Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
Žilina 6.-9. júla 2011

Využitie služby Open Build Service pre knižnicu na spracovanie objemových dát

HUČKO, Michal, (SK)

Abstrakt. Kompilácia a inštalácia softvéru býva častokrát komplikovaný proces. Špeciálne v prípade softvéru s veľkým počtom závislostí na iných knižniciach, prípadne softvéru so širokými možnosťami konfigurácie. Docieliť bezproblémovú distribúciu takéhoto softvéru na väčšie množstvo platforiem, resp. distribúcií je pomerne pracné. V článku popisujeme využitie služby Open Build Service na binárnu distribúciu knižnice a nástrojov na spracovanie objemových dát. Knižnica obsahuje implementáciu filtrovacích operácií pre rôzne hardvérové a softvérové platformy, akými sú OpenGL, CUDA, či OpenCL.

Kľúčové slová. OBS, Open Build Service, spracovanie objemových dát.

Utilisation of the Open Build Service for a volume data processing library

Abstract. Building and installing of software can be a tedious process. This is especially true if the software has large number of dependencies or it's highly configurable. Ensuring problem-free distribution of such software to wide range of platforms or distributions can be rather complicated. In the paper we describe our experience with utilisation of the Open Build Service for binary distribution of a library for volume data processing. The library contains implementation of filtering operations on various hardware or software platforms like OpenGL, CUDA or OpenCL.

Key words and phrases. OBS, Open Build Service, volume data processing

1 Introduction

Delivering an application to a user can be a non-trivial process. This is especially true if the application has software or hardware dependencies that are not common. Either the application is distributed as a source code which needs to be built or compiled to a package or executable installer. The first option gives higher demands on a user of the application when all the dependencies have to be preinstalled and build process needs to be configured by the user in respect to his/her hardware and software. In the case of binary distribution

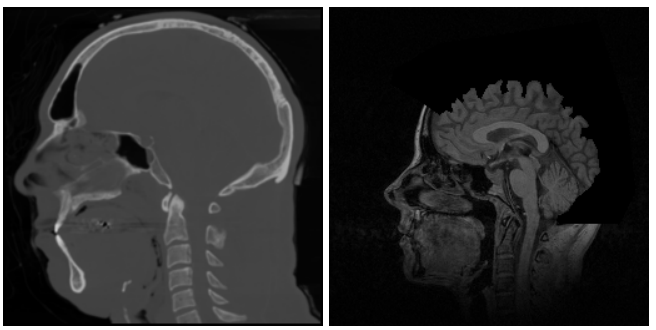
all the tedious work lies on the packager lifting the burden of compilation from the end-user. Still there is lot of work to be done to deliver all configurations of the software to all possible target operating systems. Luckily, this can be automated with certain tools or services – for Linux based operating systems the openSUSE's Open Build Service (OBS) aims to provide the necessary tools and infrastructure. We prospected utilisation of the OBS for distribution of the f3d library – a collection of tools for processing of volumetric data (e.g. CT or MRI scans). As the f3d library takes advantage of special hardware like graphics cards, we explored possibilities of using CUDA or OpenCL under OBS.

2 The f3d library

The f3d acronym stands for 'format three-dimensional' [1] and actually is also a name of the file format used by the library. Purpose of the library is to provide tools for volumetric data processing and analysis. In the following text we'll explain motivation for such library, it's organisation and implementation.

2.1 Volumetric data

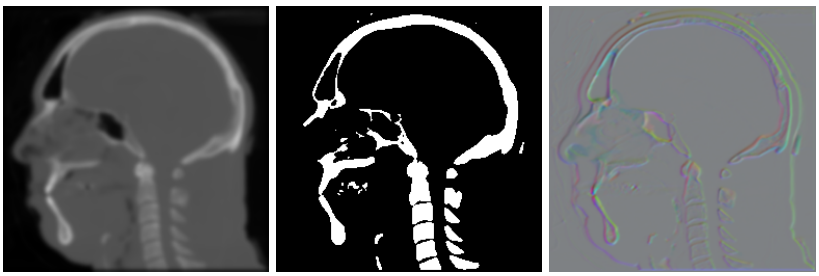
The f3d library is aimed at working with volumetric data used in medicine. These include data from computed tomography (CT), magnetic resonance imaging (MRI) and various other acquiring methods. All the scanning devices create not just a single image, but a volume of voxels (volume elements). One can imagine that the data consist of a stack of common 2D images called slices. In fig. 1 there are example images from CT and MRI. Dependent on the resolution of the used scanner and size of the scanned subject the measured data size ranges from tens of MiB to of couple of GiB. For example, data of dimensions of 1024 cubed having values stored in single precision floating point numbers has 4GiB.



