Provenance for the Cloud

Kiran-Kumar Muniswamy-Reddy,

Peter Macko, and Margo Seltzer

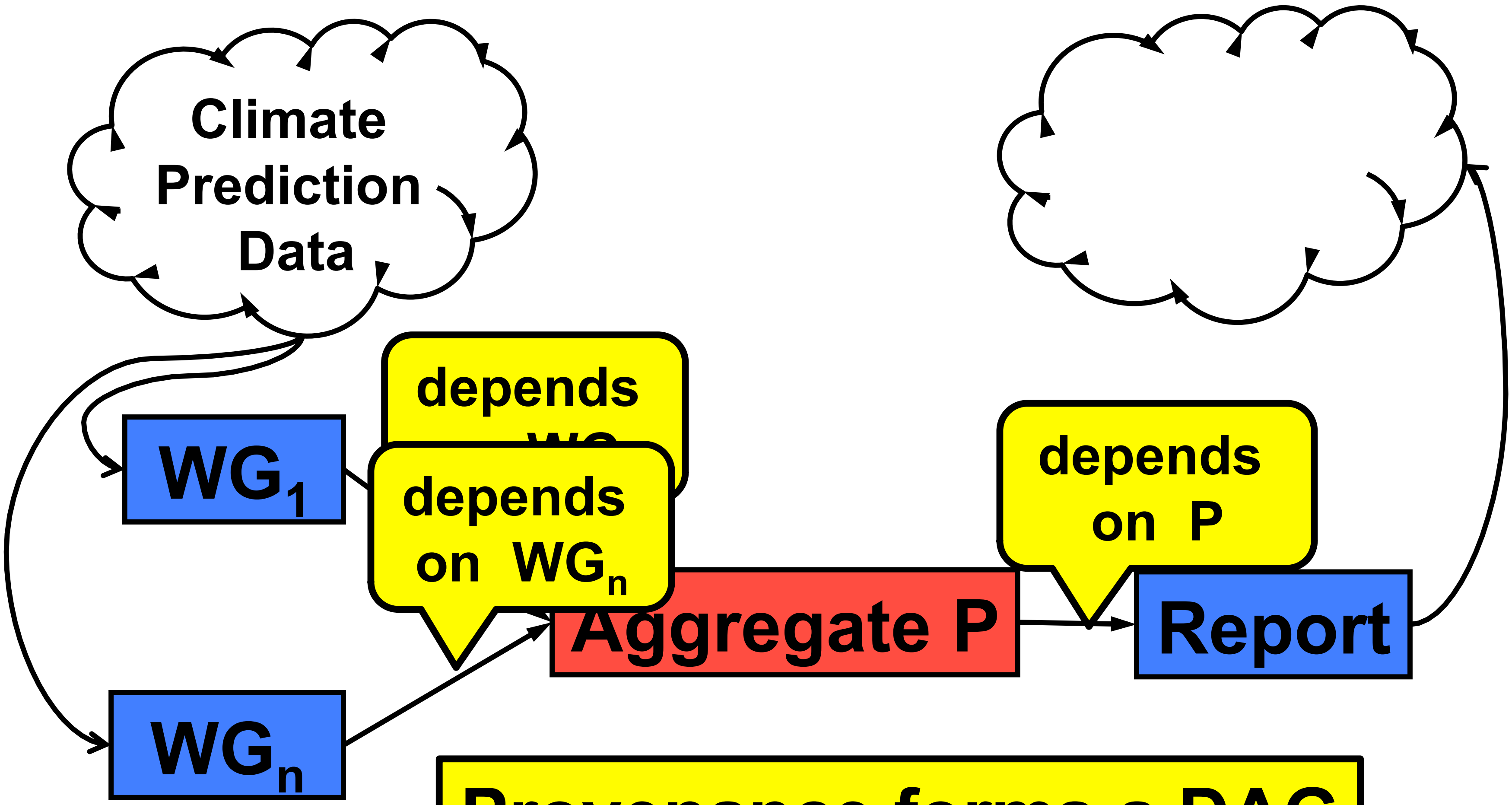
Cloud Stores

- Becoming increasingly important
  - Backups
  - Host shared scientific data
  - Store web application data
  - Serve web pages
- However, not designed to store provenance

What is Provenance?

- Meta-data describing the history of an object
  - What objects does this object depend on?
  - What applications modified/generated this object?

What is Provenance? (2)



Provenance forms a DAG

Why cloud provenance?

- Provides information regarding structure of data and applications
  - Validate Data sets
  - Identify how data spread through the system
  - Search [Shah-Usenix’07]
  - Generate data on-demand [Adams-HotCloud’09]

Goal

- Provenance is vital
- How do we store provenance given today’s cloud offerings?

Outline

- Introduction
- **Design Issues**
- Protocols
- Evaluation
- Conclusions

Setting

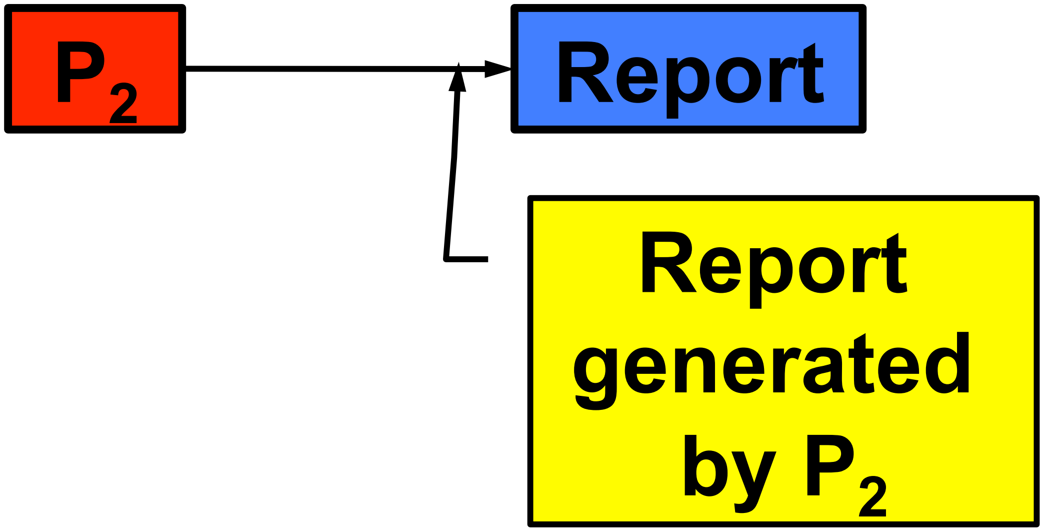
- Provenance-Aware Storage system (PASS) tracks and collects provenance
  - Observes system calls that applications make and infers relationships between objects
  - Designed for local file system/NFS backend
- Modified it to use AWS services as backend

