Serving Unseen Deep Learning Models with Near-Optimal Configurations: a Fast Adaptive Search Approach

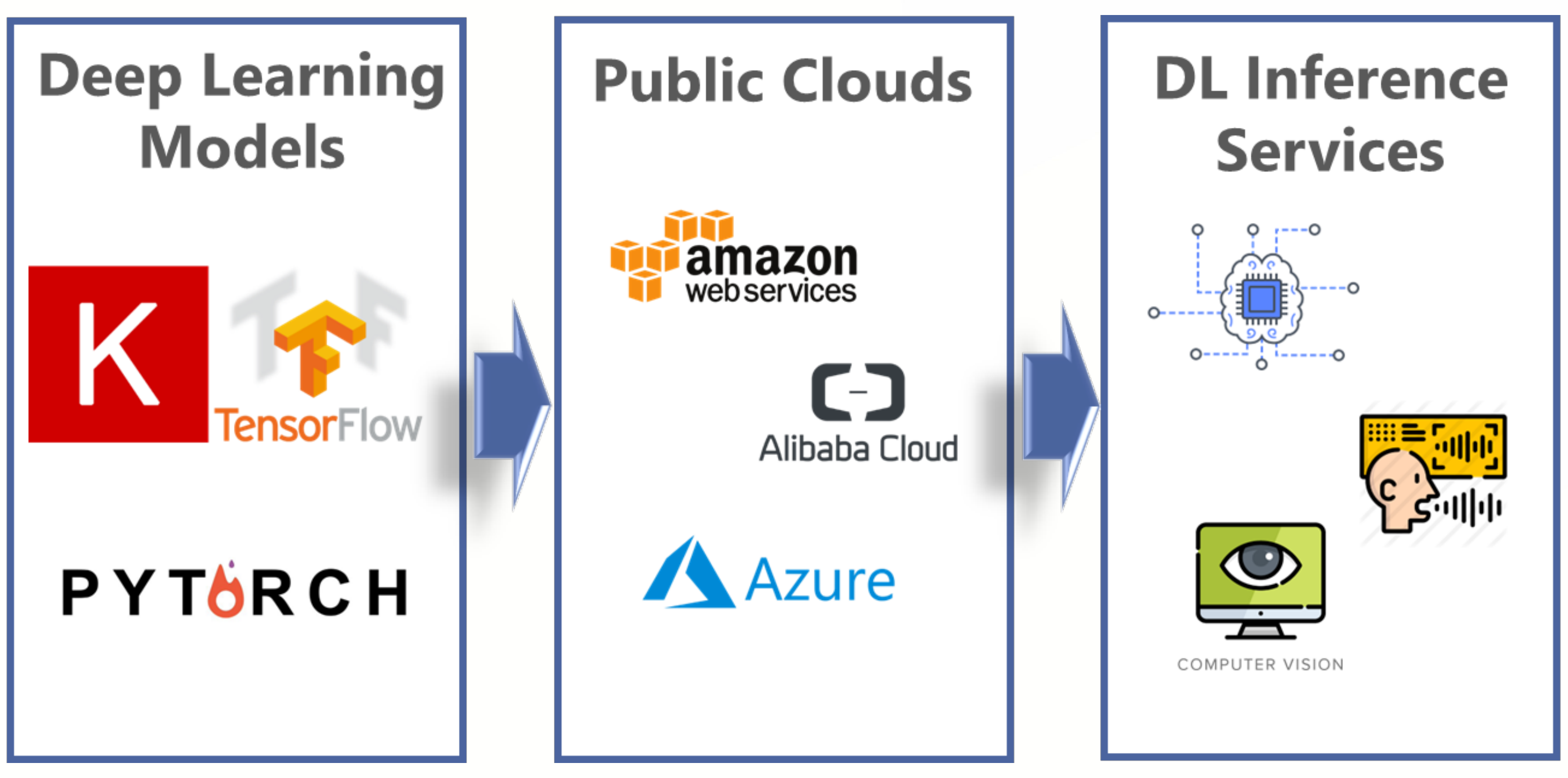
为未见过的深度学习模型选择接近最优的配置：一种快速适配搜索方法

In the13th edition of the annual *ACM Symposium on Cloud Computing, ACM SoCC 2022*.

