УДК 004.052.3 DOI: 10.14529/cmse150301

МОДЕЛИРОВАНИЕ ОТКАЗОВ В ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ В РАМКАХ СТАНДАРТА MPI И ЕГО РАСШИРЕНИЯ ULFM[1]

А.А. Бондаренко, М.В. Якобовский

Рассматривается проблема выполнения длительных расчетов на высокопроизводительных вычислительных системах, компоненты которых подвержены отказам. Для программ, запускаемых на подобных системах, существенным является возможность обработки отказов путем автоматического продолжения расчета на оставшихся работоспособных узлах системы. Возможность обработки отказов предусматривается в разрабатываемом стандарте MPI 3.1. В работе кратко описывается библиотека моделирования отказов для тестирования отказоустойчивых алгоритмов, использующих функционал разрабатываемого стандарта MPI 3.1. Описана техника отказоустойчивости на примере тестовой задачи. Проведено сравнение записи контрольных точек в оперативную память и в распределенную файловую систему.

Ключевые слова: параллельные вычисления, отказоустойчивость, контрольные точки, MPI, ULFM, моделирование отказов.

Введение

Несмотря на прогресс, достигнутый в последние годы, «проблема устойчивости для Экзафлопсных систем не решена, а перед сообществом по-прежнему стоит сложная задача выдачи корректных результатов приложениями, работающими на неустойчивых системах» [1]. Одновременное повышение надежности и производительности высокопроизводительной системы, как единого целого, на аппаратном уровне является трудноразрешимой задачей. В связи с этим, для организации вычислений на современных суперкомпьютерах необходимо развитие новых отказоустойчивых технологий, позволяющих с помощью программных решений корректно продолжать вычисления даже при отказе части оборудования.

Программное обеспечение отказоустойчивости осуществляется на системном уровне или на уровне пользователя. Отказоустойчивость на уровне системы основана на записи всей памяти приложения в глобальную контрольную точку и последующем перезапуске программы из этой контрольной точки в случае отказа. Операции сохранения и перезапуска могут быть выполнены автоматически, поскольку не зависят от специфики прикладной программы. Преимуществом автоматических операций сохранения и перезапуска является простота использования, так как от пользователя не требуется внесения изменений в код программы, что достигается за счет очевидного недостатка — высоких накладных расходов. В контрольную точку записывается все данные, используемые приложением, в том числе данные, которые не являются существенными для запуска программы из контрольной точки.

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015».

В случае реализации отказоустойчивости на уровне пользователя, в контрольную точку входит только то, что явно укажет прикладной программист. В идеале, только те данные, которые необходимы для восстановления утерянной в результате сбоя информации. Так же на уровне пользователя контролируется частота и момент создания контрольных точек при выполнении программы. То есть накладные расходы могут быть значительно уменьшены, однако, потребуется дополнительная работа прикладного программиста для реализации отказоустойчивости в приложении.
Использование методов организации отказоустойчивости вычислений на системном уровне представляется трудно реализуемым для систем Экзафлопсного уровня, поэтому значительное внимание специалистов в этой области уделяется именно методам уровня пользователя [1, 2]. Чтобы применять методы организации отказоустойчивости вычислений на уровне пользователя, средства обмена сообщениями должны обеспечивать возможность коммуникации между вычислительными процессами даже при наличии отказов в вычислительной системе, а также обеспечивать способность восстановления приложения в согласованное состояние, из которого вычисления могут быть продолжены.

Для удовлетворения этих требований ключевым для стандарта MPI является условие [2], согласно которому любая операция должна завершиться за конечное время, даже в случае отказа части оборудования. В противном случае, если некоторая MPI-операция никогда не завершается, то MPI-процесс, вызвавший ее, не может принять участия в восстановлении приложения, тем самым организация отказоустойчивых вычислений становится невозможной. Все обеспечивающие связь между процессами MPI-операции должны возвращать коды ошибок, информирующие приложение о любом состоянии, возникшем в ходе выполнения операции. После того как некоторые процессы были уведомлены об отказе в системе работа по восстановлению может быть начата. Для предотвращения блокировок в MPI должны быть реализованы механизмы предупреждения всех процессов об отказах, когда это необходимо.

