Peer-to-Peer Data Sharing for Scientific Workflows on Amazon EC2

Rohit Agarwal

Indian Institute of Technology, Ropar

ragarwal@iitpr.ac.in

Gideon Juve, Ewa Deelman

USC Information Sciences Institute

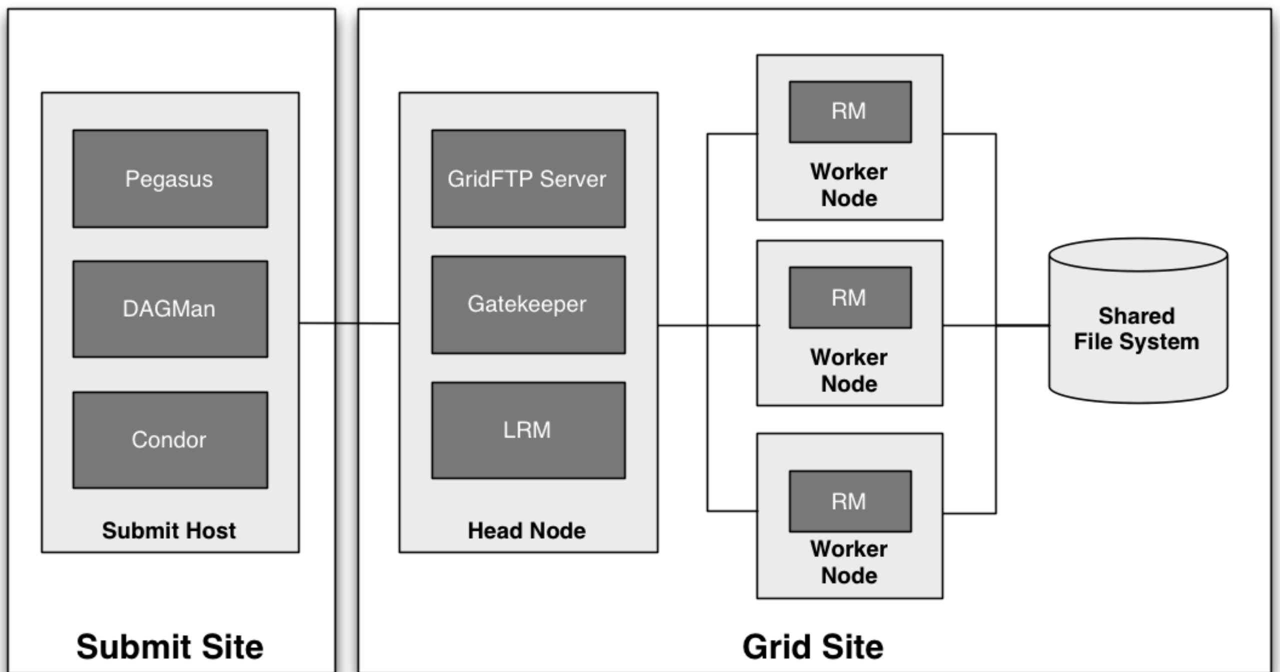
{gideon,deelman}@isi.edu

Workflows in the Cloud

- Advantages
  - Provisioning (compute and storage)
  - Elasticity
  - Reproducibility
  - Appliances (e.g. Galaxy)
  - Control over environment (esp. for legacy)
- Disadvantages
  - Administration
  - Virtualization overhead
  - Resource limitations (not really infinite, no queuing)
  - Cost relative to alternatives (campus clusters, grid)
  - Cost/Performance tradeoffs

Deploying Workflows in the Cloud

- Could develop Workflow as a Service (PaaS or SaaS)
- Can deploy existing software on IaaS clouds
- "Virtual Clusters"
- New tools: Nimbus Broker, cloudinit.d, Wrangler, Precip



