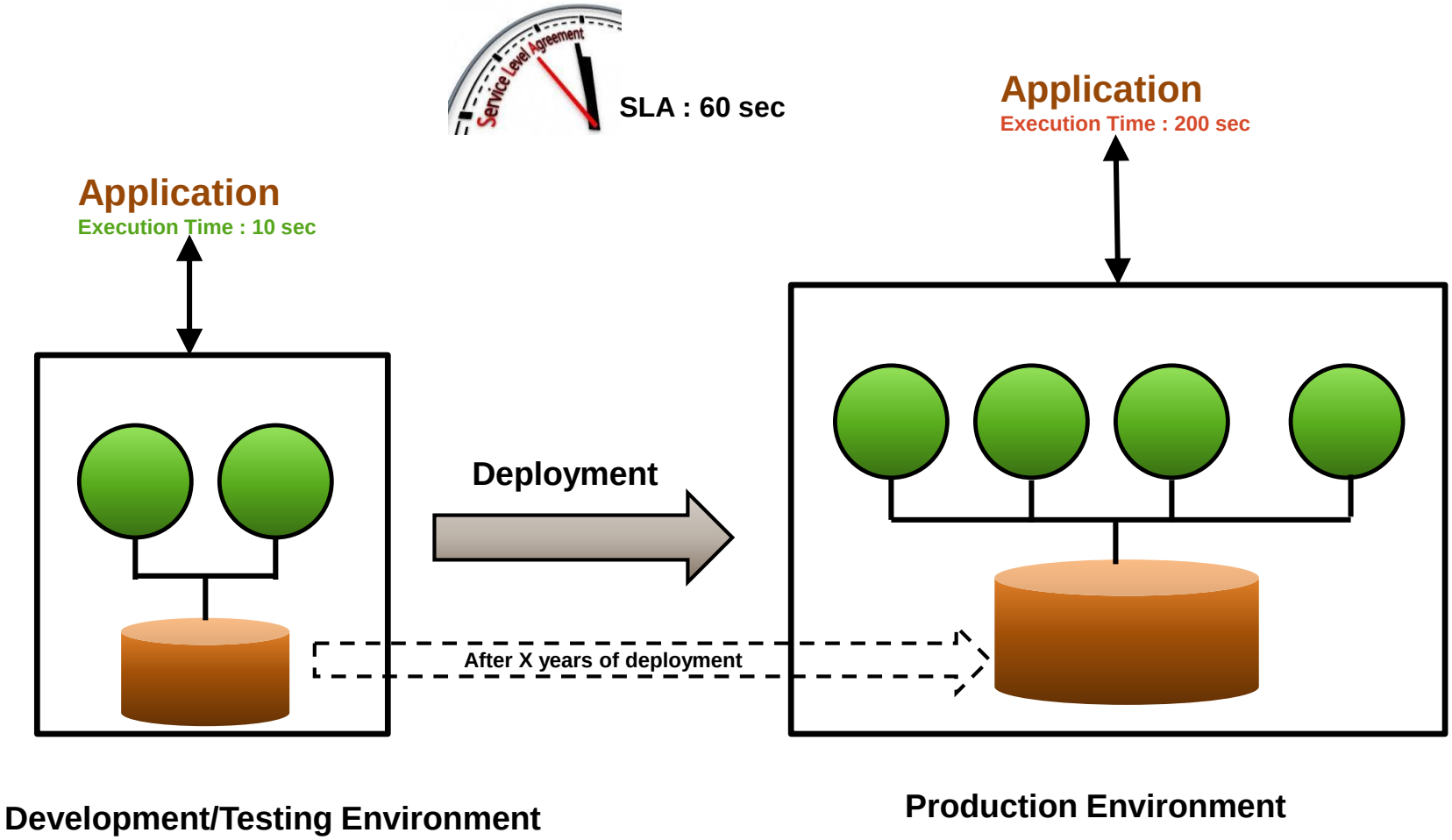
TATA CONSULTANCY SERVICES
Experience certainty.

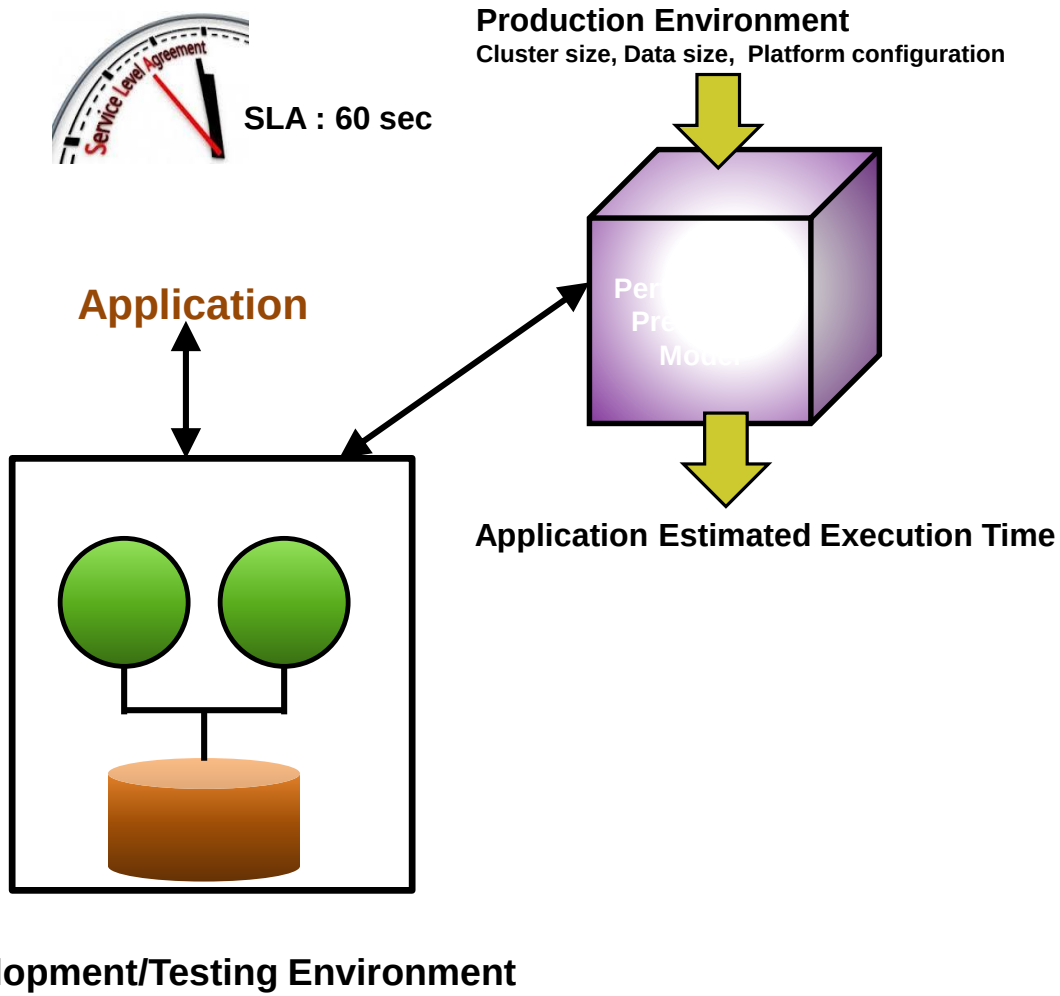
Spark Job Performance Analysis and Prediction Tool

