# CHORD-BASED USER SERVICE


ÁLVARO ALBERO GRAN

TYLER JONES

SEAN MCBRIDE

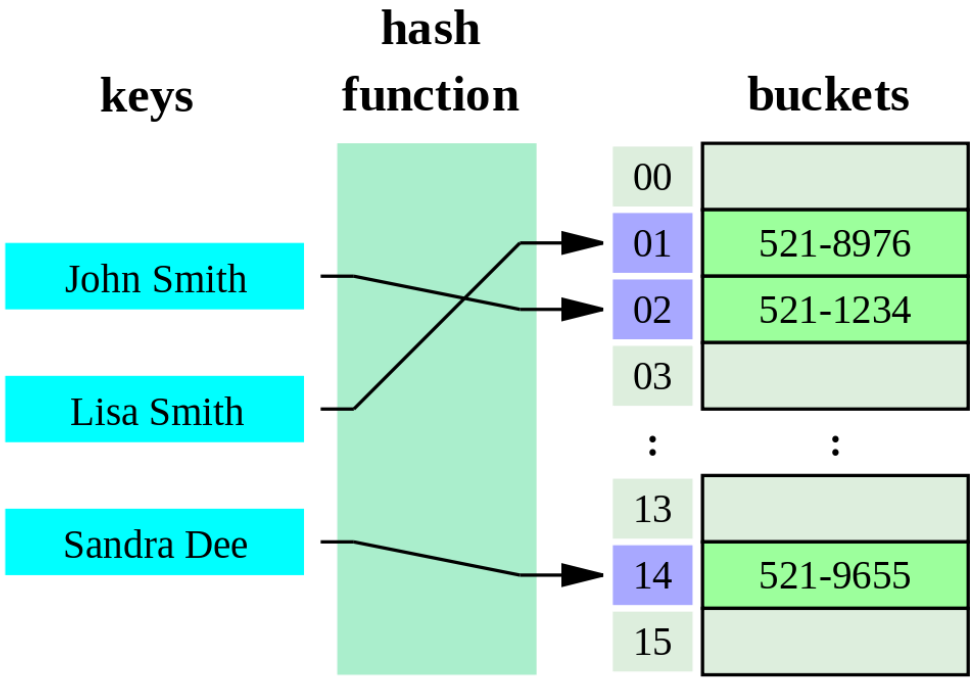
NAIM MERHEB


CSCI 6421 DISTRIBUTED SYSTEMS

PROF. ROOZBEH HAGHNAZAR KOOCHAKSARAEI

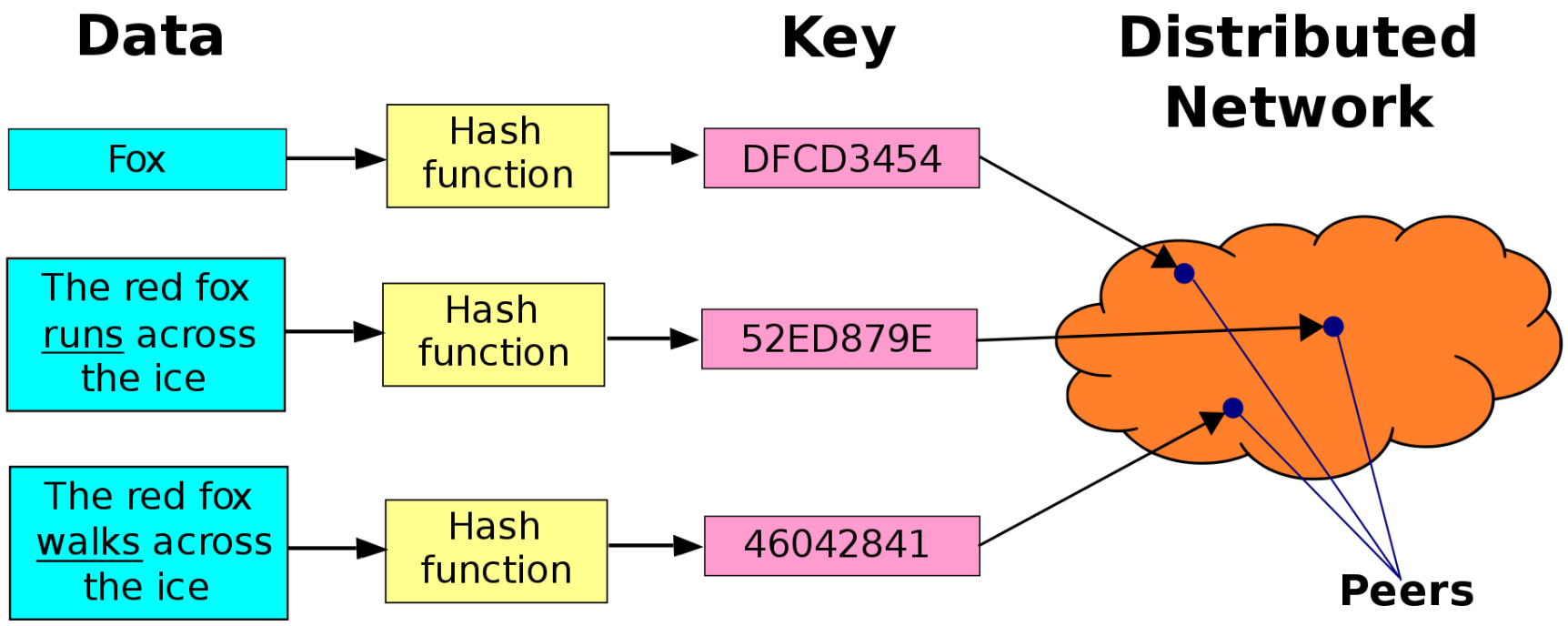
GEORGE WASHINGTON UNIVERSITY

