

# TidyFS: A Simple and Small Distributed Filesystem

Dennis Fetterly<sup>1</sup>, Maya Haridasan<sup>1</sup>, Michael  
Isard<sup>1</sup>, and Swaminathan Sundararaman<sup>2</sup>

<sup>1</sup>Microsoft Research, Silicon Valley

<sup>2</sup>University of Wisconsin, Madison

# Introduction

- Increased use of shared nothing clusters for Data Intensive Scalable Computing (DISC)
- Programs use a data-parallel framework
  - MapReduce, Hadoop, Dryad
  - PIG, HIVE, or DryadLINQ

# DISC Storage Workload Properties

- Data stored in streams striped across cluster machines
- Computations parallelized so each part of a stream is read sequentially by a single process
- Each stream is written in parallel
  - each part is written sequentially by a single writer
- Frameworks re-execute sub-computations when machines or disks are unavailable

# TidyFS Design Goals

- Targeted only to DISC storage workload
  - Exploit this for simplicity
  - Data invisible until committed, then immutable
- Rely on fault-tolerance of the framework
  - Enables lazy replication

# TidyFS Data Model

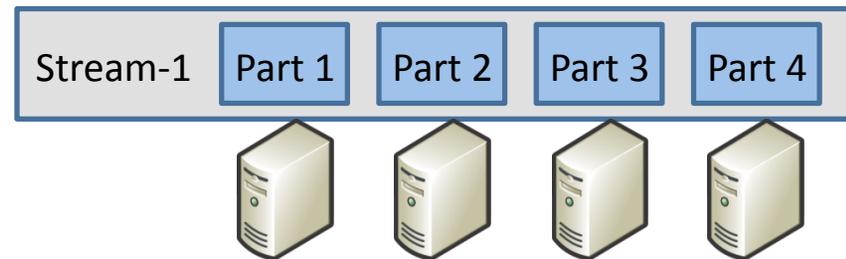
- Data
  - Stored as blobs on compute nodes
  - Immutable once written
- Metadata
  - Stored in centralized, reliable component
  - Describe how datasets are formed from data blobs
  - Mutable

# Client Visible Objects

- Stream: a sequence of parts
  - i.e. tidyfs://dryadlinquers/fetterly/clueweb09-English
  - Names imply directory structure

- Part:

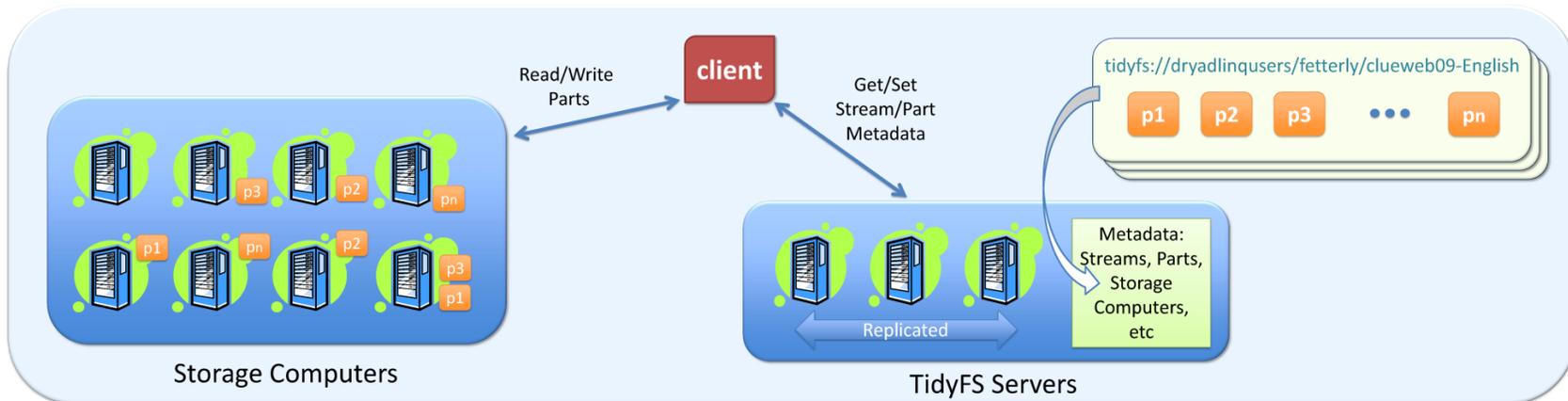
- Immutable
- 64 bit unique identifier
- Can be a member of multiple streams
- Stored on cluster machines
- Multiple replicas of each part can be stored



# Part and Stream Metadata

- System defined
  - Part: length, type, and fingerprint
  - Stream: name, total length, replication factor, and fingerprint
- User defined
  - Key-value store for arbitrary named blobs
  - Can describe stream compression or partitioning scheme

# TidyFS System Architecture



- Metadata server
- Node Service
- TidyFS Explorer

# Metadata Server

- Maintains metadata for the system
  - Maps streams to parts
  - Maps parts to storage machine and data path
    - NTFS file, SQL table
  - Contains stream and part metadata
  - Maintains machine state
  - Schedules part replication and load balancing
- Replicated for scalability and fault tolerance

# Node Service

- Runs on each storage machine
- Garbage Collection
  - Delete parts that have been removed from TidyFS server (i.e. parts from deleted streams)
  - Verify machine has all parts expected by TidyFS server to ensure correct replica count
- Replication
  - TidyFS server provides list of part ids to replicate
  - Machine replicates partition to local filesystem
- Validation
  - Validate checksum of stored parts

# TidyFS Explorer

The screenshot shows the TidyFS Explorer application window. The interface is divided into several sections:

- Directory View:** A tree view on the left showing the directory structure. The selected path is `tidyfs://dryadlinquers/fetterly/clueweb09/english-1-aa`.
- Stream List:** A table in the center showing a list of streams with their names and sizes. The selected stream is `english-1-aa` with a size of 452.39 GB.
- Partition List:** A table on the right showing a list of partitions with their indices, identifiers, and sizes. The selected partition is index 1 with identifier `00000000056BBA4` and a size of 144.85 MB.
- Metadata:** A section at the bottom left showing detailed information for the selected stream, including its size, replication factor, lease expiration time, creation and last access times, fingerprint, and the number of partitions.

