# Performance Models to support HPC Co-Scheduling

***Athanasios Tsoukleidis-Karydakis**, Efstratios Karapanagiotis, Nikolaos Triantafyllis, Nectarios Koziris and Georgios Goumas*

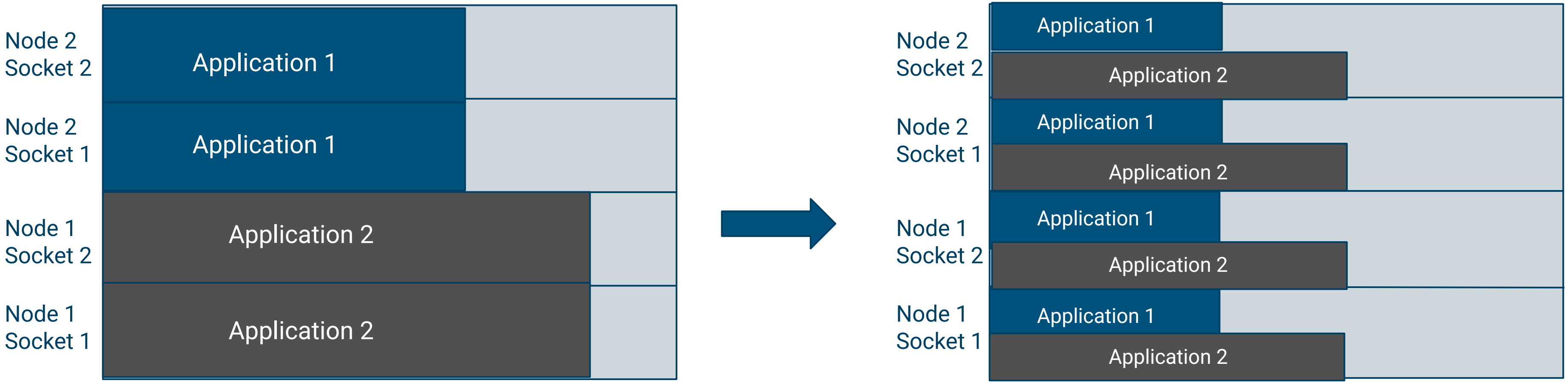
National Technical University of Athens
CSLab
IPDPS
2025 • Milano, Italy
JSSPP

# What is Co-Scheduling?

**Problem :** System Throughput

**Proposed Solution:** Co-Scheduling

- An application typically spreads across twice the number of nodes compared to the classical scheduling case, but occupies half the number of cores on each node. Thus, two jobs can be allocated to the same node at the same time

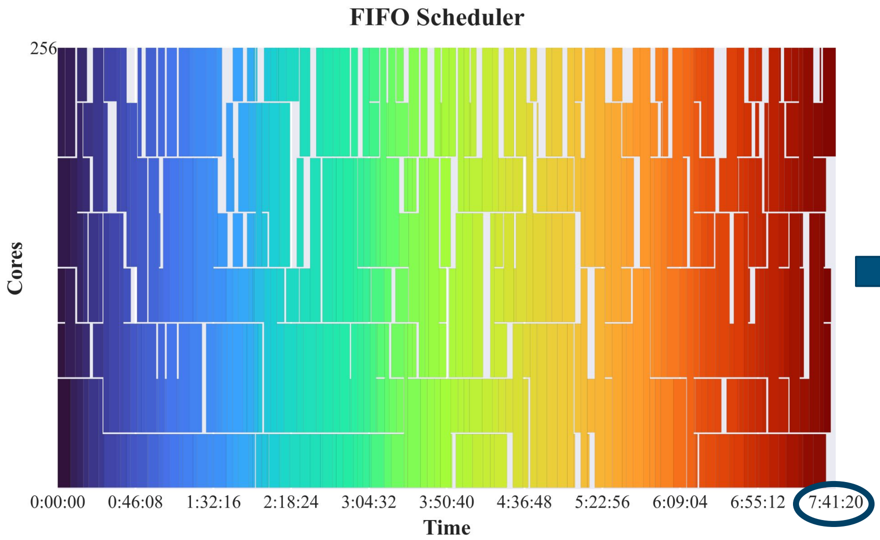


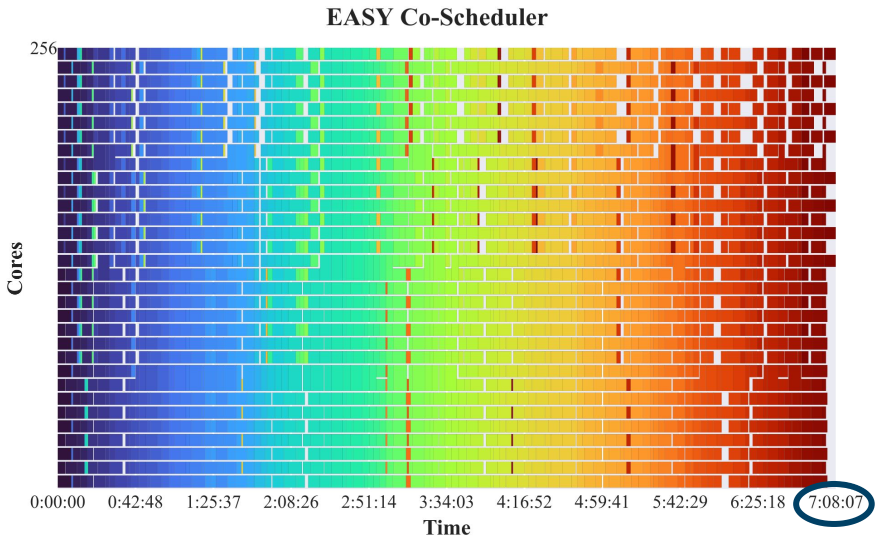
# What is Co-Scheduling?

**Problem :** System Throughput

**Proposed Solution:** Co-Scheduling

- An application typically spreads across twice the number of nodes compared to the classical scheduling case, but occupies half the number of cores on each node. Thus, two jobs can be allocated to the same node at the same time



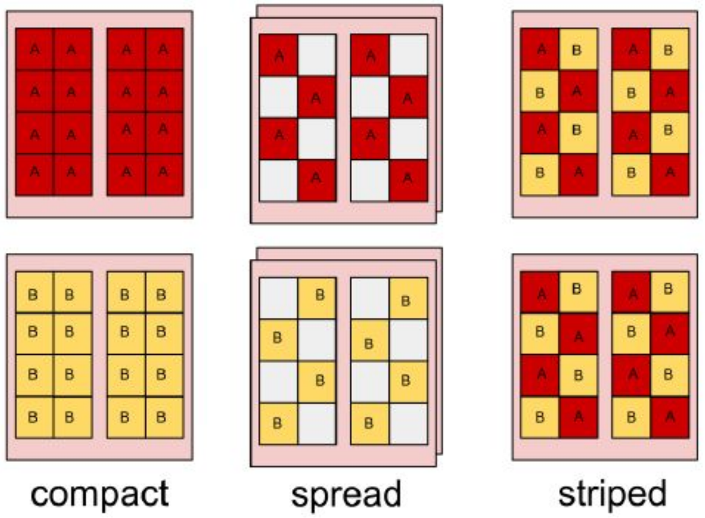




decreased makespan

# What is Co-Scheduling?

**Compact mode:**
- Traditional scheduling
- Exclusive node allocation

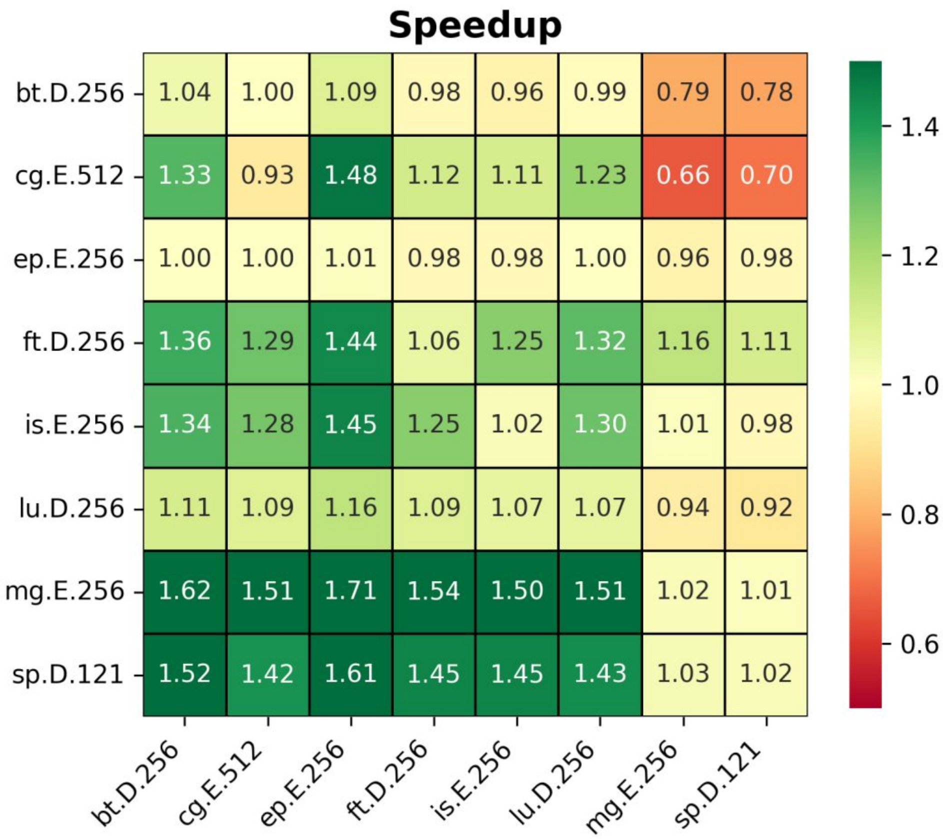
**Spread mode:**
- Urgent computing
- Double node spread
- Half-node occupancy
- Remaining half left idle




**Co-location (striped) mode:**
- Spread mode execution
- Idle halves occupied by other jobs

resource contention in shared node resources can introduce performance degradation, leading to job slowdowns and counteracting these benefits

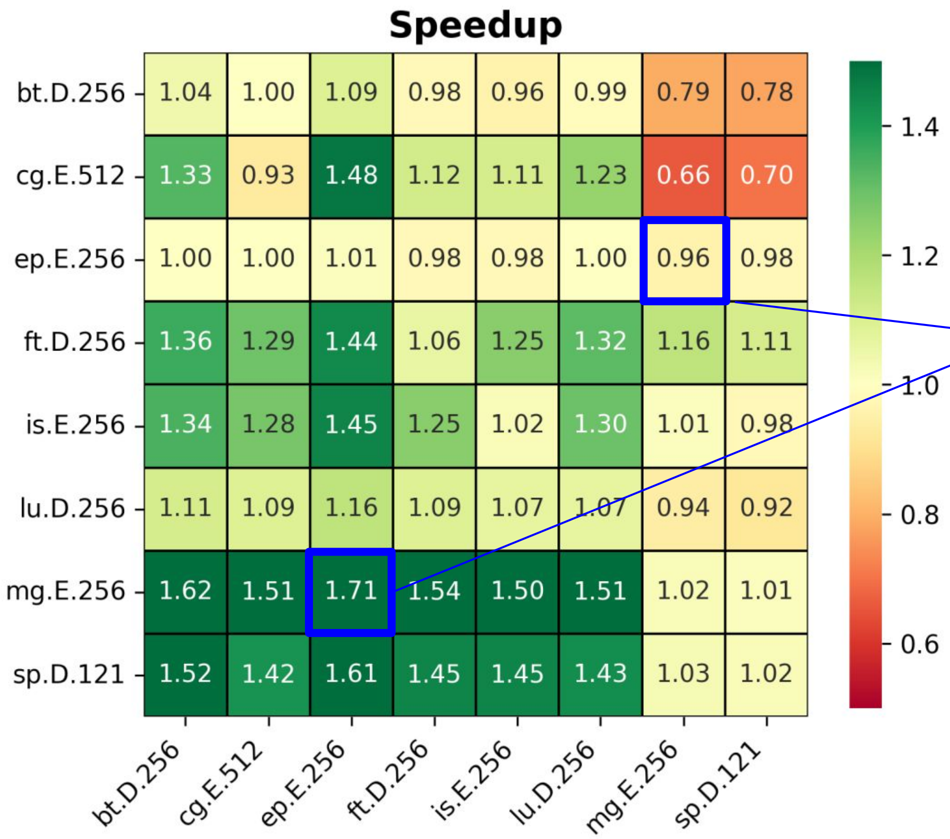
# Heatmap of application co-locations




$$\text{JobSpeedup} = \frac{\text{Compact Execution Time}}{\text{Co-located Execution Time}}$$

