объединения всех сегментов, его левый ребенок - результат объединения элементов с индексом от 0 до (n / 2 - 1), а правый - результат объединения элементов с индексом от (n / 2) до n и так далее до листьев. Существует две реализации построения дерева отрезков, но обе они, как от корня к листьям, так и от листьев к корню, реализуются на базе массива, размерностью ближайшей следующей степени 2. При построении снизу алгоритм поднимается от листьев к корню. Для этого просто начинаем заполнять элементы массива от большего индекса к меньшему, таким образом при заполнении элемента i его дети 2 * i + 1 и 2 * i + 2 уже будут заполнены, а при построении сверху спускается от корня к листьям.

Алгоритм построения дерева сегмента является коротким и действенным способом решения подобных задач. Временная сложность O(logN), что значительно лучше, чем O(N). Например, при массиве длинной 10^9 элементов, необходимо примерно 32 сравнения. Изменить отрезок в этой структуре так же просто — надо пройти по всем вершинам от заданной до 1-ой, и заменить его. Это также занимает O(log N) времени.

Список использованных источников:

1. Н. Вирт Алгоритмы и структуры данных. СПб.: Невский диалект, 2001. – 352 с.
2. Т. Кормен и др. Алгоритмы: построение и анализ, М.: МЦНМО, 2002.

МОДЕЛЬ СИСТЕМЫ РАСПРЕДЕЛЕННОГО ХРАНЕНИЯ ДАННЫХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Бернацкий В.В.

Бранцевич П.Ю. - кандидат технических наук, доцент

Надежность, доступность, согласованность данных являются основными требованиями предъявляемыми к базам данных. Рост объема и трафика данных, а также использование распределенных вычислений требуют возможности горизонтальной масштабируемости баз данных. В докладе предложена модель системы распределенного хранения данных удовлетворяющая данным требованиям.

Целью данной работы является разработка модели системы распределенного хранения данных с сохранением основных требований к базам данных. Эта система должна быть отказоустойчивой, выход из троя отдельных узлов системы не должны приводить к остановке работы всей системы. Также эта система должна быть доступной: операции чтения и записи должны быть быстрыми. Операции записи не должны приводить систему в несогласованное состояние. Вся нагрузка в системе должна быть равномерно распределенной между всеми узлами (вершинами).

Исходя из требований к данной системе были выделены следующие задачи: определение структуры хранимых данных, обеспечение равномерного распределения нагрузки по всем узлам, обеспечение высокой доступности и согласованности данных.

Разрабатываемая система будет представлять собой коллекцию элементов ключ, имя признака, значение признака, где ключ является уникальным идентификатором логической сущности. Имя признака является именем атрибута этой логической сущности, а значение признака является значением атрибута заданного в поле имя признака. Данный подход позволяет обеспечить равномерное распределение данных основываясь на значении ключ, а добавление понятия признак позволяет уменьшить размер передаваемых данных и уменьшить количество неудавшихся транзакций (за счет применения изменений не ко всей сущности, а лишь к ее части).

Ключевым моментом для обеспечения высокой доступности и надежности является реплицируемость данных - хранение копий одних и тех же данных на нескольких вершинах. Это позволяет распределить запросы для работы с этими данными по вершинам, а в случае выхода из троя отдельных вершин продолжить работу с этими данными используя оставшиеся копии. В данной системе репликации будут подвержены сущности.

В распределенных системах хранения добиться полной согласованности данных практически невозможно, поэтому здесь применяют концепцию согласованности в конечном счете. Данная концепция является менее строгой формой согласованности. Согласно ней данные будут согласованы если никаких новых изменений не происходило. Для обеспечения выполнения данного условия можно использовать подход основной копии. Согласно нему есть 2 категории копий данных (сущности): основная и неосновная. Операции по изменению данных происходят лишь над основной копией данных, а все изменения затем транслируются на оставшиеся копии. Таким образом мы получаем надежность изменения данных как в синхронной модели, но скорость чтения увеличивается пропорционально количеству неосновных копий.

Равномерное распределение значений по узлам осуществляется за счет подсчета хеша ключа сущностей. В зависимости от того, в какой диапазон попало значение, определяется вершина для хранения сущности. Аналогичным образом определяются вершины для хранения других копий значений сущности.

