EARL London | 10-12 September, 2019

Large-Scale Time Series Forecasting in Apache Spark

Timothy Wong, Phuong Pham

Centrica PLC

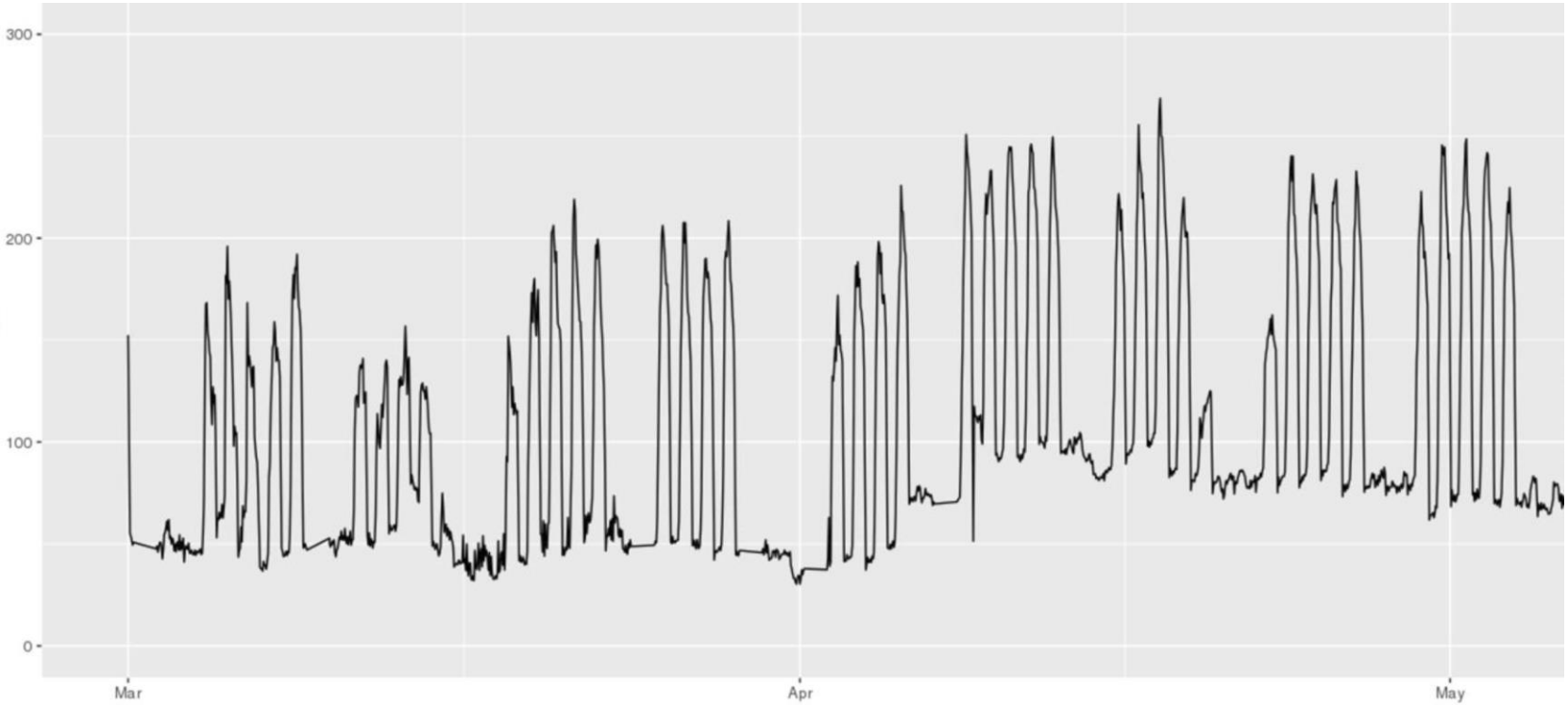
Centrica

- We supply energy and services to over 27 million customer accounts
- Supported by around 12,000 engineers and technicians
- Our areas of focus are Energy Supply & Services, Connected Home, Distributed Energy & Power, Energy Marketing & Trading

Power Forecasting

- Account level hourly forecast for every smart meter customer in the ERCOT (Texas, USA) market

Power usage at account level
(A typical example)

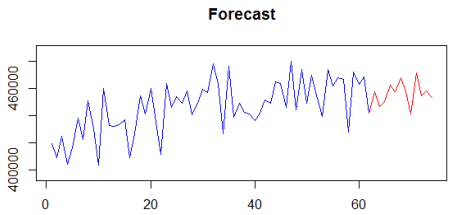


How the data is used?

