УДК 681.03.06

И.Г. Данилов

СОВРЕМЕННЫЕ ПОДХОДЫ К СОЗДАНИЮ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ ДЛЯ МНОГОМАШИННЫХ КОНФИГУРАЦИЙ С ЭМУЛЯЦИЕЙ ОБЩЕЙ ПАМЯТИ

Представлены модели облачных вычислений и транзакционной памяти возникли относительно недавно и привлекают все большее внимание со стороны специалистов в области компьютерных наук. Транзакционная память предоставляет неблокирующий синхронизационный управляющий механизм для многопоточных приложений. В модели облачных вычислений пользователю по сети предоставляется информационный ресурс. В данной работе рассматриваются некоторые особенности совмещения данных подходов для достижения цели: улучшение и упрощение использования вычислительных ресурсов.

Облачные вычисления; транзакционная память; распределенная транзакционная память.

I.G. Danilov

MODERN APPROACHES TO CREATING MULTITHREADED APPLICATIONS FOR CLUSTERS WITH DISTRIBUTED SHARED MEMORY EMULATION

The cloud computing and transactional memory paradigms have recently emerged and have been receiving increasing attention from the computer science researchers. Transactional memory provides non-blocking concurrency control mechanism for multithreaded applications. In the cloud computing paradigm information resource provided for end-user over network. This paper described some composition features of these approaches to achieving the goal which sought by both approaches: the improvement and simplification of computing resources utilization.

Cloud computing; transactional memory; distributed transactional memory.

В ближайшее время в области высокопроизводительных вычислений должен произойти революционный переход к использованию вычислительных мощностей от петафлопс- до exascale-диапазона [1]. Для более эффективного использования этих вычислительных супер-ресурсов в будущем уже сейчас необходимо пересматривать существующие парадигмы параллельного программирования. Одним из современных подходов, пытающихся решить данную проблему, является модель облачных вычислений. Реализованные на данный момент решения уровня платформы (PaaS) предлагают сервисы по запуску приложений «в облаке». Однако класс этих приложений довольно узок и требует применения специального прикладного программного интерфейса (API), что накладывает ограничения на использование уже существующего программного обеспечения, написанного для исполнения на многопроцессорных/многоядерных платформах. Возможным решением данной проблемы является создание облачной SMP-машины для многопоточных приложений. Центральным вопросом при создании такой машины является общая память, которая в данном случае должна быть распределена между исполнительными узлами. В плане исследования актуальным является применение нетрадиционных моделей разделяемой памяти такой, как транзакционная память, в общем случае предоставляющей неблокирующие механизмы синхронизации и обеспечивающей определенную консистентность данных. Использование современных очень быстрых межсистемных коммуникаций, таких как Infiniband, позволяет уменьшить накладные расходы при реализации протокола когерентности данных кэшей и других различных коммуникационных протоколов системы.

Кластер как единая система и распределенная разделяемая память. До настоящего времени проблема запуска многопоточных приложений на кластерных/многомашинных платформах в той или иной форме решалась с помощью двух основных подходов: эмуляцией единого образа системы [2] (*Single System Image, SSI*) или распределенной разделяемой памяти (*Distributed Shared Memory, DSM*).

Вопрос распределенной разделяемой памяти был очень хорошо изучен в последнюю декаду ушедшего столетия. Были предложены различные реализации: (1) на основе страниц памяти [3]; (2) на основе специальных разделяемых переменных, что требует языковой поддержки; (3) на основе объектно-ориентированных технологий. Наибольшее распространение получила страничная разделяемая память, которая предоставляет последовательную консистентность данных, используя протокол «один писатель/много читателей» на уровне страничного механизма памяти. Однако ряд ключевых проблем [4] данного подхода не позволил получить приемлемой для реальных систем производительности.

Облачные вычисления и MapReduce. Модель облачных вычислений возникла относительно недавно, но уже успела привлечь большое внимание со стороны специалистов в области компьютерных наук [5]. В целом облачные вычисления базируются на идее представления услуг (сервисов) и на трех подходах: IaaS (инфраструктура как сервис, Infrastructure As A Service), PaaS (платформа как сервис, Platform As A Service =IaaS + специальная программная платформа, например, Google AppEngine), SaaS (приложение как сервис, Software As A service).

