开源社
kaiyuanshe

2016中国开源年会

China Open Source Conference 2016

Linux性能分析工具的现状和我们的优化

Barry Song & Bob Liu & Xining Xu

Runtime resources profiling

- A variety of tools such as top, iotop, iostat, vmstat, sar oprofile, perf etc exist in Linux system; however, it is extremely difficult for people to address the problems from the raw data printed by these tools
- LEP(Linux Easy Profiling) is a web-based open-source tool suite to make them be all-in-one and visualized, and its key developers, for this moment, include

✓Baohua Song

✓Bob Liu

✓Xining Xu

✓More developers are coming(eg, Ping Liu….)

Problems in existing tools

- RAW data, it is difficult for non-experts to understand

e.g. load average…

```
top - 18:10:33 up 12 min,  2 users,  load average: 0.24, 0.40, 0.39
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
%Cpu0  :  8.9 us,  1.0 sy,  0.0 ni, 90.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  5.1 us,  0.3 sy,  0.0 ni, 94.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Problems in existing tools(cont.)

- Not visualized

e.g. how many memory used in Linux?

```
baohua@baohua-VirtualBox:~$ free
             total       used       free     shared    buffers     cached
Mem:       1079116     863188     215928       3068     171600     305600
-/+ buffers/cache:     385988     693128
Swap:       522236          0     522236
```

Problems in existing tools(cont.)

- Lack of the description for changing…

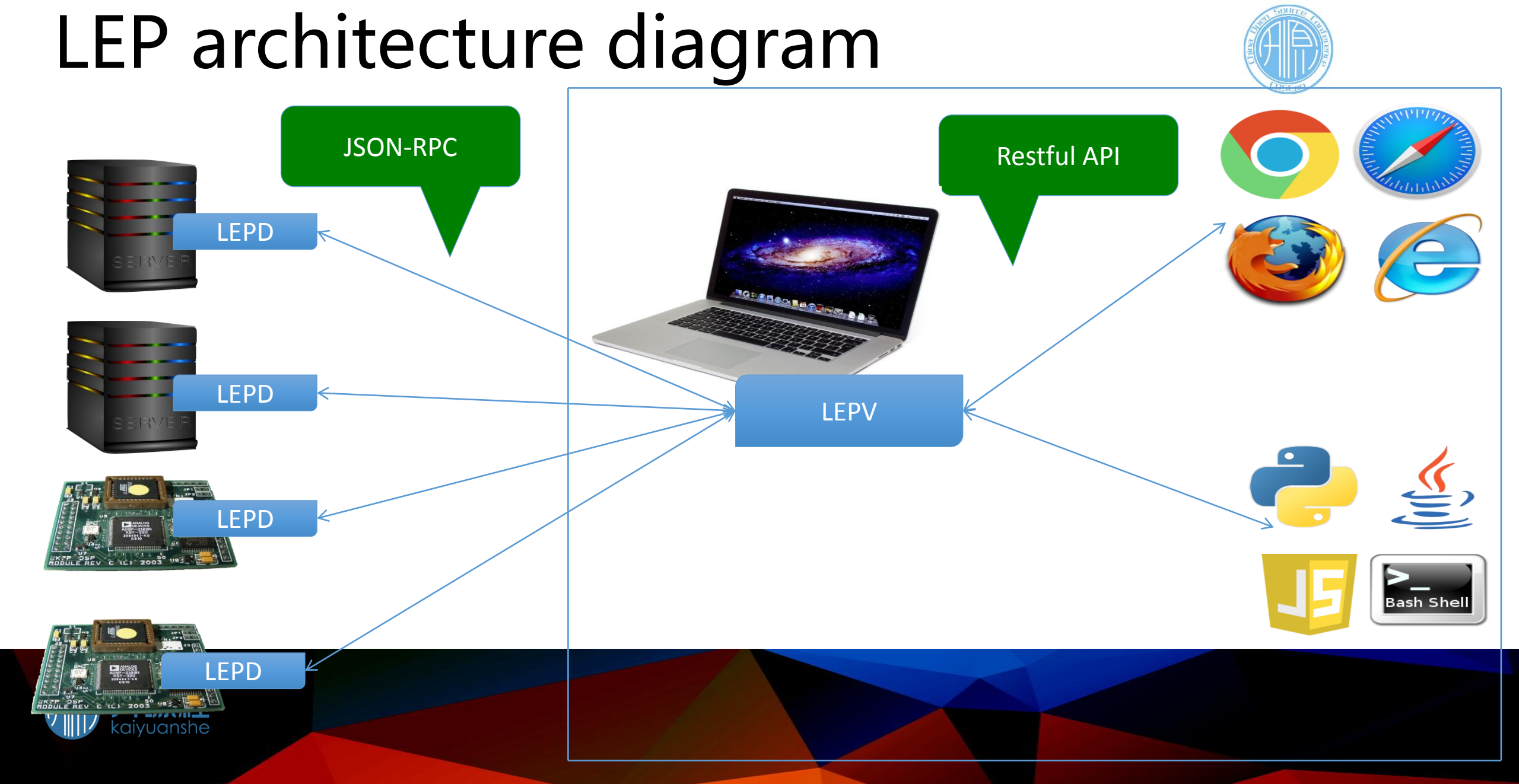
e.g. how the cpu usage is changing during a period?

