\Orchestrating a brighter world NEC

Distributed Heterogeneous Mixture Learning on Spark

Masato Asahara and Ryohei Fujimaki
NEC Data Science Research Labs.
Jun/08/2016 @Spark Summit 2016

Who we are?

Masato Asahara (Ph.D.)

Researcher, NEC Data Science Research Laboratory

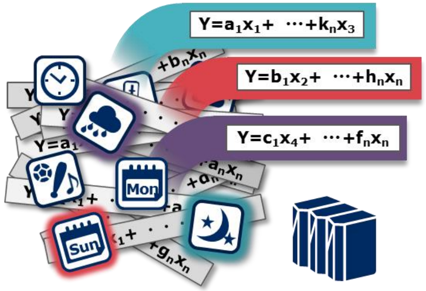
- Masato received his Ph.D. degree from Keio University, and has worked at NEC for 6 years in the research field of distributed computing systems and computing resource management technologies.
- Masato is currently leading developments of Spark-based machine learning and data analytics systems, particularly using NEC's Heterogeneous Mixture Learning technology.

Ryohei Fujimaki (Ph.D.)

Research Fellow, NEC Data Science Research Laboratory

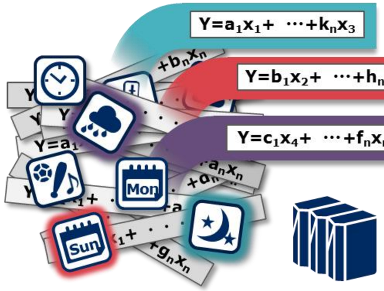
- Ryohei is responsible for cores to NEC's leading predictive and prescriptive analytics solutions, including "heterogeneous mixture learning" and "predictive optimization" technologies. In addition to technology R&D, Ryohei is also involved with co-developing cutting-edge advanced analytics solutions with NEC's global business clients and partners in the North American and APAC regions.
- Ryohei received his Ph.D. degree from the University of Tokyo, and became the youngest research fellow ever in NEC Corporation's 115-year history.

Agenda



HML

Agenda



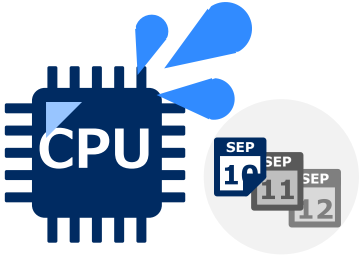
HML

Spark

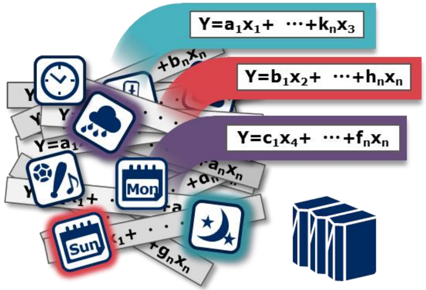
Data Shuffling

MapReduce Synchronization

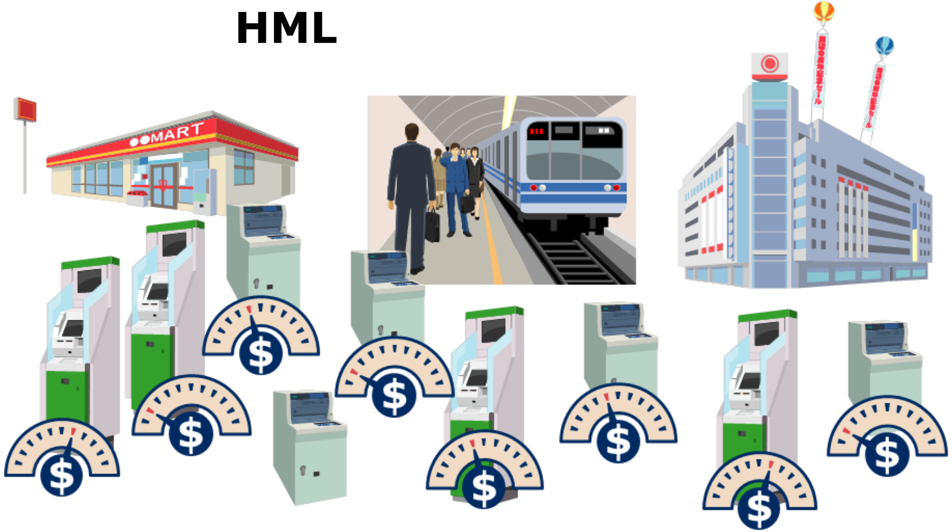
Matrix Computation

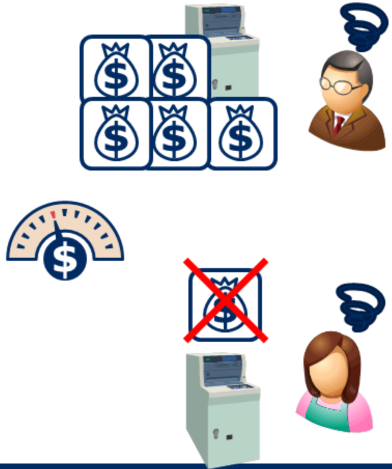


Agenda


$Y=a_1x_1+ \cdots +k_nx_3$
$Y=b_1x_2+ \cdots +h_nx_n$
$Y=c_1x_4+ \cdots +f_nx_n$
Mon
Sun

HML


MART



NEC's Predictive Analytics and Heterogeneous Mixture Learning

Enterprise Applications of HML

Energy/Water Operation Mgmt.

Sales Optimization

Product Price Optimization

Driver Risk Assessment

Predictive Maintenance

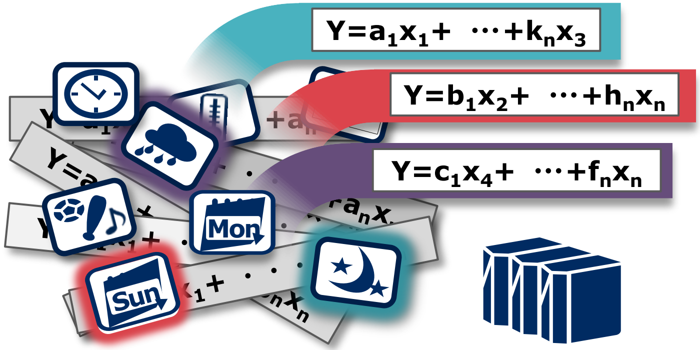
Churn Retention

Inventory Optimization

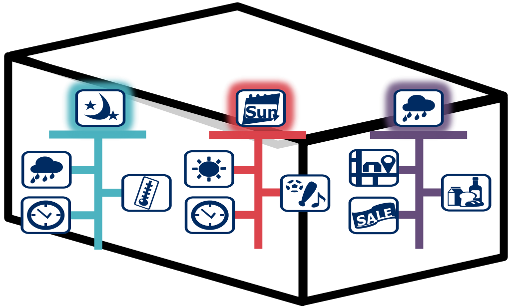
NEC’s Heterogeneous Mixture Learning (HML)

NEC’s machine learning that automatically derives “accurate” and “transparent” formulas behind Big Data.

