Enhanced execution trace abstraction approach using social network analysis methods

Ji Wang, Naser Ezzati-Jivan

Oct 11, 2020

Symposium on Software Performance (SSP) 2020

Execution tracing is used for performance analysis and debugging.

Usually the trace size is huge, and we need some heuristics to cope with it.

Process	TID	PTID	Birth time	4
gnome-pty-helpe	3609	3596	18:15:38.7695138	4
▼ bash	3610	3596	18:15:38.7695147	ŀ
ssh	8972	3610	18:15:38.7695557	ŀ
▼ bash	5332	3596	18:15:38.7695248	ŀ
ls	15298	5332	18:15:39.9718121	ŀ
▼ bash	10234	3596	18:15:38.7695260	ŀ
ssh	8989	10234	18:15:38.7695567	ŀ
▼ bash	11275	3596	18:15:38.7695713	ŀ
▼ bash	15266	11275	18:15:38.7729050	ŀ
▼ sudo	15291	15266	18:15:38.7729062	ŀ
lttng	15292	15291	18:15:38.7729070	ŀ
ab	15294	15266	18:15:38.8254856	ŀ
▼ sudo	15302	15266	18:15:46.0692929	F
lttng	15303	15302	18:15:46.0735909	ŀ
dconf worker	3599	2779	18:15:38.7695104	ŀ
gmain	3600	2779	18:15:38.7695113	ŀ
gdbus	3601	2779	18:15:38.7695120	ŀ
	1	1		Έ.

Process	TID	PTID	Birth time
unity-greeter	1082	1070	12:21:27.6053039
llvmpipe-0	1159	1070	12:21:27.6053052
llvmpipe-1	1160	1070	12:21:27.6053064
gmain	1175	1070	12:21:27.6053082
gdbus	1176	1070	12:21:27.6053093
dconf worker	1193	1070	12:21:27.6053111
threaded-ml	1295	1070	12:21:27.6053126
gmain	1037	851	12:21:27.6052899
gdbus	1038	851	12:21:27.6052916
lightdm	1200	851	12:21:27.6053416
gmain	872	1	12:21:27.6052339
gdbus	874	1	12:21:27.6052352
▼ mysqld	861	1	12:21:27.6052366
mysqld	1545	861	12:21:27.6665481
mysqld	941	1	12:21:27.6052379
mysqld	955	1	12:21:27.6052392
mysqld	956	1	12:21:27.6052408





🗏 ls-traces 🖾

							_
	Тгасе	Timestamp	Channel	CPU	Event type	Contents	
P	<srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>	
	ls-traces/kernel	2018-01-29 15:00:45.876 797 081	kernel_1	1	kmem_mm_page_alloc	page=0xffffea0009902680, pfn=2506906, order=0, gfp_flags=131290, migratetype=2, contextperf_cpu_mi	
	ls-traces/kernel	2018-01-29 15:00:45.876 797 698	kernel_1	1	syscall_exit_write	ret=4096, contextperf_cpu_migrations=23	
	ls-traces/kernel	2018-01-29 15:00:45.876 797 918	kernel_1	1	syscall_entry_write	fd=28, buf=139984728244528, count=4096, contextperf_cpu_migrations=23	
	ls-traces/kernel	2018-01-29 15:00:45.876 798 223	kernel_1	1	kmem_mm_page_alloc	page=0xffffea000dce7940, pfn=3619301, order=0, gfp_flags=131290, migratetype=2, contextperf_cpu_mi	
	ls-traces/kernel	2018-01-29 15:00:45.876 799 528	kernel_1	1	syscall_exit_write	ret=4096, contextperf_cpu_migrations=23	
	ls-traces/kernel	2018-01-29 15:00:45.876 799 754	kernel_1	1	syscall_entry_write	fd=28, buf=139984728244528, count=4096, contextperf_cpu_migrations=23	
	Is-traces/kernel	2018-01-29 15:00:45 876 800 059	kernel 1	1	kmem mm page alloc	0200-0xffffe2000e078880 pfp-3677730 order-0 afp flags-131200 migratetype-2 context, perf cou mi	



Current methods



→Sampling or event filtering.

