Auto-tuning Spark Big Data Workloads on POWER8:Prediction-Based Dynamic SMT Threading

Research Question

- ◆ Different big data workloads have disparate architectural characristics
- ◆ Identify the most efficient SMT configuration to achieve the best performance
- ◆ Focus on auto-tuning SMT configuration for Spark-based big data workloads on POWER8
- ◆ Methodology could be gengerlized to Hadoop, Flink

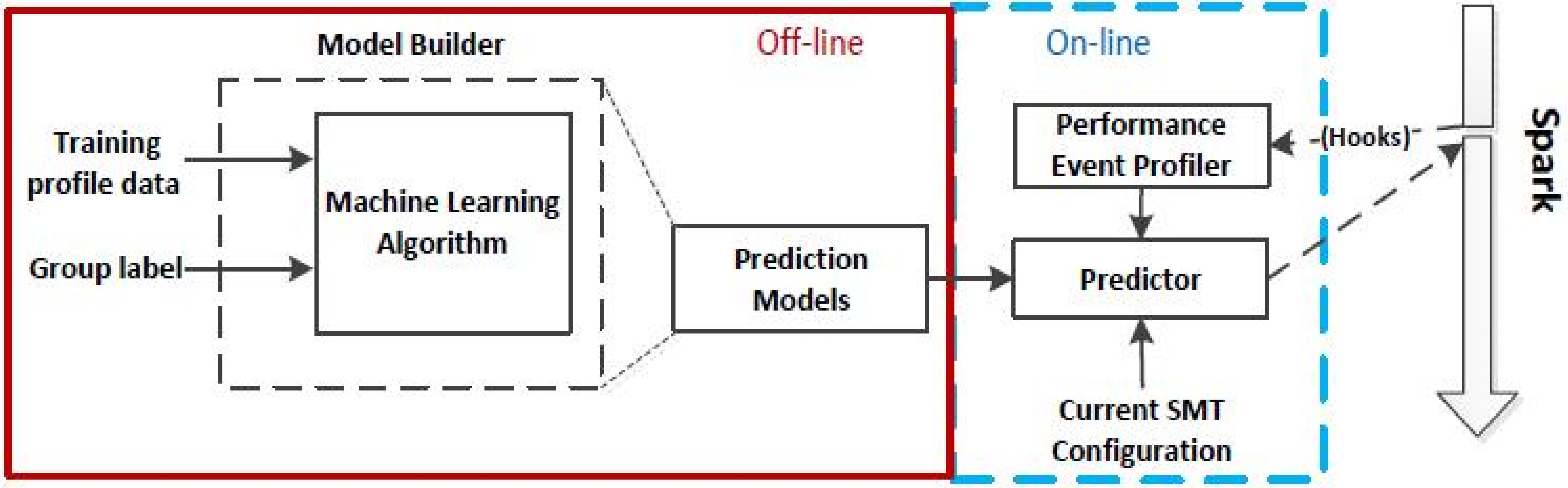
Summary of Methodology

- Propose a prediction-based dynamic SMT threading(PBDST) framework to adjust the thread count in SMT cores on POWER8 processor by eight machine learning algorithm

Methodology-Definition

- SMT: simultaneous multithreading has potential to increase the processor resource utilization
- POWER8: a processor architecture, support up to 8-way multithreading
- Profile: hardware performance counters
- Configuration: number of threads

Methodology-PBDS framework



Methodology- dataset

- 29 Spark application
- 20 application used for traing set
- 9 application used for testing set

	No.	Name	Source
Training Set	1	Spark-als	Spark-perf
	2	Spark-kmeans	Spark-perf
	3	Spark-spearman	Spark-perf
	4	Spark-chi-sq-feature	Spark-perf
	5	Spark-NaiveBayes	BigDataBench
	6	Spark-terasort	BigDataBench
	7	Spark-grep	BigDataBench
	8	Spark-sort	BigDataBench
	9	Spark-pykmeans	Spark-perf
	10	Spark-pyals	Spark-perf
	11	Spark-pearson	Spark-perf
	12	Spark-pybayes	Spark-perf
	13	Spark-pycout-w-fltr	Spark-perf
	14	Spark-summary-statistics	Spark-perf
	15	Spark-pyspearman	Spark-perf
	16	Spark-count	Spark-perf
	17	Spark-chi-sq-mat:	Spark-perf
	18	Spark-gmm	Spark-perf
	19	Spark-svd	Spark-perf
	20	Spark-WordCount	BigDataBench
Testing Set	21	Spark-ConnectedComponent	BigDataBench
	22	Spark-block-matrix-mult	Spark-perf
	23	Spark-word2vec	Spark-perf
	24	Spark-pca	Spark-perf
	25	Spark-pypearson	Spark-perf
	26	Spark-glm	Spark-perf
	27	Spark-PageRank	BigDataBench
	28	Spark-int-sory-by-key	Spark-perf
	29	Spark-fp-growth	Spark-perf

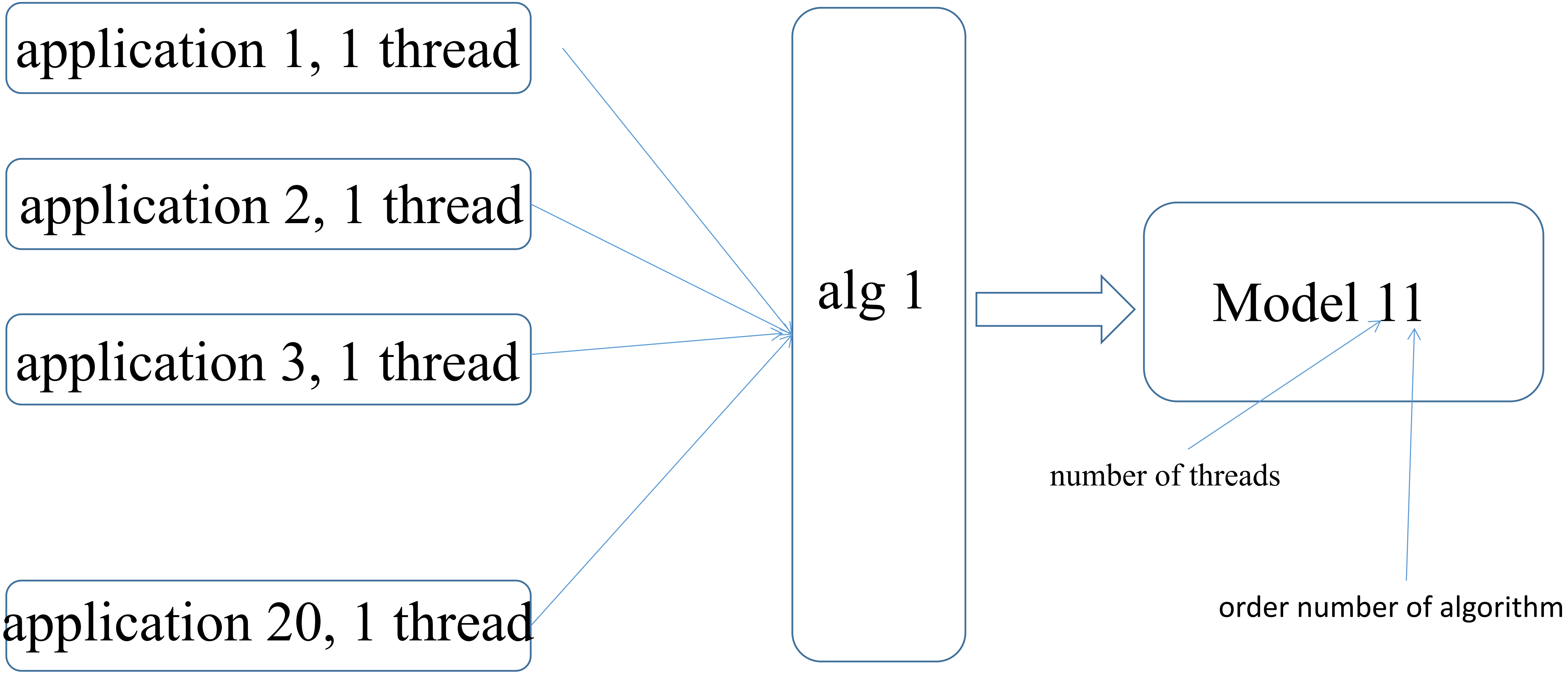
Methodology- offline training

- run training set with different SMT configurations, from one thread per core to eight threads per core

- for each workload, collect the microarchitecture level statistics during the whole lifetime

