GridAI: Requirements & Engineering Standards

sdmay21-23@iastate.edu

Karthik Prakash

Abir Mojumder

Justin Merkel

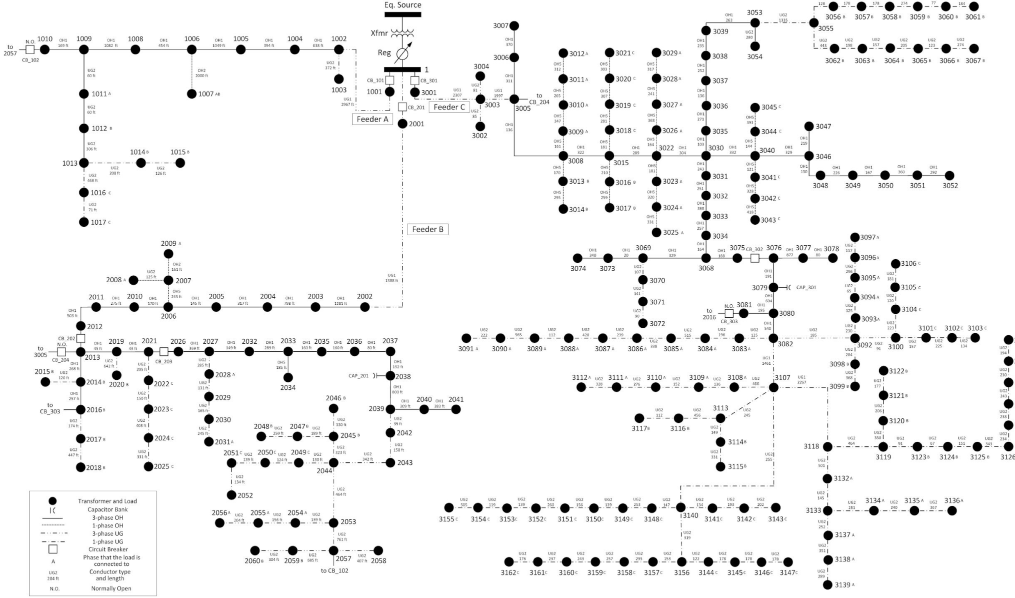
Abhilash Tripathy

Patrick Wenzel

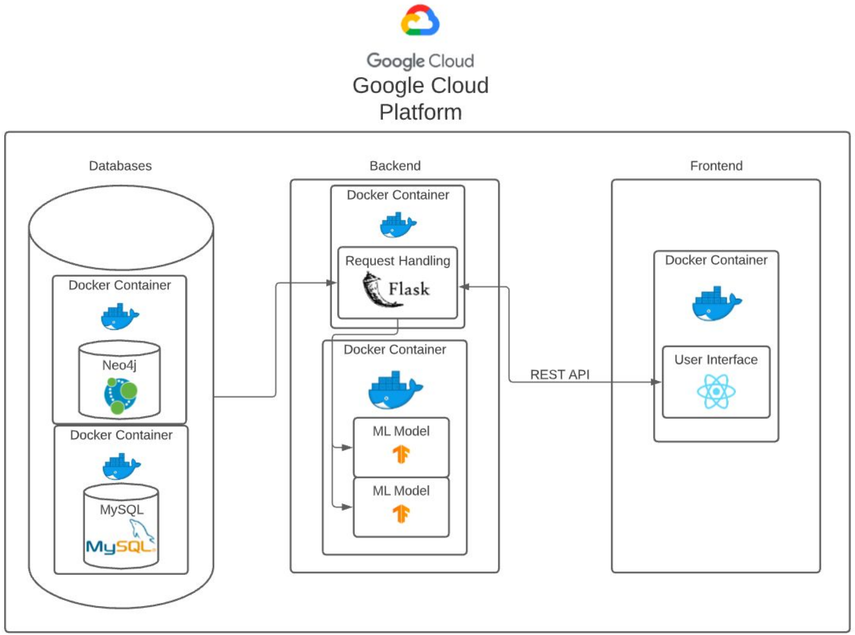
Client:
Dr. Gelli
Ravikumar

Project Context

- Use Machine Learning on a simulated power grid to provide analytics and anomaly detection
  - Every node has some power output data associated
  - Static electrical properties
  - Location and connections in network



High-Level Design



Project Requirements

- Provide Real-time analysis of grid data
- Implement Machine Learning models to generate data predictions and anomaly detection over real time power grid data streams
- Frontend interface for grid data, predictions and anomaly visualization
- Use of Docker Containers and PowerCyber testbed working environment

