Separate vs. Combined Server Clusters for App Workloads & Shared Storage

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

- **Separate vs. combined server clusters for app workloads & shared storage**

  - Datacenter operators need scalable, high-availability infrastructure that provides processing capacity and shared-storage services for application workloads. One approach is to deploy a scale-out server cluster for application processing, and a separate cluster for shared storage. An alternate approach, sometimes called "hyper-converged", combines application processing and shared storage in a single scale-out cluster. This tutorial provides a simple framework for comparing implementations of scale-out server clustering for application processing and shared storage, and then presents some examples of potential pros and cons of the combined-cluster approach. While both approaches also include networking, the focus of this tutorial is on the application processing and shared storage aspects of these approaches.

This Tutorial Provides:

- Simple framework for comparing implementations of scale-out server clustering for application processing & shared storage
- Examples of potential pros & cons for separate vs. combined (“hyper-converged”) scale-out clusters for applications & shared storage

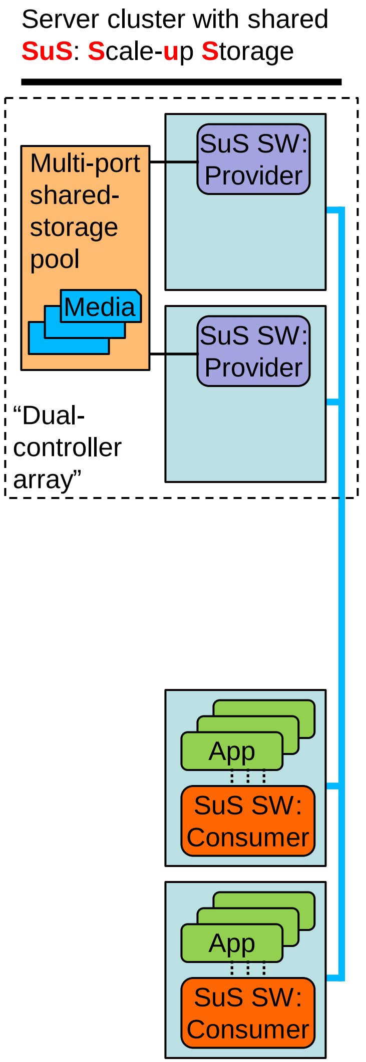
Why Server Clustering?

- **Availability**
  - Deliver services continuously, despite failures of individual hardware, firmware, and software components
    - Design-out single points of failure
- **Scalability**
  - Application processing performance
  - Shared-storage capacity
  - Shared-storage access performance

Shared Storage: Scale-up vs. Scale-out

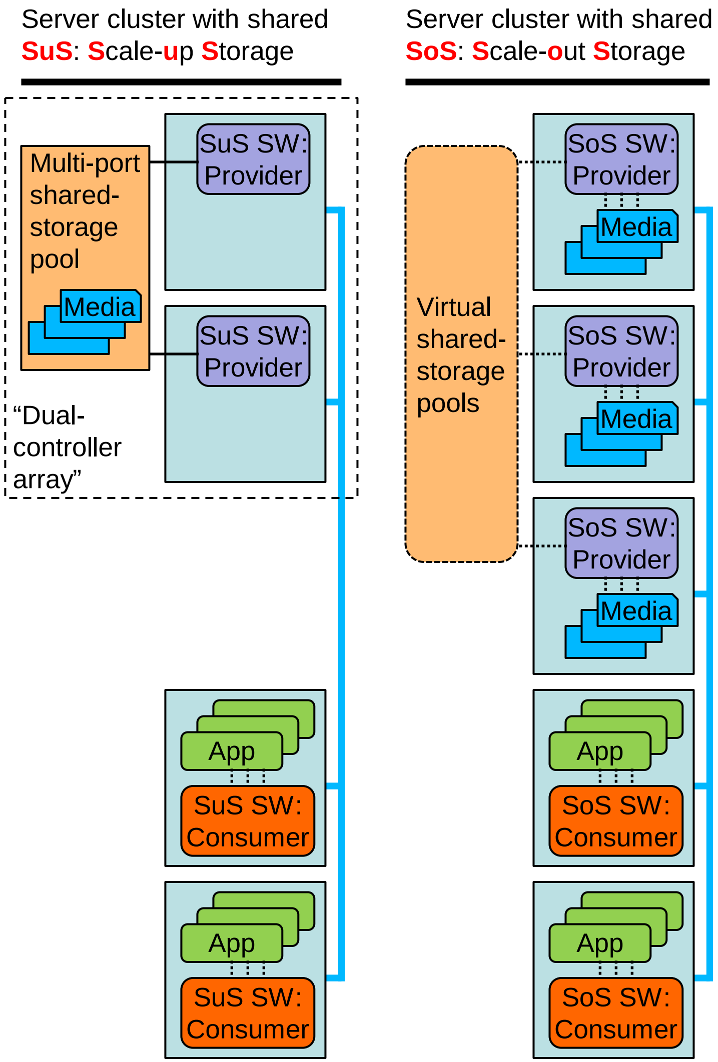
Scale-up storage (SuS)

- Multiple storage servers ("controllers"), most commonly two
- Physical shared-storage pool



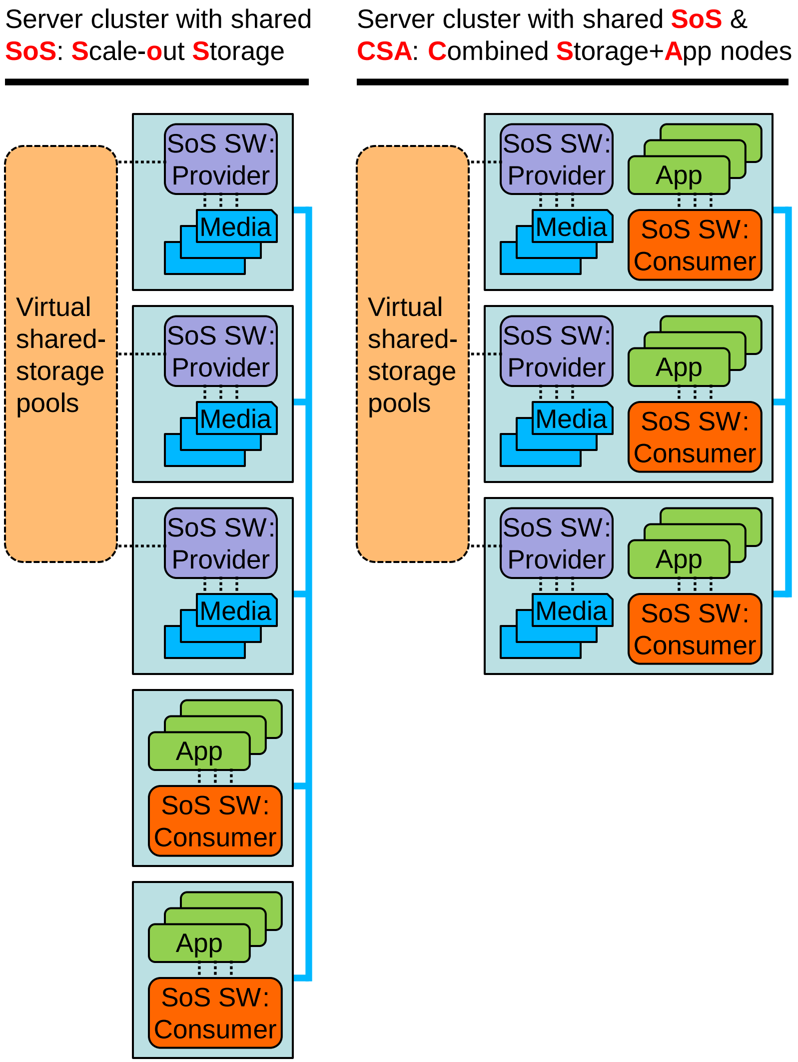
Shared storage: Scale-up vs. Scale-out

- **Scale-up storage (SuS)**
  - Multiple storage servers (“controllers”), most commonly two
  - Physical shared-storage pool
- **Scale-out storage (SoS)**
  - Virtual shared-storage pools
  - Enables simpler, lower-cost hardware configurations
    - No specialized networking hardware for storage
  - Enables higher scalability & lower costs
    - General-purpose networking, e.g. Ethernet: more ports per switch module, lower cost per port
    - Lower-cost access performance, capacity



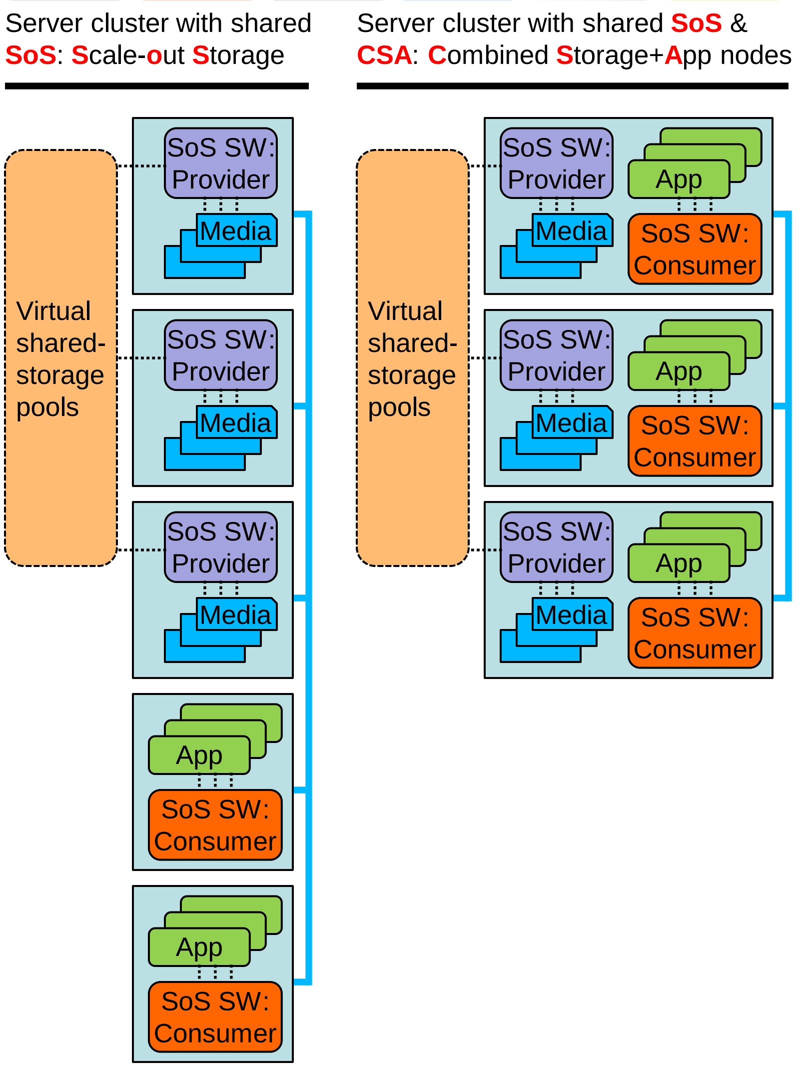
Separate vs. Combined Scale-out Clusters for Apps & Shared Storage

- Many commercial & Open Source implementations of SoS
  - Ongoing acceleration of SoS development & innovation
  - Mix of young & established SoS implementations (up to 10+ years)
  - Design space still lightly explored
- CSA: Combined Storage+App nodes ("hyper-converged")
  - Feature that a SoS implementation may include
  - Rapidly growing number of SoS implementations supporting CSA, as optional or required node config

