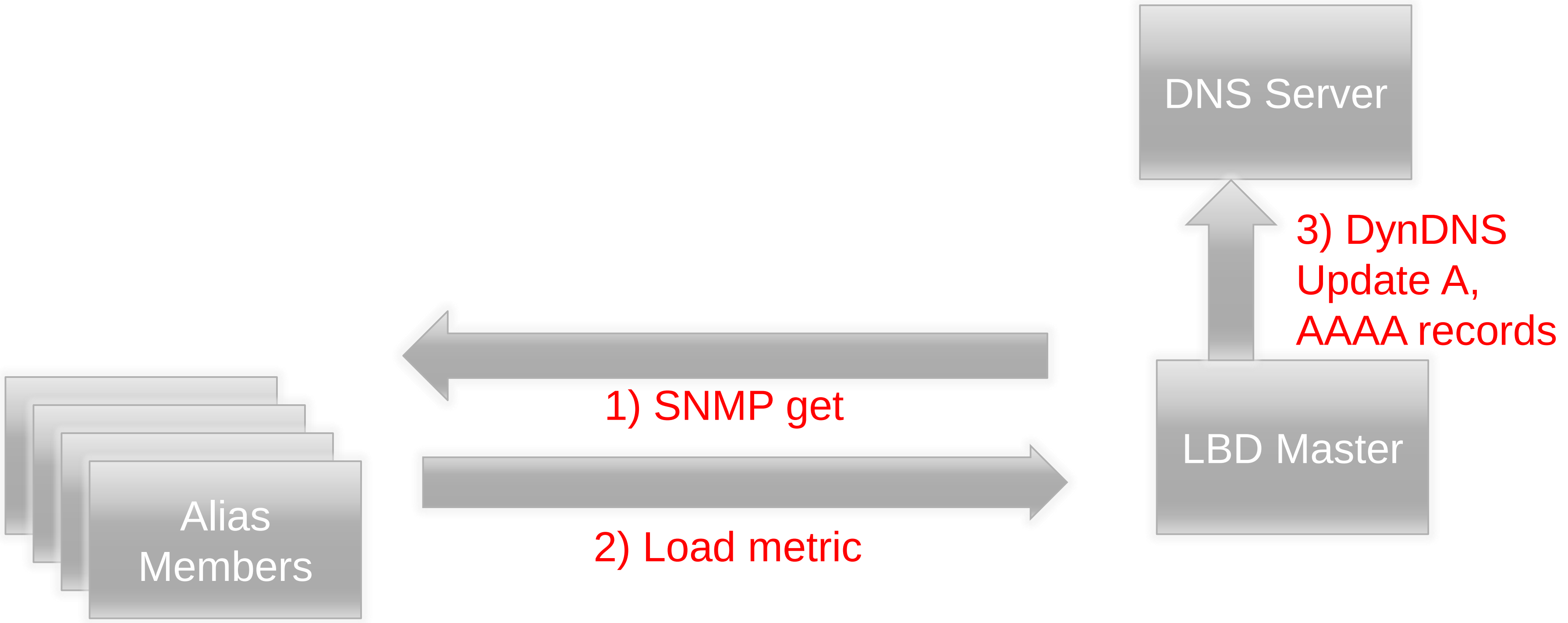
Rewriting the LBD DNS Load Balancer with concurrency in GO

Ignacio Reguero

ITLT-11 16 September 2016

LBD: The Basics



DNS Server

3) DynDNS Update A, AAAA records

1) SNMP get

2) Load metric

Alias Members

LBD Master

Current (Perl) LBD Loops on the LB Alias list

```
while (not $sig_term) {
  foreach my $cluster (values %clusters) {
    # check wether the cluster has to be updated
    if ($cluster->time_to_refresh() ) {
      $cluster->find_best_hosts(); # detects sig term automatically
      if (should_update_dns()) {
        $cluster->update_dns($config->tsig_key_prefix().'internal', $config-
>tsig_internal_key(), $config->dns_manager());
      }
    }
  }
}
```

