Распределенные системы

Оглавление

0.1 Формализм. Логические часы Лампорта (свойства и алгоритм)

Кратко опишем используемые далее обозначения.

Обозначение	Объект
$P, Q, R, \ldots \in \mathbb{P}$	Процессы
$a, b, c, \ldots \in \mathbb{E}$	События в процессах `proc(e)` $\in \mathbb{P}$
$m \in \mathbb{M}$	Сообщения, `snd(m)`, `rcv(m)` $\in \mathbb{E}$.

Таблица 1: Общие обозначения

Определение. Отношение *Произошло-до* ($\rightarrow$) – минимальный строгий частичный порядок на $\mathbb{E} \times \mathbb{E}$ такой, что

- $e \rightarrow f$, если e, f в одном процессе и e идет перед f.
- Если m – сообщение, то `snd(m)` $\rightarrow$ `rcv(m)`.

Определение. *Логические часы*. Определим функцию $C : \mathbb{E} \rightarrow N$ так, чтобы

$$\forall e, f \in \mathbb{E} \; e \rightarrow f \implies C(e) < C(f).$$

Алгоритм. (Логические часы Лампорта)

- Каждый процесс хранит счетчик.
- Перед посылкой процесс увеличивает счетчик на единицу.
- При посылке дополнительно посылается счетчик.
- Получатель обновляет свое время следующим образом:

$$C \leftarrow \max(C, C_r) + 1.$$

Свойства логических часов Лампорта:

- Время события не уникально.
- Являются логическими часами в смысле определения.

0.2 Формализм. Векторные часы (свойства и алгоритм)

Определение. *Векторные часы*. Определим функцию $VC : \mathbb{E} \to N^k$ так, чтобы

$$\forall e, f \in \mathbb{E}\ e \to f \iff VC(e) < VC(f).$$

Сравнение производится покомпонентно.

Алгоритм. (Векторное время)

- Каждый процесс хранит свой вектор-время (размер – число процессов).
- Перед посылкой сообщения процесс увеличивает свою компоненту на единицу.
- При приеме сообщение берется покомпонентный максимум:

$$VC \leftarrow \max(VC, VC_r).$$

Свойства векторного времени:

- Векторное время уникально для каждого события.
- Векторное время полностью передает отношение произошло-до.
- $$\forall e, f \in \mathbb{E}\colon\ \texttt{proc(e)} = P_i,\ \texttt{proc(f)} = P_j \Longrightarrow \left(e \to f \iff \begin{pmatrix} VC(e)_i \\ VC(e)_j \end{pmatrix} < \begin{pmatrix} VC(f)_i \\ VC(f)_j \end{pmatrix}\right).$$

0.3 Формализм. Часы с прямой зависимостью (свойства и алгоритм)

Определение.

$$e \to_d f \iff e < f \vee \exists m \in \mathbb{M}\colon\ e \leqslant \mathtt{snd(m)} \wedge \mathtt{rcv(m)} \leqslant f.$$

Определение. *Часы с прямой зависимостью*. Определим функцию $VC_d\colon \mathbb{E} \to N^k$ так, чтобы

$$\forall e, f \in \mathbb{E}\colon\ e \to_d f \iff VC_d(e) < VC_d(f).$$

Алгоритм. (Часы с прямой зависимостью)
Алгоритм полностью повторяет алгоритм для векторных часов, за исключением того, что посылается только та компонента времени, которая соотвествует процессу-отправителю.

0.4 Взаимное исключение в распределенной системе. Централизованный алгоритм.