Properties

- Provenance-data coupling
- Multi-object ordering
- Data-independent persistence
- Efficient query

Provenance-data coupling

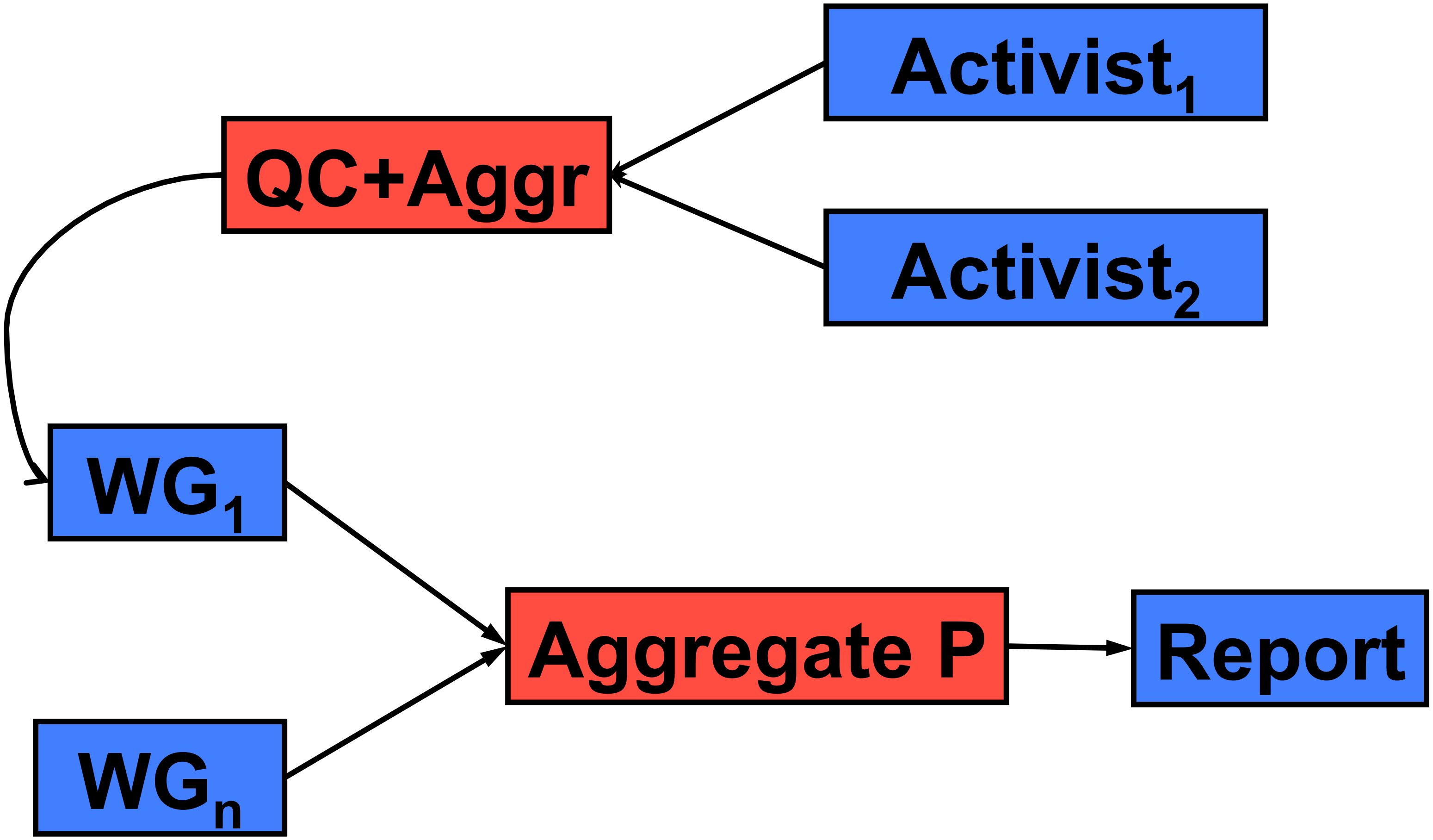
- Provenance accurately describes the data object
- Data must be what is described by provenance
  - Can mislead users if violated
  - Detection, if not coupling



Multi-object Ordering

- The provenance and data of an ancestor object must be recorded in the provenance system
  - No dangling references

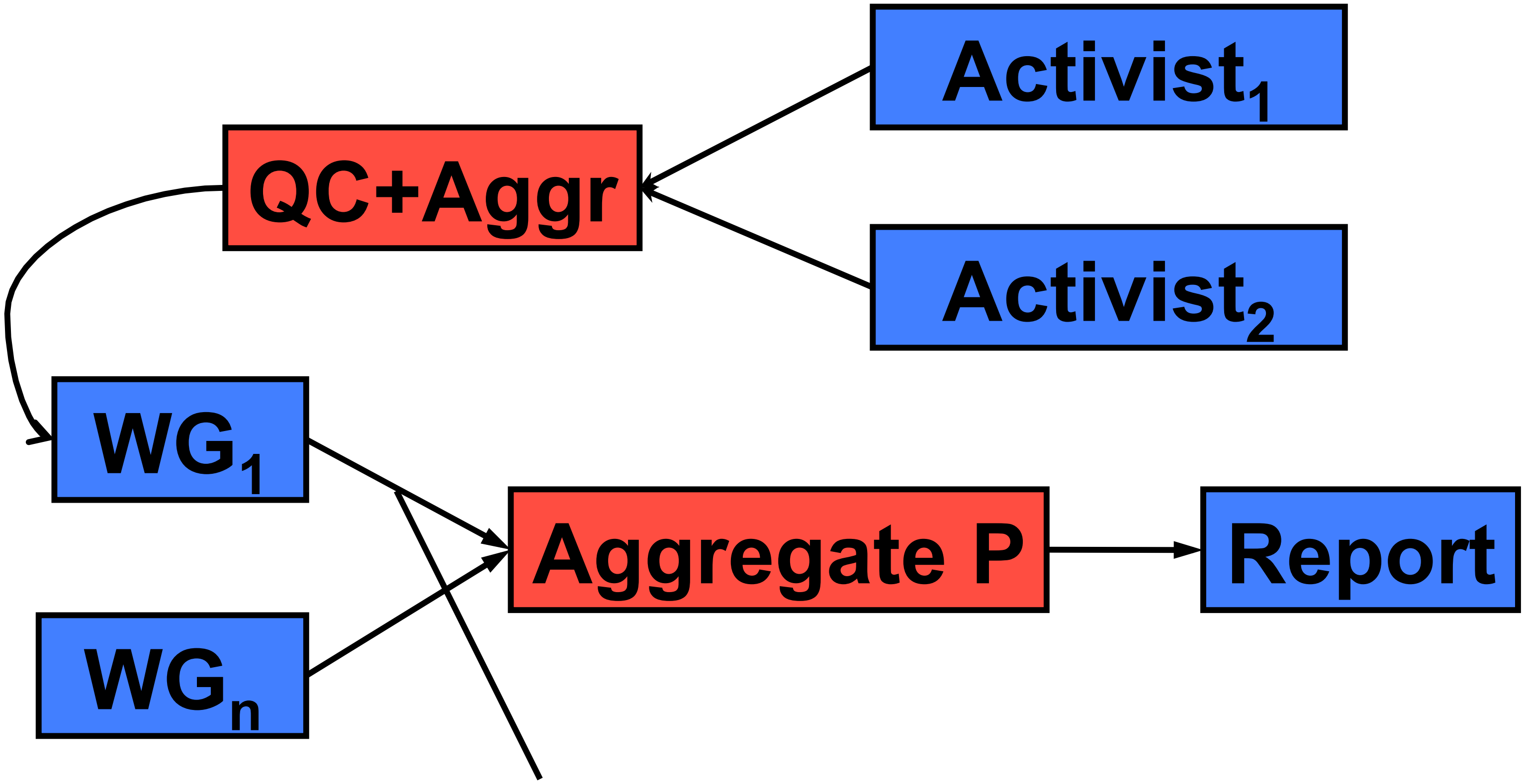
Multi-object Ordering(2)



