Major Projects Ext.

Udit Jain

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

April 15, 2020

Introduction

Introduction

Interests: Systems Design and Engineering, Kernel and Operating System Design, Distributed Systems, Deep learning and Applications of AI, Building Application specific systems (Machine Learning, Computer Graphics). Internships:

- Samsung Research, Seoul, Korea.
- Kyutech Institute of Technology, Fukuoka, Japan.

I am going to talk about some of my major projects:

- Kernel Hacking and OS Design - Improving FreeBSD Kernel Code, OS from scratch. (Prof. Smruti Sarangi, IIT-Delhi)
- Picking and Segregation of Clothes using Deep Learning and Baxter. (Prof. Tomohiro Shibata, Kyutech, Japan)
- APEX: Recovery aware Adaptive File Allocation System for Edge devices. (Prof. Rajkumar Buyya, Univ of Melbourne)
- Semantic Information Extraction(Jaehun Lee, Samsung)
- Disjunctive Rado Number (Prof. Amitabha Tripathi, IIT-Delhi)

Kernel Hacking and OS Design

Goals

1. Improving the Page Replacement Algorithm in FreeBSD.
2. Comparison of Schedulers, Linux CFS and FreeBSD ULE.
3. Implementing OS functionalities from scratch, building on top of xV6(bare-bones OS).
4. Future: Analyzing system calls timings and functions in depth for Linux and FreeBSD and transposing algorithms in between.

Improving Page Replacement

1. Run memory-intensive benchmarks on FreeBSD.
2. Find scope of optimization in page replacement policies.
3. Explore other policies: Least Frequently Used (TinyLFU).

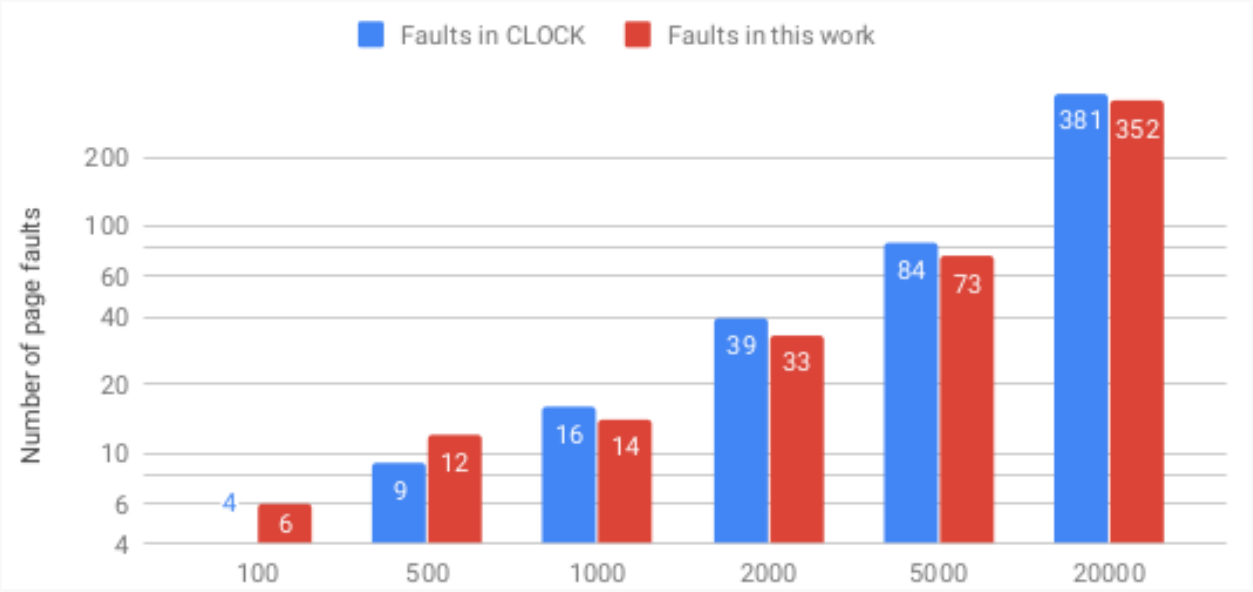
Currently, FreeBSD uses CLOCK [5] page replacement policy . Changed to LFU policy (TinyLFU [6] cache replacement policy). Used SPEC(libquantam and mcf), Filebench, and Sysbench to benchmark the system. Compared policies in terms of:

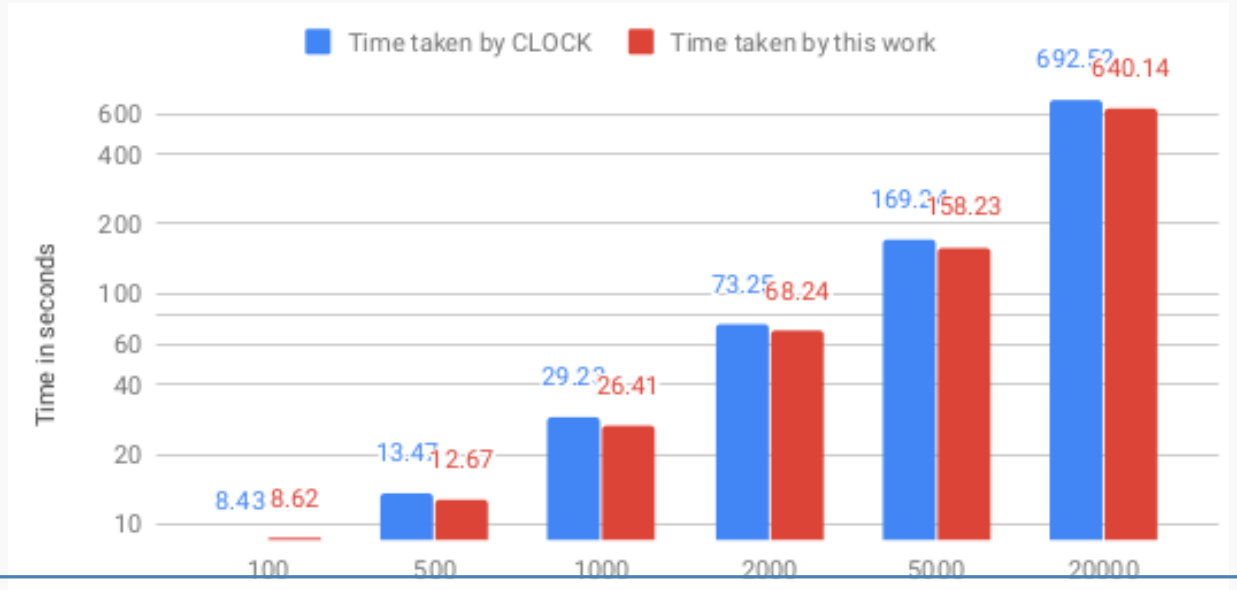
- Memory usage
- Bandwidth usage
- Page faults

Used Minimal Counting bloom filter to implement the **TinyLFU** policy in FreeBSD Kernel. Made it compliant with existing data structures in FreeBSD. Quantitive comparison over the benchmarks: time taken and number of page faults were decreasing.

Improving Page Replacement Results

Timings and Page Fault information for SPEC benchmarks.





