ISBN 5-94052-127-4 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2006

УДК 519.68

С. М. Абрамов, И. М. Загоровский, М. Р. Коваленко,
Г. А. Матвеев, В. А. Роганов

Миграция от MPI к платформе OpenTS: эксперимент с приложениями PovRay и ALCMD

Аннотация. В статье мы рассматриваем известные параллельные приложения, широко используемые на различных суперкомпьютерных платформах — приложение рендеринга изображений с использованием метода трассировки лучей PovRay и приложение, моделирующее молекулярную динамику ALCMD.

Ключевые слова и фразы: T-система, распараллеливание, OpenTS, технология, инструментальные средства, кластерная реализация, язык программирования, платформа, PovRay, ALCMD.

Введение

Мотивации перехода от MPI к высокоуровневым средствам организации параллельного счета сходны с теми же мотивациями, которые были во времена перехода от языка программирования ассемблер к высокоуровневым языкам программирования.

Мы уверены, что новейшие технологии высокоуровневого распараллеливания могут использоваться для получения эффективных и надежных реализаций многих реальных суперкомпьютерных приложений, значительно сократят усилия на программирование в сравнении с использованием просто MPI.

Здесь мы рассматриваем известные параллельные приложения, широко используемые на различных суперкомпьютерных платформах — приложение рендеринга изображений с использованием метода трассировки лучей PovRay и приложение, моделирующее молекулярную динамику ALCMD. Цель данной работы — исследование следующего вопроса: является ли подход, основанный на применении

Работа выполнена при поддержке Программы фундаментальных научных исследований ОИТВС РАН «Оптимизация вычислительных архитектур под конкретные классы задач, информационная безопасность сетевых технологий» и РФФИ (грант РФФИ № 05–07–08005).

OpenTS более эффективным, чем подход, основанный на применении MPI для реализации данных приложений.

1. Технология распараллеливания OpenTS

Система OpenTS [1–3] является технологией распараллеливания программ, базирующейся на идеологии и принципах T-Системы. Она была разработана в рамках суперкомпьютерного проекта «СКИФ» (2000–2004 годы).

Система OpenTS [4] предоставляет язык программирования T++ (очень простой параллельный диалект C++) и наследует бесконфликтную сущность распараллеливания фукциональной парадигмы. Будучи императивным внутри тел T-функций, T++ язык позволяет смешивать чистую функциональную и императивную мощь C++ на уровне T-вызовов. Подробное описание OpenTS может быть получено на главном сайте проекта «СКИФ»: `http://skif.pereslavl.ru`.

2. Приложение PovRay

Известное приложение PovRay (Persistence Of Vision), широко применяемое для получения реалистичных изображений, использует технологию рендеринга изображений — трассировку лучей. Пакет PovRay распространяется с исходным кодом, который эволюционировал от C к C++ в течение последних лет. Так как трассировка лучей требует много вычислительных ресурсов даже для простых сцен, различными авторами было разработано несколько параллельных версий PovRay. Для распараллеливания работы PovRay на мультикомпьютерах использовались различные подходы: от тривиальных rsh-скриптов, вызывающих исполняемый файл PovRay для фрагментов сцены на разных вычислительных узлах, до более эфективных реализаций на PVM и MPI, поддерживающих динамическую балансировку загрузки и даже оригинальную фукциональность, такую как анимация и функции интерактивного вывода на экран.

2.1. MPI распараллеливание. Существует две наиболее известных MPI адаптации для PovRay:

- MPI-PovRay Povray 3.1g;
- ParaPov PovRay 3.50c.

Обе MPI-адаптации в действительности являются программными вставками в соответствующий исходный код PovRay.

Присутствующие комментарии в исходном коде говорят о наличии ошибок, все еще присутствующих на некотых MPI платформах. Описание сценариев зависания прораммы были повторены при нашем тестировании MPI-PovRay.

Тем не менее, после более детального изучения мы обнаружили, что реальный смысл реализованного MPI кода полностью соответствует модели динамического распределения задач, уже существующей в технологии T-Системы.

Но эта логика была закодирована вручную, с ручной синхронизацией и пересылкой данных во многих местах, и различными способами в нескольких исходных файлах PovRay. Как результат — сложность реализации и наличие скрытых ошибок.

Другое слабое место реализации — не поддержан режим фантомного мастера.

