

АЛГОРИТМИЧЕСКИЙ И ПРОГРАММНЫЙ ИНСТРУМЕНТАРИЙ ДЕЛЬТА-ОПТИМИЗАЦИИ КОНТРОЛЬНЫХ ТОЧЕК ВОССТАНОВЛЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

А. Ю. Поляков, О. В. Молдованова*

Институт физики полупроводников им. А. В. Ржанова СО РАН,
630090, Новосибирск, Россия

* Сибирский государственный университет телекоммуникаций и информатики,
630102, Новосибирск, Россия

УДК 004.451.24

Предложены алгоритмы оптимизации контрольных точек восстановления параллельных программ по времени их создания и объему. В основе созданных алгоритмов лежит технология дельта-сжатия. В частности, разработан адаптивный алгоритм ADCA дельта-сжатия контрольных точек, обеспечивающий субоптимальный выбор между инкрементным и дифференциальным дельта-сжатием. Создан алгоритм комбинированного сжатия PaComp, позволяющий сочетать преимущества дельта-сжатия и алгоритмов, применяемых в программах-архиваторах. Путем экспериментального моделирования на мультикластерной вычислительной системе показана эффективность предложенных алгоритмов для набора прикладных параллельных программ.

Ключевые слова: распределенные вычислительные системы, контрольные точки восстановления, отказоустойчивость.

In this work new algorithms for checkpoint size and creation time optimization are proposed. Developed algorithms are based on delta-compression technology. In particular, new adaptive algorithm ADCA that provides suboptimal selection between incremental and differential delta-compression is developed. Also combined compression algorithm called PaComp is proposed. PaComp combines advantages of delta-compression and algorithms that are used in archivers. An experimental simulation that shows effectiveness of proposed algorithms for a set of parallel applications is provided.

Key words: distributed computer systems, checkpointing, fault tolerance.

Введение. Распределенные вычислительные системы (ВС) являются важнейшим инструментом решения сложных научных, инженерных и экономических задач [1]. Современные ВС являются крупномасштабными, количество процессорных ядер в их составе варьируется от десятков до сотен тысяч, а число узлов ввода-вывода (УВВ) — от нескольких десятков до сотен. Например, ВС К computer состоит из 705 024 процессорных ядер и 5508 УВВ, а система Cray XT5 Jaguar — из 224 162 процессорных ядер и 256 УВВ. Обычно несколько процессорных ядер физически размещаются на одном вычислительном узле (ВУ).

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (коды проектов 11-07-12014, 11-07-00109, 11-07-00105, 10-07-00157, 09-07-00185), Совета по грантам Президента РФ по государственной поддержке ведущих научных школ РФ (грант № НШ 5176.2010.9) и в рамках государственного контракта с Министерством образования и науки РФ № 07.514.11.4015.

Количество трудоемких задач, решаемых на крупномасштабных ВС, постоянно увеличивается. При этом для реализации некоторых из них требуются миллионы часов работы процессора. Статистические данные [1] показывают, что среднее время λ^{-1} безотказной работы вычислительных узлов распределенных ВС находится в интервале $10^4 \div 10^6$ ч (λ — интенсивность потока отказов для одного ВУ). Однако среднее время μ^{-1} между частичными отказами в ВС, т. е. между отказами одного или нескольких ВУ, составляет несколько дней [2, 3]. Таким образом, с увеличением количества задействованных ресурсов ВС и продолжительности их использования вероятность потери результатов вычислений, полученных параллельной программой (ПП), возрастает.

Важным элементом комплекса мер по обеспечению отказоустойчивости распределенных ВС является организация отказоустойчивого выполнения ПП. Данный подход предусматривает сохранение промежуточных состояний ПП в контрольных точках (КТ). В случае отказа ресурсов ВС любая доступная КТ позволяет перезапустить (возобновить) исходную программу в состоянии, которое не требует повтора вычислений, завершенных к моменту формирования данной КТ.

