АРХИТЕКТУРА ПАРАЛЛЕЛЬНОЙ СУБД НА ПЛАТФОРМЕ GPU-КЛАСТЕРА

Р.Ш. Минязев

Казанский национальный исследовательский технический университет им. А.Н. Туполева

E-mail: r.minyazev@gmail.com

Предлагается архитектура параллельной СУБД, функционирующей на платформе GPU-кластера, каждый вычислительный узел утяжелен двумя GPU-акселераторами. На GPU выполняются операции SELECT, PROJECT, JOIN. База данных делится горизонтально с использованием операции хеширования по ключевому полю каждого отношения. Каждый GPU-акселератор хранит в своей памяти свой фрагмент базы данных. Вычислительный узел хранит все фрагменты исходной базы данных в своей оперативной памяти. Рассматривается случай работы с консервативными данными, объем которых соизмерим с объемом оперативной памяти вычислительного узла.

Постановка задачи

Серьезной проблемой для высокопроизводительных параллельных реляционных СУБД является размещение данных между узлами и повышение скорости обработки сложных запросов с большим числом операции соединения [1, 2]. Один из перспективных подходов к повышению производительности (как уменьшения времени обработки запросов) СУБД – использование GPU-акселераторов, для исполнения наиболее трудоемких реляционных операций [3]. Рассматривается случай консервативных СУБД (преобладают операции чтения данных без модификации), ориентированных на аналитическую обработку накопленных данных. Для таких систем характерно исполнение множества сложных запросов с большим числом операций соединения [1].

Предлагаемая архитектура параллельной СУБД, получившая название Clusterix-G, представлена на рис. 1. Система функционирует на платформе GPU-кластера. База данных (БД) делится горизонтально путем хеширования по ключевому полю на 2n частей. При этом каждый вычислительный узел хранит всю расхешированную БД целиком. Рассматривается случай работы с базами данных, объем которых сравним с объемом оперативной памяти, установленной в вычислительном узле.

В системе выделяется два типа логических узлов: узел управления MGM и рабочий узел NODE. На узле управления функционируют следующие программные модули:

- MONITOR – получение запросов от пользователей и передача обратно результатов обработки, управление всеми другими модулями системы. Все поступающие запросы собираются во внешней очереди, откуда передаются модулю TRANS;
- TRANS – транслирует запрос в пакет подзапросов для исполнения на рабочем узле NODE, сообщает о готовности очередного пакета модулю MONITOR;
- SORT – получает частичные результаты от модулей IO-JOIN, формирует из них конечный результат в виде отношения, в котором устраняет дубликаты и над которым при необходимости выполняет операции сортировки и агрегации. Результат передается модулю MONITOR для отправки пользователю.

На рабочем узле NODE функционирует модуль IO-JOIN. Он получает оттранслированный пакет запросов от MONITOR (одинаковый для каждого модуля IO-JOIN),

помещает его в свою внутреннюю очередь в виде набора подзапросов, выбирает очередной подзапрос из внутренней очереди и запускает два одинаковых ядра (KERNEL_A, KERNEL_B) на GPU-акселераторах. Каждое ядро (функция) выполняет подзапрос над своей частью исходной БД.

