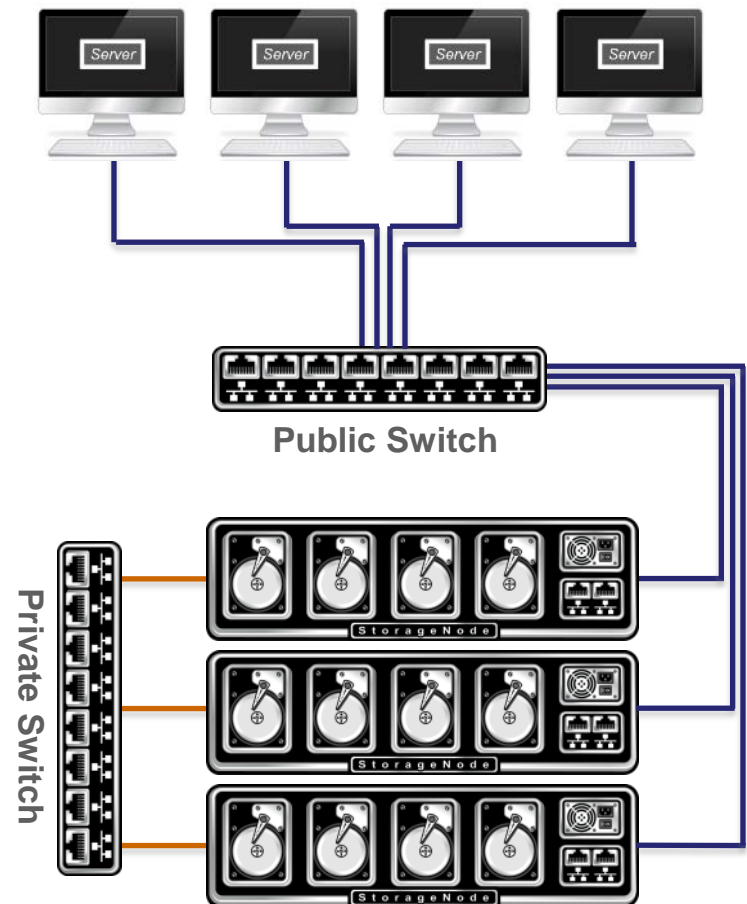


An Extensible Open-Source iSCSI & SCSI Test Tool

Richard Sharpe
Scale Computing

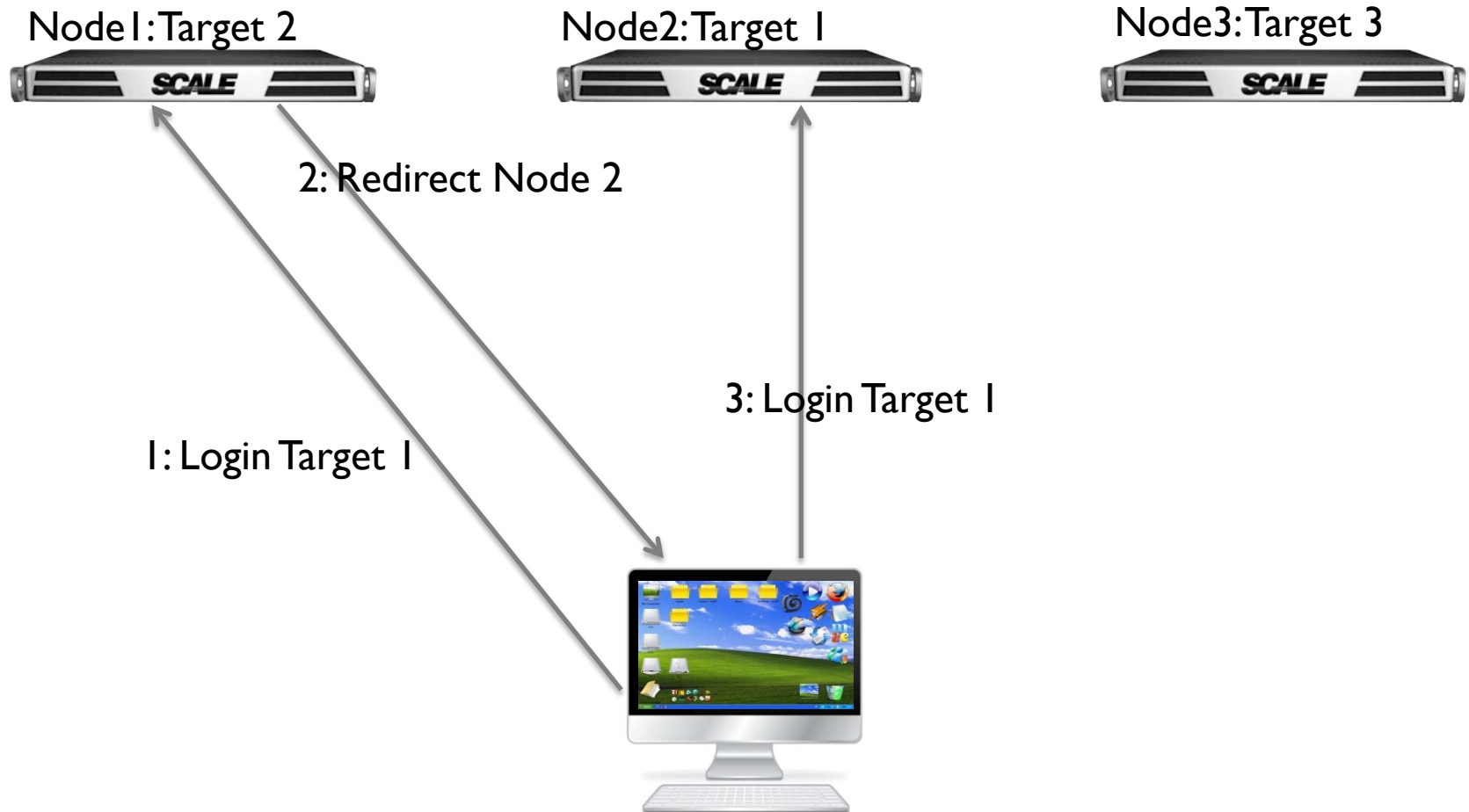
- ❑ Rationale
- ❑ Scale Automated Testing Environment
- ❑ Alternatives for iSCSI/SCSI Testing
- ❑ Approach Chosen
- ❑ SCSI Classes
- ❑ iSCSI Classes
- ❑ A sample test
- ❑ Source Code Availability

- Scale Computing, inc, develops scale-out unified storage appliances
 - NFS, CIFS and iSCSI

