Распределенные системы и технологии.

Проблема консенсуса

Дмитрий Юрьевич Чалый
декан факультета ИВТ,
зав. кафедрой информационных и сетевых технологий

22 мая 2016 г.

Формальная постановка задачи

- N равноправных процессов;
- каждый процесс p_i содержит:
  - входная переменная x_p: 0 или 1;
  - выходная переменная y_p: изначально b;
  - переменная y_p может быть изменена только один раз.
- проблема консенсуса: разработать протокол такой что в конце работы все процессы устанавливают y_p в 0, либо в 1.

Варианты постановки задачи

- Validity. Если все предлагают одно и то же значение, то оно и устанавливается;
- Integrity. Определяемое значение должно быть предложено определенным процессом;
- Нетривиальность. Существует хотя бы одно начальное состояние системы, которое ведет к установке всех нулей, либо всех единиц.

Сведение других проблем к консенсусу

- Идеальное определение сбоев;
- определение лидера (выбрать в точности одного лидера и все процессы об этом знают);
- . . . и многое другое!

Разрешима ли проблема консенсуса?

Синхронные распределенные системы

- Все сообщения доставляются за ограниченное время;
- значение смещения локальных часов процессов ограничено;
- каждое действие процесса происходит за ограниченное время.

Асинхронные модель систем

Нет оценок времени:

- за которое доставляются сообщения;
- на которое смещаются локальные часы процессов;
- за которое выполняются действия процессов.

Все реальные системы, включая Интернет.

Консенсус в синхронных системах

- Пусть кол-во упавших процессов ограничено f;
- алгоритм работает за $f + 1$ раунд;
- между процессами надежная передача данных;
- V_i^r — множество предложенных для установки значений, известные p_i на начало раунда r.

Консенсус в синхронных системах

- Инициализация: $V_i^0 = \varnothing, V_i^1 = \{v_i\}$;
- в каждом раунде k:
  - широковещательная передача $V_i^k \setminus V_i^{k-1}$;
  - $V_i^{k+1} = V_i^k$;
  - для каждого полученного V_j: $V_i^{k+1} = V_i^{k+1} \cup V_j$;
- $y_i = min(V_i)$.

Консенсус в асинхронных системах

- Неразешимая проблема в исходной постановке;
- Алгоритм Paxos: eventual liveness (когда-нибудь в будущем консенсус будет достигнут).

Paxos: общая характеристика

- Алгоритм работает по раундам;
- аналогия с выборами;
- асинхронный алгоритм: если процесс отрабатывает раунд j и ему приходит сообщение из раунда $j + 1$, то бросаем все и переходим в раунд $j + 1$;
- используем тайм-ауты.

Paxos. Фаза 1: выборы

- p_i, потенциальный лидер, выбирает уникальный id, больший чем все, которые он до сих пор видел;
- p_i отправляет id всем процессам;
- p_j ожидает получения сообщений и отвечает один раз на больший полученный id:
  - если p_j потенциальный лидер и получает больший id, то он не может быть лидером;
  - может быть несколько лидеров;
  - p_j записывают полученные id на диск.
- если процесс в предыдущем раунде выбрал значение y_i, то оно включается в ответ;
- если большинство ответило на сообщение лидера, то он им назначается;
- если большинства не получилось, начинаем следующий раунд.

Paxos. Фаза 2: предложение

- лидер отправляет значение v всем другим процессам;
- $v = v'$, если ранее каким-то процессом было предложено v';
- получение сообщения подтверждается.

Paxos. Фаза 3: установление

Если лидер получает подтверждение от большинства, то отправляет остальным информацию чтобы установить это значение.

Paxos: что может пойти нет так

- Процесс терпит крах:
  - он не входит в большинство;
  - после перезапуска восстанавливает последнее состояние с диска.
- Лидер потерпел крах/потерялись сообщения: стартуем следующий раунд;
- может быть бесконечное число раундов до обретения консенсуса: что-то не так с вашей системой.