Моделирование синхронизации параллельных потоков при выполнения гибридных MPI+threads программ

А. В. Табаков[1], А. А. Пазников[2]
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
[1]komdosh@yandex.ru, [2]apaznikov@gmail.ru

***Аннотация*. Параллельные вычисления являются одним из наиболее приоритетных направлений в компьютерных науках. Основным средством параллельной обработки информация является распределенные вычислительные системы (ВС) – композиции элементарных машин, взаимодействующих через коммуникационную среду. Современные распределенные ВС реализуют параллелизм уровня потоков (thread-level parallelism, TLP) в рамках одного вычислительного узла (многоядерная ВС с общей памятью), так параллелизм уровня процессов (process-level parallelism, PLP) для всей распределенной ВС. Основным средством разработки параллельных программ для таких систем является стандарт Message Passing Interface (MPI). Необходимость создания масштабируемых (scalable) параллельных программ, эффективно использующих вычислительные узлы с общей памятью, определила развитие стандарта MPI, который сегодня поддерживает создание гибридных многопоточных MPI-программ. Под гибридной многопоточной MPI-программой понимается объединение вычислительных возможностей процессов и потоков. В стандарте определяется четыре вида многопоточности: Одиночка (Single) – один поток выполнения в рамках процесса; Воронка (Funneled) – несколько потоков в рамках процесса, при этом только главный поток может выполнять коммуникационные MPI операции; Серийный (Serialized) – только один поток в один момент времени может выполнять вызов MPI функций; Множественный (Multiple) – каждый поток программы может выполнять MPI функции в любой момент времени. Основной задачей при реализации режима Multiple является необходимость синхронизации коммуницирующих потоков в рамках каждого процесса. В настоящей работе представлен обзор работ, рассматривающие проблему синхронизации процессов, запущенных на удалённых машинах и синхронизацию внутренних потоков программы. Предложен метод синхронизации потоков на основе очередей с ослабленной семантикой выполнения операций.**

Ключевые слова: распределенные вычислительные системы; MPI; параллельные программы; распределенные структуры данных; гибридные параллельные программы; MPI+threads

Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 19-07-00784, 18-57-34001 и при поддержке Совета по грантам Президента РФ для государственной поддержки молодых российских ученых (проект СП-4971.2018.5)

I. Введение

К многоядерным и многопроцессорным вычислительными системам (ВС) относится широкий класс систем – от встроенных систем и мобильных устройств до кластерных вычислительных систем, систем с массовым параллелизмом, GRID-систем и облачных ВС. Алгоритмический и программный инструментарий параллельного программирования [1] является базой при построении современных систем обработки больших данных, в том числе машинного обучения и искусственного интеллекта. Основным классом систем, применяемых для высокопроизводительной обработки информации, являются распределенные ВС – коллективы элементарных машин, взаимодействующих через коммуникационную среду.

При разработке параллельных программ для распределенных ВС стандартом де-факто является модель передачи сообщений, представленная в первую очередь стандартом MPI (Message Passing Interface). Сегодня использование исключительно модели передачи сообщений (MPI everywhere) может быть недостаточным для разработки эффективных масштабируемых MPI-программ. В связи с этим перспективным является подход, предполагающий использование MPI для взаимодействия между вычислительными узлами и систем поддержки многопоточности (PThreads, OpenMP, Intel TBB) внутри многоядерных вычислительных узлов. Основной задачей при реализации поддержки гибридного режима в библиотеках MPI является организация масштабируемого доступа потоков к разделяемым структурам данных (идентификаторы контекста, виртуальные каналы, очереди сообщений, пулы запросов и др.).

Стандартом MPI 3.1 предусматривается следующие режимы для многопоточных MPI-программ:

- однопоточный (MPI_THREAD_SINGLE) – один поток выполнения в рамках процесса;
- воронка (MPI_THREAD_FUNNELED) – несколько потоков в рамках процесса, при этом только главный поток может выполнять коммуникационные MPI операции;

- сериализуемый (MPI_THREAD_SERIALIZED) – только один поток в один момент времени может выполнять вызов MPI функций;
- множественный (MPI_THREAD_MULTIPLE) – каждый поток программы может выполнять MPI функции в любой момент времени.

Одной из реализаций гибридной многопоточной MPI-программы в режиме MPI_THREAD_MULTIPLE является библиотека MPICH версии CH4, в которой определены стандарты для использования неблокируемых структур данных. В данном режиме доступно использование двух типов синхронизации: trylock – при котором программа циклично пытается захватить mutex и обратиться к очереди; handoff – потокобезопасная очередь, при обращении к которой вызывается активное ожидание элемента потоком.

II. Методы синхронизации потоков в гибридных MPI-программах

При разработке многопоточных MPI-программ главной проблемой становится синхронизация обращений потоков к разделяемым структурам данных, которые хранятся в общей памяти. Представителями данных структур являются: массивы, списки, очереди, стеки, деревья и графы.

