PVFS and more...

Julian M. Kunkel

Ein-/Ausgabe – Stand der Wissenschaft

15. April 2013

Outline

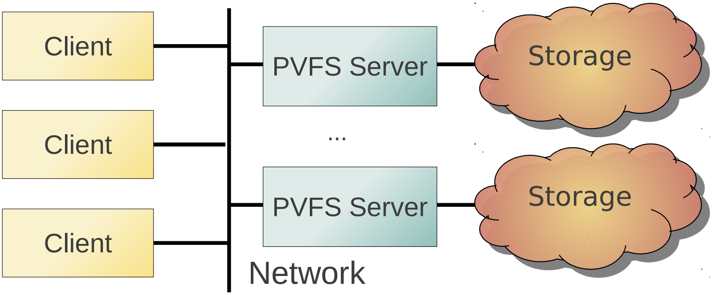
Outline

Overview of PVFS2

- Redevelopment of the Parallel Virtual File System
- Open source
- Developed at Argonne National Lab and Clemson University
- “OrangeFS” is an extended PVFS
- Commercial support by Omnibond
- Client-Server architecture

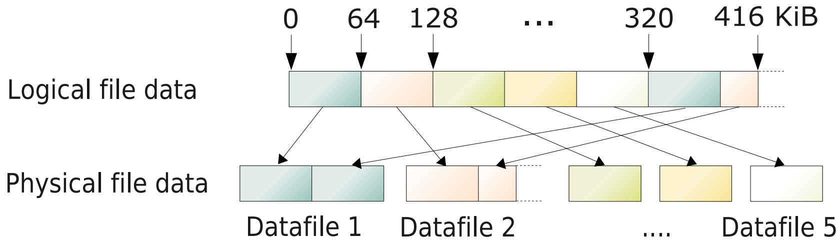
Logical View of a Parallel File System

- Multiple servers collaborate to provide the file system
- Concurrent access to file system objects is possible
- Data of one file is distributed among multiple servers
- PVFS server can be configured to manage data and/or metadata



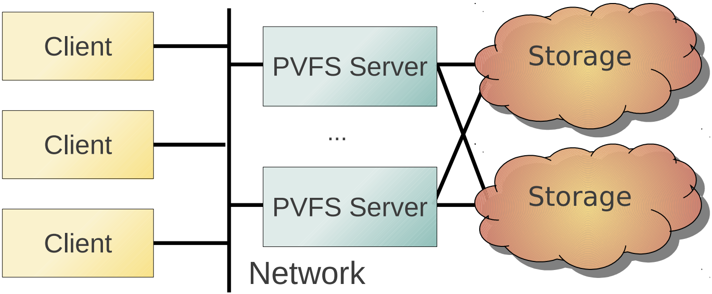
Distribution of Data

- Selectable distribution function
- Data is typically striped over data servers in 64 KByte chunks (RAID-0)
- A datafile (strip) is placed on exactly one server
- No software redundancy: relying on hardware

