Utilizing tools made for *Big Data* to analyse Ftrace data: making it fast and easy

Yordan Karadzhov

VMware Inc. - OSTC



2019 VMware, Inc.

Motivation

a. Ftrace: The official tracer of the Linux kernel (No need to explain this)

b. Is the Ftrace data Big Data?

- * Not necessarily. It depends how you use it.
- * Extremely sophisticated instrument. Large variety of use cases.

Motivation

a. OK, I have a nontrivial or very user-specific problem.

b. I have recorded a lot of tracing data.

c. What should I do now?

Motivation

a. OK, I have a nontrivial or very user-specific problem.

b. I have recorded a lot of tracing data.

c. What should I do now?

KernelShark Not going to explain it here. See Steven Rostedt's presentation at OSS NA 2019.



KernelShark (Something More)

	ter	Plots Tools Help	p			Ker	nel Shark (/ho	ome/rostedt/demo/trace-migrate.dat)	
Pointer	265	55208 695370	inrate-27494 CPI	13 db	brtimer o	ancel brtimera	=0xfffb659854	97,690	
on ter.	120.	15200.055370 m	ignace 27454 Cro	o o u.m.	in uniter_c	ancer mamer-	-0,11110035031		
<	+	· > ++	Marke	er A 2659	5208.755	795 Marke	r B 2655209.	760 168 A.B Delta: 1.004 373 856	
2	6552	207.441307						2655209.994748 2655212.54	8189
									- L
CPU 0	- 11			W. W. L		UL AUDIO DI DI UL		an a	
CPU 1					mand - 10			ER ENTRY JANNENA EVEN ETTE E EVEN ALERE ETTE STELLE ALERE EN	11
:PU 2		13.0.0.0000.0.000	a a sua ana		111 I I I I I I I I I I I I I I I I I I	1.1.1.1.1.1.1.1.1.1	a dha an	a na sa ang ang ang ang ang ang ang ang ang an	
CPU 3	П								
CPU 3	Ц			NUM.	we a			A A MARINA TARA MANANANA MANINA M	
CPU 3					1 11				
CPU 3			an naan araa	No.	w i		WWWWWW	A FAMA PERSONALAR AND	
CPU 3	colu	mn #	· contains	• I	11 I I	ke	xt Prev	a bana <mark>banan burun an ana ana ana ana ana ana ana ana an</mark>	
CPU 3 earch:	Colu	mn #	contains	* PiD	Latency	Ne	xt Prev	A A A A A A A A A A A A A A A A A A A	
earch:	Colu CPL	mn # Time Stamp 2655209.759071	contains Task migrate	• PID 27499	Latency	Ne Event print	xt Prev	A 1999 A 1999 V Creph follows Info Inscing_mark_write: thread 7 liter 50 sleeping	
earch: 2540 2541	Colu CPL 3 0	mn # J Time Stamp 2655209.759071 2655209.759071	contains Task migrate migrate	• PID 27499 27498	Latency	Ne Event print sched_switch	xt Prev	V Creph follows Info Incide_mark_write: thread 7 liter 50 sleeping I	
earch: 2540 2541 2542	Colu 3 0 3	mn # 7 Time Stamp 2655209.759071 2655209.759072 2655209.759072	Contains Task migrate migrate migrate	 PID 27499 27498 27499 	Latency	Ne Event print sched_switch hrtimer_init	xt Prev	Vorsph follows Trans, write: thread 7 iner 50 alegging mignet 22495 (94) R ==> migrate: 22496 (94) mignet 22495 (94) R ==> migrate: 22496 (94) migrate: 22496 (94) R ==> migrate: 22496 (94)	
earch: 2540 2541 2542 2543	Colu 3 0 3 3	mn # J Time Stamp 2655209.759071 2655209.759072 2655209.759072	Contains	 PID 27499 27498 27499 27499 	Latency	Ne Event print sched_switch hrtimer_init hrtimer_start	xt Prev	Y Christian Annual Control Contro	1861
earch: 2540 2541 2542 2543 2544	Colu 3 0 3 3 0	mn #	contains Task migrate migrate migrate migrate	* PID 27499 27498 27499 27499 27499 27499	Latency d d d	Ne Print sched_switch hrtimer_init hrtimer_start sched_migrate	xt Prev	VCraph follows Image: Comparing the standard	.1861
earch: 2540 2541 2542 2543 2544 2546	Colu 3 0 3 3 0 0	mn #	 contains Task migrate migrate migrate migrate migrate migrate 	 PID 27499 27498 27499 27499 27499 27498 27498 27498 27498 27498 	Latency d d d	Ne Event print sched_switch hrtimer_init hrtimer_start print	xt Prev	VCreph follows Image: Strength of the strengt of the strength of the strength of the strength of the s	.1861
earch: 2540 2541 2542 2543 2544 2546 2546	Colu 3 0 3 3 0 0 1	mn #	Task migrate migrate migrate migrate migrate migrate	 PID 27499 27498 27499 27499 27498 27498 27498 27498 27498 	Latency d d d d	Ne Fvent print sched_switch hrtimer_int hrtimer_start sched_migrate print	xt Prev	Voraph follows Toring_mark_write: Thread 7 Intr 50 sleeping migrate 27496 (91) httms-outfflossbab/2r0 oldekal-cock_MANG40 opters=2539972088611861 softeepires=253997208861 httms-outfflossbab/2r0 oldekal-cock_MANG40 foldekal-cock_MANG40 httms-outfflossbab/2r0 oldekal-cock_MANG40 httms-outfflossbab/2r0 oldekal-cock_MANG40 httms-outfflossbab/2r0 softeepires=253997208861	1861
earch: 2540 2541 2542 2543 2544 2546 2546 2676	Colu 3 0 3 3 0 0 1 1	mn #	contains Task migrate migrate migrate migrate migrate migrate migrate	 PID 27499 27498 27499 27499 27499 27498 	Latency d d d d d d	Ne print sched_switch hrtimer_init sched_migrate print print	xt Prev	VCreph follows Impact of the state of the s	1861