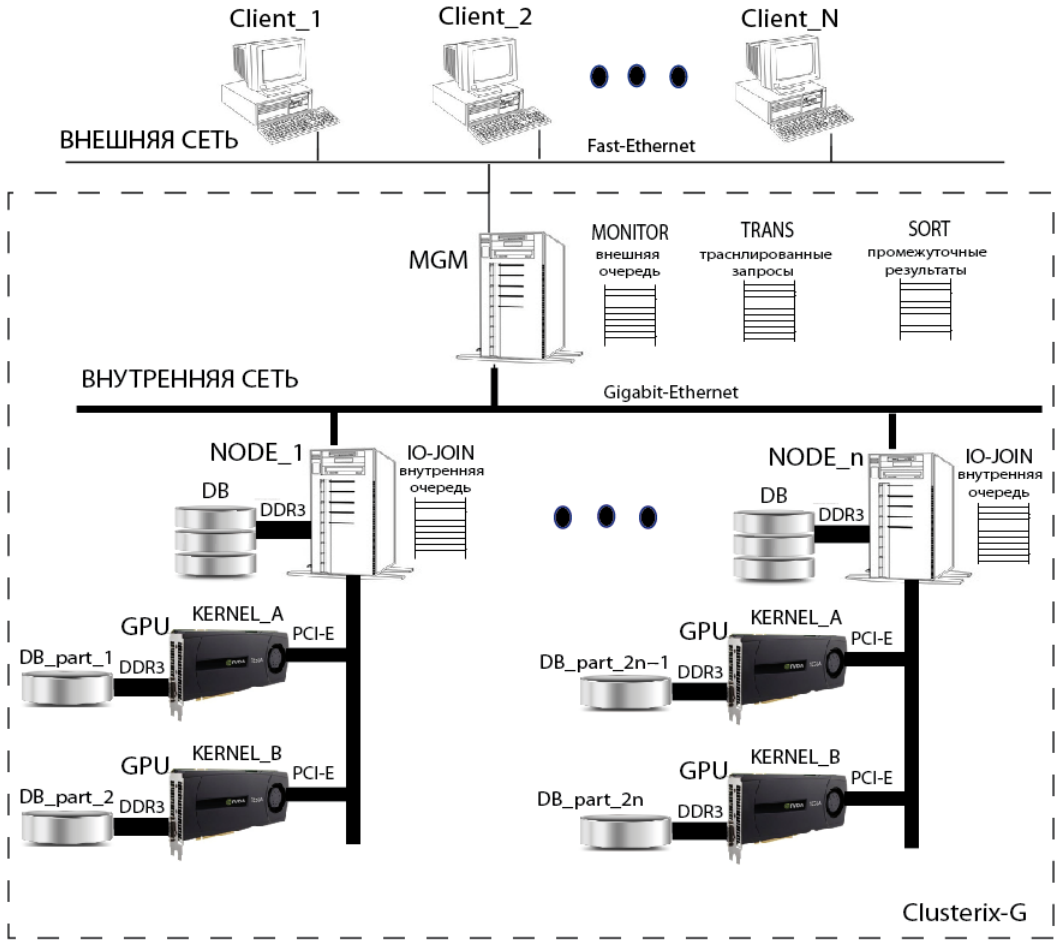

Рис.1. Архитектура параллельной СУБД Clusterix-G

Организация работы GPU

Полученный SQL запрос пользователя транслируется модулем TRANS в пакет простых SQL подзапросов согласно регулярному плану обработки (рис.2) [4]. Правомерность его использования показана, в частности, в работе [5].

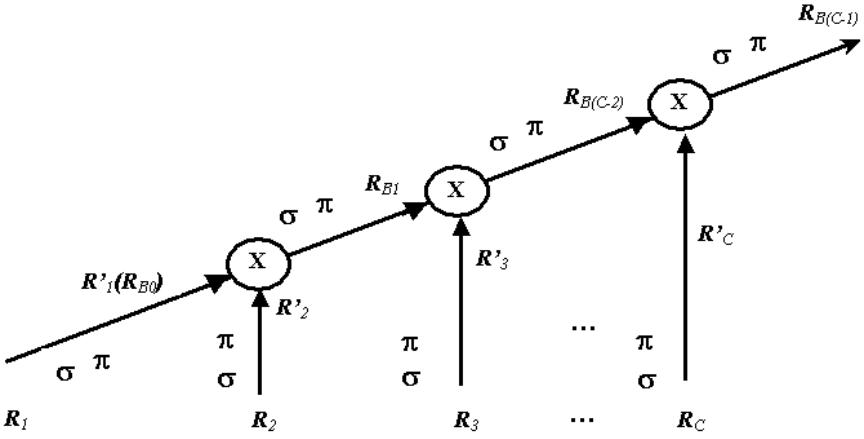

Рис. 2. Регулярный план обработки запроса

Промежуточное отношение $R_i`$ является результатом проведения операций селекции ($\sigma$) и проекции ($\pi$) над исходным отношением $R_i$; C – количество отношений БД, задействованных в текущем запросе. Временное отношение $R_{Bj}$ получается при соединении (x) временного $R_{B(j-1)}$ и промежуточного $R_{(j+1)}`$ отношений с последующим применением к ним операций проекции; R_{Bj} формируется в результате исполнения подзапроса (ядра) и сохраняется в памяти GPU, после исполнения следующего подзапроса оно может быть удалено. Результат последовательного выполнения всех ядер (функции) на GPU в виде временного отношения $R_{B(C-1)}$ передается из модуля IO-JOIN в SORT.

