Математические вопросы кибернетики и вычислительной техники 30, 47--53, 2008.

# Создание контрольных точек и восстановление MPI программ

Мгер Ю. Мовсисян

Ереванский физический институт им. А.И.Алиханяна
mher.movsisyan@gmail.com

**Аннотация**

Выполнение программ на вычислительных кластерах обычно занимает довольно большое время. В процессе выполнения может возникнуть потребность изменения физического местоположения отдельных процессов параллельной программы или временная остановка всей программы. В этой статье описана разработанная система CROM (Checkpointing and Recovery of MPI), которая предоставляет возможность создания контрольных точек для остановки и последующего возобновления выполнения MPI программы. Функциональность создания контрольных точек и восстановления реализована в виде дополнительных компонент MPICH2 и не требует изменений в коде MPI программы.

## 1. Введение

Время выполнения приложения на вычислительном кластере может достигать нескольких суток, а в отдельных случаях - недель или, даже, месяцев. В таком длительном промежутке времени может возникнуть потребность изменения физического местоположения отдельных процессов параллельной программы или временная остановка всей выполняемой программы. В основном эти потребности связаны с аварийными ситуациями в аппаратуре или с планированием использования ресурсов кластера. Восстановление выполняемой программы с использованием заранее созданных контрольных точек является одним из способов миграции процессов. Выполняемая программа сохраняться в виде данных, а потом выполнение возобновляется в новой среде.

Существующее стандартное программное обеспечение для кластеров не имеет средств создания контрольных точек и восстановления выполняемых программ. Приложения для кластеров в основном реализуются на основе технологии MPI [2, 3], которая является библиотекой, обеспечивающей средства связи между распределенными процессами. Процессы взаимодействуют посредством сообщений. Основные реализации MPI [4, 5] также не содержат функций создания контрольных точек. Для обеспечения этой функциональности программисты сами создают в своих программах логические точки сохранения.

Разработанная система CROM (Checkpointing and Recovery of MPI) предоставляет возможность создания контрольных точек и восстановления выполнения программы без изменения кода MPI-программы. Вся функциональность создания контрольных точек и

восстановления инкапсулирована в вызовах MPI. Нужно только линковать MPI программу с MPICH2 [4], который специально сконфигурирован с использованием системы CROM.

Создание контрольных точек в CROM выполняется командой mpicheckpoint, использующей менеджер процессов, входящий в систему CROM, или локально посылая одному из процессов системный сигнал. Восстановление MPI программы выполняется командой mpirecover. Система CROM может быть внедрена и в систему управления заданиями.

## 2. Глобальное непротиворечивое состояние

Создание контрольных точек для параллельных программ отличается от случая последовательных программ тем, что сообщения, которыми взаимодействуют процессы параллельной программы приводят к межпроцессной зависимости. Для того, чтобы параллельная программа была в непротиворечивом состоянии нужно организовать создание контрольных точек таким образом, чтобы сохраненное состояние параллельной программы было в непротиворечивом состоянии [6, 12]. В противном случае, восстановление программы после создания контрольных точек может повлиять на результат вычислений. То есть, независимо от того, были ли во время работы программы остановки, или нет, результат вычисления должен быть одним и тем же.

Например, на рисунке 1 показаны два глобальных состояния параллельной программы. Черными квадратами изображены состояния процессов, а кривые соединяющие процессы представляют глобальное состояние программы. Из рисунка видно, что первое состояние является непротиворечивым, а второе противоречивым. Заметим, что в первом случае существуют сообщения, которые отправлены, но не достигли цели. Это состояние является непротиворечивым, поскольку оно представляет состояние, в котором сообщение было отправлено, но не получено. А во втором случае программа находится в противоречивом состоянии, поскольку состояние процесса P2 показывает, что сообщение m2 получено от P1, но на состоянии P1 это не отражается.

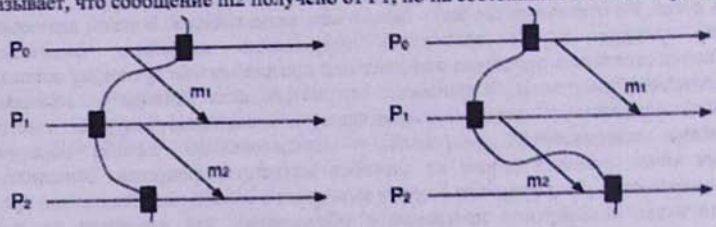


Рис. 1

Такая ситуация не может возникнуть при нормальной работе программы. Она только может возникнуть при неправильном создании контрольных точек. Например, если сохранение состояния P1 процесса выполняется раньше, чем P2, то может возникнуть такая ситуация, что во время сохранения процесса P2 сообщение m2 будет получено, но в состоянии P1 это не отражено.