Контрольная точка, созданная для ПП, является распределенной и представляет собой набор локальных КТ, хранящих состояния процессов этой программы. Распределенная КТ называется целостной, если она позволяет сформировать допустимое состояние ПП [4].

Существует два основных подхода к созданию распределенных КТ [4]: синхронный (координированный) и асинхронный. При синхронном подходе все процессы ПП сохраняют свое состояние одновременно, при этом процесс создания распределенной КТ разбит на несколько этапов, между которыми происходит глобальная синхронизация. Это позволяет обеспечить целостность КТ, однако приводит к формированию высокой нагрузки на подсистему ввода-вывода распределенной ВС. При асинхронном подходе каждый процесс создает КТ независимо, что позволяет обеспечить равномерную нагрузку на подсистему ввода-вывода. Недостатком данного подхода является то, что при возобновлении ПП требуется выполнять поиск ее целостного состояния на основе набора локальных КТ, созданных в различные моменты времени. При этом возможно возникновение “эффекта домино”, когда наиболее поздним целостным состоянием программы оказывается ее начальное состояние.

Анализ существующих средств отказоустойчивого выполнения ПП показывает, что в большинстве из них для создания распределенных КТ используется синхронный подход [5, 6]. Таким образом, актуальной является задача снижения указанных выше затрат, обусловленных нагрузкой на подсистему ввода-вывода ВС. В настоящей работе предложен подход к решению данной задачи, предусматривающий уменьшение объема КТ и как следствие снижение времени их создания.

1. Дельта-оптимизация контрольных точек. Методы и алгоритмы, реализуемые в ПП, характеризуются различными шаблонами использования памяти. Например, в процессе работы многие итерационные методы лишь частично модифицируют обрабатываемые данные. Очевидно, что набор $\Lambda = \{\Lambda_1, \Lambda_2, \dots, \Lambda_T\}$ контрольных точек, созданных для таких программ в моменты времени $t = \{1, 2, \dots, T\}$, содержит дублирующуюся информацию. Данное свойство позволяет производить дельта-оптимизацию КТ набора Λ с использованием технологии дельта-сжатия [7–9]. Эта технология предполагает создание сжатой КТ $\Sigma_t = \Delta(\Lambda_t, \Lambda_b)$, содержащей только те фрагменты КТ Λ_t , которые были модифицированы относительно Λ_b ($b < t$, $t, b \in \{1, 2, \dots, \infty\}$; $\Delta(X_1, X_2)$ — функция, выполняющая дельта-сжатие структуры данных X_1 относительно X_2). Контрольная точка Λ_t может быть восстановлена с использованием обратной функции Δ^{-1} : $\Lambda_t = \Delta^{-1}(\Sigma_t, \Lambda_b)$.

Выбор КТ Λ_b , относительно которой выполняется дельта-оптимизация, в значительной мере определяет эффективность дельта-сжатия. Наиболее распространенным видом дельта-сжатия КТ является инкрементное, предполагающее сжатие любой КТ относительно предыдущей, т. е. $b = (t - 1) \forall t$. В современных системах резервного копирования используется также дифференциальное дельта-сжатие, которое предусматривает сжатие любой КТ относительно некоторой базовой КТ (обычно первой: $b = 1$).

Каждый из описанных видов дельта-оптимизации имеет преимущества и недостатки. При дифференциальном дельта-сжатии для формирования результирующей КТ Λ_t необходима базовая КТ Λ_1 и только одна дельта-сжатая КТ Σ_t . Это позволяет уменьшать объемы хранящихся данных за счет удаления устаревших дельта-сжатых КТ. В то же время для дифференциальных КТ характерно постоянное увеличение их объема. Инкрементные КТ, в отличие от дифференциальных, формируются относительно более актуального состояния программы и в большинстве случаев имеют меньший объем. Недостаток данного подхода заключается в необходимости хранить все созданные КТ:

$$\Lambda_t = \Delta^{-1}(\Sigma_t, \Delta^{-1}(\Sigma_{t-1}, \Delta^{-1}(\dots, \Delta^{-1}(\Sigma_2, \Lambda_1), \dots))).$$

Очевидно, что рассмотренные виды дельта-сжатия являются предельными случаями на множестве допустимых решений. Поэтому актуален поиск промежуточных вариантов, сочетающих преимущества инкрементного и дифференциального дельта-сжатий.