При выполнении операции соединения ($R_{B(j-1)}$ x $R_{(j+1)}`$) внутри ядра в память GPU последовательно подкачиваются части расхешированного отношения $R_{j+1}$ из БД, хранящейся в оперативной памяти узла, при этом подкачиваются только те поля, которые определены в $R_{(j+1)}`$. Каждый кортеж отношения хранится в памяти GPU в виде структуры, поля структур соответствуют атрибутам отношения, отношение в целом состоит из массива таких структур.

Каждый вычислительный узел кластера имеет два GPU-акселератора (см. рис. 1). В модуле IO-JOIN запускается два потока, каждый из которых контролирует работу своего GPU. Ядра KERNEL_A и KERNEL_B работают асинхронно и независимо друг от друга, каждое обрабатывает свою часть исходной БД. Суммарное число потоков, на котором запускается i-е ядро KERNEL ($i \in \{\overline{1,C}\}$), определяется числом кортежей в отношении Ri, каждый поток обрабатывает свой кортеж.

Обсуждение

Использование GPU-акселераторов обладает большим потенциалом повышения производительности (как уменьшения времени обработки сложных запросов) параллельных СУБД кластерного типа. Реализация этого потенциала требует развития теории таких систем для выработки практических рекомендаций к их построению. Это возможно путем детального анализа процессов, в них протекающих. Только знание особенностей динамики параллельных СУБД с утяжеленными GPU-акселераторами узлами позволит дать объективные рекомендации по их построению. Проведение подобных исследований возможно с использованием специализированной системы моделирования рассматриваемых СУБД. Детальный анализ динамики на ее основе поможет развитию элементов теории и выработки практических рекомендаций. Архитектура системы, предложенная в данной работе, может быть принята за основу построения такой специализированной системы моделирования.

Литература

1. Xu Y., Kostamaa P., Zhou X., Chen L. Handling data skew in parallel joins in shared-nothing systems // ACM SIGMOD international Conference on Management 9of Data Canada, 2008, proceedings. ACM. 2008. P. 1043–1052.
2. Лепихов А.В. Параллельная обработка запросов в СУБД для кластерных вычислительных систем // Отчет в рамках гранта МК-3535.2009.9
3. Saecker M., Leischner N. GPU processing in database systems. Technical report // AMD Fusion Developer Summit, 2011.
4. Райхлин В.А. Моделирование машин баз данных распределенной архитектуры // Программирование. 1996. №2. С.7–16.
5. Райхлин В.А., Минязев Р.Ш. Мультикластеризация распределенных СУБД консервативного типа // Нелинейный мир. 2011. №8. С. 473–481.

ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ДЛЯ ПОИСКА ГРАНИЧНЫХ КОНТУРОВ КАРЬЕРОВ РУДНЫХ МЕСТОРОЖДЕНИЙ

В.М. Михелев, Д.В. Петров

Белгородский государственный национальный исследовательский университет

Рассматриваются основные принципы применения высокопроизводительных вычислительных грид-систем для расчета предельных границ рудных месторождений с использованием параллельного генетического алгоритма. Описывается алгоритм решения задачи и приводятся результаты вычислительных экспериментов.

Введение

Grid-система считается одной из разновидностей кластерных систем, но имеет свои особенности, обусловленные масштабом. Грид-системы в первую очередь характеризуются географической распределенностью вычисляемых ресурсов системы. Предоставляемая ими возможность объединять в рамках одной системы огромные вычислительные мощности делает интересным их использование для высокопроизводительных вычислений.

Вычислительные узлы Grid всегда расположены далеко друг от друга, слабо связаны между собой через интернет-каналы, и доступность того или иного из них в произвольный момент времени не гарантирована. Это накладывает дополнительные требования на управление ресурсами. Существует класс промежуточного программного обеспечения, который обеспечивает прозрачную работу приложений в неоднородной сетевой среде. Это так называемое межплатформенное связующее программное обеспечение (middleware).

