Национальный технический университет Украины
«Киевский политехнический институт»
Институт телекоммуникационных систем
Кафедра информационно-телекоммуникационных сетей

Информационные и программные ресурсы в ТСМ

Тема 6
ОТКАЗОУСТОЙЧИВОСТЬ

Тихоненко Ю. Ю.,
30.03.2010

Содержание

1. Основные определения
2. Модели отказов
3. Маскирование ошибок при помощи избыточности
4. Отказоустойчивость процессов
5. Надежная связь клиент-сервер

Содержание

6. Надежная групповая рассылка

7. Распределенное подтверждение

8. Восстановление

9. Критерии отказоустойчивости

10. Отказоустойчивые сервера

Выводы

Список использованной литературы

1. Основные определения

Отказоустойчивость относится к надежностным характеристикам системы. Надежность — это термин, охватывающий множество важных требований к распределенным системам, включая:

- *доступность* (availability) - это свойство системы находиться в состоянии готовности к работе;
- *безотказность (reliability)* - свойство системы работать без отказов в течение продолжительного времени;
- *безопасность* (safety) определяет, насколько катастрофична ситуация временной неспособности системы должным образом выполнять свою работу;
- *ремонтопригодность* (maintainability) - определяет, насколько сложно исправить неполадки в описываемой системе.

1. Основные определения

Система *отказывает (fail*), если она не в состоянии выполнять свою работу.

Ошибкой (error) называется такое состояние системы, которое может привести к ее неработоспособности.

Причиной ошибки является *отказ (fault).*

Управление отказами означает предотвращение, исправление и предсказание отказов.

Отказоустойчивость (fault tolerance) - способность системы предоставлять услуги даже при наличии отказов.

1. Основные определения

Отказы обычно подразделяются на проходные, перемежающиеся и постоянные.

Проходные отказы (transient faults) происходят однократно и больше не повторяются. Если повторить операцию, они не возникают.

Перемежающиеся отказы (intermittent faults) появляются и пропадают, а потом появляются снова и т. д. Перемежающиеся отказы нередко бывают вызваны потерей контакта в разъеме.

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 или ошибки в программном обеспечении.

2. Модели отказов

Различные типы отказов:

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

2. Модели отказов

Поломка (crash failure) имеет место при внезапной остановке сервера, при этом до момента остановки он работает нормально. Типичный пример поломки — полное зависание операционной системы.

Пропуск данных (omission failure) возникает в том случае, когда сервер неправильно реагирует на запросы.

В случае *пропуска приема (receive omission)* сервер не получает запросы. Пример: на сервере не запущен процесс для приема приходящих запросов.

Пропуск передачи (send omission) происходит, когда сервер выполняет свою работу, но по каким-либо причинам не в состоянии послать ответ. Пример: при переполнении буфера передачи.

2. Модели отказов

Ошибки синхронизации (timing failures) возникают при ожидании ответа дольше определенного временного интервала.

Ошибки отклика (response failures) - ответы сервера просто неверны. Существует два типа ошибок отклика. В случае *ошибки значения (value failure)* сервер дает неверный ответ на запрос. Другой тип ошибок отклика — *ошибки передачи состояния (state transition failures)* - характеризуются реакцией на запрос, не соответствующей ожиданиям.

Произвольные ошибки (arbitrary failures) - сервер генерирует сообщения, которые он не должен генерировать, но система не опознает их как некорректные.

3. 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).*

Три типа избыточности :

- информационная - к сообщению добавляются дополнительные биты, по которым можно произвести исправление ошибочных битов;
- **программная избыточность - сопоставление результатов обработки одинаковых исходных данных разными программами;**
- аппаратная избыточность - добавление в систему дополнительного оборудования, которые делают возможной работу системы при утрате или неработоспособности некоторых компонентов.

4. Отказоустойчивость процессов

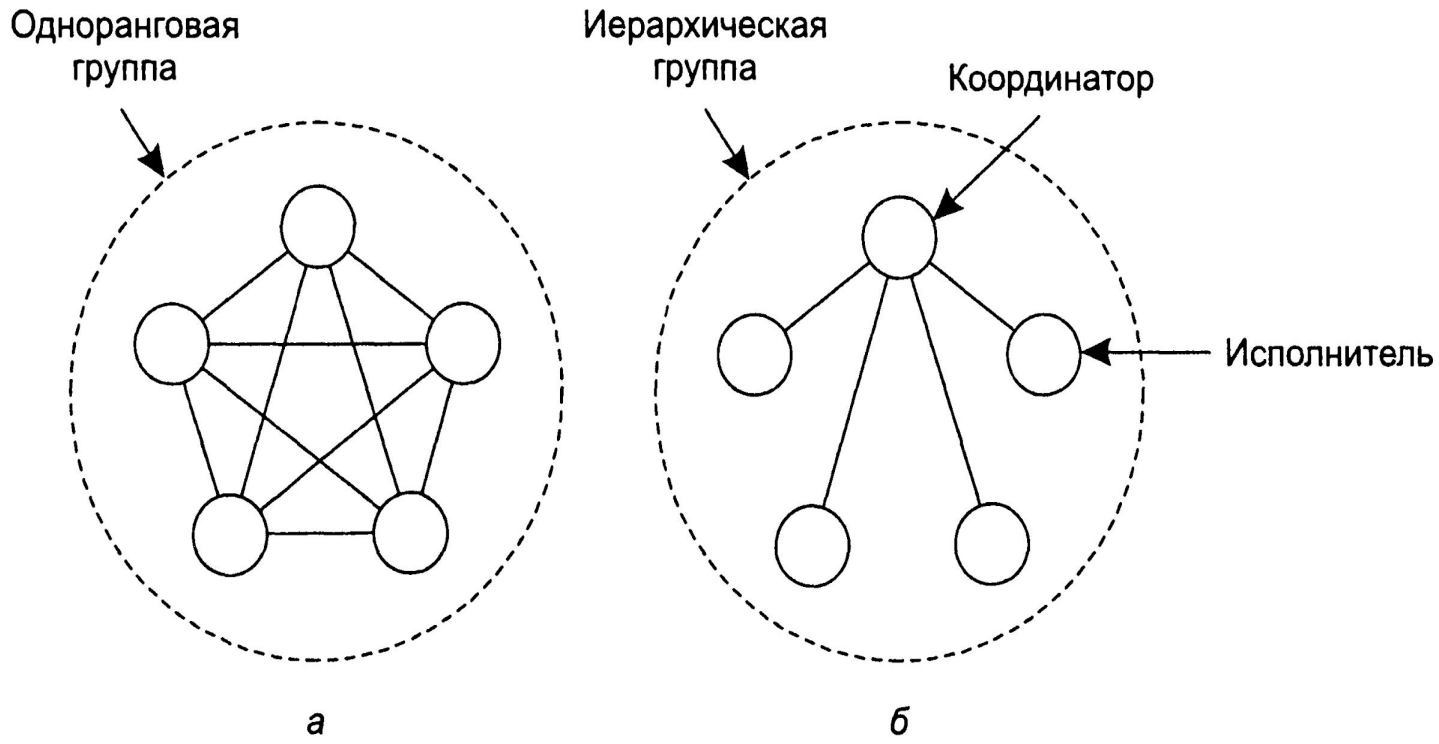
Отказоустойчивость процессов организуется путем репликации процессов в группах.

