

KNIT10 FABRIC Tutorial:

# Efficiently Accessing Public Big Data Sets

The University of Kentucky MF Team

Presenters: James Griffioen, Charles Carpenter

Other Contributors: Pinyi Shi, Mami Hayashida, Yongwook Song, Hussamuddin Nasir,  
Vaiden Logan, Dakota Morgan, Ken Calvert, Zongming Fei

March 11, 2025


# Preliminaries:

## Slides and Notebooks

- **Slides** for this tutorial can be found in the [KNIT10 Presentations folder](#) on the Workshop Materials page <https://knit.fabric-testbed.net>
- **Notebooks** can be found by using the [FABRIC Artifact Manger](#) to search for KNIT10 and then selecting “KNIT10 Accessing Public Big Data Sets”

See the next slide for instructions to get the files using the FABRIC JupyterHub.

# Accessing This Tutorial's Slides and Notebook

- 1) Login to FABRIC's Jupyter <https://jupyter.fabric-testbed.net> using the “default” server option (although any of the server options will work).
- 2) Open the directory **jupyter-examples-rel1.8.0** (any rel > 1.7.1 will work)
- 3) Open and run the **artifact\_manager.ipynb** notebook
- 4) Type *knit* into the **Title:** search field. 
- 5) Find **KNIT10 Accessing Public Big Data Sets** in the results and click the **Download** button.
- 6) The files will automatically be downloaded and extracted to **/home/fabric/work/KNIT10\_Accessing\_Public\_Big\_Data\_Sets/vYYYY-MM-DD/tutorial\_files** where *YYYY-MM-DD* is the latest version.

# FABRIC In-Network Data Caching Service (FAB-Cache)

# Background

- Researchers from all research domains are now leveraging big data sets for AI, data analysis, simulation/modelling, etc. These big datasets are often (redundantly) requested by many processing nodes on the network. For example the same data may be used by researchers all over the world, or by many nodes in a cluster processing different parts of the data.
- Frequent redundant downloads of large datasets (GBs to TBs) from remote repositories strain network bandwidth, increase costs, slow down access times, and overload source servers, leading to availability and performance issues.

# Need for a Distributed In-Network Caching System

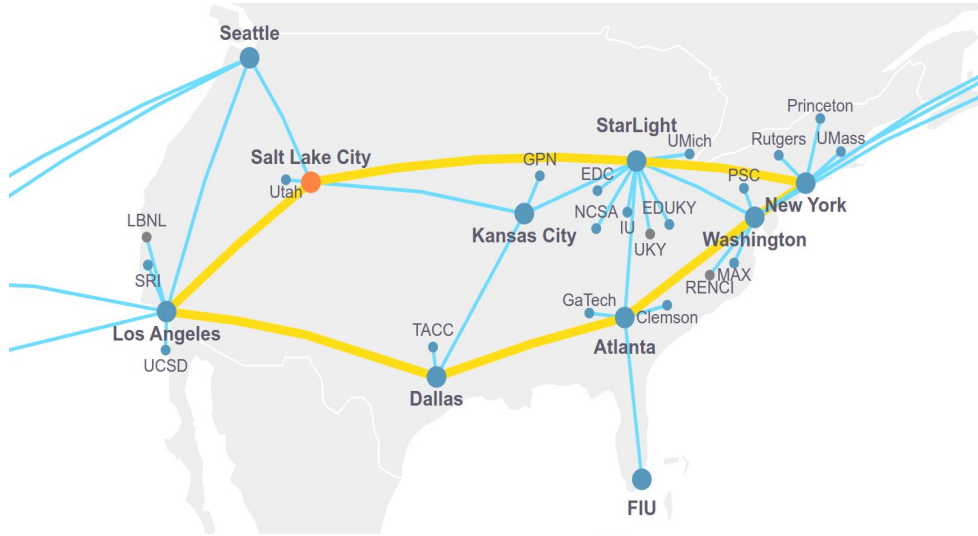
## In-Network Caching:

- Cache data at network routers as data flows from a server to a client
- Network provides data to clients if it has it cached at a router on the path to the server.

## Benefits:

- **Reduce Bandwidth Usage:** Avoid redundant downloads.
- **Faster Access:** Retrieve datasets from nearby cached copy.
- **Less Load on Data Providers:** Improve availability of public datasets.
- **Scalability:** Support more users without network congestion.

