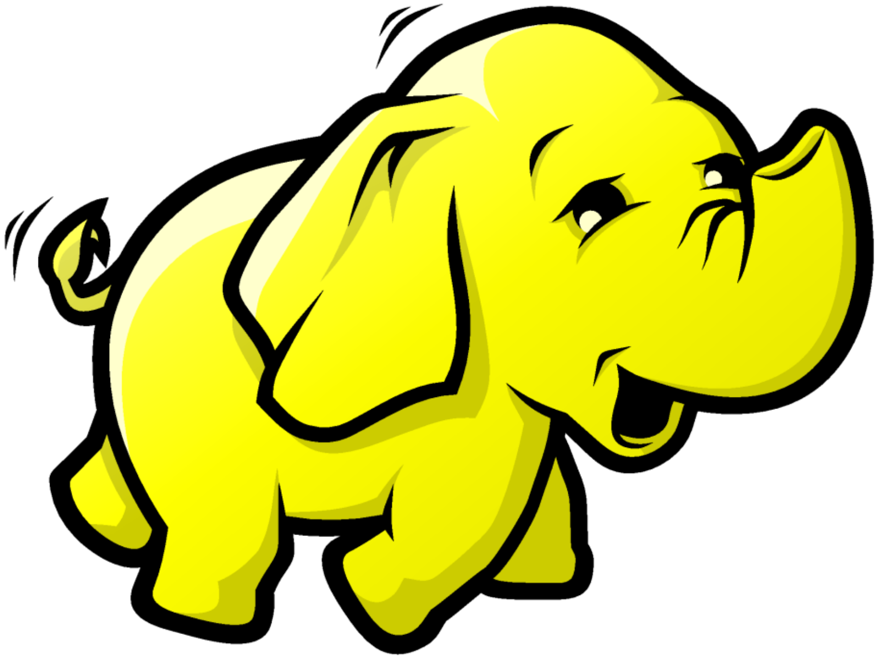
Data Analytics and Storage System (DASS) – Mixing POSIX and Hadoop Architectures

13 November 2016

Carrie Spear (carrie.e.spear@nasa.gov)
HPC Architect/Contractor at the
NASA Center for Climate Simulation (NCCS)



DASS Concept

Read access from all nodes within the ADAPT system
- Serve to data portal services
- Serve data to virtual machines for additional processing
- Mixing model and observations

ADAPT

Climate Analytics as a Service

Analytics through web services or higher level APIs are executed and passed down into the centralized storage environment for processing; answers are returned. Only those analytics that we have written are exposed.

Request goes into the storage.

Answer is returned.

HyperWall

Read access from the HyperWall to facilitate visualizing model outputs quickly after they have been created.

Data Analytics and Storage System (DASS) ~10 PB

Mass Storage

Read and write access from the mass storage
- Stage data into and out of the centralized storage environment as needed

HPC - Discover

Write and Read from all nodes within Discover – models write data into GPFS which is then staged into the centralized storage (burst buffer like). Initial data sets could include:
- Nature Run
- Downscaling Results
- Reanalysis (MERRA, MERRA2)
- High Resolution Reanalysis

Note that more than likely all the services will still have local file systems to enable local writes within their respective security domain.

Data Analytics Storage System (DASS)

Data movement and sharing of data across services within the NCCS is still a challenge

Large data sets created on Discover (HPC)

