УДК 004.75

Ю.А. Алдунин

СИНХРОНИЗАЦИЯ ВРЕМЕНИ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ, В ЗАВИСИМОСТИ ОТ ВЫБРАННОЙ МОДЕЛИ НЕПРОТИВОРЕЧИВОСТИ

Рассматриваются основные методы синхронизации времени в распределенных системах. Дается краткая характеристика моделей непротиворечивости. Проводится анализ возможности использования различных методов синхронизации времени для обеспечения функционирования приведенных моделей непротиворечивости.

Введение. Синхронизация времени в распределенных системах хранения и обработки данных играет важную роль, поскольку без нее невозможно согласовать работу отдельных узлов системы. Но особенно она важна при синхронизации данных в подобных системах. Независимо от выбранной модели непротиворечивости данных для синхронизации необходимо максимально точное совпадение текущего времени синхронизируемых узлов в момент начала как предыдущей синхронизации данных, так и текущей (должно соблюдаться единство данного временного интервала для точного определения синхронизируемого объема информации), а также в промежутке между этими событиями для определения очередности наступления событий. Например, в момент начала последней синхронизации системное время на одном из узлов было на секунду больше, чем на другом; если за начало интервала синхронизации взять системное время второго узла, то изменения, произошедшие на первом узле за эту секунду, будут утеряны.

Цель работы – проанализировать и систематизировать существующие методы синхронизации времени в распределенных системах и оценить их применимость для различных моделей непротиворечивости данных в распределенных системах хранения и обработки данных.

Теоретическая часть. Любой компьютер имеет механизм подсчета времени. Хотя его обычно называют «часы», это не совсем верно – это, скорее, таймер. Таймер реализован следующим образом: имеется кристалл кварца; находясь под напряжением, он колеблется с определенной частотой, которая зависит от свойств конкретного кристалла и подаваемого напряжения. Каждое колебание вызывает изменение счетчика, ответственного за формирование системного времени. Таким образом, из-за различной частоты колебания кристалла имеет место рассинхронизация времени в распределенной системе.

Рассмотрим основные методы синхронизации времени в распределенных системах, сразу сформулировав два основных требования:

– в определенные моменты системное время узлов системы должно максимально совпадать;

– при синхронизации недопустим перевод часов в обратную сторону (уменьшение системного времени), поскольку это может привести к выходу за начало интервала синхронизации либо к нарушению работы механизма определения очередности событий; таким образом, мы можем оперировать только переводом времени вперед либо замедлением его хода.

Если в распределенной системе присутствует узел, имеющий внешние физические часы либо приемник сигналов точного времени, например WWV, то задача синхронизации сводится к необходимости синхронизации остальных узлов системы с данным эталонным.

Алгоритм Кристиана. Периодически каждый узел посылает эталонному запрос его текущего времени. Эталонный узел максимально быстро отвечает на этот запрос. Но следует помнить, что между отправкой сообщения эталоном и его получением узлом проходит некоторое время, которое непостоянно и зависит от текущей загрузки сети передачи данных. Для оценки этого времени Кристиан предложил принимать время передачи ответа как половину времени между отправкой запроса и получением ответа. Для повышения точности следует производить серию таких замеров.

Для случая, когда такой источник времени отсутствует, предназначены следующие методы.

Алгоритм Беркли. Данный алгоритм предназначен для случая, когда источник точного

времени отсутствует. Сервер времени опрашивает все узлы для выяснения их текущего времени, усредняет это значение и рассылает команды на установку нового значения времени либо замедление часов.

Усредняющие алгоритмы. Семейство алгоритмов, имеющих общий принцип работы. С заданной периодичностью все узлы системы производят широковещательную рассылку текущего времени. Затем в течение определенного времени принимают сообщения о текущем времени от других узлов. Когда все сообщения приняты, запускается алгоритм вычисления текущего времени. Простейший вариант – усреднение полученных значений. Алгоритм может быть усовершенствован путем отброса m самых больших и m самых маленьких значений для защиты от заведомо неверных значений. Другой путь усовершенствования алгоритма – попытка оценки времени прохождения сообщения от источника (возможно с учетом знаний о топологии сети) для получения более точных значений.

Отметки времени Лампорта. При обмене данными между взаимодействующими узлами происходит также обмен сведениями об их локальном времени. Если отметка времени, указанная в сообщении, оказывается меньше, чем локальное время узла, такая ситуация считается штатной, так как сообщение было отправлено ранее, чем получено. Если же в сообщении указано более раннее время, чем локальное время узла, то есть сообщение было послано позже, чем получено, делается вывод о необходимости коррекции системного времени узла. Время выставляется в значение, равное отметке в сообщении плюс единица.

