ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 38 за 2012 г.

Дьяченко С.В., Багдасаров Г.А., Ольховская О.Г.

Средства профилирования и анализа многопоточных приложений Oracle (Sun) Studio Performance Analyzer

Рекомендуемая форма библиографической ссылки: Дьяченко С.В., Багдасаров Г.А., Ольховская О.Г. Средства профилирования и анализа многопоточных приложений Oracle (Sun) Studio Performance Analyzer // Препринты ИПМ им. М.В.Келдыша. 2012. № 38. 15 с. URL: http://library.keldysh.ru/preprint.asp?id=2012-38

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

С.В. Дьяченко, Г.А. Багдасаров, О.Г. Ольховская

Средства профилирования и анализа многопоточных приложений Oracle (Sun) Studio Performance Analyzer

Москва — 2012

С.В. Дьяченко, Г.А. Багдасаров, О.Г. Ольховская

Средства профилирования и анализа многопоточных приложений Oracle (Sun) Studio Performance Analyzer

Свободно распространяемое ПО Oracle Studio Performance Analyzer может использоваться разработчиками для выполнения различных типов анализа параллельных приложений для систем как с распределенной, так и с общей памятью. При помощи этого инструментария можно получить ответы на различные вопросы, относящиеся к производительности и эффективности параллельного кода. К сожалению, в свободном доступе отсутствует описание этого полезного инструмента на русском языке. Настоящий документ ставит своей целью восполнить этот пробел.

Ключевые слова: свободное ПО, параллельные приложения, анализ производительности.

S.V. Dyachenko, G.A. Bagdasarov, O.G. Olkhovskaya

Oracle (Sun) Studio Performance Analyzer - profiling and analysis tools for multithreaded applications.

Free software Oracle Studio Performance Analyzer can be used by developers to perform various types of performance analysis for both distributed and shared memory parallel applications. With this tool one can obtain various information related to the performance and efficiency of a parallel code. Unfortunately, there is no free description of this useful tool in Russian. This document aims to fill this gap.

Key words: Free software, parallel applications, performance analysis.

Введение

Oracle Solaris Studio (OSS, ранее Sun Studio) — интегрированная среда разработки программ для языков программирования Си, C++ и Фортран, разработанная компанией Sun Microsystems. В OSS включены средства сборки, отладки, профилирования и анализа многопоточных приложений.

Ранее Oracle Solaris Studio называлась Sun Workshop, Forte Developer, Sun ONE Studio и была доступна только на платформе Sun Solaris. Сейчас Oracle Solaris Studio доступна также для OpenSolaris и дистрибутивов на её основе, есть также версия и для Linux. После покупки Sun корпорацией Oracle продукт сменил название с Sun Studio на Oracle Solaris Studio.

Начиная с версии 11, выпущенной в 2005 году, компания Sun Microsystems стала предоставлять разработчикам, зарегистрировавшимся на сайте в Sun Developer Community, возможность бесплатно скачать Sun Studio для Solaris и Linux, который раньше продавался только за отдельную плату. С 2010 года, в результате перехода активов Sun Microsystems в Oracle, среда доступна бесплатно с правом дальнейшего бесплатного распространения по специализированной лицензии. Обновления среды выпускаются синхронно с крупными обновлениями Solaris. Адрес ресурса:
http://www.oracle.com/technetwork/server-storage/solarisstudio/overview/index.html

