ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ НА ЗАДАЧЕ АЭРОДИНАМИКИ МЕТОДОМ ПРОФИЛИРОВАНИЯ С ПОМОЩЬЮ СЧЕТЧИКОВ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА

К.С. Солнушкин

Производительность ЭВМ всегда заботила исследователей. Для целого ряда больших задач сама возможность их решения определяется наличием высокопроизводительных ЭВМ. Исследования НИВЦ МГУ убедительно доказали, что ключом к высокой производительности является согласование структуры алгоритмов и архитектуры ЭВМ [1]. В первую очередь это означает поиск численного метода, его алгоритма и реализации в виде программы, которые бы учитывали особенности конкретной вычислительной системы.

Однако существует большой объем наработанного программного обеспечения, в котором численные методы и алгоритмы уже фиксированы. Речь идет о CAE-системах – инженерном программном обеспечении (ПО), предназначенном для численного моделирования широкого спектра задач, возникающих в естественных и прикладных науках. Ключевой особенностью передовых CAE-систем является их верифицированность – иными словами, является общепризнанным, что лежащие в основе ПО численные методы и алгоритмы на широком диапазоне входных данных выдают корректный результат. Это свойство настолько важно, что многие прикладные специалисты не занимаются разработкой собственного программного обеспечения, а используют существующие CAE-системы.

При этом проблема согласования структуры алгоритмов и архитектуры ЭВМ ни в коем случае не отменяется. Просто к ее решению подходят с другой стороны: если алгоритм и программу изменить нельзя, то придется подбирать подходящую ЭВМ. Зачастую пользователи CAE-систем формулируют вопрос именно так: "Какая ЭВМ будет для применяемого нами ПО максимально эффективна?" Таким образом, приходится решать дуальную задачу – поиск ЭВМ, архитектура которой согласуется с применяемым ПО.

Решение задачи согласования при этом заключается в оценке характеристик ресурсоемкости вычислительного процесса, их сравнении с параметрами ЭВМ (доступными ресурсами) и поиске таких значений параметров, которые бы обеспечили требуемое согласование. Если исходный текст ПО недоступен, а именно этот случай характерен для коммерческих CAE-систем, одним из вариантов получения информации о характеристиках вычислительного процесса является низкоуровневая профилирование – проведение измерений с использованием счетчиков, встроенных в центральный процессор (ЦП). Счетчики способны регистрировать такие события, как завершение машинной инструкции, выполнение операции с плавающей точкой, промах кэш-памяти и т.п.

Анализируя результаты измерений, можно делать содержательные выводы о степени согласованности алгоритма и архитектуры ЭВМ и о возможных направлениях оптимизации ЭВМ либо алгоритма. В случае, если исходные тексты ПО доступны, значение профилирования еще более возрастает, так как результаты измерений можно сопоставить отдельным функциям и операторам программы на языке высокого уровня, что открывает широкие возможности для оптимизации [2].

В процессе исследования производительности ЭВМ методом профилировки анализируют характеристики существующих алгоритмов и программ. В частности, для инженерных приложений характерно значительное количество операций с плавающей точкой в общем объеме машинных инструкций. Анализ позволяет выявить потребности программ в ресурсах ЭВМ и обоснованно ответить, какие изменения в архитектуре ЭВМ способны привести к росту производительности. Среди таких изменений наиболее предпочтительны те, которые ведут к уменьшению значения критерия "стоимость ЭВМ/производительность на данной задаче". Иными словами, прирост стоимости ЭВМ допустим, но он должен сопровождаться опережающим приростом производительности.

Анализ показаний счетчиков, встроенных в центральные процессоры ЭВМ, требует от исследователя определенной аккуратности. Пусть ЭВМ требуется записать значение регистра ЦП в оперативную память. Сначала ЭВМ попытается сохранить это значение в кэш-памяти. Если в кэш-памяти отсутствует строка, соответствующая нужному участку основной памяти ("промах при записи", "write miss"), и используется политика "размещения при записи" ("write allocate"), то ЭВМ произведет чтение из памяти в объеме строки кэша, а затем уже сохранит значение регистра в кэше. Таким образом, выполнение инструкции записи может привести к увеличению значения счетчика, отвечающего за операции чтения. Подобные детали необходимо учитывать.

В данной статье рассмотрено профилирование CAE-системы "FLUENT" [3] (версия 6.3.26) на типовой задаче аэродинамики с применением статистического профилировщика "OProfile" [4] (версии 0.9.3) и ядра ОС Linux (версии 2.6) на ЭВМ с центральным процессором "AMD Athlon 64 X2 4400+" (тактовая частота 2,2 ГГц). В качестве задачи исследовано численное моделирование обтекания кузова автомобиля методом конечных элементов с учетом турбулентности [5]. Количество элементов сетки составляет 3,6 млн.