Data Independent Persistence

- Cannot always delete provenance when object is deleted
  - Can disconnect the provenance DAG

Data Independent Persistence (2)

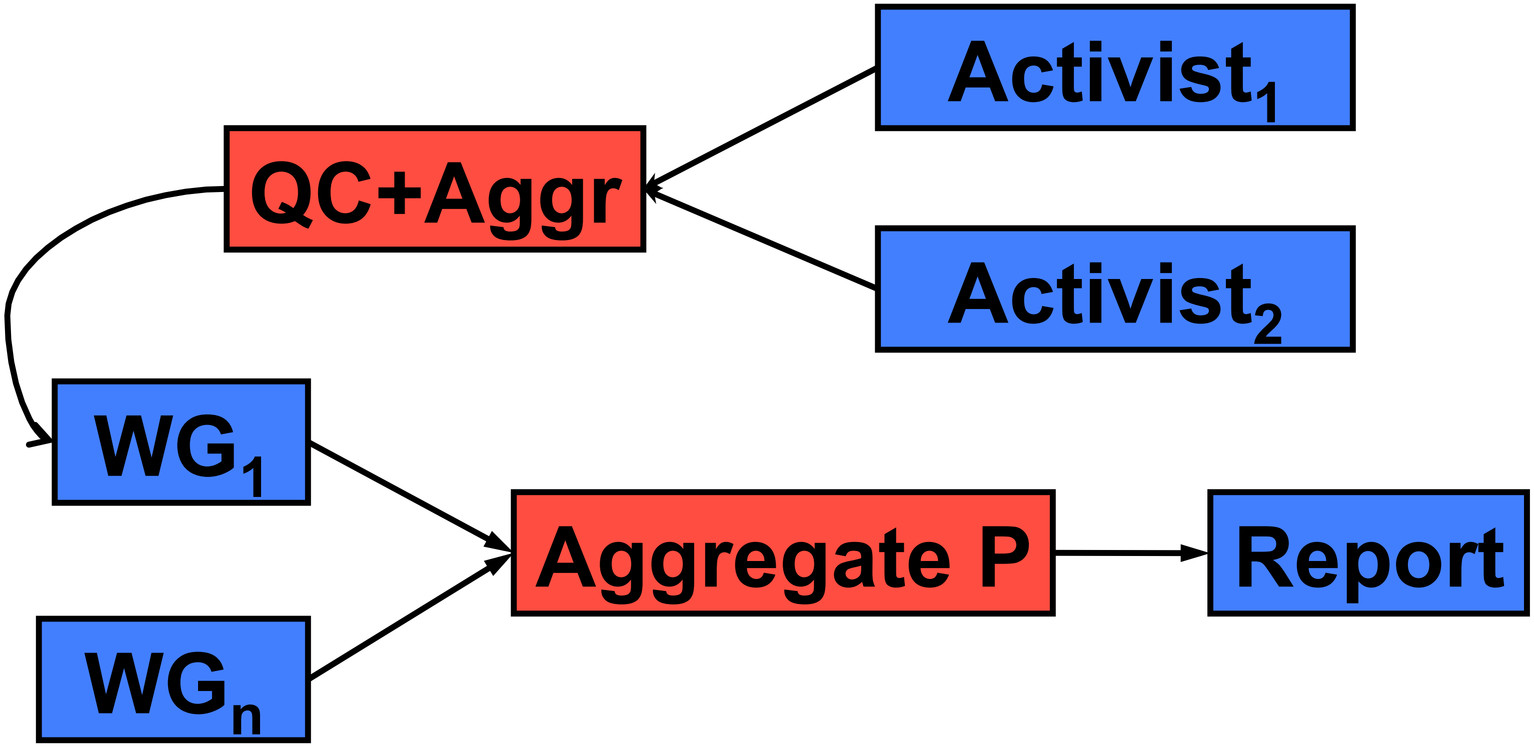


Deleting WG$_1$'s provenance with WG$_1$ will disconnect DAG

Efficient Query

- Provenance must be accessible to users who want to verify properties of their data or simply be aware of its lineage
  - If provenance is not readily accessible, the provenance is of questionable value.

Efficient Query (2)