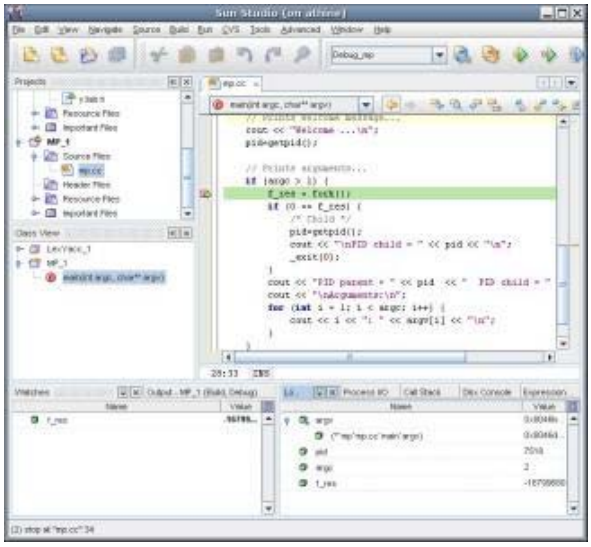

Рисунок 1 — Общий вид рабочего окна Solaris Studio.

Работа поддержана отделением математических наук РАН (ПФИ 3.6 ОМН РАН) и Российским фондом фундаментальных исследований (проект 11-07-93939-Джи8_а)

Компоненты Oracle Solaris Studio:

— Компиляторы Си, C++ и Fortran

— Графическая среда разработки, базирующаяся на NetBeans

— Отладчик dbx, интегрированный со средой разработки

— Статические верификаторы кода lint и lock_lint

— Инструмент для распределенной или параллельной сборки приложений dmake

— Профилировщик Performance Analyzer

— Инструмент для поиска ситуаций «data race» — Thread Analyzer

— Инструмент для поиска утечек памяти и ошибок, связанных с неправильным доступом памяти — RTC (Run-Time Checking); является частью dbx.

Свободно распространяемое ПО Oracle Studio Performance Analyzer (далее просто Performance Analyzer) может использоваться разработчиками для выполнения различных типов анализа параллельных приложений для систем как с распределенной, так и с общей памятью. При помощи этого инструментария можно получить ответы на различные вопросы, относящиеся к производительности и эффективности параллельного кода:

— Приведет ли дальнейшая оптимизация параллельной составляющей кода к существенному приросту производительности?

— Что играет более важную роль в производительности кода: синхронизация вычислений или передача данных?

— Насколько сбалансирована загрузка вычислительных узлов и процессоров?

— Какова характерная длина итерации параллельного вычисления?

— Сколько времени требуется для достижения асимптотической производительности?

— Каковы характерные шаблоны обмена данными в приложении?

— Какие сообщения играют большую роль: длинные или короткие?

— Есть ли синхронизация отправителей и получателей сообщений?

Процесс анализа параллельного приложения с помощью Performance Analyzer:

— трансляция исходного кода приложения с соответствующими опциями;

— выполнение эксперимента, т.е. профилирование приложения с помощью Performance Analyzer;

— анализ результатов эксперимента, т.е. обработка собранной информации средствами Performance Analyzer.

Performance Analyzer может использоваться как в консольном режиме, так и с применением графического интерфейса пользователя (GUI). При выполнении анализа результатов экспериментов в числе прочего доступна следующая информация:

— метрики по отдельным функциям;
— метрики по графам вызовов;
— метрики по отдельным строкам исходного кода;
— метрики по дизассемблированному коду;
— метрики по вычислительным системам;
— метрики по отдельным типам данных;
— метрики по объектам данных;
— шкала времени с отображением различных событий в программе.

Описание интерфейса Performance Analyzer

В разделе кратко описаны вкладки Performance Analyzer, доступные метрики перечислены ниже в таблице 1.

