Hewlett Packard Enterprise

TRELLIS

AN ANALYTICS FRAMEWORK FOR UNDERSTANDING SLINGSHOT PERFORMANCE

Madhu Srinivasan

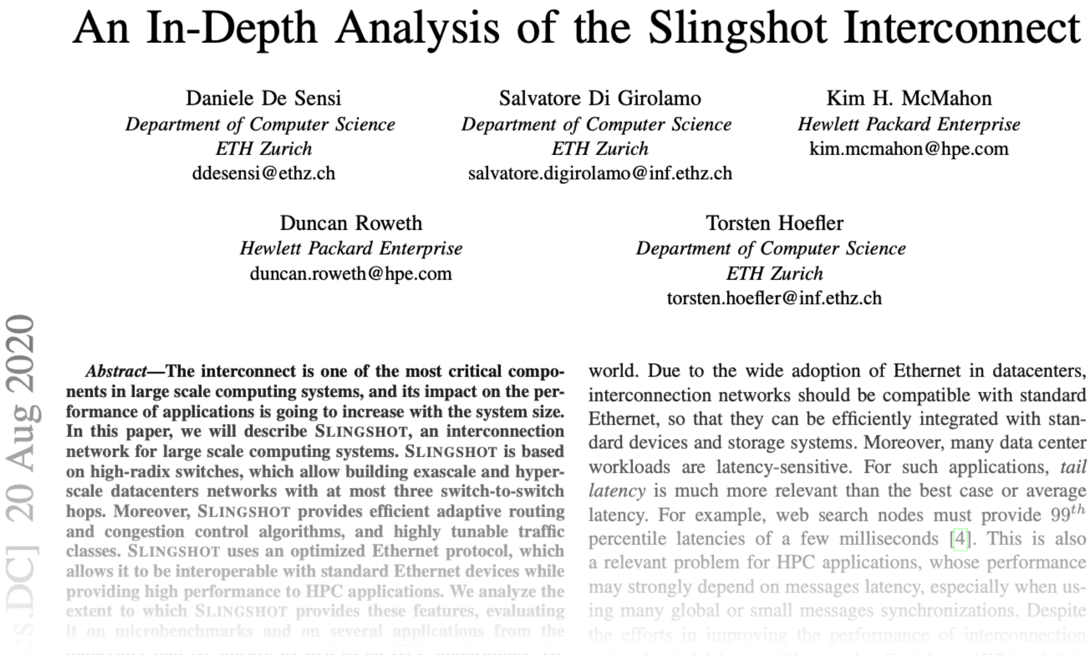
HPE Cray

Cray User Group Conference

May 4, 2021

MOTIVATION

- Application Performance can be sensitive to Network
  - Topology
  - Routing and Congestion Control
- Slingshot
  - Adaptive Routing
  - Advanced Congestion Control
- **Problem:** *"What is happening in the network?"*
- **Solution:** Monitor the HPC Interconnect
  - At the global fabric level
  - At the application level



An In-Depth Analysis of the Slingshot Interconnect

Daniele De Sensi
Department of Computer Science
ETH Zurich
ddesensi@ethz.ch

Salvatore Di Girolamo
Department of Computer Science
ETH Zurich
salvatore.digirolamo@inf.ethz.ch

Kim H. McMahon
Hewlett Packard Enterprise
kim.mcmahon@hpe.com

Duncan Roweth
Hewlett Packard Enterprise
duncan.roweth@hpe.com

Torsten Hoefler
Department of Computer Science
ETH Zurich
torsten.hoefler@inf.ethz.ch

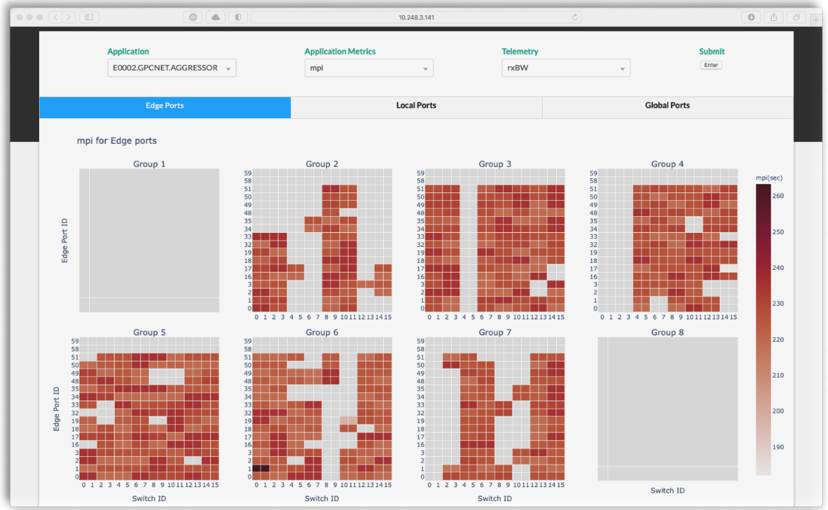
***Abstract*—The interconnect is one of the most critical components in large scale computing systems, and its impact on the performance of applications is going to increase with the system size. In this paper, we will describe SLINGSHOT, an interconnection network for large scale computing systems. SLINGSHOT is based on high-radix switches, which allow building exascale and hyperscale datacenters networks with at most three switch-to-switch hops. Moreover, SLINGSHOT provides efficient adaptive routing and congestion control algorithms, and highly tunable traffic classes. SLINGSHOT uses an optimized Ethernet protocol, which allows it to be interoperable with standard Ethernet devices while providing high performance to HPC applications. We analyze the extent to which SLINGSHOT provides these features, evaluating it on microbenchmarks and on several applications from the**

