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.