Влияние характеристик программно-аппаратной среды на производительность приложений

А. С. Антонов*

Создаваемый в НИВЦ МГУ имени М.В.Ломоносова процессорный полигон позволяет исследовать влияние базовых характеристик программно-аппаратной среды на производительность пользовательских приложений. Такой анализ необходим для получения на реальных программах максимально возможной производительности.

1. Введение

Практика показывает, что для реальных приложений, написанных на языке высокого уровня, весьма трудно получить производительность процессорного ядра, превышающую 15-20% от пиковой. Причём это касается всех наиболее распространённых на данный момент серийных микропроцессоров. Применяя достаточно сложные ухищрения или используя низкоуровневую оптимизацию, иногда удаётся существенно приблизиться к пиковым характеристикам [1], но это обычно неприменимо для реальных пользовательских задач.

Для получения высокой эффективности пользовательского приложения нужно учитывать большое количество факторов, влияющих на производительность конкретного процессора. Эти факторы определяются как внутренней архитектурой вычислительного узла, так и влиянием всего комплекса программного обеспечения. При этом всё множество факторов действует одновременно, в результате

*Научно-исследовательский вычислительный центр МГУ

чего и получается столь существенное уменьшение производительности.

Исследовать влияние базовых характеристик программно-аппаратной среды на производительность приложений можно при наличии подходящей аппаратной базы, максимально полного комплекта системного программного обеспечения и разработанной системы тестов, позволяющей проводить комплексное исследование максимального количества факторов, отражающихся на производительности.

2. Аппаратная база и системное программное обеспечение

В НИВЦ МГУ имени М.В.Ломоносова создаётся уникальный процессорный полигон из вычислительных узлов на основе современных серийных микропроцессоров. К настоящему моменту он содержит 20 серверов. Это вычислительные сервера на базе:

- 5 серверов на базе процессоров AMD Opteron со следующими характеристиками:

Название узла	Тип процессора	Тактовая частота, ГГц	Число проц.	Число ядер на проц.	Объём ОП, ГБ	Пиковая производительность узла, GFlop/s
opteron1	Opteron 244	1.8	2	1	2	7.2
opteron2	Opteron 244	1.8	2	1	2	7.2
opteron3	Opteron 280	2.4	2	2	4	19.2
opteron4	Opteron 265	1.8	2	2	4	14.4
opteron5	Opteron 265	1.8	1	2	2	7.2

- 1 сервер на базе процессора IBM Power5 со следующими характеристиками:

Название узла	Тип процессора	Тактовая частота, ГГц	Число проц.	Число ядер на проц.	Объём ОП, ГБ	Пиковая производительность узла, GFlop/s
power1	POWER5	1.65	2	2	4	26.4

- 14 серверов на базе процессоров Intel со следующими характеристиками:

Название узла	Тип процессора	Тактовая частота, ГГц	Число проц.	Число ядер на проц.	Объём ОП, ГБ	Пиковая производительность узла, GFlop/s
dempsey1	Xeon 5050	3	1	2	4	12
xeon1	Xeon EM64T	3.2	2	1	4	12.8
pentiumd1	PentiumD 945	3.4	1	2	4	13.6
pentium41	Pentium 4 531	3.0	1	1	2	6
woodcrest1	Xeon 5150	2.66	1	2	16	21.28
woodcrest2	Xeon 5150	2.66	2	2	8	42.56
woodcrest3	Xeon 5150	2.66	1	2	4	21.28
woodcrest4	Xeon 5150	2.66	1	2	4	21.28
woodcrest5	Xeon 5160	3	2	2	10	48
woodcrest6	Xeon 5130	2	2	2	8	32
woodcrest7	Xeon 5130	2	2	2	8	32
clovertown1	Xeon 5310	1.6	2	4	16	51.2
clovertown2	Xeon 5345	2.33	1	4	6	37.28
clovertown3	Xeon 5355	2.66	1	4	6	42.56

Суммарная пиковая производительность серверов процессорного полигона составляет 475.44 GFlop/s, суммарный объём оперативной памяти 118 Гбайт. Сервера в процессорный полигон подбираются таким образом, чтобы отследить основные тенденции развития компьютерного рынка и предоставить возможность исследования максимального количества факторов, влияющих на производительность современных серийных микропроцессоров. Все сервера установлены в стойку, имеют форм-фактор 1–2U и доступны посредством системы очередей Cleo [3].

