More for Your Money: Exploiting Performance Heterogeneity in Public Clouds

Presenter: Ting Xie, tingxie2@illinois.edu

Motivation:

- We pay for virtual resources (e.g. computing cycles,storage,virtual machine instance), but logically equal resources may have different performance due to underlying hardware , contention,etc.
- ‘Good’ resources may not stay ‘Good’ all the time.

- Goal:

  Design strategies to dynamically select ‘good’ resources over time in order to maximize performance.

Subject of Analysis

- Subject: Amazon EC2 service in US-East Region
- Resources: EC2 instances
- Pricing: flat hourly rate to run VMs
- Strategy type: placement gaming (remove or add instances to the current working set)
- Trade-off: performance gain versus overhead of launching new instances
- Question: Is performance difference large enough to motivate our strategies?

Experiment on Performance Heterogeneity

- Three types of heterogeneity:

  1. Inter-architecture 2. Intra-architecture 3. Temporal

- BenchMarks used:

  CPU: slurp and Nqueens

  Memory:mcf and sphinx3

  Outbound bandwidth: iperf

  Storage: Bonnie++ on both Elastic Block Store and local disk which uses block reads and block wirtes

  Metric: normalized speedup relative to worst performing architecture

Conclusions

- The maximum speedup, in percent, of best performance over the worst

Resource	Inter-arch.	Intra-arch.	Intra-node
CPU	282%	15%	13%
Network	88%	125%	25%
Local Disk	67%	269%	20%
EBS	33%	106%	41%

Problem Formation

- Placement Model

  1. Quantum: min granularity of replacement intervals

  (here is 1 hour==min billing unit in Amazon EC2)

  2. A placement schedule P is a sequence of sets of instances.

  Let P[t] be the set of instances on time quantum t.

  3. Metric of Performance: rate of job execution of instance S at quantum t is $r_t(S)$. (job-specific, assume independence over each instance S)

  4. For a placement schedule P of duration T, the cost is the sum of set cardinality of each set in the sequence.

  5. time penalty m(overhead of lauching a new instance): the fraction of the first quantum consumed

  6. support of a placement schedule P: min num of instances preserved in any quantum.

Problem Formation

- Placement Strategies

  For each quantum t:

  1. which instances to keep: K[t] set of remaining instances
  2. which fresh instances to launch: set F[t]

  Input: P[t-1] output: K[t]+F[t] (assume markov)

  Various Objectives:

  1. minimize cost or latency given work load
  2. bound the cost and maximize work done
  3. maximize efficiency (work/cost)

  (min and max are in the sense of expected value or worst case value)

Problem Formation

Based on restricted strategy space (A,B)-strategies

(fixed cost TA+B)

- Types of Strategies

1. Up-front exploration and Opportunistic replacement
2. Gray-box strategies versus Black-box strategies

Concrete Design: PERF-M

A black box strategy that combines Up-front and Opportunistic

Algorithnm:

1. launch A+B instance at t=0
2. Keep top A, kick out those in Top A whose performance fall under threshold $\delta = \text{avg} - \frac{2m}{T - t}$, add the same number of fresh ones to reach the minimum requirement of A instances
3. for subsequent quantum, kick those fall under threshold and adds back the same number of instances.

Synthetic Simulations

- How major changes in cloud affect the strategy

1. Architechture performance variation

   If this dominates, then Up-front works better

2. Instance performance variation

   If this dominates, then opportunistic better

3. Distribution of different architectures

   if ‘good’ machine doniminates then up-front best

   if ‘bad’ doniminates,up-front performance grows linearly with parameter B

   for opportunistic, complex, but seems bell shaped.

Synthetic Simulations

- Simulating the effect of Gaming Contention

1. First group achieves first-mover advantage
2. As number of participants grows, first group ‘s performance decreases
3. First group’s gaming strategy decreases the performance of second group
4. Second group can ‘catch up’ by opportunistic replacement if total ‘good’ machines outnumber requested ones

Questions:

Different jobs and perf metrics will have different definition of ‘good’

No overall system performance simulated as m increases

Real World Application

- Strategies used (gray-box and black box)

Strategy	Up-front Exploration	Opportunistic replacement rule
CPU	CPU	Never
CPU-M	CPU	$\overline{C} - cpu(S) \geq \delta$
CPU-MAX	CPU	$cpu(S) \neq \max_i\{cpu(S_i)\}$
PERF	Performance	Never
PERF-M	Performance	$\overline{avg}_1 - r_t(S) \geq \delta$

Real World Application

- Strategies used (gray-box and black box)

