# Constructing Workflow-Centric Traces in close to Real Time for the Hadoop File System

Neel Bhalla, Lexington High School

Adviser: Prof. Raja Sambasivan, Tufts University

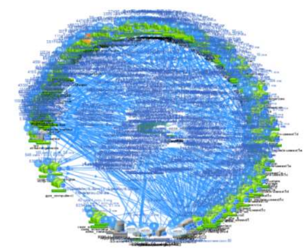
June 8, 2020

# Motivation

- Most modern services use a distributed architecture
- We can't debug distributed services – complicated, with 1000's of nodes and services
- To find performance issues in distributed systems we need to do tracing

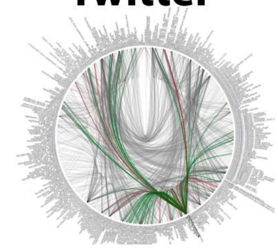
**Debugging services in a distributed system is a complicated problem**

**Netflix**

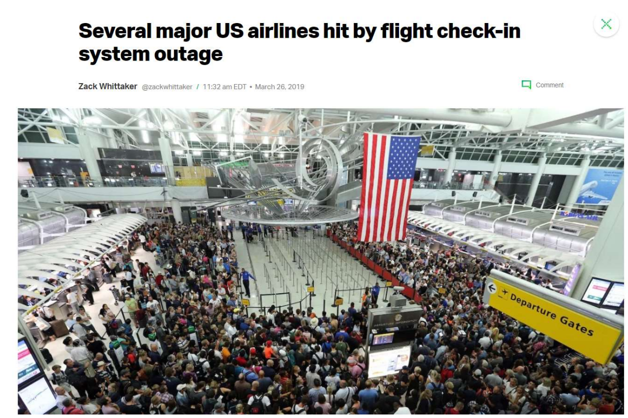


Why is my movie lagging?

**Twitter**



My feed not loading!


