**GEETHANJALI INSTITUE OF SCIENCE&TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING CS6703 – GRID AND CLOUD COMPUTING**

# Unit – IV – PROGRAMMING MODEL PART – A

## Name any four services offered in GT4. (Nov/Dec 16)

* + **GRAM** (Global Resource Allocation Manager) – Grid resource access and management.
  + **Nexus** – used for unicast and multicast communication
  + **GSI** (Grid Security Infrastructure) – Used for Authentication and security.
  + **MDS** (Monitory and Discovery Service) – Distributed acces to structure and state information.
  + **HBM** (Heart Beat Monitor) – monitor heart beat of system components.
  + **GASS** (Global Access of Secondary Storage) – Grid access of data in remote secondary storage.
  + **GridFTP** (Grid File Transfer) – used for inter-node fast file transfer

## What are the advantages of using Hadoop? (Nov/Dec 16)

* + Scalable
  + Flexible
  + Fast
  + Resilient to Failure
  + Independent

## Write the significant use of GRAM. (Apr/May 17)

The Globus Toolkit includes a set of service components collectively referred to as the Globus Resource Allocation Manager (GRAM). GRAM simplifies the use of remote systems by providing a single standard interface for requesting and using remote system resources for the execution of "jobs".

## Name the different modules in Hadoop framework. (Apr/May 17)

* + **Hadoop Common:** this includes the common utilities that support the other Hadoop modules
  + **HDFS:** the Hadoop Distributed File System provides unrestricted, high-speed access to the application data.
  + **Hadoop YARN:** this technology accomplishes scheduling of job and efficient management of the cluster resource.
  + **MapReduce:** highly efficient methodology for parallel processing of huge volumes of data.

## “HDFS is fault tolerant. Is it true? Justify your answer. (Nov/Dec 17)

HDFS is highly fault tolerant. It handles faults by the process of replica creation. The replica of users data is created on different machines in the HDFS cluster. So whenever if any machine in the cluster goes down, then data can be accessed from other machine in which same copy of data was created. HDFS also maintains the replication factor by creating replica of data on other available machines in the cluster if suddenly one machine fails.

## What is the purpose of heart beat in hadoop? (Nov/Dec 17)

* + In **Hadoop**, Namenode and Datanode are two physically separated machines, therefore **Heartbeat** is the signal that is sent by the datanode to the namenode after the regular interval to time to indicate its presence, i.e. to indicate that it is alive.
  + In case Namenode does not receive the heartbeat from a Datanode in a certain amount of time(within 10 mins), Namenode then considers that datanode as a dead machine.
  + Datanode along with heartbeat also sends the block report to Namenode, block report typically contains the list of all the blocks on a datanode.

## How does divide-and-conquer strategy relates to MapReduce paradigm? (Apr/May 18)

In MapReduce, you divide the work up serially, execute work packets in parallel, and tag the results to indicate which results go with which other results. The merging is then serial for all the results with the same tag, but can be executed in parallel for results that have different tags. In more previous systems, the merge step became a bottleneck for all but the most truly trivial tasks. With MapReduce it *can* still be if the nature of the tasks requires that all merging be done serially.

## Brief out the main components of Globus toolkit. (Apr/May 18)

* + Common runtime components
  + Security
  + Data management
  + Information services
  + Execution management

## What is distributed file system? (Nov/Dec 18)

When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. File systems that manage the storage across a network of machines are called distributed file systems.

## How MapReduce framework executes user jobs? (Nov/Dec 18)

To begin, a user runs a MapReduce program on the client node which instantiates a Job client object. Next, the Job client submits the job to the JobTracker. Then the job tracker creates a set of map and reducetasks which get sent to the appropriate task trackers.

## What are the functionalities of grid middleware?

Grid middleware is a specific software product, which enables the sharing of heterogeneous resources, and Virtual Organizations. It is installed and integrated into the existing infrastructure of the involved company or companies, and provides a special layer placed among the heterogeneous infrastructure and the specific user applications. Middleware glues the allocated resources with specific user applications. Major grid middlewares are Globus Toolkit, gLite, UNICORE, BONIC, CGSP, Condor-G and Sun Grid Engine etc.

## Write short notes on GT4?

The Globus Toolkit was initially motivated by a desire to remove obstacles that prevent seamless collaboration, and thus sharing of resources and services, in scientific and engineering applications. The toolkit addresses common problems and issues related to grid resource discovery, management, communication, security, fault detection, and portability.