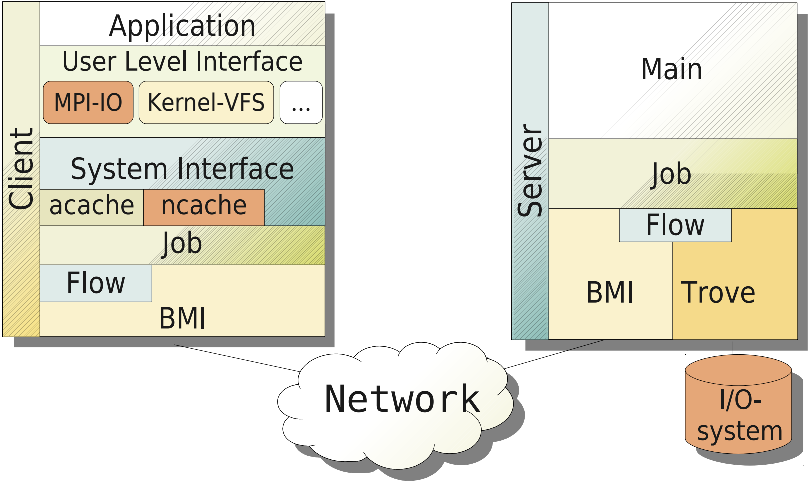


Fault-Tolerance and High-Availability

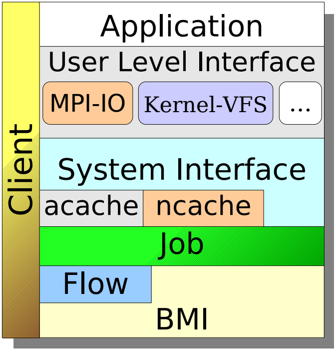
- No fault-tolerance mechanisms in software
- HA in hardware requires shared storage, e.g. Storage Area Network
- Multiple servers can access the same storage device
- Heartbeat (communication) between servers to detect failure
- Pairwise redundancy (active-active or active-passive)



Architecture of PVFS2



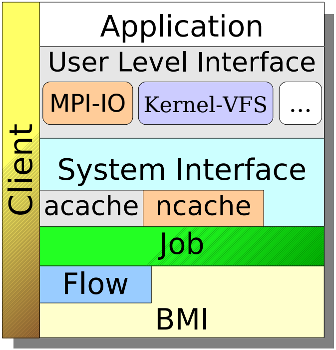
Architecture of PVFS2 - Client



Description of the layers

- User-level-interface
  - Integration into linux VFS for POSIX access
  - ROMIO module in MPICH2
- System Interface
  - Provides API for manipulation of file system objects
  - Contains caches for directory hierarchy and object attributes
- Job
  - Thin layer, controls lower layers

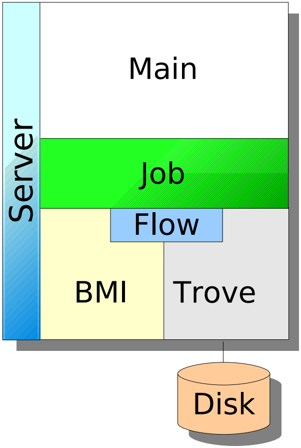
Architecture of PVFS2 - Client



Description of the layers

- Flow
  - Reliable transfer of data between two endpoints
  - Defines data flow policy e.g. parallel streams
- BMI
  - Network interface
  - TCP, Myrinet, IB, ...

Architecture of PVFS2 - Server



Description of the layers

- Main process
  - Accepts new requests
  - Starts server statemachines
- Trove
  - Persistency layer
  - Implementation uses:
    - Berkeley DB (metadata)
    - Local file system (data files)

Outline

Simple Model

Performance limitations

- CPU
  - Use hash tables etc. $\Rightarrow$ constant time needed per request
  - Limits the number of requests
- Input/Output subsystem
  - Access time
  - Throughput
- Network
  - Latency
  - Bandwidth $>$ Throughput