На узлах процессорного полигона установлена ОС Linux, дистрибутив SUSE 10.1, на узле power1 — AIX 5.3.

На всех узлах процессорного полигона (кроме узла power1) установлены следующие компиляторы:

- GNU (C, C++, Fortran 77/90/95) 4.1.2 (4.0.2);
- Intel Compilers (C, C++, Fortran 77/90/95) 9.1;
- Portland Group Inc. Compilers (C, C++, Fortran 77/90/95) 6.2-5;
- PathScale EKOPath Compiler Suite (C, C++, Fortran 90/95) 2.5;
- Absoft Fortran (Fortran77/90/95) 9.0.

Для корректного сравнения различных компиляторов, если не указано дополнительно, использовалась только опция стандартной оптимизации -O3. Использованию более специфичных для каждого компилятора опций посвящено отдельное исследование.

Для задач, требующих параллельного исполнения, использовалась библиотека Intel MPI 3.0. Для компиляции теста HPL были установлены альтернативные библиотеки ATLAS 3.7.30, MKL 5.2 и GotoBLAS 1.11.

3. Комплексное тестирование вычислительных серверов

Для решения задач комплексного тестирования вычислительных серверов процессорного полигона предназначена система тестов, включающих как широко известные тесты, так и тесты, разработанные в НИВЦ МГУ имени М.В.Ломоносова.

HPL (High Performance Linpack) — стандартный тест, реализующий решение больших систем линейных алгебраических уравнений методом LU-разложения. Вычисления производятся с помощью вызовов процедур BLAS. Тест HPL традиционно используется для упорядочивания списка наиболее мощных компьютеров мира Top500 (http://www.top500.org/) и списка наиболее мощных компьютеров СНГ Top50 (http://supercomputers.ru).

Peak — небольшой тест, адаптированный из [4], предназначенный для достижения максимально возможной производительности процессорного ядра на фрагменте программы, написанном на языке высокого уровня (Фортран). Для этого используются пары операций сложение+умножение, а все данные потенциально могут быть расположены на регистрах.

Operations — тест, определяющий производительность процессорного ядра на ряде простых арифметических операций и их комбинаций.

STREAM — стандартный тест, измеряющий производительность на операциях $a(i) = b(i)$; $a(i) = q*b(i)$; $a(i) = b(i)+c(i)$; $a(i) = b(i) + q * c(i)$ и производящий замер скорости передачи данных из

оперативной памяти в процессор.

Flo52 — программа из пакета тестов Perfect Club Benchmarks, реализующая упрощённый вариант одной из задач вычислительной гидродинамики [5].

Применяется также ряд других тестов, и этот набор постоянно пополняется с расширением множества анализируемых факторов, влияющих на производительность приложений.

Формируемый в результате комплексного тестирования производительности вычислительных серверов процессорного полигона документ Performance Guide [6] содержит информацию о сравнительных характеристиках серверов процессорного полигона, позволяет оценить их пиковые характеристики, найти узкие места при выполнении тех или иных операций, определить эффективность базового программного обеспечения (операционных систем, компиляторов, оптимизированных низкоуровневых библиотек и т.д.), а также решать многие другие задачи.

4. Результаты тестирования

Рассмотрим некоторые результаты, полученные в ходе тестирования серверов процессорного полигона. Целью тестирования являлось определение влияния различных характеристик программно-аппаратной среды на производительность приложений.

Сначала посмотрим, какой производительности можно достичь в принципе на процессорных ядрах серверов полигона на программе, написанной на языке высокого уровня. Рис. 1 демонстрирует производительность теста peak в сравнении с пиковой производительностью процессорных ядер. Для компиляции на всех узлах был использован компилятор gfortran с опцией оптимизации -O3.

