UNIVERSITY OF CENTRAL FLORIDA

Data Semantics Aware Cloud for High Performance Analytics

Microsoft Future Cloud Workshop 2011
June 2nd 2011,
Prof. Jun Wang,
Computer Architecture and Storage System Laboratory (CASS)

DEPARTMENT OF EECS
ELECTRICAL & COMPUTER ENGINEERING DIVISION

Acknowledgement

UNIVERSITY OF CENTRAL FLORIDA

- **Thanks to NSF CCF-0811413 and NSF CAREER CCF-0953946 grants support.**
- **Other project members**
  - Qiangju Xiao
  - Pengju Shang
  - Christopher Mitchell
  - Grant Mackey
  - Junyao Zhang
- **Department Of Energy Los Alamos national laboratory**

DEPARTMENT OF EECS
ELECTRICAL & COMPUTER ENGINEERING DIVISION

Application Trend: more scientists will perform analytics on clouds!

- **More and more scientists from many different disciplines rely on data analytics to advance scientific and engineering discovery**
  - Accelerate dramatically the pace of discovery by removing a range of mundane but timeconsuming tasks from our consciousness (I. Foster Argonne).
- **Scientific grand challenge problems require global-scale interactive data sharing and collaborations**

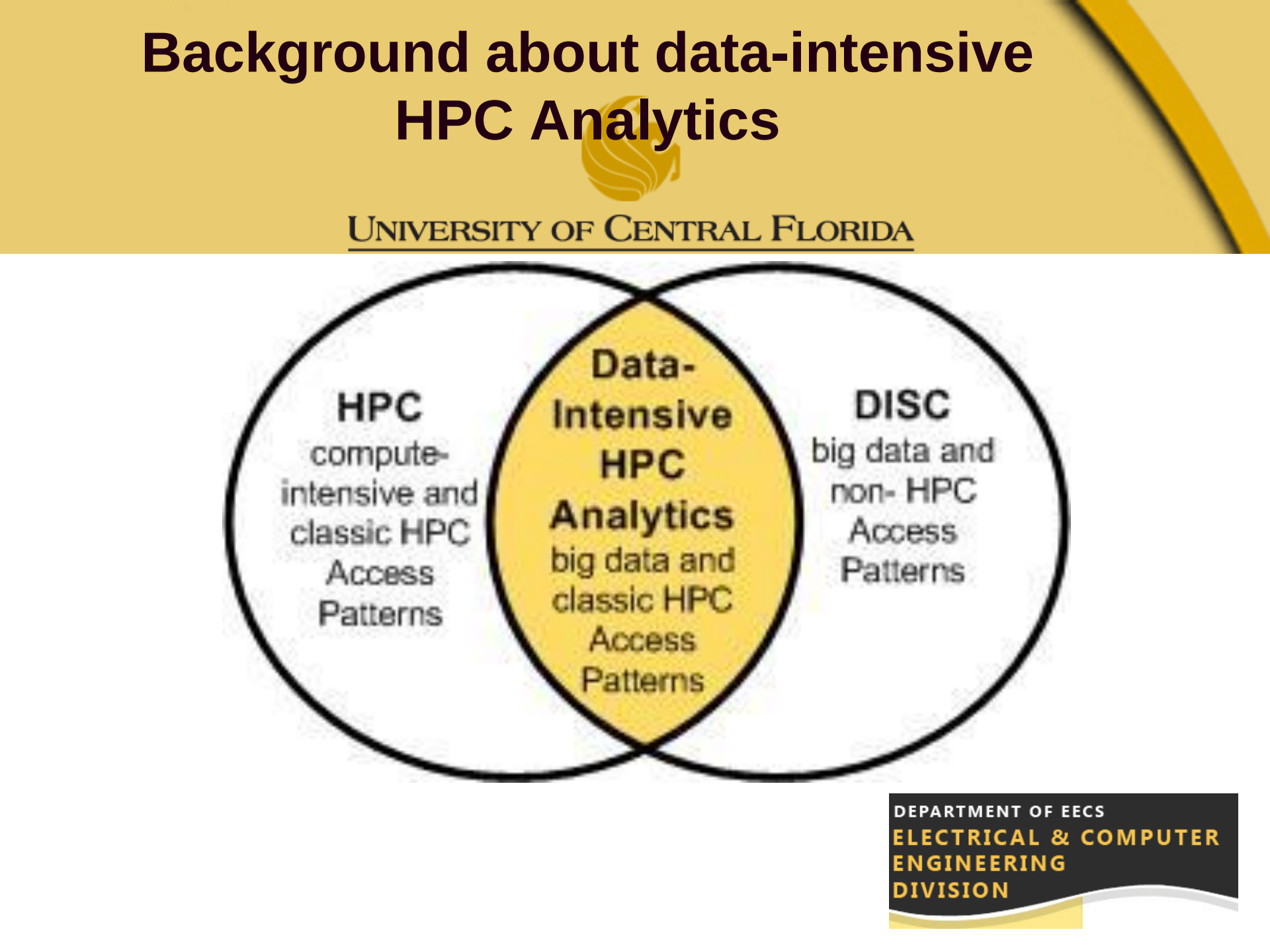
Background about data-intensive HPC Analytics

UNIVERSITY OF CENTRAL FLORIDA

- **Similar to e-science and data intensive computing in that**
  - Manipulate oceans of data
  - Performing dominant file I/O & moderate network I/O
- **Distinct features**
  - HPC data access pattern
    - Strided access/non-contiguous access
    - Small-sized requests
  - Some coordination efforts among many parallel tasks
    - all matching

DEPARTMENT OF EECS
ELECTRICAL & COMPUTER ENGINEERING DIVISION

Background about data-intensive HPC Analytics



Our Preliminary Works at HPC Analytics

UNIVERSITY OF CENTRAL FLORIDA

- **MRAP (ACM HPDC'10)**
  - Motivation: there exists a semantics gap between simulation output data at source sites and analyzed data at destination sites by analysis programs in HPC analytics
  - Idea: Passing data semantics information from domain scientist down to the physical file storage level at analyses sites
    - Do a better job of Data Staging: read only needed data for analysis
    - Do a better job of Analysis: avoid unnecessary file I/O by combining analysis step with data preprocessing step into one step
  - By extending current Hadoop framework, results show a throughput improvement of 33% using a bioinformatics app (CloudBurst) and an I/O kernel of Halo catalogs

DEPARTMENT OF EECS
ELECTRICAL & COMPUTER ENGINEERING DIVISION

Challenges to enable data-intensive HPC analytics in Cloud environment

- What if we have similar infrastructure support as in cloud such as Open Cirrus UIUC clouds
  - Problems have been solved in our ACM HPDC2010 MRAP
- What if not
  - Amazon EC2, Microsoft Azure
  - Challenges
    - Scalability: DIFS = *co-located compute and storage + chunk based distributed file system*
    - Software resiliency at data access level
      - Multi-replication, runtime scheduling support
    - Programming language: MapReduce, share nothing only allows accessing one source data per Map task

The state of the art: HPC in Microsoft Azure and Amazon EC2

- **Amazon**
  - The Amazon EC2 Cluster Compute and Cluster GPU instance types
  - Amazon Elastic MapReduce
  - Public Data Sets on AWS
- **Azure**
  - IPDPS'10: escience in the cloud: A modis satellite data reprojection and reduction pipeline in the windows azure platform.
  - Supercomputing (SC) 2010 conference, Microsoft Corp. announced the release of NCBI BLAST on Windows Azure.
  - Windows HPC Server 2008 R2 released by Windows Azure.
  - Many others ongoing

