Go--

Chen Chen, Yuyan Ke, Yang Li, Arya Lingyu Zhao

01

Go-- Overview

Imperative, statically typed

Support for concurrency

Hide pointers

Hide memory management

C-like syntax

02

Language Features

Data Types and Data Structures

Gofunction

Channel

Data Types & Data Structures

Data Types:

int: 4 bytes

float: 8 bytes

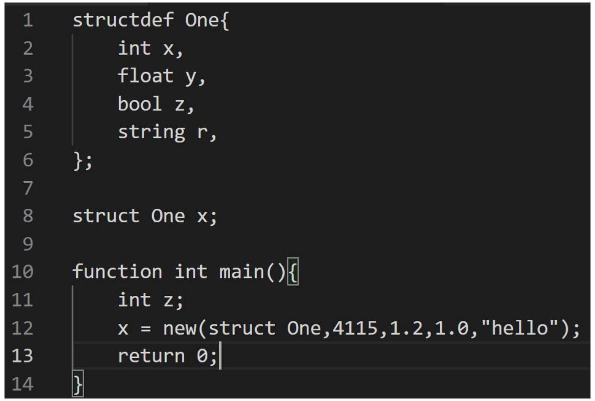
bool: 1 bit

string: char *

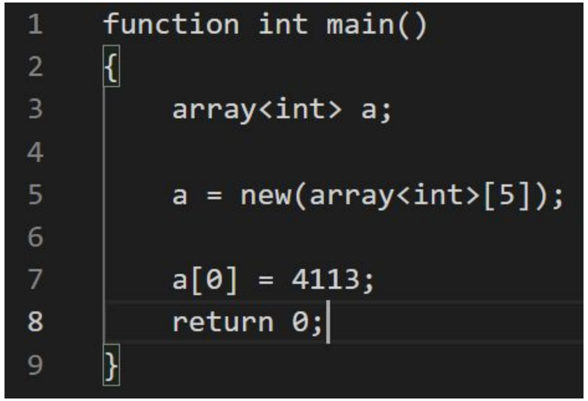
Data Structures:

Array: any 1 of data types + struct

Struct: combination of any data types



```
structdef One{
    int x,
    float y,
    bool z,
    string r,
};

struct One x;

function int main(){
    int z;
    x = new(struct One,4115,1.2,1.0,"hello");
    return 0;
}
```



```
function int main()
{
    array<int> a;

    a = new(array<int>[5]);

    a[0] = 4113;
    return 0;
}
```

Gofunction and threads

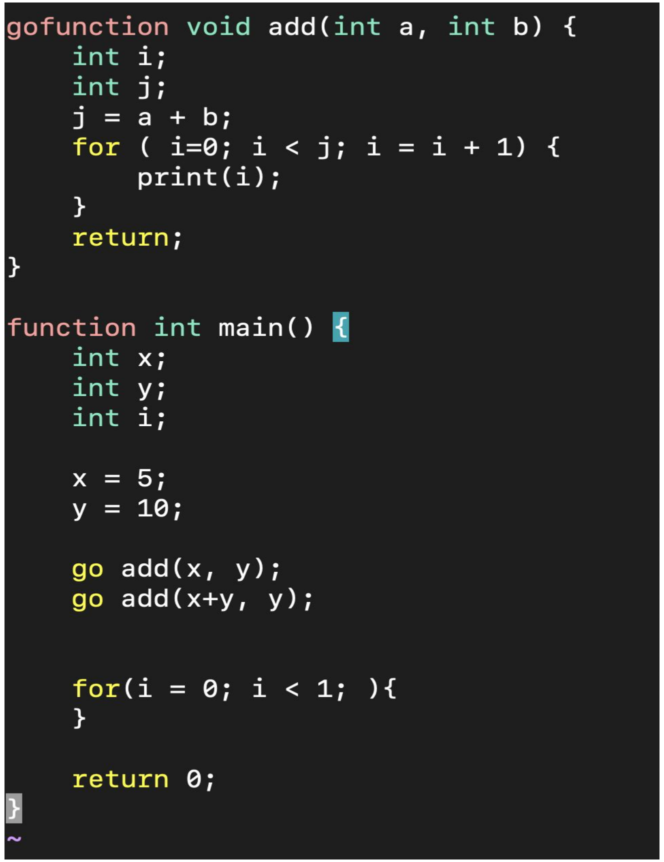
Lightweight concurrency

Thread pool implementation, initiated during first gofunction call

Same declaration as normal function with “gofunction” keyword

Calling function is easy, just like goroutine in golang.

