

# Scalla + Castor2

Andrew Hanushevsky

Stanford Linear Accelerator Center

Stanford University

27-March-07

Root Workshop

<http://xrootd.slac.stanford.edu>

**Castor2/xrootd Integration – The Details**

# Outline

---

## # Introduction

- Design points
- Architecture
- Some Details

## # Data serving and Clustering

- xrootd and olbd

## # Casor2 Integration

- Dirty Details

## # Conclusion



# What is **Scalla**?

## # Structured Cluster Architecture for Low Latency Access

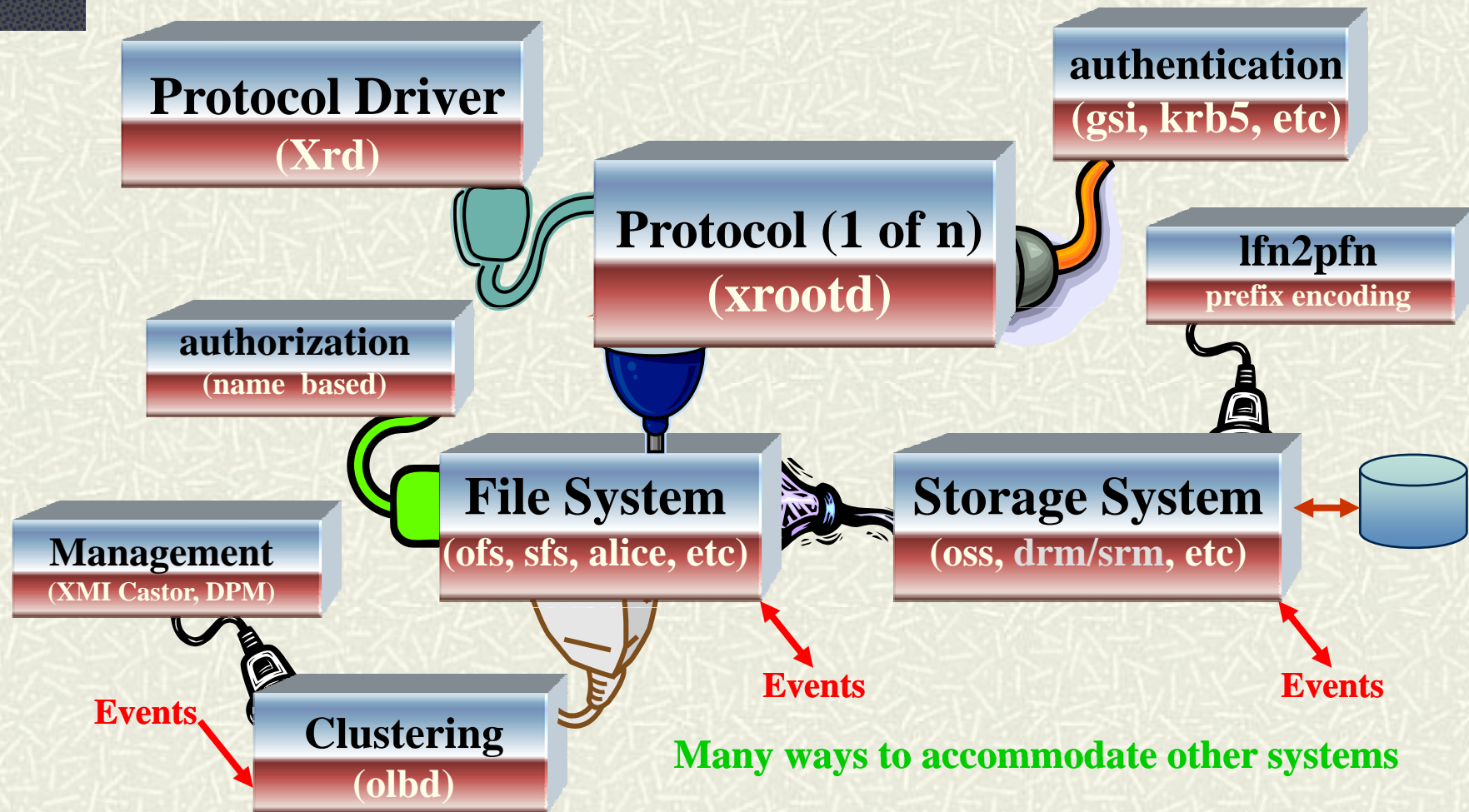
- Low Latency Access to data via **xrootd** servers
  - POSIX-style byte-level random access
    - By default, arbitrary data organized as files
    - Hierarchical directory-like name space
      - Does not have full file system semantics
  - Protocol includes high performance features
- Structured Clustering provided by **olbd** servers
  - Exponentially scalable and self organizing

# Scalla Design Points (the commercial)

- # High speed access to *experimental* data
  - Write once read many times processing mode
  - Small block sparse random access (e.g., root files)
  - High transaction rate with rapid request dispersal (*fast* opens)
- # Low setup cost
  - High efficiency data server (low CPU/byte overhead, small memory footprint)
  - Very simple configuration requirements
  - No 3<sup>rd</sup> party software needed (avoids messy dependencies)
- # Low administration cost
  - Non-Assisted fault-tolerance
  - Self-organizing servers remove need for configuration changes
  - No database requirements (no backup/recovery issues)
- # Wide usability
  - Full POSIX access
  - Server clustering for scalability
  - Plug-in architecture and event notification for applicability (HPSS, **Castor**, etc)