Power usage data

Hourly granularity, received 1x per day

Scheduled to run 3x per day



Forecast algorithm

Volume traded on market

Weather data

Hourly granularity, received 3x per day

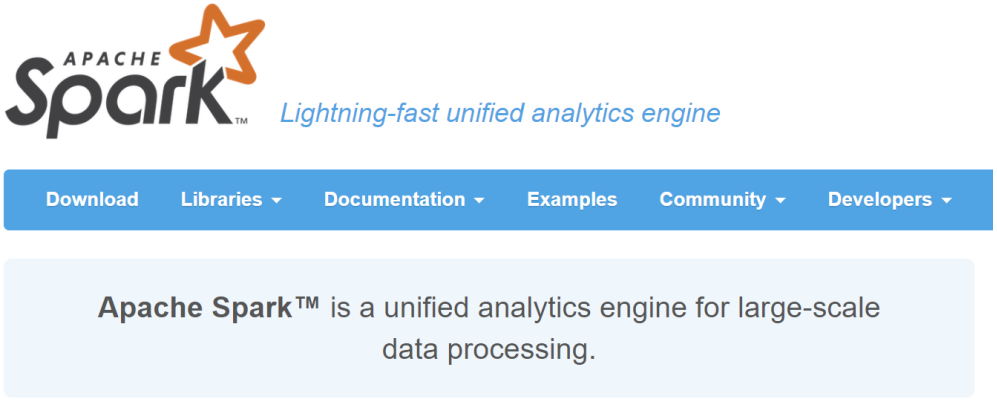
EARL CONFERENCE 2019

Problems

- Forecast is required at account (meter) level
  - We have many accounts (75K)
  - Assume 1 sec per account, sequential run takes 21 hours!
- Power forecast is time-sensitive
  - Multiple runs per-day
  - Forecast must be produced on time
- Accuracy is important
  - Desire for sophisticated models
  - Sophisticated models are usually long-running!

