ISBN 5-94052-127-4 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2006

УДК 519.68

С. М. Абрамов, Г. И. Есин, И. М. Загоровский, Г. А. Матвеев, В. А. Роганов

Принципы организации отказоустойчивых параллельных вычислений для решения вычислительных задач и задач управления в Т-Системе с открытой архитектурой (OpenTS)

Аннотация. В статье рассматриваются принципы организации отказоустойчивых параллельных вычислений. Приводится анализ мер, которые следует предпринять в Т-Системе с целью поддержки отказоустойчивых вычислений на основе модели перевычислений.

Ключевые слова и фразы: Т-Система, OpenTS, вычислительный кластер, отказоустойчивость, язык параллельного программирования.

Введение

В статье используется следующий перечень сокращений и специальных терминов:

- ПО — программное обеспечение;
- MPI — интерфейс передачи сообщений;
- DMPI — динамический интерфейс передачи сообщений;
- OpenTS — Т-система с открытой архитектурой;
- Суперпамять — специальный вид распределенной общей памяти, использованный в реализации OpenTS;
- Т-суперструктура — ортогональная к синтаксису и семантике Т-надстройка над стандартной средой исполнения.

Отказоустойчивость является одним из желательных свойств высокопроизводительных программных комплексов, функционирующих на современных кластерных и метакластерных аппаратных платформах. Это связано прежде всего с высокой вероятностью частичного отказа оборудования, которая тем выше, чем больше общее количество процессорных и коммуникационных компонент в составе вычислительной установки.

Работа выполнена при поддержке Программы фундаментальных научных исследований ОИТВС РАН «Оптимизация вычислительных архитектур под конкретные классы задач, информационная безопасность сетевых технологий» и РФФИ (грант РФФИ № 05–07–08005).

Отказоустойчивость предполагает способность программного обеспечения нормально исполнять свои функции в случае частичной потери мощности вычислительной установки, обусловленной штатными (запланированное обслуживание отдельных стоек и блоков) или внештатными причинами (выход стойки или модуля по причине аварии).

Разумеется, такая способность не является универсальной, изначально присущей стандартным на сегодняшний день программно-аппаратным платформам, таким, например, как большинство современных Linux-кластеров и реализаций MPI.

Безусловно, необходимым требованием для возможности реализации такой способности ПО является использование оборудования, которое не утрачивает своих вычислительных и коммуникационных способностей *полностью* в случае *частичных сбоев*.

Но даже в случае способности аппаратуры к продолжению работы в случае частичных сбоев, от программного обеспечения требуются достаточно неординарные действия, позволяющие произвести динамическую реконфигурацию своих структур и параметров алгоритмов, чтобы продолжить вычисления на ходу, перевычислив ту часть данных, которая была утрачена в результате отказа вышедшего из строя оборудования.

Одному из таких способов динамической реконфигурации, основанному на модели перевычислений, и посвящена данная статья.

1. Общий обзор основных способов обеспечения отказоустойчивости вычислений

В настоящее время известно несколько подходов к реализации отказоустойчивости вычислений.

Рассмотрим кратко некоторые из них:

(1) перекладывание функции обеспечения отказоустойчивости полностью на прикладное ПО;
(2) автоматическая или полуавтоматическая поддержка контрольных точек (checkpointing) прикладной программы и сохранение состояния программы в надежную память;
(3) автоматический повтор вычисления неудачно завершившихся (из-за отказов оборудования) отдельных подпрограмм.

1.1. Перекладывание функции обеспечения отказоустойчивости полностью на прикладное ПО. Данный способ требует от прикладного программиста решать данную задачу самостоятельно, что приводит в некоторых случаях к весьма значительным усилиям и не соответствует современной тенденции автоматизации различных аспектов программирования. С другой стороны, от него стараются не отказываться полностью, поскольку некоторые аспекты состояния программы крайне тяжело автоматически сохранить и восстановить после сбоя. В современных системах для таких случаев предусматриваются возможности явного определения прикладным программистом методов для сохранения и восстановления состояния отдельных (исключительных) объектов, каковыми могут являться, например, элементы пользовательского интерфейса.

Большую же часть работы по сохранению/восстановлению состояния вычислительного процесса стараются выполнять автоматически.

1.2. Автоматическая или полуавтоматическая поддержка контрольных точек (checkpointing) прикладной программы и сохранение состояния программы в надежную память. Этот способ основывается на том факте, что с точки зрения системного ПО все данные прикладной программы состоят из некоторого количества сегментов, находящейся в виртуальной памяти процессов, и их содержимое можно автоматически сохранить (и впоследствии восстановить) практически в любой момент времени исполнения программы (исключения составляют промежутки времени исполнения системных вызовов и некоторые другие «трудные» состояния). Кроме этого, состояние процессов частично содержится в совокупности открытых файловых и прочих системных дескрипторов, информация о которых также является доступной.