Rekha Singhal

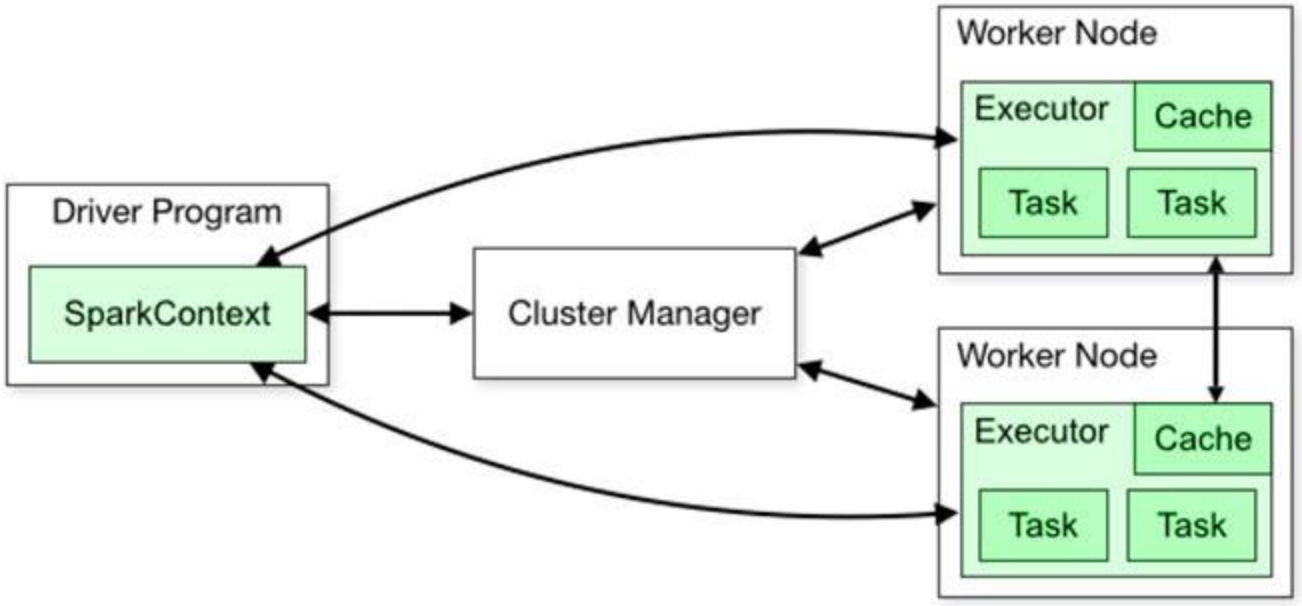
Problem Motivation



Problem Solution



Spark Architecture



source:https://intellipaat.com/tutorial/spark-tutorial/spark-architecture/

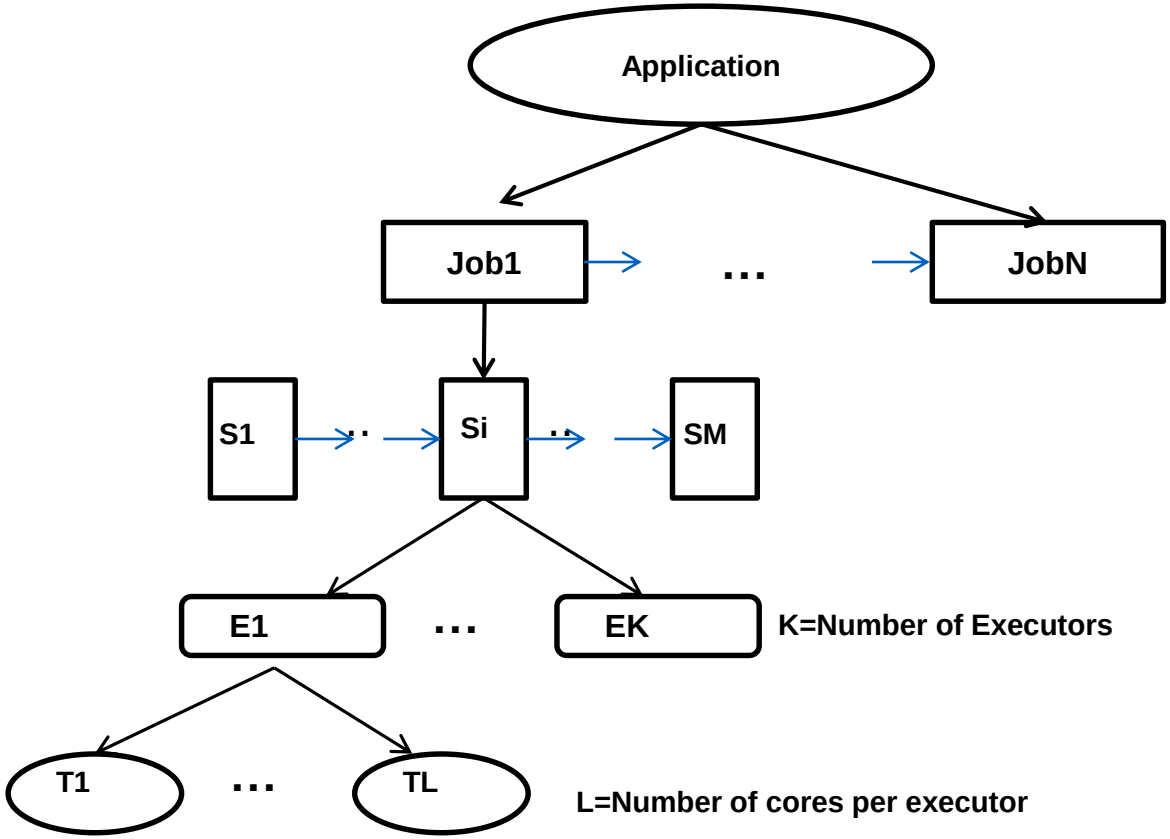
Assumptions

- Development/Testing environment has at least one instance for each type of node in production environment
- Application representative small size data sets are available.
- Focus on 3 parameters - #executors, #cores per executor, ExecutorMemorySize
- Good network connectivity in the cluster

Approach

- Set up small size Spark cluster with one instance of each node type in production
- Execute the application in development environment with given small data size.
- Collect Spark logs created during application execution
- Parse the log and collect parameters used in the model
- Build the prediction model using the collected measurements
- Apply to model for give production environment – data size, Spark parameters and cluster size.

