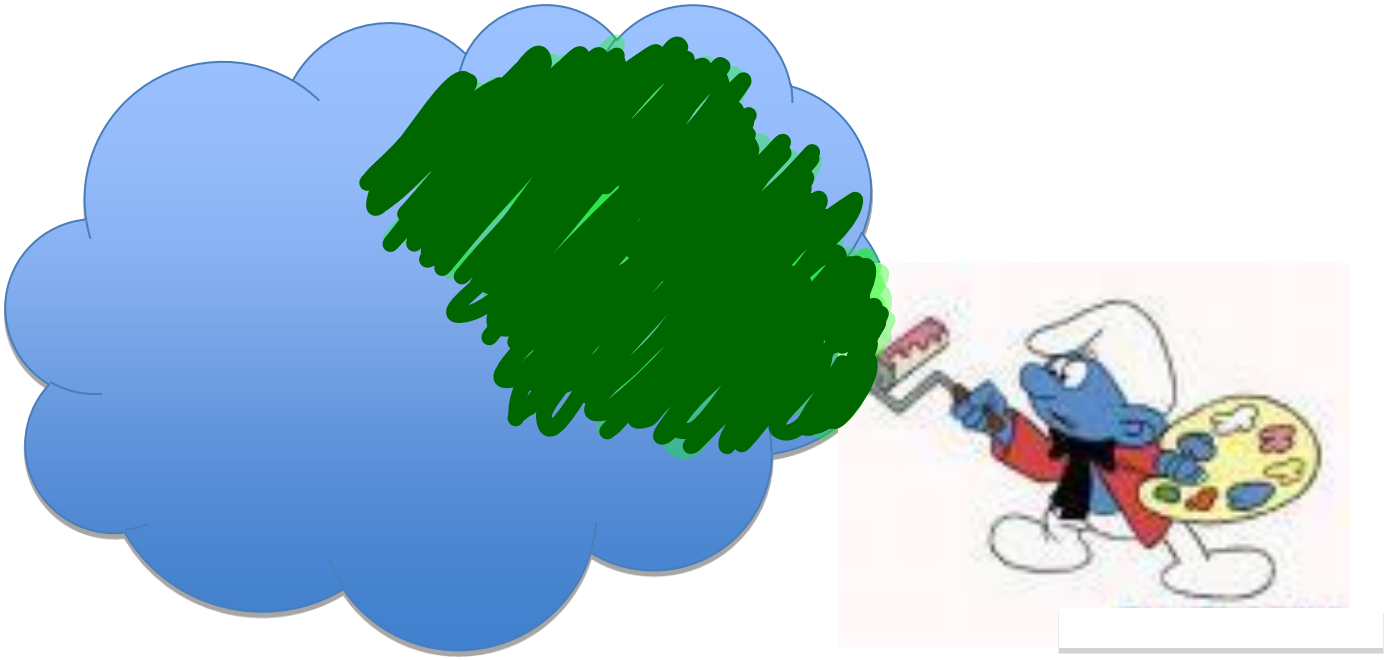


What is new in the cloud?
- A Database Perspective –

Donald Kossmann

Systems Group, ETH Zurich

http://systems.ethz.ch

Acknowledgments

Reference

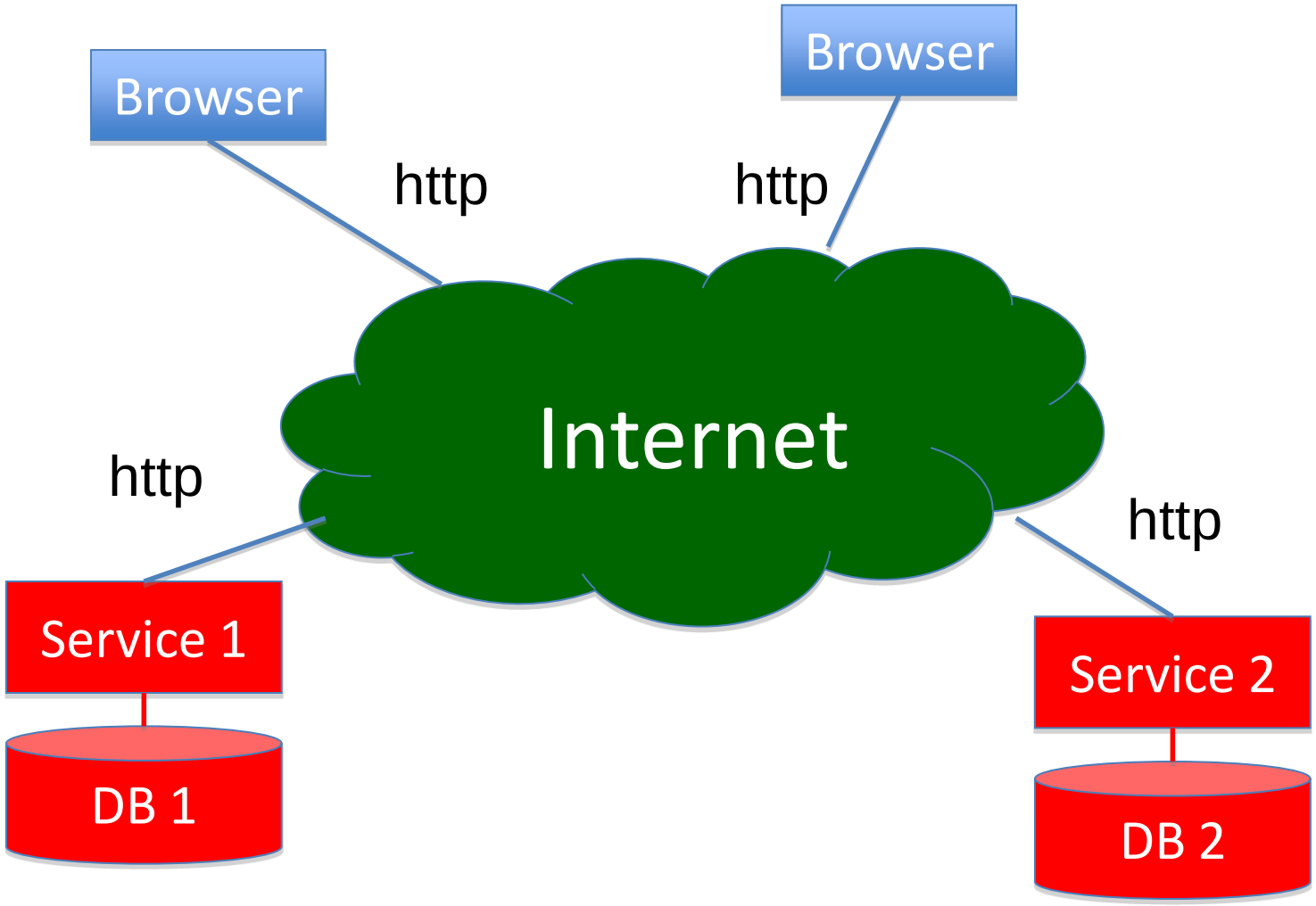
- Kossmann, Kraska: *Data Management in the Cloud: Promises, State-of-the-art, and Open Questions.* Datenbank Spektrum, 2010.
  - *http://www.pubzone.org/dblp/journals/dbsk/KossmannK10*
  - (Published by Springer Open Access.)

- (Summary of several talks and papers of the DB community on cloud computing.)