# HASH TABLE




Source https://en.wikipedia.org/wiki/Distributed_hash_table

# DISTRIBUTED HASH TABLE




Source https://en.wikipedia.org/wiki/Distributed_hash_table

# CHORD OVERVIEW

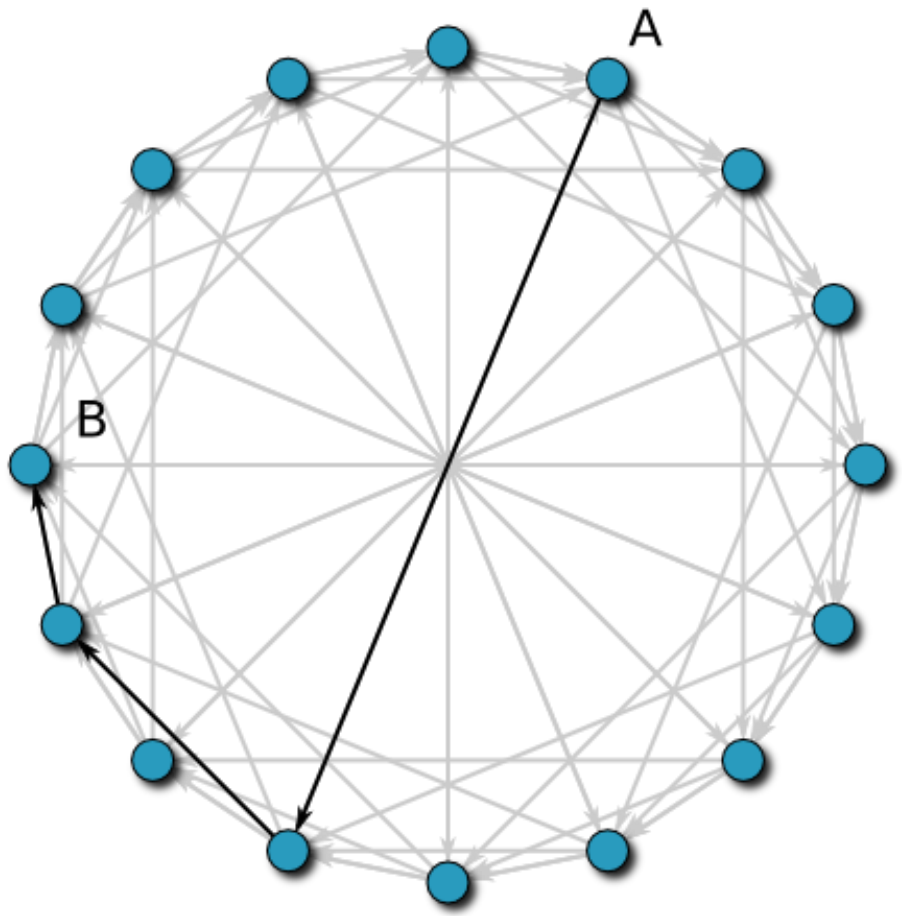
A peer-to-peer distributed hash table.