Several major US airlines hit by flight check-in system outage

Zack Whittaker @zackwhittaker / 11:32 am EDT • March 26, 2019

Comment

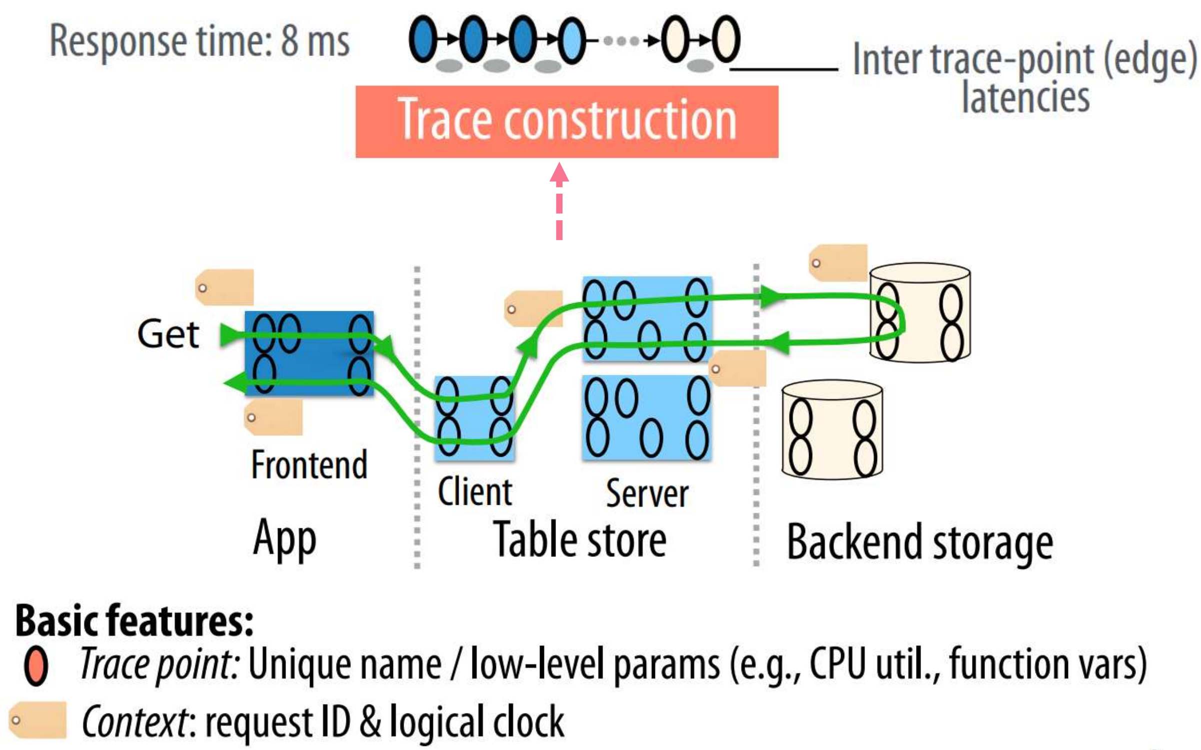


March 26, 2019

# Workflow-Centric Tracing

- Every request involves a workflow
- Machine-centric techniques are insufficient
  - E.g., GDB, GProf, perf. counters, strace, Dtrace
- Workflow: Structure & timing of work done to process them
- Structure: Order, concurrency & synchronization

**Challenge with tracing: real-time construction**



# Problem: Trace Reconstruction is Slow

- Tracking performance requires tracing services through a distributed system
  - Data and computation intense
  - Facebook collects 1 Billion Traces/day

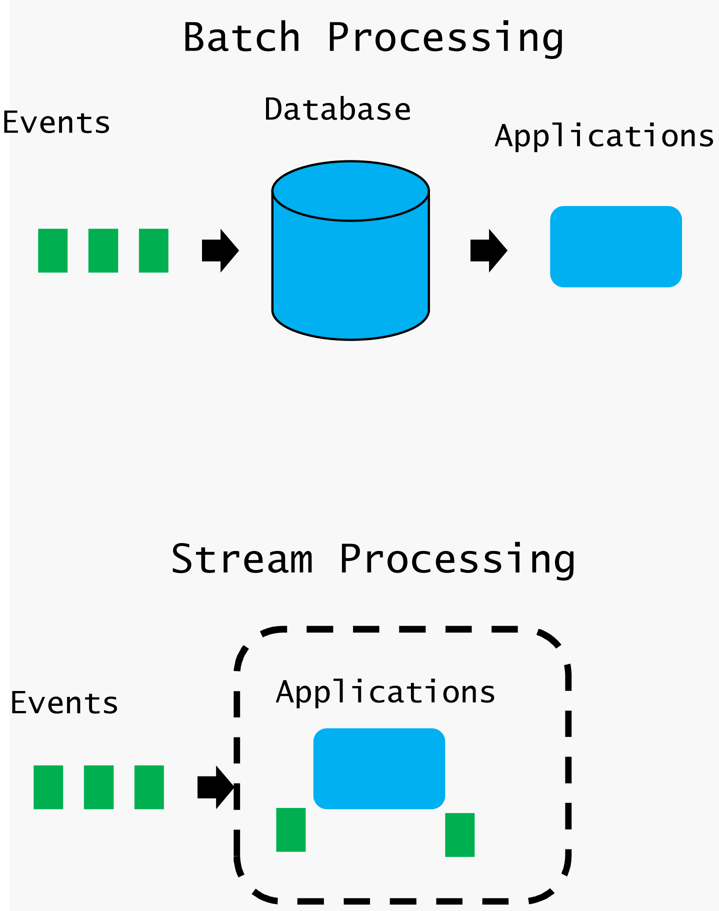
- Existing tracing systems operate in mostly non-real time batch mode (e.g. Facebook ~ 1 day turnaround to find problems)

- The "look-back" approach for finding anomalies or performances issues requires archiving and analyzing massive amount of unorganized log data

**Finding problems in distributed systems is like finding a needle in the haystack**

# Stream Processing Framework

- Tracing data is created as a series of events over time.
- In batch processing data is stored in a database. Applications would query the data or compute over the data as needed
- Stream Processing turns this paradigm around: The application logic, analytics, and queries exist continuously, and data flows through them continuously.
- Upon receiving an event from the stream, a stream processing application reacts to that event: it may trigger an action, update an aggregate or other statistic, or “remember” that event for future reference.



