QUERYING

TRACES

ADMIN

- Midterm scores, etc.
  - Will send out over weekend/by Monday

- Final Project
  - ↳ Haven't really heard back for the cases where I asked for clarification
  - → NEW TASK: This week: Post what you plan to work on + rough timeline

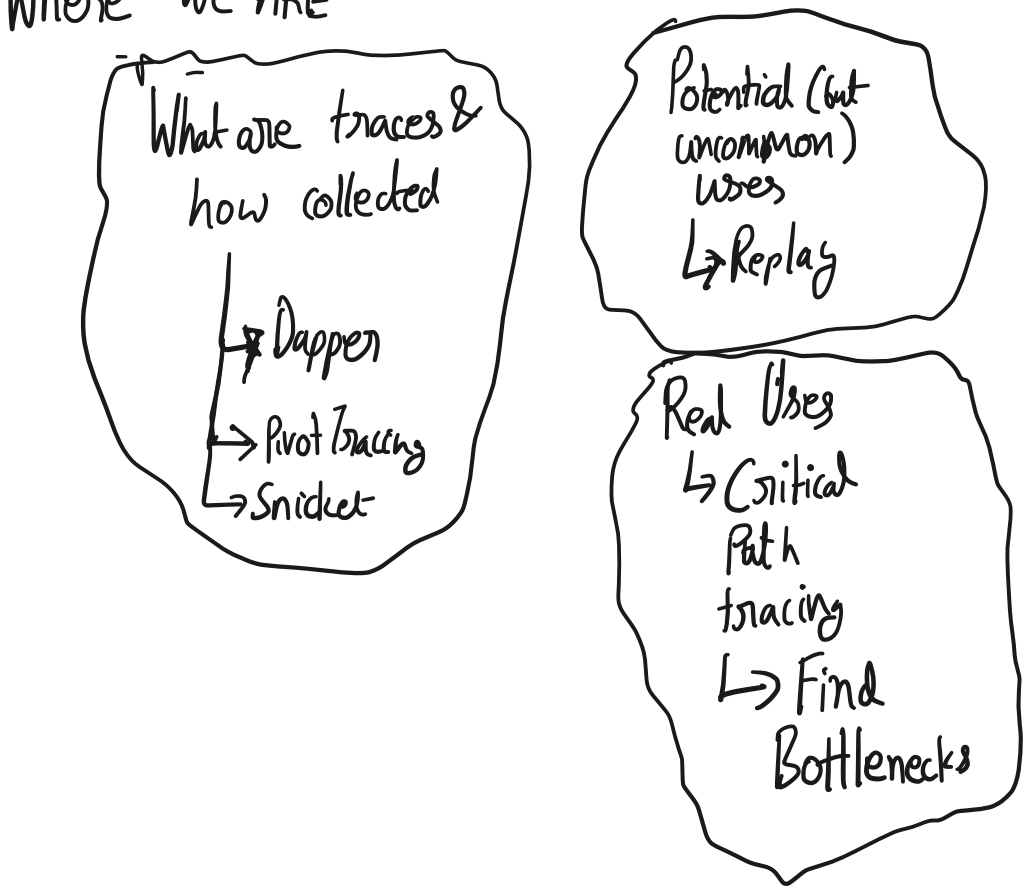
START WORKING ON FINAL PROJECT!!

- STATE OF THE COURSE

→ Topics of interest?

→ Any interest in student presenters?

Where We ARE



Today: "Querying" traces.

↳ What request traces do you want to look at?

↳ Depends on what one is trying to do?

↳ Why is tail-latency high?

- Cannot just look at critical path

↳ Must decide what requests to check

↳ Localize what part of a service

...

