OpenTelemetry Tracing for Monoliths

Phillip Carter, Principal PM @ Honeycomb.io



Monoliths vs. Microservies

Often a meaningless distinction

A common "monolith"

- Most of the app is a single process
- Running behind a load balancer
- Talks to a database server (or two)
- Maybe talks to an auth server somewhere
- Might have an external vendor's API somewhere
- ...
- It's already a distributed system



Microservices grow too!

- Just a little more code here and there...
- "This code is poorly organized, let's modularize it"
- ...
- Not uncommon to end up with distributed monoliths
- ...
- And that's okay!



You already need distributed tracing

- Tracing is how you correlate events happening in different parts of a system
 - How many database calls did this request make?
 - What's my average overall system latency per request? P95? P50?
 - Is that API we're calling the cause of a slowdown?
 - o etc.
- Reign in problems at the edges of your systems with automatic instrumentation



Example: Intercom

- Several services, but one big 'ole monolith doing the bulk of work
- Added instrumentation for the edges of their systems
 - API calls
 - DB calls
 - Calling other services
- Iterated to add quality tracing instrumentation for their main monolithic service



Tracing for a single service

Call it a monolith if you like!

Traces are better structured logs

- A trace is a collection of spans correlated by a trace id that encode an order of operations and parent-child hierarchy
- A span is just a structured log with some stuff on it
 - Name
 - Span ID
 - Parent Span ID
 - Key-value pairs of data
 - A list of span events
 - A list of span links



ns' current_sign_in_at = '2022-06-24 10:01:49', 'admins' sign_in_count' = 2. 'admins' 'updated_at' = '2022-06-24 10:01:49' WHEE 'admins' 'id' = 112 /*application:IntercomApp.controller:admin_session | Application | ← Trace b2a47a1f-fbd1-443a-830e-13912f36b538 at 2022-06-24 10:37:53 intercom_app > MatchingSystem::Private::... [b2088do2-0b87-4dlo-889b-f35be8eb0205] [Redis] command=EXEC orgs= 1938452-987-445-489-798-6482033 [Reigi Oil.time-8.13 m]
[D284852-987-445-189-798-6482033] [Reigi Oil.time-8.13 m]
[D284852-987-4482033] [R ckpointProcessor.process! Search spans 0 spans with errors | | Fields D2885602-08E7-4dia-8890-47550e868285] Cache write: IDC;8:blob:Assignee:14081091992455931172:112 ({:compress->false, :cross_app_protections->false, :raise_errors->false, :compress_threshold->1824, :expi Distribution of span duration ① [b2888dd_0887-4dla-8890-f35bede08265] (ache write: IDC:8:attrs:Assignee:id:id/deleted:11785111631378283444 ({:compress>>false, :cross_app_protections>>false, :raise_errors>>false, :compress_threshold>> 1 MatchingSystem: Priv...noozedMatchProcessor intercom_app 1.964ms [b2088do2-0b87-4d]a-8890-f35be8eb0205] Cache write: IDC:8:attrs:Assignee:id:email:17423680638938127583 ({:compress-false, :cross_app_protections->false, :raise_errors->false, :compress_threshold->1024 regires.in=1800 seconds. [feature_nome=nil])
[D288662-087-4-id-880-7550e402925] Aminimisasion Delete Ali (2.8ms) DULTE FROM odmin_pessions | MRIE odmin_messions | massion_id = "Smootl/Scill/27796288041753defe2c594381a6673d55796 0.6172ms 1 MatchingSystem: Pri_chProcessor.process! intercom_app 775550fc000bd7f4/Eb/20mffdee367430bb75638e91c76dd5' /*application Interconden controller admin sessions action reports line /appl/controller/admin sessions controller rb:114 in 'undate MatchingSystem:__snoozed_matches! intercom_app 0.2572ms [b2088do2-0b87-4d1a-889b-f35be8eb0205] [Redis] command-MULTI args= 1.629ms MatchingSystem:Priva_r.CheckpointProcessor intercom_app MatchingSystem::Pri_ntProcessor.process1 intercom_app 0.3100ms | ser meter, "alant, assistation", "assistation," "a "Assistation," "a "Assistation," and "assistation," and "assistation," assistant, "a "Assistation," assistant, 6 Api: Messenger: Web: Pin...oller.build_ping_response intercom_app 954.5ms Filter fields and values in span [b2088do2-9b87-4dlo-889b-f35heReb0205] [Redis] command=FXEC gras-21 Api:Ping.initialize 352.9ms [1038662-867-466-289-195e4e8000] [Delis] onl.timed.79 %:
[Delisid onltwin [Intercassing, controller:delin_serior, action:renze, line:/app/controllers/admin_seriors_controller./billin_seriors_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_controllers_control 1 UserInbox.List.unrea..ps_for_active_company intercom_app 159.3ms DANAGA-092-440-440-450-780-640027]. Abrickszion Creste (1.0e) INSER IND Gebin.aszions Creste (1.7) sexion.id [crested in [d]] sex [11] UserInbox List unre. a for active company intercom app 151.5ms creats, lies (app/cat-rallers/admin.assxions, cartroller-rb:114 in "update, assxions, multi-sethad, sign-rich", new force of the controllers/admin.assxions, cartroller-rb:114 in "update, assxions, multi-sethad, sign-rich", assxions, action:create, line:/app/cantrollers/admin.assxions, cortroller-rb:114 in "update, assxions, cartroller-rb:114 in "update, assxions, action:create, line:/app/cantrollers/admin.assxions, cortroller-rb:114 in "update, assxions, action:create, line:/app/cantrollers/admin.assxions, action:create, line:/app.create, line:/app.create, line:/app.create, line:/app.create, line:/app.create, line:/app.create, line:/app.create, line:/app.cre str app_id_code SHOW FULL FIELDS _ a' 'conversations' Sharded DB Readers 5.815ms [b2988dd2-db87-4dla-8899-f35be8e08295] AdminSession Load (1.2ms) SELECT 'admin_sessions' FROM 'admin_sessions' WHERE 'admin_sessions' admin_id' = 112 ORDER BY created at desc LINIT 184467449737895 tx2p130c 0.8453ms SIALS OF ALL IS Propried that is benerously, control for an inclusion, control for all and an inclusion of the control for all and an including an inclusion of the control for all and an including and an including an includ air controller SELECT 'conversatio_sations', 'read' = 7 Sharded DB Readers 2.724ms Api::Messenger::Web::PingController SqlFingerprintWorker 0.1978ms 1.007ms -- MURTI SADD queues "\"arms\" EXEC redis true [35aeff-dba1-44a3-3871-9955974dc62] Admin Load (0.8ms) Stict admins, FROM admins admins type = "Admin AND admins, id = 112 ONDER BY admins, id ASC LDMIT 1/"application:Inte SELECT 'conversati...ns'.'id' IN (RANGE) Sharded DB Readers 0.8411ms (Book) detailed trace we_controller.rb:Z9:in 'redirect_to_sign_in_if_logged_out'*/ [158aafff-dhal-44a3-98f1-99a5d974dc62] [Badis] command-MilTI acase [Staeff-ded-443-981]-Mesorracoz [Lenis] comand-MCLT argu-[Staeff-ded-443-981]-Mesorracoz [Lenis] comand-MCDB argu-"gaves" "sql_fingerprints" [Staeff-ded-443-981]-Mesorracoz [Lenis] comand-MCDB argu-"gaves: "sql_fingerprints" ["{\"class\":\"SqlFingerprintWorker\",\"argu\":[{\"time\":1.28002296875,\"raw_sql\":\"StLECT "admins".* FROM SalFingerprintWorker 0.2483ms MULTI SADD queues ...".\"args\".... EXEC redis 0.9130ms (m) duration_ms 0.30996499117463827 11 70ms ConversationsServi_hardedConversation intercom_app

