ЗАДАЧИ И ВОЗМОЖНЫЕ РЕШЕНИЯ В ОБЛАСТИ ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

А.А. Бондаренко, М.В. Якобовский

Институт прикладной математики им. М.В. Келдыша РАН

Создание суперкомпьютеров экзафлопсного уровня производительности сопряжено с необходимостью решения ряда задач [1], среди которых важное место занимает обеспечение возможности выполнения расчетов на вычислительных системах, несмотря на отказ ряда задействованных в расчете вычислительных узлов. Современные суперкомпьютеры состоят из объединенного сетью большого количества вычислительных узлов, каждый из которых содержит множество элементов, в том числе: процессоры, различные ускорители, модули оперативной памяти, сетевые контроллеры, диски и другие. Кратное увеличение количества компонент системы естественным образом ведет к экспоненциальному увеличению вероятности отказа одного из них. Уже сейчас для вычислительной системы время наработки на отказ сравнимо со временем проведения одного расчета, в связи с чем основной стратегией является периодическое сохранение промежуточных результатов (глобальной контрольной точки) на надежное устройство хранения. Однако, в ближайшей перспективе время между отказами компонент суперкомпьютера может стать меньше не только времени проведения всего расчета, но и времени, требуемого на загрузку контрольной точки, проведения части вычислений и сохранения следующей контрольной точки. Таким образом, актуальным становится вопрос об обеспечении самой возможности осуществления длительного расчета на перспективных вычислительных системах. «Сами принципы отказоустойчивости надо пересмотреть: задача должна уметь сама себя восстанавливать после сбоя, без вмешательства человека» [2].

Проблема усугубляется тем, что в текущем стандарте MPI 3.0 и его реализациях отсутствуют механизмы позволяющие работать с отказами. Фактически, сегодня, на уровне прикладной программы, невозможна полноценная разработка и тестирование автоматических средств восстановления расчета в случае отказа элемента вычислительной системы. Требуемый функционал не поддерживается доступными базовыми библиотеками, в первую очередь, - библиотеками, обеспечивающими взаимодействие процессов.

Обеспечение отказоустойчивой работы на вычислительных системах больших масштабов связано с развитием в двух направлениях. Первое – повышение надежности системы на аппаратном уровне, дублирование основных управляющих компонентов, поддержка горячей замены отказавшего оборудования и т. п. Второе – разработка программ, использующих новые подходы к организации отказоустойчивых распределенных вычислений. В данной работе рассматриваются три важные задачи второго направления и описываются пути их решения.

Первая задача заключается в модернизации и разработке новых методов, позволяющих снизить накладные расходы, возникающие из-за интенсивного использования узлов ввода-вывода системы для записи глобальной контрольной точки на общее устройство хранения.

В литературе [3 – 10] широко представлены различные техники обеспечения отказоустойчивости, основными из которых являются методы основанные на контрольных точках различные протоколы восстановления и их модификации. Создание контрольных точек связано с накладными расходами, возникающими из-за интенсивного использования узлов ввода-вывода системы при записи на общее устройство хранения, поэтому среднее время сохранения контрольных точек может быть весьма значительным. Например, согласно [3], характерное время сохранения контрольных точек для некоторых систем составляет от 20 до 30 минут. С целью уменьшения затрат на частое создание контрольных точек были разработаны алгоритмы, которые включают в себя дополнительные механизмы сохранения. Перечислим группы подобных алгоритмов. Первая группа алгоритмов [5, 6, 7] использует механизмы регистрирования сообщений, в этом случае после создания контрольной точки посылающий или принимающий процесс записывает свои сообщения в журнал. Вторая группа алгоритмов [10] основана на частом сохранении изменений в контрольных точках, к ним относятся инкрементный, дифференциальный алгоритмы и их модификации. Третья группа алгоритмов [3, 4] предполагает сохранение контрольных точек вычислительных узлов, то есть локальных контрольных точек, на локальные устройства хранения.

Несмотря на наличие значительной теоретической базы, практическая реализация алгоритмов обеспечения отказоустойчивости затруднена отсутствием программно-аппаратных средств поддержки необходимого функционала в доступных языках и библиотеках разработки параллельных программ. Основной широко применимой практической разработкой является специальный модуль BLCR для ядра Линукса [11]. Данный модуль был использован в различных реализациях MPI [12 – 14] в основном для обеспечения отказоустойчивости по координированному протоколу восстановления.