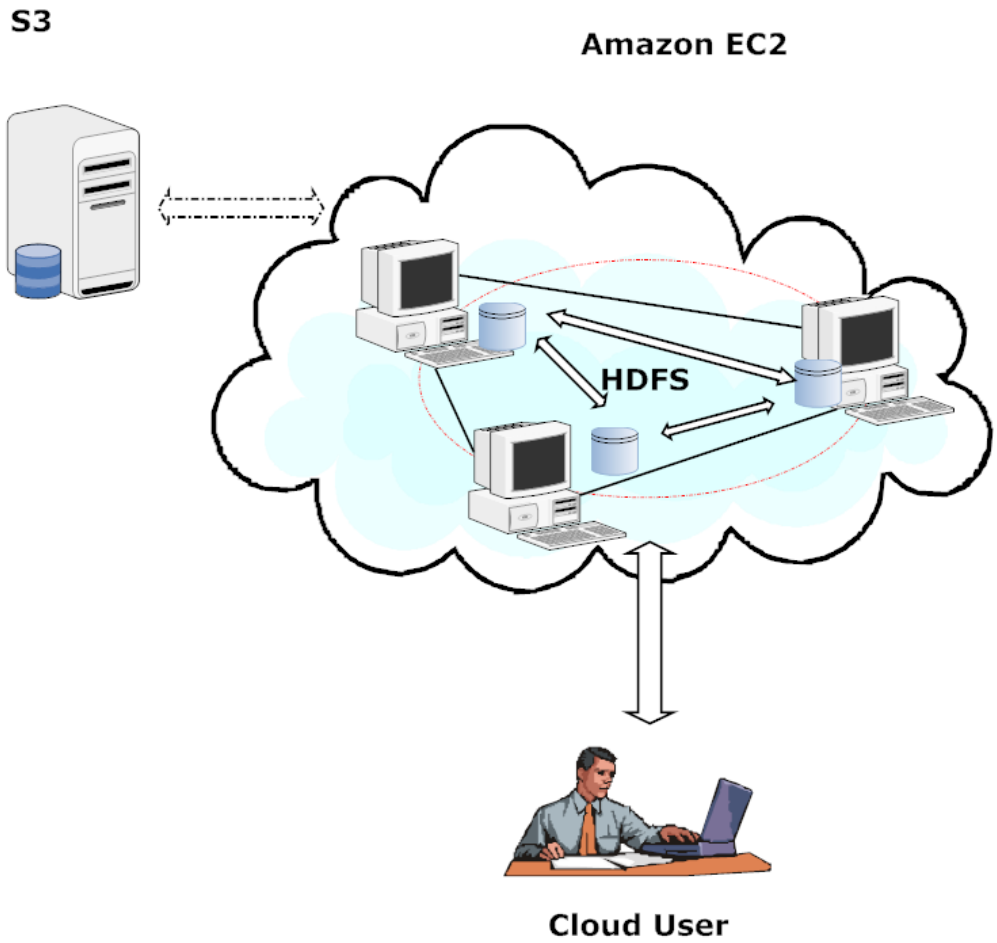
Limitations of Current Cloud Solutions on implementing co-located compute & storage

- **Different clouds share the following drawbacks (Amazon EC2, Azure, Linux Open Cirrus)**
  - **High Performance overhead**:
    - Use of virtualization technology leads to sharing of physical disks, lacking of direct harnessment of resources;
  - **Local storage associated with a VM instance is not persistent**:
    - Instances are randomly assigned to users. Once the instance is released, it can be assigned to other users, then data on that is erased; unsuitable for constructing HDFS.
  - **Lack of a physical distributed persistent storage**: centrally managed storage;
  - **Lack of a data transfer function between local storages of instances in Azure**, making data sharing across instances is disabled.

Limitations of Current Solutions on supporting co-located compute and storage (cont…)

- **For Amazon, some possible solutions:**
  - Constructing DIFS using local storage of virtual machines;
    - If input and output data are still stored at S3, then cost goes up for the HDFS replica and there might be bottleneck for input and output transfer;
    - If not, it increases the risk of data loss since the data is not copied to the permanent locations.
  - Taking advantage of EBS which exists independently from the life of an instance to store the data.
    - EBS is charged according to the volume size you allocate instead of the data size you store;
    - The data communication between the instance and the EBS volume is still through the network and it costs a lot of time.

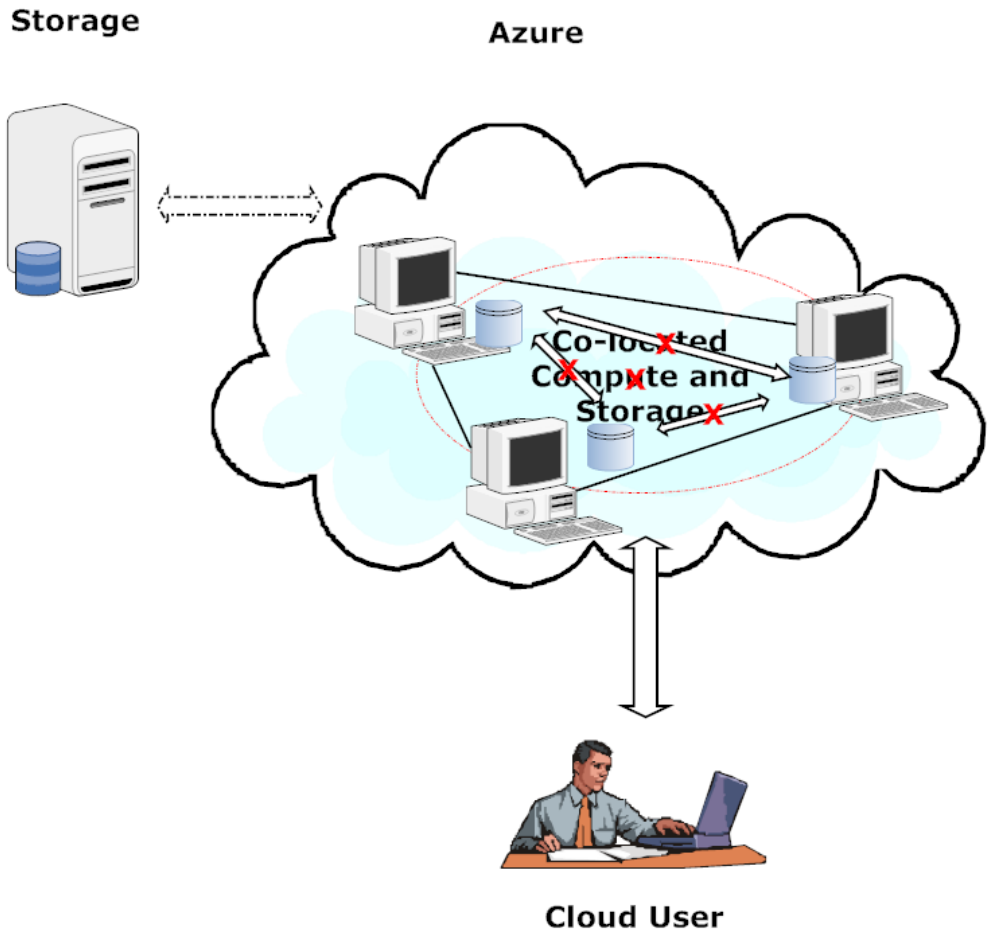
Limitations of Current Solutions on supporting co-located compute and storage (cont…)



Co-located Compute and storage on Azure (1)

- Instances can not access local storage from other instances, so it is not possible to construct HDFS on current infrastructure;
- VMs are assigned to users randomly, it can not be guaranteed that the instance you are using is the same one, so the data might be lost at any point of time of a failure, or after you resume you work.

Co-located Compute and storage on Azure (2)



Our HPC analytics Solutions for Azure

- **Phase1: Data Preprocessing (i.e. Data Staging)**

  Our previous work has proved that in HPC analytics applications, data format and access patterns could be used to significantly improve the performance of future data analysis.

