Academy Agent Observability over Diaspora

Gulesh Shukla, Advisor: Ben Clifford

APEX Summit 2026 — University of Chicago, MS in Computer Science

Agenda

1. Introduction
2. Problem Statement
3. Diaspora overview
4. Why Diaspora?
5. What I built?
6. Performance Analysis
7. Closing/Q&A

Introduction

- MS in Computer Science, University of Chicago
- BS in Computer Science, Haverford College
- 3 years as Software Engineer, Dassault Systèmes
- Core interest: low-level, performance-critical systems (C/C++)
- This summer: distributed logging in Python, Globus Labs
- Graduating December 2026 — actively looking for full-time opportunities

Problem Statement

- Academy agents run as separate, distributed processes
- Built-in logging (console, file, JSON-pool) is process-local
- No config merges writers across processes — even Academy's own "pool" logger gives each process its own file
- Result: no durability, no central place to read from

Diaspora Overview

- Kafka-based event fabric — Event-Driven Architecture
- Globus auth → per-user namespace (prefixes your topics)
- Python client: create / clear / delete topics
- producer.send(topic, event) + consumer fetch (one-shot or live)
- One event = one log line — same unit as a line in a traditional log file

Why Diaspora?

- **Persistent** — topic retains messages whether or not a consumer is listening; replay by timestamp
- **Concentrated** — many distributed producers write to one topic concurrently
- **Decoupled, higher throughput** — measured ~35–55× queue's throughput ceiling at matched volume (see Performance Analysis)
- **Rich detail for free** — full structured log record ships via Academy's own config system
- **Just need a Globus account** — no separate signup, API keys, or infra: your existing Globus identity is your Diaspora namespace and Kafka credentials

What I built - Features

- **academy-diaspora-logger** — pip-installable Python package
- Built on Academy's own LogConfig — DiasporaLogConfig extends it, so every agent gets Academy's full structured log record for free, no bespoke serialization layer
- **Producer**: local execution builds a KafkaProducer from cached Globus credentials; Globus Compute (remote) uses .prefetch() to embed AWS IAM keys locally, so the remote worker never touches Globus auth itself
- **Consumer**, two methods: one-shot replay by timestamp (get_diaspora_events), or a persistent live stream (stream_diaspora_events)
- **CLI**: diaspora setup / context / clear

