Оптимальное сохранение контрольных точек на локальные устройства хранения*

А.А. Бондаренко, М.В. Якобовский

ИПМ им. М.В. Келдыша РАН

Рассматривается техника обеспечения отказоустойчивости, основанная на сохранении контрольных точек на локальные устройства хранения. В работе предлагается разделение отказов на два вида: легкие и тяжелые отказы. Для данного разделения отказов получено время оптимального периода сохранения контрольных точек. Для рассмотренных примеров характерно, что в случае среднего времени между отказами менее часа использование средств разделения на легкие и тяжелые отказы позволяет сократить время вычислений более чем на 10%.

Введение

Для обеспечения отказоустойчивости в высокопроизводительных вычислительных системах наиболее используемыми являются методы восстановления из контрольных точек [1], в частности, основанные на координированном протоколе, когда периодически состояние приложения (вычислительной системы) записывается в надежное устройство хранения. Если возникает отказ, то записанные данные используются для восстановления системы и/или приложения к предыдущему “хорошему” состоянию. Однако для будущих экзафлопсных вычислительных систем применение подобной техники сталкивается с рядом трудностей [2], к ним относят высокую частоту отказов и большие накладные расходы при сохранении контрольных точек. В результате нескольких независимых исследований [3,4, 5] было предсказано, что потенциальные экзафлопсные вычислительные системы будут тратить более 50% своего времени на чтение или запись контрольных точек. Такие опасения побудили исследователей к разработкам новых методов, в том числе повышающих масштабируемость техник сохранения контрольных точек.

Один из подходов, уменьшающих накладные расходы работы с контрольными точками, состоит в том, чтобы предоставить пользователю возможность самому реализовать в приложении различные техники отказоустойчивости, в том числе варьировать объем и содержание контрольных точек. На данный момент стандарт MPI, позволяющий реализовывать методы обеспечения отказоустойчивости на уровне пользователя, находится в разработке [6]. Программная реализация этого стандарта уже разрабатывается в рамках проекта Open-MPI [7]. Однако сроки появления пригодных к эксплуатации реализаций нового стандарта MPI в настоящее время не определены.

В данной работе рассматривается применение метода сохранения контрольных точек на локальные устройства хранения на уровне пользователя согласно [8]. Авторов работы интересует эффективность рассматриваемого метода сохранения контрольных, если будут доступны средства работы с легкими и тяжелыми отказами (определение ниже). Общая задача исследования заключается в определении границ применимости метода сохранения контрольных точек при работе на экзафлопсных вычислительных системах. Эта сложная задача, включающая в себя моделирование работы текущих и будущих вычислительных систем. Цель данной работы заключается в определении оптимального периода сохранения контрольных точек на локальные устройства хранения.

В первой части статьи описываются рассматриваемые виды отказов в вычислительных системах. Во второй части приведена модель сохранения контрольных точек и определен оптимальный период сохранения контрольных точек в рамках рассматриваемых видов отказов.

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

1. Легкие и тяжелые отказы

Пусть MPI-процессы координированно сохраняют контрольные точки в локальную память, которой может быть оперативная память, HDD или SSD диск. Тогда, при отказе хотя бы одного вычислительного узла, MPI-процессы, запущенные на других узлах, не смогут получить доступ к контрольным точкам, расположенным в локальной памяти отказавшего. Таким образом, восстановление процесса вычислений будет невозможно. Для выхода из такого положения следует обеспечить избыточность хранения локальных контрольных точек в системе за счет их дублирования в памяти различных вычислительных узлов. Такое дублирование может быть организованно, например, согласно схема сохранение описанной в [8]. В рамках данной схемы для каждого MPI-процесса определяются номера MPI-процессов, в локальную память которых должны быть сохранены копии исходных контрольных точек.

Значение сохранения контрольных точек в локальные устройства хранения отражает прогноз работ [9, 10]: «… более чем 83% отказов в системах петафлопсного уровня могут быть восстановлены с использованием локальных контрольных точек, в то время как оставшиеся 17%, включающие сложные ошибки или потерю вычислительного узла, требуют использования доступной согласованной глобальной контрольной точки».

Проведем разделении отказов на легкие и тяжелые, в зависимости от расположения данных необходимых для восстановления. Данное разделение возможно как для схемы сохранения контрольных точек представленной в [8], так и для двухуровневой схемы сохранения контрольных точек [11].