## Explain the different types of GT4 Data management?

* + Globus Toolkit 4 Data Management tools within the toolkit fall into either of two categories data replication and data movement
  + Data Replication consists of Replica Location Service (RLS)
  + Data Movement consists of GridFTP and Reliable File Transfer (RFT)

## List the security issues of Globus Toolkit?

* + Has to cross administrative domains.
  + Need agreed mechanisms and standards.
  + Focus on Internet security mechanisms, modified to handle the special needs of Grid computing.
  + Distributed resources must be protected from unauthorized access

## What is the role of namenode in HDFS?

The namenode (the master) manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log.

## What is the role of datanode in HDFS?

Datanodes (workers) are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.

**PART - B**

## Draw and explain the global toolkit architecture. (16) (Nov/Dec 16)

* + The Globus Toolkit is an open middleware library for the grid computing communities. These open source software libraries support many operational grids and their applications on an international basis.
  + The toolkit addresses common problems and issues related to grid resource discovery, management, communication, security, fault detection, and portability. The software itself provides a variety of components and capabilities.
  + The library includes a rich set of service implementations. The implemented software supports grid infrastructure management, provides tools for building new web services in Java, C, and Python, builds a powerful standard-based.
  + Security infrastructure and client APIs (in different languages), and offers comprehensive command-line programs for accessing various grid services.
  + The Globus Toolkit was initially motivated by a desire to remove obstacles that prevent seamless collaboration, and thus sharing of resources and services, in scientific and engineering applications. The shared resources can be computers, storage, data, services, networks, science instruments (e.g., sensors), and so on.

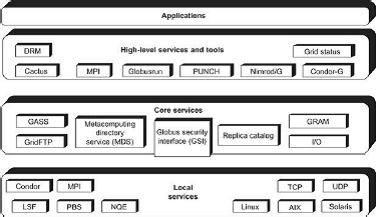


Figure: Globus Toolkit GT4 supports distributed and cluster computing services

**The GT4 Library**

* + The GT4 Library GT4 offers the middle-level core services in grid applications. The high- level services and tools, such as MPI, Condor-G, and Nirod/G, are developed by third parties for general purpose distributed computing applications.
  + The local services, such as LSF, TCP, Linux, and Condor, are at the bottom level and are fundamental tools supplied by other developers.

**Globus Job Workflow**

* + A typical job execution sequence proceeds as follows: The user delegates his credentials to a delegation service. The user submits a job request to GRAM with the delegation identifier as a parameter.
  + GRAM parses the request, retrieves the user proxy certificate from the delegation service, and then acts on behalf of the user. GRAM sends a transfer request to the RFT, which applies GridFTP to bring in the necessary files.
  + GRAM invokes a local scheduler via a GRAM adapter and the SEG initiates a set of user jobs. The local scheduler reports the job state to the SEG. Once the job is complete, GRAM uses RFT and GridFTP to stage out the resultant files.

**Client-Globus Interactions**

* + There are strong interactions between provider programs and user code. GT4 makes heavy use of industry-standard web service protocols and mechanisms in service
  + Description, discovery, access, authentication, authorization, and the like. GT4 makes extensive use of Java, C, and Python to write user code. Web service mechanisms define
  + specific interfaces for grid computing. Web services provide flexible, extensible, and widely adopted XML-based interfaces.

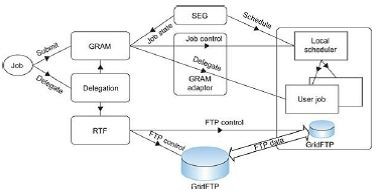


Figure: Globus job workflow among interactive functional modules.

These demand computational, communication, data, and storage resources. We must enable a range of end-user tools that provide the higher-level capabilities needed in specific user applications. Developers can use these services and libraries to build simple and complex systems quickly.

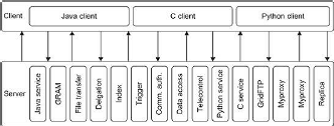
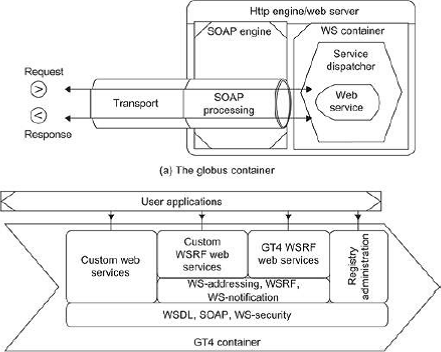


Figure: Client and GT4 server interactions; vertical boxes correspond to service programs and horizontal boxes represent the user codes.

