Technische Universität Berlin

Enterprise Computing – Cloud Service Benchmarking

Prof. Stefan Tai

ISEngineering
Information Systems Engineering

Introduction	Thu, Oct 20 Introduction	
Cloud Services	Thu, Oct 27 + Thu, Nov 3 + Wed, Nov 9 Web Engineering Fundamentals Cloud Fundamentals Microservices/DevOps	
Cloud Storage	Thu, Nov 17 Intro NoSQL	Thu, Nov 24 Systems 1
	Thu, Dez 1 Systems 2	
Cloud Service Benchmarking	Thu, Dez 8 Qualities, Benchmarking 1	Thu, Dez 15 Benchmarking 2
Assignment (6 weeks)	Thu, Dec 15 Kick-Off	Thu, Feb 9 End of Assignment
Wrap-Up	Thu, Feb 16 Q&A	

ISEngineering
Information Systems Engineering

Next

Three “unconventional” cloud benchmarking experiments conducted by the TUB-ISE research group:

1. Consistency benchmarking
2. Benchmarking security-performance trade-offs
3. Demystifying Web API usage

Consistency Benchmarking

Sources:

[Bermbach2011] D. Bermbach, S. Tai. "Eventual Consistency: How soon is eventual? An evaluation of Amazon S3's consistency guarantees", ACM MW4SOC, 2011

[Bermbach2014] D. Bermbach, S. Tai. "Benchmarking Eventual Consistency: Lessons Learned from Long-Term Experimental Studies", IEEE IC2E, 2014

Motivation

Developing applications on top of only eventually consistent cloud storage services is difficult and requires a change in mindsets

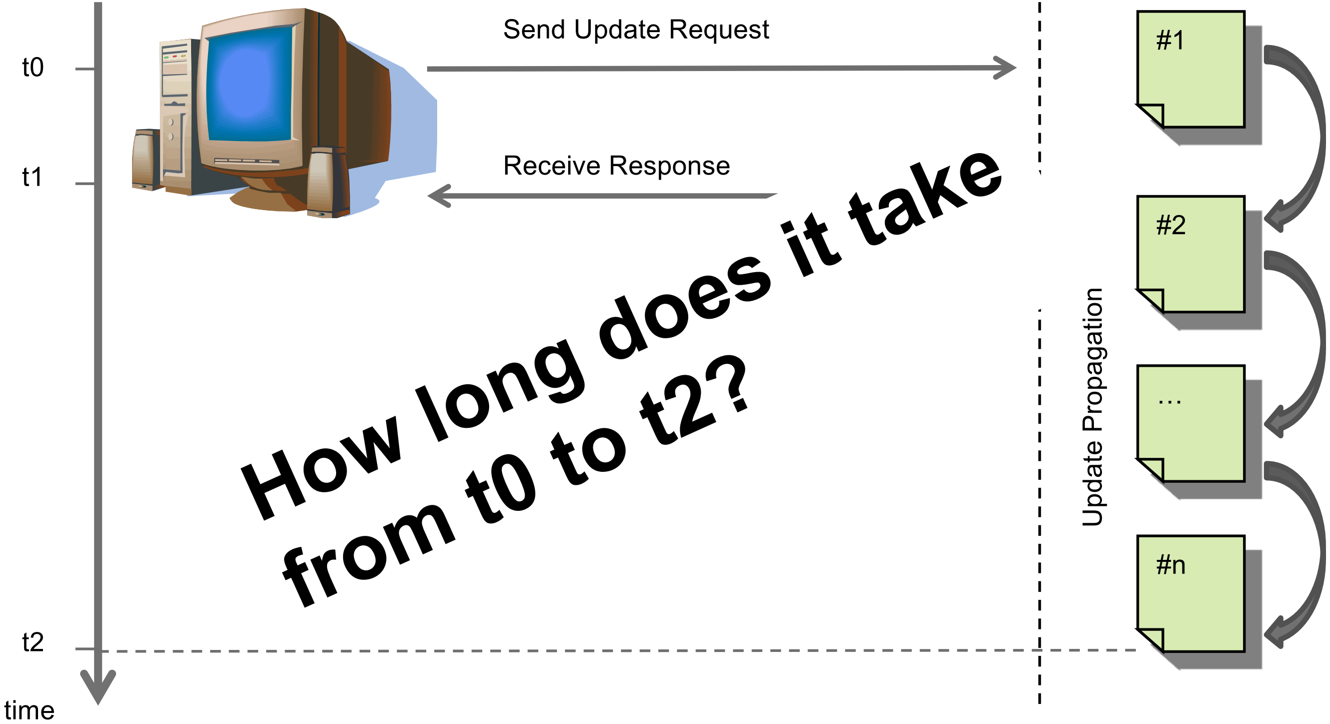
Most applications can tolerate a certain degree of uncertainty as long as they know about the extent

In some cases eventual consistency is just not acceptable but changing the behavior of storage systems requires knowledge of their guarantee interna

Can we measure the degree of eventuality?

Source: [Bermbach2011]

Experiment



Source: [Bermbach2011]

