 3-4 U. Обслуживание такого решения, наверное, самое легкое. Любое дополнительное периферийное оборудование устанавливается в такие серверы без особых проблем, охлаждение в них производится легко. Основным недостатком этого решения является то, что много таких узлов в стойку не войдет.

Стоечное решение с серверами 1-2 U (рис. 3.2)**.** Вариант компактен, таких узлов в стойку войдет уже больше, чем в предыдущем

случае, однако установить в них какую-то дополнительную плату иногда бывает весьма сложно. Тщательно продумайте конфигурацию узлов. Охлаждение в серверах небольшого размера, как правило, организуется сложнее, поэтому еще раз проверьте параметры и общий план разворачивания системы кондиционирования. Холодный воздух должен подаваться к передней части стойки, и кондиционер не должен располагаться слишком близко.

Стоечное решение на блейд-серверах (лезвиях). Самое компактное решение, поэтому и стоимость его при прочих равных условиях немного выше. Управление блейд-серверами обычно организуется с помощью специальных средств, которые нужно будет дополнительно изучить. Такие серверы, как правило, ограничены в плане расширения, и далеко не всегда в них можно установить необходимую новую плату. Например, если штатная коммуникационная сеть на базе Gigabit Ethernet не устраивает, то стоит задуматься о целесообразности такого решения. Некоторые компании предлагают на выбор несколько вариантов сетевого комплектования.

В последнее время появились компактные и исключительно мощные стоечные кластерные решения. Например, компания Rackable Systems (`http://www.rackable.com`) в конце 2006 года анонсировала выпуск кластерных систем, построенных на основе стоечных серверов размера 1 U и четырехядерных процессоров Intel Xeon серии 5300. В одной стандартной стойке на 42 U может быть

Рис. 3.3. Пример расположения кластера из 16 узлов на стеллажах

собран кластер с пиковой производительностью 6,5 Тфлопс.

Немного особняком стоит вариант, в котором *обычные корпуса серверов располагаются на стеллажах* (рис. 3.3)**.** В этом случае кластер займет значительно больше площади, чем стоечное решение. Оборудование обойдется дешевле, однако обслуживание кластера в будущем станет организационно сложнее, а его развитие будет сильно ограничено.

Если число вычислительных узлов невелико, то может быть полезен KVM – переключатель, позволяющий использовать одну клавиатуру и один монитор для всех узлов кластера. Более перспективным и удобным решением является использование сервисной сети (о ней будет рассказано ниже), в этом случае переключатель, скорее всего, не потребуется.

В некоторых случаях идут на совмещение функций вычислительного кластера и отдельных рабочих мест (учебного класса). На каждый узел ставится полный комплект: монитор, клавиатура, мышь, а разграничение функций производится административно, чаще всего, по времени: днем оборудование используется в режиме отдельных рабочих мест, а ночью в режиме кластерной вычислительной системы.

Выбор архитектуры и параметров вычислительных узлов, во многом, определит характеристики будущего кластера. Тип и частота процессоров, свойства чипсета, частота системной шины, объем и частота оперативной памяти, ее конструктив, параметры кэш-памяти, состав портов и поддержка периферии, возможности для будущей модернизации узлов – со всем этим нужно определиться сейчас. Желательно хотя бы приблизительно представить круг задач, которые будут решаться на кластере, чтобы подбирать состав вычислительных узлов, уже исходя из требований, предъявляемых этими задачами.

Основа – это **выбор процессора**, который будет диктовать многие дальнейшие шаги. Одни вычислительные задачи оптимизированы под процессоры Intel Xeon EM64T, какие-то приложения показывают хорошие

характеристики на AMD Opteron, для кого-то очевидные преимущества окажутся у Intel Itanium-2, а в каких-то случаях программа оптимально использует технологию Hyper-Threading. Самым лучшим и правильным вариантом является предварительное тестирование узлов на типичных приложениях, которое покажет реально достижимые в каждом случае параметры. В некоторых случаях узлы на тестирование может предоставить потенциальный поставщик кластерной системы либо сама компания-производитель процессоров. Если будет использоваться новое оборудование или же компоненты в нестандартной конфигурации, то предварительное тестирование нужно выполнить в обязательном порядке либо заручиться гарантийными обязательствами со стороны поставщика. Если есть хотя бы малейшие сомнения в достижении заявленных параметров, то обязательно проконсультируйтесь с профессионалами. А если совсем честно, то проконсультироваться лучше в любом случае.

В последнее время все производители процессоров переходят на многоядерные технологии, что нужно обязательно учитывать. Выбирая одноядерные модели процессоров, можно значительно сэкономить в цене на процессоры, однако больших перспектив это решение не имеет. Вычислительный мир стал параллельным, число ядер на кристалле будет только увеличиваться. Мы уже говорили о том, что первый основной толчок к массовому использованию параллельных вычислительных технологий дали кластерные системы. Окончательный переход к параллелизму определила многоядерность современных процессоров. Подумайте еще раз, настолько ли важна сиюминутная экономия, чтобы отказываться от будущих технологий.

Останавливаясь на многоядерном варианте, проверьте, что эта пока еще новая особенность поддерживается на всех уровнях ПО: в компиляторах, библиотеках, в необходимых прикладных пакетах и системах. Есть ли смысл вкладываться в дорогую аппаратуру, если потом ее нельзя будет эффективно использовать? Однако при всех “за” и “против” различных вариантов, хотим подчеркнуть еще раз: **в конечном итоге все**

определяется способностью кластера решить задачи проекта. Если с помощью такой-то конфигурации кластера задачи будут решены оптимально, то этот вариант и будет хорошим.

При проектировании архитектуры узлов обратите внимание на то, что иногда с увеличением тактовой частоты процессоров растет не только их производительность и стоимость. Для некоторых моделей это ведет к росту энергопотребления, следовательно, выбор более мощных узлов может потребовать перерасчета необходимых характеристик по электричеству и охлаждению для кластерного проекта в целом. Например, можно остановиться на выборе двуядерных процессоров Intel Xeon серии 5100 (Woodcrest) с тактовыми частотами 1,6 ГГц, 1,86 ГГц, 2 ГГц, 2,33 ГГц, 2,66 ГГц и 3 ГГц. При этом энергопотребление всех моделей составляет 65 Вт, а у старшей модели – 80 Вт. Аналогично у четырехядерного процессора Intel Xeon серии 5300 (Clovertown): энергопотребление моделей на 1,6 ГГц, 1,86 ГГц, 2,33 ГГц равно 80 Вт, а у старшей модели на 2,66 ГГц – 120 Вт. Четырехядерный процессор AMD Barcelona будет выпускаться в нескольких версиях, различающихся по тепловым характеристикам: основная модель будет выделять 95 Вт, низковольтовая – 68 Вт, а высокопроизводительная – 120 Вт.

Для многих задач критически важным является **объем и скорость работы оперативной памяти**. В этом случае лучше пожертвовать одним-двумя вычислительными узлами в пользу покупки памяти с требуемыми характеристиками. Если рассматривается вариант возможного увеличения объема оперативной памяти в будущем, то обратите внимание на конструктив материнской платы и число свободных слотов для дополнительных модулей.

Схожий и весьма важный вопрос – **структура и объем кэш-памяти**. В зависимости от типа процессора эти характеристики могут сильно меняться, влияя на эффективность работы конечных приложений. При этом важны не только количественные показатели, но и такие свойства, как разделение кэш-памяти последнего уровня и доступа к

системной шине между отдельными ядрами процессора. Например, двуядерный процессор Intel Itanium-2 Montecito/9050 имеет по 12 Мбайт кэш-памяти третьего уровня на каждое ядро (24 Мбайт на кристалл), однако оба ядра разделяют один канал доступа к системной шине процессора. На практике, соотношение между локальностью использования данных и локальностью вычислений в программе, степенью пересечения между ядрами по используемым данным для каждого конкретного приложения определят реальный выигрыш как от структуры и иерархии памяти, так и от метода доступа к общим ресурсам каждого конкретного процессора.

Наличие **локальных дисков** на узлах кластера, на первый взгляд, кажется излишним. В самом деле, зачем тратить лишние деньги? Эти диски работать почти не будут, загрузку ОС можно сделать по сети, да и ее обновления в будущем делать будет проще. Примерно так нередко и рассуждают при построении кластера.

Верно, однако отсутствие локальных дисков влечет за собой и потенциальные проблемы, о которых сначала не всегда задумываются. Например, если возникнут сетевые проблемы или же проблемы с сервером DHCP/NFS/TFTP, то их решение и даже определение причин будет изрядно затруднено. Опять-таки, как только задача исчерпает всю оперативную память и затребует еще, ее просто снимут, поскольку области свопинга нет. Есть возможность использовать swap-over-nfs, но в этом случае скорость работы задачи упадет в десятки, а то и сотни раз. Более того, может существенно снизиться скорость работы других приложений, так как нагрузка на общий NFS-сервер, естественно, отразится на всех узлах. При этом цена обсуждаемого вопроса по сравнению со стоимостью самого узла крайне невысока. Достаточно поставить на узел один IDE- или SATA-диск минимального объема для того, чтобы в будущем избежать массы проблем.

Еще одним весомым аргументом за включение дисков в состав узлов является возможность локализации ввода/вывода для определенного класса приложений. Если этого не предусмотреть, то все файловые

операции программ будут идти через файл-сервер и тот или иной вариант сетевой файловой системы, что медленнее, а иногда и намного медленнее, чем использование жестких дисков на самих узлах. Это особенно касается приложений класса Out-of-Core, работающих с данными, объем которых намного превосходит объем суммарной оперативной памяти компьютера в целом. Единственным вариантом эффективной работы таких приложений является аккуратная организация обменов с дисками для подкачки новых данных в память, что должно проходить на фоне и за время обработки данных, уже лежащих в памяти.

Floppy- и CD/DVD-приводы на узлах используются редко, в основном для начальной установки операционной системы. Некоторые модели серверов позволяют использовать виртуальные приводы, подключаемые по сети с головного узла. Если такой возможности не предусмотрено, можно воспользоваться приводом, подключаемым через USB. В этом случае узлы должны поддерживать USB и загрузку с подключаемых приводов, а использованный в них USB-контроллер должен поддерживаться выбранной операционной системой.

Если планируется использовать платы ServNet или аналогичные для организации сервисной сети, то может потребоваться наличие в узлах COM-порта.

Говоря об общей архитектуре, отметим, что кроме вычислительных узлов, необходимо предусмотреть **головной узел кластерной системы**. На головном узле пользователи компилируют свои программы, готовят данные для счета, проводят предварительную обработку данных. С головного узла производится запуск задач.

Совмещать головной узел с одним из вычислительных узлом не рекомендуется, так как и работа пользователей будет затруднена, и эффективность параллельных программ, распределенных на такой узел, будет ниже. Вместе с тем, в некоторых случаях подобное совмещение вполне оправдано и само по себе не может рассматриваться дефектом

проекта: все определяется исключительно режимом использования будущей кластерной системы и стоящими задачами.

Так как на головном узле, скорее всего, будут работать несколько пользователей одновременно, то дополнительная оперативная память на нем лишней не будет. Для ускорения работы локальных приложений стоит установить в него быстрый жесткий диск или рейд-контроллер. К процессору головного узла больших требований, как правило, не предъявляется, вычислительные задачи на нем работать не будут, а для компиляции большой мощности не нужно. Если заранее известно, что пользователей будет много, то имеет смысл использовать компьютер с несколькими процессорами, в противном случае и это не является обязательным требованием.

Весьма тонкий момент, который следует продумать заранее, это использование на головной машине процессора, отличного от процессоров вычислительных узлов. Часто компиляторы устроены так, что выполняют различного рода оптимизацию кода именно под тот процессор, на котором идет собственно процесс компиляции. Эта проблема, как правило, легко решается, поскольку многие компиляторы поддерживают режим кросс-компиляции, генерируя код целевого процессора, однако убедиться в эффективности и целесообразности такого режима работы нужно до заказа оборудования.

Кроме головного узла, в составе кластера **необходим файл-сервер**. Его функции могут быть переложены на головной узел, если предполагаемые нагрузки на сетевой диск будут не очень большими.

На файл-сервере стоит предусмотреть аппаратный рейд-контроллер. Оптимальным является использование RAID-5 или RAID-6, так как в этом случае надежность, а часто и скорость работы повышаются по сравнению с одиночным диском. Выбирая рейд-контроллер, обратите внимание на следующие параметры:

- максимальное количество подключаемых дисков,
- число дисков, использующихся для контроля четности в RAID-5,

- возможность “горячей” замены дисков,
- поддержка контроллера операционной системой.

В настоящее время большинство серверных материнских плат имеют интегрированный рейд-контроллер, но к нему не всегда можно подключить более 2-4 дисков. На рынке подобные решения представлены исключительно широко: от внутренних плат рейд-контроллеров, до аппаратных файл-серверов.

Если предполагается использовать RAID-1, -5 или -6, то полезно подумать о запасных (spare) дисках. Они не используются до тех пор, пока основные диски работают нормально. Как только один из жёстких дисков выходит из строя, вместо него подключается запасной. На него записывается вся необходимая информация, которая восстанавливается с работающих носителей за счёт изначальной избыточности хранения (поэтому такие диски не столь полезны в RAID-0), и работа RAID-массива продолжается в нормальном режиме. На один RAID-массив стоит выделить хотя бы один такой диск. Убедитесь, что RAID-контроллер поддерживает работу с запасными дисками и умеет включать их в работу автоматически, без остановки работы массива.

Сетевая инфраструктура в современных кластерных системах представлена тремя вариантами сетей: коммуникационная, транспортная и сервисная. Во многих проектах все три сети присутствуют одновременно.

Коммуникационная сеть – это тот компонент, который во многом определит эффективность работы программ на будущем кластере, поскольку именно с помощью этой сети процессы параллельных программ обмениваются данными между собой. Какими характеристиками выражается производительность коммуникационных сетей в кластерных системах? Для работы параллельных приложений в наибольшей степени важны две характеристики: латентность и пропускная способность сети. Латентность – это время, необходимое для передачи сообщения нулевой длины от одного процесса параллельной программы другому. Пропускная способность сети определяет собственно скорость передачи информации

по каналам связи. Если в программе много маленьких сообщений, то на эффективность ее выполнения сильно скажется латентность. Если сообщения передаются большими порциями, то важна высокая пропускная способность каналов связи. Заметим, что из-за латентности максимальная скорость передачи по сети не может быть достигнута на сообщениях с небольшой длиной, а насколько быстро реальные показатели приближаются к заявленным характеристикам – это будет определяться и типом сети, и набором оборудования от конкретного производителя. На практике не столько важны указанные производителем пиковые характеристики коммуникационной аппаратуры, сколько реальные показатели, достигаемые на уровне приложений пользователей, в частности, на уровне MPI-программ или прикладных пакетов.

Выбор коммуникационной сети полностью определяется свойствами решаемых задач (выполняемых программ) и целями создания кластерной системы. Если обменов между параллельными процессами приложения мало, то достаточно использовать какой-либо вариант Ethernet. В зависимости от объемов пересылок это 100 Мбит/с или 1 Гбит/с. Серьезным недостатком сети Ethernet является высокая латентность, составляющая около 130-150 мкс на пакет. Некоторые современные коммутаторы позволяют снизить этот показатель до 50-60 мкс, однако для многих вычислительных задач это все равно не решает проблемы.

В данный момент на рынке доступны несколько стандартов сетевого оборудования, дающего скорости более 1 Гбит/с, и со значительно более низкой латентностью. Рассмотрим наиболее распространенные и доступные варианты.

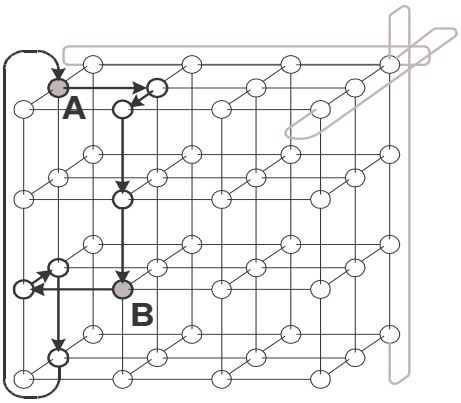

Рис. 3.4. Соединение узлов в трехмерный тор в сети SCI

Сетевая технология SCI. Скорость передачи данных около 700 Мбайт/с (5,6 Гбит/с), аппаратная латентность на уровне 0,2 мкс. На MPI-приложениях скорость передачи данных достигает значений около 350 Мбайт/с, а латентность – 1,4 мкс. Особенностью данной сетевой технологии является то, что она не требует коммутатора. Все сетевые адаптеры соединяются в кольцо либо тор (поддерживаются двумерные и трехмерные топологии). На рис. 3.4 показан пример объединения машин кластера с топологией трехмерного тора. Два кластера суперкомпьютерного комплекса НИВЦ МГУ использовали технологию SCI, объединяя узлы в соответствии со структурой двумерного тора. Кластер СКИФ-К500, установленный в ОИПИ НАНБ, использует технологию SCI с объединением 64 узлов в трехмерный тор.

С одной стороны такая архитектура кажется выгодной, так как не нужно приобретать дополнительного оборудования, только сетевые адаптеры. При добавлении новых узлов также не надо заботиться о покупке нового коммутатора. С другой стороны, опыт показывает, что эта технология весьма трудоемка при первоначальной настройке. Более того, находить ошибки или сбои в коммутации такой сети крайне сложно. Средств для ее диагностики существует очень мало. Но самым неудобным является то, что выход их строя одного вычислительного узла означает отключение двух (для двумерного тора) или трех (для трехмерного) колец связи. Сбой двух узлов в двумерном торе приведет к отключению еще двух

других узлов, лежащих на пересечении отключенных колец. Для трехмерного тора аналогичное отключение произойдет при сбое трех узлов.

Сетевая технология Myrinet-2000/Myri-10G. Скорость передачи данных – 2 Гбит/с и 10 Гбит/с соответственно. Для MPI-приложений достигается скорость 247 Мбайт/с и 1,2 Гбайт/с, латентность – 2,6 мкс и 2 мкс. Данная технология использует оптоволоконные соединения. При построении сети используются коммутаторы, которые можно соединять между собой. Драйверы и программное обеспечение распространяются бесплатно. Стандарт Myrinet существует довольно давно, поэтому широко поддерживается. Технология Myrinet в течение уже многих лет активно используется в Межведомственном суперкомпьютерном центре РАН для построения нескольких поколений суперкомпьютерных кластерных систем.

Сетевая технология InfiniBand. Скорость передачи данных до 10 Гбит/с, латентность – менее 1 мкс. На MPI-приложениях скорость передачи данных в дуплексном режиме достигает 2,5 Гбайт/с, латентность – 1,3-1,7 мкс. Реальная скорость передачи в настоящее время ограничивается скоростью шин PCI-X и PCI-Express. В случае PCI-X адаптеров скорость составит не более 1 Гбайт/с. Не так давно анонсировано сетевое оборудование, построенное по этой же технологии, со скоростью передачи данных до 30 Гбит/с.

Технология InfiniBand разработана альянсом ведущих мировых разработчиков: Intel, IBM, Cisco, Sun и рядом других. В настоящее время на компьютерном рынке представлено множество брендов, поставляющих это оборудование, например, только что упоминавшаяся компания Cisco, Qlogic или Mellanox.

Данная технология одна из самых молодых среди аналогов, но и одна из самых быстро развивающихся. При построении сети используются коммутаторы, которые можно объединять между собой. Аппаратно поддерживается маршрутизация пакетов, при этом дублирующиеся

маршруты используются для балансировки нагрузки.

При соединении нескольких коммутаторов часто используется топология “Fat-Tree”. В этом случае коммутаторы соединяются несколькими каналами параллельно, что увеличивает скорость канала между ними. Базовая поддержка InfiniBand уже есть в последних ядрах Linux. Существуют библиотеки MPI и программное обеспечение для работы с InfiniBand, распространяемые бесплатно.

В настоящее время коммуникационные технологии развиваются очень быстро. Недавно появились и уже сегодня доступны на рынке технологии Quadrics и 10 Gbit Ethernet, представлены опытные образцы технологии 100 Gbit Ethernet.

Кроме коммуникационной сети, по которой будут обмениваться информацией параллельные приложения, рекомендуем отдельно поставить **транспортную сеть** (иногда ее называют управляющей сетью). По ней происходит передача данных сетевой файловой системы и может осуществляться управление вычислительными узлами.

Разделение коммуникационной и транспортной сетей необходимо для того, чтобы вспомогательный трафик не мешал работе параллельных приложений. Даже если в качестве коммуникационной сети используется Ethernet, поставьте второй коммутатор и настройте на нем независимую транспортную сеть. Затраты небольшие, а в трудных ситуациях это поможет не только спасти данные, но и значительно облегчить работу администратору и избежать долгих часов простоя системы.

