ИНСТРУМЕНТАЛЬНАЯ СРЕДА ПОДДЕРЖКИ ПРАКТИКУМА ПО ПАРАЛЛЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Н.Н. Попова, В.Ю. Воронова

Говоря о «больших» вычислительных задачах, часто имеют в виду программы, требующие очень мощных вычислительных ресурсов. Как правило, время выполнения таких задач – от нескольких часов до нескольких суток (и даже недель); эти задачи имеют множество входных параметров, и эффективность их выполнения на многопроцессорных системах во многом зависит от выбранной комбинации входных параметров задачи.

Для анализа поведения и оценки эффективности параллельных приложений предназначены, прежде всего, отладчики и трассировщики параллельных программ – такие инструментальные средства, как Jumpshot, Pablo, Kojak и др. Они предоставляют не только аналитическую информацию о трассах программ, но и удобные графические средства для их визуального анализа. Общая проблема использования таких трассировщиков для больших вычислительных задач состоит в том, что при запусках на большом количестве процессоров приходится иметь дело с файлами трасс очень больших объемов (как правило, количество файлов с трассами совпадает с числом процессоров, на которых производился счет). Это затрудняет визуальный анализ поведения программы (не все визуализаторы способны работать с файлами очень больших объемов), а аналитическое исследование становится и вовсе невозможным. Кроме того, трассировщик может изменить поведение программы; формат трасс не унифицирован и обычно ориентирован на конкретные библиотеки параллельного программирования. Иногда возникает необходимость сделать выводы о поведении программы, запуская ее на нескольких вычислителях, что также осложняет сбор и последующий анализ полученных данных.

Эти проблемы могут быть решены системами, позволяющими проводить анализ без вмешательства в код программы; они собирают, хранят и анализируют данные о параллельном приложении, абстрагируясь от особенностей конкретных трассировщиков и платформ. Примером подобной системы может служить, например, SUMS [1] – инструментальная среда для сопровождения процесса разработки параллельных приложений. Она позволяет отслеживать динамику изменения исходного кода программы, ее отладку, собирая при этом данные от пакетных планировщиков о времени ее выполнения, количестве запускаемых процессоров и других характеристиках задачи. Впоследствии с помощью собранных данных проводится анализ “истории” развития параллельного приложения и его эффективности.