Heterogeneous mixture data

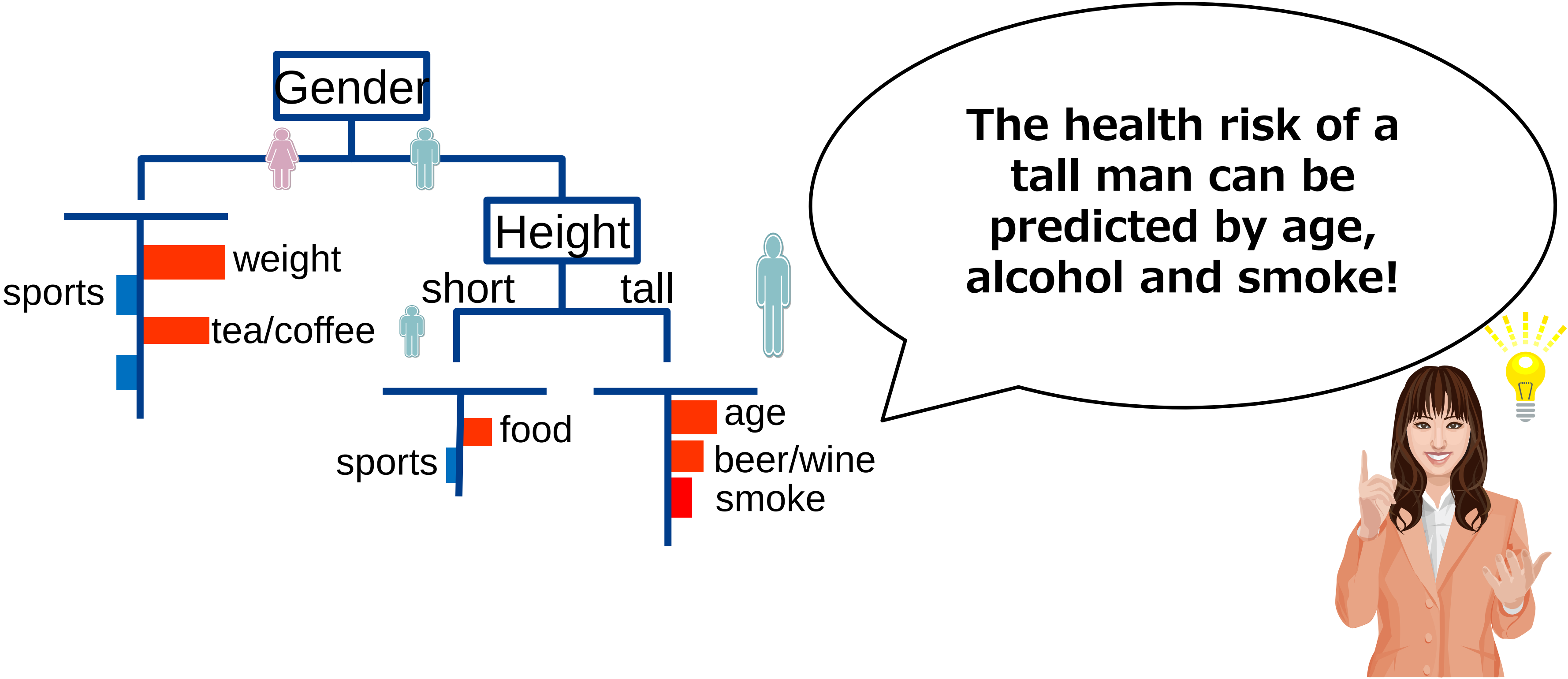


Explore Massive Formulas

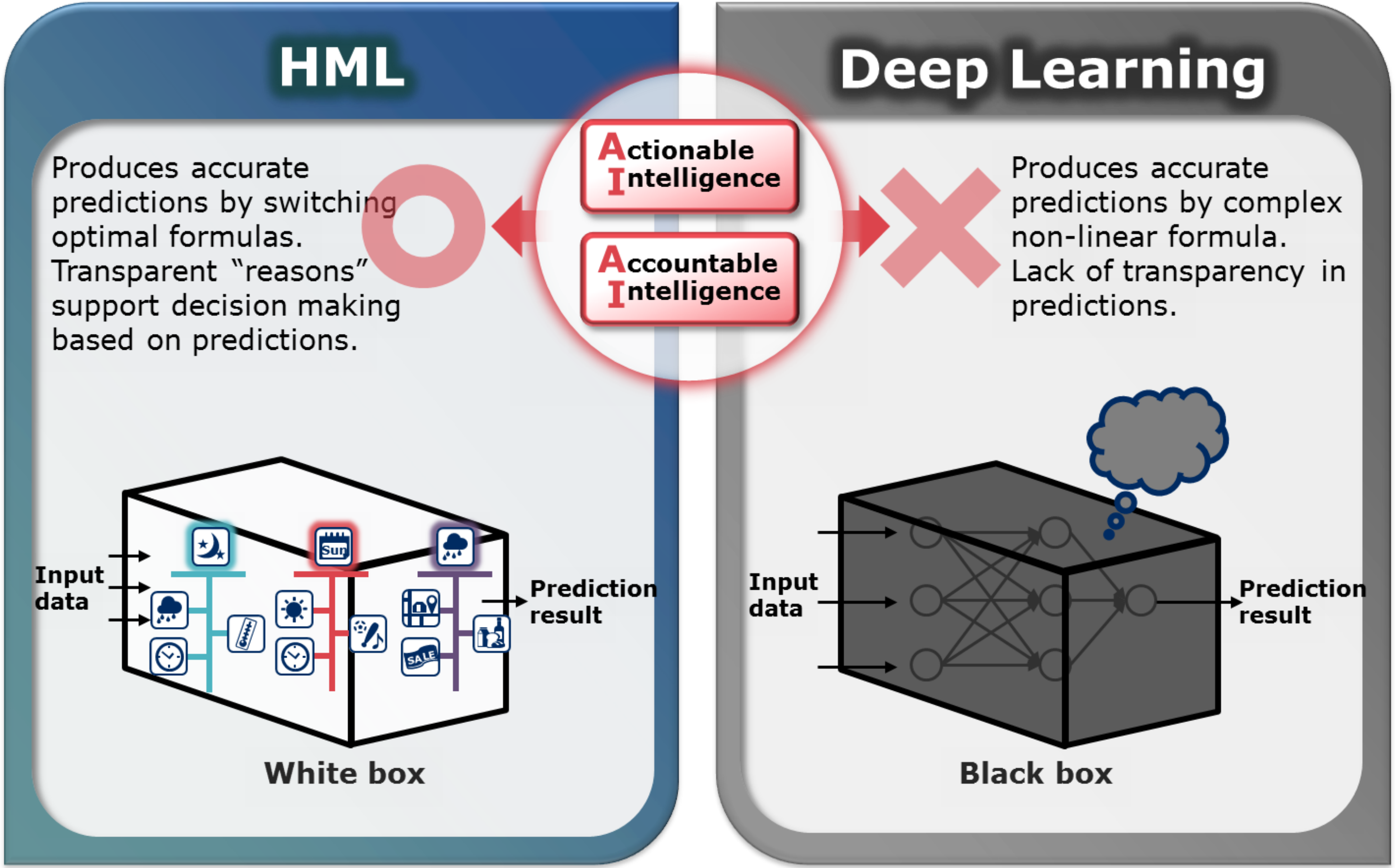


Transparent data segmentation and predictive formulas

The HML Model



HML (Heterogeneous Mixture Learning)



HML (Heterogeneous Mixture Learning)

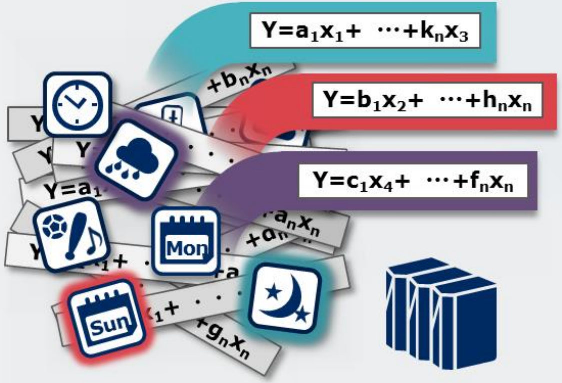
HML

Discovers highly accurate predictive formulas in a fully data-driven manner. Prediction gets smarter along with data evolution.

Autonomous Intelligence

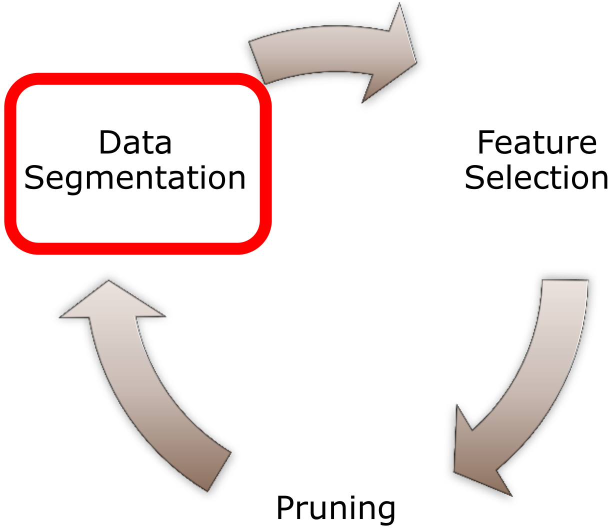
Standard Machine Learning

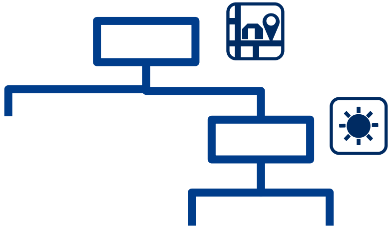
Requires trial-and-error efforts by data scientists to build good formulas. Time-consuming yet limited accuracy.



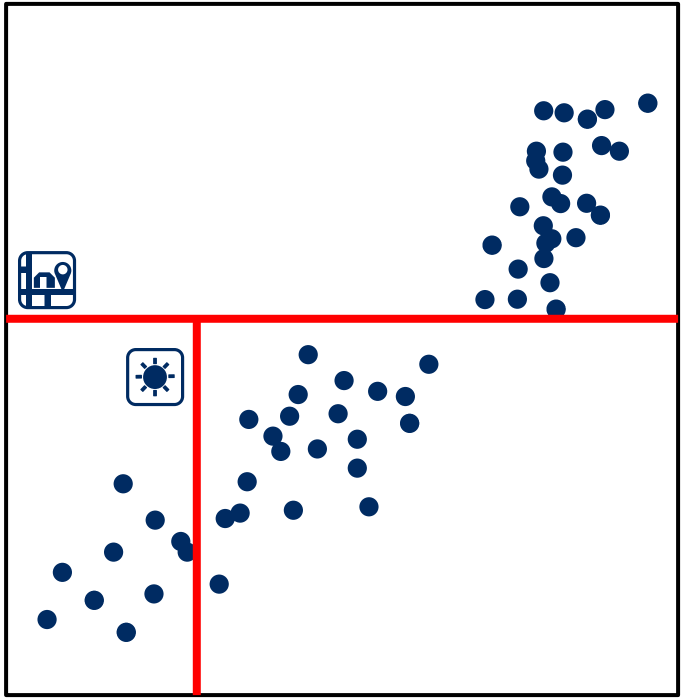
HML Algorithm

Segment data by a rule-based tree



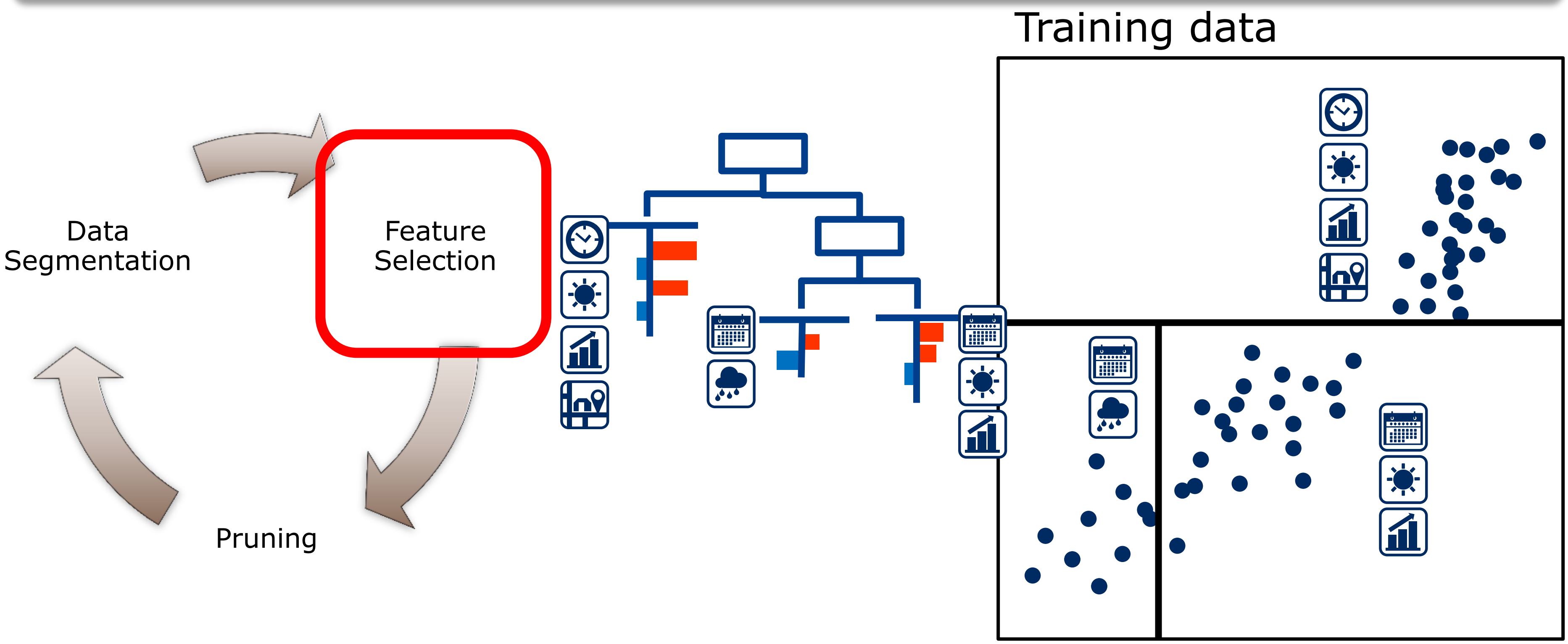


Training data



HML Algorithm

Select features and fit predictive models for data segments



Enterprise Applications of HML

Energy/Water Operation Mgmt.

Sales Optimization

Product Price Optimization

Driver Risk Assessment

Predictive Maintenance

Churn Retention

Inventory Optimization

Demand for Big Data Analysis

Energy/Water Operation Mgmt.

Sales Optimization

Product Price Optimization

24(hour)×365(days)×3(year)×1000(shops)

~26,000,000 training samples

Demand for Big Data Analysis

5 million(customers)×12(months)
=60,000,000 training samples

Driver Risk Assessment

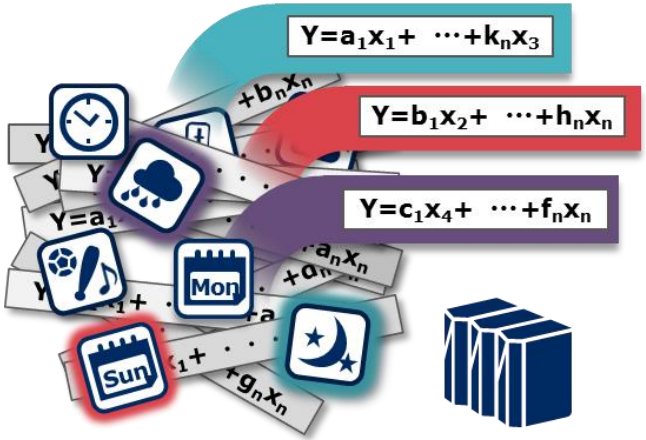
Predictive Maintenance

Churn Retention

Inventory Optimization

Demand for Big Data Analysis

Energy/Water Operation Mgmt. | **Sales Optimization** | **Product Price Optimization**

