https://github.com/akio/rosgo.git

Go言語によるROSプログラミング

自己紹介

自己紹介

- Akiyoshi(Akio) Ochiai
- Sr Engineer @ Toyota Research Institute 

- モバイルマニピュレーションロボットの研究開発に従事
- GitHub: http://github.com/akio
- 今日は個人的なOSS活動の話をします

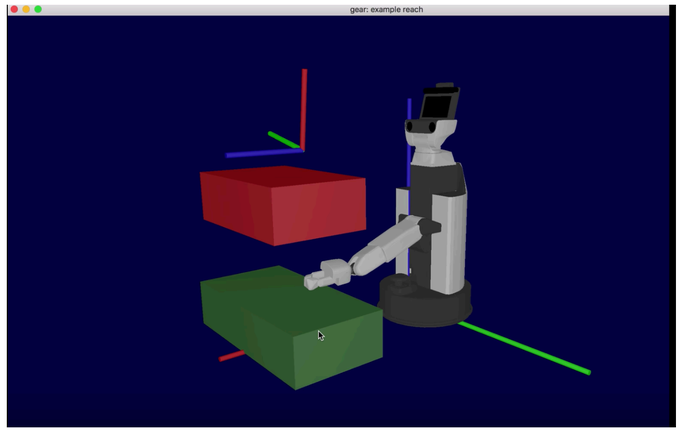
ROS関係で作ったもの

- **rosrb**
  - ROSのRubyクライアントライブラリ
  - 当時同じ会社にいた@OTL氏とかぶった上に出遅れてボツ
- **vim-ctrlp-ros**
  - Vim Ctrl-P ROS プラグイン
  - msg/srv/actionなどをVimからインクリメンタルサーチ
- **RosCraft**
  - MinecraftをROSから制御できるぞ
  - 何がしたかったのかよくわからない
- **fluent_logger_ros**
  - ROSのログをfluentdに流し込む小品
- **mask_rcnn_ros**
  - Advent Calendarのネタのために開発
- **menoh_ros**
  - ONNXをROSの画像処理パイプラインにドロップイン
- **rosgo**
  - 今回発表する内容
- **その他活動**
  - MoveIt! へのPR、メンテナー会議参加など
  - ROSコアパッケージへのPRなど

公開ROSお仕事

- 開発チームの協力のもと、トヨタHSRのロボットモデルファイルをオープンソース化しました

- hsr_description
  - https://github.com/ToyotaResearchInstitute/hsr_description
- hsr_meshes
  - https://github.com/ToyotaResearchInstitute/hsr_meshes



ROS以外での活用例

https://github.com/OTL/gear

Why Go?

動機

- 2013当時、ROS開発人材の育成に悩んでいた
  - C++

    - 複雑な言語仕様
    - Python/Matlabなどから来た人のハードルが高い
    - C言語経験者が新概念を学ぶのも難しい（OOP、ジェネリックプログラミング等）
  - Python