Architecture

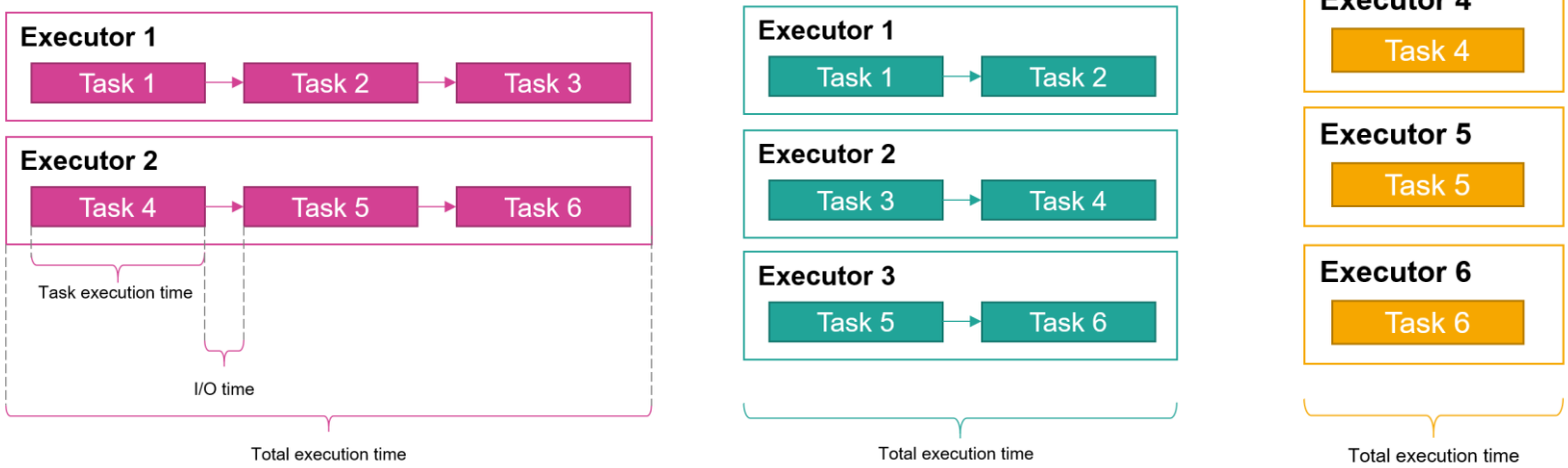
Framework



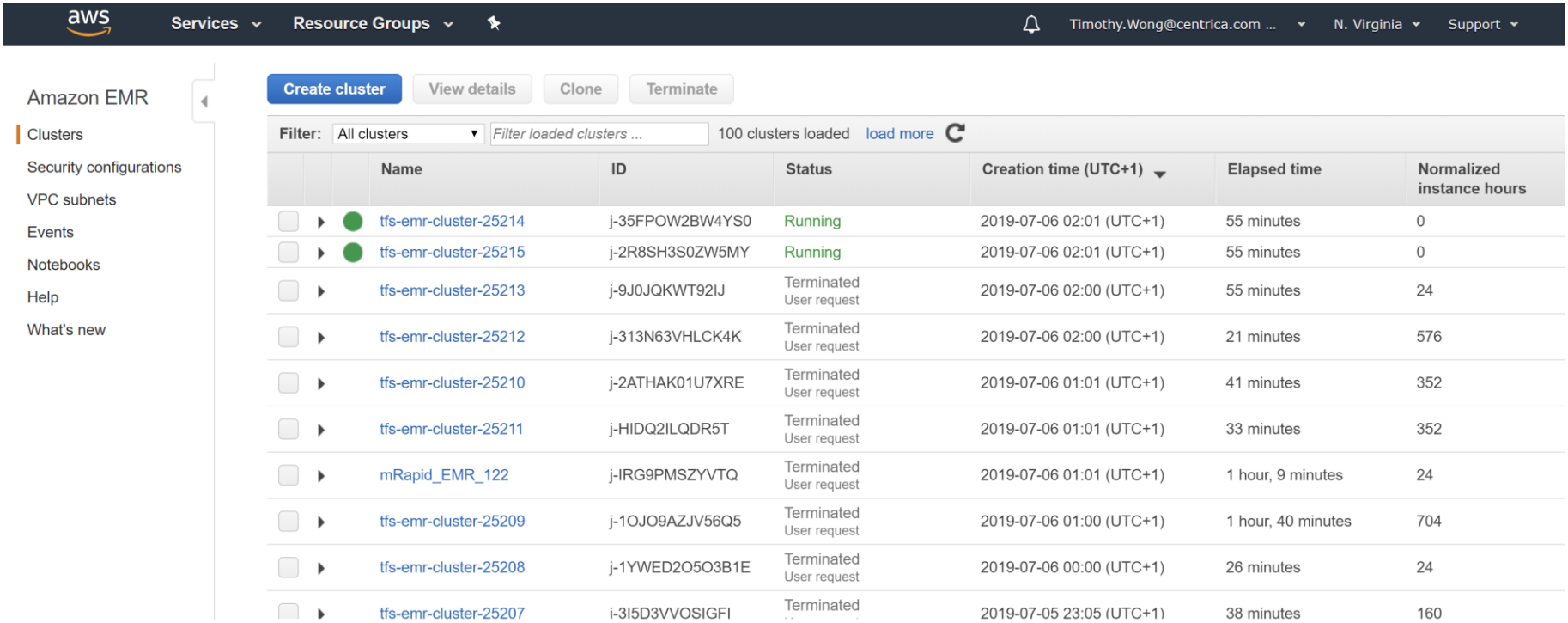
- Apache Spark
  - Allow high degree of parallelism
  - Handling large datasets
  - Run native R code in Spark
- Benefits
  - Make use of sophisticated algorithms in R
  - Parallelism can save a lot of time
  - Scalable by design
- Disadvantages
  - Compute power can be expensive

Parallelism

- Apache Spark is a parallel compute framework
- Divide data into many small partitions
- Increasing the number of partitions will encourage parallelism
- Too many partitions may incur excessive I/O overhead



AWS
Elastic MapReduce (EMR)



Setting up EMR

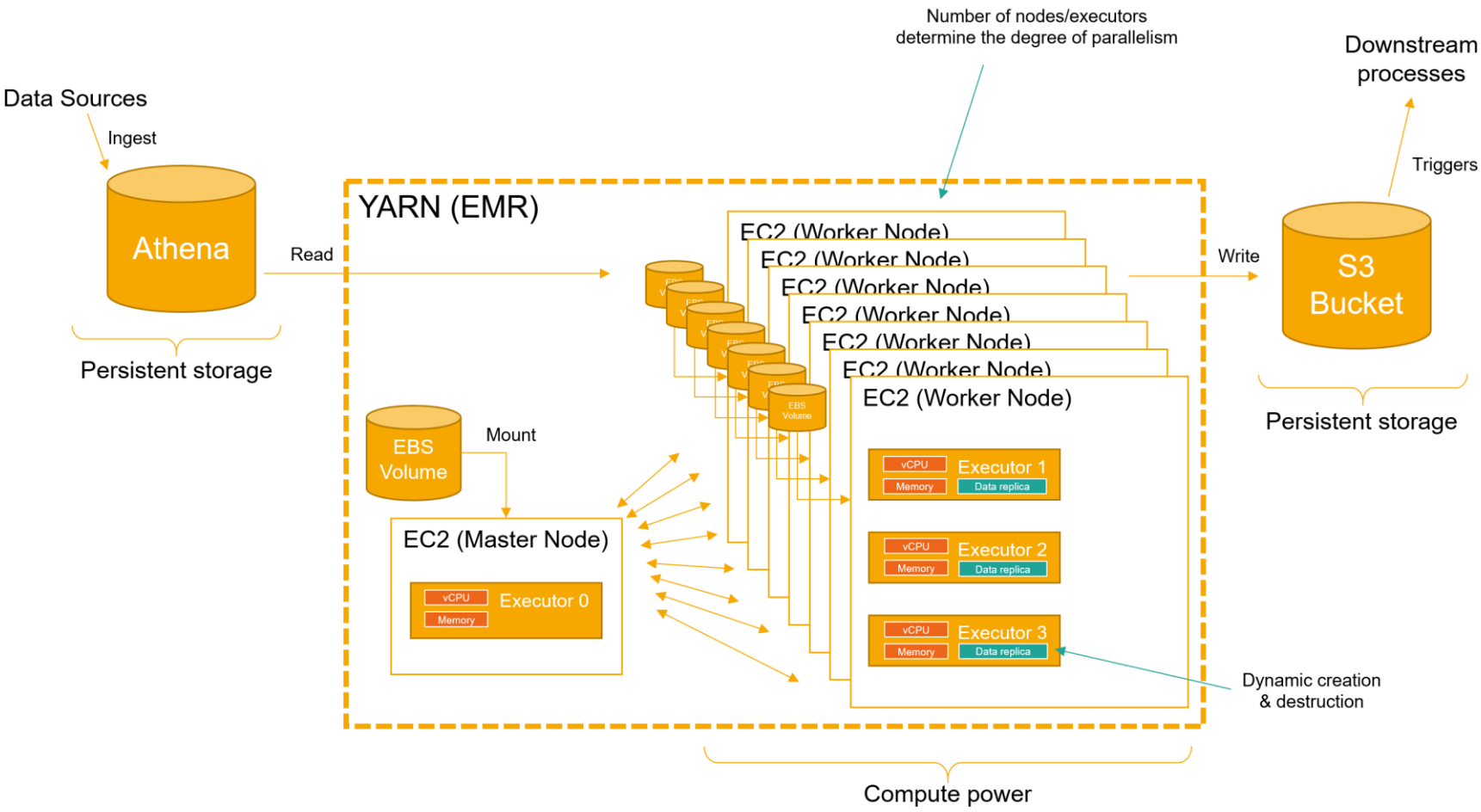
- EC2 charge + EMR charge
- Billed per second
- Terminate once job is completed
- Pay for what you use

