DaMoOp*
An Interactive Framework for Cost-Aware NoSQL Data Modeling [WISE26]



Man Vijaybhai Patel, Jihane Mali, Nicolas Travers

De Vinci Higher Education, De Vinci Research Center, Paris, France



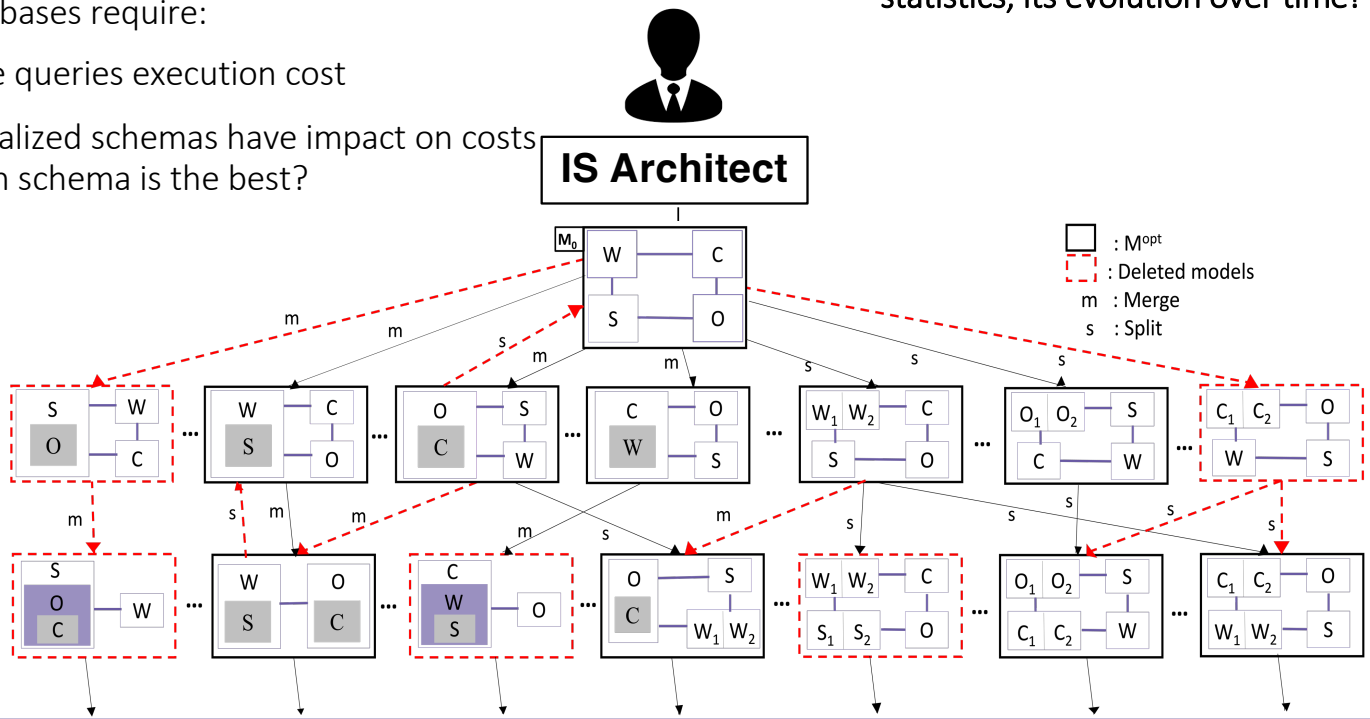
PROBLEMATIC

In a Big Data storage environment,

NoSQL databases require:

- Estimate queries execution cost
- Denormalized schemas have impact on costs
  => which schema is the best?

It depends on queries, on workload & statistics, its evolution over time!