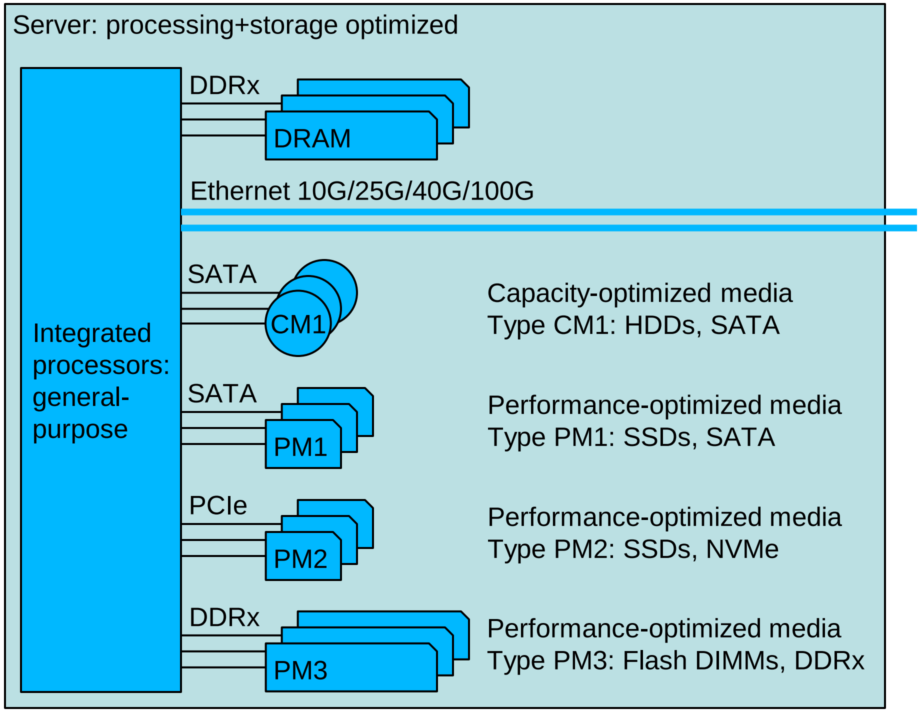


Roadmap: Rest of This Presentation

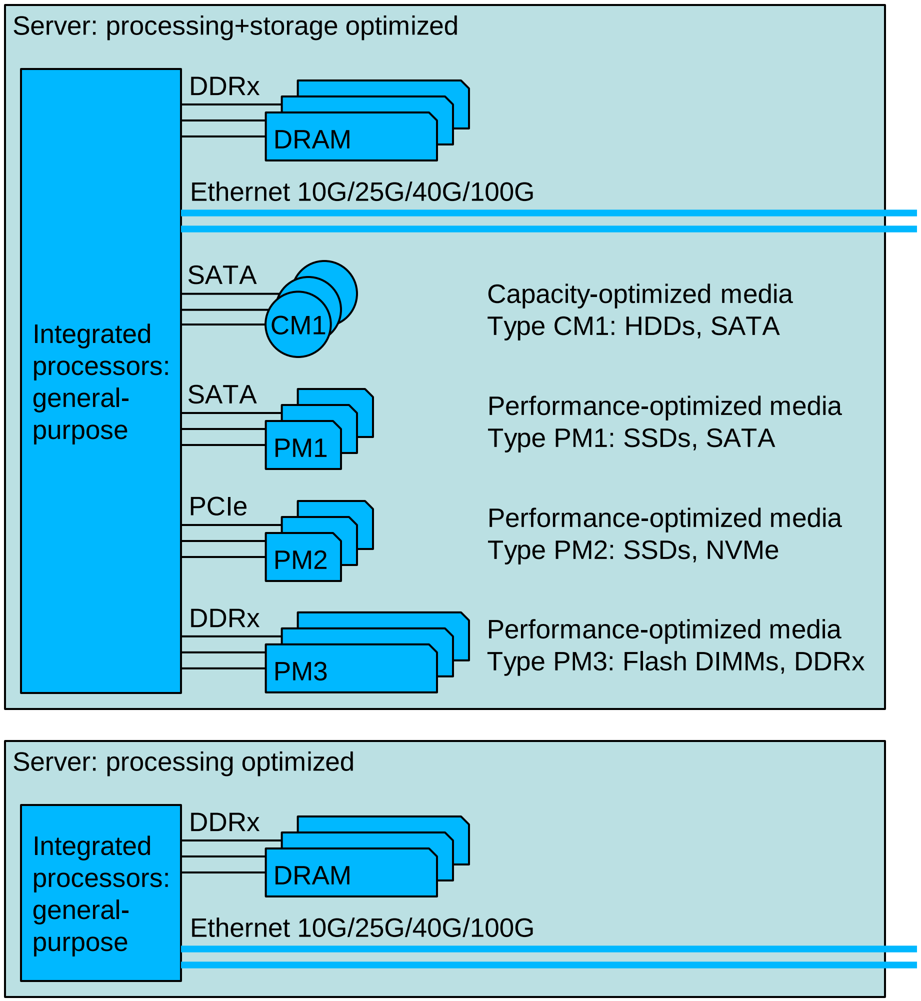
- More-detailed example SoS cluster
- Simple questions for specific SoS implementations
- Combined Storage & App nodes: some examples of potential pros & cons
- Closing thoughts



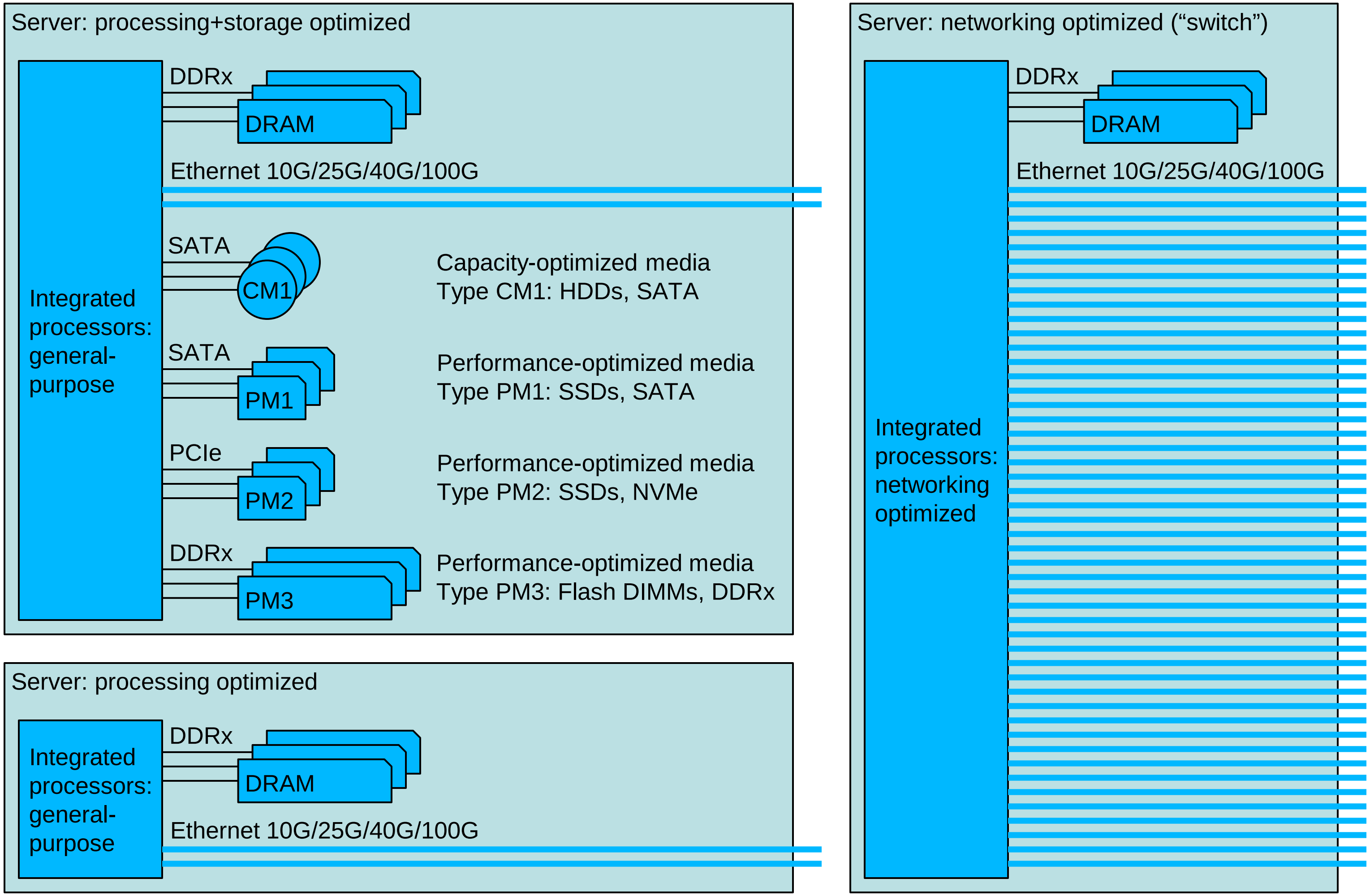
Example Cluster: Servers Processing+Storage Optimized



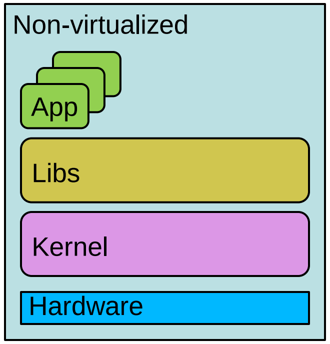
Example Cluster: Servers Add: Processing Optimized



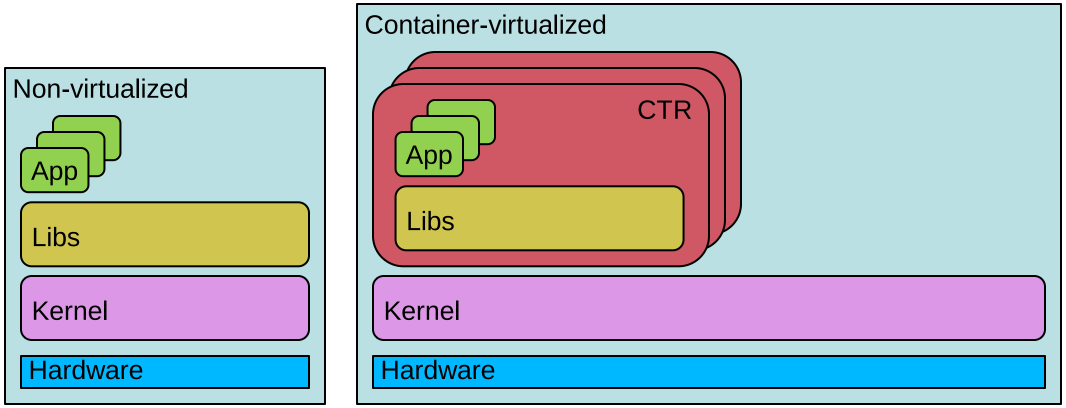
Example Cluster: Servers Add: Networking Optimized



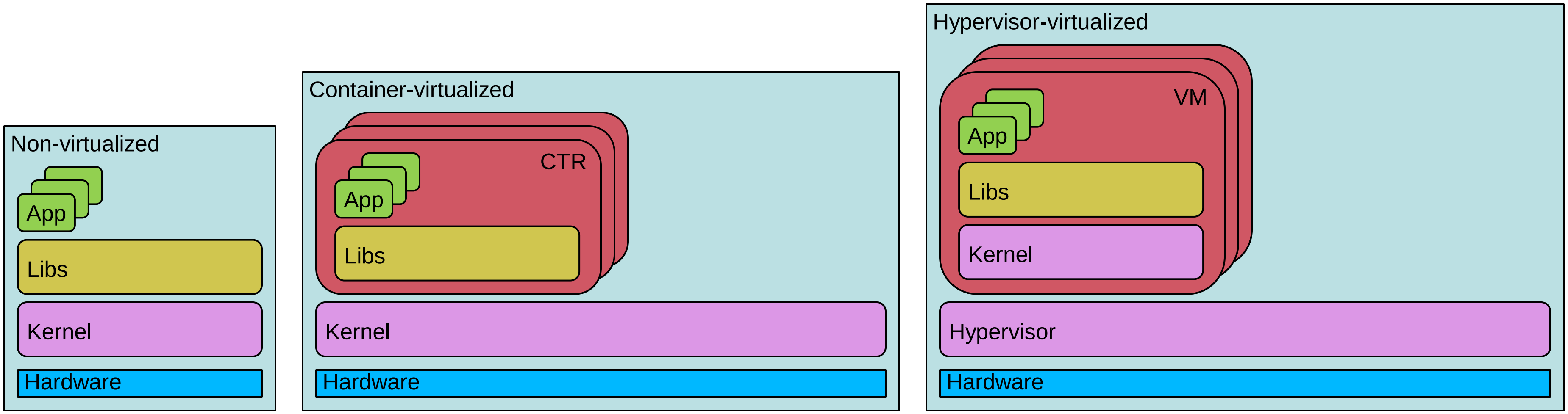
Example Cluster: Software Stacks
Standard Non-Virtualized