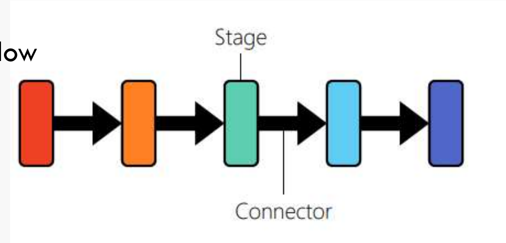
# Timely Dataflow

- Framework for writing dataflow programs

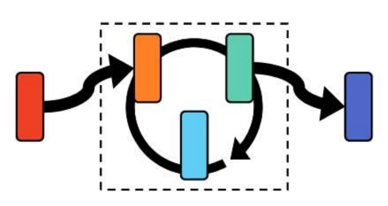
- Dataflow programming is a programming model in which the computation can be represented as a directed graph: The data flows along edges, while the computational logic in the vertices transforms it

- The messages flowing along edges are annotated with timestamps.

Dataflow




Dataflow Iteration



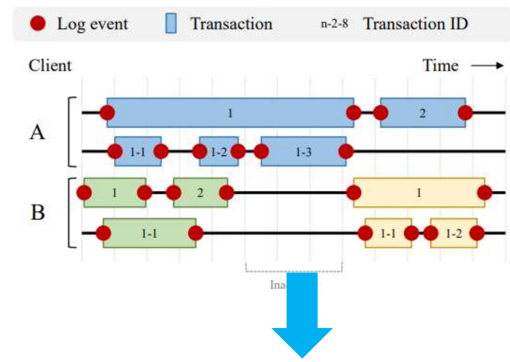
**Timely Dataflow used as the streaming framework**

Reference Naiad: A Timely Dataflow System

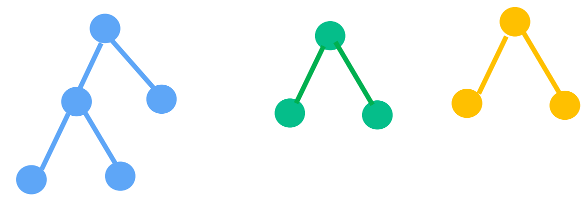
# Previous Work

- Several Non-Real Time Approaches
  - Google's Dapper, Facebook's Canopy, Brown's X-Trace
- ETH's Strymon [1] is a real-time stream processing system
  - It builds on Naiad Streaming Timely Dataflow
  - Traces are modelled as trees
- General infrastructure tracing frameworks (OpenTracing, X-Trace) represent traces as DAG (Directed Acyclic Graphs)
  - DAG can capture concurrency
- Strymon's approach can be modified to incorporate DAG's

**Trees**




**DAG**

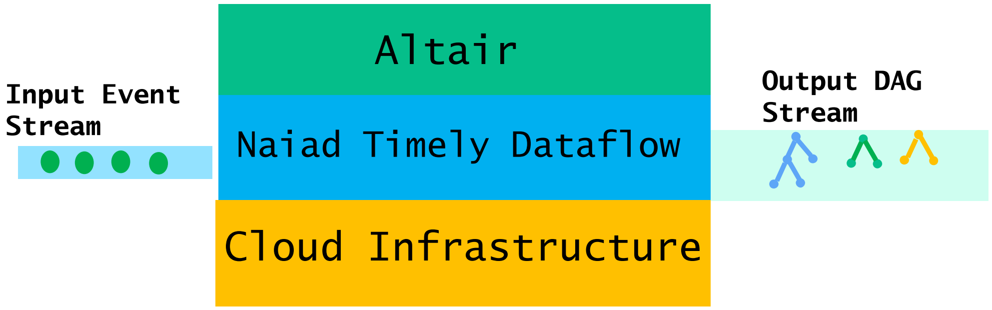


**We developed a new system for processing traces called Altair**


[1] Chothia, et. All, Online Reconstruction of Structural Information from Datacenter Logs, EuroSys '17

# Proposed System For Real Time Trace Reconstruction – **Altair**

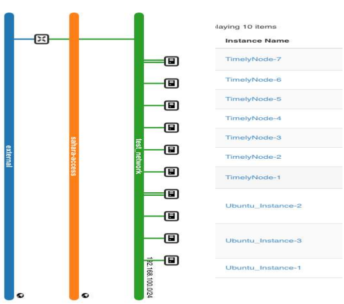
- Altair used Timely dataflow as the stream processing framework
- Input tracing event stream are reconstructed to create workflow model which are represented as DAG
- Interesting DAG’s can be stored or queried for analysis



