6.S194 Open Source Entrepreneurship

Lecture 1

Staff

Saman Amarasinghe
Professor EECS/CSAIL
saman@mit.edu
32-G744

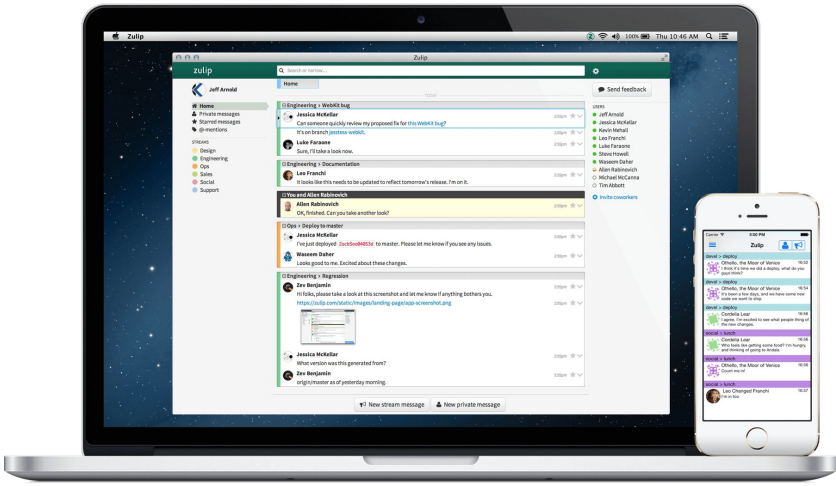
Jeff Bosboom
PhD student CSAIL
jbosboom@csail.mit.edu

Nick Meyer
Entrepreneur in Residence, Trust Center
nmeyer@mit.edu

Projects

Zulip

Group chat and private messaging built around message streams (threads)



Server built on Python, Django

- Apps
  - React Native-based mobile app
  - Electron-based desktop app
  - Snipe terminal IM client
- Interactive chat bot framework
- Separating Django libraries
- Federating Zulip with Jabber or IRC
- Dockerizing Zulip, deployment path
- Voice and video chat
- Real-time collaborative editing
- On-boarding path for open source projects

Attribute-based encryption library

Attribute-based encryption allows users to selectively decrypt certain ciphertexts based on access control permissions.

Useful in databases where each user has one key that can decrypt different (overlapping) slices of the database.

Existing libraries are outdated due to advances in cryptography and in hardware.

Project is to implement a modern attribute-based encryption library for the research community and benchmark it against previous implementations.

Re-encryption library

Encrypted data stored on a server must be re-encrypted if the key is compromised or to revoke access from another user.

Downloading the data, decrypting, and re-encrypting it is expensive. Decrypting the data on the server exposes it.

Proxy re-encryption schemes allow the server to re-encrypt the data without first decrypting it.

Some basic libraries have been implemented, but they have not been open sourced and have not been thoroughly tested. The goal of this project is to open source this library so that others can use it. There is already interest from industry in having such an open source library.

Function secret sharing library

Function Secret Sharing (FSS) is a recent cryptographic primitive that allows a client to divide a function f into function shares f_1, f_2,...,f_k so that multiple parties can help evaluate f without learning the input.

This is a powerful technique that allows users to do private querying and anonymous community with very little bandwidth overhead.

Currently, there are no good open source libraries to do this. The goal of the project would be to create a library with a clean API so that other projects can build up on this. There is a lot of interest around this in the research community, and it would be heavily used.

Aurum

A system for data discovery at scale.

