中国Linux内核开发者大会

2016

Linux Performance Profiling & Visualization

Barry Song & Bob Liu & Mac Xu

performance measurements

- top
- iotop
- vmstat
- Sar oprofile
- perf
- … … … … … … … … … … … … …

Problems in existing tools

RAW data, 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

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

Lack of the description for changes

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
```

Problems in existing tools

Cannot interact with users smoothly

e.g. what if we only care about a particular process in “top”?

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3005	baohua	20	0	157392	79036	35196	S	6.0	7.3	0:20.68	compiz
1979	root	20	0	115640	49936	16792	S	2.3	4.6	0:08.01	Xorg
1771	root	0	-20	3844	3828	2636	S	1.3	0.4	0:00.17	atop
3425	baohua	20	0	129308	32356	24324	S	1.0	3.0	0:01.56	gnome-ter+
3595	**baohua**	**20**	**0**	**7012**	**3176**	**2768**	**R**	**0.7**	**0.3**	**0:00.11**	**top**
3045	baohua	20	0	38300	7808	5280	S	0.3	0.7	0:00.10	indicator+
1	root	20	0	4632	3752	2556	S	0.0	0.3	0:02.52	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.14	ksoftirqd+
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0+
6	root	20	0	0	0	0	S	0.0	0.0	0:00.10	kworker/u+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.38	rcu_sched
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration+
