**Отказоустойчивость**

Характерной чертой распределенных систем, которая отличает их от единичных машин, является возможность частичного отказа. Частичный отказ происходит при сбое в одном из компонентов распределенной системы. Этот отказ может нарушить нормальную работу некоторых компонентов, в то время как другие компоненты это никак не затронет. В противоположность отказу в распределенной системе отказ в нераспределенной системе всегда является глобальным, в том смысле, что он затрагивает все ее компоненты и легко может привести к неработоспособности всего приложения.

При создании распределенной системы очень важно добиться, чтобы она могла автоматически восстанавливаться после частичных отказов, незначительно снижая при этом общую производительность. В частности, когда бы ни случился отказ, распределенная система в процессе восстановления должна продолжать работать приемлемым образом, то есть быть устойчивой к отказам, сохраняя в случае отказов определенную степень функциональности.

Далее мы познакомимся со способами обеспечения отказоустойчивости распределенной системы, ограничившись изложением определенных базовых сведений об отказоустойчивости. Под отказоустойчивостью процессов мы понимаем методы, при помощи которых отказ одного или более процессов проходит для остальной части системы почти незаметно. С этим вопросом связана проблема надежной групповой рассылки, при которой передача сообщений набору процессов производится с гарантией доставки. Надежная групповая рассылка часто необходима для поддержания синхронности процессов.

Атомарность — это свойство, важное для многих приложений. Так, например, в распределенных транзакциях необходимо гарантировать, что все операции, входящие в транзакцию, либо происходят, либо нет. Фундаментальным для атомарности в распределенных системах является понятие распределенных протоколов подтверждения.

И, наконец, важным вопросом является то, как восстанавливать систему после отказов. В частности, когда и как следует сохранять состояние распределенной системы на тот случай, если позже это состояние потребуется восстанавливать.

Исследованию отказоустойчивости посвящено большое количество трудов. Этот раздел мы посвятим рассмотрению базовых концепций обработки отказов, а далее обсудим модели отказов. Основа всех методик ликвидации последствий отказов — избыточность, о ней мы тоже поговорим.

Основные концепции

Чтобы понять роль отказоустойчивости в распределенных системах, сначала необходимо выяснить, что для распределенных систем означает «быть отказоустойчивыми». Отказоустойчивость тесно связана с понятием надежных систем (dependable systems). Надежность — это термин, охватывающий множество важных требований к распределенным системам, включая:

- доступность (availability);

- безотказность (reliability);

- безопасность (safety);

- ремонтопригодность (maintainability).

Доступность — это свойство системы находиться в состоянии готовности к работе. Обычно доступность показывает вероятность того, что система в данный момент времени будет правильно работать и окажется в состоянии выполнить свои функции, если пользователи того потребуют. Другими словами, система с высокой степенью доступности — это такая система, которая в произвольный момент времени, скорее всего, находится в работоспособном состоянии.

Под безотказностью имеется в виду свойство системы работать без отказов в течение продолжительного времени. В противоположность доступности безотказность определяется в понятиях временного интервала, а не момента времени. Система с высокой безотказностью — это система, которая, скорее всего, будет непрерывно работать в течение относительно долгого времени. Между безотказностью и доступностью имеется небольшая, но существенная разница. Если система отказывает на одну миллисекунду каждый час, она имеет доступность порядка 99,9999 %, но крайне низкую безотказность. С другой стороны, система, которая никогда не отказывает, но каждый август отключается на две недели, имеет высокую безотказность, но ее доступность составляет всего 96 %. Эти две характеристики — не одно и то же.

Безопасность определяет, насколько катастрофична ситуация временной неспособности системы должным образом выполнять свою работу. Так, многие системы управления процессами, используемые, например, на атомных электростанциях или космических кораблях, должны обладать высокой степенью безопасности. Если эти управляющие системы даже временно, на короткий срок, перестанут работать, результат может быть ужасен. Множество примеров происходивших в прошлом событий показывают, как тяжело построить безопасную систему (и может быть еще больше таких примеров ожидают нас в будущем).

И, наконец, ремонтопригодность определяет, насколько сложно исправить неполадки в описываемой системе. Системы с высокой ремонтопригодностью могут также обладать высокой степенью доступности, особенно при наличии средств автоматического обнаружения и исправления неполадок. Однако, как мы увидим позже, говорить об автоматическом исправлении неполадок гораздо проще, чем создавать способные на это системы.

Часто надежные системы требуют также повышенного уровня защиты, особенно когда дело доходит до такого вопроса, как непротиворечивость. Мы поговорим о защите в следующей главе.