Pricing for Amazon EMR and Amazon EC2 (On-Demand)

Region: US East (N. Virginia)

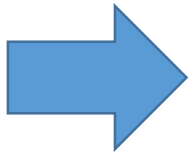
	Amazon EC2 Price	Amazon EMR Price
General Purpose - Current Generation		
m5.xlarge	$0.192 per Hour	$0.048 per Hour
m5.2xlarge	$0.384 per Hour	$0.096 per Hour
m5.4xlarge	$0.768 per Hour	$0.192 per Hour
m5.12xlarge	$2.304 per Hour	$0.27 per Hour
m5.24xlarge	$4.608 per Hour	$0.27 per Hour
m5a.xlarge	$0.172 per Hour	$0.043 per Hour
m5a.2xlarge	$0.344 per Hour	$0.086 per Hour

Cloud Architecture



Other Options

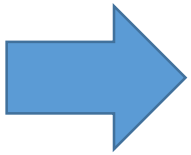
Google Cloud Platform



GCP Dataproc

Amazon EMR

Microsoft Azure

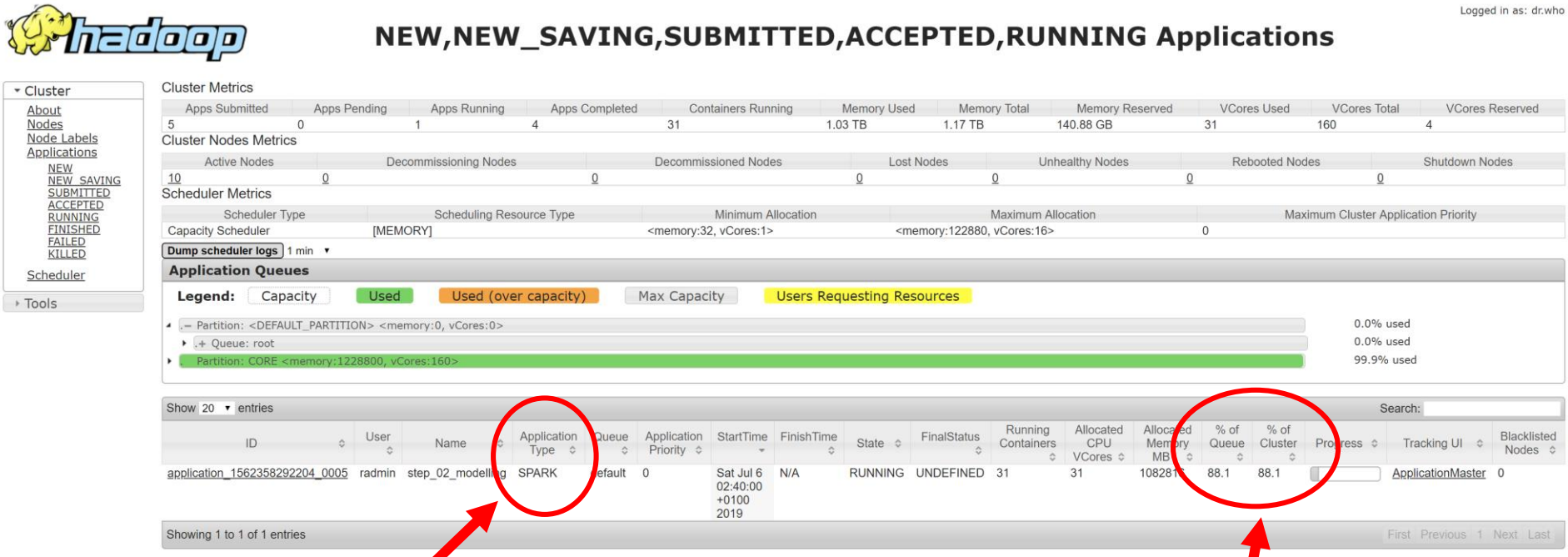


databricks

Connecting to Spark (EMR)

```
37 library(sparklyr)
38 conf <- spark_config()
39 conf["sparklyr.log.console"] <- TRUE
40 conf["spark.network.timeout"] <- "3600s"
41 conf["spark.sql.crossJoin.enabled"] <- "true"
42 conf["spark.sql.parquet.compression.codec"] <- "snappy"
43 conf["spark.r.command"] <- "/opt/R/3.5.0/lib64/R/bin/Rscript"
44 conf["spark.sql.shuffle.partitions"] <- "8000"
45 conf["spark.default.parallelism"] <- "8000"
46 conf["spark.scheduler.listenerbus.eventqueue.capacity"] <- "100000"
47 conf["spark.dynamicAllocation.enabled"] <- "true"
48 conf["spark.shuffle.service.enabled"] <- "true"
49 conf["spark.driver.maxResultSize"] <- "32G"
50 conf["spark.serializer"] <- "org.apache.spark.serializer.KryoSerializer"
51 conf["spark.kryoserializer.buffer.max"] <- "2047m"
52 conf["sparklyr.apply.serializer"] <- "org.apache.spark.serializer.KryoSerializer"
53
54 sc <- spark_connect(master = "yarn",
55                     config = conf)
```

