MAKING HADOOP HIGHLY AVAILABLE

Using an alternative File system – HP IBRIX

Johannes Kirschnick, Steve Loughran
June 2010

SOMETHING ABOUT ME

- I work at HP Labs, Bristol, UK
  - Degree in computer science, TU Munich
- Automated Infrastructure Lab
  - Automated, secure, dynamic instantiation and management of cloud computing infrastructure and services
- Personal interest
  - Cloud Services
  - Automated service deployment
  - Storage Service

WHAT DO I WANT TO TALK ABOUT

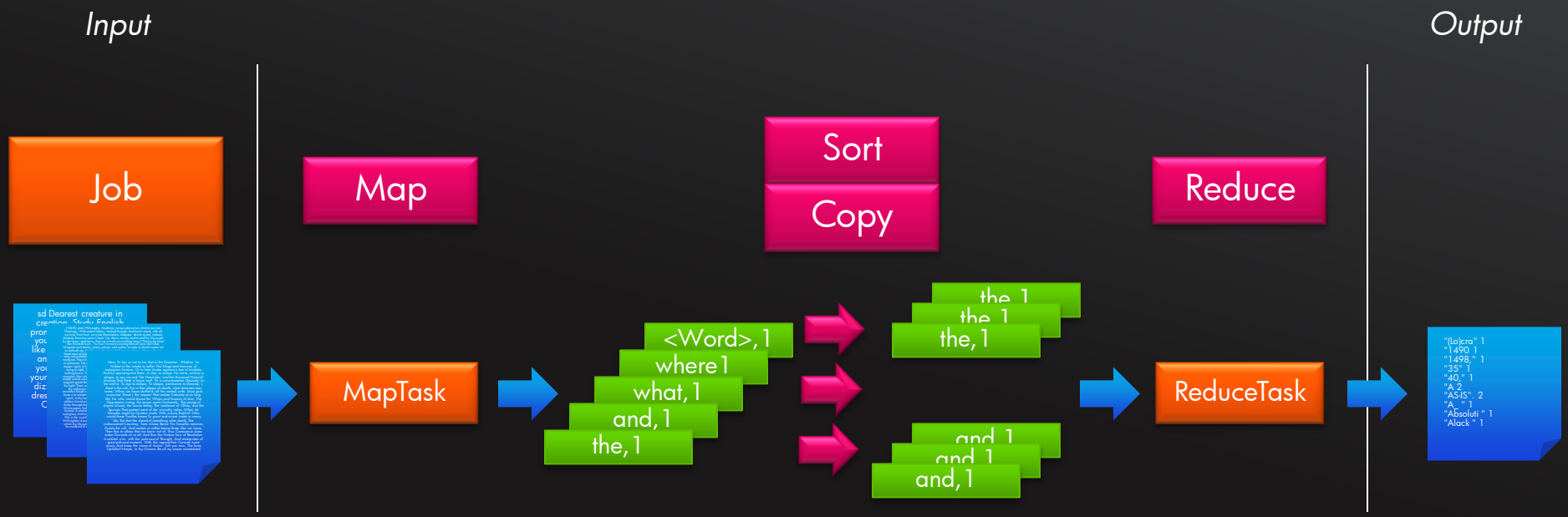
- Motivate High Availability, introduce the context
- Overview about Hadoop
- Highlight the Hadoop modes of failure operation
- Introduce HP IBRIX
- Performance Results
- Summary

CONTEXT OF THIS TALK

- High availability
  - Continued availability in times of failures
    - Hadoop Service
    - Data operated on
- Fault tolerant operation
  - What happens if a node dies
- Reduce time to restart

HADOOP IN A NUTSHELL

Example: Wordcount across a number of documents



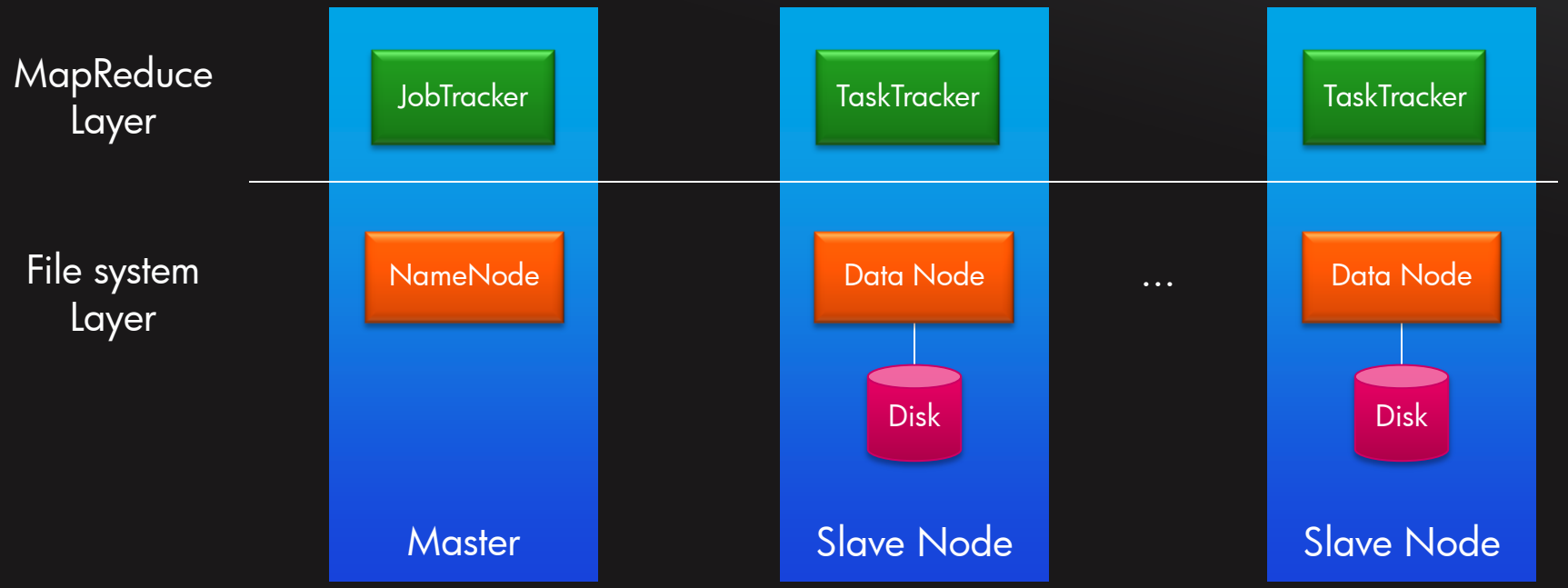

```
map(name, document) {
  for each word w in document:
    emitIntermediate(w, 1)
}
```

```
reduce(key, values ...) {
  count = 0
  for each value v in values:
            count+=v
   emit(key,count)
}
```