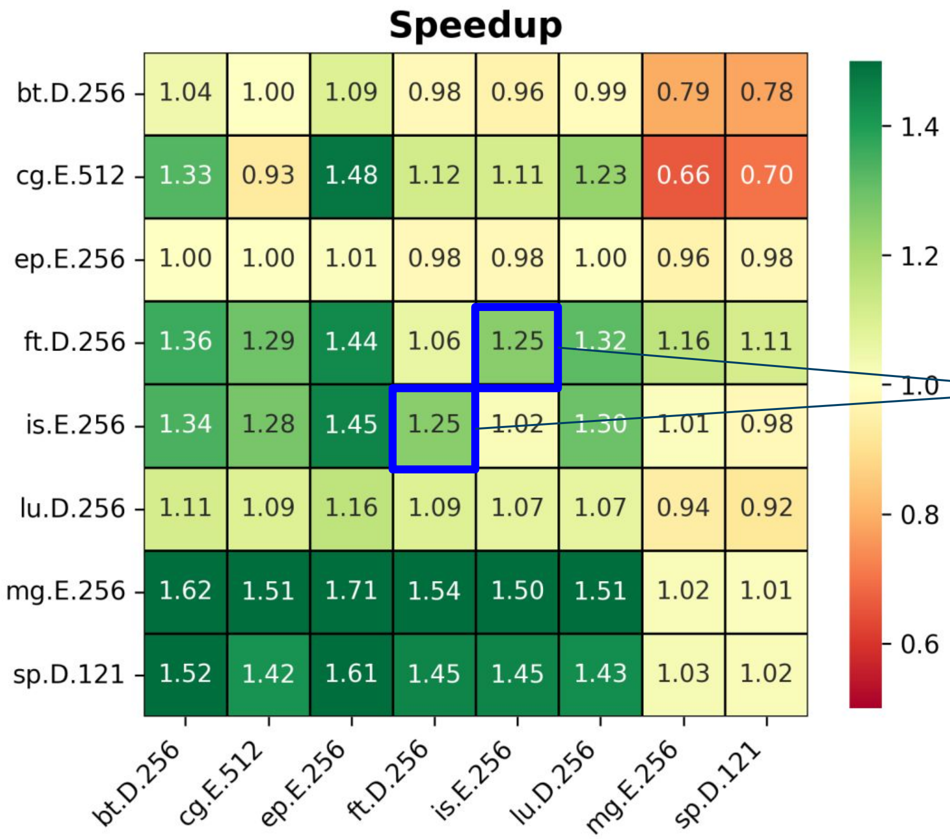
- pairwise execution of NAS benchmarks
- Classes : D, E
- Various number of processes

# Heatmap of application co-locations




one application benefits tremendously from co-location and the other stays stationary

# Heatmap of application co-locations

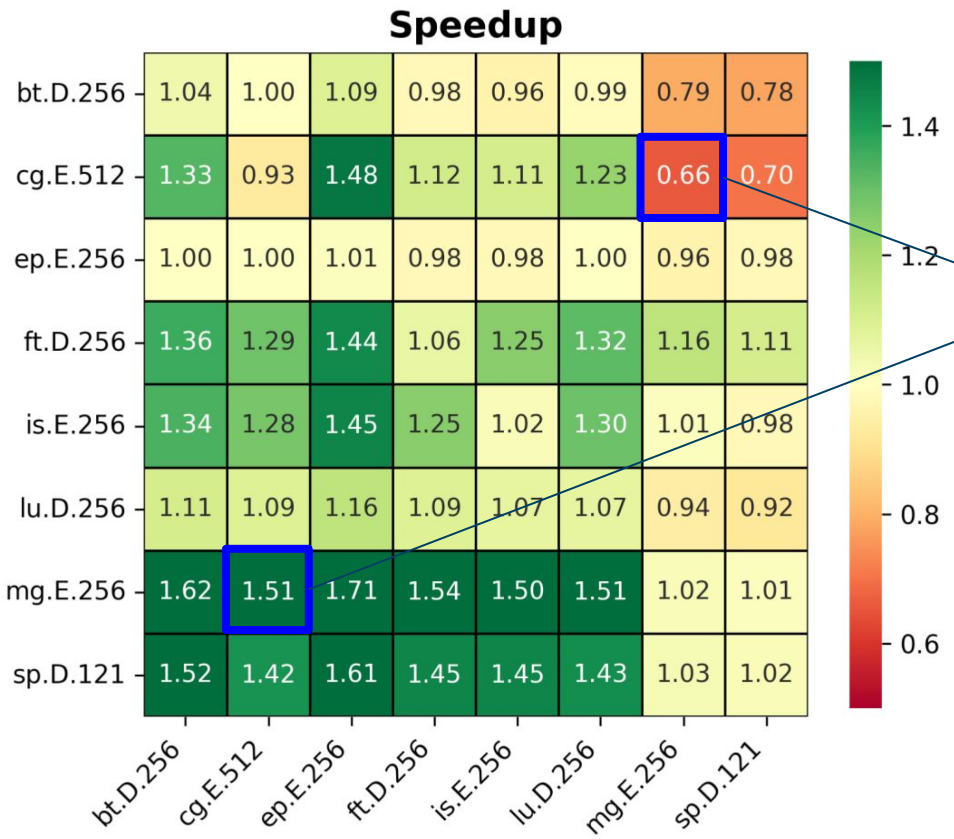



both applications exhibit speedups

# Heatmap of application co-locations

need for insights into the applications in the waiting queue to make good pairs

Application Models




one application suffers unacceptable slowdown

# Goal and Infrastructure

🎯 **Goal of this work :** Develop models that predict an application's performance improvement or degradation when co-located with other applications

- **ARIS supercomputer** (operated by GRNET, Athens, Greece)
- We utilized the **perf** Linux tool and **mpiP**
- Collected Metrics:
  - FLOPS
  - Memory Bandwidth
  - LLC hits/misses
  - MPI stats
    - MPI time
    - MPI time per directive
- **NAS Parallel Benchmarks** (NPB) and **SPEChpc 2021** benchmarks

# Model Requirements

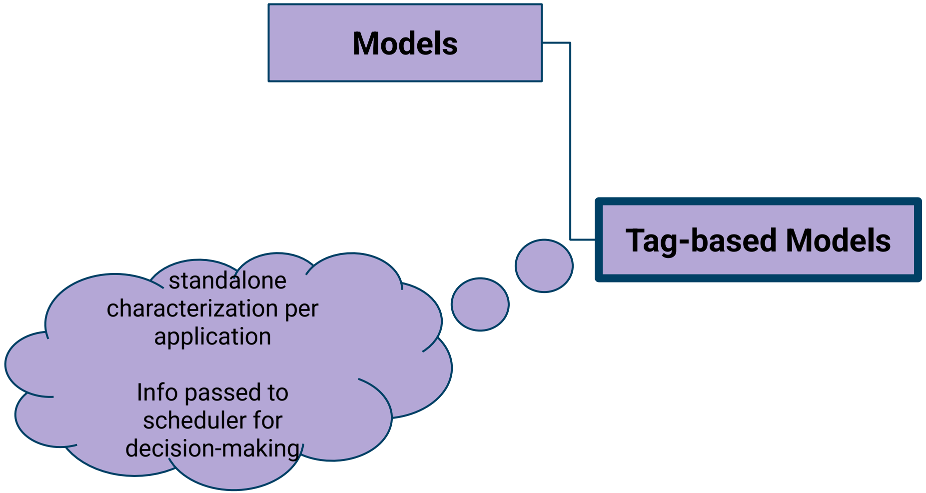
## Need to have:

- High accuracy and precision
- Minimal data use
- Lightweight profiling
  - few hardware counters
  - few application runs
- Fast decision

## Nice to have:

- Interpretability
- Generalizable to other machines

# Taxonomy of Application models



# Taxonomy of Application models

**Models**

- Tag-based Models
  - Resource-centric

standalone characterization per application

Info passed to scheduler for decision-making

**characterize applications according to their resource consumption patterns**

# Taxonomy of Application models

**Models**

Tag-based Models

**Co-location-centric**

Resource-centric

standalone characterization per application

Info passed to scheduler for decision-making

**each application is labeled as either co-location friendly or co-location unfriendly**

# Taxonomy of Application models

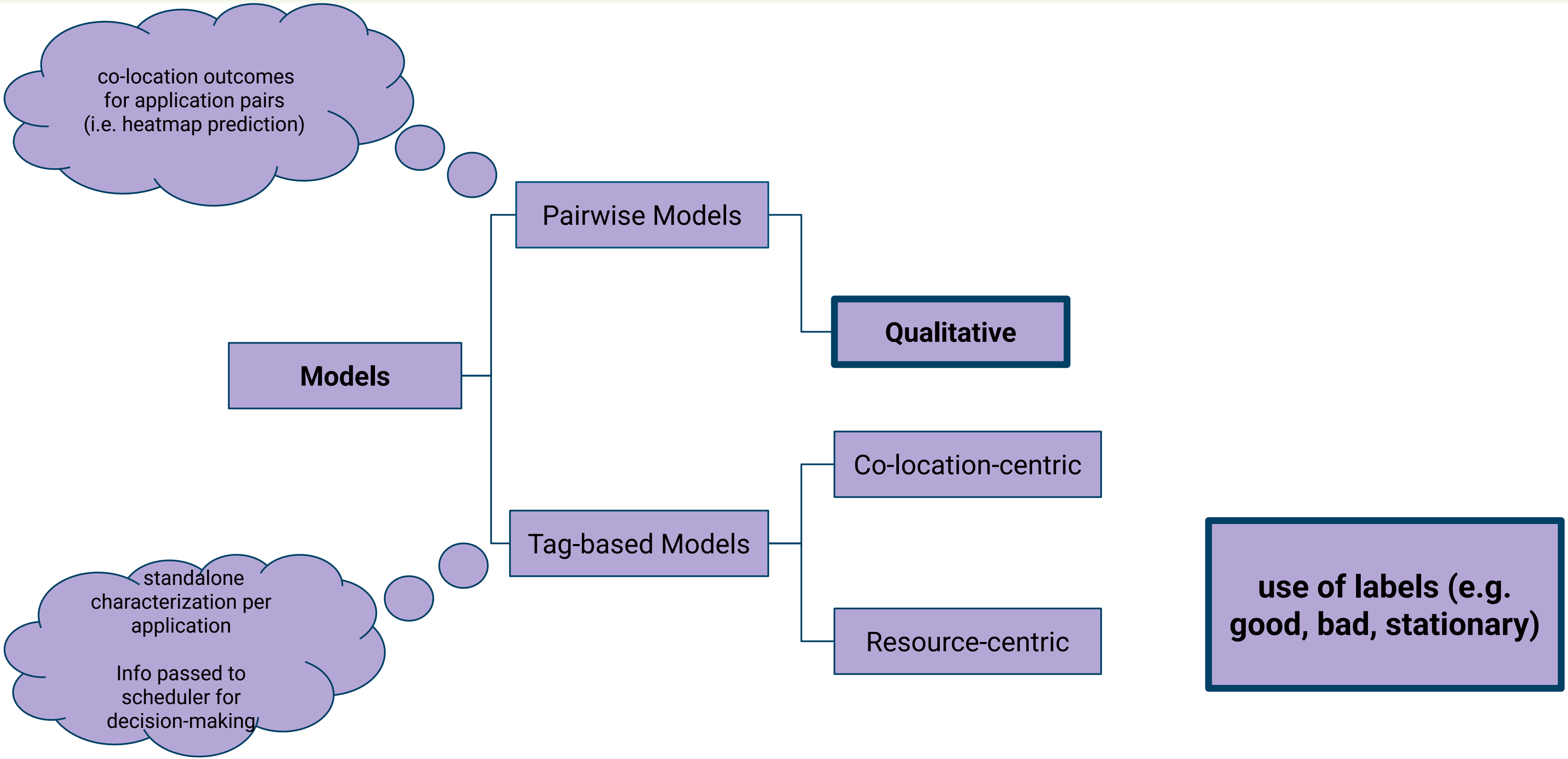
- **Models**
  - **Pairwise Models**
  - Tag-based Models
    - Co-location-centric
    - Resource-centric

co-location outcomes
for application pairs
(i.e. heatmap prediction)