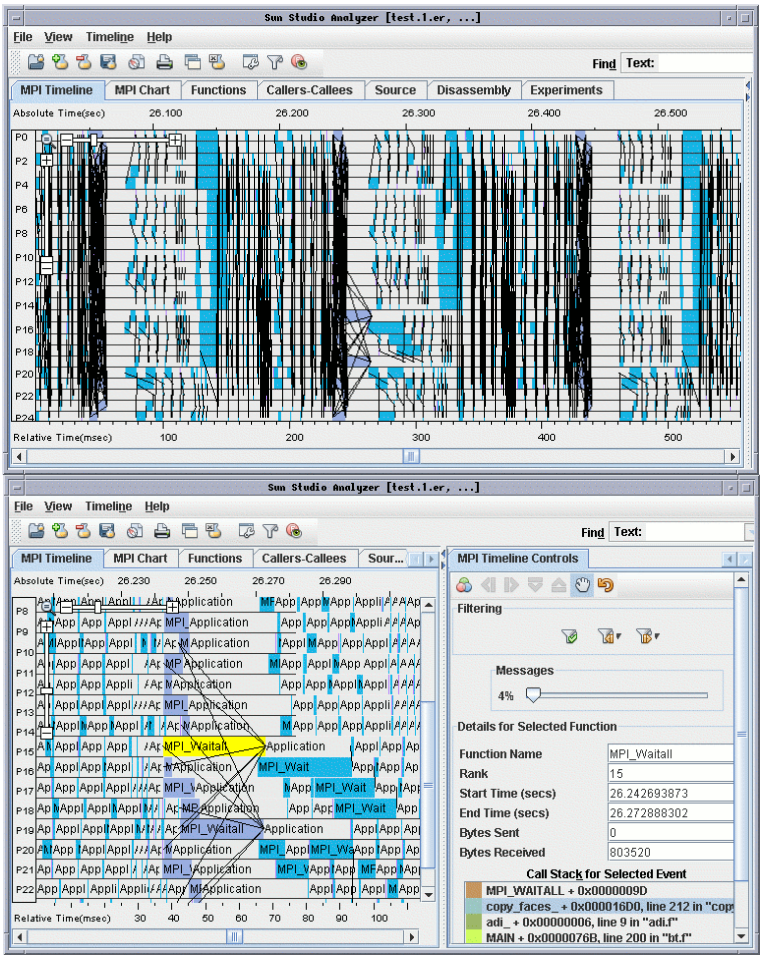

Рисунок 2 — Performance Analyzer. Шкала времени MPI.

Шкала времени MPI (вкладка **MPI Timeline**, рис. 2) отображает относящиеся к MPI события, возникшие во время исполнения программы, в удобной графической форме. По вертикали отображаются процессы параллельной программы, а по горизонтали — ось времени. Возможно выполнение дополнительных манипуляций с параметрами отображения (масштабирование, фильтрация и т.д.) с целью акцентирования тех или иных деталей параллельной работы программы.

