Baskin Engineering UC SANTA CRUZ

Capstone Project

A Real-Time Heatmap WebApp for Storage Array Diagnostics Visualization

Dominic Balassone, Ben D'Costa,
Boris Matusov, Taylor Stratton, John Ta

PURE STORAGE

Abstract

Pure Storage provides flash storage arrays, a much faster storage solution compared to traditional disk drives. Each array maintains diagnostics data, or information about the array's health. Analysis of this diagnostic data is an important part of the support given to clents using this flash storage. Since there can be many arrays, and each array's diagnostic data file can be up to 500MB, parsing and making sense of the diagnostic data in a timely fashion can be complex.

Our project goal was to design and implement a back-to-front solution to provide users with an informative visualization to allow for easy monitoring of all arrays' health.

Approach

We divided our software engineering process into two main phases: search for technologies and implementation.

In our search we looked for technologies that provide real-time and scalable capabilities. We chose a number of AWS services and RethinkDB to provide a source of truth. We also chose Node.js as the Javascript runtime environment for Socket.io and D3. These libraries combined would allow us to create a user friendly and interactive visualization that updates in real-time.

For implementation we used an Agile/ Scrum methodology with sprint cycles and iterative design.

Productivity Tools

Git - repository for revision and version control

Slack - team communication and collaboration tool

Trello - virtual Scrum board for task management

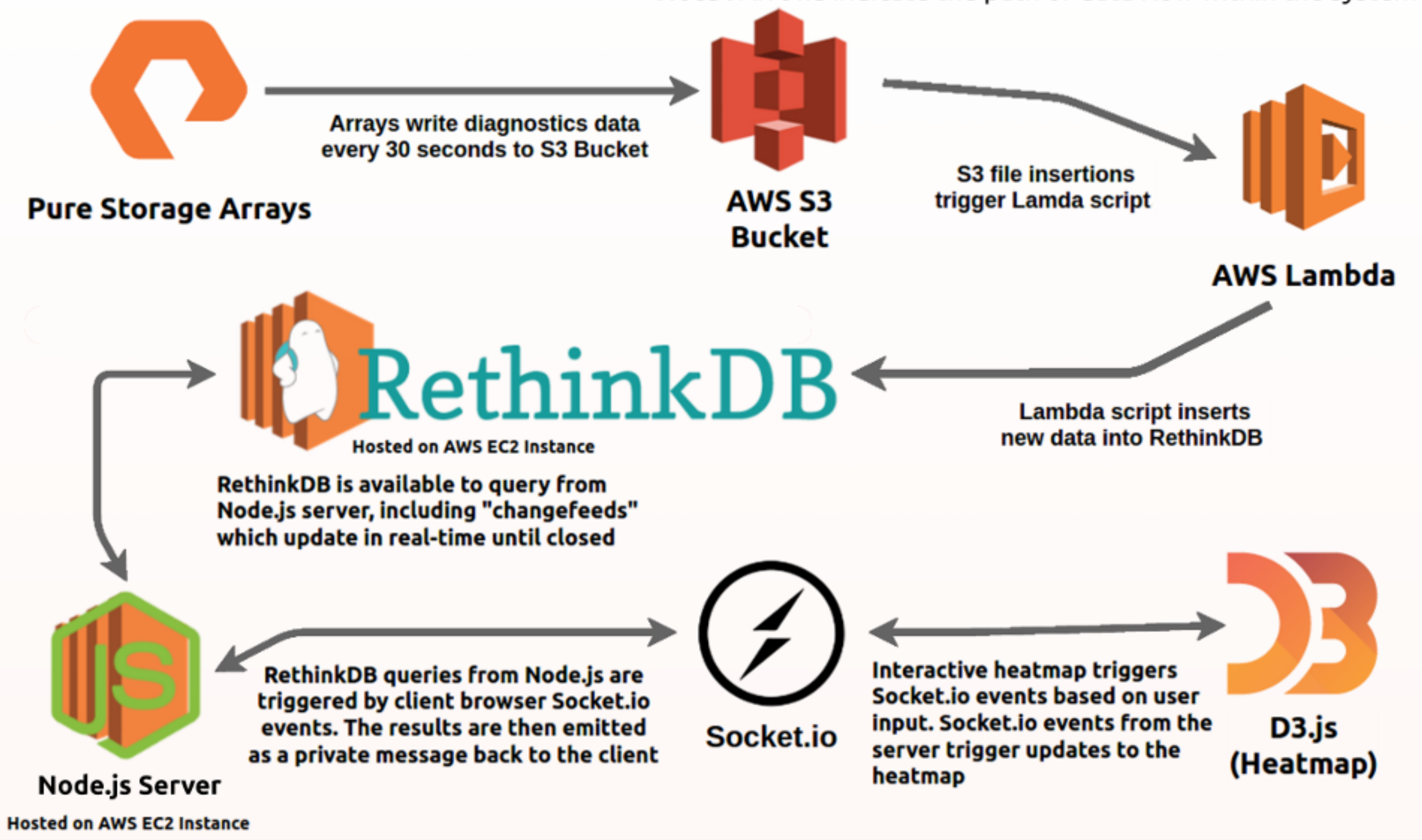
Google Drive - collaboration tool for presentation and design documents.