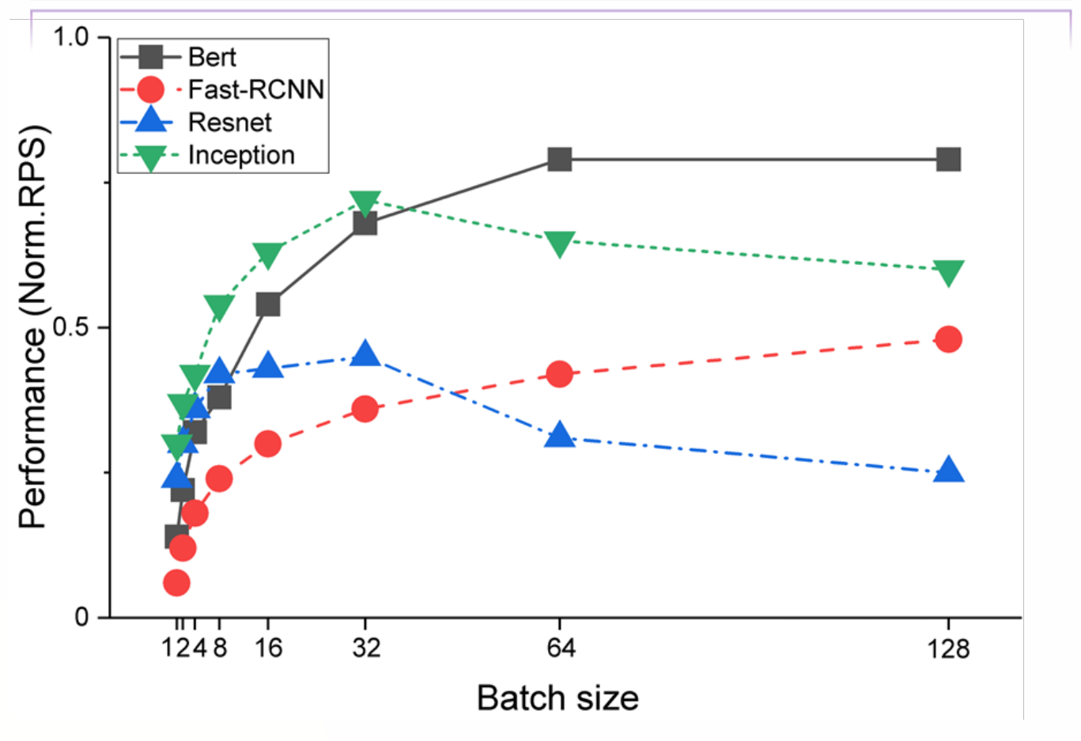
吴悦文，吴恒，罗钧寒，许源佳，胡艺，张文博，钟华

联系人: 吴悦文，18600612053，wuyuewen@otcaix.iscas.ac.cn

Serving deep learning models on public clouds becomes popular

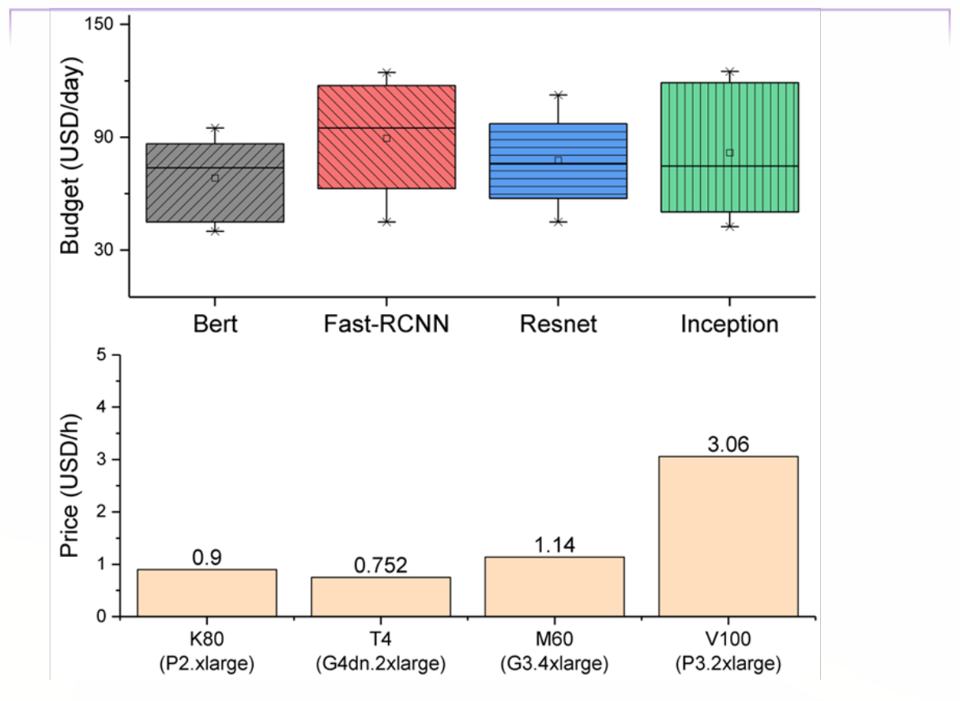


* The performance is evaluated by *request per second*, or RPS.



Improving up to 8 times performance

* The budget is evaluated by *USD*, and it is calculated by *instance price* × *running time*.