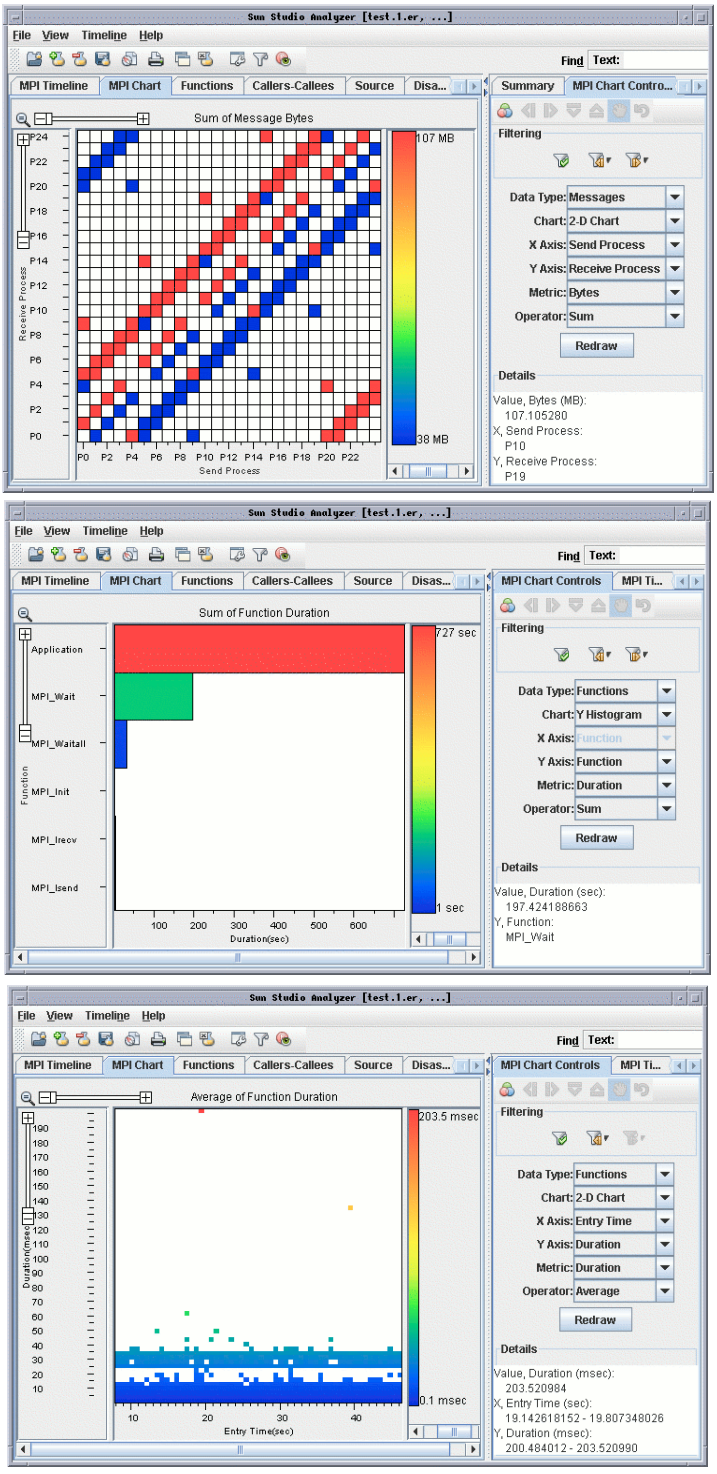

Рисунок 3 — Performance Analyzer. Графики MPI.

Графики MPI (вкладка **MPI Charts**, рис. 3) используются для визуализации диаграмм рассеяния и гистограмм , относящихся к параметрам работы функций MPI (длительность, количество вызовов и т.д.). Графики MPI могут использоваться для идентификации шаблонов взаимодействия процессов, нахождения нарушений балансировки нагрузки и разного рода аномалий в поведении параллельной программы. Типы визуализируемых данных, используемые метрики и параметры отображения могут настраиваться пользователем в широких пределах.

Функции (вкладка **Functions**, рис. 4) может использоваться для поиска функций, потребляющих много вычислительных ресурсов (по той или иной метрике, выбираемой пользователем). Каждая строка таблицы 6 содержит имя функции и набор значений выбранных пользователем метрик для этой функции. Можно производить фильтрацию таблицы, а также выполнять сортировку данных по любому столбцу (т.е. по любой метрике). При выборе конкретной функции в таблице становятся доступны дополнительные детали, а также имеется возможность перейти к исходному коду функции или соответствующей части графа вызовов.

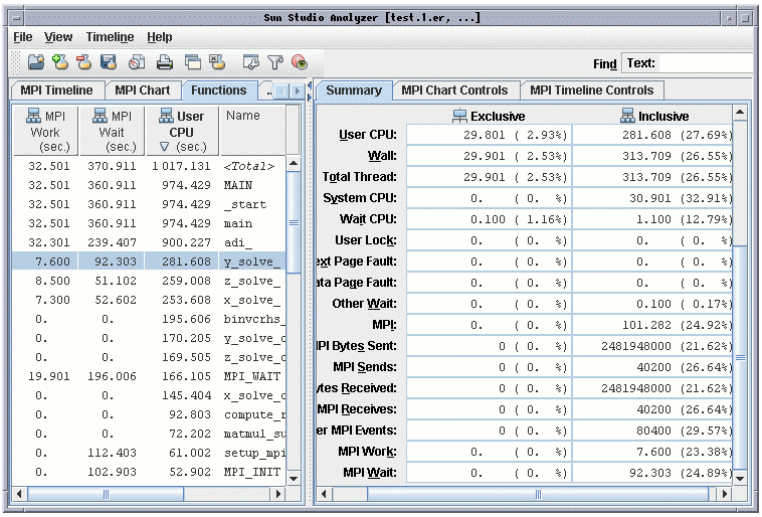

Рисунок 4 — Performance Analyzer. Функции.

