Отказоустойчивая реализация метода молекулярной динамики на примере одного приложения*

А.А. Бондаренко, В.О. Подрыга, С.В. Поляков, М.В. Якобовский
ИПМ им. М.В. Келдыша РАН

Обеспечение отказоустойчивых высокопроизводительных вычислений является одной из задач связанных с переходом на вычислительные системы Экзаскейл уровня. Ряд специалистов в этой области рассматривают интерфейс User Level Failure Mitigation (ULFM), как наиболее перспективное средство в решении задачи обеспечения отказоустойчивости. ULFM является дополнением разрабатываемого стандарта MPI, которое позволяет восстанавливать MPI среду и обрабатывать отказы в системе на уровне пользователя. Рассматривается алгоритм, обеспечивающий возможность проведение длительных вычислений на примере решения задачи моделирования процессов взаимодействия газовой смеси с металлической поверхностью с помощью метода молекулярной динамики. Предложена схема обеспечения отказоустойчивости, которая осуществляет сохранение контрольных точек в оперативную память вычислительных узлов, и реализован гибридный метод, сохраняющий как в оперативную память, так и в распределенную файловую систему. Моделирование отказов в системе осуществлялось с помощью разработанной библиотеки моделирования отказов, реализующей функции ULFM на основе MPI стандарта 3.0. Показано, что использование специальной схемы записи в оперативную память позволяет сократить до 15% времени вычислений по сравнению с гибридным методом сохранением контрольных точек.

Ключевые слова: параллельное программирование, метод молекулярной динамики, ULFM, контрольные точки, моделирование отказов, отказоустойчивые вычисления.

1. Введение

Современные суперкомпьютерные центры мира оснащены системами, содержащими сотни тысяч вычислительных ядер [1] и других аппаратных компонент. Отказы аппаратуры в вычислительных системах такого масштаба возникают регулярно [2-4]. Общая частота возникновения отказов возрастает так же за счет отказов в программах, обусловленных сложностью вычислительных систем [2-4]. Оценивая частоту отказов будущих суперкомпьютеров, исследователи говорят о ее возрастании. Предположительно, время безотказной работы перспективных вычислительных систем сократится до величин, порядка одного 1 часа и меньше [3, 4]. Для обеспечения отказоустойчивости, как правило, используется один из методов сохранения контрольных точек, обеспечивающий возможность, в случае возникновения отказа, перезапуска приложения с целью продолжения расчета с сохраненных данных. Однако для приложений использующих большое число параллельных процессов использование классического метода сохранения контрольных точек чревато, во-первых, повышенными накладными расходами на систему ввода-вывода. Согласно [2, 5], будущие вычислительные системы экзафлопсного уровня будут тратить более 50% своего времени на чтение или запись контрольных точек. Во вторых, возникает принципиальная проблема обеспечения возможности записи контрольной точки и рестарта в том случае, когда время между отказами оказывается недостаточным для выполнения записи и чтения требуемых данных.

Таким образом, для обеспечения самой возможности проведения высокопроизводительных вычислений на системах экзафлопсного класса необходимо разработать:

- принципы сохранения контрольных точек за время, меньшее характерной продолжительности безотказной работы системы;

* Работа выполнена при поддержке Российского фонда фундаментальных исследований по грантам 15-29-07090 офи_м, 15-07-04213 А.

- алгоритмы, обеспечивающие, в случае отказа части оборудования, быстрое автоматическое возобновление расчета на работоспособной части вычислительного поля.

На данный момент ведутся разработки разнообразных средств обеспечения отказоустойчивости: на уровне системы, основанные на BLCR [6]; на уровне пользователя (ярким примером являются Charm++[7] и ULFM [8]); средства, основанные на односторонних операциях обмена [9]. Однако общепризнанных решений в настоящее время нет. Наиболее предпочтительным выглядит делегирование ответственности за реализацию техник отказоустойчивости пользователю, что позволит варьировать объем и содержание контрольных точек, в соответствии с известной пользователю семантикой проводимых вычислений. На данный момент стандарт MPI, позволяющий реализовывать методы обеспечения отказоустойчивости на уровне пользователя, находится в разработке [8]. Программная реализация этого стандарта развивается как ответвление проекта Open MPI [10]. Однако сроки появления, пригодных к эксплуатации, реализаций нового стандарта MPI в настоящее время не определены.