Developed by Ion Stoica, Robert Morris, David Karger, Kaashoek, and Hari Balakrishnan from MIT

Keys and IP addresses are hashed into a uniform identifier space (typically using SHA-1)

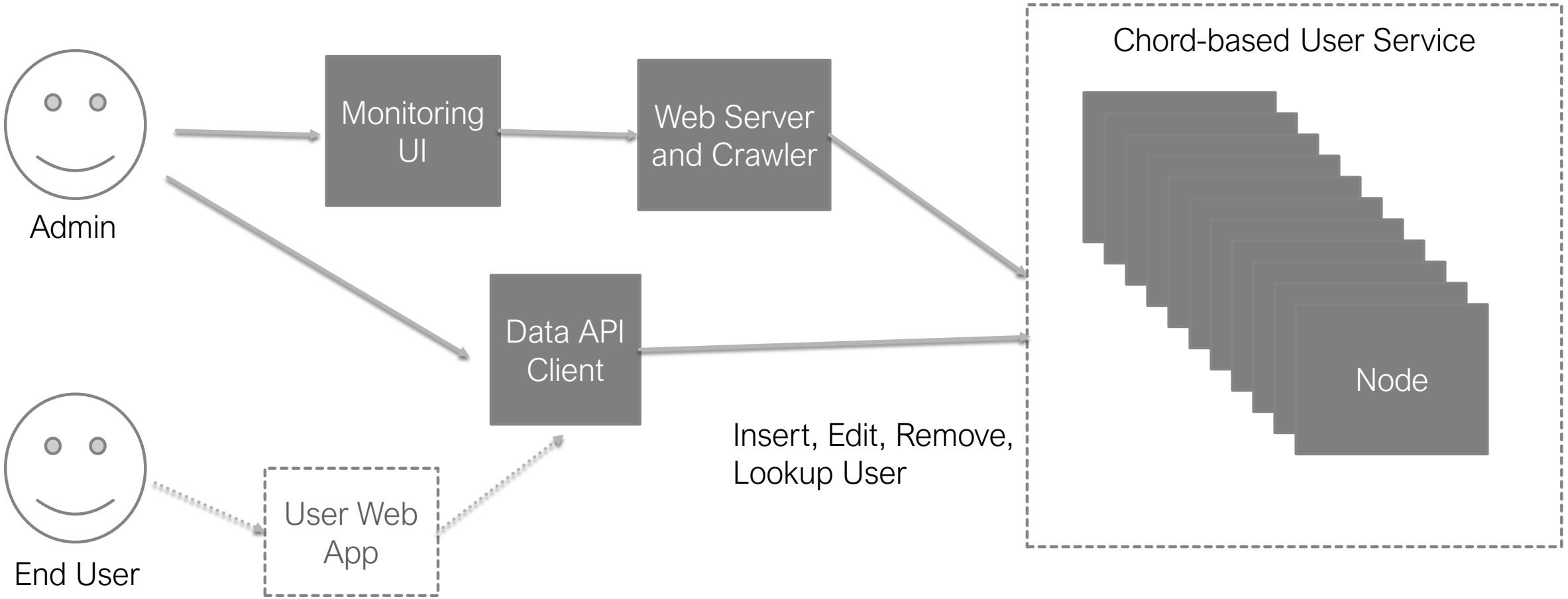
The keys of the distributed hash table are stored on their closest successor node.