Еще одна сеть, о которой нужно обязательно упомянуть, это **сервисная сеть**. Эта сеть используется исключительно для обслуживания вычислительных узлов. Оконечным оборудованием для этой сети должен быть, строго говоря, даже не вычислительный узел, а устройство, способное его контролировать. На многих современных серверных платформах такие устройства устанавливаются изначально. Эти устройства доступны и отдельно, например, платы Hewlett-Packard Lights-Out или платы ServNet, разработанные в Институте программных систем РАН. Все

они предоставляют различный уровень сервиса от минимальной возможности удаленно перезагрузить узел, включить или выключить питание, до получения графической консоли.

Крайне важной может оказаться информация с системной консоли при диагностике сбоев: узел завис, и получить доступ к консоли штатными средствами невозможно. Большинство имеющихся на рынке устройств доступ к системной консоли дают. Например, плата ServNet позволяет получить не только доступ к консоли через стандартный COM-порт, но также управлять питанием узла. Такой функциональности вполне достаточно для решения большинства задач по удаленному обслуживанию кластера.

Получить дополнительную информацию об упомянутых технологиях можно на сайте `http://skif.pereslavl.ru/` (поиском слова ServNet) и поиском по ключевым словам “Lights-Out 100 Remote Management Card” на сайте `http://www.hp.com`.

Наличие сервисной сети сильно упрощает обслуживание кластера. Администратору не надо идти в другое помещение для того, чтобы перезагрузить зависший узел, можно быстро выяснить причину зависания, а в некоторых случаях даже переустановить ОС прямо со своего рабочего места.

Очень важный момент, про который забывают или на котором часто возникает желание сэкономить, – это обеспечение качественного электропитания. От этого будет зависеть и долговечность работы кластера, и степень его доступности, и работоспособность критических приложений. Включить в конфигурацию **источник бесперебойного питания** стоит хотя бы для того, чтобы элементарно защитить вычислительные узлы. Корректное выключение кластера в случае аварийного выключения питания поможет спасти данные долгих расчетов, а очень часто и значительное время, необходимое на восстановление работоспособности кластера.

На практике чаще всего возникают кратковременные скачки

напряжения, с чем любой UPS прекрасно справится. Более серьезный вариант предполагает, что UPS сможет обеспечить работу кластера в течение 5-10 минут. Рассчитать мощность UPS можно исходя из мощности подключенного к нему оборудования. Для работоспособности приложений во время непредвиденных скачков по питанию не забудьте подключить к UPS не только вычислительные узлы, но и все сетевые коммутаторы.

Обязательно завершите проектирование **составлением подробной схемы кластерной системы**, показывающей архитектуру комплекса, его составные части, детали компоновки, особенности размещения, структуру коммуникаций. Не забывайте в будущем модифицировать схему сразу после модернизации кластера, поскольку даже очевидные действия со временем забываются, и много времени будет уходить на восстановление и правильное описание текущей конфигурации.

И еще раз повторим вопрос, поставленный в конце предыдущего раздела. Будет ли кластер расширяться или как-то модифицироваться в будущем? Финансирование проекта часто появляется порциями и возникает большое желание сначала купить часть системы, а потом ее нарастить еще каким-то числом узлов. К такому сценарию развития кластерного проекта подталкивает и тот факт, что архитектуры кластерных систем хорошо масштабируются, поэтому принципиальных препятствий для расширения нет.

Если выбирается вариант последовательного развития кластерной системы, то, как правило, выбирают два пути. Первый предполагает добавления какого-то числа новых вычислительных узлов в существующую систему. Основная проблема здесь одна: если возможность для расширения появилась через полгода-год после создания первой версии системы, то возникает вопрос о целесообразности покупки уже устаревшего оборудования. Приобретать такие же узлы уже не хочется, так как на рынке по схожей же цене есть более привлекательные модели. Но если купить современные варианты узлов, то кластер перестанет быть однородным, и возникнут очевидные проблемы с его использованием.

Второй путь развития – это полная замена каких-либо компонентов системы на новые аналоги: более мощные процессоры, больший объем памяти, новые диски, новые материнские платы с увеличенной частотой системной шины и т.п. И по такому пути вполне можно двигаться, но постарайтесь сразу ответить на сакраментальный вопрос: что делать со старыми комплектующими?

Любое оборудование, даже фирменное и самое дорогое, ломается. Не пренебрегайте изучением **гарантийных обязательств** производителя, а также его **технической поддержкой** и **сервисным обслуживанием**. Эти позиции можно и нужно включать в проект. Гарантия производителя позволит заменить сбойную деталь или целый узел в случае брака, но на это может уйти несколько дней или даже недель. Техническая поддержка позволит избавиться от необходимости искать самим сбойный компонент и перепоручить эту обязанность профессионалам. Особенно удобно, если предусмотрена поддержка “on-site”, когда специалист выезжает прямо на место установки кластера. Сервисное обслуживание позволит проводить не только обмен сбойных компонент, но и ремонт узлов.

Наличие гарантии, безусловно, дает уверенность в том, что сбойный компонент будет заменен. Однако оно не дает уверенности в том, что он будет заменен быстро. Если приложению необходимы все процессоры, а один из них вышел из строя, то дело встанет. Придется ждать замены, которая займет от нескольких дней, до нескольких недель в зависимости от гарантийных обязательств поставщика и производителя оборудования.

Чтобы избежать подобной неопределенности хорошим решением станет **резервный узел**. Его можно заранее сконфигурировать, но не включать в общее счетное поле. При выходе одного из узлов из строя его можно заменить резервным на время ремонта. Это особенно важно для тех кластерных систем, в которых узлы расположены по некоторой жестко заданной структуре, а выход из строя одного узла может нарушить структуру целого сегмента (например, SCI-кластеры с топологией

двумерного или трехмерного тора).

Другое решение – это **резервные запчасти**. Вместо целого узла можно приобрести в резерв только наиболее часто ломающиеся детали, в частности, жесткие диски, вентиляторы для охлаждения, модули памяти, блоки питания, а также запасные коммуникационные карты.

По какому пути обеспечения резерва пойти в каждом конкретном проекте: резервные узлы, компоненты или же вообще отказаться от резерва, – это предстоит решить руководству каждого кластерного проекта самостоятельно. Решение определяется и бюджетом, и стоящими задачами, и требуемой степенью доступности кластерной системы. Что-либо сказать априори одинаково хорошо подходящее для любого случая здесь практически невозможно.

Всегда хочется как-то проверить принятые решения и сравнить с чем-то уже работающим, поэтому в **Приложении 1** приведено несколько примеров программно-аппаратных конфигураций реально существующих кластерных систем. Множество других вариантов можно найти на страницах информационно-аналитического центра Parallel.ru, а также на сайте `http://www.supercomputers.ru` списка Top50 самых мощных компьютеров СНГ – большинство из них являются кластерами.

Глава 4.
Поставка, монтаж и первичное тестирование кластера

Определившись с архитектурой кластерной системы, нужно решить еще один, казалось бы, формальный и не столь существенный вопрос: кто будет выполнять поставку кластерной системы? Предвидим, что первая реакция может быть примерно такой: железо – оно и есть железо, если цена устраивает, то какая разница, что за фирма нам его привезет? Не все так просто. Кроме стоимости есть немало вопросов, которые нужно принимать в расчет. Каковы сроки поставки? Входит ли в стоимость контракта монтаж кластера, его тестирование и настройка? Над этим стоит задуматься, если нет собственного опыта работы с подобного рода оборудованием или уверенности в качественном выполнении всех перечисленных этапов своими силами. И над этим стоит особо серьезно задуматься, если будут использоваться нестандартные устройства или комбинации компонентов. В одном из проектов Московского университета поставщик после оплаты и отгрузки нам коробок с оборудованием свое общение с нами фактически прекратил. Все наши настойчивые указания на то, что в материнских платах есть явный дефект, не позволяющий получать больше 30% от заявленной скорости передачи данных по интерфейсу SCI, оставались без внимания. Поставщик полностью устранился от решения проблем. То, что в такой ситуации принципиально снижается эффективность решения задач, для чего и создавался кластер с дорогой высокопроизводительной сетью SCI, в расчет не принималось. Осознав бессмысленность общения с этими людьми, мы стали работать с фирмой-производителем данных плат напрямую, которая к ее чести признала обнаруженный нами дефект собственной ошибкой и за свой счет заменила неисправное оборудование. На поиски проблем, осознание несостоятельности возлагавшихся изначально на поставщика надежд и

урегулирование всех вопросов потребовался примерно год. Естественно, что “услугами” того поставщика мы больше не пользовались, и всем советовали держаться от него подальше.

Влияет ли поставщик оборудования на качество решения прикладных задач? Как мы только что увидели, да, может повлиять самым непосредственным образом. Другая компания в недавнем проекте в конфигурацию кластера заложила интересную модель многовходового коммутатора InfiniBand: параметры превосходные, однако сильно смущало отсутствие практики ее использования. Решились, поскольку поставщик, будучи со своей стороны заинтересованным в освоении новой модели, взял решение потенциальных проблем на себя. Предчувствия не обманули, вопросы посыпались как из рога изобилия, но все они были решены на удивление быстро. Поставщик заранее установил необходимые контакты, быстро наладил линию общения с инженерами в режиме on-line, а затем для устранения найденной в оборудовании ошибки привез специалиста компании-производителя данного коммутатора на место установки кластерной системы. Все было сделано в кратчайшие сроки, никаких попыток переложить ответственность на кого-то еще не было. **Выбор надежного партнера по поставке оборудования** – это один из ключевых моментов, определяющих успех кластерного проекта в целом.

Других вопросов, о которых нужно также подумать при выборе поставщика, набирается немало. Входит ли в указанную стоимость гарантийное и сервисное обслуживание? Каковы сроки гарантийной замены вышедшего из строя оборудования? Выполняет ли поставщик настройку программного обеспечения или просто привозит ящики с аппаратурой? Компания может поставить только кластер или она в состоянии спроектировать весь вычислительный комплекс “под ключ”, включая системы хранения данных, энергоснабжения, климатического контроля, мониторинга, выполнить интеграцию всего комплекса в инфраструктуру организации? Немаловажный вопрос – наличие у компании опыта выполнения подобных проектов и наличие в штате

собственных квалифицированных инженеров. Не так сложно начать дело, гораздо сложнее постоянно поддерживать его выполнение на достойном уровне. С первичной настройкой поставщику смогут помочь сторонние специалисты, а кто поможет оперативно разобраться с нестандартными вопросами в последующие годы работы кластера? Хорошей проверкой основательности подхода к делу служит состав документации, которую поставщик дает вместе с кластерной системой: руководство оператора, администратора, пользователя, описание программно-аппаратной среды, параметров аппаратуры и настроек операционной системы, состав базового и специализированного программного обеспечения. Во всяком случае, при заключении контрактов и проведении конкурсных торгов ставьте это обязательным условием, тогда есть надежда получить, быть может, и не безупречный, но все же вариант документации.

А в целом, выстраивая общение с поставщиком, не забывайте время от времени задавать себе вопрос: “Что мне хочется больше: заниматься своими задачами или переквалифицироваться в системного программиста с инженерным уклоном?”. Действуйте далее, исходя из честного ответа.

Если принято решение собирать кластерное оборудование в комплекс самостоятельно, то продумайте и заранее опишите четкий план работы и последовательность всех

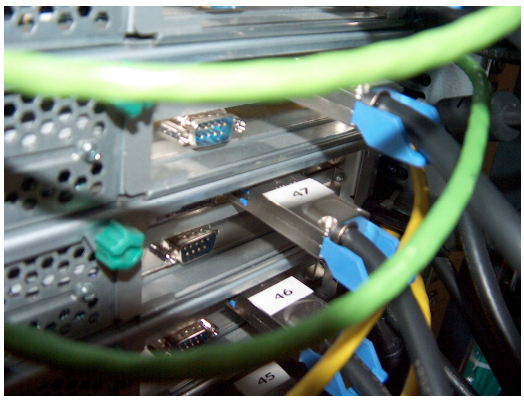

Рис. 4.1. Пример маркировки кабелей

действий. Начните с более общих вопросов и постепенно детализируйте их до тех пор, пока не получите для себя предельно ясную картину. Чтобы не запутаться в процессе сборки и настройки, используйте составленную на предыдущем этапе схему кластера, дополняя ее по ходу работ новой информацией.

Чтобы не запутаться в проводке, а ее будет много, проведите **предварительную маркировку кабелей** (рис. 4.1). Обратите внимание, что это необходимо сделать перед (!) коммутацией узлов. Сами узлы также полезно промаркировать. Для маркировки кабелей можно использовать обычные маркеры либо пластиковые стяжки с площадками и наклейки.

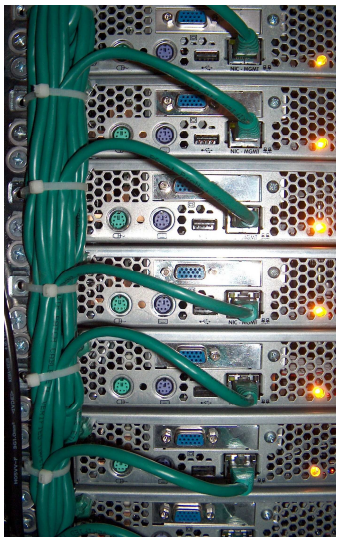

Рис. 4.2. Пример организации кабелей пластиковыми стяжками

Очень удобным может оказаться специальный маркировочный принтер (например, DYMO Letratag или аналогичный). Такой принтер печатает наклейки на самоклеющейся ленте и имеет размер чуть больше калькулятора. Полученные наклейки можно использовать как для маркировки кабелей (на площадках стяжек), так и для маркировки оборудования.

Большую помощь в креплении кабелей могут оказать **пластиковые стяжки**, не пренебрегайте ими (рис. 4.2). Это не только выглядит опрятно, но и намного удобнее в эксплуатации. Попробуйте разобраться с проблемами, которые возникли где-то в кабельном хозяйстве, показанном на рис. 4.3.

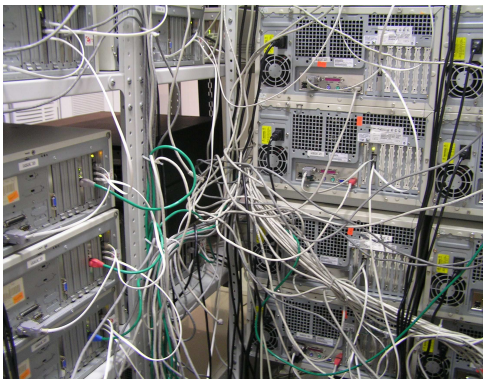

Рис. 4.3. Пример неудачной организации кабельного хозяйства

Установите и подключите источники бесперебойного питания. Как правило, подключение должен выполнять квалифицированный специалист. Обязательно тщательно **изучите инструкцию по установке и подключению UPS**. Ни в коем случае не подключайте UPS через сетевые фильтры ("пилоты" и т.п.), поскольку подобные фильтры крайне негативно влияют на работу источников. Не стоит включать UPS и в обычную бытовую сеть, так как частые перепады напряжения не прибавят им срока службы. Если инженерные возможности помещения позволяют, то оптимально включить их в специально выделенную компьютерную сеть через собственный автоматический выключатель.

После этого можно устанавливать вычислительные узлы и коммутаторы. Заранее продумайте, как будут проведены все кабели: питание, коммуникационная, транспортная и сервисная сети. Возможно, стоит заранее провести все или часть кабелей, а уже потом устанавливать узлы. Иногда некоторые проблемы при монтаже создают большие устройства, занимающие по 8-10 U и перекрывающие после своей установки доступ к некоторым гнездам, разъемам или же мешающие прокладке кабелей. Если такие устройства в конфигурации кластера есть, то их монтаж спланируйте особо тщательно, чтобы не пришлось подобные тяжести снимать и монтировать много раз.

После установки оборудования убедитесь, что источники бесперебойного питания находятся в рабочем состоянии. Подключите

питание ко всем компонентам. Теперь можно провести пробный запуск оборудования. Проследите, все ли узлы стартовали нормально.

Специально проверьте, что источники бесперебойного питания выполняют свою роль и в состоянии обеспечить нормальное функционирование необходимого оборудования при исчезновении питания. Это и в самом деле важный шаг, который сделать нужно именно сейчас до перевода кластера в режим эксплуатации. Реальный случай. В представительстве крупной европейской компании все критически важное оборудование подключили через мощный UPS, желая подстраховаться на случай возможного отключения питания. Такой случай возник один раз за пять лет работы, причем первым отключился тот самый UPS…

Для многих монтаж и интеграция кластера из россыпи оборудования в стойки является вполне выполнимым делом. Главное – это правильно оценить свои силы и возможности. И целесообразность. Стоит ли изучать особенности сборки, если даже не понятно, когда еще понадобится полученный в результате всего этого опыт. Может быть стоит обратиться за помощью к профессионалам, которые знают про подводные камни, да и гарантию дадут? Над этим выбором имеет смысл подумать.

Чтобы дать некоторое представление о трудоемкости сборки узлов кластера в стойку, опишем пример из нашей практики. Рассмотрим сборку кластера, состоящего из 80 узлов размера 1 U, четырех стандартных стоек с UPS на 12 кВт в каждой стойке, коммутатора InfiniBand, коммутатора Ethernet. В каждый узел нужно было дополнительно вставить второй процессор и сетевую карту InfiniBand, поставляемые отдельно от узлов.

Последовательность сборки можно описать следующим образом.

1. Установить в каждую стойку UPS и подключить блоки розеток.
2. Привинтить рельсы для всех узлов к стойке. Нами были заранее определены позиции узлов в стойке, чтобы в дальнейшем их не пришлось бы перемещать выше или ниже.
3. Выделить в комнате место для узлов, в которые уже вставлен процессор и плата InfiniBand.

4. Порядок операций с каждым узлом:
    - положить узел на ровную поверхность;
    - отвинтить крышку;
    - отвинтить заглушку разъёма для карты, вставить и привинтить карту InfiniBand;
    - отвинтить заглушку процессора, достать новый процессор, установить его в разъём, установить и привинтить радиатор;
    - завинтить крышку;
    - привинтить направляющие.
5. Установить узлы в стойки.
6. Установить коммутаторы в стойку.
7. Подключить узлы к UPS.
8. Соединить узлы сервисной сетью.
9. Подключить узлы к коммутатору Ethernet.
10. Подключить узлы к коммутатору InfiniBand.

Обратите внимание, что для каждого узла нам пришлось иметь дело с 19 винтами и целым множеством операций типа открутить/закрутить: 2 винта у крышки узла, 1 винт отвинтить от заглушки платы и завинтить его уже с платой, 2 винта открутить от заглушки процессора и завинтить их с радиатором, 8 винтов ушло на крепление рельсов в стойке, 4 – на крепление направляющих. Если фиксировать узлы в стойке, то нужно добавить ещё 2 винта. Столь детальный подсчет, возможно, и вызвал улыбку, однако из расчёта 80 узлов получим почти 2000 (!) операций завинчивания и отвинчивания. Советуем очень аккуратно сопоставить свои возможности с трудоемкостью всех операций по сборке кластера, что особенно актуально для больших конфигураций. Кластер СКИФ Cyberia – это 283 узла со всеми вытекающими отсюда последствиями для монтажа и сборки.

Операция установки процессора и радиатора очень тонкая, так как легко повредить процессор. Чуть криво поставленный радиатор при закреплении может просто расколоть керамическую основу процессора,

поэтому выполнять эту операцию нужно чрезвычайно аккуратно.

Несмотря на большую площадь помещения, при сборке неожиданно выяснилось, что очень трудно разместить в одном месте все имеющиеся коробки с оборудованием, уже собранные узлы, коробочки с процессорами и платами InfiniBand, а также пустые коробки и образующийся мусор. А расположить все это так, чтобы было еще и удобно работать – почти невозможно.

Для ускорения работы сначала во все узлы были установлены процессоры и платы InfiniBand, а затем все узлы были установлены в стойки. На каждый узел уходило от 15 до 25 минут, поэтому, даже работая вдвоём, на сборку всех узлов ушло более двух рабочих дней. Очень помогло использование аккумуляторных шуруповёртов. Узлы устанавливались вдвоём, для установки узлов в верхней части стоек не лишней оказалась стремянка. Монтаж каждого UPS смогли выполнить только втроём: использованные в нашем случае UPS HP R12000 RX вместе с батареями весили 220 кг каждый. Столь внушительный вес не является уникальной особенностью именно этой модели, источники бесперебойного питания всегда являются одной из самых тяжелых частей кластерной системы. Вес каждого устройства APC Smart-UPS 2200 VA RM в другом проекте составил почти 44 кг.