Functional Requirements for Backend

- Technology Requirement
    - Python, Neo4j Database, TensorFlow 2.0, Docker, PowerCyber Testbed Environment
- Machine learning algorithms
    - Provide analyses and insight for simulated power grid
        - Predict transformer output
        - Classify potential anomalies within grid

Functional Requirements for Frontend

- Front-end receives data from backend
- Front-end interface for data visualization
  - Interface directly with backend
  - Graph-based visualization
  - Geographical representation of power grid
  - Charts for history and predictions for each node
  - Tabular data showing anomaly status for every node

Non-functional Requirements

- Maintainability
  - Keep code modular
- Response time
  - Lightweight frontend to accommodate response rate of work heavy backend
- Clear Documentation of code
  - Future senior design teams can take over and upgrade the application

Engineering Constraints

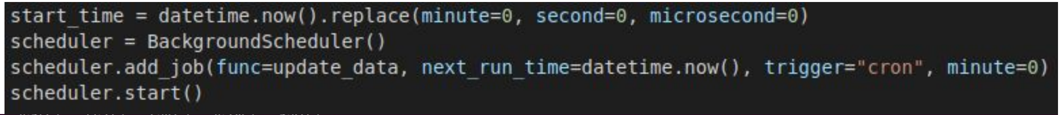
- Only getting $300 in trial credits when using the Google Cloud
- Only have 1 year’s worth of real data and the rest has to be simulated

Engineering Standards

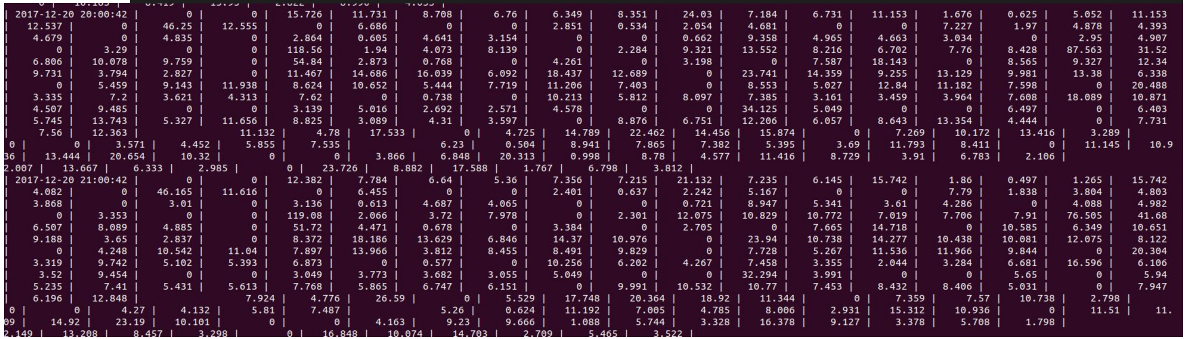
- IEEE/ISO/IEC 12207-2017: Software life cycle processes
    - IEEE/ISO/IEC 29148-2018: Systems and software engineering - Life cycle processes -- Requirements engineering
- IEEE/ISO/IEC 23026-2015: Systems and software engineering - Engineering and management of websites for systems, software, and services information

Technical Challenges: Backend

- Parsing data from database
    - SQLAlchemy
- Implementing real-time updates
- Simulating the openDSS model of the grid to generate new data



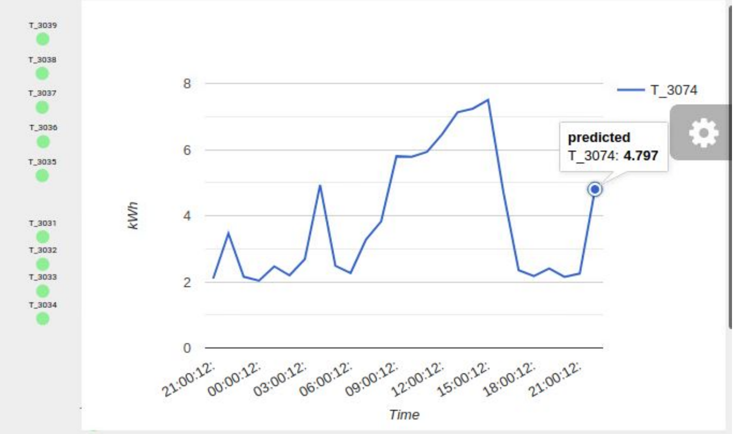
```
start_time = datetime.now().replace(minute=0, second=0, microsecond=0)
scheduler = BackgroundScheduler()
scheduler.add_job(func=update_data, next_run_time=datetime.now(), trigger="cron", minute=0)
scheduler.start()
```

