



Red Storm Systems Management: Topics on Extending Current Capabilities

James H. Laros III

jhlaros@sandia.gov

Sandia National Laboratories

Presented by

Robert A. Ballance

raballa@sandia.gov



System Description Language (SDL)

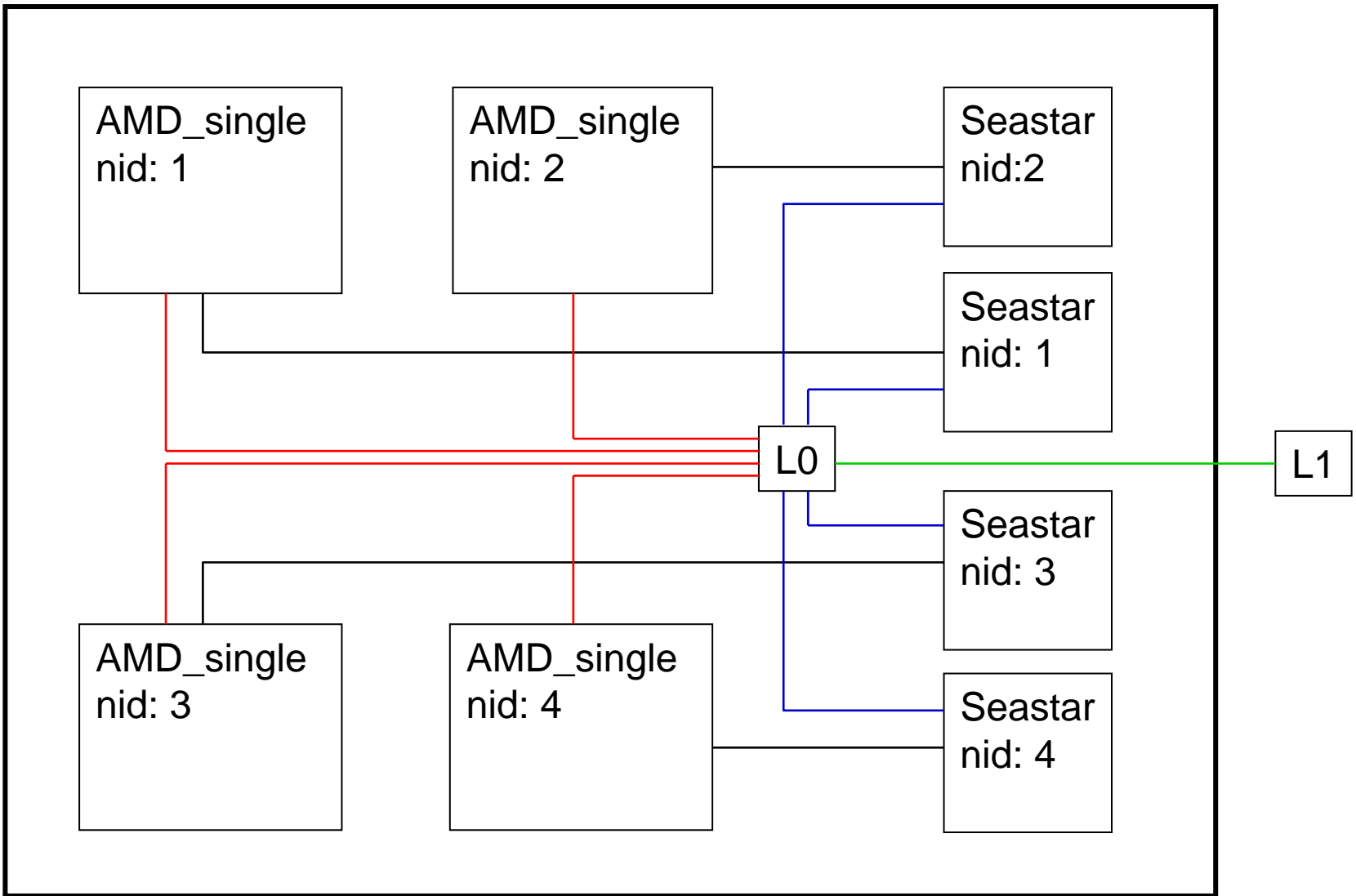
- **Describe a system in software**
 - **Information about components in system**
 - Anything relevant can be stored
 - **Relationships between components in system**
 - Any simple or complex relationship between two or many components
 - **Functional capabilities of components**
 - Leverage anything a component can do
- **Store resulting description in database**
 - **Populate database from existing devices**
 - **Compare database to state of devices**
 - **Make devices reflect the current database**
- **Provides the foundation for many capabilities**
- **Leverage Object Oriented concepts heavily**