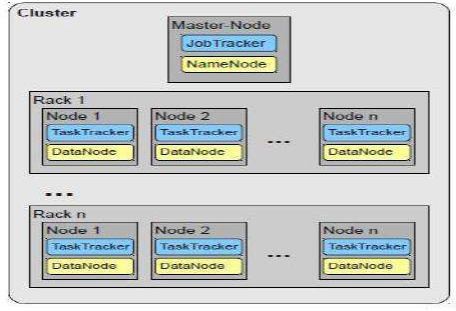
The horizontal boxes in the client domain denote custom applications and/or third-party tools that access GT4 services. The toolkit programs provide a set of useful infrastructure services.

Three containers are used to host user-developed services written in Java, Python, and C, respectively. These containers provide implementations of security, management, discovery, state management, and other mechanisms frequently required when building services.



## Give a detailed note on Hadoop framework. (16) (Nov/Dec 16)

* + The Hadoop Framework This section contains a general overview of the components and their association in the Hadoop Framework without getting into the depths of what exactly they do.
  + The Hadoop Framework consists of several modules which provide different parts of the necessary functionality to distribute tasks and data across a cluster.



## Fig. 1. Abstract view on a Cluster.

* + A cluster consists of several nodes organized into racks, each node running a TaskTracker for the MapReduce tasks and a DataNode for the distributed storage system.
  + One special node, the Master-Node runs the JobTracker and the NameNode which are organizing the distribution of tasks and data.

## Hadoop Framework

* + The Hadoop Framework is written in Java but can run MapReduce programs expressed in various languages, e.g. Ruby, Python and C++.
  + This functionality is implemented using Unix standard streams, the input for the Map function is streamed to the program which implements it, no matter which language it is implemented in as long as it can read and write from standard input and to standard output.
  + Another way to achieve this is with the \Hadoop Pipes", an interface to C++ which uses sockets for the communication with the Map and Reduce functions.
  + The central parts of the framework are in the Common module which contains generally availabe interfaces and tools to support the usage of the other modules.
  + These modules include MapReduce, a framework to distribute the processing of large data sets on compute clusters, and HDFS, the Hadoop Distributed Filesys- tem which enables large clusters to save and access data with high throughput.

## Communication

* + In every cluster running Hadoop there is one node addressed as the \Master-Node". This node constitutes a single point of failure which is why Hadoop is not a highavailability system. Communication from outside of the cluster is completely handled by the Master-Node, because it keeps all the necessary information about the cluster,usage of the nodes, disk- space and distribution of files.
  + Every MapReduce task is sent to the MasterNode where the JobTracker, the component that manages the MapReduce jobs, and the NameNode, which is responsible for everything concerning the system, are running.
  + The JobTracker communicates with Task-Trackers on the nodes which receive work units and send reports about their status back to the JobTracker.

## Cluster

* + The frameworks' con\_guration should include information about the toplogy to allow the underlying filesytem the use of the location for its replication strategy.
  + Also it allows the framework to place MapReduce tasks at least near the data they operate on if it happens to be impossible to place the task at the same machine because that way the not- so-precious" in-rack bandwith is used.

## Elaborate HDFS concepts with suitable illustrations. (16) (Apr/May 17)

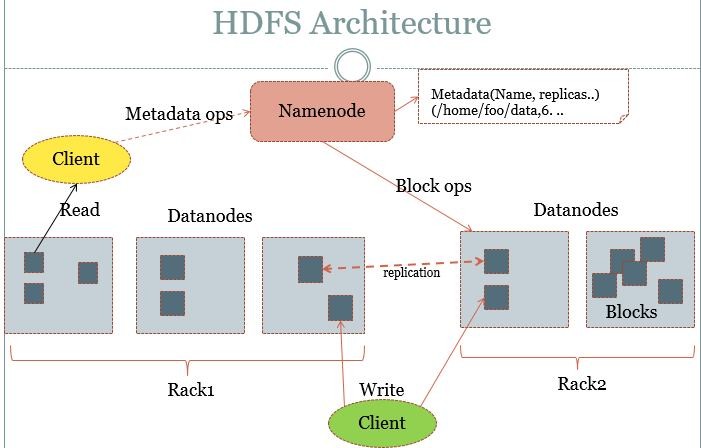
**Explain the Hadoop Distributed file system architecture with a diagram (16) (ND-18).**

# Blocks

* + A disk has a block size, which is the minimum amount of data that it can read or write. File system blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes.
  + HDFS has the concept of a block, but it is a much larger unit—64 MB by default.File0sin HDFS are broken into block-sized chunks, which are stored as independent units.

## Namenodes and Datanodes

* + An HDFS cluster has two types of node operating in a master-worker pattern: a *namenode* (the master) and a number of *datanodes* (workers). The namenode manages the file system namespace. It maintains the file system tree and the metadata for all the files and directories in the tree.
  + The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.



* + A *client* accesses the filesystem on behalf of the user by communicating with the namenode and datanodes. Datanodes are the workhorses of the file system. Hadoop can be configured so that the namenode writes its persistent state to multiple file systems.
  + These writes are synchronous and atomic. The usual configuration choice is to write to local disk as well as a remote NFS mount.
  + It is also possible to run a *secondary namenode*, which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large.

## HDFS Federation

* + The namenode keeps a reference to every file and block in the filesystem in memory, which means that on very large clusters with many files, memory becomes the limiting factor for scaling.
  + HDFS Federation, introduced in the 0.23 release series, allows a cluster to scale by adding namenodes, each of which manages a portion of the filesystem namespace.
  + For example, one namenode might manage all the files rooted under */user*, say, and a second Namenode might handle files under */share*.Under federation, each namenode manages a *namespace volume*, which is made up of the metadata for the namespace, and a *block pool* containing all the blocks for the files in the namespace.
  + Namespace volumes are independent of each other, which means namenodes do not communicate with one another, and furthermore the failure of one namenode does not affect the availability of the namespaces managed by other namenodes.

## HDFS High-Availability

* + The combination of replicating namenode metadata on multiple filesystems, and using

the secondary namenode to create checkpoints protects against data loss, but does not provide high-availability of the filesystem.

* + The namenode is still a *single point of failure* (SPOF), since if it did fail, all clients— including MapReduce jobs—would be unable to read, write, or list files, because the namenode is the sole repository of the metadata and the file-to-block mapping.
  + In such an event the whole Hadoop system would effectively be out of service until a new namenode could be brought online.

A few architectural changes are needed to allow this to happen:

* + - The namenodes must use highly-available shared storage to share the edit log.

When a standby namenode comes up it reads up to the end of the shared edit log to synchronize its state with the active namenode, and then continues to read new entries as they are written by the active namenode.

* + - Datanodes must send block reports to both namenodes since the block mappings are stored in a namenode’s memory, and not on disk.
    - Clients must be configured to handle namenode failover, which uses a mechanism that is transparent to users.
  + If the active namenode fails, then the standby can take over very quickly since it has the latest state available in memory: both the latest edit log entries, and an up-to-date block mapping.
  + The actual observed failover time will be longer in practice (around a minute or so), since the system needs to be conservative in deciding that the active namenode has failed

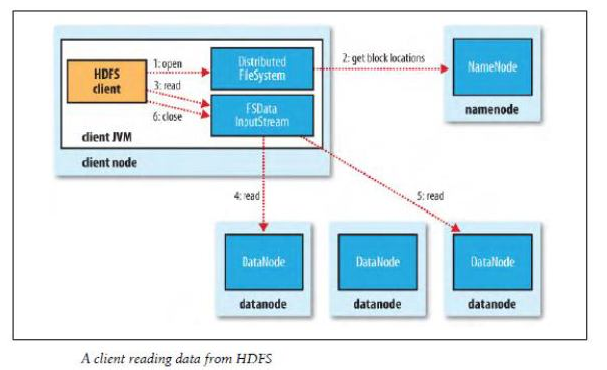
## Failover and fencing

* + The transition from the active namenode to the standby is managed by a new entity in the system called the*failover controller*.
  + Failover controllers are pluggable, but the first implementation uses ZooKeeper to ensure that only one namenode is active.
  + Each namenode runs a lightweight failover controller process whose job it is to monitor its namenode for failures and trigger a failover should a namenode fail.
  + Failover may also be initiated manually by an administrator, in the case of routine maintenance, for example.

## Illustrate data flow in HDFS during file read/write operation with suitable diagrams. (16) (Nov/Dec 17)

**FILE READ OPERATION**

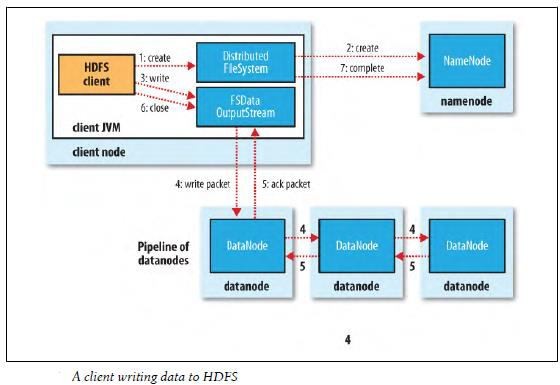
To get an idea of how data flows between the client interacting with HDFS, the namenode and the datanode, consider the below diagram, which shows the main sequence of events when reading a file.



