AnalyzeThis: An Analysis Workflow-Aware Storage System

Hyogi Sim, Youngjae Kim, Sudharshan S. Vazhkudai, Devesh Tiwari,
Ali Anwar, Ali R. Butt, Lavanya Ramakrishnan

Virginia Tech, Oak Ridge National Laboratory, Lawrence Berkeley Laboratory

Motivation

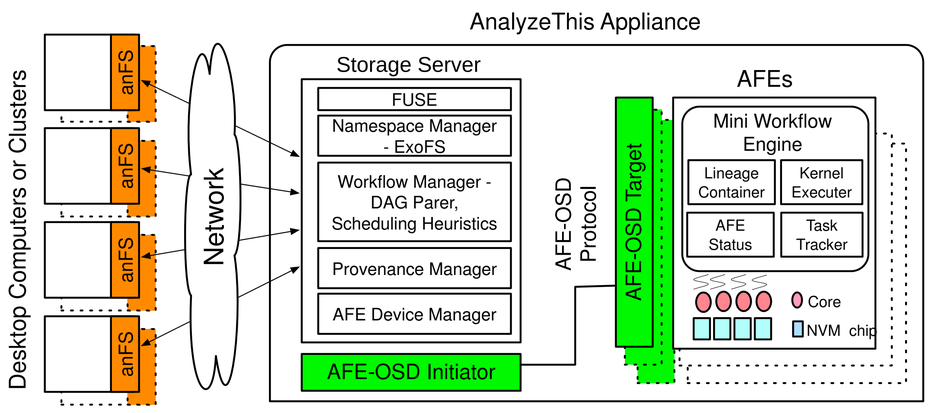
- *Storage Wall* on large-scale systems
  - Titan: 27 PFLOPs and 1 TB/s on 18,688 nodes, 56 MB/s per a node
  - Raw simulation data is read repeatedely for multiple analysis jobs
- Enabling trend, *Active Flash*
  - Running computations using SSD controllers
  - Oasis (MSST'11), iSSD (ICS'13), Active Flash (MSST'12, FAST'13)

AnalyzeThis: Goals

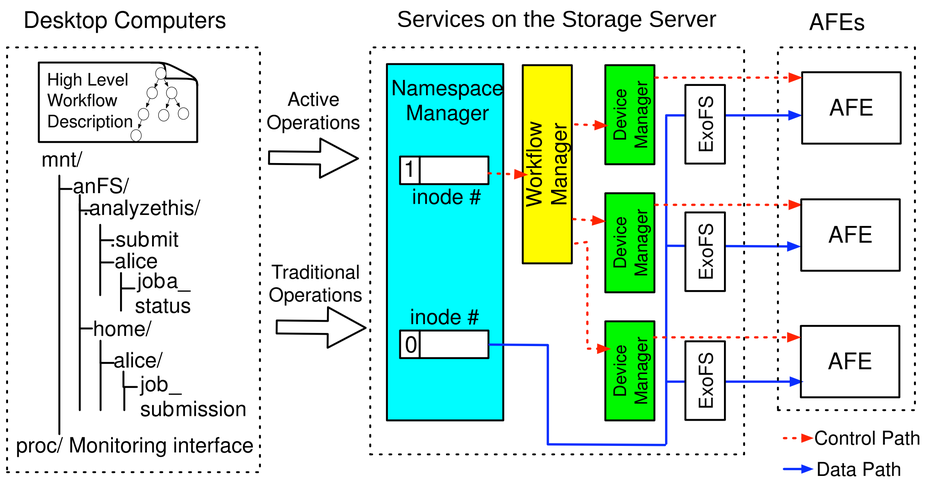
- Analysis awareness: introduce analysis-aware semantics into the storage system, where data already resides
- Reduce data movement: minimize inter- and intra-data movement
- Capture lineage: track provenance and intermediate data products for future references
- Easy-to-use file system interface: provide easy-to-use and familiar interface for submitting and monitoring jobs

AnalyzeThis: Design and Implementation

- Analysis data object abstraction: integrally tie together the dataset, analysis sequence and lineage
  - Powerful construct that is much more than just a pointer to a byte stream
  - Realized using SCSI T10 OSD2 standard
  - Analysis object overlaid on Active Flash refers to an Active Flash Element (AFE)
- Integrate workflow scheduling into the storage system
  - Mimic how user interacts with batch computing systems and integrate similar semantics into the storage system
  - Scheduling heuristics across AFEs: RR (Round Robin), IL (Input Locality), MW (Minimum wait), and HY (hybrid)
- `anFS` file system interface
  - A FUSE filesystem atop an array of AFEs
  - `/proc`-like virutal interface for users to submit and track jobs
- AFEs and `anFS` together track provenance

