Dbfs - Database filesystem [1]

Timo Minartz

Software project WS 2008/09

April 6, 2009

[1] supervised by Julian Kunkel

Inhalt

Project goal

Problem case specific

- map filesystem sources and database tables in one namespace
- implement a lightweight filesystem with FUSE [Sou]
- easy to maintain database design
- minimize database overhead

General

- reusable software
- well documented
- usability

Project goal

Problem case specific

- map filesystem sources and database tables in one namespace
- implement a lightweight filesystem with FUSE [Sou]
- easy to maintain database design
- minimize database overhead

General

- reusable software
- well documented
- usability

Problem case

Initial situation

- a microscope generates lots of data in a specific folder hierarchy
- in particular it creates a tiff-File with a size of a few MByte
- this tiff-File is identicated by a *collaboration*, *project*, *plate*, *replicate*, *well* and *file name*
- there are multiple *collaborations*, *projects*, etc. so lots of tiff-Files are created

Further situation

- tiff-Files should be evaluated by different *applications*
- these *applications* store their results in simple files
- it should be easy to manage these files (i.e. by a database system)

Problem case

Initial situation

- a microscope generates lots of data in a specific folder hierarchy
- in particular it creates a tiff-File with a size of a few MByte
- this tiff-File is identicated by a *collaboration*, *project*, *plate*, *replicate*, *well* and *file name*
- there are multiple *collaborations*, *projects*, etc. so lots of tiff-Files are created

Further situation

- tiff-Files should be evaluated by different *applications*
- these *applications* store their results in simple files
- it should be easy to manage these files (i.e. by a database system)

Problem case (2)

Initial filestructure (base filesystem)

/collaboration/project/plate/replicate/well-file.tiff

Resulting filestructure (fuse filesystem, dbfs)

/collaboration/project/**application**/plate/replicate/**well**/**file.tiff**

Problem case (2)

Initial filestructure (base filesystem)

/collaboration/project/plate/replicate/well-file.tiff

Resulting filestructure (fuse filesystem, dbfs)

/collaboration/project/**application**/plate/replicate/**well**/**file.tiff**

Example

Base filesystem structure

/collab0/project0/plate0/replicate0/000-file1.tiff
/collab0/project0/plate0/replicate0/000-file2.tiff
/collab0/project0/plate0/replicate0/001-file3.tiff
/collab0/project0/plate0/replicate0/metadata

Dbfs filestructure

/collab0/project0/**application0**/plate0/replicate0/**000**/file1.tiff
/collab0/project0/**application0**/plate0/replicate0/**000**/file2.tiff
/collab0/project0/**application0**/plate0/replicate0/**001**/file3.tiff
/collab0/project0/**application0**/plate0/replicate0/metadata

Example

Base filesystem structure

/collab0/project0/plate0/replicate0/000-file1.tiff
/collab0/project0/plate0/replicate0/000-file2.tiff
/collab0/project0/plate0/replicate0/001-file3.tiff
/collab0/project0/plate0/replicate0/metadata

Dbfs filestructure

/collab0/project0/**application0**/plate0/replicate0/**000**/file1.tiff
/collab0/project0/**application0**/plate0/replicate0/**000**/file2.tiff
/collab0/project0/**application0**/plate0/replicate0/**001**/file3.tiff
/collab0/project0/**application0**/plate0/replicate0/metadata

Virtual files examples

Dbfs filesystem

/collaboration0/project0/application0/plate0/replicate0/000/**ergs**
/collaboration0/project0/application0/plate0/replicate0/001/**ergs**

- virtual files are stored in database
- virtual files are identificated by *collaboration*, *project*, *plate*, *replicate*, *well*, *file name* **AND** *application*

Virtual files examples

Dbfs filesystem

/collaboration0/project0/application0/plate0/replicate0/000/**ergs**
/collaboration0/project0/application0/plate0/replicate0/001/**ergs**

- virtual files are stored in database
- virtual files are identificated by *collaboration*, *project*, *plate*, *replicate*, *well*, *file name* **AND** *application*

Further constraints

Virtualization layers

- one for the *application* and
- one for the *well*

Permissions

- only read permission to tiff-Files
- permissions for metadata files inherited from base filesystem
- read and write permissions to virtual files on application level
- no structural changes allowed (chmod,mkdir,. . .)

