13. Лекция: Обеспечение высокой доступности

Доступность

Основные понятия

Информационная система предоставляет своим пользователям определенный набор услуг (сервисов). Говорят, что обеспечен нужный уровень доступности этих сервисов, если следующие показатели находятся в заданных пределах:

- **Эффективность услуг**. Эффективность услуги определяется в терминах максимального времени обслуживания запроса, количества поддерживаемых пользователей и т.п. Требуется, чтобы эффективность не опускалась ниже заранее установленного порога.
- *Время недоступности*. Если эффективность информационной услуги не удовлетворяет наложенным ограничениям, услуга считается недоступной. Требуется, чтобы максимальная продолжительность периода недоступности и суммарное *время недоступности* за некоторой период (месяц, год) не превышали заранее заданных пределов.

В сущности, требуется, чтобы информационная система почти всегда работала с нужной эффективностью. Для некоторых критически важных систем (например, систем управления) *время недоступности* должно быть нулевым, без всяких "почти". В таком случае говорят о вероятности возникновения ситуации недоступности и требуют, чтобы эта вероятность не превышала заданной величины. Для решения данной задачи создавались и создаются специальные ***отказоустойчивые системы***, стоимость которых, как правило, весьма высока.

К подавляющему большинству коммерческих систем предъявляются менее жесткие требования, однако современная деловая жизнь и здесь накладывает достаточно суровые ограничения, когда число обслуживаемых пользователей может измеряться тысячами, время ответа не должно превышать нескольких секунд, а *время недоступности* – нескольких часов в год.

Задачу обеспечения *высокой доступности* необходимо решать для современных конфигураций, построенных в технологии клиент/сервер. Это означает, что в защите нуждается вся цепочка – от пользователей (возможно, удаленных) до критически важных серверов (в том числе серверов безопасности).

Основные угрозы доступности были рассмотрены нами ранее.

В соответствии с ГОСТ 27.002, под ***отказом*** понимается событие, которое заключается в нарушении работоспособности изделия. В контексте данной работы изделие – это информационная система или ее компонент.

В простейшем случае можно считать, что *отказы* любого компонента составного изделия ведут к общему *отказу*, а распределение *отказов* во времени представляет собой простой пуассоновский поток событий. В таком случае вводят понятие ***интенсивности отказов*** и ***среднего времени наработки на отказ***, которые связаны между собой соотношением

$$T_i = \frac{1}{\lambda_i}$$

Рис. 13.1.

где **i** – номер компонента,

λ_i – *интенсивность отказов*,

T_i – *среднее время наработки на отказ*.

Интенсивности отказов независимых компонентов складываются:

$$\lambda = \lambda_1 + \ldots + \lambda_n$$

Рис. 13.2.

а *среднее время наработки на отказ* для составного изделия задается соотношением

$$T = \frac{1}{\lambda}$$

Рис. 13.3.

Уже эти простейшие выкладки показывают, что если существует компонент, *интенсивность отказов* которого много больше, чем у остальных, то именно он определяет *среднее время наработки на отказ* всей информационной системы. Это является теоретическим обоснованием принципа первоочередного укрепления самого *слабого звена*.

Пуассоновская модель позволяет обосновать еще одно очень важное положение, состоящее в том, что эмпирический подход к построению систем *высокой доступности* не может быть реализован за приемлемое время. При традиционном цикле тестирования/отладки программной системы по оптимистическим оценкам каждое исправление ошибки приводит к экспоненциальному убыванию (примерно на половину десятичного порядка) *интенсивности отказов*. Отсюда следует, что для того, чтобы на опыте убедиться в достижении необходимого уровня доступности, независимо от применяемой технологии тестирования и отладки, придется потратить время, практически равное *среднему времени наработки на отказ*. Например, для достижения *среднего времени наработки на отказ* 10^5 часов потребуется более $10^{4,5}$ часов, что составляет более трех лет. Значит, нужны иные методы построения систем *высокой доступности*, методы, эффективность которых доказана аналитически или практически за более чем пятьдесят лет развития вычислительной техники и программирования.

Пуассоновская модель применима в тех случаях, когда информационная система содержит одиночные точки *отказа*, то есть компоненты, выход которых из строя ведет к *отказу* всей системы. Для исследования систем с резервированием применяется иной формализм.