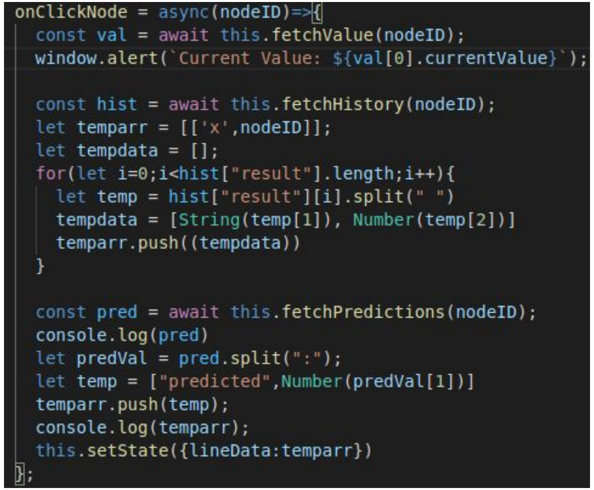


```
| 2017-12-20 20:00:42 |       0 |       0 |  15.726 |  11.731 |   8.708 |    6.76 |   6.349 |   8.351 |   24.03 |   7.184 |   6.731 |  11.153 |   1.676 |   0.625 |   5.052 |  11.153
|  12.537 |       0 |   46.25 |  12.555 |       0 |   6.686 |       0 |       0 |   2.851 |   0.534 |   2.054 |   4.681 |       0 |       0 |   7.227 |    1.97 |   4.878 |   4.393
|   4.679 |       0 |   4.835 |       0 |   2.864 |   0.605 |   4.641 |   3.154 |       0 |       0 |   0.662 |   9.358 |   4.965 |   4.663 |   3.034 |       0 |    2.95 |   4.907
|       0 |    3.29 |       0 |       0 |  118.56 |    1.94 |   4.073 |   8.139 |       0 |   2.284 |   9.321 |  13.552 |   8.216 |   6.702 |    7.76 |   8.428 |  87.563 |   31.52
|   6.806 |  10.078 |   9.759 |       0 |   54.84 |   2.873 |   0.768 |       0 |   4.261 |       0 |   3.198 |       0 |   7.587 |  18.143 |       0 |   8.565 |   9.327 |   12.34
|   9.731 |   3.794 |   2.827 |       0 |  11.467 |  14.686 |  16.039 |   6.092 |  18.437 |  12.689 |       0 |  23.741 |  14.359 |   9.255 |  13.129 |   9.981 |   13.38 |   6.338
|       0 |   5.459 |   9.143 |  11.938 |   8.624 |  10.652 |   5.444 |   7.719 |  11.206 |   7.403 |       0 |   8.553 |   5.027 |   12.84 |  11.182 |   7.598 |       0 |  20.488
|   3.335 |     7.2 |   3.621 |   4.313 |    7.62 |       0 |   0.738 |       0 |  10.213 |   5.812 |   8.097 |   7.385 |   3.161 |   3.459 |   3.964 |   7.608 |  18.089 |  10.871
|   4.507 |   9.485 |       0 |       0 |   3.139 |   5.016 |   2.692 |   2.571 |   4.578 |       0 |       0 |  34.125 |   5.049 |       0 |       0 |   6.497 |       0 |   6.403
|   5.745 |  13.743 |   5.327 |  11.656 |   8.825 |   3.089 |    4.31 |   3.597 |       0 |   8.876 |   6.751 |  12.206 |   6.057 |   8.643 |  13.354 |   4.444 |       0 |   7.731
|    7.56 |  12.363 |                  11.132 |    4.78 |  17.533 |       0 |   4.725 |  14.789 |  22.462 |  14.456 |  15.874 |       0 |   7.269 |  10.172 |  13.416 |   3.289 |
 0 |       0 |   3.571 |   4.452 |   5.855 |   7.535 |                    6.23 |   0.504 |   8.941 |   7.865 |   7.382 |   5.395 |    3.69 |  11.793 |   8.411 |       0 |  11.145 |    10.9
36 |  13.444 |  20.654 |   10.32 |       0 |       0 |   3.866 |   6.848 |  20.313 |   0.998 |    8.78 |   4.577 |  11.416 |   8.729 |    3.91 |   6.783 |   2.106 |
2.007 |  13.667 |   6.333 |   2.985 |       0 |  23.726 |   8.882 |  17.588 |   1.767 |   6.798 |   3.812 |
| 2017-12-20 21:00:42 |       0 |       0 |  12.382 |   7.784 |    6.64 |    5.36 |   7.356 |   7.215 |  21.132 |   7.235 |   6.145 |  15.742 |    1.86 |   0.497 |   1.265 |  15.742
|   4.082 |       0 |  46.165 |  11.616 |       0 |   6.455 |       0 |       0 |   2.401 |   0.637 |   2.242 |   5.167 |       0 |       0 |    7.79 |   1.838 |   3.804 |   4.803
|   3.868 |       0 |    3.01 |       0 |   3.136 |   0.613 |   4.687 |   4.065 |       0 |       0 |   0.721 |   8.947 |   5.341 |    3.61 |   4.286 |       0 |   4.088 |   4.982
