

Janus: Lightweight Container Orchestration for High-performance Data Sharing

Ezra Kissel SNTA'22 June 30th 2022





Overview and motivation

- Develop a managed data movement service capability
 - Support a pool of transfer software images that "just work"
 - Reduce reliance on varying levels of network/system expertise for deployments
 - Enable automated operations on Data Transfer Nodes (DTNs)
- Make use of containerization supported by lightweight orchestration
- Target high-speed data transfer deployments with dual-stack and multihomed networking requirements
- Evaluate container networking with data transfer tools used in R&E nets



Data Transfer Nodes

- DTNs fulfil a key role in enabling high-performance data movement
- Deployment models well-understood
 - There exists a wealth of knowledge in building, tuning, troubleshooting and maintaining DTN hardware and software
- Goal: expand availability, explore new usage models, and meet future demands → DTN-as-a-Service





Janus concept

- 1. Data mover software in containers
- 2. Network and storage performance optimization
- 3. Configuration and tuning flexibility
- 4. Lightweight service orchestration





Extensible profiles

- Provide common configuration sets for service containers
- Helpful for consistency and re-use for larger deployments
- Specify capabilities once, then apply often

		cpu:	В
ID : Status Nodes/Se 3 : STOPPED lbl-dev- 5 : STARTED chic-cac 6 : STARTED bost-cac	rvices Image dtn [None] wharf.es.net/dtnaas/openscier hel [None] wharf.es.net/dtnaas/openscier hel [None] wharf.es.net/dtnaas/openscier	Profile affin: cegrid/cms-xcache:fresh macvlan2x mgmt_j cegrid/cms-xcache:fresh chic-cms-xcache0l data_j cegrid/cms-xcache:fresh bost-cms-xcache0l data_j	ity: netwo net: bridg net:
janus>	age dipass/ofed profile lbpl 400g 1	name	a: net3002
Janus> Session create tont-ton-1 im	aye uthaas/ored profile tont-400y-1		10.33.2.2

features: rdmacap: devices: - devprefix: "/dev/infiniband" names: - rdma cm - uverbs caps: - IPC LOCK - NET ADMIN limits: - Name: memlock Soft: -1 Hard: -1 profiles: lbnl-400g-1: cpu: 4 affinity: network mgmt net: bridge data net: name: net3001 eth200 ipv4 addr: -10.33.1.20ipv6 addr: -2001:400:2202:2191::3features: - rdmacap privileged: false lbnl-400g-2: rk e eth200 0 10.33.2.21 features: - rdmacap privileged: false net volumes: - data

Linux virtual networking and containers



...and ipvlan, bonds, teams, etc.

ESnet

¹ <u>https://developers.redhat.com/blog/2018/10/22/introduction-to-linux-interfaces-for-virtual-networking/</u>
² <u>https://community.mellanox.com/s/article/Docker-RDMA-SRIOV-Networking-with-ConnectX4-ConnectX5-ConnectX6-Connect</u>

Sources:

Example data transfer node configuration

- 1. Multi-homed physical nodes
- 2. Slow path control
- 3. Fast path data plane
- 4. Dual-stack IP networking
- 5. Local agents for resource discovery and customized tuning





Container networking performance

- A number of container network attachments are supported and available
 - Which works best for high-performance networking in each deployment?
- Relatively easy to achieve matching performance using tuning params
 - E.g., container execution mapped to appropriate NUMA nodes for cores and memory





100G links, GridFTP, forking server, 5 clients with -p4 parallelism

Evaluation topology (ESnet testbed)



ESnet

Evaluation environment

- Performant data movers: EScp, GridFTP, mdtmFTP, Zettar zx
- Janus Python client and CLI
 - Programmatic execution
 - Jupyter Notebook scripting
- Standard 100G host and network interface tuning

