# Directory-Based Metadata Optimizations for Small Files in PVFS

## Euro-Par 2008

Michael Kuhn Julian Kunkel Thomas Ludwig

Parallel and Distributed Systems
Institute for Computer Science
Ruprecht-Karls-Universität Heidelberg

2008-08-28

# Motivation

- There are cases when many small files must be stored in a cluster file system
- For each file additional metadata is stored
  - For example: ownership, permissions and timestamps
- Metadata overhead is significant for small files
- If these files are accessed frequently, metadata performance plays an important role
  - Especially since every metadata operation has to go over the network

# MitoCheck Project

- Genes in cells are knocked out
- The cells' behavior is then monitored
  - Pictures taken by microscopes
- About 22 TB of pictures in 17 million files
- Interpret pictures automatically
- Generate videos from these pictures
  - Read 100 pictures, write 1 video
- Even just listing all pictures is **slow**
  - Several minutes on traditional file systems

# Ideas

- Several approaches can be taken to increase metadata performance
  - Focus on individual file system operations or try to improve the overall scalability
- The changes presented here simply remove all metadata
- For reasons presented later, this effectively makes striping of file data impossible
  - These optimizations are only useful for small files

# Parallel Virtual File System

- PVFS is a parallel cluster file system
  - It supports multiple data and metadata servers
- The whole file system is made up of several objects, each identified by a unique handle
- Each server is responsible for a so-called handle range
  - Each object is managed by exactly one server
- File data is striped across all available data servers
- Metadata of a single file is not distributed
  - The metadata for any file is managed by exactly one metadata server
- Supported by PIOviz
  - It is possible to visualize internal server operations

# Object Types

- PVFS distinguishes several different types of physical objects
  - These can be stored and combined to make up logical objects like files and directories
  - For example, a (logical) file is made up of a (physical) metafile object and multiple (physical) datafile objects
- The servers are just object stores, all the real work is done by the clients

