

Emetric

Justin Kirby



2011-03-23 Wed

follow along

If you would like to follow along

```
git clone git://github.com/justinkirby/emetriic.git
```

what it does

injects itself into an erlang node then starts dumping data

inspiration

rebar

how to make things simple

inspiration

rebar

how to make things simple

eper

- where data is hiding
- graceful code injection

os level

- memory usage
- cpu usage
- block io

erlang vm

- run queue
- reductions
- garbage collection
- context switches
- process count

mnesia

- locks
- subscribers
- transactions
 - counts
 - commits
 - failures
 - restarts
 - log writes
- checkpoints
- memory
- size

making it simple

rebar
is as simple as it gets

zip ebins

```
zip:create("mem",Files, [memory])
```

pack zip into script

```
Header= "#!/usr/bin/env escript\n%%! -noshell -noinput\n",  
Script = <<Header,ZipBin/binary>>,  
file:write_file("emetric",Script)
```

show the code

show the code

- `emetric/rebar.config`
- `emetric/priv/escripts/boot__emetric`

adding new metrics

It is trivial to plug anything into it.

```
deps() ->[emetric_hooks].  
sup() -> ?CHILD(?MODULE,worker).  
tick(Tick,Acc) ->  
    vogon_stats(Tick,Acc).
```

tagged proplist

illustration purposes.

```
{vogons, [{poems, 10},  
          {duration, infinity},  
          {avg_pain, 11},  
          {frobnications, 42}]}
```

register with hook

```
emetric_hooks:add(gather_hooks,  
  fun(T,A) -> vogons:tick(T,A) end,2),
```

show the code

- `emetric/src/emetric_stats_mnesia.erl`

into csv file

All data goes into a csv file (only built in filter atm).

vogon csv

```
vogons_tick,vogons_poems,vogons_duration,\  
vogons_avg_pain,vogons_frobnications  
1,0,infinity,11,42  
2,2,infinity,11,42  
3,4,infinity,11,42  
4,8,infinity,11,42  
5,16,infinity,11,42
```

running it

- node
- cookie
- stats to gather
- where it is going

example

```
./emetric --node=ejabberd@localhost --cookie=monster \  
  start gather=sys,mnesia scatter=file \  
  mods=emetric_filter_csv
```


connect

simple rpc ping to remote node with gotchas

no net_kernel in escript

```
net_kernel:start(['emetric@localhost',shortnames]),
```

connect

simple rpc ping to remote node with gotchas

no net_kernel in escript

```
net_kernel:start(['emetric@localhost',shortnames]),
```

epmd can be out of sync

```
global:sync(),
```

connect

simple rpc ping to remote node with gotchas

no `net_kernel` in `escript`

```
net_kernel:start(['emetric@localhost',shortnames]),
```

`epmd` can be out of sync

```
global:sync(),
```

need the same cookie

```
erlang:set_cookie(node(),Cookie),
```

Finally a ping

now we can finally ping

```
pong = net_adm:ping(Node),
```

inject

with lots of checks for sanity here

```
rpc:call(Node,code,load_binary,[Mod,Fname,Bin]).
```

inject

with lots of checks for sanity here

```
rpc:call(Node,code,load_binary,[Mod,Fname,Bin]).
```

and here

start

this was tough.

- rpc calls spawn a process
- no way to just start a linked supervisor

gen server proxy

- put a gen server between rpc and supervisor

tsung 101

- configuration is xml
- you WILL write an xml config generator/template
- it can be baroque if you want to be precise
- ...

...

but it works and works well.

tsung gotchas

watch for ulimits.

put max user counts everywhere possible

read example configs helps a lot

steps to running tsung

- need something to stree (ejabberd)
- create users
- write config

a walk through tsung

the \$0.25 tour of tsung config

now it is demo time

step by step demo

now it is demo time

step by step demo

spin up ejabberd

now it is demo time

step by step demo

spin up ejabberd

spin up emetric

now it is demo time

step by step demo

spin up ejabberd

spin up emetric

run tsung

making the data pretty

welcome to plotr, the red headed step child of emetric

to avoid the dreaded spreadsheet

I learned numpy and matplotlib

not a work of art
but it does the job

plotr

- extracts 'interesting' data (std deviation $\neq 0.0$)
- generates charts pivoting on ejd sessions
- interactive gui

plotr

just the data please

```
plotr.py --data=/tmp/emetric_node@host_date.csv \  
--out=interesting.csv
```

plotr

just the graphs please

```
plotr.py --data=/tmp/emetric_node@host_date.csv --graph
```

plotr

the ui (it is ok to laugh)

```
plotr.py --data=/tmp/emetric_node@host_date.csv --ui
```


What has this found

ulimits

do NOT store mnesia archive tables in memory

disc_only_copies is too slow

global shared rosters are bad
especially with large ldap