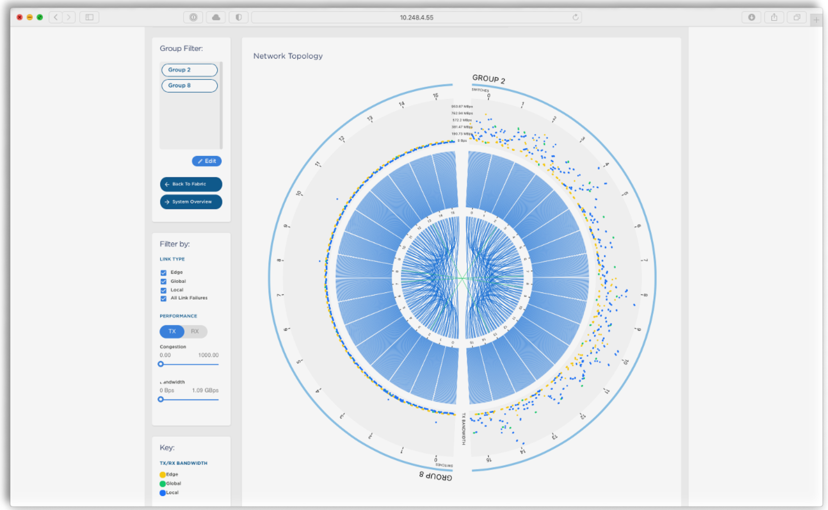
world. Due to the wide adoption of Ethernet in datacenters, interconnection networks should be compatible with standard Ethernet, so that they can be efficiently integrated with standard devices and storage systems. Moreover, many data center workloads are latency-sensitive. For such applications, *tail latency* is much more relevant than the best case or average latency. For example, web search nodes must provide 99^{th} percentile latencies of a few milliseconds [4]. This is also a relevant problem for HPC applications, whose performance may strongly depend on messages latency, especially when using many global or small messages synchronizations. Despite the efforts in improving the performance of interconnection

Daniele De Sensi et.al. SC20: International Conference for High Performance Computing, Networking, Storage and Analysis

INTRODUCTION : TRELLIS

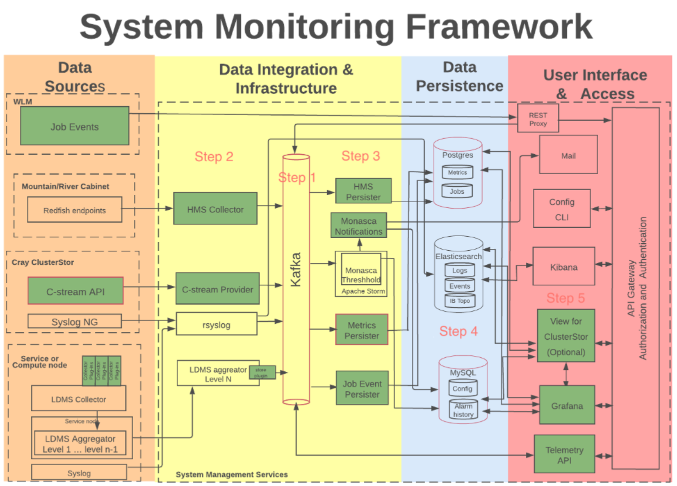
- Analytics Framework to Observe and Understand Network Performance
    - Initial target - Slingshot
- With *trellis*, you can
    - Infer topology and layout
    - Get telemetry as timeseries dataframe
    - Get aggregated, thresholded telemetry
    - Map Job Characteristics to Topology/Telemetry
- Build customized, targeted workflows
- Design UIs for a broader use-case





INTRODUCTION : WHY TRELLIS ?

Raw Telemetry

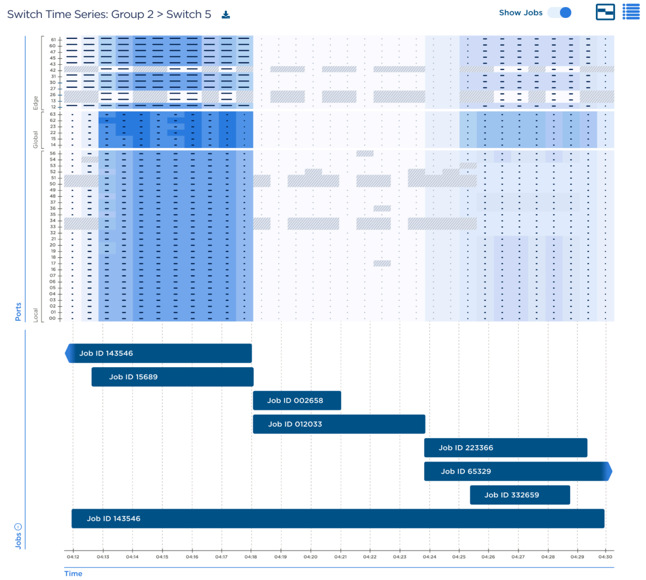


Exploring New Monitoring and Analysis Capabilities on Cray's Software Preview System. Jim Brandt et.al., 2019

~160,000 metrics @1Hz (1024-node Cray EX System)

