ТЕОРИЯ И ПРАКТИКА МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Тема 12

Консенсус в системе со сбоями. Теорема FLP о невозможности.

Д.ф.-м.н., профессор А.Г. Тормасов
Базовая кафедра «Теоретическая и Прикладная Информатика», МФТИ

Тема

- Консенсус в системе со сбоями
- Один из самых фундаментальных и достаточно неожиданных теоретических результатов, полученных в современной науке о вычислительных системах.
- Теорема Фишера, Линч и Патерсона о невозможности консенсуса в системе со сбоями (FLP impossibility)
- Следствия теоремы

свобода от ожидания

- Определение подразумевает «устойчивость ко сбоям»
  - Алгоритм, вне зависимости от поведения соседей, должен завершиться за конечное число шагов
  - Даже если сосед замедлился, застрял или умер
- Рассмотрим утверждение, касающееся работы таких систем «со сбоями» – в обобщенном смысле

асинхронная система

- Полностью асинхронная система
  - нет никаких предположений о последовательности операций
  - рассматривается в работе Fischer, Lynch, и Paterson

асинхронная система

- Потоки обмениваются сообщениями через буфер
  - send(p,m) – помещает сообщение m, адресованное p, в буфер
    - обычно помещается пара (p,m) – полная характеристика *события*
  - receive(p) – удаляет из буфера сообщение для p, и возвращает m
    - может вернуть предопределенное значение null – нет ничего
- нет обязательств системы доставки сообщения по порядку их доставки
- нет обязательств даже по их присутствию в буфере
  - может вернуться null даже если пара (p,m) присутствует в буфере

FLP impossibility

Теорема.

Система соисполняемых потоков, обменивающихся сообщениями, в которой потоки могут быть неограниченно задержаны (сбоить), или сообщения могут быть переставлены, не может достигнуть консенсуса.

Идея

если среди коммуницирующих объектов один завис на неопределенное время (по любой причине, например, из-за сбоя канала, собственного сбоя и т.д.), то у нас нет никакого основания думать, что все участвующие в консенсусе «дождутся» неработающего объекта, чтобы принять действительно правильное решение за ограниченное время.

FLP impossibility

Доказательство

оригинальное доказательство доказывает существование последовательность событий (обмена сообщениями), которое никогда не ведет к консенсусу. То есть, из некого начального бивалентного состояния можно сделать ход в другое бивалентное состояние, и всегда может существовать новое бивалентное состояние, куда мы можем перейдти (и, собственно, доказыватеся что оно всегда существует).

Рассмотрим более конструктивное доказательство, которое показывает, КАК можно найдти тот самый «плохой» путь, «ломающий» асинхронную систему (предложено в 2004 году Фольцером).

FLP impossibility…

Конфигурация c

- состоит из вектора состояний процессов, соотнесенного с идентификатором каждого процесса (P->S) и конечного мультинабора $msgs_c$ сообщений, выполняемых в c (состояние процессов + содержимое буферов сообщений).

Событие (p,m) разрешено в c если

- $msgs_c$ содержит в себе (p,m)

c-(p,m)->c’ – последующая конфигурация через событие (p,m)

- Пусть c’ есть последующая за c конфигурация, если она получается из c удалением (p,m) из $msgs_c$, изменением состояния p и добавлением к $msgs_c$ набора состояний в соответствии с возможностями p в новом состоянии p.

FLP impossibility…

$\sigma = c_0, x_1, c_1, x_2, \ldots$ - последовательность исполнения

- Конечная или бесконечная последовательность переходов конфигураций c_i, такая, что для любого i c_i-(x_{i+1})->c_{i+i}

c => c’ – c’ достижимо из c

- если существует последовательность исполнения (возможно, пустая), начинающаяся в c и заканчивающаяся в c’

c =Q=> c’ – c’ достижимо из c на Q

- Q – подмножество процессов в последовательности исполнения

c =-Q=> c’ – c’ достижимо из c на не-Q

- Q – подмножество процессов, НЕ участвующих в последовательности исполнения

FLP impossibility…

Лемма о бриллианте («коммутативность» последовательностей исполнений)
если $c =Q=> c_1$ и $c =-Q=> c_2$, то существует c' такой, что
$c_1 =-Q=> c'$ и $c_2 =Q=> c'$

- Конфигурация достижима, если существует σ- последовательность исполнения из некоторого начального значения, встречающаяся в σ.
- Процесс p является корректным в σ, если для любого i, разрешающего c_i, существует $j > i$ такая, что x_j есть шаг p (отсутствие сбоев в процессе).
- Последовательность исполнения σ честная, если для каждого сообщения m существует корректный в σ процесс p, и для каждого события (p,m), разрешенного в c_i, существует $j>i$ такой, что $x_j = (p,m)$.

FLP impossibility…

Честная последовательность исполнения σ является t-допускающей ($t \leq n$), если существует не более t процессов, не являющихся корректными в σ.

Алгоритм решает задачу о *t-устойчивом консенсусе*, если каждая t-допускающая последовательность удовлетворяет условиям консенсуса (непротиворечивости, корректности и завершенности).

- Пусть для всех процессов есть начальное состояние с 1 сообщением, и все каналы (передачи сообщений) пусты.
- Назовем «принятием v» ситуацию, когда есть процесс, который принимает значение v в последовательности исполнения, а конфигурацию «v-решенной», если есть «выход» из нее с решением.

FLP impossibility…

Алгоритм решает задачу о t-устойчивом псевдо-консенсусе для $t \leq n$, если

- каждая t-допускающая последовательность удоволетворяет условиям консенсуса о непротиворечивости и корректности
- для каждой достижимой конфигурации c и каждом наборе процессов $Q \subseteq P$ количеством более или равном n-t, существует конфигурация c’, являющаяся v-решенной для некоторых значений v c =Q=>c’.

- Существует алгоритм, решающий эту задачу в нашей модели тогда и только тогда, когда $n \geq 2t+1$

Идея доказательства…

- Очевидно, что каждый алгоритм решения задачи t-устойчивого консенсуса является алгоритмом t-устойчивого псевдо-консенсуса.
- Если каждый 1-устойчивый алгоритм псевдоконсенсуса имеет имеет допустимый процесс исполнения, который никогда не принимает решение, то не существует и 1-устойчивого алгоритма консенсуса.

FLP impossibility…

- Пусть c есть достижимая конфигурация и p – процесс. Скажем, что значение v ∈ {0,1} есть *p-игнорирующее решение c*, если существует v-решенная конфигурация c’ такая, что c =-p=>c’.

- Обозначим набор всех *p-игнорирующих решений* как val(p,c).

- достижимая конфигурация c является *v-равной*, если val(p,c)={v} для всех p и *неравной* если она не является ни 0-равной, ни 1-равной.

v-равность

- понятие *v-равности* является более слабым, чем понятие бивалентности
- Конфигурация бивалентна, если 0-решающая, так же как и 1-решающая конфигурации достижимы из нее.
- Из *v-равности* следует бивалентность, но не наоборот, бивалетность не означает *v-равности*.

Лемма

- Лемма
  пусть c – достижимая конфигурация.
  1. Для каждого процесса p: val(p,c) ≠ ∅.
  2. Если c есть v-решенная конфигурация, то она также v-равная

Лемма

Лемма
пусть c =(p,m)=>c’ и q есть процесс.

1. $p \neq q$ влечет $val(q,c') \subseteq val(q,c)$
2. $p = q$ влечет $val(q,c) \subseteq val(q,c')$
3. $val(q,c) = \{0\}$ влечет $val(q,c') \neq \{1\}$.

Доказательство
1 – следует из определения.
2 – пусть v принадлежит val(p,c). Тогда существует v-решенная конфигурация c_v такая, что c =-p=> c_v. Из леммы о бриллианте следует существование c’’ такая, что c_v =p=> c’’ и c’ =-p=> c’’. Из первой части следует, что c’’ есть v-решенная конфигурация, и вместе с последним утверждением, $v \in val(p,c')$.
3 – следует непосредственно из 1 и 2.

Лемма

Лемма
существует начальная конфигурация, являющаяся неравномерной.

Доказательство
пусть $P=\{p_0,...,p_{n-1}\}$ и c_i обозначает конфигурацию, где процесс p_j имеет вход 1 тогда и только тогда, когда $j<i$ для $i=0,...,n$ (c_i и c_{i+1} отличаются только входом p_i).

Тогда c_0 есть 0-равномерная, и c_n есть 1-равномерная. Следовательно, существует индекс j, такой, что c_j есть 0-равномерная, а c_{j+1} – нет. Так как c_j есть 0-равномерная конфигурация, то $0 \in val(p_j,c_j)$. Следовательно, существует 0-решенная конфигурация c, такая, что $c_j = -p_j => c$. Так как c_j и c_{j+1} отличаются только входом p_j, то $c_{j+1} = -p_j => c$ и $0 \in val(p_j,c_{j+1})$. Отсюда следует, что c_{j+1} не является и 1-решенным, то есть, по определению, эта конфигурация неравномерна.

Лемма

Лемма
для каждой неравномерной конфигурации c и каждого процесса p существует конфигурация c’ такая, что c=>c’ и val(p,c)={0,1}.

Доказательство
Если val(p,c)={0,1}, то утверждение доказано. Если нет, то пусть val(p,c)={0} (без потери общности). Так как c неравномерна, то существует процесс q такой, что $1 \in$ val(q,c).

Следовательно, существует 1-решенная конфигурация c_1, такая, что c =>c_1. Так как c_1есть 1-решенная, то val(p,c1) = {1}. Из 3 утверждения доказанной леммы, существует c’ такая, что c=>c’=>c_1 и val(p,c’)={0,1} – что и требовалось доказать.

Теорема о невозможности

Пусть есть конечная последовательности исполнения $\sigma = c_0, x_1, c_1, x_2, \ldots, c_k$ и (p,m) разрешено в c_k. Время разрешения для (p,m) есть самое малое значение позиции l в σ, и c_j разрешает (p,m) для всех j=l,..,k и $x_j \neq$ (p,m) для всех j = l+1,...,k.

Теорема
Каждый 1-устойчивый алгоритма псевдо-консенсуса имеет 0-допускающую последовательность исполнения, которая не принимает решения.

Доказательство.
Построим такую последовательность, которая не принимает решения. Начнем с начальной неравной конфигурации (существующей в соответствии с доказанной выше леммой). Затем, будем повторять следующие действия:

Теорема о невозможности

Возьмем разрешенный шаг (p,m) с минимальным временем разрешения

В соответствии с доказанной леммой, расширимся до конфигурации с такой, что val(p,c) = {0,1}. Эта конфигурация с является неравной.

Если (p,m) разрешено в c, то пусть оно произойдет и мы перейдем в c'.

Из леммы, val(p,c')={0,1} и c' является неравным.

Таким образом, мы получили честную последовательность исполнения, где все процессы корректны и которая всегда неравная, то есть не принимает решения!

Еще раз: если v не является p-игнорирующим решением, то движемся в сторону решения по v, пока оба значения не станут p-игнорирующими – а это нам гарантирует, что после события (p,m) оба значения могут встретиться.

Замечание

Библиотеки параллельного программирования часто используют очереди

- Fastflow
  - Вычисления на сети вычислителей (или конвейере)
  - Для их соединения используется SPSC очередь FIFO (lock/interlock free), использующая только регистры чтения-записи!
- Intel TBB и другие
  - Конвейеризация и соединение объектов MPMC FIFO – один из основных способов

- То есть мы организовали «композицию» объектов типа регистры для решения универсальных проблем программирования? А как же число конесенуса (1 для регистров, 2 для очереди и тд?)

Замечание

Чем отличается FIFO общего типа (MPMC) от описанного? Почему можно использовать их для решения общих задач?

MPMC отличается от SPSC дополнительными требованиями в последовательности сообщений, для реализации которых испольуются дополнительные атомарные команды с более высоким числом консенсуса

используемые атомарные регистры не обладают некоторыми коммутативными свойствами буферов сообщений

- так как в FIFO все сообщения упорядочены последовательностью их добавления, а в буферах сообщений могут быть не упорядочены
- FLP - теорема, и утверждение о числе консенсуса FIFO равное 2, относятся к разным случаям.

Замечание

Пусть объект консенсуса реализован только через регистры как свободный от зависаний или свободный от взаимных исключений объект взаимной блокировки – то есть, получился мутекс (возможно ли это?)

- Реализация гарантирует только условный прогресс
- ОС должна гарантировать выход из критической секции для глобального прогресса

- То же самое можно сделать через свободные объекты (как упражнение)
  - Решить проблему консенсуса свободными объектами невозможно.
  - А где доказательство невозможности будет «ломаться» если у нас есть «оракул», тормозящий некоторые потоки так, чтобы наблюдался прогресс?

Замечание

- Мы рассматриваем только детерминистические объекты
  - Их состояние полностью определяет ответ любой применяемой операции и новое состояние объекта после нее
  - Пример «естественного» недетерменитстического объекта – очередь с приоритетами и возможностью размещения одинаковых значений
    - несколько значений имеют один приоритет – естественно разрешить взять любой из них
  - То есть работа с недетерменистическими объектами часто необходима на практике

надежность иерархии

- Понятие надежности иерархии заключается в 100% невозможности создать объект более высокой иерархии из любых объектов низшей иерархии
- Существует доказательство надежности иерархии консенсуса для детерменистических объектов
- Но, существуют разные варианты недетерменистических расширений, для которых доказано, что иерархия является ненадежной (условно или безусловно)
- Любые модификации требований к объекту синхронизации могут вывести его из рассматриваемого класса и предоставить дополнительные возможности разработчику
  - Обычно за это надо платить чем-то – сложностью, условным прогрессом и т.д.

Выводы

- Рассмотрели консенсус в системе с непредсказуемыми событиями
- Доказали его невозможность в классической постановке
  - Для доказательства ввели серию понятий, в частности, более слабое, чем бивалентность, понятие неравности.
- Практическое использование базовых определений сложно, и часто используются всевозможные «послабления»
  - Любое изменение требований обычно выводит задачу в область, где иерархия может быть ненадежной
    - особенно, связанное с внесением асинхронности в систему
  - Есть много систем, практически использующих неблокирующие примитивы для построения реально работающих систем общего назначения
    - Анализ таких систем требует аккуратности в рассмотрении требований и условий

(c) А. Тормасов, 2010-11 г.

Базовая кафедра «Теоретическая и Прикладная Информатика» ФУПМ МФТИ
tor@ crec .mipt .ru_