```
top - 18:13:55 up 16 min,  2 users,  load average: 0.05, 0.23, 0.32
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
%Cpu(s):  4.6 us,  0.3 sy,  0.0 ni, 95.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 s
KiB Mem:   1079116 total,   869656 used,   209460 free,   171608 buffers
KiB Swap:   522236 total,        0 used,   522236 free.   305616 cached Mem
```

What we need?

- CURVED LINE
- PIE diagram
- BIN diagram
- COLORED

LEP architecture diagram



LEP Summary

- C/S
- PC / Embedded Board
- JSONRPC
- Web App
- Restful API -> Extensibility

LEPD implementation

- cJSON & jsonrpc-c
- RPC provided

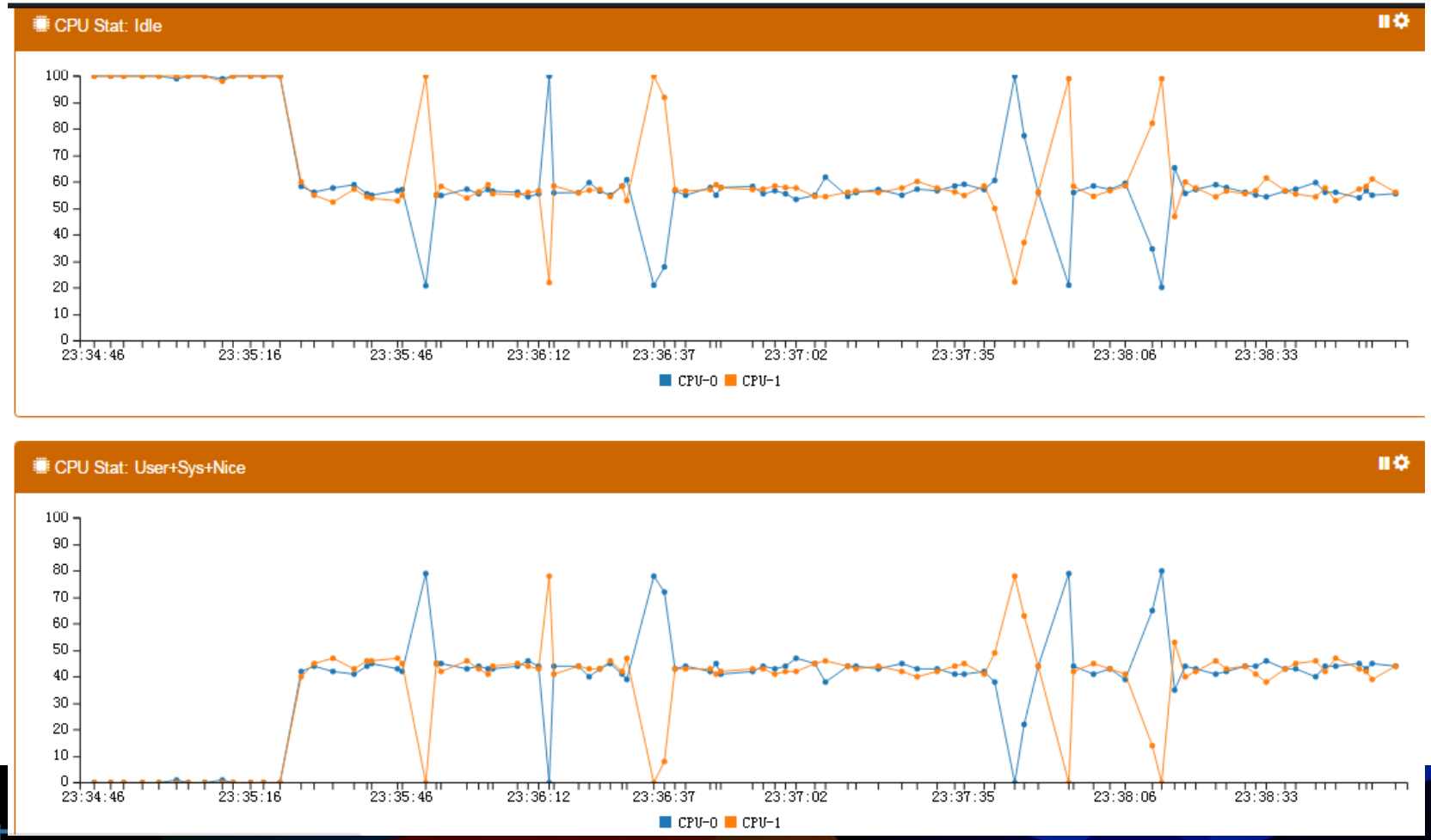
- Read proc entries / Execute a command

RPC - methods

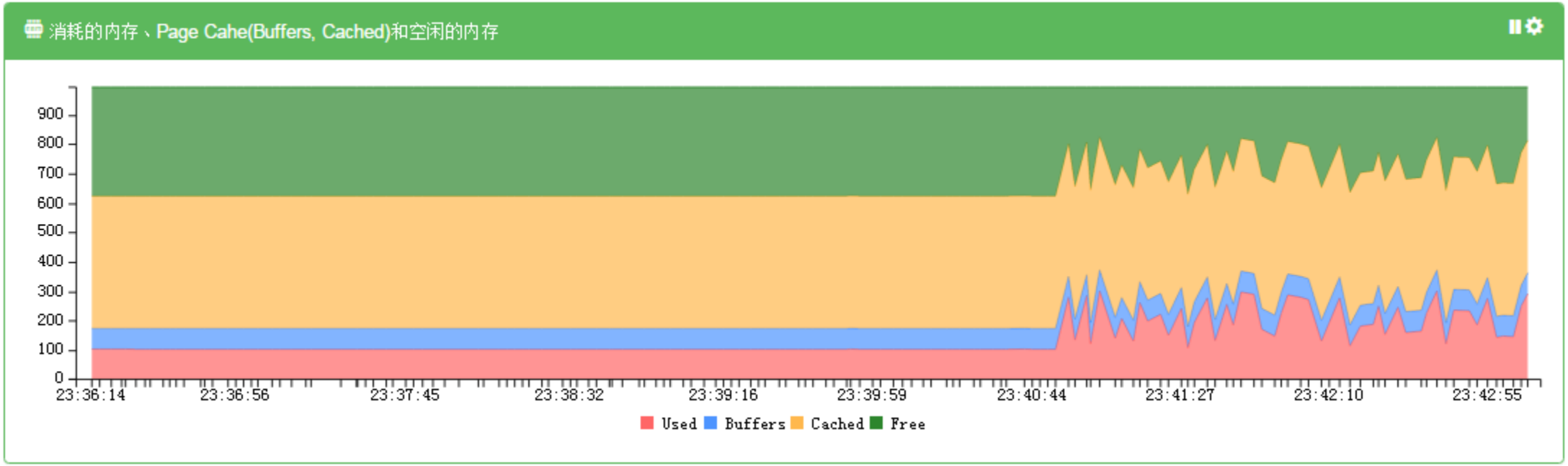
- Get lepd response by command lines

