
# Storage in combined service/product data infrastructures

Craig Dunwoody
CTO, GraphStream Incorporated

# SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.
  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

# Abstract

- Storage in combined service/product data infrastructures
  - It is increasingly common to combine as-a-service and as-a-product consumption models for elements of an organization's data infrastructure, including applications; development platforms; databases; and networking, processing, and storage resources. Some refer to this as "hybrid" architecture.
  - Using technical (not marketing) language, and without naming specific vendors or products, this presentation covers some improved storage capabilities becoming available in service and product offerings, and some scenarios for integrating these kinds of offerings with other data infrastructure services and products.

# This presentation

- These slides will be available via Web
  - www.snia.org.education/tutorials
  - Slide sharing site; use your favorite search engine
- Please feel free to ask questions

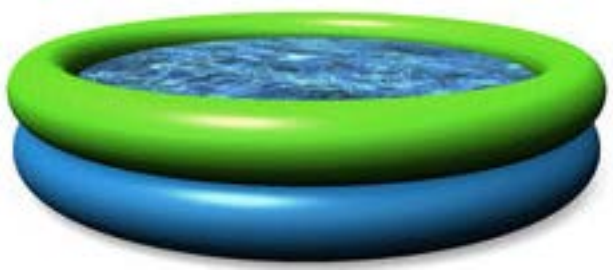
# Context

- IT
  - Support organization's missions by applying available info technologies effectively & efficiently
- Data management
  - Key component of IT
- Storage
  - Key component of Data management

# Data management

# "What": datapools

- Org might have from one to thousands++ of datapools
  - Each has unique set of attributes, some time-varying
- Small sampling of attributes:
  - Content characteristics
    - Structured/unstructured
    - Specific data types
    - Sensitivity of data
    - Value of data to org
    - Consequences of data loss
    - Possibility to re-create data
  - Analytics
    - Data volume & growth patterns
    - Transaction statistics
    - Data-efficiency performance
      - Dedupe
      - Compression
  - Policies/requirements/governance
    - SLAs
    - Access/security
    - Data sovereignty
    - Transaction performance requirements
    - Recovery targets: RPO, RTO
    - Snapshot or Continuous Data Protection parameters
    - Transaction logging
    - Backup
    - Archiving
    - Retention
    - Audit
    - E-discovery



# “Where”: endpoints & storage resources

- Endpoints
  - Machines that initiate transactions against datapools
  - Datapools might need to support from one to billions++ endpoints
  - Location
    - May be anywhere on Earth
    - May be time-varying, e.g. mobile devices
    - May be fixed, e.g. office/plant site
    - IT may have choice, e.g. data services that could run anywhere

- Storage resources
  - Building blocks of storage capability
  - Location
    - Constrained by locations of datacenters,net connectivity
    - Physical & network proximity to endpoints may be beneficial
- May be possible ot move some endpoints & storage resources closer to each other

# Storage infrastructure design

- Challenging optimization problem
- Inputs
  - Set of datapools
  - Set of endpoints
  - Set of available storage building blocks
  - Numerous other factors
- Output
  - Set of specific storage resources deployed in specific locations
- Very little software tooling/automation currently available to assist

- Future infrastructure-design software frameworks could enable better results
  - Intelligent infrastructure-design assistants
  - Better analytics
  - Possibility to apply machine learning & related techniques

# “How”: storage building blocks

- Each building block provides external data interfaces, typically one or more of:
  - APIs / wire-protocols, typically one or more of:
    - Block, e.g. iSCSI
    - File, e.g. NFS, SMB
    - Object, e.g. Swift, S3
    - Distributed-File, e.g. HDFS
    - Platform management, e.g. Redfish, IPMI
  - CLIs
  - HTML GUIs
- Each building block supports capacity pools, typically one or more of:
  - Other storage building blocks
  - Linear magnetic tape
  - Optical disk
  - Magnetic disk
  - NAND flash
  - Storage class memory, e.g. PCM
  - Powerloss-protected DRAM, e.g. NVDIMM-N