- Estimate and compare performance with measured throughput

Performance implications of the PVFS2 architecture

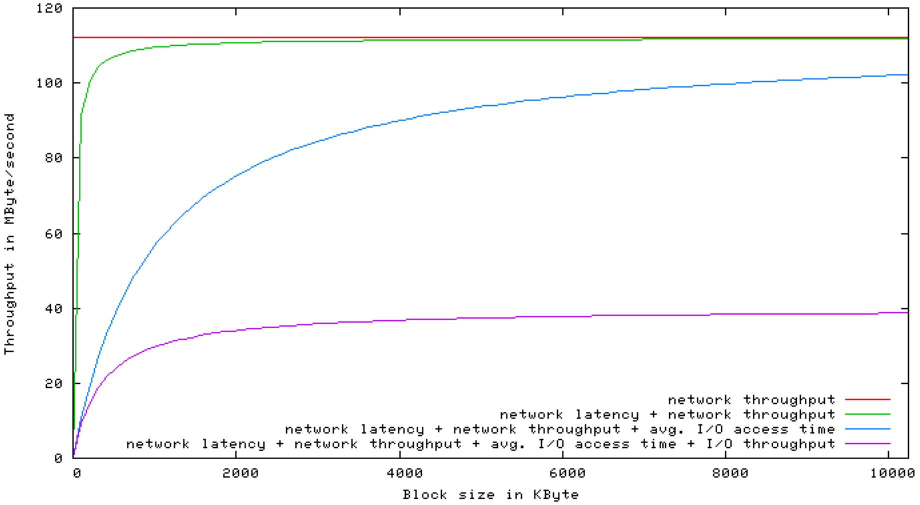
I/O

- No client side cache for data $\Rightarrow$ each I/O operation requires at least one message exchange
- Small I/O requests with initial requests (read) or response (writes)
- Larger requests require rendezvous protocol $\Rightarrow$ $+1$ round-trip (writes)

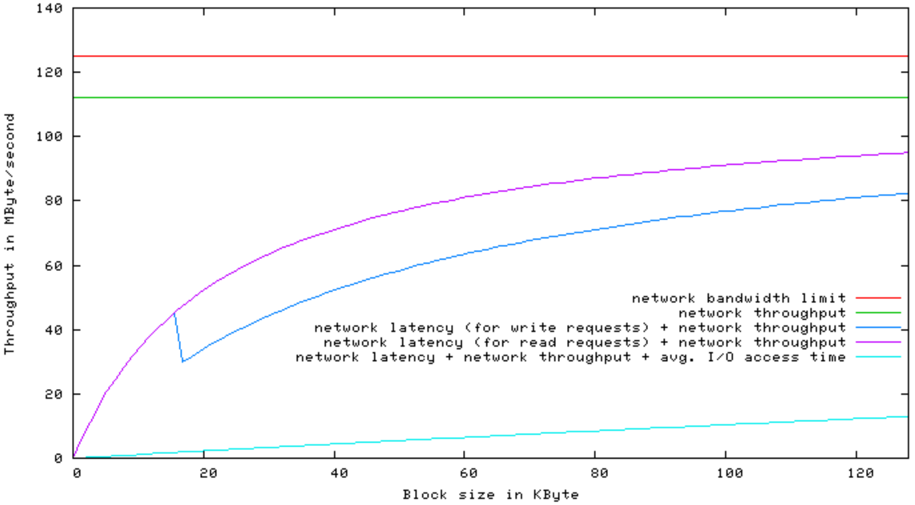
Metadata

- Read-only operations are cached for a small time frame
- Modifying operations typically consist of multiple requests
- Each request requires one message exchange

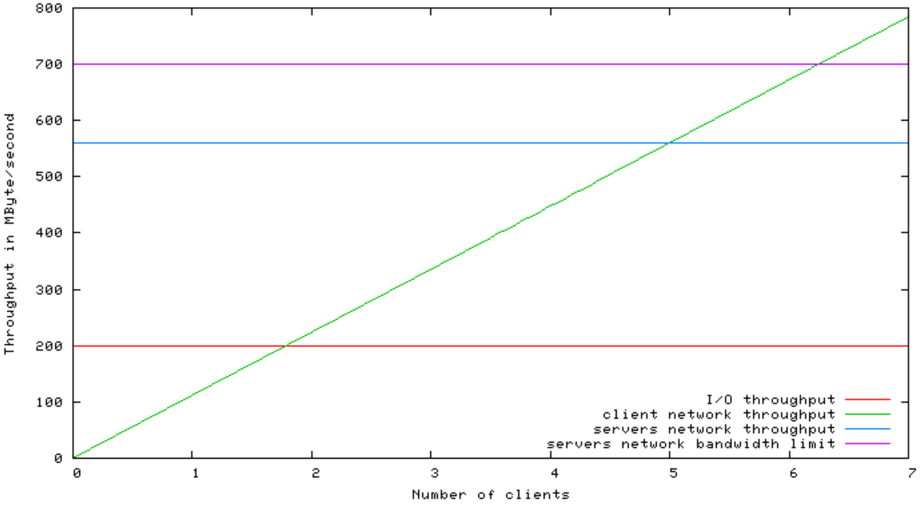
Estimated performance for small contiguous I/O requests