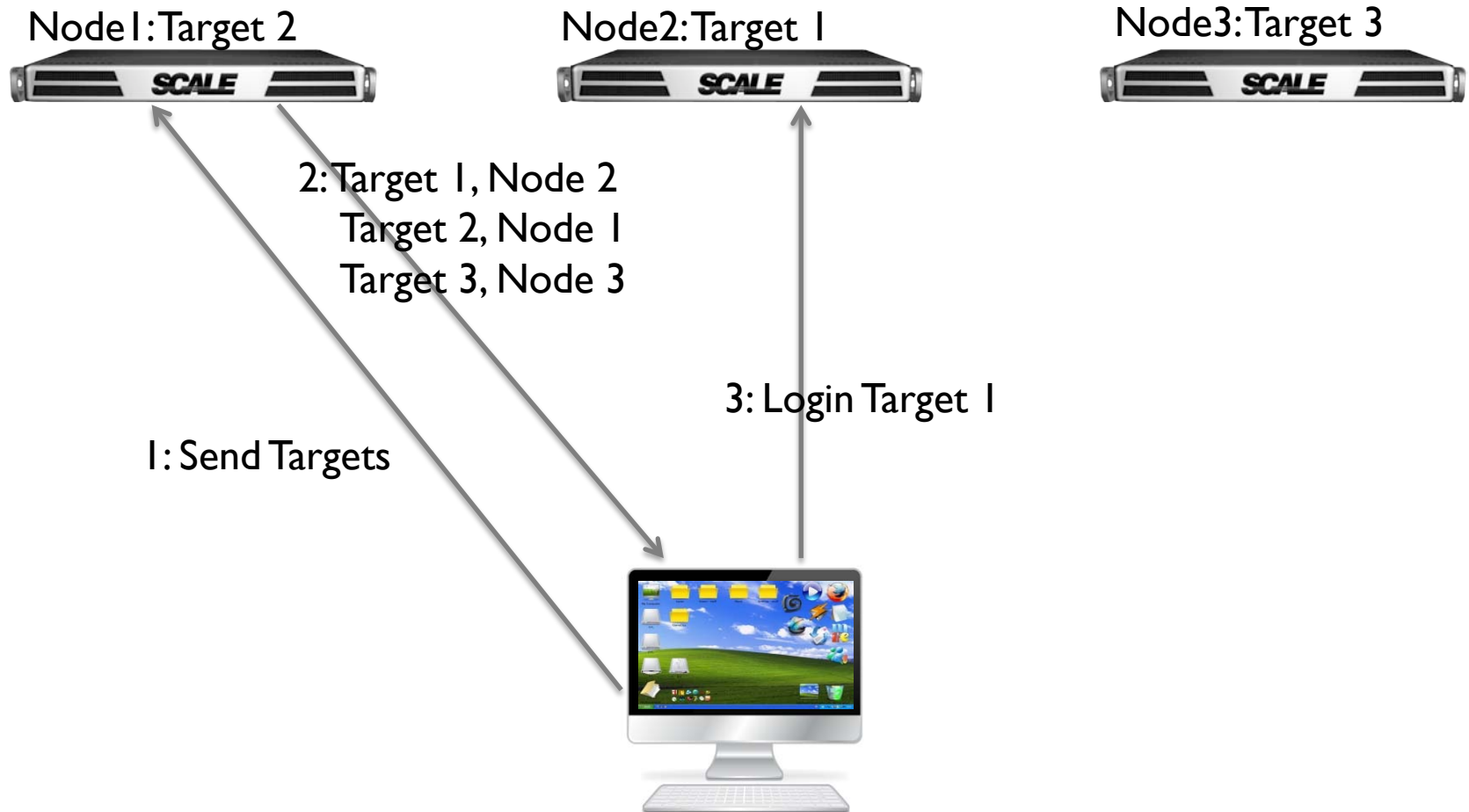


- ❑ Original SCSI implementation only provided cluster-wide reserve and release
- ❑ Implementing SCSI Persistent Reservations in a clustered iSCSI environment
 - ❑ Used iSCSI redirect to initiators to the node handling a specific target