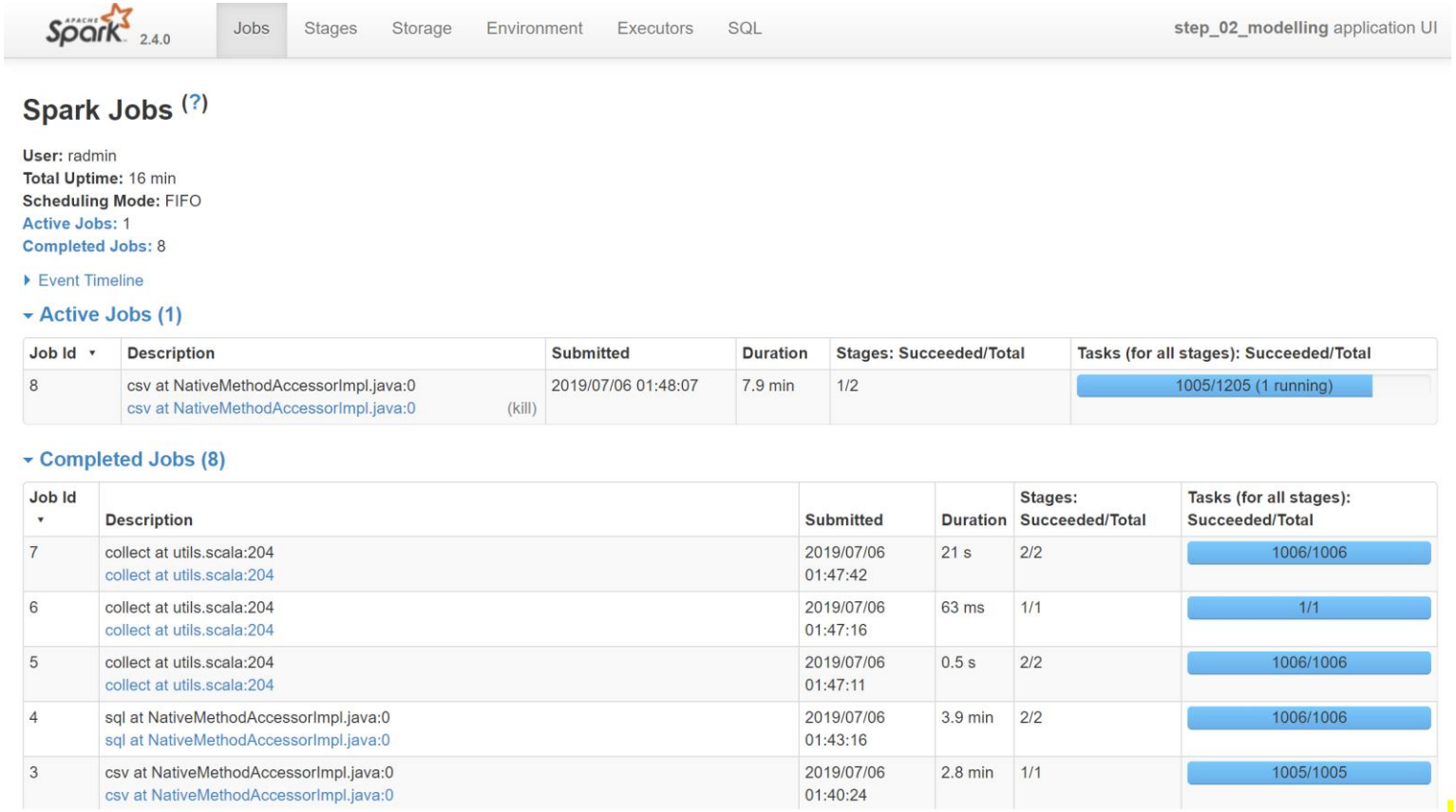
Resource Manager



Spark job is running in YARN

Resource is allocated to this job

Spark UI


Spark 2.4.0
Jobs
Stages
Storage
Environment
Executors
SQL
step_02_modelling application UI
Spark Jobs (?)
User: radmin
Total Uptime: 16 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 8
Event Timeline
Active Jobs (1)
Job Id
Description
Submitted
Duration
Stages: Succeeded/Total
Tasks (for all stages): Succeeded/Total
8
csv at NativeMethodAccessorImpl.java:0
csv at NativeMethodAccessorImpl.java:0 (kill)
2019/07/06 01:48:07
7.9 min
1/2
1005/1205 (1 running)
Completed Jobs (8)
Job Id
Description
Submitted
Duration
Stages: Succeeded/Total
Tasks (for all stages): Succeeded/Total
7
collect at utils.scala:204
collect at utils.scala:204
2019/07/06 01:47:42
21 s
2/2
1006/1006
6
collect at utils.scala:204
collect at utils.scala:204
2019/07/06 01:47:16
63 ms
1/1
1/1
5
collect at utils.scala:204
collect at utils.scala:204
2019/07/06 01:47:11
0.5 s
2/2
1006/1006
4
sql at NativeMethodAccessorImpl.java:0
sql at NativeMethodAccessorImpl.java:0
2019/07/06 01:43:16
3.9 min
2/2
1006/1006
3
csv at NativeMethodAccessorImpl.java:0
csv at NativeMethodAccessorImpl.java:0
2019/07/06 01:40:24
2.8 min
1/1
1005/1005