На графике видно, что тест peak позволяет значительно лучше приблизиться к пиковой производительности на процессорах AMD, чем на процессорах Intel или IBM. На процессорах Intel можно добиться большего при задействовании команд из наборов SSE2 и SSE3, но эта операция требует дальнейшего преобразования текста используемого теста.

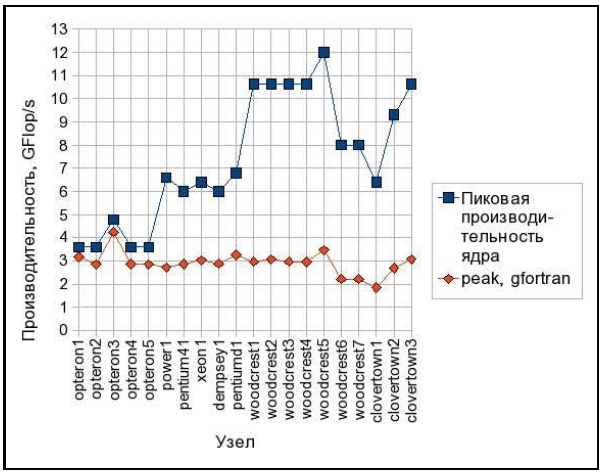

Рис. 1. Производительность теста peak на узлах полигона

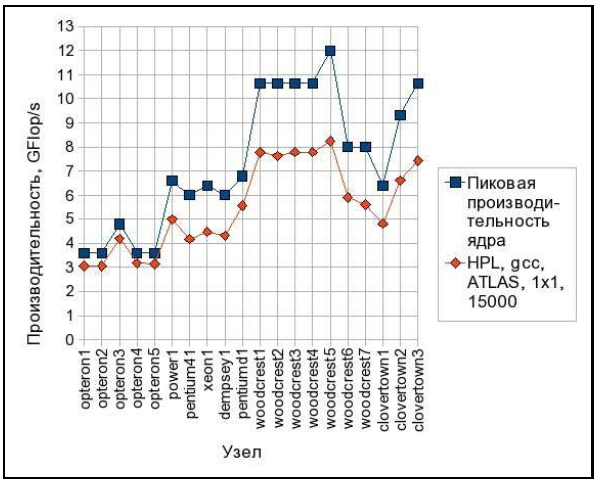

Рис. 2. Производительность теста HPL на узлах полигона

Программа HPL традиционно используется в высокопроизводительных вычислениях как некоторая мера производительности вычислительного узла. Она позволяет оценить возможность достижения максимальной производительности вычислительного узла с использованием низкоуровневых библиотек. Результаты теста HPL приведены на рис. 2. Для компиляции на всех узлах был использован компилятор gcc с опцией оптимизации -O3. В качестве реализации BLAS была взята библиотека ATLAS. Тест запускался на одном ядре с матрицей 15000×15000.

График показывает, что с использованием оптимизированной под конкретную архитектуру низкоуровневой библиотеки можно значительно ближе подойти к пиковой производительности процессорного узла, особенно это касается процессоров Intel и IBM.

Многие стандартные алгоритмы многократно реализованы на доступных платформах. Зачастую бывает гораздо проще использовать существующую реализацию, а не писать свой вариант. Но насколько эффективны те или иные реализации стандартных библиотек на конкретной платформе, нужно показать при помощи набора тестов. Посмотрим, как влияет выбор конкретной специализированной библиотеки на производительность приложения. Для теста HPL сравним производительность при использовании библиотек ATLAS, MKL и GotoBLAS (рис. 3). Для компиляции на всех узлах был использован компилятор gcc с опцией оптимизации -O3. Тест запускался на одном ядре узла opteron1 с матрицей 15000×15000.

Наиболее эффективной библиотекой в данном случае является GotoBLAS. Её использование даёт наилучшие результаты и на процессорах Intel, но на них библиотека MKL эффективнее, чем библиотека ATLAS.

Теперь посмотрим, какова производительность процессорных ядер полигона на вычислительном ядре реального алгоритма. Для этого возьмём большой (large) вариант программы flo52 (рис. 4).