Среди свободно распространяемых проектов связующего ПО можно выделить трех лидеров – Globus Toolkit, UNICORE и gLite. Всё это сервис-ориентированные системы, во многом реализующие общие стандарты и технологии работы с кластерами.

Globus Toolkit представляет собой набор модулей для построения виртуальной организации распределенных вычислений. Каждый модуль определяет интерфейс, используемый высокоуровневыми компонентами, и имеет реализацию для различных сред выполнения. Вкупе они образуют виртуальную машину Globus. Существуют следующие группы модулей:

- поиска и выделения ресурсов;
- коммуникаций;
- аутентификации;
- информационные;
- доступа к данным;
- создания процессов.

На основе этих низкоуровневых компонентов строятся сервисы более высокого уровня.

Цель данной статьи – продемонстрировать применение grid-системы для поиска предельных границ рудных месторождений.

Задача нахождения предельных границ карьеров рудных месторождений является одним из важнейших этапов планирования разработки полезных ископаемых открытым способом. Ее решение, во-первых, позволяет оценить объем получаемой прибыли, а во-вторых, является фундаментом для следующих этапов проектирования, таких как нахождение оптимальной сети карьерных транспортных путей и выбор мест расположения отвалов и перерабатывающих фабрик.

При нахождении границ карьера необходимо учитывать пространственное распределение компонентов полезных ископаемых и принятых устойчивых или технологически допустимых углов откосов бортов карьера. С вычислительной точки зрения данная задача является крайне сложной, т.к. для моделирования месторождений даже среднего размера приходится обрабатывать большие массивы данных.

Для решения задачи оптимизации с применением ЭВМ используют упрощенную блочную математическую модель месторождения полезных ископаемых. Каждый блок данной модели характеризуется числом (весом), показывающим чистую прибыль, получаемую в ходе его добычи, с учетом процентного содержания полезных элементов, себестоимости его выработки и рыночной стоимости полезных элементов. В этом случае задача оптимизации сводится к нахождению конечного набора соседних блоков, сумма весов которых будет максимальна. Предлагается применить для этого генетический алгоритм.

Генетические алгоритмы оптимизации основаны на моделировании процессов мутации, скрещивания и естественного отбора. Поиск в них начинается со случайной популяции индивидуумов, однозначно характеризующихся неким набором данных – хромосомой. Качество каждого индивидуума оценивают посредством расчета функции пригодности, аргументом которой является его хромосома. Путем применения генетических операторов (отбора, мутации, скрещивания) данная популяция улучшается до тех пор, пока либо функция пригодности одного из индивидуумов не достигнет приемлемого значения, либо не пройдет допустимое количество циклов воспроизводства.

Формат представления хромосом

Для реализации генетического алгоритма в первую очередь необходимо разработать формат представления хромосом. В контексте задачи нахождения границ карьера можно предложить следующее решение: форма любого допустимого (с учетом углов наклона) карьера представляется с помощью массива целых чисел. Каждый элемент такого массива показывает глубину карьера в текущем столбце трехмерной модели месторождения.

Пусть имеется трехмерная блочная модель месторождения P_{IxJxK}, каждый элемент которой характеризуется числом (весом)

$$p_{ijk},\ i \in [0, I],\ j \in [0, J],\ k \in [0, K], \qquad (1)$$

показывающим чистую прибыль, получаемую в ходе его добычи, с учетом процентного содержания полезных элементов, себестоимости его выработки и рыночной стоимости полезных компонентов. Тогда ее можно охарактеризовать вектором $X = =\{x_1,\ldots,x_n\}$, где $n = I * J$, в котором значение глубины столбца с координатами (i, j) помещается в позицию

$$x_q,\ q = i * I + j. \qquad (2)$$

Этот массив является хромосомой, т.к. он полностью характеризует один индивид – одну конкретную форму карьера. Путем итеративного применения генетических операторов к набору таких индивидов (популяции) находится оптимальная форма поверхности карьера.

