Using BlobSeer for concurrency optimized VM storage

Bogdan Nicolae Alexandra Carpen-Amarie

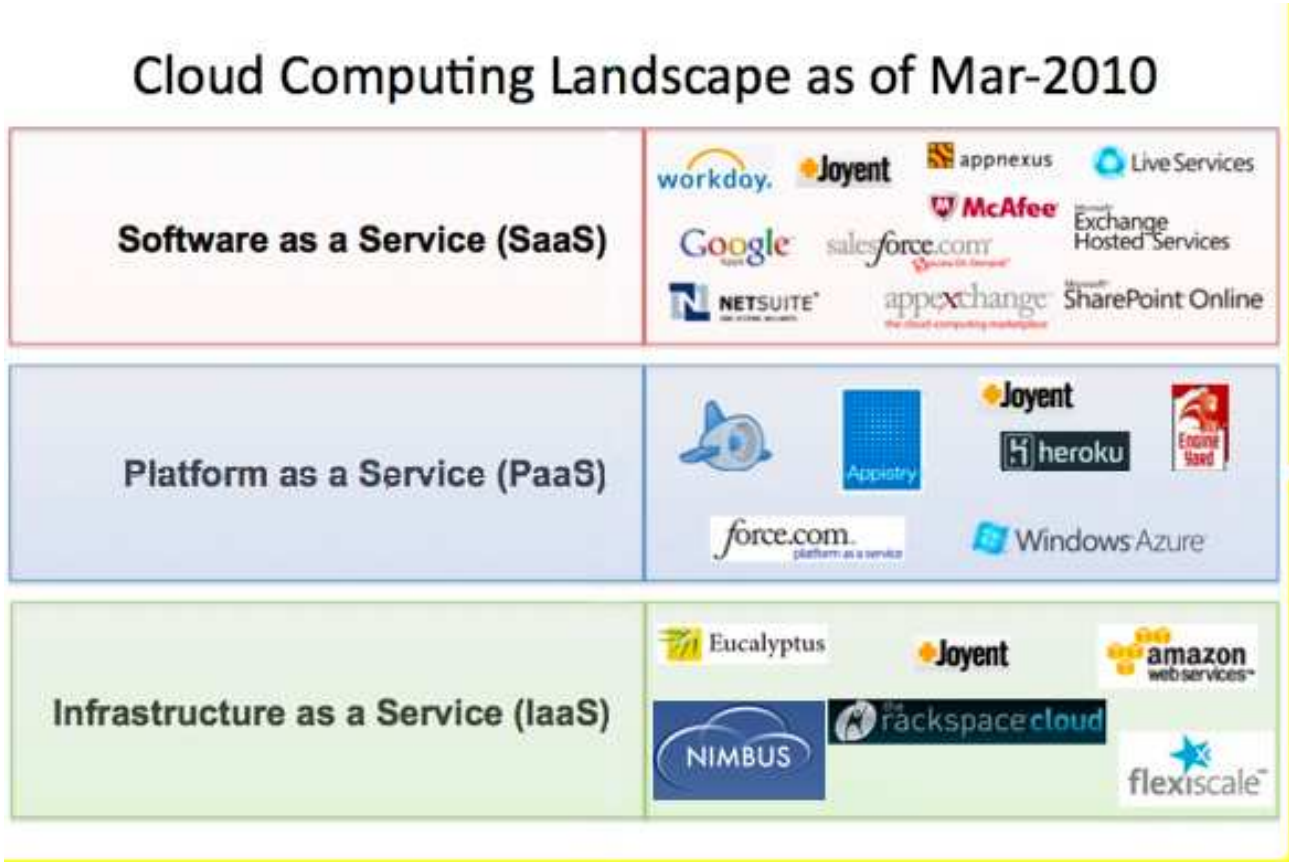
KerData Team, INRIA

Outline

The Cloud Computing landscape

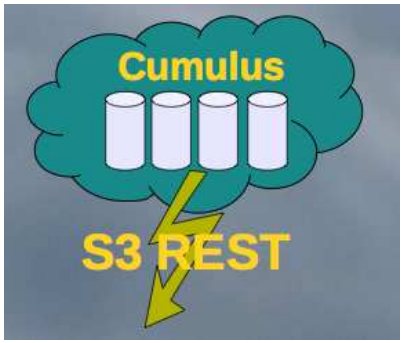
Shared computing and storage resources

- Easily accessible
- Pay-per-use model
- Elastic
- Reliable



Data management in Cloud Computing

- Goal:
  - VM management
  - Application data
- Pay for duration, size and traffic
- High availability
- Flat address space

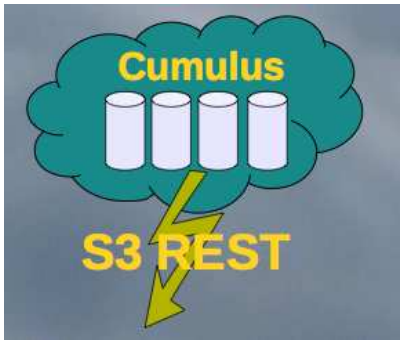


Limitations:

- No support for concurrent accesses
- No fine-grain data access
- Low throughput

Data management in Cloud Computing

- Goal:
  - VM management
  - Application data
- Pay for duration, size and traffic
- High availability
- Flat address space



Limitations:

- No support for concurrent accesses
- No fine-grain data access
- Low throughput

VM management challenges

Typical scenario:

- The user uploads a customized VM image to the Cloud repository.
- The same VM image is deployed simultaneously on a many compute nodes.
- Checkpointing for the running instances to capture application state

Limitations of existing approaches:

- Image propagation delays
- Huge storage space needed
- Important network traffic

VM management challenges

Typical scenario:

- The user uploads a customized VM image to the Cloud repository.
- The same VM image is deployed simultaneously on a many compute nodes.
- Checkpointing for the running instances to capture application state

Limitations of existing approaches:

- Image propagation delays
- Huge storage space needed
- Important network traffic

Our proposal

(Bogdan Nicolae's internship at ANL, advised by K. Keahey and G. Antoniu)

Principles:

- Optimize VM disk access by using on-demand image mirroring
- Reduce contention by striping the image

BlobSeer

- Data striping
- High throughput under concurrency
- Versioning-based concurrency control

Our proposal

(Bogdan Nicolae's internship at ANL, advised by K. Keahey and G. Antoniu)

Principles:

- Optimize VM disk access by using on-demand image mirroring
- Reduce contention by striping the image

BlobSeer

- Data striping
- High throughput under concurrency
- Versioning-based concurrency control