Очевидно, что производительность процессоров на реальном приложении значительно ниже, чем на специально разработанном тесте или на тесте, использующем оптимизированные библиотеки.

Оценим теперь, какой вклад в производительность процессорных

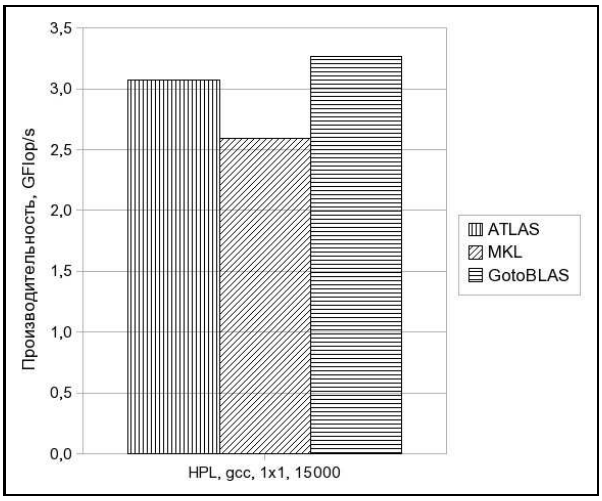

Рис. 3. HPL на узле opteron1 в зависимости от реализации BLAS

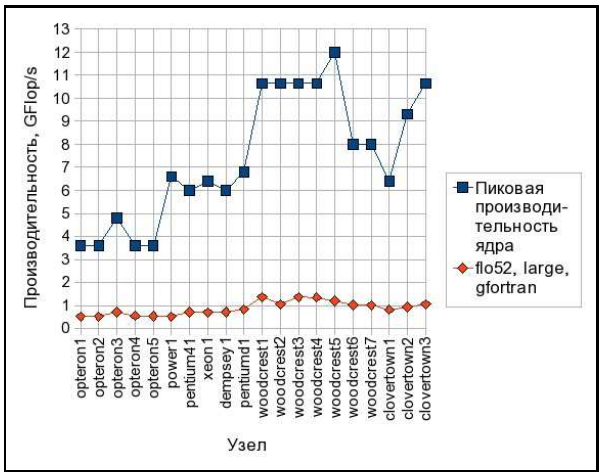

Рис. 4. Производительность теста flo52 на узлах полигона

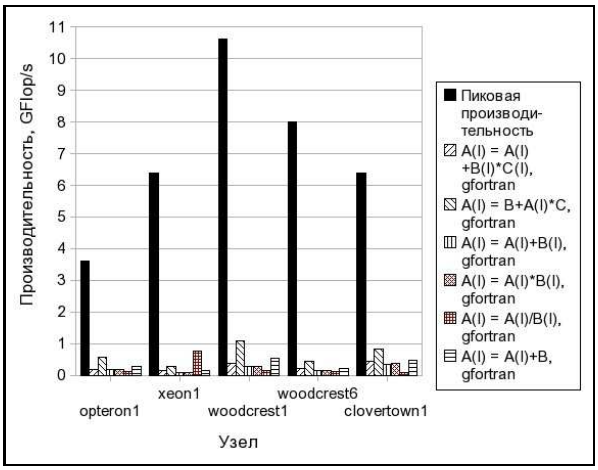

Рис. 5. Производительность арифметических операций на узлах полигона

ядер на программах, написанных на языках высокого уровня, вносят различные базовые арифметические операции. Посмотрим часть результатов теста operations на нескольких серверах процессорного полигона (рис. 5). Для компиляции на всех узлах был использован компилятор gfortran с опцией оптимизации -O3.

Результаты теста показывают, что даже на, казалось бы, «хороших» с точки зрения архитектуры процессоров базовых операциях получается производительность, составляющая лишь небольшую долю от пиковой производительности.

Теперь хотелось бы проверить, насколько велико влияние компилятора на эффективность использования аппаратных средств. Посмотрим, как зависит производительность большого варианта теста flo52 от использования различных компиляторов (рис. 6). Для компиляции на всех узлах были использованы компиляторы с опцией оптимизации -O3.