В [2] рассматривается задача обеспечения потокобезопасного выполнения основных MPI-функций. Предлагается алгоритм вычисления целочисленного идентификатора контекста (context id), необходимого для создания нового MPI-коммуникатора. В гибридных MPI-программах данная процедура может выполняться в режиме MPI_THREAD_MULTIPLE, при котором каждый поток может выполнять информационные обмены. В однопоточном режиме идентификатор выделяется для каждого процесса на основе глобальной структуры, содержащий допустимые идентификаторы процесса. Каждый раз, когда новый процесс подключается к коммуникатору, ему выдается новый идентификатор с помощью операции MPI_Allreduce. Данная операция работает с целочисленным идентификатором как с массивом бит, реализуя побитовое умножение всех существующих в коммуникаторе идентификаторов, затем вычисляется последняя позиция выставленного в единицу бита, которая и служит идентификатором. Описанный подход требует модификации для многопоточной MPI-программы. Так как потоки выполняются одновременно, порядок захвата мьютекса, защищающий изменение идентификатора в глобальной памяти, в MPI может быть различным, в том числе одновременным, что может привести к взаимной блокировке в методе MPI_Allreduce. Представленный в статье алгоритм решает данную задачу с помощью дополнительных локальных копий глобальных переменных, которые распределены по процессам.

В [3] авторы предлагают четыре подхода к синхронизации потоков в MPI+threads-программах: Global, Brief Global, Per Object, Lock-free. Метод Global предполагает организацию единой блокировки для всех процессов, которая используется для всех MPI-функций (кроме функций, которые могут заблокировать выполнение коммуникационных операций). В рамках Brief Global также используется единая блокировка, однако, в отличие от Global, критическая секция организуется только в тех функциях, которые обращаются к разделяемым структурам данных, при этом другие функции могут выполняться параллельно в разных потоках. Реализация данного подхода требует значительно больших усилий, нежели Global, так как требуется более тщательный анализ кода. Per Object – раздельные критические секции для различных объектов и классов объектов, например может использоваться несколько критических секций для обращения к одному и тому же процессу; Lock-free – в данном подходе синхронизация потоков происходит за счёт атомарных операций, реализованные возможностями процессора.

В статье [4] показано, что в существующих реализациях режима MPI_THREAD_MULTIPLE применяются несправедливые (unfair) алгоритмы синхронизации потоков, что приводит к снижению эффективности выполнения информационных обменов. Используемые в MPI-библиотеках алгоритмы блокировки потоков не гарантирует справедливый захват критических секций потоками, что может вызывать монополизацию доступа одним из потоков. Кроме того, в вычислительных узлах с неоднородным доступом к памяти (архитектура NUMA) высокая латентность доступа к удаленным сегментам памяти увеличивает время передачи права выполнения критической секции (интервал между освобождением блокировки и захватом её другим потоком) между потоками. В работе предложено два алгоритма синхронизации потоков, обеспечивающие справедливое выполнение критических секций и снижающие накладные расходы в узлах на основе архитектуры NUMA. Первый алгоритм реализует метод блокировки потоков Ticket Lock, позволяющий организовать очередь потоков по времени вызова операции. Второй алгоритм является доработанной версией первого, с тем отличием, что некоторые задачи имеют повышенный приоритет и выполняются раньше менее приоритетных задач.

В [5] проведен анализ справедливости захвата мьютексов Pthread при реализации гибридной модели MPI+threads. Опираясь на результат, представленный в работе [4], создана улучшенная версия алгоритма блокировки потоков на основе Ticket Lock и предложен алгоритм на основе алгоритма CLH- и его модификация с добавлением приоритета (CLH-LPW) для захвата блокировки. Данный подход позволяет эффективнее использовать критические секции, уменьшая голодание потоков.

В [6] предложено несколько методов оптимизации выполнения гибридных MPI+threads программ. В рамках первого метода создана потокобезопасная хеш-таблица на основе блокировок, которую предлагается использовать для поиска соответствий между полученными сообщениями и соответствующими запросами. Второй метод направлен на оптимизацию планирования потоков. Авторами разработан планировщик легковесных потоков (light-weight threads), использующий битовый массив для индикации работающих потоков. В рамках третьего метода предлагается изменить пул пакетов сообщений,

разделив централизованный пул в приватные пулы для каждого потока. Изначально каждый поток имеет фиксированное число пакетов для обработки, однако в ходе исполнения программы применяется подход work-stealing, позволяющий незадействованным потокам обрабатывать пакеты из чужих пулов.

В [6] авторами предложено два алгоритма оптимизации синхронизации потоков. Суть первого алгоритма заключается в уменьшении количества потоков для выбора из пула ожидающих потоков в рамках одной критической секции. Другие потоки должны ожидать выполнения их запроса за пределами критической секции. Это позволит уменьшить время поиска свободного потока для выполнения операции, так как снижается время задержки и нагрузка на сеть. Второй алгоритм реализует выбор потока для выполнения следующей операции, отдается предпочтение только что завершившим выполнять работу потокам, а ожидающие потоки встают в очередь с меньшим приоритетом.