* + The client opens the file it wishes to read by calling open() on the FileSystem object, which for HDFS is an instance of DistributedFileSystem (step 1).
  + DistributedFileSystem calls the namenode, using RPC, to determine the locations of the blocks for the first few blocks in the file (step 2).
  + For each block, the namenode returns the addresses of the datanodes that have a copy of that block. Furthermore, the datanodes are sorted according to their proximity to the client.
  + If the client is itself a datanode (in the case of a MapReduce task, for instance), then it will read from the local datanode.
  + The DistributedFileSystem returns a FSDataInputStream to the client for it to read data from. FSDataInputStream in turn wraps a DFSInputStream, which manages the datanode and namenode I/O. The client then calls read() on the stream (step 3).
  + DFSInputStream, which has stored the datanode addresses for the first few blocks in the file, then connects to the first (closest) datanode for the first block in the file. Data is streamed from the datanode back to the client, which calls read() repeatedly on the stream (step 4).
  + When the end of the block is reached, DFSInputStream will close the connection to the datanode, then find the best datanode for the next block (step 5).
  + This happens transparently to the client, which from its point of view is just reading a continuous stream. Blocks are read in order with the DFSInputStream opening new connections to datanodes as the client reads through the stream. It will also call the namenode to retrieve the datanode locations for the next batch of blocks as needed. When the client has finished reading, it calls close() on the FSDataInputStream (step 6).
  + One important aspect of this design is that the client contacts datanodes directly to retrieve data, and is guided by the namenode to the best datanode for each block. This design allows HDFS to scale to large number of concurrent clients, since the data traffic is spread across all the datanodes in the cluster.
  + The namenode meanwhile merely has to service block location requests (which it stores in memory, making them very efficient), and does not, for example, serve data, which would quickly become a bottleneck as the number of clients grew.

## FILE WRITE OPERATION

The case we’re going to consider is the case of creating a new file, writing data to it, and then closing the file



* + The client creates the file by calling create() on DistributedFileSystem (step 1). DistributedFileSystem makes an RPC call to the namenode to create a new file in the filesystem’s namespace, with no blocks associated with it (step 2).
  + The namenode performs various checks to make sure the file doesn’t already exist, and that the client has the right permissions to create the file. If these checks pass, the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException.
  + The DistributedFileSystem returns a SDataOutputStream for the client to start writing data to. Just as in the read case, FSDataOutputStream wraps a DFSOutputStream, which handles communication with the datanodes and namenode. Internal queue, called the *data queue*.
  + The data queue is consumed by the DataStreamer, whose responsibility it is to ask the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas. The list of datanodes forms a pipeline we’ll assume the replication level is 3, so there are three nodes in the pipeline.
  + The DataStreamer streams the packets to the first datanode in the pipeline, which stores the packet and forwards it to the second datanode in the pipeline. Similarly, the second datanode stores the packet and forwards it to the third (and last) datanode in the pipeline (step 4).
  + DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the *ack queue*. A packet is removed from the ack queue only when it has been acknowledged by all the datanodes in the pipeline (step 5).
  + If a datanode fails while data is being written to it, then the following actions are taken, which are transparent to the client writing the data. First the pipeline is closed, and any packets in the ack queue are added to the front of the data queue so that datanodes that are downstream from the failed node will not miss any packets.
  + The current block on the good datanodes is given a new identity, which is communicated to the namenode, so that the partial block on the failed datanode will be deleted if the failed datanode recovers later on.
  + The failed datanode is removed from the pipeline and the remainder of the block’s data is written to the two good datanodes in the pipeline.
  + The namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node. Subsequent blocks are then treated as normal.When the client has finished writing data it calls close() on the stream (step 6).
  + This action flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete (step7).
  + The namenode already knows which blocks the file is made up of (via Data Streamer asking for block allocations), so it only has to wait for blocks to be minimally replicated before returning successfully.

