Towards an Adaptive, Fully Automated Performance Modeling Methodology for Cloud Applications

Ioannis Giannakopoulos [1], Dimitrios Tsoumakos [2] and Nectarios Koziris [1]

[1]:Computing Systems Laboratory, School of ECE, National Technical University of Athens, Greece
{ggian, nkoziris}@cslab.ece.ntua.gr
[2]: Department of Informatics, Ionian University, Greece
dtsouma@ionio.gr

Introduction

Performance Modeling (or profiling)

*"The ability to **predict** an application's behavior, given the parameters that affect it as input"*

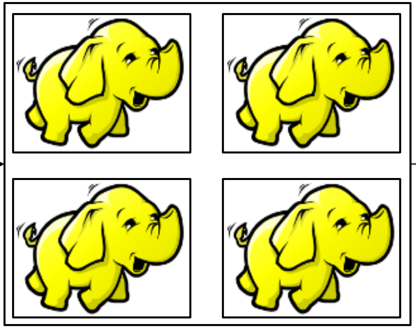
Valuable for:

- Optimal Resource Configuration
- Bottleneck Identification
- Application Scaling
- Multi-Engine Analytics

Approaches

Existing approaches:

- White-box
  - Analytical Modeling
  - Interpretable Models
  - Good Accuracy for simple cases
  - Require Experience
  - Poor Results for complex cases



Variables	Expressions
D_{m-avg}^{output}	The average output data size of a map task.
T_m^{total}	The total execution time of a map phase.
D_{m-avg}^{input}	The average input data size of a map task.
$M_{selectivity}$	The map selectivity which is the ratio of a map output to a map input.
N_m	The total number of map tasks.
T_m^{avg}	The average execution time of a map task.
N_m^{slot}	The total number of configured map slots.
D_{Sh-avg}	The average size of a shuffled data.
T_{sh}^{total}	The total execution time of a shuffle phase.
N_r	The total number of reduce tasks.
T_{sh}^{avg}	The average execution duration of a shuffle task.
N_r^{slot}	The total number of configured reduce slots.
N_{sh}^{w1}	The total number of shuffle tasks that complete in the first wave.
N_{sh}^{w2}	The total number of shuffle tasks that complete in other waves.
T_{w1}^{avg}	The average execution time of a shuffle task that completes in the first wave.
T_{w2}^{avg}	The average execution time of a shuffle task that completes in other waves.
D_{r-avg}^{output}	The average output data size of a reduce task.
T_r^{total}	The total execution time of a reduce phase.
D_{r-avg}^{input}	The average input size of a reduce task.
$R_{selectivity}$	The reduce selectivity which is the ratio of a reduce output to a reduce input.
T_r^{avg}	The average execution time of a reduce task.

$$D_{m-avg}^{output} = D_{m-avg}^{input} \times M_{selectivity}$$

$$T_m^{total} = \frac{T_m^{avg} \times N_m}{N_m^{slot}}$$

$$D_{Sh-avg} = \frac{D_{m-avg}^{output} \times N_m}{N_r}$$

$$T_{sh}^{total} = \frac{(T_{w1}^{avg} \times N_{sh}^{w1}) + (T_{w2}^{avg} \times N_{sh}^{w2})}{N_r^{slot}}$$

$$D_{r-avg}^{output} = D_{r-avg}^{input} \times R_{selectivity}$$

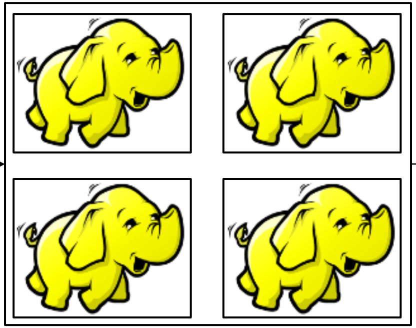
$$T_r^{total} = \frac{T_r^{avg} \times N_r}{N_r^{slot}}$$

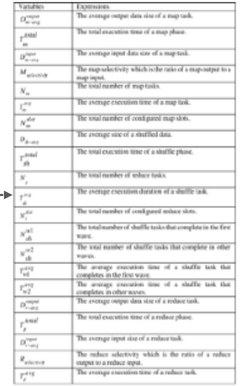
Approaches

Existing approaches:

- White-box
  - Analytical Modeling
  - Interpretable Models
  - Good Accuracy for simple cases
  - Require Experience
  - Poor Results for complex cases
- Black-box
  - Benchmarking & ML
  - Require no experience
  - Accuracy
  - Require time/computation

WordCount





$$D_{m-avg}^{output} = D_{m-avg}^{input} \times M_{selectivity}$$

$$T_m^{total} = \frac{T_m^{avg} \times N_m}{N_m^{slot}}$$

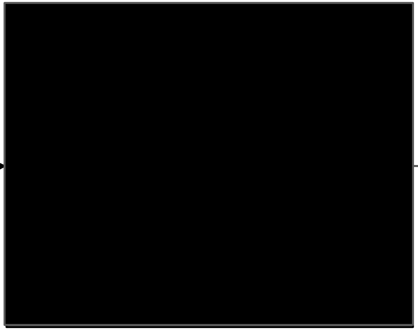
$$D_{Sh-avg} = \frac{D_{m-avg}^{output} \times N_m}{N_r}$$