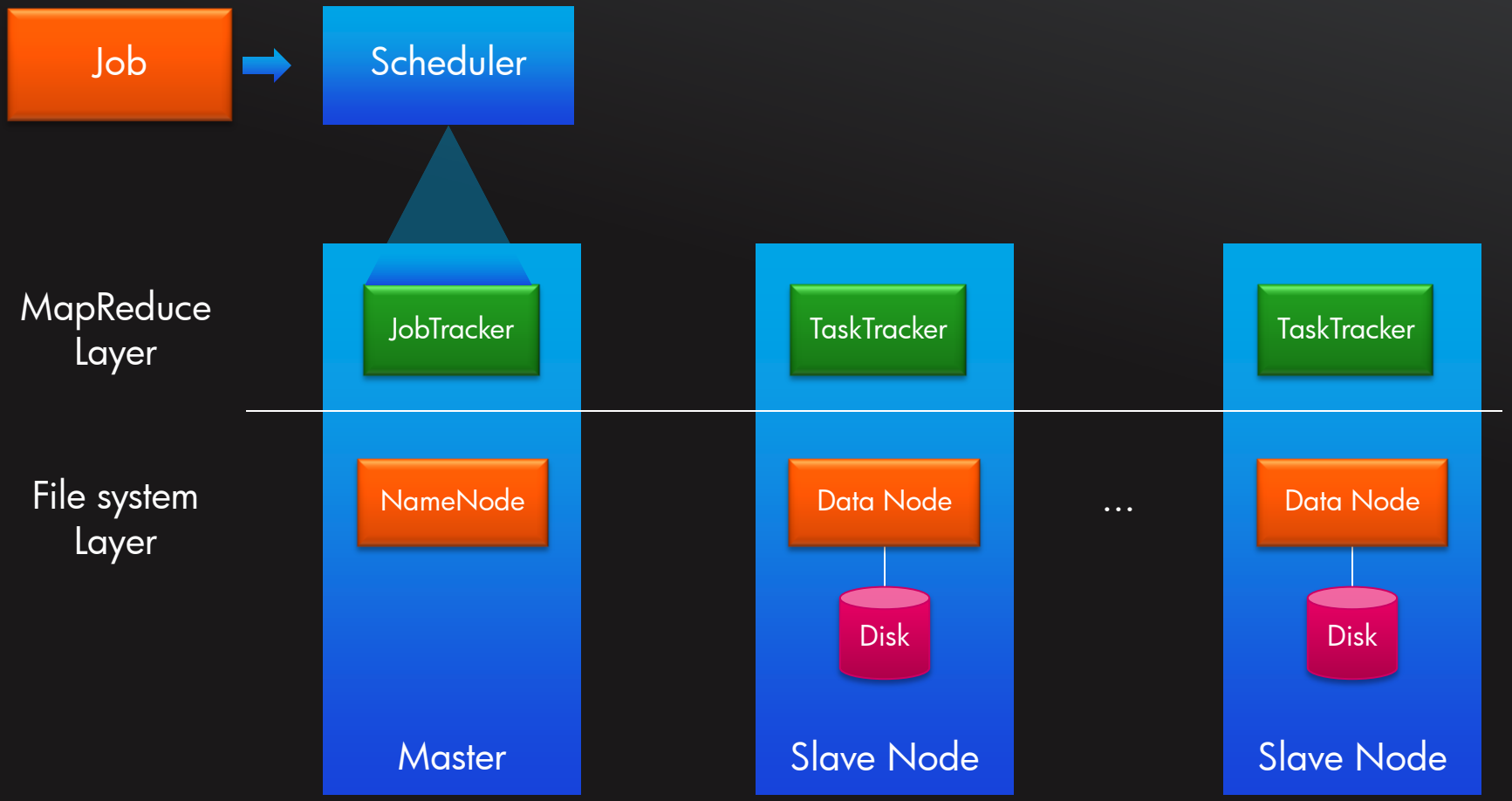
HADOOP COMPONENTS

- Map Reduce Layer
  - Provides the map and reduce programming framework
  - Can break up Jobs into tasks
  - Keeps track of execution status
- File system Layer
  - Pluggable file system
  - Support for location aware file systems
  - Access through an API Layer
  - Default is HDFS (Hadoop Distributed File system)
  - HDFS
    - Provides fault high availability by replicating individual files
    - Consists of a central metadata server – NameNode
    - And a number of Data nodes, which store copies of files (or parts of them)

