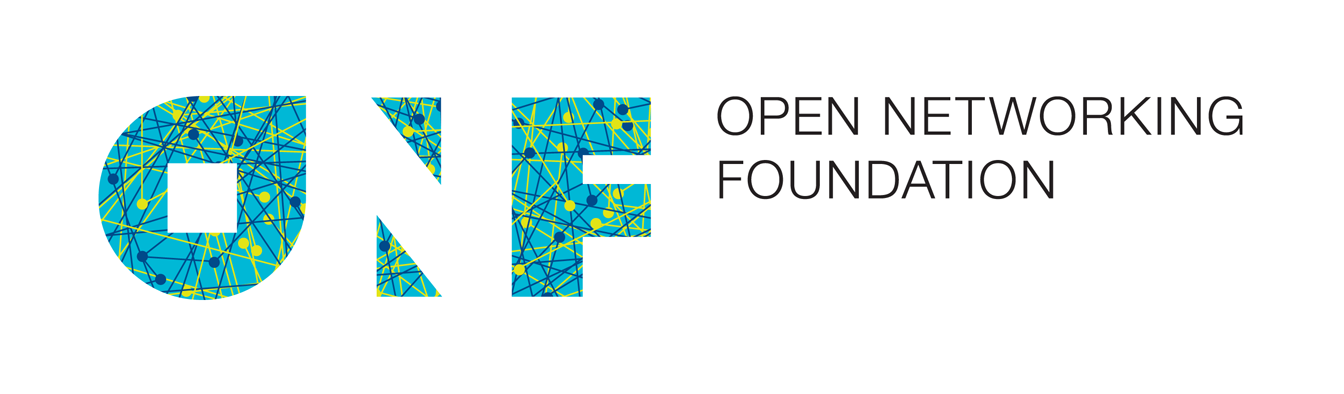
<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.A.8\_v1.3\_OnfCoreIm-Appendix-TimingAndSynchronizationExamples.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreModel.uml' element=’{0}’ importedBundles='gmf;papyrus' searchMetamodels='true'/>

<gendoc><drop/>

Change path substrings above from “{path for output files}\” to your local path for the output files and “{path for CoreModel}\” to your local path for the Core Model. <drop/>

DELETE: Prior to publishing this –gd.docx (including for review), change path substrings above from “C:\Users\ndavis\git\OnfInfoModelOutput\” to “{path for output files}\” and from “C:\Users\ndavis\git\ONFInfoModel\OnfModel\” to “{path for CoreModel}\” <drop/>



Core Information Model (CoreModel)

TR-512.A.14

Appendix – Storage Examples

Version 1.5

December 2019

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.5

**Disclaimer**

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Any marks and brands contained herein are the property of their respective owners.

Open Networking Foundation

2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303

[www.opennetworking.org](http://www.opennetworking.org)

©2017 Open Networking Foundation. All rights reserved.

Open Networking Foundation, the ONF symbol, and OpenFlow are registered trademarks of the Open Networking Foundation, in the United States and/or in other countries. All other brands, products, or service names are or may be trademarks or service marks of, and are used to identify, products or services of their respective owners.

Finalizing this document once generated… delete this text prior to publication:

* Replace “{{..}}” with square brackets (which trip up Gendoc)
* Select text in document from beginning of table of contents (first line) to end of document
  + Click menu item “Update Field” (on this large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
  + Repeat “update fields” 2 more times (on the same large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
* Remove reviewer comment

Note that the table of contents and figures need to be updated several times as the table length changes the page numbering and the cross references will need to be re-updated.

Table of Contents

[Disclaimer 2](#_Toc16002941)

[Open Networking Foundation 2](#_Toc16002942)

[Document History 4](#_Toc16002943)

[1 Introduction 5](#_Toc16002944)

[1.1 References 5](#_Toc16002945)

[1.2 Definitions 5](#_Toc16002946)

[1.3 Conventions 5](#_Toc16002947)

[1.4 Viewing UML diagrams 5](#_Toc16002948)

[1.5 Understanding the figures 5](#_Toc16002949)

[1.6 Appendix Overview 5](#_Toc16002950)

[2 Introduction to this Appendix document 6](#_Toc16002951)

[3 Storage Examples 6](#_Toc16002952)

[3.1 Single Disk 6](#_Toc16002953)

[3.2 Partitioned Single Disk 7](#_Toc16002954)

[3.3 RAID 1 (Mirror) 9](#_Toc16002955)

[3.4 Raid 0 (Striping) 11](#_Toc16002956)

[3.5 Partitioned RAID 1 Mirror 12](#_Toc16002957)

[3.6 Hosted Vmm {type-2} virtual Disk 13](#_Toc16002958)

[3.7 Linux Logical Volume Manager 15](#_Toc16002959)

[3.8 Network Storage – NAS 18](#_Toc16002960)

[3.9 Lustre and Ceph 21](#_Toc16002961)

[4 Fragment: Insert class <drop/> 23](#_Toc16002962)

[5 Fragment: Insert standard diagram <drop/> 24](#_Toc16002963)

[6 Fragment: Insert small diagram <drop/> 25](#_Toc16002964)

[7 Fragment: Insert attribute row brief not Obsolete<drop/> 25](#_Toc16002965)

[8 Fragment: Insert attribute row brief <drop/> 26](#_Toc16002966)

[9 Fragment: Start attribute table brief <drop/> 26](#_Toc16002967)

[10 Fragment: Insert Attribute table brief <drop/> 26](#_Toc16002968)

[11 Fragment: Insert Ten Specified Attribute table brief <drop/> 27](#_Toc16002969)

[12 Fragment: Insert DataType <drop/> 28](#_Toc16002970)

[13 Fragment: Start Data Type attribute table brief <drop/> 29](#_Toc16002971)

[14 Fragment: Insert Data Type Attribute table brief <drop/> 29](#_Toc16002972)

[15 Fragment: Insert enums <drop/> 29](#_Toc16002973)

[15.1.1.1 [dt.name/] 29](#_Toc16002974)

List of Figures

[Figure 1 - Partitioned Disk 8](#_Toc16087051)

[Figure 2 - Partitioned Disk (Renumbered) 9](#_Toc16087052)

[Figure 3 - RAID support 10](#_Toc16087053)

[Figure 4 - RAID 1 (Mirror) 11](#_Toc16087054)

[Figure 5 - Raid 0 Striping 12](#_Toc16087055)

[Figure 6 - Extent mapping 12](#_Toc16087056)

[Figure 7 - 3.4 Partitioned RAID 1 Mirror 13](#_Toc16087057)

[Figure 8 - Storage Model 14](#_Toc16087058)

[Figure 9 - Hosted Vmm {type-2} virtual Disk 15](#_Toc16087059)

[Figure 10 - Linux Logical Volume Manager 16](#_Toc16087060)

[Figure 11 - Linux Logical Volume Manager mapping to model 17](#_Toc16087061)

[Figure 12 - Linux Logical Volume Manager instance diagram 18](#_Toc16087062)

[Figure 13 – NAS 19](#_Toc16087063)

[Figure 14 - NAS Detail 20](#_Toc16087064)

[Figure 15 - CFS/NFS shares 21](#_Toc16087065)

[Figure 16 - Lustre Architecture http://lustre.org/ 22](#_Toc16087066)

[Figure 17 - Lustre instance Diagram 23](#_Toc16087067)

[Figure 5-1 [diagramTitle/] 25](#_Toc16087068)

[Figure 6-1 [diagramTitle/] 26](#_Toc16087069)

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 1.0 |  | Initial Version |
|  |  |  |

# Introduction

This document is an appendix of the addendum to the TR-512\_v1.5 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_v1.3_OnfCoreIm-Overview.pdf).