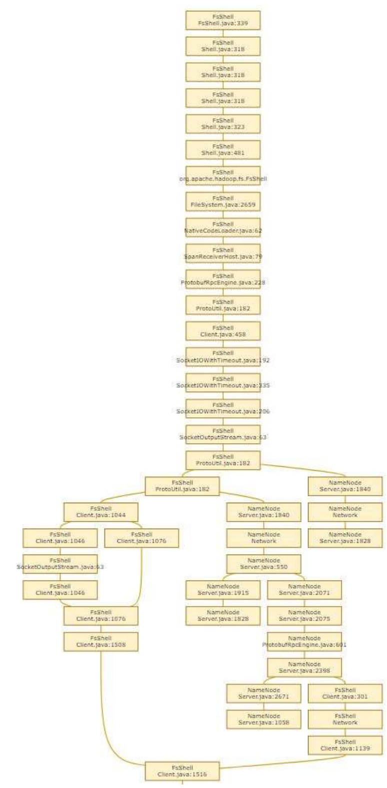
# Evaluation: HDFS Tracing

- Mass Open Cloud runs OpenStack
- Access to 10 compute instances, Altair runs on 8 instances, 2 instances run Trace compression
- Instrumented HDFS and X-Trace server
  - HDFS (Hadoop Distributed File System) is distributed file system, used with MapReduce applications in datacenters
  - Performance of HDFS can directly affect the performance of jobs
- Event Test Data: 3000 Traces, ~350 graph nodes/traces, 0.525 Million event/Epoch
- Streaming simulator to generate event stream, replay and add anomalies, latency in event stream

**Sample File Access HDFS DAG (part of a trace)**



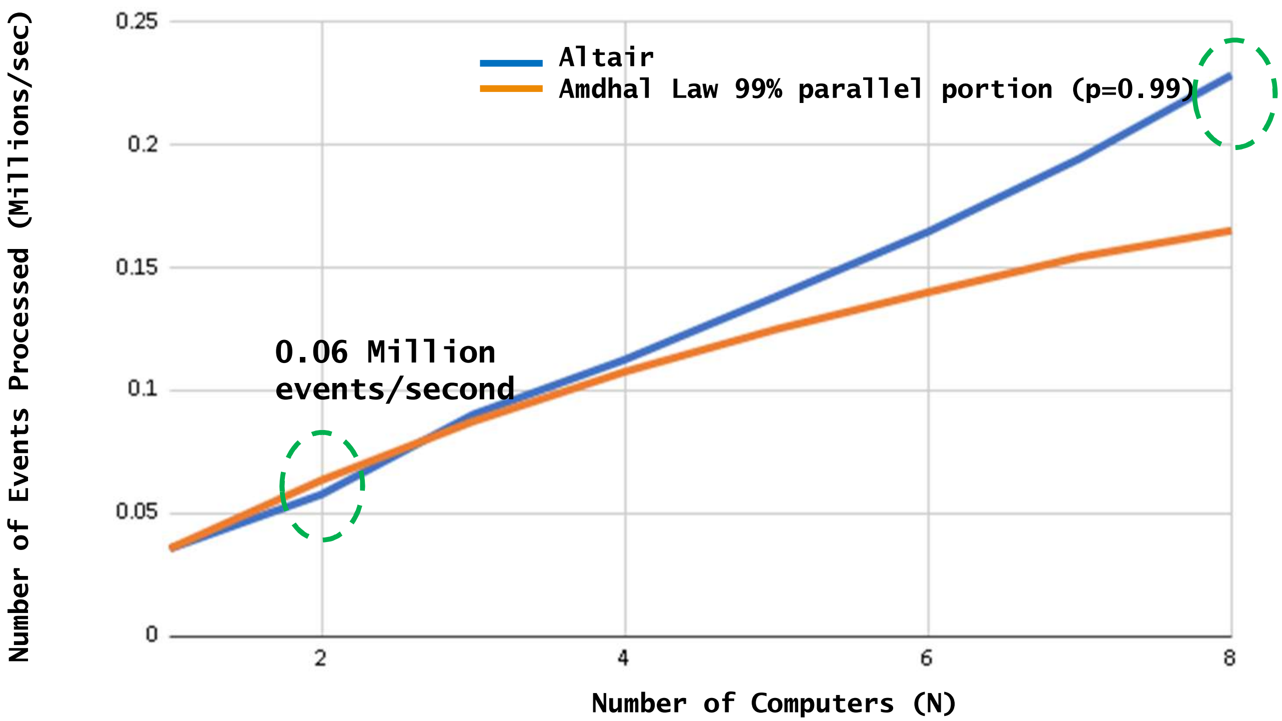

**Altair implemented in Rust**



Acknowledgement : Mass Open Cloud for their support

# Results : Altair

**Altair Events Processing Throughput**




**0.23 Million events/second**

Facebook collects 11 Million events/second (1 Billion Traces/day ~ (1000 nodes/trace))