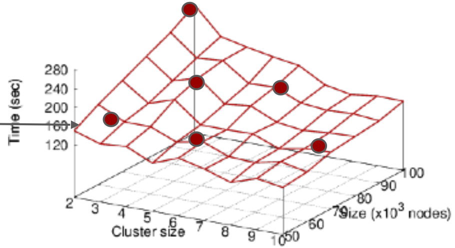
$$T_{sh}^{total} = \frac{(T_{w1}^{avg} \times N_{sh}^{w1}) + (T_{w2}^{avg} \times N_{sh}^{w2})}{N_r^{slot}}$$

$$D_{r-avg}^{output} = D_{r-avg}^{input} \times R_{selectivity}$$

$$T_r^{total} = \frac{T_r^{avg} \times N_r}{N_r^{slot}}$$

WordCount
Nodes
Cores
Data Size





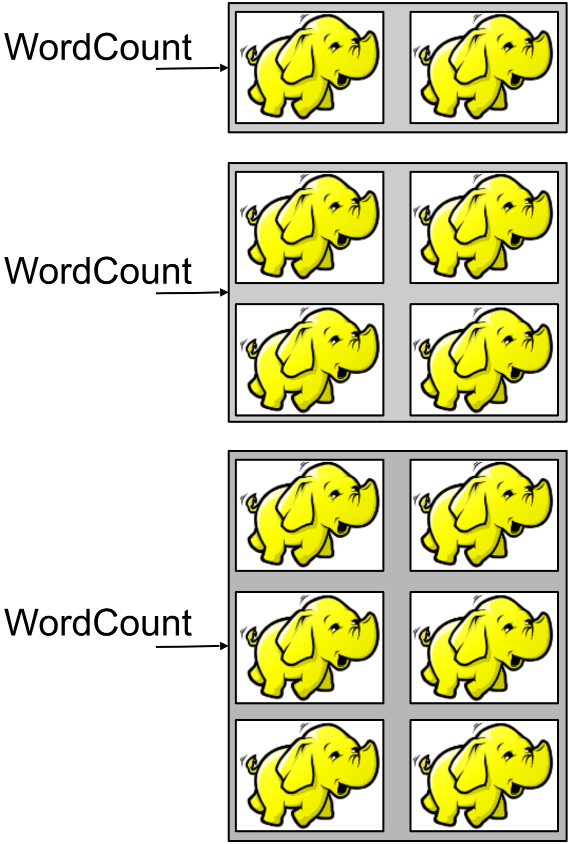
“Black-box” Profiling Challenges

Traditional Challenges:

1. Which dimensions?
   a. data (e.g., dataset size, # of columns, distribution, etc.)
   b. resources (e.g., # nodes, # cores, RAM, etc.)
   c. workload (e.g., k parameter of k-means, clustering implementation, etc.)
2. Which model?
   a. Linear Models
   b. Neural Networks
   c. Support Vector Machines

New Challenge: **Massive Configuration Space**

The challenges - Distributed Design



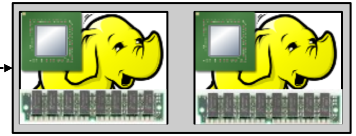
Nodes = [2]

Nodes = [2,4]

Nodes = [2,4,6]

The challenges - Cloud Computing

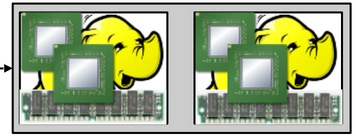
WordCount



Nodes = [2,4,6]
Cores = [1]
RAM= [1]

3 combinations

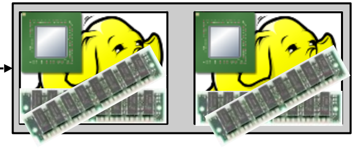
WordCount



Nodes = [2,4,6]
Cores = [1,2]
RAM= [1]

6 combinations

WordCount



Nodes = [2,4,6]
Cores = [1,2]
RAM= [1,2]

12 combinations

The configuration space grows exponentially with the #dimensions and the #values per dimension!

Problem statement

Given:

1. Application A
2. Configuration space $D = d_1 \times d_2 \times \ldots \times d_n$
3. A positive number B

"Estimate an application profile F_A:D→R using D_s of size B, with the highest possible accuracy!"

or:

*"Find the top-B **representative** configurations to train a ML classifier"*

Optimality & Observations

Optimality & Observations

Optimal solution is **NP-Hard:**

- For each possible set of size B:
  - Deploy configuration
  - Train ML classifier
  - Evaluate it (test set)

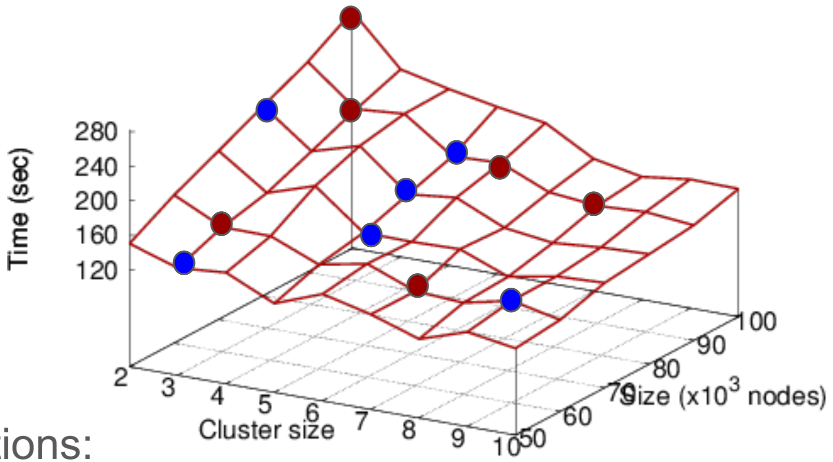
Even deploying the entire D is not feasible:

- Prohibitive time + cost

Two observations regarding performance functions:

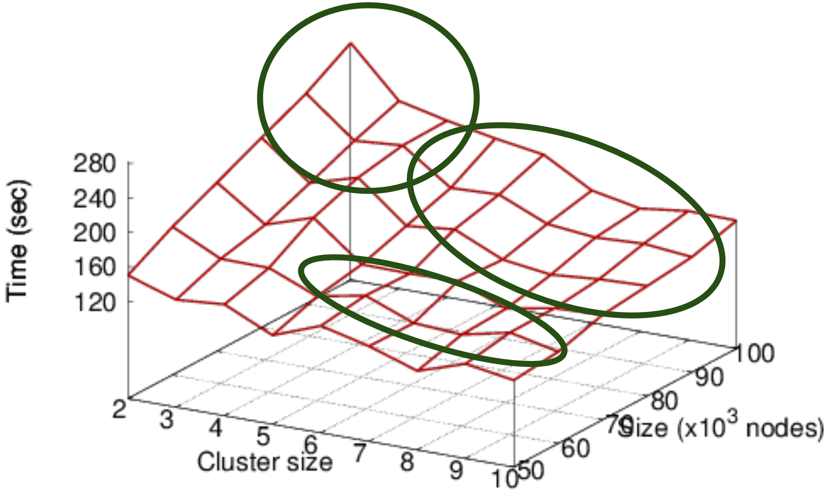
- Locality
- (piecewise) Linearity

Set 1 and Set 2



Locality

“Neighboring” configurations (tend to) give similar performance



Hint:

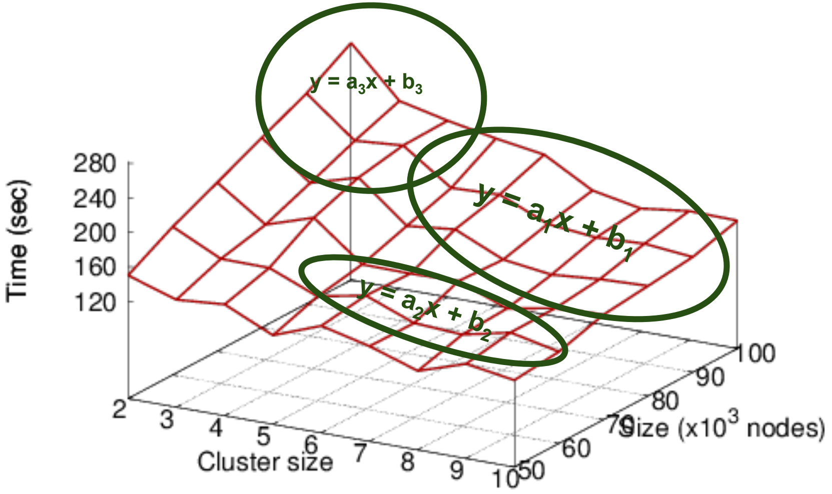
Divide-and-Conquer to solve more (but simpler) problems

(piecewise) Linearity

“Neighborhoods” (tend to) present linear behavior (especially for resource-related dimensions)

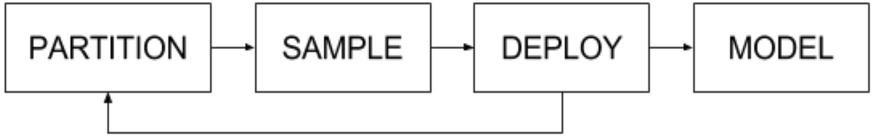
Hint:

Start with linear models



Methodology

Overview



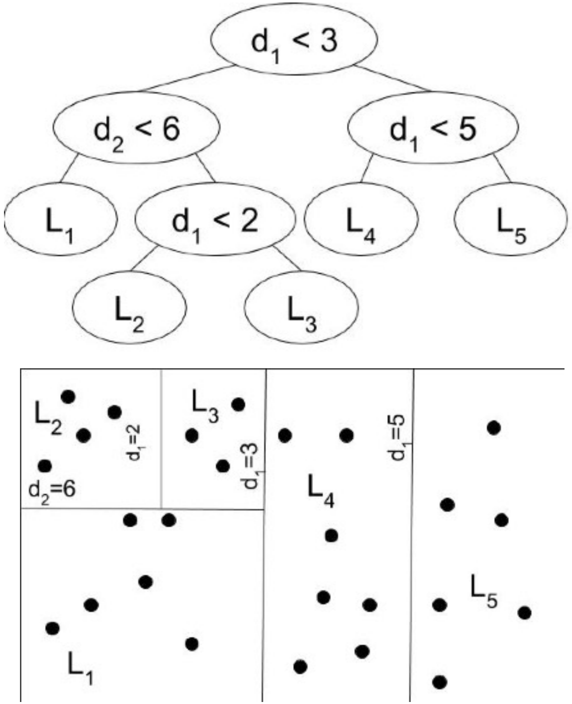
Algorithm 1 DT-based Adaptive Profiling Algorithm

