СУПЕРКОМПЬЮТЕРНЫЙ КОНСОРЦИУМ УНИВЕРСИТЕТОВ РОССИИ

Проект

Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения

Московский государственный университет им. М.В. Ломоносова

Нижегородский государственный университет им Н.И. Лобачевского
Факультет Вычислительной математики и кибернетики

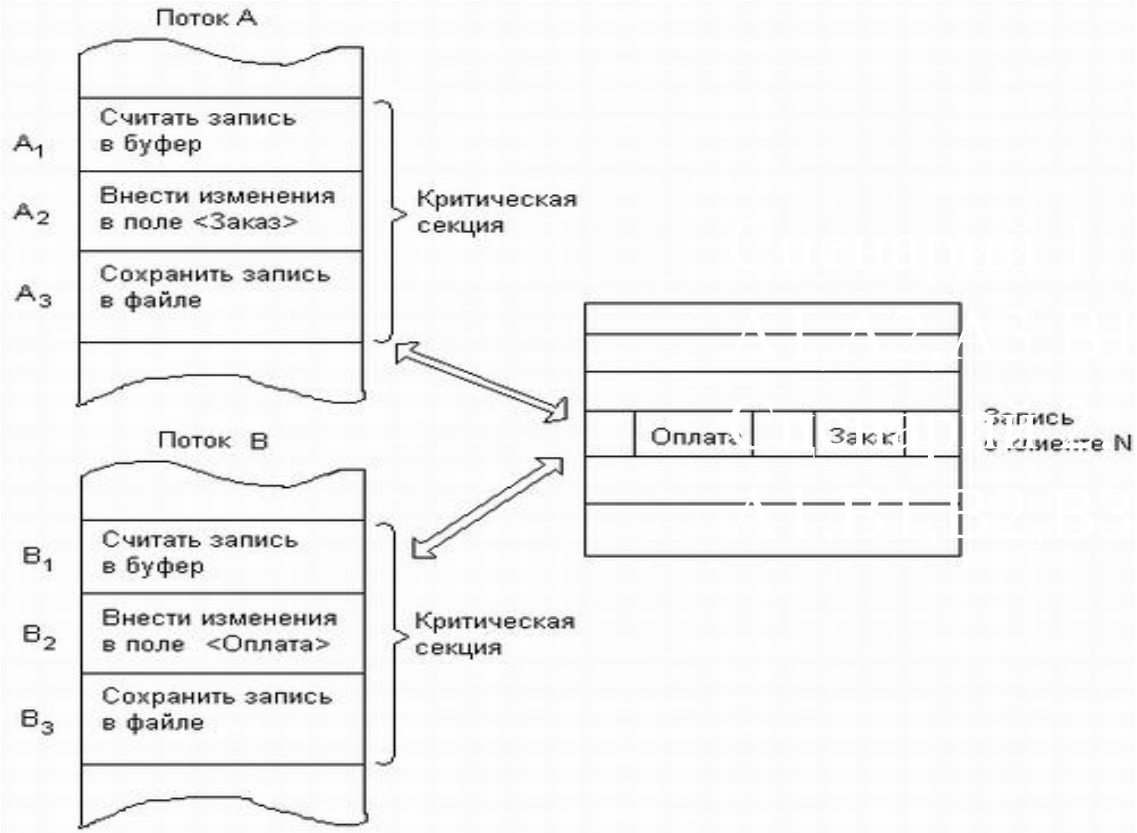
Методы и средства построения распределенных программных систем с использованием технологии Java

Лекция 6.
Параллелизм

Лекция 6

- Какие проблемы, возникают при одновременном доступе к ресурсам в параллельной среде?
- Какие существуют механизмы разрешения проблем параллельного доступа к ресурсам
  - Контроль на уровне алгоритма (критические секции)
  - Контроль на уровне ресурса (блокировки)
- Тупики, как последствия применяемых механизмов
- Понятие транзакции
- Уровни изоляции транзакций
- Вложенные и распределенные транзакции
- Репликация

Параллельные потоки (процессы)



Параллельные потоки (процессы)

- Условия возникновения:
  - Одновременное выполнение нескольких процессов (например, в случае многопоточного сервера, наличие двух и более одновременно выполняемых клиентских запросов)
  - Доступ к одним и тем же данным (объектам)
- Синхронизация процессов (потоков)
  - Критические секции
- Блокирование ресурсов (данных)
- Понятие транзакционности при обработке данных
  - Принципы АСИД