Перечисленные выше требования учитываются в расширении ULFM [2, 3] и дорабатываются для внесения в стандарт MPI 3.1. Программные реализации MPI с расширением ULFM позволят реализовывать различные техники отказоустойчивости на уровне пользователя. Следует отметить, что сроки появления пригодных к эксплуатации реализаций MPI с расширением ULFM в настоящее время не определены, а отладка и тестирование отказоустойчивых алгоритмов на реальных вычислительных системах сегодня неэффективны в силу, пока еще, относительно большой длительности времени безотказной работы. Поэтому актуальна задача проверки и тестирования приложений, основанных на техниках отказоустойчивости.

Цель данной работы состоит в разработке методов и средств моделирования отказов в высокопроизводительных системах согласно спецификации ULFM, обеспечивающих возможность изучения и тестирования различных стратегий построения отказоустойчивых параллельных приложений.

Статья организована следующим образом. В первом раздел статьи приведены принципы и основные функции спецификации ULFM, а также кратко описывается содержание библиотеки моделирования отказов. Во втором разделе описаны техники отказоустойчивости для решения тестовой задачи и их программные реализации. В заключении приводится описание результатов работы и направление дальнейших исследований.

1. Моделирование отказов с помощью расширения ULFM

В стандарте MPI 3.0 отсутствует описание каких-либо функций связанных с обеспечением отказоустойчивости в вычислительных системах. Желая предоставить пользователю гибкий интерфейс, авторы ULFM ушли от выбора конкретной техники отказоустойчивости или обязательств восстанавливать приложение целиком после отказа. «Этот интерфейс должен информировать приложение о специальных условиях мешающих передаче сообщений, и предоставлять конструкции и определения, которые позволят приложению восстановить MPI–объекты и способность передавать сообщения» [2, стр. 246]. Выбор наиболее подходящей для данного приложения техники восстановления и реализация остается за пользователем.

Обработка отказов в ULFM основывается на следующих двух положениях [2, 3]. В случае отказа хотя бы одного из MPI-процессов:

- MPI-операция, включающая в себя отказавший MPI-процесс, не должна блокироваться бесконечно, а должна быть либо выполнена успешно, либо вызвать MPI-исключение.
- MPI-операция, не включающая в себя отказавший MPI-процесс, должна завершиться нормально, если только не будет прервана пользователем с помощью специальных функций ULFM.

MPI-исключение возникает только у участников операции, содержащей отказавший процесс, однако может так быть, что не все MPI-процессы, участвующие в этой операции, вернут MPI-исключение. Может возникнуть ситуация, когда MPI-процесс завершит свое участие в этой операции, а отказ произойдет в следующий момент времени. Еще раз подчеркнем, что MPI-исключение отражает только локальное воздействие отказа на операцию, и нет никаких гарантий, что другие процессы, не участвующие в данной операции, также будут уведомлены о возникновении этого отказа. Асинхронное распространение информации о возникновении отказа не гарантируется, и пользователи должны проявлять осторожность при выявлении множества процессов, для которых будет зафиксирован отказ и возникнет MPI-исключение.

Обработка исключений и реализация различных техник восстановления осуществляется, в основном за счет функций [3]:

- MPI_COMM_REVOKE — прекращает все текущие нелокальные операции на коммуникаторе и отмечает коммуникатор в качестве аннулированного. Все последующие вызовы нелокальных функций, связанные с этим коммуникатором, должны завершаться значением MPI_ERR_REVOKED, за исключением функций MPI_COMM_SHRINK и MPI_COMM_AGREE;
- MPI_COMM_SHRINK — создает на основе существующего коммуникатора новый, не содержащий отказавшие процессы;
- MPI_COMM_FAILURE_GET_ACKED — возвращает группу, состоящую из процессов, которые были определены как отказавшие к моменту последнего вызова функции MPI_COMM_FAILURE_ACK;
- MPI_COMM_AGREE — согласовывает значение булевой переменной, если нет отказавших MPI-процессов в коммуникаторе или возвращает исключение о наличии отказа всем не отказавшим процессам в коммуникаторе.