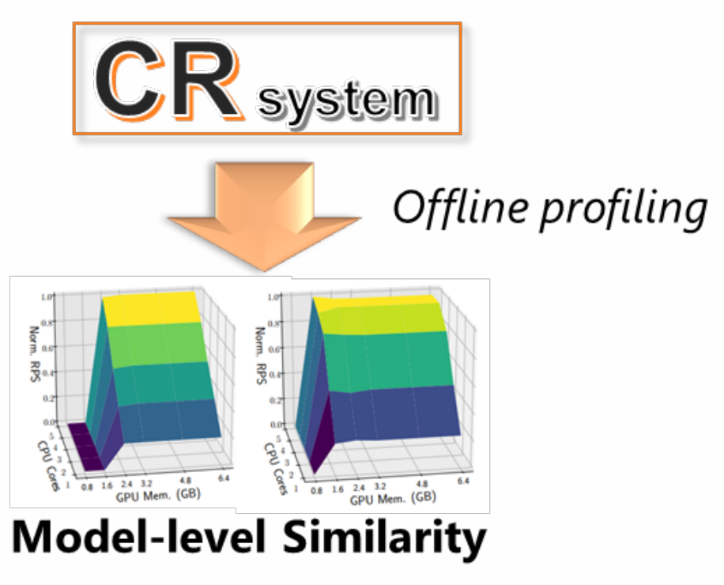
Reducing over 60% budget

Configuration is the key of improving performance and reducing budget !!!

✠ Resource & runtime configurations: GPU type, GPU memory, batch size ✠

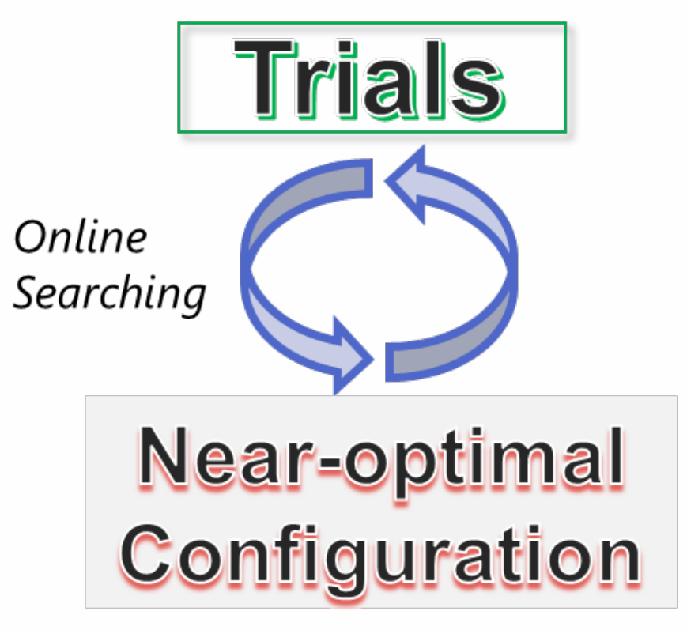
Existing configuration recommender (CR) systems suffer from a severe cold start problem, especially for unseen DL models.