DISCONNECT

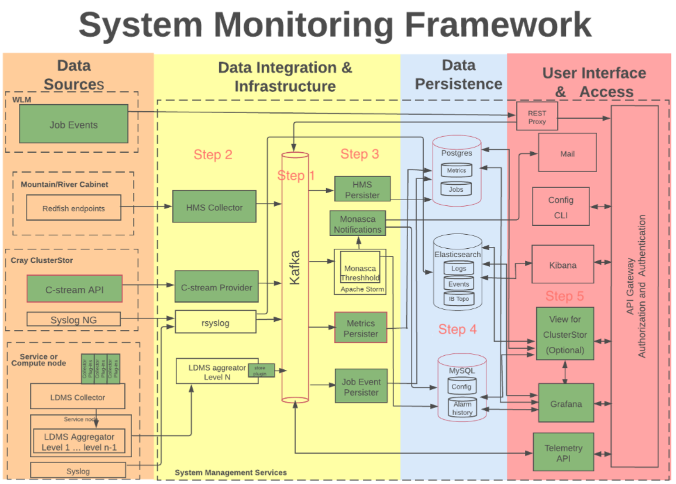
Actionable Insight



Application Performance

INTRODUCTION : WHY TRELLIS ?

Raw Telemetry

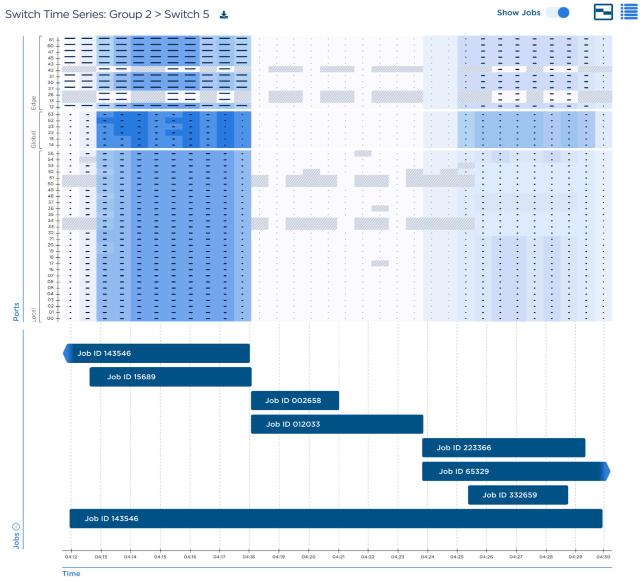


Exploring New Monitoring and Analysis Capabilities on Cray's Software Preview System. Jim Brandt et.al., 2019

~160,000 metrics @1Hz (1024-node Cray EX System)