Motivations for this Work

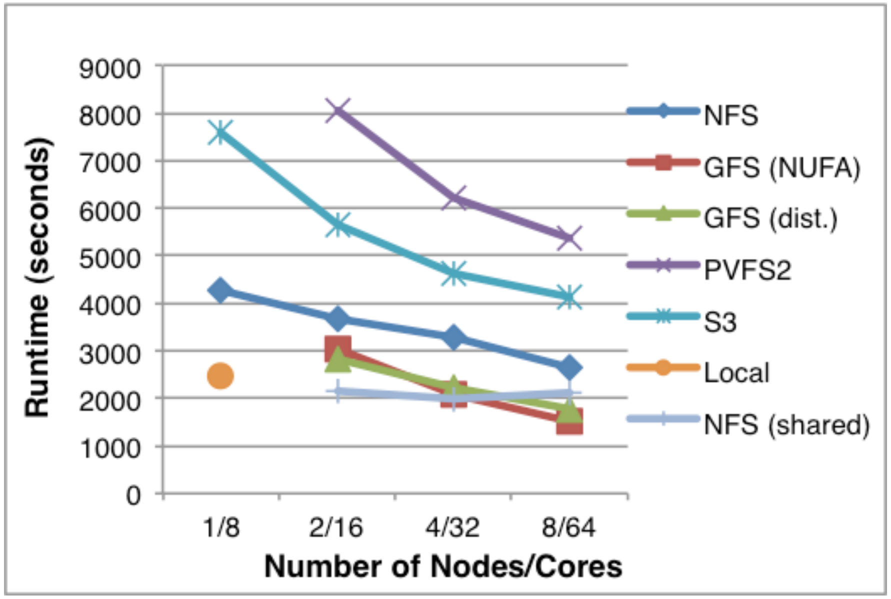
- Data-intensive workflows are limited by I/O performance
  - I/O is becoming the bottleneck rather than throughput
- Many workflows share data using files
  - Task A writes a file, task B reads it
  - File management is critical
- Write-once
  - Typically, files are only written once, never updated
  - Can replicate files without worrying about consistency
- Three ways to share files
  1. Use a shared storage system (POSIX or non-POSIX)
  2. Transfer files from submit host to workers and back
  3. Transfer files directly from one worker to the next

Previous Study on Data Sharing Options

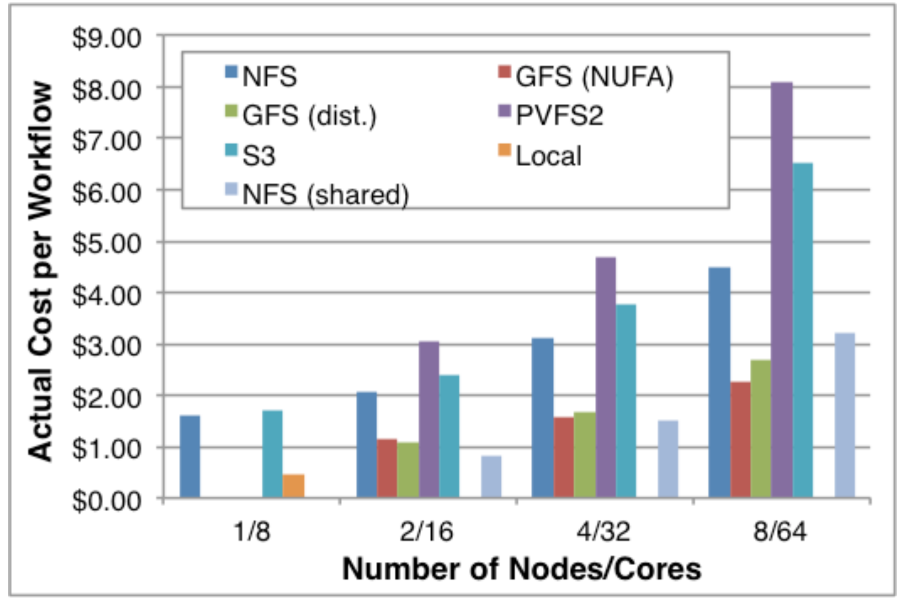
- **Goal**
  - Better understand how storage systems affect performance
  - Compare storage costs on commercial clouds
- **Deployed several different storage systems**
  - Local, NFS, S3, PVFS2, GlusterFS (distribute and NUFA)
- **Used three different workflow applications with different resource requirements**
  - Montage (astronomy, data-intensive)
  - Broadband (seismology, memory-intensive)
  - Epigenome (bioinformatics, CPU-intensive)
- **Compared performance and cost of different file system options**

