Тупики при транзакциях  
…….  
как много раз она обращала внимание наше, методы борьбы с тупиками используются те же, что и в централизованных системах  
…..  
предварительная проверка запросов процессов, обход тупика, чей процесс не может быть удовлетворен, может быть завершен, обнаружение тупиковых систуация, система работает не задумываясь, что может попасть в тупиковую ситуацию (подозрения по таймаутам), тогда в системе может быть запущена проверка, находится ли система в тупике

Мы уже рассмотрела недопущение тупиков, (предотвращение), аналогично централизованной системе, как предполагается – транзакция должна запросить все блокировки (среда обесп монопольного доступа к разделяемым ресурсам – локи)

Избежание тупиков (обход – мы называем), аналогично ситуации в центр системе – избежать распределенный тупик можно за счет предвар обработки ситуации, которая может возникнуть, если запрос транзак на блокировку будет удален.

Дополнительно в расперед системах должны быть решены вопросы положения транакций (локейшин) и контроль транакции, при этом могут возникнуть именно из распределенной природы траназкций след конфликты, след конфликты (при этом надо разделять конфликты между двумя транзакциями на одной сайте и 2 транзак на разных сайтах)

Случае конфликта одна транакция может быть прервана или ей разрешено ждать в соответствие с распредел алгоритмами ожидания, или можно коротко нахвать – в соответствии с распределенным ожиданием.

Предположит имеются 2 траназк Т1 и Т2. Т1 – эрайвз (превыбает) на сайт Р и пытается зав=блокировать элемент данных, который уже завачен траназ Т2 на этом же сайте. Этот сайт называют зоной Р (сайт Р – зона Р) ((хз Р англ или П русская)

И возможны 3 алгоритма

1. Distributed wound-die – распределенная смертельная рана)

первый способ – если Т1 старше Т2, то есть Т2 пришла позже, то Т1 может подождать, при этом Т1 может возобновить свое выполнение после того, как сайт Р получит сообщение о том, что транз Т2 успешно завершилась или была прервана.

второй способ – если Т1 младше Т2, то Т1 прерывается. При этом специальный менеджер на сайте Р должен отправить сообщение всем сайтам, которые посещала Т1, чтобы корректно ее прервать. При этом управляющий сайт должен уведомить пользователя, что Т1 была прервана на всех сайтах успешно.

1. Distrubuted wait-wait – распределенное ожидание ожидание (да да, именно 2 раза слово ожидание). Опять же два способа

первый способ– Если Т1 старше Т2, то Т2 следует прервать. Если Т2 активна (то есть выполняется на сайте Р), то сайт Р прерывает и откатывает Т2, после этого рассылает сообщение об этом на другие сайты. Если Т2 покинула сайт Р, (то ест завершилась на сайте Р), но выполняется на сайте (фул???), то сайт Р оповещает, что Т2 была прервана. Но по логике вещей сайт Q должен прервать эту Т2 и отправить соответственно сообщение другим сайтам, или это может сделать сайт управляющие транзакцией (скажем Лидер).

второй способ?? – Если Т1 младше Т2, то Т1 разрешено ждать ( тое сть Т1 блок в ожидании). При этом т1 сможет возобновить выполнения полсе того, как сайт Р получит сообщение , что Т2 завершилась.

Это формальные вещи и она это понимает. Почему нам это рассказывают?? – это современный материал проф значимый относительно тупиков, это больша проблема для распред прилож, который в большей части выполняются как транзак. (Т – очень интересный механизм).------ тут все про транакк началось (я расписывалась и не записала) ВСЕ БЫЛО ИЗЛОЖЕНо!!!

И последний метод связан с распредел транк

1. Обнаружение распределенных тупиков (англ вариант я не запишу). Также как и в центр системах, обнаружить, что распределенная транзакция попала в тупик можно с помощью графа, но при этом создаются так называемые глобальные графы ожидания (картинка есть – она уникальная, взята из старой книги …автор…. больше она ее не встречала нигде!!!!!!) Наличие цикла ожидание в глобальном графе является признаком тупика или взаимно блокировки (дедлоком). Но несмотря на то, что механизм известен, обнаружить такую блокировку сложно, потому что транакция распределенная и это значит, что она ожидает ресурсов в сети, поэтому для выявления таких ситуаций используются таймера. Каждая Т связана с таймером, установленным на определенный интервал времени (будильник). Очевидно, что этот интервал выбирается предположительно разработчиками и отрадает одидаемое время завершение траназ. Если Т не завершает за установленный интервал времени, то формируется признак – возможный…… тут ты мне писала и я не успела….

Есть еще один инструмент для обнаружение тупиковых ситуация, так называемый детектор тупиковых ситуаций. В централизованной системе имеется один детектор тупика. В распределенной системе может быть несколько. При этом детектор в распед системе может обнаруживать тупиковых ситуаций для подконтрольных ему сайтов.

Обратить внимание на очевидность подхода – параллельное обнаружение (сократить время обнаружения). Имеются 2 альтернативы для обнаружения тупиков в распредел системе.

1. Централиз детектор тупика – то есть одна локация назначается центральным детектором взаимоблокировки
2. Иерархический детектор тупика. Детекторы сформированы в иерархию ну и уже дело разработчика системы , как будет построено взаимодействие системы в этой иерархии.
3. Детектора распредел тупиков. Все сайты участвуют в обнаружении и устранении тупика.