TRELLIS

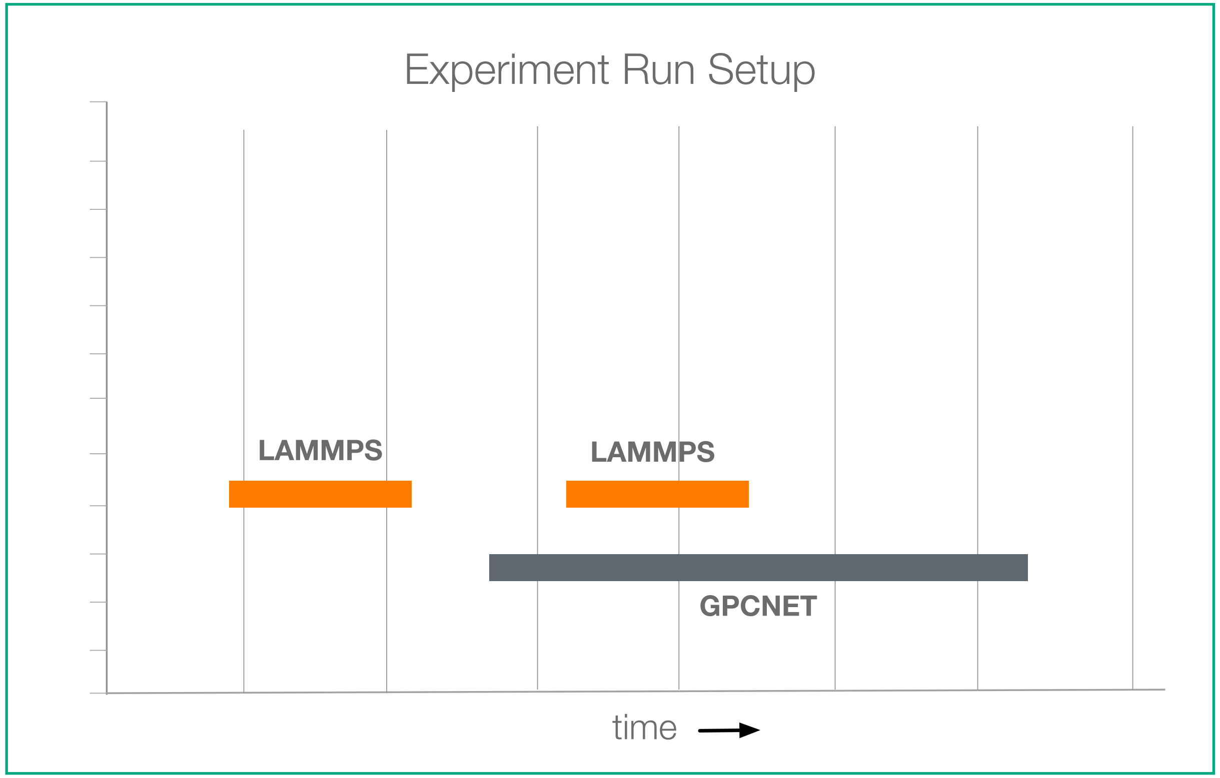
Actionable Insight



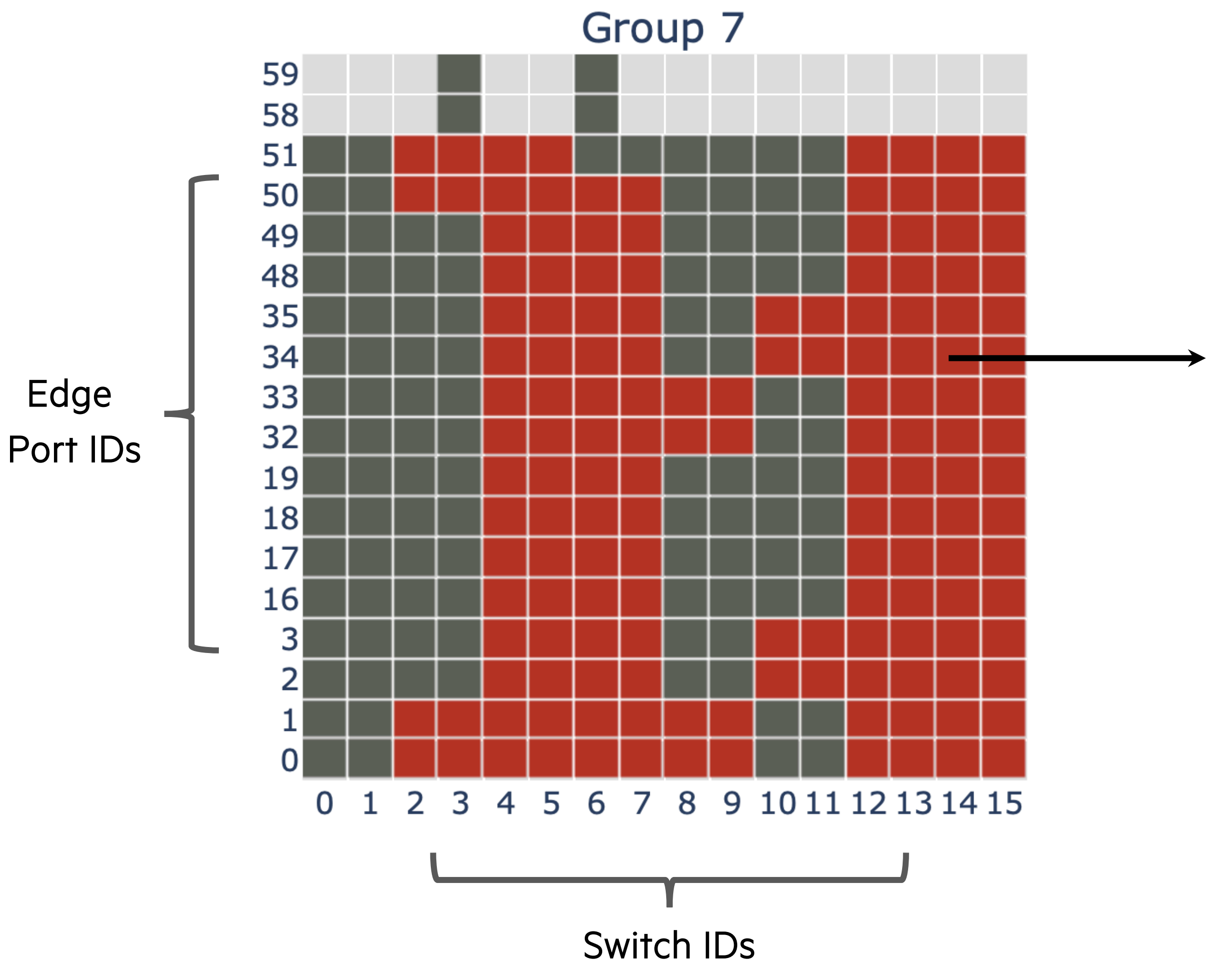
Application Performance

EXAMPLE : MONITORING AT THE APPLICATION LEVEL

- Goal
  - Use trellis with an instrumented application
  - Recognize communication patterns
- Application
  - GPCNET (128 PPN)
  - Pair-wise all-to-all and Incast pattern
- System: “Shandy” (version 1.3)
  - 1024 Node, dual-socket, EPYC Rome
  - Network Topology
    –8 Groups, 16 Switches per Group
    –128 Nodes per Group
    –Dual Mellanox ConnectX-5 NICs per Node



EXAMPLE : VISUALIZATIONS



Edge Port connected to compute node

`allocation:` ports used by GPCNET

`mpi:` avg. MPI latencies in GPCNet

`rxBW:` avg. bandwidth received (MBps)

`txBW:` avg. bandwidth transmitted (MBps)

`rxBlocked:` avg. frames blocked (per second)