Схема проводки электричества и коммуникационных сетей была продумана заранее. Силовые электрические кабели и кабели управляющей сети укладывались вдоль противоположных боковых стенок стоек во время прикручивания рельсов в стойки.

Для подведения электричества от UPS к распределительному блоку потребовался кабель, способный длительно выдерживать ток в 50 А. Увы, выяснилось это только в процессе сборки кластера, так как устройство стойки не было изучено заранее. Да к тому же для аккуратной укладки купленного позднее кабеля нужного сечения пришлось открутить какое-то число уже установленных рельсов.

Перед покупкой кабеля рассчитывалось его сечение, для чего воспользовались данными из таблицы 4.1: для трехжильного (земля, ноль, фаза) медного кабеля получили значение 10 мм^2. Заметим, что для проводки внутри стоек необходимо пользоваться последними пятью колонками таблицы.

Питание к самим UPS было подведено заранее с привлечением специалиста. Подключение батарей и блоков логики в HP R12000 делается по строгой схеме, поэтому нельзя просто вставить все блоки внутрь и подключить нагрузку. После установки батарей, была запущена программа самотестирования, которая работала несколько десятков минут. До окончания ее работы никакой нагрузки к UPS подключать нельзя. Читайте инструкции!

Сечение жилы, мм^2	Ток, А, для проводов, проложенных					
	открыто	в одной трубе				
		двух одно-жильных	трех одно-жильных	четырех одно-жильных	одного двух-жильного	одного трех-жильного
1	17	16	15	14	15	14
1,2	20	18	16	15	16	14,5
1,5	23	19	17	16	18	15
2	26	24	22	20	23	19
2,5	30	27	25	25	25	21
3	34	32	28	26	28	24
4	41	38	35	30	32	27
5	46	42	39	34	37	31
6	50	46	42	40	40	34
8	62	54	51	46	48	43
10	**80**	**70**	**60**	**50**	**55**	**50**
16	100	85	80	75	80	70
25	140	115	100	90	100	85
35	170	135	125	115	125	100
50	215	185	170	150	160	135
70	270	225	210	185	195	175
95	330	275	255	225	245	215
120	385	315	290	260	295	250
150	440	360	330	–	–	–

Табл. 4.1. Допустимый длительный ток для проводов и шнуров с медными жилами с резиновой или поливинилхлоридной изоляцией

После узлов в стойки устанавливались коммутаторы InfiniBand и Ethernet. Всё оборудование сразу подключалось к работающим UPS, так как кабели питания были проложены заранее. Далее были проложены кабели InfiniBand и транспортной сети Ethernet.

В каждый узел приходит четыре кабеля: транспортный и сервисный Ethernet, InfiniBand, электропитание. Это означает, что всего было проложено более 320 кабелей, коммутировано более 640 разъёмов. Чтобы элементарно не запутаться в таком количестве разъёмов и кабелей, использовались Ethernet-кабели различного цвета, а также маркировка разъёмов. Для организации кабелей пришлось использовать более 200 пластиковых стяжек.

Такова реальная последовательность шагов, такова примерная трудоемкость. Хотите ли делать это самостоятельно, можете ли сделать это самостоятельно, целесообразно ли делать это самостоятельно или все же имеет смысл обратиться к профессионалам – решайте сами, оценив особенности своего проекта, свои технические, финансовые, кадровые и временны́е ресурсы. Оценивайте не абстрактно, а с точки зрения будущего использования кластера как инструмента решения конкретных задач.

Итак, оборудование собрано, и осталось его оживить. Без необходимого программного обеспечения это всего лишь красивый и внушительный монумент, однако, именно на этой основе будет создаваться полноценный, удобный и функциональный инструмент – вычислительный кластер. В последующих разделах данной работы мы обсудим вопросы установки и настройки операционной системы, систем и сред параллельного программирования, тестирование кластерной системы и определение характеристик ее работы, состав вспомогательных инструментов, вопросы использования специализированных прикладных пакетов.

Глава 5.
Установка, настройка и оптимизация системного программного обеспечения

От вопросов, связанных с аппаратной составляющей кластера, перейдем к базовому и специализированному программному обеспечению. Заметим сразу, что в задачи данного раздела не входит обучение установке стандартных вариантов операционных систем. Это уже описано во множестве книг и предполагается, что читатель обладает необходимыми минимальными навыками и знаниями. Сосредоточимся на “кластерных” особенностях, позволяющих множеству независимых компьютеров согласованно работать в рамках единого комплекса.

Стандартом de-facto **операционной системы** для вычислительных кластеров в настоящее время является Linux. Какой дистрибутив предпочесть? Иногда это решает сам системный администратор, который будет поддерживать работу кластера, в некоторых случаях выбор однозначно определяется прикладным программным обеспечением, оптимизированным под конкретную версию ОС. С точки зрения собственно построения кластера большой разницы в дистрибутивах нет, однако специализированные прикладные пакеты могут оказаться к ним чувствительными. Это же касается и драйверов ко всему набору аппаратного обеспечения, особенно к новым моделям или нестандартному оборудованию.

В последнее время все большую популярность приобретает система Windows Compute Cluster Server 2003, разработанная компанией Microsoft для поддержки кластерных платформ. Вариант интересный, особенно если учесть большой объем прикладного ПО, работающего именно под MS Windows. Его миграция под кластерный вариант этого семейства операционных систем, безусловно, является лишь вопросом

времени. Однако к настоящему моменту во всем мире опыта использования Windows Compute Cluster Server 2003 не много, система только недавно анонсирована, поэтому в данной работе мы будем предполагать, что выбрана ОС семейства Linux.

Существует несколько продуктов, позволяющих провести быструю установку и самую начальную базовую настройку целого кластера. В качестве примеров можно назвать достаточно популярные на практике пакеты Rocks (`http://rocksclusters.org/`) или OSCAR (`http://oscar.openclustergroup.org/`). С их помощью можно установить на сервер и кластерные узлы готовые образы Linux, которые в дальнейшем дополняются необходимыми пакетами и настраиваются на работу в кластере. Стоит отметить дистрибутив Parallel Knoppix (`http://idea.uab.es/mcreel/ParallelKnoppix/`), который нет необходимости устанавливать ни на сервер, ни на узлы, а достаточно просто загрузиться с CD и задать конфигурацию кластера.

Рассматривая возможность использования продуктов подобного рода, нужно четко представлять возможные последствия этого решения. С одной стороны, необходимо понижать трудоемкость сопровождения и администрирования кластерных систем за счет автоматизации рутинных и предписанных регламентом процессов – это правильно, в таком направлении как раз и идет развитие данной области, в частности, для снижения стоимости владения сложными компьютерными системами. Но с другой стороны, кластерная система с самого начала должна быть максимально эффективной по отношению к задачам и оставаться такой все время своего существования. Значительным недостатком упомянутых выше продуктов является то, что в качестве основы коммуникационной среды везде предполагается Ethernet и взаимодействие через TCP/IP, что может самым печальным образом сказаться на эффективности работы кластерной системы в целом. Если для работы в таком режиме система и ставилась, если есть уверенность, что все остается под контролем, или есть время для накопления опыта и экспериментов, то вполне можно

воспользоваться и таким путем.

Установив и настроив обычный вариант операционной системы на головном узле кластера, проведем аналогичную операцию на файловом сервере. Настоятельно рекомендуем использовать Logical Volume Manager для раздела, на котором будут располагаться данные пользователей кластера. Это позволит легко проводить дальнейшее расширение хранилища и безболезненно переносить логический раздел с данными пользователей в будущем.

На головную машину операционная система ставится обычным образом. А вот для того, чтобы **установить ОС на вычислительные узлы кластера**, как правило, требуются дополнительные усилия. С одной стороны, на вычислительных узлах не принято устанавливать привычные в такой ситуации CD или Floppy-приводы, а с другой – установить и настроить ОС на десятках узлов само по себе является занятием долгим и утомительным.

Для упрощения процесса можно воспользоваться виртуальными приводами, если они поддерживаются сервисной сетью. Если нет, то достаточно подключить через USB или напрямую к одному из узлов CD-привод и провести установку операционной системы на этом узле.

Проведя начальную установку и стандартную настройку узла, нужно сделать еще несколько шагов для его дальнейшей работы в составе кластера. Убедитесь, что на узле установлен сервер `ssh`, и что пользователю `root` разрешен удаленный вход. Настройте беспарольный вход на узел для пользователя `root`, используя авторизацию по ключу, для чего воспользуйтесь командой `ssh-keygen`. Необходимо иметь возможность заходить на этот узел привилегированным пользователем `root` с головного узла – это значительно облегчит жизнь в дальнейшем.

Настройте на файловом сервере сетевой каталог, разрешив доступ к нему с вычислительных узлов и головного узла. Это можно сделать, прописав соответствующую строку в файл `/etc/exports` и запустив

сервер NFS. Убедитесь, что сервер NFS стартует автоматически при загрузке файл-сервера.

Крайне желательно добавить в строку экспорта в файле `/etc/exports` опцию `no_root_squash`. По умолчанию NFS отменяет права суперпользователя для удаленных хостов, а указанная опция не позволяет этого сделать.

На вычислительном и головном узлах создайте каталог с одинаковым именем (например, `/common` или даже `/home`) и настройте автоматическое монтирование в него каталога с файл-сервера. Это можно сделать, прописав соответствующую строку в файл `/etc/fstab`. Убедитесь, что каталог монтируется без проблем, до того как будете перезагружать узлы!

Обратите внимание на то, что многие современные дистрибутивы (такие как SuSE, RedHat, Fedora Core и другие) делают жесткую привязку настроек сетевого интерфейса к MAC-адресу сетевой карты. Если просто скопировать такие настройки на другой узел, сетевой интерфейс просто не заработает. Для того чтобы настройки можно было безболезненно перенести на любой узел, необходимо убрать привязку к MAC-адресу из файла `/etc/sysconfig/network/ifcfg-eth-XXXXXX`, если она там есть, переименовать этот файл в `ifcfg-eth0` или `ifcfg-eth1`. Точно также необходимо убрать явные переименования сетевых интерфейсов в подсистеме `udev`. Чтобы найти остальные не столь очевидные привязки к MAC-адресу, поищите все его упоминания командой '`grep -ri MAC /etc`', где `MAC` замените реальным значением MAC-адреса. Узнать его можно командой `ifconfig`.

Синхронизация времени в системе. Она осуществляется с помощью пакета `xntp` или его аналогов. При существенном расхождении часов на различных узлах могут наблюдаться сбои в работе параллельных программ. Необходимо настроить ntp-сервер на головном узле, а на всех остальных узлах – клиентов, которые будут с ним синхронизироваться. При старте все узлы должны явно синхронизироваться с головным узлом

командой `ntpdate` (она обычно входит в состав пакета `xntp`), что необходимо, поскольку при большом расхождении часов клиент `ntp` коррекцию времени может и не выполнить.

Отслеживание состояния UPS. Большинство современных источников бесперебойного питания способны сообщать о своем текущем состоянии через COM-порт, USB или по сети через SNMP. Некоторые производители включают программы под Linux для отслеживания состояния UPS в поставку, но, увы, делают это далеко не все. Существует свободный пакет для мониторинга состояния UPS, который называется NUT. Он поддерживает большое число моделей UPS, но перед покупкой оборудования лучше проверить, поддерживается ли конкретная модель. NUT включает в себя сервер, снимающий данные с UPS, и клиентов. Сервер работает на головном узле, к нему же должен быть подключен управляющий кабель UPS. Клиенты должны быть установлены на все узлы. В случае отключения питания клиенты дадут команду на выключение узлов. Таким образом, можно избежать потери данных и сохранить работоспособность кластера. Если UPS достаточно мощный и может поддерживать работу кластера в течение какого-то времени, то команда выключения узлов может быть настроена на отсроченное выключение. В этом случае, если электропитание восстановится, то выключение будет отменено.

Синхронизация системных файлов (`passwd`, `shadow`, `hosts`, ...). Для того, чтобы системные изменения затрагивали не только головной узел, но весь кластер сразу, можно применять различные схемы. NIS – это одно из самых простых решений, которое, однако, чревато проблемами в случае сетевых сбоев. Следует специально отметить, что, несмотря на “простоту”, хорошая настройка этой схемы может оказаться весьма нетривиальным делом для неискушенного администратора. Использование `rsync` является более простым решением, требующим лишь включения нужного сервиса на всех узлах и начальной настройки на головном узле. Третий вариант можно условно назвать “ручным” копированием, что

предполагает использование `scp` в скрипте для автоматического дублирования всех нужных файлов на узлы. Каждый из перечисленных методов требует явного вызова определенной команды после изменения системных файлов. Есть и другие методы, но все они, в целом, аналогичны `rsync`.

Следующим шагом в настройке программного обеспечения кластерной системы является тиражирование установленной ОС на все остальные узлы. Для этого можно воспользоваться тремя схемами.

- Берем любой доступный компьютер. Вставляем в него жесткий диск из вычислительного узла с уже настроенной ОС и один (или несколько) дисков из других узлов. Затем с помощью команды `dd` копируем на чистые диски содержимое первого диска.

- Можно воспользоваться программой типа Acronis True Image или Norton Ghost для клонирования диска первого узла на диски остальных узлов.

- Можно создать образ установленной ОС с помощью архиватора `tar` или `cpio` (исключите при архивировании содержимое файловых систем `proc`, `sysfs`, `devfs`, `usbfs` и им подобных). Установите `syslinux`. С помощью пакета `syslinux` и серверов `dhcpd` настройте сетевую загрузку. Далее нужно создать сетевой NFS-диск, на котором будет создана минимальная система, достаточная для подготовки жесткого диска (разбиение и форматирование), для разворачивания архива с образом ОС и установки загрузчика. Убедитесь, что ядро данной минимальной системы поддерживает корневой каталог на NFS. Затем скопируйте в каталог сетевого диска образ ОС и в качестве стартового сделайте скрипт, который автоматически установит это образ. После этого произведите сетевую загрузку всех узлов, где операционная система еще не установлена.

Третий способ, конечно, сложнее, но он более универсален и позволяет в дальнейшем быстро добавлять новые узлы и восстанавливать

испорченные простой перезагрузкой (с указанием “грузиться по сети”). Для подготовки минимальной системы, которая будет грузиться по сети и устанавливать ОС на узлы, можно воспользоваться любым мини-дистрибутивом из сети Интернет или подмножеством программ из уже имеющегося дистрибутива.

В минимальной системе обязательно должны присутствовать: `bash`, `tar` (или `cpio`), `sfdisk`, `mke2fs` (`mkreiserfs` или иное в зависимости от выбранной файловой системы), полный пакет `grub` (или `lilo`) и набор библиотек с динамическим линкером, необходимые для работы этих программ.

С помощью программы `sfdisk` можно записать в файл разметку жесткого диска с первого узла и в стартовом скрипте использовать ее для разметки жестких дисков чистых узлов.

О безопасности кластера нужно позаботиться заранее. Не стоит полагаться на соображения типа: “Да кому нужно взламывать наш кластер?”. Будьте уверены, что желающих найдется много. Совсем не обязательно их целью будет помешать вашей работе. Скорее всего, задачей станет использовать взломанные компьютеры как плацдарм для будущих хакерских действий или рассылки спама.

О том, как повысить безопасность Linux-сервера, написано немало книг и статей, желательно с ними ознакомиться. Приведем лишь несколько основных советов.

- Для удаленного входа на кластер и удаленной передачи файлов используйте `ssh`. Под Windows есть немало программ, реализующих этот протокол, например, свободно распространяемая программа `putty`. Не используйте для этих целей `telnet`, `ftp`, `nfs` или `samba` (windows share).
- Отключите все ненужные сервисы.
- Включите файрволл, продумайте политику его использования.

- По возможности, дополнительно ограничьте доступ пользователей. Это можно сделать, задав список адресов, с которых разрешено заходить на головной узел, либо запретив авторизацию по паролю и обязав пользователей использовать для авторизации ключи.
- Включите "устаревание" паролей пользователей.
- Включите проверку сложности паролей.
- Установите одну из программ проверки целостности системы (такую, как `tiger`, `ossec`, `tripware`).
- Регулярно проверяйте журналы на головном узле, воспользуйтесь пакетом `logcheck` или `logwatch`.
- Время от времени проверяйте систему на наличие "закладок" программами типа `chkrootkit`, `rkhunter`. Не храните эти программы в доступном с головного узла каталоге, лучше запускайте их с USB-Flash или с дискеты.

О том, как произвести такие настройки, можно прочесть в man-страничках `chage`, `sshd`, `sshd_config`, `pam`, а также в документации к упомянутым пакетам. Подчеркнем еще раз: обеспечение безопасности – это очень важный вопрос. Сразу уделите безопасности особое внимание, поскольку решение этих проблем после обнаружения факта взлома уже может быть сопряжено с потерями.

В данном разделе мы сразу предположили использование NFS в качестве сетевой файловой системы кластера. Она хорошо известна, у многих есть опыт ее использования, поэтому именно на NFS чаще всего останавливается выбор администраторов. Но хочется предостеречь сразу: этот вариант, во-первых, не единственный и, во-вторых, совсем не идеальный. Основное слабое место связано с плохой масштабируемостью NFS и очень сильным падением характеристик ее работы при увеличении числа узлов. Какова цель кластерного проекта и чего хотелось бы достичь?

Максимум процессоров, максимум “флопсов”? Бывает и такое. В частности, именно такие требования выдвигают некоторые масштабные задачи физики высоких энергий, для которых чем больше в компьютерной системе вычислительных узлов, тем лучше. Если говорить про общий случай, то каждое приложение устанавливает некоторый порог, соотношение между вычислительными способностями кластера и характеристиками подсистем ввода/вывода, за которым выполнение приложения перестает быть эффективным, а использование кластера становится полностью нецелесообразным. Для параллельных приложений порог может меняться в зависимости от числа используемых процессоров, что создает дополнительные трудности в его определении на практике. Но сделать это необходимо, причем сделать на этапе проектирования архитектуры, чтобы вместо сбалансированной кластерной системы не получить однобоко развитого монстра. Здесь уместно вспомнить определение суперкомпьютера, приведенное в предисловии данной книги, согласно которому в системах подобного класса “проблема вычислений сводится к проблеме ввода/вывода”...

Альтернативные варианты файловых систем, к которым стоит приглядеться: Panasas File System, Lustre, Terragrid, Parallel Virtual File System (PVFS2), General Parallel File System (GPFS). Данные файловые системы изначально предназначались для параллельных компьютеров, они активно развиваются и реально используются на многих больших кластерных системах. Имеет смысл подумать об этих вариантах. Сделать “как все” не означает принять оптимальное для себя решение: не исключено, что именно данные файловые системы лучше всего подойдут для решения задач проекта.

Итак, на все узлы кластера установлена операционная система, проведена начальная настройка, кластер “задышал”. Следующим шагом будет переход к содержательной работе кластера и запуск параллельных программ. Для этого нам нужен набор компиляторов, одна или несколько

сред параллельного программирования, дополнительные библиотеки и пакеты, специализированные прикладные системы.

Глава 6.
Средства разработки и прикладное программное обеспечение

Набор компиляторов GNU (`gcc/g77`) доступен бесплатно для всех основных аппаратных платформ, на которых строятся кластеры. Однако по производительности эти компиляторы нередко отстают от коммерческих вариантов – Intel C/Fortran Compiler, PGI Compiler, PathScale EcoPath, Absoft и других. Все эти продукты можно запросить у производителей и протестировать бесплатно на своей конкретной задаче и платформе, а после чего уже решать, стоит ли покупать и переходить на один из них. Если кластер строится на основе многоядерных процессоров, то обратите внимание, поддерживается ли многоядерность в выбранном компиляторе, в частности, есть ли поддержка технологии OpenMP.

Если принято решение установить коммерческий компилятор, то перед заказом **внимательно изучите условия лицензирования**. Одни лицензии позволяют запустить не более, чем оговоренное в лицензии число процессов компиляции одновременно, другие варианты ограничивают круг пользователей, которым разрешена компиляция.

Компилятор достаточно установить только на головном узле и покупать лицензии на все узлы не нужно. Если компилятор использует свои динамические библиотеки (так это делает, например, компилятор компании Intel), то их надо скопировать на вычислительные узлы и прописать к ним пути в файле `/etc/ld.so.conf`, после чего запустить на узлах команду `ldconfig`. Заметим, что это надо сделать обязательно, так как при запуске параллельных программ на узлах не отрабатывают скрипты типа `/etc/profile` или `~/.bash_profile`, а значит и не будут прописаны пути к библиотекам через переменную `LD_LIBRARY_PATH`, как это делается в штатных скриптах, поставляемых с большинством компиляторов.