Forms a logical ring with “finger tables” to reduce access time

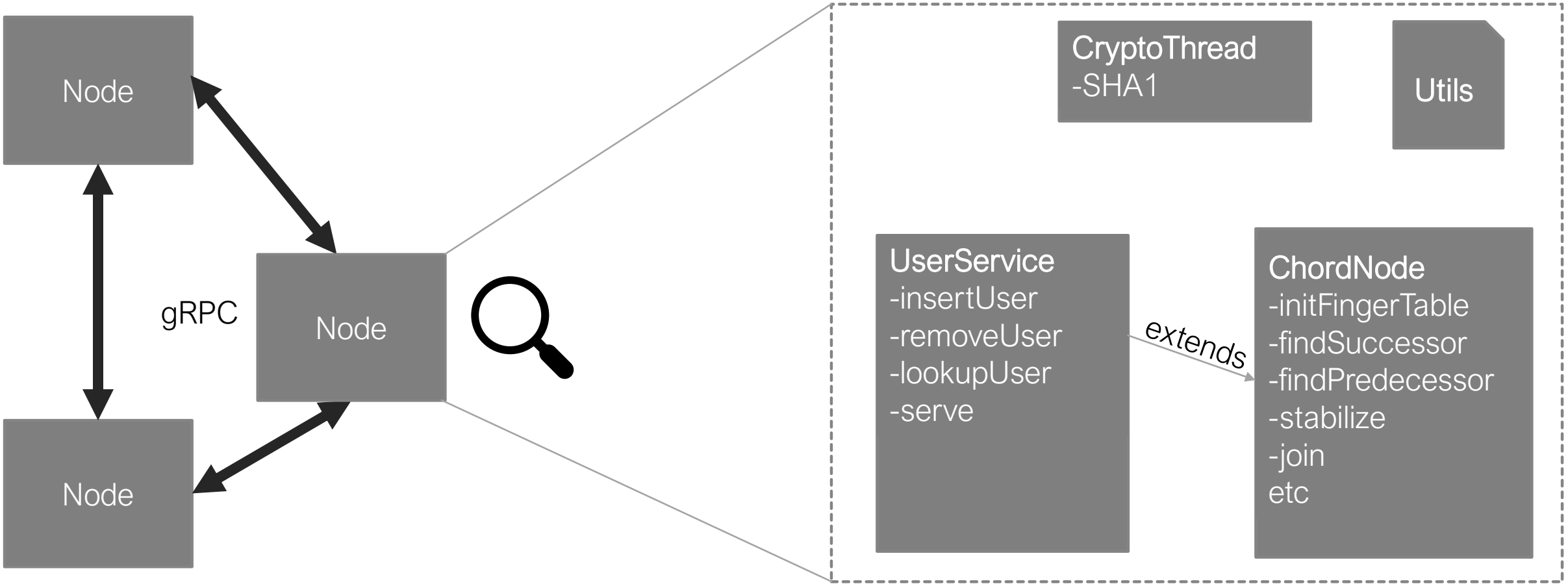



Source: https://en.wikipedia.org/wiki/Chord_(peer-to-peer)