Отдельно стоит отметить подход основанный на использовании односторонних операций обмена. В статье [15] представлены новые протоколы обеспечения отказоустойчивости, в которых, в отличие от известных протоколов восстановления [5, 6, 7], другим образом реализованы процедуры координации работы MPI процессов. Однако на данный момент практическое использование односторонних операциях обмена осложнено необходимостью разработки эффективных функций синхронизации.

Вторая задача заключается в необходимости учета при разработке алгоритмов обеспечения отказоустойчивости возможных отказов в устройствах хранения и в иерархии слоев аппаратного обеспечения в распределенных вычислительных системах. Данная задача обсуждается в работах [15, 16, 17] и обусловлена тем, что наиболее продолжительными в восстановлении являются отказы в устройствах хранения. В работе [15] предложены протоколы учитывающие иерархичность слоев аппаратного обеспечения (ядра на схеме, схемы в узле и т.д.). В работах [16, 17] получены новые закономерности в области отказов больших групп дисков.

Третья задача заключается в разработке методов и средств, обеспечивающих саму возможность изучения и тестирования различных стратегий построения отказоустойчивых параллельных приложений.

Стандарт MPI и его программные реализации играют ключевую роль в развитии распределенных вычислений. Однако в стандарте MPI 3.0 [18] отсутствует описание каких-либо функций связанных с обеспечением отказоустойчивости в вычислительных системах. На устранение этого пробела направлена работа [19], в которой представлено расширение ULFM [20] стандарта MPI. Это расширение предназначено для решения задач идентификации отказов в вычислительной системе, восстановления связи между процессами, исключения из вычислительного поля отказавших процессов. Оно предоставляет пользователю возможность реализовать в программном приложении различные техники отказоустойчивости, в том числе варьировать объем и содержание контрольных точек. Однако, с одной стороны, сроки появления пригодных к эксплуатации реализаций MPI с ULFM расширением в настоящее время не определены, а с другой — отладка и тестирование отказоустойчивых алгоритмов на реальных вычислительных системах сегодня неэффективны в силу, пока ещё, большой длительности интервалов между возникновением сбоев.

Для решения первой и второй задач в настоящей работе осуществляется разработка методов проведения длительных расчетов на подверженных сбоям вычислительных системах путем создания механизмов поддержания локальных контрольных точек – распределенных промежуточных данных. Соответствующие данные, в случае выхода из строя ряда вычислительных узлов, должны обеспечивать возможность продолжения расчета, ценой повторения части выполненных вычислений. Похожие подходы развиваются в НИИСИ РАН, в РФЯЦ-ВНИИЭФ и в работах [3, 4, 15].

Для решения третьей задачи разрабатывается среда, моделирующая, как сами отказы, так и работу функций расширения ULFM для стандарта MPI, обеспечивающих идентификацию виртуальных отказов. Использование спецификации ULFM вызвана естественным стремлением сохранить будущую совместимость с разрабатываемым стандартом MPI 3.1. Обработка отказов в ULFM основывается на следующих двух положениях [20].

В случае отказа одного из процессов:

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

MPI исключение отражает только локальное воздействие отказа на операцию, и нет никаких гарантий, что другие процессы также будут уведомлены о возникновении этого отказа. Асинхронное распространение информации о возникновении отказа не гарантируется, и пользователи должны проявлять осторожность при рассуждении о множестве процессов, для которых будет зафиксирован отказ и возникнет MPI исключение. Обработка исключений и реализация различных техник восстановления осуществляется, в том числе с помощью функций [20]:

- MPI_COMM_REVOKE – прекращает все текущие нелокальные операции на коммуникаторе, отмечает коммуникатор в качестве аннулированного и все последующие вызовы нелокальных функций должны завершаться значением MPI_ERR_REVOKED, за исключением функций MPI_COMM_SHRINK и MPI_COMM_AGREE.
- MPI_COMM_SHRINK – создает новый коммуникатор на основе существующего, который не содержит отказавшие процессы. Функция осуществляет это путем создания дублирующего коммуникатора, пропуская 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 – вычисляет конъюнкцию значений всех процессов, считая что отказавшие процессы принимают значение ложь.

