Master's Defense

Ellias Palcu

THE UNIVERSITY OF TENNESSEE KNOXVILLE

October 20, 2017

Defense Overview

- My Life
  - Education
  - Experience
  - Extracurricular
- Courses
  - Fall 2016
  - Spring 2017
  - Summer 2017
  - Fall 2017
- Projects
  - COSC 594
  - ECE 551
  - COSC 534
- Future

My Life

- Education:
  - UTK: Computer Engineering, BS (Aug 2012 - Dec 2016)
  - UTK: Computer Engineering, MS (Jan 2017 - Present)
    - Program: 5-year BS/MS
    - Concentration: Embedded Systems
- Experience:
  - ORNL: SULI Computational Science Internship (Jun - Aug 2015)
  - TVA: Admin Programmer (Nov 2015 - Aug 2016)
  - UTK: GTA (Jan 2017 - Present)
  - Eaton: Embedded Software Engineer, intern (Sep 2016 - Present)
- Extracurricular:
  - Active Life
  - Traveling

Courses

- Fall 2016
  - COSC 530: Computer Systems Organization
  - COSC 594: Supercomputer Design & Analysis
  - ECE 551: Digital System Design
- Spring 2017
  - COSC 560: Software Systems
  - COSC 534: Network Security
- Summer 2017
  - ECE 567: Forensic Engineering
  - COSC 575: High Perf Comp Model/Visualization
- Fall 2017
  - ECE 491: Applicat Linear Alg/Engr Systems
  - ECE 529: Radio and Satellite Communications
  - ECE 599: Adv Cyber-Physical Sys Security

COSC 594

Parallelized Facial Recognition

✣ Fall 2016 ✣

Prediction Overview

- Process:
  - Read in images. (Potentially on the order of hundreds)
  - Separate into two numpy arrays:
    - Samples: Image vectors.
    - Labels: Assign label to each image vector for classification.
  - Train algorithm(s) on image datasets.
  - Predict on face(s) using trained data.
- Implemented Algorithms:
  - Classification/Supervised learning:
    - KNN
    - Random Forest
    - Ada Boost
    - SVM

Prediction Observations

- Implementation:

```
0, Ellias , 39, 40, 41, 45, 46, 47
1, Greg , 78, 79, 80, 81, 82, 83
2, Aaron , 84, 85, 87, 88, 89, 90
3, Jordan , 93, 94, 95, 96, 97, 98
4, Caleb , 99, 100, 101, 102, 103, 104
5, Dean , 105, 106, 107, 108, 109, 110
6, Parker , 111, 112, 113, 118, 119, 120
7, Kelley , 121, 122, 123, 124, 125, 126
```

  - Image file with image ranges corresponding to each label.

- Results:
  - Dependent on camera.
  - Dependent on training data.
  - Struggles with low-resolution images.
  - Superior prediction on high-resolution images.

