Project Presentation

Layered Queueing Network Modeling of A Mobile app System

Presented by:

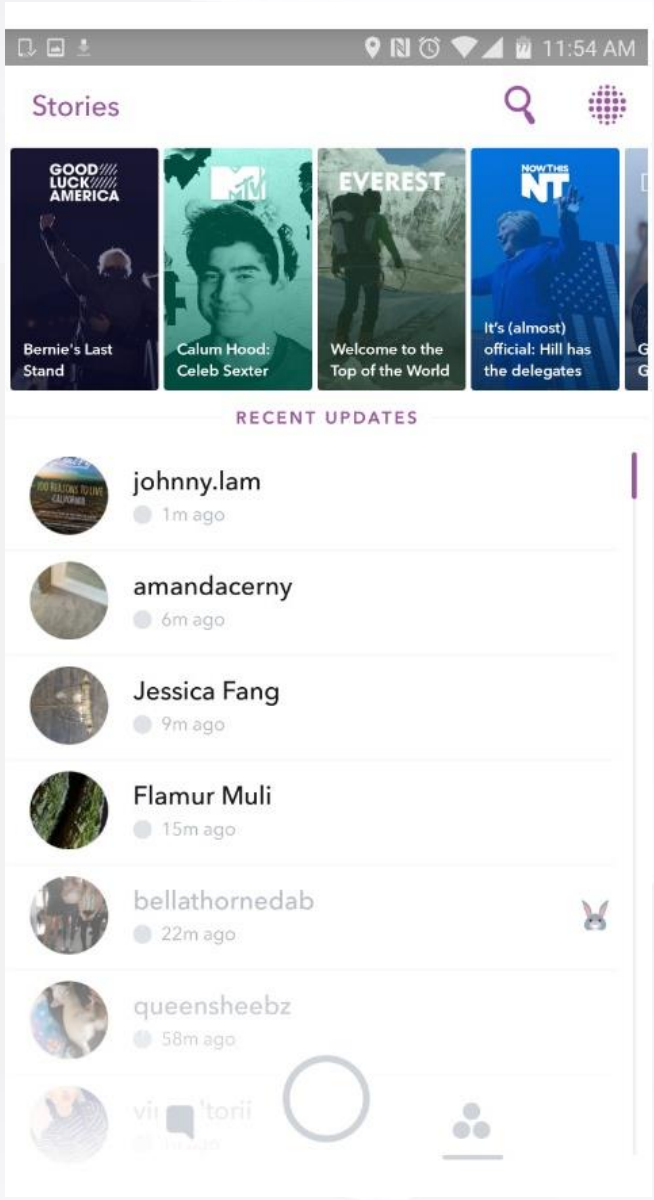
Alghamdi, Hamzah (7590886)

Chen, Mo (101062544)

00 Introduction

A Mobile Application

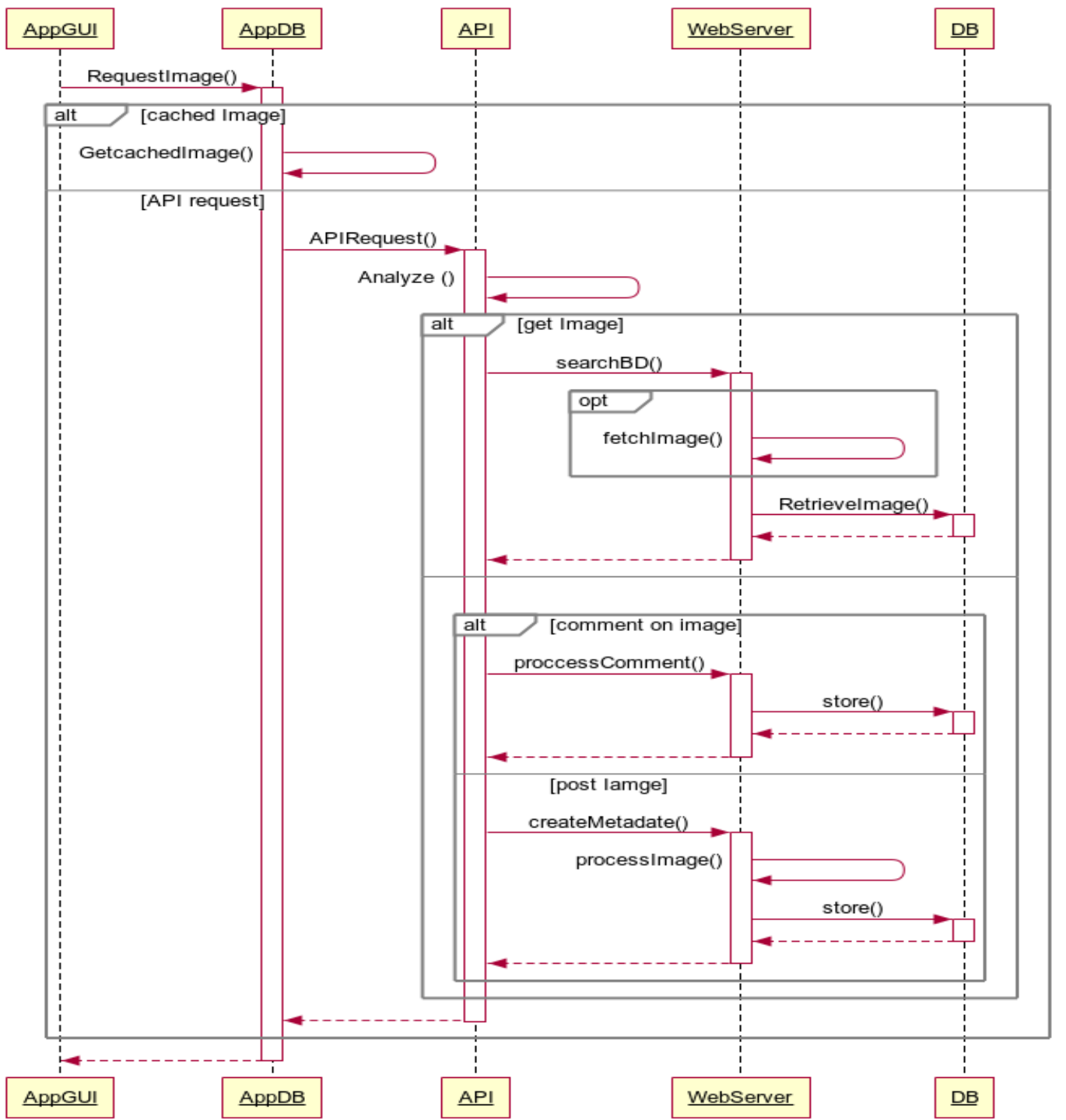
- A lot of IT companies provide a mobile application for their platform.
- Instagram and snapchat are examples of those mobile apps.
- Most of mobile apps use APIs (Application programming interface) for communications.



01 Scenario

A Mobile Application

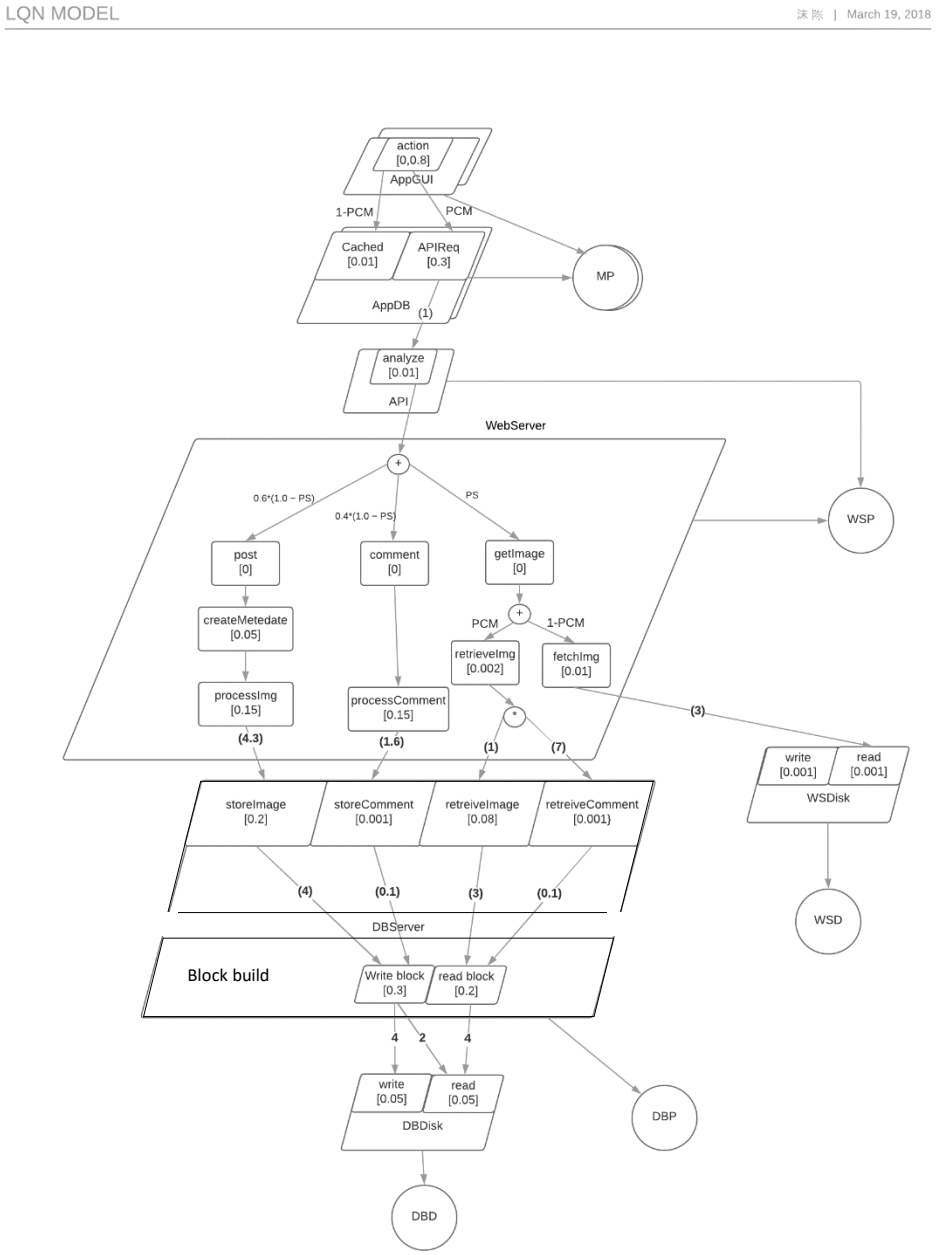
- Components, shown as Sequence diagram with annotation.
- AppGUI and AppDB represents a population of users
- Three types of API requests from users:
  1. Request of the Image
  2. Comment on the image
  3. Post an image