В случае IaaS пользователю предоставляется виртуализированная среда на базе некоторых серверов (в облаке). Пользователю предоставляется виртуальная машина (или несколько), внутри которой есть все возможности для установки сначала ОС, а потом уже настройки необходимого ПО. Предоставляемые аппаратные ресурсы могут быть гибко изменены в сторону увеличения или уменьшения. Однако верхним пределом увеличения ресурсов являются границы физического сервера, на котором в данный момент размещена виртуальная машина.

В случае PaaS пользователю предоставляется не просто виртуальная машина, а прикладные библиотеки и API. Пользователю предоставляется возможность запускать собственные приложения, которые имеют возможность гибко получать ресурсы по запросу (например, Google AppEngine).

В случае SaaS пользователь имеет доступ только к конкретному приложению. Ни к API, ни к программному коду приложения пользователь доступа не имеет. Наиболее известным примером такого сервиса является Google Docs для работы с офисными документами.

Одним из широко распространённых примеров специального прикладного программного интерфейса, предоставляемого на уровне PaaS, является MapReduce. MapReduce – функциональная модель программирования, которая позволяет автоматическое распараллеливание и исполнение программ на очень больших масштабируемых кластерах, решая задачу распределения данных, балансировки нагрузки и отказа отдельных вычислительных узлов. Данная модель включает в себя две последовательные фазы: (1) в map-фазе данные распределяются по рабочим узлам для предварительной обработки данных; (2) в reduce-фазе происходит сбор на главном узле предварительно обработанных данных и формирование окончательного результата. Основная проблема, которая мешает широкому применению данного подхода в системах облачных вычислений, – это сложность преобразования уже имеющегося решения задачи, которое, во-первых, требует определённой узкой квалификации программиста, и, во-вторых, может привести к значительному снижению производительности в ряде случаев [6].

Транзакционная память. Принципы многопоточного программирования начинают играть все более важную роль при создании программного обеспечения для современной вычислительной техники любого типа. Однако стоит отметить, что кроме трудностей, свойственных последовательному программированию, в параллельном программировании перед разработчиком возникает ряд дополнительных трудноразрешимых задач. Одна из них – проблема координации взаимодействия параллельно выполняемых потоков, и – прежде всего – координация доступа к совместно разделяемым ресурсам, основным из которых является память. Существующие средства явной синхронизации потоков (например, блокировки, семафоры) требуют от программиста глубокого понимания всех процессов, лежащих в основе данных средств, а программы, написанные с их помощью, трудно разрабатывать, отлаживать и сопровождать. Альтернативным вариантом является использование других механизмов синхронизации доступа к разделяемым ресурсам, одним из которых является транзакционная память (*transactional memory, TM*).

Транзакционная память предоставляет механизм, позволяющий частям программы выполняться в изоляции, независимо от других параллельно выполняемых задач [7]. В общем случае транзакция определяется как группа последовательных операций, которая представляет собой логическую единицу работы с данными. Транзакции гарантируют, что все запросы к разделяемым ресурсам произведут тот же самый результат, как если бы они выполнялись последовательно в некотором порядке. Данный механизм базируется на двух основных принципах: управление версиями данных и обнаружение конфликтов. Если транзакция, выполняемая в данном потоке, выполняется успешно (т.е. никакая другая транзакция не использовала общий с данной транзакцией ресурс), то она фиксируется и все изменения, производимые с данными, считаются видимыми другим транзакциям системы. Иначе – если произошёл конфликт доступа к ресурсу – производится откат назад, изменения в данных восстанавливаются и транзакция отмечается для повторного выполнения. При этом политика выбора откатываемой транзакции и условия перевыполнения транзакции влияют на производительность всей системы в целом. В общем случае системы транзакционной памяти обеспечивают атомарность и изолированность параллельно выполняемых задач; согласованность и долговечность не гарантируются. Реализация систем транзакционной памяти может быть чисто программной (*Software Transactional Memory, STM*), аппаратной (*Hardware Transactional Memory, HTM*) и гибридной, сочетающей возможности STM и HTM. Следует отметить, что транзакции сами по себе не заменяют всю синхронизацию в параллельной программе. Для этого вводятся дополнительные механизмы, наподобие механизма предохранителей, который не дает транзакции начаться до тех пор, пока соответствующий ей предикат не примет значение *истина*.

