УДК 519.6

ПРОГРАММНЫЕ СРЕДСТВА STK ДЛЯ ИССЛЕДОВАНИЯ ЭФФЕКТИВНОСТИ ВЫПОЛНЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРИЛОЖЕНИЙ

Д. А. Новаев, Ю. Г. Бартенев, Д. И. Липов, С. И. Колпаков,
А. Б. Киселев, Т. Н. Серова, Л. В. Худякова
(РФЯЦ-ВНИИЭФ)

Описаны состав и возможности программных средств для сбора и представления данных по эффективности использования вычислительных ресурсов многопроцессорных ЭВМ задачами, выполняющимися в режиме распараллеливания.

Ключевые слова: многопроцессорная ЭВМ, эффективность выполнения *параллельных* задач, интерфейс межпроцессорных обменов (MPI), ввод-вывод, средства профилирования параллельных приложений.

Введение

Эффективное управление и использование больших многопроцессорных ЭВМ требует специализированного программного обеспечения, которое включает в себя средства мониторинга и анализа эффективности выполнения задач в режиме распараллеливания (*параллельных* задач) как непосредственно во время счета (online), так и после его завершения (offline). Для реализации данных средств необходимо измерение большого количества параметров, характеризующих эффективность использования аппаратных ресурсов параллельными задачами, таких как процессор, память и коммуникационная среда.

Современные средства анализа и отладки выполнения параллельных задач (программ) включают отладчики и средства профилирования (Vampir [1], MPE [2], Intel Trace and Analyzer Collector [3] и др.). Цель профилирования — обеспечить механизм, с помощью которого разработчики параллельных приложений могли бы анализировать поведение программ с целью улучшения их производительности посредством сбора информации о программе (трассировки) во время ее выполнения. Обычно для целей трассировки в исследуемую программу встраиваются *профилировочные вызовы*, которые фиксируют наступление определенных событий или продолжительность интервалов и записывают собираемую информацию в журнал трассировки. Затем полученная *трасса* просматривается и анализируется.

Следует отметить, что к недостаткам всех существующих на сегодня в мире средств анализа можно отнести, во-первых, отсутствие унифицированных форматов трасс — трассировщики обычно ориентированы на конкретную библиотеку передачи сообщений. Во-вторых, возможности настройки фильтров событий, с помощью которой задается, какие события и какую информацию следует или не следует включать в трассу (журнал характеристик программы при ее выполнении), достаточно слабы. Кроме того, средства трассировки достаточно сильно влияют на поведение программы, а также приводят к нежелательному увеличению накладных расходов при возрастающей потребности в вычислительных ресурсах.

Указанные недостатки не дают возможности использовать существующие средства профилирования в условиях долговременного счета задач с неограниченным числом MPI-вызовов на большом количестве процессоров (1 000 и более), а также в качестве штатных (постоянно работающих) систем сбора данных по эффективности счета параллельных задач и эффективности использования многопроцессорных ЭВМ. Поэтому в РФЯЦ-ВНИИЭФ разработаны собственные программные средства STK [4].

STK (Statistics Tool Kit) — это программные средства для исследования эффективности вы-

полнения параллельных задач, которые используют MPI (интерфейс передачи сообщений) [5, 6] и OpenMP (интерфейс многопоточного распараллеливания на вычислительных SMP-узлах с общей памятью) [7] на многопроцессорных ЭВМ под управлением операционной системы Linux [8, 9]. Прежде всего система STK разрабатывалась для мониторинга и анализа эффективности использования вычислительных ресурсов многопроцессорных ЭВМ программными комплексами с целью контроля над выполняемыми пользовательскими задачами и обеспечения более эффективного их счета. В то же время STK выполняет роль *специализированного профилировщика* параллельных задач, предоставляющего разработчикам параллельных программ средства для определения признаков и причин неэффективной работы как всей программы в целом, так и отдельных ее фрагментов. Все это помогает оптимизировать как программные комплексы, так и использование дорогостоящего оборудования, применяемого при таких расчетах.

В STK эффективность выполнения задачи основана на подсчете отношения времени полезных вычислений ко всему времени выполнения, а именно:

1) для каждого MPI-процесса параллельной задачи измеряются $T_{\text{обмена}}$ — время, затраченное на выполнение вызовов функций MPI и ввода-вывода (IO), а также $T_{\text{общее}}$ — общее время выполнения; затем вычисляются

$$T_a = T_{\text{общее}} - T_{\text{обмена}};$$
$$E_i = \frac{T_a}{T_a + T_{\text{обмена}}} \cdot 100\,\%,$$

где T_a считается временем арифметической работы, а E_i — показателем эффективности выполнения i-го MPI-процесса задачи;

2) для всей задачи вычисляется

$$E = \frac{\sum\limits_{i=1}^{N} T_a}{\sum\limits_{i=1}^{N} (T_a + T_{\text{обмена}})} \cdot 100\%,$$

где N — количество процессов; E считается показателем эффективности выполнения параллельной задачи (средней эффективностью выполнения MPI-процессов задачи).