(a) Learning **model-level similarity** (resource sensitivity curves) by offline profiling.

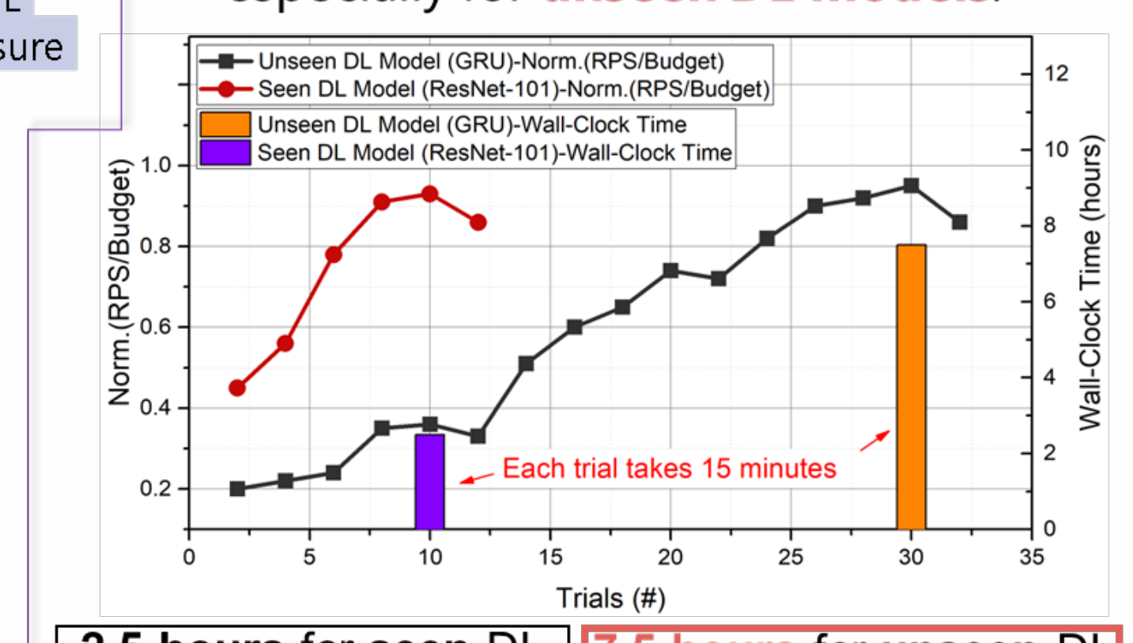


(b) Running *trials*[1] for the target DL model to search online for a near-optimal configuration.

[1] The *trial* is a stress test of the target DL model in a certain configuration to measure its performance.



(c) However, we find that **model-level similarity** suffers from severe **cold start problem**, especially for **unseen DL models**.



2.5 hours for **seen** DL models because of **high** model-level similarity

7.5 hours for **unseen** DL models because of **low** model-level similarity

Existing CR systems require dozens of *trials* to find near-optimal configurations for unseen DL models, mainly because:

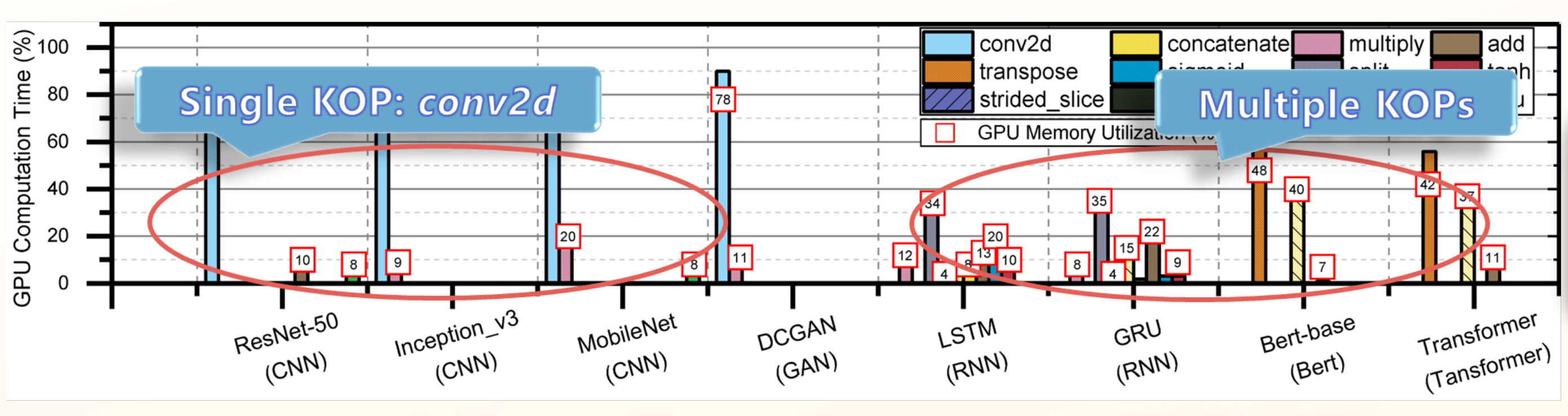
- ◆ **Large search space:** over 1,000 configuration candidates.
- ◆ **Complex DL models and model variants**.
- ◆ **Poor model-level similarity**.

