Tracing Service for Tracing-Driven Glidein Optimizations

Namratha Urs, John Tyndall, Ralph Ortiz, Marco Mambelli
Fermi National Accelerator Laboratory
Throughput Computing 2023
14 July 2023, Madison, WI
GlideinWMS (GWMS)

- Simplifies resource provisioning for distributed high throughput computing (dHTC)
  - From heterogenous resources to uniform virtual clusters
  - Leverages HTCondor for scheduling and job control

- Three components:
  1. **Frontend**: look for user jobs and request the Factory to provide glideins
  2. **Factory**: self-advertise, listen for requests from Frontend and submit glideins
  3. **Glideins**: provide a customized execution environment for user jobs

Animation Credits: [GWMS Project Documentation](#)
Glideins (aka Pilot Jobs)

- Provide tailored execution environments for user jobs to run on diverse, complex resources in a distributed setting
  - Scout for resources and validate the worker node
  - Customize the worker node (environment, libraries, containers, etc.)
  - Starts and monitors the execution of the job

Provides a tested and customized execute node to HTCondor

Illustration adapted from here
Current State of Telemetry in GWMS

- **Telemetry**: data emitted from a system about its behavior in the form of logs, metrics or traces¹

  - Different logging and monitoring capabilities exist

  - Telemetry in a distributed, heterogenous ecosystem is essential
    - Cater to multi-location, multi-language, multi-channel aspects
    - Maintain security
    - Diversify paths

Examples

1. Glidein logging to standard output and error; report back to the Factory
2. A remote logging API to generate shards that coalesce to a single log sent to a web server
3. Packing HTCondor logs as a base64 encoded zip file
4. GlideinMonitor that filters, archives and unpacks glidein logs
5. Feed of logs to ElasticSearch
6. Logs fed to remote syslog by OSG

¹ [https://opentelemetry.io/docs/concepts/observability-primer/#reliability--metrics](https://opentelemetry.io/docs/concepts/observability-primer/#reliability--metrics)
Motivation

• Existing heuristics useful for Glidein optimizations
  • E.g., how many to request, how long to wait for new jobs, the wait time for termination etc.

• Awareness of resources utilized and time exhausted can be invaluable
  • Failure and resubmission details of Glideins may help in their optimization

• Distributed high throughput computing + production setting $\rightarrow$ CHALLENGE!
  • Difficulty in understanding the cause of performance problems/degradation
  • Lack of visibility directly proportional to our inability to reproduce issues locally

Observability in distributed computing ecosystems can make a difference!
Proposed Solution

• Use of distributed tracing from two perspectives:

1. **Generate traces for Glideins** [internal-facing]
   • **Goal:** Insights on end-to-end Glidein lifecycle for operators and developers
   • Find answers for optimization questions

2. **Provide a tracing service for user jobs** [user-facing]
   • **Goal:** Why is my job not running yet?
   • Motivate users to run jobs efficiently
(Distributed) Tracing

<table>
<thead>
<tr>
<th>Date &amp; Time</th>
<th>Status of Item</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2, 2021</td>
<td>Delivered, Front Door/Porch</td>
<td>SILVER SPRING, MD 20901</td>
</tr>
<tr>
<td>12:12 pm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>January 2, 2021</td>
<td>Out for Delivery</td>
<td>SILVER SPRING, MD 20901</td>
</tr>
<tr>
<td>6:10 am</td>
<td></td>
<td></td>
</tr>
<tr>
<td>January 2, 2021</td>
<td>Arrived at Post Office</td>
<td>SILVER SPRING, MD 20904</td>
</tr>
<tr>
<td>5:00 am</td>
<td></td>
<td></td>
</tr>
<tr>
<td>January 1, 2021</td>
<td>In Transit to Next Facility</td>
<td></td>
</tr>
<tr>
<td>December 31, 2020</td>
<td>Arrived at USPS Facility</td>
<td>WASHINGTON DC NETWORK DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>10:51 am</td>
<td></td>
<td></td>
</tr>
<tr>
<td>December 29, 2020</td>
<td>Departed USPS Regional Facility</td>
<td>BALTIMORE MD DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>2:50 pm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>December 27, 2020</td>
<td>Arrived at USPS Regional Destination Facility</td>
<td>BALTIMORE MD DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>9:50 pm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>December 22, 2020</td>
<td>Departed USPS Regional Origin Facility</td>
<td>FORT WORTH TX DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>1:53 am</td>
<td></td>
<td></td>
</tr>
<tr>
<td>December 19, 2020</td>
<td>Arrived at USPS Regional Origin Facility</td>
<td>FORT WORTH TX DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>8:16 pm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>December 19, 2020</td>
<td>Accepted at USPS Regional Origin Facility</td>
<td>FORT WORTH TX DISTRIBUTION CENTER</td>
</tr>
<tr>
<td>7:01 pm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the USPS Tracking Plus Statement was prepared before the item reached its final destination, the tracking history will continue to refresh and populate status updates as the item travels through the USPS network. Return to USPS.com to prepare a new USPS Tracking Plus Statement.

