УДК 519.68

ЭКСПЕРИМЕНТАЛЬНАЯ РЕАЛИЗАЦИЯ ОТКАЗОУСТОЙЧИВОЙ ВЕРСИИ СИСТЕМЫ OPENTS ДЛЯ ПЛАТФОРМЫ WINDOWS CCS

А.А.Кузнецов, В.А.Роганов
Институт программных систем РАН, Переславль-Залесский

Приводится описание проекта по созданию отказоустойчивой версии системы параллельного программирования OpenTS для платформы Windows Compute Cluster Server. Представлено обоснование необходимости проекта, краткая характеристика системы OpenTS и её подсистемы DMPI, и описаны методы обеспечения отказоустойчивости их функционирования.

1. Введение

Вопрос широкой применимости высокопроизводительных вычислений сегодня становится все более зависимым от решения проблем в области программного обеспечения. На первый план выходят проблемы удобства разработки параллельных приложений, сокращения сроков их разработки, повышение надежности процесса параллельных вычислений (как за счет механизмов контрольных точек, так и за счет других схем организации отказоустойчивых вычислений), совершенствование инструментария разработки параллельных программ.

Отказы оборудования — это одна из наиболее частых причин остановки счета в супервычислительных установках. При этом повышенная надежность оборудования весьма заметно отражается на его цене.

С точки зрения экономической целесообразности, значительно выгоднее наращивать способность дешевых систем к реконфигурированию в случае отказов, нежели создавать «бесконечно» надежные программно-аппаратные комплексы. Это одна из причин, обуславливающих высокую актуальность обеспечения отказоустойчивости вычислений.

В рамках данной работы были разработаны различные инструменты для реализации отказоустойчивых вычислений:

- доработанная в части отказоустойчивости реализация наиболее употребимого подмножества функций MPI — DMPI — дополненная функциями автоматического мониторинга исправной вычислительной конфигурации сильно- либо слабосвязанного множества вычислительных узлов. Такая реализация позволяет в каждом конкретном случае наиболее эффективно реализовать восстановление нормального счета прикладной программы в случае аппаратного сбоя (путем переповтора пострадавшего фрагмента вычислений) на новой исправной конфигурации.
- доработанная в части отказоустойчивости реализация среды исполнения T++ программ. Эта среда опирается на отказоустойчивую реализацию DMPI и T-систему с открытой архитектурой и автоматически обеспечивает меры для отказоустойчивости для программ, реализованных в функциональном стиле на языке T++ (без MPI-вызовов), которые могут работать как в сильно-, так и в слабосвязанной вычислительной среде.

2. Краткая характеристика DMPI и OpenTS

T-система — концепция автоматического динамического распараллеливания программ, составленных в функциональном стиле.

OpenTS — современная реализация подхода T-системы в виде системы автоматического динамического распараллеливания программ с открытой архитектурой [3, 1].

T++ — синтаксически и семантически гладкое расширение языка C++; основной входной язык, распознаваемый синтаксическим анализатором OpenTS.

DMPI — Dynamic MPI. Инфраструктура, реализующая идею динамической загрузки коммуникационной библиотеки MPI по используемой разновидности сценария запуска MPI-программ.

Базовыми системными компонентами, обеспечивающими отказоустойчивость исполнения T-приложений, являются DMPI и микроядро OpenTS.

3. Методы обеспечения отказоустойчивости

В последнее время классическая идея избыточных вычислений (перевычислений) принимает новые эффективные формы, которые являются более привлекательным, чем простое сохранение полных контрольных точек в надежную энергонезависимую память.

Использованы следующие оригинальные методы обеспечения отказоустойчивости:

1. Сохранение пренатальных процессов (только что вызванных T-функций), так как они еще не получили стека для своей работы и находятся в наиболее компактной (и даже адресно-независимой) форме.