Есть и другой минус: не реализован режим фантомного мастера, когда мастер процессор работает также, как и рендерер, что объяснятся сложностью аккуратной организации такого режима. Более того, MPI не имеет функциональности, подобной сигналам, поэтому если заставить управляющий MPI процесс формировать изображение, то ему будет сложно своевременно реагировать на запросы рабочих процессов.

Общий размер файлов MPI кода — более чем 1500 линий кода, без учета множественных изменений в других файлах, выполненых для адаптирования кода к вычислениям в параллельной среде. Изучение MPI изменений в PovRay 3.1g представляет сложную задачу для исследователя. Идея минимизации изменений оригинального кода при MPI адаптации релизована для PovRay 3.50c на C++, но привела к увеличению кода — около 3000 строк.

2.2. Распараллеливание программ при помощи T-Системы. Для корректности сравнения мы использовали одни и те же версии PovRay (обе 3.1g и 3.50c).

Распараллеливание программ при помощи T-Системы заключалось в удалении ненужного MPI-кода по управлению задачей, и замена его только двумя T-функциями.

Управляющая T-функция разбивает целевое изображение на соответствующее число фрагментов и собирает воедино результаты работы рабочих T-фукнций.

Рабочая T-функция подставляет параметры рендеринга в опции PovRay и вызывает главную подпрограмму рендеринга PovRay.

Некоторый дополнительный код нужен для корректной инициализации и интерактивного просмотра сформированного изображения.

Файл с исходным кодом T-программы — меньше 200 строк. Сделано несколько незначительных изменений в одном файле PovRay в соответствии со спецификой параллельных вычислений.

3. Сравнение производительностей MPI-PovRay и OpenTS-PovRay

3.1. Сценарий тестирования. Для тестирования использовалась сцена шахматной доски, взятая из оригинального дистрибутива PovRay, со следующими опциями — ширина сцены 1500 и высота сцены 1500. Сцена шахматной доски описывается 430 примитивами в объектах и 41 в ограничивающих формах.

Все тестируемые реализации PovRay — оригинальная C -реализация (для одного процессора), MPI-реализация и T-реализация — были скомпилированы и собраны одними и теми же компилятором и редактором связи с одинаковыми опциями препроцессирования, оптимизации и линковки.

C-реализация (C-PovRay) запускалась на одном процессоре. MPI-реализация (MPI-PovRay) и T-реализация (T-PovRay) запускались на кластере с использованием от 1 до 16 процессоров.

C-PovRay, MPI-PovRay и T-PovRay, использованные при тестировании, базируются на PovRay 3.50c.

3.2. Погрешность вычисления быстродействия. В процессе наших исследований мы отметили, что дополнительные библиотеки, такие как MPI или OpenTS, слинкованные с кодом PovRay, могут уменьшить быстродействие на 5–15% по отношению к оригинальному C-PovRay. Существует несколько возможных причин: увеличение числа команд, выпадающих из кэша, изменение загрузки данных (пропадание данных из кэша и буфера быстрого преобразования адреса) и т. д. Причины также могли объясняться зависимостью эффекта от типа процессора (см. таблицу 1).

Один процессор, изменение быстродействия	AMD Athlon MP 1800+	AMD Opteron 248 2.2 GHz
MPI-Povray 3.50c vs. C-Povray 3.50c	+0.3%	+5.6%
T-Povray 3.50c vs. C-Povray 3.50c	+15.7%	+4.3%

Таблица 1. Разница производительности программ на разных типах процессоров

Один процессор, изменение быстродействия	200x200 pixels	1500x1500 pixels
MPI-Povray 3.50c vs. C-Povray 3.50c	+1.2%	-2.5%
T-Povray 3.50c vs. C-Povray 3.50c	-5.2%	-6.1%

Таблица 2. Сравнение производительности разных программ на разных обьемах данных

Более того, несколько наших предыдущих экспериментов (PovRay 3.50c на AMD Opteron 248) демонстрировали преимущество T-Povray и MPI-PovRay над оригинальным однопроцессорным C-Povray (см. таблицу 2).

Описываемый эффект не устойчив, и изменение быстродействия различно от запуска к запуску. Согласно нашим опытам, скорость выполнения может варьироваться от 5% до 10% от среднего времени одного и того же приложения даже для однопроцессорных приложений.

3.3. Корректность результатов тестирования. Результаты (файл изображения) всех тестов сравнивались с мастер-изображением C-PovRay. Было найдено небольшое, но ожидаемое отличие, возникающее из-за различий в разбиении изображения. Разное разбиение приводит к различию в результатах глобальных операций — фокусное размывание и подобных ему. Тот же эфект имел место как для MPI-Povray, так и для T-Povray.

