С.Н. Карпенко, А.В. Дергунов

Нижегородский государственный университет им. Н.И. Лобачевского

АВТОМАТИЗАЦИЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ MPI-ПРИЛОЖЕНИЙ

Цель параллельных вычислений – решить задачу быстрее. Чтобы определить, где именно тратится время на вычисления в параллельной программе, т.е. найти узкие места производительности, используются инструменты для измерения и визуализации производительности. Пример такого инструмента – Intel Trace Analyzer and Collector, который обеспечивает визуализацию трассы выполнения программы. Но трасса параллельной программы зачастую бывает очень большой, поэтому ее сложно анализировать вручную. Разработанная автором система позволяет автоматизировать повышение производительности в MPI-программах.

Повышение производительности MPI приложений. Процесс повышения производительности MPI-программ с использованием измерительных инструментов, собирающих трассу выполнения приложения, состоит из следующих задач:

1. Выбор метрик или параметров оценки производительности программы. В данной работе в качестве такой метрики рассматривается общее время работы программы.
2. Сбор данных производительности.
3. Анализ данных производительности программы на предмет выявления проблем производительности.
4. Выявление причин возникновения проблем производительности.
5. Принятие решения о способе устранения проблем производительности
6. Устранение причин возникших проблем производительности.

Этот процесс может повторяться до достижения требуемой производительности. Разработанная автором система предназначена для автоматизации 3, 4 и 5 задач.

Для повышения производительности MPI-приложений система анализирует коммуникационный профиль приложения [1]. Длина сообщений, интенсивность передачи сообщений в разные моменты времени работы программы, топология передач сообщений между процессорами, накладные расходы на организацию взаимодействия, масштабируемость коммуникационной структуры программы, структура синхронных и асинхронных посылок – эти и ряд других характеристик определяют коммуникационный профиль приложения. Повышенные накладные расходы MPI-программы могут быть препятствием к достижению хорошей масштабируемости и часто являются следствием плохо реализованного коммуникационного профиля.

Ниже перечислены некоторые способы улучшения коммуникационного профиля приложения и, как следствие, повышения производительности MPI-программ:

- сокращение количества пересылаемых данных и по возможности пересылка одного большого сообщения вместо нескольких маленьких;
- отправка данных как можно раньше, а также маскирование ожидания приема данных с использованием неблокирующих операций MPI;
- использование топологий MPI, чтобы эффективно распределить процессы между процессорами параллельной вычислительной системы;
- распараллелить вычисления и назначить процессам процессоры так, чтобы сбалансировать вычислительную нагрузку;
- в некоторых случаях имеет смысл продублировать вычисления определенных данных вместо его пересылки и т.д.

Общая архитектура системы. Схема работы системы представлена на рис. 1. Исходными данными является трасса выполнения MPI-приложения (полученная, например, с помо-

щью Intel Trace Collector). Трасса конвертируется в специальный формат, в котором выполнение MPI-приложения представлено набором операций и их параметров. Входные данные анализируются модулем автоматического анализа проблем производительности. Для этого используется описание проблем производительности на специальном языке. Результат анализа – список обнаруженных проблем производительности с указанием степени их влияния на общее время работы приложения.

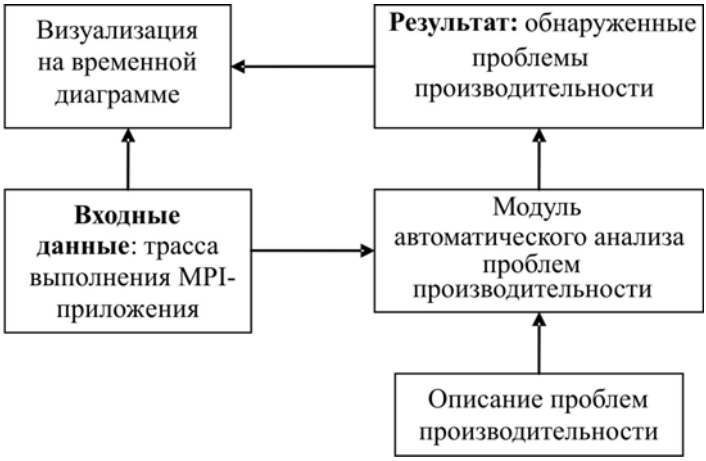

Рис. 1. Схема работы системы

Описание трассы выполнения MPI-приложения. В рамках разработанной системы результат запуска MPI-программы представляется с помощью общей статистики (общее время работы программы, количество процессов и т.д.) и набором коммуникационных операций, выполненных при ее работе, и параметров этих операций.

