ИНТЕГРАЦИЯ ВОЗМОЖНОСТЕЙ FAULT-TOLERANCE OPENMPI В СИСТЕМУ УПРАВЛЕНИЯ КЛАСТЕРОМ ИММ УРО РАН

А.С. Игумнов, А.Ю. Берсенёв

Введение

Одна из проблем, связанных с увеличением количества вычислительных узлов в кластере, - рост вероятности отказа одного или нескольких из них. Для минимизации потерь используют отказоустойчивые (fault-tolerance — FT) системы. Одной из стратегий достижения отказоустойчивости является сохранение исправного состояния программы и файловой системы (ФС) в контрольной точке (КТ) с последующим восстановлением из неё после устранения причины сбоя.

Вторая проблема связана с необходимостью распределения ресурсов кластера между несколькими задачами. В частности, при нехватке процессорных ресурсов приходится принудительно завершать низкоприоритетные задачи при запуске высокоприоритетных задач. Система сохранения КТ позволяет заменить принудительное завершение задачи вытеснением её на диск с возможностью продолжения счёта после освобождения ресурсов. В данном случае сохранение состояния файловой системы не обязательно, так как программа после создания КТ завершается и не может изменить файлы с данными.

В данной работе рассматривается решение указанных проблем в системе прохождения задач СУППЗ (разработка ИПМ им.Келдыша и ИММ УрО РАН) с помощью отказоустойчивой версии OpenMPI FT.

Структура системы прохождения задач

Система прохождения задач состоит из следующих компонентов.

1. Фронтальная машина, обеспечивающая пользовательский интерфейс.
2. Сервер очереди задач, отслеживающий наличие требуемых и свободных ресурсов,а также выполняющий запуск и завершение задач.
3. Вычислительные узлы.
4. Файловый сервер, обеспечивающий совместный доступ к данным для предыдущих компонентов.

Жизненный цикл задачи состоит из следующих этапов.

1. Пользователь на фронтальной машине командой mpirun-imm формирует паспорт задачи,включающий в себя параметры запуска, и передаёт его серверу вместе с информацией о необходимых ресурсах (количество процессоров и предполагаемое время счёта).
2. При наличии доступных ресурсов, сервер очереди помечает необходимое количество узлов как "занятый";на каждом из этих узлов выполняет скрипт инициализации - prepare_node;затем запускает задачу, выполнив на одном из этих узлов (далее "стартовый узел" - СУ)команду mpirun с параметрами, указанными в паспорте задачи.Одновременно с запуском основной задачи на сервере очереди запускается вспомогательная задача-таймер.
3. В случае завершения основной задачи или задачи-таймера на всех выделенных узлах запускается скрипт очистки узла - close_node. Его предназначение — уничтожение всех процессов и временных файлов, принадлежащих задаче.

Архитектура системы создания КТ в OpenMPI

Библиотека OpenMPI имеет модульную структуру, поделённую по функциональности на три слоя.

1. Слой OpenMPI (OMPI) отвечает за взаимодействие с пользовательской программой и предоставляет ей интерфейс,описанный в стандарте MPI.
2. Слой Open Run-Time Environment (ORTE) осуществляет поддержку системы во время выполнения(runtime)и позволяет пользователям запускать их приложения в распределенной среде.
3. Слой Open Portable Access Layer (OPAL) скрывает особенности отдельного вычислительного устройства,обеспечивает взаимодействие с операционной системой и сервисами, которые она предоставляет.

Управление контрольными точками выполняется модулями, реализующими следующие функции.

1. Реализация протокола координации, который бы гарантировал целостное состояние программы при создании КТ (ORTE).
2. Оповещение подсистем OpenMPI о запросах на создание и восстановление КТ, а также координация этих подсистем (ORTE).
3. Запуск и агрегация запросов на создания КТ, наблюдение за процессом создания (ORTE).
4. Взаимодействие с системами создания КТ одного процесса (OPAL).
5. Управление относящимися к КТ файлами и каталогами в распределенной среде, потенциально включающей как локальные, так и глобальные файловые системы (ORTE).

В настоящий момент в стандартную поставку OpenMPI входят модули КТ уровня OPAL ориентированные на ОС Linux и библиотеку создания локальных КТ - BLCR.

Для сборки OpenMPI FT необходимо установить модули ядра и библиотеку BLCR, после чего сконфигурировать OpenMPI со следующими параметрами:

./configure --enable-ft-thread --with-ft=cr --enable-mpi-threads [--with-blcr --with-blcr-libdir=DIR]

Для инициализации соответствующих функций во время выполнения необходимо запускать параллельную программу командой:

mpirun -am ft-enable -cr –mca crs_base_snapshot_dir <dir> progname <args>

где <dir> - имя каталога в котором будут сохраняться контрольные точки. При интеграции в СУППЗ имя каталога генерируется на основе имени задачи и времени запуска задачи.

