Подходы к дельта-оптимизации контрольных точек восстановления параллельных программ*

А.Ю. Поляков
ИФП СО РАН
e-mail: artpol84@gmail.com

О.В. Молдованова
ГОУ ВПО «СибГУТИ»
e-mail: ovm@csc.sibsutis.ru

Б.И. Карасев
e-mail: karasev_b@ngs.ru

Рассмотрены подходы к дельта-оптимизации контрольных точек (КТ) восстановления параллельных программ, выполняемых на распределенных вычислительных системах, по объему и времени их создания. Разработан адаптивный алгоритм дельта-сжатия КТ. Создан комбинированный алгоритм сжатия, обеспечивающий субоптимальное время формирования исходной КТ. Предложен параллельный алгоритм формирования исходной КТ из набора дельта-сжатых, который выполняет поиск наиболее позднего целостного состояния параллельной программы.

Введение

Распределенные вычислительные системы (ВС) являются важнейшим инструментом решения сложных научных, инженерных и экономических задач [1]. Современные ВС большемасштабны, количество процессорных ядер в их составе варьируется от десятков до сотен тысяч, а число узлов ввода-вывода (УВВ) – от нескольких десятков до сотен. Например, ВС IBM Roadrunner состоит из 122 400 процессорных ядер и 216 УВВ, а система Cray XT5 Jaguar – из 224 162 процессорных ядер и 256 УВВ. Физически несколько процессорных ядер обычно размещаются на одном вычислительном узле (ВУ).

Количество трудоемких задач, решаемых на большемасштабных ВС, постоянно увеличивается, при этом реализация некоторых из них требует миллионы часов процессорного времени. Статистические данные [2] показывают, что среднее время (λ^{-1}) безотказной работы вычислительных узлов распределенных ВС находится в промежутке 10^4 – 10^6 ч. (λ – интенсивность потока отказов для одного ВУ). Но даже при использовании таких высоконадежных компонентов в распределенных ВС среднее время (μ^{-1}) между частичными отказами ВС, т. е. отказами одного или нескольких ВУ, в среднем составляет несколько дней [2, 3]. Таким образом, вероятность потери результатов вычислений, полученных параллельной программой (ПП), возрастает с увеличением продолжительности использования ею ресурсов ВС.

Одним из важных элементов в комплексе мер по обеспечению отказоустойчивости распределенных ВС является организация отказоустойчивого выполнения параллель-

*Работа выполнена при поддержке Совета по грантам Президента РФ (ведущая научная школа НШ 5176.2010.9) и Российского фонда фундаментальных исследований (гранты 10-07-00157, 09-0700185, 09-07-00095).

ных программ. Данный подход предусматривает сохранение промежуточных состояний ПП в контрольных точках (КТ). В случае отказа ресурсов ВС любая доступная КТ позволяет перезапустить (возобновить) исходную программу в состоянии, которое не требует повтора вычислений, завершенных к моменту создания данной КТ.

Контрольная точка, созданная для параллельной программы, является распределенной и представляет собой набор локальных КТ, хранящих состояния процессов этой программы. Распределенная КТ называется целостной, если она позволяет сформировать допустимое состояние ПП [4].

Существует два основных подхода к созданию распределенных КТ [4]: синхронный (координированный) и асинхронный. При синхронном подходе все процессы ПП сохраняют свое состояние одновременно, при этом процесс создания распределенной КТ разбит на несколько этапов, между которыми происходит глобальная синхронизация. Это позволяет обеспечить целостность КТ, однако вызывает высокую нагрузку на подсистему ввода-вывода распределенной ВС. При асинхронном подходе каждый процесс создает КТ независимо от других, что позволяет обеспечить равномерную нагрузку на подсистему ввода-вывода. Но при возобновлении параллельной программы требуется выполнять поиск ее целостного состояния на основе набора локальных КТ, созданных в различные моменты времени. При этом возможно возникновение «эффекта домино», когда наиболее поздним целостным состоянием программы оказывается ее начальное состояние.