02 The LQN Model

Overview

- The App itself.
- The App database
- API interface
- The WebServer
- The Database



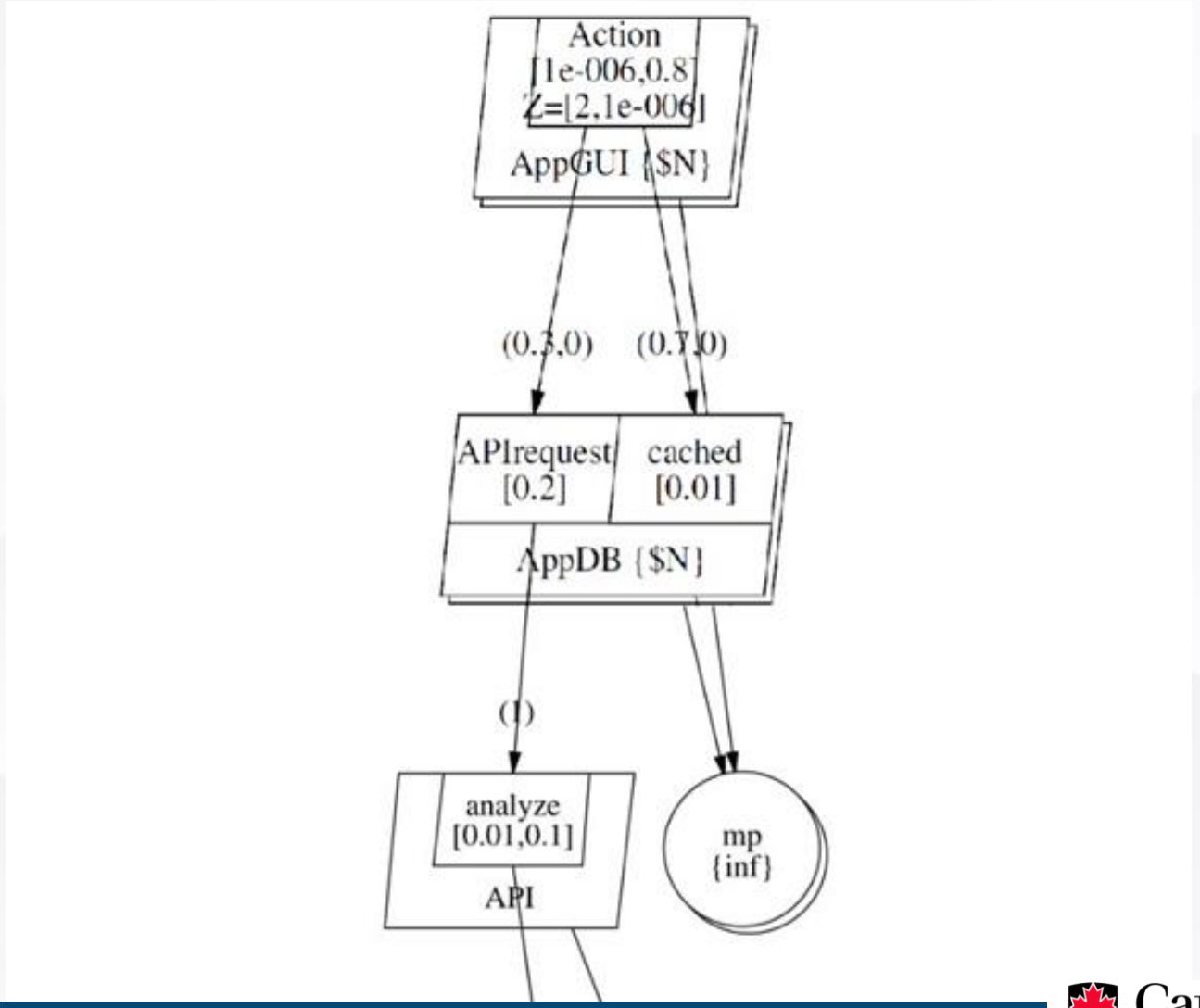
02 The LQN Model

AppGUI

- Action : entry that start this model
- It represents the user himself

AppDB

- Cached: requested Image is cached in the mobile app database
- APIRequest: if image is not cached or the action is other than getIamge()



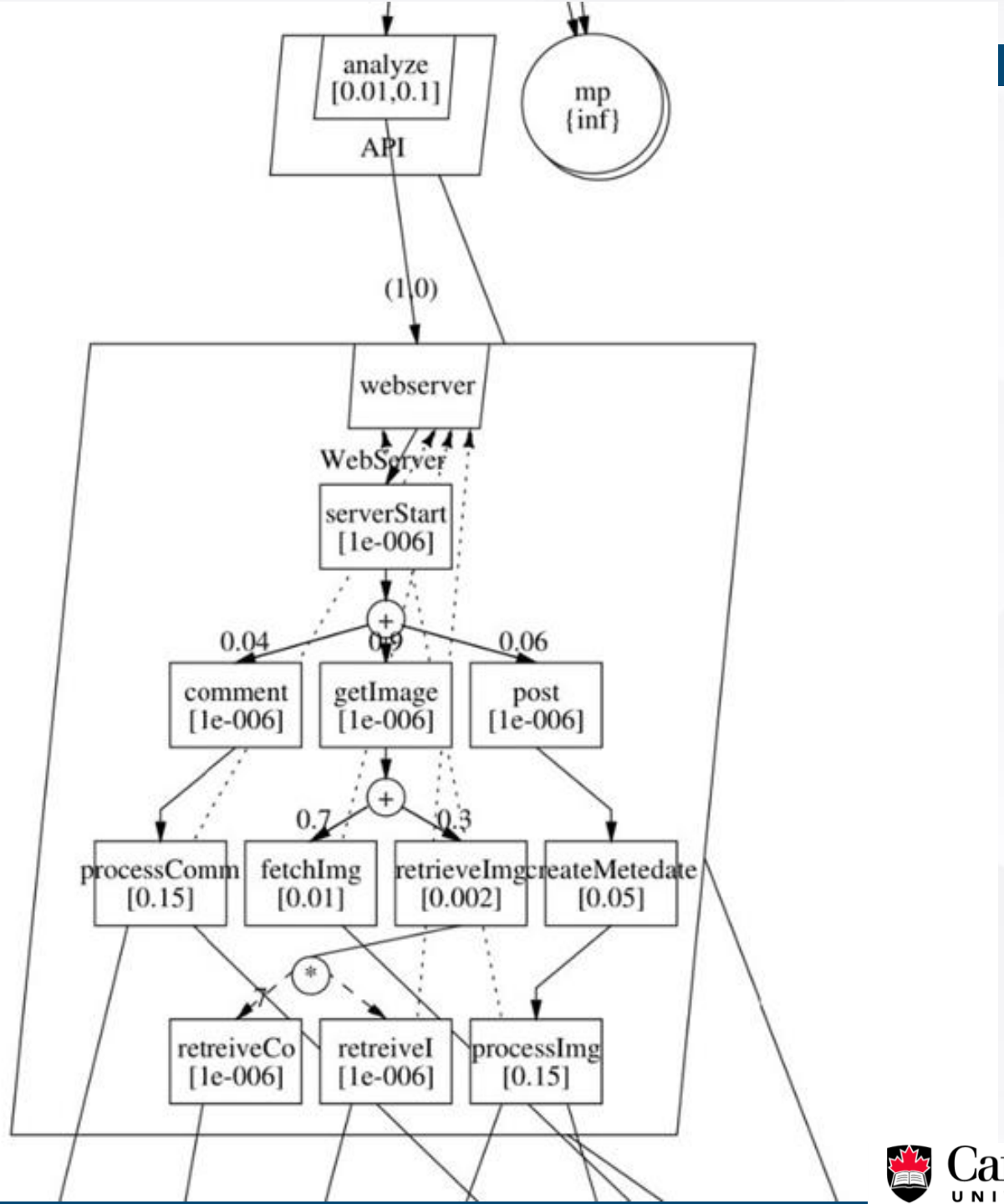
02 The LQN Model

API

- Analyze: serve as the communications gate for the webserver where all traffic goes through it

