The Premier Advanced Recording Technology Forum

THIC Inc.

OpenSMS

Stephen Cranage
StorageTek
15050 La Jolla Place, Colorado Springs, CO 80921-2564
Phone:+1-719-484-0584
E-mail:Steve_Cranage@stortek.com

Presented at the THIC Meeting at the National Center for Atmospheric Research, 1850 Table Mesa Drive, Boulder CO 80305-5602

July 19-20, 2005

OpenSMS Defined

- Open Source Systems Managed Storage Implementation
- HSM - Directed toward Solving Problems of Scalability, Data Protection, and Archiving
- Linux/DMAPI Based
- TMS Namespace vs. Filesystem Centric

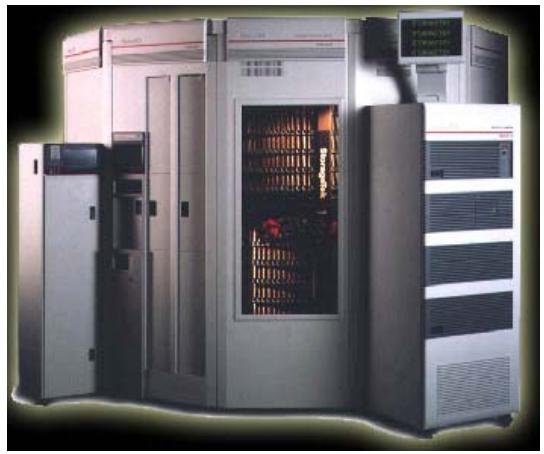
Theme: "Growth and Evolution of Data Protection and Archiving Technologies"

Scalable Archives Require Removable Media

- Technology Longevity – Accommodating Increases in Arial Density
- Complexity of Retirement of Random vs Sequential Access Storage
- Environmental Costs
- Maintenance Costs/Availability
- Transportability

19 Year Life of StorageTek's "Silo"

Same ACS
Storage Capacity - 1000 X
Robot Speed - 5 X
Bandwidth - 200 X
2 Generations of Media
3 Generations of Transports



1987 6000 Slots @ 200 MB
Max 16 Transports at 3MB/Sec
Shared across 4
90 Exchanges/Hour
Total Capacity – 1.2 TB
Uncompressed

2005 6000 Slots @ 200 GB
Max 80 Transports at 30MB/Sec Per Transport
450 Exchanges/Hour
Total Capacity – 1.2 PB
Uncompressed

4400/9310 Silo Longevity Study

Year	Item	List	Maint	P & C	Cap	Transfer	EPH
1988	**Original ACS Purchase**	**$984,000**	**$99,600**	**$29,800**	**1.2TB**	**12MB/Sec**	**90**
1989			**$99,600**	**$29,800**			
1990			**$99,600**	**$29,800**			
1991			**$99,600**	**$29,800**			**110**
1992	**ICRC**	**$70,000**	**$99,600**	**$29,800**	**2.4TB**	**20MB/sec**	
1993	**36TR ->9310 Robot**	**$338,000**	**$94,760**	**$29,800**	**4.8TB**		**190**
1994			**$94,760**	**$29,800**			**350**
1995	**9490/EE Media**	**$897,000**	**$91,180**	**$12,576**	**9.6TB**	**192MB/sec**	**450**
1996			**$91,180**	**$12,576**			
1997			**$91,180**	**$12,576**			
1998	**9840A/9330**	**$322,000**	**$66,528**	**$12,576**	**64.8TB**	**256MB/sec**	
1999			**$66,528**	**$12,576**			
2000	**9940A**	**$268,000**	**$29,244**	**$8,322**	**240TB**	**320MB/sec**	
2001	**9840B**		**$29,244**	**$8,322**		**464MB/sec**	
2002	**9940B**	**$316,000**	**$29,244**	**$8,322**	**1.32PB**	**800MB/sec**	
2003	**9840C**	**$304,000**	**$29,244**	**$8,322**	**1.44PB**	**960MB/sec**	

Disk Lifespan

- 3-4 Years, Driven by Environmental & Maintenance Costs
- Retirement is Labor Intensive, Users Are Denied Access to Data
  - Disk is used as DB or Filesystem Storage
  - In either event, record level updates have to be stopped to move data
  - 100TB @ 100MB/Sec ~ 2 weeks to move….