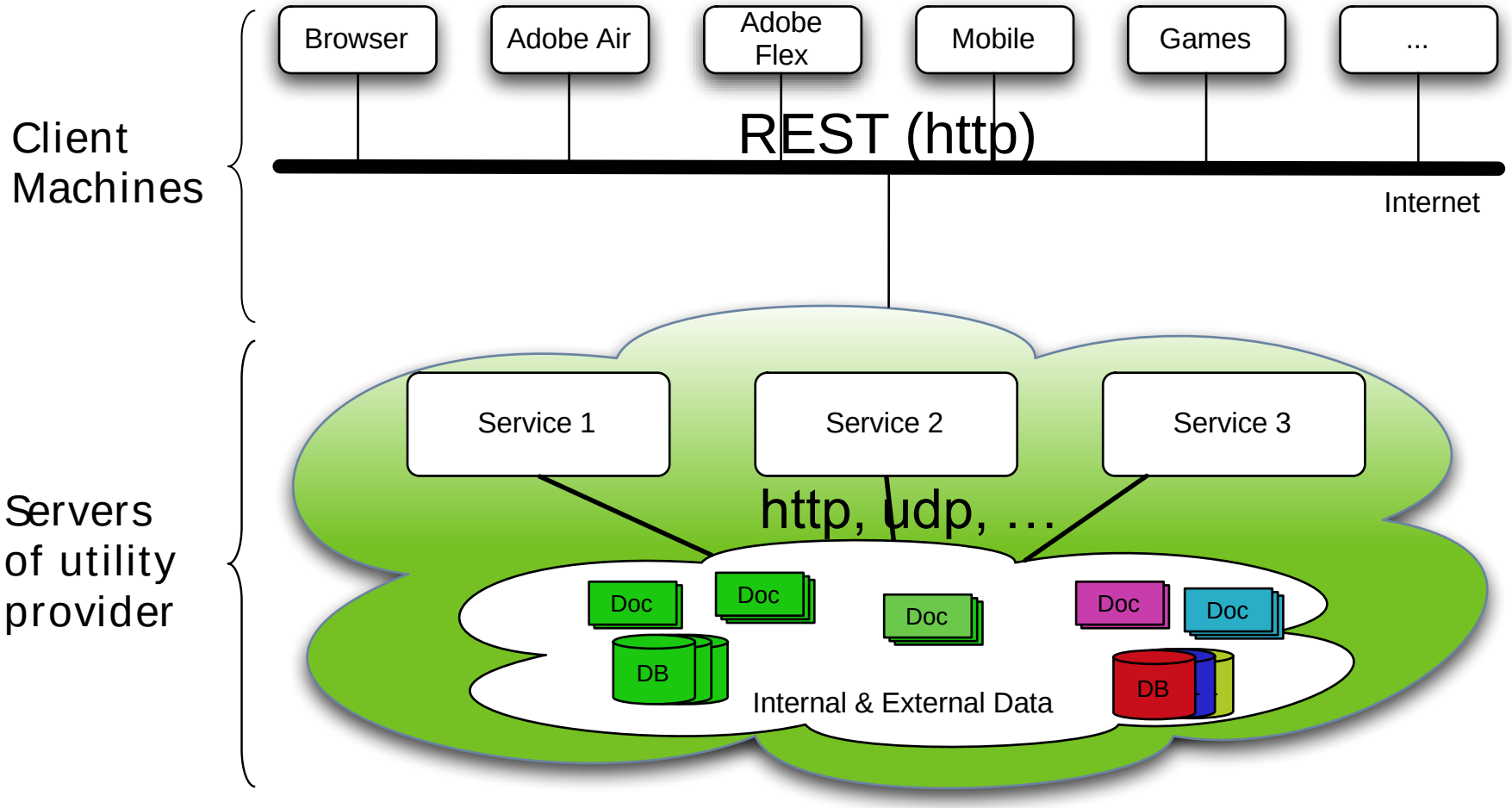
Agenda

- Promises of Cloud Computing
- Benchmarking the State-of-the Art [SIGMOD 2010]
  - Amazon, Google, Microsoft
- Towards a Silver Bullet [ICDE 2010]
  - Asking the right questions

The Cloud 15 Years Ago



The Cloud Today



Promises of Cloud Computing

- Cost
  - reduce cost
    - utilization, commoditization, optimization
    - prevention of failures, having failures
  - „pay as you go“ (cap-ex vs. op-ex)
  - easy to test!

- Time to market
  - avoid unnecessary steps
    - HW provisioning, puchasing, testing
  - difficult to test!
    - can only test scalability / elasticity

Cow vs. Supermarket

Cow	Supermarket: Bottle of Milk
Big upfront investment	Small continous expense
Produces more than you need	Buy what you need
Uses up resources	Less resources needed
Maintenance needed	No maintenance
Unpleasant waste product	Waste is not my problem

[Abadi 2009], [Kraska 2009]

Questions to ask

- Questions you ask your supermarket
  - How expensive is one litre of milk?
  - Is the price always the same?
  - What if I have a big party? Can you deliver?
- Questions you do not ask
  - How fast is the supermarket?
  - Good enough is good enough!

What to optimize?

Feature	Traditional	Cloud
Cost [$]	**fixed**	**optimize**
Performance [tps, secs]	optimize	fixed
Scale-out [#cores]	**optimize**	**fixed**
Predictability [σ($)]	**-**	**fixed**
Consistency [%]	fixed	???
Flexibility [#variants]	-	optimize

Put $ on the y-axis of your graphs!!!

[Florescu & Kossmann, SIGMOD Record 2009]

Agenda

- Promises of Cloud Computing
- **Benchmarking the State-of-the Art** **[SIGMOD 2010]**
  - **Amazon, Google, Microsoft**
- Towards a Silver Bullet [ICDE 2010]
  - Asking the right questions

What is out there?

- Amazon
  - IaaS: S3 and EC2
  - PaaS: SQS, SimpleDB, RDS
- Google App Engine
  - PaaS: Integrated DB, AppServer, and Cache
- Microsoft Azure
  - PaaS: Windows Azure, SQL Azure, .Net Azure
- Gazillions of start-ups (e.g., 28msec ☺)
- Research prototypes (e.g., Crescando ☺)

What is out there?

- Amazon
  - IaaS: S3 and EC2
  - PaaS: SQS, SimpleDB, RDS
- Google App Engine
  - PaaS: Integrated DB, AppServer, and Cache
- Microsoft Azure
  - PaaS: Windows Azure, SQL Azure, .Net Azure

- Gazillions of start-ups (e.g., 28msec ☹)
- Research prototypes (e.g., Crescando ☹)

Scope of this study

- Workloads: **Focus on OLTP**
  - OLAP under heavy debate by others
  - streaming not addressed yet (~ OLTP)
  - testing, archiving, etc. is boring
- Types of clouds: **Focus on public clouds**
  - believe that applicable to any kind of cloud
    - only difference: private clouds have planned downtime
  - cloud on the chip
  - swarms: ad-hoc private clouds
- IaaS vs. PaaS vs. SaaS: **Focus on PaaS**

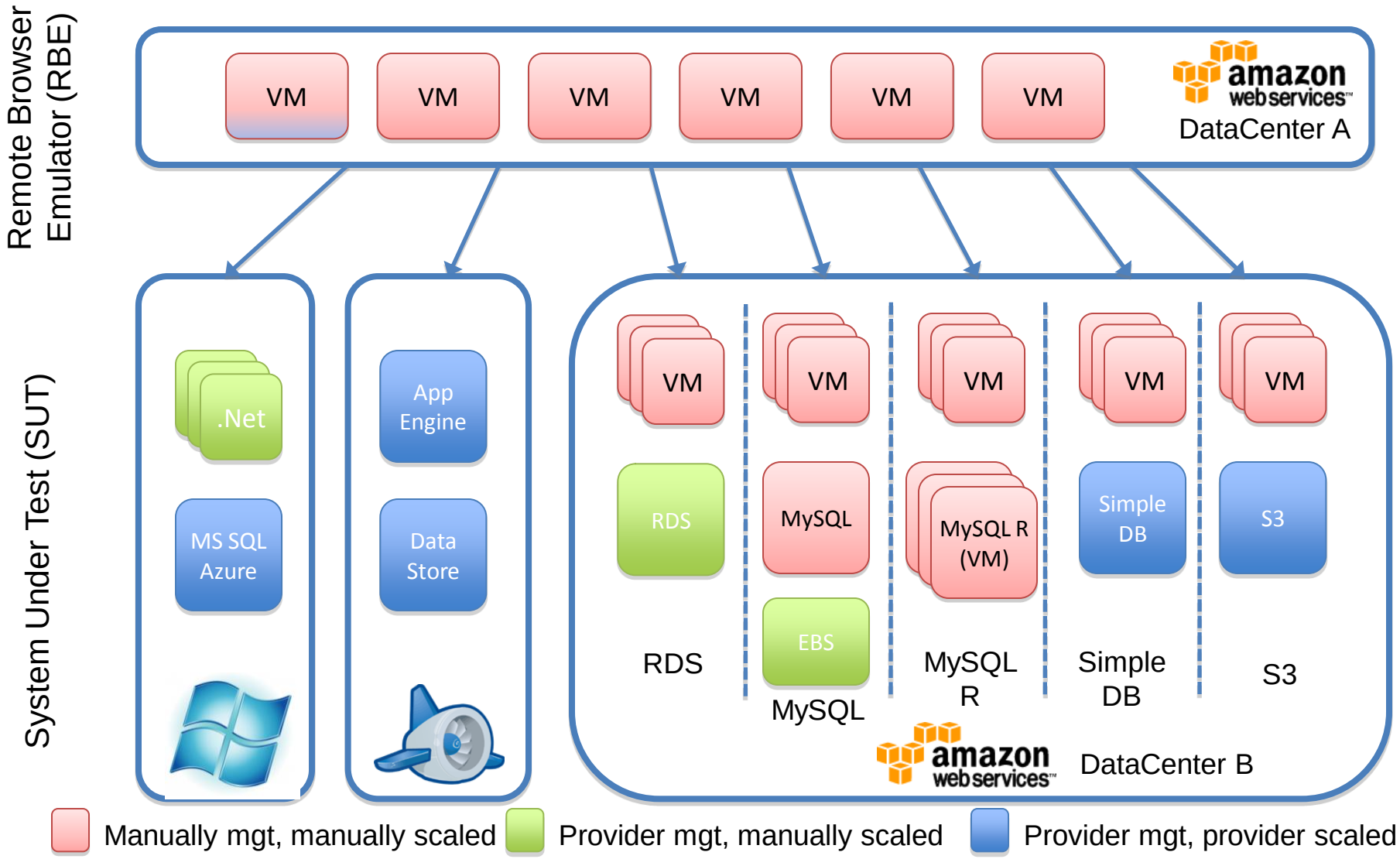
Tested Services

	MS Azure	Google App Eng	AWS RDS	AWS S3
Business Model	PaaS	PaaS	PaaS	IaaS
Architecture	**Replication**	**Part. + Repl. (+Dist. Control)**	**Classic**	**Distr. Control**
Consistency	**SI**	**≈ SI**	**Rep. Read**	**EC**
Cloud Provider	Microsoft	Google	Amazon	Flexible
Web/App Server	.Net Azure	AppEngine	Tomcat	Tomcat
Database	SQL Azure	DataStore	MySQL	--
Storage / FS	Simple DataStore	GFS	--	S3
App-Language	C#	Java/AppEngine	Java	Java
DB-Language	SQL	GQL	SQL	Low-Lev. API
HW Config.	Part. automatic	Automatic	Manual	Manual