Для расширения области поиска, уменьшения вероятности преждевременного схождения и уменьшения времени вычислений был разработан параллельный вариант

данного алгоритма – иерархический параллельный генетический алгоритм с двумя уровнями параллелизма.

Первый уровень параллелизма организуется за счет применения островной модели многопопуляционного параллельного генетического алгоритма. Здесь ускорение достигается за счет выделения нескольких начальных популяций, развивающихся независимо, и периодически обменивающихся наиболее хорошим генетическим материалом. Данный обмен осуществляется посредством механизма миграции особей между популяциями. Такой подход обеспечивает снижение вероятности преждевременного вырождения популяций, увеличению их разнообразия и ускорению схождения алгоритма поиска.

Второй уровень иерархии организуется за счет применения для каждой подпопуляции однопопуляционной модели параллельного генетического алгоритма типа «хозяин–подчиненный». Она заключается в том, что в рамках одной популяции функция приспособленности каждого индивидуума вычисляется в отельном потоке, что в итоге приводит к ускорению работы алгоритма. При этом один поток является главным, «хранителем» популяции и отвечает за работу генетических операторов, а ряд потоков-подчиненных только вычисляют функцию приспособленности.

Кроме того, внимательно проанализировав структуру последовательной версии алгоритма, можно выделить еще несколько вычислительно независимых участков, которые можно выполнять параллельно в рамках развития популяции. Так, после разбиения хромосом по парам каждую пару можно скрещивать параллельно. Также и процедуру мутации можно выполнять для каждой хромосомы независимо.

Вычислительные эксперименты

В качестве технической платформы для проведения вычислительных экспериментов использовался сегмент грид-системы под управлением платформы Globus Toolkit. В состав данного сегмента входит координирующий сервер и вычислительные узлы на базе процессоров Intel Xeon. Структурная схема взаимодействия всех этих компонентов представлена на рис. 1.

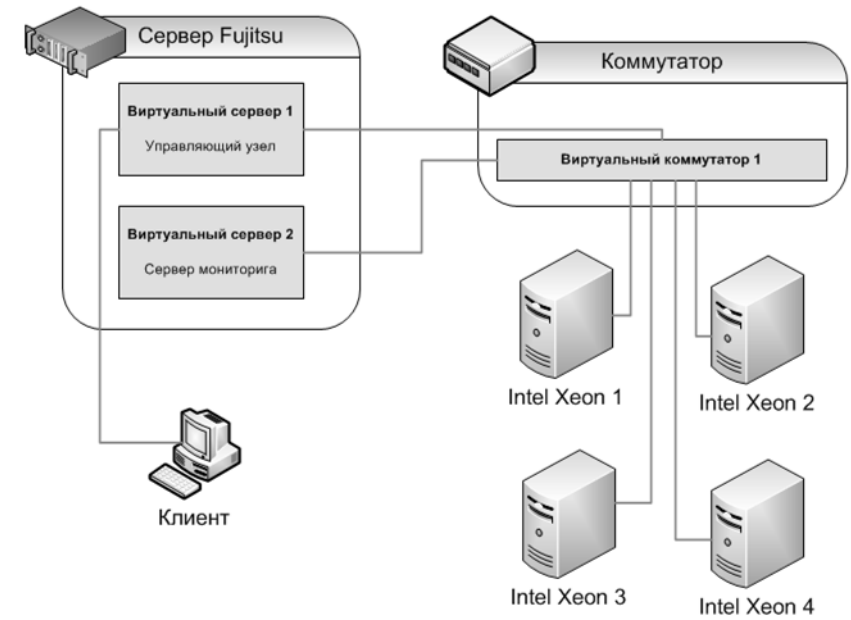

Рис. 1. Схема оборудования

Суммарные технические характеристики кластерной системы приведены в табл. 1.

Таблица 1. Суммарные технические характеристики кластерной системы

Характеристика	**Значение**
Семейство процессора	QuadCore Intel Xeon
Частота процессора	1.6 ГГц
Количество процессоров	12
Количество ядер	32
Объем ОЗУ	20 Гб
Объем HDD	1.2 Тб
Сеть	Gigabit Ethernet (1000 Mbps)

Алгоритм тестировался на нескольких моделях пространственного распределения полезных компонентов в земной поверхности: наклонное послойное залегание, вертикальное залегание, равномерное случайное распределение. На рис. 2 приведен пример визуального представления граничной формы карьера размером 100 × 100 × 100 метров с разрешением 1 метр, рассчитанного генетическим алгоритмом.