Further constraints

Virtualization layers

- one for the *application* and
- one for the *well*

Permissions

- only read permission to tiff-Files
- permissions for metadata files inherited from base filesystem
- read and write permissions to virtual files on application level
- no structural changes allowed (chmod,mkdir,. . .)

Virtual files model

- table for every *application*
- table has columns for every subfolder and one for every virtual file

Table: Example database table `collaboration0_project0_application0`

plate	replicate	well	ergs
plate0	replicate0	000	"ergs for well 000"
plate0	replicate0	001	"ergs for well 001"

Virtual files model

- table for every *application*
- table has columns for every subfolder and one for every virtual file

Table: Example database table `collaboration0_project0_application0`

plate	replicate	well	ergs
plate0	replicate0	000	"ergs for well 000"
plate0	replicate0	001	"ergs for well 001"

Permissions model

- permissions on *project* level
- second table for permissions
- containing one column for *application* and one for the owner (user id from operating system)

Table: Example permission table `permissions_collaboration0_project0`

name	owner
application0	1000
application1	1001

Permissions model

- permissions on *project* level
- second table for permissions
- containing one column for *application* and one for the owner (user id from operating system)

Table: Example permission table `permissions_collaboration0_project0`

name	**owner**
application0	1000
application1	1001

Managing the directory structure

General

- changes in the base filesystem
- and in the database tables (i.e. new virtual files)

Howto

- "by hand", see documentation and/or README file
- using a simple GUI

Managing the directory structure

General

- changes in the base filesystem
- and in the database tables (i.e. new virtual files)

Howto

- "by hand", see documentation and/or README file
- using a simple GUI

Managing the directory structure (2)

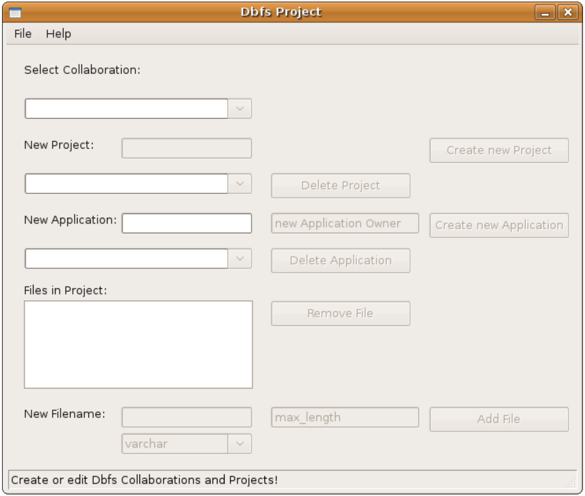

Figure: Graphical user interface to manage the directory structure

Optimizations and restrictions

Database overhead

- multiple users who need own database connections
- lots of queries are generated for a simple command (like ls)

Optimization

- thread-safe database pooling
- simple caching for query results
- both can be enabled in the sourcecode

Restrictions

- cache consistency problem
- if underlying base filesystem changes (creating new (sub-)folders etc.)

Optimizations and restrictions

Database overhead

- multiple users who need own database connections
- lots of queries are generated for a simple command (like ls)

Optimization

- thread-safe database pooling
- simple caching for query results
- both can be enabled in the sourcecode

Restrictions

- cache consistency problem
- if underlying base filesystem changes (creating new (sub-)folders etc.)

Optimizations and restrictions

Database overhead

- multiple users who need own database connections
- lots of queries are generated for a simple command (like ls)

Optimization

- thread-safe database pooling
- simple caching for query results
- both can be enabled in the sourcecode

Restrictions

- cache consistency problem
- if underlying base filesystem changes (creating new (sub-)folders etc.)

Implementation in C++

Implemented classes can be spread into 4 modules

- handling filesystem issues
- database access
- GUI and
- the helper classes and functions

Implemented filesystem operations

- gettattr
- readdir
- read and
- write

Implementation in C++

Implemented classes can be spread into 4 modules

- handling filesystem issues
- database access
- GUI and
- the helper classes and functions

Implemented filesystem operations

- gettattr
- readdir
- read and
- write

Implementation in C++ (2)

Further implementation details

- documentation (PDF)
- in-line documentation (doxygen)
- type make doc in software project root

FUSE stumbling blocks

Mounting fuse without administrative privileges

- mount: ./dbfs mountpoint [args]
- umount: fusermount -u mountpoint

Logging

- fuse forks a new process, so logging to stdout is not possible
- the parameter -f prevents fuse from forking
- alternative: logging to a file (implemented)

Debugging with valgrind

- problem with older kernel versions: fusermount not traceable
- workaround available: see README in project root
- with kernel 2.6.27-11-generic working out-of-the-box

FUSE stumbling blocks

Mounting fuse without administrative privileges

- mount: ./dbfs mountpoint [args]
- umount: fusermount -u mountpoint

Logging

- fuse forks a new process, so logging to stdout is not possible
- the parameter -f prevents fuse from forking
- alternative: logging to a file (implemented)

Debugging with valgrind

- problem with older kernel versions: fusermount not traceable
- workaround available: see README in project root
- with kernel 2.6.27-11-generic working out-of-the-box

FUSE stumbling blocks

Mounting fuse without administrative privileges

- mount: ./dbfs mountpoint [args]
- umount: fusermount -u mountpoint

Logging

- fuse forks a new process, so logging to stdout is not possible
- the parameter -f prevents fuse from forking
- alternative: logging to a file (implemented)

Debugging with valgrind

- problem with older kernel versions: fusermount not traceable
- workaround available: see README in project root
- with kernel 2.6.27-11-generic working out-of-the-box

The benchmark process

Testsets

- comparision of Dbfs and tmpfs
- evaluation of Dbfs
  - tmpfs as base filesystem
  - ext3 / tmpfs filesystem for the mysql database
  - clean / dirty database

Different use cases

- reading filesystem attributes
- reading metadata files and tiff-Files
- reading virtual files
- writing virtual files

The benchmark process

Testsets

- comparision of Dbfs and tmpfs
- evaluation of Dbfs
    - tmpfs as base filesystem
    - ext3 / tmpfs filesystem for the mysql database
    - clean / dirty database

Different use cases

- reading filesystem attributes
- reading metadata files and tiff-Files
- reading virtual files
- writing virtual files