Scheduler Comparison

Compare Linux's CFS and FreeBSD's ULE in terms of in terms of:

- Fairness
- Starvation
- Load balancing

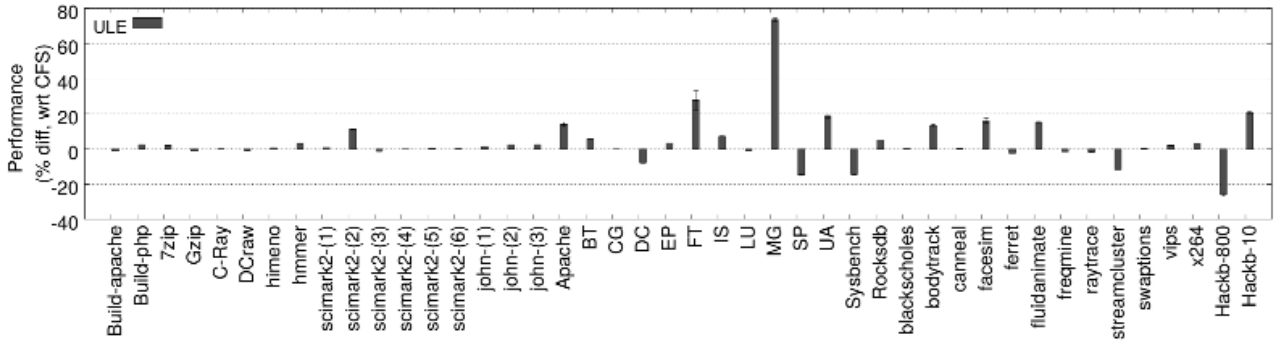
for **Phoronix Phoronix Muli-Core suite, Sysbench, Fibo, Blackoles Ferret** benchmark suites to compare :

- File IO performance
- Scheduler performance,
- Memory allocation and transfer speed
- POSIX thread implementation performance

Implemented custom system calls in Free-BSD for experiments.
The graphs were generated to draw conclusions over which kernel is better for which kind of applications. [7]

Scheduler Comparison Results

The results for Phoronix and Phroronix multi core benchmarks:



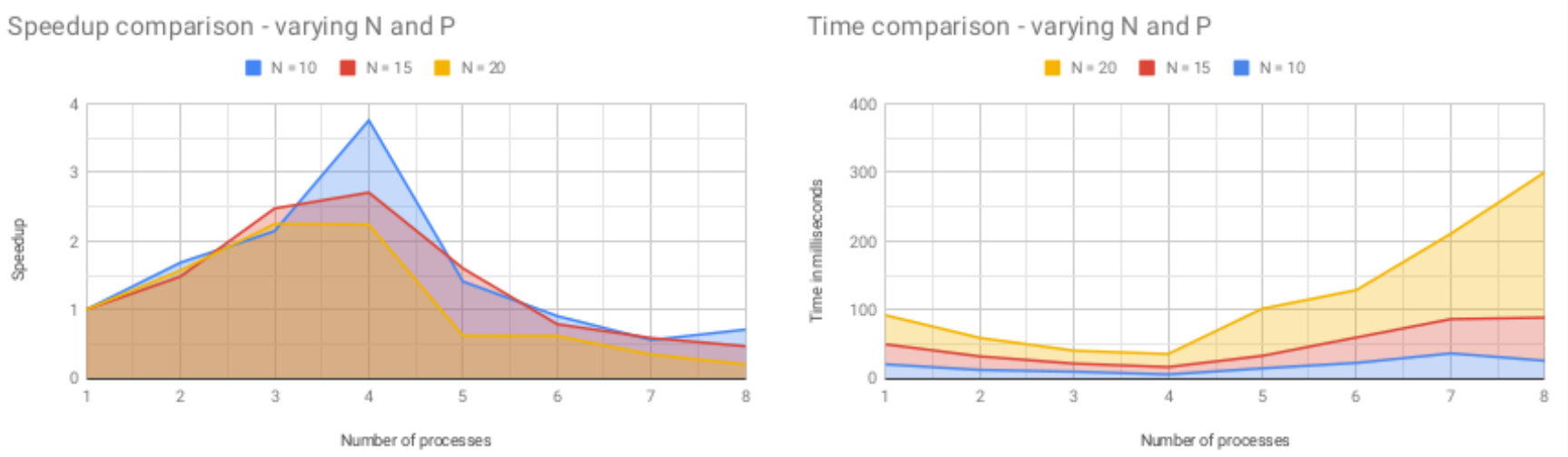
Performance					
Gzip	Scimark2 (jacobi)	Apache	LuaJIT	Sysbench	Carneal
0.028112	1237	13263	1792.12	1229.26	3912
0.027862914	1289	13982.49	1767.98	1209.12	3922
-0.886047023	**4.203718674**	**5.4247908**	**-1.347008**	**-1.6383841**	**0.25562372**

FreeBSD has a significantly lower context switching overhead compared to linux and thus performs better in the multi-app category. One exception is Blackoles due to it's fairness requirement FreeBSD's ULE fails.

Operating System Design

Implemented the following functionalities in xV6:

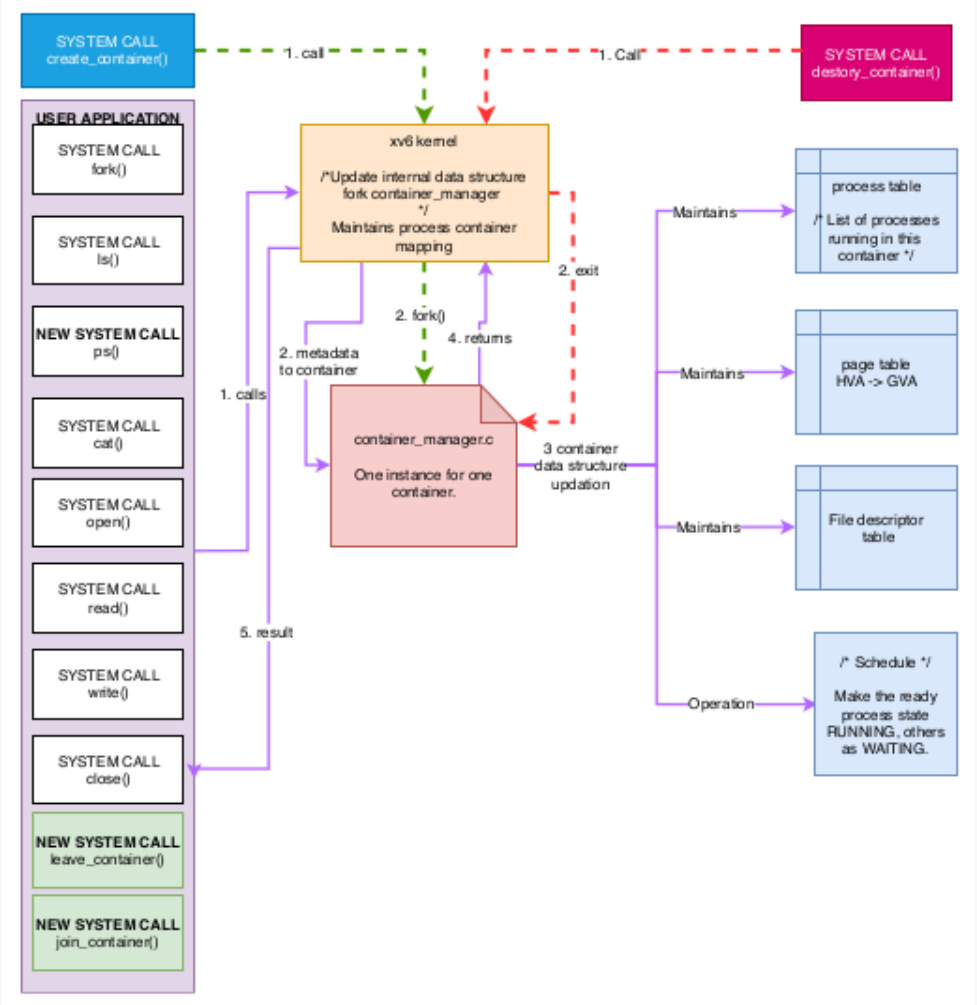
- Signals, Trap handling and synchronization primitives: Barriers.
- Inter-Process Communication: Unicast and Multicast.
- Jacobi and Maekawa algorithm for scheduling. performance comparison and correctness verification of both.