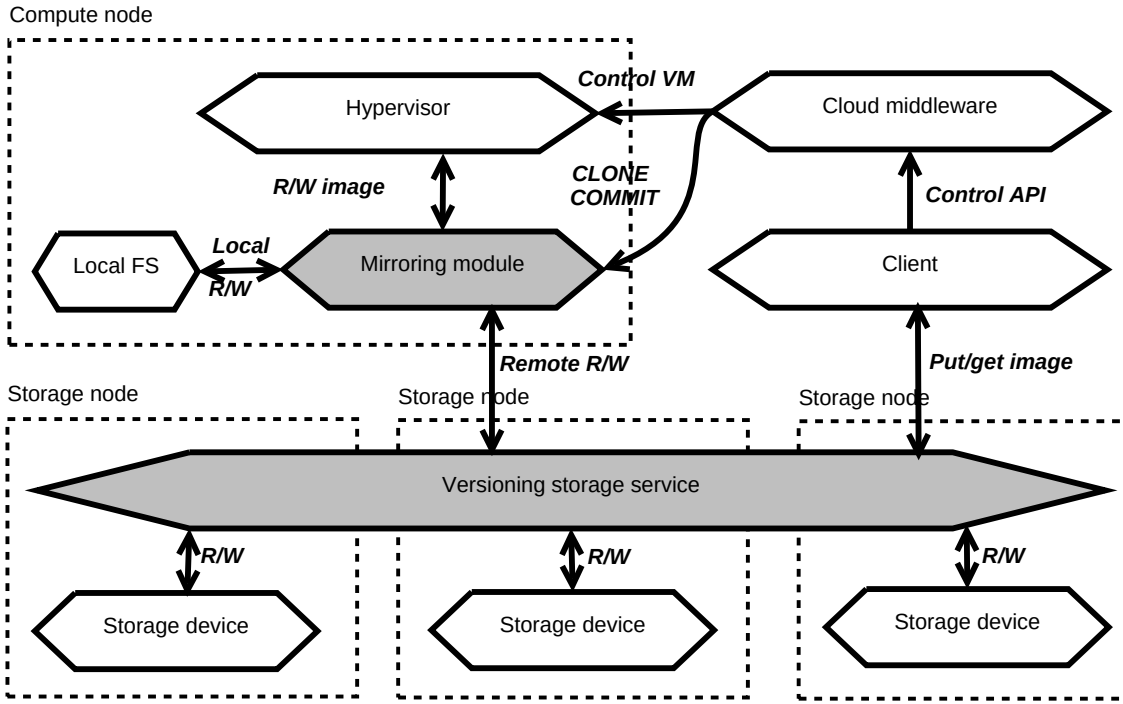
Our proposal

BlobSeer

- Store initial images and snapshots
- Runs on the storage nodes

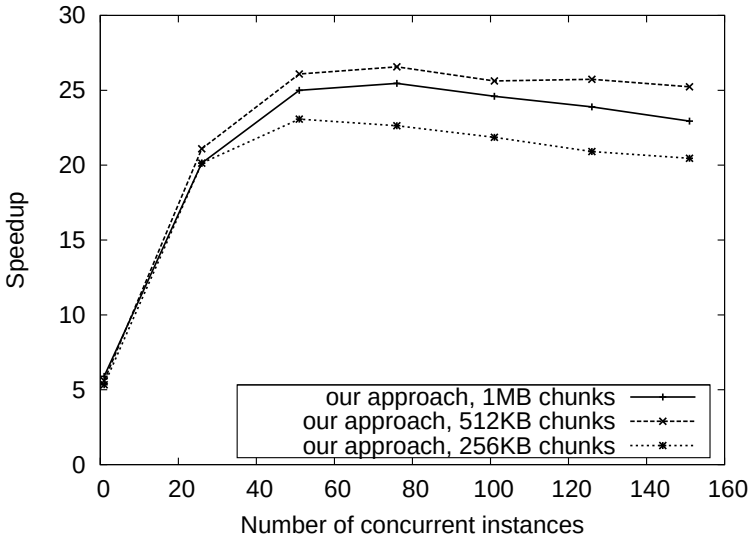
Mirroring module

- Implemented as a FUSE module
- Runs on the compute nodes

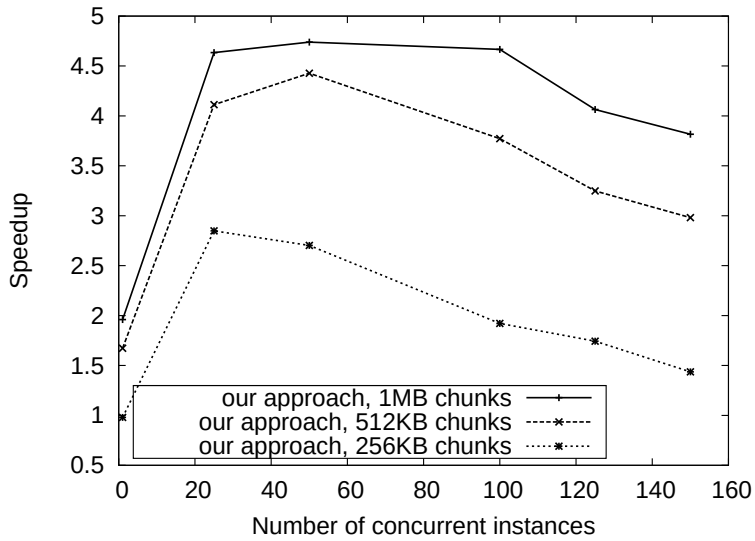


Performance of VM boot process

- Experiments performed on Grid'5000
- 50 storage nodes
- Up to 150 compute nodes



CPU-intensive application



Data-intensive application

Cloud Storage Systems

Open source Cloud repositories:

- GridFTP
- Cumulus

Challenges:

- Support efficient boot/checkpointing of images
- Concurrent uploading of VM images by multiple clients.