Stream Name	Size
clueweb-English-catA-anchors.txt	10.42 MB
clueweb-English-catA-anchors.txt.dl	857.00 B
clueweb-English-catA-doc-url-hashco	849.00 B
clueweb-English-catA-pruned-anchor	563.01 MB
clueweb-English-catA-pruned-anchor	852.00 B
clueweb-English-catA-text.txt	823.42 GB
clueweb-English-catA-text.txt.dl	852.00 B
clueweb-english-docidurlmap-cond.t	12.61 GB
clueweb-english-docidurlmap-cond.t	1.82 KB
<b>english-1-aa</b>	<b>452.39 GB</b>
english-1-ab	451.83 GB
english-2-aa	522.49 GB
english-2-ab	508.94 GB
webgraph	42.75 GB

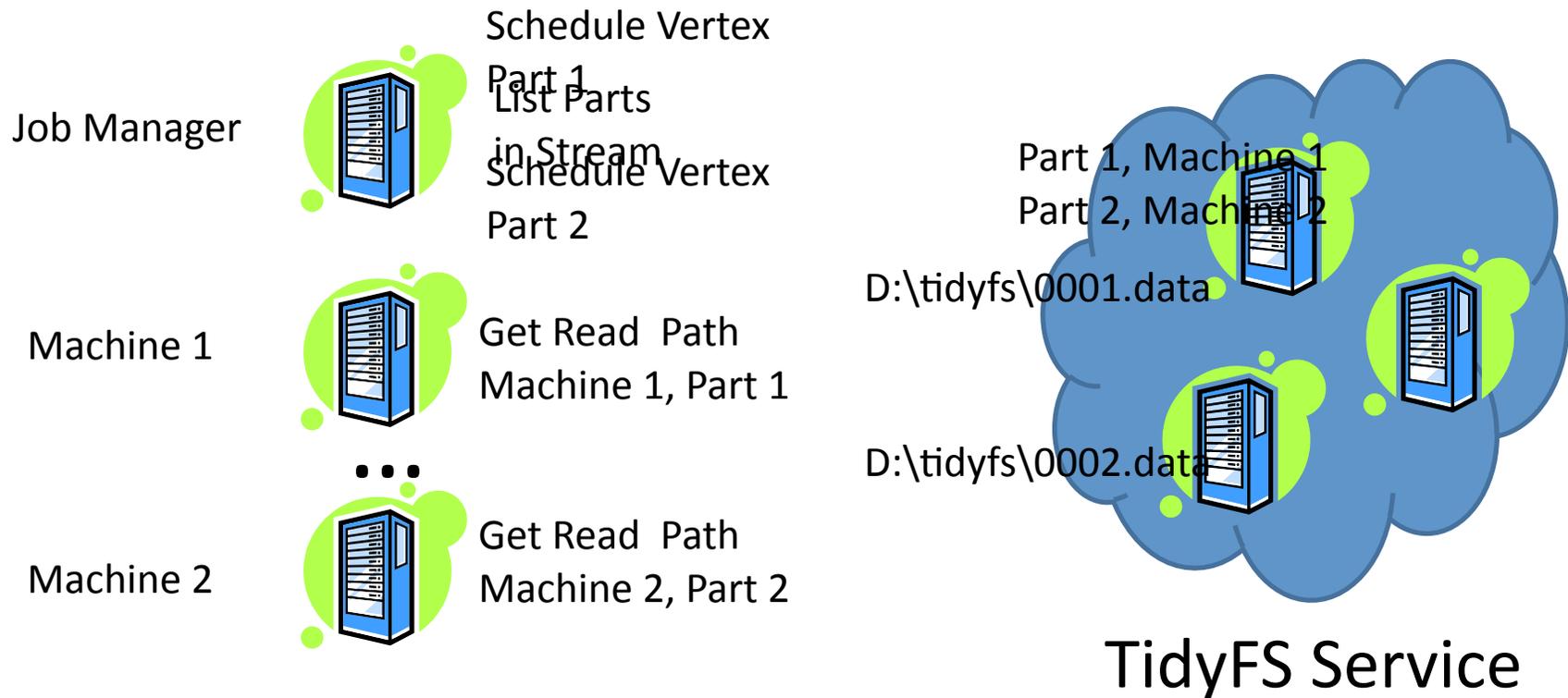
Index	Partition Identifier	Size
<b>1</b>	<b>00000000056BBA4</b>	<b>144.85 MB</b>
2	00000000056BBA5	143.11 MB
3	00000000056BBA6	142.35 MB
4	00000000056BBA7	152.86 MB
5	00000000056BBA8	145.01 MB
6	00000000056BBA9	145.03 MB
7	00000000056BBAA	148.18 MB
8	00000000056BBAB	153.90 MB
9	00000000056BBAC	152.76 MB
10	00000000056BBAD	136.88 MB
11	00000000056BBAE	147.66 MB
12	00000000056BBAF	151.21 MB
13	00000000056BBB0	148.80 MB

Size: 452.39 GB [485,749,068,864 B]  
Replication Factor: 2  
Lease Expiration Time: Infinite  
Creation Time: 9/8/2010 7:14:02 PM  
Last Access Time: 9/8/2010 7:14:02 PM  
Fingerprint: 6D0F41F942C7A069  
3125 Partitions

