ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

Cloudy Work at the Systems Group, ETH Zurich

Tim Kraska

Outline

- Building Web Applications without a DBMS
- Consistency Rationing
- Cloudy/Smoky
- CloudBench

Outline

- **Building Web Applications without a DBMS**
- Consistency Rationing
- Cloudy/Smoky
- CloudBench

Motivation

- Building a web page, starting a blog, and making both searchable for the public have become a commodity
- But providing your own service (and to get rich) still comes at high cost:
  - Have the right (business) idea
  - Run your own web-server and database
  - Maintain the infrastructure
  - Keep the service up 24 x 7
  - Backup the data
  - Tune the system if the service is used more often

And then comes the digg-effect

Requirements for DM on the Web

- Scalability
  - response time independent of number of clients
- No administration
  - “outsource” patches, backups, fault tolerance
- 100 percent read + write availability
  - no client is ever blocked under any circumstances
- Cost ($$$)
  - pay as you go along, no investment upfront
  - get cheaper every year, leverage new technology

Consistency: Optimization goal, not constraint

Why Cloud Computing?

For the moment focus on IaaS!!!

- Commoditization of computing
  - CPU, storage, network
- Goal 1: Reduction of cost
  - principle: fine-grained renting of resources
  - “Pay as you go” (cost grow linearly)
- Goal 2: Simplification of management
  - potentially non-breakable computing resources
  - potentially no administration
- Goal 3: Scalability
  - scale linearly
  - no scalability limit

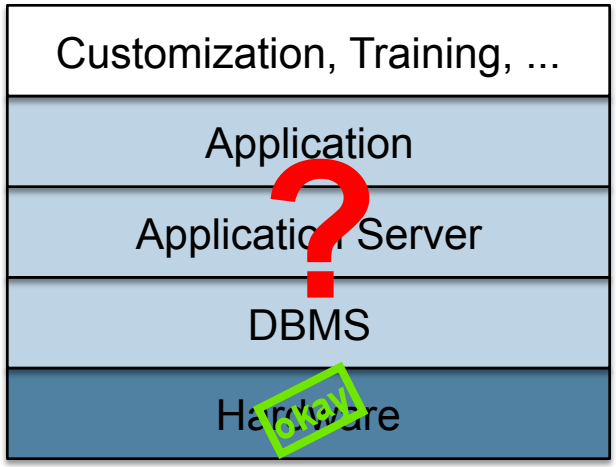
Requirements Revisited

- Scalability
  - response time independent of number of clients
- No administration
  - “outsource” patches, backups, fault tolerance
- 100 percent read + write availability
  - no client is ever blocked under any circumstances
- Cost ($$$)
  - get cheaper every year, leverage new technology
  - pay as you go along, no investment upfront

Consistency: Optimization goal, not constraint

Databases and the Cloud

- IaaS a great starting point
- unfortunately, only a fraction of the stack



Two camps of thought for DBMS:

- Camp 1: Install standard DBMS in the Cloud
  - take a traditional DBMS (e.g., Oracle, MySQL, ...)
  - install it on an EC2 instance
  - use S3 or EBS as a persistent store
- Camp 2: Rethink the whole system architecture
  - do NOT use a traditional DBMS
  - optimize the system for cost and consistency

Camp 1: Install standard DBMS in the Cloud

- **Advantages**
  - traditional databases are available
  - proven to work well; many tools
  - people trained and confident with them
- **Disadvantages**
  - traditional DBMS solve the wrong problem anyway
    - focus on throughput and consistency
    - SAP and Oracle misuse the DBMS already today
  - traditional DBMS make the wrong assumptions
    - e.g., DBMS optimizers fail on virtualized hardware
    - e.g., DBMS bulkloading tools collapse on shared storage