В данной работе в качестве функции Δ рассматривался алгоритм дельта-сжатия, который в литературе имеет различные названия: FsCH (fixed-size compare-by-hash) [7], FIC (fixed incremental checkpoint technique) [10] и FHash (fixed hash size algorithm) [11]. Недостатком данного алгоритма является его чувствительность к вставке и удалению фрагментов данных. Существуют алгоритмы, способные обнаруживать все виды изменений одной КТ относительно другой, но они обладают квадратичной вычислительной сложностью относительно размера КТ [11], в то время как вычислительная сложность алгоритма FHash линейна.

2. Адаптивный алгоритм дельта-оптимизации контрольных точек. В данной работе предложен адаптивный алгоритм (adaptive delta-compression algorithm (ADCA)) дельта-оптимизации КТ Λ_t , который осуществляет субоптимальный выбор базовой КТ Λ_b с целью минимизации объема $\Sigma_t = \Delta(\Lambda_t, \Lambda_b)$, а также уменьшения количества КТ, необходимых для формирования результирующей КТ. Алгоритм предусматривает дельта-сжатие КТ относительно некоторой базовой КТ Λ_b (изначально $b = 1$). Однако в данном случае, в отличие от случая дифференциального дельта-сжатия, базовая КТ не является фиксированной в процессе формирования набора Λ . Если различия объемов данных, модифицированных относительно базовой КТ Λ_b и предыдущей КТ Λ_{t-1} , превышают некоторое пороговое значение B :

$$\frac{S(\Delta(\Sigma_t, \Sigma_b))}{S(\Delta(\Sigma_t, \Sigma_{t-1}))} > B$$

($S(X)$ — размер структуры данных X , байт), то базовая КТ изменяется на текущую: $b = t$.

В результате работы алгоритма формируется дельта-сжатая КТ

$$\Sigma_t = \langle V_t, p_t, d, R_t, x_t, b_t \rangle.$$

Здесь b_t — номер базовой КТ; V_t — набор модифицированных блоков данных; p_t — частичная сигнатура, содержащая информацию о хеш-значениях, вычисленных для блоков в V_t ;

d — размер блока в V_t ; R_t — общее количество блоков в КТ Λ_t ; x_t — размер последнего блока КТ Λ_t .

3. Алгоритм пакетного сжатия контрольных точек. Распространенным подходом к оптимизации КТ по объему является их сжатие универсальными алгоритмами [6–9], например одной из модификаций алгоритма Зива — Лемпеля. Тот же подход может быть применен и к блокам данных, обнаруженным в процессе дельта-сжатия. Далее такой тип оптимизации будем называть комбинированным сжатием. Сжатие всех модифицированных блоков как единого массива данных не является эффективным, поскольку при формировании исходной КТ (распаковке) требуется доступ к произвольным блокам, сохраненным в дельта-сжатых КТ. Такой вариант комбинированного сжатия может приводить к необходимости распаковки каждой дельта-сжатой КТ, даже если требуется лишь последний блок.

В данной работе предложен алгоритм комбинированного сжатия PaComp, обеспечивающий константный по времени доступ к произвольным блокам дельта-сжатой КТ. Суть алгоритма заключается в том, что набор V_t модифицированных блоков данных КТ Λ_t разбивается на пакеты фиксированного размера, которые сжимаются независимо друг от друга универсальным алгоритмом (например, LZ77). Поскольку после применения процедуры универсального сжатия два набора данных одного размера с высокой вероятностью будут иметь различный размер, в КТ также будут сохранены смещения (или размеры) каждого сжатого пакета. Псевдокод алгоритма PaComp представлен в листинге:

```

for  $i \leftarrow 1$  to  $\lceil R_t/Q \rceil$  do
   $f \leftarrow (i - 1) \cdot Q + 1$ 
   $e \leftarrow i \cdot Q$ 
  if  $e > R_t$  then
     $e \leftarrow R_t$ 
  end if
   $V'_{ti} \leftarrow stdcomp(V_{tf}, V_{t(f+1)}, \dots, V_{te})$ 
   $S_{ti} \leftarrow S(V'_{ti})$ 
end for

```

Входными данными для PaComp являются количество Q блоков в пакете, функция $stdcomp$, реализующая алгоритм универсального сжатия, и КТ, сформированная адаптивным алгоритмом дельта-оптимизации.

