Далеко ли до пика?

Александр Антонов, 31.08.2006.
Журнал ‘Открытые системы’, #06/2006

Эффективное использование современных микропроцессоров — сложная задача. Заявляемые производителями пиковые характеристики процессоров практически недостижимы без низкоуровневого программирования.

Для написания на языке высокого уровня теста, скорость выполнения которого приближается к пиковой производительности, требуется изучить особенности конкретного процессора и тонкие настройки используемых компиляторов. Большинство же вычислительных задач выполняются на массовых микропроцессорах с эффективностью 10-20%.

Современные 64-разрядные процессоры AMD Opteron, Intel Xeon EM64T и IBM Power5, безусловно, имеют свои особенности, но их использование приводит к схожим результатам. С точки зрения высокопроизводительных вычислений извлекать максимум из доступной программно-аппаратной платформы крайне сложно — даже при том, что за рамками рассмотрения остаются вопросы применения многопроцессорности и многоядерности. Прикладного программиста, в конечном счете, интересует только одно: его программа должна выполняться точно и быстро. При этом он зачастую слабо осведомлен о характеристиках конкретного процессора и тонких настройках программных средств. Рассмотрим скорость выполнения вычислительных алгоритмов на разных программно-аппаратных платформах при однопроцессорной обработке.

Время выполнения программы — характеристика важная. Однако она не позволяет понять, насколько далек программист от оптимального использования предоставленных ему ресурсов, поэтому имеет смысл говорить о производительности компьютера. При фиксированном объеме вычислений производительность в некотором фрагменте однозначно соответствует времени его выполнения. Согласно [1], реальной производительностью вычислительной системы является количество операций, выполненных за единицу времени, а пиковой производительностью — максимальное число операций, которое может быть выполнено за то же время при отсутствии связей между функциональными устройствами. Таким образом, пиковая производительность характеризует потенциальные возможности аппаратуры и не зависит от выполняемой программы.

Для вычислительных задач пиковую производительность принято определять как количество вещественных операций, выполняемых в единицу времени (FLOPS). Важную роль играет также формат используемых данных. Все результаты экспериментов, которые мы приведем, относятся к 64-разрядным процессорам. Их производительность характеризуется количеством операций с вещественными числами двойной точности.

Ясно, что применительно к производительности нельзя говорить только об аппаратуре: на быстродействии сказывается эффективность операционной системы и программного обеспечения. Если компилятор будет выдавать плохой код, аппаратура не сможет быстро его выполнять. А значит, нужно вести речь о тестировании программно-аппаратной среды, компоненты которой трудно, а подчас и невозможно разделить.

Равнение на Linpack

Существует огромное количество тестов, в той или иной мере характеризующих производительность вычислительных систем, но общепризнанным стал тест Linpack, реализующий достаточно простой алгоритм решения системы линейных алгебраических урав-

нений с плотной матрицей с выбором главного элемента по строке. Для высокопроизводительных систем самой распространенной реализацией этого теста является HPL (High Performance Linpack), которая используется для формирования рейтинга Top500.

Однако эффективность HPL сильно зависит от реализации библиотеки BLAS (Basic Linear Algebra Solver). Наиболее известные реализации BLAS для современных микропроцессоров содержатся в библиотеках ATLAS, MKL и GotoBLAS. Посмотрим, насколько удается приблизиться к пиковой производительности тестируемых процессоров при использовании HPL с этими библиотеками (табл. 1). Во всех таблицах в скобках указан процент пиковой производительности процессора. В экспериментах с HPL задавалась матрица 15000×15000 элементов, близкая к максимальной матрице, которая целиком помещается в 2 Гбайт оперативной памяти.

	ПИК	ATLAS	MKL	GOTOBLAS
AMD Opteron/1,8 ГГц	3600	3068 (85,2%)	2594 (72,1%)	3263 (90,6%)
AMD Opteron/2,4 ГГц	4800	4197 (87,4%)	2931 (61,1%)	4352 (90,7%)
Intel Xeon EM64T/3,2 ГГц	6400	4537 (70,9%)	5390 (84,2%)	5479 (85,6%)
IBM Power5/1,65 ГГц	6600	4993 (75,7%)	н/д	н/д

Таблица 1. Производительность в тестах Linpack (MFLOPS)

Казалось бы, ситуация замечательная, ведь в достаточно типичной задаче удается получать до 90% пиковой производительности, но все не так просто. Самый существенный вклад в высокую производительность процессоров в данном случае вносит использование хорошо оптимизированных библиотек, реализующих функции BLAS, значительная часть которых написана на ассемблере. На что же можно рассчитывать при программировании на языке высокого уровня без применения оптимизированных библиотек и ассемблерных вставок?