Кроме того, среди всех E_i находятся $E_{\min}$ и $E_{\max}$, которые вместе с E дают представление о дисбалансе вычислений параллельной задачи. Близость E к $E_{\max}$ свидетельствует о слабом вычислительном дисбалансе (небольшое число MPI-процессов выполняют вычислительную работу меньше, чем остальные); близость E к $E_{\min}$ при $E \ll E_{\max}$ свидетельствует о сильном вычислительном дисбалансе (небольшое число MPI-процессов выполняет существенно большую вычислительную работу, чем остальные).

Отметим, что эти показатели эффективности STK вычисляет в каждый момент счета параллельной задачи.

Программные компоненты STK

В состав STK входят следующие компоненты (рис. 1):

- ядро STK для сбора данных об использовании процессами параллельной задачи вычислительных ресурсов;
- транспортная подсистема STK для доставки и объединения собираемой информации;
- пользовательские средства (API) для разработки собственных приложений обработки данных по эффективности выполнения параллельных задач;
- база данных (БД) STK для хранения информации;
- система многомерного анализа данных, реализованная с использованием OLAP-технологии;
- система графического представления данных.

Ядро STK. Предназначено для сбора данных об использовании многопоточными MPI-процессами вычислительных ресурсов. При реализации ядра STK было выбрано направление, при котором оно интегрируется в пользовательскую задачу.

Ядро STK представлено в виде двух библиотек: статической и динамической. В статической библиотеке собраны *функции-обертки* MPI-вызовов и операций ввода-вывода, в динамической — их обработчики. Применение динамической библиотеки ядра STK позволяет модифицировать и внедрить ядро STK в пользовательскую программу без ее перетрансляции. Это технологически очень полезно, так как новые возможности в STK автоматически становятся доступными для всех программ, собранных с динамической библиотекой ядра STK.

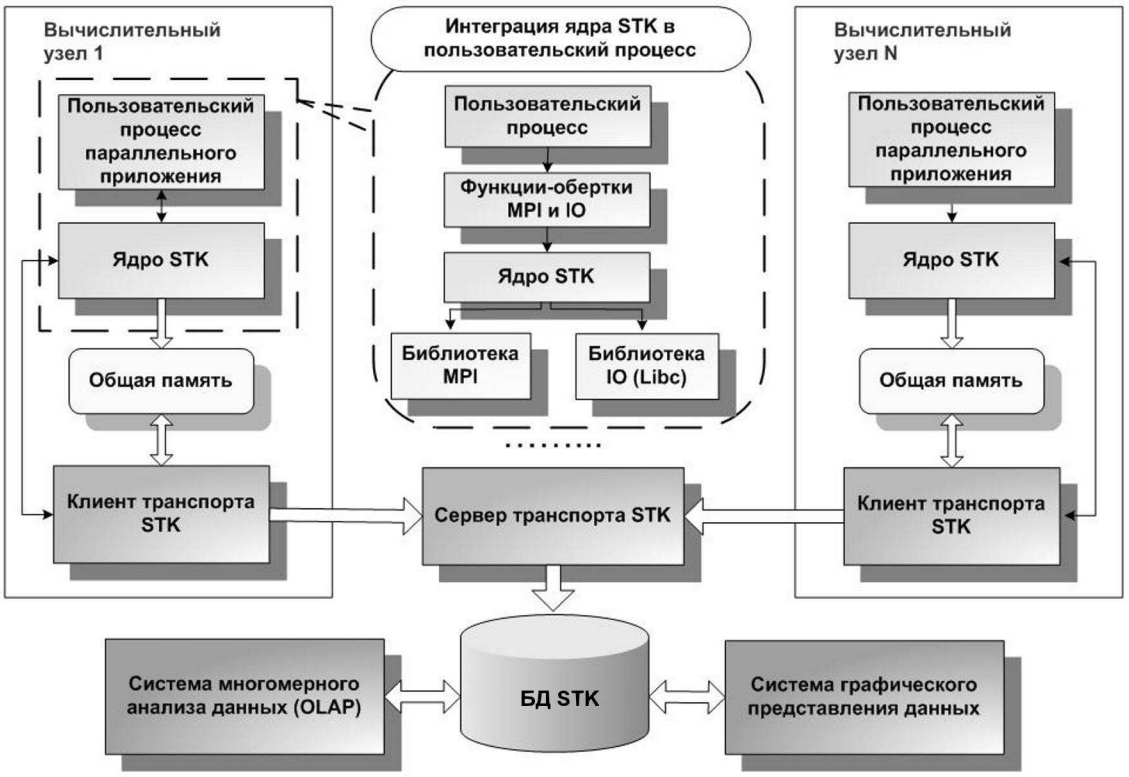

Рис. 1. Архитектура STK

Транспортная подсистема STK. Это распределенная система, предназначенная для получения от ядра STK данных по выполнению процессов параллельных программ, их транспортировки и хранения.

Транспортная подсистема разрабатывалась с той целью, чтобы реализовать не зависимый от системной конфигурации используемой многопроцессорной ЭВМ механизм получения и пересылки данных в течение всего времени выполнения процессов параллельной программы, обеспечив при этом минимальные накладные расходы.

Средства разработки пользователя (API). Написаны на языке C и реализованы в виде Фортран- и C-интерфейсов, которые позволяют использовать API-функции в программах, написанных на языках C/C++ и Фортран. API является расширением основных функциональных возможностей STK для разработчиков параллельных программ. Используя API-функции, разработчики параллельных программ могут получить в них доступ к любым данным, которые предоставляются средствами STK, а также сохранить эти данные в файлах для дальнейшего анализа.