Prediction Results

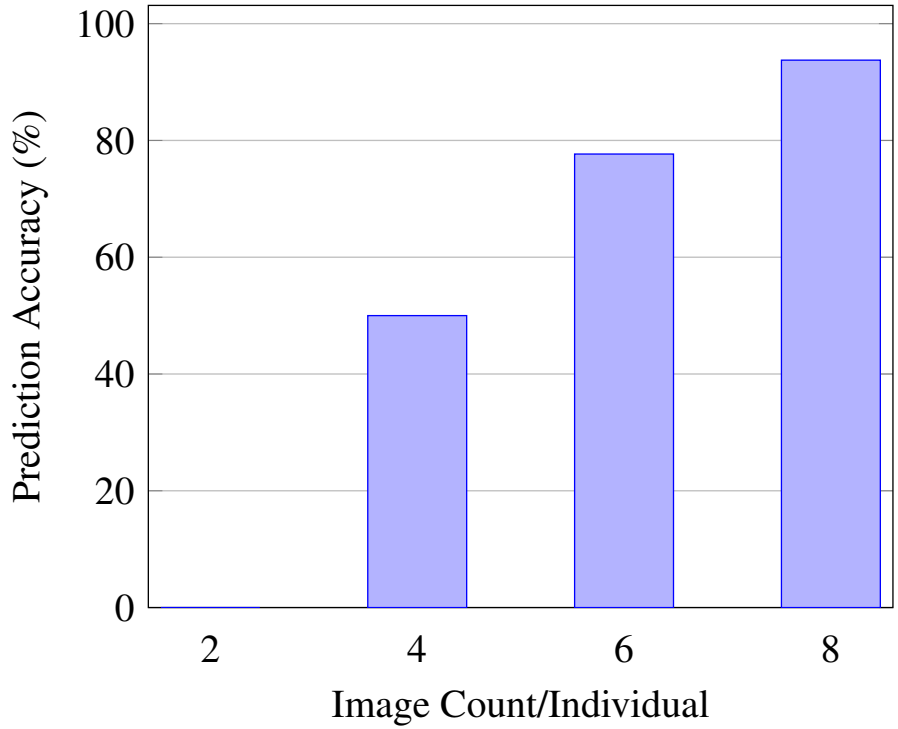
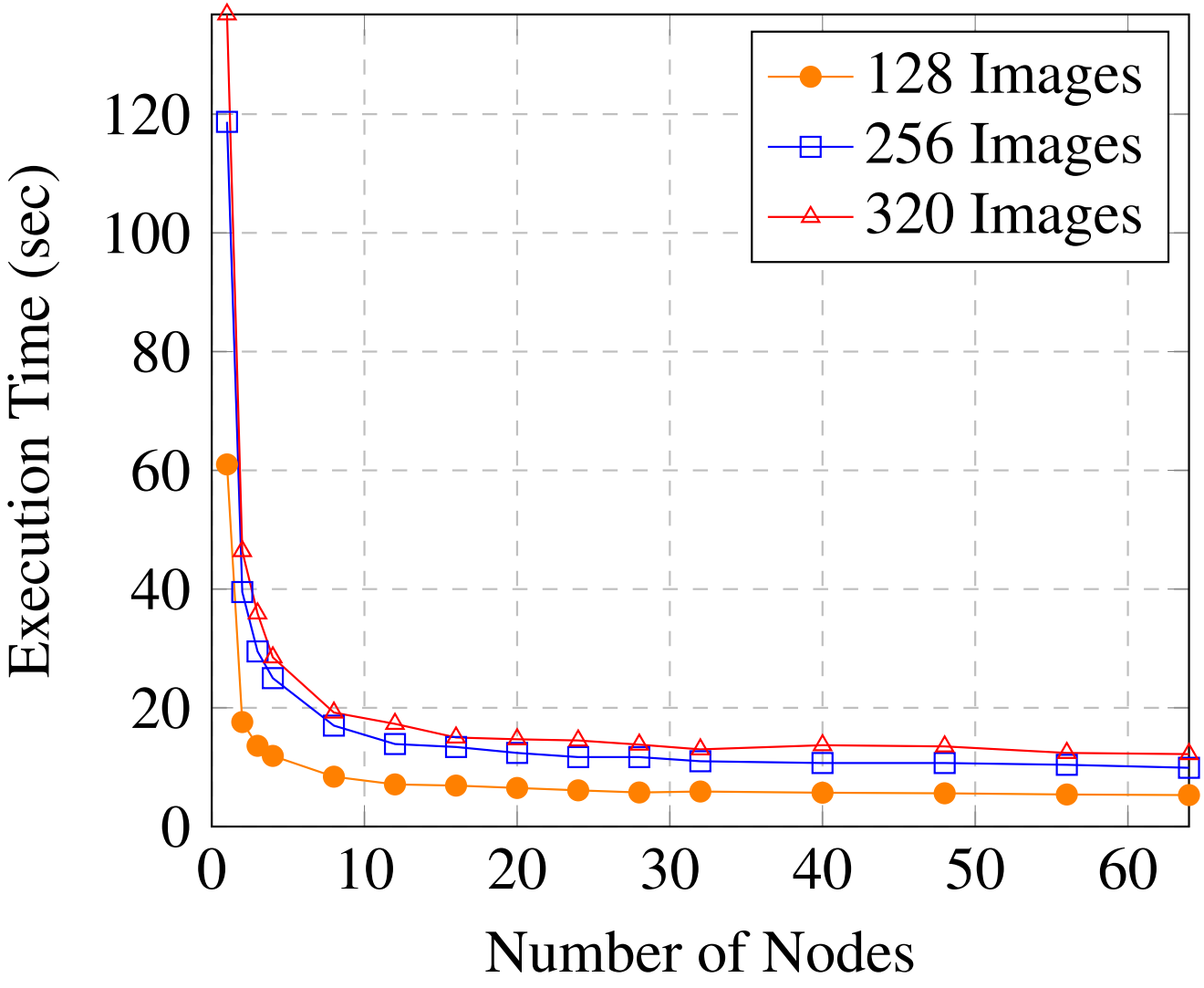


Figure 1: KNN facial prediction results.