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	watchdog/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	migration+
13	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoftirqd+

A Solution

- Intuitive
- Visualized
- Interactive
- Ease of Use

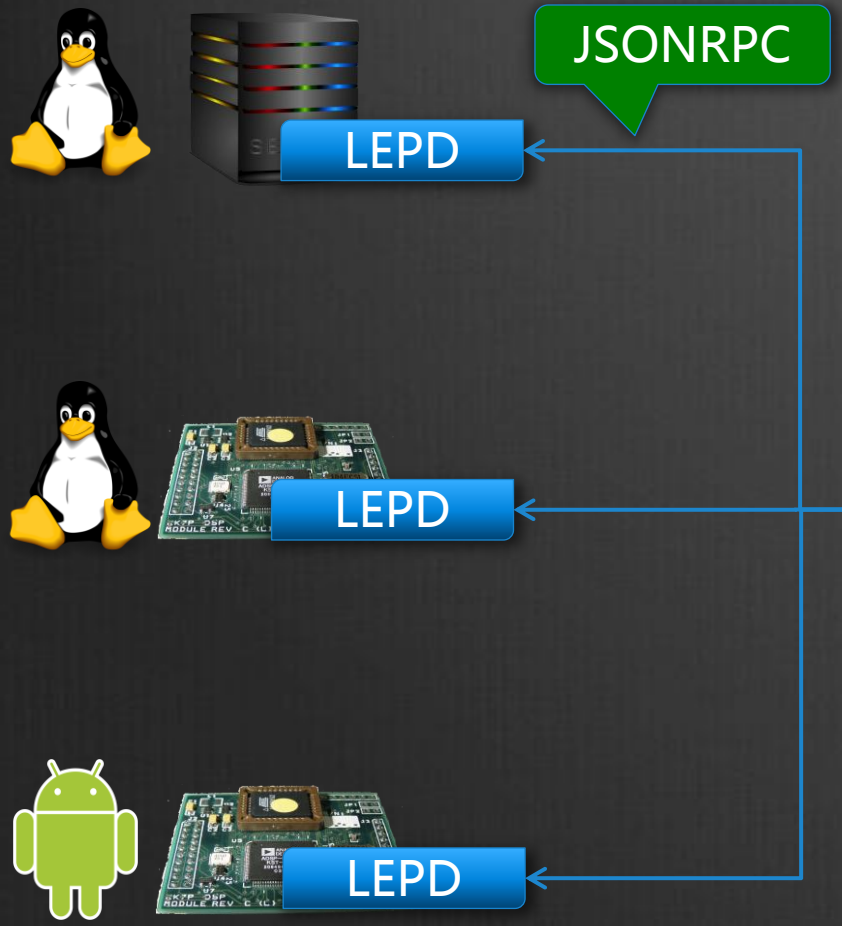
LEP

What is LEP

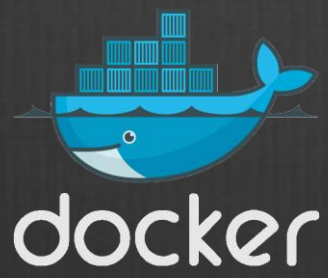
Linux Easy Profiling

- Web-based
- Open-source
- All-In-One

LEP architecture



LEPV



Restful API

LEP Summary

- C/S
- PC / Embedded Board
- Linux / Android
- JSONRPC
- Web App
- Restful API
- Docker

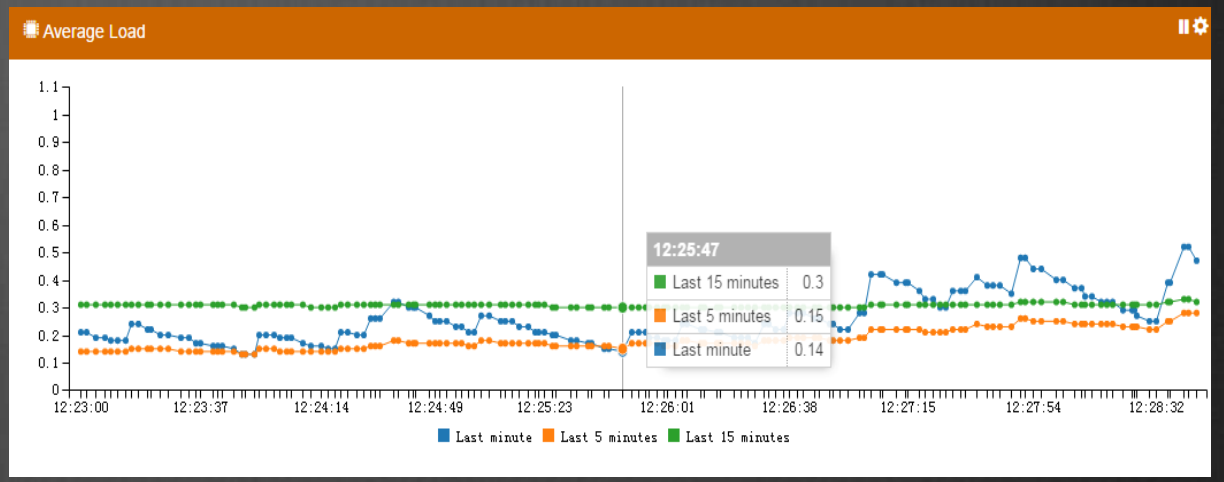
LEP Dataflow

HTTP GET
AVG LOAD

JSON RPC
AVG LOAD

LEPV

LEPD



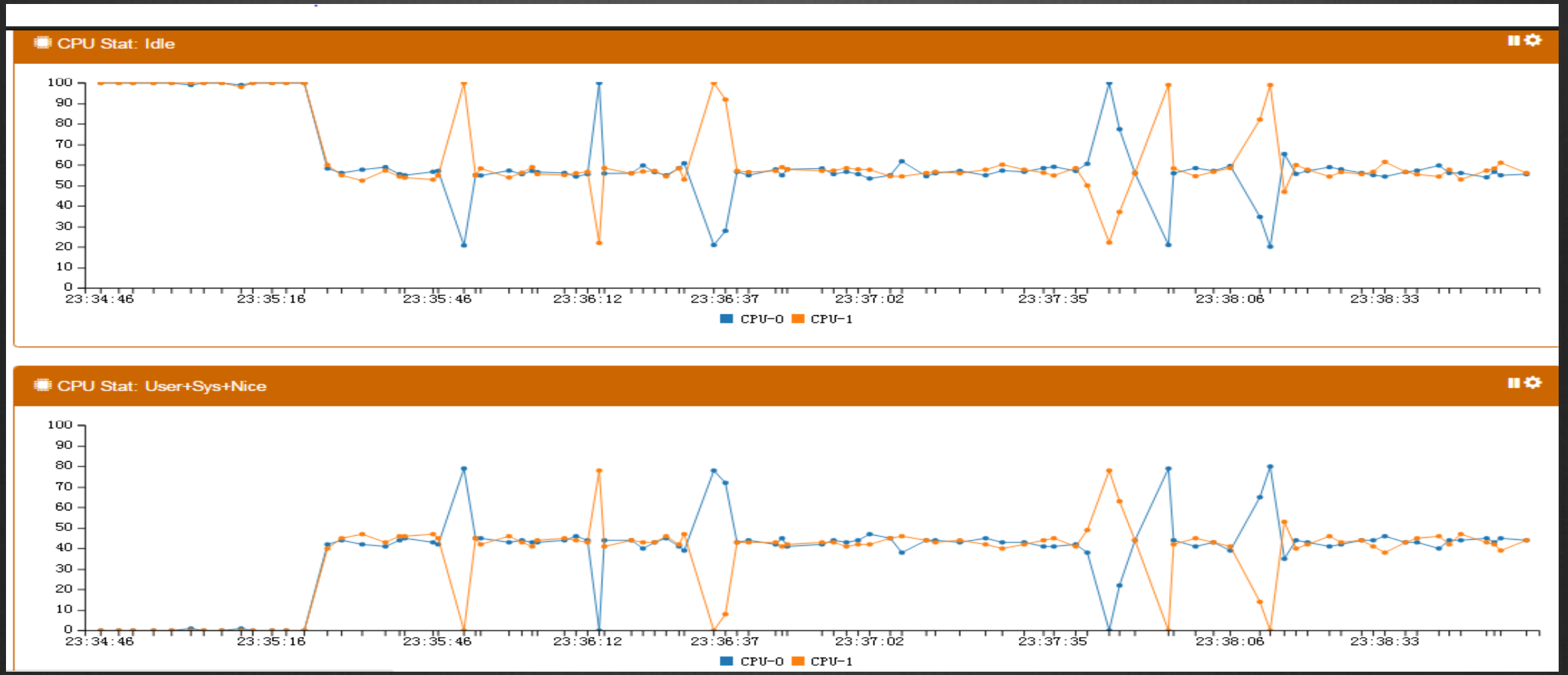

