Cookfs

Wojciech Kocjan, 2010-10-13

Open Source Made Easy.

- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

Introduction

- Tcl one of first languages to ship applications as single executable
  - Prowrap – first Tcl solution for putting multiple files in single executable; uses ZIP based archives
  - freeWrap – one of first freely available solutions; uses ZIP
  - TclKit – one of most popular solutions for bundling applications; uses Mk4-based VFS and uses Tcl 8.4 VFS API

Introduction

- Why cookfs?
    - Optimized for Tcl code
        - Groups small files
        - Optimized for Tcl files such as pkgIndex.tcl
    - Multiple compressions
        - Currently zlib, bz2 – plans for including LZMA as option
        - Can use multiple compressions in one archive
    - Designed for shipping all types of files
        - Handles large number of small files and very large files
        - Provides efficient memory management – VFS operations consume very little memory

Introduction

- Using cookfs
  - Obtaining and building cookfs
    - Available from SourceForge - source code on SVN; cookfs source and cookit binaries as file downloads
    - Using TEA; uses standard configure & make approach
  - Using cookfs from Tcl
    - Similar to any VFS – `vfs::cookfs::Mount`, `vfs::unmount`
    - Commands (i.e. `file`) work same as for any VFS
    - Direct write command for faster addition of files

- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

Overview and design

- Cookfs major elements
  - Cookfs pages
    - Storage of parts of files, entire files or group of files
    - Referenced by integer indexes, starting from 0
  - Index and directory hierarchy
    - Keeps information on files and directories in VFS
    - Maps hierarchy to cookfs pages
  - Cookfs VFS layer
    - Uses pages and index elements to provide VFS
    - Offers access to archive from Tcl and Tcl C API

Overview and design – pages

- Generic solution for storing content of files
  - Each page can store arbitrary amount of bytes
  - Pages are referenced by integer indexes, starting at 0
- Pages can only be added and read
  - Pages are immutable – update or delete not allowed
  - Page reading uses LRU cache; this speeds up retrieval of small files
  - Page cache is configurable – faster reads at a cost of larger memory use

Overview and design – pages

- Metadata keeps track of page sizes
  - Keeps information on all pages' sizes
  - Provides space for storing cookfs indexes

- Offer write-aside feature
  - Storing changes to cookfs archive in a separate file
  - Useful for read-only media (CD, DVD) or for storing updates to application in a separate file

  - Currently written in C; work on pure Tcl implementation currently in progress

Overview and design – index

- Keeps structure of an archive
  - Index stores a tree of all files and directories
  - Contains information such as mtime for all entries
  - All information read when VFS is mounted and kept entirely in memory

- Prevents illegal operations
  - Not possible to create a file/directory as child of a file
  - Operation such as changing a file to a directory are also blocked by index

Overview and design – index

- Stores mapping of pages to file contents
  - Information on each file has references to pages
  - Keeps list of pages – page number, offset in page and data size
  - Multiple files can reuse same page(s) if their contents is the same – automatically detected by VFS layer

  - Currently written in C; work on pure Tcl implementation currently in progress

Overview and design – VFS layer

- Uses pages and index to provide complete VFS
  - Provides handlers for VFS operations
  - Cookfs index used for file information operations such asl isting files, getting and setting file metadata

- Provides channels for reading mechanism
  - Handles seek and read operations – data retrieved directly from pages, not read entirely to memory
  - Uses `chan create` or `rechan` for channel creation

Overview and design – VFS layer

- Uses memchan for writing to cookfs archive
  - Initiated as empty or containing previous content of a file, depending on how file was opened

- When memchan is closed closed, its contents is added to archive or queued for addition
  - Files above specified size are added right away
  - Small files added in batches and grouped by file names

- VFS layer is currently implemented in Tcl; uses `tclvfs` package for providing Tcl virtual filesystem

- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

Comparison with existing technologies

- Mk4 VFS
  - Both Mk4 VFS and cookfs use zlib compression
    - Mk4 VFS compresses each file individually
    - Cooks compresses smaller files in groups
  - Cookfs offers multiple compression algorithms
    - Currently bzip2 also available
    - LZMA support planned for future releases

