УДК 681.3

ИНТЕЛЛЕКТУАЛЬНЫЙ АГЕНТ РАСПАРАЛЛЕЛИВАНИЯ ЗАПРОСОВ

В.В. Гудзенко

Институт кибернетики им. В.М. Глушкова НАН Украины,
03680, Киев, проспект Глушкова, 40,
тел.: (044) 526 3603, vovique@rambler.ru

Изложен принцип построения интеллектуального агента распараллеливания запросов для кластеров с общей дисковой памятью. Описаны основные модули агента и приведены алгоритмы функционирования таких модулей.

Concept of intellectual query-distribution agent for shared-disk clusters is given. Main modules of the agent are described with given functioning algorithms.

Вступление

Кластерные системы, занимающие ведущее место на рынке высокопроизводительных комплексов, используют для решения трех основных классов задач: вычислительных, обработки баз данных и смешанных.

Вычислительные задачи не используют интенсивные обмены данными с СУБД, а выполняют сугубо вычислительные операции, например, для рендеринга. В широком спектре приложений данные обрабатываются преимущественно в связке с СУБД, например, в таких комплексах, как электронные магазины, учетные системы, хранилища данных. Смешанные классы задач совмещают обработку баз данных и выполнение мощных вычислений, например, в системах поддержки принятия решений с интенсивной обработкой больших массивов сложно организованных данных.

Главный фактор производительности кластерных систем – естественный параллелизм вычислений. Именно через одновременность выполнения отдельных частей одной задачи с последующим объединением результатов кластеры вышли на передовые рубежи. Соответственно, чем лучше задача распараллеливается, тем большей производительности кластера можно достичь в ее решении.

Кластеры семейства СКИТ

Сегодня Национальная академия наук Украины использует кластеры семейства СКИТ Института кибернетики им. В.М. Глушкова НАН Украины (табл. 1). Вычислительный кластер СКИТ - это набор компьютеров-узлов, объединенных коммуникационной сетью. Каждый вычислительный узел имеет свою оперативную память и работает под управлением своей (одинаковой для всех) операционной системы, поэтому все узлы кластера однородны, т.е. абсолютно идентичны по архитектуре и производительности. Для каждого кластера выделенный хост-компьютер управляет запуском программ на процессорах. Вычислительные процессы пользователей запускаются на узлах кластера так, что на каждый процессор приходится не более одного вычислительного процесса. Пользователи имеют терминальный доступ на хост-компьютер кластера, но не на узлы кластера [1].

Характеристики кластеров семейства СКИТ — Таблица 1

Характеристики кластера	СКИТ-1	СКИТ-2
Количество узлов	16	32
Количество процессоров	32	64
Элементная база процессоров	Intel Xeon 2,67 ГГц	Intel Itanium 2 1,4 ГГц
Разрядность процессоров	32 бита	64 бита
Оперативная память узла	1 Гб	2 Гб
Хранилище данных	440 Гб	1 Тб

Кластеры семейства СКИТ использовались вначале для вычислительных задач, но затем появились смешанные задачи. В 2005 г. на кластерах инсталлирована СУБД MySQL в многосерверной реализации (Multi-Daemon MySQL), в которой ввиду отсутствия дисковой памяти в узлах несколько серверов единой БД инсталлированы и запускаются на хост-компьютере, предоставляя через разные порты доступ к БД, размер которой ограничен емкостью хранилища (см. табл.1).

Как клиент СУБД MySQL разработано приложение корпусной разметки украиноязычных текстов в рамках Национального корпуса украинского языка. На начальных этапах построения корпуса большое количество текстов преобразуется в форму, пригодную для анализа текста специалистами-лексикологами. Суть преобразования состоит в выделении в тексте словоформ и проведении над ними морфологического разбора по правилам, сформулированным в словарной БД [2]. Словарная БД приложения содержит более 816 тысяч словоформ. Приложение формирует соответствующий SQL-запрос к словарной БД на поиск каждого слова

исходного текста, получая искомый морфологический разбор слова. Входной текст разбивается на равные части (количество которых соответствует количеству используемых узлов кластера). В табл. 2 приведены достигнутые скоростные характеристики обработки текстов на кластере СКИТ-2 [2].

