Framework for workflows with task dependencies on HPC

Scott Zimmerman, MPH
Ahern Research Group
Overview

• Motivation from research
• Software solution
• Use cases
• Current state, future developments
Research Problem

e.g. Compare the finite-sample performance (Bias, MSE) of 5 estimation procedures for the average treatment effect (ATE) of A on Y given:

User knowledge:
- Degree of confounding $\approx 0.2$
- Most extreme propensity scores $\approx 0.95$

Causal Structure
Research Problem: Example Results

Red Line: True ATE in data-generating system

Blue line: Finite population ATE
Calculating Bias to Convergence

• What is the finite-sample bias of a statistical estimator based on our simulations? (X=truth, Xn = estimate)

$$\lim_{n \to \infty} \sum \frac{(X_n - X)}{n}$$

• Single iteration may take a long time
  – (e.g. ensemble method w/machine learning)
Example: Convergence

n too small: Value doesn’t converge

n too large: Waste computational resources and time

Don’t know best # of iterations at start.
Task C depends on task B
Task A queues tasks B and C
Task A creates dependency of C on B

P_A may queue more S and P_A jobs if a condition is met, and add the new P_A as a dependency for the CS_A job
Request types

- **Assign**: Create a new row in tasks table
- **Queue**: Change status to “queued”
- **Finish**: Change status to “finished”
- **Cancel**: Change status to “cancelled” (propagate through descendants)
- **Receive**: Pull next task whose parents are all “finished”, or return null
- **Add dependency**: Add row (parent, child) to dependencies table
Transition to Savio

Savio

Node

Node

Node

Stored Procedures

MySQL Database

AWS VPC

Tasks

Dependencies

Web Server

Web Server
Multiple clients

Connection container
- Connection (Port 51717)
- Connection (Port 51718)
- Connection (Port 51719)
- Connection (Port 51720)

Event Queue

Scheduler

Node
- Client
- Client

Node
- Client
- Messages

Protocol Buffer
Client-side view of a task

0) Client calls “receive” every 5 seconds
1) “receive” returns path to instructions
2) Read instructions
3) Run task
4) Read in other data according to instructions
5) Assign new task IDs, add dependencies, and queue tasks
6) Output new data
7) call “finished” (or “cancel”)

Scheduler Client

Main Process

Child Process

Storage (e.g. Scratch, DB or S3)
Compilation

• C++ client-server pair of programs
• Compiled with cmake
• Dependencies need to be installed first:
  – Boost
    • [http://www.boost.org/](http://www.boost.org/)
  – Google Protocol Buffers
    • (de)serialization of language-specific objects
    • [https://developers.google.com/protocol-buffers/](https://developers.google.com/protocol-buffers/)
Use Cases

• Useful for:
  – Complex dependency structures
  – Unknown task dependency structure at offset
    • Dynamic scheduling
    • Can also use for static scheduling if you don’t want to set up the queries yourself
    • Less useful:
      – “Process 10,000 images using a given script”
  – Distributed task processing
Future

• Additional features planned:
  – Ability to remove dependencies
  – Logging thread
  – Ability to query job status
  – Grouping of tasks by job

• I invite recommendations, guidance, or contributions from anyone interested
Acknowledgements

Thank you to everyone who has contributed to this project!

Ahern Research
Jennifer Ahern
Ellie Colson Matthay
Kara Rudolph
Dana Goin
Kriszta Farkis
Chi Zhang
Jess Galin
Laura Balzer
Erin Ledell
Aaron Culich
Krishna Muriki
UC Berkeley cloud computing working group

Funding for this research was provided in part by NICHD/NIH Office of the Director DP2HD080350, the Robert Wood Johnson Health and Society Scholars Program, and the University of California, Berkeley Committee on Research.