PORTING LCG APPLICATIONS

J.A. Lopez-Perez, I. Reguero CERN, Geneva, Switzerland

Abstract

Our goal is twofold. On one hand we wanted to address the interest of CMS users to have the LCG Physics analysis environment on Solaris. On the other hand we wanted to assess the difficulty of porting code, written in Linux without particular attention to portability to other UNIX implementations. Our initial assumption was that the difficulty would be manageable even for a very small team. This is because the implicit respect by Linux of most UNIX interfaces and standards such as the IEEE (PASC) 1003.1 1003.2 specifications.

We started with the LCG External software (http://spi.web.cern.ch/spi/extsoft/platform.html) in order to use it to build the LCG applications such as POOL and SEAL (http://lcgapp.cern.ch/project/). We will discuss the main problems found with the system interfaces as well as the advantages and disadvantages of using the GNU compilers and development environment versus the vendor-provided ones.

INTRODUCTION

The world's largest and most powerful particle accelerator, the Large Hadron Collider (LHC), is being built at CERN [1], the European Organization for Nuclear Research. The computational requirements of the experiments that will use the LHC are enormous: over 1000 PetaBytes of data will be generated each year. The job of the LHC Computing Grid project (LCG) [2] is to prepare the computing infrastructure for the simulation, processing and analysis of LHC data by deploying a worldwide computational grid service.

The Applications area [3] is one of four activity areas in the LHC Computing Grid Project. The work of the Applications area is conducted within five projects: Software Process and Infrastructure (SPI), persistency framework (POOL and conditions database), core libraries and services (SEAL), physicist interface (PI), and simulation.

In particular, the SEAL project [4] is to provide the software infrastructure, basic frameworks, libraries and tools that are common among the LHC experiments. The SEAL package is required by POOL and PI.

SYSTEM ENVIRONMENT

We chose Solaris 8 as the target environment because it is the leading commercial Unix implementation and CMS users had expressed interest in having the LCG environment on Solaris.

Although the interest of the LCG community in this platform seems to be decreasing, it still is a good example of a Unix implementation other than Linux and most of the problems found would apply in porting to any other commercial Unix or POSIX 1003.1 1003.2 compliant operating system.

We chose the GNU compilers because they have been used to develop most of the code and their portability guarantees porting with minimal code changes.

The Sun compilers seem to provide better performance on Solaris, but lack of support for some C++ template features, such as templates with template arguments would make the port more complicated.

We do not have the GNU binutils package available on Solaris so we have to use the Sun linker to make shared libraries and dynamic executables. We also have to use the Solaris dynamic linking mechanisms. In this environment the GNU options, such as "–shared", are not understood by the linker, so we have to use the "-Wl" gcc option to pass options to the sun linker directly. The typical options required are "-G" to make shared libraries and "-h" to set the internal name of the libraries.

We have built all object files, including the External Software ones, as pure text using the "-fPIC" gcc option in order to avoid errors due to relocations against non-writable, allocatable sections when building shared libraries and dynamic executables.

We use the default instruction set architecture available in our environment, this is ILP32, this is SPARC V8 architecture as defined by Version 8 of the SPARC Architecture Manual. However we have to emulate the behaviour of the Sun compiler (ANSI C plus K&R C compatibility extensions) by defining __STDC__ = 0 for the system headers to define the explicit 64 bit types, such as int64_t and unit64_t required by SEAL in 32 bit environment. These types are implemented using the "long long" type of gcc.

We also had to make sure that other "longlong" types defined by the ReflectionBuilder in SEAL are identified with the explicit 64 bit types as otherwise they were not recognized as identical when building the dictionary.

In this environment we have ported SEAL 1.3.3 and all the required External Software.

EXTERNAL SOFTWARE

Most of the following ported programs are installed using the "configure + make + make install" method with the appropriate options. There are many variations in

which they also use some ”make check” or ”make test” step or no ”configure” or ”make install” command is necessary so the method cannot be guessed before reading carefully the installation instructions. In other cases the installation method is completely different involving other tools, ad hoc scripts, compilation of special files in some directories, links, etc. In principle we ported just the version required by SEAL, POOL and PI but we also ported other program versions in the cases in which the SPI project supports them or when the required version gave some problem