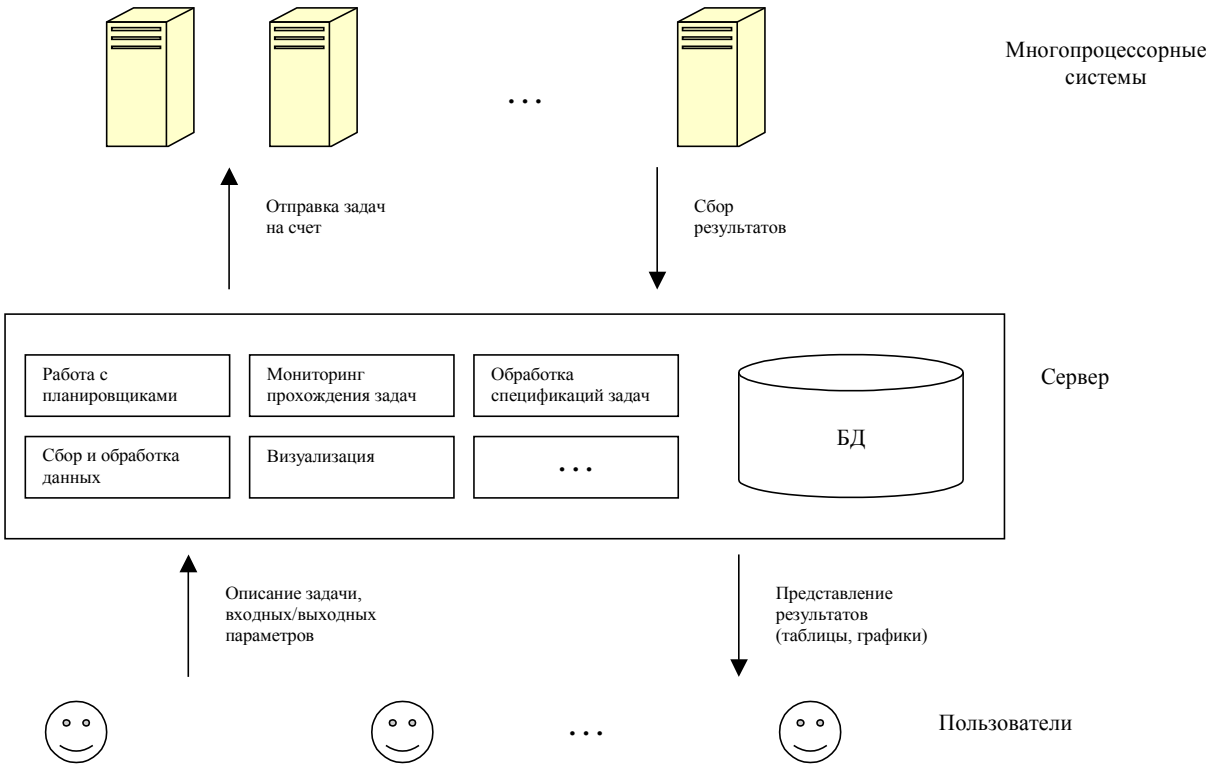

Рис. 1. Архитектура системы

Представляемая нами инструментальная система поддержки параллельных вычислений так же строит свою работу на сборе «истории» выполнения программ, но в данном случае речь идет не о процессе разработки параллельной программы, а скорее об отладке ее эффективности, зависящей от комбинации входных параметров. Создание такой системы было обусловлено, прежде всего, необходимостью организации работ студенческого практикума по параллельному программированию.

Данная система может применяться для исследования поведения не только небольших демо-программ, которыми обычно иллюстрируются курсы по основам технологий параллельного программирования, но и «больших» научных приложений. Предоставляется довольно удобный набор средств для исследования ускорения и эффективности параллельных программ и сравнения основных характеристик их выполнения на разных многопроцессорных системах. Планируется провести эксперимент по применению данной инструментальной системы для поддержки практикума по параллельным вычислениям в 2008/2009 учебном году на факультете ВМиК МГУ.

Архитектура системы схематично представлена на рис. 1. Пользователь может работать сразу с несколькими многопроцесорными комплексами. Связь с ними может осуществляться как через машину-сервер, так и напрямую. Система включает в себя средства для обработки спецификаций пользовательских задач, сбора результатов и данных о задачах и их графического представления. После того, как поставленные в очередь задачи закончат свое выполнение, результаты их работы будут записаны в серверную СУБД. Работа с вычислителями и базой данных осуществляется с помощью команд, встроенных в систему.

Функциональность системы основана на стандартных командах операционных систем семейства Unix и базовых возможностях пакетных планировщиков. Для хранения данных используется СУБД MySQL.

Необходимо заметить, что параллельная программа к моменту начала работы с системой должна быть рабочей, полностью отлаженной. Предвареительно пользователь должен формально описать задачу с точки зрения количества и типа параметров, их возможных значений; также формально должны быть описаны выходные характеристики программы. Некоторые ограничения накладываются и на формат вывода программы.

В систему входят следующие утилиты:

- Генераторы спецификаций задач - утилиты-«помощники» для создания формального описания задачи по заданным пользователем параметрам;
- Команды, вносящие необходимые изменения в базу данных при изменении формата задачи (например, при изменении количества параметров);
- Постановка задач в очередь на счет. Эта команда автоматически “перебирает” все возможные комбинации входных параметров и ставит задачи с заданными параметрами в очередь. Таким образом, после окончания работы этих программ будет собрана статитстика по всем возможным запускам приложения;
- Мониторинг состояния выполнения задач – оповещение пользователя о состоянии поствленных в очередь задач;
- Команды для сбора выходных данных (и записи этой статистики в базу данных) уже отработавших задач;
- Генераторы отчетов и визуального представления собранной статистики по заданным пользователем характеристикам.

Предлагаемая система может использоваться для работы с многопроцессорными системами, вообще говоря, с любой пакетной системой управления задач (тестирование проводилось на многопроцессорных системах, работающих под управлением LoadLeveler, Sun Grid Engine и планировщиками суперкомпьютеров МСЦ – МВС-15000ВМ [4] и МВС-100k [5]). Кроме того, она предоставляет пользователю возможность анализировать поведение задач с любым количеством входных и выходных параметров; значительно облегчается работа пользователя по запуску задач – все возможные комбинации параметров будут автоматически учены при запуске, что избавит пользователя от их перебора вручную. У пользователя не возникнет необходимости изучать огромные текстовые файлы: отчеты, как в текстовом, так и в графическом представлении быстро генерируются по заданным характеристикам. В систему заложена возможность расширения: можно увеличивать число исследуемых параметров, дополнительно подключать сбор аппаратных характеристик и т.д.

