# Quality of Service Guarantees for Cloud Services

CS848 Project presentation

by Alexey Karyakin

David R. Cheriton School of Computer Science
University of Waterloo

March 2010

## Outline

1. Performance of cloud services (Amazon EC2, Amazon S3)
2. Current QoS frameworks (media streaming, network communication)
3. QoS in databases and other IT applications
4. QoS contract specification
5. Enforcing QoS requirements

# Implications of cloud computing on performance expectations

The cost of cloud services is claimed to be lower due to extensive sharing of resources;

However, a higher degree of sharing would automatically mean less *guarantees* of performance;

Since performance expectations of users may be affected by the resource sharing, *explicit* quality of service requirements could be used;

## The types of CPU usage pricing in Amazon AWS

1. Paid per machine-hour
   - Amazon EC2, Elastic MapReduce, Relational Database Services (MySQL)
   - On-Demand, Spot, Reservation pricing
2. Paid per request
   - Amazon S3, SQS
3. Paid per actual CPU consumption
   - Amazon SimpleDB

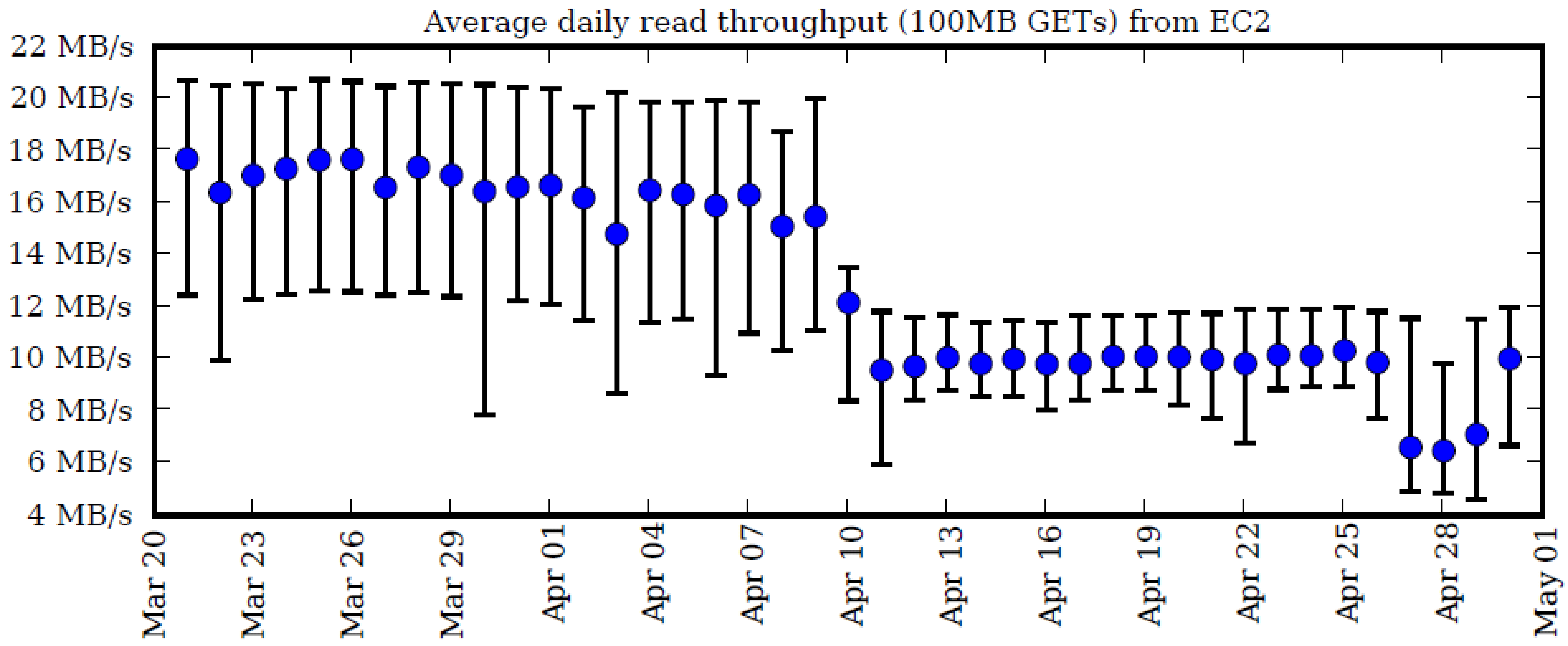
According to Amazon SLA, the provider can throttle any user's activity at any time