Видно, что для flo52 лучшим вариантом на всех узлах является использование компилятора PathScale. Это касается как процессо-

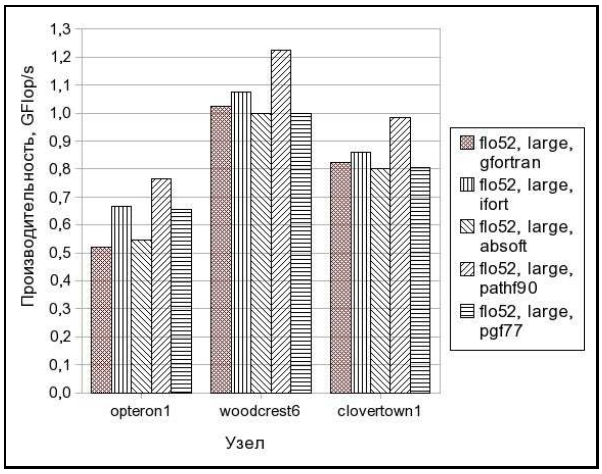

Рис. 6. flo52 на узлах полигона в зависимости от компилятора

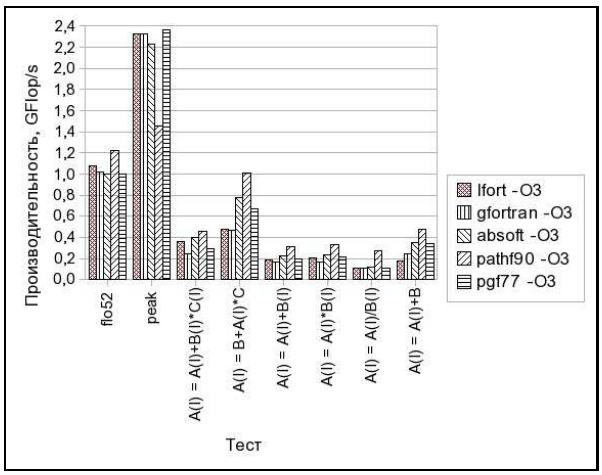

Рис. 7. Разные тесты на узле woodcrest6 в зависимости от используемого компилятора

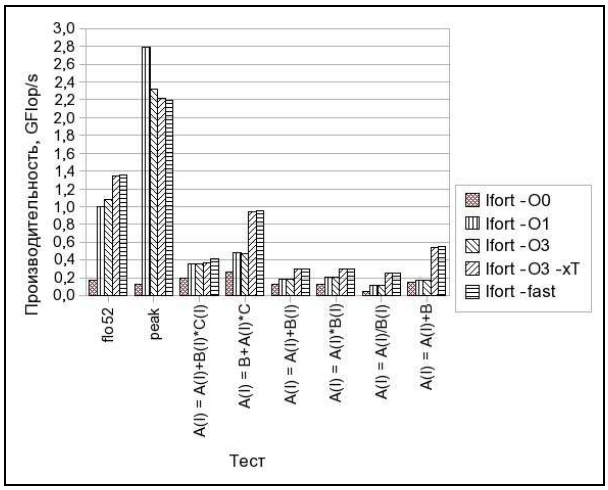

Рис. 8. Разные тесты на узле woodcrest6 в зависимости от используемых опций компилятора ifort

ров AMD, так и процессоров Intel.

На рис. 7 приведено сравнение эффективности различных компиляторов на наборе из нескольких тестов. Для компиляции на узле woodcrest6 были использованы компиляторы с опцией оптимизации -O3.

На всех тестах кроме peak снова лучшим является компилятор PathScale. С тестом peak чуть лучше других справляется компилятор Portland Group.

Каждый компилятор является сложной программой. Для его эффективного использования на каждой конкретной платформе нужно знать много тонких вещей. Все предыдущие эксперименты проводились со стандартным уровнем оптимизации -O3. Посмотрим, чего можно добиться при использовании других специфических опций оптимизации. На рис. 8 показана зависимость производительности набора тестов от используемых опций компилятора ifort. Результаты приводятся для узла woodcrest6.

