MR – Advisor: A Comprehensive Tuning Tool for Advising HPC Users to Accelerate MapReduce Applications on Supercomputers

Md. Wasi-ur-Rahman, Nusrat Sharmin Islam, Xiaoyi Lu, Dipti Shankar and Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University

Introduction

MR - Advisor, a comprehensive tuning tool for MapReduce, is generalized to provide performance optimizations for Hadoop, Spark, and RDMA-enhanced Hadoop MapReduce designs over different file systems such as HDFS, Lustre, and Tachyon

MR - Advisor recommends the user with the best possible configuration settings for a near-optimal performance of user-defined applications and benchmarks.

This paper also generalized configuration param- eter space that makes this design capable of running with different file systems as well

Background

RDMA and Lustre-based advanced designs: Leverage RDMA(high performance interconnect protocol) and Lustre (parallel file system) to alleviate the major performance bottlenecks of running Hadoop on modern HPC clusters

Architecture Overview

--provide recommendations for configuration parameters' values of underlying file systems used for MapReduce, such as HDFS, Tachyon

--provide tuning facilities for the cluster resource managers, such as YARN, Mesos

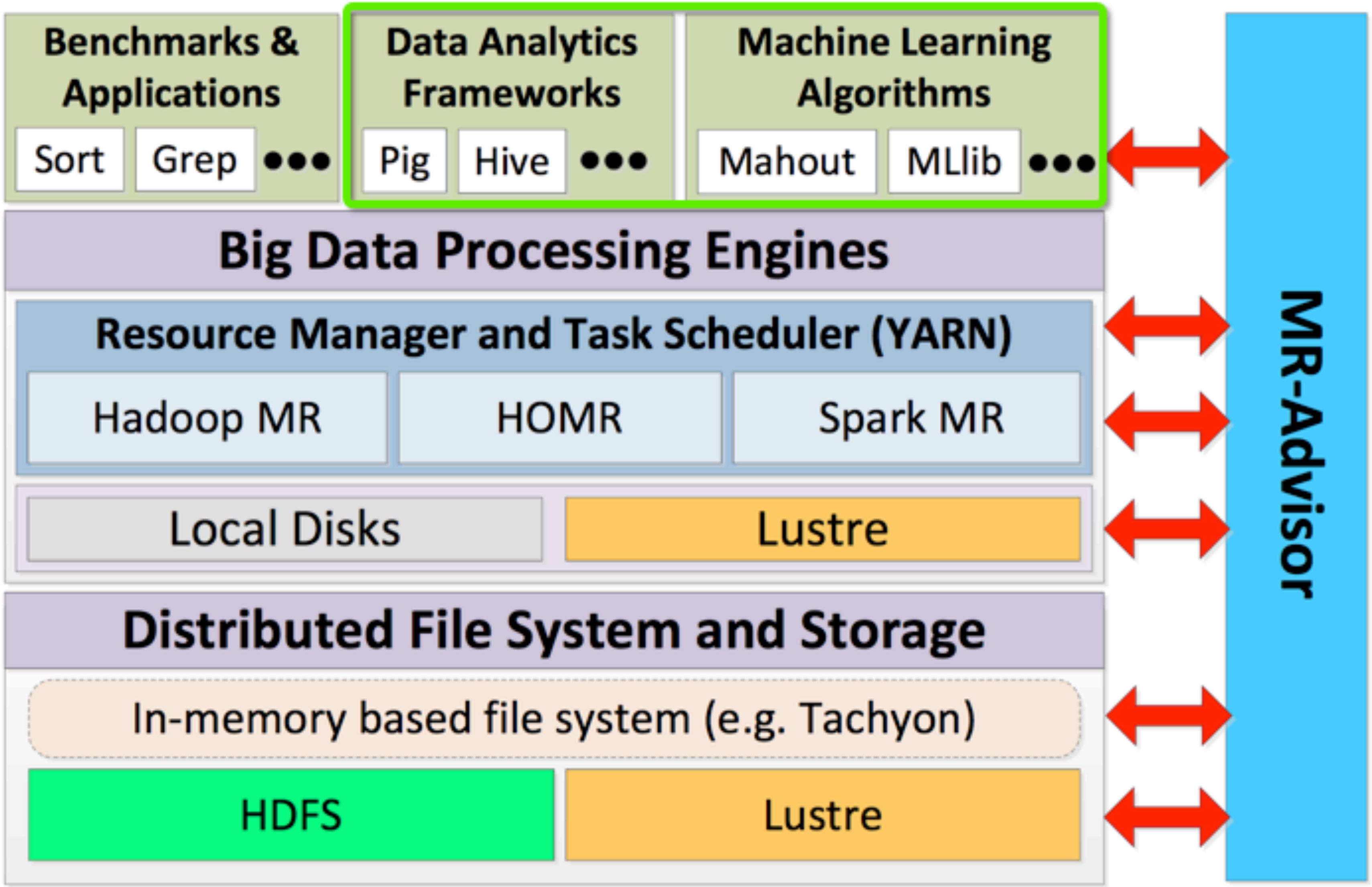
--can also be easily extended to provide tuning facilities for data analytics frameworks that use MapReduce as the processing engine, such as Hive and Pig

--MR-Advisor can also be utilized to provide recommended values for different intermediate data directories, such as Lustre or local disks

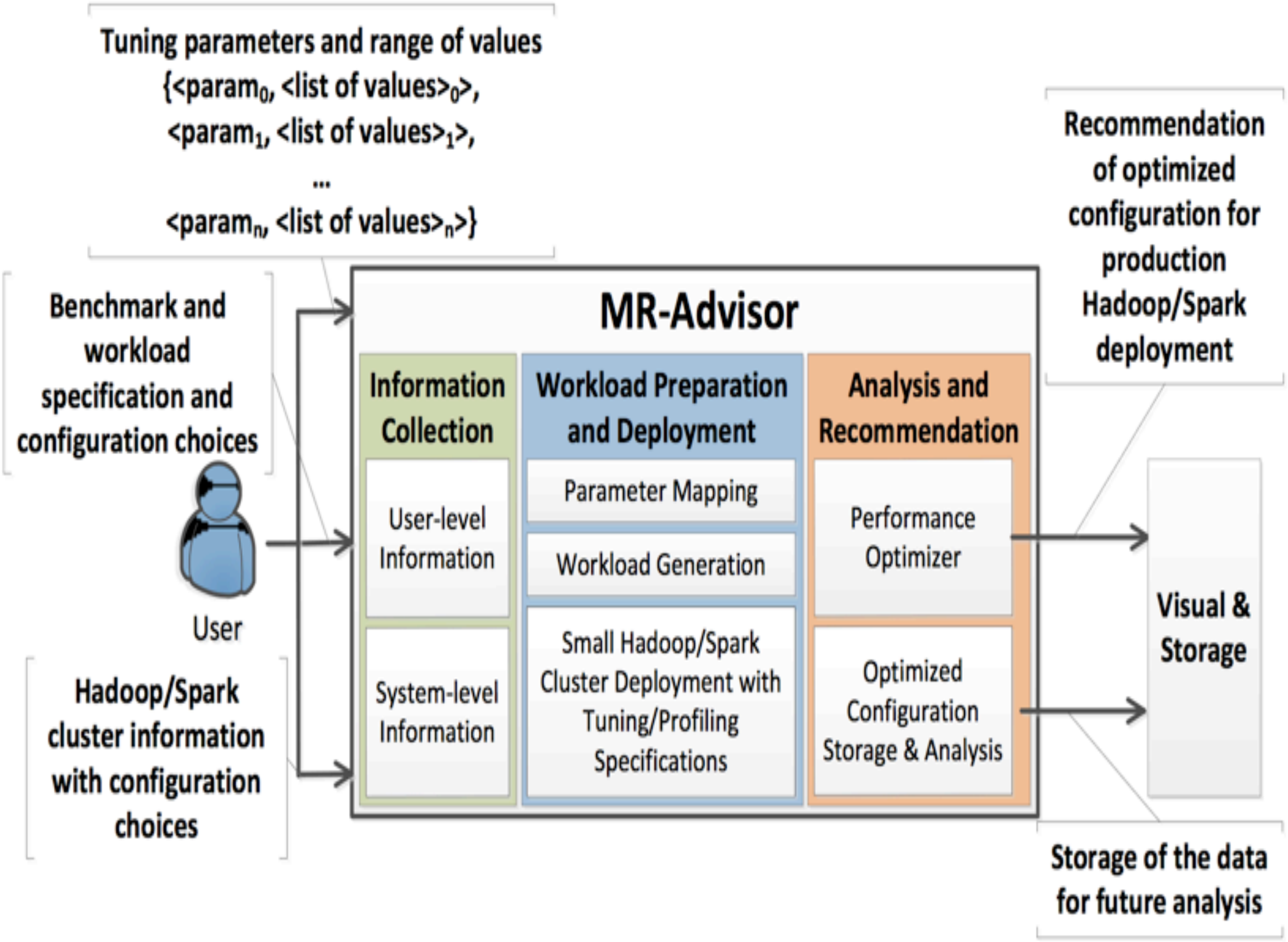
--also capable of adhering to the growing changes of hardware and software distributions