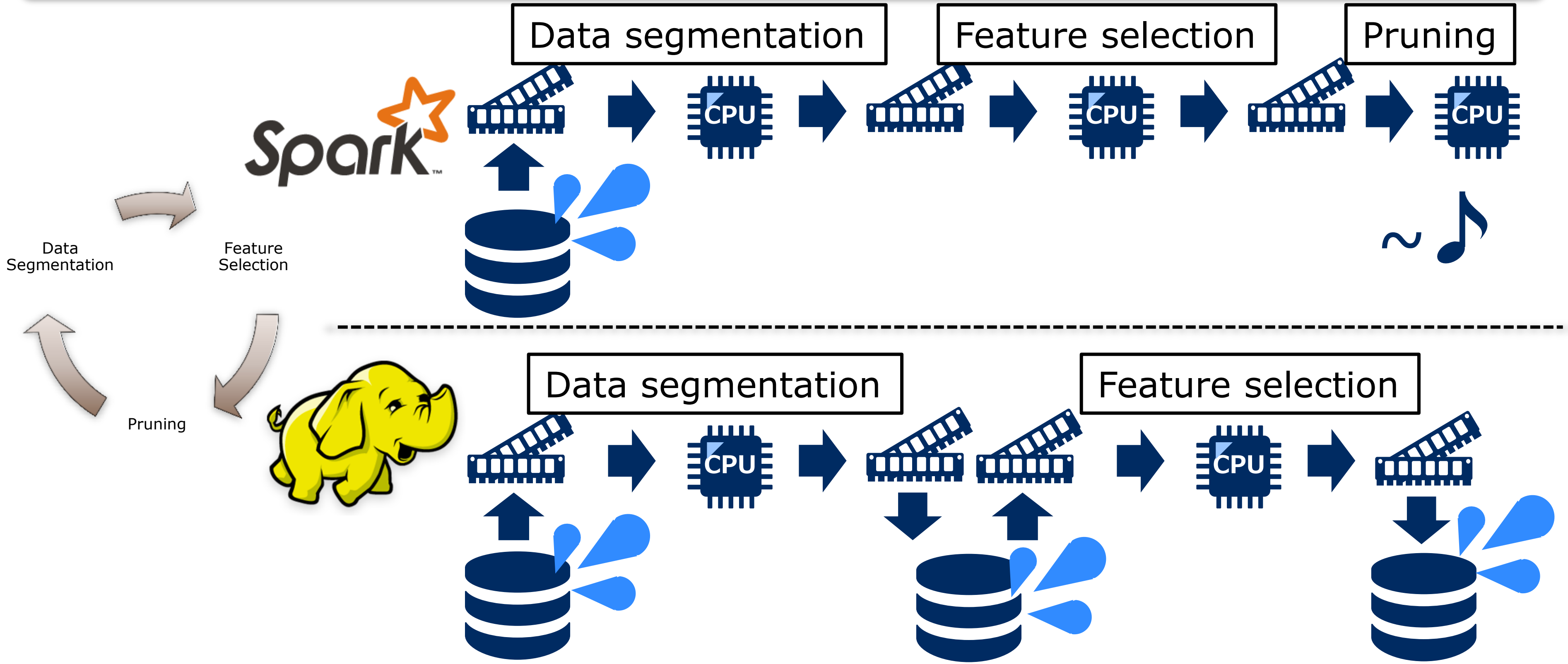


Spark

Distributed HML on Spark

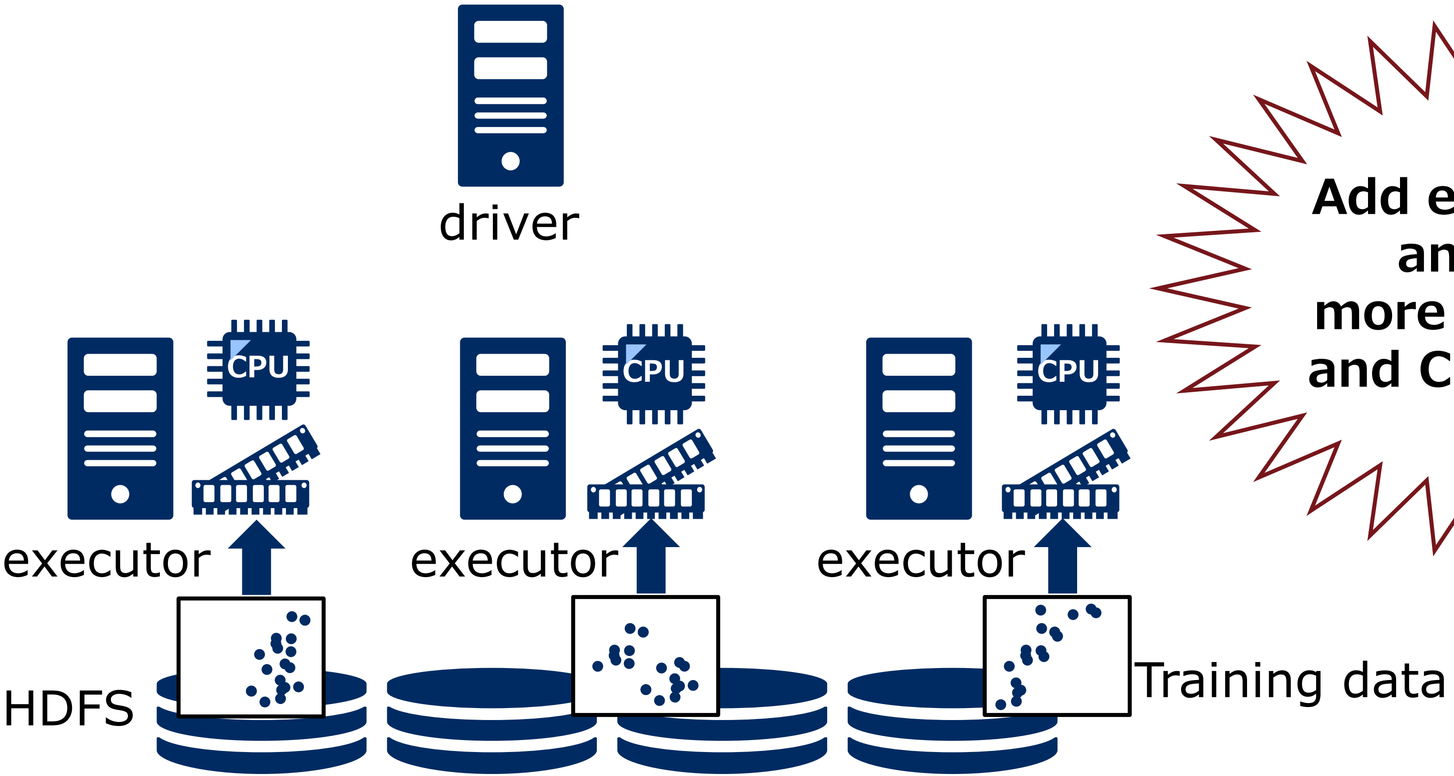
Why Spark, not Hadoop?