✓Time view filtering.

✓Global filtering

	Timestamp	Channel	CPU	Event type	Contents
4	<srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>	tid=861
	12:21:27.650 850 349	more-subbuf_0	0	syscall_entry_connect	fd=3, uservaddr=0x7ffe5564c030, addrlen=0x6e, family=1, dport=0, _v4addr_length=0, v4addr=[], _v6addr_length=0, v6addr=[], contex
۲	12:21:27.650 863 113	more-subbuf_0	0	sched_waking	comm=mysqld, tid=861, prio=20, target_cpu=0, contexttid=1544, contextpid=1544, contextprocname=sysbench
⇒	12:21:27.650 868 663	more-subbuf_0	0	sched_wakeup	comm=mysqld, <mark>tid=861</mark> , prio=20, target_cpu=0, contexttid=1544, contextpid=1544, contextprocname=sysbench
	12:21:27.650 870 198	more-subbuf_0	0	syscall_exit_connect	ret=0, contexttid=1544, contextpid=1544, contextprocname=sysbench
⇒	12:21:27.650 873 554	more-subbuf_0	0	sched_switch	prev_comm=sysbench, prev_tid=1544, prev_prio=20, prev_state=0, next_comm=mysqld, next_ <mark>tid=861</mark> , next_prio=20, contexttid=154 [,]
⇒	12:21:27.650 886 455	more-subbuf_0	0	syscall_exit_poll	ret=1, nfds=2, fds_length=1, fds=[[fd=23, raw_events=0x1, eventsPOLLIN=1, eventsPOLLPRI=0, eventsPOLLOUT=0, eventsPOLL
⇒	12:21:27.650 895 764	more-subbuf_0	0	syscall_entry_accept	fd=23, upeer_sockaddr=0x7fffd9470d50, upeer_addrlen=128, context <mark>tid=861</mark> , contextpid=861, contextprocname=mysqld
⇒	12:21:27.650 904 076	more-subbuf_0	0	syscall_exit_accept	ret=39, upeer_addrlen=2, family=1, sport=0, _v4addr_length=0, v4addr=[], _v6addr_length=0, v6addr=[], context <mark>tid=861</mark> , contextpic
⇒	12:21:27.650 918 648	more-subbuf_0	0	syscall_entry_futex	uaddr=31402148, op=133, val=1, utime=1, uaddr2=31402144, val3=67108865, context <mark>tid=861</mark> , contextpid=861, contextprocname=r
⇒	12:21:27.650 923 327	more-subbuf_0	0	sched_waking	comm=mysqld, tid=1291, prio=20, target_cpu=0, context <mark>tid=861</mark> , contextpid=861, contextprocname=mysqld
⇒	12:21:27.650 926 771	more-subbuf_0	0	sched_wakeup	comm=mysqld, tid=1291, prio=20, target_cpu=0, context <mark>tid=861</mark> , contextpid=861, contextprocname=mysqld
٢	12.21.27 650 928 588	more-subbuf 0	0	syscall exit futex	ret=1 uaddr=31402148 uaddr2=31402144 context_tid=861 context_nid=861 context_nrorname=mvsnld

te Resources = Control Flow ∞ = Statistics									🖞 🗟 🗟	R & & -	∂ ℓ € €	어 '수 수' 10	rộ ¢† (ộ ¢) ≝ ▼ ▽ ⊐	
Process	TID	PTID	Birth time		12:21:28.000	12:21:28.500	12:21:29.000	12:21:29.500	12:21:30.000	12:21:30.500	12:21:31.000	12:21:31.500	12:21:32.000	
▼ normal0/kernel														
▼ mysqld	861	1	12:21:27.6052366											
mysqld	1545	861	12:21:27.6665481				+ - -	10-00-000000000	00.0000000000000000			01) (041 (140 - 140) (090)	COLORADO CONTRACTOR	(CRO) (C
mysqld	941	1	12:21:27.6052379											
mysqld	955	1	12:21:27.6052392		_									_
mysqld	956	1	12:21:27.6052408		_				_				-	
musald	057	4	10-01-07 6050405											
💷 normal0/kernel छ											🗙 mysqld	Q		

Graph theory





 Extract the processes interactions based on the execution graph.

