PERFORMANCE ENGINEERING

Lecture 6: Statistical models and Performance counters

May 9th, 2022

Ana Lucia Varbanescu
a.l.varbanescu@uva.nl
(room C3.109)

Today

- Data-centric/statistical models
  - … and examples

Remaining topics

- Bottleneck analysis (L7)
- Large scale systems modelling and analysis (L7)
- Queuing theory and its applications to PE (L8)
- Simulators (L8/L9)
- The polyhedral model and its applications to PE (L8/L9)

No lecture/lab in the last week (May 29th), focus on project and exam.

Reminder …

- **Analytical modeling**
  - **Goals**: “can I predict the performance of my system/application?”
    - System = application + data + hardware
  - **Idea**: model the operation of the system in a symbolic model and calibrate for real system

- **Data-centric/Statistical modeling**
  - **Goals**: “can I predict the performance of my application for an unseen input and/or unseen hardware?”
  - **Idea**: use past execution to build a model of the application performance

- **Simulation-based modeling (next week)**
  - **Goals**: “can I predict the performance of my system?”
    - System = application + data + hardware
  - **Idea**: simulate the operation of the system and measure different events

Statistical/data-centric/data-driven performance modeling

What's in a name?

- Data-driven/data-centric/statistical/machine-learning/… modeling: building a model based on historical data to predict future behavior.
  - Typically considered some form of *black-box modeling*.

- Requirements:
  - Historical/representative data
  - Statistical/machine-learning approaches
    - (Linear) regression
    - Decision tree(s)
    - Random forest
    - Neural networks
    - …

Why (not) statistical modeling?

Pro:

- Analytical modeling cannot capture all the complexities of the system
  - Data-dependent behavior
  - Hardware-specific behavior
- Faster than analytical modeling and simulation
  - Modulo training time …
- Could provide additional information/intuition on the inner workings of the system

Con:

- Sensitive to incorrect/insufficient data
- Sensitive to the statistical method
- Not automatically insightful

Basic idea / workflow

1. Determine **target variable**.
   - What to predict (some performance metric).
2. Collect **sufficient representative** performance data points
   - Obtain a labeled training set.
3. Determine **features (aka, predictor variables)**.
   - What does the performance depend on.
4. Choose a **modeling technique** with enough accuracy
5. Train your model
   - Random-split or cross-validation
6. Validate your model

- Challenges?

1. Target variable

- Any performance metric …
  - Supervised learning => labeled data => must have it as part of the training set!
- Examples
  - Execution time
  - Cycles
  - Instructions per cycle
  - Ranking of performance
  - Atomic contention
  - Cache miss/hit
  - …

2. Obtain/collect labeled training data

- Information about past performance
- Collection requires
  - Instrumentation
  - Execution
  - Statistics on feature and target variable
  - … and proper storage!

- How much data?
  - Depends on the method you choose/aim for
- Representative?
  - Difficult to assess …

3. Selecting features

- Naïve feature selection
  - Select all parameters that might impact performance

- Too many? Feature pruning!
  - Sensitivity analysis
    - Check with specific datasets whether some features are indeed useful
  - Variable importance (as part of different machine learning models)
  - Combined features

- Too few?
  - Collect more (detailed) statistics.

4. Choose a modeling technique

- Regression – most commonly used
  - Simple linear regression
  - General polynomial regression
- Decision trees
- Random forest
- …

- How to select?
  - Use the simplest that match the expected behavior
  - Use the one that needs the least data
  - Use the one that is less sensitive to overfitting
  - Use the one that provides added benefits (e.g., variable importance)
  - Use the most generic one
  - Use the fastest to apply

5. Train the model

- As simple as running the model training functions on the test data
  - Typically an iterative process
  - Implications on steps 4 (technique) and 3 (features)

- Be careful
  - Make sure prediction variables match method requirements
  - Use good practices for the selection of test and training data
  - Revisit predictor variables if needed
  - Store model and outcome analysis if possible
    - Model itself for post-analysis
    - Variable importance

6. Validate the model

- Apply the model on the test set.
- Report/analyze errors.
- Revisit 3,4,5 for tweaking.
- Revisit 1,2 in case accuracy is really low.

- Save final model for further use!

Example 1: Diffusion Monte Carlo (DMC)

For **generation = 1 … Nsteps** Do

For **walker = 1 … Nwalk** Do

$\mathbf{r} := \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$

For **electron i = 1 …** ***n*** Do

Set $\mathbf{r}'_i = \mathbf{r}_i + \boldsymbol{\delta}$

Compute $\mathrm{D}(\mathbf{r}') = \mathrm{D}(\mathbf{r}_1, \ldots, \mathbf{r}'_i, \ldots, \mathbf{r}_n)$