Система CROM реализует функциональность создания контрольных точек и восстановления, добавляя в реализацию MPI два дополнительных свойства. Первое - это протокол создания контрольных точек. Для того чтобы глобальное состояние MPI программы было в непротиворечивом состоянии, создание контрольных точек производится по специальному протоколу [1]. Суть протокола состоит в том, что перед сохранением состояния процессов каналы связи очищаются специальными маркер сообщениями.

Второе дополнение связано с преодолением ограничений, которые накладываются программными пакетами при создании контрольных точек для процессов операционной системы. Существующие механизмы создания контрольных точек для процессов операционной системы накладывают ограничения на типы ресурсов, состояния которых можно сохранить. В основном, это открытые соединения и ресурсы межпроцессной коммуникации. Для этого перед созданием контрольных точек те ресурсы, которые не могут быть сохранены, освобождаются и заново создаются во время восстановления.

Созданная система CROM реализована на основе MPICH2 в виде специальных компонент. Функциональность создания контрольных точек и восстановления реализована как дополнение к существующей реализации MPI, а не как новая реализация. Это обстоятельство позволяет сохранить производительность MPI программы. Производительность играет очень важную роль для приложений выполняемых на кластерах. Обычно MPI программы оптимизируются для конкретных реализаций MPI, чтобы обеспечить максимальную производительность. Обеспечение функциональности создания контрольных точек и восстановления на основе существующих инструментов является критичным для сохранения производительности кластерной программы.

## 3.1. MPICH2

MPICH2 [4] является основной реализацией стандарта MPI. MPICH2 создана на компонентной основе. Функциональность MPICH2 разделена между компонентами, которые взаимодействуют между собой через четко определенные интерфейсы. На рисунке 2 представлена компонентная архитектура MPICH2. Компонентная архитектура позволяет отделить основную реализацию стандарта MPI от способа создания процессов и механизмов передачи сообщения.

Выполняемая MPI программа в MPICH2 состоит из процессов параллельной программы, которые вызывают функции MPI для отправки и приема сообщений и менеджера процессов, который обеспечивает создание и управление процессами.

Функциональность MPICH2 можно разделить на три основные компоненты: менеджер процессов, ядро MPI и транспортное средство передачи данных. Менеджер процессов выполняет создание процессов параллельной программы и предоставляет информацию процессам для взаимосвязи. В MPI адресация происходит по индексу параллельной программы. Для того чтобы отправить сообщения процесс отправитель во время вызова функции отправки сообщения MPI указывает индекс процесса получателя. Реализация MPI транслирует индекс процесса в физический адрес и отправляет сообщение. В MPICH2 трансляция между индексом процесса и физическим адресом производит менеджер процессов. Менеджер процессов по запросу ядра транслирует индекс процесса в физический адрес. Менеджер процессов и ядро MPI взаимодействуют между собой по протоколу PMI [8]. В ядре реализована основная часть стандарта MPI. Реализация ядра MPI абстрагирована двумя интерфейсами: PMI и ADI [8]. Ядро MPI не зависит от топологии процессов, от способа создания процессов и от типа сети, по

которой происходит обмен сообщениями. Ядро взаимодействует с менеджером процессов по протоколу PMI. Для отправки сообщения ядро использует интерфейс ADI. ADI абстрагирует отправку сообщения по конкретной сети. Например, для TCP/IP и для Myrinet [7] сетей существуют отдельные реализации интерфейса ADI.

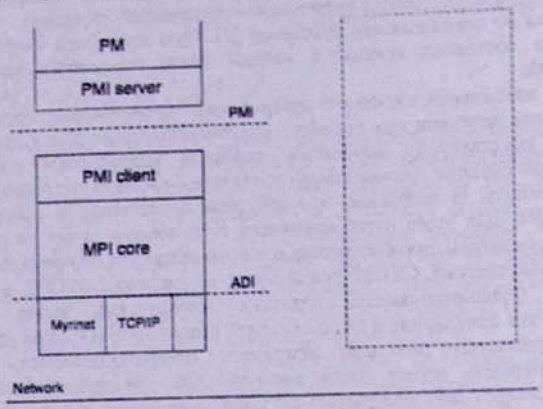


Рис. 2

## 3.2. BLCR

BLCR (Berkeley Lab Checkpoint/Restart) [9] это библиотека для создания контрольных точек. BLCR позволяет сохранить и восстановить процессы на уровне ядра. BLCR работает как загружаемый модуль ядра Linux (loadable kernel module) [9, 10]. BLCR имеет простой программный интерфейс, который позволяет программно инициировать сохранение процесса или регистрировать пользовательские функции которые будут вызваны перед сохранением процесса или перед восстановлением.

