Harjot Mangat

EECS 268 – Datacenter-Scale computing

Project Report - Datacenter Throughput Simulation Using SimPy

Abstract

This project was inspired by the tiering architecture of modern datacenters. The goal was to develop a simulation environment that can approximate the throughput of a datacenter based on the amount of hardware resources that the datacenter has. This project uses SimPy to build the datacenter environment, which is evaluated on several configurations of hardware resources. The results show that this simulation can help show an approximate throughput of a datacenter system and help to identify any bottlenecks that hurt this throughput.

Motivation

The motivation behind building this simulation was the concept of "back-of-the-envelope" math and microbenchmarks that we discussed in module 1 of the class. The purpose of these tools is to help identify performance optimization opportunities and discover bottlenecks that hurt utilization. This simulation focuses on the tiered architecture of modern datacenters and aims to help identify any bottlenecks that might exist in the hardware configuration.

SimPy

This simulation is built using the SimPy library in Python. This library provides the ability to run simulations of discrete events step-by-step. The main features that are used in this simulation include environments, events, and resources. SimPy environments are where the simulation takes place and can help control the duration of the simulation and allow the creation and execution of events. Events in SimPy are the series of actions that happen in the simulation, and they can be classified as completed, currently running, or upcoming. SimPy resources help represent a limited capacity item that would need to be shared among the different processes running in the simulation. In summary, a SimPy simulation consists of several processes existing in an environment running events and sharing resources.

System Design

This specific datacenter simulation consists of one process type that is defined as packets. These packets represent the jobs/packets that would arrive at the datacenter from the clients across the

internet. The packets interact with the environment built with SimPy. There are three resources provided to the packets in this environment: network switches, compute nodes, and storage. These resources are used to represent the hardware resources available in modern datacenters.

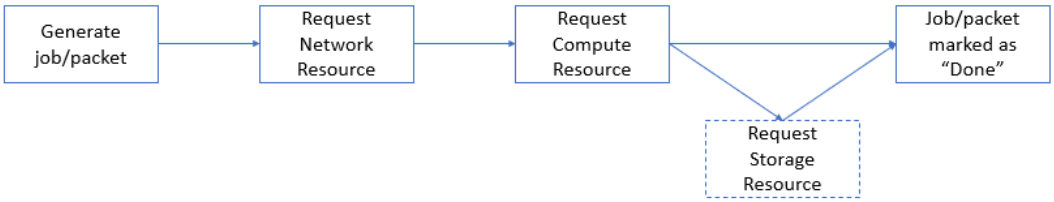


Figure 1: System Overview of the datacenter simulation

The flow of the simulation is shown in *figure 1*. Each box represents an event that occurs within the datacenter simulation. The simulation starts with a process (packet) being generated and beginning its journey through the datacenter. The process first sends a request for the first resource, which is network switches. In my implementation, each request will trigger an event to hold the process, representing time that is used by the hardware resource to do work. Once the network resource has released the packet process, the packet sends a resource request to the compute resource. This is repeated for the storage resource with the caveat that the storage resource request is a random 50/50 chance. Once the packet process has passed across all the resources, it is marked as done.

This entire simulation is meant to represent the flow of a packet/job travelling through a datacenter. The use of resources allows us to approximate the time spent by the packet as it is routed through the network switches, processed by a CPU core, and potentially needing to access some form of database structure for information. The parameters of each resource can be adjusted to approximate the average time spent at any tier in a real datacenter through micro-benchmarking. By adjusting the number of resources, we can scale the size of the datacenter to match any configuration we want to examine.

Evaluation

The evaluation for this simulation is done with several configurations of our simulated datacenter. The baseline is to have one of each resource, or 1/1/1 (compute, network, storage). Other configurations that were evaluated were 5/5/5, 10/10/10, and 1/10/100. The setup for the simulation is to generate a packet every 10 steps of the environment and to run the simulation for 1000 steps. The

primary metric for this evaluation is throughput, which is measured in the number of packets/jobs that can be completed in the 1000 steps.