Говорят, что система отказывает (fail), если она не в состоянии выполнять свою работу. В частности, если распределенная система создавалась для предоставления пользователям некоторых услуг, то система будет считаться находящейся в состоянии отказа в том случае, если она не сможет предоставлять все или некоторые услуги.Ошибкой (error) называется такое состояние системы, которое может привести к ее неработоспособности. Так, например, при передаче пакетов по сети может случиться, что некоторые пакеты, пришедшие к получателю, окажутся поврежденными. Повреждения в данном случае будут означать, что получатель может неверно прочесть значения битов (например, 1 вместо 0) или оказаться не в состоянии определить сам факт прихода пакета.

Причиной ошибки является отказ (fault). Понятно, что найти причину ошибки очень важно. Так, например, вызвать повреждение пакетов вполне может неисправная или некачественная среда передачи. В этом случае устранить отказ относительно легко. Однако ошибки передачи в беспроводных сетях могут быть вызваны, например, плохой погодой. Изменение погоды с целью предупредить возникновение ошибок нам пока не под силу.

Построение надежных систем тесно связано с управлением отказами. Управление в данном случае означает нечто среднее между предотвращением, исправлением и предсказанием отказов. Для нашей цели наиболее важным вопросом является отказоустойчивость (fault tolerance), под которой мы будем понимать способность системы предоставлять услуги даже при наличии отказов. Отказы обычно подразделяются на проходные, перемежающиеся и постоянные.

Проходные отказы (transient faults) происходят однократно и больше не повторяются. Если повторить операцию, они не возникают. Птица, пролетевшая через луч микроволнового передатчика, в некоторых сетях может привести к потере битов. Если передатчик, выждав положенную паузу, повторит отправку, то по всей вероятности передача пройдет как положено.

Перемежающиеся отказы (intermittent faults) появляются и пропадают, «когда захотят», а потом появляются снова и т. д. Перемежающиеся отказы нередко бывают вызваны потерей контакта в разъеме. Из-за трудностей в диагностике перемежающиеся отказы часто вызывают сильное раздражение. Обычно когда приходит ремонтник, система работает просто прекрасно.

Постоянные отказы (permanent faults) — это отказы, которые продолжают свое существование до тех пор, пока отказавший компонент не будет заменен. Примерами постоянных отказов могут быть сгоревшие микросхемы, ошибки в программном обеспечении или сместившиеся головки дисков.

**Модели отказов**

Отказавшая система не в состоянии корректно выполнять ту работу, для которой она была создана. Если рассматривать распределенную систему как набор серверов, взаимодействующих друг с другом и с клиентами, то некорректное выполнение работы будет означать, что серверы или коммуникационные каналы, а возможно и оба этих компонента, не в состоянии делать то, для чего они, как предполагалось, предназначены. Однако сами по себе нефункционирующие серверы не всегда приводят к отказу системы, как мы его понимаем. Если сервер для корректной работы нуждается в услугах других серверов, причину ошибки, может быть, следует искать в другом месте.

Таких зависимостей в распределенных системах великое множество. Отказавший диск осложняет работу файлового сервера, который разрабатывался для реализации файловой системы с высокой степенью доступности. Если этот файловый сервер является частью распределенной базы данных, под угрозой находится работа всей базы, поскольку доступной оказывается только часть данных. Чтобы лучше понимать, насколько серьезен на самом деле конкретный отказ, были разработаны различные схемы классификации. Одна из таких схем, приведена в табл. 7.1.

Таблица 7.1.

|  |  |
| --- | --- |
| Тип отказа | Описание |
| Поломка  Пропуск данных  Пропуск приема  Пропуск передачи  Ошибка синхронизации  Ошибка отклика  Ошибка значения  Ошибка передачи состояния  Произвольная ошибка | Сервер перестал работать, хотя до момента отказа работал правильно  Сервер неправильно реагирует на входящие запросы  Сервер неправильно принимает входящие запросы  Сервер неправильно отправляет сообщения  Реакция сервера происходит не в определенный интервал времени  Отклик сервера неверен  Сервер возвращает неправильное значение  Сервер отклоняется от верного потока управления  Сервер отправляет случайные сообщения в случайные моменты времени |

Поломка (crash failure) имеет место при внезапной остановке сервера, при этом до момента остановки он работает нормально. Важная особенность поломки состоит в том, что после остановки сервера никаких признаков его работы не наблюдается. Типичный пример поломки — полное зависание операционной системы, когда единственным решением проблемыявляется перезагрузка. Многие операционные системы персональных компьютеров претерпевают поломки настолько часто, что люди уже начинают полагать, что для них это обычное дело. В этом смысле перенос кнопкиResetс задней части корпуса компьютера на переднюю панель был, несомненно, оправдан.

