Evaluating Storage Systems for Scientific Data in the Cloud

Ketan Maheshwari, Justin M. Wozniak, Hao Yang,

Daniel S. Katz, Matei Ripeanu, Victor Zavala, Michael Wilde

Argonne National Laboratory

University of Chicago

University of British Columbia

Introduction

- Clouds offer ad-hoc clusters with computation, storage and networking resources to carry out distributed application execution
- To effectively utilize these resources, additional setup and systems are required
- Goals of the current work:
  - Characterize IaaS clouds for data oriented applications
  - Evaluation of contemporary storage solutions on clouds
  - Combine Many-Task execution systems with backend storage solution providers to obtain an operational environment for application execution and report on performance

Motivation

- According to a 2013 XSEDE cloud survey report, a majority of users have difficulty in managing data in clouds. About 27% of the users use the Amazon S3 storage system for their data needs.
- A quote from a 2011 report on Magellan experience:

  Tools [are needed] to simplify using cloud environments ... and enhancements to Map Reduce models to better fit scientific data and workflows [are needed] for scientific applications.

- Big Data and increasingly I/O intensive workflows
- Different application requirements: read, write, read-after-write
- Availability: In clouds, node-local storage is available during the life of a VM instance and can be effectively utilized

Overview

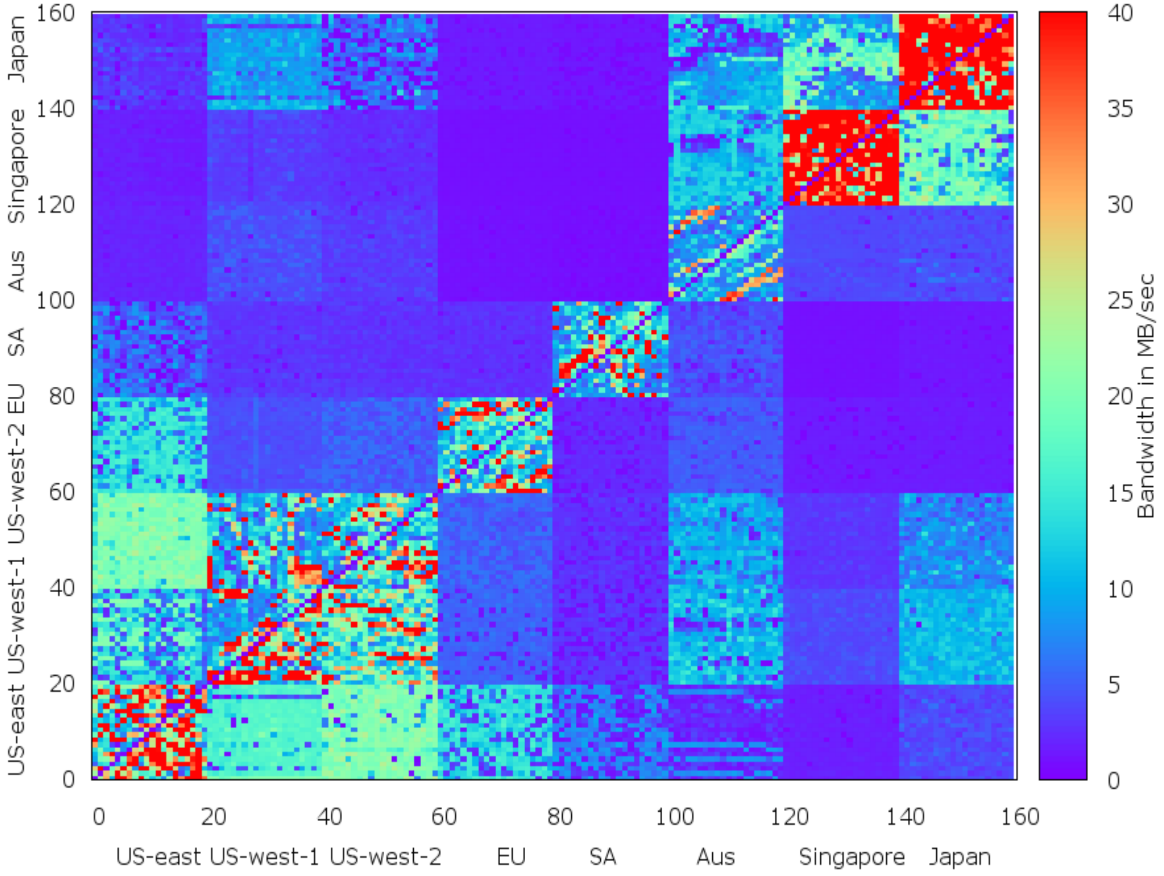
- Introduction
- Motivation
- The Nature of the cloud
  - Network characteristics between cloud regions
- Storage systems
  - MosaStore, Chirp/Parrot
  - Amazon S3, HDFS
- Swift
- Experiments
  - Raw Performance
  - Real-World Applications
  - Application Results
- Summary

The Nature of the Cloud

- Physically, cloud systems comprise of geographically distributed resources.
- Unlike traditional clusters, these resources are non-uniformly distributed with irregular connectivity
- Crucial to understand the network connectivity for data oriented distributed applications in the clouds
- We perform two experiments on Amazon AWS cloud:
  - Measure bandwidths between instances of each of the eight global regions
  - Measure latencies between instances of each of the eight global regions
- We chose a representative 20 instances from each region resulting in a 160X160 matrix

Cloud Regions Bandwidths: Some Observations

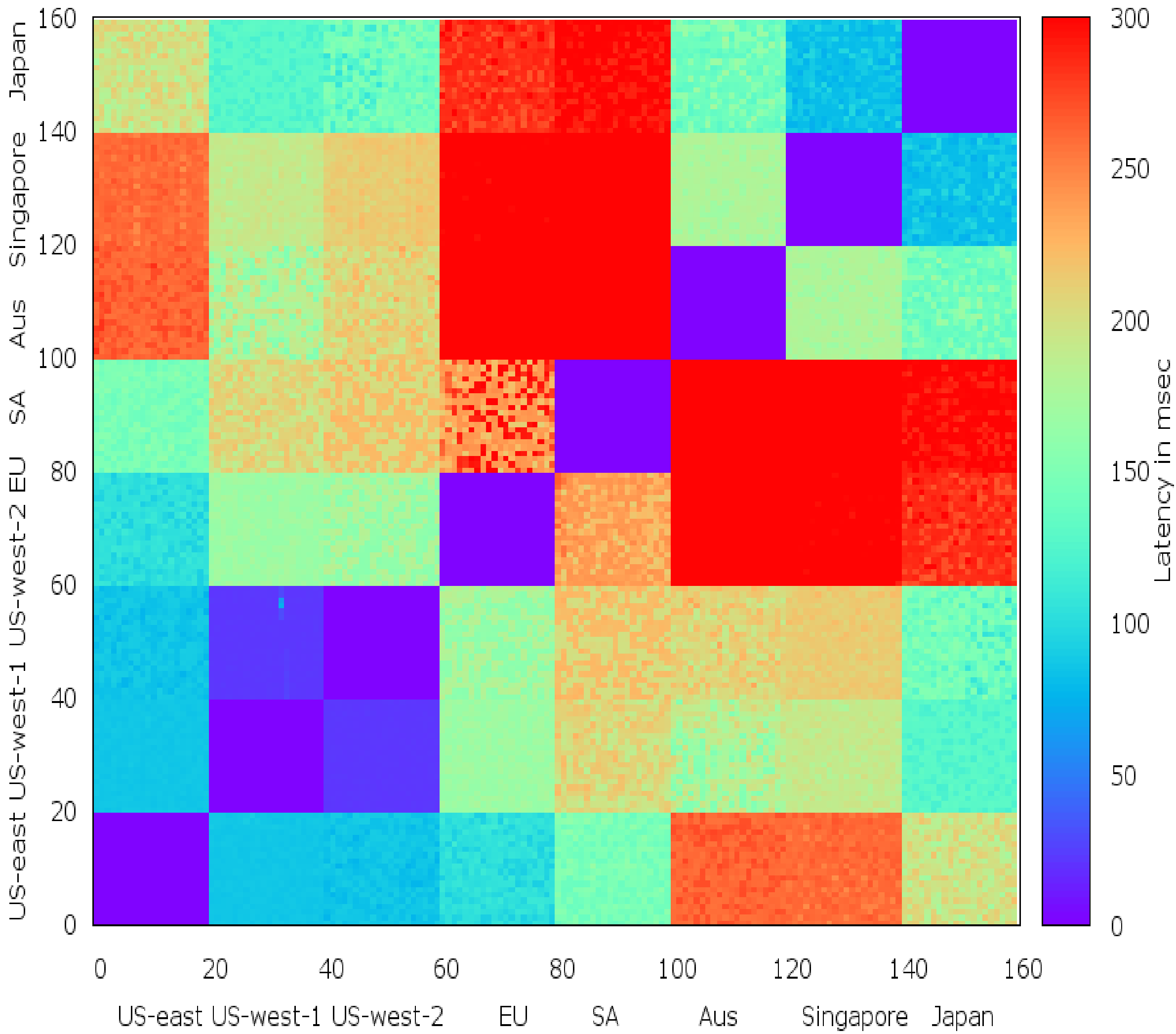
- North American regions well-connected
- EU well-connected to US-east
- Aus well-connected to US-west and Japan, Singapore
- Japan and Singapore well-connected among themselves but poorly connected with rest of the world



Maheshwari et. al., swift-lang.org

Cloud Connectivity: Latencies

- Similar pattern as bandwidths (lower the better)
- More symmetrical and islands
- Fast connections between US regions
- Fast connections between Aus-Singapore-Japan



Conclusions from Cloud Network Analysis

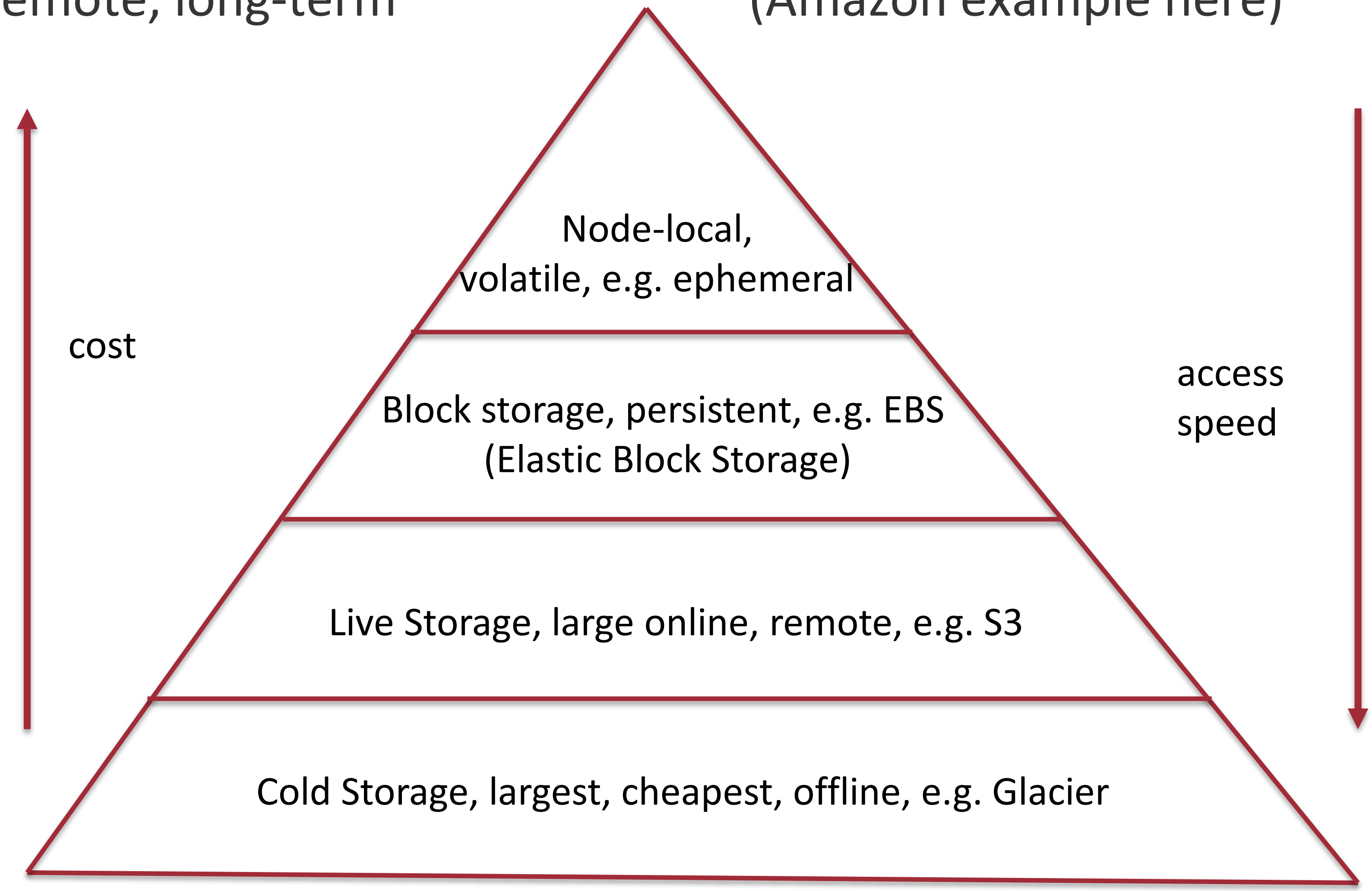
- Want to answer: How much data can we move in cloud and how fast?
- Resources from global cloud must be chosen carefully to improve performance versus cost
- For instance, a cluster of 1000 nodes between Japan and Singapore might be faster than the one between US-east and US-west
- Isolated regions such as South America and EU with one datacenter each may not be combined with other regions for distributed computing
- Smart storage strategies are very relevant in this scenario: exploit locality, replication, caching
- Carefully chosen storage servers can benefit cloud executions

Overview

- Introduction
- Motivation
- The Nature of the cloud
  - Network characteristics between cloud regions
- Storage systems
  - MosaStore, Chirp/Parrot
  - Amazon S3, HDFS
- Swift
- Experiments
  - Raw Performance
  - Real-World Applications
  - Application Results
- Summary

Storage Systems

- Clouds offer different storage solutions: node-local, extended, remote, long-term (Amazon example here)



Maheshwari et. al., swift-lang.org

Storage Systems

- Clouds offer different storage solutions: node-local, extended, remote, long-term

- Modern performance oriented storage systems
- Widely used in modern cloud applications: e.g., Google Drive
- Why are they important?
  - Gives unified view of distributed physical systems
  - Fast, synchronous, consistent
  - Enables implicit data movement across shared-nothing nodes
- Example systems: Distributed File systems, Key-Value stores
- Here we evaluate:
  - Research storage systems: Mosastore, Chirp/Parrot
  - Commercial storage systems: Hadoop HDFS, Amazon S3

Research Storage Systems: Chirp and MosaStore

Chirp

- A user-level storage system that provides a virtualized, unified view of data over multiple real file systems (e.g., over file systems deployed over independent clusters)
- Parrot is an interceptor layer that traps an application's POSIX file system calls and redirects them to Chirp
- A combination of Parrot and Chirp can thus provide a POSIX-accessible storage environment

MosaStore

- A low-overhead, user-level distributed storage system based on FUSE
- Optimize data distribution under-the-hood via striping and replication
- Can expose the details of data location for workflow level optimization

Commercial Storage Systems: Amazon S3 and Hadoop HDFS

Amazon S3

- A remote object storage system provided by Amazon
- Access via a get/put API or FUSE-enabled mount
- Preconfigured and ready-to-use but a paid service

Hadoop HDFS

- A High-throughput filesystem designed to store data on share-nothing cluster of machines
- Well-suited to node-local computational models such as MapReduce but can be used with workflow models via external APIs

Overview

- Introduction
- Motivation
- The Nature of the cloud
  - Network characteristics between cloud regions
- Storage systems
  - MosaStore, Chirp/Parrot
  - Amazon S3, HDFS
- Swift
- Experiments
  - Raw Performance
  - Real-World Applications
  - Application Results
- Summary

Swift

- A parallel scripting framework with many-task dataflow execution system
- Swift composed workflows drives the execution and data movements concurrently in conjunction with application logic thus stressing the underlying storage systems
- Two implementations
  - Classic Swift/K (Karajan), mostly HTC oriented, single task store (submit host), uses explicit data movement on non-storage enabled, non-shared filesystems, has some optimizations for collective data movement
  - New Swift/T (Turbine), more HPC focused, distributed task store, much faster task dispatching rates, requires shared storage systems (either physical, e.g. HPC, or via software, e.g. w/ Mosa on clouds)

Maheshwari et. al., swift-lang.org

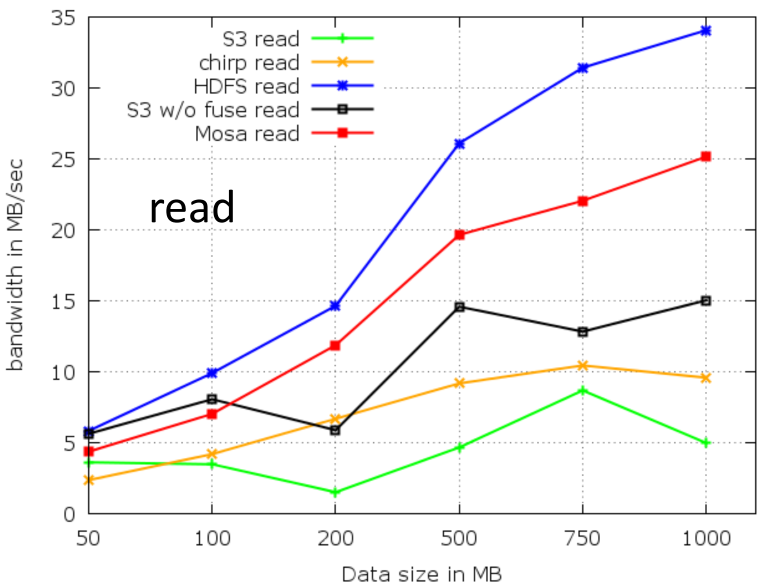
Overview

- Introduction
- Motivation
- The Nature of the cloud
  - Network characteristics between cloud regions
- Storage systems
  - MosaStore, Chirp/Parrot
  - Amazon S3, HDFS
- Swift
- Experiments
  - Raw Performance
  - Real-World Applications
  - Application Results
- Summary

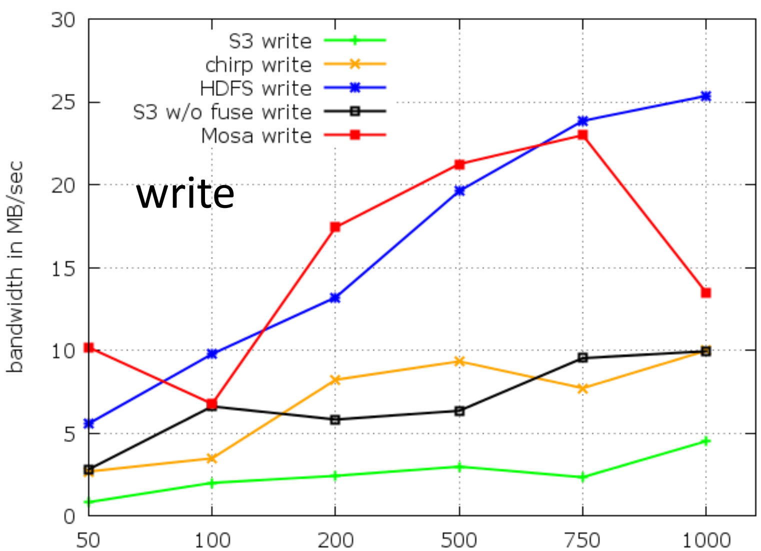
Experiments

- Workflow-driven raw I/O performance benchmarks:
  - Concurrent reads from storage system to local file system
  - Concurrent writes to storage systems from cloud nodes
  - Read-after-Write
- Used 40 “m1.large” (2-cores, 8G memory) Amazon instances spread between two regions: US-east and US-west
- Measure bandwidths for data sizes: Between 50 and 1000 MB
- Mosa, Chirp and HDFS use node-local storage to aggregate space
- S3 use remote S3 object store via FUSE-mounted S3FS and remote get-put operations on named S3 bucket

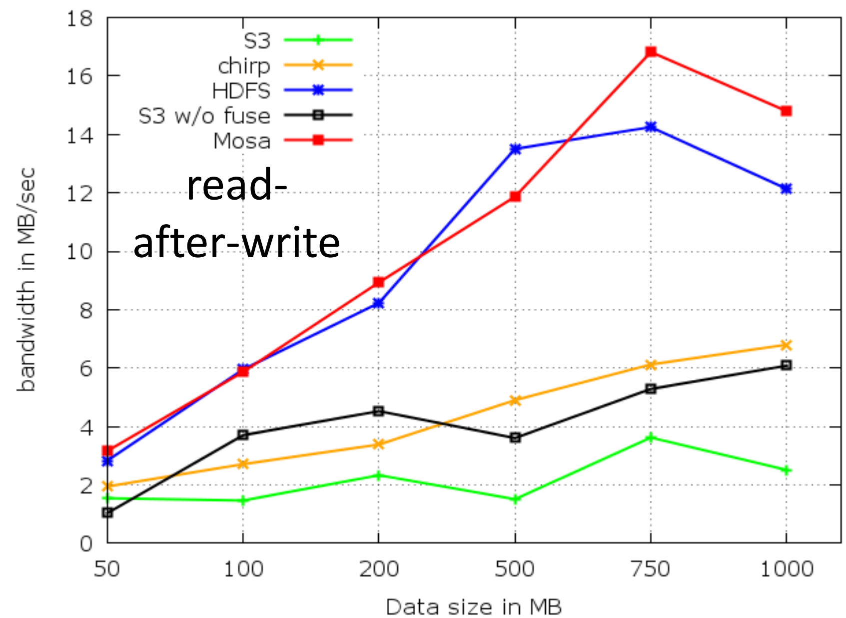
Raw performance benchmarks



- HDFS and MosaStore leads the performance
- In the crucial read-after-write benchmarks, both MosaStore and HDFS performs closely with MosaStore outperforming HDFS for large data sizes
- Amazon S3 remote storage significantly slower than MosaStore and HDFS
- We chose MosaStore for further application execution

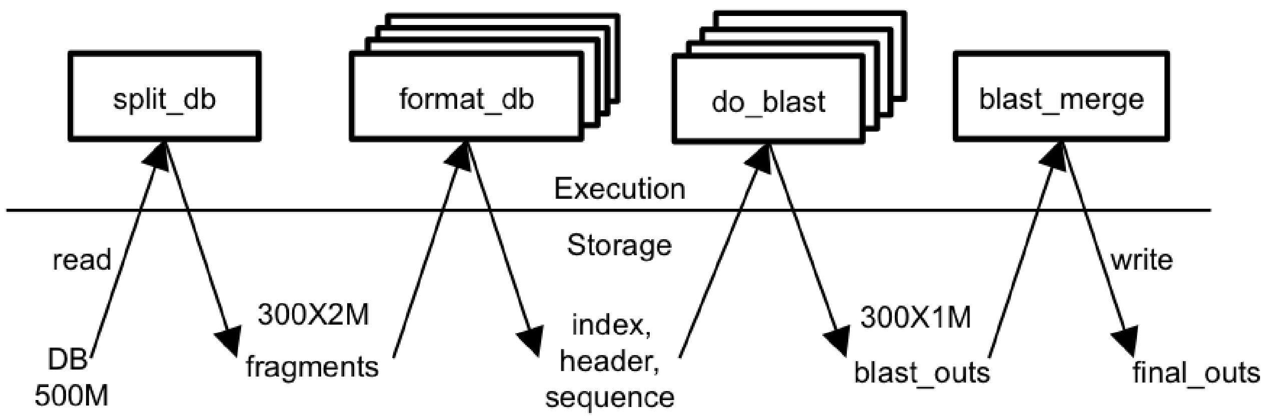


Maheshwari et. al., swift-lang.org



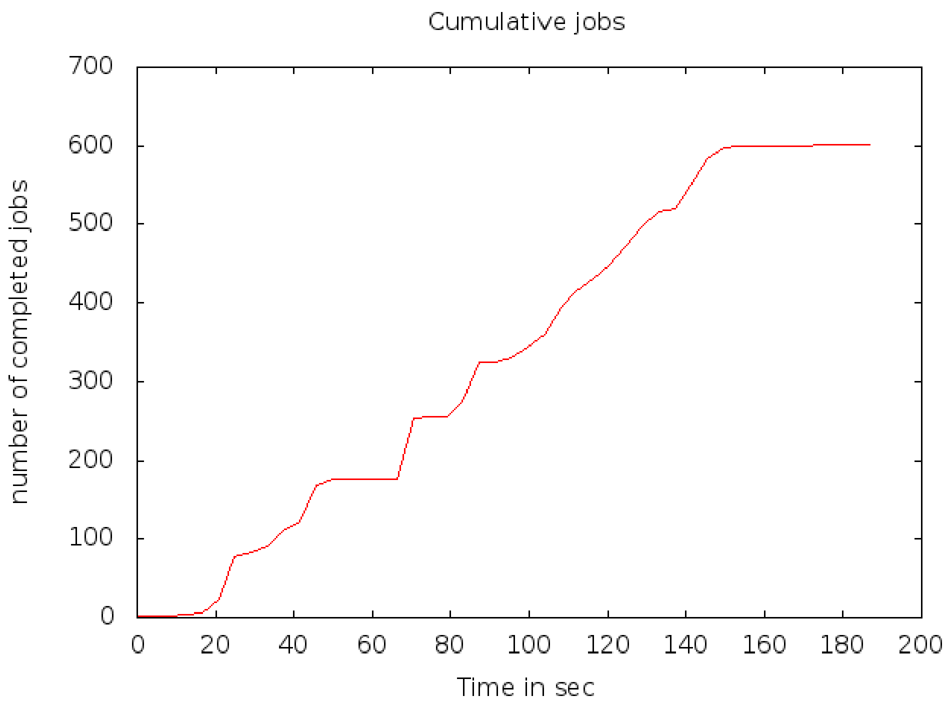
Real-World Applications (1) : Parallel BLAST