Первым этапом является исследование самого профилировщика "OProfile" на тестовой задаче. Это своего рода "поверка инструмента". В качестве тестовой задачи выбран известный тест быстродействия оперативной памяти "STREAM" [6]. Суть его работы заключается в следующем. Три больших одномерных массива данных типа "float" (вещественное число с плавающей запятой, как правило, длиной 8 байт) инициализируются начальными значениями. Затем в 4-х вычислительных ядрах происходит поэлементная (в цикле) обработка массивов. Ядра различаются количеством операций с ОЗУ и арифметико-логическим устройством центрального процессора. Каждое ядро соответствует какой-либо распространенной вычислительной конструкции, так, ядро "Triad" поэлементно выполняет операцию "a[j]=b[j]+scalar·c[j]". Для выполнения этой операции компилятор генерирует 5 машинных инструкций сопроцессора (компиляция с параметрами "gcc –g3 –m32 stream.c"). Из них 2 инструкции загрузки (fld), 1 инструкция умножения (fmul), 1 инструкция сложения (faddp) и 1 инструкция сохранения (fstp). Суммарно все вычислительных ядра выполняют 6 инструкций загрузки, 4 инструкции сохранения и по 2 инструкции умножения и сложения (соотношение 6:4:2:2).

Для профилировки был использован подсчет числа событий "RETIRED_INSTRUCTIONS" и "DISPATCHED_FPU_OPS" (описание событий см. в [7; 4]). Измерения показали, что при размере массива 20 млн. элементов (2·107) и повторении каждого ядра t=102 раз ЭВМ выполнила порядка 7,32·1010 машинных инструкций, из них порядка 2,84·1010 инструкций математического сопроцессора, с разбивкой по типам: 1,21·1010 инструкций загрузки, 0,82·1010 инструкций сохранения, 0,41·1010 инструкций сложения и 0,40·1010 инструкций умножения. Как видим, соотношение, полученное в результате измерения, составляет примерно 6:4:2:2, что совпадает с известным a priori соотношением. Отклонения вызваны двумя факторами. Во-первых, так как "OProfile" является статистическим профилировщиком, собранные с его помощью данные подвержены статистическим отклонениям (для уменьшения этого эффекта можно воспользоваться многократной профилировкой программы на одних и тех же входных данных). Во-вторых, кроме операций непосредственно над массивами, тест "STREAM" производит и вспомогательные операции, в том числе с плавающей точкой. Полученные данные подтверждают возможность применения профилировщика для оценки характеристик вычислительных процессов.

Вторым этапом стало исследование CAE-системы "FLUENT" на типовой задаче аэродинамики. Использовалась 64-битная версия исполняемого кода, оптимизированная производителем для процессоров "AMD" архитектуры "K8", к которым относится и примененный ЦП "Athlon 64 X2 4400+". Из всех процессов, запускаемых системой "FLUENT" во время счета, профилировке подвергался только процесс "fluent.6.3.26".

В целом ход вычислительного процесса можно охарактеризовать следующим образом. Счет задачи требует наличия не менее 2,5 ГБайт оперативной памяти. Планировщик операционной системы распределяет процессы так, чтобы оба ядра ЦП были заняты поровну. Поэтому часть квантов времени счет происходил на одном ядре, а часть – на другом. Соотношение операций сопроцессора по типам: загрузка, сохранение, умножение, сложение составляет примерно 2:4:3:3 (событие 0x00). Всего было выполнено 76·1010 инструкций с плавающей точкой, причем только типов SSE и SSE2. Типы инструкций x87, MMX и 3DNow не использовались (событие 0xCB). Количество инструкций над числами с плавающей точкой составляет 76,0/173,4=43,8 % всех выполненных машинных инструкций.

Было решено проанализировать те характеристики вычислительного процесса, которые инженеры компании AMD считают главными по влиянию на производительность [8]. Общая эффективность исполнения характеризуется количеством машинных инструкций в расчете на один такт ЦП (Instructions per cycle, IPC) и дается соотношением частоты событий 0xC0 и 0x76. В данном случае значение IPC равно 0,58. Здесь даже речи не идет о выполнении двух инструкций за такт, чего можно ожидать исходя из пиковой производительности данного ЦП.