Query: find all descendants of Activist$_1$

Design Decisions

- Protocols, not system
- Use CloudDB
- Limited guarantees

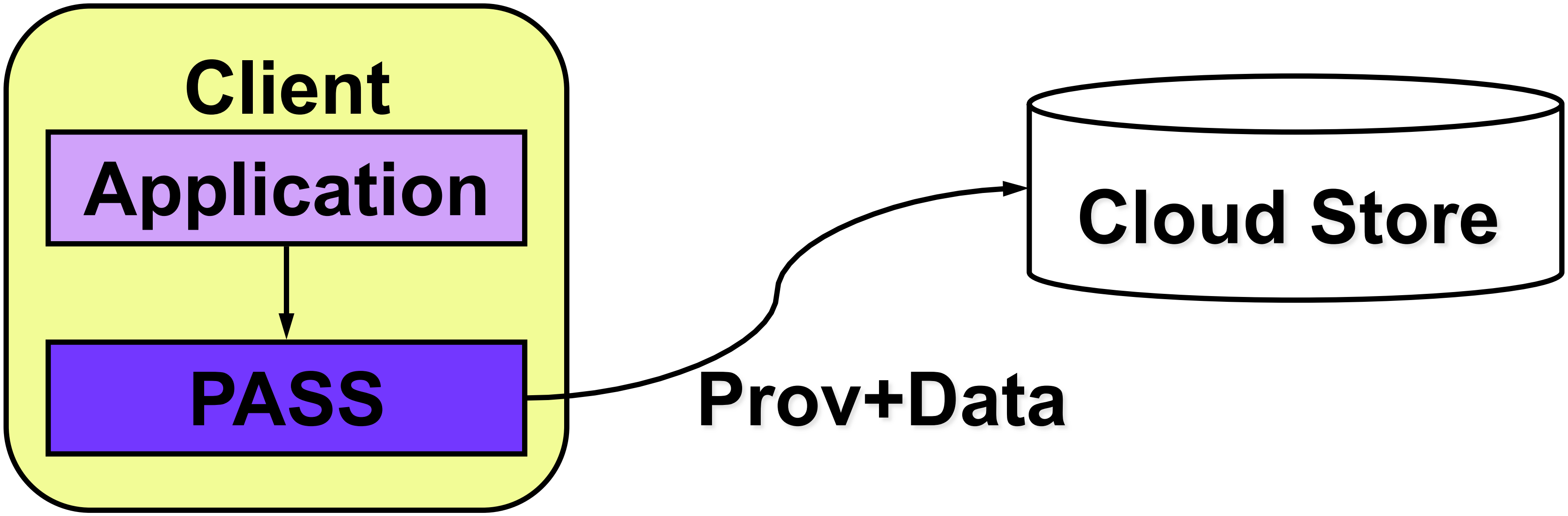
Outline

- Introduction
- Design Issues
- **Protocols**
- Evaluation
- Conclusions

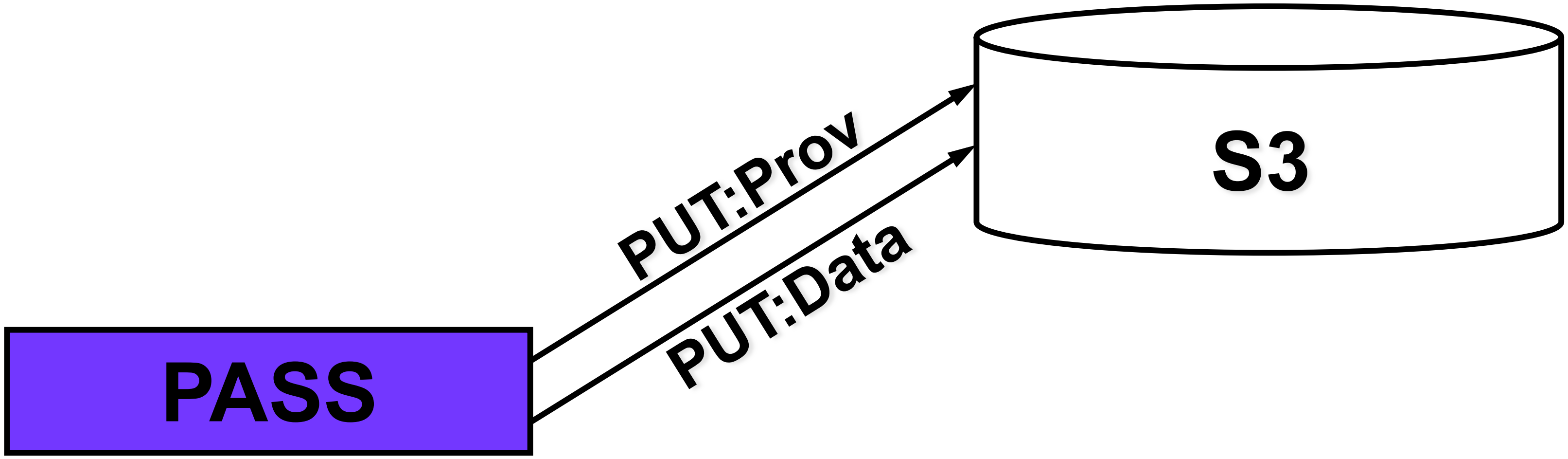
P1: Standalone Cloud Store

- Stores both provenance and data on cloud object store
  - Provenance as a separate object
- Amazon S3 and Azure Blob
  - Object identified by URI
  - SOAP or REST interface
  - Operations: PUT, GET, COPY, DELETE
  - Cost: data storage + bandwidth + num ops
  - S3 - Eventual consistency