Главным недостатком BLCR является то, что она не позволяет сохранить открытые соединения. Дескрипторы открытых соединений после восстановления процесса из созданной контрольной точки становятся недействительными, и последующие попытки использования соединения оказываются неудачными.

BLCR используется в CROM для создания контрольных точек для отдельных процессов. BLCR вызывается для каждого процесса MPI программы после того как вся программа приведена в непротиворечивое состояние и не сохраняемые ресурсы освобождены.

## 3.3. Менеджер процессов

В CROM для поддержки функциональности создания контрольных точек и восстановления реализован специальный менеджер процессов, который помимо команды запуска MPI программ mpiexec имеет команды mpicheckpoint и mpirecover для сохранения и восстановления процессов (рис. 3). Команда mpicheckpoint имеет ключ –t, при наличии которого после создания контрольных точек выполнение MPI программы завершается.

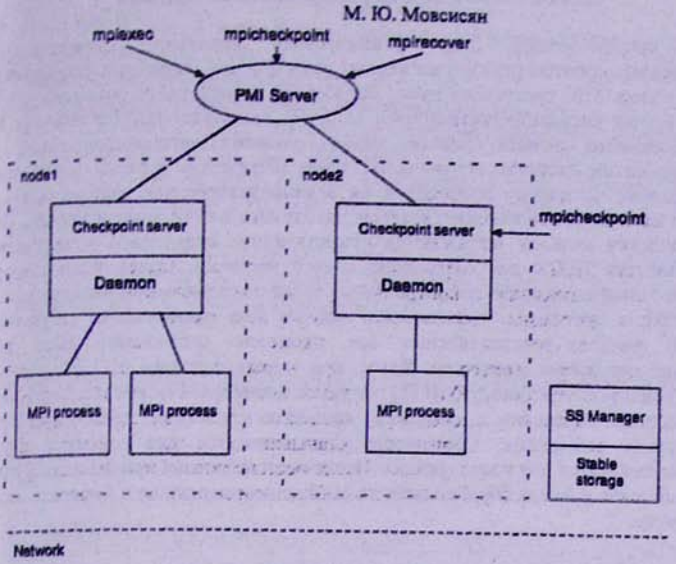


Рис. 3

Выполнения команды mpiexec начинается с выбора машин, на которых должны выполняться процессы. Менеджер процессов из своей конфигурации получает имена доступных машин. В зависимости от количества запускаемых процессов, менеджер процессов случайным образом выбирает машины, на которых должны выполняться процессы формирующие задачу. Если число доступных машин меньше числа требуемых процессов, то процессы равномерно распределяются между машинами. После выбора на каждой машине запускается специальный демон. Демон создает выполняемые процессы, после чего инициирует работу специального сервера, который обслуживает запросы создания контрольных точек и восстановления. Сервер контрольных точек является простым XML-RPC сервером. Команды создания контрольных точек могут быть посланы как программно, так и из веб-браузера.

При получении запроса сохранения сервер посылает всем процессам специальный системный сигнал сохранения. Запросы создания контрольных точек могут поступать как от mpicheckpoint так и локально. Системный сигнал обрабатывается процессами, в результате создаются контрольные точки. После создания контрольных точек сервер перемещает контрольные точки в стабильное хранилище.

Во время восстановления команда mpirecover инициализирует менеджер процессов, который заново распределяет местоположение процессов и из контрольных точек возобновляет работу процессов. Нужно отметить, что во время восстановления менеджер процессов может использовать отличные от изначальных адреса. При соединении процессы получают адреса заново.

4. Создание контрольных точек и восстановление

В системе CROM инициирование создания контрольных точек не централизовано. Оно может быть выполнено как локально на каждом из узлов параллельной программы, так и от управляющей машины из менеджера процессов. Локальное создание

контрольных точек может быть инициировано локальным демоном или администраторским скриптом (при случае перезагрузки или при аварийных ситуациях).