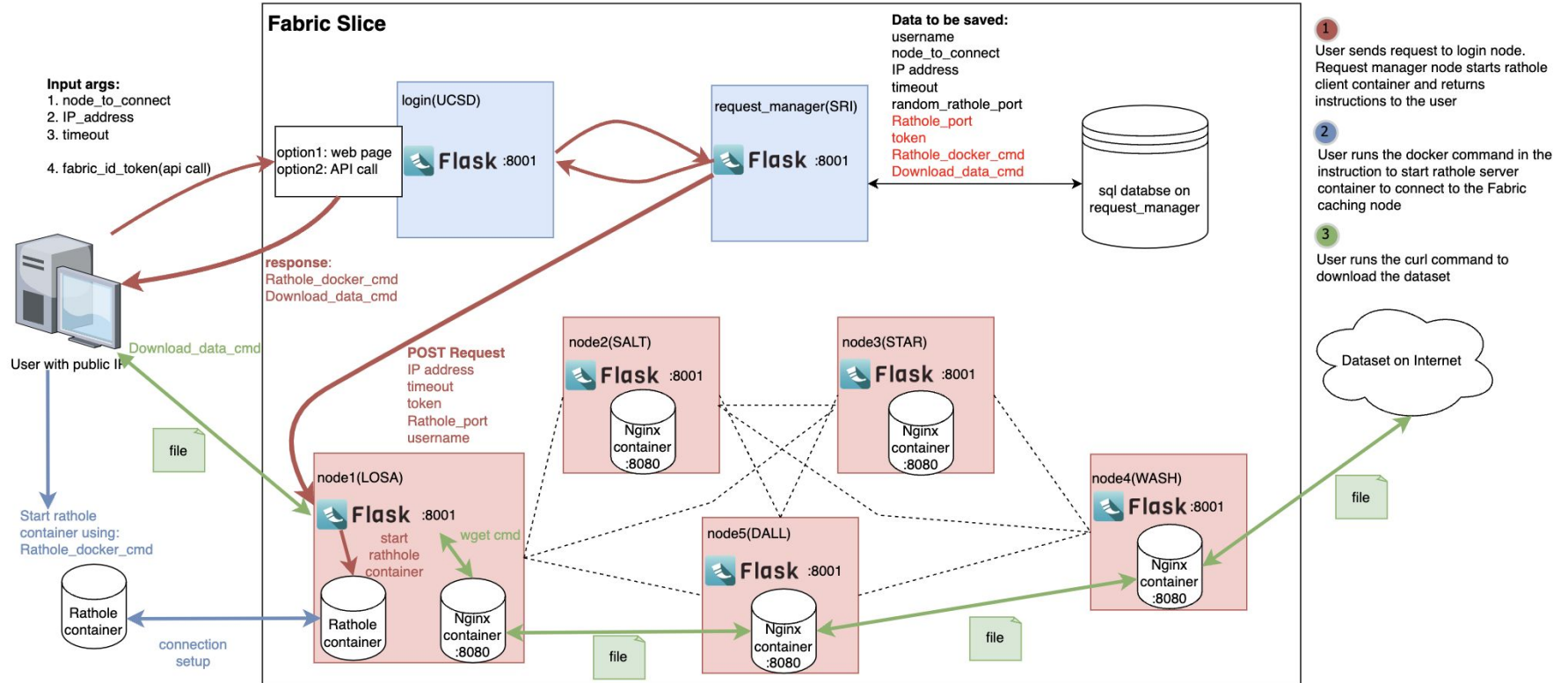
# Implementing an In-Network Caching System in Fabric



Fabric sites are geographically distributed and each site has large storage volumes that can be used to cache big datasets.

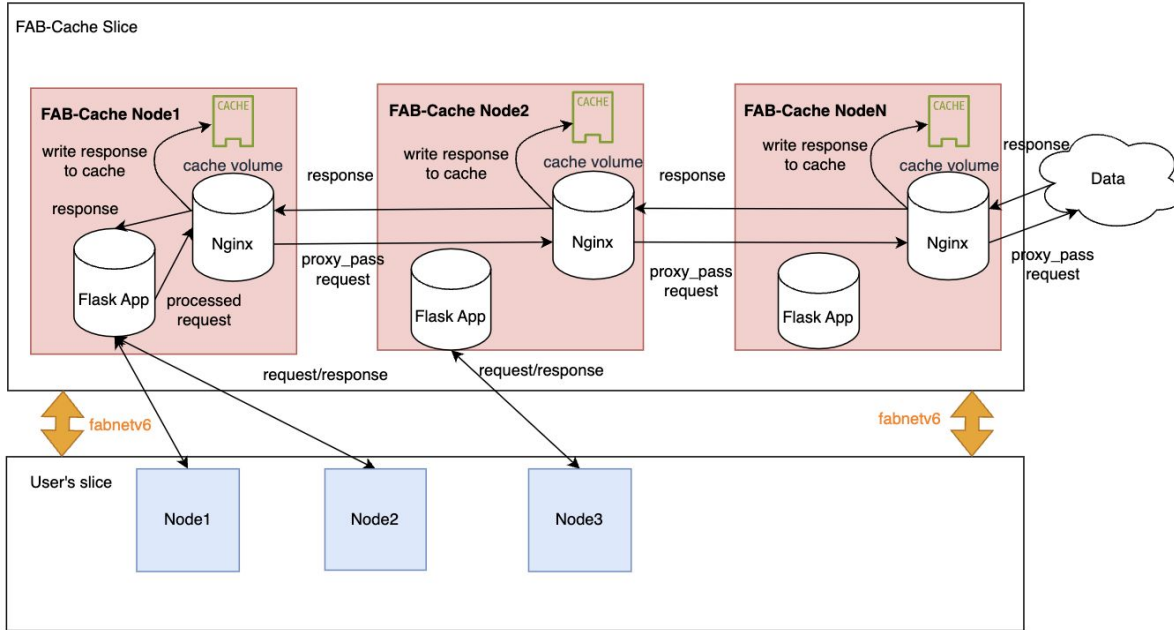
FABRIC's high performance network links make data transfer faster.