Application Execution on Spark



Prediction of Application Execution Time

$$\sum_{i=0}^{i=N} pJob_i$$

'i'th Job Estimated Execution Time in production environment

Estimated Execution time of 'j'th stage of job 'i in production

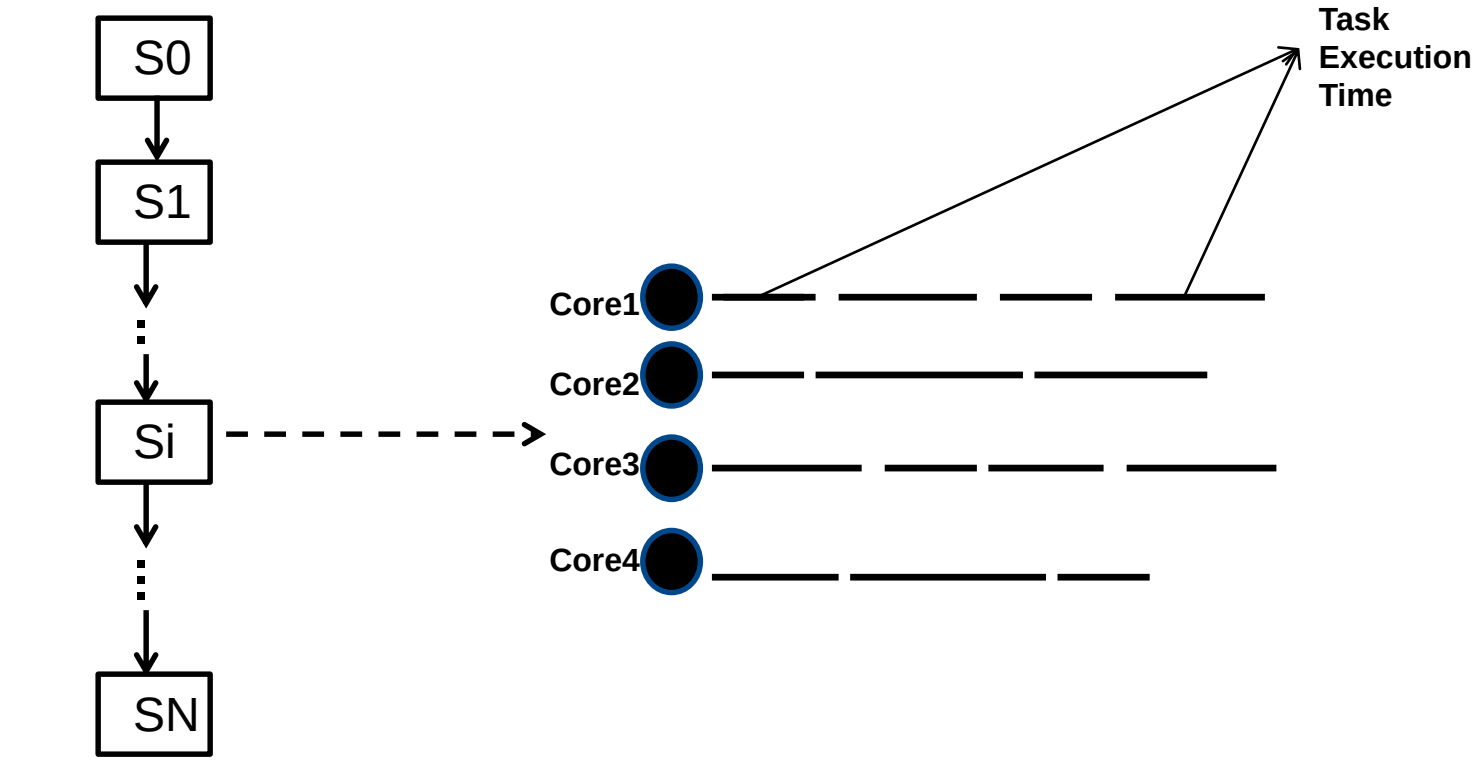
Job startup overheads

Job cleanup overheads

$$pJob_i = JobStart_i + \sum_{j=0}^{j=SN_i} pStage_i^{\,j} + JobCleanup_i$$

STAGE EXECUTION SIMULATOR for ESTIMATIONS !!