## What is GT4? Describe in detail the components of GT4 with a suitable diagram. (16) (Nov/Dec 17)

The Globus Toolkit 4 is composed of several software components. As shown in the following figure, these components are divided into five categories: Security, Data Management, Execution Management, Information Services, and the Common Runtime

## Common Runtime:

The common Runtime components provide a set of fundamental libraries and tools which are needed to build both WS and non-WS services

## Security:

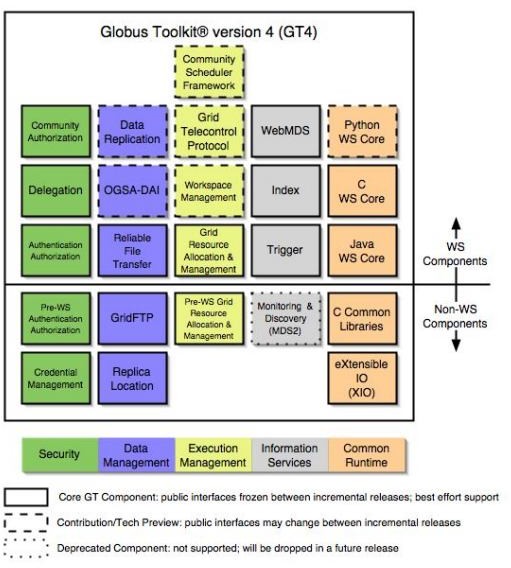
Using the security components, based on the grid security Infrastructure (GSI), we can make sure that our communications are secure.

## Data Management:

These components will allow us to manage large sets of data in our virtual organization.

## Information services:

The information services , more commonly referred to as the monitoring and Discovery services (MDS), includes a set of components to discover and monitor resources in virtual organization.Note that GT4 also includes a non-WS version of MDS for legacy purposes. This components is deprecated and will surely disappear in future release of the toolkit.

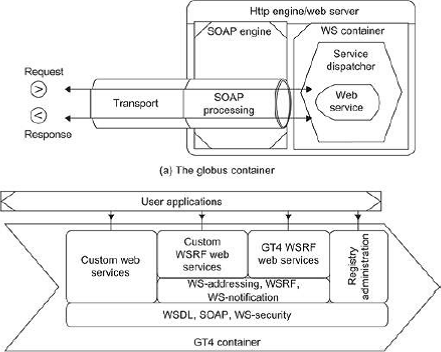


## Execution management:

Execution management components deal with the initiation, monitoring, management, scheduling and coordination of executable programs, usually called jobs, in a grid Three containers are used to host user-developed services written in Java, Python, and C, respectively.

These containers provide implementations of security, management, discovery, state management, and other mechanisms frequently required when building services.

They extend open source service hosting environment with support for a range of useful web services specifications, including WSRF, WS- Notification, and WS- Security



.

A set of client libraries allow client programs in java, c, and python to invoke operations on both GT$ and user developed services . In many cases , Multiple interfaces provide different levels of control :

For Example , IUn case of Grid FTP, there is not only a simple command –line client but also control and data channel libraries for use in programs and the XIO library allowing for the integration of alternatives transports. The use of uniform abstractions and mechanisms means clients can interact with different services in similar ways , which facilitates construction of complex , interoperable system s and encourages code reuse.

1. **List the characteristics of Globus tool kit. With a neat sketch describe the architecture of Globus GT4 and the services offered. (16) (Apr/May 18)**
   * The Globus Toolkit is an open middleware library for the grid computing communities. These open source software libraries support many operational grids and their applications on an international basis.
   * The toolkit addresses common problems and issues related to grid resource discovery, management, communication, security, fault detection, and portability. The software itself provides a variety of components and capabilities.
   * The library includes a rich set of service implementations. The implemented software supports grid infrastructure management, provides tools for building new web services in Java, C, and Python, builds a powerful standard-based.
   * Security infrastructure and client APIs (in different languages), and offers comprehensive command-line programs for accessing various grid services.
   * The Globus Toolkit was initially motivated by a desire to remove obstacles that prevent seamless collaboration, and thus sharing of resources and services, in scientific and engineering applications.
   * The shared resources can be computers, storage, data, services, networks, science instruments (e.g., sensors), and so on.

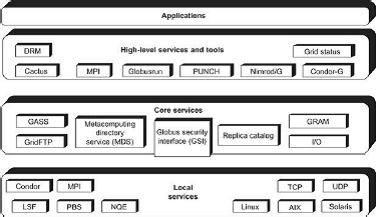


Figure: Globus Toolkit GT4 supports distributed and cluster computing services

# The GT4 Library

The GT4 Library GT4 offers the middle-level core services in grid applications. The high-level services and tools, such as MPI, Condor-G, and Nirod/G, are developed by third parties for general purpose distributed computing applications. The local services, such as LSF, TCP, Linux, and Condor, are at the bottom level and are fundamental tools supplied by other developers.

