ОБЕСПЕЧЕНИЕ ПРЕДЕЛЬНОЙ ОТКАЗОУСТОЙЧИВОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ С ИНТЕРФЕЙСОМ ПЕРЕДАЧИ СООБЩЕНИЙ MPI

HIGH-SPEED EDGE FAULT TOLERANCE BY USING PARALLEL PROGRAMMING, MESSAGE PASSING INTERFACE (MPI)

H. Sudani

Summary. The solution of the problem of providing stability to failures of high-performance parallel computing produced by cluster systems with interface of MPI message transmission is considered. To solve this problem, we consider the synchronization of recording time in the computational processes of their own checkpoints, and the exception of the formation of messages in the system, not received by applications, until the recording of the corresponding checkpoint. The fault tolerance limit will be determined by the number of free nodes in the cluster system.

Keywords: parallel programming, message passing interface, MPI, fault tolerance, clustered systems, distributed applications, control points.

Судани Хайдер Хуссейн
Аспирант, Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского
hayder.1977@mail.ru

Аннотация. Рассматривается решение проблемы обеспечения устойчивости к сбоям высокопроизводительных параллельных вычислений, производимых кластерными системами с интерфейсом передачи сообщений MPI. Для решения указанной проблемы рассматривается синхронизация по времени записи в вычислительных процессах собственных контрольных точек, и исключение формирования в системе сообщений, не полученных приложениями, до момента записи соответствующей контрольной точки. Предел отказоустойчивости будет определяться количеством свободных узлов в кластерной системе.

Ключевые слова: параллельное программирование, интерфейс передачи сообщений MPI, отказоустойчивость, кластерные системы, распределённые приложения, контрольные точки.

Одной из проблем обеспечения отказоустойчивых высокопроизводительных вычислений, производимых кластерными системами, является минимизация количества и быстрое преодоление возможных сбоев, возникающих в узлах рассматриваемых систем.

Кластерные системы содержат тысячи вычислительных узлов и распределённых приложений, которые осуществляют обработку данных в течение длительного промежутка времени (несколько суток и более) [1]. Ввиду сложности параллельных вычислительных процессов в течение суток вероятность возникновения сбоев в интерфейсах передачи сообщений MPI весьма высока [2]. В простейшем случае, если распределённые приложения не требуют взаимодействия подзадач во время их выполнения, то программные и аппаратные сбои преодолеваются посредством перезапуска незавершённой подзадачи на других узлах кластерной системы либо сначала, либо с определённой контрольной точки. По этой причине, вычислительная задача решается путём сокращения множества работоспособных узлов.

Однако, при взаимодействии подзадач с разделяемыми переменными или передаваемыми сообщениями, труднее обеспечить отказоустойчивость параллельных приложений [1]. В подобных случаях, простым произвольным перезапуском подзадачи можно потерять сообщения, которые были отправлены в её адрес, но не были получены из-за сбоя, либо произвести повторную отправку сообщений с результатами во второй раз выполненного фрагмента вычислений. Для решения указанной проблемы предлагаются различные способы: первый способ — синхронизация по времени записи в процессах собственных контрольных точек, и второй способ– отсутствие в системе переданных, но не полученных сообщений до момента записи контрольной точки. При реализации каждого из способов достигается предельная отказоустойчивость и обеспечивается непротиворечивость знаний обо всех параллельных процессах, что даёт информацию обо всей системе в целом, и при этом отсутствуют сообщения, находящиеся в процессе передачи на узлы во время сбоя [2].

Предел отказоустойчивости будет определяться количеством свободных узлов в кластерной системе. Предлагается использовать процедуру запоминания сообщений в интервалах времени между их отправкой и их приёмом на узлы или записи всех сообщений, сгенерированных согласно последним контрольным точкам процессов-получателей таких сообщений. Таким образом, на каждом из узлов параллельными процессами произвольно создаются контрольные точки, что позволяет осуществлять восстановление таких процессов после каждого сбоя. Кластерная сетевая система после предложенного изменения будет характеризоваться множеством динамически подключаемых (или отключаемых) взаимосвязанных компонент, запускаемых на отдельных узлах.

Система подразделяется на две подсистемы: подсистема библиотечных низкоуровневых коммуникаций [2] и подсистема по запуску и сопровождению параллельных MPI- приложений [1].

В первой подсистеме имеется единственный компонент — коммуникационная низкоуровневая библиотека. Во второй подсистеме имеются компоненты с диспетчерскими и исполнительскими функциями, а также функциями сохранения контрольных точек и канальной памяти. В зависимости от объёма выполняемой задачи системой выделяется определённое количество компонент с заданными функциями. Диспетчерский компонент является центральным, поскольку он подключает и отключает другие компоненты, определяет и переназначает ресурсы, устанавливает очерёдность выполнения приложений, разрешает ситуации отказов и сбоев. Компонент сохранения канальной памяти запоминает и обслуживает одну очередь сообщений, передаваемых от процессов параллельно исполняемых приложений, и осуществляет такое обслуживание очередей, начиная с контрольной точки, последней во времени. Для ликвидации сбоя процесс перезапускается компонентом сохранения канальной памяти на основе полученного сообщения в интервале, ограниченного временем регистрации сбоя и последней контрольной точки.