# xrootd Plugin Architecture



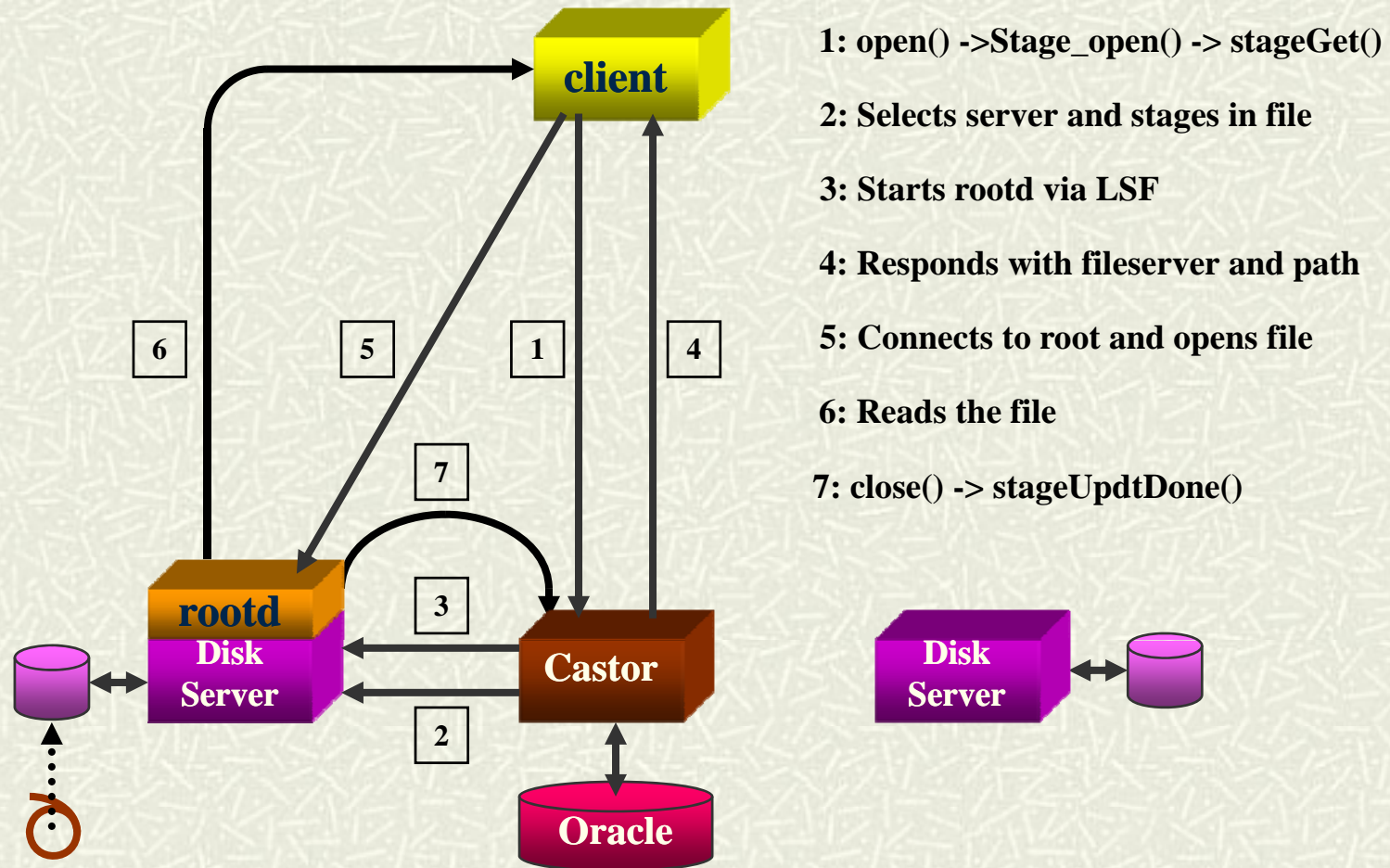
# The Upshot

---

- # Plug-in Architecture Plus Events
  - Easy to integrate other systems
- # Low Latency Data Server
  - Improved performance
    - Depending on integration open latency may be high
- # An actual integration follows
  - How Castor works now
  - How Castor works with xrootd