Ниже приведен простой пример описания работы MPI-приложения:

```
program
    total_time               2.471200748267274
    total_communication_time 2.003308519420723;
```

```
operation point_to_point
    send_op_name          "MPI_Ssend"
    send_op_type          "blocking"
    receive_op_name       "MPI_Recv"
    receive_op_type       "blocking"
    send_start_time       0.4687189432899993
    send_finish_time      2.4709433821505122
    receive_start_time    2.4695157509991628
    receive_finish_time   2.469596837040292
    send_process_rank     0
    receive_process_rank  1
    send_source_code      "latereceiver.cpp:20"
    receive_source_code   "latereceiver.cpp:31";
```

Здесь описывается программа с двумя процессами. За время работы приложения была осуществлена одна операция двухточечного обмена данными («operation point_to_point»). Процесс номер 0 переслал данные с помощью процедуры MPI_Ssend процессу номер 1, который принял эти данные с использованием процедуры MPI_Recv.

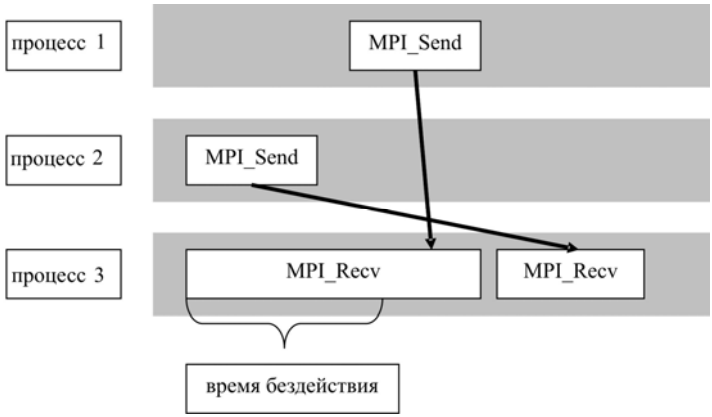

Рис. 2. Прием сообщений в неверном порядке

Язык описания типов проблем производительности в MPI-приложениях. В рамках системы разработан язык для описания типов проблем производительности, встречающихся в MPI-приложениях при использовании коммуникационных

операций. Рассмотрим в качестве примера проблему производительности «Прием сообщений в неверном порядке». Эта проблема возникает, когда порядок посылки и приема сообщений не совпадают. Из-за этого блокирующие операции приема затрачивают лишнее время на ожидание приема соответствующего сообщения. Схематично эта проблема представлена на рис. 2.

Рассмотрим, как описывается эта проблема производительности на разработанном языке:

```
    problem "Прием сообщений в неверном порядке"
        description "Порядок посылки и приема сообщений
не совпадают"
        find op1 type point_to_point
        find op2 type point_to_point
        where
            op1.receive_op_type eq "blocking" and
            op2.receive_op_type eq "blocking" and
            op1.receive_process_rank                    =
op2.receive_process_rank and
            op1.receive_start_time                      <
op2.receive_start_time and
            op1.send_start_time  >  op2.send_start_time
and
            op1.send_start_time                         >
op1.receive_start_time
        severity
            (op1.send_start_time                        -
op1.receive_start_time) /
            total_communication_time;
```

Здесь осуществляется поиск двух операций типа «point_to_point», которые удовлетворяют следующим условиям:

- тип принимающей процедуры обеих операций – блокирующий (т.е. процедура приема данных обязательно блокируется до начала посылки данных);
- прием данных в этих операциях осуществляется на одном и том же процессе;
- прием данных во время первой операции произошел раньше приема данных для второй операции;
- посылка данных во время первой операции, напротив, произошла позднее посылки данных для второй операции;

• в рамках первой операции посылка данных произошла позднее начала их приема, что обеспечивает ожидание.

Если проблема обнаружена, то для нее вычисляется степень ее влияния на производительность операций обмена в MPI-программе как разница между временем начала посылки данных и временем начала приема данных, поделенное на суммарное время, затраченное всеми процессами приложения при вызове коммуникационных операций MPI. Это значение указывает, в какой мере можно улучшить производительность программы, если избежать этой проблемы.

Предложены модель выполнения MPI-приложения и язык описания типов проблем производительности при взаимодействии процессов. Приведен пример описания на этом языке одной проблемы производительности: прием сообщений в неверном порядке. Использование предложенного языка позволяет автоматизировать анализ трассы MPI-программы и находить часто встречающиеся проблемы производительности. В разработанной на основе предложенной модели системе повышения производительности MPI-приложений предусмотрена возможность расширения количества анализируемых проблем с помощью описания новых типов операций и типов проблем производительности. Анализ степени влияния на производительность позволяет ранжировать обнаруженные проблемы и позволяет начать с решения наиболее важных из них. В [2] перечислены типы проблем производительности, которые были описаны на рассмотренном языке.

Список литературы

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.

2. Карпенко С.Н., Дергунов А.В. Модели и программные средства повышения производительности MPI-приложений // Технологии Microsoft в теории и практике программирования: материалы конф. – Н.Новгород: Изд. Нижегород. гос. ун-та, 2009. – С. 188–192.