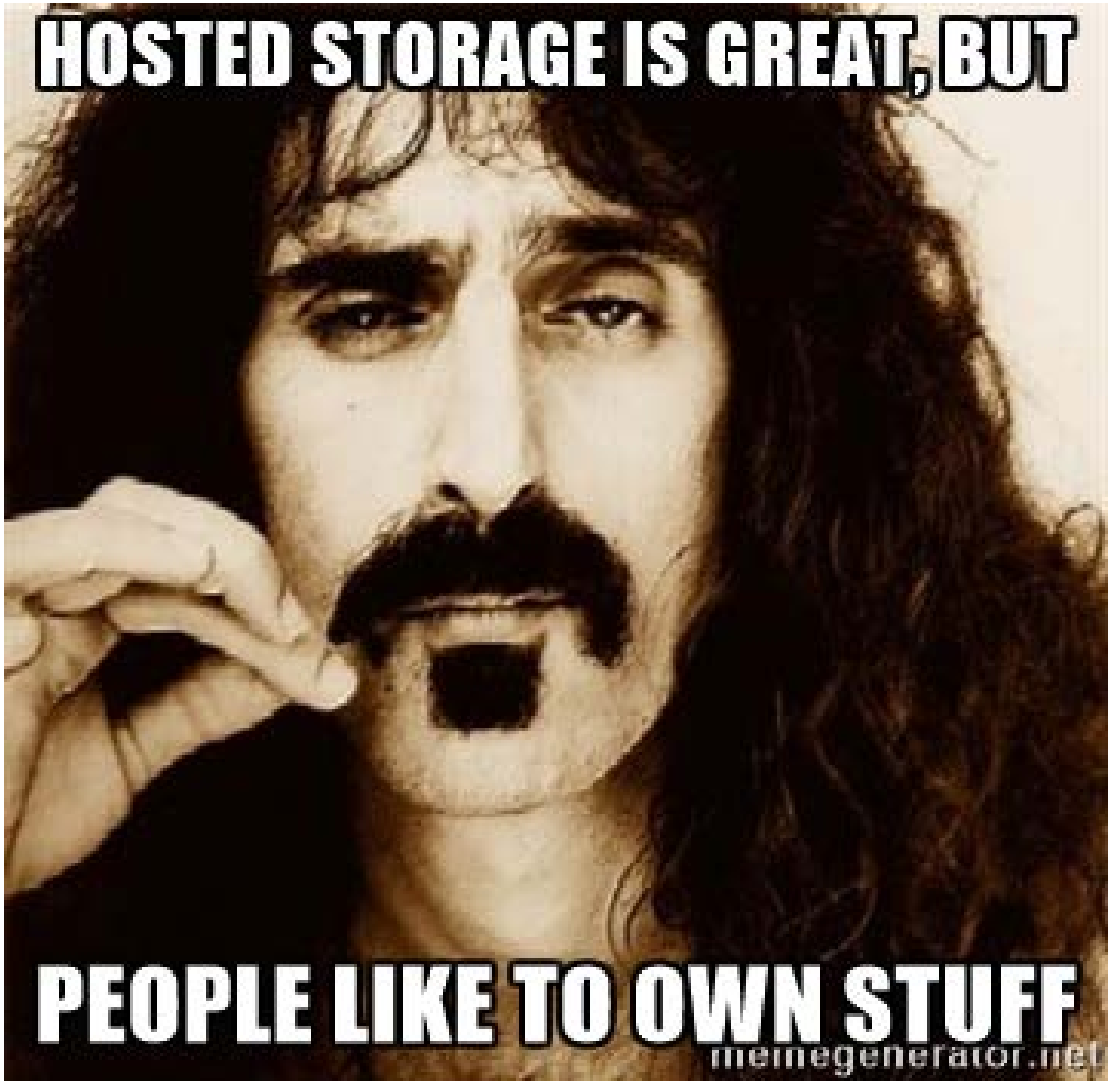
# Example storage building block types

- Many variations on the following themes:
  - Unbundled
    - Self-op and/or hosted at building block level
    - Packaged as software to run on servers
    - Choose any of multiple supported server platforms
      - Specific physical & virtual makes/models
      - Self-op or hosted at server level
    - Typical scaling architecture: one or both of
      - Scale-up: single node; multiple controllers share capacity pool
      - Scale-out: multiple nodes, each with controller(s) + capacity
    - May include “hyper-converged” ability to run encapsulated app workloads, e.g. containers or virtual machines
  - Physical Appliance
    - Similar to Unbundled but packaged as integrated unit, physical-server + software
    - Software may also be available as Unbundled