Approach

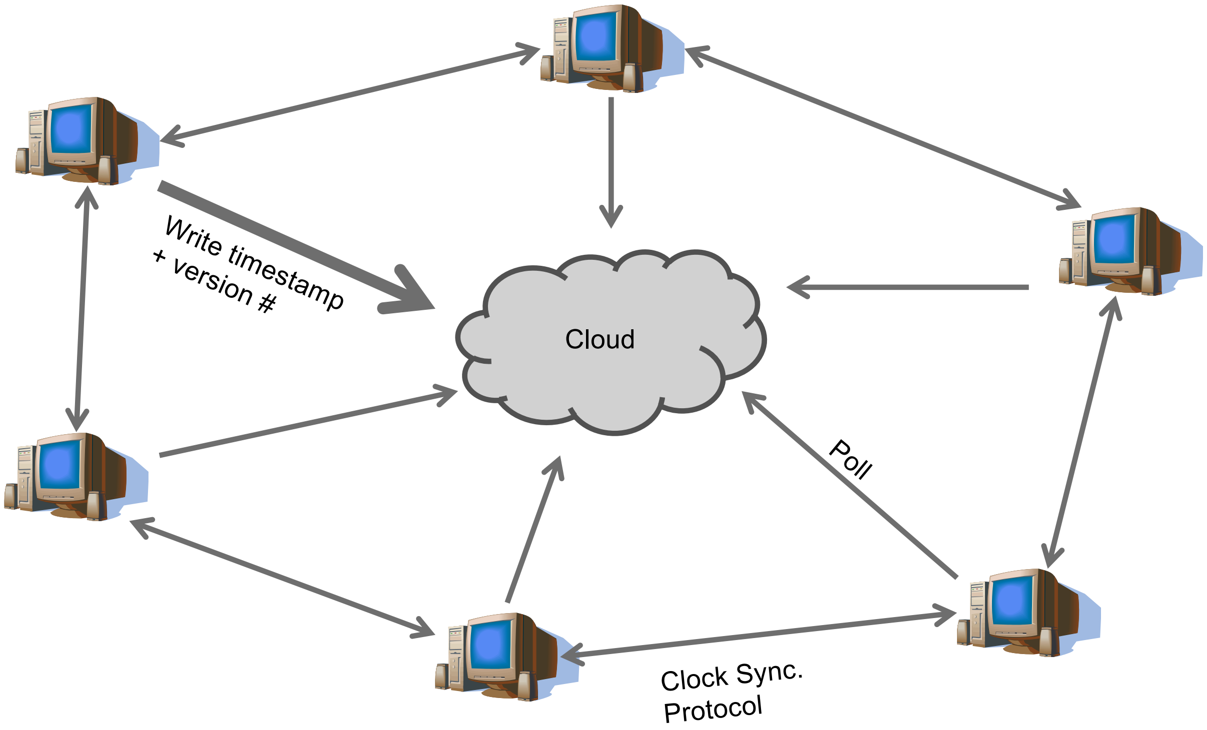
Trivial: Add application hooks to the Cloud storage system which notify the client when all replica have been written
Problem: Usually only blackbox access

Idea: For a client it does not matter if all replica have been written – it only matters whether the Cloud will return inconsistent data

- Write data to the Cloud and poll until only consistent data is returned
- Polling must be distributed to circumvent caching and continuously reading the same replica

Source: [Bermbach2011]

Setup



Source: [Bermbach2011]

Example

14:57:38h: A writes “1 14:57:38h“ to S3.
14:57:39h: B still reads version 0.
14:57:40h: C reads the timestamp “1 14:57:38h“ and sends a “2s“ message to A.
14:57:41h: B still reads an version 0.
14:57:42h: D reads the timestamp “1 14:57:38h“ and sends a “4s“ message to A.
14:57:43h: B reads the timestamp “1 14:57:38h“ and sends a “5s“ message to A.
14:57:44h: A has received all messages and defines the inconsistency window as 5s.
14:57:45h: A writes the next timestamp.
…

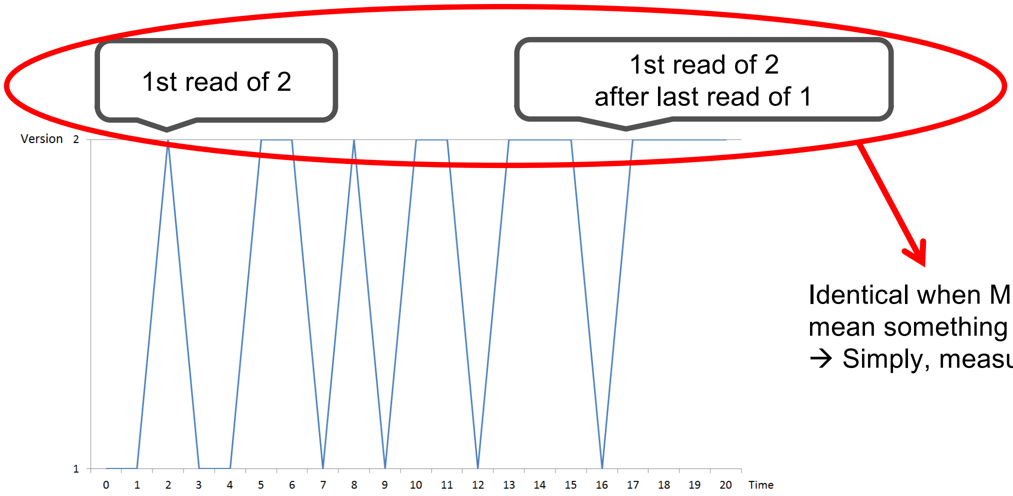
Source: [Bermbach2011]

Problem: Monotonic Read Consistency (MRC)

MRC means: If some client has read version N it will subsequently never read a version < N

- cf. Vogels “Eventually Consistent“ article

Approach needs to be adapted when monotonic read consistency is violated, example:



Identical when MRC is preserved. If not: Both mean something different!
→ Simply, measure last stale read

Source: [Bermbach2011]