If **Accepted** Then

Update Invers $\mathrm{B}(\mathbf{r}) \leftarrow \mathrm{B}(\mathbf{r}')$

End If

End Do

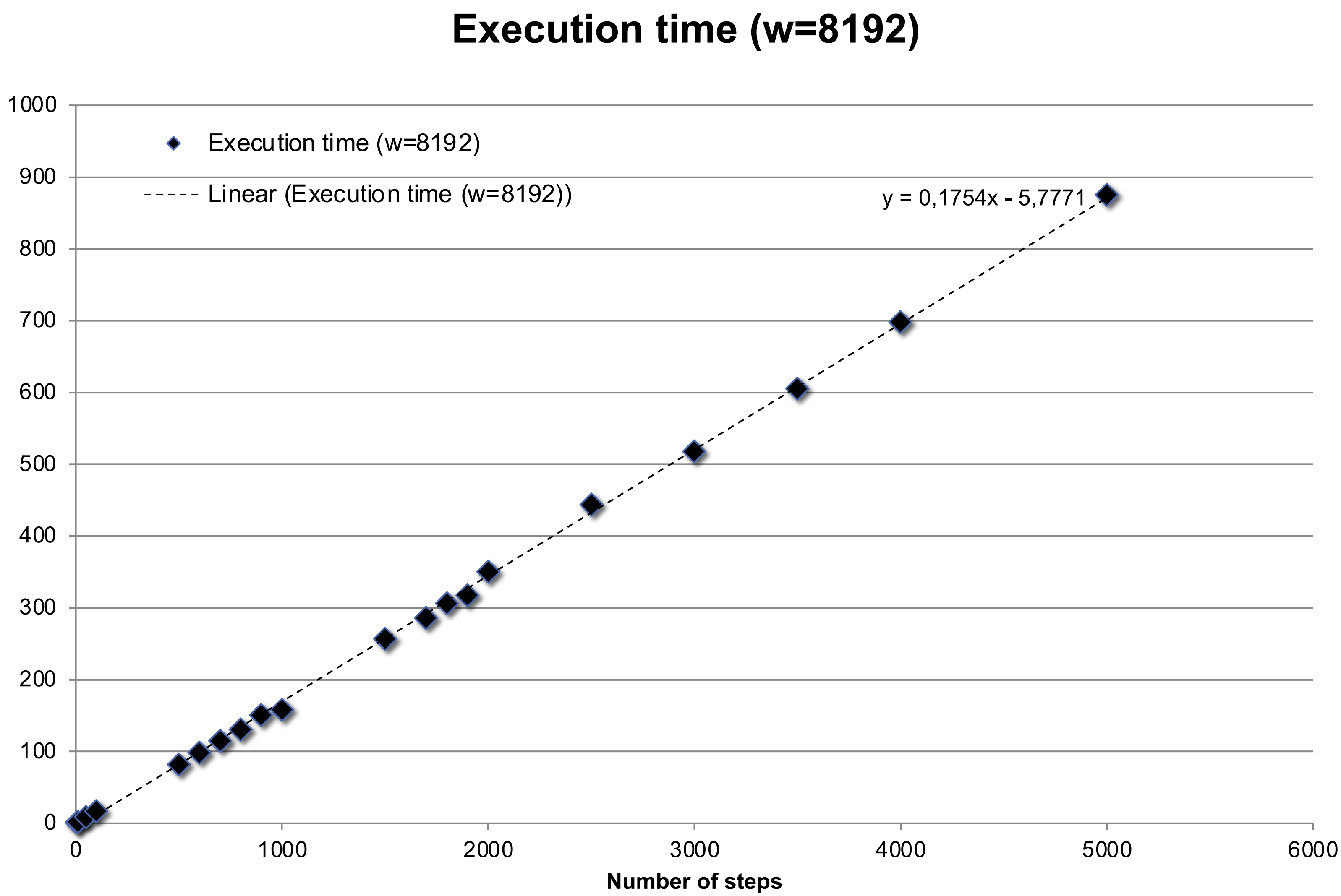
Compute Energy

End Do

Branch and Accumulate Averages

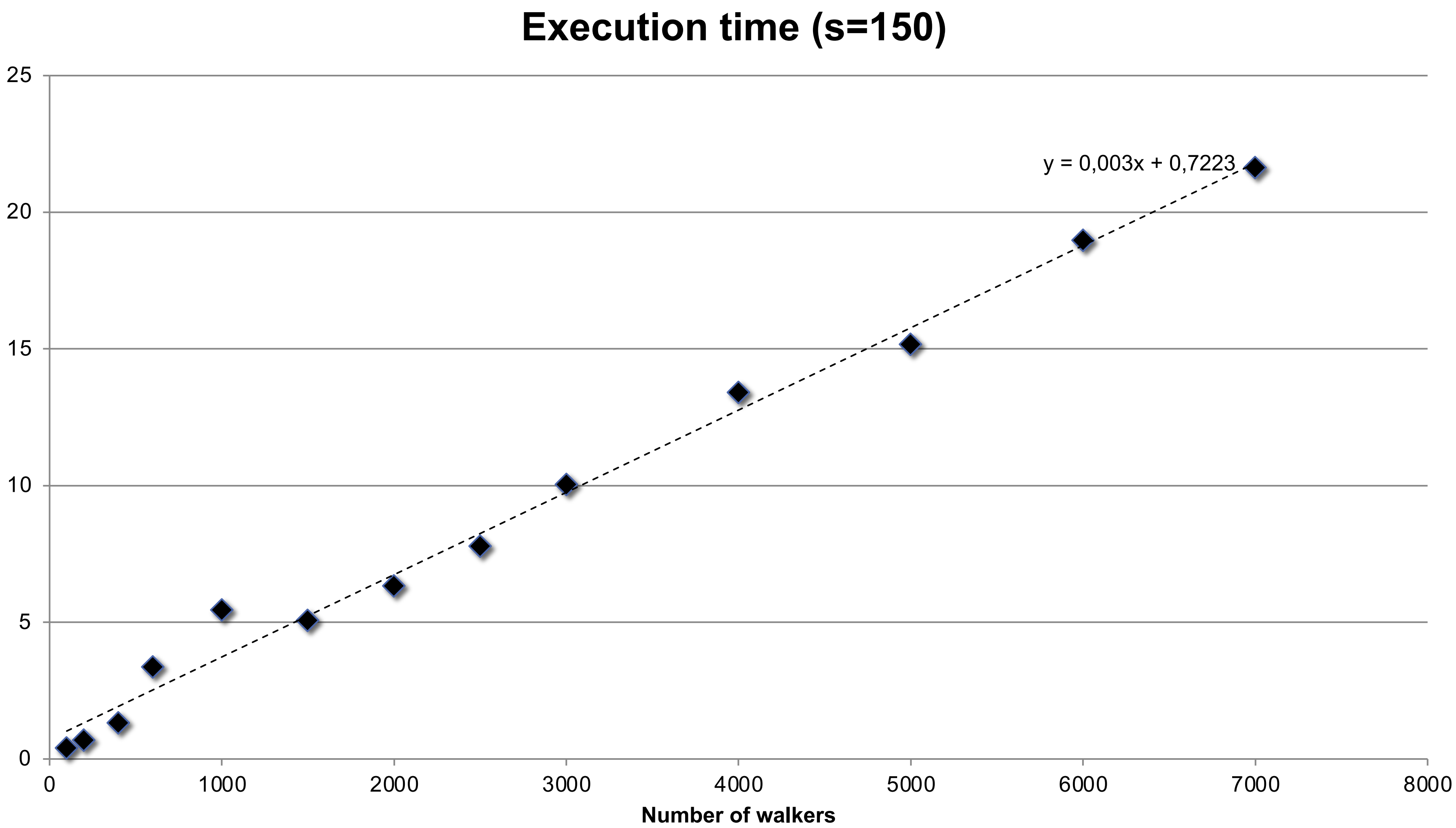
End Do

DMC: T = f(steps)



Execution time (w=8192)
1000
900
800
700
600
500
400
300
200
100
0
Execution time (w=8192)
Linear (Execution time (w=8192))
y = 0,1754x - 5,7771
0
1000
2000
3000
4000
5000
6000
Number of steps

DMC: T = f(walkers)

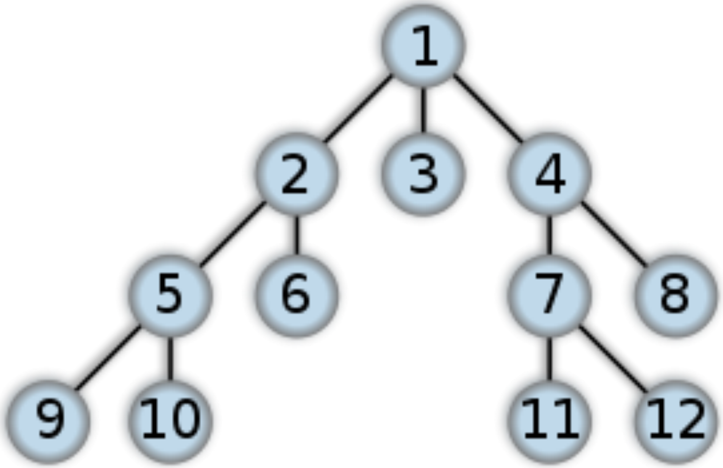


DMC: conclusions