Tested Services

	AWS SimpleDB	AWS MySQL	AWS MySQL/R
Business Model	PaaS	IaaS	IaaS
Architecture	**Replication**	**Classic**	**Replication**
Consistency	**EC**	**Rep. Read**	**Rep. Read**
Cloud Provider	Amazon	Flexible	Flexible
Web/App Server	Tomcat	Tomcat	Tomcat
Database	SimpleDB	MySQL	MySQL
Storage / File System	--	EBS	EBS
App-Language	Java	Java	Java
DB-Language	SimpleDB Queries	SQL	SQL
HW Config	Partly Automatic	Manual	Manual

Benchmark Setup



Benchmarking Cloud Services

- Goal: Test if cloud promises are fulfilled
- Benchmarking Approach [DBTest09, SIGMOD10]
  - TPC-W Benchmark, Ordering Mix
  - Adapted for testing elasticity and cost ($)
  - Vary load (EB): clicks per second
- Benchmark metrics
  - Cost: $ / WI
  - Cost Predictability: σ($/WI)
  - Throughput: WIPS

Cost [m$/WI]

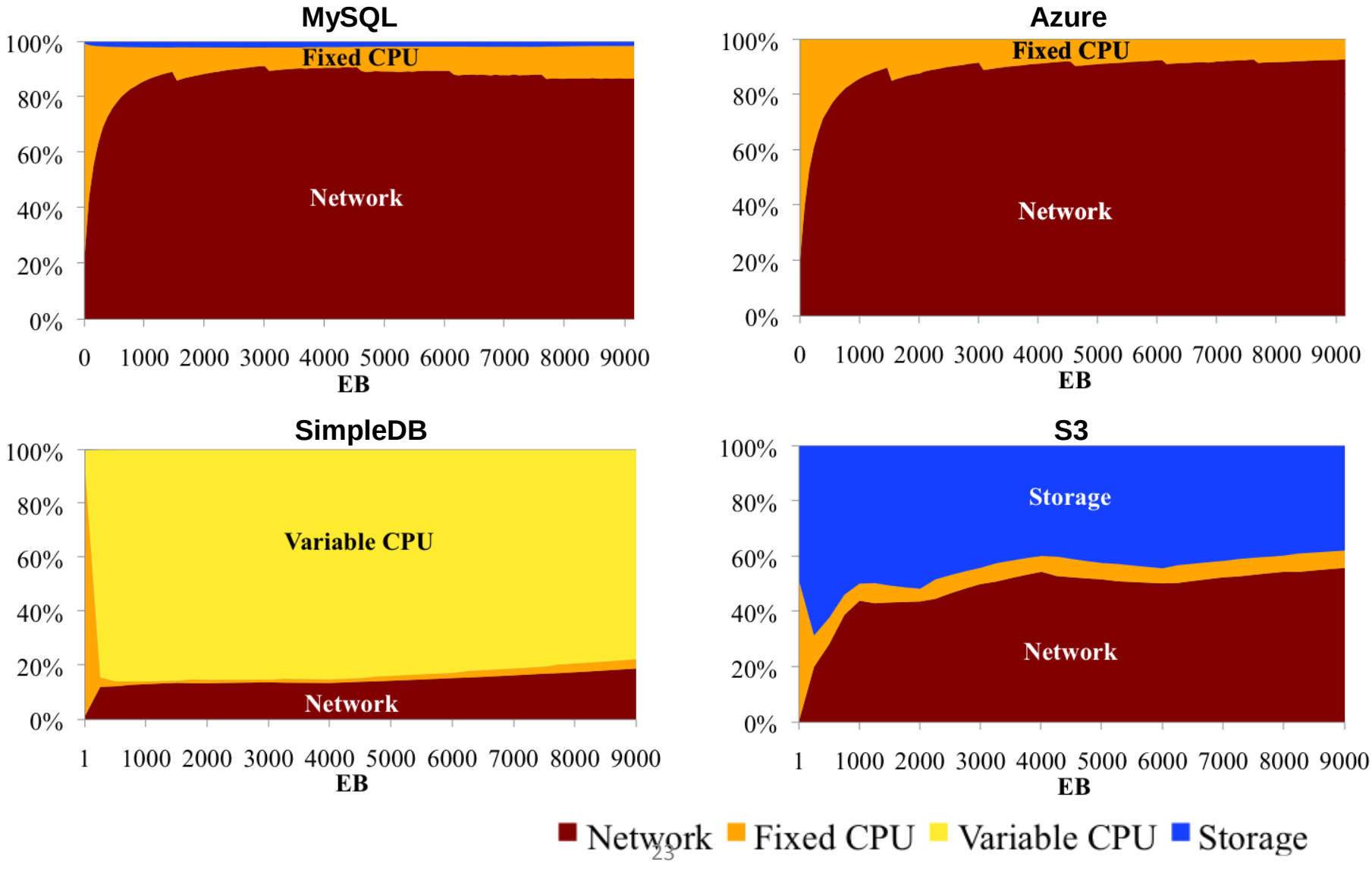
	EBs					Fully Utilized
	1	10	100	500	1000	
MySQL	0.635	0.072	0.020	0.006	**0.006**	0.005
MySQL/R	2.334	0.238	0.034	0.008	**0.006**	**0.005**
RDS	1.211	0.126	0.032	0.008	**0.006**	0.005
SimpleDB	0.384	0.073	0.042	0.039	0.037	0.037
S3	1.304	0.206	0.042	0.019	0.011	0.009
Google AE	**0.002**	0.028	0.033	0.042	0.176	0.042
Google AE/C	**0.002**	0.018	0.026	0.028	0.134	0.028
Azure	0.775	0.084	0.023	0.006	**0.006**	0.005

Cost predictability [m$/WI]

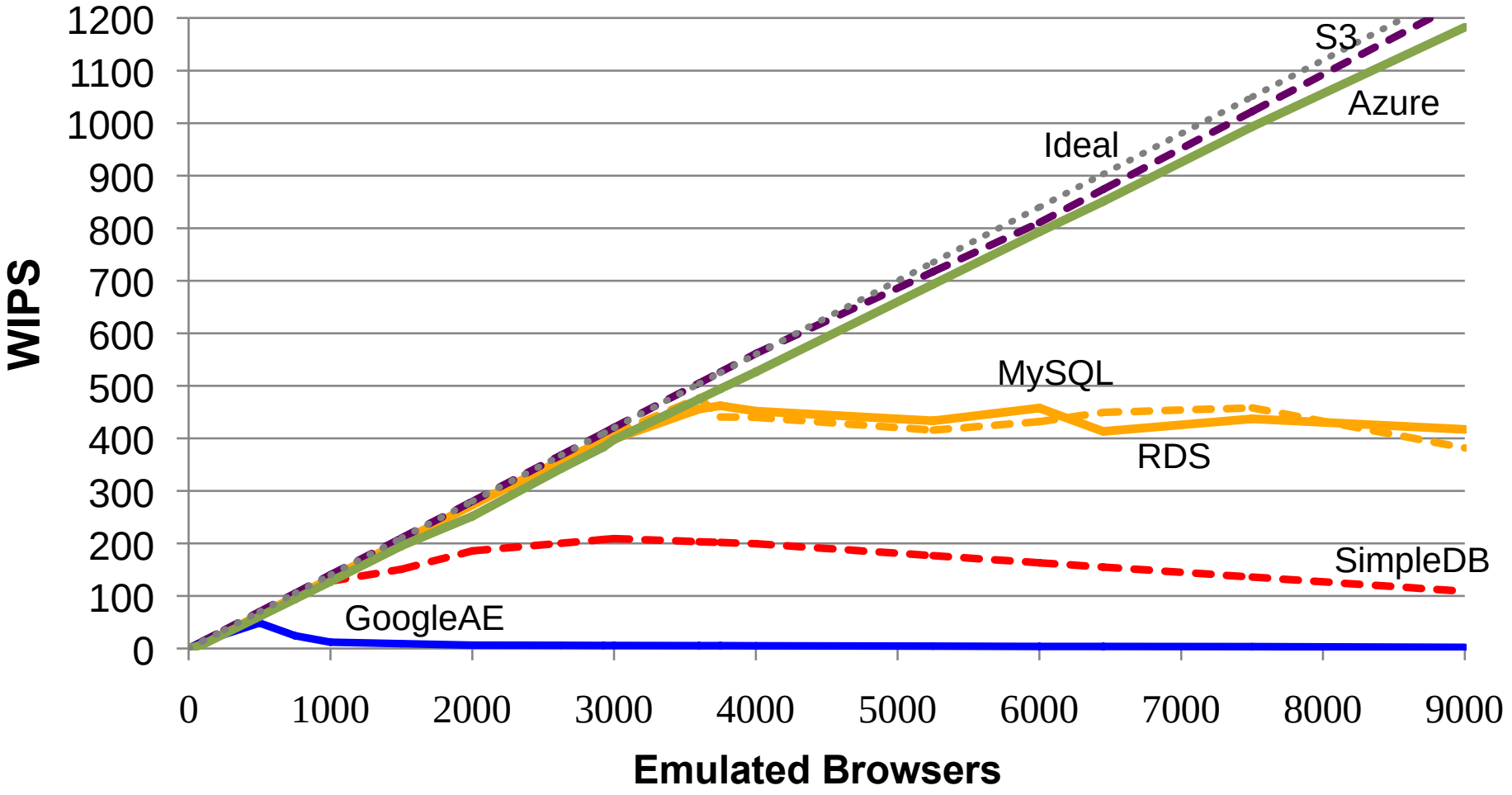
	mean ± σ
MySQL	0.015 ± 0.077
MySQL/R	0.043 ± 0.284
RDS	0.030 ± 0.154
SimpleDB	0.063 ± 0.089
S3	0.018 ± 0.098
Google AE	0.029 ± 0.016
Google AE/C	0.021 **± 0.011**
MS Azure	0.010 ± 0.058

- Ideal: Cost / WI is constant
  - low σ better (ideal: σ=0)
  - cost independent of load
- Google clear winner here!