- **Bison 1.875**: As is written in the Doxigen package installation instructions, ”Versions 1.31 to 1.34 of Bison contain a ‘bug’ that results in compiler errors (. . .) This problem has been solved in version 1.35”. Then, we installed the latest version of Bison, the 1.875 one that worked fine.
- **BLAS 20030829**: This product does not use the usual installation method. It requires a compilation of several FORTRAN ?les into different directories.
- **Boost 1.30.2, 1.31.0**: Required by SEAL, POOL and PI. This package uses an installation method based on the ”Jam” program. The installation of the 1.30.2 version, required by SEAL, fails because of incompatible linker options. The latest version, 1.31.0, was installed without errors but SEAL gave symbol referencing errors while installing because of the changes between the two versions, so we decided to correct the previous one. We needed to modify the compiler commands in the Jam configuration for Boost (gcctools.jam) to pass the right options to the Solaris linker to make shared libraries.
- **bz2lib 1.0.2**: Required by SEAL. We had problems when building the shared library libbz2.so because the Makefile was assuming RedHat 7.2 Linux on an x86 box. To solve the problems, we needed to change the gnu linker options to the Sun ld ones. We used ”-G -h”.
- **CERNLIB 2003**: Required by CompHEP. The CERNLIB maintainers regularly make this package in Solaris, however the latest currently ported version is the 2002 one so we tried to port it anyway. It uses its own installation method using its own scripts so it’s difficult to trace compilation errors when found. There were problems related with compiler and linking options because the scripts assume the Sun compiler and there are no options to change these so we had to find and edit the required files.
- **CLHEP 1.8.1**: Required by SEAL and PI. It needs a compilation of specific files on a temporary directory and manual creation of links.
- **CMake 1.8.3**: This compilation tool is used by the GCC-XML package. The compilation gave symbol referencing errors. Anyway we downloaded compiled binaries for Solaris from the official web page. They were compiled for Solaris 7 but we got no errors using them with Solaris 8.
- **CppUnit 1.8.0, 1.10.2**: Required by SEAL and PI. This package was already ported by the SPI project to Solaris but they used the Sun compiler instead of gcc so we had to redo it. It gave errors when compiling on an AFS directory so we had to compile it in a local file system. One of the tests failed (the hierarchy one) so we tried the latest version, 1.10.2. That one passed all the tests.
- **Doxygen 1.3.3, 1.3.7**: There were two problems installing the documentation because the Makefile did not find an image (we can make the documentation without that) and it did not make the table of contents. This problem was solved just by running the ”make pdf” command twice.
- **Expat 1.95.5**: Required by PI.
- **GCC-XML 0.4.2**: Required by SEAL, POOL and PI. It requires a compilation program called CMake. It gave no installation errors using the method described in the package documentation but afterwards we found that a module was missing. To solve the problem we had to perform a procedure to patch the cc1plus program of the gcc 3 compiler replacing a file (xml.c [7]) distributed by the SPI project. We also found that when gccxml called the cc1plus binary, some compiler flags were lost along the way so gcc couldn't find some headers and didn't take the right options. To solve the problem we had to change the share/gccxml-0.4.2/config file adding all the missing C preprocessor symbols and include directories to the GCCXML_USER_FLAGS entry. We obtained these settings by running gcc in verbose mode.
- **GSL 1.4**: Required by SEAL and PI.
- **Jam 3.1.9**: This program is required to install Boost. It’s installed using its own script.
- **IgProf 1.3.0**: This is a profiling tool working only on x86linux machines.
- **LAPACK 3.00**: To install this package we had to modify the INSTALL/make.inc file. We must give there a platform name, used to create a directory with the built libraries, and the appropriate compiler name and options. We had to change the options in order to make shared libraries in Solaris.
- **MySQL 4.0.13**: Required by POOL. The MySQL developers recommend using the binaries they provide and they have already made the Solaris port. Anyway, the SPI project installed that version with their own compiler options so we did the same. We got no errors but one test failed, the func_crypt one.