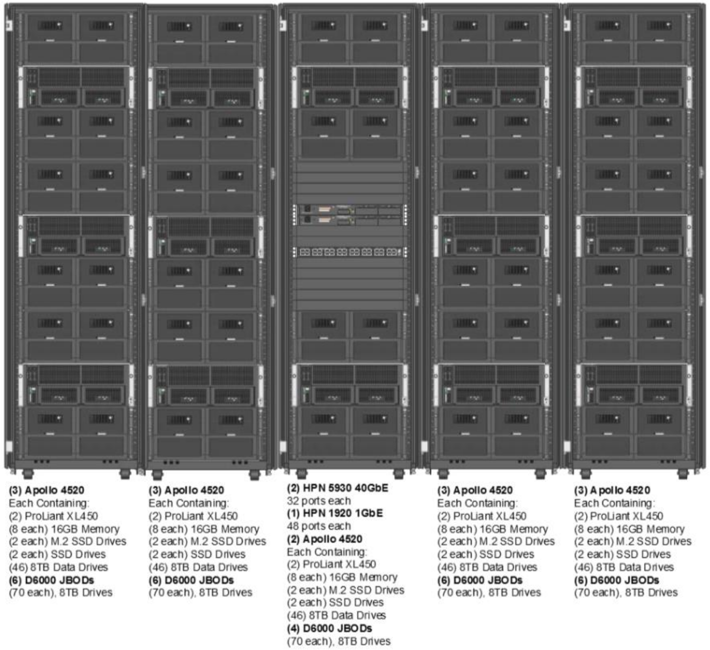
- On which users perform many analyses
- And may not be in a NASA Distributed Active Archive Center (DAAC)

Create a true centralized combination of storage and compute capability

- Capacity to store many PBs of data for long periods of time
- Architected to be able to scale both horizontally (compute and bandwidth) and vertically (storage capacity)
- Can easily share data to different services within the NCCS
- Free up high speed disk capacity within Discover
- Enable both traditional and emerging analytics
- No need to modify data; use native scientific formats

Initial DASS Capability Overview

- Initial Capacity
  - 20.832 PB Raw Data Storage
  - 2,604 by 8TB SAS Drives
  - 14 Units
  - 28 Servers
  - 896 Cores
  - 14,336 GB Memory
  - 16 GB/Core
  - 37 TF of compute
- Roughly equivalent to the compute capacity of the NCCS just 6 years ago!
- Designed to easily scale both horizontally (compute) and vertically (storage)



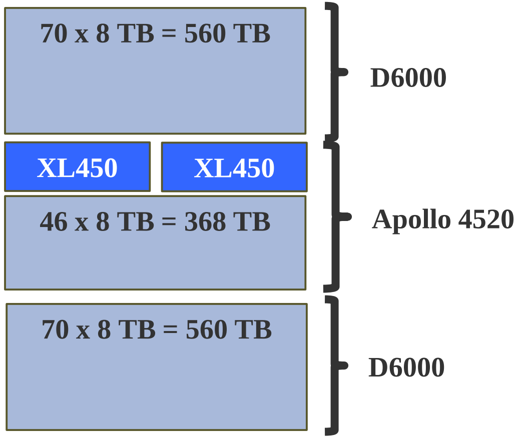
DASS Compute/Storage Units

HPE Apollo 4520 (Initial quantity of 14)

- Two (2) Proliant XL450 servers, each with
- Two (2) 16-core Intel Haswel E5-2697Av4 2.6 GHz processors
- 256 GB of RAM
- Two (2) SSD's for the operating system
- Two (2) SSD's for metadata
- One (1) smart array P841/4G controller
- One (1) HBA
- One (1) Infiniband FDR/40 GbE 2-port adapter
- Redundant power supplies
- 46 x 8 TB SAS drives

Two (2) D6000 JBOD Shelves for each Apollo 4520

- 70 x 8TB SAS drives



DASS Compute/Storage Units

Traditional

Data moved from storage to compute.

Open, Read, Write, MPI, C-code, Python, etc.

POSIX Interface

Infiniband, Ethernet

Shared Parallel File System (GPFS)

MapReduce, Spark, Machine Learning, etc.

RESTful Interface, Custom APIs, Notebooks

Cloudera and SIA

Shared Parallel File System (GPFS) Hadoop Connector

Native Scientific Data stored in HPC Storage or Commodity Servers and Storage

Emerging

Analytics moved from servers to storage.

