Consistency, Availability and Partition Tolerance

Курс «Базы данных»

Цесько Вадим Александрович
http://incubos.org
@incubos

Computer Science Center

16 сентября 2013 г.

Содержание

Идея сервиса

Never forget, even without remembering![a]

[a] http://ksat.me/a-plain-english-introduction-to-cap-theorem/

Ever felt bad that you forget so much? Don't worry. Help is just a phone away!
When you need to remember something, just call 555-55-REMEM and tell us what you need to remember. For eg., call us and let us know of your boss's phone number and forget to remember it. When you need to know it back, call back the same number 555-55-REMEM and we'll tell you what's your boss's phone number.
Charges: only $0.1 per request

Типичный диалог

- *Customer*: Hey, can you store my neighbor's birthday?
- *You*: Sure... When is it?
- *Customer*: 2nd of Jan
- *You*: (write it down against the customer's page in your paper note book) Stored. Call us any time for knowing your neighbor's birthday again!
- *Customer*: Thank you!
- *You*: No problem! We charged your credit card with $0.1

Растёт нагрузка

- Нужен только блокнот и телефон
- Отлично работает
- Сарафанное радио
- Получили финансирование от YCombinator
- Сотни звонков в день

Проблемы подхода

- Клиенты висят в очереди на телефоне и злятся
- Вы заболели и пропустили рабочий день и день оповещений

Масштабируемся

- Берём жену в долю
- У каждого добавочный номер
- Клиенты используют всё тот же 555–55-REMEM
- АТС балансирует клиентов между сотрудниками

Проблема

- *Customer*: Hey!
- *You*: Glad you called “Remembrance Inc!”. What can I do for you?
- *Customer*: Can you tell me when is my flight to New Delhi?
- *You*: Sure... 1 sec, sir. (You look up your notebook. WOW! There is no entry for "flight date"in customer’s page!!!)
- *You*: Sir, I think there is a mistake. You never told us about your flight to Delhi.
- *Customer*: What?! I just called you guys yesterday! (Cuts the call!)

Чиним неконсистентность

- Перед тем как записать что-либо, говорим партнёру
- Таким образом, обновления хранятся у нас обоих
- Когда клиент спрашивает, то вся информация под рукой

Проблемы подхода

- Дольше операции обновления, но операций чтения большинство
- Если кто-то из нас заболеет — проблема **недоступности**

Consistent & Available

Новая версия алгоритма:

- Перед записью говорим партнёру
- Партнёр недоступен — отправляем ему email
- В начале рабочего дня разбираем почту и обновляем блокнот

Partition Tolerance

- Жена обиделась, всё-таки пришла на работу, но решила не сообщать об обновлениях
- Можно реализовать partition tolerance, если решить не принимать запросы, пока не восстановится связность с женой
- Но тогда жертвуем availability

CAP theorem

E. Brewer. *Towards Robust Distributed Systems*, 2000.

Суть

Выберите 2 из 3:

- Consistency
- Availability
- Partition Tolerance

Пояснения

На самом деле[1]:

- Невозможность **полного** обеспечения **C**, **A** и **P**
- **P** in a distributed system is a **MUST HAVE**
- Партиции относительно редки — необязательно жертвовать **C**/**A** при их отсутствии
- Выбор различного соотношения **C**/**A** для разных подсистем, запросов, данных, пользователей и т. д.
- Свойства **непрерывны**, а не бинарны

[1] http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

Latency

CAP проявляется во время **таймаута**:

- Отменить операцию = снизить доступность
- Подтвердить операцию = снизить консистентность
- Перезапрос = отложить решение
- Бесконечный перезапрос = выбираем консистентность

Обработка партиций

- Обнаружение партиционирования
  - Не все узлы могут признать партиционирование
  - Можно подбирать таймауты $\Rightarrow$ false positives
- Ограничение набора возможных операций/снижение консистентности
  - Можно запомнить и «отложить» операцию
  - Yahoo PNUTS: локальный мастер per user + асинхронная репликация $\Rightarrow$ меньше задержки
  - Facebook[2]: мастер всегда один, запись в удалённый мастер и чтение 20 сек из мастера, затем из локальной реплики
- Восстановление консистентности и устранение ошибок