Strategy	Up-front Exploration	Opportunistic replacement rule
CPU	CPU	Never
CPU-M	CPU	$\overline{C} - cpu(S) \geq \delta$
CPU-MAX	CPU	$cpu(S) \neq \max_i\{cpu(S_i)\}$
PERF	Performance	Never
PERF-M	Performance	$\overline{avg}_1 - r_t(S) \geq \delta$

Experiments on EC2

- Cost and Performance Model

  Def of Performance:

  num of records for NER jobs

  bytes should have delivered for Apache jobs

Estimate of Cost:

Assume 1 min to stop instance

Assume 2 min to start instance

Coarsed-grained billing, by hours

Costs that are not considered:

Cost of controller

Storage cost of launching new instance

Experiments on EC2

- Experiments with PERF-M

approximately 12 hours with A=10 for NER, 12 hours for Apache

Runs	Speedup	# Migrations	Total migration cost (sec)
NER 1	5.7%	13	376
NER 2	1.5%	12	402
NER 3	3.8%	16	472
Apache 1	3.7%	10	283
Apache 2	23%	7	296
Apache 3	35%	16	472

My Comments and Opinions

1. The way it combines up-front and opportunistic in its concrete design is not natural to me. use learning?
2. It seems there is no explaination of the heuristic threshold and no detailed test of how this parameter would affect the results
3. For gray-box, strategies it is essential that one really understands how CPU,storage and network bandwidth affects your job-specific performance. Otherwise I prefer black box
4. What if every body do this in Amazon aggresively(trying to beat the highest performance? Will the individual optimization lead to collective optimal collective
5. I would use machine learning to deal with the black box parameter setting rather than some predetermined heuristic criteria

Questions?

- 1. cloud provider blacklist these ‘aggressive’ players?
- 2. should cloud provider provide additional service to avoid people doing this?

PROFILING, WHAT-IF ANALYSIS, AND COSTBASED OPTIMIZATION OF MAPREDUCE PROGRAMS

Presenter: Ting Xie, tingxie2@illinois.edu

Motivation:

- Before submitting a job to a cloud computing framework,say, Hadoop, a number of choices have to be made in order to fully specify how the job should execute.
- Seems to give users the flexibility, but the performance will be greatly affected if these parameters are not properly set,
- Goal:

. Tune these parameters automatically before running or after a short period of profiling of your code.

Parameters for Tuning

1. The number of map tasks and workers
2. The number of reduce tasks and workers
3. The amount of memory to allocate to each map (reduce) task to buffer its outputs (inputs).
4. The settings for multi-phase external sorting used by most MapReduce frameworks to group map output values by key.
5. Whether the output data from the map (reduce) tasks should be compressed before being written to disk (and if so, then how).
6. Whether a program-specified Combiner function should be used

Difficulties

1. Black-box map and reduce functions:

   language like Java and C++ is not as restrictive or declarative like SQL, hard to analyze

2. Lack of schema and statistics about the input data:

   User only provides a interface or URL of the input data

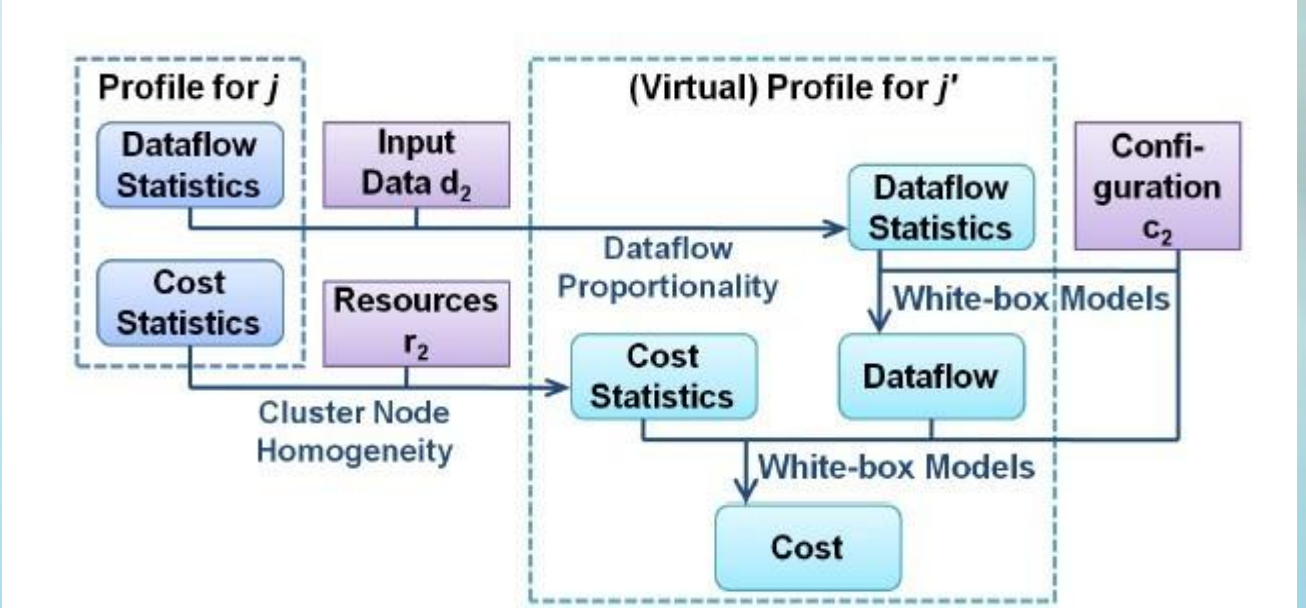
3. Differences in plan spaces:

   The execution plan space of configuration parameter settings for MapReduce programs is very different from the plan space for SQL queries.

Important Terms

Job Profile:

1. Dataflow fields
2. Cost fields
3. Dataflow Statistics fields
4. Cost Statistics fields



Cluster Node Homogeneity

Data Flow Proportionality (unless informed by system)

(usually 2 and 4 contains all factors in performance metric)

Components

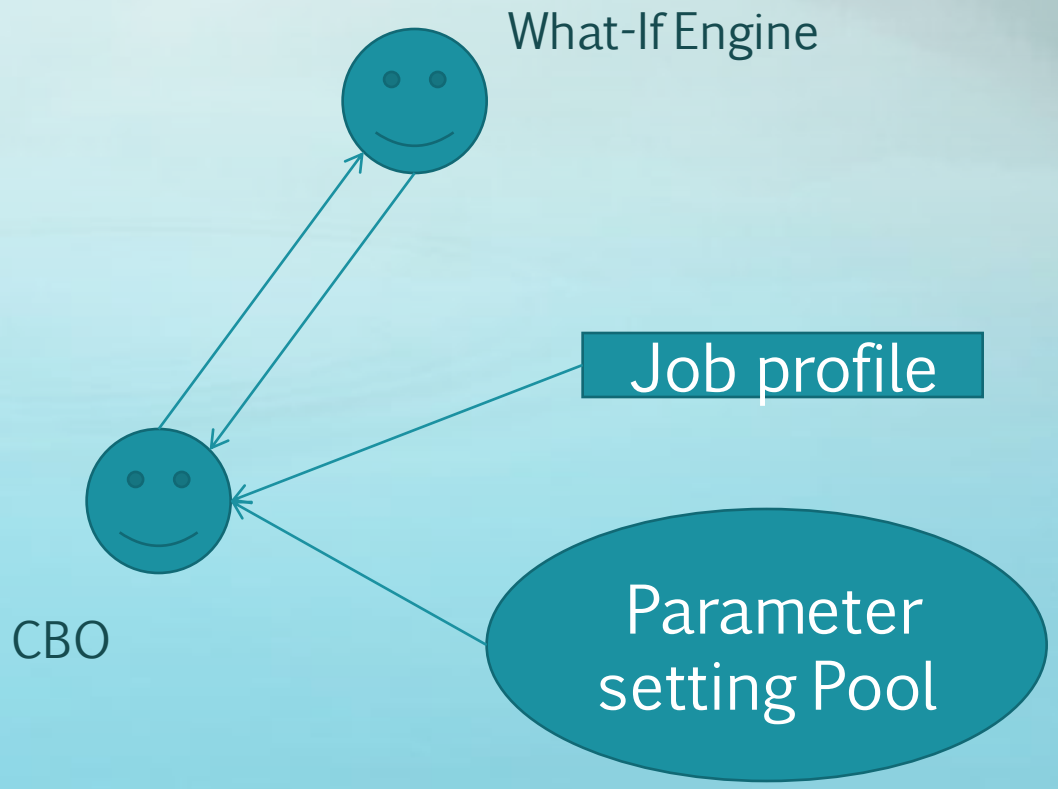
- Profiler (dynamic instrumentation)
- What if Engine
- CBO(cost-based optimizer)