N is the size of Jacobi matrix and P is the number of processes.
Time is in ms and Speedup is $T_{seq}/T_{parallel}$

Operating System Design

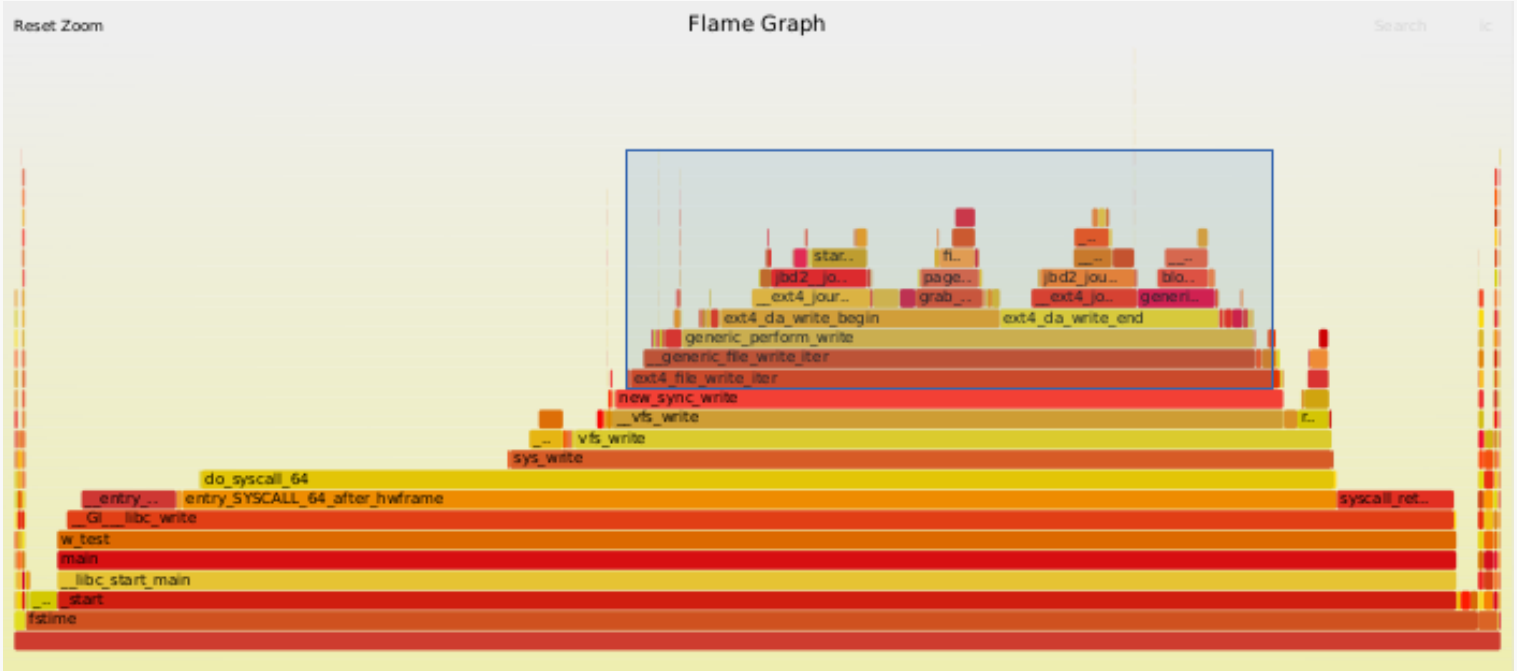
- Virtualization and Container management keeping in mind:
  - Virtual Scheduling
  - Resource Isolation - Page tables and Virtual file system



The Continuation

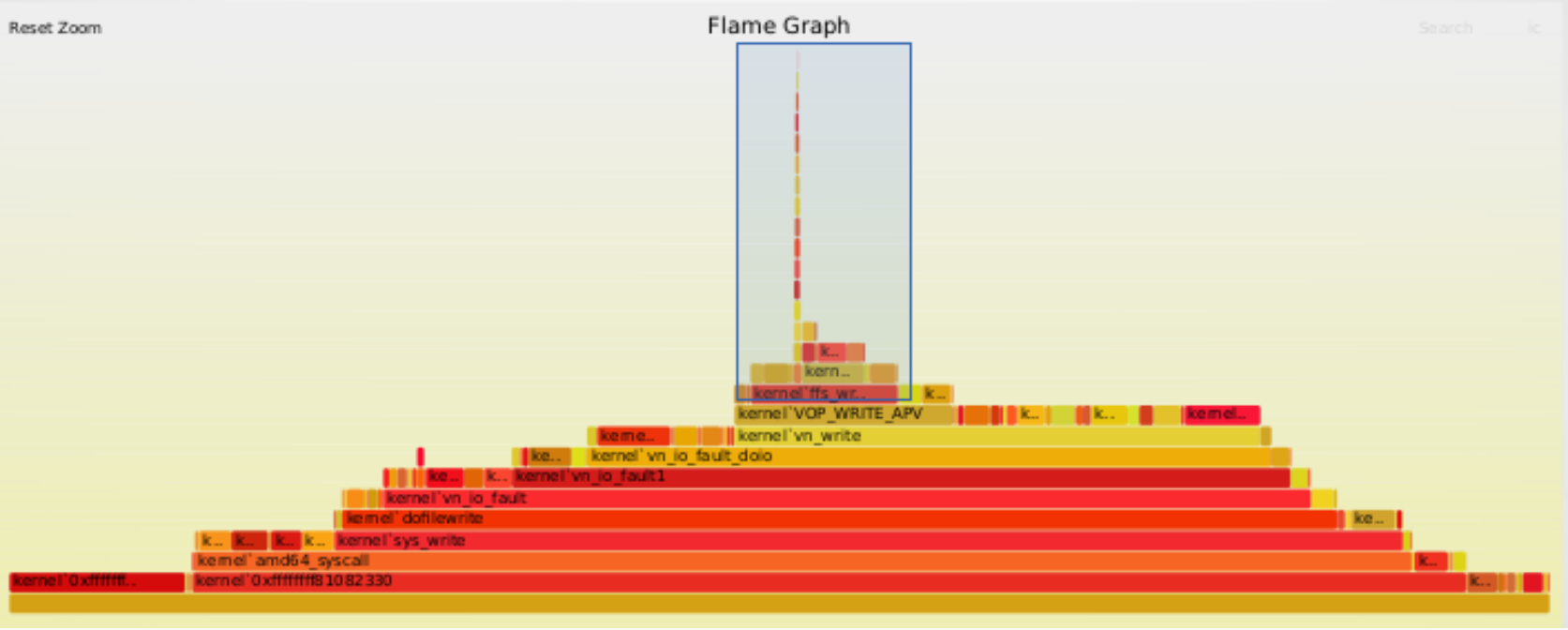
Continued this semester as Comparing Linux and FreeBSD kernels, and making hybrid Kernel. Objectives:

- Using Flamegraphs to analyze the trace of Linux and FreeBSD on the benchmarks Sysbench and Filebench.