Because Spark's in-memory architecture can run HML faster



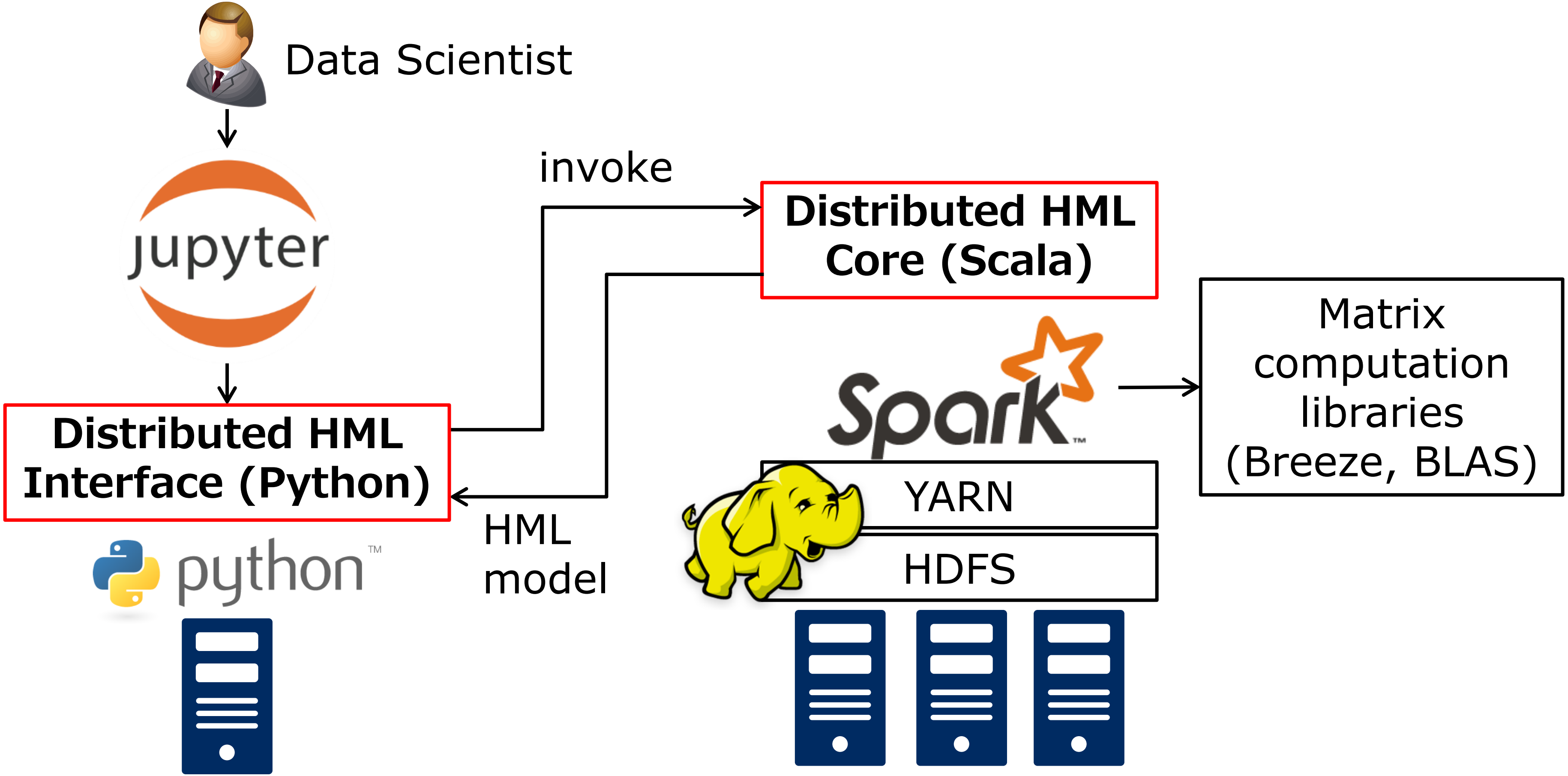
Data Scalability powered by Spark

Treat unlimited scale of training data by adding executors



Add executors, and get more memory and CPU power

Distributed HML Engine: Architecture

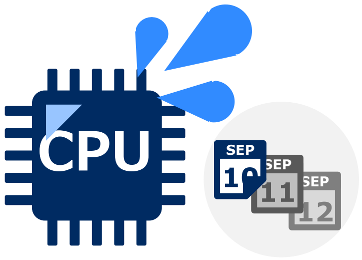


3 Technical Key Points to Fast Run ML on Spark

Data Shuffling

MapReduce Synchronization

Matrix Computation

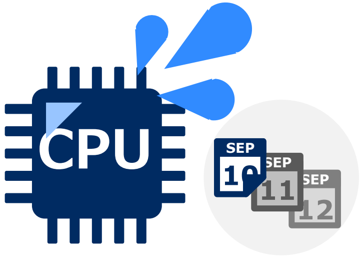


3 Technical Key Points to Fast Run ML on Spark

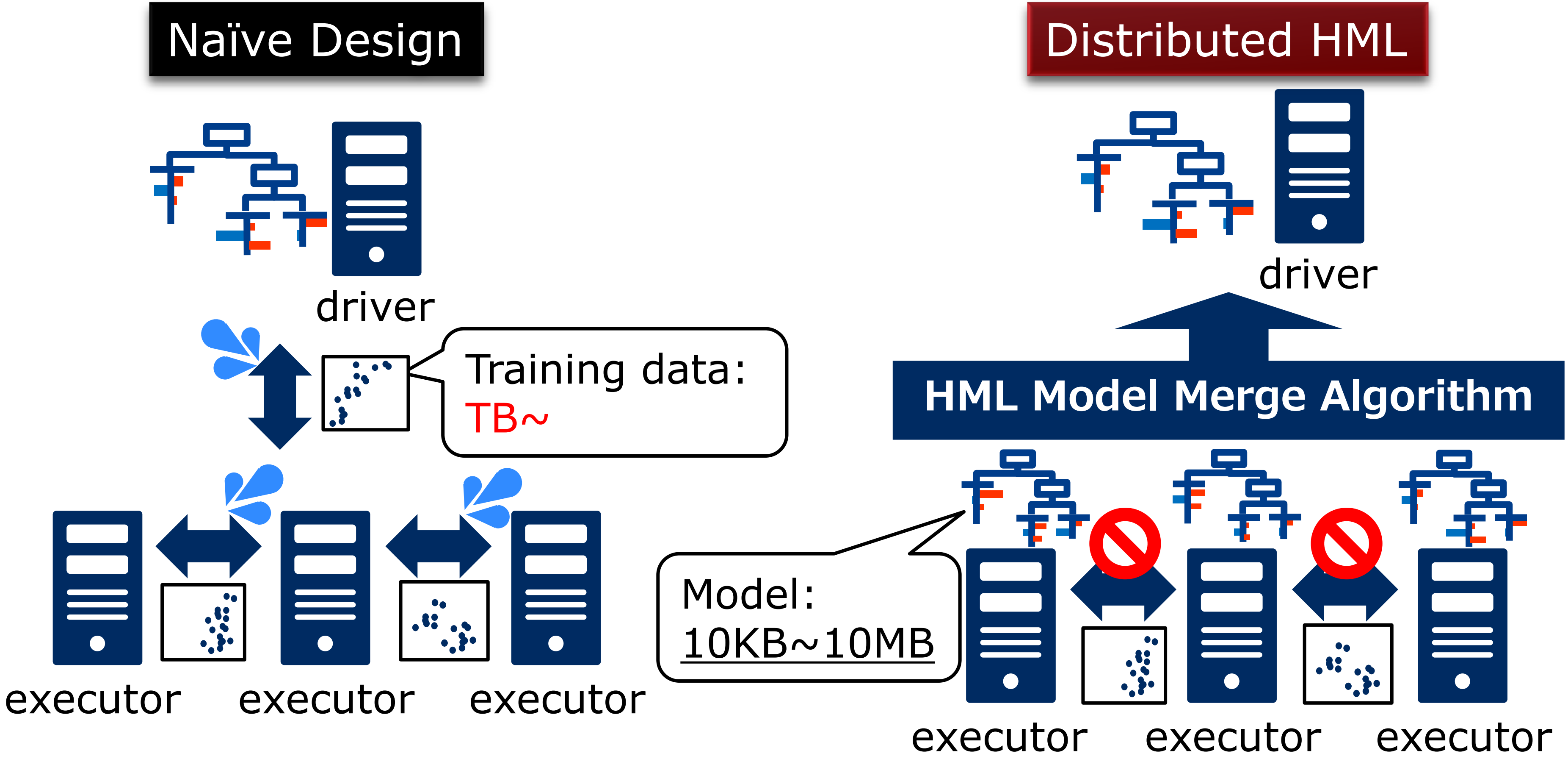
Data Shuffling

MapReduce Synchronization

Matrix Computation

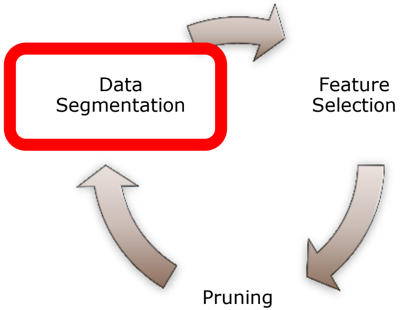


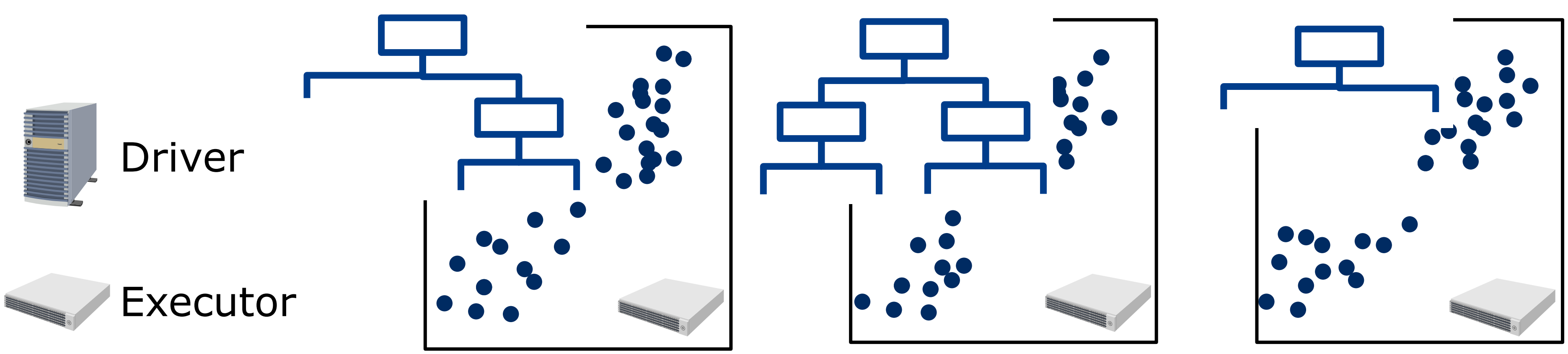
Challenge to Avoid Data Shuffling



Execution Flow of Distributed HML

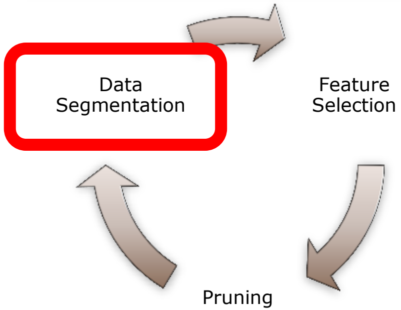
Executors build a rule-based tree from their local data in parallel