В поисках максимума

При написании программ на языке высокого уровня большое значение имеет эффективность компилятора. В наших экспериментах были доступны компиляторы gcc/gfortran 3.3.*/4.0.* и Intel Compiler 9.0. Использовались только стандартные уровни оптимизации +O/O2/O3, хотя в каждом случае более тонкая настройка может дать дополнительный эффект. Напишем на языке Фортран простой фрагмент, в котором предположительно должна быть получена хорошая производительность:

```
DO I = 1, N
A(I) = B + A(I) * C
END DO
```

Известно, что максимальная производительность современных процессоров достигается в парах 'сложение + умножение’. Для того чтобы обеспечить значительное количество независимых операций, зададим большую длину векторов N (в данном эксперименте — 131072 элемента), а для минимизации затрат на обращения к памяти оставим лишь

один массив A. Посмотрим теперь, какая производительность получается на различных процессорах при использовании разных компиляторов (табл. 2).

	ПИК	GNU	INTEL
AMD Opteron/1,8 ГГц	3600	567,20 (15,76%)	562,35 (15,62%)
AMD Opteron/2,4 ГГц	4800	750,72 (15,64%)	727,33 (15,15%)
Intel Xeon EM64T/3,2 ГГц	6400	778,36 (12,16%)	908,69 (14,20%)
IBM Power5/1,65 ГГц	6600	960,36 (14,55%)	

Таблица 2. Производительность выполнения исходного фрагмента (MFLOPS)

Даже при выполнении достаточно ‘хорошей’ операции получаем лишь 15% пиковой производительности. Попытаемся добиться большего, заменив операцию над массивом скалярными операциями, операнды которых предположительно можно расположить на регистрах. Следующий фрагмент (рис.1а) взят из [2]. В нем опять используются пары ‘сложение + умножение’, а все данные потенциально могут быть расположены на регистрах (в табл. 3 указана производительность тестируемых процессоров в данном фрагменте).

```
FP1 = 0.00000100
FP2 = 0.00000001
FP3 = 0.00000002
FP4 = 0.00000003
FP5 = 0.00000004
FP6 = 0.00000001
FP7 = 0.00000011
FP8 = 0.00000017
FP9 = 0.0000041
DO I = 1, 1000000
FP8 = FP8 * FP9 + FP2
FP7 = FP7 * FP9 — FP1
FP6 = FP6 * FP9 + FP8
FP5 = FP5 * FP9 — FP7
FP4 = FP4 * FP9 + FP6
FP3 = FP3 * FP9 — FP5
FP2 = FP2 * FP9 + FP4
FP1 = FP1 * FP9 — FP3
END DO
```

а)

```
DO I = 1, 1000000
FP8 = FP8 * FP9 + FP2
GP8 = GP8 * GP9 + GP2
FP7 = FP7 * FP9 — FP1
GP7 = GP7 * GP9 — GP1
FP6 = FP6 * FP9 + FP8
GP6 = GP6 * GP9 + GP8
FP5 = FP5 * FP9 — FP7
GP5 = GP5 * GP9 — GP7
FP4 = FP4 * FP9 + FP6
GP4 = GP4 * GP9 + GP6
FP3 = FP3 * FP9 — FP5
GP3 = GP3 * GP9 — GP5
FP2 = FP2 * FP9 + FP4
GP2 = GP2 * GP9 + GP4
FP1 = FP1 * FP9 — FP3
GP1 = GP1 * GP9 — GP3
END DO
```

б)

Рис. 1. Модификации с целью оптимизации

Все множество операций оптимизированного фрагмента распадается на два независимых подмножества, в каждом из которых операции должны выполняться последовательно. Таким образом, при идеальном исполнении можно одновременно выполнять две пары операций ‘сложение + умножение’. Если процессор потенциально способен выполнять две операции с плавающей точкой за такт (как это и заявлено для процессоров AMD Opteron и Intel Xeon EM64T), то данный фрагмент удастся выполнять с производительностью, близкой к пиковой. Однако результаты эксперимента показывают, что в действительности используется лишь половина этого потенциала.

	ПИК	GNU	INTEL
AMD Opteron/1,8 ГГц	3600	1787,33 (49,65%)	1786,28 (49,62%)
AMD Opteron/2,4 ГГц	4800	2387,28 (49,74%)	2387,03 (49,73%)
Intel Xeon EM64T/3,2 ГГц	6400	1692,19 (26,44%)	2311,55 (36,12%)
IBM Power5/1,65 ГГц	6600	1100,11 (16,67%)	

Таблица 3. Производительность выполнения оптимизированного фрагмента (MFLOPS)

Попробуем увеличить потенциал одновременно выполняемых операций. Мы модифицируем предыдущий фрагмент, продублировав пары ‘сложение + умножение’ с другими скалярными переменными (рис. 1б). Количество операций фрагмента увеличилось в два раза; одновременно изменилась и производительность (табл. 4).