    - 学習コスト低い
    - 性能が必要になると結局C/C++/Cythonなどを書かざるえなくなったりする

覚えやすい

そこそこ速い

ある程度リアルタイム性のある

プログラミング言語はないか？

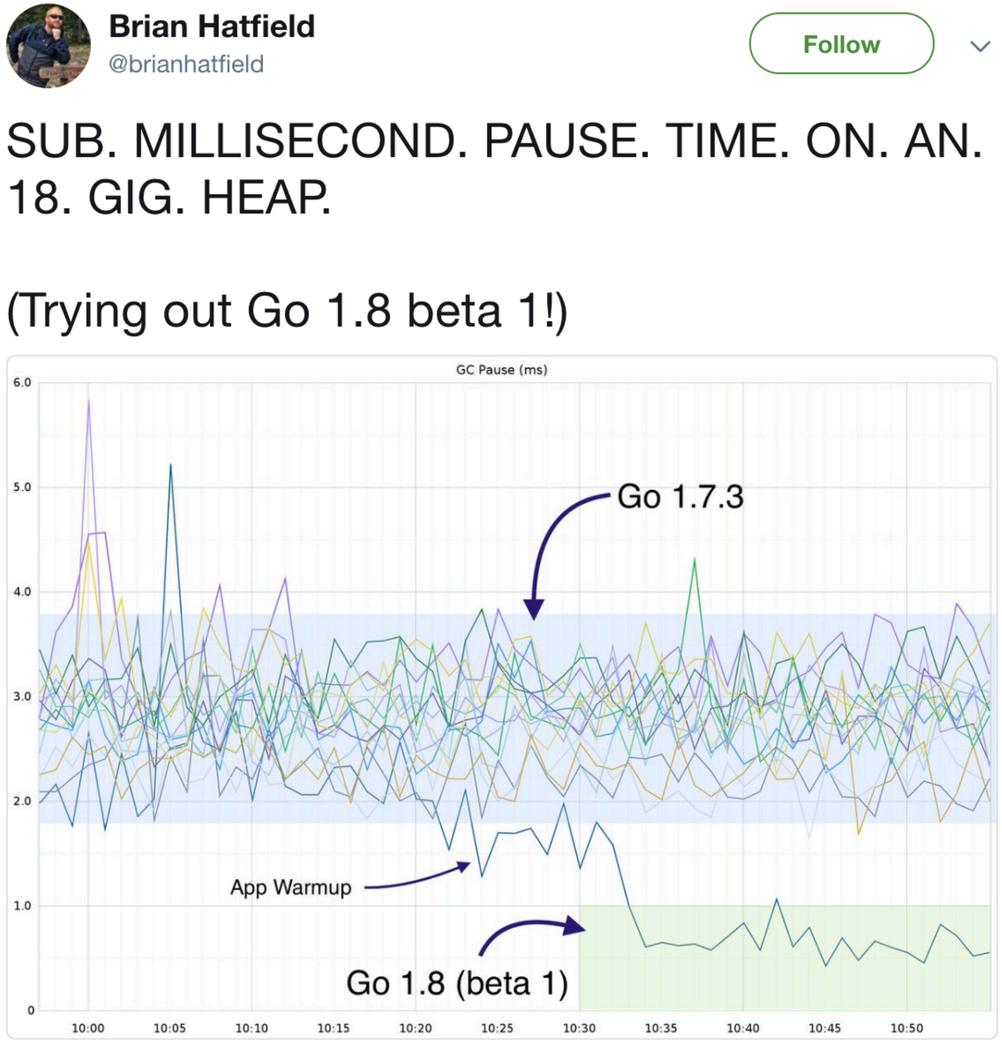
２０１３当時の候補比較

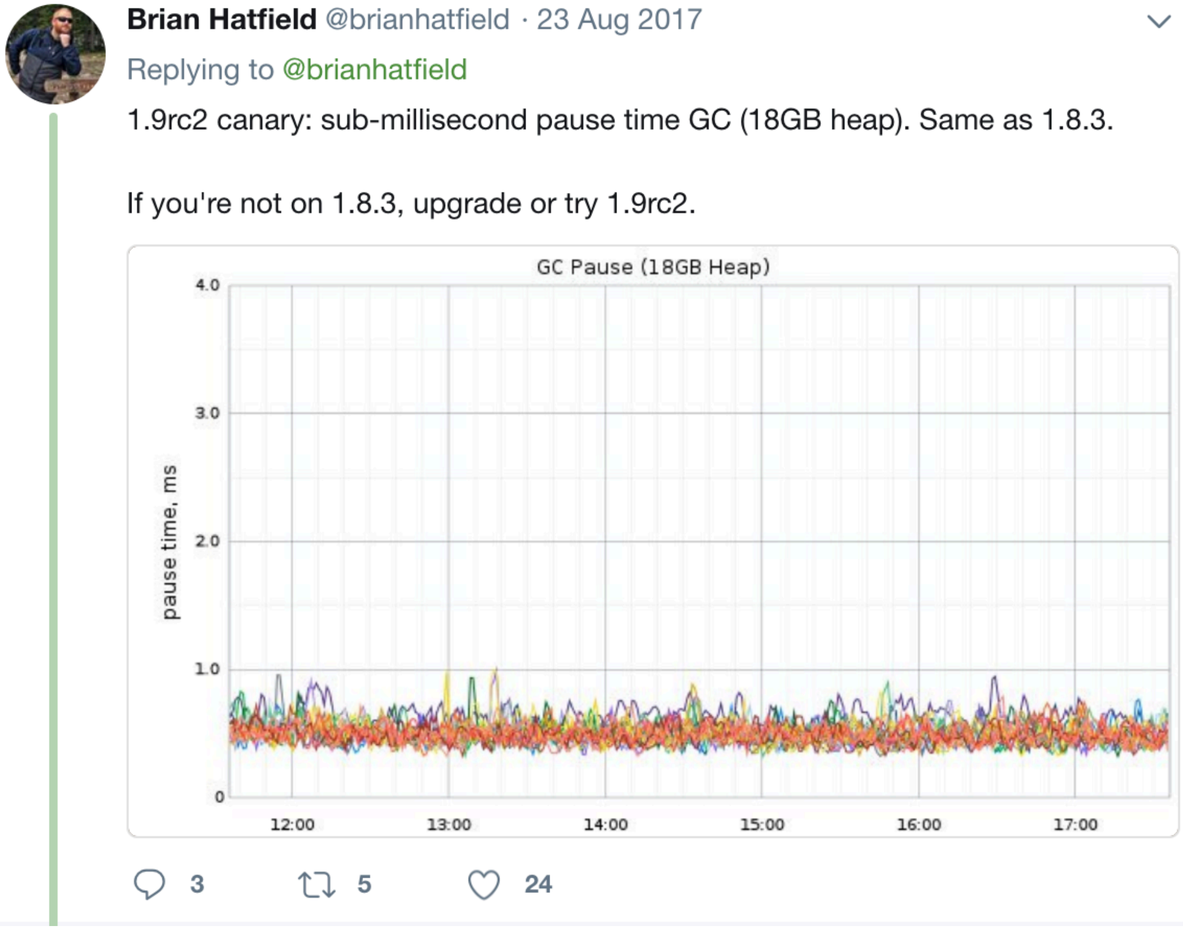
- Rust
  - 速い、安全、気持ちいい
  - 自分が求めていたものが大体ある
  - ありすぎてやっぱり学習コストが高いのでは…？
- Julia
  - 書きやすくて速くて最強に見える
  - でも当時できたばかりで色々機能が足りなかった
  - 並列処理がつらそう
- Java
  - rosjavaがすでに存在
  - ネイティブコンパイルできない
  - パッケージ配布がつらい
- **Go**
  - Better C 的な文法
  - 機能が絞られ覚えることが少ない
  - ネイティブコンパイルで結構速い
  - GCがメモリの世話をしてくれる