# Client Read Access Patterns

- To read data in stream, a client will:
  - Obtain sequence of part ids that comprise stream
  - Request path to directly access part data
    - Read only file in local file system
    - CIFS path if remote file
    - Paths to DB and log file for DB part
    - Metadata server uses network topology to return the part replica closest to reader

# How a Dryad job reads from TidyFS



Cluster Machines

# Client Write Access Patterns

- To write data to a stream, a client will:
  - Determine a destination stream
  - Request Metadata server to allocate part ids assigned to destination stream
- To write to a given part id, the
  - Request write path for that part id
  - Write data, using native interface
  - Close file, supply size and fingerprint to server
    - Data becomes visible to readers

# Write Replication

- Default is lazy replication
- Client can request multiple write paths
  - Write data to each path provides fault tolerance
- Client library also provides byte-oriented interface
  - Used for data ingress/egress
  - Will optionally perform eager replication

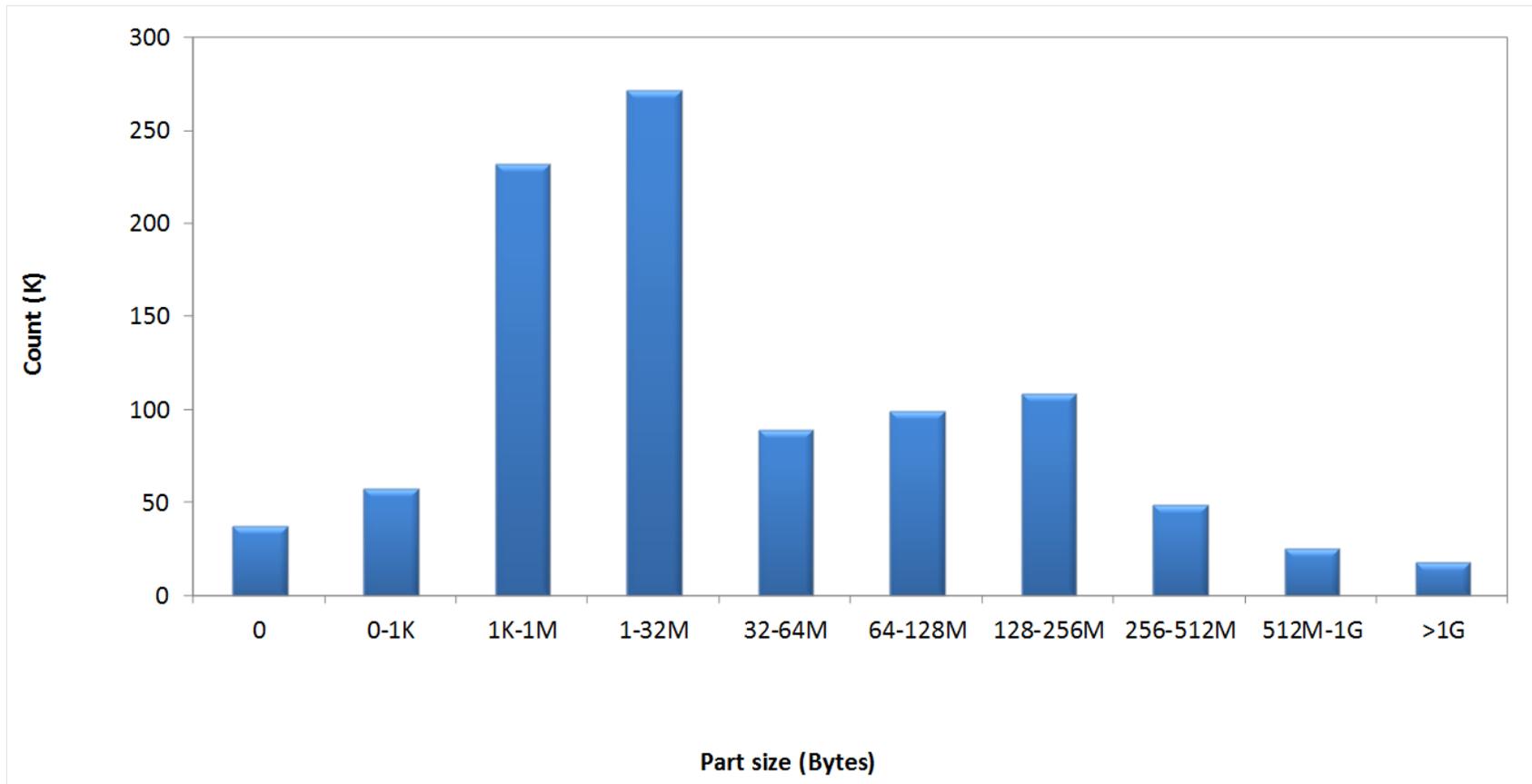
# Design Points – Direct Access to Parts

- Enables application choice of I/O pattern
- Avoids extra layer of indirection
- Simplifies legacy applications
- Enables use of native ACL mechanisms
- Fine grained control over part sizes