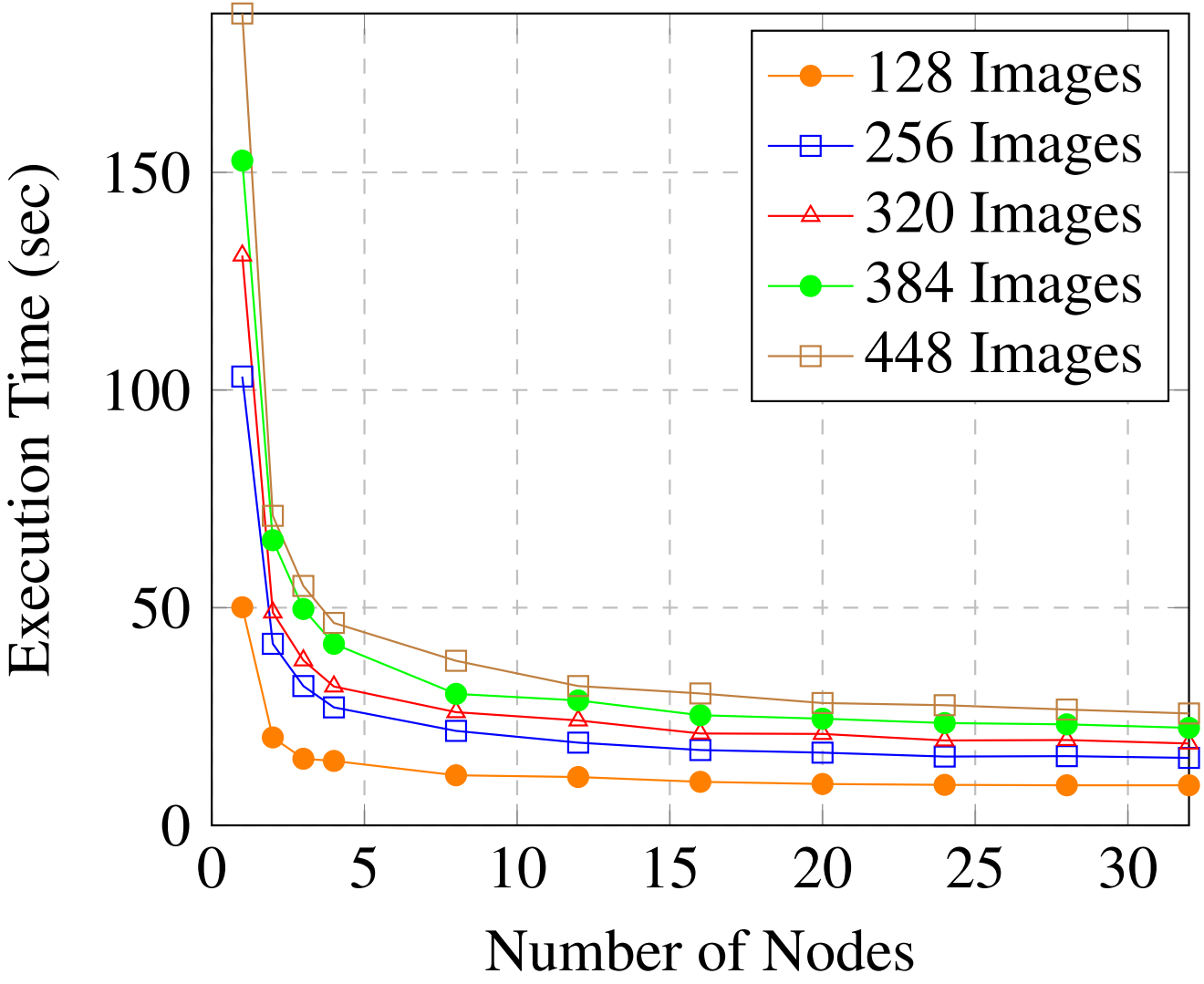
Parallelization Overview

- Approaches:
  - Point-to-point communication:
    - Master process responsible for distributing jobs.
    - Interconnect between nodes too slow.
    - Overhead from constant communication.
  - Dynamic process management:
    - Each process gets copy of data.
    - Upon job completion, worker broadcasts to surrounding nodes.
    - Suffers from slow interconnect speeds.
  - Collective communication:
    - Slices data into even junks.
    - Scatters among workers.
    - Worker performs processing on individual chunk.
    - Master process gathers all processed images.

