Operating Systems

Distributed File Systems

Me

April 5, 2016

План лекции

- Назначение распределенных файловых систем.
- Взаимодействие клиента и сервера. Andrew File System.
- Распределение метаданных ФС.
- Консенсус и связанные задачи.
- Ненадежный канал связи. Задача двух генералов.
- Асинхронные системы. FLP Impossibility. Paxos.

Назначение

- Распределенные ФС дублируют информацию:
  - повышение доступности (надежности хранения);
  - повышение скорости доступа.

Назначение

- Распределенные ФС дублируют информацию:
  - повышение доступности (надежности хранения);
  - повышение скорости доступа.
- Распределенные ФС объединяют узлы:
  - на одной машине можно хранить порядка 10 TB (очень грубо);
  - объединим 100 таких машин.

Andrew File System

- Andrew File System - ФС разработанная в CMU в 1980-х:
  - основная цель - масштабирование;

Andrew File System

- Andrew File System - ФС разработанная в CMU в 1980-х:
  - основная цель - масштабирование;
- Принцип работы очень простой:
  - файлы *целиком* кешируются локально;
  - если файл не в кеше запрашиваем файл с сервера - передаем полное имя файла;
  - если файл в кеше проверяем валидность копии и запрашиваем новую если нужно;
  - когда закончили обновляем версию на сервере;

Проблемы AFS

- Сервер должен для каждого клиента выполнять поиск по имени:
  - поиск может приводить к нескольким обращениям к диску;
  - если клиентов много поиск может тратить много ресурсов CPU.

Проблемы AFS

- Сервер должен для каждого клиента выполнять поиск по имени:
  - поиск может приводить к нескольким обращениям к диску;
  - если клиентов много поиск может тратить много ресурсов CPU.
- Серевер может быть загружен запросами на проверку версии файла:
  - клиенты должны проверять валидность своей копии файла;
  - даже если файл изменяется редко, проверять его валидность нужно всегда;

Решения проблем AFS

- Callback - сервер сам нотифицирует об изменениях файла:
  - клиенты не заваливают сервер запросами на проверку валидности копии;
  - сервер рассылает сообщения заинтересованным клиентам, когда копия обновилась;

Решения проблем AFS

- Callback - сервер сам нотифицирует об изменениях файла:
  - клиенты не заваливают сервер запросами на проверку валидности копии;
  - сервер рассылает сообщения заинтересованным клиентам, когда копия обновилась;
- Поиск файла по имени перекладывается на клиента:
  - клиент делает запрос к серверу на каждый каталог в имени;
  - клиент кеширует ответы от сервера и тем самым уменьшает нагрузку;
  - на каждый полученный каталог регистрируется Callback;

Финальные замечания про AFS

- AFS определяет протокол взаимодействия клиента и сервера:
  - NFS (стандарт де-факто) тоже определяет протокол;
  - протокол не приближает нас к реализации серверной части;
  - как распределять информацию по среверам?
  - как бороться с отказами на сервере?

Реализация распределенной ФС

- Необходимо решить следующие задачи:
  - как распределить данные по серверам?
  - как распределить метаданные по серверам?
  - данные и метаданные должны быть реплицированы;
    - несколько копий нужны для надежности и для скорости доступа;
    - копии должны быть согласованными, как этого добиться?

Реализация распределенной ФС

- Необходимо решить следующие задачи:
  - как распределить данные по серверам?
  - как распределить метаданные по серверам?
  - данные и метаданные должны быть реплицированы;
    - несколько копий нужны для надежности и для скорости доступа;
    - копии должны быть согласованными, как этого добиться?
    - а если сервера могут ломаться?