Credit: Intercom Engineering Blog



V1-23

How to trace a single service

- Add autoinstrumentation / instrumentation libraries to track incoming/outgoing requests
- Use the **opentelemetry api** in your language to create more spans
- Change thinking from "log these things" to "track this operation with a span"
 - Start the span at the start of a function/method
 - Add relevant "log data" as attributes to the span
 - Use Span Events to capture "this thing happened at this time!", like exceptions
 - End the span at the end of the function



Organizing spans in big services

- Big services organized into modules/bounded contexts/etc. can have that organization represented in telemetry
- Create a wrapper function/method for starting a span
- Calling the wrapper adds a name for your module/bounded context/etc.
 - Wrapper always adds a bounded.context attribute with a configured name
 - Bounded Context name can be passed in, configuration-driven, or defined by something else (e.g., assembly name for .NET apps)
- Now every span has the right organizational attribute on it to let you differentiate them!



GenerateQueryFromPrompt	poodle	8.149s
- launchdarkly.BoolVariation	poodle	40.6µs
- launchdarkly.NumberVariation	poodle	58.4µs
- launchdarkly.StringVariation	poodle	31.0µs
launchdarkly.StringVariation	poodle	19.1µs
Schema Store Get	poodle	23.9µs
- Schema Store Get	poodle	10.7µs
queryml.FindAllSuggestedQueriesForDataset	poodle	106.7ms
1 queryml.MostRelevantColumns	poodle	5.815s
queryml.filterColumnsUsingEmbeddings	poodle	5.813s
queryml.embeddings.Embeddings	poodle	188.7ms
queryml.embeddings.getFromCache	poodle	0.8656ms
queryml.embeddshalEmbeddings	poodle	23.3µs
TruncateColumnList	poodle	72.0µs
—• openai.Embeddings	poodle	186.7ms
- queryml.embeddings.savelnCache	poodle	0.8410ms
queryml.embeddings.Embeddings	poodle	5.612s
queryml.embeddings.getFromCache	poodle	727.2ms
queryml.embeddshalEmbeddings	poodle	608.2ms
TruncateColumnList	poodle	11.60ms
—• openai.Embeddings	poodle	4.782s
- queryml.embeddings.saveInCache	poodle	84.99ms
CreateChatPrompt	poodle	24.47ms
- CreateCustomExamplesPromptText	poodle	0.1182ms
TruncateColumnList	poodle	0.7687ms
1 queryml.GenerateQuery	poodle	2.193s
openai.ChatCompletion	poodle	2.193s
- renderJSON	poodle	44.1µs



Bring your logs along for the ride!

Don't throw away your past work

OpenTelemetry Logs Bridge

- Set up OTel Logs in your SDK setup code
- Have a trace, such as from request autoinstrumentation
- And that's it!
 - Each log gets emitted as an OTel log
 - Each log record has the span id and trace id for the current active span/trace at the time that the log gets created
 - Automatic correlation between existing logs and a trace



Create new spans for new things

- Track operations with spans in code for new instrumentation
 - Remember: spans are like structured logs, but better
 - New instrumentation -> spans
 - Old instrumentation -> logs
- Keep your existing logs, or not, up to you!
 - Migrate to spans if you feel like it
 - Don't migrate these logs if they work fine



Takeaways for your team

Traces are for monoliths!

- Distributed traces doesn't mean you need tons of services
- You can add tracing to one service and get a lot of value out of it
- Traces are the superior, evolved form of structured logging
- You can bring your existing logs with you





Questions?



www.honeycomb.io