Для обеспечения отказоустойчивости необходимо минимизировать вред от выхода из строя (недоступность) узлов системы. Это можно достичь путем равномерного распределения основных и неосновных копий сущностей и ведением списка приоритета узлов для каждой сущности. Данный список

позволяет для каждой сущности в случае отключения или выхода из строя узла с основной копией выбрать другой узел с копией, который будет использоваться в качестве основного для данной сущности. В совокупности с равномерным распределением основных и неосновных копий по узлам, в случае отключения одной из вершин, произойдет автоматическое перераспределение ролей копий и работа всей системы продолжится. Отдельным случаем стоит рассмотреть вариант при разделении сети на 2 части, где связь в каждой половины будет работать, но соединение между вершинами двух половин будет утеряно. Для того, чтобы у нас не было параллельного функционирования двух основных копий для некоторых сущностей, можно добавить условие, что копия может являться основной лишь при наличии связи с большей половиной узлов, содержащих копию данной сущности.

Таким образом, была предложена модель системы для обеспечения распределенного хранения данных. Выбранная структура данных позволила обеспечить возможность распределенного хранения данных, а также увеличить надежность системы. В качестве базы для обеспечения доступности и надежности использован принцип реплицируемости данных. А использование метода основной копии, хеш-функции для определения вершин для хранения сущностей и алгоритма поведения системы в случае потери связи между вершинами позволили добиться согласованности, доступности, распределения нагрузки и надежности.

Список использованных источников:

1. Mustaque Ahamad , Mostafa H. Ammar , Shun Yan Cheung - "Replicated Data Management in Distributed Systems" - Emory University, 1992

2. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels - "Dynamo: Amazon's Highly Available Key-value Store" - Amazon.com, 2007

МЕТРИКИ ОЦЕНКИ ФУНКЦИОНИРОВАНИЯ ПРИЛОЖЕНИЙ ПОД УПРАВЛЕНИЕМ JAVA EE СЕРВЕРОВ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Бродницкий В.В.

Куликов С. С. – к. т. н., доцент

Мониторинг состояния и производительности приложений является критически важной составляющей при поддержке работы приложений, и чем больше возрастает сложность и масштаб системы, тем большую важность приобретает анализ её работы для сохранения стабильности функционирования и целостности данных. Большинство Java-приложений являются не одиночными программными средствами, а состоят из множества различных компонентов, обладают масштабируемой структурой, что значительно увеличивает зависимость состояния системы от ряда внешних факторов.

Количество показателей и метрик работы приложения, доступных для мониторинга, зависит от типа анализируемого программного средства, типа его компонентов и использованных сервисов, а также от конкретной реализации виртуальной машины JAVA. Важной задачей является группировка их по критерию описываемой области, что позволит увеличить эффективность их выборки и анализа. Можно выделить несколько ключевых областей, характеризующих стабильность и производительность любой программной системы:

Показатели виртуальной машины. В данную группу выделяют частоту событий виртуальной машины (количество вызовов и откликов методов, ошибок), степень параллелизма потоков, информация о блокировках, данные о компиляции.

Показатели загруженности памяти. Основными критериями данной группы являются показатели отношения свободной и занятой памяти к общему её количеству. Однако важными показателями являются также сообщения менеджеров памяти о превышении пороговых значений загруженности памяти для определённых пулов и необходимости её увеличения.

Показатели сборщика мусора. Для платформ с автоматическим управлением памятью сборка мусора является ключевым моментом, от эффективности работы которого зависит общая производительность системы. Здесь следует выделить показатели количества произведённых сборок, освобождённой в них памяти, подробные данные о состоянии поколений.

Показатели операционной системы. Так как приложения, выполняемые виртуальной машиной, располагаются на конкретном аппаратном и программном обеспечении, то характеристики их эффективности и готовности непосредственно влияют на показатели производительности самой программной системы. Очевидно, что мониторинг их состояния является важной частью диагностики любого программного средства. В основном здесь выделяют уровень загрузки процессора, доступность и скорость передачи по сети, операций ввода-вывода информации.

Виртуальная машина Java представляет собой сложную многоуровневую систему, включающую в себя множество компонентов и крайне тяжело выделить единую группу показателей, полностью характеризующую её производительность и надёжность. В 5 версии был добавлен специальный пакет объектов и интерфейсов, который позволяет выделить определённые группы метрик, сгруппированные по описанию определённых подсистем виртуальной машины. Мониторинг