G. Juve, et al., "Data Sharing Options for Scientific Workflows on Amazon EC2", Supercomputing, 2010.

Results for Montage



Makespan



Cost

- **PVFS didn't handle small files well**
- **S3 had too much overhead**
- **NFS did comparatively well**
- **GlusterFS came out on top**

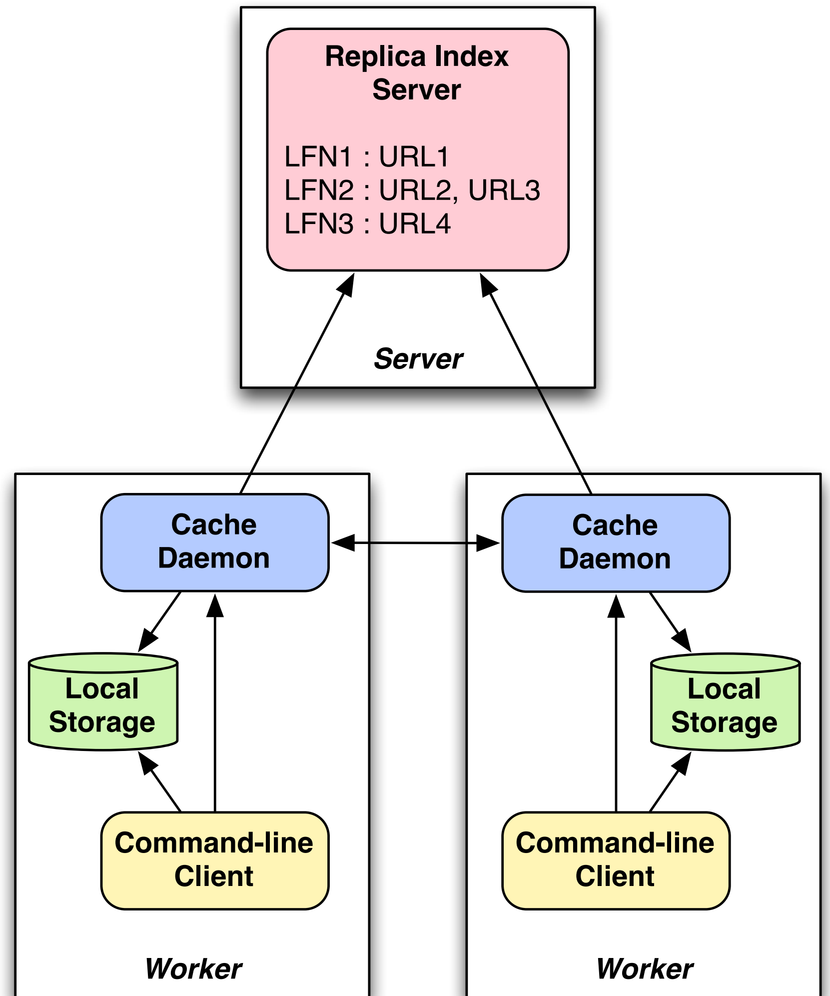
- **NFS and S3 have extra costs**
- **Performance improvement does not offset increased cost**

Approach

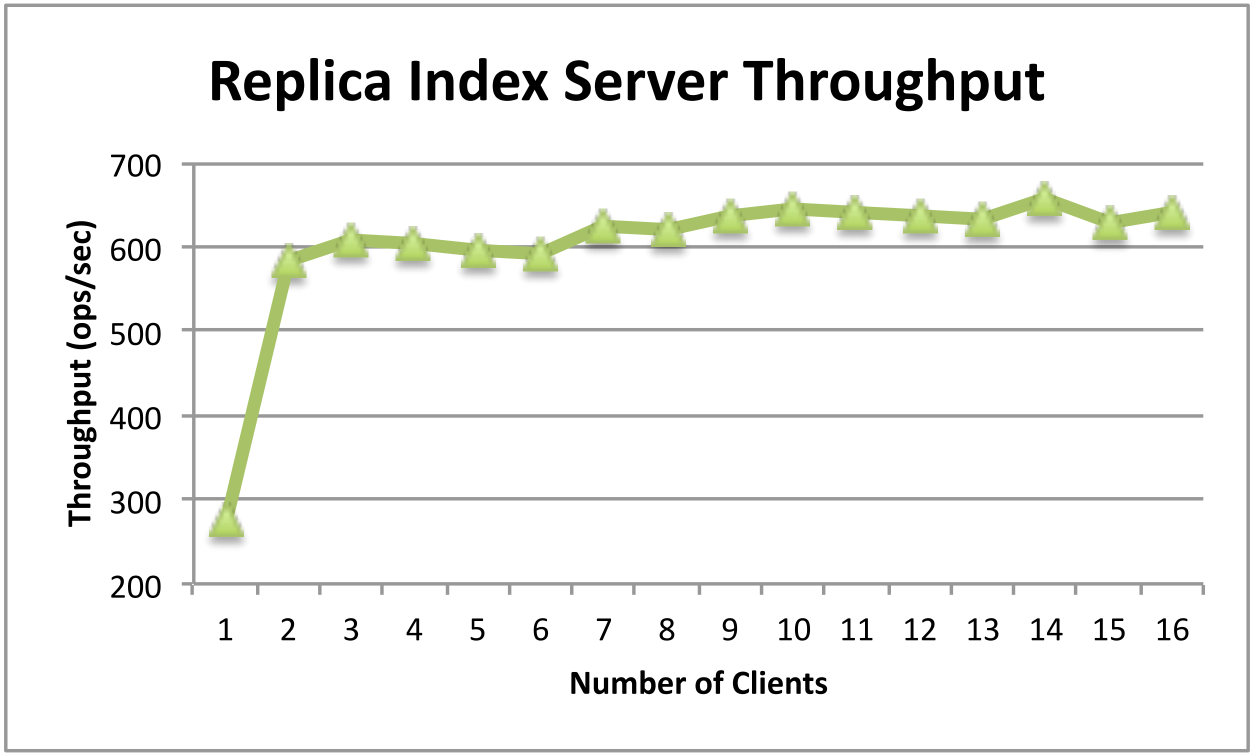
- Develop storage service to facilitate peer-to-peer transfers
  - Applies to environments other than clouds
- New files are written to the local disk
  - No network I/O for writes
- Files are replicated on-demand
  - Each time a task runs on a worker, all of its input files are replicated to that worker
- Files cached on each worker node
  - Enabled by write-once, no consistency issues
- Workflow tasks are wrapped by I/O operations
  1. Fetch input files
  2. Run task
  3. Register output files

System Design

- Replica Index Server
  - Stores mappings of logical file names to URLs
- Cache Daemon
  - Manages local storage on each worker
  - Serves local replicas to peers
  - Retrieves remote replicas from peers
- Command-line Client
  - Get files from remote storage
  - Put files into local storage



Replica Index Server Throughput Benchmark



- Set up RIS on m1.xlarge, issued 1000 *add* operations each from 1-16 clients on m1.medium instances
- RIS achieved a peak throughput of ~650 ops/sec

Benchmarked vs. Observed RIS Throughput

Average requests per second observed for a 10-degree Montage workflow

Nodes / Cores	Entries in RIS	Workflow runtime (sec)	Average put requests/second
2 / 8	63558	6699	9.5
4 / 16	76688	4705	16.3
8 / 32	N/A	3690	N/A
16 / 64	87073	3704	23.5

- Ran 10 degree workflow using 8-64 cores (m1.xlarge)
- Observed RIS throughput (10-25 ops/sec) is much less than benchmarked throughput (650 ops/sec)
- RIS should not be the bottleneck for workflows and resource pools of this size

Cache Daemon Benchmarks

Put Latency (sec)