Support up to three arguments and no return value



```
gofunction void add(int a, int b) {
    int i;
    int j;
    j = a + b;
    for ( i=0; i < j; i = i + 1) {
        print(i);
    }
    return;
}

function int main() {
    int x;
    int y;
    int i;

    x = 5;
    y = 10;

    go add(x, y);
    go add(x+y, y);


    for(i = 0; i < 1; ){
    }

    return 0;
}
~
```

Channel

Can take in data of int, float, bool, string, and user-defined structs

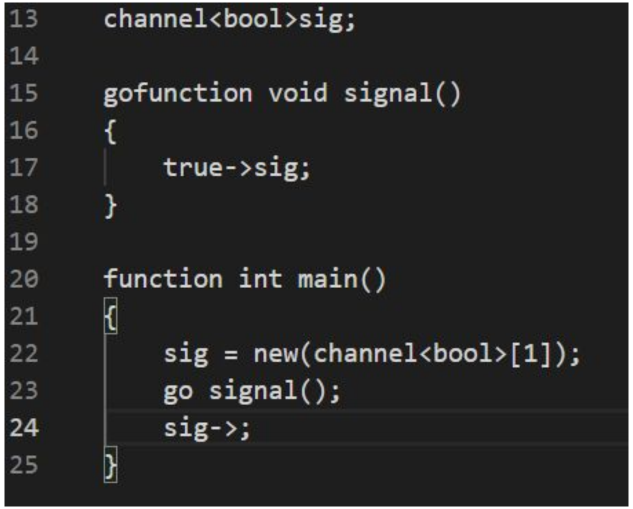
Atomic and synchronized access to channel in a multi-threaded environment

Part of Control Flow

- Blocked on dequeue when reading from an empty channel
- Blocked on enqueue when writing into a channel at full capacity

Channel capacity can be determined at runtime

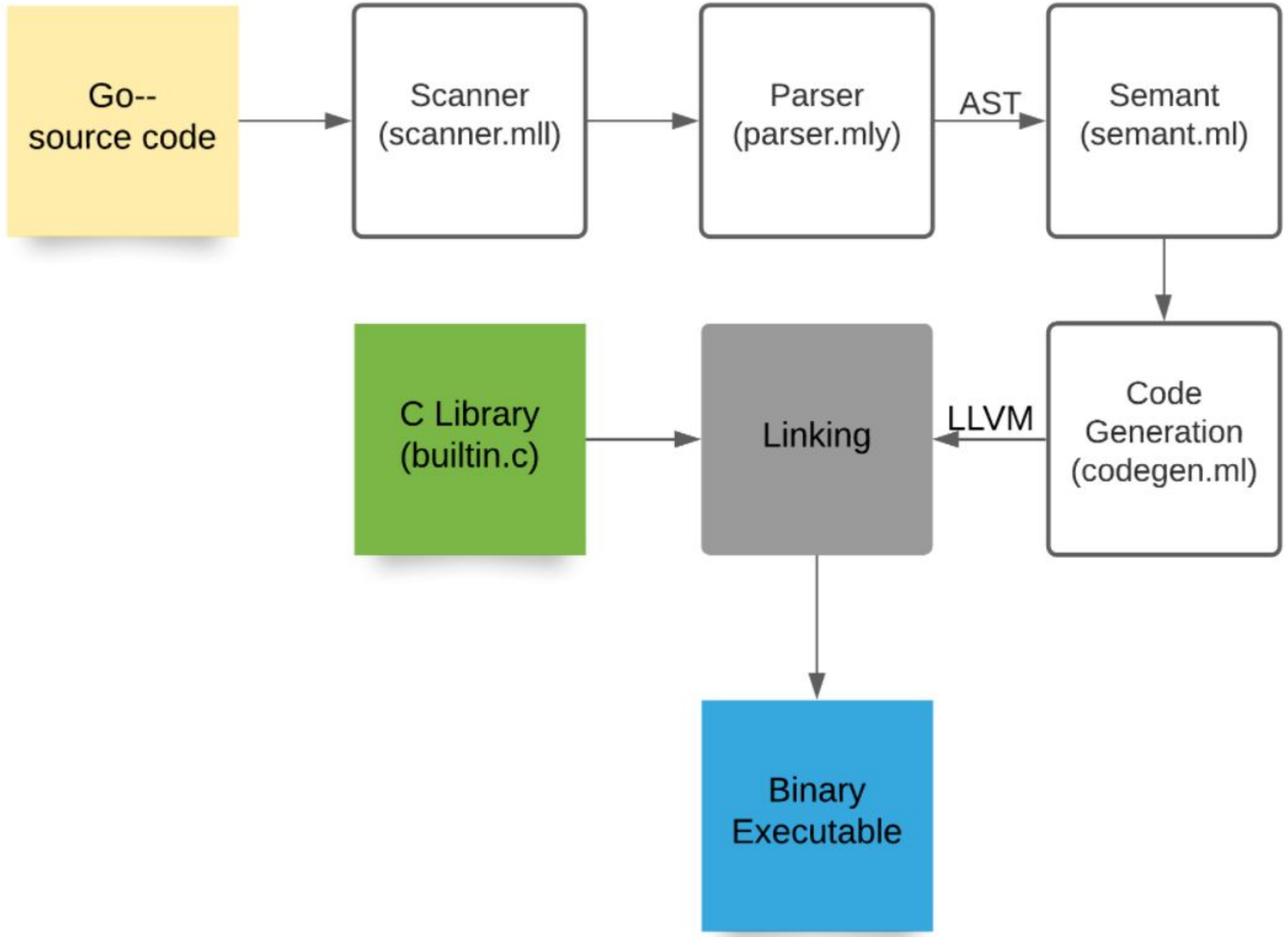
Suitable for producer-consumer paradigm



```
channel<bool>sig;

gofunction void signal()
{
    true->sig;
}

function int main()
{
    sig = new(channel<bool>[1]);
    go signal();
    sig->;
}
```