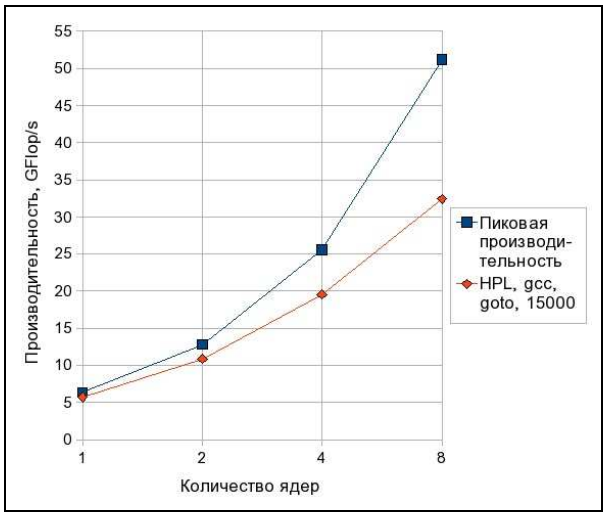

Рис. 9. HPL на узле clovertown1 в зависимости от количества ядер

Видно, что для компилятора Intel Fortran на всех тестах кроме peak лучшие результаты получаются с опцией -fast, которая является сокращением для набора опций -O3 -ipo -no-prec-div -static -xP. На тесте peak лишние оптимизации только ухудшают ситуацию, и оптимальным остаётся вариант с уровнем оптимизации -O1. Дальнейшие исследования специфических опций каждого компилятора для каждой конкретной архитектуры могут дать дополнительный эффект, но вряд ли очень существенно изменят общую картину.

До сих пор речь шла о производительности одного процессорного ядра. В настоящее время почти все процессоры делаются многоядерными, и эта тенденция находит отражение в составе процессорного полигона. На многоядерных и многопроцессорных узлах можно проверить масштабируемость параллельных программ, то есть изменение производительности тестовых приложений при увеличении количества процессов. График на рис. 9 демонстрирует масштабируемость теста HPL при увеличении количества задействованных ядер узла clovertown1. Для компиляции на всех узлах был использован

компилятор gcc с опцией оптимизации -O3. В качестве реализации BLAS была взята библиотека GotoBLAS. Тест запускался с матрицей 15000×15000.

При увеличении количества задействованных процессорных ядер большее влияние начинают оказывать каналы межъядерного и межпроцессорного взаимодействия. Поэтому производительность теста всё более удаляется от пиковой. Здесь показана масштабируемость только внутри одного сервера, при задействовании нескольких серверов всё большее влияние начнёт оказывать и используемая коммуникационная сеть.

5. Заключение

Часть факторов, влияющих на производительность приложений, осталась за рамками данной статьи. Это, например, оценка скорости работы с различными уровнями памяти, оценка влияния на производительность объёма оперативной памяти, частоты системной шины, эффективность операционной системы и другие. Для каждой такой задачи требуются свои подходы и формируются свои наборы тестов. Часто влияние одного фактора очень трудно отделить от влияния другого, а в процессе работы программы сказываются почти все факторы. Важно, что проанализировав результаты комплексного тестирования пользователь может оценить важность для его задачи тех или иных факторов и попытаться найти лучшее соответствие реализуемого алгоритма и целевой программно-аппаратной среды.

Список литературы

[1] Антонов А.С. Далеко ли до пика?// Открытые системы, N 6, 2006. С. 64–66.

[2] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2002.

[3] Жуматий С.А. Система анализа производительности параллельных программ на кластерных установках// Вычислительные методы и программирование. 2005. Т 6, N 2. С. 57–64.

[4] AIX Version 3.2 for RISC System/6000. Optimization and Tuning Guide for Fortran, C, and C++.

[5] Воеводин Вл.В., Антонов А.С. Эффективная адаптация последовательных программ для современных векторно-конвейерных и массивно-параллельных супер-ЭВМ// Программирование, 1996, N 4, С.37–51.

[6] Антонов А.С. Комплексное тестирование производительности вычислительных серверов// Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийской научной конференции (24-29 сентября 2007 г., г.Новороссийск).-М.: Изд-во МГУ, 2007. С. 135–138.