Comparison with existing technologies

- Mk4 VFS – continued
  - Mk4 VFS and cookfs implemented in C+Tcl
    - Mk4 VFS uses Mk4tcl for underlying storage
    - Mk4tcl written in C++ - requires additional libraries (libstdc++)
    - Cookfs uses C code for storage; pure Tcl version work in progress – C version will provide better performance
    - Both technologies depend on tclvfs

Comparison with existing technologies

- ZIP VFS
  - Both ZIP VFS and cookfs use zlib compression
    - ZIP VFS compresses each file individually
    - Cooks compresses smaller files in groups
    - Cookfs provides other compressiona algorithms
  - ZIP VFS uses a known standard for archive
    - Multiple tools for managing ZIP archives available
    - Can be created from various tools such as `ant`

Comparison with existing technologies

- Other related projects and/or alternatives
  - Trofs
    - Pure C solution for single file archives; designed for Tcl modules in mind
    - Does not use compression, simply concatenates file contents and metadata on stored files
    - Does not depend on tclvfs
  - Tar VFS
    - Uses tar archive format for storing files
    - Does not use compression by default

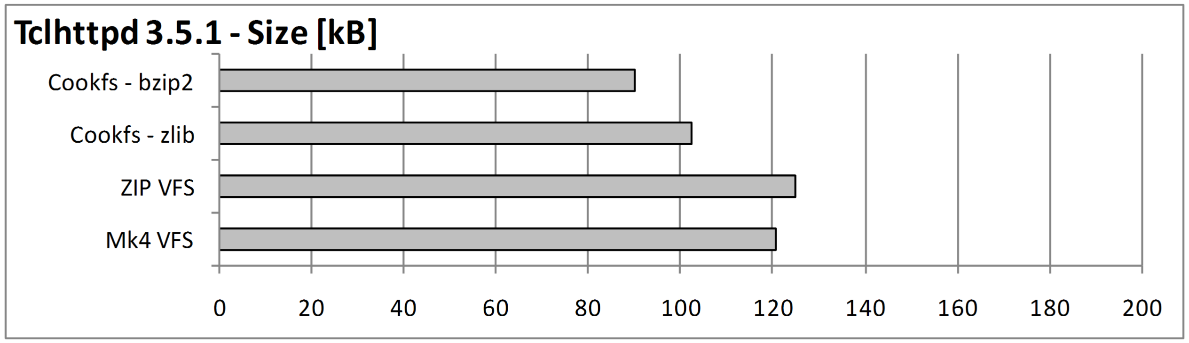
- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

