Brewer’s CAP Theorem

E. Brewer – Towards Robust Distributed Systems
S. Shim – The CAP Theorem’s Growing Impact
Historical Background

- **ACID** – Atomicity, Consistency, Isolation, Durability
  - Works fine on one Server
  - Latency issues in Distributed Systems

- **BASE** – Basically Available, Soft state, Eventual Consistency
  - Invented in order to achieve high Availability in Distributed Systems
The Origin

- Brewer’s talk at the Symposium on Principles of Distributed Computing (PODC) 2000
- Brewer’s background:
  - Professor at the University of Berkely
  - Co-Founder & Chief Scientist of Inktomi
- Formally proven by Seth Gilbert and Nancy A. Lynch in 2002
The CAP Theorem

3 Properties:
- Consistency
- Availability
- Network Partition Tolerance

You can have at most two of these properties for any shared-data system
Problematic Part: Update from $D_1$ to $D_2$ if network gets partitioned
CAP in an Example

- Give up **Partition Tolerance**
  - 2 Phase Commit

→ In case of network partition the system won‘t work at all
Give up **Availability** (ACID)
- Pessimistic locking

→ In case of network partition we need to wait
CAP in an Example

- Give up **Consistency** (BASE)
  - Optimistic locking
- In case of network partition we read inconsistent data
Since P cannot be forfeited, we need to choose between A and C

Except: Not only 2 options but trade-off between Availability and Consistency
Trade-off examples

- Amazon’s Dynamo: eventual Consistency
  - Updates will be executed eventually, but the object can still be accessed meanwhile

- Yahoo’s PNUTS: guarantee that updates arrive in the same order they were issued
  - Each tuple has master node that orders updates
The 3 properties of CAP are not “equal“
- Partition Tolerance is true/false
- Availability and Consistency are a spectrum

What is Network Partition?
- Is high latency (→ low availability) a network partition or not?
- Partition Tolerance and Availability overlap

→ Rather speak of C–A Tradeoff