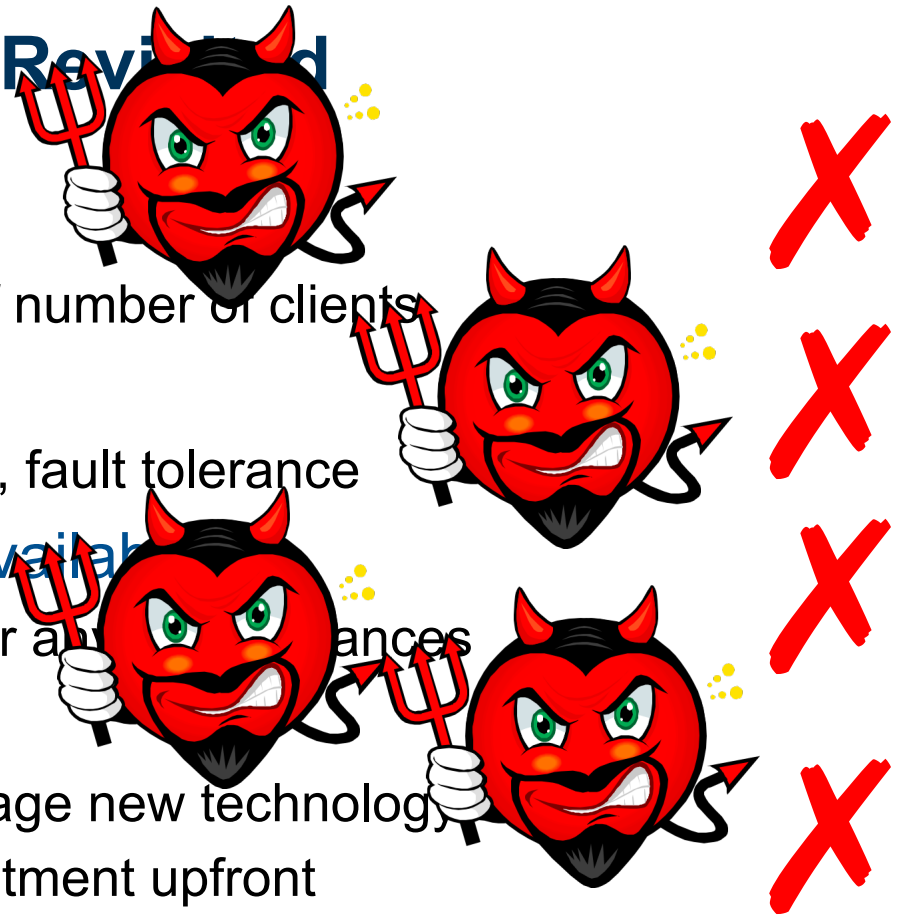
Camp 1: Requirements revisited

- Scalability
  - response time independent of number of clients
- No administration
  - “outsource” patches, backups, fault tolerance
- 100 percent read + write availability
  - no client is ever blocked under any circumstances
- Cost ($$$)
  - get cheaper every year, leverage new technology
  - pay as you go along, no investment upfront

Consistency: Optimization goal, not constraint

Camp 1: Requirements Revised

- Scalability
  - response time independent of number of clients
- No administration
  - "outsource" patches, backups, fault tolerance
- 100 percent read + write availability
  - no client is ever blocked under any circumstances
- Cost ($$$)
  - get cheaper every year, leverage new technology
  - pay as you go along, no investment upfront



Consistency: Optimization goal, not constraint

Camp 2: Rethink the whole system architecture

- Rethink the whole system architecture
  - do NOT use a traditional DBMS
  - optimize the system for cost & consistency
- Here: Bulding web application without a DBMS
  - create new breed of **application server with DB**
  - run application server on virtual instances
  - use cloud storage + distributed consistency protocols
- Advantages and Disadvantages
  - requires new breed of (immature) systems + tools
  - solves the right problem and gets it right
  - optimized for cost
  - leverages organization‘s investments in SOA