Slingshot Analytics

Application: E0001.GPCNET.AGGRESSOR

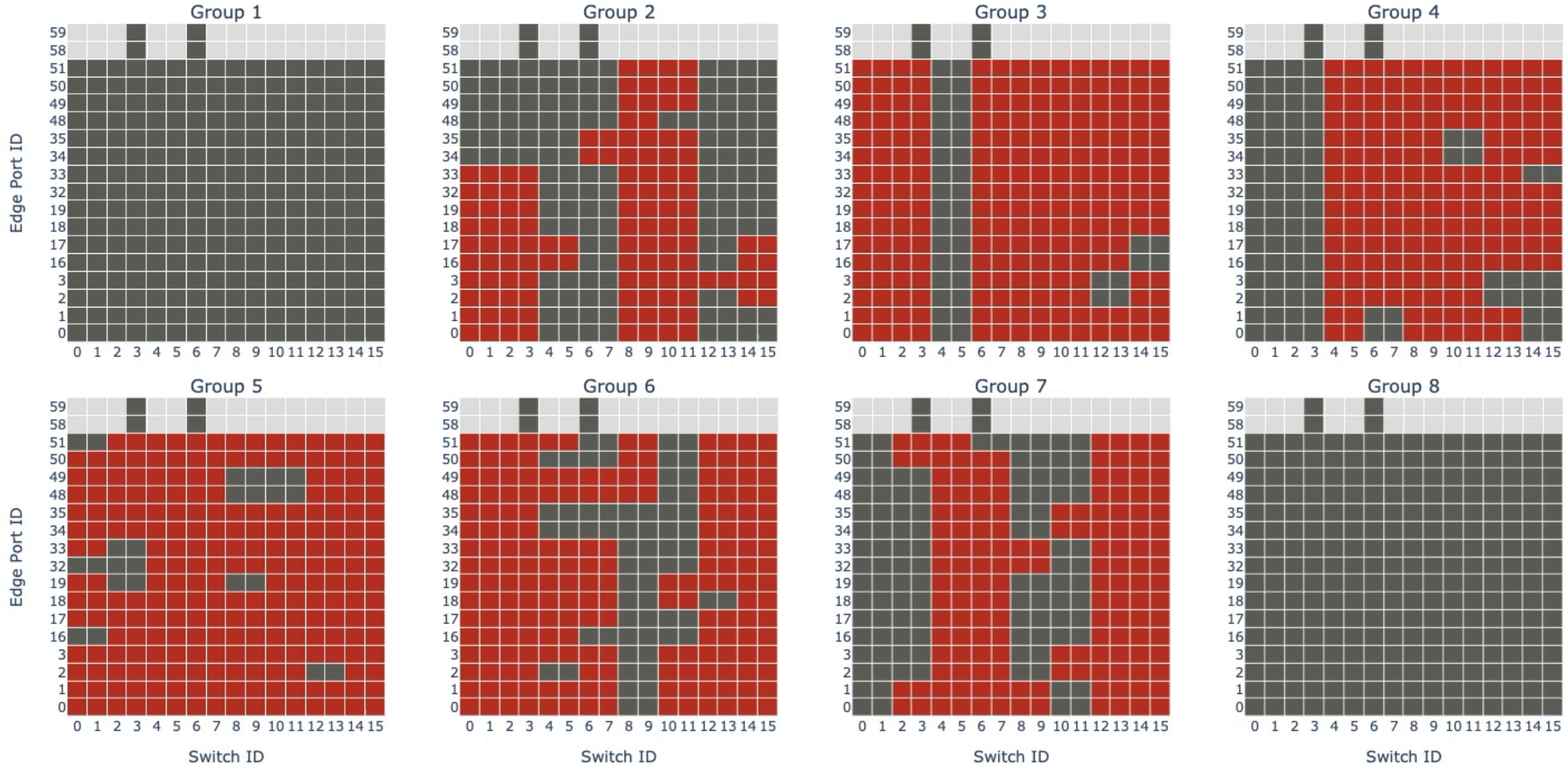
Application Metrics: allocation

Telemetry: rxBW

Submit: Enter

Edge Ports	Local Ports	Global Ports

allocation for Edge ports



Slingshot Analytics

Application: E0001.GPCNET.AGGRESSOR

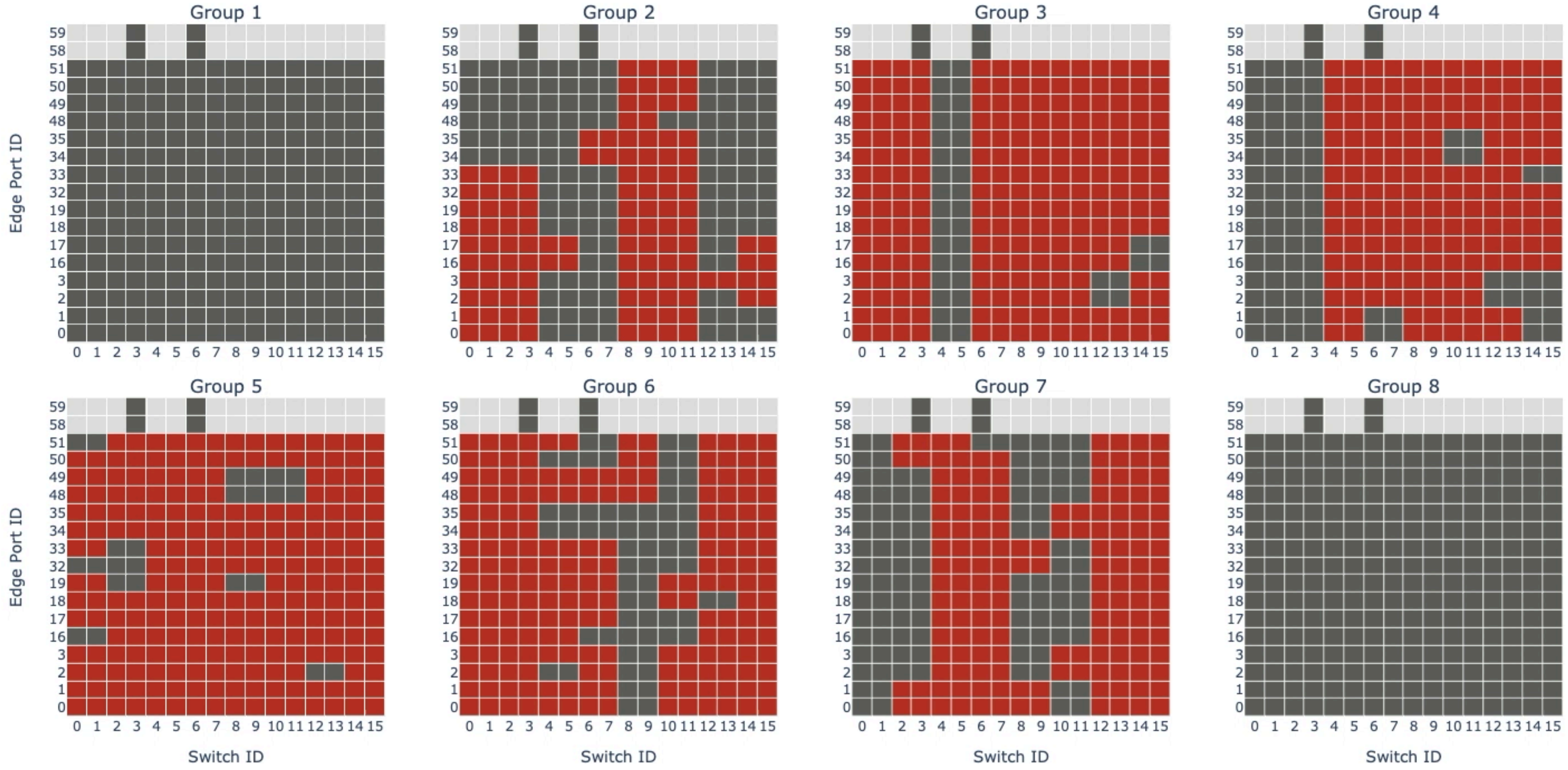
Application Metrics: allocation

Telemetry: rxBlocked

Submit: Enter

Edge Ports | Local Ports | Global Ports

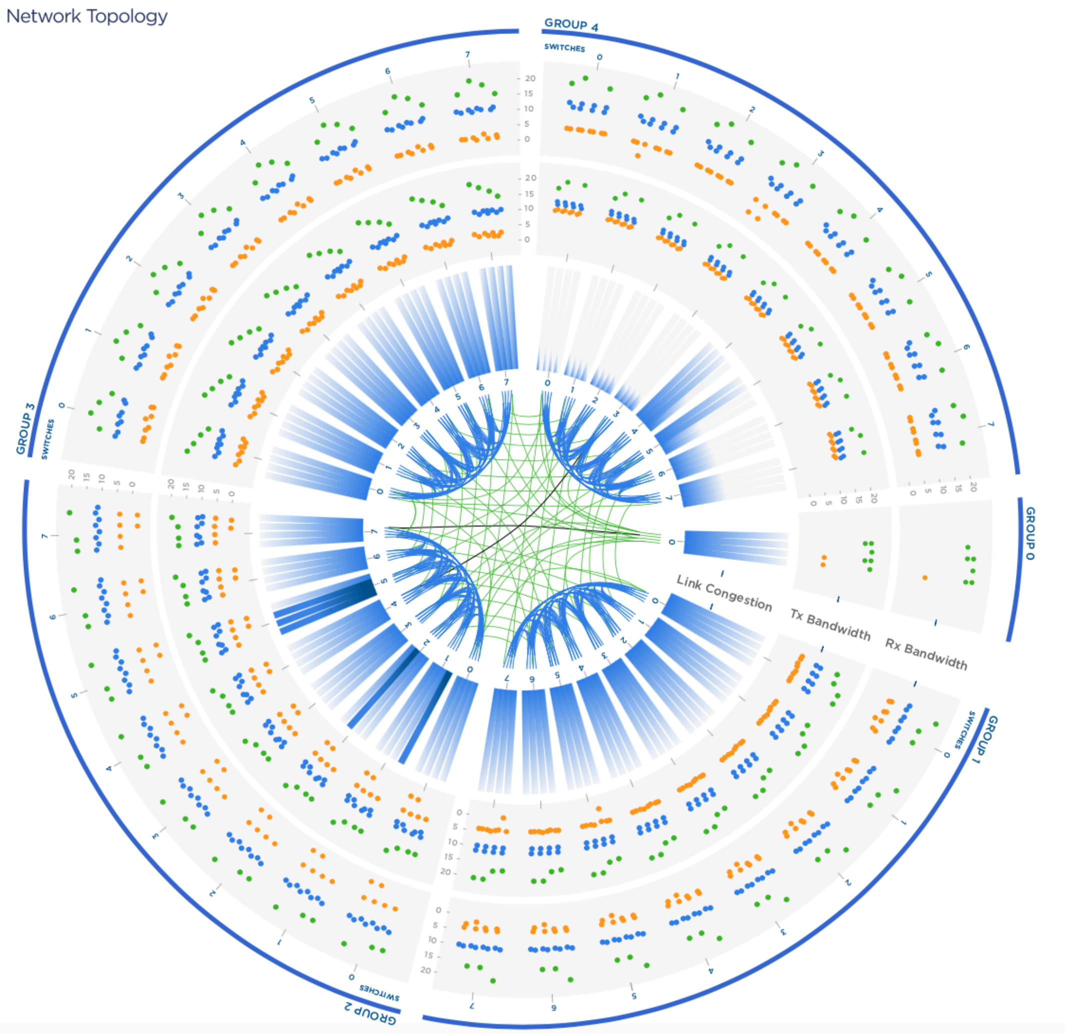