STATE OF THE ART

Schema Denormalization

- Rule-based approaches
  [Chebotko, Big Data 2015; Abdelhedi, DaWaK 2017]
- Workload-driven optimization
  [Mior, IEEE TKDE 2017; Hewasinghage, EDBT 2021]
- Model-driven approaches
  [de la Vega, FGCS 2020]
- Multi-objective schema design
  [Hewasinghage, EDBT 2021; Chen, DKE 2022]

➢ *Single optimized schema vs full denormalization space*

Cost Estimation: Analytical Models

- Analytical cost models
  [Mior, IEEE TKDE 2017; Forresi, ISF 2023]
- Simulation-based approaches
  [Mali, Information Systems 2026] => *DaMoOp*
- Schema migration cost
  [Hillenbrand, SIGMOD 2019; Gallinucci, Information Systems 2026]

➢ *Repeated cost estimation when configurations evolve*

Workload Evolution and Adaptation

- Static workload optimization / snapshots
  [Mior, IEEE TKDE 2017; Hewasinghage, EDBT 2021]
- Workload forecasting
  [Ma, SIGMOD 2018; Huang, PACMMOD 2024]
- Dynamic database reconfiguration
  [Mahgoub, USENIX ATC 2019]
- Schema evolution & migration under time-dependent workloads
  [Wakuta, arXiv 2023]

➢ *Workload forecasting, schema selection, and cost estimation are typically addressed independently*

DRIVING EXAMPLE USING DAMOOP [INFORMATION SYSTEMS JOURNAL'26]

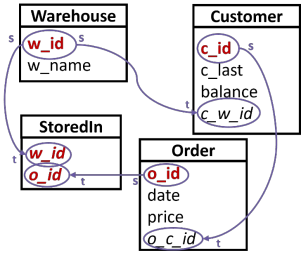
Inputs
Normalized schema + query workload

→

DaMoOp pipeline
DMG · MCC · DMS

→

Generated data models
With time, financial and environmental costs



Denormalized examples:

M0: {C(c_ID,c_last,balance,w_c_ID)}, {O(o_ID,o_carrier_id,c_o_ID)}, {W(w_ID,w_name,w_city)}

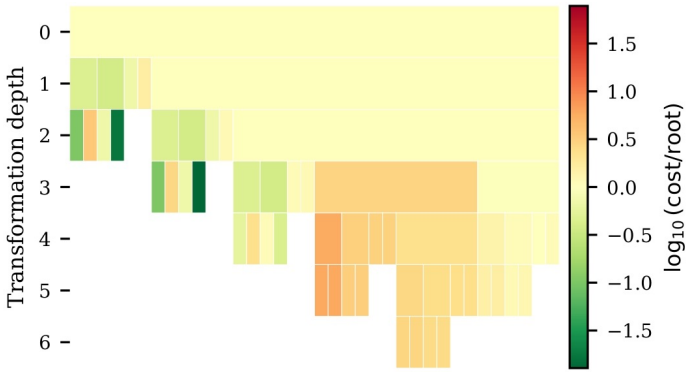
M16: {C1(c_ID,c_last), C2(c_ID, balance), C3(c_ID, orders[O(o_ID,o_carrier_id,c_o_ID)])}, {W(w_ID,w_name,w_city)}

M30: {C(c_ID,c_last,balance,w_c_ID, orders[O(o_ID,o_carrier_id,c_o_ID)])}, {W(w_ID,w_name,w_city)}

M134: {O(o_ID,o_carrier_id, C(c_ID,c_last,balance,w_c_ID))}, {W(w_ID,w_name,w_city)}

M135: {O(o_ID,o_carrier_id, C(c_ID,c_last,balance, W(w_ID,w_name,w_city)))}

- *DMG*: 1012 different denormalized data models.
  Out of which, 135 can be considered relevant according to queries.
  (attributes in filter & join queries)
- *MCC*: Cost simulation of each data model a various given settings.
  With big workloads, costs differ by up to 14 orders of magnitude between candidates (can vary with time, finance & carbon footprint)
- *DMS*: DaMoOp chooses the best compromise (stability)