# Example storage building block types, cont.

- Service: storage-focused
  - Hosted only
  - Hardware+software implementation details invisible behind external interfaces
  - May support low-overhead on-demand provisioning of resources via data interfaces, e.g. APIs, CLIs, GUIs
  - Implementations may serve large numbers of clients and maintain resource pools with significant headroom, thereby enabling extensive elasticity of resource provisioning for individual clients
  - Data may be replicated automatically across multiple physical datacenters for durability & availability
- Service: storage-plus
  - Similar to storage-focused service, except:
    - Encapsulates storage together with layered functionality, e.g. DBMS, ERP, or CRM
    - External data interfaces expose layered functionality, not encapsulated storage

# “Why”: combining self-op & hosted storage



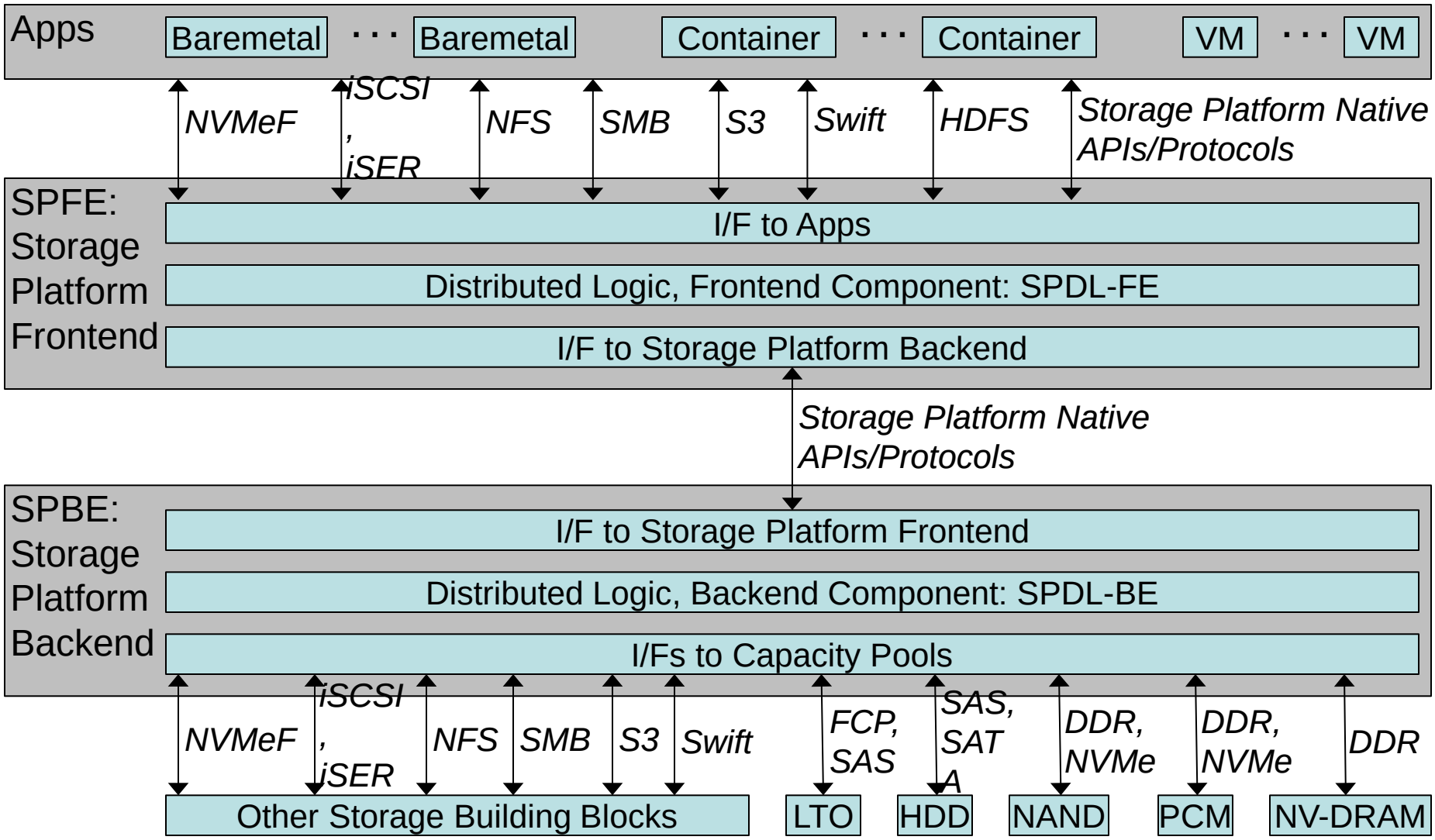
# Combining self-op & hosted storage

- Self-op strengths vs. Hosted
  - Many more options for geo-location of storage resources
    - May be able to move resources closer to endpoints
      - Better, lower-cost network connections
        - Better latency, jitter, throughput
      - May be able to reduce vulnerability to network outages
  - Lifecycle cost significantly lower for some use cases
  - Can use CapEx to help reduce OpEx
- Potential advantages vs. largest hosters
  - Smaller target for hackers
  - Potential for faster recovery from outages
    - May be able to focus more resources directly on prioritizing recovery of organization's specific storage footprint
    - Massive scale of largest hosters increases potential for massive outages
    - Largest hosters have lean staffs; during recovery from outages, human resources stretched thin across many client storage footprints

# Combining self-op & hosted storage, cont.

- Hosted strengths vs. Self-op
  - Lifecycle cost significantly lower for some use cases
  - Can use OpEx to help reduce CapEx
  - Convenience
  - Low-overhead provisioning
  - Elastic capacity
  - Built-in multi-datacenter
  - Operational excellence of largest hosters
    - Security
    - Availability
    - Data durability
- Strengths of combining Self-Op & Hosted
  - Many strengths of Self-Op & Hosted are fundamentally complementary
  - More options to optimize data placement across infrastructure elements
  - Risk reduction via platform diversity
    - Avoid technological monoculture