[2] http://www.facebook.com/note.php?note_id=23844338919&id=9445547199

Картинка

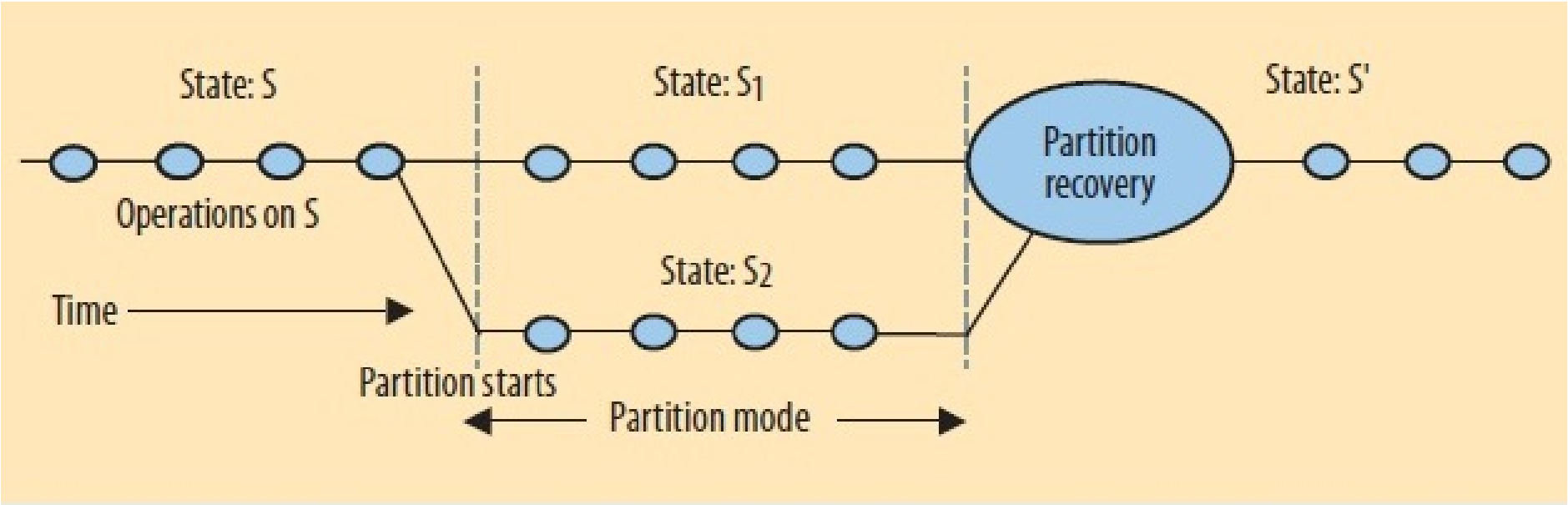

Figure 1. The state starts out consistent and remains so until a partition starts. To stay available, both sides enter partition mode and continue to execute operations, creating concurrent states S_1 and S_2, which are inconsistent. When the partition ends, the truth becomes clear and partition recovery starts. During recovery, the system merges S_1 and S_2 into a consistent state S' and also compensates for any mistakes made during the partition.