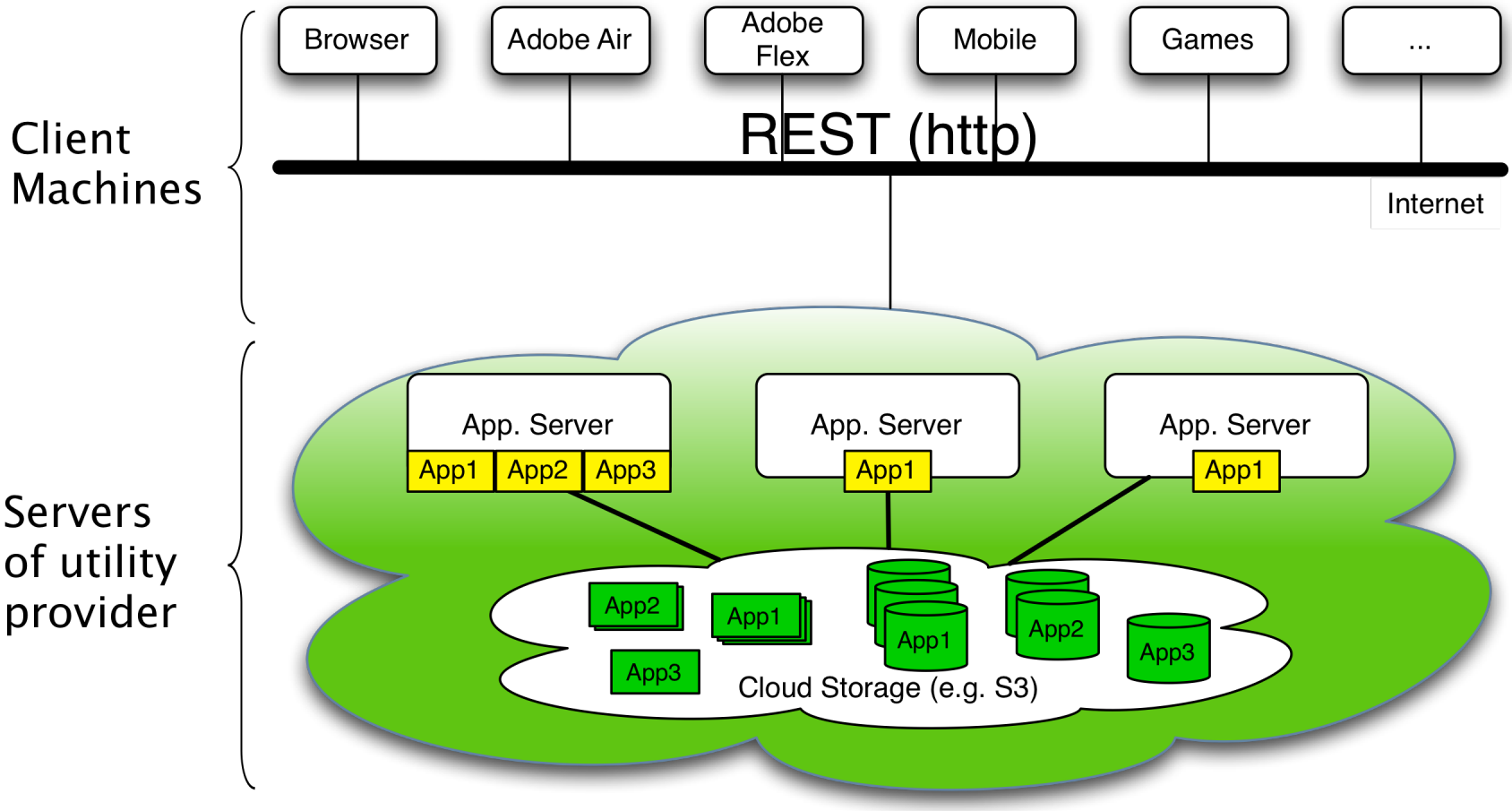
Requirements for DM on the Web

- Scalability
  - response time independent of number of clients
- No administration
  - “outsource” patches, backups, fault tolerance
- 100 percent read + write availability
  - no client is ever blocked under any circumstances
- Cost ($$$)
  - get cheaper every year, leverage new technology
  - pay as you go along, no investment upfront



