IOPm

Modeling the I/O Path with a Functional Representation of Parallel File System and Hardware Architecture

Julian M. Kunkel, Thomas Ludwig

julian.martin.kunkel@informatik.uni-hamburg.de

Scientific Computing
Department of Informatics
University of Hamburg

2012-02-16

informatik
die zukunft

informatik die zukunft

Motivation

Goal

Systematic analysis of the data flow from client to block devices

Perspective

- Understanding enables analysis of existing bottlenecks
- Instrumentation of I/O path to gain knowledge

informatik
die zukunft

Need for a systematic analysis

Variability in the I/O path

- Architecture of supercomputers is diverse
  - Network topology, potential I/O forwarding, I/O subsystem ...
- File-systems
  - Architecture and configuration determine usage of hardware
- I/O software stack
  - I/O middleware, intermediate-layers, client-side caching, ...

informatik die zukunft

Logical/Physical view

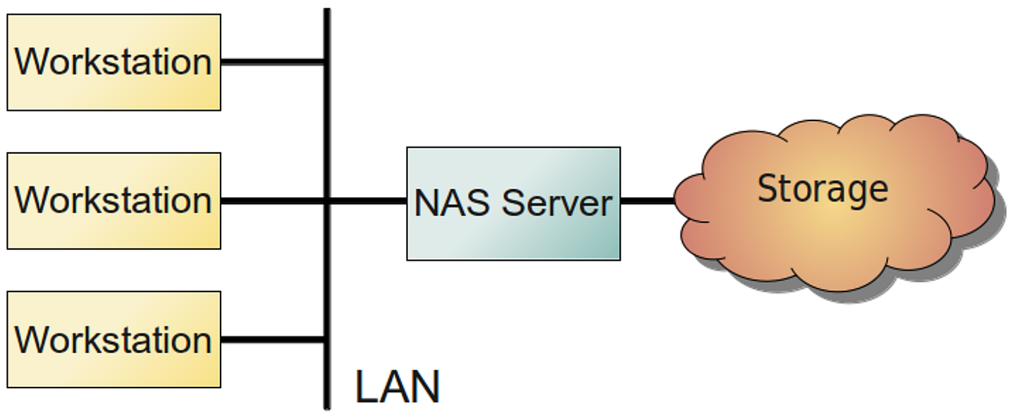


Figure: Logical view on network attached storage

- Component oriented, focuses on participating components
- Physical view very similar; deployed topology + hardware