Трюки

- Вся система недоступна — есть, например, HTML5 on-client persistent storage
- Различные области консистентности при партициях (primary partition, quorum)
- Шардирование
  - Нет глобальных инвариантов
  - Но возможен прогресс даже при партициях

Восстановление консистентности

- Нужно знать инварианты (не всегда очевидно)
- Почти то же, что параллельные обновления в многопоточном программировании
- Векторные часы для обнаружения конфликтов
- Откат ошибок (желательно иметь лог операций — см. DVCS)
- Commutative Replicated Data Types (CRDTs)[3][4]

[3] M. Shapiro et al. Conflict-Free Replicated Data Types. 2011

[4] M. Shapiro et al. Convergent and Commutative Replicated Data Types. 2011

Пример: ATM

- Инвариант: баланс больше 0
- Выбираем availability
- Связь пропала — лимит на снятие наличных
- Операция коммутативна
- Связь восстановилась — считаем баланс, возможно, штраф за overdraft
- См. Check kiting[5]

[5] http://en.wikipedia.org/wiki/Check_kiting

Use Case: Ebay

Principles for Scaling[6]:

- Partition Everything
  - Vertical, horizontal, no session state
- Asynchrony Everywhere
  - Events, multicast
- Automate Everything
  - Adaptive configuration, machine learning
- Remember Everything Fails
  - Failure detection, rollback, graceful degradation
- Embrace Inconsistency
  - Eventual consistency, no distributed transactions

[6] Randy Shoup at LADIS 2008, `http://www.cs.cornell.edu/projects/ladis2008/materials/eBayScalingOdyssey%20ShoupTravostino.pdf`

Определение

Транзакция

A unit of work performed against a database, and treated in a coherent and reliable way independent of other transactions[a]

[a] http://en.wikipedia.org/wiki/Database_transaction

Две философии:

- ACID[7] — focus on consistency
- BASE — focus on high availability

[7] Name was coined (no surprise) in California in 60's

ACID-транзакции

- **Atomicity** — «либо всё, либо ничего»
  - `commit`/`rollback`, undo-log, атомарные операции, ...
- **Consistency** — переход из одного «корректного» состояния в другое
  - Индексы, форматы, ссылки, ограничения, триггеры, ...
- **Isolation** — иллюзия последовательного выполнения транзакций
  - Уровни изоляции: Read uncommitted / dirty reads, Read committed / non-repeatable read, Repeatable reads / phantom reads, Serializable, MVCC[8], ...
- **Durability** — надёжное долговременное хранение

[8] `http://en.wikipedia.org/wiki/Multiversion_concurrency_control`

Распределённый commit

- **Two-phase commit protocol**[9]
- **Three-phase commit protocol**[10]
- **Paxos**[11]
- Возможно, рассмотрим позже

[9] http://en.wikipedia.org/wiki/2PC

[10] http://en.wikipedia.org/wiki/3PC

[11] http://en.wikipedia.org/wiki/Paxos_(computer_science)

ACID in CAP

- **Atomicity**
  - Используется во всех партициях
  - Упрощает восстановление
- **Consistency**
  - Невозможна при партиционировании
  - Запрет некоторых операций
  - Восстановление инвариантов впоследствии
- **Isolation**
  - Можно работать только с одной партицией
  - Либо более слабая корректность, компенсируемая восстановлением
- **Durability**
  - Durable history позволяет исправлять ошибки при восстановлении
  - Иногда ослабляют из-за дороговизны

BASE

Dan Pritchett. BASE: An Acid Alternative[12]. 2008:

- **Basically Available**
- **Soft State**
- **Eventually consistent**

[12] http://queue.acm.org/detail.cfm?id=1394128

Чтение на ночь

Fallacies of Distributed Computing[13][14]:

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous
- ...

[13] http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing

[14] Arnon Rotem-Gal-Oz. Fallacies of Distributed Computing Explained, http://www.rgoarchitects.com/Files/fallacies.pdf

Библиотечка

- Google. Distributed Systems and Parallel Computing[15]
- Quora: What are the top startup engineering blogs?[16]

[15] http://research.google.com/pubs/DistributedSystemsandParallelComputing.html

[16] http://www.quora.com/What-are-the-top-startup-engineering-blogs

Вопросы?

- http://incubos.org/contacts/
- Общие вопросы — в Twitter: @incubos
- Вопросы по лекциям — в комментариях: http://incubos.org/blog/
- Частные вопросы — в почту vadim.tsesko@gmail.com