GSOC 2020 - XAPIAN GO BINDINGS

Mar 31, 2020

CONTENTS

CHAPTER

ONE

ABOUT YOU

- Name: S SRINIVAS YADAV
- E-mail address: vasu.srinivasvasu.14@gmail.com
- IRC nickname(s): srinivasyadav227
- Any personal websites, blogs, social media, etc: NIL
- github URL: https://github.com/srinivasyadav18
- Biography:

I am srinivas , an 18 year old second year undergraduate student from Keshav Memorital Institute of Technology ,Hyderabad, India. I am a tech enthusiast and a person who loves sports.

1.1 Background Information

Have you taken part in GSoC and/or GCI (https://codein.withgoogle.com/) and/or similar programmes before? If so, tell us about how it went, and any areas you would have liked more help with.

No

Please tell us about any previous experience you have with Xapian, or other systems for indexed text search.

I dont not have any previous experience with xapian or any other indexed text search.

Tell us about any previous experience with Free Software and Open Source other than Xapian.

There is a software Barrier,an Open source KVM Software where you can use one keyboard and mouse on different computers nearby when all are connected to same network..I was trying their source code and have implemented few small c programs in linux for capturing device events from one computer and sending to other system in the local network with known IP address.

What other relevant prior experience do you have (courses taken at college, hobbies, holiday jobs, etc)?

I have taken course on Machine Learning and its science applications where I have learnt the basics of machine learning and deep learning and have worked with Convolutional Neural Networks and facial recongnition, a course on C programming and assembly language where I have learnt how C code gets compiled and to use inline assembly in C for 8086 microcontroller, I am familiar with cuda and parallel programming. I have worked with small scale electronics and IoT projects(with raspberry,arduino). I love problem solving and I keep learning different algorithms and datastructures. My hobbies are playing cricket and exploring new places.

What development platforms, tools and methods do you prefer to use?

I prefer Visual Studio Code when I'm learning a new language becuase it provides auto-completion and gives information like what that particular class or function is about, when I get the basics I switch to vim ,the editor I use for

more focus , speed and accuracy. I use Docker for testing in linux and recently started using Travis CI . I prefer Guake terminal for fast switching between code and terminal.

Have you previously worked on a project of a similar scope? If so, tell us about it.

No

What timezone will you be in during the coding period?

GMT+5.30 (IST)

Will your Summer of Code project be the main focus of your time during the program?

Yes , Summer of Code is my main focus this summer.

Expected work hours (e.g. Monday–Friday 9am–5pm UTC)

Monday–Saturday 5am-3pm UTC

Are you applying for other projects in GSoC this year? If so, with which organisation(s)?

No

CHAPTER

TWO

IMPLEMENT BINDINGS FOR GO

2.1 Motivations

Why have you chosen this particular project?

My main areas of interest includes C++ and Go.I like the thread support provided in Go and C++. As go achieves concurrency in two ways one using goroutines and other using channels . Goroutines are often called light-weight threads as they occupy very less stack space(8KB) compared to posix threads(1-2MB) and more over many go threads can be multiplexed to one os thread where as pthreads occupy one OS thread that could be huge performance boost when dealing with millions of go threads which are going to perform tiny tasks(go routines) (here is an intresting article on go threads vs java threads https://rcoh.me/posts/why-you-can-have-a-million-go-routines-but-only-1000-java-threads/). I have previously used cgo and swig in small projects(Implementing fourier transformations in Go using C as backend for as fourier transformation is used heavly on larger data sets in machine learning). Docker and Kubernetes are the two famous projects written in golang and are widely used. So I would like to provide support for xapian in Go.

Who will benefit from your project and in what ways?

As Go lang is emerging now a days rapidly and its being used in servers and distributed systems, providing a Go API for the people who want to use Xapian search engine in thier servers would be a huge benefit. It would be benificial for Golang community ,Swig and Xapian.

2.2 Project Details

Describe any existing work and concepts on which your project is based.

I have done some intial implementation of go bindings from the existing swig base wrappers written (in /xapian/xapian-bindings) which are available in my git repository.

Do you have any preliminary findings or results which suggest that your approach is possible and likely to succeed?

Yes, my intial work on bindings definitly show that bindings can be implementend in Go with c++ shared libraries.

What other approaches have you considered, and why did you reject those in favour of your chosen approach?