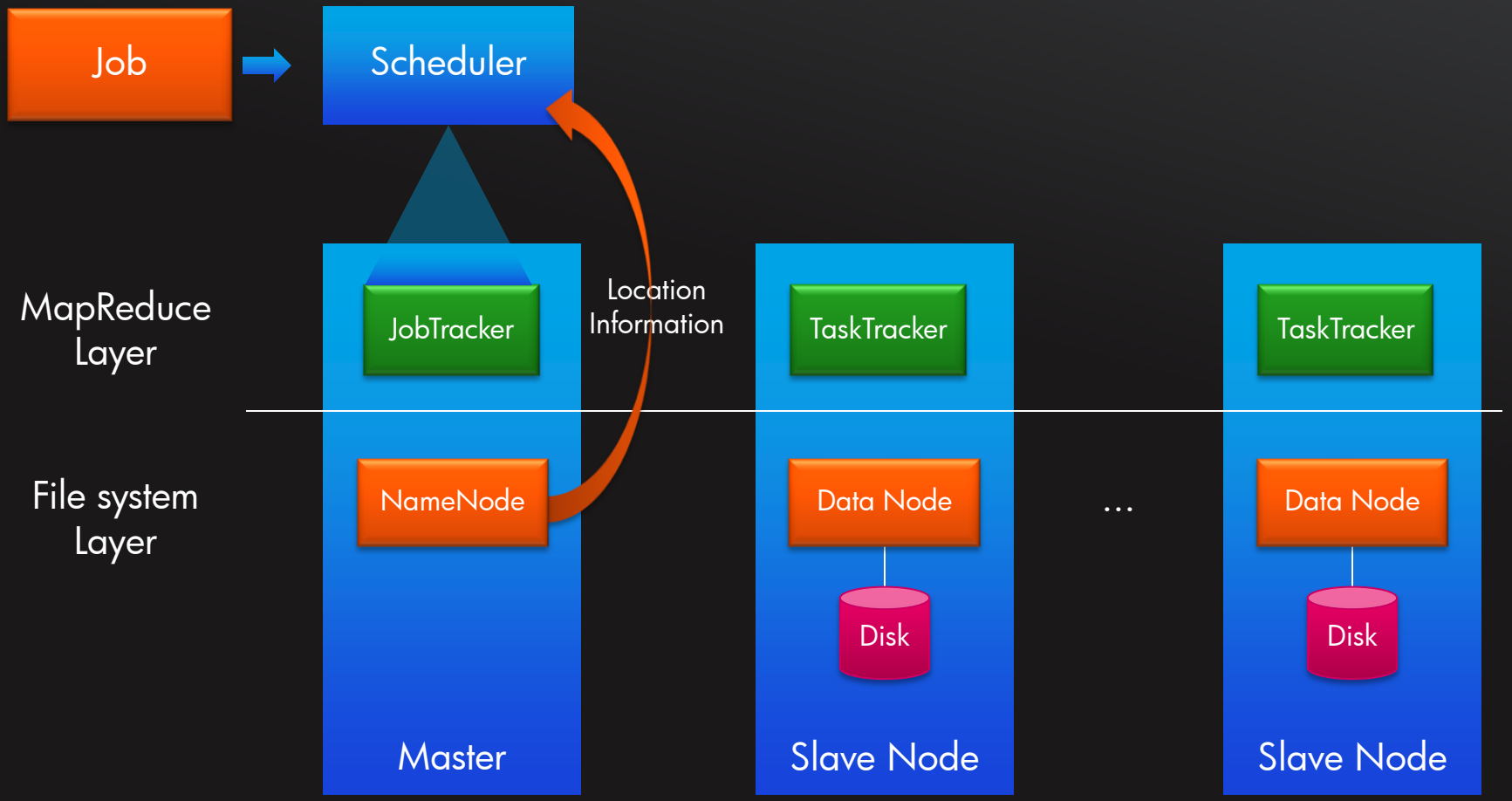
HADOOP OPERATION (WITH HDFS)



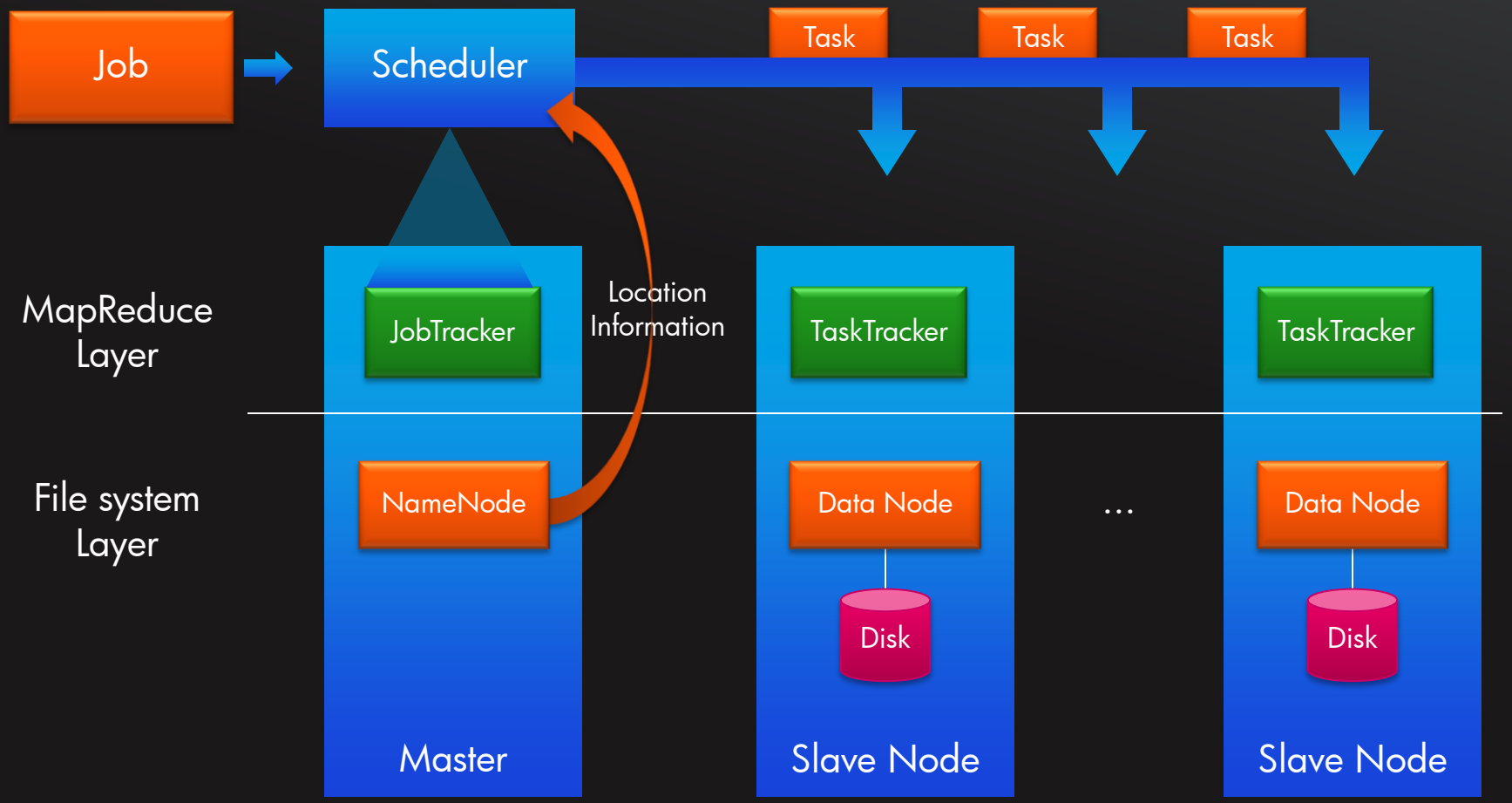
HADOOP OPERATION (WITH HDFS)