Подведем краткие итоги.

Алгоритм Кристиана. Достоинства: простота реализации, высокая эффективность в небольших сетях и сетях с малой загрузкой: поскольку в системе присутствует источник точного времени и синхронизация производится по нему, то время в системе в целом соответствует реальному.

Недостатки: требует внешнего источника точного времени, не имеет встроенной защиты от перевода часов в обратную сторону, некорректно работает в сетях с резкими скачками загрузки сети, не позволяет корректно устанавливать время в случае, если запрос и ответ передаются по разным маршрутам (требуется разное время для передачи данных сообщений).

Алгоритм Беркли. Достоинства: простота реализации, высокая эффективность для систем, некритичных к отклонениям по времени.

Недостатки: требуется выделенный узел – сервер времени, поскольку источник точного времени отсутствует, а за общесистемное время принимается усредненное время узлов, оно может иметь мало общего с реальным временем, узлы с замедленным для коррекции временем (все еще некорректным) оказывают влияние на общесистемное время.

Усредняющие алгоритмы. Достоинства: не требуется выделенный узел – сервер времени, высокая эффективность подстройки под конкретные задачи.

Недостатки: сильная зависимость эффективности от загрузки сети, низкая степень синхронизации, не обеспечивают установки на всех узлах одинакового времени (часть сообщений на конкретном узле может не быть получена, не получена вовремя, отброшена как заведомо ложная).

Отметки времени Лампорта. Достоинства: не требуется выделенный узел – сервер времени, высокая эффективность при малом числе узлов, высокая степень синхронизации (время переводится только вперед, и нет необходимости ждать эффекта от замедления на спешащих узлах).

Недостатки: не принимается в расчет время передачи сообщений между узлами, поскольку имеет место постоянный перевод часов вперед – общесистемное время имеет мало общего с реальным.

Кратко рассмотрим основные модели непротиворечивости данных.

Репликация позволяет повысить надежность системы либо ее производительность. Повышение надежности достигается за счет наличия нескольких копий данных, что позволяет в спорных ситуациях апеллировать к нескольким источникам одних и тех же данных. Повышение производительности достигается за счет увеличения совокупной производительности системы хранения и обработки данных либо за счет территориально более близкого расположения источника данных к пользователю.

Но репликация, кроме очевидных преимуществ, имеет и недостатки, так как она требует больших затрат на поддержание в актуальном состоянии нескольких копий данных. В зависимости от требований к системе и принципов ее функционирования применяют несколько моделей непротиворечивости данных. Следует иметь в виду, что чем проще модель, тем легче ее внедрить и использовать, но тем ниже ее эффективность. Рассмотрим основные из них более подробно.

Строгая непротиворечивость. Строгая непротиворечивость строится на следующем

принципе: всякое чтение элемента данных должно возвращать значение, соответствующее результату последней записи данного элемента. Другими словами, если производится запись значения в одну копию данных, то чтение этого значения из любой копии в любой момент (до следующей записи) должно возвращать именно это значение.

Последовательная непротиворечивость. Последовательная непротиворечивость характеризуется следующим утверждением: результат любого действия такой же, как если бы операции чтения и записи всех процессов в хранилище данных выполнялись в некотором последовательном порядке, причем операции каждого отдельного процесса выполнялись бы в порядке, определяемом его программой. То есть если два процесса последовательно изменяют элемент данных ***x***, записывая значения ***a*** и ***b*** соответственно, а любой другой процесс при чтении получает последовательно значения ***a*** и ***b*** либо ***b*** и ***a***, то данное условие выполняется, но если возникает ситуация, когда один процесс последовательно получает значения ***a*** и ***b***, а другой ***b*** и ***a***, то последовательная непротиворечивость нарушается.

Причинная непротиворечивость. Причинная непротиворечивость является ослабленным вариантом последовательной. Для ее выполнения необходимо условие: операции записи, потенциально связанные причинно-следственной связью, должны наблюдаться всеми процессами в одинаковом порядке, а параллельные операции записи – в произвольном. Таким образом, если процесс 1 записывает в элемент данных ***x*** значение ***a***, а затем процесс 2 производит чтение элемента ***x***, после чего записывает в него новое значение ***b*** (то есть новое значение потенциально зависит от старого), то все процессы должны видеть сначала значение ***a***, а потом ***b*** (либо сразу ***b***), но ни в коем случае не сначала ***b***, а потом ***a***. С другой стороны, если процесс 1 записывает в элемент данных ***x*** значение ***a***, а затем процесс 2 записывает в него же новое значение ***b*** (не произведя предварительно его чтения), то неважно, в какой последовательности значения ***a*** и ***b*** будут прочитаны другими процессами.

