Lecture 5: Mystery Machine.

Administrivia

- Project proposals & groups

  ↳ Perhaps useful to send me a short description of what you were thinking.

  Two options

  ⓐ Post a note, only visible to instructor, on Campuswire [PREFERRED]

  ⓑ E-mail me to set up time to talk in-person [Will do it in 15-30 minute meetings next week]

- Poll on office hours.

- Class feedback?

localizing problems & using localization information

- Why?

  ↳ Again a use of trace data & infrastructure

    ↳ But less influenced by my tastes & preferences: $\exists$ big company doing the same

→ Presents two different approaches

OmegaGen – White box /intrusive

[But does it gather the same information]

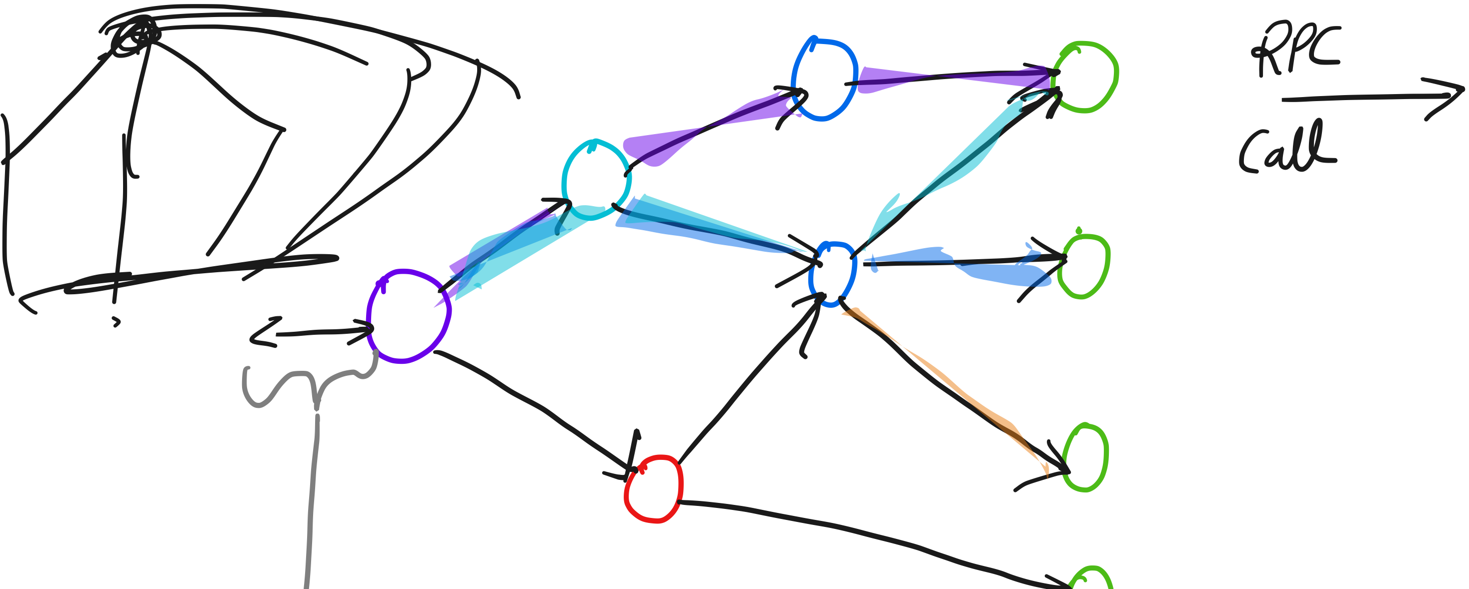
Mystery Machine – Black box

↳ Assumes no knowledge about

- What services communicate.
- Service dependencies

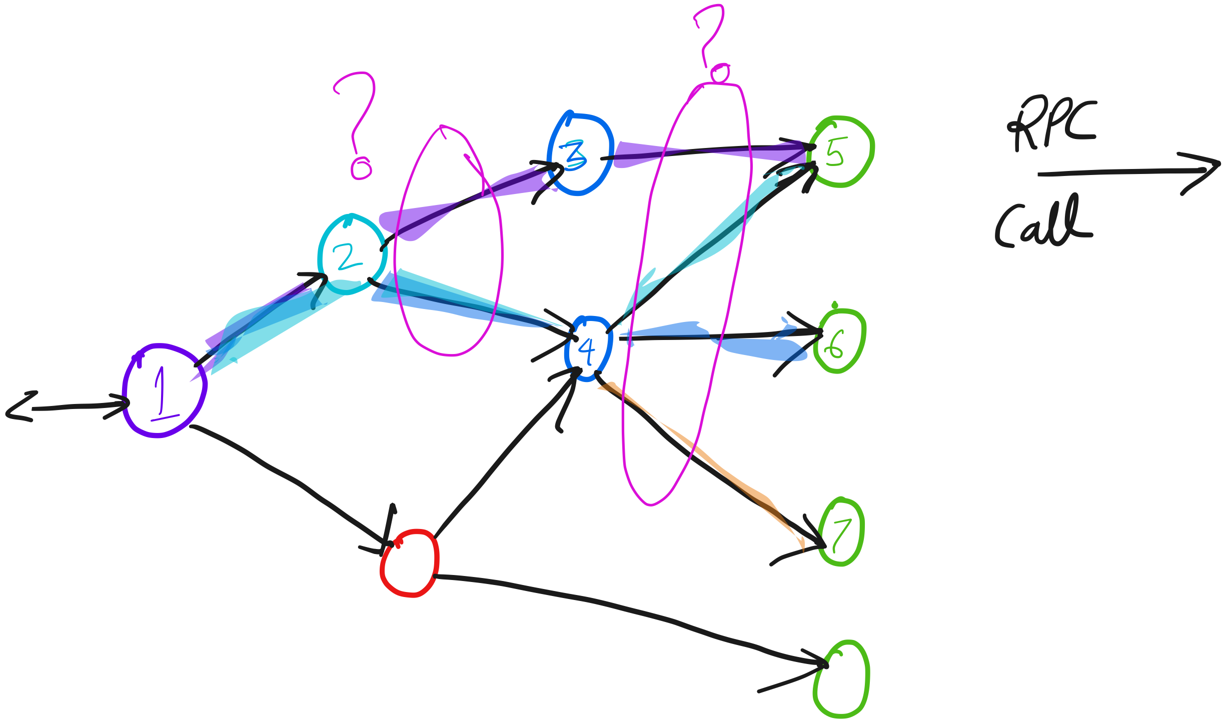
Focus in both cases is on finding PERFORMANCE BUGS.

CRITICAL PATHS & PERFORMANCE IN CONCURRENT SYSTEMS



↳ QUERY EXECUTION TIME :- Determined by Longest Path

What do we need to determine critical Path



① Knowledge about dependency relations

1→2→3→5
1→2→4 } Paths.

Core message: These dependency relations are often unknown.

Huh? Is this a Facebook only problem?

Why?

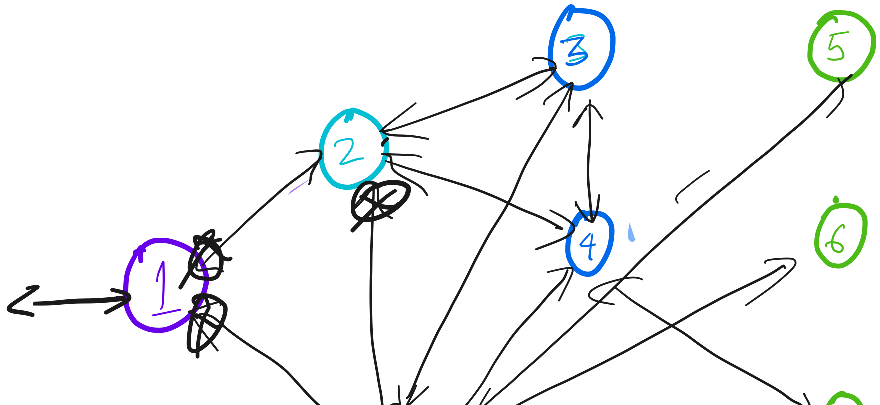
Approach: Mine dependency from traces

What do we need to determine critical Path?