Пропуск данных (omission failure) возникает в том случае, когда сервер неправильно реагирует на запросы. Эту ошибку могут вызывать различные причины. В случаепропуска приема (receive omission) сервер может, например, не получать запросов. Отметим, что такая ошибка может произойти, в частности, и в том случае, когда соединение между клиентом и сервером установлено совершенно правильным образом, но на сервере не запущен процесс для приема приходящих запросов. Пропуск приема обычно не влияет на текущее состояние сервера, но сервер остается в неведении о посланных ему сообщениях.

Похожая ошибка — пропуск передачи (send omission) — происходит, когда сервер выполняет свою работу, но по каким-либо причинам не в состоянии послать ответ. Подобная ошибка может произойти, например, при переполнении буфера передачи, если сервер не готов к подобной ситуации. Отметим, что в противоположность пропуску приема в данном случае сервер может перейти в состояние, соответствующее полному выполнению услуги для клиента. Впоследствии, если обнаружится, что имел место пропуск передачи, сервер, вероятно, должен быть готов к тому, что клиент повторно пошлет свой последний запрос. Другие типы пропусков не имеют отношения к взаимодействию и могут быть вызваны ошибками в программе, такими как бесконечные циклы или некорректная работа с памятью, которые способны «подвесить» сервер.

Следующий класс ошибок связан с синхронизацией. Ошибки синхронизации (timing failures) возникают при ожидании ответа дольше определенного временного интервала. Как мы говорили во время обсуждения изохронных потоков данных, слишком раннее предоставление данных легко может вызвать у принимающей стороны проблемы, связанные с отсутствием места в буфере для хранения получаемых данных. Чаще, однако, сервер отвечает слишком поздно, в этом случае говорят, что произошлаошибка производительности (performance failure).

Еще один важный тип ошибок — ошибки отклика (response failures), при которых ответы сервера просто неверны. Существует два типа ошибок отклика. В случаеошибки значения (value failure) сервер дает неверный ответ на запрос. Так, например, эту ошибку демонстрирует поисковая машина, систематически возвращающая адреса web-страниц, не связанных с запросом пользователя.

Другой тип ошибок отклика — ошибки передачи состояния (state transition failures). Этот тип ошибок характеризуется реакцией на запрос, не соответствующей ожиданиям. Так, например, если сервер получает сообщение, которое он не в состоянии распознать, и никаких мер по обработке подобных сообщений не предусмотрено, возникает ошибка передачи состояния. В частности, сервер может неправомерно осуществить по умолчанию некие действия, производить которые в данном случае не следовало бы.

Весьма серьезны произвольные ошибки (arbitrary failures), известные также под названиемвизантийских ошибок (Byzantine failures). Когда случается произвольная ошибка, клиент должен приготовиться к самому худшему. Например, может оказаться, что сервер генерирует сообщения, которые он в принципе не должен генерировать, но система не опознает их как некорректные. Хуже того, неправильно функционирующий сервер может, участвуя в работе группы серверов, приводить к появлению заведомо неверных ответов. Эта ситуация показывает, почему для надежных систем очень важна защита. Термин «византийские» восходит к Византийской империи (Балканы и современная Турция) времен 330-1453 годов, когда бесконечные заговоры, интриги и ложь считались в правящих кругах обычным делом. Ниже мы вернемся к разговору об ошибках такого рода.

Произвольные ошибки похожи на поломки. Поломка — наиболее распространенная причина остановки сервера. Поломки известны также под названием ошибок аварийной остановки (fail-stop failures). В действительности аварийно остановленный сервер просто прекращает генерировать исходящие сообщения. По этому признаку его остановка обнаруживается другими процессами. Например, по настоящему дружественный сервер может предупредить нас о том, что находится на грани поломки.

Разумеется, в реальной жизни серверы, останавливаясь по причине пропуска данных или поломок, не настолько дружественны, чтобы оповестить нас о надвигающейся остановке. Другие процессы должны сами обнаружить «безвременную кончину» сервера. Однако в подобных системах остановки без уведомления (fail-silent systems) другие процессы могут сделать неверный вывод об остановке сервера. Сервер может просто медленно работать, то есть может иметь место ошибка производительности.

И, наконец, возможно, что сервер производит случайные сообщения, которые другие процессы считают абсолютным мусором. В этом случае мы имеем дело с наиболее простым случаем произвольной ошибки. Подобные ошибки называют безопасными (fail-safe).