# Unbundled example: software components

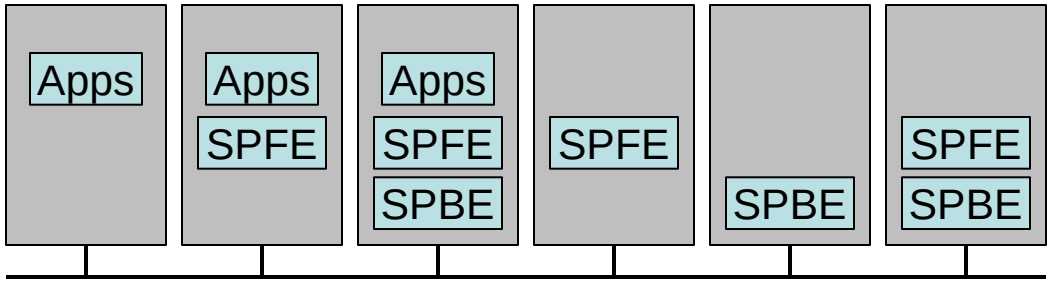


# Unbundled example: cont.1

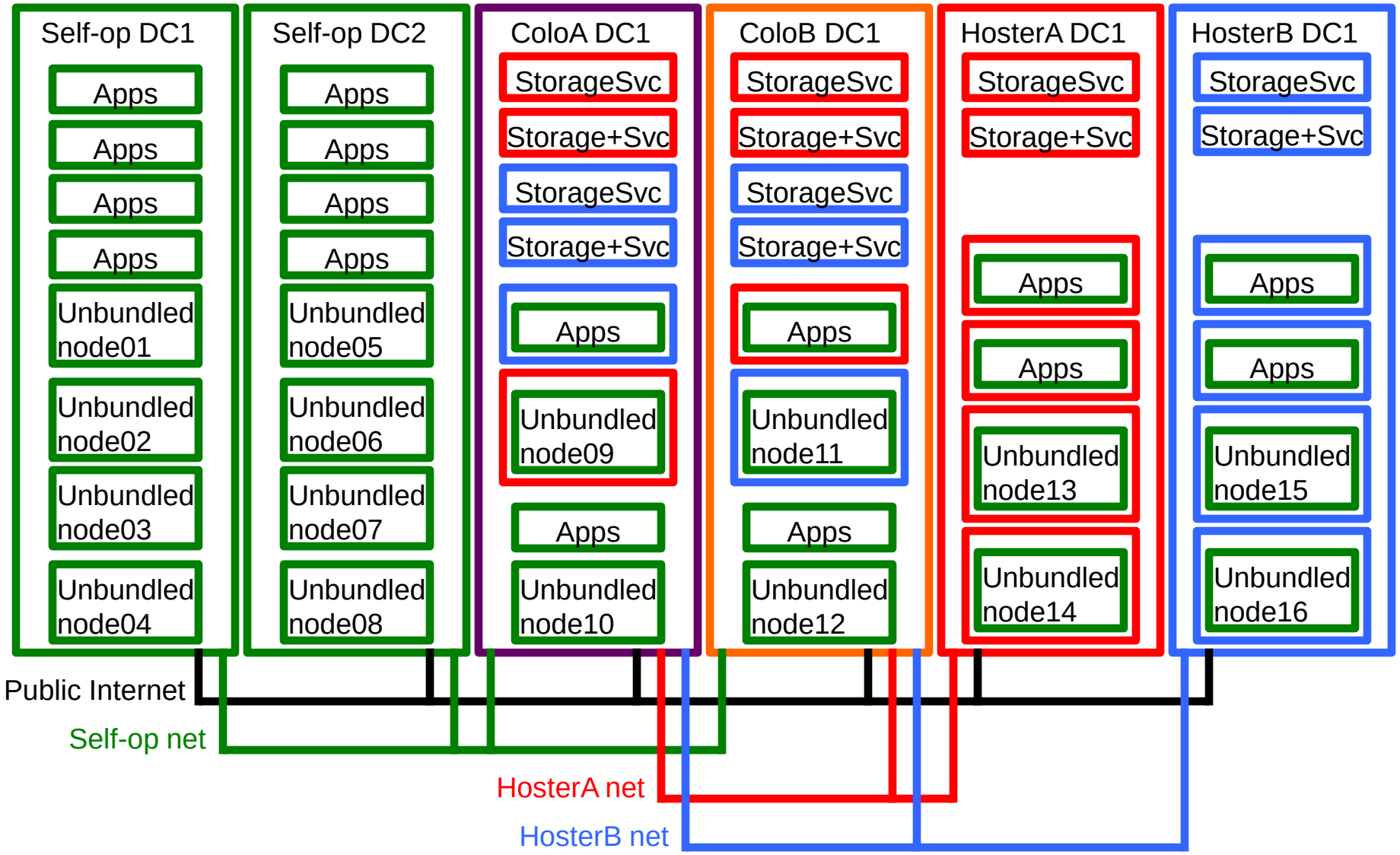
- Storage Platform Distributed Logic
  - Separate Frontend and Backend components: SPDL-FE, SPDL-BE
  - Platform resource management
    - Interfaces
      - API
      - CLI & GUI, built atop API
    - Platform control
    - Platform self-monitoring/alerting
    - Provisioning
    - QOS
    - Multi-tenancy
    - Multi-datacenter federation
  - Data security
    - Access management
    - Encryption
  - Data placement/movement
    - Data map
    - Auto-tiering
    - Rack & datacenter awareness
  - Data durability
    - Replication
    - Erasure coding
  - Data integrity
    - End-to-end checking
    - Auto-scrubbing
  - Data efficiency
    - Thin provisioning
    - Deduplication
    - Compression
  - Data services
    - Continuous Data Protection
    - Snapshots, clones
  - Performance
    - Caching
    - Concurrent/parallel multi-node requests
  - Analytics
    - Platform
    - Data

# Unbundled example cont.2

- Example node configurations



# Combining self-op & hosted storage



# Summary

- Storage infrastructure design is challenging optimization problem
  - Will benefit from future intelligent tooling/automation incorporating analytics
- Unbundled storage platforms increasingly competitive
  - Single deployment can simultaneously support:
    - Multiple server infrastructures, self-op and/or hosted
    - Scale-out
    - Multiple interfaces: Block, File, Object, Distributed-FS
    - Multiple data management services, across all interfaces
    - Multiple media pools, auto-tiering
- Hosted storage, storage-plus services also increasingly competitive
  - Radically more convenient than alternatives
  - Low-overhead provisioning
  - Elastic capacity
  - Built-in multi-datacenter
- Genius of the AND
  - Unbundled storage platforms plus hosted storage services
    - Can be highly complementary
    - Combinations can be superior for many use cases
    - Improving interconnection options making combinations more attractive

# Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

## Authorship History

**Craig Dunwoody, 2016/05/27**

## Additional Contributors

*Please send any questions or comments regarding this SNIA Tutorial to **tracktutorials@snia.org***