The Flamegraph of fsbuffer-w from unixbench test suit in ubuntu, traced with perf.

The Continuation



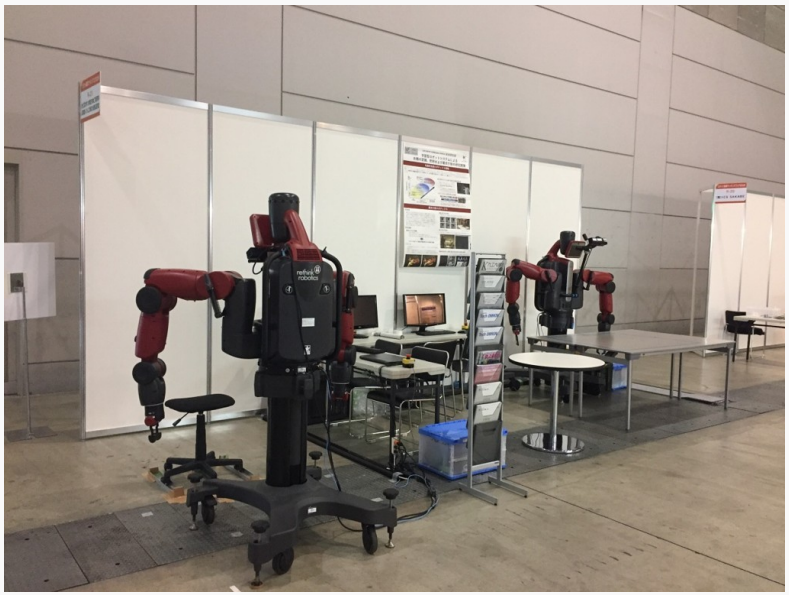
The Flamegraph of fsbuffer-w from unixbench test suit in freeBSD, traced with dtrace.

- Identifying the relevant system calls and what primitives they use and how exactly they differ in functionality accross OSes.
- Transposing the code and using better algorithms to get an improved kernel, which is application specific.

Segregation of Clothes using Deep learning and Baxter

Motivation

Studies have concluded rapidly ageing population in Japan.
Old-Age care homes have the need for automation of sorting and putting on clothes on people.
Baxter is a (human safe certified) robot with large arm and grippers which have actuators to be controlled by ROS operating system.
There is a Kinect and a camera mounted on the Baxter gives us the 2D image and 3D point cloud data. Using this data we have to detect the objects and contours in front of us.



Objectives

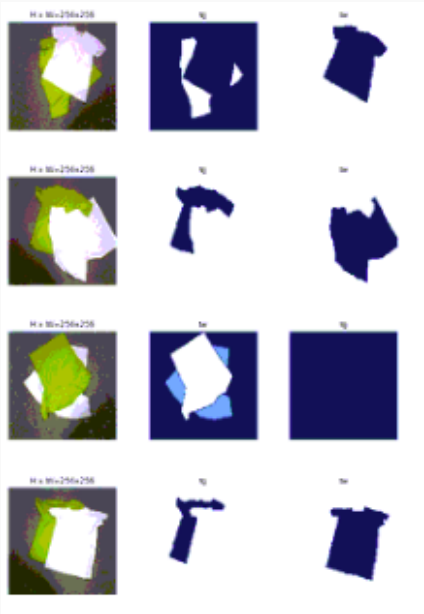
The program for clothing the person once the T-Shirt is in Baxter's grippers was already working perfectly. HAFY robot [1] . The next steps was to pick up the clothes form the table. Subtasks:

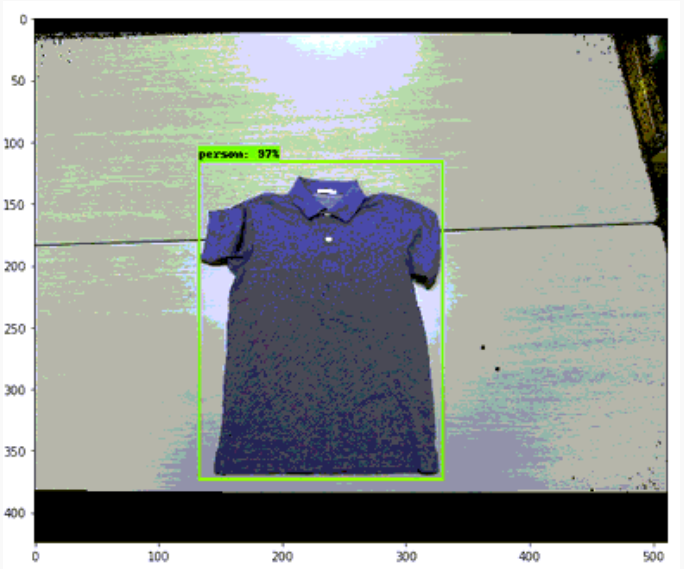
- Identify and differentiate the clothes: T-shirt, Jeans etc.
- Identify the extremities of clothes: Collar, Hands, Waist-corners.
- Map the pixels identified in the 2D plane to the point cloud data in 3D.
- Estimate the position of the cloth, move gripper in the 3D space and grip the cloth.

Project Details

For the T-shirt Detection and identification, I used 2 approaches:

- Segmentation: Get the pixels where the T-Shirts lie with Mask-RCNN [2].
- Object Detection: Getting the bounding box of the T-shirts. Used Deep-Lab library