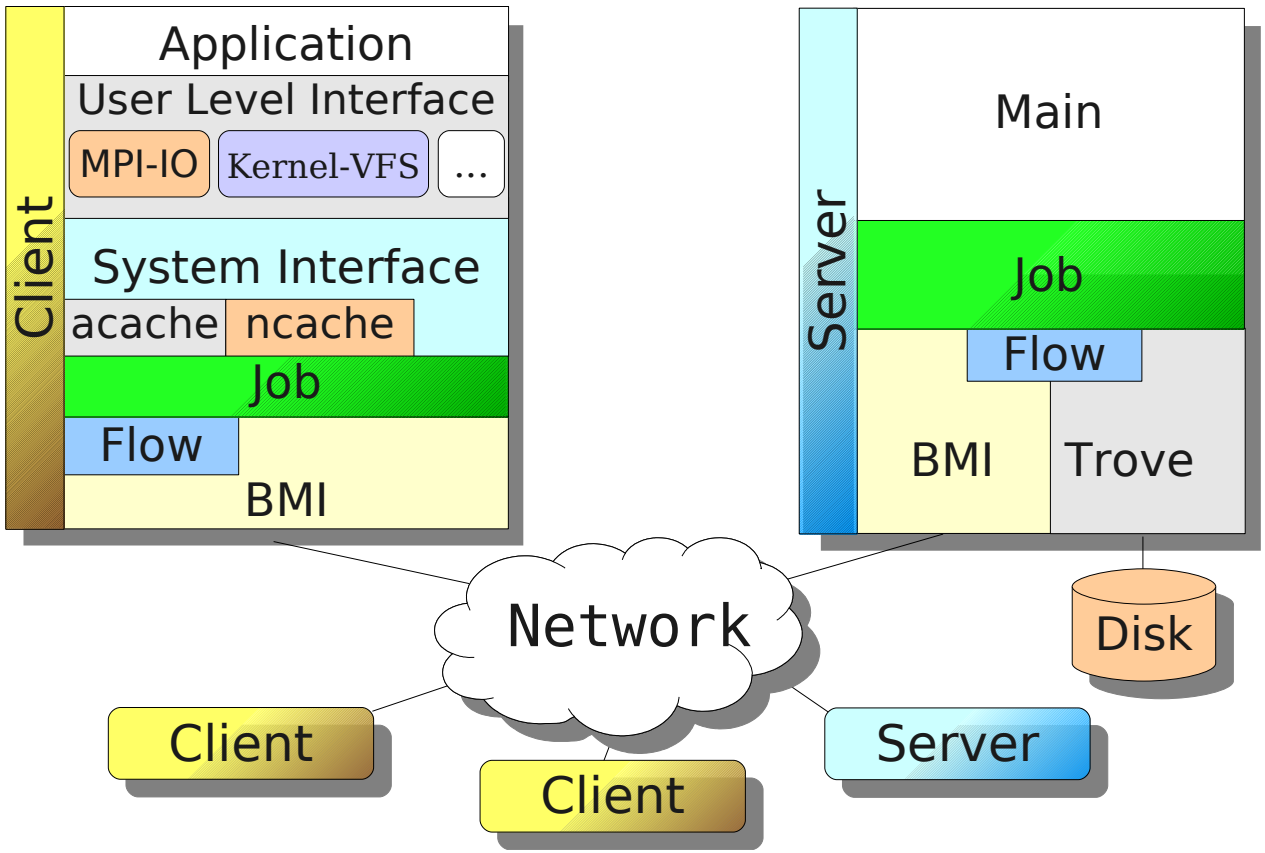
# Object Types

- Metafile objects represent logical files
  - Used to store file metadata like ownership and permissions
  - Also stores all handles of the datafile objects associated with this particular file
- Datafile objects are used to store the actual data of files
  - They are distributed across all data servers
  - Metadata is not stored with each datafile object
- Directory objects represent logical directories
  - They store directory metadata like ownership and permissions
  - So-called directory hints can be set on these directory objects
  - These hints affect all files within the directory

# Directory Hints

- The directory hints are currently mostly used to control the distribution of file data across the data servers
  - The `num_dfiles` hint is simply used to assign the number of datafile objects that should be used for a file
  - Normally one datafile object is created on each data server
- Directory hints can be used to influence other behavior of objects within the directory they are set on
  - This can be used for our optimizations

# PVFS Architecture

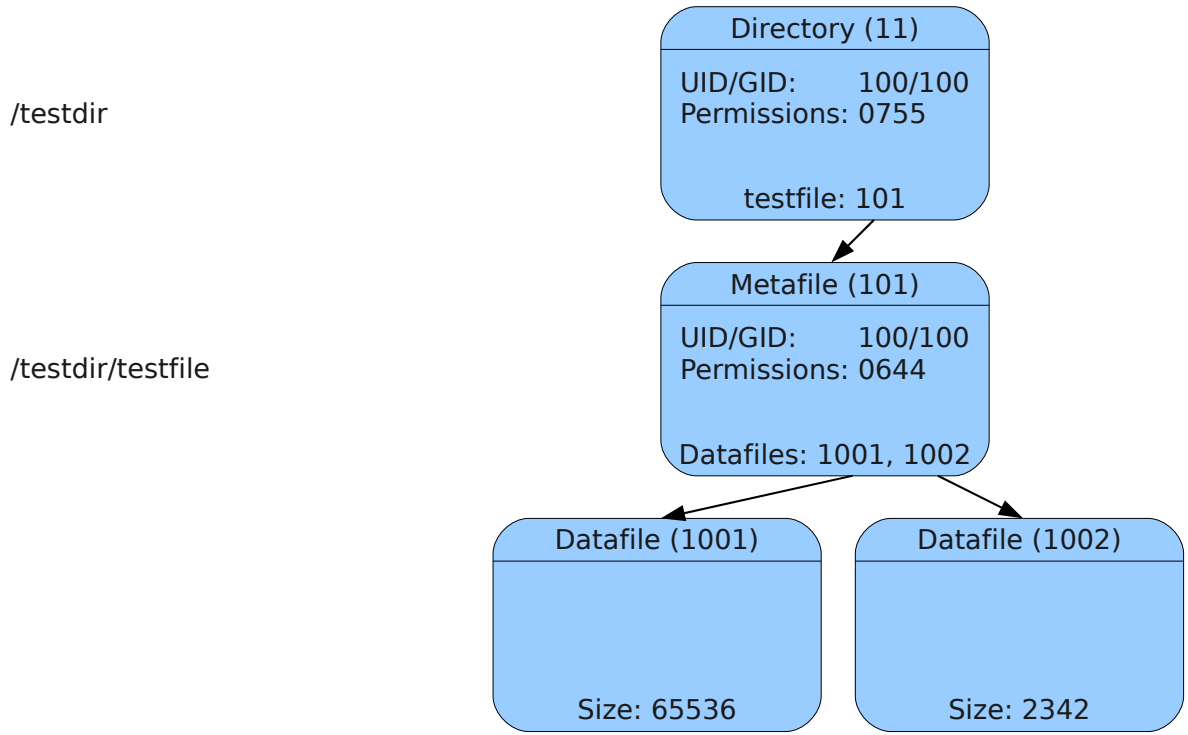



- Most changes in the client's System Interface layer
- All higher layers automatically benefit from the changes