Stolen slide from Steven's presentation at OSS NA 2019.

1212121001200140 Internet inte

vmware^{*}

KernelShark (Something More)



Steven is doing:

- * Switching to Marker A and clicking at the right event.
- * Switching to Marker B and clicking at the right event.
- * Getting the latency value that shows up in **A B Delta**.

						-	
62824	0	2655209.761039 n	nigrate 2	27498	d.h.	hrtimer_cancel	hrtimer=0xffff89861e61d200
62828	0	2655209.761039 m	nigrate 2	27498	d.h.	hrtimer_expire_entry	hrtimer=0xffff89861e61d200 now=2539972883578655 function=tick_sched_timer/0x0
62830	0	2655209.761040 n	nigrate 2	27498	d.h.	softirq_raise	vec=1 [action=TIMER]
62838	0	2655209.761042 n	nigrate 2	27498	dNh.	hrtimer_expire_exit	hrtimer=0xfff89861e61d200
4 20/1	0	2655200 761042 0	oigrato -	7400	ANK	hrtimor start	httimar=Aufff00061a61d300 function=tick color timar/0v0 avairac=3530073007577645 coftavairac=3530073003577645*

KernelShark (Something More)



Steven is doing:

- * Switching to Marker A and clicking at the right event.
- * Switching to Marker B and clicking at the right event.
- * Getting the latency value that shows up in A B Delta.

Now imagine doing this 10K times - Ugh!!!

There must be a better way to get this job done.

Imagine having something like this:



```
import ksharkpy as ks
ks.open file('trace.dat')
data = ks.load data()
data size = ks.data size(data)
for i in range(data size):
   if data['event'][i] == my_event_a:
      action1
   elif data['event'][i] == my_event_b:
      action2
ks.close()
print('some summary of the results')
plot('some cool histograms or graphs for my presentation')
```



a. General purpose languages: C, Perl, Python ...

b. Numerical languages: Fortran, MATLAB, R, ...

* Written mostly for scientific numerical use.

$Python\,+\,Scientific\,\,computing\,=\,NumPy$

NumPy is not built in to the Python language. It is a library.

It provides:

- 1. Powerful densely packed N-dimensional arrays of homogeneous type.
- 2. Large collection of high-level mathematical functions to operate on these arrays.
- 3. Tools for integrating C/C++ and Fortran code.
- 4. Complementary packages like:
 - a. Matplotlib plotting package that provides MATLAB-like plotting functionality.
 - b. SciPy library that adds functionalities for optimization, linear algebra, integration, interpolation, FFT, signal and image processing.

NumPy

- * Many Numpy operations are implemented in C.
- * In fact the Numpy arrays are very similar to the C arrays.
- * Numpy array can be initialized from C-computed array without data copying COOL!!!