Obrázok 1: CT (left) and MRI (right) sample image fig:CT_MRI

2.2 Data processing

Similar to the case of 2D images also for volumetric data there exist methods which aim at increasing quality of the data or removing artifacts caused by measuring device. For example, if the data is noisy we might want to smooth it in order to avoid errors in subsequent analysis. Operations implemented in the library include: distance transform, edge detection, Gaussian filtering (smoothing), enhancement of tubular structures, thresholding, watershed transform and others [2] (examples are shown on fig. 2). fig:proc_imgs



Obrázok 2: From left to right – blurred image, thresholded image, gradient image fig:proc_i

The previously mentioned operations can be classified in three groups – point, local and global operations. The first two groups require for processing of one voxel either no or only a local neighbourhood of the voxel. Global operations generally might visit any voxel in the data in order to process a single voxel of the volume. When common dimensions of the data are taken into account this yields a great number of computations which – especially in the case of local operations – are identical and rather primitive. When processing of one voxel is independent of the processing of other voxels we have a task which can be easily parallelised. This makes utilisation of SIMD (single instruction – multiple data) capabilities of recent processors and graphics cards an apparent optimisation.

2.3 The library organisation

The f3d suite consist of couple of parts. Each part is written in C/C++ and is managed by the CMake build system. Libraries can be built either as static ones or as shared (we use shared variants for the packaging). Executables (tools) might be in the form of a C++ application or a shell script. We now introduce all the parts and specify their dependencies.

f3dformat Written in pure C, contains functions for reading and saving f3d files.

Depends on zlib

Output libf3dformat.so*

f3dla Basic liner algebra classes used in the processing operations.

Depends on none

Output libf3dla.so*

f3dview Viewer application for the f3d data. Displays slices of the volume in the x, y and z axes.

Depends on f3dformat, wxWidgets

Output f3dview (executable)

f3dfilter A library containing implementations of various filters for different software and hardware architectures.

Depends on yasm, Mesa, CUDA, OpenCL, GSL

Output libf3dfilter.so*, libf3dfilterCPU.so.*, libf3dfilterSSE.so.*, libf3dfilterOpenGL.so.*, libf3dfilterCUDA.so.*, libf3dfilterOpenCL.so.*

f3dclass Collection of tools (executables) executing various filter operations on the volume data.

Depends on f3dformat, f3dla, f3dfilter, zlib

Output libf3dclass.so*, various filter executables named f3d*

The base part of the f3d project is the f3dformat library. It provides functions for reading from and writing to f3d files. External projects which use f3d file format and other f3d packages which need access to f3d files use this library.

For simple inspection of the volume files the f3dview application can be used. It uses wxWidgets as a GUI framework and of course is dependent on f3dformat library.

The rest of the executables come from f3dclass part of the project. Some of the operations are implemented directly in the f3dclass library while rest of the operations are implemented in the f3dfilter libraries. This includes operations which can be optimised by utilising the SIMD processing capabilities of modern processors or graphics cards. SIMD stands for "single instruction – multiple data". For example, modern processors with SSE instructions can perform single operation (i.e. multiplication) simultaneously on four pairs of numbers instead of just one. Similarly, today graphics cards offer possibilities of general parallel computing having multiple computing units. Operations like separable or non-separable convolution can be easily parallelised.

Currently, in addition to basic CPU code, there are implementations of filters for processors with SSE instructions and graphics cards using OpenGL and CUDA (nVidia only). OpenCL versions of certain filters also exist supporting both processors and graphics cards. Because each variant has different hardware and software requirements, it is essential to allow installation of only certain types of filters (i.e. no CUDA on a system with an AMD graphics card). This of course requires checking of the available implementations on-the-fly.

As it was shown earlier, the f3dfilter part of the project creates one f3dfilter library and number of f3dfilterXXX libraries – one for each implementation variant (XXX denotes the variant). The general library contains no implementation of the filters at all, it contains manager classes instead. A manager class probes for the available shared libraries containing hardware dependent filter implementations and returns list of those which were found. Upon request it creates instance of the filtering class and returns it to the caller. Therefore, to successfully filter a volume with some operation the f3dfilter library and at least one f3dfilterXXX library having hardware implementation of the requested operation is required.

3 Open Build Service (OBS)

The Open Build Service[1] (only recently renamed from openSUSE Build Service) is an environment and a service provided by openSUSE/Novell for building, packaging and distribution of software. Originally developed at openSUSE to be used for building add-on packages, it evolved to an open-source build tool supporting also non-SUSE distributions. It can be either installed on private servers or used as a service at `https://build.opensuse.org/`, where anyone can register and build packages in his/her own home project. All this makes OBS an ideal solution for packaging needs. Therefore, we employed OBS in f3d building and distribution noticing weak-spots of the process in respect to the f3d library.

