Моделирование отказов в высокопроизводительных вычислительных системах в рамках стандарта MPI и его расширения ULFM[1]

А.А. Бондаренко, М.В. Якобовский

Институт прикладной математики им. М.В. Келдыша РАН

Рассматривается проблема выполнения длительных расчетов на высокопроизводительных вычислительных системах, компоненты которых подвержены отказам. Для программ, запускаемых на подобных системах, существенным является возможность обработки отказов путём автоматического продолжения расчета на оставшихся работоспособных узлах системы. Возможность обработки отказов предусматривается в разрабатываемом стандарте MPI 3.1. В работе описывается библиотека для тестирования отказоустойчивых алгоритмов, использующих новый функционал стандарта MPI 3.1. Приведены результаты тестовых расчетов.

1. Введение

Для существующих вычислительных систем Петафлопсного уровня производительности и проектируемых систем Экзафлопсного уровня существенное значение имеет показатель среднего времени безотказной работы [1]. Разработка надежных высокопроизводительных систем 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 [2]. В работе [3] были систематизированы исследования в области организации отказоустойчивой работы вычислительных систем с помощью программного и аппаратного обеспечения.

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 на последующем перезапуске системы 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, достигаемая за счет очевидного недостатка - высоких накладных расходов. В контрольную точку записывается все данные, используемые приложением, в том числе данные, которые не являются существенными для восстановления состояния программы.

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. На уровне пользователя также можно контролировать время и частоту создания контрольных точек при выполнении программы, выбирая наиболее удобные моменты для их создания. 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 [1-3]. 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

[1] Работа выполнена при поддержке Российского фонда фундаментальных исследований по гранту 13-01-12073 офи_м

обеспечивать способность восстановления приложения в согласованное состояние, из которого вычисления могут быть продолжены.

Для удовлетворения этих требований ключевым для стандарта MPI является условие [4], 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.

Все эти требования учитываются в расширении ULFM [4, 5] 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 ULFM расширением в настоящее время не определены, а отладка и тестирование отказоустойчивых алгоритмов на реальных вычислительных системах сегодня неэффективны в силу, пока ещё, 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.

2. Стандарт MPI и его расширение ULFM

В данной работе для решения поставленной цели разрабатывается среда, моделирующая, как сами отказы, так и работу обеспечивающих идентификацию виртуальных отказов функций расширения ULFM для стандарта MPI. Использование спецификации ULFM вызвана естественным стремлением сохранить будущую совместимость с разрабатываемым стандартом MPI 3.1. Обработка отказов в ULFM основывается на следующих двух положениях [5].

В случае отказа одного из MPI-процессов:

- MPI-операция, включающая в себя это MPI-процесс, не должна блокироваться бесконечно, а должна быть либо выполнена успешно, либо вызвать MPI-исключение.
- MPI-операция, не включающая в себя этот процесс, должна завершиться 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 выявлении множества процессов, для которых будет зафиксирован отказ и возникнет MPI-исключение. Обработка исключений и реализация различных техник восстановления осуществляется, в том числе с помощью функций [5]:

- 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 – вычисляет конъюнкцию значений всех процессов, считая, что отказавшие процессы принимают значение ложь.

Разработана библиотека, содержащая модифицированные MPI функции обмена. Функции данной библиотеки возвращают исключения, не предусмотренные стандартом MPI 3.0, позволяющие моделировать отказ в нормально функционирующей распределенной вычислительной системе согласно спецификации [5] расширения ULFM. Также в этой библиотеке с помощью функций стандарта MPI 3.0 реализованы функции MPI_COMM_REVOKE, MPI_COMM_SHRINK, MPI_COMM_FAILURE_ACK, MPI_COMM_FAILURE_GET_ACKED, MPI_COMM_AGREE, что позволяет проводить тестирование различных техник отказоустойчивости.

3. Тестовые задачи, основанные на сохранении контрольных точек в оперативную память и в распределенную файловую систему

В данной работе были разработаны три программы. В первой программе реализован параллельный алгоритм решения задачи о распространении тепла в однородном стержне. Во второй и в третьей программах реализован тот же алгоритм, дополненный средствами моделирования отказов и автоматического восстановления вычислений.

Для второй и третьей программы все множество MPI-процессов делится на рабочие процессы, формирующие рабочее поле, в котором выполняется основной алгоритм прикладной программы, и на дополнительные 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 дополнительных процессов. Предполагаем, что сохранение контрольных точек проходит по координированному протоколу. Надо обратить внимание на то, что для MPI-процессов, введенных в рабочее поле, может быть недоступна информация о контрольных точках отказавших MPI-процессов.