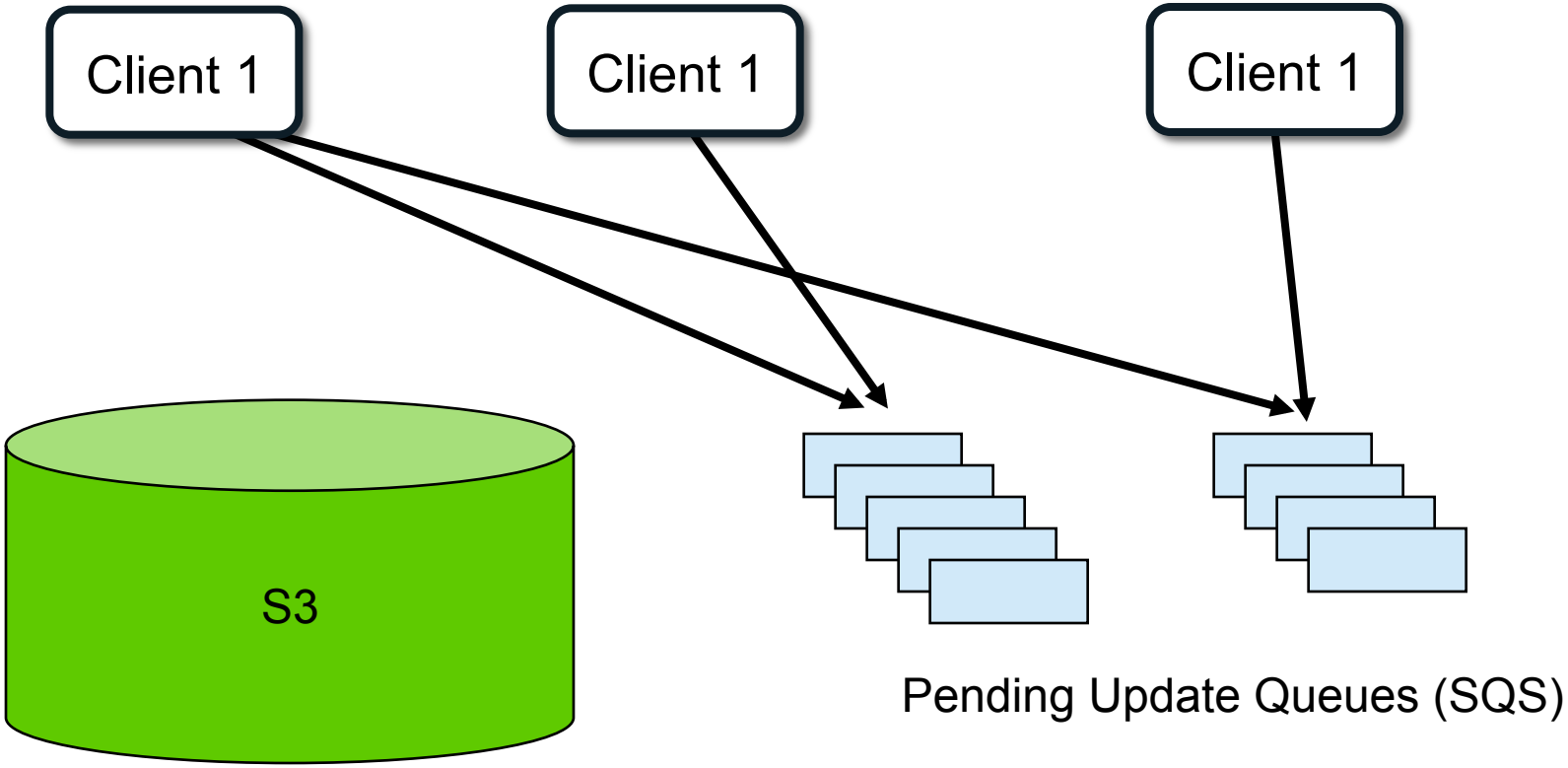

Consistency: Optimization goal, not constraint

Bulding web application without a DBMS



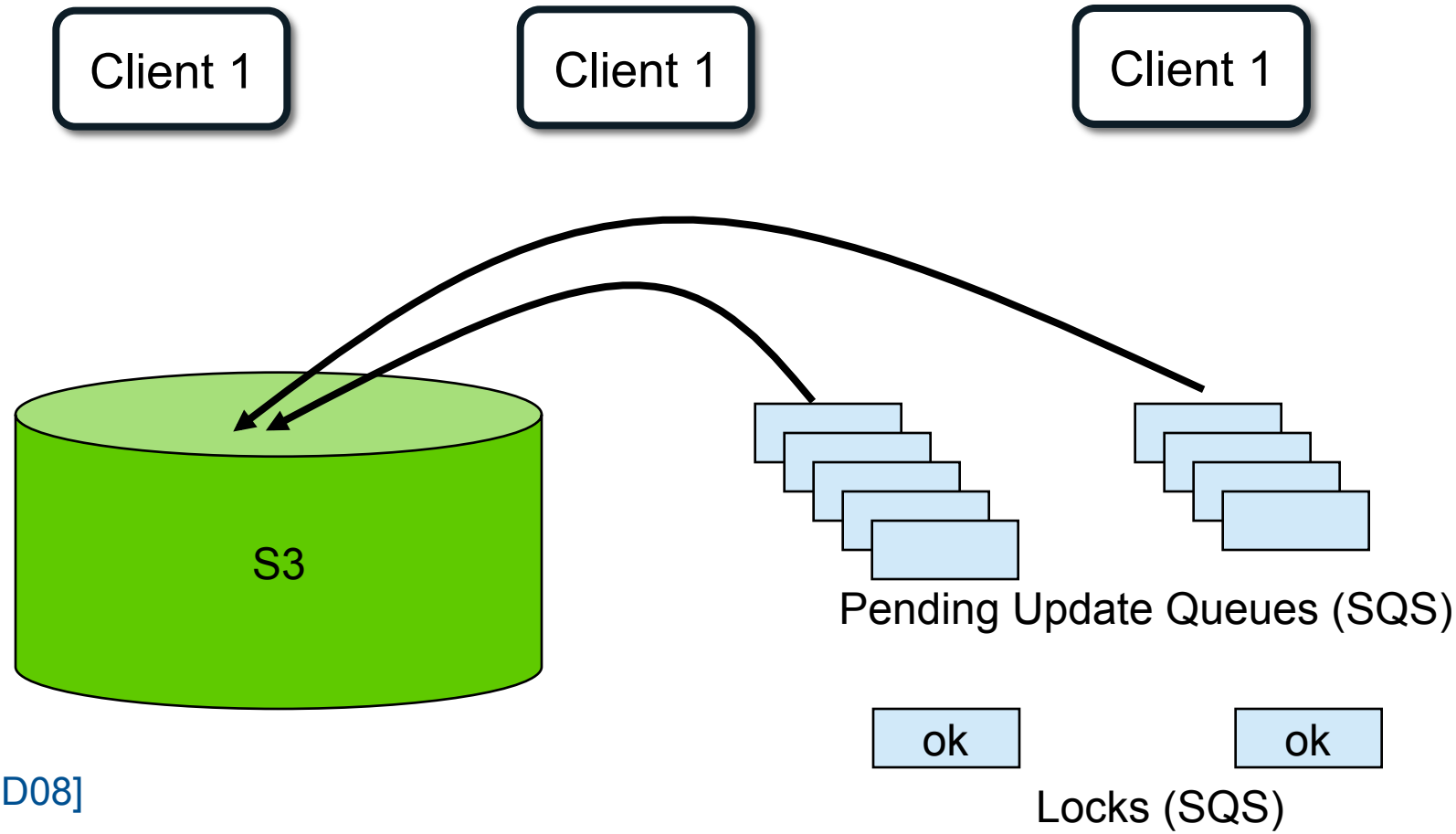
Concept commercialized in Sausalito (28msec, Inc.)

Step 1: Clients commit update records to pending update queues



[SIGMOD08]

Step 2: Checkpointing propagates updates from SQS to S3



[SIGMOD08]

Optimization goal: Consistency

Choose the consistency level the application requires

- Eventual consistency (basic protocol)
  - updates become visible any time and will persist
  - no lost update on page level
- Atomicity
  - all or no updates of a transaction become visible
- Monotonic reads, read your writes, monotonic writes, ...
- Strong consistency
  - database-style consistency (ACID) via OCC

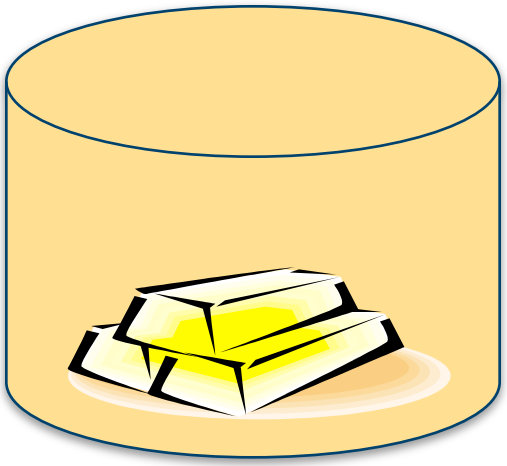
Pay the price in performance, cost and availability!!!!