# Fabric In-Network Data Caching Service(FAB-Cache) Design





# FAB-Cache Components(Used in this tutorial)



## Nginx proxy chain:

Proxy\_pass the request to the next nginx container.

Response will be cached to the volume.

## Flask app:

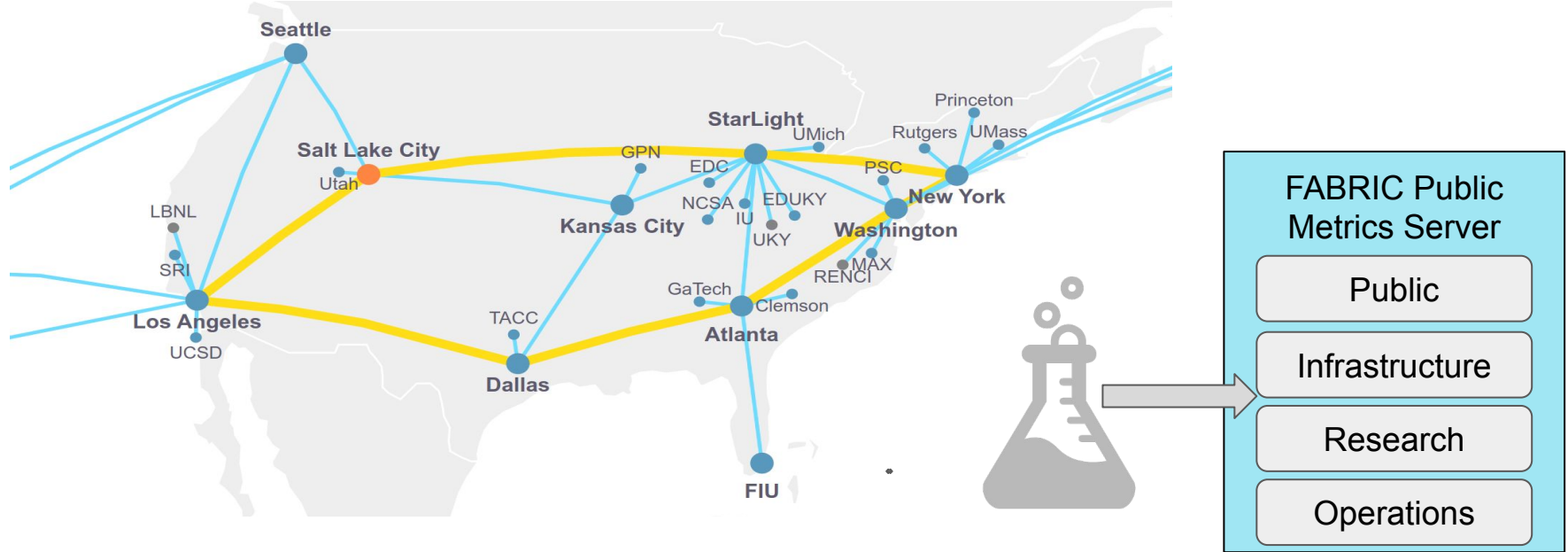
User-facing application for submitting requests and check status of FAB-Caching nodes

We will create a fabric slice with fabnetv6 and use the fabnetv6 addresses of the caching nodes for connection. (No Rathole tunnel)