- **MySQL++ 1.79**: Required by POOL. This package uses the automake and autoconf tools and, during the installation, we found a "missing separator" error running the make command related with an automake problem. The installation of the latest autoconf version didn't solve the problem. Anyway, as documented in the SPI project MySQL++ web page [8]: "As mysql++ needs to be patched by the POOL team (due to obsolete include style), they have decided not to use it in the future. So, it will not be compiled by icc/ecc and furthermore it will be replaced by POOL internal functions". Currently MySQL++ is still required by the latest POOL version, 1.7.0, but in the near future it will not be required anymore.
- **OPro?le 0.7.1**: This is a platform-specific profiler for x86-linux so we have not made the port for this package. Anyway, it is not required by SEAL or POOL.
- **Otl 4.0.67**: Required by POOL. It is just a C++ library which we copied without compilation.
- **Oval 2.15.3, 3.5.0**: Required by SEAL and POOL. The version ported by the SPI project was 2.15.3 but we realized that the version required for SEAL and POOL is 3.5.0 so we downloaded this version.
- **Pcre 4.50, 4.40, 4.30, 4.20**: Required by SEAL, POOL and PI.
- **Python 2.2.2, 2.2.3, 2.3.3**: Required by SEAL, POOL and PI. The 2.2.2 version is the one required by SEAL 1.3.3 and POOL 1.6.5 . We changed the value of _FILE_OFFSET_BITS to 32 in the config header in order to be compatible with the default settings in the Solaris. When installing SEAL we realized that some of the scripts distributed with Python in lib/python2.2/xml were taken by precedence over similar ones from the pyexpat program in the PyXML[9] package that we were referencing on the PYTHONPATH environment variable. The solution to this problem is to install the package in the python lib hierarchy. We had to modify the Makefile in order to generate libpython2.2 as a shared library in order to facilitate linking with SEAL.
- **PyXML 0.8.3**: Required by SEAL in the cases in which the pyexpat package is not installed by default on Python. It is installed with a Python script. We found a problem when installing PyXML on an AFS directory that was accessible to two users because when one user tries to run the product, some .pyc files are compiled and python fails when trying to regenerate them for the second user because they are owned by the first one. The problem can be solved by write-protecting the whole installation just after the installation.
- **QMTest 2.0.3, 2.2.0**: Required by SEAL and POOL. One Solaris version had been made by the SPI project but no log information is available so we cannot know which compiler was used and we installed the package from the source. We got some warnings and a failed test. The latest version, 2.2.0, gave the same result.
- **Root 3.10.02**: Required by SEAL, POOL and PI. We found three bugs that were preventing Root from being installed on Solaris. They have been reported and they were corrected on the 4.00.02 version, so they don't affect the last SEAL version, the 1.5.0 one which requires Root 4.00.08, but they affected the previous ones. Anyway, we installed the required Root 3.10.02 version just modifying slightly three files to avoid the bugs. We also got errors related to the libpng and libungif libraries but they were corrected by installing the latest versions of these libraries.
- **SCRAM 0.20.0**: Required by SEAL, POOL and PI. It is the software configuration, release and management tool used by the CERN applications [11]. It is installed using a Perl script that gave no problems. Please note that the source directory cannot be moved or renamed after we install SCRAM because the program looks for it with the same name and location. The documentation is also on that directory and is not installed. We had problems finding the sources and the installation instructions due to the lack of documentation. You can find them at [12].
- **SLOCCount 2.22, 2.23**: It doesn't need to be installed. You just need to run the make command to build the executables and the man pages on a predefinded directory.
- **UnixODBC 2.26, 2.28**: Required by POOL. As done by the SPI project on the installed Linux version, the GUI components were disabled because they require Qt and it is not installed by default on CERN machines.
- **Uuid 1.32, 1.34, 1.35**: Required by SEAL, POOL and PI. This package needs a patch and is part of the general Ext2 Filesystem Utilities package (e2fsprogs). To install just the uuid library you have to do the "make install" step only in the lib/uuid directory. This will give some errors if you are not root but you can ignore them. We called the "make" command using the "-e LN=cp" option. Installing 1.32 we got some symbol referencing errors. The newer 1.34 and 1.35 versions work better and SEAL can use them without problems.
- **Valgrind 2.0.0**: Required by SEAL, POOL and PI. This is a platform-specific memory debugger for x86linux so this package has not been ported. You can instruct SEAL to ignore that package during the installation, setting in the

SCRAMToolBox/LCGcon?gs/toolsCERN.conf configuration file, "+ VALGRIND_BASE : ".

- **wxPython 2.3.3.1, 2.4.0.1, 2.4.2.4**: Required by POOL. The installation of this package is rather complex because it requires to use the make command in several directories, it also requires to run a Python script and the tests are also Python scripts located in a different directory. We found many errors building the 2.4.0.1 version, required by POOL up to the 1.6.5 version. Firstly we got some file processing errors that we solved taking out the "--enable-rpath" con?gure option, used by default on Linux. In this case we must include the /lib directory path in the LD_LIBRARY_PATH environment variable. We also got errors installing the /locale files, solved using the GNU msgfmt system program instead of the Sun's one. In addition, the Python script had problems finding some source files. We noticed that these files were located at the contrib directory so the references "contrib/(…)/contrib." needed to be changed by just "contrib.". We just added some links to the needed directories. We didn't find this error in the latest wxPython version (2.4.2.4). Please note that to install it out of our Python directory the option "build_ext –inplace" has to be added to the Python installation script.
- **XercesC 1.6.0, 2.1.0, 2.2.0, 2.3.0, 2.5.0**: Required by POOL. We built also the optional samples.
- **Zlib 1.1.4, 1.2.1**: Required by SEAL and PI.

SEAL

The installation instructions [13] inform us that SEAL needs an external program, SCRAM, which downloads the required sources from a cvs repository and then performs the required installation commands generating the required Makefiles from different locations, using the configuration files at the SEAL cvs repository, written in xml style. These Makefiles are not stored locally so in many cases we have no access to the compilation commands which may complicate debugging.