Outline

- Building Web Applications without a DBMS
- **Consistency Rationing**
- Cloudy/Smoky
- CloudBench

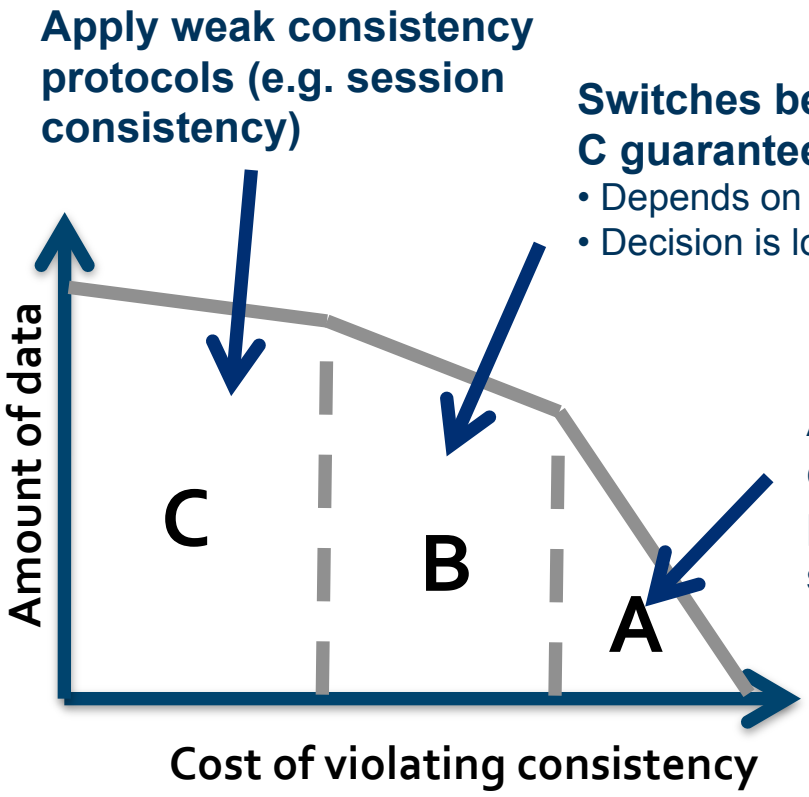
Consistency Rationing - Idea

- ACID prevents scaling & availability (CAP theorem)!
- Strong consistency is expensive
- But not everything is worth gold!



- Idea: Handle data according to the cost of inconsistency/lost etc.
- Violating consistency or even losing data is OK as long as it helps to reduce the overall cost

Consistency Rationing - Guarantees



- Consistency requirements per category instead of transaction level
- Different categories can mix in a single transaction

Dynamic Strategies

Use strong consistency only if it is cost-efficient

- Conflict rate
  - use case: Collaborative editing
  - collect temporal statistics (update rate)
  - use strong consistency protocol only if the likelihood of a conflict is high
- Value constraint
  - use case: Web shop / ticket reservation
  - collect temporal statistics (value changes)
  - use strong consistency protocol only if the likelihood of violating the constraint is high
- Time based
  - use case: Auction systems
  - consistency protocol depend on the time to deadline X

[VLDB09]

Outline

- Building Web Applications without a DBMS
- Consistency Rationing
- **Cloudy/Smoky**
- CloudBench