See the FAB-cache Demo/Tutorial

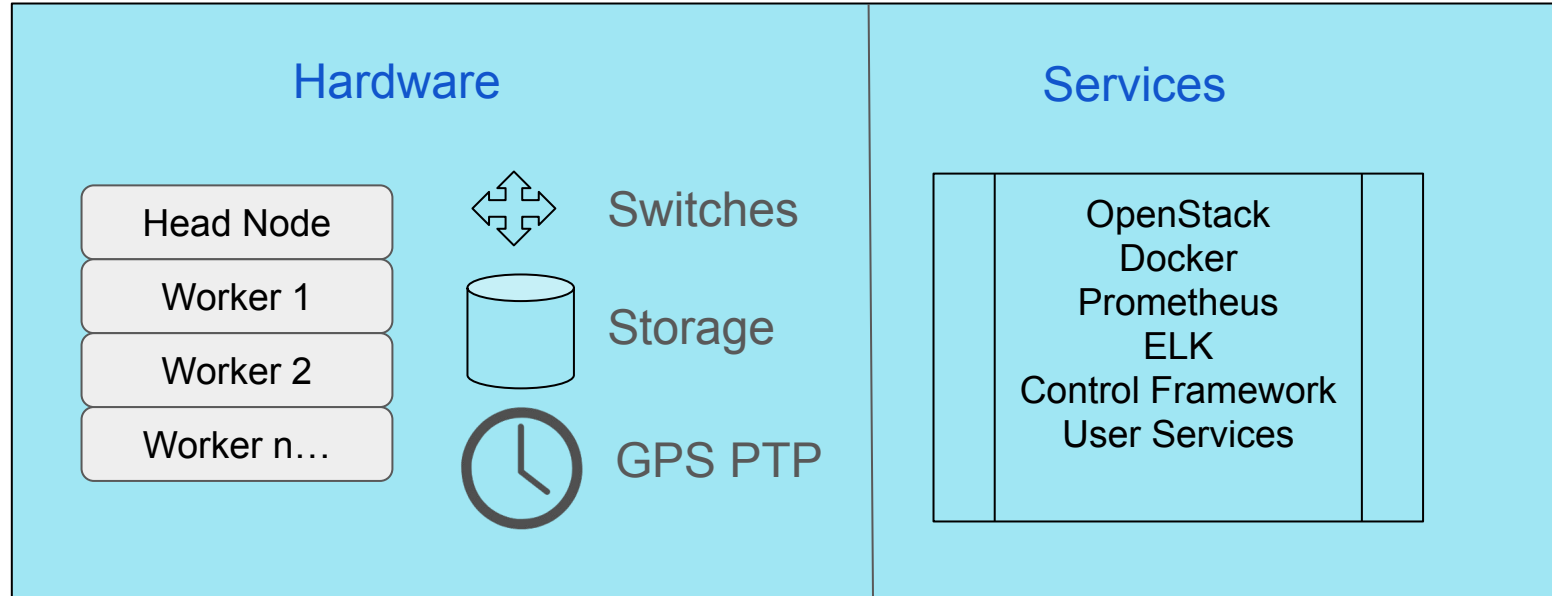
# Monitoring FABRIC Usage

# FABRIC Infrastructure



The FABRIC Measurement Framework collects detailed measurements (metrics) from each FABRIC rack. Unlike other research infrastructure, FABRIC makes that data available to the research community based on differing levels of trust: *Public*, *Infrastructure*, *Research*, and *Operations*

# FABRIC Rack Infrastructure (Components to Measure)



# FABRIC Rack Measurement Data

## Example: Prometheus Exporter Data

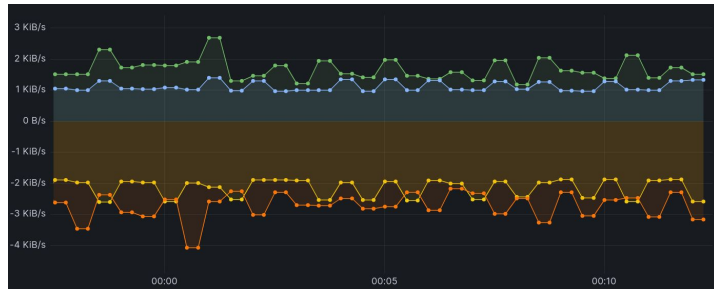
```
# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="post",code="200"} 1027 1395066363000
http_requests_total{method="post",code="400"} 3 1395066363000
```

Measurement Data includes the **Metric Name**, a **Numeric** value(s) stored with a **Timestamp** and a set of **Labels**



Prometheus is used to periodically gather metrics and store them in a Time Series DataBase (TSDB)

Grafana is used to visualize the metrics.