- Execution time scales linearly with the number of steps and the number of walkers.
- Minor errors at small simulation sizes
  - To be investigated in case small sizes are relevant

- Lessons learned:
  - Linear regression was sufficient
  - Target (execution time) and predictor variables (input size) obvious

Example 2: BFS



- Graph traversal algorithm
- Visits nodes in levels
- Many parallel algorithms exist
  - Some are platform specific!

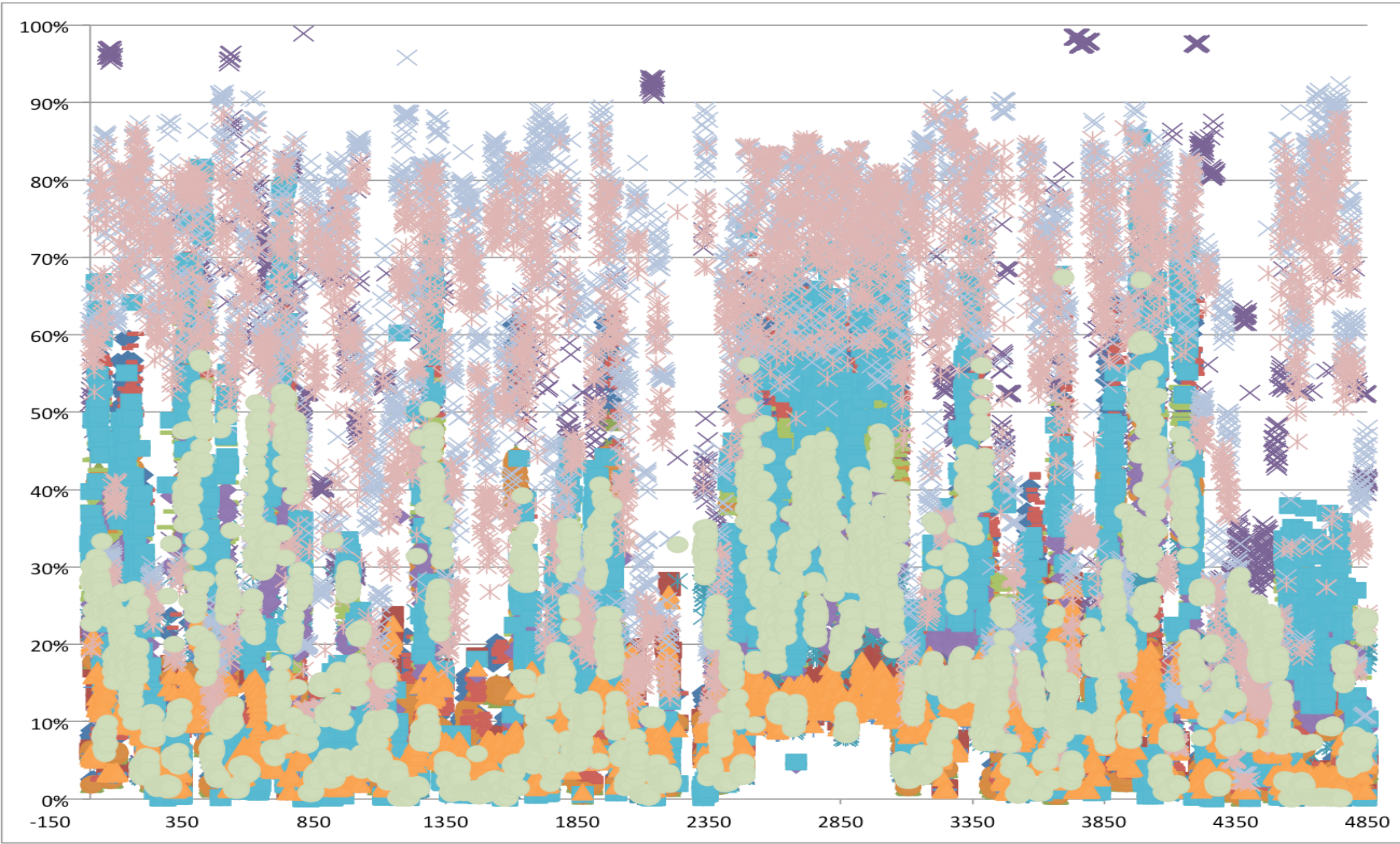
- Performance is known to be graph dependent
  - Topology impacts execution time
  - Start node impacts execution time