allocation for Edge ports



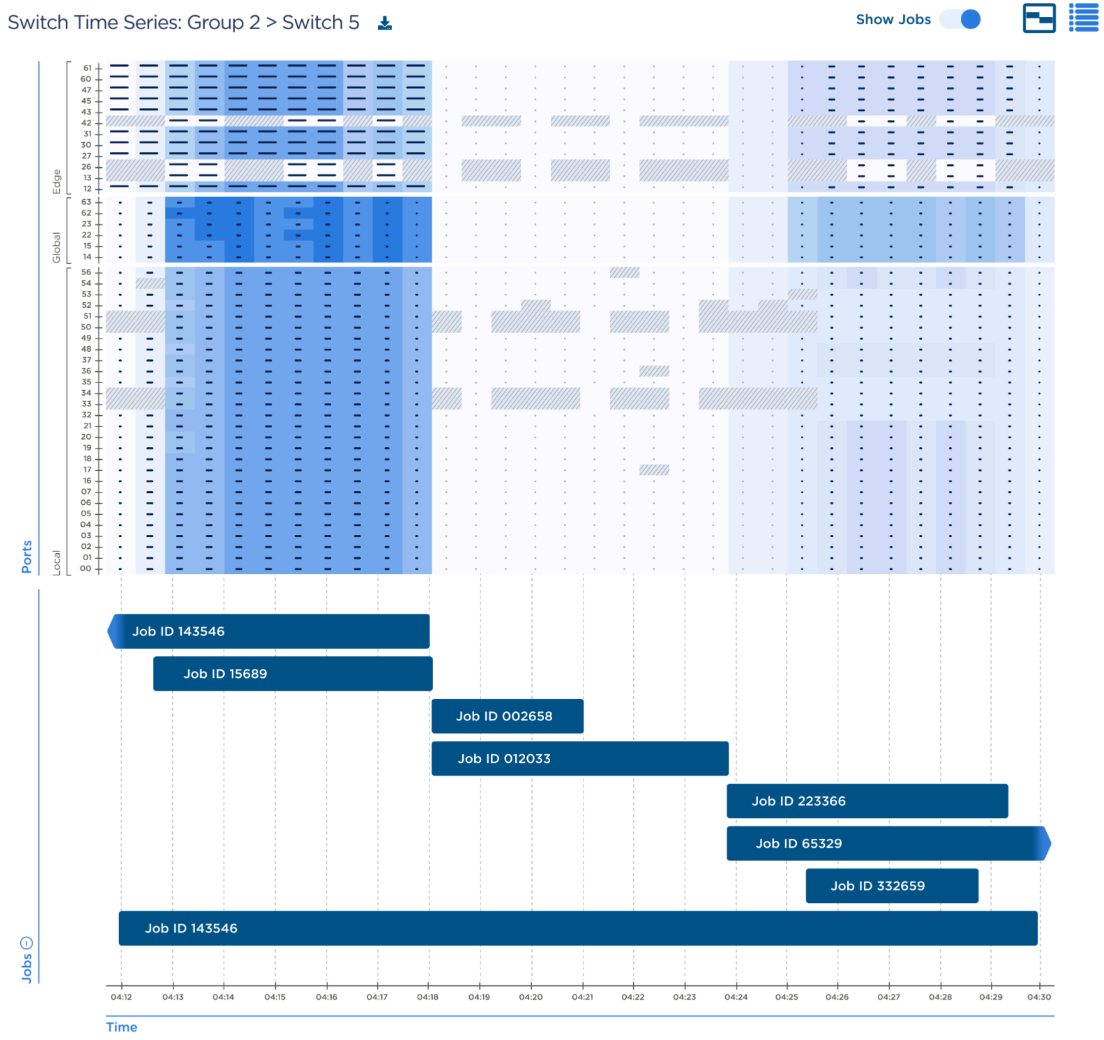
DIVING INTO SLINGSHOT BEHAVIOR WITH TRELLIS

EXAMPLE : MONITORING THE NETWORK FABRIC

Network Overview - Fabric Health and Link Failures



Overlay Jobs and Network Performance



Slingshot Network Visualization

Fabric View | Network Topology | System Overview | Job Overview

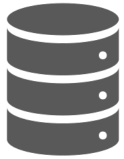
DATE & TIME FILTER

Start Date	Start Time		End Date	End Time	
01/23/2021	00:37:00	>	01/23/2021	01:15:59	Apply

No aggregate telemetry data...

TRELLIS : IMPLEMENTATION

- Fast, User-friendly Pythonic APIs
- Fuse multiple data sources
- Transformations on Telemetry



OmniSciDB Columnar Store

Topology | Telemetry | Jobs

Resample
Threshold
Aggregate

API QUICK TOUR

Topology Queries

Topology queries let you get connectivity information at the fabric, group and switch level, as an adjacency list in a pandas dataframe.