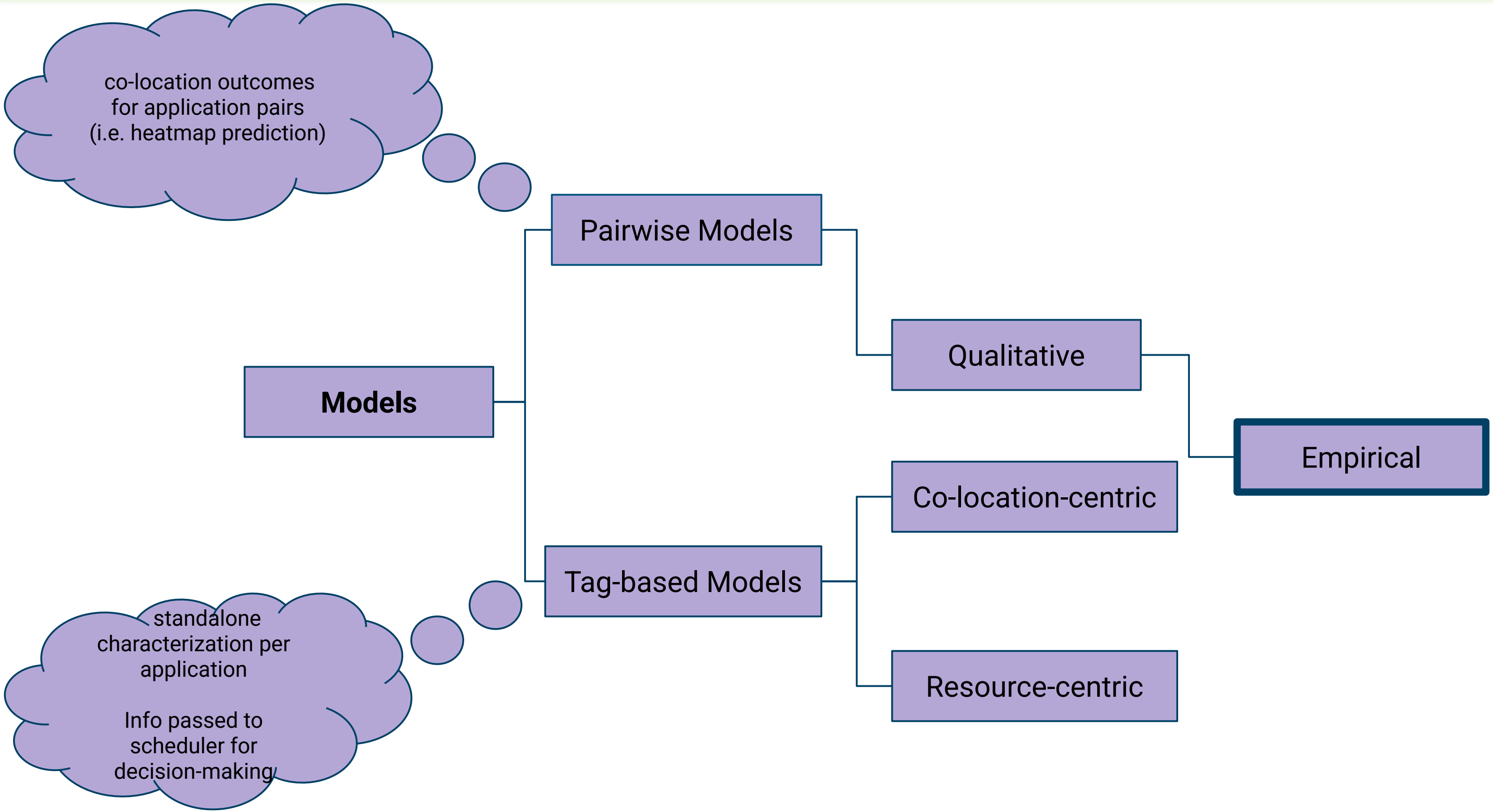
standalone
characterization per
application