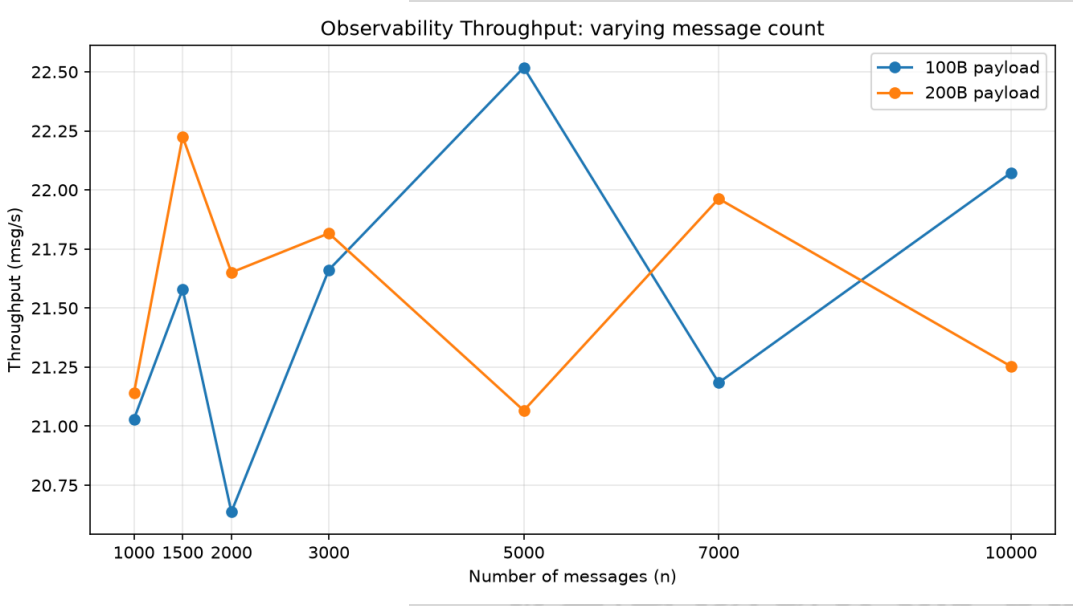
What I built - Application

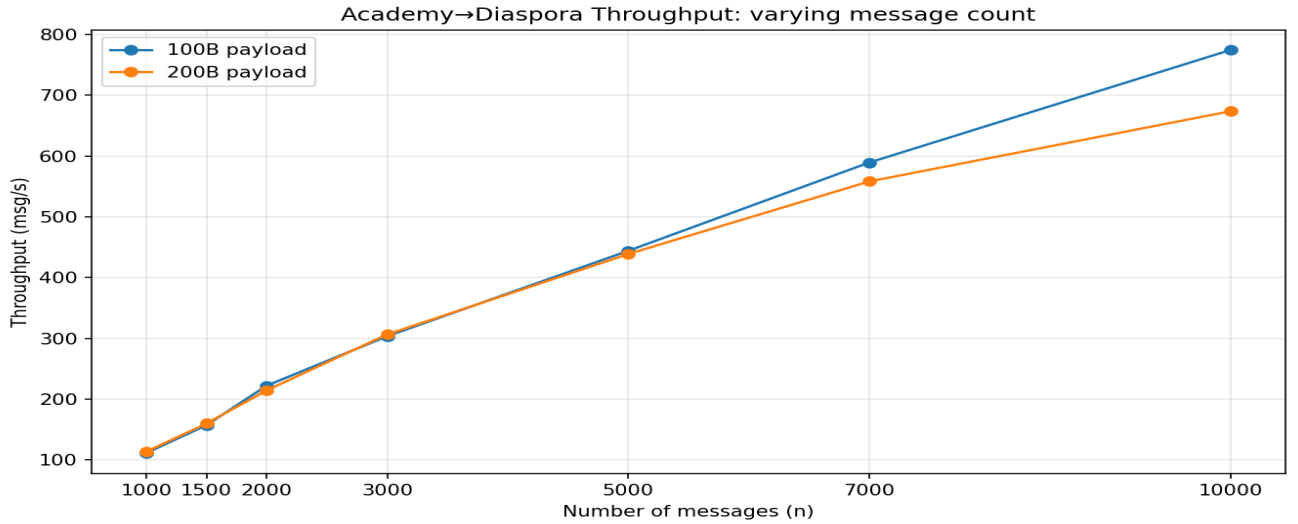
- Applied to the UserAgent/MonitoredAgent dashboard as a launch-time log_backend choice — queue or diaspora, a full swap, not a side channel
- MonitoredAgent = producer — Diaspora handler attached to the root logger
- UserAgent = consumer — uses the **live-stream** method from the previous slide, feeding the same Dashboard.push_log() / SSE pipeline the queue path already used
- Result: the LOGS tab looks identical either way — only the transport underneath changes

Queue = today's UserAgent↔MonitoredAgent path: a synchronous, in-memory action call per log line, no persistence.
Message = one log line.

Performance Analysis – Queue vs Diaspora

- Queue: ~20–23 msg/s remote — flat from n=1,000 to n=10,000
- Diaspora: ~111 msg/s at n=1,000 → ~775 msg/s at n=10,000 (remote, 100B)
- Diaspora, single sender best: 160,000 messages, **0 loss**, **1,239.8 msg/s** — ~55× queue's best observed rate
- Tradeoff: Diaspora's per-message latency is higher (~57–131ms vs queue's ~22–56ms p50) — the cost of the extra Kafka/Diaspora hop