Скорость обработки слов в зависимости от количества используемых узлов кластера Таблица 2

Количество процессоров	Количество обработанных слов за секунду					
	1-й эксперимент длина текста 74525 слов		2-й эксперимент длина текста 22084 слов		3-й эксперимент длина текста 4616 слов	
	слов/сек	прирост	слов/сек	прирост	слов/сек	прирост
1	2404	-	818	-	385	-
2	2661	10,69%	1162	42,11%	577	50,00%
3	3387	40,89%	1840	125,00%	385	0,00%
4	3387	40,89%	1840	125,00%	308	-20,00%

Эксперименты показали, что использование многосерверной реализации СУБД с параллельным доступом к ней с разных узлов для задачи корпусной разметки текстов максимально ускоряет обработку данных при использовании трех серверов чтения, причем дальнейшее увеличение их количества не приводит к росту производительности, а в ряде случаев даже понижает ее. Это объясняется расположением серверов БД на хост-компьютере, ресурсы которого полностью распределяются между всеми запущенными на нем службами и приложениями. Возможно, для других приложений потребуется другой процессорный ресурс обработки БД.

Для разработки любого параллельного приложения для решения смешанной задачи разработчику приходится решать многие вопросы параллельной обработки БД (балансировка нагрузки узлов, межузловые обмены и т.п.), используя для этого API конкретной БД. Это и наличие большого количества уже разработанных программных продуктов требуют иного подхода к организации взаимодействия параллельных приложений с несколькими серверами БД.

Агент распараллеливания запросов

Для ускорения работы взаимодействующих с БД приложений и для облегчения написания таких параллельных программ, целесообразно использовать агент-посредник, назначение которого – анализ SQL-запросов клиента, их автоматическое распараллеливание с передачей соответствующему серверу БД и поддержка обратной связи. В системах без такого агента пул параллельных соединений организуется и управляется непосредственно клиентским приложением, которое работает с БД. Такой пул создается при соединении приложения с БД и закрывается по завершении его работы (рис. 1).

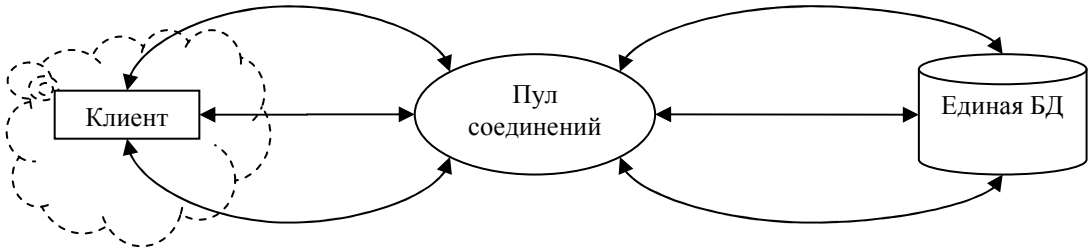

Рис. 1. Общая схема обмена данными в системе без агента распараллеливания запросов

Задача агента-посредника взять на себя организацию распараллеливания SQL-запросов и балансировку нагрузки на сервера БД (см. рис. 2), динамически изменяя конфигурацию серверов БД.

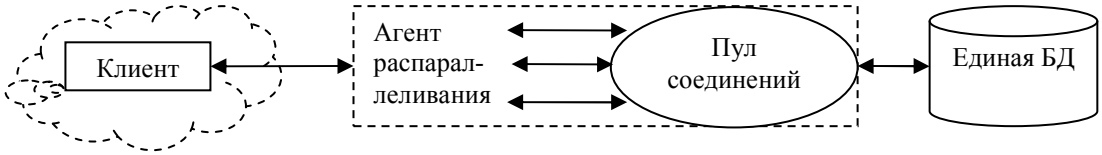

Рис. 2. Общая схема обмена данными в системе с агентом распараллеливания запросов

Агент принимает SQL-запросы на указанный порт, анализирует полученные данные и пересылает их на соответствующий сервер БД. Анализ происходит сначала в SQL-парсере для определения типа запроса (на чтение или записывание). Запрос на записывание передается на выделенный сервер записывания в БД, а запрос на чтение – балансировщику загрузки, который контролирует нагрузку на серверы чтения БД и передает этот запрос наименее загруженному серверу БД. Также агент возвращает полученные результаты от серверов БД к соответствующему клиенту (см. рис. 3).