Running R in Spark

```
30  my_tbl_lcl <- data.frame(x = rnorm(100),
31                           y = rnorm(100),
32                           group = c(1,2))
33
34  my_tbl <- sc %>% copy_to(my_tbl_lcl)
35
36  my_results <- my_tbl %>% spark_apply(function(df){
37    my_model <- lm(y ~ x, df)
38    return(my_model$residuals)
39  }, group_by = "group")
```

Model Training / Deployment

- Training a new model is time consuming
- Batch job to retrain all models regularly
  - Retrain all accounts
  - Takes ~20 hours on AWS EMR
  - 1,200 vCPU & 9.6 TB memory
- Model objects are persisted
  - Saved to AWS S3 as compressed `.rda` file
  - Can be loaded back later for deployment
  - I/O to S3 is operated on Spark *worker* nodes

Saving R object to S3

Boto 3

AWS SDK for Python

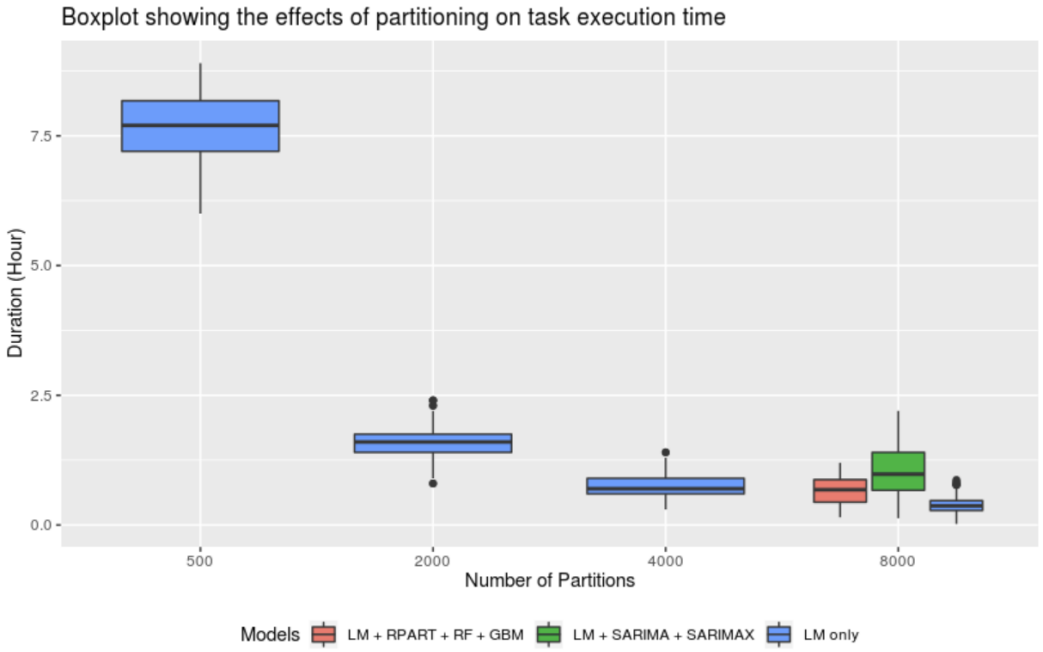
```
11 my_data <- tibble(y = rnorm(1000),
12                    x1 = rnorm(1000),
13                    x2 = rnorm(1000))
14 my_model <- lm(y ~ x, my_data)
15
16 library(reticulate)
17 boto3 <- import("boto3")
18 s3 <- boto3$resource("s3")
19 my_file <- s3$Object("my-bucket-name","my_folder/my_file.RData")
20 my_tmp <- tempfile()
21 saveRDS(my_model, my_tmp)
22 my_file$upload_file(my_tmp)
23 unlink(my_tmp)
```

Loading R object from S3

```
1 library(reticulate)
2 boto3 <- import("boto3")
3 s3 <- boto3$resource("s3")
4 my_file <- s3$Object("my-bucket-name","my_folder/my_file.RData")
5 my_tmp <- tempfile()
6 my_file$download_file(my_tmp)
7 my_object <- readRDS(my_tmp)
8 unlink(my_tmp)
```

Effects of Partitioning

- The execution time of the tasks in the longest stage (***spark_apply***) is measured
  - Timed using **r5.4xlarge** with 10 worker nodes
- This stage governs the total execution time of the pipeline
- There is additional overhead time for data preparation (trivial) and I/O to S3



Partitions	Models	Duration (Minutes)				
		5%	25%	Median	75%	95%
500	LM only	396	432	462	490	519
2000		64	84	96	105	126
4000		24	36	42	54	66
8000		9	17	22	28	38
8000	LM + SARIMA + SARIMAX	18	40	59	84	119
8000	LM + RPART + RF + GBM	16	26	41	53	66