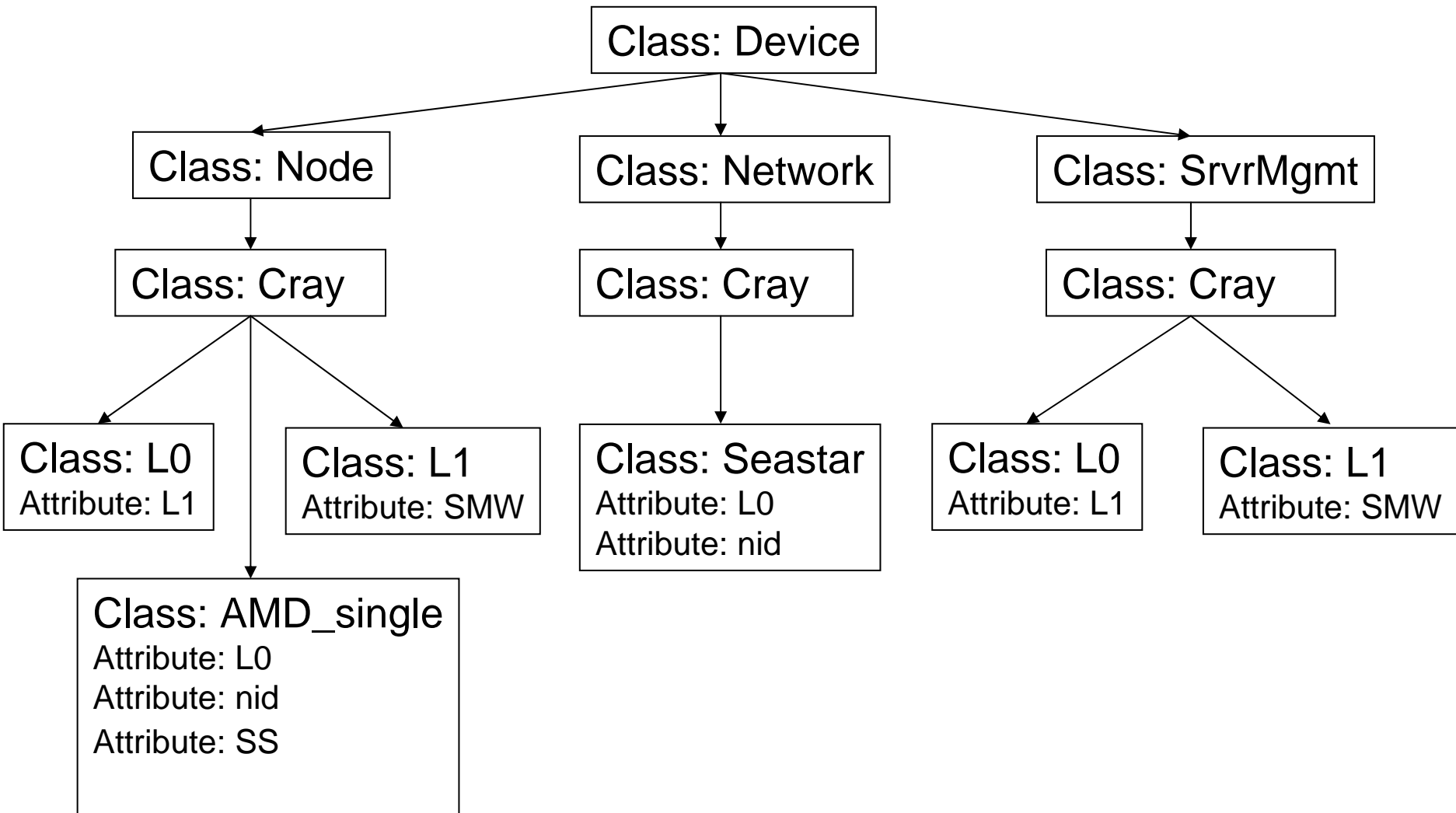
Can we describe Red Storm this way?

