TraViz: Visualization of Distributed Traces

- Matheus Stolet
- Vaastav Anand
What are Distributed Systems?

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

- Leslie Lamport
Distributed Systems are everywhere

- Distributed systems are widely deployed [1]
  - Graph processing
  - Stream processing
  - Distributed databases
  - Failure detectors
  - Cluster schedulers
  - Version control
  - ML frameworks
  - Blockchains
  - KV stores
  - ...

Need for Observability: Ability to answer questions

- **Which nodes/services** did the request go through?
- **Where were the bottlenecks** for the request?
- **What happened at every node/service** to process the request?
- **Where did the errors happen?**

- **How different** was the execution of 1 request?
- **How do different groups of requests** differ?
- **Axes for differences**
  - Structural
  - Performance
- **Root cause** analysis
Need for Observability: Ability to answer questions

- Which nodes/services did the request go through?
- Where were the bottlenecks for the request?
- What happened at every node/service to process the request?
- Where did the errors happen?

- How different was the execution of 1 request?
- How do different groups of requests differ?
- Axes for differences
  - Structural
  - Performance
- Root cause analysis

Distributed tracing can answer these questions
What is Distributed Tracing?

- Each trace represents path of 1 request through the system
- Trace collects and contains timing info, events across nodes, processes, and threads.
- Depending on verbosity, may also contain stack traces.

“Story of a request through a system”
Datasets

- 2 Trace Datasets & respective source code
  - Hadoop: [https://gitlab.mpi-sws.org/cld/systems/hadoop](https://gitlab.mpi-sws.org/cld/systems/hadoop)
- DSB: 22390 traces
- Hadoop: 72030 traces
Data Abstraction
Tasks

- Outlier Finding + Overview of Dataset
- Source Code Integration
- Timing analysis of a single trace
- Service dependency analysis
- Comparison of 2 traces
- Aggregation of multiple traces
Outlier finding + overview

- **What:** data
  - Traces

- **Why:** tasks
  - Find outliers and patterns

- **How:** reduce
  - Filter items using # events, duration, and day attributes

- **How:** show
  - Sortable and filtered table with traces
Source code relationship

- **What: data**
  - Traces

- **What: derived attrs**
  - # Events triggered by each line of code

- **Why:**
  - What files are producing events
  - What lines in a file produced the most or least events

- **How: aggregate**
  - Aggregate # events from all src code lines in a file

- **How: encode**
  - Encode number of events or number of lines with size of bar
  - Encode number of events or number of lines with colour of bar
Show an individual trace

- **What**: data
  - 1 trace

- **Why**:
  - Look at time of events in a trace related to each other
  - Find parent and child relationships between events

- **How**: encode
  - Encode each thread as a lane
  - Encode time of event as position on x-axis
  - Encode thread of event as position on y-axis
  - Encode parent/child relationships with connecting lines
Dependency Graph

- What: derived items
  - Total messages issued by a service
- Why:
  - Understand dependency relationship between services
- How: arrange services into a node-link graph
  - Service is a node
  - Dependency is a link between nodes
- How: encode
  - Encode degree of a node with area of circle
Compare

- **What: data**
  - 2 traces
- **What: derived**
  - For each event add it to group between 1-3
- **Why: find difference between traces**
- **How: arrange events into a node-link graph**
  - Event is a node
  - Link is parent-child relationship between nodes
- **How: encode**
  - Encode group 3 nodes as squares and groups 1-2 as circles
  - Encode group of event by node colour
- **How: aggregate**
  - Aggregate group 3 nodes so that it maintains its structure
Aggregate

- **What:** data
  - Traces

- **Why:** see the big picture

- **How:** arrange events into a node-link graph
  - Event is a node
  - Link is parent-child relationship between nodes

- **How:** aggregate
  - Aggregate events from same source code line

- **How:** encode
  - Encode number of events in a node with luminance
    - High luminance = many events
    - Low luminance = few events
Discussion

- Overview page provides a nice way of exploring the trace dataset.
- First viz tool to provide source code integration for distributed traces.
- Graph layouts are not great. Suffer from hairball effect.
- The compare and aggregate idioms are confusing for users.
Future Work

- Better layouts for graph visualizations to remove hairball effect
- Add detail view for swimlane
- Add viz idiom for comparing 1 trace against an aggregation of traces
- Add viz idiom for comparing 2 different aggregation of traces
- Integrate/Replace existing tools :)
- Usability Study
- Integrate it with backend server of X-Trace tracing system.