Агент содержит четыре основных модуля: SQL-парсер, балансировщик нагрузки, менеджер запросов и монитор очередей. Рассмотрим детальнее каждый из ключевых модулей агента.

SQL-парсер выполняет первичный анализ полученных запросов, т.е. главную задачу – синтаксический анализ полученных SQL-запросов с целью установления типа операций с данными в БД. Если запрос содержит команды записывания данных (например, операторы create, insert, update, alter), то запрос передается на сервер записывания в БД, единственный в кластере. Это поясняется относительно меньшим количеством запросов записывания, чем чтения, и проблемой блокировки записываемых данных. Если запрос оказывается запросом чтения (например, оператор select), то запрос перенаправляется на балансировщик нагрузки.

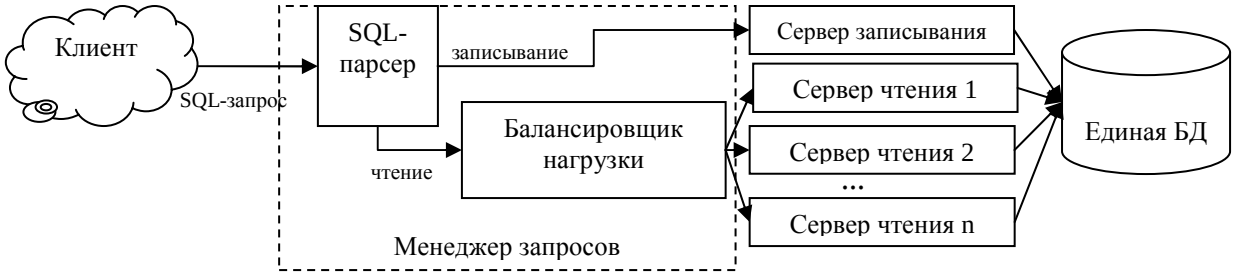

Рис. 3. Схема обработки SQL-запросов агентом

Балансировщик нагрузки контролирует нагрузку серверов чтения из БД. При получении нового запроса он отправляется на свободный сервер чтения (при наличии такого) или на сервер с наименьшим количеством запросов в очереди (рис. 4).

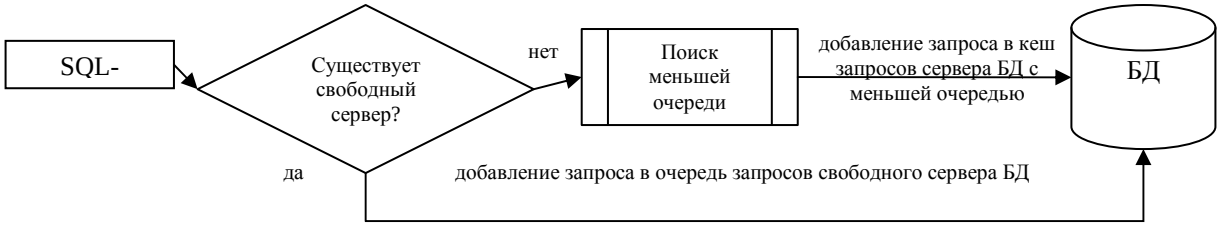

Рис. 4. Общая схема работы балансировщика нагрузки

Количество запросов в очереди к каждому серверу регистрируется и изменяется при добавлении нового запроса к БД (увеличение очереди на сервере) или выполнении запросов и возвращении данных (уменьшение очереди на сервере). Если очередь запросов одного из серверов БД пуста (все запросы в этой очереди выполнены), а очередь второго сервера БД содержит более одного запроса, то для балансировки нагрузки некоторые запросы из очереди второго сервера передаются в очередь первого (рис. 5). Отметим, что такую межсерверную передачу запросов невозможно организовать посредством самого агента, поскольку очереди расположены в памяти серверов БД и внешнего доступа к ним нет. Однако наличие исходного кода серверов MySQL дает возможность запрограммировать эту часть непосредственно на серверах MySQL [3].