# SYSTEM ARCHITECTURE



# CHORD NODE ARCHITECTURE



# DEMO

1. USING DATA API
2. LARGE CLUSTER

# AGENDA

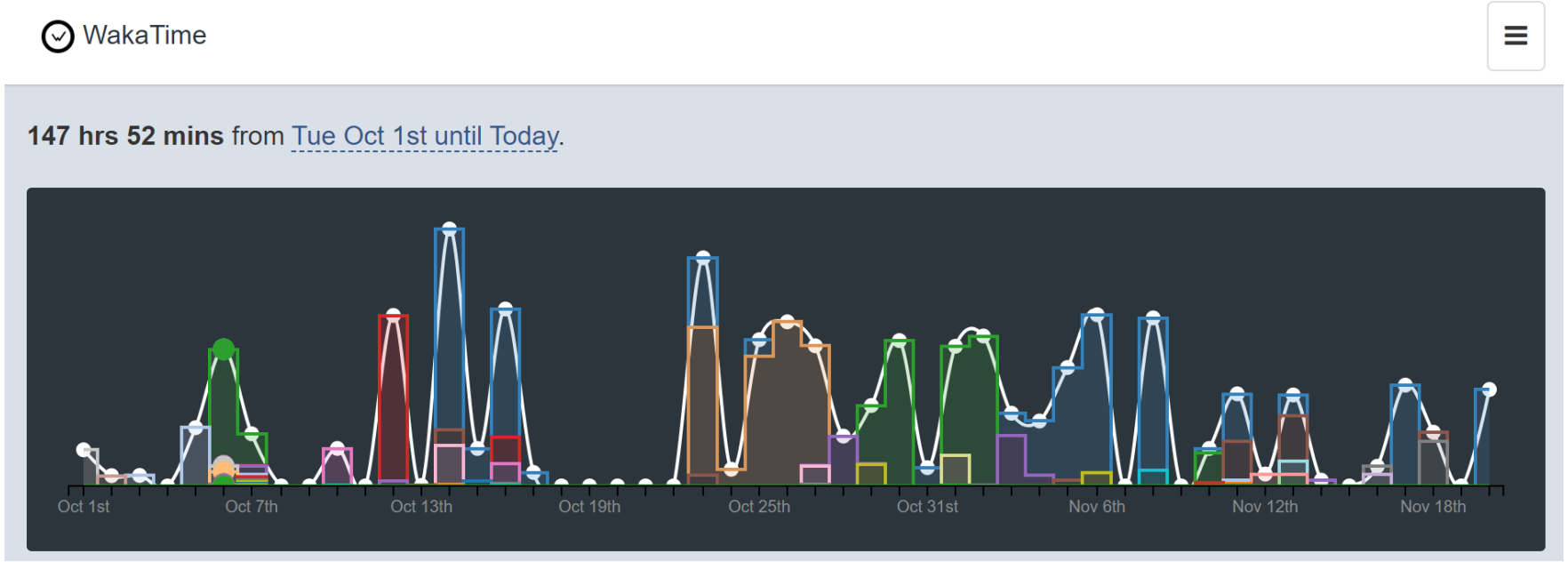
- Methodology
  - “Scrum Light” (with a bit of XP)
  - Git and GitHub
  - Mob / Pair Programming via Visual Studio Code “Live Share”
- Software Components
  - VS Code
  - Node.js, our “least common denominator” language
  - gRPC, a modern and popular RPC framework
  - Docker
  - Azure Container Instances
- Architecture
- Future Tasks
- Demo

# METHODOLOGY: “SCRUM LIGHT” (WITH A BIT OF XP)

- Kickoff meeting to fill out the backlog, using GitHub issues
- Kanban board, using GitHub Projects, to track progress
- “Asynchronous Standup” on Slack answering the following:
  - What did you do yesterday?
  - What are you doing today?
  - Do you have any blockers?
- We using both pair programming and mob programming
  - One person is the “driver.” One or more people are the “navigators.”
- GitHub Flow
  - Work done in features branches.
  - Mandatory Reviews via GitHub Pull Requests before merging to master

# PROBLEMS

- Agile assumes that everyone is working on the same thing at the same time
- Academic Semesters are “event-driven” and bursty, as below
- We have different classes, so our bursts often don’t overlap
- Because it is hard to quick Pull Requests reviewed quickly, this increases the temptation to have lump unrelated work together



# VS CODE

- Shared configuration and extensions, checked into the git repo
- Integrated Pair Programming Tooling
- Interactive Debugger