Cloudy

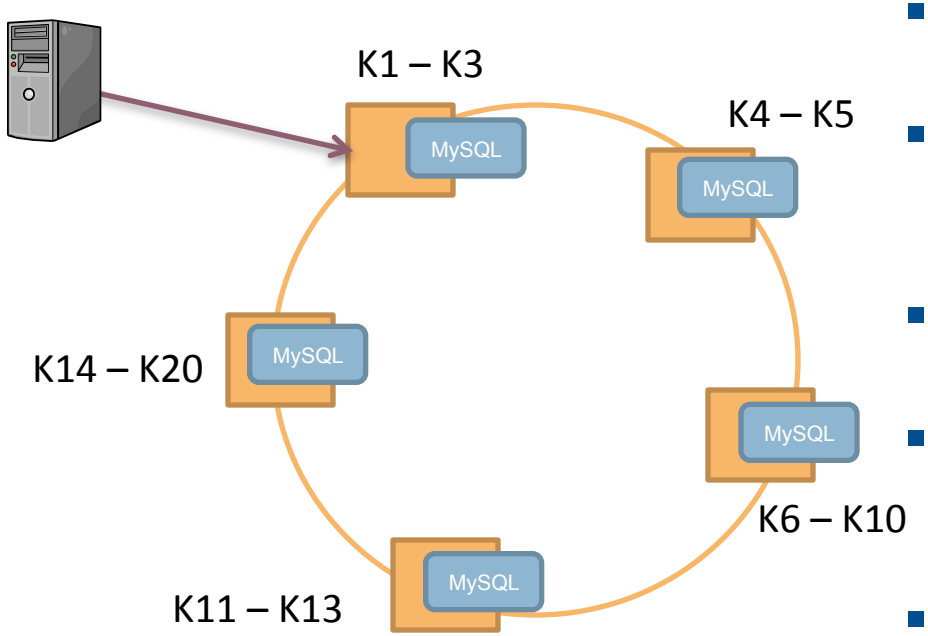
- A cloud storage system with integrated transactions guarantees
- Based on Dynamo/BigTable
- Allows to relax all ACID properties
- A research platform for:
  - Data rationing
  - Cloud bursting
  - Large scale query processing

Configurable ACID Guarantees

- Hierarchy of consistency levels
  - System → changes rarely – copy on each Node
  - Collection → changes sometimes – copy on each responsible Node
  - Row → changes often – copy next to Data

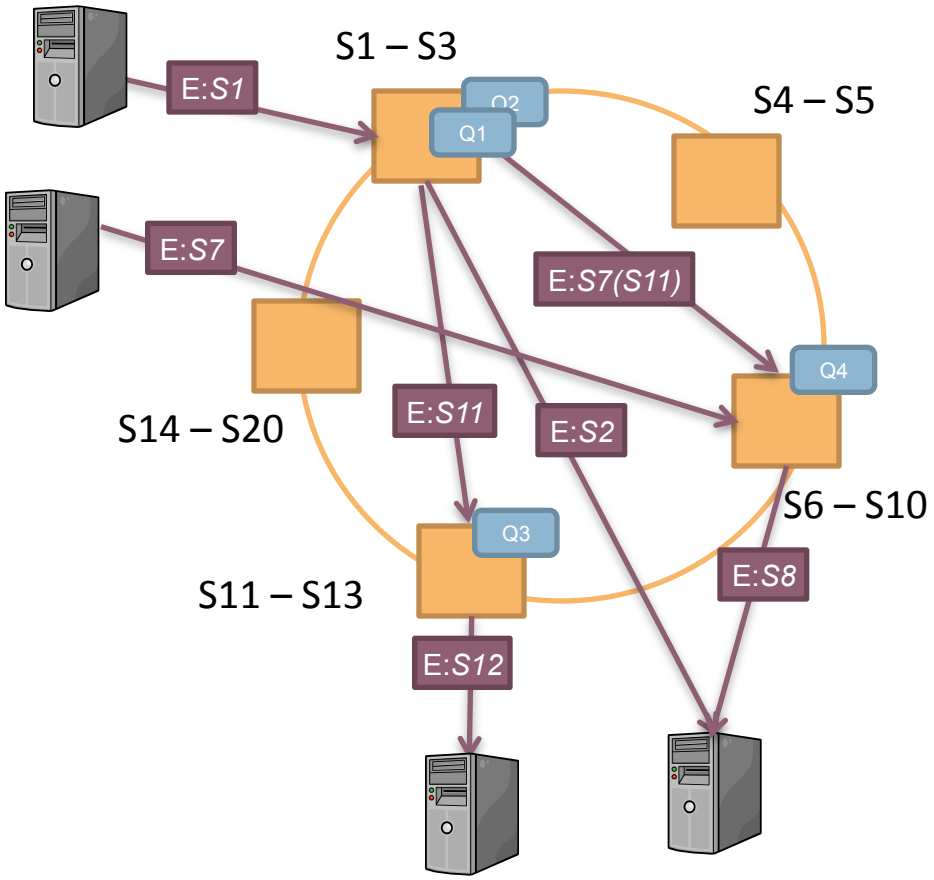
	Range	System	Collection	Row
Durability	Replication # Nodes → later Percentage	Yes	Yes	No
Consistency	Read-/Write-Quorum/Caching	Yes	Yes	Yes
Atomicity	No /EC/Yes	Yes	Yes	Yes
Isolation	No / Rule-based/OCC,PCC	Yes	Yes	Yes