Результатом работы алгоритма является КТ

$$\Sigma'_t = \langle V'_t, p_t, d, R_t, x_t, b_t, Q, S_t \rangle$$

(V'_t — набор сжатых пакетов, содержащих модифицированные блоки; S_t — набор, содержащий размеры сжатых пакетов).

4. Реализация предложенных алгоритмов. На основе рассмотренных алгоритмов дельта-оптимизации разработан программный инструментарий НВІСТ (hash based incremental checkpointing tool) [12], предназначенный для работы с произвольными средствами создания КТ. Реализация выполнена для ОС GNU/Linux. Как отмечено выше, объемы КТ для некоторых программ сопоставимы с объемом оперативной памяти вычислительных узлов ВС, поэтому базовая КТ Λ_b и наиболее поздняя КТ Λ_{t-1} на вычислительных узлах ВС представляются в виде сигнатур g_b и g_{t-1} соответственно. Сигнатура КТ содержит набор хеш-значений, вычисленных для каждого блока этой КТ. Поскольку размер блока значительно больше размера соответствующего ему хеш-значения, для любых i справедливо утверждение $S(\Lambda_i) \gg S(g_i)$.

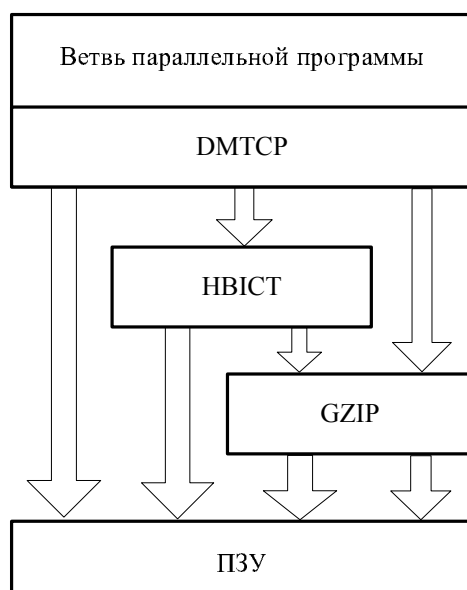


Рис. 1. Схема взаимодействия HVIC и DMTCP

В HVIC предусмотрено два режима дельта-сжатия: контролируемый и автоматический. Контролируемый режим позволяет: 1) создавать базовую КТ Λ_b и ее сигнатуру g_b ; 2) выполнять дельта-сжатие некоторой КТ Λ_i относительно КТ Λ_b , представленной в виде сигнатуры g_b , используя алгоритм FHash; 3) обновлять текущую сигнатуру g_b до g_i для заданной КТ Λ_i . Вычисление порядкового номера КТ, хранение и управление сигнатурами и другой информацией осуществляются внешними программными средствами.

Предложенный адаптивный подход к дельта-сжатию доступен только в автоматическом режиме. В этом случае создание, хранение и управление сигнатурами, а также вычисление порядковых номеров КТ производятся автоматически средствами HVIC. Для хранения сигнатур применяются служебные директории.

В качестве универсального сжатия в алгоритме PaComp используется реализация алгоритма LZ77, выполненная в программе GZIP. Взаимодействие HVIC и GZIP осуществляется через программный канал ОС GNU/Linux. Содержимое блоков данных, обнаруженных при дельта-оптимизации, поступает на стандартный ввод программы GZIP, а ее стандартный вывод направляется в формируемый файл КТ.

