УДК 519.6

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ИССЛЕДОВАНИЯ НЕСТАБИЛЬНОСТИ РАБОТЫ МУЛЬТИПРОЦЕССОРНЫХ СИСТЕМ

Ю. Г. Бартенев, А. М. Варгин, В. А. Ерзунов, В. В. Кошелев, Д. В. Кульнев, Д. И. Липов, П. С. Лобанов, А. В. Ломтев, Д. А. Новаев, А. Н. Петрик, Т. В. Семёнова, Г. П. Семёнов, С. А. Фирсов, Е. Б. Щаникова
(РФЯЦ-ВНИИЭФ)

Дается описание пакета программ Noise Measurement Suite для измерения *шума* (noise) в многопроцессорных ЭВМ и оценки его влияния на эффективность распараллеливания счета задач. Описываются программы, входящие в пакет. Представлены результаты, полученные с помощью программ пакета.

Ключевые слова: многопроцессорные ЭВМ, параллельное приложение, процесс, операционная система, язык программирования Python.

Введение

Системы, выполняющие параллельные вычисления, становятся все более востребованными в областях науки и производства, где требуется выполнение большого объема вычислений. Размеры таких систем уже сегодня достигают нескольких тысяч узлов и десятков тысяч процессоров (ядер). Наиболее актуальной является проблема эффективного использования вычислительных ресурсов сверхбольших вычислительных систем.

Одним из препятствий в решении этой проблемы является *шум* (noise) в вычислительной системе[1], который способен негативно повлиять на эффективность счета задачи, использующей большое число процессоров. По этой проблеме имеется ряд публикаций [1—5]. В работе Ф. Петрини [2] потери эффективности (задача SAGE, более 1 000 процессоров) связываются с характеристиками шумов операционной системы (ОС). Как правило, внешним признаком шума в ЭВМ является разброс времени счета одной и той же задачи при разных запусках (нестабильность счета). Шум на одном процессоре — это фактически такие накладные расходы ОС, которые не связаны с выполнением запросов исполняемой задачи, и они, как правило, составляют доли процента от всего времени решения задачи. Шум на множестве процессоров способен существенно повлиять на время счета задачи с параллельными процессами (*параллельной задачи*), поскольку помехи на множестве процессов в один и тот же промежуток времени случаются чаще, чем на одном.

Действительно, параллельные процессы задачи, как правило, выполняют на каждом процессоре последовательность небольших по времени шагов алгоритма, заканчивающихся общей синхронизацией. Назовем такие шаги *зернами*. В таком случае возникающее из-за помехи удлинение времени выполнения зерна случается чаще, чем на одном процессоре, и приводит к увеличению количества зерен в процессе счета параллельной задачи, на которых случилась такая задержка. Поэтому небольшие помехи счету на каждом процессоре способны привести к значительному увеличению времени счета параллельной задачи. Чем меньше длительность зерен в задаче (и, значит, больше их количество), тем большее влияние на увеличение времени задачи оказывает шум, что будет показано далее.

[1] Случайные помехи вычислительному процессу, например, вследствие неравномерной работы множества элементов из-за возникающих между ними конфликтов, разновременной работы системы управления многопроцессорной ЭВМ на различных процессорах.

Предметом данной работы является методика измерения шума в вычислительной системе и оценка его возможного влияния на эффективность счета параллельных задач. В статье описывается разработанный авторами пакет программ Noise Measurement Suite (далее NoM Suite). NoM Suite обеспечивает автоматизированный процесс измерения, сбора, представления и анализа результатов измерения нестабильности функционирования аппаратно-программных компонентов и подсистем созданной ЭВМ. На основании исследования с помощью NoM Suite можно сделать качественные выводы о характере шума и его значимых составляющих, чтобы затем предпринять меры по оптимизации ЭВМ, направленные на снижение шума. Пакет позволяет также измерить нестабильность выполнения как параллельных прикладных приложений, входящих в его состав, так и пользовательских параллельных приложений. При создании NoM Suite решались следующие задачи:

- выявить шум;
- охарактеризовать шум небольшим количеством макроскопических параметров;
- предсказать влияние шума на эффективность распараллеливания.

NoM Suite — это средство диагностики шума в многопроцессорной ЭВМ и прогнозирования его влияния на снижение масштабируемости приложений.