# Castor 2 Instance With Rootd



# Limitations & Solutions

## # Not Scalable

- One *rootd process* per open file
  - Resources are quickly consumed

## # Not Very Good Performance or Robustness

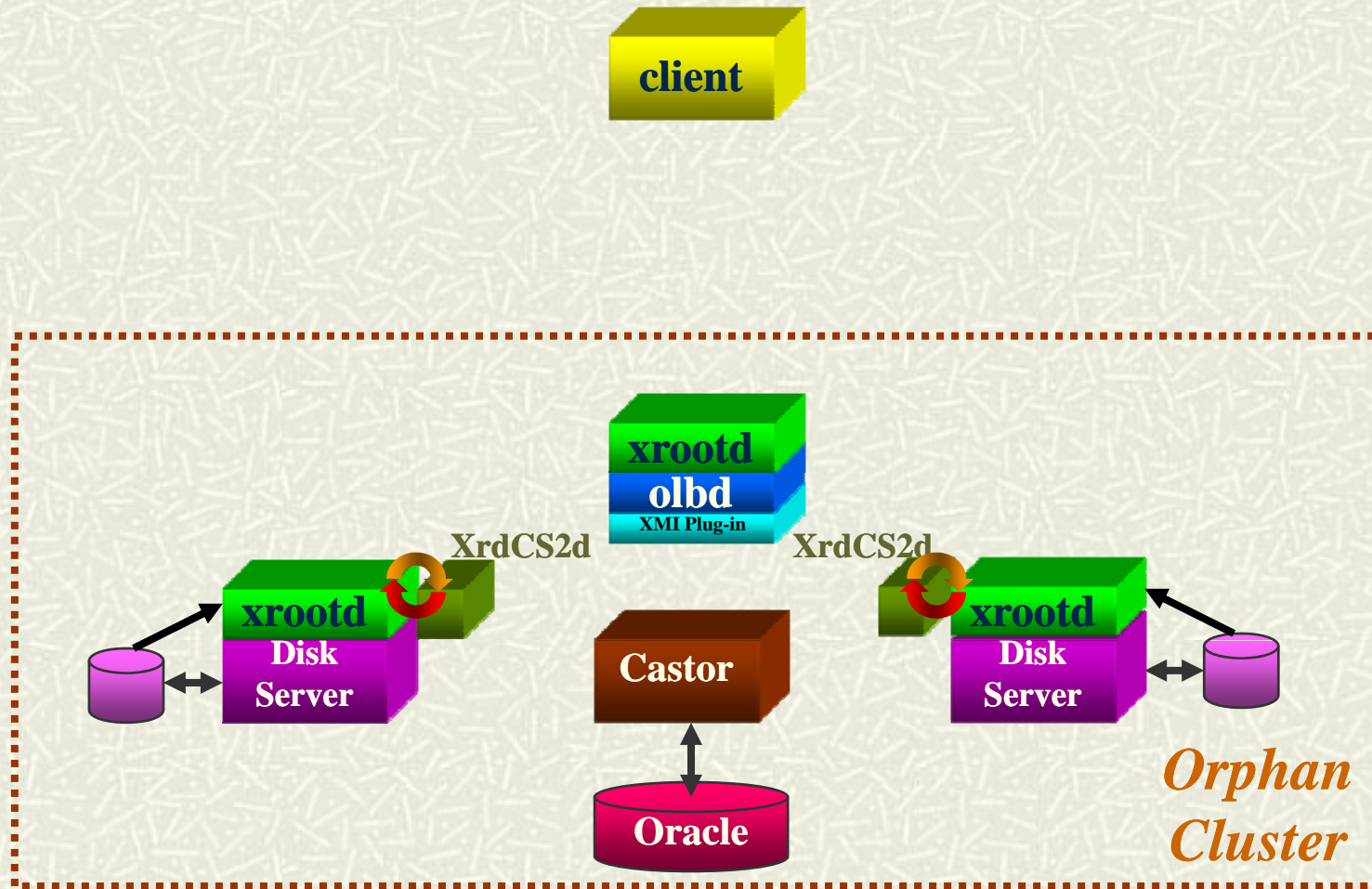
- *rootd* is old generation
  - We've learned a lot about servers since then

## # What We Really Want

- A long-lived ultra efficient data server
  - *xrootd* is a likely candidate
    - Plug-in architecture ideal for integration
      - So surprise here