Aliases have to be evaluated periodically

```
sub time_to_refresh {
  my $self = shift;
  return ( ($self->{time_of_last_evaluation} + $self->{parameters}{polling_interval}) <= time);
} # end sub
```

- To be able respect the ‘polling interval’ we need

Evaluation time of all LB aliases < ‘polling interval’

- This is a Scalability limit
  - Possible solutions:
    - Split (shard) the LB aliases list among several servers (and live with the complexity for operations)
    - Do a version of the LBD that evaluates the LB aliases concurrently:

To Evaluate LB Aliases concurrently considered

- Threads in Perl:
  - Although already used by LBD to do SNMP requests in parallel:
    - The use of interpreter-based threads in Perl is officially discouraged.
    - Not supported by most libraries
- Threading library in Python
  - Limited by the Global Interpreter Lock (GIL) that makes any CPU work single-threaded. Can be OK when a lot o I/O involved.
- Goroutines in Golang
  - Native to the language
  - Programming model with tools for concurrency:

  Goroutines, channels, select, GOMAXPROC…

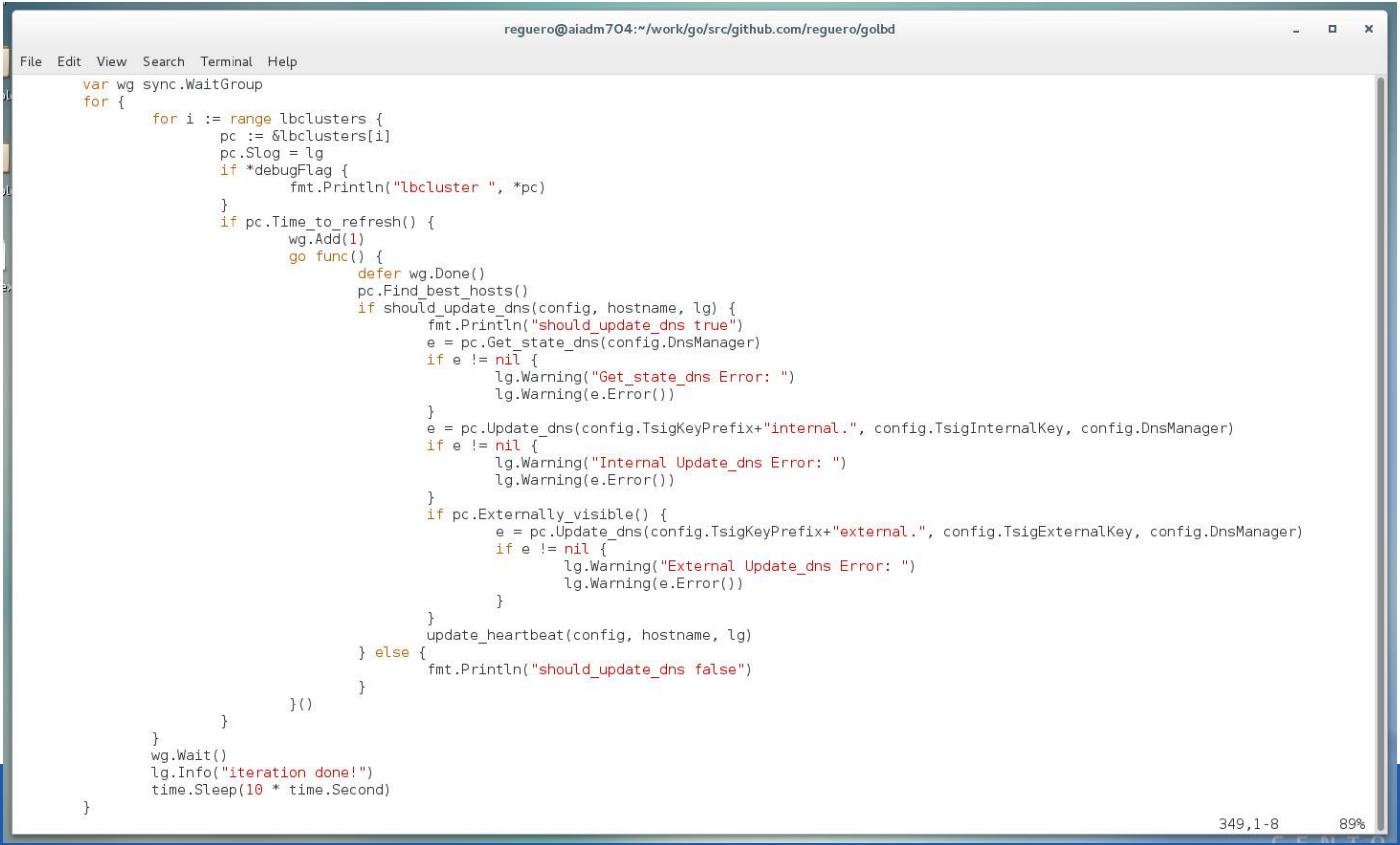
Equivalent LB Alias Loop in Go

```
for {
    for i := range lbclusters {
        pc := &lbclusters[i]
        if pc.Time_to_refresh() {
            pc.Find_best_hosts()
            if should_update_dns(config, hostname, lg) {
                pc.Update_dns(config.TsigKeyPrefix+"internal.", config.TsigInternalKey, config.DnsManager)
            }
        }
    }
}
```

