LABORATOIRE DE L'ACCÉLÉRATEUR LINÉAIRE

GoCxx: A tool to easily leverage C++ legacy code for multicore-friendly Go libraries and frameworks

S. Binet

http://golang.org

Context: why Go ?

Current HENP libraries and frameworks were written before multicore systems became widely deployed and used. A *"single-thread"* processing model emerged and is now greatly impairing our abilities to scale in a multicore/manycore world.
Could HENP migrate to a language with none of the deficiencies of C++ (build time, deployment, low level tools for concurrency) and with the fast turn-around time, simplicity and ease of coding of Python ?

Could this be Go ?

Go: http://golang.org

- **concurrent**, compiled, **garbage collected**
- **open-source** general programming language
- *now stable: version 1* since March 2012
  - feel of a **dynamic language**
  - safety of a **static type system**
  - compiled down to machine language (so it is **fast**)

GoCxx: bitbucket.org/binet/go-cxxdict

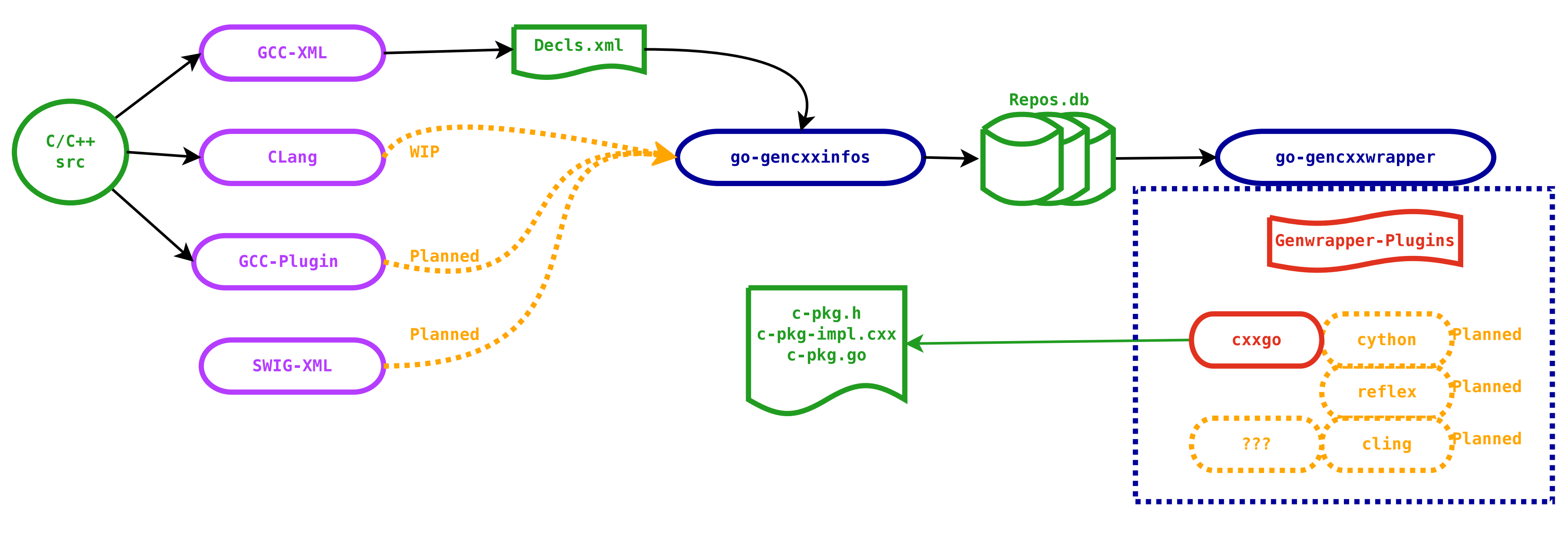
- The Go standard library already ships with a *FFI*, called cgo
  - **C support**
  - **no support for C++**
- SWIG-2 has support for Go
  - supports gccgo and gc compilers
  - **can not parse TObject.h**
- GoCxx
  - address all shortcomings of cgo and SWIG
  - reuse all the know-how of Reflex+GCC-XML
  - generate bindings for all C++ constructs
  - provide access to C/C++ legacy libraries

Architecture & Strategy

UNIX-y model: separate binaries doing one thing.

- go-gencxxinfos: generates Go serialized objects describing C/C++ libraries
- go-gencxxwrapper: generates bindings for the C/C++ libraries
- pkg/cxxtypes: models C++ types and identifiers
- pkg/wrapper: infrastructure and logic to load the C/C++ types and identifiers informations. Schedules the plugins to generate the bindings.

GoCxx components & Workflow



go-gencxxinfos

- loads a GCC-XML-produced XML file
- creates types (classes, structs, enums, . . .)
- creates identifiers (enum values, variables, functions, namespaces)
- serializes representation into a repository file

```
shell> gccxml lib.h -fxml=lib.xml
shell> go-gencxxinfos \
         -fname lib.xml -o repo.db
```

go-gencxxwrapper

- loads repository files
- loads wrapper plugins (cxxgo, reflex,...)
- selects types and identifiers to be wrapped
- runs each loaded plugin to create the wrappers

```
shell> go-gencxxwrapper \
   -fname repo.db -sel sel.xml \
   -pkg foo -o outdir
shell> ls outdir
foo.h foo-impl.cxx foo.go
```

End user code

```
import pkg "some/path/to/foo"
func main() {
     p1 := pkg.P4EEtaPhiM()
     p2 := pkg.P4PxPyPzE(/*...*/)
     println("dR=",pkg.DeltaR(p1, p2))
}
```

Future work

- Finish CLang and GCC-plugin input plugins
- Investigate other output plugins (cling, reflex, cython, Java, C#, . . .)