- Standard access interfaces for the client

Solution

BlobSeer-based storage back-end

Cloud Storage Systems

Open source Cloud repositories:

- GridFTP
- Cumulus

Challenges:

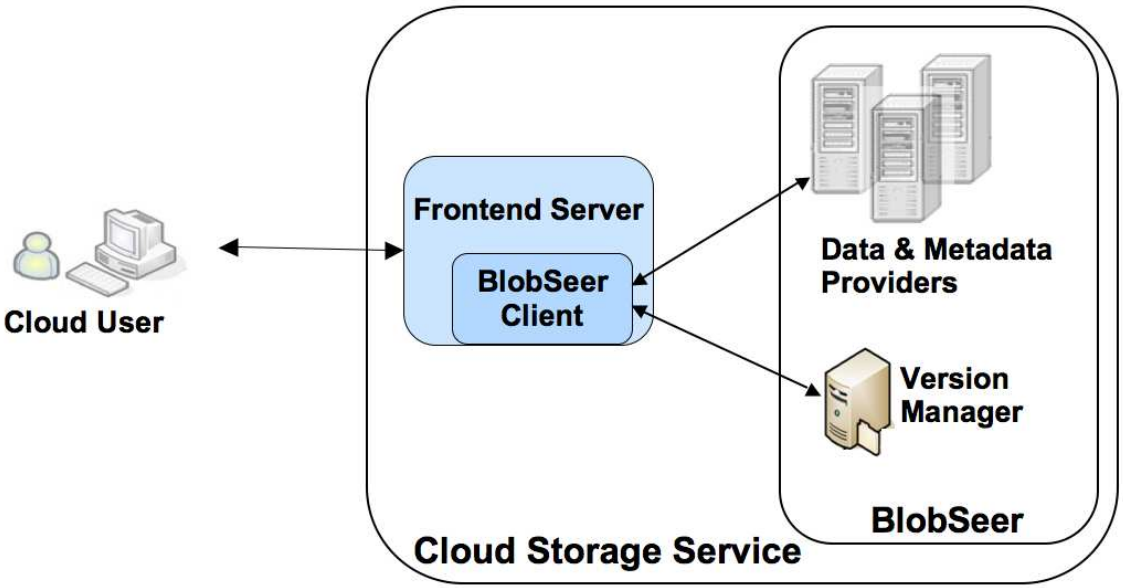
- Support efficient boot/checkpointing of images
- Concurrent uploading of VM images by multiple clients.
- Standard access interfaces for the client

Solution

BlobSeer-based storage back-end

BlobSeer as a storage backend

- High throughput
- Efficient support for concurrency
- Versioning



Design

Repository server acts as a BlobSeer client

- Standard protocol between client and server
- Efficient BlobSeer-specific transfer from the server to the storage nodes.