Estimated performance for small contiguous I/O requests



Estimated performance for large contiguous I/O requests

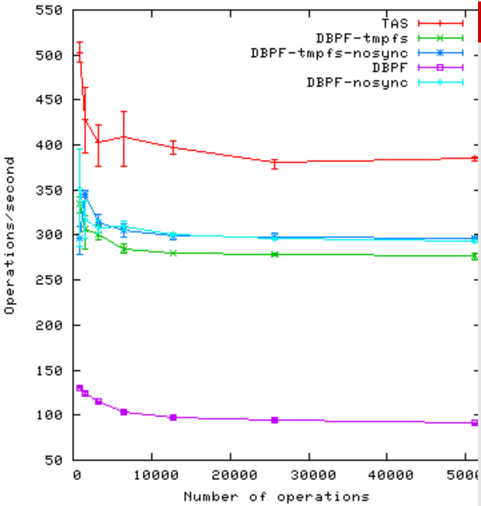


Outline

Benchmarking Program for Metadata

- MPI program
- Operates in one directory
- Each client creates a disjoint set of files with MPI_MODE_CREATE
- Runtime is measured on each process and maximum time used to calculate metadata throughput in operations/second.

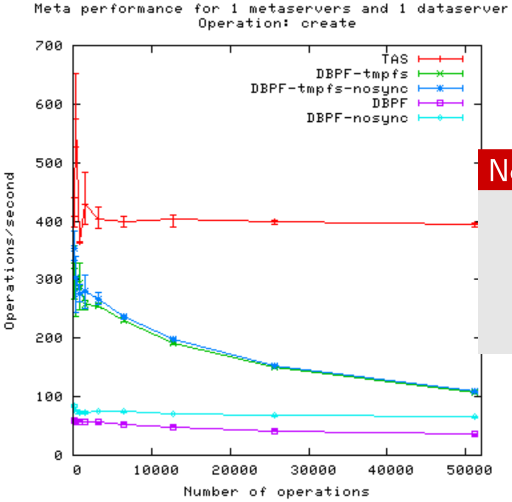
One data server, one metadata server, one client



Notes

- Data is from July 06
- DBPF is the current persistency implementation
- DBPF normally synchronizes modifications to disk, thus mainly limited by disk latency (and throughput)
- Nosync refers to a variant where metadata synchronization with the I/O subsystem is omitted

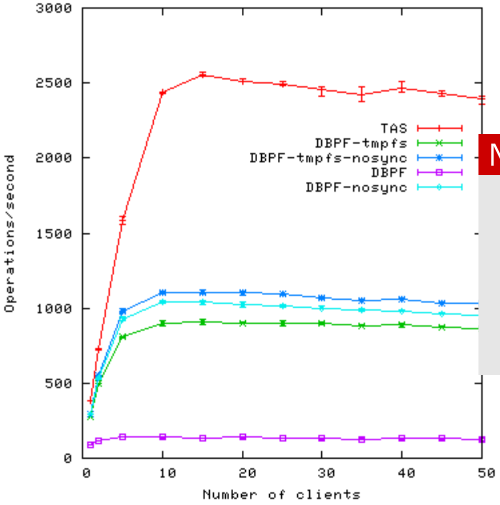
One data server, one metadata server, one client



Notes

- Data is from December 05
- Different metadata implementation

One data server, one metadata server



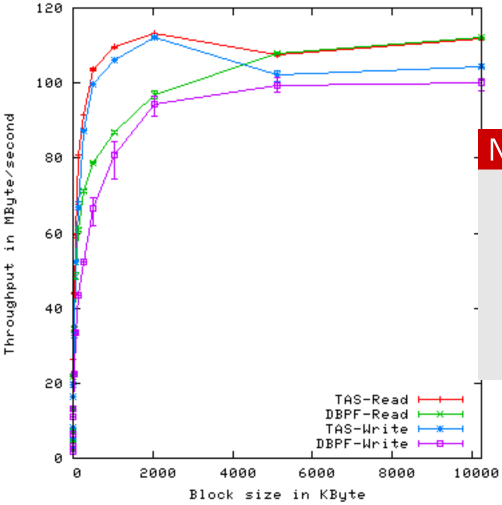
Notes

- Multiple clients creating a total of 51200 files
- DBPF tmpfs and nosync results are close together