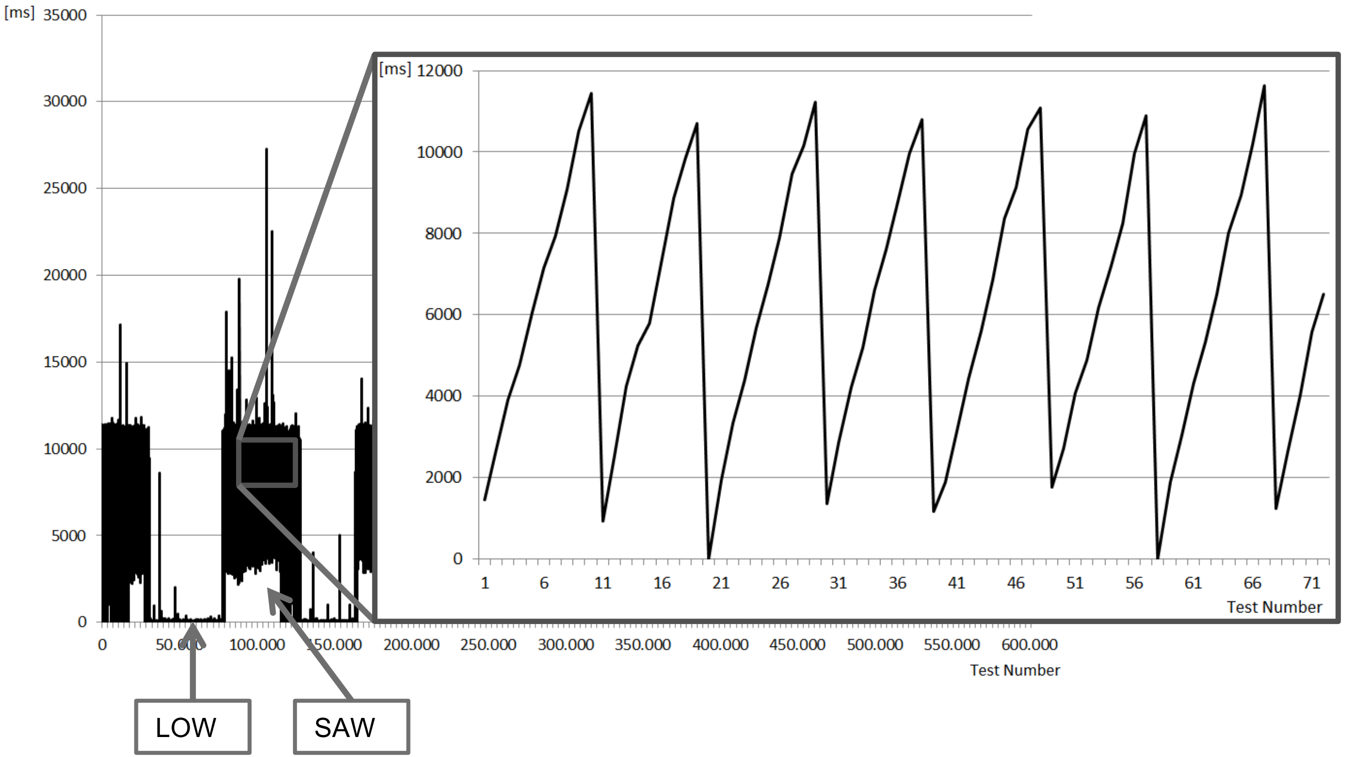
Additional Comments

Setup requires huge numbers of requests which in itself may influence the result
Depending on the storage system it is necessary to have access to machines distributed worldwide
Side effect: We can measure read availability and latency

➔ What do the results look like?

Source: [Bermbach2011]

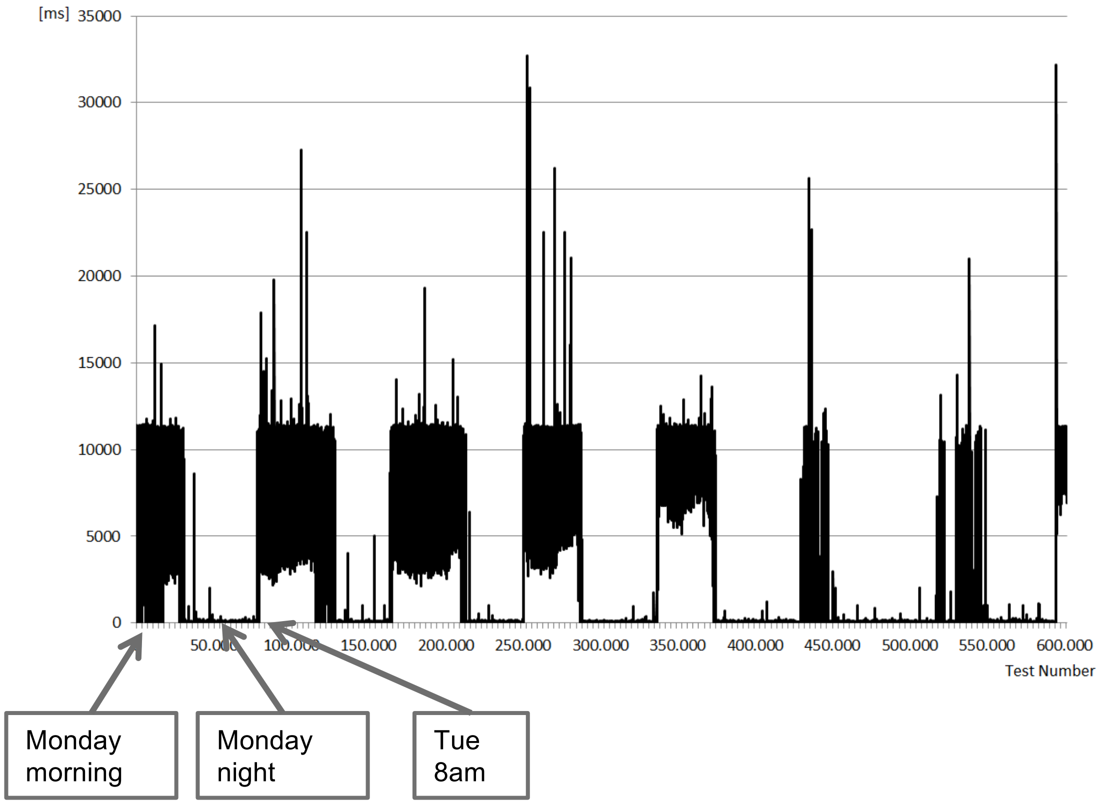
Evaluation of S3 – Staleness Measurements