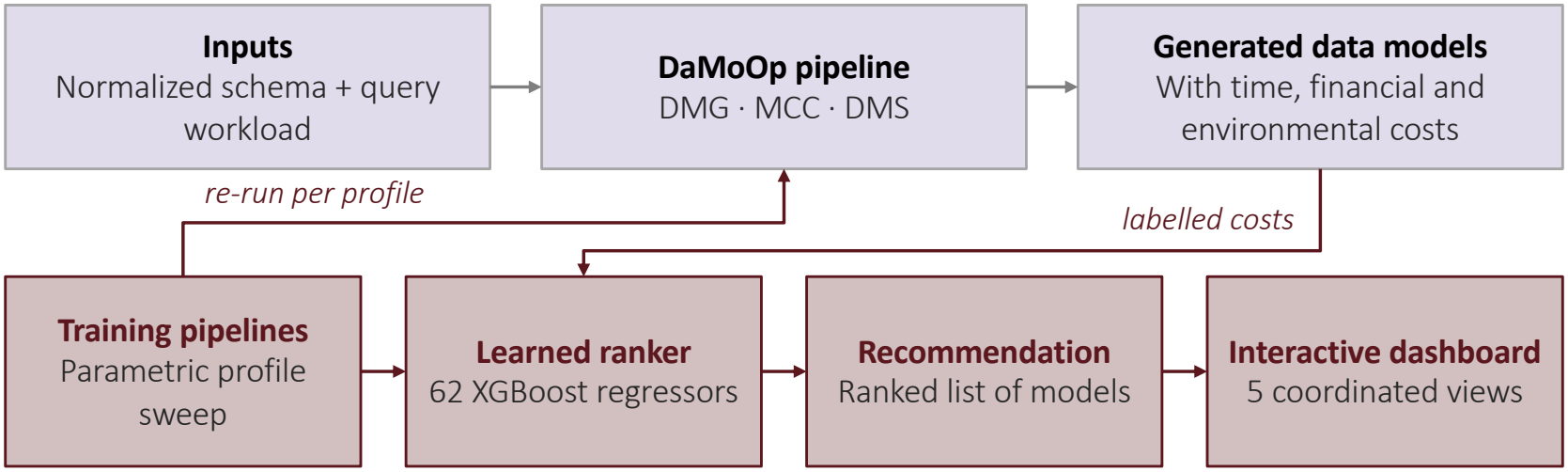
ISSUES

- Simulation time: ~8s per single simulation
  - Data model (schema) + queries (DB profile) + workloads (scenario)
  - To search the space => millions of simulations
  - The most frequent winner is optimal in only 28.8% of cases

➢ How to design the search space of the workload?

➢ How to predict cost efficiently?

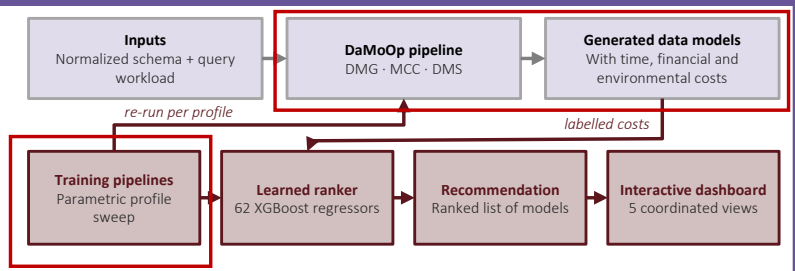
DAMOOP*



New in DaMoOp*

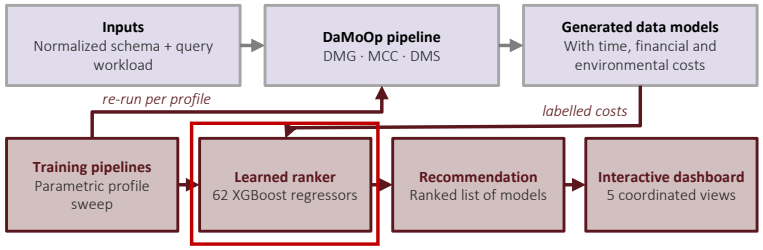
Labelled dataset generation (simulation based) + cost prediction (ML + analytical)

BUILDING THE LABELLED DATASET



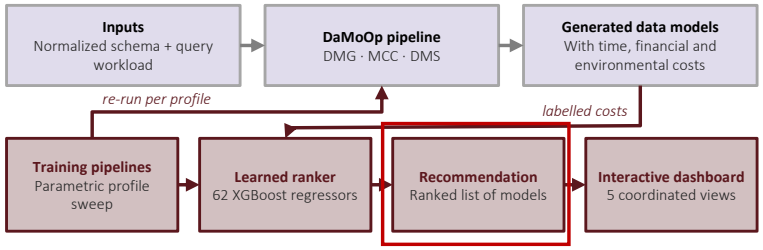
- Settings
  - Data volume (x10 to x100M DB size) *=> integrate data growth*
  - Cluster size (10 to 10k servers) *=> integrate scalability*
  - ➢ 42 settings
- **Profile** sweep over 18 schema dimensions
  - 8 attribute selectivities *=> disrupt filter queries behaviour*
  - 8 index flags *=> study impact of index choices*
  - 2 foreign keys cardinalities *=> disrupt join queries behaviour*
  - *Sampling with* Latin Hypercube Sampling => 500 profiles
- 135 models x 42 settings x 500 profiles = 2.835M labelled rows (*MCC*)
- *Coverage checked before training: every dimension spans ≥99.7% of its range*