Info passed to
scheduler for
decision-making

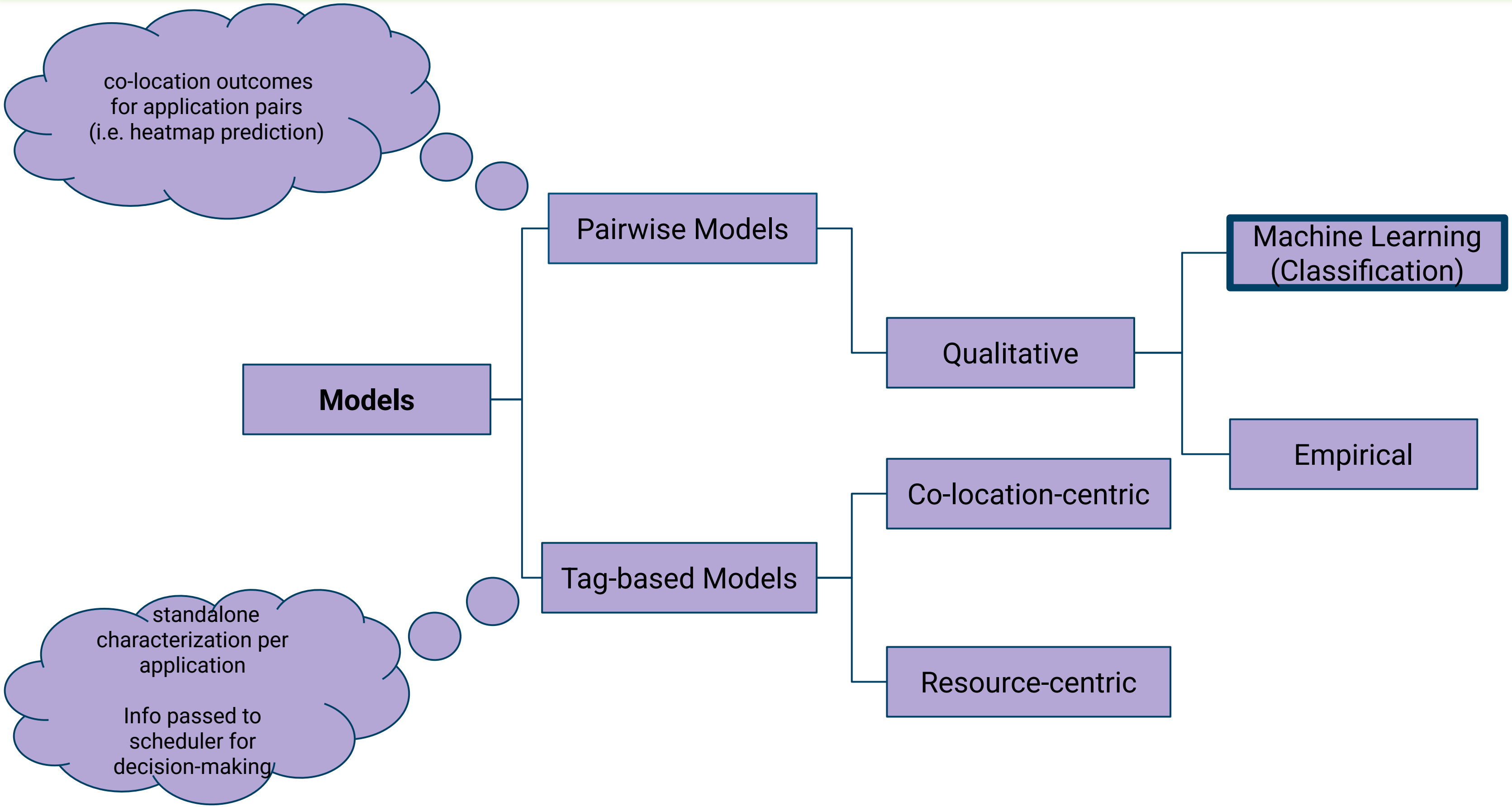
# Taxonomy of Application models



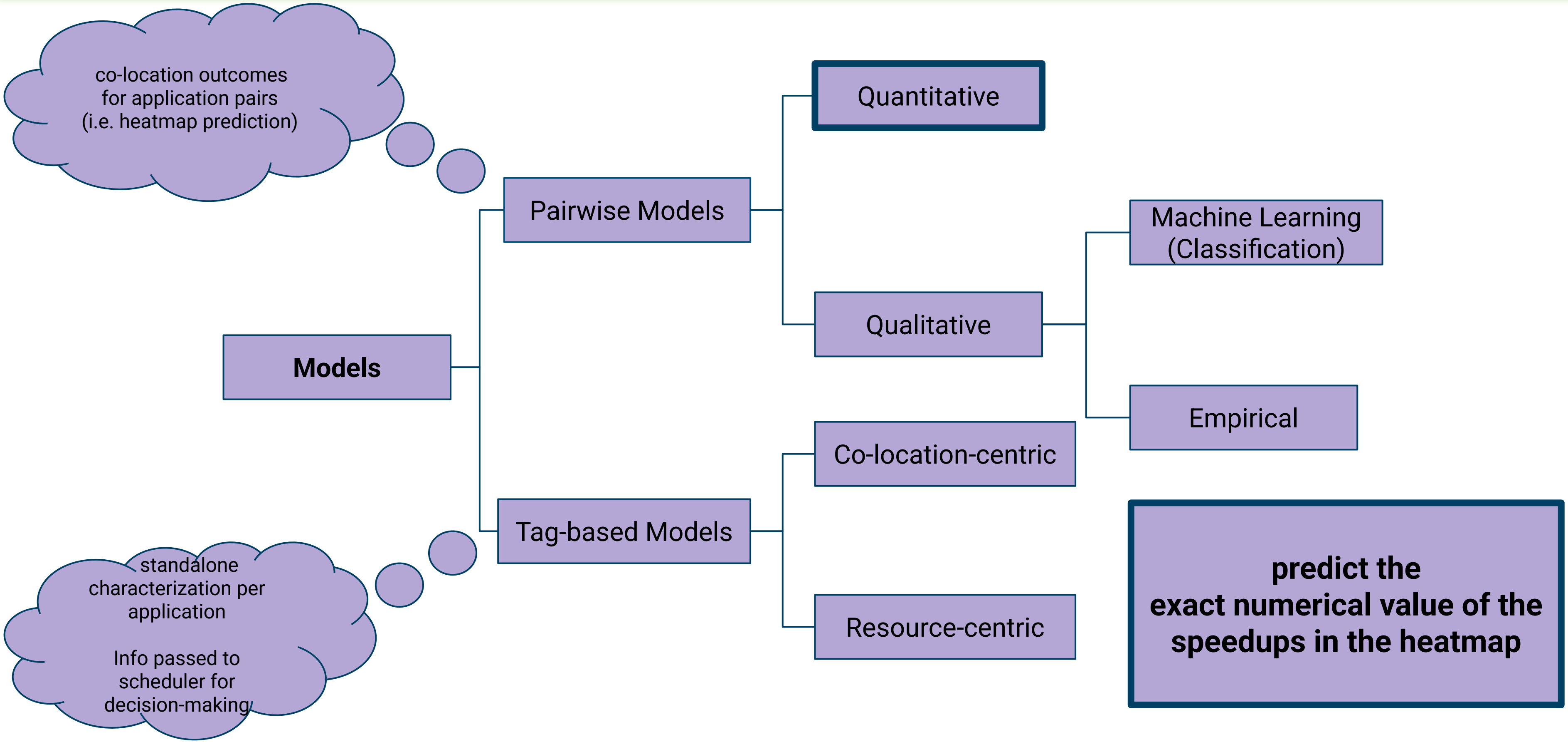
# Taxonomy of Application models



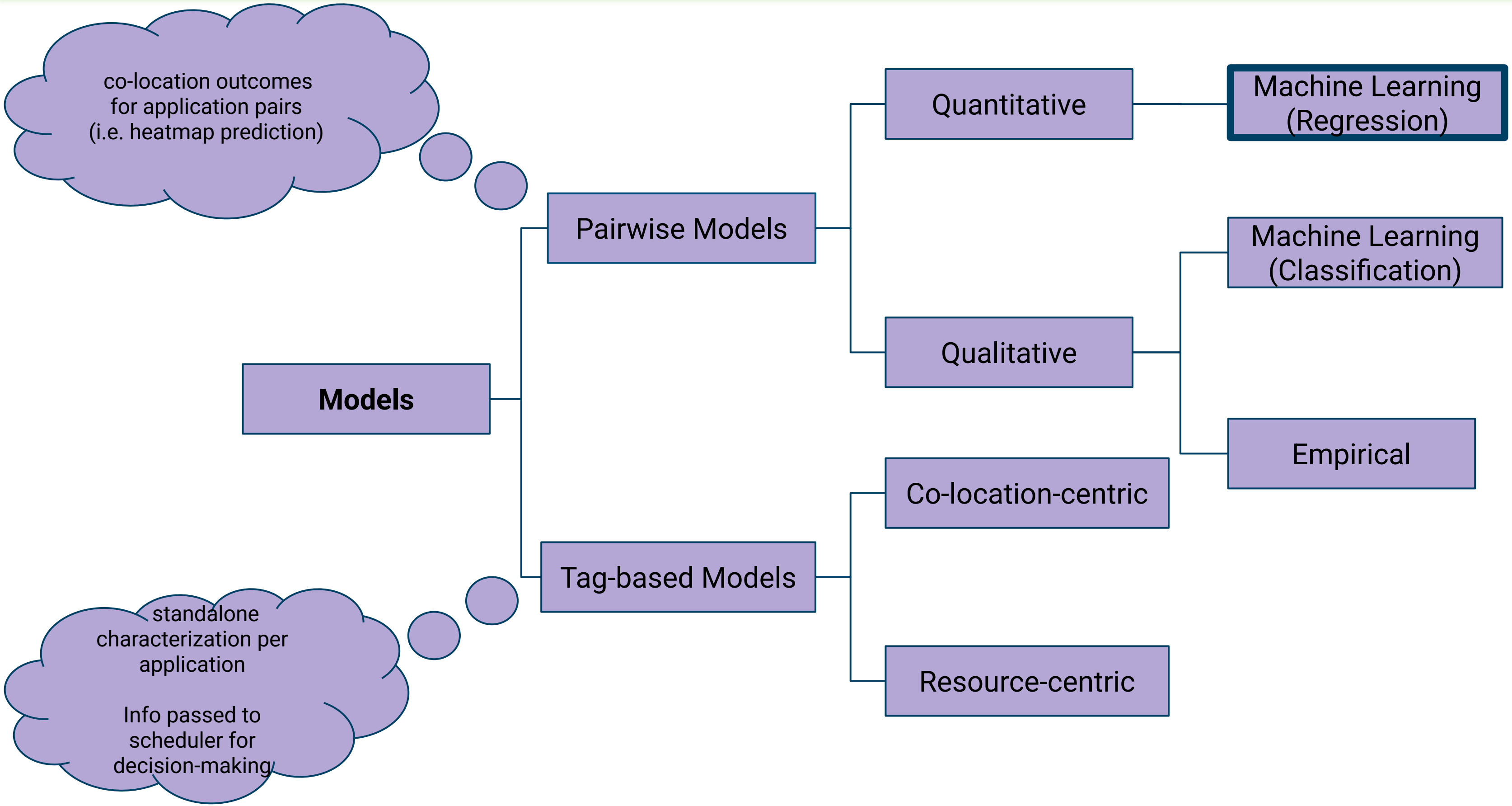
# Taxonomy of Application models



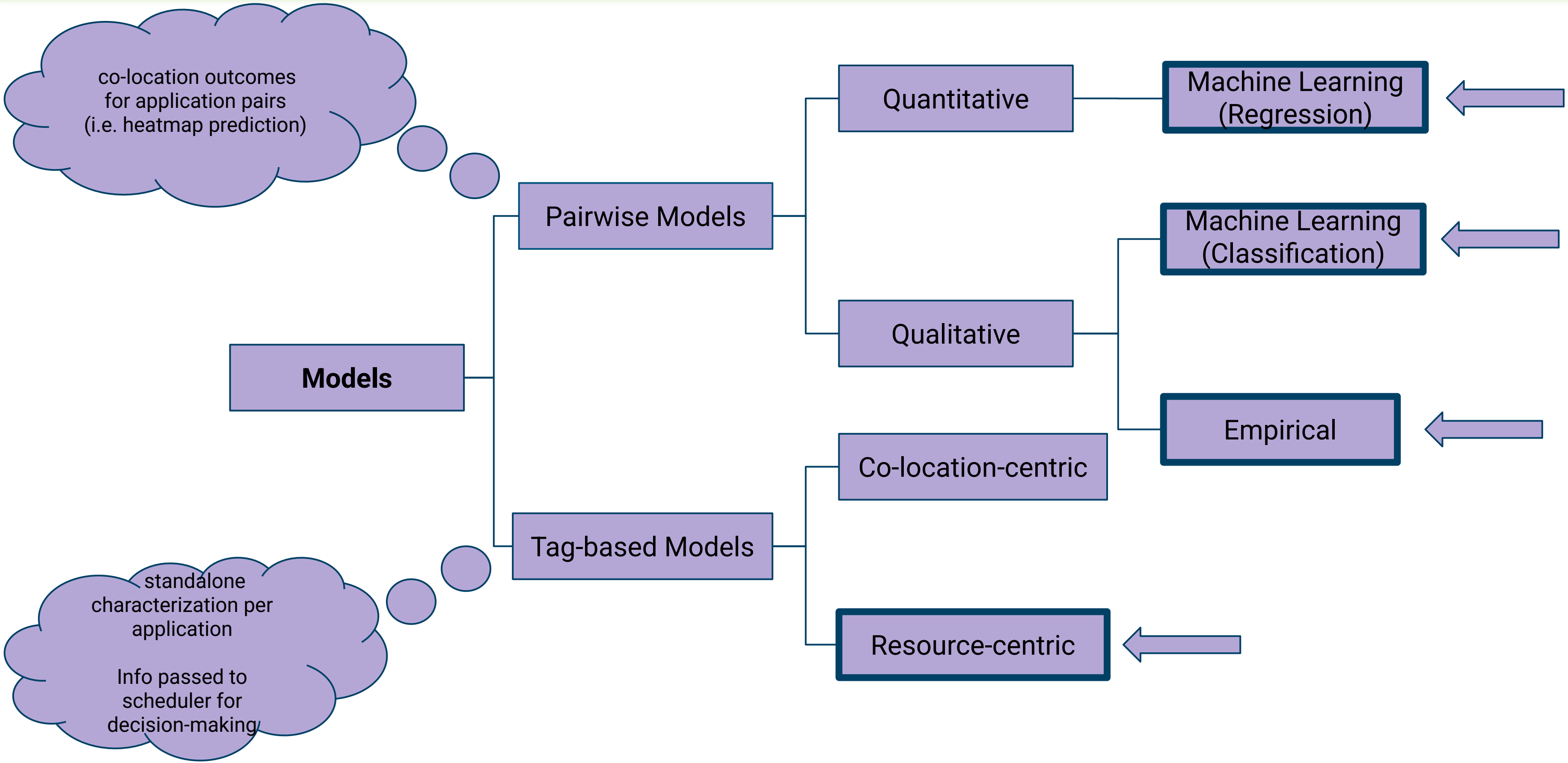
# Taxonomy of Application models



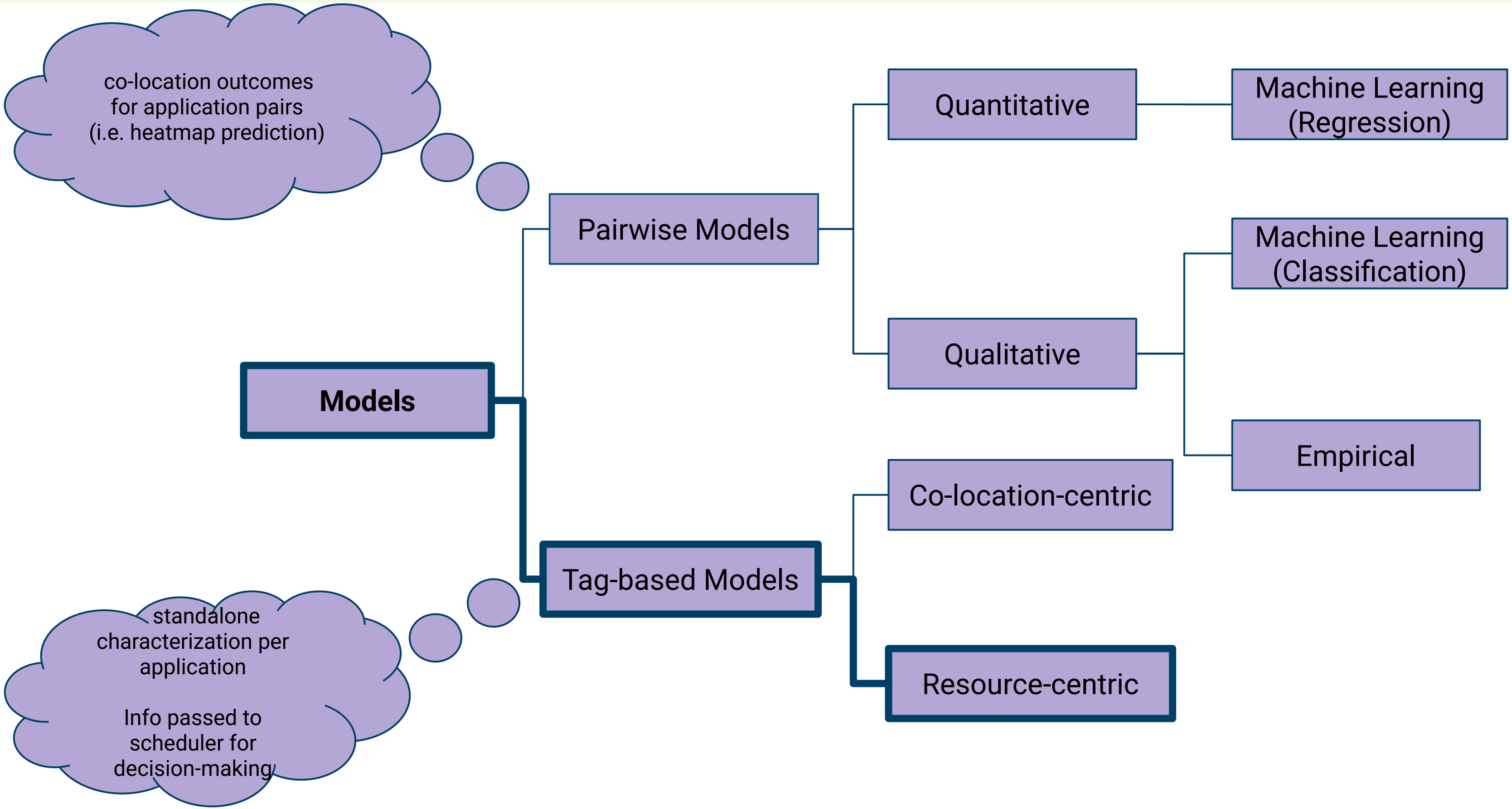
# Taxonomy of Application models



# Taxonomy of Application models



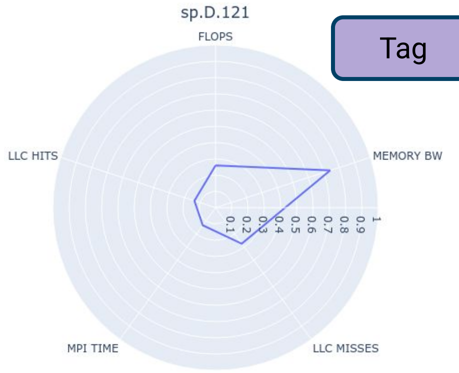
# Taxonomy of Application models



# Resource-centric tag-based models

**Goal :** Retrieve a <u>standalone resource consumption-based characterization</u> for each application in the queue

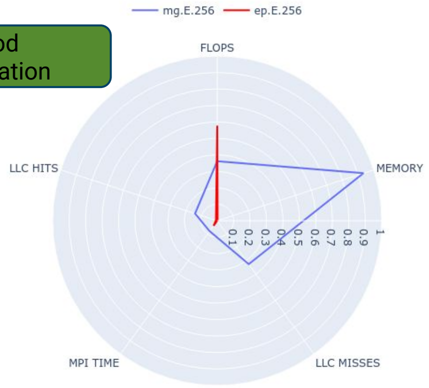
Tag



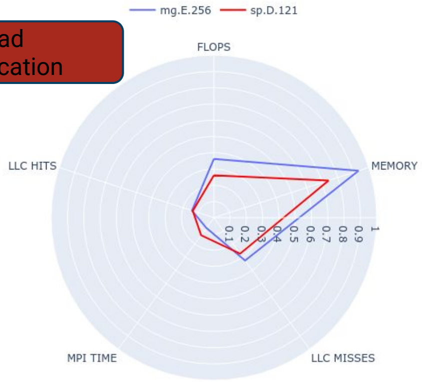

**<u>Starting Point:</u>** «one can reduce the average slowdown experienced by co-located applications by simply preventing instances of the same program from being co-located together» (de Blanche & Lundqvist, 2016)

