S3 API Deep Storage Extensions for Hadoop

Stacy Schwarz-Gardner

Spectra Logic

SNIA™

What is Hadoop?

- A framework that allows for the distributed processing of large data sets across clusters of computers
- Hadoop is an open source, Java-based ecosystem of technologies
- **MapReduce** - The framework that understands and assigns work to the nodes in a cluster
- **HDFS** - A file system that spans all the nodes in a Hadoop cluster for data storage. HDFS assumes nodes will fail, so it achieves reliability by replicating data across multiple nodes

Hadoop Storage Challenges

- Storing Multiple Copies
  - The standard setting for Hadoop is to have (3) copies of each block in the cluster.
- Distributed Storage Architecture
  - Compute nodes must scale to meet storage requirements
  - Compute nodes under utilized
  - Clusters grow to meet storage capacity demands
- Centralized Storage Architecture
  - Increased % of high performance storage will be allocated to inactive datasets over time

Introducing

Deep Storage

Deep Storage Paradigm

- Deep storage is **extremely low-cost, power efficient and dense** storage for data that does not need immediate access.
- Deep storage is **accessed over open interfaces** such as REST interfaces and web protocols.
- Data in deep storage is **stored as objects** that are self-describing and written in an open file format.

Deep Storage Enablers

- Restful Interfaces
  - PUT, GET, CREATE, DELETE etc.
- Object Based Storage Architecture
  - Object ID, Metadata, File System Independence
- Organizational Structure
  - “Bucket” Oriented Logical Containers

A REST/S3 interface Enabling Deep Storage

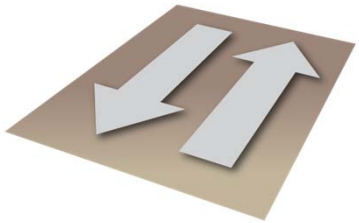
S3 (Simple Storage Service)

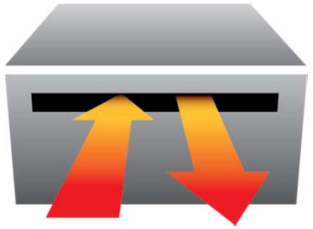
Created by Amazon, S3 simplifies web-scale storage and computing development.

- Defines a cloud storage web service
- Object Based
- Proven technology and wide adoption (2+ trillion objects)
- Many clients and client developers
- **Designed for random disk devices**

