

Multi-Level Tracing of Containerized Application Orchestrator

Benjamin Saint-Cyr

Polytechnique Montréal DORSAL

Cloud Computing

- aggregation and distribution of computing resources in a homogeneous way to clients
- **2** Allows customers to scale up operations instantly
- 3 Also facilitate the restitution of resources as needed
- **4** Flexible billing based on usage

Containers

- Containers have emerged as a more efficient method for resource allocation and isolation
- Share the same Kernel
- 8 Containers simplify packaging and distribution services
- Provide a consistent runtime environment in both development and production



- Orchestrators are an easy way to deploy containers to the cloud
- 2 They enable the deployment of services using descriptive configurations
- Orchestrators facilitate achieving and maintaining the desired state
- Also offers utilities for automatic scaling, implementing rollout strategies, and managing configurations and secrets

- 1 Performance is an important factor
- 2 Reduced latency enhances the quality of service provided
- **3** Quicker boot times improves elastic scaling
- One efficient resource utilization directly leads to reduced hosting costs

Research Objective

To observe and analyze the behaviour of a distributed system of orchestrated containers, with the aim of facilitating the diagnosis of performance issues



Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

Performance Aspects

- 1 Quicker boot times enhance elasticity in scaling
- Better utilization of resources improves Quality of Service (QoS) and reduces costs



Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

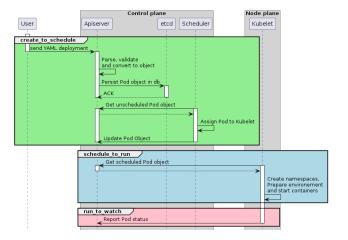


Figure: Components and phases of a pod start up

Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

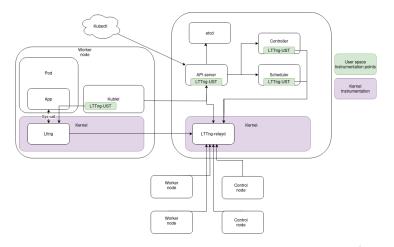


Figure: Tracing Architecture

Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

Common Issue During Execution : Managing CPU Limits Limits are specified in cores, but enforced as quotas

2 A container can use more cores than allocated

Quota = Cores Limit × *Period*

Throttled Time = $Period \times (Cores Used - Cores Limit)$



Expectation

Running	Running
Running	Running
Iddle	Iddle

Reality

Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED
Running	PREEMPTED	Running	PREEMPTED

Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr



The app needed 100ms The app was throttled 120ms

Figure: Example of Latency Introduced by CPU Limit



Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

Algorithm to characterize CPU usage per pod

Algorithm 1 CPU usage calculation per Cgroup

```
1: Define the function normalize(prevTime, time, cpuTime) as \frac{\text{cpuTime}}{\text{time-prevTime}} \times 100
 2: procedure ANALYZECPUUSAGE(fetchParameters, stateSystem)
       Initialize totalCpu \leftarrow 0
 3.
       Initialize prevTime \leftarrow Get the start time from fetchParameters
 4:
       Initialize xAxis \leftarrow Initialize xAxis based on the time range and resolution
 5:
       Initialize map thread Usage \leftarrow Initialize the cpu usage of each thread based on stateSystem
 6:
       Initialize map CaroupUsage \leftarrow Initialize the cpu usage of each caroup based on stateSystem
 7:
       for i \leftarrow 0 to xAxis.length - 1 do
 8:
           time \leftarrow xAxis[i]
9:
           for all (threadName, cpuTime) \in fetchThreadsCpuUsage(time) do
10.
11:
              totalCpu \leftarrow totalCpu + cpuTime
12:
              normalizedThreadCpuUsage \leftarrow normalize(prevTime, time, cpuTime)
13.
              threadUsage[threadName][i] \leftarrow normalizedThreadCpuUsage
14 \cdot
              if TidToCgroup.contains(threadName) then
15:
                  key \leftarrow TidToCgroup.get(threadName)
                  CaroupUsage[key][i]
                                                                    CaroupUsage[key][i]
16
                                                                                                +
    normalizedThreadCpuUsage
17:
              end if
           end for
18:
           prevTime \leftarrow time
19-
       end for
20:
       return CgroupUsage, threadUsage
21:
```