III. Использование очередей с ослабленной семантикой выполнения операций

На данный момент популярной реализацией стандарта MPI является свободно распространяемая библиотека MPICH. Текущая версия CH4, в качестве рабочей очереди с задачами, использует библиотеку izem, которая предоставляет потокобезопасные структуры данных, такие как потокобезопасная очередь и др., а также различные механизмы синхронизации. Предлагается замена потокобезопасной рабочей очереди с задачами из библиотеки izem заменить на потокобезопасную очередь с ослабленной семантикой выполнения операций Multiqueues, в которой улучшен механизм обращений к элементам очереди [8].

В основе подхода ослабления семантики выполнения операций лежит компромисс между масштабируемостью (производительностью) и корректностью семантики выполнения операций. Предлагается ослабить семантику выполнения операций для повышения возможности масштабирования. Например, при поиске максимального элемента в массиве, поток может пропустить, заблокированные другими потоками, участки массива для повышения производительности операции поиска, при этом теряется точность выполнения данной операции.

Использование очереди с ослабленной семантикой выполнения операций Multiqueues позволит избежать возникновения узких мест при синхронизации потоков (рис. 1). В отличие от большинства существующих неблокируемых потокобезопасных структур данных и алгоритмов блокировки, где существует единая точка выполнения операций над структурой, в ослабленных структурах данных используется набор простых последовательных структур, композиция которых рассматривается как логически единая структура. Вследствие этого увеличивается количество возможных точек обращений к данной структуре. Данный подход позволит достичь значительно большей пропускной способности, по сравнению с существующими структурами данных.

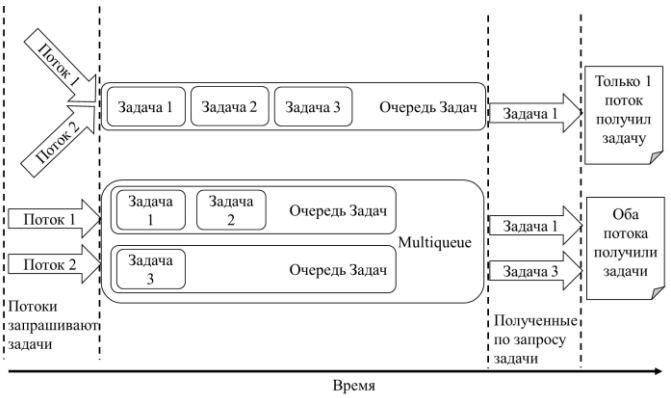

Рис. 1. Сравнение реализаций очередей задач Izem (верхняя очередь задач) и Multiqueue

Заключение

В представленных работах описаны и предложены подходы для увеличения пропускной способности при синхронизации потоков в гибридных многопоточных MPI-программах (модель MPI+threads). Предлагается использовать масштабируемую потокобезопасную очередь с ослабленной семантикой, которая позволит сократить накладные расходы на синхронизацию потоков при выполнении операций с рабочей очередью задач.

Выражение признательности

Выражаем благодарность за предоставленные вычислительные ресурсы Новосибирский государственный университет, а также Владислава Калюжного за поддержку в работе с вычислительным ресурсом.

Список литературы

[1] Хорошевский В.Г., Курносов М.Г., Мамойленко С.Н., Павский К.В., Ефимов А.В. Пазников А.А Масштабируемый инструментарий параллельного мультипрограммирования пространственно-распределенных вычислительных систем // Вестник СибГУТИ. 2011. № 4. С. 3-19.

[2] Gropp W., Thakur R. Thread-safety in an MPI implementation: Requirements and analysis // Parallel Computing. 2007. №. 9. С. 595-604.

[3] Balaji P. et al. Fine-grained multithreading support for hybrid threaded MPI programming // The International Journal of High Performance Computing Applications. 2010. №. 1. С. 49-57.

[4] Amer A. et al. MPI+ threads: Runtime contention and remedies //ACM SIGPLAN Notices. 2015. №. 8. С. 239-248.

[5] Amer A. et al. Locking aspects in multithreaded MPI implementations // Argonne National Lab., Tech. Rep. P6005-0516. 2016.

[6] Towards millions of communicating threads / Dang H. V., Snir M., Gropp W. // Proceedings of the 23rd European MPI Users' Group Meeting. ACM, 2016. С. 1-14.

[7] Dang H. Advanced thread synchronization for multithreaded MPI implementations // Cluster, Cloud and Grid Computing (CCGRID), 2017. С. 314-324.

[8] Tabakov A, Paznikov A. Algorithms for Optimization of Relaxed Concurrent Priority Queues in Multicore Systems. // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2019, С. 360-365.

[9] Табаков А.В., Пазников А.А. Алгоритмы оптимизации потокобезопасных очередей с приоритетом на основе ослабленной семантики выполнения операций // Известия СПбГЭТУ «ЛЭТИ», 2018. С. 42-49