WebServer

- ServerStart: receive requests from the API and re-route the requests per API analyses.
- We have three routes for the request

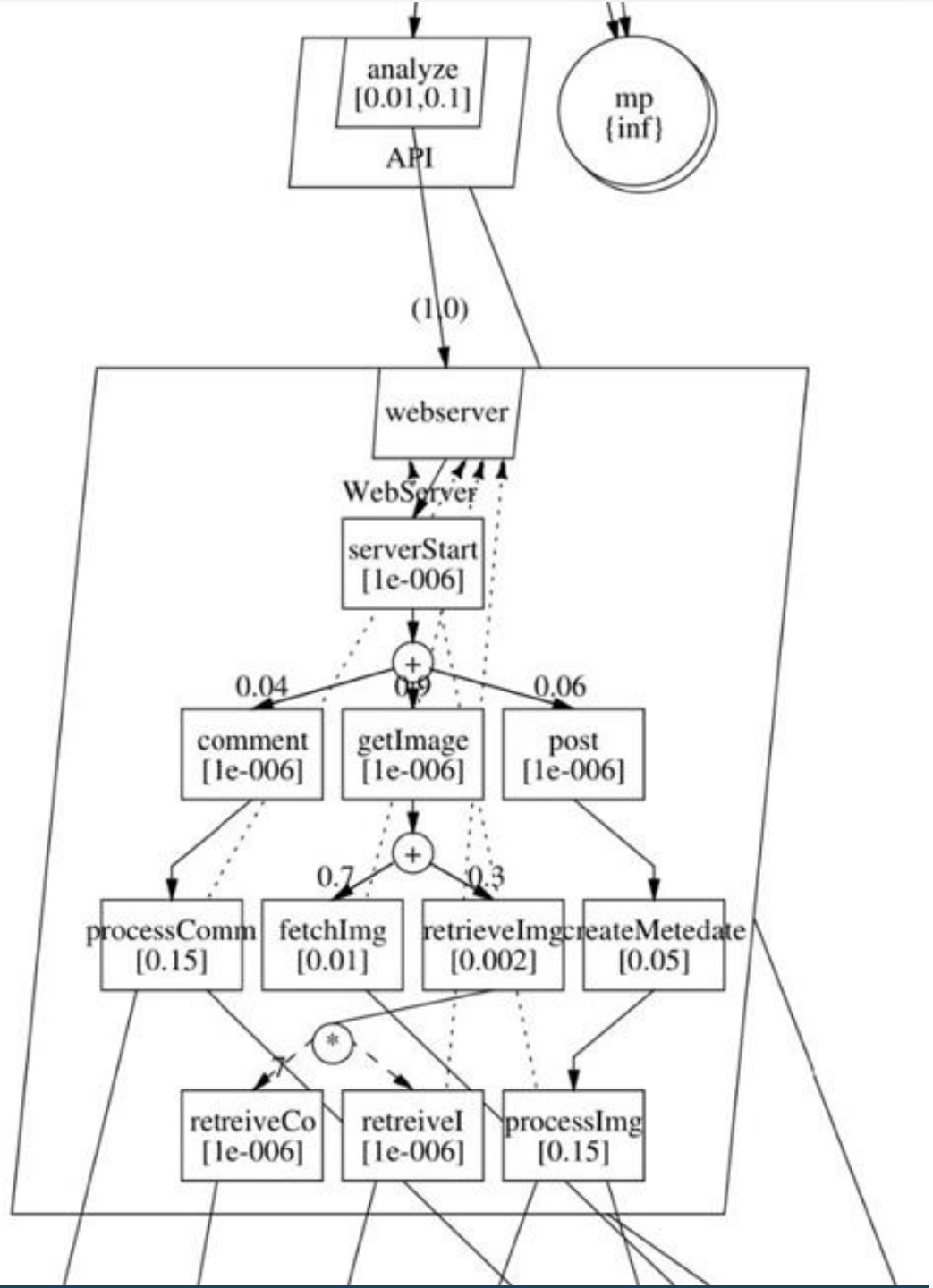


Carleton UNIVERSITY

02 The LQN Model

First type: request of Image:

- getImage: pseudo entry for the fork .
- fetchImg: fetch the image from the web server Disk -> diskread.
- retreveImage: Retrieve image from DB, possibly with embedded comments -> retreiveCo + retreiveI (both pseudo entries for the fork) which will call the database server.

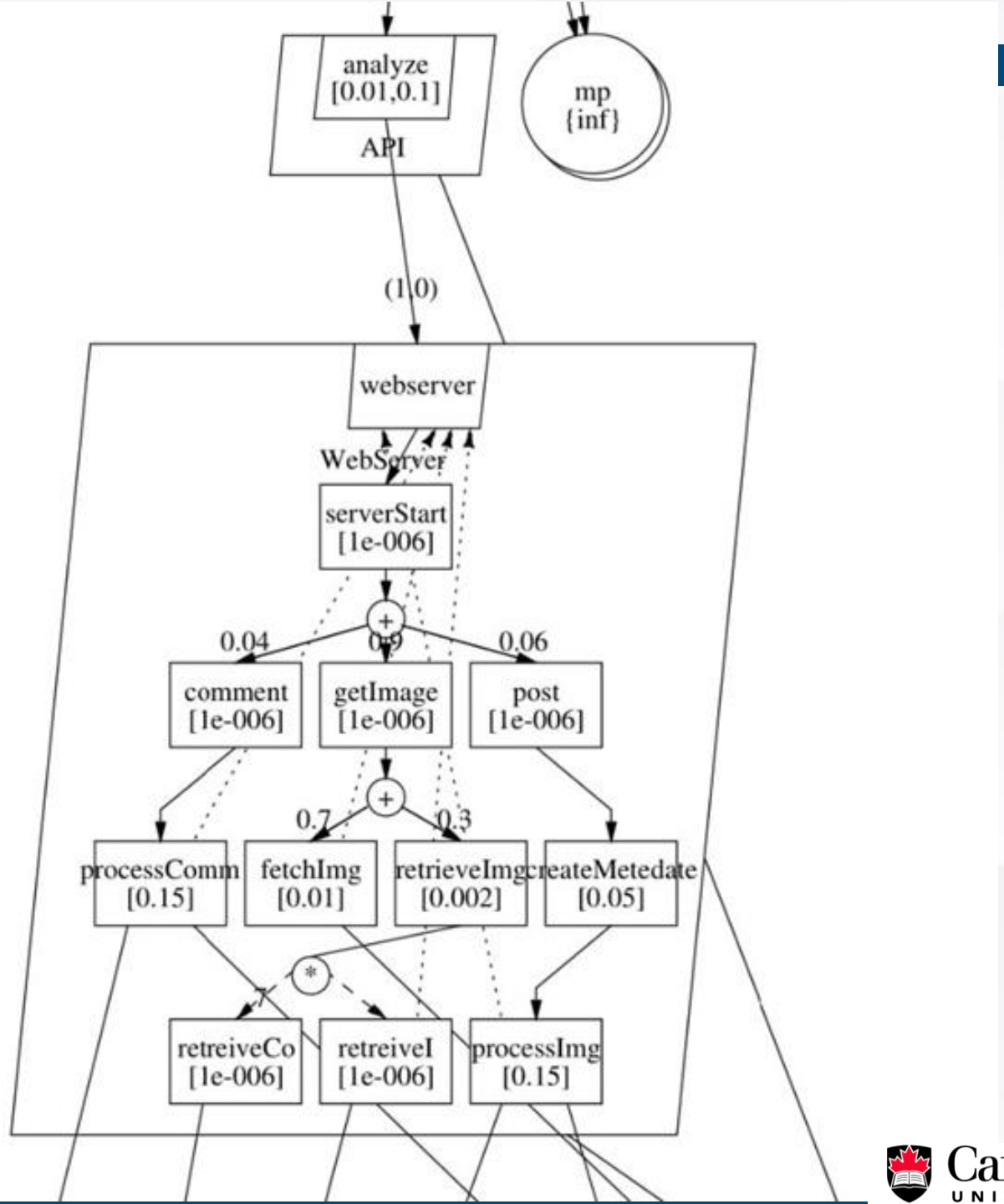


Carleton UNIVERSITY

02 The LQN Model

Second type: comment on image:

- **Comment: gets the data of a comment that is going to be added to image.**
- **processComm: do all the work needed to add the comment to a post. store it in the data base and webserver disk.**

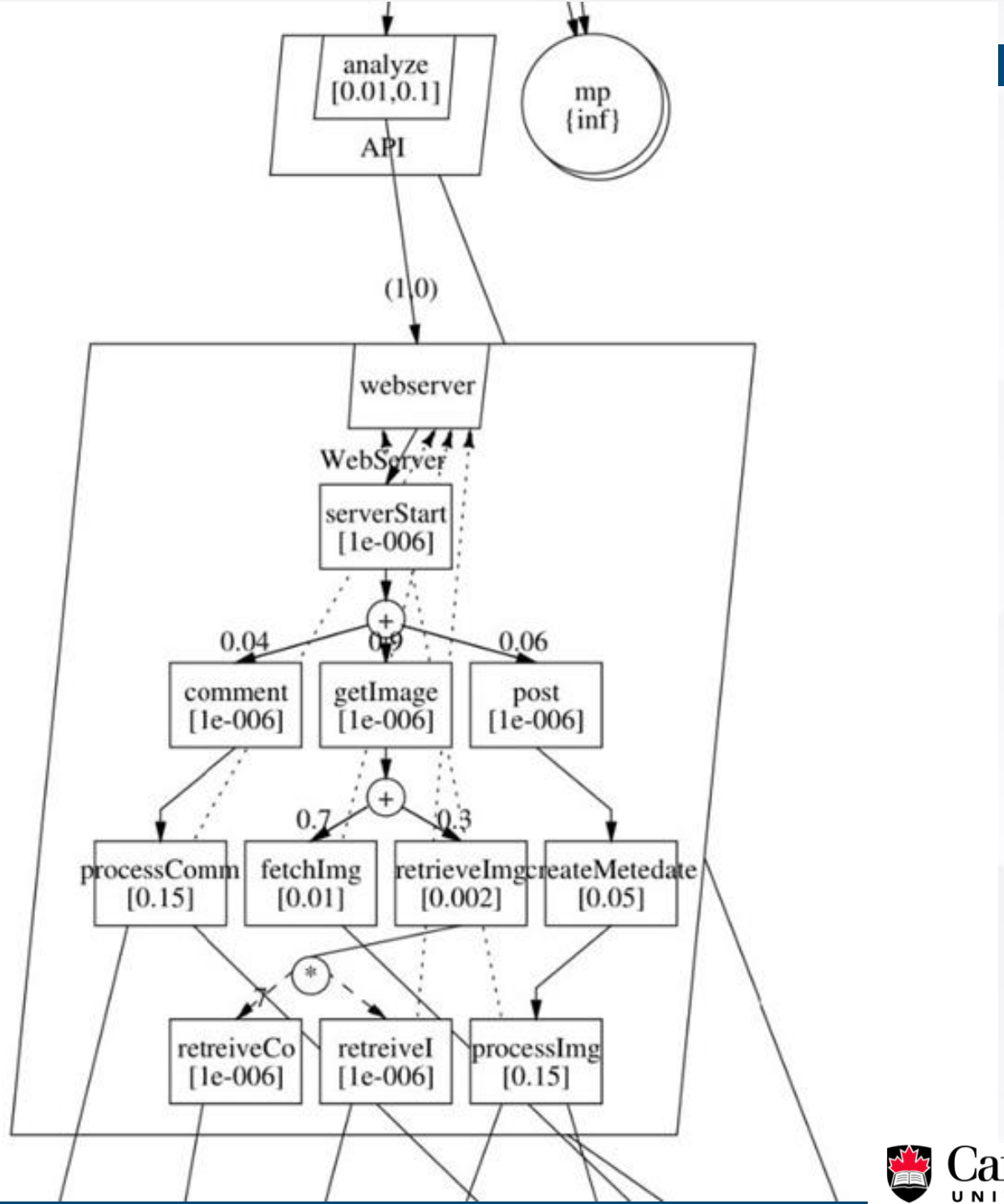


Carleton UNIVERSITY

02 The LQN Model

Third type: post an image:

- CreateMetadata :Construct Image metadata.
- processImg : process Iamge data. store it in the data base and webserver disk.



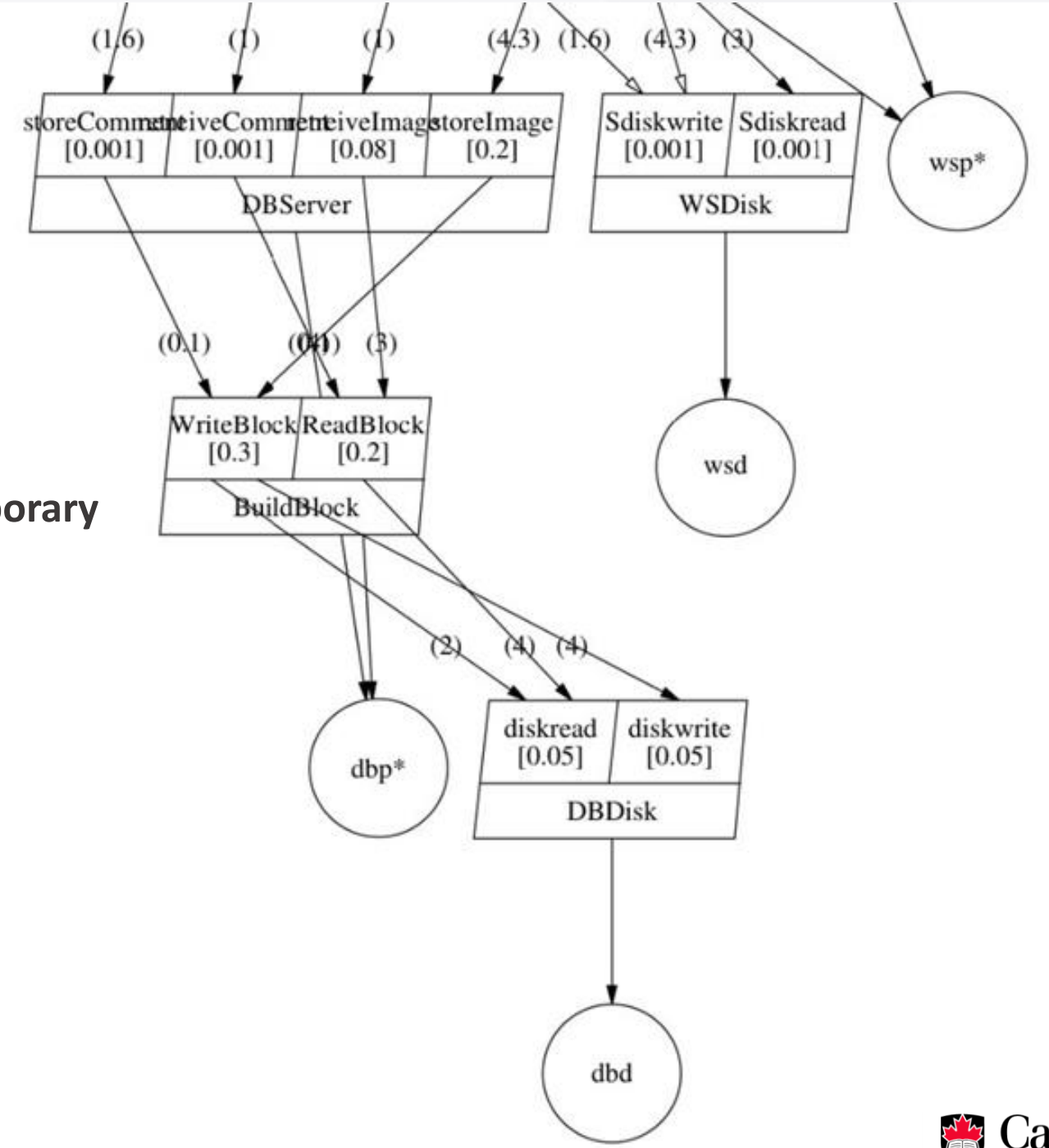
02 The LQN Model

WSDisk

- SDiskread: get the image from the web server disk with all of its metadata and comments.
- SDiskwrite(forwords only): save a temporary copy of new images and comments.

DBServer

- StoreComment + StoreImage: store the data into the database disk.
- retrieveImage + retrieveComment: read data from the database disk.