Сохранение контрольной точки выполняется на СУ командой
ompi-checkpoint [--term|--stop] <pid>
где <pid> идентификатор процесса mpirun, а опции --term или --stop - указывают на то, что после сохранения контрольной точки всем процессам параллельной программы будут отправлены сигналы SIGTERM или SIGSTOP, соответственно.

Восстановление контрольной точки производится командой

ompi-restart <ckpt-dir>

где <ckpt-dir> - каталог, содержащий контрольную точку

"Снимок" файловой системы

На настоящий момент поддержка "снимков" состояния файловой (КТФС) системы заявлена в нескольких ФС.

В данной работе использовалась ФС BTRFS. Выбор этой ФС был обусловлен тем, что она включена в основную ветку ядра Linux, а так же тем, что BTRFS является кандидатом на роль базовой ФС многих дистрибутивов Linux.

Создание КТФС выполняется на файловом сервере с помощью команды

btrfsctl -s <snapshot_name> <directory>

где snapshot_name - имя КТФС, directory - имя каталога, в который смонтирована BTRFS. При интеграции в СУППЗ имя КТФС генерируется на основе имени задачи и текущего времени.

Руководство по btrfsctl описывает создание КТФС для произвольного каталога в BTRFS.

Это не совсем верно - создание КТФС возможно для так называемых subvolume - подразделов диска, связанных с каталогами.

Для обеспечения независимого сохранения КТФС для каждой из счётных задач было принято решение ограничивать данные задачи одним каталогом, оформленным как subvolume командой

btrfsctl -S <subvolume_name> <directory>

Модификация системы прохождения задач

В скрипт постановки задачи в очередь mpirun-imm добавлены ключи, указывающие на режим создания КТ:

- без КТ; режим завершения по времени с сохранением КТ;
- режим защиты от сбоев, с указанием периодичности сохранения КТ;
- режим старта из сохранённой КТ.

В скрипт подготовки вычислительных узлов prepare_nodes добавлены команды загрузки модуля ядра blcr.

В скрипте завершения задачи close_nodes изменён код отвечающий за завершение задачи на СУ.

Отправка сигнала SIGTERM вычислительной задаче заменена на сохранение КТ с завершением задачи openmpi-checkpoint --term <pid_mpirun>.

В режиме защиты от сбоев на СУ запускается вспомогательный скрипт-таймер. Этот скрипт периодически выполняет следующие действия.

1. Создание КТ с остановкой программы ompi-checkpoint --stop <pid_mpirun>.
2. Удалённое выполнение на файловом сервере команды создания КТФС.
3. Запуск приостановленной программы, путём отправки сигнала SIGCONT процессу mpirun.

Тестовые задачи

Надёжность восстановления из КТ тестировалась на задачах из NASA Parallel Benchmark.B качестве критерия успешного прохождения теста использовалась внутренняя проверка целостности данных, выполняемая при завершении задачи. Проведено 10000 циклов сохранение/восстановление КТ. Сбоев не наблюдалось.

Для тестирования КТФС использовалась простая тестовая задача, построчно пишущая в файл последовательность целых чисел с принудительным выталкиванием буферов после каждой строки. Данный тест не охватывает такие ситуации как удаление и переименование файлов, но достаточно удобен для отладки системы прохождения задач. Создано 500 КТФС. Сбоев не наблюдалось.

Время сохранения КТ в основном определяется памятью, занимаемой данными задачи на всех узлах, и пропускной способностью дисковой подсистемы файлового сервера. При скорости записи на диск 40 МБайт/с запись контрольной точки объёмом 64ГБайт занимает более 30 минут. Возможно, что решением данной проблемы может стать использование распределённой ФС (например Ceph) совместно с BTRFS.

Конфигурация и версии ПО

Тестирование проводилось на кластере из 16 двухядерных узлов на основе процессора Xeon(2.4 ГГц); 4 ГБ ОЗУ на узел; коммуникационная среда - Gigabit Ethernet.

CentOS 5.4 ядро Linux 2.6.18

Система запуска задач СУППЗ

OpenMPI версия 1.5 SVN репозиторий

Файловая система BTRFS Git репозиторий

BLCR версия 0.8.2

Работа выполнена в рамках Программы фундаментальных исследований Президиума РАН № 14 "Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация" при поддержке УрО РАН, проект 09-П-1-1003

ЛИТЕРАТУРА:

1. СУППЗ http://www.jscc.ru/scomputers.shtml
2. OpenMPI http://www.open-mpi.org
3. BTRFS https://btrfs.wiki.kernel.org/index.php/Main_Page
4. BLCR https://ftg.lbl.gov/CheckpointRestart/CheckpointRestart.shtml
5. Ceph http://ceph.newdream.net/