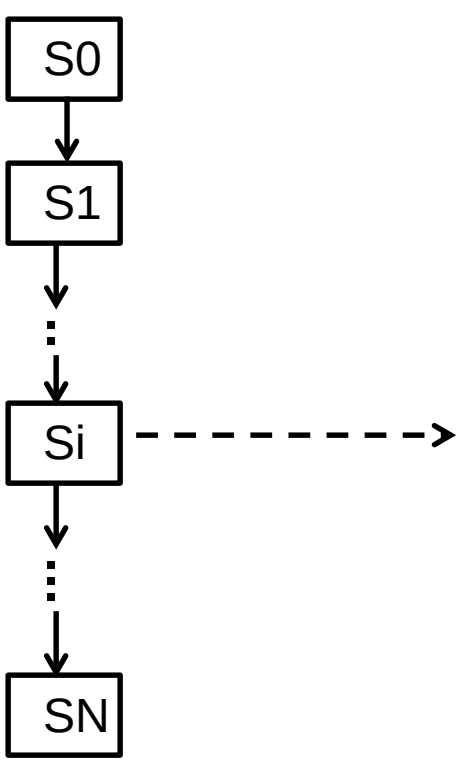
Stage Execution Behaviour

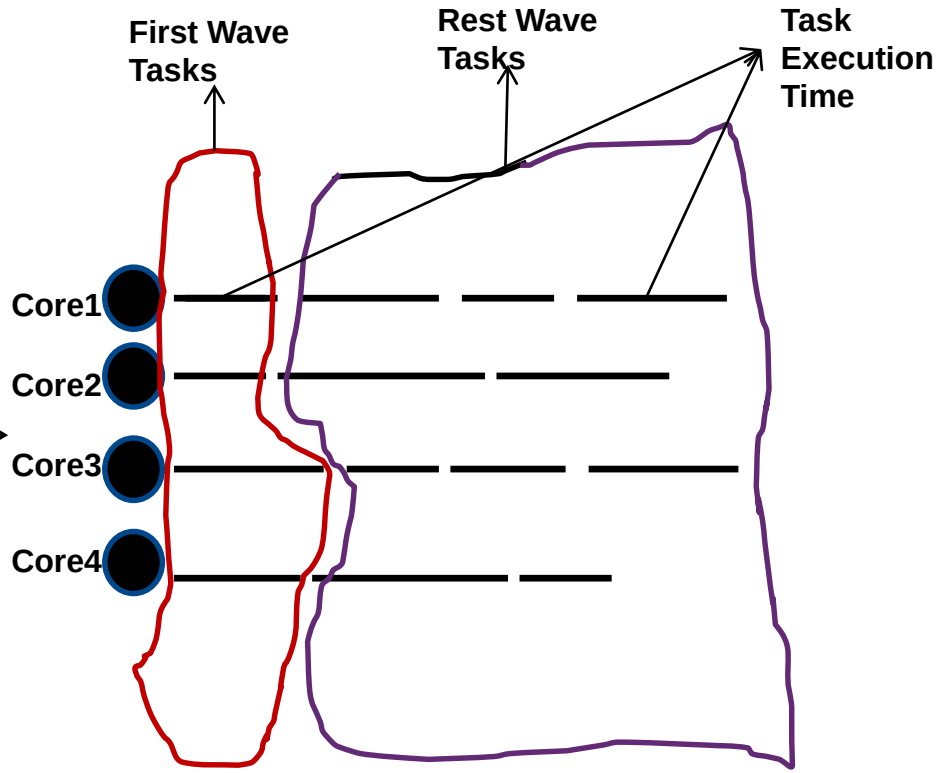


Spark Job Stages

Execution of tasks in an executor in stage Si

Stage Execution Behaviour





Spark Job Stages

Execution of tasks in an executor in stage Si

Task Execution Time Components

Scheduler delay

Serialization & de-serialization

JVM Time

Shuffle Time

Computation Time

Task Execution Time Components

Scheduler delay - # Tasks, Task launch wave

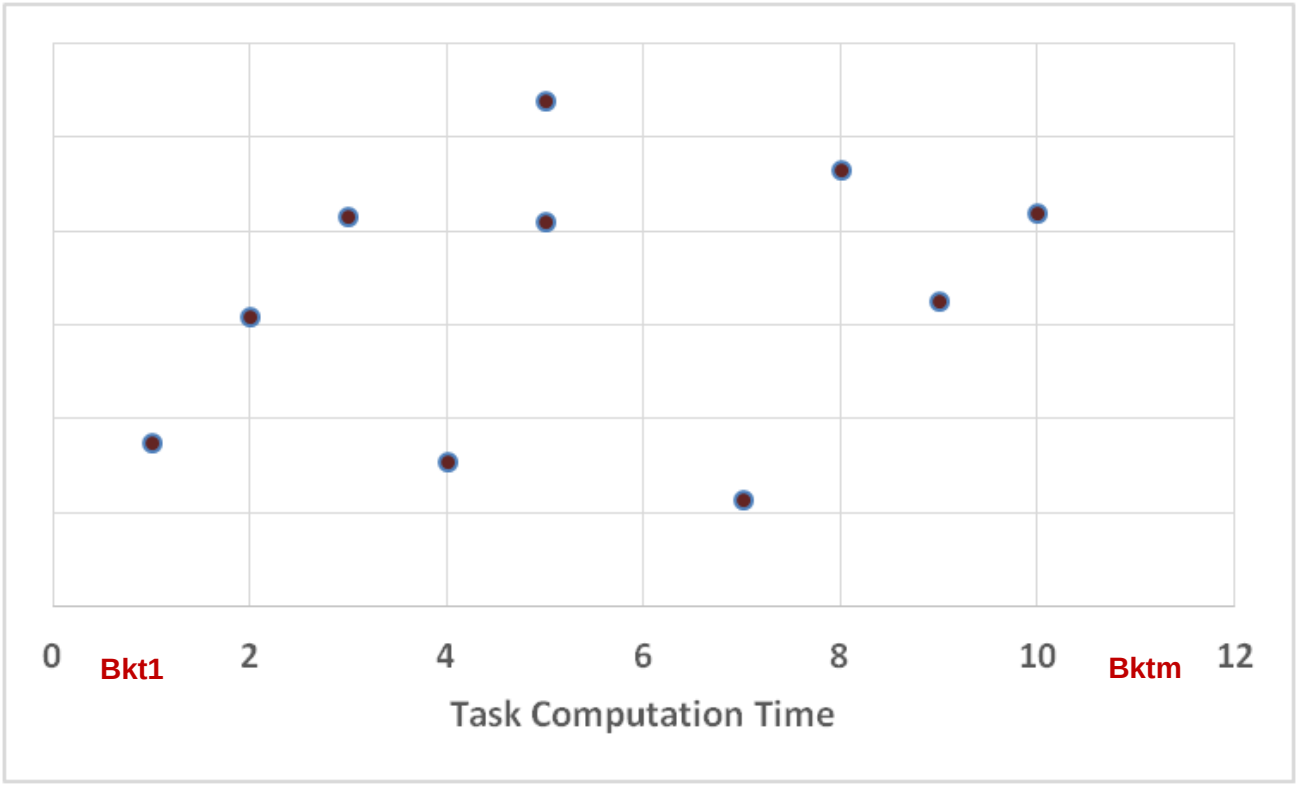
Serialization & de-serialization - Block size

JVM Time - Processing type, Cores per executor, #executors per node

Shuffle Time- Data size per executor, Executor Memory

Computation Time - computation type, block size, data skew, heterogeneity

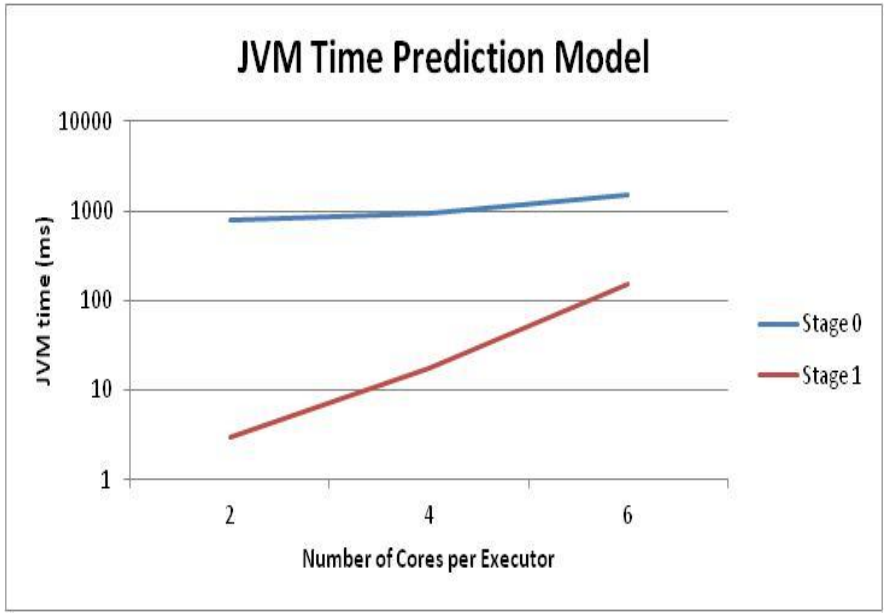
Task Computation Time Variability



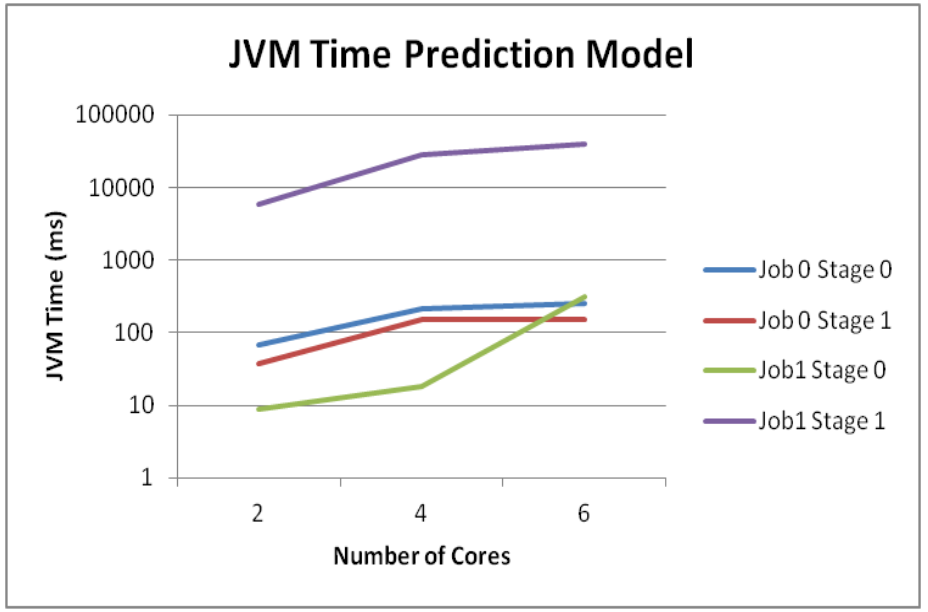
Performance Summary of Stage

- First Wave Average Scheduler Delay
- Rest Wave Average Scheduler Delay
- Number of tasks in each Bucket ‘p’
- Average computation time (duration) of each Bucket ‘p’

Task JVM Time Prediction



(a) Wordcount Application



(b) Terasort Application

Task Scheduler Delay Estimation

First Wave Scheduler delay increases **linear** to total number of tasks

Rest Wave Scheduler delay is same

Task Shuffle Time Estimation

Data Size per Task remains Same since Block Size same

$$ExecutorShuffleTime = AvgTskShuffleTime * EstimatedExecutorTasks + SpillOverheads$$

Spilloverheads estimated by generating Spurious spills in constrained Development environment

Task Execution Time Estimation

Scheduler delay - prediction model

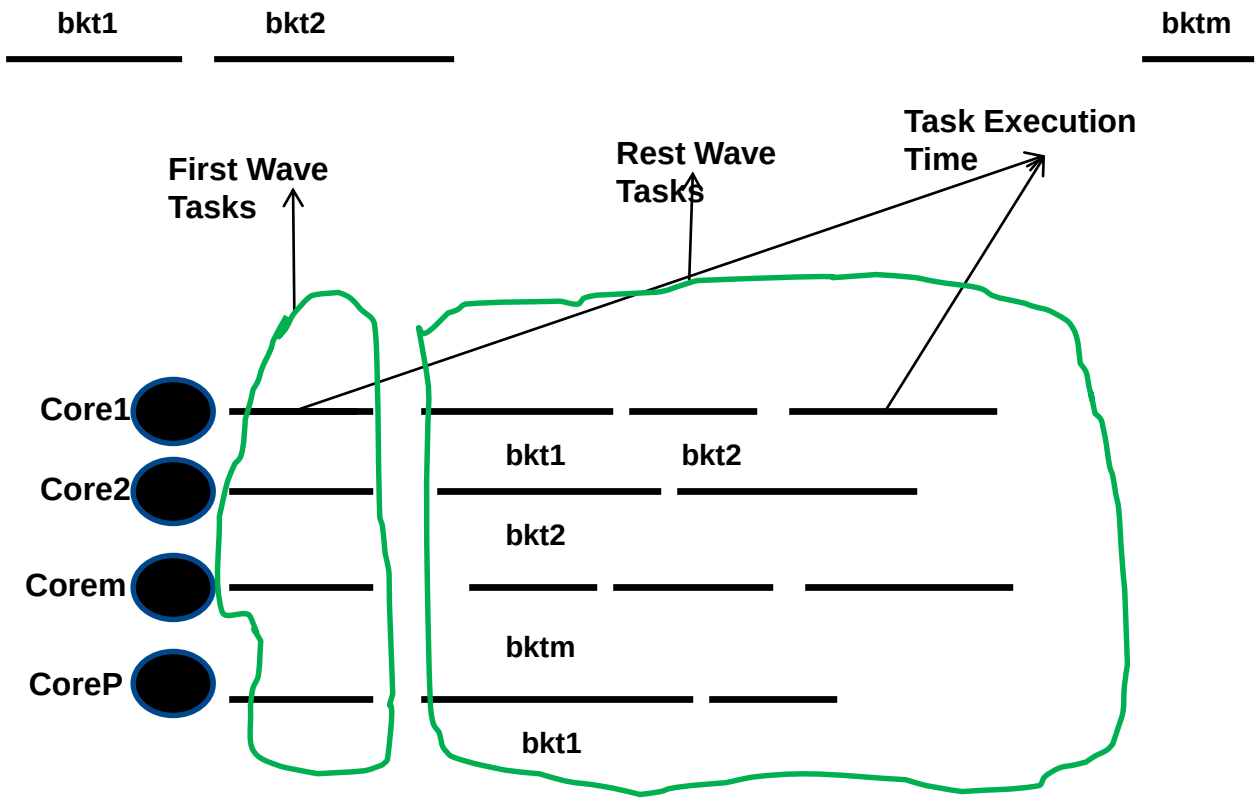
Serialization & de-serialization - from measurements

JVM Time - using prediction model based on measurements

Shuffle Time- prediction model

Computation Time - linear estimation of number of tasks in each bucket.
Each bucket duration is average of tasks' execution time in the bucket

