РЕАЛИЗАЦИЯ НАДЕЖНЫХ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В МНОГОАГЕНТНОЙ СИСТЕМЕ

А.В. Отвагин

Белорусский государственный университет информатики и радиоэлектроники,
кафедра ЭВМ
220013, ул. П.Бровки, 6, г.Минск, Беларусь
телефон: + 375(17)293-80-39; e-mail: forlelik@mail.ru

Рассмотрена проблема управления надежностью параллельных вычислений в распределенной среде, отличающейся гетерогенностью архитектуры и допускающей отказы вычислительных узлов в процессе решения задачи. На базе многоагентной архитектуры предложена система управления вычислительным процессом с возможностью динамической реконфигурации приложений.

Ключевые слова – надежность вычислений, многоагентная система, динамическая оптимизация.

1 ВВЕДЕНИЕ

Современные высокопроизводительные вычислительные системы ориентированы на решение сложных вычислительных задач, требующих оптимального управления вычислениями с целью достижения высокой производительности. Для этого в процессе проектирования и анализа параллельного приложения разрабатывается схема алгоритма, позволяющая реализовать наиболее оптимальную схему вычислений с учетом специфики решаемой задачи. Однако, в последнее время круг задач, решаемых с помощью суперкомпьютеров, существенно расширяется за счет переноса существующих программных решений в параллельную вычислительную среду. В этом случае параллельное приложение строится с использованием компонентной архитектуры, допускающей реализацию нерегулярной структуры вычислений. Оптимизация такой структуры приложения на этапе проектирования не всегда целесообразна, поскольку во время выполнения возникают эффекты пространственной и временной гетерогенности вычислительной среды, влияющие на производительность. В этом случае более эффективной техникой является динамическое управление приложением с учетом загруженности отдельных элементов вычислительной среды в конкретный момент времени.

Важным аспектом современных вычислительных систем является гарантия надежности вычислений, то есть получения результата даже в случае изменения вычислительной среды. К таким изменениям можно отнести выход из строя или перегрузку узлов вычислительной системы, коммуникационные проблемы, ошибки операционной платформы. Любая нештатная ситуация способна существенно повлиять на производительность параллельного алгоритма и привести к остановке вычислений. В случае решения критических задач, требующих гарантированного получения результата в указанное время, нештатные ситуации ведут к прямым убыткам.

Таким образом, важной задачей является обеспечение не только оперативного управления вычислительным процессом с учетом информации о ходе его выполнения, но и реализация средств контроля и восстановления вычислений при сбоях, позволяющих без существенных затрат и потери промежуточных результатов восстановить вычислительный процесс. При этом средства контроля и восстановления вычислительного процесса не должны вносить существенную дополнительную нагрузку в процесс вычислений.

Существующие методы динамического планирования вычислительного процесса основаны на оценке состояния вычислений с помощью распределенного механизма сенсоров и выработке рекомендаций по изменению структуры приложения. Методы восстановления вычислительного процесса, основанные на введении контрольных точек и перезапуске приложения, требуют существенного вмешательства в код и достаточно сложны для реализации.

Одной из перспективных архитектур для реализации распределенных алгоритмов решения задач являются многоагентные системы (МАС) [1, 2]. На базе этой архитектуры предлагается система организации и управления вычислительным процессом с возможностью контроля и восстановления вычислений в случае возникновения нештатной ситуации.

2 ПРИНЦИП ФУНКЦИОНИРОВАНИЯ МНОГОАГЕНТНОЙ СИСТЕМЫ

Наиболее общей моделью представления распределенных вычислений является модель MIMD [3] (много программ для множества данных), которая представляет параллельное приложение в виде совокупности исполняемых модулей, взаимосвязанных по данным и управлению. Для наглядного представления чаще всего используется графовая модель, в которой вершины графа соответствуют операциям алгоритма, а дуги определяют связь между ними.

Пример графа распределенного приложения приведен на рисунке 1.

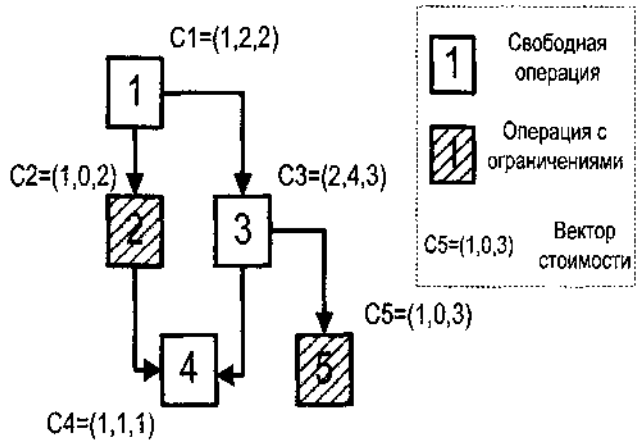

Рис.1. Графовое представление распределенного приложения.