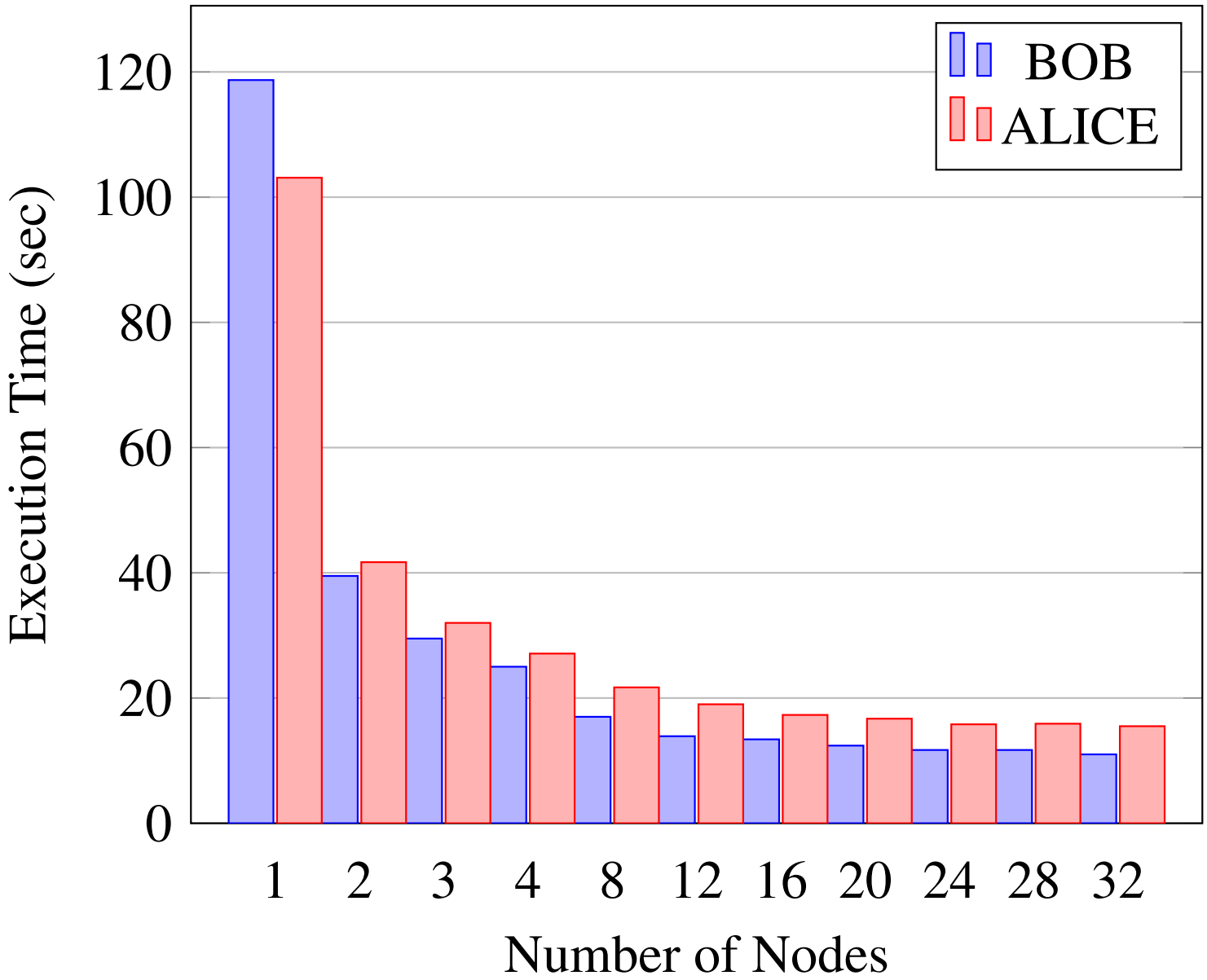
BOB Results


128 Images
256 Images
320 Images
Execution Time (sec)
0
20
40
60
80
100
120
0
10
20
30
40
50
60
Number of Nodes

ALICE Results



BOB vs ALICE (256 Images)



Parallelization Conclusions

- BOB:
  - Speedup across 64 nodes:
    - 128 images: 11.5x
    - 256 images: 12x
    - 320 images: 11.2x
- ALICE:
  - Speedup across 32 nodes:
    - 128 images: 5.4x
    - 256 images: 6.7x
    - 320 images: 7x
    - 384 images: 6.8x
    - 448 images: 7.3x