В соответствии с постановкой задачи будем считать, что существует количественная мера эффективности предоставляемых изделием информационных услуг. В таком случае вводятся понятия ***показателей эффективности*** отдельных элементов и эффективности функционирования всей сложной системы.

В качестве меры доступности можно принять вероятность приемлемости эффективности услуг, предоставляемых информационной системой, на всем протяжении рассматриваемого отрезка времени. Чем большим запасом эффективности располагает система, тем выше ее доступность.

При наличии *избыточности* в конфигурации системы вероятность того, что в рассматриваемый промежуток времени *эффективность информационных сервисов* не опустится ниже допустимого предела, зависит не только от вероятности *отказа* компонентов, но и от времени, в течение которого они остаются неработоспособными, поскольку при этом суммарная эффективность падает, и каждый следующий *отказ* может стать фатальным. Чтобы максимально увеличить доступность системы, необходимо минимизировать время неработоспособности каждого компонента. Кроме того, следует учитывать, что, вообще говоря, ремонтные работы могут потребовать понижения эффективности или даже временного отключения работоспособных компонентов; такого рода влияние также необходимо минимизировать.

Несколько терминологических замечаний. Обычно в литературе по теории надежности вместо доступности говорят о ***готовности*** (в том числе о высокой *готовности*). Мы предпочли термин "доступность", чтобы подчеркнуть, что *информационный сервис* должен быть не просто "готов" сам по себе, но доступен для своих пользователей в условиях, когда ситуации недоступности могут вызываться причинами, на первый взгляд не имеющими прямого отношения к сервису (пример – отсутствие консультационного обслуживания).

Далее, вместо *времени недоступности* обычно говорят о ***коэффициенте готовности***. Нам хотелось обратить внимание на два показателя – длительность однократного простоя и суммарную продолжительность простоев, поэтому мы предпочли термин "*время недоступности*" как более емкий.

Основы мер обеспечения высокой доступности

Основой мер повышения доступности является применение структурированного подхода, нашедшего воплощение в объектно-ориентированной методологии. *Структуризация* необходима по отношению ко всем аспектам и составным частям информационной системы – от архитектуры до административных баз данных, на всех этапах ее жизненного цикла – от инициации до выведения из эксплуатации. *Структуризация*, важная сама по себе, является одновременно необходимым условием практической реализуемости прочих мер повышения доступности. Только маленькие системы можно строить и эксплуатировать как угодно. У больших систем свои законы, которые, как мы уже указывали, программисты впервые осознали более 30 лет назад.

При разработке мер обеспечения *высокой доступности информационных сервисов* рекомендуется руководствоваться следующими архитектурными принципами, рассматривавшимися ранее:

- апробированность всех процессов и составных частей информационной системы;
- унификация процессов и составных частей;
- управляемость процессов, контроль состояния частей;
- автоматизация процессов;
- модульность архитектуры;
- ориентация на простоту решений.

Доступность системы в общем случае достигается за счет применения трех групп мер, направленных на повышение:

- ***безотказности*** (под этим понимается минимизация вероятности возникновения какого-либо *отказа*; это элемент *пассивной безопасности*, который дальше рассматриваться не будет);
- ***отказоустойчивости*** (способности к *нейтрализации отказов, "живучести"*, то есть способности сохранять требуемую эффективность, несмотря на *отказы* отдельных компонентов);
- ***обслуживаемости*** (под *обслуживаемостью* понимается минимизация времени простоя отказавших компонентов, а также отрицательного влияния ремонтных работ на *эффективность информационных сервисов*, то есть быстрое и *безопасное восстановление* после *отказов*).

Главное при разработке и реализации мер обеспечения *высокой доступности* – *полнота* и *систематичность*. В этой связи представляется целесообразным составить (и поддерживать в актуальном состоянии) ***карту информационной системы*** организации (на что мы уже обращали внимание), в которой фигурировали бы все объекты ИС, их состояние, связи между ними, процессы, ассоциируемые с объектами и связями. С помощью подобной *карты* удобно формулировать намечаемые меры, контролировать их исполнение, анализировать состояние ИС.

Отказоустойчивость и зона риска

Информационную систему можно представить в виде графа сервисов, ребра в котором соответствуют отношению "сервис А непосредственно использует сервис В".

