CAP Theorem Impact in Reliable Data Processing

Pavel Krobot (xkrobo01@stud.fit.vutbr.cz)

Brno University of Technology,
Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

10. 12. 2015

OUTLINE

1) CAP Theorem
   a) Explanation
   b) Difficulties, misunderstandings, implications

2) Proof of CAP Theorem
   a) Asynchrnous network model
   b) Partialy synchrnous network model

- Eric Brewer, 2000, University of California
- *"A shared-data system can have at most two of the three following properties:*

  - ***C**onsistency*

  - ***A**vailability*

  - ***P**artition tolerance"*

CONSISTENCY

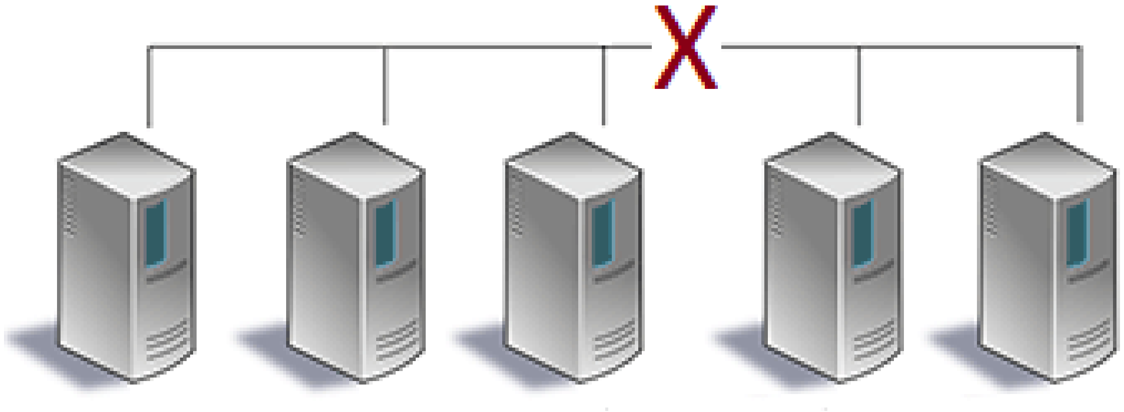
- Equivalent to having single up-to-date copy of data
- Formal definition uses an existence of total order on all operations
- Any read operation must return a result of the last write operation

Availability

- Access to the data at any time
- *„Every request by a non-failing node in a distributed system must result in a response"*
- Every request has to terminate

PARTITION TOLERANCE

- Ability to operate as usual when a network partition occurs
- *„All messages sent from nodes in one component of the partition to nodes in another component are lost "*



Never sacrifice partition tolerance

- **We can never sacrifice partition tolerance**
- Every networked distributed system experiences a network partition at some point
- Trade-off between consistency and availability

CONSISTENCY VS. AVAILABILITY

- Not a binary decision
- Both have its use in particular use cases
- Prefer **C**: refuse/postpone some requests (writes mainly)
- Prefer **A**: always response, even if results will not be complete and writes could be conflicting

PROOF – ASYNCHRONOUS NETWORK MODEL (I.)

- Seth Gilbert and Nancy Lynch
- Asynchronous network model from the book ‘Distributed algorithms’:
    - No clock
    - Nodes makes decisions based only on the received messages and local computations

PROOF – ASYNCHRONOUS NETWORK MODEL (II.)

- Distributed system component by **I/O automaton**:
  - Simple state machine with transitions
  - Transitions associated with actions:

    Input } communicaton
    Output

    Internal – visible only for automaton itself

- Fariness, liveness, safety

Proof – asynchronous network model (III.)

- Theorem **T1**:

  "It is impossible in the asynchronous network model to implement read/write data object that guarantees:

  - *availability and*
  - *atomic consistency*

  in all fair executions (including those in which messages are lost)."

- Proof by contradiction

PROOF – ASYNCHRONOUS NETWORK MODEL (IV.)

- Algorithm **A** that meets: atomicity, availability, partition tolerance
- Construct an execution of **A** with an inconsistent response
- Network:
  - at least two nodes
  - could be divided into two disjoint, non-empty sets: $\{\mathbf{G_1},\mathbf{G_2}\}$
  - all messages between $\mathbf{G_1}$ and $\mathbf{G_2}$ are lost
- Write in $\mathbf{G_1}$, later read in $\mathbf{G_2}$ -> ***read cannot return result of earlier write*** *(no messages between G1 and G2 during network partition)*

- $\mathbf{v_0}$ - initial value of the atomic object
- $\boldsymbol{\alpha_1}$ - prefix of an execution of **A**.