Анализ существующих средств отказоустойчивого выполнения параллельных программ показывает, что в большинстве из них для создания распределенных КТ используется синхронный подход [5, 6]. Таким образом, актуальной является задача снижения вышеупомянутых накладных расходов, связанных с нагрузкой на подсистему ввода-вывода ВС. В работе предложен подход к решению данной задачи, предусматривающий уменьшение объема КТ и, как следствие, снижение времени их создания.

Адаптивный алгоритм дельта-оптимизации контрольных точек

Методы и алгоритмы, реализуемые в параллельных программах, характеризуются различными шаблонами использования памяти. Например, многие итерационные методы лишь частично модифицируют обрабатываемые данные в процессе работы. Набор $\Lambda = \{\Lambda_1, \Lambda_2, ..., \Lambda_T\}$ контрольных точек, созданных для таких программ в моменты времени $t = \{1, 2, ..., T\}$, будет содержать дублирующуюся информацию. Данное свойство позволяет производить дельта-оптимизацию КТ набора Λ с использованием технологии дельта-сжатия [7, 8, 9]. Эта технология предполагает создание сжатой КТ $\Sigma_t = \Delta(\Lambda_t, \Lambda_b)$, содержащей только те фрагменты КТ Λ_t, которые были модифицированы относительно Λ_b, где $b < t,\ t, b \in \{1, 2, ..., \infty\}$, а $\Delta(X_1, X_2)$ – функция, выполняющая дельта-сжатие структуры данных X_1 относительно X_2. Контрольная точка Λ_t может быть восстановлена с использованием обратной функции Δ^{-1}: $\Lambda_t = \Delta^{-1}(\Sigma_t, \Lambda_b)$.

Выбор КТ Λ_b, относительно которой выполняется дельта-оптимизация, в значительной мере определяет эффективность дельта-сжатия. Наиболее распространенным видом дельта-сжатия КТ является инкрементное, которое предполагает сжатие любой КТ относительно предыдущей, т. е. $\forall t, b = (t–1)$. В современных системах резервного копирования используется также дифференциальное дельта-сжатие, которое предусматривает сжатие любой КТ относительно некоторой базовой (обычно первой: $b = 1$).

Каждый из описанных видов дельта-оптимизации имеет свои преимущества и недостатки. При дифференциальном дельта-сжатии для формирования результирующей

КТ Λ_t необходима базовая КТ Λ_1 и только одна дельта-сжатая КТ Σ_t . Это позволяет сокращать объемы хранимых данных за счет удаления устаревших дельта-сжатых КТ. С другой стороны для дифференциальных КТ характерно постоянное увеличение их объема. Инкрементные КТ по сравнению с дифференциальными формируются относительно более актуального состояния программы и, в большинстве случаев, будут иметь меньший объем. Недостатком данного подхода является необходимостью хранить все созданные КТ:

$$\Lambda_t \;=\; \Delta^{-1}(\Sigma_t, \Delta^{-1}(\Sigma_{t-1}, \Delta^{-1}(\ldots, \Delta^{-1}(\Sigma_2, \Lambda_1)\ldots))). \qquad (1)$$

Очевидно, что рассмотренные виды дельта-сжатия являются предельными случаями на множестве допустимых решений. Поэтому актуальным является поиск промежуточных вариантов, которые будут сочетать в себе преимущества инкрементного и дифференциального дельта-сжатий.

В данной работе предложен адаптивный алгоритм Adaptive Delta-Compression Algorithm (*ADCA*) дельта-оптимизации КТ Λ_t, который осуществляет (суб)оптимальный выбор базовой КТ Λ_b с целью минимизации объема $\Sigma_t = \Delta(\Lambda_t, \Lambda_b)$, а также уменьшения количества КТ, которые потребуются для формирования результирующей КТ. Алгоритм предусматривает дельта-сжатие КТ относительно некоторой базовой КТ Λ_b (изначально $b = 1$). Однако, в отличие от дифференциального дельта-сжатия, базовая КТ не является фиксированной в процессе формирования набора Λ. Если различия в объемах данных, модифицированных относительно базовой КТ Λ_b и предыдущей КТ Λ_{t-1}, превышают некоторое пороговое значение, то базовая КТ изменяется на текущую: $b = t$. На рис. 1 показана эффективность алгоритма *ADCA* при дельта-сжатии КТ, которые формировались для программы *Airbag* моделирования газодинамических процессов в подушке безопасности автомобиля [10].