Carleton UNIVERSITY

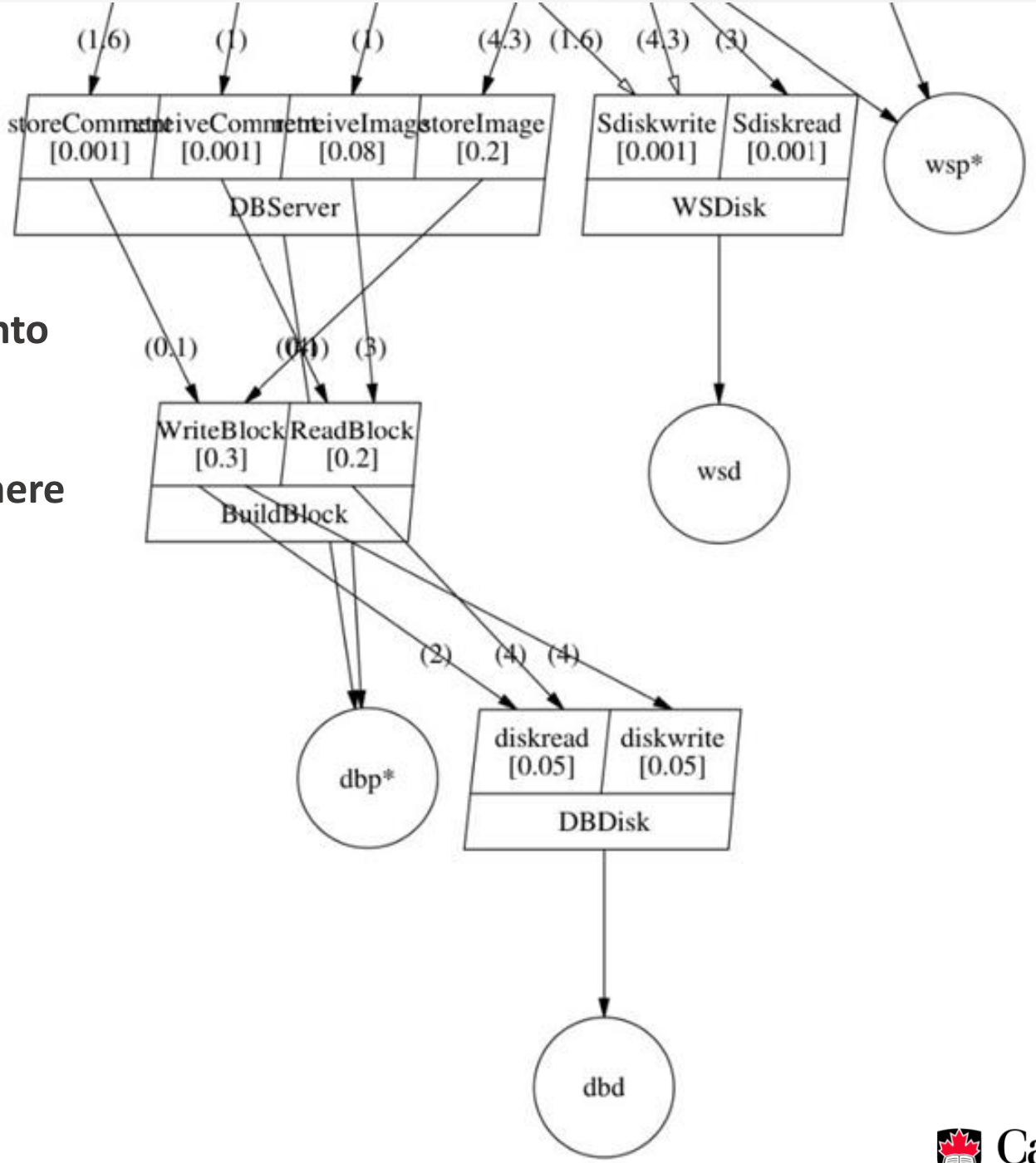
02 The LQN Model

BuildBlock

- WriteBlock: process data to be written into blocks to be stored into Database disk.
- ReadBlock: gets data from the blocks where it is lactated.

DBDisk

- Diskread : disk read operation for the database disk.
- Diskwrite: disk write operation for the database disk.



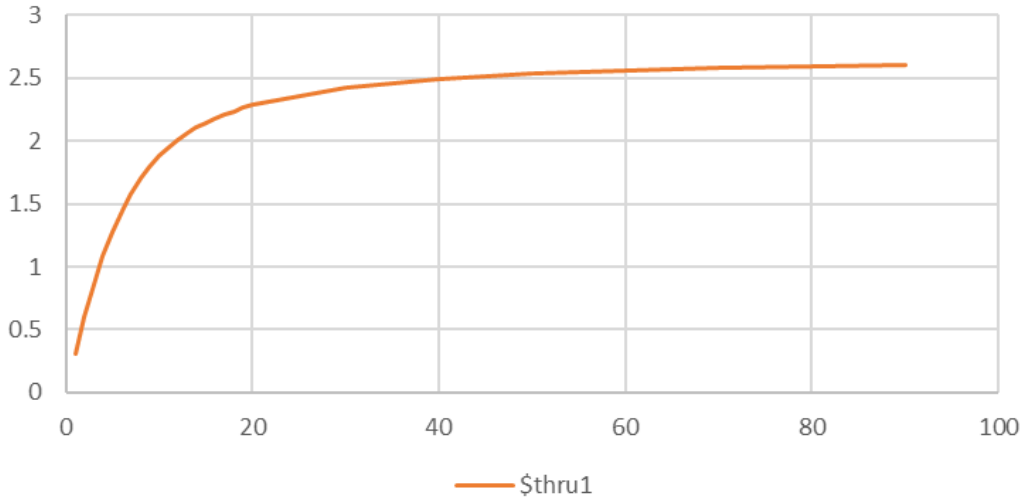
03 Results – Base Case

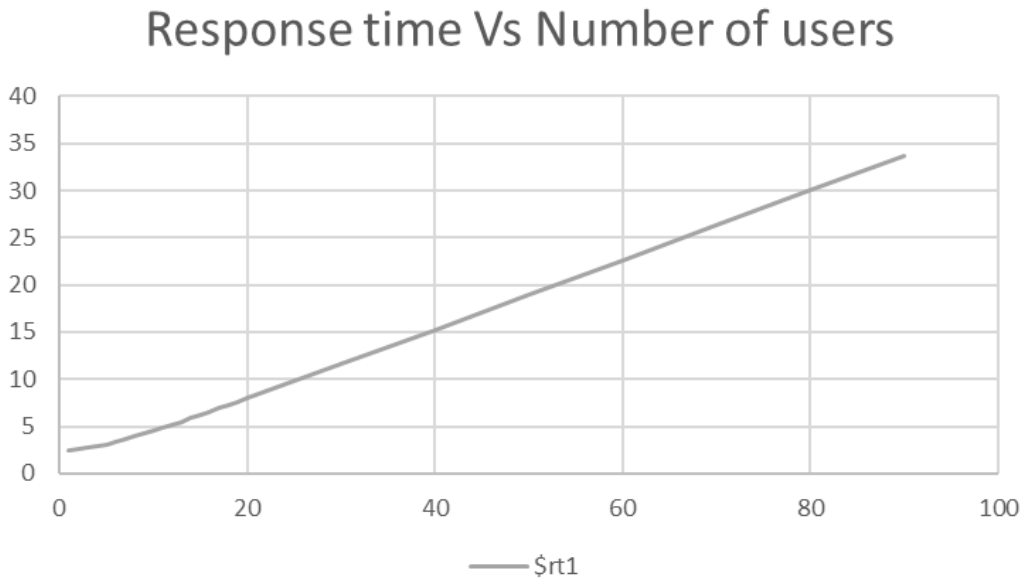
Setup of the model

- PS = probability of getting Image request = 0.9
- PCM = prob. of cache miss= 0.3