その他の強み

- 強力な開発体制（by Google)
- 低レイテンシなガベージコレクション（後述)
- 豊富な開発実績(Dockerなど)
- デフォルトで静的リンクされたシングルバイナリを生成
  - ツール開発などに向く
  - デプロイが簡単

サブミリ秒GC





出典：https://twitter.com/brianhatfield/status/804355831080751104

弱み

- メモリを大量に確保・解放するような使い方をするとスループットが悪化
- 科学技術計算用のライブラリが少ない（少しずつ増えてきた、後で紹介）
- 演算子オーバーロードがない（↑にちょっと影響）
- バイナリサイズが大きくなりがち

GoでROSプログラミングできるようにしよう

rosgo 初期開発版(2013)

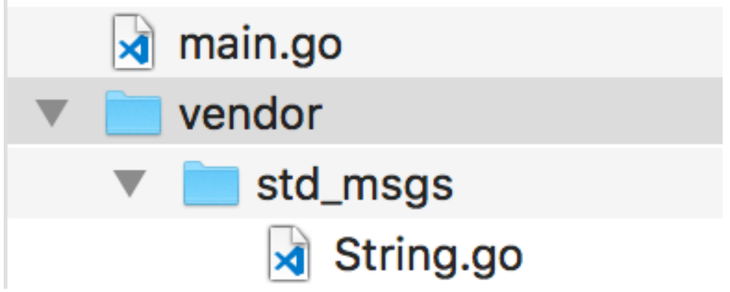
- がんばりポイント
  - XMLRPCライブラリがなかったので自分で実装した
  - Catkinツールチェーンに統合

最近のアップデート(2018)

- Remappingを実装
  - クライアントライブラリの機能が一通りそろった
  - v1.0.0の準備が整った？
- 独立したmsg/srvコードジェネレータを実装
- Good bye, Catkin and CMake!
  - Go製のコードジェネレータを実装して Pure Go パッケージ化
  - Go generateとvendoringを利用（後述）