Java-based profiler summarizes terabytes of data into profiles that contain signatures that represent that data

Python graph builder finds relations between profiler signatures using minhash signatures and locality-sensitive hashing

Discovery layer named SRQL allows users to declare discovery queries

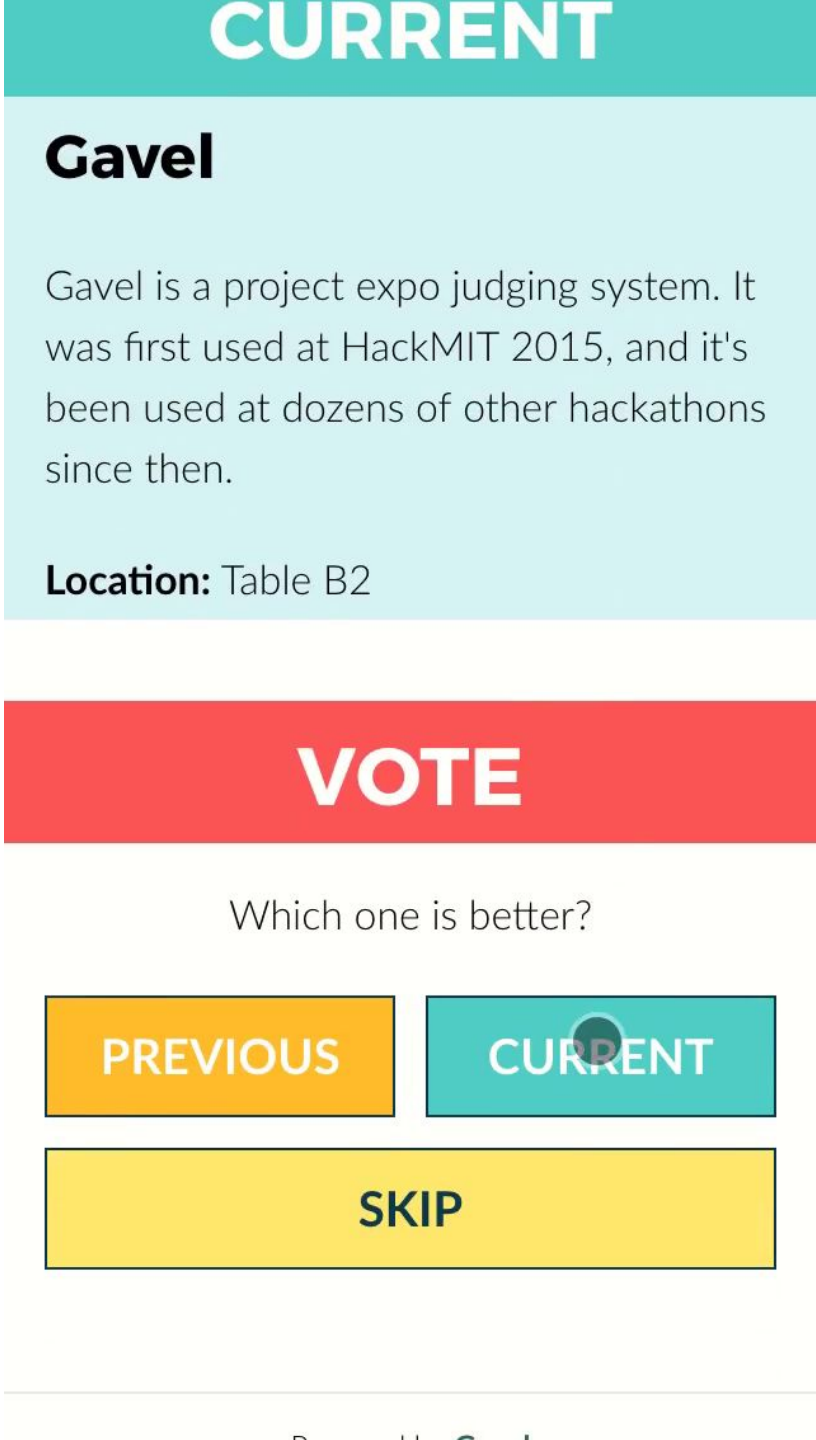
In use by companies, including a big pharma company and a data integration company, and is to be deployed in New York City’s analytics infrastructure.