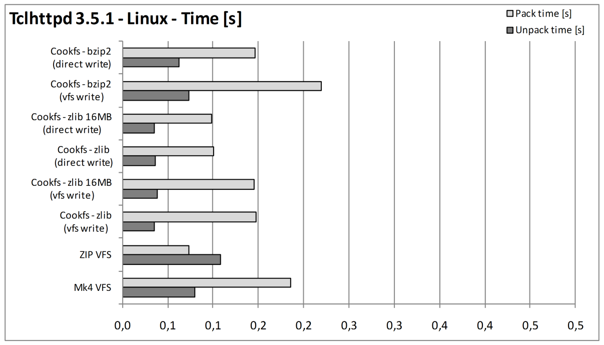
BITROCK

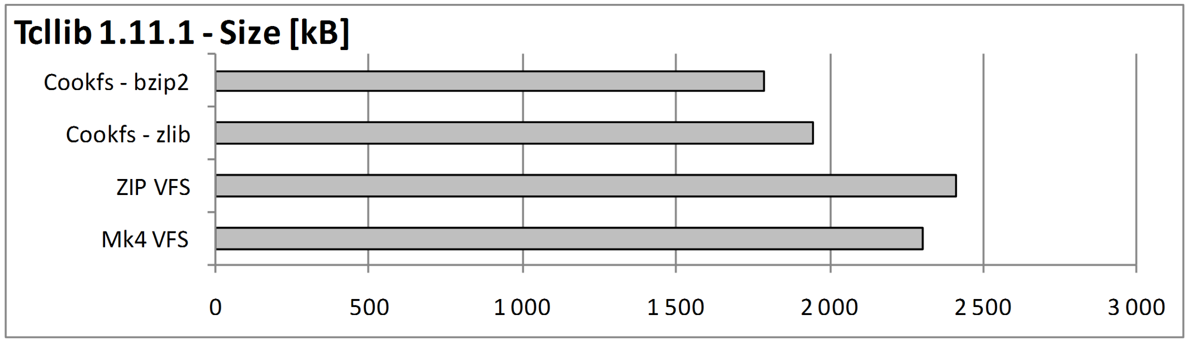
Comparison – sample archive contents

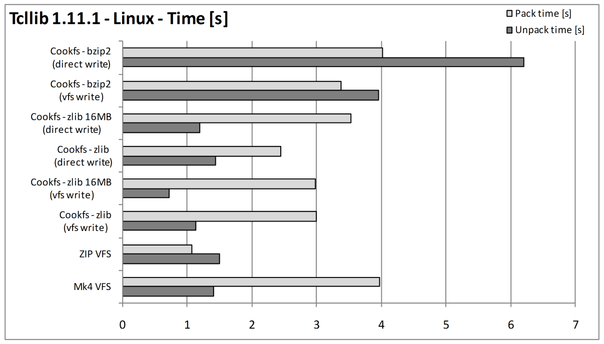
- Tcl related samples
  - Tclhttpd 3.5.1 – embedded web server
    - relatively small package, often used in Tcl applications
    - uncompressed size: 460kB
  - Tcllib 1.11.1 – set of commonly used Tcl packages
    - uncompressed size: 12MB
  - ActiveTcl 8.4 – sample of Tcl binaries and packages
    - Installation of ActiveTcl 8.4.19.0 for Linux
    - uncompressed size: 71MB