Source: [Bermbach2011]

ISEngineering
Information Systems Engineering

Evaluation of S3 – Staleness Measurements



Source: [Bermbach2011]

Evaluation of S3 in Comparison to Cassandra

Amazon S3[1]

Two different periodicities

12% violations of monotonic read consistency

One availability zone out of three usually lags behind (in 50% of all tests)

> 99 % of all LOW writes create consistent data after 175ms

Read availability > 8 nines

Apache Cassandra[2]

Geometric distribution

0.0006% violations of monotonic read consistency

No influence of geographic distribution

> 99% of all writes create consistent data after 35ms

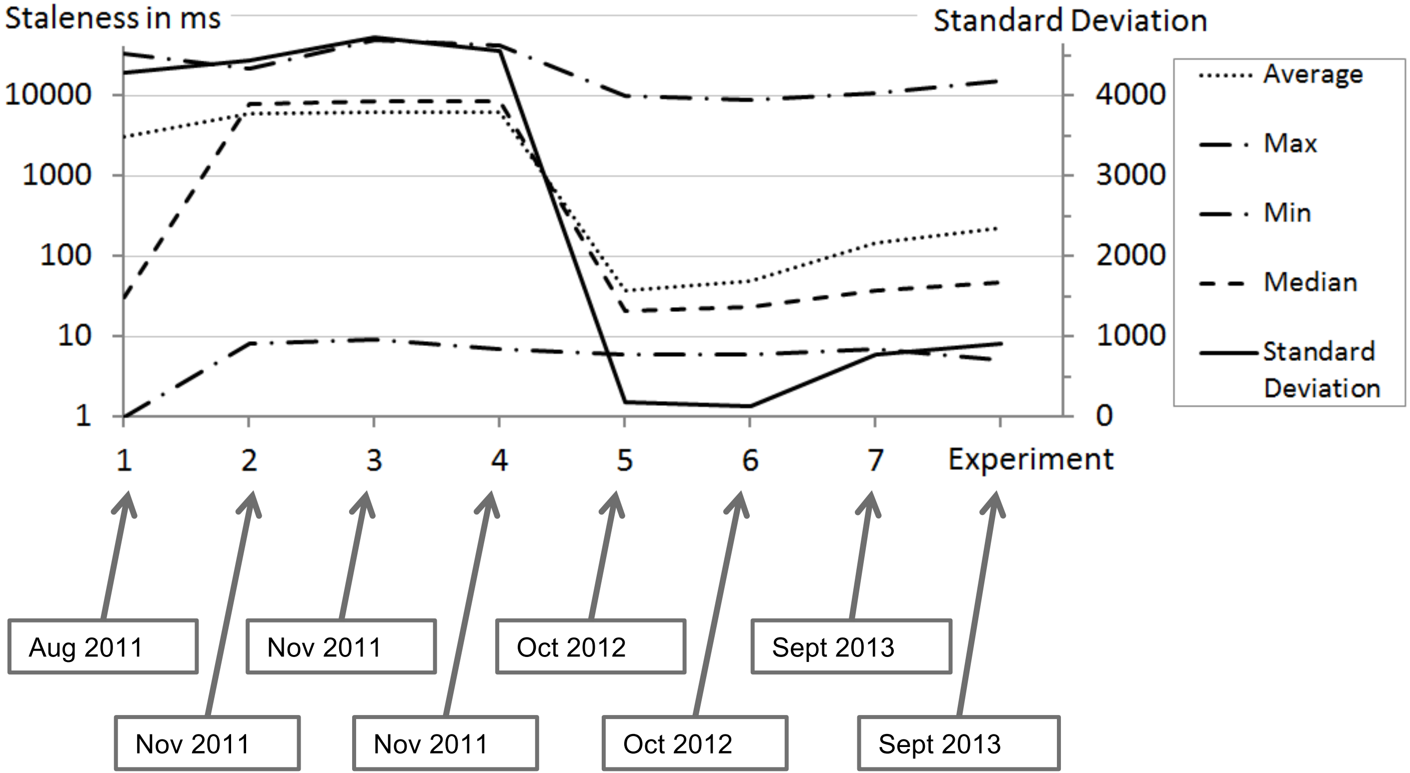
Read availability 100%

[1] Setup: high redundancy with replication over 3 availability zones; test duration 7 days

[2] Setup: deployed on 3 large EC2 instances in different availability zones; Consistency level ONE; 3 replica; test duration 24h

Source: [Bermbach2011]

Development of S3's Consistency Behavior



Source: [Bermbach2014]

Benchmarking Cloud Security-Performance

References:

[MetaStorage] Bermbach, Klems, Tai, Menzel: “MetaStorage: A Federated Cloud Storage System to Manage Consistency-Latency Tradeoffs”, Proc. of CLOUD‘11, 2011, pp. 452-459.

[MimoSecco] Achenbach, Gabel, Huber: “MimoSecco: A Middleware for Secure Cloud Storage”, Proc. of Improving Complex Systems Today, London, 2011, S. 175-181.

[Müller14] Müller, Bermbach, Pallas, Tai: “Benchmarking the Performance Impact of Transport Layer Security in Cloud Database Systems”, Proc. of IC2E’14, 2014.

[Sandhu03] Sandhu: “Good-enough security”, Internet Computing, IEEE, 2003, 7, 66-68.

Starting Point

“Good-enough security” [Sandhu 03]

Understanding trade-offs in specific cloud systems

Prototyping, experimentation, and assessment of different Cloud storage systems

ISEngineering
Information Systems Engineering

Secure Cloud Storage Systems

Security Goal	Confidentiality	Integrity	Availability	Access control	Accountability	Non-Repudiation	Authenticity	Privacy	...
Security Mechanism	Encryption	Access control	Data integrity mechanisms	Digital signature	Authentication exchange mechanisms	Replication	Fragmentation	Routing control	...
Deployment Model	Public	Private	Hybrid						
Storage Type	Relational Stores	Column-oriented Stores	Key-Value Stores	Document Stores	...				