  ❖**Our proposal:**

  HPC upload middleware to preprocess the data before being uploaded to the Azure storage

- **Phase 2: Conducting Analysis**

  Duplicate a MapReduce/Hadoop context on Azure

Develop a MapReduce/Hadoop framework on Azure

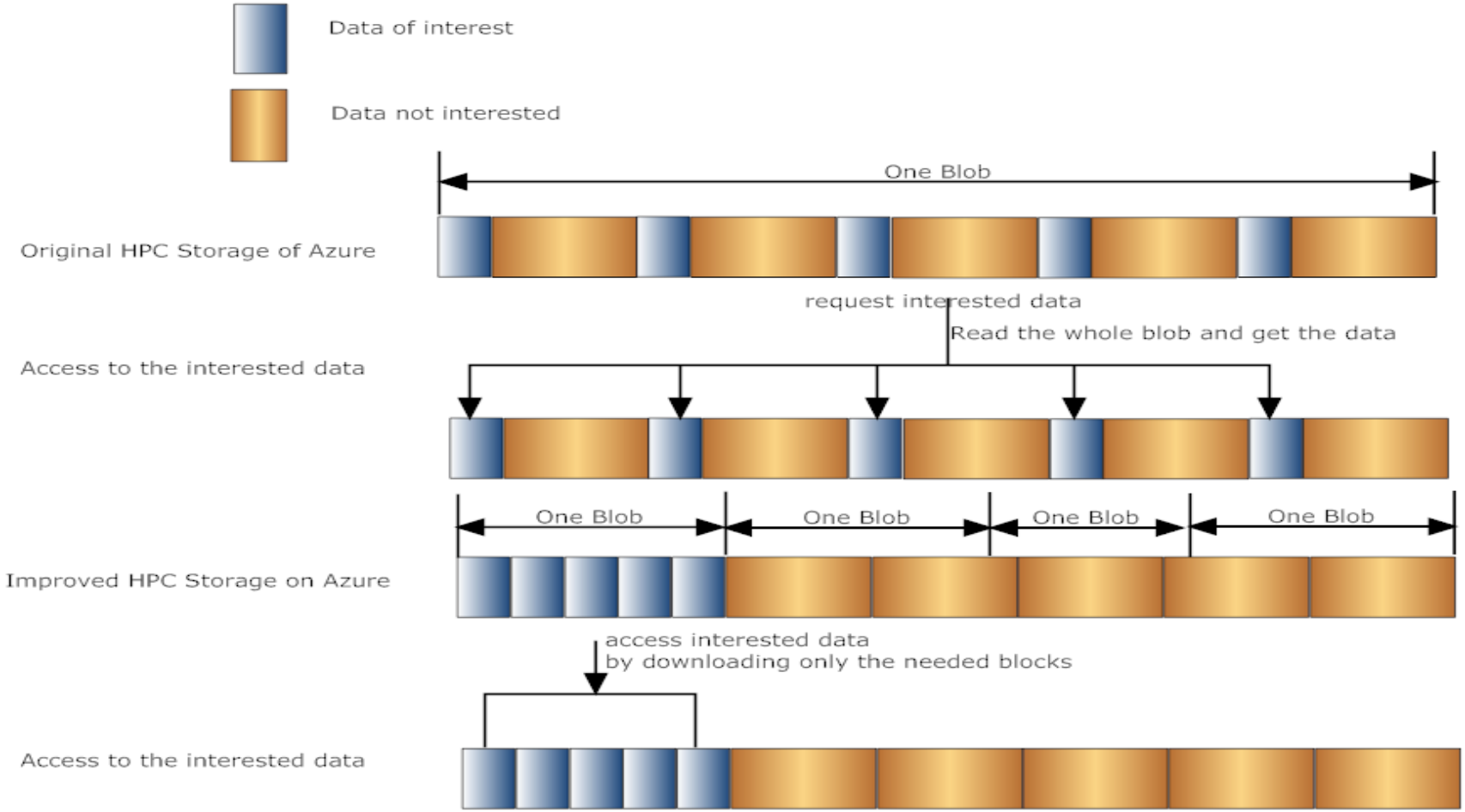
Phase 1: HPC Upload Middleware

- Motivation
- System Design
- Example: Gadget-II, Astrophysics data set

HPC Upload Middleware -- Motivation

- **There are special data semantics for HPC data well defined in HPC: NetCDF, HDF5, etc**
- **Azure storage does not consider HPC data semantics;**
- **Make Azure store the HPC data with the awareness of HPC data semantics using a Uploading Middleware**
  - Recognize NetCDF, HDF5, Gadget, etc structured data in HPC

HPC Upload Middleware -- Motivation

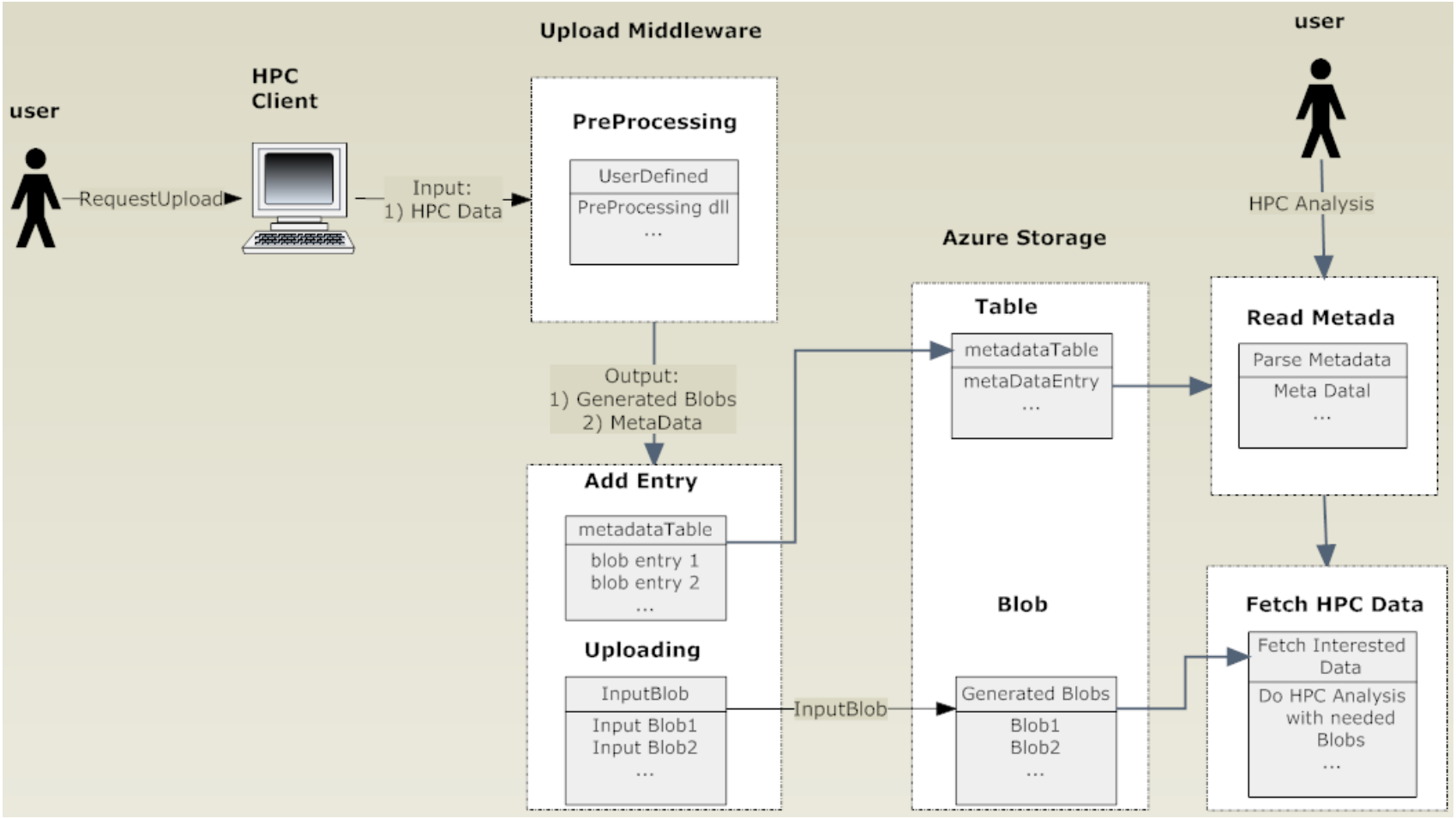


HPC Upload Middleware -- System Design (1)