Graphic Credits: USPS

Client:

- /api
- /payment Gateway
- /authZ
- DB
- Ext. Merchant
- /dispatch
- /dispatch/search
- /poll
- /poll
- /poll
- /pollDriver/{id}

Graphic Credits: OpenTelemetry Docs
(Distributed) Tracing

- One of the three pillars of observability
- Captures “big picture” information about the system when an event flows from start to finish
- Facilitates debugging and diagnosis of common problems with software
- Useful for understanding the behavior of distributed systems
- Not the same as monitoring\(^2\)
  - Monitoring $\rightarrow$ something is wrong
  - Observability $\rightarrow$ what is wrong and why it happened using telemetry data

Illustration inspired from OpenTelemetry Docs

Generating and Processing Telemetry

1. System instrumentation

**Focus**: generate, collect, manage, export traces

- OpenCensus
- OpenTracing
- **OpenTelemetry**

2. Do something with it (aka the back-end)

**Focus**: storage, visualize, analyze traces

- Zipkin
- Prometheus
- **Jaeger**
- Tempo
- Honeycomb
- Datadog
- DynaTrace …
<table>
<thead>
<tr>
<th>OpenTelemetry³ (OTel)</th>
<th>Jaeger⁴</th>
</tr>
</thead>
</table>
| • Open-source, vendor-agnostic and tool-agnostic framework  
  - Specification  
  - Protocol  
  - Semantic conventions  
  - APIs, SDKs and much more | • Open-source distributed tracing system |
| • Data owned by YOU! | • Supports  
  - OpenTelemetry SDKs  
  - Storage backends  
    - Cassandra  
    - ElasticSearch  
    - In-memory  
    - ... | • Minimal deployment (all-in-one binary)  
| • Single set of APIs and conventions | • Scalable as needed  
| • Standardized data format | |
| • Easy integrations, extensible, native support across vendors | |

³ https://opentelemetry.io/docs/what-is-opentelemetry/  
⁴ https://github.com/jaegertracing/jaeger

Jaeger is an open-source distributed tracing system that supports OpenTelemetry SDKs and various storage backends such as Cassandra, ElasticSearch, and in-memory storage. It also offers scalable deployment as needed.
GWMS Tracing Service Architecture

Access Point (AP) ➔ Factory ➔ Compute Entrypoint ➔ Monitoring Server

Frontend ➔ Factory ➔ Compute Entrypoint ➔ Monitoring Server

Monitoring Server ➔ Tracing Server ➔ Jaeger

Web UI
Next Steps and Beyond…

1. Working prototype
   - Glidein instrumentation
   - Job instrumentation via user-facing API
   - Investigate generated traces using Jaeger UI

2. Integration with GlideinWMS
   - Knobs to trigger trace generation
   - REST interfaces to query and visualize stored trace data

3. Instrumentation for other GWMS components

4. Migrate to Grafana Tempo (long-term)

* Work in progress icon from here
Acknowledgements

• Special shout-out: John Tyndall and Ralph Ortiz!
• DOE Omni Technology Allliance Internship Program
• Kevin Retzke (FNAL) for being our go-to person for Observability best practices

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.
Summary

• Make use of existing heuristics on Glidein behavior
• Enable Glideins to generate traces as they progress through their lifecycle
• Provide custom API to generate traces during user job execution
• Make collected trace data available via a UI and REST interface
• With distributed tracing:
  • Drive optimizations of Glideins (and eventually, other GWMS components)
  • Corroborate efficiency improvements using statistics on heuristics
  • Increase understanding of GWMS (as a framework) and in real-time dHTC settings
  • Enable observability in dHTC ecosystems

Questions and/or suggestions? Reach us at glideinwms-support@fnal.gov