# Overview

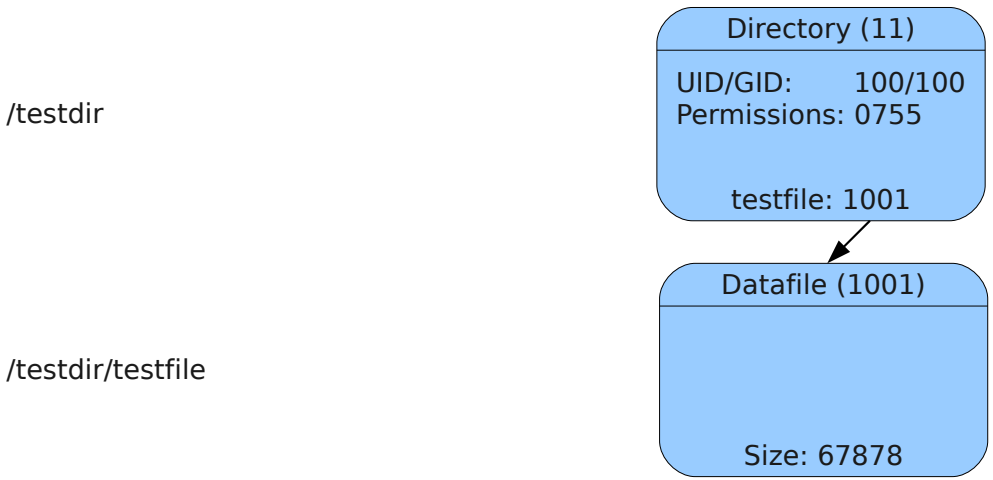
- Metadata optimizations are targeted at small files
  - Striping does not improve performance considerably and can therefore be disabled
  - They can still be used for files of any size, but may degrade performance for larger files
- Introduce new directory hint: `no_metafile`
  - It is possible to turn the metadata optimizations on and off on a per-directory basis
  - Must be enabled explicitly
- Change file system semantics to achieve better performance

# Unmodified PVFS File System




- Metafile objects link together all datafile objects that belong to a particular file
- It is not really necessary to create multiple datafile objects for small files

# Optimized PVFS File System




- The metafile object can be omitted if only one datafile object exists
- Use the datafile object's handle instead of the metafile object's

# Optimization Effects

- Several problems have to be considered
  - The limit of one datafile object per file must be enforced
    - Basically what `num_dfiles` already does
  - The client and server expect a metafile object to be present
  - This metafile object stores all metadata of a file, so this information must be faked in some way
  - No permission checks are possible
    - PVFS clients can send arbitrary credentials anyway
- The following advantages become apparent
  - No metadata server has to be contacted if a file needs to be read or written
  - Only one data server needs to be contacted for each file

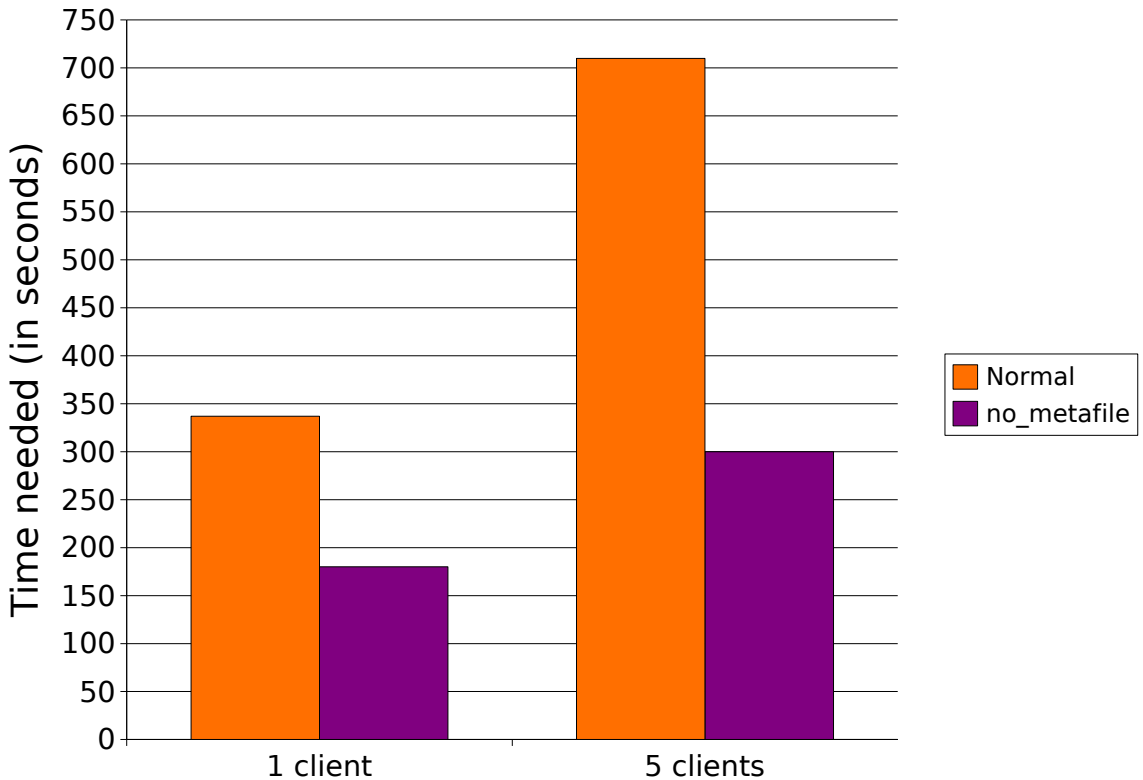
# Environment

- A relatively simple benchmark program was used to measure the benefits of the optimizations
  - The program creates, lists and removes a big number of files in a relatively flat directory hierarchy
  - Number of concurrently accessing clients and underlying storage were varied
    - Storage space on `ext3` and `tmpfs` partitions
  - Simulated moderate load with one client and heavy load with five concurrent clients
- Five machines from our evaluation cluster were used
  - Two machines acted as data servers, another two as metadata servers and a fifth machine was used for the clients
  - Each with two Intel Xeon 2.0 GHz, 1 GByte RAM, an ATA disk and a 1 GBit/s network interface

# File Creation Benchmark

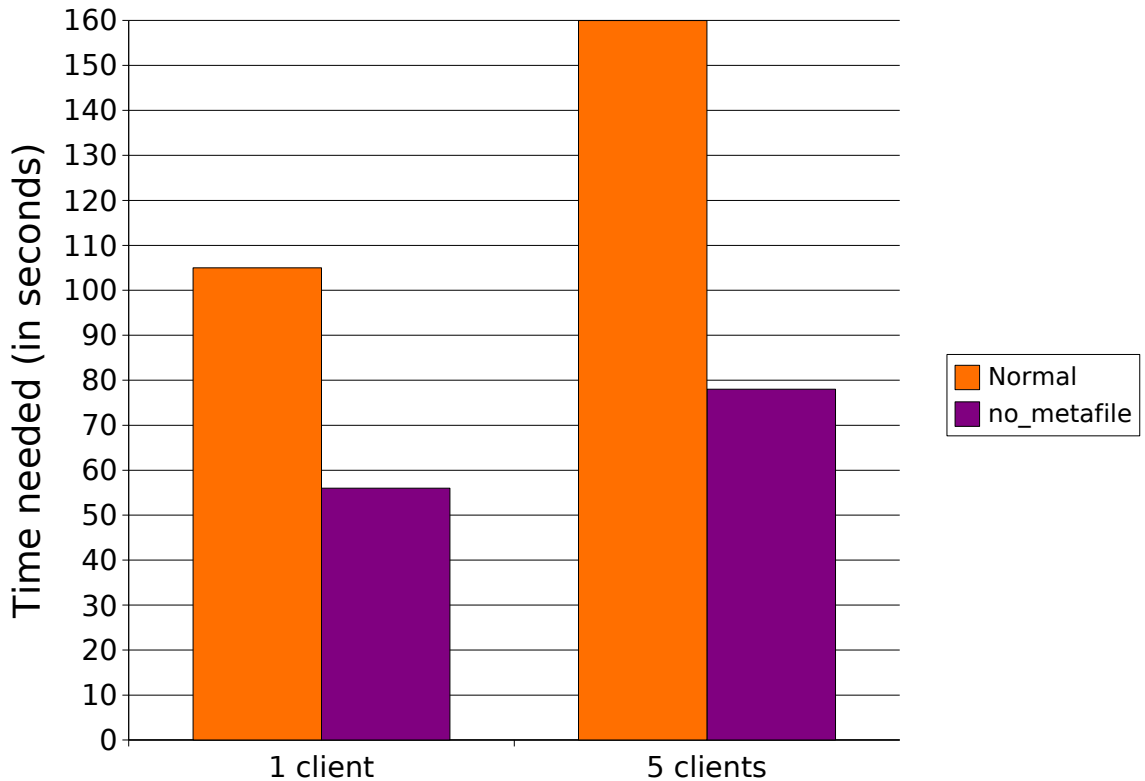
- Each client creates 100 child directories in a single parent directory
- Populates each with 500 files
  - Files of size 0 are created

# File Creation (ext3)




- One client: 147 files/s vs. 286 files/s (195%)
- Five clients: 355 files/s vs. 833 files/s (235%)

# File Creation (tmpfs)

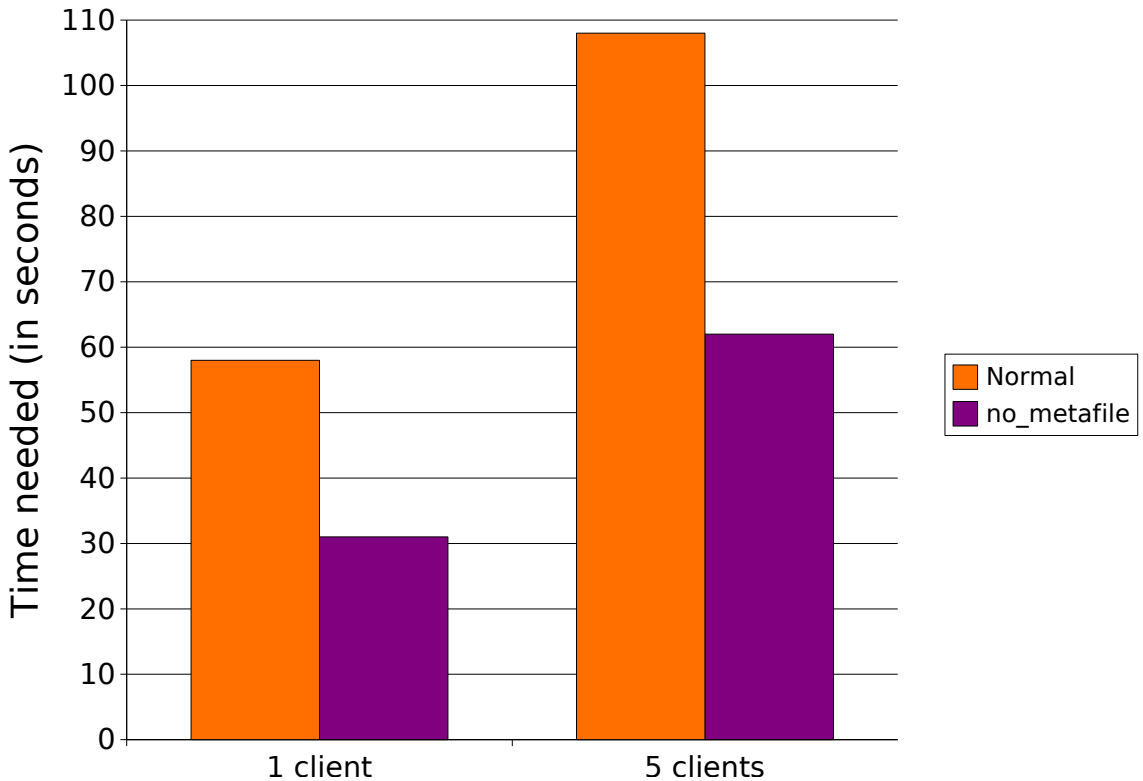



- Speedup with five concurrent clients is less drastic
  - No disk seek times could be avoided

# File Listing Benchmark

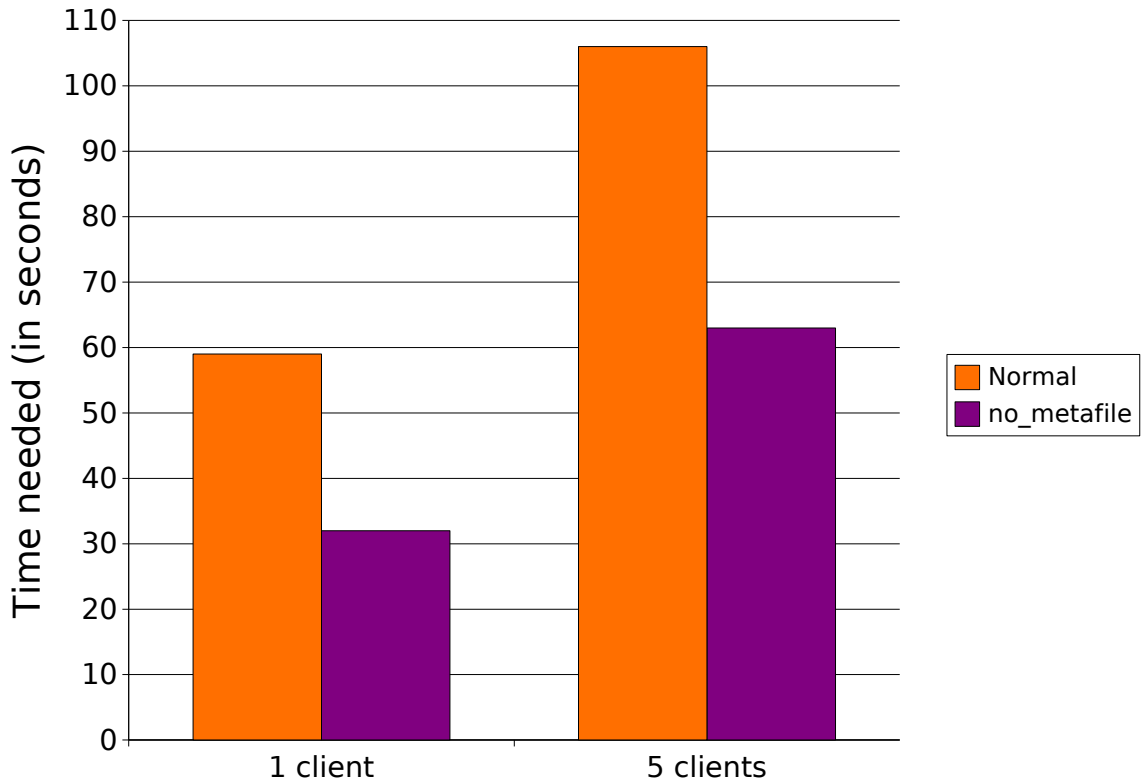
- Each client lists the files in all directories
  - Details like permissions, ownership, etc. are shown
  - Forces the client to contact each datafile object's server
  - Otherwise only the names would need to be fetched from the metadata server

# File Listing (ext3)




- One client: 862 files/s vs. 1613 files/s (187%)
- Five clients: 2315 files/s vs. 4098 files/s (177%)
  - No slow metadata writes to skip