# Research of actual cloud performance

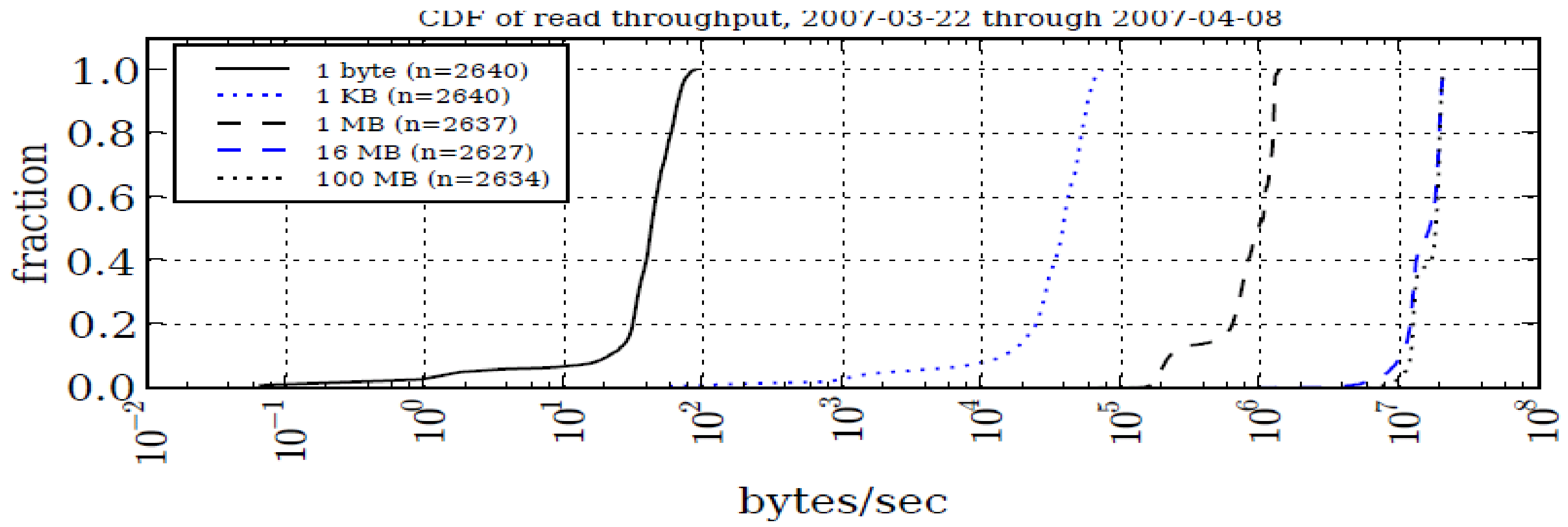
1. Simson L. Garfinkel, Technical Report TR-08-07: An Evaluation of Amazon’s Grid Computing Services: EC2, S3 and SQS, Harvard University, 2007

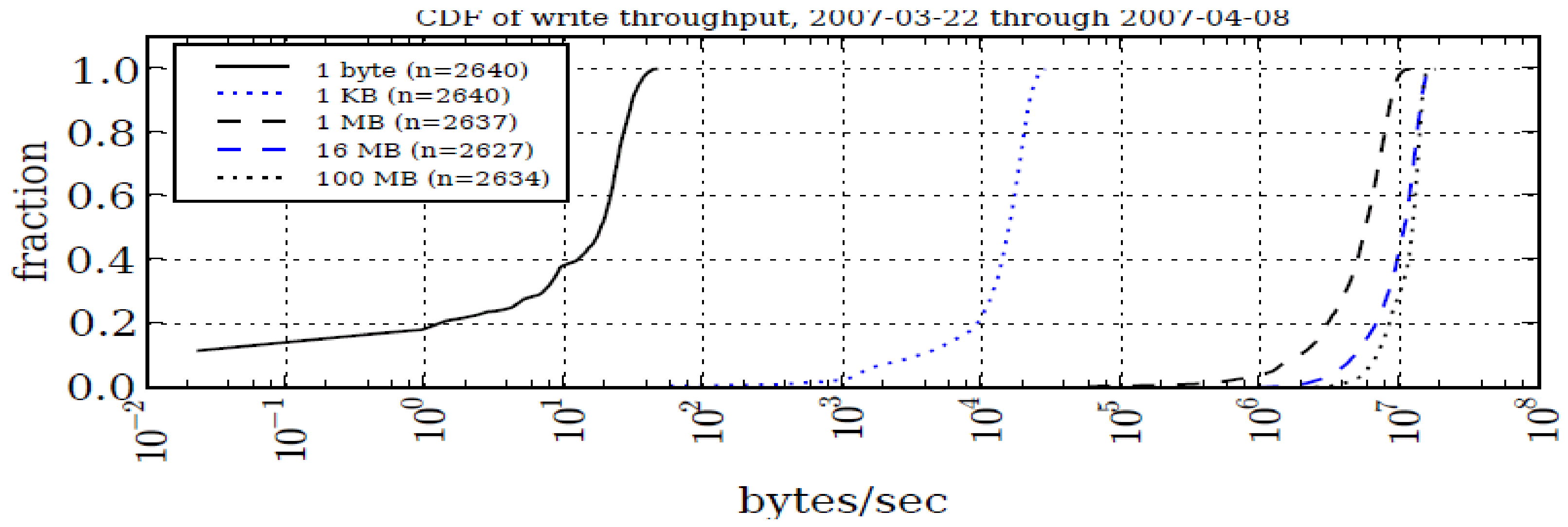
2. Edward Walker. Benchmarking Amazon EC2. In Proceedings of USENIX'2008, Volume: 33, Issue: 5, pages 18-23, October 2008.

# Garfinkel: variance of read throughput from EC2



# Garfinkel: variance of S3 throughput from EC2





# Garfinkel: some conclusion

1. Response time stability: between 10% to 20% requests suffer 5x worse performance;
2. For small requests (1B), response time for successive requests to the same bucket is independent, that mean that in case of delay an application could issue another requests that may complete sooner;
3. That is not the case for larger requests (100MB): several requests are likely to complete at the same time;
4. No legal guarantees of service; Amazon has a right to terminate the service any time for any reason
5. AWS is generally “fast, responsive , and very reliable, availability is excellent”