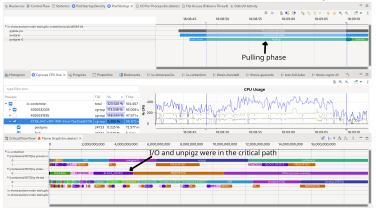
22: end procedure

Pod Lifecycle

	10:26:40	10:27:00	10:27:20	10:27:40	10:28:00	10:28:20	10:28:40	10:29:00	10:29:20	10:29:40	10:30:00	10:30:20	10:30:40	
T user-monoodb	1	-						ScalinaReplica	Set					
▼ user-mangodb-756ccb5d69		_						SuccessfulCre	976					
user-morgodb-756ccb5d69-s7hrf				Pi/litg	0.033	Off Back	óv i	BackOff	1		BackOff	-	-	
" user-review-mongada	_	_						ScalingRep&ca						
▼ user-review-mongodo-98/857c68	_	_						SuccessfulCre						
user-review-mongodb-981857c68-v7tzh				Pulling	8.0	OF B	eckOff		BackOff	1	BackOff			
user-review-redis								ScalingReptic Successfull o						
user-review-redis-/44482b108 user-review-redis-744482b108				Pulling				SICOSSILKO	Started					
user-review redis-/444800/d8-ffw0l				PLAND		-		ScalingState						
V user review-service-77c6456649								Successfully						
user-review-service-77c64566d9-adcka				Bulling	-	auted .	Started 0	State Star		BARROW	Started	Revision		
user-service					_			ScalingRepti						
▼ user-service-74bf5c6c7d		-						Successfulk	reate					
user-service-74bf5c6c7d-87dfp		-		Puling		tarted	Started	130007 524	ted	EackOff .	Started	BackOff		
wrk-client				Pulán	2				Start	ed			Kiin	q
													10000	2012
127.0.0.1 CPU Usage 🕥 Pod Startup Density 🗴 📴 Context S	witch (Incubator)	Flame Gra	aph (incubato	or) 🁌 Flame I	Graph (incubato	r) 🔿 Contro	(FLOW - D)	D Per Process	(Incubator)				۹. ۱	
CPU Usage O Pod Startup Density × 3+ Context S	witch (Incubator)		aph (incubato	or) 🁌 Flame I	Graph (incubato	r) 笋 Contro	R FLOW -D L	O Per Process	(Incubator)					
CPU Usage O Pod Startup Density × 2+ Context S art Time End Time Duration	Pod Name		aph (incubato	x) 🁌 Flame I	Graph (incubato	r) 🖻 Contro	Rowell	D Per Process	(Incubator)					
CPUUsage Pod Startup Density × > Context S art Time End Time Duratio 22:17.965 479 892 10:28:27.500 440 486 69;534;	Pod Name 60,55 wrk-client	-	aph (incubato	x) 🁌 Flame I	Graph (incubato	r) 🗩 Contro	A FLOW OL	0 Per Process	(Incubator)	1				
CPU Usage Pod Startup Density x Context 5 art Time End Time Duratio 27:07.965 479 892 10:28:27.500 440 486 69,534, 27:07.311581 774 10:28:08.138 677 450 60.827, 36	Pod Name 60,55 wrk-client 95,67 promether	JS-V2t8k	_	xr) 🁌 Flame I	Graph (incubato	r) Contro	I Flow 21	O Per Process	(Incubator)	\downarrow	 > т _а		۹	-
CPU Usage Pod Startup Density > Context 5 rrt Time End Time Duratio 27:07.965 479 892 102.82.7500 440 486 69,534, 20,82.7500 440 486 69,534, 27:07.311581774 10.28.08.138 677 450 60,827, 20.28.06 86 768 109 60,188,	 Pod Name 60,55 wrk-client 95,67 promether 94,38 rating-red 	JS-V2t8k IS-67b4d4bd	85-b8pqi	ar) 🁌 Flame I	āraph (incubato	r) Contro	4 Flow PL	D Per Process	(Incubator)	\downarrow	≥Ta	il La	۹	-
CPU Utage Pod Startup Density × 3+ Context int Time End Time Duratio 27.778 65 478 52 70222:750 448 c6 63,534, 27.078 458 158 1774 1028 c6 818 677 450 66,827, 27.08.498 573 722 1028 c68 667 66 109 60,588, 27.09.677 168 279 759,941,	Pod Name 60,55 wrk-client 95,67 promether 94,38 rating-red 65,52 rating-ser	us-v2t8k is-67b4d4bd	85-b8pqi 866-djp4z	x) à Flame I	āraph (incubato	r) 🖶 Contro	(Flow)		(Incubator)	\downarrow	≥Ta	il La	۹	-