What is DS3?
Deep Simple Storage Service

S3 + Sequential Media Support + Removable Media Support = ds3

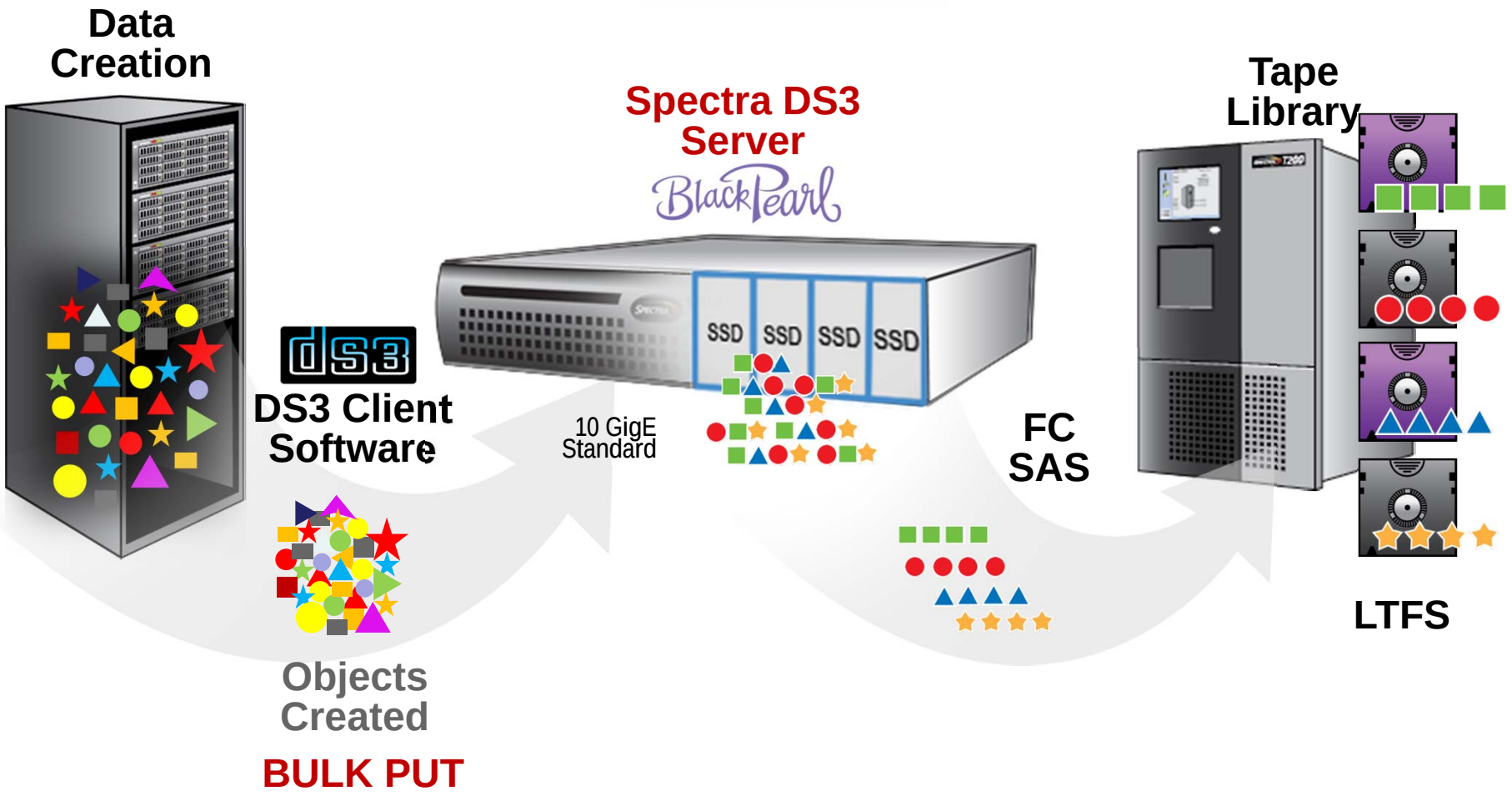




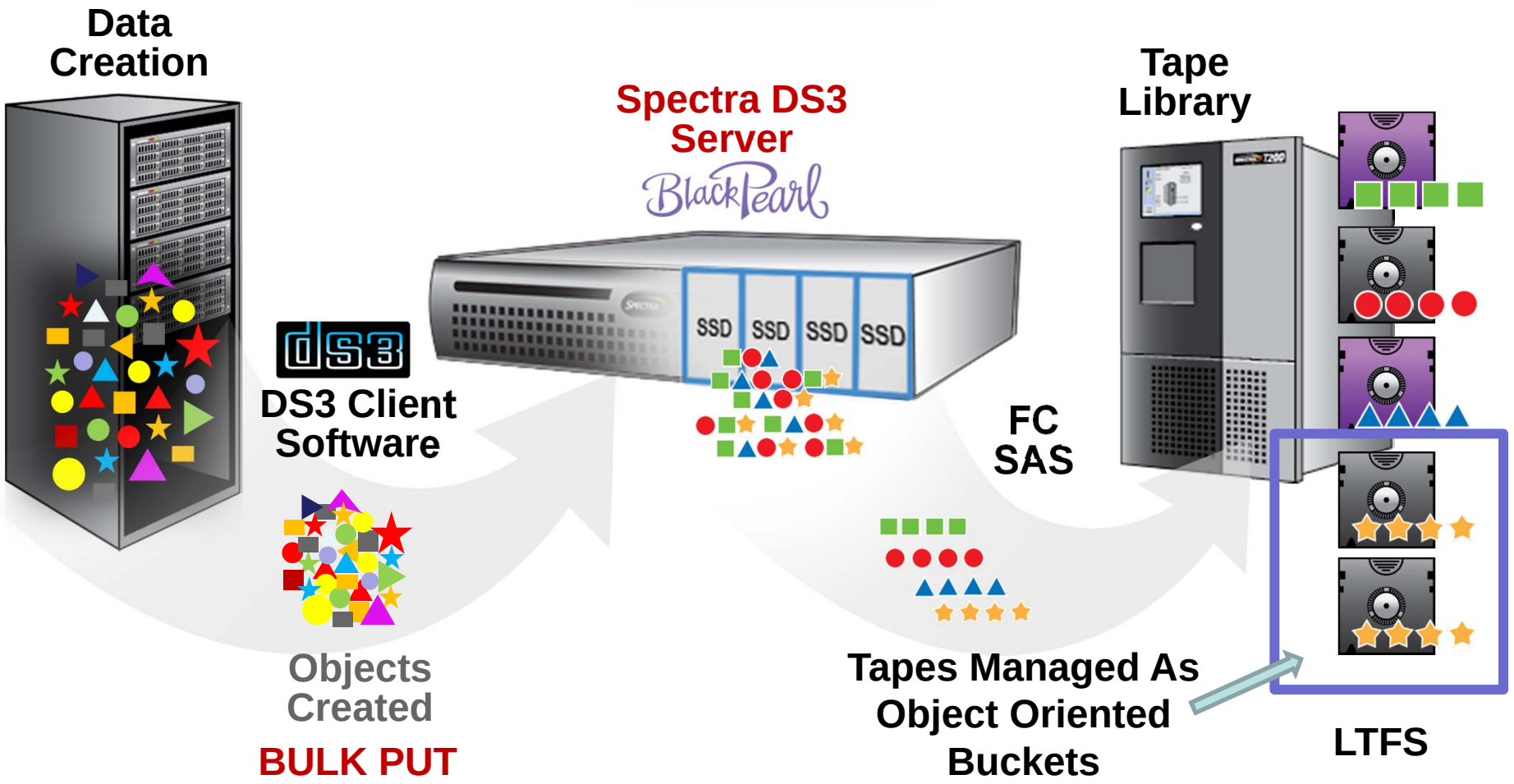
DS3 Objectives

- **Optimized Data Transport**
  - Bulk Data Movement
- **Optimized Data Placement**
  - Bucket Alignment
- **Optimized Data Storage Management**
  - Object Capability
- **Optimized Data Recovery**
  - Bucket, Object, Directory, File
- **Ability to Leverage Sequential Media**
  - Focused on Tape today
  - Other Storage Media in the Future

DS3 Data Flow Example



DS3 Data Flow Example



DS3 Specifics

- **Placement / Optimization policy**:
- There are two ways objects can be specified to be stored on tape.
  - The first is to **optimize tape usage**. This policy is designed around using a fewer number of tapes.
  - The second is to **optimize drive usage and is the default policy**. This policy is aimed at data throughput and will pick an appropriate number of drives to write the data quickly.

DS3 Specifics

- **Bulk commands**:
  - Bulk commands are used to **prime the DS3 server** for both PUT and GET requests to more efficiently service those requests.
  - Each DS3 server has an **ordered list of objects** associated with that specific DS3 server and the placement/optimization policies associated with buckets specified. The object list is the order in which the DS3 client must send the corresponding GET or PUT commands

DS3 Specifics

- **Bucket ejection**:
  - Ejection of buckets is made possible through an extension to the DS3 PUT Bucket operation by specifying the **eject-bucket** query parameter.
  - After ejecting a bucket, it is no longer available by the DS3 interface. Ejecting a bucket is required to physically eject the media out of the tape drive library.