**Amdhal Law:** Maximum expected improvement to a system when only part of the system in the parallelized

$$\text{Speedup} = \frac{1}{(1-P)+(\frac{P}{N})}$$

**The Altair approach is scalable with more than 99% parallelization**

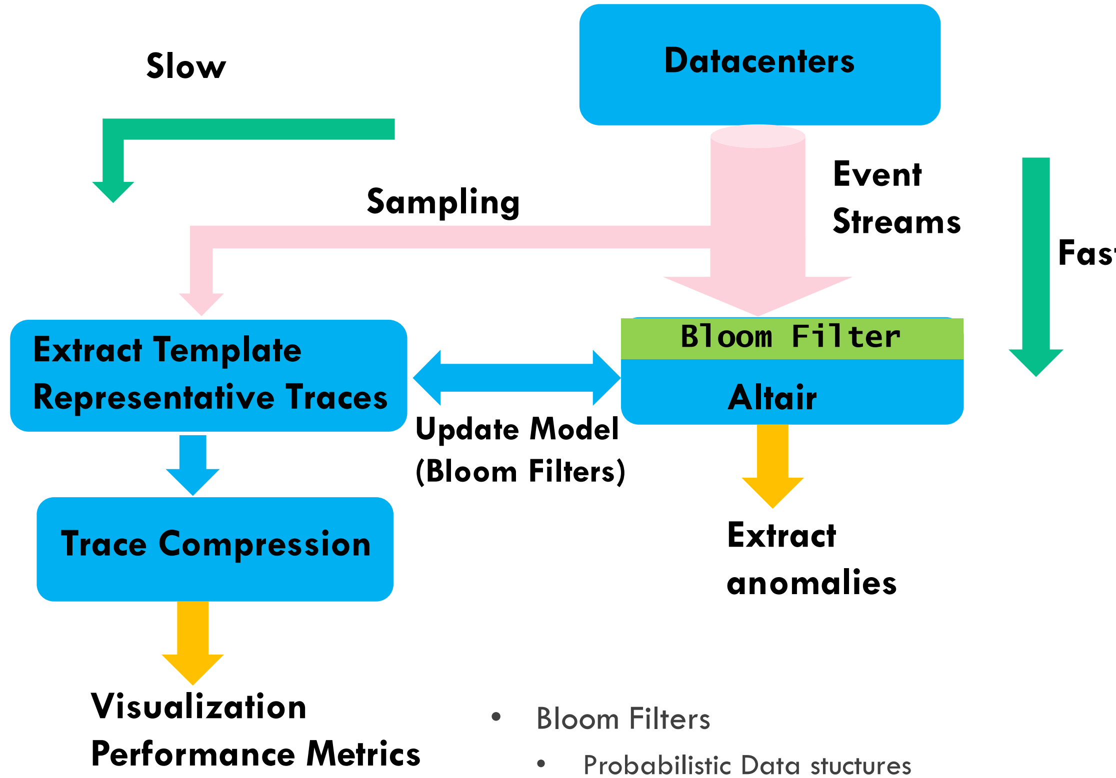
# Altair Use Cases

- Anomaly Detection in distributed systems
- Cyber Intrusion Detection
- Failover Management
- Performance Issues

# Altair Use Case: Anomaly Detection

- **Anomaly Detection application will be run continuously in two steps**
  - The first step involves designing the Bloom filter. The design of the Bloom filter requires a representative set of graphs. This step is not real-time.
  - We are proposing graph clustering approach to extract template representative graphs that would be programmed into Bloom Filter
  - Second Step with Altair will run in real time
  - Any anomaly traces will be flagged by the Bloom Filter with little overhead