A conceptual security framework as a feature model.

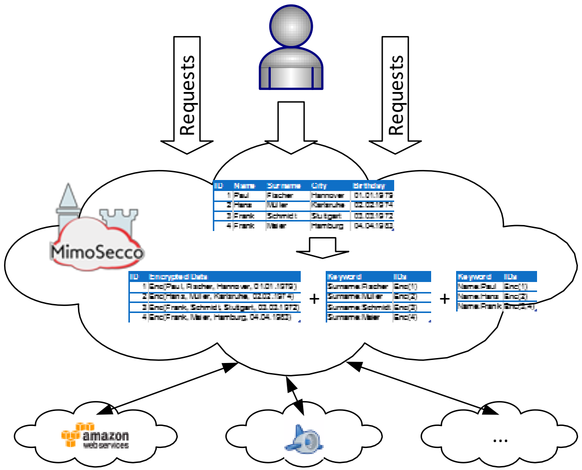
Two guiding research questions:

1. Which feature configurations apply to cloud storage systems?
2. „How good is a feature configuration?“, that is, what are the trade-offs for a given configuration, especially with regards to system performance?

ISEngineering
Information Systems Engineering

Prototype/Experiment 1: MimoSecco

Security Goal	Confidentiality	Integrity	Availability	Access control	Accountability	Non-Repudiation	Authenticity	Privacy	...
Security Mechanism	Encryption	Access control	Data integrity mechanisms	Digital signature	Authentication exchange mechanisms	Replication	Fragmentation	Routing control	...
Deployment Model	Public	Private	Hybrid						
Storage Type	Relational Stores	Column-oriented Stores	Key-Value Stores	Document Stores	...				



MimoSecco is a relational store (SQL-queries)

Uses encryption, fragmentation, and routing control mechanisms to hide associations in the data from the providers (honest-but-curios attacker)

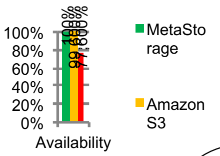
Prototype/Experiment 2: MetaStorage

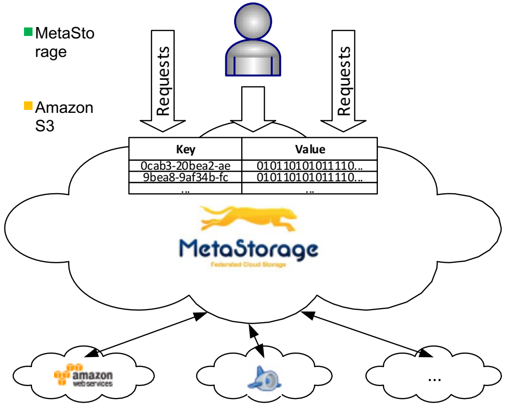
Security Goal	Confidentiality	Integrity	Availability	Access control	Accountability	Non-Repudiation	Authenticity	Privacy	...
Security Mechanism	Encryption	Access control	Data integrity mechanisms	Digital signature	Authentication exchange mechanisms	Replication	Fragmentation	Routing control	...
Deployment Model	Public	Private	Hybrid						
Storage Type	Relational Stores	Column-oriented Stores	Key-Value Stores	Document Stores	...				

MetaStorage is a
- high-available key-value store

Leverages the concept of horizontal
- cloud storage federation

Different routing and replication
- techniques increase availability and ease migration compared to single provider cloud storage services





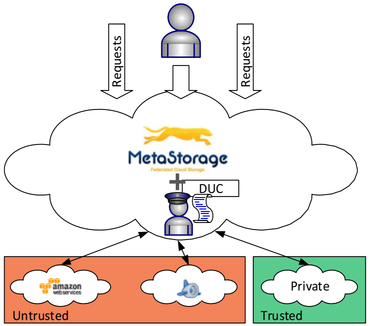
Prototype/Experiment 3: UC4MetaStorage

Security Goal	Confidentiality	Integrity	Availability	Access control	Accountability	Non-Repudiation	Authenticity	Privacy	...
Security Mechanism	Encryption	Access control	Data integrity mechanisms	Digital signature	Authentication exchange mechanisms	Replication	Fragmentation	Routing control	...
Deployment Model	Public	Private	Hybrid						
Storage Type	Relational Stores	Column-oriented Stores	Key-Value Stores	Document Stores	...				

Integration of a Data-driven Usage Control (DUC) framework into MetaStorage

Fine-grained policies for transparent and compliant data distribution and replication with

- Temporal constraints, e.g., “Data must be deleted after two years”
- Spatial constraints, e.g., “Data must not leave the European Union”
- Qualitative constraints, e.g., “Data which is stored at provider A must be encrypted with AES256”



ISEngineering
Information Systems Engineering

TLSBench

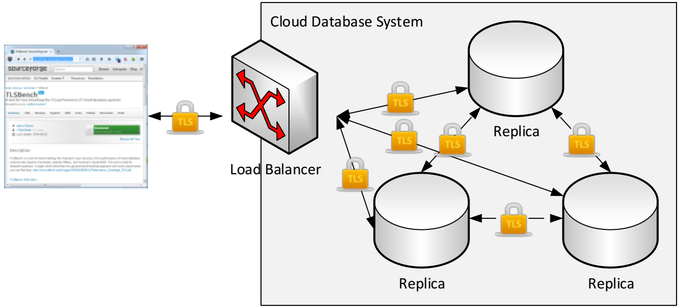
Enabling Transport Layer Security (TLS) impacts the performance of cloud storage systems

TLSBench [Müller 14] is a tool for benchmarking the performance impact of TLS in cloud storage systems