03

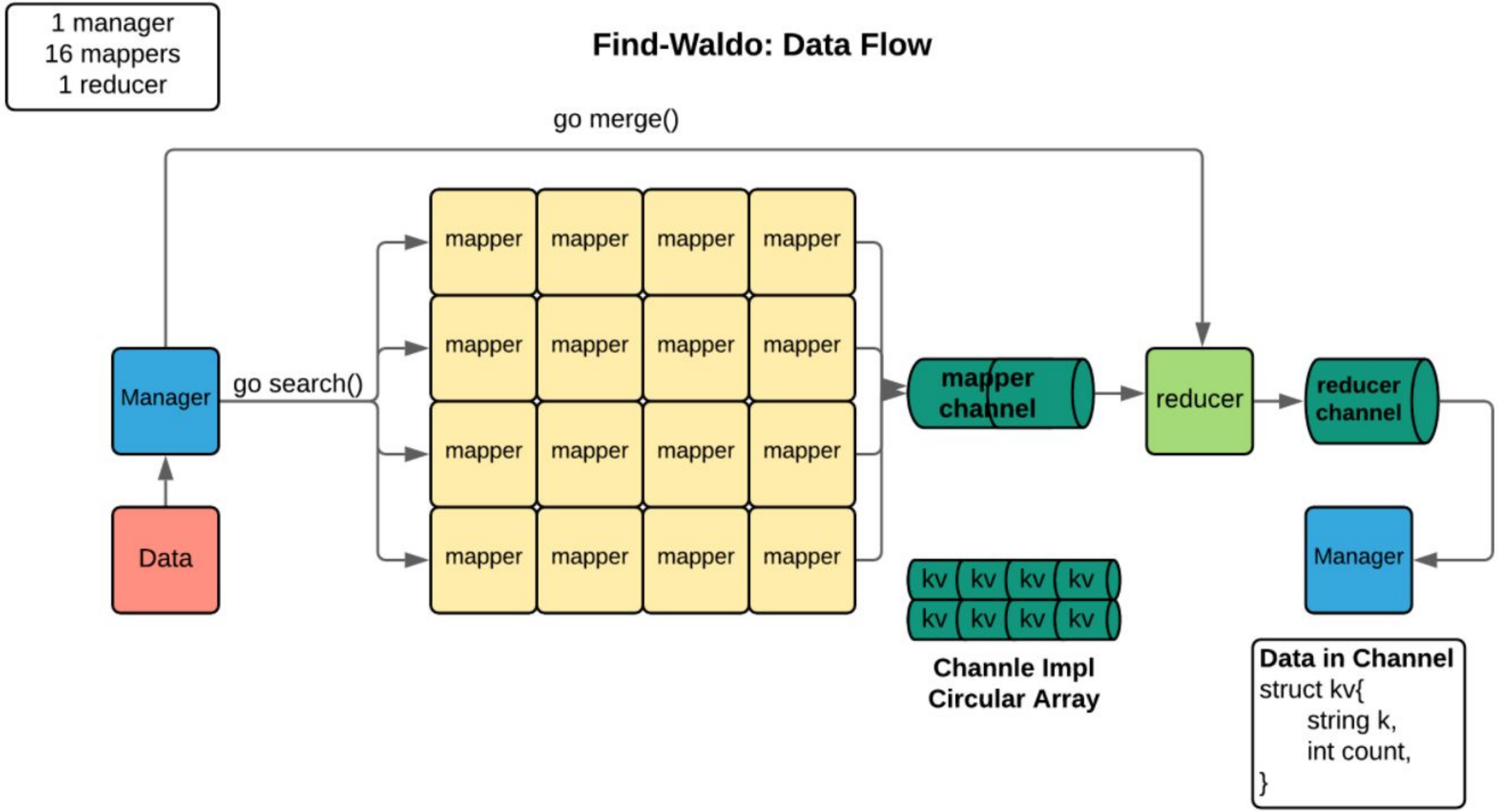
Architecture


Go--
source code
Scanner
(scanner.mll)
Parser
(parser.mly)
AST
Semant
(semant.ml)
Code
Generation
(codegen.ml)
LLVM
Linking
C Library
(builtin.c)
Binary
Executable