P1: Standalone Cloud Store



P1: Standalone S3



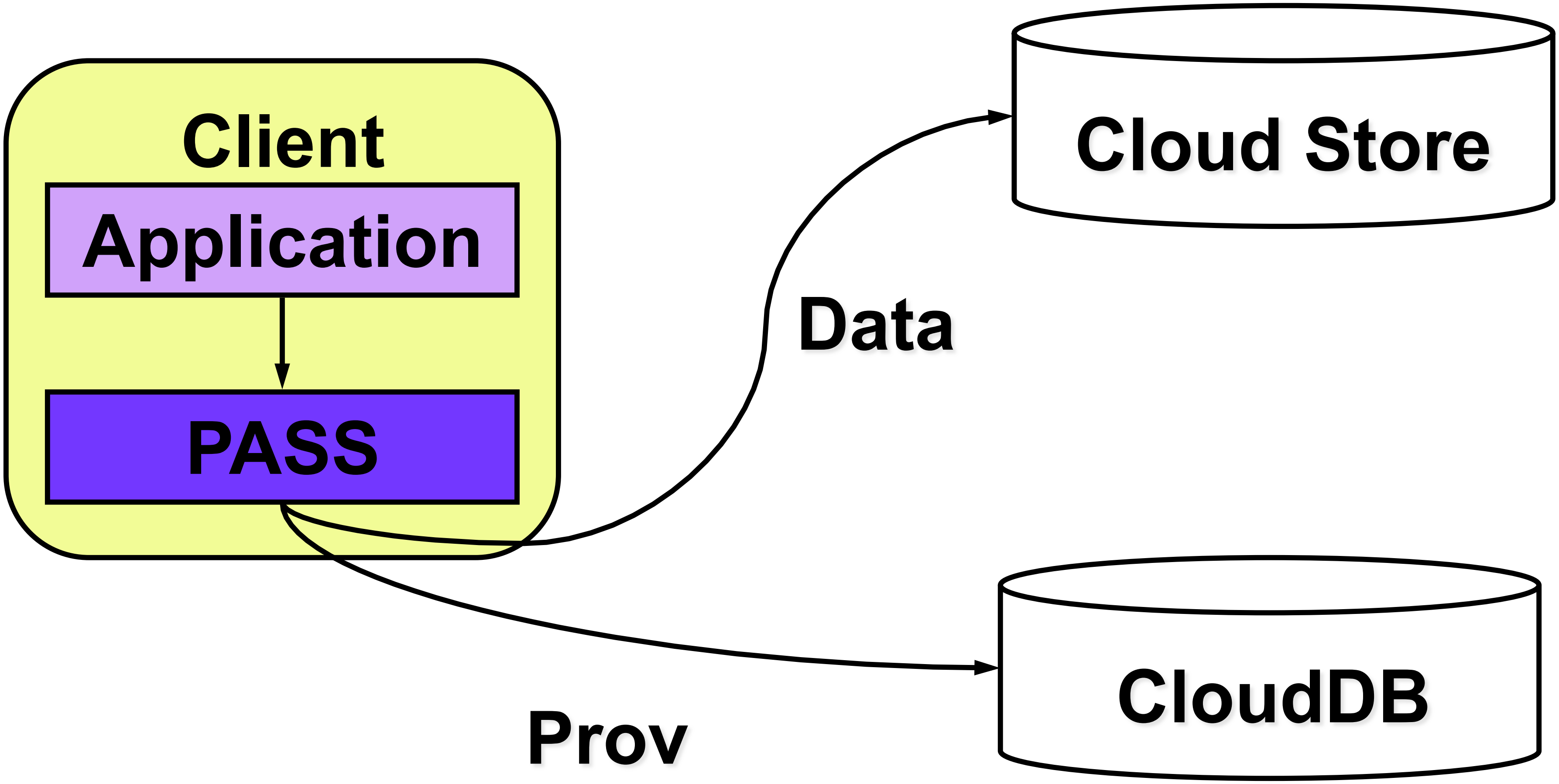
Properties

	Data Coupling	Causal Ordering	Persistence	Efficient Query
P1	✘	✓	✓	✘

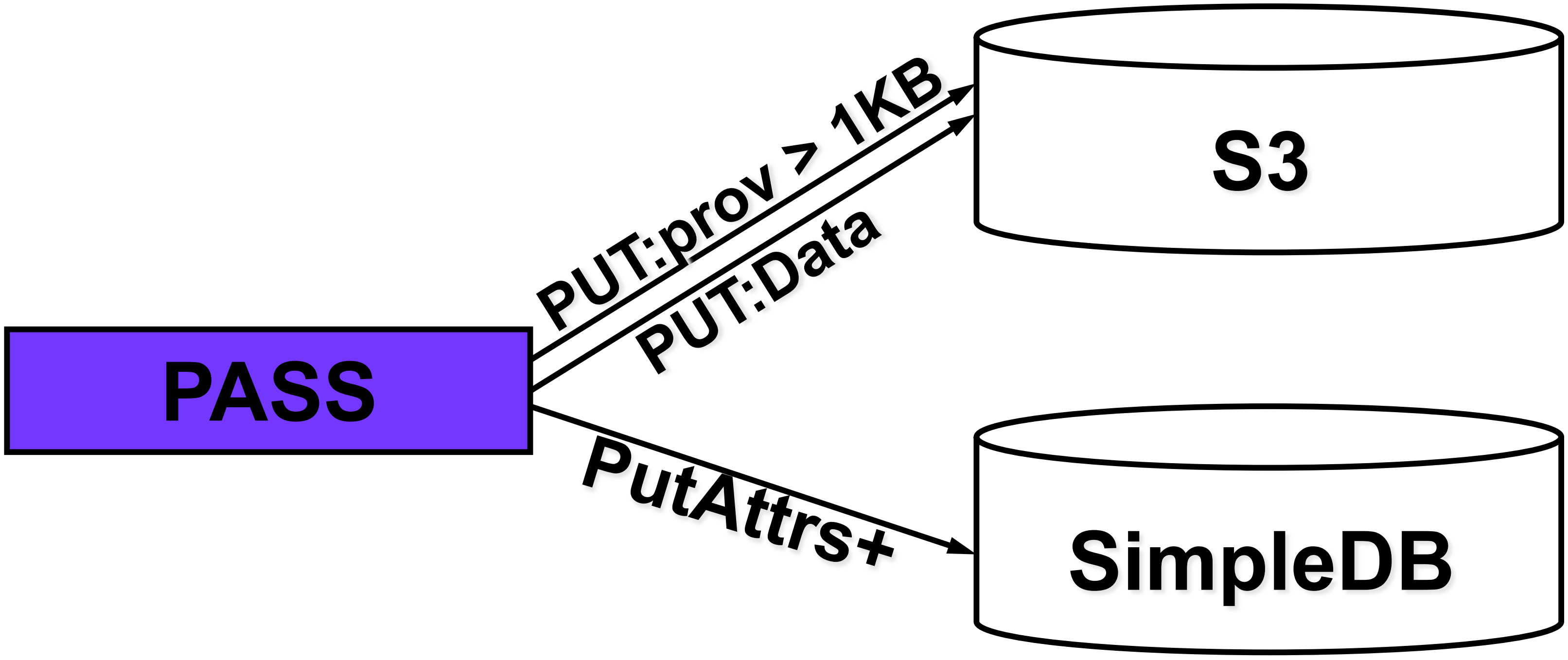
P2: Cloud Store + Cloud Database

- Store data in cloud blob store
- Store provenance in cloud database
- Amazon SimpleDB, Azure Table
  - Semi-Structured Data model: items described by attribute-value pairs
  - Operations: PutAttributes, GetAttributes, DeleteAttributes
  - Query: SELECT/LINQ
  - name/value size: 1KB or 64KB
  - Cost: bandwidth + storage + num ops + machine hrs