ECE 551

Bread Proofing Box

✣ Fall 2016 ✣

Project Overview

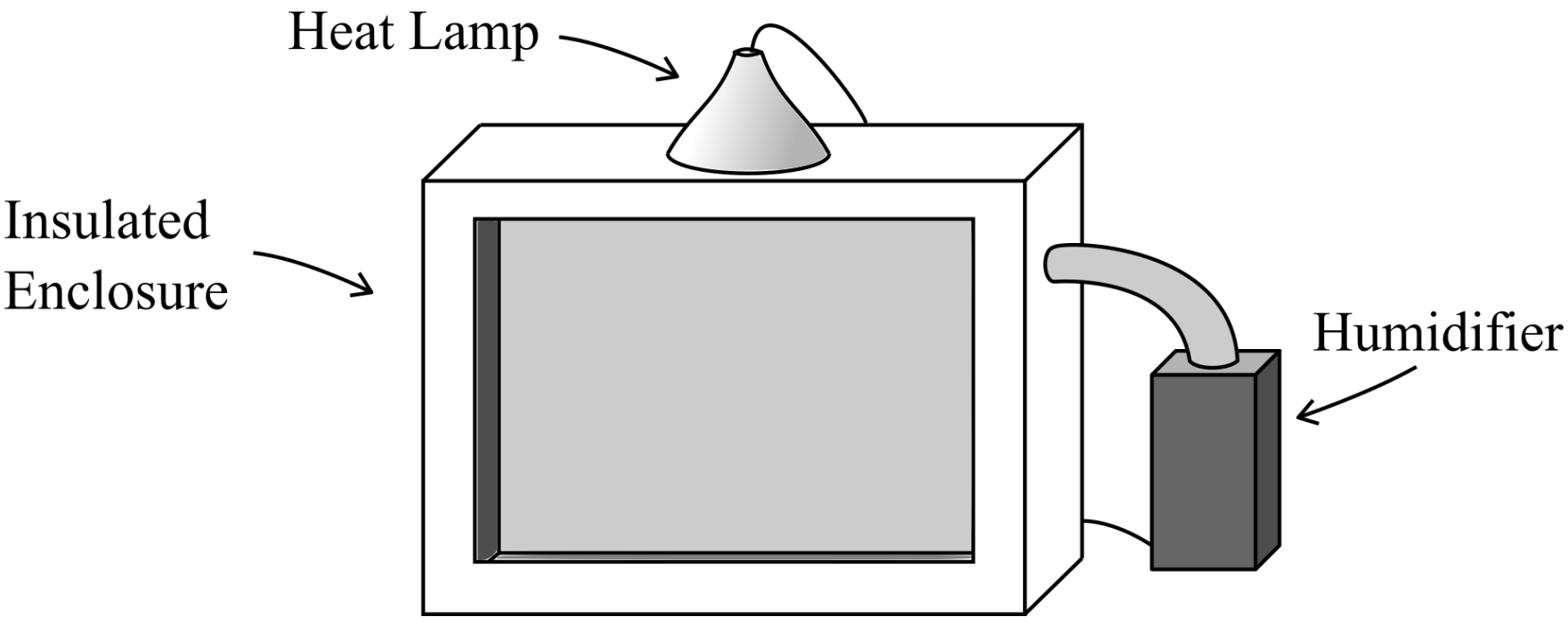


Figure 2: Bread Box

High level block diagram

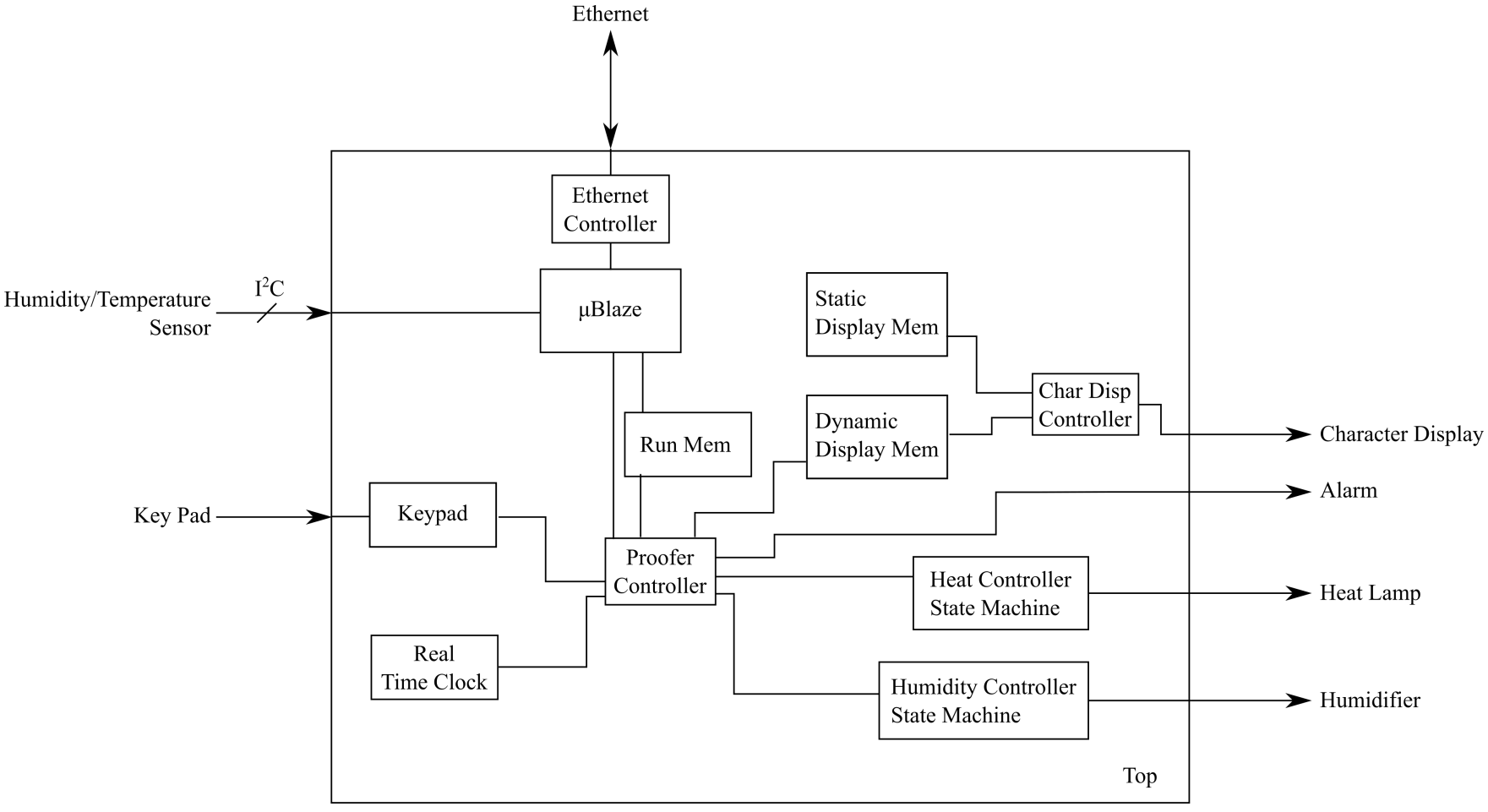


Figure 3: High level block diagram.

