Технологии параллельных вычислений для решения расчетных задач диспетчерского управления транспортом газа

В.А. Швечков (РГУ нефти и газа имени И.М. Губкина)

Рассмотрены технологии параллельного и распределенного программирования, позволяющие повысить эффективность расчетных комплексов моделирования режимов работы газотранспортных систем. Проведен краткий обзор и анализ наиболее популярных из существующих технологий параллельного программирования, применяемых для ОС Windows. Представлены результаты расчетов модели компрессорного цеха с применением технологий MPI, OpenMP и Windows API.

Разработка распределенного вычислительного комплекса (ВК) моделирования режимов работы газотранспортных систем (ГТС) позволяет эффективно организовать взаимодействие с удаленными информационными узлами Intranet и Internet сетей. Распределенная архитектура наилучшим образом соответствует реальной иерархической системе диспетчерского управления транспортом газа.

В настоящее время для практической реализации моделей объектов ГТС широко используются расчетные методы и алгоритмы [1], на базе которых разработаны как отечественные: "АСТРА" (Тюменский филиал ООО "Информгаз"); "САМПАГ" (ООО "Мострансгаз", ООО "Ингойл"), так и зарубежные комплексы: GAMOS/GANESI фирмы debis Systemhaus (Германия); SIMONE фирмы SIMONE Research Group (Чехия).

Большинство существующих комплексов моделирования содержат в своей основе реализацию последовательных расчетных моделей, тесно связанных с графическим интерфейсом, что по сути своей не дает действительного выигрыша при использовании кластерной, многопроцессорной или многоядерной платформы.

Предложенная в [2] архитектура распределенного ВК позволяет воспользоваться всеми преимуществами научно-технического прогресса в области компьютерных технологий, что явилось результатом проведения работ по выделению моделей отдельных объектов в собственные расчетные методы, разделению кода и данных, созданию независимых от графического интерфейса пользователей библиотек расчетных объектов.

Проведенная работа позволяет внедрить в ВК методы параллельного и распределенного программирования. В предложенной архитектуре сочетаются две различные парадигмы программирования [3]:

- методы параллельного программирования, которые позволяют распределить работу программы между двумя (или больше) процессорами в рамках одного физического или одного виртуального компьютера;

- методы распределенного программирования, обеспечивающие распределение работы программы между двумя (или больше) процессами, причем процессы могут существовать на одном и том же ПК или на разных, связанных сетью.

Следует отметить, что не все распределенные программы включают параллелизм. Например, получение информации из SCADA-системы и проведение расчета могут выполняться по различным запросам в различные периоды времени. При этом могут быть задействованы ресурсы различных ПК. В настоящее время используется гибридный подход, который сочетает приемы параллельного и распределенного программирования.

Для написания эффективной программы необходимо выполнить процесс проектирования, который состоит из трех основных функций.

Декомпозиция. Процесс разбиения на части задачи и ее решения, который часто сводится к декомпозиции работ. При этом одной из проблем параллельного программирования является идентификация естественной декомпозиции работ. Параллелизм обнаруживается в процессе моделирования. Причем, если "естественный" параллелизм не обнаруживается, то не стоит его навязывать насильно.

Связь. После декомпозиции программы на ряд параллельно выполняемых частей возникает задача их связи между собой. При этом, если разные части программы разнесены по процессам или различным компьютерам, то необходимо ответить на ряд вопросов. Каким образом одна часть программы узнает о том, что другая справилась со своей задачей? Какая часть программы должна первой приступить к работе? Как один компонент узнает об отказе другого? На эти и многие другие вопросы необходимо ответить на стадии проектирования.

Синхронизация. Задача синхронизации возникает, когда в рамках выполнения одной работы необходимо скоординировать работу одновременно выполняющихся подзадач. При этом возникают вопросы последовательности запуска задач, порядка их завершения, обмена и доступа к данным и прочие.

