Big Data

and

Machine Learning

Spring 2023

Hi, I am Panda.

This is a seminar on Big Data and ML

↳ But we are not really going to look at this.

→ Why? Too broad, unclear what I can say that is not better said by an ML class.

Instead going to take a page from last year

Focus on a single area / topic.

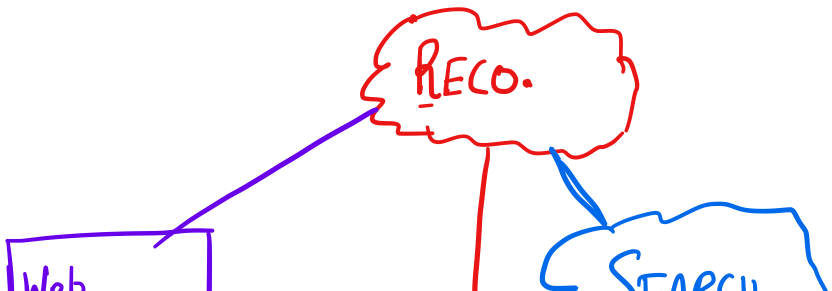
Last year: Scheduling

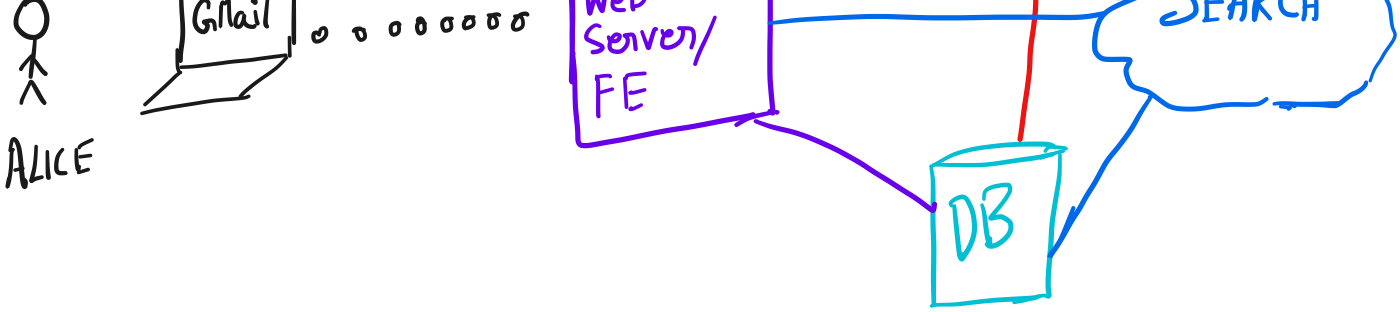
This year: Distributed tracing for debugging & profiling.

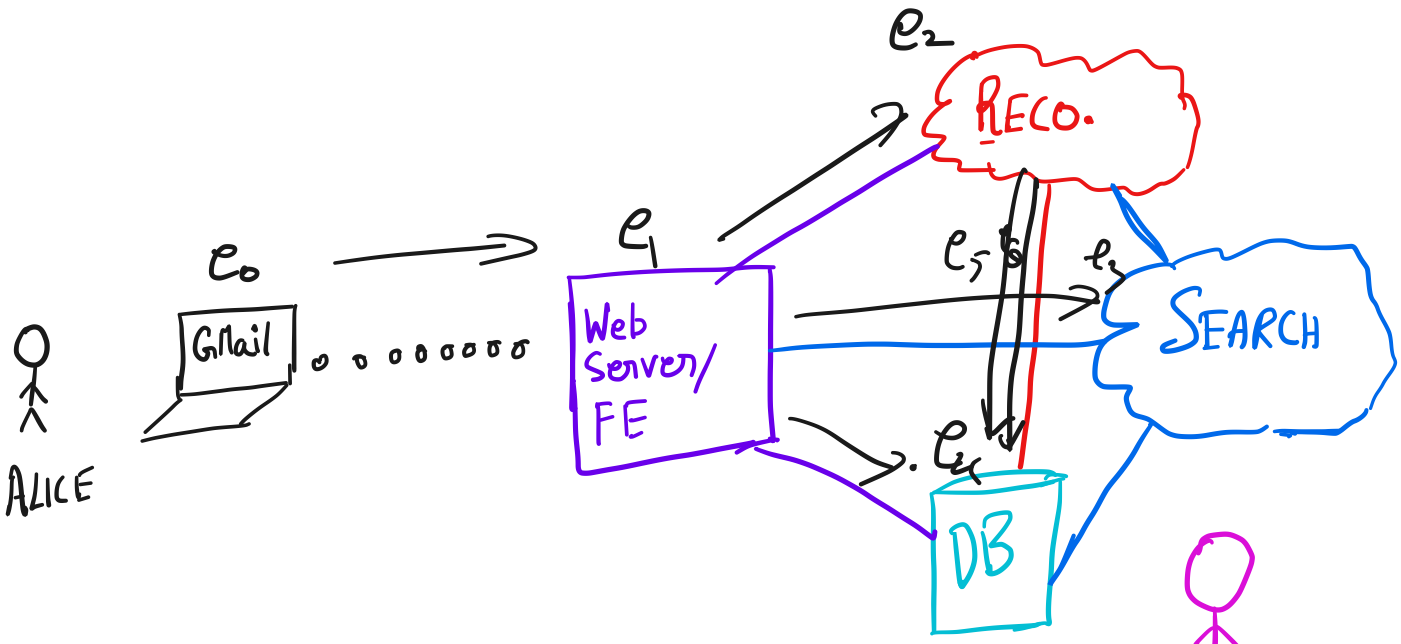
Three initial questions

- What?
- Why?
- How?
  ↳ Class mechanics

What?