High level logic diagram

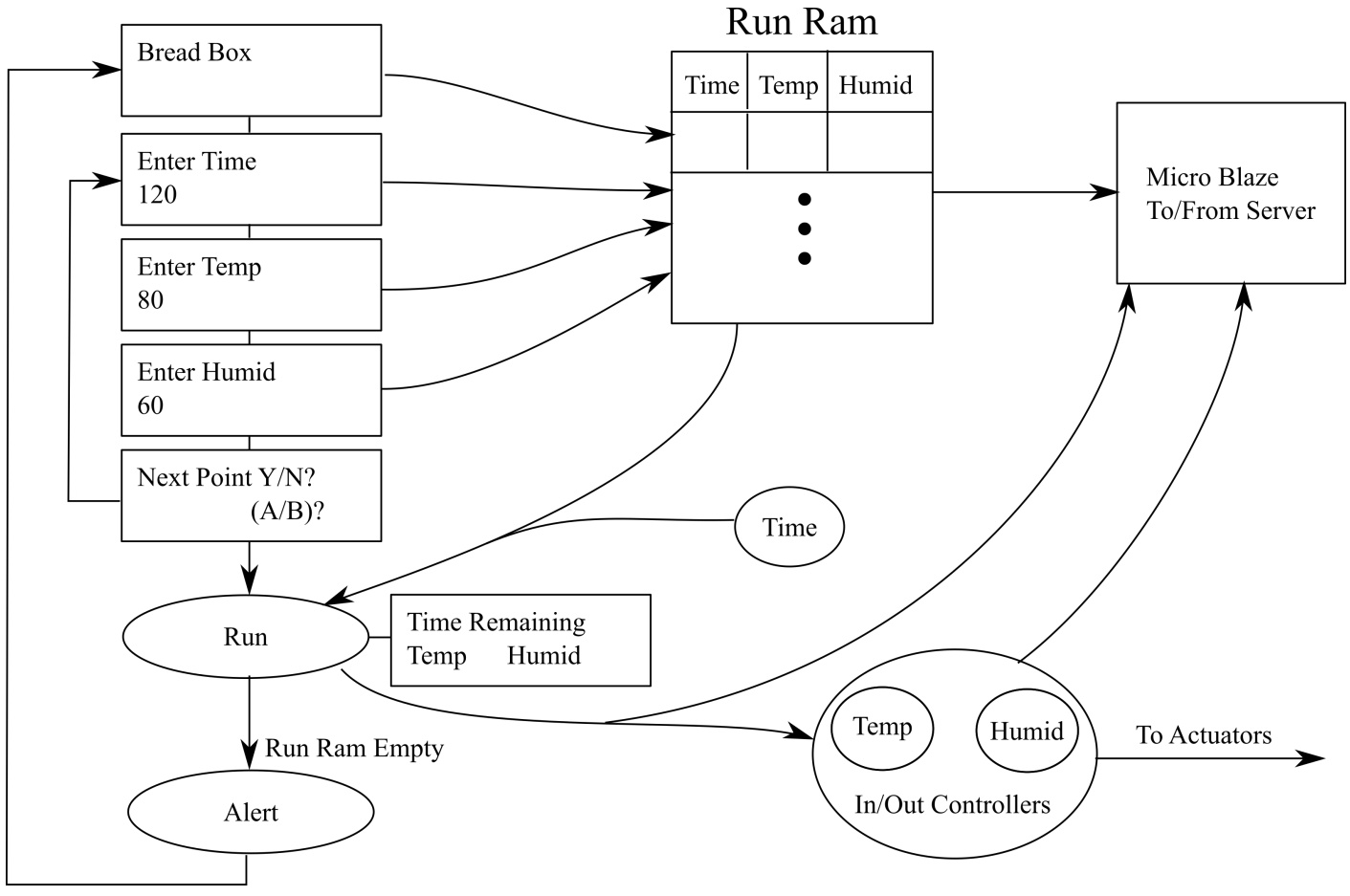


Figure 4: High level logic diagram.