Modelling

Multi-model approach

- Power usage at account level have different behaviours
  - Some are sensitive to weather (e.g. heating / cooling systems)
  - Some have weekly patterns (e.g. office building)
  - Some have regular behaviour (e.g. warehouse)
  - Some have predictable spikes (e.g. cooking at lunch time)
  - Some have unpredictable spikes (?)
  - …
- We need different modelling techniques
  - Run several models for *all* accounts

Algorithms

Acronym	Algorithm	Description
GLM	Generalised Linear Model	Linear regression
SARIMA	Seasonal Autoregressive Integrative Moving Average	Time series model which can deal with seasonality
SARIMAX	Seasonal Autoregressive Integrative Moving Average with Exogeneous Variables	Same as SARIMA but can handle covariate variables (e.g. temperature)
MSARIMA	Multi-Seasonal Autoregressive Integrative Moving Average	Same as SARIMA but handle multiple seasonalities (e.g. daily, weekly, annual…)
MSARIMAX	Multi-Seasonal Autoregressive Integrative Moving Average with Exogeneous Variable	Same as MSARIMA but can handle covariate variables.
RPART	Recursive Partitioning	Decision tree model grows deep and pruned afterwards.
RF	Random Forest	Ensemble of large decision trees with randomly selected variable at each split.
GBM	Gradient Boosting Machine	Shallow decision trees are introduced additively, forming a strong learner.
ELASTICNET	Linear Regression with ElasticNet Regularisation	Regularised linear regression

Dynamic Weighting

GLM
SARIMA
SARIMAX
MSARIMA
MSARIMAX
RPART
RF
GBM
ELASTICNET

- Models are combined using on-line dynamic weighting
  - Calculate Loss function (MAPE) for each model
  - Penalise model with high loss
  - Get weighted forecast