## References

For a full list of references see [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_v1.3_OnfCoreIm-Overview.pdf).

## Definitions

For a full list of definition see [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_v1.3_OnfCoreIm-Overview.pdf).

## Conventions

See [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_v1.3_OnfCoreIm-Overview.pdf) for an explanation of:

* UML conventions
* Lifecycle Stereotypes
* Diagram symbol set

## Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

## Understanding the figures

Figures showing fragments of the model using standard UML symbols and also figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_v1.3_OnfCoreIm-Overview.pdf) for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

## Appendix Overview

This document is part of the Appendix to TR-512. An overview of the Appendix is provided in [TR-512.A.](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.A.1_v1.3_OnfCoreIm-AppendixOverview.pdf)15.

# Introduction to this Appendix document

This document provides a description of Storage and provides examples of the use of the CIM abstractions to model these functions.

The examples in this document extend the simple example given in [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.11_v1.3_OnfCoreIm-ProcessingConstruct.pdf)5.

# Storage Examples

## Single Disk

Refer [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.11_v1.3_OnfCoreIm-ProcessingConstruct.pdf)5.

## Partitioned Single Disk

This is a simple example of storage partitioning (one in – many out).

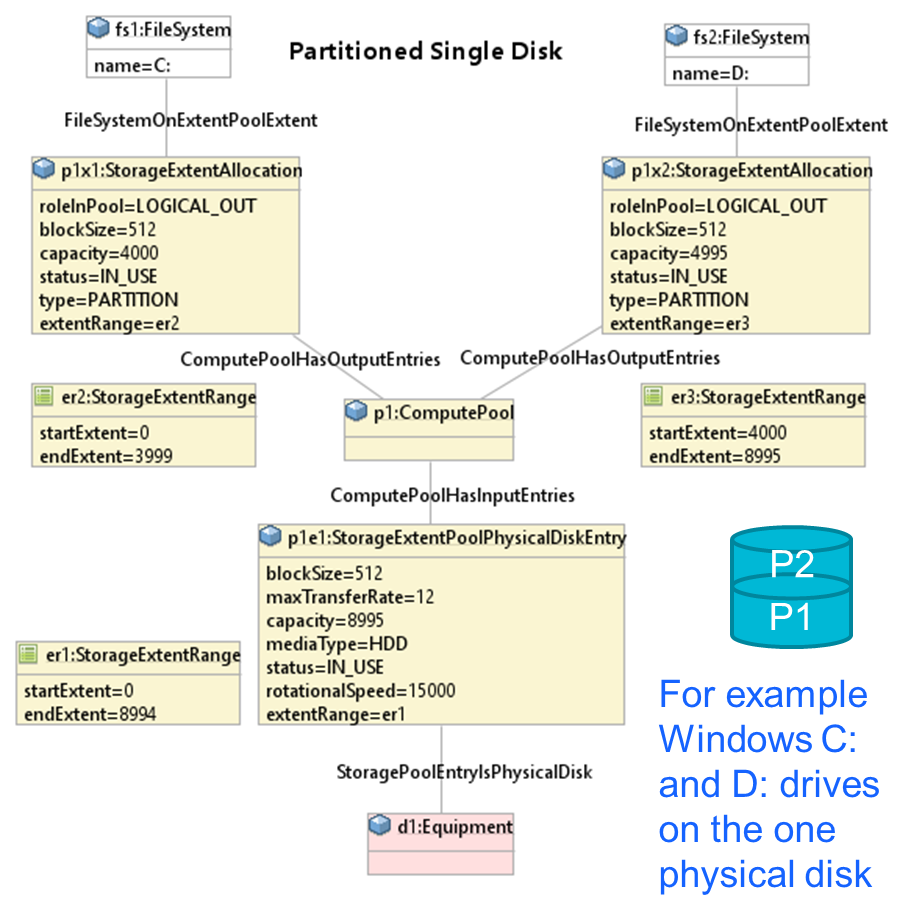


Figure 1 - Partitioned Disk

Note that in the example above the extent numbering is preserved from the input to the output.

If all partitions are presented with zero as the start number then we would have the following instead.

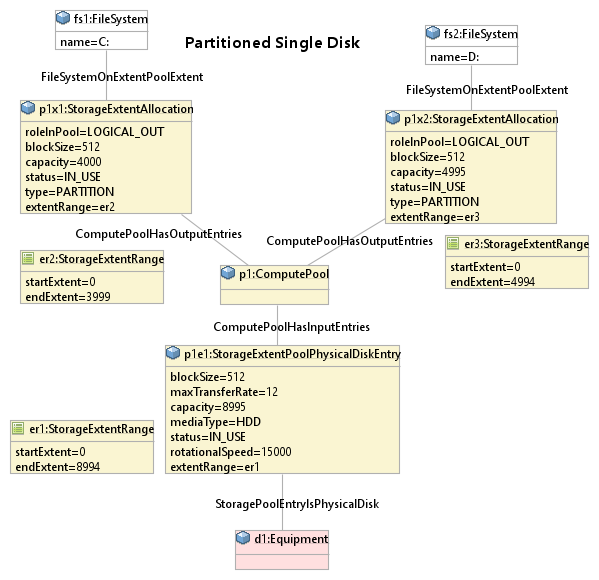


Figure 2 - Partitioned Disk (Renumbered)

## RAID 1 (Mirror)

This is the first example of extent aggregation (many in – one out).

Because of the way that mirroring works, these should be no need for extent renumbering.

Also the RAID unit has a separate control interface and this can be represented if needed too.

For RAID technology support, the core model will be extended using the Decorator pattern.

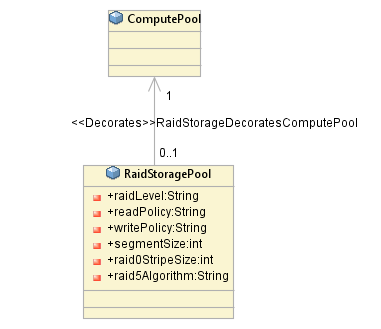


Figure 3 - RAID support

A good RAID reference is <http://www.accs.com/p_and_p/RAID/index.html>

Note that the raid0StripeSize and segmentSize is the size on each disk, not the total across all disks.

A stripe is also the smallest chunk of addressable data within a RAID array.