Rationale, cont

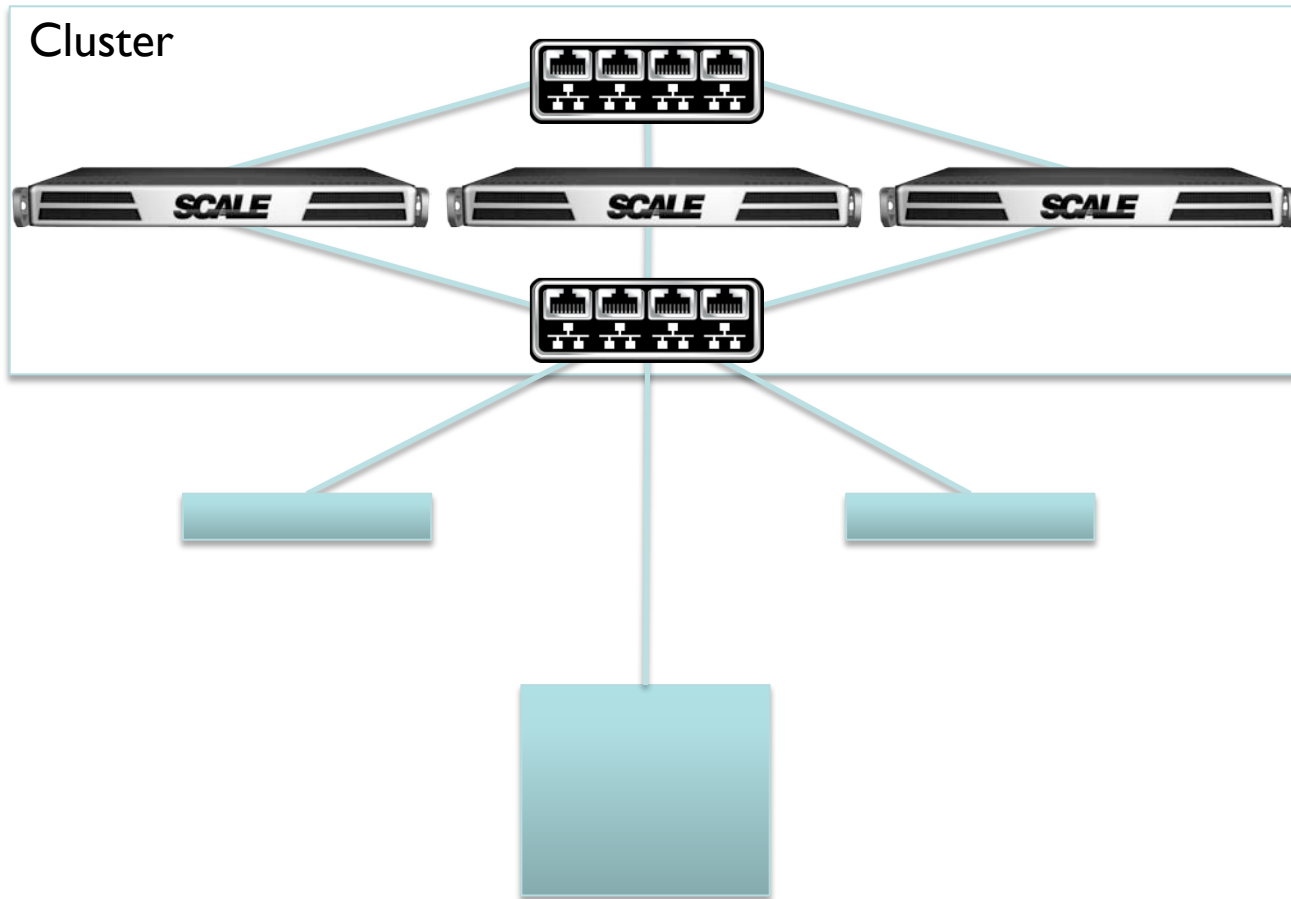


Rationale, cont



- Needed to test:
 - iSCSI redirection
 - Allocation of iSCSI targets to nodes
 - SCSI Functionality was not broken
 - Legacy targets still functioned correctly
 - Persistent Reservation and Reserve Release functionality correct

Scale Automated Testing Environment



Backplane Switch

Cluster Nodes

Cluster Switch

Linux/Windows
Test Nodes

Developer &
Automated
Build/Test Nodes

Scale Automated Testing Environment, D C

cont

STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2011

- ❑ Automated tests
 - ❑ Run after a build is complete
- ❑ Scale QA Daemon (scqad)
 - ❑ Server runs on each node (including Linux/Windows)
 - ❑ Use thrift APIs to access configuration functionality
 - ❑ scqad client (test program) runs on any build system
- ❑ Developer writes C++ test programs
 - ❑ Configure cluster as required
 - ❑ Test that the correct things happen
 - ❑ Testing for regressions

Alternatives for iSCSI/SCSI testing

- ❑ Use existing tools on a Linux node
 - ❑ sg3 utils
 - ❑ sg_persist, sg_inq, ...
 - ❑ Requires the uses of SSH from test code
 - ❑ Must use iscsiadm to connect to cluster
 - ❑ Figure out what SCSI Generic device to use, etc
 - ❑ Hard to parse the result
 - ❑ Does not allow easy testing of iSCSI redirects, etc

Alternatives for iSCSI/SCSI testing

cont

- ❑ Build a C++ class library layered on libiscsi
 - ❑ Gets us iSCSI testing
 - ❑ Requires that we create iSCSI & SCSI classes
 - ❑ Have to work around issues with libiscsi
 - ❑ Had to extend libiscsi as well 😊

Problems with sg3_utils

```
[root@localhost ~]# sg_inq /dev/sg2
```

standard INQUIRY:

```
PQual=0 Device_type=0 RMB=0 version=0x05 [SPC-3]  
[AERC=0] [TrmTsk=0] NormACA=0 HiSUP=1 Resp_data_format=2  
SCCS=0 ACC=0 TPGS=0 3PC=0 Protect=0 BQue=0  
EncServ=0 MultiP=0 [MChngr=0] [ACKREQQ=0] Addr16=1  
[RelAdr=0] WBus16=0 Sync=0 Linked=0 [TranDis=0] CmdQue=1  
length=36 (0x24) Peripheral device type: disk
```

Vendor identification: SCALE

Product identification: Storage Node

Product revision level: 56

Unit serial number: 605d2acf

Problems with sg3_utils, cont

```
[root@localhost ~]# sg_inq -e /dev/sg2
```

```
VPD INQUIRY, page code=0x00:
```