Пусть в результате осуществления некоторой атаки (источником которой может быть как человек, так и явление природы) выводится из строя подмножество сервисов S1 (то есть эти сервисы в результате нанесенных повреждений становятся неработоспособными). Назовем S1 **зоной поражения.**

В **зону риска** S мы будем включать все сервисы, эффективность которых при осуществлении атаки падает ниже допустимого предела. Очевидно, S1 – подмножество S.

S строго включает S1, когда имеются сервисы, непосредственно не затронутые атакой, но критически зависящие от пораженных, то есть неспособные переключиться на использование эквивалентных услуг либо в силу отсутствия таковых, либо в силу невозможности доступа к ним. Например, зона поражения может сводиться к одному порту концентратора, обслуживающему критичный сервер, а зона риска охватывает все рабочие места пользователей сервера.

Чтобы система не содержала **одиночных точек** *отказа*, то есть оставалась "живучей" при реализации любой из рассматриваемых угроз, ни одна зона риска не должна включать в себя предоставляемые услуги. *Нейтрализацию отказов* нужно выполнять внутри системы, незаметно для пользователей, за счет размещения достаточного количества избыточных ресурсов.

С другой стороны, естественно соизмерять усилия по обеспечению "*живучести*" с рассматриваемыми угрозами. Когда рассматривается набор угроз, соответствующие им зоны поражения могут оказаться вложенными, так что "*живучесть*" по отношению к более серьезной угрозе автоматически влечет за собой и "*живучесть*" в более легких случаях. Следует учитывать, однако, что обычно стоимость переключения на резервные ресурсы растет вместе с увеличением объема этих ресурсов. Значит, для наиболее вероятных угроз целесообразно минимизировать зону риска, даже если предусмотрена нейтрализация объемлющей угрозы. Нет смысла переключаться на резервный вычислительный центр только потому, что у одного из серверов вышел из строя блок питания.

Зону риска можно трактовать не только как совокупность ресурсов, но и как часть пространства, затрагиваемую при реализации угрозы. В таком случае, как правило, чем больше расстояние дублирующего ресурса от границ зоны риска, тем выше стоимость его поддержания, поскольку увеличивается протяженность линий связи, время переброски персонала и т.п. Это еще один довод в пользу адекватного противодействия угрозам, который следует принимать во внимание при размещении избыточных ресурсов и, в частности, при организации резервных центров.

Введем еще одно понятие. Назовем **зоной нейтрализации** угрозы совокупность ресурсов, вовлеченных в *нейтрализацию отказа*, возникшего вследствие реализации угрозы. Имеются в виду ресурсы, режим работы которых в случае *отказа* изменяется. Очевидно, зона риска является подмножеством зоны нейтрализации. Чем меньше разность между ними, тем экономичнее данный механизм нейтрализации.

Все, что находится вне зоны нейтрализации, *отказа* "не чувствует" и может трактовать внутренность этой зоны как безотказную. Таким образом, в иерархически организованной системе грань между "*живучестью*" и *обслуживаемостью*, с одной стороны, и *безотказностью*, с другой стороны, относительна. Целесообразно конструировать целостную информационную систему из компонентов, которые на верхнем уровне можно считать безотказными, а вопросы "*живучести*" и *обслуживаемости* решать в пределах каждого компонента.

Обеспечение отказоустойчивости

Основным средством повышения "*живучести*" является внесение *избыточности* в конфигурацию аппаратных и программных средств, поддерживающей инфраструктуры и персонала, **резервирование** технических средств и **тиражирование** информационных ресурсов (программ и данных).

Меры по обеспечению *отказоустойчивости* можно разделить на **локальные** и **распределенные**. Локальные меры направлены на достижение "*живучести*" отдельных компьютерных систем или их аппаратных и программных компонентов (в первую очередь с целью нейтрализации внутренних *отказов* ИС). Типичные примеры подобных мер – использование кластерных конфигураций в качестве платформы критичных серверов или "горячее" резервирование активного сетевого оборудования с автоматическим переключением на резерв.

Если в число рассматриваемых рисков входят серьезные аварии поддерживающей инфраструктуры, приводящие к выходу из строя производственной площадки организации, следует предусмотреть распределенные меры обеспечения *живучести*, такие как создание или аренда резервного вычислительного центра. При этом, помимо дублирования и/или тиражирования ресурсов, необходимо предусмотреть средства автоматического или быстрого ручного переконфигурирования компонентов ИС, чтобы обеспечить переключение с основной площадки на резервную.