Исполнительный компонент осуществляет либо с начальной, либо с последней контрольной точки по команде диспетчерского компонента запуск подзадачи параллельного вычисления. Компонент сохранения контрольных точек производит приём, хранение и выдачу информации по последним контрольным точкам выполняемых подзадач. Контрольные точки характеризуются уникальными идентификаторами, которые определяются номерами приложений и подзадач, исполняемых в кластерной системе.

Коммуникационная низкоуровневая библиотека осуществляет процессы подключения к компонентам канальной памяти с передачей сообщений на протоколах TCP/IP, формирования служебных сообщений и сообщений для пользователей с целью параллельно вычисляемой генерации контрольных точек, основанных на библиотечных функциях кластерной системы [1]. Быстродействующая граничная отказоустойчивость системы достигается в процессе её эксплуатации, а именно: в процессе формирования и обслуживания процесса запуска системы; в процессе выполнении параллельного вычисления, его восстановлении после отказов и последующего завершения данного приложения. Диспетчерский компонент начинает процесс формирования системы со своего запуска на отдельном узле данной системы, соединённом с другими узлами с помощью специально выделенных портов. Диспетчерский компонент образует базу для параллельно вычисляемых приложений, которые регистрируются в системе, с последующим ожиданием возможных произвольных соединений по IP-адресу с другими компонентами кластерной системы. Новые подсоединяемые компоненты подлежат регистрации диспетчерским компонентом и размещаются среди множества доступных ресурсов. Отключаемые компоненты при отсутствии работающих приложений диспетчерским компонентом исключаются из состава доступных ресурсов. Новые компоненты могут войти в вычислительную систему при выполнении параллельных приложений, и в виде свободных ресурсов могут быть использованы другими приложениями, ожидающими необходимого объёма ресурсов.

При подключении нового компонента канальной памяти, он либо начинает работать в уже выполняющемся приложении при потребности последнего в данном компоненте, либо данный компонент регистрируется в качестве свободного ресурса. Для запуска приложения требуется подготовительный этап, в ходе которого происходит компиляция исходного кода приложения с привлечением модифицированных библиотек. На следующем этапе клиентский компонент пересылает сообщение с файлами и параметрами диспетчерскому компоненту, который определяет очерёдность для приложений и запускает их, если имеются свободные ресурсы. Приложение с параллельными вычислениями запускается при наличии в кластерной системе двух компонентов канальной памяти, одного компонента сохранения контрольных точек и одного исполнительного компонента. Исполнительные компоненты относятся к выделенным ресурсам, которые не могут использоваться для одновременного решения.

При выполнении параллельных MPI-приложений в кластерных системах возникают сбои их компонентов, которые считаются фатальными, если подключение компонента в систему после сбоя производится таким же образом, как и его первоначальное подключение, то есть при этом не выявлена причина, вызвавшая сбой. Отказоустойчивость достигается в кластерной системе с параллельными вычислениями только с использованием компонента канальной памяти. Указанный компонент хранит сообщения, передаваемые процессу распределённого приложения от момента последней контрольной точки до момента регистрации отказа для восстановления данного сообщения при продолжении прерванного процесса на свободном узле. Ввод в кластерную систему дополнительных компонентов нуждается в поддержании их собственной устойчивости к отказам или сбоям с выполнением дополнительных требований по надёжности узлов. Если компоненты канальной памяти будут размещены на ненадёжных узлах, то потребуется применение процедуры дублирования

соседних компонент канальной памяти с минимальными расходами. Сбой исполнительного компонента потребует наличия свободного узла и аналогичного компонента для перезапуска его подзадачи.

Таким образом, в данной статье было рассмотрено решение проблемы обеспечения устойчивости к сбоям высокопроизводительных параллельных вычислений, производимых кластерными системами, имеющими не надёжные узлы. Разделение системы на две подсистемы со следующими компонентами: коммуникационной низкоуровневой библиотекой с компонентами с диспетчерскими и исполнительскими функциями, функциями сохранения контрольных точек и канальной памяти, а также наличием свободных узлов позволяют обеспечить предельную отказоустойчивость при параллельно программированных вычислениях и с интерфейсом передачи сообщений MPI.

ЛИТЕРАТУРА

1. Параллельные вычисления в моделировании региональной экономики: учебное пособие / А. И. Долматова, Н. Н. Оленев.— Киров: ПРИП ФГБОУ ВПО «ВятГУ», 2012.— 125 с.
2. Оленев, Н. Н. Основы параллельного программирования в системе MPI. / М.: ВЦ им. А. А. Дородницына РАН, 2005.-81 с.

Журнал «Современная наука: актуальные проблемы теории и практики»

Саратовский национальный исследовательский государственный университет