Open Source Software Stack on DASS Servers

- Centos Operating System
- Software RAID
- Linux Storage Enclosure Services
- Pacemaker
- Corasync

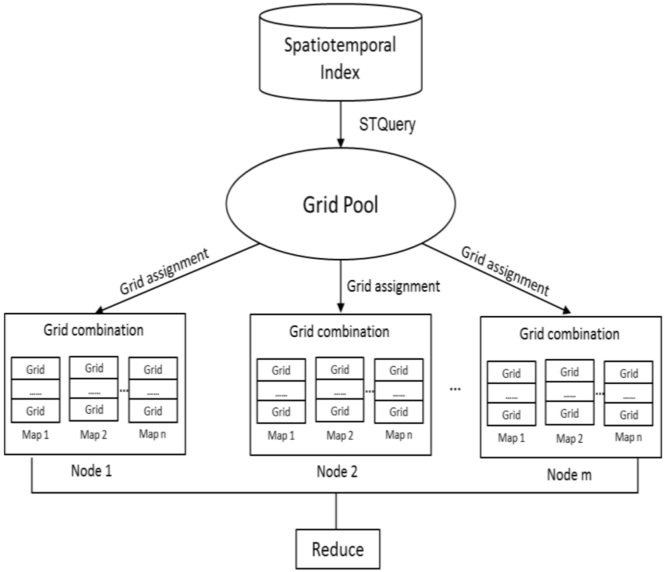
Spatiotemporal Index Approach (SIA) and Hadoop

Use what we know about the structured scientific data

Create a spatiotemporal query model to connect the array-based data model with the key-value based MapReduce programming model using grid concept

Built a spatiotemporal index to

- Link the logical to physical location of the data
- Make use of an array-based data model within HDFS
- Developed a grid partition strategy to
- Keep high data locality for each map task
- Balance the workload across cluster nodes



A spatiotemporal indexing approach for efficient processing of big array-based climate data with MapReduce
Zhenlong Lia, Fei Hua, John L. Schnase, Daniel Q. Duffy, Tsengdar Lee, Michael K. Bowen and Chaowei Yang
International Journal of Geographical Information Science, 2016
http://dx.doi.org/10.1080/13658816.2015.1131830

Analytics Infrastructure Testbed

Test Cluster 1
SIA
Cloudera
HDFS

- 20 nodes (compute and storage)
- Cloudera
- HDFS
- Sequenced data
- Native NetCDF data
    - Put only

