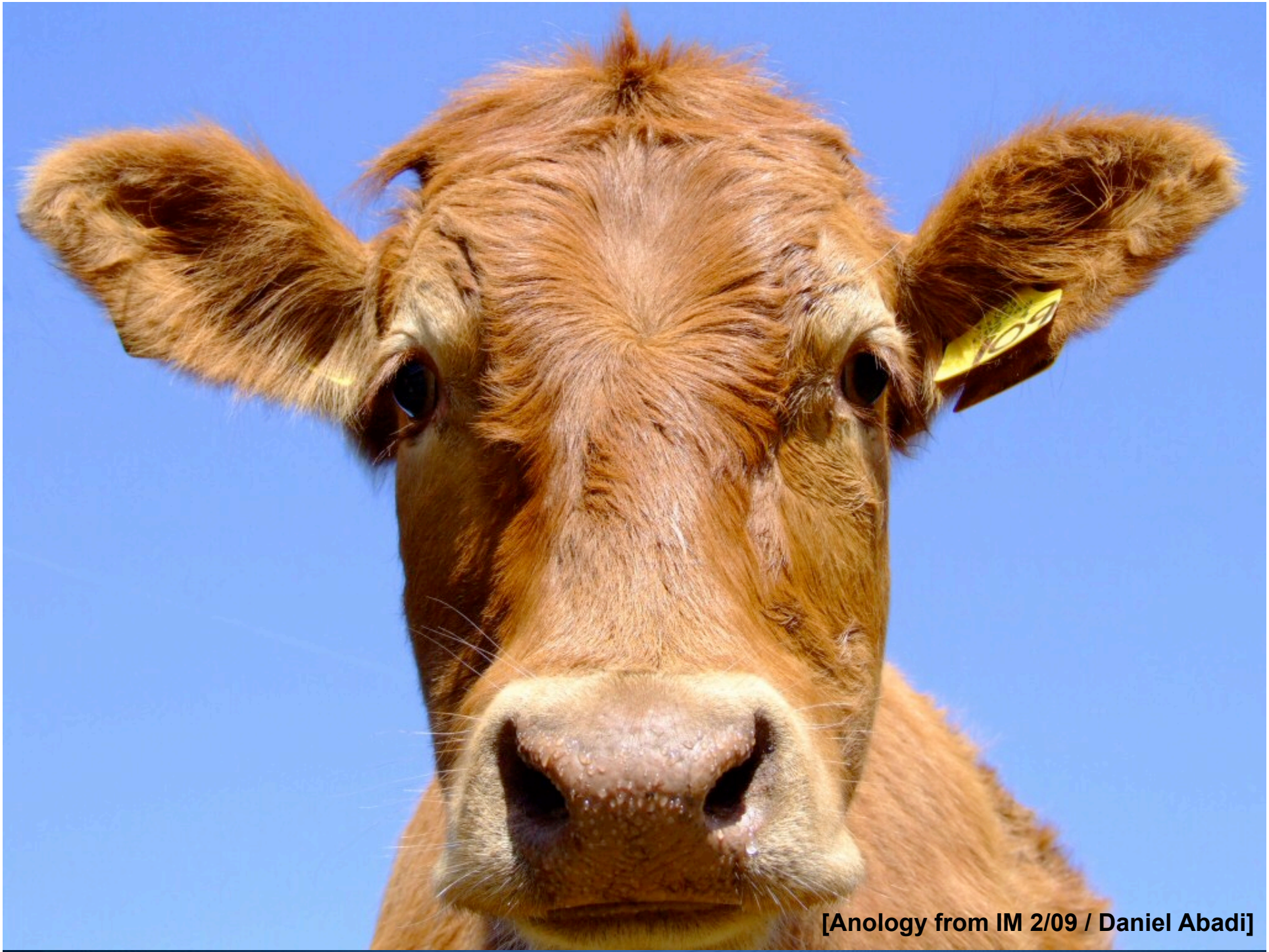
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

Towards a Benchmark for the Cloud

Carsten Binnig, Donald Kossmann, Tim Kraska, Simon Losing


[Anology from IM 2/09 / Daniel Abadi]

Do you want milk?