Распределенная транзакционная память. В настоящее время транзакционная память – область активных научных исследований. Основные вопросы, которым уделяется внимание: производительность, вложенность транзакций, политики выбора откатываемой транзакции, гранулярность транзакции, взаимодействие с существующими средствами синхронизации и другими библиотеками, сильная и слабая атомарность. Однако большинство исследований [8] в данной области рассматривает не распределённые, а кэш-когерентные системы или системы с общей памятью [9], что уменьшает возможности масштабирования данного подхода. Лишь совсем недавно стал рассматриваться вопрос о расширении границ использования транзакций для их применения в широко масштабируемых распределённых многомашинных конфигурациях [10].

Системы распределенной транзакционной памяти (Distributed Software Transactional Memory, DSTM) можно классифицировать в зависимости от применяемой при реализации программной модели [11]: 1) для модели потока управления характерны «неподвижные» объекты данных, доступ к которым транзакции получают посредством вызова удаленных процедур; 2) с другой стороны, в модели потока данных неподвижны сами транзакции, а объекты данных перемещаются по сети к запрашивающей их транзакции, при этом консистентность данных гарантируется протоколом когерентности кэшей.

Многие современные системы распределенной транзакционной памяти используют модель потока данных и обычно составлены из двух важных элементов. Первый – стратегия разрешения конфликта. Две транзакции находятся в конфликте, если они обе получили доступ к одному и тому же объекту и, по крайней мере, одна из них – на запись. Основные существующие реализации DSTM просто откатывают одну из конфликтных транзакций, исходя из определенных критериев. Второй элемент – распределенный протокол когерентности кэшэй (cache-coherence протокол, cc-протокол). Когда транзакция пытается получить доступ к объекту по сети, распределенный cc-протокол должен определить местонахождение последней актуальной версии объекта и переместить его копию к запрашивающей транзакции.

Транзакционная память «в облаке». Одной из главных проблем, которые в настоящее время пытается решить компьютерное сообщество для того, чтобы полностью раскрыть потенциал облачных технологий и повысить эффективность утилизации вычислительных ресурсов, является разработка программных моделей и средств, упрощающих реализацию параллельных облачных приложений. Уже предложенные модели (такие, как MapReduce) спроектированы для различных целей и имеют как достоинства, так и недостатки. Одним из основных недостатков является сложность разработки приложений в рамках данных моделей. Возможен альтернативный подход, который поможет решить данные проблемы, – создание облачной SMP-машины, построенной на принципах транзакционной памяти и предоставляющей сервис прозрачного выполнения многопоточных приложений в облаке.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Patrick Thibodeau* Scientists, IT community await exascale computers // Computerworld. URL: http://www.computerworld.com/s/article/345800/Scientists_IT_Community_Await_Exascale_Computers (дата обращения: 5.04.2011).
2. *Single System Image* // Материал из Википедии – свободной энциклопедии. URL: http://en.wikipedia.org/wiki/Single_system_image (дата обращения: 7.04.2011).
3. *Amza C., Cox A. L., Dwarkadas S., Keleher P., Lu H., Rajamony R., Yu W., Zwaenepoel W.* TreadMarks: Shared memory computing on networks of workstations // In: IEEE Computer. – 1996. – Vol. 29 (1). – P. 18-28.
4. *Protic J., Tomasevic M., Milutinovic V.* Distributed Shared Memory: Concepts and Systems // In: *IEEE Parallel Distrib. Tech.* – 1996. – Vol. 4 (2). – P. 63-71.
5. *Хашковский В.В., Данилов И.Г.* Применение облачных вычислений и GRID-технологий для организации коллективного использования вычислительных ресурсов в научно-исследовательской и учебной работе // Известия ЮФУ. Технические науки. – 2011. – № 1 (114). – С. 139-144.
6. *Ranger C., Raghuraman R., Penmetsa A., Bradski G., Kozyrakis C.* Evaluating mapreduce for multi-core and multiprocessor systems // In Proc. of the International Symposium on High-Performance Computer Architecture (HPCA-13), 10-14 February 2007, Phoenix, Arizona, USA. – P. 13-24.
7. *Larus J., Kozyrakis C.* Transactional Memory // In: Communications of the ACM. – 2008. – Vol. 51 (7). – P. 80-88.