I^2C Devices

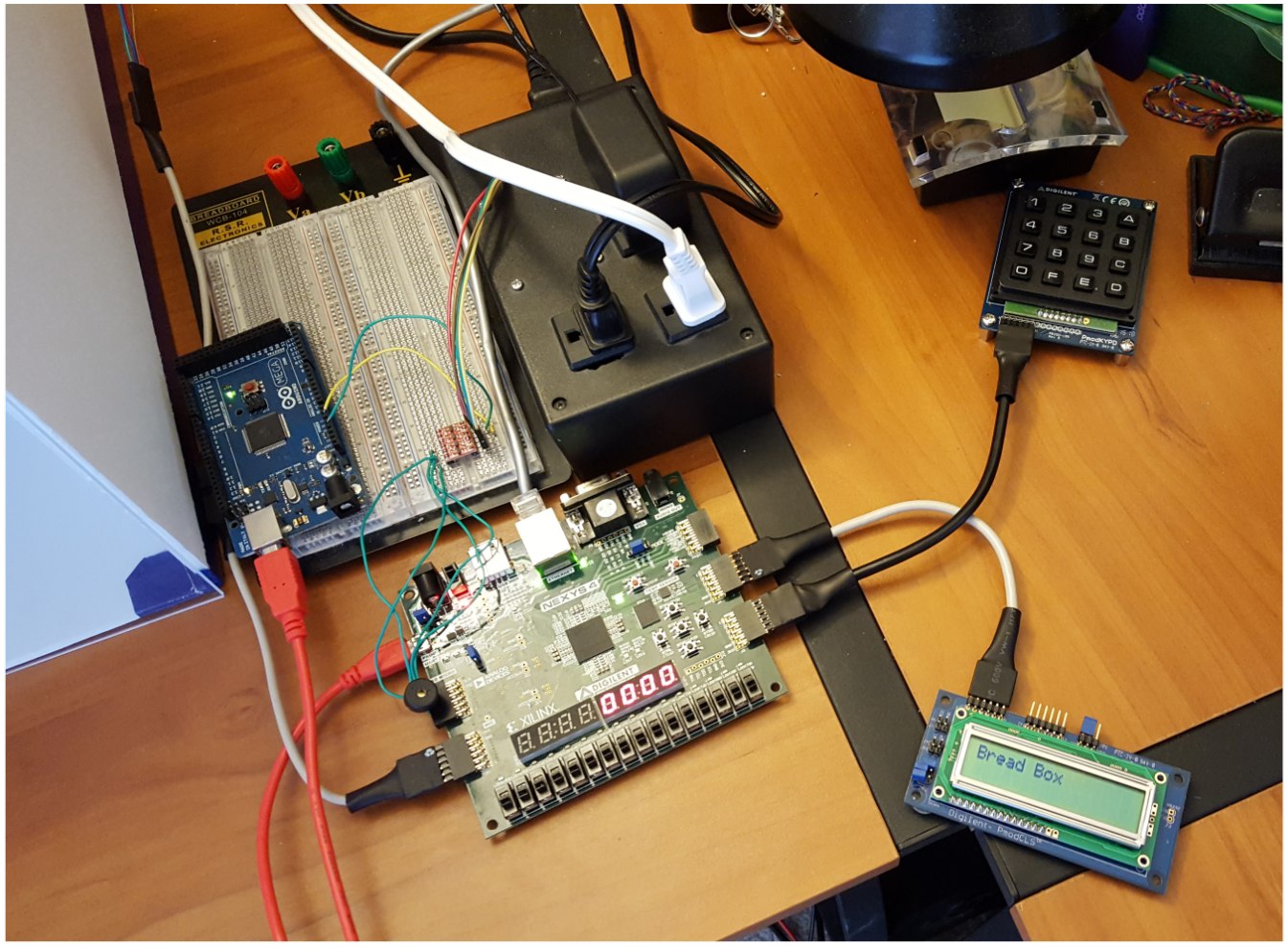
- Design utilizes AIX IIC MicroBlaze module to connect to:
  - HIH6130 Temperature and Humidity Sensor
- Able to connect to the PmodRTCC Real Time Clock, but unable to successfully configure it
  - Utilize Nexys 4's on-board, hardware clock for approximate timekeeping
- Accomplished goals:
  - Able to read temperature and humidity successfully from sensor
  - Able to make state decisions based on this input data
  - Able to actively configure output (heat lamp and humidifier)

Interface

- Server Development:
  - Lack of hardware availability
    - Simulate on mock server, before implementing on FPGA
  - Utilize Microblaze to setup server for transmitting temperature, humidity, and time readings
- Accomplished goals:
  - Receive temperature/humidity and timing information
  - Graph incoming data of current runs
- Accomplished stretch goals:
  - Real-time view of graphical data.
  - Saving of previous runs.
  - Program the proofing run via the web interface.
    - Issues regarding web security.

Results

Results



Results

Demo

https://youtu.be/CaS0XT1j_Mc

COSC 534

Extended Statistical Disclosure Attack

✣ Spring 2017 ✣

Background

- Communication:
  - Senders
  - Receivers
- Mix Networks:
  - Chaum:
    - Messages wrapped in layers of public-key cryptography.
    - Mixes (potentially many) then decrypt, delay, re-order messages before dispatching to recipients.
  - GOAL: Allow senders to *anonymously* send messages to recipients.

Statistical Disclosure Attack

$$\vec{v} = b\frac{\sum_{i=1}^{t} \vec{o_i}}{t} - (b-1)\vec{u}$$

- $\vec{v}$: Alice's behavior vector
  - Consists of N elements corresponding to each of Alice's potential recipients
- **b**: Batch size of the mix
- **t**: Rounds, number of observations
- $\vec{o_i}$: Observation/probability vector that Alice sent message to each potential recipient
  - Element is 1/b if user received message in round; else, 0
- $\vec{u}$: Known vector of cover traffic sent by other users

Extended Statistical Disclosure Attack

$$\overrightarrow{v} = \frac{1}{\bar{m}}[b * \bar{O} - (b - \bar{m})\bar{U}]$$

- $\bar{m}$: Arithmetic mean of messages sent by Alice
- $\bar{O}$: Arithmetic mean of batches in which Alice contributed to
- $\bar{U}$: Arithmetic mean of batches in which Alice did *not* contribute to
- Differences:
  - Alice can send multiple messages and non-uniformly to any recipient for any given round
  - Remove assumption that adversary has full knowledge of the distribution, $\overrightarrow{\mathbf{u}}$, of background/cover traffic

