Google Summer of Code 2018

CernVM-FS powered WebAssembly I/O

Student: Saurav Sachidanand
Mentors: Jakob Blomer, Radu Popescu

Emscripten

- Compiler toolchain for compiling C/C++ to WebAssembly or asm.js.
- C/C++ -> LLVM IR -> asm.js -> wasm
- Also accepts any valid LLVM IR, compiled from any frontend, to generate wasm.
- Toolchain is advanced enough that it’s been used to compile several large, mature code bases such as CPython, SQLite, Vim, Unity Engine, Quake 3.
- Demo: Physics simulation using the Geant4 library, on the Web! -> https://saurvs.github.io/geant4-B1

Emscripten FS backend

- Emscripten has an internal, undocumented file system implementation API
- Similar to FUSE
  - Common file system infrastructure such as inode management, path/inode conversion is abstracted.
  - I/O routines like read, write, readdir are re-routed to file system module mounted at that point.
- My project was to write a new FS module for CernVM-FS using Emscripten's internal APIs
  - Used browser APIs mostly for fetching and caching data

CernVM-FS backend

- Network filesystem on Linux using FUSE and C++.
- Now ported to JavaScript, and embedded in an Emscripten fork.
- Essentially, a C/C++ program that does the following

```
int fd = open("/cvmfs/sft.cern.ch/my/dataset", O_RDONLY);
read(fd, buf, len);
```

- when compiled to WebAssembly using the Emscripten fork, works seamlessly by downloading the appropriate metadata and data, and also caches it on the browser's local storage for later use.

CernVM-FS features implemented

- Regular files, chunked files, directories, different hash algorithms
- Symbolic links, dynamic variable substitution
- Nested catalogs
- Bind mountpoints
- Automatic mounting if accessed under /cvmfs
- Offline mode
- I/O routines: mount, lookup, read, readlink, readdir, stat
- In future: external files, getxattr

Event generators

- Physics event generators written in C/C++: Pythia8, Geant4
  - Can be compiled to WebAssembly using Emscripten to run on browser.
- But require large datasets for computations.
- Without CernVM-FS backend, will need to package several required data files along with HTML & JS files.
- With CernVM-FS however, as the program accesses certain files, they are fetched automatically on-demand and cached locally.
- Makes it easier to compile, no need to track which files to package.
- Also more network efficient: download only required (compressed) chunks

Results with Pythia8

Pythia8 main03	**Without CVMFS**	**With CVMFS**
Compiled files	36 MB	7 MB
Data downloaded from CernVM-FS	-	750 KB
Link	saurvs.github.io/pythia8-main03	saurvs.github.io/cvmfs-pythia8-main03

Results with Geant4

Geant4 B1	**Without CVMFS**	**With CVMFS**
Compiled files	270 MB	19 MB
Data downloaded from CernVM-FS	-	3.3 MB
Link	saurvs.github.io/geant4-B1	saurvs.github.io/cvmfs-geant4-B1

github.com/cvmfs-contrib/cvmfs-emscripten

medium.com/@saurvs

Thank you!