Given profile, CBO efficiently search

through the setting pool to find optimal

(i) subspace enumeration

(ii) search within each enumerated subspace

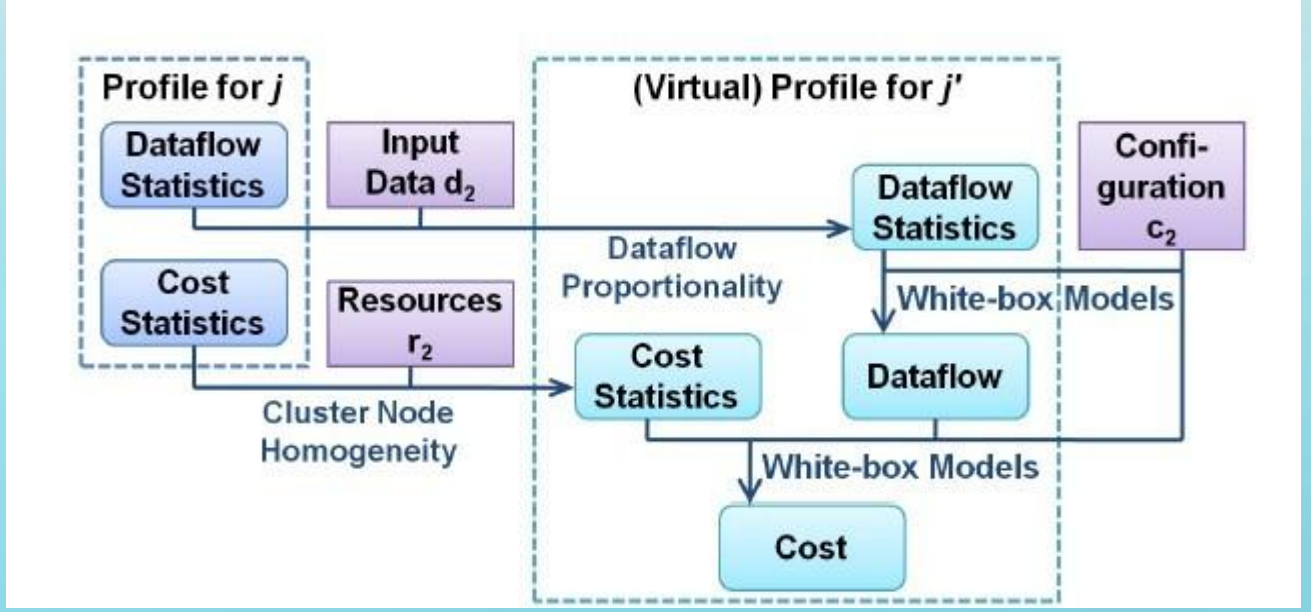


Generating Profiles or Virtual Profiles

- Via Measurement

  Task-level sampling to generate approximate profiles

- Via Estimation



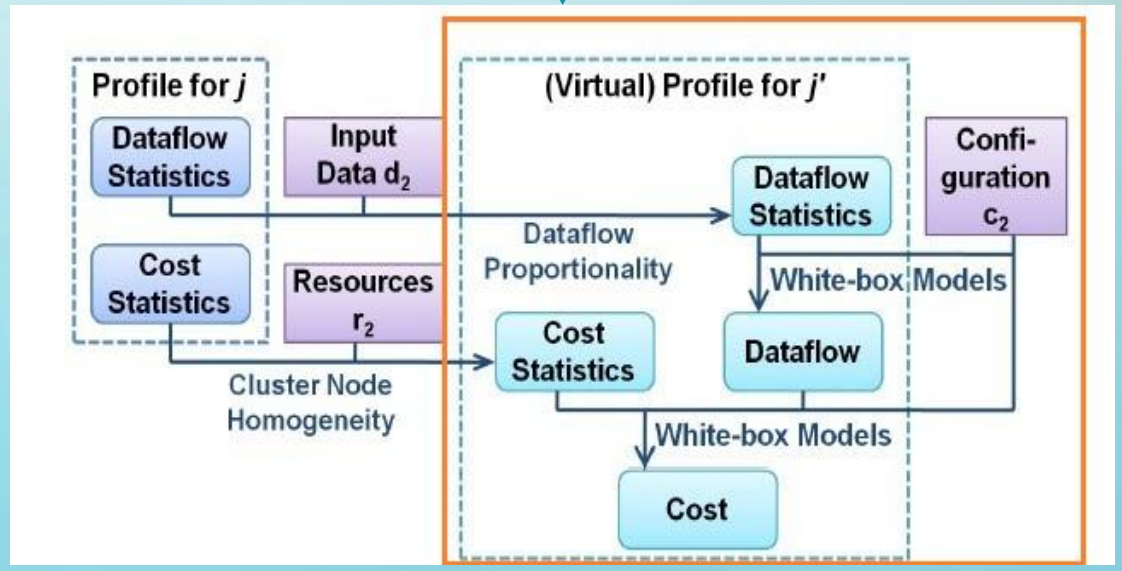
What-If Engine

- A complex module created by experts that answers questions like:

(detailed example formulas given in the appendix)

	What-if Questions on MapReduce Job Execution
WIF_1	How will the execution time of job j change if I increase the number of reduce tasks from the current value of 20 to 40?
WIF_2	What is the new estimated execution time of job j if 5 more nodes are added to the cluster, bringing the total to 20 nodes?
WIF_3	How much less/more local I/O will job j do if map output compression is turned on, but the input data size increases by 40%?

$$mapOutputPairs = \frac{mapInputSize \times mapPairsSelectivity}{mapInputPairWidth}$$



Cost-Based Optimizer (CBO)

Question: trust the profile captured by different system or not?

1. Subspace Enumeration

   cluster elements in the space first

2. Search optimal within subspace

   1) Gridding (discretization)

   2) Recursive Random Search (RRS)

   It first samples randomly to identify promising regions that contain the optimal setting with high probability. Then recursively sample a point in this region, as a starting point, to find local optimal.

3. Global optimal from all cluster optimals

Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts to tune MapReduce jobs.

Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts or past experience to tune MapReduce jobs.

MapReduce Conf. Parameter in Hadoop	Brief Description and Use	Default Value
io.sort.mb	Size (MB) of map-side buffer for storing and sorting key-value pairs produced by the map function	100
io.sort.record.percent	Fraction of io.sort.mb for storing metadata for every key-value pair stored in the map-side buffer	0.05
io.sort.spill.percent	Usage threshold of map-side memory buffer to trigger a sort and spill of the stored key-value pairs	0.8
io.sort.factor	Number of sorted streams to merge at once during multiphase external sorting	10
mapreduce.combine.class	The (optional) Combiner function to preaggregate map outputs before transfer to reduce tasks	null
min.num.spills.for.combine	Minimum number of spill files to trigger the use of Combiner during the merging of map output data	3
mapred.compress.map.output	Boolean flag to turn on the compression of map output data	false
mapred.reduce.slowstart.completed.maps	Proportion of map tasks that need to be completed before any reduce tasks are scheduled	0.05
mapred.reduce.tasks	Number of reduce tasks	1
mapred.job.shuffle.input.buffer.percent	% of reduce task's heap memory used to buffer output data copied from map tasks during the shuffle	0.7
mapred.job.shuffle.merge.percent	Usage threshold of reduce-side memory buffer to trigger reduce-side merging during the shuffle	0.66
mapred.inmem.merge.threshold	Threshold on the number of copied map outputs to trigger reduce-side merging during the shuffle	1000
mapred.job.reduce.input.buffer.percent	% of reduce task's heap memory used to buffer map output data while applying the reduce function	0
mapred.output.compress	Boolean flag to turn on the compression of the job's output	false

Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts or past experience to tune MapReduce jobs.

Conf. Parameter (described in Table 4)	RBO Settings	CBO Settings
io.sort.factor	10	97
io.sort.mb	200	155
io.sort.record.percent	0.08	0.06
io.sort.spill.percent	0.80	0.41
mapred.compress.map.output	TRUE	FALSE
mapred.inmem.merge.threshold	1000	528
mapred.job.reduce.input.buffer.percent	0.00	0.37
mapred.job.shuffle.input.buffer.percent	0.70	0.48
mapred.job.shuffle.merge.percent	0.66	0.68
mapred.output.compress	FALSE	FALSE
mapred.reduce.tasks	27	60
min.num.spills.for.combine	3	3
Use of the Combiner	TRUE	FALSE

Cost-Based Vs. Rule-Based Optimizer

Rule Based Optimizer is based on rules of thumb used by Hadoop experts to tune MapReduce jobs.