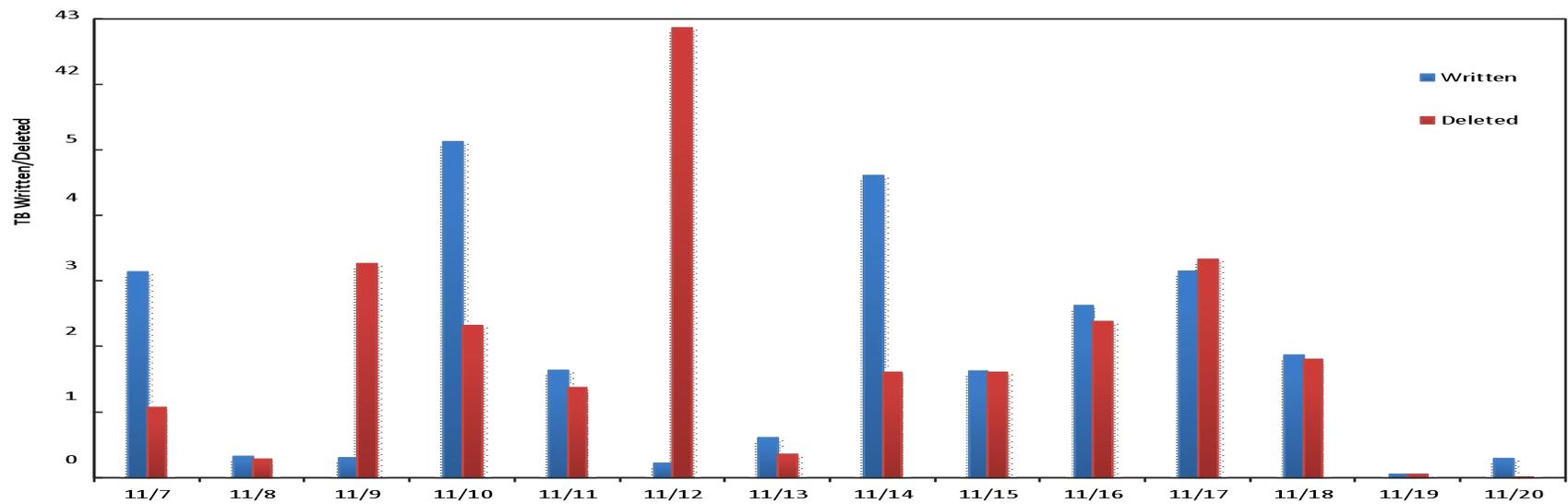
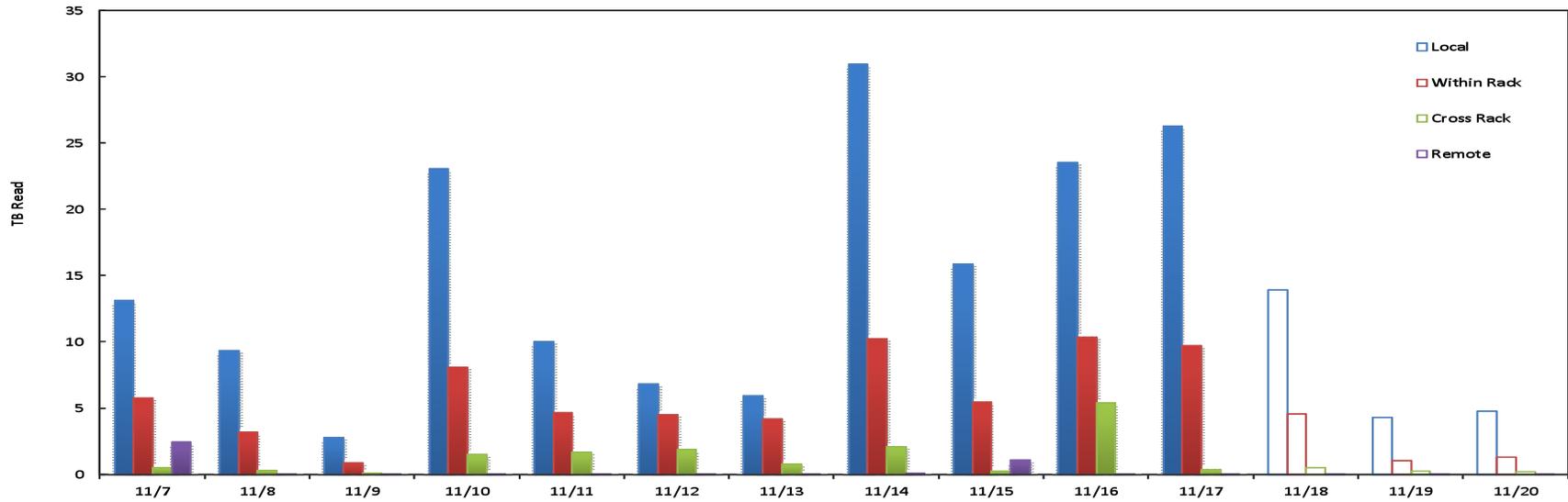
# Operational Experiences

- 18 month deployment and active use
- 256 node research cluster
  - Exclusively for programs run using DryadLINQ
  - DryadLINQ programs are executed by Dryad
  - Dryad is a dataflow execution engine
    - Dryad uses TidyFS for input and output
  - Dryad processes are scheduled by Quincy
    - Attempts to maintain data-locality and fair sharing