Исходный код (вкладка **Source**, рис. 5) демонстрирует интересующие пользователя (например, относящиеся к некоторым конкретным функциям) фрагменты исходного кода параллельной программы, аннотированные значениями метрик производительности на уровне отдельных строк. Строки кода могут быть отсортированы как в естественном порядке (как в исходном коде), так и по любой из доступных метрик производительности.

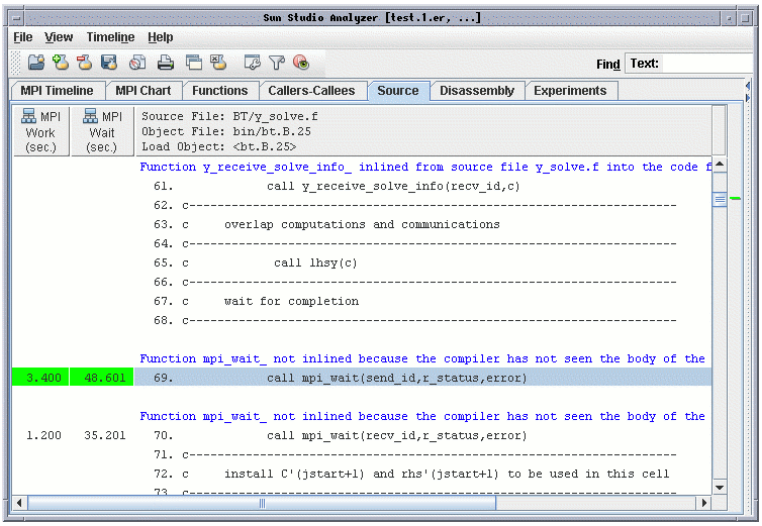

Рисунок 5 — Performance Analyzer. Исходный код.

Граф вызовов (вкладка **Callers-Callees**, рис. 6) используется для визуализации требуемого фрагмента графа вызовов параллельной программы, аннотированного значениями метрик производительности для каждой из приведенных функций. В центральной панели указывается рассматриваемая в текущий момент функция, в верхней панели — все функции, вызывающие рассматриваемую, а в нижней панели — все функции, вызываемые из рассматриваемой. Каждая из панелей содержит значения метрик для функций и списки могут быть отсортированы по любой из метрик. По графу вызовов можно перемещаться, выбирая функции для переноса в центральную панель (они при этом становятся текущими и визуализируемый фрагмент графа вызовов соответствующим образом меняется).

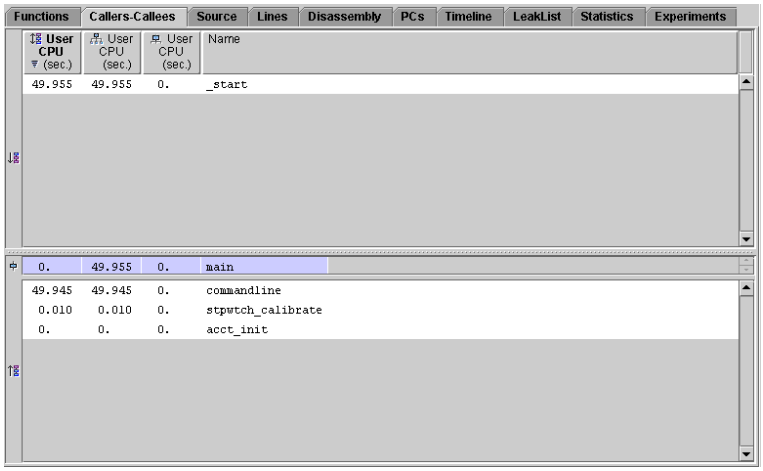

Рисунок 6 — Performance Analyzer. Граф вызовов.