1. Состав пакета программ

В состав пакета NoM Suite входят:

1. Библиотека общих низкоуровневых функций (инструменты для измерения, сохранения, передачи и чтения результатов измерения).
2. *Функции-обертки* (wrappers) MPI-вызовов, регистрирующие время и тип MPI-вызовов в параллельном приложении.
3. Программы-измерители шума для измерения нестабильности времени:
   - вычислительной работы;
   - операций ввода-вывода;
   - работы процессора с оперативной памятью;
   - выполнения MPI-обменов.
4. Программы генерации шума.
5. Прикладные тесты.
6. Тестовая подсистема для автоматического запуска приложений.
7. Подсистема представления результатов измерений и анализа:
   - интегральных показателей шума;
   - влияния шума на время выполнения приложения путем моделирования и применения аналитических формул.

2. Библиотека низкоуровневых функций

В состав этой библиотеки входят средства временных замеров. Библиотека поддерживает прозрачный для пользователя механизм сохранения результатов измерений в базе данных. Для высокоточных замеров (с точностью до такта) в библиотеке используется прямой доступ к регистрам процессора.

Реализовано два типа измерений:

1) времени исполнения фиксированного объема кода;
2) объема выполненного кода (в элементарных инструкциях) при фиксированном времени исполнения.

В библиотеке реализовано несколько методов, предназначенных для преобразования результатов временных замеров. Библиотека низкоуровневых функций предоставляет интерфейс для доступа к сохраненным данным в базе данных.

3. Библиотека функций-оберток MPI-вызовов и функций ввода-вывода

Функции библиотеки предназначены для получения данных о *зернистости* различных реальных параллельных приложений, нагрузке на коммуникационную сеть и систему ввода-вывода, о времени прерывания арифметической работы в ожидании операций обмена и ввода-вывода.

Накапливаемые данные сохраняются в текстовых файлах либо по завершении параллельного приложения, либо через заданный пользователем интервал времени.

4. Набор измеряющих программ

Сборщик шума на вычислительных узлах ЭВМ. Эта параллельная программа запускается на всех процессорах ЭВМ или на их представительном множестве и обеспечивает сбор шума на каждом процессоре. Все процессы, стартуя одновременно, в цикле измеряют длительность выполнения одной и той же короткой вычислительной работы (*кванта*). Длительность выполнения кванта в этой программе не зависит от скорости доступа к памяти и составляет несколько микросекунд. Во время всего цикла процессы не выполняют никаких обменов[2]. Поскольку длительность исполнения кванта отличается от эталонной длительности только за счет работы ОС на каждом узле и демонов системы управления многопроцессорной ЭВМ[3], то данная программа позволяет выявить шум от системы управления ЭВМ.

В начале работы определяются показания часов на каждом процессе в один и тот же момент времени для последующего определения одновременности событий на разных процессах. Для корректного измерения длительности выполнения мелкой работы сборщик использует высокоточные функции замера времени из библиотеки низкоуровневых функций. Для получения наиболее достоверной картины возникновения шума программа-сборщик должна работать длительное время, например 1 час. По окончании измерений каждый процесс сохраняет в отдельном файле:

- минимальное время выполнения кванта;
- длительности всплесков шума по разности фактического и минимального времени выполнения кванта;
- время старта всплеска шума (по существенному отклонению фактического времени выполнения кванта от минимального).

Измеритель нестабильности решателей систем линейных алгебраических уравнений. Данная программа разработана на базе библиотеки параллельных решателей систем линейных уравнений PMLP/Parsol [6]. С помощью нее измеряются отклонения от номинального времени выполнения следующих операций:

- решения системы линейных уравнений стабилизированным методом бисопряженных градиентов;
- умножения матрицы на вектор;
- скалярного умножения векторов $(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$;
- вычисления второй нормы вектора $||\mathbf{x}||_2 = \sqrt{\sum |x_i|^2}$.

Измеритель времени доступа процессора к различным участкам оперативной памяти. Основные возможности программы:

- получение скоростных характеристик обмена процессора с памятью при выполнении целочисленных вычислений и вычислений с плавающей точкой;
- задание режимов и параметров тестирования:

[2]Подключение библиотеки MPI к сборщику мотивировано необходимостью сбора информации в начале и в конце работы сборщика.

[3]Возможна также различная номинальная производительность процессоров ЭВМ, что предварительно выявляется с помощью программы-измерителя.

- объема обрабатываемых данных;
- количества повторов теста;
- количества процессов.