① Knowledge about dependency relations

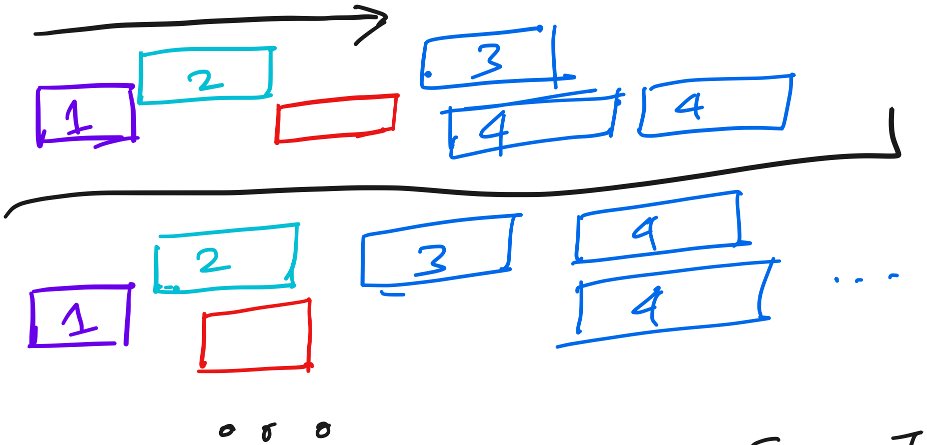
② Time spent on each component In a path.

Mining Dependency Relations H.B.



① Assume all possible relations

② Use trace events as counter examples



SOUNDNESS: WILL THIS METHOD EVER INCORRECTLY DECIDE TWO RELATED SERVICES ARE UNRELATED? (IN EXAMPLE ABOVE NO H.B.)

COMPLETENESS: WILL THIS METHOD EVER INCORRECTLY DECIDE TWO UNRELATED SERVICES ARE RELATED?

IMPACT ON TOOL: CRITICAL PATH?

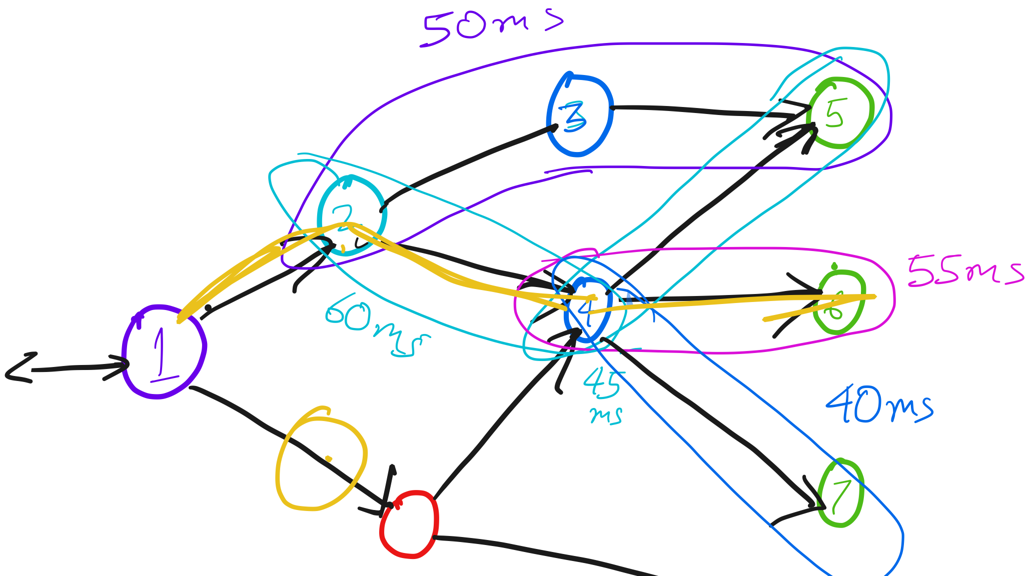
ASSUMPTIONS: WHY IS THIS A REASONABLE DESIGN IN PRACTICE?