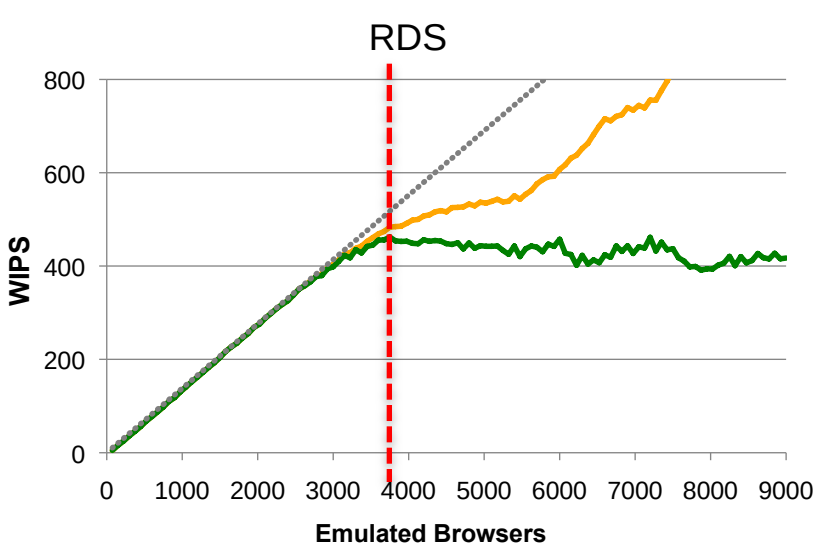
Relative Cost Factors

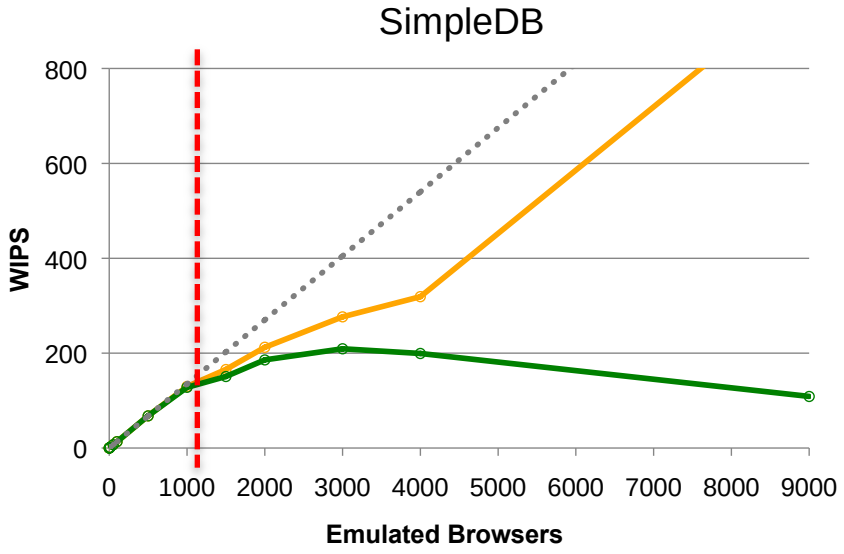


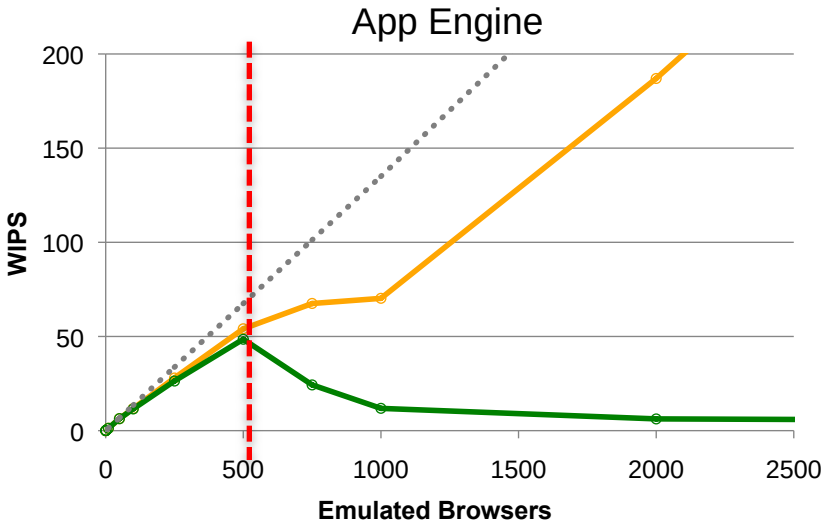
Throughput [WIPS]



Overload Behavior







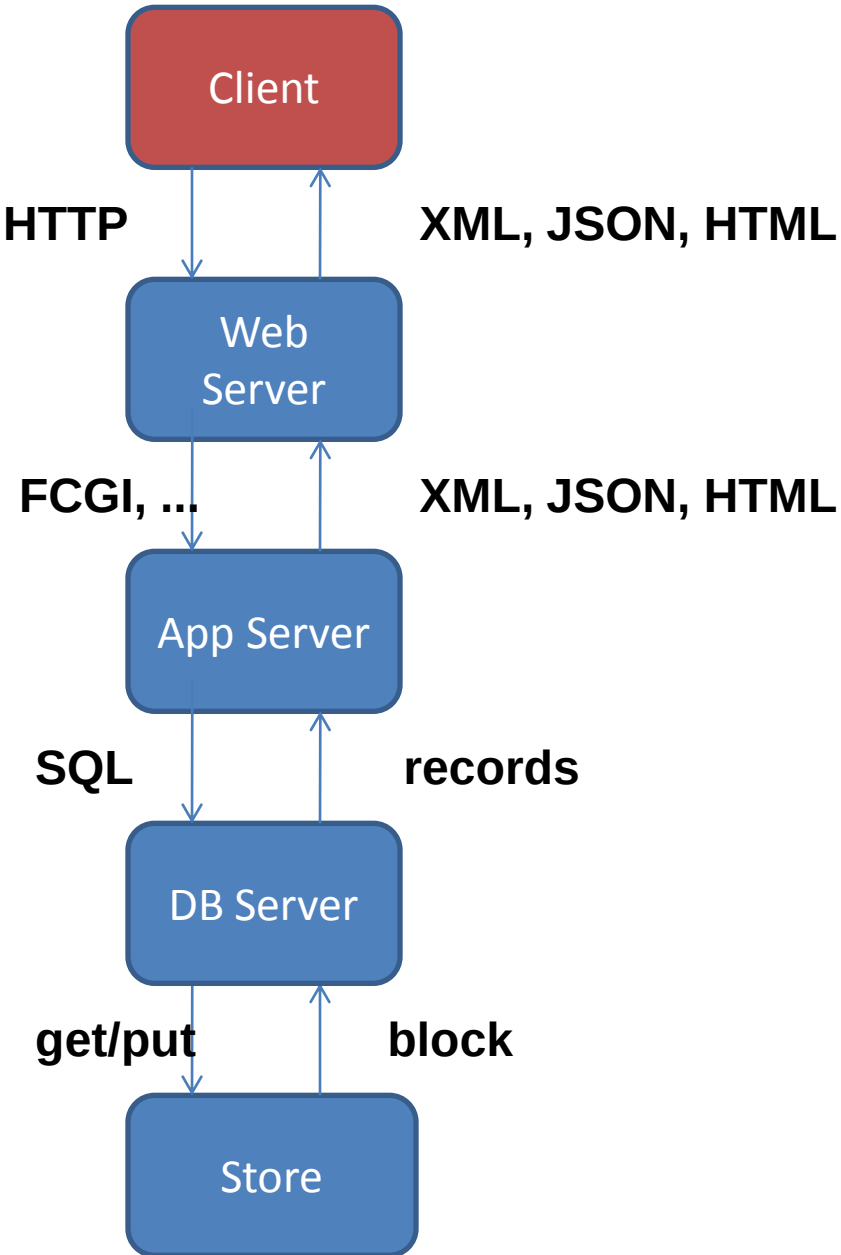
Agenda

- Promises of Cloud Computing
- Benchmarking the State-of-the Art [SIGMOD 2010]
  - Amazon, Google, Microsoft
- **Towards a Silver Bullet [ICDE 2010]**
  - **Asking the right questions**