Gavel

Project expo/hackathon judging system used at HackMIT

Based on pairwise comparisons

Python 3, Flask, NumPy, SciPy



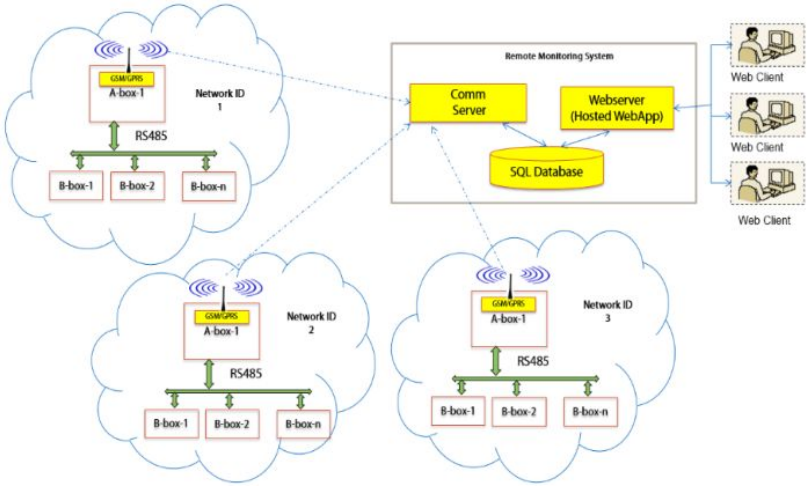
uLink - smart microgrids for affordable electricity access

uLink is a system for building smart microgrids for electrification of rural/undeveloped areas.

uLink manages disparate power sources (solar, diesel generators, hydro turbines) and sinks.

uLink devices connect via a wired MODBUS network and to the Internet via GSM.

uLink learns demand and generation profiles using unsupervised learning, then uses these profiles to determine a pricing strategy that matches supply and demand.



MOOCdb

The MOOCdb project aims to brings together education researchers, computer science researchers, machine learning researchers, technologists, database and big data experts to advance MOOC data science.

The project founded at MIT includes a platform agnostic functional data model for data exhaust from MOOCs, a collaborative-open source-open access data visualization framework, a crowd sourced knowledge discovery framework and a privacy preserving software framework.

The team is currently working to release a number of these tools and frameworks as open source.

Minimalist Isopycnal Model

Oceanographers use numerical models to simulate the ocean and its effects on climate. Most researchers write their own simple models, but don't publish the source code, making their results unreproducible.

MIM is a simple ocean model written in Fortran 90 meant to become a basis for other researchers.

MIM's source code comes with a couple of examples, but no tests and the documentation is incomplete. The model can be extended to work in periodic domains and its performance could be improved.

TACO: THE TENSOR ALGEBRA COMPILER

Linear and Tensor algebra are the building blocks of the modern optimization, simulation, machine learning and data analytics applications. Hundreds of libraries exist for linear algebra and recently libraries like Google's TensorFlow provides dense tensor algebra, but support for sparse tensor algebra is lacking.

The Tensor Algebra Compiler (taco) makes fast and portable sparse and dense tensor and linear algebra possible.

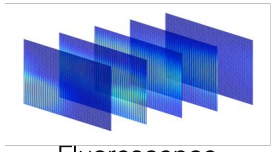
Taco is written in C++ and promises unprecedented flexibility, performance and portability. With your help we will make taco the default library for these domains.

NETFLIX

Movie ratings

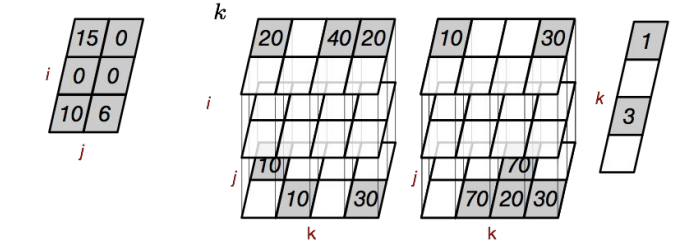
facebook

Social interactions



Fluorescence spectroscopy

$$A_{ij} = \sum_k \left(B_{ijk} + D_{ijk} \right) * c_k$$



(dense, dense) (sparse,sparse,sparse) (sparse,sparse,sparse) (sparse)

CliNER

CliNER is a natural language processing system for named entity recognition in the clinical text of electronic health records. CliNER is designed to identify clinically-relevant entities mentioned in a clinical narrative, such as diseases/disorders, signs/symptoms, medications, procedures, etc.