- **Represent system components by classes**
 - **Class attributes hold information about instantiated objects**
 - **Class attributes describe relationships between instantiated objects**
- **Instantiate objects for each component in system**
 - **Can also express relationships by grouping objects in collections**
- **Persistently store objects in database**

Red Storm Module (conceptual view)



Red Storm SDL Class Hierarchy (topological view)





Red Storm Component Class Names (from class hierarchy)

Device::Node::Cray::AMD_single

Device::Node::Cray::L0

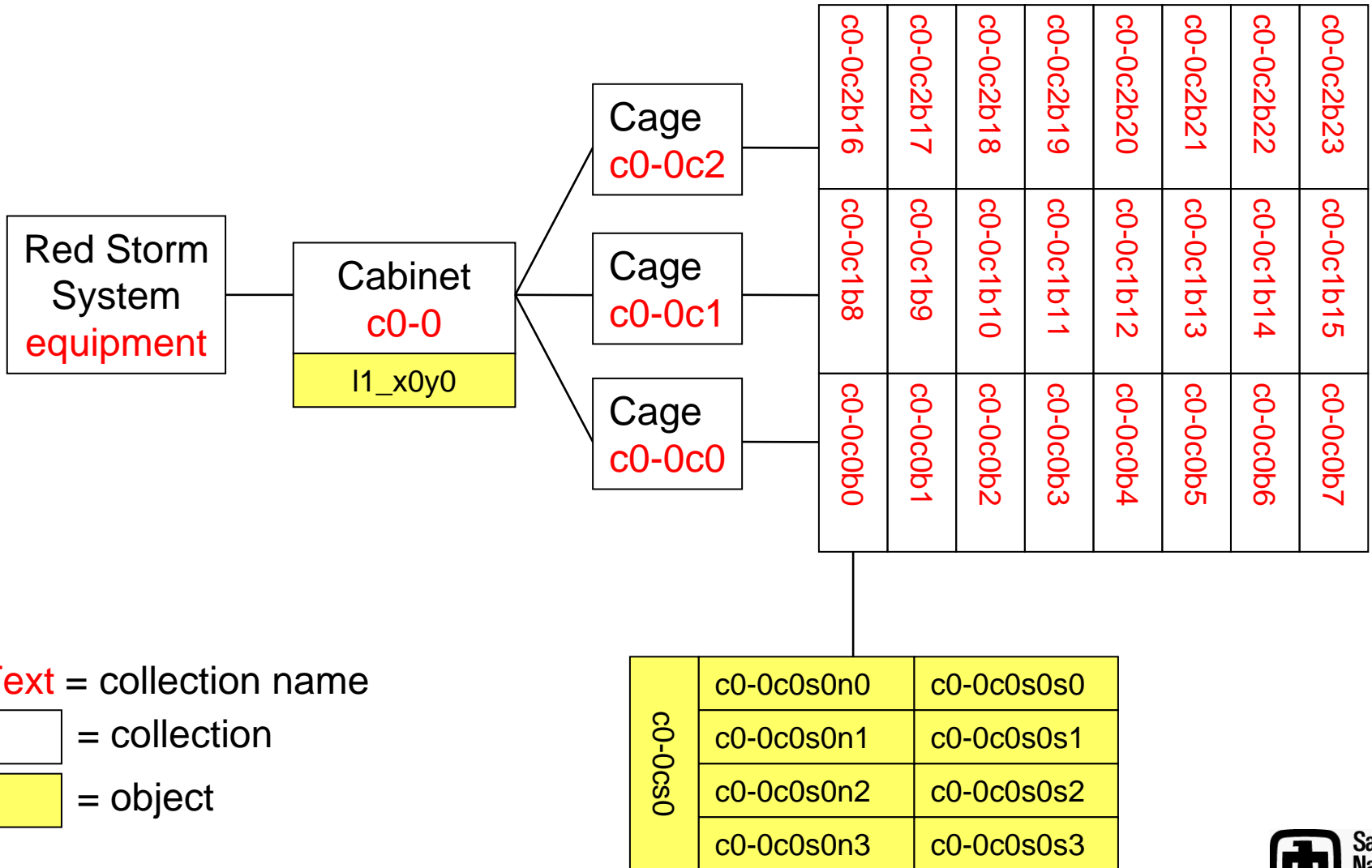
Device::Node::Cray::L1

Device::Network::Cray::Seastar

Device::SrvrMgmt::Cray::L0

Device::SrvrMgmt::Cray::L1

Red Storm Component Collections and Objects





Viewing the stored Collections and Objects

```
# collection_mgr equipment
c0-0
c0-1
c0-2
.
.
```

```
# collection_mgr c0-0
c0-0c0
c0-0c1
c0-0c2
l1_x0y0