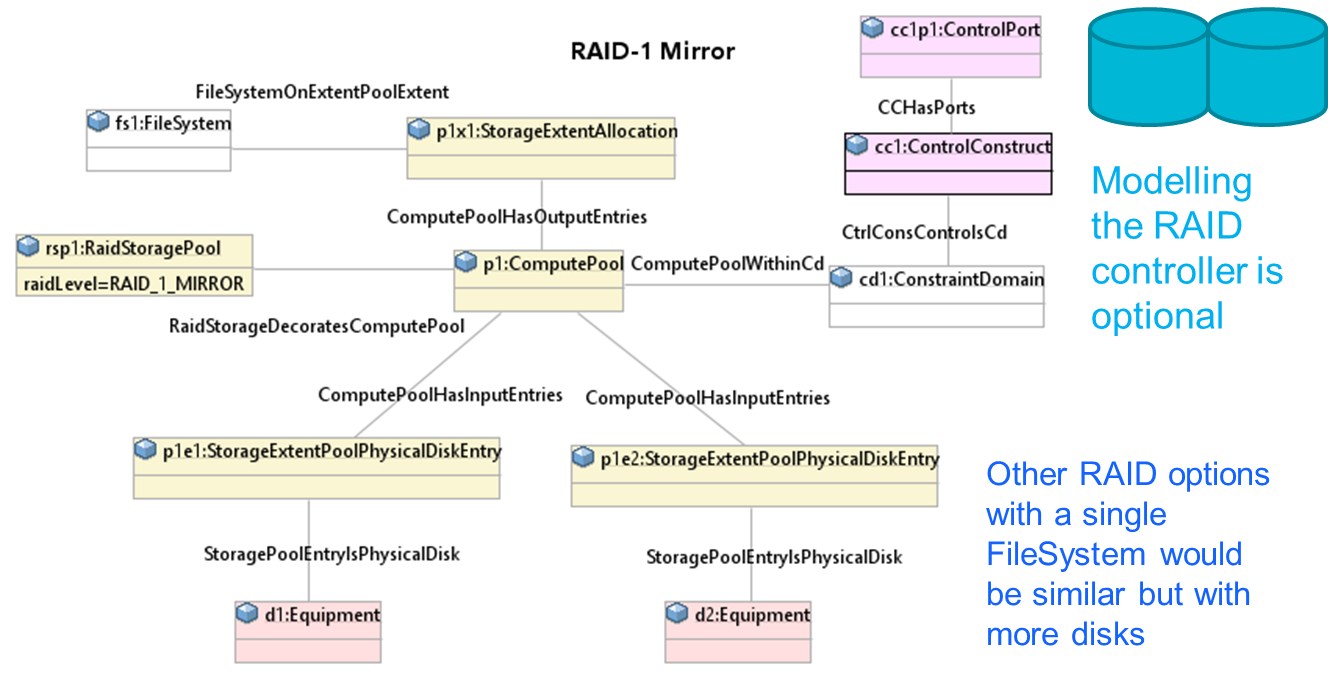


Figure 4 - RAID 1 (Mirror)

Note that when we are concatenating drives together the order may be significant.

This is supported in the model because association end ComputePoolHasInputEntries. computePoolInputEntry is ordered.

Note that RAID can be implemented in hardware or software, for instance, see <https://wiki.archlinux.org/index.php/Software_RAID_and_LVM>

## RAID 0 (Striping)

RAID 0 is very similar to RAID 1.

The difference is that this time we need to also record the stripe size (as shown in fig 3).

Assume the disk block size is 4 kiB and we choose a stripe size of 32 kiB, This means that each stripe will consist of 8 blocks.

Because association end ComputePoolHasInputEntries. computePoolInputEntry is ordered we know that the stripe will be on Disk1 first and then Disk2

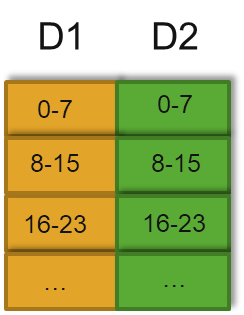


Figure 5 - Raid 0 Striping

So internally in the pool we can consider the inputs to be concatenated as shown by the bottom strip.

The pool can then allocate output blocks as shown by the two yellow upper blocks, each with their own numbering. The internal numbering allows the input to output mapping to be implied without having to directly tie the inputs and outputs together.

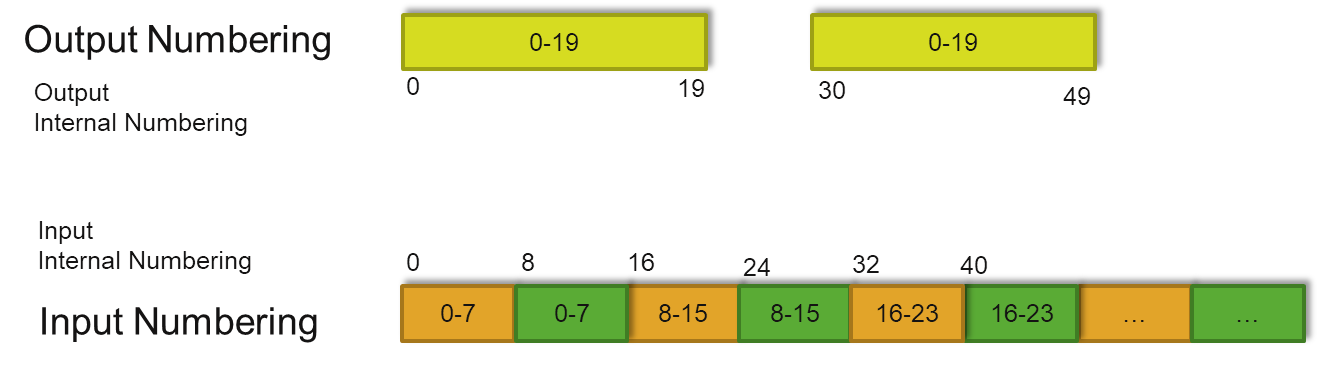


Figure 6 - Extent mapping

## Partitioned RAID 1 Mirror

This example shows both aggregation and partitioning (many in – many out).

Comparing this with figure 4 shows that the inputs are the same, but now the pool has two outputs (like in figure 2). The decoupling of input and output in the pool gives a clean and simple representation.