**<u>Extension:</u>**
Minimal overlap between spider plots of two applications

good co-location




bad co-location



# Resource-centric tag-based models

**Static Experiment:** (300 applications, 100 experiments)

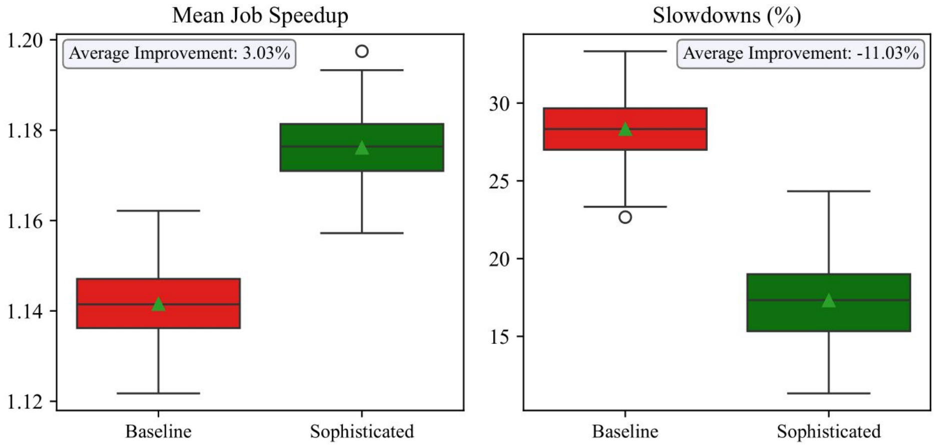
*Baseline Case*

**random co-location**

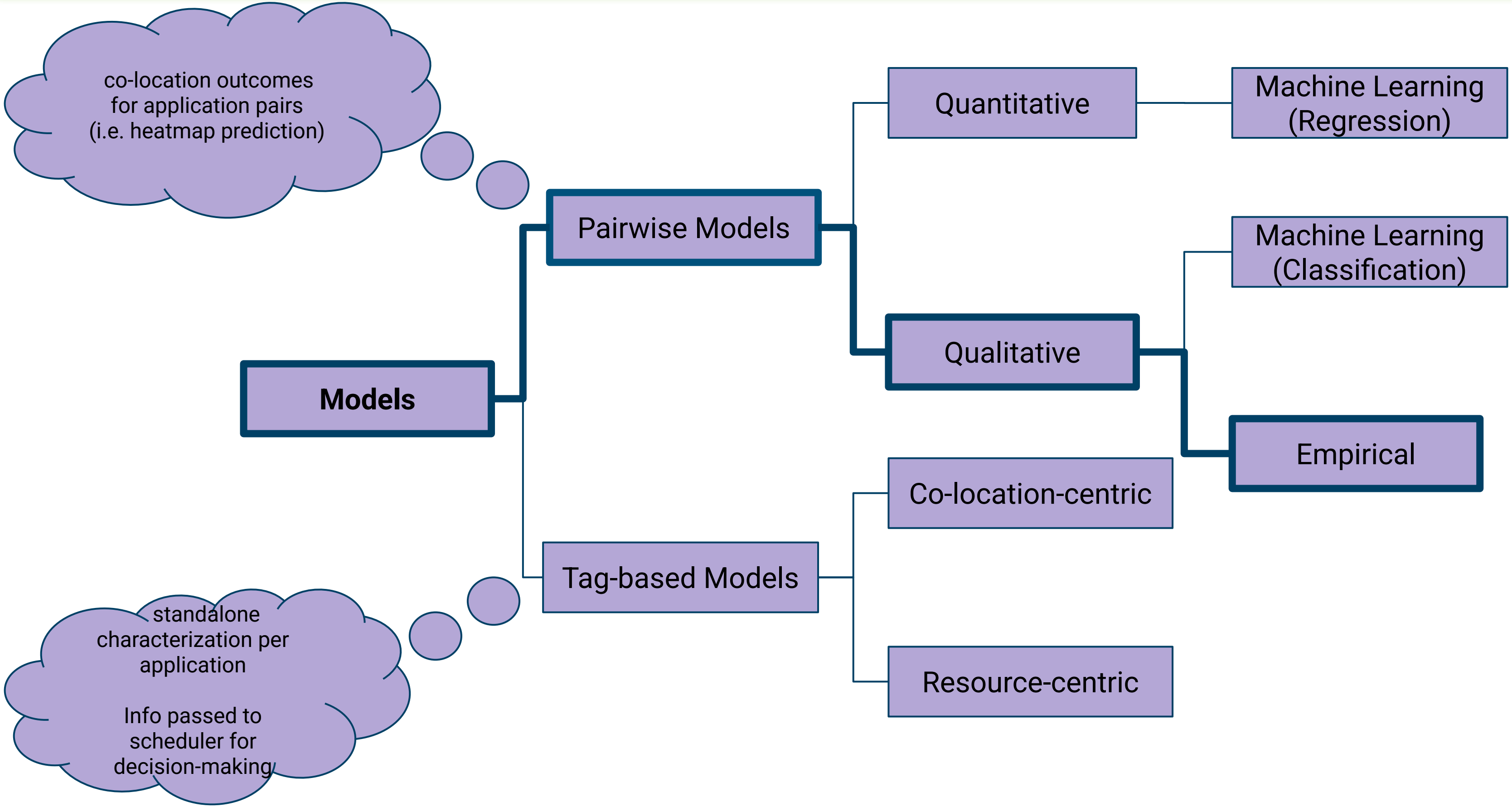
*Sophisticated Case*

**minimize spider plot overlap greedily**

**Evaluation:**



# Taxonomy of Application models

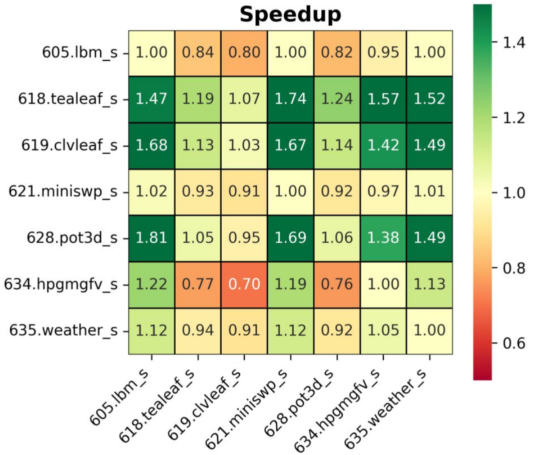


# Empirical Pairwise Model

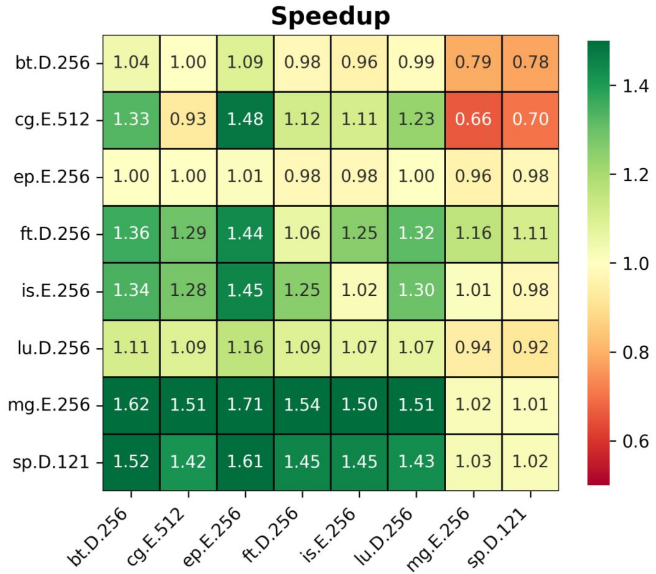
🎯 **Goal :** devise an empirical way to qualitatively predict the heatmap of an application pool