БД STK. В STK реализован механизм хранения оперативной информации о работе параллельных программных комплексов в БД STK, что позволяет всем авторизованным пользователям получить доступ к единому банку данных, проводить различные выборки данных по всем полям БД, переложить часть вычислений на сервер БД. Также в БД STK поддерживается механизм записи информации в режиме реального времени. Это позволяет получать необходимую информацию в ходе выполнения параллельных задач.

В настоящее время в STK реализована возможность сохранения данных в собственной файловой БД, а также в широко используемых СУБД, таких как MySQL [10], Oracle [11], PostgreSQL [12]. Кроме того, для хранения данных были опробованы, а в дальнейшем применены кластерные СУБД, которые позволяют оптимизировать время отклика на этапе анализа и визуализации требуемой информации.

Система многомерного анализа данных. Разработка БД STK позволила применить OLAP-технологии [13] для визуализации данных.

OLAP (On-Line Analytical Processing) — это технология комплексного анализа многомерных данных. OLAP является оптимальным решением для большого класса приложений, где пользователи имеют дело с данными, зависящими от многих параметров.

Технология OLAP включает в себя следующие компоненты:

- OLAP-сервер, который получает и обрабатывает данные из БД с помощью соответствующего провайдера;
- OLAP-клиент (MS Excel) для визуализации данных, который позволяет получать двух- и трехмерные сечения гиперкуба данных, полученного с помощью OLAP-сервера.

Пользователь может построить по своему усмотрению итоговую диаграмму любого вида, а также включить ее в интерактивном виде в документ MS Word.

Пример использования технологии OLAP показан на рис. 2, который содержит объемную гистограмму, представляющую эффективность счета различных программных комплексов за определенный год. На графике можно видеть суммарные оценки времен на использование MPI, ввод-вывод, полезный счет для различных методик, показывающие, насколько данные методики эффективны. Также можно построить любую диаграмму как по всем задачам, рассчитываемым по определенной методике, так и по задачам конкретного исполнителя.

Система графического представления данных (рис. 3). Реализована на языке Java и поэтому может быть легко интегрирована как приложение в браузер, доступный в операционной системе пользователя.

Система имеет удобный пользовательский интерфейс, позволяет сократить время обработки большого объема информации, а также может функционировать в различных операционных системах и на различных платформах.

Данные могут быть представлены для конкретной задачи, выбранной пользователем, в виде таблиц и графиков (рис. 4, 5) как после завершения задачи, так и во время ее выполнения.

В системе имеется возможность сохранения таблиц в Excel-формате, а графиков — в графическом формате PNG. Для получения различных видов статистики пользователь может задать дополнительные настройки для фильтрации данных, просмотра статистики по задачам за определенный промежуток времени. Также можно сохранить собственные настройки.