Для тестирования различных техник отказоустойчивости была создана библиотека моделирования отказов, основанная на расширении ULFM для стандарта MPI. Исполь-

зование спецификации ULFM вызвано естественным стремлением сохранить будущую совместимость с разрабатываемым стандартом MPI 3.1. В данной библиотеке каждому MPI-процессу поставлен в соответствие атрибут, значение которого отражает состояние этого процесса: отказал он или нет. Также в ней реализованы модификации MPI функций обмена. Функции обмена данной библиотеки возвращают исключения согласно спецификации [3] расширения ULFM, не предусмотренные стандартом MPI 3.0, позволяющие моделировать отказ в нормально функционирующей распределенной вычислительной системе. Для реализации техник отказоустойчивости с помощью функций стандарта MPI 3.0 в этой библиотеке реализованы функции MPI_COMM_REVOKE, MPI_COMM_SHRINK, MPI_COMM_AGREE, MPI_COMM_FAILURE_ACK, MPI_COMM_FAILURE_GET_ACKED.

2. Два примера реализации техники отказоустойчивости

В данной работе были разработаны три программы. В первой программе реализован параллельный алгоритм решения задачи о распространении тепла в однородном стержне. Остальные программы используют библиотеку моделирования отказов и содержат процедуры автоматического восстановления вычислений, при этом во второй программе сохранение контрольных точек осуществляется в оперативную память вычислительных узлов, а в третье программе запись контрольных точек осуществляется в распределенную файловую систему.

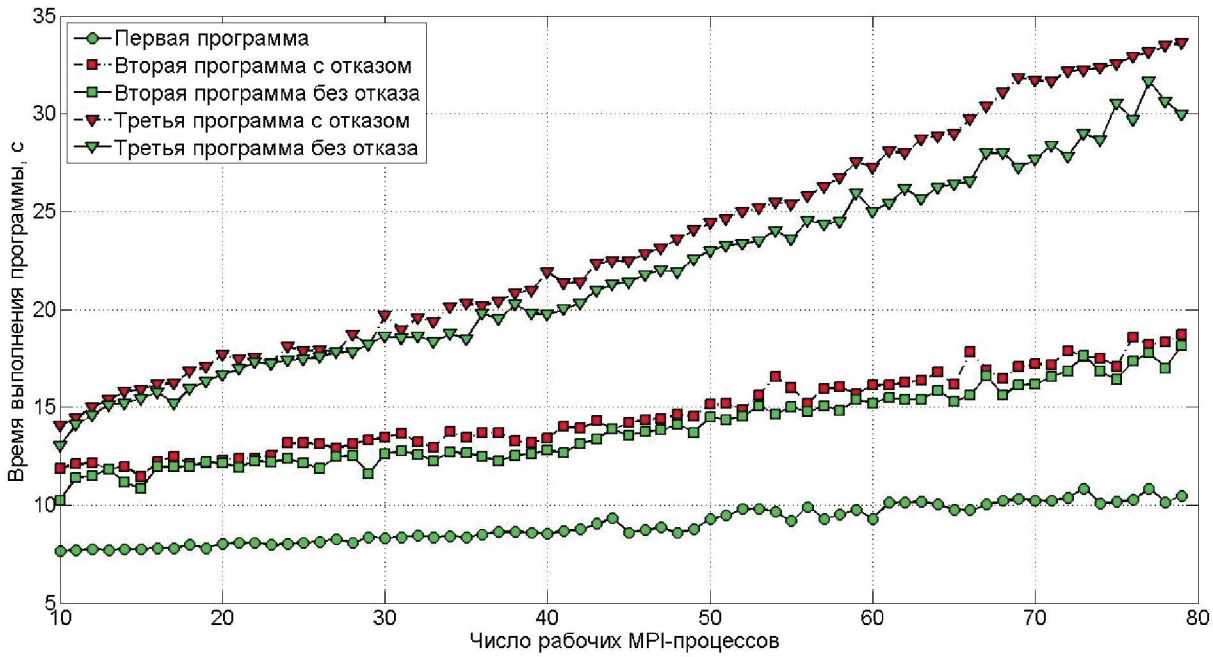
Опишем используемую технику обеспечения отказоустойчивости. Для реализации автоматического восстановления вычислений все множество MPI-процессов разделено на рабочие процессы, формирующие рабочее поле, в котором выполняется основной алгоритм прикладной программы, и на резервные процессы, которые простаивают в ожидании вызова обработчика отказа в системе. В случае возникновения отказа, MPI-процессы, отмеченные как отказавшие, выводятся из расчетного поля, а их место занимают новые из списка резервных процессов. Здесь следует отметить, что разделение на рабочие и резервные процессы происходит внутри самой программы, с помощью стандартных средств MPI. Количество резервных MPI-процессов является параметром для программы, и должно определяться пользователем исходя из предположений о возможном количестве отказов во время выполнения программы. Такое разделение на рабочие и резервные MPI-процессы, связано с тем, что для большинства алгоритмов, описывающих решение задач математической физики, затратными являются процедуры перераспределения работы на меньшее число MPI-процессов, по сравнению с начальными условиями запуска программы.