Критические секции

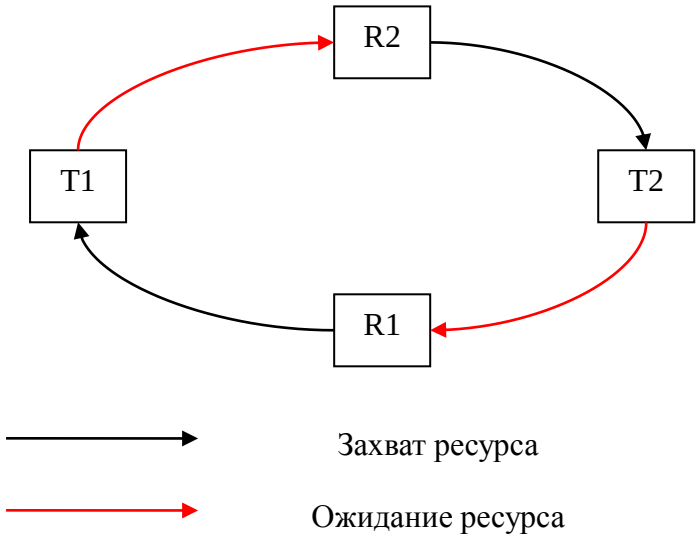
- Части кода приложения, для которых не допускается одновременное выполнение несколькими потоками
- Реализация
  - Внешние (ОС) примитивы синхронизации (например - семафоры)
  - Внутренние примитивы синхронизации (поддержка не на уровне ОС, а на уровне прикладных библиотек)

Блокировка ресурсов

- Ресурсы, допускающие совместное использование не более, чем n процессами (часто n=1), *захватываются* процессами. При успешном захвате ресурса n уменьшается на 1. При освобождении ресурса n увеличивается на 1 (семафор)
- При попытке захватить ресурс, у которого n = 0, процесс блокируется. Выход из блокировки – при увеличении n (при доступности ресурса)
- Реализация
  - Внешние (ОС) примитивы синхронизации (например - семафоры)
  - Внутренние примитивы синхронизации (поддержка не на уровне ОС, а на уровне прикладных библиотек)

Проблема тупиков

- Возникают при использовании критических секций и блокировок ресурсов



Проблема тупиков - решение

- Ответственность на программисте
  - Аккуратное программирование. Один из методов – соблюдение одной и той же последовательности наложения блокировок во ВСЕХ процессах.
- Ответственность на менеджере блокировок (ОС, …)
  - Обнаружение: использование графа ожидания ресурсов (пример – на предыдущем слайде). Блокировка = цикл в графе
  - Устранение:
    - Метод 1: отменить (снять, прервать) один из процессов (любой), участвующих в блокировке
    - Метод 2: отслеживается начало каждой блокировки и устанавливается ее максимальная продолжительность

Транзакции:

- Транзакция - единая неделимая последовательность действий, представляющую собой некоторую операцию в системе и удовлетворяющая принципам АСИД:
  - **Атомарность**. Транзакции атомарны (выполняются все действия, входящие в транзакцию, или ни одно из них)
  - **Согласованность**. Транзакции защищают данные согласованно. Это означает, что транзакции переводят одно согласованное состояние системы в другое без обязательной поддержки согласованности во всех промежуточных точках
  - **Изоляция**. Транзакции отделены одна от другой (не оказывают друг на друга влияния)
  - **Долговечность**. Когда транзакция выполнена, ее обновления сохраняются, даже если в следующий момент произойдет сбой системы

Транзакции: типичная семантика

- Типичная реализация механизма транзакций обеспечивает, по меньшей мере, следующие операции:
  - операция начала транзакции (beginTransaction)
  - операция фиксации транзакции (commit)
  - операция отмены транзакции (rollback)
- для реализации предложенного подхода необходимо наличие координатора (менеджера) транзакций – специального компонента, обеспечивающего управление транзакциями. Типичным является размещение этого компонента на стороне сервера.