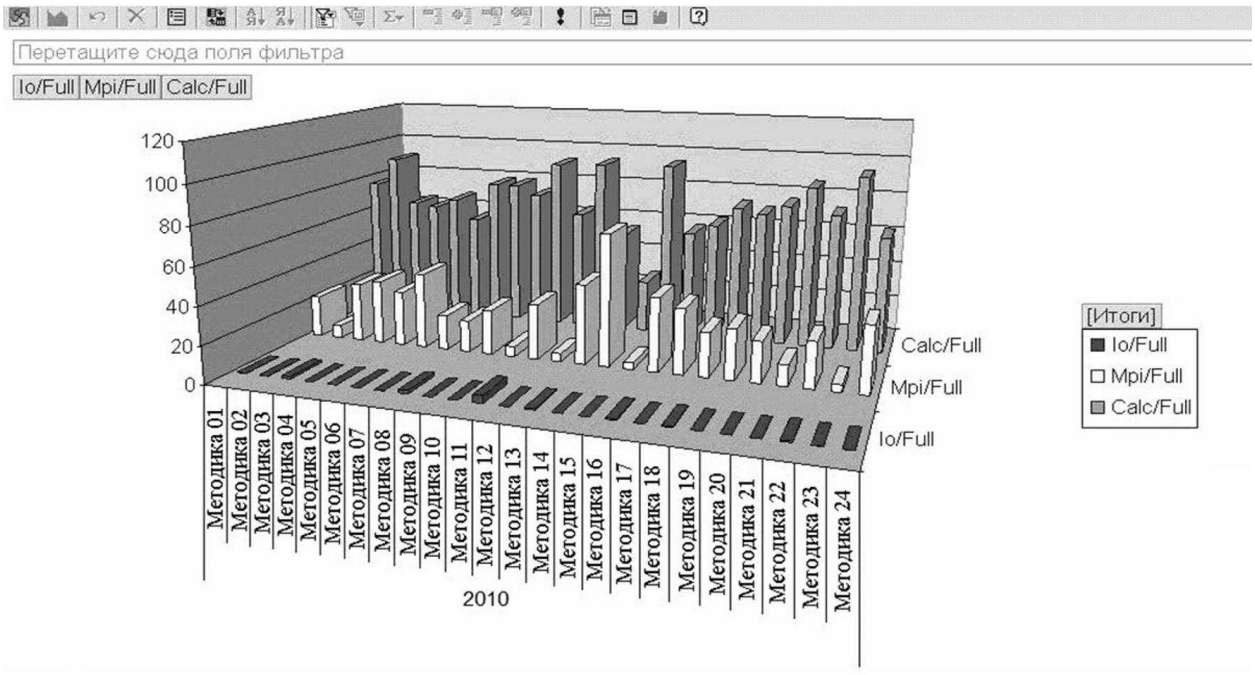

Рис. 2. Пример использования OLAP-технологии

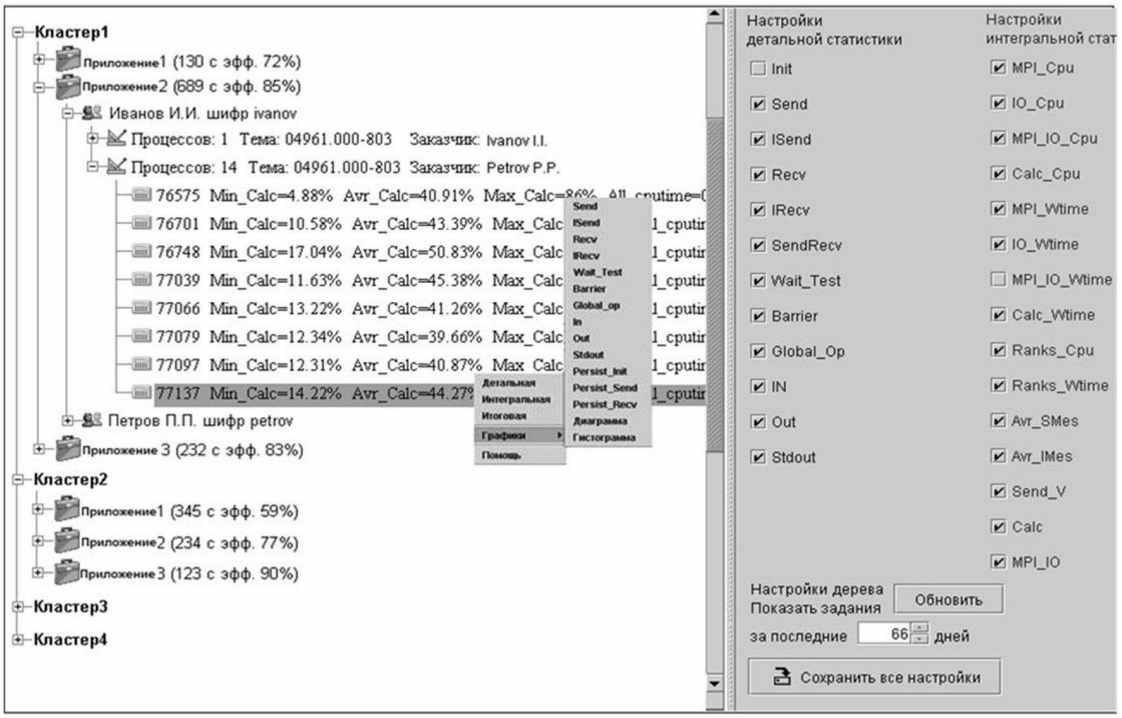

Рис. 3. Главная форма системы графического представления данных

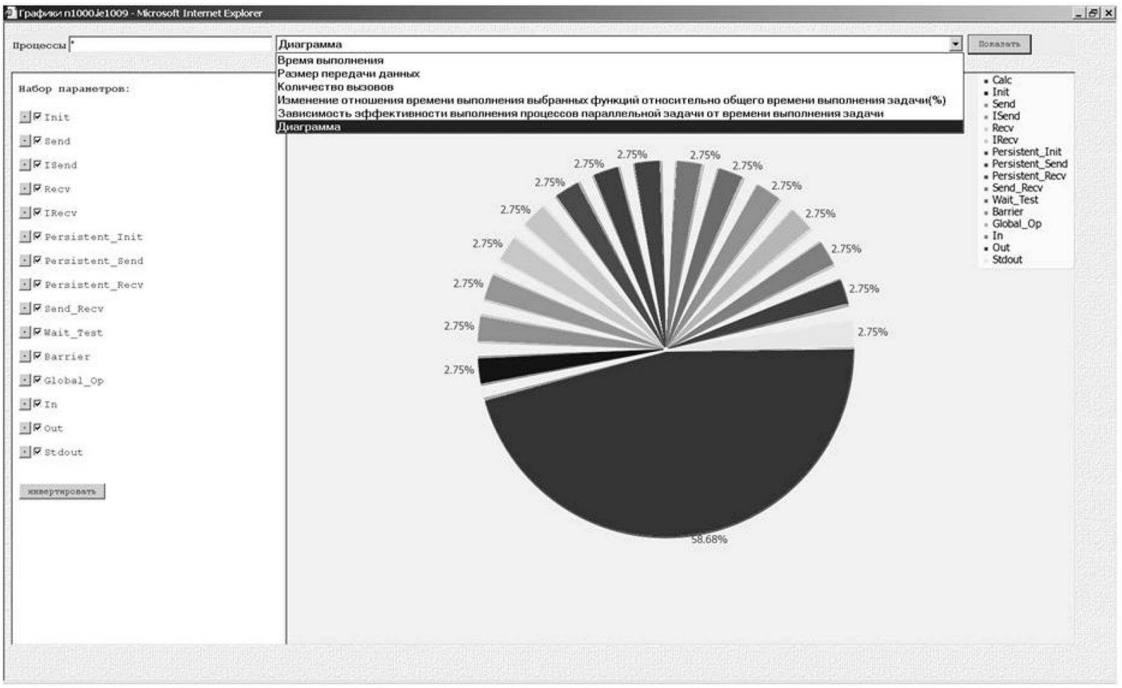

Рис. 4. График долевого распределения времен на выполнение полезных вычислений, групп MPI-функций и операций ввода-вывода относительно календарного времени счета задачи

Важным свойством системы является возможность представления в интегральной, структурированной форме (рис. 6) коммуникационной нагрузки (частота двухточечных и глобальных обменов и объем передаваемой информации) при выполнении параллельных задач. Это позволяет получить экспериментальные данные по счету реальных задач, необходимые для создания усо-

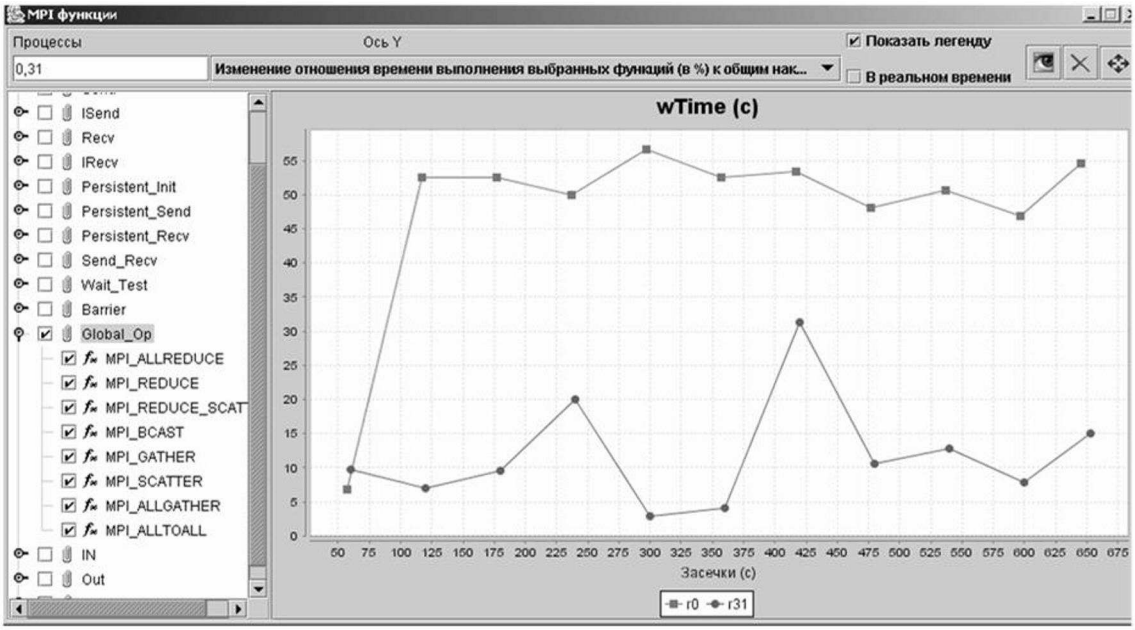

Рис. 5. График изменения отношения времени выполнения коллективных операций обменов MPI относительно общего времени выполнения для MPI-процессов

Расширенная интегральная статистика