Создание контрольных точек происходит по координированному протоколу, то есть в определенный момент работы программы, когда отсутствуют обмены между MPI-процессами, осуществляется запись контрольных точек для всех рабочих MPI-процессов. В случае сохранения контрольных точек в локальные устройства хранения, в том числе оперативную память, для MPI-процессов, введенных в рабочее поле, может быть недоступна информация о контрольных точках отказавших MPI-процессов. Для того чтобы устранить этот недостаток необходимо осуществлять дублирование при сохранении контрольных точек в локальные устройства хранения. Во второй программе сохранение контрольных точек в оперативную память с дублированием осуществляется по схеме сохранения описанной в работе [4]. При сохранении контрольных точек в рас-

пределенную файловую систему все контрольные точки непосредственно доступны MPI-процессам, введенным в расчетное поле в случае отказа. Таким образом, для третьей программы дублирование не требуется.

Из вышесказанного следует, что процедуры восстановления для второй и третьей программы отличаются. Для второй программы необходимо: осуществить восстановление MPI-среды; осуществить восстановление рабочего поля; определить глобальную контрольную точку, с которой необходимо продолжить вычисления; осуществить копирование необходимых локальных контрольных точек новым MPI-процессам, введенным в расчетное поле; затем осуществить восстановление работы прикладной программы с локальных контрольных точек. Для третьей программы необходимо: осуществить восстановление MPI-среды; осуществить восстановление рабочего поля; затем перейти к процедуре восстановление процесса вычислений с последней глобальной контрольной точки, записанной в распределенной файловой системе.

Вторая и третья программы запускались в двух режимах: без отказов во время вычислений и с одним отказом двух MPI-процессов в одном и том же месте основного алгоритма. В обоих режимах работы осуществлялось сохранение контрольных точек через одинаковое число итераций основного цикла. Подобный эксперимент позволяет оценить объем накладных расходов для организации отказоустойчивости, а именно временные затраты на сохранение контрольных точек и на восстановление системы после отказа.



Сравнение программ, реализующие разные техники отказоустойчивости

Параметры задачи были выбраны таким образом, чтобы размер контрольной точки был порядка 2 Мб. Для второй и третьей программ разбиение на рабочие и запасные MPI-процессы было проведено так, чтобы было два запасных MPI-процесса. Вычисления проводились на научном кластере ИПМ [5]. Уже на небольшом числе MPI-процессов (до 80), накладные расходы на организацию отказоустойчивости, при сохранение контрольных точек в распределенную файловую систему, превышают соответствующие накладные расходы, при сохранении контрольных точек в оперативную па-

мять с дублированием. Теоретические оценки показывают, что если запустить вторую или третью программу на всем суперкомпьютере Sequoia и взять локальные контрольные точки объемом равным 100 MB каждая, тогда для координированного протокола запись глобальной контрольной точки в распределенную файловую систему будет составлять около 3 минут, а сохранение в оперативную память с дублированием (параметры схемы сохранения SD = 2, DF = 3 [4]), — порядка 2,5 секунд. Отметим, что при больших объемах локальных контрольных точек сохранение в оперативную память вычислительных узлов по описанной в [4] схеме может и не иметь смысла в силу ограничения объема оперативной памяти. А при наличии более быстрых коммуникационных соединений для распределенной файловой системы может существовать объем контрольных точек, при котором будут равны накладные расходы для обеспечения отказоустойчивости второй и третьей программы.