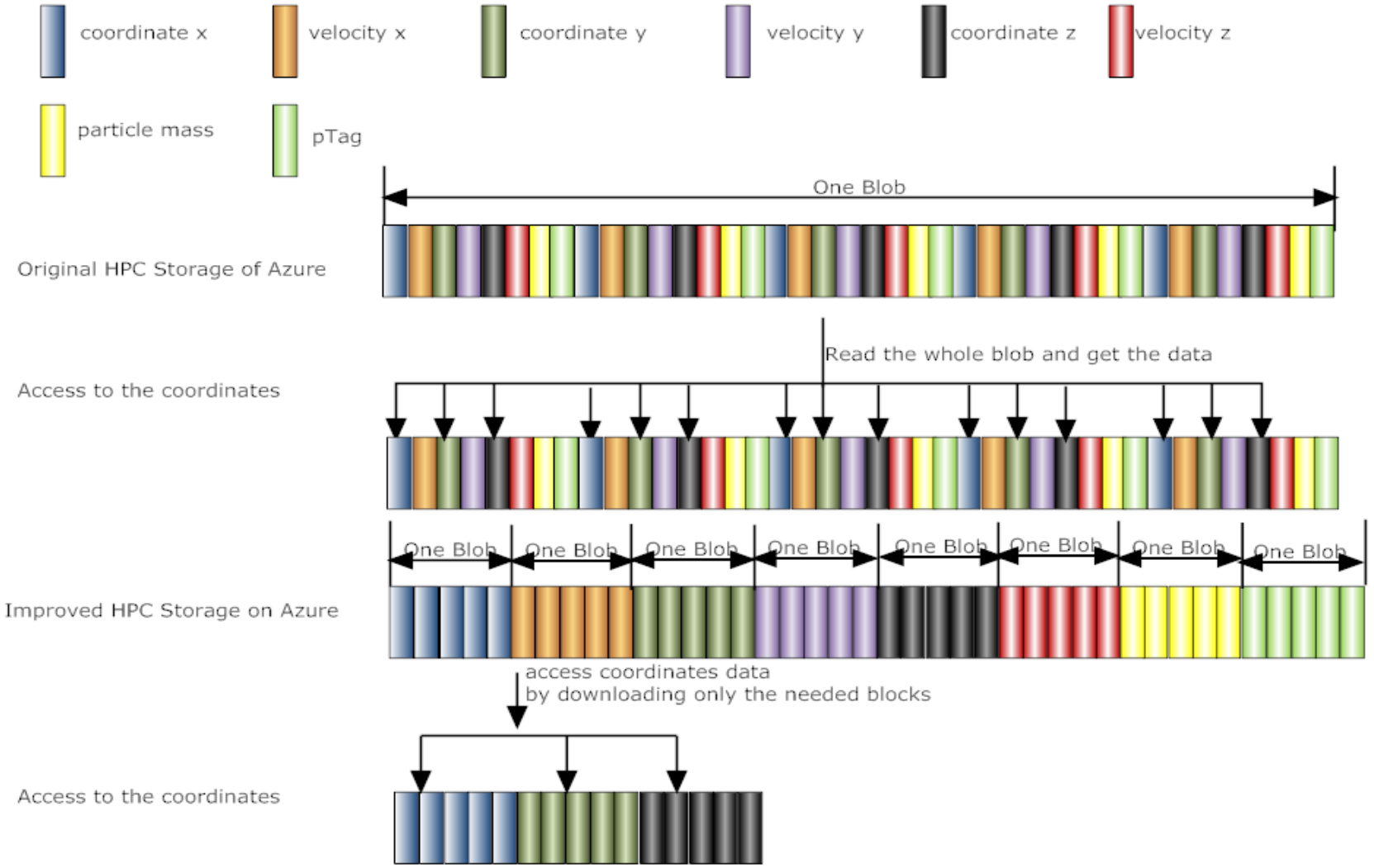
- **Two models: block blob and page blob**
  - Data Preprocessing
  - ❖ Preprocessing the HPC data using user defined method (e.g., a DLL implementation);
  - ❖Generate preprocessed HPC data together with their semantics tags (i.e., metadata describes data content)
  - Data Uploading (block blob or page blob)
    - ❖Upload generated HPC data blobs using functions

      PutBlock and PutBlockList of CloudBlockBlob for block blob uploading, and WritePages for page blob uploading respectively

HPC Upload Middleware -- System Design (2)



HPC Upload Middleware -- Gadget II



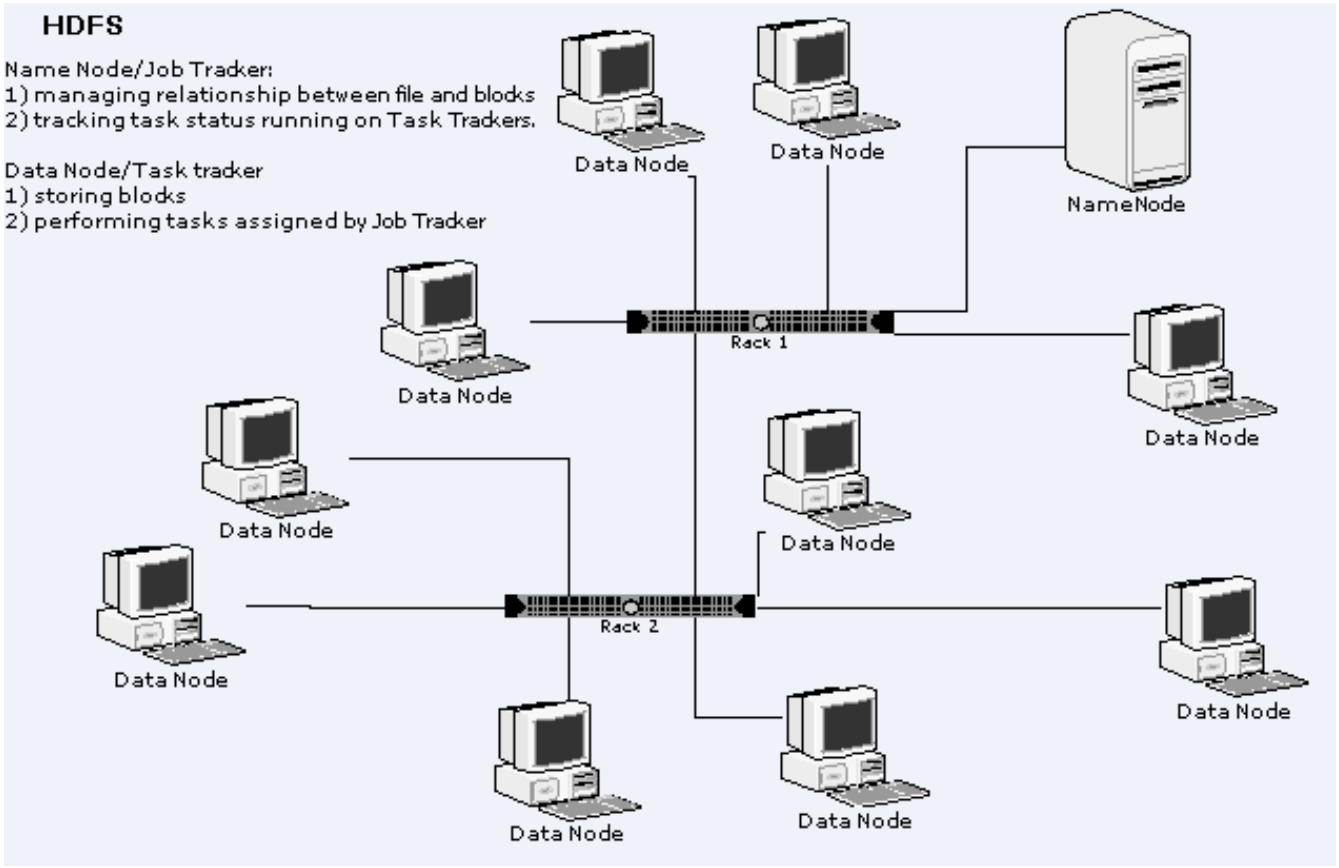
Phase 2: Running an Analysis program on MapReduce/HDFS & Azure

- HDFS
- MapReduce
- Challenges to deploy MapReduce/HDFS over Azure

HDFS (1)

- HDFS system is a chunk-based storage system.
- Comprised of one namenode managing metadata and block locations, and several datanodes storing chunks of the file.
- Namenode often performs as JobTracker to schedule and track the tasks and Datanodes as TaskTrackers to perform specific tasks.
- Files are split into fix-sized chunks and stored across datanodes.

