HARSHA RASTOGI

+1 412.499.1634 | harshar@cmu.edu | rastogiharsha.com

SUMMARY

Back end software engineer who is keen about tackling the challenges in building large scale, fault tolerant distributed applications and big data engineering solutions. Quick learner with a strong background in storage, distributed systems and applied machine learning

EDUCATION

Master of Science
2014 - 2015

Carnegie Mellon University (CMU), Pittsburgh, USA
Electrical and Computer Engineering, 3.9/4.0, Eta Kappa Nu Member

Bachelor of Engineering
2009 - 2013

Birla Institute of Technology and Science (BITS), Pilani, India
Electrical and Electronics Engineering, 9.55/10

SKILLS

Courses	Mobile and Pervasive Computing*, Distributed Systems, Storage Systems, Machine Learning, Computer Systems, Computer Architecture, Digital Electronics and Computer Organization
Languages	C, Java, Python, CSS3, HTML5, SQL, x86, ARM, Node.js*, JavaScript*
Tools	Cloudlet**,** Django, Kaggle, Eclipse, GDB, QEMU, MATLAB, Ubuntu 14.04, Cadence Palladium

EXPERIENCE

CMU, Pittsburgh, USA
Teaching Assistant
Sep ’15 - Present

Responsible for designing, pretesting labs and supervising students for the course *Storage Systems* of 100 students *(C++, C)*

SanDisk, Milpitas, USA
Software Engineer Intern
May’15 - Aug’15

Enhanced code coverage by developing a fault injection framework which supports run time control and system fault simulation for FlashSoft server caching solution *(C, Python)*

Broadcom, Bangalore, India
Hardware Design Engineer
Aug’13 - Jul’14

Enhanced mobile baseband SoC performance by identifying logical and functional flaws in the design and power management code *(C, Cadence Palladium)*

CMU, Pittsburgh, USA
Summer Research Intern
May’12 - Jul’12

Under Professor Onur Multu (*SAFARI*), worked on designing efficient memory scheduling algorithms at the memory controller in a multi core system. Increased system throughput by 1% and fairness by 5.9%. Also, performed exploratory work on *Blacklisting Memory Scheduler* presented at 32nd IEEE, ICCD, 2014

PROJECTS

Distributed Systems
(*C, Java)*

Scalable Web Service (Load Balancer) - Designed a simulated multi-tier web hosting service (online store) to maximize total revenue by ensuring short client response times, while minimizing running cost (number of VMs running). Service scales out dynamically and has a caching tier to reduce storage latency

Distributed Proxy Caching Servers (CDN) - Designed a multiple proxy system which performs whole file caching in a client-server RPC system. Proxy servers implement LRU eviction policy to reduce latency

Remote Procedure Call - Build an RPC system which supports transparent client file operations (open, read write, lseek, lstat etc.) at the server

Storage Systems
(*C, FUSE)*

Hybrid File System - Designed a hybrid file system based on FUSE which uses local SSD and amazon S3 servers for storage. It supports caching, segment level de-duplication and snapshots

File System Checker - Designed a fsck utility to identify, parse, read, write on-disk image of ext2 file system

Flash Translation Layer (FTL) * - Designing an FTL on an emulated solid state drive (SSD) which supports address translation, wear leveling and garbage collection

Machine Learning
(*Kaggle, MATLAB)*

Designed an image classifier to perform image classification task on CIFAR-10 data set. Configured a voting algorithm (Accuracy 62%) that uses results of the best supervised classifiers to classify the image

Mobile Computing*
(Java, JavaScript, Cloudlet)

Designing a mobile application which implements interactive hyper-lapse system. It uses google street view as data source and Intel cloudlets for reduced latency and data caching

Systems Programming
(C, ARM, QEMU)

Memory Allocator - Designed a custom version of C’s standard library function malloc(). Stored the free blocks in a segregated list structure ensuring a good balance between memory utilization and performance

Multi-threaded Proxy - Designed a multi threaded HTTP proxy server that supports caching of web objects and concurrent client requests

Linux Shell - Designed an interactive command line interpreter in C supporting job control, signal handling and I/O redirection

Kernel Programming - Designed a single-task mini-kernel on QEMU that supports read/write/exit system calls and user space applications

*ongoing