04

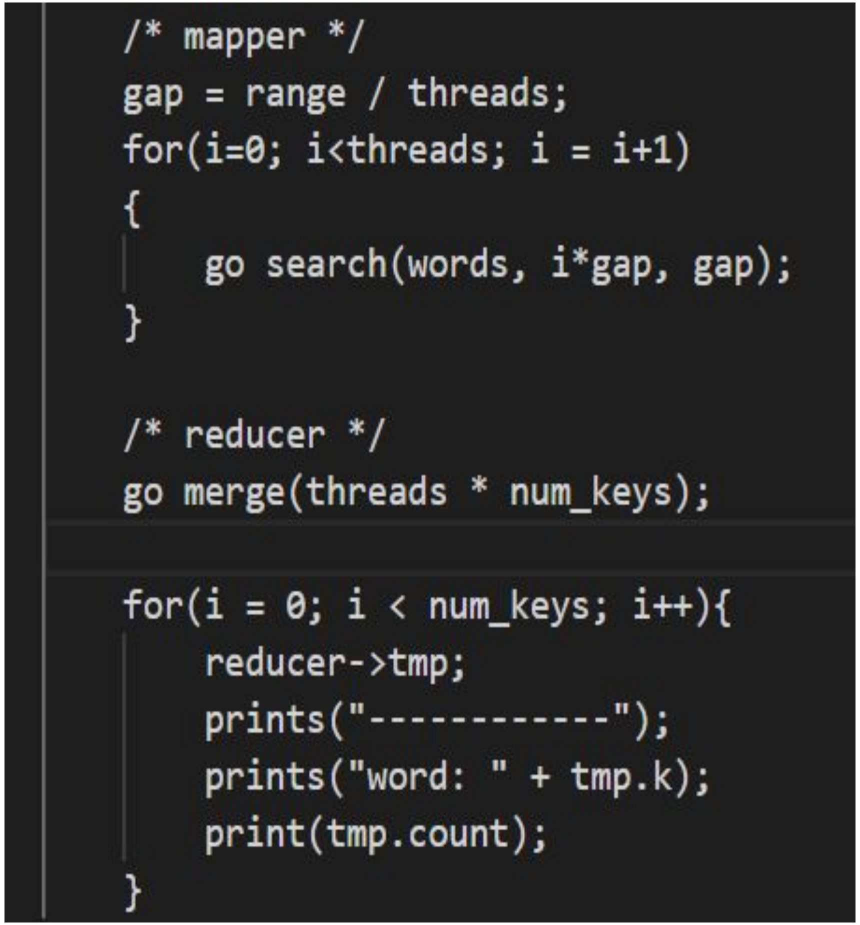
Demo

Find-Waldo: Data Flow

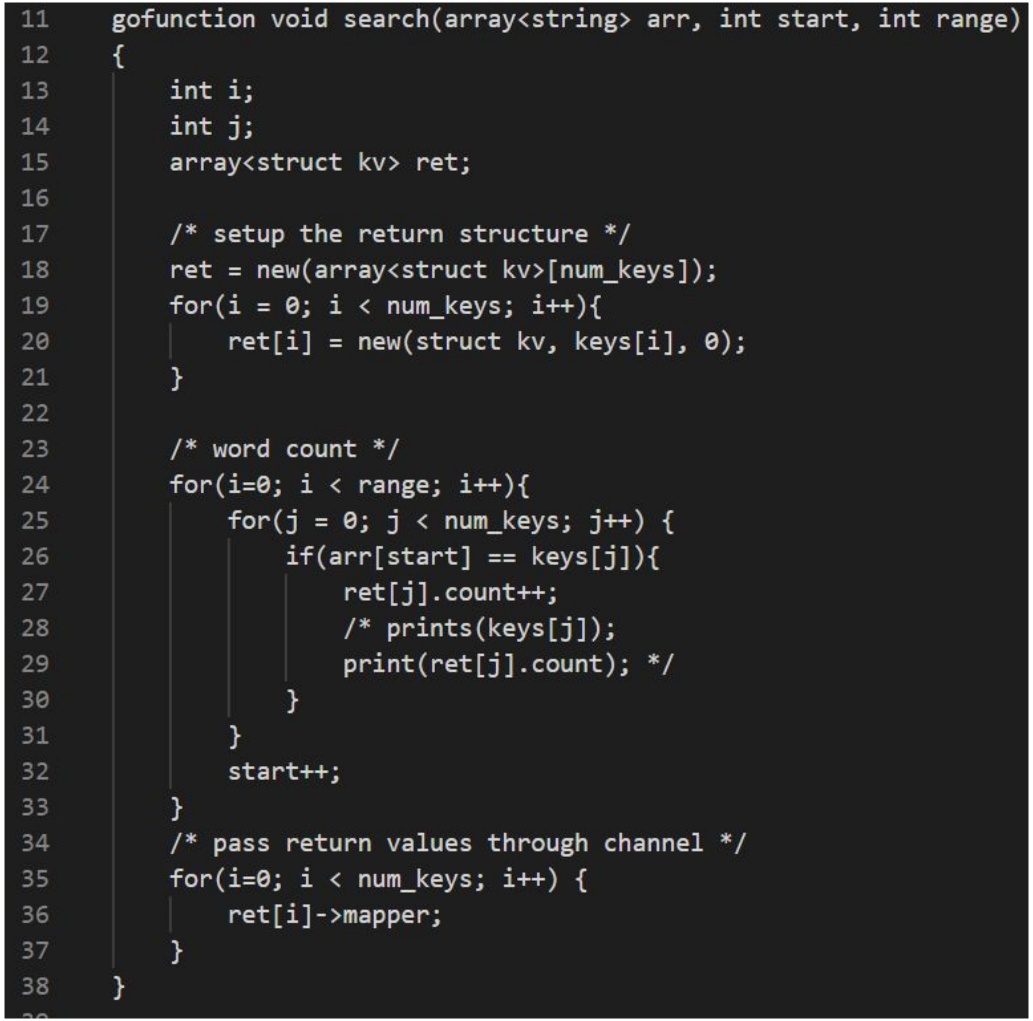


Data in Channel

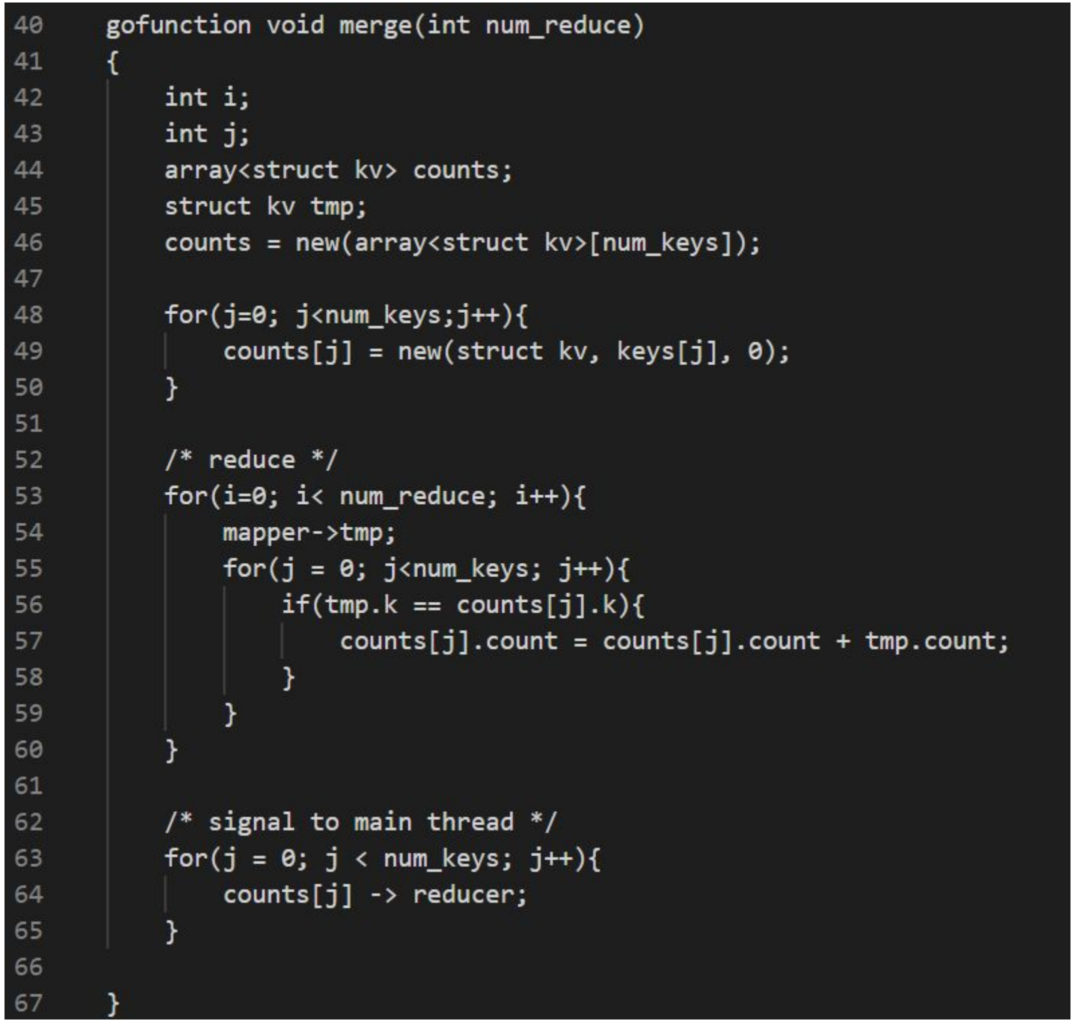
```
struct kv{
      string k,
      int count,
}
```



```
/* mapper */
gap = range / threads;
for(i=0; i<threads; i = i+1)
{
    go search(words, i*gap, gap);
}

/* reducer */
go merge(threads * num_keys);

for(i = 0; i < num_keys; i++){
    reducer->tmp;
    prints("------------");
    prints("word: " + tmp.k);
    print(tmp.count);
}
```