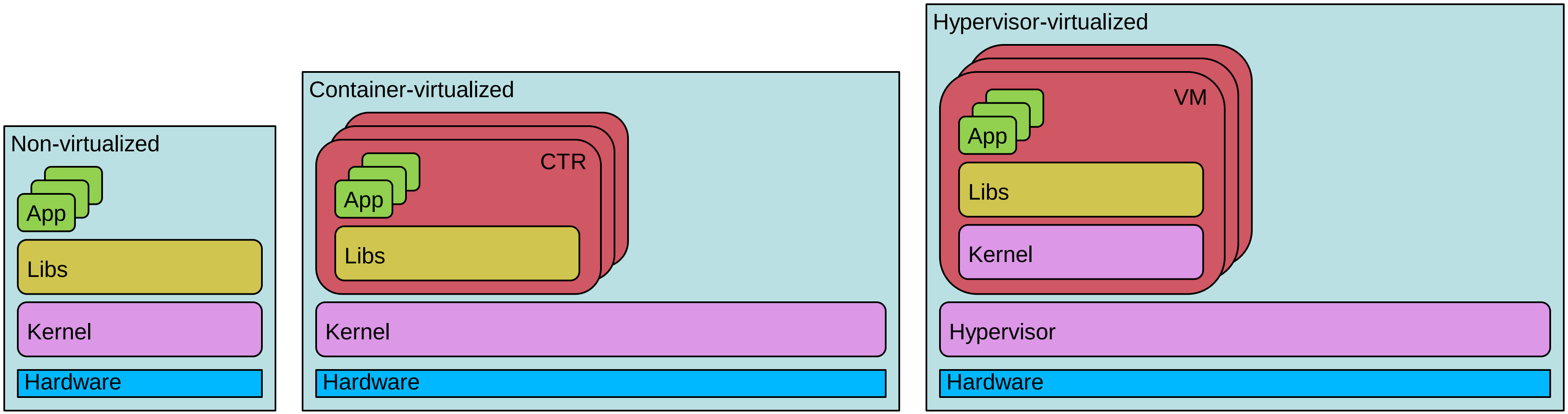
Example Cluster: Software Stacks
Add: Standard Container-Virtualized



Example Cluster: Software Stacks
Add: Standard Hypervisor-Virtualized

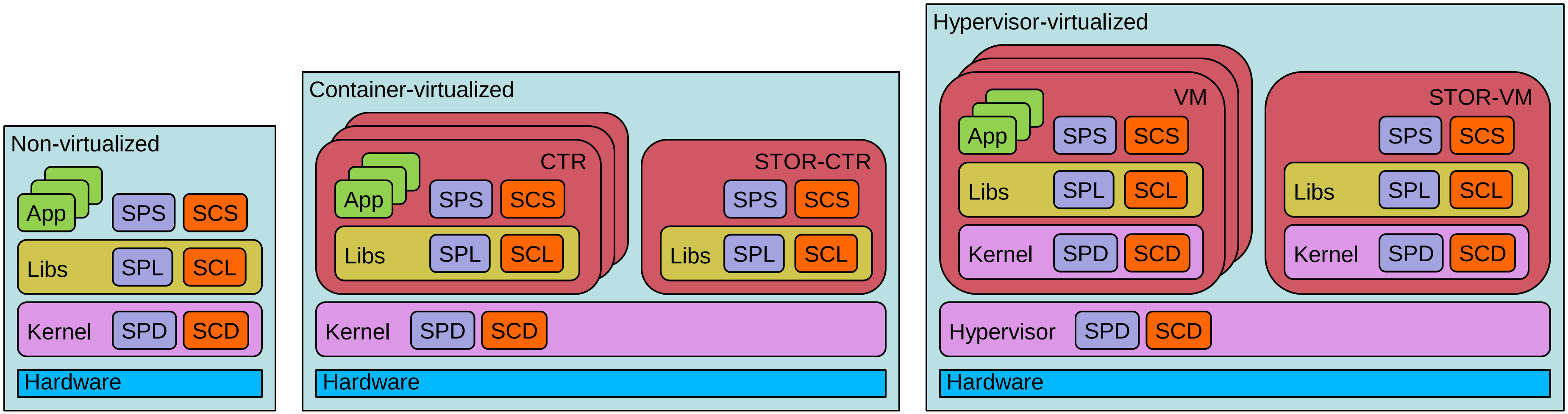


Example Cluster: Software Stacks
Add: Some Example SoS Hooks



SPS	SoS Provider Service User-space daemon	SCS	SoS Consumer Service User-space daemon
SPL	SoS Provider Library User-space	SCL	SoS Consumer Library User-space
SPD	SoS Provider Driver Kernel-space	SCD	SoS Consumer Driver Kernel-space

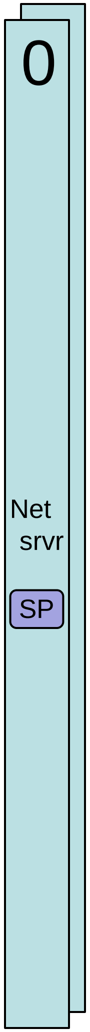
Example Cluster: Software Stacks
Add: Example SoS Hooks in Stacks



SPS	SoS Provider Service User-space daemon	SCS	SoS Consumer Service User-space daemon
SPL	SoS Provider Library User-space	SCL	SoS Consumer Library User-space
SPD	SoS Provider Driver Kernel-space	SCD	SoS Consumer Driver Kernel-space

Example Cluster: Node Configs

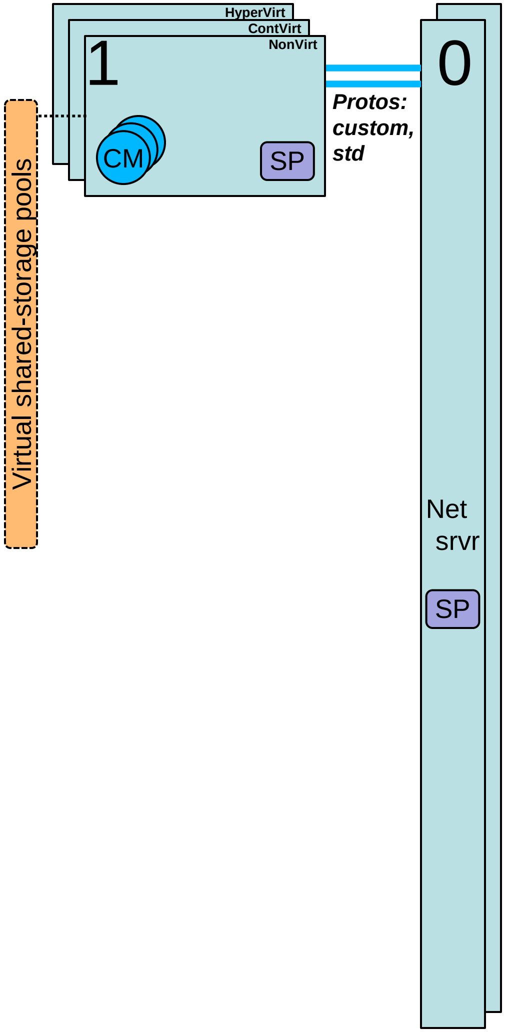
- Config 0: Network
  - Two instances, to eliminate single point of failure
  - Optional SoS Provider software



Example Cluster: Node Configs

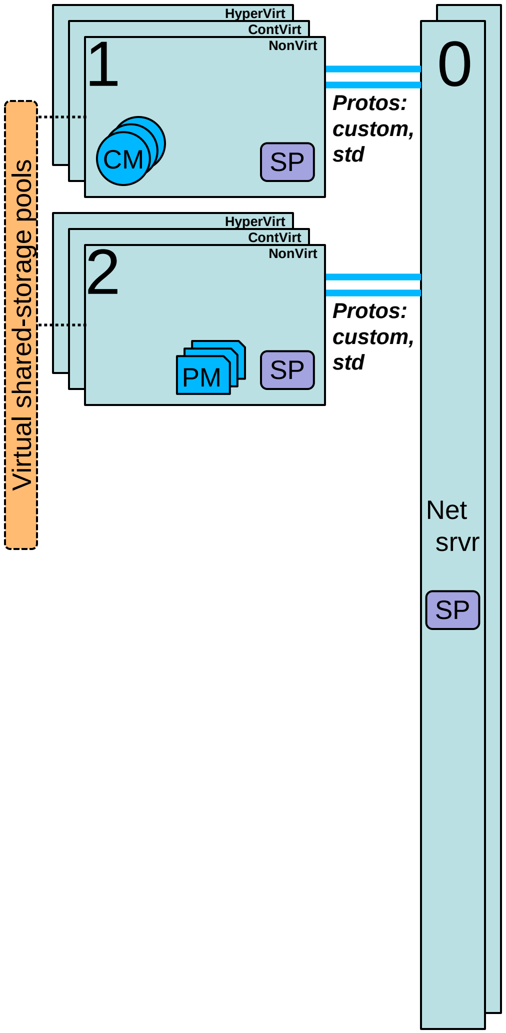
Config 1: Storage

- Versions: NonVirt, ContainerVirt, HypervisorVirt
- Capacity-optimized media
- SoS Provider software



Example Cluster: Node Configs

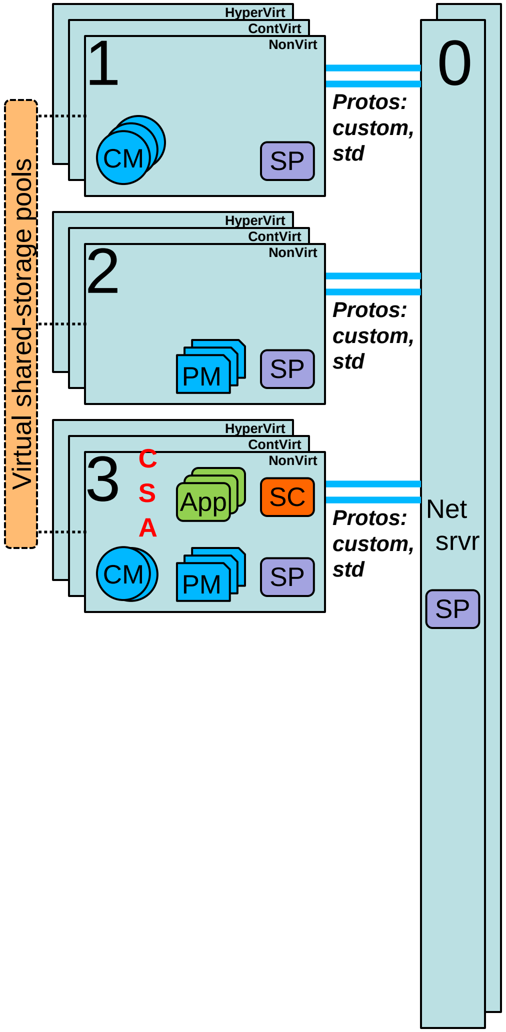
- Config 2: Storage
  - Versions: NonVirt, ContainerVirt, HypervisorVirt
  - Performance-optimized media
  - SoS Provider software