Вопрос с лицензиями на программное обеспечение нужно изучить очень аккуратно, поскольку он тесно связан с планированием бюджета всего проекта. Существует несколько видов лицензий для программ и программных пакетов для кластеров, причем нередко один и тот же пакет может поставляться с различными вариантами лицензий, учитывающих:

- число процессоров или узлов,
- фиксированный круг пользователей и рабочих мест,
- число одновременно запущенных копий программного продукта.

Как мы уже обсуждали, для компилятора не нужно покупать лицензии на все узлы кластера. Если людей, занимающихся разработкой программ, немного и их круг не будет меняться ближайший год-два, то можно купить лицензию с фиксированным числом пользователей. Если такой уверенности нет, то подумайте о варианте лицензии на одновременную компиляцию одним-двумя пользователями. Такой вариант на практике используется достаточно часто.

Для специализированных параллельных вычислительных пакетов или библиотек может потребоваться лицензия, в которой нужно будет явно указать максимальное число процессоров, доступных для работы пакета. Стоимость лицензии для больших конфигураций может оказаться внушительной: десятки и сотни тысяч долларов, поэтому еще раз советуем внимательно изучить условия лицензирования всех коммерческих пакетов, планируемых к использованию. Нужно найти оптимальный компромисс между удобством работы пользователей, эффективностью решения задач кластерного проекта и стоимостью устанавливаемого ПО.

Если говорить о средах параллельного программирования, то на практике это, как правило, различные варианты MPI. Не рекомендуем устанавливать параллельную среду из пакетов, поставляемых вместе с дистрибутивом Linux. Как правило, эти варианты плохо оптимизированы, и настроить их на эффективную работу с кластером очень непросто. Детали и особенности установки наиболее популярных реализаций MPI приведены в **Приложении 2**.

Проведя установку параллельной среды, убедитесь в том, что она работоспособна. Для этого воспользуйтесь стандартными тестами, поставляемыми практически со всеми дистрибутивами MPI. Скомпилируйте командой `make` простейшую программу, которой обычно является программа `cpi.c` параллельного вычисления числа π. Затем скопируйте ее в каталог на сетевом диске и перейдите в него. Теперь можно выполнить команду `mpirun -np N ./cpi`, где `N` – это число процессоров, на которых будет запускаться задача.

Если все прошло успешно, нужно провести более серьезную проверку – тестирование производительности. Сюда входит тестирование скорости передачи данных и величины латентности коммуникационной сети, а также определение производительности всего кластера на тестах типа Linpack или HPCC.

Скорость передачи данных и латентность можно измерить пакетом `mpi-benchsuite`, доступным по адресу:

`http://parallel.ru/ftp/tests/mpi-bench-suite.zip`

Распакуйте архив, исправьте файл конфигурации `config/make.def`, если это нужно, и соберите исполняемые файлы командой `make`. В каталоге `bin` появятся файлы отдельных тестов `transf1`, ..., `transf5`, `mpitest`, `nfstest` и `nettest`. Сейчас для нас наиболее интересны `transf1` (блокирующая передача данных) и `mpitest` (тестирование основных операций MPI).

Для тестирования скорости передачи воспользуйтесь `transf1` на двух узлах. Используйте параметры запуска "`m1 M1024000 T5`". Параметр `m1` предписывает начать замеры с сообщения длиной в 1 байт, `M1024000` – остановиться на длине сообщений в 1024000 байт, `T5` – повторить каждую передачу 5 раз (для набора статистики и повышения точности измерения). Более подробно все параметры описаны в документации к данному пакету.

Результаты будут выданы на экран, где будет показана скорость передачи данных по коммуникационной сети (в Мбайт/с). Можно посмотреть динамику изменения скорости в зависимости от длины

сообщения, что будет полезно в дальнейшем при проектировании новых или адаптации существующих параллельных приложений на данном кластере.

Латентность, возникающая при передаче сообщений, измеряется этим же тестом `transf1` с параметрами "`m0 M0 T10`".

Хорошим вариантом тестирования производительности системы является использование пакета Intel MPI Benchmarks, ранее известного под названием Pallas MPI Benchmarks. Пакет измеряет базовые характеристики кластерной системы, а также показывает эффективность основных функций MPI, предназначенных для реализации коллективных операций и передачи сообщений между двумя процессами, односторонних операций, функций для выполнения операций ввода/вывода и ряда других.

Некоторое представление о работе кластера в целом даст измерение производительности его работы на тесте Linpack. В принципе, это отдельная большая тема, обсуждать которую можно очень долго. Сам тест является программой решения системы линейных уравнений, поэтому если что-то подобное является вычислительным ядром большинства будущих приложений, то некоторую оценку эффективности их работы на кластере получить можно. Чаще всего данный тест используют для того, чтобы убедиться в отсутствии явных проблем в работе аппаратуры и настройках программного обеспечения кластера. Для сборки теста под конкретную платформу потребуется хорошо оптимизированная библиотека BLAS и собственно исходные тексты теста High Performance Linpack (`http://www.netlib.org/benchmark/hpl/`). Все эти компоненты необходимо скомпилировать, используя установленную библиотеку MPI, и провести серию запусков с различными входными параметрами, которые и покажут максимально достигаемую на данном тесте производительность.

Более подробное описание процесса сборки и запуска теста Linpack представлено в **Приложении 3**. Если хочется быстро получить представление о работе кластерной системы, то можно воспользоваться пакетом HPL, уже собранным с библиотекой MKL:

http://www3.intel.com/cd/software/products/asmo-na/eng/perflib/mkl/266857.htm.

Заметим, что работать он будет только на процессорах Intel. В этом же пакете есть программа nodeperf, которую желательно запустить перед самим тестом для предварительной оценки производительности узлов на операции типа DGEMM.

Неплохим показателем работы кластера на множестве разных приложений могут служить данные, полученные на тестах пакета HPC Challenge Benchmark, (HPCC, http://icl.cs.utk.edu/hpcc/). Пакет объединяет несколько приложений с разными свойствами, что позволяет выполнить более детальный анализ эффективности работы кластера. В частности, в состав пакета входит тест Linpack и несколько программ для определения базовых характеристик коммуникационной среды. Тест DGEMM (умножение матриц) показывает скорость выполнения операций с плавающей запятой над данными с двойной точностью. Тест STREAM измеряет соответствие между скоростью работы процессора и скоростью извлечения данных из памяти. Всего в состав HPCC в настоящее время включено семь различных наборов тестов.

Если планируется использовать специализированные прикладные пакеты и программы, то имеет смысл провести серию дополнительных тестовых испытаний. Практически каждый пакет содержит в своем дистрибутиве набор тестовых задач, проверяющих и правильность его установки, и эффективность его работы в конкретной программно-аппаратной среде. Запустите тесты и сравните с эталонными результатами. Если обнаружилось неожиданное расхождение, то необходимо провести дополнительное исследование и выяснить причину, прежде чем отдавать инструментарий в эксплуатацию в штатном режиме.

Кстати, отметим интересный и немаловажный факт: всю работу по тестированию кластерной системы, начиная от базового уровня до проверки эффективности работы сложных прикладных пакетов, может (а иногда и должен) выполнить поставщик оборудования. Просто заложите

требование предоставления документального подтверждения достигнутых показателей эффективности работы необходимого программного обеспечения в условия тендера или контракта на поставку, и часть забот по оптимизации настроек кластерной системы перейдет на поставщика. Шаг достаточно очевидный, но редко используемый на практике.

Если кластерная система будет интенсивно использоваться для создания нового программного обеспечения, то очень важно предоставить набор эффективных инструментальных средств разработки. Компиляторы, отладчики, профилировщики, трассировщики, анализаторы и системы исследования специальных свойств программ – все это в какой-то момент потребуется и будет с благодарностью воспринято пользователями. Разработка параллельного программного обеспечения является делом непростым и требует целого множества вспомогательных инструментов.

Хорошую инфраструктуру для поддержки разработки ПО можно создать на основе **программных инструментов Intel**, многие из которых доступны как под ОС семейства Linux, так и под ОС MS Windows:

- компиляторы,
- анализаторы производительности,
- специализированные библиотеки,
- инструменты для многопоточного программирования,
- инструменты программирования для кластеров.

Компиляторы Intel (языки C/C++, Fortran77/Fortran90) поддерживают и различные уровни оптимизации для 32-х и 64-х разрядных приложений в одном пакете, и технологию параллельного программирования OpenMP, что позволяет создавать эффективные программы для современных многоядерных процессоров. С компиляторами поставляется символьный отладчик Intel Debugger, который может работать в режимах совместимости с gdb или dbx и интегрируется с такими графическими оболочками для отладки, как ddd, Eclipse, Allinea. Отладчиком поддерживаются как многонитевые приложения OpenMP, так и написанные с использованием интерфейса

native threads. Порожденные нити автоматически попадают под контроль отладчика, причем большинство его команд можно применять либо к одной, либо ко всем нитям одновременно.

Анализатор Intel VTune позволяет находить и устранять узкие места, препятствующие повышению производительности программы, за счет сбора, анализа и отображения данных вплоть до отдельных функций, модулей и команд. Замеряя производительность различных участков кода и помогая интерпретировать результаты замеров, VTune позволяет быстро найти участки для оптимизации. Поддерживаются многонитевые приложения для различных операционных систем и разных сред разработки.

Библиотека Intel Integrated Performance Primitives предоставляет набор оптимизированных подпрограмм, которые могут быть использованы для обработки аудио- и видеоданных, распознавания речи, кодирования JPEG и видео, сжатия данных, специальной обработки матриц, в решении задач криптографии, векторной алгебры и обработки сигналов.

Библиотека Intel Math Kernel Library широко используется для решения вычислительно сложных задач, где от платформ Intel требуется максимальная производительность. К функциональным возможностям этой библиотеки можно отнести модули линейной алгебры (BLAS, Sparse BLAS, LAPACK и пакет Sparse Solvers), функции быстрых преобразований Фурье (FFT), векторные математические функции (VML), генераторы случайных чисел.

Анализатор Intel Thread Checker позволяет упростить разработку и сопровождение многопоточных приложений. В паре с Intel Thread Profiler он призван решить большинство проблем, связанных с многопоточностью. Выявляет недетерминировано работающие участки кода, являющиеся причиной трудно находимых “плавающих” ошибок. Определяет ситуации неопределённости порядка записи данных, потерянные нити, блокировки, потерю сигналов, неверно отмененные блокировки. Поддерживает стандарты OpenMP, POSIX и Windows API. Вообще говоря, Intel Thread

Checker не зависит от компилятора, способен работать с любым исполняемым файлом, но в сочетании с компилятором Intel проводит более глубокий анализ.

Инструменты для программирования кластеров объединены в пакет Intel Cluster Tools. Сюда входит библиотека Intel MPI, оптимизированная параллельная математическая библиотека Intel Cluster MKL и специальный инструмент Intel Trace Analyzer & Collector, предназначенный для создания эффективных масштабируемых параллельных программ.

Не обязательно использовать все указанные выше технологии разработки параллельного программного обеспечения, создавая самостоятельно весь параллельный код. Часто на практике прикладные программисты вообще не используют никаких явных параллельных конструкций, обращаясь в критических по времени счета фрагментах к подпрограммам и функциям параллельных предметных библиотек. Весь параллелизм и вся оптимизация спрятаны в вызовах, а пользователю остается лишь написать внешнюю часть своей программы и грамотно воспользоваться стандартными блоками. Примерами подобных библиотек являются Lapack, ScaLapack, Cray Scientific Library, HP Mathematical Library, PETSc, MKL и многие другие.

И, наконец, одно из самых важных направлений – это использование специализированных пакетов и программных комплексов из конкретных прикладных областей. Как правило, в этом случае пользователю вообще не приходится программировать. Основная задача – это правильно указать все необходимые входные данные и правильно воспользоваться функциональностью пакета. Главное, чтобы пакет с нужной функциональностью уже был создан и был бы доступен для использования на кластере. Так, многие конструкторы для выполнения специализированных инженерных расчетов на параллельных компьютерах пользуются пакетом LS-DYNA, не очень задумываясь о том, каким образом реализована параллельная обработка данных в самом пакете. Примеры

наиболее известных инженерных пакетов, используемых для решения реальных промышленных задач, приведены в **приложении 5**. Важно то, что и предметная сторона пакета, и его параллельная реализация уже выполнены профессионалами, а пользователям остается воспользоваться результатами их работы для эффективного решения своих задач.

Проведите исследование потребностей пользователей, определитесь с набором необходимого прикладного ПО, выберите оптимальную форму оформления лицензий, установите и проверьте эффективность работы прикладных библиотек и пакетов, подготовьте и распространите информацию о доступном наборе программных средств на кластере среди пользователей.

Если пользователей кластера будет больше одного или надо будет запускать целые серии расчетов, то запуск задач вручную станет непростым делом. Задачи начнут конкурировать друг с другом за процессоры, память и другие общие ресурсы, и об эффективности выполнения программ нужно будет забыть.

В такой ситуации необходима система управления заданиями, распределяющая множество задач по процессорам. В настоящее время наиболее распространены следующие системы: Torque/OpenPBS, LSF (коммерческая), OpenPBS PRO (коммерческая), LoadLeveler (на серверах IBM), Sun Grid Engine, Cleo (описана в **Приложении 4**). Распределяя программы пользователей по узлам кластера, система управления заданиями должна поддерживать согласованную работу с некоторой **системой квотирования и бюджетирования кластерных ресурсов**.

На практике, это относится, в первую очередь, к контролю за использованием выделенного объема процессорного времени и занимаемого места на дисках. Вроде бы рутина и текучка, однако вопросы эффективной организации коллективной работы пользователей исключительно важны. Если это не предписано соображениями приоритетности, то не должно складываться ситуации, когда одна или несколько задач пользователя надолго занимают все ресурсы кластера, не

позволяя более никому выполнить даже небольшой тестовый расчет. На практике, кластер часто разбивается на несколько независимых логических разделов, в каждом из которых устанавливаются свои ограничения и политики: максимальное время выполнения программ, число занимаемых процессоров работающими приложениями пользователя, число одновременно запущенных программ пользователей и другие.

В такой структуре легко предусмотреть отдельный раздел для программ с небольшим временем работы, что помогает ускорить процесс отладки кластерных приложений. Разбиение на разделы, во многих случаях, можно сделать с помощью самих систем управления заданиями.

Убедившись в успешном прохождении всех этапов, описанных в предыдущих главах книги, администратор заканчивает стартовый период своей деятельности и открывает пользователям доступ на кластер. Начинается их работа.

Глава 7.
Пользователь в кластерной среде

Создание кластерной системы – это не цель, а только шаг к решению реальных задач. После установки узлов, настройки ОС, средств удаленного доступа и параллельного программирования, кластер готов к работе. В принципе, пользователи уже могут заходить на него со своих рабочих мест, компилировать и запускать программы. Но вот будут ли они его использовать? Окажется ли данный инструмент удобным и эффективным, станет ли он востребованным вычислительным сообществом? Сложные вопросы, но от них в конечном итоге и зависит успех проекта в целом. Программа-минимум предполагает создание комфортной и полноценной среды для работы пользователей, установки набора вспомогательных программ и систем. Развернутый вариант включает профессиональный консультационный центр, создание службы поддержки работы пользователей, систему их обучения и многие другие шаги. Если использование кластерной системы стало неотъемлемой частью работы пользователей, если кластер прочно занял место в цепочке получения новых результатов, то только в этом случае проект можно назвать успешным. Попробуем наметить некоторые шаги в этом направлении.

Вопросов у пользователей возникнет много, и к этому нужно приготовиться сразу. Будут вопросы “периода роста”, как правило, они простые и быстро заканчиваются. Сложнее отвечать на содержательные вопросы, связанные с тонкими характеристиками работы параллельных программ. Нам приходилось работать в рамках многих проектов на большом числе параллельных компьютеров, и практически всегда перед нами стояла задача – помощь в создании действительно эффективных параллельных программ. Интересно то, что почти всегда все множество

потенциальных проблем и вопросов пользователей сводилось к двум формулировкам: “Что-то не так с моей программой” либо “Это плохой компьютер, предыдущий был лучше”. Как мы увидим, жизнь намного богаче, чем столь категоричные заявления, просто истинная причина низкой эффективности не так очевидна и кроется где-то в другом месте.

Иногда проблема снималась легко – достаточно было, например, разобраться в документации по работе с системой или компилятором. В некоторых случаях приходилось довольно глубоко вникать в суть решаемой задачи. Чем дольше мы работаем в этой области, тем отчетливей проявляется многогранность исходной проблемы. Как и для любых параллельных компьютеров, речь должна идти о **создании целостной инфраструктуры, сопровождающей большинство прикладных работ на кластерных системах**.

Давайте посмотрим, на какие составные части может разбиваться исходная проблема пользователя, и почему нужен комплексный подход к анализу ситуации в каждом конкретном случае. Практика показала, что если “с программой что-то не так”, то вопросы могут возникать на самых разных уровнях:

АНАЛИЗ КОНФИГУРАЦИИ КОМПЬЮТЕРА
↓
АНАЛИЗ ЭФФЕКТИВНОСТИ СИСТЕМНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
↓
АНАЛИЗ СТРУКТУРЫ ПРИКЛАДНОЙ ПРОГРАММЫ
↓
АНАЛИЗ АЛГОРИТМИЧЕСКОГО ПОДХОДА

Начинать исследование приходилось с анализа конфигурации конкретного компьютера: тип и число процессоров, уровни и объем памяти, параметры и топология коммуникационной среды, особенности

организации ввода/вывода и т.п. Даже эти, казалось бы, простые и очевидные понятия, на практике могут оказаться причиной вопросов и недовольства пользователей.

Сильно увлеченный прикладной областью пользователь до поры до времени может и не знать, что число процессоров для запуска пакета, если не задано явно, по умолчанию берется из конфигурационного файла. Запускает задачу, получает результат, все нормально. Перешел на другой кластер, запускает такой же вариант, время получения ответа увеличилось в два раза, хотя “в программе он ничего не менял, а процессоры такие же”. Все верно, и программа не испортилась, и новый кластер не хуже, разница лишь в числе процессоров, используемых пакетом по умолчанию на каждой из этих кластерных систем.

Похожие вопросы вызывает изменение структуры памяти, сильно влияющей на эффективность работы программ: объем и частота работы оперативной памяти, количество уровней кэш-памяти и объем каждого уровня, возможность разделения какого-либо уровня несколькими ядрами или процессорами.

Еще одна ситуация. Человек приходит и говорит, что его программа стала “неожиданно” работать медленнее, почему? Стали разбираться, и оказалось, что раньше он работал на кластере, у которого было по два жестких диска на узел, а у нового кластера на каждом узле установлено лишь по одному. На всех узлах кластеров по два процессора, все процессы приложения исключительно интенсивно работают с локальными дисками. На узлах первого кластера потоки ввода/вывода от двух процессоров бесконфликтно разводились по разным дискам, а на втором кластере единственный жесткий диск на каждом узле становился узким местом и сдерживал работу процессоров. Пользователь знал об этой особенности своей программы, но на конфигурацию кластеров не обратил внимания.

Другой пользователь, но проблема та же: медленная работа программы. Стали разбираться, и оказалось, что человек изменил всего

лишь один параметр программы, но не учел, что при этом значительно увеличились области памяти, выделяемые под массивы данных. В результате программа перестала помещаться в оперативную память, появился отсутствующий прежде свопинг, из-за чего время работы программы выросло на порядок.

Почему латентность в моей программе получается намного выше той, что мне обещали, с моей программой что-то не так? Оказалось, что пользователь ориентировался на характеристики коммуникационной сети нового поколения, которая уже была анонсирована, но реально еще не установлена. С программой было “все так”, и в рамках существующей конфигурации она работала идеально. Одновременно заметим, что негативного осадка этот вопрос не оставил, напротив, порадовал весьма высокий уровень подготовки пользователя, который отслеживает столь тонкие параметры и может задавать такие вопросы.

Обсуждая подобные случаи, мы не ставим своей целью обвинить пользователя в некомпетентности, в нежелании или неспособности разобраться с возникающими проблемами. Задача обеспечения эффективного выполнения программ сложна и многогранна, в расчет нужно принимать весьма тонкие свойства программно-аппаратной среды кластера, для исследования которых у пользователя просто может и не быть нужного инструментария. Посмотрим на данные мониторинга динамики выполнения параллельной программы, показанные на рис. 7.1.