- После легкого отказа для восстановления расчетов каждому MPI-процессу достаточно данных записанных на соответствующем ему локальном устройстве хранения. Отметим, что к легким отказам, можно отнести, кратковременные сбои, сбои в коммуникационном оборудовании, которые могут быть решены, например, его перезагрузкой.
- После тяжелого отказа для восстановления расчетов некоторым MPI-процессам понадобятся данные записанные на недоступных ему локальных устройствах хранения. В этом случае эти MPI-процессы должны осуществить запрос необходимых данных у соответствующих MPI-процессов, через стандартный протокол обмена данных. К тяжелым отказам относятся сбои в локальных устройствах хранения, сбои в центральных процессорах и прочее.

Самая простая и применяемая в практике модель для описания вероятности отказов механического и электрического оборудования – экспоненциальное распределение [12]. Для оборудования должно быть известно μ – среднее время между отказами, тогда функция распределения примет вид

$$f(t) = \frac{1}{\mu} e^{-t/\mu}$$

Отметим, что среднее время между легкими и тяжелыми отказами будут различаться. Так, например, средняя продолжительность безотказной работы сокета составляет около 50 лет [13], а одного процессора около 125 лет [14].

2. Оптимизация периода сохранения контрольных точек

В работе [14] предложена модель описывающая сохранение контрольных точек по координированному протоколу, причем контрольные точки сохраняются через одинаковые интервалы, после того как фиксированная часть работы будет сделана. Отметим, что место сохранения контрольных точек, распределенная файловая система или локальные устройства хранения, является не существенным для модели и отражается лишь на значении параметров модели.

Приведем описание модели представленной в [14]. Весь расчет разбит на периоды продолжительностью T, каждый из них включает в себя сохранение контрольной точки продолжительностью C. Для учета асинхронного сохранения контрольных точек, введен параметр ω, $0 \leq \omega \leq 1$. Примем, что на протяжении сохранения контрольной точки длительностью C, проводятся вычисления в объеме ωC. Таким образом, объем вычислений $(1-\omega)C$ будет потерян на организацию сохранения контрольной точки. Значение $\omega = 0$ соответствует сохранению с

блокировкой, а $\omega = 1$ означает, что процесс сохранения выполняется одновременно с вычислениями.

При возникновении отказа необходимо произвести восстановление системы, а именно, определение функционирующих элементов системы, перезагрузку некоторых элементов системы, восстановление параллельной среды выполнения программы и прочее, обозначим D – время необходимое на эти действия. После этого каждый MPI-процесс должен прочитать и произвести восстановление расчетов из соответствующей части глобальной контрольной точки, что займет время R.

В работе [14] представлен вывод общего времени выполнения приложения. Используя этот результат, представим общее время выполнения приложения на вычислительной системе, в которой аппаратные и программные средства позволяют различать и обрабатывать как легкие, так и тяжелые отказы. Пусть время выполнения параллельного приложения τ_{base} – время без учета накладных расходов на реализацию методов отказоустойчивости и на обработку самих отказов, а τ_{final} – общее время выполнения приложения с учетом времени на сохранение контрольных точек и обработку отказов. Представим τ_{final} как сумму τ_{ff} – времени выполнения приложения с учетом сохранения контрольных точек и τ_{fails} – времени потерь на обработку отказов.

Для каждого периода T вычисления занимают $T - C$, а также во время сохранения контрольной точки выполняется вычисления ωC. То есть общий объем работы в период T равен $T - C + \omega C = T - (1 - \omega)C$. Таким образом, получаем зависимость τ_{ff} от τ_{base}.

$$\tau_{ff} = \tau_{base} \frac{T}{T-(1-\omega)C}.$$

Пусть средние времена между двумя отказами составляют величины μ_i, где $i = 1$ соответствует легкому отказу, $i = 2$ соответствует тяжелому отказу. Тогда среднее число отказов каждого вида во время вычислений равно $\frac{\tau_{final}}{\mu_i}$. Для каждого отказа необходимо учитывать D_i и R_i. Также надо учитывать время на выполнение работы ωC, которая была уже сделана в процессе предыдущей итерации сохранения контрольных точек, но не была сохранена в последнюю контрольную точку.