Цель работы заключается в разработке методов обеспечения отказоустойчивости длительных высокопроизводительных вычислений на примере моделирования динамики поведения потока газа в ограниченном канале с помощью метода молекулярной динамики. Для тестирования предлагаемых методов используется разработанная библиотека моделирования отказов.

2. Библиотека моделирования отказов

Как уже было отмечено, для решения поставленной цели используется библиотека, позволяющая моделировать, как сами отказы, так и работу функций расширения ULFM для стандарта MPI. Использование спецификации ULFM вызвано естественным стремлением обеспечения совместимости с разрабатываемым в настоящее время стандартом MPI 4.

Разработана библиотека, содержащая модифицированные MPI функции обмена. Функции данной библиотеки возвращают значения, не предусмотренные стандартом MPI 3.1, позволяющие моделировать отказ в нормально функционирующей распределенной вычислительной системе согласно спецификации [8] расширения ULFM. Также в этой библиотеке с помощью функций стандарта MPI 3.1 реализованы аналоги функций MPI_COMM_REVOKE, MPI_COMM_SHRINK, MPI_COMM_FAILURE_ACK, MPI_COMM_FAILURE_GET_ACKED, MPI_COMM_AGREE, что позволяет проводить тестирование на надежных вычислительных системах различных техник отказоустойчивости.

Обнаружение и обработка отказа происходит согласно спецификации ULFM [8], а именно, моделирование отказа осуществляется следующим образом: при вызове некоторой MPI-функции один из участвующих процессов объявляется отказавшим, а в остальных участвующих в коммуникации процессах функции обмена данными возвращают код ошибки, свидетельствующий о наличии отказа в системе.

Операция восстановления после отказов в системе может осуществляться различным образом, например, с помощью функций Failure_ack, Failure_get_acked [8]. Наиболее естественным является проверка кода возврата и, в случае наличия отказа, вызов функции Revoke, которая помечает коммуникатор, как не пригодный для обменов. Таким образом, все последующие вызовы MPI функций на данном коммуникаторе возвращают код ошибки, свидетельствующий о наличии отказа в системе. В программе, в ключевых местах кода, расположены условные операторы, проверяющие наличие отказа в системе (осуществляется с помощью функции Agree). При положительном ответе осуществляется вызов функции восстановления для всех не отказавших процессов коммуникатора. Вызывается функция Shrink, создающая новый коммуникатор без отказавших процессов, на котором осуществляется восстановление глобальной контрольной точки и продолжается расчет.

Заметим, что вызов Shrink приведет к сокращению числа MPI-процессов в программе. Для алгоритмов, использующих метод коллективного решения, Shrink является естественным и удобным способом восстановления. Для других методов распараллеливания сокращение числа процессов требует дополнительных действий. В худшем случае может потребоваться проведение перебалансировки задачи на оставшееся число MPI-процессов. Ввиду очевидно большой трудоемкости выполнения перебалансировки предлагается использовать другой подход, основанный на использовании предварительно зарезервированных процессорных узлов.

3. Моделирование процессов взаимодействия газовой смеси с металлической поверхностью

Постановка задачи моделирования динамики поведения газа и описание используемых численных методов представлено в [11]. Далее приведено описание соответствующего отказоустойчивого алгоритма. В основе программы численного моделирования [11] лежат принципы геометрического и функционального параллелизма. В нашем случае расчетная область в целом разбивается на локальные домены одинаковой мощности. Мощность домена измеряется в количестве элементарных боксов, в каждом из которых молекулы обязательно взаимодействуют друг с другом. Разбиение на домены производится в рамках топологии "трехмерная решетка", поскольку расчетная область является трехмерным параллелепипедом. Каждый расчетный домен попадает на свой вычислитель, в качестве которого используется узел кластера или суперкомпьютера. Взаимодействие процессов, выполняющихся на разных вычислительных узлах, реализовано с помощью библиотеки MPI.

Каждый вычислительный узел обрабатывает один расчетный домен, объединяющий некоторое количество элементарных боксов, сгруппированных в трехмерную подрешетку. Такая структура используется для дальнейшего распределения вычислений по тредам центральных процессоров (ЦПУ), поскольку обмены внутри боксов имеют более высокую интенсивность, чем между боксами. В итоге, межмолекулярные взаимодействия в боксах реализуются в параллельном режиме за счет использования технологии OpenMP.