- Challenges?
  - How to measure topology
  - How to get sufficient representative training data

BFS: experimental setup

- GPU platforms
- 15 different versions of BFS
  - Some are variants, some are truly different
  - Differ in amount of parallelism (vertex-based vs. edge-based)
  - Differ in synchronization mechanims (atomics vs. lock-free)
- ~200 graphs from KONECT
  - Different starting nodes
- Collected statistics
  - Execution time (total)
  - Execution time (per level)
  - Level & next level size
  - *Performance counters* (added later)*

Normalized performance



BFS: attempt 0

- Analytical modeling

- How would you build that?

BFS: attempt 1

- Target variable: execution time per algorithm
- Feature variables: graph properties
  - Number of edges
  - Number of vertices
  - Diameter
  - …

- Random forest => failure
- Decision tree => failure
- (many other failures)

What can we do?

BFS: attempt 2

- Target variable: execution time per level
- Feature variables: graph properties
  - Number of edges
  - Number of vertices
  - Level and frontier size
  - Percentage of graph visited

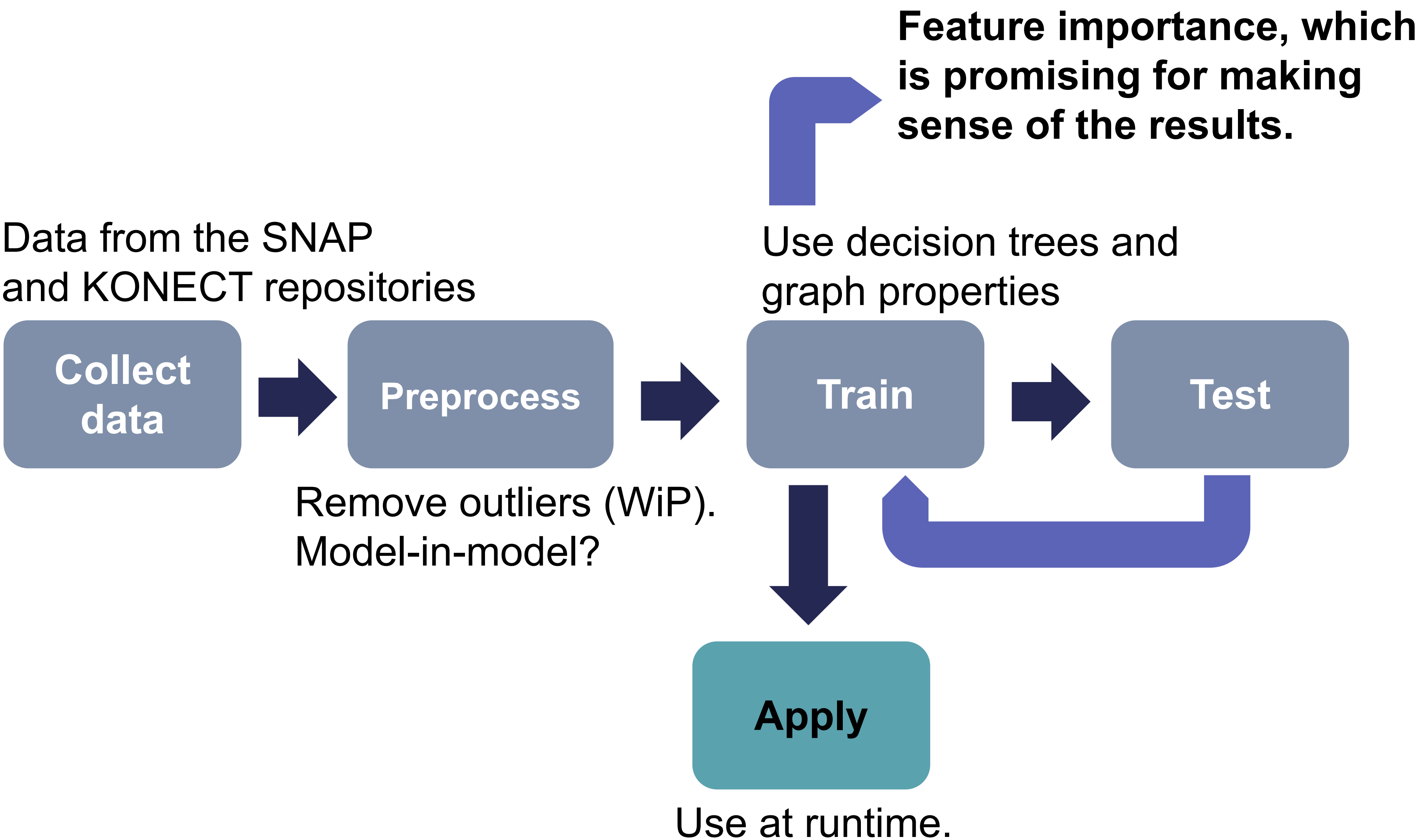
- Analytical model: failure due to atomics impact
- Regression: not accurate enough
- Random forest:
  - High accuracy
  - High prediction cost