Транзакции: примеры использования

- Банковский перевод
- Перевод представляет собой согласованную последовательность ДВУХ действий
  - Снять сумму со счета–источника
  - Положить сумму на счет–приемник
- Важные замечания:
  - В случае ошибки при выполнении ЛЮБОЙ операции из этой последовательности, должны быть отменены (откат транзакции) все выполненные ранее операции (понятие обратимости операции)
  - Откат транзакции может быть вызван как нарушением логики системы (счет–приемник закрыт), так и отказом (в том числе - оборудования)
  - Два состояния – перед началом транзакции и после ее окончания являются согласованными. Состояние «в процессе выполнения» транзакции является не консистентным для системы

Транзакции: примеры использования

- Процедура подсчета суммы средств на счетах:

	t1	t2
Иванов	120	20
Петров	150	150
Сидоров	240	240
…		
…		
Яковлев	400	500
…		

- Вывод – с введением понятия транзакции исчезают старые проблемы и появляются новые

Транзакции: возможные проблемы

- Проблема потери результатов обновления
- Проблема незафиксированной зависимости
- Проблема несовместного анализа

Проблема потери результатов обновления

Время	Транзакция 1	Транзакция 2
t1	beginTransaction	beginTransaction
t2	B:= Иванов.ballance	
t3		B:= Иванов.ballance
t4	B:=B+100;	B:=B+200;
t5	Иванов.ballance:=B;	
t6	Commit;	Иванов.ballance:=B;
t7		Commit;

В результате выполнения этого сценария, изменения, сделанные первой транзакцией оказались потерянными

Проблема незафиксированной зависимости

Время	Транзакция 1	Транзакция 2
t1	beginTransaction	beginTransaction
t2		Иванов.ballance+=100;
t3	B:= Иванов.ballance	
t4	B:=B+100;	
t5	Иванов.ballance:=B;	
t6	Commit;	
t7		Rollback;

В результате выполнения этого сценария, транзакция1 внесла изменения, рассчитанные на основании данных, которых «не существует» (транзакция2, которая эти данные внесла, отменила свои изменения)

Проблема несовместимого анализа

	t1	t2
Иванов	120	20
Петров	150	150
Сидоров	240	240
…		
…		
Яковлев	400	500
…		

Проблема несовместимого анализа возникает в том случае, если одна из транзакций в силу длительности выполнения застает часть данных в одном состоянии (неизмененными), а часть в другом (уже измененными)

Транзакции: уровни изоляции

- Уровень изоляции транзакций определяет влияние, которое оказывают друг на друга параллельно выполняющиеся транзакции (имеет смысл только в случае параллельно выполняющихся процессов). Уровень изоляции показывает, какие из «типичных» проблем возможны при его использовании, а какие – нет.

Основные проблемы, возникающие при параллельной обработке данных следующие:

- проблема потери результатов обновления
- проблема незафиксированной зависимости
- проблема несовместимого анализа.

Транзакции: уровни изоляции

- Уровни изоляции транзакций определены относительно некоторых эффектов параллельной обработки: уровень изоляции определяет возможность возникновения этих эффектов

Уровень изоляции	Грязное чтение	Неповторяемое считывание	Фиктивные элементы
Грязное чтение (Dirty read)	да	да	да
Завершенное считывание (Read commited)	нет	да	да
Повторяемое чтение (Repeatable read)	нет	нет	да
Способность к упорядочиванию (Serializable)	нет	нет	нет

Транзакции: уровни изоляции

- **Неаккуратное (грязное) считывание**. Допустим транзакция T1 выполняет обновление с некоторым объектом, затем транзакция T2 извлекает состояние этого объекта, после чего выполнение T1 отменяется. В результате транзакция T2 обнаружит, что извлеченное ей состояние объекта никогда не существовало (поскольку не зафиксирована T1)
- **Неповторяемое считывание**. Допустим, транзакция T1 извлекает состояние объекта, транзакция T2 изменяет это состояние и фиксирует транзакцию, после чего T1 вновь извлекает состояние этого же объекта. Получается, что один и тот же объект для T2 в разные моменты времени будет иметь разные состояния
- **Фиктивные элементы**. Допустим, что транзакция T1 извлекает какое-то множество элементов (процесс продолжителен во времени), транзакция T2 добавила в это множество элемент и зафиксировала транзакцию. T1, продолжая извлекать элементы, обнаружит элемент, которого раньше во множестве не было

Транзакции: откат

- **Важнейшая задача транзакционной обработки – атомарность – предполагает возможность отката сделанных изменений в случае, если какое-то из действий, составляющих транзакцию, закончилось неудачей**
- Возможные реализации:
  - изменения, которые делает транзакция в процессе работы, записываются не в долговременную память, а во временную. В случае фиксации транзакции происходит перенос данных из временной памяти в основную, а в случае отката временная память просто очищается
  - *журналирование* операций, выполняемых транзакцией. Все изменения, которые производит транзакция записываются в журнал операций. В случае отката транзакций часть журнала, относящегося к прерванной транзакции, может быть просто уничтожена (часто в этом случае в журнал операций записывается специальная операция отката транзакции), а в случае фиксации транзакции – операции из журнала должны быть «проиграны» на основной долговременной памяти