8. *Romano P., Rodrigues L., Carvalho N., Cachopo J.P.* Cloud-TM: harnessing the cloud with distributed transactional memories // In Proc. ACM SIGOPS Operating Systems Review. – 2010. – Vol. 44 (2). – P. 1-6.
9. *Harris T., Fraser K.* Language support for lightweight transactions // In Proc. ACM SIGPLAN Conf. on Object-Oriented Prog., Systs., Langs., and Apps (OOPSLA 2003), 26-30 October 2003, Anaheim, CA, USA. – P. 388-402.
10. *Bocchino R. L., Adve V. S., Chamberlain B. L.* Software transactional memory for large scale clusters // In Proc. ACM SIGPLAN Symp. on Principles and Practice of Parallel Prog. (PPOPP 2008), 20-23 February 2008, Salt Lake City, UT, USA. – P. 247-258.
11. *Herlihy M., Sun Y.* Distributed transactional memory for metric-space networks // In Proc. International Symposium on Distributed Computing (DISC 2005), 26-29 September 2005, Cracow, Poland. – P. 324-338.

Статью рекомендовал к опубликованию д.т.н., профессор Н.И. Витиска.

Данилов Игорь Геннадьевич – Технологический институт федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге; e-mail: vainamon@hpcmop.ru; 347928, г. Таганрог, пер. Некрасовский 44; тел.: 88634371773; кафедра математического обеспечения и применения ЭВМ; аспирант.

Danilov Igor Gennad'evich – Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education "Southern Federal University"; e-mail: vainamon@hpcmop.ru; 44, Nekrasovskiy, Taganrog, 347928, Russia; phone: +78634371773; the department of software engineering; post-graduate student.

УДК 004.896

Ю.В. Чернухин, Р.В. Сапрыкин, П.А. Бутов, Ю.С. Доленко

МОБИЛЬНАЯ РОБОТОТЕХНИЧЕСКАЯ ПЛАТФОРМА С ПЕРЕСТРАИВАЕМОЙ ГЕТЕРОГЕННОЙ СИСТЕМОЙ УПРАВЛЕНИЯ

Рассматриваются недостатки использования робототехнических платформ с жесткой архитектурой для реализации на их базе бортовых систем, реализующих эвристические алгоритмы управления. Предложена гетерогенная перестраиваемая архитектура робототехнической платформы, объединяющая PC-совместимый микрокомпьютер, используемый для реализации высокоуровневой логики системы управления и ПЛИС для реализации низкоуровневой логики и специфических аппаратных решений. Рассмотрены первый прототип разрабатываемой платформы и первые экспериментальные исследования на ее базе с использованием виртуальной моделирующей среды NAME.

Мобильный робот; перестраиваемая структура; гетерогенная система управления; ПЛИС.

U.V. Chernukhin, R.V. Saprykin, P.A. Butov, U.S. Dolenko

MOBILE ROBOTIC PLATFORM WITH CONFIGURABLE HETEROGENEOUS CONTROL SYSTEM

Shortcomings of robotic platforms with hard architecture for on-board systems implementing heuristic control algorithms are reviewed. A robotic system with heterogeneous configurable architecture is offered, combining PC-compatible microcomputer used to implement high-level control system logic and an FPGA for low-level logic and specific hardware solutions implementation. The first prototype of the platform being designed is analyzed and first experimental research on its base with NAME virtual modelling system is discussed.

Mobile robot; configurable architecture; heterogeneous control system; FPGA.