HADOOP OPERATION (WITH HDFS)



HADOOP OPERATION (WITH HDFS)



FAILURE SCENARIOS AND RESPONSES

Failure in Map Reduce components

- TaskTracker
  - Sends heartbeat to JobTracker
  - If unresponsive for x seconds, JobTracker marks TaskTracker as dead and stop assigning work to it
  - Scheduler reschedules tasks running on that TaskTracker
- JobTracker
  - No build in heartbeat mechanism
  - Checkpoints to filesystem
  - Can be restarted and resumes operation
- Individual Tasks
  - TaskTracker monitors progress
  - Can restart failed Tasks
  - Complex failure handling
    - E.g. skip parts of input data which produces failure

FAILURE SCENARIOS AND RESPONSES (2)

Failure of Data storage

- Pluggable file system implementation needs to detect and remedy error scenarios
- HDFS
  - Failure of Data Node
    - Keeps track of replication count for files (parts of files)
    - Can re-replicate missing pieces
    - Tries to place copies of individual physically apart from each other
      - Same rack vs. different racks
  - Failure of NameNode
    - Operations are written to logs, makes restart possible
      - During restart the filesystem is in read only mode
    - A secondary NameNode can periodically read these logs, to speed up time to become available
    - BUT
      If secondary namenode takes over, restart of the whole cluster is needed, since assigned hostnames have changed.