Unfold the loop with a goroutine

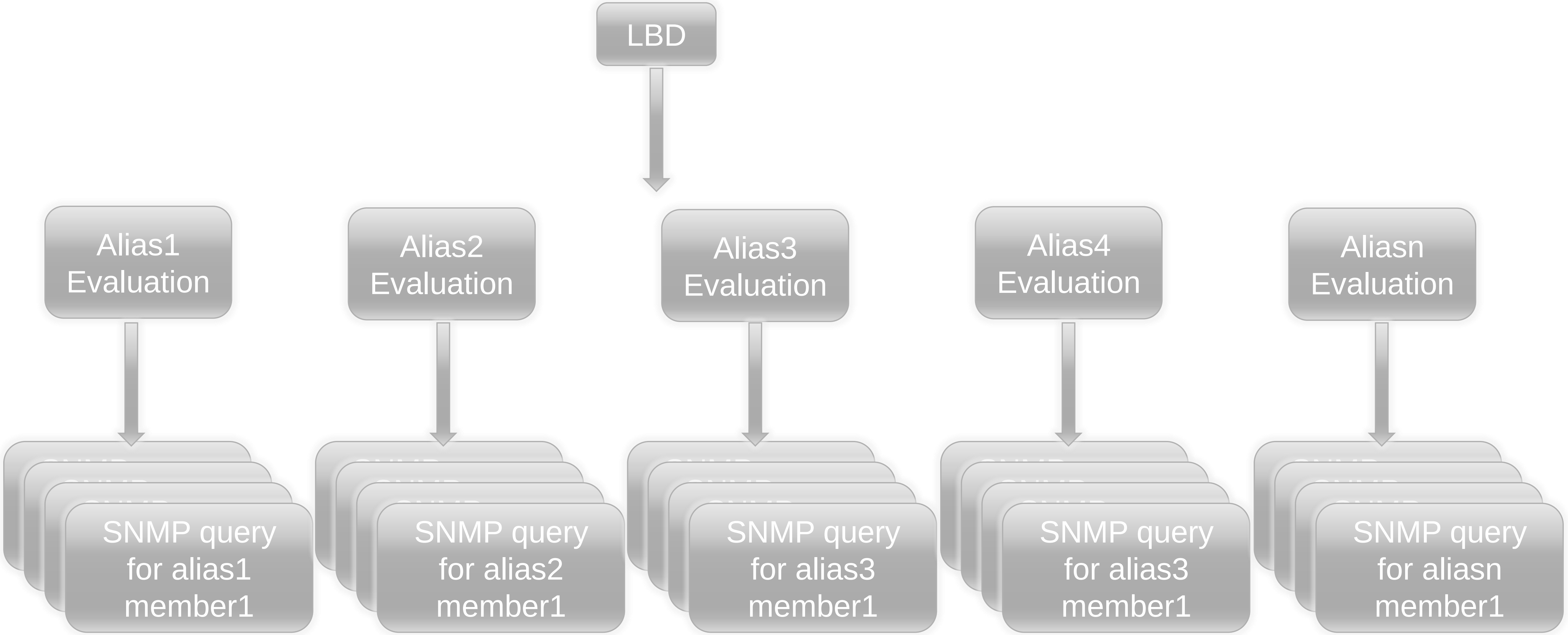
```
var wg sync.WaitGroup
for {
    for i := range lbclusters {
            pc := &lbclusters[i]
            if pc.Time_to_refresh() {
               wg.Add(1)
               go func() {
                    defer wg.Done()
                    pc.Find_best_hosts()
                    if should_update_dns(config, hostname, lg) {
                         pc.Update_dns(config.TsigKeyPrefix+"internal.", config.TsigInternalKey, config.DnsManager)
                    }
               } ()
            }
    }
     wg.Wait()
}
```