Основной подход к защите от последствий отказа процессов — объединить несколько идентичных процессов в группу. Основное свойство всех подобных групп состоит в том, что когда сообщение посылается группе, его получают все члены этой группы. Таким образом, если один из процессов группы перестает работать, можно надеяться на то, что его место займет другой.

Процесс может посылать сообщения группе серверов, не зная ничего о том, сколько их там и где они находятся, причем состав группы серверов при каждом вызове может быть разным.

4. Отказоустойчивость процессов

Взаимодействие в одноранговой и иерархической группах:



5. Надежная связь клиент-сервер

Во многих распределенных сетях надежная сквозная (point-to-point) передача реализуется путем использования надежного транспортного протокола, такого как TCP. TCP маскирует пропуски, проявляющиеся в виде потери сообщений, с помощью механизма подтверждений и повторных посылок (Positive Acknowledgment and Retransmission — PAR). Эти ошибки остаются абсолютно незамеченными клиентом TCP.

TCP работает по механизму скользящего окна. Размер окна показывает количество байт данных, которые отправитель может послать без ожидания подтверждения приема. Начальные размеры окон определяются при настройке соединения, но могут изменяться при передаче данных для управления потоком.

5. Надежная связь клиент-сервер

Назначение RPC — скрыть сам факт взаимодействия путем вызовов удаленных процедур, которые выглядят так же, как локальные вызовы.

Проблемы возникают, когда начинаются ошибки. Причина проблем кроется в том, что при наличии ошибок скрыть разницу между локальными и удаленными вызовами гораздо сложнее.

Ошибки, которые могут возникнуть в системах RPC:

- клиент не в состоянии обнаружить сервер;
- потеря сообщения с запросом от клиента к серверу;
- поломка сервера после получения запроса;
- потеря ответного сообщения от сервера к клиенту;
- поломка клиента после получения ответа.

5. Надежная связь клиент-сервер

1. **Клиент не в состоянии обнаружить сервер**

Решение: заставить ошибку возбуждать *исключение (exception).* Для этой цели можно использовать обработчики сигналов, т.е. можно определить новый тип сигнала и потребовать его обработки наравне с любыми другими сигналами.

2. Потеря сообщения с запросом

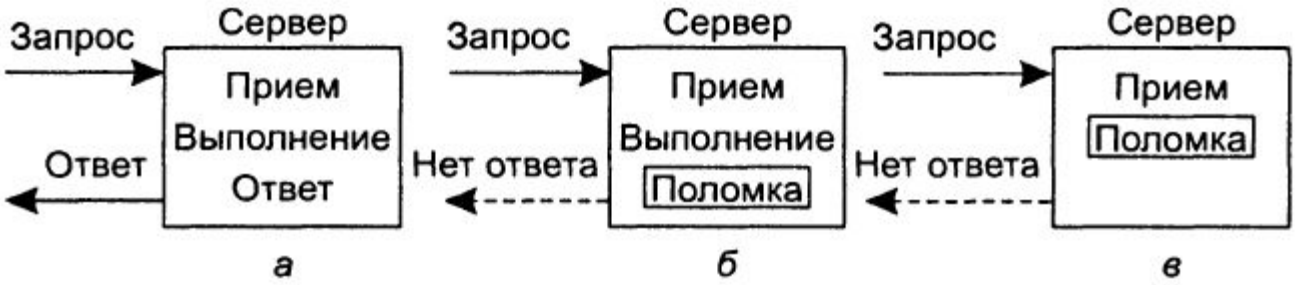
Решение: таймер. Если таймер переполнится, а ответ или подтверждение так и не будет получен, сообщение посылается повторно.

3. Потеря ответного сообщения

Решение: таймер, установленный операционной системой клиента. Если за определенное время не было получено ответа, необходимо просто послать запрос еще раз.

5. Надежная связь клиент-сервер

4. Поломка сервера



Решение: В случае (б) система должна передать клиенту сообщение об ошибке (например, возбудить исключение), в то время как в случае (в) она может просто послать запрос повторно.

Варианты:

- повторять попытки до тех пор, пока сервер не выдаст ответ, который дойдет до клиента;
- повторять попытки до тех пор, пока клиент не получит подтверждения доставки запроса на сервер;
- повторить запрос в случае подтверждения о доставке запроса;
- отказаться от дальнейших попыток и вернуть сообщение об ошибке.