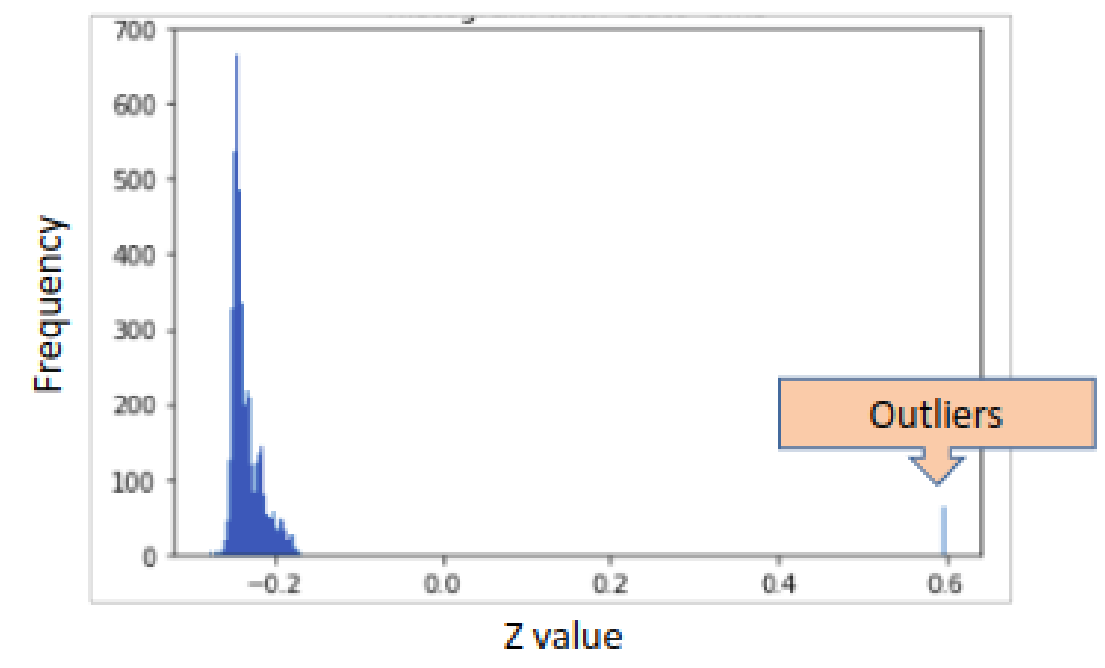




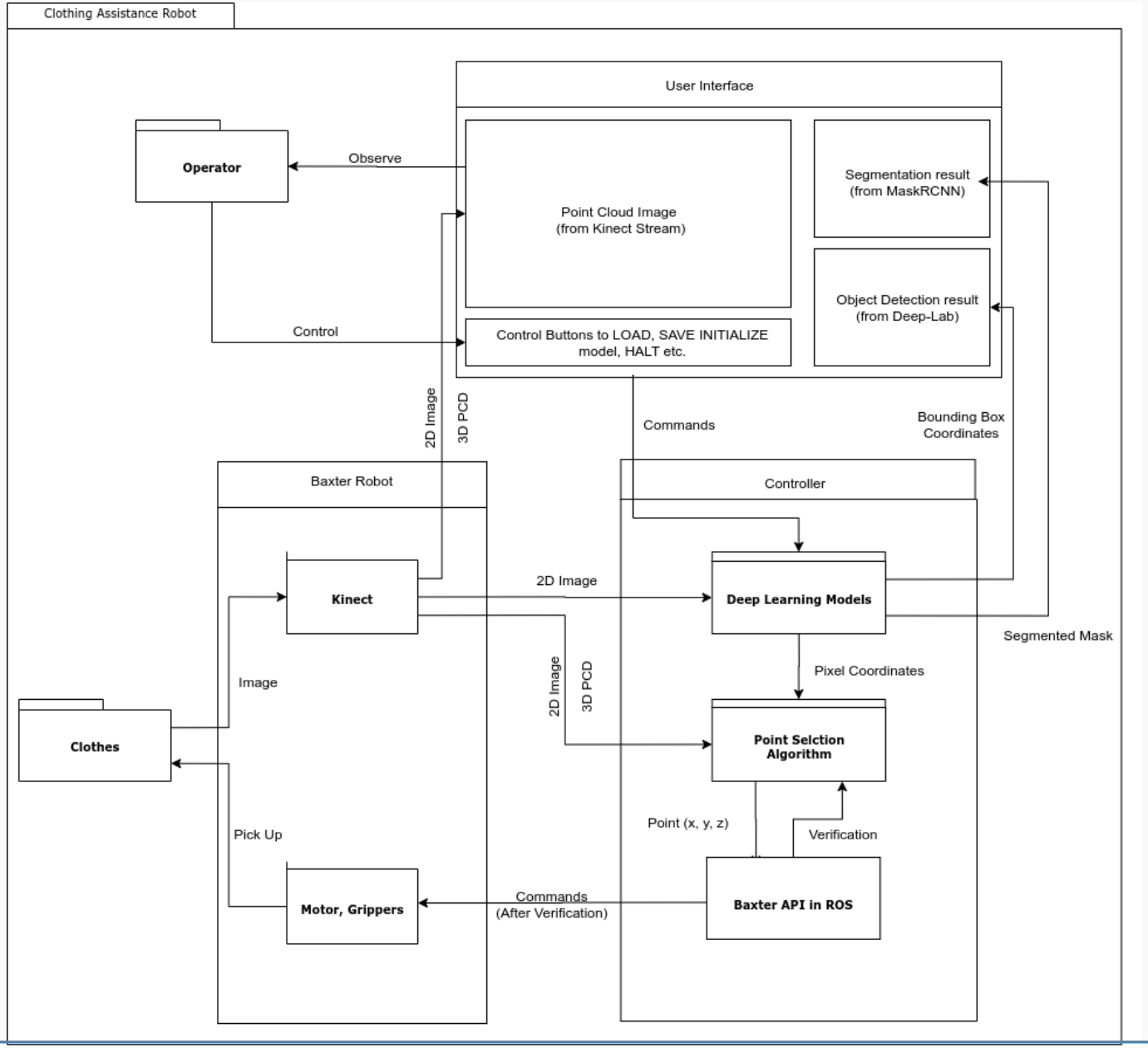
Project Details CONT.

Picking Up:

1. Due to reflective surfaces there was noise in the PCD.
2. Modeled the distribution of z-coordinates and wrote algorithm to extract the topmost cloth point.
3. The gripper hand was custom designed and 3D printed with a rough material to provide enough friction to pick up the cloth.
4. Gripper was sent to the coordinates to pick up the cloth.



System Design



Experimental Setup

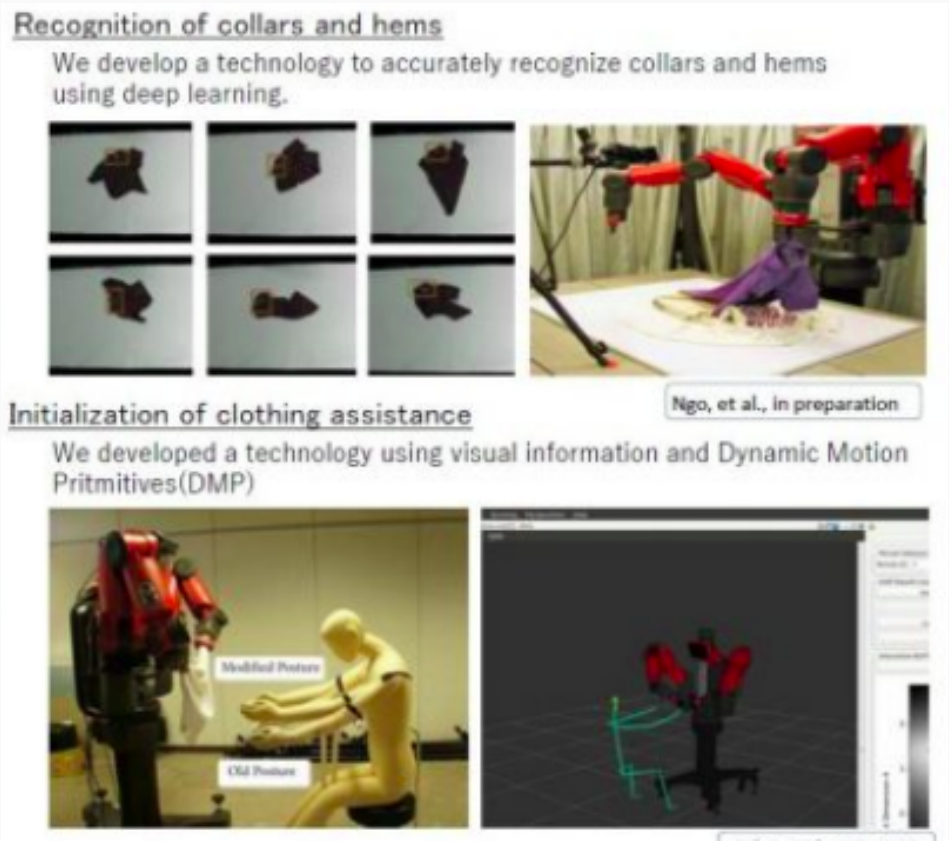
Deep Learning models trained on the Deep-Station.

