Classdesc: C++ Reflection for Scientific Computing

Russell Standish
High Performance Coders
hpcoder@hpcoders.com.au

Abstract

Reflection is the capability of querying aspects of an object's type and structure at runtime, that normally is discarded as part of the compilation process. It is the key to automatically supporting object serialisation and automatically binding scripting languages to compiled models. In a typical scientific model, code supporting initialisation/configuration as well as checkpoint/restart can often be a significant chunk of the overall codebase, and creates a constant maintenance burden as the scientific model changes over time.

Whilst a number of programming languages support reflection intrinsically, popular high performance programming environments like C++ and Fortran do not. Classdesc is a system for automatically providing reflection capability to C++ programs that has been in constant use and development since 2000. This presentation covers the concept of a class descriptor, from which the name Classdesc is derived, and the existing descriptors for binary, XDR, JSON and XML serialisation, and automatic bindings for Python, Javascript and a REST API. The binary serialisation capability is leveraged to provide easy-to-program distributed memory parallel programming libraries based on industry standard MPI, called ClassdescMP and Graphcode.

Classdesc

Classdesc generate recursive "compiler-generated" *descriptors*:

```
void pack(pack_t& buf, string d, Foo foo)
{
  pack(buf, d+".a", foo.a);
  pack(buf, d+".b", foo.b);
}
```

These are analogous to compiler generated assignment or copy operators.

Features

- Open Source under MIT License.
- Under continuous development since 2000.
- Automated code generation means less user written code, and more maintainable code bases.
- Binary & XDR machine independent serialisation, enabling *checkpoint/restart* code to be automatically generated, and persistent objects.
- TCL bindings in the EcoLab simulation framework, allowing *rapid application development*, and dynamic investigation of simulation whilst it is running.
- ClassdescMP, an MPI parallel distributed memory library, using binary serialisation.
- Graphcode represents a computation as the nodes of a graph, and the communication as the edges. This allows for unbalanced parallel computations to be dynamically balanced as the simulation proceeds.
- XML serialisation.
- JSON serialisation.

What's new

- Modern C++ support (ie C++11 and above), including `shared_ptr` and `unique_ptr` support for polymorphic descriptors.
- Constructor bindings — constructors can be called to instantiate C++ objects from a scripting language, such as Python.
- Overloaded methods — methods are distinguished by number of arguments, and type of arguments.
- Automated Python bindings, built on top of boost::python
- RESTService bindings — allows your model to be run as a REST Service, which can be queried and manipulated by another system (eg a web page in javascript) over the internet.
- Javascript (Emscripten) bindings coming soon, which will allow automated embedding of C++ models in a web page using *web assembly* (wasm).

See https://classdesc.sf.net

1. Standish R.K. (2019) "C++ Reflection for Python Binding", *Overload Journal*, No. 152, 15–18.
2. Standish R.K. (2016) "Classdesc: A Reflection System for C++11", *Overload Journal*, No. 131, 18–23.
3. Standish, R.K. and Madina, D. (2008) "Classdesc and Graphcode: support for scientific programming in C++", arXiv:cs.CE/0610120
4. Standish, R.K. and Madina, D. (2003) "ClassdescMP: Easy MPI programming in C++" in *Computational Science*, Sloot *et al.* (eds), LNCS **2660**, Springer, 896. arXiv:cs.DC/0401027
5. Leow, R. and Standish, R.K. (2003) "Running C++ models under the Swarm Environment", in *Proceedings SwarmFest 2003*. arXiv:cs.MA/0401025
6. Madina, D. and Standish, R.K. (2001) "A system for reflection in C++", in *Proceedings AUUG 2001: Always on and Everywhere*, 207. ISBN 0957753225 arXiv:cs.PL/0401024