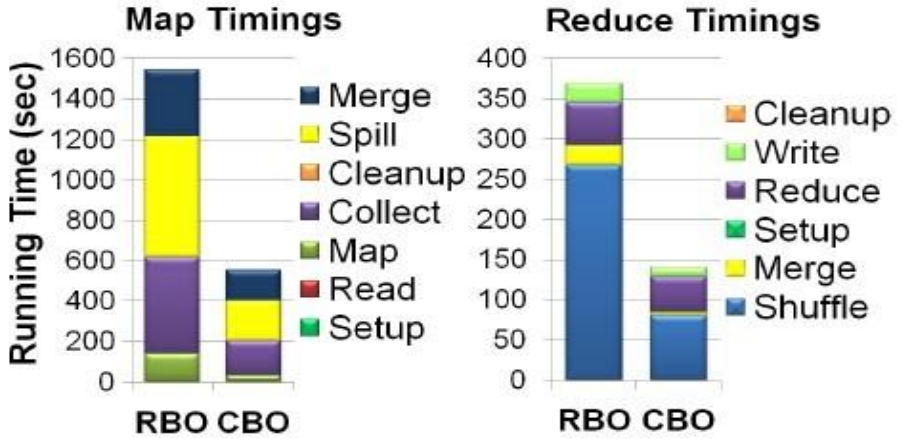
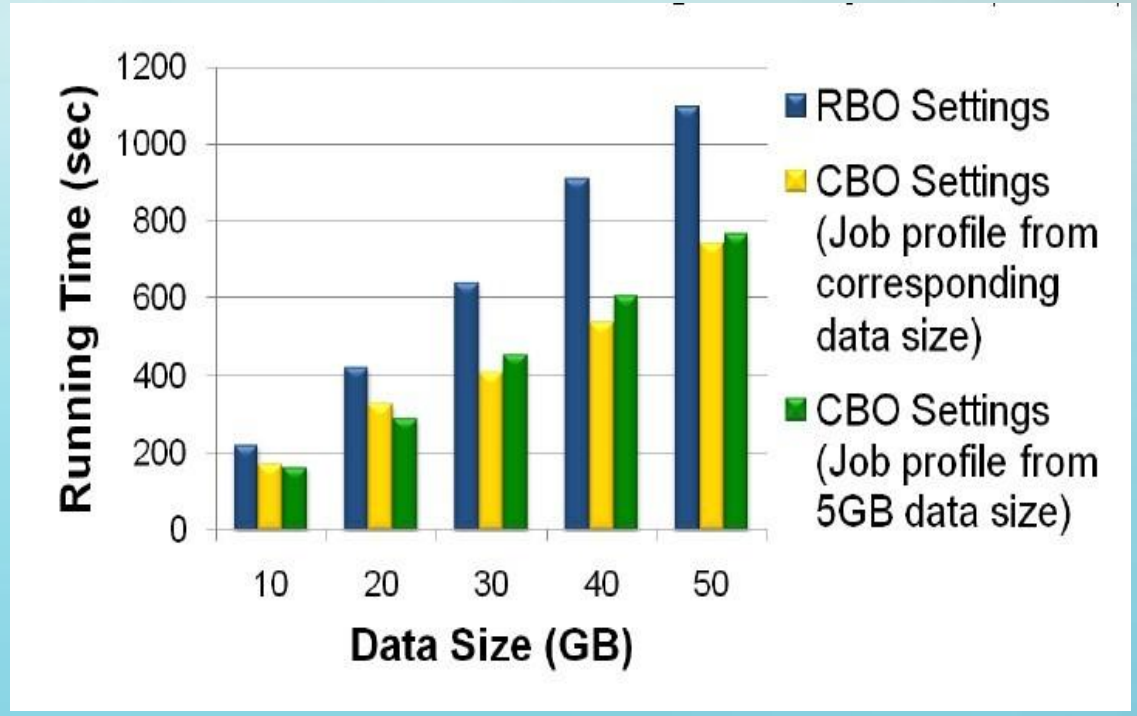


Figure 7: Map and reduce time breakdown for two CO jobs run with configuration settings suggested by RBO and CBO.



