

Performance analysis of DPDK-based applications

Adel Belkhiri Michel Dagenais May 15, 2020

> Polytechnique Montréal Laboratoire **DORSAL**

Agenda

Introduction

- Linux Kernel bypassing
- What is DPDK ?
- Motivations and goals

Investigations and preliminary results

Use cases

Conclusion and future work



Linux Kernel bypass

- NICs are getting faster and faster : 10Gbps, 100Gbps, etc.
- Linux kernel network stack prevents packets from being processed quickly.
 - Costly context switches and system calls (read, write, etc.)
 - Huge *skb_buff* data structure
 - Interrupts and NAPI (New API)
 - Lack of batching
- Several network stack bypass solutions : PF_RING/DNA, DPDK, PacketShader, OpenOnload, RDMA/IBverbs etc.

What is DPDK ?

- Intel DPDK (Data Plane Development Kit).
- Open source networking framework written in C, supporting a wide range of NICs and processors.
- Higher levels of packet processing throughput via Kernel bypassing.
 - Processor affinity
 - Huge pages
 - Lockless ring buffers
 - Poll Mode Driver
 - Batch processing of packets (burst)



Source : https://blog.selectel.com /introduction-dpdk-architecture-principles/

Motivation and Goals

Motivation

 Incapacity of existing tools to monitor NICs that are managed by DPDKbased applications.

Goals

- Leverage tracing techniques to analyze the performance of DPDKbased applications.
- Shed light on the potential causes of packet processing latencies.
- Analyze the cost of tracing and its impact on frame processing performance

Work Environment

- Software :
 - DPDK (version 19.05)
- Data Collection :
 - LTTng (version 2.10)
 - Userspace tracing / static instrumentation
- Performance Analyses :
 - Trace Compass framework





DPDK Architecture



Source : www.dpdk.org

Vhost-user library (1)



Source :

https://www.redhat.com/en/blog/journey-vhostusers-realm

Vhost-user library (2)

- How to identify which entity (Host or Guest) is responsible for a TX/RX performance degradation ?
- How to measure the rate of enqueuing/dequeuing Mbuff to/from each queue ?



Introduction

Use Cases

Conclusion

- Experiment setup :
 - Run dpdk-testpmd in the host.

\$ sudo dpdk-testpmd -I 0,1 --socket-mem=1000 -n 1 \
 -- vdev="net_vhost0,iface=/tmp/vhost-user1" \
 --vdev="net_vhost1,iface=/tmp/vhost-user2" -- ...

- Configure the guest to connect to the created virtual devices.
 - <interface type='vhostuser'> <mac address='56:48:4f:53:54:01'/> <source type='unix' path='/tmp/vhost-user1' mode='client'/> <model type='virtio'/> <driver name='vhost' rx_queue_size='256' tx_queue_size='256'/>

</interface>



Figure : Host is sending packets to the guest.

(UP) Rate of MBuff enqueuing. (DOWN) Percentage of TX queue occupancy.



Figure : Guest is sending packets to the host.

(UP) Rate of MBuff dequeuing. (DOWN) Percentage of RX queue occupancy.

*Zoom into the previous figure



Logical Cores

• The term "lcore" refers to an EAL pthread pinned to a CPU core. "EAL pthreads" are created by EAL to execute the tasks issued via *remote_launch* functions.

Introduction



Use Cases

Conclusion

Service Cores

- DPDK has support for a new dynamic way of executing workloads on DPDK lcores.
- Service
 - Runnable work item
 - Runs an iteration of work then returns
- Service Core
 - Dedicated core to running services. These services are scheduled in a simple round-robin run-to-completion.



If there are many services running on a core this could potentially lead to high waiting times for some of the services.