BFS: attempt 3

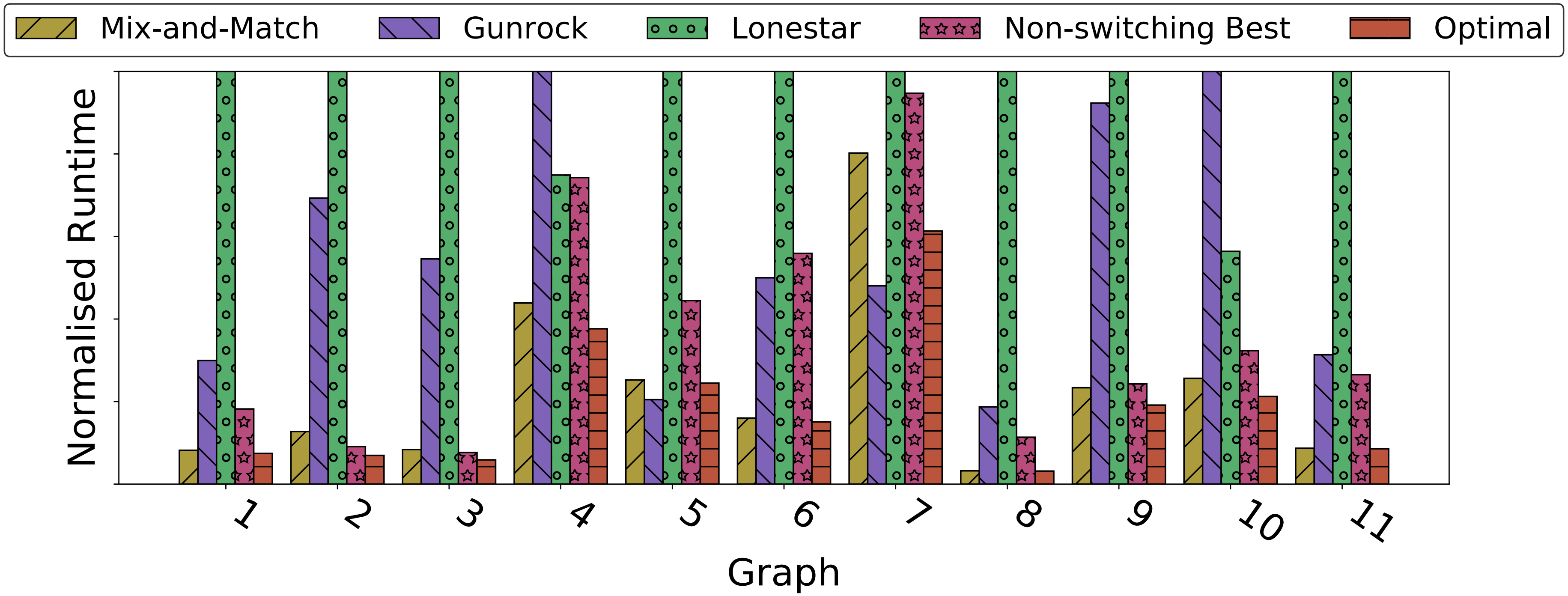
- Target variable: ranking of best algorithm
- Feature variables: graph properties
  - Number of edges
  - Number of vertices
  - Level and frontier size
  - Percentage of graph visited

- Random forest:
  - High accuracy
  - High prediction cost
- Decision tree: worked!

BFS: Current workflow



Does it really work?

