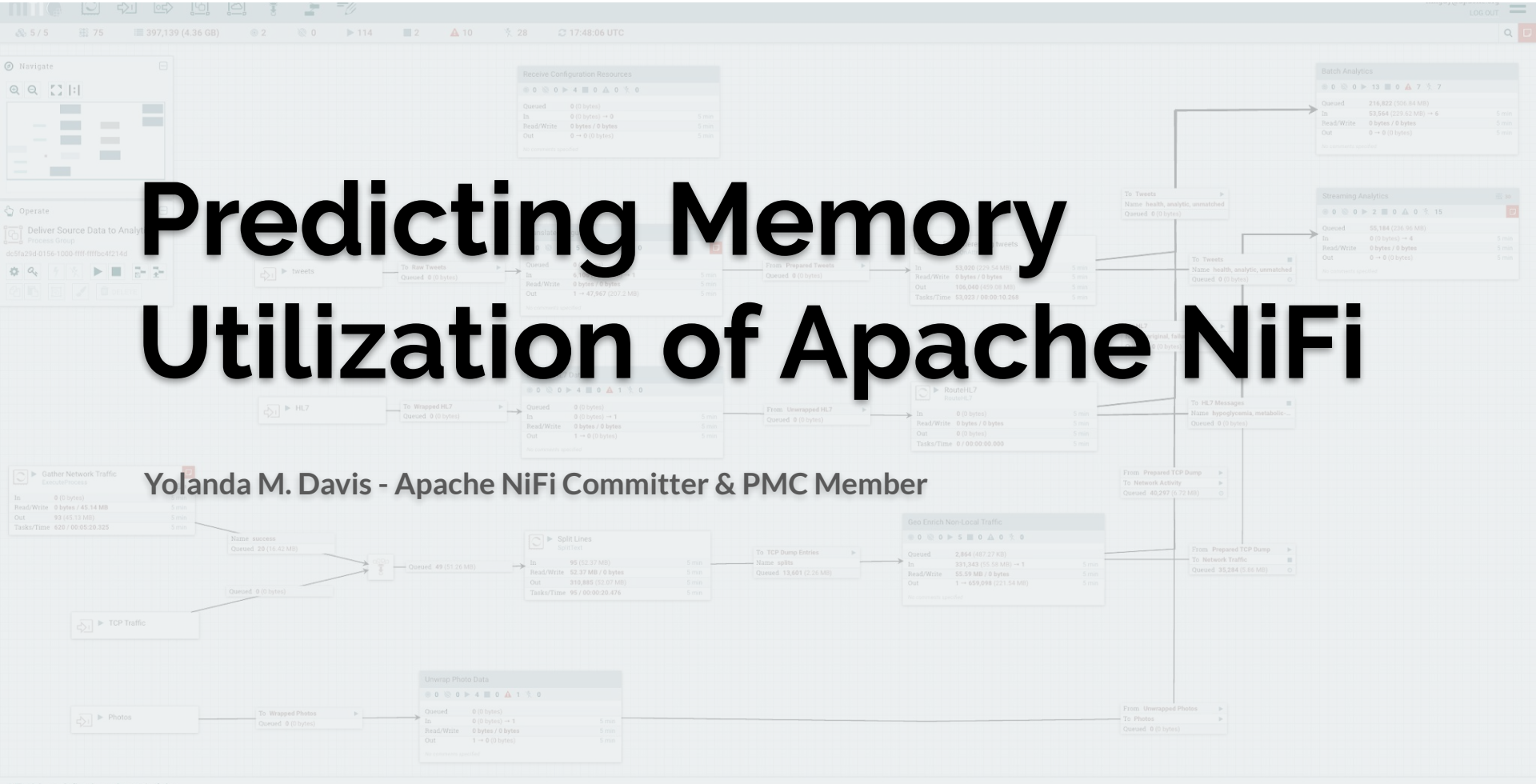


Predicting Memory Utilization of Apache NiFi

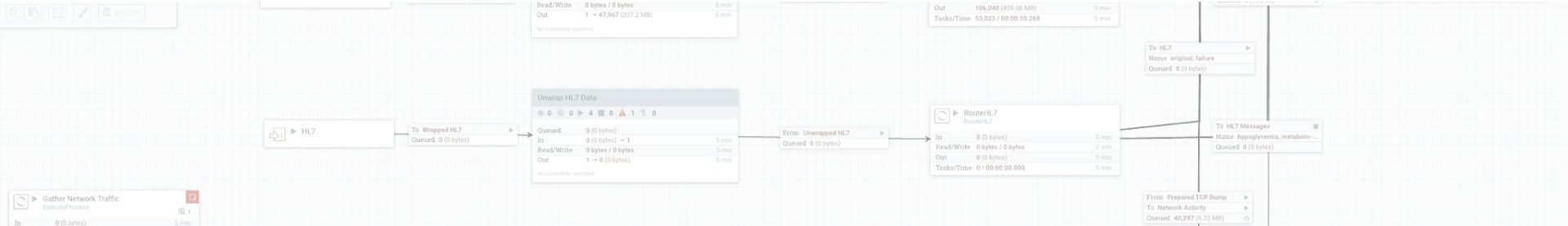
Yolanda M. Davis - Apache NiFi Committer & PMC Member

What is Apache NiFi

Apache NiFi is an application which supports the automation of data processing and exchange (or flow) between systems.

NiFi allows users to create their own flows by connecting several "black-box" processes (called processors) together via queues (or connections), allowing data to flow from one function to the next until it lands in its final destination.

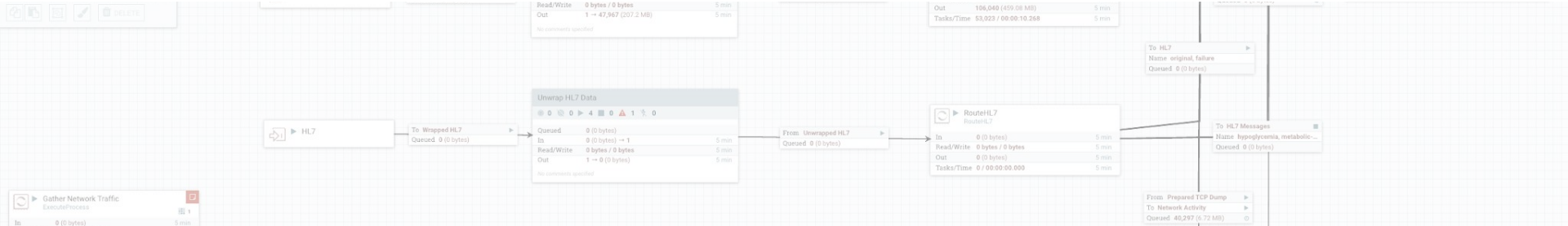
The goal for optimal flow design is to have data enter and exit the flow as quickly as possible but there are lots of factors that can hamper this including the size data, the speed of the individual processors, the size of the queues holding data, or memory available to the cluster.