Операции графа можно разделить на свободно мигрирующие и подвергающиеся ограничениям. Свободно мигрирующая операция может исполняться на любом вычислительном узле распределенной системы, не выдвигая требований к ресурсам, превосходящих возможности узла. Операция с ограничениями, напротив, существенно зависит от характеристик узла и может быть назначена только на определенные платформы, обладающие требуемым ресурсом. При реализации алгоритма планирования этот аспект должен быть учтен планировщиком.

Технология динамического планирования, являющаяся наиболее перспективной для организации гибких параллельных вычислений, работает с единицами диспетчеризации, которыми являются операции алгоритма. Чем меньше операции, тем эффективнее управление ими, но выше затраты на организацию управления за счет возрастания количества единиц диспетчеризации. Продолжительные операции сокращают количество элементов системы, участвующих в алгоритме планирования, но при этом система теряет гибкость. К тому же, промежуточные результаты, получаемые в момент завершения операции, могут быть использованы для восстановления вычислений в случае сбоя. Таким образом, следует выбирать количество и вычислительную сложность операций, допускающую построение эффективного плана выполнения и возможность получения промежуточных результатов.

Сформулируем задачу оптимизации распределенных вычислений с точки зрения многоагентной системы (МАС). Каждый агент МАС является абстракцией физического вычислительного узла параллельного компьютера. Основной задачей функционирования МАС является выполнение множества операций, заданных схемой распределенного приложения, над объектами данных, поступающими для обработки. При этом целью работы МАС является достижение определенных показателей эффективности (сокращение времени обработки, повышение скорости обработки, сокращение потребления ресурсов и т.д.). Все агенты МАС ориентированы на достижение общей цели, которая служит фактором оценки при выработке агентами решений относительно своего дальнейшего поведения в МАС. Пусть информационный объект i начинает обрабатываться приложением в момент времени Si, и заканчивает обработку в момент времени Fi. Целью работы МАС можно зафиксировать минимизацию времени обработки поступивших объектов

$$Z = \min(\sum_{i=1,n} F_i). \quad (1)$$

В качестве убеждений, которым следует агент при осуществлении своих функций, установим возможности выполнения агентом определенных операций обработки. Каждый агент способен выполнять операции в зависимости от характеристик того вычислительного узла, которым он управляет. Сам по себе агент способен самостоятельно ограничить множество выполняемых операций, фиксируя для себя только те из них, которые «удобны» для выполнения данным агентом. Так, например, при наличии у агента специального ресурса (спецпроцессора, аппаратного устройства, системного ресурса) этот агент может выполнять некоторую операцию более эффективно, повышая показатели производительности. С другой стороны, недостаточная загрузка агента операциями приводит к простоям вычислительного ресурса и к ухудшению общей цели МАС.

Зафиксируем для агента промежуточную цель, которой он будет руководствоваться при осуществлении операций и взаимодействии с другими агентами. Пусть общее время функционирования МАС составляет T единиц, а общее время выполнения операций агентом A – T_A единиц. Коэффициент полезности агента

$$W_A = T_A / T, \quad (2)$$

таким образом, индивидуальная цель каждого агента

$$Z_A = \min(Z / W_A). \quad (3)$$

Функционирование агентов заключается в определении множества предпочитаемых операций и их выполнении при готовности данных для соответствующего объекта. После выполнения операции ее результаты предоставляются другим агентам для продолжения работы алгоритма.

3 ВЗАИМОДЕЙСТВИЕ АГЕНТОВ МАС

При реализации динамического планирования широко применяется архитектура «главный-подчиненный». В этом случае главный процесс управляет реализацией вычислительной схемы алгоритма, а подчиненные непосредственно выполняют операции алгоритма. На основе информации от подчиненных процессов главный процесс планирует дальнейшие действия, назначая задачи в соответствии с производительностью и загрузкой подчиненных.

Эта архитектура централизованного управления хорошо зарекомендовала себя при решении сложных задач,

однако, она не лишена недостатков. Основным недостатком является наличие в ней «слабого звена» - главного процесса, - который полностью управляет вычислениями. В случае перегрузки или выхода из строя узла, на котором размещен главный процесс, распределенное приложение может полностью остановиться или столкнуться с простоями подчиненных процессов из-за задержки планирования.

Другая важная функция главного процесса – обеспечение взаимодействия с подчиненными для сбора и анализа информации о состоянии вычислительного процесса и его информационной среды. В настоящее время наиболее популярна техника передачи сообщений между процессами. Взаимодействие процессов через центрального посредника, которым является главный процесс, может привести к задержкам в процессе передачи сообщений и как следствие, к потери производительности приложения.