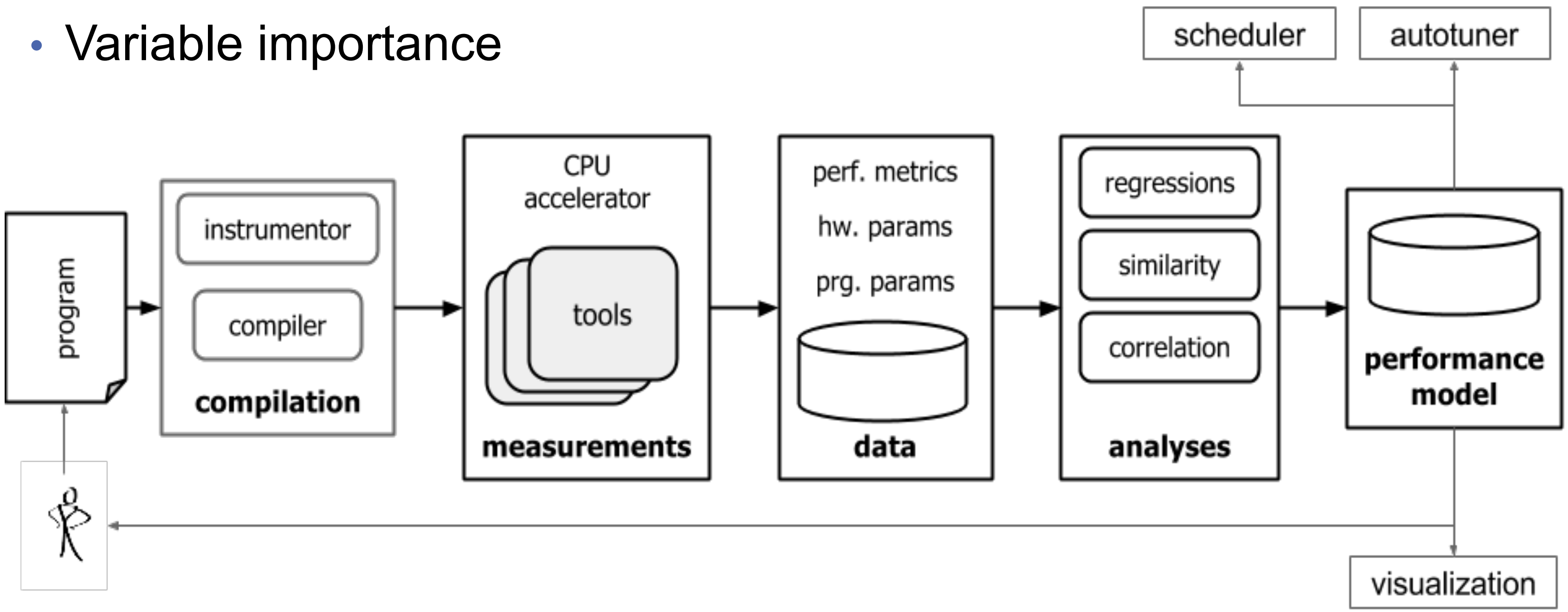


- Runtime switching is possible, (currently) with some memory overhead
- We are faster than the state-of-the art, on average, by 3x

Mix-and-match uses performance variability to build the best BFS per graph!

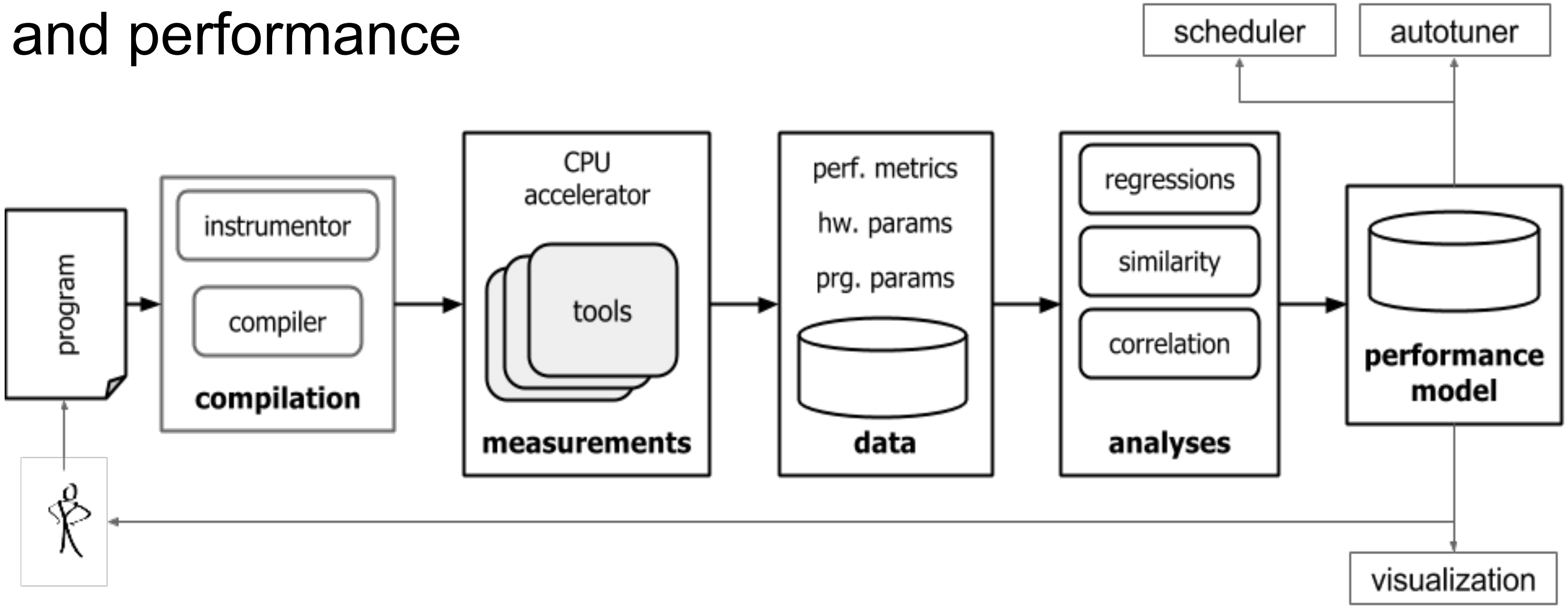
Example 3: the BlackForest framework

- Automate the process of statistical modeling
- Target variable: execution time
- Predictor variables: performance counters
- Outcome:
  - Prediction model
  - Variable importance



