Operating Systems

Distributed Systems

Me

December 9, 2016

Назначение распределенных систем

- Масштабируемость
  - система масштабируема, если ее легко увеличивать и тем самым увеличивать объем выполняемой ей работы.
- Производительность
  - threoughtput;
  - latency.
- Возможность переживать ошибки/доступность
  - если ваша система продолжает работать после отказа каких-то ее частей, то это хорошо;
  - при этом, зачастую, имеются ввиду какие-то типы отказов.

Модели в распределенных системах

- Модель описывает свойства и гарантии распределенной системы и условия, в которых она должна функционировать
  - свойства окружения;
  - типы возможных ошибок;
  - гарантии консистентности.

Процессы

- В данном контексте, процессы - это узлы распределенной системы, решающие совместными услилиями общую задачу
  - агенты/ноды/компьютеры/etc;
  - например, можем рассматривать разные модели ошибок для процессов:
    - мы можем считать их надежными;
    - они могут падать и, возможно, восстанавливаться после падений;
    - византийские ошибки - совершенно произвольные ошибки/злонамеренное поведение.

Взаимодействие процессов

- Один из самых очевидных способов взаимодействия - обмен сообщениями
  - зачастую это просто обмен данными по сети.
- Про обмен сообщениями полезно знать:
  - могут ли они переупорядочиваться;
  - могут ли они теряться.

Предположения о времени

- Есть, более или менее, две крайние модели, описывающие временные характеристики системы:
  - синхронная модель - время доставки сообщений ограничено и граница известна, время на обработку сообщения ограничено и известно;
  - асинхронная модель - время доставки и обработки сообщений не ограничено.

Проблема достижения консенсуса

- Процессы должны согласиться на одном значении
  - изначально у каждого процесса есть некоторое свое значение;
  - процессы должны выбрать из всех значений какое-то одно.

Требования к решению

- Согласованность
  - все корректные процессы должны согласиться на одном и том же значении.
- Целостность
  - значение должно быть предложено одним из процессов.
- Завершаемость:
  - все корректные процессы должны принять решение в конечном итоге.

Проблема двух генералов

- Есть две армии под командованием двух генералов
  - и как обычно бывает в армии - они не замышляют ничего хорошего;
  - генералы должны договориться о времени атаки на некий город.
- При этом есть ограничения:
  - генералы могут обмениваться сообщениями используя гонцов;
  - гонцов могут перехватить и сообщение не будет доставлено.

Обмен сообщениями и подтверждения

- Пусть один из генералов предложил время и отправил гонца к другому
  - может ли он считать, что время согласовано?
  - очевидно нет, т. к. сообщение могло потеряться.
- Пусть в ответ на сообщение нам отвечают подтверждением получения:
  - пусть второй генерал получил сообщение со временем и отправил ответ;
  - может ли второй генерал считать, что время атаки согласованно?

Договориться нельзя

- Допустим для задачи есть решение состоящие из обмена N сообщениями
  - последнее сообщение могло потеряться, но результат не должен измениться, т. к. корректное решение должно быть готово к потерям.
- Получается последнее сообщение не было нужно?
  - удалив его получим решение за $N - 1$ сообщение;
  - и так далее.

Каналы связи ненадежны

- В действительности каналы связи не гарантируют доставку сообщений без потерь
  - никакой протокол не поможет, если вы сожгли маршрутизатор, обрезали провода, не заплатили за Интернет, ...
  - оставляя особые обстоятельства в стороне TCP, обычно, дает неплохую надежность.
- В конечном итоге повышение надежности сводится к повторениям:
  - мы повторяем отправку сообщений много раз, пока не убедимся что оно получено.

Консенсус в асинхронных системах

- Пусть каналы связи надежны, но ограничение на время доставки сообщения отсутствуют
  - асинхронная модель считается довольно реалистичной.
- Но теперь, пусть процессы могут падать
  - без падений и потерь сообщений все тривиально;
  - предполагаем, что после падения процесс никогда не восстанавливается.

Минимальный консенсус с одним падением

- Все процессы предлагают в качестве значения 1 или 0:
  - не фундаментальное ограничение - научившись достигать консенсуса на одном бите мы сможем достигнуть консенсуса и на серии бит.
- Упасть может не более одного процесса:
  - если даже с одним падением не справимся, за большее количество браться не стоит.

Состояние процесса

- Каждый процесс обладает некоторым внутренним состоянием
  - внутреннее состояние и внешние события определяют поведение процесса;
  - внутреннее состояние процесса неизвестно никому кроме самого процесса.
- Если процесс не делает никаких шагов - внутренее состояние не изменяется
  - каждый шаг состоит из получения не более чем одного сообщения;
  - и отправки не более чем одного сообщения.

Состояние системы

- Состояние системы складывается из
  - предлагаемого значения каждого процесса;
  - внутреннего состояния каждого процесса;
  - множества отправленных, но еще не доставленных сообщений.

Валентность состояния системы

- Состояние бивалентно, если из него можно достичь как решения 0, так и решения 1
  - т. е. это состояние, в котором решение еще не принято, и может быть достигнут любой из двух возможных исходов.
- Состояние унивалентно, если из него можно достичь либо только решения 0, либо только решения 1
  - т. е. решение уже принято, хотя возможно не все об этом знают;
  - будем называть состояние i-валентным, если из него можно достичь только решения i, где i это либо 0 либо 1.