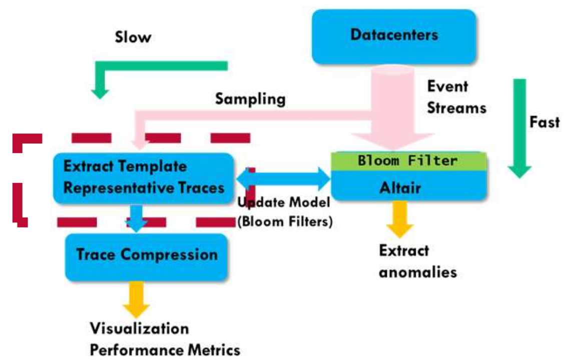
N epochs




- Bloom Filters
  - Probabilistic Data stuctures
  - "No" answers are always correct
  - Vector bit array (m) and hash functions (k)

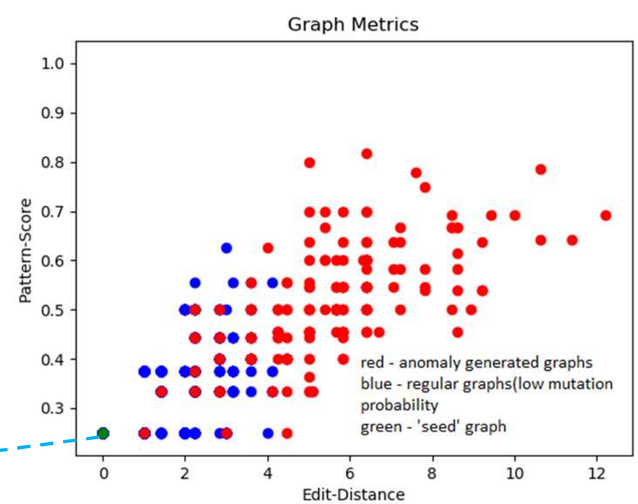
# Clustering to Extract Representative Workflows

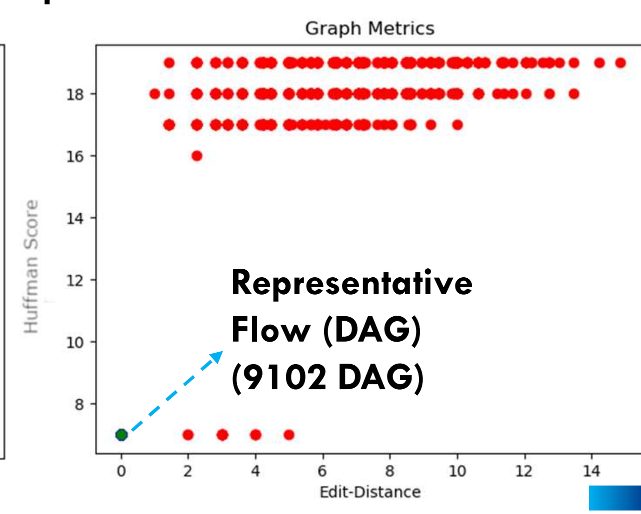
- Use clustering to find unique representative flows in a sample of the event stream
- Trace Compression using feature vectors
  - MDL Score
  - String Edit Distance (Levenshtein Distance)
  - Hoffman Coding
- Feature Vectors provide insight into the flows
  - Used to extract performance metrics
  - Visualization
- Computationally Expensive
- Representative traces used to design Bloom Filters




**Example**

**Representative Flow (DAG) (9072 DAG)**



# Integration with Pythia

- Pythia is a tracing system
- Places trace points for calls and programs and stores trace points in logs.
- Utilizes constructed traces' structure to figure out root cause analysis and adjusts granularity of traces based on it.
- Traces can be constructed faster at a smaller granularity if they don't pertain to the task.
- Altair functions as the primary workflow collection.  It provides data in the form of DAG's from which the workflow skeletons can be generalized
- Integration is ongoing