0		Kernel Tabs	s Settings Holp
	+ m ±	c	💌 400g. dema.tpynb 💿 🕱 ESCPevel.py X
_			B + X (C C) + B C +> Coop ~
0		<u> </u>	- uverbs
	•/		- IPC_LOCK
IP	Name A Las	st Moaned	- NET ADMIN
	• 400g_dem 20	0 days ago	- Name: memlock
≣	👌 client_test.py 9 m	nonths ago	Soft: -1 Hard: -1
		0 days ago	
*	ESCPeval.py 20	0 days ago	profiles: tbnt-400g-1:
	SC21Dem 7 m	nonths ago	cpu: 4 affinity, petynek
			ngmt_net: bridge data net:
			nane: net3001_eth200
			10/33.1.20
			10.33.1.21
			- rda-deno
			privileged: false
			lbnl-400g-2:
			cpu: 4 affinity: network
			serv port range: mull
			ngmnet; data_net;
			name: met3002_eth200 inv4_addr:
			10.33.2.20
			- 10.33.2.21 features:
			- rdna-demo
			lbnl-400g-host: cou: mull
			men: null
			serv_port_range: nucl ngnt_net: host
			privileged: false
			vounics. - data
			- tap - proc
			Les Investor
			<pre>inport the from janus_client import Client, Service</pre>
			from ESCPeval import setup, run_job, stop_job
			JANUS IBI -"https://persr.srv.l testhed100_es_net:5650*
			user = kisset: mykey = 'sSh-rsa
			AAAABNaciyc2EAAADAAABAAABAAC/DAAkhubakfj05Y667twwtr57Roig-0AAShithMik7Caha4gTv/C3ri1DH3FKR2J90AJVSissj18tHp1ELDHXCYR02dH74B3PcPd2xqL Ry5PubAKV56ci8trY1iBabK1KM023d92CVFKYe4098b6C(Kkx597cu0.00YCGR/H7BK12TX7ziFAHA806TVH861y51eb40E2DPH3b1DH3Bc/sK72D3h27L2F2FD5A7
			<pre>client = Client(JAMUS_URL, auth=(user, 'd3m8p4ss'))</pre>
			<pre>sess_rdma1 = client.getSession()</pre>
			<pre>srv1 = Service(instances=['nersc-tbn-6'], image='dtnaas/ofed:S.4-3',</pre>
			<pre>srv2 = Service(instances={'nersc-tbn-7'], image='dtnass/ofed:5.4-3', profile='lbn -400e.2', usernamesuser, public keyamykey)</pre>
			sess_rdmal.addService(srv1)
			sess rdma2 = rlient netSession()
			<pre>srvl = Service(instances([nersc-srv-1]], image=dtnaas/ofed:5.4-3',</pre>
			<pre>prorite="tbht.400g.1", username=user, public_key=nykey) srv2 = Service(instances=('nersc-dtnaas-1'], inage='dtnaas/ofed:5.4-3',</pre>
			profile='lbl.4089-2', username=user, public.key=nykey) sess rdm2_adfsrvice(srv1)
			sess_rdma2.addService(srv2)
			<pre>sess_tcp = client.getSession()</pre>
			<pre>Srv = Service(instances=1'nersc-tbn-1', 'nersc-tbn-2'],</pre>
			profile='lbnl-400g-host', username=user,
			public_key=mykey) sess trn =#d5enu(relsru)
5	imple 🔵 👘 0 🚛 2 🕲		

Storage baseline



elbencho storage sweep, sequential write benchmark

https://github.com/breuner/elbencho/tree/master/contrib/storage_sweep



Container network comparison (LAN)



Each src/dst container instance launched by Janus with specified net profile

SR-IOV provides best performance compared to native



CPU usage comparison



CPU overheads are a non-trivial factor, destination usage shown

Bridge networking incurs the highest overheads



Data mover comparison over WAN



Performance of various transfer software over 88ms WAN link

Transfer parallelism approach has a large impact across file size distributions



Conclusion and Future work

- Containers for high performance data movement is a viable approach
- 200G/400G evaluation
- Extend tuning automation
- Provide guidance for best practice and common concerns
 - Performance, security, network orchestration, measurement and monitoring





Thank you

Questions?

Ezra Kissel

kissel@es.net