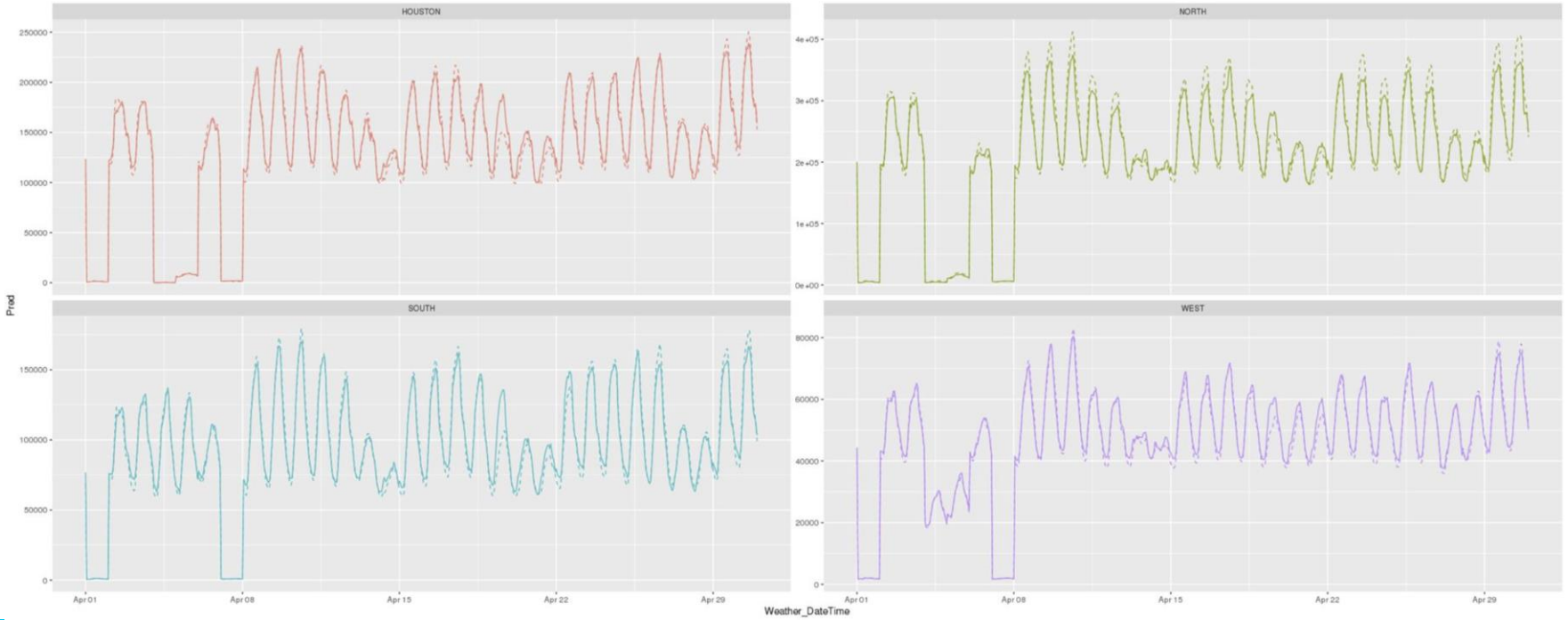
Dynamic Weighting → Weighted forecast

$$w_{t,m} = w_{t-1,m} e^{-\eta L}$$

Results

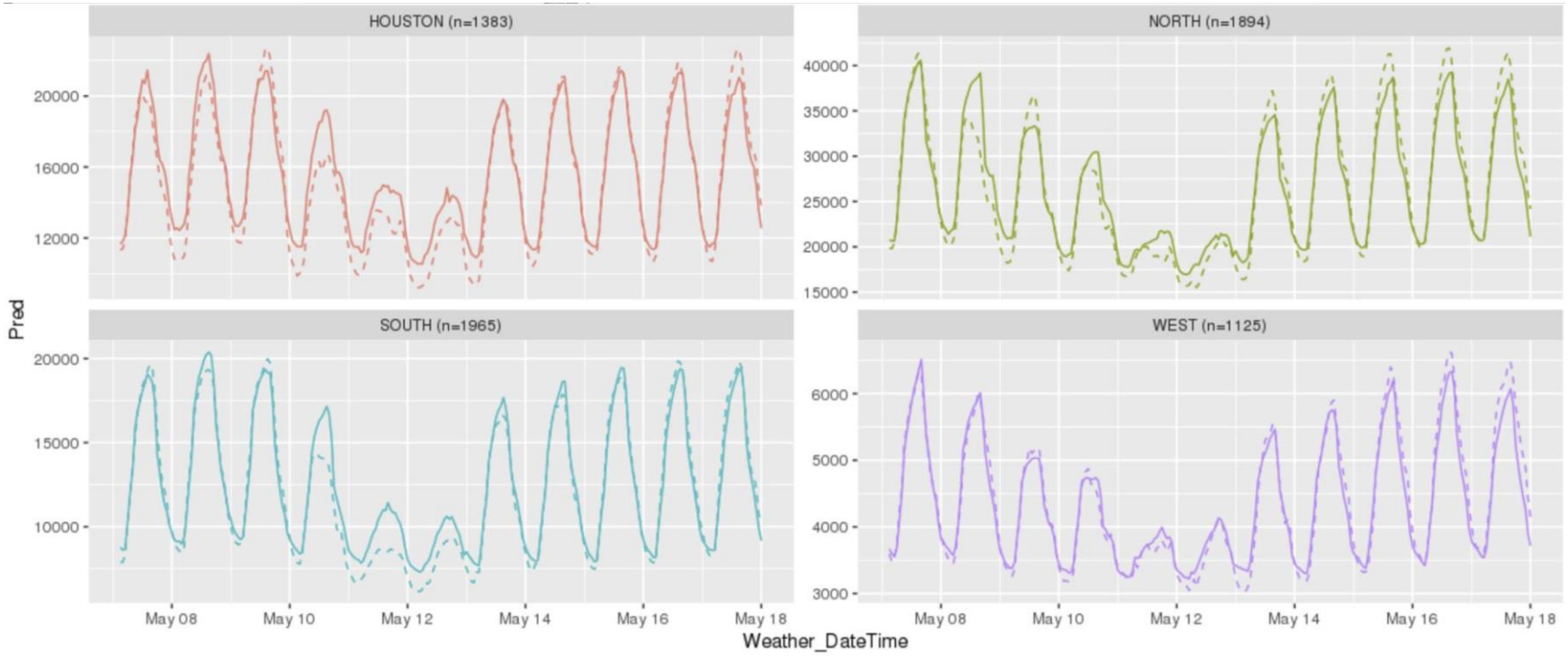
Results - Backcast

- Models were trained using 2 years worth of data up to the end of March 2019
- Backcast is produced using actualised weather data in April 2019 (full month)
- Chart below shows the backcast results (Solid = backcast; Dotted = actual)



Results - Forecast

- 10 days ahead forecast, aggregated at zonal level



Accuracy

- The MAPE of each model were calculated for the 1 month backcast period
- Tree-based methods (GBM, RF, RPART) outperform regression methods
  - Likely due to the discovery of interaction effects
  - Deals with non-linearity very well
- Time series methods (SARIMA, SARIMAX, MSARIMA, MSARIMAX) have worst accuracy
  - Account level usage may have many change points and intermittent spikes, rendering time series methods less useful.

Model	HOUSTON	NORTH	SOUTH	WEST
GBM	5.67	5.83	4.86	3.02
RPART	6.24	5.96	5.10	3.56
RF	6.90	7.20	5.93	3.84
ELASTICNET	9.68	6.54	7.19	4.18
GLM	7.11	7.20	7.16	4.37
MSARIMA	16.97		18.69	10.68
MSARIMAX	17.69		19.59	11.24
SARIMA	35.16		36.97	14.24
SARIMAX	49.35		97.26	38.27

Summary

Technologies Used

Potential Improvements

- **Apache Airflow**
  Workflow monitoring and scheduling
- **Apache Arrow**
  Cross-language development platform for in-memory data (sparklyr)
- **Statistics / ML Modelling**
  Implement better models

  Higher accuracy
  Computational time trade-off

Orchestration

Speed

Accuracy

Process Automation

Cost Optimisation & KPI

Q & A

Timothy Wong CSci CStat CEng MBCS
Principal Data Scientist, Centrica plc

timothy.wong@centrica.com

linkedin.com/in/timothywong731

@timothywong731

timothywong731.github.io

Phuong Pham
Data Scientist, Centrica plc

phuong.pham@centrica.com

linkedin.com/in/phuong-pham-87498251