  - single write of value in $\mathbf{G_1}$ (value is not equal to $\mathbf{v_0}$)
- $\boldsymbol{\alpha_2}$ - prefix of an execution of **A**.

  - single read of value in $\mathbf{G_2}$ (value is not equal to $\mathbf{v_0}$)
- No other client requests
- No messages between $\mathbf{G_1}$ and $\mathbf{G_2}$ in $\boldsymbol{\alpha_1}$ or $\boldsymbol{\alpha_2}$
- $\boldsymbol{\alpha}$ – execution **A** of beginning $\boldsymbol{\alpha_1}$ with continuing with $\boldsymbol{\alpha_2}$

- In the **α** execution the read from **α_2** must still return **v_0**
- Read request does not begin until write from **α_1** completes
- ***Atomic consistency is broken -> no such algorithm exists***

□

- Partially synchronous network model from the book ‘Distributed algorithms’:
  - Every node has a clock (increase at the same rate, but not synchronously)
  - Clocks can be observed to measure how much time has passed

- Every message is either:
  - Delivered within given, known time $\mathbf{t_{msg}}$ or lost
  - Processed by node in given, known time $\mathbf{t_{local}}$
- General timed automata – from Timed automaton, with fairness conditions replaced with lower and upper bound on time.

- Theorem **T1** holds also in partialy synchronous network model:
- Again divide network to $\{\mathbf{G_1},\mathbf{G_2}\}$
- Construct similar execution as in case of **T1** – write in $\mathbf{G_1}$, later read in $\mathbf{G_2}$ -> ***read cannot return result of earlier write*** *(no messages between G1 and G2 during network partition)*

- $\mathbf{v_0}$ - initial value of the atomic object
- $\boldsymbol{\alpha_1}$ - same as in the case **T1** proof
- $\boldsymbol{\alpha_2}$´ - slightly different then $\boldsymbol{\alpha_2}$

  - begins with time interval at least as long as duration of $\boldsymbol{\alpha_1}$ followed by events of $\boldsymbol{\alpha_2}$
- No other client requests
- No messages between $\mathbf{G_1}$ and $\mathbf{G_2}$ in $\boldsymbol{\alpha_1}$ or $\boldsymbol{\alpha_2}$´
- $\boldsymbol{\alpha}$ – execution **A** of begining $\boldsymbol{\alpha_1}$ with continuing with $\boldsymbol{\alpha_2}$´

- Again, In the **α** execution the read from $\boldsymbol{\alpha_2}'$ must still return $\mathbf{v_0}$
- Read request return initial value instead of new value from write request in **G1**
- ***Atomic consistency is broken -> no such algorithm exists***

□

- CAP Theorem: properties of distributed systems – can have at most two of **C** – **A** – **P**.
- We could never sacrifice **P**artition tolerance
- Always trade-off between **C**onsistency and **A**vailability
- Proven in asynchronous and partialy synchronous network models

1. **SALOMÉ, Simon.** ***Report to Brewer’s CAP Theorem*** [online]. 2012. Available at: https://fenix.tecnico.ulisboa.pt/downloadFile/845043405442708/10.e-CAP-3.pdf
2. **BREWER, Eric.** ***CAP twelve years later: How the "rules" have changed*.** 45(2): 23-29. DOI: 10.1109/MC.2012.37. ISSN 0018-9162. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6133253
3. **HALE, Coda.** ***You Can’t Sacrifice Partition Tolerance.*** [online]. 2010. Available at: http://codahale.com/you-cant-sacrifice-partition-tolerance/
4. **GILBERT, Seth, and LYNCH, Nancy.** ***Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*** 2002. *SIGACT News 33, 2* (June 2002), 51-59. DOI=http://dx.doi.org/10.1145/564585.564601
5. **LYNCH, Nancy A.** ***Distributed algorithms*.** San Francisco: Morgan Kaufmann, 1997, xxiii, 872: ISBN 978-1-558-60348-6.

Thank You For Your Attention