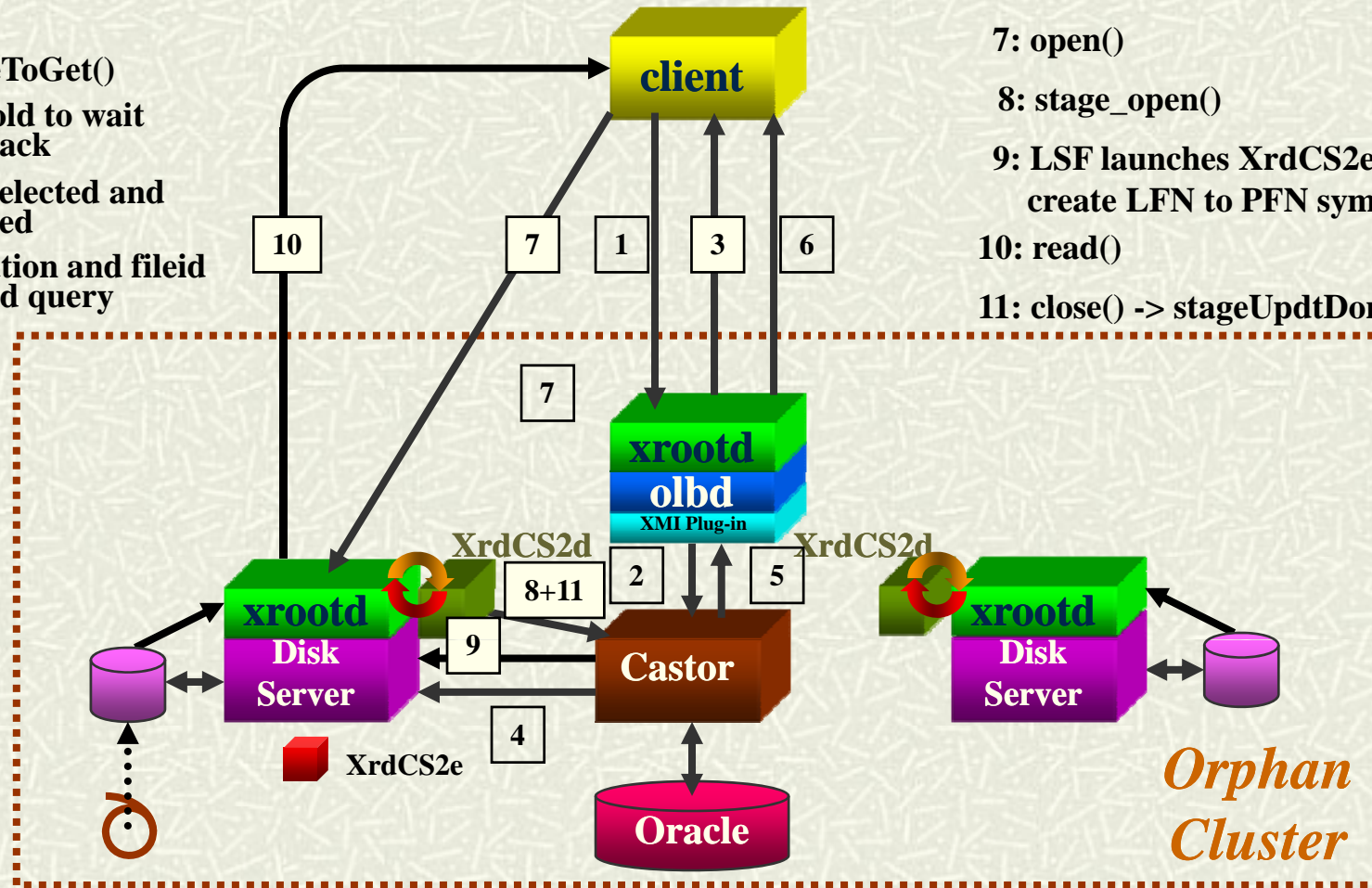
# Adding xrootd Access



# Using Orphan xrootd Access

- 1: open()
- 2: prepareToGet()
- 3: Client told to wait for callback
- 4: Server selected and file staged
- 5: Get location and fileid via polled query

- 6: Client redirected
- 7: open()
- 8: stage\_open()
- 9: LSF launches XrdCS2e to create LFN to PFN symlink
- 10: read()
- 11: close() -> stageUpdtDone()





# Limitations & Solutions

---

## # Performance only after open()

- All open requests must go through Castor
  - Good for management but bad for ultimate speed
  - Reliant on Castor availability
    - Single point of failure

## # Mitigate with xrootd based clustering

- Maneuver around Castor whenever possible
  - Reduce open() latency
  - Reduce single-point dependence

