Оптимизация исполнения фрагментированных программ на основе профилирования*

В.А. Перепелкин
Институт вычислительной математики
и математической геофизики СО РАН, Новосибирск

Рассматривается задача оптимизации исполнения параллельной программы на основе профилирования в системе фрагментированного программирования LuNA. Предлагается алгоритм анализа профиля и конструирования более эффективного распределения ресурсов. Приводятся результаты тестирования.

При реализации больших численных моделей на суперкомпьютерах необходимо обеспечивать высокую эффективность прикладной параллельной программы (тут и далее эффективность понимается в смысле уменьшения времени выполнения параллельной программы на вычислителе заданной конфигурации). Достижение высокой эффективности прикладной параллельной программы является сложной задачей системного параллельного программирования и требует существенных усилий от программиста, в общем-то, не связанных напрямую с его прикладной задачей, а лишь с её реализацией на конкретном классе вычислителей [2]. В дальнейшем с развитием параллельных вычислителей достижение высокой эффективности прикладных программ будет только усложняться. В связи с этим большое значение имеют системы параллельного программирования, автоматизирующие ту часть работы прикладного программиста, которая связана с системным параллельным программированием. Один из важных подходов в автоматической оптимизации исполнения параллельных программ – это *профилирование* – форма динамического анализа программы, при которой фиксируются некоторые важные особенности поведения программы, такие как продолжительность выполнения операций и коммуникаций, дисбаланс загрузки процессоров и т.п. Собранная статистика называется *профилем программы* и используется как источник дополнительной информации для последующей оптимизации программы.

Автоматическая оптимизация исполнения параллельных программ на основе профилирования является привлекательным подходом, т.к. с точки зрения пользователя означает, что каждый последующий запуск программы будет работать быстрее, чем предыдущий, приближаясь к

* Проект поддержан грантом РФФИ №10-07-00454а.

некоторому субоптимальному значению по времени. Однако профилирование имеет и ряд недостатков, которые будут «унаследованы» и в настоящей работе. Во-первых, профиль программы всегда привязан к конфигурации вычислителя. Это означает, что при последующем запуске прикладной параллельной программы на другом вычислителе профиль становится непригодным. Во-вторых, профиль программы привязан к *входным данным* программы, т.е. при запуске той же программы на вычислителе той же конфигурации но на другом наборе входных данных временны́е характеристики её исполнения могут отличаться. Как следствие, оптимизация на основе профилирования может понизить эффективность программы. Тем не менее, несмотря на описанные недостатки, профилирование является весьма востребованным, т.к. в численном моделировании вполне обычной является ситуация, когда один и тот же численный эксперимент проводится многократно с различными параметрами. При э́том изменение параметров далеко не всегда сказывается на временны́х характеристиках исполнения программы. В некоторых программах (например, операциях над плотными матрицами) конкретные значения переменных вообще практически не влияют на время и ход работы программы. В подобных случаях профилирование является востребованным.

Работа посвящена попытке реализации подхода автоматической оптимизации исполнения параллельных программ на основе профилирования для системы фрагментированного программирования LuNA [3], разрабатываемой в ИВМиМГ СО РАН (г. Новосибирск). Эта система ориентирована на реализацию больших численных моделей для суперкомпьютеров. Одной из особенностей этой системы является представление прикладного алгоритма в виде множеств фрагментов данных и вычислений, представляющих соответственно переменные и операции прикладного алгоритма [1]. Такая фрагментированная структура программы сохраняется и во время исполнения, благодаря чему возможно профилирование программ на уровне фрагментов безотносительно их содержимого (т.е. операций и значений переменных).

Необходимые определения

Здесь приводятся упрощенные формальные определения фрагментированной программы, рекомендаций и профиля. Упрощения сделаны для простоты изложения и не затрагивают существенных аспектов с рассматриваемой точки зрения. Более подробный формализм представлен, например, в [4].