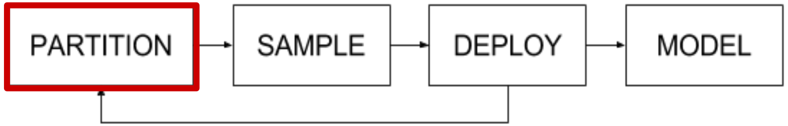
procedure DTADAPTIVE(D, B, b)
  $tree \leftarrow$ TREEINIT($\emptyset$), $samples \leftarrow \emptyset$
  while $|samples| \leq B$ **do**
    $tree \leftarrow$ PARTITION($tree$, $samples$)
    $s \leftarrow$ SAMPLE(D, $tree$, $samples$, b)
    $d \leftarrow$ DEPLOY(s)
    $samples \leftarrow samples \cup d$
  $model \leftarrow$ CREATEMODEL($samples$)
  return $model$

Space Partitioning with Decision Trees

Decision Trees:

- Tree Structures
- Space Partitioning
- Node types:
  - Test nodes: Space Boundaries
  - Leaves: Linear Models



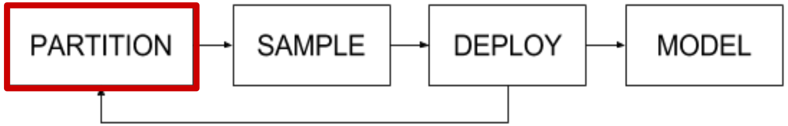


Space Partitioning with Decision Trees

Space Partitioning → Decision Tree construction

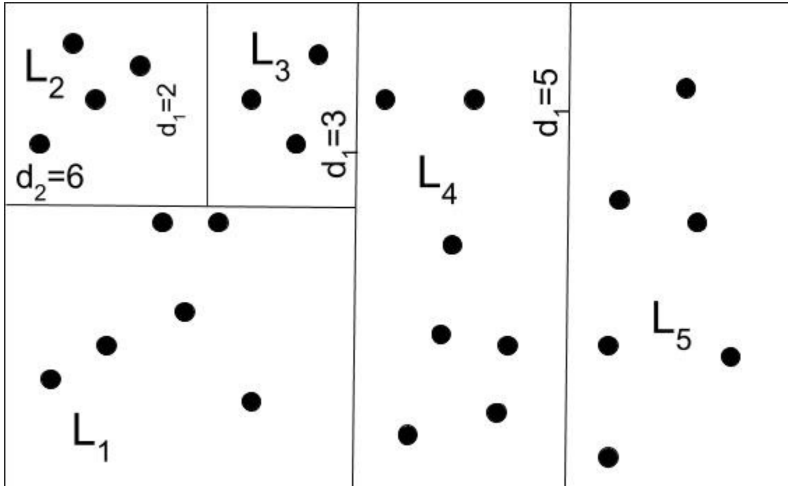
- Substitute Leaves with Test nodes
- Which line to use?
  - Maximize leaf homogeneity
  - Well-known criteria: GINI Impurity, Entropy, Variance minimization, etc.
- Optimization Problem
  - Objective Function: $Score(l) = -\frac{|L_1|R_1^2 + |L_2|R_2^2}{|L_1| + |L_2|}$

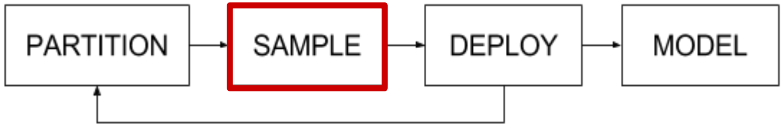
Intuitively: the split line will generate two regions which best fit to linear models



Space Sampling

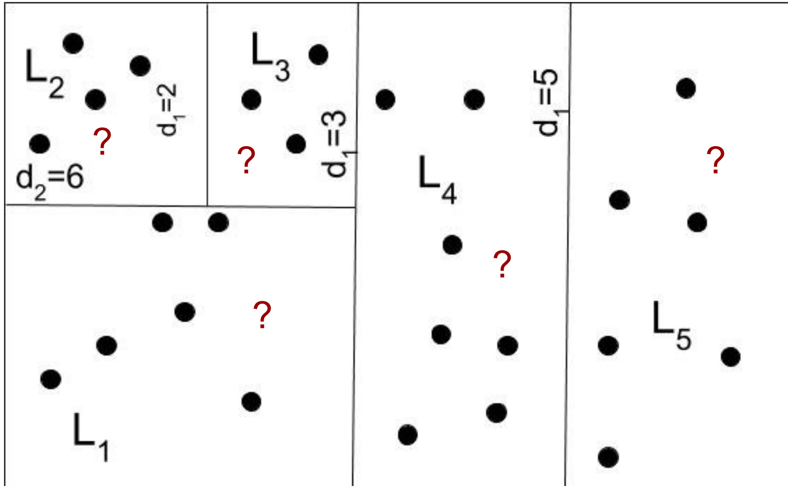
Objective: distribute a deployment budget **b** (b<B) to each leaf

