МАСШТАБИРОВАНИЕ БАЗ ДАННЫХ

МАСШТАБИРОВАНИЕ

{ ПОНЯТИЕ И ОПРЕДЕЛЕНИЕ

Масштабируемость — в информатике означает способность системы, сети или процесса справляться с увеличением рабочей нагрузки (увеличивать свою производительность) при добавлении ресурсов (обычно аппаратных).

Масштабируемость — важный аспект электронных систем, программных комплексов, систем баз данных, маршрутизаторов, сетей и т. п., если для них требуется возможность работать под большой нагрузкой.

Система называется **масштабируемой**, если она способна увеличивать производительность пропорционально дополнительным ресурсам.

Масштабируемость можно оценить через отношение прироста производительности системы к приросту используемых ресурсов.

В системе с **плохой масштабируемостью** добавление ресурсов приводит лишь к незначительному повышению производительности, а с некоторого «порогового» момента добавление ресурсов не даёт никакого полезного эффекта.

МАСШТАБИРОВАНИЕ

{ ВЕРТИКАЛЬНОЕ ИЛИ ГОРИЗОНТАЛЬНОЕ?

Вертикальное м**асштабирование (вглубь**) - это традиционный подход наращивания вычислительных мощностей.

Вместо небольшого сервера приобретается более крупный многопроцессорный сервер (или кластер серверов). Если этого недостаточно, возможно, понадобится мэйнфрейм или даже суперкомпьютер.

У этого подхода есть существенный недостаток - он требует неоправданно больших затрат времени и финансовых средств.

Горизонтальное масштабирование (вширь) - это активно развивающийся подход к наращиванию вычислительных мощностей системы с помощью добавления в систему новых вычислительных узлов.

Вместо одного сервера множество объединённых между собой серверов разделяют нагрузку между собой.

Этот подход позволяет оперативно наращивать производительность системы. Ограничения на масштабируемость накладывает лишь архитектура системы.

МАСШТАБИРОВАНИЕ

{ ВИДЫ И ОТЛИЧИЯ

Партиционирование (**partitioning**) - это разбиение больших таблиц на логические части по выбранным.

Партиционирование рекомендуется использовать случае, если в базе данных есть несколько огромных таблиц (обычно всю нагрузку обеспечивают всего несколько таблиц в базе данных из всех имеющихся). Причем чтение в большинстве случаев приходится только на самую последнюю их часть (т.е. активно читаются те данные, которые недавно появились). Примером тому может служить блог - на первую страницу (это последние 5-10 постов) приходится 40…50% всей нагрузки, или новостной портал или системы личных сообщений.

Партиционирование таблицы позволяет базе данных делать интеллектуальную выборку - сначала СУБД уточнит, какой партиции соответствует Ваш запрос (если это реально) и только потом сделает этот запрос, применительно к нужной партиции (или нескольким партициям). Таким образом, в рассмотренном случае, нагрузка распределена на таблицу по ее партициям.

Репликация - это синхронное/асинхронное копирование данных с ведущих серверов на ведомые (или возможно тоже ведущие) сервера. Ведущие сервера называют мастерами (master), ведомые - слейвами (slave). Для решения проблемы чтения, используется master-slave репликация. Для решения проблемы записи необходимо использовать master-master репликацию.
В работе приложения обязательно следует учитывать задержку репликации данных на между серверами - то есть время, через которое система придет в полностью целостное состояние. Существует синхронная или асинхронная репликация. В случае первой не придется заботиться о задержке репликации, но это отразится на скорости отработки запросов на изменение/вставку данных. Во втором случае могут возникнуть ситуации, когда временно нарушается целостность данных на отдельных серверах. Репликация также помогает изолировать нагрузку. Если у Вас есть проблемные тяжелые запросы, которые выполняются не так часто, но несут за собой промедления работы всей СУБД, следует выносить их на слейвы.

Шардинг (шардирование) - разделение данных на уровне ресурсов. Концепция шардинга заключается в логическом разделении данных по различным ресурсам исходя из требований к нагрузке.

Вертикальный шардинг - разделение одной базы данных веб-приложения на две и более базы данных за счет выделения отдельных модулей, без изменения логики работы веб-приложения. Например размещение разных таблиц одной базы данных на нескольких серверах.

Горизонтальный шардинг - распределение однотипных данных веб-приложения между отдельными базами данных. Примером можно привести размещение учетных записей пользователей на различных серверах с разделением по их идентификатору.