NumPy

- * Many Numpy operations are implemented in C.
- * In fact the Numpy arrays are very similar to the C arrays.
- * Numpy array can be initialized from C-computed array without data copying COOL!!!
- Let's use something that is already (almost ;-) available in KernelShark libkshark.so

Already available in KernelShark 1.0

Example of loading data using libkshark.so:

```
#include "libkshark.h"
int main(int argc, char **argv)
    struct kshark_context *kshark_ctx = NULL;
    struct kshark_entry **data = NULL;
    int data size:
    kshark instance(&kshark ctx);
    kshark_open(kshark_ctx, "trace.dat");
    data_size = kshark_load_data_entries(kshark_ctx, &data);
    for (r = 0; r < data_size: ++r) {</pre>
        if (data[i]->event id == mv event a)
            action1:
        if (data[i]->event_id == my_event_b)
            action1:
```

```
.
/* Free the memory. */
for (r = 0; r < data_size; ++r)
free(data[r]);
free(data);</pre>
```

```
/* Close the file. */
kshark_close(kshark_ctx);
```

```
/* Close the session. */
kshark_free(kshark_ctx);
```

```
printf("some summary of the results")
/*
 * Unfortunately, no simple way to show cool histograms/graphs here :(
 */
```

```
return 0;
```

Summary

- a. PoC NumPy interface for accessing Ftrace data in Python (via NumPy arrays).
- b. The implementation is just a tiny wrapper around libkshark

Summary

- a. PoC NumPy interface for accessing Ftrace data in Python (via NumPy arrays).
- b. The implementation is just a tiny wrapper around **libkshark**

Let's see some examples.

Example 1

Trying to reproduce the study done by **Prof. Dr. Wolfgang Mauerer** and **Daniel Wagner**.

See:

Cyclic Tests Unleashed: Large-Scale RT Analysis with Jitterdebugger Open Source Summit Japan 2019



Example 1

Goal:

- * Statistical estimate of the probability of exceeding the Worst Case Execution Time (WCET)
- * Remember that this is just an example demonstrating the PoC NumPy interface for Ftrace data. The whole credit for the development of the analysis itself goes to Wolfgang and Daniel.

Example 1: Jitterdebugger - Idle machine

😣 –	0				Kernel S	hark (trace-jitter.dat)	
File Filter	Plots	Tools Help					
Pointer: 3	11.1625	95					
< +	1.1	> ++	Marker	A 311.	162 311	Marker B 311.	163 311 A,B Delta: 0.001 000 327
		311.10210	,	_		311.103047	311.103700
CPU 1							
jitterdebu	Jgger-16	54					
							•
Search: Co	lumn #	¥	contains	•		Next Prev	✓ Graph follows
#	CPU	Time Stamp	Task	PID	Latency	Event	Info
4551550	1	311.162305	<idle></idle>	0	d.h20	sched_waking	comm=jitterdebugger pid=1654 prio=19 target_cpu=
4551551	1	311.162306	<idle></idle>	0	dNh10	hrtimer_expire_exit	hrtimer=0xffffc9000269be70
4551552	1	311.162307	<idle></idle>	0	d20	sched_switch	swapper/1:0 [120] R ==> jitterdebugger:1654 [19]
4551553	1	311.162310	jitterdebugger	1654	0	print	tracing_mark_write: jitter=4
4551554	1	311.162311	jitterdebugger	1654	0	hrtimer_init	hrtimer=0xffffc9000269be70 clockid=CLOCK_MONO
4551555	1	311.162311	jitterdebugger	1654	d10	hrtimer_start	hrtimer=0xffffc9000269be70 function=hrtimer_wake
4551556	1	311.162313	jitterdebugger	1654	d20	sched_switch	jitterdebugger:1654 [19] S ==> swapper/1:0 [120]
4551557	7	311.162477	<idle></idle>	0	d.h20	hrtimer cancel	hrtimer=0xffffc90002c6fe70
4551558	7	311.162478	<idle></idle>	0	d.h10	hrtimer expire entry	hrtimer=0xffffc90002c6fe70 now=311582109672 fur
4551559	7	311.162478	<idle></idle>	0	d.h20	sched waking	comm=iitterdebugger pid=1660 prio=19 target cpu=
4551560	7	311.162479	<idle></idle>	0	dNh10	hrtimer expire exit	hrtimer=0xffffc90002c6fe70
4551561	7	311.162480	<idle></idle>	0	d20	sched switch	swapper/7:0 [120] R ==> jitterdebugger:1660 [19]
4						-	•