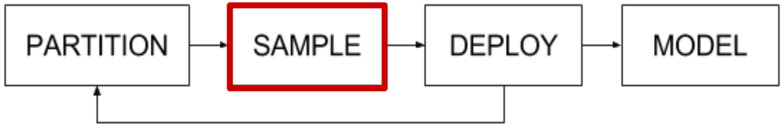




Space Sampling

Objective: distribute a deployment budget **b** (b<B) to each leaf



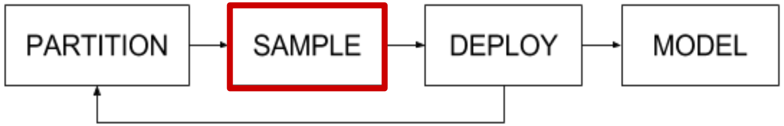


Space Sampling

- How can we decide on the budget of **each leaf?**
  - Leaf error (exploitation)
  - Leaf size (exploration)

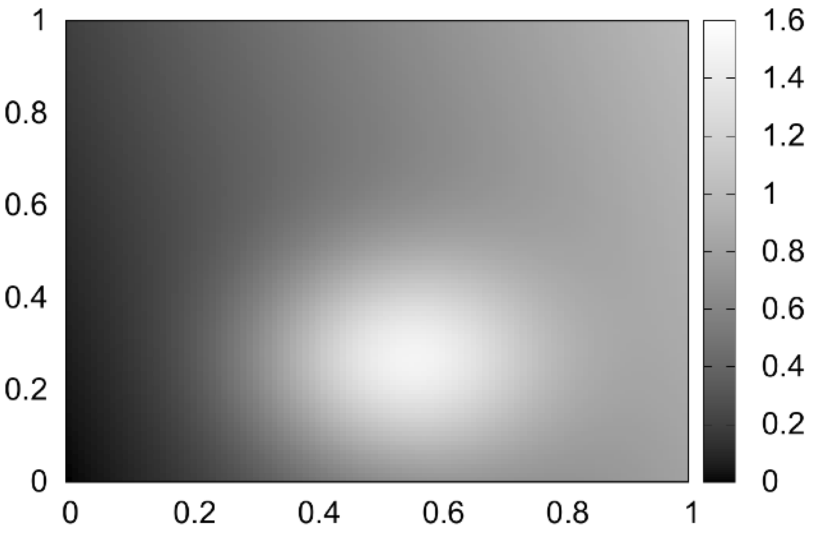
- Leaf's score: $$Score(leaf) = w_{error} \cdot \frac{error(leaf)}{maxError} + w_{size} \cdot \frac{size(leaf)}{maxSize}$$

- # of samples assigned to each leaf *l*: *Score(l) x b*
- Uniform sampling inside the leaf