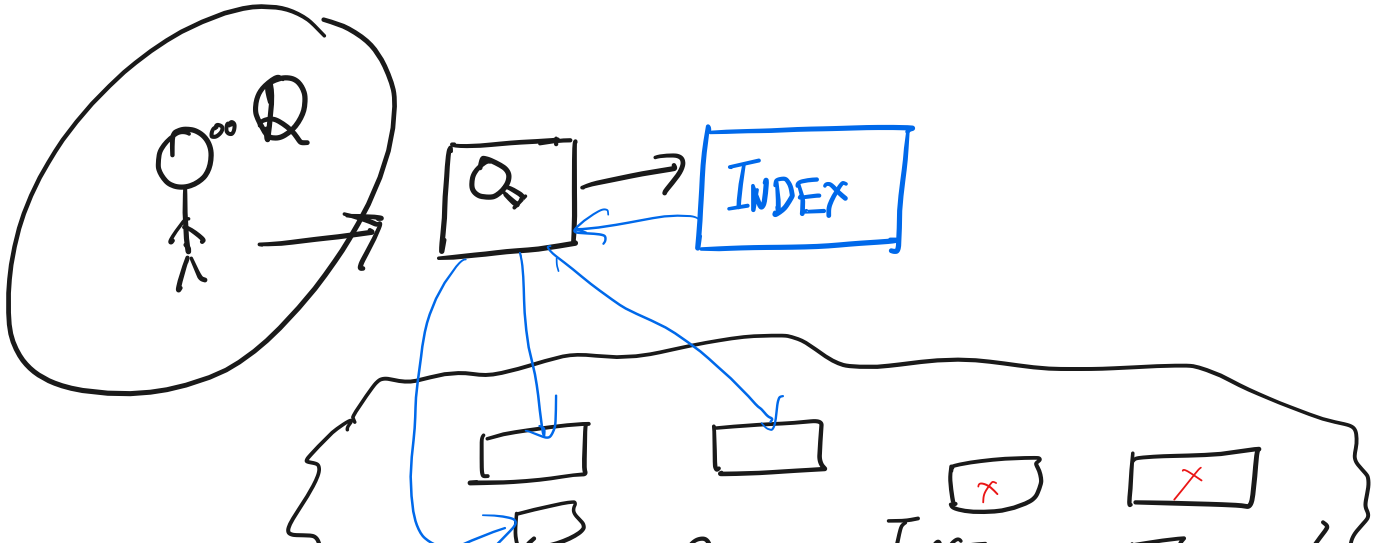
↳ Why are requests slower now?

↳ Pick two or more request traces & compare.

But which ones?

...

What is desirable



REQUEST TRACES

Another way to look at the problem: How to build an INDEX?

Q: What type of index is useful?

↳ Depends on common queries.

[tprof] ↳ Find requests whose latency / resource utilization / ... is > abs value / nth percentile / average / ...

[tprof] → Find requests where component has

[GMTA] → Find requests corresponding to some business logic

[GMTA] ↳ Find requests that are affected by a recent code change.

...

Of course we also want to allow people to combine these queries.

Want indices with two types of information

Type (business logic, code change, ...) → R_1, R_2, R_3, ... + Aggregate Statistic

R_4
⋮

(But over what?)

New Problems

(a) How to update index?

Request —assign→ Type

+ Compute aggregate

(b) Index selection: What types to use-
↳ Automatic [tprof] (sort of)
→ Semi-automatic [GMTA]

Observations for computing aggregates

↳ Need to compute aggregates over groups

Q1: How to group?

- Determines what types of aggregates are meaningful
  ↳ OR WHAT THEY SHOW.

Example: - 99 percentile of B's latency when called from A must be computed only on A→B

+ [illegible] → B when processing

- Median input for requests of type X

- ...

Observe: Can always compute an aggregate on demand, but not efficient.

Desirable: Maintain stats ahead of time by choosing groupings that are likely to be DESIRABLE.

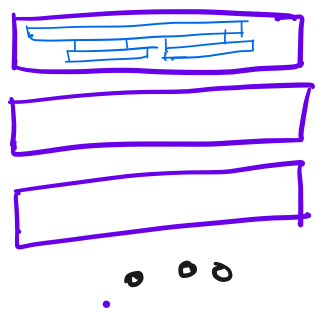
What Groupings are desirable

- Types indexed
  - Request type
  - user
  - Business flow
  - ...
- Finer grained structures
  - Service
  - Request-level call graph
  - subspans?
  - ...

  ] Will discuss later.

Q2: What aggregates to maintain?

Ex. Request type



} - stats on response latency

Note: Related to but not the same as groupings.

Why?

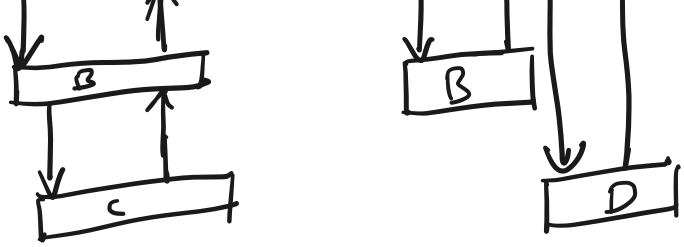
So really, we are back to the question of what granularity of stats/aggregation are people likely to find useful.

