Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра Автоматизации Систем Вычислительных Комплексов

ШПИЛЕВОЙ Владислав Дмитриевич

Исследование механизмов синхронизации времени от различных аппаратных источников в ядре Linux

КУРСОВАЯ РАБОТА

Научный руководитель:
А.В.Герасёв

Москва, 2015

Аннотация

Содержание

1 Введение

В последнее время системы, используемые в самых разных отраслях науки, промышленности, бизнеса и развлечений, все больше приобретают тенденцию развиваться не *"вертикально"*, то есть наращивая мощности своих компонент, а *"горизонтально"*, увеличивая количество этих компонент, что обходится дешевле и во многих случаях эффективнее. Но вместе с ростом масштабов систем растет и сложность управления и работы с ними.

В распределенных системах, будь то большой комплекс программ и аппаратных средств, рассредоточенных на большой площади, или один компьютер с подключенными к нему устройствами (мониторы, внешние диски, модемы и т.д.), всегда возникает необходимость синхронизации работы между различными узлами таких систем. Прикладные программы, работающие в системе, должны как-то взаимодействовать с другими узлами (непосредственно, или через некоторый центр управления), и обычно в качестве механизма взаимодействия используется обмен сообщениями.

В сообщения записываются наборы данных и, как правило, метка времени, которая соответствует тому моменту времени, когда сообщение было отправлено. В разных системах и даже узлах одной системы время может считаться по-разному, и не обязательно это будет время астрономическое: это может быть значение некоторого аппаратного счетчика, инкрементирующегося через некоторые промежутки времени. Обычно частота инкрементации счетчиков имеет порядок десятков наносекунд, а столь высокая точность в сочетании с несовершенством процессоров и условий, в которых они работают, всегда имеет некоторую погрешность, которая может и не быть значимой каждый отдельный раз, но такие ошибки накапливаются, что при игнорировании этого факта может привести к серьезным ошибкам. Пусть, например, два банкомата отправляют центральному серверу запрос сначала на зачисление денег на счет, а затем на их списание, но в силу перегрузки сети, сообщения пришли одновременно - если время на этих банкоматах было неверно синхронизировано, то запрос на списание денег не сработает, и клиент, стоящий у банкомата, не получит своих денег.

Определим, какие метки времени могут использоваться в сообщениях, которыми обмениваются узлы систем. В UNIX-системах время определяется, как количество секунд, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года, время с этого момента называют "временем с начала эпохи". Такое время вполне подходит для большинства прикладных программ и приложений и для отображения пользователю, а так же для каких-либо служебных пометок, не задействованных в синхронизации, но для высоко точных систем недостаточно порядка секунд. Сервера любой крупной IT-компании или популярного сайта ежесекундно принимают тысячи запросов от пользователей, что делает синхронизацию на основе секундных меток непригодной.

Современные процессоры позволяют получать время с точностью до наносекунд благодаря развитию процесса производства кристаллов для процессоров. Разумеется, такая высокая точность подразумевает некоторые погрешности, но современные алгоритмы синхронизации времени позволяют эти погрешности сгладить. Использовать везде метки с наносекундной точностью нельзя, так как процессоры - дорогие, а распределенная система подразумевает дешевые компоненты. В силу этого ограничения в большинстве узлов системы временем считается значение некоторого аппаратного счетчика, точность которого может быть на несколько порядков хуже, чем наносекунда. Например, значение такого счетчика может инкрементироваться каждые 7 микросекунд (+- некоторая погрешность).

Какой бы точностью ни обладал счетчик, он не может убывать. Это утверждение логично, так как, например, сообщение не может иметь метку, соответствующую более раннему времени, чем сообщение было отправлено. Нарушение этого фундаментального правила может повлечь за собой тяжелые последствия. Значение счетчика в программе можно получать через вызовы специальных функций, которые описаны в документации к каждому конкретному устройству-узлу системы. Очевидно, что запрос на получение значения счетчика тоже занимает время, так как происходит обращение к операционной системе, считывание значения и передача его в программу, так что следует учитывать, что полученное значение счетчика может уже не соответствовать его текущему значению.

С ростом масштабов распределенных систем так же растет физическое расстояние, разделяющее узлы, что приводит к необходимости создания сетей, по которым узлы будут взаимодействовать. В больших сетях могут происходить коллизии, разрывы соединения, перегрузки сети, что приводит к новой проблеме -

пока сообщение от одного узла дойдет до целевого узла, значение счетчика в нем может потерять актуальность.

Переходя к конкретным примерам распределенных систем, можно назвать:

1) Hadoop-кластер, который может иметь несколько сотен узлов, где машины могут выходить из строя по несколько штук в неделю, где обрабатываются сотни терабайт данных, а центральных узлов может быть несколько

2) Система контроля полета в летательных аппаратах, где от точности синхронизации компонент системы могут зависеть сотни жизней

3) Банковские системы, которые могут простираться на целые города, страны и материки

4) Ядро операционных систем Linux, состоящее из большого количества модулей, которые могут добавляться и удаляться уже в процессе работы ядра.

Мы рассмотрим четвертый из описанных выше примеров. На сегодняшний день ядро Linux - монолитная система, закрытая от прикладных программ, запущенных на той же машине. В ядре работают расширения ядра - модули, а так же драйвера устройств, которые обеспечивают взаимодействие с устройствами (считывание из них данных, запись, отключение, подключение). Ядро можно назвать распределенной системой: модули и устройства с соответствующими драйверами - это ее узлы. И тем узлам, которые друг с другом взаимодействуют, необходимо это взаимодействие синхронизировать.