GridFTP - initial Nimbus data service

(Master internship, advised by G. Antoniu, in collaboration with R. Kettimuthu)

- Widely-spread data transfer protocol
- Implemented within the Globus Toolkit
- High-performance data transfers for data-intensive applications
- Designed to support different storage back ends

Default storage backend

- POSIX-compliant
- Centralized server

BlobSeer-based backend

- Efficient distributed storage
- Hide BlobSeer protocols from the client
- Concurrency support

GridFTP - initial Nimbus data service

(Master internship, advised by G. Antoniu, in collaboration with R. Kettimuthu)

- Widely-spread data transfer protocol
- Implemented within the Globus Toolkit
- High-performance data transfers for data-intensive applications
- Designed to support different storage back ends

Default storage backend

- POSIX-compliant
- Centralized server

BlobSeer-based backend

- Efficient distributed storage
- Hide BlobSeer protocols from the client
- Concurrency support

GridFTP - initial Nimbus data service

(Master internship, advised by G. Antoniu, in collaboration with R. Kettimuthu)

- Widely-spread data transfer protocol
- Implemented within the Globus Toolkit
- High-performance data transfers for data-intensive applications
- Designed to support different storage back ends

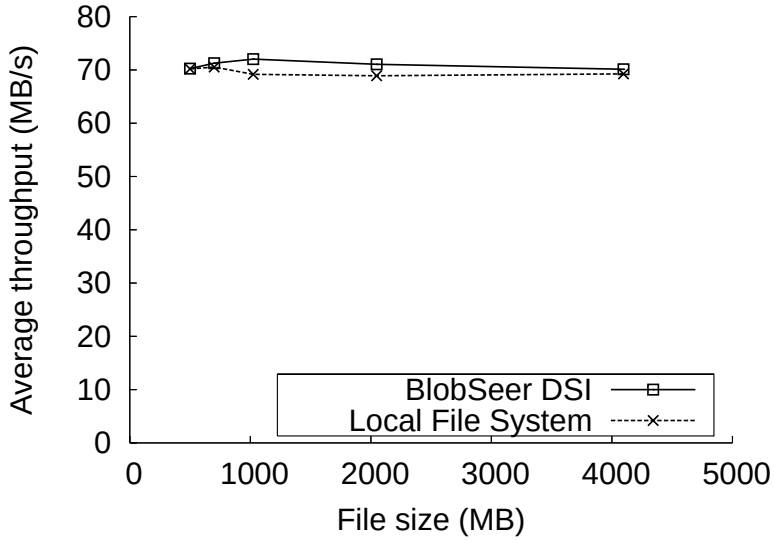
Default storage backend

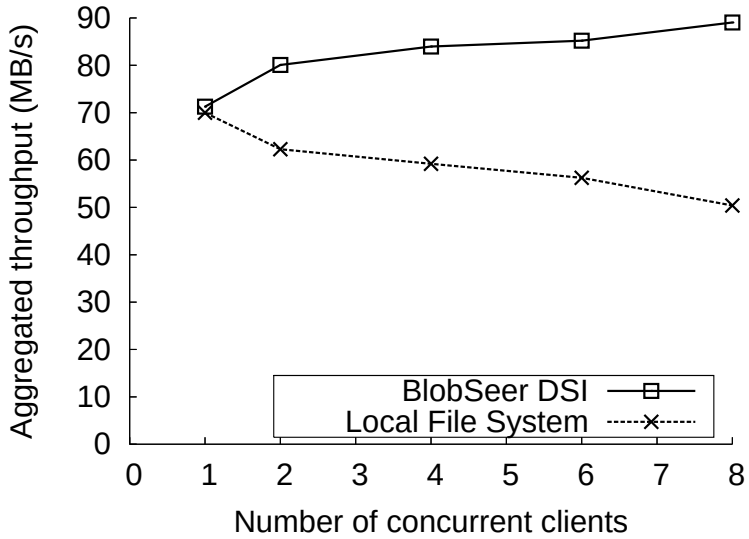
- POSIX-compliant
- Centralized server

BlobSeer-based backend

- Efficient distributed storage
- Hide BlobSeer protocols from the client
- Concurrency support