	N_RANK	POINT2POINT_RATE	COLLECTIVE_RATE	ALL_RATE	P2P_SENDBYTE	COLLECTIVE_SENDBYTE	WAIT_TEST_RATE	BARRIER_RATE	BCAST_RATE	SCAT
1	0	252,56	65,26	523,35	20244,685	2758,420	205,53	13,32	5,705	0,0
2	1	492,84	162,30	1052,85	17283,092	2973,327	397,70	35,22	15,082	0,0
3	2	282,92	59,65	548,83	18731,666					
4	3	718,06	150,85	1425,44	15992,507					
5	4	279,03	58,62	540,41	18618,390					
6	5	690,03	144,45	1367,41	15885,922					
7	6	160,75	33,86	311,78	21112,386					
8	7	688,56	144,65	1366,87	15992,507					
9	8	85,35	28,57	179,55	22433,704					
10	9	482,60	160,56	1036,60	16062,016					
11	10	197,17	65,60	414,19	18717,151					
12	11	534,02	175,86	1140,81	15548,393					
13	12	277,99	58,60	539,26	17353,137					
14	13	694,51	145,90	1378,68	14625,041					
15	14	282,21	59,28	546,57	17277,338					
16	15	694,14	145,31	1375,56	14527,896					
17	16	159,95	33,70	310,24	19751,806					
18	17	718,74	150,99	1426,78	14625,041					
19	18	85,70	28,56	179,45	20828,621					
20	19	489,35	162,80	1051,09	14385,462					