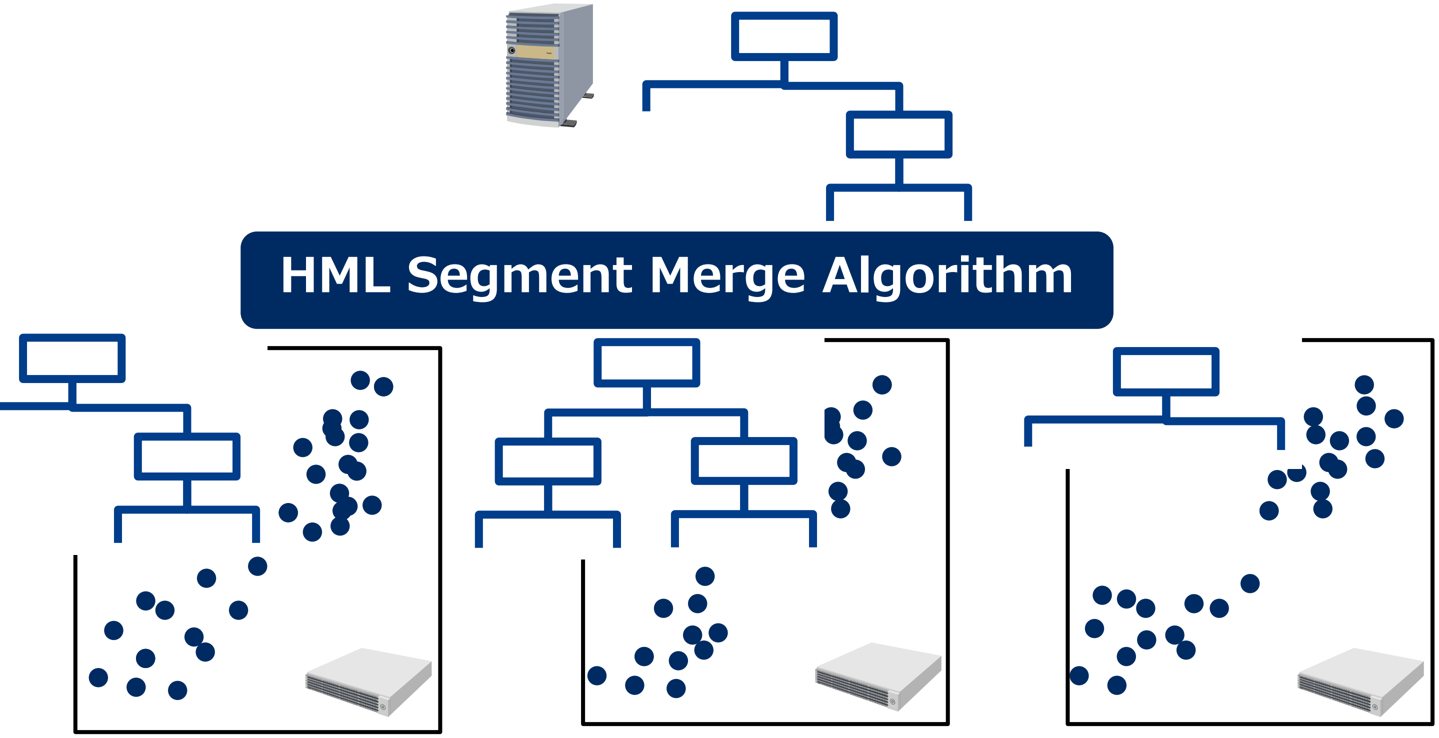




Execution Flow of Distributed HML

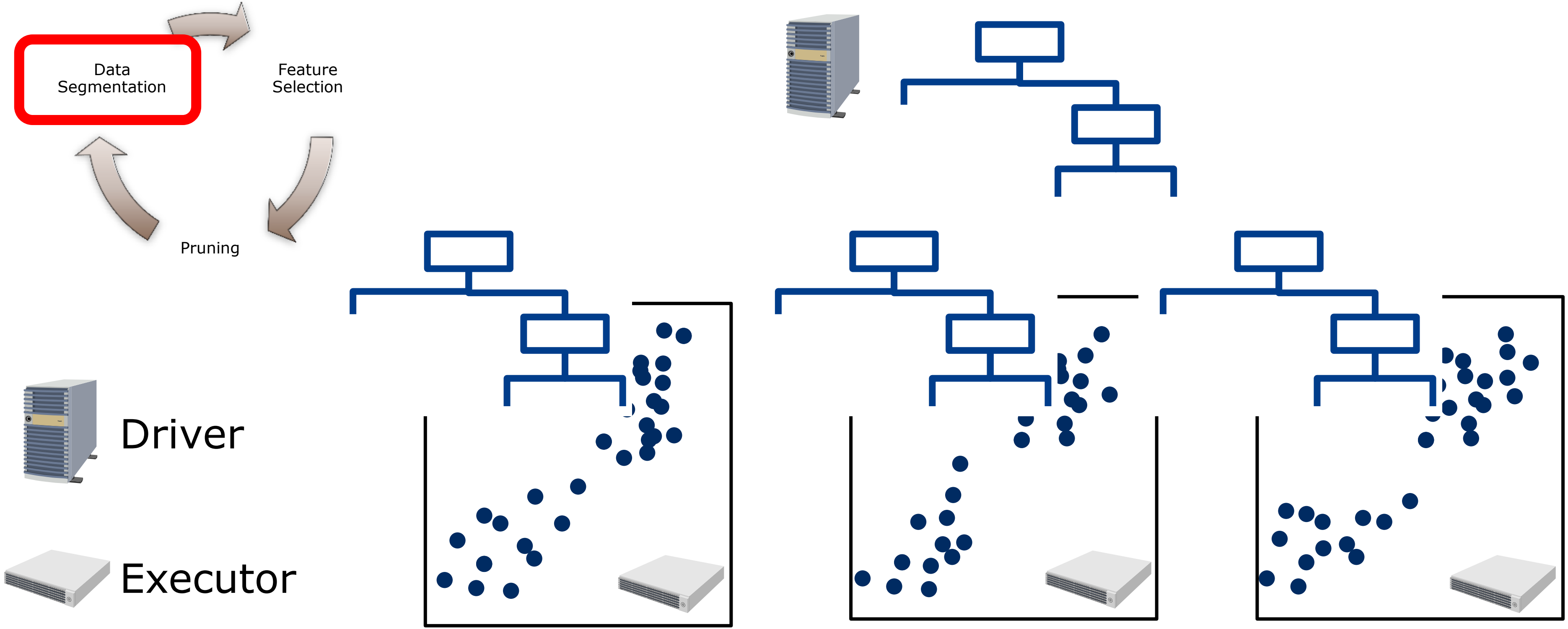
Driver merges the trees





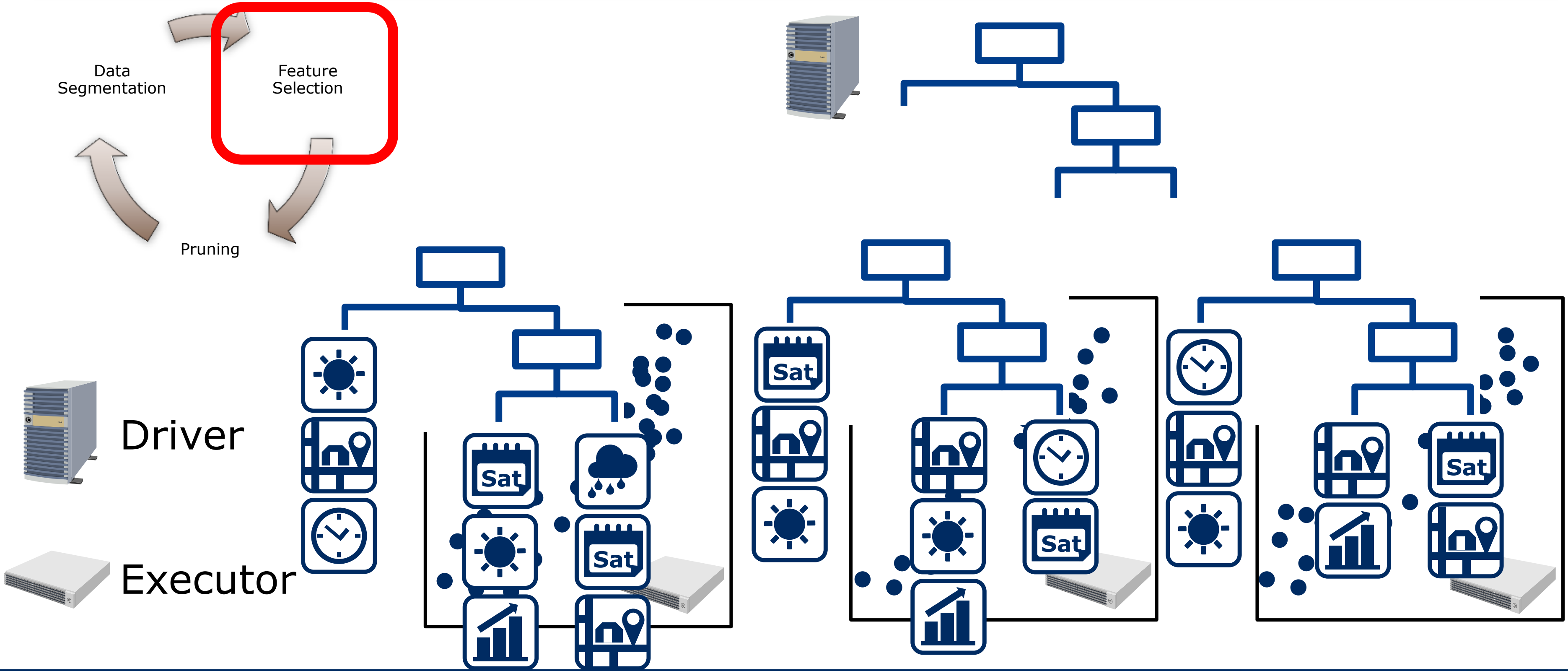
Execution Flow of Distributed HML

Driver broadcasts the merged tree to executors



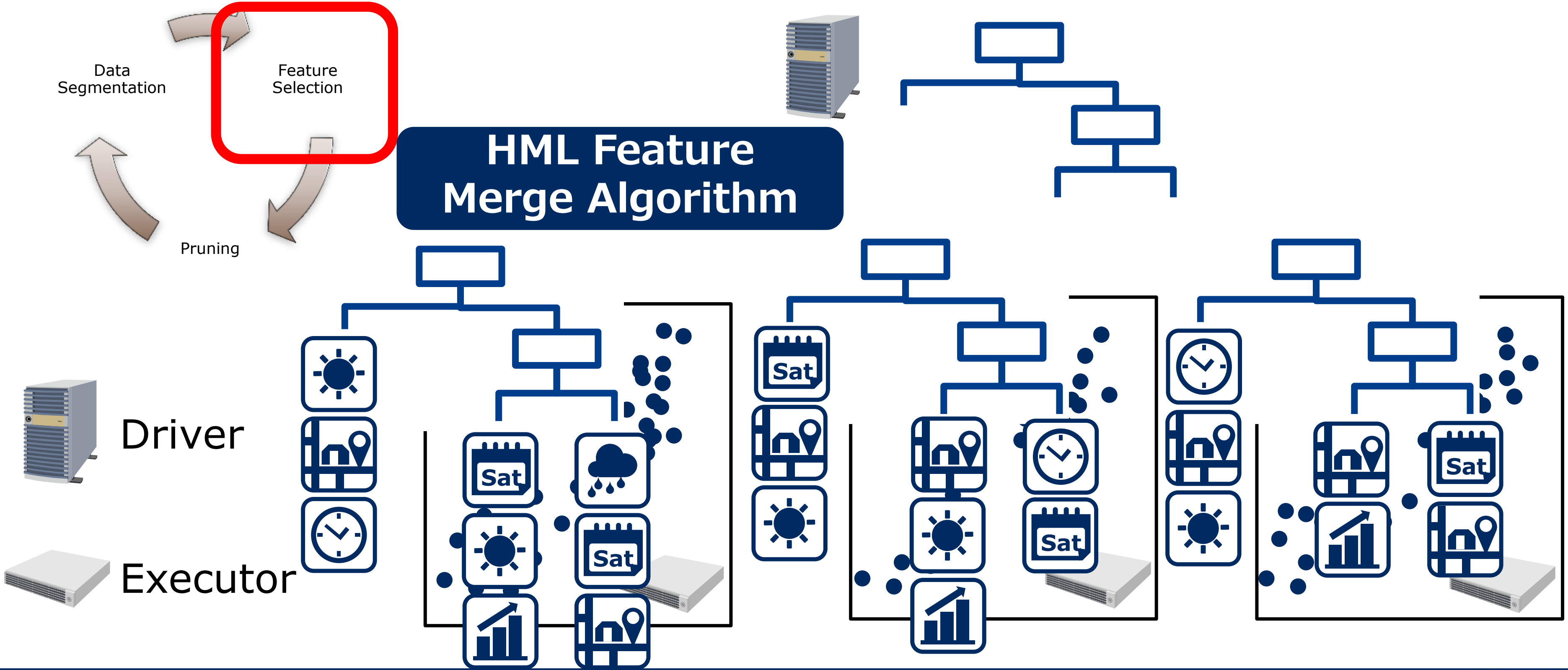
Execution Flow of Distributed HML

Executors perform feature selection with their local data in parallel



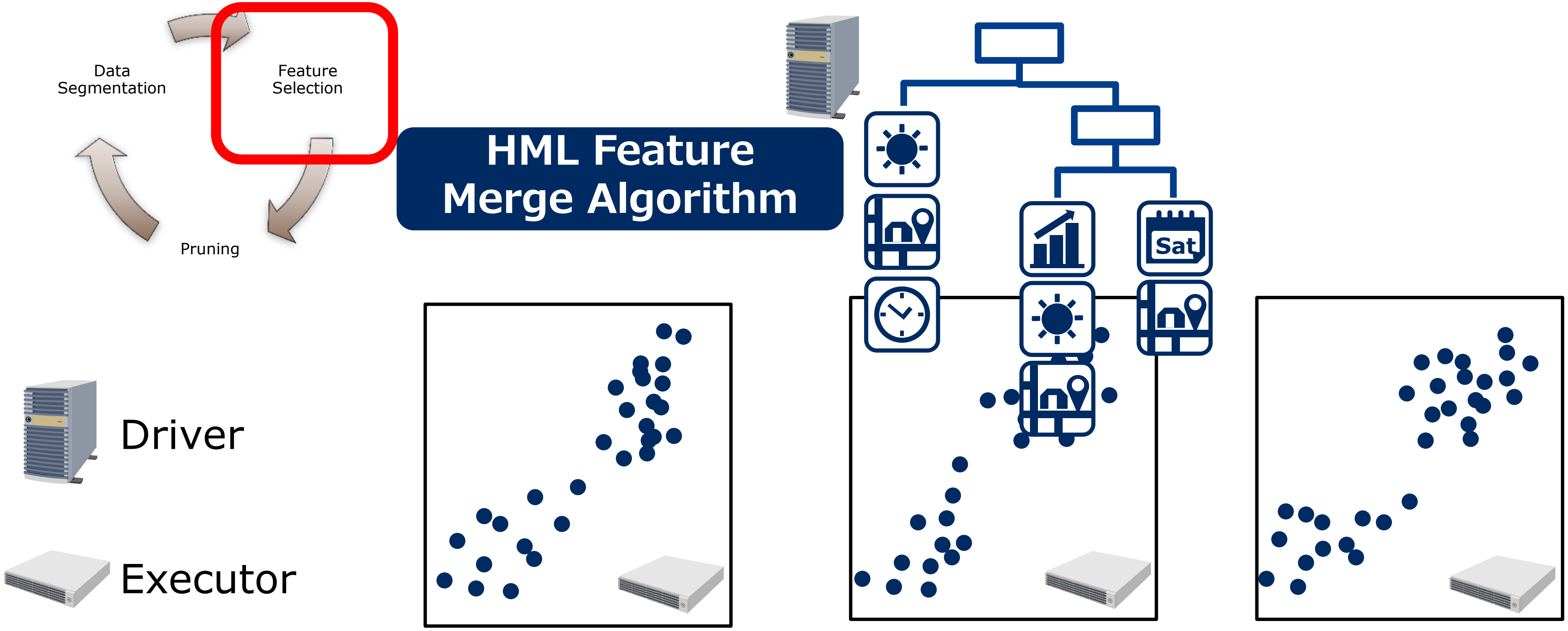
Execution Flow of Distributed HML

Driver merges the results of feature selection



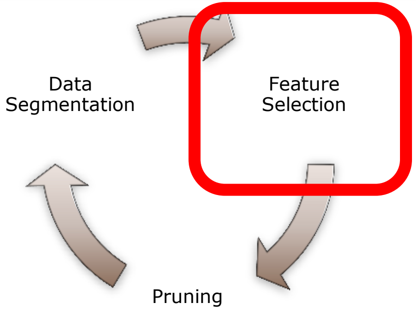
Execution Flow of Distributed HML

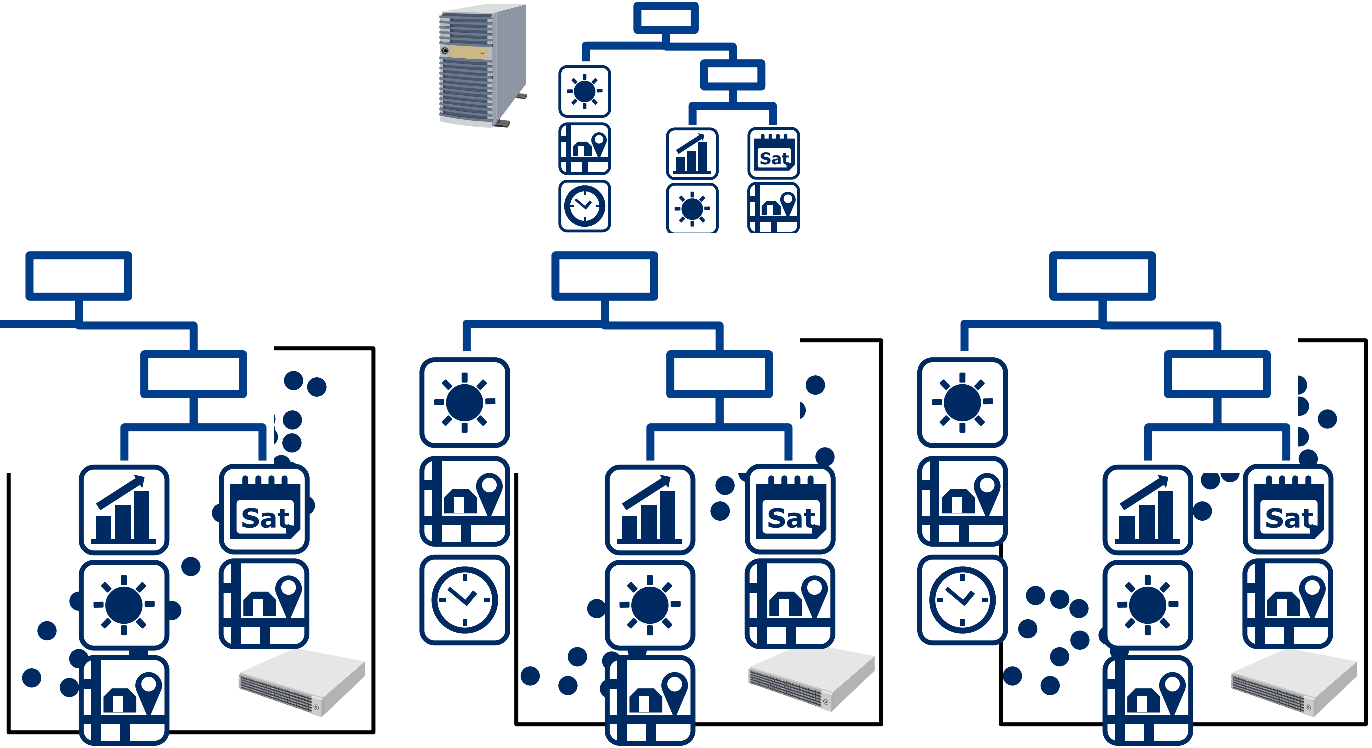
Driver merges the results of feature selection