Несмотря на то, что МАС также используют механизмы передачи сообщений в процессе функционирования системы, агенты МАС являются равноправными участниками решения задачи. При отключении какого-либо агента от процесса поиска решения остальные агенты способны полностью взять на себя его функции без существенного ущерба производительности. Для большей надежности, гибкости и автономности процесса взаимодействия агентов целесообразно применять архитектуры с независимым разделяемым пространством памяти, например, системы «классной доски» (СКД) [4, 5].

Системы с архитектурой «классной доски» (blackboard systems) являются широко применяемой технологией интеграции «сотрудничающих» программных модулей. Цель разработки архитектуры состоит в реализации гибкого механизма решения задач в процессе коллективного обсуждения, присущего группе экспертов-людей, сотрудничающих в процессе решения сложной проблемы. Полученная технология позволяет создавать приложения, являющиеся наиболее сложными системами искусственного интеллекта. Исследования в области МАС повторно обратились к этой архитектуре при переходе от централизованного к распределенному управлению приложениями, поскольку цель МАС также состоит в организации эффективного сотрудничества группы независимых программных объектов.

СКД состоит из трех основных компонентов:

Источники знаний (ИЗ) - независимые вычислительные модули, содержащие набор алгоритмов решения проблемы. ИЗ могут сильно различаться по своей внутренней структуре и используемым вычислительным методам и являются анонимными, поскольку они не взаимодействуют непосредственно друг с другом и явно не знают о том, что в системе присутствуют другие ИЗ. В нашем случае источниками знания являются агенты МАС, реализующие операции распределенного алгоритма.

Доска - глобальный репозиторий данных, содержащий исходные данные и результаты, которые используются на различных стадиях получения решения. Все взаимодействие ИЗ осуществляется посредством изменения содержимого доски. Поскольку основная цель функционирования МАС - управление вычислительным процессом, то в качестве содержимого доски используется информация о ходе вычислений.

Диспетчер – компонент, принимающий во время выполнения решения о ходе вычислительного процесса и планирующий использование вычислительных ресурсов. Диспетчер отделен от ИЗ и является отдельным модулем. Архитектура на базе диспетчера обеспечивает целостность репозитария, поскольку требуется жесткий механизм синхронизации для его своевременного обновления после изменений и поддержания информационной структуры репозитария. Однако, для перехода к полностью распределенной системе, целесообразно передать отдельные функции диспетчера (добавление, удаление, поиск информации на доске) в состав поведения ИЗ. Это позволит повысить устойчивость системы, однако потребует реализации механизмов разделения доступа и синхронизации работы с доской.

Архитектура МАС, ориентированной на взаимодействие с СКД, приведена на рис. 2.

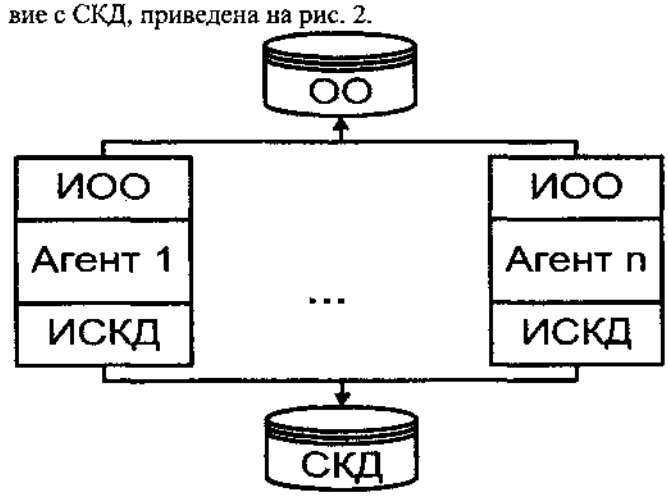

Рис.2. Архитектура МАС управления распределенными вычислениями.

Агент снабжен двумя интерфейсами для взаимодействия с системами на базе СКД. Первый интерфейс – ИОО (интерфейс объектов и операций) – предназначен для взаимодействия с базой данных вычислительных операций и объектов данных. Вычислительные операции реализуются в виде загружаемых исполняемых объектов (статическая часть ОО), которые встраиваются в агент и исполняются им [6]. Контекст выполнения операции формируется динамически по дескриптору операции, содержащему сведения об операции и ее данных. Данные для выполнения операции, а также ее результаты сохраняются в базе данных объектов (динамическая часть ОО).

