An Evaluation of the Go Programming Language

Simon Salloum
s1347664

What, why and how?

- **What?** Comparing performance of OpenMP and Go
- **Why?** The rise of parallel computing
- **Why?** Go vs. libraries -> general purpose vs. specificity
- **How?** Set of benchmarks to run on clusters
  - 4 sets of benchmarks: sequential, micro, component, suite
  - Different algorithms and patterns

Benchmarks

- **Sequential**
  - Binary search, bubble sort, gcd/lcm, matrix multiplication
- **Micro**
  - Broadcast, multiplex, ping, ping-pong
- **Component**
  - Amicable numbers, merge sort, mandelbrot, dot product
- **Suite**
  - Needleman-Wunsch (Bioinf.), SRAD (image processing), Particle Filter (medical imaging)

Workflow

- Github
- Focus on automation and incremental implementation
- Issue -> Implement on dev. branch -> Push -> Merge

Progress

- **Stage 1**
  - **Sequential:** implemented
  - **Micro:** implemented
- **Stage 2**
  - **Component:** halfway through
  - **Suite:** only needs translation to Go
- **Experiments:** stage one done

Ping-pong example

```c
void pingpong(int tid, int var) {
#pragma omp parallel shared(var)
  {
#pragma omp master
    {
      tid = omp_get_thread_num();
      var = 1;
    }
#pragma omp flush(var)
#pragma omp barrier

#pragma omp critical
    {
      tid = omp_get_thread_num();
      if (tid == 1) {
        var = 2;
      }
    }

#pragma omp flush(var)
#pragma omp barrier
#pragma omp master
    {
      tid = omp_get_thread_num();
      // printf("Tid: %d - value: %d\n", tid, var)
    }
  }
}
```

```go
/* Sends pings */
func ping(ping chan<- string, msg string) {
  ping <- msg
}

/* Receives pings and sends pongs */
func pong(pings <-chan string, pongs chan<- string) {
  msg := <-pings
  msg = "pong"
  pongs <- msg
}

func main() {
  copies := 2
  N, err := strconv.Atoi(os.Args[1])
  runtime.GOMAXPROCS(copies)

  pings := make(chan string, 1)
  pongs := make(chan string, 1)

  for i := 0; i <= N; i++ {
    if err == nil {
      for i := 0; i < copies; i++ {
        go ping(pings, "ping")
        go pong(pings, pongs)
        // fmt.Println(<-pongs)
      }
    }
  }
}
```

Next steps

- https://github.com/ss1891/go-parallel-benchmarks
- Github issue tracker to keep track
- No deadlines, different levels of urgency
- **Urgent:**
  - implement component and suite benchmarks
  - run second set of experiments

Timeline

- No deadlines, but rough timeline from IRP:
  - **July 15th:** Component benchmarks implemented
  - **August 1st:** Suite algorithms implemented
  - **August 5th:** Analysis of results done
  - Plotting of results and writing done incrementally and concurrently with implementation