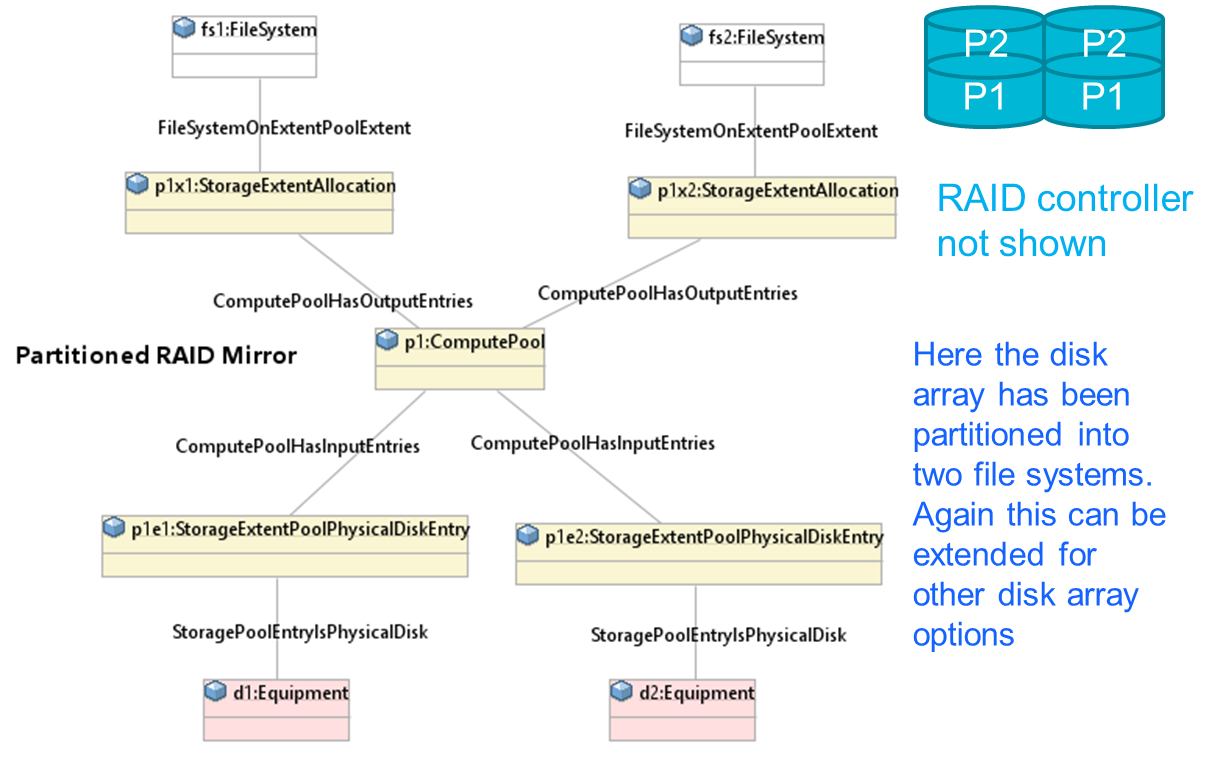


Figure 7 - 3.4 Partitioned RAID 1 Mirror

## Hosted VMM {type-2} virtual Disk

The figure below shows a simplified class diagram of the Storage model.

For a Hosted VMM, the hypervisor typically uses one or more files in the host filesystem to present to a VM as an emulated disk.

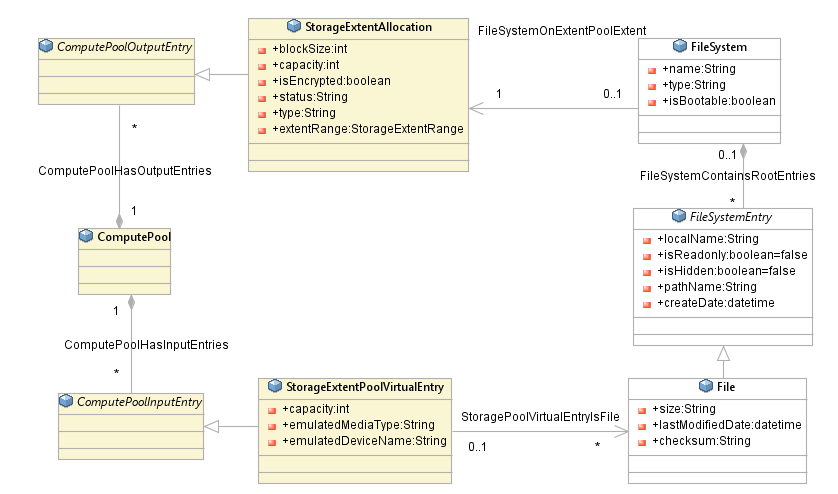


Figure 8 - Storage Model

What is not clear from the class diagram is that there are actually 2 distinct FileSystems : the one on the Host machine and the one within the VM. This difference is clarified in the instance diagram below. For simplicity the Host is shown as having a locally attached disk drive, but the host could be using NAS or SAN storage instead.

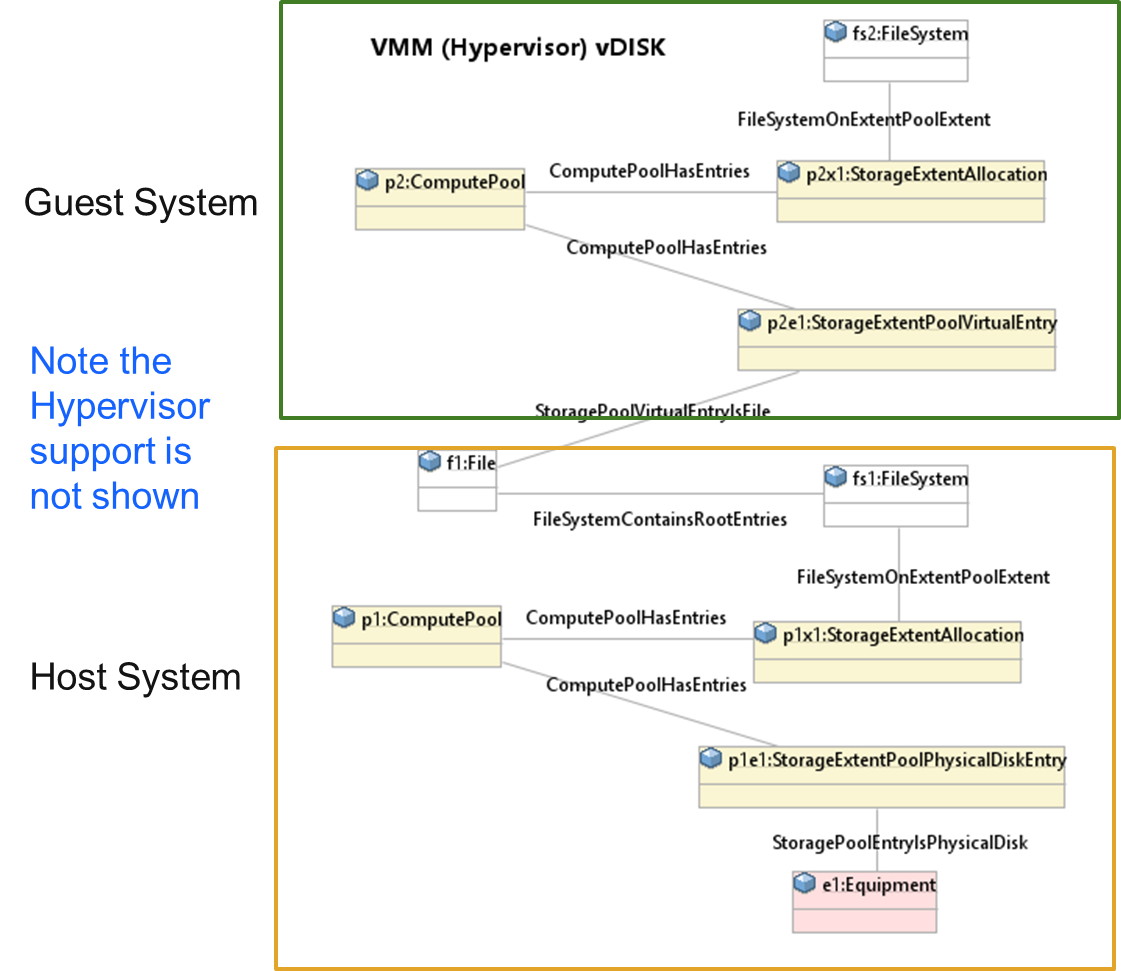


Figure 9 - Hosted Vmm {type-2} virtual Disk

## Linux Logical Volume Manager

A volume manager supports much more complex storage allocation and management.

The figure below shows a simplistic representation of how it works.

Raw disks or disk partitions (or hardware RAID volumes) can be put under the control of the volume manager. These are all aggregated into one large pool of storage that can then be allocated as required.

Again note that the ordering of inputs to the volume group is critical to enable mapping from the input extent numbers to the output extent numbers.