The installation method involve a cvs check out of some SEAL configuration files, a SCRAM bootstrap command and a SCRAM build command in the suitable directory.

To modify the SEAL source, we had to create our own cvs repository and check the entire SEAL cvs repository in.

For each change in the code we had to check it out, modify it and then check it in with the appropriate cvs tag (SEAL_1_3_3), because this tag is also in the configuration files and is the one which SCRAM will download.

We also had to create our own cvs repository for the SCRAMToolBox. This is a set of configuration files where SCRAM finds all the information about the compilers and external programs that SEAL will use.

Afterwards, we had to change the SEAL configuration files to point to these repositories instead of to the default ones. In the check out step, we hit a cvs bug where it complains about not being able to open a temporary file. We circumvent this by touching the file in question. The bug will be solved in the next cvs version.

Many SCRAM configuration files have a separated block for each installation architecture so we just had to add the missing Solaris ones. In some of the configuration blocks the Solaris architecture was already there but only with the Sun CC compiler so we had to add a new one using gcc.

We had to change all the required external software paths as well as the compiler paths and options. The compiler options are distributed on different files that we had to modify.

We had to configure the link options to use the Sun linker. As part of the configuration, we also had to add all the required system libraries to the link command line in order to avoid symbol referencing errors.

The lcgdict script used tcsh syntax when invoking /bin/csh, while scripts within gccxml used bash syntax when invoking /bin/sh. Csh and sh are not identical to tcsh and bash in Solaris so the references had to be corrected.

We had to change wrong "defines" in several places in order to cater for Solaris, in particular we had to replace "ifdef __linux" by "ifndef _WIN32" in seal/Scripting/PyLCGDict/src/LCGDictWrapper.cpp.

We found that seal/Foundation/SealBase/src/SharedLibrary.cpp hardcoded the Linux method to find the link map for dynamic linking, so we had to add support for Solaris using the dlinfo() library call.

Some SEAL files in the cvs repository were missing the version tag, therefore SCRAM was not downloading them and SEAL could not find them. All the files under "seal/Scripting/PyLCGDict2" only had the SEAL_1_4_0 tag. Similarly the seal/Documentation/WebSite/doxygen.css file, and the seal/Documentation/WebSite/workbook directory did not have any tag.

CONCLUSIONS

We have ported most of the External Software as well as SEAL Version 1.3.3 to Solaris 8 using the GNU compilers.

We have not been able to make POOL yet but we believe that thanks to the work done on the External Software it should be easy to do now if some interest on the Solaris platform persists.

Changes in the code have been minimal thanks to the portability of the GNU compilers, however the port is not trivial because of the use of a different linker with different options and different dynamic loading code, as well as typing differences that produce some subtle errors.

Using the Sun compilers would have provided better integration with the Sun linker and system headers, but

differences in code support, limitations in support for C++ templates would have made the port more difficult.

The complexity of the layer introduced by the SCRAM software configuration tool and relying on a collection of heterogeneous software built using different tools increases the challenge.

The External Software is available in AFS in the "/afs/cern.ch/project/sun/solaris/lcg/external" hierarchy using the SPI layout and conventions.

The modified SEAL sources as well as the SCRAMToolBox ones for Solaris are available in the UI CVS repository in http://ui.cvs.cern.ch/cgi-bin/ui.cgi/

ACKNOWLEDGEMENTS

We wish to thank Philippe Defert, Manuel Guijarro, Andreas Pfeiffer, Shaun Ashby and Lassi Tuura for kindly answering our questions.

REFERENCES

[1] http://www.cern.ch/
[2] http://lcg.web.cern.ch/LCG/
[3] http://lcgapp.cern.ch/project/
[4] http://savannah.cern.ch/projects/seal/ http://lcgapp.cern.ch/project/cls/
[5] http://spi.cern.ch/extsoft/index.html
[6] http://service-spi.web.cern.ch/service-spi/external/gccxml/0.4.2_patch1/install.txt
[7] http://service-spi.web.cern.ch/service-spi/external/gccxml/0.4.2_patch1/_SPI/xml.c
[8] http://spi.cern.ch/extsoft/mysql%2B%2B.html
[9] http://pyxml.sourceforge.net/
[10] http://lcgapp.cern.ch/project/spi/lcgsoft/index.html
[11] JP.Wellisch, C.Williams, S.Ashby, Computing in High Energy and Nuclear Physics, (2003)
[12] http://spi.cern.ch/cgi-bin/scrampage.cgi
[13] http://seal.web.cern.ch/seal/snapshot/devguide/howtorelease.html