Таким образом, мы убедились в корректности результатов тестирования всех наших прогонов: функционирование кода не изменилось

при переходе от MPI-распараллеливания к OpenTS-распараллеливанию.

3.4. Регистрация результатов тестирования. Для всех тестовых прогонов измерялись и вычислялись следующие величины:

- $time_C$, $time_{MPI}(N)$, $time_T(N)$ — времена выполнения в секундах C-реализации, MPI-реализации и T-реализации соответственно, которое зависит от числа процессоров N, используемых в тестовом запуске.
  - $time_C$ измерялось для $N = 1$;
  - $time_{MPI}(N)$ измерялось для $N \in [1 \dots 16]$, где для $N > 1$ один процессор использовался управляющей («master») подпрограммой, и процессоры использовались исполнителями («workers»);
  - $time_T(N)$ измерялось для $N \in [1 \dots 16]$;
- $time\%_{MPI}(N) = \dfrac{time_{MPI}(N)}{time_C}$ вычислялось для $N \in [1 \dots 16]$; обычно значение этой функции находится в диапазоне
$$0\% < time\%_{MPI}(N) \le 100\%$$
- $time\%_T(N) = \dfrac{time_T(N)}{time_C}$ вычислялось для $N \in [1 \dots 16]$;
- $CoE_{MPI}(N) = \dfrac{1}{N \times time\%_{MPI}(N)}$ — коэффициент эффективности MPI-реализации (вычислялся для $N \in [1 \dots 16]$). Обратим внимание, что $K_{MPI}(N) = \dfrac{1}{time\%_{MPI}(N)}$ показывает, насколько $time_{MPI}$ меньше $time_C$ (то есть коэфициент ускорения MPI-реализации) и коэффициент полезного действия $CoE_{MPI}(N) = \dfrac{K_{MPI}(N)}{N}$ показывает, насколько ускорение $CoE_{MPI}(N)$ отклоняется от линейного ускорения N.
- $CoE_T(N) = \dfrac{1}{N \times time\%_T(N)}$ — коэфициент эфективности T-реализации (вычислялся для $N \in [1 \dots 16]$).
- $\dfrac{time_{MPI}(N)}{time_T(N)}$ вычислялось для $N \in [1 \dots 16]$. Если время выполнения MPI-PovRay больше, чем время выполнения T-PovRay (MPI-PovRay медленнее), то отношение $> 100\%$ и наоборот.

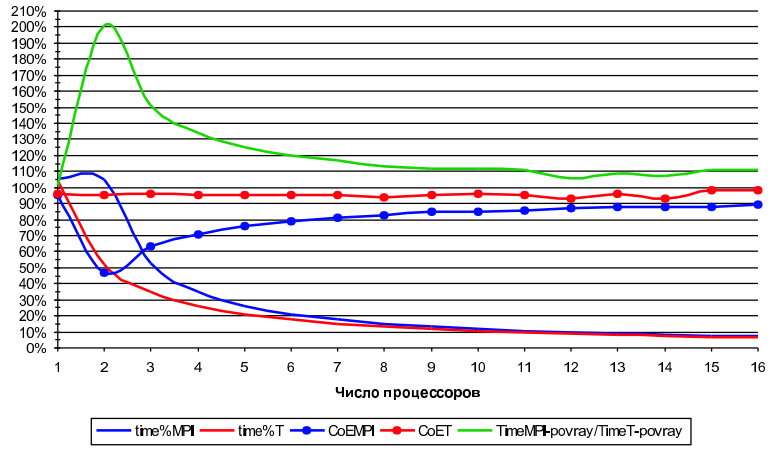

Рис. 1. Результаты тестирования MPI-PovRay и T-PovRay на кластере «СКИФ К-1000»

3.5. Тестирование на кластере «СКИФ К-1000». Этот раздел содержит результаты тестирования на кластере «СКИФ К-1000», имеющем следующую конфигурацию:

- операционная система Linux;
- ядро — 2.6.5;
- 286 узлов; каждый узел: 2 процессора AMD Opteron 248 2.2, GHz оперативная память 4GB, жесткий диск 80GB.

Все реализации PovRay — оригинальный C-PovRay, MPI-PovRay и T-PovRay — были собраны компилятором gcc 3.3.3 с одинаковыми опциями препроцессирования, оптимизации и сборки. На рисунке 1 представлены результаты тестирования под LAM 7.1.1 на Gigabit Ethernet для $N = 1 \ldots 16$.