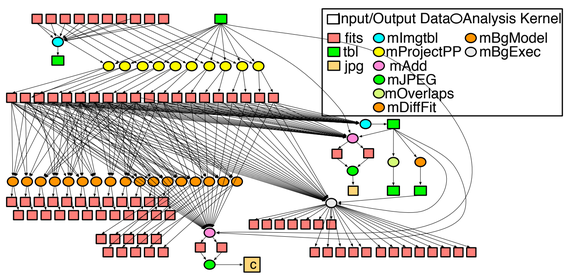


(a) *AnalyzeThis* Architecture

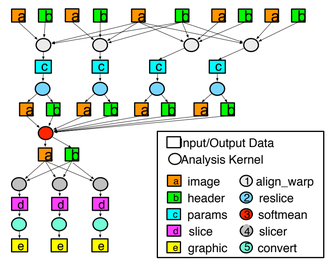


(b) anFS Architecture and Data and Control Paths

AnalyzeThis: Usage



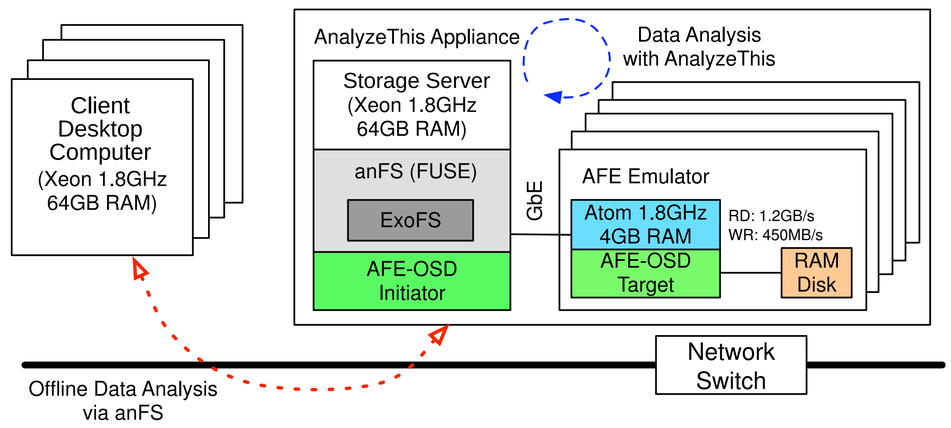
(a) Montage Workflow



(b) Brain Atlas Workflow

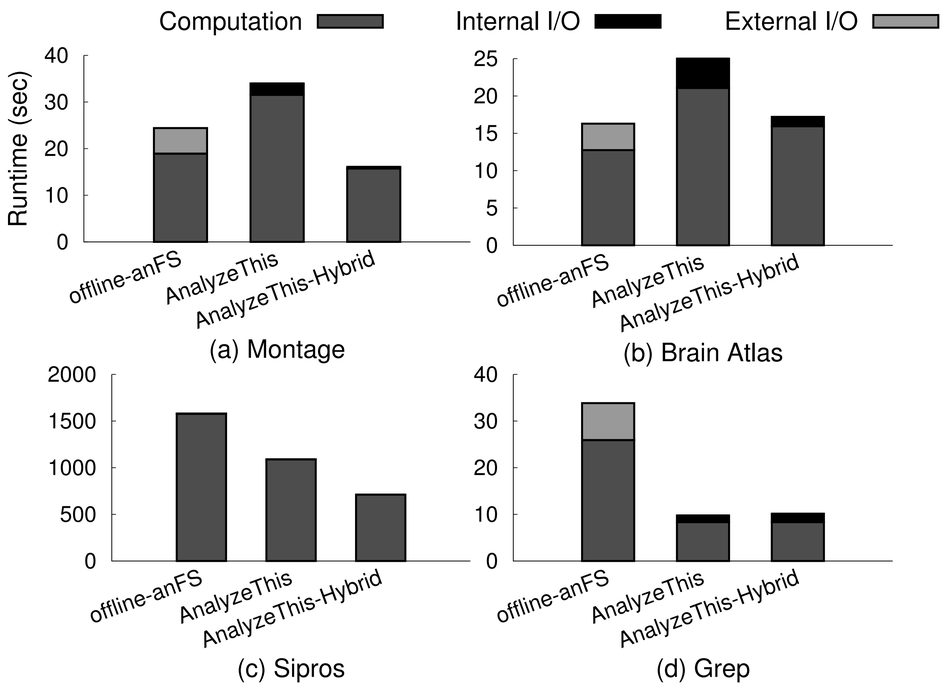
```
$ echo /mnt/anFS/me/job1 > /mnt/anFS/.analyzethis/.submit
$ cat /mnt/anFS/.analyzethis/me/job1/status ...
```