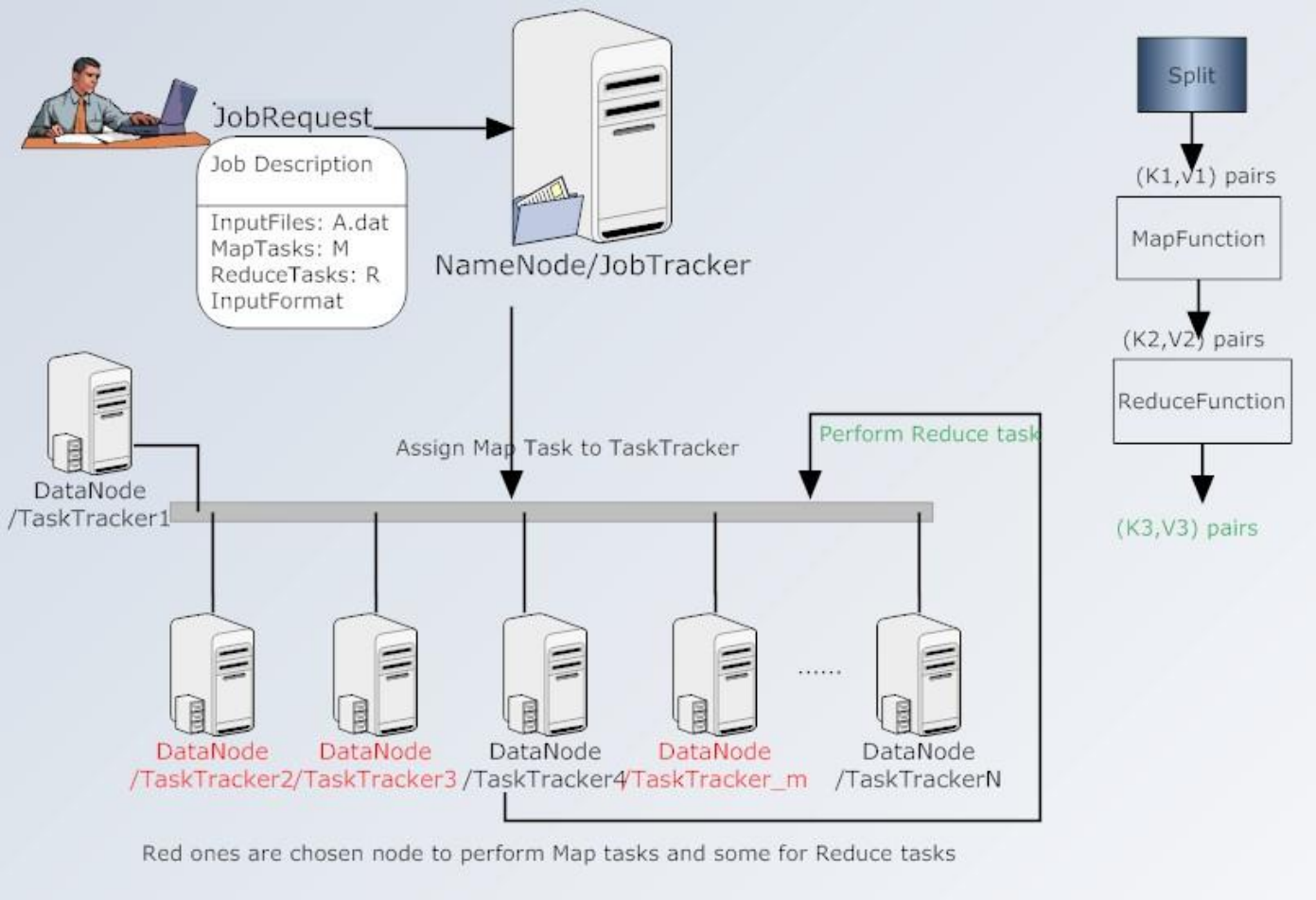
HDFS (2)


HDFS
Name Node/Job Tracker:
1) managing relationship between file and blocks
2) tracking task status running on Task Trackers.
Data Node/Task tracker
1) storing blocks
2) performing tasks assigned by Job Tracker
Data Node
Data Node
NameNode
Rack 1
Data Node
Data Node
Data Node
Data Node
Rack 2
Data Node
Data Node
Data Node
Data Node

MapReduce (1)

- MapReduce is a programming framework model;
- Comprised of 2 phases: map and reduce.
- Data -> (key, value) pairs -> map function -> (keyx, valuex) pairs -> reduce function -> (keyr, valuer) pairs

MapReduce (2)


JobRequest
Job Description
InputFiles: A.dat
MapTasks: M
ReduceTasks: R
InputFormat
NameNode/JobTracker
Split
(K1,V1) pairs
MapFunction
(K2,V2) pairs
ReduceFunction
(K3,V3) pairs
Assign Map Task to TaskTracker
Perform Reduce task
DataNode
/TaskTracker1
DataNode
/TaskTracker2
DataNode
/TaskTracker3
DataNode
/TaskTracker4
DataNode
/TaskTracker_m
.....
DataNode
/TaskTrackerN
Red ones are chosen node to perform Map tasks and some for Reduce tasks

Challenges to deploy HDFS/MapReduce over Azure

- Virtual machines in Azure make instances in HDFS dynamic, resulting in data loss if data stored in VMs.
- Currently the instance can not access the local storage of other instances, disabling the data transfer between local storages of instances.
- Centralized managed storage makes co-located computation and storage impossible.

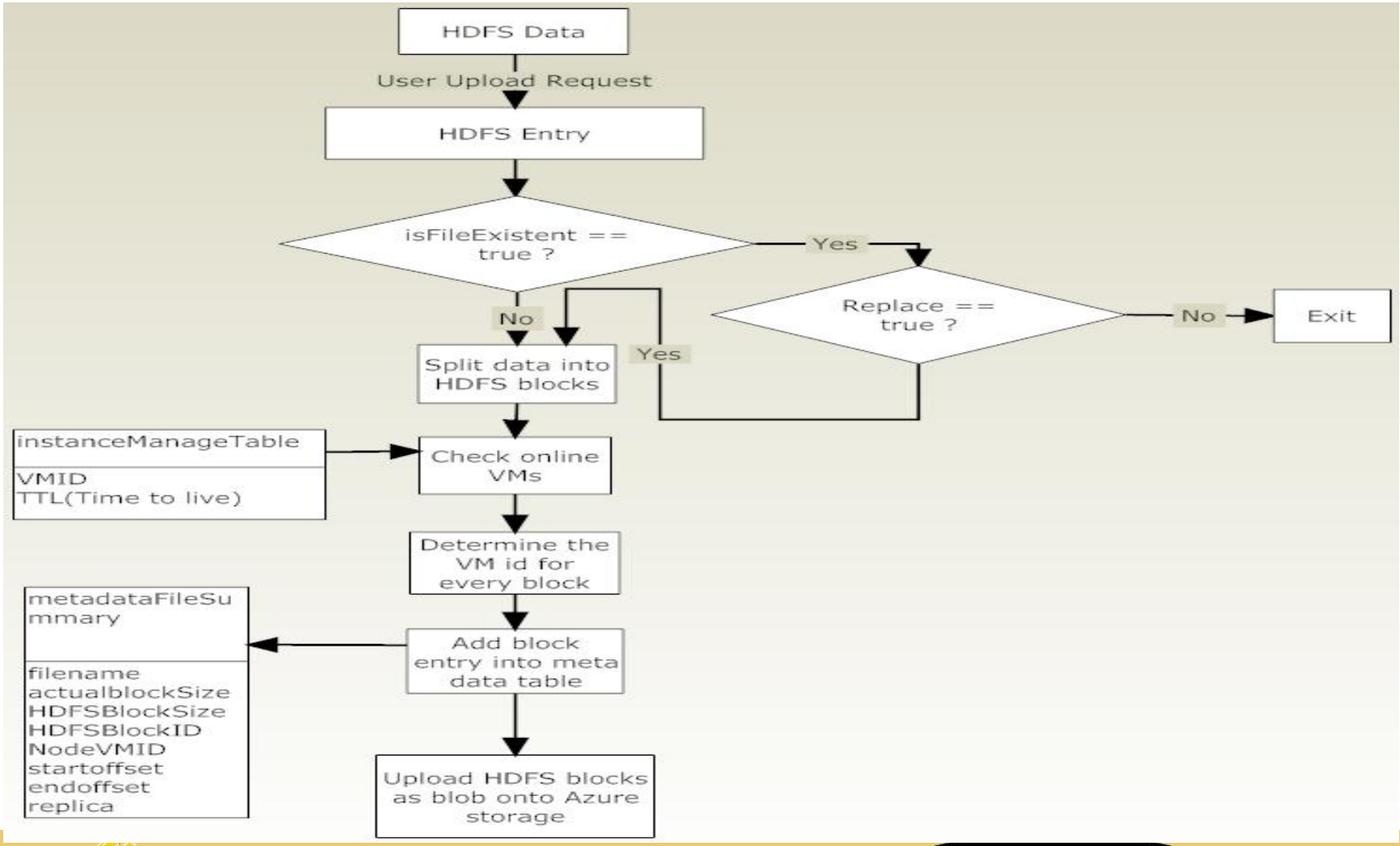
Our current solution: constructing a logical HDFS layer on top of Azure's central storage