Metadata

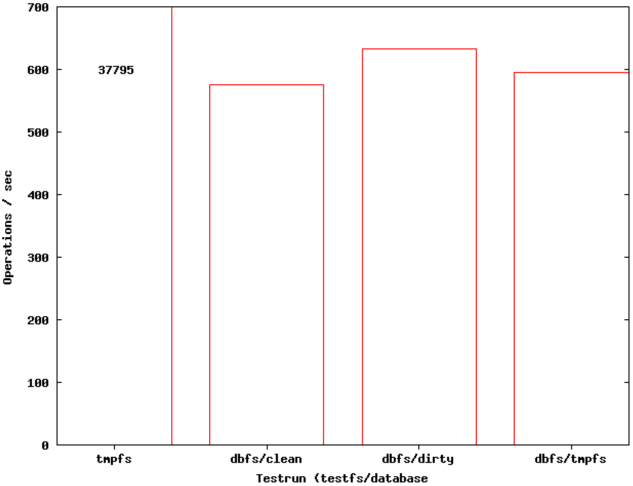

Figure: Reading filesystem attributes

Physical files

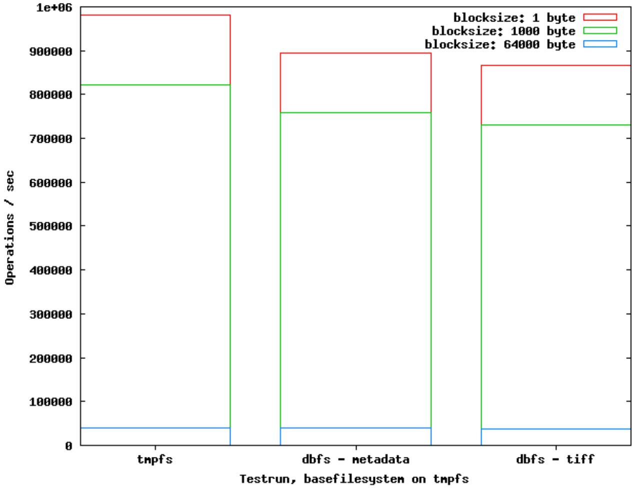

Figure: Read test for the physical files depending on blocksize

Physical files (2)

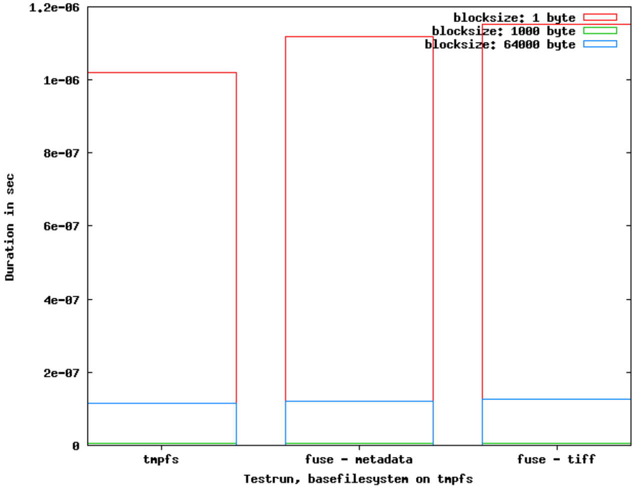

Figure: Read test for the physical files, time for reading one byte

Physical files (3)

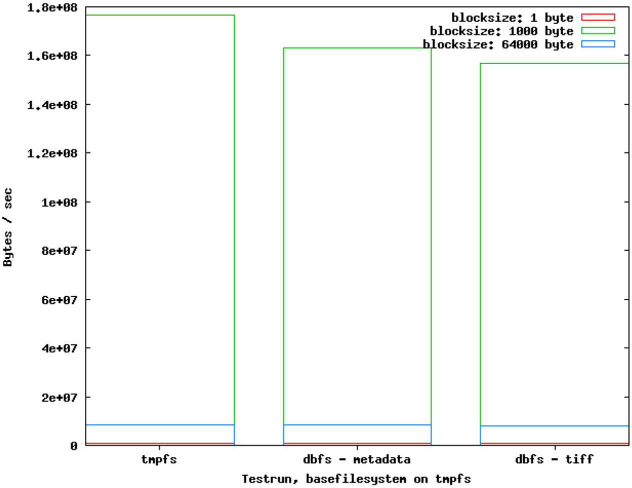

Figure: Read test for the physical files, bytes per sec

Virtual files

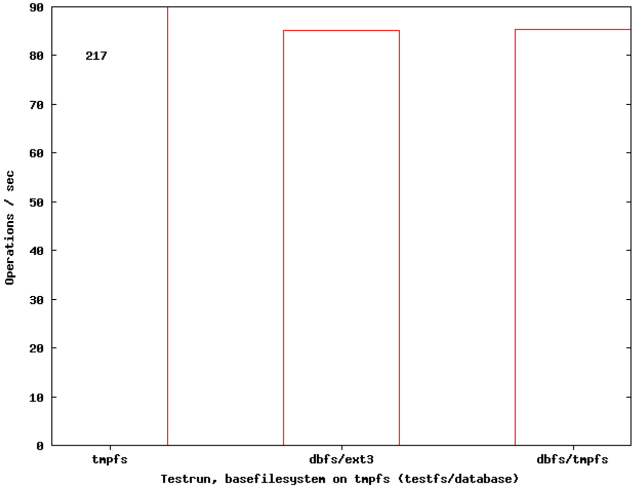

Figure: Read test for virtual files

Virtual files (2)