➢ use of categorical labels : <u>good</u>, <u>stationary</u>, <u>bad</u>

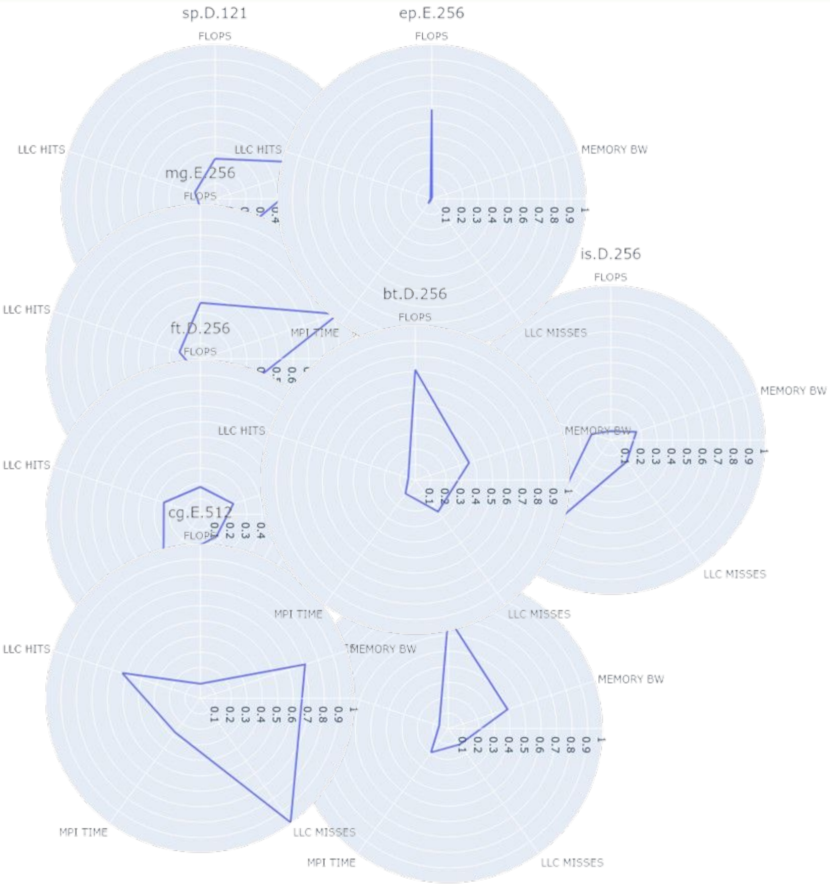
**SPEChpc 2021 benchmarks :**



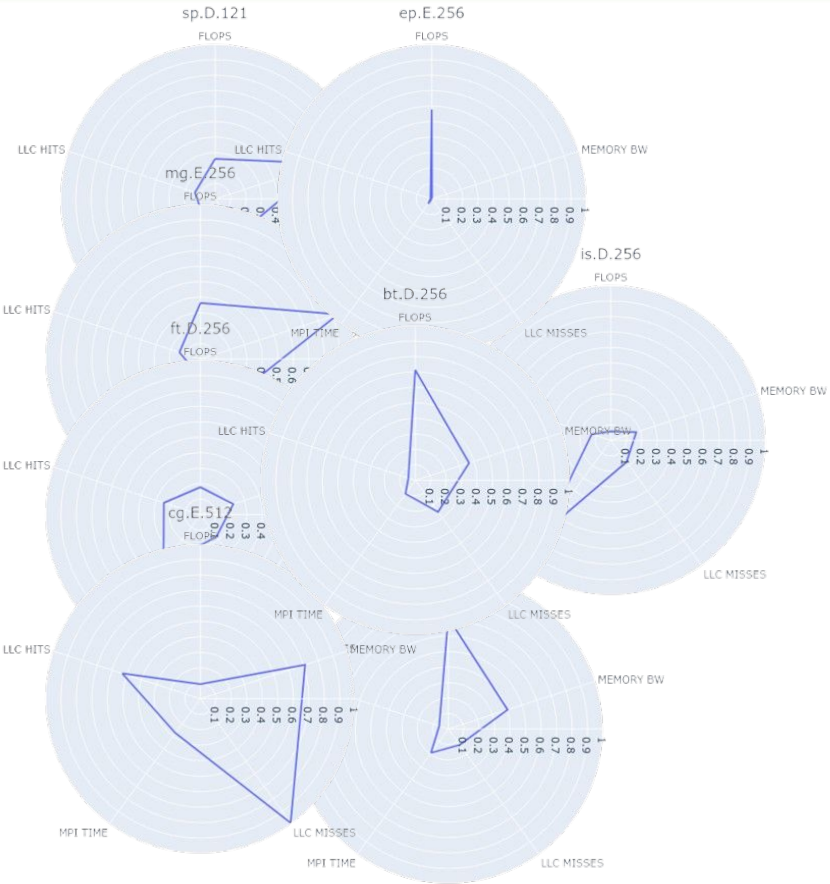

**NAS benchmarks:**

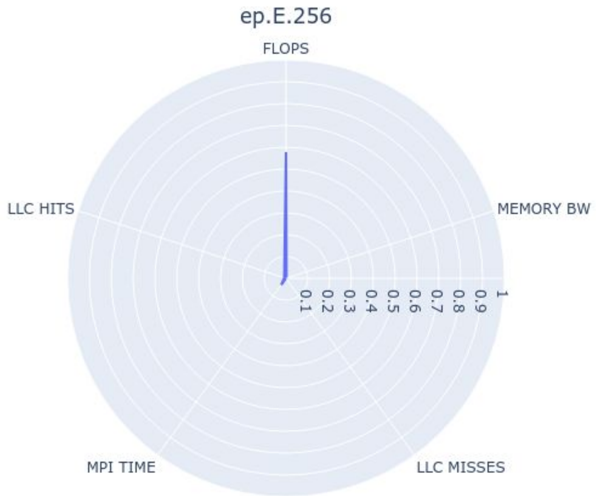


# Empirical Pairwise Model



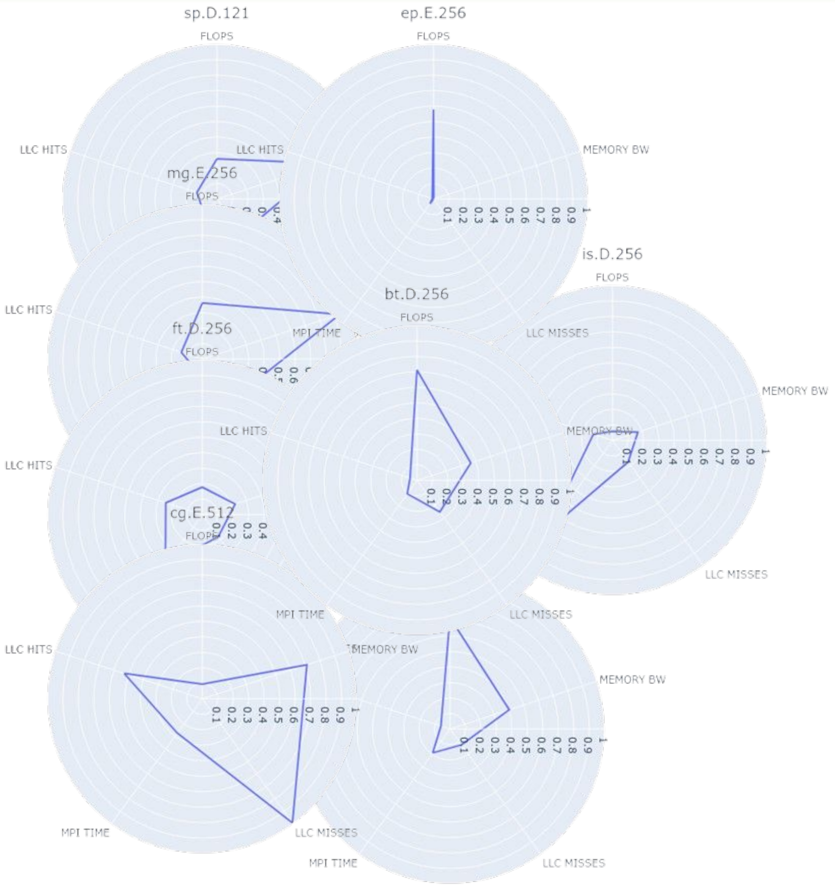
# Empirical Pairwise Model

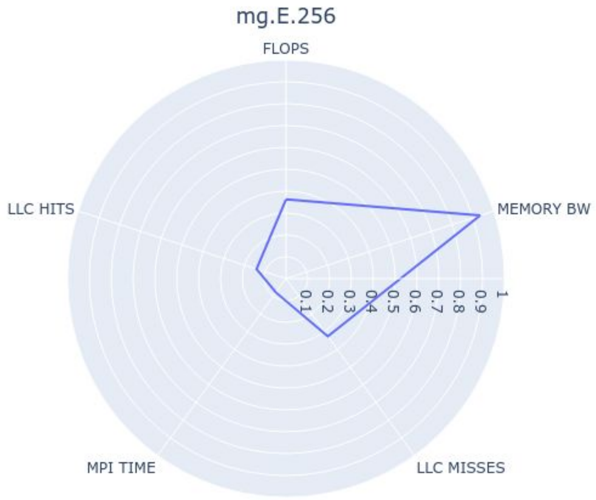



compute-bound application

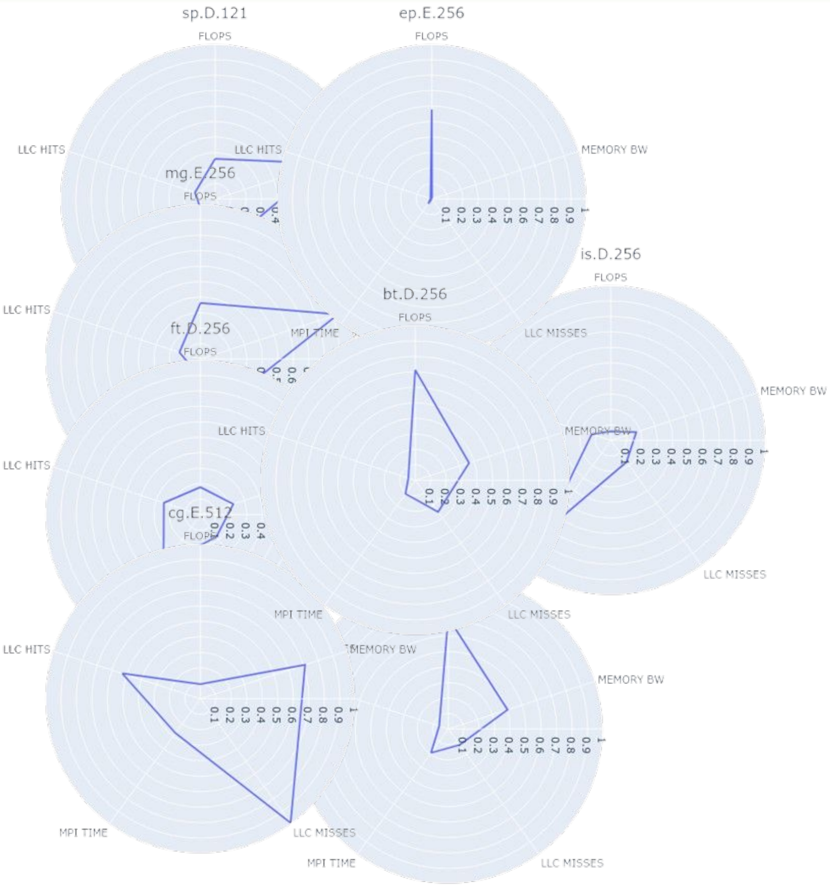
# Empirical Pairwise Model

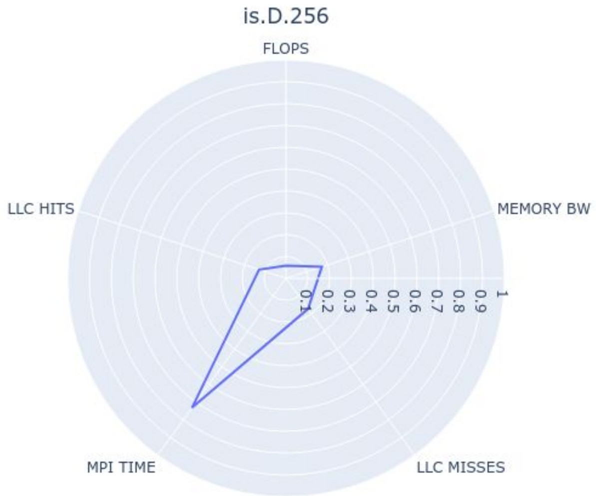



memory-bound application

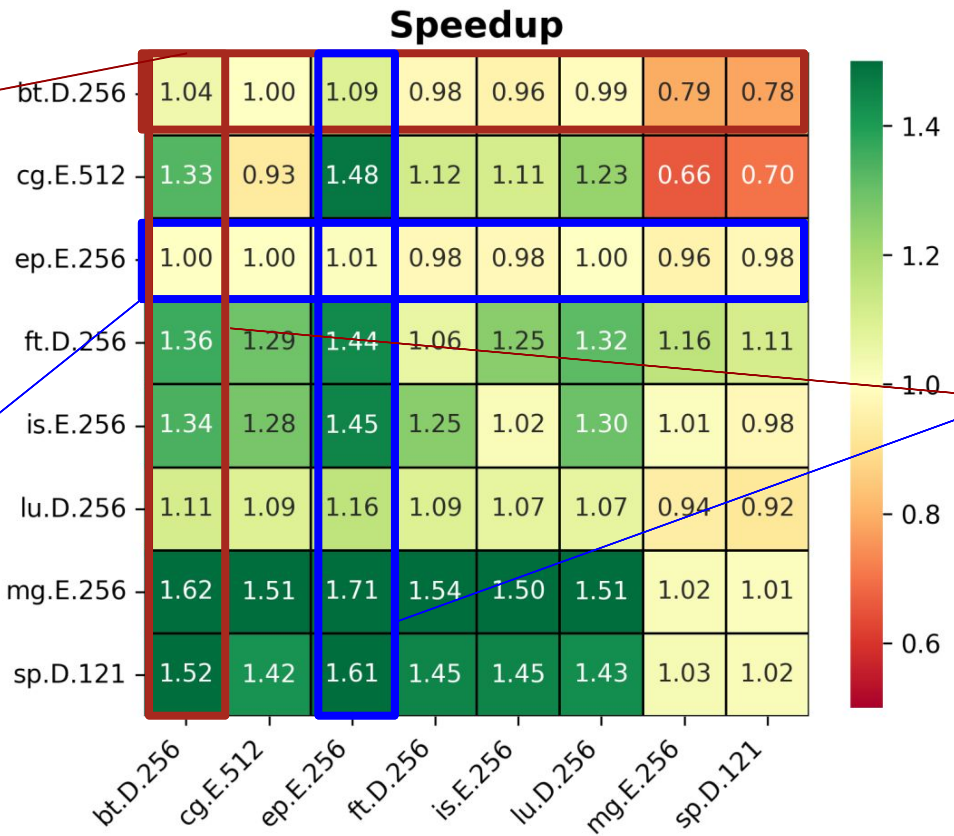
# Empirical Pairwise Model




communication-bound application

# Compute-bound applications



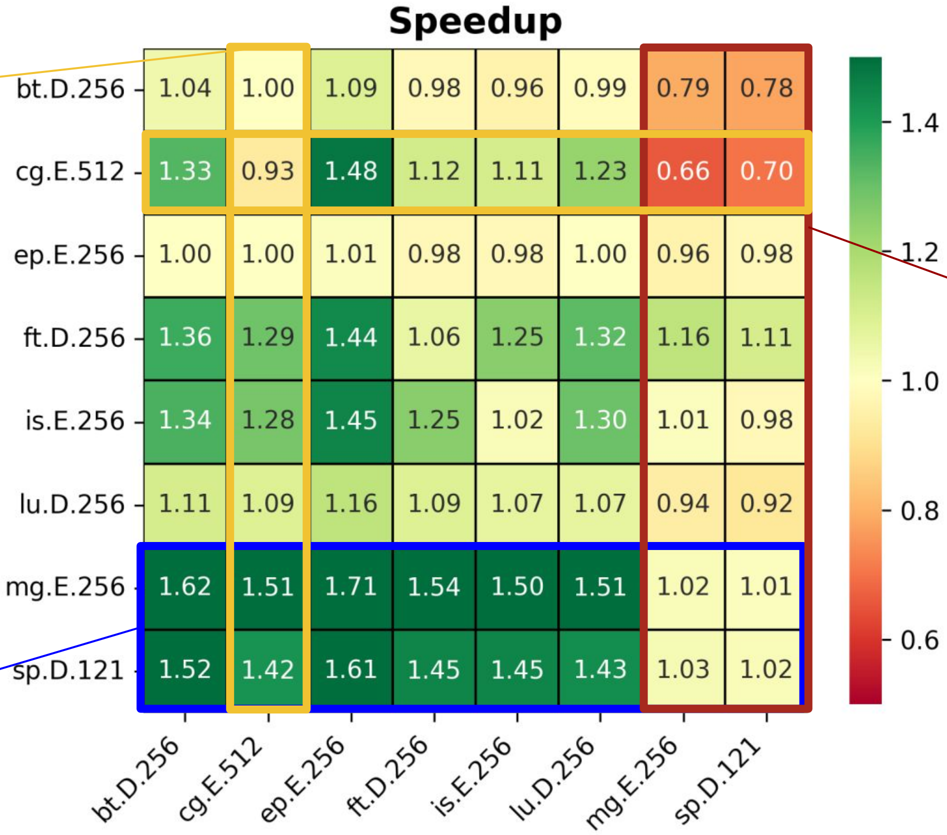