Метаданные

- Метаданные - данные которые описывают структуру данных:
  - взаимосвязи между файлами и каталогами;
  - атрибуты файлов и каталогов (размер, дата создания, расположение на диске);
  - права доступа (без них все гораздо проще);

Метаданные

- Метаданные - данные которые описывают структуру данных:
  - взаимосвязи между файлами и каталогами;
  - атрибуты файлов и каталогов (размер, дата создания, расположение на диске);
  - права доступа (без них все гораздо проще);
- Метаданные ФС обладают следующими особенностями:
  - небольшой объем по сравнению с данными;
  - частота использования - нельзя избежать обращения к метаданным (нужно проверить права доступа);
  - тесная зависимость - метаданные ФС образуют дерево (или DAG);

Разделение метаданных

- статическое разделение:
  - администратор распределеяет поддеревья серверам;
  - плохо адаптируется к изменяемым нагрузкам;

Разделение метаданных

- статическое разделение:
  - администратор распределеяет поддеревья серверам;
  - плохо адаптируется к изменяемым нагрузкам;
- hash-based разделение:
  - хеш идентификатора файла определяет сервер;
  - равномерно распределяет нагрузку;
  - не учитывает локальность;

Разделение метаданных

- статическое разделение:
  - администратор распределеяет поддеревья серверам;
  - плохо адаптируется к изменяемым нагрузкам;
- hash-based разделение:
  - хеш идентификатора файла определяет сервер;
  - равномерно распределяет нагрузку;
  - не учитывает локальность;
- динамическое разделение:
  - динамически назначает поддеревья серверам;
  - совсем не просто, но может распределять нагрузку равномерно;
  - учитывает локальность и адаптируется к изменениям нагрузки;

Репликация данных

- В распределенной ФС данные реплицируются:
  - если есть несколько копий данных, то читать можно из любой - больше скорость чтения;
  - не страшно потерять копию данных;

Репликация данных

- В распределенной ФС данные реплицируются:
  - если есть несколько копий данных, то читать можно из любой - больше скорость чтения;
  - не страшно потерять копию данных;
- Возникает проблема совместного обновления всех этих копий:
  - если сервера не ломаются и сообщения не теряются - то проблемы нет;
  - но сервера могут выходить из строя и возвращаться в строй;
  - обновления могут приходить параллельно от нескольких клиентов - нужна сериализация;

Консенсус

- Консенсус - теоретическая модель, к которой сводится целый класс задач:
  - репликация;
  - выбор лидера;
  - атомарный broadcast;

Консенсус

- Консенсус - теоретическая модель, к которой сводится целый класс задач:
  - репликация;
  - выбор лидера;
  - атомарный broadcast;
- Задача достижения консенсуса (неформально):
  - имеется N участников (агентов, процессоров, узлов);
  - участники предлагают некоторые значения v_i;
  - участники должны договорится и выбрать одно из предложенных v_i;

Консенсус

- Более формальная постановка задачи:
  - имеется N процессов;
  - каждый процесс описывается 3 значениями: v_i, d_i и c_i;
  - v_i - значение предложенное процессом i;
  - d_i - значение выбранное процессом i;
  - $c_i = True$, если процесс i выбрал значение, и $c_i = False$ в противном случае (изменяется только один раз);

Консенсус

- Более формальная постановка задачи:
  - имеется N процессов;
  - каждый процесс описывается 3 значениями: v_i, d_i и c_i;
  - v_i - значение предложенное процессом i;
  - d_i - значение выбранное процессом i;
  - $c_i = True$, если процесс i выбрал значение, и $c_i = False$ в противном случае (изменяется только один раз);
- требуется придумать способ выбора значения такой чтобы выполнялись свойства:
  - согласованность: $\forall i, j : c_i \wedge c_j \implies d_i = d_j$;
  - валидность: $(\forall i : v_i = k) \implies (\forall i : c_i \implies d_i = k)$;
  - завершаемость: все корректные процессы выберут значение за конечное число шагов, т. е. $c_i = True$;

Ошибки передачи сообщений

- Процессы могут обмениваться сообщениями произвольного вида
  - мы можем рассматривать решение задачи о консенсусе в различных ограничениях;
  - например, мы можем предполагать, что сообщения могут теряться;

Ошибки передачи сообщений

- Процессы могут обмениваться сообщениями произвольного вида
  - мы можем рассматривать решение задачи о консенсусе в различных ограничениях;
  - например, мы можем предполагать, что сообщения могут теряться;
- Допустим сообщения теряются:
  - теряются именно сообщения, а не процессы падают и не могут их получить;
  - как можно решить задачу о консенсусе при потере сообщений?