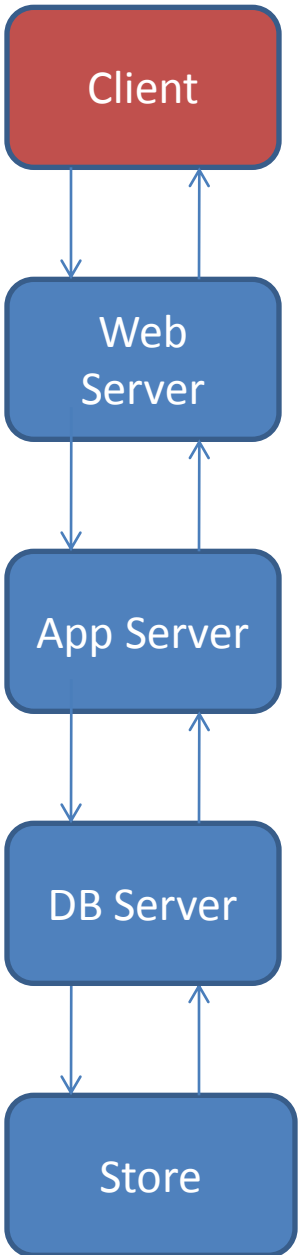
Principles of Distributed Databases

- Partitioning [Ceri,Pelagatti 1984]
  - Large number of possible partitioning schemes
  - Repartitioning is very expensive
- Replication, Caching [Bernstein et al. 1987]
  - Replicating data increases fault-tolerance and read performance
  - Replication needs a mechanism to keep replicas consistent
- Distributed Control [Tanenbaum 2002]
  - Client-side consistency protocols
- Combination of approaches difficult! [Unterbrunner+ 2010]

Reference Architecture

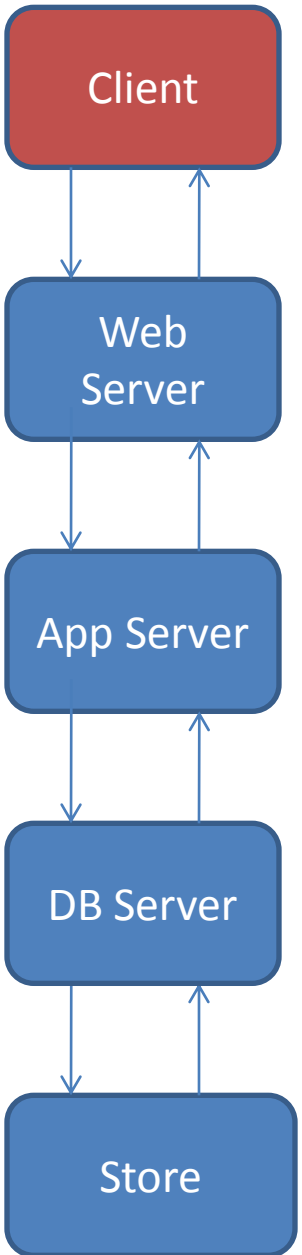


Open Questions



- How to map stack to IaaS?
- How to implement store layer?
- What consistency model?
- What programming model?
- Whether and how to cache?