Обозначение	Объект
CS_i	Критическая секция с номером
$Enter(CS_i)$	Вход в критическую секцию
$Exit(CS_i)$	Выход из критической секции

Таблица 2: Общие обозначения

Определение. *Взаимное исключение.* Основное требование

$$Exit(CS_i) \to Enter(CS_{i+1}).$$

Определение. *Требование прогресса:*

- Каждое желание процесса попасть в критическую секцию будет рано или поздно удовлетворено
- Может быть гарантирован тот или иной уровень честности удовлетворения желания процессов о входе в критическую секцию

Алгоритм. (Централизованный алгоритм)

- Весь процесс контролируется выделенным координатором
- Общение происходит по следующему протоколу:

Вид запроса	Действие
$request$	Запрос разрешения у координатора
ok	Одобрение координатором входа в секцию
$release$	Освобождение пользователем критической секции

Таблица 3: Виды запросов

- При входе в критическую секцию узел шлёт запрос координатору, дожидается разрешения, затем входит в критическую секцию. При завершении работы узел посылает координатору сообщения, что секция свободна. Данный алгоритм всегда требует 3 сообщения для работы с критической секцией.
- Не масштабируется из-за необходимости иметь выделенного координатора

0.5 Взаимное исключение в распределённой системе. Алгоритм Лампорта

Вид запроса	Действие
$request$	От запрашивающего ко всем другим узлам
ok	Подтверждение получения (не даёт права входа в CS)
$release$	Освобождение узлом критической секции (всем узлам)

Таблица 4: Виды запросов алгоритма Лампорта

Алгоритм. (Алгоритм Лампорта)

- Координатор отсутствует, все узлы равны
- Сообщения request и release рассылаются всем другим узлам, всего $3n - 3$ сообщения на CS
- Используются логические часы лампорта. Для установления порядка "кто раньше". Обязательно требуется порядок FIFO на сообщениях
- Все узлы хранят у себя очередь запросов
- В критическую секцию можно войти, если
  - Мой запрос первый в очереди, т.е. его время меньше времени остальных запросов (при равенстве времен порядок определяется по номеру узла, который посылается вместе с часами)
  - Получен ok от всех других узлов, т.е. они знают о вашем запросе
- Если узел хочет войти в CS, то он посылает всем другим узлам request со своими часами и id. Ждёт от всех ok. Если других запросов не поступало, либо время нашего запроса меньше времени других запросов, то входим в критическую секцию. Иначе ждем release от всех узлов, которые раньше нас в очереди.

0.6 Взаимное исключение в распределённой системе. Алгоритм Рикарда и Агравалы

Вид запроса	Действие
$request$	От запрашивающего ко всем другим узлам
ok	После выхода из критической секции

Таблица 5: Виды запросов алгоритма Рикарда и Агравалы

Алгоритм. (Алгоритм Рикарда и Агравалы)

- Оптимизация алгоритма Лампорта
- Всего $2n-2$ сообщений
- Если узел хочет войти в CS, то он шлет request всем узлам. Если узел получивший запрос не хочет войти в CS, либо его номерок запроса (в часах) больше, то он отсылает разрешение ok. Узел, который входит в CS, хранит в очереди какие ok-ответы он должен послать после выхода.

0.7 Взаимное исключение в распределённой системе. Алгоритм обедающих философов.

Определение. В частном случае ресурсы – вилки, процессы – философы, граф конфликтов – кольцо

Теорема 0.7.1. В ориентированном графе без циклов всегда есть исток

Теорема 0.7.2. Если у истока перевернуть все ребра, то граф останется ациклическим

Алгоритм. (Алгоритм обедающих философов)

- Философ владеет вилкой, если ребро в графе конфликтов исходит из его вершины
- Философ может принять пищу, если владеет обеими вилками, т.е. он исток
- После еды вилки надо отдать (ленивый способ):
  - После еды вилки помечаются грязными
  - Моем вилки и отдаём их по запросу, даже если сами хотим есть
  - Чистые вилки не отдаём, если сами хотим есть. Ожидаем все вилки, едим, отдаем, если был запрос

Алгоритм. (Обобщение алгоритма обедающих философов на произвольный граф)

- Взаимное исключение эквивалентно полному графу конфликтов (ребро между каждой парой процессов)
- При инициализации вилки раздаются в каком-то порядке (например, по порядку id процессов)

Замечание. (Результат)

- 0 сообщений на повторный заход в критическую секцию
- В худшем случае $2n-2$ сообщения
- Количество сообщений пропорционально числу желающих попасть в критическую секцию