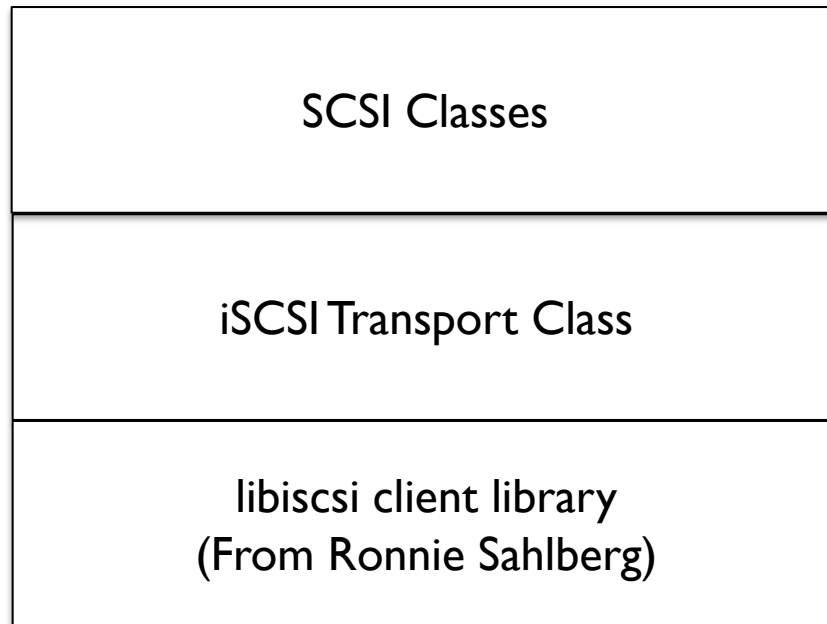
```
[PQual=0 Peripheral device type: disk]
```

```
Supported VPD pages:
```

0x0	Supported VPD pages
0x80	Unit serial number
0x83	Device identification

Approach Chosen

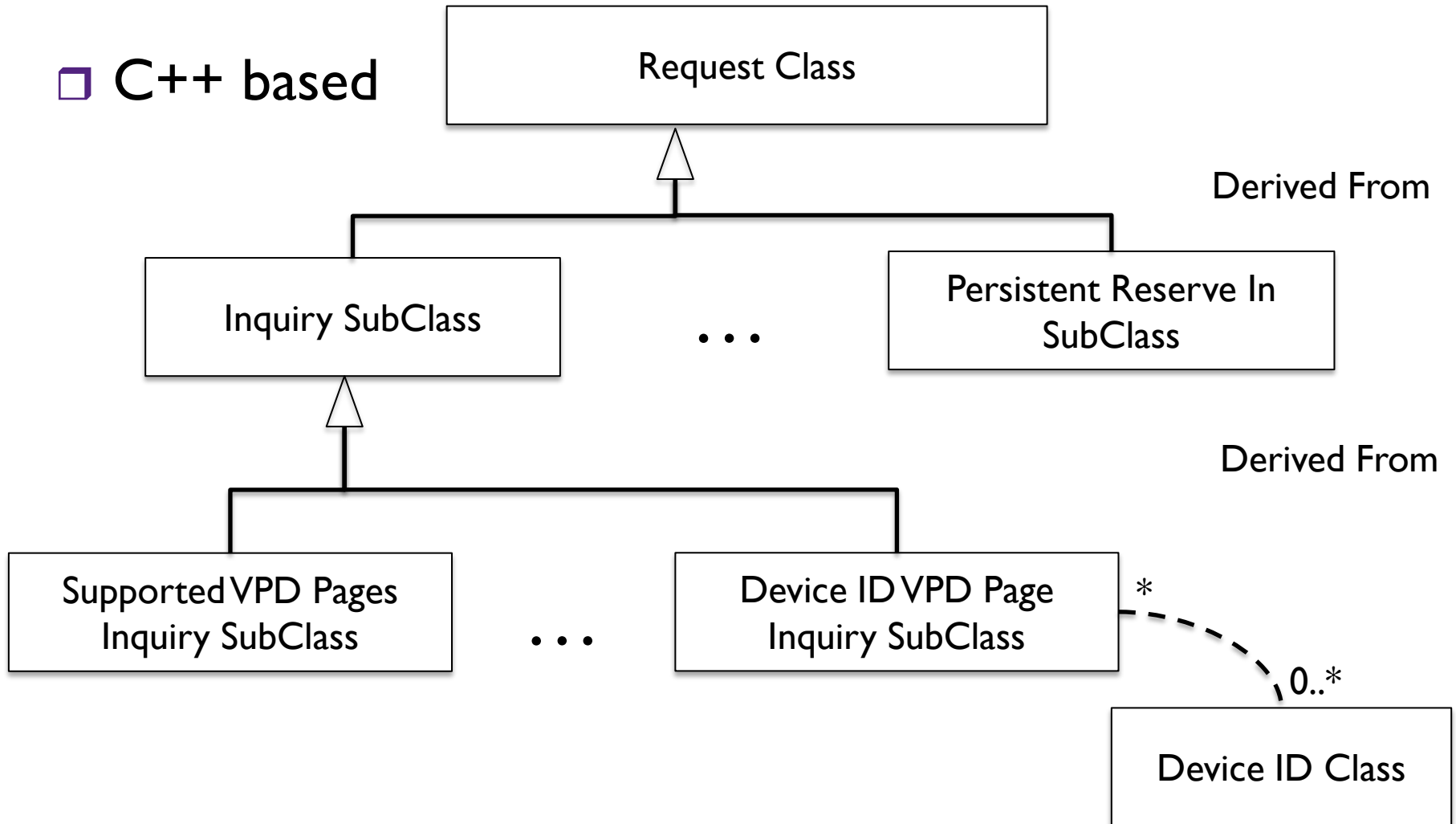
□ C++ iSCSI Client Library and simple SCSI Initiator