- **Technical challenge**: Extremely large size of the data-set, it could not be loaded into the memory(bottleneck) at once for pre-processing by pandas in python.
- **Solution**: Efficient batch data loading function which was dynamically parallelized in scheduling jobs.

The model with pre-trained weights were loaded into memory and then trained on a general clothes dataset and then fine-tuned on a locally generated dataset from Baxter's camera. Computation was performed on GPU with PyTorch code. The final trained model was saved as a pickle.

The software interface was implemented in PyQT.

Final



This setup was then show-cased in the RoboMech 2018 exhibition in Japan. Our stall was also covered by Japanese National Television. Shibata sensei's blog [3] with images.

APEX: Adaptive Ext4 File System for Data Recoverability in Edge Devices

Motivation

- Edge computing, preferred paradigm for modern computational applications. Computational resources closer to the edge of the network (user). Results in low response time and better QoS. Application demands have been increasing but edge devices are resource constrained.
- Constraints limit the number of security layers that can be integrated without compromising performance or QoS.
- Systems are vulnerable to cyber-attacks and compromise of data. Eg. Data-breach in which hackers maliciously corrupt or delete crucial data.
- Modern viruses or trojan-horses are powerful and their nature of attacks has become very sophisticated. Existing works have focused on this aspect with limited perspective.

Previous Work

Work	Recovery specific allocation	Low Overhead				Selective files can be marked critical	Allows custom policies	For Edge/Fog/Cloud	Adaptive	User specific	Cross Platform
		Computation	Memory	Disk	File I/O						
Ulrich et al. [19]									✔		
AFS [15]	✔					✔	✔				
Hanson [20]	✔					✔			✔	✔	
Breeuwsma et al. [21]		✔									✔
Alhussein et al [22]	✔	✔	✔	✔		✔		✔			
Stoica et al. [13]		✔	✔						✔		
Huang et al. [23]	✔	✔	✔	✔	✔		✔	✔			
Alhussein et al [24]	✔			✔	✔	✔					
Baek et al. [25]		✔	✔					✔			✔
Lee et al. [26]	✔	✔	✔	✔		✔		✔			
Continella et al. [14]	✔	✔	✔	✔	✔				✔		
APEX [this work]	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

This is the full list of critical factors.
Baselines for quantitative comparison : AFS, FFS, Shield FS, ExtSFR which are as recent as Jan, 2019.

Defining factors

Define 4 factors for each block to contribute to a 'priority' function of the block (computed by linear combination of these factors) :

- History Factor (HF): Number of write and delete operations.
- Usage Factor (UF): Read/Write operations.
- Spatial Factor (SF): Characterizes spatial locality.
- Linking Factor (LF): 0(for unlinked) or 1 (for ELF, archive type files)

So the Priority function is:

$$PF = \lambda \cdot HF - \sigma \cdot UF + \rho \cdot SF + \mu \cdot LF$$

hyper-parameters

Objective Function

Allocate blocks in the decreasing order of PF. (From the exponential number of arrangements, NP hard problem). After the attack, compute the RR and Optimize the Performance.

1. **Recovery Ratio (RR):**
   - 1 complete recovery, 0 partial or no recovery — Linked files
   - $\frac{Data\ recovered\ in\ bytes}{Original\ file\ size\ in\ bytes}$ — Unlinked files
2. **Approximate Access Time (AAT):** Average time for read/write operation.

$$\underset{\lambda,\sigma,\rho,\mu}{\text{maximize}} \quad P = \alpha \frac{100 \cdot \sum_{all\ deleted\ files} RR \cdot UF}{\sum_{all\ deleted\ files} UF} - \beta \frac{\sum_{all\ current\ files} AAT}{Number\ of\ current\ files}$$

$$\text{subject to} \quad \lambda, \sigma, \rho, \mu \in [1, 10];\ \lambda, \sigma, \rho, \mu \in \mathbb{N},$$

$$Blocks\ allocated\ to\ new\ files\ in\ decreasing\ order\ of\ PF = \lambda HF - \sigma UF + \rho SF + \mu LF$$

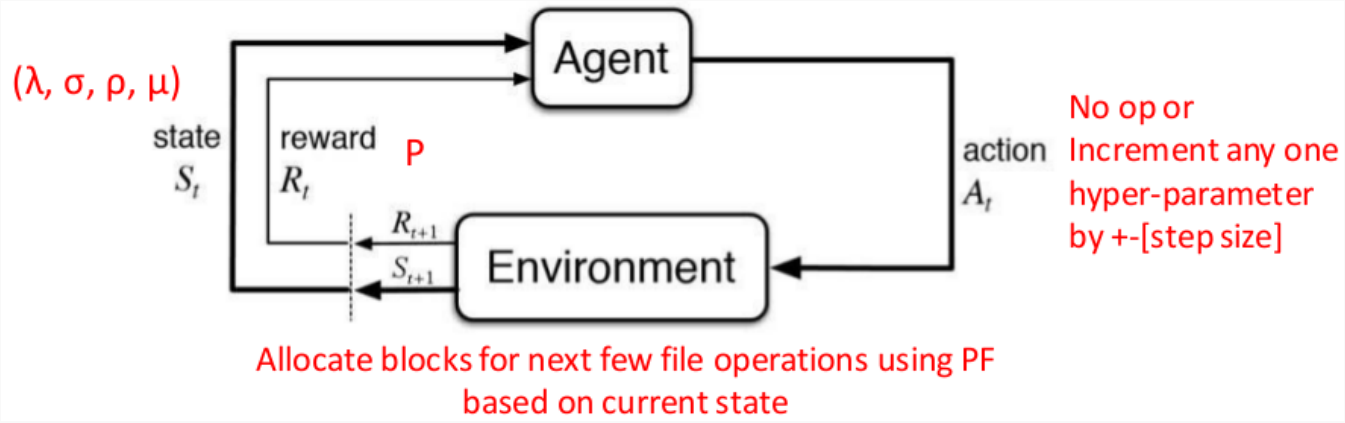
Components

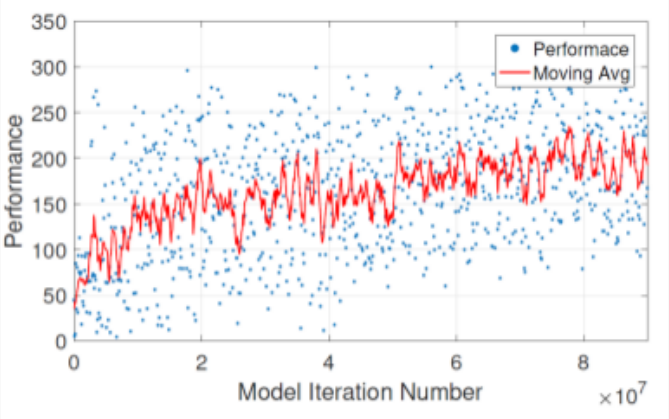
The project has 3 main components:

- File system APEX using FUSE library. Paired up with a disk simulator which simulates workloads.
- Using Q learning with ϵ-greedy approach (RL) to optimize the set of hyper parameters for maximizing recovery ratio.
- Testing the file system on Edge Node(Raspberry Pi) against CryPy malware to simulate real world scenarios.