Классическая ситуация. Сначала задача на всех узлах считается с максимальной эффективностью: левые области всех картинок верхнего ряда закрашены полностью, что говорит об использовании процессора на 100%. Затем в некоторый момент эффективность резко падает. Начинаем исследовать причины: одновременно с падением эффективности видим усиление свопинга (третий ряд сверху) на фоне отсутствия каких-либо иных процессов, которые могли бы мешать работе программы (второй ряд сверху, значение параметра `loadaverage` равно 2, т.е. числу процессоров на узле) и отсутствия обменов в сети (нижний ряд). После выполненного

анализа становится понятным источник проблем: недостаток оперативной памяти.

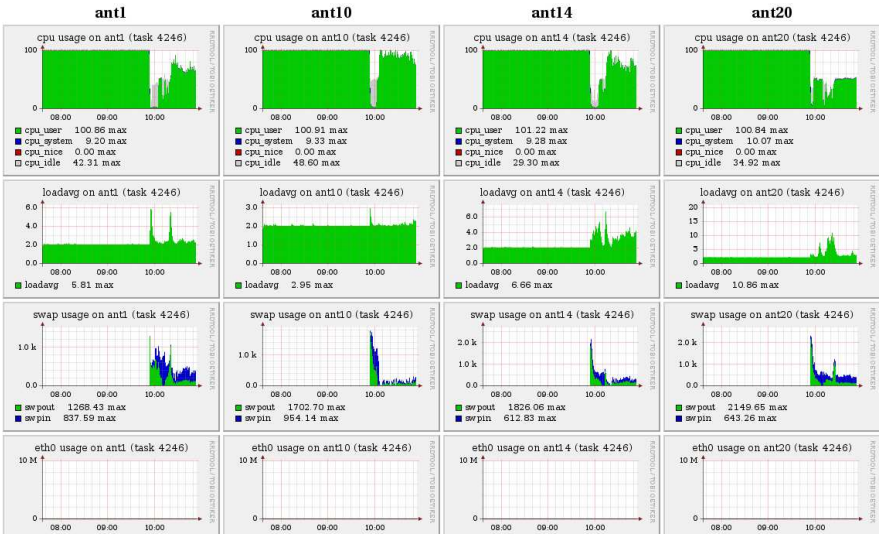

Рис. 7.1. Снижение эффективности выполнения программы из-за недостатка оперативной памяти и активизации свопинга

Увы, несмотря на очевидную полезность, с такими данными, как правило, работает только системный администратор, пользователю они редко бывают доступны. Рассмотрим еще два примера использования данных системного мониторинга характеристик программно-аппаратной среды кластера.

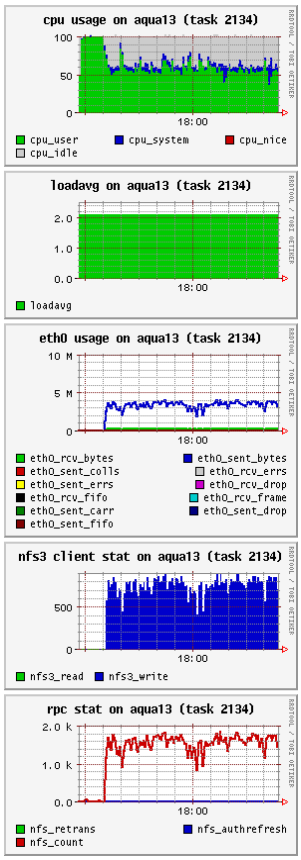

Рис. 7.2. Снижение эффективности работы программы из-за регулярного появления посторонних процессов

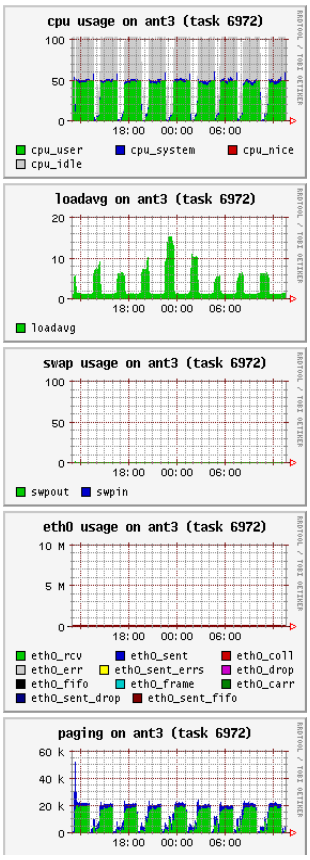

Рис. 7.3. Падение эффективности выполнения программы из-за медленной работы сетевого диска

Весьма поучительный пример для администраторов кластерных систем показан на рис. 7.2. Из него ясно следует, что эффективность работы программ определяется не только квалификацией пользователей, но и аккуратной настройкой системного окружения администраторами. Более того, пользователям разобраться самостоятельно в ситуациях подобного рода почти невозможно. По серому цвету верхней половины верхней диаграммы видно, что двухпроцессорный узел используется лишь наполовину, один процессор все время простаивает – на узле выполняется однопроцессорная задача. Работа другого процессора (нижняя часть верхней диаграммы) регулярно прерывается, причем с той же периодичностью резко возрастает как значение параметра `loadaverage` (вторая диаграмма сверху), так и активность обмена страницами оперативной памяти (нижняя диаграмма) при отсутствии сколько-нибудь заметного использования области свопинга (вторая диаграмма снизу).

В системе явно порождаются новые более приоритетные процессы, захватывающие процессор на значительное время. Детальный разбор показал, что причиной такой ситуации стала ошибка в одной строке таблицы программы `cron`, в результате которой трудоемкая системная операция, обычно запускаемая раз в неделю, выполнялась намного чаще.

Падение эффективности работы программы, которое показано на рис. 7.3, вызвано другими причинами. С параметром `loadaverage` никаких проблем нет, его значение всегда около 2. Зато резко возрастает активность работы с сетевым диском (вторая диаграмма снизу, `nfs_client`), и пользовательский процесс все время проводит в ожидании завершения операций записи на диск. Кстати, плохо организованная работа с сетевыми дисками очень часто является причиной низкой эффективности программ. Причем слова “плохо организованная работа” могут относиться и к архитекторам кластерного проекта, и к администраторам, и к пользователям. Архитекторы могли бы учесть специфику будущих задач и предусмотреть для передачи файлов, скажем, не Gigabit Ethernet, а

подумать об использовании намного более эффективных решений, например, на основе Panasas. Администраторы могли бы проверить эффективность установки NFS на своей системе, воспользовавшись уже существующим набором тестов, и изменить значения ключевых параметров, выставляемых автоматически по умолчанию в некоторые средние значения. Пользователь вполне мог бы задуматься о том, что на кластере есть как сетевые, так и локальные диски.

Сетевые диски разделяются между всеми пользователями, доступны через сетевую файловую систему и, как правило, физически расположены на отдельных устройствах. Все это приводит к дополнительным задержкам, и работа с локальными дисками вычислительных узлов проходит намного быстрее. В нашей практике был случай, когда в программе изменили лишь путь к каталогу для размещения временных файлов, за счет чего время работы программы уменьшилось в 15 раз: исключили взаимодействие с сетевым диском, а всю активность перенесли на локальные диски узлов.

Следующий большой срез – это анализ эффективности системного и прикладного программного обеспечения. Отчасти мы его уже затронули, обсуждая аккуратную настройку сетевой файловой системы, но спектр возможных вопросов намного шире. У многих в процессе работы на компьютере были нарекания на работу штатных компиляторов, что можно отнести к проблемам этого же уровня. Просмотрите документацию к компилятору, попробуйте подобрать оптимальный набор ключей и опций для его работы на данной платформе, а если есть такая возможность, то поэкспериментируйте с альтернативными вариантами компиляторов. Очень полезно посоветоваться с системными администраторами, которые помогут решить и ряд смежных вопросов. Один из пользователей самостоятельно установил в свой домашний каталог прикладной пакет, который по умолчанию вызывал стандартный, но совсем не оптимизированный вариант MPI. Естественно, что характеристики работы программы были ужасны, однако ситуация была полностью исправлена за

три минуты внесением одной строки в конфигурационный файл пакета. Проблема с программой? Да нет, опять-таки не в самой программе дело... Компьютер плох? И этого нельзя сказать.

Быстро найти причину бед пользователей удается далеко не всегда, иногда приходится проводить аккуратный детальный анализ характеристик программного окружения. В одном проекте при переходе на новый компьютер программа явно замедлилась, хотя никаких изменений в ее текст не вносилось. Причина была найдена достаточно быстро: чрезмерно большие задержки при выполнении операции барьерной синхронизации процессов. Почему? Оказалось, что на систему поставили экспериментальный вариант реализации MPI, в которой каждый процесс, доходя до точки синхронизации, выполнял следующие действия. Если он оказывался последним, и все ждали только его, то процесс рассылал всем уведомление о своем приходе и продолжал выполнение дальше. В противном случае он “засыпал” на достаточно продолжительный промежуток времени, после которого проверял, двигаться ли дальше или опять “заснуть”. Время на “сон” процессам было отведено достаточно большим, и ничего хорошего от такой реализации нельзя ожидать.

Важно осознать и то, что решить самостоятельно проблемы этого уровня пользователь, как правило, не может: знаний-то может и хватает, но прав что-либо менять в установках системы маловато. Здесь он должен работать в тесном контакте с системными администраторами, которые, в свою очередь, должны внимательно прислушиваться к вопросам и пожеланиям своих пользователей.

Основная задача в рамках анализа структуры прикладной программы состоит в поиске ответа на вопрос: “Можно ли не меняя алгоритма улучшить эффективность работы программы?”. Алгоритм может обладать всеми нужными свойствами, в частности достаточной степенью параллелизма или локальностью использования данных, однако форма записи программы может скрыть эти особенности, и компилятор не сможет сгенерировать эффективный код. Нужно изменить структуру программы,

сделать ее особенности явными и видимыми для компилятора, для чего в распоряжении есть целый арсенал приемов и методов эквивалентного преобразования программ. Занимаясь исследованиями в данном направлении в течение двух десятков лет, мы разработали и используем на практике комбинацию методов статического и динамического анализа, реализованных в экспериментальной системе исследования структуры программ V-Ray. Эффективных вариантов решения проблем подобного сорта может быть много, в конечном итоге все зависит от личного опыта, предпочтений, наработанных технологий, сложившихся правил по разработке больших программных комплексов авторскими коллективами.

Проведите простой эксперимент. Возьмите программу перемножения двух плотных квадратных матриц `A=B*C` размера, скажем, `N=5000`, примерно такого вида:

```
DO I = 1, N
  DO J = 1, N
    DO K = 1, N
      A(I,J) = A(I,J) + B(I,K) * C(K,J)
```

Информационная структура данного фрагмента позволяет выбрать любой порядок следования данных трех циклов, на правильность результата работы программы это никак не повлияет. Однако выбранный порядок сильно повлияет на время ее работы. Запустите шесть различных вариантов программы, определяемых возможными комбинациями циклов на компьютере с любым из процессоров Intel, AMD, Power, PA-RISC, и замерьте время работы. Получите шесть разных значений, причем минимальное время будет отличаться от максимального в несколько (!) раз. Слова “программа примерно такого вида” означают, что совершенно не важно, на каком языке будет записана программа: C, Pascal, Fortran или на каком-то ином, подобный эффект будет наблюдаться для каждого из них.

И, наконец, если анализ приложения показал, что структура программы не соответствует особенностям архитектуры компьютера, то проблемы эффективности исходной программы кроются в свойствах используемых алгоритмов. Единственное, что остается – это перейти к

алгоритмическому анализу. Возможно, что в результате исследований этого этапа пользователю придется поменять алгоритм и полностью переписать некоторые части программы. В ряде случаев другого пути просто нет, к этому нужно быть готовым. В идеальном варианте **разработка программы для кластера должна сразу проходить с учетом его архитектуры**, тогда недоразумений со свойствами алгоритмов не возникнет.

Как мы видим, проблема анализа и повышения эффективности работы параллельных программ является комплексной. Именно поэтому мы говорим не о каком-то одном средстве, системе или методе для ее решения, а о целой инфраструктуре. В полной мере это касается и всего спектра программного обеспечения. Понимая сложность проблемы и острую необходимость в разного рода вспомогательных инструментах, к настоящему моменту разработано немало полезных систем, входящих в состав штатного программного обеспечения компьютеров. Компиляторы, отладчики, анализаторы, конверторы, профилировщики – всем этим богатством нужно уметь пользоваться, и нужно учить пользоваться, если есть желание получать реальный эффект от уникальных возможностей современных кластерных систем. Насколько это реально, насколько развит программный сервис на доступном кластере – это вопрос к сервисной службе и администраторам каждой конкретной кластерной системы.

Глава 8.
Сопровождение кластерных систем

Кластер построен, установлен и настроен. Как поддерживать его работоспособность и как решать возникающие вопросы? **Необходим квалифицированный обслуживающий персонал**. Нужно сразу отбросить иллюзии и четко осознать, что если нет администратора, отвечающего за работу кластера, то нет и самого кластера. Никакой серьезной работы на такой системе выполнить будет нельзя. В нашей практике был пример, когда купленный кластер проработал месяц, но из-за отсутствия администратора и постоянных споров между пользователями был попросту выключен: кто не мог, а кто не хотел брать на себя ответственность за его работоспособность. В таком состоянии кластер простоял два года. Естественно возникает вопрос, зачем покупали или же почему потом не продали? Не все поддается разумному объяснению…

В зависимости от масштабности проекта в состав обслуживающего персонала могут входить следующие категории специалистов:

- системный администратор, отвечающий за настройки кластера, обеспечение его безопасности и текущую работу с пользователями,
- инженер по сопровождению силовой электрики и систем климат-контроля,
- специалист по системному программному обеспечению,
- консультант, способный решать проблемы с прикладным программным обеспечением и помогающий пользователям в работе на кластерной системе.

Желательно иметь штатных операторов, которые могут оперативно решать текущие вопросы. Естественно, что все специалисты

должны иметь соответствующую квалификацию. Персонал в целом должен быстро решать задачи, связанные с безопасностью и администрированием, управлением пользователями и программным обеспечением, разбираться с возникающими аппаратными проблемами, связываться с поставщиками и сервисными службами.

Операторы и администраторы должны быть хорошо подготовлены, и **процесс начального обучения персонала необходимо закончить до того, как кластер будет введен в эксплуатацию**. Минимальный уровень знаний администратора должен включать знание всех используемых на кластере программных пакетов и хорошее понимание их взаимосвязей. Это не предполагает знание тонкостей программирования в математических, физических, химических и других специализированных прикладных комплексах, но предполагает понимание механизмов их установки и настройки.

Кроме подготовки администраторов, **необходима подготовка пользователей**. Пользователи кластера должны хорошо уметь работать с командной строкой UNIX, знать основные технологические приемы, уметь пользоваться компиляторами и программами для удаленного доступа, знать как передать и получить файлы с кластера, уметь запускать параллельные программы, владеть азами параллельных вычислений и разбираться в технологиях параллельного программирования. Признаком не только хорошего тона, но и показателем отношения к пользователям является организация on-line ресурса, посвященного организации работ на кластере. Такой ресурс поможет не только новичку быстро войти в курс дела, он будет полезен и опытному человеку, ранее работавшему в другом окружении. Регламент работы системы в целом, состав программного обеспечения, описание политик и квот в различных разделах кластера, новости, примеры программ, документация, учебные материалы, советы и описание типичных ситуаций, вопросы и ответы – все это, как масса другой полезной информации может быть размещено на подобных страницах.

Говоря о персонале и пользователях, нельзя не затронуть вопрос их взаимоотношений. Сложный вопрос. В некоторых случаях администраторы вводят жесткие меры, заставляя пользователей выполнять чрезмерно строгие правила, ходить на цыпочках и записываться к ним на прием заранее в случае возникновения каких-либо вопросов. В других местах ситуация диаметрально противоположная: пользователи исполнены ощущением собственной значимости, считая невозможным или ненужным общение с администраторами в силу своей высокой миссии. И в том, и в другом случае хороший результат маловероятен. Выяснять кто главнее: администратор или пользователь, столь же бессмысленно, как пытаться отдать первенство пилоту или болиду в гонках Формулы-1. Только вместе, помогая и подсказывая друг другу.

Оставим пафос и вернемся к технической стороне дела. Для повышения эффективности работы системного администратора очень рекомендуем установить пакеты для **мониторинга состояния кластера**. Важно не только уметь получать данные о текущем состоянии кластера и истории его функционирования, но и оперативно реагировать на возникающие нештатные ситуации.

Не все пакеты одинаково хороши. Например, пакет Ganglia (`http://ganglia.sourceforge.net/`) позволяет получать наглядную информацию об истории состояния кластера, но абсолютно не способен ее анализировать.

Пакет Nagios (`http://nagios.org/`) ориентирован на мониторинг доступности компьютеров и сервисов. Он может быть весьма полезен для отслеживания работоспособности кластера и оповещения в случае серьезных сбоев. Однако, он не способен отслеживать оперативную картину загрузки и использования кластера без серьезных дополнительных нагрузок на систему.

Постоянно следите за состоянием жестких дисков и вентиляторов, которые по статистике чаще всего выходят из строя. В этом сильно могут помочь пакеты `smartd` и `lm_sensors`.

Чтобы понимать, насколько эффективно используются ресурсы кластера, не пренебрегайте статистикой. В частности, в пакет Cleo входит программа `cleo-stat`, позволяющая получить информацию об использовании ресурсов кластера как в целом, так и с детализацией по пользователям за любой период времени.

Чтобы не пришлось часто разбираться с множеством мелких проблем на вычислительных узлах, постарайтесь минимизировать эти проблемы заранее. В этом может помочь пакет `logcheck`, а также периодическая (например, с помощью утилиты `cron`) очистка временных каталогов. В частности, команда

```
find /tmp -mtime +3 -exec rm -f \{\} \;
```

удалит из `/tmp` все файлы и каталоги, которые не изменялись за последние трое суток.

Важная деталь, о которой нередко забывают в начале, а потом регулярно откладывают на “завтра” – это **резервное копирование**. О важности сохранения данных говорить излишне. Надеяться на аппаратный RAID можно, но это не панацея. Из-за нелепой ошибки или программного сбоя можно потерять важные данные на логическом уровне, т.е. на уровне файловой системы. Тут никакой RAID не спасет.

Эффективная организация резервного копирования подробно описана во многих специальных изданиях по администрированию, обсуждать этот вопрос в деталях здесь мы не будем. Можно использовать архиваторы `tar` или `cpio`. Или даже `zip` или `rar`, однако имейте в виду, что эти утилиты не сохраняют UNIX-атрибуты файлов, поэтому восстанавливать данные из этих архивов будет сложнее. Можно воспользоваться многочисленными системами для автоматизации резервного копирования, такими как Bacula, Amanda, Backupninja, Cdbackup, af-backup, ibackup, Taper и многими другими.

Производить резервное копирование можно как вручную, так и в автоматическом режиме. Если ведется инкрементальное копирование, при котором сохраняются лишь изменения со времени последнего

копирования, то хотя бы раз в две недели надо делать полный сброс информации. В противном случае, на восстановление будет уходить очень много времени, да и надежность копирования снизится.

Ни в коем случае не делайте резервной копии на том же физическом или логическом диске, что и оригинал. Лучший вариант – это делать ее на другом компьютере либо воспользоваться для копирования ленточным устройством или пишущим приводом CD/DVD. Хранить только одну резервную копию тоже плохо: если были потеряны данные, и такое состояние попало на резервную копию, то восстановить пропавшие файлы не получится.

Часто принимают схему, при которой полные резервные копии делаются каждые одну-две недели (лучше всего во время выходных дней, когда пользователи работают не так активно), а инкрементальные выполняются ежедневно. Если нет специальных требований, то хранить резервные копии имеет смысл в течение периода от месяца до полугода. Вариантов организации резервного копирования и в самом деле существует много, но самое главное – помнить о его необходимости и поставить этот процесс на регулярную основу.

Постарайтесь максимально автоматизировать все процессы и процедуры, связанные с сопровождением и администрированием кластера. Кластерное хозяйство является сложным и динамичным, важных вопросов, требующих личного участия администраторов будет много, поэтому все то, что может выполняться в автоматическом или автоматизированном режиме, так и должно быть настроено. Это в полной мере касается только что обсуждавшихся систем мониторинга и резервного копирования, за которыми должен быть оставлен лишь минимальный “ручной” контроль, позволяющий быть уверенным в их нормальном функционировании. Грамотное использование подобных технологий позволяет повысить надежность и эффективность работы кластера, и одновременно, что также немаловажно, понизить такой важный показатель, как стоимость владения кластерной системой.