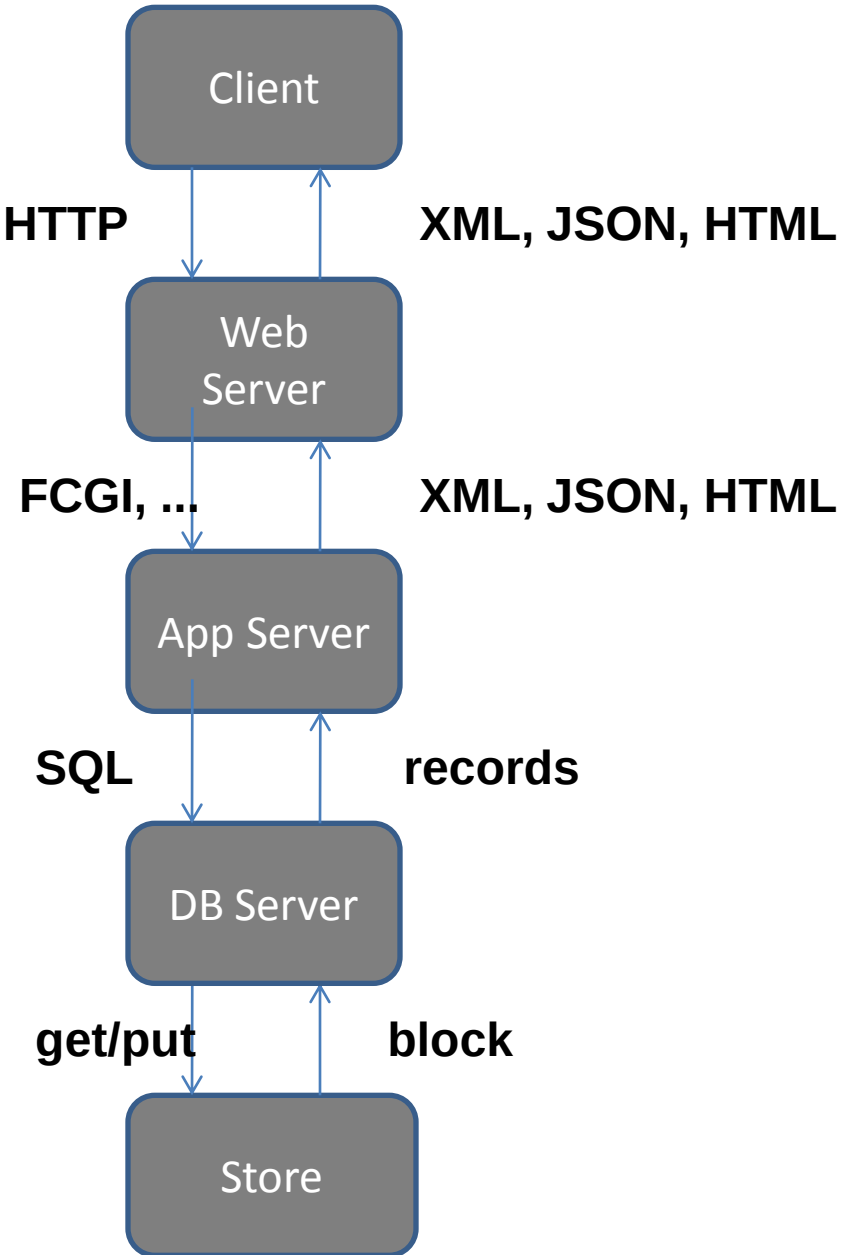
Open Questions

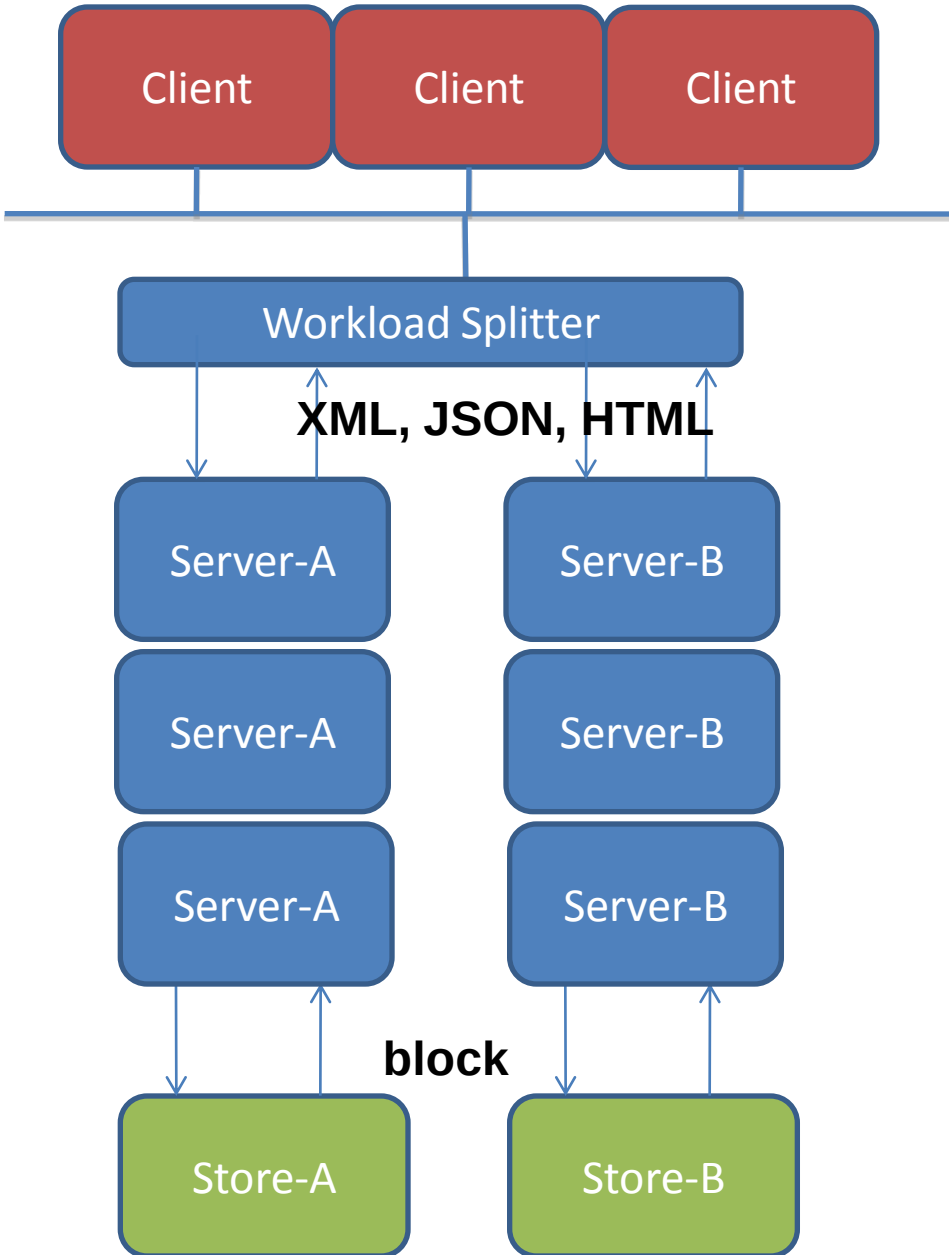


- **How to map stack to IaaS?**

- How to implement store layer?
  - [VLDB 2009a]
- What consistency model?
  - [VLDB 2009b]
- What programming model?
  - [VLDB 2009c]
- Whether and how to cache?
  - [SIGMOD 2010]

Variant I: Partition Workload by „Tenant“

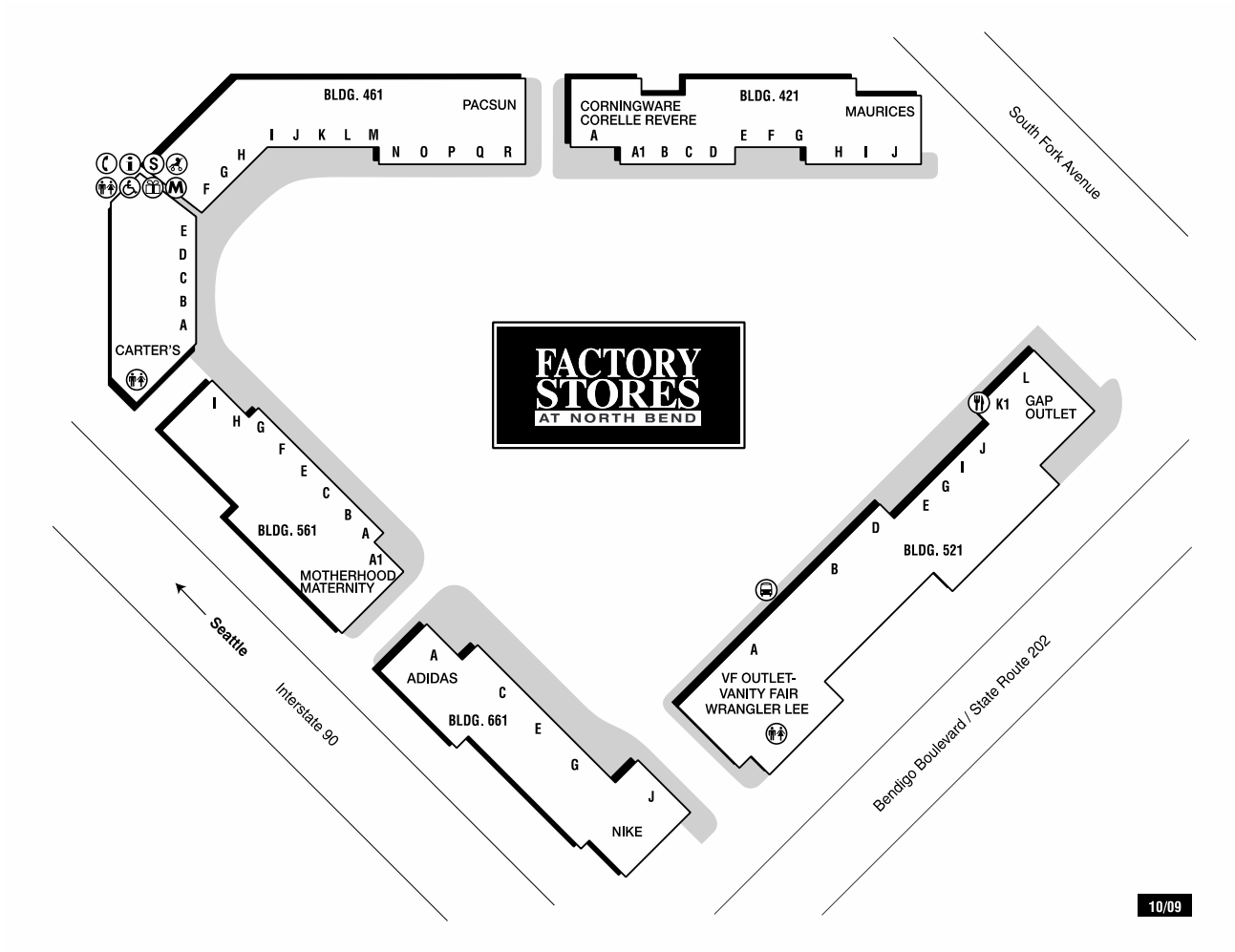




Partition Workload by „Tenant“

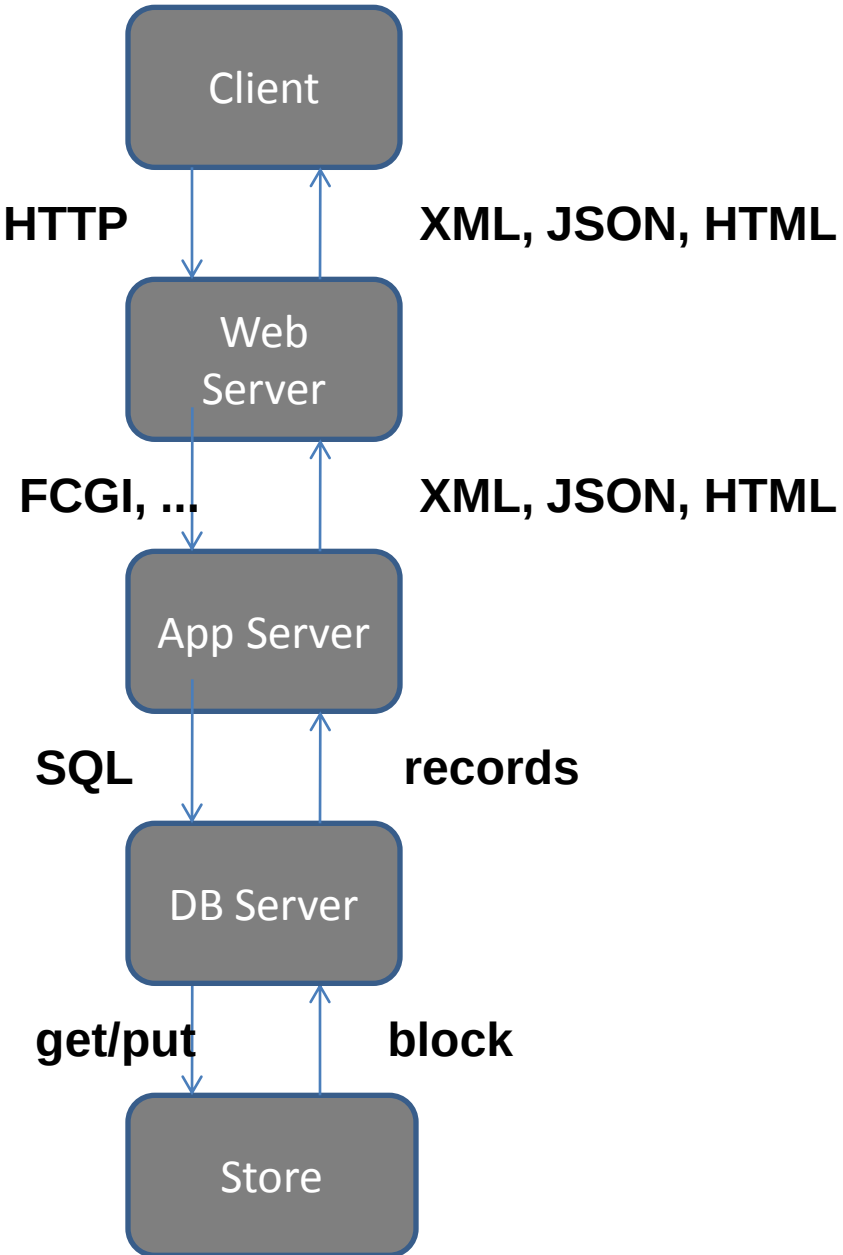
- Principle
  - partition data by „tenant“
  - route request to DB of that tenant
- Advantages
  - reuse existing database stack (RDBMS)
  - flexibility to use DAS or SAN/NAS
- Disadvantages
  - multi-tenant problem *[Salesforce]*
    - optimization, migration, load balancing, fix cost
  - silos: need DB federator for inter-tenant requests
  - expensive HW and SW for high availabilty