Software architecture

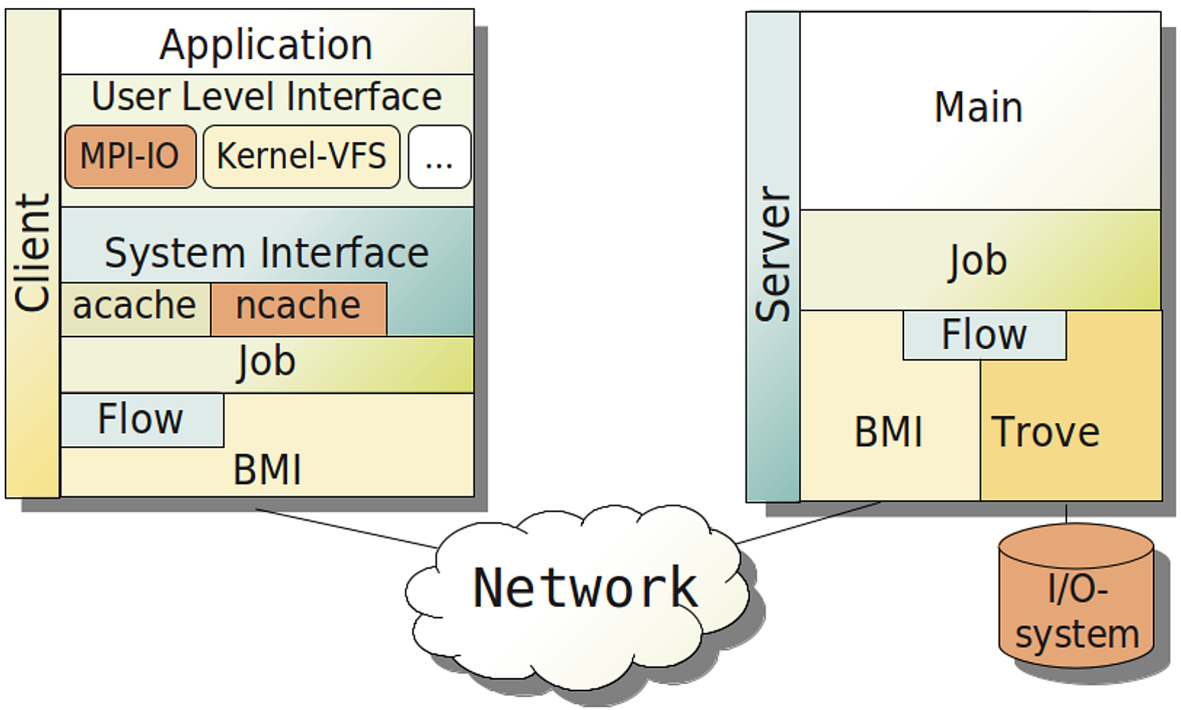
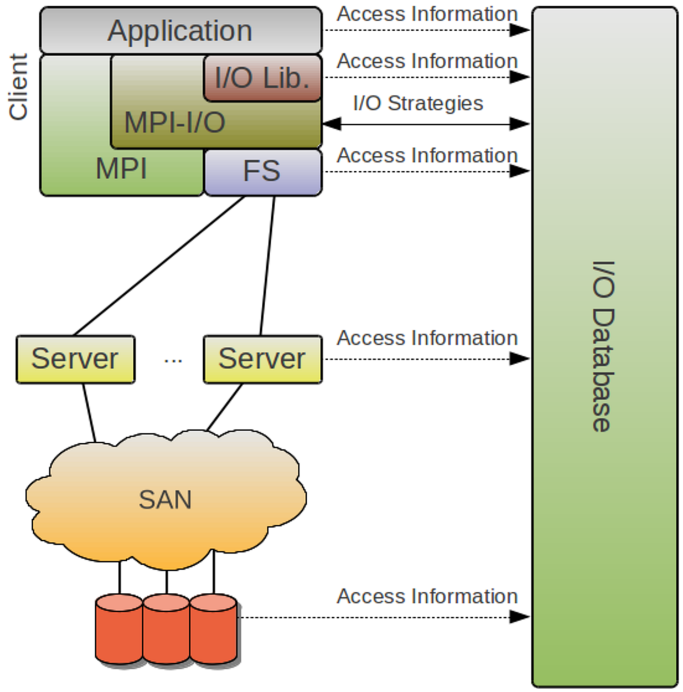


Figure: Software architecture of PVFS (with references to hardware)

SIOX:[1] Scalable I/O for Extreme Performance

Project approach

- Record access information on all layers and components
- Recognize access patterns
- Characterize the I/O system
- Localize the I/O bottlenecks
- Propose automatic optimizations



Requires deployment of monitors on involved layers/components

- Characterized by the I/O path

[1] http://www.hpc-io.org/

Introducing *IOPm*: I/O path model

Content

- IOPm abstraction level: focus on logical functionality
- Graphical representation
- Help for performance analysis

Logical functionality

- Client: initiates I/O
- Network: transports I/O
- Translation: changes the access reference
- Cache: may perform write-behind, read-ahead and scheduling
- Block storage
- (Redundancy)

Translation

Motivation

- User accesses logical files by “file name” in the namespace
- Reference to (meta)data of an object depends on the layer
  - Forwarding, altering or splitting of the previous reference
  - In most cases the information about the accessed “file” is lost
- Needed to track access of logical files across layers