P2: Cloud Store + Cloud Database



P2: S3 + SimpleDB



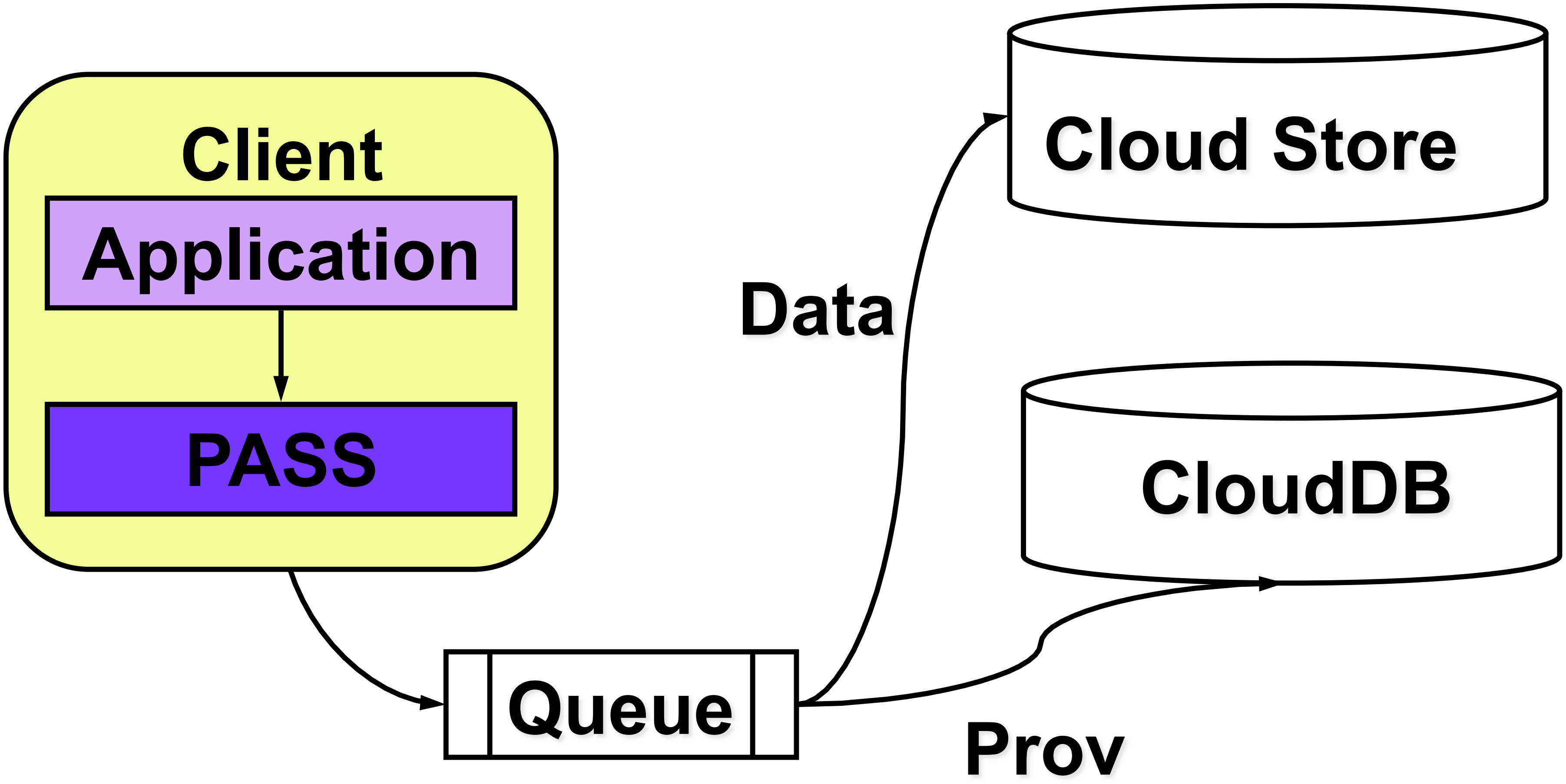
Properties

	Data Coupling	Causal Ordering	Persistence	Efficient Query
P1	✗	✓	✓	✗
P2	✗	✓	✓	✓

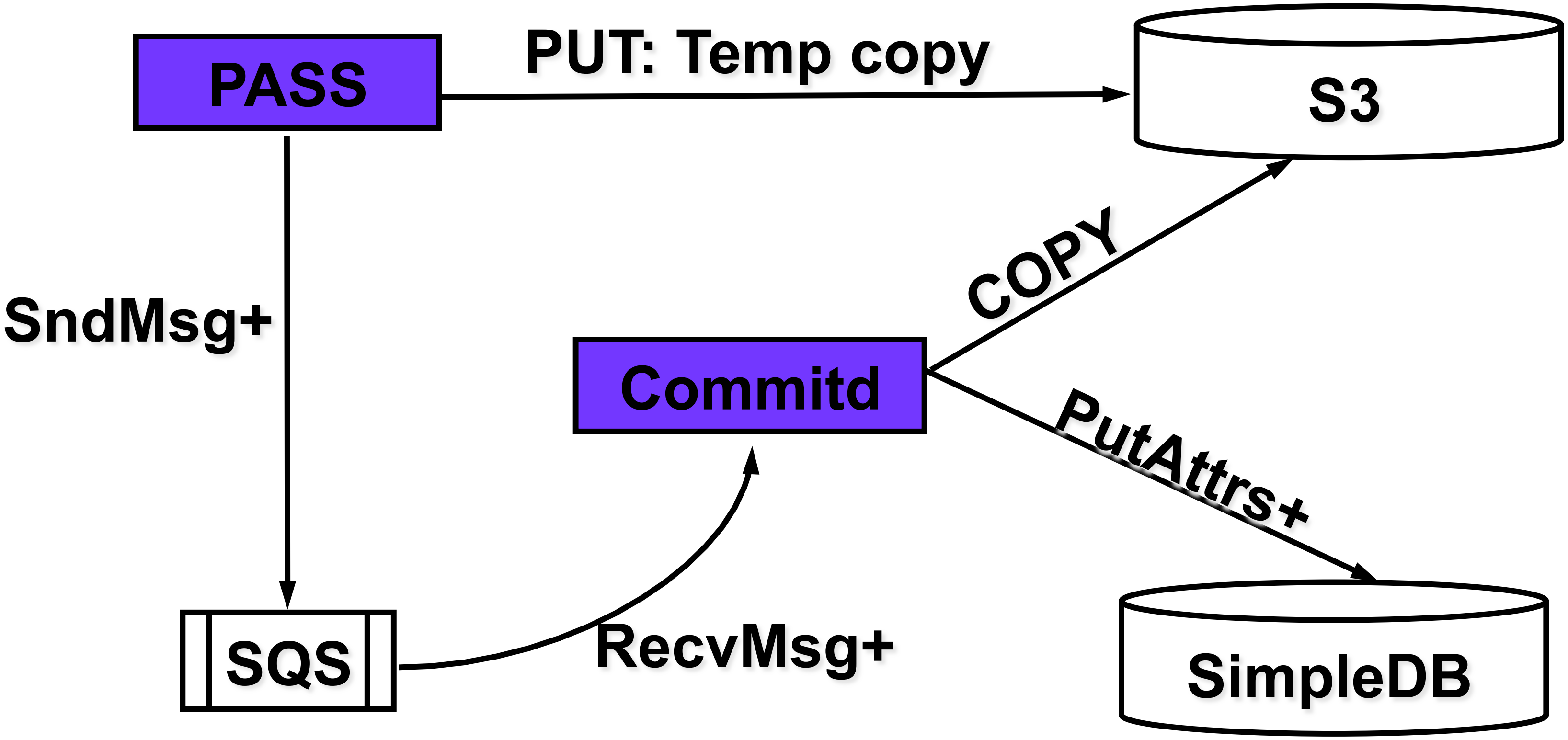
P3: Cloud Store + Cloud DB + Messaging Service