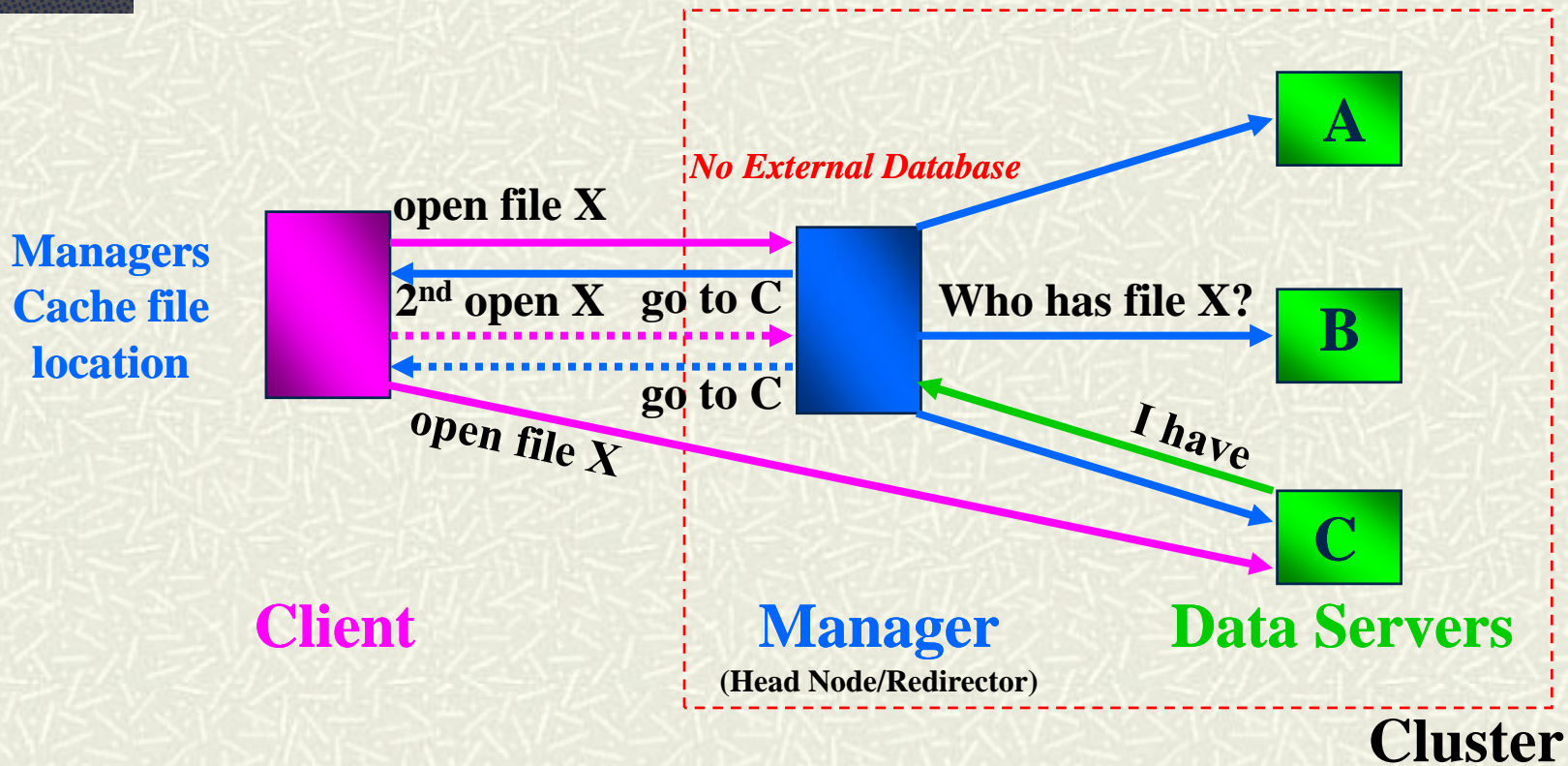
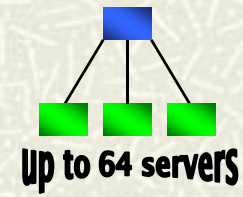
# Quick Note on Clustering

- # xrootd servers can be clustered
  - Increase access points and available data
  - Allows for automatic failover
- # Structured point-to-point connections
  - Cluster overhead (human & non-human) scales linearly
  - Cluster size is not limited
  - I/O performance is not affected
- # Always pairs xrootd & olbd servers
  - Symmetric cookie-cutter arrangement
  - Architecture can be used in very *novel* ways