DS3 SDK API Examples

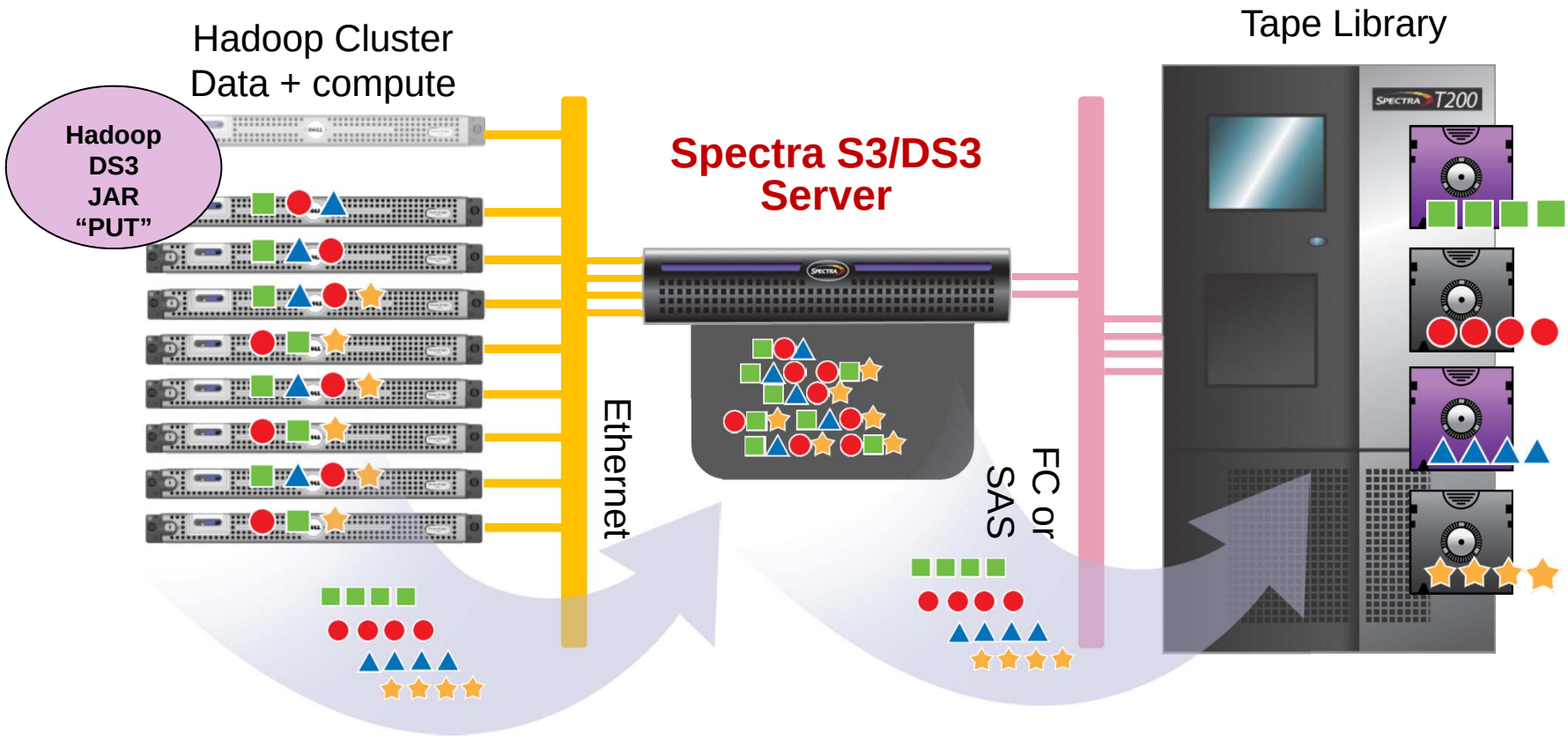
- CreateBucket (Bucket Name)
  - **config bucket create name={text} optimization={PERFORMANCE|MINIMIZE_TAPES} owner={ident} [readonly={Y|N}] [versioning={OFF|ON }]**
- BulkPut (Bucket Name (**rockonbaby**), Directory with List of Files (**TheB52s**))
  - **$ ./bulk -v -l="bulk.log" -authid="dGltcA==" -accesskey="Securetimp" put TheB52s http://vm-bluestormtest1:8080/rockonbaby**
- BulkGet (Bucket Name **(bucket1),** Directory with List of Files **(test1),** Destination **(ds3testfiles))**
  - **$ ./bulk -n -v -l "bulk.log" -authid="MQ==" -accesskey="Secure1" get /ds3testfiles http://vm-bluestormtest1:8080/bucket1/test1**
- ExportBucket (Bucket Name)
  - **config bucket eject {ident} [tapelocation={text}] [trackingid={text}]**

Hadoop Integration

Hadoop Deep Storage

- Follows the standard Hadoop model for running applications inside of Hadoop
- Single JAR (**J**ava**AR**chive) file - run from the command line of the NameNode
- Copy a Directory from Hadoop to DS3
  - ***hadoop jar fileMigrator.jar -c put -b newBooks -i books -o result***
  - The –c specifies the command to be executed – in this case a PUT
  - The –b specifies which deep storage bucket to place the copied files into
  - The -i specifies the directory to copy. -o is used to record any errors that occurred when transferring the files. To verify that the objects have been put successfully list all the objects in the bucket.