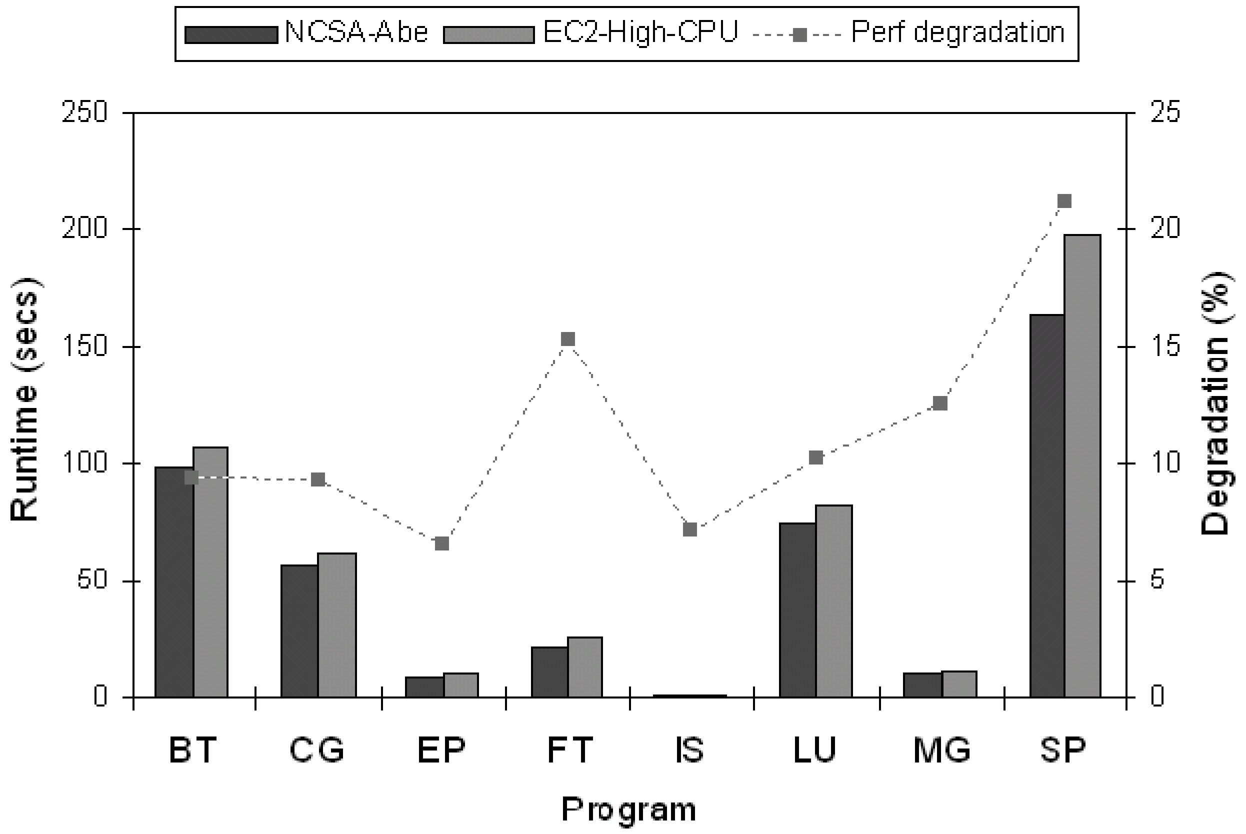
## Walker: NAS parallel benchmark

1. The benchmark containing 8 test programs simulating schientific computations, e.g. fluid dynamics;

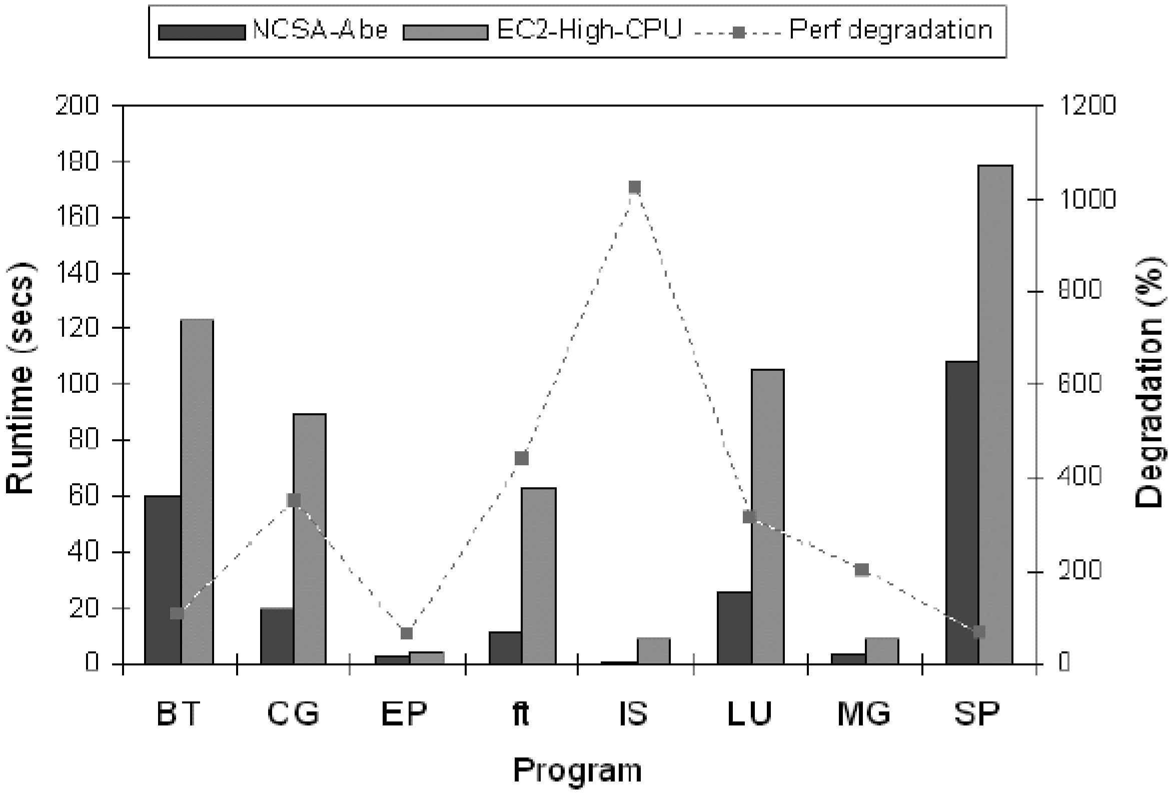
2. OpenMPI used for interconnect API

3. The configurations for the test are the same for a local cluster and EC2 “cluster”

# Walker: single node result



# Walker: cluster results



## Walker: some conclusion

1. The benchmark is a parallel scientific benchmark with high sensitivity to interconnect latency so great degradation in the cluster test comparing to Infiniband is expected;
2. In the single node test the degradation is moderate;

# Existing QoS frameworks: media streaming

1. End-to-end (intergation principle)
2. Declarative
3. Types of requirements:
    - flow synchronization – an analog to consistency
    - flow performance
    - level of service (deterministic, predictive, best-effort)
    - management policy (tradeoff between media quality and bandwidth)
    - cost of service
4. Mechanisms to achieve QoS:
    - QoS mapping to underlying services (thread priorities, etc)
    - admission testing
    - resource reservation protocol

# Existing QoS frameworks: network communication

1. Approaches:

    - intServ: reservation of resources, RSVP (RFC 2205)

    - diffServ: priorities based on the type of traffic, implemented as Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474)

2. QoS isn't widely used: only used in some enterprise networks, not on Internet, supported by some routers

3. Problems with internet QoS:

    - hard to support reservation in high-volume routers for thousands of concurrent sessions

    - encryped traffic cannot be inspected

    - ability to abuse QoS policies by both users and providers

4. It is argued that increasing internet bandwidth is a better way to meet users' quality expectations instead of employing some form of QoS

## Types of quality (QoS dimensions)

1. Performance: response time, mean throughput
2. Quality of result set: data “freshness”, correctness, amount of data returned
3. Economic factors of query execution – especially important for a scalable utility service