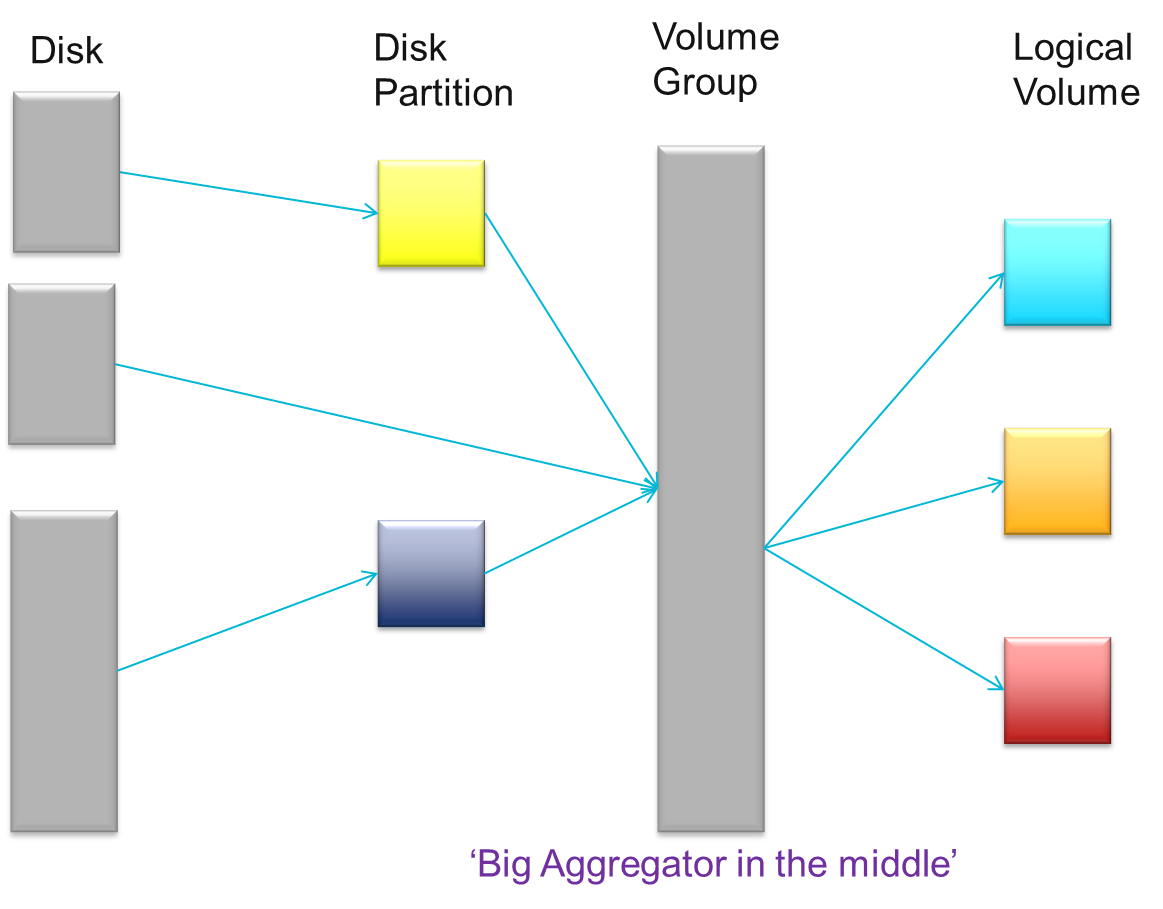


Figure 10 - Linux Logical Volume Manager

Also see the diagram at <https://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux)>

For a diagram showing the various elements of the LVM and how they are related.

The diagram below shows how the ONF storage model is mapped to the volume manager concepts.

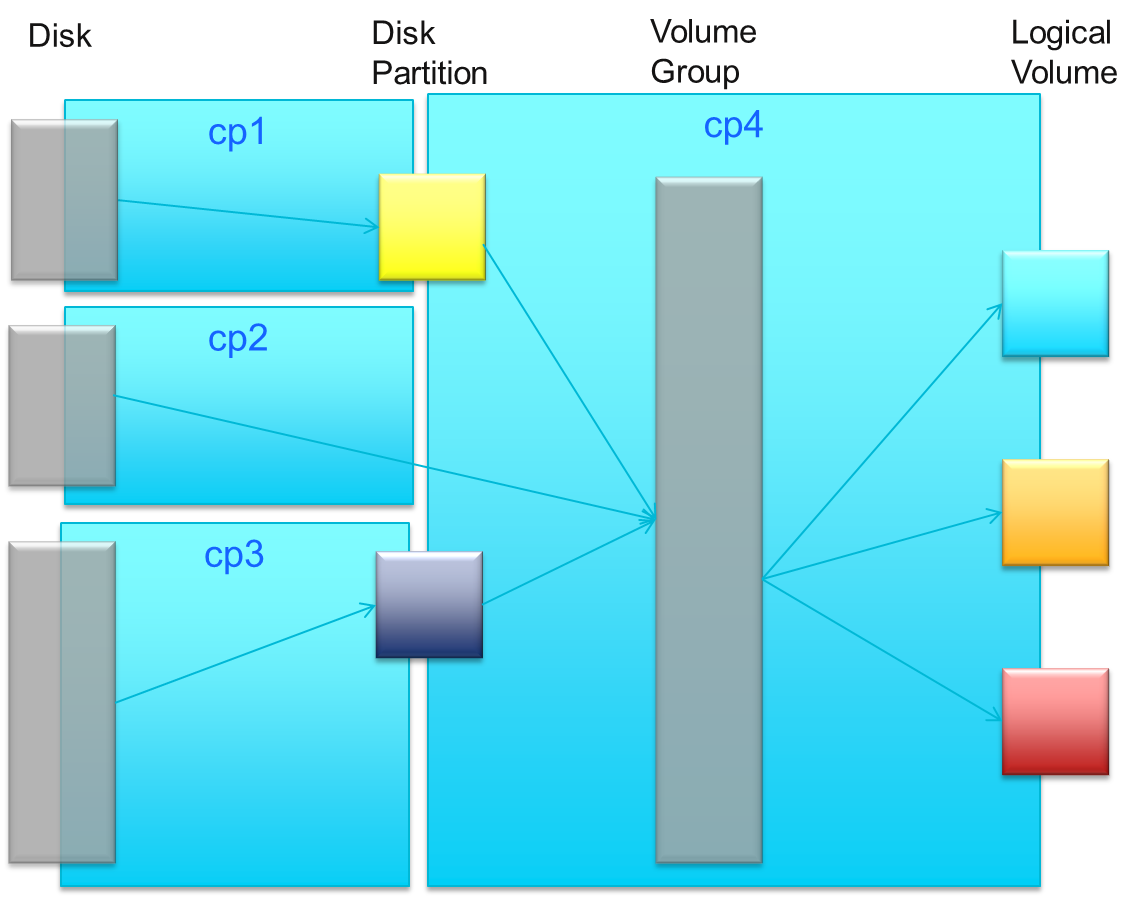


Figure 11 - Linux Logical Volume Manager mapping to model

This page has some comments on using LVM and RAID together <https://serverfault.com/questions/217666/what-is-better-lvm-on-raid-or-raid-on-lvm>

LVM can also create RAID logical volumes, see <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/logical_volume_manager_administration/raid_volumes>

The instance diagram below shows the basic structure of how the concepts are interlinked.

It also shows how ComputePools can be combined via layering.