Efficiency and Effectiveness of CBO

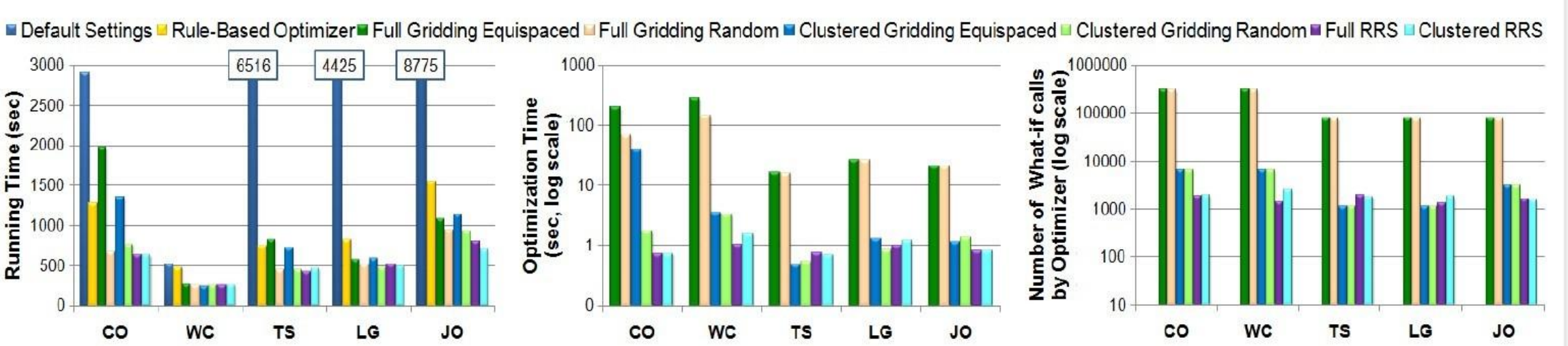


Figure 11: (a) Running times for all MapReduce jobs running with Hadoop's Default, RBO-suggested, and CBO-suggested settings; (b) Optimization time, and (c) Number of what-if calls made (unique configuration settings considered) by the six CBOs.

Accuray of What if Analysis

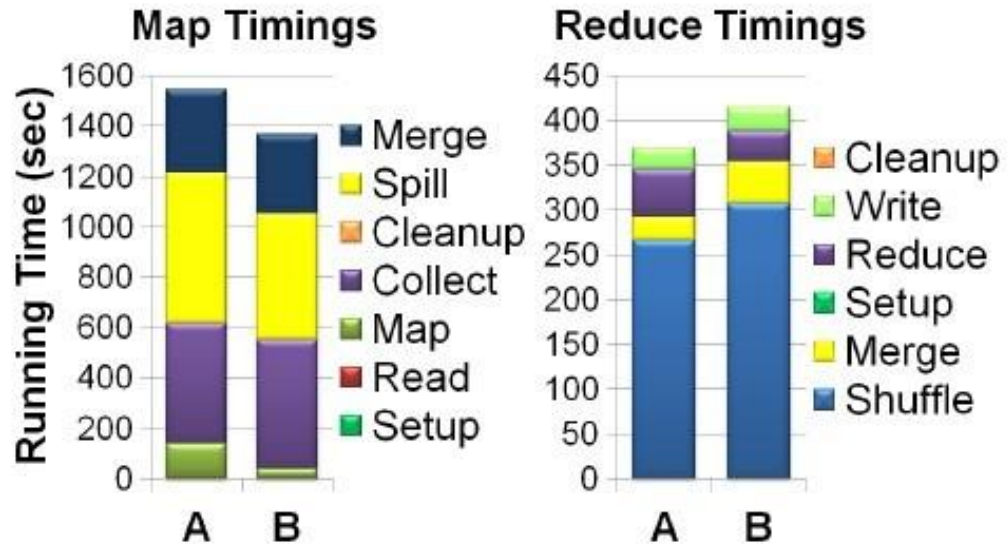


Figure 8: Map and reduce time breakdown for CO jobs from (A) an actual run and (B) as predicted by the What-if Engine.

