Expressing classic synchronization Problems in Google Go

Harald Andre Veen

Institute for Computer Science, FU Berlin

18. April 2013

Motivation

- Cores vs Ghz
- Concurrent programming much harder than sequential programming
- Go is a new programming language with a focus on concurrency

Problem

"Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines."
– The Go Documentation

1. Does the claim hold?
2. Is Go an ideal Language to implement classic synchronization Algorithms?

Go's Memory model

- Go's memory model is not very deterministic
- go keyword (new goroutine)

```
1  var ( i, j int //initialized to 0 )
2
3  func update () {
4  i = 1; j = 2; i = j
5  }
6
7  func main() {
8  go update()
9  fmt.Println(i, j)
10 }
```

Possible outputs: '2 2', '0 0', '0 2', '1 0', '1 2'

Go Concurrency 1/2 - Communicating by messages

- <- operator
- chan keyword (channels)
- select keyword

```
1  var ch := make(chan int)
2
3  func update(i int) {
4  i = 1;   j = 2;   i = j
5  ch<-1
6  }
7
8  func main() {
9  go b()
10 rcvd:=<-ch
11 fmt.Println(i,j)
12 }
```

Possible outputs: '2 2'

Go Concurrency 2/2 - Shared memory

- no operations are atomic
- external synchronization
  - 'sync' package
  - 'sync/atomic' package

Examples:

```
1 sync.Mutex me
2 me.Lock(); me.Unlock()
3 atomic.AddInt32(&i,j)
4 atomic.CompareAndSwapInt32(&i,1,2)
5 atomic.StoreInt32(&i,1)
```

Iterative process. For each algorithm:

1. Describe requirements and
2. Program the algorithm
3. Evaluate results and experiences
4. if Algorithm can be improved or other version needed goto 1
5. Evaluate whole process

Szymanski's Algorithm - Pseudocode

- shared memory
- n processes
- 4 flags

```
1  for each process Pi 0<=i<=n-1
2  specific shared flag in 0..4
3  local integer j in 0..n-1
4  P10: flag[i]:=1;
5  P11: wait until forall j: flag[j]<3;
6  P20: flag[i]:=3;
7  P21: if exists j: flag[j]=1 then begin flag[i]:=2
8  P22: wait until exists j: flag[j]=4; end;
9  P30: flag[i]:=4;
10 P31: wait until forall j<i: flag[j]<2;
11 //critical section
12 E0: wait until forall j>1: flag[j]<2 & flag[j]>3;
13 E1: flag[i]:=0;
```

Szymanski's Algorithm

1. based on given pseudo c code
   - proven correct
   - include ugly goto syntax
2. based on pseudo code
3. why is it correct?
4. why is the runtime.Gosched() needed?
5. comparison of performance
6. potential improvements?
7. rewrite to use messages?

Suzuki-Kasami

- Message sharing and shared memory algorithm
- n nodes
- Each node stores info about all nodes
  - Number of requests received from other nodes
  - "Addresses" of other nodes
- Token shared between nodes, has token = may enter critical section
- Token contains a Queue of nodes, and and number of requests.
- Different process handles requests from nodes

Design decisions:

- Process = goroutines
- Token sent over channels
- Expandable to netbased communication

Results Overview

Problems and solutions:

- Where to put the locks?
- Receive and sending on channels are blocking operations
- Debugging multiple goroutines
- `runtime.Gosched()` and `runtime.GOMAXPROCS(n)`
- Modifying algorithms from sharing memory to communicating messages
- Buffered channels
- Packages does not provide a queue

Conclusion

- Go is design to be concurrent, and is optimized for it
- Go handles shared memory and messages in a more natural way than most languages
- C code and pseudo code can quickly be translated to go code
- Ideal language to implement the classic algorithms?

And now for something completely different..

Questions?

What are Go's ancestors? What's your Quest? Is Google using Go internally? Is Graph Isomorphism in P? Do Go Programs link with C/C++ programs? Where's the fish? Why goroutines instead of threads? What do you do with witches? What's the difference between new and make? Why do we dream? What's your favourite colour? What's the airspeed velocity of an unladen swallow? What is the relationship between BQP and NP? What is the capital of Assyria? Can I stop these complaints about unused variable/import? What Is Dark Matter? Why are there no pointer arithmetic? Is this the right room for an argument?

Sources

Used directly in this presentation

- Boloeslaw K. Szymanski - A Simple Solution to Lamports Concurrent Programming Problem with Linear Wait
- http://golang.org/doc/