Также к особенностям данной системы можно отнести возможность анализа эффективности задач, требующих для выполнения очень большого числа процессоров (терафлопные вычисления), и поддержку работы сразу с несколькими многопроцессорными системами. Кромер того, систему можно использовать для тестирования многопроцессорных систем и исследования межплатформенной эффективности, поскольку предоставляется возможность сравнивать результаты выполнения программ (в том числе различных бенчмарков) на различных платформах.

В качестве примеров работы с системой приведем две программы: учебное MPI-приложение, реализующее умножение матрицы на вектор, и тест HPL (переносимый высокопроизводительный тест Linpack для систем с распределенной памятью) [2].

В случае программы умножения матрицы на вектор имеет смысл исследовать время работы программы в зависимости от размера данных (матрицы и вектора) и количества процессоров, на которых эта программа выполняется. (Размер данных и количество процессоров – универсальные параметры, присутствующие практически в любом параллельном приложении, поэтому они приняты за параметры по умолчанию и будут исследованы, если пользователь формально не описал никаких других параметров задачи.) Результаты обработки собранной статистики представлены на рис. 2 и 3 – трехмерные графики ускорения (рис. 2) и эффективности (рис. 3), полученные на машине Regatta (IBM eServer pSeries 690, regatta.cs.msu.su) [3].

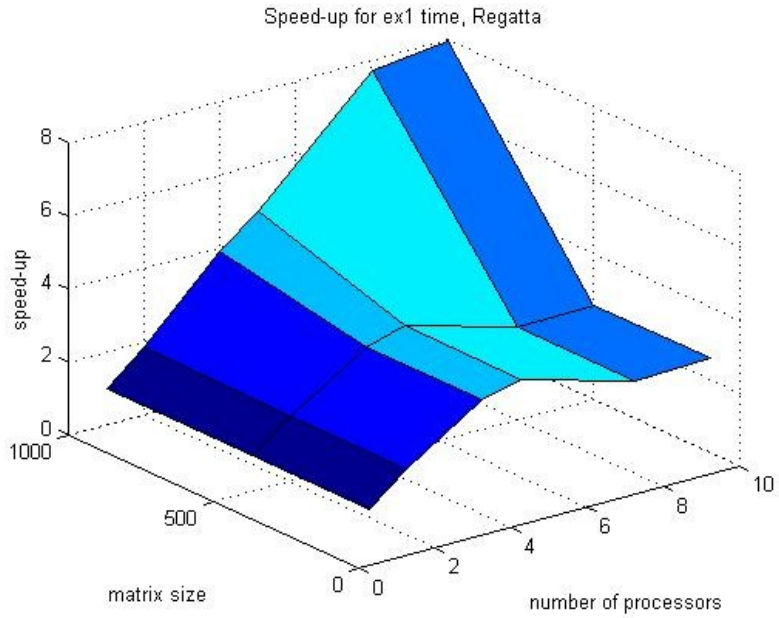

Рис. 2. Ускорение программы умножения матрицы на вектор (Regatta)

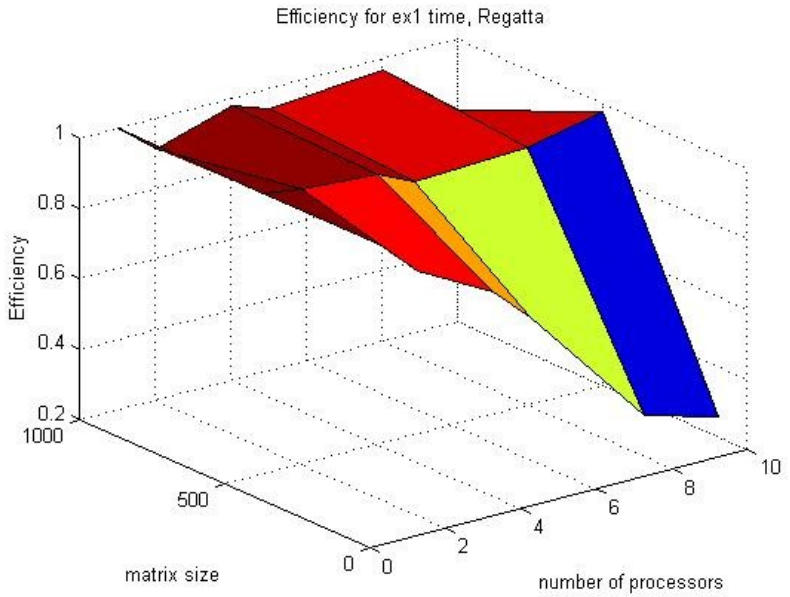

Рис. 3. Эффективность программы умножения матрицы на вектор (Regatta)