The obvious ones:

- Request type

- By service.

Looking Beyond (sort of)



Similar control/data flow.

Why?

Within a service

```
. . .
f1(..)
lock(x)   } Time spent here ?
  :       }   ↳ Or elsewhere.
  :       }
unlock(x) }
```

f2(...)

...

Why interesting?

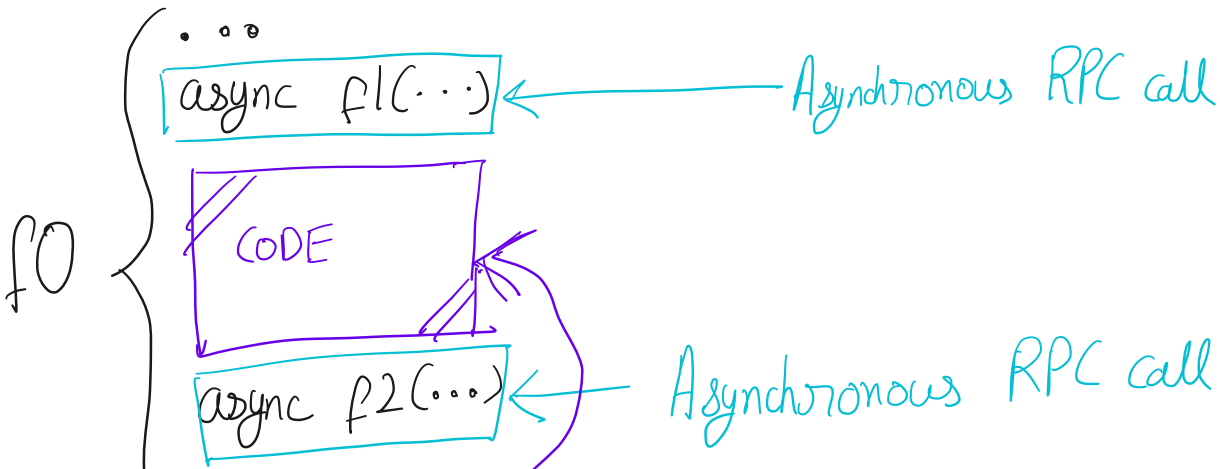
Problem

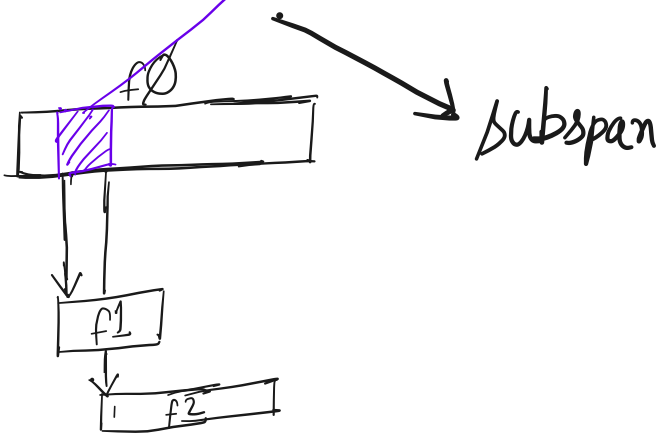
↳ Generally do not have this information*

↳ Could add it but...

↳ Can we infer it?

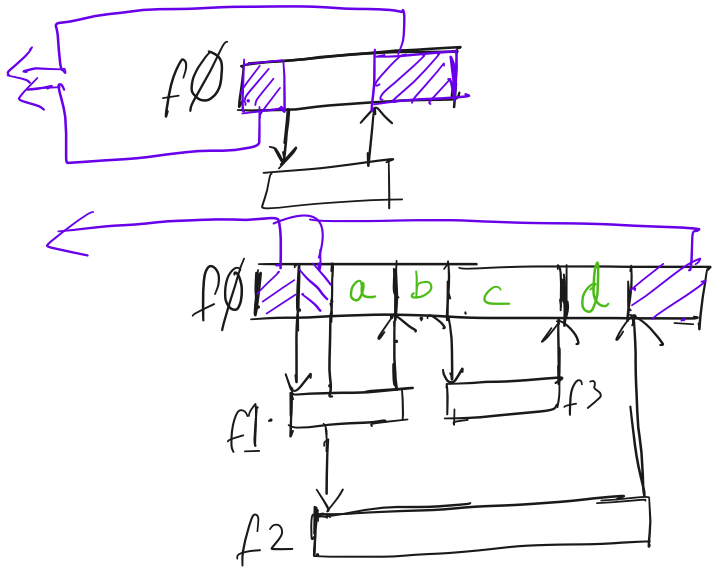
Answer: Not really, but can try to approximate in some cases