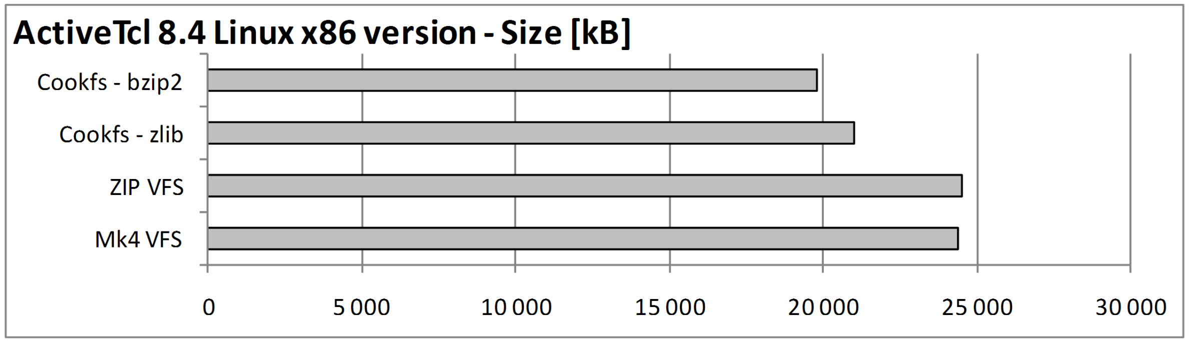
Comparison – sample archive contents

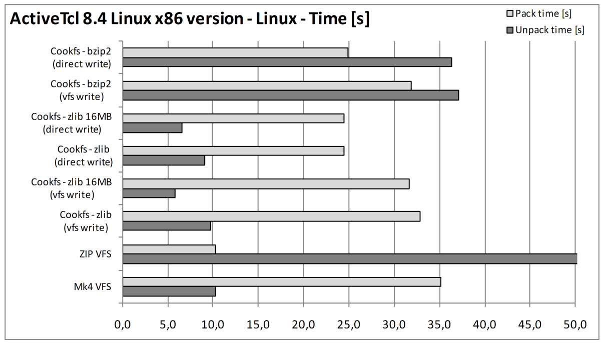








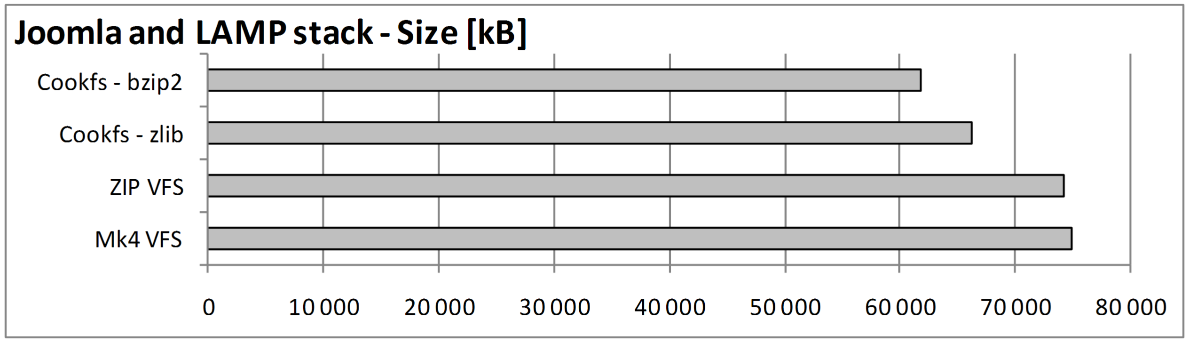


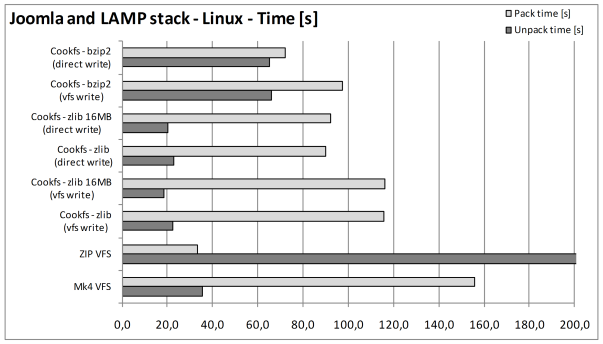


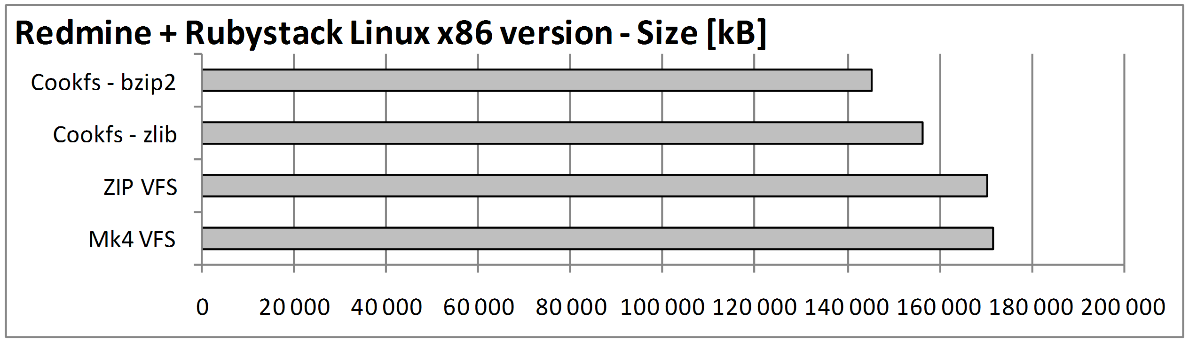
Comparison – sample archive contents

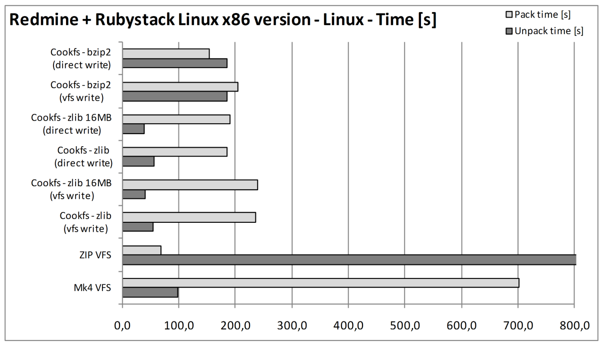
- Non-tcl samples
  - Packaging Joomla and LAMP – BitNami Joomla stack
    - example of packaging non-Tcl content into an archive
    - Joomla stack for Linux with Apache, MySQL and PHP
    - uncompressed size: 224MB
  - Packaging Redmine – BitNami Redmine stack
    - Redmine application with Ruby and all other artifacts
    - uncompressed size: 535MB