2. Используемая T-системой коммуникационная библиотека DMPI способна сигнализировать о сбоях, продолжая нормально функционировать и отслеживать, какие сообщения следует перепослать, а какие нет.

Реализовано запоминание T-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 T-функций, которые относятся к последней контрольной точке, производится после реконфигурации коммуникационной подсистемы и Суперпамяти.

4. Архитектура ПО отказоустойчивого DMPI

4.1. Краткое описание реализации DMPI над протоколом TCP/IP

DMPI TCP/IP реализует функциональность подмножества стандарта MPI 2.0 над протоколом TCP/IP. В ходе данного проекта отказоустойчивая версия DMPI перенесена на платформу Windows Compute Cluster Server.

4.2. Программный интерфейс отказоустойчивого DMPI

Отказоустойчивый DMPI, как и обычный DMPI, в основном используется другим, более высокоуровневым системным ПО, таким как ядро T-системы, поэтому от него не обязательно требовать каких-то особенных свойств, характерных для компонент, видимых прикладному программисту. Тем не менее, чем ближе он будет по способу

использования к привычным MPI/DMPI, тем менее обременительным будет его использование.

Если говорить о минимально необходимой функциональности, видимой со стороны использования отказоустойчивого варианта DMPI, то это, безусловно, сохранение базовой работоспособности — способности передачи сообщений, в условиях отказов отдельный узлов, а также как можно более раннее информирование вышестоящего уровня ПО о возникшей проблеме.

Обычная практика для оповещения — регистрация пользовательской процедуры, которая вызывается в случае обнаружения сбоев. Данный способ обладает высокой универсальностью и оставляет достаточную свободу для развития возможностей ПО.

В DMPI используется наиболее простой и безусловно отказоустойчивый транспортный уровень TCP/IP, изначально созданный для работы в условиях сбоев коммуникационного оборудования, и обладающий логикой надежной доставки сообщений.

Безусловно, использование TCP/IP в качестве базового уровня для реализации MPI-взаимодействия может внести некоторые накладные расходы при использовании внутри тесносвязанных кластеров. Однако для нас сейчас важнее уверенность в безукоризненной отказоустойчивости базового уровня коммуникаций; кроме того, уже начиная с небольших метакластерных установок, накладные расходы, вносимые уровнем TCP/IP, не будут такими уж существенными хотя бы потому, что сами кластерные установки обычно связаны именно по этому протоколу.

Изучение открытых ресурсов показало, что существует простая, но достаточно эффективная надстройка над протоколом TCP/IP, реализующая наиболее часто употребимые функции библиотеки MPI — MP_Lite. Была проведена рефакторизация исходного кода, выделен ряд процедур, на базе которых реализован драйвер DMPI_TCP.

4.3. Доработка архитектуры DMPI

Доработка DMPI состоит в том, что попытка запуска приложения на каждом узле производится не однократно, а многократно, то есть в случае сбоя/перезапуска узла возникает новая «реинкарнация» вычислительного процесса, который готов продолжить работу.

Кроме запуска командой mpiexec, разработана схема динамического вхождения вычислительного узла в расчетное поле по собственной инициативе. При этом, вычислительный узел может не обладать собственным IP-адресом, так как соединение с коммуникационным сервером происходит по инициативе самого узла.

В этой схеме головной процесс работает дополнительно в режиме сервера, принимая запросы на соединение с дополнительных узлов. После успешного установления такого соединения каждый подключенный по собственной инициативе узел взаимодействует со всеми остальными узлами по протоколу TCP/IP, обмениваясь активными сообщениями уровня OpenTS.

В том случае, если функция-обработчик не установлена, предполагается, что следует вызывать функцию-обработчик, существующую в системе по умолчанию. В случае вызова при возобновлении работоспособности узла данная функция печатает соответствующее сообщение на консоль оператора; в случае сбоя любого узла производится печать его номера и аварийное завершение вычислительного процесса.