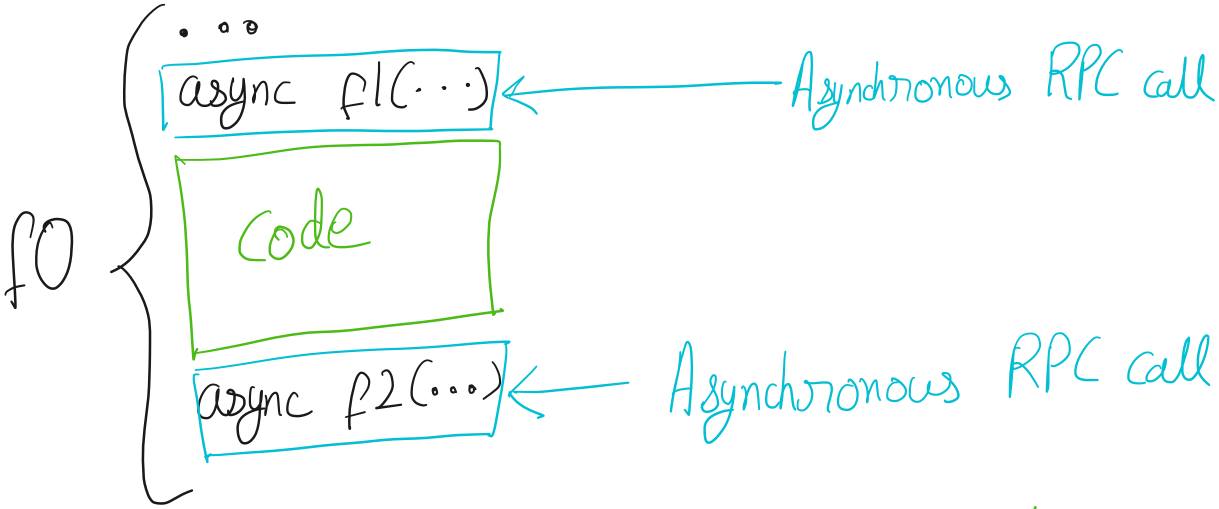


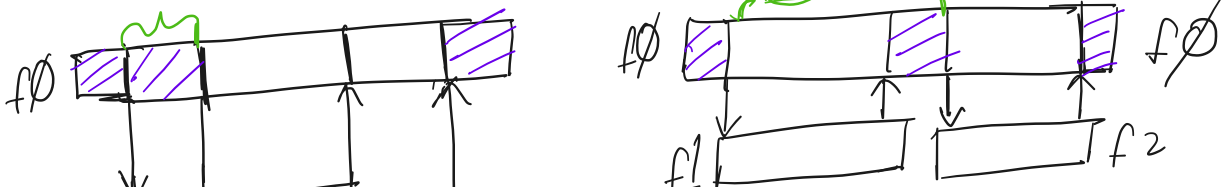
More generally

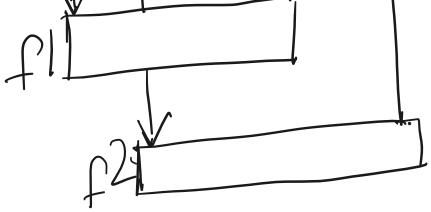
- fØ cannot be waiting for an RPC to complete (?)



Problem: Not very accurate.







Can we use previous tricks to do better?

- Aggregate by dependency/service

- Aggregate by "Business Process"

Where We Are

- Have a way to assign type to request
  - ↳ Reusing some ideas from before + information in traces
    - * Request type
    - * Dependency information → request type by mining previous traces
      - ← service granularity
    - * Call-graph → Spans
    - * ...

- Have a way to aggregate metrics for each type
  - ↳ Using info contained directly in traces + some inferred information

Q: How to use any of this?

- Great that one can query and get requests, but how do people use it in general?

- CPA: Points out bottleneck - allows one to decide where to focus optimization efforts.

- But in general often want to determine what is different

  ↳ How do slow & fast request differ?

  → How do old & new system differ for requests?

  → ..