Throughput in the system

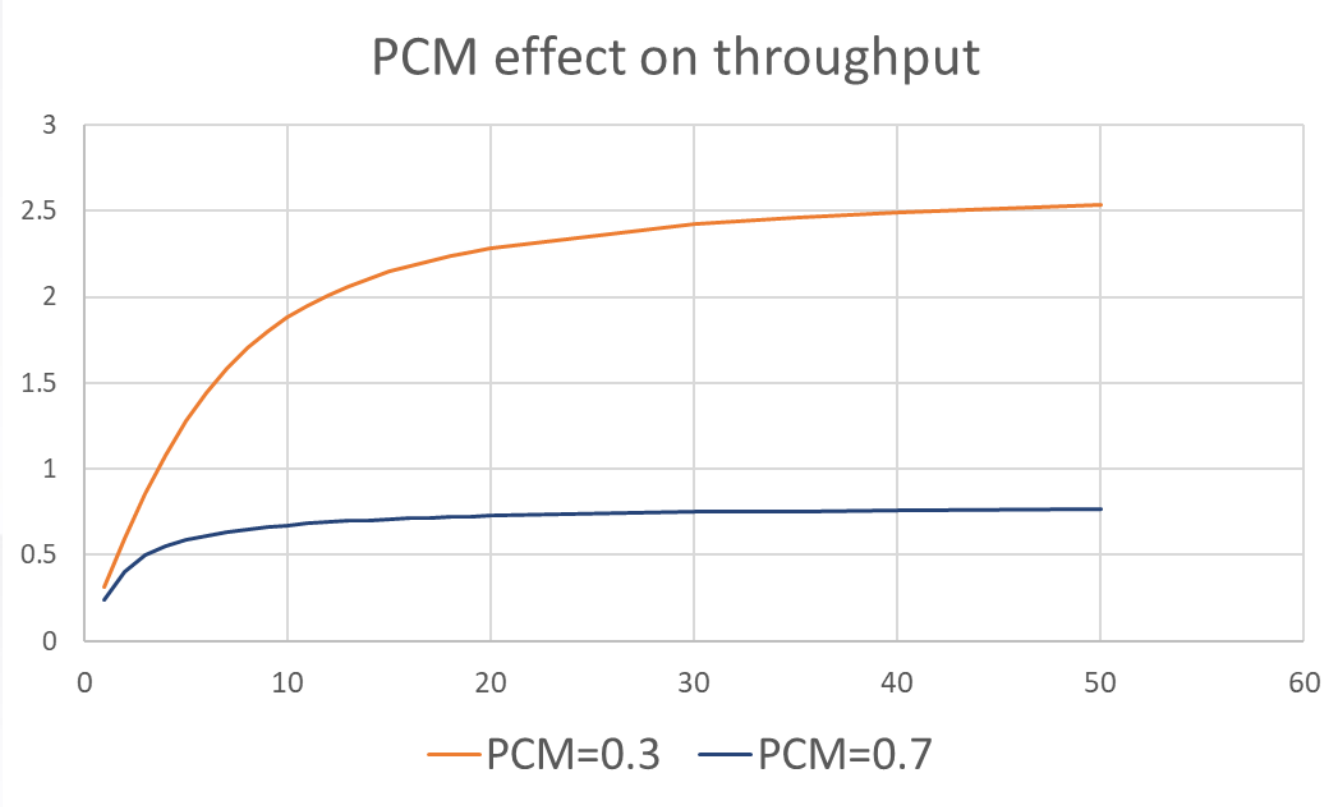
- leveling off at NU = 50
- Its saturation has been reached





04 Sensitivity to PCM

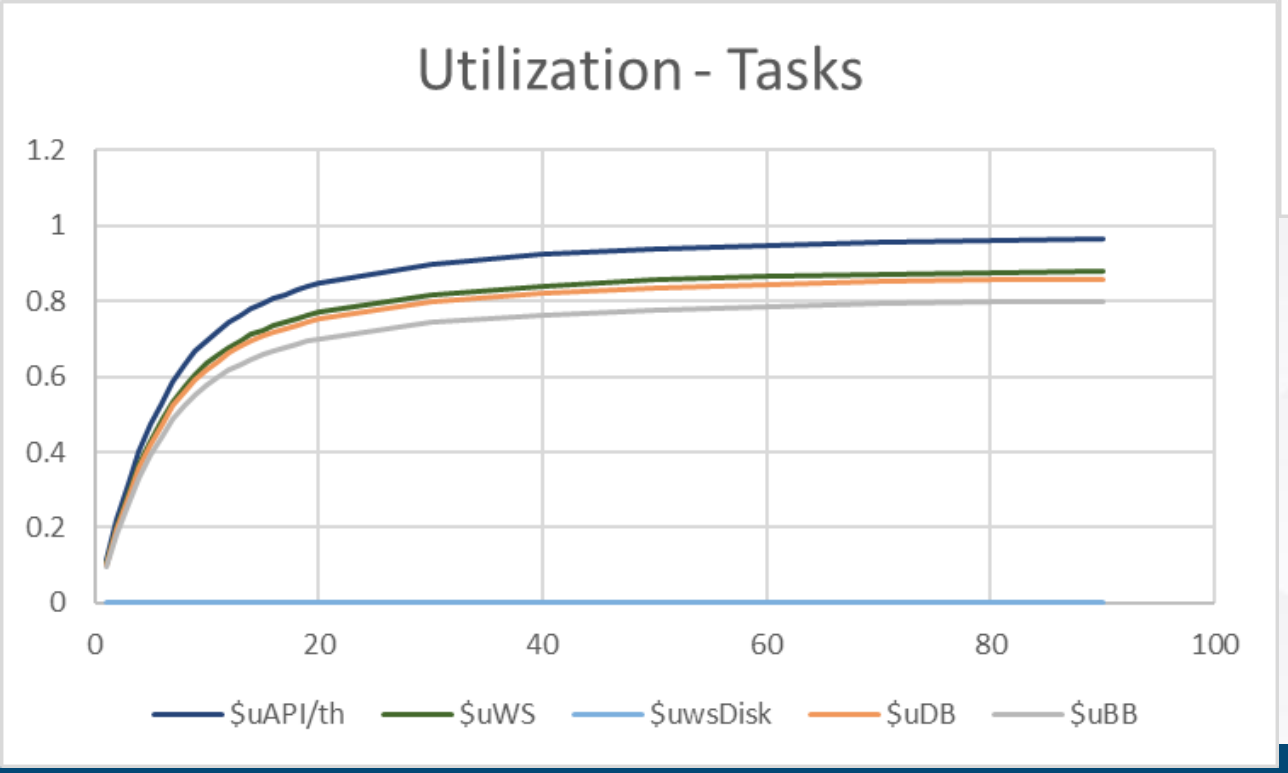
- Our System is high sensitive to the cache hit rate PCM.
- It is mainly duo to the fact that the cache is at the start of the system.
- This is expected from mobile apps since they are design to reduce waiting time.

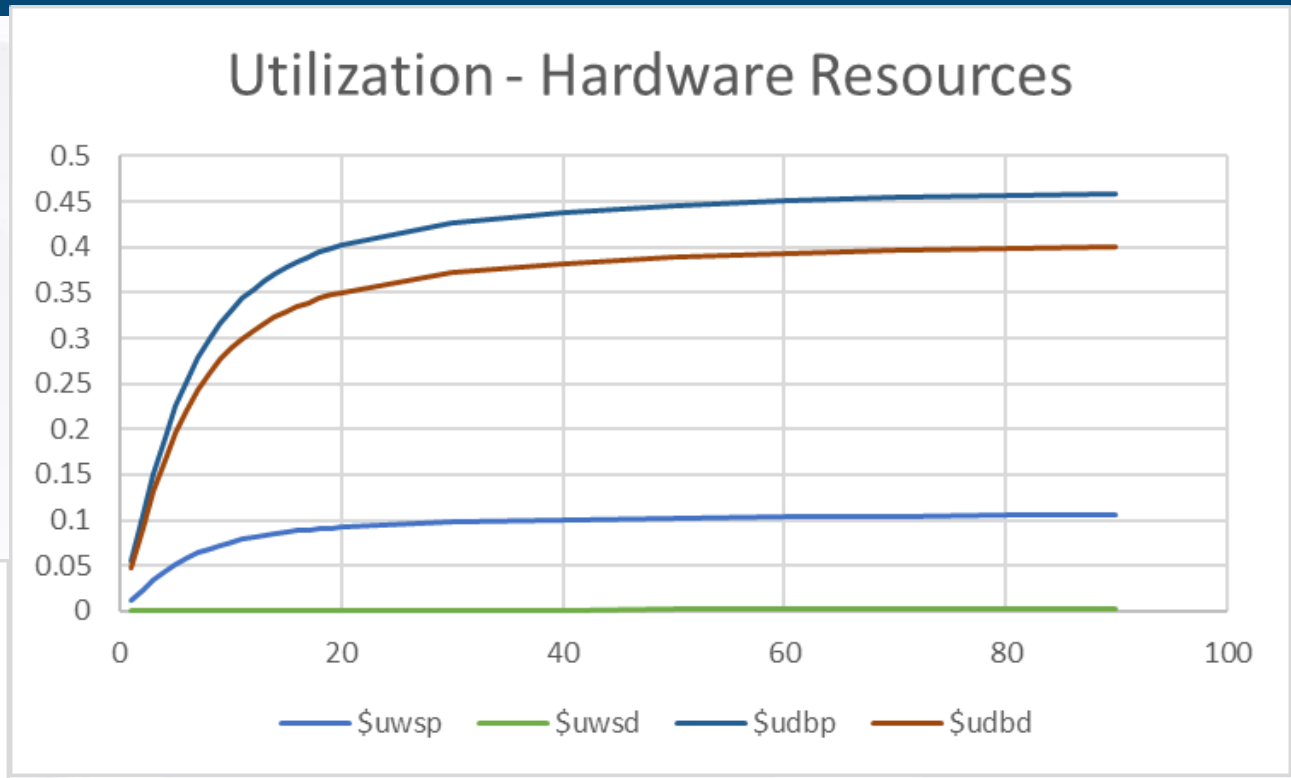


05 Results – Base Case

Bottleneck – Task API

- From the utilization graphs, it shows the API task saturates first.
- It shares the web server processor with web server.