# File Listing (tmpfs)

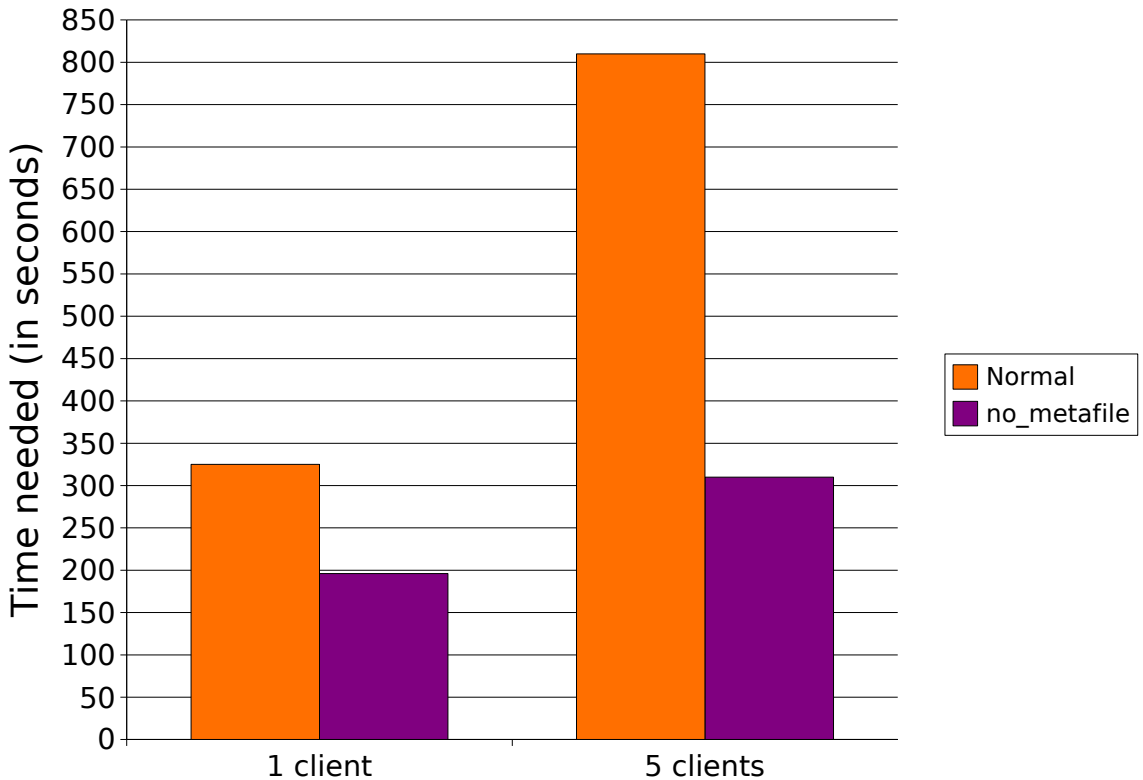



- The times are nearly identical
  - Reads on `ext3` from cache

# File Removal Benchmark

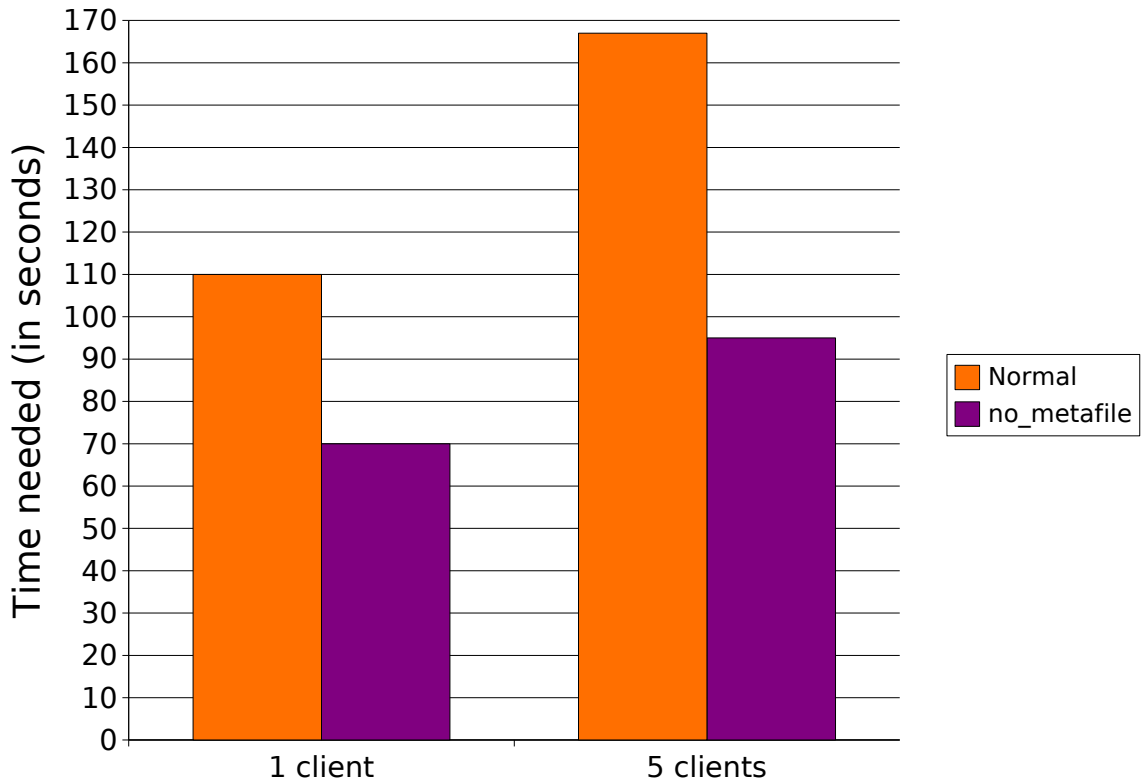
- Each client removes all files and directories