Для проведения натуральных экспериментов программный инструмент HVIC был интегрирован со средством создания КТ distributed multithreaded checkpointing (DMTCP) [6]. Схема взаимодействия программных компонентов показана на рис. 1.

5. Результаты моделирования. Исследования эффективности предложенных алгоритмов выполнялись на пространственно-распределенной мультикластерной вычислительной системе Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ СибГУТИ) и Института физики полупроводников им. А. В. Ржанова СО РАН (ИФП СО РАН) [13].

В качестве исследуемых параллельных программ выбраны следующие:

1) программа численного моделирования газодинамических процессов в камере сгорания автомобильного устройства безопасности Airbag [14];

2) программа моделирования процессов молекулярной динамики LAMMPS (large-scale atomic/molecular massively parallel simulator) [15], входящая в состав теста SPEC MPI2007 для оценки производительности суперкомпьютеров.

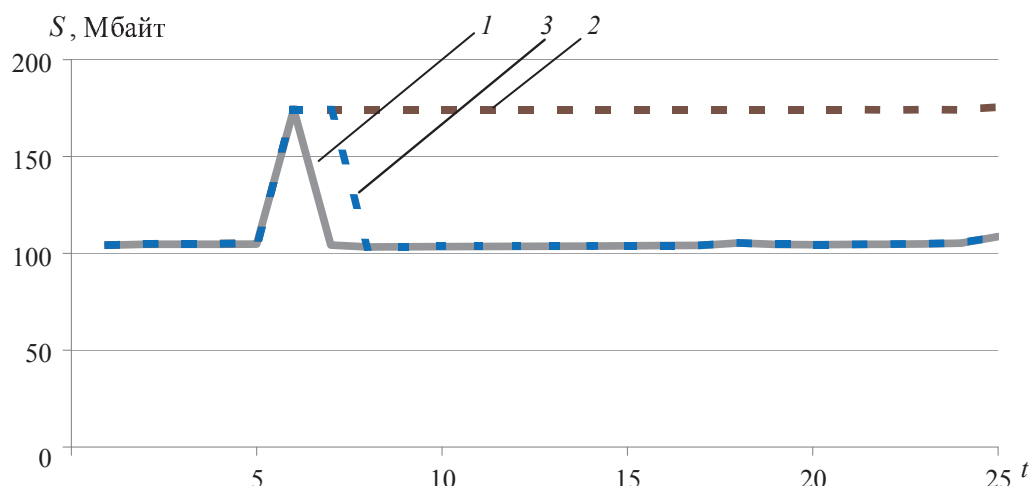


Рис. 2. Эффективность алгоритма ADCA для программы Airbag: S — объем КТ; t — номер КТ; 1 — инкрементное дельта-сжатие; 2 — дифференциальное дельта-сжатие; 3 — адаптивное дельта-сжатие

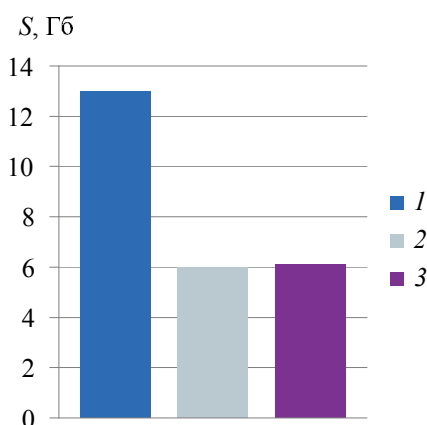


Рис. 3. Средний объем S распределенных КТ, созданных для параллельной программы LAMMPS с помощью исследуемых алгоритмов: 1 — алгоритм ADCA; 2 — алгоритм ADCA с последующим сжатием алгоритмом LZ77; 3 — алгоритм ADCA с последующим сжатием алгоритмом PaComp

На рис. 2 показана эффективность алгоритма ADCA при дельта-сжатии КТ, которые формировались для программы Airbag. Видно, что размер КТ, созданной с помощью алгоритма ADCA, сопоставим с размером инкрементной КТ. При этом для формирования результирующей КТ Λ_t при $t \leq 7$ требуются базовая КТ Λ_1 и дельта-сжатая КТ Σ_t , при $8 \leq t \leq 25$ — базовая КТ Λ_1 , дельта-сжатая КТ Σ_7 и КТ Σ_t .