Cloudy



- Data model
  - Columns and super-columns
- Routing
  - DHT with memorization
  - Range-preserving hash function
- Scaling
  - Cloud bursting
- Fault tolerance
  - Gossiping
  - Adaptable quorums
- Load-balancing
  - Virtual nodes
  - Moving nodes
- Query Processing
  - MySQL
- Indexes
  - Distributed

Smoky



Provide streaming as a service: No installation, maintenance etc.

- Use Cases
  - Server/workflow monitoring
  - message transformations
  - combining RSS streams
  - ...
- Leverage established cloud techniques
- Idea:

Storage ring:
Key-> Data

Stream ring:
Event-type ->Query

Cloudy & Smoky – Work in progress

- First results available (TPC-W/CloudBench)
- Cloudy still slower than MySQL-Cluster (does it matter?)
- Future work
  - extending the probabilistic guarantees
  - distributed query processing/combining with Hadoop
  - extend Smoky to other scenarios (application hosting)
  - …

Outline

- Building Web Applications without a DBMS
- Consistency Rationing
- Cloudy/Smoky
- **CloudBench**

Benchmarking the cloud

- Traditional benchmarks are not suited for the cloud
  - Test a fix setup
  - Report some maximum number of the system under test (SUT)
- Examples:
  - Linear road
    - increase the workload until the response time to events is longer than 5s
    - report the maximum load factor (e.g 2.5LR)
  - TPC-W benchmarks
    - increase the workload until SLA is not longer fullfiled (90% of the WI in 2s)
    - report the maximum WIPS

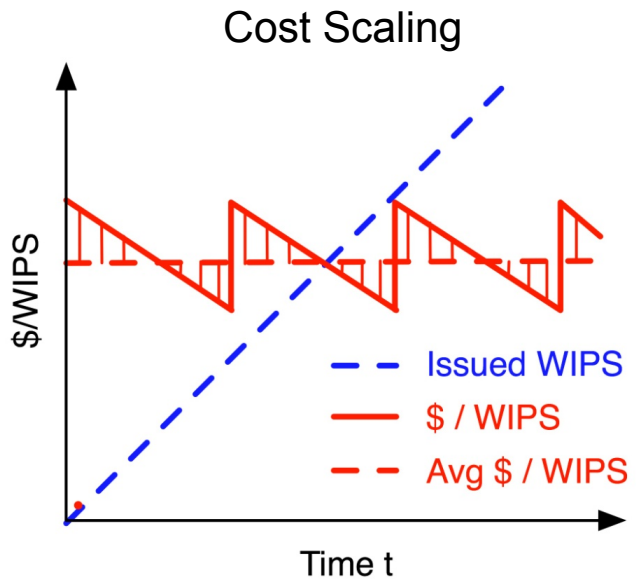
What makes benchmarking the cloud difficult?




- Consistency
- Varity of products
- Lot sizes
- Different fault tolerance/SLA guarantees
- Scalability limits

CloudBench

- Extended TPC-W scenario (by user reviews, audio-video)
- 3 configuration
  - Low: All WI use only BASE guarantees
  - Medium: Mix between Base and ACID
  - High: All web interactions require ACID
- 3 experiments:
  - Scalability
  - Scale-Up and Down
  - Fault tolerance
- New metrics
  - Scalability (Correlation Coefficient)
  - Avg. Cost + Cost Var.
  - SLA Ratio



CloudBench - Work in progress

- Paper [DBTEST09]
- First results availalable for different architectures
  - Amazon SimpleDB
  - MySQL Cluster on Amazon EC2
  - Sausalito
  - *Consistency Rationing (in progress)*
  - *Cloudy (in progress)*
  - *Google App Engine (in progress)*

Other (selected) projects

- Barrelfish
  - new written-from-scratch OS kernel
  - targeted for multi- and many-core processor systems
- Rhizoma
  - constraint-based runtime system for distributed applications
  - self-hosting
- Concierge
  - blur the burden between mobile and cloud applications
- Xadoop
  - XQuery on Hadoop


tim.kraska@inf.ethz.ch