# File Removal (ext3)



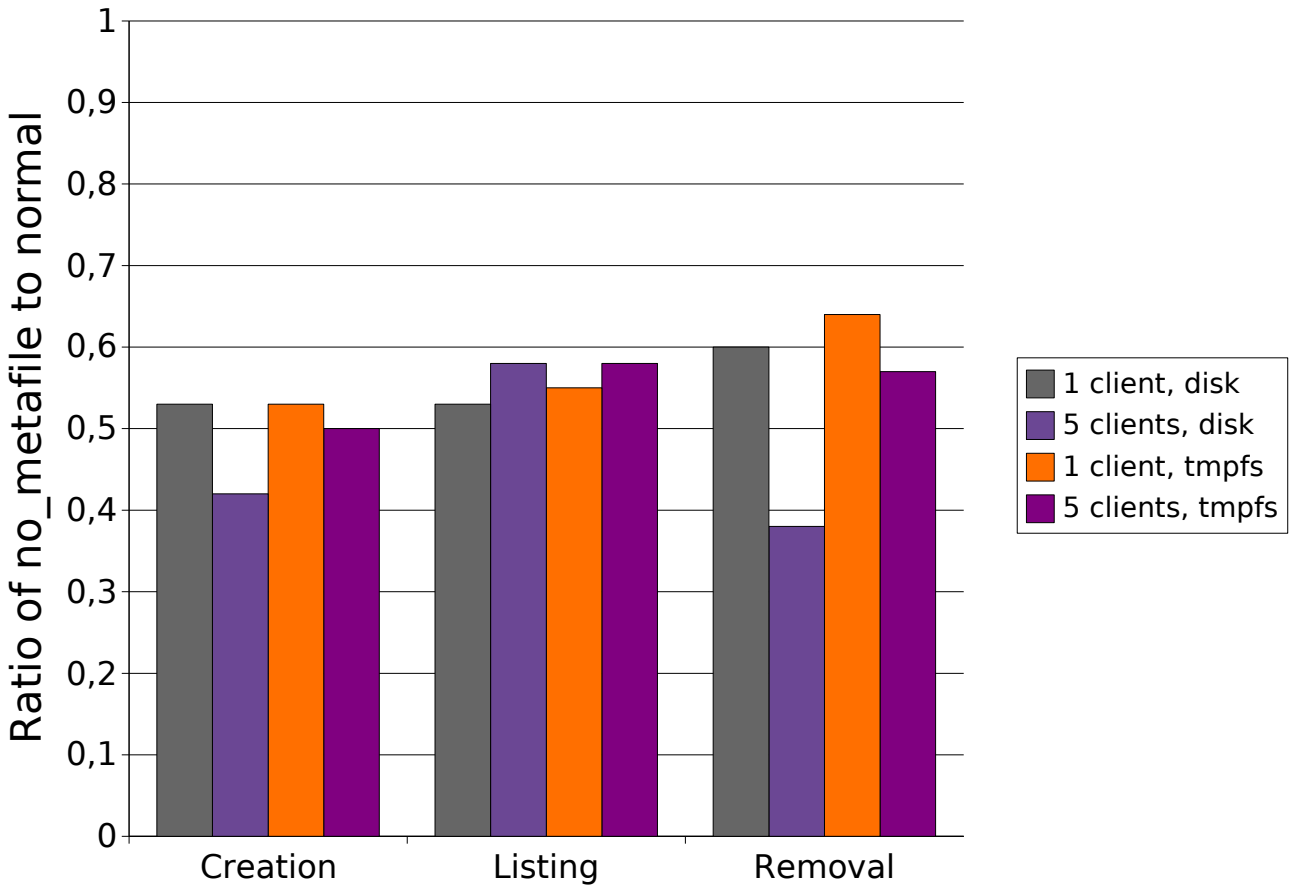

- One client: 154 files/s vs. 256 files/s (167%)
- Five clients: 309 files/s vs. 806 files/s (261%)

# File Removal (tmpfs)




- Speedup with five concurrent clients is less drastic
  - No disk seek times could be avoided

# Overview

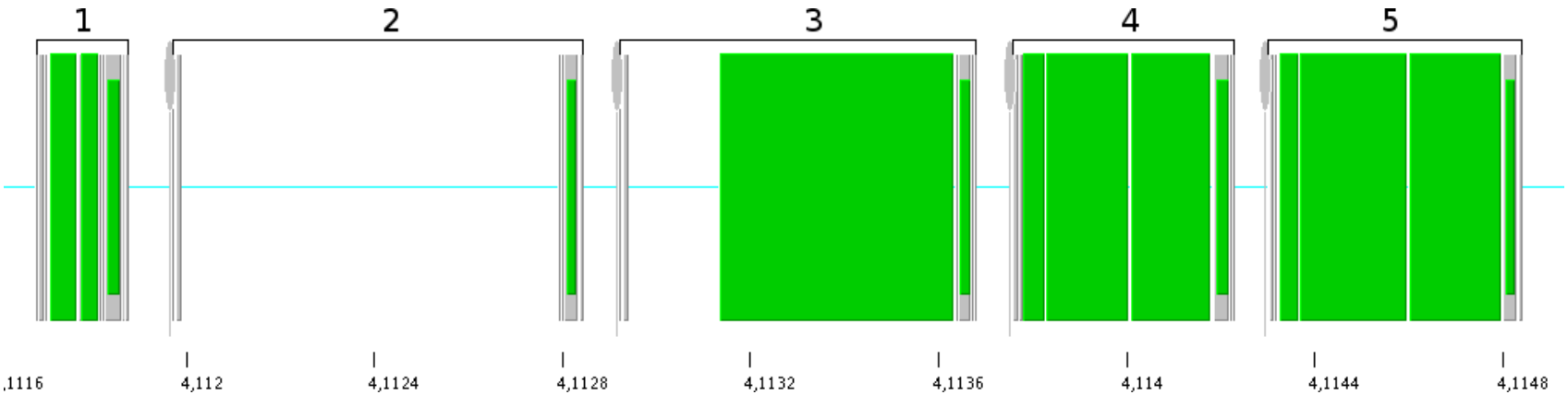



- On average, all operations are about twice as fast

# PIOviz

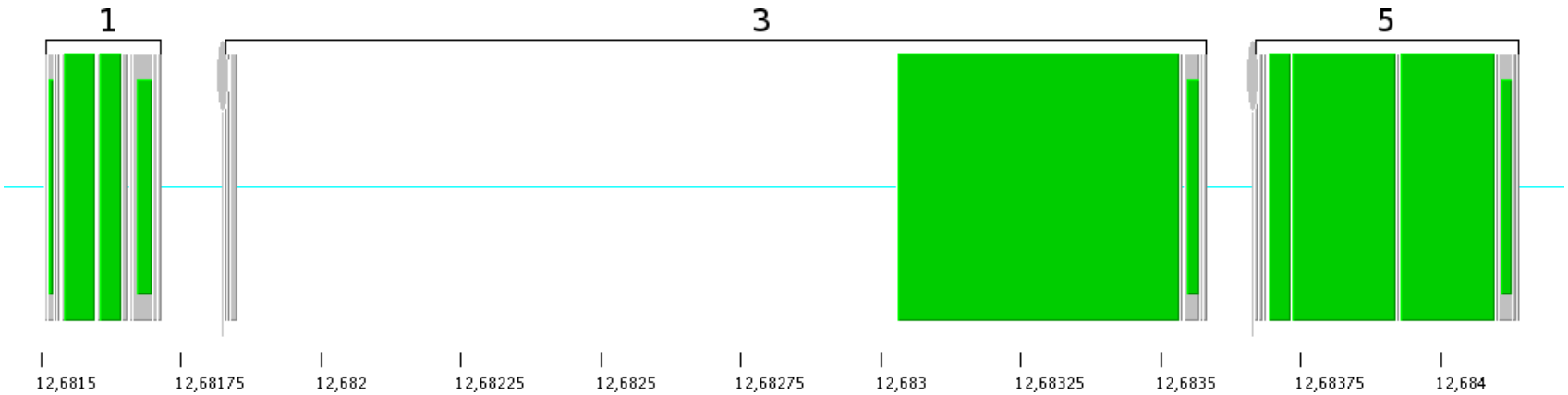
- PIOviz allows PVFS's internal operations to be visualized
  - Consists of MPICH2, (modified) PVFS and some tools
  - MPICH2 already allows tracing of MPI programs
  - PIOviz does the same for PVFS (and more)
  - PVFS and MPI traces can then be merged
- The MPI part was not used at all
- Trace files were visualized with Jumpshot

# File Creation (normal)




1. Read the parent directory's attributes
2. Create the metafile object
3. Create the datafile objects
4. Write the metafile object's attributes
5. Create a directory entry for the new file

# File Creation (no_metafile)




1. Read the parent directory's attributes
3. Create the datafile object
5. Create a directory entry for the new file

# Conclusion

- The performance of some common file system operations could be doubled
- Only a relatively small amount of changes were made
  - 13 files changed, 249 insertions(+), 11 deletions(-)
- Optimizations must be enabled explicitly
  - They do not influence the normal operation of PVFS
- Not yet ready to be used in production environments
  - Implementation is based on a modified development version between versions 2.6.2 and 2.7.0
  - The modified version offers enhanced tracing capabilities used for visualization

# Future Work

- File systems should provide some mechanism to tune file system semantics
  - (Power) Users usually know best what they need
- These optimizations are just a first step in this direction
- The user should have control over as many file system aspects as possible
  - Locking strategies
  - Data safety
  - Consistency
    - File data and metadata
  - ...