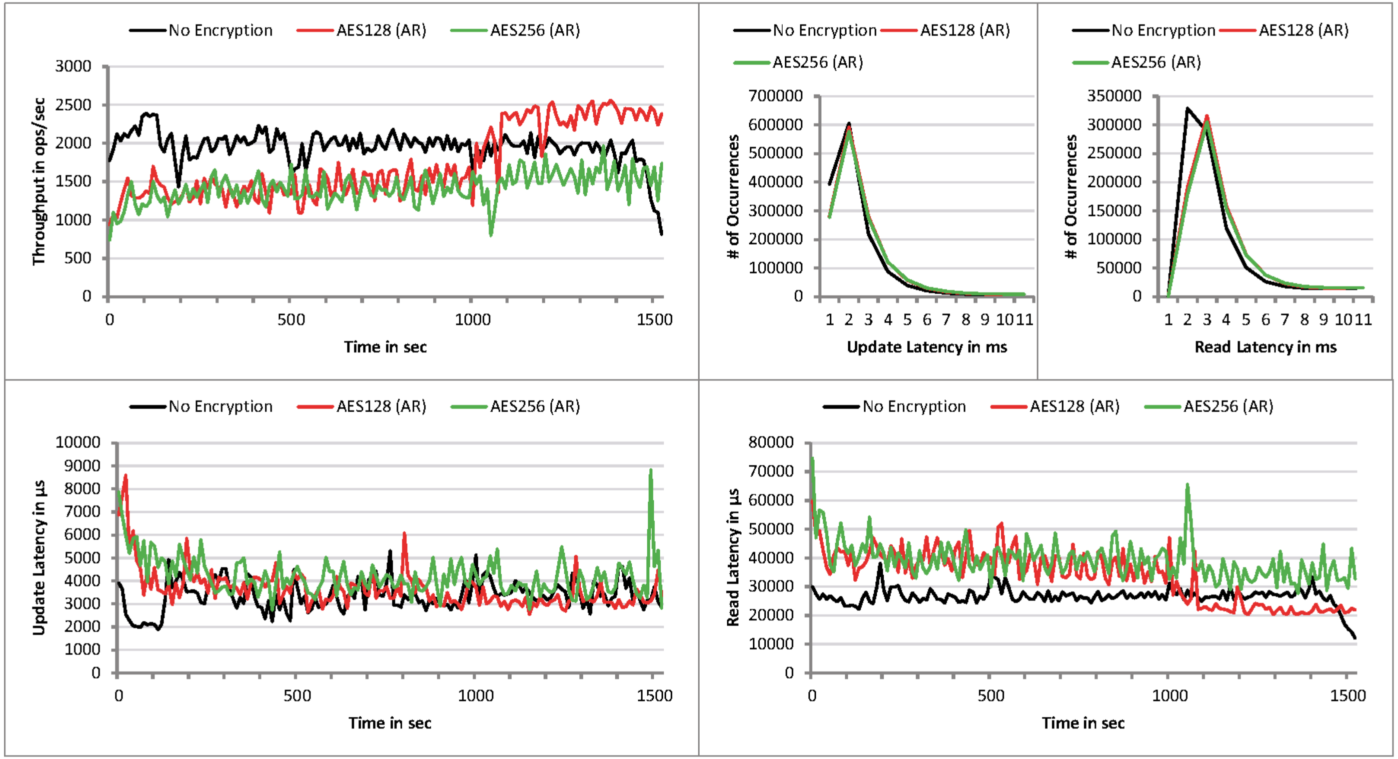
Benchmarking from a client's perspective

Can be applied to different cloud storage systems and different communication types of cloud storage systems

Partial automation of measurements with fine-grained control of the workload



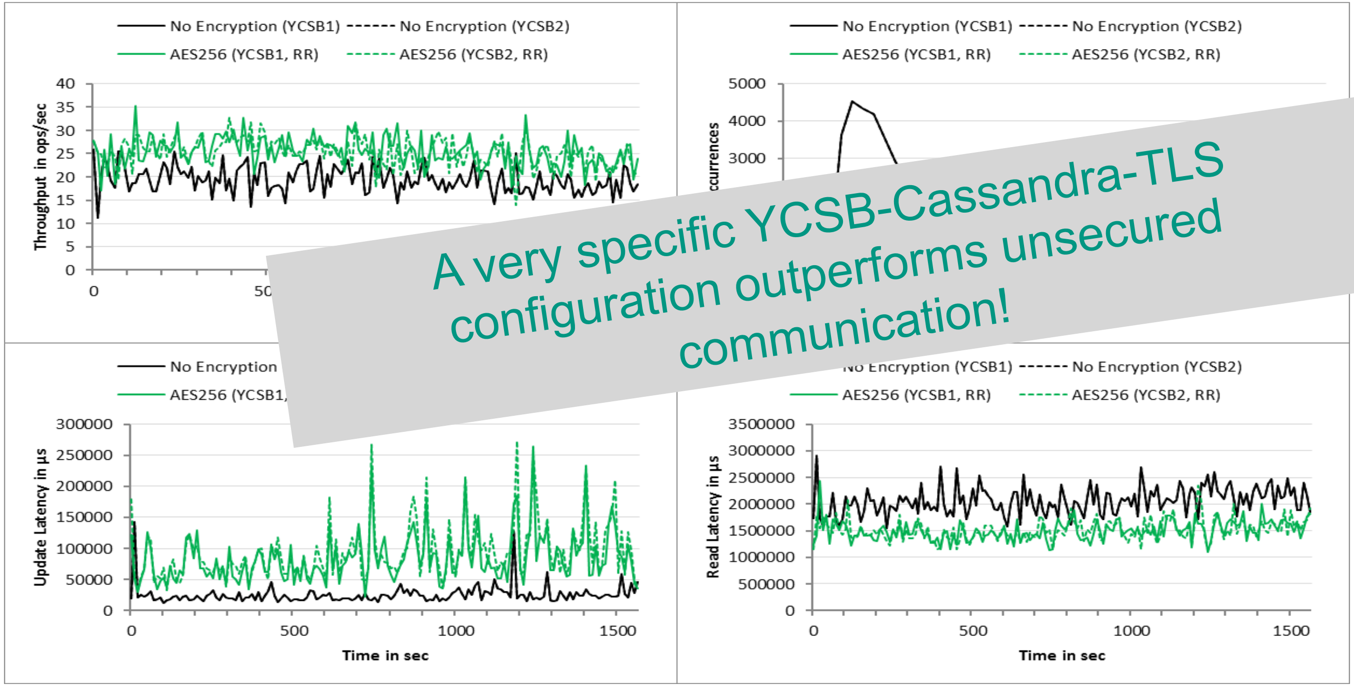
TLSBench Sample Benchmarking Results (1/2)