# Globus Job Workflow

* + A typical job execution sequence proceeds as follows: The user delegates his credentials to a delegation service. The user submits a job request to GRAM with the delegation identifier as a parameter.
  + GRAM parses the request, retrieves the user proxy certificate from the delegation service, and then acts on behalf of the user. GRAM sends a transfer request to the RFT, which applies GridFTP to bring in the necessary files.
  + GRAM invokes a local scheduler via a GRAM adapter and the SEG initiates a set of user jobs. The local scheduler reports the job state to the SEG. Once the job is complete, GRAM uses RFT and GridFTP to stage out the resultant files.

# Client-Globus Interactions

* + There are strong interactions between provider programs and user code. GT4 makes heavy use of industry-standard web service protocols and mechanisms in service
  + Description, discovery, access, authentication, authorization, and the like. GT4 makes extensive use of Java, C, and Python to write user code. Web service mechanisms define specific interfaces for grid computing. Web services provide flexible, extensible, and widely adopted XML-based interfaces.

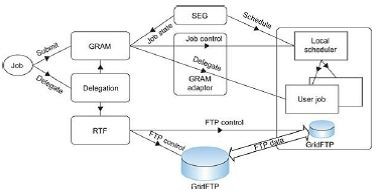


Figure: Globus job workflow among interactive functional modules.

These demand computational, communication, data, and storage resources. We must enable a range of end-user tools that provide the higher-level capabilities needed in specific user applications. Developers can use these services and libraries to build simple and complex systems quickly.

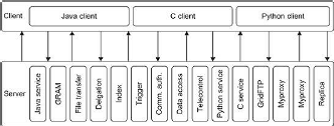


Figure: Client and GT4 server interactions

vertical boxes correspond to service programs and horizontal boxes represent the user codes. The horizontal boxes in the client domain denote custom applications and/or third-

party tools that access GT4 services. The toolkit programs provide a set of useful infrastructure services.

Three containers are used to host user-developed services written in Java, Python, and C, respectively. These containers provide implementations of security, management, discovery, state management, and other mechanisms frequently required when building services.

1. **with an illustration, emphasize the significance of MapReduce paradigm in Hadoop framework. List out the assumption and goals set in HDFS architecture for processing the data based on divide-and-conquer strategy.(16) (Apr/May 18)**

The model is based on two distinct steps for an application:

* **Map**: An initial ingestion and transformation step, in which individual input records can be processed in parallel.
* **Reduce**: An aggregation or summarization step, in which all associated records must be processed together by a single entity.
  + The core concept of MapReduce in Hadoop is that input may be split into logical chunks, and each chunk may be initially processed independently, by a map task.
  + The results of these individual processing chunks can be physically partitioned into distinct sets, which are then sorted. Each sorted chunk is passed to a reduce task.
  + A map task may run on any compute node in the cluster, and multiple map tasks may be running in parallel across the cluster. The map task is responsible for transforming the input records into key/value pairs.
  + The output of all of the maps will be partitioned, and each partition will be sorted. There will be one partition for each reduce task. Each partition’s sorted keys and the values associated with the keys are then processed by the reduce task. There may be multiple reduce tasks running in parallel on the cluster.
  + The application developer needs to provide only four items to the Hadoop framework: the class that will read the input records and transform them into one key/value pair per record, a map method, a reduce method, and a class that will transform the key/value pairs that the reduce method outputs into output records.
  + My first MapReduce application was a specialized web crawler. This crawler received as input large sets of media URLs that were to have their content fetched and processed. The media items were large, and fetching them had a significant cost in time and resources.

The job had several steps:

1. Ingest the URLs and their associated metadata.
2. Normalize the URLs.
3. Eliminate duplicate URLs.
4. Filter the URLs against a set of exclusion and inclusion filters.
5. Filter the URLs against a do not fetch list.
6. Filter the URLs against a recently seen set.
7. Fetch the URLs.
8. Fingerprint the content items.
9. Update the recently seen set.
10. Prepare the work list for the next application.

# Introducing Hadoop

Hadoop is the Apache Software Foundation top-level project that holds the various Hadoop subprojects that graduated from the Apache Incubator. The Hadoop project provides and supports the development of open source software that supplies a framework for the development of highly scalable distributed computing applications. The Hadoop framework handles the processing details, leaving developers free to focus on application logic.

The introduction on the Hadoop project web page states:

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing, including:

**Hadoop** Core, our flagship sub-project, provides a distributed filesystem (HDFS) and support for the MapReduce distributed computing metaphor.