Расширенная итоговая статистика

ПАРАМЕТР	MIN	AVG	MAX
POINT2POINT_RATE	85,35	272,25	718,74
COLLECTIVE_RATE	28,56	66,89	175,86
BARRIER_RATE	5,75	14,72	38,17
WAIT_TEST_RATE	65,19	208,18	557,04
ALL_RATE	179,45	547,32	1426,78
P2P_SENDBYTE	14385,46	17463,39	22433,70
COLLECTIVE_SENDBYTE	24,91	1167,55	2973,33
BCAST_RATE	2,87	6,41	16,34
SCATTER_RATE	0,00	0,00	0,00
GATHER_RATE	0,00	0,00	0,01
ALLGATHER_RATE	2,04	5,22	13,54
ALL2ALL_RATE	0,00	0,00	0,00
ALLREDUCE_RATE	13,79	29,99	80,47
REDUCE_RATE	4,12	10,54	27,33
REDUCE_SCATTER_RATE	0,00	0,00	0,00
BCAST_SENDBYTE	0,00	136,18	1220,47
SCATTER_SENDBYTE	0,00	0,00	0,00
GATHER_SENDBYTE	656,00	1221,82	3768,00
ALLGATHER_SENDBYTE	9,60	10761,60	26889,60
ALL2ALL_SENDBYTE	0,00	0,00	0,00
ALLREDUCE_SENDBYTE	11,31	97,95	433,20
REDUCE_SENDBYTE	83,35	83,35	83,35

Рис. 6. Структура коммуникационной нагрузки, выдаваемой параллельной задачей

вершенствованных параллельных программ и многопроцессорных ЭВМ следующего поколения.

Возможности STK

Собираемые характеристики выполнения параллельных задач. В STK по умолчанию собираются данные по 17 группам функций, используемых в параллельных программных комплексах: инициализация и завершение среды выполнения MPI, блокирующие передача или прием, неблокирующие передача или прием, операции передачи-приема, синхронизация процессов при выполнении неблокирующих обменов, барьерная синхронизация, коллективные операции, создание возобновляемых коммуникационных запросов, возобновляемые запросы на передачу или прием, чтение данных из файла, запись данных в файл, передача данных на стан-

дартный вывод, работа с метаданными файлов, открытие или закрытие файлов. Существует возможность детализировать эти группы, что позволяет предоставлять детальную информацию по выполнению конкретных функций из групп.

Исходя из 129 параметров, собираемых по каждому процессу параллельной задачи, вырабатываются интегральные характеристики выполнения параллельных задач:

1) время арифметической работы MPI-процессов задачи, т. е. время полезных вычислений без включения времени обменов;
2) время ожидания окончания обменов между процессами задачи;
3) объемы данных, передаваемых между процессами задачи;
4) объемы данных, записанных во время файловых операций;
5) объем информации на стандартном потоке вывода задачи;
6) время простоев задачи из-за обменов с файлами;
7) темпы вызовов операций MPI или ввода-вывода;
8) процент временных затрат на выполнение полезных вычислений относительно общего времени выполнения, т. е. показатель эффективности выполнения параллельной задачи.

В отличие от средств профилирования параллельных приложений STK накапливает не историю вызовов MPI-функций, а интенсивность вызовов различных типов MPI-функций и их основные параметры. Это позволяет использовать небольшой объем памяти, накапливать и собирать данные нужного типа в течение долговременного счета задачи с неограниченным числом MPI-вызовов.

Режимы сбора данных. Работа STK состоит из накопления данных по выполнению каждого MPI-процесса в оперативной памяти вычислительного узла, не доступной пользовательской программе, и сбора этих данных на внешнем носителе (в БД), обеспечивающем их сохранение в ходе счета задач.

Сбор данных может происходить в двух режимах: производственном и отладочном. Производственный режим (режим по умолчанию) осуществляет сбор накопленных STK данных через каждые 30 минут и по окончании выполнения задачи, что обеспечивает минимальные затраты как времени на обработку, так и ресурсов для хранения информации. Отладочный режим осуществляет сбор и предоставление данных в ходе выполнения задачи через определенный пользователем временной интервал.

Управляет режимами сам пользователь при помощи переменной окружения *STK_MODE*. Можно задать следующие ее значения: 0 — отключить сбор статистики, 1 — установить производственный режим, 2 — установить отладочный режим.

Диагностика случаев "зависания" программ при выполнении функций MPI. STK обладает функцией диагностики *зависания* (тупика в процессе выполнения) MPI-процессов задачи из-за невозможности завершения MPI-функций, предназначенных для синхронизации или блокирующего приема, таких как MPI_Barrier, MPI_Wait, MPI_Recv и др. Распечатка функций MPI, приведших к тупику, позволяет пользователю найти ошибку в своей программе.

Анализ эффективности выполнения участков кода параллельных задач. STK поддерживает возможность сбора данных по выполнению отдельных участков кода параллельных задач. Это позволяет найти и устранить "узкие" места в параллельных программах. Участки кода параллельной задачи формируются пользователем при помощи двух API-функций: stk_codepart_start и stk_codepart_stop, задающих начало и конец участка кода параллельной задачи соответственно. Затем в системе визуализации можно увидеть статистику по эффективности выполнения отмеченных участков кода параллельной задачи.