Разработанная для решения режимно-технологических задач библиотека содержит набор моделей объектов ГТС таких, как труба, кран, газоперекачивающий агрегат (ГПА), группа ГПА, аппарат воздушного охлаждения (АВО), компрессорный цех (КЦ), компрессорная станция (КС) и газотранспортная система (ГТС) целиком. Помимо моделей объектов ГТС, в библиотеке реализованы различные алгоритмы и методы решения режимно-технологических задач. При рассмотрении данных моделей и методов с позиций параллельного программирования возникает целый ряд задач, содержащих естественный параллелизм таких, как:

- технологический расчет режима работы КЦ – расчет параллельно работающих групп агрегатов в цехе;

- технологический расчет режима работы КС – расчет параллельно соединенных КЦ;

- технологический расчет режима работы ГТС в целом и в режиме on-line;

- расчет областей допустимых режимов работы ГПА, КЦ по давлению и расходу с учетом начальных значений и ограничений.

Создание эффективной параллельной программы, прежде всего, зависит от правильного выбора технологии. Выбор конкретной технологии реализации программного решения определяется следующими факторами: аппаратной платформой; ОС; масштабируемостью и переносимостью приложения; совместимостью с языком программирования. Определение этих параметров позволяет существенно ограничить задачу выбора подходящей технологии, так как даже при поверхностном анализе существующих технологий параллельного программирования [4] получается список из более чем 100 различных средств.

Большинство комплексов моделирования режимов работы ГТС, предназначенные для использования диспетчерскими службами газотранспортных предприятий, разработаны под ОС Microsoft Windows. В результате имеет смысл рассматривать применимость технологий параллельного программирования, исходя из следующих предпосылок: аппаратной платформы (многопроцессорная или многоядерная на базе 32- и 64-битных процессоров Intel и AMD), платформы (Win32/64), языка программирования (C++).

Безусловно, нельзя забывать о существующих вычислительных системах, построенных на аппаратно-программной базе ведущих мировых производителей под управлением UNIX-подобных систем. Они являются надежными, высоко производительными системами и заслуживают отдельного рассмотрения, выходящего за рамки данной статьи.

Для указанных ограничений подходят несколько технологий [3, 4, 6, 8-11] таких, как:

- MPI (The Message Passing Interface) – библиотека функций, предназначенная для поддержки работы параллельных процессов в терминах передачи сообщений. Это наиболее распространенная технология программирования параллельных компьютеров с распределенной памятью. Для программирования в среде Windows известны как коммерческие, так и бесплатные реализации библиотек. Одним из примеров является библиотека MPICH, поддерживающая крупномасштабное сложное программирование кластеров;

- OpenMP – библиотека функций для программирования на масштабируемых SMP-системах (нескольких однородных процессоров с массивом общей памяти) с разделяемой памятью. В стандарт OpenMP входят спецификации набора директив компилятора, процедур и переменных среды. OpenMP является одним из наиболее популярных средств программирования компьютеров с общей памятью. Бесспорным преимуществом данной технологии является поддержка интерфейса OpenMP компанией Microsoft и включение библиотек версии 2.0 в состав интегрированного средства разработки Microsoft Visual Studio 2005;

- HPF (High Perfomance Fortran) – расширение языка Фортран для параллельных вычислительных систем. В код программы на Фортране вставляются специальные директивы компилятора, в которых содержатся такие указания, как данные и операции над ними должны быть распределены по оперативной памяти и процессорам;

- DVM-система, созданная в Институте прикладной математики им. М.В.Келдыша РАН, позволяет разрабатывать на языках C-DVM и Fortran-DVM параллельные программы для ЭВМ различной архитектуры и сетей ЭВМ. Аббревиатура DVM соответствует двум понятиям: Distributed Virtual Memory и Distributed Virtual Machine. Первое отражает наличие единого адресного пространства. Второе отражает использование виртуальных машин для двухступенчатой схемы отображения данных и вычислений на реальную параллельную машину;

- PVM – параллельная виртуальная машина – одна из популярных реализаций модели распределенных вычислений. Эта модель ориентирована на распределенные вычисления в сети разнородных машин. PVM видит сетевой вычислитель как совокупность пользовательских процессов, кооперирующихся для решения общей задачи.