- P2 + use messaging service as a log
- Amazon Simple Queuing Service (SQS), Azure Queue
  - Distributed Messaging System
  - Queues are identified by URL
  - Operations: SendMessage, ReceiveMessage, DeleteMessage
  - Limits: 8KB message size

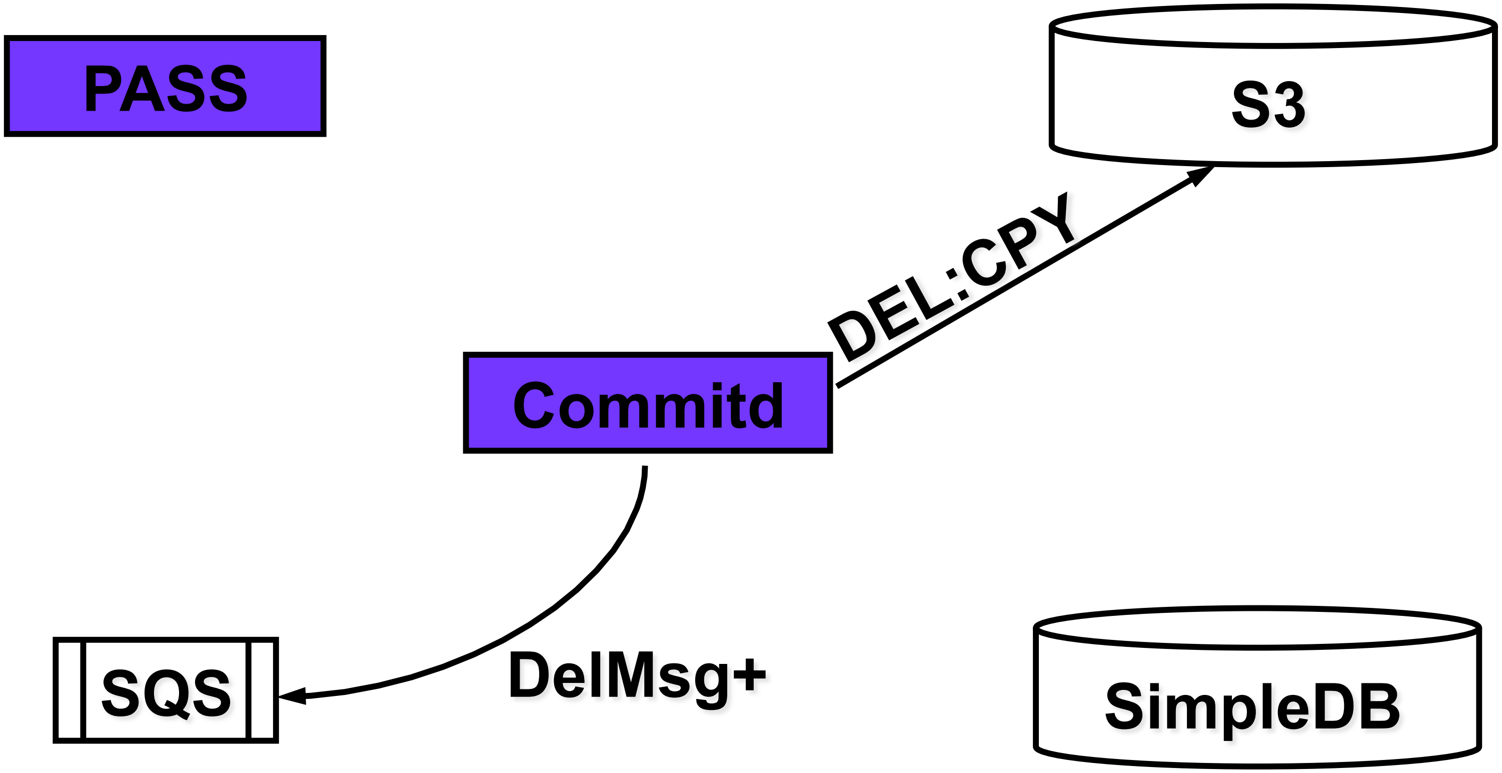
P3: Store + Database + Queue service



Protocol 3: S3 + SimpleDB + SQS



Protocol 3: S3 + SimpleDB + SQS



Properties

	Data Coupling	Causal Ordering	Persistence	Efficient Query
P1	✗	✓	✓	✗
P2	✗	✓	✓	✓
P3	✓	✓	✓	✓