Во избежание необходимости перебалансировки после использования Shrink предлагается все множество запущенных MPI-процессов разделить на «рабочие», «дополнительные» и «отказавшие». В начале работы все множество процессов разбивается на «рабочие», формирующие рабочее поле, и «дополнительные», которые простаивают, в ожидании вызова обработчика отказа в системе. По мере возникновения отказов, часть «рабочих» переходит в «отказавшие», а «дополнительные» восполняют соответствующую часть «рабочих». В результате, число «рабочих» процессов остается постоянным. Начальное число необходимых «дополнительных» процессов зависит от ожидаемой продолжительности расчета и оцениваемой частоты отказов в системе.

Основной алгоритм расчета выглядит следующим образом.

Этап 1. Инициализация элементов библиотеки моделирования отказов.

Этап 2. Чтение данных (исходных/из контрольной точки) MPI-приложением и инициализация структур данных каждым рабочим MPI-процессом.

Этап 3. Основные вычисления в цикле по времени.

Этап 4. Проверка нормального выхода из цикла, в случае необходимости запуск функции восстановления и последующий переход на этап 2.

Этап 5. Выполнение результирующих вычислений и завершение работы.

Ключевым для первого этапа является разбиение множества процессов на «рабочие» и «дополнительные», а так же создание соответствующих коммуникаторов.

На втором этапе, в частности, выполняется определение числа молекул в расчетной области, генерация их размещения, а также генерация равномерного по углам и максвелловского по модулю распределения их импульсов и расчет стартовых сил.

Отличием описанного подхода от других является использование структур данных, опирающихся на элементарные боксы, размеры которых связаны с радиусом обрезания потенциалов взаимодействия и параметрами кристаллической решетки металла. Такой подход является достаточно затратным по объему необходимой оперативной памяти, однако значительно экономит время при вычислениях. Дело в том, что обработка малого числа частиц в нескольких смежных боксах, расположенных практически в одной или нескольких смежных страницах оперативной памяти, выполняется максимально быстро и хорошо кэшируется. Фактически, используется прием локализации памяти, при котором процессору при интенсивных вычислениях не требуется переключаться между далеко отстоящими друг от друга страницами оперативной памяти.

Кроме того, удалось избежать дорогостоящей процедуры определения принадлежности различных частиц к конкретным боксам, проводящейся в других кодах на каждом шаге по вре-

мени. Еще один плюс от использования боксовой структуры заключается в том, что пересылка частиц между узлами многопроцессорной вычислительной системы (МВС), связанная с движением последних по расчетной области, интегрируется как с реализацией периодических граничных условий, так и с обменами информацией между смежными по кубической решетке MPI-процессами. Последнее достигается использованием теневых (виртуальных) боксов с фиктивными частицами, взаимодействие с которыми основных частиц должно учитываться в потенциальной энергии системы.

В рамках цикла по времени реализована следующая последовательность действий. Сначала вычисляются новые значения координат. Далее производится их коррекция с помощью периодических граничных условий. После этого производится обмен частицами между конкретными боксами, который осуществляется как внутри расчетных доменов, так и между расчетными доменами с помощью функций MPI из библиотеки моделирования отказов. Осуществляется проверка наличия отказа и, в случае положительного ответа, осуществляется переход на Этап 4.

Далее происходит расчет сил, на основании которых производится коррекция скоростей частиц и вычисляются все необходимые интегральные характеристики. При достижении контрольных моментов времени необходимые данные сохраняются в контрольные точки.

В случае наличия отказа в системе, на четвертом этапе запускается процедура восстановления. Она заключается в запуске функции Shrink, и в последующем преобразовании «рабочих» и «дополнительных» процессов, таким образом, чтобы дополнительные процессы получили номера отказавших процессов. Это преобразование позволяет сделать процедуру чтения контрольных точек проще, так как всем ранее «рабочим» процессам достаточно прочитать данные из своей оперативной памяти. Чтение новыми рабочими процессами осуществляется в зависимости от метода сохранения контрольных точек, средствами MPI, в случае хранения данных в оперативной памяти узлов, или файловыми операциями, в случае использования общего дискового массива. Переход на второй этап.

4. Вычислительный эксперимент

В данной работе были разработаны две программы, в которых реализован алгоритм моделирования процессов взаимодействия газовой смеси с металлической поверхностью, описанный в третьем параграфе статьи. Во время выполнения одной итерации из расчетного цикла происходят обмены информации между процессами, в связи с чем необходимо использовать координированный протокол, либо нескоординированный с методами логирования [12]. В данной работе сохранение контрольных точек проходит по координированному протоколу. Следует отметить, что сохранение контрольных точек осуществляется через определенные периоды согласно [13], которые зависят от параметров, описывающих функционирование вычислительной системы, работу программы и моделируемые отказы.