Example Cluster: Node Configs

Config 3: CSA = Combined Storage & Apps

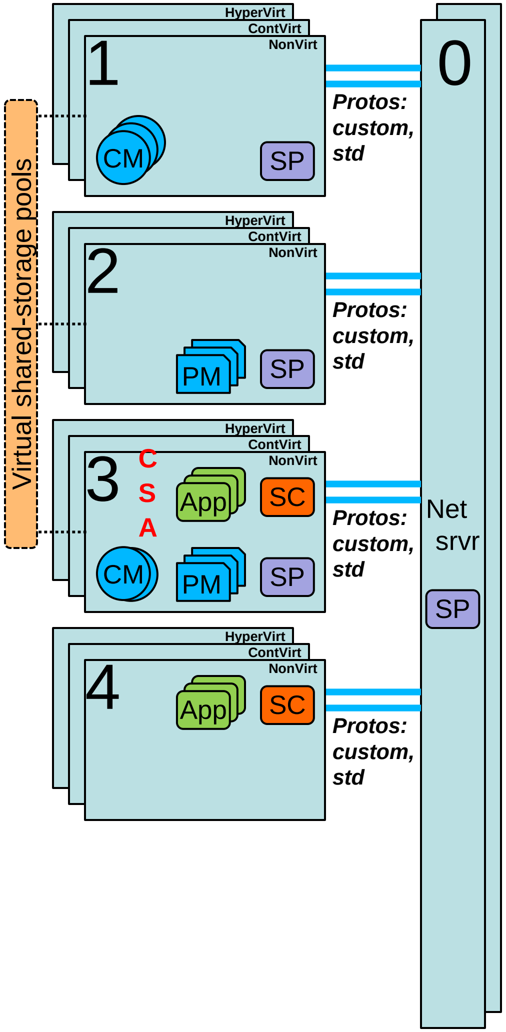
- Versions: NonVirt, ContainerVirt, HypervisorVirt
- Capacity-optimized media
- Performance-optimized media
- SoS Provider software
- Apps
- SoS Consumer software



Example Cluster: Node Configs

Config 4: Apps

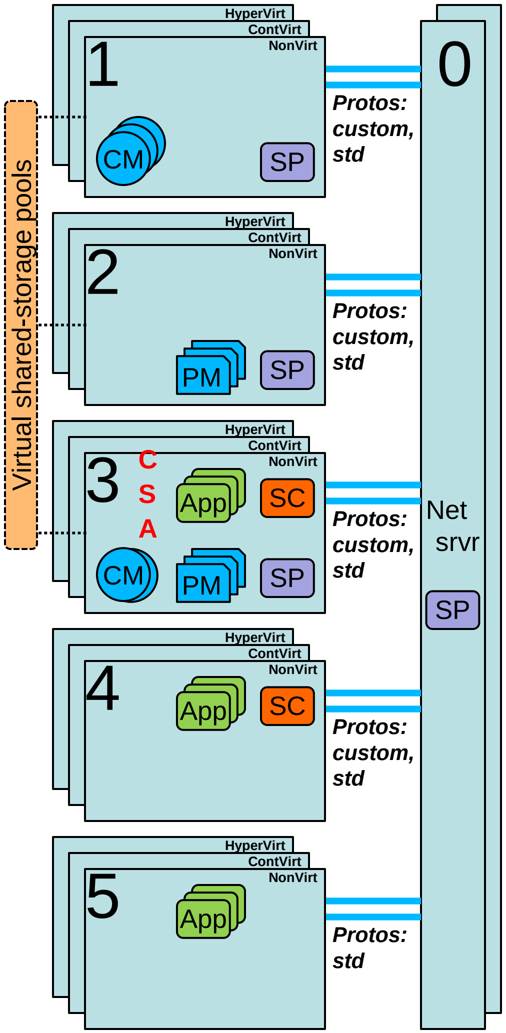
- Versions: NonVirt, ContainerVirt, HypervisorVirt
- Apps
- SoS Consumer software



Example Cluster: Node Configs

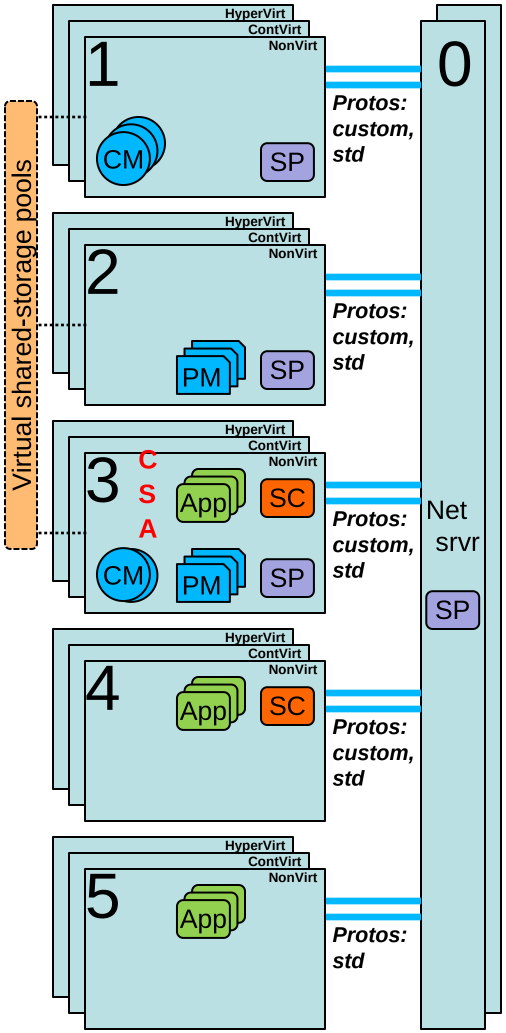
Config 5: Apps

- Versions: NonVirt, ContainerVirt, HypervisorVirt
- Apps
- No SoS software
- Uses only standard storage-access protocols, e.g. NFS, iSCSI



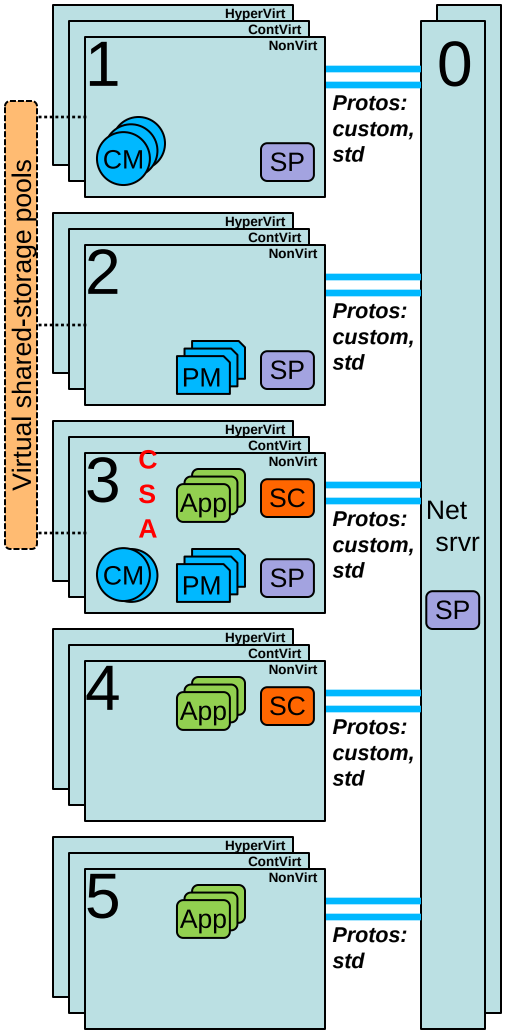
Key Design Choices for Specific SoS Implementations

- **Expect:**
  - Many differences between implementations
  - Lots of “No” & “On roadmap”
- **Node configs from example cluster**
  - Combined Storage & App: 3 out of 16 configs
- **Storage media pools**
  - HDD, SATA/SAS SSD, NVMe SSD, Flash DIMM
  - Auto-tiering, caching
  - Original / primary design center
- **Storage APIs**
  - Block, file, object, VM-image
  - Features, e.g. POSIX byte-range locks
  - Consistency models



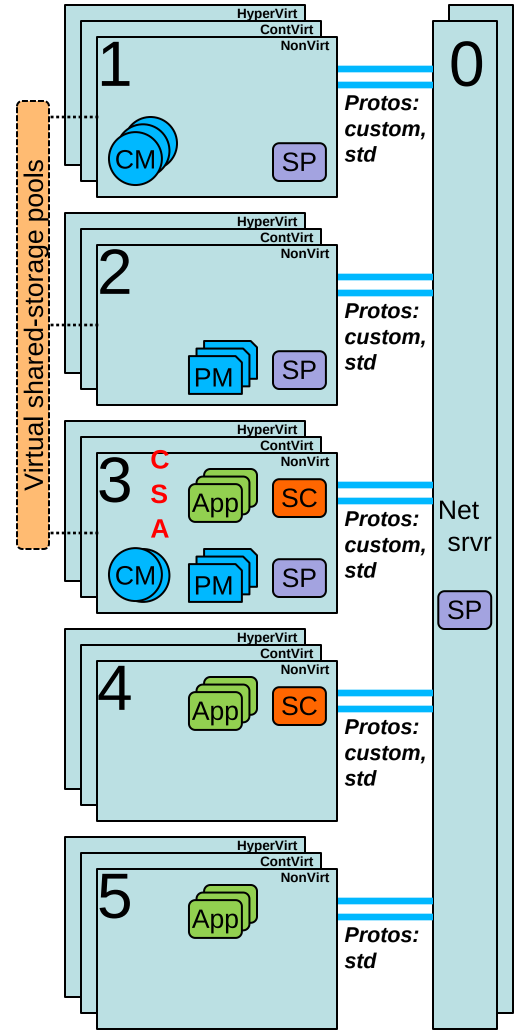
Key Design Choices for Specific SoS Implementations

- Storage wire protocols
  - Block, e.g. iSCSI
  - File, e.g. NFS v.x
  - Object, e.g. Swift
  - Custom (specific to SoS implementation)
- Data durability, e.g. replication, erasure code
- Data efficiency, e.g. dedupe, compression
  - Inline, post-process
- Data services, e.g. snapshots, clones
- Hardware configs
  - Hardware Compatibility List for end-user integration
  - Integrated HW+SW appliances



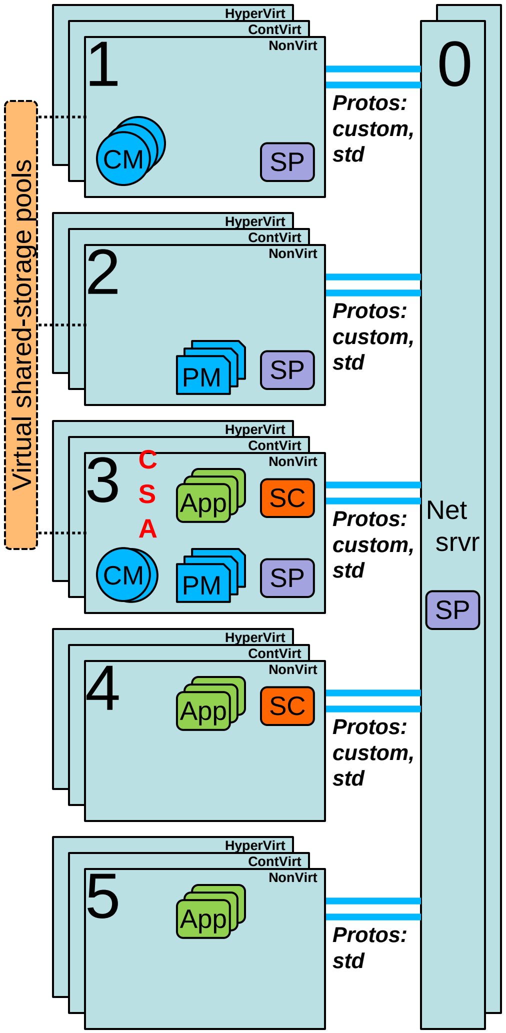
Combined Storage & App Nodes
Some Examples of *Potential* Pros & Cons

- Categories
  - Scalability
  - Efficiency
  - Maintainability
  - Fault exposure
  - Cost-effectiveness
  - Security & stability
- *Potential* may become *actual*
  - Based on specific use case, SoS implementation
- Caveats
  - Narrow focus on CSA: single architectural element
    - Just one of *many* aspects of complete SoS system
  - *Far* from a comprehensive list!