-- easily extensible to any future middleware distributions running with advanced hardware

Architecture Overview



The current version of the MR-Advisor provides tuning functionalities for each of the component shown in Figure except the data analytics and machine learning frameworks



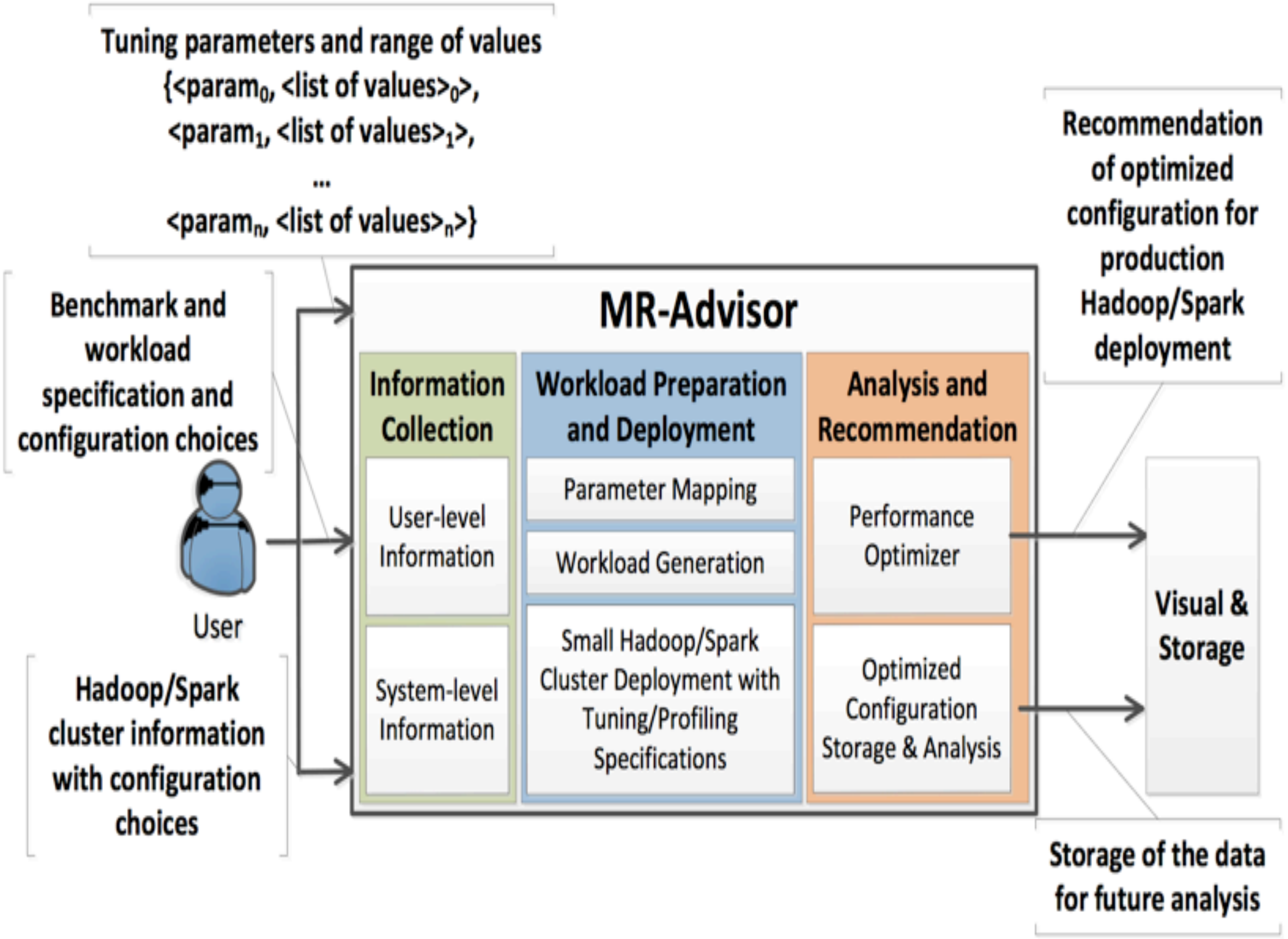
Information Collection

First input

-- the user has to provide the Hadoop or Spark cluster installation that user desires to optimize, predetermined configuration choices

--- MR-Advisor takes such choices and then maps them to the appropriate (Hadoop or Spark) parameter in the next stage.

Information Collection



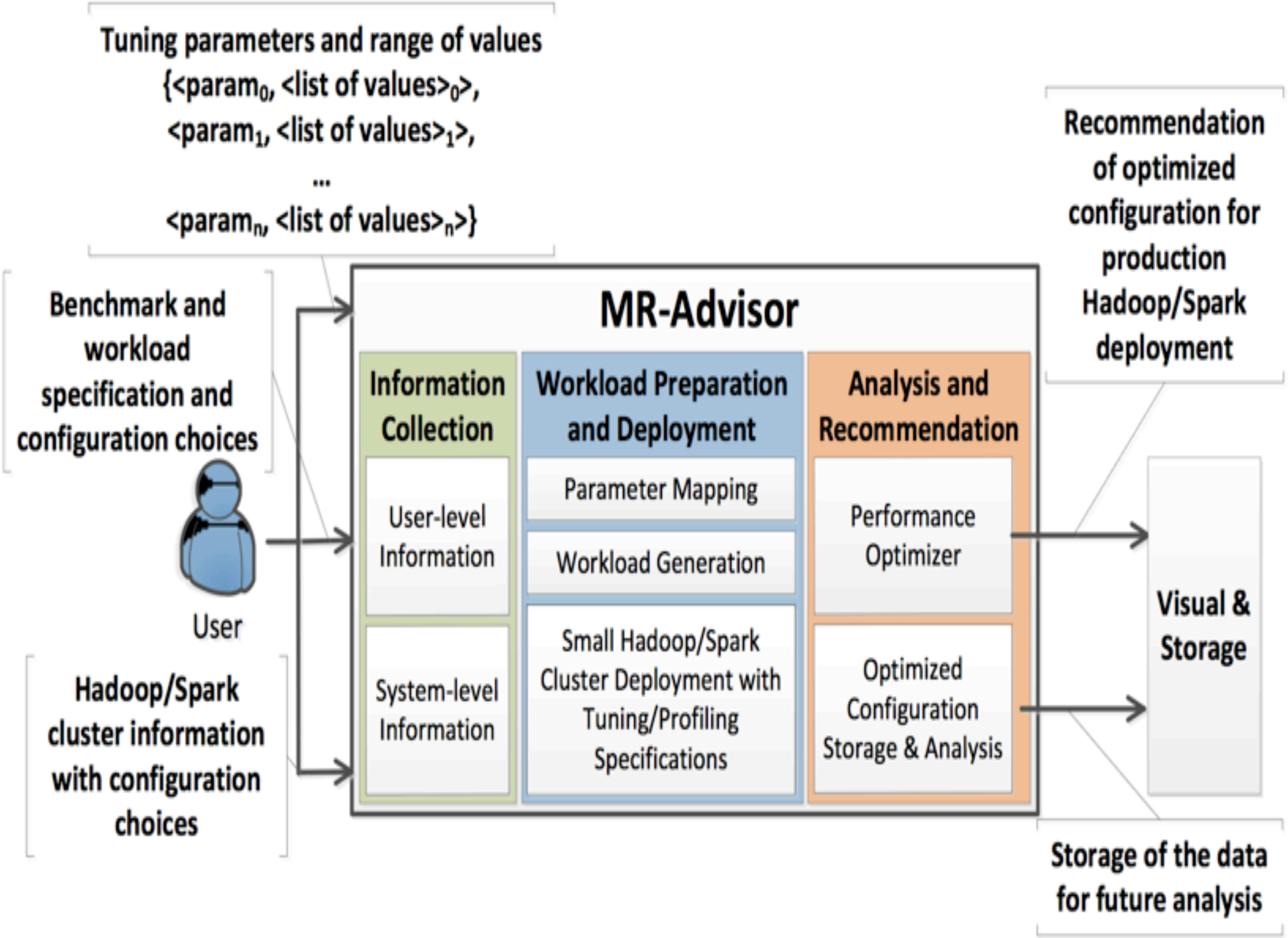
Information Collection

Second input

-- the user must provide the benchmarks , workload specifications and mention the workload size