В первой программе сохранение контрольных точек осуществляется в оперативную память вычислительных узлов по схеме сохранения описанной в работе [14]. А именно, исходя из параметров, описывающих работу программы и моделируемые отказы (время работы программы без средств обеспечения отказоустойчивости, объем контрольной точки, размер оперативной памяти на один MPI-процесс, ожидаемая частота отказа в системе), для каждого MPI-процесса определяются номера MPI-процессов, которым следует передавать данные для сохранения.

Во второй программе осуществляется гибридное сохранение контрольных точек как в оперативную память, так и в распределенную файловую систему. Отметим, что для возможности восстановления расчетов в первой программе необходимо дублирование локальных контрольных точек, что реализуется с помощью специальной схемы сохранения [14]. А для второй программы дублирование не требуется, так как через распределенную файловую систему все локальные контрольные точки непосредственно доступны новым MPI-процессам, введенным в расчетное поле. Отметим, что в случае отказа MPI-процесса, он заменяется на новый, который считывает контрольную точку из файловой системы, а остальные процессы считывают свои контрольные точки из оперативной памяти.

Описание алгоритмов восстановления для первой и второй программы приведены в табл 1. Основными отличиями являются место хранения контрольных точек, а также необходимость определять глобальную контрольную точку для восстановления в первой программе (во второй

осуществляется только чтение последней) и, как следствие, возможный пересчет нескольких итераций расчетного цикла. Получаем, что минусами для первой программы является больший пересчет итераций при кратных отказах (отказывают несколько процессов одновременно); для второй – накладные расходы на сохранение контрольных точек в распределенную файловую систему.

Таблица 1. Отличия в алгоритмах восстановления программ после отказа

<table>
<tr><th>Первая программа</th><th>Вторая программа</th></tr>
<tr><td colspan="2">• Восстановить MPI-среду;
• восстановить рабочее поле;</td></tr>
<tr><td>• определить последнюю достижимую глобальную контрольную точку (она расположена в оперативной памяти MPI-процессов не пострадавших от отказа);
• передать (через MPI интерфейс) необходимые локальные контрольные точки новым рабочим MPI-процессам;</td><td>• прочитать (из распределенной файловой системы) последнюю локальную контрольную точку новыми рабочими MPI-процессам;</td></tr>
<tr><td colspan="2">• затем осуществить восстановление работы прикладной программы с локальных контрольных точек.</td></tr>
</table>

Моделируемая микросистема содержала пластинку никеля с размерами 288×288×24 ребер (элементарной ячейки кристаллической решетки); число атомов никеля составляет 8128512; бокс имеет размеры 96×96×8 ребер (элементарной ячейки кристаллической решетки), размер контрольной точки был порядка 4 Мб. Время работы программы реализованной без средств обеспечения отказоустойчивости составляет 225 минут. Первая и вторая программы запускались на 256 MPI-процессах при разных значениях среднего времени между отказами $MTBF = \{30,60,90,120,150,180\}$ (мин), соответствующее количество модельных отказов составило – $\{7,4,2,2,1,1\}$. В данном примере моделировались одинарные отказы.