- **Idea: mimic a HDFS context**
  - Create two tables on top of Azure's central storage infrastructure
  - One table manages HDFS blocks
  - One table manages Azure compute nodes
- **Advantages:**
  - Port many existing MapReduce/Hadoop applications to Azure
  - Support co-located compute and storage
- **Drawbacks:**
  - Still need to save all important data to central storage

HDFS attempt on Azure (1)

- **Uploading**
  - A table named metadataFileSummary is used to store the block entries;
  - Another table named instanceManageTable is used to store the online VM IDs (which is the unique identifier for every VM instance);
  - HDFS blocks are uploaded as block blobs;
  - Data is split into HDFS blocks using predefined size;
  - Check the available VM instances in table instanceManageTable and distribute the blocks randomly across them;
  - Add block entry in table metadataFileSummary ;
  - Upload HDFS blocks onto the Azure storage;

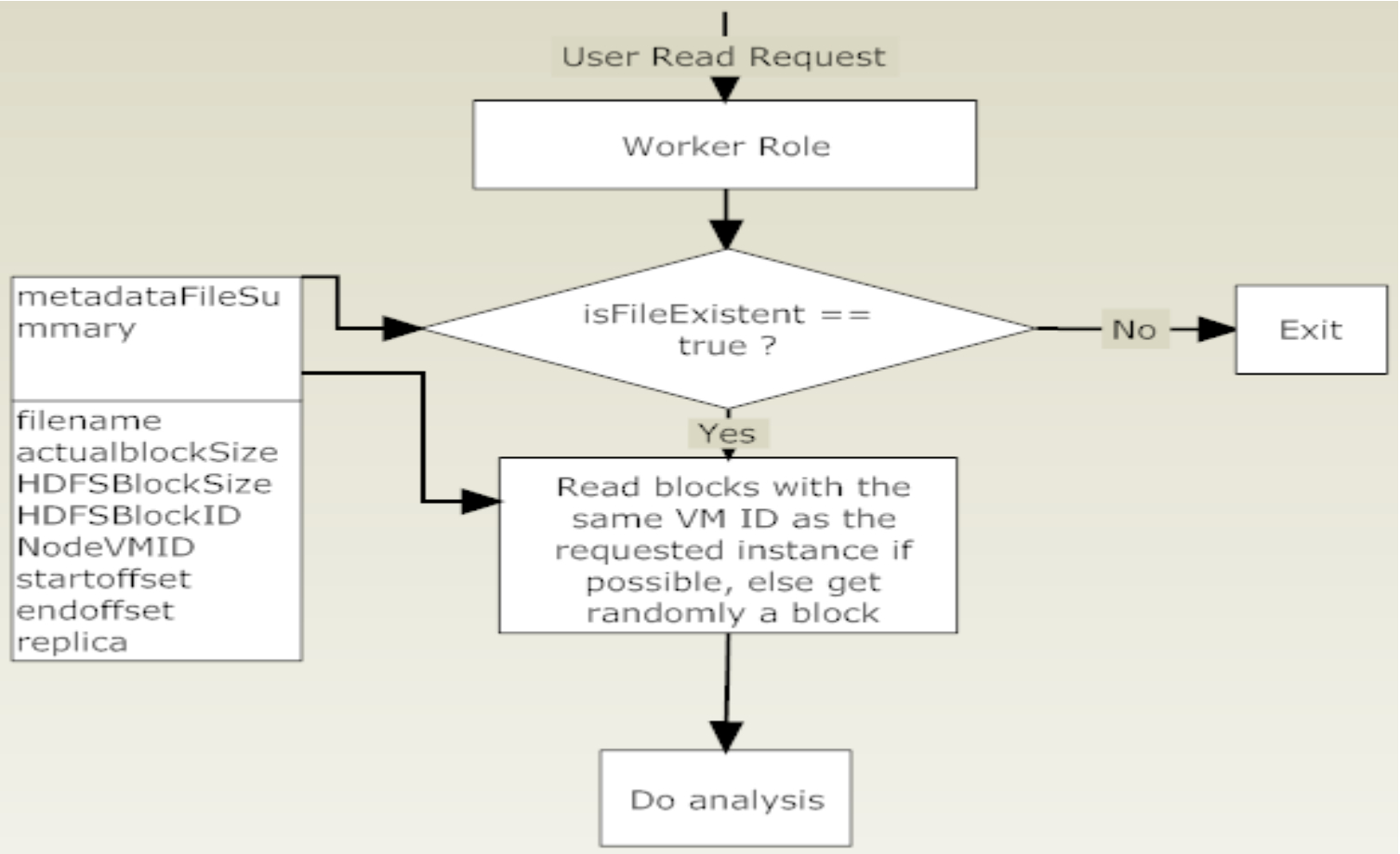
HDFS attempt on Azure (2)



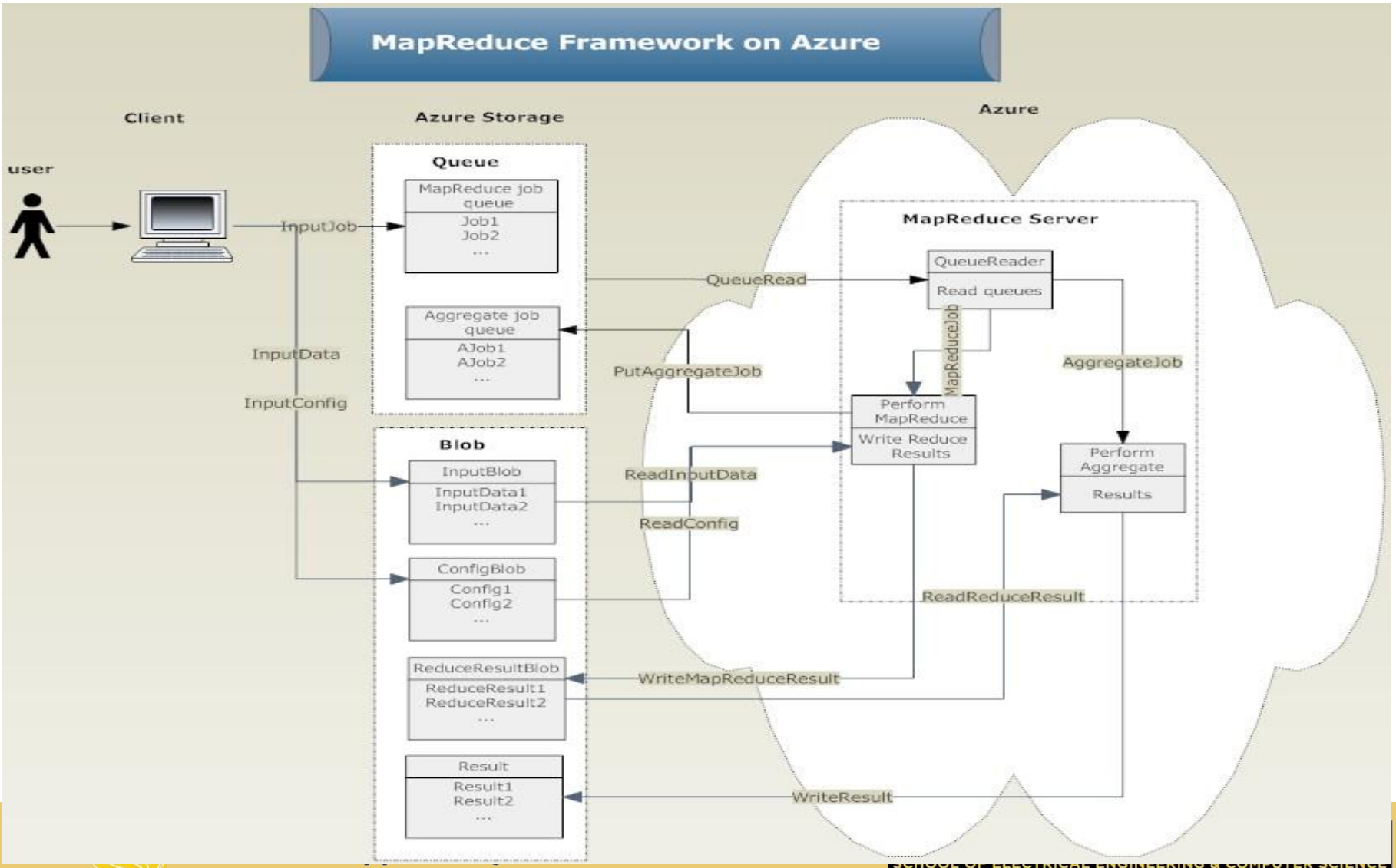
HDFS attempt on Azure (3)