Проблема двух генералов

- Два генерала руководят двумя армиями и хотят совместно вторгнуться в город
  - армии должны напасть на город одновременно, т. е. нужно договориться о времени атаки;
  - генералы не могут видеть что делает другая армия, но могут обмениться сообщениями;
  - гонец сообщениями проходит мимо города и его могут подстрелить;

Проблема двух генералов

- Два генерала руководят двумя армиями и хотят совместно вторгнуться в город
  - армии должны напасть на город одновременно, т. е. нужно договориться о времени атаки;
  - генералы не могут видеть что делает другая армия, но могут обмениться сообщениями;
  - гонец сообщениями проходит мимо города и его могут подстрелить;
- Один из генералов (самый высокий, или самый толстый или просто самый главный) предлагает время
  - он отправляет сообщение с гонцом к другому генералу;
  - может ли армия атаковать город в предложенное время?

Проблема двух генералов

- Атаковать в предложенное время не безопасно:
  - мы не знаем получил ли второй генерал сообщение;
  - нужно подождать подтверждения;

Проблема двух генералов

- Атаковать в предложенное время не безопасно:
  - мы не знаем получил ли второй генерал сообщение;
  - нужно подождать подтверждения;
- Второй генерал получает сообщение и соглашается на указанное время
  - он отправляет ответное сообщение с подтверждением;
  - может ли армия теперь атаковать город в указанное время?

Проблема двух генералов

- Атаковать в указанное время опять не безопасно:
  - мы не знаем получил ли первый генерал наше подтверждение;
  - если он его не получил, то он может думать, что сообщение не доставлено;
  - нужно ждать подтверждения на подтверждение;

Проблема двух генералов

- Атаковать в указанное время опять не безопасно:
  - мы не знаем получил ли первый генерал наше подтверждение;
  - если он его не получил, то он может думать, что сообщение не доставлено;
  - нужно ждать подтверждения на подтверждение;
- Первый генерал получает подтверждение
  - он отправляет ответное сообщение с подтверждением на подтверждение;
  - может ли армия теперь атаковать город в выбранное время?

Проблема двух генералов

- Атаковать в указанное время опять не безопасно:
  - мы не знаем получил ли второй генерал наше подтверждение на подтверждение;
  - если он его не получил, то он может думать, что подтверждение не доставлено;

Проблема двух генералов

- Атаковать в указанное время опять не безопасно:
  - мы не знаем получил ли второй генерал наше подтверждение на подтверждение;
  - если он его не получил, то он может думать, что подтверждение не доставлено;
- Детерминированный алгоритм для достижения консенсуса при потерях не существует:
  - допустим противное и такой алгоритм существует и он завершается за k сообщений;
  - алгоритм должен переживать потери сообщений, а значит он может потерять k-ое сообщение;
  - т. е. тот же алгоритм должен завершится и за $k - 1$ сообщение;

Синхронность и асинхронность

- Пусть соединение теперь надежное
  - обычно мы полагаемся на TCP;
  - TCP не дает гарантий, но на практике ломается в очень суровых ситуациях;

Синхронность и асинхронность

- Пусть соединение теперь надежное
  - обычно мы полагаемся на TCP;
  - TCP не дает гарантий, но на практике ломается в очень суровых ситуациях;
- Можем ли мы решить задачу о консенсусе? Зависит от условий:
  - могут ли процессоры "падать" или нет?
  - существуют ли ограничения на время доставки и обработки сообщения или нет?