Считаем, что с вероятностью $\frac{T-C}{T}$ отказ произойдет во время вычислений (не сохранения контрольной точки) и потери времени при этом в среднем составят $A = \frac{T-C}{2}$. А с вероятностью $\frac{C}{T}$ отказ произойдет во время сохранения контрольной точки и потери времени при этом в среднем составят $B = T - C + \frac{C}{2}$. Мы пренебрегаем вероятностью того, что отказ произойдет во время восстановления системы после другого отказа.

Для каждого отказа потери составят

$$D_i + R_i + \omega C + \frac{T-C}{T} A + \frac{C}{T} B = D_i + R_i + \omega C + \frac{T}{2}.$$

Таким образом, получаем

$$\tau_{fails} = \frac{\tau_{final}}{\mu_1 \mu_2} \left((D_1 + R_1)\mu_2 + (D_2 + R_2)\mu_1 + \left(\omega C + \frac{T}{2}\right)(\mu_1 + \mu_2) \right).$$

Общее время выполнения приложения составит

$$\begin{aligned} \tau_{final} &= \tau_{ff} + \tau_{fails} = \\ &= \tau_{base} \frac{T}{T-(1-\omega)C} + \frac{\tau_{final}}{\mu_1 \mu_2} \left((D_1 + R_1)\mu_2 + (D_2 + R_2)\mu_1 + \left(\omega C + \frac{T}{2}\right)(\mu_1 + \mu_2) \right) = \\ &= \tau_{base} \frac{T}{(T-(1-\omega)C)\left(1 - \frac{1}{\mu_1 \mu_2}\left((D_1+R_1)\mu_2 + (D_2+R_2)\mu_1 + \left(\omega C + \frac{T}{2}\right)(\mu_1+\mu_2)\right)\right)} \end{aligned} \quad (1)$$

Из этого уравнения получаем оптимальное значение для периода сохранения контрольных точек

$$T_{period}^{opt} = \sqrt{2(1-\omega)C \left(\frac{\mu_1 \mu_2}{\mu_1 + \mu_{2_1}} - \omega C - \frac{(D_1+R_1)\mu_2 + (D_2+R_2)\mu_1}{\mu_1 + \mu_{2_1}} \right)} \quad (2)$$

3. Оценка времени выполнения программ в среде с внедренными средствами обеспечения отказоустойчивости

Проведем сравнение времени выполнения программы в среде, учитывающей легкие и тяжелые отказы, и среде с однородными отказами. В работе [14] для случая однородных отказов приведены следующие формулы

$$T_{period}^{opt} = \sqrt{2(1-\omega)C(\mu - (D + R + \omega C))} \quad (3)$$

$$\tau_{final} = \tau_{base} \frac{T}{(T - (1-\omega)C)\left(1 - \frac{D + R + \omega C}{\mu} - \frac{T}{2\mu}\right)} \quad (4)$$

Пусть есть две программы и время их выполнения без средств обеспечения отказоустойчивости составляет 12 и 24 часа. Таким образом, $\tau_{base} = 720$мин, 1440 мин. Согласно работе [14] примем следующие значения параметров: $C = 10$ мин, $D = D_1 = D_2 = 1$ мин.

Время восстановления из контрольных точек после тяжелого отказа примем равным $R_2 = 10$ мин. Аналогично время восстановления для однородных отказов: $R = 10$ мин.

Время восстановления из контрольных точек после легкого отказа должно быть меньше, чем после тяжелого отказа, мы будем рассматривать два значения $R_1 = 1$ мин и $R_1 = 5$ мин. Пусть $\omega = 0.5$, а среднее время между отказами будет принимать значение от 0.5 часа до 3 часов.