- one model for each SMT configuration

Methodology- offline training



Methodology- Input & Output data

application 1, 1 thread

Input data: *traing profile data*, this is a matrix and each row is the micro-architecture level statistics of applications in thread 1

Output data: *group label*, this is an array which records the best SMT configurations for application 1,application 2,...application 20

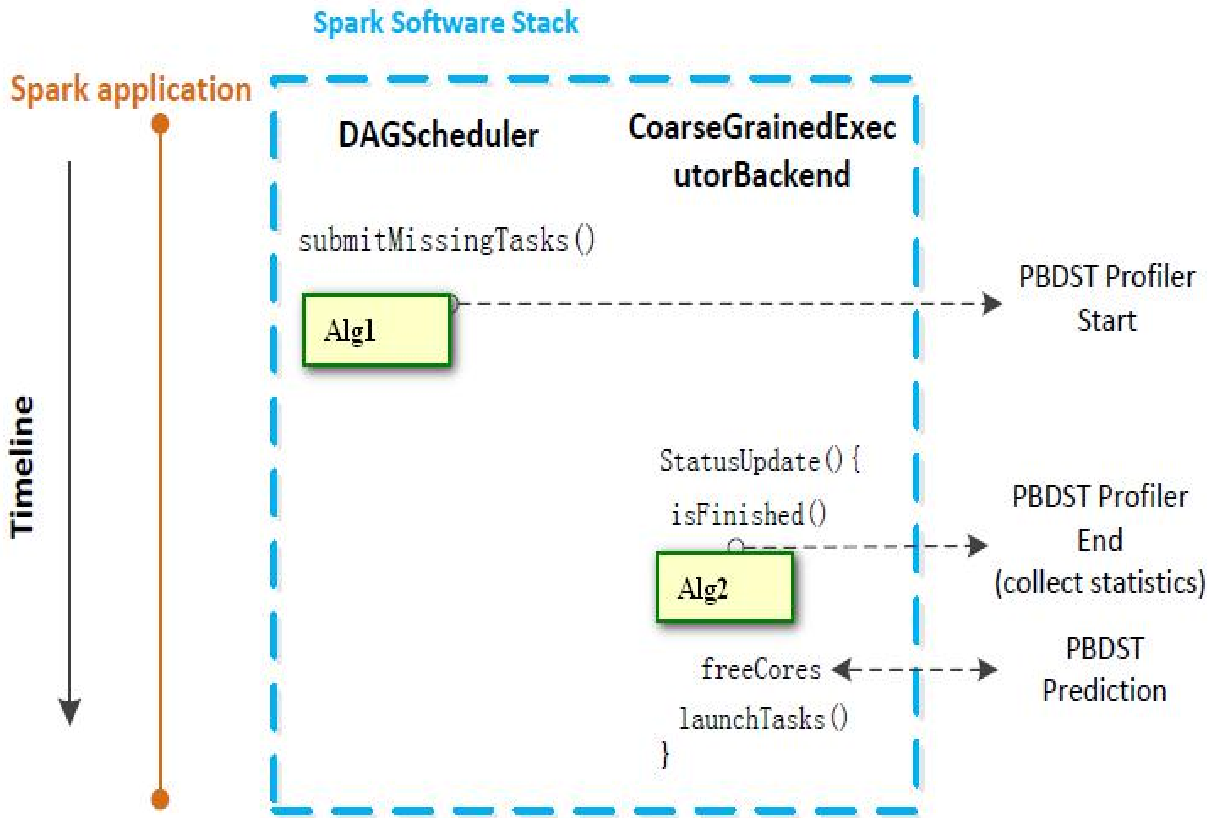
In 9 testing application, for every application, use the first round of tasks' micro-architecture level profiles to predict the optiaml SMT configuration for the upcoming tasks in the same stage

fist round task's
(sampling)

model according to the current SMT configuration

prediction SMT configuration for upcoming tasks

Auto-tuning



Algorithm 1 Algorithm used to trigger the PBDST hooks in *DAGScheduler*

```
Input: Variables:
    Current SMT configurations: CurSMT;
    The number of cores: cores;
    The TaskSet information: taskSet;
if submitTaskSet() then
    if noStageIsRuningWithProfiler() then
        if taskSet.size > CurSMT * cores then
            PBDST.profiler.start();
        end if
    end if
end if
```