Initially I was able to provide bindings after installation of the shared library on the host system , but then I have realised that before installing the core we need to build the bindings and I was using pkg-config , later I started using flags from configure as one of my mentor (olly) suggested.

Please note any uncertainties or aspects which depend on further research or investigation.

Compatiblilty of libtool with the go build system.

How useful will your results be when not everything works out exactly as planned?

I am confident enough that the project would be implemented and completed properly on time. If in case some thing goes wrong , I would continue the work on go bindings after GSoC timeline as I am very much interested in this project and love to contribute to xapian and would like to maintain xapian go bindings if I am allowed to. After GSoC I would like to take up Java Bindngs Improvements and Python Bindings Improvements If none has take up those projects in GSoC.

2.3 Project Timeline

In this summer I would like to do three things:

1. Make the bindings in Go compatible with the Xapian build system.
2. Provide bindings in go and complete API reference (as it is provided for C++) using godoc,Go lang's official documentation tool where any one can look at the documentation if they just have go installed.The user can view the xapian documentation by just running go doc xapian from the terminal or command prompt. And the documentation would be providied in godoc official website and xapian.org.
3. Provide the go get feature for using xapian bindings in go with proper guide lines regarding whether the xapian core is installed or not.

Plan :

- Go is Statically-typed language and each variable can only be of one type.

  Example :

  ```
  type Integer int
  func example(x Integer){
      fmt.Println(x)
  }
  func main(){
      var VarA Integer =10;
      var VarB int = 20
      example(VarB)
  }
  ```

  This produces an error when VarA is passed to func.

  Error : cannot use x (type int) as type Integer in argument to example

  When swig wraps enums of a particular class ,for example during wrapping of stem_strategy enum in TermGenerator class swig generates a type for enum (xapian.XapianTermGeneratorStem_strategy) and defines variables for the elements of that particular enum(such as TermGeneratorSTEM_NONE,TermGeneratorSTEM_SOME of type int).So since each variable of int type , TermGeneratorSTEM_NONE(of type int) should be converted to XapianTermGeneratorStem_strategy type before passing to function which needs the element an enum as an argument.

  Swig generated code :

  ```
  type XapianTermGeneratorStem_strategy int
  var TermGeneratorSTEM_NONE int = _swig_getTermGenerator_STEM_NONE_TermGenerator()
  var TermGeneratorSTEM_SOME int = _swig_getTermGenerator_STEM_SOME_TermGenerator()
  ```

  Error and Insertion of code for type conversion :

func main(){

tm := xapian.NewTermGenerator()

// tm.Set_stemming_strategy(xapian.TermGeneratorSTEM_NONE) –> fails // an explicit type conversion is required . So Providing automatic conversoin would be simpler for user

tm.Set_stemming_strategy(xapian.XapianTermGeneratorStem_strategy(xapian.TermGeneratorSTEM_SOME))

}

Output when failed :

/root/xapian-enum.go:8:26:

cannot use xapian.TermGeneratorSTEM_NONE (type int) as type xapian.XapianTermGeneratorStem_strategy in argument to tm.Set_stemming_strategy

- Go supports constants which can be defined with const keyword. Swig wrapps all the #define macros and constants declared in %constant directive as Go constants (const). But the constants which are defined using %constant directive in xapian-headers.i are being wrapped as variables.

  Swig generated code :

  var DB_CREATE_OR_OPEN int = _swig_getDB_CREATE_OR_OPEN()

  DB_CREATE_OR_OPEN is a constant in c++ but this is being wrapped as a variable.So all the constants need proper wrapping such as

  const DB_CREATE_OR_OPEN = 0

- Go supports Iterators by natural syntax using channels and convential methods such as Iter.Next().

  1. Using channels one could use for-range construct.

     for i := range doc.terms(){ i.GetTerm() // methods to get term from the iterator at that position.

     }

  2. Using methods such as Iter.Next() as used in Go lang standard library (Container List https://golang.org/pkg/container/list/).