Buy a cow

- High upfront investment
- High maintenance cost
- Produces a fixed amount of milk
- Stepwise scaling

Buy bottled milk

- Pay-per-use
- Lower maintenance cost
- Linear scaling
- Fault-tolerant

Traditional DB vs. Cloud DB

Traditional DB

- High upfront investment
- Fixed amount of TRX
- Stepwise scaling
- High maintenance cost

Cloud DB

- Pay-per-use
- Linear scaling
- Fault-tolerant
- Lower maintenance cost

How to benchmark the milk supply?

- Milk production per cow
- Investment
- Maintenance cost

- Use relative metric:
  Price per liter

Traditional Benchmark

- Test fix setup
- Metrics typical:
  - Maximum performance
  - Cost per max. performance
- 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

Towards a benchmark for the cloud

Use relative metrics instead of maximum metrics

- Cost per TRX/WIPS
- Performance (e.g. Response time per TRX/WIPS)

Are we done?

What makes benchmarking the cloud difficult?




Varity of products

- Different features
- Different consistency guarantees (CAP theorem)
- Different lot size, varying price

What makes benchmarking the cloud difficult (ctd)

Ideally	Reality
Scalability ▪ Infinite ▪ The system adapts itself ▪ Maximum performance not reportable	▪ Hidden scalability limits / CAP ▪ E.g. MegaStore or MS SQL Data Services ▪ Adaption times?
Pay-Per-Use (Dollar) ▪ Cost increases linearly with usage ▪ Cost completely variable	▪ Step-wise functions / lot sizes ▪ Price plans
Fault Tolerance ▪ 100% availability ▪ Never loose data	▪ Single data center vs. multi-data center replication ▪ SLA?

Cloud Application Benchmark - Requirements

- Report on performance and cost
- Test the cloud charecteristics
- Test the complete application stack rather than single aspects
- Take the different cloud offerings into consideration
  - Platform as a Service / Infrastructure as a Service
  - Consistency guarantees
  - Replication strategies
  - ....

TPC-W

- Specifies a online bookstore
- Workload through simulated web interaction (WI)
- Main metrics:
  - Maximum Web Interactions Per Seconds (WIPS)
  - $/WIPS = (TC for 3 years - development) / maximum WIPS

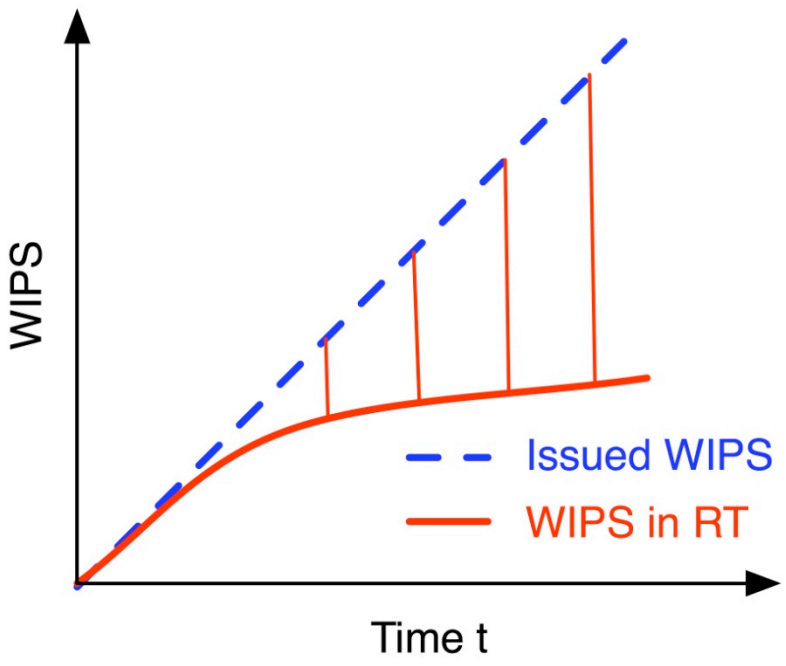
TPC-W is not appropriate for the cloud

- Benchmark requires ACID → Most cloud services provide less
- No maximum WIPS reportable → System should scale infinitely
- $/WIPS does not work → How to deal with lot sizes
- *Outdated web interactions → less writes, no user content etc.*