Project Details

- 260 users: 'a0' through 'z9'
- Each user has two friends, *not* bidirectional
- Batch size of 32:
  - Set of 32 users is picked uniformly at random
  - Each user picks one of its two friends to send message to
- Part 1:
  - Find friends of targets 'a0', 'b0', 'c0', 'd0', ... , 'z0'

```
S: ['b9', 'm1', 'j1', 'p7', 'v5', 's5', 'a0', 't7', 'x1', 'r7', 'r2', 'c2', 'g1',
    'q7', 'u8', 'w8', 'o8', 't5', 'j9', 'j8', 'c1', 'e0', 'd1', 'b7', 'l0', 'w0',
    'k8', 'r9', 'i2', 'g5', 'c0', 'c6']

R: ['i7', 'a9', 'j3', 'l2', 'q5', 'c2', 'o8', 'a0', 'j2', 'z1', 'e6', 'u2', 'h2',
    'v4', 't7', 'o3', 'f0', 'r5', 'd5', 'x3', 'j7', 'o1', 'v6', 's9', 'f4', 'a8',
    'm4', 'x7', 'm1', 'i8', 'h4', 'e4']
```

- Part 2:
  - Manual harvesting of rounds
- Part 3:
  - Find private/unreachable peers within network

Implementation

```
– Initialize empty targets dictionary
– Initialize receiver dictionary of all receivers in mix with value of 0
– For each round:
    – Set targets list ['a0', ... , 'z0']
    – Copy receiver dictionary into a round dictionary
    – For each receiver in round:
        – Update its value in round dictionary to 1/(num messages)
    – For each sender in round:
        – If sender is in targets list:
            – Pop target from list
            – If sender not in targets dictionary:
                – Initialize sender key, value = { "o_list": [round dictionary],
                                                   "u_list": []
                                                 }
            – Else:
                – Add round dictionary to sender's "o_list"
    – For each target:
        – If sender not in targets dictionary:
            – Initialize sender key, value = { "o_list": [],
                                               "u_list": [round dictionary]
                                             }

        – Else:
            – Add round dictionary to sender's "u_list"
– Return targets dictionary
```

Implementation Cont'd

```
- Initialize empty reduced dictionary
- For each key, value in targets dictionary:
    - Initialize empty O dictionary
    - Initialize empty U dictionary
    - oCount = Number of "o_lists" for key
    - uCount = Number of "u_lists" for key
    - m = number of messages
    - For each dict in "o_lists":
        - For each recipient, prob in dict:
            - If recipient in O dictionary:
                - value = value + prob/oCount
            - Else:
                - value = prob/oCount
    - For each dict in "u_lists":
        - For each recipient, prob in dict:
            - If recipient in U dictionary:
                - value = value + prob/uCount
            - Else:
                - value = prob/uCount
    - For each (oKey, oValue), (uKey, uValue) in O, U:
        - v[i] = (oKey, m*oValue - (m-1)*uValue)
        - i++
```

Results

```
a0 ⟶ ('g5', 0.36), ('x7', 0.66)
b0 ⟶ ('b2', 0.28), ('z7', 0.53)
c0 ⟶ ('r5', 0.35), ('y9', 0.60)
d0 ⟶ ('s3', 0.30), ('w2', 0.32)
e0 ⟶ ('e6', 0.35), ('y0', 0.67)
f0 ⟶ ('j5', 0.41), ('q3', 0.59)
g0 ⟶ ('k4', 0.34), ('q6', 0.38)
h0 ⟶ ('d8', 0.53), ('l2', 0.53)
i0 ⟶ ('w2', 0.29), ('b7', 0.48)
j0 ⟶ ('w2', 0.44), ('y5', 0.75)
k0 ⟶ ('p9', 0.45), ('s3', 0.64)
l0 ⟶ ('s7', 0.45), ('i8', 0.48)
m0 ⟶ ('w2', 0.34), ('w4', 0.69)
n0 ⟶ ('e2', 0.58), ('s9', 0.66)
o0 ⟶ ('a2', 0.41), ('k1', 0.64)
p0 ⟶ ('w6', 0.37), ('l6', 0.46)
q0 ⟶ ('g0', 0.35), ('t8', 0.67)
r0 ⟶ ('e4', 0.35), ('x3', 0.68)
s0 ⟶ ('v6', 0.37), ('l6', 0.60)
t0 ⟶ ('r9', 0.46), ('c6', 0.54)
u0 ⟶ ('d7', 0.39), ('h5', 0.60)
v0 ⟶ ('p4', 0.35), ('a9', 0.55)
w0 ⟶ ('x7', 0.39), ('z5', 0.43)
x0 ⟶ ('d7', 0.28), ('r6', 0.38)
y0 ⟶ ('h6', 0.35), ('n9', 0.55)
z0 ⟶ ('m9', 0.39), ('i5', 0.49)
```

Future

- Next week:
  - Zurich, Switzerland
- Graduation:
  - December 16!!!
- New job:
  - Instrumentation Engineer, TVA
  - November 13
- Marriage..?
- PHD..?