Benchmarking Program for I/O Throughput

- MPI program (mpi-io-test)
- Operates on one file
- Each client
  - opens the file individually
  - writes a number of blocks of the same size with MPI_File_write
  - opens the file again
  - reads the data in chunks back
- The processes synchronize between two I/O operations
- Time is measured for each I/O operation and maximum taken

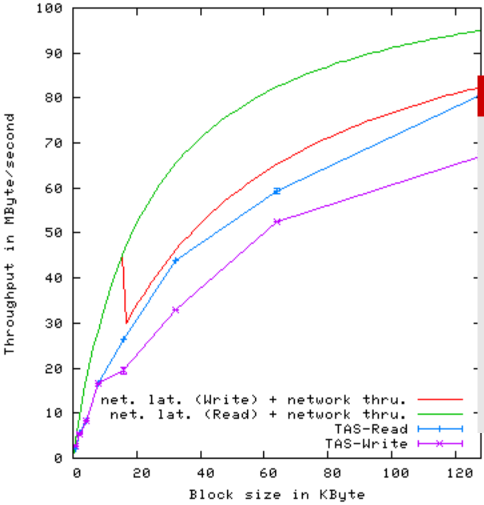
1 Client, 1 Data servers



Notes

- Total file size: 100 MByte
- Should be cached well by OS
- TAS forms a upper bound
- Close to network bandwidth

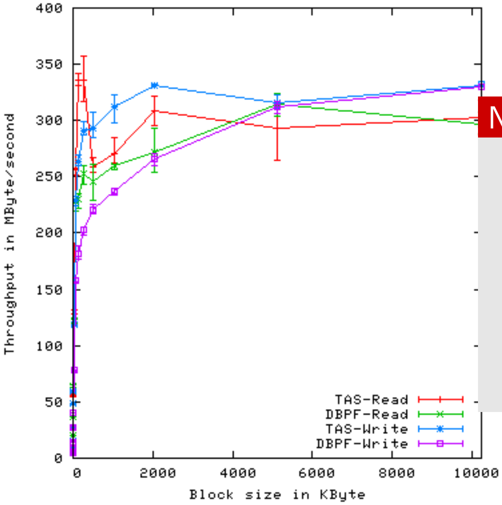
1 Clients, 1 Data server



Notes

- Estimated performance limits asymptoticly close to measured performance
- Drop due to handshake can be seen for write
- Caching strategies could boost throughput

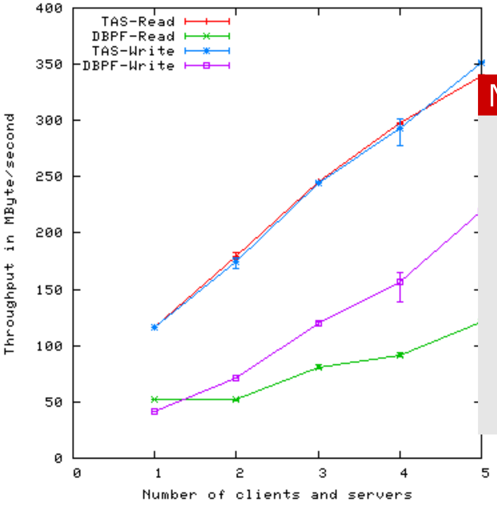
5 Clients, 5 Data servers



Notes

- TAS write performance > read performance
- Can be seen for DBPF with 10 MByte blocks
- About 3 times performance of 1 Client and 1 Data server

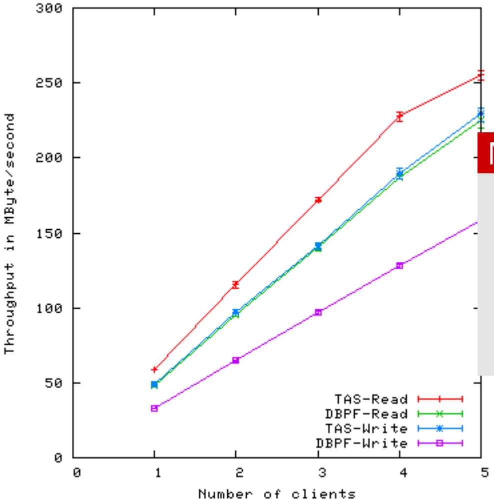
Variable number of clients and data servers