Evaluation





Single client

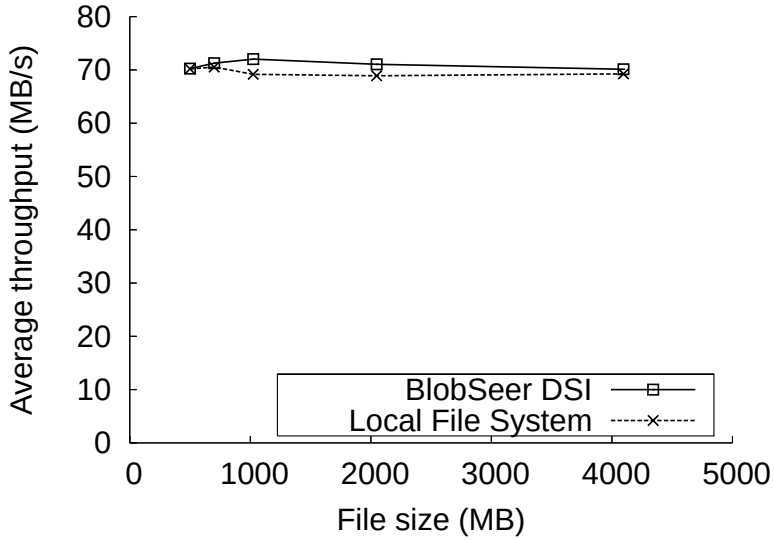
Multiple clients

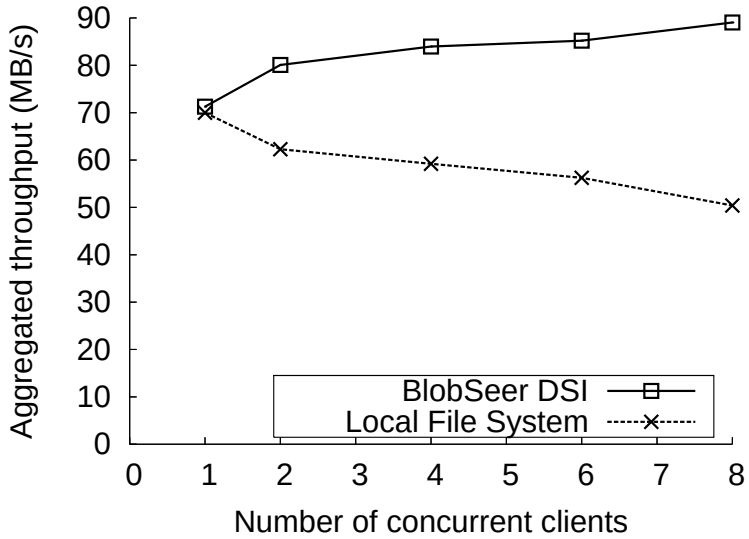
Bottleneck at the level of the GridFTP server.

Client data access limitations

Need for another approach to leverage BlobSeer's features.

Evaluation





Single client

Multiple clients

Bottleneck at the level of the GridFTP server.

Client data access limitations

Need for another approach to leverage BlobSeer's features.

Cumulus - current Nimbus data service

- Open source implementation of the Amazon S3 REST API
- Scalable and reliable access to scientific data
- Customizable backend storage system

Default storage backend

- POSIX-compliant
- Designed for VM management

BlobSeer-based backend

- Concurrency support
- Enable efficient VM management
- Improved scalability through multiple servers

Cumulus - current Nimbus data service

- Open source implementation of the Amazon S3 REST API
- Scalable and reliable access to scientific data
- Customizable backend storage system

Default storage backend

- POSIX-compliant
- Designed for VM management

BlobSeer-based backend

- Concurrency support
- Enable efficient VM management
- Improved scalability through multiple servers

Cumulus - current Nimbus data service

- Open source implementation of the Amazon S3 REST API
- Scalable and reliable access to scientific data
- Customizable backend storage system