Накладные расходы и масштабируемость STK. STK реализована как масштабируемая система. В зависимости от количества вычислительных узлов многопроцессорной ЭВМ и используемого транспортного протокола сбора параметров администратор может сконфигурировать систему таким образом, чтобы соотношение количеств серверов сбора статистики и обслуживаемых каждым сервером вычислительных узлов было оптимальным.

Алгоритмы STK, связанные со сбором и хранением данных, имеют низкие накладные расходы в условиях использования задачами большого числа (1 000 и более) процессоров. Это позволяет использовать STK в качестве *штатной* (постоянно работающей) системы сбора данных о нагрузке на коммуникационную сеть и систему ввода-вывода, об эффективности счета параллельных задач и эффективности использования параллельных ЭВМ [14].

Накладные расходы STK в ВЦ РФЯЦ-ВНИИЭФ не превышают 1 % от времени выполнения параллельной задачи, а в среднем они гораздо меньше и составляют $\sim$ 1 мкс на обработку одного MPI-вызова. Оценка накладных расходов получена на широком спектре реальных параллельных задач с высоким темпом MPI-обменов (до 10 000 обменов в секунду).

Применение STK

Для использования STK пользователю не нужно модифицировать свои исходные тексты программ. Вместо стандартных команд для компиляции и сборки параллельных программ (mpicc, mpiCC, mpif77, mpif90) достаточно указать команды mpicc_stk, mpiCC_stk, mpif77_stk, mpif90_stk.

Используя STK, пользователь может получить основные показатели эффективности выполнения своих параллельных программ (рис. 7) — доли от общего времени выполнения:

- затрат на коммуникационные обмены (MPI);
- затрат на ввод-вывод;
- полезных вычислений.

Если пренебречь возрастанием времени вычислений на многопроцессорных узлах с общей памятью (выполнение ряда параллельных вычислительных алгоритмов существенно замедляется из-за снижения скорости обмена нескольких процессоров с памятью) и затратами на подготовку данных для обмена в программе пользователя (они обычно крайне малы), то эффективность выполнения параллельной задачи, вычисленная STK, как показала практика, с хорошей точностью совпадает (STK ее немного завышает) с эффективностью распараллеливания, измеренной самим пользователем.

Таким образом, эффективность *по STK* с достаточной точностью позволяет оценивать эффективность распараллеливания задачи.

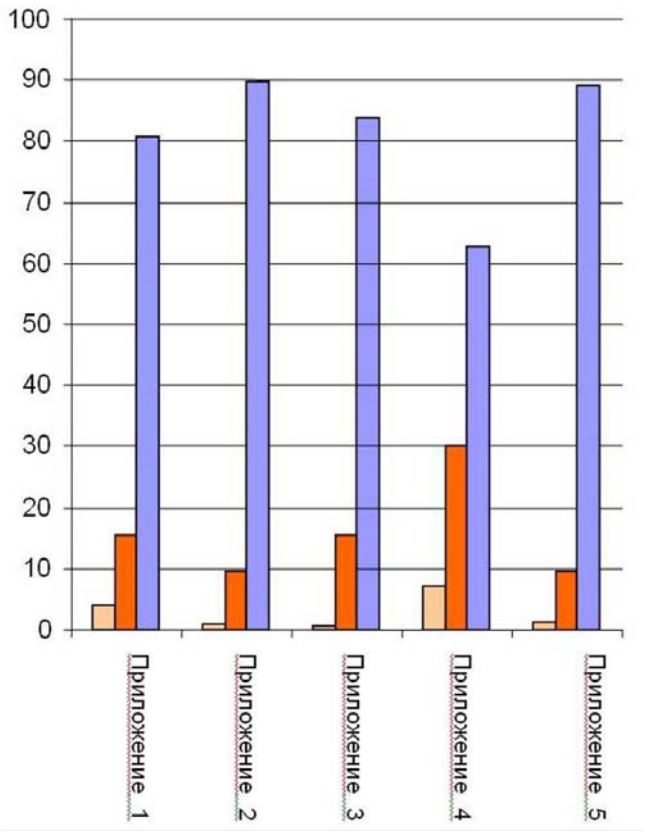

Рис. 7. Показатели эффективности выполнения параллельных задач: 1-й столбец — ввод-вывод; 2-й столбец — коммуникационные обмены (MPI); 3-й столбец — вычисления

Затраты на MPI и ввод-вывод данных составляют основные накладные расходы при выполнении параллельных задач. Одной из основных задач разработчиков параллельных программных комплексов является снижение этих накладных расходов с увеличением при этом доли полезных вычислений параллельной программы.

STK позволяет определить интегральные показатели эффективности счета параллельных программных комплексов на различных аппаратно-программных вычислительных системах (рис. 8). Это помогает выбрать оптимальную вычислительную систему для эффективного счета таких программных комплексов.