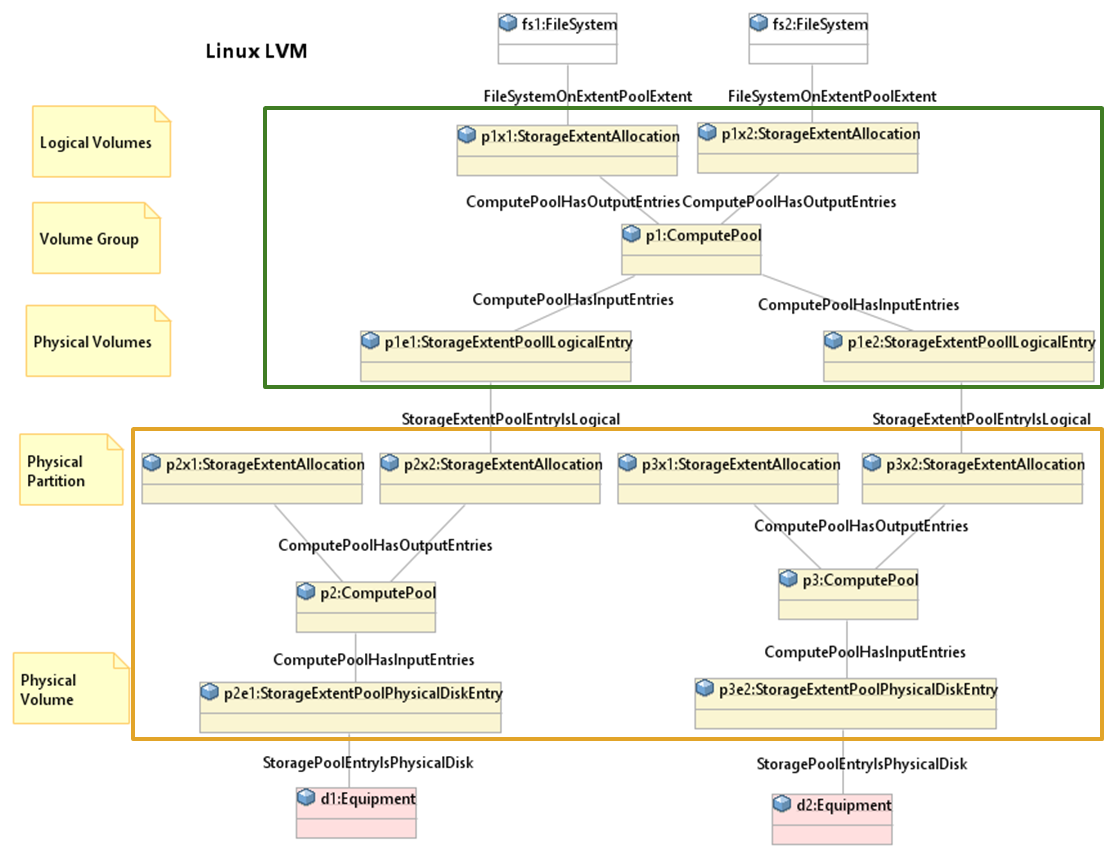


Figure 12 - Linux Logical Volume Manager instance diagram

See also :  
<https://raid.wiki.kernel.org/index.php/What_do_you_want_in_your_stack%3F>

<https://wiki.archlinux.org/index.php/LVM#LVM_Building_Blocks>

## Network Storage – NAS

The previous examples have been for directly attached (local) storage. Now we will look at a few options for ‘remote’ storage. The big benefit of ‘remote’ storage is that it allows centralization and more efficient management of storage.

As figure 1 in the main document shows, a NAS exposes a FileSystem via CFS/NFS protocols.

From a simple ‘black box point of view’ we can ignore the internals of the NAS, the internal drives and just focus on the presented FileSystems.

A NAS is more than just a disk drive with an ethernet port on the side, and below a ControlConstruct is also shown that is used to control the access and configuration of the NAS system.

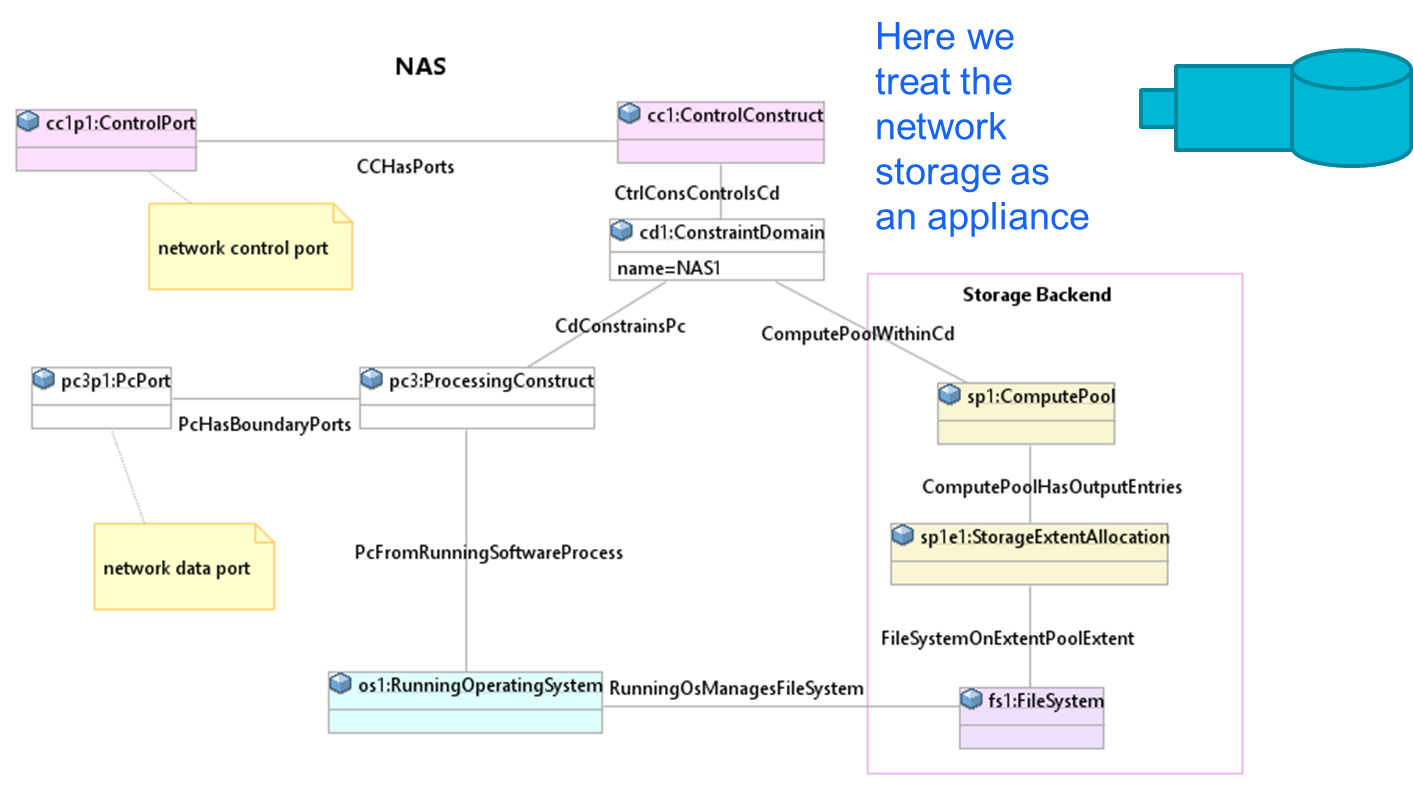


Figure 13 – NAS

Another option is to have a more detailed representation of the NAS.

The choice on the level of detail depends on what information is available and what information is needed to manage the particular situation.

Again, the ONF CIM core provides the primitives needed to represent this option.