Ideas for a Cloud Application Benchmark

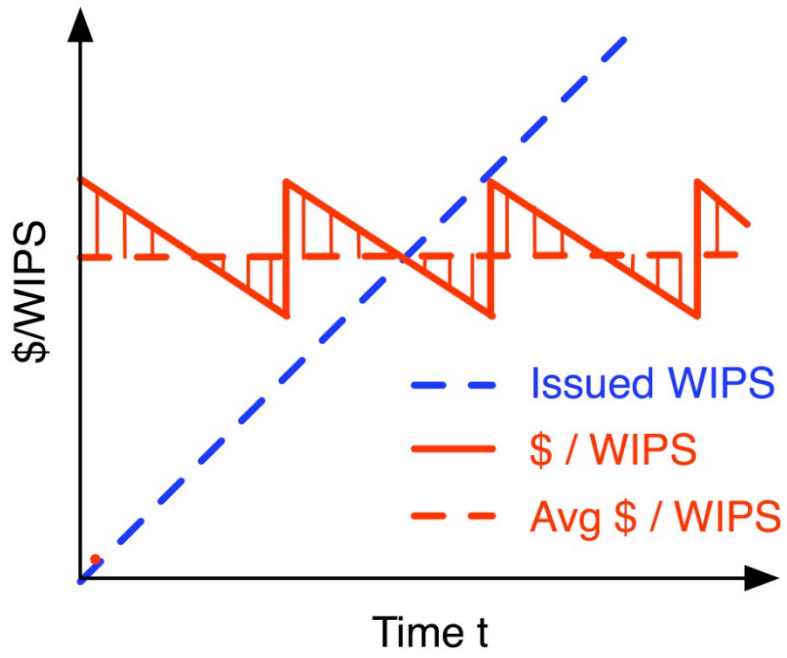
- Extended TPC-W scenario (by user reviews, audio-video)
- 3 configuration
  - Low: All WI use only BASE guarantees
  - Medium: Mix between Base and ACID
  - High: All web-interactions require ACID
- 3 experiments:
  - Scalability
  - Scale-Up and Down
  - Fault tolerance

Experiment 1: Scalability



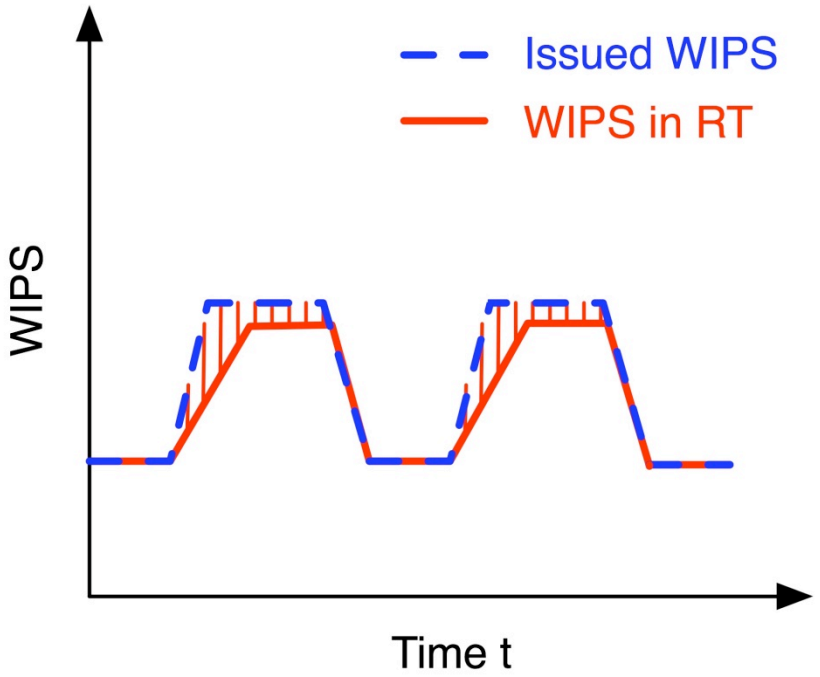
- Benchmark scale-up
- Increase issued WIPS against SUT over time
- Measure WIPS in allowed response time (RT)
- Metric:
  Correlation coefficient R^2
  between perfect linear scaling and WIPS in RT
  - 1 = perfect
  - 0 = constant behavior (no scaling)
- Stop experiment, if
  (WIPS in RT)/(Issued WIPS) < *X* or *time* > *Y* → report on time!