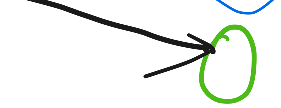
- NATURAL PERTURBATION

-

② IDENTIFYING CRITICAL PATH, AND BOTTLENECK

—





Is the critical path stable? Do all requests have the same critical path?

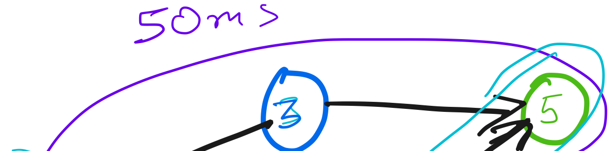
Why? Important to how we use critical path information.

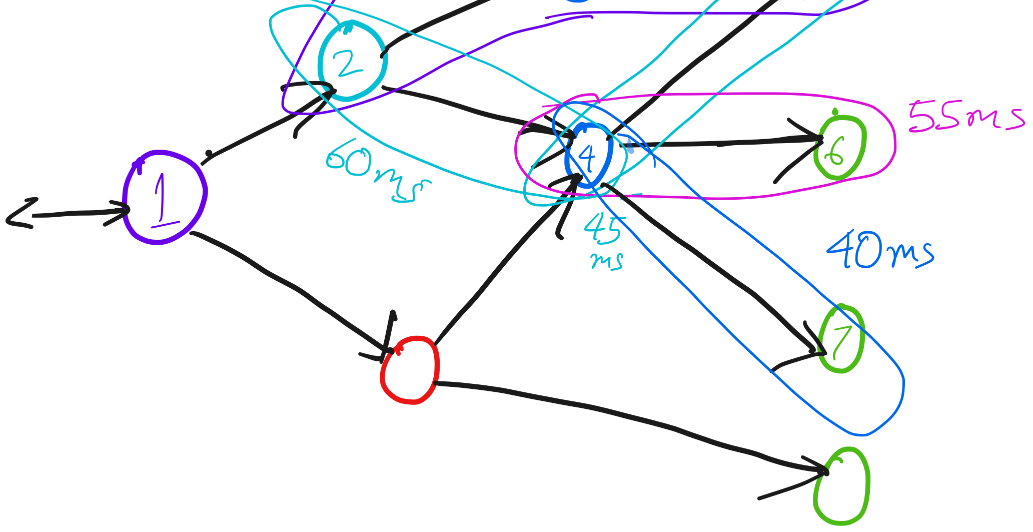
Using critical path information

- Targetting optimization ~~information~~ resources

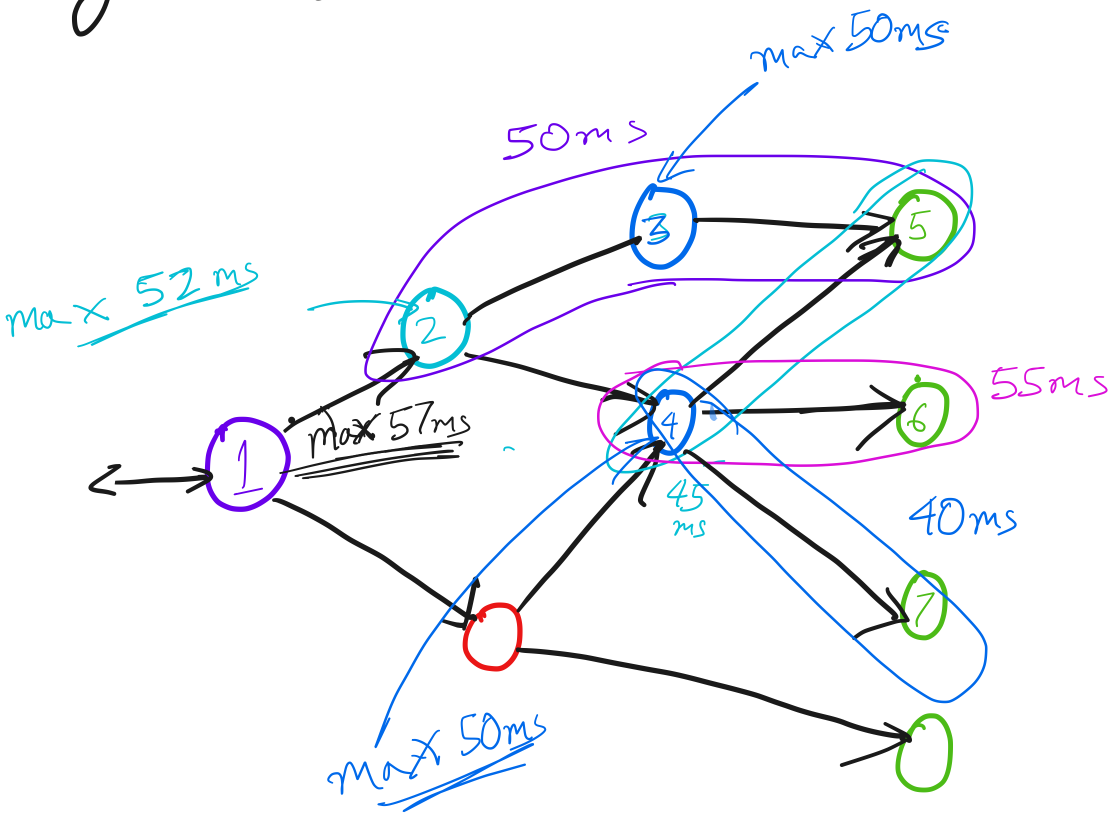
- Making resource provisioning decisions

  ↳ Slack