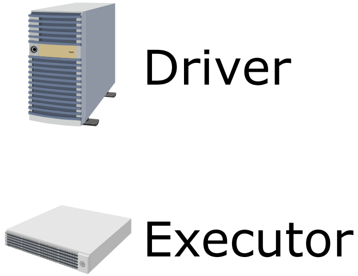


Execution Flow of Distributed HML

Driver broadcasts the merged results of feature selection to executors

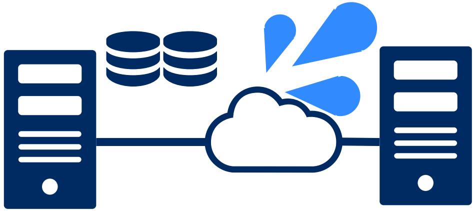






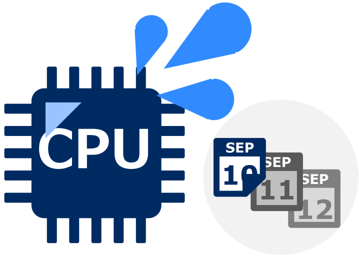
3 Technical Key Points to Fast Run ML on Spark

Data Shuffling



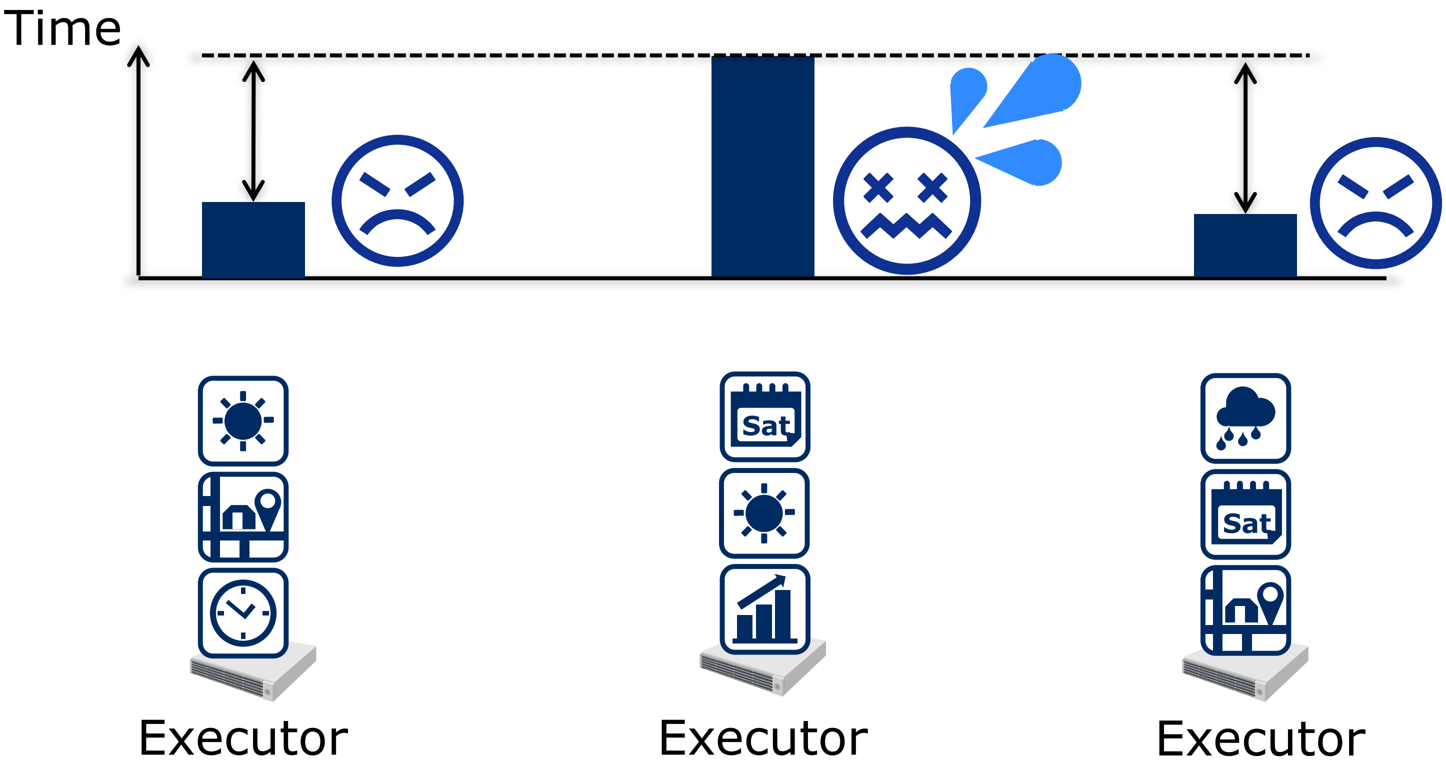
MapReduce Synchronization

Matrix Computation



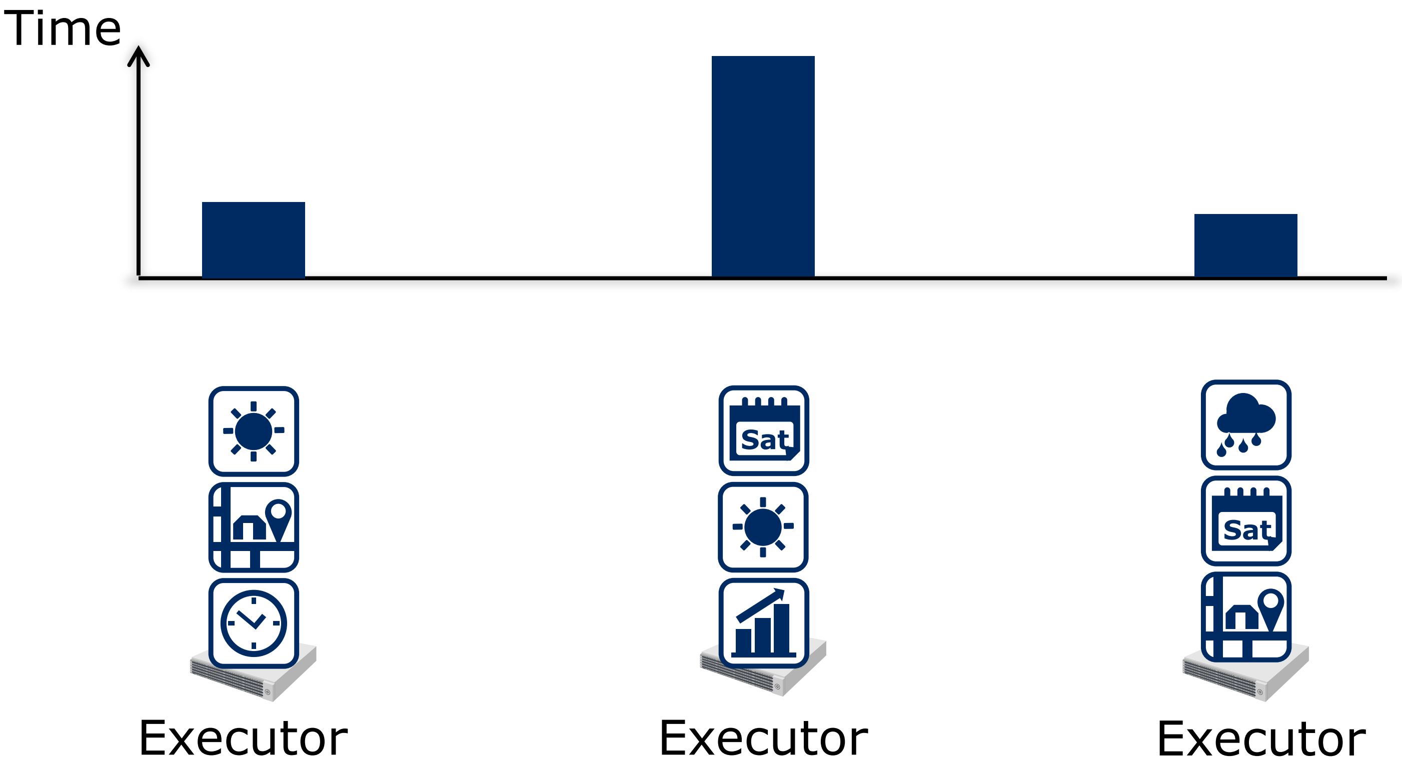
Wait Time for Other Executors Delays Execution

Machine learning is likely to cause unbalanced computation load



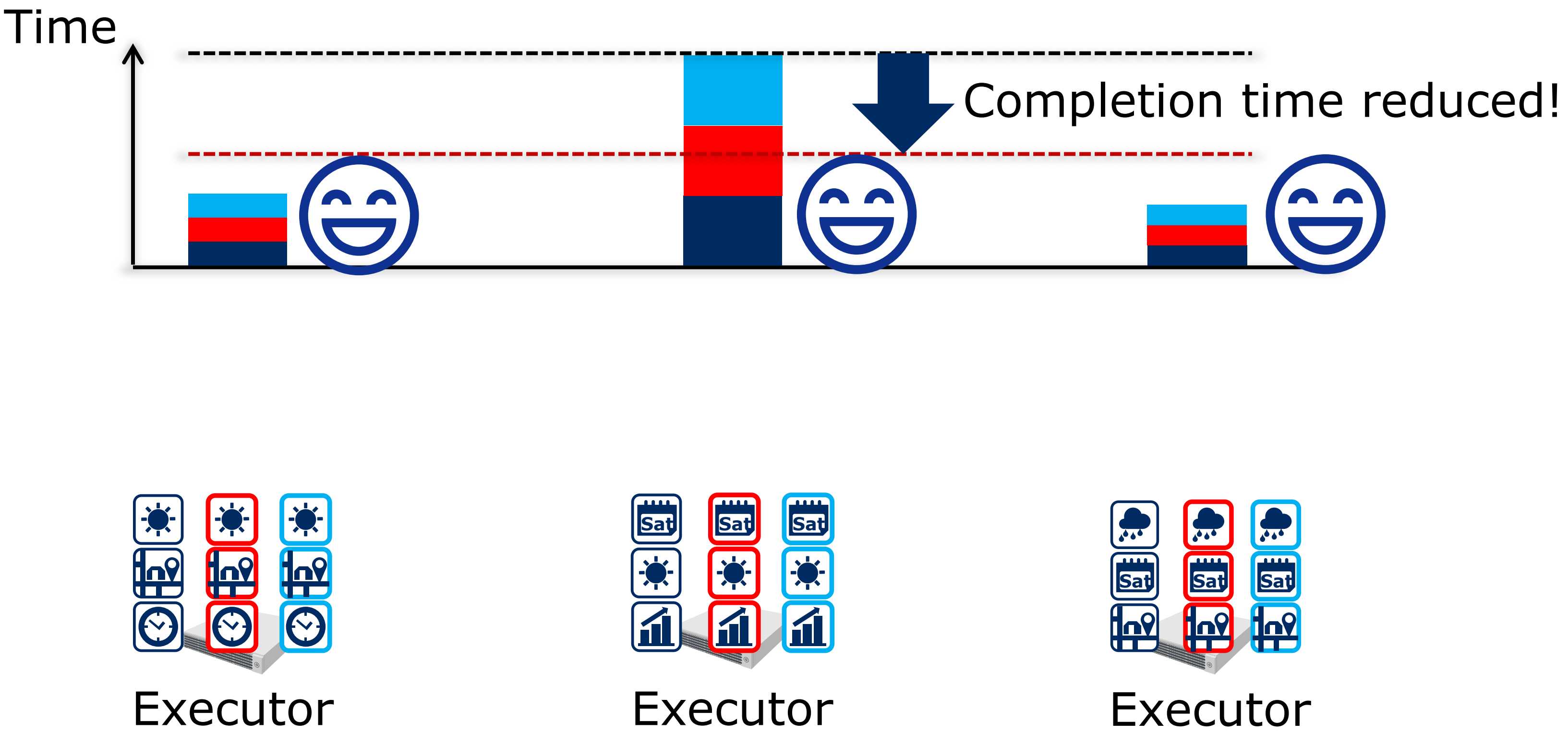
Balance Computational Effort for Each Executor

