BAKI UNİVERSİTETİNİN XƏBƏRLƏRİ

№2 Fizika-riyaziyyat elmləri seriyası 2005

ОСНОВНЫЕ ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕ К СИСТЕМЕ УПРАВЛЕНИЯ ТРАНЗАКЦИЯМИ

Э.Н.ИСРАФИЛОВА

В работе рассматриваются основные требования, предъявляемые к системе управления транзакциями. Более подробно описывается требование - отказоустойчивость распределенных систем. Служба отказоустойчивости представляется как иерархическая система, состоящая из уровней: виртуального кольца; WATCH-службы; управления фиксацией транзакций; резервирования файлов. Предлагается алгоритм, обеспечивающий отказоустойчивость распределенных систем. Первая часть алгоритма, уровня виртуального кольца, предназначена для распознавания отказа или восстановления узла. Каждое такое распознавание инициирует вторую часть алгоритма, смысл которой заключается в широковещательном распространении сведений. Эта часть алгоритма относится ко второму уровню, т.е. WATCH -службе.

Введение. Системы распределенной обработки информации на сегодня представляют одну из наиболее прогрессивных форм организации средств вычислительной техники. Нашедшие воплащение в виде компьютерных сетей и распределенных баз данных, они представляют конечным пользователям несравненно более широкий набор информационных и вычислительных услуг, обладают потенциально повышенной надежностью и доступностью, чем отдельно взятые компьютеры. Распределенные системы обработки информации (в дальнейшем будем называть их просто распределенными системами) представляют собой множество территориально отдаленных друг от друга узлов, объединенных системой передачи данных и взаимодействующих посредством обмена сообщениями. Такие системы обеспечивают распределенную обработку данных, при которой прикладной процесс (программа или оператор) из одного узла может обращаться к информации любого другого узла.

В узлах распределенной системы (РС) хранится и обновляется информация, организованная в виде баз данных (БД), контролируемых локальными системами управления базами данных (СУБД). В РС узлы способны взаимодействовать друг с другом, обеспечивая пользователю услуги СУБД. Как правило, РС в этом случае называется системой с распределенной базой данных (РБД), представляющая собой совокупность логически связанных БД, функционирующих в различных узлах, и потоков прикладных задач. Для потоков прикладных задач характерна

возможность одновременного использования нескольких БД как единого целого. В системе с РБД пользователь должен иметь возможность доступа к каждой БД сетевой системы, не задумываясь над тем, какая схема соответствует той или иной БД. Система с РБД не должна зависеть от изменений как аппаратного оборудования, так и ее прикладных программ и данных. Периодически реорганизация части системы не должна влиять на остальные ее части.

Важными функциями РС, работающих с РБД, являются управление параллельным выполнением прикладных процессов и обеспечение отказоустойчивости, т.е. способности системы правильно выполнять основные функции независимо от наличия неисправностей в аппаратуре или ошибок в программе. Эти две функции должны рассматриваться в тесной взаимосвязи. Однако при параллельной работе прикладных процессов содержимое РБД может оказаться противоречивым. Поэтому система управления транзакциями (СУТ), которая является важным компонентом РБД, должна так управлять параллельным выполнением прикладных процессов, чтобы целостность РБД не нарушилась.

Основные требования. Координация параллельной обработки транзакций в рамках всей РС проводится СУТ, имеющей своих агентов в каждом узле. Основные требования, которые предъявляются к СУТ следующие:

- атомарность;
- сериализуемость;
- статистическая справедливость;
- производительность;
- отказоустойчивость.

Требование атомарности означает, что любая транзакция, т.е. все множество ее подтранзакций, должна быть полностью выполнена либо, если выполнение ее невозможно, должны быть предприняты меры, чтобы никаких следов от ее обработки в РБД не осталось. Принцип атомарности часто называют также принципом «все или ничего».

Наряду с обеспечением атомарности выполняемых транзакций к СУТ предъявляется требование сериализуемости. Некоторый распределенный план называется сериализуемым, если он эквивалентен, по крайней мере одному последовательному, т.е. сериальному плану.

Должна быть обеспечена статистическая справедливость по отношению ко всем инициированным транзакциям. Это означает, что любая инициированная некоторым узлом транзакция должна быть выполнена за приемлемое время, более того, ни одна из конкурирующих друг с другом транзакций не должна обладать какими-либо существенными преимуществами, в вероятностном смысле, по отношению к другим.