|       0 |   3.353 |       0 |       0 |  119.08 |   2.066 |    3.72 |   7.978 |       0 |   2.301 |  12.075 |  10.829 |  10.772 |   7.019 |   7.706 |    7.91 |  76.505 |   41.68
|   6.507 |   8.089 |   4.885 |       0 |   51.72 |   4.471 |   0.678 |       0 |   3.384 |       0 |   2.705 |       0 |   7.665 |  14.718 |       0 |  10.585 |   6.349 |  10.651
|   9.188 |    3.65 |   2.837 |       0 |   8.372 |  18.186 |  13.629 |   6.846 |   14.37 |  10.976 |       0 |   23.94 |  10.738 |  14.277 |  10.438 |  10.081 |  12.075 |   8.122
|       0 |   4.248 |  10.542 |   11.04 |   7.897 |  13.966 |   3.812 |   8.455 |   8.491 |   9.829 |       0 |   7.728 |   5.267 |  11.536 |  11.966 |   9.844 |       0 |  20.304
|   3.319 |   9.742 |   5.102 |   5.393 |   6.873 |       0 |   0.577 |       0 |  10.256 |   6.202 |   4.267 |   7.458 |   3.355 |   2.044 |   3.284 |   6.681 |  16.596 |   6.106
|    3.52 |   9.454 |       0 |       0 |   3.049 |   3.773 |   3.682 |   3.055 |   5.049 |       0 |       0 |  32.294 |   3.991 |       0 |       0 |    5.65 |       0 |    5.94
|   5.235 |    7.41 |   5.431 |   5.613 |   7.768 |   5.865 |   6.747 |   6.151 |       0 |   9.991 |  10.532 |   10.77 |   7.453 |   8.432 |   8.406 |   5.031 |       0 |   7.947
|   6.196 |  12.848 |                   7.924 |   4.776 |   26.59 |       0 |   5.529 |  17.748 |  20.364 |   18.92 |  11.344 |       0 |   7.359 |    7.57 |  10.738 |   2.798 |
 0 |       0 |    4.27 |   4.132 |    5.81 |   7.487 |                    5.26 |   0.624 |  11.192 |   7.005 |   4.785 |   8.006 |   2.931 |  15.312 |  10.936 |       0 |   11.51 |    11.
09 |   14.92 |   23.19 |  10.101 |       0 |       0 |   4.163 |    9.23 |   9.666 |   1.088 |   5.744 |   3.328 |  16.378 |   9.127 |   3.378 |   5.708 |   1.798 |
2.149 |  13.208 |   8.457 |   3.298 |       0 |  16.848 |  10.074 |  14.703 |   2.709 |   5.465 |   3.522 |
```

Technical Challenges: Frontend

- React uses asynchronous methods to update information before updating screen. This might cause very large data api calls to have more latency.
- Some components (d3-Grid/google linechart/react) required specific versions for them to work together, leading to outdated/fewer functions.





```
onClickNode = async(nodeID)=>{
  const val = await this.fetchValue(nodeID);
  window.alert(`Current Value: ${val[0].currentValue}`);

  const hist = await this.fetchHistory(nodeID);
  let temparr = [['x',nodeID]];
  let tempdata = [];
  for(let i=0;i<hist["result"].length;i++){
    let temp = hist["result"][i].split(" ")
    tempdata = [String(temp[1]), Number(temp[2])]
    temparr.push((tempdata))
  }

  const pred = await this.fetchPredictions(nodeID);
  console.log(pred)
  let predVal = pred.split(":");
  let temp = ["predicted",Number(predVal[1])]
  temparr.push(temp);
  console.log(temparr);
  this.setState({lineData:temparr})
};
```

Questions?