—------------------------------------------------

три показательные теоремы (на прошлой лекции она забыла)

1. Теорема 1: граф – речь идет о бихроматическом направленном графе Холда, Граф явл полностью сокращаемым, если существует такая последовательность сокращений, которая устраняет все дуги. Если граф нельзя сократить полностью, то анализируемое состояние является тупиком (см рисунок полностью сокращаемого графа). Какие дуги можно удалить, дуги запросов…… Какая дуга запросов может быть удалена, та которая отражает запрос на ресурс, который есть у системы, или который освобожден.
2. Теорема 2: Цикл в графе повторных ресурсов является необходимым условием тупика.
3. Если состояние S (С большое) не является состоянием тупика и из S->T (из С система переходит в Т), то состояние Т(может) – явл состоянием тупика, если операция процесса Pi (п итое) явл запросом. И в результате Pi находится в T в тупике. Тупик может наступить в системе в результате запроса на доп ресурсы.

—--------------------------------------------------

Дополнительно: АЛгоритм обнаружения тупиков в распредел системах, который имеет название

Алгоритм Chundy-Misra-Hoas (по именам авторов)

Если при очередном запросе в системе необходимый ресурс занят, то процесс запросивший этот ресурс генерирует спец сообщение и посылает его процессам которые предположительно могли захватить данный ресурс, это система взаимодействующих процессов,...........................................

В этом СПЕЦИАЛЬНОМ сообщении указываются 3 числа: 1- свой идентификатор (номер в системе), 2 - также свой номер, 3 - номер процесса, которому посылается сообщение. Процесс получивший сообщение проверяет ждет ли он сам ресурс, занятый другим процессором (не этот же самый, а просто находится ли он в состоянии блокировки). Если он тоже блокирован, (вопрос – не получает проц время и не может выполняться —ответ это какая-то спец блокировка может быть), то прописывает свой номер во второе поле полученного сообщения. В 3 поле записывает номер процесса, захватившего ресурс, освобождение которого он ожидает, то есть он редактирует полученное сообщение и посылает его дальше. В результате такой многократной пересылке: Если процесс получает сообщение и в первом поле этого сообщения обнаруживает свой номер, и главное в 3 поле он также обнаруживает свой номер, то ничего ему не остается, как сделать вывод о том, что система в тупике (круговое ожидание). Очевидно, что можно уничтожить этот процесс. Однако если в тупике много процессов, то такой подход довольно расточителен, очевидно, что можно по разному реализовывать выход из тупика, например можно доп, чтобы процессы в своем сообщение прописывали свои номер, но еще и отдельный список процессов– получивших это сообщение. И тогда , например, можно завершить процесс с наибольшим номером.

Обращает наше внимание на след обстоятельства, которые не указывала в методе убеждения (Ха может и не такой метов) тупика.

тут ломаный англ, дестребьютет девойд ……

Каждая Т хранит время своего создания, при этом очевидно, предполагается, что время синхронизировано, например по алгоритму Лампорта, в результате можно определить, какая Т старше, а какая моложе, на этом построены способы предубеждения. И также обращает внимание: Например, блокирование Т происходит тогда, когда Т запроащив ресур моложе той, которая ресурс удерживает.

—------------------

Дополнительно обращает внимание.

В алгоритме, где 3 автора (выше написан). Примитивное решение – посылаются сообщения зомби…. (не уверена).

Если процесс обнаруживает, что слишком должно ждет, то может послать такое сообщение и также возможно использование сообщения зомби.

Это распространенный способ в локальных сетях (чтобы быть уверенным, что сеть свободна и передача выполняется надежно)

НОВАЯ ТЕМА

Архитектура ядер операционных систем

Существует 2 типа ядер

1. монолитное
2. микро- ядро

2 тип – микроядерной архитектуры, как только не называют, и нано и экза, но идея одна и та же

1 тип – исторически первый. Монолионое ядро – единая программа, сотсоящая из подпрограмм, собруутинс (ее английский)

В результате – любое изменение вносимое в такую программу, требует ее перекомпиляции.

Очевидно, что современные системы с монолитным ядром, ……….

Юникс, линукс– с монолитными ядрами.

В обоих системах ОС , которые позволяют вносить доп функционал без его перекомпиляции.

2 тип – микро керал (ее англ)

В отличии от 1 типа, очень небольшое ядро, которое выполняет только самые низкоуровневые действия, при этом другие компоненты ОС являются самостоятельными программами, которые выполняются в разных АП, причем в режиме пользователя.

Начнем с монолитного ядра.

Юникс линукс в отличии от винды….

Все монолитное ядро построено на прерываниях, как мы говорили , в ОС существует система прерывания, в современных системах принято выделять 2 типа прерывания, 1 системные вызовы, 2 – это исключения, 3 – аппаратные прерывания, ну только аппаратные прерывания называются интераптс (мой коверканный англ)

Когда хотят подчеркнуть, что ОС представляет интерфейс пользователя., то говорят об опи, при этом Опи определены в позикс

но например функция сигнал, в позикс не входит, но входит в анси-с, и от этого она не перестает быть ОПИ.

еще в середине 1960 годов, товарищи Меник и Доновал презентовали иерархическую машину, она отображает расположения соответствующих функций ядра по отношению к аппаратной части.

сейчас будет рисунок…..

Вич машина – это программно управляемое устройство (железка, аппаратура)

на последние минут 10 меня не хватило……………………………