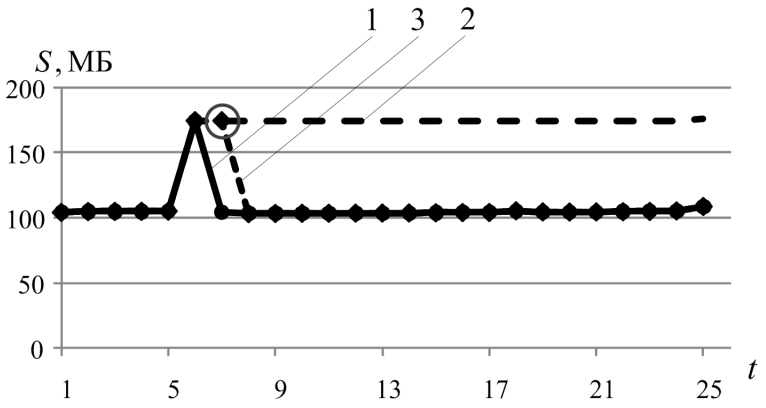

Рис. 1. Эффективность алгоритма *ADCA* для программы *Airbag*: S – объем КТ в МБ; t – номер КТ.

Алгоритм пакетного сжатия контрольных точек

Распространенным подходом к оптимизации КТ по объему является их сжатие универсальными алгоритмами [6, 9], например, одной из модификаций алгоритма Зива–Лемпеля. Этот же подход может быть применен и к блокам данных, обнаруженным в процессе дельта-сжатия. Далее будем называть такой тип оптимизации комбинированным сжатием. Сжатие всех модифицированных блоков как единого массива данных не

является эффективным, потому что при формировании исходной КТ требуется доступ к произвольным блокам, сохраненным в дельта-сжатых КТ. Такой вариант комбинированного сжатия может приводить к необходимости распаковки всей дельта-сжатой КТ, даже если требуется только последний блок.

В данной работе предложен алгоритм *PaComp* комбинированного сжатия, который обеспечивает константный по времени доступ к произвольным блокам дельта-сжатой КТ. Суть алгоритма заключается в том, что набор V_t модифицированных блоков данных КТ Λ_t разбивается на пакеты фиксированного размера, которые сжимаются независимо друг от друга универсальным алгоритмом (например *LZ77*). Так как два набора данных одного размера после применения универсального сжатия с высокой вероятностью будут иметь различный размер, то в КТ также сохраняются смещения (или размеры) каждого сжатого пакета.

Было показано, что эффективность сжатия алгоритмом *PaComp* сопоставима с эффективностью используемого универсального алгоритма (рис. 2). При этом он позволяет существенно снизить время формирования исходной КТ по набору сжатых (рис. 3).

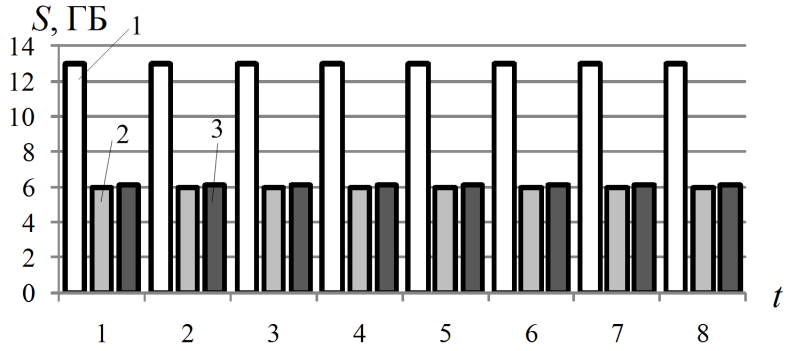

Рис. 2. Объем S распределенной КТ с номером t, созданной для параллельной программы *LAMMPS* с помощью: 1 – алгоритма *ADCA*, 2 – алгоритма *ADCA* с последующим сжатием агоритмом *LZ77*; 3 – алгоритмом *ADCA* с последующим сжатием алгоритмом *PaComp*.