Непротиворечивость FIFO. Следующим шагом ослабления требований непротиворечивости является непротиворечивость FIFO. Ее основным условием является следующее: операции записи одного процесса должны наблюдаться остальными в том порядке, в котором они осуществляются, а операции разных процессов – в произвольном.

Теперь рассмотрим применимость методов синхронизации времени для каждой из моделей непротиворечивости.

Строгая непротиворечивость. Реализация строгой непротиворечивости в распределенных системах невозможна. Таким образом, рассматривать ее далее не имеет смысла.

Последовательная непротиворечивость. Последовательная непротиворечивость критична к точности синхронизации времени во всей системе, поэтому наиболее подходящим для нее методом является метод отметок времени Лампорта (при использовании этого метода часы переводятся только вперед и поэтому синхронизируются «мгновенно»), алгоритм Кристиана может вызывать, а Беркли вызывает замедление времени на отдельных узлах (чтобы избежать перевода часов назад), что влечет за собой невозможность «мгновенной» синхронизации времени. Усредняющие алгоритмы неприменимы ввиду отсутствия гарантии совпадения времени на всех узлах.

Причинная непротиворечивость. Для модели причинной непротиворечивости важен порядок выполнения команд над одними и теми же данными, даже на разных узлах. Таким образом, данная модель выдвигает жесткие требования по синхронизации времени в распределенной системе. Поскольку каждый из обращающихся к участку памяти узлов инициирует обмен данными, в этом случае целесообразно использовать метод отметок времени Лампорта. Также возможно использование алгоритма Кристиана при условии, что в сети присутствует источник точного времени, топология сети позволяет организовать с ним синхронизацию и реализована блокировка перевода часов назад. Алгоритм Беркли применим при условии, что в данной системе отсутствуют часы, отстающие на большие величины, и присутствует выделенный сервер времени. Усредняющие алгоритмы неприменимы, так как не обеспечивают установки одинакового времени на всех узлах.

Непротиворечивость FIFO. Поскольку для этой модели непротиворечивости данных важен лишь порядок выполнения в рамках одного процесса, то модель некритична к выбору метода синхронизации. Единственное условие: поскольку порядок выполнения команд в рамках одного узла важен, то сдвиг времени назад, при синхронизации недопустим.

Вывод. Таким образом, можно сделать вывод, что строгая непротиворечивость в многопоточных, и, в частности, в распределенных системах неприменима. Непротиворечивость FIFO не накладывает ограничений на использование метода синхронизации времени (синхронизация в

данном случае даже не является необходимым условием). Для моделей последовательной и причинной непротиворечивости наиболее подходит метод отметок времени Лампорта, использование алгоритмов Кристиана и Беркли возможно с некоторыми ограничениями. Усредняющие алгоритмы неприменимы кроме случая непротиворечивости FIFO.

Библиографический список

1. *Cristian F.* Probabilistic Clock Synchronization Distributed Computing, vol. 3, pp. 146-158, 1989.

2. *Drummond R. and Babaoglu O.* Low-Cost Clock Synchronization Distributed Computing, vol. 6, pp. 193-203, 1993.

3. *Gusella R. and Zatti S.* The accuracy of the Clock Synchronization Achieved by TEMPO in Berkley UNIX 4.3BSD IEEE Trans. Softw. Eng., vol. 15, no. 7, pp. 847-853, July 1989.

4. *Hutto P. and Ahamad M.* Slow Memory: Weakening Consistency to Enhance Concurrency in Distributed Shared Memories Proc. Tenth Int'l Conf on Distributed Computer Systems. IEEE pp. 302-311, 1990.

5. *Lamport L.* Concurrent Reading and Writing of Clocks ACM Trans. Comp. Syst., vol. 8, no. 4, pp. 305-310, Nov. 1990.

6. *Lamport L.* How to Make a Multiprocessor Computer that Correctly Executes Multiprocessor Programs IEEE Trans. Comp., vol. C-29, no. 9, pp. 690-691, Sept. 1979.

7. *Mosberger D.* Memory Consistency Models Oper. Syst. Rev., vol.27, no. 1, pp.18-26, Jan. 1993.

8. *Ramanathan P., Shin K., and Butler R.* Fault-Tolerant Clock Synchronization in Distributed Systems IEEE Computer, vol. 23, no. 10, pp.-33-42, Oct. 1990.