l1_x0y0-mgt
```


```
# device_mgr l1_x0y0
name => l1_x0y0
interface => 0
    nic => 0
    name => eth0
    address => 10.1.100.100
    net_mask => 255.255.0.0
    boot_if => 1
    hostname => l1_x0y0
    is_primary => 1

role => RAS
vmname => Management
x_pos => 0
y_pos => 0
isa => Device::Node::Cray::L1

# device_mgr l1_x0y0-mgt
name => l1_x0y0-mgt
interface => 0
    nic => 0
    name => eth0
    address => 10.1.100.100
    net_mask => 255.255.0.0
    boot_if => 1
    hostname => l1_x0y0-mgt
    is_primary => 1

role => RAS
vmname => Management
x_pos => 0
y_pos => 0
isa => Device::SrvrMgmt::Cray::L1
```





```
# collection_mgr c0-0c0
c0-0c0b0
c0-0c0b1
c0-0c0b2
c0-0c0b3
c0-0c0b4
c0-0c0b5
c0-0c0b6
c0-0c0b7
```

```
# collection_mgr c0-0c0b0
c0-0c0s0n0
c0-0c0s0n1
c0-0c0s0n2
c0-0c0s0n3
c0-0c0s0s0
c0-0c0s0s1
c0-0c0s0s2
c0-0c0s0s3
c0-0c0s0
c0-0c0s0-mgt
```

Viewing the stored Collections and Objects

(continued)