Чтобы проиллюстрировать применение представленной системы к «большим» вычислительным задачам, были собраны данные о выполнениях теста HPL на различных многопроцессорных системах с различными комбинациями параметров (описание самого теста и специфику его параметров можно найти в

[2]). Исследовались комбинации всех возможных входных параметров: размеры основной матрицы и подматриц, размерности решетки, алгоритм широковещательной рассылки полос матриц, и т.д. Вообще говоря, входной файла теста позволяет передавать некоторым параметрам сразу несколько значений; таким образом, в рамках одного запуска тест сам переберет все возможные их комбинации. Однако, для некоторых входных параметров это невозможно (например, норма невязки (threshold), формы хранения матриц разложения не могут меняться в рамках одного запуска); для перебора таких параметров и будет полезна предлагаемая система.

Здесь приведены результаты, полученные на 16-процессорной машине numa-архитектуры Regatta и суперкомпьютерах МВС-15000ВМ и МВС-100k Межведомственного Суперкомпьютерного центра. В докладе планируется дополнить приведенные здесь данные результатами, полученными на системах Blue Gene/P[6] и Скиф МГУ “Чебышев”[7]

Пример простейшего отчета представлен на рис. 4 (зависимость времени решения СЛАУ от размера матрицы на различных вычислителях, здесь представлены МВС-100k (на рисунке обозначен как MVS100k) и МВС-15000 (на рисунке - Blade1)) и рис. 5 (зависимость времени решения СЛАУ от размера матрицы и конфигурации процессорной решетки). Аналогичным образом можно исследовать поведение программы в зависимости от других характеристик и провести соответствующую настройку параметров на конкретном вычислителе.

К недостаткам системы можно отнести довольно строгие требования к формальному описанию параметров и выводу задачи; тем не менее, описание параметров выполняется полуавтоматически с помощью генераторов спецификации, а вывод задачи, как правило, можно привести к нужному виду довольно простым конвертером. Кроме того, предусмотрен режим сбора параметров по умолчанию – в этом случае исследуется время работы программы в зависимости от размера задачи и количества процессоров. Работа выполняется при поддержке гранта РФФИ 08-07-00445-а.