Естественно, что СУТ должна обеспечить максимальную производительность РС за счет высокой степени параллелизма выполнения подтранзакций в различных узлах.

Еще одним требованием, предъявляемым к СУТ, является отказоустойчивость РС.

Отказоустойчивость – одна из надежных характеристик компьютерных систем, отражающая способность выполнять возложенные на систему функции при различных отказах. Наиболее характерными классами компьютерных систем, для которых ведутся интенсивные поисковые исследования и инженерные разработки с целью повышения их отказоустойчивости, являются системы управления техническими и технологическими объектами, а также системы массового сервиса, как правило, реализующие транзактную технологию.

Свойством отказоустойчивости обладают многие технические системы, но компьютерные являются в данном случае наиболее характерными, так как они способны адаптироваться к изменяющимся условиям, т.е. перестраивать алгоритмы своего функционирования в широком диапазоне. Среди компьютерных систем свойство отказоустойчивости в наибольшей степени присуще, во всяком случае, потенциально распределенным системам, функционирующим на основе вычислительной сети. Мало того, можно утверждать, что полезное функционирование РС, не обладающей свойством отказоустойчивости, попросту невозможно.

Службу отказоустойчивости представим как иерархическую систему, состоящую из следующих уровней: виртуального кольца; $WATCH$ -службы; управления фиксацией транзакций; резервирования файлов.

Виртуальное кольцо предназначено для быстрейшего обнаружения изменения состояния узла (т.е. отказа или восстановления) при малом числе пересылаемых служебных сообщений. Виртуальное кольцо образуется следующим образом. Узлы сети нумеруются последовательными натуральными числами $1,...,N$. Эта нумерация и определяет кольцевую упорядоченность узлов сети согласно отношению следования:

$$k \to (k+1) mod\, N .$$

При выполнении данного отношения будем называть узел $(k+1)$ старшим, а узел k - младшим в данной паре. Стрелка означает предшествование. Отметим, что номера присваиваются узлам в произвольном порядке, возможно, из прагматических соображений, например, исходя из правила: два узла с последовательными номерами не должны быть слишком далеки друг от друга по числу промежуточных транзактных узлов. Полученное, согласно введенной кольцевой упорядоченности множество узлов будем называть виртуальным кольцом. Узел $(k-1)$ считается доступным со стороны узла k, если:

а) между узлами $(k-1)$ и k существует связь;

б) узел $(k-1)$ способен обрабатывать запросы узла k;

в) узел k знает о выполнении двух первых условий.

Последнее условие будет соблюдаться, если обязать $(k-1)$ периодически или по требованию k посылать сообщение “I am up”, которое будем называть up - сообщением. Если за определенный интервал времени узел k не получает up - сообщения, то считается, что узел $(k-1)$ недоступен.

В данной модели различают два состояния узлов РС: узел функционирует (SUP) и узел отказал $(SDOWN)$. Ниже предлагается алгоритм, обеспечивающий отказоустойчивость РС. Первая часть алгоритма, уровня виртуального кольца, называется $Control$ и предназначена для распознавания отказа или восстановления узла. Каждое такое распознавание инициирует вторую часть алгоритма, которая называется $Change$, смысл которой заключается в широковещательном распространении сведений, полученных от $Control$, и дополнительно собранных сведений о состоянии всех узлов. Эта часть алгоритма относится ко второму уровню, т.е. $WATCH$ -службе.

Алгоритм

Часть $Control$.

Шаг 1. Узел k посылает по виртуальному кольцу up - сообщение узлу $(k+1)$ и заводит интервальный таймер на период T_S, который известен всем узлам сети. Каждое up - сообщение содержит временную метку, имеющейся у данного узла версии таблицы состояний $(State-table)$ узлов, что позволяет узлам выравнивать свои таблицы до получения единой согласованной версии.

Шаг 2. После того как узел $(k+1)$ получил up - сообщение, он заводит интервальный таймер на период $T_{Control}$ - максимально допустимый период между поступлениями двух up - сообщений:

$$T_{Control} = T_S + T_{\max} - T_{\min},$$

где $T_{\max}$ - максимальное время передачи сообщения между двумя произвольными узлами; $T_{\min}$ - минимальное время передачи.

Шаг 3. Если новое up - сообщение не поступает от узла k в заданный интервал времени, то таймер $T_{Control}$ срабатывает. Узел k считается недоступным, и инициируется вторая часть алгоритма $Change$, корректирующая таблицу состояний $State-table$, содержащую “мнения” данного узла о состоянии всех остальных.