→Building a graph G = (V, E)

✓V includes all the processes running on the system

✓E is a set of edges defined based on the interactions between the processes.

```
or kernel level lock methods.
  Algorithm 1 Execution Graph Construction algorithm, for
events collected from lock library .
  Input Input: Trace T, Threads = {t1, t2, ..., tn}
         Set1 = \{*\_lock\_req\},\
         Set2 = \{*\_lock\_acq\},\
         Set3 = {*\_unlock}
         Output Output: execution graph G
  1: THREADS \leftarrow initial set of threads active in the multi-
 thread application Declarations
 2: LOCKHOLDER \leftarrow \emptyset
 3: for all thread t ∈ THREADS do//Initialization
 4:
       Create initial vertex of thread t with timestamp t.begin
 5: end for
 6 : for all event e \in T do / / Mainprocedure
 7:
       if e ∈ Set1 then
8:
          new h edge(e.tid, e.ts, blocked)
9:
          new_v_edge(e.tid, LOCKHOLDER)
 10:
        end if
11:
        if e \in Set_2 then
12:
           new_v_edge(LOCKHOLDER, e.tid)
 13:
           new_h_edge(e.tid, e.ts, running)
14:
           LOCKHOLDER \leftarrow e.tid
15:
        end if
16:
        if e ∈ Set3 then
17:
           LOCKHOLDER \leftarrow e.tid
 18:
        end if
```



II.

Compute connected components on the graph to extract all the subgraphs. (e.g., BFS)



Method

NB. Computing connected components on the graph

Other approaches: e.g., Graph Modularity



Method

NB. Computing connected components on the graph

Other approaches: e.g., Graph Modularity



Method

III.

 Apply a ranking algorithm on each connected component to assign a rank to each process

✓Choose the top-K processes..



Modified PageRank algorithm: Includes only one iteration

Node rank initialization: equal weights, or even CPU/IO usage

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

u: a process

- B_u: the set of u's in-edges
- N_v : the number of out-edges of process v
- c: the normalization factor

A	Algorithm 2: Process Ranking Algorithm (PRA)									
1 i	f event == sched_ttwu then									
2	j = getVMvCPU(wakee_tid);									
3	k = getVMvCPU(waker_tid);									
4	wakerCR3 = getLastCR3($vCPU_k$);									
5	updateWaker(<i>vCPU_j</i> , wakerCR3);									
6 E	else if event == vm_inj_virq then									
7	j = getVMvCPU(tid);									
8	$pCR3 = getVMvCPU(vCPU_j);$									
9	if vec == IPI then									
10	wakerCR3 = queryWaker($vCPU_j$);									
11	LINK_HORIZONTAL(wakerCR3, pCR3);									
12 f	or all process $cr3_i \in CR3$ do									
13	$ [R(cr3_i);]$									
14 C	connected_subgraph = breadth-first-search(CR3);									
15 C	sustomized graph = denoise(connected subgraph, R(CR3));									

Trace Compass:

Extract process interactions using EASE scripting.

Providing a view showing top-ranked processes in each subgraph.

Filtering top-ranked processes in event view.