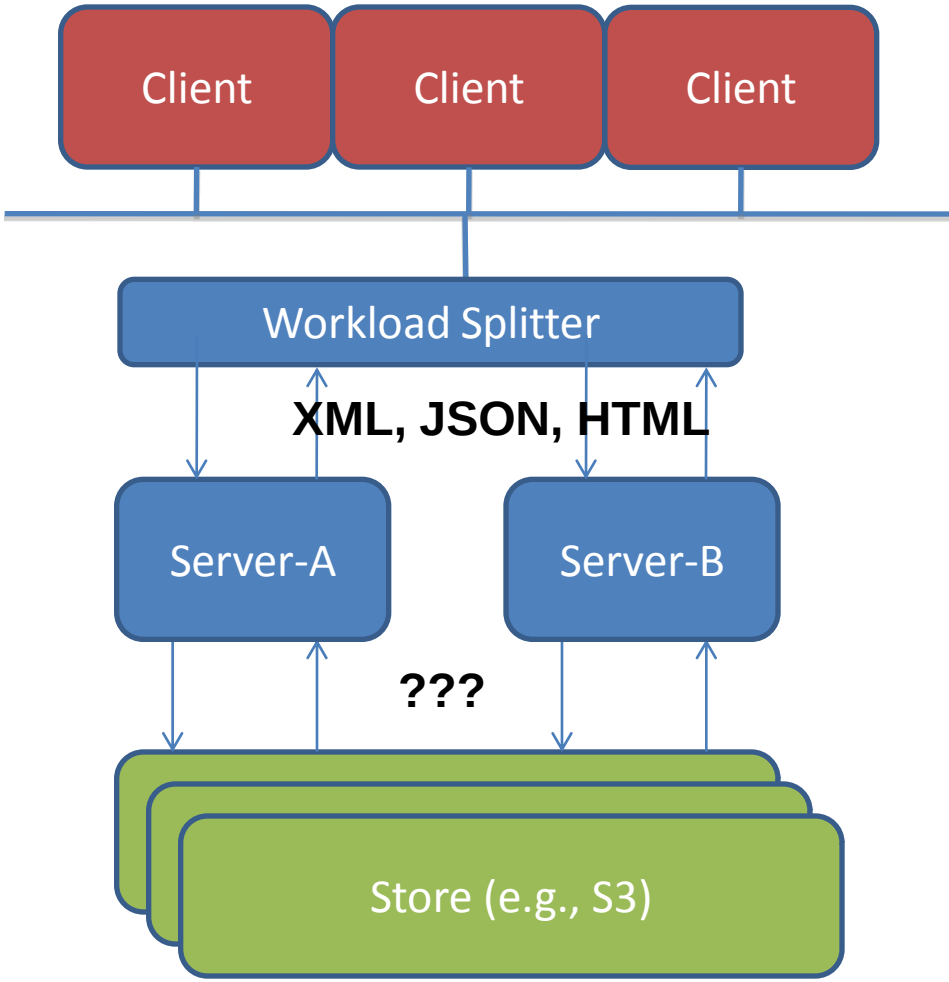
Metaphor: Shopping Mall



- If a shop is successful, you need to move it!
  - (popularity vs. growth of product assortment)

Variant II: Partition Workload by „Request“





Metaphor: Internet Department Store

- If a product is successful, you stock up its supply
  - Transparent and fine-grained reprovisioning
  - Cost of reprovisioning much lower!!!

Partition Workload by „Request“

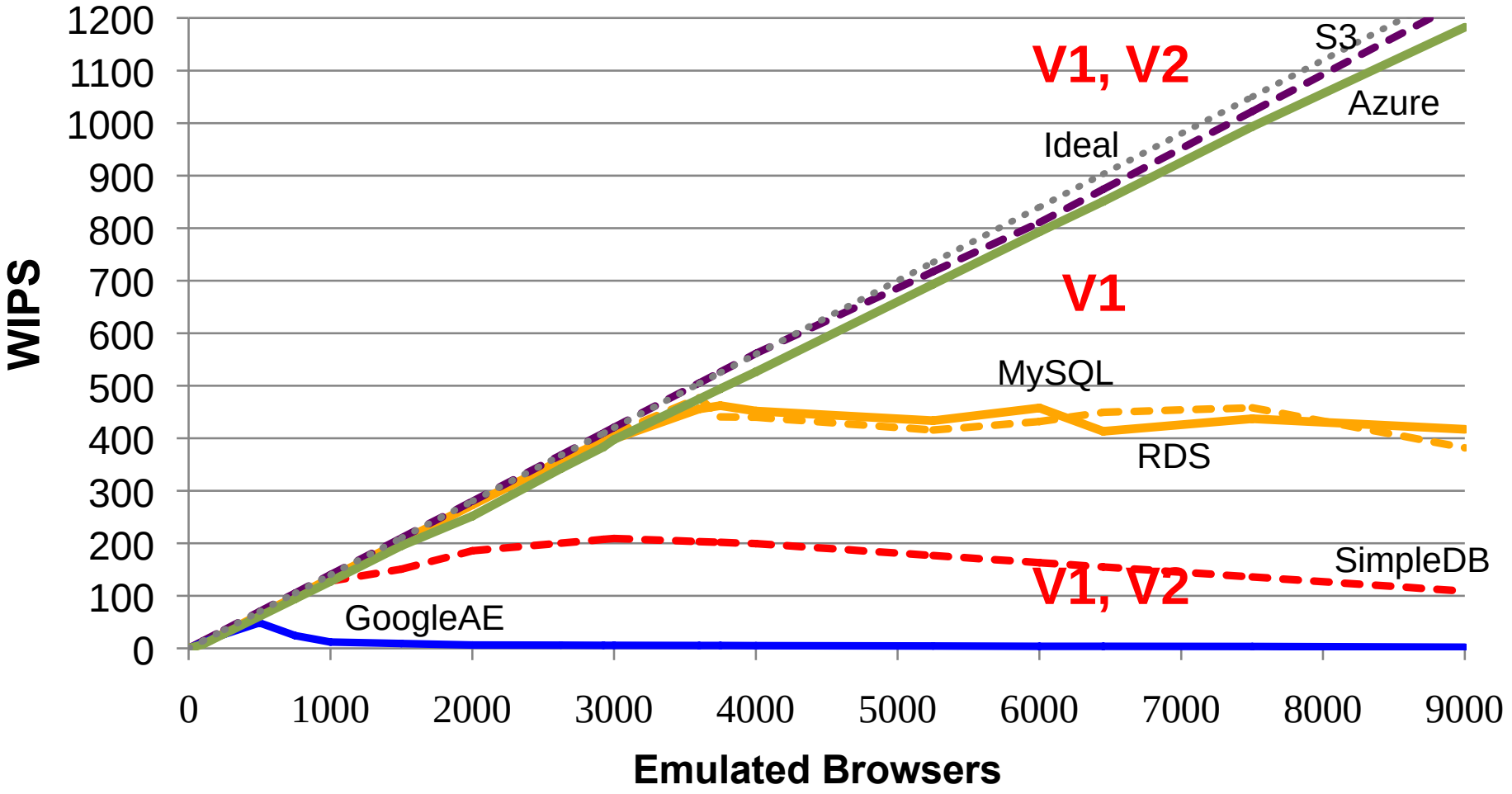
- Principle
  - fine-grained data partitioning by page or object
  - any server can handle any request
  - implement DBMS as a library (not server)
- Advantages
  - avoids disadvantages of Variant I
- Disadvantages
  - new synchronization problem
  - whole new breed of systems
  - caching not effective

Experiments Revisited: Cost / WI (m$)

	Low Load	Peak Load
Variant 1 (RDS)	1.212	0.005
Variant 2 (S3)	-	0.007
Variant 2 (Google)	0.002	0.028
Variant 1 (Azure)	0.775	0.005

Only V2 is cheap and predictable!

Throughput [WIPS]



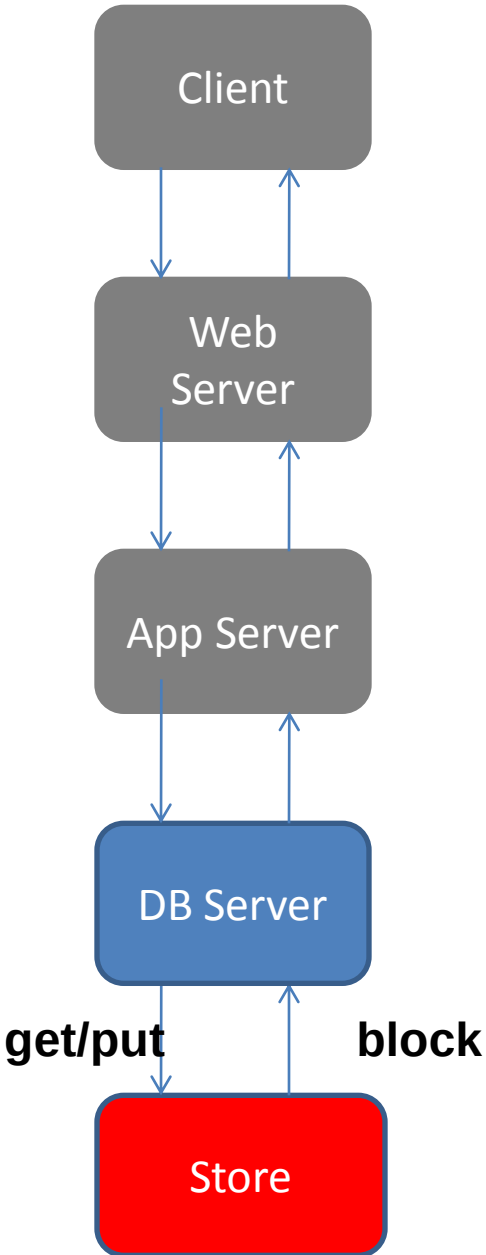
High performance can be achieved with both!