Синхронность и асинхронность

- Пусть соединение теперь надежное
  - обычно мы полагаемся на TCP;
  - TCP не дает гарантий, но на практике ломается в очень суровых ситуациях;
- Можем ли мы решить задачу о консенсусе? Зависит от условий:
  - могут ли процессоры "падать" или нет?
  - существуют ли ограничения на время доставки и обработки сообщения или нет?
- Синхронные и асинхронные системы:
  - система синхронна, если известны ограничения на время доставки и обработки сообщения;
  - система асинхронна, если ограничений нет (мы не можем узнать "упал" процессор или просто долго думает);

Консенсус в асинхронных системах

- Если сообщения не теряются и процессоры не "падают", то консенсус легко достижим
  - мы только не знаем, сколько времени нам на это понадобится;

Консенсус в асинхронных системах

- Если сообщения не теряются и процессоры не "падают", то консенсус легко достижим
  - мы только не знаем, сколько времени нам на это понадобится;
- Что если процессоры "падают"?
  - если процессор после "падения" не восстанавливается, то детерминированного алгоритма достижения консенсуса не существует;
  - причем достаточно ровно одного падения в правильном месте;
  - этот факт известен как "FLP Impossibility";

Консенсус в асинхронных системах

- Если сообщения не теряются и процессоры не "падают", то консенсус легко достижим
  - мы только не знаем, сколько времени нам на это понадобится;
- Что если процессоры "падают"?
  - если процессор после "падения" не восстанавливается, то детерминированного алгоритма достижения консенсуса не существует;
  - причем достаточно ровно одного падения в правильном месте;
  - этот факт известен как "FLP Impossibility";
- Что делать если консенсус все-таки нужен?
  - многие системы являются именно асинхронными (чем больше промежуточных узлов тем хуже);
  - считаем, что все узлы возвращаются после падения;

Paxos

- Paxos - алгоритм достижения консенсуса в асинхронных системах с падениями:
  - как известно алгоритма удовлетворяющего всем 3 свойствам не существует;
  - paxos не гарантирует завершения за конечное число шагов;
  - paxos гарантирует отсутствие deadlock-ов, т. е. если все узлы вернутся в строй, консенсус будет достигнут;
  - на практике ситуация, в которой paxos "зависнет" очень редкие;

Paxos

- Paxos состоит из одного или более раундов:
  - у каждого раунда есть лидер - тот кто раунд начал;
  - каждый раунд имеет уникальный идентификатор;
  - состоит из 3 этапов;
  - консенсус достигнут, если лидер успешно завершает свой раунд;

Paxos

- Этап первый - просим принять нас в качестве лидера:
  - выбираем уникальный идентификатор раунда;
  - связываемся с другими процессами и посылаем им идентификатор
  - в ответ получаем два значения:
    - максимальный номер раунда, для которого процесс принял значение (если принял);
    - и значение, которое он принял в этом раунде;
  - нам достаточно ответ от большинства
    - если никто из них не видел большего идентификатора - мы лидер;
    - иначе можно на этом и закончить;

Paxos

- Этап второй - предложение (значения):
  - лидер должен выбрать подходящее значение:
    - у нас есть какое-то значение, иначе зачем бы нам запускать голосование;
    - кроме того процессы могли прислать значения, на которые они согласились в одном из предыдущих раундов;
    - если наше значение единственное, то выбираем его, в противном случае выбираем самое "новое" значение из присланных;
  - посылаем выбранное значение процессам, ждем подтверждения от большинства:
    - процесс посылает подтверждение, если не видел большего идентификатора раунда;
    - процесс также сохраняет полученное значение;
    - процесс посылает отказ, если он видел больший идентификатор;

Paxos

- Этап третий - коммит:
  - мы получили подтверждение от большинства, консенсус достигнут;
  - осталось сообщить об этом заинтересованным сторонам;
- строго говоря консенсус достигнут несколько раньше:
  - как только большинство процессов сохранили себе значение;
  - просто не все (включая лидера раунда) об этом знают;