DQBarge: Using trace information in real-time



What should 4 do?

Q: When to give up on 4→6?

- 50-ε ms after 4 received request.

Pros | Cons

6 has a chance to respond	① wait ② Resource waste ·

- Before invoking the call

Pros	Cons
6 does not meltdown	6 no chence

- Others?

Pros	Cons

QUESTION: HOW DO WE DECIDE (A-PRIORI) WHETHER TO MAKE 4→6 REQUEST?

MAKE REQUEST

+ Likely have better quality result (How likely?)

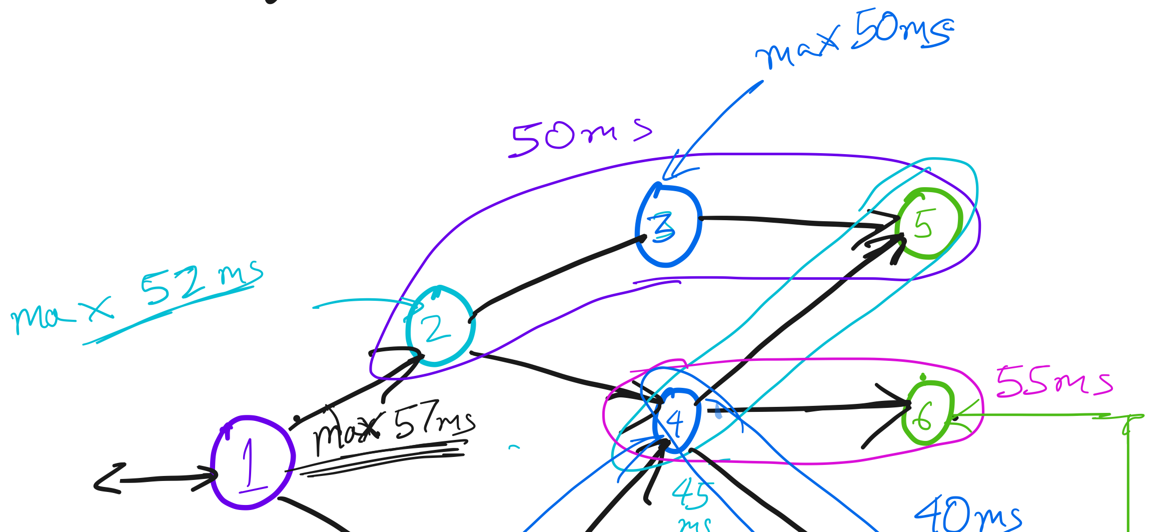
- Increase load on 6, perhaps w/o benefit to this request