Implementation	0 MB	1 MB	10 MB	100 MB
copy	0.007	0.009	0.35	4.36
symlink	0.008	0.007	0.008	0.008

Get Latency (sec)

Implementation	0 MB	1 MB	10 MB	100 MB
copy	0.016	0.031	0.178	3.951
symlink	0.017	0.033	0.146	1.841
symlink+fsync	0.017	0.073	0.373	3.182

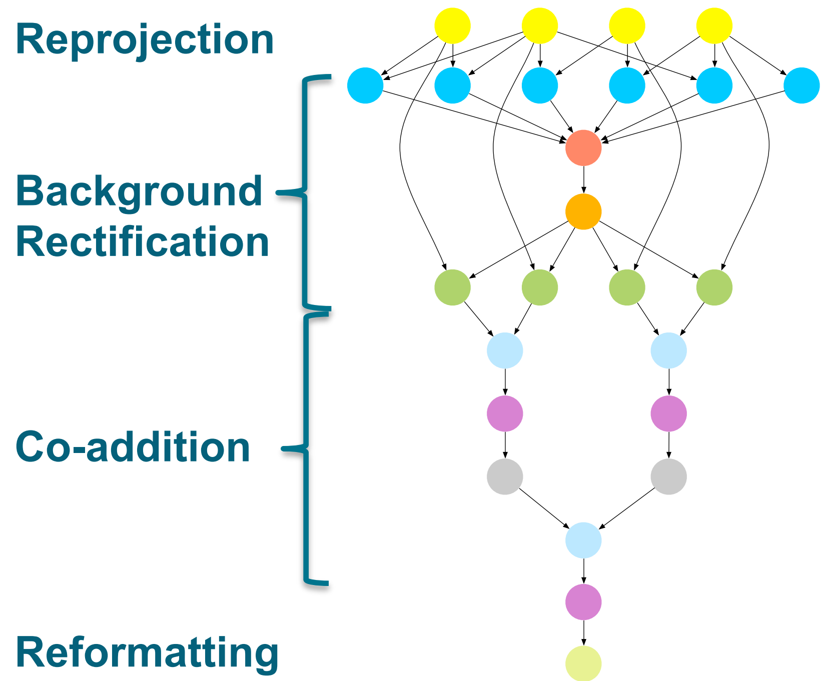
Get Bandwidth (MB/sec)

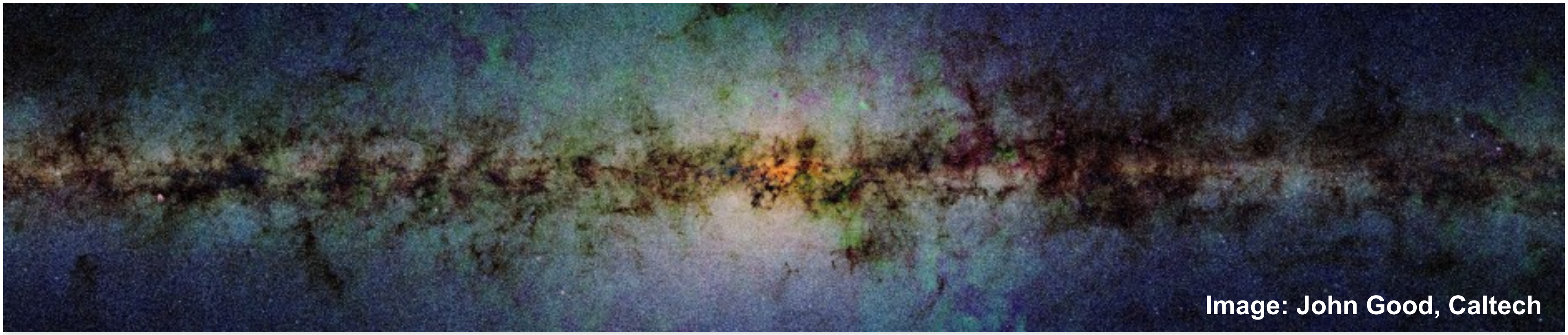
Implementation	1 MB	10 MB	100 MB
copy	31.784	56.048	25.31
symlink	30.571	68.734	54.329
symlink+fsync	13.776	26.824	31.423