For a given unseen DL model, how to find a near-optimal configuration over *a few trials* to alleviate the cold start problem?

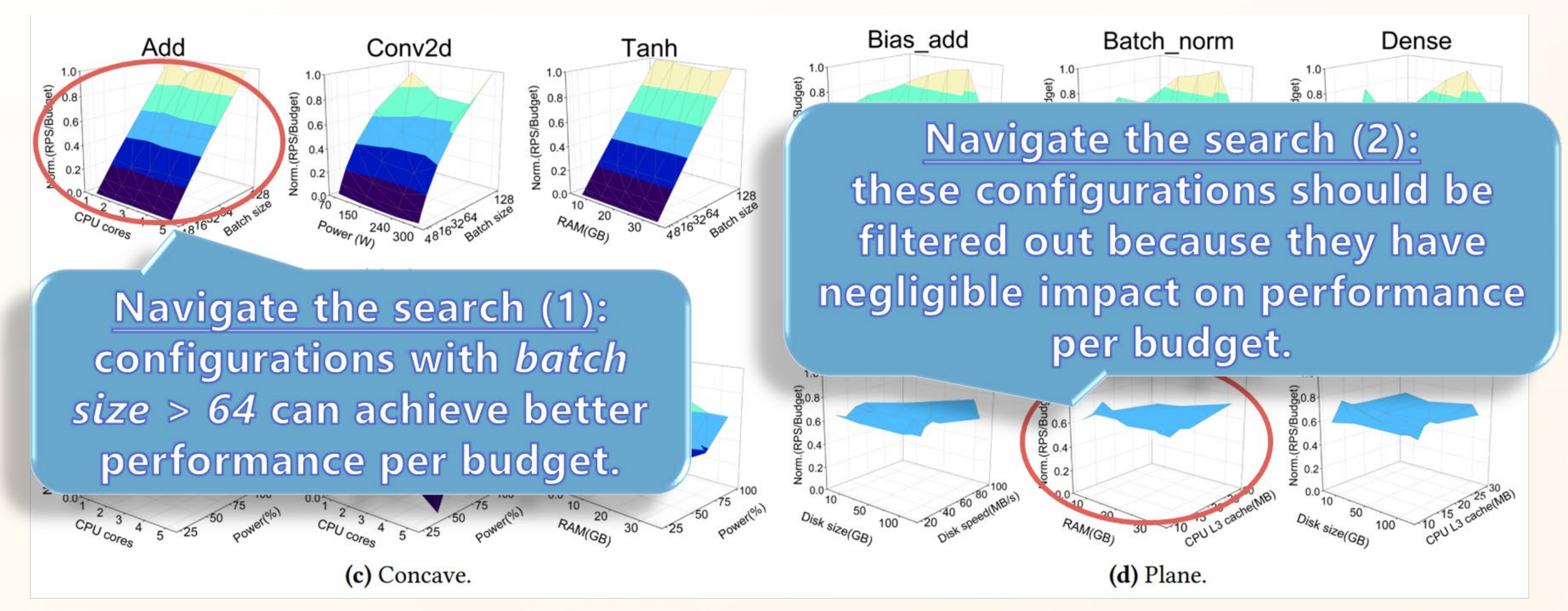
Key insight: Leveraging operator-level instead of model-level similarity

We made 2 core findings in a large-scale evaluation: KOPs and KOP-RCs.