The Problem

- Running out of memory can be extremely disruptive to flows and difficult to see in advance
- Periodic bursts might require more resources than what's considered normal
- Having access to too much unused memory may have an additional cost over time

Can we devise a method to predict the amount of memory needed in advance to avoid disruptions and/or lead to cost savings ?



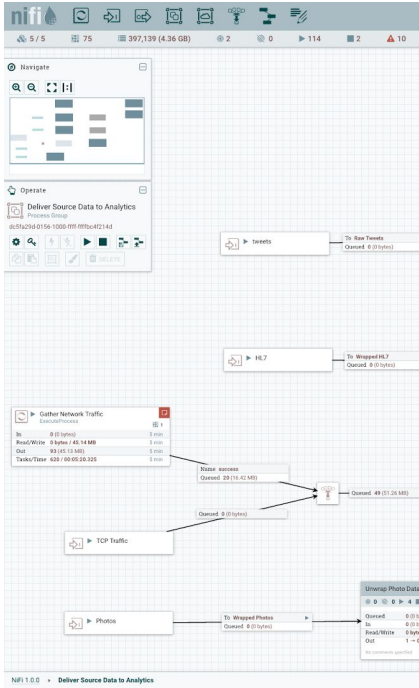
How to Measure & Predict Memory Usage

Using metrics collected from a NiFi cluster we can obtain time series information on observations such as memory used (heap/non-heap), active threads, amount of data (in bytes) received or sent, and load average on a given machine

With this data we can train an algorithm to predict the amount of memory required on a given day for a particular flow.