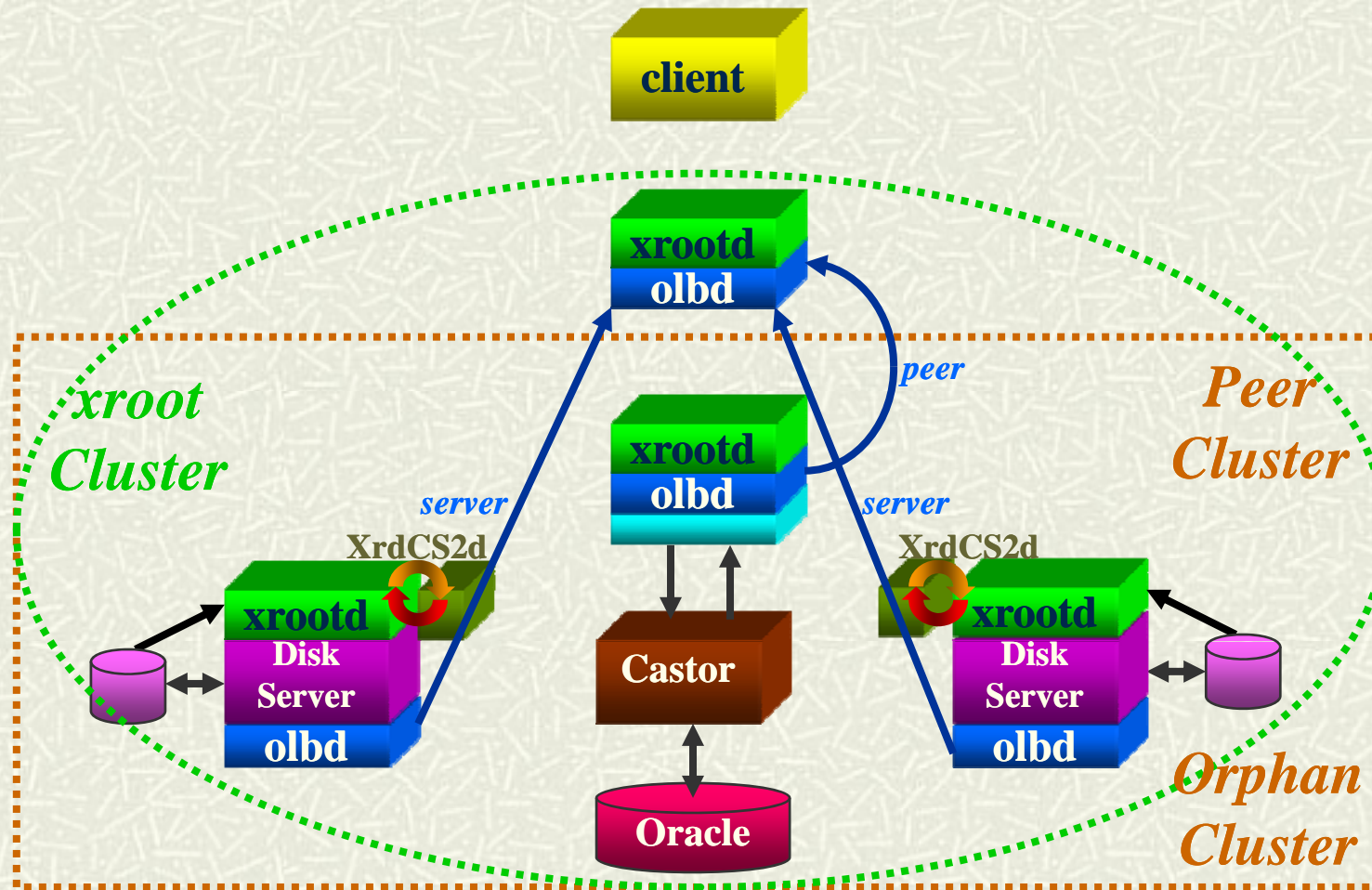


# File Request Routing



*Client sees all servers as xrootd data servers*

# Adding an xrootd Cluster

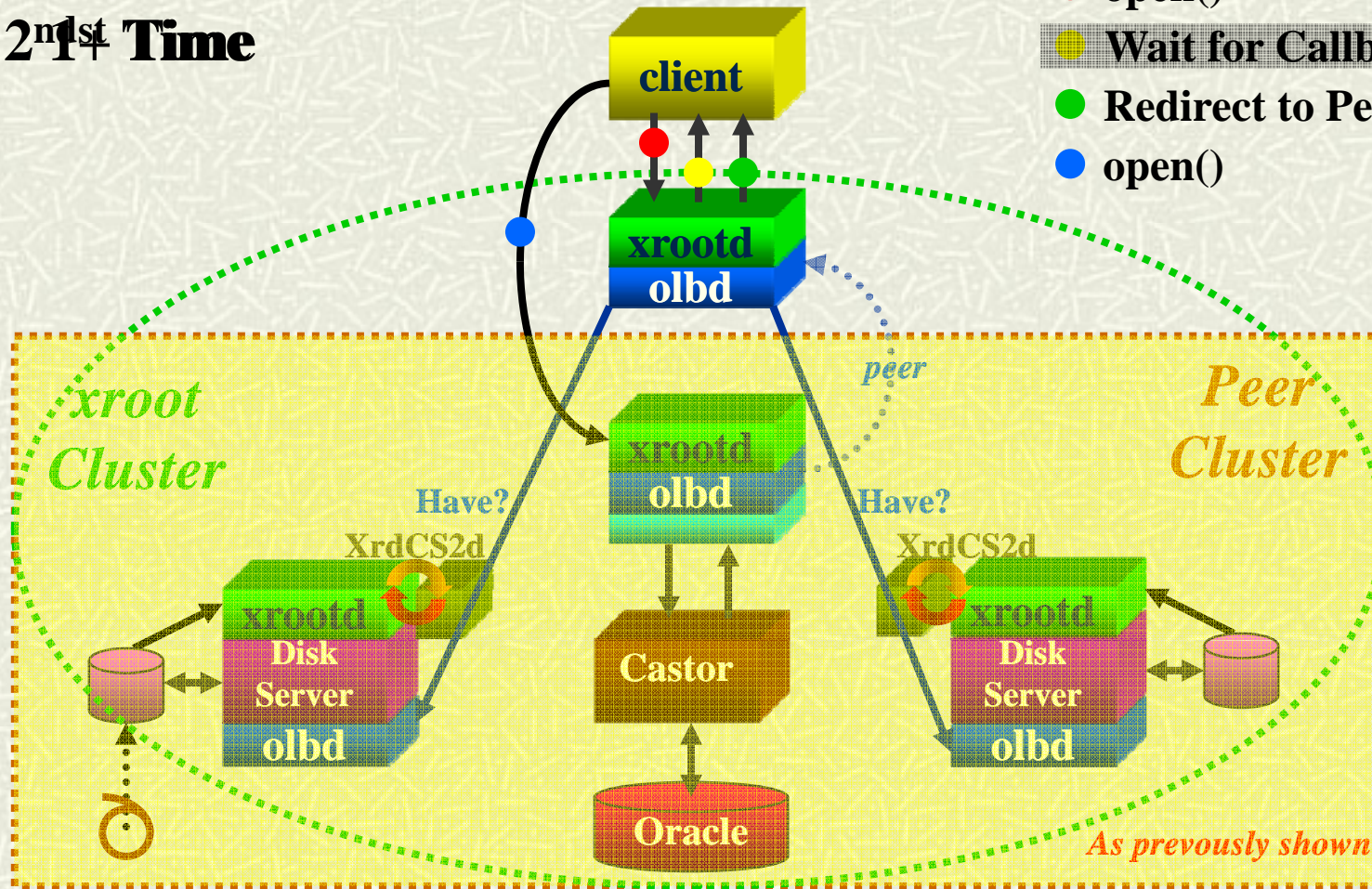