"Чистое" машинное время счета дается значением частоты события 0x76, деленным на тактовую частоту ЦП, и составляет 1368 с. Количество байт, прочитанных контроллером памяти, составляет число передач (событие 0xE0, маска 0x07), помноженное на размер блока. При работе контроллера памяти в одноканальном режиме (эта конфигурация использовалась при тестировании) размер блока составляет 32 байта. Таким образом, объем данных, прочитанных контроллером, составил порядка 672 ГБайт. Средняя пропускная способность ОЗУ за время счета составила около 0,5 ГБайт/сек. Это примерно в 5 раз меньше, чем пиковая пропускная способность ОЗУ данной ЭВМ, измеренная тестом "STREAM". Следовательно, производительность ЭВМ на данной задаче почти наверняка не ограничивается пропускной способностью памяти. Более того, можно предположить, что имеется неиспользованный задел. Для уточнения этой гипотезы желательно проводить измерения только в моменты активного обращения к памяти, единственной проблемой является определение этих моментов в ситуации, когда исследуемое приложение рассматривается как черный ящик.

Интенсивность обращений к кэшу данных первого уровня (L1D) в расчете на одну инструкцию можно найти как отношение частоты событий 0x40 и 0xC0. В данном случае оно равно 0,49, то есть в среднем каждую вторую инструкцию происходило обращение к кэшу данных. Количество промахов этого кэша дается суммой частот событий 0x42 и 0x43 и равно 3,13·1010. Чтобы найти интенсивность промахов кэша, разделим эту сумму на частоту события 0xC0 и получим 0,018. Величина, обратная к данной и равная 55,4 означает, что промах кэша данных первого уровня происходит в среднем каждые 55 инструкций.

Отношение частот событий 0x43 и 0x41 показывает интенсивность промахов кэша данных, для удовлетворения которых потребовался доступ к ОЗУ. Здесь это отношение равно 14 % (то есть в среднем для каждого 7-ого промаха пришлось обращаться к ОЗУ). Тем не менее, 86 % промахов кэша первого уровня удовлетворяется объединенным кэшем второго уровня. Локальность доступа к данным находится на среднем уровне, сказывается большой размер решаемой задачи (рабочий объем ОЗУ составляет 2,5 ГБайта). Вероятно, кэши первого и второго уровней большего размера могли бы лишь незначительно улучшить производительность. В этой связи интересно было бы провести исследование нового ЦП "AMD Barcelona", обладающего кэш-памятью 3-его уровня. Определенный выигрыш могло бы дать и более оптимальное использование инструкций предвыборки из памяти.

Что касается локальности кода, то количества событий 0x82 и 0x83 наглядно демонстрируют высокую локальность вычислительных ядер программы, которые хорошо помещаются в кэш инструкций первого уровня.

Рассмотрим теперь более детально характеристики объединенного кэша второго уровня, где размещаются как инструкции, так и данные. Согласно [8], существует 3 типа событий, которые ведут к запросам кэша второго уровня. Это промахи кэшей первого уровня (инструкций и данных) и промахи буфера преобразования адреса (TLB, translation look-aside buffer). Количество промахов кэша данных первого уровня мы уже находили, оно равно 3,13·1010. Количество промахов кэша инструкций пренебрежимо мало. Количество обращений из-за промахов TLB дается частотой события 0x7E с маской 0x04 и также пренебрежимо мало. Таким образом, в нашем случае характеристики объединенного кэша второго уровня определяются целиком промахами кэша данных первого уровня.

Интенсивность обращений к буферу DTLB первого уровня всегда равна интенсивности обращений к кэшу L1D. Промахи буфера L1 DTLB делятся на те, которые удовлетворяются из буфера второго уровня (событие 0x45), и те, для удовлетворения которых приходится обращаться к ОЗУ (событие 0x46), последних всего 1,3 %. Интенсивность промахов буфера L1 DTLB в расчете на одну инструкцию дается суммой частот этих событий, отнесенной к частоте события 0xC0, и равна 0,018. Иными словами, в среднем через каждые 55 инструкций происходит промах буфера L1 DTLB. При этом происходит обращение к буферу второго уровня, L2 DTLB. Промахи данного буфера происходят гораздо реже, их интенсивность в расчете на одну инструкцию дается отношением частот событий 0x46 и 0xC0 и составляют в среднем один промах на 4230 инструкций.

В рамках данного анализа не были рассмотрены следующие аспекты вычислительного процесса: количество неверных предсказаний ветвлений, количество исключительных ситуаций при обработке чисел с плавающей точкой (Floating point exceptions).