Notes

- 12800 MByte accessed in 10 MByte chunks
- Disjoint clients and servers
- Does not scale linear, network technology
- I/O throughput does not scale for reads

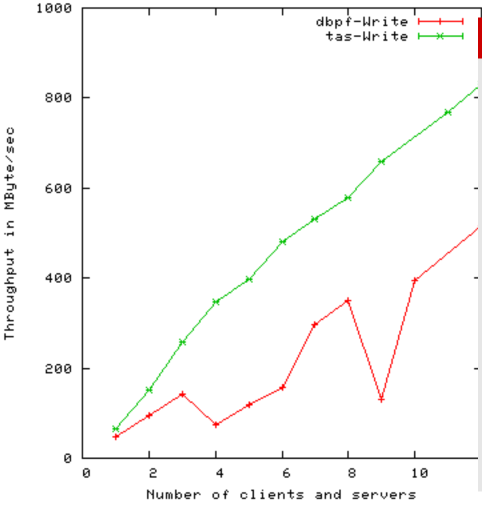
Variable number of clients and data servers



Notes

- Disjoint clients and servers
- 100 MByte accessed in 64 KByte chunks
- Performance scales linear

Chiba 150 MByte per client

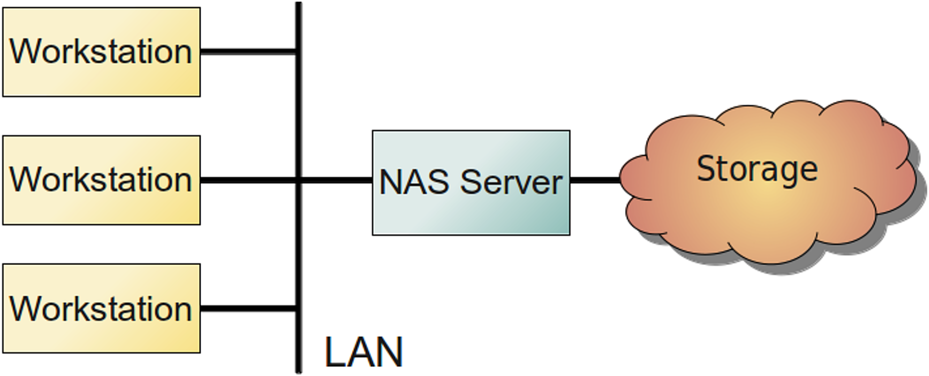


Notes

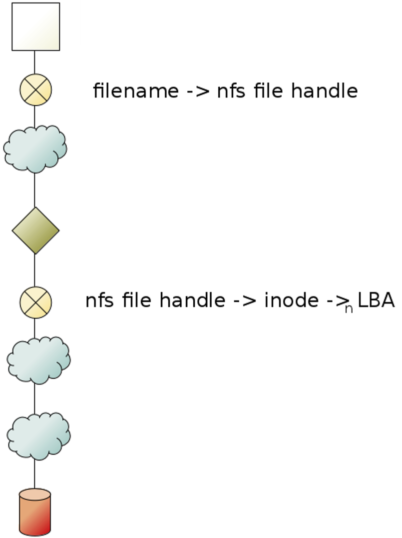
- Dual PIII 500 MHz, 512 MByte RAM, Myrinet 2000
- Effective throughput measured between two nodes: 90 Mbytes
- Hardware problems with myrinet interconnection, high packet loss
- Only qualitative analysis