Если речь идет о долгосрочной эксплуатации кластерной системы, а на практике так чаще всего и бывает, то нужно позаботиться о выделении специальной строки в бюджете организации. Какие-то расходы незаметны, например, замена одного локального диска на узле, однако в целом текущие расходы могут составить весьма значительную сумму. Оплата электроэнергии и аренды помещения, продление лицензий на компиляторы, прикладные пакеты, расширение набора используемого программного обеспечения, оплата сервисного обслуживания кластера, систем электропитания и охлаждения, замена вышедшего из строя оборудования – все подобные детали должны приниматься в расчет, чтобы организовать эффективную текущую эксплуатацию кластера и минимизировать перерывы в его работе.

Чтобы пользователи могли планировать свою деятельность, заранее определите и объявите **план проведения профилактических работ**. Именно в это время имеет смысл вносить изменения в настройки работающих программ, устанавливать новые или обновлять уже существующие пакеты, выполнять какие-то действия по резервному копированию, установке нового оборудования. Ко времени профилактики разумно привязать и многие обязательные регламентные работы, например, очистку фильтров систем кондиционирования или проведение контрольного тестирования компонентов системы. Как правило, многие текущие дела по сопровождению кластера можно немного отложить и затем спокойно выполнить во время профилактики: это и администраторам проще, и пользователям спокойнее.

В целом же, чем четче, понятнее и прозрачнее будет организована работа на вычислительной системе, тем дольше и эффективнее она будет использоваться.

Заключение

Итак, мы добрались до конца книги. Сложных, не сразу заметных, но принципиально важных вопросов, возникающих в процессе работы на высокопроизводительных кластерных вычислительных системах, оказалось много. Как много и подходящих технологий, на основе которых может формироваться программно-аппаратная среда каждого нового кластера. В одних случаях выбор технологии определяется точным расчетом и аккуратным анализом особенностей и задач конкретного проекта, иногда на первый план выходят привычки, личные симпатии и пристрастия либо же просто ориентация на общепринятое мнение, склоняющие специалистов к тому или иному решению. В данной книге мы не пытались научить технологическим основам инженерного искусства сборки и конструирования кластеров. Это вопрос, безусловно, важный, и он должен являться предметом отдельного исследования. Однако не этот вопрос был для нас определяющим. Современные технологии быстро меняются, одно поколение процессоров, сетей, интерфейсов, устройств ввода/вывода через год-два сменяется другим, более совершенным и производительным. Но сохраняется главное – необходимость соблюдения четкого соответствия принимаемых на всех этапах технологических, инженерных и административных решений тем задачам, которые будут решаться на кластере. Как, увы, сохраняются и многие ошибки. Значительная их часть повторяется из раза в раз, из года в год, они с удивительным постоянством кочуют из проекта в проект вне зависимости от выбранных технологий. Подсказать, предостеречь, что-то упростить, а какой-то процесс ускорить, чтобы в итоге создать комфортную для пользователей среду и быстрее перейти к решению задач – в этом состояла наша основная задача.

Разработка кластерного проекта в целом, определение нюансов архитектуры, заказ, размещение, монтаж, настройка и тестирование

кластера, его поддержка и эксплуатация – на всех этих этапах встречается целое множество подводных камней. Одни замечаешь сразу, другие становятся заметными лишь по истечении какого-то времени, но в любом случае столкновение с ними положительных эмоций не добавляет. Мы очень надеемся, что данная книга послужит Вам своего рода лоцией, помогающей своевременно заметить опасные места, обойти стороной кажущиеся простыми и привлекательными, но на самом деле коварные решения, и в результате найти тот путь, который приведет к построению эффективно работающей кластерной системы. Ваш собственный путь, выбранный осознанно с четким пониманием задач Вашего кластерного проекта.

А младший брат из эпиграфа пусть делает так, как ему нравится...

Приложение 1.
Примеры существующих кластерных проектов

Данное приложение содержит описание нескольких существующих кластерных установок, установленных в 2003-2007 годах в разных организациях с использованием различных технологий. Описание конфигураций в каждом случае сделано достаточно подробным и представлено по одной и той же схеме, чтобы можно было бы сравнить различные стороны каждого кластерного проекта. Дополнительную информацию, технические детали и контактную информацию по многим кластерным проектам можно найти на страницах информационно-аналитического центра Parallel.ru и сайте списка Top50 самых мощных компьютеров СНГ `http://www.supercomputers.ru`. Свои комментарии, дополнения и описания новых кластерных систем присылайте нам по адресу `ClusterBook@Parallel.ru`.

Место установки / Условное название кластера	**г. Москва, НИВЦ МГУ / Leo**
Год установки кластера	2003
Количество узлов в кластере	16
Количество процессоров на узел	2
Тип процессоров	Intel Xeon 2,6 ГГц
Объём оперативной памяти на узел	2 ГБ
Тип коммуникационной сети	SCI
Марка сетевых карт и коммутатора	SCI D335
Тип транспортной сети	100Mbit Ethernet
Тип сервисной сети	ServNet2
Количество жёстких дисков на узел	2
Форм-фактор узлов	MidiTower
Есть ли отдельный управляющий узел? Форм-фактор?	Да, Tower
Объём общего файлового хранилища	1,5 ТБ
Тип общей файловой системы	NFS
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Нет
Размеры и число стоек	Металлические стеллажи
Используется ли UPS?	Да
Суммарная потребляемая мощность кластера	3,8 кВт
Тип системы кондиционирования	Потолочные кондиционеры
Общая мощность системы	70 кВт

кондиционирования	
Площадь помещения	Около 80 кв.м.
Высота потолков в помещении	Около 5 м.
Используется ли помещение в иных целях? В каких?	Нет
Средства мониторинга состояния помещения	Аппаратный мониторинг температуры и влажности помещения с программным контролем
Средства мониторинга параметров кластера	Температура процессоров и скорости вращения вентиляторов через пакет lm_sensors с оповещением
Версия ОС	Linux RedHat 7.3
Версия параллельной среды	Scali SSP 3.1.0
Версии компиляторов	Intel C+Fortran 9.1
Дополнительное прикладное ПО	Intel MKL 8.0.2
Система управления очередями	Cleo5
Тип доступа и способ авторизации пользователей	Удалённый доступ по ssh
Производительность пиковая / HPL	166,4 / 104,3 Гфлопс

Место установки / Условное название кластера	**г. Москва, НИВЦ МГУ / Ant**
Год установки кластера	2004
Количество узлов в кластере	80
Количество процессоров на узел	2
Тип процессоров	AMD Opteron 248 2,2 ГГц
Объём оперативной памяти на узел	4 ГБ
Тип коммуникационной сети	InfiniBand
Марка сетевых карт и коммутатора	Mellanox MT23108, коммутатор Mellanox MTS3100
Тип транспортной сети	Gigabit Ethernet
Тип сервисной сети	HP LightsOut100
Количество жёстких дисков на узел	2
Форм-фактор узлов	1U
Есть ли отдельный управляющий узел? Форм-фактор?	Да, 4U
Объём общего файлового хранилища	1,5 ТБ
Тип общей файловой системы	NFS
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Нет
Размеры и число стоек	4 стойки 42U, 19”
Используется ли UPS?	Да
Суммарная потребляемая мощность кластера	40 кВт

Тип системы кондиционирования	Потолочные кондиционеры
Общая мощность системы кондиционирования	70 кВт
Площадь помещения	Около 80 кв.м.
Высота потолков в помещении	Около 5 м.
Используется ли помещение в иных целях? В каких?	Нет
Средства мониторинга состояния помещения	Аппаратный мониторинг температуры и влажности помещения с программным контролем
Средства мониторинга параметров кластера	Состояние жёстких дисков через пакет hdparm с оповещением
Версия ОС	SuSE Linux 10.0
Версия параллельной среды	mvapich-0.9.5-mlx2.0.1
Версии компиляторов	Intel C+Fortran 9.1
Дополнительное прикладное ПО	Intel MKL 8.0.2
Система управления очередями	Cleo5
Тип доступа и способ авторизации пользователей	Удалённый доступ по ssh
Производительность пиковая / HPL	704 / 512 Гфлопс

Место установки / Условное название кластера	**г. Челябинск, ЮУрГУ / Infinity**
Год установки кластера	2005
Количество узлов в кластере	26
Количество процессоров на узел	2
Тип процессоров	Intel Xeon EM64T 3,2 ГГц
Объем оперативной памяти на узел	2 ГБ
Тип коммуникационной сети	InfiniBand
Марка сетевых карт и коммутатора	Mellanox IB
Тип транспортной сети	Gigabit Ethernet
Тип сервисной сети	Нет
Количество жестких дисков на узел	1
Форм-фактор узлов	4U
Есть ли отдельный управляющий узел? Форм-фактор?	Да, 4U
Объём общего файлового хранилища	400 ГБ
Тип общей файловой системы	NFS
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Да (для некоторых задач)
Размеры и число стоек	3 стойки APC 42U, 19”
Используется ли UPS?	Только на управляющем узле
Суммарная потребляемая мощность кластера	10 кВт
Тип системы кондиционирования	Настенно-потолочный кондиционер

Общая мощность системы кондиционирования	17,6 КВт по охлаждению
Площадь помещения	28 кв.м.
Высота потолков в помещении	3,5 м.
Используется ли помещение в иных целях? В каких?	Нет
Средства мониторинга состояния помещения	Нет
Средства мониторинга параметров кластера	Нет
Версия ОС	SUSE Linux Enterprise Server 9
Версия параллельной среды	mvapich
Версии компиляторов	Intel C/C++ 9.1 EM64T Intel Fortran 9.1 EM64T Intel Fortran 77 9.1 EM64T
Дополнительное прикладное ПО	LSTC LS-DYNA, ANSYS CFX 10, ANSYS
Система управления очередями	Cleo5
Тип доступа и способ авторизации пользователей	Удаленный доступ по ssh
Производительность пиковая / HPL	333,2 / 270,1 Гфлопс

Место установки / Условное название кластера	**г. Черноголовка, ИПХФ РАН / Cli**
Год установки кластера	2005
Количество узлов в кластере	4
Количество процессоров на узел	4
Тип процессоров	Intel Itanium2 1,5 ГГц / 400 МГц
Объём оперативной памяти на узел	20 ГБ
Тип коммуникационной сети	1 Gbit Ethernet
Марка сетевых карт и коммутатора	Intel 82546-3x1Gbit PCI-X; 3Com 3C16470
Тип транспортной сети	Gigabit Ethernet
Тип сервисной сети	Нет
Количество жёстких дисков на узел	3
Форм-фактор узлов	4U
Есть ли отдельный управляющий узел? Форм-фактор?	Нет
Объём общего файлового хранилища	146 ГБ
Тип общей файловой системы	NFS
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Нет
Размеры и число стоек	19” шкаф PrimeCenter, глубина 1 м.
Используется ли UPS?	Да
Суммарная потребляемая мощность кластера	5,0 кВт

Тип системы кондиционирования	Настенные кондиционеры
Общая мощность системы кондиционирования	25 кВт
Площадь помещения	Около 70 кв.м.
Высота потолков в помещении	Около 4 м.
Используется ли помещение в иных целях? В каких?	Размещены два кластера
Средства мониторинга состояния помещения	Мониторинг температуры и влажности помещения
Средства мониторинга параметров кластера	Нет
Версия ОС	RedHat Enterprise Linux AS 4.2 for IA64
Версия параллельной среды	MPICH 1.2.7
Версии компиляторов	Intel C+9.0, Intel Fortran 9.0
Дополнительное прикладное ПО	Intel MKL 7.2.1
Система управления очередями	PBS Pro Altair 7.0
Тип доступа и способ авторизации пользователей	Удалённый доступ по telnet из локальной сети Института
Производительность пиковая / HPL	96 / 68,06 Гфлопс

Место установки / Условное название кластера	**г. Новосибирск, ИВМиМГ СО РАН / НКС-160**
Год установки кластера	2006
Количество узлов в кластере	40
Количество процессоров на узел	2
Тип процессоров	Intel Itanium 2 1,6 ГГц, 3 MB SLC
Объём оперативной памяти на узел	4 ГБ
Тип коммуникационной сети	InfiniBand
Марка сетевых карт и коммутатора	Voltaire HCA 400/Voltaire ISR 9024
Тип транспортной сети	Gigabit Ethernet
Тип сервисной сети	Fast Ethernet
Количество жёстких дисков на узел	1
Форм-фактор узлов	1U (HP Integrity rx1620)
Есть ли отдельный управляющий узел? Форм-фактор?	Да, 2U (HP Integrity rx2600)
Объём общего файлового хранилища	0,8 ТБ (дополнительно)
Тип общей файловой системы	NFS
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Да
Размеры и число стоек	2 стойки 42U, 19”
Используется ли UPS?	Да
Суммарная потребляемая мощность кластера	25 кВт
Тип системы кондиционирования	Air Blue (Италия)

Общая мощность системы кондиционирования	2 контура по 32 кВт
Площадь помещения	70 кв.м.
Высота потолков в помещении	3,2 м, фальшпол, фальшпотолок.
Используется ли помещение в иных целях? В каких?	Нет
Средства мониторинга состояния помещения	Охранная сигнализация. Автоматическая система аэрозольного пожаротушения.
Средства мониторинга параметров кластера	Ganglia, HP Systems Insight Manager
Версия ОС	Red Hat Enterprise Linux ES release 4
Версия параллельной среды	mvapich
Версии компиляторов	Intel C++ / Fortran 9.1
Дополнительное прикладное ПО	Intel MKL 9.0, Fluent 6.2
Система управления очередями	Grid Engine
Тип доступа и способ авторизации пользователей	Удалённый доступ по ssh
Производительность пиковая / HPL	512 / 404 Гфлопс

Место установки / Условное название кластера	**г. Томск, ТГУ / СКИФ Cyberia**
Год установки кластера	2007
Количество узлов в кластере	283
Количество процессоров на узел	2
Тип процессоров	Intel Xeon 5150, 2,66 ГГц
Объём оперативной памяти на узел	4 ГБ
Тип коммуникационной сети	InfiniBand
Марка сетевых карт и коммутатора	QLogic InfiniPath, Silverstorm InfinI09240
Тип транспортной сети	Gigabit Ethernet
Тип управляющей сети	ServNet
Количество жёстких дисков на узел	1
Форм-фактор узлов	1U
Есть ли отдельный управляющий узел? Форм-фактор?	Да, 2U
Объём общего файлового хранилища	10 ТБ
Тип общей файловой системы	PanFS (Panasas)
Является ли файловое хранилище одновременно и головной машиной или вычислительным узлом?	Нет
Размеры и число стоек	14 стоек 42U, 19”: 8 под вычислитель, 6 под систему бесперебойного электропитания
Используется ли UPS?	Да

Суммарная потребляемая мощность кластера	115 кВт
Тип системы кондиционирования	модульная внутрирядная
Общая мощность системы кондиционирования	96 кВт
Площадь помещения	72 кв.м.
Высота потолков в помещении	4
Используется ли помещение в иных целях? В каких?	Нет
Средства мониторинга состояния помещения	APC InfraStruXure
Средства мониторинга параметров кластера	Ganglia
Версия ОС	SuSe Linux Enterprise Server 10, Microsoft Windows Compute Cluster Server 2003
Версия параллельной среды	QLogic MPI
Версии компиляторов	gcc 4.1.0, Intel FORTRAN compiler 9.1, Intel C/C++ compiler 9.1
Дополнительное прикладное ПО	Intel MKL, Fluent 6.3, Gambit 2.3
Система управления очередями	Torque
Тип доступа и способ авторизации пользователей	ssh2
Производительность пиковая / HPL	12 / 9 Тфлопс

Приложение 2.
Среды параллельного программирования MPI

В данном приложении будут рассмотрены некоторые вопросы, связанные с установкой и использованием пакетов mpich, mpich2, Intel MPI Library, mvapich, ScaMPI и IBGOLD. Найти дистрибутивы этих пакетов в сети не представляется сложным – достаточно набрать название в любой поисковой системе, например, google, yandex или rambler.

Существующие на сегодня различные варианты реализации MPI не ограничиваются только теми, которые обсуждаются в данном приложении. На практике широко используются такие пакеты, как LAM, OpenMPI, mpich-gm, MPI/PRO, NT-mpich и некоторые другие. Для большей части из них процедура установки и настройки почти полностью совпадает с пакетами, приведенными в данном разделе, а незначительные отличия можно легко найти из описания, которое есть практически во всех дистрибутивах.

Еще раз повторим, что не рекомендуется использовать готовые пакеты mpich из дистрибутивов операционных систем, так как настроить их сложнее, а качество сборки не всегда удовлетворительное.

mpich

Идеология mpich предполагает, что обмен сообщениями будет идти через некое абстрактное устройство `device`. При сборке необходимо явно указать, какое устройство будет использовано. Наиболее часто используется устройство `ch_p4`, которое позволяет обмениваться сообщениями через TCP/IP. Запуск процессов задачи на узлах происходит через `rsh` или `ssh`.

Вариантом устройства ch_p4 является ch_p4mpd. В этом варианте для запуска программы на узлах и контроля запущенных процессов, в принципе, могут использоваться специальные программы-мониторы mpd. Увы, качество работы этого варианта оставляет желать лучшего, поэтому мы не рекомендуем его использовать.

Из других устройств можно отметить ch_shmem, при использовании которого обмен будет происходить только через общую память. Для кластера использовать не рекомендуется.

Для сборки mpich, распакуйте дистрибутив командой

```
tar xfvz mpich-1.X.X.tar.gz
```

Войдите в созданный каталог и наберите

```
./configure --prefix=/opt/mpich -with-device=ch_p4
```

Вместо /opt/mpich можно указать иной каталог, например, /usr/local. Если предполагается использование компиляторов, отличных от gcc/g77, то укажите к ним пути:

```
-c++=CXXPATH -cc=CCPATH -fc=FCPATH -f90=F90PATH\
-clinker=CLINK -c++linker=CXXLINK\
-flinker=FLINK -f90linker=F90LINK
```

Здесь CXXPATH, CCPATH, FCPATH и F90PATH – пути к компиляторам C++, C, Фортран и Фортран-90 соответственно, а CLINK, CXXLINK, FLINK и F90LINK – пути к соответствующим линкерам. Например, в случае с компилятором Intel нужно указать опции:

```
-c++=icc -cc=icc -fc=ifort -f90=ifort -clinker=icpc
-c++linker=icpc -flinker=ifort -f90linker=ifort
```

Для задания дополнительных ключей компиляторам можно указать опции -cflags= , -c++flags= , -fflags= , -f90flags.

Если нужно явно указать программу, которая будет использоваться для доступа на узлы (по умолчанию это /usr/bin/ssh), то задайте ключ rsh=RSHCOMMAND.

Если configure отработает без ошибок, можно запустить команду make. В противном случае постарайтесь разобраться, почему не отработала configure. Например, это может быть вызвано отсутствием каких-либо

компиляторов или библиотек. Отключить ненужные компиляторы можно опциями

`--disable-cxx, --disable-f77, --disable-f90.`

После успешной отработки `make`, наберите `make install` для установки всего программного комплекса.

Теперь можно компилировать программы командами `mpicc` (Си), `mpicxx` (Си++), `mpif77` (Фортран) и `mpif90` (Фортран-90). Откомпилированные программы можно запускать командой `mpirun`. Например, следующая команда

```
mpirun -np 10 ./a.out
```

запустит программу `a.out` из текущего каталога на 10 процессорах. На каких именно процессорах? Чтобы указать это явно, потребуется создать файл со списком узлов (по одному узлу в строке файла) или исправить глобальный список `$INSTALL_DIR/share/machinefile.LINUX`. Если создается отдельный файл, то его надо указать команде `mpirun` с помощью ключа `-machinefile`.

Не забудьте о том, что сама параллельная программа и ее входные файлы должны быть расположены на сетевом диске либо скопированы на вычислительные узлы.

mpich2

Процесс сборки этого программного комплекса в целом аналогичен сборке mpich, но есть и некоторые отличия, на которых мы остановимся.

- По умолчанию привязки к Си++, Фортрану и Фортрану-90 отключены, чтобы их добавить, используйте опции
  `--enable-f77, --enable-f90 и --enable-cxx.`
- Устройства в mpich2 отличаются от устройств в mpich. По умолчанию используется устройство `c3`, которое, как и `ch_p4` использует передачу сообщений через TCP/IP. Используя опции

`--with-device=c3:shm` или `--with-device=c3:ssm`, можно указать, что общаться надо через общую память или через общую память на SMP и через TCP/IP на разных машинах, соответственно.