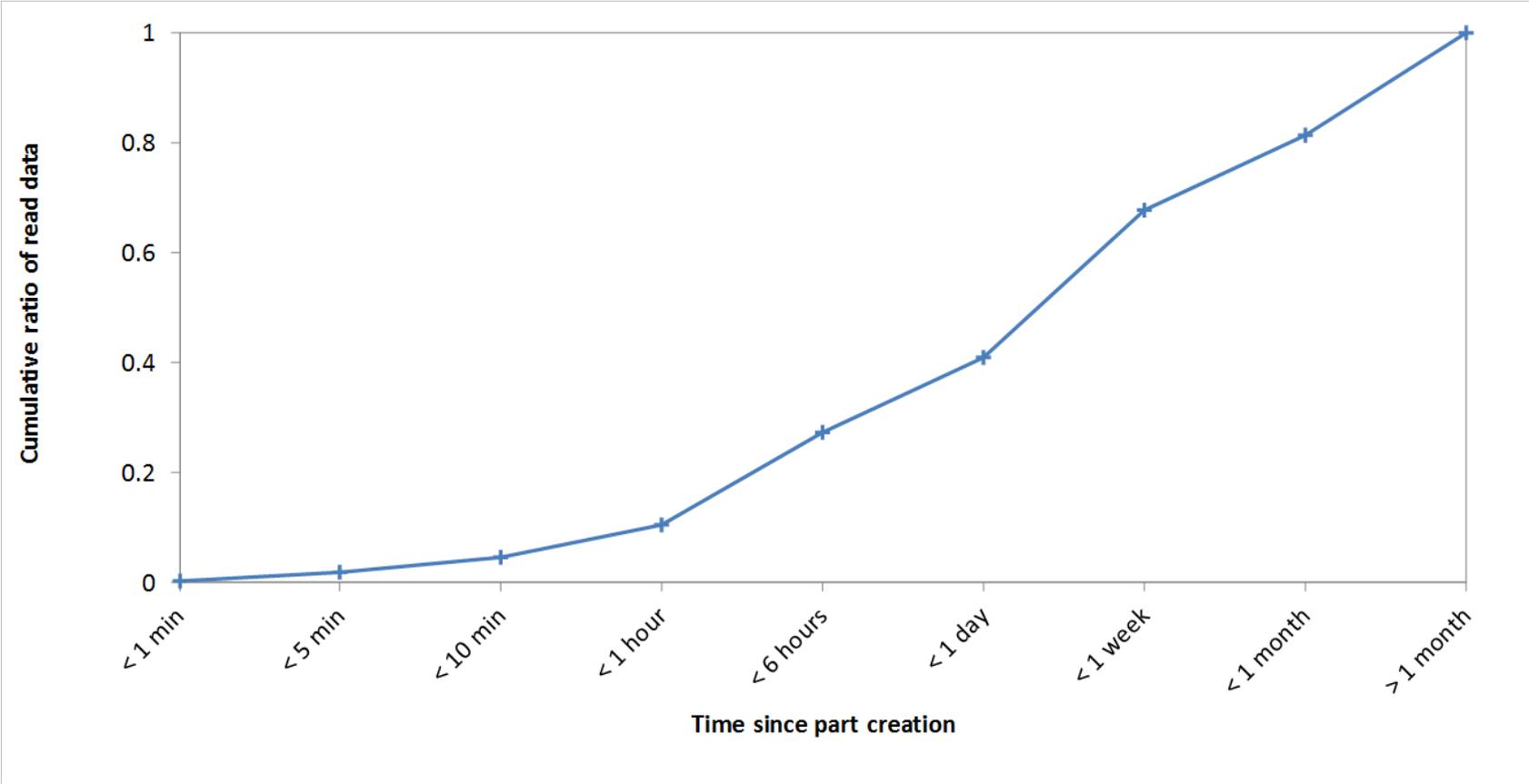
# Distribution of Part Sizes



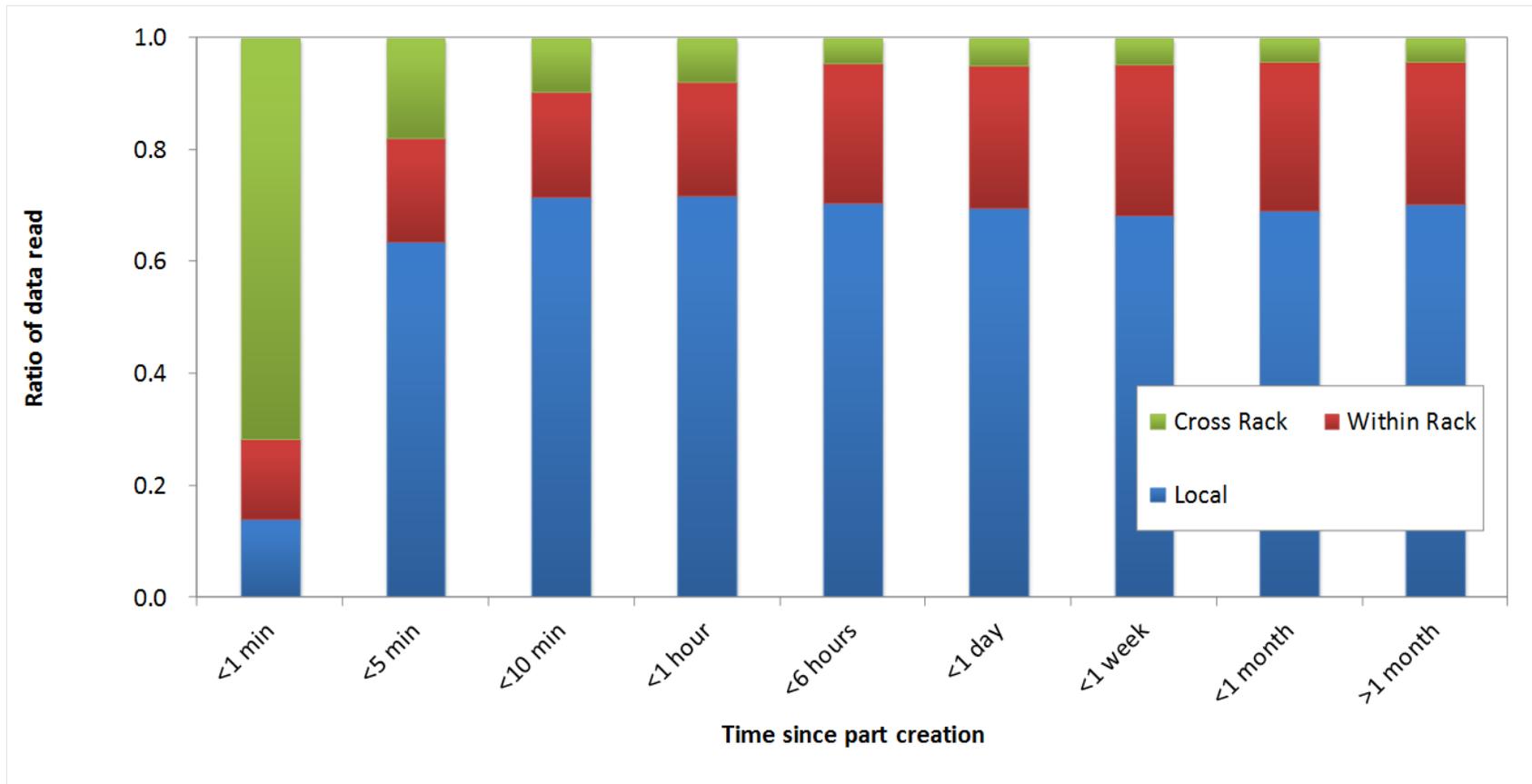
# Data Volume



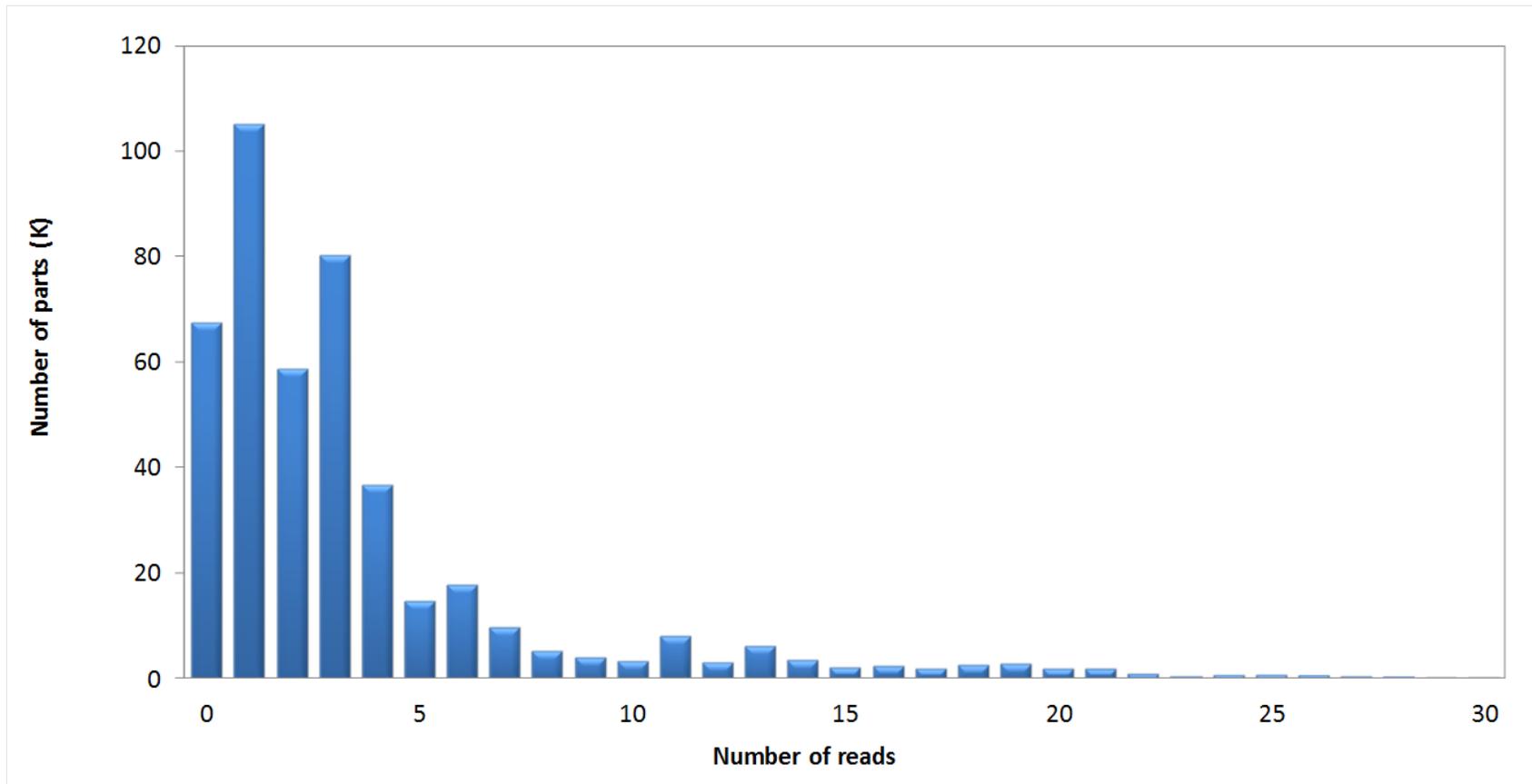
# Access Patterns



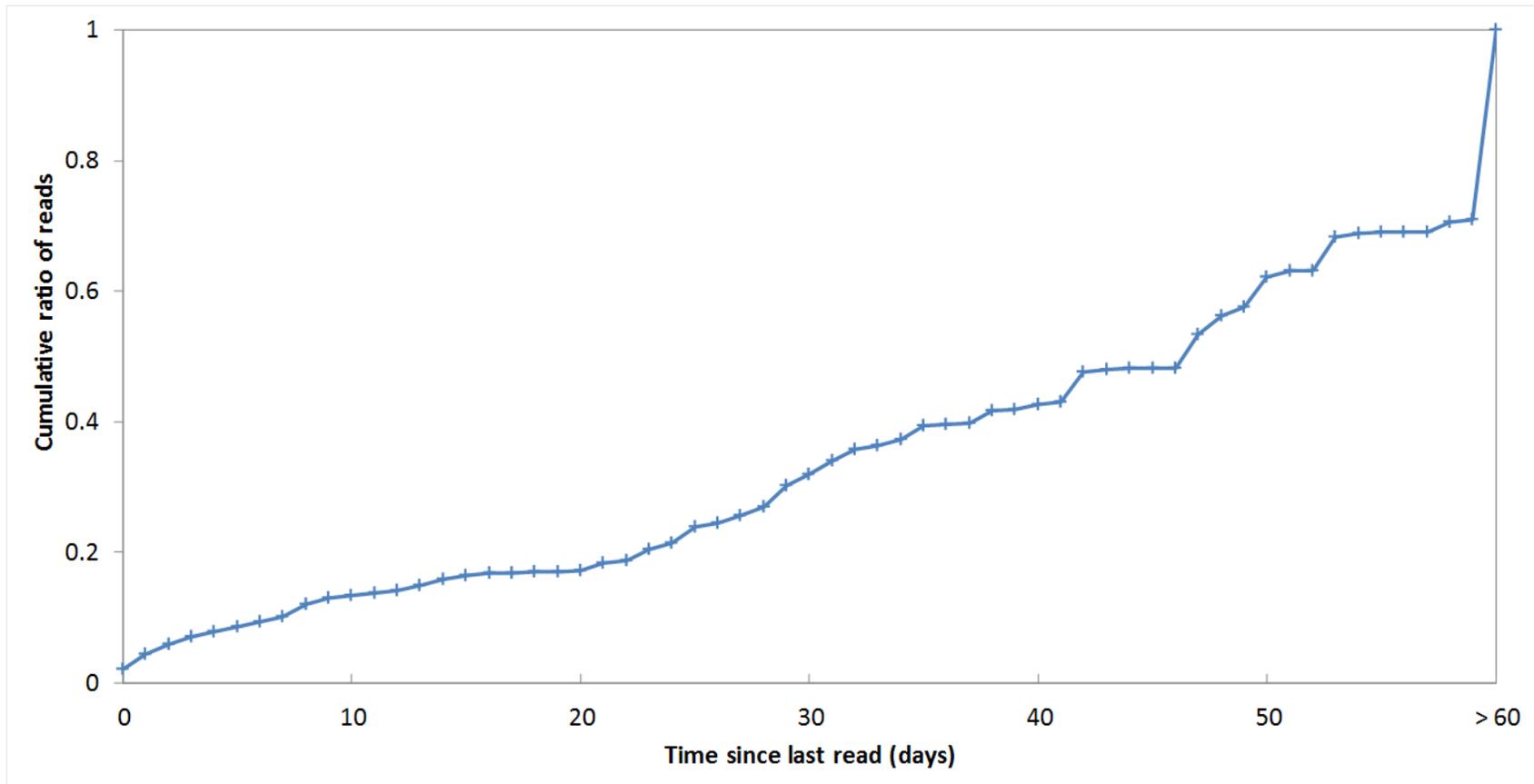
# Data Locality



# Part Read Histogram



# Distribution of Last Read Time



# Evaluation