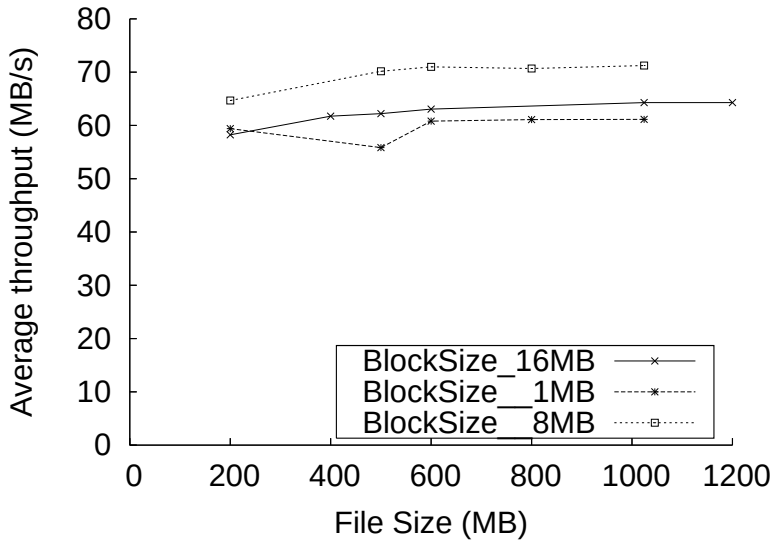
Default storage backend

- POSIX-compliant
- Designed for VM management

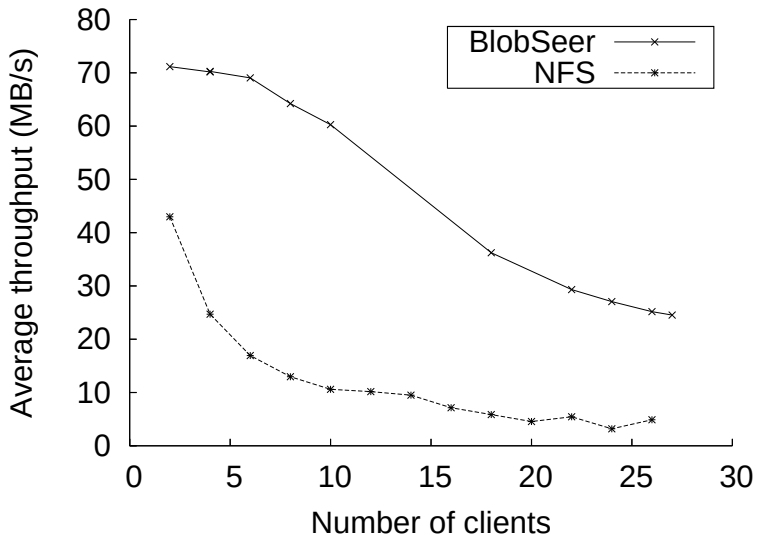
BlobSeer-based backend

- Concurrency support
- Enable efficient VM management
- Improved scalability through multiple servers

Evaluation



Single client



Replicated Cumulus Servers

Obtained performance similar to previous Cumulus evaluations (Cumulus Poster presented at SC10).

Summary

- VM management
  - Lazy VM-deployment scheme based on BlobSeer
    - Efficient multi-deployment
    - Efficient multi-snapshotting
- Cloud storage systems
  - Integrated BlobSeer with the existing Nimbus storage services
  - Standardized interfaces to handle BlobSeer data

Future work

- VM management
  - More experiments
    - LAN vs. WAN
    - Replication
  - Access pattern prediction
- Cumulus cloud storage
  - Expose versioning in the Cumulus interface
  - Optimize BlobSeer usage
  - Evaluate scalability
- Evaluate cloud storage for scientific applications
  - Scientific workflows
  - Store generated data into the cloud
    - BlobSeer interface
    - Standard interfaces

Q&A