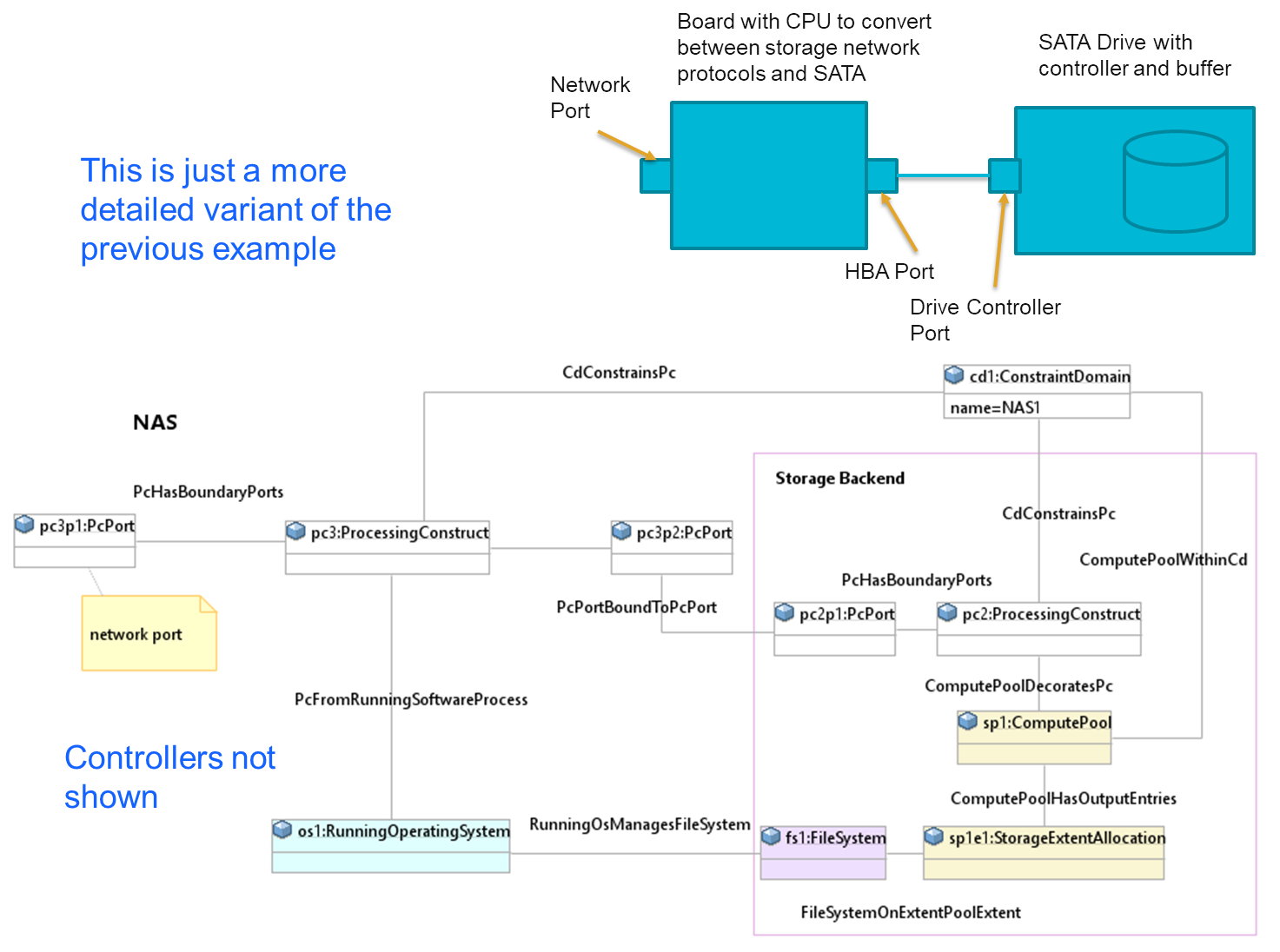


Figure 14 - NAS Detail

CIFS and NFS servers share via directories. Clients may import the shared directory as a ‘drive’ (Windows) or as a directory (Linux). To represent this, the model can be extended by adding classes to support each of the shares.

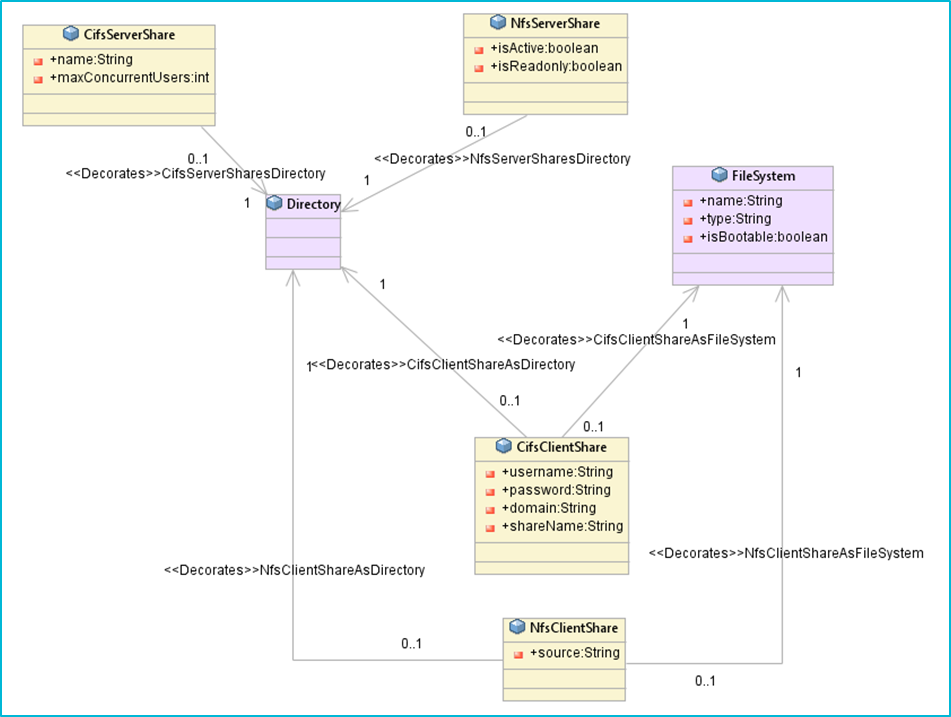


Figure 15 - CFS/NFS shares

## Lustre and Ceph

“Lustre is a type of parallel distributed file system, generally used for large-scale cluster computing.” [Wikipedia]

Systems like Lustre can be represented using the ONF CIM core primitives for each of the devices in the system.

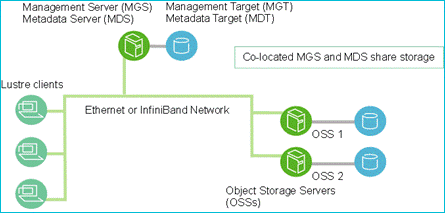


Figure 16 - Lustre Architecture <http://lustre.org/>

Using the ONF CIM core model, the CEPH architecture can be mapped as shown in the instance diagram below.