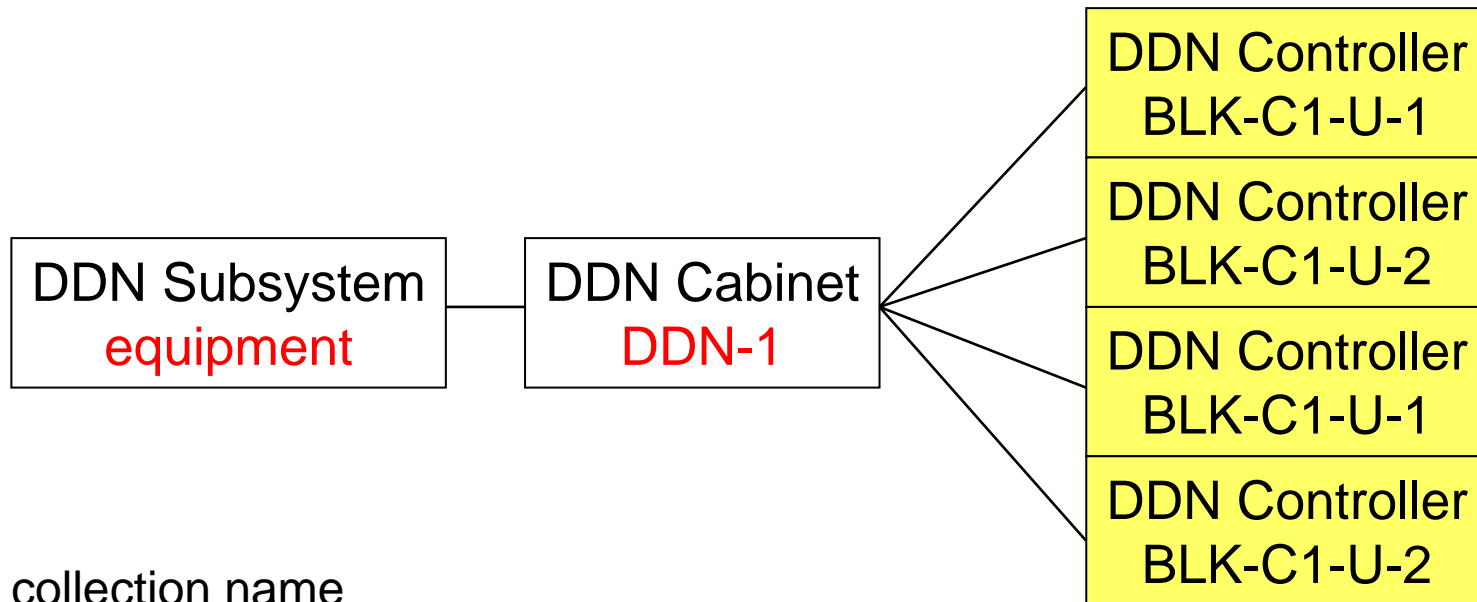
```
# device_mgr c0-0c0s0n0
name => c0-0c0s0n0
l0 => c0-0c0s0
power => c0-0c0s0-mgt
power_port => 0
cage => 0
slot => 0
x_pos => 0
y_pos => 0
nid => 0
role => compute
vmname => catamount
leader => c0-0c0s0
SS => c0-0c0s0s0
isa => Device::Node::Cray::AMD_single
```



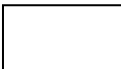
Can we use this concept to interact with components?


- Describe system the same way
- Add methods to implement component functionality
 - Methods represent the language that the components communicate in
- Leverage the methods to accomplish useful work
 - Interact with components
 - Discover information about components
 - Update database with this info
 - Automatic population of database
 - Configure components
 - Configure components based on settings stored in database