HBase builds on Hadoop Core to provide a scalable, distributed database.

**Pig** is a high-level data-flow language and execution framework for parallel computation.It is built on top of Hadoop Core.

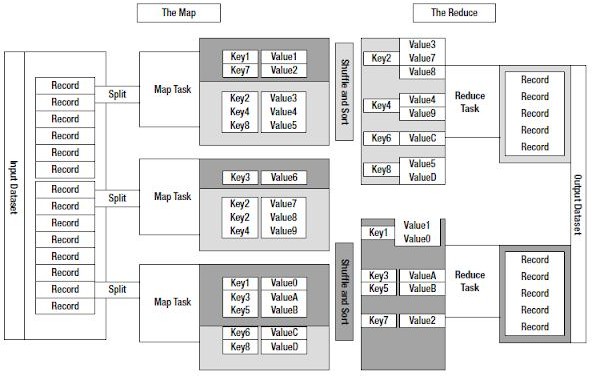


Figure: The MapReduce model

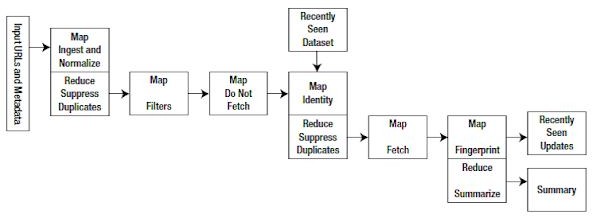
**ZooKeeper** is a highly available and reliable coordination system. Distributed applications use ZooKeeper to store and mediate updates for critical shared state.

**Hive** is a data warehouse infrastructure built on Hadoop Core that provides data summarization, adhoc querying and analysis of datasets.

The Hadoop Core project provides the basic services for building a cloud computing environment with commodity hardware, and the APIs for developing software that will run on that cloud.

The two fundamental pieces of Hadoop Core are the MapReduce framework, the cloud computing environment, and he Hadoop Distributed File System (HDFS).

* The Hadoop Core MapReduce framework requires a shared file system. This shared file system does not need to be a system-level file system, as long as there is a distributed file system plug-in available to the framework.
* The Hadoop Core framework comes with plug-ins for HDFS, CloudStore, and S3. Users are also free to use any distributed file system that is visible as a system-mounted file system, such as Network File System (NFS), Global File System (GFS), or Lustre.



The Hadoop Distributed File System (HDFS)MapReduce environment provides the user with a sophisticated framework to manage the execution of map and reduce tasks across a cluster of machines.

The user is required to tell the framework the following:

* The location(s) in the distributed file system of the job input
* The location(s) in the distributed file system for the job output
* The input format
* The output format
* The class containing the map function
* Optionally. the class containing the reduce function
* The JAR file(s) containing the map and reduce functions and any support classes

The final output will be moved to the output directory, and the job status will be reported to the user. MapReduce is oriented around key/value pairs. The framework will convert each record of input into a key/value pair, and each pair will be input to the map function once. The map output is a set of key/value pairs—nominally one pair that is the transformed input pair. The map output pairs are grouped and sorted by key. The reduce function is called one time for each key, in sort sequence, with the key and the set of values that share that key. The reduce method may output an arbitrary number of key/value pairs, which are written to the output files in the job output directory. If the reduce output keys are unchanged from the reduce input keys, the final output will be sorted. The framework provides two processes that handle the management of MapReduce jobs:

* + TaskTracker manages the execution of individual map and reduce tasks on a compute node in the cluster.
  + JobTracker accepts job submissions, provides job monitoring and control, and manages the distribution of tasks to the TaskTracker nodes.

The JobTracker is a single point of failure, and the JobTracker will work around the failure of individual TaskTracker processes.

The Hadoop Distributed File System

HDFS is a file system that is designed for use for MapReduce jobs that read input in large chunks of input, process it, and write potentially large chunks of output. HDFS does not handle random access particularly well. For reliability, file data is simply mirrored to multiple storage nodes. This is referred to as replication in the Hadoop community. As long as at least one replica of a data chunk is available, the consumer of that data will not know of storage server failures.

HDFS services are provided by two processes:

* NameNode handles management of the file system metadata, and provides management and control services.
* DataNode provides block storage and retrieval services.

There will be one NameNode process in an HDFS file system, and this is a single point of failure. Hadoop Core provides recovery and automatic backup of the NameNode, but no hot failover services. There will be multiple DataNode processes within the cluster, with typically one DataNode process per storage node in a cluster.