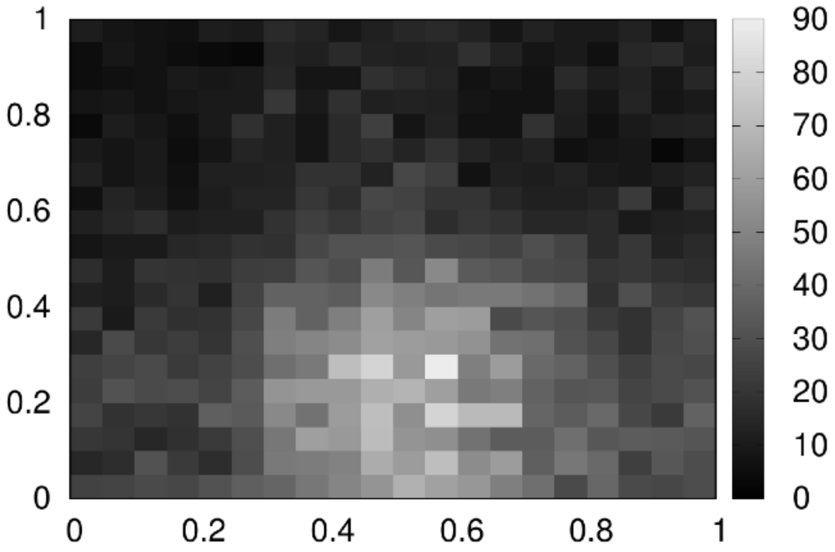


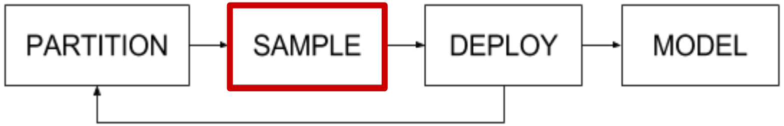
Space Sampling

Exploration vs Exploitation



$W_{error} = 1.0$
$W_{size} = 0.0$

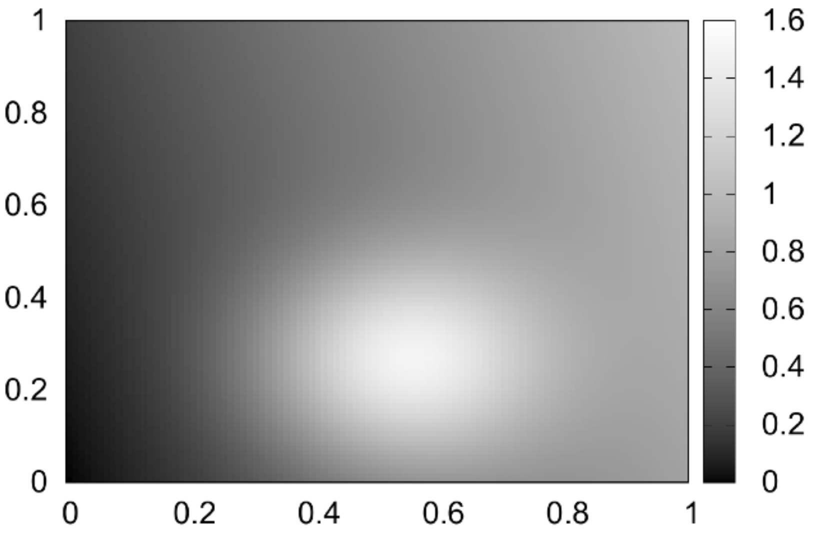


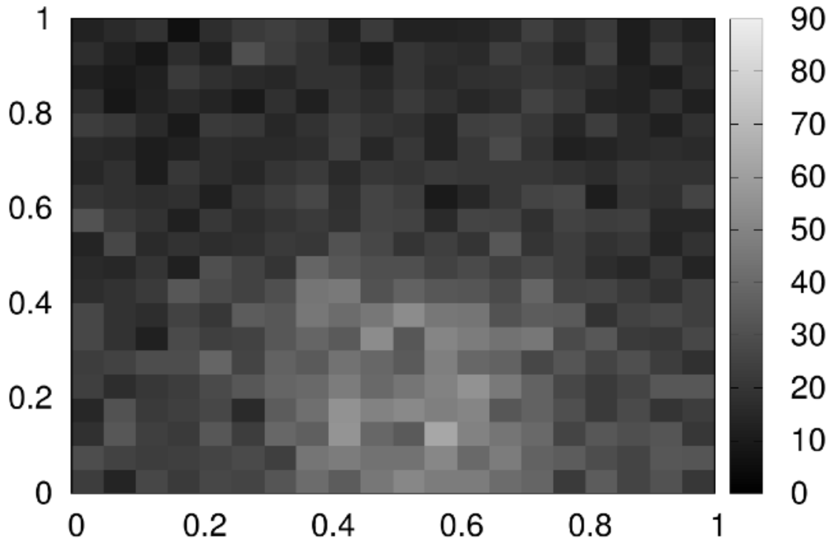


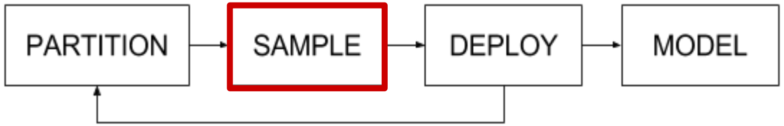
Space Sampling

Exploration vs Exploitation

$W_{error} = 1.0$
$W_{size} = 0.5$

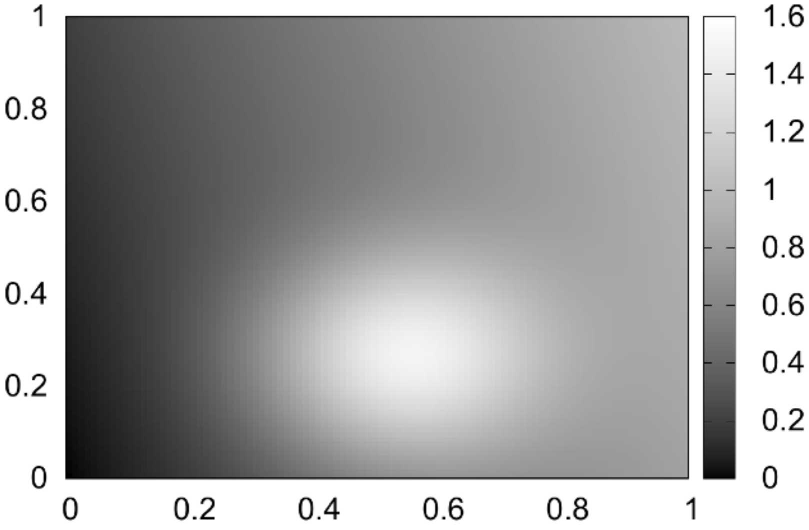






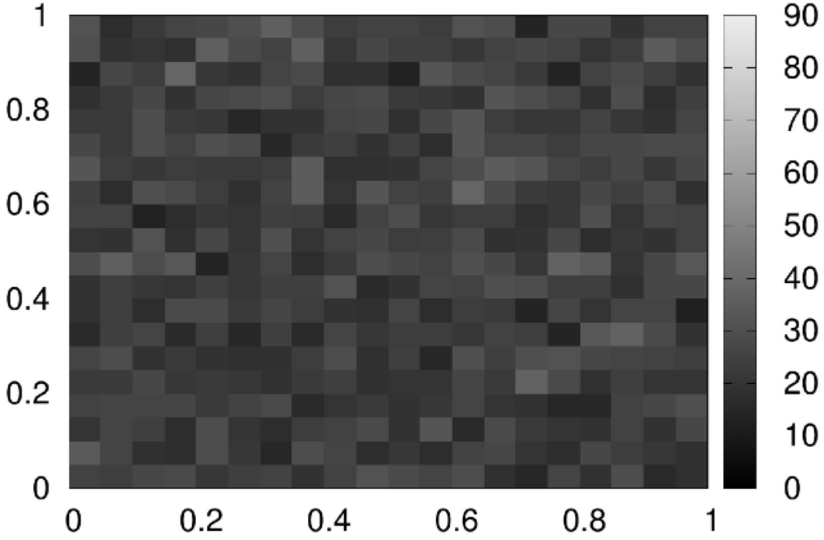
Space Sampling

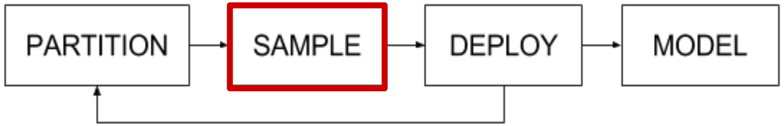
Exploration vs Exploitation



$W_{error} = 0.0$

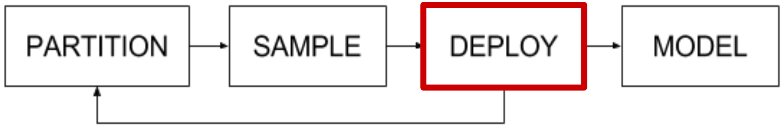
$W_{size} = 1.0$



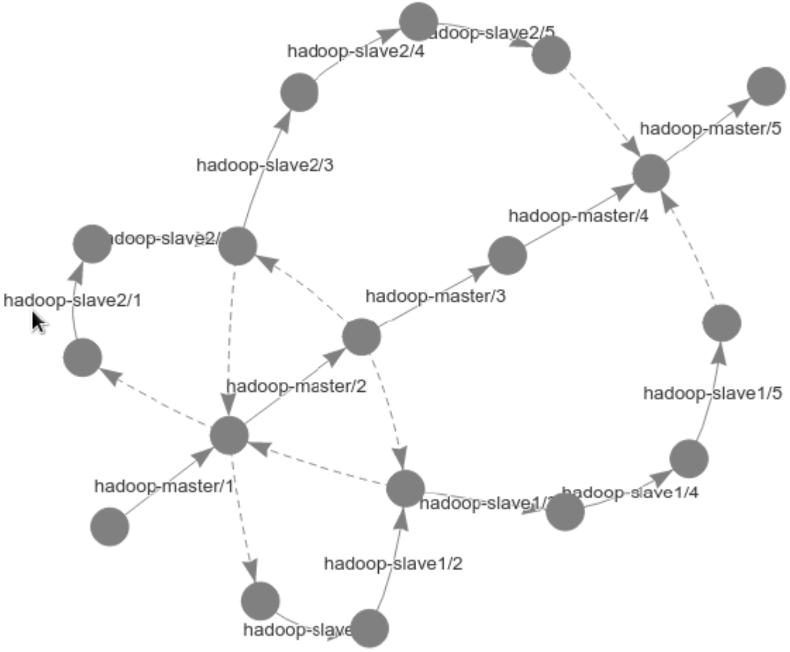


Deployment

- AURA: A Python-based deployment tool with error-recovery enhancements
- Application description: DAG of scripts for each module
- Transient failures (network glitches, poor coordination, etc.)
- AURA re-executes only scripts that failed
- Achieve idempotence via lightweight filesystem snapshotting

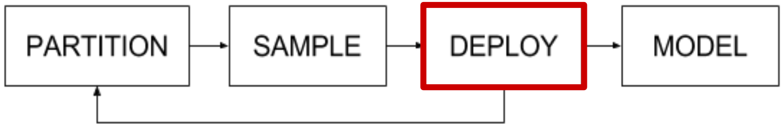