Фрагментированная программа (ФП) – это кортеж $\langle CF, \rho \rangle$, где CF – это потенциально бесконечное множество фрагментов вычислений (ФВ), представляющих агрегированные операции прикладного алгоритма. Каждый ФВ реализуется вызовом некоторой процедуры над конкретным набором значений входных фрагментов данных. Отношение $\rho \subseteq CF \times CF$ выражает информационные зависимости между ФВ. Исполнение ФП состоит в запуске всех ФВ в порядке, не противоречащем ρ, т.е. если $f_1, f_2 \in CF$ и $f_1 \; \rho \; f_2$, то запуск ФВ f_2 можно осуществлять только после того, как завершился запуск ФВ f_1.

Рекомендации – это отображение $L{:}CF \rightarrow PE$, где PE – это множество узлов мультикомпьютера, на которых происходит запуск ФП. Если для некоторого $f \in CF \; L(f)=p$, то это означает, что ФВ f будет запущен на узле p.

Профиль – это пара функций $\langle S,E \rangle$, определённых на множестве CF, которые обозначают соответственно время начала и окончания срабатывания произвольного ФВ.

Исполнение ФП происходит с учётом рекомендаций. Именно исполнительная система, которая исполняет ФП, старается запустить ФВ на тех вычислительных узлах, которые заданы в рекомендациях. Распределение ФВ по узлам существенно влияет на эффективность исполнения ФП, но при этом никак не сказывается на содержательной части работы программы, т.е. не влияет на значения, полученные в ходе вычислений. Таким образом, задавая те или иные рекомендации, возможно влиять на эффективность исполнения программы.

Формально задача оптимизации исполнения ФП на основе профилирования ставится как задача построения нового отображения L' на основе $\langle CF, \rho, L, S, E \rangle$. Неформально можно сказать, что исполнение программы с отображением L' должно занимать меньше времени, чем с отображением L.

Описание алгоритма

Идея алгоритма состоит в том, что профиль анализируется на предмет наличия дисбаланса загрузки вычислительных узлов в те или иные моменты времени. Под дисбалансом понимается ситуация, когда один узел занят исполнением более чем одного ФВ, в то время как другой узел простаивает без работы. При обнаружении дисбаланса загрузки ФВ с перегруженного узла назначаются на другие, свободные вычислительные узлы.

Таким образом, алгоритм направлен на то, чтобы по факту возникновения дисбаланса скорректировать рекомендации, чтобы этот

дисбаланс исчез или стал меньше при следующем запуске. Для положительного эффекта от алгоритма необходимо, чтобы на перегруженных узлах было какое-то количество независимых ФВ. Если же ФВ связаны информационными зависимостями, то их параллельное исполнение невозможно и, следовательно, назначение их на разные узлы не ускорит выполнение программы, а скорее всего несколько замедлит из-за дополнительной нагрузки на коммуникации.

Недостатком алгоритма является отсутствие анализа информационных зависимостей между ФВ и соответственно нагрузок на коммуникационную подсистему. Как следствие, для успешной оптимизации необходимо, чтобы накладные расходы на коммуникационную подсистему были меньше, чем выигрыш, полученный за счёт перераспределения вычислительной нагрузки.

Тестирование

Работа алгоритма была протестирована на задаче умножения плотных матриц, реализованной в системе фрагментированного программирования LuNA. Особенностью задачи является относительно большое количество операций на единицу данных, что делает эту задачу удобной для предложенного алгоритма оптимизации исполнения на основе профилирования. Тестирование проводилось на кластере НКС Сибирского суперкомпьютерного центра (г. Новосибирск). Для вычислительных кластеров характерна высокая производительность сетевой подсистемы по сравнению с другими типами вычислителей. В частности, для такой задачи, как умножение плотных матриц, влияние коммуникаций мало (порядка 10% для матриц размера 6000×6000), что также является выгодным обстоятельством для предложенного алгоритма.