В нашем подходе осуществляются модификации MPI функций обмена, в результате которых функции могут возвращать исключения для нормально функционирующей распределенной вычислительной системы, тем самым создавая виртуальный отказ в системе. В рамках работы, для тестирования различных техник отказоустойчивости, функции MPI_COMM_REVOKE, MPI_COMM_SHRINK, MPI_COMM_FAILURE_ACK,

MPI_COMM_FAILURE_GET_ACKED, MPI_COMM_AGREE реализуются в усеченном виде с помощью существующих MPI функций.

Разрабатываемая среда выполнения параллельных приложений позволит тестировать различные алгоритмы восстановления вычислений, в том числе, с использованием локальных устройств хранения, путём моделирования сбоев вычислительных узлов на штатно функционирующих многопроцессорных вычислительных системах, что существенно упростит разработку и отладку соответствующих приложений.

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

ЛИТЕРАТУРА:

1. P.M. Kogge "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems," // Univ. of Notre Dame, CSE Dept. Tech. Report TR-2008-13, Sept. 28, 2008
2. Донгарра Дж. Экзафлопсное будущее суперкомпьютеров // Суперкомпьютеры. №1(1). 2010. – с. 21–23.
3. X. Dong, N. Muralimanohar, N.P. Jouppi, Y. Xie "A Case Study of Incremental and Background Hybrid In-Memory Checkpointing" // The Exascale Evaluation and Research Techniques Workshop (EXERT) at ASPLOS 2010, March 2010.
4. N.H. Vaidya, "A Case for Two-Level Distributed Recovery Schemes," // SIGMETRICS ’95. Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, 1995, pp. 64–73.
5. E.N. Elnozahy, L. Alvisi, Y. Wang, D.B. Johnson "A Survey of Rollback-Recovery Protocols in Message-Passing Systems" // ACM Computing Surveys, 34(3):375–408, September 2002.
6. Israel Koren, C. Mani Krishna "Fault-Tolerant Systems" // Morgan Kaufmann Publishers Inc., San Francisco, CA, 2007. – 378 pages.
7. Э. Таненбаум, М. Ван Стеен "Распределенные системы: принципы и парадигмы" // СПб: Питер, 2003. – 877 с
8. Ж. Тель "Введение в распределенные алгоритмы" // Пер. С англ. В.А.Захарова. - М.: МЦМНО, 2009. – 616 с.: ил.
9. A. Gainaru, F. Cappello, M. Snir, W. Kramer "Failure prediction for HPC systems and applications: Current situation and open issues" // International Journal of High Performance Computing Applications, August 2013 27: 273-282,
10. Поляков А.Ю., Данекина А.А. Оптимизация времени создания и объёма контрольных точек восстановления параллельных программ // Вестник СибГУТИ. – Новосибирск: СибГУТИ, 2010. – № 2. – С. 87-100.
11. Berkeley Lab Checkpoint/Restart (BLCR) for LINUX [Электронный ресурс] Режим доступа: http://crd.lbl.gov/groups-depts/ftg/projects/current-projects/BLCR/
12. Open MPI: Open Source High Performance Computing [Электронный ресурс] Режим доступа: http://www.open-mpi.org
13. MPICH [Электронный ресурс] Режим доступа: http://www.mpich.org
14. MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE [Электронный ресурс] Режим доступа: http://mvapich.cse.ohio-state.edu
15. 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.
16. B. Schroeder, G.A. Gibson "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?" // Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07), February 13-16, 2007, San Jose, CA.
17. E. Pinheiro, W.D. Weber, L.A. Barroso "Failure trends in a large disk drive population"// In Proc. of the FAST ’07 Conference on File and Storage Technologies, 2007.
18. Message Passing Interface Forum [Электронный ресурс] Режим доступа: http://www.mpi-forum.org/
19. W. Bland, A. Bouteiller, T. Herault, G. Bosilca, J.J. Dongarra, "Post-failure recovery of MPI communication capability: Design and rationale" // International Journal of High Performance Computing Applications August 2013, 27, pp. 244-254
20. ICL Fault Tolerance [Электронный ресурс] Режим доступа: http://fault-tolerance.org/ulfm/ulfm-specification