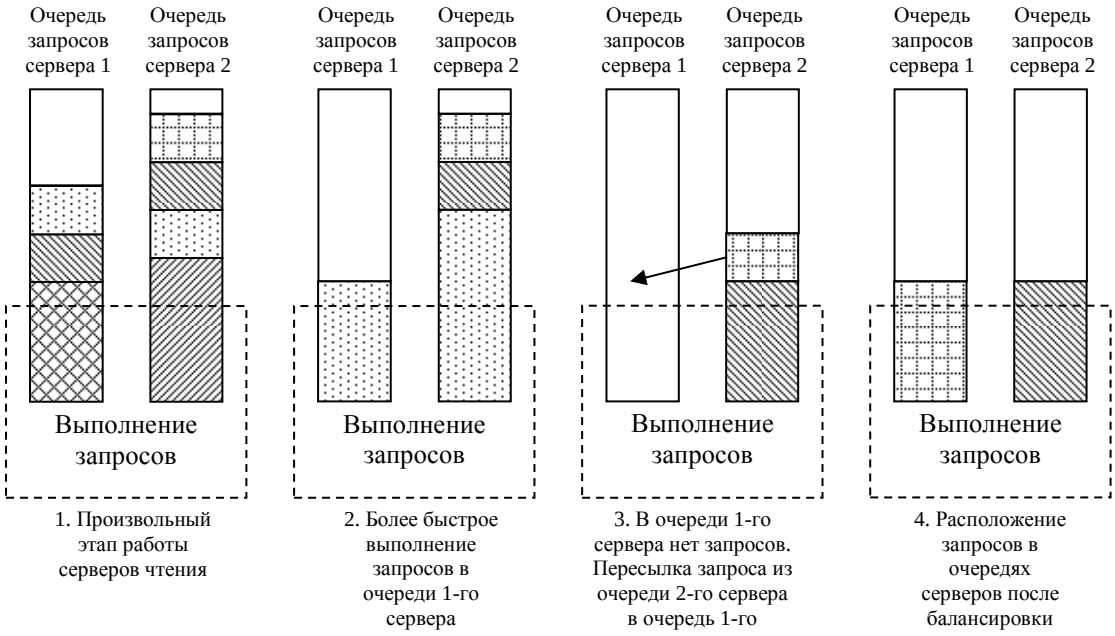

Рис. 5. Работа балансировщика нагрузки при передаче запросов между очередями двух серверов

Возможен иной вариант балансировки нагрузки – через единую очередь агента, в которой запросы сохраняются и передаются по одному на выполнение серверам БД. Передача запросов происходит на свободный сервер после получения результатов предыдущего запроса (рис. 6).

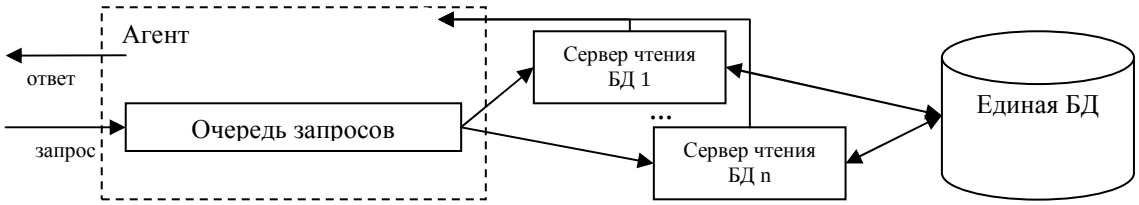

Рис. 6. Схема работы балансировщика нагрузки при использовании единой очереди запросов

Менеджер запросов как контролирующий модуль устанавливает соответствие запросов и клиентов, которые их присылают, и отвечает за дисциплину очереди. По умолчанию используется наиболее распространенное правило формирования очередей: «первый пришел – первый обслуживается». Также существует вариант обслуживания запросов по приоритету.