Experiment 1: Cost



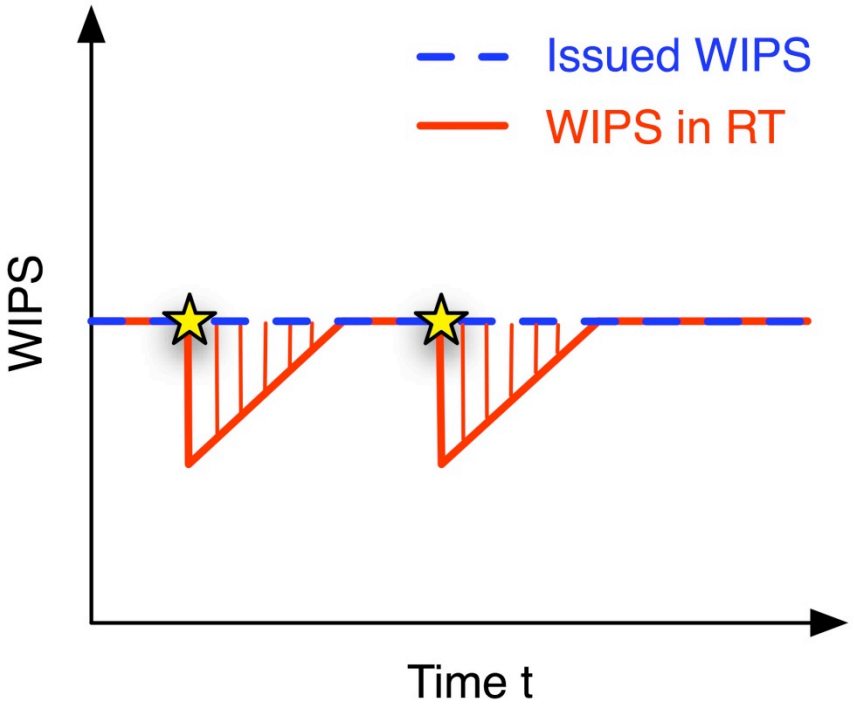
- Increase issued WIPS against SUT over Time
- Metric:
  - Average Cost **$/WIPS**
  - Standard deviation **S**
    - S = 0: perfect pay-per-use
    - S >> 0: traditional non-cloud senario

Experiment 2: Scale-Up/Down (Peaks)



- Measure scale-up and scale-down
- Metric:
  - Peak ratio =
    (WIPS in RT) / (Issued WIPS)
  - Average cost + Cost deviation
- Issues:
  - Fix load increase vs. different scenarios
  - Alternative metric: Elevation factor (use different load increases until peak ratio < 1)

Experiment 3: Fault tolerance



- Measure failure behavior
- Idea: Fail X percent (randomly) of the resources
- Metric:
  - Fault ratio:
    (WIPS in RT) /(issued WIPS)
  - Cost + cost variance
- Alternative metric: Maximum failure percentage until fault ratio < 1
- Issues:
  - Hard to measure
  - Not always possible
  - Might not be fair!

Open Questions

- Setting the load increase factor
  - Fix value
  - Different experiments
- Setting the base load
- Setting of Service Level Agreements
- Development time
  - For example, PaaS vs. combination of IaaS
- Client location


tim.kraska@inf.ethz.ch