Learning to Do things

Announcements

- We only have a class left
  - No readings for next week
  - Talk about putting together all the things we talked about this semester

- Poster session
  - Expectations.
Writeup - Due 05/08

Missed a question/request in Campaswine.

- Briefly talk about fault tolerance/disaster recovery.

- Q: How do distributed systems get 0 downtime?

  Mostly they don't. Goal is many % of uptime but reality intervenes. Even 99.9% is hard.
  
  AWS service discounts [Region]

  | 99 - 99.9% | 10% |
  | 95 - 99%   | 30% |
  | ≤95        | 100%|

Instance

  | 99 - 99.5% | 10% |
  | 95 - 99%   | 30% |
  | ≤95        | 100%|

Q: What does 0 downtime mean?

  Depends. Generally goal is availability.

  But
- want to avoid data loss but fine if not have immediate access
  - Depends on system

Q: Disaster recovery?

Again, depends what you mean.
For network ops: alternate in 0(24) hours

Q: How?

→ see last lecture.

Today: Overview on how to use data

→ Over the last several weeks talked about several algorithms that
  - identify some important characteristic
  - use measured values/history/...

→ DAGauge/AutoPilot/Firm/C3/...

  Control loops based on observations about
  performance/utilization/...
Failure injection based on observations about
job structure.

Prof/OMTA

Summarization based on likely anomalies

- In all cases: design based on some empirical or analytical observation about
  - Program structure & how failures impact them
  - Performance across dependencies

- ...

- But how do we know that we picked the right factors/combined them in the correct way?

As we don't.

At the end of the day, there is just too much possible data we can consider. All of it likely impacts performance/failure/...
Examples

- Heat affects
  - Performance (CPU throttling)
  - Likelihood of some types of failures
  - Silent data corruption

- CPU batch/Identity affects performance/failure
  - Mercarrial cores (Gloshschild et al.)

- Runtime used

-...

We just choose what seems important & design algorithms around it.

- The promise of ML for systems
Run into a few issues

- Learn from data. More features, more data
- Time & resources for training
- Cost of executing the resulting function

Often matters the most for frequently executed algorithms

Often solved by carefully engineering what features are provided & how they are represented

A few steps forward from before

- Consider more features
- Don't have to assign importance
- ...

- This keeps evolving. Used a few places in practice
- Power management at Google

- What does this have to do with traces
  > Where else does the training data come from?
  > But for real
  > Do we need to change them?
  > How?

- Papers this class
  > Picked two I like
  > Not necessarily the coolest/newest!
  
  [Someone, somewhere is looking at LM for scheduling]

What both share: RL

Though used somewhat differently.
How to model agent?

1. Decima: Neural network
   \[ \text{GNN} \rightarrow \text{will come back to this} \]

2. Polyjuice: Lookup table

\[ \text{Trade-off} > \]

<table>
<thead>
<tr>
<th>Time to Pick</th>
<th>DECIMA</th>
<th>POLYJUICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
<td>Inference</td>
<td>Faster (GT)</td>
</tr>
<tr>
<td>Categorical</td>
<td>( T, S, C )</td>
<td>( \text{ISP}(D, A #) )</td>
</tr>
</tbody>
</table>
Why is this the “correct” trade-off in this case?

Generalization/Effect on training data

PolyJuice

Actions

- When to read/wait
- What to read (committed/uncommitted/…)
- When to make writes available
- Validate reads?
Input to function

→ Txn being executed

Do not consider

- Other transactions in the system
- Load etc.

Why?

- Let us look at assumptions when training

Utility is Maximize trust

Training assumptions' workload used for training
→ current workload

- # of Transactions
- When they arrive
- ...

+ 

. Policy

⇒ What transactions are running concurrently

But generalizability?

Decima

- Actions

- Where to execute
  - Think back to Distributed Resource Management on C3°. Equivalent to what replica to pick

- Parallelism
  - How much work to have each
executor do:

- Input ?

Task + Job + Other Jobs

Problem: How to represent this?

\[
f: x, y, z \rightarrow \ldots
\]

What if I have more than 3 inputs?

What if I have an unknown number of inputs?

Diagram:

1. \( E(A) \rightarrow E(B) \rightarrow E(C) \rightarrow E(D) \)

Some fixed size summary
of D. What does it contain?

But A's performance impacts C's performance which impacts D's.

\[ E_3(A) = E(A) + G_1(E_3(B) + E_3(C)) \]

\[ E_3(B) + G_2(E_3(C)) \]

Why?

Huh?

Bottom line: \( E_3(A) \ldots E_3(D) \) contain some information about other downstream tasks.

\[ G_1(\Sigma E_3(A)) \rightarrow y_5 \]
Want to schedule A?

$$\text{Sched}(E_s(A), Y_s, \odot Z)$$

Assumptions we have made so far

- Go in reverse DAG order

- The need for G
Training assumptions:
- Jobs seen in training are "similar" to scheduled

Lessons/Promise of this approach.