Third input

-- user provides a set of parameters with an associated range of values in < param, < values > tuples, which MR-Advisor will consider for tuning experiments.

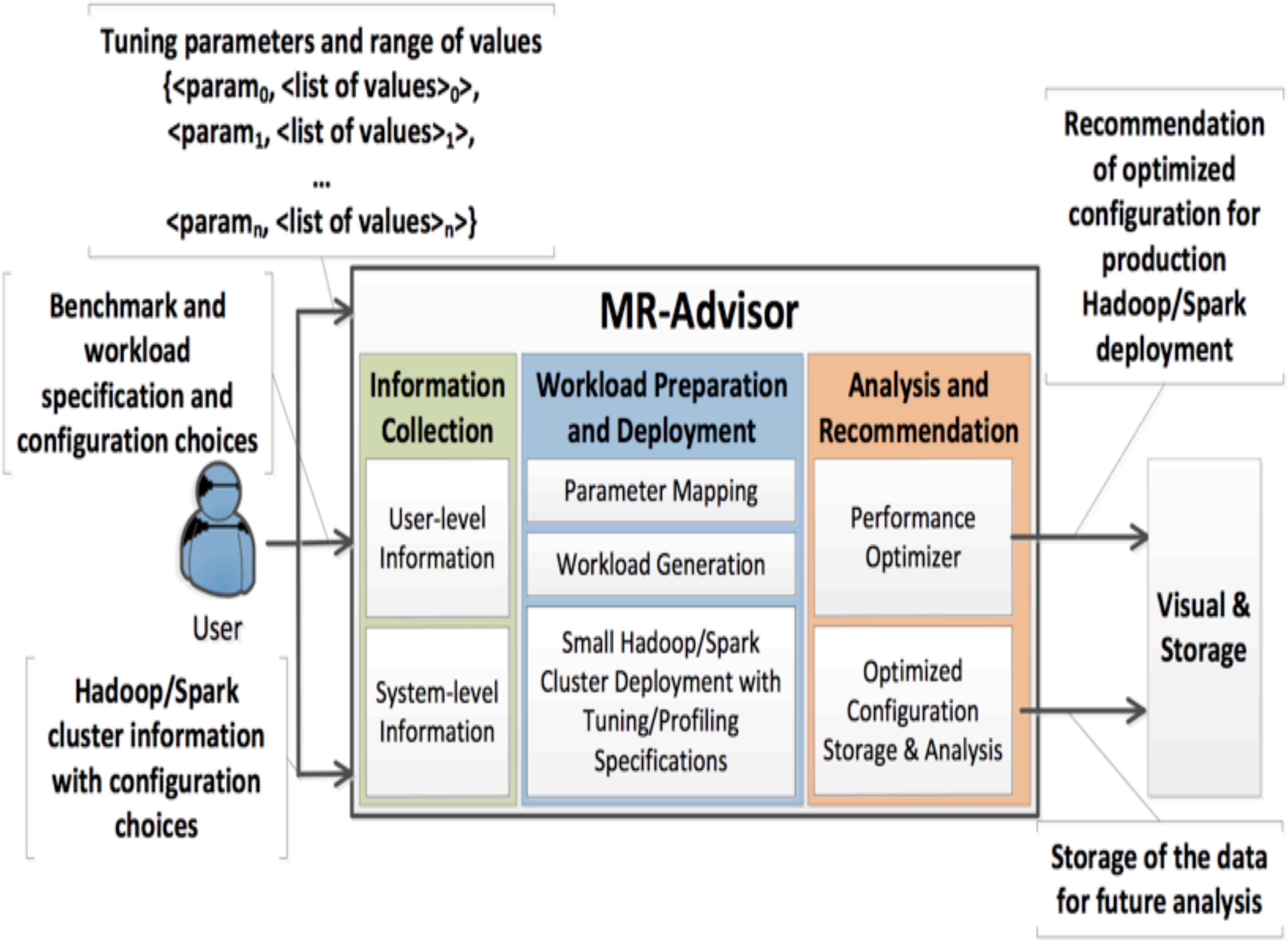


Information Collection

System-level information

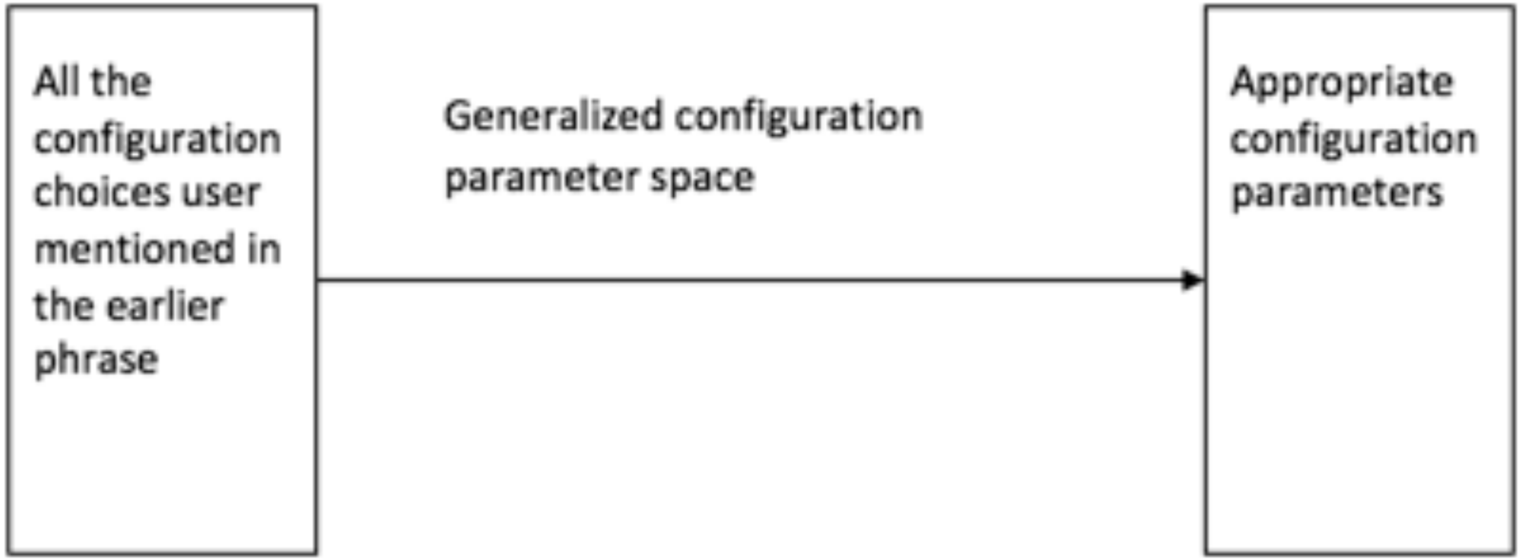
--cluster resource managers (e.g. slurm, PBS) that will be utilized to deploy small Hadoop and Spark clusters for tuning experiments

--Desired cluster size and other job submission related data (e.g. how long each experiment will run)



Workload Preparation and Deployment

Three subphases
--Parameter Mapping
--Workload generation
--Deployment and monitoring



--Eg: input data can be configurations in any MapReduce framework regardless of its implementation specifics

--configuration space: user space parameters, system space parameters

User space parameters

First dimension: Benchmarks and Applications:
---parameters related to benchmarks , applications, or any other frameworks that use MapReduce as its processing engine
---These parameters define workload characteristics and they must be provided during the runtime of the job execution.
--- These parameters usually have the highest impact on the execution performance

Modes of Operations
--- MapReduce can be configured to run in different modes based on the underlying file system and the usage of resource managers
--- each mode may require a different set of configuration parameters to be set/reset
--- MR-Advisor provides pre- built configuration groups that define different modes of operation. the user only needs to choose a particular mode of operation without learning the configuration and deployment complexities associated with it.