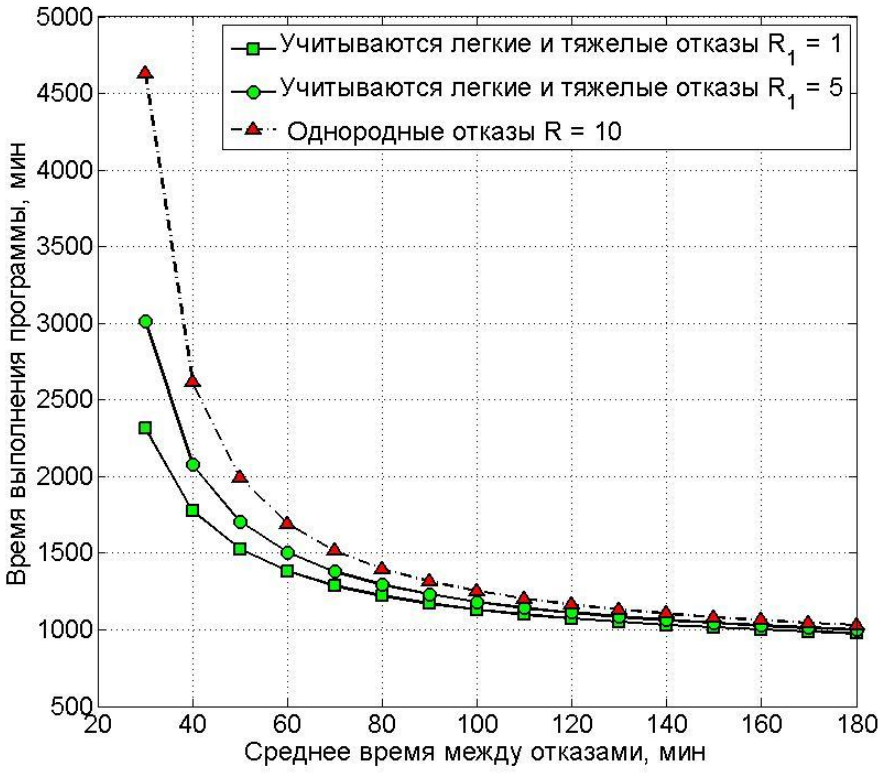

Рис.1. Оценка время выполнения программы в среде поддерживающей обеспечение отказоустойчивости. Параметры: $\tau_{base} = 720$ мин, $C = 10$ мин, $D_1 = D_2 = 1$ мин, $R_2 = 10$ мин, $\omega = 0.5$

Согласно формулам (1-4) вычислим время выполнения программ. Результаты представлены на рисунках 1 и 2. Из них следует, что использование средств обеспечения отказоустойчивости, поддерживающих разделение отказов на легкие и тяжелые, позволит существенно сократить время работы программ особенно для частых отказов. В данной работе рассмотрен уз-

кий диапазон значений параметров характеризующих вычислительные системы и средства обеспечения отказоустойчивости. Более подробная оценка значений параметров будущих вычислительных систем и анализ границ применимости данного подхода является задачей будущих исследований.

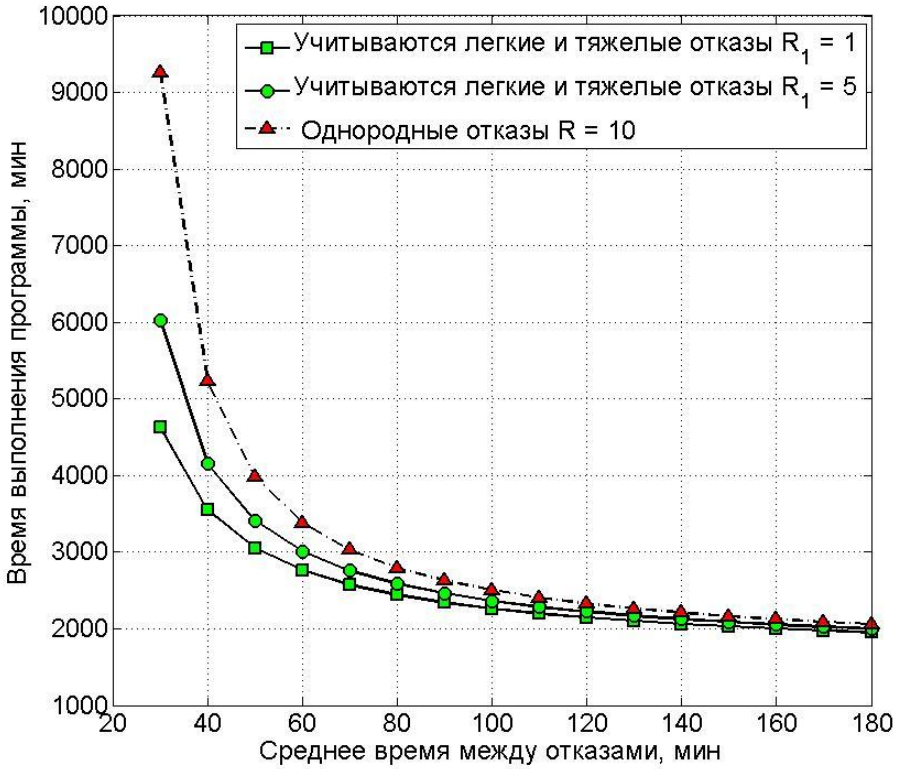

Рис. 2. Оценка время выполнения программы в среде поддерживающей обеспечение отказоустойчивости. Параметры: $\tau_{base} = 1440$ мин, $C = 10$ мин, $D_1 = D_2 = 1$ мин, $R_2 = 10$ мин, $\omega = 0.5$