Шаг 4. Узел, обнаруживший согласно данному алгоритму недоступность предыдущего узла, должен взять его под контроль вплоть до вос-

становления его доступности, а также управлять теми узлами, которые формально контролировались последними. Таким образом, в ареал контроля узла k включается предшествующий узел $(k-j)$ - доступный, по "мнению" k, и все узлы $(k-i)$, предшествующие k и следующие за $(k-j)$ - недоступны.

Часть $Change$.

Шаг 1. Узел, обнаруживший нерегулярное событие и инициировавший часть алгоритма $Change$, посылает в широковещательном режиме сообщение $state$, содержащее идентификатор координатора и временную метку сообщения во все узлы сети и стартует таймер с периодом:

$$T_{Change} = 2 * T_{\max} - T_{state},$$

где T_{state} - максимальное время обработки сообщения $state$ удаленного узла.

Шаг 2. Все узлы отправляют координатору ответ $acks$, содержащий идентификатор ответившего узла, и временную метку исходного сообщения $state$.

Шаг 3. В зависимости от полученных и неполученных координатором сообщений $acks$, маркируется состояние узлов, таких как SUP или $SDOWN$. Составляется новая версия таблицы состояния узлов и рассылается в широковещательном режиме сообщения $state$, содержащего данную таблицу состояний узлов, ко всем узлов, ко всем узлам маркированным как UP.

Шаг 4. Узлы, приняв новую версию таблицы состояния узлов, немедленно посылают up - сообщения в свои новые ареалы широковещания. В узлах снова рестартуют таймеры T_S и $T_{Control}$. Заметим, что up - сообщение также несет действительную временную метку таблицы состояния (для отличия версии таблицы) и идентификатор координатора этой версии (для разрешения возможных конфликтов).

Отметим, что если отказов не было в течение времени большего, чем интервал обработки всех процедур алгоритма $Change$, то временные метки всех таблиц $State-table$ становятся идентичны друг другу.

Решающие правила по выбору каждым узлом новой версии $State-table$ основаны на предположении, что во всех узлах имеются часы, показания которых не могут отличаться друг от друга больше, чем на ε единицу времени, т.е. для $\forall i,k \;\; t_i - t_k \le \varepsilon$, где t_i, t_k - показания часов узла i и узла k. На шаге 1 алгоритма $Change$, координатор (узел k) метит версию текущим временем своего узла, т.е. $VST = t_k$. Допустим в некий узел k последовательно пришли две существенно различные вер-

сии $State-table$ с временными метками VST и VST^* и пусть $VST^* > VST$. Тогда:

а) $VST^* - VST \geq 2(T_{max} - T_{min})$, то принимается версия $State-table$, помеченная VST^*;

б) если неравенство не выполняется, то необходимо повторное выполнение алгоритма $Change$;

в) если пришли две одинаковые версии, но с разными временными метками, то они принимаются, причем версия метится меткой VST^*.

Кроме описанного алгоритма $Change$ *WATCH*-служба организует также слежение за состоянием произвольных узлов, заданных прикладным процессом, и докладывает о любом изменении состояния узла данному прикладному процессу.

Алгоритмы третьего и четвертого уровней будут рассмотрены в следующих публикациях.

ЛИТЕРАТУРА

1. M.T.Ozsu, P.Valduries. Principles of Distributed Database Systems, Prentice-Hall, 1999.
2. Flavin Cristan Understanding Fault-Tolerant Distributed Systems // Comm. ACM. – 1991. – 43, N2. – p.56-78.

TRANZAKSİYALARIN İDARƏ OLUNMASI SİSTEMİNƏ QOYULAN ƏSAS TƏLƏBLƏR

E.N.İSRAFİLOVA

ANNOTASİYA

İşdə tranzaksiyaların idarə olunması sisteminə qoyulan əsas tələblərə baxılır. Paylanmış sistemlərin nasazlığa dayanıqlığı tələbi geniş şəkildə təsvir olunur. Paylanmış sistemlərin nasazlığa dayanıqlığını təmin edən alqoritm təklif edilmişdir.

MAIN REQUIREMENTS TO TRANZACTION CONTROL SYSTEMS

E.N.ISRAFILOVA

ABSTRACT

In this paper main requirements to tranzaction control systems are considered. An algoritm providing fault-tolerant distributed systems is proposed.