# Making FABRIC Metrics Available

## Public Metrics:

- Very basic link information (e.g., up/down, utilization)
- **Available to anyone** – no authentication required

## Infrastructure Metrics:

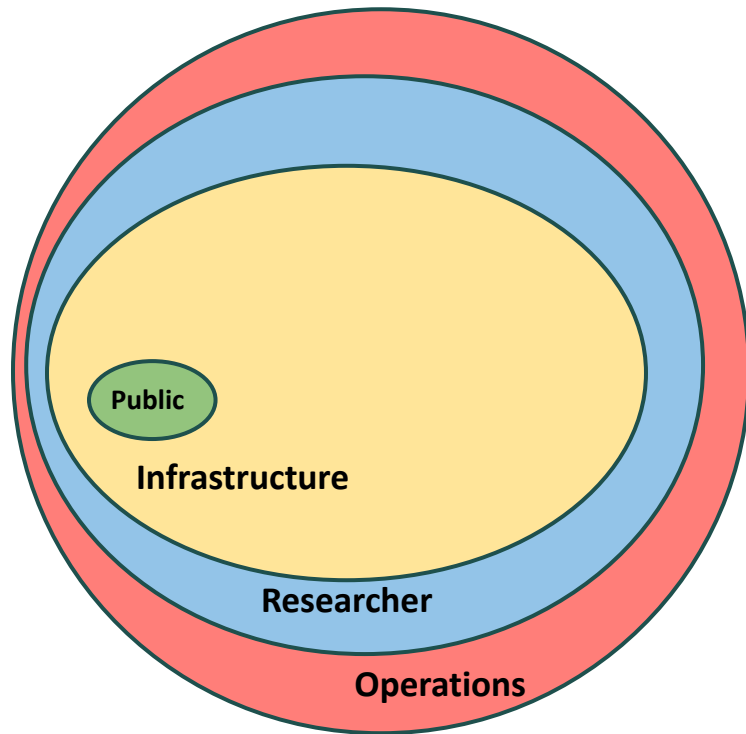
- Non-private infrastructure information (e.g., cpu, mem, disk, net loads)
- **Available to FABRIC users** – authenticate with FABRIC login

## Research Metrics:

- Non-regulated infrastructure information (e.g., most everything except IRB, PII, etc)
- **Requires a proposal** describing research use as well as NDA – only approved researchers can access the data

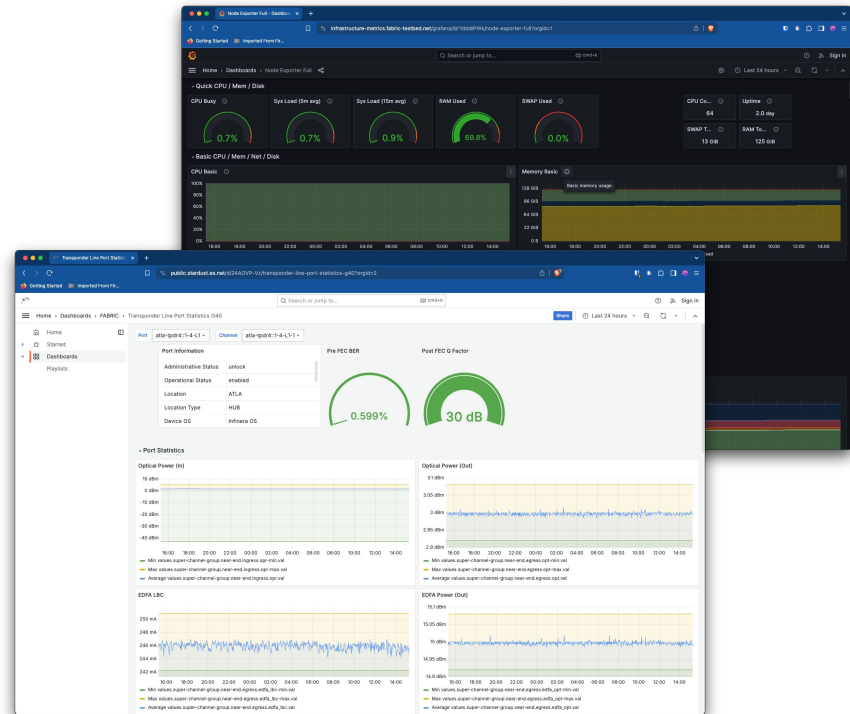
## Operations Metrics:

- All infrastructure information used by operations team
- **Available to FABRIC Operations** team – only operators can access the data