Параллельный алгоритм формирования результирующей КТ

При возобновлении параллельных программ возникает задача формирования исходной КТ Λ_t по первой КТ (Λ_1) и набору сжатых ($\{\Sigma_2, \Sigma_3, ..., \Sigma_t\}$). Для решения данной задачи предложен параллельный алгоритм *P-DCR* (Parallel Delta-Compression Restore), основанный на результатах работы [8] и модифицированный с учетом описанных в данной работе алгоритмов оптимизации КТ.

Формирование результирующей распределенной КТ предусматривает обработку наборов дельта-сжатых КТ, созданных для каждого из процессов ПП. При этом функции по проверке целостности формируемых КТ, а также распаковке пакетов при использовании комбинированного сжатия алгоритмом *PaComp* могут быть эффективно распределены между вычислительными узлами, на которых производится возобновление программы. Для обеспечения отказоустойчивости необходима проверка целостности КТ. Поэтому важной задачей является поиск наиболее позднего целостного состояния ПП.

Алгоритм *P-DCR* формируется из одной корневой и набора рабочих ветвей, количество которых совпадает с числом восстанавливаемых процессов ПП. Каждая ветвь

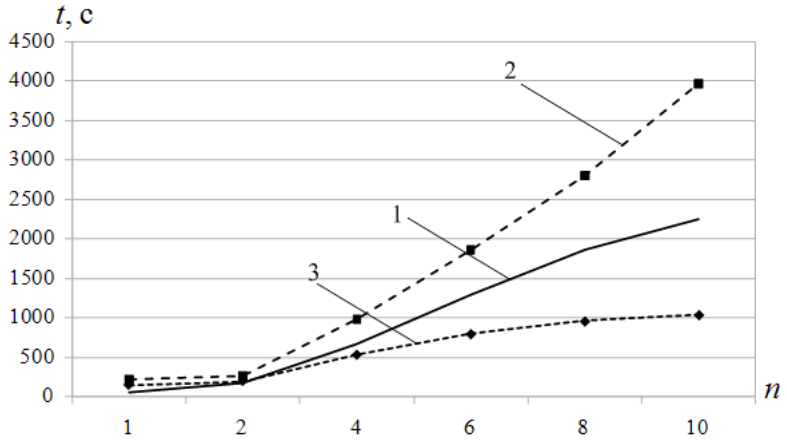

Рис. 3. Зависимость времени t создания результирующей КТ от количества n ветвей параллельной программы *LAMMPS* по набору: 1 – дельта-сжатых КТ; 2 – дельта-сжатых КТ, дополнительного обработанных алгоритмом *LZ77*; 3 – дельта-сжатых КТ, дополнительного обработанных алгоритмом *PaComp*.

выполняется на том ВУ, на котором будет запущен соответствующий процесс.

На первом шаге ($r = 1$) все ветви выполняют восстановление наиболее поздней созданной контрольной точки $\Lambda_{t^r}, t^r = t$, каждая из них проверяет целостность всех необходимых блоков. Если i-я рабочая ветвь ($i \in [1, N]$, где N – количество ветвей *P-DCR*) обнаруживает в КТ $\Sigma_{k_i}, k_i < t'$ поврежденный блок, то она сообщает корневой ветви, что она способна восстановить только КТ Λ_{k_i-1}. Корневая ветвь анализирует полученные ответы рабочих ветвей и генерирует новый номер КТ, которая может быть восстановлена всеми ветвями:

$$r = r + 1, t^r = \max_{i \in [1,N]} k_i. \tag{2}$$

Описанные шаги повторяются до тех пор, пока $t^{r+1} \neq t^r$ и не обнаружено нарушения целостности КТ Λ_1.

Важной особенностью *P-DCR* является то, что на r-том шаге рабочие ветви повторно используют блоки, принадлежащие сжатым КТ $\{\Sigma_i, i = 1, 2, ..., t^r\}$, если они уже были получены на предыдущих шагах $r' < r$ восстановления. Это позволяет значительно уменьшить время формирования результирующей КТ в условиях возможного нарушения целостности.