Q Learning - RL algorithm

Used ϵ-greedy Q Learning to optimize hyper-parameters. The model used and results are shown below:



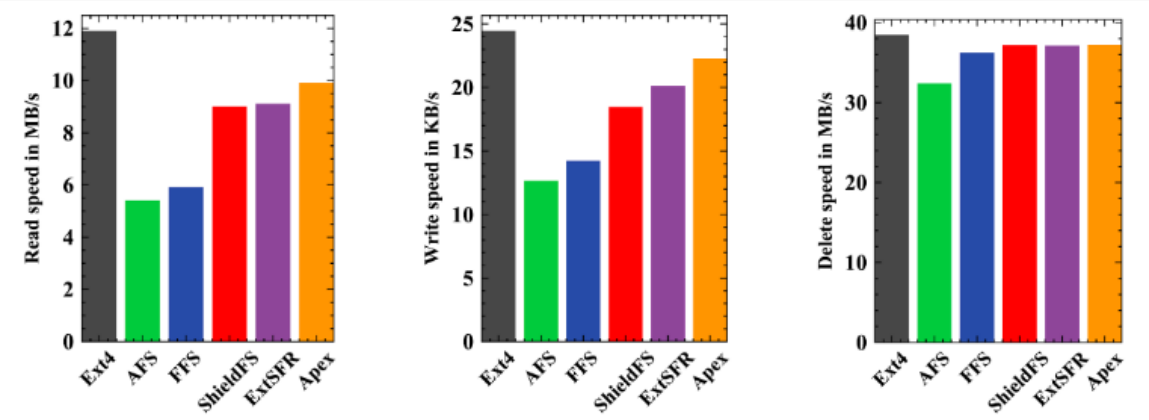


After optimization, resulting values of $\lambda, \sigma, \rho, \mu$:

$$PF = 4HF - 7UF + SF + 9LF \quad (1)$$

My Contributions and Learning

1. Implemented and tested the file system in JAVA from scratch using FUSE library, used Heap to order the blocks by Priority function.
2. Interfaced the disk simulator and RL algorithm with the file system.
3. Integrated all of these to Raspberry-Pi and analyzed the results compared to the baselines.

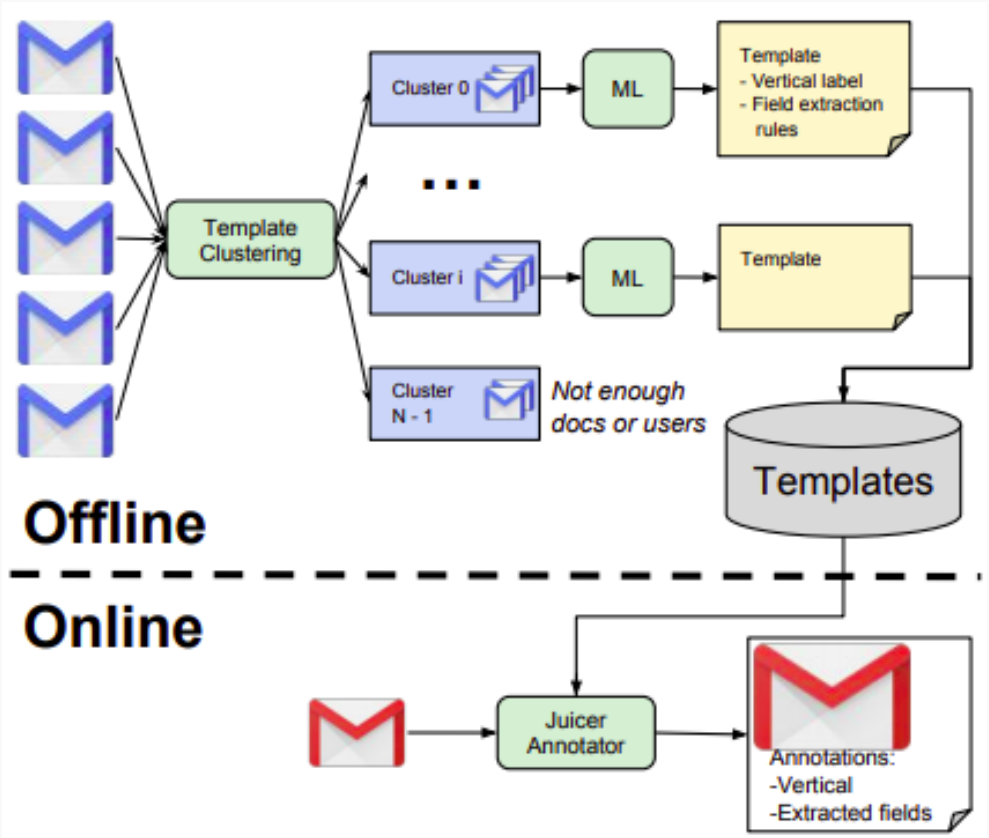


Presented at CloudCOM conference [8] in Sydney (December 2019).

Semantic Information Extraction

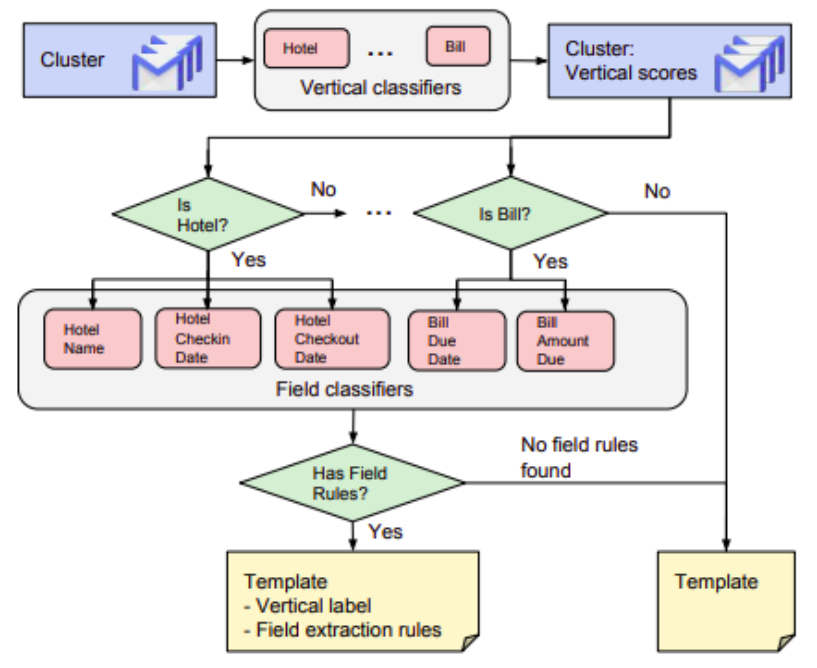
Objectives

- Semantic information extraction from email data corresponding to different verticals in Business to Consumer emails.
- To replicate Gmail's state of the art Juicer [4] system within the given constraints of data and generalizability.