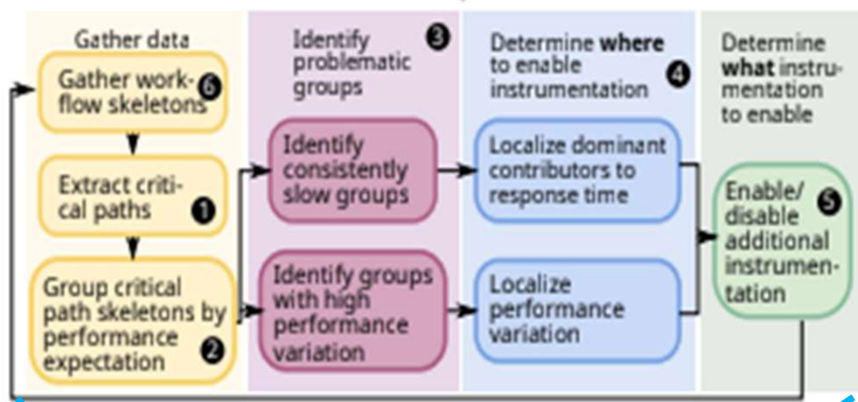
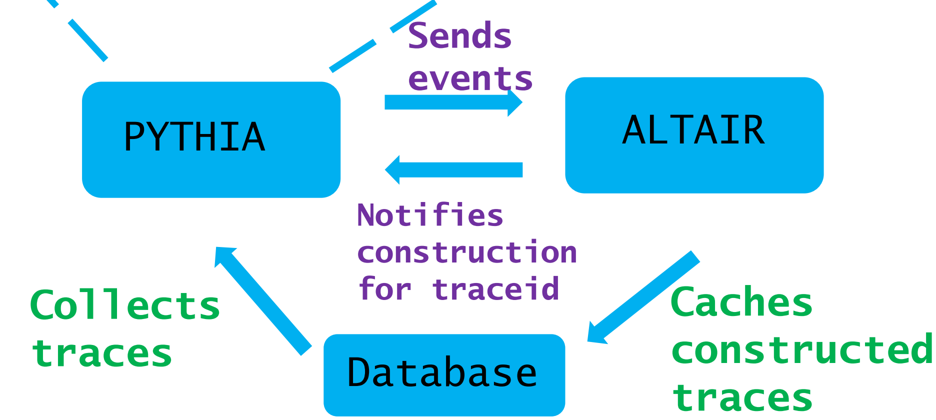



Figure 2: Pythia's continuous cycle ofoperation.

[Ref]




[Ref] Ates, et.al "An automated, cross-layer instrumentation framework for diagnosing performance problems in distributed applications"

# Conclusions

- Developed an distributed tracing framework system based on timely dataflow model called Altair that can achieve real time performance
- Evaluated the Altair System on for Anomaly Detection use case
- Evaluation shows that Altair is highly scalable and can be adapted for high production environments
- Integration with Pythia

## Acknowledgement

Thank you

- I want to thank
  - MIT, Primes for giving me the opportunity to participate in the research program
  - Prof. Raja Sambasivan, Tufts University for his invaluable guidance and spending time with me to assist me with research
  - Graduate students Emre Ates (BU) and Mania Abdi (Northeastern) for their feedback, knowledge and interactions
  - Mass Open Cloud for their access to cloud resources in support of this research


# Questions

# References


1. Canopy: An End-to-End Performance Tracing And Analysis System, Kaldor et al, SOSP '17, October 28, 2017, Shanghai, China
2. Principled workflow-centric tracing of distributed systems, Raja R. Sambasivan, et. al, In Proceedings of SoCC 2016
3. Visualizing request-flow comparison to aid performance diagnosis in distributed systems. Raja R. Sambasivan, et. al, IEEE Transactions on Visualization and Computer Graphics (Proc. Information Visualization 2013), Vol. 19, no. 12, Dec. 2013
4. Naiad: A Timely Dataflow System, Derek G. Murray, et. al., SOSP'13, Nov. 3–6, 2013, Farmington, Pennsylvania
5. https://github.com/TimelyDataflow/timely-dataflow
6. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure, Benjamin H. Sigelman, et. Al, Google Technical Report, 2010 https://research.google.com/archive/papers/dapper-2010-1.pdf
7. Diagnosing performance changes by comparing request flows, Raja R. Sambasivan, et. Al., Proceedings of NSDI 2011
8. GraphZIP: a clique-based sparse graph compression method, Ryan A. Rossi, et al., Journal of Big Data, December 2018
9. Mining of Massive Datasets, Jure Leskovec, Book, 2010
10. Emre Ates, et. al, "An automated, cross-layer instrumentation framework for diagnosing performance problems in distributed applications", SoCC '19: Proceedings of the ACM Symposium on Cloud Computing, November 2019.