Логика программы основана на известном тесте памяти Stream [7]. На рис. 1 представлены результаты выполнения программы

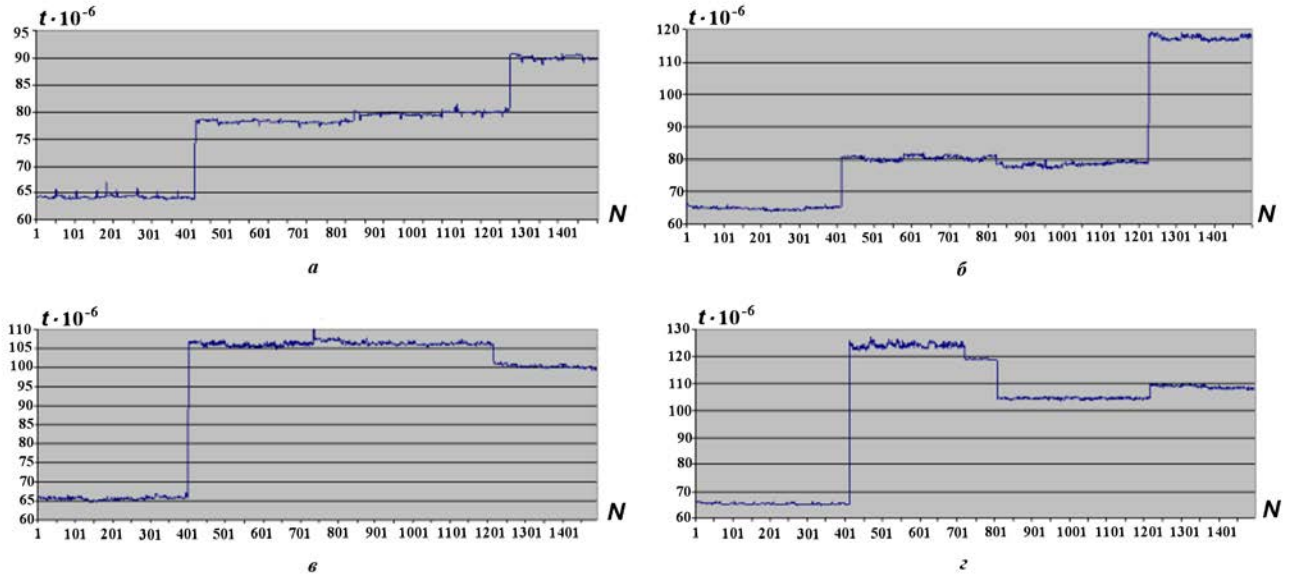

Рис. 1. Время доступа к различным участкам оперативной памяти (N — номер участка) на одном узле (Numa-архитектура): *а* — без шума; *б* — шум в одном процессе; *в* — шум в двух процессах; *г* — шум в трех процессах

5. Измерение нестабильности коммуникационной среды

С помощью представленных ниже тестов осуществляется измерение нестабильности пропускной способности коммуникационной системы многопроцессорной ЭВМ при выполнении всех типов MPI-обменов при многократном цикле выполнения одной и той же работы:

Тест с соседями. Эмулируется работа прикладных задач при обмене с соседними процессами в трехмерной коммуникационной решетке процессов с замером времени выполнения цикла обменов с ближайшими по решетке процессами[4].

Тест по направлениям. В трехмерной решетке процессов моделируется сдвиг данных от предыдущего процесса к следующему в каждом направлении X, Y, Z решетки. Сначала данные следующему в выбранном направлении процессу посылают все четные процессы, затем — все нечетные.

"Точка-точка". Последовательно перебираются все возможные пары процессов, и между ними осуществляется обмен данными заданное число раз. Каждым процессом производится замер времени выполнения этих обменов.

"Мастер". Берется один процесс (*мастер*), и осуществляется посылка данных от каждого процесса мастеру. В качестве мастера последовательно берется каждый процесс. Процесс, осуществляющий сбор данных, производит замер времени этой операции.

"Бисекция". Используется для измерения нестабильности операции бисекции.

[4]Большими порциями в направлениях X, Y, Z решетки и небольшими порциями по диагоналям решетки.

6. Программы генерации шума

При функционировании многопроцессорной вычислительной системы в многопользовательском режиме важно исследовать влияние на приложение, использующее часть вычислительных ресурсов, нагрузки со стороны другой части вычислительных ресурсов. Для возбуждения этой нагрузки могут быть применены генераторы шума.

Сетевой шум. Вычислительные узлы многопроцессорных ЭВМ, кроме выполнения вычислительных функций, вынуждены осуществлять обмены в сети управления ЭВМ, откликаясь на запросы системы управления заданиями, системы временной синхронизации, файловой системы и т. д. Программа-генератор сетевого шума предназначена для имитации сетевой активности, которая порождается реальными сетевыми сервисами, а также для измерения вызываемого шума.