Моделирование алгоритма PaComp проводилось с использованием восьми контрольных точек, созданных для программы LAMMPS. С помощью предложенного адаптивного подхода к сформированным КТ было применено дельта-сжатие. Затем копия дельта-сжатых КТ была дополнительно сжата алгоритмом LZ77, реализованным в программе GZIP. Отдельно копия указанного набора КТ была сжата с помощью алгоритма PaComp, также использующего программу GZIP. Таким образом, получено три набора дельта-сжатых КТ, размеры которых приведены на рис. 3. Видно, что эффективность сжатия алгоритмом PaComp сопоставима с эффективностью используемого универсального алгоритма. При этом алгоритм PaComp позволяет существенно уменьшить время формирования исходной КТ по набору сжатых (рис. 4).

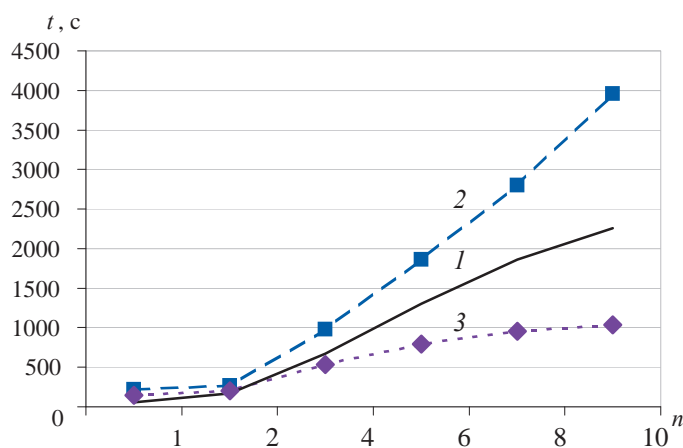


Рис. 4. Зависимость времени t создания результирующей КТ от количества n ветвей параллельной программы LAMMPS по различным наборам: 1 — дельта-сжатых КТ; 2 — дельта-сжатых КТ, дополнительно обработанных алгоритмом LZ77; 3 — дельта-сжатых КТ, дополнительно обработанных алгоритмом PaComp

Заключение. В работе предложены алгоритмы оптимизации распределенных КТ восстановления параллельных программ, основанные на технологии дельта-сжатия. Построен алгоритм ADCA (adaptive delta-compression algorithm), позволяющий адаптивно выбирать между известными видами дельта-сжатия (инкрементным и дифференциальным), основываясь на оценке их эффективности для конкретной программы. Разработан алгоритм PaComp, который совмещает дельта-сжатие и алгоритмы, применяемые в программах-архиваторах. Показано, что PaComp обеспечивает субминимальное время формирования результирующей КТ.

Созданные алгоритмы реализованы в программном инструментарии НВИСТ [11, 12], интегрированном с пакетом DMTCР [6] создания КТ. Разработанный пакет является частью системного программного обеспечения пространственно-распределенной мультикластерной вычислительной системы ЦПВТ СибГУТИ и ИФП СО РАН.