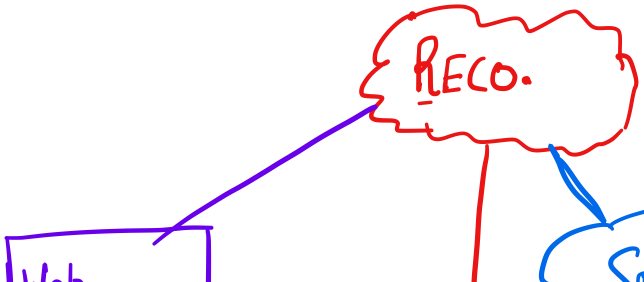






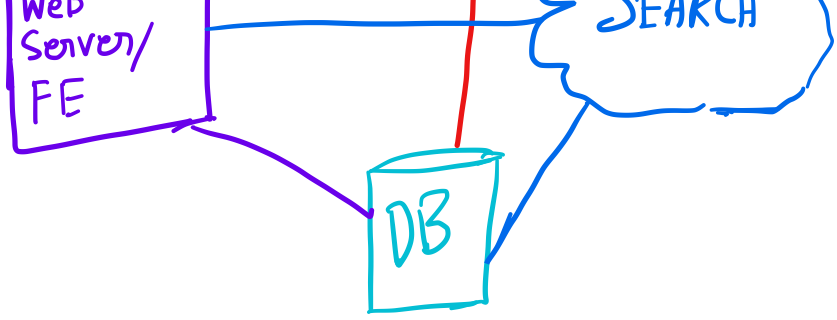
Why is Alice unable to search for e-mails about PIZZA?

– BETH SRE



- Why is Alice unable to search for e-mails about PIZZA?

BETH SRE



- Why is search using so many resources?
- Why are recommendations delayed?

- Often these problems are due to interaction b/w services

For example

- Recomendation Engine assumes low DB latency
  → Makes many queries

- DB latency depends on # of concurrent queries
  → Latency increses as # of concurrent queries increase

- A recent search feature increases # of DB & Recomendation queries

⇒ - Search requires more resources
  - Recomendations are slow

NEED TOOLS AND METHODS TO REASON ACROSS ALL OF THESE SERVICES

- EACH OF WHICH MIGHT BE COMPRISED OF SEVERAL PROCESSES / SERVICES

DISTRIBUTED TRACING IS ABOUT COLLATING AND CORELATING INFORMATION ACROSS PROCESSES

REQUIREMENTS?

- Record time — response time
  Order
- Low overhead
- Always up / fault tolerant
- Log levels

WHY?

- USEFUL ACROSS DOMAINS
  - ↳ UNDERSTANDING BOTTLENECKS / PROBLEMS IN ML
  - → " " " IN DATA PROCESSING
  - → . . .
- WE CAN ACTUALLY GENERATE TRACES, USE & ANALYZE THEM
  - ↳ TASK FOR NEXT CLASS.
  - → DOESN'T REQUIRE ACCESS TO FANCY H/W, ETC.
  - → WE HAVE PRETTY REASONABLE VISIBILITY INTO WHAT PEOPLE DO IN PRACTICE.
- THERE CONTINUE TO BE MANY OPEN QUESTIONS & PROBLEMS THAT NEED TO BE SOLVED.

HOW?

- READ & LEARN FROM
  - → A RECENT BOOK FROM PEOPLE AT MICROSOFT, TWITTER, ETC. ON THE TOPIC
    - ↳ ACCESS ONLINE, FOR FREE, FROM NYU LIBRARY

→ PAPERS

→ DOCUMENTATION & CODE

- GET YOUR HANDS DIRTY WITH COLLECTING & ANALYZING TRACES

  → NO STENCIL CODE — REALLY WRITE SOMETHING THAT EXHIBITS PHENOMENON YOU THINK ARE INTERESTING.

  → THIS WEEK: GET STARTED WITH GENERATING TRACES FROM A SINGLE PROCES

  → EVERY FEW WEEKS SNAPSHOT EXERCISES & SUBMIT AS HW (3 × 10%)

  - GENERATING & COLLECTING TRACES
  - VISUALIZING & ANALYZING TRACES
  - USING TRACES TO DEBUG PROBLEMS

- FINAL PROJECT (25%)

  - INTERESTING WAYS TO COLLECT OR USE TRACES

- Extensions to existing trace [illegible]
- ...