```
ubuntu@i-fpulknj6:~/lep/lepd-src$ echo "{\"method\":\"ListAllMethod\"}" | nc 139.198.3.82 12307
{
        "result":       "SayHello ListAllMethod GetProcMeminfo GetProcLoadavg GetProcVmstat GetProcZoneinfo GetProcBuddyinfo
GetProcCpuinfo GetProcSlabinfo GetProcSwaps GetProcInterrupts GetProcSoftirqs GetProcDiskstats GetProcVersion GetProcStat Get
ProcModules GetCmdFree GetCmdProcrank GetCmdIostat GetCmdVmstat GetCmdTop GetCmdTopH GetCmdIotop GetCmdSmem GetCmdDmesg GetCm
dDf GetCmdMpstat GetCmdPerfFaults GetCmdPerfCpuclock lepdendstring"
}
```

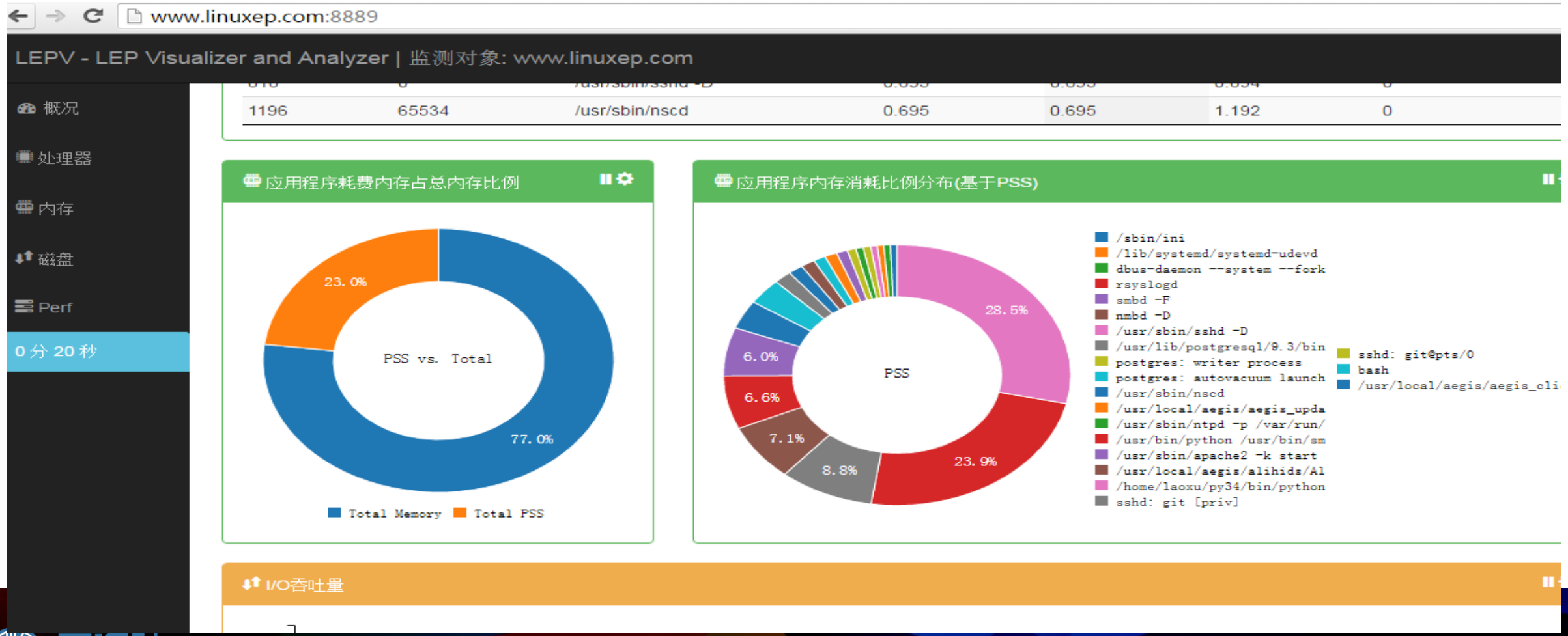
Load balance view



Memory consumption view

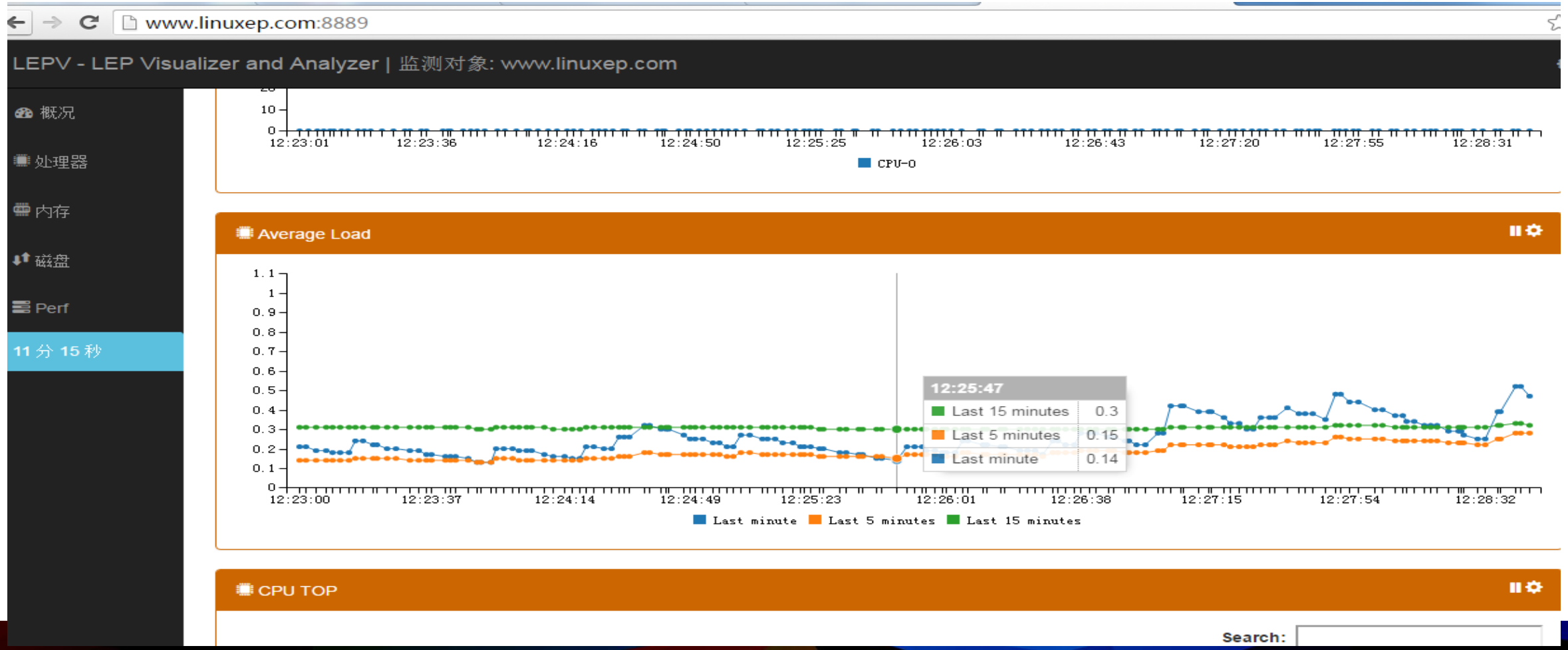


App memory usage view



开源社 kaiyuanshe

LEP: average load view



开源社 kaiyuanshe

LEP: symbol level view

基于Symbol的时间分布 (perf top)

Search:

Command	Overhead	Shared Object	Symbol
malloc	53.53%	malloc	[.] main
swapper	32.87%	[kernel.kallsyms]	[k] native_safe_halt
malloc	12.50%	[kernel.kallsyms]	[k] clear_page
malloc	0.15%	[kernel.kallsyms]	[k] free_pages_prepare
malloc	0.13%	[kernel.kallsyms]	[k] get_page_from_freelist
malloc	0.10%	[kernel.kallsyms]	[k] _cond_resched
malloc	0.08%	[kernel.kallsyms]	[k] clear_huge_page