We used the service of `https://build.opensuse.org` for building by registering and creating an f3d subproject in our home project[2]. Because our main goal was distribution of the f3d package to potential users, we wanted to provide packages for as many distributions as possible. After some experiments we narrowed the supported distributions to openSUSE, Fedora, Debian and Ubuntu. On other distributions we had problems with meeting some requirements which in our case was the CMake build system and the wxWidgets library.

Earlier, f3d was distributed only as a source and it was necessary to build it by hand. For users having previously no experience with the library it was often a tedious process, consisting of installing requirements, configuring the project with CMake and selecting the desired variants of the f3dfilter library. When a problem arose it was often too complicated to solve it quickly. Now, all problems are solved by the packager when sources are uploaded to the OBS and the user needs only to install the built packages.

3.1 Problematic requirements

In addition to the earlier mentioned requirements which were not met on some distributions we had also problems with certain hardware variants of the f3dfilter library. The SSE version of the library has simple requirements – it requires only an assembler, which is easy to satisfy. Also no special hardware drivers are necessary. The OpenGL variant requires

[1] `https://build.opensuse.org`

[2] `https://build.opensuse.org/project/show?project=home:granxarixia:f3d`

graphics drivers which provide OpenGL implementation. This requirement can be easily met with the Mesa package which provides open-source implementation of the OpenGL standard. For execution of the filter it may be necessary to have proprietary drivers installed to achieve decent computational times or even run the application because of the missing OpenGL extensions. However, for packaging purposes it's important that the library can be built without the need of any proprietary software and that the produced package has clean and easily met dependencies.

The problematic technology in respect to packaging are OpenCL and CUDA. The latter is a proprietary technology of nVidia – one requires a proprietary SDK from nVidia to build the software. The SDK is not available in the common distribution repositories. Another problem lies in the availability of the proprietary drivers. Although there exists repository with these drivers for openSUSE (and possibly other distributions) it is hosted by nVidia and therefore inaccessible to OBS. This prohibits us from mirroring the needed packages and providing a full dependency tree to users of our repository.

Similar situation to CUDA is with the OpenCL technology. Though the specification is free and managed by the Khronos[3] group, implementations are proprietary. For the consumer hardware AMD provides an OpenCL implementation supporting their processors and graphics cards, nVidia supports their graphics cards and only recently Intel released the Linux version of their OpenCL implementation supporting Intel processors. However, we observed that the code compiled with nVidia's SDK successfully ran on a computer with no nVidia hardware or software having AMD components and AMD OpenCL libraries installed. This shows a possibility to have an open-source library like Mesa which would give a basic OpenCL support. If such library would be created, it would be possible to build and package software using OpenCL. On the client side dependencies would be easily met. Such library would not need to implement the whole functionality as for the utilisation of OpenCL one has to choose hardware platform from the list of supported platforms. With no proprietary drivers empty list could be returned, prohibiting the application from continuing. This would create a similar situation as in case of OpenGL when software using the graphics library can be distributed practically on every system, however runs only when appropriate drivers are installed. No direct dependency on non-open-source software exists.

3.2 The provided packages

Because of the previously mentioned problems our repository currently provides only CPU, SSE and OpenGL variants of the f3dfilter library. The user is free to select which packages he/she wants according to his/her needs. For example, there's no need in installing the OpenGL variant of the filters on a machine without a graphics card and with no X server.

[3] `http://www.khronos.org/`

4 Conclusions

In the paper we presented the f3d library for volume data processing and our experiences with the usage of the OBS service for it's building, packaging and distribution. We mentioned observed problems with proprietary software on which parts of our library is dependent. With the perspective of OpenCL replacing CUDA in the future and possible creation of open-source OpenCL implementation (containing management code only) we hope to deliver full-featured f3d suite. Still we greatly simplified distribution of the f3d library and tools supporting major Linux distributions in comparison to previously required manual building of the software.

Acknowledgement

This work was supported by the Science Grant Agency (VEGA) under contract No. 1/0631/11.

Literatúra

[1] ŠRÁMEK, M. – DIMITROV, L. I.: *f3d – a file format and tools for storage and manipulation of volumetric data sets* : 1st International Symposium on 3D Data Processing, Visualization and Transmission, pages 368-371, 2002

[2] ŠRÁMEK, M. – DIMITROV, L. I. – STRAKA, M. – ČERVEŇANSKÝ, M.: *The f3d tools for processing and visualization of volumetric data* : Journal of Medical Informatics and Technologies, pages MIP-71-MIP-79 2004

Kontaktná adresa

Michal HUČKO,
Katedra aplikovanej informatiky FMFI UK v Bratislave, Mlynská dolina,
842 48 Bratislava, `michal.hucko@fmph.uniba.sk`