System space parameters

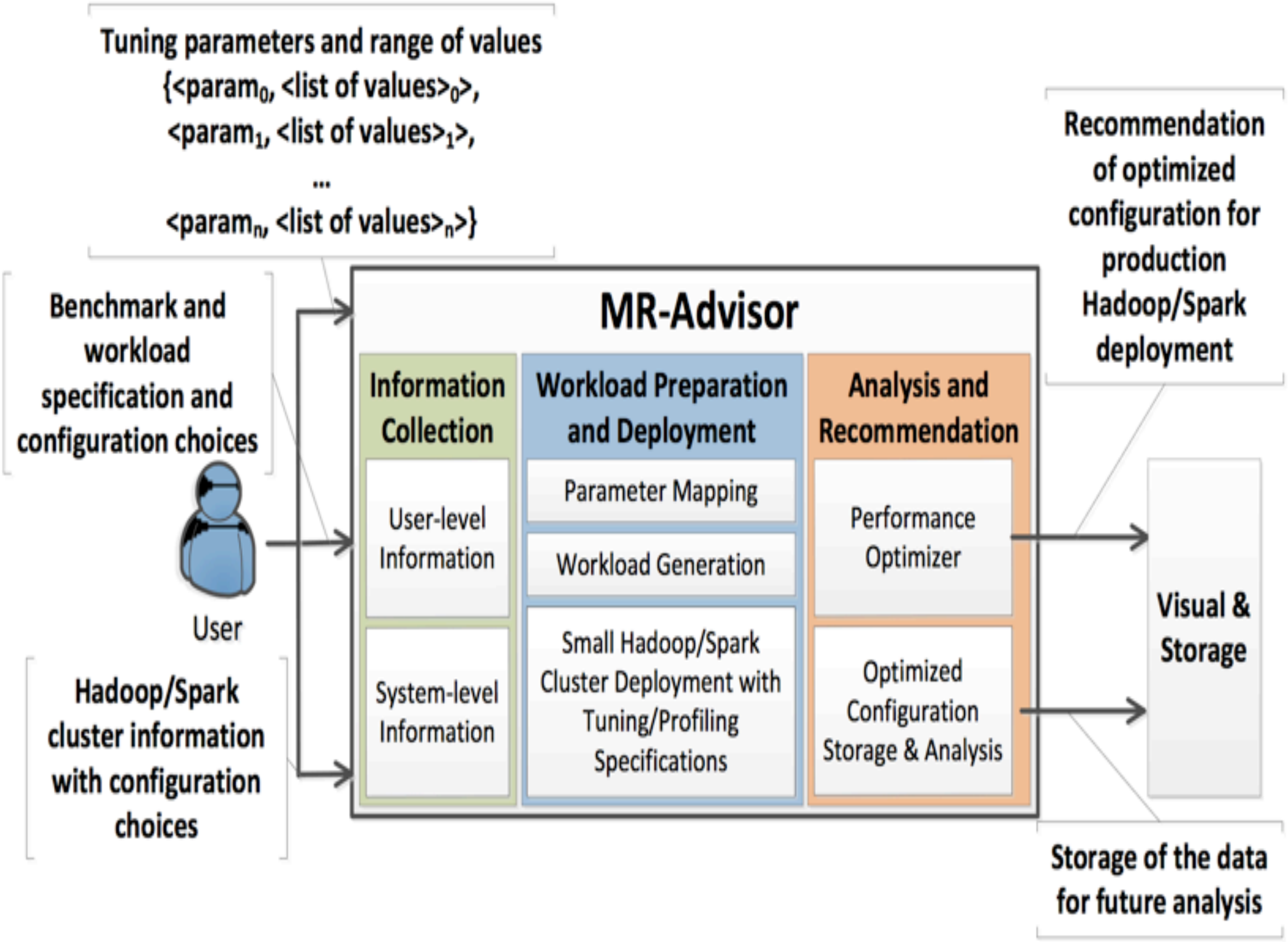
Table I

DIMENSIONS OF SYSTEM SPACE PARAMETERS IN MR-ADVISOR

Dimension	Example Parameter
Interconnects and Protocols	Hadoop: mapred.tasktracker.dns.nameserver
	Spark: SPARK_LOCAL_IP
Memory	Hadoop: mapreduce.task.io.sort.mb
	Spark: SPARK_EXECUTOR_MEMORY
CPU	Hadoop: mapreduce.map.cpu.vcores
	Spark: SPARK_EXECUTOR_CORES
File System and Storage	Hadoop: mapreduce.cluster.local.dir
	Spark: SPARK_WORKER_DIR

Based on all these dimensions of parameters, any Map- Reduce implementation can be tuned for a particular HPC cluster environment.

Workload Generation

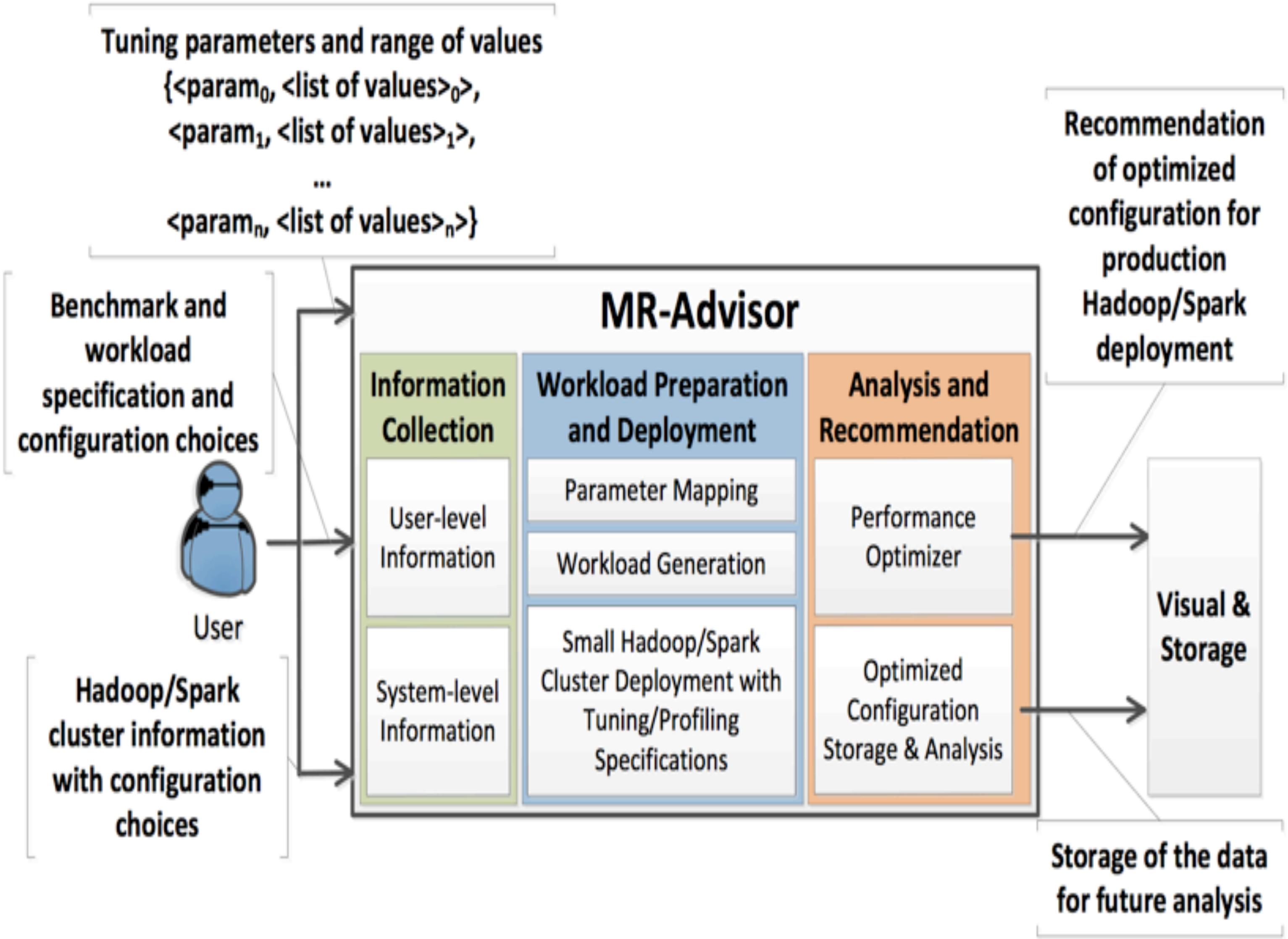


Workload Preparation and Deployment

Workload Generation

--It generates the workloads based on the number of tuning parameters and the range of values for each of these parameters