TRAINING ML MODEL – XGBOOST

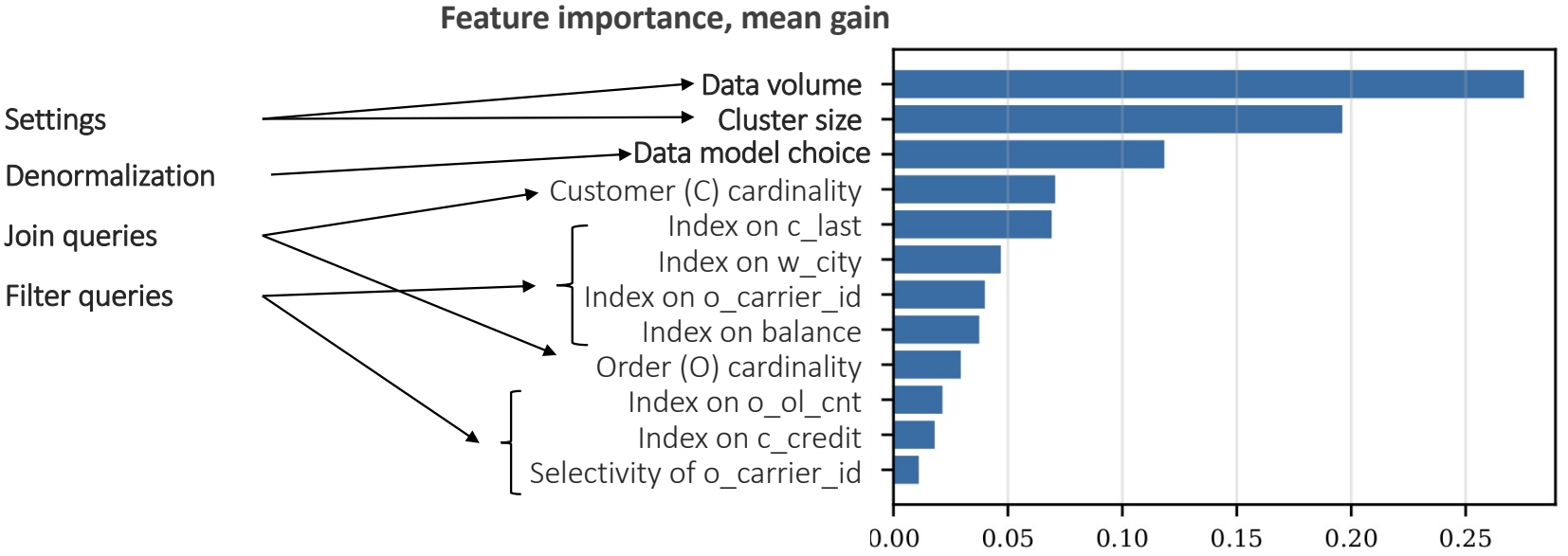


- 62 independent XGBoost regressors:
  - Settings-based: 2 query-independent cost (cluster cost)
  - Profile-based: 60 per-query unit costs (20 queries × 3 cost dimensions)
- Split **by profile**, never by row

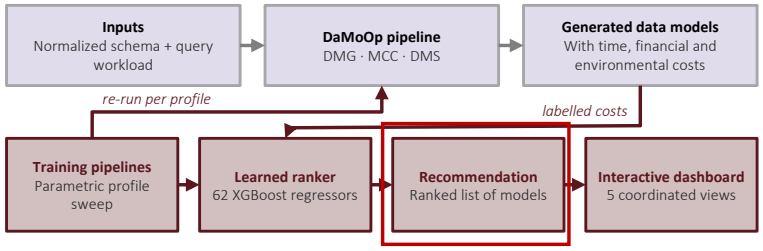
RECOMMENDATION



- For a given scenario
  - Predict each query cost based on: data model + setting + profile
  - Apply the workload scenario