Заключение

Для тестирования и определения эффективности разрабатываемых техник отказоустойчивости необходимо создание соответствующих программных средств. В данной работе представлена библиотека функций моделирования отказов в нормально функционирующих системах, основанная на существующей реализации стандарта MPI 3.0. Она позволяет тестировать различные алгоритмы восстановления вычислений, в том числе, с использованием локальных устройств хранения. В частности, для тестовой задачи на вычислительной системе [5] показано, что при организации автоматического восстановления вычислений сохранение в оперативную память предпочтительнее, чем в распределенную файловую систему. При этом временные затраты на сохранение контрольных точек в оперативную память и на восстановление системы после отказа относительно малы.

Основным направлением дальнейших исследований является разработка и изучение эффективных техник обеспечения отказоустойчивости для решения мультимасштабных задач механики сплошной среды на вычислительных системах сверхвысокой производительности.

Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту 13-01-12073 офи_м.

Литература

1. Cappello, F. Toward Exascale Resilience: 2014 update / F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, M. Snir // Supercomputing frontiers and innovations. — 2014. — Vol. 1, No. 1. — P. 1–28. DOI: 10.14529/jsfi140101.
2. Bland, W. Post-failure recovery of MPI communication capability: Design and rationale / W. Bland, A. Bouteiller, T. Hérault, G. Bosilca, J. Dongarra // International Journal of High Performance Computing Applications. — 2013. — Vol. 27, No. 3. — P. 244–254. DOI: 10.1177/1094342013488238.
3. ICL Fault Tolerance URL: http://fault-tolerance.org/ulfm/ulfm-specification (дата обращения: 01.03.2015).
4. Бондаренко, А.А. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек / А.А. Бондаренко,

М.В. Якобовский // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». — 2014. — Том. 3, No. 3. — С. 20–36. DOI: 10.14529/cmse140302.

5. Scientific Cluster of Keldysh Institute of Applied Mathematics RAS. URL: http://imm6.keldysh.ru/~informer/(дата обращения: 01.03.2015).

Бондаренко Алексей Алексеевич, к.ф.-м.н., научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация), bondaleksey@gmail.com.

Якобовский Михаил Владимирович, д.ф.-м.н., заведующий сектором «Программное обеспечение вычислительных систем и сетей», Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация), lira@imamod.ru

Поступила в редакцию 13 апреля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 3, pp. 5–12

DOI: 10.14529/cmse150301

SIMULATION OF FAILURES IN HIGH-PERFORMANCE COMPUTING SYSTEMS UNDER MPI-ULFM

A.A. Bondarenko, Keldysh Institute of Applied Mathematics (Moscow, Russian Federation) bondaleksey@gmail.com,

M.V. Iakobovski, Keldysh Institute of Applied Mathematics (Moscow, Russian Federation) lira@imamod.ru

In this paper, we consider one of the main problems that occur in the area of high-performance computing is to continue computations despite of failures. For the programs running on such systems it is very important to handle failures and continue computations on working nodes. One of the MPI 3.1 standardization efforts aim is adding new techniques, approaches, or concepts to support for fault tolerance in MPI applications. The paper briefly describes a library for simulation of failures and testing fault-tolerant algorithms using functional of developing MPI 3.1 standard. In the test problem we describe one of the techniques of fault tolerance and we compare checkpoint in operational memory versus checkpoint in the distributed file system.

Keywords: parallel computing, fault tolerance, checkpoint, simulation of failures, MPI, ULFM.

References

1. 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. DOI: 10.14529/jsfi140101.
2. Bland W., Bouteiller A., Hérault T., Bosilca G., Dongarra J. Post-failure recovery of MPI communication capability: Design and rationale // International Journal of High

Performance Computing Applications. 2013. Vol. 27, No. 3. P. 244–254. DOI: 10.1177/1094342013488238.

3. ICL Fault Tolerance URL: http://fault-tolerance.org/ulfm/ulfm-specification (accessed: 01.03.2015).
4. Bondarenko A.A., Yakobovskiy M.V. Obespechenie otkazoustoychivosti vysokoproizvoditel'nykh vychisleniy s pomoshch'yu lokal'nykh kontrol'nykh tochek [Fault Tolerance 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. DOI: 10.14529/cmse140302.
5. Scientific Cluster of Keldysh Institute of Applied Mathematics RAS. URL: http://imm6.keldysh.ru/~informer/ (accessed: 01.03.2015).

Received April 13, 2015.