Parallel discrete event simulation Basics

https://www.eg.bucknell.edu/~xmeng/Course/CS6337/Note/master/node20.html

Основы

Параллельное моделирование дискретных событий (PDES) относится к выполнению одной программы моделирования дискретных событий на параллельном компьютере.

Модель моделирования дискретных событий предполагает, что моделируемая система меняет состояние только в дискретные моменты моделируемого времени.

Имитационная модель переходит из одного состояния в другое при возникновении события .

В модели реальной системы многие вещи (события) могут происходить примерно в одно и то же время, но лишь немногие происходят в один и тот же момент. Кроме того, у них нет регулярного интервала между событиями.

Асинхронные системы, в которых события не синхронизируются по глобальным часам.

Возможный механизм PDES заключается в использовании синхронного выполнения с использованием глобальных часов моделирования между многими процессорами. На каждом шаге моделируемого времени проверяются списки событий на разных процессорах и выполняются события, подлежащие вовремя.

Этот подход работает очень плохо, поскольку очень немногие события происходят в одно и то же время.

Требуется одновременное выполнение событий в разные моменты моделируемого времени! Это приводит к интересным проблемам синхронизации, которые лежат в основе проблемы PDES.

В последовательном моделировании обычно используются три структуры данных:

1. переменные состояния , которые описывают состояние системы,
2. список событий , содержащий все ожидающие события, которые были запланированы, но еще не вступили в силу, и
3. глобальная переменная часов , обозначающая, насколько далеко продвинулась симуляция.

В этой парадигме выполнения крайне важно всегда выбирать наименьшее событие с меткой времени из списка событий в качестве события, которое будет обрабатываться следующим.

Если событие с большей меткой времени было выполнено раньше события меньшего размера, которое запланировало бы это событие с большей меткой времени, произойдет логическая ошибка. Мы называем этот тип ошибок ошибками причинности .

Пример: если событие отъезда клиента обрабатывается до события его прибытия, возникла причинно-следственная ошибка.

Самая большая возможность параллельного моделирования — это одновременная обработка событий на разных процессорах.

Типичная стратегия состоит в том, чтобы сопоставить каждый физический процесс с логическим процессом (ЛП), и каждый ЛП выполняет моделирование в своем собственном темпе.

Пример: если нам нужно смоделировать заправочную станцию ​​с двумя независимыми обслуживающими персоналами, они, естественно, образуют два LP, каждый из которых может выполнять собственное моделирование.

Можно гарантировать отсутствие ошибок причинно-следственной связи, если придерживаться следующего ограничения:

**Ограничение локальной причинности (Local Causality Constraint)**

Моделирование дискретных событий, состоящее из логических процессов (LP), которые взаимодействуют исключительно посредством обмена сообщениями с метками времени, подчиняется ограничению локальной причинности тогда и только тогда, когда каждый LP обрабатывает события в неубывающем порядке меток времени.

Вышеупомянутый LCC по существу говорит, что если события обрабатываются в неубывающем порядке временных меток, то мы говорим, что они подчиняются ограничению причинности; если события не обрабатываются в неубывающем порядке временных меток, то мы говорим, что они не подчиняются ограничению причинности.

Соблюдение этого ограничения достаточно, хотя и не всегда необходимо, чтобы гарантировать отсутствие ошибок причинно-следственной связи. Другими словами, нарушение ограничения причинности не всегда может привести к ошибке моделирования. Это связано с тем, что два события в пределах одного ЛП могут быть независимыми друг от друга, и в этом случае их обработка вне последовательности временных меток не приводит к ошибке причинно-следственной связи.

Пример: в супермаркете есть стойка обслуживания и несколько кассовых очередей. Клиентов, проходящих через службу поддержки, можно считать независимыми от тех, кто проходит через кассы. Если мы обработаем их вне порядка временных меток, это не приведет к ошибке причинно-следственной связи.

С другой стороны, если в одной и той же строке оформления заказа для одного покупателя мы обрабатываем событие упаковки до события оформления заказа , возникает ошибка причинно-следственной связи.

Задача PDES заключается в одновременном выполнении ЛП и получении правильных результатов моделирования.

Механизмы PDES можно разделить на две категории: консервативные и оптимистические .

Консервативные подходы строго исключают возможность какой-либо ошибки причинно-следственной связи. Обычно эти подходы основаны на некоторой стратегии, позволяющей определить, безопасно ли обрабатывать событие.

Оптимистические подходы используют стратегию обнаружения и восстановления : ошибки причинно-следственной связи допускаются, но обнаруживаются, и для восстановления ошибок запускается механизм отката .

**Консервативный подход**

* Ключом к консервативному PDES является отсутствие причинно-следственных ошибок перед обработкой события. Существуют разные стратегии.
* Статически укажите ссылки, указывающие, какие процессы могут взаимодействовать с какими другими процессами. LP может отправлять сообщения только указанным LP.

Например: при моделировании работы нескольких аэропортов с взлетом и посадкой самолетов мы можем указать, что самолеты, вылетающие из A, могут приземляться только в B и C. Таким образом, мы имеем фиксированную связь.

* Сообщения, поступающие по каждому входящему каналу, сохраняются в порядке FIFO для обработки LP. Каждое из входящих каналов поддерживает часы, равные либо метке времени сообщения в начале очереди, если очередь не пуста, либо метке времени последнего полученного сообщения, если очередь пуста.
* LP неоднократно выбирает канал с наименьшими тактовыми частотами и, если в очереди этого канала есть сообщение, обрабатывает его.
* Если выбранная очередь пуста, процесс блокируется.
* Вышеупомянутый протокол гарантирует, что каждый процесс будет обрабатывать события только в неубывающем порядке временных меток, тем самым обеспечивая соблюдение локального ограничения причинности.
* Проблемы: если возникает цикл пустых очередей, все из которых имеют небольшое значение тактовой частоты, каждый процесс в цикле должен блокироваться, и симуляция блокируется. (См. рисунок 2 на стр. 34 статьи Фудзимото CACM 1990 года).
* Для решения этой проблемы используется *нулевое* сообщение. LP отправляет сообщения всем своим исходящим каналам по круговому принципу. Если в какой-то момент LP не имеет сообщения для определенного исходящего канала, $T_{нуль}$по этому каналу отправляется нулевое сообщение с отметкой времени. Это гарантирует, что принимающему LP не придется блокироваться, что позволяет избежать взаимоблокировки.
* Нулевое сообщение с $LP_A$меткой времени $T_{нуль}$ по существу сообщает процессу получения, что $LP_A$не будет отправлять события в будущем на другие LP с меткой времени меньше $T_{нуль}$.
* Как вычислить $T_{нуль}$? $T_{нуль}$— это минимальное значение всех временных меток входящих ссылок и временных меток первого события в собственном списке событий LP.
* Этот протокол работает правильно. Однако он генерирует большое количество нулевых сообщений, что приводит к потере времени обработки.
* Большой объем работы был проделан для улучшения эффективности консервативного подхода к PDES.

**Оптимистические механизмы**

* Оптимистический механизм позволяет всем ЛП продолжить работу, поскольку существует вероятность того, что ошибка причинно-следственной связи не возникнет.
* Если на более позднем этапе обнаруживается причинно-следственная ошибка, происходит процесс восстановления, отменяющий последствия обработанных событий (откат).
* Событие, вызывающее откат, называется *отстающим* .
* Событие может выполнить две вещи, которые необходимо откатить:
  1. он может изменить состояние логического процесса;
  2. он может отправлять сообщения о событиях другим процессам
* Откат состояния осуществляется путем периодического сохранения состояния процесса и восстановления старого вектора состояния при откате. Состояние необходимо откатить до времени моделирования, равного или меньшего времени *отстающего* .
* «Отмена отправки» ранее отправленного сообщения осуществляется путем отправки негативного сообщения или *антисообщения* , которое уничтожает исходное сообщение, когда оно достигает пункта назначения.
* Если процесс получает антисообщение, соответствующее уже обработанному положительному сообщению, то это антисообщение становится отстающим, вызывая откат на этом LP.
* Рекурсивное повторение этой процедуры позволяет в конечном итоге устранить все последствия ошибочных вычислений. Можно доказать, что этот процесс сходится и всегда прогрессирует при определенных условиях.
* В основе оптимистического подхода лежит *виртуальное время* , которое является одним из важнейших понятий в распределенных вычислениях.
  1. Система *виртуального времени* — это распределенная система, работающая в координации с воображаемыми *виртуальными часами* , которые отсчитывают *виртуальное время* .
  2. Виртуальное время само по себе представляет собой глобальную, одномерную, временную систему координат, налагаемую на распределенные вычисления.
     + Он используется для измерения прогресса вычислений и определения синхронизации.
     + Это может иметь или не иметь связь с реальным временем.
     + Это реальное положительное значение, полностью упорядоченное по $<$.
  3. Системы виртуального времени подчиняются двум фундаментальным правилам.
     + Виртуальное время отправки каждого сообщения должно быть меньше виртуального времени его получения.
     + Виртуальное время каждого события в процессе должно быть меньше виртуального времени следующего события в этом процессе.

Эти правила в точности соответствуют условиям часов Лампорта (известное соотношение «происходит-перед»).

* 1. Можно сформулировать основное ограничение на реализацию виртуального времени: *если событие A вызывает событие B, то выполнение A и B должно быть запланировано в реальном времени, чтобы A завершилось до начала B.*

Это означает, что если A не вызывает B, B может быть выполнено раньше A в реальном времени, даже если логическое время B наступает после времени A.

* 1. Одной из реализаций виртуального времени является система Time Warp.
     + Для корректной реализации виртуального времени необходимо и достаточно, чтобы *в каждом процессе* сообщения обрабатывались в порядке временных меток.
     + Структура представления времени выполнения.
       - Уникальное *имя процесса* .
       - Локальные *виртуальные часы* (LVT), которые должны быть совместимы с *глобальным виртуальным временем* (GVT), но не обязательно должны иметь то же значение, что и GVT.
       - Состояние *,* которое представляет собой набор переменных.
       - Очередь *состояний* , содержащая сохраненные копии последних состояний процесса.
       - Входная *очередь,* содержащая все недавно поступившие сообщения, отсортированные по виртуальному времени получения.
       - Очередь *вывода* , содержащая все отрицательные копии сообщений, недавно отправленных этим процессом, в виртуальном порядке времени отправки. Они будут использоваться в случае отката.
     + Для иллюстрации см. рисунки 1 и 2 в статье Джефферсона 1985 года.
     + Глобальное виртуальное время: GVT в реальном времени *r* — это минимум из (1) всех локальных виртуальных времен во всех локальных виртуальных часах в момент времени r и (2) виртуальных времен отправки всех сообщений, находящихся в процессе перехода.
       - ГВТ никогда не уменьшается.
       - GVT может служить основой виртуального времени, к которому любой процесс может когда-либо снова вернуться.
       - GVT можно рассматривать как движущийся *горизонт обязательств* : любое событие, виртуальное время которого меньше GVT, не может быть отменено и может быть удалено из системы.

Отсюда и название « *Искажение времени»* .

* + - * Если (1) каждое событие завершается нормально, и (2) сообщения доставляются надежно, и (3) планировщик не откладывает на неопределенный срок выполнение самого отстающего процесса, и (4) имеется достаточно памяти, то *GVT в конечном итоге должен увеличиться.* .
  1. Примеры систем виртуального времени включают распределенное моделирование дискретных событий, управление параллельным доступом к распределенным базам данных, связь по виртуальным каналам.
  2. Виртуальное время имеет аналогию с виртуальной памятью в управлении памятью.
     + Виртуальное адресное пространство страницы — это ее пространственная координата; виртуальное время события является его временной координатой.
     + Страница, находящаяся в основной памяти в момент времени *t* , аналогична событию с виртуальным временем в будущем процесса *x* , где к странице можно получить доступ в будущем, событие будет обработано в будущем.
     + Страница из памяти в момент времени *t* аналогична событию в настоящем или прошлом процесса *x* .
     + Доступ к странице в памяти относительно недорог, но доступ к странице вне памяти в момент времени *t* очень дорог ( *ошибка страницы* ); аналогично отправка сообщения, которое поступает в виртуальное будущее принимающего процесса, обходится относительно недорого, тогда как отправка сообщения в его виртуальное прошлое вызывает очень дорогостоящий *временной сбой* , то есть откат.
     + В системе виртуальной памяти экономически эффективно только выполнение программ, подчиняющихся принципу пространственной локальности, так что большая часть обращений к памяти осуществляется к страницам, уже находящимся в памяти, а ошибки страниц относительно редки. Аналогично, в системе виртуального времени экономически эффективно запускать программы, которые подчиняются принципу временной локальности, то есть большинство сообщений поступает в виртуальное будущее процессов назначения, поэтому ошибки времени относительно редки.
     + Термин «отображение памяти» относится к преобразованию виртуальных адресов в реальный адрес. Мы могли бы использовать *отображение времени* для обозначения отображения виртуального времени в реальное время. Один и тот же виртуальный адрес может быть сопоставлен с разными физическими адресами в памяти в разное время, и аналогичным образом одно и то же виртуальное время может быть сопоставлено (запланировано) с разным реальным временем в разных местах.
     + Единственными приемлемыми картами памяти являются функции «один к одному», поскольку они сохраняют *различимость* , сопоставляя разные виртуальные адреса (страницы) с разными реальными адресами (фреймами). В любой момент некоторые виртуальные адреса могут быть не сопоставлены, поскольку они относятся к страницам, которых нет в памяти. Аналогично, единственными приемлемыми картами времени являются строго возрастающая функция, поскольку они сохраняют *порядок* , отображая отдельные виртуальные времена в отдельные реальные времена. В любом месте некоторые события могут быть не отображены (еще не запланированы), поскольку они происходят в локальном будущем.
     + Для процесса, работающего в виртуальной памяти, частоту ошибок страниц обычно можно снизить, увеличив количество страниц, которые разрешено иметь в основной памяти. Точно так же частота ошибок во времени для процесса, работающего в виртуальном времени, обычно может быть уменьшена за счет увеличения количества событий, происходящих в ближайшем будущем. По сути, большее взаимодействие между LP снижает вероятность ошибки времени. Но это может замедлить ход моделирования.
     + Если процесс может иметь достаточно страниц в памяти, частота ошибок страниц может быть уменьшена до нуля; в целом это нежелательно, поскольку приводит к неэффективному использованию основной памяти. Аналогично, если процесс имеет достаточно большое количество событий, которые можно обработать за короткий период времени, его частота ошибок по времени может быть снижена до нуля; это тоже нежелательно, поскольку тогда этот процесс становится узким местом, сдерживая развитие GVT, что приводит к неэффективному использованию реального времени (другим процессам приходится ждать).
* Расчет ГВТ. Существует много разных способов расчета GVT.
  1. Централизованный расчет: каждый LP отправляет свой LVT централизованному менеджеру, минимум из них — это рассчитанный GVT.
  2. Распределенный расчет: LP периодически транслируют свои LVT. Каждый LP вычисляет GVT как минимум из всех LVT (требуется неблокирующее чтение).
* Расстояние отката и расстояние пересылки (модель случайного блуждания)
  1. Откат сходится (частичное упорядочение событий, случаи для событий с одинаковой и разной отметкой времени)
* Управление памятью: все проблемы связаны с памятью и скоростью.
  1. Сохранение чередующегося состояния (как часто?) Вместо сохранения каждого отдельного события ввода/вывода стратегия чередующегося сохранения состояния сохраняет только моментальный снимок состояния в определенное время. Когда необходим откат, симуляция откатывается до точки, в которой состояние сохраняется.
  2. Коллекция ископаемых: события, значение тактовой частоты которых меньше GVT, никогда больше не будут использоваться, их удаление может восстановить часть памяти.
     + ленивый: не собирать, пока не запустим нашу память
     + агрессивный: сбор через фиксированный интервал или при определенном событии.
  3. Возвращать необработанные сообщения

Когда памяти не хватает, можно вернуть необработанные входные события обратно отправителю, чтобы освободить часть памяти. Те из них, у которых самые высокие отметки времени, должны быть возвращены в первую очередь, поскольку они с меньшей вероятностью окажут влияние на систему.

* 1. Протокол Гафни: вместо того, чтобы всегда возвращать необработанное сообщение, освобождается память в соответствии с характером деятельности.
     + Если причиной является выходное событие, отправляется соответствующее антисобытие. Локальное моделирование восстанавливается в предыдущее состояние.
     + Если причиной является входное событие, выполните обычное возвратное сообщение.
     + Если причиной является сохранение состояния, отбросьте этот набор переменных состояния. При необходимости их можно пересчитать позже.
     + Искусственный откат: при отсутствии отставшего откат вызывается искусственно для освобождения памяти.
* Коммуникации в ПДЕС
  1. Накладные расходы на связь
  2. Разница между параллельным моделированием (общая физическая память) и распределенным моделированием (передача сообщений)
  3. В распределенном моделировании: меньшее количество коммуникаций уменьшит общую задержку и может вызвать больший откат.
  4. Учитывается относительное значение: т.е. соотношение времени вычислений и времени связи.
* Модели производительности:
  1. Работа Чена: унификация модели производительности для влияния памяти на распределение сообщений, откат, развитие GVT в среде общей памяти с использованием модели процессов Маркова.
  2. Гупта, Акилдиз и Фудзимото: модель производительности для однородной и гетерогенной многопроцессорной среды с общей памятью
  3. Кляйнрок и Никол по отдельности разработали некоторые пределы производительности системы Time Warp.
  4. Работа Любачевского, Вайса и Шварца: среднее время $\tau$завершения моделирования системы с *N* узлами и *R* событиями в PRAM *p* -процессора удовлетворяет

\begin{displaymath}\tau ~= ~ O \left( \frac{R}{p} + \frac{R}{N} log~p \right) \end{displaymath}