5. Надежная связь клиент-сервер

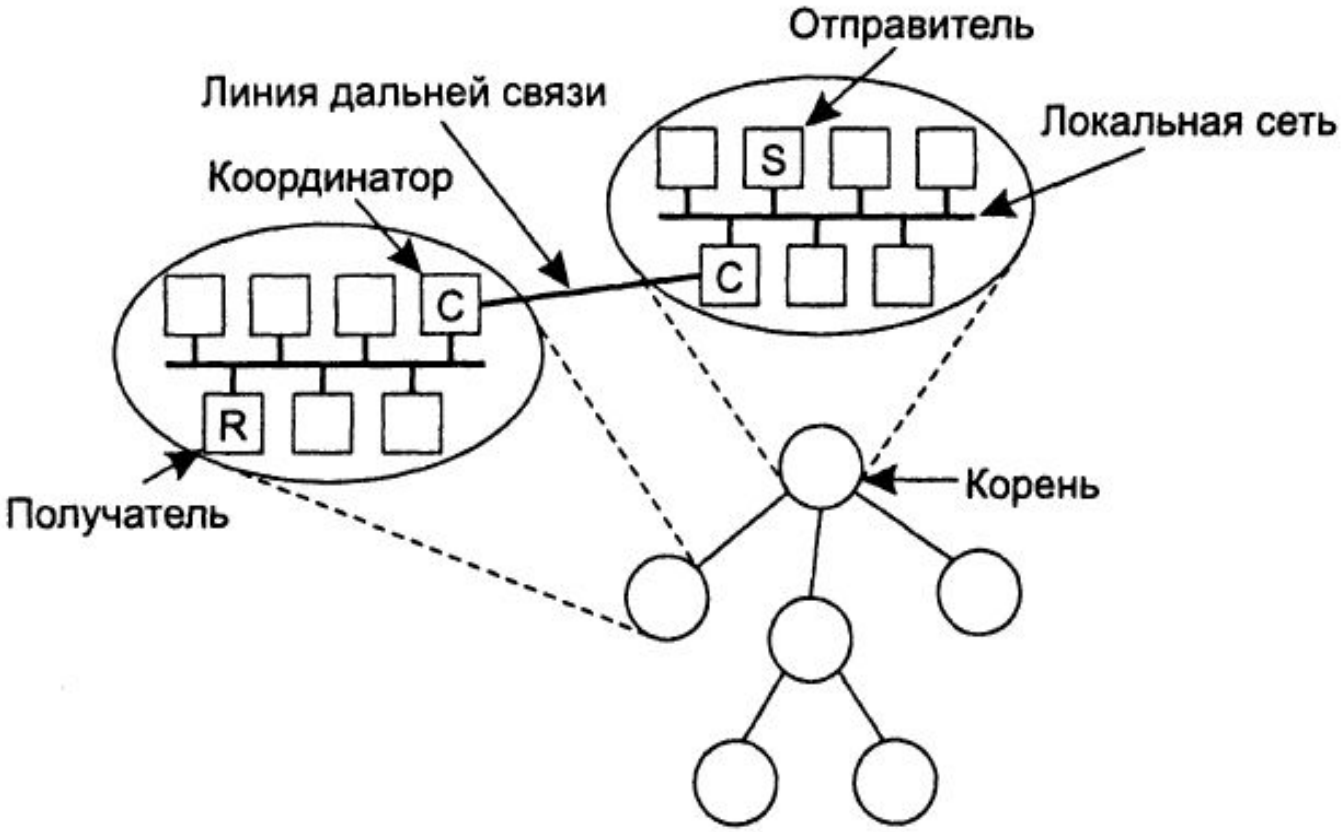
5. Поломка клиента

Не имеющие заказчика вычисления называются с*иротами (orphans).*

Решение:

- перед тем как клиент пошлет вызов RPC, создается запись в журнале с описанием того, что происходит. Журнал хранится в устройстве долговременного хранения информации, способном пережить перезагрузку;
- разбиение времени на последовательно пронумерованные эпохи. При перезагрузке клиента он путем широковещательной рассылки отправляет всем машинам сообщение, объявляющее о начале новой эпохи. Когда эта рассылка приходит на сервер, все удаленные вычисления, производимые там по заказу этого клиента, прекращаются;
- когда приходит сообщение о смене эпох, каждая машина проверяет, происходят ли на ней какие-либо удаленные вычисления, и если да, пытается найти их владельца. Вычисления прекращаются только в том случае, если владельца найти не удалось;
- каждому вызову RPC приписывается стандартная продолжительность работы *T.*

6. Надежная групповая рассылка



Каждый локальный координатор пересылает сообщения своим потомкам, а затем обрабатывает запросы на повторную передачу.

6. Надежная групповая рассылка

Если производится серия изменений и в ходе выполнения одного из них случается поломка реплики, то обновление этой реплики не происходит, в то время как обновления других реплик происходят успешно.

Если базовая распределенная система поддерживает **атомарную групповую рассылку**, то операция изменения, разосланная всем репликам перед тем, как произошла поломка одной из них, будет выполнена на всех корректно работающих репликах или не выполнена ни на одной из них.

7. Распределенное подтверждение

Задача распределенного подтверждения включает в себя операции, производимые либо с каждым членом группы процессов, либо ни с одним из них. В случае надежной групповой рассылки операцией будет доставка сообщения. В случае распределенных транзакций операцией будет подтверждение транзакции на одном из сайтов, задействованных в транзакции.

Распределенное подтверждение часто организуется при помощи координатора.

Варианты:

1. Протокол однофазного подтверждения (one-phase commit protocol)

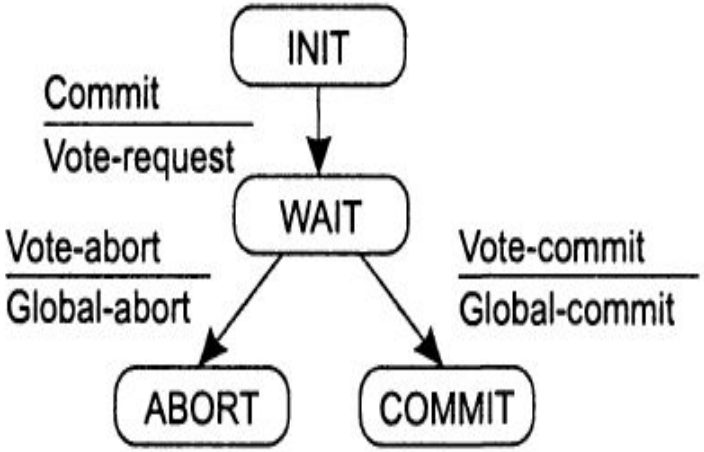
2. Протокол двухфазного подтверждения (two-phase Commit Protocol 2PC)

3. Протокол трехфазного подтверждения (three-phase Commit Protocol 3PC)

7. Распределенное подтверждение

Двухфазное подтверждение:

Координатор



Участник

7. Распределенное подтверждение

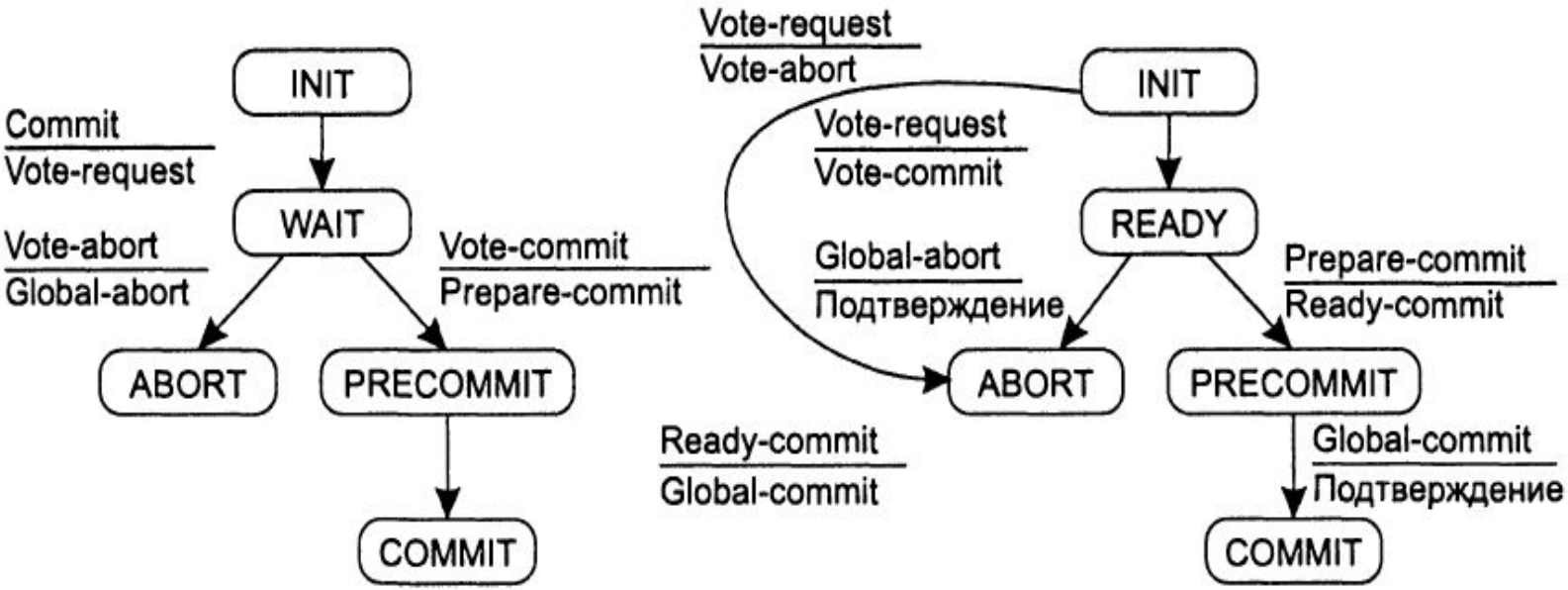
Двухфазное подтверждение:

1. Координатор рассылает всем участникам сообщение *VOTE_REQUEST.*
2. После того как участник получит сообщение *VOTE_REQUEST,* он возвращает координатору либо сообщение *VOTE_COMMIT,* указывая, что он готов локально подтвердить свою часть транзакции, либо сообщение *VOTE_ABORT* в противном случае.
3. Координатор собирает ответы участников. Если все участники проголосовали за подтверждение транзакции, координатор начинает осуществлять соответствующие действия и посылает всем участникам сообщение *GLOBAL_COMMIT .* Однако если хотя бы один участник проголосовал за прерывание транзакции, координатор принимает соответствующее решение и рассылает сообщение *GLOBAL_ABORT*
4. Каждый из участников, проголосовавших за подтверждение, ожидает итогового решения координатора. Если участник получает сообщение *GLOBAL_ COMMIT,* он локально подтверждает транзакцию. В случае же получения сообщения *GLOBAL_ABORT* транзакция локально прерывается.

Первая фаза (фаза голосования) состоит из шагов 1 и 2, вторая (фаза решения) — из шагов 3 и 4.

7. Распределенное подтверждение

Трехфазное подтверждение:



7. Распределенное подтверждение

Трехфазное подтверждение:

Не существует такого состояния, в котором невозможно принять итоговое решение, но возможен переход в состояние *COMMIT.*

Координатор 3PC начинает с рассылки всем участникам сообщения *VOTE_REQUEST* после чего ожидает прихода ответов. Если хотя бы один участник голосует за прерывание транзакции, это становится итоговым решением и координатор рассылает участникам сообщение *GLOBAL_ABORT.* Однако если транзакция может быть подтверждена, рассылается сообщение *PREPARE_COMMIT.* Только после того, как все участники подтвердят свою готовность к подтверждению, координатор посылает итоговое сообщение *GLOBAL_COMMIT*, в результате которого транзакция действительно подтверждается.

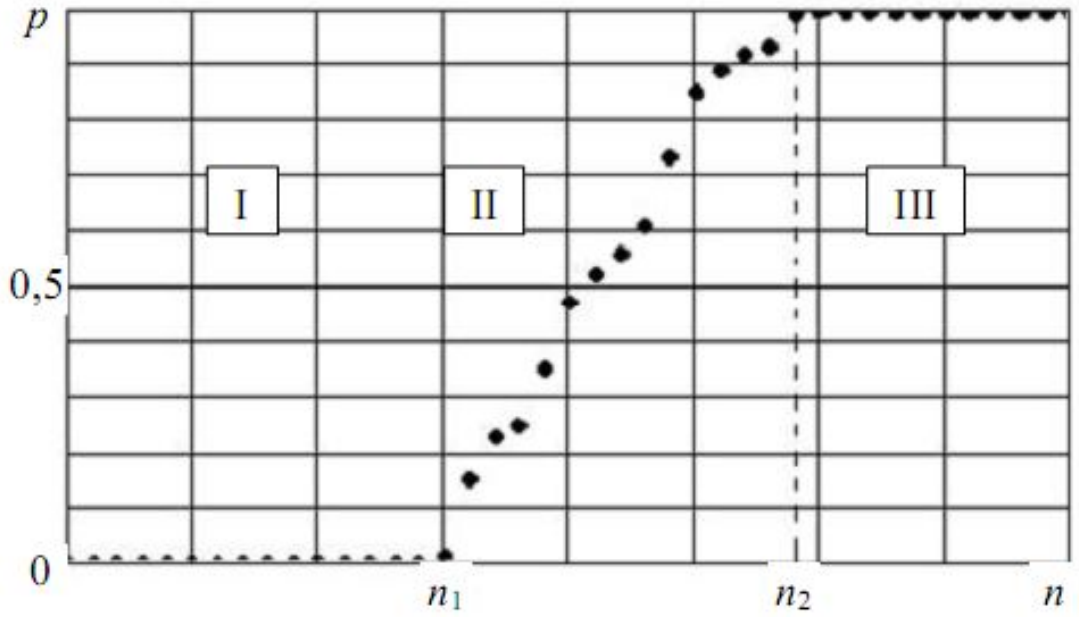
8. Восстановление

Основа отказоустойчивости — исправление после ошибок.
Два основных способа восстановления после ошибок:

- ***обратное исправление*** *(backward recovery)* - задача состоит в возвращении системы из текущего ошибочного состояния к предыдущему безошибочному состоянию. Чтобы сделать это, необходимо время от времени записывать состояние системы и восстанавливать ее в предыдущем состоянии. При каждой записи текущего состояния системы (или его части) говорят, что создается *контрольная точка (checkpoint).*
- ***прямое исправление*** *(fonoard recovery)* - при входе системы в ошибочное состояние вместо отката назад, делается попытка перевести систему в новое корректное состояние, в котором она могла бы продолжать работать.