	File	Edit Navigate Projec	t Run File	e Tools	Window Help Window H	elp											
	1	Quick Acc Quick Acc															k Access 🔡 😰 🔁
		g															- 8
	ß	Timestamp	Channel	CPU	Event type		Contents			Binary Location	Function Locatio	n Source Location					
		🖋 <srch></srch>	<srch></srch>	<srch></srch>	<srch></srch>		<srch></srch>			<srch></srch>	<srch></srch>	<srch></srch>					
	물리	00:42:44.515 991 272	channel0	1 1	librte_eal:thread_lcore_run	ining	lcore_id=1, f=0x7f	1ed857e1e1, arg=0x0									
		00:42:44.515 994 549	channel0_	1 1	librte_eal:thread_lcore_wai	iting	lcore_id=1, ret=0,	status=2									
		00:42:44.516 059 174	channel0	0 0	librte_eal:thread_remote_la	aunch	f=0x7f1ed857e1e1	I, arg=0x0, lcore_id=2, r	c=0								
		00:42:44.516 080 290	channel0	2 2	librte_eal:thread_lcore_run	ining	lcore_id=2, f=0x7f	1ed857e1e1, arg=0x0									
		00:42:44.516 082 975	channel0_	2 2	librte_eal:thread_lcore_wai	iting	lcore_id=2, ret=0, :	status=2									
		00:42:44.516 088 979	channel0	0 0	librte_eal:thread_remote_la	aunch	f=0x7f1ed857e1e1	I, arg=0x0, lcore_id=3, r	c=0								
		00:42:44.516 153 126	5 channel0_	3 3	librte_eal:thread_lcore_run	ining	lcore_id=3, f=0x7f	1ed857e1e1, arg=0x0									
		00:42:44.516 155 707	channel0	3 3	librte_eal:thread_lcore_wai	iting	lcore_id=3, ret=0, s	status=2									
		00:42:44.520 525 551	channel0_	0 0	librte_eal:service_compone	ent_register	r id=0, service_name	e=service_1, cb=0x5650	d2ba6ee4, arg=0x0								
		00:42:44.520 530 338	3 channel0_	0 0	librte_eal:service_compone	ent_register	r id=1, service_name	e=service_2, cb=0x5650	d2ba6ee4, arg=0x0								
		00:42:44.520 532 231	channel0_	0 0	librte_eal:service_compone	ent_register	r Id=2, service_name	e=service_3, cb=0x5650	d2ba6ee4, arg=0x0								
		00:42:44.520 533 996	channel0_	0 0	librte_eal:service_compone	ent_register	r Id=3, service_name	e=service_4, cb=0x5650	d2ba6ee4, arg=0x0								
		00:42:44.520 535 648	s channelo	00	librte_eal:service_compone	ent_register	r Id=4, service_name	e=service_5, cD=0x5650	d2ba6ee4, arg=0x0								
		00:42:44.520 542 242	channel0_	00	librte_eallicore_role_chang	e	lcore_id=1, lcore_i	ole=2									
		00.42.44.520 545 521	channelo_		ibite_eai.service_icore_rea	iuy '			-								
		🔕 Tasks 🔚 State System	m Explorer	23											13	10 5 3 5 5 5 - 0 0	, , , , , , , , , , , , , , , , , , ,
Master		State System / Attribute	e		 Quark 	C	00:42:44.520	00:42:44.530	00:42:44.540	00:42:44.550	00:42:44	1.560 00:42:4	4.570	00:42:44.580	00:42:44.590	00:42:44.600	
		▼ LCores			0		n	•	+					1			
	•	▼ 0			1 (
		role			2						L	CORE_RTE					
Core	•	func			4	_						IDLE					
		▼1			13												
	4	role			14	LCORE_RTE				I		LCORE_SERVICE					
		status			15	IDLE						RUNNING 7f1ed856700c					
		▼ services			17							Theosonote					
		▼ 0			18												
Service		name			19		cup ponding		un poodiag			service_1	run pending in	un pooding		andina (run) - pandina (
		¥ 1			20		ron pending	run pe run penuing r	un pending	Tun pe Tun (pending Tun pe		un pending	Tun pe Tun p	ending run pending	Ton pe Ton
Cores		name			22							service_2					
		status			23		pe run run	pending (run) pendi	ng (run p e run	pending 👔	un pending run pe	e run pending	run pendi	ng (run p e (run	pending ru	n pending run pe run	pending 📲
		₹2			24							service 3	_				
		status			26		pending run p	e run pending run	pending run pe ru	un pe run pendi	ng run pending r	un pe run pe run	pending run	pending run pe ru	in pe run pendin	g run pending run per	run pe run pend
		₹ 2			5												
		role			6	LCORE_RTE						LCORE_SERVICE					
		func			8	IDLE			-			7f1ed856700c					
	,	▼ services			27												
		▶ 0			28												
•		▶ 1			31												
	T: 20	020-05-14 00:42:44.520581	1241 T1: 2	020-05-14	00:42:44.520515060												

Figure : Execution of "dpdk-service_cores" sample application and illustration of the distribution of service executions across "service cores".

• Execution of "dpdk-testpmd" with a master core and two lcores.



Conclusion

- DPDK is one of the most important open-source Linux projects ^[1] and many successful projects depend on it : OVS-DPDK, FD.io VPP, PfSense, TREX, etc.
- Tracing is an efficient technique to extract low-level performance data and solve many performance issues : multi-core synchronization issues, latency measurements, etc.
- A Native DPDK CTF trace support has been added to release 20.05^[2].
 - No dependency on any third-party library.
 - Ability to trace on Windows platforms.



Future Work

- Continue the instrumentation of the most popular DPDK libraries (eventDev, LPM, ACL, ...)
- Refine the instrumentation in the DPDK packet processing datapath to identify possible improvements.
- Develop more comprehensive analyses.



Questions?

adel.belkhiri@polymtl.ca



POLYTECHNIQUE MONTREAL – Adel Belkhiri

Reference :

[1] https://www.linuxfoundation.org/projects/
[2] https://doc.dpdk.org/guides/prog_guide/trace_lib.html
[3] https://blog.selectel.com/introduction-dpdk-architecture-principles/
[4] www.dpdk.org
[5] https://www.redhat.com/en/blog/journey-vhost-users-realm