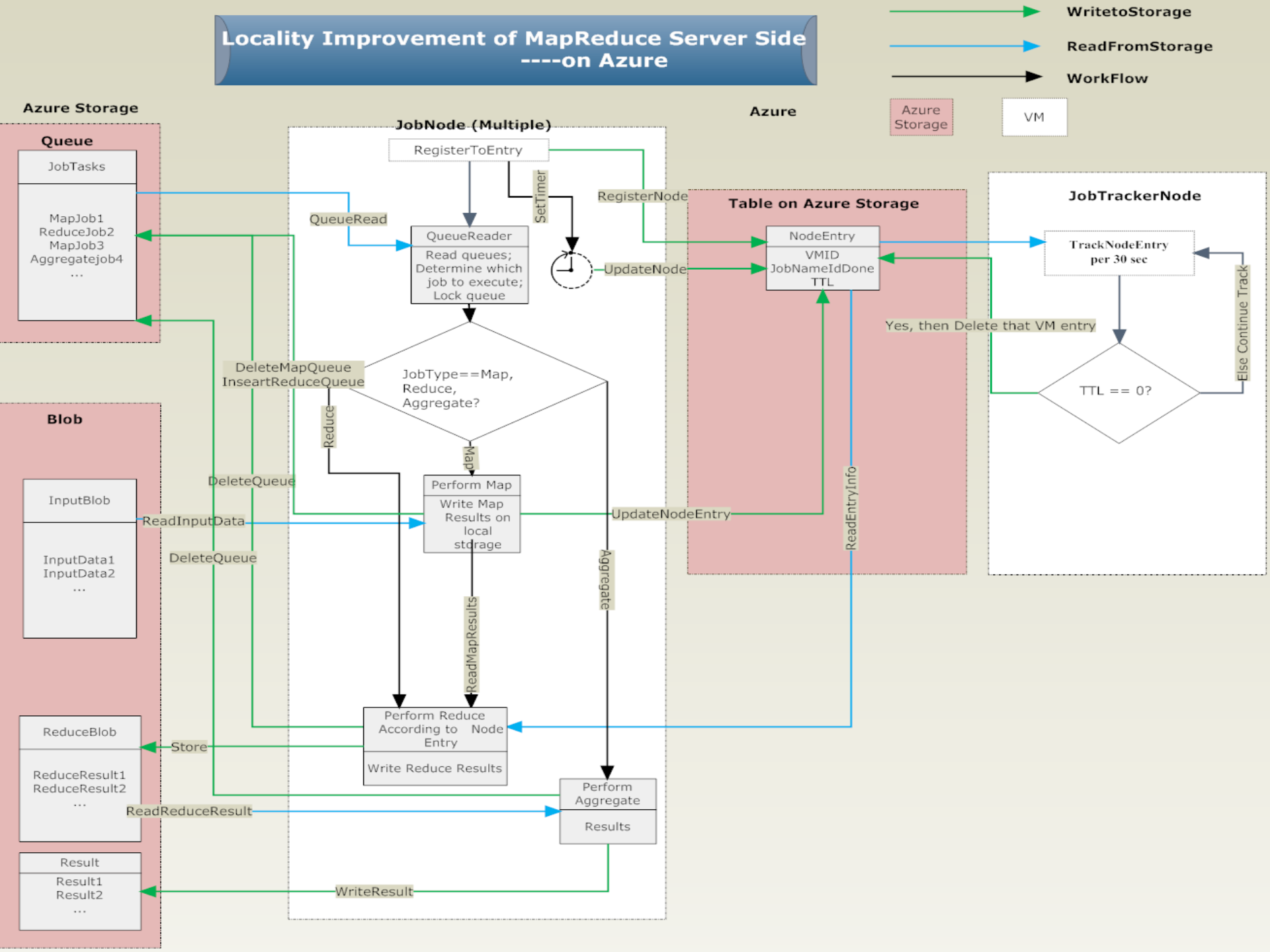
- **Downloading**
  - A table named metadataFileSummary is used to store the block entries;
  - Fetch all the block entries for the file requested;
  - Read the block ID and and related VM IDs;
  - Pick the one with the same VM ID as the requested instance, other wise random get one;
  - HDFS blocks are downloaded as block blobs.

HDFS attempt on Azure (4)