Таблица 1 — Метрики Performance Analyzer.

Категория	Идентификатор	Полное название и описание
Классическое профилирование (процессорное время)	user	User CPU time - процессорное время в контексте пользователя
	wall	Wall-clock time - физическое время
	total	Total LWP time - время легковесных процессов (light-weight process)
	system	System CPU time - процессорное время в контексте системы
	wait	CPU wait time - процессорное время ожидания
	ulock	User lock time - время пользовательской блокировки
	text	Text-page fault time - потери времени на страничные ошибки для текста
	data	Data-page fault time - потери времени на страничные ошибки для данных
	owait	Other wait time - время ожидания по другим причинам
Метрики, связанные с синхронизацией	sync	Synchronization wait time - время синхронизационных задержек
	syncn	Synchronization wait count - количество синхронизационных задержек
Метрики, связанные с трассировкой MPI	mpitime	Time spent in MPI calls - время, затраченное на вызовы MPI
	mpisend	Number of MPI send operations - количество операций отправки сообщений MPI
	mpibytessent	Number of bytes sent in MPI send operations - количество байт, переданных функциями отправки сообщений MPI

	mpireceive	Number of MPI receive operations - количество операций приема сообщений MPI
	mpibytesrecv	Number of bytes received in MPI receive operations - количество байт, полученных функциями приема сообщений MPI
	mpiother	Number of calls to other MPI functions - количество вызовов других функций MPI
Метрики, связанные с выделением оперативной памяти	alloc	Number of allocations - количество выделений памяти (аллокаций)
	balloc	Bytes allocated - количество выделенных байт
	leak	Number of leaks - количество утечек памяти
	bleak	Bytes leaked - количество байт памяти, потерянных за счет утечек
Метрики, связанные с аппаратными счетчиками	cycles	CPU cycles - количество циклов процессора
	insts	Instructions issued - количество инструкций процессора
Метрики, связанные с анализом потоков	raccesses	Data race accesses - количество операций доступа в условиях гонки за данными (data race)
	deadlocks	Deadlocks - количество дедлоков

Графический режим представления информации

Графический режим представления информации, собранной Performance Analyzer, используется при запуске утилиты analyzer. Вызов утилиты осуществляется командой:

% analyzer [*список экспериментов*]

Например, для анализа результатов эксперимента test.1.er следует воспользоваться командой:

% analyzer test.1.er

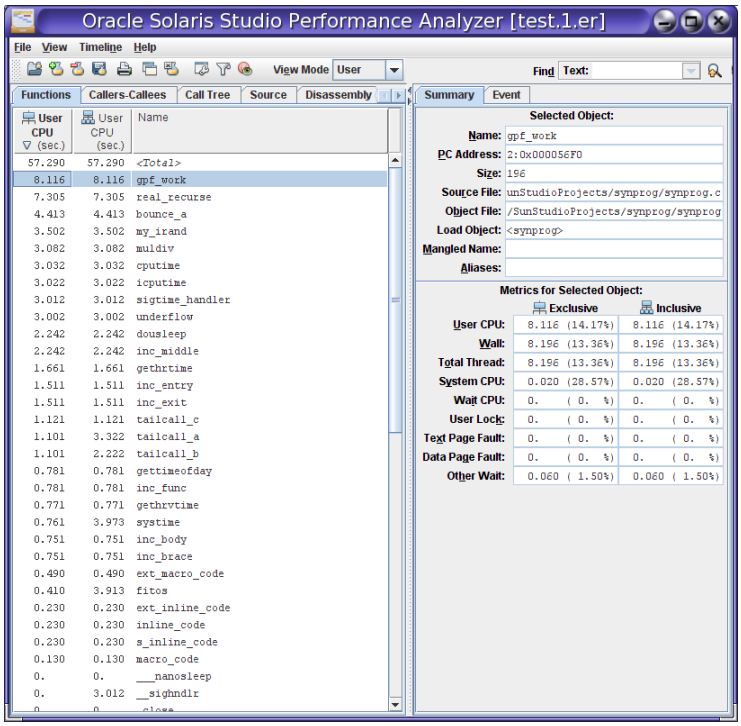

Рисунок 7 — Performance Analyzer. Графическое представление информации.

Консольный режим представления информации

Консольный режим представления информации, собранной Performance Analyzer, используется при запуске утилиты er_print. Вызов утилиты er_print осуществляется с помощью команды:

% er_print -*command список экспериментов*

Например, для вывода информации о функциях (названия и метрики) по эксперименту test.1.er, можно воспользоваться следующей командой:

```
%  er_print -functions test.1.er
/opt/solstudio12.2/bin/../prod/bin/sparcv9/er_print:
Processed /home/user/.er.rc for default settings
test.1.er:
Functions sorted by metric: Exclusive User CPU Time
```

Результаты будут выведены на консоль в следующем виде:

```
Excl.        Incl.        Name
User CPU  User CPU
 sec.      sec.
57.290    57.290    <Total>
 8.116     8.116    gpf_work
 7.305     7.305    real_recurse
 4.413     4.413    bounce_a
 3.502     3.502    my_irand
 3.082     3.082    muldiv
 3.032     3.032    cputime
 3.022     3.022    icputime
 3.012     3.012    sigtime_handler
 3.002     3.002    underflow
 2.242     2.242    dousleep
 2.242     2.242    inc_middle
 1.661     1.661    gethrtime
 1.511     1.511    inc_entry
 1.511     1.511    inc_exit
 1.121     1.121    tailcall_c
 1.101     3.322    tailcall_a
 1.101     2.222    tailcall_b
 0.781     0.781    gettimeofday
 0.781     0.781    inc_func
 0.771     0.771    gethrvtime
 0.761     3.973    systime
 0.751     0.751    inc_body
 0.751     0.751    inc_brace
 0.490     0.490    ext_macro_code
.
.lines deleted
```

Утилита er_print может также использоваться в интерактивном консольном режиме (команды вводятся пользователем во внутренней командной оболочке утилиты).

Методика использования Performance Analyzer

Performance Analyzer предоставляет богатые возможности детализированного анализа производительности параллельного кода и дальнейшей оптимизации кода с целью улучшения параметров производительности, включая масштабируемость. Общую схему работы можно представить как итеративный процесс, состоящий из следующих этапов:

1) Профилирование текущей сборки параллельного кода с помощью Performance Analyzer.

2) Детализированный анализ результатов профилирования с помощью Performance Analyzer.

3) Определение узких мест (bottlenecks) параллельного кода в смысле производительности и принятие решений о целесообразности и видах дальнейших оптимизаций кода.

4) Проведение запланированных оптимизаций кода и генерация новой сборки параллельного кода.

5) Переход к этапу № 1.

Следует отметить, что в пунктах №№ 1 и 2 имеется в виду обобщенное профилирование кода, включая замеры масштабируемости (ускорение, эффективность, стоимость параллельных вычислений — см. выше), оценки затрат процессорного времени (CPU time) и физического времени (Wall time), замеры потерь времени на синхронизацию процессов и т.д. Итеративный процесс прекращается, когда удовлетворены технические требования к производительности и масштабируемости параллельного кода, либо выполнение итеративного процесса не приводит к положительным результатам (в последнем случае может потребоваться масштабный рефакторинг исходного кода или даже пересмотр базовых алгоритмов работы кода).