CPU Utage Pod Startup Dentify × > Context5 art Time End Time Duratile 27.17.965.479.892 10.28.27.500.440.482 69.534, 27.07.315.61774 10.28.66.818.677.450 60.382, 27.08.957.0718 27.002.08.09.051.897 59.941, 27.09.857.0718 60.22.002.800.90.051.897 59.941, 27.09.857.0718 60.22.002.800.90.051.897 59.941, 27.09.857.0718 60.22.002.800.90.051.897 59.941, 27.09.857.0718 60.22.002.800.90.051 60.25 59.941, 20.20.859.0718 61.22 61.2	Pod Name Pod Name G0,55 wrk-client 95,67 promether 94,38 rating-red 65,52 rating-ser 22,19 review-sto	us-v2t8k is-67b4d4bd vice-5978ff71 vrage-mongo	85-b8pql 366-djp4z 3b-64874-	x) 🁌 Flame I	Sraph (incubato	r) 🖶 Contro	A FLOW Q L	0 Per Process	(Incubator)	\downarrow	≥Ta	il La	۹	-
CPU Usage Ped Startup Density Is - Context vrt Time End Time Duratilio vrt Time Dozz 200 Dozz 466 69,534,1 2/270.3718/5749 Dozz 000 Bord Time Duratilio 2/270.3718/5749 Dozz 000 Bord Time Duratilio 2/270.3718/5749 Dozz 000 Bord Time Duratilio 2/270.4845 T/27 Dozz 000 Bord Time Duratilio 2/270.4845 T/27 Dozz 000 Bord Time Duratilio Duratilio 2/270.4845 T/27 Dozz 000 Bord Time Duratilio Duratilio Duratilio 2/270.4845 T/27 Dozz 000 Bord Time Duratilio	 Pod Name 60,55 wrk-client 95,67 promether 94,38 rating-red 65,52 rating-ser 22,19 review-sto 00,62 review-sto 	us-v2t8k is-67b4d4bd vice-5978ff7f rage-mongo irage-service	85-b8pql 866-djp4z 8b-64874- -6567btc8	ar) 🁌 Flame (Sraph (incubato	r) D Contro	A FLOW Q L	0 Per Process	(incubator)	\downarrow	≥Ta	il La	۹	-
CPU Usage Ped Startup Density Is - Content 51 art Time End Time Duratio 2707.395.479.823 02.827.350.440.446.456.453.41 05.93.41 2707.395.479.827 02.820.66.838.677.450 06.93.21 2708.495.7172 02.820.68.886.761.816 06.93.81 2709.897.195.81 02.820.936.018187 59.44.1 270.398.550.195 02.820.393.0133.45 39.46.3 270.398.520.185 02.820.393.0133.45 39.46.3 277.53.846.03.86 02.820.394.0133.15 59.46.3 272.03.822.828 02.820.394.0133.15 59.46.3 275.53.846.03.86 02.820.394.0133.15 59.46.3	 Pod Name 60,55 wrk-client 95,67 promether 94,38 rating-red 65,52 rating-ser 22,19 review-sto 00,62 review-sto 33,29 text-servio 	is-v2t8k is-67b4d4bd vice-5978ff7f irage-mongo irage-service re-7d8895b6	85-b8pql 866-djp4z 8b-64874- -6567bfc8 54-w8qlg	(x)	5raph (incubato	r) D Contro	4 Flow	D Per Process	(Incubator)	\downarrow	≥Ta	il La	۹	-
CPU Usage Ped Startup Density S - Cented 15 unt Time End Time Duratise 277.7365479 20227.7306474 202267.20040446 270.73651774 20260.8186 677405 60.827 270.848779 20260.8186 677405 60.827 270.848877 20260.8186 677405 60.827 270.98718612 20260.0186 6775 59.41 270.848676 20260.070461370 59.64 271.53146046 22610.027494567 38.610 271.138315880 22610.027494567 59.641	Pod Name 60,55 wrk-client 95,67 promether 94,38 rating-red 65,52 rating-ser 22,19 review-sto 30,62 review-sto 33,29 text-servio 03,01 unique-id-	is-v2t8k is-67b4d4bd vice-5978ff7f irage-mongo irage-service :e-7d8895b6 service-7f7df	85-b8pql 866-djp4z 8b-64874- 6567bfc8 54-w8qig 19797-bt8	conut	Sraph (Incubato	r) 🍚 Contro	a Flow	D Per Process	(Incubator)	\downarrow	≥Ta	il La	۹	-