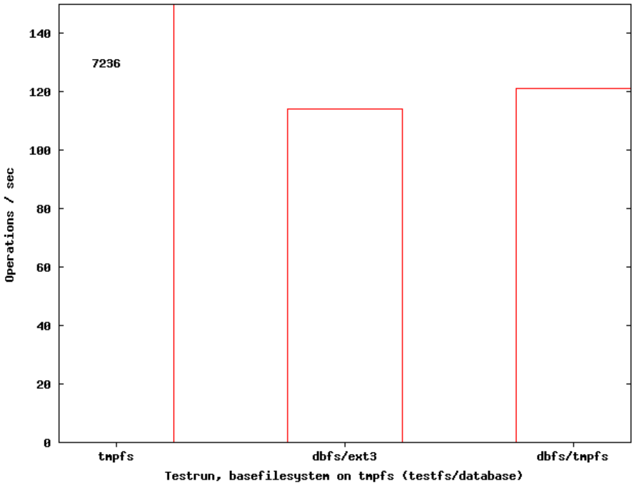

Figure: Write test for virtual files

Virtual files (3)

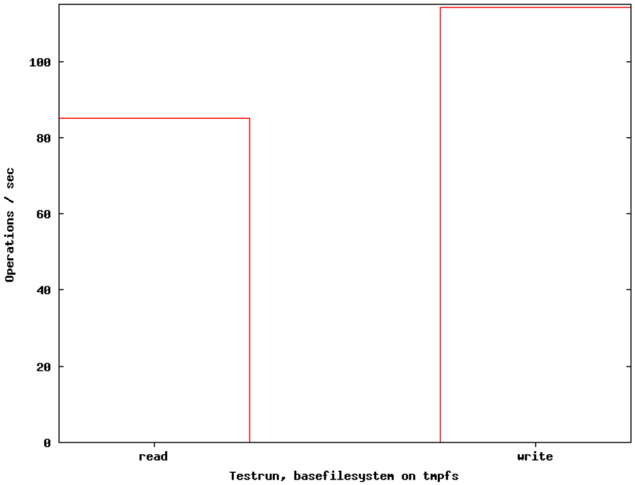

Figure: Read and write for virtual files

Conclusion

Software project goal

- mapping filesystem and database sources in one namespace can be solved by a fuse implementation
- good performance for physical files (stored on underlying filesystem)
- bottleneck for virtual files is not the database access itself
- concrete use case must take decision about using this implementation

Future work

- implementation issues (sql injection, dynamic virtualization layers)

Conclusion

Software project goal

- mapping filesystem and database sources in one namespace can be solved by a fuse implementation
- good performance for physical files (stored on underlying filesystem)
- bottleneck for virtual files is not the database access itself
- concrete use case must take decision about using this implementation

Future work

- implementation issues (sql injection, dynamic virtualization layers)

- *ROFS, the Read-Only Filesystem for FUSE.*
  http://mattwork.potsdam.edu/projects/wiki/index.php/Rofs
- IEEE, The ; GROUP, The O.:
  The Open Group Base Specifications Issue 6.
  http://www.opengroup.org/onlinepubs/009695399/functions/contents.html
- MICROSYSTEMS, Sun:
  MySQL 6.0 Reference Manual.
  http://dev.mysql.com/doc/refman/6.0/en/index.html
- SOURCEFORGE.NET:
  Main Page - fuse.
  http://apps.sourceforge.net/mediawiki/fuse/index.php?title=Main_Page