Интерфейс СКД служит для обмена управляющей информацией между агентами МАС. Сама СКД построена из трех таблиц, хранящих дескрипторы операций, входящих в структуру параллельного алгоритма. В начале работы первая таблица СКД – готовые операции – заполняется дескрипторами операций, являющихся начальными для запуска параллельного приложения. По мере запуска на агентах дескрипторы перемещаются во вторую таблицу СКД – исполняемые операции. После завершения выполнения они поступают в третью таблицу СКД – завер-

шенные операции. Завершение операции также приводит к появлению дескрипторов для новых готовых операций в первой части СКД.

4 ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ И ВОССТАНОВЛЕНИЕ ВЫЧИСЛЕНИЙ

Надежность вычислений предполагает получение результата в случае отказа некоторых вычислительных узлов. Отказ, как правило, выражается в отсутствии связи с вычислительным узлом и невозможностью планировать вычисления для данного узла. Отказавший узел должен своевременно исключаться из алгоритма планирования, а сам алгоритм учитывать изменение конфигурации вычислительной системы.

Традиционная схема с централизованным управлением является уязвимой к отказам центрального диспетчера вычислений – главного процесса. В связи с этим для повышения устойчивости вычислений более эффективным использовать равноправное взаимодействие с неявной коммуникацией между узлами. В этом случае каждый узел самостоятельно определяет для себя правила поведения, способствующие достижению его конкретной цели. Совокупность узлов реализует более общую цель, являющуюся результатом действия индивидуальных агентов.

Компонентное построение параллельного алгоритма и его деление на вычислительные операции существенно упрощают задачу сохранения результата в процессе вычислений. Каждый вычислительный компонент способен получать исходные данные и сохранять результаты во внешней среде, в качестве которой используется СКД.

Дескрипторы операций, размещаемые в СКД, служат сигналами для агентов, сообщающими о том, что некоторые операции выполнены и их данные готовы для передачи в другие операции. Агенты используют таблицы дескрипторов для синхронизации своей деятельности. Для поддержания целостности таблиц дескрипторов вводятся механизмы блокировки, обеспечивающие раздельный доступ для агентов. Каждый агент может выполнять три основные операции: искать в таблице готовый дескриптор из множества предпочитаемых, брать первый готовый дескриптор, если предпочитаемых нет, или корректировать таблицу.

Корректировка таблицы возникает в двух случаях: если агент аварийно завершил работу во время вычислений или во время работы с таблицами СКД. Если агент отказал во время вычислений, то результаты данного этапа не будут записаны в СКД, а дескриптор соответствующей операции не будет перемещен из таблицы исполняемых в таблицу завершенных операций. В этом случае контроль над таблицей исполняемых операций заключается в слежении за временем нахождения дескриптора в таблице. Если это время превышает установленный лимит, то считается, что агент не выполнил операцию и ее следует перепланировать. Если агент опоздал, и не обнаружил дескриптор в таблице исполняемых операций, он просто сбрасывает результаты вычислений, объявляя их недействительными.

Второй случай отказа проявляется, когда агент работает с таблицами СКД. В этом случае результат, полученный на этапе расчета, является действительным. Поскольку агент блокирует СКД, здесь также используется механизм таймаута, который позволяет отменить блокировку после его истечения. Все изменения, которые проведены до этого момента, считаются действительными.

5 ЗАКЛЮЧЕНИЕ

Использование многоагентной архитектуры для управления вычислениями позволяет реализовать гибкую систему управления, основанную на интеллектуальных коллективных алгоритмах решения оптимизационной задачи. Равноправное участие агентов и выполнение ими всех этапов планирования дает возможность сохранить управляемость системы и повысить ее надежность в условиях возможных отказов вычислительных узлов. Использование СКД ведет к реализации неявного взаимодействия агентов, допускающего эффективный обмен информацией в ходе реализации сложных алгоритмов управления.

ЛИТЕРАТУРА

[1] Городецкий, В.И. Многоагентные системы (обзор) / В.И. Городецкий, М.С. Грушинский, А.В. Хабалов // Новости искусственного интеллекта. – 1997. – №1. - С. 64-117.

[2] Тарасов, В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте / В.Б. Тарасов // Новости искусственного интеллекта. - 1998. - № 2. - С. 5-63.

[3] Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. - СПб.: БХВ-Петербург, 2002. - 608 с.

[4] Corkill, D. D. Blackboard systems // D. D. Corkill / AI Expert. - 1991 – Vol. 6(9). – pp .40–47.

[5] Jagannathan. V. et al. // Blackboard Architectures and Applications. - Academic Press, 1989.

[6] Отвагин, А.В. Инструментальные средства параллельной обработки потоков изображений в системах технического зрения / А.В. Отвагин, А.А. Дудкин // Искусственный интеллект. - 2006. - № 3. - С. 623-633.