Аппаратура – относительно статичная составляющая, однако было бы ошибкой полностью отказывать ей в динамичности. В большинстве организаций информационные системы находятся в постоянном развитии, поэтому на протяжении всего жизненного цикла ИС следует соотносить все изменения с необходимостью обеспечения "*живучести*", не забывать "тиражировать" новые и модифицированные компоненты.

Программы и данные более динамичны, чем аппаратура, и резервироваться они могут постоянно, при каждом изменении, после завершения некоторой логически замкнутой группы изменений или по истечении определенного времени.

Резервирование программ и данных может выполняться многими способами – за счет зеркалирования дисков, резервного копирования и восстановления, репликации баз данных и т.п. Будем использовать для всех перечисленных способов термин "тиражирование".

Выделим следующие классы тиражирования:

- **Симметричное/асимметричное**. Тиражирование называется симметричным, если все серверы, предоставляющие данный сервис, могут изменять принадлежащую им информацию и передавать изменения другим серверам. В противном случае тиражирование называется асимметричным.
- **Синхронное/асинхронное**. Тиражирование называется синхронным, если изменение передается всем экземплярам сервиса в рамках одной распределенной транзакции. В противном случае тиражирование называется асинхронным.
- **Осуществляемое средствами сервиса**, хранящего информацию/**внешними средствами.**

Рассмотрим, какие способы тиражирования предпочтительнее.

Безусловно, следует предпочесть стандартные средства тиражирования, встроенные в сервис.

Асимметричное тиражирование теоретически проще симметричного, поэтому целесообразно выбрать асимметрию.

Труднее всего выбрать между синхронным и асинхронным тиражированием. Синхронное идейно проще, но его реализация может быть тяжеловесной и сложной, хотя это внутренняя сложность сервиса, невидимая для приложений. Асинхронное тиражирование устойчивее к *отказам* в сети, оно меньше влияет на работу основного сервиса.

Чем надежнее связь между серверами, вовлеченными в процесс тиражирования, чем меньше время, отводимое на переключение с основного сервера на резервный, чем жестче требования к актуальности информации, тем более предпочтительным оказывается синхронное тиражирование.

С другой стороны, недостатки асинхронного тиражирования могут компенсироваться процедурными и программными мерами, направленными на контроль целостности информации в распределенной ИС. Сервисы, входящие в состав ИС, в состоянии обеспечить ведение и хранение журналов транзакций, с помощью которых можно выявлять операции, утерянные при переключении на резервный сервер. Даже в условиях неустойчивой связи с удаленными филиалами организации подобная проверка в фоновом режиме займет не более нескольких часов, поэтому асинхронное тиражирование может использоваться практически в любой ИС.

Асинхронное тиражирование может производиться на сервер, работающий в режиме "горячего" резерва, возможно, даже обслуживающего часть пользовательских запросов, или на сервер, работающий в режиме "теплого" резерва, когда изменения периодически "накатываются", но сам резервный сервер запросов не обслуживает.

Достоинство "теплого" резервирования в том, что его можно реализовать, оказывая меньшее влияние на основной сервер. Это влияние вообще может быть сведено к нулю, если асинхронное тиражирование осуществляется путем передачи инкрементальных копий с основного сервера (резервное копирование необходимо выполнять в любом случае).

Основной недостаток "теплого" резерва состоит в длительном времени включения, что может быть неприемлемо для "тяжелых" серверов, таких как кластерная конфигурация сервера СУБД. Здесь необходимо проводить измерения в условиях, близких к реальным.

Второй недостаток "теплого" резерва вытекает из опасности малых изменений. Может оказаться, что в самый нужный момент срочный перевод резерва в штатный режим невозможен.

Учитывая приведенные соображения, следует в первую очередь рассматривать возможность **"горячего" резервирования**, либо тщательно контролировать использование **"теплого" резерва** и регулярно (не реже одного раза в неделю) проводить пробные переключения резерва в "горячий" режим.

Программное обеспечение промежуточного слоя

С помощью **программного обеспечения промежуточного слоя (ПО ПС)** можно для произвольных прикладных сервисов добиться высокой "*живучести*" с полностью прозрачным для пользователей переключением на резервные мощности.