of Lazy replication

- Part replication times over a 3 month window

Mean time to replication (s)	Percent
0 – 30	6.7 %
30 – 60	62.9 %
60 – 120	14.6 %
120 – 300	1.1 %
300 – 600	2.2 %
600 – 1200	4.5 %
1200 – 3600	3.4 %
3600 -	4.5%

# Conclusion

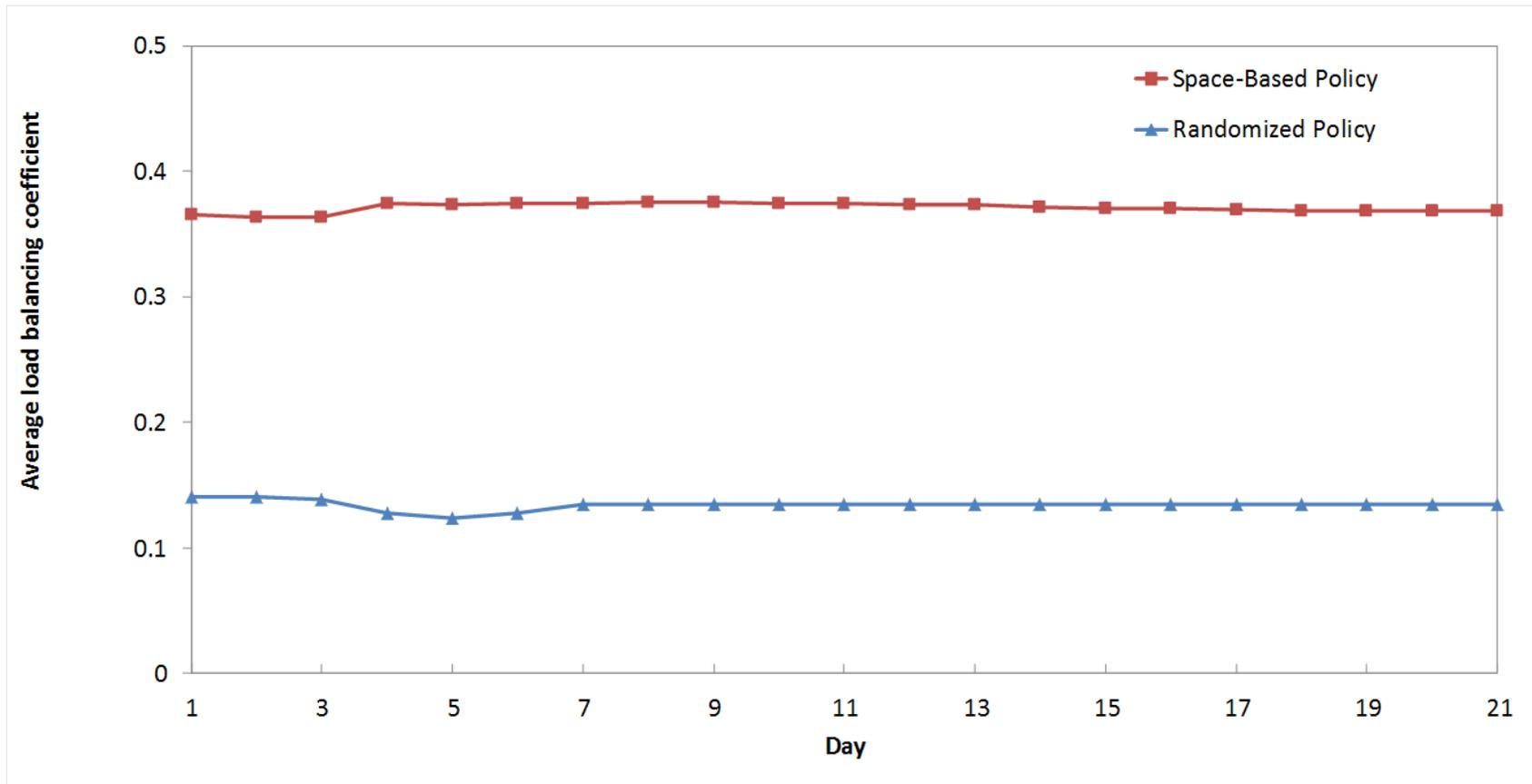
- Design tradeoffs have worked well
  - Pleased with simplicity and performance
- TidyFS gives clients direct access to part data
  - Performance
  - Easy to add support for additional part types such as SQL databases
  - Prevents providing automatic eager replication
  - Lack of control over part sizes
- Considering tighter integration with other cluster services