```
{
  'last15':Decimal('2.13'),
  'last5':Decimal('2.25'),
  'last1':Decimal('2.58')
}
```

```
{
  'result':'2.58 2.25 2.13 4/110 19674\n'
}
```

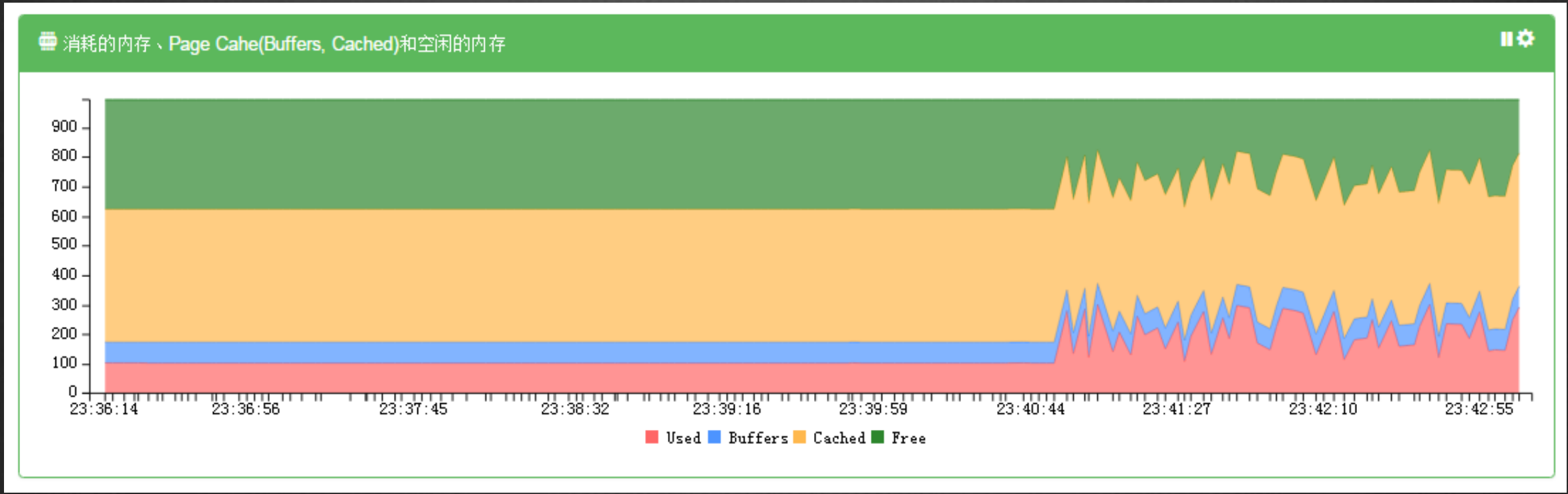
/proc/loadavg

Live Demo

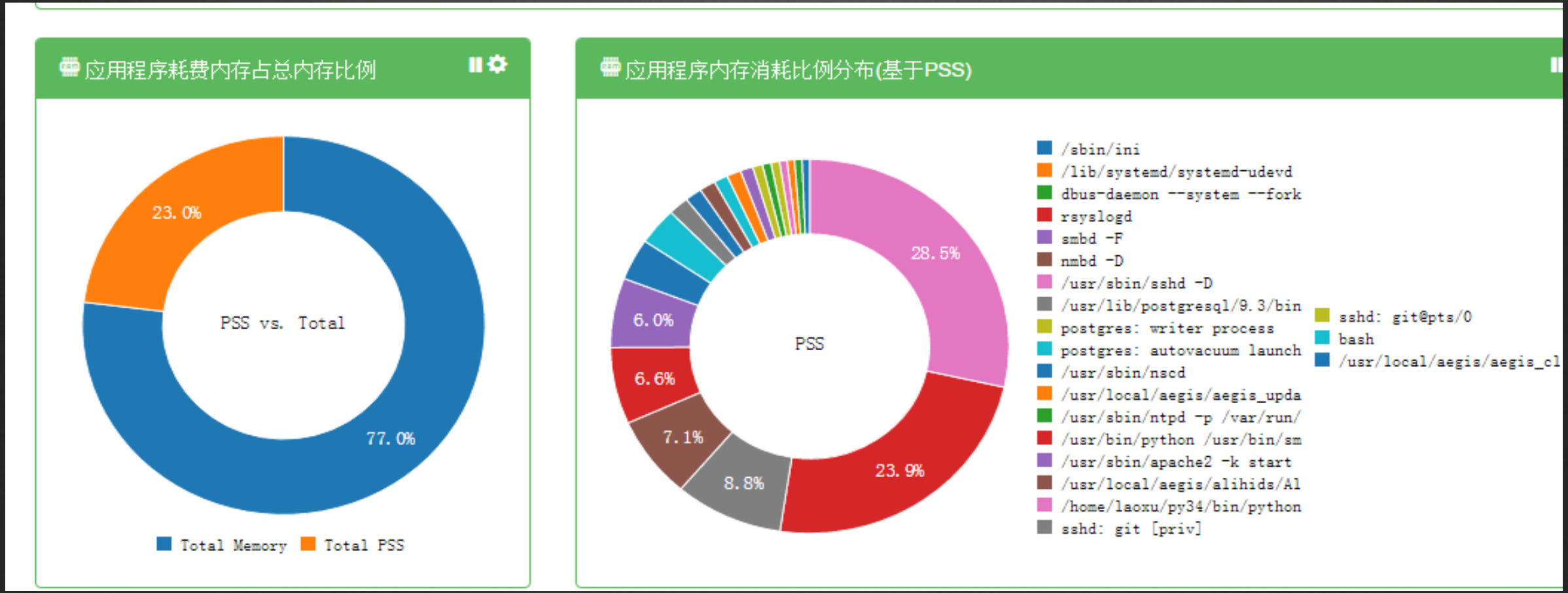
LEP: Load balance view



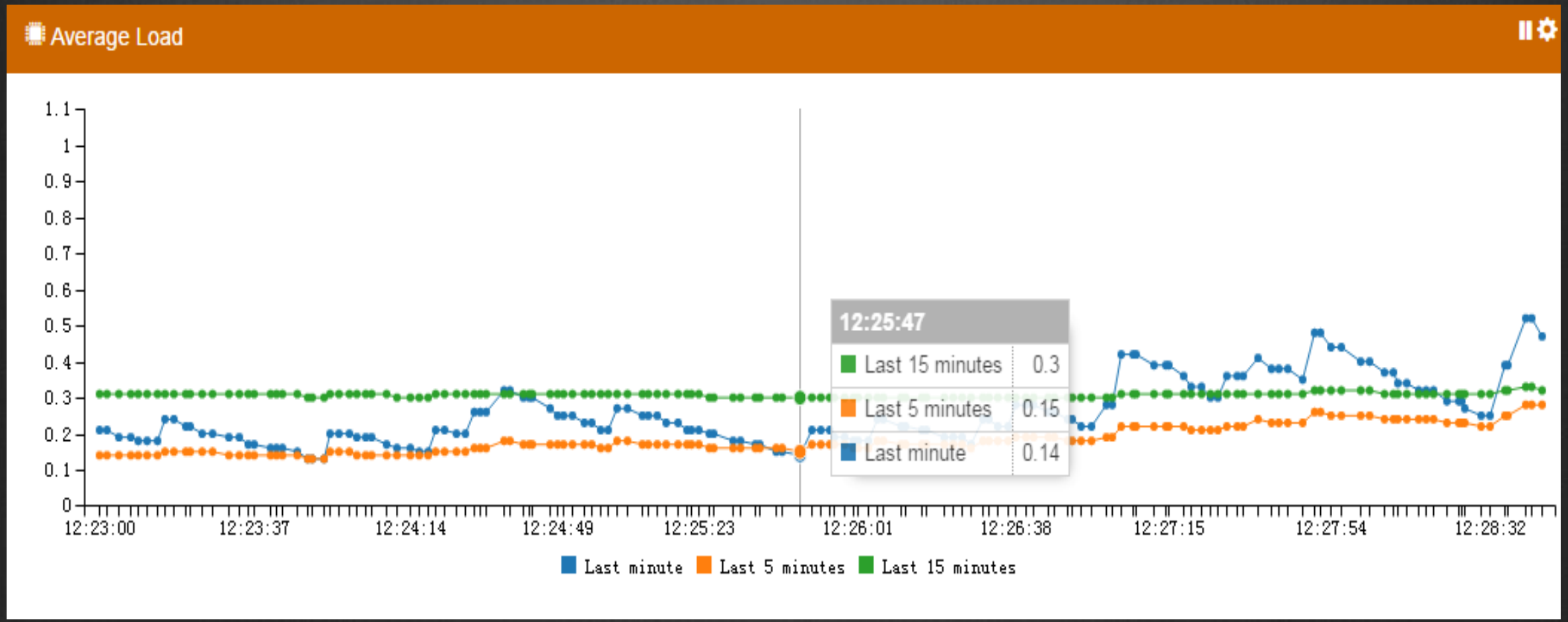
LEP: Memory consumption view



LEP: App memory usage view



LEP: Average load view