Examples

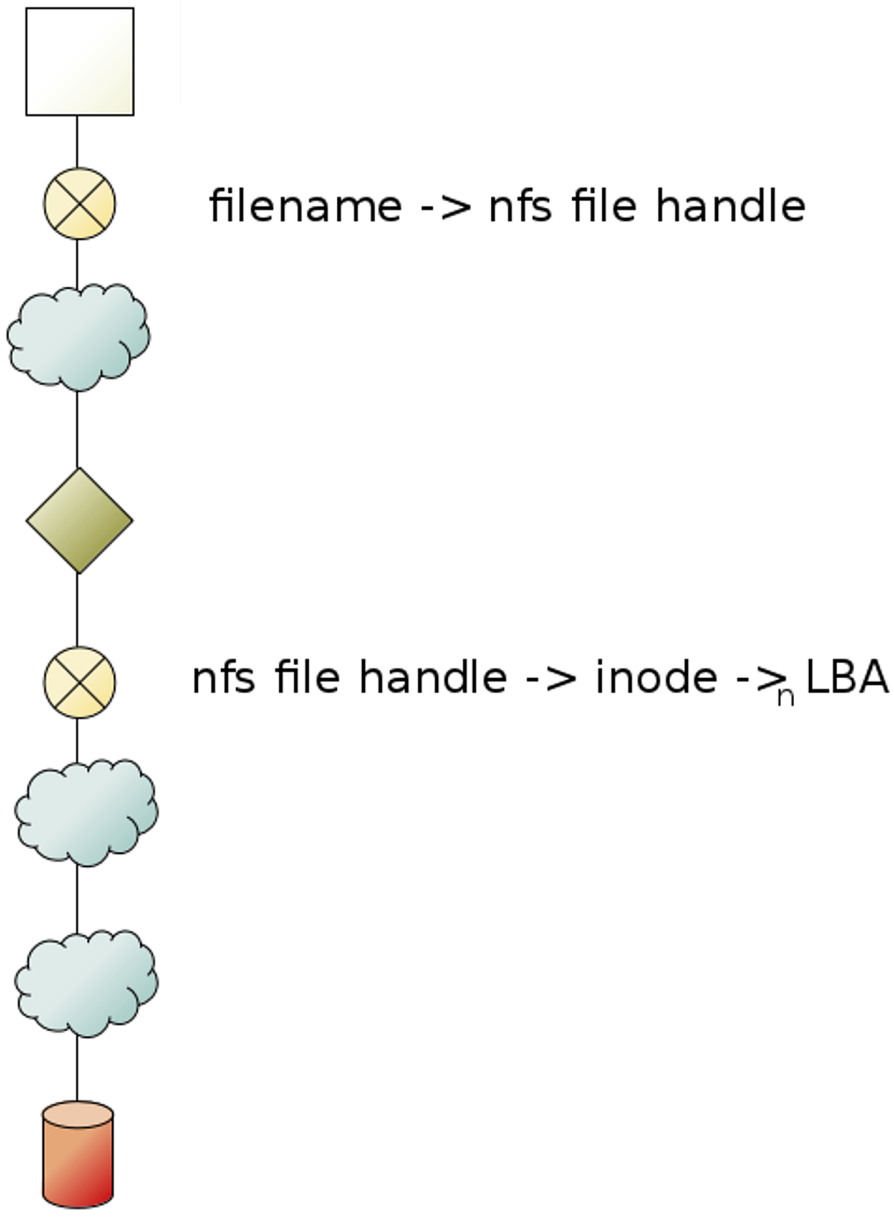
- Local file: file name $\Rightarrow_1$ file handle $\Rightarrow_1$ inode $\Rightarrow_n$ LBA
- PVFS file: file name $\Rightarrow_1$ metafile handle $\Rightarrow_n$ datafile handle $\Rightarrow_1$ (*server*, *local file*)