Through a large-scale evaluation on Amazon EC2 with 30 typical DL models, we find that DL operators are better suited to describe the performance of DL models.



For a given DL model, there are some Key Operators (KOPs) to depict its GPU computation time and GPU memory utilization



Navigate the search (1): configurations with *batch size > 64* can achieve better performance per budget.

Navigate the search (2): these configurations should be filtered out because they have negligible impact on performance per budget.

(c) Concave.

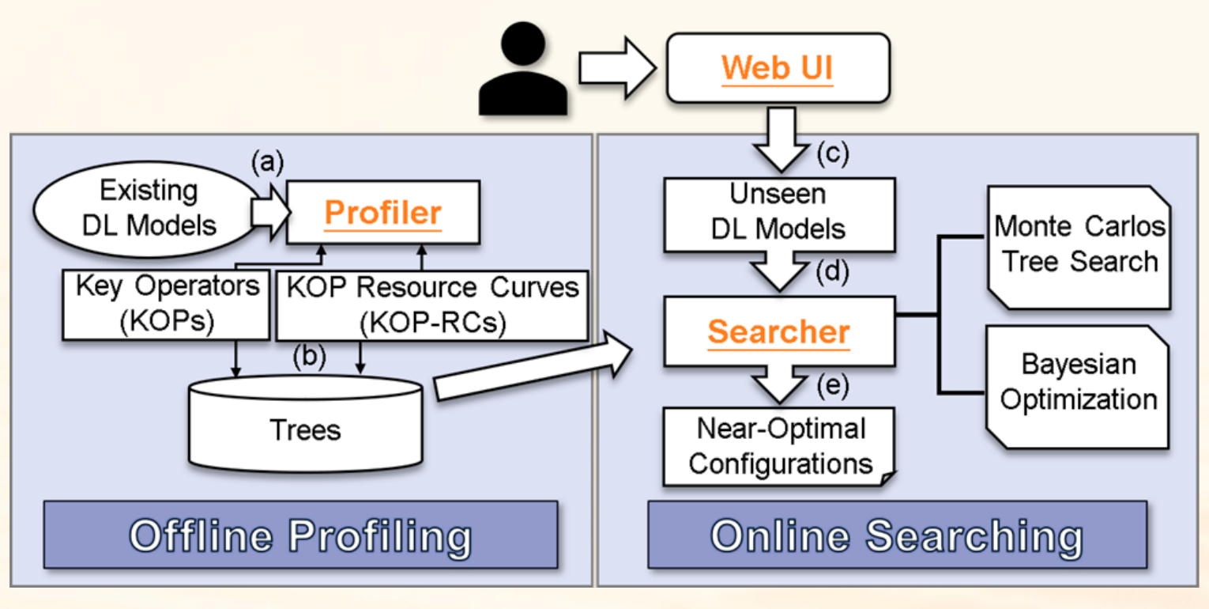
(d) Plane.