Для сравнения была выбрана реализация той же задачи в MPI, оптимизированная вручную с использованием алгоритма блочного умножения матриц. Схема распараллеливания была взята из систолических алгоритмов умножения матриц для двумерной решетки вычислительных узлов. Во ФП в качестве фрагментов кода использовалась та же процедура блочного умножения матриц. Таким образом, ручная реализация являлась теоретически достижимым пределом по времени работы для ФП. Отставание по времени работы ФП связано с двумя причинами. Во-первых, это неоптимальность в распределении ресурсов (предмет исследования) и, во-вторых, накладные расходы на работу самой исполнительной системы. На рис. 1 изображена зависимость времени работы программы для последовательности запусков. Каждый

следующий запуск был оптимизирован описанным алгоритмом на основе предыдущих запусков. Начальные рекомендации (на первом запуске) выбирались различными в очень широком диапазоне, вплоть до случайного распределения или назначения всех ФВ на один узел. Но каждый раз наблюдалась одна и та же картина, приведенная на рис. 1.

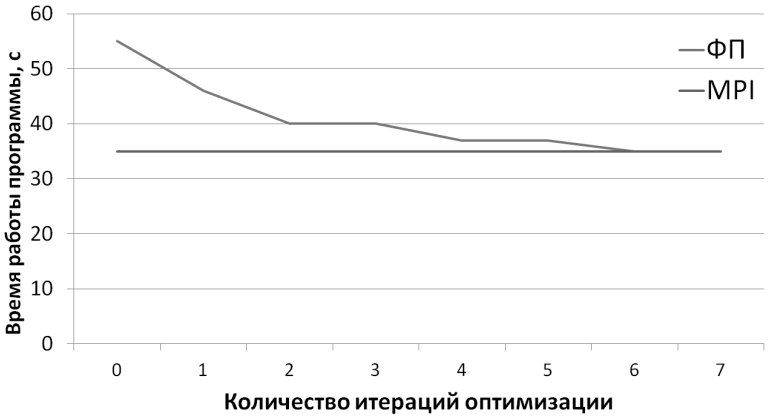

Рис. 1. Результаты тестирования

Как видно из графика, время работы программы уменьшается с каждым следующим запуском, доходя практически до времени работы ручной реализации в MPI. Этот результат означает, что для этой задачи одного только предложенного алгоритма оказалось достаточно, чтобы автоматически сконструировать распределение ресурсов, по качеству не уступающее ручной реализации. Разумеется, не для каждой задачи можно ожидать такого результата. Во-первых, в задаче существенную часть вычислений должны составлять массовые вычисления (т.е. однотипные независимые операции). Во-вторых, определяющими для времени работы программы («узким местом») должны быть вычисления, а не коммуникации. Задача умножения матриц оказалась подходящей по обоим этим критериям. Для успешной работы на других классах задач подход оптимизации исполнения ФП должен быть усовершенствован с учетом вышеописанных проблем.

Заключение

Рассмотрен подход автоматической оптимизации исполнения параллельных программ на основе профилирования в системе фрагмен-

тированного программирования LuNA. Предложен алгоритм конструирования рекомендаций на основе профиля исполнения ФП. Предложенный алгоритм реализован и протестирован на задаче умножения плотных матриц. Показана целесообразность использования рассматриваемого подхода. В дальнейшем планируется усовершенствование алгоритма автоматического конструирования рекомендаций на основе профиля исполнения ФП.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 127 с.
2. **Kraeva M.A., Malyshkin V.E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers //Int. Journal on Future Generation Computer Systems. – 2001. – Vol. 17, № 6. – P. 755–765.
3. **Malyshkin V.E., Perepelkin V.A.** LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem //Proceedings of the 11th Conference on Parallel Computing Technologis, 2011, LNCS 6873. – N.Y.: Springer. – P. 53–61.
4. **Malyshkin V.E., Perepelkin V.A.** Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system //The Journal of Supercomputing. – N.Y.: Springer, 2011. – P. 1–14. DOI: 10.1007/s11227-011-0649-6.