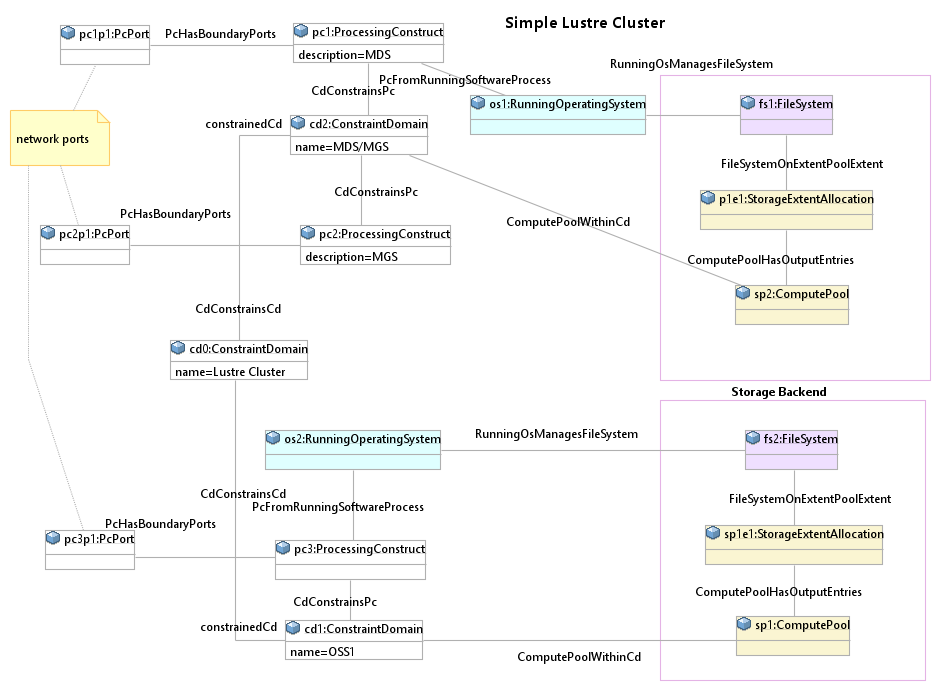


Figure 17 - Lustre instance Diagram

“In computing, Ceph (pronounced /ˈsɛf/) is a free-software storage platform, implements object storage on a single distributed computer cluster, and provides interfaces for object-, block- and file-level storage.” [Wikipedia]

In a manner similar to Lustre, Ceph ( <https://ceph.com> ) can also be represented using the ONF CIM constructs.

**End of Document**

</gendoc><drop/>

To take latest template: <drop/>

* delete text from “Template version…” to end of file <drop/>
* insert a line in “Normal” style<drop/>
* insert text (Insert 🡪 Object 🡪 Text from File… (alt njf)) from: <drop/>
  + TR-512.GT\_v1.3\_OnfCoreIm-CommonGendocTemplate-Fragments.docx <drop/>

Template version 0.0.10 17 September 2017 <drop/>

# Fragment: Insert class <drop/>

<fragment name=’insertClass’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
<arg name=’className’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (not cl.qualifiedName.contains(packageName))]<drop/>  
[else] <drop/>  
[if(cl.name.contains(className))]<drop/>

Qualified Name: [cl.qualifiedName/]

[for (co:Comment | cl.ownedComment)]<drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>  
[if (cl.isAbstract)]<drop/>

This class is abstract.

[/if]<drop/>

[if (cl.oclAsType(uml::Class).general ->notEmpty())]<drop/>

Inherits properties from:

[for (gen:Class | cl.oclAsType(uml::Class).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

[for (st:Stereotype | cl.getAppliedStereotypes())]<drop/>  
[if(not st.name.contains(‘OpenModelClass’))]<drop/>

This class is [st.name/].

[else] <drop/>  
[/if]<drop/>  
[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Insert standard diagram <drop/>

<fragment name=’insertStandardDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

[for (d:Diagram|p.getPapyrusDiagrams())]<drop/>

[if d.name.contains(diagramName)]

<drop/>

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW = ‘false’></image>

CoreModel diagram: [d.name/]

Figure 5-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert small diagram <drop/>

<fragment name=’insertSmallDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

[for (d:Diagram|p.getPapyrusDiagrams())]<drop/>

[if d.name.contains(diagramName)]

<drop/>

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW = ‘false’></image>

CoreModel diagram: [d.name/]

Figure 6-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief not Obsolete<drop/>

<fragment name=’insertAttributeRowBriefNotObsolete’ importedBundles=’commons;gmf;papyrus’><drop/>

Does not work unless we have Mature stereotype… <drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

[for (st:Stereotype | p.getAppliedStereotypes())]<drop/>

[if(not st.name.contains(‘OpenModelAttribute’))]

[if(not st.name.contains(‘Obsolete’))]

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief <drop/>

<fragment name=’insertAttributeRowBrief’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

</fragment><drop/>

# Fragment: Start attribute table brief <drop/>

<fragment name=’insertAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Attribute table brief <drop/>

<fragment name=’insertAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 1: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert Ten Specified Attribute table brief <drop/>

<fragment name=’insertTenSpecifiedAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

<arg name=’p1’ type=‘String’/><drop/>

<arg name=’p2’ type=‘String’/><drop/>  
<arg name=’p3’ type=‘String’/><drop/>  
<arg name=’p4’ type=‘String’/><drop/>  
<arg name=’p5’ type=‘String’/><drop/>  
<arg name=’p6’ type=‘String’/><drop/>  
<arg name=’p7’ type=‘String’/><drop/>  
<arg name=’p8’ type=‘String’/><drop/>  
<arg name=’p9’ type=‘String’/><drop/>  
<arg name=’p10’ type=‘String’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 1: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

[for (p:Property|cl.ownedAttribute)]<drop/>

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert DataType <drop/>

<fragment name=’insertDataType’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
<arg name=’dataTypeName’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (dt.qualifiedName.contains(packageName))]<drop/>  
[if(dt.name.contains(dataTypeName))]<drop/>

Qualified Name: [dt.qualifiedName/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>  
[if (dt.oclAsType(uml::DataType).general ->notEmpty())]<drop/>

Inherits properties from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]<drop/>

[for (gen:Class | dt.oclAsType(uml::DataType).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

[for (st:Stereotype | dt.getAppliedStereotypes())]<drop/>  
This class is [st.name/].

[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Start Data Type attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Data Type Attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertDataTypeAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
[if dt.ownedAttribute->notEmpty()]<drop/>

Table 1: Attributes for [dt.name/]

<table><drop/>

[dt.insertDataTypeAttributeTableHeader ()/]

[for (p:Property|dt.ownedAttribute)]<drop/>

[p.insertAttributeRowBrief ()/]

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert enums <drop/>

<fragment name=’insertEnums’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

#### [dt.name/]

Qualified Name: [dt.qualifiedName/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[cleanAndFormat(co.\_body.clean())/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[if dt.getAppliedStereotypes()->notEmpty()] <drop/>

[for (st:Stereotype | dt.getAppliedStereotypes())]<drop/>

* [st.name/]

[/for]<drop/>

[else] No stereotypes applied

[/if]<drop/>

[if (dt.oclAsType(uml::DataType).general ->notEmpty())]<drop/>

Inherits literals from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]

[/if]<drop/>

[if (dt.oclAsType(Enumeration).ownedLiteral->notEmpty())]<drop/>

Contains Enumeration Literals:

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]:
  + [for (co:Comment | e.ownedComment)]<drop/>
  + <dropEmpty>[cleanAndFormat(co.\_body.clean())/]
  + </dropEmpty>[/for]<drop/>
  + [if dt.getAppliedStereotypes()->notEmpty()] <drop/>
  + Applied stereotypes:
    - [for (st:Stereotype | e.getAppliedStereotypes())]<drop/>
    - [st.name/]
    - [/for]<drop/>
  + [/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>