--If multiple parameters are tuned in a combined execution, MR-Advisor generates all possible combinations of configuration files, each of them containing one instance of each parameter with one of the values in the specified value range.



Workload Preparation and Deployment

Deployment and Monitoring

---all of the configuration files are prepared, the job submission and execution phase stars.

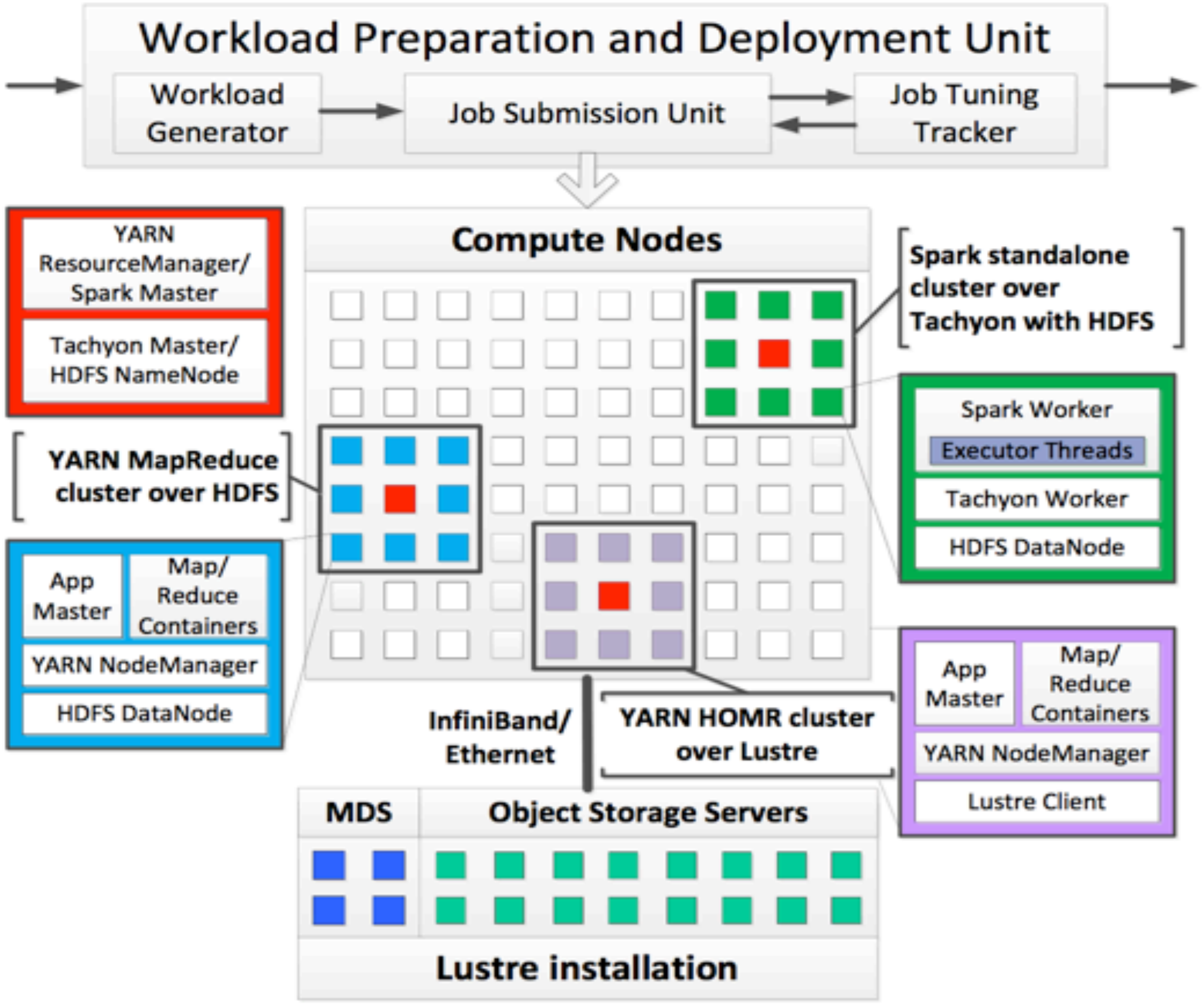


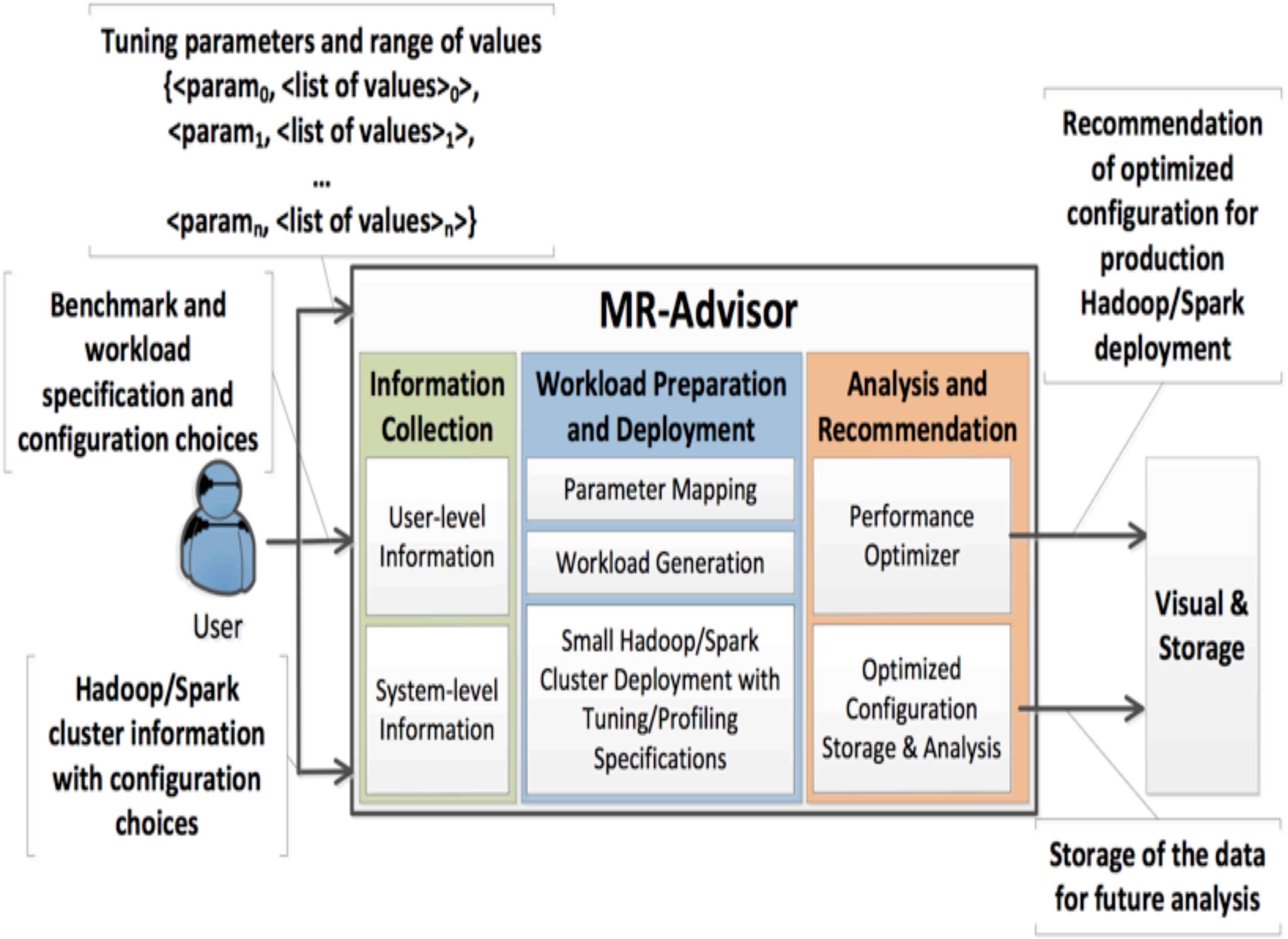
Figure 2. Job submission and monitoring of WPDU

