Arindam Das

(+91) 98316 57983 • dasarindam.mails@gmail.com • Baidyabati, India • github.com/arindas

Description

I specialize in distributed systems, deep learning inference and AI SaaS at scale.

My expertise enables me to architect and implement cloud-native software services, for different domains. I primarily have experience in medical imaging and diagnosis, real-time document processing and business inventory management.

Technical Skills

- **Languages**: C, C++, Java, Python, Golang, Rust, Javascript, Typescript, SQL
- **Frameworks and Libraries**: Django, Tensorflow, Pytorch, React, Glommio
- **Tools**: Git, Vim, Neovim, Awk, Sed
- **DevOps**: Linux, Nginx, Docker, Bash, Zsh, Github Actions, Gitlab CI, Terraform
- **Databases**: PostgreSQL, GCP Cloud Firestore, SQLite
- **Cloud**: AWS{S3, EC2, Lightsail}, Firebase, GCP{Instances, PubSub, Cloud Storage}, Azure{Instances, Blob Storage, Container Apps}

Soft Skills

Agile Software Development, Requirement Analysis, System Design, Technical Content Delivery

Experience

MLOps Engineer, Medical Imaging AI Services, Claritas Healthtech (06/2020 - Present)

Responsibilities

- Provisioning infrastructure for researchers to train and experiment with deep learning models
- Developing distributed deep learning inference services and corresponding client web applications
- Deploying and maintaining said applications on the Google Cloud Platform

Projects

- **Cloud based file storage solution built using Google Cloud Storage**
    - Supports bucket creation, bucket level user access authorization and create-read-update operations.
    - Backend is a golang web service built with, google-cloud-sdk, Cloud Firestore database and Firebase Authentication
    - Frontend is a React SPA application
    - *This solution enabled us to collaborate on sensitive datasets with over 10 different medical institutions across UK, Singapore and Europe without providing access to our GCP infrastructure.*

- **Distributed deep-learning based diagnosis on medical images for a variety of diseases**
    - Designed as an event-driven suite of microservices, in golang and python.
    - We employ golang for the web serving infrastructure and python for inference.
    - Google Cloud PubSub is used as the messaging layer.
    - Capable of integrating with dedicated inference servers like Torchserve, Nvidia Triton and Tensorflow Serving
    - We provide a React Dashboard for: Visualizing medical images, Requesting AI diagnosis, Tracking inference status and Viewing AI Medical Diagnosis reports
    - *Reduced turnaround time for a new disease prediction service deployment by 10x, along with improved audit record keeping of all predicted reports.*
- **Dedicated Inference services for research Proof-of-concepts**
    - Implemented as a Django user facing application and a inference server.
    - The Django application behaves as a sidecar for the inference server
    - The inference server is either implemented as a FastAPI service or a dedicated Torchserve server based on requirements.
    - I was responsible for productionizing over 15 different deep-learning models split across 6 different web services in a span of 2 years.

Solution Architect, DeepWrex Technologies (04/2018 - 06/2020)

Responsibilities

- Architect cloud based solutions for machine learning software services.
- Assist researchers in implementing deep learning research papers.
- Iterating from research PoC to production.

Projects

- **Real-time named entity recognition system for medical reports.**
    - In house, economic alternative to AWS Medical Comprehend which didn't exist at the time.
    - Implemented as a event-driven suite of C++ microservices, with intermediate data storage on AWS S3.
    - We used Apache Kafka (using rdkafka) as the messaging layer.
    - This solution enabled our consulting partner to make medical reports more accessible to patients.
- **Black and white image colourization system**
    - We implemented the Instance aware image colourization paper which was the state-of-the-art deep learning based image colourization paper at the time.
    - Used Torchserve inference server for scalable GPU inference and FastAPI for user facing web services.
    - Deployed on AWS on a GPU enabled EC2 instance.
    - We also developed and launched a Flutter Client Application to Google Play Store with over 50 downloads in the first month.

Satellite Onboard Computer RTOS Research Student, KIITSAT (04/2018 - 05/2019)

- Designed a shell capable of spawning programs and organizing pipes within 200 lines of C
- Developed a minimal kernel for armv6 (on Raspberry Pi 3) with basic memory management and TTL based I/O Support
- Trained a team of 8 fellow researchers on OS concepts and implementation details.
- Collaborated with 2 different student engineering departments for integrating with different Satellite subsystems

VR 3d Game Development Instructor Intern, CampK12 (03/2020 - 06/2020)

Responsibilities

- Teaching K12 students about game development which entailed:
  - Basic Programming using Javascript (Control flow, operators, functions, callbacks etc.)
  - Elementary Trigonometry for animating movements of game objects

I guided 5 different students through 8 game-dev projects over a period of 3 months. At the end of the course the students were able to independently implement features and explore new concepts.

Education

- ### B.Tech in Computer Science and Engineering

  KIIT University (06/2017 - 06/2021), **CGPA**: 9.44 / 10, **SGPA 8th Semester**: 9.69 / 10

Open Source Projects

- laminarmq: A scalable, distributed message queue powered by a segmented, partitioned, replicated and immutable log. It is a resource efficient alternative to Apache Kafka.
- generational-lru: A generational arena based LRU Cache implementation in 100% safe rust. All allocations are based off a vector.
- sangfroid: A load-balanced thread pool implemented in Rust using only the standard library. Worker threads are managed with binary heap and are prioritized by the number of pending jobs.
- quartz: A shared memory parallelized ray tracer using OpenMP. *Speciality:* Non recursive. Traditionally ray tracers are tail recursive. Image formats supported: PPM
- mac-on-linux-with-qemu: Runs MacOS on Linux with the QEMU KVM Hypervisor with some python utility scripts for downloading the disk images and shell scripts for starting QEMU.
- riakv: An append-only key-value store, with checksum based record validation. It support both in-memory usage and persisting records to the disk.
- elevate: Barebones zero dependency HTTP File upload server in Go.
- bheap: A Rust generic binary max heap implementation. It allows dynamic definition of the comparison function for the underlying domain at runtime.