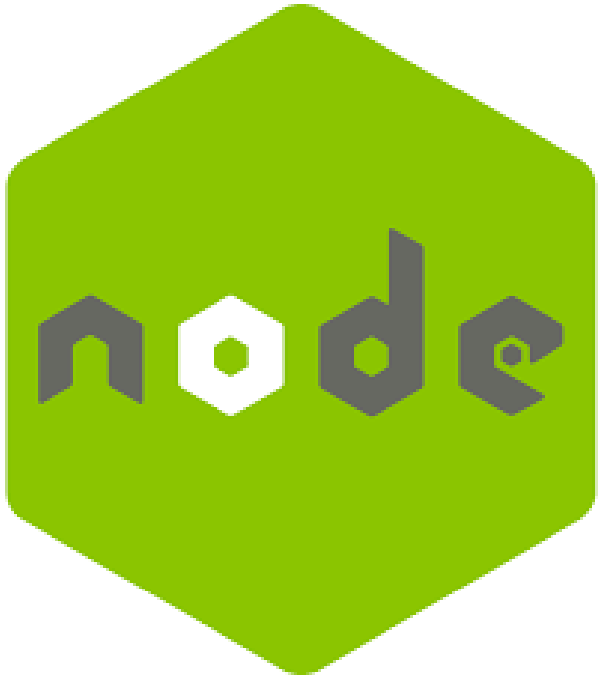
- Made it way easier to get started on the project
- If we were to do this over again, we'd try to get the interactive debugger setup WAY SOONER

# GRPC


- Language-agnostic RPC framework with support for most languages.
- Real world and popular in most Kubernetes / microservices environments
- Our end result is good! Method calls in a try/catch block.

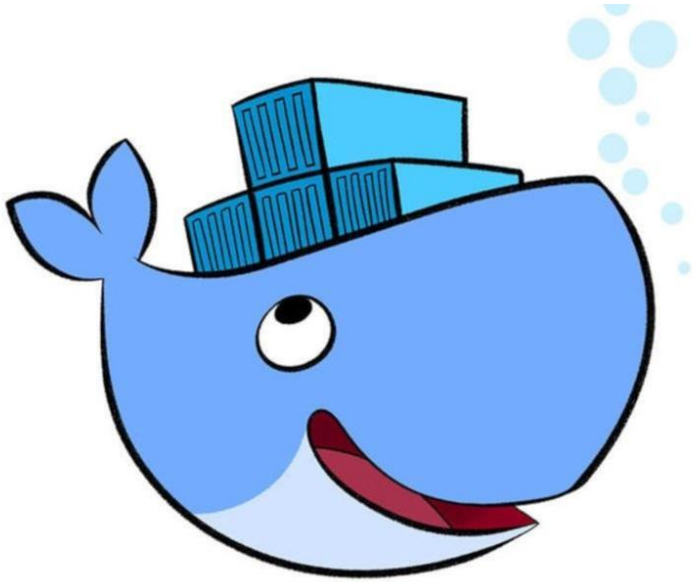
- Challenges
  - Had to learn new Protobuf Syntax
  - Struggled with asynchronous versus synchronous APIs
  - Not obvious how to make clean looking Node.js with gRPC API. Documentation site did not use modern syntax

# NODE.JS




- Provided the easiest onramp to gRPC. We could dynamically load protobufs rather than doing AOT compilation
- Álvaro prefers C#, I prefer Rust, Naim prefers Go, Tyler prefers Python. We picked Node.js as the “least common denominator” language. It looks like C, but is high level and multi-paradigm
- Advantages:
  - gRPC WAS WAY EASIER!!!
  - Isomorphic: Use the same language for client and server
  - Good 3rd Party Module Ecosystem. We were able to swap out gRPC libraries.
- Disadvantages
  - Not a close to pseudocode as Python, a more elegant language
  - Math operators worked a bit differently than expected
  - More debugging because of loose types

# DOCKER



One command to start a 25-node cluster!

docker-compose up --scale node_secondary=24 -d

- Use Case
  - Lightweight Containers run our components
  - Docker Compose defines network and how to scale horizontally
- Challenges
  - Docker for Windows required Windows 10 Professional, but Álvaro and Tyler only have Windows 10 Home
  - Docker works slightly differently between Windows and Linux

# AZURE CONTAINER INSTANCES

- Managed Service for running containerized distributed systems
- Compared to AWS Fargate, much simpler!

- Challenges:
  - Had to learn new deployment script because it can't use Docker Compose files