5. Архитектура отказоустойчивой версии OpenTS

5.1. Ограничения на стиль программирования

Т-система изначально предполагала функциональный стиль программирования, и рассматриваемый подход отказоустойчивости на основе модели перевычислений предполагает активное использование специфики функциональных программ для того, чтобы получить эффективный алгоритм восстановления в случае сбоев.

Мы будем предполагать, что прикладная программа написана в функциональном стиле, и в случае его расширения прикладной программист так же точно отвечает за отказоустойчивость своей программы при использовании отказоустойчивой OpenTS, как и за простые корректность и детерминированность результатов счета в случае обычной Т-системы.

5.2. Неготовые значения и незавершенные по причине сбоя вычисления

С точки зрения прикладного программиста наиболее существенное новшество Т-системы (языка Т++) по сравнению с базовым языком (С++) состоит в поддержке Т-функций и неготовых значений.

Неготовые значения генерируются Т-функциями, и служат удобными абстракциями, инкапсулирующими синхронизацию и обмен данными между параллельно вычисляющимися функциями-потоками.

Для того чтобы у прикладного программиста появилась возможность самостоятельно принимать решение об обработке сбоев, диаграмма состояний неготового значения была дополнена еще одним состоянием — «незавершенное». Это состояние является альтернативой готовому состоянию и отличается от него следующими ключевыми свойствами:

- Неготовое значение становится незавершенным, если его вычисление было остановлено вследствие сбоя.
- Незавершенное значение является готовым в том смысле, что попытка прямого получения находящегося в нем реального значения не приводит к остановке процесса, но ведет к исключительной ситуации в программе. То есть, значение можно получить, но если сделать это непосредственно, то произойдет исключение.
- В случае, когда прикладная программа использует примитив языка Т++ twait(), она получает событие, означающее что величина готова, но вычисление не было завершено. В этом случае она может продолжать счет, предприняв то или иное действие в соответствии с выбранной «заказной» моделью обработки сбоя.

Для того чтобы обеспечить полностью автоматическую обработку сбоев, реализован дополнительный режим, который приводит к циклической попытке вычислить значение неготовой переменной, до тех пор, пока примитив twait() не вернет успех в плане завершенности вычислений. Этот цикл организован внутри Т-системы, так что пользователь никогда не получит исключительной ситуации — счет с его точки зрения будет происходить как обычно, а Т-система будет производить перезапуск соответствующих Т-функций автоматически.

5.3. Откат незаконченных вычислений

Если проводить аналогии с транзакциями, то существуют настолько «безнадежные» состояния вычислений, которые не представляется возможным и не имеет смысла пытаться продолжать или переповторять. К таким состояниям можно отнести состояние вычисления подвыражения такого выражения, которое попало на узел, на котором наступил отказ. В этом случае собственно подвыражение может

продолжать вычисляться, но смысла в этом нет по причине того, что его результата никто не ожидает — его попросту некому будет обрабатывать.

Очевидно, Т-функции, которые вычисляют такого рода подвыражения, должны останавливаться. Сделать это каким-то внешним механизмом не так просто, вследствие того, что эти подвыражения в процессе своего вычисления могут порождать новые Т-функции и так далее. Поиск всех «безнадежных» Т-функций, которые мигрируют по устойчивому множеству узлов, может оказаться технически непростой задачей.

В данном случае целесообразно использовать другую технику, предоставив подобным Т-функциям самим распознавать ситуацию своей «безнадежности» и завершать свое исполнение наиболее естественным путем.

5.4. Доработка функциональности и логики микроядра OpenTS

Для того чтобы возобновившие свою работу или вновь появившиеся узлы могли быть вновь эффективно использованы при расчетах, необходимо реализовать логику их повторного динамического подключения. Для этого на основе информации, которая поступает от коммуникационной подсистемы DMPI, и с учетом информации о зафиксированном сбое нужно создать новую или модифицировать уже имеющуюся запись, содержащую информацию о данном узле.