Stage Execution Time Estimation

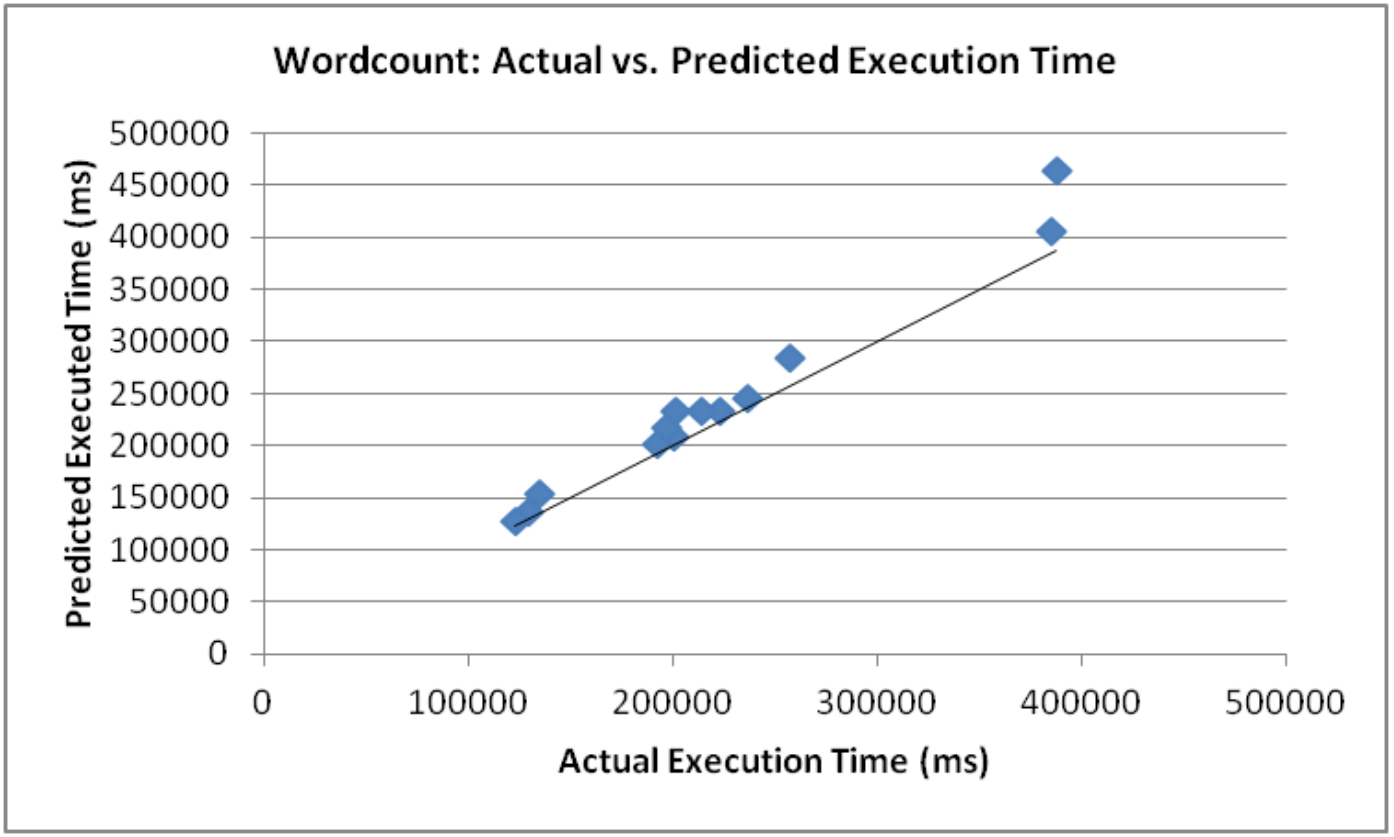


$$pStage_i^j = StageStart_i^j + Max_{onallcores} \sum_{TasksonCore} TaskExecutionTime + StageCleanup_i^j$$

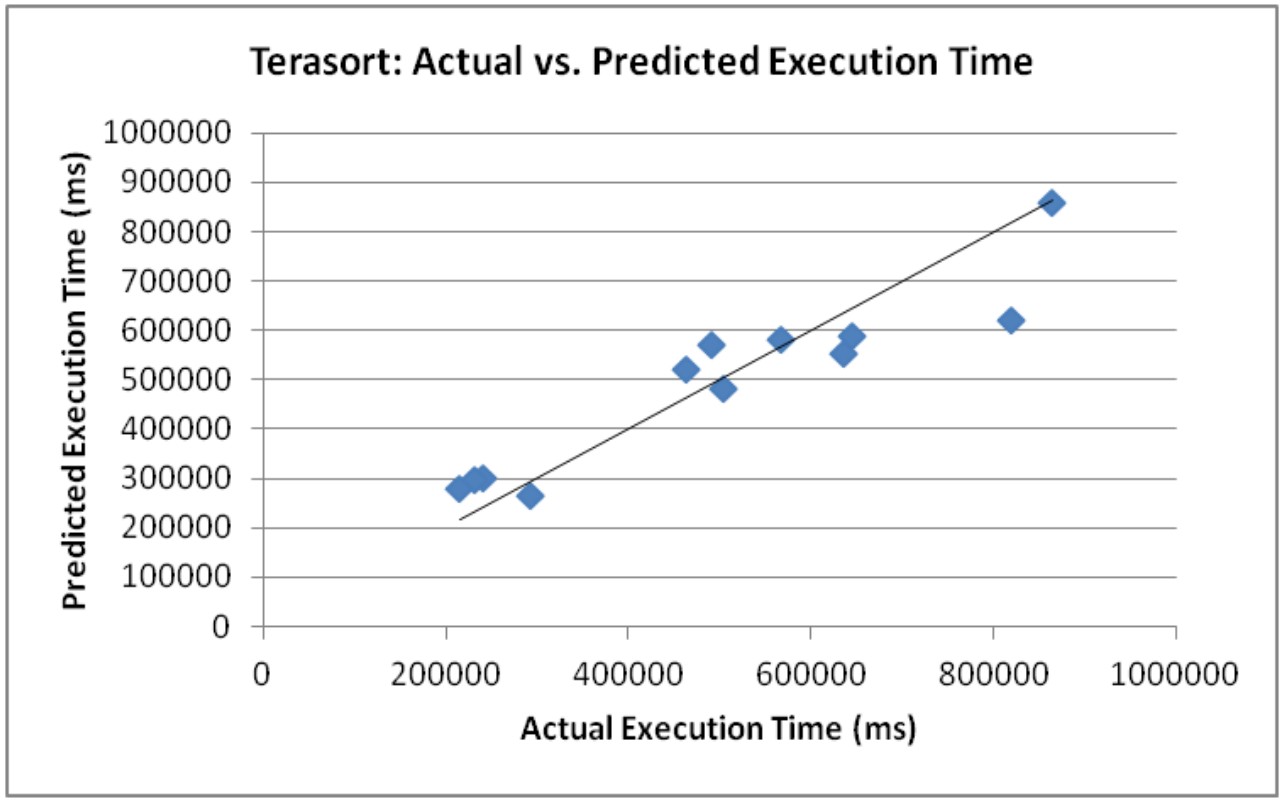
Experimental Setup for Validation

Configuration Parameter	Values
Number of Executors	2,4,6
Number of cores per Executor	2,4,6
Executor Memory	4 GB
Data Size	10 GB, 20 GB
Cluster Size	2, 4
SQL1	Average on ‘lineitem’ column
SQL 2	Join of ‘lineitem’ and ‘order’