Deployment DAG example



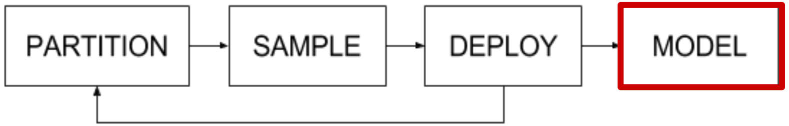
Hadoop Application Description:

- 1 Master node
- 2 Slave nodes



Modeling

- Training a new Decision Tree from scratch
- When **B** comparable to #dimensions, DT degenerates into linear regression
  - Use multiple Regressors (e.g., ANNs, SVMs, etc.)
  - Keep the one with the lowest Cross Validation score
  - In practice, such budgets are **too low** to extract meaningful models
- Piecewise linearity
  - Approximates complex functions given higher deployment budgets



Evaluation

Methodology & Performance Functions

- Testbed: private Openstack Cluster (8 nodes, 200 cores, 600G RAM, 8TB disk)
- Competitors:
  - PANIC — exploitation
    - More samples between points samples with highest variance
  - Active Learning (Uncertainty Sampling) — exploitation
    - More samples to areas of highest uncertainty
  - Uniform Sampling — exploration
    - Uniform distribution of samples
- Modeling with WEKA regressors