Experimental Setup



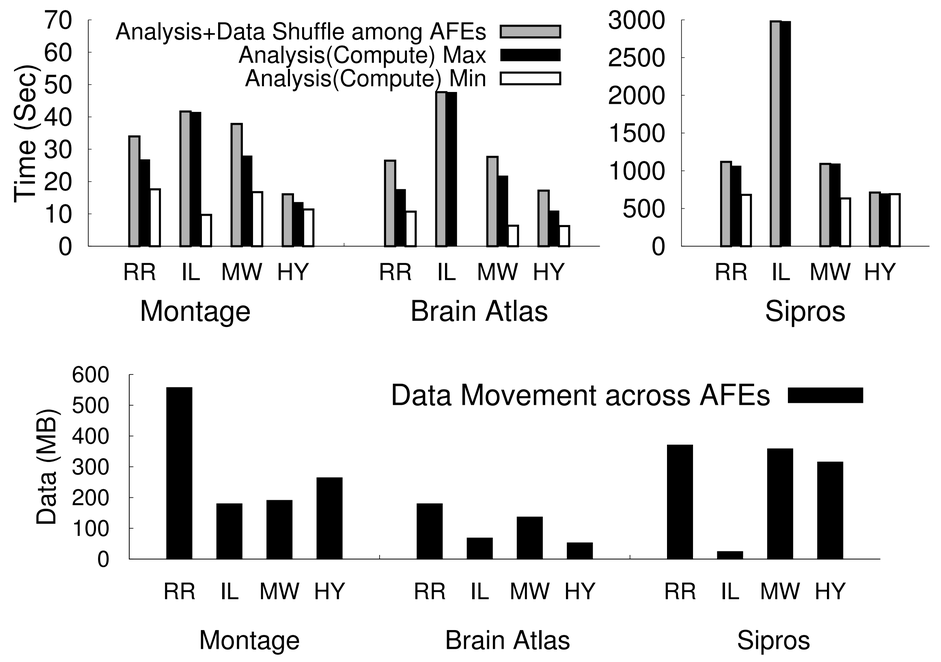
- AFEs are emulated with Atom-based machines
- Scientific workflows: Montage, Brain Atlas, Sipros, and Grep

Results: AnalyzeThis Performance



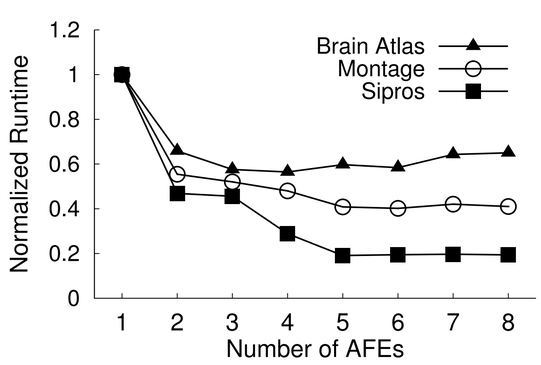
- AnalyzeThis effectively reduces the external data movement
- AnalyzeThis excels for jobs with high parallelism and data intensity

Results: Scheduling Heuristics



- MW/HY perform best reconciling AFE utilization and data movement cost

Results: AFE Scaling



- Different speedup curves with the increase in AFEs
  - Degree of parallelism in a workflow
  - Data dependencies among tasks
  - Input and output data size of each task

Acknowledgements

This research was supported in part by the U.S. DOE's Scientific data management program. The work was also supported by, and used the resources of, the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).

hyogi@vt.edu, vazhkudaiss@ornl.gov