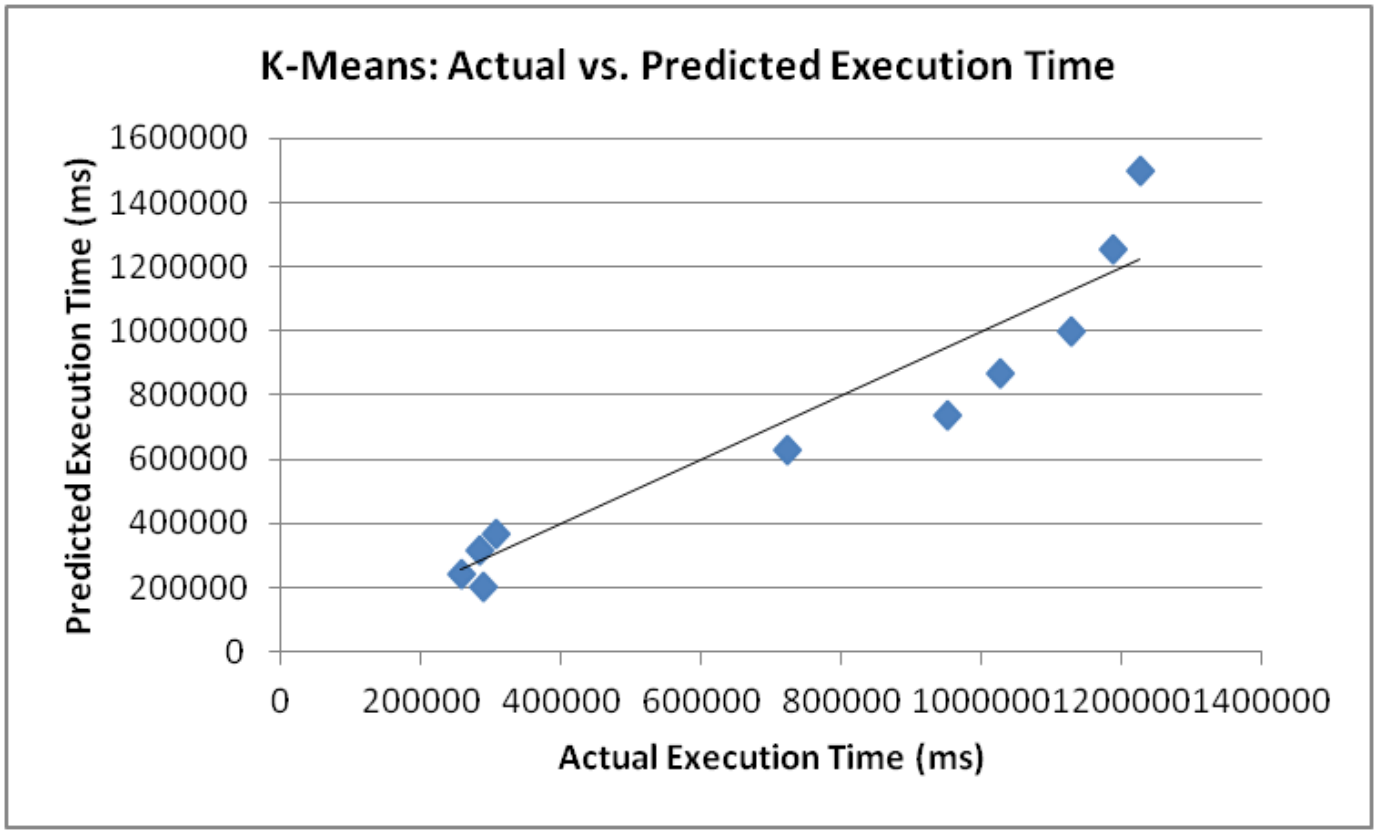
Model Validation : Wordcount



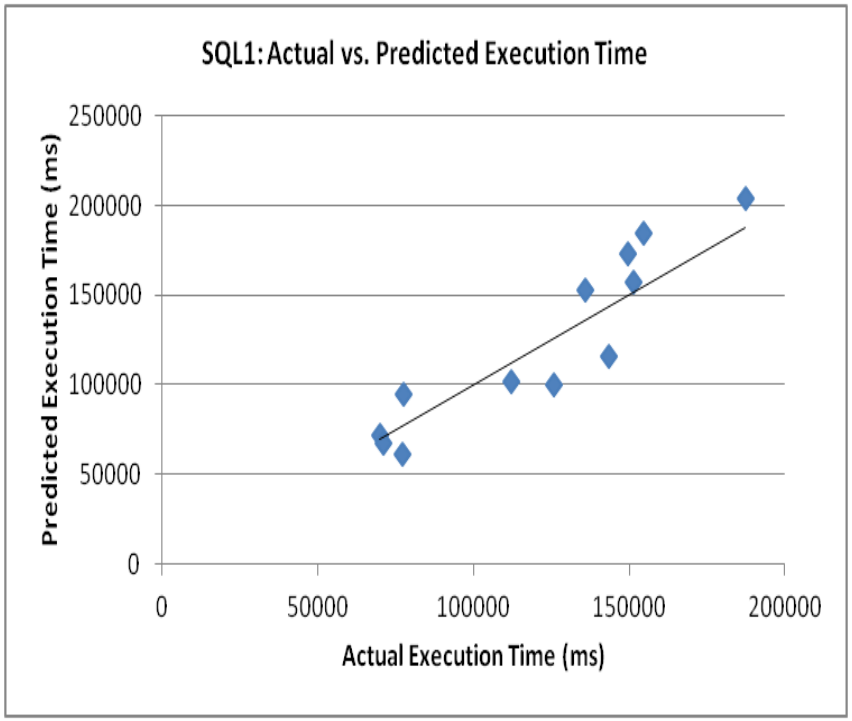
Model Validation: Terasort

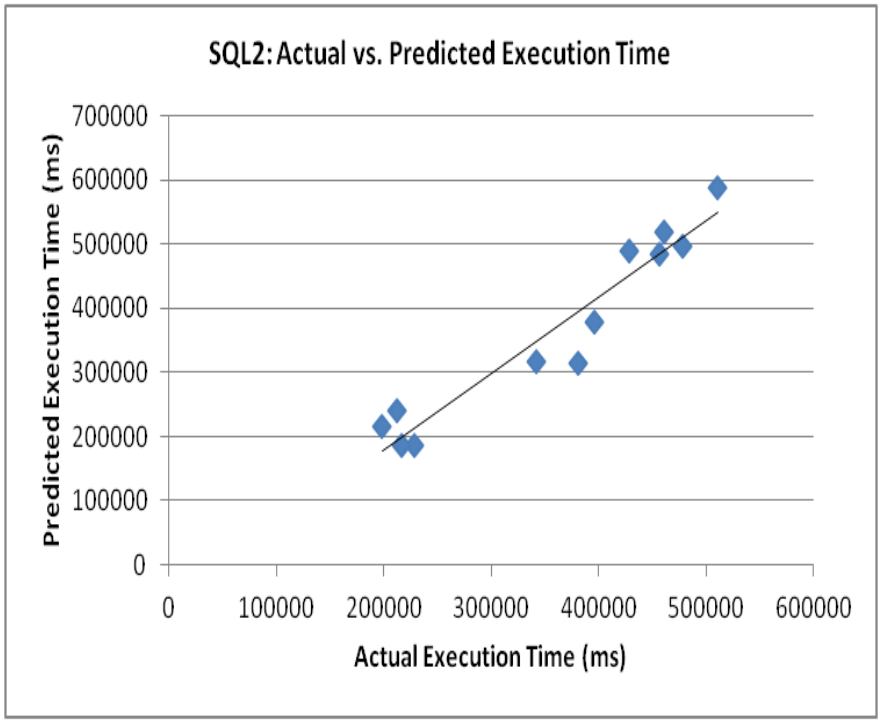


Model Validation: K-Means

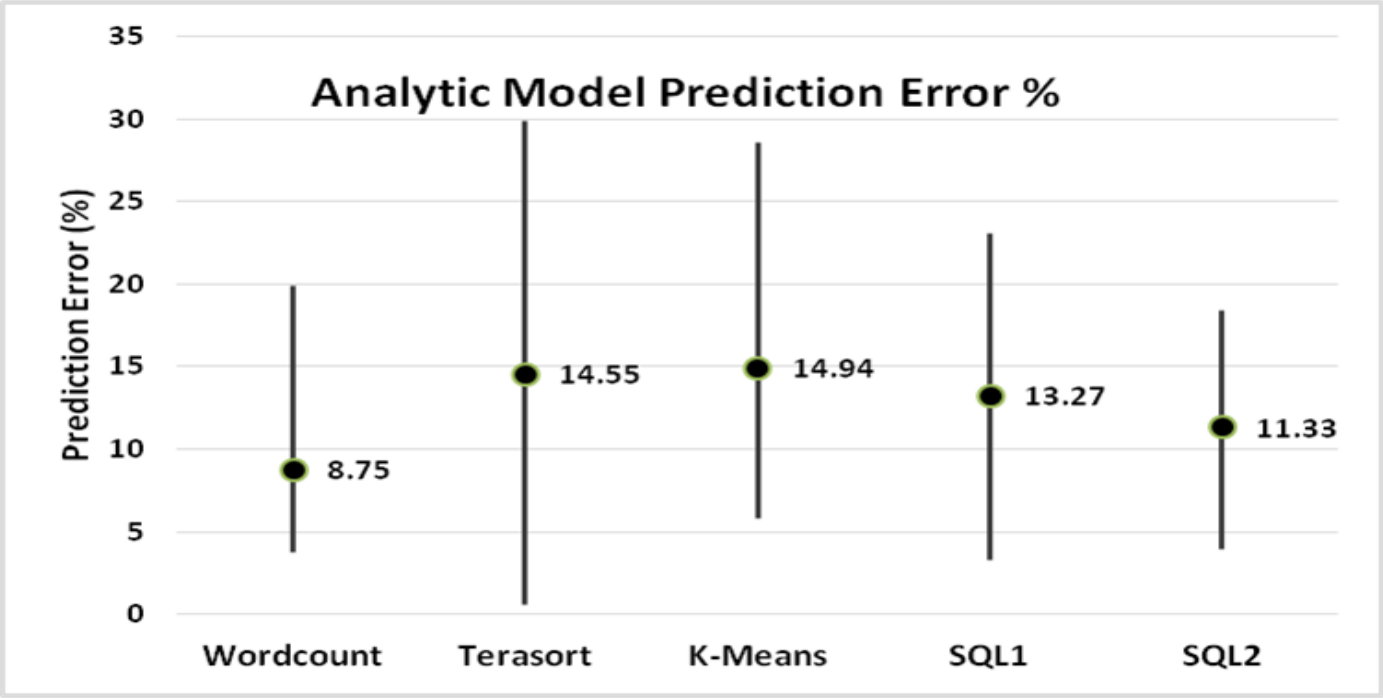


Model Validation: SQL1 & SQL 2



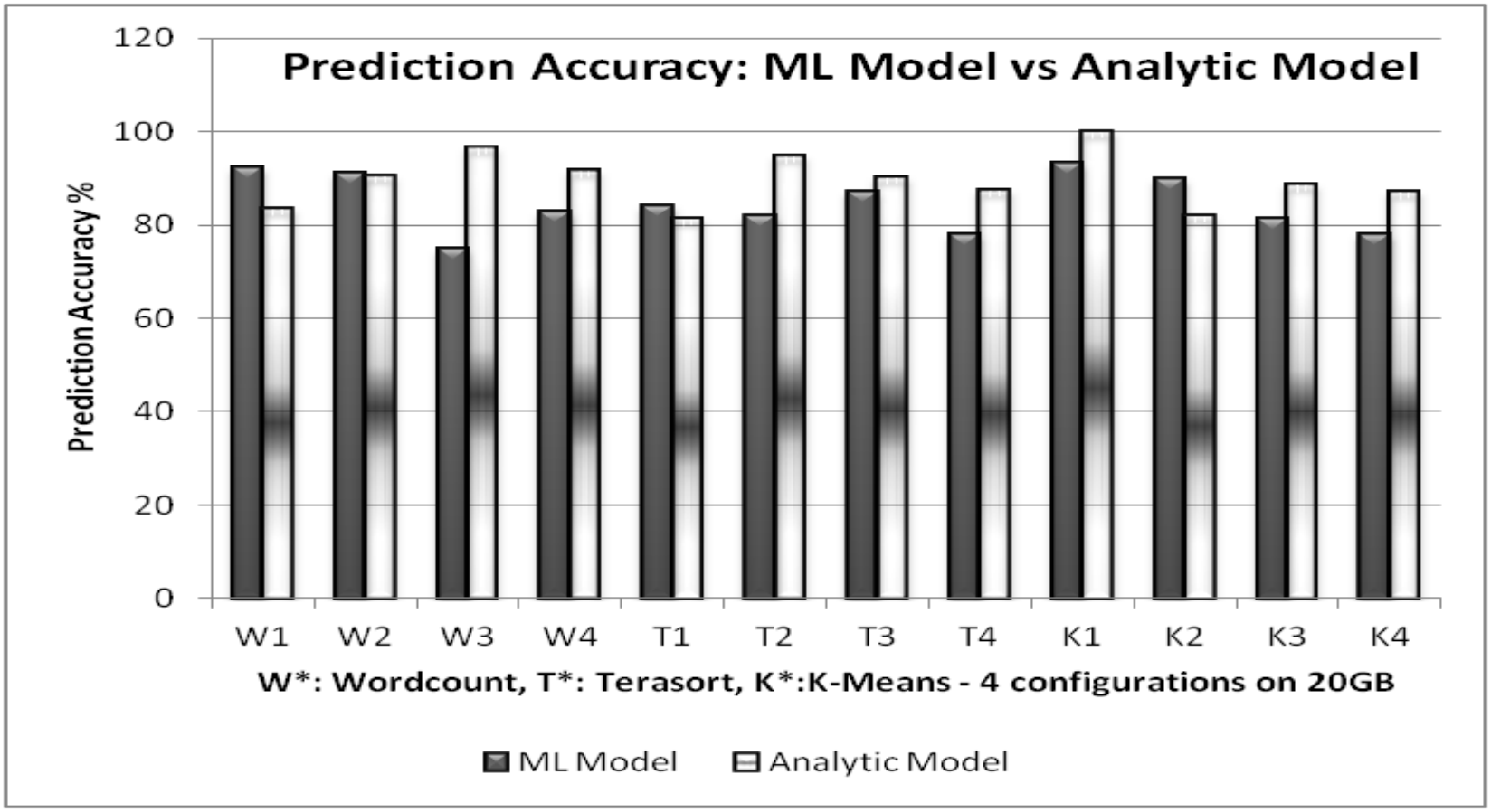


Cost Model Accuracy



Average Prediction Error < 15%

Accuracy: ML Model vs Cost Model



Auto Tuning Algorithm

```
OptimizationModule(Input: DataSize, ClusterSize)
{
Optimal_time = 9999;

For Numexecutor = 1 to max cores in the Cluster do
For Numcore_Executor = 1 to max  cores on node do
For Executormemory =  Min Size to RAM size on node do

If ValidConfiguration(cluster size, numexecutor, numcoreExecutor, Executormemory)
{
  Time = PredictTimeModel(DataSize, ClusterSize, Numexecutor, NumcoreExecutor, Executormemory)
  if  Time < optimal_time
  {
      optimal_Numexecutor = Numexecutor
      optimal_NumcoreExecutor = NumcoreExecutor
      optimal_Executormemory = Executormemory
      optimal_time = Time;
 }
}
Done
Done
Done
Return (optimal_Numexecutor, optimal_NumcoreExecutor, optimal_Executormemory)
}
```

rekha.singhal@tcs.com