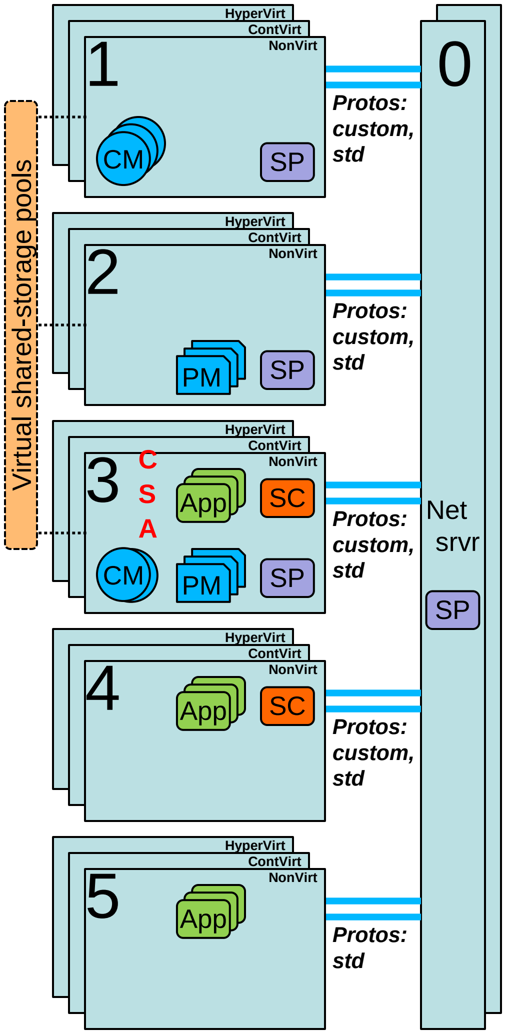
Combined Storage & App Nodes Scalability: Pro

- Smaller minimum cluster size
  - SoS clusters typically need 3+ nodes
    - CSA avoids need for additional app nodes in minimum config
  - Caveat
    - Minimum CSA cluster might not natively provide all required storage services
      - Example: CSA cluster might natively implement storage only for Virtual Machine datastores
      - If app VMs on CSA cluster also need shared file storage (e.g., for home directories), must provide via other mechanism, such as another VM (maybe lacking desirable data services) or separate NAS system
- Multi-resource scaling: lower-cost, smaller increments
  - Add single node: simultaneously grow app processing, storage performance & capacity



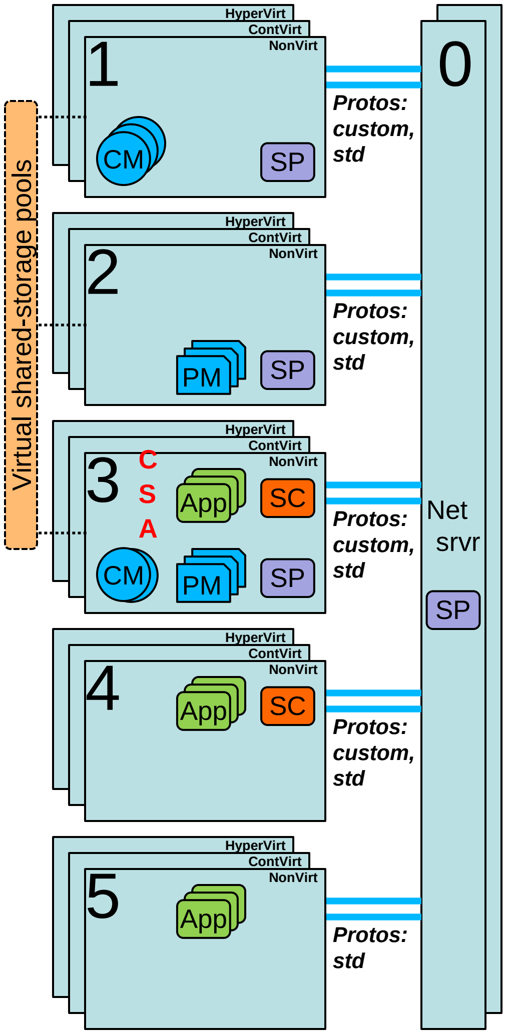
Combined Storage & App Nodes Scalability: Con

- Resource imbalance when scaled
  - To scale hardware resources most efficiently, may need separate scalability of:
    - Storage capacity
    - Storage access performance
    - Application processing performance
  - Best for efficient scalability: SoS implementations that enable all node configs from example cluster
    - With current server packaging, some use cases for scale-out clusters want more app nodes than storage nodes
    - Example: well-known service provider as of Jan 2015
      - ~104K app nodes
      - ~15K storage nodes
    - CSA config might end up with more storage than needed when adding nodes to scale-out app processing performance



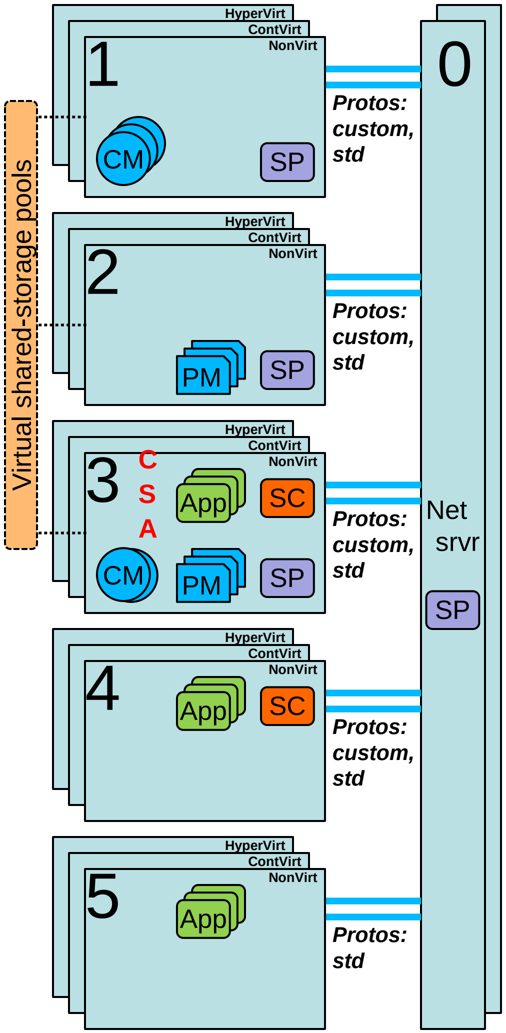
Combined Storage & App Nodes Scalability: Con

- Resource imbalance when scaled
  - Caveats
    - For some use cases, this matters a lot
    - In other cases, might be waste of effort to try to hyper-optimize hardware resource balance at this level, esp. for small cluster sizes, and for unpredictable workloads



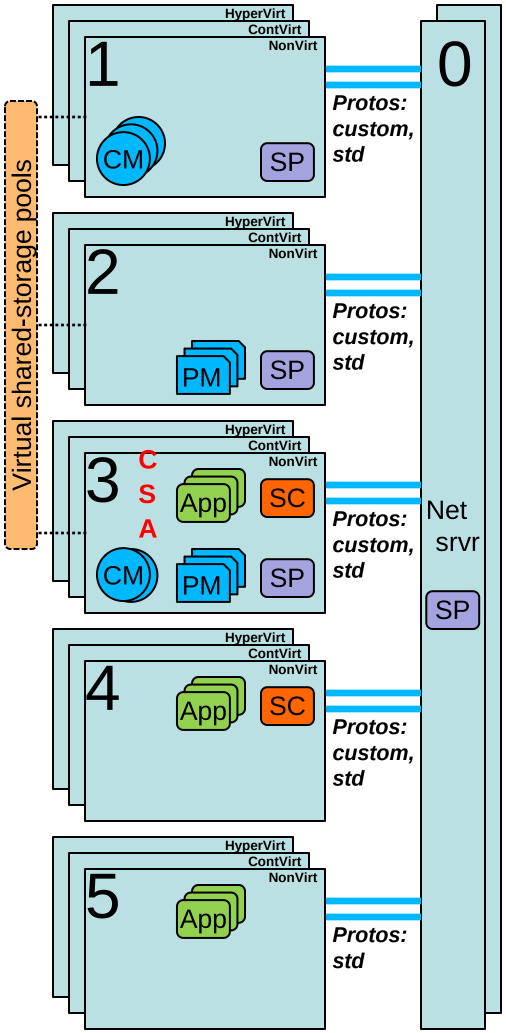
Combined Storage & App Nodes Efficiency: Pro

- Keep all hardware busy doing useful work
  - Without CSA, storage nodes may be silos of idle server resources (processor+cache, DRAM) when shared-storage demand is low
    - With CSA, can use these resources for app workloads
  - Caveat
    - Performance-optimized storage media (e.g., NAND) in SoS node might cost more than all other node components combined
    - Most valuable use of otherwise-idle SoS node resources might be to optimize effectiveness of node's most expensive media, e.g., by computing SoS cluster internal analytics to help drive auto-tiering, etc.



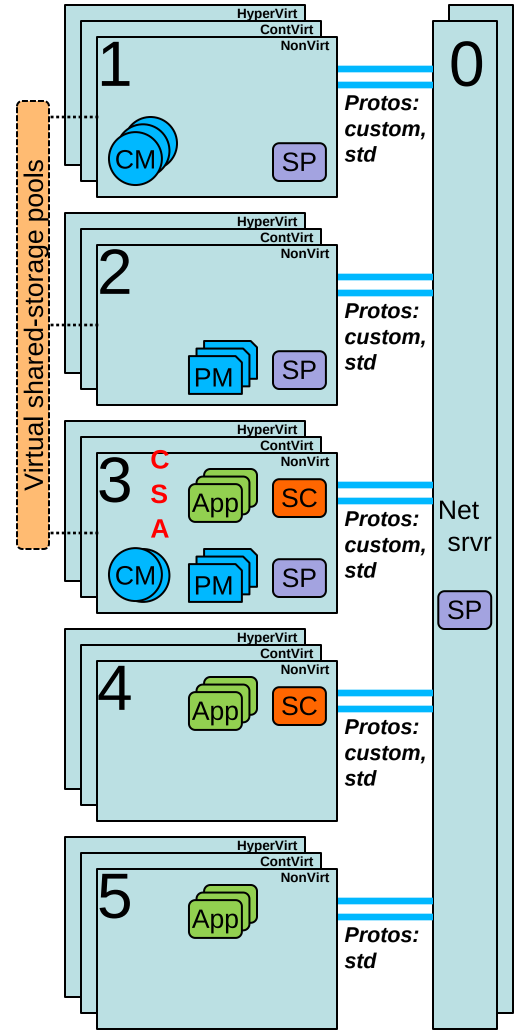
Combined Storage & App Nodes Efficiency: Pro

- Can physically co-locate processing & data
  - Perform some storage ops locally within node
    - Reduce network round-trips & associated latency
  - Example characteristics of best-fit workloads
    - All processing, and storage working set, fit entirely within single node
    - No storage shared with any other workload
    - Storage access dominated by reads
    - Storage writes non-critical; don't need synchronous replication to another node
    - Long runtime
    - Multiple concurrent instances of single app
    - Processing & storage packaged together, e.g. VM images
    - Overall workload performance highly sensitive to storage-access latency, esp. for reads
    - Cluster-aware; can drive co-location via APIs



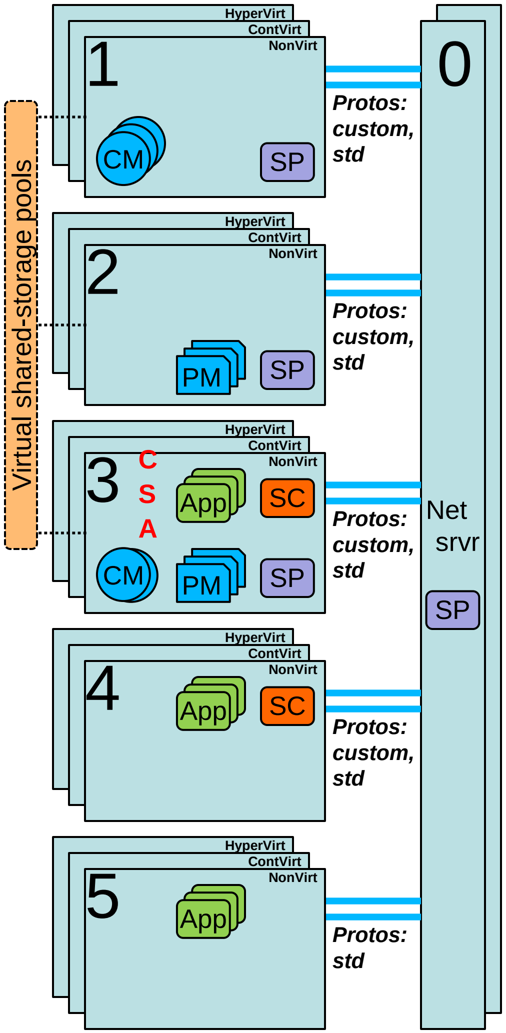
Combined Storage & App Nodes Efficiency: Pro