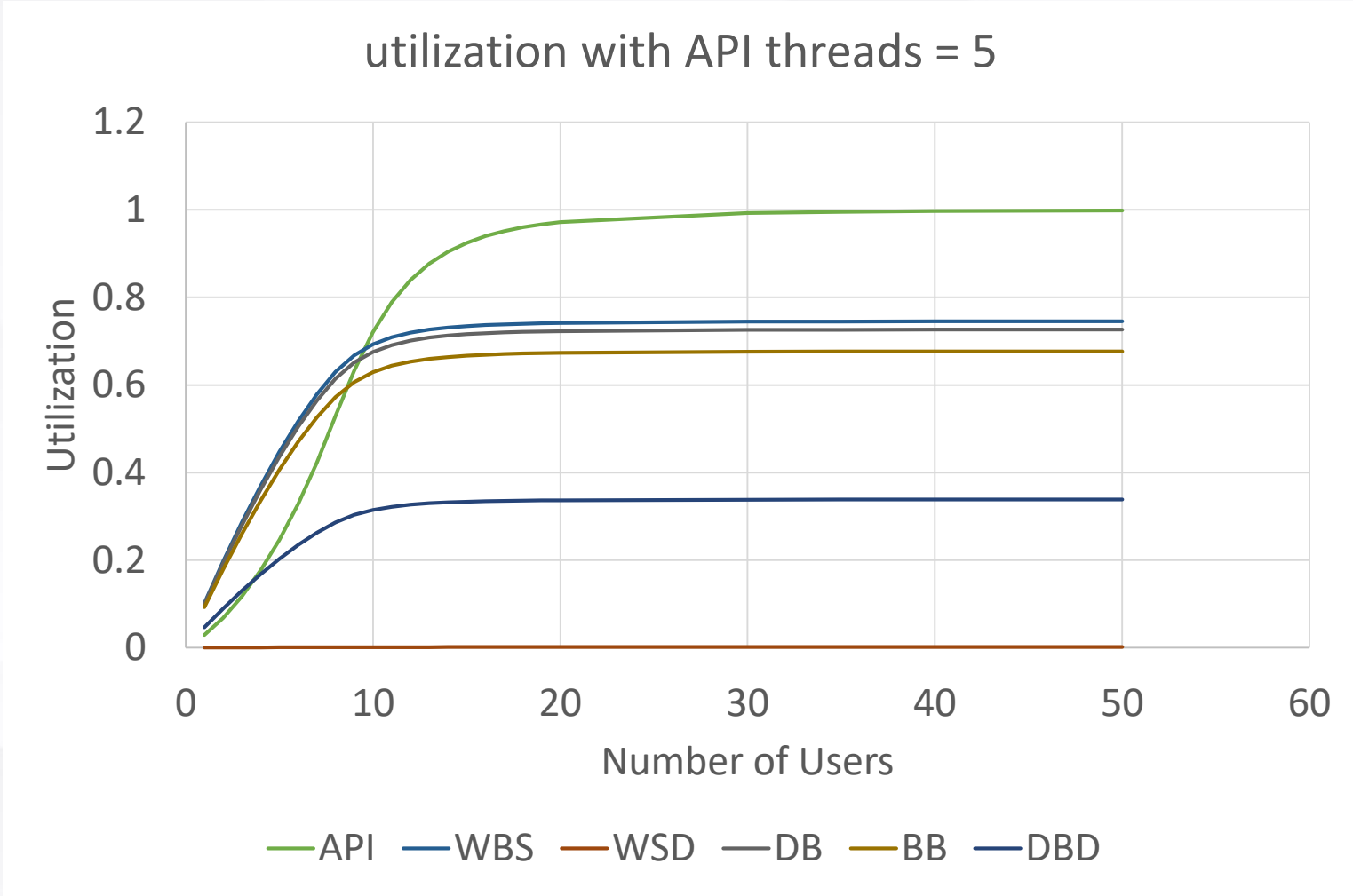


Carleton UNIVERSITY

05 Results - API threads =5

Bottleneck – Still Task API

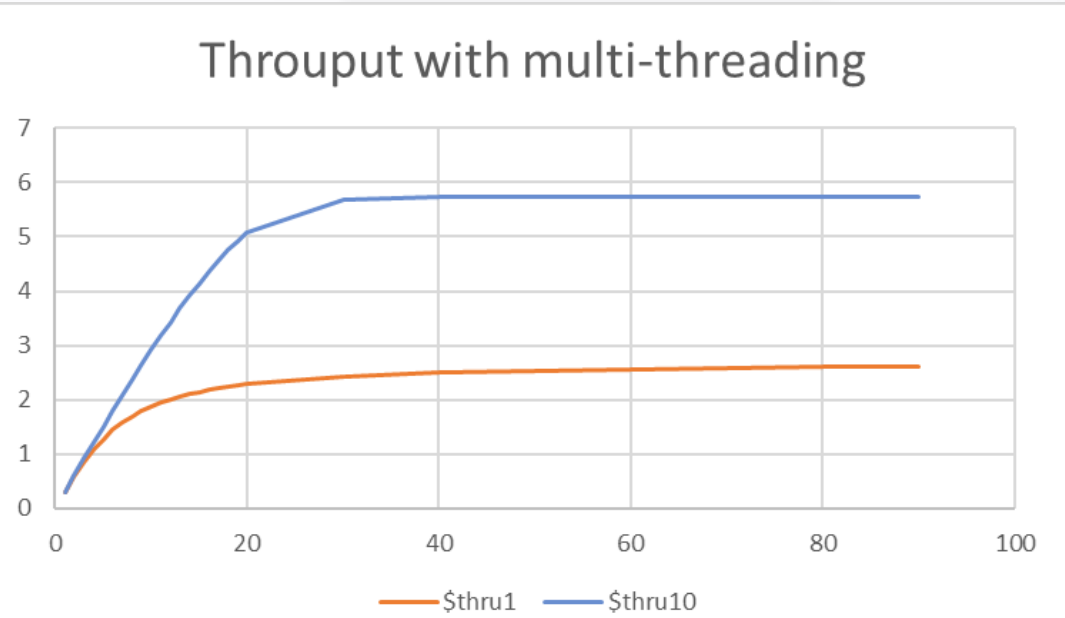
- **It is saturated first while all its children in the LQN tree are not.**
- **Therefore, we continue to multithread the tasks.**