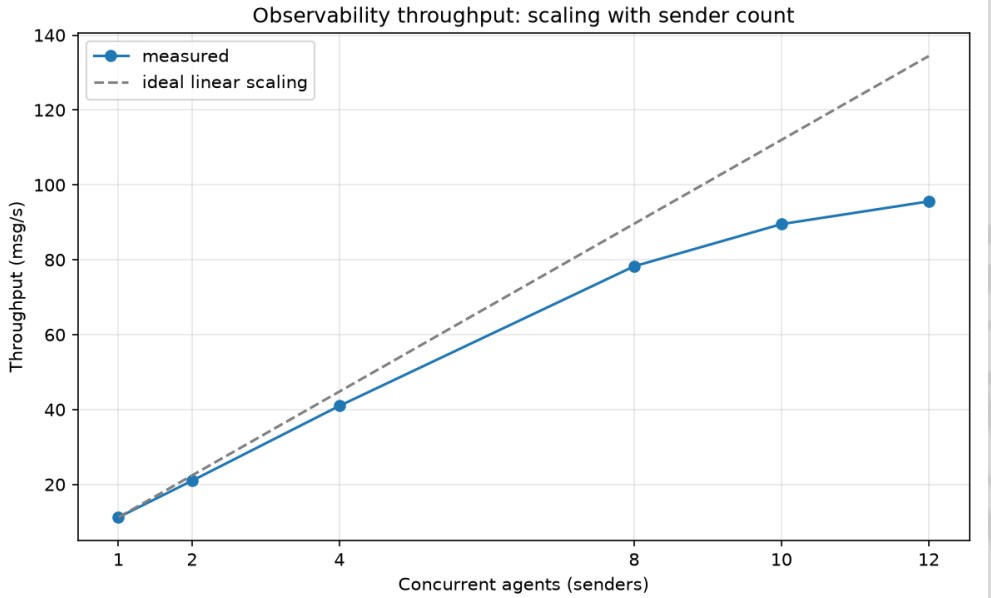


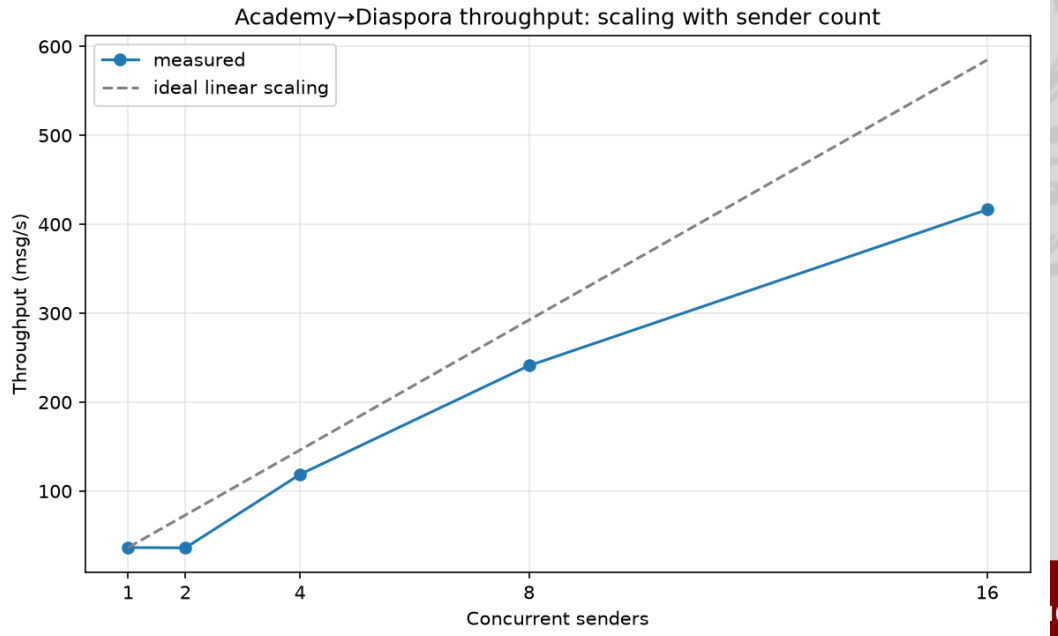
Queue = today's UserAgent↔MonitoredAgent path: a synchronous, in-memory action call per log line, no persistence.
Message = one log line.

Performance Analysis – Queue vs Diaspora

Scaling

- Both track close to the ideal-linear line as concurrent senders increase
- Queue tops out under **100 msg/s at 12 agents**; Diaspora tops out over **400 msg/s at 16 senders**
- Note : Each agent is only sending 100 log messages





Queue = today's UserAgent↔MonitoredAgent path: a synchronous, in-memory action call per log line, no persistence.
Message = one log line.

Closing

- Durable, centralized, config-native logging for distributed agents
- Benchmarked, and live in a real dashboard
- ***pip install academy-diaspora-logger***

Thank you!

gulesh.Shukla.7@gmail.com | https://gulesh.github.io/ | https://www.linkedin.com/in/gulesh-shukla/