Fun Fact

NiFi was originally developed within the National Security Agency (NSA) and released as open source in 2014

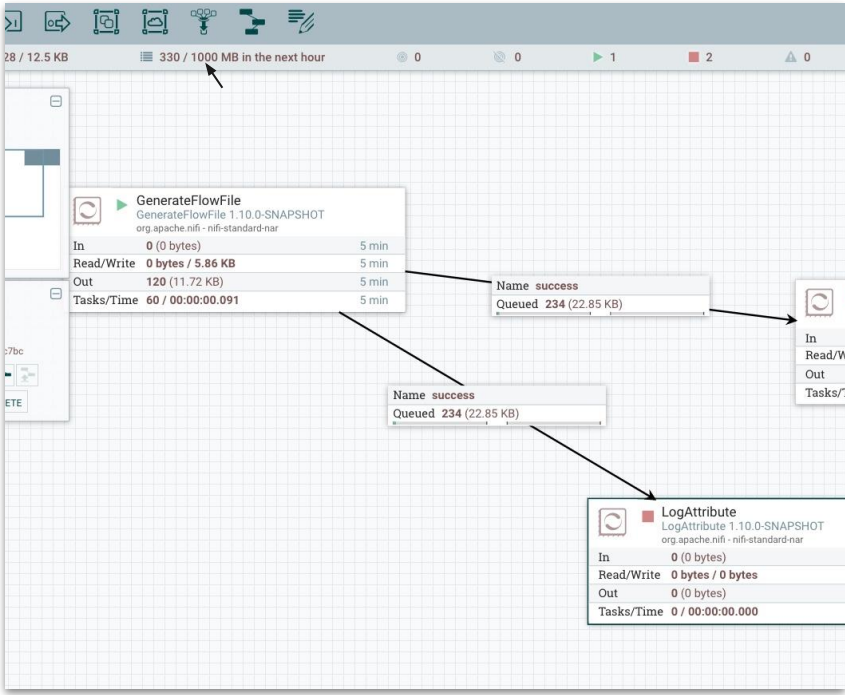


Potential Applications

Alerts & Notifications

Users working within NiFi can see predictions of memory usage to immediately gauge potential resource consumption (without waiting for a threshold to be exceeded).

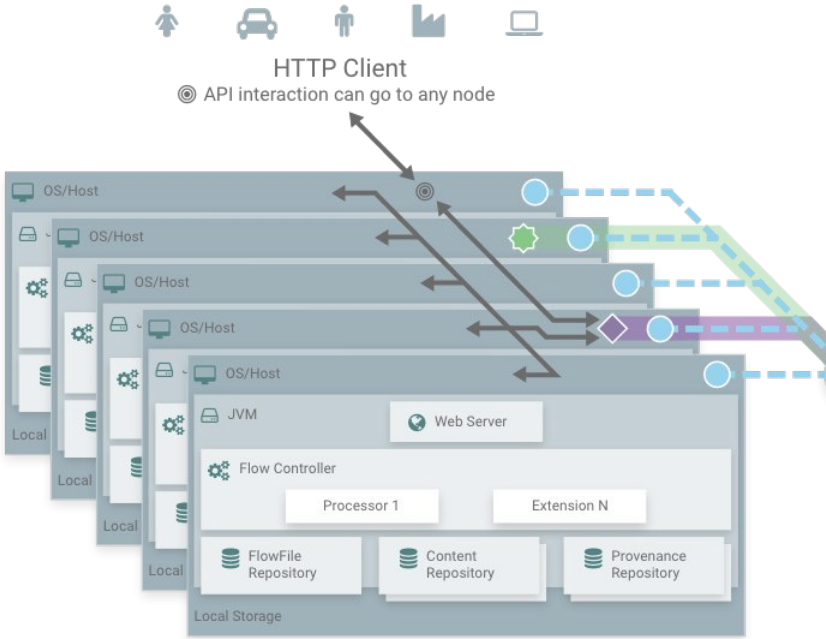
Model can be incorporated as part of recent analytics framework:
https://github.com/apache/nifi/tree/analytics-framework



Potential Applications

Auto-scale & Recommendations

These predictions can also be used to plan for scaling in or out NiFi, depending on activity for a given day, to obtain or release access to memory resources as needed. Scaling can be performance manually or automatically controlled through orchestration systems like Kubernetes



The Data

One week's worth of data was extracted from a Prometheus server which collected NiFi metrics from a single NiFi instance. The number of observations collected were **3849** rows. This data was cleansed to ensure labels were easily readable and confirmed that there were no empty values received. Data was also enhanced to create features columns from datetime (e.g. day of the week, hour of the day)