Executor optimizes all predictive formulas with equally-divided training data



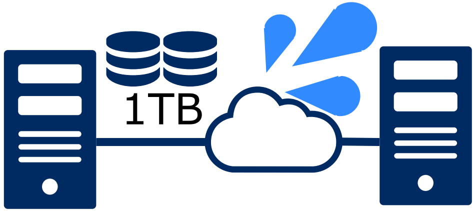
Balance Computational Effort for Each Executor

Executor optimizes all predictive formulas with equally-divided training data



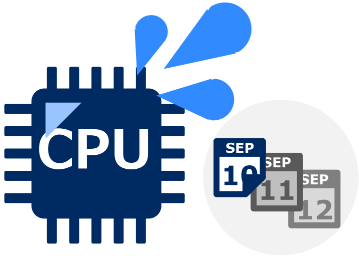
3 Technical Key Points to Fast Run ML on Spark

Data Shuffling

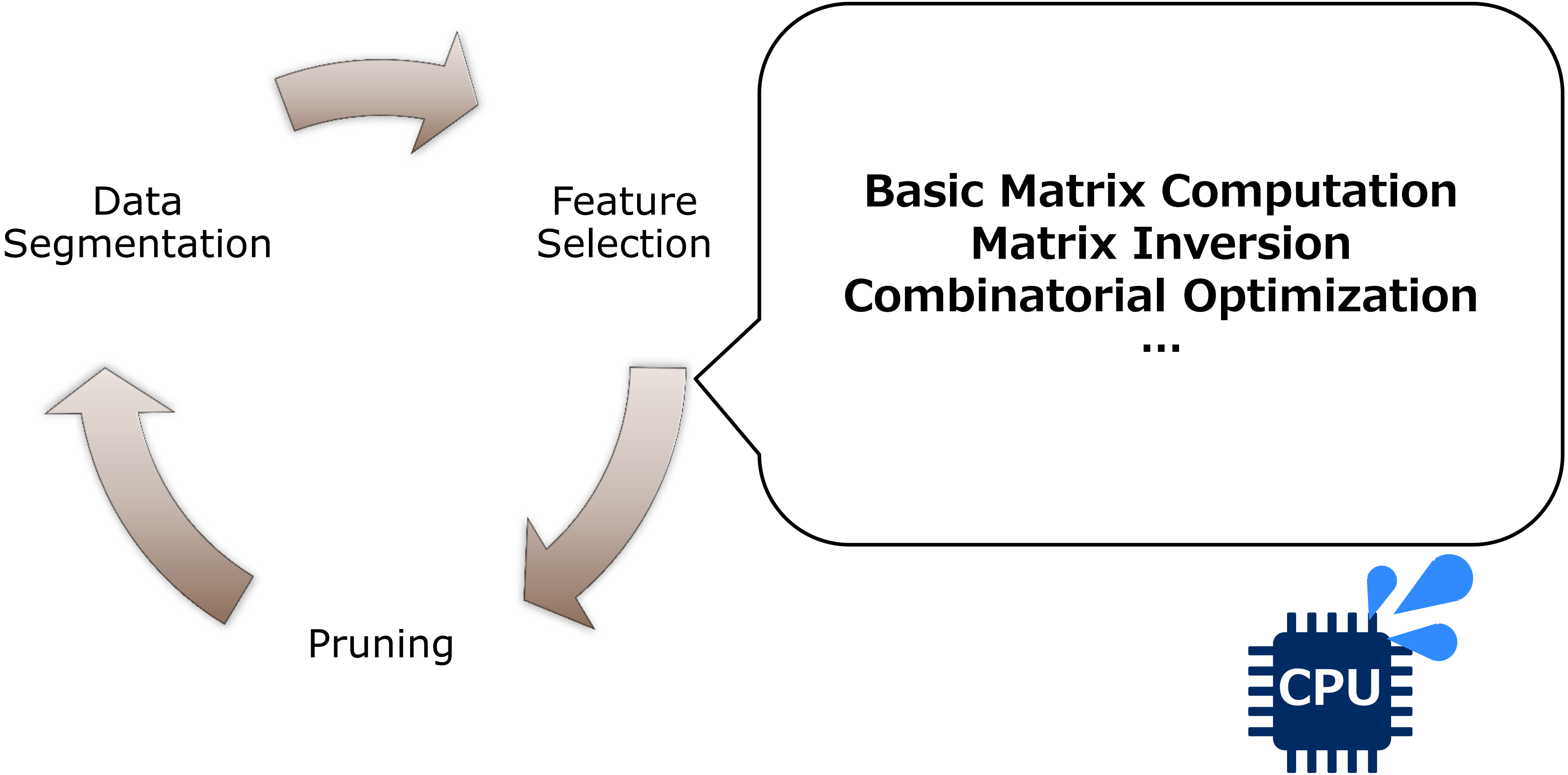


MapReduce Synchronization

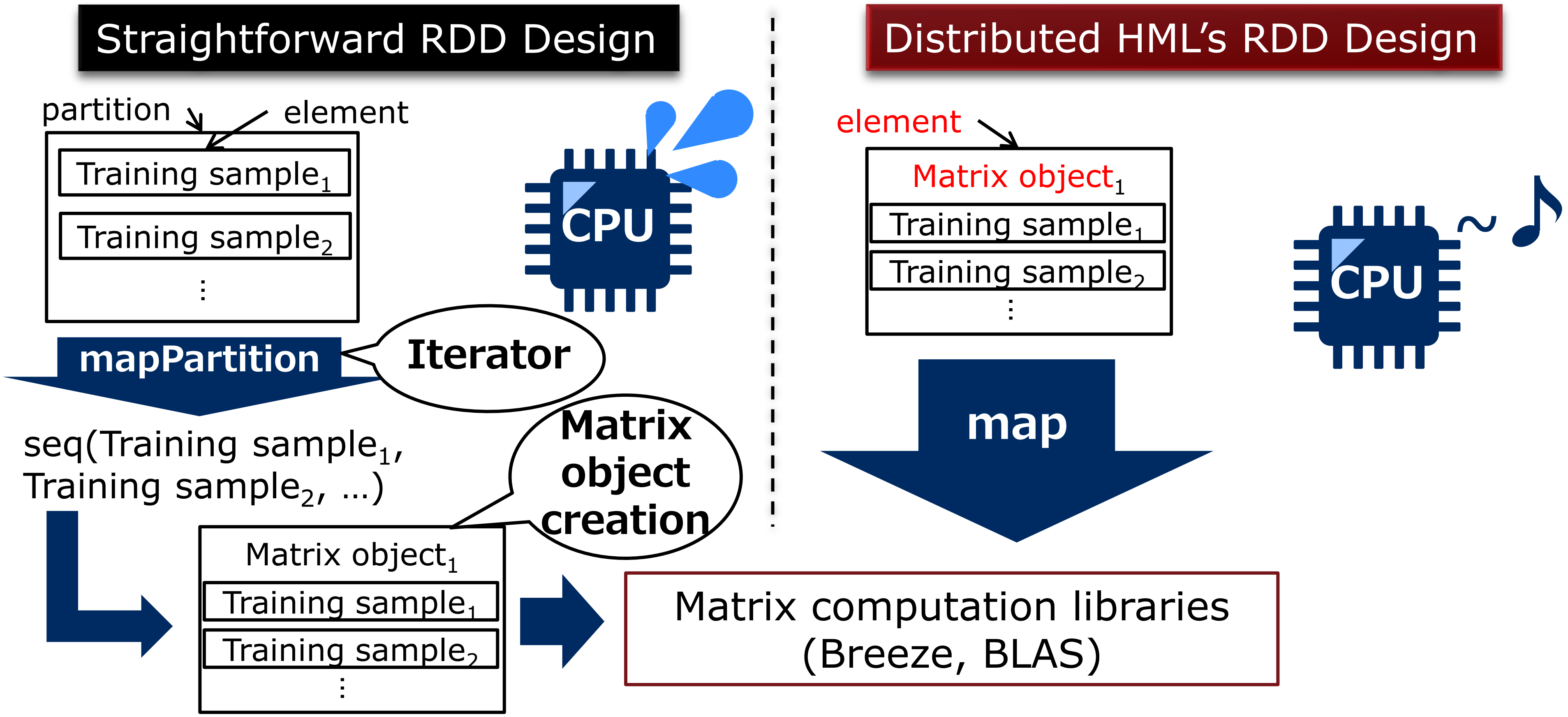
Matrix Computation



Machine Learning Consists of Many Matrix Computations

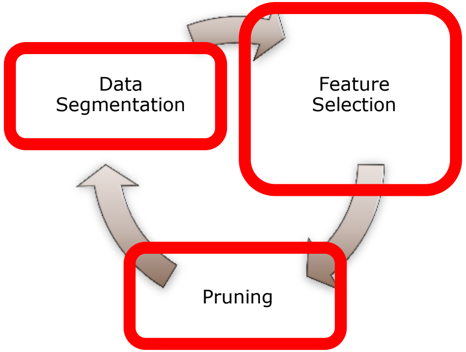


RDD Design for Leveraging High-Speed Libraries



Performance Degradation by Long-Chained RDD

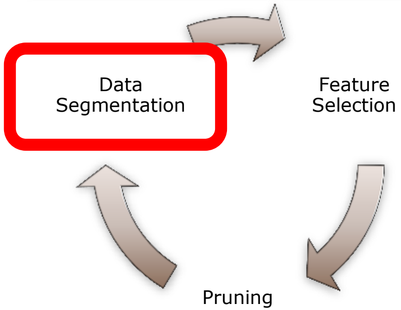
Long-chained RDD operations cause high-cost recalculations



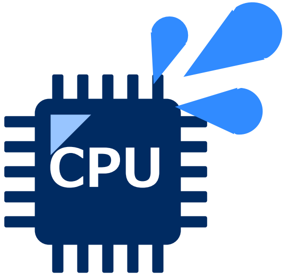

```
RDD .map(TreeMapFunc)
    .reduce(TreeReduceFunc)
    .map(FeatureSelectionMapFunc)
    .reduce(FeatureSelectionReduceFunc)
    .map(PruningMapFunc)
    .map(TreeMapFunc)
    .reduce(TreeReduceFunc)
    .map(FeatureSelectionMapFunc)
    .reduce(FeatureSelectionReduceFunc)
```

Performance Degradation by Long-Chained RDD

Long-chained RDD operations cause high-cost recalculations

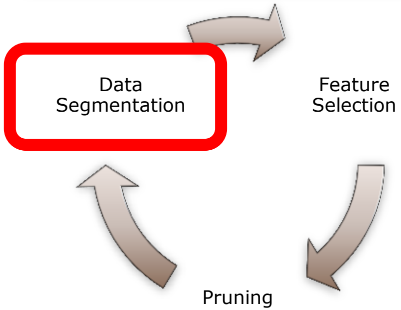



```
RDD .map(TreeMapFunc)
    .reduce(TreeReduceFunc)
    .map(FeatureSelectionMapFunc)
    .reduce(FeatureSelectionReduceFunc)
    .map(PruningMapFunc)
    .map(TreeMapFunc)
    .reduce(TreeReduceFunc)
    .map(FeatureSelectionMapFunc)
    .reduce(FeatureSelectionReduceFunc)
    .map(TreeMapFunc)
```