Barriers to Effective Use of Archival Media

- Mainframe and Monolithic HPC OS's had Integrated TMS and HSM services
- Migration > Low Cost Hardware > Low Intelligence OS Services for Storage
- Current OS Environments see Tape as Backup Target, Not for Direct Access

OpenSMS Principal Attributes

- Distributed TMS Namespace

(Complete Tape I/O subsystem for UNIX!)

- HSM Services Through DMAPI
- Rich Data Classification
- Policy Based Management of File Objects

Distributed TMS Namespace

- Hardware Independent
- Complete Abstraction for Removable Media, Including Device Allocation, Mount Request System, Low Level Device Control, Media Management and File Cataloging
- Objects are Stored in the TMS Namespace as ANSI Standard HDR4 Label Tape Datasets, Metadata Model is Based on Available HDR4 Label Fields
- File Objects in TMS have Enterprise Wide Scope
- TMS Clients are All User Level Code – Widely Ported Historically

Distributed TMS Namespace

- TMS Data Movers are FC and IP enabled
- Transports are any-to-any Shared SAN Resources
- Completely Platform Independent, Application Neutral Data Representation in the TMS Namespace
- Provides a Namespace that is Flat, and Logically Divided into Volumesets (volumeset ~ directory)
  - Policy Attributes are passed to the TMS when Volumeset is created

Distributed TMS Namespace

- Data Sharing over the SAN Facilitated by Simplicity of Metadata Model
- File Updates, no Block Updates – Gno++
- Single User Access at Any Time
- Hide Access Limitations with Private Filesystems using Archive Policies and Handlers that Service Reads on Block Released Files (Data Faults)

DMAPI hsmd

- Based on XFS DMAPI
- Performance Neutral, Managed Regions Turned off on First Write I/O
- Create/Modify Events are Synthesized Seconds after the Event
- Result is a File Copy (Copy Policy) and/or SQL insert into RDBMS Server
- RDBMS work_q Drives Surrogate Policy Engines (Archive, Block Release, etc)

Policy Engines

- Copy Policy
  - Async File Replication to Federated Filesystem
  - File Rather Than Block Level
  - Enabler for Muli-tier Filesystems i.e., SSD Front Ending ATA & FC
  - Block Release with Data Fault Servicing from Peer Filesystem
  - Source Filesystem Requires DMAPI, Target Does Not

Policy Engines

- Archive Policy
  - Near Real Time Duplication of Files into TMS Namespace
  - Supports Block Release with TMS servicing Data Faults
  - TMS Containers Created with Policy Attributes
  - Archive Policy Directs Files into Appropriate Volumesets

Archive Policy Data Classification

- Select File Objects from work_q
- Perform REGEX on name/attributes
- Disaggregate File Objects Into vshandler Queues for Archiving into the Appropriate TMS Volumeset
- Model for Other Policy Engines That Can Act Directly on the File Objects to Set User Attributes, or Block Release the File, or Execute Some Other User Process

Meta Data RDBMS Integration

- File Create/Modify and Data Fault Events all cause a filehandle to be Inserted in a work_q
- On Create/Modify, We Also Insert the Metadata into a Metadata Table
- Unused Now, but Plan on Some Block-Release Candidate Selection and SRM Functionality Later On.