- Exams (10+20)
  - ↳ Understanding of Class Material

- Participation
  - ↳ In Class (5%)
  - → Helping people online, posting tutorials or instructions, presenting tools/techniques in class (10%)

What Goes Into A Trace

Requirements

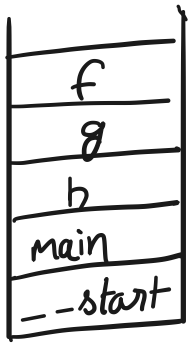
- Track Causality

What?

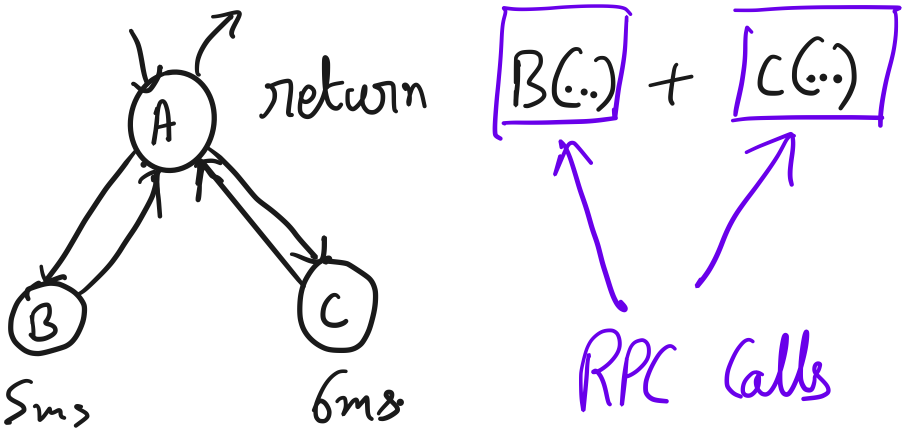
WHY?

- SUPPORT CONCURRENT CALLS

STACK TRACES FOR PROGRAMS



`bt' in gdb



EXECUTION TIME

EXECUTION TIME

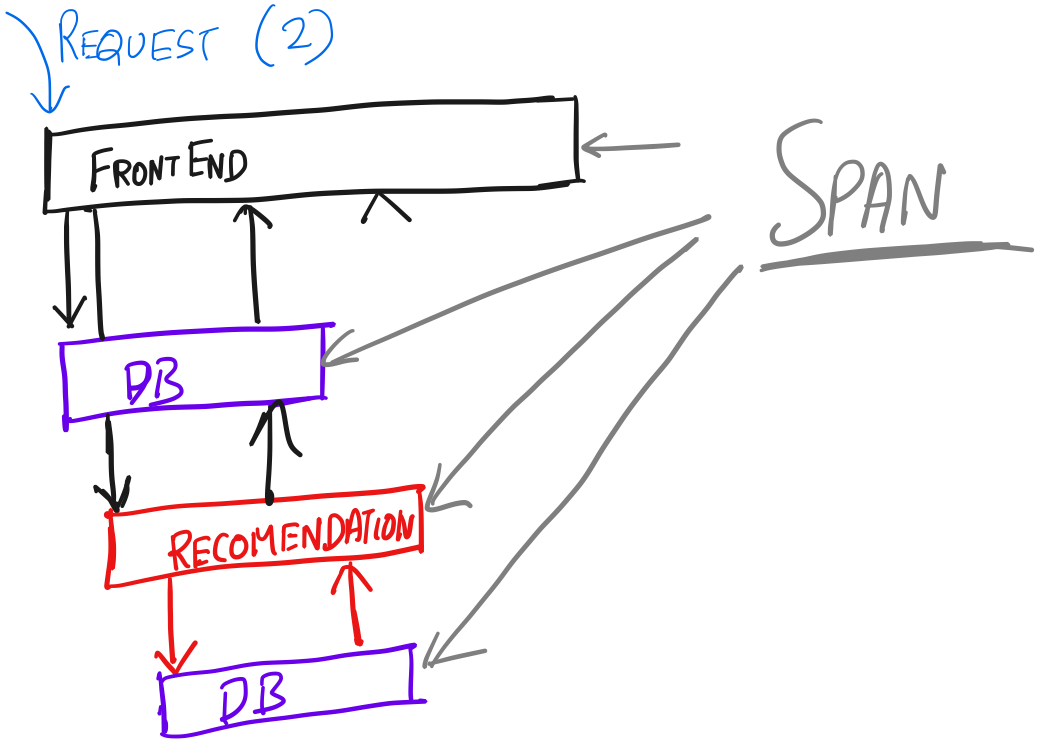
- TRACK TIME (& OTHER ATTRIBUTES)

THE STRUCTURE OF TRACES

- USE IDs (Request ID or Trace ID) to DISTINGUISH BETWEEN REQUESTS.

WHAT IS A REQUEST?

- Within a Request, Model Causality As A Tree.



- Representing This

Request Id, Span Id, Parent, Start Time, End

→ Globally unique

Unique to a request

Span ID on 

- Where Are These Stored?
- How Collected?
- Overheads