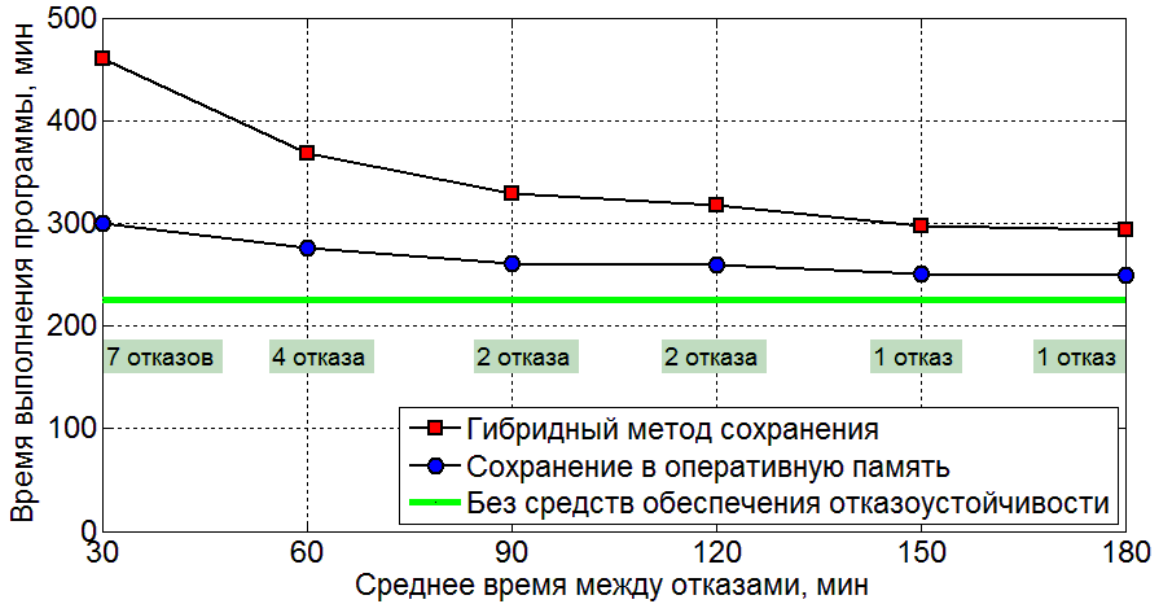

Рис. 1. Времена выполнения программ, реализующих различные техники отказоустойчивости, при различных значениях MTBF; время работы программы без сбоев и без средств обеспечения отказоустойчивости

5. Заключение

В результате тестовых расчетов было получено, что использование специальной схемы сохранения в оперативную память позволяло сократить до 15% времени вычислений по сравнению с гибридным методом сохранением контрольных точек. Однако координированный протокол распределенного сохранения в оперативную память вычислительных узлов должен столкнуться с проблемой уменьшения объема оперативной памяти на ядро при подходе к Экзаскейл уровню. Отметим, что данная схема сохранения обобщается до распределенного сохранение контрольных точек в различные средства хранения вычислительного узла HDD, SSD или другие будущие устройства хранения. Возможность применения такого сохранения контрольных точек также должна быть исследована. Но подобное тестирование на данный момент затруднено, так как многие вычислительные системы либо не имеют дисков для каждого вычислительного узла, либо не дают возможности пользователю работать с этими средствами хранения.

Литература

1. Top 500 The List. URL: http://top500.org/ (дата обращения: 25.11.2015)
2. Schroeder B., Gibson G. A. Understanding failures in petascale computers. Journal of Physics: Conference Series. 2007. Vol. 78, No 1.
3. SFT: Scalable Fault Tolerance URL: http://hpc.pnl.gov/sft/ (дата обращения: 25.11.2015)
4. Cappello F., Geist A., Gropp W., Kale S., Kramer B., Snir M. Toward Exascale Resilience: 2014 update // Supercomputing frontiers and innovations. 2014. Vol.1, No. 1. P. 1–28.
5. Elnozahy, E., Plank, J. Checkpointing for Petascale systems: a look into the future of practical rollback-recovery. Dependable and Secure Computing, IEEE Transactions on 1, 2. Apr. 2004, P. 97–108.
6. Berkeley Lab Checkpoint/Restart (BLCR) for LINUX URL: http://crd.lbl.gov/departments/computer-science/CLaSS/research/BLCR/ (дата обращения: 25.11.2015)
7. Parallel Languages/Paradigms: Charm++ URL: http://charm.cs.illinois.edu/research/charm. (дата обращения: 25.11.2015)
8. Fault Tolerance Research Hub URL: http://fault-tolerance.org/ulfm/ulfm-specification (дата обращения: 25.11.2015).
9. M. Besta, T. Hoefler "Fault Tolerance for Remote Memory Access Programming Models" // In Proceedings of the 23rd ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC'14), presented in Vancouver, Canada, ACM, Jun. 2014.
10. ULFM-1.1 Release URL: http://fault-tolerance.org/2015/11/14/ulfm-1-1-release/ (дата обращения: 25.11.2015).
11. Подрыга В.О., Поляков С.В., Пузырьков Д.В.Суперкомпьютерное молекулярное моделирование термодинамического равновесия в микросистемах газ-металл // Вычислительные методы и программирование. 2015. Том. 16, No. 3, С. 123–138.
12. Elnozahy E.N., Alvisi L., Wang Y., Johnson D.B. A Survey of Rollback-Recovery Protocols in Message-Passing Systems // ACM Computing Surveys. — 2002. — Vol.34, No. 3 — P. 375–408.
13. Aupy G., Benoit A.,Herault T., Robert Y., Dongarra J. Optimal Checkpointing Period: Time vs.Energy // High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation: 4th International Workshop, PMBS 2013, November 18, 2013, Denver, CO, USA, Proceedings. Springer. 2014.
14. Бондаренко А.А. Якобовский М.В. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек // Вестник ЮжноУральско-

го государственного университета. Серия «Вычислительная математика и информатика». 2014. Том. 3, No. 3, С. 20–36.