Algorithm 2 Algorithm used to trigger the PBDST hooks in *CoarseGrainedSchedulerBackend*

```
Input: Variables:
    Current SMT configurations: CurSMT;
    Task state: state;
    Current Stage Id: curStageId;
    Last Stage Id: lastStageId;
if isFinished(state) then
    if lastStageId != curStageId then
        lastStageId = curStageId
        if profilerIsRuning() then
            PBDST.profiler.stop();
            predictSMT = PBDST.prediction(CurSMT);
            if predictSMT != CurSMT then
                for excutor in excutorMap do
                    executor.freeCores.updating(predictSMT);
                end for
            end if
        end if
    end if
end if
```

Prediction Schema

First Schema: performs multiple predictions, each stage uses its locally optimal SMT configuration

---frequently threading-switching, rebalanceing the thread distribution, prediction. All of those will cause performance penalty

Second Schema: predicts the best SMT configuration in the first stage and keep it unchane in all stages

Results

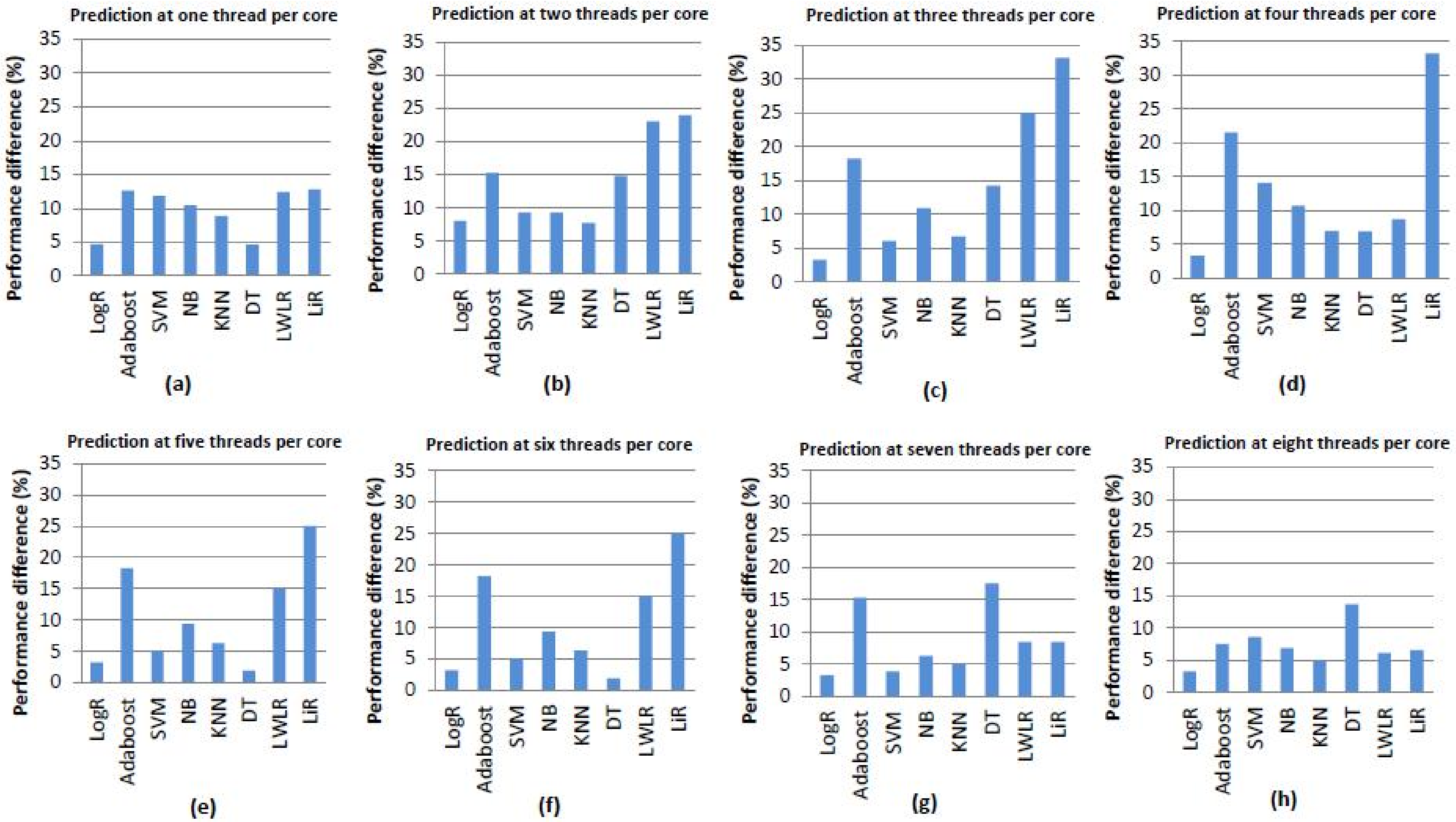


Figure 5: The average static prediction errors of different predictors with diverse SMT configurations.