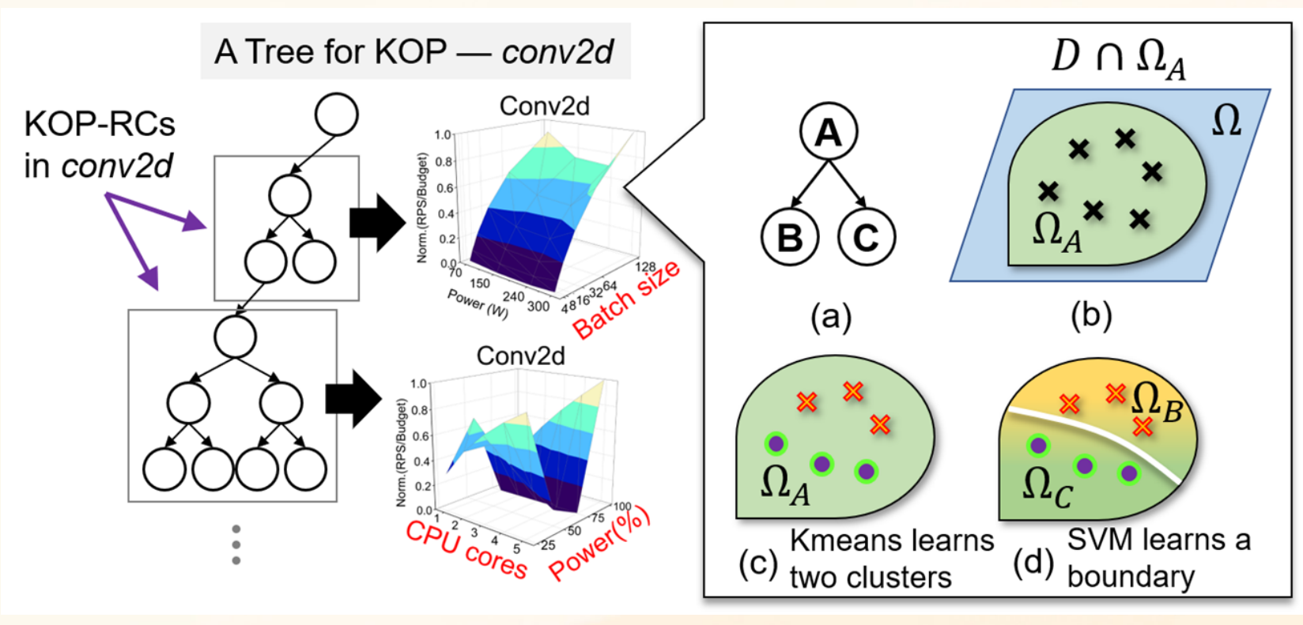
For each KOP, there are four typical Key Operator Resource Curves (KOP-RCs) to navigate the search of near-optimal configurations

Falcon: a Fast Adaptive Configuration Recommender System

Falcon works within a two-phase framework:

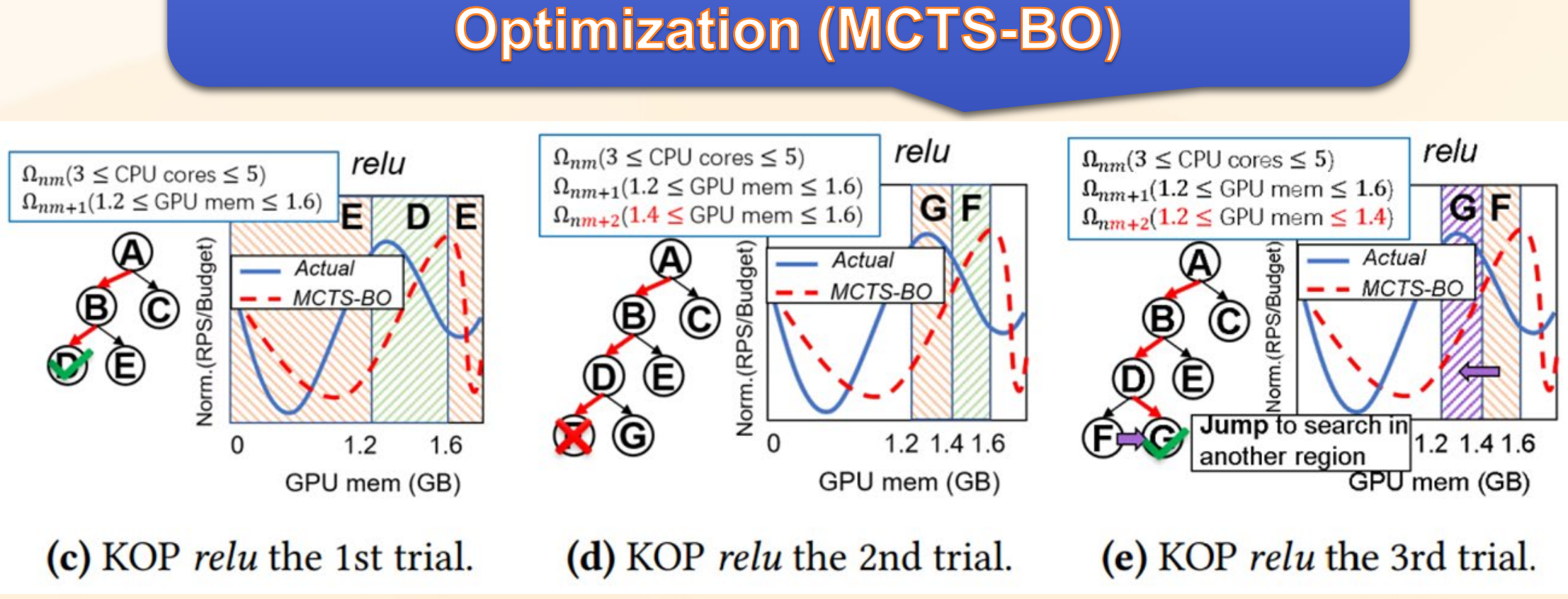
- ◆ **Offline Profiling:** learn KOPs and KOP-RCs.
- ◆ **Online Searching:** fast adaptive search by reusing KOPs and KOP-RCs.