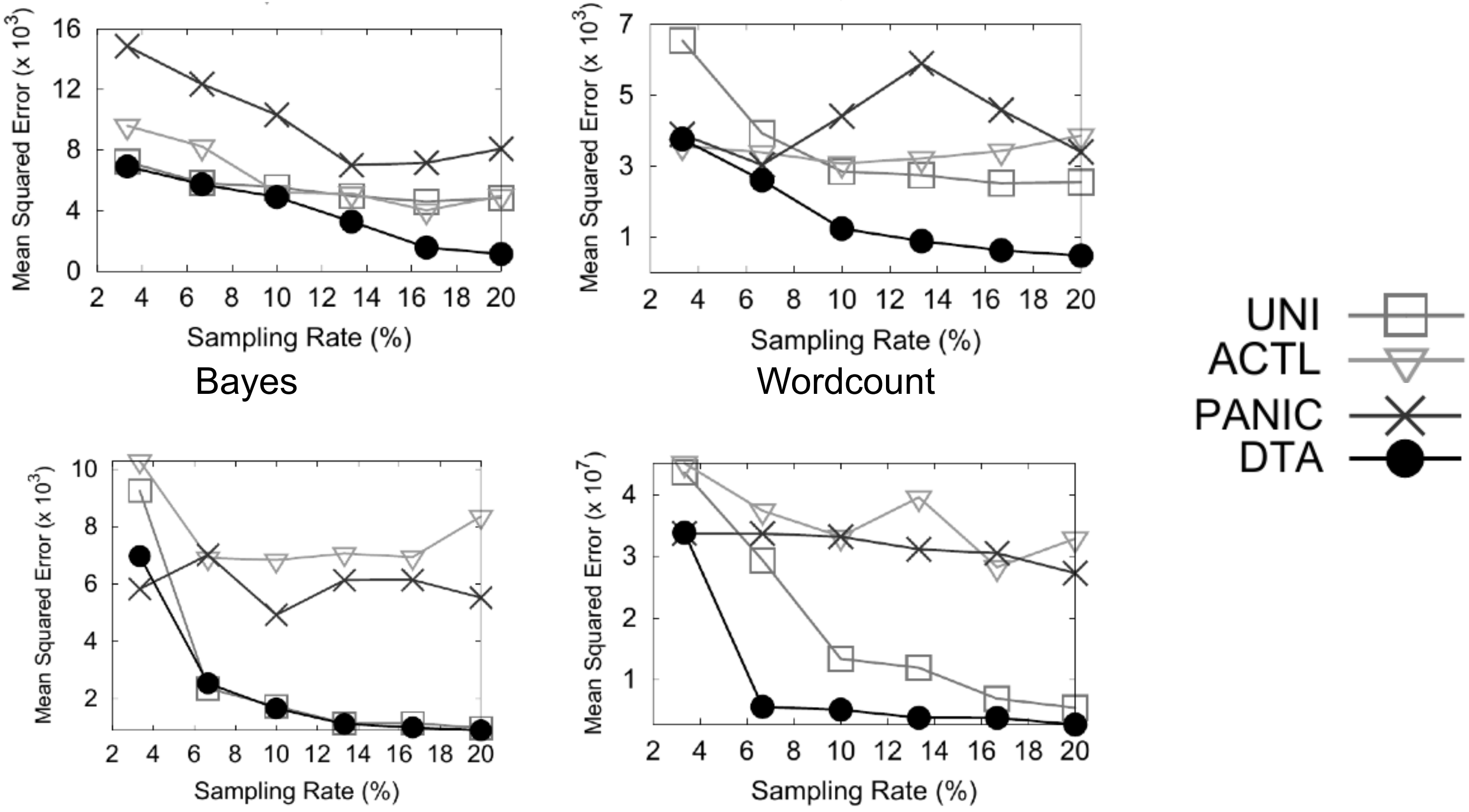
National Technical University of Athens
CSLab

Methodology & Performance Functions

Application (perf. metric)	Dimensions	Values
Spark Bayes (execution time)	YARN nodes	4–20
	# cores per node	2–8
	memory per node	2–8 GB
	# of documents	0.5–2.5 ($\times 10^6$)
	# of classes	50–200 classes
Hadoop Wordcount (execution time)	YARN nodes	2–20
	# cores per node	2–8
	memory per node	2–8 GB
	dataset size	5–50 GB
Media Streaming (throughput)	# of servers	1–10
	video quality	144p–1440p
	request rate	50–500 req/s
MongoDB (throughput)	# of MongoD	2–10
	# of MongoS	2–10
	request rate	5–75 ($\times 10^3$) req/s

SR: $|D_s| / |D| \times 100\%$

Modeling Error



Bayes

Wordcount

Media Streaming

MongoDB

Cost-aware profiling

$$Score(leaf) = w_{error}\frac{error(leaf)}{maxError} + w_{size}\frac{size(leaf)}{maxSize} - w_{cost}\frac{cost(leaf)}{maxCost}$$

App/tions	SR	MSE			Cost		
		0.2	0.5	1.0	0.2	0.5	1.0
Bayes	3% 20%	+1% +4%	-1% +10%	-2% +5%	-1% -7%	-1% -9%	-1% -12%
Wordcount	3% 20%	-1% 0%	-5% +13%	-1% +19%	-4% -6%	-6% -8%	-1% -18%
Media Str.	3% 20%	-1% -6%	-3% -2%	+3% -8%	-1% -7%	-5% -11%	-6% -26%
MongoDB	3% 20%	+6% -1%	+12% -7%	+13% -7%	-2% -6%	-3% -9%	-4% -12%

Conclusions

- Divide-and-Conquer strategy
- “zoom-in” to *interesting* areas of the configuration space
- exploration vs exploitation
- Space Partitioning: important dimensions are partitioned more frequently
- Dimension Importance: linearity


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
YAQHANYELAY
SUKSAMA
EKHMET
BİYAN
SHUKRIA
TINGKI
GRAZIE
MEHRBANI
PALDIES
BOLZÏN
MERCI
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
KOMAPSUMNIDA
MAAKE