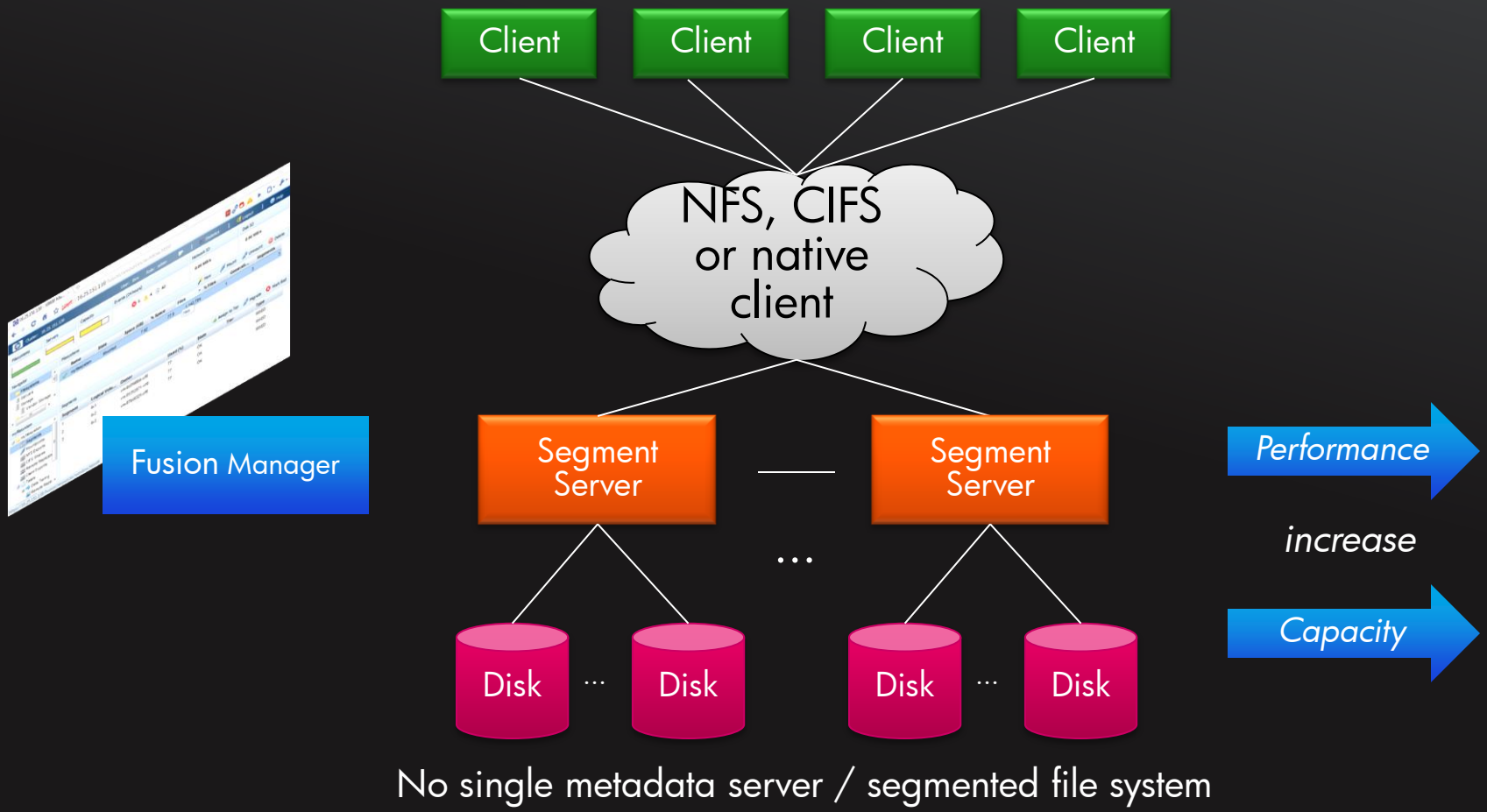
AVAILABILITY TAKEAWAY

- Map reduce Layer
  - Checkpoints to the persisting file system to resume work
  - TaskTracker
    - Can be restarted
  - JobTracker
    - Can be restarted
- HDFS
  - Single point of failure is the NameNode
    - Restarts can take a long time, depending on amount of data stored and number of operations in the log itself
    - In the regions of hours

A DIFFERENT FILE SYSTEM

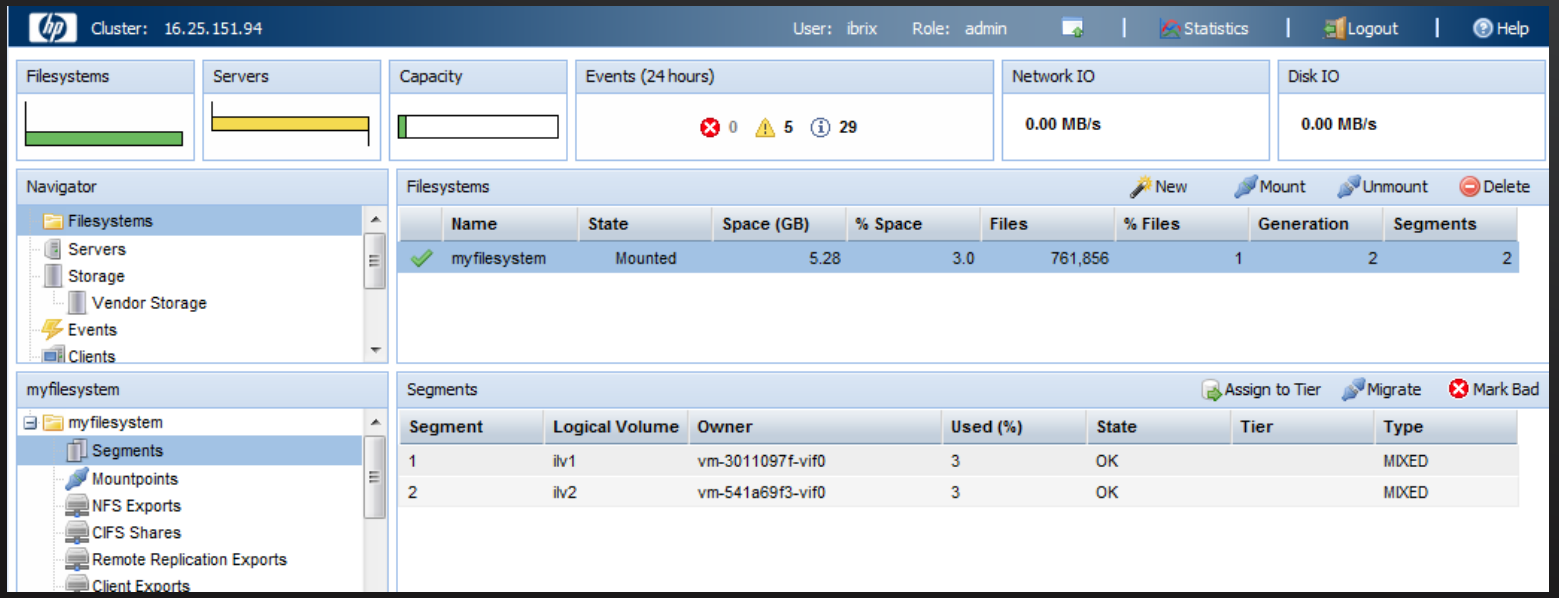
- HP IBRIX
- Software solution which runs on top of storage configurations
- Fault tolerant, high availability file system
- Segmented File system
  - Disks (Luns) are treated as Segments
- Segments are managed by Segment servers
  - Aggregated into global file system(s)
  - File systems provide single namespace
  - Each file system supports up to 16 Petabyte

IBRIX IN A NUTSHELL



No single metadata server / segmented file system

HOW DOES IT LOOK LIKE



- Fusion Manager Web Console
  - Based on command line interface
- Global management view of the installation
- Here segments correspond to disks attached to servers