Отдельно следует рассматривать базовые функции ОС Microsoft Windows по управлению процессами, потоками и нитями, являющимися частью Windows API. Они обеспечивают большую гибкость и полный контроль над работой параллельных потоков программы, различные способы синхронизации потоков и методы доступа к ресурсам системы.

Сложно определить, какой из инструментов лучше подходит для создания параллельных программ, т.к. применение конкретной технологии сильно зависит от решаемой задачи, используемых алгоритмов, архитектуры системы, мастерства команды разработчиков и прочих факторов. Однако разработчику необходимо сделать выбор в пользу той или иной технологии или к их некоторому гибриду.

Рассмотрим подробнее наиболее развитые и практически используемые технологии – MPI, OpenMP и базовые средства ОС Windows.

Использование MPI дает разработчику большой набор функций. В числе основных достоинств MPI по сравнению с интерфейсами других коммуникационных библиотек обычно выделяют следующие его возможности [5]:

- использование в языках Фортран, Си, Си++;

- совмещение обмена сообщениями и вычислений;

- предоставление режимов передачи сообщений, позволяющих избежать излишнего копирования информации для буферизации;

- наличие коллективных операций (например, широковещательная рассылка информации, сбор ин-

формации с разных процессоров), допускающих гораздо более эффективную реализацию, чем использование соответствующей последовательности пересылок точка-точка;

- наличие редукционных операций (например, суммирование расположенных на разных процессорах данных, или нахождение их максимальных или минимальных значений), не только упрощающих работу программиста, но и допускающих гораздо более эффективную реализацию, чем это может сделать прикладной программист, не имеющий информации о характеристиках коммуникационной системы;

- удобные средства именования адресатов сообщений, упрощающие разработку стандартных программ, или разделение программы на функциональные блоки;

- задание типа передаваемой информации, позволяющее обеспечить ее автоматическое преобразование в случае различий в представлении данных на разных узлах системы.

Однако разработчики MPI подвергаются и суровой критике за то, что интерфейс получился слишком громоздким и сложным для прикладного программиста. Интерфейс оказался сложным и для реализации, в итоге в настоящее время практически не существует реализаций MPI, в которых в полной мере обеспечивается совмещение обменов с вычислениями. В настоящее время реализована библиотека MPICH2, которая является более сложной и расширяет возможности стандартов MPI-1 и MPI-2.

При использовании технологии OpenMP за основу берется последовательная версия программы. Для создания параллельной версии разработчику предлагается использовать набор специальных директив, процедур и переменных окружения. Такой подход существенно упрощает перепроектирование ранее созданных последовательных программ. Кроме того, при отсутствии поддержи компилятором библиотек OpenMP внедряемые в программу директивы OpenMP просто игнорируются и программа превращается в свой последовательный аналог.

Использование стандартных средств Windows требует от разработчика дополнительных затрат на изменение кода и глубокого понимания организации работы многопоточного приложения в среде Windows.

Для исследования возможностей применения технологий параллельных вычислений рассмотрена задача поиска расхода газа, прокачиваемого через КЦ при заданных параметрах газового потока: $P_{вх}$ – давление газа на входе КЦ, $T_{вх}$ – температура газа на входе КЦ, $P_{вых}$ – давление газа на выходе КЦ. Расчетной моделью являлся компрессорный цех, состоящий из n параллельных групп однотипных ГПА. Каждая группа состояла из двух последовательно соединенных ГПА. В этом случае минимальной расчетной единицей при распараллеливании вычислений является модель группы ГПА. Таким образом, если рассматривать ГТС из m КЦ, то общее число рассчитываемых групп ГПА равно $m \times n$.