Test Cluster 2
SIA
Cloudera
Hadoop Connector
GPFS

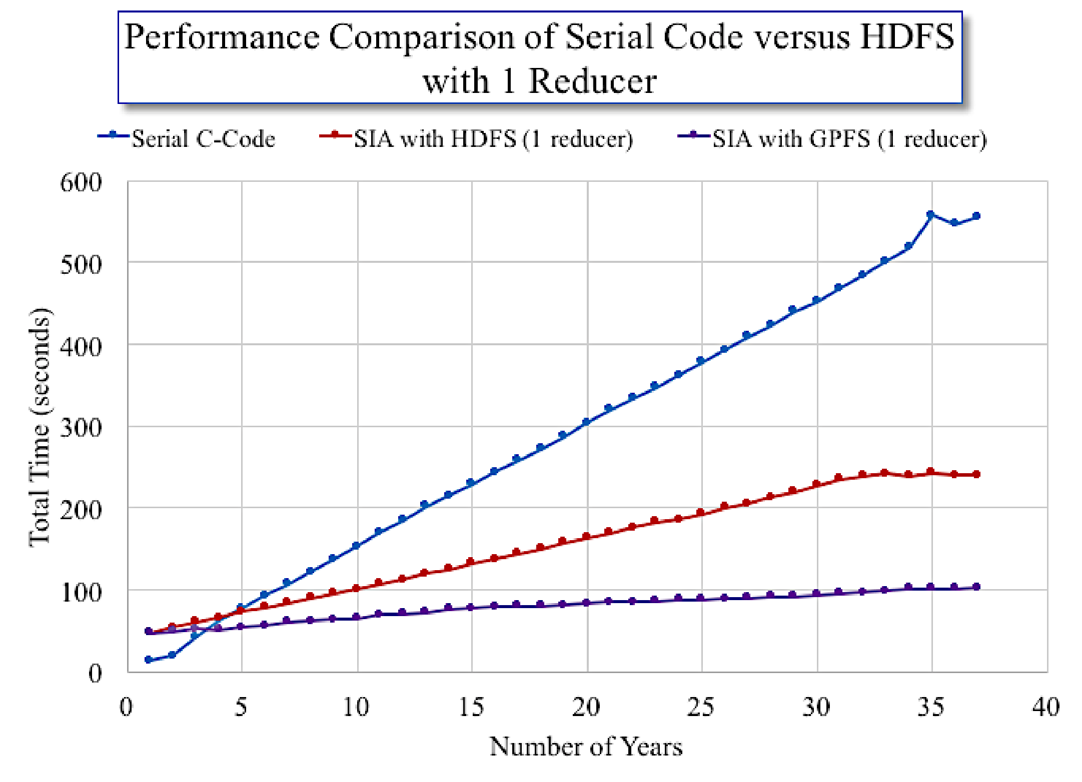
- 20 nodes (compute and storage)
- Cloudera
- GPFS
- *Spectrum Scale Hadoop Transparency Connector*
- Sequenced data
    - Put and Copy
- Native NetCDF Data
    - Put and Copy

Test Cluster 3
SIA
Cloudera
Hadoop Connector
Lustre

- 20 nodes (compute and storage)
- Cloudera
- Lustre
- *Lustre HAM and HAL*
- Sequenced data
    - Put and Copy
- Native NetCDF Data
    - Put and Copy