Для измерения и анализа уровня сетевого шума программа-генератор создает трафик различной интенсивности в соответствии с набором настраиваемых параметров, задающих:

- тип транспортного протокола;
- размер буферов приема/передачи на уровне приложения и протоколов как клиента, так и сервера;
- частоту приема/передачи данных со стороны клиента и сервера;
- количество единовременно посылаемых сообщений;
- режим обработки запросов сервером (последовательный, параллельный).

Шум ввода/вывода. Помимо основного счетного процесса, на вычислительном узле также выполняются служебные программы, реализующие операции ввода/вывода.

Подсистема ввода/вывода выполняет запросы файловой подсистемы и подсистемы управления процессами для доступа к периферийным устройствам (дискам, магнитным лентам, терминалам и т. д.). Она обеспечивает необходимую буферизацию данных и взаимодействует с драйверами, непосредственно обслуживающими внешние устройства. Соответственно каждая операция ввода/вывода требует часть вычислительных ресурсов системы, что может сказаться негативно на счетном процессе.

Для выполнения операций ввода/вывода, создающих нагрузку на подсистему ввода/вывода ОС, разработана специальная программа. Основными возможностями данной программы являются:

- указание объемов данных чтения/записи;
- указание размеров буфера ввода/вывода;
- выбор между синхронным/асинхронным режимами;
- генерация файлов;
- запись/чтение файлов;
- получение скоростных характеристик записи/чтения/генерации файлов.

Шум управления памятью. Поскольку счетный процесс использует оперативную память вычислительной системы, работа с памятью сторонних процессов сказывается на времени счета. Для исследования такого влияния разработана программа, имитирующая шум подсистемы управления памятью. Основными возможностями данной программы являются:

- указание объемов выделяемой памяти;
- указание количества повторов операции;
- выбор режима работы с памятью (запись/чтение);
- получение скоростных характеристик операций записи/чтения с памятью.

7. Тестовая подсистема

Тестовая подсистема обеспечивает автоматизированный процесс сборки приложений, их запуска и выполнения, измерения и сбора результатов измерения на исследуемой многопроцессорной вычислительной системе. Результаты измерений являются источником для системы анализа.

В тестовой подсистеме на этапе конфигурации предусмотрена пользовательская настройка на используемую систему управления заданиями. Конфигурирование подсистемы заключается в создании главного конфигурационного файла на языке XML, который формируется специальной диалоговой программой, использующей графический интерфейс, и содержит информацию о запускаемых приложениях и их очередности.

8. Подсистема анализа

Подсистема анализа включает:

- инструментальные средства представления результатов измерений в табличном и графическом виде, как из базы данных, так и из файлов;
- средства динамического представления результатов измерения (в ходе выполнения задания) в виде осциллограмм;
- инструментальные средства анализа результатов и предсказания.

Браузер. Браузер написан на языке Python с использованием графической библиотеки wxPython [8]. Программа имеет оконный интерфейс, и ее структура может расширяться посредством *плагинов*[5].

Браузер включает набор плагинов, реализующих построение специальных типов графиков по прочитанным данным с использованием модулей расширения Python: matplotlib [9], gnuplot-py [10], vtk [11] и др. Другой набор плагинов осуществляет связь с функциями, реализующими некоторую постобработку полученных результатов, например, вычисление нормы шума для узла. Существуют плагины, реализующие использование вспомогательных инструментов для дополнительного анализа полученных результатов (например OpenOffice [12], gnuplot [13], ParaView [14]).

Получение характеристик собранного шума в вычислительных узлах многопроцессорной ЭВМ. Соответствующая программа помогает выявить некоторые закономерности всплесков шума, собранных программой-сборщиком шума на вычислительных узлах кластера. Результатом ее работы являются таблицы, в которых шум разложен на несколько составляющих по диапазонам длительности всплесков. Для каждой составляющей подсчитывается средний период между всплесками, суммарное время составляющей и другие интегральные параметры (более подробно см. ниже). В частности, построив графики по результатам этой программы, можно увидеть, какие процессоры и узлы были самыми *шумными* во время сбора шума. Для самых шумных процессов выдается отдельная статистика.

Моделирование влияния шума на *мелкозернистое* параллельное приложение. Имея файлы, созданные сборщиком шума многопроцессорной ЭВМ, можно подсчитать ожидаемое время выполнения параллельной задачи, которая представляет собой последовательность вычислительных зерен разной длительности. Для этого предназначена программа моделирования выполнения параллельного приложения в условиях шума, которая на каждом шаге своей работы вычисляет время окончания выполнения зерна приложения с учетом всплесков шума, которые происходили в это же самое время (от начала счета) в программе-сборщике.