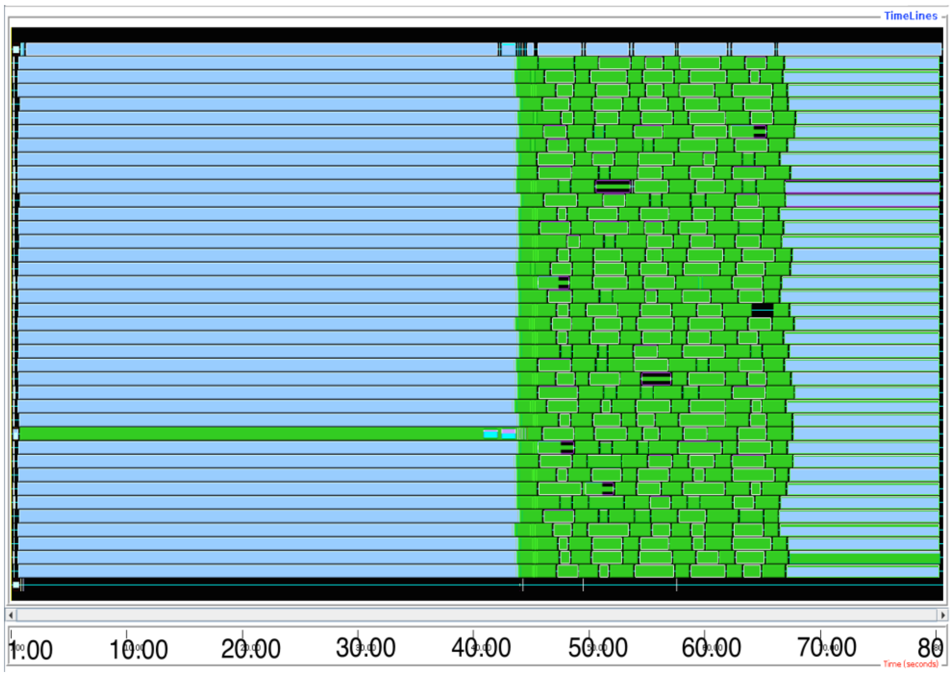
- A protein alignment search tool, BLAST performs searches from a given protein database.
- *Parallel* BLAST splits the protein database into fragments and runs many instances of BLAST simultaneously over the split database.
- The results from each of the fragment search are merged to give the final result.



Maheshwari et. al., swift-lang.org

Application results: Swift running Parallel BLAST on Amazon with MosaStore





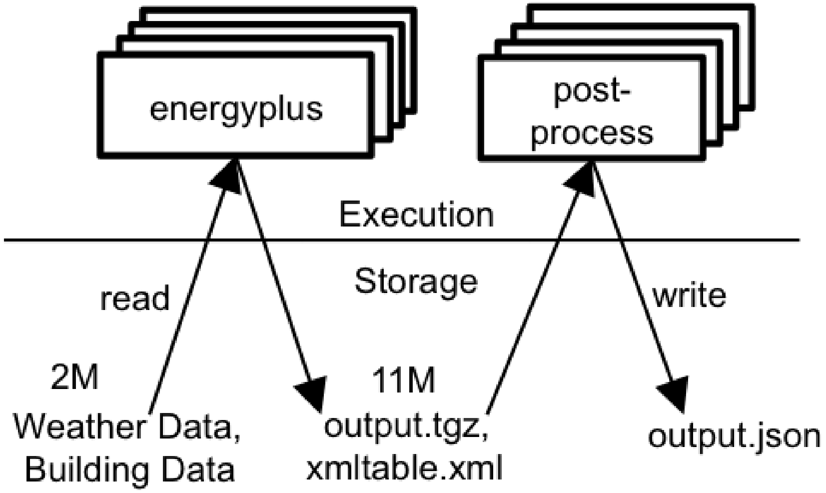
Explicit data movement between cloud instances with Swift/K

Implicit data movement by Mosastore using Swift/T
44% faster than explicit movement