Acknowledgments

Lei Zhou, PureStorage Software Engineer, Project Manager
Ankur Chandra, PureStorage Software Engineer, Project Manager
Kongposh Sapru, Graduate Teaching Assistant, UCSC
Prof. Linda Werner, Faculty Advisor, UCSC
Andy Lien, UCSC Graduate, Contributed to initial design

System Architecture Diagram

Note : Arrows indicate the path of data flow within the system


Pure Storage Arrays
Arrays write diagnostics data every 30 seconds to S3 Bucket
AWS S3 Bucket
S3 file insertions trigger Lamda script
AWS Lambda
Lambda script inserts new data into RethinkDB
RethinkDB
Hosted on AWS EC2 Instance
RethinkDB is available to query from Node.js server, including "changefeeds" which update in real-time until closed
Node.js Server
Hosted on AWS EC2 Instance
RethinkDB queries from Node.js are triggered by client browser Socket.io events. The results are then emitted as a private message back to the client
Socket.io
Interactive heatmap triggers Socket.io events based on user input. Socket.io events from the server trigger updates to the heatmap
D3.js (Heatmap)

Component Descriptions

AWS EC2 Cluster
Linux servers hosted by Amazon which can easily interact with other AWS components as well as serve whatever code we load them with. For our project we used three instances, one for the database, another for serving the website and communicating with users, and a final server for testing which mimics Pure Storage arrays writing to S3.

AWS S3 Bucket
Amazon hosted filesystem which receives and stores all incoming array diagnostics. A single bucket holds an unlimited number of files (in our case, multiple Pure Storage arrays running and filling S3 with their diagnostics simultaneously).

AWS Lambda
Amazon Web Service which detects the data that was inserted into the S3 bucket, and then adds that information to the real-time database's (RethinkDB) "**recent**" and "**historical**" tables via a ReQL query.