Входная информация:

— коллекция всплесков шума на каждом из N процессоров ЭВМ;

— список длительностей вычислительных зерен приложения.

Результат, выдаваемый программой:

1) $T_p, T_{p_{\min}}, T_{p_{\max}}$ — ожидаемые среднее, минимальное и максимальное времена работы приложения, в котором каждый из p процессов выполняет зерна. Берутся из моделирования множества

[5] *Plugins* — скрипты на языке Python, реализующие любую логическую функцию.

последовательных запусков приложения, которые выполнялись бы друг за другом в течение всего интервала измерений, когда проводился сбор всплесков шума;

2) $E_p = T_1/T_p \cdot 100\,\%$; $E_{p_{\min}} = T_1/T_{p_{\max}} \cdot 100\,\%$; $E_{p_{\max}} = T_1/T_{p_{\min}} \cdot 100\,\%$ — ожидаемые средняя, минимальная и максимальная эффективности распараллеливания приложения.

Ограничения: $p < N$; время обмена при синхронизации процессов в приложении считается равным 0.

9. Метрики шума

Для оценки влияния шума на конкретную параллельную задачу можно ввести понятие *нормы шума в приложении*, которая подсчитывается в результате многократного прогона приложения:

$$N = \frac{t_{ave} - t_{\min}}{t_{\min}},$$

где t_{ave} — среднее время одного прогона; $t_{\min}$ — минимальное время одного прогона.

Проведя такие (косвенные) измерения шума для разных характерных задач, можно представить ожидаемое снижение эффективности счета на данной ЭВМ и, зафиксировав набор задач, сопоставлять ЭВМ между собой. Однако проведение таких экспериментов — дорогостоящая операция, и выбор множества задач для оценки шума вряд ли можно стандартизировать.

Для оценки шума в ЭВМ[6] рациональнее использовать следующий подход: предварительно собрать осциллограмму (последовательность и длительность всплесков) шума на всех процессорах ЭВМ, затем промоделировать ее влияние на увеличение времени счета задач с характерной длительностью зерен. В таком случае однажды собранный шум позволяет предсказать поведение различных задач без их прогона. Хотя выполнение реальной параллельной задачи искажает шум, так как шум не является абсолютно независимым от задачи процессом, тем не менее такой подход позволяет своевременно оценить возможное снижение эффективности счета различных задач на ЭВМ, заняться поиском влиятельных источников шума и оптимизацией системы управления ЭВМ.

Реализация такого подхода дает возможность ввести обобщенные показатели (метрики) шума, а именно долю покрытия шумом интервала измерений и степень синхронности всплесков шума, позволяющие судить о шуме в вычислительных узлах количественно и сравнивать с помощью них различные многопроцессорные ЭВМ. Такое обобщение выполняет программа-обработчик шума, обнаруженного программой-сборщиком, через первичные характеристики шума.

Для этого собранный шум разлагается на составляющие по длительности всплесков (например, длительностью от 100 до 500 мкс, от 500 до 1 000 мкс и т. д.). Затем для j-й составляющей подсчитываются: d_j — средняя длительность всплеска шума на всех процессах; $D_{j,k}$ — сумма длительностей всплесков шума на k-м процессе; g_j — средний интервал между всплесками шума на всех ($p_{\max}$) процессах; $U_{j,p}$ — длительность объединения всплесков шума на p процессах[7]. По полученным значениям вычисляются макроскопические показатели шума:

$C_{j,p} = U_{j,p}/T$ — доля покрытия шумом интервала измерений T, т. е. доля объединения всплесков шума на $p \le p_{\max}$ процессах;

$S_{j,p} = \sum\limits_{k=1}^{p} D_{j,k} / (pU_{j,p})$ — коэффициент синхронности всплесков шума, где p — число процессов, на которых был обнаружен шум j-й составляющей. (Отсюда следует $S_{j,p} = 1$ при полной синхронности всплесков шума и $S_{j,p} = 1/p$ при полной асинхронности всплесков шума.)

Переходя к $p = p_{\max}$ или объединению по j, можно получить интегральные показатели шума в ЭВМ: C_j, S_j, C_p, S_p, C, S, зависящие только от одного параметра либо не зависящие от обоих, получая ту или иную степень детализации. По ним можно сравнивать различные суперЭВМ между собой по степени шума, а также степень нарастания шума и его составляющих при возрастании p до $p_{\max}$.

[6] В основном от системы управления ЭВМ — одной из наиболее "влиятельных" поставщиков шума.