<u>Compute-bound with significant memory access:</u>

- slowdown when co-located with memory-bound

good neighbors for other applications

Heavily compute-bound applications are stationary

# Memory-bound applications



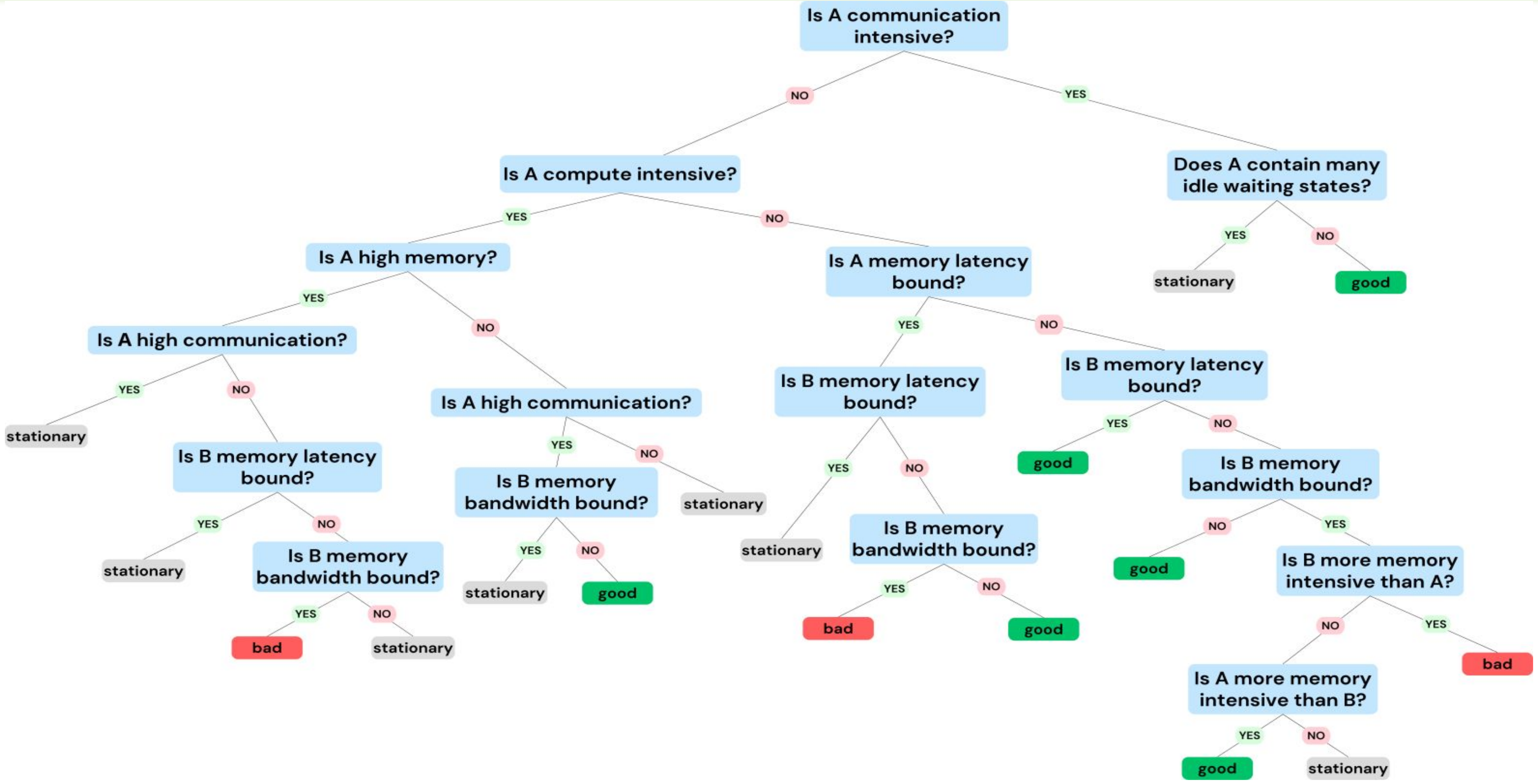

<u>Memory-bound with irregular memory access:</u>

- Low speedups
- slowdowns with memory-bound applications
- good neighbours

bad neighbors for other applications

- remarkable speedups
- worsened performances with other memory-bound applications

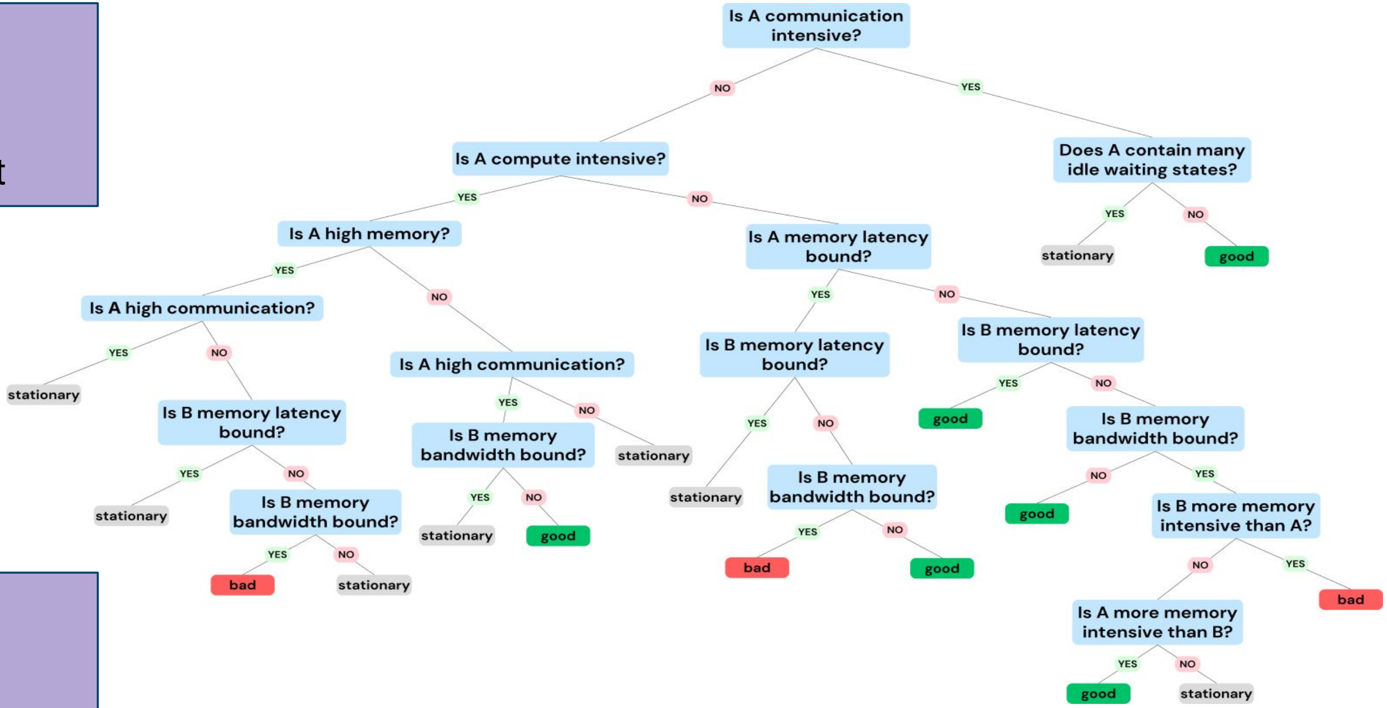
# Empirical Model



# Empirical Model - Evaluation

**NAS:**

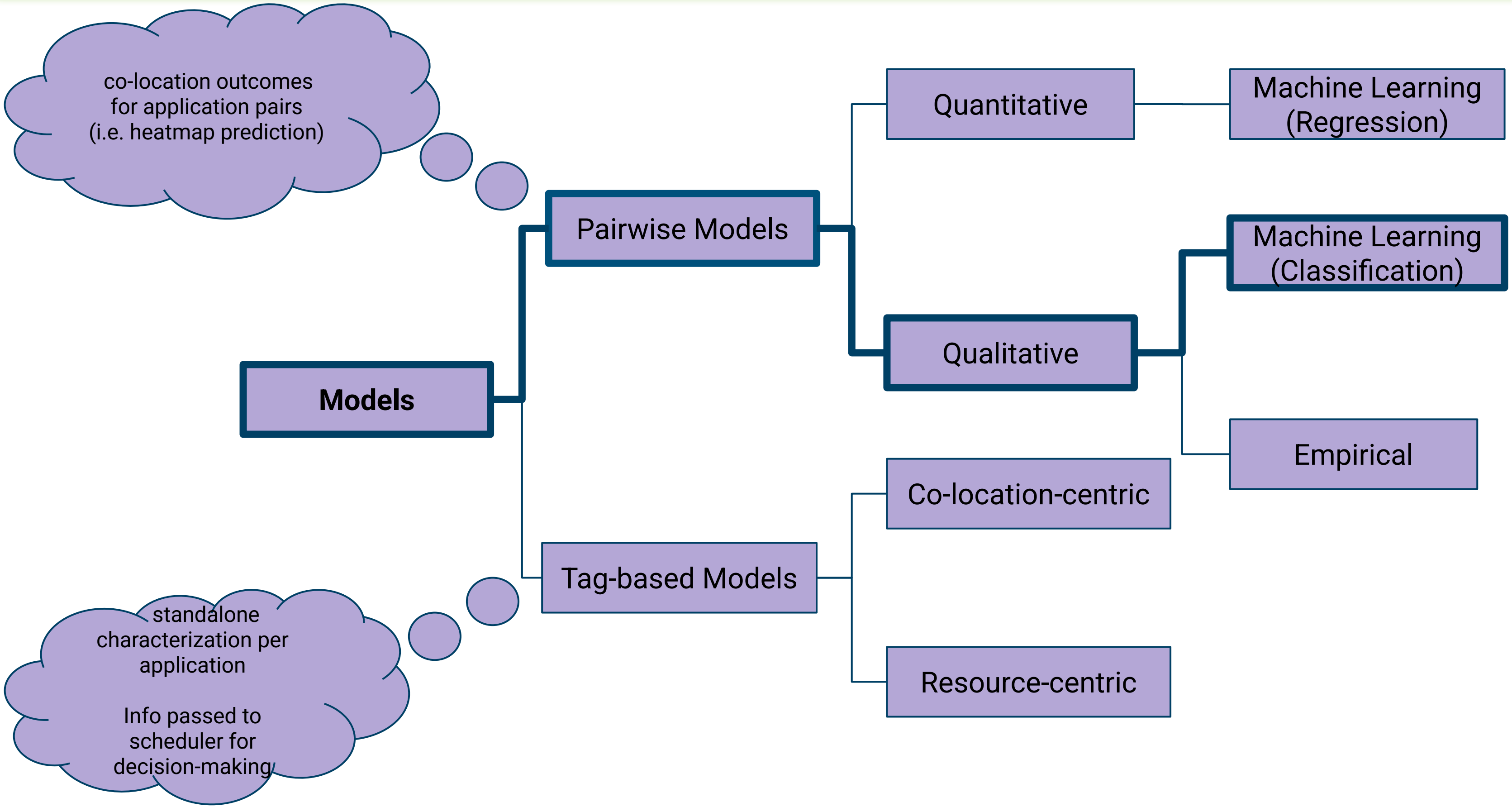
- ★ 83.43% fully correct
- ★ 7.1% sufficiently correct