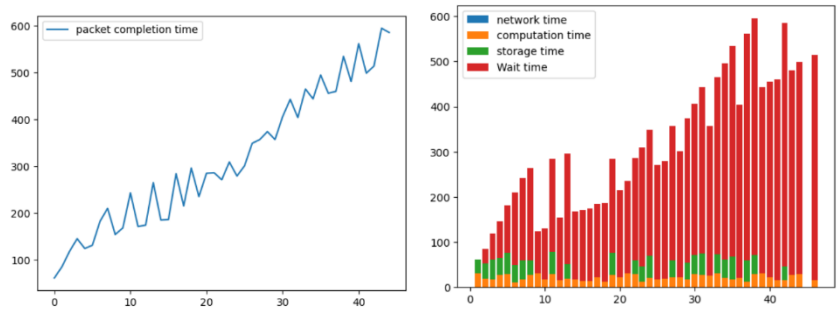


Figure 2: 1/1/1 configuration evaluation. Packet completion time graph(left), and trace of each packet broken down by time spent at each resource(right)

Results

The results of our 1/1/1 baseline configuration can be seen in *figure 2*. The main point of this configuration is to show an example with a severe bottleneck. Having only one of each resource is not enough to handle the number of packets being generated. From the packet completion time graph, we can see that the completion time continues to rise as the packets begin to queue for their resource requests. The stacked bar chart on the right of *figure 2* shows that most of the time spent by the packets was used on waiting for their turn to request resources. The throughput of this configuration was 45 packets completed in 1000 steps, which is roughly half of the number of packets generated. The results for 5/5/5 can be seen in *figure 3*.

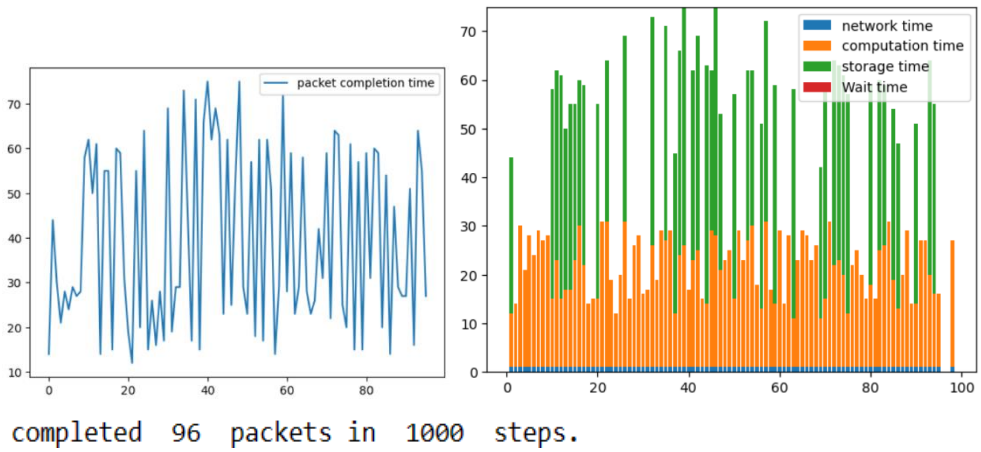
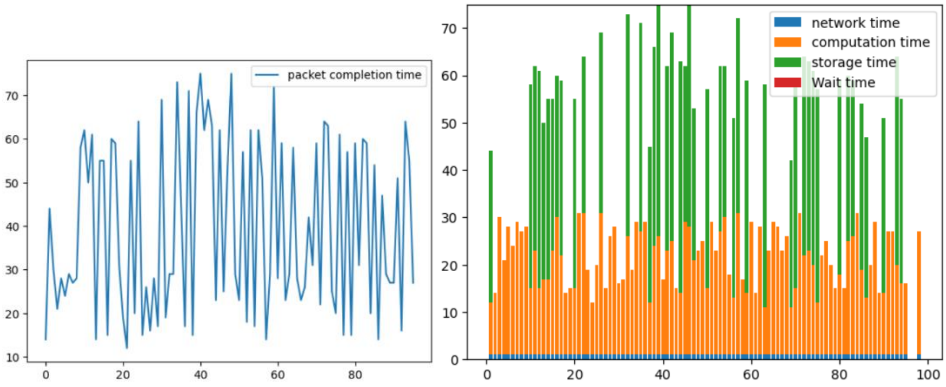