-- each job runs on a small set of nodes where the Hadoop/Spark cluster is deployed

-- The job submission unit utilizes the system resource managers to deploy these small clusters and then submits the tuning experiments

-- keep the cluster size of the tuning experiments as an input parameter from the user, also provide prediction method

Analysis and Recommendation



Workload Preparation and Deployment

Analysis and Recommendation

---After all the submitted jobs complete their execution, MR-Advisor starts collecting results for performance metrics.

---MR-Advisor utilizes a repetitive divide and conquer algorithm to find the local optimal, then it tries to find another from the rest of the performance results.

---Currently, recommendation is performed with respect to one metric (execution time) only

Experimental setups

---TACC Stampede(cluster A)

---SDSC Gordon (Cluster B)

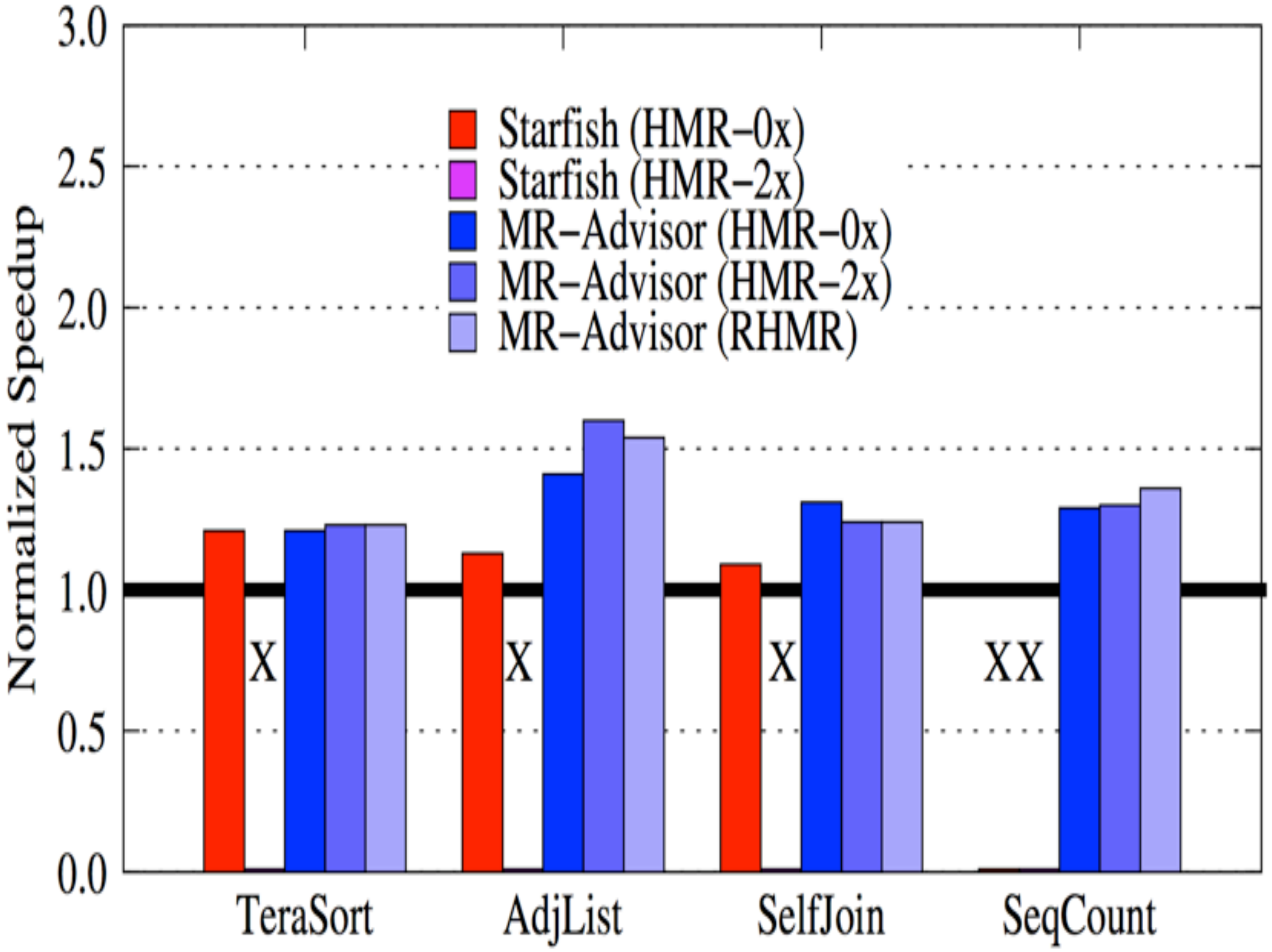
---Intel Westmere (Cluster C):

---Clusters A and C have slurm and Cluster B has PBS as the cluster resource manager.

--- refer Apache Hadoop MapReduce as HMR, RDMA-enhanced Hadoop MapReduce as RHMR, and Apache Spark MapReduce as SMR.

--- focus with batch processing workloads only

Comparison with Starfish



---MR-Advisor optimizes the job execution performance for each workload based on the performance tuning with job-level parameters (Number of maps and reduces, file system block size).

--- Starfish is only available for HMR- 0x(Hadoop-0.20.2). On the other hand, MR-Advisor is generic to any version and thus can optimize both HMR- and HMR- 2x (Hadoop-2.6.0)

Benchmark (Data Size)	Starfish	MR-Advisor
TeraSort (50 GB)	560 sec	160 sec
AdjList (30 GB)	1412 sec	439 sec
SelfJoin (80 GB)	3904 sec	660 sec

Tuning Overhead

--- Starfish employs Java- based dynamic tracing tool to obtain insights into different operation costs

--- MR-Advisor utilizes the available system resources to deploy Hadoop clusters concurrently with a different combination of tuning parameter values

--- Starfish performs 4 tuning experiments sequentially for each benchmark, whereas MR- Advisor deploys 4 concurrent clusters, each running one instance of the benchmark

Case study on Different Clusters

Case Studies
---3 case studies on different HPC systems to illustrate how MR-Advisor recommends near- optimal configuration settings based on the resource usage on that system

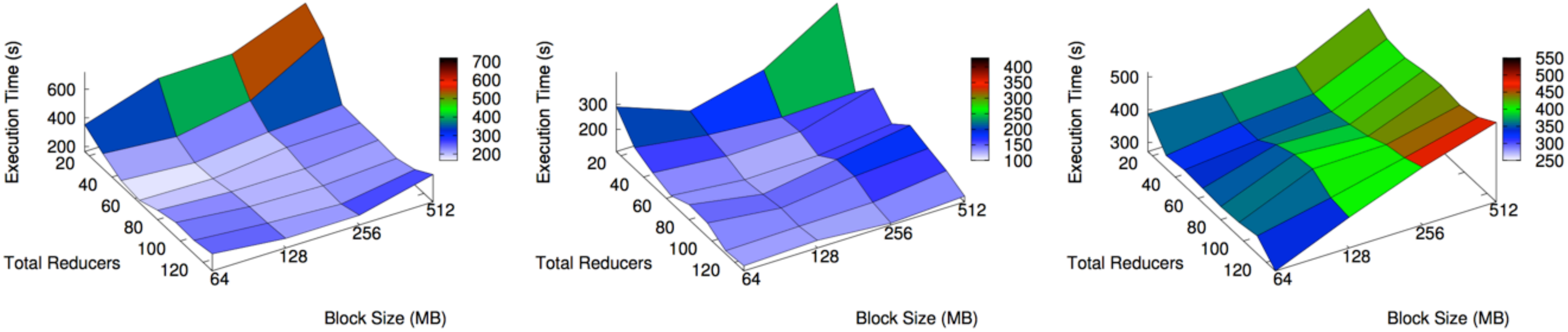
Case Study I - Cluster A with User-Space Parameters:

--- perform TeraSort experiments with a data size of 40GB on 16 Cluster A nodes

--- concurrent containers per node to six (96 concurrent containers) and vary the total reducers from 16 to 128

--- vary the file system block size from 64MB to 512MB

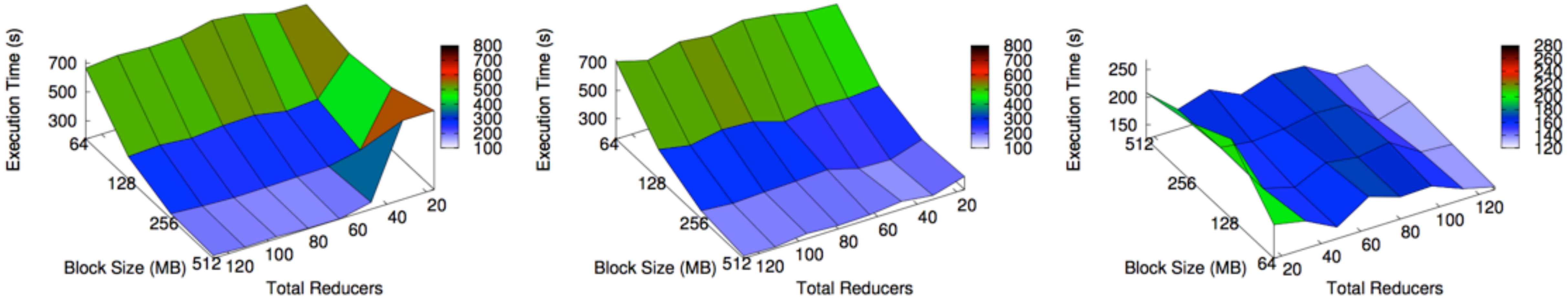
--- evaluate with different file systems



(a) HMR over HDFS (b) RHMR over HDFS (c) SMR over HDFS

--- HMR and RHMR, it optimizes with increasing number of reducers and a smaller block size.

--- SMR, the larger block sizes yield performance degradation

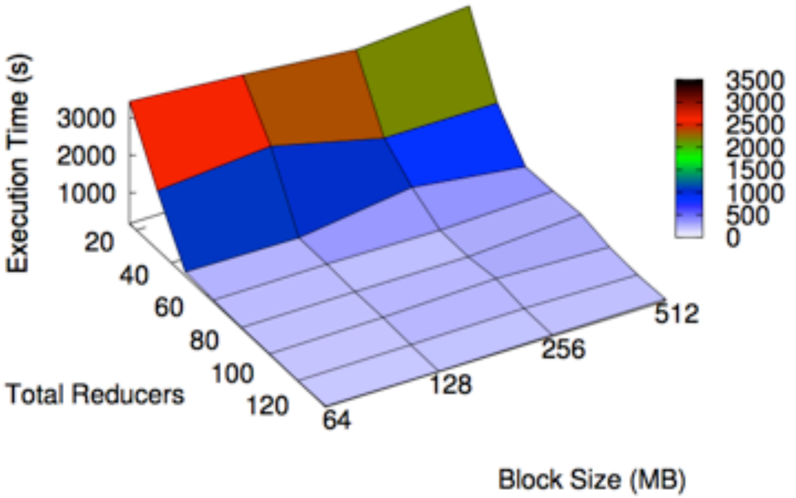


(d) HMR over Lustre

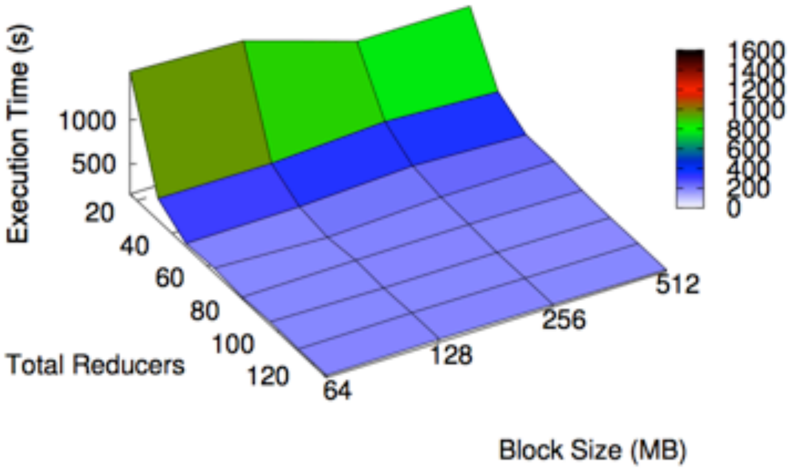
(e) RHMR over Lustre

(f) SMR over Lustre

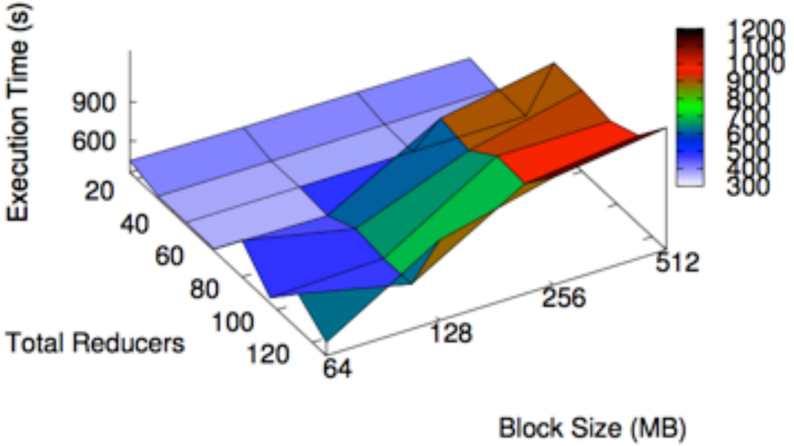
---For Lustre, larger block size (512 MB) optimizes the job execution performance more compared to the smaller block sizes,.
---the total number of reducers have very little impact on block size except for a very small number of reducers (16 for HMR and RHMR)
---SMR over Lustre, performance improves with the number of reducers irrespective of the block size.
---For these experiments, we tune the Lustre stripe size (=256 MB) and stripe count (=2) before running TeraSort experiments



(g) HMR over Tachyon



(h) RHMR over Tachyon



(i) SMR over Tachyon

---HMR and RHMR observe a similar trend over Tachyon with HDFS

--For SMR over Tachyon, we observe that with more reducers, the performance degrades considerably

Compare performance improvement obtained by MR- Advisor experiments to the current best practices used for the user-space parameters (96, 128MB)

MR stack	File System	Best configuration	Gain
HMR	HDFS	<48, 128 MB>	23%
	Lustre	<64, 512 MB>	46%
	Tachyon over HDFS	<112, 64 MB>	22%
RHMR	HDFS	<64, 128 MB>	17%
	Lustre	<32, 256 MB>	58%
	Tachyon over HDFS	<64, 128 MB>	11%
SMR	HDFS	<128, 64 MB>	34%
	Lustre	<112, 256 MB>	28%
	Tachyon over HDFS	<64, 128 MB>	17%

--- with HDFS and Tachyon over HDFS, smaller block sizes (64 MB or 128 MB) always yield better performance

--- for Lustre, larger block sizes must be used in order to achieve significant performance benefits

--- Setting number of reducers slightly less than the number of concurrent containers (e.g. 64) leads to better performance in most cases

Conclusion:
For this case study on Cluster A, MR-Advisor recommends usage of Lustre file system for MapReduce stacks with a file system block size of 256 MB and 64 reducers on 16 nodes

Case Study II - Clusters A and B with System Space Parameters:

--- present system space parameter tuning experiments

--- For space limitation, we select RHMR over Lustre [22] with Lustre mode only.