Бивалентное начальное состояние 1/2

- Начальное состояние - если ни один из процессов не делал ходов
  - очевидно они отличаются только предлагаемым значением разных процессов - это единственные входные данные, которые у нас есть.
- Существуют начальные состония приводящие как к решению 0, так и к решению 1:
  - если все предлагаемые значения равны 0 (1), то процессы должны согласиться на значениии 0 (1).

Бивалентное начальное состояние 2/2

- Более того должно существовать бивалентное начальное состояние
  - допустим противное, т. е. все начальные состояния унивалентны;
  - мы знаем, что есть как 0-валентные состояния, так и 1-валентные;
  - найдем пару таких начальных состояний, одно из которых 0-валентное, а второе 1-валентное и они отличаются только предлагаемым значением ровно одного процесса;
  - если этот процесс упадет в самом начале, то эти начальные состояния будут неразличимы, но приходят к разным решениям - противоречие.

Сохраняем бивалентное состояние 1/5

- Итак, у нас есть бивалентное состояние, назовем его , и пусть нам дан шаг e, допустимый в состоянии C
  - пусть это шаг процесса, который мы обозначим как p;
  - этот шаг останется допустимым пока p его не сделает, т. е. он не может неожиданно стать недоступным (отправленные сообщения не исчезают, пока их не получат).
- Обозначим через S множество состояний достижимых из C *без участия шага* e и *без падений*
  - обратите внимание, что процесс p может делать шаги, он не может сделать только один конкретный шаг e;
  - т. е. процесс p не упал, он просто немного ограничен, потому что некоторое сообщение долго доставляется.

Сохраняем бивалентное состояние 2/5

- Обозначим через $e(S)$ множество состояний полученных из S применением шага e
  - допустим, что все состояния в $e(S)$ унивалентны;
  - мы можем утверждать, что в $e(S)$ есть как 0-валентные состояния, так и 1-валентные (почему?);
  - т. е. есть такие состояния C_0 и C_1, что сделав шаг e из C_0 мы получаем 0-валентное состояние $C_0 e$, а сделав шаг e из C_1 мы получаем 1-валентное состояние $C_1 e$;
  - все состояния в S делятся на те, которые шаг e переводит в 0-валентное состояние и те, которые шаг e переводит в 1-валентное.

Сохраняем бивалентное состояние 3/5

- Более того, существует такой шаг e' и состояние D из S, что De - 0-валентное состояние, а $De'e$ - 1-валентное (или наоборот)
  - начальное состояние C - бивалентное, и допустим шаг e переводит его в 0-валентное;
  - есть так же последовательность шагов, которая переводит C в состояние, из которого под действием e мы приходим в 1-валентное состояние;
  - один из шагов в этой последовательности и есть наш e', а одно из состояний это и есть наш D.

Сохраняем бивалентное состояние 4/5

- Если e' и e - это шаги разных процессов
  - тогда состояния Dee' и $De'e$ неразличимы, но так как De 0-валентное состояние, то и Dee' тоже 0-валентное, но $De'e$ 1-валетное - противоречие.

Сохраняем бивалентное состояние 5/5

- Если e' и e - это шаги одного процесса p
  - но по условию один из процессов может упасть, пусть это будет p;
  - рассмотрим конечную последовательно шагов $e_1 e_2 ... e_k$, такую что процесс p в ней не участвует ни прямо ни косвенно, т. е. не только не получает сообщений, но и сообщения от p никому не доставляются;
  - с точки зрения других процессов p не отличим от упавшего процесса, т. е. последовательность $e_1 e_2 ... e_k$ должна приводить к решению;
  - $De e_1 e_2 ... e_k$ и $De' e e_1 e_2 ... e_k$ неразличимы ни для кого кроме p, а значит принятые решения должны совпадать;
  - но De 0-валентное, а $De' e$ 1-валентное - опять противоречие.

Резюме

- Что мы имеем?
  1. существует начальное бивалентное состояние;
  2. существует путь из любого бивалентного состояния в некоторое бивалентное использующий заданный шаг;
  3. т. е. мы можем построить *честную* последовательность шагов, которая никогда не приведет к решению.
- Т. е. гарантированное достижение консенсуса в асинхронной системе с падениями за ограниченное количество шагов не возможно
  - этот результат известен как FLP или, иногда, FLP Impossibility;
  - Michael J. Ficher, Nancy A. Lynch, Michael S. Paterson, 1985.

Применение консенсуса

- К сожалению задача достижения консенсуса эквивалентна многим практически значимым задачам
  - репликация данных (распределенное хранение одних данных на нескольких узлах со строгими гарантиями консистентности);
  - выбор лидера;
  - и некоторые другие.

Достижение консенсуса

- На практике существуют алгоритмы достижения консенсуса жертвующие "живостью":
  - процессы могут "мешать" друг другу, если условия не благоприятные;
  - если условия близки к синхронной модели достаточное время, то процессы достигают консенсуса;
  - примеры:
    - Paxos (Л. Лэмпорт), вероятно, самый известный и популярный;
    - Viewstamped Replication (Б. Лисков), ровесник Paxos, но не получил большой популярности, хотя описан понятнее;
    - Raft - задумывался как простая альтернатива Paxos, сравнительно новый алгоритм (2013).

Q&A