О возможностях и свойствах ПО промежуточного слоя можно прочитать в статье Ф. Бернстайна "Middleware: модель сервисов распределенной системы" (Jet Info, 1997, 11).

Перечислим основные достоинства ПО ПС, существенные для обеспечения *высокой доступности*.

- ПО ПС уменьшает сложность создания распределенных систем. Подобное ПО берет на себя часть функций, которые в локальном случае выполняют операционные системы;
- ПО ПС берет на себя **маршрутизацию запросов**, позволяя тем самым обеспечить "*живучесть*" прозрачным для пользователей образом;
- ПО ПС осуществляет **балансировку загрузки** вычислительных мощностей, что также способствует повышению доступности данных;
- ПО ПС в состоянии осуществлять **тиражирование** любой информации, а не только содержимого баз данных. Следовательно, любое приложение можно сделать устойчивым к *отказам* серверов;
- ПО ПС в состоянии **отслеживать состояние приложений** и при необходимости тиражировать и перезапускать программы, что гарантирует "*живучесть*" программных систем;
- ПО ПС дает возможность прозрачным для пользователей образом выполнять **переконфигурирование** (и, в частности, **наращивание**) серверных компонентов, что позволяет масштабировать систему, сохраняя инвестиции в прикладные системы. Стабильность прикладных систем – важный фактор повышения доступности данных.

Ранее мы упоминали о достоинствах использования ПО ПС в рамках межсетевых экранов, которые в таком случае становятся элементом обеспечения *отказоустойчивости* предоставляемых *информационных сервисов*.

Обеспечение обслуживаемости

Меры по обеспечению *обслуживаемости* направлены на снижение сроков диагностирования и устранения *отказов* и их последствий.

Для обеспечения *обслуживаемости* рекомендуется соблюдать следующие архитектурные принципы:

- ориентация на построение информационной системы из унифицированных компонентов с целью упрощения замены отказавших частей;
- ориентация на решения **модульной** структуры с возможностью **автоматического обнаружения** *отказов*, **динамического переконфигурирования** аппаратных и программных средств и **замены** *отказавших* **компонентов в "горячем" режиме**.

Динамическое переконфигурирование преследует две основные цели:

- **изоляция отказавших компонентов;**
- **сохранение работоспособности сервисов.**

Изолированные компоненты образуют зону поражения реализованной угрозы. Чем меньше соответствующая зона риска, тем выше *обслуживаемость* сервисов. Так, при *отказах* блоков питания, вентиляторов и/или дисков в современных серверах зона риска ограничивается отказавшим компонентом; при *отказах* процессорных модулей весь сервер может потребовать перезагрузки (что способно вызвать дальнейшее расширение зоны риска). Очевидно, в идеальном случае зоны поражения и риска совпадают, и современные серверы и активное сетевое оборудование, а также программное обеспечение ведущих производителей весьма близки к этому идеалу.

Возможность программирования реакции на *отказ* также повышает *обслуживаемость* систем. Каждая организация может выбрать свою стратегию реагирования на *отказы* тех или иных аппаратных и программных компонентов и автоматизировать эту реакцию. Так, в простейшем случае возможна отправка сообщения системному администратору, чтобы ускорить начало ремонтных работ; в более сложном случае может быть реализована процедура "мягкого" выключения (переключения) сервиса, чтобы упростить обслуживание.

Возможность удаленного выполнения административных действий – важное направление повышения *обслуживаемости*, поскольку при этом ускоряется начало восстановительных мероприятий, а в идеале все работы (обычно связанные с обслуживанием программных компонентов) выполняются в удаленном режиме, без перемещения квалифицированного персонала, то есть с высоким качеством и в кратчайшие сроки. Для современных систем возможность удаленного администрирования – стандартное свойство, но важно позаботиться о его практической реализуемости в условиях разнородности конфигураций (в первую очередь клиентских). Централизованное распространение и конфигурирование программного обеспечения, управление компонентами информационной системы и диагностирование – надежный фундамент технических мер повышения *обслуживаемости*.

Существенный аспект повышения *обслуживаемости* – организация консультационной службы для пользователей (**обслуживаемость пользователей**), внедрение программных систем для работы этой службы, обеспечение достаточной пропускной способности каналов связи с пользователями, в том числе в режиме пиковых нагрузок.