Comparison – sample archive contents









BitRock

- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

Cookit – standalone Tcl/Tk binary

- Cookit is a tclkit-like binary that has Tcl, core libraries and VFS included in a single file
  - Includes Tcl (optionally Tk) and cookfs linked static

- Can be used in same ways as tclkit:
  - For running other scripts – Tcl scripts, cookfs and zip archives
  - For building standalone applications – by adding files to cookfs archive, including `main.tcl`
  - As interactive `tclsh` / `wish` shell in a single file

Cookit – standalone Tcl/Tk binary

- Cookit and Tcl initialization
  - Uses `Tcl_Main()` and `Cookit_AppInit()` as custom application initialization
    - Reads initialization Tcl code from cookfs
    - Initializes tclvfs and Tcl libraries after mounting cookfs
  - If `main.tcl` file is present in cookfs archive, cookit sources it
    - Works as standalone application
    - Other packages can be placed in `lib/` directory; they can be loaded using `package require` commands

Cookit – standalone Tcl/Tk binary

- Cookit build system
  - Engine for building packages and linking cookit
    - Has logic to handle dependencies, comparing versions
    - Multiple commands to build entire cookit or just parts
  - Has definitions for cookit parts – i.e. tcl, tk, vfs, cookfs
    - Each part defines how it is configured and built
    - Handles listing files to add to cookit VFS
    - Create additional scripts for cookit initialization
    - Handle adding static packages to `Cookit_AppInit()`

Cookit – standalone Tcl/Tk binary

- Building cookit
  - Download cookit build system from SourceForge: http://sourceforge.net/projects/cookit/files/cookit/
  - Retrieve Tcl, Tk, tclvfs and cookfs sources:
    ```
    tclsh build.tcl retrievesource tcl tk vfs cookfs
    ```
  - Building default cookit (without Tk)
    ```
    tclsh build.tcl build-cookit
    ```
  - Build with Tk embedded statically (i.e. for MS Windows)
    ```
    tclsh build.tcl –tk latest build-cookit
    ```

Cookit – standalone Tcl/Tk binary

- Using cookit
  - Run `cookit`, `cookit.exe` or `cookit-ui.exe`
    - `cookit.exe` runs in command prompt
    - `cookit-ui.exe` includes Tk and only provides UI mode

- Platforms currently built
  - Windows x86
  - Linux x86
  - Mac OS X x86
  - More platforms can be built from sources …

- Introduction
- Overview and design
- Comparison with existing technologies
- Comparison – sample archive contents
- Cookit – standalone Tcl/Tk binary
- Current status and plans

Current status and plans – cookfs

- Archive format stabilized – will not change

- Implementation partially C and Tcl
- Depends on tclvfs to provide VFS layer

- Future plans
- C-only implementation – remove dependencies
- Create pure Tcl version for easier adoption
- Improve grouping and duplication detection – currently not detected well for small files

Current status and plans – cookit

- Current status
  - Modularized build system, support for multiple platforms (tested on 4 platforms, regularly built on 3)
  - Binaries work fine – no major issues
  - Support for threaded Tcl not yet complete
- Future plans
  - Wider platforms builds to be performed periodically
  - Platform for building and managing packages

Current status and plans

- Want to help?
  - Cookfs
    - Create documentation
    - Better test suite and/or add more test coverage
    - Help with implementing new features
  - Cookit
    - Platform support – build and submit binaries
    - Submit bugs/problems on less common platforms

Acknowledgements

Acknowledgements

- Acknowledgements
  - BitRock, especially Daniel Lopez and Juan José Medina Godoy – for providing feedback to cookfs and paper for this conference
  - Piotr Beltowski – for reviews and comments on cookfs, cookit and documents for this conference

Questions?

Thank you!