# Backup Slides

# Replica Placement

- Initially had a space based assignment policy
- Stream balance affects performance
- Moved to best of 3 random choices
- Evaluate balance using  $L^2$  norm

# Replica Placement Evaluation



# Cluster Configuration



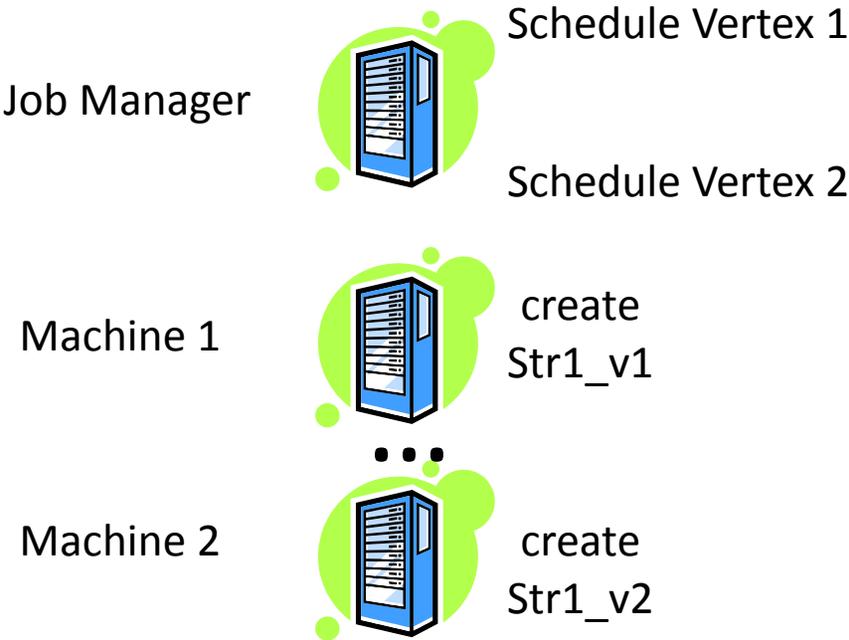
Head Node



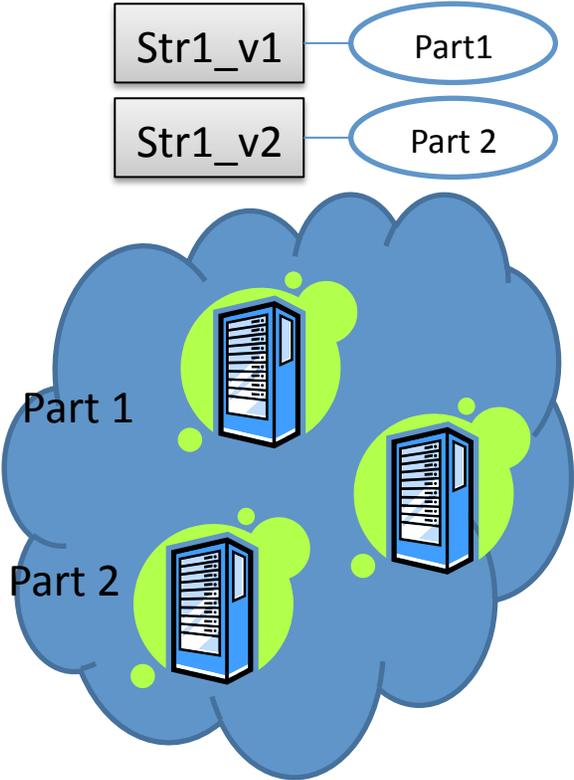
TidyFS Servers

Cluster machines running tasks  
and TidyFS storage service

# How a Dryad job writes to TidyFS



Cluster Machines



TidyFS Service

# How a Dryad job writes to TidyFS