[7] Объединению всплесков шума на p процессах принадлежат те и только те точки интервала измерений, в которых был шум хотя бы в одном процессе.

10. Оценка влияния шума на параллельное приложение

Рассмотрим влияние шума на параллельное приложение, в котором все процессы синхронно выполняют зерна вычислений (рис. 2, *а*). При возникновении шума длительность зерна увеличивается на максимальную (по всем процессам) сумму длительностей всплесков шума, возникающих во время выполнения зерна на одном процессе, что приводит к общему увеличению времени счета.

На рис. 2, *б* (см. также цветную вкладку) темно-серым цветом обозначено ожидание других процессов. Несмотря на то, что ни на одном процессе помехи не превысили время d, общая длительность выполнения приложения увеличилась на $2d$.

Доля покрытия шумом C может грубо характеризовать влияние шума в вычислительных узлах ЭВМ (в основном от компонентов системы управления) на приложение, а именно: доля покрытия шумом C может служить верхней оценкой возможной степени удлинения времени исполнения параллельного приложения из-за шума.

Для более точной оценки влияния шума получена полуэмпирическая формула времени счета параллельного *равнозернистого* приложения, выполняющего k зерен длительностью $t < g_i$ (редкие всплески, что характерно для шума от системы управления):

$$T_p = k\left(t + \sum_j P_{p,j} d_j\right), \quad P_{p,j} = 1 - \left(1 - \frac{t}{g_j}\right)^{1/S_{j,p}} \quad \text{при} \quad t < g_j,$$

где модель шума представлена в виде суперпозиции всплесков шума длительностями d_j, возникающих через промежутки g_j на p процессорах. Отношением kt — времени исполнения равнозернистого приложения на p процессорах при отсутствии шума — к T_p получим коэффициент снижения производительности вычислений из-за шума:

$$E_p = \frac{t}{t + \sum\limits_j P_{p,j} d_j}. \tag{1}$$

Подставляя в данную формулу характеристики, вычисленные программой-обработчиком шума, можно оценить влияние шума на эффективность распараллеливания приложений заданной (средней) зернистости при их масштабировании с одновременным увеличением числа процессоров.

Для более точного предсказания влияния шума на параллельное *разнозернистое* приложение лучше использовать программу моделирования выполнения параллельного приложения (см. разд. 8). Принцип ее работы состоит в том, что на цикл зерен моделируемого приложения накладывается собранный программой-сборщиком шум в той временной последовательности, в которой он случился на процессорах ЭВМ в процессе сбора, и, таким образом, получается информация по удлинению времени зерен.

На рис. 3 представлены графики, построенные по результатам применения формулы (1) и моделирования для одного из алгоритмов решения систем линейных алгебраических уравнений.

Рис. 2. Выполнение зернистого приложения: *а* — без всплесков шума; *б* — с всплесками шума (закрашены черным цветом) длительностью d и $0{,}5d$

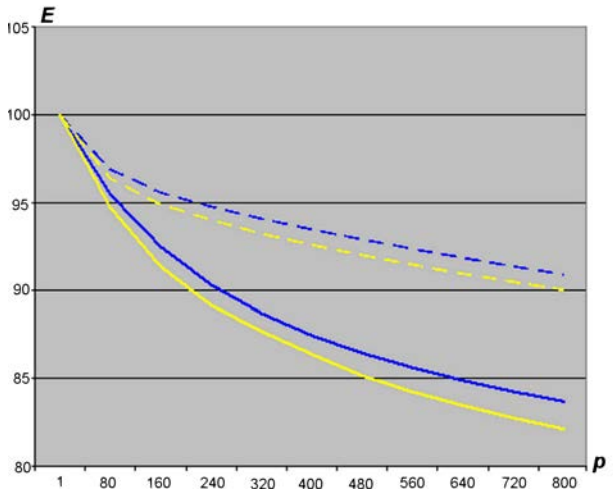

Рис. 3. Ожидаемое снижение эффективности (из-за шума от программного обеспечения ЭВМ) выполнения метода сопряженных градиентов: —— — 8 000 строк матрицы в одном процессе (средняя длительность зерна 0,001637); - - - — 27 000 строк матрицы в одном процессе (средняя длительность зерна 0,006266); темные кривые — по формуле (1); светлые кривые — по программе моделирования

11. Вероятностно-статистический метод расчета и анализа влияния шума на эффективность выполнения параллельных приложений

Рассмотрим следующую упрощенную модель счета сбалансированного параллельного вычислительного приложения. Система состоит из n процессоров. Счет разбит на L этапов, при этом данные задачи разбиты на n блоков. На каждом этапе каждый процессор обрабатывает (или повторно обрабатывает) один блок. Переход к следующему этапу происходит только тогда, когда все процессоры заканчивают счет предыдущего шага задачи. Время обработки блока одним процессором равно $t_1 = t_0 + x$, где t_0 — чистое время счета на процессоре при отсутствии шума, x— случайная задержка, равная времени шума (далее просто шум).