**Маскирование ошибок при помощи избыточности**

Если система считается отказоустойчивой, она должна пытаться маскировать факты ошибок от других процессов. Основной метод маскирования ошибок — использование избыточности (redundancy). Возможно применение трех типов избыточности — информационной избыточности, временной избыточности и физической избыточности. В случае информационной избыточности к сообщению добавляются дополнительные биты, по которым можно произвести исправление сбойных битов. Так, например, можно добавить к передаваемым данным код Хемминга для восстановления сигнала в случае зашумленного канала передачи.

При временной избыточности уже выполненное действие при необходимости осуществляется еще раз. В качестве примера к этому способу рассмотрим транзакции. Если транзакция была прервана, ее можно без каких-либо опасений повторить. Временная избыточность особенно полезна, если мы имеем дело с проходным или перемежающимся отказом.

В случае физической избыточности мы добавляем в систему дополнительное оборудование или процессы, которые делают возможной работу системы при утрате или неработоспособности некоторых компонентов. Физическая избыточность, таким образом, может быть как аппаратной, так и программной. Так, например, можно добавить к системе дополнительные процессы, так что при крахе некоторых из них система продолжит функционировать правильно. Другими словами, посредством репликации достигается высокая степень отказоустойчивости. Ниже мы вернемся к этому типу программной избыточности.

Физическая избыточность — это широко распространенный способ добиться отказоустойчивости. Она используется в биологии (млекопитающие имеют по два глаза, по два уха, по два легких и т. д.), самолетостроении (у Боинга-747 четыре двигателя, но он может лететь и на трех) и спорте (несколько судей на тот случай, если один чего-нибудь не заметит). Она также много лет используется для обеспечения отказоустойчивости в радиосхемах. Рассмотрим, например, схему, показанную на рис. 7.1, а. На ней сигнал проходит последовательно через устройства А, В и С. Если одно из них неисправно, результат, вероятно, будет неверен.

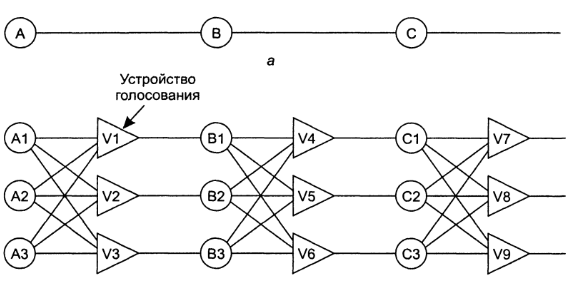


Рис 7.1

На рис. 7.1, б каждое из устройств присутствует в трех экземплярах. Переход к следующему участку схемы определяется тройным голосованием. Устройство голосования — это схема с тремя входами и одним выходом. Если два или три входных сигнала совпадают, выходной сигнал равен входному. Если все три входных сигнала различны, выходной сигнал не определен. Такая схема известна под названием тройного модульного резервирования (Triple Modular Redundancy, TMR).

Допустим, элемент А2 отказал. Каждое из устройств голосования,V1, V2 иV3, получает два правильных (идентичных) входных сигнала и один неправильный, и каждое из них передает на второй участок цепи правильное значение. В результате эффект отказаА2 оказывается полностью замаскированным, а значит, входные сигналы элементовВ1, В2 и В3 абсолютно такие же, как если бы никакого отказа не было.

Рассмотрим теперь, что будет, если в придачу к А2 откажут также элементыВ3 иС1. Эффект их отказа также будет замаскирован, и все три выходных сигнала окажутся правильными.

Прежде всего, непонятно, зачем на каждом этапе нужно использовать три устройства голосования. Вообще-то определить и донести до нас мнение большинства может и одно устройство голосования. Однако устройство голосования — это тоже компонент, который тоже может отказать. Рассмотрим, например, отказ V1 Входящий сигналВ1 в этом случае будет неверным, но до тех пор, пока все остальное работает,В2 иВЗ будут давать одинаковые выходные сигналы, и устройстваV4, V5 иV6 образуют правильный результат для третьего этапа. ОтказV1 практически ничем не будет отличаться от отказаВ1. В обоих случаях выходной сигналот В1 будет неверным, и в любом случае при голосовании он окажется в меньшинстве.

Хотя не все отказоустойчивые распределенные системы используют TMR, эта технология является очень распространенной и помогает яснее понять, что такое отказоустойчивая система и чем она отличается от системы, составленной из высоконадежных компонентов, но не обладающей отказоустойчивой структурой. Разумеется, TMR можно применять и рекурсивно. Например, можно повышать надежность микросхем, встраивая в них механизмы TMR. Для разработчиков, использующих микросхемы, это останется неизвестным.