Список литературы

1. Хорошевский В. Г., Курносков М. Г., Мамойленко С. Н., Поляков А. Ю. Архитектура и программное обеспечение пространственно распределенных вычислительных систем // Вестн. СибГУТИ. 2010. № 2. С. 112–122.
2. PHILP I. Software failures and the road to a petaflop machine // 1st Workshop on high performance computing reliability issues. Proc. of the 11th Intern. symp. high performance comput. architecture (HPCA-11), San Francisco (USA), 12–16 Feb. 2005. Washington: IEEE Comput. Soc., 2005.
3. ADIGA N. R., ALMASI G., ALMASI G. S. An overview of the BlueGene/L supercomputer // Proc. of the IEEE/ACM SC2002 conf. (SC'02), Baltimore (USA), 16–22 Nov. 2002. Washington: IEEE Comput. Soc., 2002. P. 1–22.
4. ELNOZHAN E. N., ALVISI L., WANG Y. M., JOHNSON D. B. A survey of rollback-recovery protocols in message-passing systems // ACM Comput. Surveys. 2002. V. 34, N 3. P. 375–408.
5. HURSEY J., SQUYRES J. M., MATTOX T. I., LUMSDAINE A. The design and implementation of checkpoint/restart process fault tolerance for Open MPI // Proc. of the 21st IEEE Intern. parallel and distributed processing symp. (IPDPS), Long Beach (USA), 26–30 March 2007. Washington: IEEE Comput. Soc., 2007. P. 1–8.

6. ANSEL J., ARYA K., COOPERMAN G. DMTCP: Transparent checkpointing for cluster computations and the desktop // Proc. of IEEE Intern. parallel and distributed processing symp. (IPDPS'09), Rome (Italy), 23–29 May 2009. Washington: IEEE Press, 2009. P. 1–12.
7. KISWANY S. A., RIPEANU M., VAZHKUDAI S. S., GHARAIBEH A. stdchk: A checkpoint storage system for desktop grid computing // Proc. of the 28th IEEE Intern. conf. distributed comput. sys. (ICDCS 2008), Beijing (China), 17–20 June 2008. Washington: IEEE Comput. Soc., 2008. P. 613–624.
8. SANGHO Y. S., HEO Y. J., CHO Y., HONG J. Adaptive page-level incremental checkpointing based on expected recovery time // Proc. of the 21st Annual ACM symp. on applied comput., Dijon (France), 23–27 Apr. 2006. N. Y.: ACM, 2006. P. 1472–1476.
9. PLANK J. S., XU J., NETZER R. Compressed differences: An algorithm for fast incremental checkpointing: Tech. report / Univ. of Tennessee. CS-95-302; Tennessee, 1995.
10. AGARWAL S., GARG R., GUPTA M. S., MOREIRA J. E. Adaptive incremental checkpointing for massively parallel systems // ICS '04: Proc. of the 18th Annual intern. conf. on supercomputing, Saint-Malo (France), June 26 — July 1, 2004. N. Y.: ACM Press, 2004. P. 277–286.
11. Поляков А. Ю., Молдованова О. В. Дельта-оптимизация контрольных точек восстановления параллельных программ // Вестн. ТГУ. Сер. Управление, вычислительная техника и информатика. 2011. № 2. С. 72–80.
12. Сайт проекта HBICT. [Электрон. ресурс]. <http://hbict.cpct.sibsutis.ru>.
13. Центр параллельных вычислительных технологий ФГБОУ ВПО “СибГУТИ” и ИФП СО РАН [Электрон. ресурс]. <http://cpct.sibsutis.ru/>.
14. Рычков А. Д., Шокина Н. Ю., Милошевич Х. Моделирование процесса зажигания гранулированного унитарного твердого топлива в камере сгорания айрбэга // Материалы Междунар. конф. “Вычислительные и информационные технологии в науке, технике и образовании”, Павлодар (Казахстан), 20–22 сент. 2006. Павлодар: ТОО НПФ “ЭКО”, 2006. Т. 2. С. 165–175.
15. LAMMPS molecular dynamics simulator [Электрон. ресурс]. <http://lammmps.sandia.gov/>.

*Поляков Артем Юрьевич — канд. техн. наук, мл. науч. сотр.
Института физики полупроводников им. А. В. Ржанова СО РАН;
тел.: (383) 330-56-26; e-mail: artpol@isp.nsc.ru;
Молдованова Ольга Владимировна — канд. техн. наук, доц. Сибирского
государственного университета телекоммуникаций и информатики;
тел.: (383) 269-82-75; e-mail: ovm@csc.sibsutis.ru*

Дата поступления — 20.01.12 г.