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

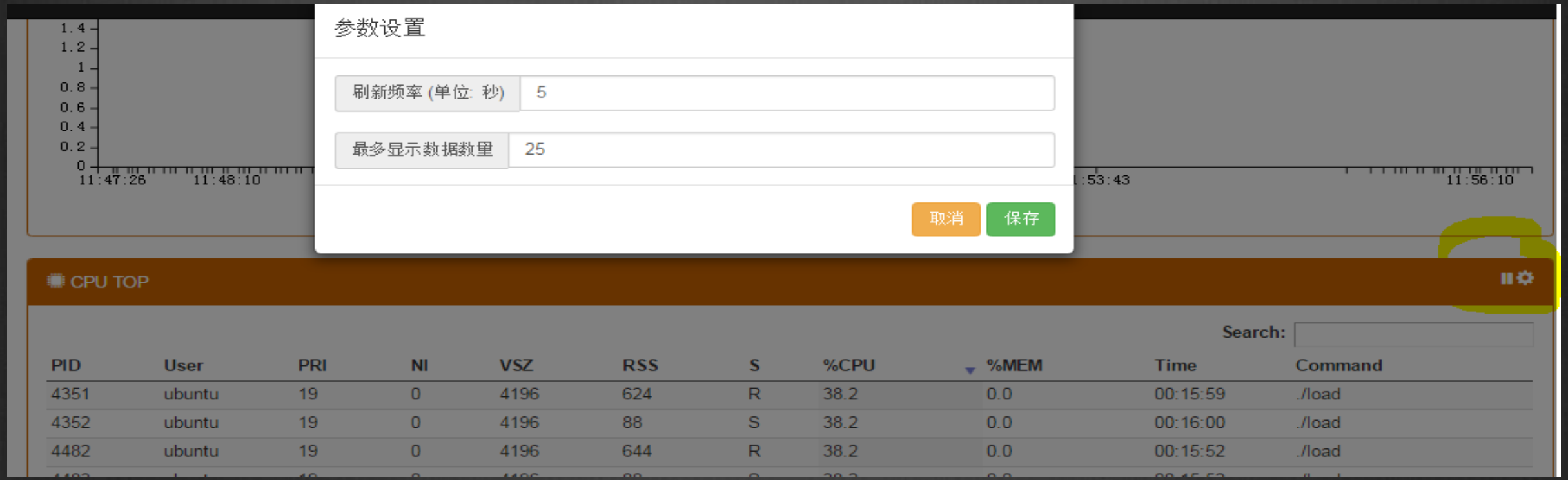
LEP: Interaction with users

CPU TOP

Search: load

PID	User	PRI	NI	VSZ	RSS	S	%CPU	%MEM	Time	Command
4351	ubuntu	19	0	4196	624	S	38.2	0.0	00:14:28	./load
4352	ubuntu	19	0	4196	88	R	38.2	0.0	00:14:28	./load
4483	ubuntu	19	0	4196	88	S	38.2	0.0	00:14:20	./load
4482	ubuntu	19	0	4196	644	R	38.1	0.0	00:14:20	./load

LEP: Interaction with users



Next steps for LEP

- More features
- Better documentations

Upcoming features

- 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
- ....

Plans - LE series

LEB

Easy Building

LET

Easy Testing

LEP

Easy Profiling

Contributors

- Barry Song
- Bob Liu
- Mac Xu
- More developers are coming
  - Ping Liu
  - Ray Chen
  - Joy Zhou
- YOU

Fork the code here:

- LEPD: git@www.linuxep.com:repo/lep/lepd
- LEPV: git@www.linuxep.com:repo/lep/lepv

Q&A