Its role as a component in clinical pipelines lends well to user-facing improvements, such as API development (currently it is primarily used as a command-line tool), identification of the best packaging option (currently several are provided), and charting a roadmap for folding in improvements (currently old methods are removed to introduce new methods).

CliNER is written in Python.

Tipsy

Tipsy is a Chrome extension for microdonations to support online content. Participating sites declare a PayPal or Dwolla account that accepts donations. Tipsy (privately) tracks the time spent on participating sites and reminds the user to donate at configurable intervals.

Tipsy is used by a couple of publishers, but further evangelism is necessary.

Other projects include adding Bitcoin support to Tipsy, finding ways to support creative commons content that can be embedded in many pages, and factoring out the history-tracking component so it can be used by other tools.

NB Classroom Discussion/Annotation Forum

NB is a system for online discussion of course lecture notes, videos and other material in the margins of that material.

NB is being used by ten thousand students at over 100 universities around the world to produce over 1 million comments.

NB is a an old crufty system (Django back-end + Javascript front end) that has grown haphazardly over many years and needs significant redesign and re-engineering to use modern themes, components, libraries, and UI designs.

For students interested in online education and research, NB provides a platform with real users and data for research about how online discussion can help people learn. There's a broad collection of features that have been requested by

ModelDB

ModelDB is an end-to-end system to track machine learning models as they are built, store models and metadata in a centralized fashion, and capture metadata and metrics to support query, analysis and reuse. It enables ML teams to version their models, make them reproducible and easy to share, compare and analyze.

ModelDB is currently being tested at several companies.

Some potential projects:

- building a ModelDB client for a new language or environment (e.g. R, Lua, Tensorflow)
- adding advanced features to ModelDB depending on feature requests and new requirements (e.g. in Java, Scala, Python)
- adding visualization capabilities to the frontend to support more flexible querying of data
- supporting online model updates.

Organization

Feb 7	Overview	
Feb 9	Goal-setting; Git and collaboration	
Feb 14	End user profile	Mentor interview and research notes
Feb 16	Life cycle use case, high-level specification	
Feb 21	(Monday schedule)	User profiles; identify 2-6 potential users
Feb 23	Studio	
Feb 28	User feedback	Full use cases and high-level spec
Mar 2	Studio	
Mar 7	Development methodologies, minimum viable product	User interview results
Mar 9	Studio	
Mar 14	Documentation	MVP, stretch goals
Mar 16	Studio	
Mar 21	Project promotion	Documentation plan; some exemplars
Mar 23	Studio	Software report
Mar 28	(Spring Break)	
Mar 30	(Spring Break)	

Apr 4	Customer feedback/testing	Promotion plan; beginnings of promotional materials for the first actual users
Apr 6	Studio	
Apr 11	Community strategy	Collated customer feedback; updates to MVP or post-MVP plan
Apr 13	Studio	
Apr 18	(Patriots' Day)	
Apr 20	Studio	
Apr 25	Software testing	Community strategy plan
Apr 27	Studio	
May 2	Open source business models	Test plan
May 4	Studio	
May 9	Licencing	
May 11	Open source panel	Final promotional material and code going online if not already; software report
May 16	Student presentations: project pitch	5 minute pitch
May 18	(no class)	Roadmap to what you did (writeup)

Grading

Entrepreneurship	40%
Engineering management	14%
Software design, development, implementation	36%
Class participation	10%