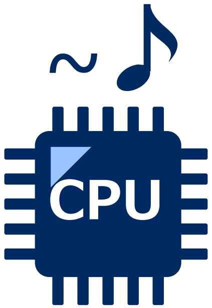


Performance Degradation by Long-Chained RDD

Cut long-chained operations periodically by checkpoint()




```
savedRDD = RDD.map(TreeMapFunc)
              .reduce(TreeReduceFunc)
              .map(FeatureSelectionMapFunc)
              .reduce(FeatureSelectionReduceFunc)
              .map(PruningMapFunc)
              .map(TreeMapFunc)
              .reduce(TreeReduceFunc)
              .map(FeatureSelectionMapFunc)
              .reduce(FeatureSelectionReduceFunc)
              .checkpoint()
              .map(TreeMapFunc)
```



Benchmark Performance Evaluations

Eval 1: Prediction Error (vs. Spark MLlib algorithms)

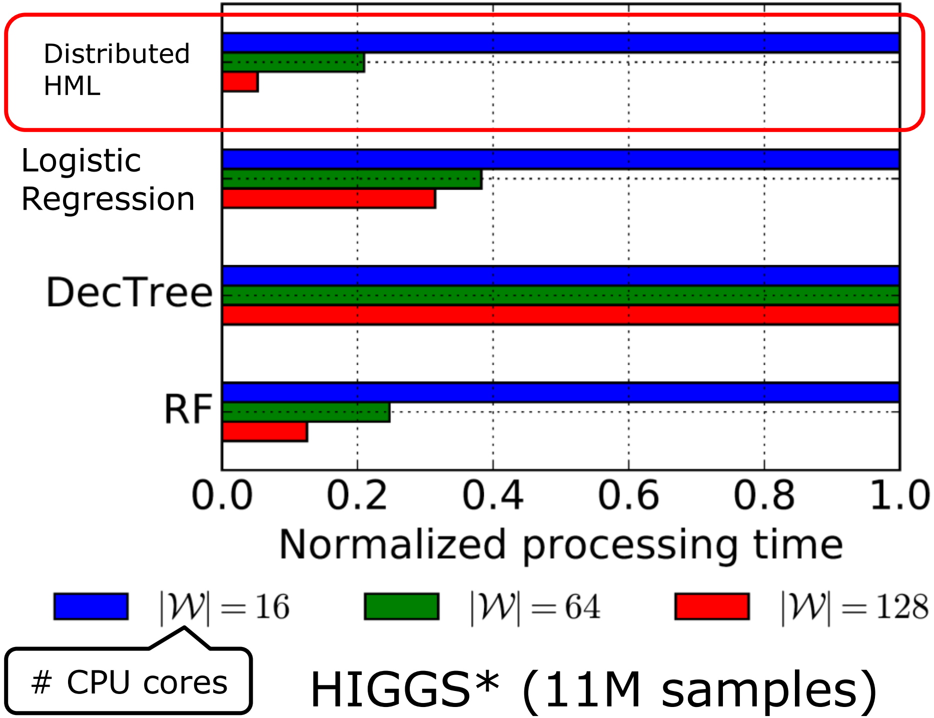
Distributed HML achieves low error competitive to a complex model

data	# samples	Distributed HML	Logistic Regression	Decision Tree	Random Forests
gas sensor array (CO)*	4,208,261	1st 0.542	0.597	0.587	0.576
household power consumption *	2,075,259	1st 0.524	0.531	0.529	0.655
HIGGS*	11,000,000	2nd 0.335	0.358	0.337	0.317
HEPMASS*	7,000,000	1st 0.156	0.163	0.167	0.175

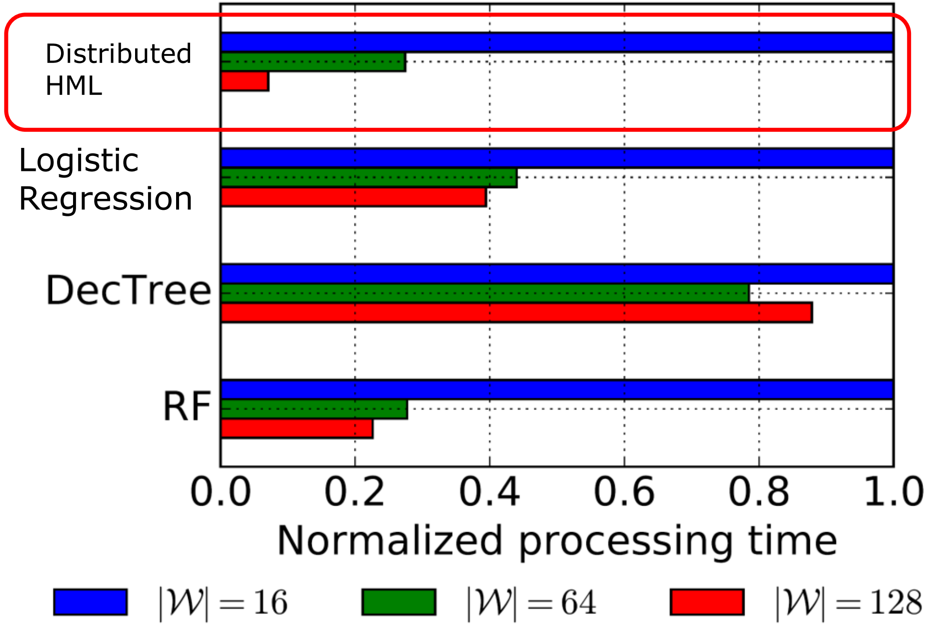
* UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/)

Eval 2: Performance Scalability (vs. Spark MLlib algos)

Distributed HML is competitive with Spark MLlib implementations.



HIGGS* (11M samples)



HEPMASS* (7M samples)

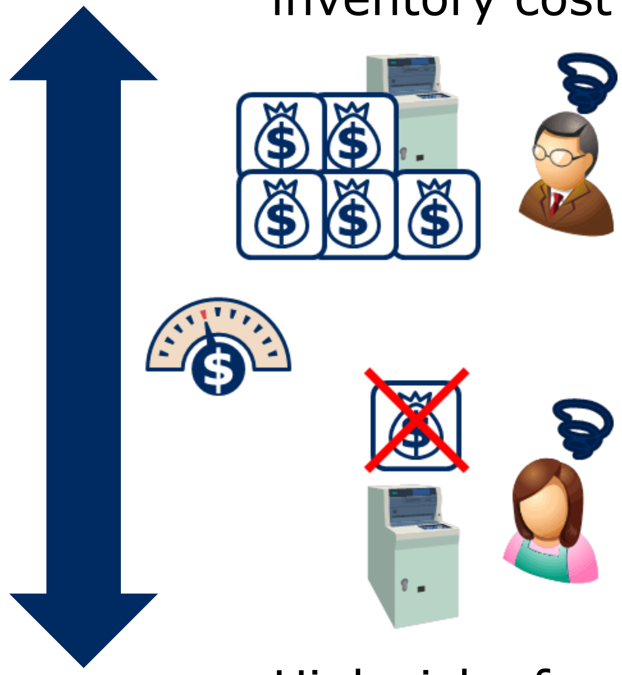
* UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/)

Evaluation in Real Case

ATM Cash Balance Prediction

Around 20,000 ATMs

Much cash ➔ High inventory cost



Little cash ➔ High risk of out of stock

Training Speed

Serial HML*

9 days

110x
Speed up

Distributed HML

2 hours

* Run w/ 1 CPU core and 256GB memory

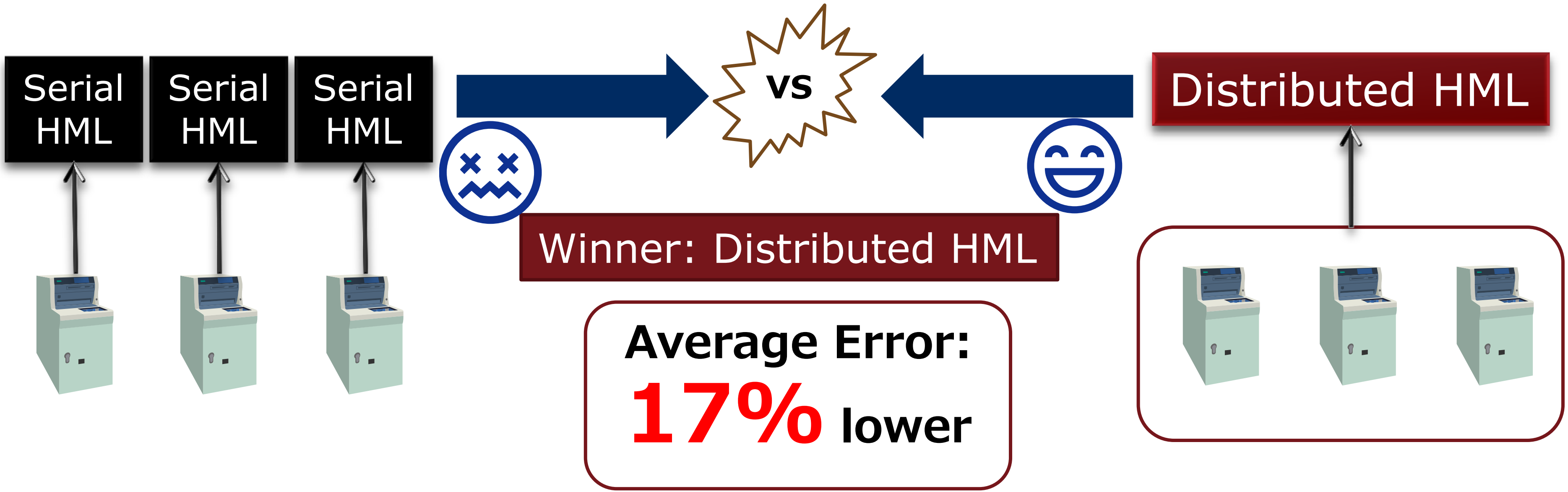
Summary of data

- \# ATMs: around 20,000 (in Japan)
- \# training samples: around 10M

Cluster spec (10 nodes)

- \# CPU cores: 128
- Memory: 2.5TB
- Spark 1.6.1,
  Hadoop 2.7.1 (HDP 2.3.2)

Prediction Error



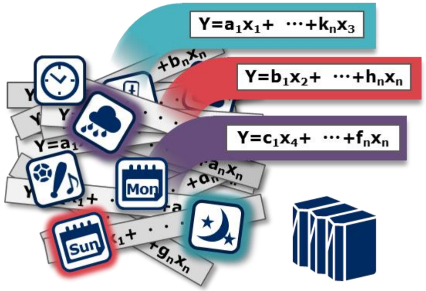
Summary of data

- # ATMs: around 20,000 (in Japan)
- # training samples: around 23M

Cluster spec (10 nodes)

- # CPU cores: 128
- Memory: 2.5TB
- Spark 1.6.1,
  Hadoop 2.7.1 (HDP 2.3.2)

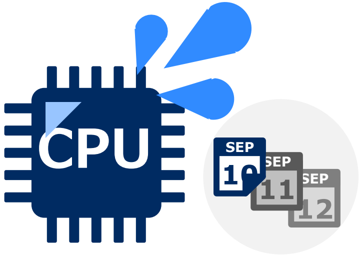
Summary



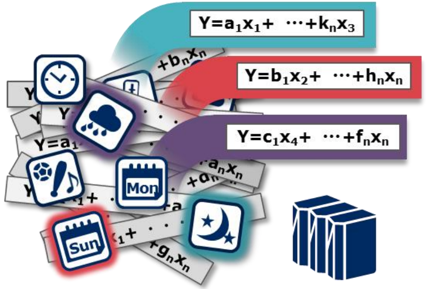
Data Shuffling

MapReduce Synchronization

Matrix Computation



Summary



\Orchestrating a brighter world

NEC