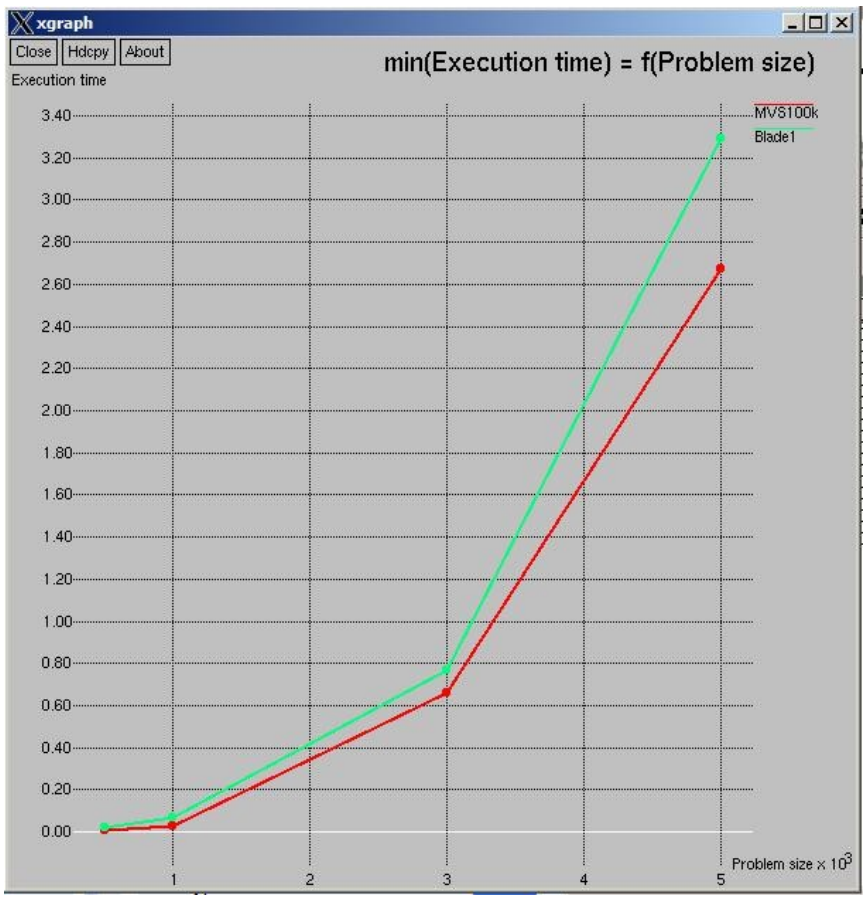

Рис. 4. Пример простого отчета - зависимость времени решения СЛАУ от размера матрицы на различных вычислителях (число процессоров фиксированно и равно 16)

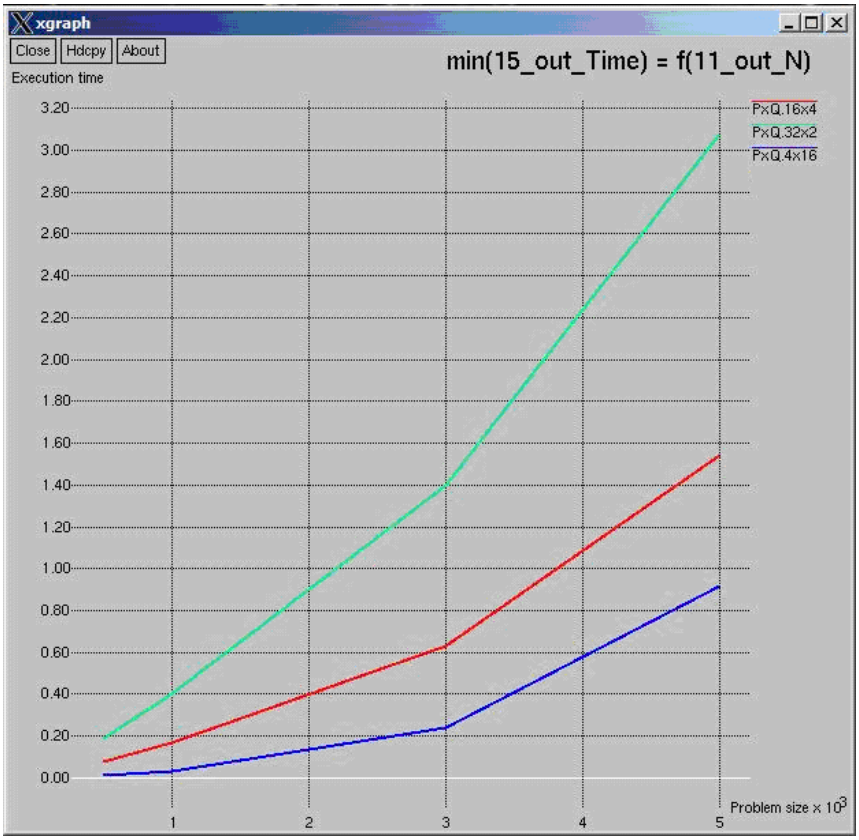

Рис. 5. Пример простого отчета - зависимость времени решения СЛАУ от размера матрицы и конфигурации процессорной решетки (количество процессоров фиксировано и равно 64, машина МВС-100К)

ЛИТЕРАТУРА:

1. N.A. Nystrom, J. Urbanic, C. Savinell. Understanding Productivity through non-intrusive instrumentation and statistical learning. //Proceedings of the 2nd workshop on productivity and performance in hign-end computing (PPHEC-05), 2005.
2. Petitet, R. C. Whaley, J. Dongarra, A. Cleary. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, 2004. [HTML] http://www.netlib.org/benchmark/hpl/
3. Moscow State University Regatta Community. [HTML] http://www.regatta.cs.msu.su/instr.htm
4. Межведомственный Суперкомпьютерный центр РАН. Суперкомпьютеры. [HTML] http://www.jscc.ru/cgi-bin/show.cgi?path=/hard/scomputer.html&type=1
5. Межведомственный Суперкомпьютерный центр РАН. Суперкомпьютер МВС-100k [HTML] http://www.jscc.ru/hard/mvs100k.shtml
6. BlueGene/P team. Overview of the IBM Blue Gene/P project // IBM Journal of Research & Development, Vol. 52 № 1-2 January/February 2008, p 199
7. Суперкомпьютер СКИФ МГУ [HTML] http://www.parallel.ru/cluster/skif_msu.html