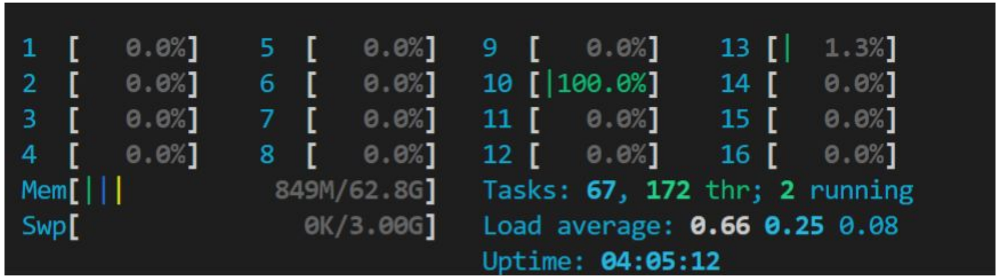


```
gofunction void search(array<string> arr, int start, int range)
{
    int i;
    int j;
    array<struct kv> ret;

    /* setup the return structure */
    ret = new(array<struct kv>[num_keys]);
    for(i = 0; i < num_keys; i++){
        ret[i] = new(struct kv, keys[i], 0);
    }

    /* word count */
    for(i=0; i < range; i++){
        for(j = 0; j < num_keys; j++) {
            if(arr[start] == keys[j]){
                ret[j].count++;
                /* prints(keys[j]);
                print(ret[j].count); */
            }
        }
        start++;
    }
    /* pass return values through channel */
    for(i=0; i < num_keys; i++) {
        ret[i]->mapper;
    }
}
```

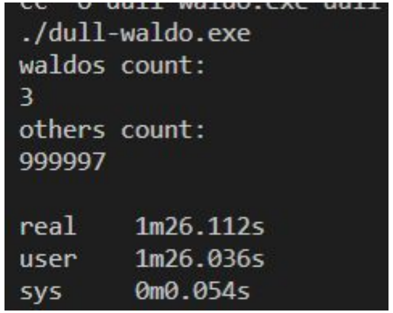


```
gofunction void merge(int num_reduce)
{
    int i;
    int j;
    array<struct kv> counts;
    struct kv tmp;
    counts = new(array<struct kv>[num_keys]);

    for(j=0; j<num_keys;j++){
        counts[j] = new(struct kv, keys[j], 0);
    }

    /* reduce */
    for(i=0; i< num_reduce; i++){
        mapper->tmp;
        for(j = 0; j<num_keys; j++){
            if(tmp.k == counts[j].k){
                counts[j].count = counts[j].count + tmp.count;
            }
        }
    }

    /* signal to main thread */
    for(j = 0; j < num_keys; j++){
        counts[j] -> reducer;
    }

}
```

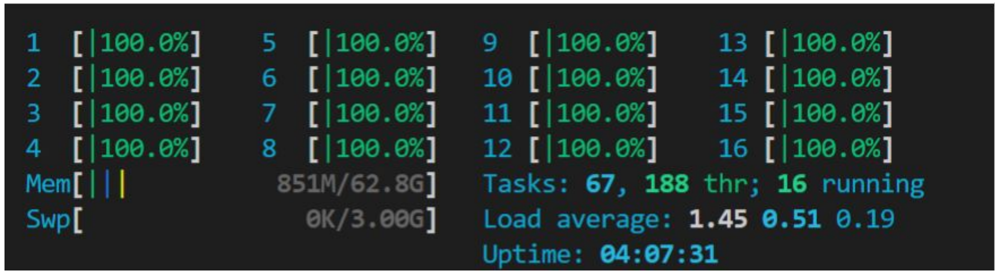
Benchmark: Find-Waldo with Fib Calculation