malloc	0.05%	[kernel.kallsyms]	[k] __do_page_fault
malloc	0.05%	[kernel.kallsyms]	[k] _raw_spin_lock
swapper	0.05%	[kernel.kallsyms]	[k] __do_softirq
malloc	0.03%	[kernel.kallsyms]	[k] page_add_new_anon_rmap
swapper	0.03%	[kernel.kallsyms]	[k] refresh_cpu_vm_stats
free	0.02%	[kernel.kallsyms]	[k] do_exit
free	0.02%	[kernel.kallsyms]	[k] page_add_file_rmap
free	0.02%	ld-2.19.so	[.] 0x0000000000009e2a
free	0.02%	libc-2.19.so	[.] 0x000000000007eae8
gapd	0.02%	[kernel.kallsyms]	[k] __fdget_raw
gapd	0.02%	[kernel.kallsyms]	[k] futex_wait
malloc	0.02%	[kernel.kallsyms]	[k] __do_softirq
malloc	0.02%	[kernel.kallsyms]	[k] __lru_cache_add

Next steps

- More features in LEPD
- More documentations
- Push codes to github
- Docker based LEPV

Coming more features in LEP

- ARM support
- Predict memory leak for an application and kernel
- Analyze cache miss
- Analyze time consumption for one native/Java processes
- Benchmark integration
- I/O queues
- Kernel memory details such as buddy, slab, CMA etc.
- Runtime scheduler(CPU/IO)
- Boot procedure
- ….

Please join LEP

- LEPD git:
- LEPV git:
- LEP documentations git:
- Project website: www.linuxep.com/trac

Plans - LE series

Easy Build

Easy Test

Easy Profiling…