В связи с этим оказывается возможным (хотя и ценой немалых усилий) реализация автоматического сохранения практически полного состояния вычислительного процесса и помещения информации об этом состоянии в надежную память с целью восстановления в случае отказа оборудования.

Примерами таких систем, поддерживающими распределенную модель вычислений с передачей данных MPI, может служить разработанная в университете Беркли система BLCR. Успешные эксперименты по комплексированию этой библиотеки с ПО «T-система с

открытой архитектурой» были проведены в последний год суперкомпьютерной программы «СКИФ» Союзного государства [1–5].

Однако такому способу обеспечения отказоустойчивости присущи и определенные недостатки. Прежде всего, полное состояние распределенного вычислительного процесса может иметь значительный объем, сохранение которого во вторичную память может потребовать значительного времени. При этом использовать диски вычислительных модулей не всегда возможно (ибо в случае отказа диск может оказаться недоступным), так что файлы с контрольными точками приходится перемещать. Далее, после отказа количество доступных узлов уменьшается, и, следовательно, для восстановления нужно иметь запас заведомо исправных вычислительных блоков, которые будут вынужденно простаивать в случае исправно работающей аппаратуры.

1.3. Автоматический повтор вычисления неудачно завершившихся (из-за отказов оборудования) отдельных подпрограмм. В большинстве случаев отказ возникает у небольшого количества узлов кластера, при этом вовсе необязательно восстанавливать глобальное состояние распределенного вычислительного процесса из ближайшей контрольной точки, что может привести к длительному отказу в обслуживании в силу неспособности ПО выполнять свои функции пока не завершится полностью фаза отката. Состояние детерминированной неинтерактивной программы — следствие предыдущих состояний. И если сохранять состояние распределенной программы не изохронно, а в те моменты, когда частицы такого состояния компактны, то можно получить значительно более эффективную систему.

В зависимости от модели вычисления этот способ требует различных усилий на свою реализацию. В случае T-системы естественные частицы, порождающие состояние программы — это пренатальные процессы, представляющие собой мобильные замыкания (применения функции к аргументам). Каждая такая частица может путешествовать по кластеру в поисках наименее загруженного процессора, и в мобильном состоянии занимает объем от нескольких сотен байт. Кроме того, если программа написана в функциональном стиле (и не имеет «сторонних эффектов»), то *состояние* вычисления такой программы *можно повторить*, *перевычислив* некоторые функции (а именно, те из них, которые попали на отказавшие узлы и не успели вернуть свои результаты по причине отказа оборудования).

Скорость восстановления в этом случае может быть значительно выше, чем при использовании глобальных контрольных точек. Более того, даже временные характеристики ПО нарушаются только частично: если T-система отрабатывает несколько внешних запросов (работая в режиме вычислительно сервера), то лишь несколько из них могли ожидать результаты «невезучих» пренатальных процессов.

2. Анализ необходимых мер для обеспечения режима отказоустойчивости на основе модели перевычислений

В соответсвии с архитектурой T-системы для обеспечения режима отказоустойчивости на основе модели перевычислений необходимо:

(1) обеспечить возможность автоматического распознавания выхода из строя отдельных узлов и автоматической реконфигурации вычислительного поля и коммуникационного оборудования;
(2) обеспечить возможность автоматической реконфигурации Суперпамяти;
(3) обеспечить возможность автоматического повторного вычисления T-функций при аварии.

На практике это соответствует прежде всего доработке подсистемы DMPI, введение соответствующих механизмов в существующую реализацию Суперпамяти и механизма сохранения пренатальных T-функций на породивших их вычислительном узле.

Кроме этого предполагается доработка существующего механизма Heartbeat с целью выявления *отдельных* отказавших узлов и коммуникационных соединений (в настоящий момент heartbeat — логика обнаруживает только нарушение коммуникационной среды в целом). Также в случае реконфигурации необходима соответствующая реинициализация механизма доски объявлений системы OpenTS.

2.1. Отказоустойчивость на уровне работы с MPI-функциями. В соответствии с требованиями, которые накладывает отказоустойчивый режим вычислений, проведен анализ и выбраны три основных вида обеспечения отказоустойчивости на уровне работы с MPI-функциями:

(1) отделение распределенного коммуникационного процесса от счетных процессов с целью его реконфигурации и автоматического перезапуска в случае отказа оборудования;
(2) реализация DMPI-драйвера поверх отказоустойчивых протоколов, что должно позволить DMPI-подсистеме самостоятельно производить реконфигурацию коммуникационной среды в случае сбоев;
(3) использование специализированных реализаций стандарта MPI, непосредственно поддерживающих режим отказоустойчивых вычислений (таких, например, как FT-MPI).