4. Приложение ALCMD

Пакет ALCMD [5] предназначен для моделирования молекулярной динамики. Доступны исходные коды пакета на языке Fortran-77. Пакет ALCMD использует легковесную MPI библиотеку (см. Рис. 2, левая часть), называемую MP_Lite [6].

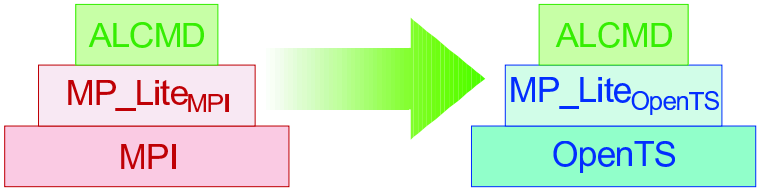

Рис. 2. Миграция ALCMD от MPI к платформе OpenTS

Исходную реализацию ALCMD мы будем обозначать как MPI-ALCMD. В рамках данной работы была разработана реализация пакета ALCMD, названная нами OpenTS-ALCMD (см. Рис. 2, правая часть), следующим образом:

(1) Оригинальная библиотека MP_Lite переписана на OpenTS: $\text{MP\_Lite}_{\text{MPI}} \rightarrow \text{MP\_Lite}_{\text{OpenTS}}$;
(2) Код на Fortran для ALCMD оставлен нетронутым, он был собран с нашей реализацией библиотеки MP_Lite, которую мы будем называть $\text{MP\_Lite}_{\text{OpenTS}}$ и с библиотекой поддержки периода исполнения системы OpenTS. Таким образом, мы получили новую версию пакета ALCMD, реализованного с использованием системы OpenTS.

Для сравнения OpenTS и MPI реализаций ALCMD, мы сделали следующее:

(1) Было выполнено некоторое количество тестов для разных входных данных на разных кластерных установках; результаты этих тестов представлены и проанализированы в этой статье;
(2) Оценивались производительность MPI-ALCMD и OpenTS-ALCMD, сравнивались сложность и размер исходного кода $\text{MP\_Lite}_{\text{MPI}}$ и $\text{MP\_Lite}_{\text{OpenTS}}$.

4.1. Распараллеливание программ при помощи T-Системы. При использовании T-Системы существуют разные способы распараллеливания ALCMD кода. Мы рассматриваем здесь простой способ: сохранение оригинального ALCMD Fortran кода и замена MP_Lite кода (написанного поверх MPI) новым разработанным T++ модулем (см. Рис. 2).

Оригинальный размер библиотеки MP_Lite составляет более чем 20000 строк C кода. Мы выбрали из MP_Lite подмножество функций, которые используются в Fortran коде ALCMD. Выбранное подмножество исходного кода MP_Lite, которое дает возможность запускать ALCMD поверх MPI, в самом деле составляет около 3500 строк. Поддержка всех функций (необходимых для запуска ALCMD) нашим абстрактным уровнем выполнения ($\text{MP\_Lite}_{\text{OpenTS}}$) составляет меньше, чем 500 строк кода на языке T++, что в 7 раз меньше, чем аналогичный код (3500 строк), написанный с использованием библиотеки MPI.

5. Сравнение производительностей MPI-ALCMD и OpenTS-ALCMD

Тест выполнялся с двумя наборами входных данных:

- *md.256K.in* — запрос на моделирование системы из 256,000 атомов;
- *md.512K.in* — запрос на моделирование системы из 512,000 атомов.

Тест выполнялся на разных кластерах:

- Кластер «СКИФ Первенец-М», использующий для передачи данных Fast Ethernet и SCI-соединения;
- Суперкомпьютер «СКИФ К-1000», использующий для передачи данных Gigabit Ethernet и Inifiband-соединения.

Все программы, реализованные нами (SCALAR-ALCMD, MPI-ALCMD, OpenTS-ALCMD), были собраны одним и тем же компилятором и редактором связи с одинаковыми опциями.

5.1. Регистрация результатов тестирования. Для всех исполняемых тестов измерялись и вычислялись такие же величины, как те, что были рассмотрены в разделе 3.4.

5.2. Тестирование на кластере «СКИФ К-1000». Этот раздел содержит результаты теста, который выполнялся на кластере «СКИФ К-1000».

На рисунке 3 представлены результаты тестирования (моделирование системы из 512,000 атомов) под LAM 7.1.1 на Gigabit Ethernet для $N = 1 \ldots 16$.