Каждый процесс MPI программы имеет обработчик специального системного UNIX сигнала. Обработчик сигнала регистрируется во время инициализации библиотеки MPI. Получение системного сигнала означает начало создания контрольных точек. При получении запроса на создание контрольных точек обработчик сигнала в каждом из процессов начинает процедуру приведения их в непротиворечивое состояния [1]. Во время очистки каналов все сообщения, которые находились в пути, буферизуются. После выполнения очистки каналов закрываются существующие соединения и вызываются функции библиотеки BLCR для сохранения самого процесса. После этого процессы синхронизируют окончание создания контрольных точек и возобновляют выполнение.

Восстановление программы начинается с запуска всех процессов из сохраненных состоянии. В начале восстановления все процессы отправляют свое новое местоположение менеджеру процессов. Далее, при вызове функции MPI для отправки сообщения, процессы по протоколу PMI [8] получают новые физические местоположения соседних процессов. Обращение к менеджеру процессов происходит только при первой попытке отправки сообщения. Соединение устанавливается для отправки первого сообщения и используется для последующих. После восстановления при вызовах функции получения сообщения сначала обрабатываются сообщения находящие в буферах, а потом пришедшие позже.

## 5. Интеграция с системой управления заданиями

В кластерных системах выполнение программ организуется с использованием систем управления заданиями (Job Management System - JMS), которые выполняют контроль над пользовательскими программами, диспетчеризацию и планирование. Пользователи независимо друг от друга отправляют свои программы системе управления заданиями кластера, который ставит их в очередь, планирует и выполняет.

Создание контрольных точек и восстановление может быть произведено как из менеджера процессов CROM, так и внедрено в систему управления заданиями. Для внедрения в систему управления заданиями нужно использовать реализацию протокола PMI, которая включена в CROM.

## Литература

[1] M. Movsisyan, V. Sahakyan, "Transparent checkpointing protocol for MPI programs with decentralized initiator", CSIT 2007, pp. 227-229.

[2] Message passing interface forum, "MPI: A Message-Passing Interface Standard", Version 1.1, June 1995. http://www.mpi-forum.org/docs/docs.html

[3] Message passing interface forum, "MPI-2: Extensions to the Message-Passing Interface", July 1997, http://www.mpi-forum.org/docs/docs.html

[4] MPICH2, http://www-unix.mcs.anl.gov/mpi/mpich2/

[5] Open MPI, http://www.open-mpi.org/

[6] M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems", In ACM Transactions on Computing Systems, 3(1): pp. 63-75, 1985.

[7] Myrinet, http://www.myri.com/myrinet/overview/

[8] The MPICH Team Argonne National Laboratory, "Process Management in MPICH2" DRAFT 2.1. March 30, 2007.
[9] Berkeley Lab Checkpoint/Restart (BLCR), http://ftg.lbl.gov/CheckpointRestart/CheckpointRestart.shtml
[10] H. Hargrove and C. Duell, "Berkeley Lab Checkpoint/Restart (BLCR) for Linux Clusters", In Proceedings of SciDAC 2006: June 2006.
[11] J. Duell, P. Hargrove, and E. Roman, "The design and implementation of Berkeley Lab's linux Checkpoint/Restart", Technical Report LBNL-54941, Lawrence Berkeley National Laboratory, 2003.
[12] M. Elnozahy, L. Alvisi, Y. M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message passing systems", Technical Report CMU-CS-96-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1996.

## Ստուգման կետերի ստեղծումը և վերականգնումը MPI ծրագրերում

Մ. Մովսիսյան

### Ամփոփում

Հաշվողական կլաստերներում ծրագրերի կատարումը սովորաբար տևում է բավականին երկար: Ծրագրերի կատարման ժամանակ կարող են ծագել զուգահեռ ծրագրի ընթացքների ֆիզիկական դիրքի փոփոխության կամ ամբողջ ծրագրի կատարման ժամանակավոր դադարեցման անհրաժեշտություններ: Այս հոդվածում նկարագրվում է հեղինակի կողմից մշակված CROM (Checkpointing and Recovery of MPI) համակարգը, որը հնարավորություն է ընձեռում պահպանել կատարվող MPI ծրագիրը և վերաթողարկել: Պահպանման և վերաթողակման ֆունկցիոնալությունը իրագործված է MPICH2 բաղադրիչների տեսքով և չի պահանջում MPI ծրագրի կոդի փոփոխություն: