ПОДХОДЫ К ОРГАНИЗАЦИИ СИСТЕМЫ УПРАВЛЕНИЯ ЗАЩИЩЕННЫМИ КАРТОГРАФИЧЕСКИМИ БАЗАМИ ДАННЫХ

Р.Ф. Гибадуллин, С.В. Пыстогов

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

Рассматривается серверная часть системы управления защищенными картографическими базами данных на основе СУБД MySQL. Приводятся два архитектурных подхода к построению такой системы с применением вычислительных кластеров: монокластер и мультикластер. Выделяются и анализируются два способа построения монокластера. Даются предпосылки к разработке системы управления защищенными картографическими базами данных в среде PostgreSQL.

Введение

В настоящее время системы управления картографическими базами данных развиваются на стыке многих научных дисциплин и применяются в различных сферах деятельности. Например, при изучении природных ресурсов и управлении ими, при создании и ведении кадастров и управлении муниципальными образованиями, и в других сферах.

Коммерческая и другая ценность картографической продукции требует организации ее информационной безопасности. Обработка запросов к защищенным базам данных картографических сцен связывается со значительными временными затратами. Это приводит к необходимости организации параллельной обработки.

1. Серверная часть системы управления защищенными картографическими базами данных на основе СУБД MySQL

Принципы формирования защищенных картографических баз данных для разных типов объектов подробно рассматриваются в работах [1] и [2].

На рис. 1 представлена серверная часть системы.

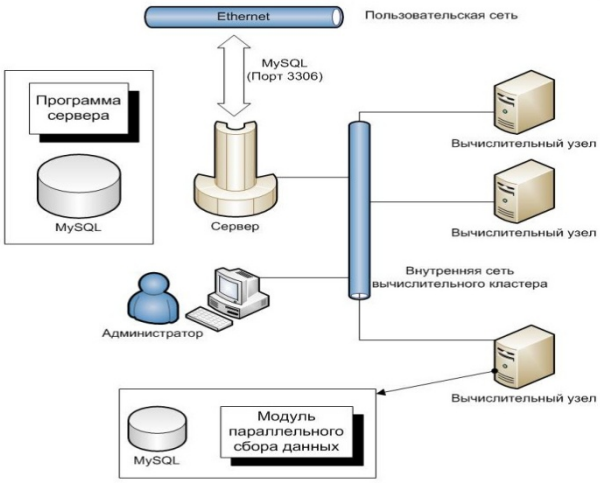

Рис. 1. Серверная часть системы

Как видно из рисунка, серверная часть – это вычислительный комплекс, состоящий из управляющего узла и множества вычислительных узлов. Управляющий узел является ключевым в задачах получения запросов от клиентов, распределения задач между счетными узлами кластера, сбора результатов и т.д. Остальные узлы кластера являются вычислительными, кроме того совместно выполняют роль распределенного хранилища.

К серверу по корпоративной сети подключается множество пользователей.

В системе за основу берется «интеллектуальная» файл-серверная организация взаимодействия. Сервер выполняет не только функции хранения, но и функции предварительной обработки поступающих запросов и «интеллектуальный» сбор результатов.

Файловый сервер принимает запросы, поступающие по сети от компьютеров-клиентов, и, в соответствии с запросами, передает им требуемые данные. Клиентская часть получает от сервера ответ (результат).

При построении такой системы с применением вычислительных кластеров можно выделить два архитектурных подхода.

– ***Монокластер*** («весь кластер на запрос»). В реализации данного подхода все вычислительные узлы серверной части в один момент времени обрабатывают только один запрос. Основные особенности:

- Все базы данных равномерно распределяются по узлам кластера.
- Каждый вычислительный узел выполняет поиск удовлетворяющих условиям запроса элементов в своей локальной части БД.
- Конечный результат получается путем конкатенации результатов со всех узлов, где были получены промежуточные результаты.
- Серверу нет необходимости выполнять функцию роутера. Запросы обрабатываются в порядке очередности их поступления на сервер.
- Возможна задержка барьерной синхронизации.

– ***Мультикластер*** («один узел на один запрос»): параллельное выполнение множества клиентских запросов на кластере. Система способна обрабатывать одновременно до N запросов, где N – число вычислительных узлов кластера. Особенности:

- Репликация баз данных по узлам. Каждый вычислительный узел хранит полную копию всех защищенных баз данных картографических сцен (ЗБД КС).
- На управляющий узел возлагаются дополнительные функции распределения запросов между узлами – функции роутера.
- Необходимость контролирования когерентности данных в ЗБД между узлами при изменении информации.

Можно выделить два способа построения монокластера:

1. Запрос отправляется для выполнения на каждый вычислительный узел кластера, где происходит полный просмотр всех фрагментов и поиск необходимых объектов. При этом, в случае селективного запроса, часть узлов кластера загружается «вхолостую».
2. На сервере хранится информация о расположении конкретных фрагментов по узлам вычислительного кластера. Селективный запрос направляется только на определенные узлы кластера, где просматриваются лишь предписанные фрагменты.

Для реализации способа 2 в разработанную структуру ЗБД КС необходимо ввести дополнительное отношение, которое хранит в себе пары связей «Номер фрагмента картографической сцены – Номер (адрес) узла кластера». Запросы для обработки берутся из очереди запросов, как и в способе 1. В ходе предварительной обработки, кроме выполнения раскрытия сокрытых параметров, синтаксического анализа и проверки прав пользователя на выполнение запроса, выполняется поиск узлов кластера с искомой информацией. Для этого используется табличное отношение с информацией о расположении всех фрагментов картографической сцены на узлах кластера.

Разработанные программные модули для обоих способов тестировались с использованием сгенерированных ЗБД КС. В качестве тестовых запросов выбраны:

1. select * from db; – выборка всех объектов сцены;
2. select * from db where x>1562340; – выборка участка сцены;
3. select * from db where x<710276; – выборка участка сцены.

Любой из них применим ко всем БД: DB1, DB2, DB3. Тестирование проводилось на 10 узлах суперкомпьютера КНИТУ-КАИ (с характеристикой узла: 2 Quad-Core Intel(R) Xeon(R) E(5450) (2*6 MB L2 cache, 3.00 GHz, 1333 MHz FSB), 32 GB Memory DDR2-667 MHz, 2*Ethernet 10/100/1000 BASE-T, 2*InfiniBand с пропускной способностью до 20 Gbit/s).

Полученные времена обработки выбранных запросов по первому и второму способам приведены в табл. 1. Для первого способа даны замеры времени при работе системы на кластере и на одном вычислительном узле (в скобках). Видно, что при выборке участков сцены второй способ более скоростной. Но при выборке данных всей сцены преимуществ не наблюдается.

Таблица 1. Замеры времени для способов 1 и 2 архитектуры монокластера

	DB1		DB2		DB3	
	Способ 1	Способ 2	Способ 1	Способ 2	Способ 1	Способ 2
Запрос 1	37 сек. (45 сек.)	39 сек.	41 сек. (57 сек.)	42 сек.	59 сек. (71 сек.)	60 сек.
Запрос 2	36 сек. (48 сек.)	17 сек.	43 сек. (54 сек.)	20 сек.	55 сек. (75 сек.)	26 сек.
Запрос 3	40 сек. (45 сек.)	18 сек.	50 сек. (59 сек.)	22 сек.	61 сек. (66 сек.)	28 сек.

Это объясняется тем, что в последнем случае поиск ведется по всем узлам. Использование вычислительного кластера по способу 1 дает лишь небольшой выигрыш по сравнению с обработкой тех же запросов на одном вычислительном узле, несмотря на десятикратное преимущество в вычислительных ресурсах.

Для дальнейшего ускорения процесса обработки потока запросов по способу 2 было предложено его развитие – способ 3 (рис. 2):

- На управляющем узле формируется начальная очередь клиентских запросов: n>N, где n – размер очереди, N – число узлов кластера (рис. 2, левый столбец).
- Производится предварительная обработка запросов согласно случаю 2 (рис. 2, правый столбец). Формируется очередь предобработанных запросов с информацией о целевых узлах кластера.

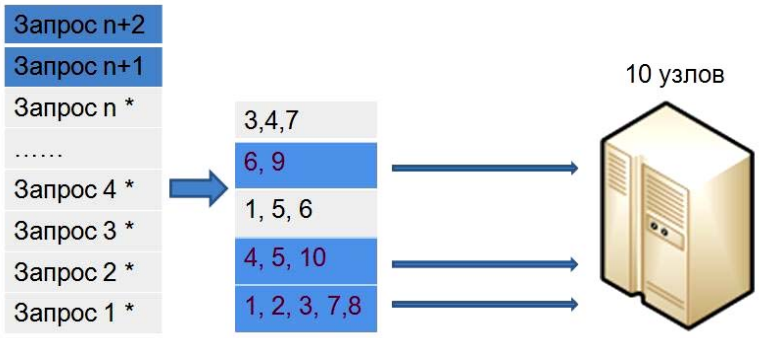

Рис. 2. Развитие способа 2

- Определяются запросы для начала обработки на кластере при возможной загрузке всех вычислительных узлов с учетом порядка следования запросов.

- Как только какой-либо из запросов будет обработан и освободит вычислительные узлы, в очереди предобработанных запросов будет искаться следующий запрос, удовлетворяющий условию максимальной загрузки кластера.
- После очередного считывания запроса из очереди происходит сдвиг, и новые запросы с номерами > n проходят предварительную обработку.

Были проведены эксперименты и для способа 3 на тех же БД. В качестве очереди запросов использовался пул из 20 запросов селективного типа. Шаблон запросов: select * from db{1-3} [where x {<,>,<=,>=} … [and y {<,>,<=,>=} …]]. Полученные результаты:

- Для точечных объектов: обработка 20 запросов без распределения потока запросов – 365 с, с распределением – 256 с. Выигрыш 30%.
- Для линейных объектов: без распределения – 403 с, с распределением – 280 с. Выигрыш 31%.
- Для площадных объектов: без распределения – 418 с, с распределением – 293 с. Выигрыш 33%.

Выигрыш времени на предварительной обработке нивелируется накладными расходами при поиске подходящих запросов для отправки на расчет.

2. Предпосылки к разработке системы управления защищенными картографическими базами данных в среде PostgreSQL

В целях коммерциализации разработанных модулей были проведены хоздоговорные работы с ООО «Геоинформационный центр «Зенит» (г. Казань). В рамках этих работ была установлена целесообразность построения системы управления защищенными картографическими базами с ориентацией на малые предприятия на базе одного многопроцессорного сервера и тонких клиентов. Правомерность этого отмечается и описанными выше результатами: рост производительности «в разы» от использования вычислительного кластера не наблюдается.

Для реализации такой системы в качестве базы выбрана СУБД PostgreSQL, которая обладает следующими преимуществами сравнительно с СУБД MySQL:

- обладает полным набором пространственных возможностей (соответствующих стандартам OGC [3]) и позволяет осуществлять любые виды операций над геоданными;
- включает поддержку пространственных индексов R-Tree/GiST, повышающую скорость обработки SQL-запросов к пространственным базам данных;
- поддержка баз данных практически неограниченного размера, надежность и устойчивость на очень больших нагрузках (что очень критично при организации системы на базе одного сервера);
- поддержка системы аутентификации Kerberos версий 4 и 5.

Таким образом инициирован проект по разработке системы управления защищенными картографическими базами данных в среде PostgreSQL. Ставятся и решаются следующие задачи:

1. Разработка модуля формирования защищенной картографической базы данных, включающая в себя полное множество атрибутов геоданных в соответствии со стандартом OGC.
2. Разработка параллельных модулей управления защищенной картографической базой данных, ориентированных на один вычислительный сервер с многоядерной архитектурой.
3. Разработка интерфейса управления защищенными картографическими базами данных с применением PostGIS и веб-интерфейса клиента для управления защищенной картографической базой данных, по сути реализация идеологии тонкого клиента.

По третьей задаче следует отметить, что использование связки PostgreSQL/PostGIS предоставляет довольно широкие возможности по работе с пространственными данными. Рассмотрим некоторые основы работы с PostGIS [4].

Таблицы метаданных OGC

При создании пространственной базы данных автоматически создаются две таблицы метаданных – SPATIAL_REF_SYS и GEOMETRY_COLUMNS. Они создаются в соответствии со спецификацией Open Geospatial Consortium Simple Features for SQL specification, выпущенной OGC и описывающей стандартные типы объектов ГИС, функции для манипуляции ими и набор таблиц метаданных.

Таблица GEOMETRY_COLUMNS хранит информацию о таблицах базы данных, содержащих пространственную информацию. Ее заполнение осуществляется вручную либо как следствие выполнения специальной процедуры OGC AddGeometryColumn().

Таблица SPATIAL_REF_SYS содержит числовые идентификаторы и текстовые описания систем координат, используемых в пространственной базе данных. Одним из полей этой таблицы является поле SRID – уникальный идентификатор, однозначно определяющий систему координат. SRID представляет из себя числовой код, которому соответствует некоторая система координат. Например, распространенный код EPSG 4326 соответствует географической системе координат WGS84.

Пространственные запросы

Для создания таблицы, содержащей пространственные данные, применим нижеприведенные SQL запросы:

create table points (pt geometry, name varchar);
insert into points values ('POINT(0 0)', 'Origin');
insert into points values ('POINT(4 0)', 'X Axis');
insert into points values ('POINT(0 3)', 'Y Axis');
select name, ST_AsText(pt), ST_Distance(pt, 'POINT(4 3)') from points;

В данном примере мы создали таблицу *points*, содержащую два поля: поле *pt* типа *geometry* и поле *name* типа *varchar*, после чего добавили (*insert*) в нее три записи, содержащие информацию о точках. Затем осуществили выборку с использованием функций PostGIS: *ST_Distance()* и *ST_AsText()*. В качестве параметра обе функции используют объект типа geometry. Функция *ST_Distance()* рассчитывает расстояние между двумя указанными точками плоскости, а *ST_AsText()* возвращает геометрию объекта в текстовом формате *WKT (Well-Known Text)*. Формат *WKT* включает информацию о типе объекта и координаты, составляющие объект.

Литература

1. Райхлин В.А., Вершинин И.С., Гибадуллин Р.Ф. Конструктивное моделирование систем в приложении к защите данных картографии // Методы моделирования: Труды Респ. научн. семинара АН РТ. – Казань: ФЭН (Наука), 2010. Вып. 4. С. 68–95.

2. Гибадуллин Р.Ф., Пыстогов С.В. Параллельная система управления полнообъектными защищенными базами данных картографических сцен // Высокопроизводительные параллельные вычисления на кластерных системах (HPC-2012): Материалы 12-й Всерос. конф. – Нижний Новгород: ННГУ, 2012. С. 91–95.

3. Стандарты Open Geospatial Consortium – [URL: http://www.opengeospatial.org].

4. Основы работы с PostGIS – [URL: http://gis-lab.info/qa/postgis-work.html].