Заключение

Для системы позволяющей работать с легкими и тяжелыми отказами и для приложений, использующих схему сохранения контрольных точек на локальные устройства хранения, получены формулы полного времени выполнения приложения и времени уходящего на работу с отказами. Для некоторого диапазона значений параметров роботы вычислительных систем получено, что использование средств обеспечения отказоустойчивости, поддерживающих работу с легкими и тяжелыми отказами, позволит существенно сократить время работы приложения. Для рассмотренных примеров характерно, что в случае среднего времени между отказами менее часа использование средств разделения на легкие и тяжелые отказы позволяет сократить время вычислений более чем на 10%.

Основным направлением дальнейших исследований является определение границ применимости техники сохранения контрольных точек на локальные устройства хранения экзафлопсных вычислительных систем. Подобные оценки границ применимости метода, позволят выявить необходимость или отсутствие таковой в развитии рассматриваемого метода и разработки аппаратных и программных средств позволяющих его реализовать.

Литература

1. Elnozahy E. N. M., Alvisi L., Wang Y.-M., Johnson D. B. A survey of rollback-recovery protocols in message-passing systems. ACM Comput. Surv. 2002. Vol. 34, No 3, P. 375–408.

2. Kogge P.M. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems — Tech. Report TR-2008-13. — Univ. of Notre Dame, CSE Dept. — 2008. / P.M. Kogge, et al. URL: http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf (accessed: 25.07.2014).
3. 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.
4. Oldfield R. A., Arunagiri S., Teller P. J., Seelam S., Varela M. R., Riesen R., Roth P.C. Modeling the impact of checkpoints on next-generation systems. In 24th IEEE Conference on Mass Storage Systems and Technologies. Sept. 2007, pp. 30–46.
5. Schroeder B., Gibson G. A. Understanding failures in petascale computers. Journal of Physics: Conference Series. 2007. Vol. 78, No 1.
6. Fault Tolerance Research Hub [Электронный ресурс] Режим доступа: http://fault-tolerance.org/ulfm/ulfm-specification (дата обращения: 1.06.2015).
7. Fault Tolerance Research Hub [Электронный ресурс] Режим доступа: http://fault-tolerance.org/2014/11/15/tutorial-sc14-fault-tolerance-for-hpc-theory-and-practice/ (дата обращения: 1.06.2015).
8. 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, No. 3. С. 20–36.
9. Dong X., Muralimanohar N., Jouppi N., Xie Y., A Case Study of Incremental and Background Hybrid In-Memory Checkpointing // Proceedings of the 2010 Exascale Evaluation and Research Techniques Workshop (Pittsburgh, PA, USA March 13 – 14, 2010), ACM, 2010. P. 119–147.
10. Dong X., Muralimanohar N., Jouppi N., Kaufmann R., Xie Y., Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exscale systems // Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (Portland, Oregon USA November 14-20, 2009). ACM, 2009. P. 57-68.
11. Vaidya, N.H. A Case for Two-Level Distributed Recovery Schemes // Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems (Ottawa, Canada, May 15-19 1995) ACM, 1995. P. 64–73.
12. Dongarra J. Herault T. Robert Y. Fault tolerance techniques for high-performance computing. http://www.netlib.org/lapack/lawnspdf/lawn289.pdf (дата обращения: 1.06.2015).
13. Ferreira K., Stearley J., Laros J. H. I., Oldfield R., Pedretti K., Brightwell R., Riesen R., Bridges P. G., Arnold D.. Evaluating the Viability of Process Replication Reliability for Exascale Systems. In Proc. of the ACM/IEEE SC Conf., 2011
14. 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.

Optimal checkpointing to the local storage device

Aleksey Bondarenko and Mikhail Iakobovski

Keywords: Parallel computation, checkpoints, fault tolerance

We consider the fault tolerance technique based on saving checkpoint files on the local node. We are proposing a division of failures into two kinds: light and heavy failures. For this separation we obtain the optimal checkpoint interval. Examples show that if MTBF is less than an hour then tools working with light and heavy failures reduce the computation time by more than 10%.