В ходе дальнейших работ предполагается провести разработку пилотных образцов наиболее перспективных из перечисленных видов обеспечения отказоустойчивости с целью их предварительного тестирования и апробации на реальных задачах, допускающих указанный режим отказоустойчивых вычислений.

К настоящему моменту выполнена разработка и проведены предварительные испытания DMPI-драйвера для протокола TCP/IP.

2.2. Драйвер DMPI для протокола TCP/IP. Предыдущие версии DMPI поддерживали в качестве транспортного уровня только MPI и PVM. Тем самым, нельзя было использовать T-Систему без этих двух библиотек, например, на TCP/IP сети вычислительных машин.

В рамках данной работы, нами разработан и реализован драйвер DMPI, который использует в качестве транспортного уровня стек протоколов TCP/IP.

Тем самым, теперь T-приложения могут без перекомпиляции запускаться как на суперкомпьютерных и метакластерных установках, имеющих в составе программного обеспечения библиотеку MPI, так и на сетях, поддерживающих стандартные протоколы TCP/IP.

Поскольку TCP/IP соединения есть соединения между парами узлов (в отличие от глобального коммуникационного пространства MPI), то имеется возможность для обеспечения продолжения коммуникаций между исправными узлами и возобновления соединений с отказавшими узлами после восстановления их работоспособности.

2.3. Доработка Суперпамяти для поддержки отказоустойчивых вычислений. Для обеспечения возможности автоматической реконфигурации в случае сбоев предполагается ввести в состав ячеек суперпамяти идентификатор контрольной точки. В случае недостижения следующей контрольной точки из-за аварии предполагается реализовать процедуру автоматической реконфигурации, при которой будет производиться освобождение всех помеченных данным идентификатором ячеек и их возвращение в общий список свободных ячеек Суперпамяти.

2.4. Доработка механизма повторного запуска Т-функций в случае сбоев. Необходимо реализовать запоминание Т-функций и их аргументов, а также назначенных для их исполнения вычислительных узлов с целью обеспечения возможности их повторного вычисления на исправных узлах в случае аварии.

Повторный запуск Т-функций, которые относятся к последней контрольной точке, следует производить после реконфигурации коммуникационной подсистемы и Суперпамяти.

Заключение

Тема отказоустойчивости всегда актуальна для высокопроизводительных программных комплексов, функционирующих на современных кластерных и метакластерных аппаратных платформах. Поэтому, дальнейшие исследования в этом направлении всегда будут востребованы.

Нами предполагается провести разработку пилотных образцов наиболее перспективных из выше перечисленных видов обеспечения отказоустойчивости с целью их предварительного тестирования и апробации на реальных задачах.

Список литературы

[1] Суперкомпьютерная программа «СКИФ»: Электронный сетевой ресурс, 2003, `http://skif.pereslavl.ru/`.

[2] Абрамов С. М., Адамович А. И., Инюхин А. В., Московский А. А., Роганов В. А., Шевчук Ю. В., Шевчук Е. В. *Т-система с открытой архитектурой* // Суперкомпьютерные системы и их применение SSA'2004: Труды Международной научной конференции, 26–28 октября 2004 г. Минск, ОИПИ НАН Беларуси. — Минск, 2004, с. 18–22.

[3] Абрамов С. М., Адамович А. И., Коваленко М. Р., Роганов В. А. Биатлон для СКИФов: быстро и точно // Математика, информатика: теория и практика. Сборник трудов, посвященный 10-летию Университета города Переславля / ред.А. К. Айламазян. — Переславль-Залесский: Изд-во «Университет города Переславля», 2003. — ISBN 5–901795–02–4, с. 91–96.

[4] Абрамов С. М., Васенин В. А., Мамчиц Е. Е., Роганов В. А., Слепухин А. Ф. *Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы* // Всероссийская научная конференция «Высокопроизводительные вычисления и их приложения», 30 октября–2 ноября 2000 г., г. Черноголовка: Труды конференции. — М.: Изд-во МГУ, 2000, с. 261–264.

[5] Абрамов С. М., Васенин В. А., Мамчиц Е. Е., Роганов В. А., Слепухин А. Ф. *Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы* // Научная сессия МИФИ–2001, 22–26 января 2001 г.: Сборник научных трудов. — Т. **2**, 2001, с. 234.

Исследовательский центр мультипроцессорных систем ИПС РАН

S. M. Abramov, G. I. Esin, G. A. Matveev, V. A. Roganov, I. M. Zagorovsky. *Principles of organization of the failure tolerance parallel computations for the computational and control tasks in the T-system with the open architecture (OpenTS).* (in Russian.)

Abstract. The article considers principles of organization of the failure tolerant parallel computations and analyses necessary steps for providing failure tolerance on the base of recomputations in the T-System with the open architecture (OpenTS).