vmware

Example 1: Jitterdebugger - Idle machine

😣 –					Kernel S	hark (trace-jitter.dat)	
File Filte	er Plots	Tools Help	1				
Pointer	211 1623	14 105	ardabuggar-165				
Fonter.	511.1055	Jee Jee	erdebugger-ros	4 CFU I			
< +	• • •	> ++	Marker	A 311.1	63 311	Marker B 311.	163 310 A,B Delta: -0.000 001 635
		311.10329	,			311.103321	311.10393
CPU 1							
					1		
Uthordob		54					
Jitterdeb	ugger-16	54					
					·		
							*
Saarshi C	aluma #		contains	-		blowh Drow	d Crash follows
Search: C	otumn #		concains	•		Next Pre-	V Graph rollows
#	CPU	Time Stamp	Task	PID	Latency	Event	Info
4551715	1	311.163305	<idle></idle>	0	d.h10	hrtimer_expire_entry	hrtimer=0xffffc9000269be70 now=311582937099 fu
4551716	1	311.163305	<idle></idle>	0	d.h20	sched_waking	comm=jitterdebugger pid=1654 prio=19 target_cpu=
4551717	1	311.163306	<idle></idle>	0	dNh10	hrtimer_expire_exit	hrtimer=0xffffc9000269be70
4551718	1	311.163307	<idle></idle>	0	d20	sched_switch	swapper/1:0 [120] R ==> jitterdebugger:1654 [19]
4551719	1	311.163310	jitterdebugger	1654	0	print	tracing_mark_write: jitter=4
4551720	1	311.163311	jitterdebugger	1654	0	hrtimer_init	hrtimer=0xffffc9000269be70 clockid=CLOCK_MONO
4551721	1	311.163311	jitterdebugger	1654	d10	hrtimer_start	hrtimer=0xffffc9000269be70 function=hrtimer_wake
4551722	1	311.163313	jitterdebugger	1654	d20	sched_switch	jitterdebugger:1654 [19] S ==> swapper/1:0 [120]
4551723	7	311.163477	<idle></idle>	0	d.h20	hrtimer_cancel	hrtimer=0xffffc90002c6fe70
4551724	7	311.163478	<idle></idle>	0	d.h10	hrtimer_expire_entry	hrtimer=0xffffc90002c6fe70 now=311583109687 fur
4551725	7	311.163478	<idle></idle>	0	d.h20	sched_waking	comm=jitterdebugger pid=1660 prio=19 target_cpu=
4551726	7	311.163479	<idle></idle>	0	dNh10	hrtimer_expire_exit	hrtimer=0xffffc90002c6fe70
4							•

vmware

To make it more interesting, let's do the test on a heavy loaded system.

Hackbench: stress test for the Linux kernel scheduler.



Example 1: Jitterdebugger

Extreme Value Theory: Peak over Threshold (PoT) approach



Example 1: Jitterdebugger



Generalized Pareto distribution: A continuous probability distributions. It is often used to model the tails of other distributions.

Example 1: Jitterdebugger ~2.4M cycles



vmware^{*}

Example 1: Jitterdebugger ~2.4M cycles



vmware^{*}

Example 1: Jitterdebugger

Idle vs. Hackbench



Example 1: Jitterdebugger

Let's see some real code

KernelShark session description file (JSON)

Allows loading predefined sessions.

```
"type": "kshark.config.session".
    "time": 1567097989
    "type": "kshark.config.model",
        5831246982148,
        5831250072340
    ],
    "bins": 1000
}.
```

```
"isSet": true,
       "row": 8467891
       "isSet": true,
       "row": 8469063
    26456
"ViewTop": 8467886
```