Approach Chosen, cont

- ❑ Uses C++ exceptions for handling problems
- ❑ Must use `try {...} catch (exception &e) {...}`

□ C++ based



SCSI Classes Implemented

- ❑ Request
- ❑ Inquiry
- ❑ PersistentReserveIn
- ❑ PersistentReserveOut
- ❑ Reserve
- ❑ Release
- ❑ ReadCapacity
- ❑ ReportLUNs
- ❑ TestUnitReady
- ❑ Read

SCSI Class Examples

Inquiry

```
class SCSIInquiry : public SCSIRequest
{
public:
    SCSIInquiry() :
        SCSIRequest(),
        mEvpd(false)
    {
        createInBuffer(36);
        setCdbByte(0, 0x12); // That's an INQUIRY
        setCdbShort(3, mInBufferSize);
        SetXferDir(SCSI_XFER_READ);
    }
}
```

...

SCSI Class Example, cont

Inquiry

```
SCSIInquiry(unsigned int allocationLength) :  
    SCSIRequest(),  
    mEvpd(false)  
{  
    createInBuffer(allocationLength);  
    setCdbByte(0, 0x12); // That's an INQUIRY  
    setCdbShort(3, mInBufferSize);  
    SetXferDir(SCSI_XFER_READ);  
}
```

SCSI Class Example, cont

SupportedVPDPages

```
class SCSIInquirySupportedVPDPages : public SCSIInquiry
{
public:
    // Default to 32 supported pages in the response
    SCSIInquirySupportedVPDPages();

    // Do we have that page?
    bool HasPage(uint8_t page);

private:
    unsigned int mPageCount;
};
```

SCSI Class Example, cont

SupportedVPDPages code

```
bool SCSIInquirySupportedVPDPages::HasPage(uint8_t page)
{
    unsigned int pageCount;

    // Check that the residual did not reduce the pages returned!
    // If we got an underflow, ie, there was less data than we asked
    // for, just use the returned count, otherwise subtract from
    // mPageCount the number of bytes that are missing, which will be zero
    // in the case that there was no residual :-)
    pageCount = (GetResidualType() == SCSI_RESIDUAL_UNDERFLOW) ?
                GetInBufferByte(3) : mPageCount - GetResidual();

    for (unsigned int i = 0; i < pageCount; i++)
        if (page == GetInBufferByte(4+i))
            return true;
    return false;
}
```

- Depends on the request
 - ReportLuns.GetLunCount()
 - ReportLuns.GetLun()
 - InquiryDeviceIdVPD.GetDescriptorCount()
 - InquiryDeviceIdVPD.GetDescriptor()
 - ...