😣 🖱 🗊 runtime-EclipseApplicatio	on - T	racing/extractGraphIr	nfo.js - Eclipse	Platfo	orm								
🖻 👻 🖬 🚺 💁 🐑 😫	- 6	• \$ • • •									Quick Acc	ess 🗈	b 🛤
Project Explorer 🛛 🗧		🖩 Resources 📑 Cor	ntrol Flow 🛙	🗆 Stati	istics		😐 🛸 🗄	= 🟠 👒 🤜	🖪 🗞 🗞 👻	🔒 🦊 🔍 🔍	X 🐢 🔹 🗠 🕯	× € × ×	- 8
E 🕏	~	Process	TID	PTID	Birth time	18:15:39	18:15:40	18:15:41	18:15:42	18:15:43	18:15:44	18:15:45	
Active Thread Active Thread Active Thread Context CallStacks Context switch Counters If Counters If Counters If Counters If Access A Futex Contention Analysis If Input/Output A INQ Analysis If Kernel memory usage ALIN Kernel SQ OS Execution Graph Q'Original Flow View		mysald mysald mysald mysald mysald mysald mysald mysald mysald mysald mysald wysald sabd * dismaaq dismaaq	1211 1212 1213 1214 1424 1433 1436 1436 1437 1437 1437 1437 1437 2266 2266	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 265	18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765 18:15:38.765								
? State Machine Automatic. ? State Machine Backend St. • Statistics • System Call Latency • System Call Latency	Analy ate Sy file	E normal0/kernel // load additi loadModule('/I loadModule('/I // open messag var analysis = if (analysis = print(Tra exit();) var ss = analy var ssion = var dictSTrin var dictsTrin var dictsTrin var session = var pids = {}; var pids = {}; var nesses - var nesses -	<pre>E ranking-a onal UI comm raceCompass/ getAnalysis - null) { cc is null"} sis.getState g = ""; 0; </pre>	upache- ands Analys View') ("acti ; System	<pre>sql/kernel is'); vetid.js"); (false);</pre>	SextractGrap	hinfo.js ⊠						-
		🔲 Properties 💷 Bo	okmarks 📮 C	onsole	Stritical Flo	ow View 😫		œ	🖞 🦻 🤜 👘	B B - 1	÷ • • •		- 8
		Process		Eli	apsed P	18:15:39	18:15:40	18:15:41	18:15:42	18:15:43	18:15:44	18:15:45	
		▼ *c1df0453-2088-4300- [/usr/bin/termin,3596 [Xorg,2771] [InputThread 2775]	b1bc-d41083b32 5]	13c" 10. 17. 10. 9 7	586743570 14 696122680 24 586743570 14 31711430 11	a F u e tada			10. 11	Adt m	: An 17	.	4
					Writable	Smart In:	sert 24:13	?				a 🛛 🖛 🗡	0



Top-ranked processes in subgraph I



Top-ranked processes in subgraph II and III



Top-ranked processes in other subgraphs:



						\checkmark								
					•	•								
			1					_					1	
▼ kernel														
CPU 0 Threads														
▶ CPU 0 States														
CPU 0 Frequency	_						3.6 GHz							
CPU 1 Threads	_			chrome (275)					gmain (1410		hrom	gmain (1502		Linom
▶ CPU 1 States			-chrome	1	penat	line. C. Carl	uncolo gr	main	g r gmain	g poll	nunsaal gmaan 🚺	gmain		
CPU 1 Frequency	_						3.6 GHz							
CPU 2 Threads	_	Xorg (1056												
▶ CPU 2 States	-			recymsg					011	in Chirch	ie_IOThread	relevinting - m	Circle 1011	and
CPU 2 Frequency	_						50 (GHz							
CPU 3 Threads	_													
▶ CPU 3 States	-		Chrome-Ci	HdlOT []ah]	Chrome_Ci	IdiOT	- ChrometChilder		Chrome_ChildIOT	recymag h		1		- O I
CPU 3 Frequency	_						3.2 GHz.							
▶ IRQ 1	_													
▶ IRQ 9	_													

▼ kernel											
CPU 0 Threads	-				1111111						
▶ CPU 0 States	_				11 1 11 1						
CPU 0 Frequency	- i	3.6		 	 inin - inin			3. 6			-
CPU 1 Threads	T				11 11						
▶ CPU 1 States											-
CPU 1 Frequency	-	3.	б						3		
CPU 2 Threads					T 1						
▶ CPU 2 States					T						+
CPU 2 Frequency	- in				 -				3		1
CPU 3 Threads							101011 000				(*****
▶ CPU 3 States	_									11 11	-
CPU 3 Frequency	_ <u> </u>		3								+
▶ IRQ 1)		11 111	 	 	-		111			-
▶ IRQ 9				 	 						-

Conclusions & Future work

- •Filtering based on process ranking
- •Top interacting processes in one system
- Interaction between VM's processes in a cloud environment
- •Time-window feature extraction for machine learning applications
- •Enhance graph structure analysis, e.g., graph modularity

Questions

Thanks for your attention! nezzati@brocku.ca