Example 1: Jitterdebugger

kernelshark -s max_lat.json

0 -	a				ĸ	ernel Shark (trace-jitter-load.dat)	1			
File Filte	er Plots	Tools Help								
Pointer:	5831.249	024								
				140.006	Marker B 5821	149.047 A # Dalta: 0.000.027.104				
					(marker of pages	247 042 Au Ditta: 0.000 037 194				
		5831.248950				5831.249037 5831.2491				
			1.11	1.1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1					
CPU 0					and the second state of th	and the second	•			
jitterdeb	ugger-26	456					- 1			
Search: Co	olumn #	 contains 	*		Next Prev	✓ Graph follows				
	CPU	Time Stamp Task	PID	Laten	cy Event	info				
8469039	0	5831.249001 hackbench	43623	d.h10	sched_waking	comm-jitterdebugger pid=26456 prio=19 target_cpu=000				
8469040	0	5831.249002 hackbench	43623	dNh.0	hrtimer_expire_exit	hrtimer=0xffffc9000f80fe70				
8469041	7	5831.249003 hackbench	43592	d.20	sched_switch	hackbench:43592 [120] 5> swapper/7:0 [120]				
8469042	1	5831.249005 hackbench	43593	d.20	sched_switch	hackbench:43593 [120] 5 ==> hackbench:44650 [120]				
8469043	6	5831.249005 hackbench	43594	d.20	sched_switch	hackbench:43594 [120] 5 ==> swapper/6:0 [120]	_			
8469044	0	5831.249005 hackbench	43623	d.20	sched_switch	hackbench:43623 [120] R ==> jRterdebugger:26456 [19]	41			
8469045	1	5831.249007 hackbench	44650	d.20	sched_switch	hackbench:44650 [120] D ==> swapper/1:0 [120]				
8469046		5831.249017 hackbench	44064	0.21	sched_switch	hackbench:44064 [120] D ==> hackbench:45788 [120]				
8469047		5831.249019 hackbench	45788	0.20	sched_waking	comminackbench pid=46497 prio=120 target_cpu=005				
8469048		5831.249022 <1085		0.20	sched_switch	swapper/cu [120] R => hackbence.obj/ [120]				
8469049		5831.249024 hackbench	40497	4.30	sched_waking	commenteckbench pid=42852 prio=120 target_cpu=006				
8469051	7	5831.249023 Hackberton	0	d 20	sched_switch	swapper/2.0 [120] B === backbends/32052 [120]				
8449052	5	5831 249027 backbeach	43037	d 20	sched waking	comparisher biologi and a station and a station of the station of				
8469053	6	5831.249029 backbench	46497	d.20	sched switch	hackbench:(6697 [120] D ==> swapper/60 [120]				
8469054	7	5831.249029 hackbench	42852	d.20	sched waking	comm+hackbench pid=44912 prio=120 target_cpu+005				
8469055	6	5831.249031 «idie»	0	d.20	sched_switch	swapper/6:0 [120] R ==> hackbench:42757 [120]				
8469056	5	5831.249033 hackbench	43037	d.20	sched_switch	hackbench:43037 [120] D ==> swapper/5:0 [120]				
8469057	6	5831.249033 hackbench	42757	d.20	sched_waking	comm=hackbench pid=44305 prio=120 target_cpu=007				
8469058	1	\$831.249036 <idie></idie>	0	d.20	sched_switch	swapper/1:0 [120] R ==> hackbench:44305 [120]				
8469059	6	5831.249038 hackbench	42757	d.20	sched_switch	hackbench:42757 [120] D ==> hackbench:44912 [120]				
8469050	1	5831.249039 hackbench	44305	d.20	sched_waking	comm=hackbench pid=44004 prio=120 target_cpu=007				
8469061	6	5831.249040 hackbench	44912	d.20	sched_waking	comm+hackbench pid=42859 prio=120 target_cpu=005				
8469062	4	5831.249041 sh	42743	d.21	sched_waking	comm+hackbench pid=44064 prio=120 target_cpu+005	-1			
8469053	0	5831.249042 itterdebugg	er 26456	0	print	tracing_mark_write: Jitter=7	-			
8469054	5	5831.249042 <idie></idie>	0	0.20	scned_switch	amabbei/am [150] H +=> ustabeucucataaa [150]				
8409065	0	5831.249044 jitterdebugg	er 26456	0	nrtimer_init	hrbmer-oxrmcooperore ro cooper-cLOCK_MONOTONIC mode+0x0				
0409060		5631.249044 hackbench	+4305	0.20	scrieu_switch	nacuencin+4505 [120] D === swapper/10 [120]				
8469067	4	5831.249045 Jitterdebugg	44013	d.10	recover_start	mumer=uximicsourourezo runcuommicimer_wakeuppvk0 expires=5831733759675 softexpires=5831733759675 backbarebi46812 [120] D === runcuomer/60 [120]				
8440040		SETT 140045 hackbench	41912	4 30	sched suitch	hackbarch/19818 [130] B hackbarch/1004 [130]				
0409009		5031.249043 Tabadah	42.033	4.00	sched with					



Example 2: Page Faults

Demo

will ware inc.