Applying DS3 to Hadoop
Copy a Directory

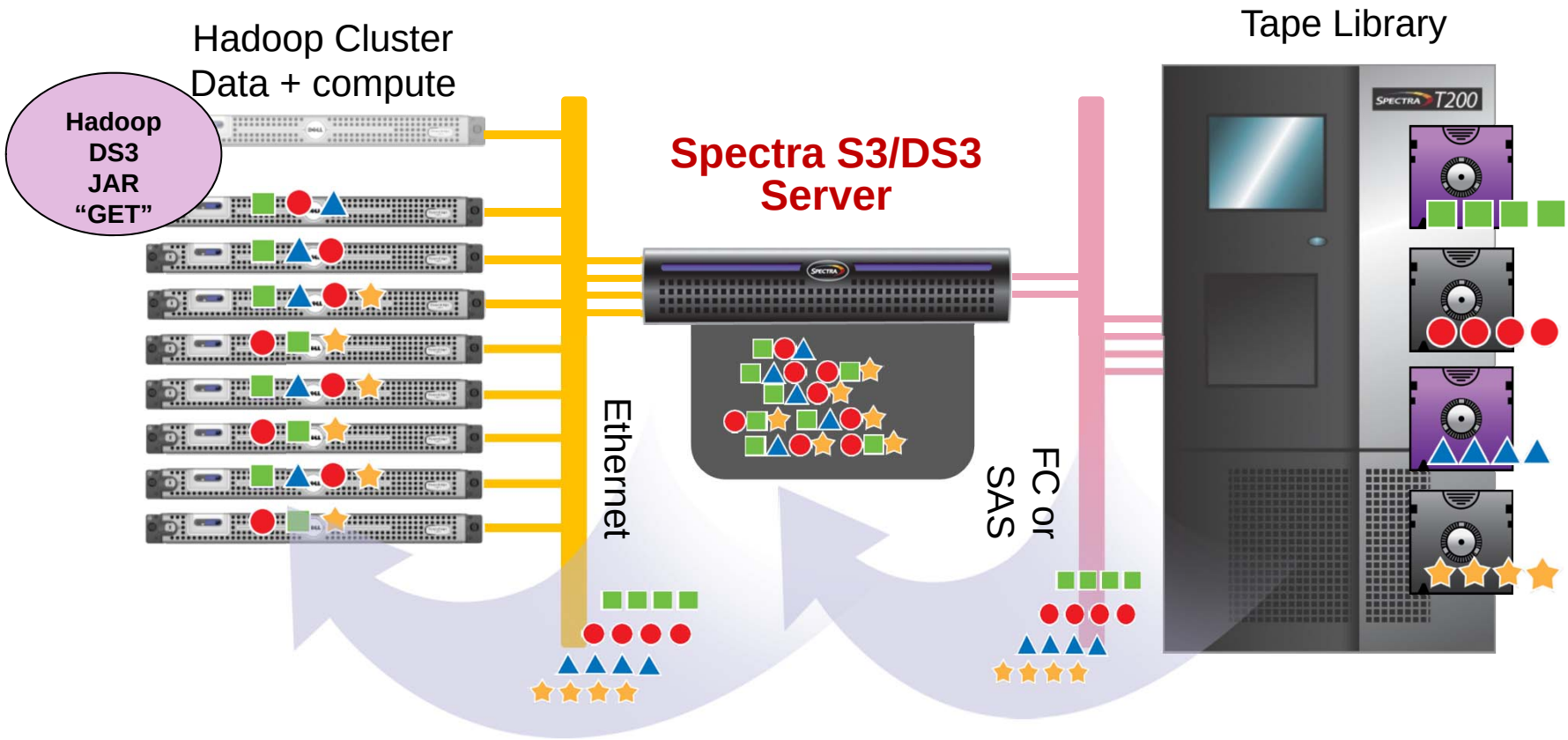


Hadoop Deep Storage

Copy A Directory (PUT)

- Administrators can “stage” inactive datasets in a directory within HDFS
- Run DS3 JAR executable on the NameNode
- List of files to be copied is sent to DS3 Server
- Prime list returned from the DS3 Server is distributed among Mappers in the cluster
- Mappers will then copy the files from HDFS into buckets in Deep Storage environment
- Data can then be deleted from HDFS by Hadoop Administrators

Applying DS3 to Hadoop – Copy Back



Hadoop Deep Storage – Copy Back

- **Copy a Bucket from DS3 and store in Hadoop**
  - ***hadoop jar fileMigrator.jar -c get -b newBooks -o results***
  - The files in the bucket will be restored to the current working directory in hadoop. To store the files in an alternative directory location the -p <prefix> argument can be used. -o is used for reporting errors.
- **GET**
  - Same JAR file format with different command syntax
  - Bring back complete buckets

DS3 Client Development

- Client Development Tools
  - Software Development Kits (SDK)
  - Simulator
  - CLI
  - Several Client Development Projects are Underway – ***Hadoop Client is Available***

In Summary

- Hadoop Clusters continue to grow inefficiently
  - Compute under utilized
  - Storage over utilized
  - Datasets are no longer just transient
- Longer Term Hadoop Storage Options must be considered
- DS3 Enabled Deep Storage provides:
  - Low Cost Sequential Media options for storing less active/inactive data for future analytical purposes
  - While assisting with long term Hadoop Cluster growth management

THANK YOU

Stacy Schwarz-Gardner

stacys@spectralogic.com