STK дает возможность наблюдать за эффективностью выполнения параллельных задач в процессе счета (рис. 9). Пользователи и службы управления производственным счетом могут своевременно завершить выполнение параллельной задачи с нехарактерно низким показателем эффективности счета (например, обусловленно-

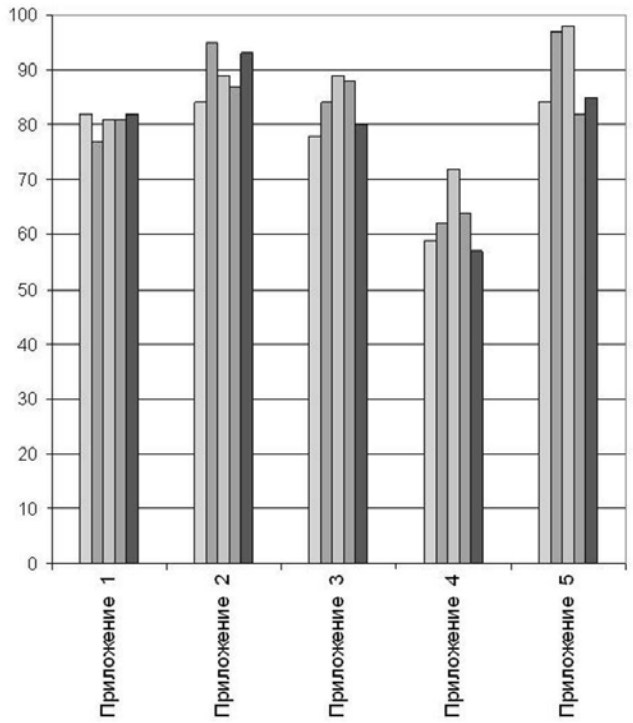

Рис. 8. Показатели эффективности выполнения параллельных задач на различных вычислительных системах (номер столбца диаграммы соответствует номеру вычислительной системы)

го неоптимально выбранной декомпозицией задачи), препятствуя неэффективному использованию вычислительных ресурсов.

STK позволяет удовлетворить требованиям широкого круга пользователей, таких как администраторы, аналитики, разработчики параллельных программ, диспетчерские службы управления производственным счетом, и организовать на этой основе контроль производственного счета потока задач.

Заключение

Система STK прошла длительный путь верификации, модернизации и применения на различных аппаратно-программных платформах.

Использование STK во всех параллельных задачах и на всех вычислительных комплексах ВЦ РФЯЦ-ВНИИЭФ является обязательным, что помогает при отладке параллельных программ и позволяет в реальном времени наблюдать за эффективностью выполнения всех параллельных задач и эффективностью использования всех многопроцессорных ЭВМ.

В ходе эксплуатации STK собрано большое количество информации, которая помогла оптимизировать как счетные программные комплексы, так и использование дорогостоящего оборудования при проведении расчетов по этим комплексам.

Использование STK позволило определить требования различных параллельных программных комплексов к оборудованию и программному обеспечению, что помогло для каждого из них выбрать оптимальную вычислительную систему. Кроме того, эти требования были учтены при проектировании последующих многопроцессорных вычислительных систем.

Список литературы

1. Vampir — Performance Optimization. http://www.vampir.eu.

Режим отображения: все задания

	Статус	ID сессии	Исполнитель	Проц-ов	Исп.врем	minЭфф	avgЭфф	maxЭфф
1	▶	377383	[illegible]	288	249:44:24	6,37	22,78	98,17
2	▶	382744	[illegible]	320	58:03:48	68,68	74,51	81,53
3	▶	382808	[illegible]	320	57:12:45	72	77,15	83,85
4	▶	383407	[illegible]	40	29:38:30	1,96	77,32	87,45
5	▶	383388	[illegible]	40	23:20:37	0,26	84,01	89,3
6	▶	383235	[illegible]	240	15:28:13	79,46	85,13	89,57

Рис. 9. Мониторинг эффективности выполнения параллельных задач во время счета

2. MPE — MultiProcessing Environment. http://www.mcs.anl.gov/research/projects/mpi/www/www4/MPE.html.
3. Intel Trace and Analyzer Collector. http://soft-ware.intel.com/ru-ru/intel-trace-analyzer.
4. *Новаев Д. А., Бартенев Ю. Г., Варгин А. М. и др.* Инструментальные средства исследования эффективности параллельных приложений — STK // Труды РФЯЦ-ВНИИЭФ. 2007. Вып. 11. С. 92—99.
5. *Snir M., Otto S., Huss-Lederman S. et al.* MPI: The Complete Reference. MIT Press, 1996.
6. MPICH 2: High Performance and Widely Portable Implementation of the Message Passing Interface (MPI) Standard. http://www-unix.mcs.anl.gov/mpi/mpich.
7. The OpenMP API Specification for Parallel Programming. http://openmp.org.
8. *Керниган Б., Пайк П.* UNIX — универсальная среда программирования. М.: Финансы и статистика, 1992.
9. *Робачевский А. М.* Операционная система Unix. С-Пб.: БХВ—Петербург, 2003.
10. MySQL. http://www.mysql.com.
11. Oracle. http://www.oracle.com/ru/products/database.
12. PostgreSQL: The World's Most Advanced Open Source Database. http://postgresql.org.
13. OLAP: Business Intelligence-Effective Data Mining & Analysis. http://www.olap.ru.
14. *Бартенев Ю. Г., Близнюк Г. Г., Логвин Ю. В., Шатохина Ю. В.* Интегральные показатели оценки работы многопроцессорных вычислительных систем // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2010. Вып. 4. С. 44—51.

Статья поступила в редакцию 05.04.11.