9. Критерии отказоустойчивости

В РС возможно большое количество мест повреждений. Вероятность ее безотказной работы может быть оценена статистически (при помощи САПР или экспериментально).



Результаты статистической оценки вероятности отказа РС

9. Критерии отказоустойчивости

Такая оценка связывает отказоустойчивость с количеством возникших повреждений, при этом

p — статистическая оценка вероятности отказа системы;

n — количество отказавших элементов и/или связей между ними;

n1 — максимальное количество отказавших элементов и/или связей, при которых вероятность отказа равна нулю;

n2 — минимальное количество отказавших элементов и/или связей, при которых вероятность отказа равна единице.

9. Критерии отказоустойчивости

Полученное семейство точек условно разбито на три зоны:

— **зона I**: $0 \le n \le n1$; $p = 0$ — повреждений не больше, чем n1; система абсолютно работоспособна;

— **зона II**: $n1 < n < n2$; $0 < p < 1$ — повреждений больше, чем n1, но меньше, чем n2; система сохраняет работоспособность лишь при некотором наборе этих повреждений;

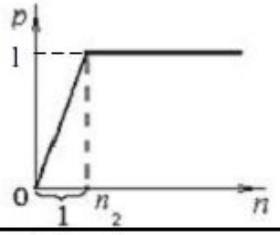
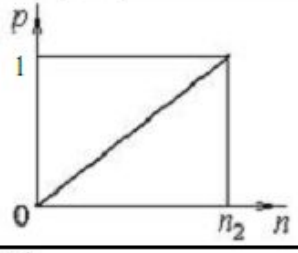
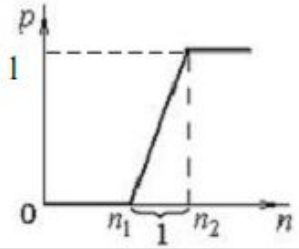
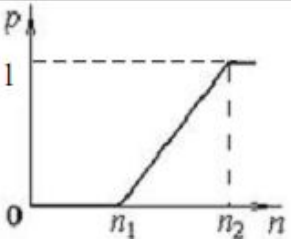
— **зона III**: $n2 \le n$; $p = 1$ — повреждений больше, чем n2–1; система неработоспособна при любом их наборе.

9. Критерии отказоустойчивости

Очевидно, что отказоустойчивость системы зависит от значений n1 и n2 и тем выше, чем дольше сохраняется нулевая вероятность отказа (т. е., чем больше n1), и чем больше наклонена к оси абсцисс прямая, аппроксимирующая экспериментальные точки в зоне II, (т.е. чем больше отношение (n2 – n1)/1 , или просто (n2 – n1).

В зависимости от наличия и протяженности вдоль оси n этих зон, можно классифицировать **четыре типа** отказоустойчивости системы.

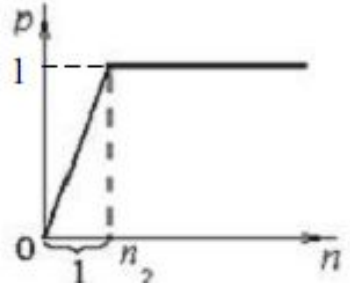
9. Критерии отказоустойчивости

№ типа	Тип отказоустойчивости	Графическая аппроксимация	Значения границ зон	Характеристика
1	Абсолютно неустойчива		$n_1 = 0;$ $n_2 = 1$	Первое же любое повреждение приводит к отказу
2	Негарантированная отказоустойчивость		$n_1 = 0;$ $n_2 > 1$	Отказоустойчивость пропорциональна p при $n > 0$
3	Гарантированная отказоустойчивость I		$n_1 > 0;$ $n_2 - n_1 = 1$	Система отказоустойчива до $n \le n_1$. При $n > n_1$ первое же любое повреждение приводит к отказу
4	Гарантированная отказоустойчивость II		$n_1 > 0;$ $n_2 - n_1 > 1$	Система отказоустойчива до $n \le n_1$.

9. Критерии отказоустойчивости

Приведенные типы отказоустойчивости представляют собой условную аппроксимацию экспериментальных точек прямыми.

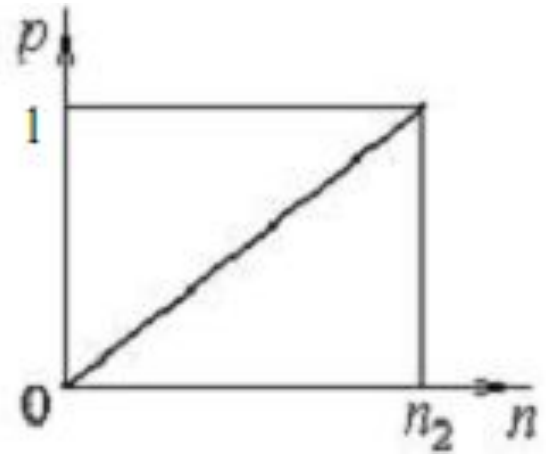
Первый тип по своим показателям соответствует **абсолютно неустойчивой** и в этом смысле ненадежной системе, первое же любое повреждение любого элемента (или связи) приводит к отказу. Такая система не нуждается в критериях оценки отказоустойчивости.



9. Критерии отказоустойчивости

Второй тип соответствует системе с **негарантированной отказоустойчивостью**, т.к. первое же любое повреждение может привести к отказу. В качестве критерия отказоустойчивости в таких системах можно использовать логарифмическую функцию:

$$K_2 = -\sum_{i=n_1+1}^{n_2-1} \log_2 p_i \,.$$

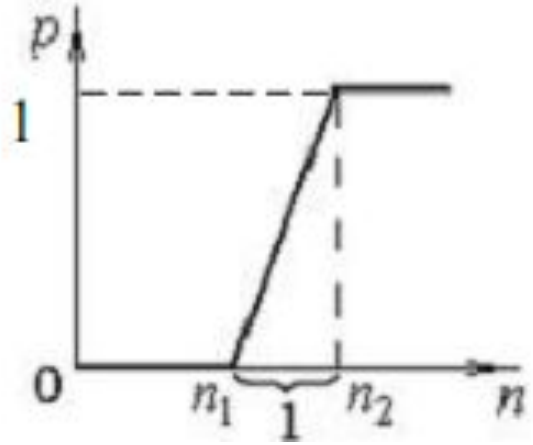


9. Критерии отказоустойчивости

Третий тип отказоустойчивости также является **ненадежным** с точки зрения эксплуатационных характеристик системы, поскольку после некоторого количества повреждений вероятность отказа скачком превращается из 0 в 1.

