

## **SNIA Technical Council: SDS Automation and Orchestration**

## Mark Carlson, Toshiba Leah Schoeb, Evaluator Group

## This presentation is the work of the SNIA Technical Council

The SNIA Technical Council is a group of industry experts elected by SNIA members to oversee the technical work of the SNIA. We also sponsor the annual Storage Developers Conference and occasionally produce position papers on current storage architecture topics such as SDS.

The current TC members are Don Deel (chair), Bill Martin (vice chair), Craig Carlson, Mark Carlson, Bruno Guiet, Chin-Fah Heoh, Arnold Jones, Fred Knight, Carlos Pratt, Yukinori Sakashita, Leah Schoeb, Udayan Singh, Dave Thiel, Doug Voigt, Steve Wilson and Alan Yoder



## Defining Software Defined Storage (SDS)

SDS is often defined by describing a set of attributes

**Dis-aggregated** Automated **Policy Based Incremental** Commodity Pooled Self-service Service Levels **Build It Yourself** What is the underlying value of SDS? How can SDS be structured as an ecosystem

to deliver this value?





#### Flexible construction of services

#### Separation of control and data planes

### Deployment Simplicity



## **SDS Value**

#### Flexible construction of services

#### □ SDS spans the boundaries between servers and storage

- Data services can be executed in servers or storage
- This has potential impacts on security and reliability



## **SDS Value**

#### Flexible construction of services

#### SDS spans the boundaries between servers and storage

- Data services can be executed in servers or storage
- This has potential impacts on security and reliability

#### Separation of control and data planes

#### SDS builds on Virtualization of the Data Path

Data Path Virtualization alone is not SDS



## **SDS Value**

#### Flexible construction of services

#### SDS spans the boundaries between servers and storage

- Data services can be executed in servers or storage
- This has potential impacts on security and reliability

#### Separation of control and data planes

SDS builds on Virtualization of the Data Path

Data Path Virtualization alone is not SDS

## Deployment Simplicity

- Storage service interface
  - Expresses Requirements for the Cloud/DC/Storage/Data Administrator
  - Receives Service Levels from the Cloud/DC/Storage/Data Administrator
  - Automates the matching of services to requirements



## **SDS Value Map**

- Flexible construction of services
  - Works with standard hardware as well as specialized hardware
  - Works with scale out or scale up architectures
  - Enables incremental building of storage/data services solutions
- Separation of control and data planes
  - Includes pooling of resources
  - Includes Service Level Management (metadata tagging)
    - □ Large grain "Labels" for your storage/data containers
    - □ Fine Grain "Knobs" on individual data objects
  - Enables dis-aggregation of storage and data services
- Deployment Simplicity
  - Includes self-service interface that supports provisioning
  - May include policy based management automation
  - Simplifies management of scale

## What is needed: Virtualized Data Path

- File, Block, Object
- Without a virtualized data path the "hardware" is defining the storage





## What is needed: Management APIs

- Preferably Standardized
- Must be programmatic
- Storage services include capacity, performance, availability, security, data paths





## What is needed: Data Services

Data services provide containers for files, objects, blocks

Data services are deployed dynamically





## What is needed: Policy Driven Service Levels

- SDS API is used to define service levels
- Metadata is used to match requirements with capabilities



SD @

## **Storage/Services Pool**

- A storage services pool specifies storage that may be used and data services that are applied to meet certain ranges of requirements
- Requirement granularity depends on implementation: Volume, File, Object, Container
- Resources are aggregated into pools
- Data services are added to meet service level requirements
- New resources are added to pools that need them
- Failed resources are removed from pools until repaired



## What is needed: Data management API

Gives SW developers easy access to data services

Metadata controlled service selection



## **Big Picture**



## How are requirements conveyed traditionally?





# How are requirements conveyed with SDS?

Requirements flow through the data storage interface





## **Automation and Orchestration**

- Orchestration is already happening due to Cloud implementations
  - Largely single-vendor implementations
  - Pooling is a foundational requirement
- Data Services are still applied at large granularity (whole device, filesystem, etc.)
  - Micro-service based data services emerging
- Automation needs a standard API to storage services (not just for interoperability)



## **Developing for SDS**

- Best practice is to use standard WBEM protocols for managing the storage from automation software (future proof your development)
- Client libraries available in multiple languages from open source projects
- Build up functionality, exposing Policy decisions to the administrator
- Tie into Data System Metadata from Self Service I/F (i.e. CDMI)





- OpenPegasus and OpenPegasus Wiki
- □ <u>SBLIM</u>
- □ <u>Purgos</u>
- □ <u>PyWBEM</u>
- J WBEM Server
- Small Footprint CIM Broker
- CDMI Reference Implementation



## **Develop for SDS**





## Thank You