- Can physically co-locate processing & data
  - Example use cases
    - Virtual Desktop Infrastructure
      - Poster child for CSA benefits
      - Many characteristics match examples for best-fit
    - Read-intensive distributed parallel analytics
      - Move computation to data, not vice versa
    - Storage-latency intolerant workloads
      - E.g., some financial-services apps
  - Managing placement
    - Data objects, executables, containers, VMs
    - Manual
      - Sensing/control via GUI, CLI, scripting
    - Scheduling automation & cluster-aware workloads
      - Sensing/control via APIs



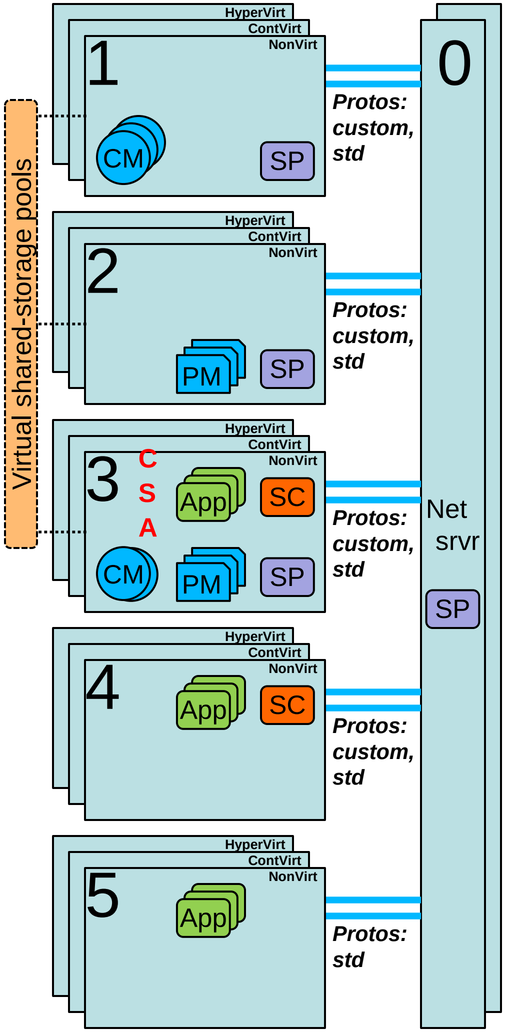
Combined Storage & App Nodes Efficiency: Pro

- Can physically co-locate processing & data
  - Numerous caveats
    - Many workloads very different from best-fit examples
      - Might not benefit much or at all from co-location with data
    - Co-locating processing, data creates additional data-management constraints & challenges for CSA implementors
      - E.g., if/when/how to move data after workload migrates to different node
      - Some high-profile CSA implementations have chosen to not relocate data after initial placement
    - When reading data from remote node, latency-mitigation techniques such as caching & prefetching to DRAM can be highly effective for many workloads
    - For safety, storage writes must be synchronously replicated to another node, so cannot avoid a network round-trip, even if data co-located with workload



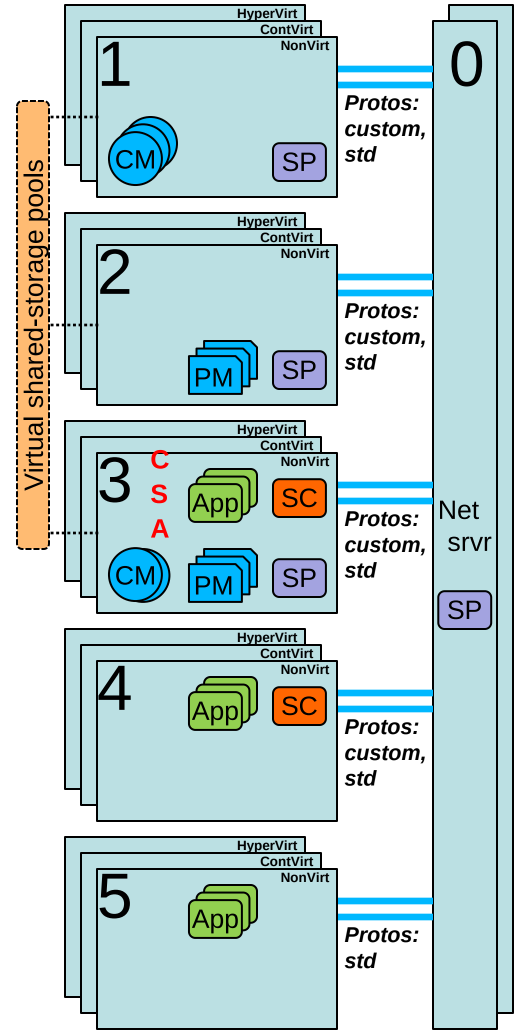
Combined Storage & App Nodes Efficiency: Pro

- Can physically co-locate processing & data
  - Numerous caveats
    - In many cases, network round trip latency not disastrously large relative to media latency
      - Network & media latencies continuing to drop in successive technology generations
      - Example measurements from 2014
        - NVMe SSD latencies, 4K random write: 120-250 usec
        - 10G Ethernet round-trip, user space: 40 usec
        - NVMe over 40G Ethernet: round-trip 8 usec higher than local



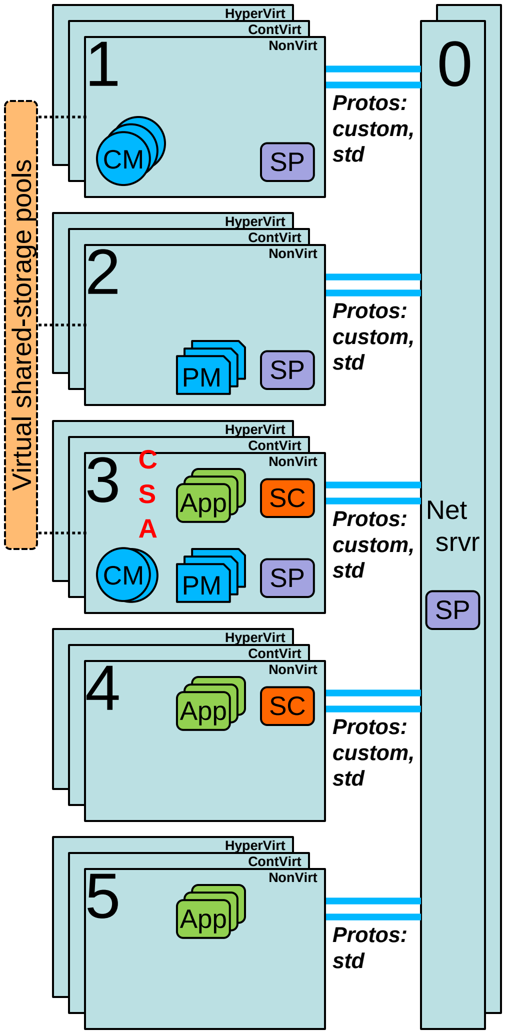
Combined Storage & App Nodes Efficiency: Con

- Bottlenecks with max-performance media
  - Current-generation storage media modules span wide ranges of cost, capacity, performance
    - SATA HDD: $/capacity, baseline performance
    - SATA SSD: $$/capacity, +performance
    - NVMe SSD, $$$/capacity, ++performance
    - Flash DIMM, $$$$/capacity, +++performance
  - For some use cases, can meet aggregate cluster-wide shared-storage performance requirement most cost-efficiently using max-performance media (currently NVMe SSDs, flash DIMMs), instead of larger # of lower-performance modules



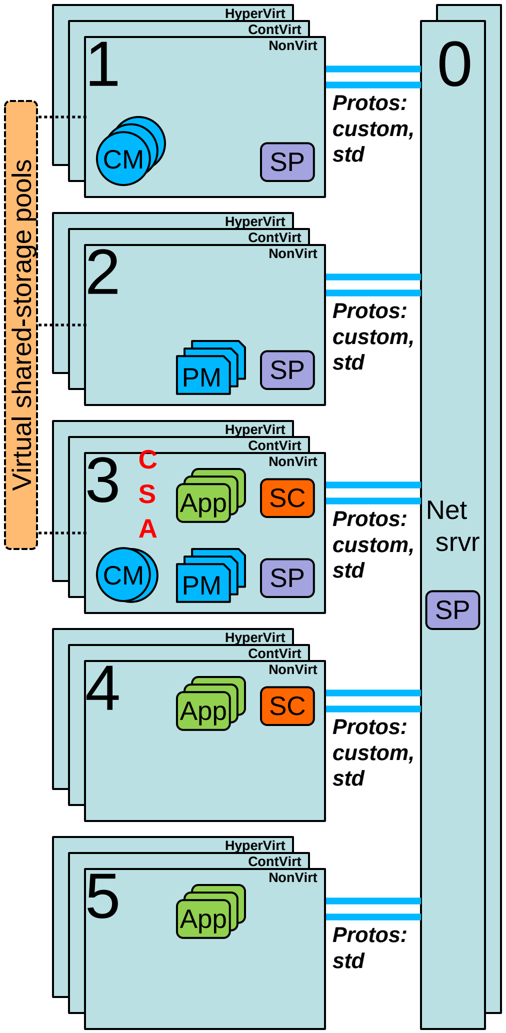
Combined Storage & App Nodes Efficiency: Con

- Bottlenecks with max-performance media
  - Max-performance media can be cost-efficient only if driven to full performance potential by host CPUs, network links
  - A current-generation server CPU & 10+G Ethernet link can barely deliver enough performance to drive a ***single*** current-generation max-performance media module to full performance when running only shared-storage services, & not also running application workloads
  - Accordingly, CSA node configs can be inefficient for max-performance media; in some cases would need larger total #nodes to deliver required aggregate shared-storage performance
- Across successive future technology generations, media performance might improve faster than CPU & network performance
  - Would expand range of use cases where CSA configs are inefficient



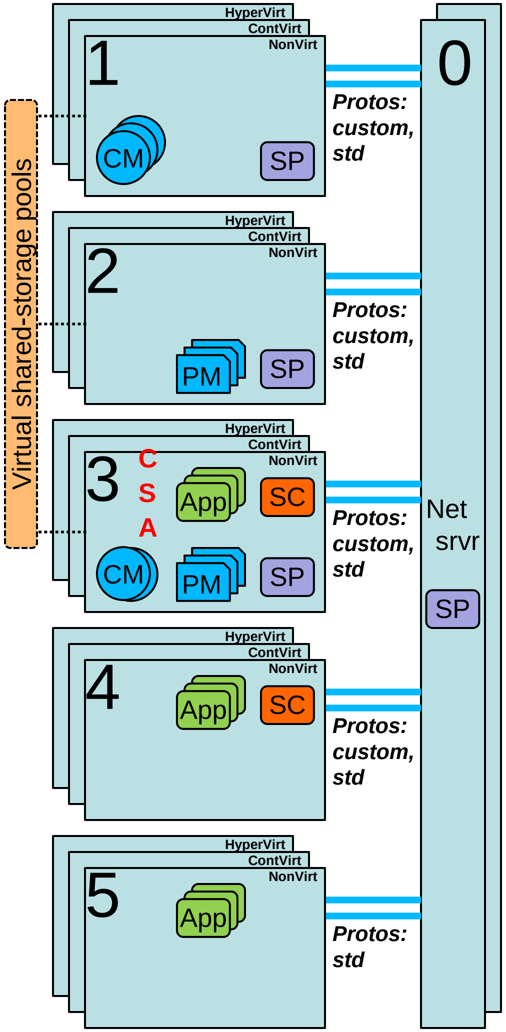
Combined Storage & App Nodes Efficiency: Con