Во второй программе сохранение контрольных точек осуществляется в оперативную память вычислительных узлов по схеме сохранения описанной в работе [6]. В третьей программе сохранение локальных контрольных точек осуществляется в распределенную файловую систему. Отметим, что для возможности восстановления расчетов во второй программе необходимо дублирование локальных контрольных точек, что реализуется с помощью специальной схемы сохранения. А для третьей программы дублирование не требуется, так как через распределенную файловую систему все локальные контрольные точки непосредственно доступны новым MPI-процессам, введенным в расчетное поле.

Из выше сказанного 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 координированное сохранение контрольных точек через одинаковое число итераций основного цикла. Подобный эксперимент позволяет оценить объём накладных расходов для организации отказоустойчивости, а именно временные затраты на сохранение контрольных точек и на восстановление системы после отказа.

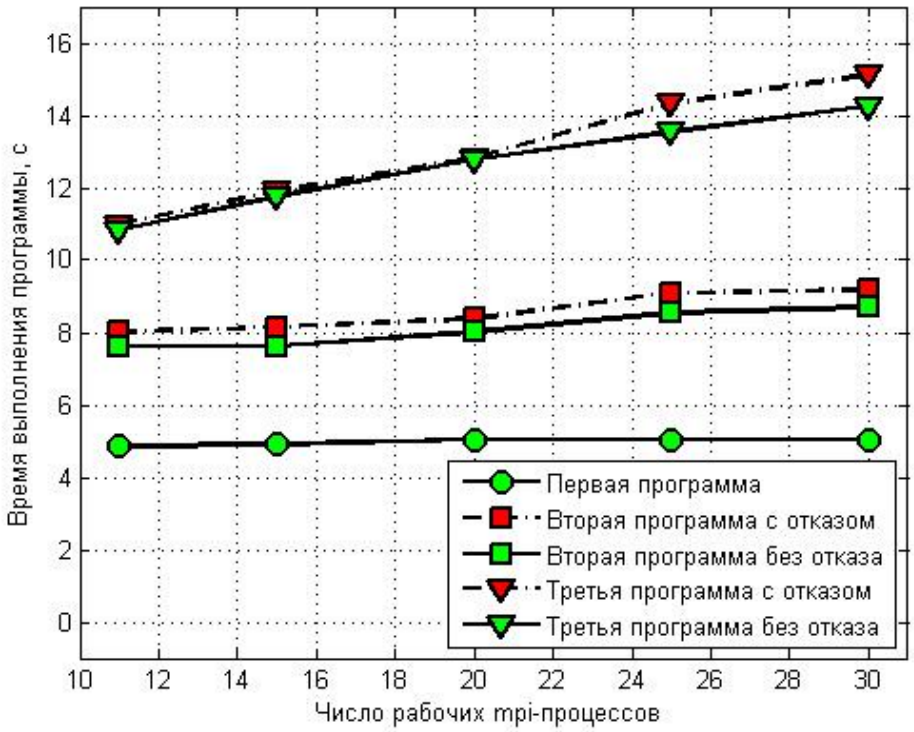

Рис. 1. Сравнение работы тестовых программ, реализующие разные техники отказоустойчивости

Параметры задачи были выбраны таким образом, чтобы размер контрольной точки был порядка 2 Мб. Вычисления проводились на научном кластере ИПМ [7]. Отметим, что уже на небольшом (до 30) числе MPI-процессов, накладные расходы на организацию отказоустойчивости, при сохранение контрольных точек в распределенную файловую систему, превышают соответствующие накладные расходы, при сохранении контрольных точек в оперативную память 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 [6]), - порядка 2,5 секунд. Отметим, что при больших объемах локальных контрольных точек сохранение в оперативную память вычислительных узлов по описанной в [6] схеме может и не иметь смысла в силу ограничения объема оперативной памяти для узла системы. 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.

4. Заключение

Для тестирования и определения эффективности разрабатываемых техник отказоустойчивости необходимо создание соответствующих программных средств. В данной работе разрабатывается 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 [7] 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 и на восстановление системы после отказа относительно малы.

Литература

1. Egwutuoha I.P., Levy D., Selic B., Chen S. A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems // The Journal of Supercomputing. 2013. Vol. 65, No. 3. P. 1302–1326.
2. 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.
3. Liberty D., et al. Addressing failures in exascale computing // International Journal of High Performance Computing ApplicationsMay 2014 Vol. 28, No. 2. P. 129-173.
4. 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.
5. ICL Fault Tolerance URL: http://fault-tolerance.org/ulfm/ulfm-specification (дата обращения: 29.10.2014).
6. Бондаренко А.А., Якобовский М.В. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». 2014. Том 3. № 3 С. 20-36.
7. Scientific Cluster of Keldysh Institute of Applied Mathematics RAS. URL: http://imm6.keldysh.ru/~informer/(дата обращения: 29.10.2014).