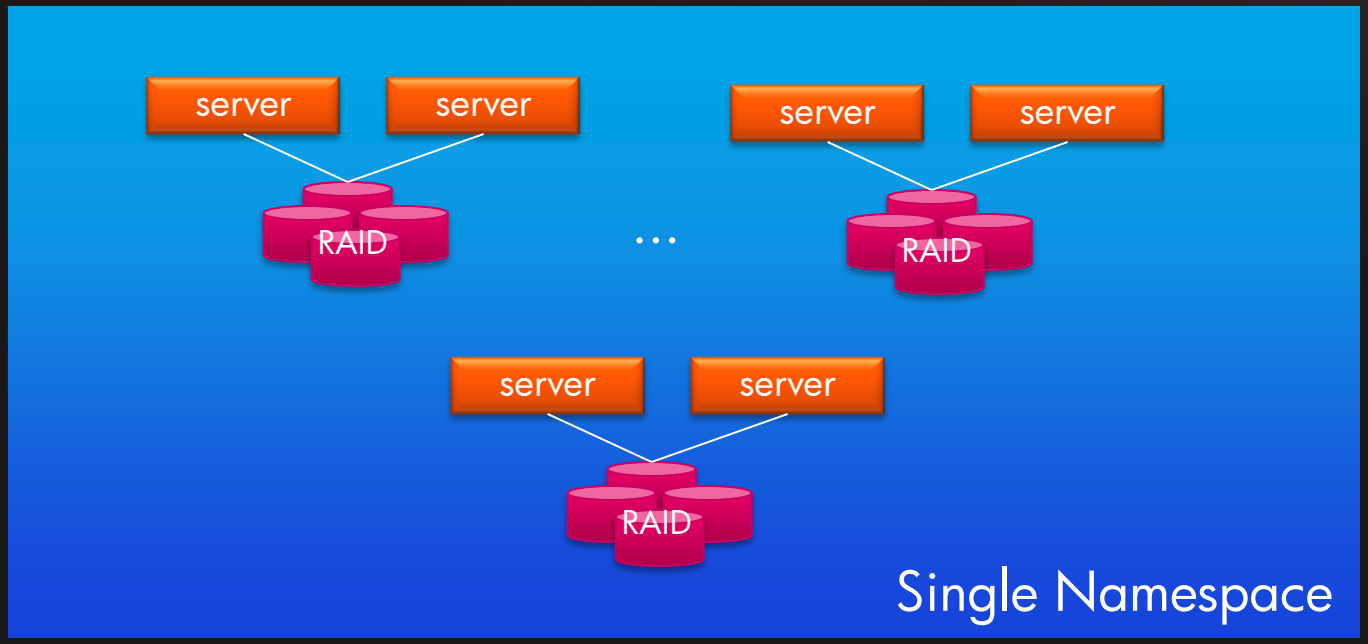
HOW DOES IT LOOK LIKE (2)

- A client simply mounts the file system via:
  - NFS
  - CIFS / Samba
  - Native Client
  - Each segment server is automatically a client
- Mount points and exports need to be created firsts
  - on the fusion manager
- Clients access file system via “normal” file system calls

```
[root@vm-18f07295-vif0 bin]# ./get_seg /myfilesystem/input-data/*
seg#             inode #       file offset        local size   version
   2           200000018                 0         117111196      3600 /myfilesystem/input-data/part-00000
   3           30000001a                 0         117111196      3591 /myfilesystem/input-data/part-00001
   4           400007c05                 0         117111196      3591 /myfilesystem/input-data/part-00002
   1           100000015                 0         117111196      3591 /myfilesystem/input-data/part-00003
```

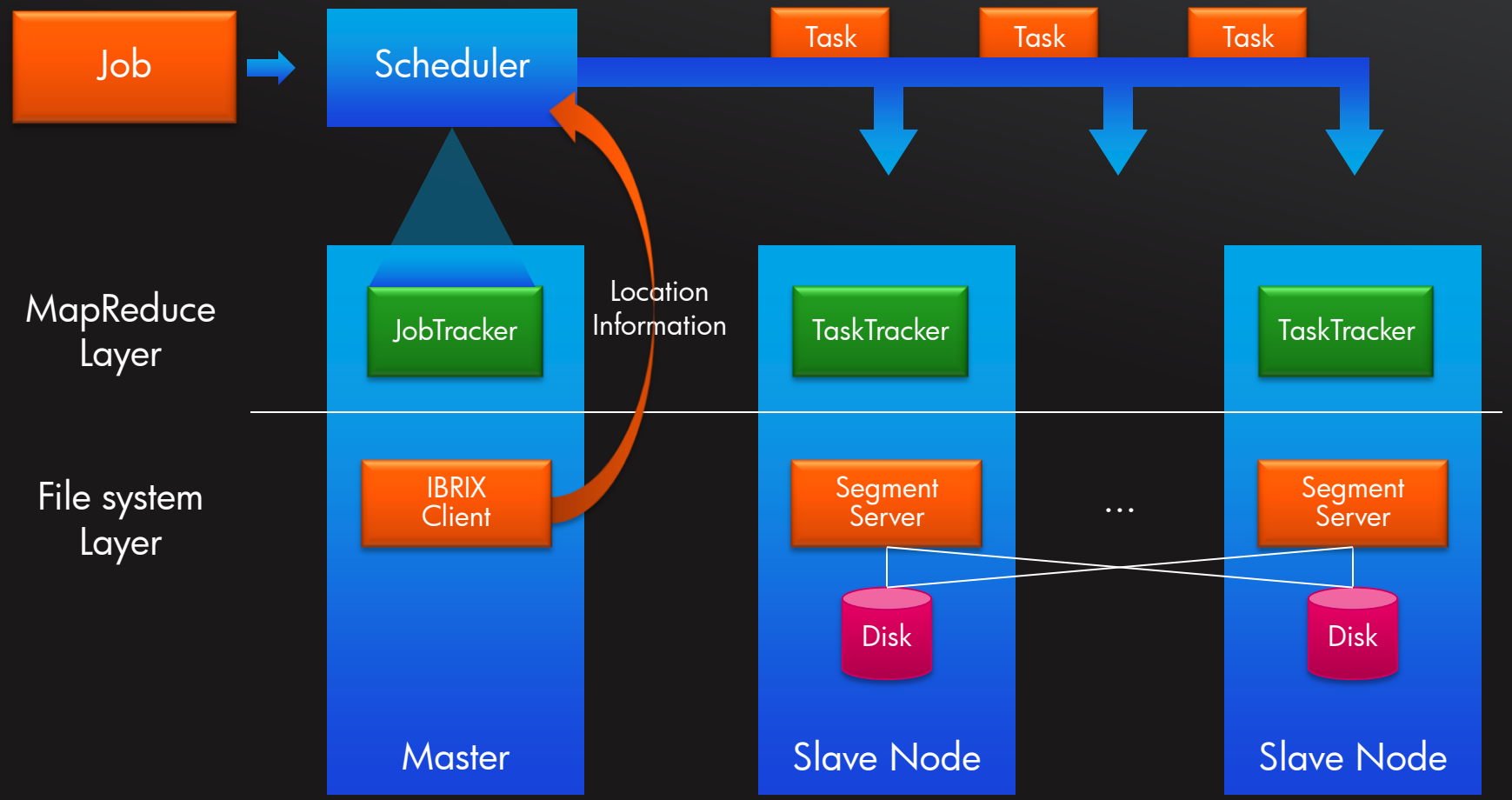
FAULT TOLERANT

- Supports failover
- Different hardware topologies configurations
- Couplet configuration
  - Best suited for hybrid of performance and capacity