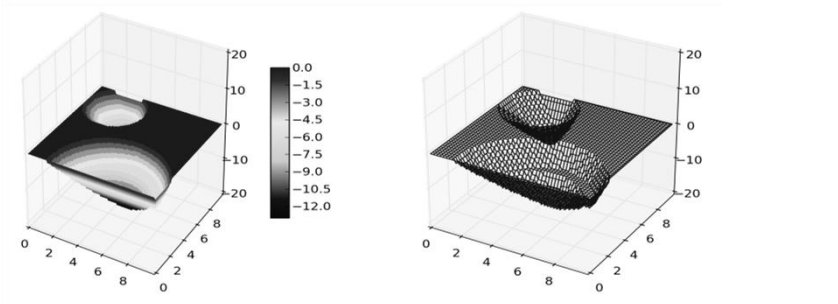

Рис. 2. Трехмерное изображение формы карьера, масштаб 1:10000

В рамках данного эксперимента проводилась проверка работы алгоритма на нескольких узлах грид-системы, при этом было использовано два уровня параллелизма. Цель данного эксперимента – выяснить, как меняется время расчетов в зависимости от количества вычислительных узлов.

В качестве тестовых данных использовалась модель карьера со случайным пространственным распределением полезных компонентов размером 100 × 100 × 100 блоков. После проведения вычислительных экспериментов, был получен результат, представленный на рис. 3.

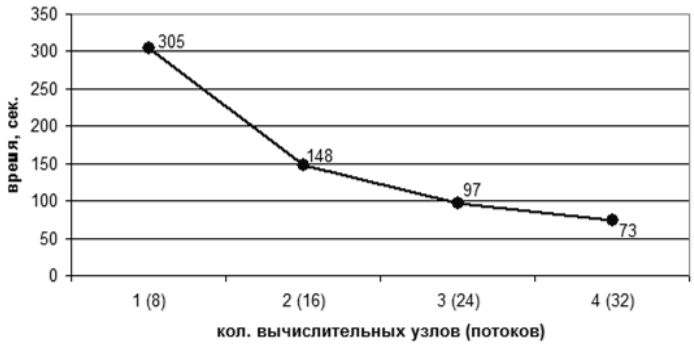

Рис. 3. Зависимость времени выполнения от количества задействованных узлов грид-системы

По результатам эксперимента можно сделать вывод, что алгоритм хорошо масштабируется на распределенную вычислительную систему. И, хотя за счет задержек на об-

мен данными между вычислительными узлами наблюдается некоторое замедление вычислительного процесса, все же целесообразно применять подобные системы для снижения времени вычислений при обработке больших моделей месторождений полезных ископаемых.

Выводы

Результаты вычислительных экспериментов показали высокую перспективность предложенного метода для выполнения расчетов на регулярных блочных моделях месторождений твердых полезных ископаемых, разрабатываемых открытым способом. Основные преимущества предложенного метода заключаются в предоставлении нового принципа решения задачи оптимизации карьеров, позволяющего работать напрямую с трехмерной моделью месторождения, что значительно повышает адекватность получаемой модели. Кроме того, возможности гибкого масштабирования вычислительного процесса позволяют сокращать время обсчета модели почти линейно с увеличением количества вычислительных узлов.

Литература

1. Васильев П.В. Ускорение моделирования и оптимизации извлечения запасов рудных месторождений на основе параллельных вычислений // Горный информационно-аналитический бюллетень. – М.: МГГУ , 2012. – №3. – С. 205-211.
2. Гергель В.П. Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н. Новгород: Изд-во Нижегородского государственного университета, 2003. – 325 с.
3. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем: учебник для студентов вузов. – М.: МГУ, 2010. – 544 с.
4. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. Горячая Линия Телеком, 2007.
5. Шпаковский Г.И. Реализация параллельных вычислений: MPI, OpenMP, кластеры, грид, многоядерные процессоры, графические процессоры, квантовые компьютеры. – Минск: БГУ, 2011.