# FABRIC Measurement Data

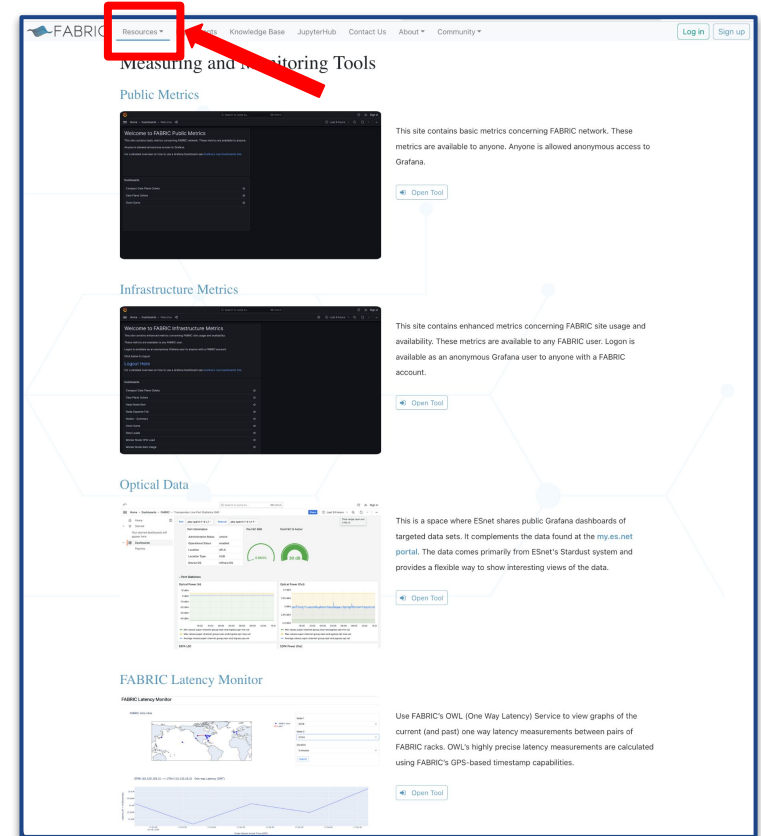
- System level Data
  - Bandwidth, packet counts, one-way packet latency, CPU/memory/disk usage, etc.
- Low-level Optical Data
  - ESnet portal
  - Optical Power, EDFA Power, Pre-FEC Data, Optical Frequency, Differential Group Delay, etc.





# Accessing FABRIC Infrastructure Performance Data

- FABRIC Measurement Services:  
<https://portal.fabric-testbed.net/resources/tools>
- Public Network Data:  
<https://public-metrics.fabric-testbed.net/grafana>
- Infrastructure Measurement Data:  
<https://infrastructure-metrics.fabric-testbed.net/grafana>
- Optical Performance Data: Follow the link on  
<https://portal.fabric-testbed.net/resources/tools>
- Latency Data:  
<https://public-metrics.fabric-testbed.net/latency>
- PerfSonar Measurements: Link coming soon



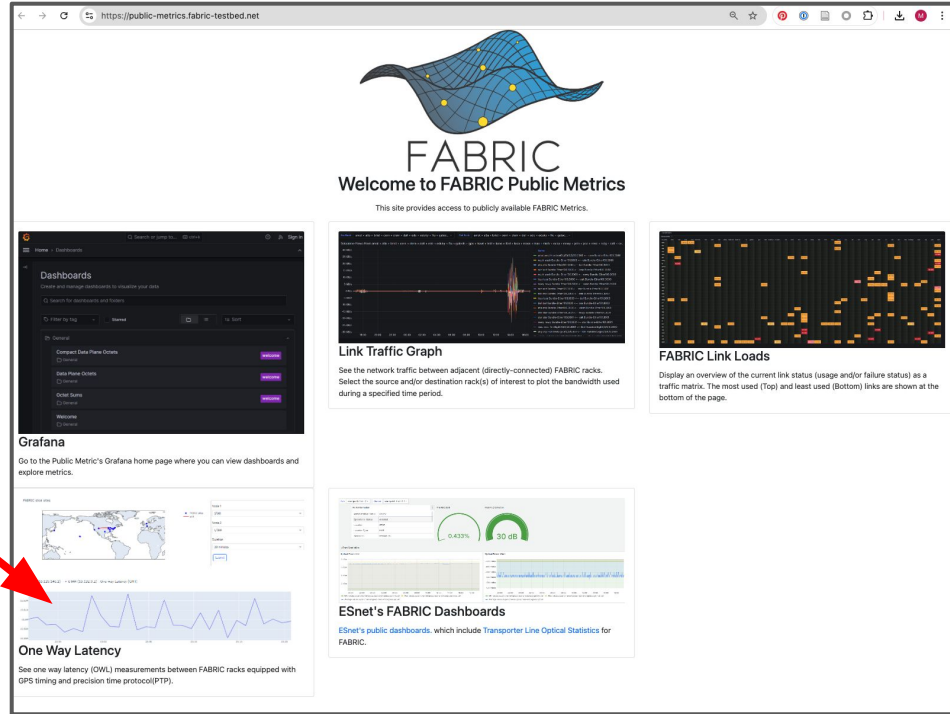
# Demonstration:

Working with FABRIC's  
Public and Infrastructure Performance Data

OWL

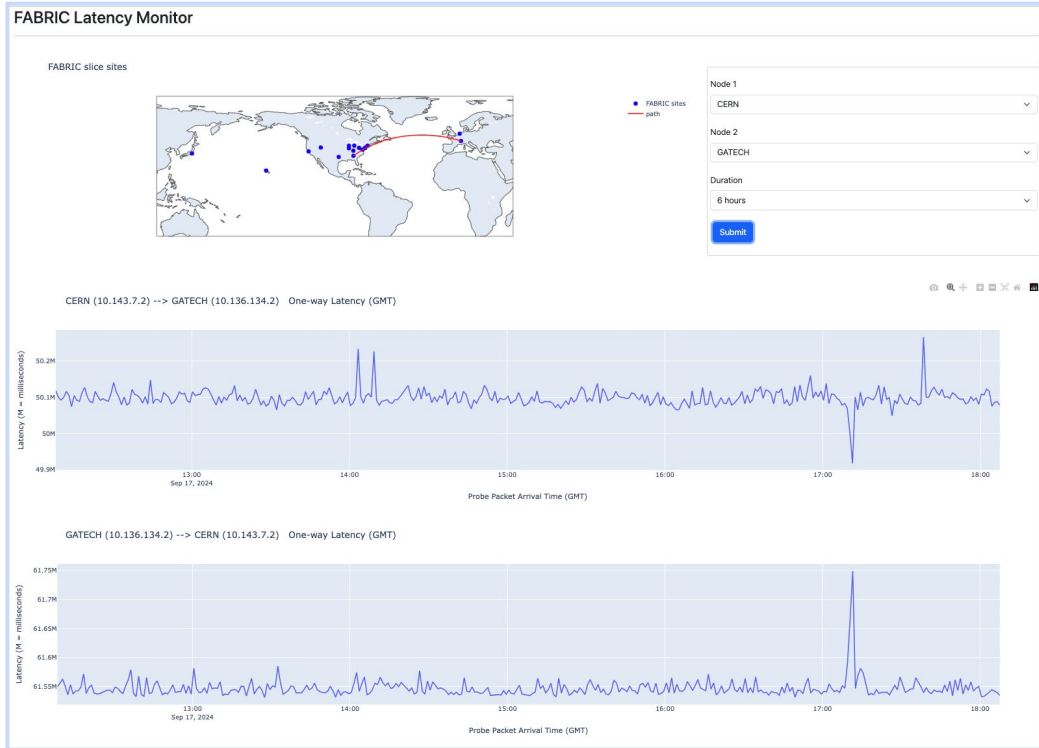
FABRIC's One Way Latency Service

# FABRIC's One Way Latency (OWL) Service



<https://public-metrics.fabric-testbed.net> site and clicking on the Latency link.

# One Way Latency (OWL) Measurements



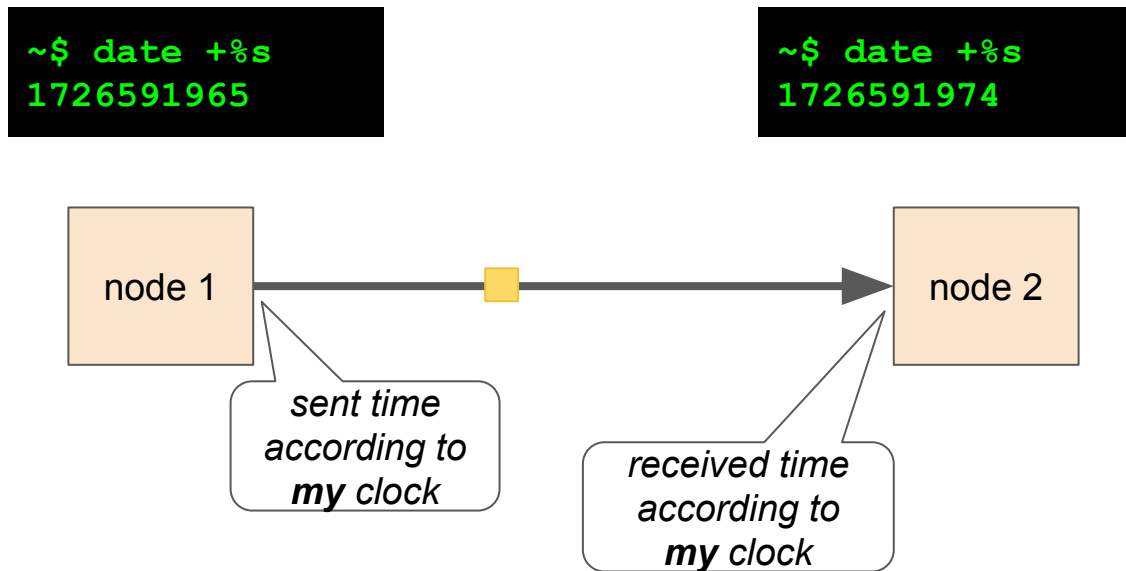
<https://public-metrics.fabric-testbed.net/latency/>