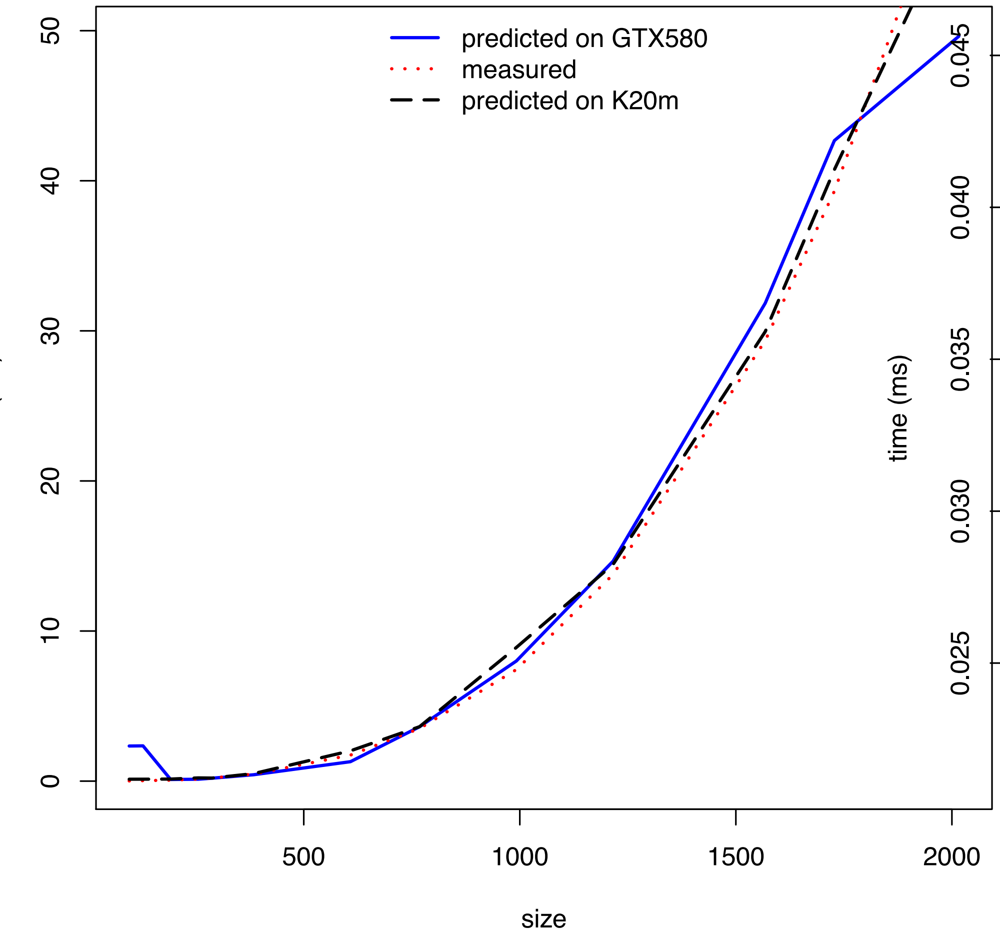
Example 3: the BlackForest Framework

- Compilation: optional, scope limitation by instrumentation
- Measurements: performance data collection via hardware performance counters
- Data: repository, file system, database
- Analyses: reveal correlation between counter behavior and performance