Maheshwari et. al., swift-lang.org

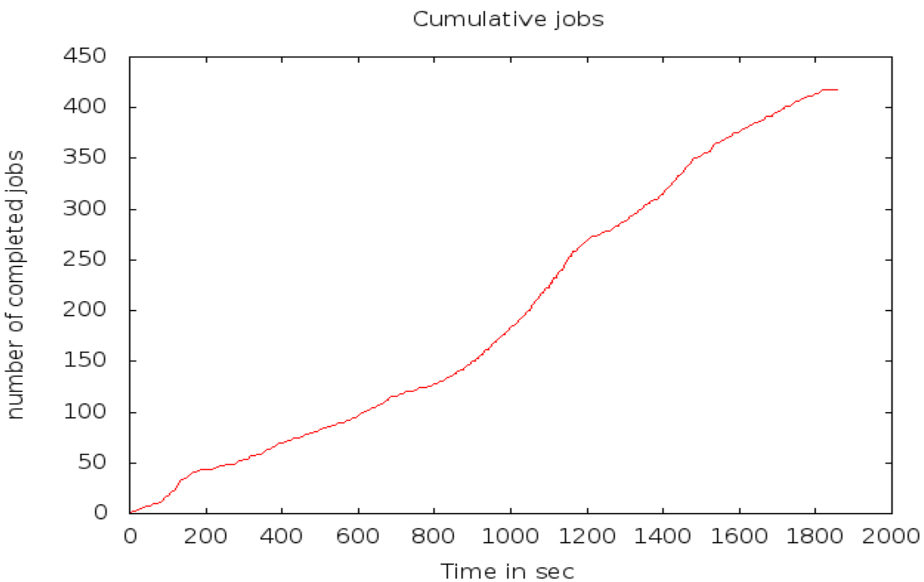
Real-World Applications (2) : EnergyPlus

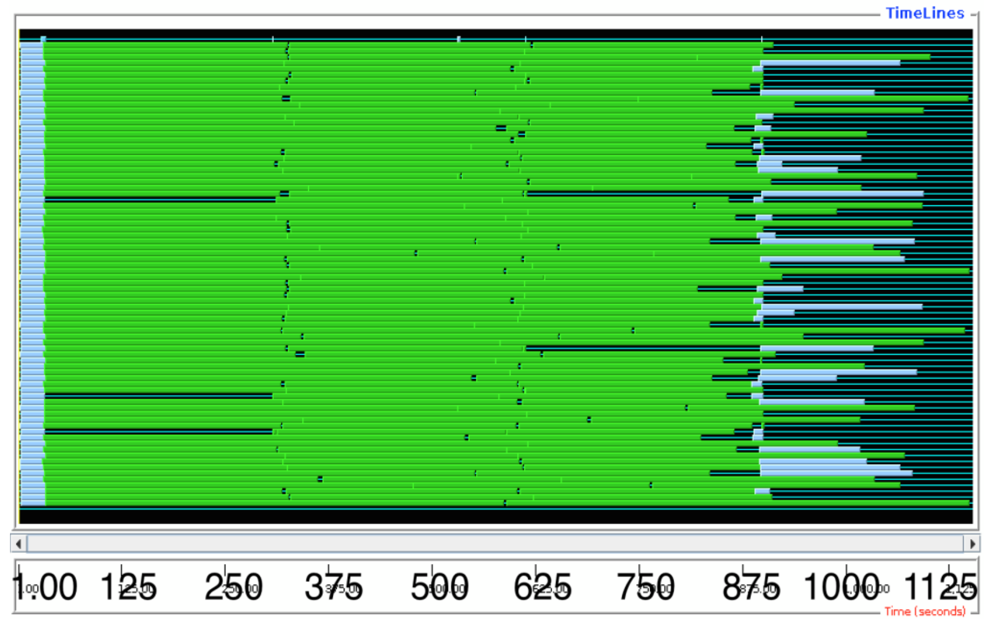
- A suite of energy analysis and thermal load simulation programs for buildings.
- Takes an ensemble of climate, historical and structural parameters as input and projects the future energy requirements
- Two steps: run ensemble and do results formatting as post-process.



Maheshwari et. al., swift-lang.org

Application results: Swift running EnergyPlus on Amazon with MosaStore





Explicit data movement between cloud instances with Swift/K

Implicit data movement by Mosastore using Swift/T
59% faster than explicit data movement

Maheshwari et. al., swift-lang.org

Summary

- Globally implemented clouds rely heavily on Internet backbone, resulting in non-uniform and variable network characteristics, which application deployments must take into account
- Applications with medium immediate storage requirements can run effectively by aggregating the cloud node-local space with the help of storage solutions; these solutions almost always perform better that the dedicated object store provided by clouds such as Amazon S3
- Swift has been shown to perform better on clouds with implicit files systems (e.g. MosaStore), but can fall back to explicit data movement if needed

Acknowledgements

- Swift is supported in part by NSF grants OCI-1148443 and PHY-636265, NIH DC08638, DOE Office of Science ASCR Division, and the UChicago SCI Program
- Swift Team:
  - Tim Armstrong, Ian Foster, Mihael Hategan, Daniel S. Katz, David Kelly, Justin Wozniak, Mike Wilde, Justin Wozniak, Zhao Zhang
- Science application collaborators discussed in the paper:
  - Argonne Power grid simulation project: V. Zavala, M. Hereld
- Some work by DSK was supported by the National Science Foundation, while working at the Foundation. Any opinion, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.