Location aware Hadoop on IBRIX

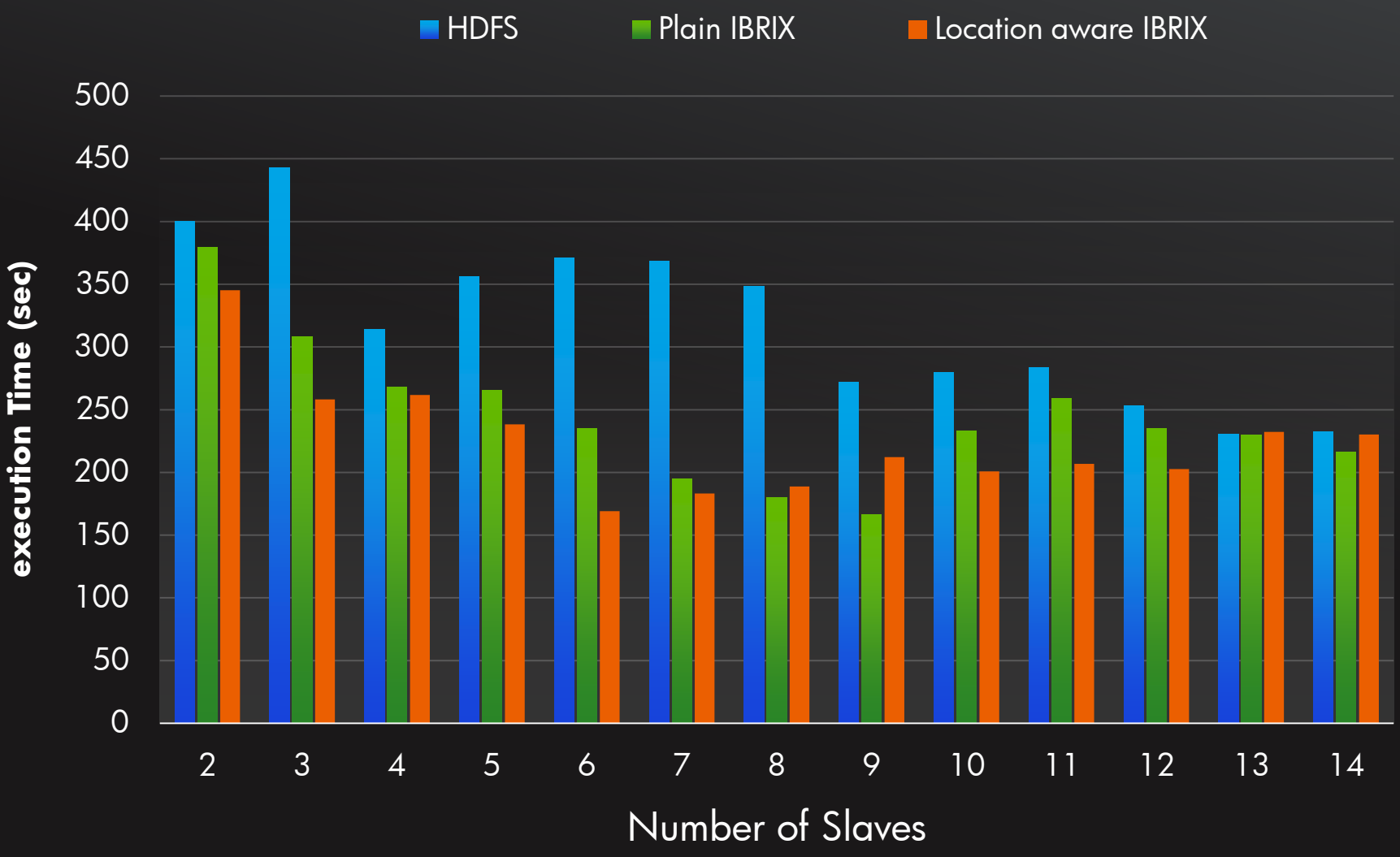
HADOOP INTERNALS – WITH IBRIX



PERFORMANCE TEST

- 1 GB of randomly generated data, spread across 10 input files *RandomWriter*
- Use Hadoop *Sort* to sort the records, measure time spend sorting
  - Includes mapping, sorting and reducing time
- Vary the number of slave nodes
- File access test
  - Actual computation on each TaskTracker is low
  - Governing factors for execution time are
    - Time to read and write files
    - Time to distribute data to the reducers

PERFORMANCE RESULTS



PERFORMANCE RESULTS

- Comparable performance to native HDFS system
  - For smaller workload even increased performance – due to no replication
- Can take advantage of location information
- Is dependent on distribution and type of input data
  - Across the segment servers
- Prefers many smaller files, since they can be distributed better

FURTHER FEATURE COMPARISON

FEATURE	HDFS	IBRIX
Single Point of Failure	Yes, namenode	No
Needs RAID	No, replicates	Yes
Can expose location information	Yes	Yes
Individual file replication	Yes	No, only complete filesystems
Respond to node failure	Re-Replication mark as dead	Failover mark as dead, can fallback
Homogenous file system	Yes	No, can define Tiers
Split files across nodes	Yes - files are split into chunks which are distributed individually	Only if a segment is full