- ❑ iSCSI transport management
- ❑ iSCSIBackgroun for concurrency management
 - ❑ libiscsi is synchronous, can't handle NOP IN when your code is not calling into libiscsi
- ❑ DiscoveryPair
 - ❑ Results of discovery

- ❑ SetInitiator
- ❑ SetTarget
- ❑ SetAddress
- ❑ GetTarget
- ❑ GetAddress
- ❑ IsRedirected
- ❑ iSCSIConnect
- ❑ iSCSIDiscoveryLogin
- ❑ iSCSIPerformDiscovery
- ❑ iSCSIDiscoveryLogout
- ❑ iSCSINormalLogin
- ❑ iSCSINormalLoginWithRedirect
- ❑ iSCSINormalLogout
- ❑ iSCSIDisconnect

iSCSI Methods, cont

- ❑ iSCSIExecSCSISync
- ❑ ISCSITaskAbort
- ❑ iCSITaskSetAbort
- ❑ iCSILUNReset
- ❑ iCSITargetWarmReset
- ❑ iCSITargetColdReset

A Sample Test

```
iSCSILibWrapper iscsi;
```

```
iscsi.SetInitiator(std::string("iqn.2011-07.com.testiscsi.initiator1"));
```

```
iscsi.SetTarget(std::string("iqn.2011-07.com.example:testtarget1"));  
iscsi.SetAddress(argv[1]);
```

```
try {  
    iscsi.iSCSIConnect();  
    iscsi.iSCSIDiscoveryLogin();  
    iscsi.iSCSIPerformDiscovery();  
}
```

A Simple Test, cont

```
std::vector<WrapperDiscoveryPair> &discoveryList =  
    iscsi.GetDiscoveryList();
```

```
if (discoveryList.size() == 0)  
    throw CException("No targets defined!");
```

```
iscsi.iSCSIDiscoveryLogout();  
iscsi.iSCSIDisconnect();
```

```
// Now connect to one of the discovered targets  
iscsi.SetTarget(discoveryList[1].GetTarget());  
iscsi.SetAddress(discoveryList[1].GetAddress());
```

A Simple Test, cont

```
iscsi.iSCSIConnect();
iscsi.iSCSINormalLogin();

// Now, find out how many LUNs there are.
SCSIReportLuns *reportLuns = new SCSIReportLuns();

iscsi.iSCSIExecSCSISync(*reportLuns, 0); // Always for LUN 0
if (reportLuns->GetResidualType() == SCSI_RESIDUAL_OVERFLOW)
{
    // get a new SCSIReportLuns with larger size buffer and retry
    unsigned int size = reportLuns->GetInBufferTransferSize() +
        reportLuns->GetResidual();
    // Etc
}
```

A Simple Test, cont

```
for (unsigned int i = 0; i < reportLuns.GetLunCount(); i++)
{
    unsigned int lun = reportLuns.GetLun(i);

    SCSI TestUnitReady tur;

    iscsi.iSCSIExecSCSISync(tur, lun);

    if (tur.GetStatus() != SCSI_STATUS_GOOD)
    {
        // Check if SCSI_SENSE_ASCQ_BUS_RESET etc
    }
}
```

A Simple Test, cont

```
SCSIInquiry inq; // Just a normal inquiry
iscsi.iSCSIExecSCSISync(inq, lun);

if (inq.GetStatus() != SCSI_STATUS_GOOD)
{ // Check why ... }

printf("T10 Vendor ID for LUN %u is \"%s\"\n",
      lun, inq.GetT10VendorID().c_str());
printf("Product ID for LUN %u is \"%s\"\n",
      lun, inq.GetProductID().c_str());
printf("Product Rev for LUN %u is \"%s\"\n",
      lun, inq.GetProductRev().c_str());
```

- ❑ Add support for more transports
 - ❑ FCoE?
 - ❑ SCSI PassThrough on Linux or Windows?
- ❑ Implement full set of SPC/SBC commands?
 - ❑ Exercise for the reader
- ❑ Implement SSC
 - ❑ Exercise for the reader

- ❑ The source has been released under the GPL
 - ❑ [git@github.com:RichardSharpe/SCSITest.git](https://github.com/RichardSharpe/SCSITest.git)
- ❑ libiscsi also available under the GPL
 - ❑ [git://github.com/sahlberg/libiscsi.git](https://github.com/sahlberg/libiscsi.git)
- ❑ Use SWIG to develop
 - ❑ PERL interface
 - ❑ Python interface

- ❑ Send us any improvements

Conclusions

- ❑ Allowed Scale to quickly develop tests of iSCSI and SCSI functionality
- ❑ Almost a cookie-cutter approach
- ❑ New SCSI commands easy to implement