TMS Namespace Utilization

- Use Private Filesystems as a Means to Convey File Objects into a Global TMS Namespace
  - Distribute Metadata (inodes from dump/restore) to Create Empty File Objects in other Private, Local Filesystems
  - Data Fault Will Return the Object From TMS
  - Archive Policy will Update the TMS Object as New Generation on Local Modify
  - Block Release the Local File to Have a Subsequent Read Data Fault Return the Most Recent Version

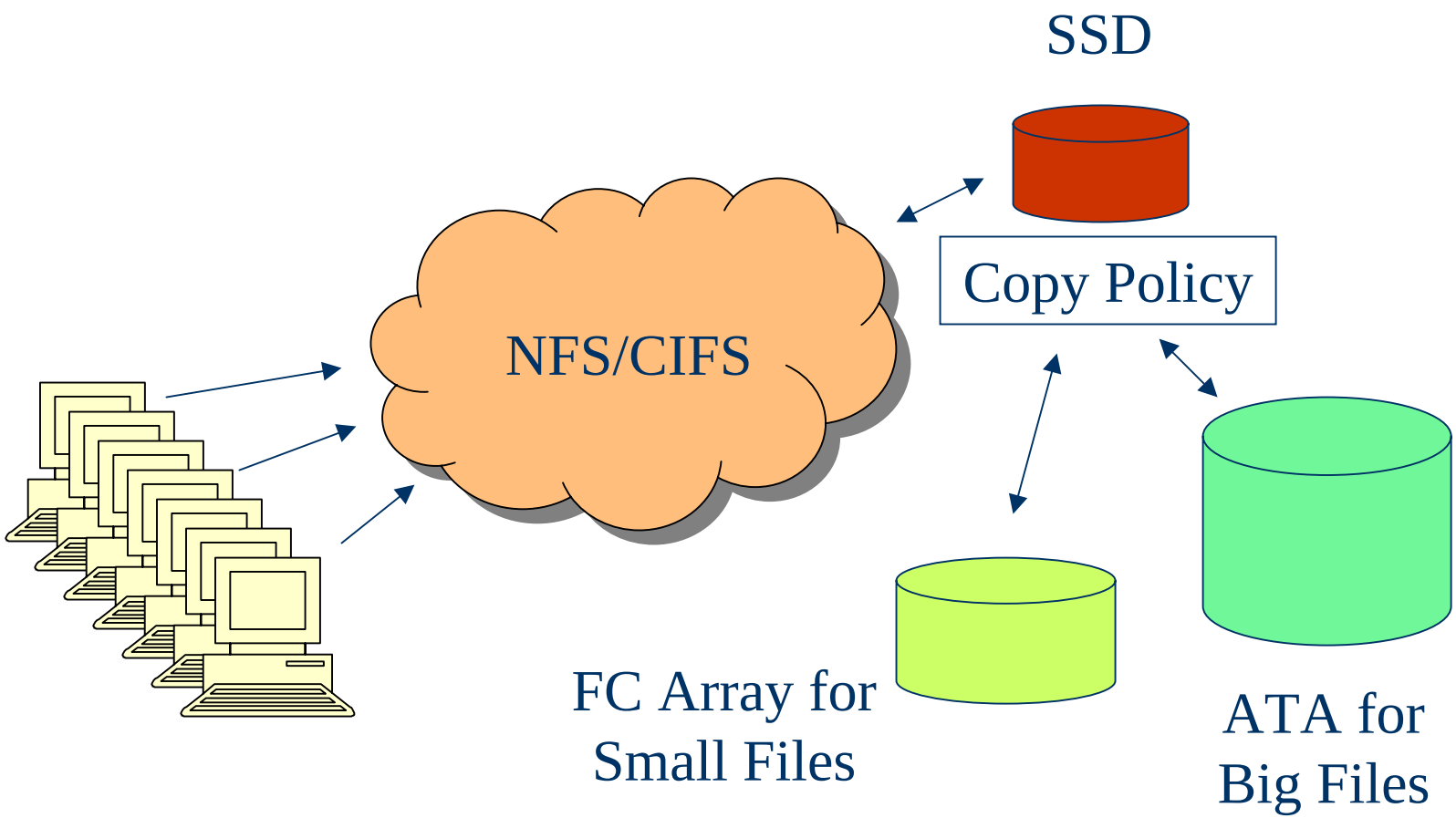
TMS Namespace Utilization

- Access the Volumeset from Command Line or API
- Perform Stream I/O from the Command Line or API