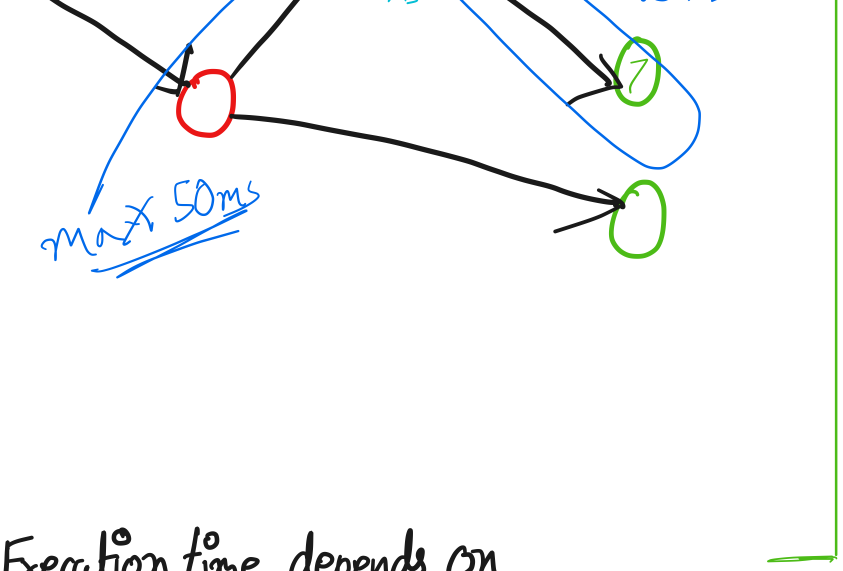
Do Not Make Request

- Guaranteed to have lower quality result

- Has no impact on 6.

Desirable: Make request only when success is likely. But how do we know?





Execution time depends on

- Load (# of outstanding requests)
- Background tasks (GC, rebalancing, etc.
- Request type
- Request arguments
- Request history
- Downstream Tasks

→ Depends on other services + likely varies gradually

→ Maybe unpredictible, but likely to be periodic

→ Requires semantic knowledge

Maybe best captured by trace exception caveats above.

Core argument: Request execution time roughly depends on

- Other requests in the system

- System health.

- Execution time when processing request at other processes

→ Proxy for request type & argument

When running, see a large enough variety of requests, workloads & system load conditions

↳ Use data to learn a model for system behavior M

→ M's inputs inclue

- Metrics at current service (e.g. 4)
- Metrics at upstream services (e.g. 2,1)
- Accesses, etc. at upstream services

- Output:- Yes or no on call + what to do if not making call

→ Challenge: How to collect metrics for the current request.

Sol^{n}: Do what Pivot Tracing Did.

OmegaGen

- How to improve accuracy for detecting perf bugs/ anomalies

+ Localize them at a finer granularity

w/o ~~many~~ any other requests.

CORE PROBLEM W/LOCALIZATION

PROGRAMMER DOESN'T A-PRIORI KNOW CRITICAL PATH/ CRITICAL OP

CRITICAL

- MYSTERY MACHINE: USE DATA TO FIND CRITICAL PATH/CRITICAL OP POST FACTO
- OMEGAGEN: GUESS POTENTIAL BOTTLENECK
  - DECIDE IF THEY ARE ACTUALLY A PROBLEM

① Potential bottlenecks

- I/O?

Why? What else could/should include.

② Decide if they are a problem

→ Monitor execution

- [illegible] operations

↳ Mimic & Reexecute operation

Why?

Fidelity?

Soundness?

Completeness?

Do these systems really do same/similar things?

Returning back to DQBarge

↳ How have others solved this problem?