- Disk performance: ~38 MB/s write, ~109 MB/s read
- Network performance: ~89 MB/s
- Bottom line: Latency limits performance for smaller files

Workflow Performance Comparison

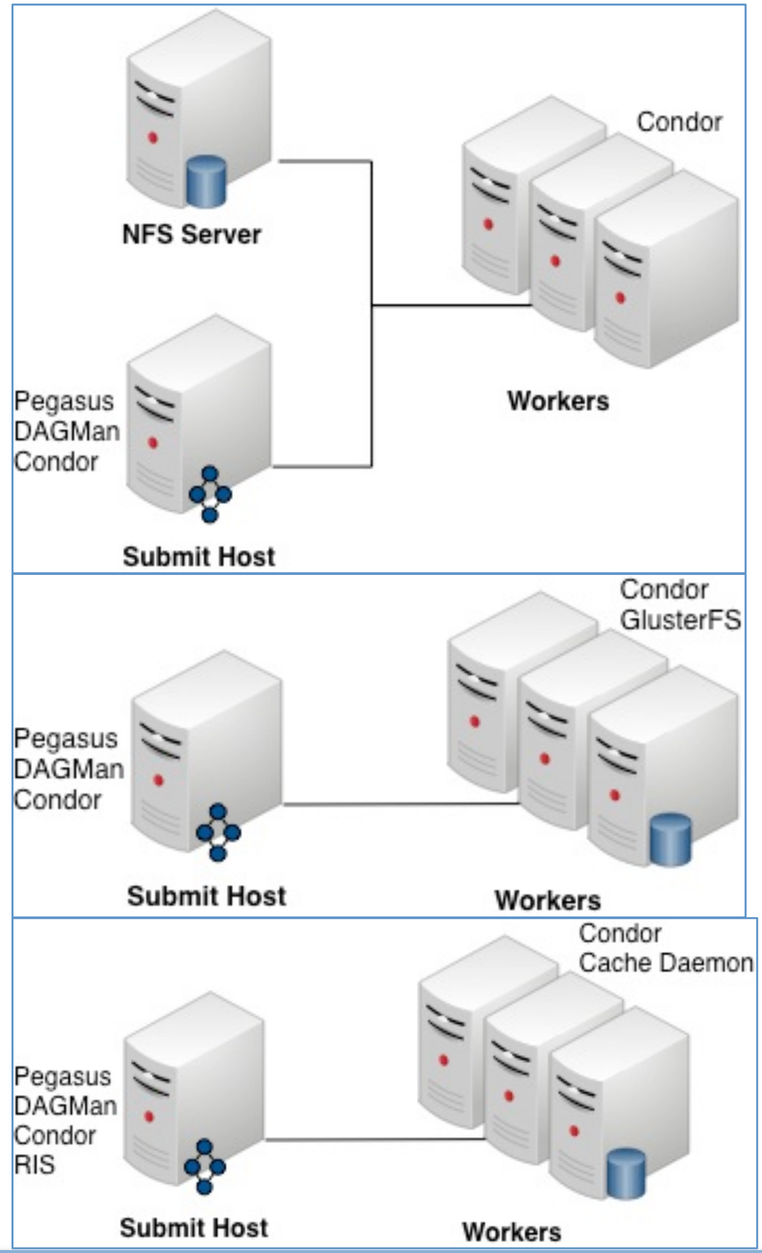
- **Application: Montage**
  - Creates science-grade astronomical image mosaics
- **Test workflow**
  - 10 degree square area
  - 19,320 tasks
  - 13 GB input, 88 GB output