The real code



```
    var wg sync.WaitGroup
    for {
        for i := range lbclusters {
            pc := &lbclusters[i]
            pc.Slog = lg
            if *debugFlag {
                fmt.Println("lbcluster ", *pc)
            }
            if pc.Time_to_refresh() {
                wg.Add(1)
                go func() {
                    defer wg.Done()
                    pc.Find_best_hosts()
                    if should_update_dns(config, hostname, lg) {
                        fmt.Println("should_update_dns true")
                        e = pc.Get_state_dns(config.DnsManager)
                        if e != nil {
                            lg.Warning("Get_state_dns Error: ")
                            lg.Warning(e.Error())
                        }
                        e = pc.Update_dns(config.TsigKeyPrefix+"internal.", config.TsigInternalKey, config.DnsManager)
                        if e != nil {
                            lg.Warning("Internal Update_dns Error: ")
                            lg.Warning(e.Error())
                        }
                        if pc.Externally_visible() {
                            e = pc.Update_dns(config.TsigKeyPrefix+"external.", config.TsigExternalKey, config.DnsManager)
                            if e != nil {
                                lg.Warning("External Update_dns Error: ")
                                lg.Warning(e.Error())
                            }
                        }
                        update_heartbeat(config, hostname, lg)
                    } else {
                        fmt.Println("should_update_dns false")
                    }
                }()
            }
        }
        wg.Wait()
        lg.Info("iteration done!")
        time.Sleep(10 * time.Second)
    }
                                                                      349,1-8         89%
```

Go LBD Concurrency: two levels of Goroutines



LBD

Alias1 Evaluation

Alias2 Evaluation

Alias3 Evaluation

Alias4 Evaluation

Aliasn Evaluation

SNMP query for alias1 member1

SNMP query for alias2 member1

SNMP query for alias3 member1

SNMP query for alias3 member1

SNMP query for aliasn member1

Another goroutine loop

```
func (self *LBCluster) evaluate_hosts() {
        result := make(chan RetSnmp, 200)
        for h := range self.Host_metric_table {
                currenthost := h
                self.write_to_log("contacting cluster: " + self.Cluster_name + " node: " + currenthost)
                go self.snmp_req(currenthost, result)
        }
        for range self.Host_metric_table {
                select {
                case metrichostlog := <-result:
                        self.Host_metric_table[metrichostlog.Host] = metrichostlog.Metric
                        self.write_to_log(metrichostlog.Log)
                }
        }
}
```

Time to cycle through all aliases (430 aliases)

- Perl LBD
  - Over 240 secs
- Go LBD
  - Around 14 secs with one system thread
  - Between 11 and 12 secs with 8 system threads
- Please note that the LBD SNMP timeout is 10 seconds

So the Go LBD full cycle is pretty close to the worst case for a single alias:

- The parallelism is very good
- No visible scalability issue

- Any Questions ?
  - https://gitlab.cern.ch/lb-experts/lbd
  - https://gitlab.cern.ch/lb-experts/golbd
  - https://golang.org/