Этап № 2 требует некоторых дополнительных пояснений. На этом этапе основной интерес представляет описанный выше по тексту инструмент Functions (Функции) Performance Analyzer, который выводит список функций исходного кода, аннотированный значениями различных метрик для каждой из функций. Список может сортироваться по любому столбцу (метрике), что позволяет легко выделить узкие в смысле производительности по выбранной метрике места исходного кода. Наибольший интерес представляют собой значения метрик Inclusive CPU Time (суммарное процессорное время, затраченное на работу функции, включая все вызываемые из нее функции) и Inclusive MPI Time (суммарное физическое / системное время, затраченное на обслуживание всех вызовов MPI из данной функции и всех вызываемых из нее функций). После отбора конкретных представляющих дальнейший функций на основании значений метрик из вкладки Functions можно воспользоваться инструментом Callers / Callees (Граф вызовов), например, для

детализированного исследования распределения значений метрик по функциям, вызываемым из рассматриваемой и вкладе рассматриваемой функции в метрики функций, ее вызывающих. Кроме того, для детального исследования распределения значений метрик на уровне отдельных строк проблемных функций может оказаться полезным инструмент Source (Исходный код). При его использовании становятся доступны отдельные строки исходного кода интересующих пользователя функций, аннотированные значениями метрик.

Для определения функций, неэффективно расходующих процессорное время (метрика CPU time) или физическое / системное время (метрика Wall time), либо неэффективно использующих средства MPI (метрика MPI time), использование инструментов Functions, Callers / Callees и Source программы Performance Analyzer представляется достаточным. Однако для определения проблем с синхронизацией процессов, нахождения аномалий в стандартных шаблонах параллельного взаимодействия, выявления влияния размера сообщений MPI на производительность могут потребоваться дополнительные инструменты Performance Analyzer, описанные выше по тексту: MPI Timeline (шкала времени MPI) и MPI Charts (графики MPI). В частности, при помощи инструмента MPI Timeline можно визуально, абстрагируясь от численных значений метрик, находить различные виды аномалий во взаимодействии параллельных процессов. Инструмент MPI Charts позволяет более детально анализировать выявленные аномалии поведения параллельного кода, предоставляя необходимую численную информацию (как непосредственно в численной, так и в графической форме — в виде графиков различных типов).

Служебный код сбора и сохранения данных профилирования Performance Analyzer является распределенным, работает в параллельном режиме и хорошо масштабируется как минимум в пределах нескольких тысяч вычислительных ядер. При этом за счет использования при сборе большинства метрик специальных неинтрузивных методик, не подвергающих профилируемый параллельный код дополнительному инструментированию (которое в некоторых случаях существенно искажает данные профилирования инструментированного кода по сравнению с первоначальным кодом), численные результаты профилирования по большинству метрик заслуживают высокой степени доверия, в отличие от целого рядка конкурирующих профилировщиков, использующих другие методики (принудительное инструментирование, исполнение кода в виртуальной среде, эмуляция работы и т.д.).

Этапы 4 и 5 выполняются при необходимости по результатам анализа этапа 3.

Заключение

Oracle Studio Performance Analyzer полностью удовлетворяет предъявляемым требованиям масштабируемости вычислительного эксперимента на суперкомпьютерах (не менее 1000 ядер и пиковая производительность не ниже 10 Тфлопс). Более того, служебный код сбора данных Performance Analyzer работает в параллельном режиме, хорошо масштабируется в указанном диапазоне и весьма слабо влияет на производительность профилируемого параллельного кода за счет использования специальных методов сбора данных профилирования, что выгодно отличает Performance Analyzer от ряда конкурирующих параллельных профилировщиков. Performance Analyzer полностью удовлетворяет всем необходимым техническим требованиям, являясь при этом общедоступным и бесплатным программным обеспечением.

В работе сделан краткий обзор возможностей, предоставляемых Performance Analyzer, и представлены базовые приемы работы с этой программой.