  Both standard method /* for iter.Next(){ . . . code } */ *and* */* for-range construct would be made available for user (these are just like auto-ranged based loops in c++) */*

  Go support multiple return values , therefore rewrapping the interfaces which return iterators to both begin and end iterators in one function call as below. /* begin,end := doc.Terms() */

  Code for Iterator :

  %rename (Wrapped_Document) Document;

  %insert(go_wrapper) %{

  //rewrapping the Document interface currently adding only extra method Terms() to show how term iterator can be

  //used with go for-range construct

  type Document struct {

  Obj Wrapped_Document

  }

  func (d *Document) Terms()<-chan TermItertor {

```
    ch := make(chan string)
    begin := d.Obj.Termlist_begin()
    end := d.Obj.Termlist_end()
    go func() {
      for !begin.Equals(end) {
        ch <- begin
        begin.Next()
      }
      close(ch)
    }()
    return ch }
%}
    Usage :
    for term := range myDoc.Terms() {
      fmt.Println(term.Get_term())
    }
```

- Go supports errors as return values . A language like c++ have try catch block Go has three similar constructs for dealing with exceptions, they are panic defer and recover.A Panic is similar to an exception which can occur an runtime exception. C++ exceptions can be handled in go from swig wrappers as shown below. Which ever class function throws an exception in c++ , the wrapped function in Go returns the error as value. Providing error in same pattern as provided in standard libraries of Go lang. Example Os package error handling - https://golang.org/pkg/os/ /*

  Code for exceptions (DatabaseOpeningError):

```
    %exception {
    try {
    $action;
    } catch (Xapian::DatabaseOpeningError & e){
      //calls the panic in go
      _swig_gopanic(e.get_error_string());
    } catch (std::exception & e){
      _swig_gopanic(e.what());
    }
    }
    //Rewrapping of NewDatabase function to add a return value for error.
    %rename (Wrapped_Database) Database;
    %go_import("fmt")
    %insert (go_wrapper) %{
      type Database struct {
```

```
    Obj Wrapped_Database
  }
  func NewDatabase(a ... interface{}) (db Database,err error){
    defer catch(&err)
    db.Obj = NewWrapped_Database(a...)
    return
  }
  func catch(err *error){
    if r := recover(); r != nil {
    *err = fmt.Errorf("error %v",r)
    }
  }
%}
```

Usage :

```
db, err := xp.NewDatabase("/no_database")
if err != nil {
    fmt.Println(err)
    os.Exit(2)
}
```

OUTPUT:

```
error No such file or directory
exit status 2
```

- Go does not support constructors but this can be done with an extra helper function that takes slice of interfaces which swig does by default during the wrapping, but little extra code need to be added for constructors and functions when overloaded as they take slice of interfaces in ellipse syntax. interface{} in golang means any type. Slice of interfaces mean collection of interfaces(resizable array). Go supports variable number of arguments of different type to functions as func myfun(a ... interface{}) which is used during constructor and function overloading.The word rewrapping is used often a lot, this means wrapping the swig generated object inside a go struct with additional methods and for cutomisation to provide more Natural Go API as described in SWIG Go Documentation.This is one of key features SWIG provides for cutomising and adding additional code.
- Go has its own documentation tool for generating documentation for the go code . Providing documentation for the classes each week that I work on particular week. Most of the classes require rewrapping to provide a simple interface as SWIG generated interfaces is not that simple to use.
- Go has its own test package for testing the packages which would be put in Makefile. (https://golang.org/pkg/testing/)
- In month April :
  - Understand the exceptions in other bindings which are auto-generated.
  - Understand go build system deeper and work on it if it can be integrated with libtool or possibly prepare a plan to create a new separate build system with only auto tools.

  - Next Two weeks - Understand Xapian implementation of existing bindings and Xapian classes more. (Im not quite familiar with classes related to latlong, MatchSpy, KeyMaker).

Community Bonding Period :

- Implement the build system properly for bindings in go using the existing wrapper(for linux) and review it with the mentors.
- Understand Xapian bindings for other implemented languages.

First Month :

June 1st-6th :

- Support Iterators (Position, Posting, Term, Value).
- Wrapping c++ constants as natural Go constants.
- Change all the function names which are to be exported to PascalCase.(1 day)

  (Go uses Pascal Case when exporting functions from a package and camelCase for unexported functions.)
- Rewrapping of the wrapped interface generated by SWIG by defining a struct that includes the swig-wrapped object and adding necessory go code using %insert(go_wrapper) in go.i swig interface file.(4 days)
- Document the iterators interface using godoc and reStructuredText file.(1 day)

June 8th-13th :

- Rewrap the Database class for providing errors(as return values) to the functions that throw errors.(3 days)
- Rewrap the Document class for rewrapped iterator interfaces.(2 days)
- Document both the classes.(1 day)

June 15th-20th :