Вложенные и распределенные транзакции

- Транзакции могут затрагивать «не локальные системы» (несколько таких систем) – быть распределенными
- Транзакции могут включать в себя другие транзакции (образовывать деревья транзакций) – содержать вложенные транзакции
- Пример: бронирование тура
  - Поиск подходящего отеля и бронирование в нем места
  - Поиск подходящего авиарейса и бронирования места на нем
  - Поиск подходящего обратного авиарейса и бронирование места на нем
  - Заказ необходимых дополнительных услуг (экскурсий, и т.д.)
  - Оплата через банк всего пакета услуг (проживание, перелет, экскурсии)
- Вовлечены несколько независимых систем
- Каждая из операций – может быть транзакцией в терминах своей системы

Поддержка распределенных транзакций

- Двухфазный протокол фиксации транзакций (2PC)
  - В системе выделяется менеджер транзакций (управляющий компонент; во многих реализациях он называется координатором транзакций)
  - Фиксация изменений транзакции происходит в два этапа. На первом этапе изменения верифицируются. Производятся все необходимые проверки, которые гарантируют, что внесенные изменения полностью готовы к переносу в долговременную память (некоторые реализации уже на этой фазе переносят изменения в долговременную память, помечая их как «незафиксированные»). Вторая фаза начинается только после того, как все узлы, вовлеченные в транзакцию, отрапортуют координатору об успешном завершении первой фазы. Вторая фаза состоит в простом переносе уже подготовленных изменений в долговременную память

Протокол координатора (пример)

- Фаза 1
  - Координатор посылает сообщение CanCommit на все узлы, вовлеченные в транзакцию
  - Когда узел получает сообщение CanCommit он отвечает координатору Yes или No. Перед тем, как ответить Yes, узел выполняет подготовку к фиксации изменений, записывая объекты в долговременной памяти. В случае ответа No (или в том случае, когда при подготовке ответа Yes произошла ошибка) узел немедленно откатывает свои изменения
- Фаза 2
  - Координатор собирает ответы от всех узлов
    - Если все узлы ответили Yes, координатор фиксирует транзакцию и рассылает всем узлам сообщения doCommit
    - В противном случае, координатор откатывает транзакцию, рассылая всем узлам сообщение doAbort
  - Узлы, которые ответили Yes на запрос CanCommit ожидают сообщения doCommit или doAbort от координатора. При получении одного из этих сообщений, узел выполняет фиксацию или откат своих изменений, после чего пересылает координатору сообщение HaveEnded

Блокировки в распределенных системах

- В распределенных системах блокировки являются механизмом борьбы с нежелательными эффектами параллельной обработки
- Для их поддержки часто вводится диспетчер блокировок, который управляет блокировками на уровне системы
- Разрешение тупиков может быть реализовано методом анализа графа ожидания ресурсов и применением одного из рассмотренных ранее алгоритмов

Репликация данных

- Мотивация
  - Требование повышения производительности
  - Требование повышения надежности системы путем резервного копирования части критичных данных