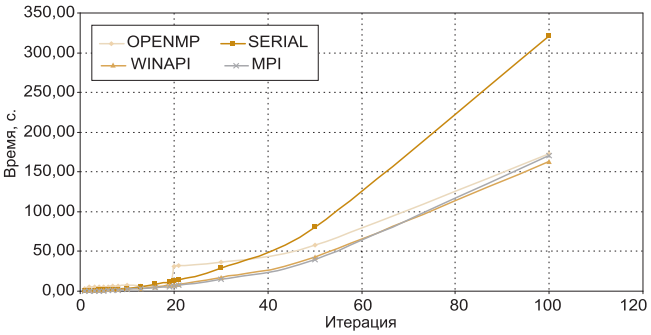

Рис. 1. Зависимость времени решения задачи поиска расхода через КЦ от числа итераций с использованием различных технологий

Проведен расчет задачи в зависимости от числа итераций для модели с общим числом рассчитываемых групп ГПА $m \times n = 100$. В качестве аппаратно-программной платформы использовался двухпроцессорный сервер с частотой 1,2 ГГц и ОС Windows Server 2003. В результате расчета были получены данные (рис. 1) о временных затратах использования последовательных вычислений и технологий параллельного программирования.

Как видно на рис. 1, разница между временем последовательного расчета и расчетов с применением технологий параллельных вычислений увеличивается с ростом числа итераций или, другими словами, с увеличением расчетной сложности задачи или числа повторных итераций увеличивается выигрыш в производительности. Автоматически напрашивается вопрос: "Какой максимальный выигрыш в производительности может быть достигнут? С какого момента он наступает?". Казалось бы, что ответ очевиден – максимальный рост производительности будет равен числу процессоров (ядер) компьютера. Но это далеко не так, ведь при параллельных вычислениях существуют дополнительные накладные расходы, связанные с созданием и уничтожением потоков (процессов), их синхронизацией и координацией процесса вычислений. Причем выигрыш в производительности сильно зависит от типа решаемой задачи и необходимых для ее решения ресурсов ОС. Чем больше ресурсов памяти, устройств ввода/вывода, периферийных устройств и т.д. используется программой, тем ниже будет ожидаемый показатель общей производительности программы.

Так, например, в [6] осуществлен расчет с применением двухпроцессорной системы и достигнут выигрыш производительности в 1,7 раза.

Показатели производительности (для приведенного в статье примера), раз:

последовательные вычисления	*1,0*
OpenMP	*1,942*
Windows API	*1,966*
MPI	*1,995*

Следует отметить, что для расчетов модели не использовались обращения к периферийным устройствам. Основная нагрузка приходилась на процессор и

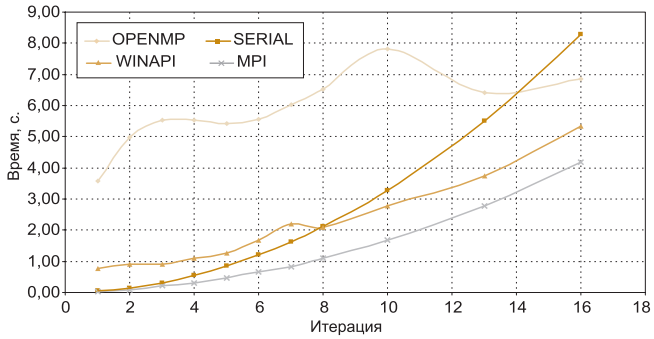

Рис. 2. Сравнение производительности последовательной и параллельных расчетов модели

память ПК. Однако при решении реальных задач обращение к подсистеме ввода/вывода осуществляется достаточно часто и выигрыш производительности в этом случае будет меньше.

Полученные результаты показывают, что различие в росте производительности, в зависимости от применяемой технологии, невелико и наблюдается только во втором порядке и измеряется десятками миллисекунд.

Особо следует заметить, что выигрыш в производительности от применения параллельных аналогов последовательной программы наступает не с первой итерации, а лишь при достаточном числе итерации или при изначально высоком уровне вычислительной сложности задачи (рис. 2). Так, для приведенного примера, рост производительности при применении технологий параллельных вычислений наблюдается уже после 10-го прогона, но стабильного значения он достигает ближе к 100-й итерации (рис. 1).