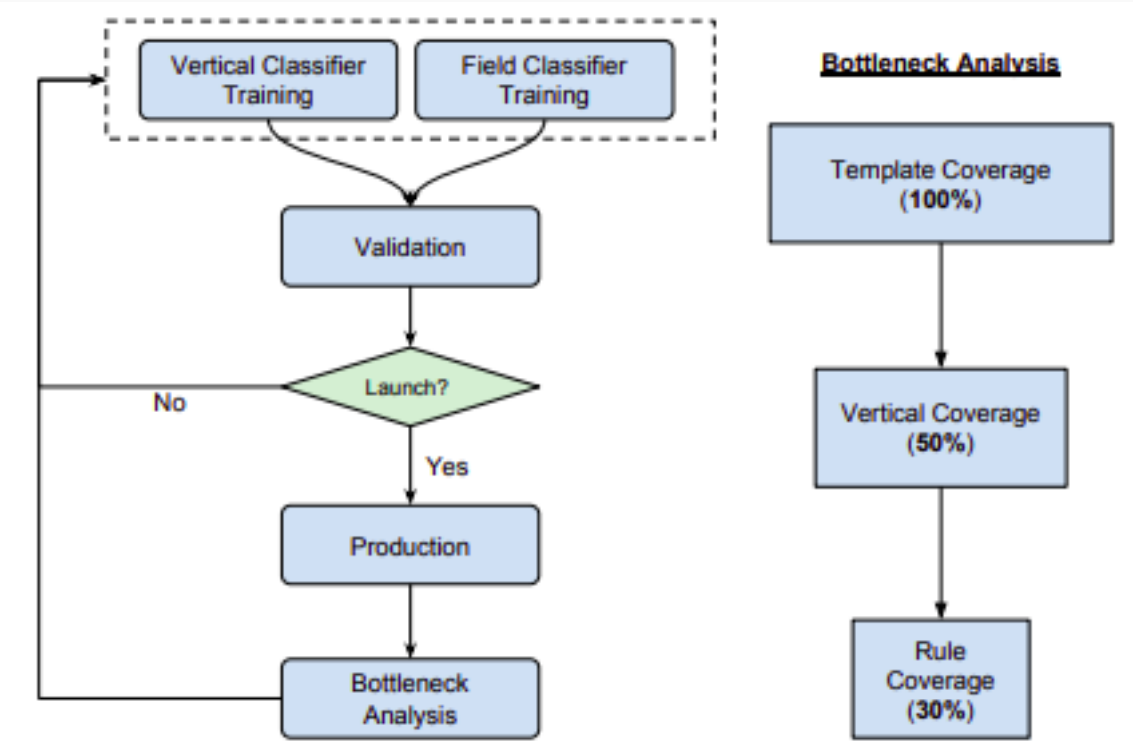


Challenges

- The Emails were in multiple languages often mixed and with different writing scripts.
- Emails had unstructured data.
- Different companies send different template emails which also changed over time.
- Preserve privacy and anonymity of the emails, and less data for training.



Details and Results



- Used Spacy, KlonPY, NLTK, OpenIE, StanfordIE to develop generic adaptive model to work for different segments.
- Generated custom post-processing heuristics for better field extraction, with semantic analysis. (Vertical specific)
- Adapted the model from Python to JAVA, to integrate with the existing project workflow of Lab.

Disjunctive Rado Number

Problem Definition: Disjunctive Rado Number

- A r-colouring of 1, . . . , N is a mapping χ : 1, . . . , N to 1, . . . , r. Given a system L of one or more linear equations, an r-colouring χ of 1, . . . , N admits a monochromatic solution to L provided there exists a solution $x_1, ..., x_k$ to L such that $\chi(x1) = \cdots = \chi(xk)$.
- The 2-colour Rado number for equation E, $Rad_2(\epsilon)$ is the least positive integer N such that any 2-colouring of 1, . . . , N admits a monochromatic solution to E.

Given two equations E_1 and E_2, the **2-color Disjunctive Rado** number for E_1 and E_2 is the least integer n, provided that it exists, such that for every coloring of the set 1, 2, . . . , n with two colors there exists a monochromatic solution to either E_1 and E_2. If no such n exists, then the disjunctive Rado number for E_1 and E_2 is infinite. Let $Rad_2(E_1, E_2)$ represent the disjunctive Rado number for the equations E_1 (eg. $x_1 + x_2 + c = x_3$) and E_2 $(x_1 + x_2 + k = x_3)$.

Objectives

- We are considering a non-homogeneous equation of the form $x_1 + ax_2 - x_3 = c$ for different ranges of a and c.
- Generalize the results to $\sum_{i=1}^{m-2} x_i + ax_{m-1} - x_m = c$ for all choices of a and c, $\forall m \geq 4$

The search space for Rado number a particular value of a, c and m **explodes exponentially**. For a rado number R, in theory we would have to check for 2^R colourings, every one of them should have a solution. If any one of them does not have a solution for that N, then Rado number is more than N, and if all of them have a monochromatic solution in the system of equations, then we have to verify if that is the minimum such N.

Challenges

- Using the system of equations we have to approximate the upper and lower bound of R so that we can have a mathematically exact formulation for this function of a, c & m.
- Since the search space is large, an effective search algorithm is needed, to trim it and verify for every coloring by using some key Numbers in the range. (Decided by the system of equations).

Ex. For the equations: $\sum_{i=1}^{m-2} x_i + ax_{m-1} - x_m = c1'$ and $\sum_{i=1}^{m-2} x_i + ax_{m-1} - x_m = c2'$, we can form a definitive set of colored numbers:

0	**1**
0	$-c_1'$
$-(a'+2)c_1'$	$-c_2'$
	$-(a'+1)c_1'$

Details and Results

- Developed an efficient coloring method, using multiple queues, priority queue, to estimate the lower and upper bounds of the system very fast, and then calculating the Rado number.
- Generating adversarial cases of collision for a given Rado number of verifying that it's the correct for that set of parameters.
- By analysing a large set of parameters and their respective Rado numbers, fit the curve in different ranges of a, c and m. And estimate the function.

Theorem 4. *Let a, m be integers of the same parity, with $a \geq 3$. For*

$$c_2 \leq c_1 \begin{cases} \leq -a(a-3)/2 & \text{if } m = 3; \\ < -(a+m-3)(a-2) & \text{if } m \geq 4, \end{cases}$$

we have

$$\mathscr{R}(c_1, c_2) = \begin{cases} (a'+3)(a'-c_1)+1 & \text{if } c_1 - a' \leq c_2 \leq c_1; \\ (a'+2)(a'-c_1)+1 & \text{if } (a'+2)c_1 - a'(a'+1) \leq c_2 < c_1 - a'; \\ (a'-c_2)+1 & \text{if } (a'+3)c_1 - a'(a'+2) < c_2 < (a'+2)c_1 - a'(a'+1) \\ (a'+3)(a'-c_1)+1 & \text{if } c_2 \leq (a'+3)c_1 - a'(a'+2), \end{cases}$$

where $a' = a + m - 3$.

References

References

1. HAFY Robot blog.
2. MaskRCNN: State of the art Segmentation Network.
3. Shibata Blog Hibikino Station.
4. Juicer: Gmail Information Extraction system.
5. CLOCK algorithm: FreeBSD Documentation.
6. TinyLFU: A Highly Efficient Cache Admission policy.
7. Battle of Schedulers: Linux vs FreeBSD, USENIX ATC'18.
8. Arxiv link to paper published in CloudCOM.

Thank You

Questions?