Approximately

3,849

Observations gathered over one week

25

Features available from original extraction

53

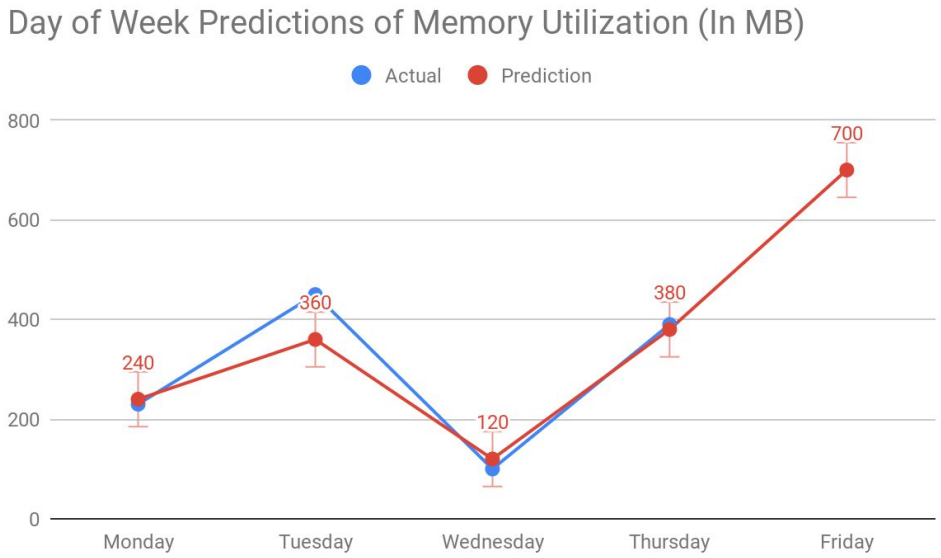
53 features total after data cleansing and feature engineering

The Approach

Because of the small size of the data set, my approach was to perform cross validation on two types of regression models. Linear Regression with Polynomial Features included (specifically interaction variables) and Random Forest Regression.

To score the models, I reviewed their **R-squared scores**, which demonstrates how well a model does compared to simply predicting based on the average memory usage, and **RMSE**, which gives a general idea of how much a prediction can be above or below the actual value. In other words with RMSE we can assume that the prediction can be + or - a certain amount of bytes.

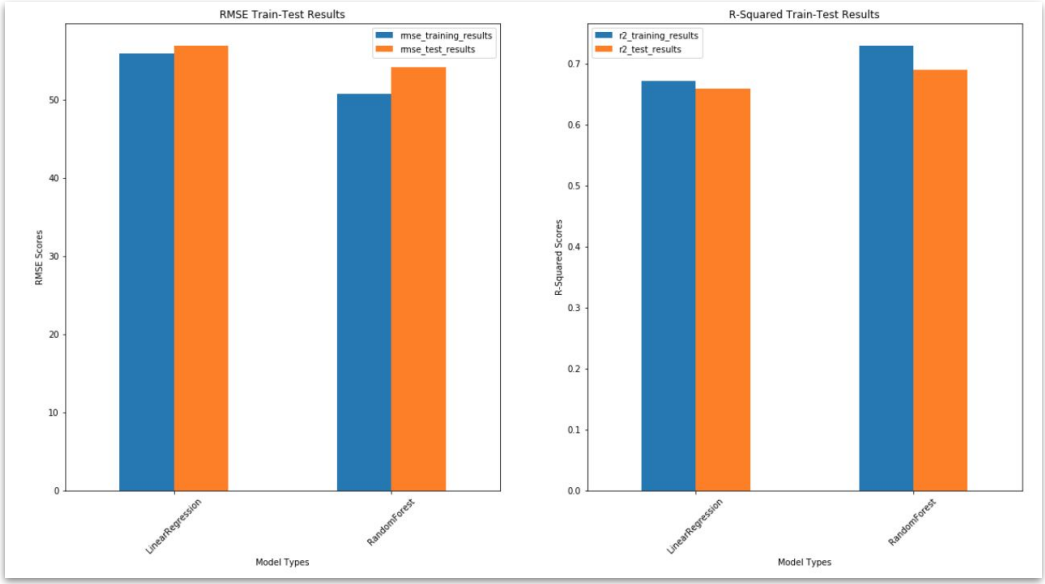
Having both values helps to not only compare models **but also can be used in production to assess if the model's prediction is good enough to use**.