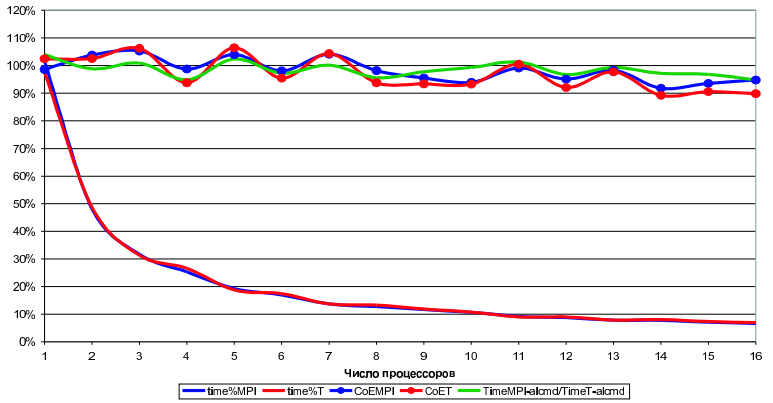

Рис. 3. Результаты тестирования MPI-ALCMD и OpenTS-ALCMD на кластере «СКИФ К-1000»

Значение Time% вычислено относительно времени выполнения однопроцессорной версии SCALAR-ALCMD. CoE вычислен как отклонение мультипроцессорной версии от линейного ускорения однопроцессорной версии SCALAR-ALCMD.

Выводы

Наши исследования показали следующее:

(1) Параллельная реализация PovRay под OpenTS отличается компактностью и простотой в сравнении с параллельной реализацией под MPI:
   - MPI код PovRay 3.1g — более 1500 строк;
   - MPI код для PovRay 3.50c — около 3000 строк;
   - T++ код — менее 200 строк; дополнительно добавлены незначительные изменения в оригинальный код в соответствии со спецификой параллельных вычислений.

(2) Эффективность параллельной реализации T-PovRay выше эффективности MPI-PovRay.

(3) Реализация MP_Lite с OpenTS более компактная и намного проще, чем реализация MP_Lite с MPI:
   - исходный код MPI-версии содержит 3500 строк;
   - исходный код OpenTS-версии содержит менее 500 строк кода на языке T++.

(4) Эффективности параллельных реализаций MPI-ALCMD и OpenTS-ALCMD вполне сравнимы.

Тем самым, система OpenTS показала себя как средство для эффективной реализации приложений, а также прикладных библиотек.

Список литературы

[1] Абрамов С. М., Адамович А. И., Инюхин А. В., Московский А. А., Роганов В. А., Шевчук Ю. В., Шевчук Е. В. *Т-система с открытой архитектурой* // Суперкомпьютерные системы и их применение SSA'2004: Труды Международной научной конференции, 26–28 октября 2004 г. Минск, ОИПИ НАН Беларуси. — Минск, 2004, с. 18–22.

[2] Abramov S., Adamovich A., Inyukhin A., Moskovsky A., Roganov V., Shevchuk E., Shevchuk Yu., Vodomerov A. *OpenTS: An Outline of Dynamic Parallelization Approach* // PACT'2005, LNCS, 2005.

[3] Afonin S., Shundeev A., Roganov V. *Semistructured data search using dynamic parallelisation technology* // Proceedings MIRPO 2003, 2003, с. 152–157.

[4] Roganov V., Slepuhin A. *Distributed Extension of the Parallel Graph Reduction. GRACE: Compact and Efficient Dynamic Parallelization Technology for the Heterogeneous Computing Systems* // International Conference on Parallel and Distributed Processing Techniques and Applications. — Las Vegas, Nevada, USA, June 25.

[5] Сайт пакета молекулярной динамики ALCMD: Электронный сетевой ресурс, `http://www.cmp.ameslab.gov/cmp/CMP_Theory/cmd/alcmd_source.html`.

[6] Официальный сайт библиотеки MP_Lite: Электронный сетевой ресурс, `http://www.scl.ameslab.gov/Projects/MP_Lite/`.

Исследовательский центр мультипроцессорных систем ИПС РАН

S. M. Abramov, M. R. Kovalenko, G. A. Matveev, V. A. Roganov, I. M. Zagorovsky. *Migrating from MPI to OpenTS Platform: experience with the PovRay and ALCMD applications.* (in Russian.)

Abstract. In this article we consider two well-known parallel applications, widely used on different supercomputing platforms — ray tracing application PovRay and molecular dynamics simulation application ALCMD.