Get the topology for the entire fabric

```
df = trellis_topology_api.get_fabric_topology()
df[df["src_port_type"]=="global"].reset_index(drop=True).head()
```

	src_group_id	src_switch_id	src_switch_name	src_port_type	src_port_name	src_port_id	dst_group_id	dst_switch_id	dst_switch_name	dst_port_type	dst_port_name	dst_port_id	node_id
0	0	0	x3000c0r39b0	global	x3000c0r39j8p0	6	1.0	9.0	x1000c2r3b0	global	x1000c2r3j17p0	58.0	None
1	0	0	x3000c0r39b0	global	x3000c0r39j8p1	7	1.0	9.0	x1000c2r3b0	global	x1000c2r3j17p1	59.0	None
2	0	0	x3000c0r39b0	global	x3000c0r39j26p0	54	8.0	9.0	x1003c6r3b0	global	x1003c6r3j17p0	58.0	None
3	0	0	x3000c0r39b0	global	x3000c0r39j26p1	55	8.0	9.0	x1003c6r3b0	global	x1003c6r3j17p1	59.0	None
4	0	1	x3000c0r40b0	global	x3000c0r40j8p0	6	2.0	9.0	x1000c6r3b0	global	x1000c6r3j17p0	58.0	None

Get all compute nodes connected to switch "x1000c5r1b0"

```
df[(df['src_switch_name'] == "x1000c5r1b0") & (df['src_port_type'] == "edge")][['src_port_name', 'node_id']].reset_index(drop=True)
```

	src_port_name	node_id
0	x1000c5r1j103p0	nid001173-nmn
1	x1000c5r1j103p1	nid001172-nmn
2	x1000c5r1j101p1	nid001165-nmn
3	x1000c5r1j101p0	nid001164-nmn
4	x1000c5r1j105p0	nid001180-nmn
5	x1000c5r1j105p1	nid001181-nmn

API QUICK TOUR

Aggregate telemetry

Get an aggregated telemetry for the entire fabric. Telemetry can be aggregated by using one of `[max, min, mean, sum]`, for a given time period, and a metric (i.e. `rxBW`, `txBW`, `rxBlocked`).

```
%%time

# get the average rx Bandwidth across the fabric
# for a time-interval
df = trellis_telemetry_api.get_agg_telemetry(
    datetime_range = [
        pd.Timestamp("2021-01-23 00:38:00"),
        pd.Timestamp("2021-01-23 01:15:00")
    ] ,
    agg_type='mean',
    counter_name='rxBW'
)

df.head()
```

```
CPU times: user 47.8 ms, sys: 3.2 ms, total: 51 ms
Wall time: 2.61 s
```

	group_id	switch_id	port_type	port_id	port_name	counter_type	counter_name	counter_value	switch_name	node_id
0	1	0	edge	0	x1000c0r1j105p0	Congestion	rxBW	3.500000e+01	01:00	nid001020-nmn
1	1	1	edge	0	x1000c0r3j105p0	Congestion	rxBW	3.420610e+02	01:01	nid001020-nmn
2	1	2	edge	0	x1000c0r5j105p0	Congestion	rxBW	3.337969e+07	01:02	nid001022-nmn
3	1	3	edge	0	x1000c0r7j105p0	Congestion	rxBW	1.180098e+07	01:03	nid001022-nmn
4	1	4	edge	0	x1000c1r1j105p0	Congestion	rxBW	3.448334e+07	01:04	nid001052-nmn

CLOSING REMARKS

- Problem: *"What is happening in the network?"*
- Foundation: Monitoring Application/Network Performance with trellis
  - Identify communication patterns
  - User-friendly UI for Network Performance Overview
- Pythonic APIs
  - Powerful and Efficient
  - Enable custom workflows
- Paper: Additional examples with
  - Job-mixes
  - Communication Patterns

FUTURE WORK

- Scaling Considerations
  - Goal: Interactivity while maintaining accuracy
  - Today: 1024-node System, 4-hour telemetry
    - 0.5 billion data-points
  - Trade-offs: Resolution of
    - Data acquisition
    - Analysis
    - Visualization
- Modeling Performance
  - Modeling sensitivity of applications to network performance
  - Optimizing job-mixes
- Looking forward to some exciting collaborations !

FORWARD LOOKING STATEMENTS

This presentation may contain forward-looking statements that involve risks, uncertainties and assumptions. If the risks or uncertainties ever materialize or the assumptions prove incorrect, the results of Hewlett Packard Enterprise Company and its consolidated subsidiaries ("Hewlett Packard Enterprise") may differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to any statements regarding the expected benefits and costs of the transaction contemplated by this presentation; the expected timing of the completion of the transaction; the ability of HPE, its subsidiaries and Cray to complete the transaction considering the various conditions to the transaction, some of which are outside the parties' control, including those conditions related to regulatory approvals; projections of revenue, margins, expenses, net earnings, net earnings per share, cash flows, or other financial items; any statements concerning the expected development, performance, market share or competitive performance relating to products or services; any statements regarding current or future macroeconomic trends or events and the impact of those trends and events on Hewlett Packard Enterprise and its financial performance; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include the possibility that expected benefits of the transaction described in this presentation may not materialize as expected; that the transaction may not be timely completed, if at all; that, prior to the completion of the transaction, Cray's business may not perform as expected due to transaction-related uncertainty or other factors; that the parties are unable to successfully implement integration strategies; the need to address the many challenges facing Hewlett Packard Enterprise's businesses; the competitive pressures faced by Hewlett Packard Enterprise's businesses; risks associated with executing Hewlett Packard Enterprise's strategy; the impact of macroeconomic and geopolitical trends and events; the development and transition of new products and services and the enhancement of existing products and services to meet customer needs and respond to emerging technological trends; and other risks that are described in our Fiscal Year 2018 Annual Report on Form 10-K, and that are otherwise described or updated from time to time in Hewlett Packard Enterprise's other filings with the Securities and Exchange Commission, including but not limited to our subsequent Quarterly Reports on Form 10-Q. Hewlett Packard Enterprise assumes no obligation and does not intend to update these forward-looking statements.

THANK YOU

Alexandria Team
alexandria@hpe.com