Figure 3: 5/5/5 configuration evaluation. Packet completion time graph(left), and trace of each packet broken down by time spend on each resource(right)

The results of the 5/5/5 configuration show an example where there are enough of each resource to handle the number of packets being generated. From the packet completion time graph, we see that the completion time of any packet is maximally about 85 steps, which is the highest regular amount possible in this environment, with no waiting. The stacked bar chart shows that many of the packets had no wait time, and all the time was spent on their resource requests. The overall throughput of 96 packets in 1000 steps is the maximum throughput of this environment, since some packets will be generated too late in the environment to be processed in time, such as packets generated at step 980, 990, and 1000.




```
completed  96  packets in  1000  steps.
```

Figure 4: 10/10/10 configuration evaluation. Packet completion time graph(left), and trace of each packet broken down by time spent on each resource(right)

The results of the 10/10/10 configuration (*figure 4*) show the same results as the 5/5/5 configuration. This is an example of overprovisioned resources. Since the 5/5/5 configuration was already able to handle the number of packets for this environment, additional resources were not able to provide any additional throughput improvements. In a real datacenter, this would be an underutilization of resources as fewer resources could provide the same result.

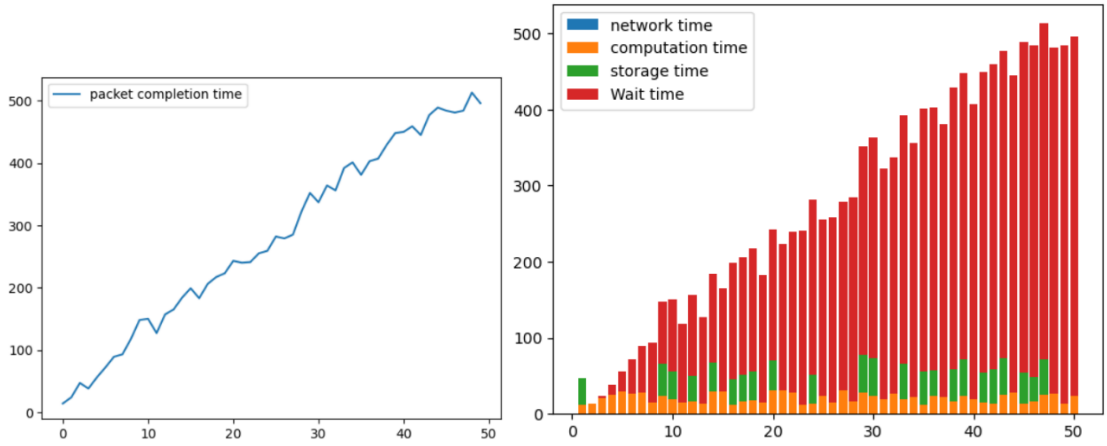


Figure 5: 1/10/100 configuration evaluation. Packet completion time graph(left), and trace of each packet broken down by time spent on each resource(right)

The final evaluation was done with the 1/10/100 configuration, the results can be seen in *figure 5*. This configuration shows a result similar to the results of our baseline 1/1/1 configuration. We can see a rising packet completion time as all the packets begin to queue for the singe compute resource in this configuration. This evaluation was done to show that this simulation can be helpful in identifying which specific resources could be causing bottlenecks.

Conclusion

This simulation was designed to approximate the multi-tiered architecture of modern datacenters. The system design is focused on simulating the flow of a packet/job travelling across the main systems of a datacenter, including network routing, computations being run on servers, and any storage requests that the packet/job might require. The system also has tracing capabilities to help identify which resources are causing wait times. This can be helpful for identifying bottlenecks based on insufficient resources. The results of the evaluation show that the system can approximate the throughput of a datacenter based on the configuration of resources that the specific datacenter has.