Result

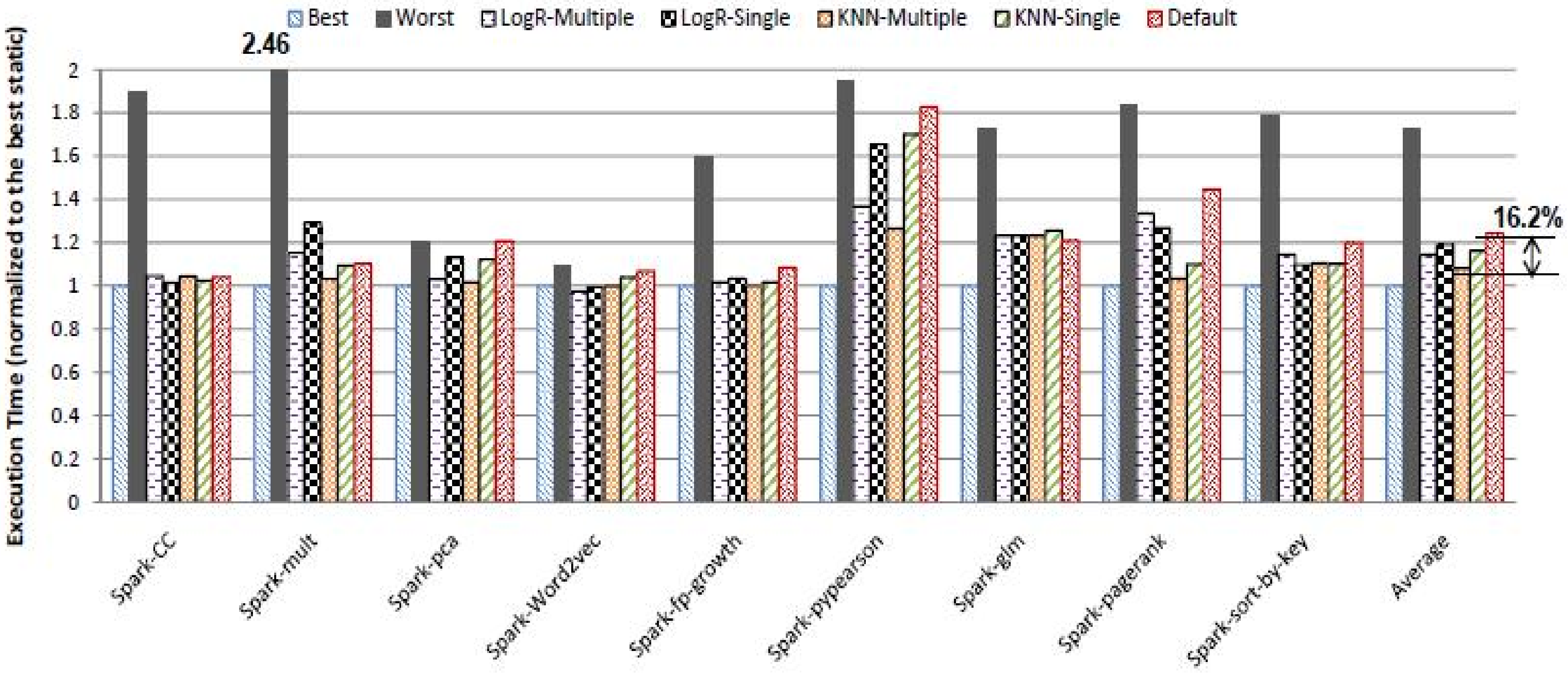


Figure 6: Effects of using different machine learning algorithms when perform automatic predictions.

PBDST framework can achieve avarage perfomance improvement rangeing from 5.2% to 16.2% for different predictions with different prediction schemes.

Result