SUMMARY

SUMMARY

- Hadoop provides a number of failure handling methods
  - Dependent on persistent file system
- IBRIX as alternative file system
  - Not specifically build for Hadoop
    - Light weight file system plug-in for Hadoop
  - Location aware design enables computation close to the data
  - Comparable performance while gaining on fault tolerance
  - Fault tolerance persistence – no single point of failure
  - Reduced storage requirement
  - Storage not exclusive to Hadoop
- Future work
  - Making the JobTracker failure independent
  - Moving into a virtual environment
  - Short lived Hadoop Cluster

Q&A

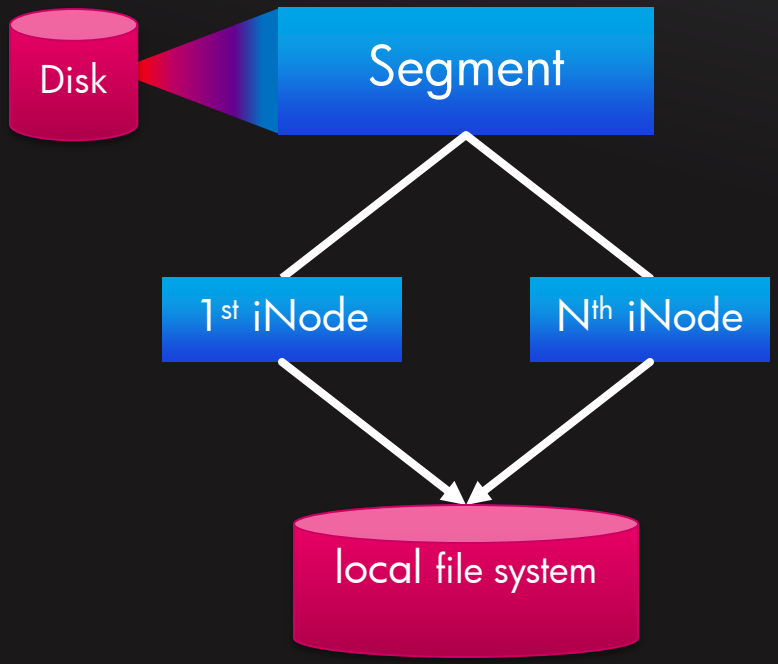
BACKUP

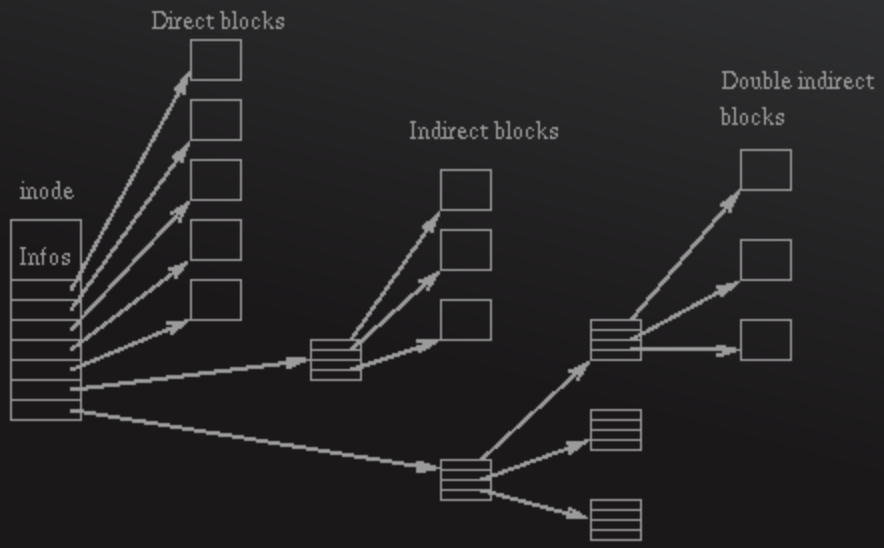
IBRIX DETAILS

- IBRIX uses iNodes as backend store
- Extends them by a file-based globally unique identifier
- Each Segment server is responsible for a fixed number of iNodes)
  - Determined by blocksize within that segment and overall size
  - Example
    - 4 GB segment size, 4kb block size → 1,048,576 iNodes (1M)
  - Simplified calculation example
    - Where is iNode 1,800,000
    - divide by 1M ≈ 1.71 → lives on segment server 1
- iNodes do not store the data but have a reference to the actual data
  - Backend storage for iBrix is ext3 filesystem

MORE DETAILS

– Based on distributed iNodes





SECURITY

- File system respects POSIX like interface
  - Files belong to user/group and have read/write/execute flags
- Native Client
  - Needs to be bound to a Fusion Manager
  - Export control can be enforced
    - Mounting only possible from the Fusion manager console
- CIFS / Samba
  - Requires Active Directory to translate windows ids to Linux id
  - Export only sub path of the file system (e.g. /filesystem/sambadirectory)
- NFS
  - Create exports on Segment server
  - Limit clients by IP Mask
  - Export only sub path of the file system (e.g. /filesystem/nfsdirectory)
  - Normal NFS properties (read/write/root squash)

FEATURES

- Multiple logical file systems
  - Select different segments as base for them
- Task Manager / Policy
  - Rebalancing between different segment servers
  - Tiering of data
    - Some segments could be better/worse than others
    - Move data to from them based on policy/rule
  - Replicate complete logical file systems - Replicate to remote cluster
- Failover
  - Buddy system of two (or more) segment servers (active/active standby)
  - Native clients will automatically failover
- Growing
  - Segment servers register with Fusion Manager
  - New segments (discs) need to be programmatically discovered
    - Can be added to logical file systems
- Is location aware
  - By nature of design
  - For each file, the segment server(s) where it is stored can be determined

FEATURES (2)

- De-duplication
- Caching
  - On segment server owning a particular file
- Distributed Metadata
  - No single point of failure
- Supports snap shooting of whole file systems
  - Creates a new virtual file system
- Policy for storing new files
  - Distribute them randomly across segment servers
  - assign them to the “local” segment server
- Separate data network
  - Allows to configure the network interface to use for storage communication