Outline

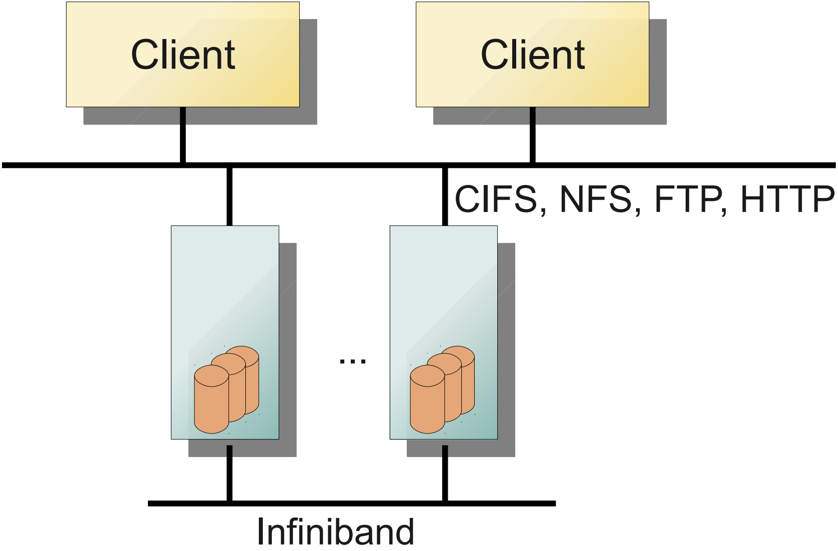
Network attached storage, e.g. NFS



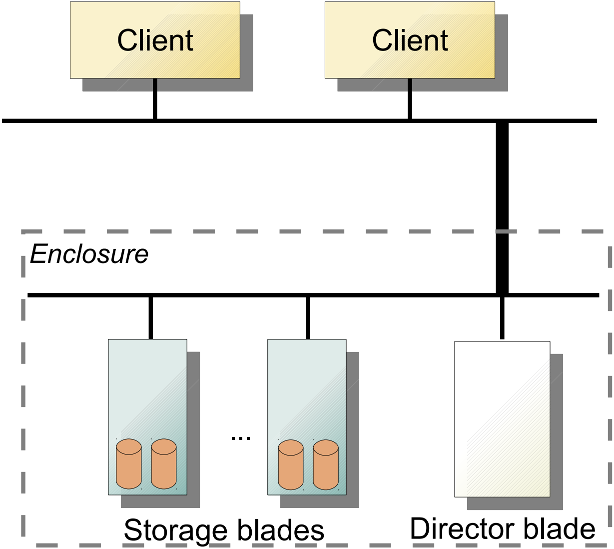
NFS Dataflow and Addressing (IOP model)



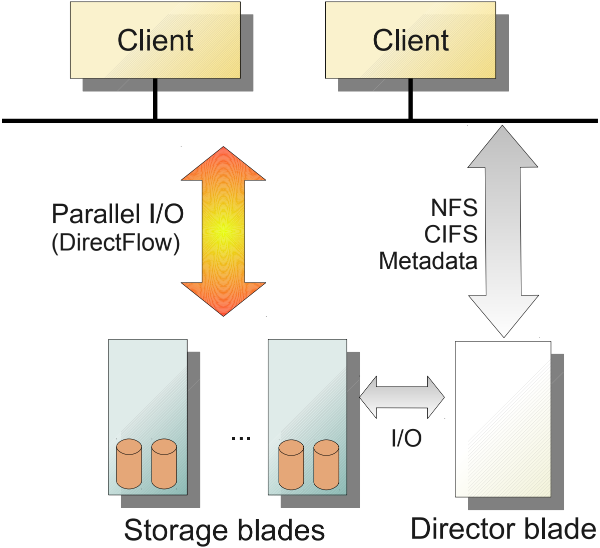
EMC / ISILON (OneFS)



Panasas ActiveStor (PanFS)



Alternative I/O Paths with ActiveStor



Outline

Summary

- Layered architecture of PVFS
- Hardware limits performance
- A performance reference is useful for comparison and evaluation
- Analysis stubs reduce complexity of analysis
- There are different I/O-paths

Eure Präsentationen

- Diese Präsentation ist NICHT repräsentativ für eure Präsentationen
- Einführung in Leistungsbewertung und zu viel Leistungsergebnisse
- Etwas über Firmen bzw. kommerziellen Hintergrund erzählen
- Details zu internen Algorithmen bspw. Optimierungsmöglichkeiten