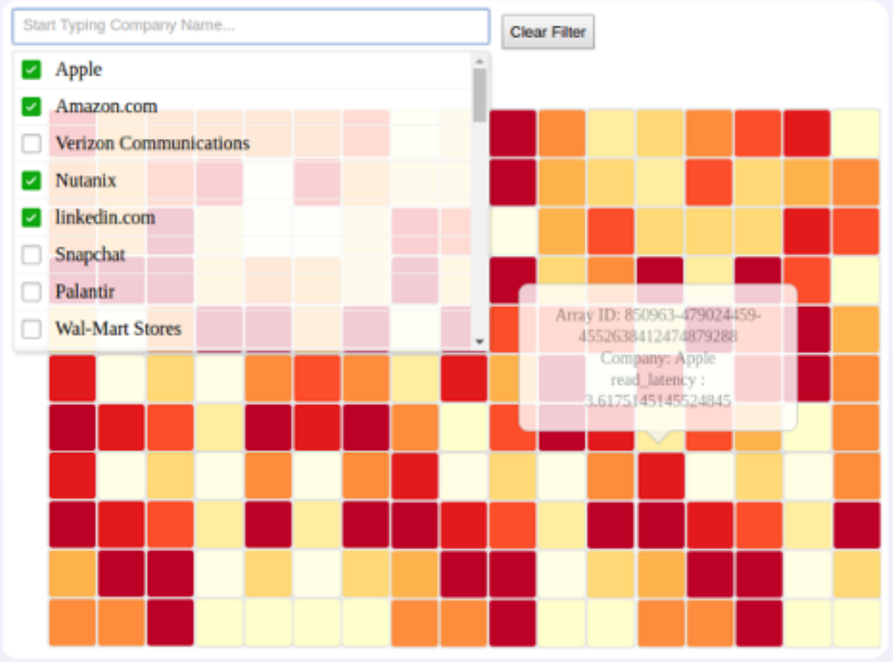
RethinkDB (AWS EC2 Instance)
NoSQL JSON document based database with built in query language (ReQL) containing features designed for real-time applications, such as simplified timestamp querying via a RethinkDB timestamp object, as well as the ability to open "**changefeeds**" which broadcast the changes of any specific query, in real-time. Serves as front-end's source of truth.

Node.js Server (AWS EC2 Instance)
A Javascript, event based environment, with the role of receiving requests for heatmap data from clients via Socket.io, query Rethinkdb with ReQL for the data, and return a live feed of those queries (and changefeeds) via Socket.io. It also serves the web page contents which allows users to view the D3 heatmap in their browser.

Socket.IO
Unlike a normal query, which involves a "call and response" behavior, socket.io allows the client to connect to the server socket, which acts like a phone call, both sides able to interact in real-time, as long as the connection is open. By only allowing clients to interact with Socket.io from the browser, RethinkDB is protected from malicious queries.

D3.js
D3 (Data-Driven Documents) is a javascript library for creating interactive visualizations embedded within web pages, and also what we used to create our real-time heatmap. As data changes at the Socket.io connection, the elements of our D3 heatmap are automatically refreshed to show the user the most up-to-date values. Connected to each element of the heatmap is a Pure Storage array's data, as well as a Socket.io trigger for historical data, which can be inspected by interacting with the visualization.

Heatmap User Interface



- ✓ Apple
- ✓ Amazon.com
- ☐ Verizon Communications
- ✓ Nutanix
- ✓ linkedin.com
- ☐ Snapchat
- ☐ Palantir
- ☐ Wal-Mart Stores

Array ID: 850963-479024459-455263841247487928
Company: Apple
read_latency :
3.6175145145524845

Heatmap Features

- Each heatmap cell represents a single array's health.
- Cells update in real-time based on most recent Socket.IO emission.
- Users can filter by company and metrics such as IOPS, latency, and temperature.
- Cell mouseover provides a hoverbox with detailed information.
- When a cell is clicked, historical data for that array is displayed in time-series line graph form populated with 24 hours of recent data.

Challenges and Resolutions

C: *An initial heatmap prototype had zoomable features that were difficult to incorporate into our real-time update.*
R: We threw away the zoomable prototype and built a heatmap with less features that suits real-time updates.

C: *RethinkDB's changefeeds are an untraditional way of querying for data. It was not obvious how to design a database schema to transmit only user requested data.*
R: We iterated through many RethinkDB schemas and found one that fit the time-sensitive demands of our application.

C: *Collaboration issues included simultaneous updates of code base and security permission failures within AWS.*
R: We learned more effective use of the productivity tools to increase team cohesion.