Тогда на следующем проходе планирования эта информация будет учтена и узел начнет получать подзадачи, которые являются адекватными для его уровня надежности.

6. Вектор перерождений

6.1. Определение и свойства вектора перерождений

Популяция исправных узлов, образующих устойчивое множество, может меняться в случае временного выхода отдельных узлов из строя.

Будем называть процесс выхода узла из устойчивого множества гибелью, а входа — рождением. Тогда в каждый момент времени устойчивое множество характеризуется вектором чисел — порядковыми номерами перерождения для каждого узла и текущим их статусом работоспособности.

6.2. Кодирование и передача вектора перерождений

Если ограничить значение счетчика перерождений некой разумной величиной (например, типом данных **int**), то можно хранить вектор перерождений в массиве байт (каждый элемент содержит 31 бит на порядковый номер и 1 бит на текущий статус работоспособности).

Такой способ кодирования очень удобен для представления вектора перерождений, так как он является «монотонным объектом» — каждый байт меняется строго последовательно: в момент старта подсистема DMPI, переведенная в отказоустойчивый режим, начинает с полностью нулевого вектора перерождений. Далее, по мере обнаружения работоспособных узлов, их байты перерождений становятся равными единице (то же воплощение, но переход из нерабочего состояния в рабочее — младший бит установлен). В случае сбоя байт становится равным двум (следующее воплощение, пока не работоспособен), затем трем и так далее.

Вектор перерождений хранится в одной из ячеек суперпамяти, размещенной на нулевом узле, так как это глобальный объект.

7. Вектор посещений

Каждой Т-функции мы сопоставим вектор посещений, который содержит номера перерождений тех узлов, на которых либо она находилась сама, либо находились ее предки.

В случае обнаружения несовместимости вектора посещений Т-функции с вектором перерождений (при несовпадении порядкового номера перерождения хотя бы одного из узлов, где считалась данная Т-функция либо ее предки) данная Т-функция является «безнадежной» и подлежит самоуничтожению с освобождением всех захваченных ресурсов.

Очевидно, вектор посещений является характеристикой Т-функции и должен передаваться (и корректно при этом модифицироваться) вместе с ней в случае миграции.

В Т-системе Т-функции могут многократно мигрировать между узлами, что в нашем случае приведет к сложностям с определением и осмысленностью вектора посещений. Поэтому в отказоустойчивом режиме целесообразно разрешить лишь однократную миграцию Т-функций.

8. Результаты и выводы

Система OpenTS доработана для поддержки отказоустойчивого режима работы. Такой режим позволяет эффективно проводить длительный счет в неоднородной среде, состоящей из априорно неизвестного и постоянно изменяющегося множества как отдельных компьютеров, так и эпизодически предоставляемых небольших кластеров. А именно такая форма вычислительной мощности является наиболее дешевой, и, как следствие, перспективной для сверхтяжелых расчетов.

Список литературы

1. Абрамов, С.М. Кроссплатформенная версия Т-системы с открытой архитектурой / С.М. Абрамов, А.А. Кузнецов, В.А. Роганов // Proc. Труды Международной научной конференции "Параллельные вычислительные технологии (ПаВТ'2007)", Челябинск, 29 января - 2 февраля 2007 г., Челябинск, изд. ЮУрГУ, Т 1, с. 115-121

2. Перенос средств параллельного программирования OpenTS на платформу Windows Compute Cluster Server / С.М. Абрамов [и др.] // Proc. Международная конференция "Программные системы: теория и приложения", Переславль-Залесский, Т. 1, октябрь 2006, Наука, Физматлит, М.. с. 233-243

3. Т-система с открытой архитектурой / С.М. Абрамов [и др.] // Суперкомпьютерные системы и их применение SSA'2004. Труды Международной научной конференции, 26-28 октября 2004 г. Минск.