The fault-tolerant implementation of the method of molecular dynamics on the example of one application

A.A. Bondarenko, V.O. Podryga, S.V. Polyakov, M.V. Iakobovski

Keldysh Institute of Applied Mathematics, Russian Academy of Sciences

Fault-tolerant is one of problems which appear on the road to Exascale computing. HPC scientists consider ULFM (User Level Failure Mitigation) as perspective tool for solving fault-tolerant challenge. ULFM is a set of MPI interface extensions enabling Message Passing programs to restore MPI and handle with faults on user level. We consider an algorithm which simulates interaction of the gas mixture with a metal surface using the method of molecular dynamics. For this algorithm we propose a resiliency scheme, that checkpointing in the memory of the computing nodes and implement hybrid method of checkpointing both in operative memory and in the distributed file system. For fault simulations we use special library which realize some of the ULFM functions by MPI standard 3.0. Experiments show that checkpointing in operative memory can reduce overhead to 15% compare to hybrid checkpointing.

Keywords: parallel programming, the method of molecular dynamics, ULFM, checkpoint, simulation of failures, fault-tolerant computation.

References

1. Top 500 The List. URL: http://top500.org/ (accessed: 25.11.2015)
2. Schroeder B., Gibson G. A. Understanding failures in petascale computers. Journal of Physics: Conference Series. 2007. Vol. 78, No 1.
3. SFT: Scalable Fault Tolerance URL: http://hpc.pnl.gov/sft/ (accessed: 25.11.2015)
4. Cappello F., Geist A., Gropp W., Kale S., Kramer B., Snir M. Toward Exascale Resilience: 2014 update // Supercomputing frontiers and innovations. 2014. Vol.1, No. 1. P. 1–28.
5. Elnozahy, E., Plank, J. Checkpointing for Petascale systems: a look into the future of practical rollback-recovery. Dependable and Secure Computing, IEEE Transactions on 1, 2. Apr. 2004, P. 97–108.
6. Berkeley Lab Checkpoint/Restart (BLCR) for LINUX URL: http://crd.lbl.gov/departments/computer-science/CLaSS/research/BLCR/ (accessed: 25.11.2015)
7. Parallel Languages/Paradigms: Charm++ URL: http://charm.cs.illinois.edu/research/charm. (accessed: 25.11.2015)
8. Fault Tolerance Research Hub URL: http://fault-tolerance.org/ulfm/ulfm-specification (accessed: 25.11.2015).
9. M. Besta, T. Hoefler "Fault Tolerance for Remote Memory Access Programming Models" // In Proceedings of the 23rd ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC'14), presented in Vancouver, Canada, ACM, Jun. 2014.
10. ULFM-1.1 Release URL: http://fault-tolerance.org/2015/11/14/ulfm-1-1-release/ (accessed: 25.11.2015).
11. Podryga V.O., Polyakov S.V., Puzyr'kov D.V.Superkomp'yuternoe molekulyarnoe modelirovanie termodinamicheskogo ravnovesiya v mikrosistemakh gaz-metall [Supercomputer Molecular Modeling of Thermodynamic Equilibrium in Gas–Metal Microsystems]. Vychislitel'nye metody i programmirovanie. 2015. Vol. 16, No. 3, P. 123–138.

12. Elnozahy E.N., Alvisi L., Wang Y., Johnson D.B. A Survey of Rollback-Recovery Protocols in Message-Passing Systems // ACM Computing Surveys. — 2002. — Vol.34, No. 3 — P. 375–408.
13. Aupy G., Benoit A.,Herault T., Robert Y., Dongarra J. Optimal Checkpointing Period: Time vs.Energy // High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation: 4th International Workshop, PMBS 2013, November 18, 2013, Denver, CO, USA, Proceedings. Springer. 2014.
14. Bondarenko A.A., Yakobovskiy M.V. Obespechenie otkazoustoychivosti vysokoproizvoditel'nykh vychisleniy s pomoshch'yu lokal'nykh kontrol'nykh tochek [Fault Toler-ance for HPC by Using Local Checkpoints]. Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya «Vychislitel'naya matematika i informatika» [Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering]. 2014. Vol. 3, No. 3. P. 20–36.