メッセージ生成

- Goの機能 go generate (v1.4+) と vendorng (v1.5+)を活用
- ソースコード中に以下のように書いておく
  - //go:generate gengo msg your_pkg/MsgType
- $ go generate <GO_PKG_NAME> を実行
- すると、$GOPATH/your_pkg/vendor以下にGoコードが生成



rosgo コード例

インストール

1. Goのインストール

2. `$ go get -u github.com/akio/rosgo/ros`

3. `$ go install github.com/akio/rosgo/gengo`

4. `import “github.com/akio/rosgo/ros"`

Publisher

```
package main

//go:generate gengo msg std_msgs/String
import (
  "fmt"
  "github.com/akio/rosgo/ros"
  "os"
  "std_msgs"
  "time"
)

func main() {
  node, _ := ros.NewNode("/talker", os.Args)
  defer node.Shutdown()
  pub, _ := node.NewPublisher("/chatter", std_msgs.MsgString)

  for node.OK() {
    node.SpinOnce()
    var msg std_msgs.String
    msg.Data = fmt.Sprintf("hello %s", time.Now().String())
    pub.Publish(&msg)
    time.Sleep(time.Second)
  }
}
```

Subscriber

```
package main

//go:generate gengo msg std_msgs/String
import (
  "fmt"
  "github.com/akio/rosgo/ros"
  "os"
  "std_msgs"
)

func callback(msg *std_msgs.String) {
  fmt.Printf("Received: %s\n", msg.Data)
}

func main() {
  node, _ := ros.NewNode("/listener", os.Args)
  defer node.Shutdown()
  node.NewSubscriber("/chatter", std_msgs.MsgString, callback)
  node.Spin()
}
```

Example: Service Server

```
package main

//go:generate gengo srv rospy_tutorials/AddTwoInts
import (
  "fmt"
  "github.com/akio/rosgo/ros"
  "os"
  "rospy_tutorials"
)

func callback(srv *rospy_tutorials.AddTwoInts) error {
  srv.Response.Sum = srv.Request.A + srv.Request.B
  fmt.Printf("%d + %d = %d\n", srv.Request.A, srv.Request.B, srv.Response.Sum)
  return nil
}

func main() {
  node, _ := ros.NewNode("server", os.Args)
  defer node.Shutdown()
  server, _ := node.NewServiceServer("/add_two_ints", rospy_tutorials.SrvAddTwoInts, callback)
  defer server.Shutdown()
  node.Spin()
}
```

Service Client

```
package main

//go:generate gengo srv rospy_tutorials/AddTwoInts
import (
  "fmt"
  "github.com/akio/rosgo/ros"
  "os"
  "rospy_tutorials"
)

func main() {
  node, _ := ros.NewNode("client", os.Args)
  defer node.Shutdown()

  cli, _ := node.NewServiceClient("/add_two_ints", rospy_tutorials.SrvAddTwoInts)
  defer cli.Shutdown()

  var srv rospy_tutorials.AddTwoInts
  srv.Request.A = 1
  srv.Request.B = 2
  _ = cli.Call(&srv)
  fmt.Printf("%d + %d = %d\n", srv.Request.A, srv.Request.B, srv.Response.Sum)
}
```

Parameters

```
package main

import (
   "fmt"
   "github.com/akio/rosgo/ros"
   "log"
   "os"
)

func main() {
   node, _ := ros.NewNode("/test_param", os.Args)
   defer node.Shutdown()

   hasParam, _ := node.HasParam("/rosdistro")

   param, _ := node.GetParam("/rosdistro")

   err := node.SetParam("/test_param", 42)

   param, _ := node.GetParam("/test_param")

   err := node.DeleteParam("/test_param")

   foundKey, _ := node.SearchParam(“/rosdistro")
}
```

これから

機能追加

- Action Client/Server
- TF
- 他？

ロボット開発に使えるGoライブラリの紹介

- GoCV
  - OpenCVのGoバインディング
- Tensorflow
  - 公式のGoバインディングがある
- GoNum
  - Numpy的な数値計算ライブラリ
- gobot
  - 主にIoTな外部デバイスとのI/O処理ライブラリ
- g3n
  - ゲームエンジン、可視化などに使える
- GoRO
  - 色々足りないので自分で作り始めた
  - https://github.com/akio/goro
  - まだURDF parser くらいしかない

Go 2

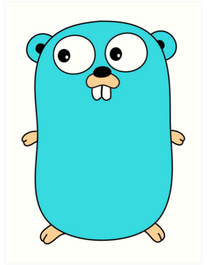
- Go言語のカンファレンス GopherCon2018にて、Go 2のドラフトが発表
  - エラー処理文法の改良
  - ジェネリクスの追加

ROS2

- DDSを実装するのは大変そう
- rclをcgoでラップする？

まとめ

- Go言語で使えるROSクライアントライブラリを開発した
- 平成が終わる前に発表できて良かった



Thank you!

https://github.com/akio/rosgo.git