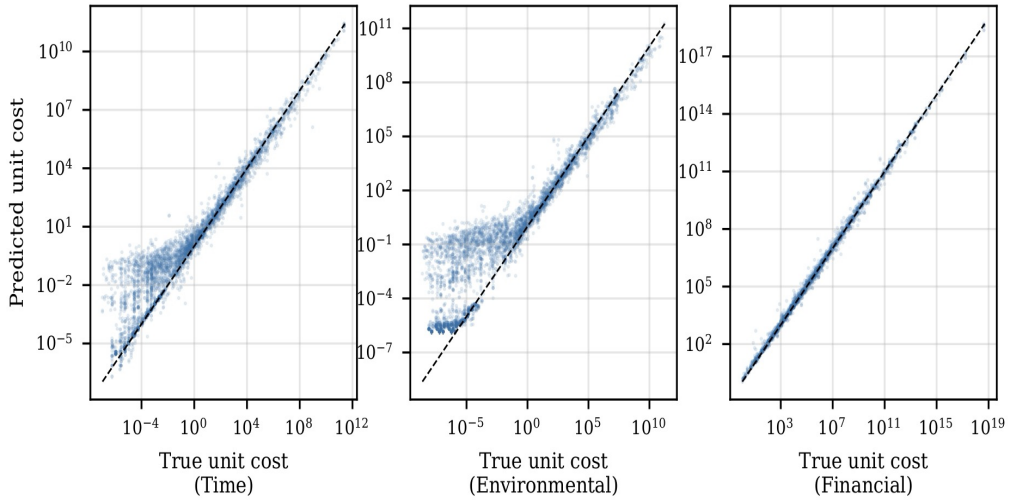


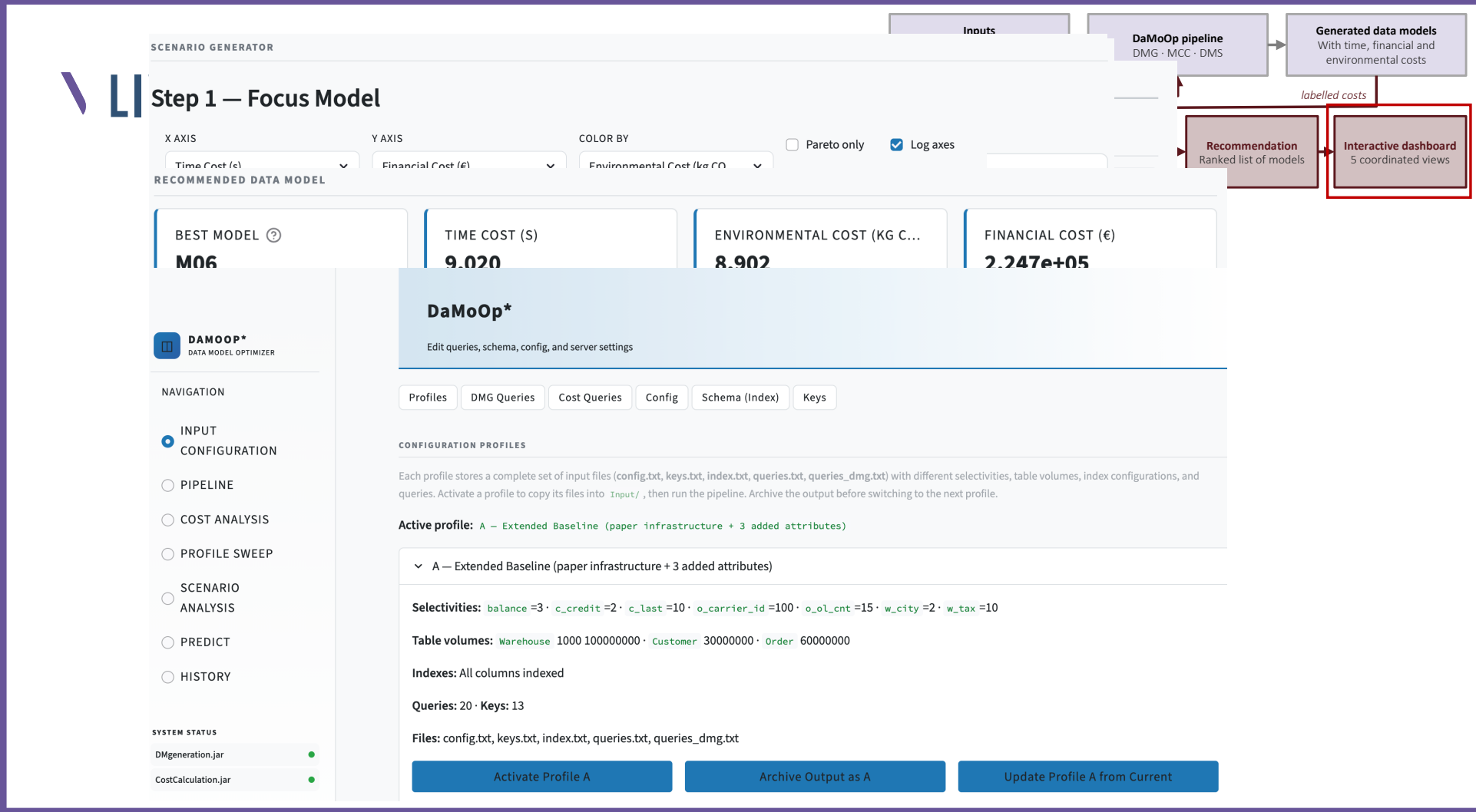
PREDICTION ACCURACY



Results on 13,440 unseen cases

61.1%	75.6%	0%	627 ms
top-1 accuracy (cost)	top-3 recall (DM order)	median regret (diff inferred cost wrt. optimal)	per recommendation vs 7.85 s by simulation




Inputs
DaMoOp pipeline
DMG · MCC · DMS
Generated data models
With time, financial and environmental costs
labelled costs
Recommendation
Ranked list of models
Interactive dashboard
5 coordinated views
SCENARIO GENERATOR
Step 1 — Focus Model
X AXIS
Y AXIS
COLOR BY
Pareto only
Log axes
RECOMMENDED DATA MODEL
BEST MODEL
M06
TIME COST (S)
9.020
ENVIRONMENTAL COST (KG C...
8.902
FINANCIAL COST (€)
2.247e+05
DAMOOP*
DATA MODEL OPTIMIZER
NAVIGATION
INPUT CONFIGURATION
PIPELINE
COST ANALYSIS
PROFILE SWEEP
SCENARIO ANALYSIS
PREDICT
HISTORY
SYSTEM STATUS
DMgeneration.jar
CostCalculation.jar
DaMoOp*
Edit queries, schema, config, and server settings
Profiles
DMG Queries
Cost Queries
Config
Schema (Index)
Keys
CONFIGURATION PROFILES
Each profile stores a complete set of input files (config.txt, keys.txt, index.txt, queries.txt, queries_dmg.txt) with different selectivities, table volumes, index configurations, and queries. Activate a profile to copy its files into Input/, then run the pipeline. Archive the output before switching to the next profile.
Active profile: A — Extended Baseline (paper infrastructure + 3 added attributes)
A — Extended Baseline (paper infrastructure + 3 added attributes)
Selectivities: balance =3 · c_credit =2 · c_last =10 · o_carrier_id =100 · o_ol_cnt =15 · w_city =2 · w_tax =10
Table volumes: Warehouse 1000 100000000 · Customer 30000000 · Order 60000000
Indexes: All columns indexed
Queries: 20 · Keys: 13
Files: config.txt, keys.txt, index.txt, queries.txt, queries_dmg.txt
Activate Profile A
Archive Output as A
Update Profile A from Current

FUTURE WORKS

Gamma objective: statistically indicated (mean–variance slope 1.99), numerically non-viable

MSLE objective: trains fine, ranks badly (326% median regret)

Unseen queries: leave-one-query-out gives **41.1% median regret** against a 10% gate set in advance

- diagnosis: coverage, not method — cross-query correlation 0.149
- queries with a structural sibling: regret 0.000, 0.001, 0.006
- queries with no sibling: fail badly
- Focus on decoders to better predict queries with no similar features
- Inject new schemas to better train the model

Integrate migration cost between models

https://github.com/leonard-de-vinci/damoop2