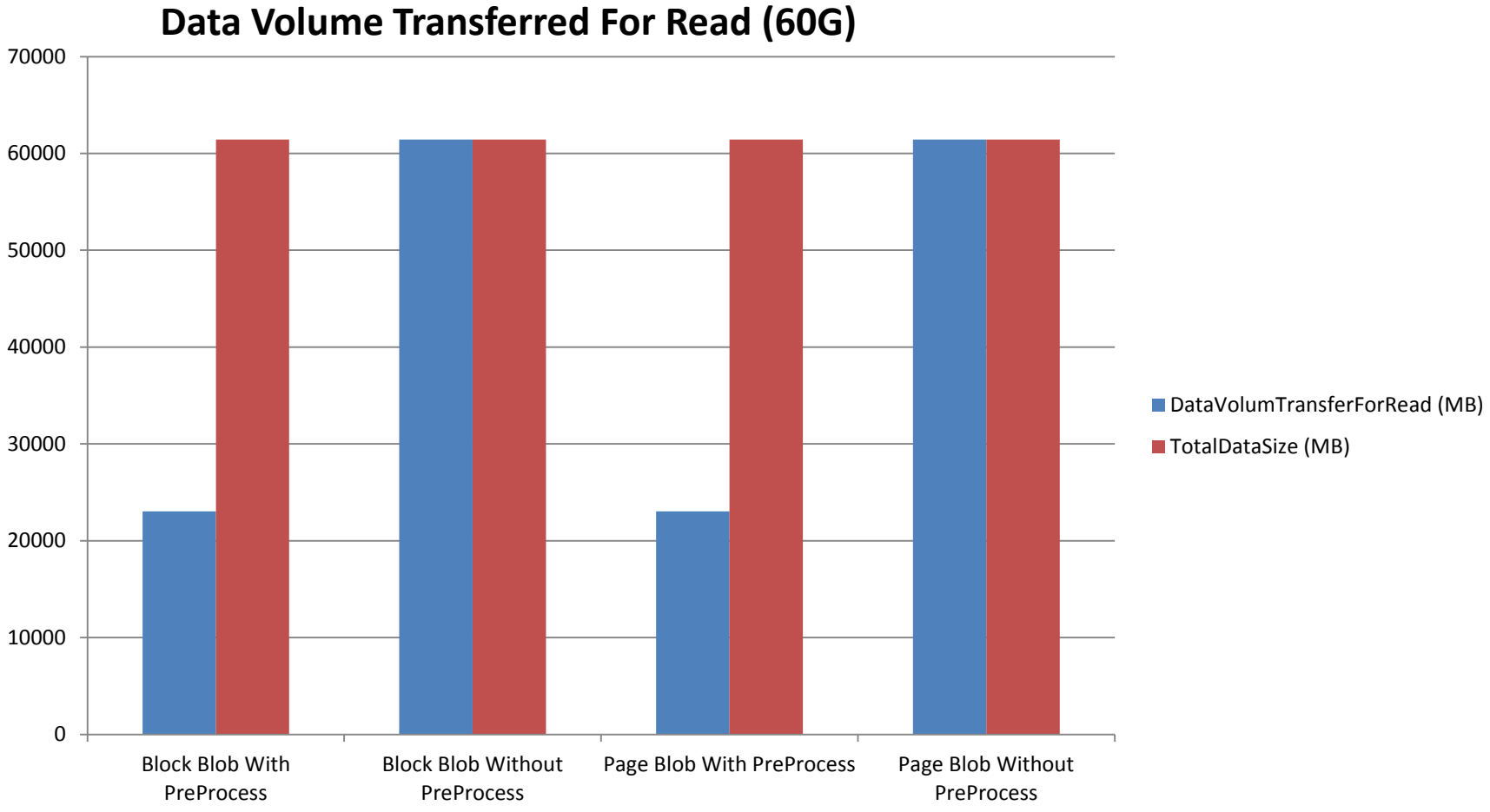


MapReduce over Azure -- architecture

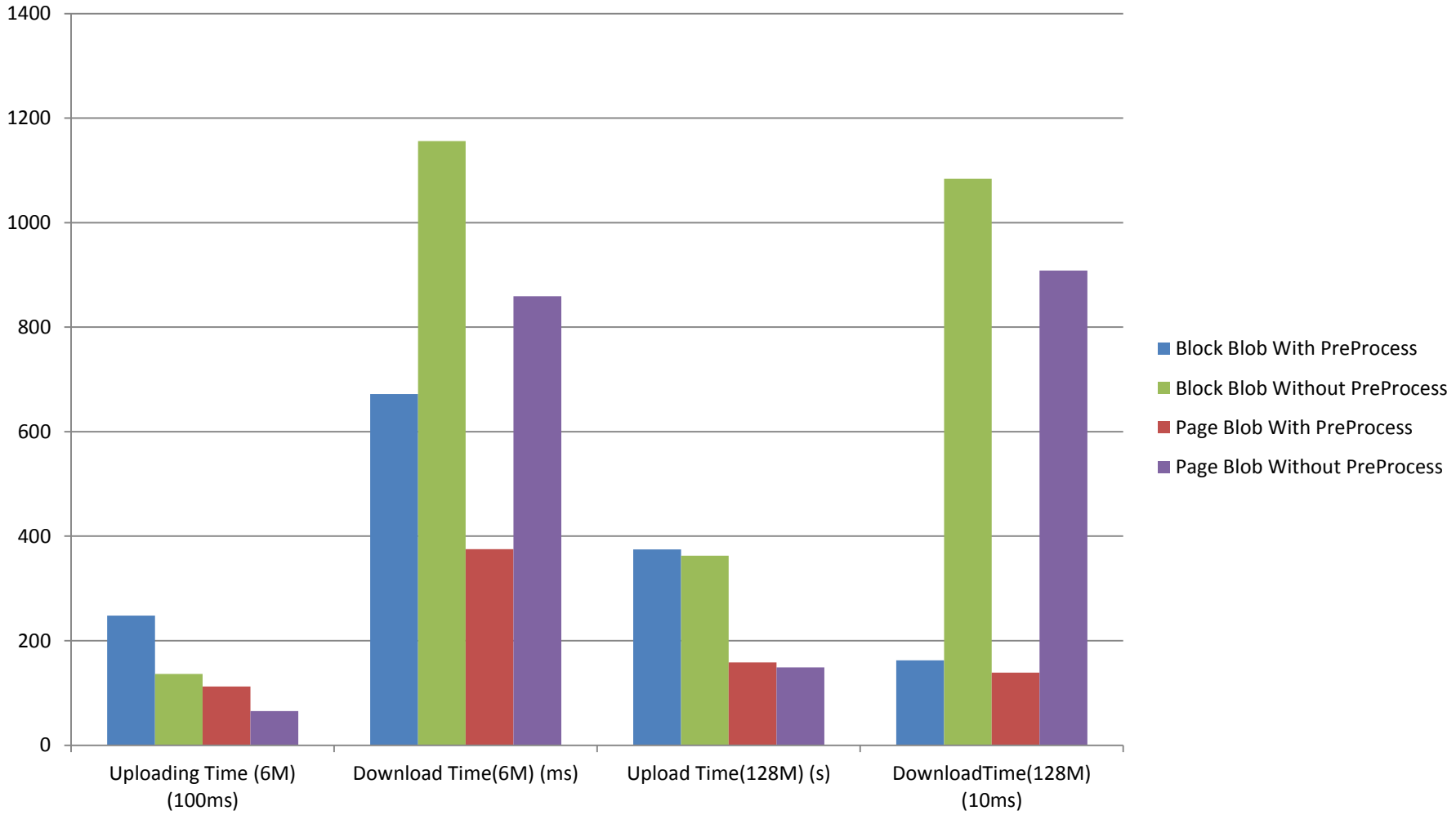



Locality Improvement of MapReduce Server Side ----on Azure
WritetoStorage
ReadFromStorage
WorkFlow
Azure Storage
VM
Azure Storage
Queue
JobTasks
MapJob1
ReduceJob2
MapJob3
Aggregatejob4
...
Blob
InputBlob
InputData1
InputData2
...
ReduceBlob
ReduceResult1
ReduceResult2
...
Result
Result1
Result2
...
JobNode (Multiple)
RegisterToEntry
SetTimer
QueueRead
QueueReader
Read queues; Determine which job to execute; Lock queue
RegisterNode
UpdateNode
DeleteMapQueue
InseartReduceQueue
JobType==Map, Reduce, Aggregate?
Reduce
Map
DeleteQueue
ReadInputData
DeleteQueue
Perform Map
Write Map Results on local storage
UpdateNodeEntry
Aggregate
ReadMapResults
Perform Reduce According to Node Entry
Write Reduce Results
Store
Perform Aggregate
Results
ReadReduceResult
WriteResult
Azure
Table on Azure Storage
NodeEntry
VMID
JobNameIdDone
TTL
ReadEntryInfo
JobTrackerNode
TrackNodeEntry per 30 sec
Yes, then Delete that VM entry
TTL == 0?
Else Continue Track

Preliminary Results of HPC uploading middleware on Azure (1)



Preliminary Results of HPC uploading middleware on Azure (2)



Preliminary Results of HPC uploading middleware on Azure (3)

- **Access Time improvement 128 MB (%)**

  blockblob with preprocess 85.00922509
  pageblob with preprocess 84.69162996

- **Access Time improvement 6 MB (%)**

  blockblob with preprocess 41.87716263
  pageblob with preprocess 56.34458673

Breakdown of results

- **With Preprocessing:**
  - Uploading Time = PreProcessing Time + AddEntry Time + Data Uploading Time
  - Downloading Time = Entry Query Time + Data Downloading Time
- **Without Preprocessing:**
  - Uploading Time = Data Uploading Time
    Downloading Time = Data Downloading Time + Strided Access Time
- **User related latency:**
  - Preprocessing Time means Overhead time caused by uploading middleware: Adding Entry Time, Entry Query Time

Conclusion

- **We demonstrate the feasibility of:**
  - Deploying the HPC analysis applications on Azure;
  - Deploying the HDFS over Azure;
  - Deploying the MapReduce over HDFS at Azure.
- **With data semantics awareness at both data staging and analysis phases, the performance of HPC analytics programs can be significantly improved at Clouds**
  - About 85% improvement on data access latency

Thanks!