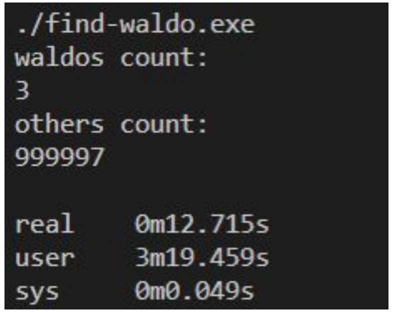
```
1  [    0.0%]    5  [    0.0%]    9  [    0.0%]    13 [|   1.3%]
2  [    0.0%]    6  [    0.0%]    10 [|100.0%]     14 [    0.0%]
3  [    0.0%]    7  [    0.0%]    11 [    0.0%]    15 [    0.0%]
4  [    0.0%]    8  [    0.0%]    12 [    0.0%]    16 [    0.0%]
Mem[|||          849M/62.8G]    Tasks: 67, 172 thr; 2 running
Swp[               0K/3.00G]    Load average: 0.66 0.25 0.08
                                Uptime: 04:05:12
```



```
[illegible]
./dull-waldo.exe
waldos count:
3
others count:
999997

real    1m26.112s
user    1m26.036s
sys     0m0.054s
```



```
1  [|100.0%]    5  [|100.0%]    9  [|100.0%]    13 [|100.0%]
2  [|100.0%]    6  [|100.0%]    10 [|100.0%]    14 [|100.0%]
3  [|100.0%]    7  [|100.0%]    11 [|100.0%]    15 [|100.0%]
4  [|100.0%]    8  [|100.0%]    12 [|100.0%]    16 [|100.0%]
Mem[|||         851M/62.8G]    Tasks: 67, 188 thr; 16 running
Swp[              0K/3.00G]    Load average: 1.45 0.51 0.19
                               Uptime: 04:07:31
```



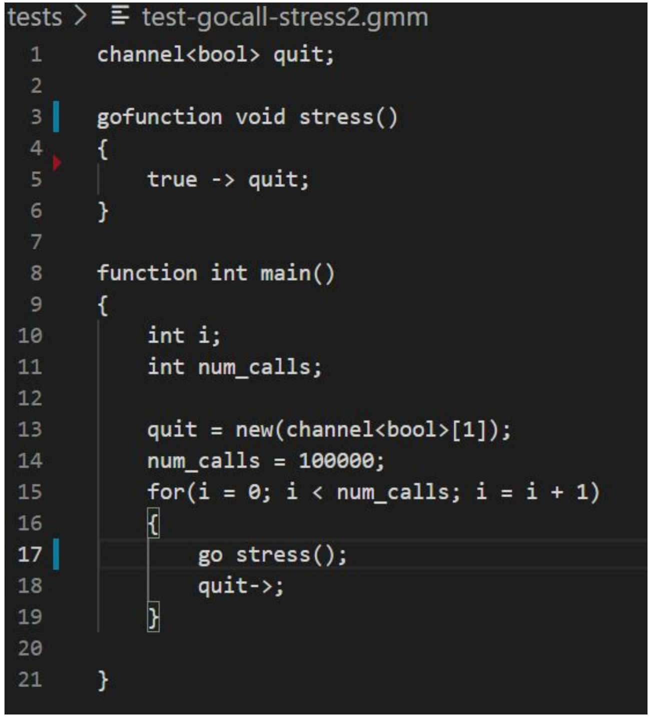
```
./find-waldo.exe
waldos count:
3
others count:
999997

real    0m12.715s
user    3m19.459s
sys     0m0.049s
```

m510: 8 Intel Xeon D-1548 at 2.0 GHz, 16 logic CPUs

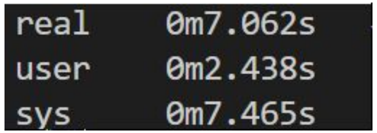
Array of 1 million elements, find-waldo with 16 gocalls, dull-waldo with 1 main function

Total loads of calculation: fib(1) to fib(62500) calculations repeated 16 times + wordcount operations

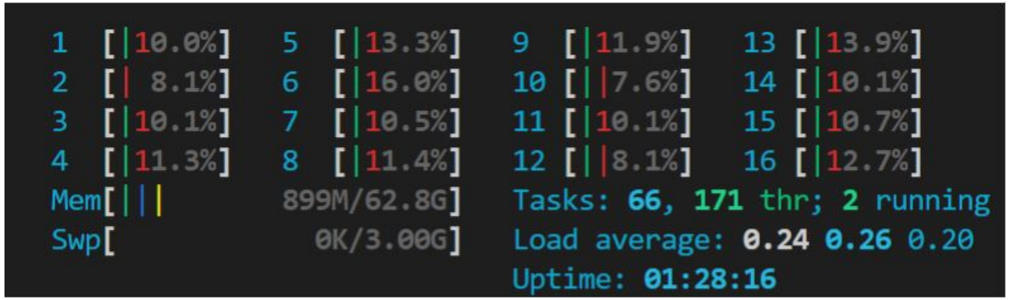
Benchmark: No Threadpool vs Threadpool



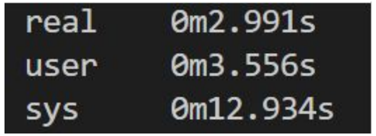
```
tests > test-gocall-stress2.gmm
channel<bool> quit;

gofunction void stress()
{
    true -> quit;
}

function int main()
{
    int i;
    int num_calls;

    quit = new(channel<bool>[1]);
    num_calls = 100000;
    for(i = 0; i < num_calls; i = i + 1)
    {
        go stress();
        quit->;
    }

}
```