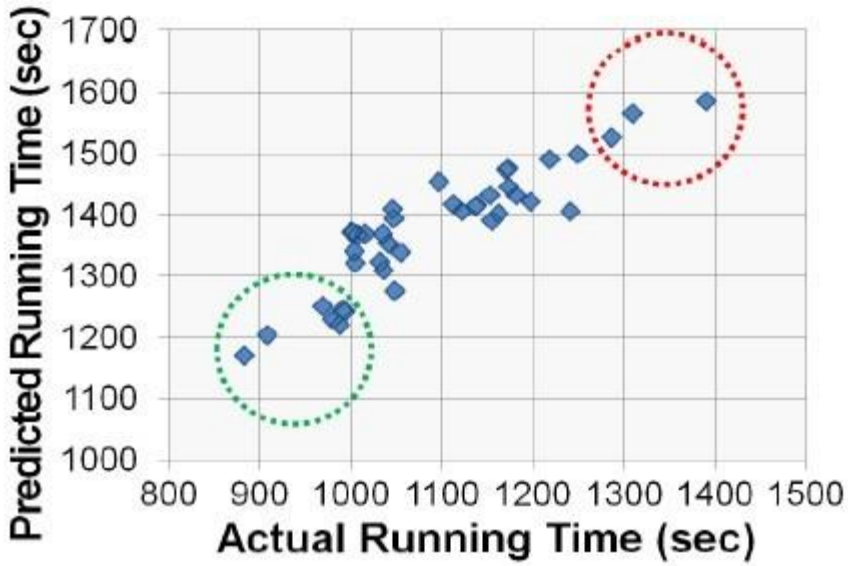
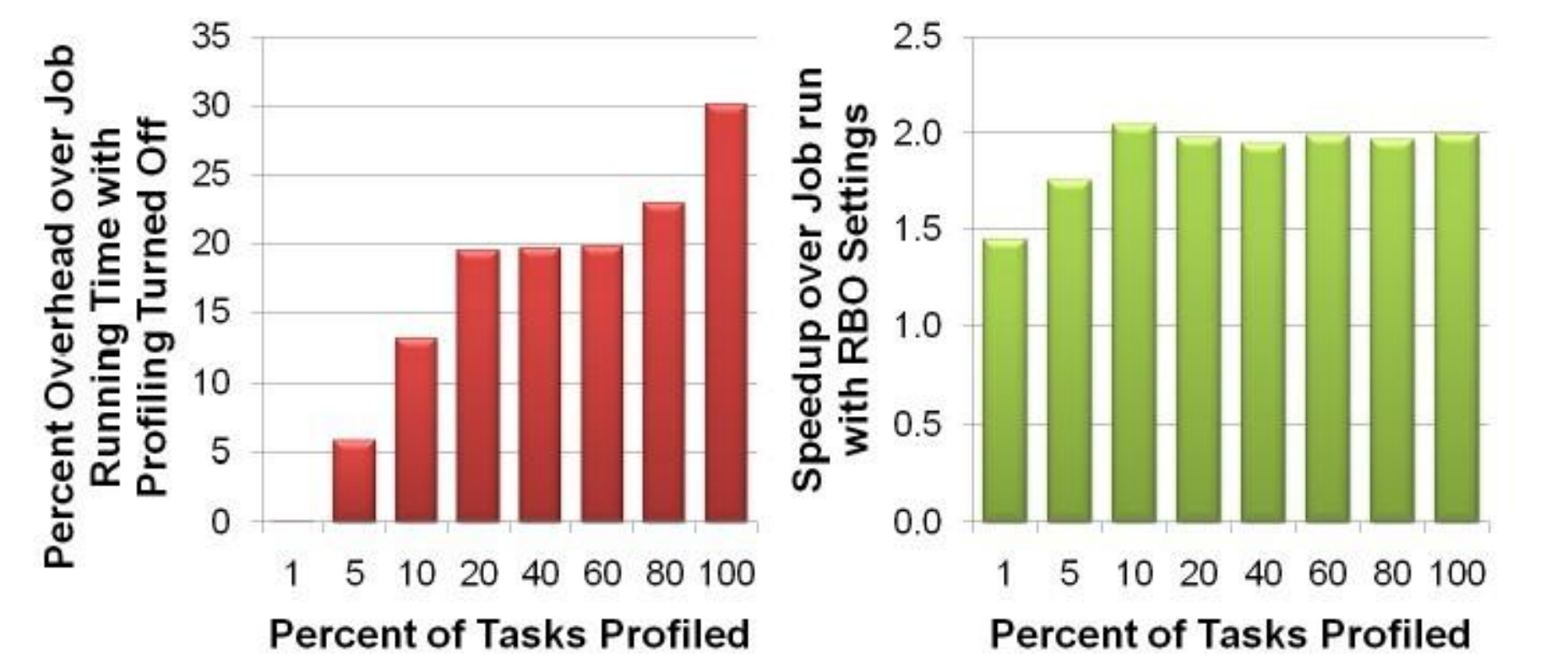


Figure 9: Actual Vs. Predicted (by What-if Engine) running times for CO jobs run with different configuration settings.

Approximate Profiles through Sampling



Future Work

Challenge:

Some systems submit several jobs together in the form of job workflows which exhibit data dependencies.

In addition, the optimization space now grows to include logical decisions such as selecting the best partitioning function, join operator, and data layout.

Future Direction:

Integrate the What-if Engine with tools like data layout and dynamic resource allocators for complex MapReduce workflows

My Comments and Opinions

1. Prediction could be done at the cloud computing provider side using learning (assume the system is stable over time,namely the history record is reliable till now). Current optimal setting may be far from global optimal, there will be some other profile-equivalent programs with different setting achieve higher performance.
2. The assumption of data flow proportionality and node homogeniety is controversial
3. level of sampling really matters, there is a trade-off between finer level sampling vs. overhead brought by it.

Questions?