Однако он позволяет использовать в качестве критерия отказоустойчивости логарифмический критерий

$$K_3 = \log_2 n_1 .$$



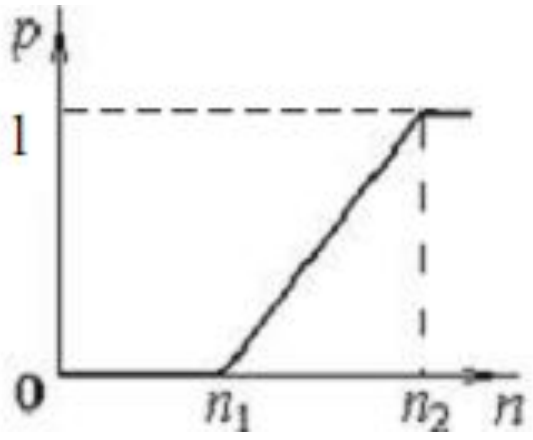
9. Критерии отказоустойчивости

В качестве критерия отказоустойчивости **четвертого типа** рекомендуется произведение

$$K_4 = K_2 K_3 = -\log_2 n_1 \sum_{i=n_1+1}^{n_2-1} \log_2 p_i ,$$

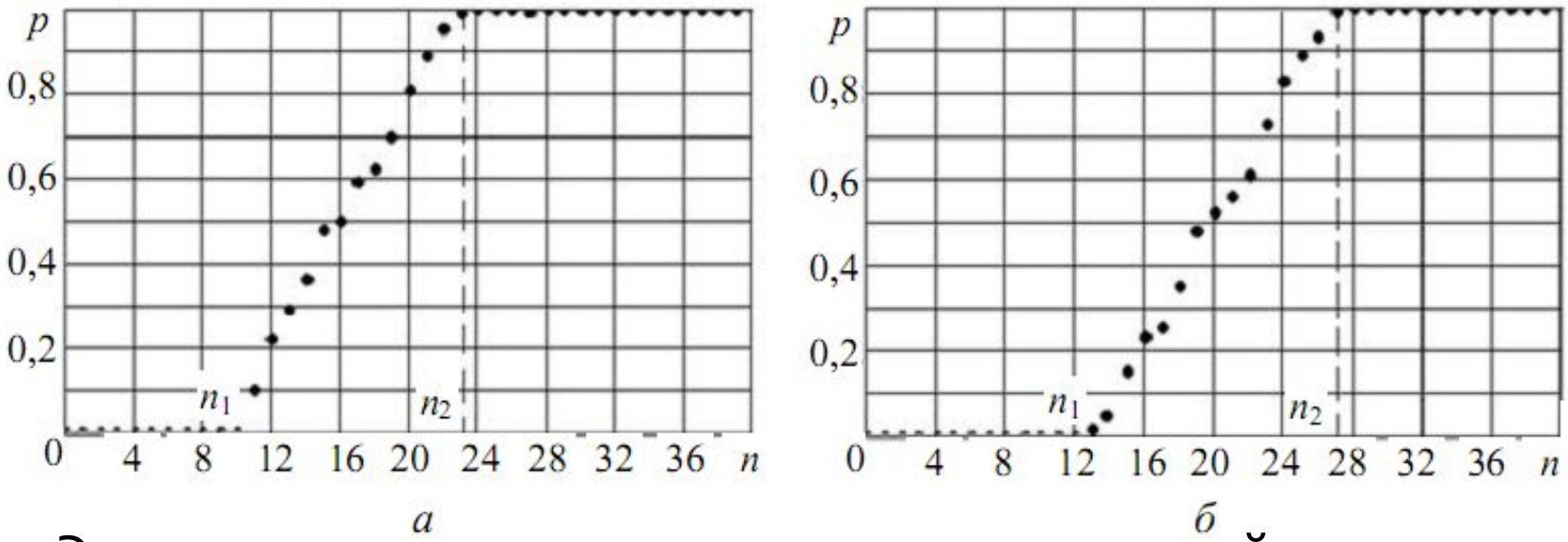
т.к. аппроксимирующая кривая отказов этого типа содержит участки, характерные для типов 2 и 3.

При сравнении отказоустойчивости нескольких систем предпочтение следует отдавать той, у которой больше и величина n1, и величина (n2 – n1).



9. Критерии отказоустойчивости

Пример



Экспериментальные значения статистической оценки вероятности отказа для двух различных систем

Учитывая изложенное, а также исходя из того, что у системы б величины n1 и (n2 – n1) больше, чем у системы а, можно предположить, что и значение логарифмического критерия отказоустойчивости у системы б больше, т.е. $K_4^{б} > K_4^{а}$.

9. Критерии отказоустойчивости

Каждая i-я экспериментальная точка содержит в себе информацию о вероятности двух событий, представляющих собой **полное множество**: “система отказала — pi” и “система работоспособна — (1 – pi)”.

Поэтому для оценки отказоустойчивости в пределах зоны II можно применить логарифмическую зависимость

$$K = -\sum_{i=n_1+1}^{n_2-1} [p_i \log_2 p_i + (1-p_i)\log_2(1-p_i)].$$

10. Отказоустойчивые сервера

Катастрофоустойчивый кластер

10. Отказоустойчивые сервера

Отказоустойчивость виртуальных серверов:

- В широком смысле слова, **виртуализация** – это программная имитация некоей физической сущности. К примеру, компьютерные игры – это ничто иное, как виртуализация реальной жизни, раздел жесткого диска, видимый в системе как отдельный диск – это виртуальный диск, аналогично - эмуляция физического привода CD-ROM – программа так и называется – Virtual CD.
- Здесь же и далее мы будем говорить о виртуализации серверов. Примеры – Microsoft Virtual PC и продукция VMWare

  В настоящее время виртуальные машины все чаще используют не только в тестовых целях, но и в промышленной среде. Какой же в этом смысл?