--- observe the performance difference while enabling the disk-assisted shuffle on both clusters

--- use 16 nodes for this experiment with a data size of 60GB for Sort

Feature	Options	Cluster A	Cluster B
Disk-assisted Shuffle	False	191 sec	355 sec
	True	203 sec	263 sec
Lustre-Read and RDMA Ratio	0	211 sec	328 sec
	0.5	195 sec	281 sec
	1	178 sec	309 sec

---For Cluster A, the performance degrades because of the added disk operations, with more RDMA, the job execution performance improves

--- Cluster B has significant improvement in performance with the disk- assisted shuffle, the performance improves with a hybrid of 50% RDMA and 50% Lustre Read

Conclusion:
MR-Advisor recommends utilizing disk shuffle when fast storage is available (Cluster B). For Lustre Read and RDMA ratio, MR-Advisor recommends using only RDMA on Cluster A and a hybrid of both on Cluster B

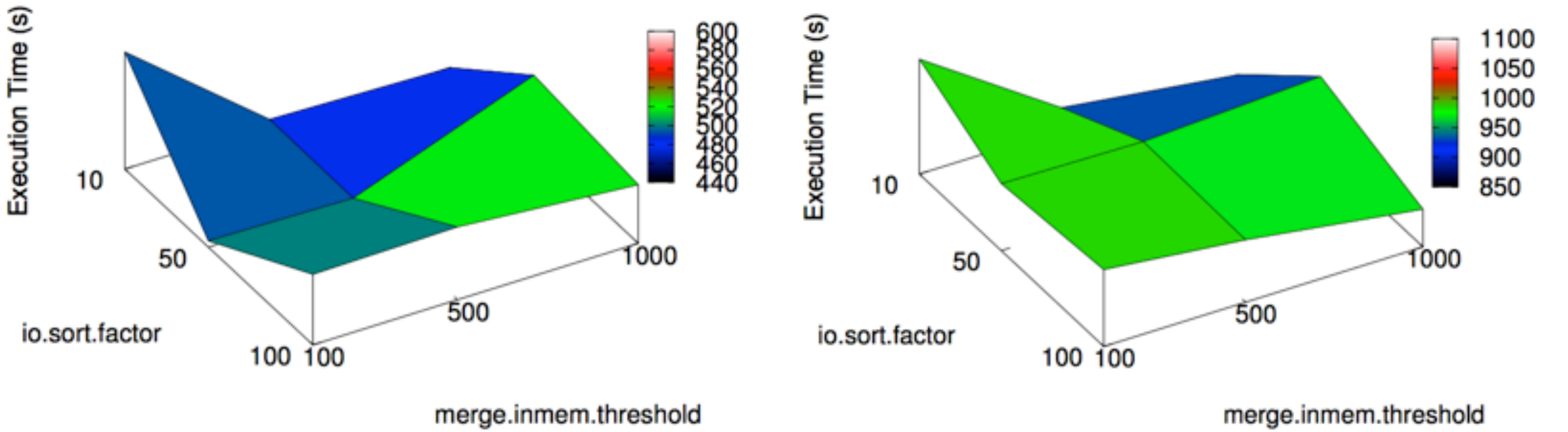
Case Study III - Cluster C with System-Space Parameters

---As Cluster C is equipped with the latest InfiniBand (EDR) adapters and a large memory space, we perform interconnect and memory tuning in this study

---choose HMR over HDFS and perform experiments with the number of in-memory and on-disk file handles during the merge operation

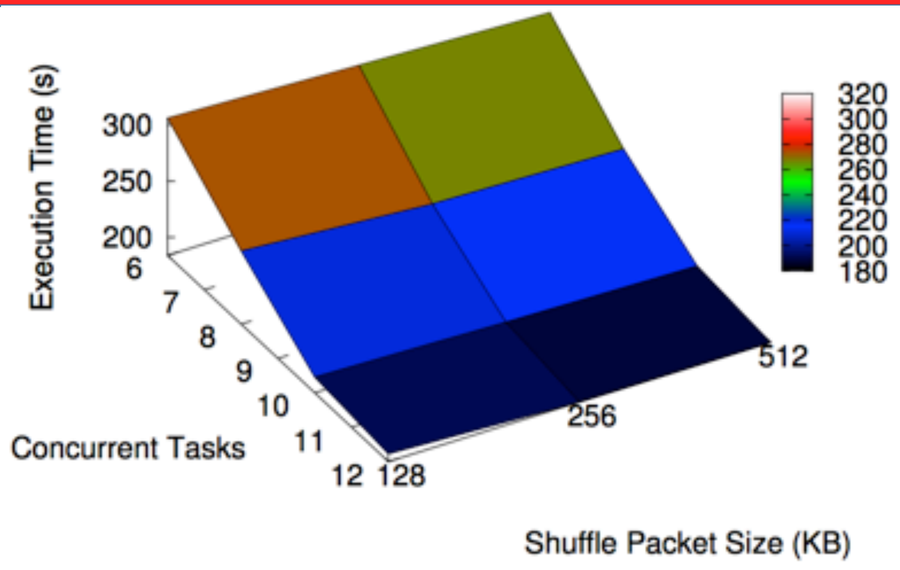
---The parameter mapreduce.reduce.merge.inmem.threshold defines the number of in-memory data segments before merge takes place, range (100 to 1,000), (default 1,000); whereas the parametermapreduce.task.io.sort.factor defines the number of open disk file handles during an on-disk merge operation, range (100 to 1,00), (default 10)

--- TeraSort (100GB) and AdjList (30 GB) benchmarks

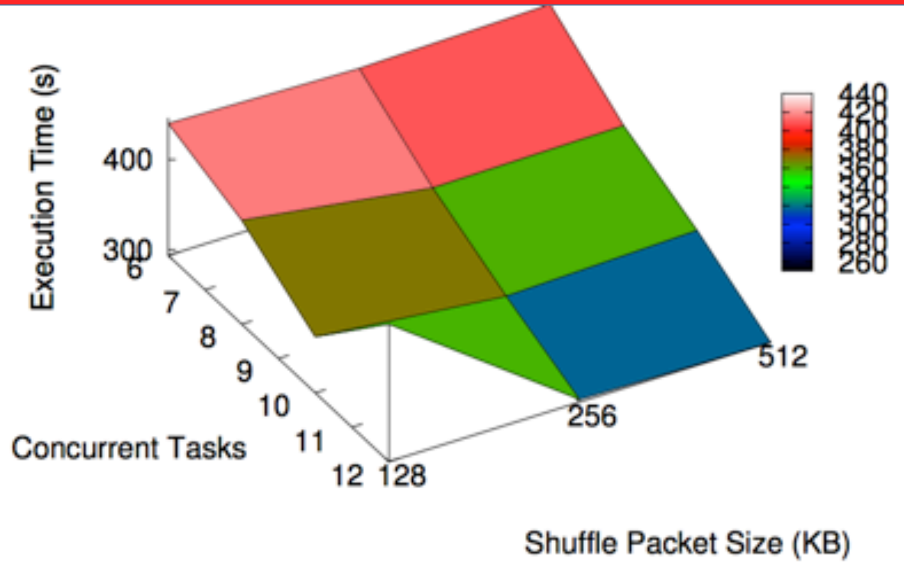


(a) TeraSort with HMR (b) AdjList with HMR

-- HMR performs better with the default values of these parameters
-- storing more in-memory data segments before in-memory merge while reducing the number of open disk file handles



(c) TeraSort with RHMR



(d) AdjList with RHMR

--optimize the shuffle packet size with the number of concurrent tasks

--vary the shuffle packet size from 128 KB (default) to 512KB with a variation of concurrent tasks from 6 to 12.

--with larger shuffle packet sizes and with more concurrent tasks, RHMR can better utilize the network bandwidth on Cluster C

Conclusion:

MR-Advisor recommends utilizing memory aggressively for both HMR and RHMR. With more memory per task and larger packet sizes, MapReduce stacks can achieve near-optimal performance in such cluster