Применительно к процессорам AMD Opteron удалось подойти весьма близко к пиковой производительности. Intel Xeon EM64T может приблизиться к пиковой производительности при использовании команд из наборов SSE2 и SSE3, позволяющих выполнять операции над 128-битными данными, что при обработке данных двойной точности эквивалентно работе с двухкомпонентными векторами. В нашем случае можно вместо переменных FP* и GP* ввести массивы из двух элементов FP*(2), после чего задействовать инструкции векторизации по дополнительному измерению. Такое преобразование позволило получить на Xeon EM64T 3,2 ГГц производительность 4625,90 MFLOPS, что составляет 64,25% пиковой производительности.

	ПИК	GNU	INTEL
AMD Opteron/1,8 ГГц	3600	3178,71 (88,30%)	2820,59 (78,35%)
AMD Opteron/2,4 ГГц	4800	4245,78 (88,45%)	3758,55 (78,30%)
Intel Xeon EM64T/3,2 ГГц	6400	1102,16 (17,22%)	3015,99 (47,12%)
IBM Power5/1,65 ГГц	6600	2202,06 (33,36%)	

Таблица 4. Производительность модифицированного фрагмента (MFLOPS)

Процессор IBM Power5 теоретически способен выполнять за такт четыре операции с плавающей точкой. Попробуем, аналогично предыдущему преобразованию, увеличить количество операций фрагмента еще вдвое (табл.5). Производительность процессора Power5 продолжает расти, в то время как производительность Opteron и Xeon EM64T падает. К сожалению, в наших экспериментах для Power5 был доступен только компилятор gcc/gfortran.

	ПИК	GNU	INTEL
AMD Opteron/1,8 ГГц	3600	1392,2 (38,69%)	1764,00 (49,00%)
AMD Opteron/2,4 ГГц	4800	1991,17 (41,48%)	2352,24 (49,01%)
Intel Xeon EM64T/3,2 ГГц	6400	1793,63 (28,03%)	2676,62 (41,82%)
IBM Power5/1,65 ГГц	6600	2733,64 (41,42%)	

Таблица 5.

Таким образом, наличие в программе множества независимых операций, казалось бы идеально соответствующих архитектуре микропроцессора, не гарантирует достижения пиковой производительности. Только использование искусственных приемов позволяет — да и то лишь в некоторых случаях — приблизиться к теоретическому пику.

Производительность в реальных приложениях

Теперь посмотрим, как соотносится производительность, полученная в тестовых примерах, с производительностью реальных приложений. Для примера возьмем программу flo52 [3], представляющую собой упрощенный вариант реализации одной из задач вычислительной гидродинамики. Эта программа достаточно хороша для получения высокого быстродействия, поскольку работает с регулярными структурами данных, а ее основное вычислительное ядро содержит множество независимых пар операций ‘сложение + умножение’. В табл. 6 приведены результаты выполнения ‘большого’ варианта flo52.

	ПИК	GNU	INTEL
AMD Opteron/1,8 ГГц	3600	520,73 (14,46%)	666,56 (18,52%)
AMD Opteron/2,4 ГГц	4800	715,38 (14,90%)	907,61 (18,91%)
Intel Xeon EM64T/3,2 ГГц	6400	650,90 (10,17%)	802,10 (12,53%)
IBM Power5/1,65 ГГц	6600	516,78 (7,83%)	

Таблица 6. Производительность программы flo52 (MFLOPS)

Таким образом, при работе с реальной программой получается производительность не более 10-20% пиковой. Как показывает наш опыт вычислительных экспериментов, это неплохой результат.

Производители микропроцессоров уже осознали, что дальнейшее усложнение внутренней архитектуры не приносит ожидаемого выигрыша в производительности, а дальнейшее наращивание тактовой частоты если и возможно технологически, то неоправданно экономически. В результате приоритетным направлением становится создание многоядерных процессоров. При этом требуется реальное распараллеливание, а сложность задачи получения высокой производительности сдвигается на более высокий уровень приложений или программного обеспечения промежуточного слоя.

Не очень искушенный пользователь, реализуя вычислительный алгоритм, может получить буквально несколько процентов от ‘пика’. Зачастую можно доказать, что в рамках данных программно-аппаратных платформ принципиально больше получить нельзя. А если такова ситуация с производительностью одного процессора, то какую долю пиковой производительности удастся получить в сложных многопроцессорных/многоядерных вычислительных системах?

Литература

1. В.В.Воеводин, Вл.В.Воеводин. Параллельные вычисления.— СПб.: БХВ-Петербург, 2002.
2. AIX Version 3.2 for RISC System/6000. Optimization and Tuning Guide for Fortran, C and C++.
3. Вл. В.Воеводин, А.С.Антонов. Эффективная адаптация последовательных программ для современных векторно-конвейерных и массивно-параллельных супер-ЭВМ // Программирование. 1996, № 4,.

*Александр Антонов (*asa@parallel.ru*) — старший научный сотрудник НИВЦ МГУ (Москва).* *Работа выполнена при поддержке гранта РФФИ-БРФФИ 04-01-81022.*