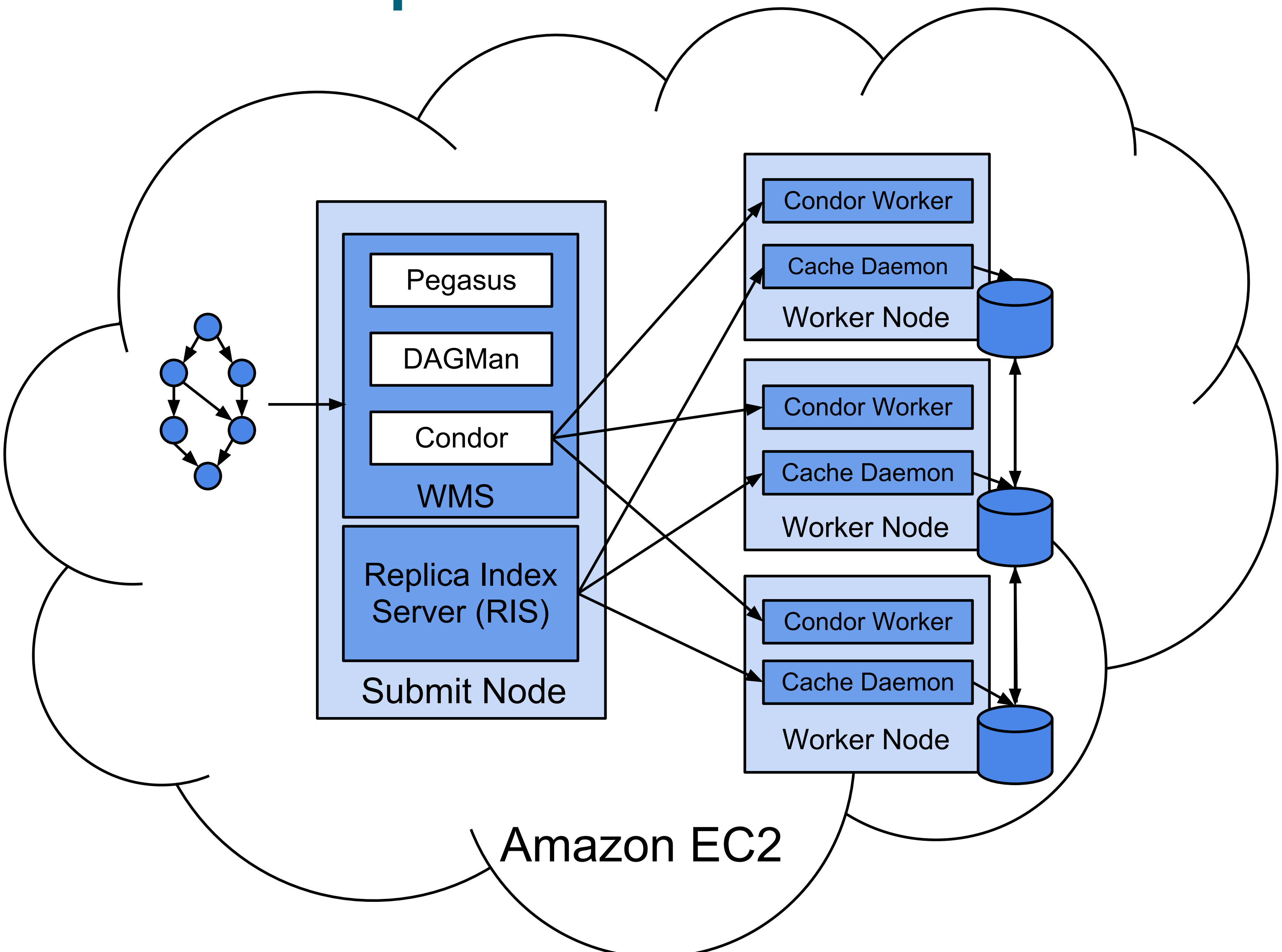


Storage Systems

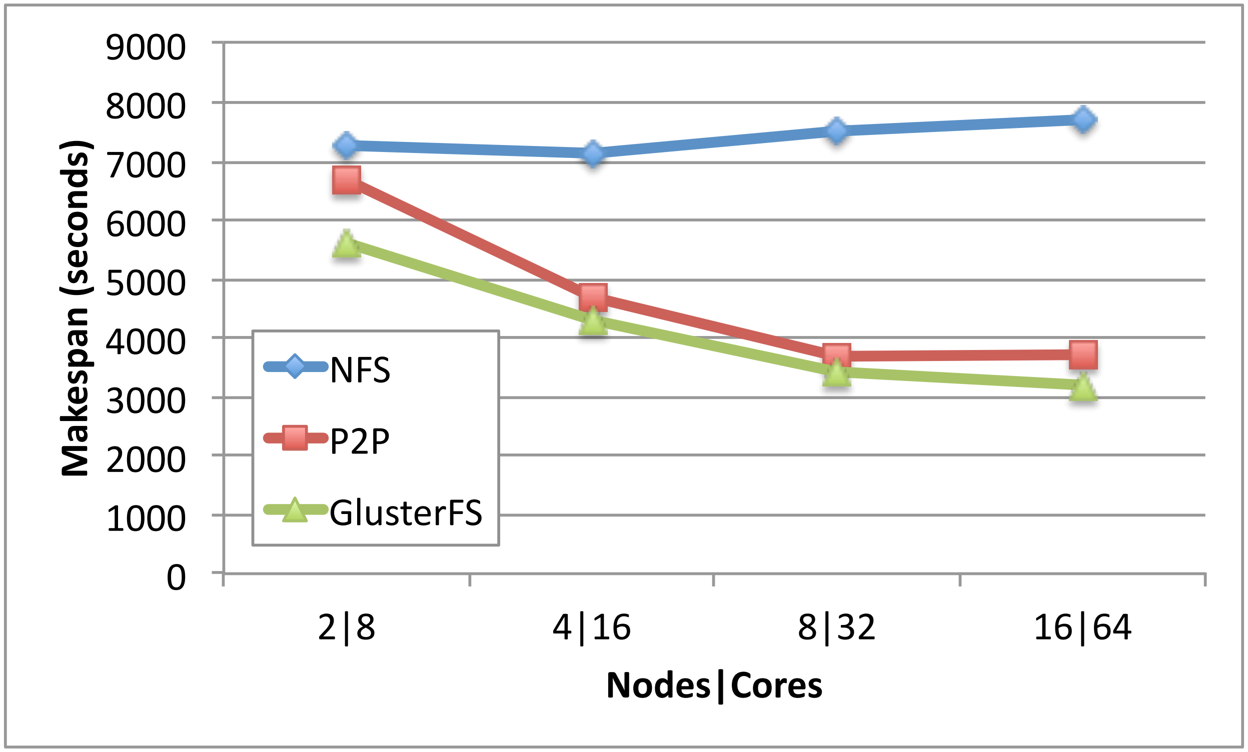
- NFS
  - Centralized file system
  - Used a dedicated m1.xlarge instance
- GlusterFS
  - Distributed file system
  - Used “distribute” mode
  - Each worker participates in the file system
- P2P
  - Our approach
  - RIS co-located with submit host



Experiment Setup



Performance Comparison



- **NFS performance is flat, as expected**
- **Performance flattens out due to workflow structure**
- **GlusterFS performs 13-16% better than P2P**

Discussion

- Bottlenecks
  - Main problem with NFS
  - GlusterFS has no central server
  - P2P RIS is not a bottleneck based on benchmarks
- Latency
  - P2P query overhead harms small file performance
  - Not an issue for GlusterFS (just a hash to find the host)
- Load Balancing
  - P2P does not try to control data placement
  - GlusterFS distributes data more evenly
- Small reads
  - P2P always fetches the entire file
  - GlusterFS can fetch only the blocks required
  - Can overlap communication and computation

Conclusion

- Our experiment did not work out as we hoped, but produced some valuable results
  - RIS server was not a bottleneck
  - Overheads were significant for small files
- We now have a better understanding of the problem
  - Partial reads may be important for some workflows
  - Locality and load balancing are important
  - Need to consider planning and scheduling data movement

Future Work

- Experiment with more workflows
- Compare with alternative data storage solutions
  - e.g. SRM, IRODS
- Study the I/O patterns of different workflows
  - e.g. partial reads
- Optimize the system, especially latencies
- Investigate techniques for planning data placement
- Make use of data-aware scheduling heuristics