DDN Component Collections and Objects



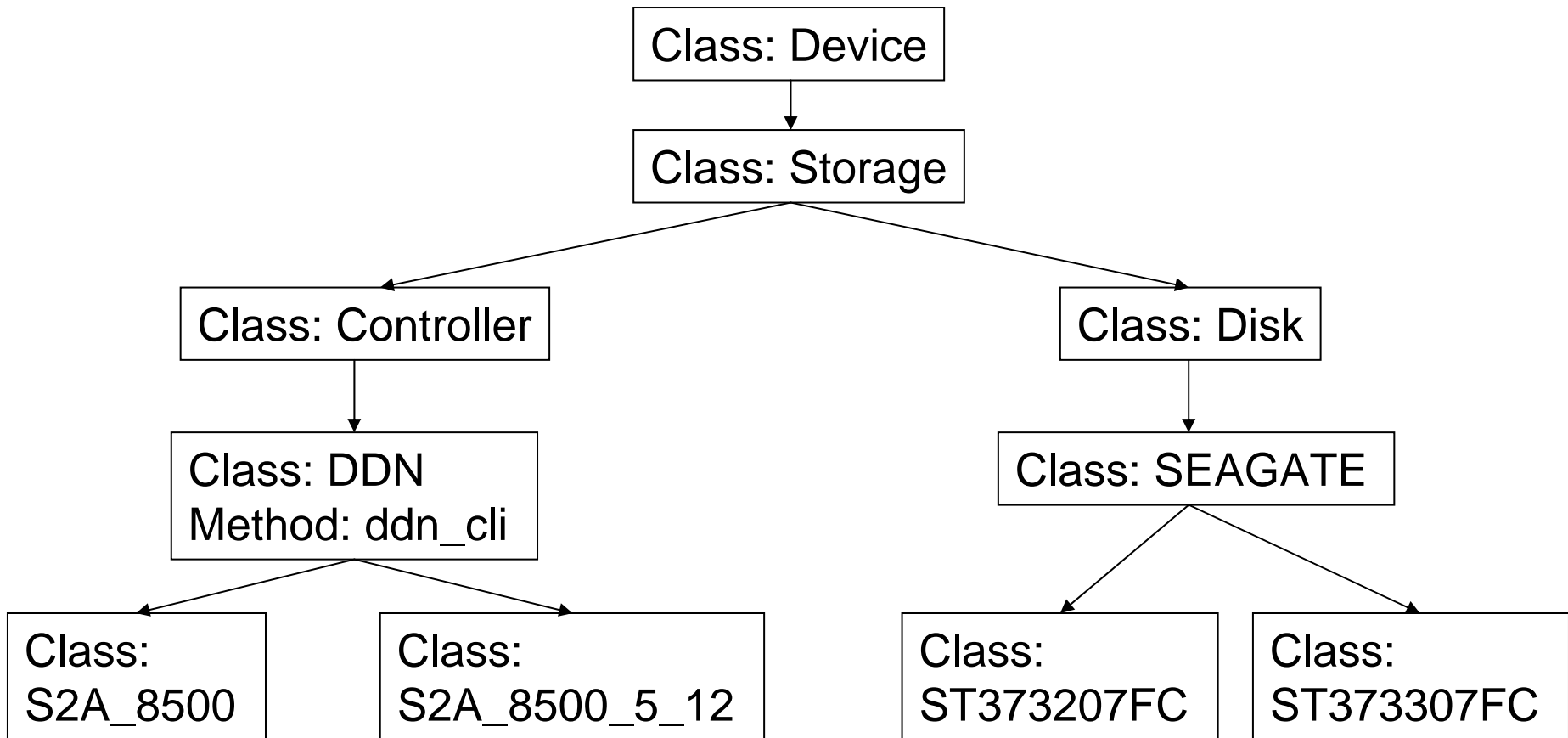
Text = collection name

 = collection

 = object



DDN Component Class Hierarchy





Command formats and specific examples

ddn <flag> <component or collection> - *format for 1st level DDN CLI command*

ddn -faults BLK-C1-U-1 – *example: execute the DDN CLI faults command on controller BLK-C1-U-1*

ddn -faults DDN-1 – *example: execute the DDN CLI faults command on all controllers in the DDN-1 collection*

ddn_<1st level cmd> --<2nd level parameter> <component or collection>
- *format for 2nd level DDN CLI command*

ddn_disk -list BLK-C1-U-1 – *example: executes “disk list” on controller BLK-C1-U-1*

ddn -check_faults BLK-C1-U-1 – *example: executes “custom” check_faults command*

ddn -check_faults DDN-1 | xargs ddn -faults – *example: combining commands*

ddn -populate_disks <controller or component> - *example: discovering attached disks on controller or entire system*



Command execution times

```
# time ddn_disk -status BLK-C1-U-1  
real 0m1.315s  
user 0m0.190s  
sys 0m0.000s
```

```
# time ddn -check_connection BLK-C1-U-1  
real 0m1.290s  
user 0m0.190s  
sys 0m0.010s  
  
# time ddn -check_connection all_controllers  
real 0m2.420s  
user 0m0.910s  
sys 0m0.430s
```

```
# time ddn -stats BLK-C1-U-1  
real 0m1.291s  
user 0m0.190s  
sys 0m0.010s  
  
# time ddn -stats all_controllers  
real 0m5.447s  
user 0m1.130s  
sys 0m0.590s
```



Conclusions

- **This foundation can be leveraged for many purposes**
 - **Reliability Availability and Serviceability (RAS) systems**
 - We are currently working on a RAS software architecture using the SDL as a foundation
 - **Schedulers**
 - **Run-time systems**
 - **Anything that needs information about a system to perform a task**
- **Very easy to extend for use on other systems**
- **Capable of storing information for very large systems**
 - **We are storing approx xxx components in our redstorm database**



Questions?

- **Say thank you to Bob for me... Jim 😊**