- Bottlenecks with max-performance media
  - Additional perspectives:
    - Current-gen max-performance media modules can support far more shared-storage load than could be generated by workloads running locally in host node
    - Accordingly, to be cost-efficient these media modules require aggregation of shared-storage load across multiple app nodes, via network
    - CSA aims to optimize storage performance by keeping storage-access traffic ***off*** the network. Conversely, storage-only nodes with max-performance media optimize performance by putting storage-access traffic ***on*** the network
    - CSA works best for "shared storage" when that storage is not actually being shared



Combined Storage & App Nodes Efficiency: Con

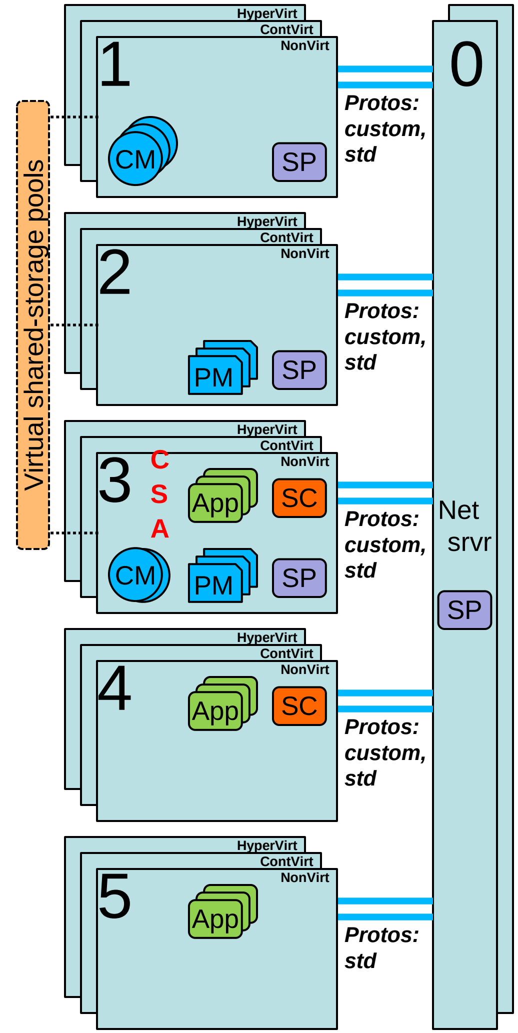
- Variability of shared-storage performance
  - Apps can be "noisy neighbors" for shared-storage services
  - Can have a negative effect on all workloads using these services



Combined Storage & App Nodes Efficiency: Con

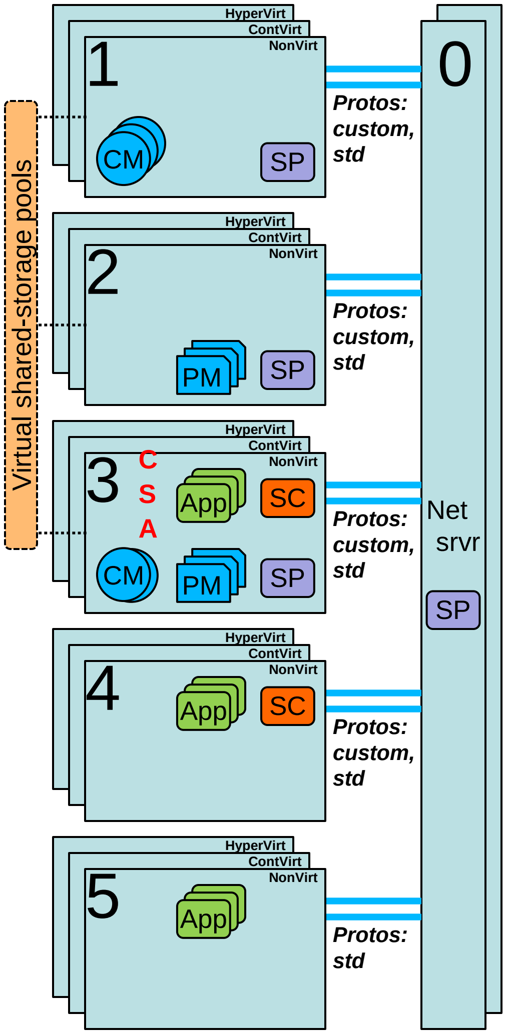
Software-stack constraints

- In a CSA node, need to support general-purpose application workloads may force use of software-stack elements that impair shared-storage services for all workloads
  - Example: because of application workload requirements, on a CSA node shared-storage services may be forced to run in a virtual machine on top of a general-purpose hypervisor, which might restrict I/O performance relative to running directly on hardware
  - On a dedicated storage-only node, entire software stack can be optimized exclusively for shared-storage services



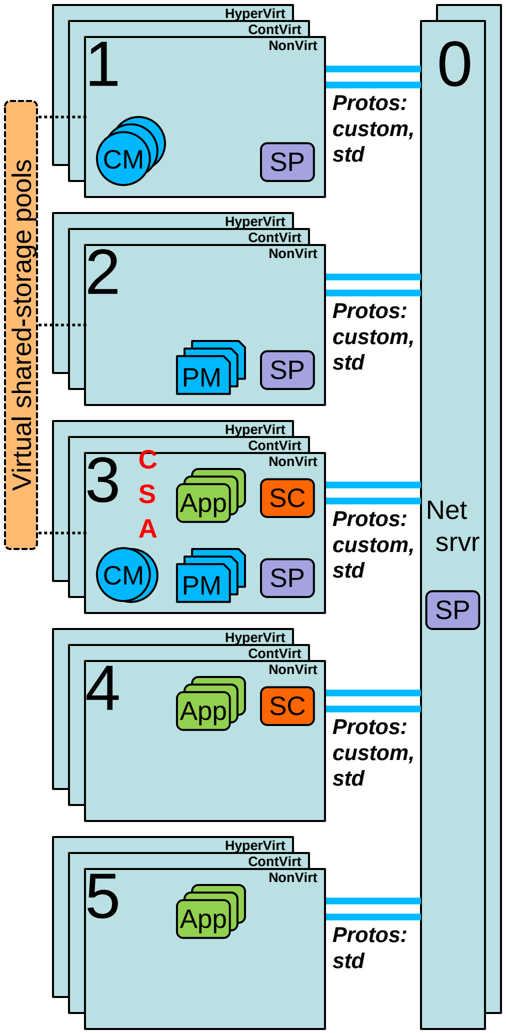
Combined Storage & App Nodes Efficiency: Con

- Net result
  - Just like people, servers can be less efficient when multi-tasking -- in the case of CSA, between providing shared storage services based on directly attached media, & running application workloads
  - In some cases, CSA may require more nodes & storage modules for same aggregate sustained storage performance



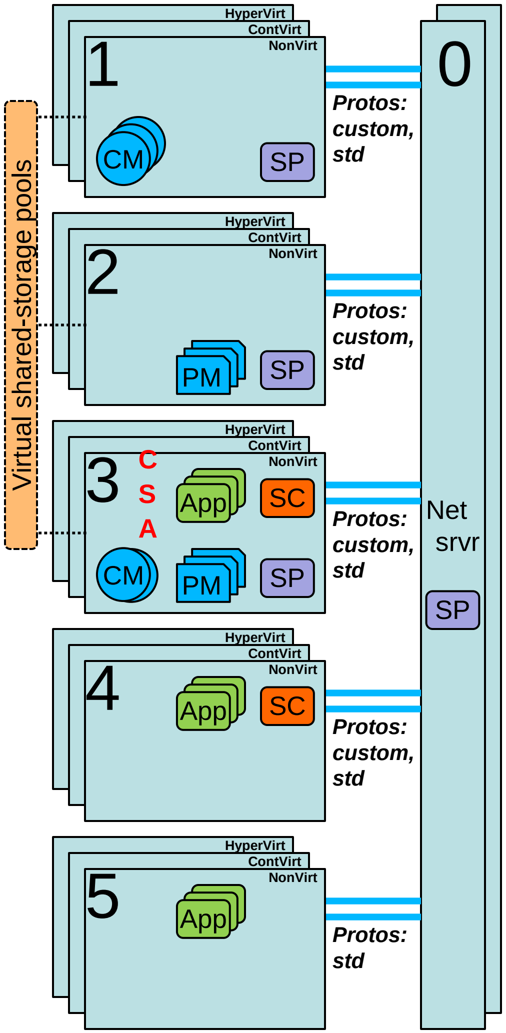
Combined Storage & App Nodes Maintainability: Pro

- Uniform software & hardware configs across cluster
  - Common management tools, software configurations, maintenance procedures across all nodes



Combined Storage & App Nodes Maintainability: Con

- When scaled, increases #nodes with persistent data
  - Node is basic unit of hardware maintainability
  - Stateless node: in virtualized environment, can evacuate workloads & bring down for maintenance with relatively small impact
  - Node containing persistent data: bringing down for maintenance has larger impact, affecting shared-storage services used by all workloads
  - Separation of concerns with separate storage & app nodes
    - If storage problem, can bring down storage node, not apps
    - If app problem, can bring down app node, not storage



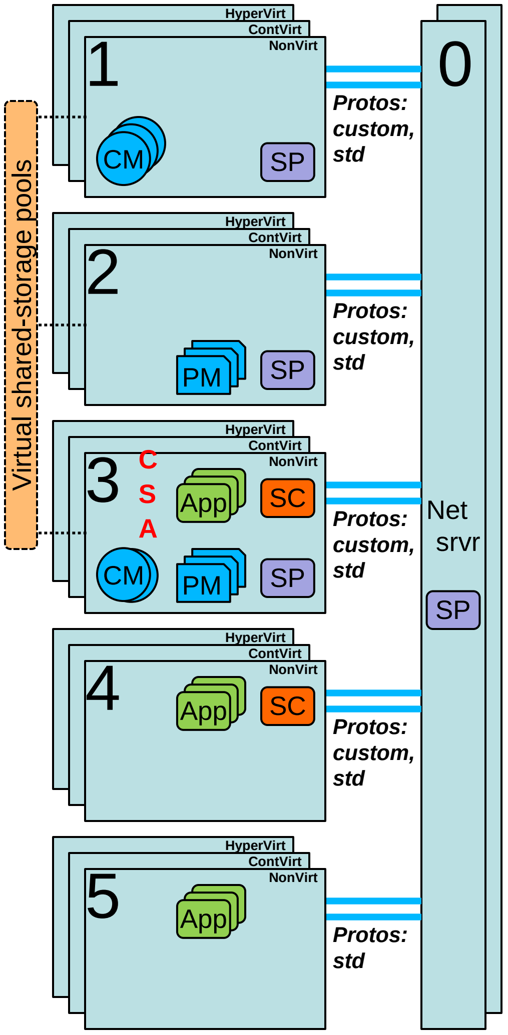
Combined Storage & App Nodes Fault exposure

Pro

- When scaled, shared-storage capacity & performance spread across larger number of smaller fault domains
- In some cases (e.g., minimum-size clusters), *smaller* total # of nodes & hardware components; less total fault exposure

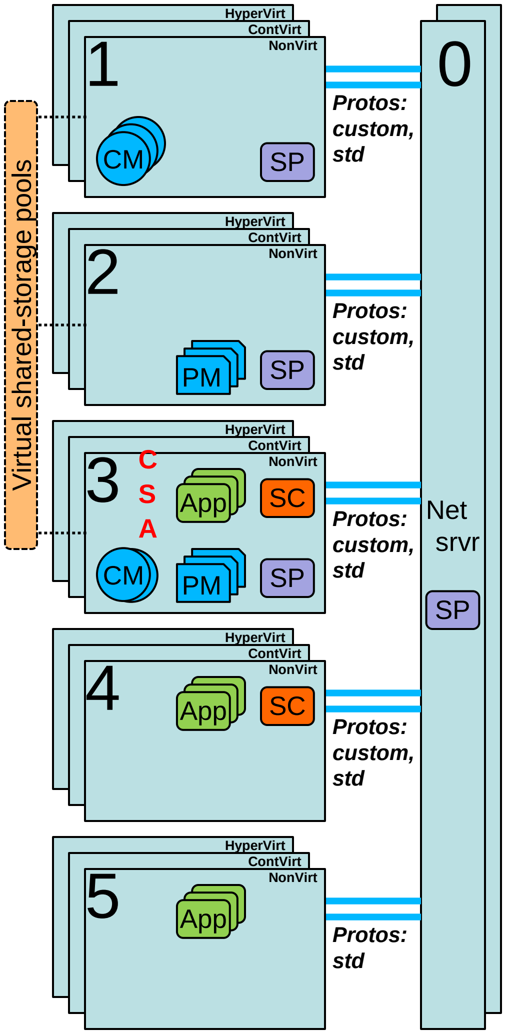
Con

- In some cases (e.g., using max-performance media to meet aggregate storage-performance requirement), *larger* total # of nodes & hardware components; more total fault exposure