- В mpich2 есть встроенная поддержка InfiniBand (на данный момент только через Mellanox Verbs API), включить ее можно следующим набором ключей:
  `--with-device=ch3:ib -with-ib=vapi`
  `--with-ib-path=PATH_TO_MELLANOX_IB`.
- Можно использовать поддержу Myrinet, используя драйвер GM и библиотеку GASNet. Для этого надо указать опции
  `--with-device=ch3:gasnet -with-gasnet-conduit=gm`
  `--with-gm=GM_INSTALL_DIR`.

Перед началом работы пользователь должен создать в домашнем каталоге файл `.mpd.conf`, содержащий строку '`secretword=<pass>`', где `<pass>` – это некоторое "секретное" слово (ни в коем случае не пароль в систему!). После этого надо установить права доступа на него командой: `chmod 600 .mpd.conf`.

В mpich2 программы-мониторы `mpd` используются в обязательном порядке. Для запуска мониторов необходимо создать файл `/etc/mpd.hosts` со списком всех узлов. Затем надо запустить команду `mpdboot -n N`, где `N` – число узлов+1 (один `mpd` должен работать на головном узле). Можно также запускать `mpd` вручную или автоматически на каждом узле. Проверить работоспособность мониторов можно командой `mpdtrace`.

Теперь уже можно запускать задачи, что делается либо старой программой `mpirun`, либо с помощью новой программы `mpiexec`. Список узлов команде `mpiexec` можно передать ключом `-machinefile mfile`. В файле `mfile` должны быть перечислены узлы в формате: `<node:ncpu>`, где `node` – имя узла, `ncpu` – число процессоров на нем. Ключом `-np N` указывается общее число процессоров. Альтернативный способ указать

узлы для запуска – это указать ключи `-hosts N host1 ... hostN`, где `N` равно числу процессов, а `host1 ... hostN` описывают нужные узлы.

Intel MPI Library

Данная реализация MPI основана на mpich2. Её ключевой особенностью является прозрачная поддержка различных коммуникационных сред. Это значит, что в среде с гетерогенной коммуникационной средой можно будет запускать параллельное приложение даже без перекомпиляции. Например, к кластеру на InfiniBand можно подключить ещё несколько узлов через Ethernet или Myrinet и приложения сразу смогут их использовать.

Это достигается за счёт использования динамического выбора типа коммуникации. Доступные варианты: через общую память, через Ethernet или через любое устройство, поддерживающее RDMA (Remote Direct Memory Access) через библиотеку DAPL. Такая библиотека поставляется с большинством высокоскоростного оборудования, такого как InfiniBand и Myrinet. Несмотря на возможность динамической поддержки работы приложения с различными коммуникационными сетями, одновременное использование процессоров с различным набором команд не допускается.

Intel MPI Library полностью реализует стандарт MPI-2 и поддерживает работу программ на архитектурах x86, x86_64 и процессорах семейства Itanium. Поддерживаются компиляторы Intel C/C++/F77/F90 версий 8.1 и выше, а также любые компиляторы, совместимые с форматом GNU.

Установка производится запуском скрипта `install.sh` из каталога, где был распакован архив с дистрибутивом. Для успешной инсталляции необходимо иметь лицензию, которую для удобства установки пакета нужно сохранить в файле. В процессе работы инсталлятор попросит указать каталог для установки (по умолчанию это `/opt/intel/mpi/версия`), а также путь к файлу с лицензией и набор

компонент для установки. Для успешной работы необходимо наличие на узлах и сервере интерпретатора python версии не ниже 2.2.

После установки Intel MPI Library команды компиляции и запуска будут доступны только с указанием полных путей, что, конечно, неудобно. Для комфортной работы на 32-битных процессорах необходимо выполнить скрипт `<путь_к_intelmpi>/bin/mpivars.sh`, если вы работаете в bash или zsh или `<путь_к_intelmpi>/bin/mpivars.csh`, если вы работаете в csh или tcsh. На 64-битных процессорах вместо `<путь_к_intelmpi>/bin/` пишите `<путь_к_intelmpi>/bin64/`. Чтобы не выполнять эти действия каждый раз, пропишите их запуск в файлах `/etc/profile` и/или `/etc/csh.cshrc` соответственно. Во многих дистрибутивах Linux есть каталог `/etc/profile.d`, в котором хранится набор скриптов, выполняющихся bash и tcsh при запуске. Вы можете не «захламлять» `/etc/profile` и `/etc/csh.cshrc`, а просто сделать жёсткие ссылки указанных выше скриптов в этот каталог. Символические ссылки, скорее всего, не сработают в силу специфической обработки каталога `/etc/profile.d`.

Компиляция программ производится обычными скриптами `mpicc/mpiCC/mpif77/mpif90`. Для запуска программ необходимо предварительно создать файл `$HOME/.mpd.conf`, с правами на чтение-запись владельцу и запрещением всего остальным (`0600`). В этом файле необходимо прописать строку '`secretword=<mpd_secret_word>`', где `mpi_secret_word` – это произвольный набор символов. Не пишите туда свой пароль! Убедитесь, что файл доступен на всех узлах кластера. Создайте файл `mpd.hosts` со списком всех узлов кластера, по одному в каждой строке.

Перед запуском пользовательской программы запустите команду `mpdallexit`, а затем `mpdboot -n <Nodes_Number>`, где `Nodes_Number` – число узлов, перечисленных в `mpd.hosts`. Для работы с узлами должен быть настроен беспарольный доступ по `rsh` или `ssh`. Если используется `ssh`, то `mpdboot` надо запускать с ключом `-r ssh`.

Собственно запуск программ пользователей осуществляется командой `mpiexec -n <Nodes_Number> <путь_к_программе>`. Если по какой-то причине нужно использовать определённую среду, то можно запретить автоматический выбор, указав `mpiexec` ключ `-genv I_MPI_DEVICE <device>`, где параметр `device` может принимать значения: `rdma` (использовать библиотеку DAPL), `shm` (общая память), `sock` (TCP/IP через Ethernet), `ssm` (`socket+shm`) или `rdssm` (`socket+shm+rdma`). Если среда была указана явно, то смена среды в динамике становится невозможной.

Дополнительная информация по Intel MPI Library доступна в Интернет на странице `http://www.intel.com/go/mpi`.

mvapich

Несмотря на то, что этот программный комплекс в свое время выделился из mpich, процесс его установки значительно отличается. Мы будем описывать версию gen2, как наиболее перспективную. mvapich ориентирован на работу с InfiniBand и его основное устройство (в терминах mpich) называется `ch_gen2`.

Для установки комплекса потребуются драйвер InfiniBand и библиотека `vapi`. Процесс установки начинается запуском скрипта `mvapich.make.gcc`. Предусмотрены варианты для компилятора Intel: `mvapich.make.icc` и `mvapich.make.ecc`, а также для компилятора Portland Group – `mvapich.make.pgi`. Можно воспользоваться и ручным методом с запуском `configure`, но в этом случае надо внимательно почитать руководство.

После завершения настройки можно запустить команду `make`, и, если сборка прошла успешно, то `make install`. Так как программы по умолчанию будут собираться с динамическими библиотеками mvapich, то нужно скопировать каталог с динамическими библиотеками на все узлы. После этого, необходимо на всех узлах дописать путь к этим библиотекам в файл `/etc/ld.so.conf` и выполнить команду `ldconfig`.

Программа `mpirun` будет присутствовать в числе установленных, но работать она не будет. Запуск программ осуществляется командой `mpirun_rsh`. Синтаксис запуска:

```
mpirun_rsh -np N host1 ... hostN program args...
```

где `N` – число процессов, `host1 ... hostN` – имена узлов, а `program` и `args` – запускаемая программа и ее аргументы соответственно.

ScaMPI (Scali MPI Connect)

Этот пакет разработан компанией Scali (`http://www.scali.no`) и поддерживает Ethernet, Myrinet, SCI и InfiniBand. Пакет является коммерческим. Процесс установки полностью автоматизирован, администратору надо лишь ответить на несколько вопросов в процессе установки. Это единственное хорошо работающее решение для сети SCI.

IBGOLD/OpenFabrics

Пакет IBGold был изначально разработан компанией Mellanox и распространялся бесплатно. В последствии развитие и поддержка IBGold была прекращена, а все исходные тексты переданы группе разработчиков OpenFabrics. В данный момент этой группой разработан пакет `OFED`, который включает в себя базовые драйверы к картам InfiniBand, обширный набор библиотек, open subnet manager и реализацию MPI на базе mvapich.

Установка пакета автоматизирована, но только для одного узла. Чтобы установить `OFED` на все узлы, следует запустить скрипт `install.sh` и выбрать пункт «Build OFED Software RPMs». Затем все пакеты из каталога `RPMS`, не содержащие в имени суффикса `-devel`, кроме пакета `opensm`, необходимо установить на всех узлах кластера. Не исключено, что драйверы InfiniBand из пакета `kernel-ib` будут конфликтовать с уже установленными драйверами штатного ядра. В этом случае требуется форсировать установку, указав команде `rpm` ключ `-replacefiles`. На одном из узлов необходимо установить пакет `opensm`. На управляющем

узле нужно установить все пакеты библиотек (`lib*.rpm`), пакет `mpich_mlx` и все пакеты с суффиксом `-devel`. Затем достаточно перезагрузить узлы или запустить сервисы `openibd` на всех узлах и `opensm` на том узле, где он установлен, после чего кластер должен быть готов к работе.

Приложение 3.
Работа с тестом High Performance LINPACK

Для того, чтобы измерить производительность кластерной системы на тесте High Performance Linpack (HPL), потребуется сам тест и библиотека BLAS. Собственно тест HPL доступен по адресу `http://www.netlib.org/benchmark/hpl/hpl.tgz`. Настоятельно не рекомендуем использовать библиотеку BLAS, поставляемую с дистрибутивами Linux. Этот вариант очень неэффективен на современных вычислительных платформах, поскольку рассчитан на работу на процессорах Intel 80386. Наиболее удачными библиотеками, содержащими BLAS, являются:

- Atlas (`http://www.netlib.org/atlas/`),
- GotoBLAS (`http://www.tacc.utexas.edu/resources/software/#blas`),
- Intel MKL (`http://www3.intel.com/cd/software/products/asmo-na/eng/perflib/`).

Установите библиотеку BLAS (или содержащую BLAS), распакуйте архив с дистрибутивом HPL. Перейдите в созданный каталог, скопируйте из каталога `setup` любой файл вида `Make.<ARCH>`. Переименуйте его в `Make.MyArch`, где `MyArch` – произвольное название архитектуры узлов. Например, `Lin_Xeon`. Теперь отредактируйте этот файл, указав пути к библиотекам BLAS и компиляторам.

Обязательно измените название переменной `ARCH` в соответствии с названием архитектуры. В качестве компиляторов Си, Фортран и линкеров укажите команды `mpicc` и `mpif77`. Переменные `MPinc` и `MPlib` укажите пустыми.

Теперь можно запустить команду `make arch=MyArch` (`MyArch` – имя выбранной архитектуры). Если компиляция прошла успешно, перейдите в каталог `bin/MyArch` и убедитесь в наличии исполняемого файла `xhpl`. Скопируйте в этот каталог файл `bin/HPL.dat` и отредактируйте его в соответствии с конфигурацией кластера.

Для высоких значений производительности важно подобрать размер задачи так, чтобы использовалась вся оперативная память узлов. Подробное описание формата `HPL.dat` можно найти в файле `TUNING`. Особенно обратите внимание на строки 6 – это размеры используемых матриц, они всегда квадратные, и строки 11-12 – это варианты разбиения матрицы по процессорам.

Пример файла `HPL.dat` (слева указаны номера строк):

```
01: HPLinpack benchmark input file
02: Innovative Computing Laboratory, University of Tennessee
03: HPL.out      output file name (if any)
04: 6            device out (6=stdout,7=stderr,file)
05: 2            # of problems sizes (N)
06: 25000 30000  Ns
07: 3            # of NBs
08: 200 224 250  NBs
09: 0           PMAP process mapping (0=Row-,1=Column-major)
10: 3            # of process grids (P x Q)
11: 5 6 3        Ps
12: 6 5 10       Qs
13: 16.0         threshold
14: 1            # of panel fact
15: 1            PFACTs (0=left, 1=Crout, 2=Right)
16: 1            # of recursive stopping criterium
17: 4            NBMINs (>= 1)
18: 1            # of panels in recursion
19: 2            NDIVs
20: 1            # of recursive panel fact.
21: 1            RFACTs (0=left, 1=Crout, 2=Right)
```

```
22: 1             # of broadcast
23: 0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
24: 1             # of lookahead depth
25: 0             DEPTHs (>=0)
26: 2             SWAP (0=bin-exch,1=long,2=mix)
27: 64            swapping threshold
28: 0             L1 in (0=transposed,1=no-transposed) form
29: 0             U  in (0=transposed,1=no-transposed) form
30: 1             Equilibration (0=no,1=yes)
31: 8             memory alignment in double (> 0)
```

В примере используются матрицы на 25000x25000 и 30000x30000 элементов (строки 05 и 06), размеры блоков 200x200, 224x224 и 250x250 (строки 07 и 08). Матрица распределяется между 30 процессорами тремя вариантами: 5x6, 6x5 и 3x10 (строки 10-12). Остальные параметры обычно не столь значимы, и подробно их рассматривать здесь не будем.

После редактирования файла `HPL.dat` уже можно запускать тест HPL, для чего нужно воспользоваться командой

```
mpirun -np N ./xhpl
```

Число процессоров `N` должно быть не меньше максимального произведения соответствующих вариантов пар `Ps` и `Qs` (строки 11-12).

В конце расчета каждого варианта будет выдана информация о корректности выполнения теста и достигнутая на нем производительность.

Приложение 4.
Использование системы управления заданиями Cleo

Система управления заданиями Cleo контролирует запуск задач на многопроцессорных вычислительных установках, в том числе на кластерах. Она позволяет автоматически распределять вычислительные ресурсы между задачами, управлять порядком их запуска, временем работы, получать информацию о состоянии очередей. При невозможности немедленного запуска задач, они ставятся в очередь и ожидают, пока не освободятся нужные ресурсы.

Система позволяет работать с различными типами заданий, в том числе с последовательными и написанными с использованием различных версий MPI: MPICH, MPICH-gm, LAM, ScaMPI, MVAPICH и другими.

В качестве вычислительной единицы используется процессор. Система оперирует и понятием вычислительного узла, который может объединять несколько процессоров. В рамках одной очереди система считает все процессоры равноправными.

Сама система Cleo написана на языке perl, что позволяет использовать ее на различных платформах.

Общая структура системы

Система Cleo состоит из сервера (запускаемого с правами суперпользователя), управляющего заданиями, программ-мониторов, запускаемых на всех рабочих узлах, и набора клиентских программ, осуществляющих взаимодействие с сервером по TCP/IP либо через файлы состояния сервера.

В качестве клиентских программ в поставку входит так называемый основной клиент (программа `cleo-client`), который поддерживает полный набор команд сервера Cleo. Его используют

скриптовые программы, осуществляющие простые операции с сервером, например, постановку задания в очередь, снятие задания и другие.

Возможно написание произвольных клиентов, использующих протокол взаимодействия сервера и клиентских программ. Предусмотрена возможность анализировать файл состояния сервера для получения данных об очередях, задачах и т.п.

Иерархия очередей

Система очередей способна поддерживать несколько очередей одновременно. Все они организованы иерархически, когда одни очереди включают в себя другие. Это значит, что любой процессор может использоваться одновременно несколькими очередями.

Рассмотрим следующий пример:

<table>
<tr><td colspan="8">main</td></tr>
<tr><td colspan="4">Long</td><td colspan="4">short</td></tr>
<tr><td>p1:1</td><td>p1:2</td><td>p2:1</td><td>p2:2</td><td>p3:1</td><td>p3:2</td><td>p4:1</td><td>p4:2</td></tr>
</table>

Представлена иерархия из трех очередей над 4 вычислительными узлами, содержащими по 2 процессора каждый. Узлы имеют имена p1, p2, p3 и p4, а их процессоры соответственно p1:1, p1:2 для узла p1 и т.д.

Очередь main включает в себя все процессоры. Такая объемлющая очередь должна присутствовать всегда, даже если ей никогда не будут пользоваться (это имеет смысл, например, если она объединяет несколько несвязанных кластеров или разделов одного кластера, управление которыми осуществляется независимо).

В данном примере очередь main имеет две дочерних очереди – long, включающую в себя процессоры p1:1, p1:2, p2:1 и p2:2, и очередь short, включающую в себя процессоры p3:1, p3:2, p4:1 и p4:2.

Дочерние очереди могут пересекаться, но в этом случае информация о занятости процессоров из пересечения не передается между очередями одного уровня, поэтому такой режим нежелателен.

Система следит за тем, чтобы каждый процессор использовался одновременно только одной задачей (если явно не оговорен иной режим). Это достигается за счет того, что задачи очередей верхнего уровня автоматически попадают и в дочерние очереди. Таким образом, когда задача фактически идет на счет, она присутствует и помечается как считающаяся во всех очередях, чьи процессоры она занимает. Например, поставим в очередь short задачу на 4 процессора, а затем в очередь main задачу на 6 процессоров. Последняя задача автоматически попадет в очереди short и long. В очереди long она будет помечена как предзапущенная, то есть для нее будут зарезервированы процессоры, но на счет она отправлена не будет. В очереди short будет считаться первая задача, а вторая будет ждать окончания ее счета.

Как только первая задача досчитается, в очереди short будет предзапущена вторая задача. Очередь main (которой и принадлежит вторая задача) получит 8 процессоров для ее запуска. Так как заказано для нее было только 6 процессоров, то ровно 6 процессоров будет отобрано для счета. Остальные 2 процессора будут сняты с резервирования и могут быть использованы для запуска новых задач.

Описанную выше ситуацию можно условно представить в таком виде:

До постановки задач

main	
long	**short**

После постановки задач

Main	
задача2	
long	**short**
задача2	*задача1*
	задача2

Каждая из очередей может быть настроена независимо. Иными словами, в различных очередях могут быть заданы различные политики для пользователей, лимиты, стратегии планирования.

В качестве лимитов могут быть заданы количество процессоров на одну задачу, количество одновременно занятых процессоров (если пользователь одновременно запускает несколько задач), максимальное время счета задачи, максимальный приоритет задачи и другие.

Ограничения могут также задаваться и для отдельных пользователей, при этом такие ограничения могут быть как строже, так и мягче, чем для очереди в целом.

Приоритеты

В очереди задачи сортируются по приоритету. Приоритет – это целое число в диапазоне от 0 до 256. Приоритет по умолчанию задается в настройках очереди. Если он не задан, то он принимается равным 10. Задачи с более высоким приоритетом всегда ставятся в очередь выше задач с более низким приоритетом и будут запущены раньше, даже если они были поставлены в очередь позже по времени. Приоритет можно повышать и понижать после постановки задачи в очередь.

Пользователь может понизить приоритет своей задачи, а также повысить его до величины, не превышающей установленного для него лимита.

Ограничение времени счета

Для задач можно задать ограничение времени счета. Обычно оно задано по умолчанию, но можно его понизить, если это необходимо. По истечении указанного лимита задача будет принудительно снята со счета.

Система Cleo ориентируется на то, что задача будет считаться не дольше указанного лимита времени, и учитывает это при планировании времени старта других задач.

Стратегии выбора процессоров

При старте задачи для нее выделяются процессоры. В системе есть возможность управлять стратегией выбора процессоров для задач. Есть три встроенные стратегии:

`random`	Процессоры выбираются случайным образом.
`random_hosts`	Процессоры выбираются на случайно выбранных узлах, предпочтительно занимая все доступные процессоры на узле.
`hosts_alone`	Процессоры выбираются на случайно выбранных узлах, предпочтительно занимая лишь по одному процессору на узле.

Элемент случайности в выборе процессоров присутствует для достижения двух целей: лучшей сбалансированности нагрузки на процессоры и предотвращения блокировок, связанных с неудачным заказом конфигурации процессоров.

В системе есть возможность подключения внешних модулей для задания своих стратегий. В случае использования внешнего модуля, он должен быть описан в разделе `pe_sel_method` секции [`server`], где также будет описано его имя. Для подключения модуля в систему достаточно просто указать его имя вместо имени встроенного метода.

Политики пользователей

Как для всех очередей, так и для каждой очереди в отдельности можно задать список пользователей, которые имеют право ставить на счет свои задачи. Все остальные пользователи не будут иметь доступа к соответствующим очередям. Можно запретить доступ к очередям фиксированному списку пользователей, тогда для всех остальных доступ будет открыт.

Права доступа не наследуются очередями-потомками. В контексте нашего примера, если пользователю user1 разрешен доступ в очередь main, то ему не обязательно разрешен доступ в очередь short или long.