- 3-node Cassandra cluster (EC2 m.1 large instances, Software RAID-0)
- Consistency-level: ONE
- Replication Factor: 1
- Initial load: ca. 30 GB
- YCSB workload A, Field size: 1,000B, Operation count: 3,000,000
- TLS V1.0 protocol

TLSBench Sample Benchmarking Results (2/2)

Technische Universität Berlin



A very specific YCSB-Cassandra-TLS configuration outperforms unsecured communication!

- 3-node Cassandra cluster (EC2 m.1 large instances, Software RAID-0)
- Consistency-level: QUORUM
- Replication Factor: 2
- Initial load: ca. 150 GB
- YCSB workload A, Field size: 1,000KB, Operation count: 40,000
- TLS V1.0 protocol

Demystifying Web API Usage

Source:
[BW2016] D. Bermbach, E. Wittern. ICWE, 2016

Why Web API Benchmarking?

- Increasingly popular
- Used for core features of web applications and mobile apps
- Geo-mobile users

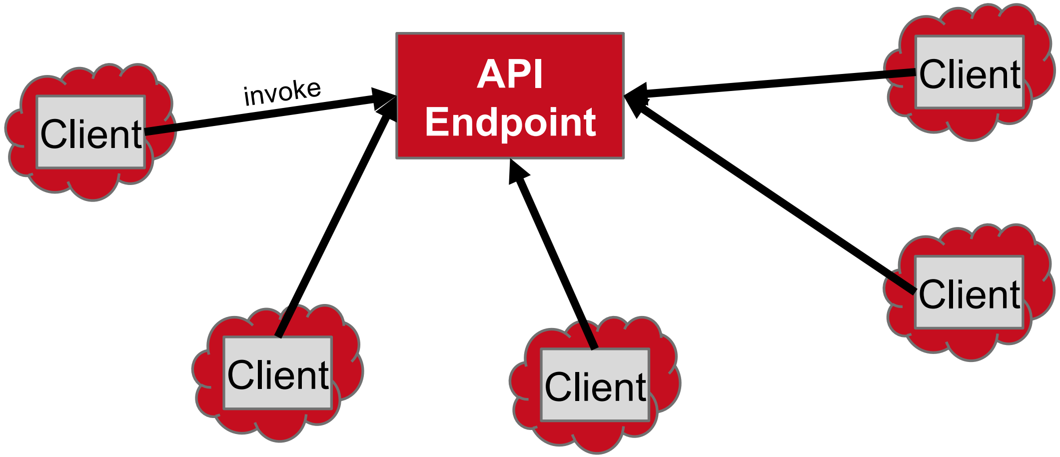
- Quality is considered as a given
- No insight into API implementation
- No control over API
- Local testing only

Mismatch!

Benchmarking can provide insights into web API quality
Its results can be used to design applications that can tolerate poor quality

Approach

1. Frequently invoke APIs over a long period of time
2. Use different protocols (HTTP, HTTPS, ICMP) for this
3. Track availability, latency, and security settings
4. Use geo-distributed measurement clients based on cloud servers



ISEngineering
Information Systems Engineering

Experiment Setup

One measurement client each in AWS regions ireland, oregon, sao-paulo, singapore, sydney, tokyo, us-east

Test period: Aug 20 to Nov 20, 2015

Every API was invoked every 5 minutes through all supported protocols

Benchmarked APIs:
Amazon S3, Apple iTunes, BBC, ConsumerFinance, Flickr, Google Books, Google Maps, MusicBrainz, OpenWeatherMap, Postcodes.io, Police.uk.co, Spotify, Twitter, Wikipedia, Yahoo

Analysis Phase

- Check results for
- Number and geo-location of API datacenters (correlation of latency values across regions, actual latency values by region)
- API deployment setup (correlation or non-correlation of availability results for HTTP, HTTPS, and ICMP)
- API provider‘s choices for security/performance tradeoff (selected cipher suites for HTTPS)
- …