Only ensure eventual data coupling

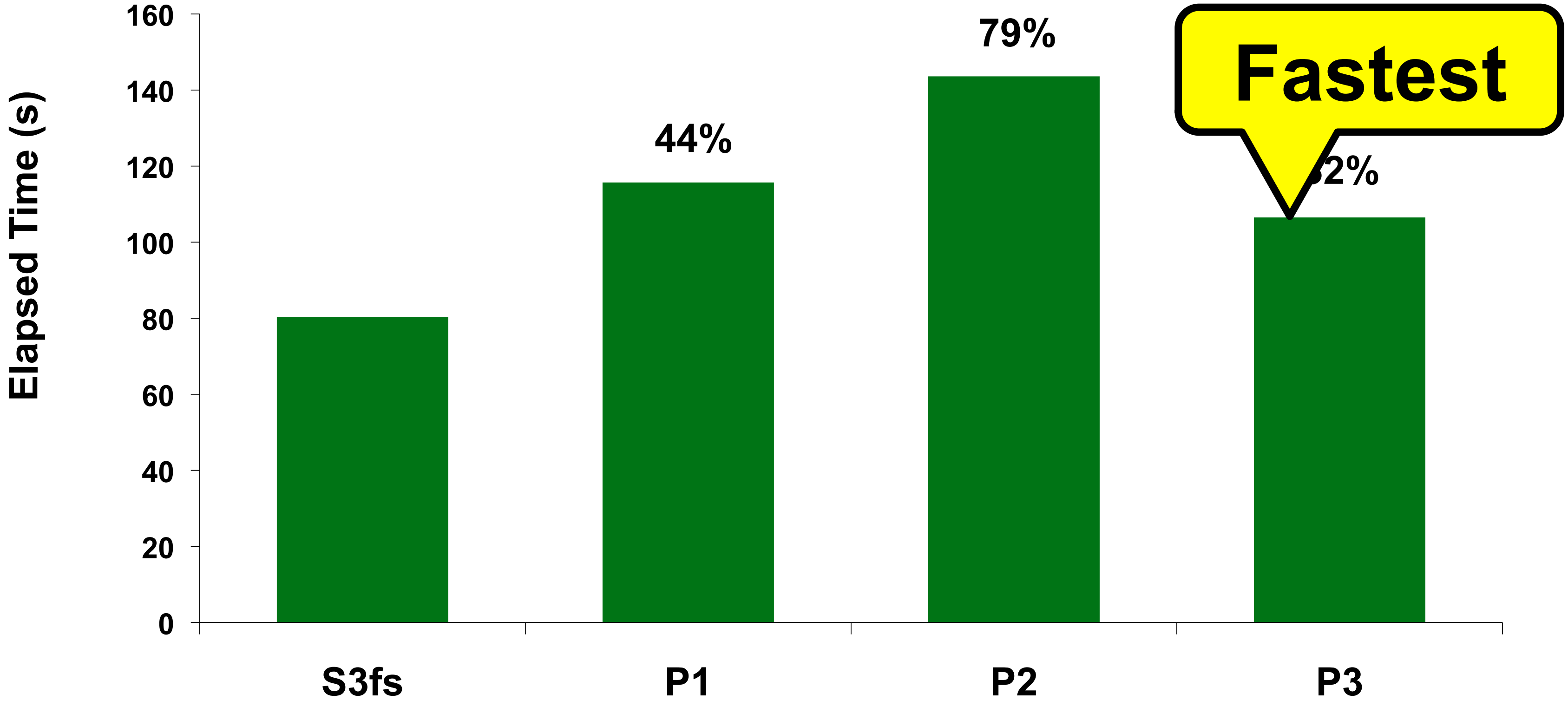
Outline

- Introduction
- Design Issues
- Protocols
- **Evaluation**
- Conclusions

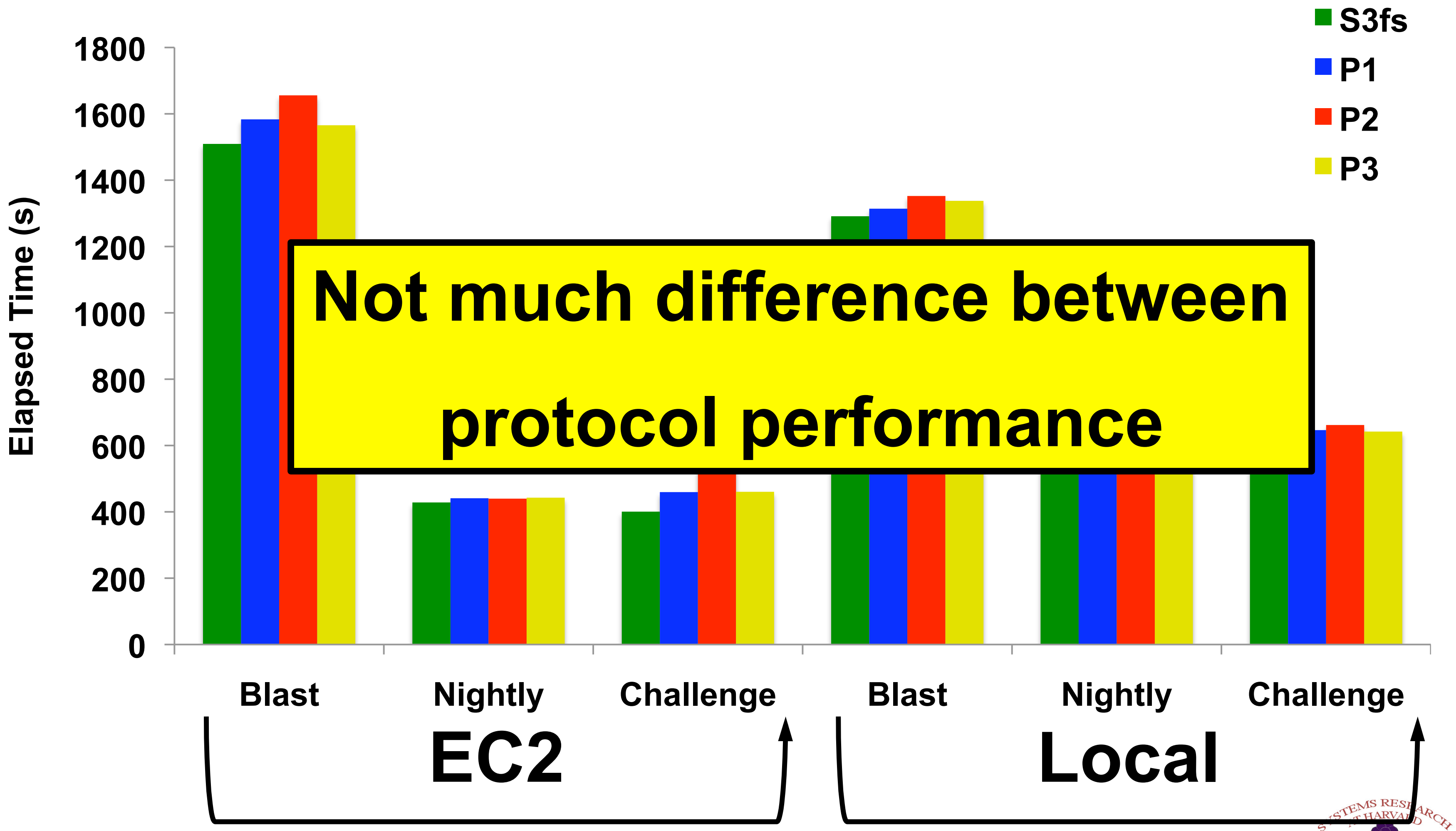
Evaluation

- Results AWS specific
- Baseline S3fs
- Workloads
  - Microbenchmarks
  - Application benchmarks
  - Query benchmarks
  - Cost overheads

MicroBenchmark Results



Application Benchmarks



Not much difference between protocol performance

Query Results

- Recall
  - P1 uses S3
  - P2,P3 use SimpleDB
- SimpleDB was much faster
  - Speedup depends on the query

Cost Overheads

	Nightly	Blast	Challenge
S3fs	$1.05	$0.37	$0.27
P1	$1.05	$0.39	$0.29
P2	$1.05	$0.38	$0.29
P3	$1.06	$0.40	$0.30

Evaluation Summary

- Obtaining statistical significance hard
  - Too many uncontrollable factors: WAN latency, service load, software version
  - Services seem to be getting better

Outline

- Introduction
- Design Issues
- Protocols
- Evaluation
- **Conclusions**

Conclusions

- We have shown how to store provenance in today’s cloud offerings
- Performance results show that we can use the most robust protocol
- Future work: Native cloud provenance
  - Architecture
  - Trusted provenance
  - Graph processing and provenance mining

Thanks!

- Questions?

kiran@eecs.harvard.edu

www.eecs.harvard.edu/~kiran