Для всех очередей и для каждой очереди в отдельности можно задать список пользователей-администраторов, которые в рамках соответствующих очередей получают возможность управлять лимитом времени работы задач, приоритетами, удалять любые задачи и выполнять другие административные действия.

Блокировки

Системой Cleo поддерживаются следующие виды блокировок:

- блокировка очереди на запуск задач;
- блокировка очереди на постановку новых задач;
- блокировка процессоров;
- отложенная блокировка процессоров;
- блокировка задач;
- автоблокировка пользователей;
- автоблокировка по времени;
- автоблокировка по ограничению ресурсов.

Блокировка очереди на запуск задач означает, что задачи, стоящие в очереди, не будут запущены на счет, пока блокировка не снята. При этом уже запущенные задачи со счета не снимаются. Любая очередь может быть в любой момент времени заблокирована на добавление новых задач. На работающих и уже стоящих в очереди задачах это никак не отражается.

Указанные блокировки могут быть установлены рекурсивно, то есть не только на конкретную очередь, но и на все ее дочерние очереди.

Блокировкой процессора можно запретить исполнение задач на конкретном процессоре или узле. Отложенная блокировка процессора активируется только после того, как процессор заканчивает выполнение пользовательской задачи. Когда это происходит, администратору высылается уведомление.

Автоматические блокировки срабатывают при наступлении определенных условий. Автоблокировка новых задач пользователя начинает действовать при постановке задачи в очередь. Автоблокировка по времени срабатывает в том случае, если задача не успевает завершиться к указанному времени. Автоблокировка по ресурсам вступает в силу, если запуск задачи пользователя нарушает наложенное ограничение, например, по числу одновременно занимаемых процессоров.

Схема запуска задач

Для запуска задач в системе Cleo предусмотрено несколько режимов, но предпочтительным является следующий. Задача запускается в псевдотерминале (используется пакет `empty-cleo`[1]) обычным для выбранной среды способом. Например, для MPICH или LAM будет запущена программа `mpirun` из соответствующего дистрибутива. При запуске программе передаются данные о процессорах, на которых надо запускаться.

На соответствующие узлы передается команда, по которой все новые процессы указанного пользователя запоминаются. В дальнейшем все процессы, порожденные от этих процессов, также запоминаются. При

[1] Основан на пакете empty, автор Михаил Захаров (zmey20000@yahoo.com)

завершении задачи все запомненные процессы принудительно завершаются через заданный промежуток времени.

Во время работы задачи можно подключиться к псевдотерминалу командой `cleo-empty` и выполнить любые интерактивные действия.

Запущенные задачи не зависят от сервера Cleo, т.к. работают в собственных псевдотерминалах. Это значит, что сервер может быть остановлен или перезапущен во время работы задач.

Возможности дополнения и расширения

Статус очередей можно получать не только посредством команд серверу, но и из файлов состояния. Именно так работает пакет `qs-web`, позволяющий представлять состояние очередей или задач в любом удобном виде, например, как web-страницу или в текстовом виде.

Система сделана расширяемой за счет использования модулей. В данный момент реализованы следующие типы модулей:

- модули стратегий распределения процессоров;
- модули, вызываемые при запуске или завершении задачи;
- планировщики задач.

Постановка заданий в очередь и удаление задач

Запуск задач осуществляется командой `mpirun`:

```
mpirun -np N [-q queue][-maxtime|-l lim][-p pri][-stdin
                        file]
    [-stdout file] [-stderr file] prog [prog_args]  ,
```

либо той же командой в таком варианте:

```
mpirun -np N [-q queue][-maxtime|-l lim][-p pri][-stdin
                        file]
    [-stdout file] [-stderr file] -f batch_file   .
```

Ключи, указанные в скобках, являются необязательными, а `prog` и `prog_args` – это исполняемый файл задачи и ее аргументы соответственно. Второй вариант запуска предполагает наличие файла `batch_file`, в котором перечислены программы с аргументами. В данном случае происходит последовательный запуск перечисленных программ в рамках одной задачи.

Ниже описаны значения ключей `mpirun`.

`-np`	– количество процессоров.
`-q`	– название очереди, в которую ставится задача.
`-maxtime`	– лимит времени счета в минутах.
`-l`	– лимит времени счета в секундах.
`-p`	– приоритет задачи в очереди.
`-stdin`	– файл для стандартного ввода.
`-stdout`	– файл для стандартного вывода.
`-stderr`	– файл для стандартного потока ошибок.

Например, команда вида:

```
mpirun -np 15 -q users -maxtime 10 my_test -dummy -i 50
```

поставит задачу `'my_test'` с параметрами `'-dummy -i 50'` в очередь `users` с лимитом работы в `10` минут и запросом на `15` процессоров.

Следующая команда:

```
mpirun -np 15 my_test2 /tmp/my_test.tmp -n 10
```

поставит задачу `'my_test2'` с параметрами `'/tmp/my_test.tmp -n 10'` в очередь по умолчанию с запросом на `15` процессоров.

Если ключ `-maxtime` (или `-l`) не задан, то будет взято значение из переменной окружения `QS_TIMELIMIT`. Если оно пустое, или данная переменная не определена, то будет использовано значение, заданное в пользовательском файле `~/.qconf` параметром `def_time`. Если в

пользовательском файле этого параметра нет, то он будет взят из глобального файла конфигурации.

Указывая ключ `-maxtime`, пользователь может ускорить запуск своей задачи, так как дает системе возможность планировать время счета этой задачи и, как следствие, возможность выполнить ее “досрочно”, если будет такая возможность.

При постановке задачи в очередь запоминаются все переменные окружения, во время запуска они будут установлены в эти значения.

После запуска задачи ее стандартный вывод будет перенаправлен в файл, имя которого определяется настройками по умолчанию, либо индивидуальными настройками пользователя, либо ключом `-stdout`. Обычно этот файл создается в том же каталоге, откуда задача была запущена, и имеет имя `<имя_задачи>.out-<номер>`, где `номер` – это номер задачи в очереди.

Удалить задачу из очереди или досрочно снять задачу со счета можно командой `tasks` с ключом `-d`. Например:

```
/home/usr2> tasks -d 2141
```

удалит задачу с номером `2141` из очереди.

По окончании работы задачи создается файл отчета (его имя обычно аналогично имени файла вывода задачи, но суффикс `out` меняется на `rep`), в котором указываются полное имя задачи, аргументы, время счета, имя файла вывода, код возврата и, если задача завершилась аварийно, причина останова.

Во время работы задачи на узлах специально для нее создается временный каталог, который будет очищен по окончании работы. В этом каталоге целесообразно создавать временные рабочие файлы. Узнать полный путь к нему можно во время работы программы из переменной окружения `TEMP_DIR`.

Просмотр состояния очереди

Посмотреть состояние очереди можно командой `tasks`:

```
tasks [-q queue][-r][-l][-t][-f][-o][-m mask][-u userlist]
                    [-b][-d id ... id|-v]
```

Указанные в скобках ключи являются необязательными, а ниже приведено их краткое описание.

`-q`	– название очереди,
`-r`	– показывать очереди рекурсивно,
`-l`	– показывать дополнительную информацию,
`-t`	– показывать лимит времени по умолчанию для пользователя,
`-f`	– учитывать чужие задачи,
`-o`	– учитывать свои задачи,
`-m mask`	– использовать маску для выборки задач,
`-u list`	– использовать список пользователей для выборки задач,
`-b`	– показывать информацию о заблокированных узлах,
`-v`	– показывать состояние очереди (по умолчанию),
`-d`	– удаление задачи.

Параметры `-f, -o` указывают выборку задач. По умолчанию в выборку попадают все задачи, если не указано иначе в файле конфигурации.

Более полная информация по системе управления заданиями Cleo, включая дистрибутив и документацию, доступна на следующей странице: `http://parcon.parallel.ru/cleo.html`.

Приложение 5.
Пакеты для выполнения инженерных расчетов

В данном приложении будут показаны примеры существующих наиболее распространенных в настоящее время пакетов для выполнения инженерных расчетов. Перечисленные пакеты либо уже имеют версию, адаптированную для использования на кластерных системах, либо такая версия находится в стадии разработки. Включив данное приложение в книгу, мы не ставили своей целью дать исчерпывающее описание возможностей и областей применения каждого пакета. Основной задачей было обратить внимание на имеющееся разнообразие уже готового и мощного инструментария высокого качества. Все эти пакеты созданы командами профессионалов со значительным опытом работы в своих предметных областях, поэтому не стоит сразу бросаться за создание своих собственных программ. Первым делом нужно аккуратно оценить, что уже сделано профессиональным сообществом, что доступно и чем можно воспользоваться для решения стоящих задач.

Аналогичные инструменты есть не только в области инженерного анализа. Большое распространение получили специализированные прикладные пакеты для выполнения на кластерах квантово-химических расчетов, решении задач сейсморазведки, биоинженерии, криптографии и многих других областей. Обо всех разработках рассказать невозможно, да и не нужно. Задача в другом: показать имеющееся разнообразие готовых пакетов в одной лишь предметной области и предостеречь от попыток немедленной разработки собственных программ для кластеров.

Более подробную информацию можно найти на сайте Parallel.ru или на сайтах компаний-производителей соответствующих программных комплексов.

ANSYS. Конечно-элементный пакет, включающий в себя целое семейство специализированных подсистем для решения в единой среде широкого спектра инженерных задач: акустика, прочность, деформация, упругость, пластичность, текучесть, теплофизика, конвекция и радиационный обмен, динамика жидкостей и газов, электромагнетизм и другие. Не так давно известные пакеты **LS-DYNA, FLUENT** и **CFX** были также интегрированы в среду ANSYS.

Дополнительная информация: `http://www.ansys.com`.

STAR-CD. Многоцелевой программный комплекс, предназначенный для проведения расчетов в области механики жидкости и газа, тепло- и массопереноса, процессов горения и решения других задач.

Дополнительная информация: `http://www.cd-adapco.com`.

LMS Virtual.Lab Acoustic. Пакет предназначен для решения широкого спектра задач акустики, моделирования рабочих характеристик механических систем, структурной целостности, уровней шума и вибрации, долговечности, характеристик движения и управления. Многим известен под названием SYSNOISE.

Дополнительная информация: `http://www.lmsintl.com`.

ABAQUS. Конечно-элементный комплекс для решения множества задач инженерного анализа: расчет прочности турбомашин и проектирование двигательных установок, расчет сварных соединений, анализ аварийных столкновений и выполнение тестов на падение, литье металлов, пробивание материала, расчет взрывных и сейсмических воздействий, анализ прочности и надежности конструкций и многие другие.

Дополнительная информация: http://www.abaqus.com.

MSC.Nastran. Расчет напряженно-деформированного состояния, собственных частот и форм колебаний, анализ устойчивости, решение задач теплопередачи, исследование установившихся и неустановившихся процессов, акустических явлений, нелинейных динамических переходных процессов, расчет вибраций роторных машин, аэроупругости.

Дополнительная информация: http://www.mscsoftware.com.

FLOW-3D. Пакет вычислительной гидродинамики общего назначения, способный решать множество задач течения жидкости и/или газа.

Дополнительная информация: http://www.flow3d.com.

COMET Acoustics. Конечно-элементный пакет для решения задач акустики и вибрации в жидких, твердых и пористых средах.

Дополнительная информация: http://www.cometacoustics.com.

GDT (GasDynamicsTool). Пакет для моделирования газо-динамических процессов в широком диапазоне граничных и начальных условий, расчет газовых струй, решение задач внутренней и внешней аэродинамики, баллистики, горения и детонации.

Дополнительная информация: http://www.cfd.ru.

FlowVision. Комплекс позволяет моделировать трехмерные стационарные и нестационарные слабосжимаемые и несжимаемые потоки жидкости в различных технических приложениях, сложные движения жидкости, включая течения с сильной закруткой и горением.

Дополнительная информация: http://www.flowvision.ru.

Приложение 6.
Основные термины и сокращения

DAPL: *Direct Access Programming Library.* Библиотека для использования прямого доступа в память удалённого компьютера без необходимости явного описания конкретного типа оборудования.

DHCP: *Dynamic Host Configuration Protocol.* Протокол, позволяющий компьютеру на стадии загрузки ОС или позже получить от сервера информацию, такую как свой IP-адрес, сетевое имя и прочее.

FTP: *File Transfer Protocol.* Протокол передачи файлов по сети.

KVM: *Keyboard and Video Monitor*. Устройство, позволяющее подключить несколько компьютеров к одному монитору и клавиатуре.

Linpack: тест для некоторой оценки реальной производительности параллельных вычислительных комплексов. Чаще всего используется модификация High Performance Linpack (HPL).

LVM: *Logical Volume Manager*. Технология построения логических дисков, используя несколько физических дисков и/или RAID-ов.

MAC-адрес: уникальный адрес сетевой карты в стандарте Ethernet

MPI: *Message Passing Interface*. Открытый стандарт библиотеки, предназначенной для передачи сообщений внутри параллельного приложения. Есть множество реализаций этого стандарта (mpich, lam, openmpi и другие).

NFS: *Network File System*. Сетевая файловая система. Стандарт deFacto в среде Unix.

NIS: *Network Information System.* Технология, позволяющая хранить учетные записи о пользователях, именах компьютеров и другую системную информацию на сервере и получать ее с любого компьютера в сети.

NTP: *Network Time Protocol*. Протокол синхронизации времени по сети.

RAID: *Redundant Array of Independent/Inexpensive Disks*. Массив из нескольких жестких дисков, логически объединенных для большей отказоустойчивости, скорости и/или объема.

RAID-0 (stripe): RAID, диски которого для повышения скорости работы с ними объединены так, что логически блоки дисков чередуются – блок1 первого диска, блок1 второго, ... блок2 первого диска, блок2 второго и т.д.

RAID-1 (mirror): RAID, диски которого для повышения надежности объединены в "зеркало" – информация пишется одновременно на все диски в блоки с одинаковыми номерами.

RAID-5: RAID, диски которого объединены в группы четности – при записи в какой-то логический блок, записанные данные складываются методом XOR с другими блоками в группе и полученная информация записывается в отдельный блок. При чтении корректность данных проверяется и, если один из блоков поврежден, то информация автоматически восстанавливается.

RDMA: *Remote Direct Memory Access.* Протокол для прямого доступа в память удалённого компьютера.

SCI: *Scalable Coherence Interface.* Стандарт на оборудование для высокоскоростной передачи данных. Подразумевает соединение сетевых плат напрямую друг с другом в кольцо либо тор (двух- или трёхмерный).

Samba: Программный пакет, реализующий протоколы SMB и CIFS, использованные в MS Windows для сетевых дисков. Позволяет обращаться к сетевым дискам Windows из Linux, а также создавать сетевые диски под Linux таким образом, чтобы ими могли пользоваться клиенты Windows.

SNMP: *Simple Network Management Protocol.* Протокол, созданный для контроля и управления оборудованием в сети.

SSH: *Secure SHell*. Протокол для удалённого доступа на компьютеры в сети, предполагающий использование шифрованного соединения.

Telnet: протокол для удалённого доступа на компьютеры в сети. Использует незашифрованный канал передачи данных.

TFTP: *Trivial File Transfer Protocol*. Протокол, позволяющий получить файлы с сервера, например, загрузочный образ.

U: *Unit*. Единица измерения высоты стоечного оборудования, равная 5/4 дюйма или около 4,5 см. Иногда используется обозначение RU – rack unit (см. *стойка*).

UPS: *Uninterruptable Power Supply*. Источник бесперебойного питания.

Кабельный органайзер: конструкция, позволяющая укладывать кабели в рамках выделенного пространства.

Коммуникационная сеть: используется для обмена данными вычислительными задачами.

Коммутатор: устройство, позволяющее нескольким сетевым адаптерам объединяться в сеть.

Латентность: время, затрачиваемое при передаче пакета по сети независимо от его длины.

ПО: программное обеспечение. Кроме собственно набора программ, сюда относят конфигурационные и иные файлы, необходимые для его работы.

Рейд-контроллер: устройство, объединяющее несколько жестких дисков в RAID.

Сервисная сеть: используется для контроля и управления состоянием вычислительных узлов.

Системная консоль: В общем случае виртуальный экран и подключенная к нему клавиатура. На системную консоль поступают сообщения от ядра ОС. С системной консоли можно управлять процессом загрузки ОС.

Стойка: конструкция, предназначенная для монтажа компьютеров и иного оборудования и отвечающая определенным стандартам (ширина оборудования, метод крепления и т.п.). Наиболее распространены стойки шириной 19 дюймов (19").

Транспортная сеть: используется для сетевой файловой системы, команд запуска на узлах и т.п.

Узел (кластера): компьютер, предназначенный для определённых задач в кластере (вычислительный, управляющий, ввода-вывода и др.).

Файл-сервер: компьютер, предоставляющий часть своей файловой системы другим компьютерам через сеть.

Файловое хранилище: оборудование, предоставляющее дисковое пространство по сети или локально, например, для файл-сервера.

Форм-фактор: стандарт, в соответствии с которым изготовляется корпус компьютера. Например, tower, minitower, 1 U, 4 U и другие.

Приложение 7.
Сетевые ресурсы по смежным областям

• Информационно-аналитический центр по параллельным вычислениям Parallel.ru:

`http://www.parallel.ru/`

• Список Top50 самых мощных систем СНГ, примеры конфигураций:

`http://www.supercomputers.ru/`

• Список Top500 самых мощных систем мира, примеры конфигураций:

`http://www.top500.org/`

• Очень содержательный сайт и подборка статей по кластерной тематике:

`http://www.clustermonkey.net/`

• Страница первой кластерной системы Beowulf:

`http://www.beowulf.org/`

• Пакет тестов HPC Challenge Benchmark:

`http://icl.cs.utk.edu/hpcc/`

• Обучение кластерным технологиям в рамках института Linux-кластеров:

`http://www.linuxclustersinstitute.org/`

• Большая коллекция ссылок по кластерным системам и технологиям:

`http://www.cbel.com/Beowulf_Parallel_Computing/`

• Информация по использованию и настройке Unix-систем:

`http://www.opennet.ru/`

• Сайт проекта MOSIX для программирования набора Linux-кластеров:

`http://www.mosix.org/`

• Библиотека аналитических материалов по программным технологиям, ОС и пакетам:

`http://www.citforum.ru/`

• Информация по Microsoft Windows Compute Cluster Server 2003:

http://www.microsoft.com/rus/windowsserver2003/ccs/

• Новостное агентство по суперкомпьютерной тематике:

http://www.hpcwire.com/

• Журнал “LINUX Magazine”:

http://www.linux-mag.com/

http://www.linuxmagazine.com

• Описание процессоров компании Intel:

http://www.intel.ru/products/processor

• Использование многоядерных процессоров Intel:

http://www.intel.ru/software/multicore

• Инструментальные средства Intel для разработки ПО:

http://www.intel.com/software/products

• Курсы обучения, предлагаемые Intel Software College:

http://www.intel.com/software/college

• Программа Intel Software Partner Program для компаний-разработчиков:

http://www.intel.com/partner/rus

• Русская версия журнала Intel Software Insight:

http://developer.intel.ru/softwaremailing

• Система управления заданиями Torque:

http://www.clusterresources.com/pages/products/torque-resource-manager.php

• Система управления заданиями OpenPBS PRO:

http://www.altair.com/software/pbspro.htm

• Система управления заданиями LSF:

http://www.platform.com/Products/Platform.LSF.Family/

• Обширный каталог изданий Intel Press по процессорам, ПО и различным смежным вопросам:

http://www.intel.com/intelpress/ – загляните, здесь много книг, очень полезных для практической работы!

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. Shainer G. Cluster Interconnects: Real Application Performance and Beyond // http://www.clustermonkey.net/

3. Таненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.

4. Андреев А.Н., Воеводин Вл.В., Жуматий С.А. Кластеры и суперкомпьютеры – близнецы или братья? // Открытые системы, 2000, N5–6. С. 9–14.

5. Система управления ресурсами и мониторинга кластерных вычислительных систем ParCon // Интернет-ресурс, http://parcon.parallel.ru/

6. Лацис А. Как построить и использовать суперкомпьютер. – М.: Бестселлер, 2003. – 240 с.

7. Reinders J. VTune Performance Analyzer Essentials. Measurement and Tuning Techniques for Software Developers. – Intel Press, 2005. – 455 p.

8. Кофлер М. Весь Линукс. Установка, конфигурирование, использование. – М.: Бином-Пресс, 2006. – 880 с.

9. Немет Э., Снайдер Г., Сибасс С., Хейн Т. UNIX. Руководство системного администратора. Для профессионалов. – СПб.: Питер, 2006. – 928 с.