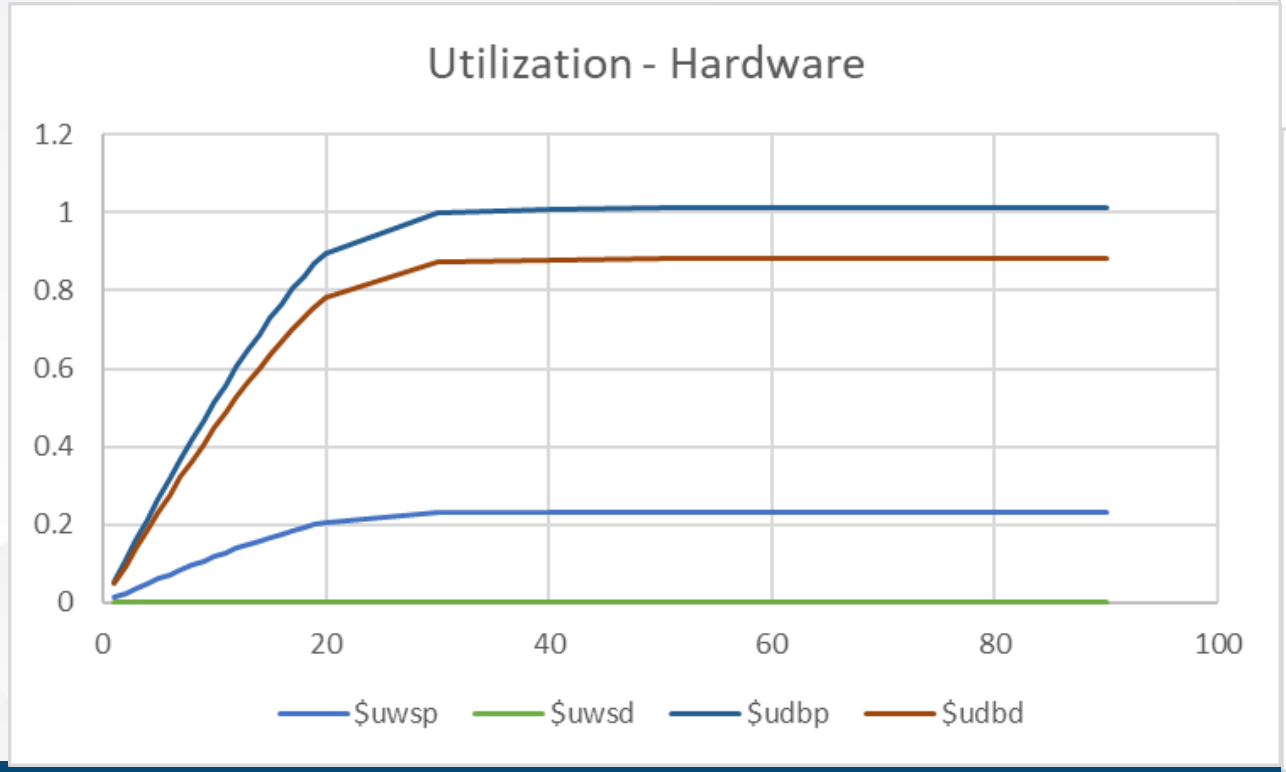


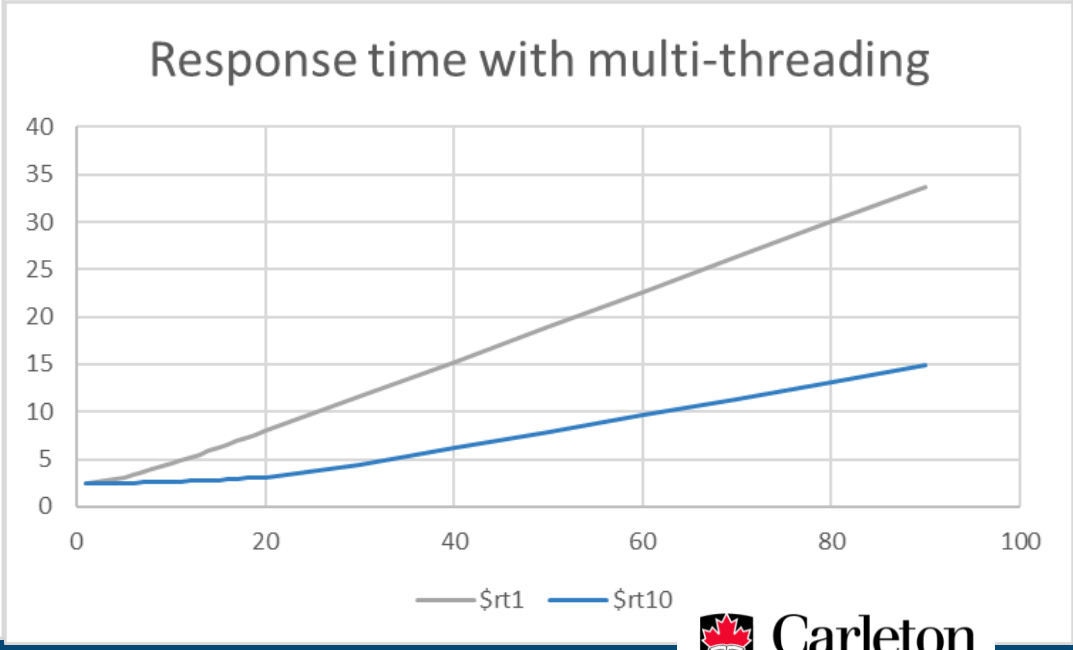
05 Results – all tasks threads =10

Next Hardware Bottleneck – DBP

- All Tasks threads = 10
- Hardware bottleneck reached







06 Improvement

Fast Path

- Dominant workload function – getImage()
- We can decrease the response time by letting PCM less by letting the AppDB request images before the AppGUI request them.

Batching

- Repeated function – retrieveComment()
- Combine requests of retrieveComments() into batches so that it executed once instead of 7 times.

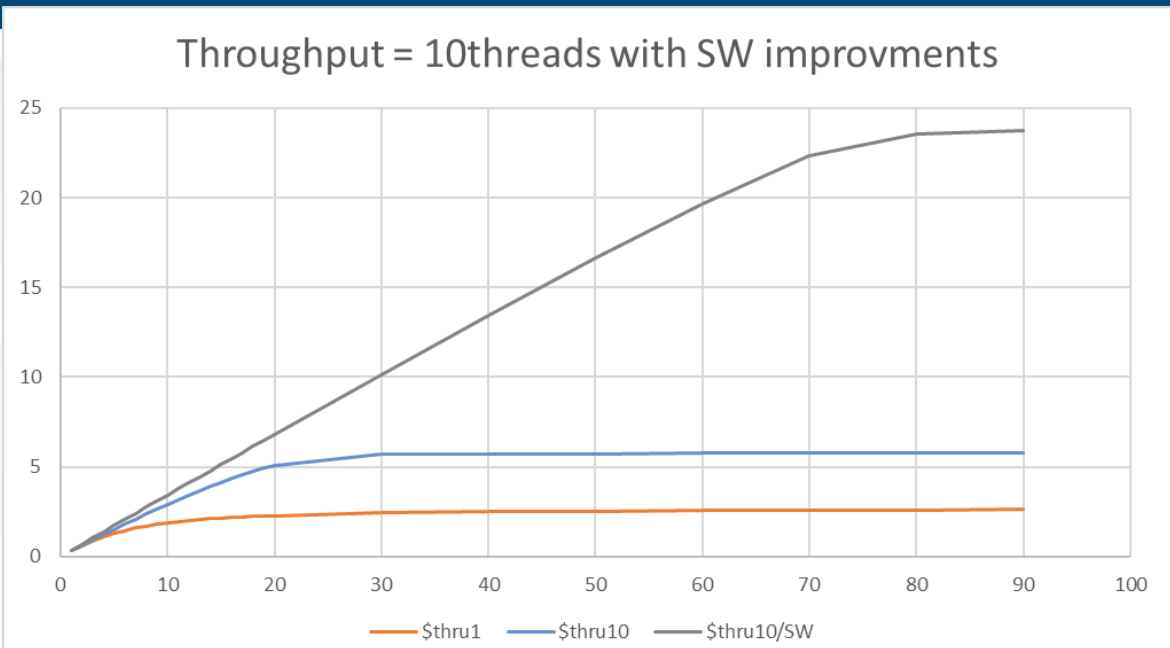
Coupling

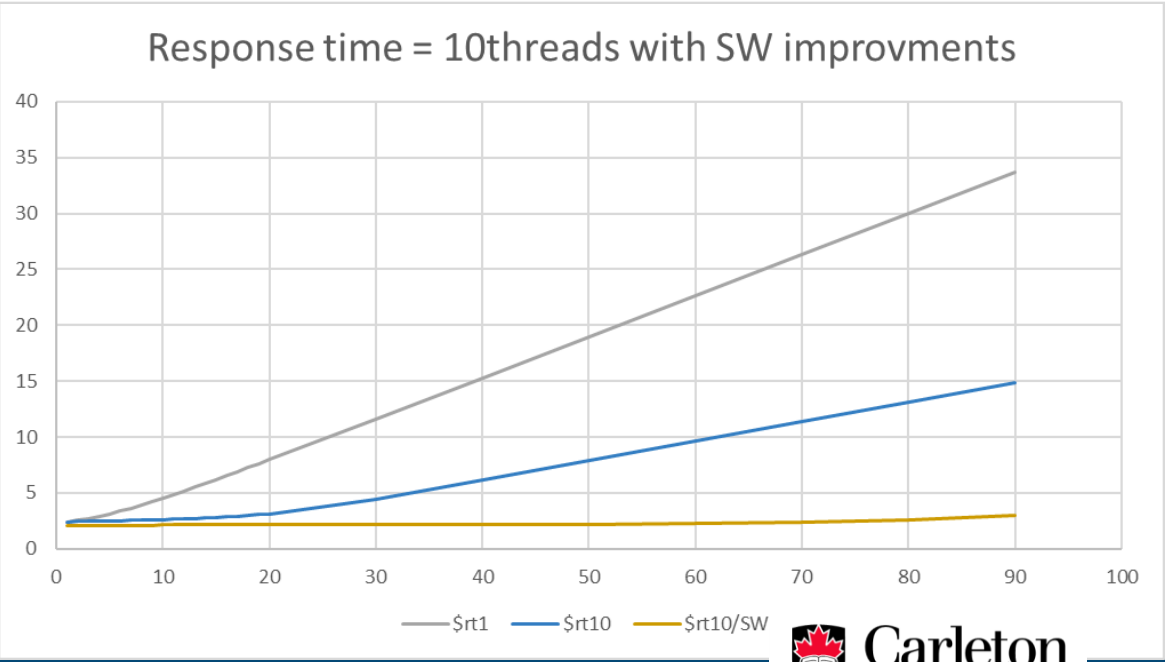
- Function – getImage()
- By making the comments part of the Image data to reduce calls and processing times.
- This will add extra processing demand on PostCommet() function but since it is not critical as the getImage one.

07 Improvement - Results

Why software improvement is important !

- All Tasks threads = 10
- Batching (1 retrieveComment call)
- fast path (PCM = 0.1).
- Throughput jumps from 2.5 in base case to 24 instruction/sec with 10threads and software improvements.
- The response time reduced greatly and now our system can handle 90 users with no extra delays.





Carleton UNIVERSITY

Any Questions ?

Thanks for listening!