Среди рассматриваемых технологий особо выделяется MPI, т.к. рост производительности наблюдается практически сразу. Одним из объяснений данного явления служит то, что параллельные вычисления с использованием MPI выполняются в различных потоках, и в этом случае ОС более эффективно распределяет нагрузку между экземплярами программы. В случае OpenMP и Windows API распределение ресурсов процессора осуществляется между потоками одного приложения. При этом при малом объеме вычислений ресурсы процессоров могут распределяться ОС неравномерно. Так например, для последовательных вычислений MS Windows 2003 сама определяет, на каком из процессоров будут проводиться вычисления, а какой процессор будет отвечать за работу остальных процессов среды.

В конечном итоге решение о выборе конкретной технологии или нескольких технологий и приемлемости ее применения возлагается на разработчиков и должно приниматься в результате анализа начальных данных, используемых алгоритмов, аппаратной и программной архитектуры системы, опыта программистов и прочих факторов.

Список литературы

1. *Сарданашвили С.А.* Расчетные методы и алгоритмы (трубопроводный транспорт газа). РГУ Нефти и газа им. И.М. Губкина, 2005.
2. *Леонов Д.Г., Швечков В.А.* Организация хранения данных в распределенном вычислительном комплексе при решении задач диспетчерского управления режимами ГТС // Автоматизация, телемеханизация и связь в нефтяной промышленности. 2005. № 9.
3. *Хьюз, Камерон, Хьюз, Трейси.* Параллельное и распределенное программирование на C++. Пер. с англ. М.: Издательский дом "Вильямс". 2004.
4. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2004.
5. *Крюков В.А.* Разработка параллельных программ для вычислительных кластеров и сетей. Институт прикладной математики им. М.В. Келдыша РАН. http://www.keldysh.ru/dvm/dvmhtm1107/publishr/cldvm2002web.htm
6. Reap the Benefits of Multithreading without All the Work. http://msdn.microsoft.com/msdnmag/issues/05/10/OpenMP/default.aspx
7. *Атавин А.А., Карасевич А.М., Сухарев М.Г. и др.* Трубопроводные системы энергетики: модели, приложения, информационные технологии. М.: ГУП Издательство "Нефть и газ". 2000.
8. Что такое OpenMP. http://www.parallel.ru/tech/tech_dev/openmp.html
9. MPI: The Message Passing Interface. http://www.parallel.ru/tech/tech_dev/mpi.html
10. MPICH2 http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm
11. MPI Documents http://www.mpi-forum.org/docs/docs.html

Швечков Виталий Александрович *– инженер РГУ нефти и газа имени И.М. Губкина.*

Контактный телефон (495)930-93-48.

Компании РЕЛЭКС и Науцилус заключили соглашение о партнерстве

Компании намерены объединить свои разработки в целях создания единого решения для систем автоматизации промышленных предприятий, в которых интегрированы системы диспетчерского управления и сбора данных (SCADA) и корпоративные БД, поддерживающие SQL-доступ к данным. В качестве SCADA-системы данное решение предусматривает использование Phocus для ОС РВ QNX или Wizcon для MS Windows, а в качестве системы управления БД – СУБД ЛИНТЕР. Так как ЛИНТЕР работает на различных программно-аппаратных платформах, включая MS Windows, клоны Unix и различные системы РВ (QNX, VxWorks, OS-9, OS/9000, ОС 2000), совместное решение двух компаний позволит пользователям эффективно, надежно и экономично реализовать задачу комплексной автоматизации деятельности производственного предприятия.

Http://www.relex.ru

DaimlerChrysler оснащает свою исследовательскую лабораторию программным обеспечением QNX

Компания DaimlerChrysler оснастит свои исследовательские лаборатории в США и Германии встраиваемым ПО QNX и станет первым участником программы QNX OEM Innovation Labs (OIL) Program, позволяющей автомобилестроительным компаниям раньше всех получать доступ к новейшим технологиям QNX для использования в своих исследовательских группах.

Http://www.swd.ru