# Offline File Access

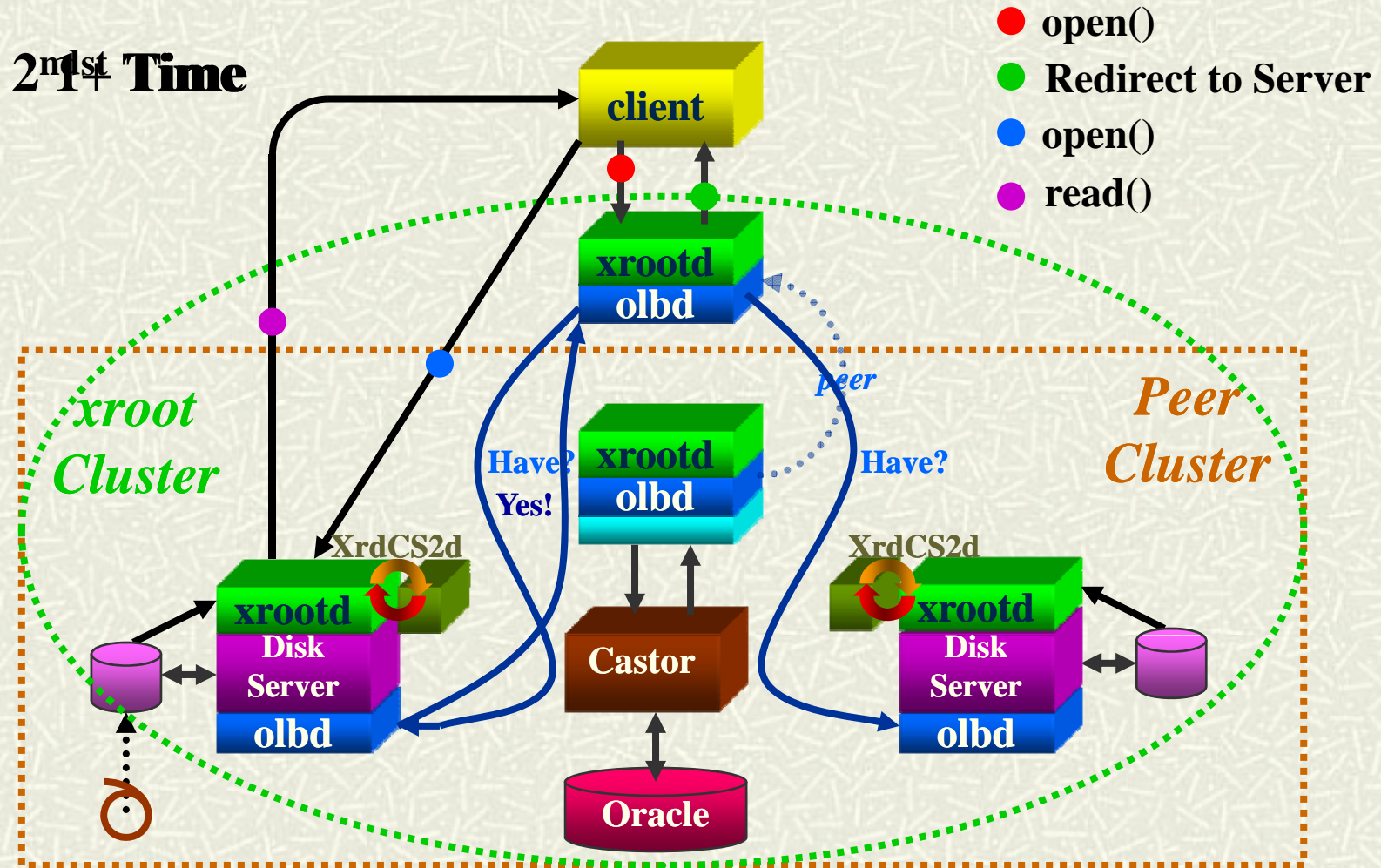
2<sup>nd</sup> Time

- open()
- Wait for Callback
- Redirect to Peer
- open()