Время счета одного этапа равно максимальному из всех затраченных процессорами времен. Если на некотором этапе счета шум на процессорах — $(x_1, x_2, \ldots, x_n)$, то время счета данного этапа будет $t_n = t_0 + \max(x_1, x_2, \ldots, x_n)$.

Поскольку шум является случайной величиной, времена счета этапа и задачи в целом также будут случайными величинами. Поэтому интересным представляется вопрос о функции распределения времени счета как данного этапа, так и всей задачи в целом. В частности, для практических целей важен вопрос о зависимости данной функции распределения от количества процессоров в ЭВМ.

Обозначим через $f(x_1, x_2, \ldots, x_n)$ плотность вероятности распределения шума при счете на одном этапе. Данная функция нормирована на единицу: $\int\limits_0^\infty \ldots \int\limits_0^\infty dx_1 \ldots dx_n f(x_1, \ldots, x_n) = 1$.

Обозначим через $g(x)$ плотность вероятности распределения шума при счете параллельного приложения как единого объекта на данном этапе. Несложно понять, что она связана с $f(x_1, x_2, \ldots, x_n)$ следующим образом:

$$g(x) = \sum_{i=1}^{n} \int\limits_0^x \ldots \int\limits_0^x dx_1 \ldots dx_{i-1}, dx_{i+1} \ldots dx_n f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n).$$

Действительно, вероятность того, что максимальное значение шума принадлежит интервалу $(x, x + dx)$, равна сумме вероятностей следующих событий: того, что первый элемент принадлежит данному интервалу, а остальные меньше; того, что второй элемент принадлежит данному интервалу, а остальные меньше, и т. д. Несложно проверить, что функция $g(x)$ также нормирована на единицу: $\int\limits_0^\infty g(x)\, dx = 1$.

Зная функцию распределения, можно вычислить среднее значение и дисперсию времени счета для одного этапа:

$$\bar{t}_n = t_0 + \int_0^\infty xg\,(x)\,dx; \qquad D_n = \int_0^\infty x^2 g\,(x)\,dx - \left(\int_0^\infty xg\,(x)\,dx\right)^2.$$

Поскольку все этапы независимы, то для достаточно большого количества этапов, т. е. при $L \to \infty$, функция распределения $F\,(T)$ полного времени счета T будет иметь вид распределения Гаусса со средним значением $L\bar{t}_n$ и дисперсией LD_n:

$$F\,(T) \approx \frac{1}{\sqrt{2\pi L D_n}} e^{-\frac{T - L\bar{t}_n}{2LD_n}}.$$

Таким образом, для получения информации о распределении времени счета параллельной задачи необходимо вычислить $\bar{t}_n$ и D_n.

Максимальная эффективность параллельного приложения, понижающаяся из-за наличия шума в вычислительной системе, может быть вычислена следующим образом:

$$\varepsilon = \frac{Lt_0}{Lt_0 + \int_0^\infty ZF\,(Z)\,dZ} \cdot 100\,\%,$$

где $F\,(Z)$ — плотность вероятности задержки параллельного приложения; $Z = z_1 + z_2 + \ldots + z_L$, z_i представляет собой случайную величину, характеризующую задержку на i-м шаге.

Для n независимых идентичных процессоров, на которых выполняется параллельное приложение, для больших значений L после некоторых преобразований было получено асимптотическое значение, не зависящее от L:

$$\varepsilon \approx \frac{t_0}{t_0 + n\int_0^\infty zh\,(z)\left(\int_0^z h\,(y)\,dy\right)^{n-1} dz} \cdot 100\,\%. \qquad (2)$$

Используя данную формулу можно получить вероятностные оценки влияния шума на эффективность выполнения сбалансированных параллельных приложений в однородных мультипроцессорных вычислительных системах. Входными параметрами являются количество процессоров и функция распределения шума.