Graphical representation

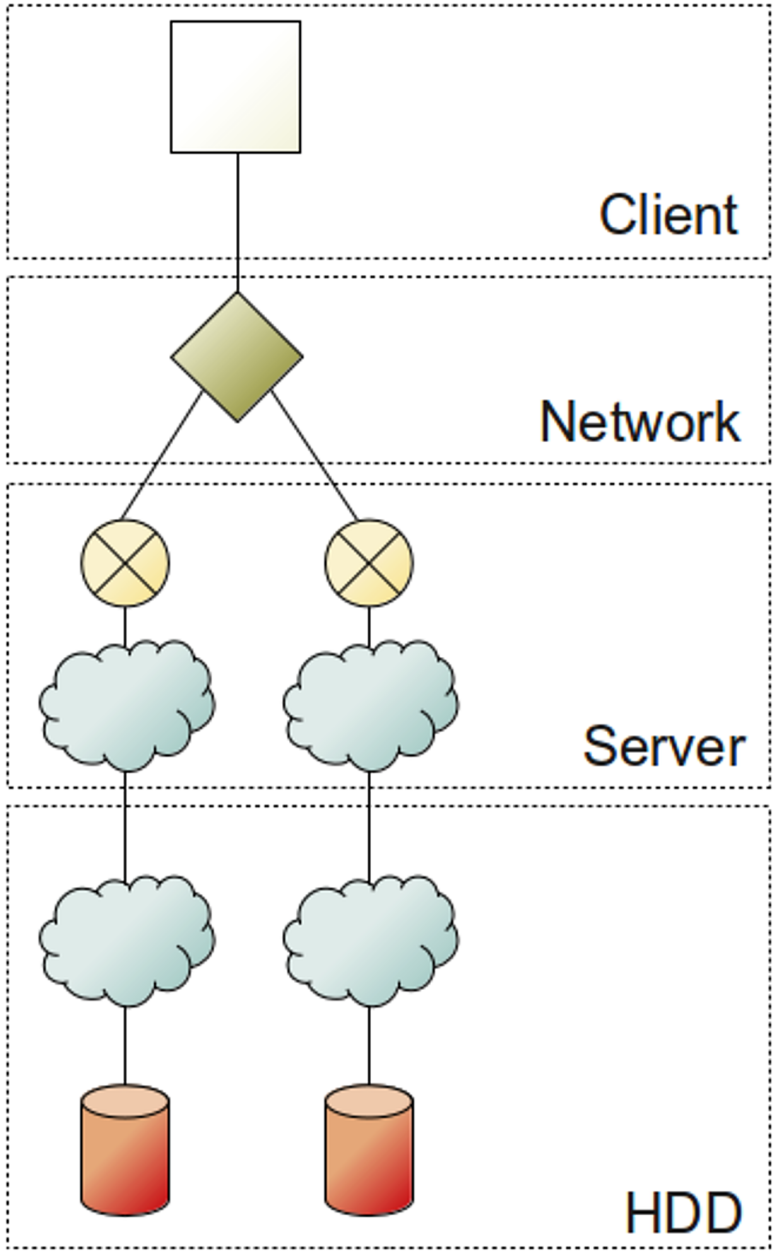
IOPm graph

- Shows all functionality involved in (one) I/O
- Symbols represent functionality
  - Attributes for Cache
  - Redundancy explicitly represented + relative overhead
- Edges represent interaction between functionality
  - Well defined in the I/O path
- Optional: boxes show physical deployment of functionality

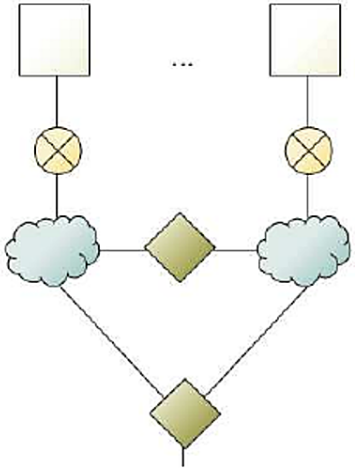
Example for an NFS server – simplified translations