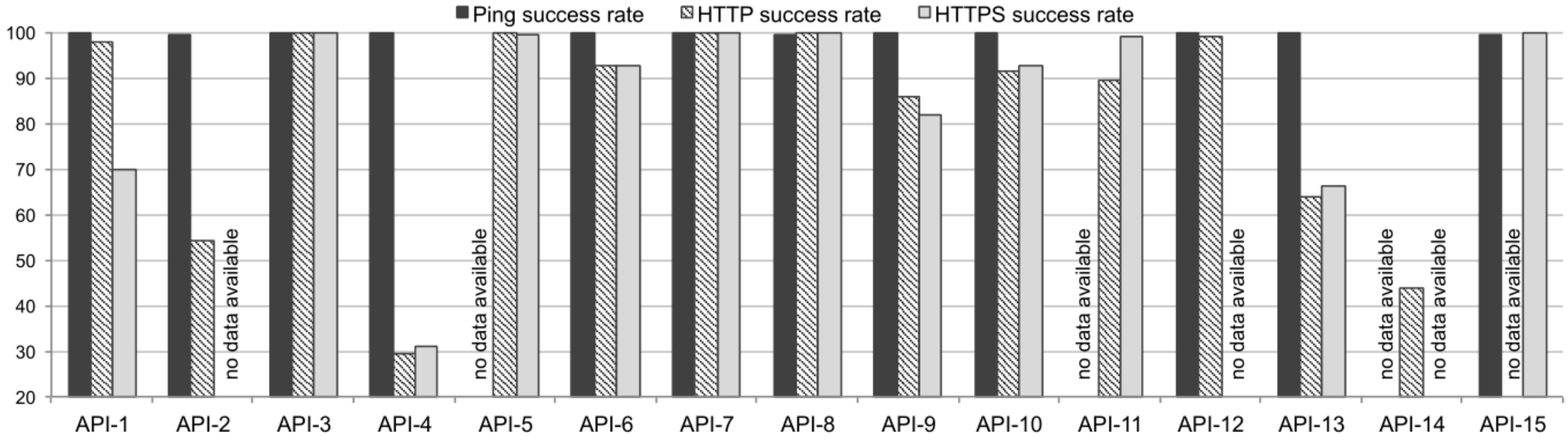
Results

Have been anonymized.

Overall:

- High volatility within regions
- Extensive differences of regions
- Some endpoints became permanently unavailable during the benchmark
- HTTP and HTTPS typically use different endpoints

Examples: Availability



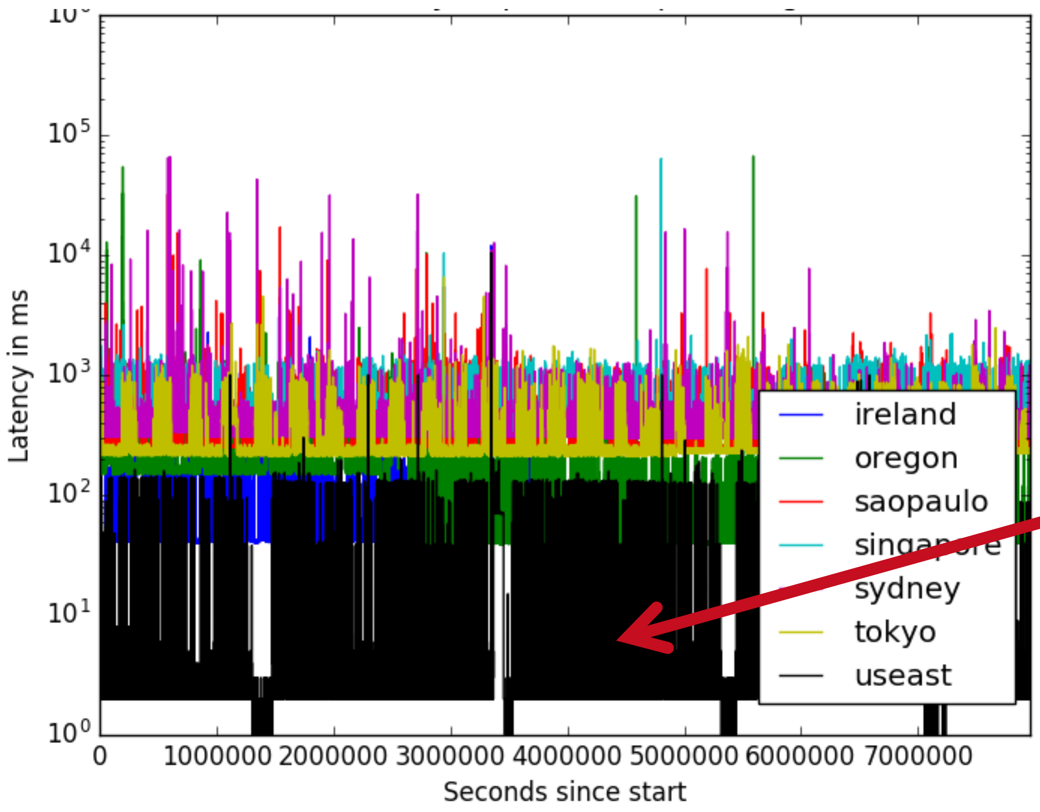
ISEngineering
Information Systems Engineering

Examples: Availability (cont.)

Endpoint	Successability [%] Ping	HTTP	HTTPS	Days with HTTP/HTTPS Successability <50% Ireland	Oregon	Sao Paulo	Singapore	Sydney	Tokyo	US East	Sum
API-1	99.97	97.80	69.73	11/11	-/72	-/-	-/-	3/29	-/-	83/-	14/195
API-2	99.35	54.16	-	43/-	43/-	43/-	43/-	43/-	43/-	43/-	301/-
API-3	99.83	99.98	99.98	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
API-4	99.86	29.43	31.08	64/64	64/64	78/64	64/64	64/64	64/67	64/64	462/451
API-5	-	99.97	99.40	-/4	-/-	-/-	-/-	-/-	-/-	-/-	-/4
API-6	99.88	92.59	92.58	49/49	-/-	-/-	-/-	-/-	-/-	-/-	49/49
API-7	99.96	99.98	99.98	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
API-8	99.33	99.96	99.96	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
API-9	99.75	85.79	81.90	-/-	-/-	-/89	-/-	6/-	57/-	-/-	63/89
API-10	99.94	91.37	92.56	49/49	-/-	-/-	-/-	-/-	-/-	-/-	49/49
API-11	-	89.44	99.13	-/-	66/-	-/-	-/-	3/6	-/-	-/-	69/6
API-12	99.79	98.86	-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
API-13	99.77	63.81	66.17	-/-	92/92	-/-	-/-	70/70	3/58	71/-	236/220
API-14	-	43.75	-	-/-	91/-	92/-	-/-	92/-	88/-	4/-	367/-
API-15	99.23	-	99.96	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-
Sum				216/177	356/228	213/153	107/64	281/169	255/125	265/64	1610/1063

Out of 92 days!

Example: Latency, HTTP



Deployment is close to us-east

Example: Latency, HTTP (cont.)