Заключение

Таким образом, в работе предложены алгоритмы дельта-оптимизации распределенных контрольных точек, хранящих состояние параллельных программ, а именно: 1) адаптивный алгоритм *ADCA*, сочетающий преимущества известных видов дельта-сжатия; 2) параллельный алгоритм *P-DCR* формирования исходной КТ по набору сжатых, который выполняет поиск наиболее позднего целостного состояния параллельной программы; 3) алгоритм *PaComp* комбинированного сжатия, обеспечивающий (суб)минимальное время формирования результирующей КТ.

Созданные алгоритмы реализованы в программном пакете Hash-Based Incremental Checkpointing Tool (HBICT) [11], интегрированном со средством создания КТ Distributed

MultiThreaded CheckPointing (DMTCP). Разработанный пакет является частью системного программного обеспечения пространственно-распределенной мультикластерной вычислительной системы Центра параллельных вычислительных технологий ГОУ ВПО «СибГУТИ» и Лаборатории ВС ИФП СО РАН.

Список литературы

[1] Хорошевский, В.Г. Архитектура и программное обеспечение пространственно-распределённых вычислительных систем [Текст]/ В.Г. Хорошевский, М.Г. Курносов, С.Н. Мамойленко, А.Ю. Поляков // Вестник СибГУТИ. – Новосибирск: СибГУТИ, 2010. – №2. – сс. 112 – 122.

[2] Philp, I. Software failures and the road to a petaflop machine [Текст] / I. Philp // In HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11). – IEEE Computer Society, 2005.

[3] Team, T.B. An overview of the BlueGene/L supercomputer [Текст] / T.B. Team // In Proceedings of SC2002: High Performance Networking and Computing. – Baltimore, MD, Nov. 2002.

[4] Elnozahy, E.N. A survey of rollback-recovery protocols in message-passing systems [Текст] / E.N. Elnozahy, L. Alvisi, Y.M. Wang, D.B. Johnson // ACM Computing Surveys. – 2002. – Vol. 34, N. 3. – P. 375 – 408.

[5] Hursey, J. The design and implementation of checkpoint/restart process fault tolerance for Open MPI [Текст] / J. Hursey, J.M. Squyres, T.I. Mattox, A. Lumsdaine // In Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS). – IEEE Computer Society, 2007. – P. 1 – 8.

[6] Ansel, J. DMTCP: Transparent checkpointing for cluster computations and the desktop [Текст] / J. Ansel, K. Arya, G. Cooperman // Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS'09). – IEEE Press, 2009. – P. 1. – 12.

[7] Kiswany, S.A. stdchk: A checkpoint storage system for desktop grid computing [Текст] / S.A. Kiswany, M. Ripeanu, S.S. Vazhkudai, A. Gharaibeh // Proc. of ICDCS 2008. – Washington: IEEE Computer Society, 2008. – P. 613 – 624. – ISSN 978-0-7695-3172-4.

[8] Sangho, Y.S. Adaptive page-level incremental checkpointing based on expected recovery time [Текст] / S. Sangho, Y.J. Heo, Y. Cho, J. Hong // Proceedings of the 2006 ACM symposium on applied computing. – USA, NY, NewYork: ACM, 2006. – P. 1472 – 1476.

[9] Plank, J.S. Compressed differences: An algorithm for fast incremental checkpointing [Текст] / J.S. Plank, J. Xu, R. Netzer // Technical Report CS-95-302, University of Tennessee, August 1995.

[10] Рычков, А.Д. Моделирование процесса зажигания гранулированного унитарного твердого топлива в камере сгорания айрбэга [Текст] / А.Д. Рычков, Н.Ю. Шокина, Х. Милошевич // Матер. междун. конфер. "Вычислительные и информационные технологии в науке, технике и образовании". – Павлодар, 2006. – Т. 2. – С. 165 – 175.

[11] Сайт проекта HBICT [Электронный ресурс]. – Режим доступа: http://sourceforge.net/projects/hbict/, свободный (дата обращения 14.04.2011 г.).