Summarizing Everything

Poster Session

- Reminder: Next week in class
  - Do not print your poster
  - Just set up your laptop etc. to show it

  - Come up with how you would present what you did to someone in 5ish minutes

Suggestions

Goals: Communicate two things
- Question or problem you
- What was interesting about
  - Your approach
  - Your findings
  - ...

**Audience:** Your classmates

Assume knowledge of things from class and general CS education

Do Not Assume knowledge of the proposal, do you posted.

> These are just suggestions. Sometimes it might be better off ignoring them &
> going a different way.

---

**Final Exam**

- Yes, we have one

May 11: 6-7:50pm; This room
- Covers everything
- Similar in structure to midterm
- Also open book, etc.

Where we are

01 - Building & deploying applications made up of many services is common now.

Some debate about how many services,

Many
  → More flexibility?
  → Easier to get more people to work on app
  → ...

Fewer
  → Easier to improve performance, reason about correctness
But using >1 per-application has been norm forever.
WS ↔ DB
WS ↔ File system

Adding more services is likely necessary now:
- Caches/edge to get lower latency.
  ✔ Reduce network cost
- Split functionalities depending on h/w on which things execute
  ⊳ Accelerators: GPU, ...

02. We expect more from the applications we use.
  → Rely on Internet services for nearly all civic functions
  → Have several services where latency matters
    - Video conferencing
    - Video streaming
Need to build more reliable applications

- Fewer failures
- More consistent performance
- ...

How?
- Build more reliable services/components
- Glue them more carefully?

03° Don’t have a lot of tools to build reliable systems

- New languages & runtimes
- Better libraries
- Static verification
Therefore, rely on post-facto analysis

Standard approaches to post-facto analysis
- Logs
- Crash dumps

Hard to know if problem on performance or service led to problem

Traces → Add information to logs to enable correlating information.

This is the main takeaway

Logging in a distributed application
Log from each service

Causal information to link things together.

Everything else is just about balancing other concerns.

- How often should one log? Efficiency
- What to log? What type of analysis would one want

- Failures:
- Latency:
- Resource Efficiency:
- Throughput:
+ Scaling

+ What algorithm to use: (PGO)

+ ...

Nearly all of these have analogues in the single machine case.

Aside: What about crash dumps
Other questions we considered

- How to achieve stable performance.

+ Generally

\[
\text{service time} \times \text{arrival rate} = \text{processing time}
\]

- If known can find desired processing time. Allocate resource for...
Problem:
- Arrival rate varies. Might change by a lot
- Processing time is not constant, even w/fixed resources.

But we want consistent performance

How?

1. Adjust system parameters dynamically
2. Drop requests to limit max arrival rate

Control loops, Rl, ...

② What to observe?
① What to change?
© By how much?

- How to test distributed applications

  Problem: Large input space

  - Request types
    - Other requests in the system
    - State of service

  Unlikely to be able to cover all possible scenarios.

  Ways to cover scenarios that are likely to matter?

Aside: Why post-facto debugging is necessary?
Aside: The effectiveness of testing.

Where do we go from here
- We didn’t really spend a lot of time on debugging.

Where do you think we are any of this