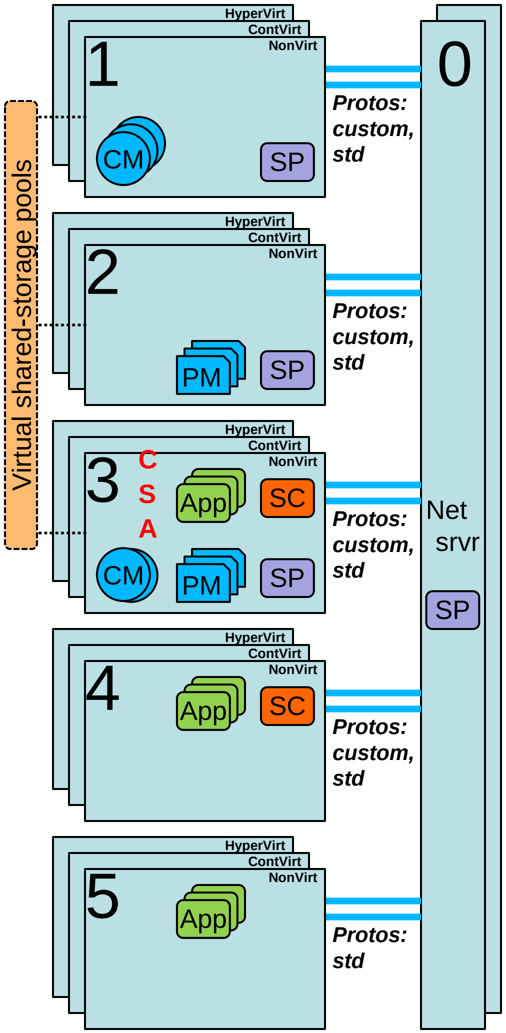
Combined Storage & App Nodes Cost-effectiveness: Pro

- In some cases, *smaller* # of nodes
  - Lower lifecycle costs
- Uniform software & hardware configs across cluster
  - Lower OPEX



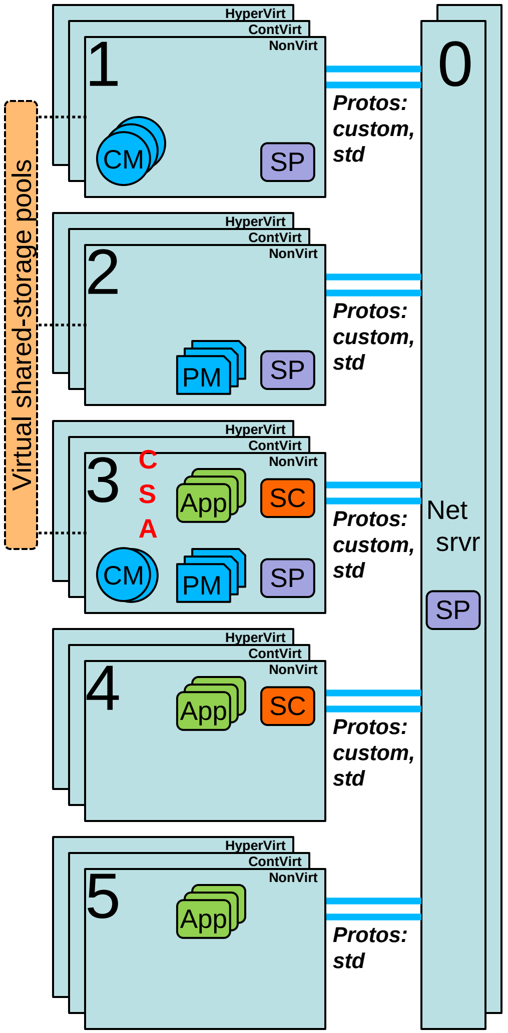
Combined Storage & App Nodes Cost-effectiveness: Con

- In some cases, *larger* # of nodes & media modules
  - Higher lifecycle costs
- May require more supporting infrastructure
  - App-only nodes contain no persistent data, & may be considered less-critical & given lower-cost supporting datacenter infrastructure. CSA may increase #nodes containing persistent data, & accordingly increase total usage of higher-cost supporting datacenter infrastructure (e.g., redundant/UPS power)



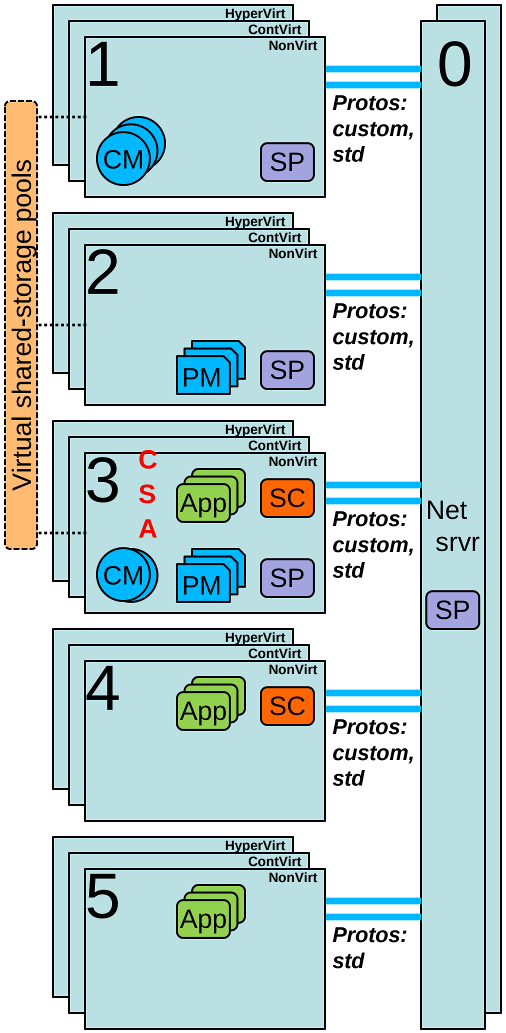
Combined Storage & App Nodes Cost-effectiveness: Con

- May pay storage-vendor "tax" for app-processing expansion
  - Storage vendor now also capturing compute spend
    - Caveat: storage vendor's custom Storage Consumer software & protocols may also add a lot of value
  - If packaged as HW+SW appliance, no HW vendor choice; higher margins
- May pay system-software "tax" for storage expansion
  - E.g., licensing for hypervisor-virt software stack



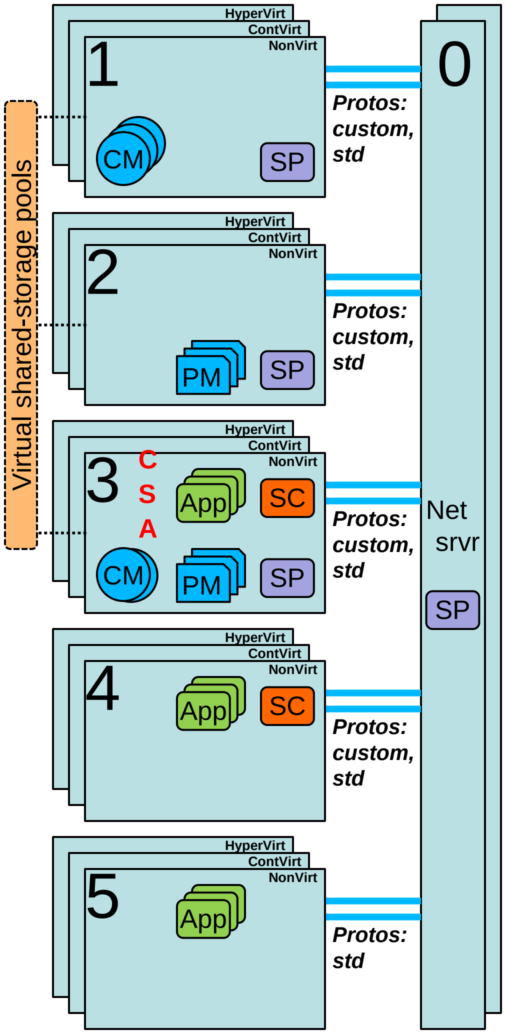
Combined Storage & App Nodes
Security & stability

- Pro: uniform system-software config across all nodes
  - Easier to set up & maintain consistent security configurations, updates to eliminate vulnerabilities
- Con: larger attack surface for shared-storage services
  - In some environments (e.g., service providers), app workloads may not all be fully trusted
  - Mixing shared storage services with untrusted app workloads within CSA nodes, may increase risks to security & stability of entire cluster
  - Compromise of single node may have less impact if only running apps, & not also shared services that may affect all nodes



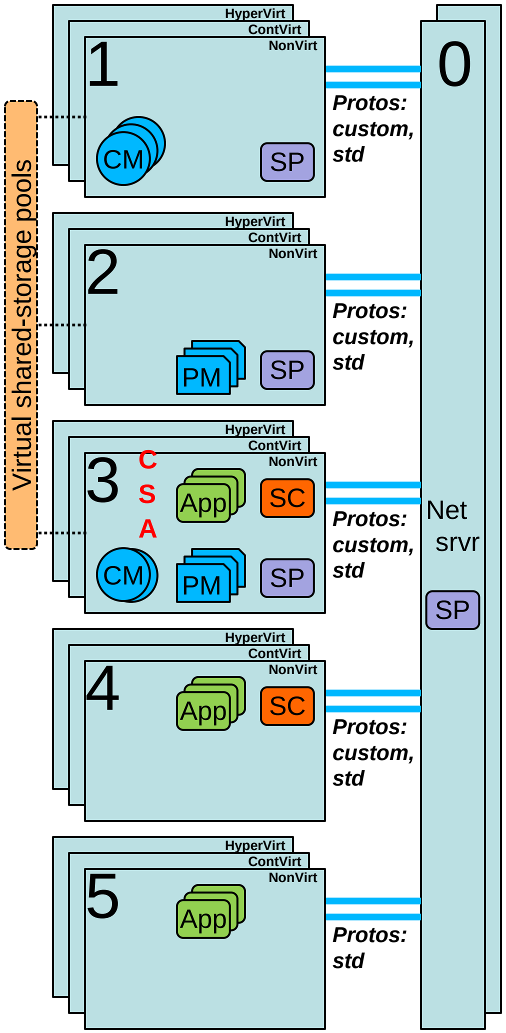
Combined Storage & App Nodes Closing thoughts

- CSA has significant pros & cons
  - vs. separate app & storage nodes
  - Pro/con balance specific to individual use cases, SoS implementations
- Incremental benefits of CSA
  - Often, NonSoS to SoS >> SoS to SoS+CSA
- Evaluating a SoS implementation
  - CSA support just one of many aspects to consider
- SoS implementation: node configs
  - CSA may be just one of many supported configs
  - CSA may be config option for any subset of nodes
    - Flexibility to optimize for wider range of use cases
    - Preferable to either requiring or prohibiting CSA across all nodes



Combined Storage & App Nodes Closing thoughts

- Field operational experience base: SoS, CSA
  - Much smaller than for older architectures
  - Much more use-case & best-practice guidance will emerge over time
- Current SoS, CSA implementations
  - Still at early stage of evolution; moving target
  - Many desirable capabilities not yet present
    - Expect much more in upcoming releases
  - Design space still lightly explored
    - Plenty of room for additional innovation
- CSA is attractive for many users & vendors
  - Technical & non-technical reasons
  - Expect continued strong growth in CSA support among SoS implementations



Attribution & Feedback

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

Authorship History

v1, Craig Dunwoody, 2015/02/17
v2, Craig Dunwoody, 2015/06/22
v3, Craig Dunwoody, 2015/07/27

Additional Contributors

Comments on v2, Thomas Rivera, 2015/07/20
Comments on v2, Joe White, 2015/07/22

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