The Results

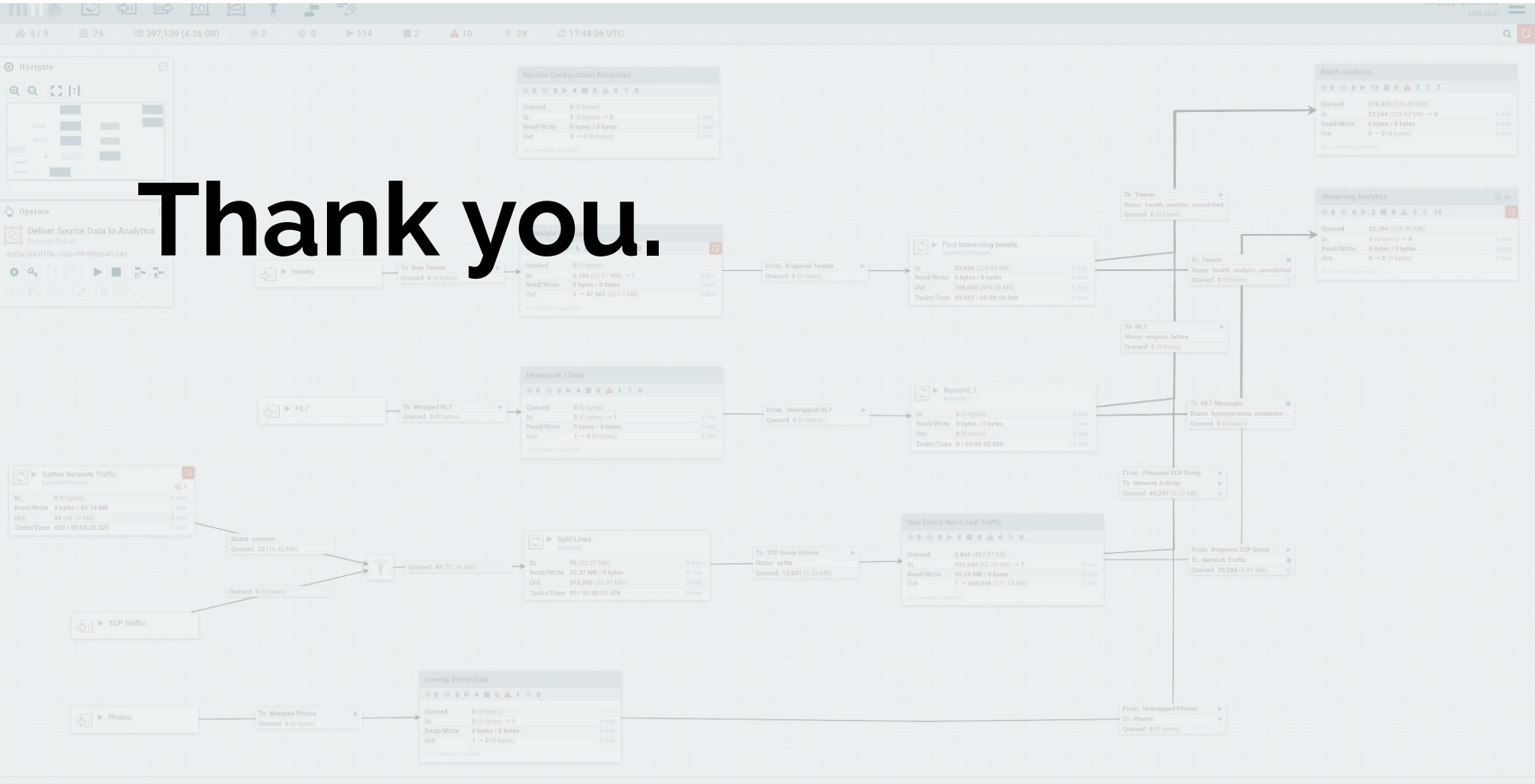
Our models both worked well when comparing against the predicting the average (aka the “null” model). The Random Forest Regression model performed slightly better than Linear Regression, however in both cases training set performed slightly better than the test set (about .02 - .04 difference in R-Squared scores). I do believe that could be improved by introducing more observations for training however was not a deterrent for using the model.

	Linear Regression	RandomForest (Best Predictor)
R-Squared (Training)	0.6708	0.7209
R-Squared (Test)	0.6578	0.6896
RMSE (in MB) (Training)	55.97	50.77
RMSE (in MB) (Test)	56.96	54.165



Recommendations for Next Steps

- Research Random Forest model further (obtain more observations that are aggregated for **max and min usage**)
- Integrate model with new Analytics Framework using remote model interface
- Deploy model as a standalone API that can be queried by auto-scaling technology


Thank you.