OR

- Use GUI to Browse a Volumeset's Files
- Execute a Stored Procedure Against the File Object

Example Topologies

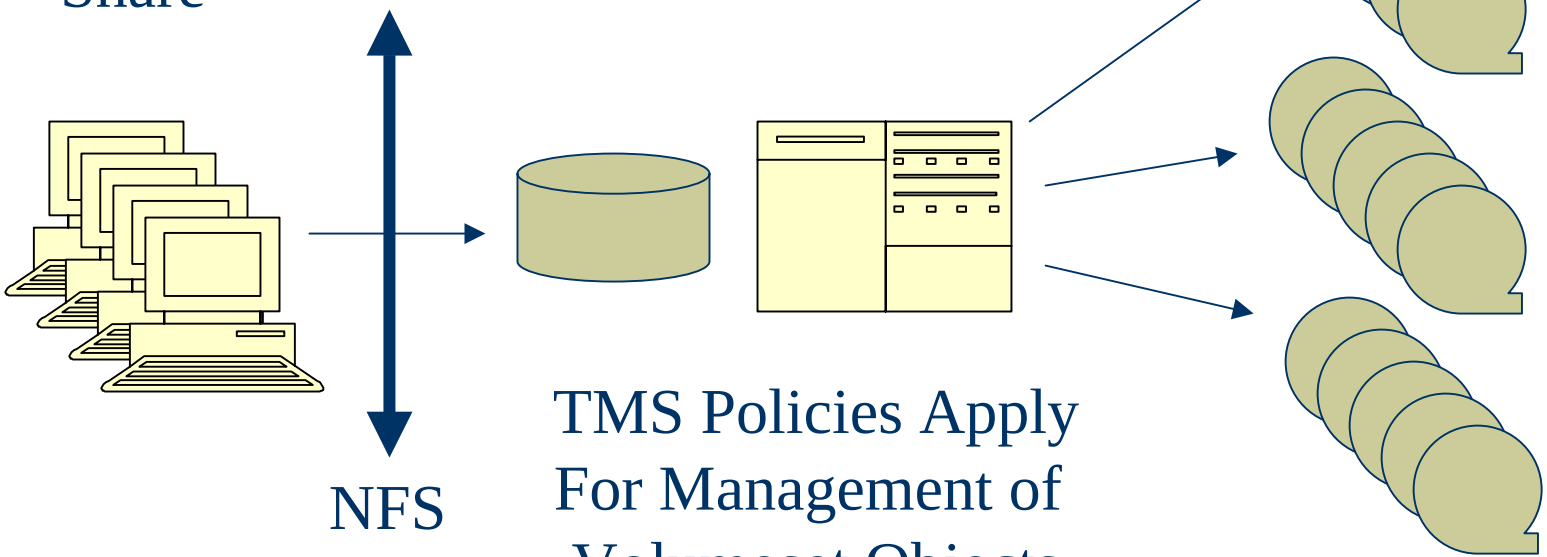


Example Topologies

Workstations
Run Copy Policy
To Network
Share

Archive Policy
Archives to Content
Specific Volumesets

AVI Files



NFS

TMS Policies Apply
For Management of
Volumeset Objects

User Files

Distributed Configuration Management

- OpenSMS is a Distributed Storage Management Toolkit
- Server Based Conf Files Added Complexity, Counter to Our Goals
- Needed and XMLish Way to Define and Distribute Rules and Policies

Distributed Configuration Management

- A GUI Builds a Set Of Nested name/value pairs to define policies, Volumeset Attributes, and all Other Configuration Variables
- SMS daemons Request Their Configuration Variables At Startup from RDBMS Server

Odds and Ends

- “Dataless” Dump Files done, Need to create an Integrated Data Protection Environment + Meta Data Distribution
- Other Policy Engines (ex. wget/wput)
- User Defined Policy Engines
- File “rollback” Capability to be Added

URLs

http://opentms.sourceforge.net

http://openhsm.sourceforge.net