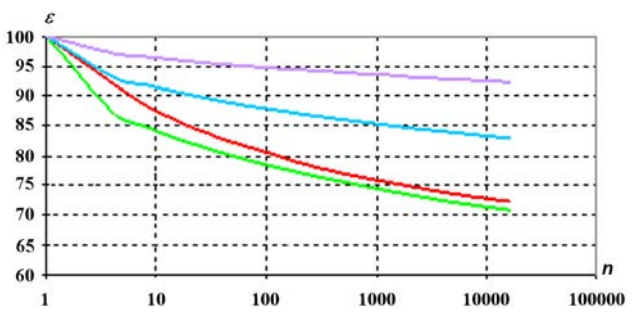

Рис. 4. Изменение эффективности при увеличении числа процессоров для разных значений σ: темная кривая — по формуле (2), $\sigma = 10$ мкс; светлые кривые — результаты моделирования при $\sigma = 10; 5; 2$ мкс

Кроме того, для рассматриваемой модели счета сбалансированного параллельного вычислительного приложения был реализован программный модуль, позволяющий определить статистическими методами распределение шума при выполнении задачи и получить аналогичные оценки путем прямого численного моделирования общего времени счета.

На рис. 4 (см. также цветную вкладку) представлены результаты численных расчетов с использованием формулы (2) и прямого численного моделирования. При этом задавались следующие параметры: время выполнения шага приложения в отсутствие шума — 100 мкс; функция распределения задержки на одном процессоре $h\,(x)$ — гауссиан с параметром $\sigma = 10$ мкс. Для сравнения на рисунке приведены также зависимости эффективности

параллельного приложения для других значений σ (5 мкс, 2 мкс), полученные в результате прямого численного моделирования.

Анализируя коллекцию шума, полученную программой-измерителем для зерна с длительностью выполнения t_0 на одном узле, и подобрав соответствующую функцию плотности, с использованием формулы (2) можно вычислить эффективность выполнения параллельного приложения с данной длительностью зерна для произвольного количества процессов.

Заключение

Разработанный пакет программ позволяет обнаружить и измерить характеристики аппаратно-программного шума многопроцессорной ЭВМ и оценить его влияние на эффективность выполнения параллельных приложений, что представляется полезным для разработчиков многопроцессорных ЭВМ.

В дальнейшем планируется провести исследования аппаратно-программного шума конкретных ЭВМ.

Работа выполнена совместно с Сандийской национальной лабораторией (грант CRDF № RUM2-1553-SV-05).

Список литературы

1. *Gioiosa R., Petrini F., Davis K., Lebaillif-Delamare F.* Analysis of system overhead on parallel computers // IEEE Int. Symp. on Signal Processing and Information Technology. Rome, Italy. December, 2004.
2. *Petrini F., Kerbyson D. J., Pakin S.* The case of missing supercomputer performance: achieving optimal performance on the 8,192 processors of ASCI Q // IEEE/ACM SC2003. Phoenix, AZ. November, 2003.
3. *Terry P., Shan A., Huttenen P.* Improving application performance on HPC systems with process synchronization // Linux Journal. 2004. Vol. 127. P. 68—73. http://portal.acm.org/citation.cfm?id=1029015.1029018.
4. *Terry P.* Improving application performance on HPC systems with process synchronization. http://linuxjournal.com/article/7756.
5. *Weinberg J., McCracken M. O., Snavely A., Strohmaier E.* Quantifying locality in the memory access patterns of HPC applications // Supercomputing. Seattle, WA. November 12—16, 2005.
6. *Artemyev A. Yu., Bartenev Yu. G., Bondarenko Yu. A. et al.* The library of solvers for sparse linear systems. Capabilities, use, development prospects // "HERCMA 2003". September 25—27, 2003. Athens University of Economics and Business. Atheths, Hellas. Book of Abstracts. P. 227B.
7. *McCalpin J. D.* STREAM. Sustainable memory bandwidth in high performance computers. www.cs.virginia.edu/stream.
8. *Rappin N., Dunn R.* wxPython in Action. Manning Publication, 2006.
9. *Dale D., Droettboom M., Firing E.* Matplotlib. Release 0.99.0. August, 2009. http://matplotlib.sourceforge.net/Matplotlib.pdf.
10. Gnuplot.py.home page. http://gnuplot-py.sourceforge.net/.
11. The VTK User's Guide. Kitware Inc., 2004.
12. *Смирнов Д., Чернов Д., Ерёменко А. и др.* OpenOffice.org для профессионала. М.: ДМК Пресс, 2008.
13. *Williams T., Kelley C.* Gnuplot. An interactive plotting programm. http://www.gnuplot.info/docs/gnuplot.pdf.
14. *Hendersen A.* The ParaView Guide. Kitware Inc., 2004.

Статья поступила в редакцию 10.06.09.