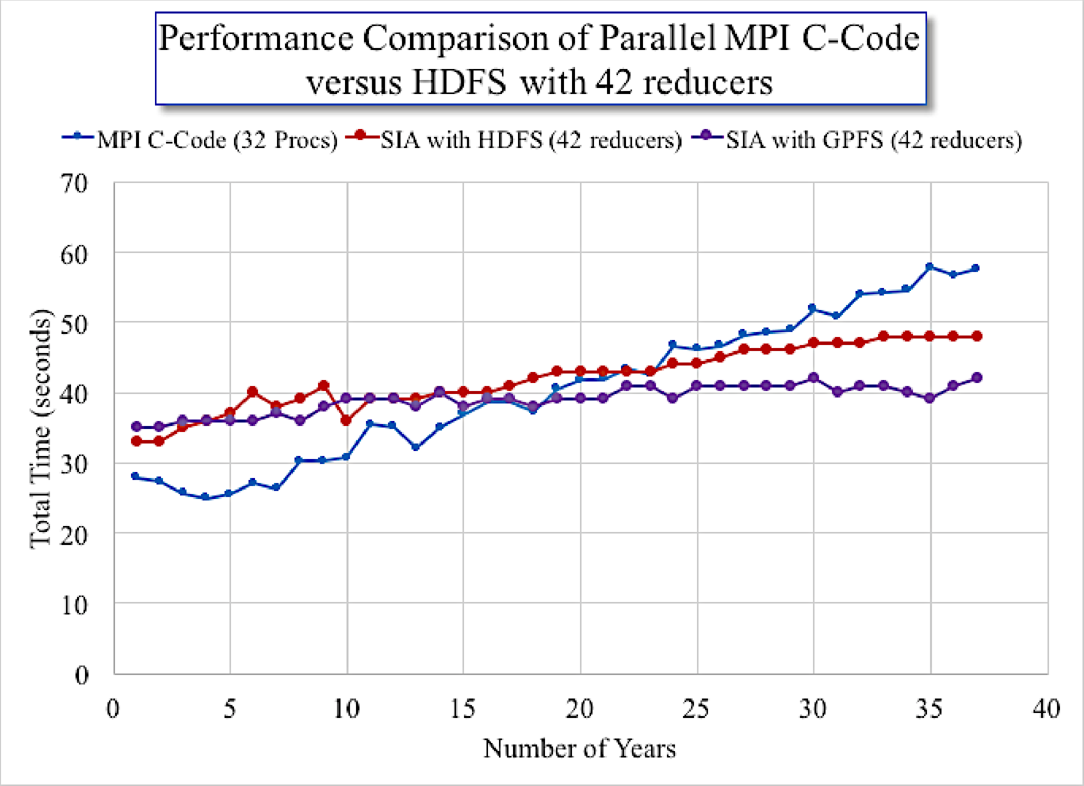
DASS Initial Serial Performance

- Compute the average temperature for every grid point (x, y, and z)
- Vary by the total number of years
- MERRA Monthly Means (Reanalysis)
- Comparison of serial c-code to MapReduce code
- Comparison of traditional HDFS (Hadoop) where data is sequenced (modified) with GPFS where data is native NetCDF (unmodified, copy)
- Using unmodified data in GPFS with MapReduce is the fastest
- Only showing GPFS results to compare against HDFS

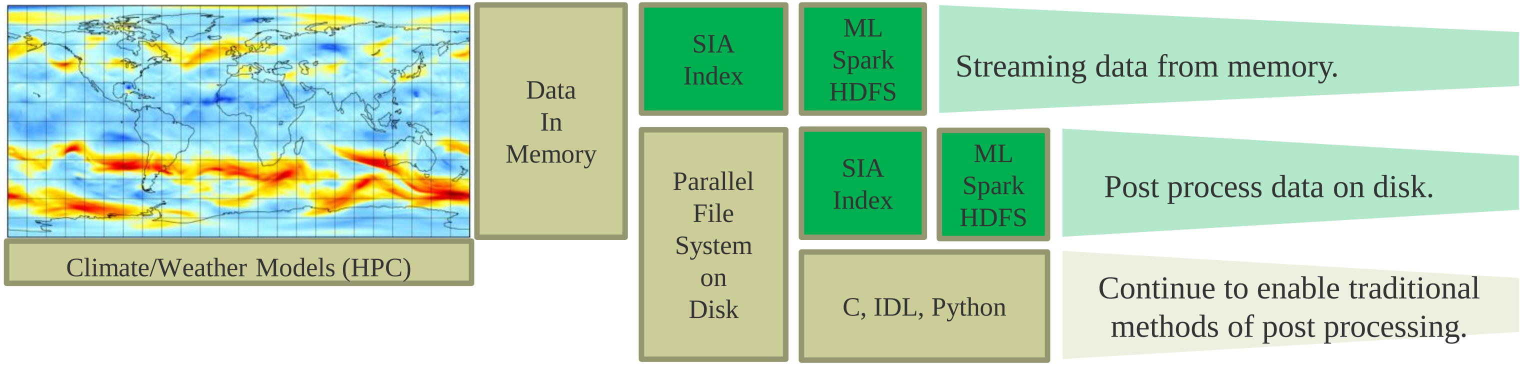


DASS Initial Parallel Performance

- Compute the average temperature for every grid point (x, y, and z)
- Vary by the total number of years
- MERRA Monthly Means (Reanalysis)
- Comparison of serial c-code with MPI to MapReduce code
- Comparison of traditional HDFS (Hadoop) where data is sequenced (modified) with GPFS where data is native NetCDF (unmodified, copy)
- Again using unmodified data in GPFS with MapReduce is the fastest as the number of years increases
- Only showing GPFS results to compare against HDFS



Future of Data Analytics



- Future HPC systems must be able to efficiently transform information into knowledge using both traditional analytics and emerging *machine learning* techniques.
- Requires the ability to be able to index data in memory and/or on disk and enable analytics to be performed on the data where it resides – even in memory
- All without having to modify the data