CTU Usage Or M Starup Density x 1: Context 11 art Time End Time Duration 2777.356.377.4 202.67.350.407.40 0.007.1 2707.371.517.7 202.60.318.67.74.50 0.007.1 2707.071.518.177.4 202.60.318.67.74.50 0.007.1 2707.071.518.177.4 202.60.336.01.75.40 0.007.1 2707.071.518.177.4 202.60.336.01.75.40 0.007.1 2707.071.518.07.00 202.60.37.60 0.017.1 0.017.1 2707.071.518.07.00 202.60.37.60 0.017.1 0.017.1 2707.071.518.07.00 202.00.37.60 0.017.1 0.017.1 2707.078.078.00 202.00.37.60 0.017.017.018.018.01 0.017.1 2707.078.078.00 202.00.37.60 0.017.018.018.01 0.017.1 2707.078.078.00 202.00.37.60 0.017.018.018.01 0.017.018.018.01 2707.078.018.01 202.00.37.60 0.017.018.018.01 0.017.018.018.01 0.017.018.018.01 2707.078.018.01 202.00.37.018.018.018.018.01 0.017.018.018.018.01 0.017.018.018.018.01 0.017.018.018.018.01 <t< td=""><td>Pod Name 60,55 wrk-client 95,67 promethe 94,38 rating-red 65,52 rating-ser 22,19 review-sto 33,29 text-servic 03,01 unique-id- 08,00 nginx-web</td><td>IS-v2t8k IS-67b4d4bd vice-5978ff7l rrage-mongo irage-service re-7d8895b6 service-7f7dt -server-7c58</td><td>85-b8pql 866-djp4z 85-64874- -6567btc8 54-w8qlg 19797-bt8 9465b8-x</td><td>xr) 🍓 Flame (</td><td>Sraph (Incubato</td><td>r) 🔿 Contro</td><td>A FLOW</td><td>0 Per Process</td><td>(Incubator)</td><td>\downarrow</td><td>≥Ta</td><td>il La</td><td>۹ </td><td>-</td></t<>	Pod Name 60,55 wrk-client 95,67 promethe 94,38 rating-red 65,52 rating-ser 22,19 review-sto 33,29 text-servic 03,01 unique-id- 08,00 nginx-web	IS-v2t8k IS-67b4d4bd vice-5978ff7l rrage-mongo irage-service re-7d8895b6 service-7f7dt -server-7c58	85-b8pql 866-djp4z 85-64874- -6567btc8 54-w8qlg 19797-bt8 9465b8-x	xr) 🍓 Flame (Sraph (Incubato	r) 🔿 Contro	A FLOW	0 Per Process	(Incubator)	\downarrow	≥ Ta	il La	۹	-
CPUUlagin Pad Startsp Demity × 1 > 6 content Init Time End Time Duration 2177.956.478 928.272.096.4444 69.354. Duration 2177.956.478 928.272.096.4444 69.354. Duration 2177.956.478 928.272.096.4446 69.354. Duration 2177.956.478 927.958.458.978 69.079. Duration Duration 2177.956.478 92.000.978 87.978.018.079.078 91.079.941.079. Duration Duration 2177.956.068 92.000.978.917.99.179.0178.0178.019.0178.0178.0178.0178.0178.0178.0178.0178	Pod Name 60,55 wrk-client 95,67 promethe 94,38 rating-red 65,52 rating-ser 22,19 review-str 33,29 text-servis 33,29 text-servis 08,00 ngirs-web 80,93 plot-servis	IS-V218k IS-67b4d4bd vice-5978ff7l rage-mongo irage-service re-7d8895b6 service-7f7dt -server-7c58 ie-5cccc4754	85-b8pql 866-djp4z 86-64874- -6567btc8 54-w8qlg 19797-bt8 9465b8-x -8r7pb	bri a Flame I	Sraph (incubato	r) 🗩 Contro	A FLOW	ID Per Process	(Incubator)	\downarrow	≥Ta	il La	۹	-
CPU Usage Ped Startup Density S - Cented 15 unt Time End Time Duratise 277.7365479 20227.7306474 202267.20040446 270.73651774 20260.8186 677405 60.827 270.848779 20260.8186 677405 60.827 270.848877 20260.8186 677405 60.827 270.98718612 20260.0186 6775 59.41 270.848676 20260.070461370 59.64 271.53146046 22610.027494567 38.610 271.138315880 22610.027494567 59.641	Pod Name 60,55 wrk-client 94,38 rating-red 65,52 rating-red 65,52 rating-ser 22,19 review-sto 30,65 review-sto 30,91 unique-id- 80,00 nginx-web 80,93 plot-servie 80,60 plot-servie	us-v2t8k is-67b4d4bd vice-5978ff7l rrage-mongo rrage-service re-7d8895b6 service-7f7dt -server-7c58 te-5cccc4754 odb-8d445fd	85-b8pql 366-djp4z 3b-64874- 6567bfc8 54-w8qig 19797-bt8 9465b8-x -8r7pb b9-memh	Count	Sraph [incubato	r) D Contro		Der Process	i (Incubator)		≥Ta	il La	ten	-

Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

Diagnosing Slow Pod Startup Issues Attributable to I/O Contention

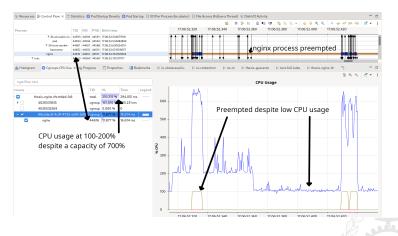


Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

CPU usage per cgroup and pod

	s 📑 Control Flow 🔟 Statistic							17:06:35		17:06:40	₩ 11 17:06:45	63	R B B * 17:06:50		ब ् ब् 17:06:55	đ •
osis-ngina-	throttled-Aul@ust/uid/0/64-bit												-		-	
ngiro. 7 naice:775	CANG						ScalingR									
	fic5bbblic-4pqhp						Pulling									
												_	-	-	_	-
Histogran	n 🔕 Cgroups CPU Usa 🗙	R Progress	Propert	ies 🛄 Be	ookmarks	E lo-sl	iowcase/lo-	E lo-contention	E no-lo	E thesis-guarante	🗄 test-full-k.	ibe E	thesis-nginx-th			
														6	e, e,	đ •
pe filter t	ext									CPU Usa	ge					
ess		TID	96	Time	Legend											
	thesis-nginx-throttled-full	total	56.255.95	16.066 s		300										
	4026531835		48.809 %	13.939 s												
	4026532269			67.875 ms												
~	88e2dac8-4c9f-4725-a50f	2d0f7 cgroup	3.015 %	861.172 ms	-	250										
	10-listen-on-ip	44589	0.006 %	1.751 ms				1								
	20-envsubst-on-	44593	0.002 %	672.217 µs					- I							
	20-envsubst-on-	44591	0.005 %	1.645 ms		200										
	30-tune-worker-	44601	0.005 %	1.354 ms				1					1			
	awk	44594	0.005 %	1.534 ms		P	P 1 1						1.1.1.			
	basename	44602	0.005 %	1.525 ms		5 150										
~	docker-entrypoi	44588	0.004 %	1.266 ms									111/1			
~	find		0.007 %	1.876 ms											11 1	
•	nginx	44616		818.178 ms		100		2								
	nginx	44571	0.097 %	27.807 ms												
	runc:[2:INIT]		0.000 %	83.191 µs				1111				1.1	al III - I I			
~	runc:(2:INIT)		0.001 %	279.536 µs		50						'M U	MINI VILA			
2	runc(2:INIT)	44572		1.442 ms		50						IIT		1 10	M W	T
	runc:[2:INIT]		0.000 %	113.694 µs										' 11	4 III	
~	runc:[2:INIT]		0.001%	285.403 µs		0	New	U WNLNI	ルレル	شالله الماله	15JWWW	Wh 1	MANN	WWW	M	ЛY
~	sort	44587	0.005 %	1.359 ms		0	1									

Diagnose latency due to limits



Conclusion

- Demonstrates how tracing aids in observing the internal states of the orchestrator
- 2 Diagnosing issues at both the orchestrator and kernel levels
- **8** Highlights the utility of multi-level tracing for diagnostic purposes



Conclusion

Future work

- Reuse methodology for other resources: Network, memory, disk, etc.
- 2 Use the trace data to characterize workload and improve scheduling

Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr

Conclusion

Thank you! Questions?



Multi-Level Tracing of Containerized Application Orchestrator - Benjamin Saint-Cyr