- Rewrap the QueryParser , Query class for enums type conversions and provide return error for the methods for corresponding functions which throw exception in c++.(2 days)
- Adding additional code for constructors and overloaded functions becuase of rewrapping, rewrap the functions which return iterators in Go idomatic way.(2 days)
- Document these classes.(2 days)

June 22nd-27th :

- Support EsetIterator , MsetIterator classes by rewrapping the swig generated interfaces and adding additional go code for iteration.(2 days)
- Document EsetIterator,MsetIterator,ESet,MSet,RSet classes using godoc and reStructuredText.(2 days)
- Providing tests for classes so far implemented with go test package and edit Makefile for adding new tests.(2 days)

June 29th :

- Phase 1 Evaluation.

Second Month :

June 29th - July 4th :

- Rewrap the Enquire class and all the overloaded functions,constructors(2 days) and functions which return the iterators to the interface which will be provided in go with multiple return values at once(two iterators).(2 days)
- Document Enquire class and provide small code examples (2 days).

July 6th-11th :

- Rewrap the Stem class and Writable Database class and provide errors as return values for the functions which throw.(3 days)
- Code for constructors and overloaded functions becuase of rewrapping.(2 days)
- Document both the classes.(1 day)

July 13th-18th :

- Rewrap the TermGenerator class and provide the type conversions for enum.(3 days)
- Code for constructors and overloaded functions becuase of rewrapping.(2 days)
- Document the TermGenerator class.(1 day)

July 20th-25th:

- Support Range Processors and Field Processors.(2 days)
- Document both the Classes and demonstrate the usage of derived classes.(2 days)
- Review the work done so far and fix the bugs if any.(1 day)
- Provide tests far classes implementend so far with go test(package) and edit the Makefile for adding new tests. (1 day)

July 27th:

- Phase 2 Evaluation.

Third Month :

August 3rd-8th :

- Support for Exapand Decider, all the derived classes of Exapand Decider are embedded inside another interface provided by swig.(4 days) In go lang inheritence is achieved by embeddeding one interface in another.
- Document for usage of Exapand Decider and its derived classes.(2 days)

August 10th-15th :

- Since erros are values and they would be returned to the respective function so far, now code fore generating exceptions automatically.(3 days)
- Provide the standard examples in docs (simple index, simple search).(3 days)

August 17th-22nd :

- Implementation of go get feature.(3 days)
- Providing tests for classes implementend so far with go test(package) and edit the Makefile for adding new tests and provide tests for go get.(3 days)

August 24th-31st [Final Week.]

- Evaluation Week and submission.

Stretch Goals:

- If time permits support for Stem Implementation , Stopper class , Weight class.

- Provide benchmarking tests in Go (with help of go test package).
- Research on Providing bindings in Rust.

Project Deliverable:

- A well documented Xapian bindings in Go with automated tests, examples and go get command for installation of bindings in Go.

2.4 Previous Discussion of your Project

So far we have discussed mainly about the build system of xapian for bindings in go. And a feature, go get to install xapian-bindings.I have made a PR (https://github.com/xapian/xapian/pull/292) where one of my mentor (olly) suggested for somechanges and I have done most of it.

2.5 Licensing of your contributions to Xapian

Do you agree to dual-license all your contributions to Xapian under the GNU GPL version 2 and all later versions, and the MIT/X licence?

For the avoidance of doubt this includes all contributions to our wiki, mailing lists and documentation, including anything you write in your project's wiki pages.

Yes, I agree.

2.6 Use of Existing Code

If you already know about existing code you plan to incorporate or libraries you plan to use, please give details.

I am using the existing xapian-headers.i and xapian-head.i from xapian.

References :

Iterator Pattern used in Golang Standard library - https://golang.org/pkg/container/list/

Iterator Pattern (for-range) - http://www.golangpatterns.info/object-oriented/iterators

Iterator using channels - https://github.com/srinivasyadav18/xapian-gsoc-plan/blob/master/channel.go#L35

Errors in Golang - https://blog.golang.org/error-handling-and-go

Swig Support for Golang - http://www.swig.org/Doc4.0/Go.html

Swig Insertion of Additional Go code into wrapper - http://www.swig.org/Doc4.0/Go.html#Go_adding_additional_code

Cgo documentation - https://golang.org/cmd/cgo/

Go Testing package - https://golang.org/pkg/testing/

Go doc tool - https://blog.golang.org/godoc