Пассивная (primary-backup) репликация

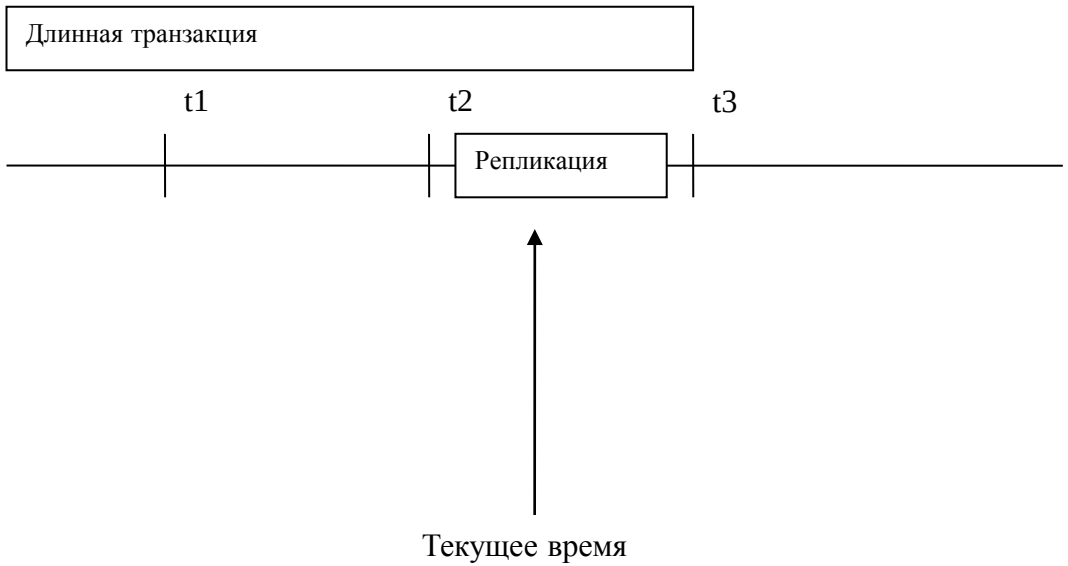
- Описание: изменение данных допускается только на одном узле (данные, размещенные на этом узле, называются мастер-копией), а на всех других узлах данные доступны только для чтения
- Применение: может использоваться как для резервного копирования данных, так и в большинстве информационных систем, в которых операции изменения данных сравнительно редки по отношению к операциям извлечения данных (при этом извлекаются данные большого объема) и могут быть сведены к операциям над мастер-копией

Пассивная репликация: реализация

- Весь массив данных по некоторому регламенту копируется на удаленные узлы. При этом подходе предполагается, что данные изменяются нечасто и на удаленных узлах не важна их высокая актуальность
- Второй подход называется репликацией на временных метках (timestamp-replication). Он состоит в том, что любая транзакция, изменяющая данные, изменяет также и временную метку (timestamp) этих данных. Таким образом, у данных, которые изменялись позднее, будет большая временная метка, чем у данных, которые изменялись раньше. Зная время последнего удачного обмена данными, менеджер репликаций «снимает» все данные, измененные позднее и передает на удаленный узел только их .
- Третий подход: перенос не данных, а журналов изменений

Пассивная репликация: замечания

- Процесс, «снимающий» измененные данные **должен** работать на уровне изоляции Serializable



  - Нужны механизмы борьбы

Активная репликация с блокировкой

- Описание: главное отличие от пассивной репликации состоит в том, что более чем один узел системы может претендовать на то, чтобы изменить объект
- Реализация: предполагается, что прежде чем изменять объект, узел запрашивает разрешение на это у центрального координатора (фактически это является блокировкой объекта, поскольку как только такое разрешение получено, остальные запросы на этот же объект координатором будут отвергаться). После того, как объект изменен, это изменение реплицируется на все узлы, которые потенциально могут изменять этот объект и только после этого блокировка объекта снимается

Активная репликация без блокировки

- Описание: рассматривается система, в которой несколько узлов могут одновременно выполнять операции над одним и тем же объектом, не блокируя его и не дожидаясь репликаций о других изменениях
- Ограничения: при применении такого метода возможны разнообразные эффекты типа «потерянного обновления» и т.д. Возможен эффект невозможности приема репликаций - например, мы разрешаем ведение справочника покупателей на двух разных узлах, при этом накладываем ограничение, что название покупателя должно быть уникально в пределах справочника
- Применение: системы, где нет редактирования сущностей – только добавление (большинство банковских платежных систем)

Итоги

- Для реализации параллельного доступа к данным необходимо применять специальные подходы
  - Использовать механизмы блокировки
  - Использовать механизм транзакций
- При использовании транзакций следует помнить о возможных негативных эффектах
  - При выборе уровня изоляции следует руководствоваться разумностью и достаточностью
- Существуют реализации систем, поддерживающих распределенные транзакции
- Для обеспечения отказоустойчивости и повышения производительности может применяться механизм репликации данных

О проекте

Целью проекта является создание национальной системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения.

Задачами по проекту являются:

Задача 1. Создание сети научно-образовательных центров суперкомпьютерных технологий (НОЦ СКТ).

Задача 2. Разработка учебно-методического обеспечения системы подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 3. Реализация образовательных программ подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 4. Развитие интеграции фундаментальных и прикладных исследований и образования в области суперкомпьютерных технологий. Обеспечение взаимодействия с РАН, промышленностью, бизнесом.

Задача 5. Расширение международного сотрудничества в создании системы суперкомпьютерного образования.

Задача 6. Разработка и реализации системы информационного обеспечения общества о достижениях в области суперкомпьютерных технологий.

См. http://www/hpc-education.ru