Шардинг принципиально отличается от вертикального масштабирования, которое при росте нагрузки и объёма данных предусматривает наращивание вычислительных возможностей и объёма носителей информации одного сервера баз данных, имеющее объективные физические пределы — максимальное количество поддерживаемых CPU на один сервер, максимальный поддерживаемый объем памяти, пропускная способность шины и т. д.

Шардинг обеспечивает несколько преимуществ, главное из которых — снижение издержек на обеспечение согласованного чтения (которое для ряда низкоуровневых операций требует монополизации ресурсов сервера баз данных, внося ограничения на количество одновременно обрабатываемых пользовательских запросов, вне зависимости от вычислительной мощности используемого оборудования). В случае шардинга логически независимые серверы баз данных не требуют взаимной монопольной блокировки для обеспечения согласованного чтения, тем самым снимая ограничения на количество одновременно обрабатываемых пользовательских запросов в кластере в целом.

ТЕОРЕМА САР

ПОНЯТИЕ И СВОЙСТВА

Теорема CAP — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

- СОГЛАСОВАННОСТЬ ДАННЫХ
- ДОСТУПНОСТЬ
- УСТОЙЧИВОСТЬ К РАЗДЕЛЕНИЮ

Акроним *CAP* в наименовании теоремы сформирован из первых букв английских наименований этих трёх свойств: (C)onsistency, (A)vailability, (P)artition tolerance.

Принцип был предложен профессором Калифорнийского университета в Беркли Эриком Брюером в июле 2000 года и впоследствии получил широкую популярность и признание в среде специалистов по распределённым вычислениям. Концепция NoSQL (Not only SQL), в рамках которой создаются распределённые нетранзакционные системы управления базами данных, зачастую использует этот принцип в качестве обоснования неизбежности отказа от согласованности данных.

СОГЛАСОВАННОСТЬ ДАННЫХ (CONSISTENCY)

Согласованность данных подразумевает, что во всех вычислительных узлах в один момент времени данные не противоречат друг другу. Иными словами, как только мы успешно записали данные в наше распределенное хранилище, любой клиент при запросе получит эти последние данные, независимо от того, к какому именно узлу он обратился.

ДОСТУПНОСТЬ (AVAILABILITY)

Свойство доступности означает, что любой запрос к распределённой системе завершается корректным откликом, независимо от того, к какому узлу был выполнен запрос. Иначе говоря в любой момент времени клиент может получить данные из распределенного хранилища в ответ на запрос, или же получить ответ об их отсутствии, если эти данные отсутствуют во всём хранилище.

УСТОЙЧИВОСТЬ К РАЗДЕЛЕНИЮ (PARTITION TOLERANCE)

Устойчивость к разделению подразумевает, что расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций. То есть потеря сообщений между компонентами системы (возможно даже потеря всех сообщений) не влияет на работоспособность системы. Здесь очень важный момент состоит в том, что если какие-то компоненты выходят из строя, то это тоже подпадает под этот случай, так как можно считать, что данные компоненты просто теряют связь со всей остальной системой.

ТЕОРЕМА CAP

{ СЛЕДСТВИЕ

Согласно теореме CAP, распределённые системы в зависимости от пары практически поддерживаемых свойств из трёх возможных распадаются на три класса.

CONSISTENCY – AVAILABILITY

Система, во всех узлах которой данные согласованы и обеспечена доступность, жертвует устойчивостью к распаду на секции. Такие системы возможны на основе технологического программного обеспечения, поддерживающего транзакционность в соответствии с требованиями ACID (Атомарность Согласованность Изолированность Надежность). Примерами таких систем могут быть решения на основе систем управления базами данных, таких как Miscrosoft SQL Server, или распределённая служба каталогов LDAP.

CONSISTENCY - PARTITION TOLERANCE

Распределённая система, в каждый момент обеспечивающая целостный результат и способная функционировать в условиях распада, в ущерб доступности может не выдавать отклик. Устойчивость к распаду на секции требует обеспечения дублирования изменений во всех узлах системы, в этой связи отмечается практическая целесообразность использования в таких системах распределённых пессимистических блокировок для сохранения целостности.

AVAILABILITY - PARTITION TOLERANCE

Распределённая система, отказывающаяся от целостности результата. Хотя системы такого рода известны задолго до формулировки принципа CAP (например, распределённые веб-кэши или DNS), рост популярности систем с этим набором свойств связывается именно с распространением теоремы CAP. Задачей при построении AP-систем становится обеспечение некоторого практически целесообразного уровня целостности данных, в этом смысле про AP-системы говорят как о «целостных в конечном итоге» или как о «слабо целостных»