Warning: Still a hypothesis. S3 is only EC.

Open Questions

- How to map traditional DB stack to IaaS?
- **How to implement the storage layer?**
- What is the right consistency model?
- What is the right programming model?
- Whether and how to make use of caching?

Storage Layer Questions



- What is the interface?
  - get/put or more?
  - return records or blocks?
- How to implement this interface?
  - data structures
  - storage media and network
  - clustering of data
  - granularity of partitioning / sharding

API of Store

- Assumption: Hardware trends will change the game
  - most data in MM or SSD / PCM
- Proposition 1: return tuples rather than blocks
  - blocks are an arte-fact of old HW (i.e., disks)
- Proposition 2: push down predicates
  - because it can be done (power depends on impl.)
  - compensates for cost of remote storage
  - (old idea; e.g., iDisks)
- Open: consistency constract of store?
  - store might have nice properties (e.g., mon. Writes)
  - store might have nice primitives (e.g., counter, set/test)

Store Implementation Variants

- DAS
  - local disks with physically exclusive access
  - put/get interface; no synchronization
  - only works for V1
- Key-value stores (e.g., Dynamo)
  - DHTs with concurrent access
  - put/get interface; no synchronization
  - works for V1 and V2; makes more sense for V2
- A DBMS (e.g., MySQL, SQL Server)
  - have been misused before
- ClockScan *[Unterbrunner et al. 2009]*
  - massively shared scans in a distributed system

Open Questions

- How to map traditional DB stack to IaaS?
- How to implement the storage layer?
- **What is the right consistency model?**
- What is the right programming model?
- Whether and how to make use of caching?

CAP Theorem

- Three properties of distributed systems
  - Consistency (ACID transactions w. serializability)
  - Availability (nobody is ever blocked)
  - resilience to network Partitioning
- Result
  - it is trivial to achieve 2 out of 3
  - it is impossible to have all three
- Highly controversial discussion [Brantner 08, Abadi 10]
  - Levels of availability (always vs. never available)
  - What is the difference between CP and CA

What have people done?

- Client-side Consistency Models *[Tannenbaum]*
  - read & write monotonicity, session consistency, ...
- Time-line consistency *[PNUTS08]*
  - all writes are consistent (except across DC)
  - read consistency: either don ‘t care or latest version
- New DB transaction models
  - Escrow, Reservation Pattern *[O ‘Neil 86], [Gawlick 09]*
  - SAGAs and compensation; e.g., in BPEL *[G.-Molina,Salem]*
  - SAP, Amadeus et al. *[Buck-Emden], [Kemper et al. 98]*
- Define logical unit of consistency *[Azure]*
  - strong consistency within LuC
  - app-defined consistency across LuCs
- Educate Application Developers *[Helland 2009]*

What have people done?

- Client-side Consistency Models *[Tannenbaum]*
  - read & write monotonicity, session consistency, ...
- Time-line consistency *[PNUTS08]*
- [illegible]
- [illegible]
  - strong consistency within LuC
  - app-defined consistency across LuCs
- Educate Application Developers *[Helland 2009]*

Sacrifice Consistency
Push problem to app programmer!

What have we done?

- Levels of Consistency, Cost Tradeoffs *[Brantner08]*
  - can achieve read/write monotonicy + „A“ + „P“
  - the more consistency, the higher the cost

- Economic models for consistency *[Amadeus], [Kraska09]*
  - Classify the data as A, B, C
    - A data always strong consistent
    - C data always eventually consistent
    - B data handled like A or C data adaptively
  - Cost model that estimates business impact of inconst.

What have we done?

- Levels of Consistency, Cost Tradeoffs *[Brantner08]*
  - can achieve read/write monotonicy + „A“ + „P“
  - –
- E*[…aska09]*
  - –
    - B data handled like A or C data adaptively
  - – Cost model that estimates business impact of inconst.

Follow tradition to
push problem to app programmer!

But for a different reason!!!

Cost per 1000 TAs [$] (TPC-W)

	Total	Adjustable	Fixed
Nothing	**0.15**	0	0.15
Durability	**1.8**	1.1	0.7
Monotonicity	**2.1**	1.4	0.7
Monotonicity+ Atomicity	**2.9**	2.6	0.3

[Brantner+08]

Latency per TAs [secs] (TPC-W)

	Avg.	Max.
Nothing	11.3	12.1
Durability	4.0	5.9
Monotonicity	4.0	6.8
Mono+Atomicity	2.8	4.6

[Brantner+08]

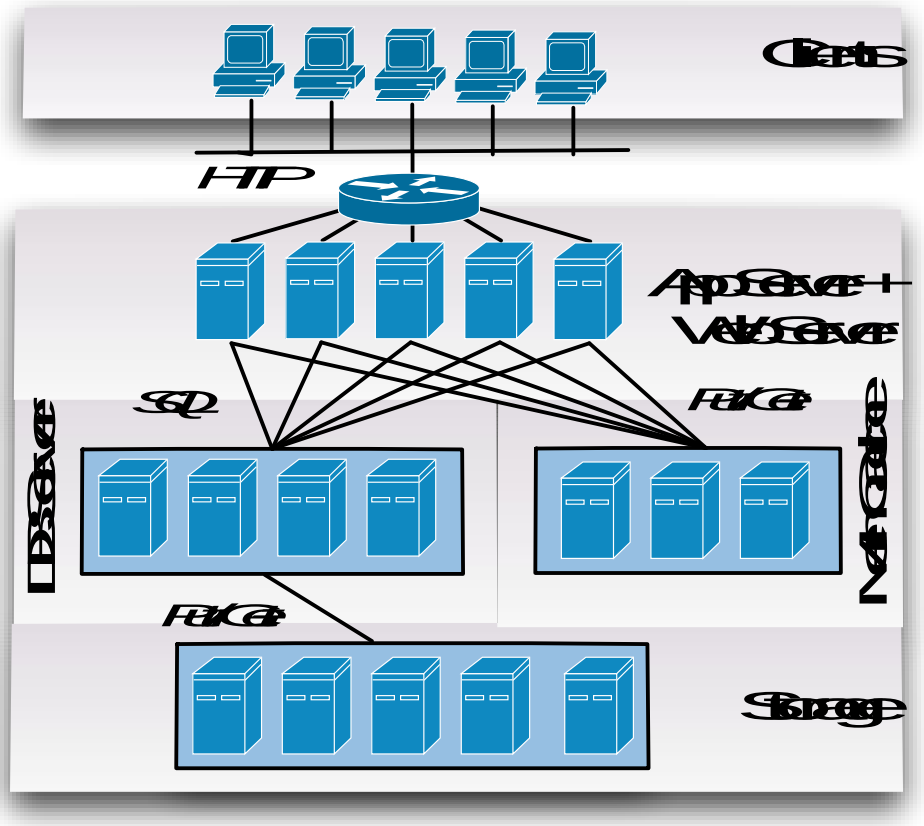
Open Questions

- How to map traditional DB stack to IaaS?
- How to implement the storage layer?
- What is the right consistency model?
- What is the right programming model?
- Whether and how to make use of caching?

Programming Model

- Properties of a programming lang. for the cloud
  - support DB-style + OO-style + CEP-style
  - avoid keeping state at app servers for V2
- Many languages will work in the cloud
  - SQL, XQuery, Ruby, .Net /LINQ, ...;
  - J2EE will not work
- Open (research) questions
  - do OLAP on the OLTP data: My guess is yes!
  - rewrite your apps: My guess is yes!

Caching



- Many Variants Possible
  - this is just one
  - V1 caching mandatory
  - V2 caching prohibitive
- TPC-W Experiments
  - marginal improvements for Google AppEngine
- No low hanging fruit

Agenda

- Promises of Cloud Computing
- Benchmarking the State-of-the Art [SIGMOD 2010]
  - Amazon, Google, Microsoft
- Towards a Silver Bullet [ICDE 2010]
  - Asking the right questions

Case Study: Bets Made by 28msec

- How to map traditional DB stack to IaaS?
  - implemented both architectures (V1 + V2)
  - V1 only in a single server variant for low end
- How to implement the storage layer?
  - EBS for V1; KVS for V2
- What is the right consistency model?
  - ACID for V1; configurable for V2
- What is the right data + programming model?
  - XML & XQuery
- Whether and how to make use of caching?
  - No! (Only for code / precompiled query plans)

Conclusion & Future Work

- Hopefully started a new benchmark war
  - Tested elasticity, cost and cost predictability
  - Vendors interested in cost experiments
  - Academia interested in scalability experiments
  - **More bits available: http://www.pubzone.org**

- Find silver bullet for data management in cloud
  - Develop a reference architecture
  - Combine partitioning + replication + distributed control
  - ETH Systems Group: **http://www.systems.ethz.ch**