Представляет определенный интерес ответ на следующий вопрос: как поведет себя на относительно новой 64-битной архитектуре устаревшее 32-битное приложение, решающее ту же задачу (то есть работающее с теми же входными данными)? На этот вопрос можно ответить, исследовав работу 32-битной версии "FLUENT". Она вызывается при установке переменной окружения "FLUENT_ARCH=lnx86". Объем рабочего пространства при счете не превышает 2,2 ГБайт. Количество инструкций (событие 0xC0) для 32-битной версии почти на треть больше и составляет 228·1010. Количество затраченных тактов ЦП (событие 0x76) также больше, но всего на 12 % и равно 338·1010. По-видимому, было бы преждевременным делать из этих данных вывод, что 64-битный процессор более эффективно исполняет 32-битный код. Количество обращений к кэшу L1D (событие 0x40) равно 125,8·1010. Отсюда интенсивность обращений к этому кэшу в расчете на одну инструкцию составляет 0,55. Над операндами с плавающей точкой приложение выполняет исключительно операции типа x87 (а не MMX и SSE), и их количество составляет 93,2·1010 (событие 0xCB, маска 0x01).

В данном исследовании мы рассмотрели характеристики вычислительного процесса при использовании широко распространенной CAE-системы "FLUENT" на типовой задаче аэродинамики. Анализ показывает, что код во многом неэффективно задействует возможности аппаратуры. Тем не менее, наиболее интересные результаты еще предстоит получить путем сравнения с ЭВМ другой архитектуры.

Код события	Обозначение события	Код маски	Описание маски	Количество событий, $x10^{10}$
0x76	CPU_CLK_UNHALTED	-		301,0
0x01	CYCLES_NO_FPU_OPS_RETIRED	-		45,8
0xC0	RETIRED_INSTRUCTIONS	-		173,4
0x00	DISPATCHED_FPU_OPS	0x01	Add pipe ops	29,0
		0x02	Multiply pipe	28,4
		0x04	Store pipe ops	39,9
		0x08	Add pipe load ops	1,7
		0x10	Multiply pipe load ops	6,4
		0x20	Store pipe load ops	11,4
		0x3F	All types of operations	116,8
0xCB	RETIRED_MMX_FP_INSTRUCTIONS	0x01	x87 instructions	0
		0x02	Combined MMX & 3DNow instructions	0
		0x04	Combined packed SSE & SSE2 instructions	30,0
		0x08	Combined packed scalar SSE & SSE2 instructions	46,0
		0x0F	All types of instructions	76,0
0x40	DATA_CACHE_ACCESSES	-		84,8
0x41	DATA_CACHE_MISSES	-		3,1
0x42	DATA_CACHE_REFILLS_FROM_L2_OR_SYSTEM	0x1E	All cache states except Invalid	2,7
0x43	DATA_CACHE_REFILLS_FROM_SYSTEM	0x1F	All cache states	0,43
0x80	INSTRUCTION_CACHE_FETCHES	-		75,1
0x82	INSTRUCTION_CACHE_REFILLS_FROM_L2	-		~0
0x83	INSTRUCTION_CACHE_REFILLS_FROM_SYSTEM	-		~0
0xE0	DRAM_ACCESSES	0x01	Page hit	1,08
		0x02	Page miss	0,63
		0x04	Page conflict	0,39
		0x07	All types	2,1
0x6C	SYSTEM_READ_RESPONSES	0x07	All types	1,7
0x7E	L2_CACHE_MISS	0x04	TLB reload	0,012
0x7D	REQUESTS_TO_L2	0x04	TLB reload	0,056
0x45	L1_DTLB_MISS_AND_L2_DTLB_HIT	-		3,1
0x46	L1_DTLB_AND_L2_DTLB_MISS	-		0,041

Таблица. Результаты профилировки вычислительного процесса

ЛИТЕРАТУРА:

1. В.В. Воеводин. Вычислительная математика и структура алгоритмов.—М.: Изд-во МГУ, 2006.—112 с.
2. P. Drongowski. An Introduction to Analysis and Optimization with AMD CodeAnalyst. http://developer.amd.com/documentation/articles/pages/111820052.aspx
3. FLUENT Software. http://www.fluent.com
4. OProfile - A System Profiler for Linux. http://oprofile.sourceforge.net
5. FLUENT. Exterior Flow Around a Passenger Sedan. http://www.fluent.com/software/fluent/fl5bench/flbench_6.3/problems/fl5l2.htm

6. J. McCalpin. Memory Bandwidth and Machine Balance in Current High Performance Computers, IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995. http://www.cs.virginia.edu/stream
7. BIOS and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron Processors. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/26094.PDF
8. P. Drongowski. Basic Performance Measurements for AMD Athlon 64 and AMD Opteron Processors. http://developer.amd.com/documentation/articles/pages/1212200690.aspx