# Online File Access

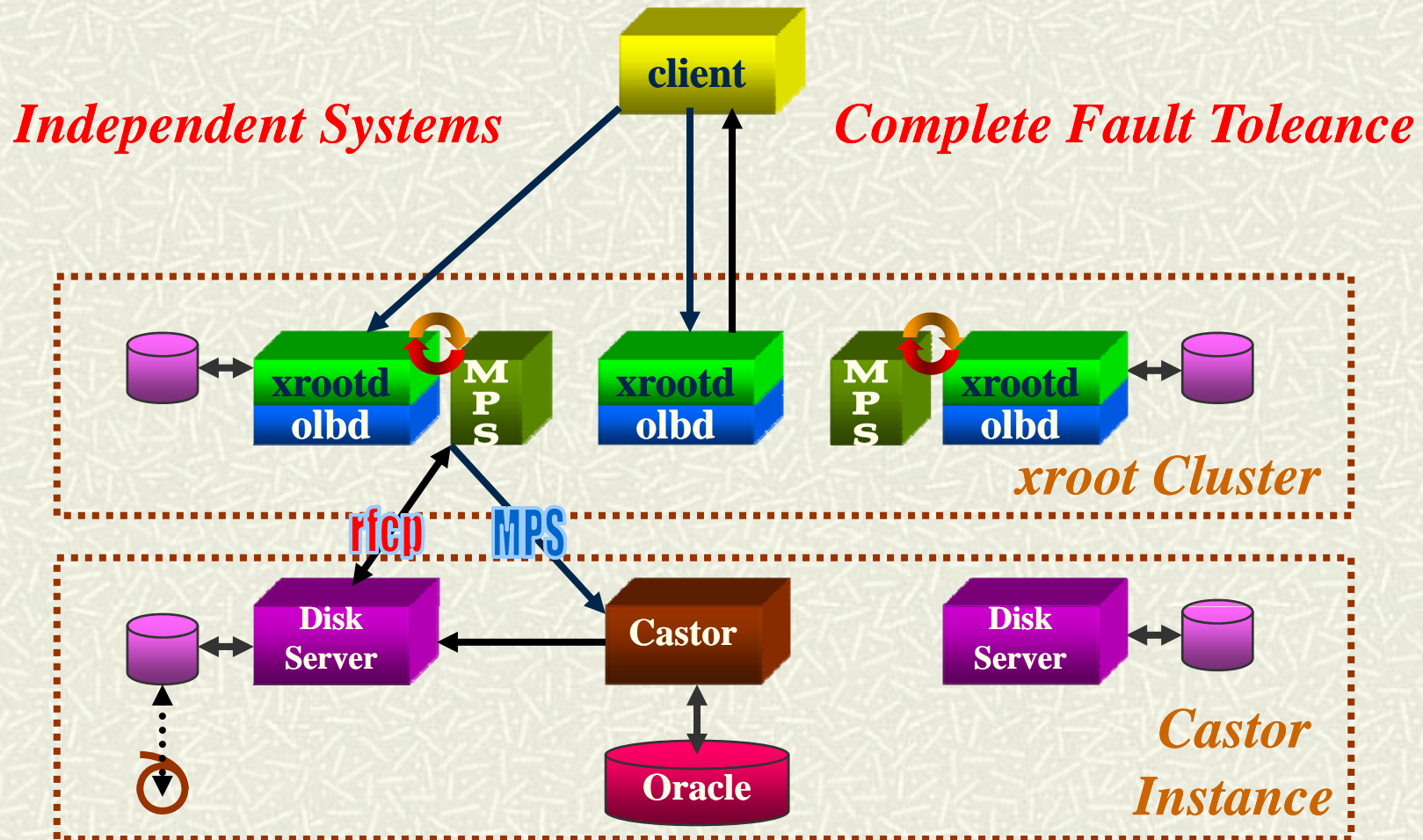




# Some Unsaid Things

- # File creation and open in write mode
  - Requests are always vectored through Castor
    - Doesn't address file creation/write scalability
- # Massive first time opens for a particular file
  - Requests will pile-up at the Castor redirector
    - Is not very efficient and has limited scalability
- # Is there another way?
  - Yes, complicated multicast call backs *or*
  - Simply additional hardware

# Alternative Castor Integration





# Advantages & Disadvantages

- # Both System Completely Independent
  - High degree of fault tolerance
    - **Better** recovery modes
  - Load isolation
    - **Better** throughput
  - Ability to speed match response time
    - **Better** transaction rate
- # Requires Additional Hardware
  - Servers and disk space
    - Approximately *one time* \$50-100K USD (depending on priorities)
      - However, about 1 to 2 FTE reduction in *ongoing* effort
- # Difficult to Share Resources Across Experiments
  - May be moot as most experiments are *always* active *and*
  - **Enables local control**

# Conclusion

- # Scalla can be integrated with a wide variety of systems
  - Castor integration test is starting
  - DPM (G-Lite) integration test is in full swing
    - Likely to be rewritten in the Castor model
- # Alternative integration modes should be considered
  - They may be classic but they have distinct advantages
- # Whatever the outcome it's the science that matters
  - Use the best model to crank out the analysis as fast as possible



# Acknowledgements

- # Castor Integration
  - Olof Barring, Sebastien Ponce, Rosa Maria Garcia Rioja
- # Software Collaborators
  - INFN/Padova: Fabrizio Furano (client-side), Alvise Dorigo
  - Root: Fons Rademakers, Gerri Ganis (security), Bertrand Bellenet (windows)
  - Alice: Derek Feichtinger, Guenter Kickingner
  - STAR/BNL: Pavel Jackl
  - Cornell: Gregory Sharp
  - SLAC: Jacek Becla, Tofigh Azemoon, Wilko Kroeger, Bill Weeks
  - BaBar: Pete Elmer (packaging)
- # Operational collaborators
  - BNL, CNAF, FZK, INFN, IN2P3, RAL, SLAC
- # Funding
  - US Department of Energy
    - Contract DE-AC02-76SF00515 with Stanford University