Those requirements are often contradictory.

## Real-time DB approach

A query must complete within a specified time. Otherwise, the query can be canceled or “downgraded” to low priority execution.

Implemented using resource reservation and priorities.

Much research exists on real-time databases, including distributed ones. Some interesting problems include scheduling, concurrency control and priority inversion avoidance.

Real-time database approach is usually not considered for “utility” service providers as it is in conflict with the concept of resource sharing in order to minimize costs.

## SLA approach

Service Level Agreement (SLA) specifies an averaged restrictions on QoS performance metrics, over a (relatively) long time interval.

Quality requirements can be specified for a class of queries or for each individual query.

The goal is to maximize the percentage of queries that fulfill the constraints.

# Economic models of query execution

Historically, many factors are included in query cost estimation:

- CPU cycles
- disk I/O, etc, etc

Economic models can combine those factors, along with query QoS requirement into single model.

The model defines Wealth value, corresponding to the importance of the workload. Resource brokers “sell” resources to Consumers to maximize its profits. Consumers compete for resources on a auction, trying to use maximize utility/cost ratio.

SLA can be integrated into the Wealth definition.

Economic models naturally fit into cloud computing infrastructure as using resources really costs money.

# Strategies to meet SLA requirements

1. Adaptive query execution

    - selecting one of many sub-plans during query execution based on its tradeoffs and the current execution monitoring data

2. Dynamic resource allocation (e.g. using feedback control loop)

    - most commonly, CPU time and memory (caches, buffer pool) is managed

    - resource allocation can be dynamically split between classes of queries. Particularly, many publications discuss optimal buffer pool allocation between parts of workload.

3. Admission control

    - Multi-Programming Limit (MPL)

4. Migration of query execution or data between physical resources

Fin