**SPEChpc 2021:**

- ★ 63.27% fully correct
- ★ 22.45% sufficiently correct

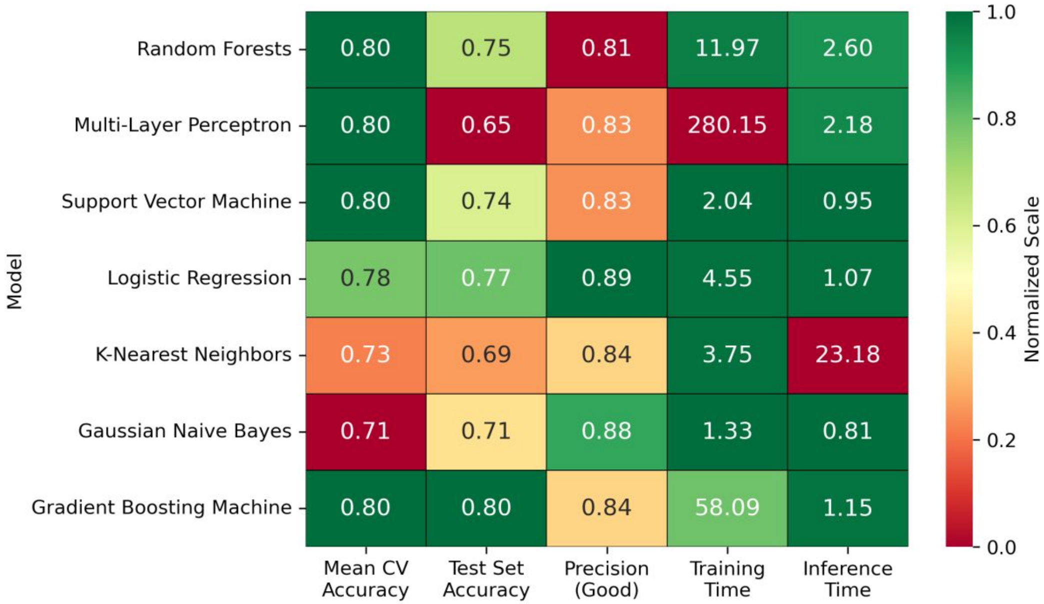
# Taxonomy of Application models



# ML-based Classifiers

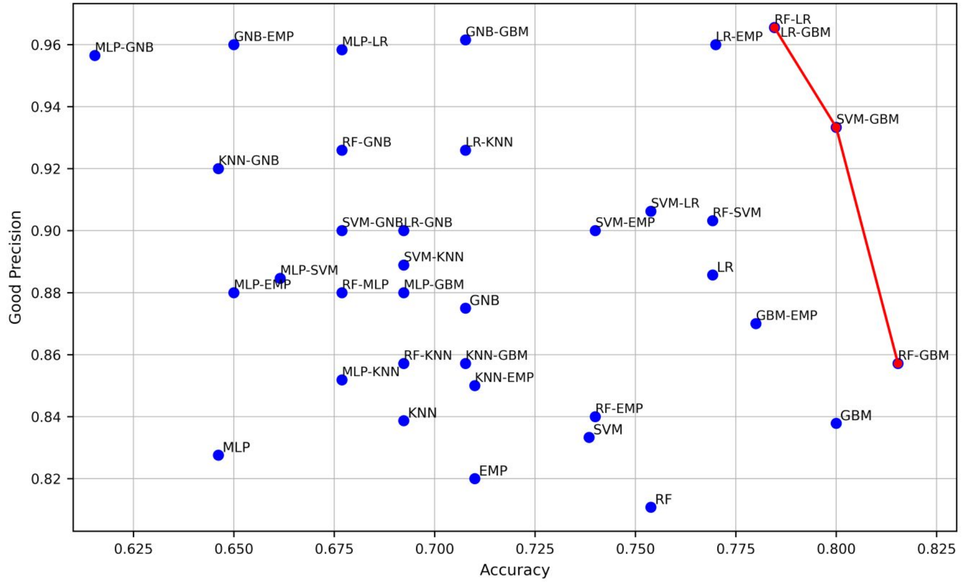
- ➔ **Features :** the five axes previously presented in the spider plots, normalized according to system bounds
- ➔ **Dataset :** 218 entries in total, with 70% allocated for training and the remaining 30% for testing
- ➔ **Labels :** good, stationary, bad
- ➔ **Methodology:** Train many models, with different hyperparameter combinations, from 7 different types of models, using 5-fold cross validation and keep the model from each type with the highest mean cross-validation accuracy score

# ML-based Classifiers



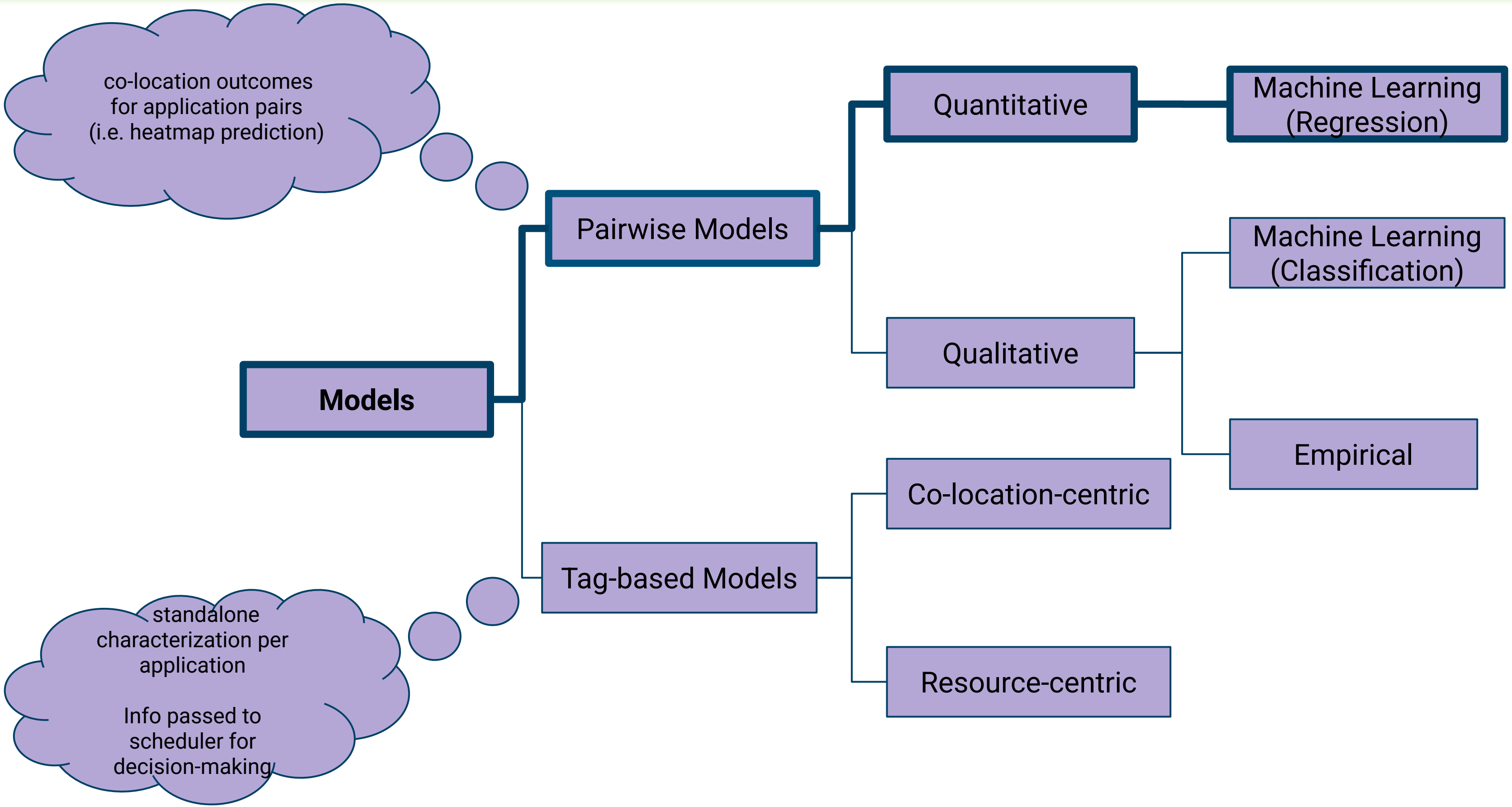

- **Ensemble boosting techniques** (e.g. RF, GBM) achieved the best overall results
- **Low-complexity models** (KNN, GNB) failed to grasp the complex, non-linear relations between the input features
- **Neural Networks** (MLP) need to be re-evaluated when bigger datasets are available
- Training and inference times appear to be reasonable

# ML-based Classifiers: Combining two models




- Mispredicting many bad co-locations as good (sacrificing precision) can result in a slowdown in the overall makespan
- Achieving <u>high accuracy</u> and <u>precision for the “good” label</u> concurrently, is a two-criteria optimization problem that can be solved by utilizing combinations of models as shown in the Pareto plot

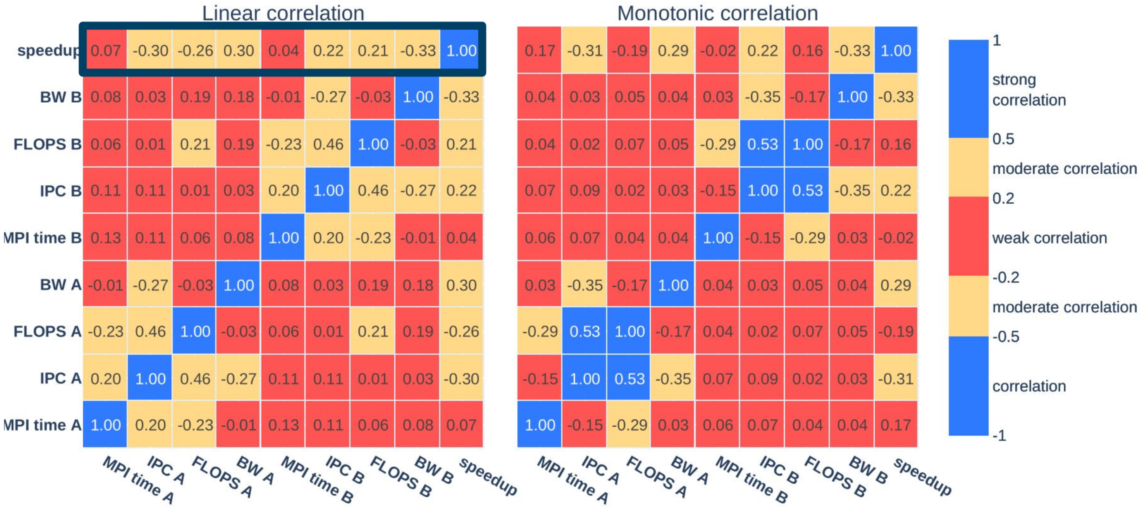
# Taxonomy of Application models



# ML-based Regression

- ➔ **Features :** MPI time, IPC, FLOPS, and Memory Bandwidth (BW)
- ➔ **Dataset :** 788 entries in total, with 65% allocated for training and the remaining 35% for testing
- ➔ **Labels :** Numerical value of the speedup for each heatmap cell
- ➔ **Methodology:** Train many models, with different hyperparameter combinations, from 7 different types of models (plus a Dummy Regressor used for comparison purposes), using 5-fold cross validation and keep the model from each type with the highest mean cross-validation accuracy score

# ML-based Regression




- moderate correlation of the input features with the dependent variable (speedup)
- Hint that *non-linear relations* are at play

# ML-based Regression


- **Linear models** underperformed due to low feature-speedup correlation
- **Ridge** best among linear models; SVR and KNN better but KNN overfitted
- **Boosting models** highest $R^2$, lowest RMSE/MAE, consistent performance across trials

# Conclusion

- Review of the presented methods with regards to the requirements:

| | Minimal Data use | Lightweight profiling | Fast Inference | High Accuracy and Precision | Interpretability | Generalizable to other machines |
|---|---|---|---|---|---|---|
| Tag-based | | | | N/A | | |
| Empirical | | | | | | |
| Linear/Simple ML | | | | | | |
| Boosting techniques | | | | | | |
| Neural Networks | | | | Needs further research | | |

# Future Work

- **Re-evaluation of Neural Networks**’ accuracy and performance utilizing bigger datasets

- Use of **new/composite features** with bigger correlation with co-location speedup

- **Integration** of these application models to a real co-scheduler or a simulation tool to test them in a dynamic co-scheduling scenario

# Thank you for your attention!

Q & A session