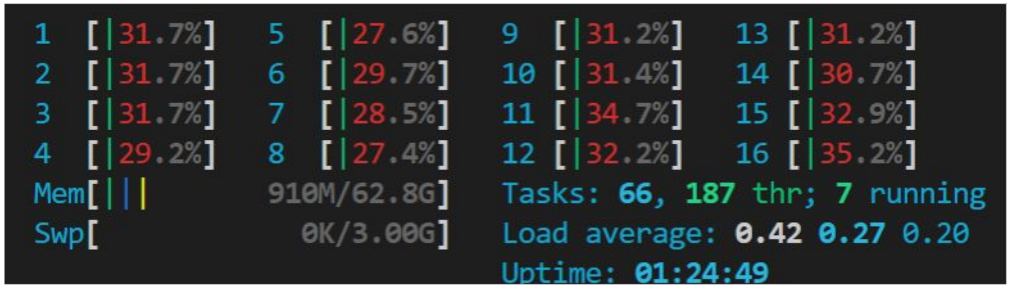
```
real    0m7.062s
user    0m2.438s
sys     0m7.465s
```



```
  1  [|10.0%]   5  [|13.3%]   9  [|11.9%]   13 [|13.9%]
  2  [| 8.1%]   6  [|16.0%]   10 [||7.6%]   14 [|10.1%]
  3  [|10.1%]   7  [|10.5%]   11 [|10.1%]   15 [|10.7%]
  4  [|11.3%]   8  [|11.4%]   12 [||8.1%]   16 [|12.7%]
  Mem[|||          899M/62.8G]   Tasks: 66, 171 thr; 2 running
  Swp[               0K/3.00G]   Load average: 0.24 0.26 0.20
                                 Uptime: 01:28:16
```



```
real    0m2.991s
user    0m3.556s
sys     0m12.934s
```



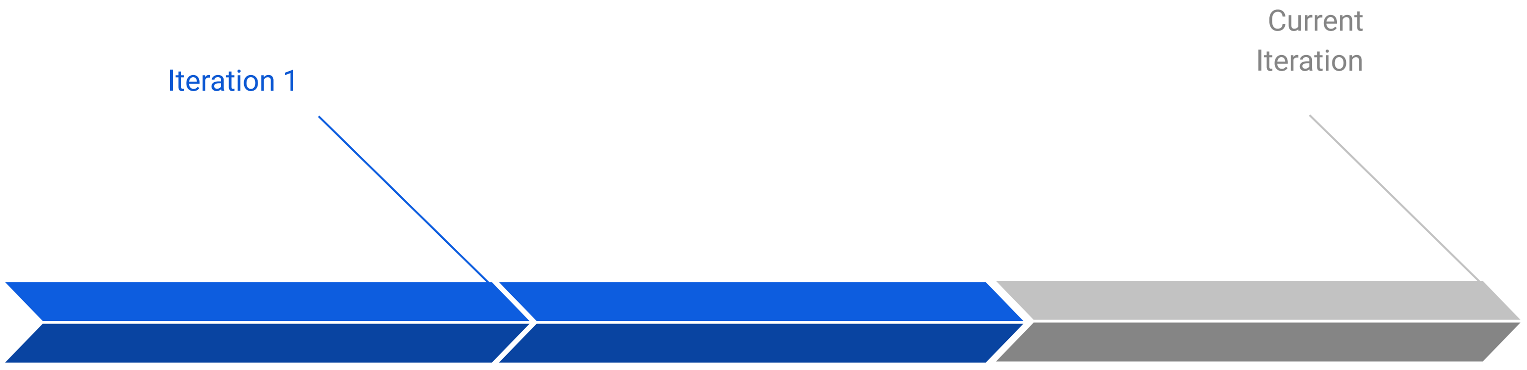
```
  1  [|31.7%]   5  [|27.6%]   9  [|31.2%]   13 [|31.2%]
  2  [|31.7%]   6  [|29.7%]   10 [|31.4%]   14 [|30.7%]
  3  [|31.7%]   7  [|28.5%]   11 [|34.7%]   15 [|32.9%]
  4  [|29.2%]   8  [|27.4%]   12 [|32.2%]   16 [|35.2%]
  Mem[|||          910M/62.8G]   Tasks: 66, 187 thr; 7 running
  Swp[               0K/3.00G]   Load average: 0.42 0.27 0.20
                                 Uptime: 01:24:49
```

m510: 8 Intel Xeon D-1548 at 2.0 GHz, 16 logic CPUs

100k operations

05

Evolution of Language



Concurrent Threads

Channels with mmap

Function types (function, gofunction)

Structs

Arrays

First-Class Functions

Concurrent threads

Function Types (function, gofunction)

Channel with malloc

Structs

Arrays

06

Future Work

Gocall

Allowing gofunction with more than three arguments

Channel and Other Data Structures

Adding a dictionary and nested dictionary/struct would be really helpful

Memory Usage

Garbage collection

Input/Output

File I/O ,reading and writing from/to stdin/stdout/stderr