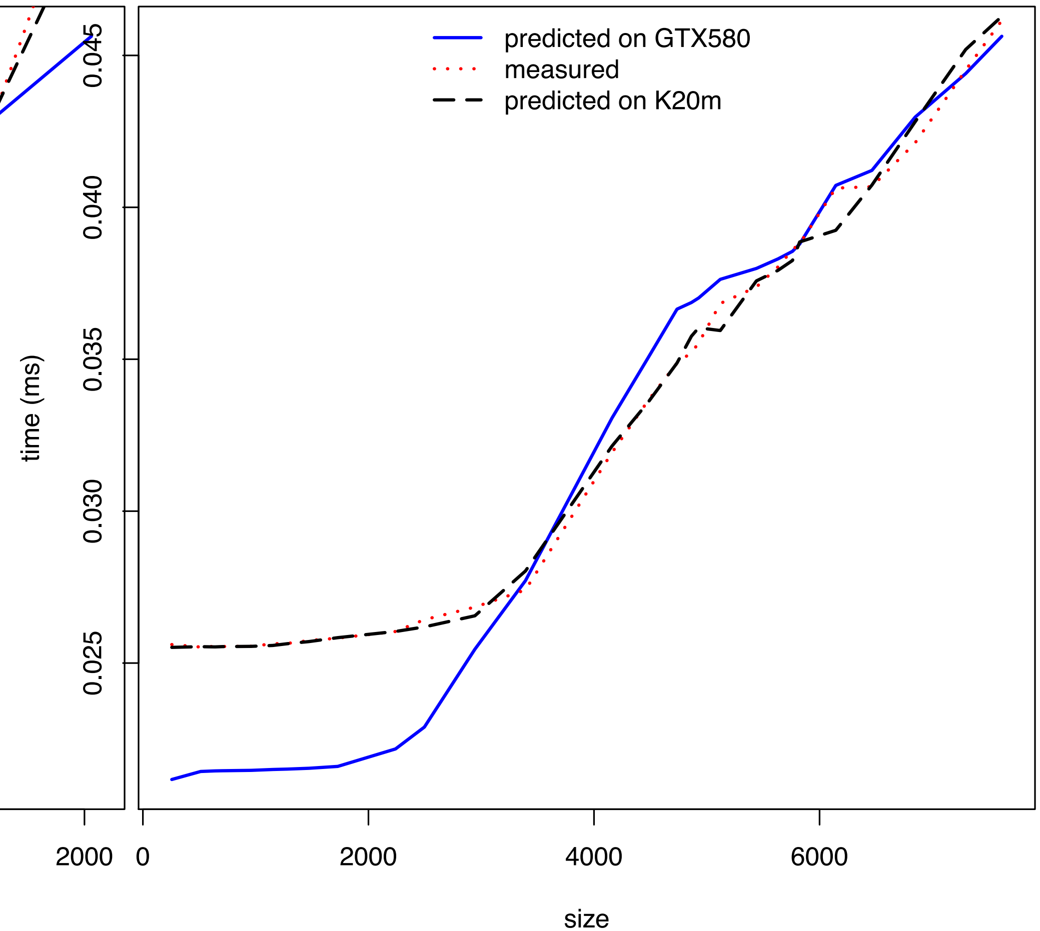


BlackForest: Results*

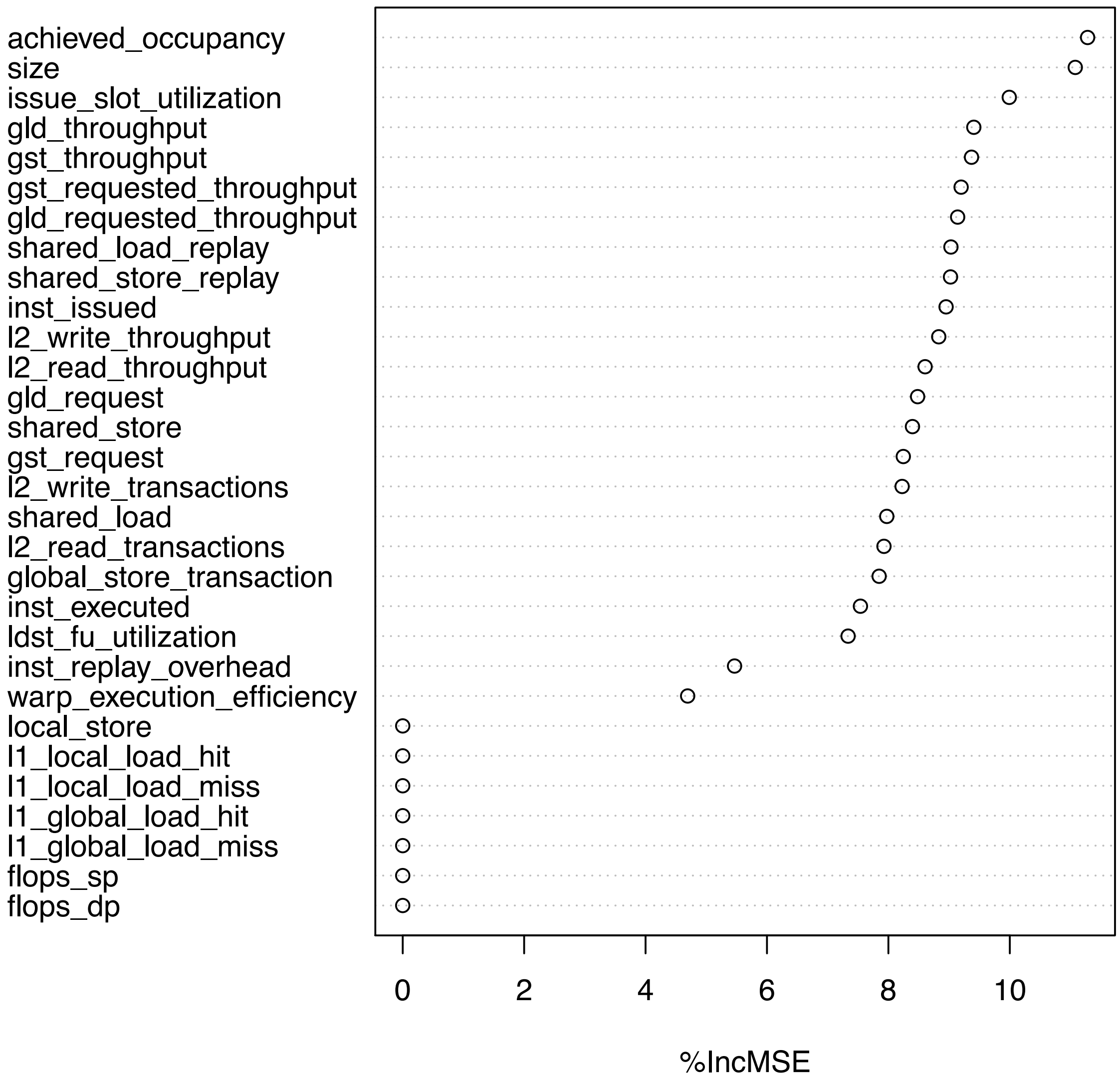
Matrix multiply



Needleman-Wunsch



BlackForest: Variable importance



Statistical Modeling recap

- Statistical approaches
  - Black-box modeling
  - Useful when "opening" the system is not feasible
    - Time or complexity issues
- Requirements
  - Labeled training data
  - Tools for training and building models
- Challenges
  - Getting/collecting representative data
  - Variable selection
  - Validation & testing
- Disadvantages
  - Low insight
  - Expensive re-training

More data for your model …

What if …

- … the data is too coarse-grain?
- … we can't understand the dependency of the target variable on the input data?
- … we can't properly calibrate?
- … we have data-dependent behavior?

We can get more insight into the performance behavior/causes for it using more detailed analysis of past execution by **monitoring hardware events**.