Монитор очередей постоянно отслеживает и протоколирует поток запросов, которые проходят через агент, для динамического оценивания:

- средней длины очередей запросов;
- среднего срока обслуживания запросов;
- среднего времени простоя серверов БД;
- распределения запросов по серверам;
- гомогенности запросов;
- зависимостей характеристик запросов от клиентов.

Постоянно анализируя получаемые данные, монитор очередей перестраивает систему для оптимальной обработки поступающих запросов путем:

- автоматического конфигурирования серверов БД;
- выбора способа распределения запросов между серверами БД при балансировке нагрузки;
- установления дисциплины очередей.

После автоматического конфигурирования серверов БД сравниваются текущие значения отслеживаемых параметров с их характеристиками до реконфигурации. При снижении производительности системы конфигурация возвращается в предыдущее состояние. Повторная попытка аналогичного конфигурирования будет выполнена лишь при иных значениях отслеживаемых параметров (рис. 7).

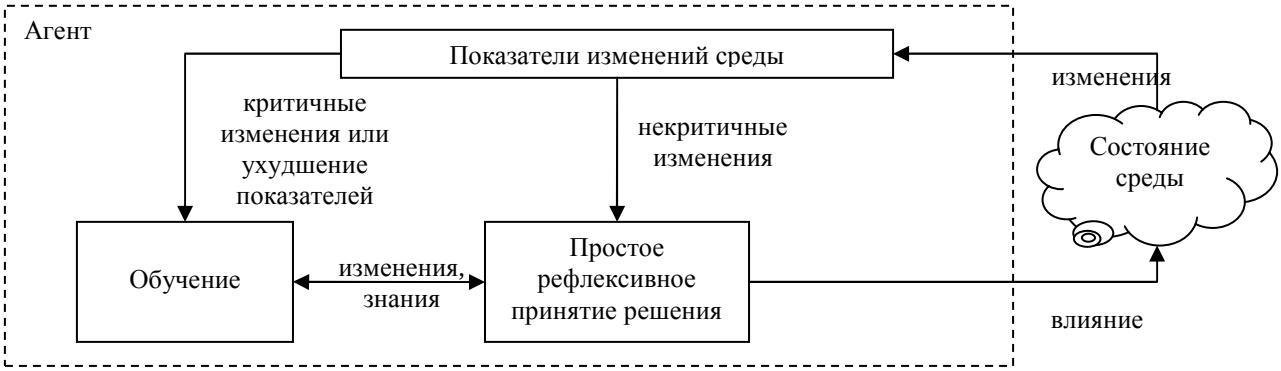

Рис. 7. Схема работы монитора очередей при динамической оценки характеристик среды

Выполненные изменения протоколируются и отправляются уведомлением администратору БД. Уведомление может выполняться как на локальном компьютере, так и через SMTP-протокол при удаленном администрировании БД.

Вывод

Агент распараллеливания запросов составляет часть системного программного обеспечения кластера и ориентирован на ускорение обработки запросов к БД путем их распараллеливания между разными серверами БД. Используя данные о парсинге запросов и балансировке нагрузки серверов чтения из БД, такой агент

обладает возможностью самообучения для динамического конфигурирования серверов БД. Ожидается, что внедрение агента ускорит выполнения запросов по меньшей мере на 25 % и избавит разработчиков приложений от необходимости вручную программировать параллельное взаимодействие приложений с БД.

1. *Официальная* страница кластеров Института кибернетики им. В.М. Глушкова НАН Украины -http://cluster.icyb.kiev.ua/about.html
2. *Гречко В.О., Гудзенко В.В.* Продуктивность параллельной СУБД в кластерных системах // Компьютерная математика. – 2005. – № 3. – С. 71–79.
3. *Официальный* сайт компании MySQL AB - http://www.mysql.com/company/legal/licensing/