# Why OWL (One Way Latency)?

## Motivation

One Way Latencies are often difficult to measure because the source and destination clocks must be synchronized.

**UNSYNCHRONIZED!**

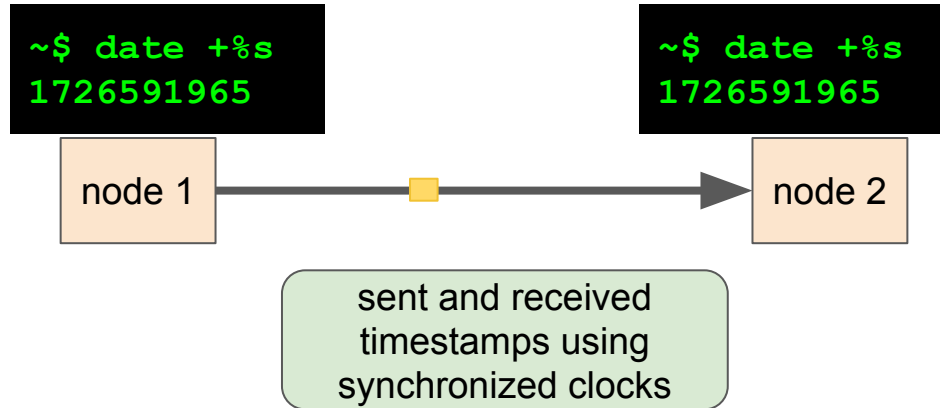
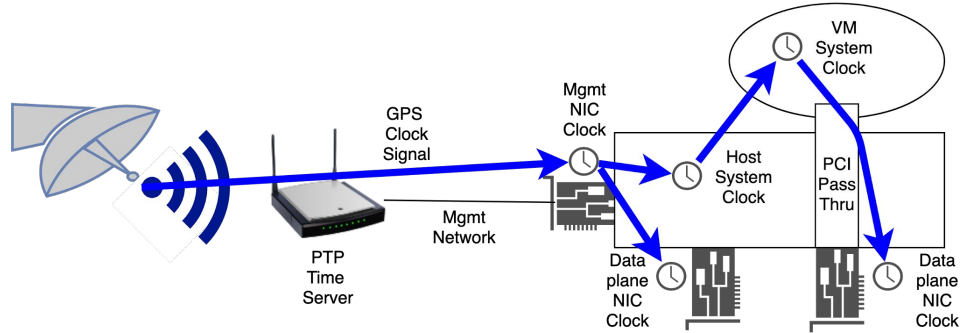


# One Way Latency (OWL) and PTP clocks

FABRIC racks use GPS disciplined clocks that are highly synchronized.

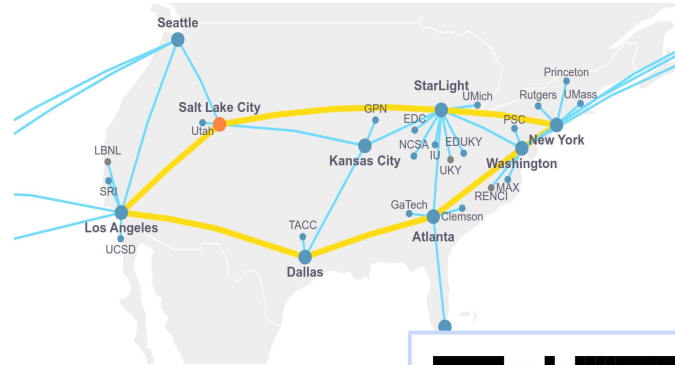


*Synchronized clocks*



# Collecting One Way Latency (OWL) Information on FABRIC

- Uses a long-lived One-Way Latency (OWL) Measurement Slice with a VM on each (GPS-driven) PTP capable site
- OWL on each VM actively sends time-stamped probe packets to other VMs
- OWL on each receiver VM calculates the one way latency using high accuracy GPS-synchronized clocks
- OWL latency information is public.



<https://public-metrics.fabric-testbed.net/latency/>



# Questions? Comments?

This work supported in part by NSF Grant numbers 1935966, 2330891, and 2029235