10. Отказоустойчивые сервера

Отказоустойчивость виртуальных серверов

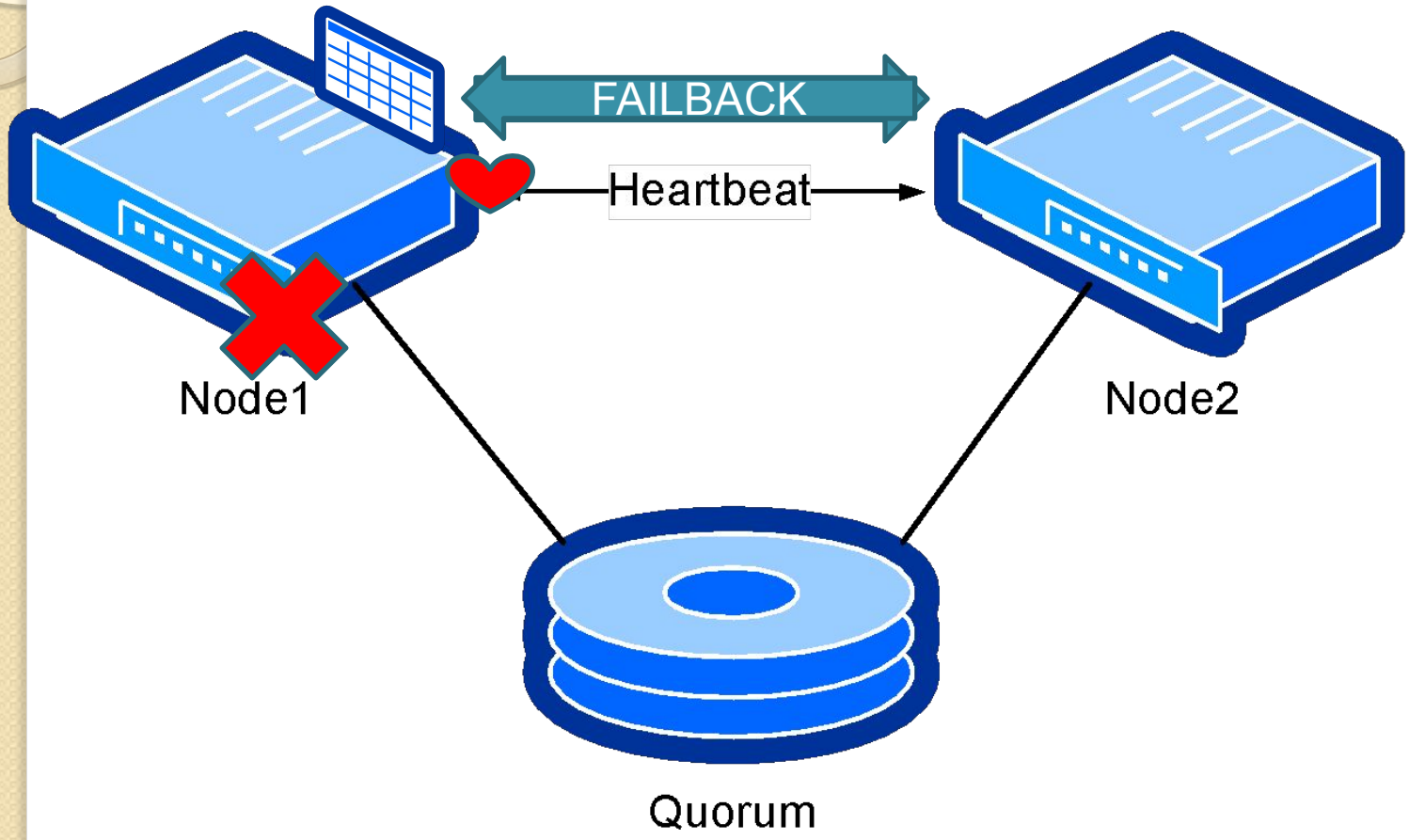
Преимущества:

-Рациональное использование аппаратных ресурсов
-Экономия денег, места и электроэнергии
-Удобство администрирования

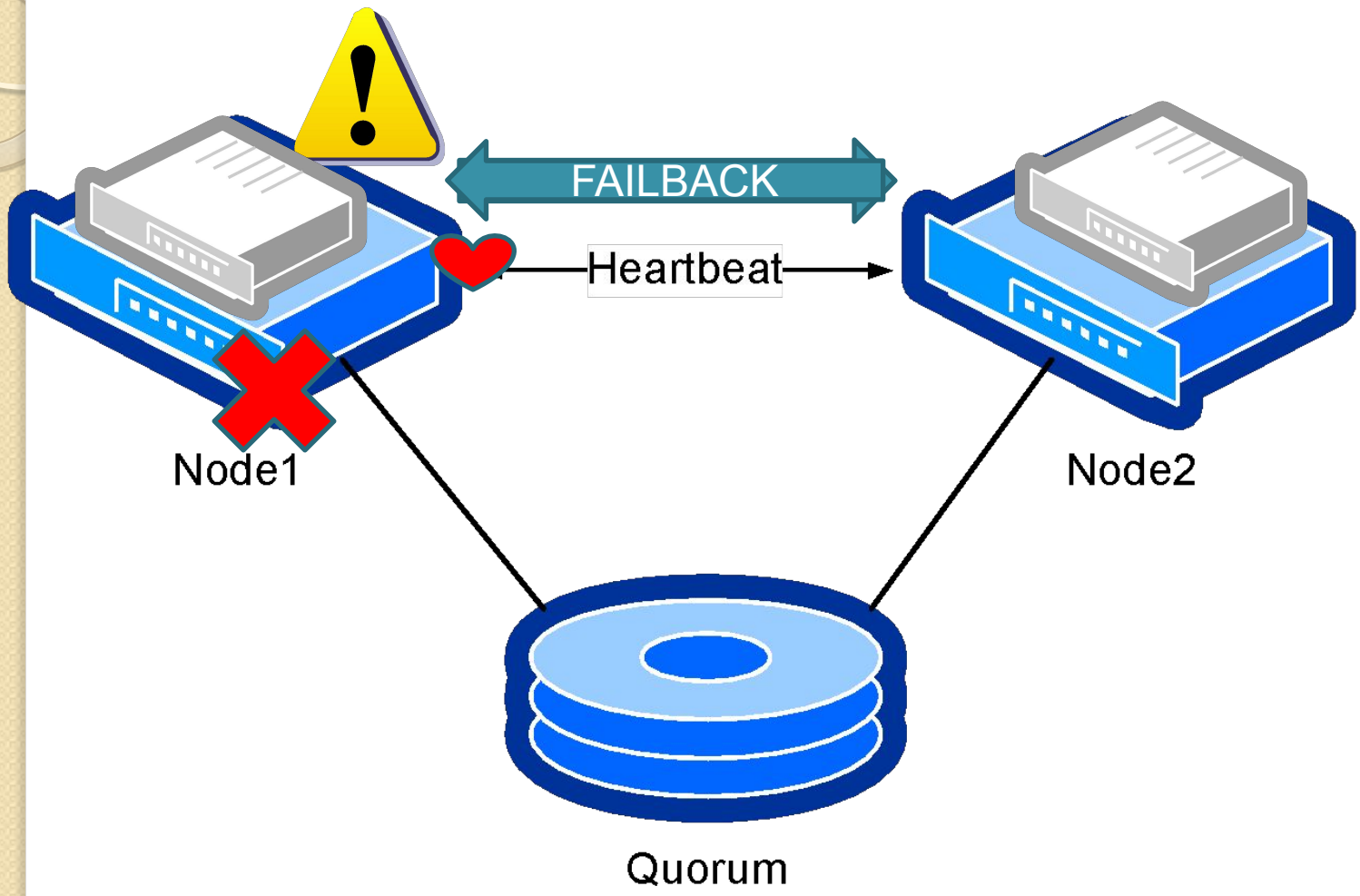
Недостатки:

-Необходимость приобретения более мощного оборудования
-Единая точка отказа – физический хост

10. Отказоустойчивые сервера



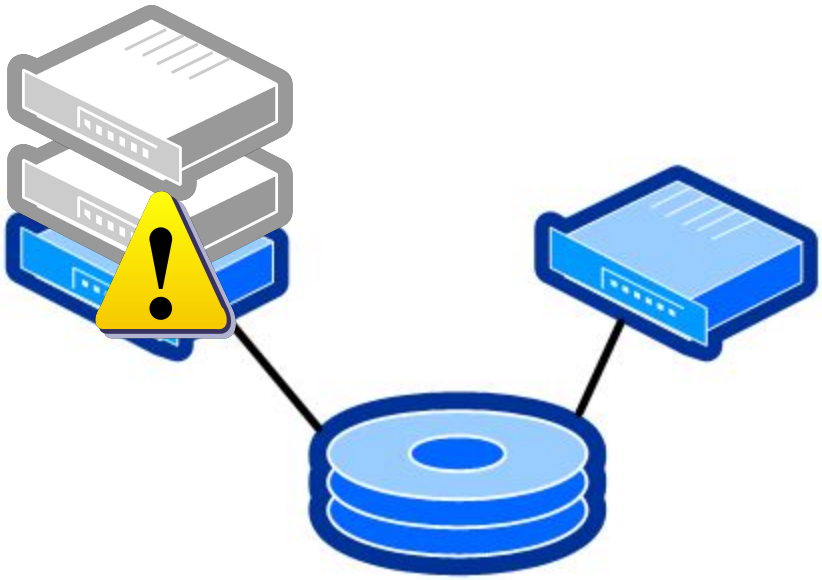
10. Отказоустойчивые сервера



10. Отказоустойчивые сервера

Обслуживание серверов

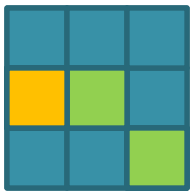
1. Переносим виртуальные машины на другой сервер
2. Производим обслуживание (установка обновлений, замена оборудования…)
3. Переносим виртуальные машины обратно на готовый сервер

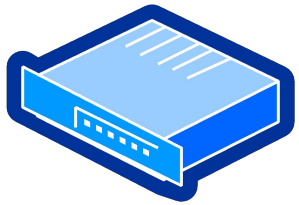


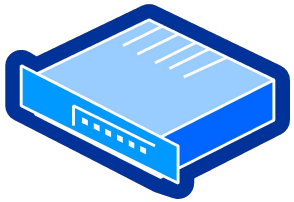
10. Отказоустойчивые сервера

Копирование содержимого памяти осуществляется по сети страницами по 4Кб.

1. Копируется все содержимое памяти виртуальной машины.
2. Если в процессе копирования содержимое некоторых страниц изменилось – копируются измененные страницы.
3. П.2 повторяется до полной идентичности содержимого памяти на обоих узлах.
4. Если полной идентичности не удается достичь за 10 итераций – выводится сообщение о невозможности осуществления миграции.







Выводы

Отказоустойчивость определяется как способность системы маскировать появление ошибок и исправлять их. Система отказоустойчива, если она продолжает функционировать при наличии отказов.

Существуют разные типы отказов. Поломка означает простую остановку системы. Пропуск данных наблюдается в том случае, если процесс не реагирует на входящие запросы. Если процесс отвечает слишком быстро или слишком медленно, говорится об ошибке синхронизации. Отклик на входящий запрос с ошибкой — пример ошибки отклика. Сложнее всего обрабатывать ситуации, когда в процессе может случиться ошибка любого типа – произвольная ошибка.

Избыточность — это стандартный способ обеспечения отказоустойчивости.

Выводы

В условиях протокола двухфазного подтверждения координатор сначала проверяет готовность всех процессов к выполнению одной и той же операции (то есть к ее подтверждению), а на втором этапе рассылает результат этого голосования. Протокол трехфазного подтверждения используется для того, чтобы справиться с отказом координатора без необходимости блокировать все процессы, для достижения соглашения дожидаясь восстановления координатора.

Восстановление в отказоустойчивых системах неизменно организуется на основе регулярного сохранения информации о состоянии системы. Работа с контрольными точками полностью распределена. К сожалению, создание контрольной точки — довольно затратная операция.

Выводы

Для повышения производительности множество распределенных систем сочетают контрольные точки с протоколированием сообщений. Благодаря протоколированию взаимодействия между процессами после отказа системы можно воспроизвести ход ее функционирования.

Рассмотрены критерии отказоустойчивости распределенных систем, приведена классификация РС по степени отказоустойчивости, рассмотрены примеры отказоустойчивости классических серверов и отказоустойчивые виртуальные сервера.

Список использованной литературы

Черкесов Г.Н. Надежность аппаратно-программных комплексов.
Рябинин И.А. Надежность и безопасность структурно сложных систем.
Курочкин Ю.А., Смирнов А.С., Степанов В.А. Надежность и диагностирование цифровых устройств и систем.
Майерс Г. Надежность ПО.
Майерс Г. Искусство тестирования программ.
Холстед М. Начала науки о программах.
ГОСТ 27.002-89. Надежность в технике. Основные понятия. Термины и определения.
ГОСТ 27103-83. Надежность в технике. Критерии отказов и предельных состояний. Основные положения.
Танненбаум. Распределенные системы.
Додонов А.Г. Введение в теорию живучести вычислительных
Плачинда О.Е. Методы оценки отказоусойчивости сложных технических систем
Моделирование отказоустойчивости в САПР сложных технических систем / А.Л. Становский, В.М. Тонконогий, О.С. Савельева и др.

СПАСИБО ЗА ВНИМАНИЕ!