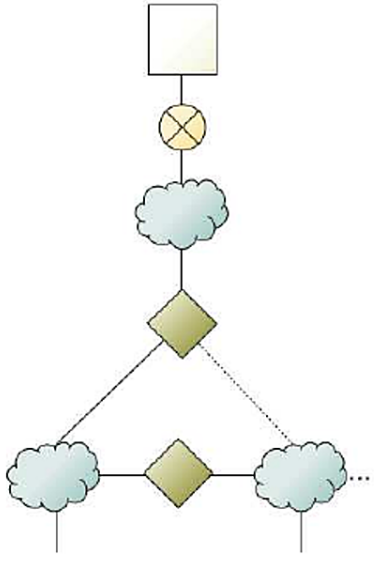
Example for accessing PVFS on two servers



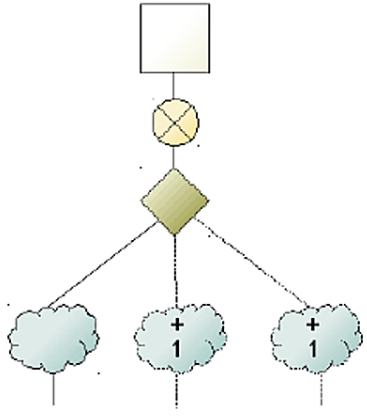
Various examples



(f) Cluster file system – meta data path

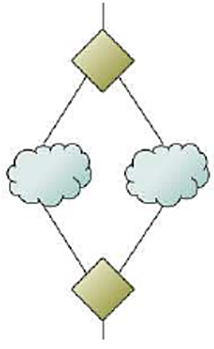


(g) Server to server communication

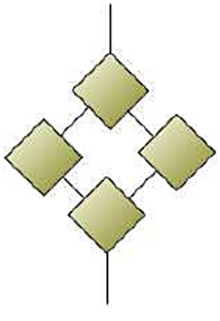


(h) Client-side replication scheme

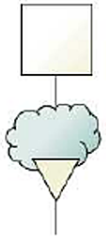
(i) Forwarder



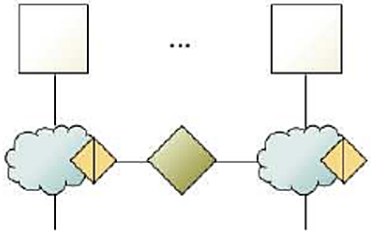
(j) SAN for two servers



(k) Alternative network path

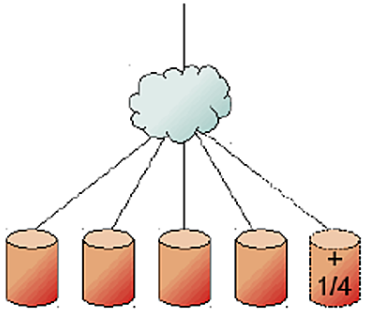


(l) I/O library

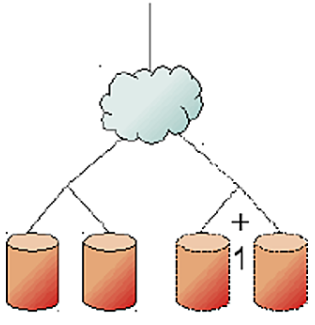


(m) MPI collective I/O

(n) RAID 1



(o) RAID 5

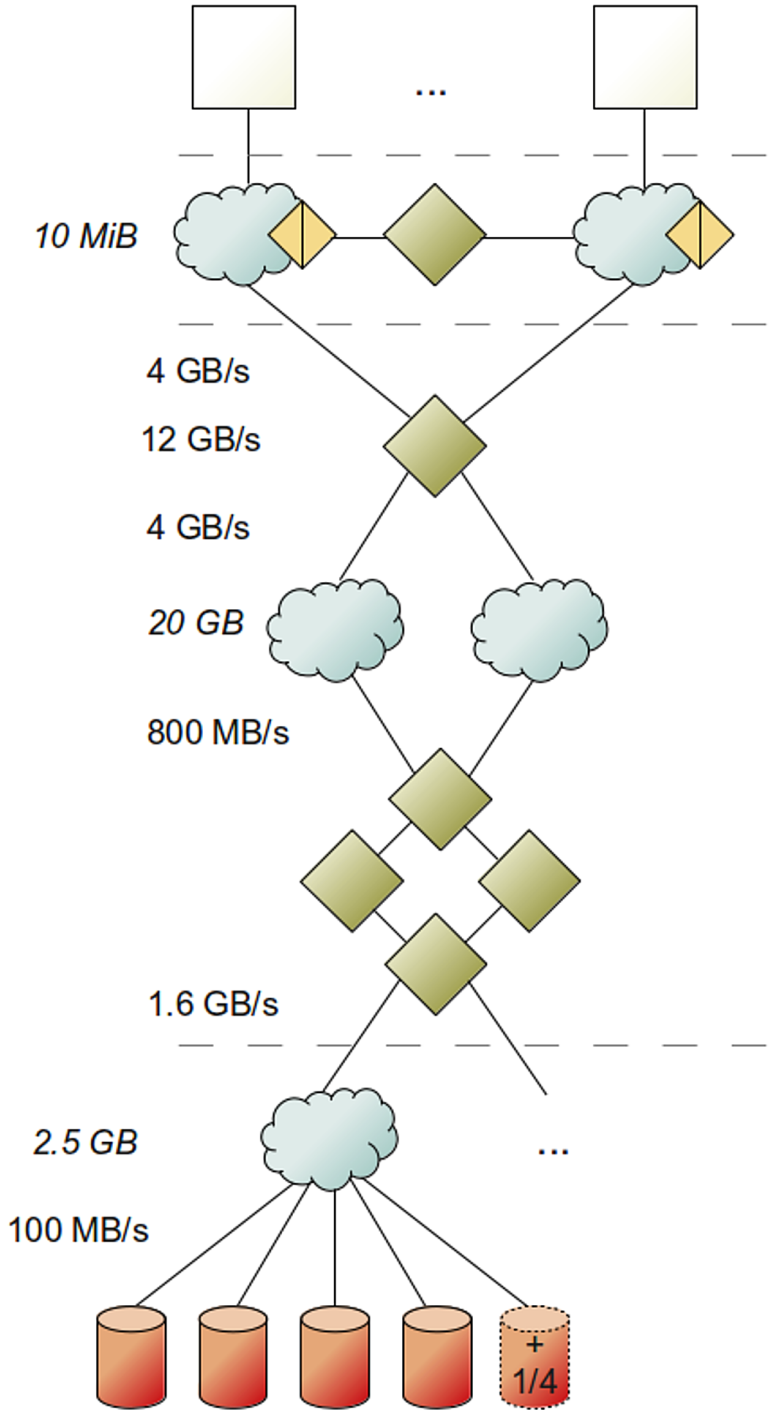


(p) RAID 0+1

Understanding performance limitations

Approach

- Focus on relevant functionality by annotating an IOPm graph
  - Throughput, latency, or (cache) capacity of edges and nodes
- By looking at the graph:
  - Analyze the performance potential and implied bottlenecks
  - Assess observation, determine efficiency of utilized functionality


...
10 MiB
4 GB/s
12 GB/s
4 GB/s
20 GB
800 MB/s
1.6 GB/s
2.5 GB
...
100 MB/s
+
1/4

Understanding performance limitations

Observations

- Sum of data flowing through a node is 0
  - Except for caches + redundant parts
- Sum of data through edges over a horizontal cut is invariant
  - Except for caches + redundant parts
- Throughput allows to determine hit-rate of caches

Summary & Conclusions

- The I/O path is an important characteristics for data access
- IOPm is a systematic description of the logical functionalities
  - Currently: client, cache, translation, network and block devices
- Holistic view: performance can be understood better
  - Bottlenecks can be identified
  - Measured and theoretical performance can be compared
- In *SIOX* we will record activities throughout the I/O path
  - To analyze and optimize access patterns
  - IOPm helps to localize where to deploy monitors
  - High-level interfaces will be offered for logical functionalities
    - We/(Vendors) can implement them for arbitrary file systems