Project Ideas

Beginner Difficulty (Difficulty Bonus = 1.0)

1. Analysing the behavior of different retry algorithms under different workloads and load spikes. Requires the analysis of the following retry algorithms (note that some may already be implemented in Blueprint; others you will need to implement yourself as Blueprint plugins)
    a. No retries (baseline)
    b. Fixed Instantaneous Retries
    c. Fixed Delay Retries
    d. Exponential Backoff Retries Without Jitter
    e. Exponential Backoff Retries With Jitter
    f. Token Bucket Retries (Pseudocode available here: https://docs.aws.amazon.com/sdkref/latest/guide/feature-retry-behavior.html)
2. Reproducing the Laser of Death emergent misbehavior with Blueprint.
    a. More information about the behavior here: https://www.usenix.org/publications/loginonline/why-health-check-sidewalk
    b. Requires implementing a load balancer that maintains state based on health checks
        i. Subsequent integration with Blueprint so that this is easily testable on existing applications
3. Implementing a runtime causal inference service for root cause analysis that can be integrated with Blueprint.
    a. Runtime Causal Inference Service should periodically get traces from Jaeger, find the anomalous traces (>99th percentile latency), and then do root cause analysis using causal inference.
    b. Requires automatically figuring out the causal graph from the application workflow.
    c. Requires doing offline training for the causal model
    d. Service should be implemented in Python and be available as a Docker container (which can then be integrated with Blueprint)
    e. More Information on how to do Root Cause Analysis with Python: https://www.pywhy.org/dowhy/v0.8/example_notebooks/rca_microservice_architecture.html
4. Implementing a critical path analysis service that can be integrated with Blueprint.
    a. Critical Path Analysis Service should periodically get traces from Jaeger, calculate the critical path for each trace, and store statistics about the common paths.
    b. Implement a causal path diff that compares the critical path of an anomalous trace (e.g. >99th percentile latency) with some critical paths of the same type of request.
    c. More information on how to calculate critical paths
        i. Paper: https://www.usenix.org/system/files/atc22-zhang-zhizhou.pdf
        ii. Code: https://github.com/uber-research/CRISP/tree/main

5. Integrating FIRM's fault injector with Blueprint as a plugin for injecting different types of faults into the system.
    a. FIRM Paper: https://www.usenix.org/system/files/osdi20-qiu.pdf
    b. FIRM Code: https://gitlab.engr.illinois.edu/DEPEND/firm/-/tree/master/anomaly-injector
    c. Use this to show the noisy neighbor effect and how that can potentially lead to a metastability failure (Capacity Degradation Trigger + Workload Amplification)
6. Implement and analyze prioritized load shedding and compare its behavior to circuit breakers in dealing with metastability failures
    a. Netflix Prioritized Load-Shedding: https://netflixtechblog.com/keeping-netflix-reliable-using-prioritized-load-shedding-6cc827b02f94
    b. Netflix Service Level Load Shedding: https://netflixtechblog.com/enhancing-netflix-reliability-with-service-level-prioritized-load-shedding-e735e6ce8f7d
7. Show the impact and benefits of autoscaling with Kubernetes
    a. Requires integrating Kubernetes with Blueprint
    b. Must be done in a 3-person team.

Intermediate Difficulty (Difficulty Bonus > 1.0, < 2.0)

1. Integrate Hindsight with Blueprint
    a. Steps required: https://docs.google.com/document/d/1HO8aKuZvu1w000ZgWmK1mxurANYwrkQiVIrioNH7MmY/edit?usp=sharing)
    b. Hindsight Paper: https://www.usenix.org/system/files/nsdi23-zhang-lei.pdf
2. Integrate Metafor (metastability analysis library) with Blueprint.
    a. This will basically require converting the Blueprint workflows into the DSL by Metafor. Each API maps into Work. Each Service has a Server and a Client configured.
    b. Library: https://github.com/mpi-sws-rse/metafor/tree/main
    c. Paper: https://sigops.org/s/conferences/hotos/2025/papers/hotos25-106.pdf
3. Integrate Blueprint with TLA+.
    a. Possibility #1: Generate Blueprint workflows into TLA+ programs.
    b. Possibility #2: Generate Blueprint workflows from TLA+ programs (or Modular PlusCal) to get integration with PGo
        i. PGo Paper: https://www.cs.ubc.ca/~bestchai/papers/asplos23-pgo.pdf
        ii. PGo Code: https://github.com/DistCompiler/pgo
4. Integrating Blueprint with OpperTune to allow for parameter tuning.
    a. OpperTune Paper: https://www.usenix.org/system/files/nsdi24-somashekar.pdf
    b. OpperTune Code: https://github.com/microsoft/OPPerTune/tree/main
5. Integrating Blueprint with MuCache

a. MuCache paper: https://angelhof.github.io/files/papers/mucache-2024-nsdi.pdf
b. MuCache code: https://github.com/eniac/mucache