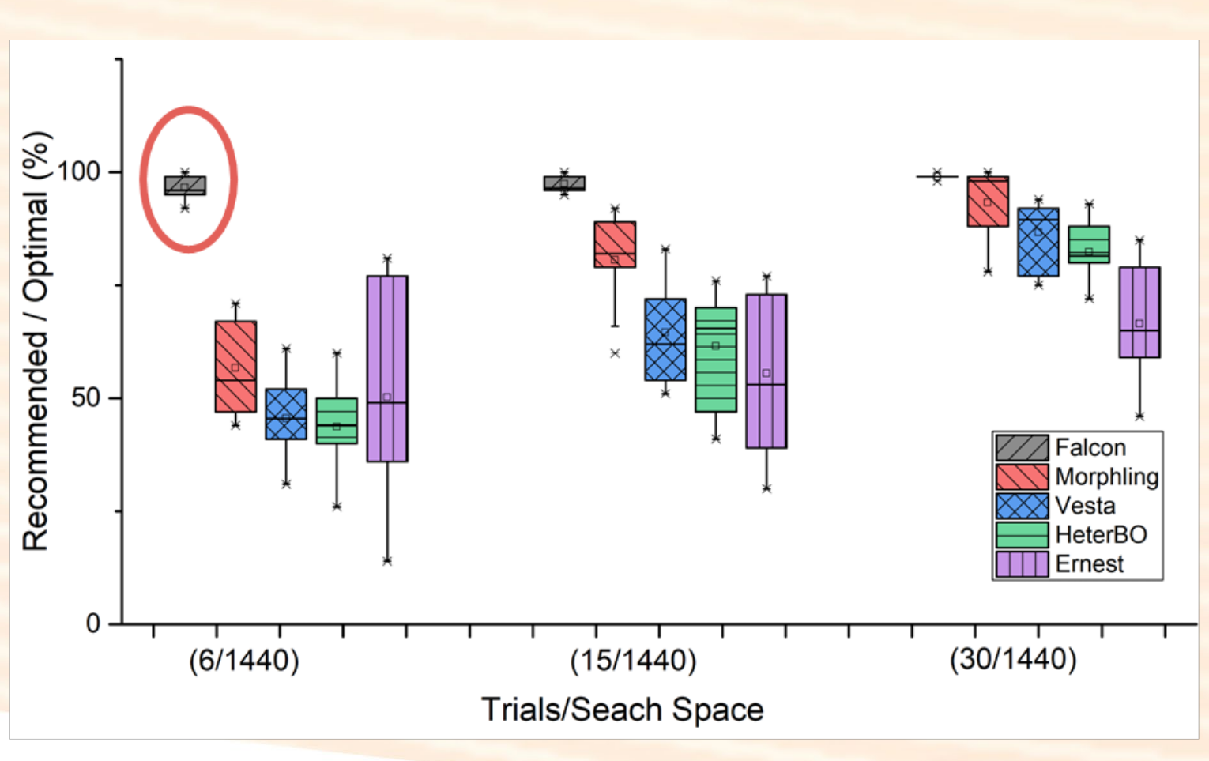
Offline: Constructing tree to locate near-optimal configurations

Online: Fast adaptive search via Monte Carlo Tree Search and Bayesian Optimization (MCTS-BO)

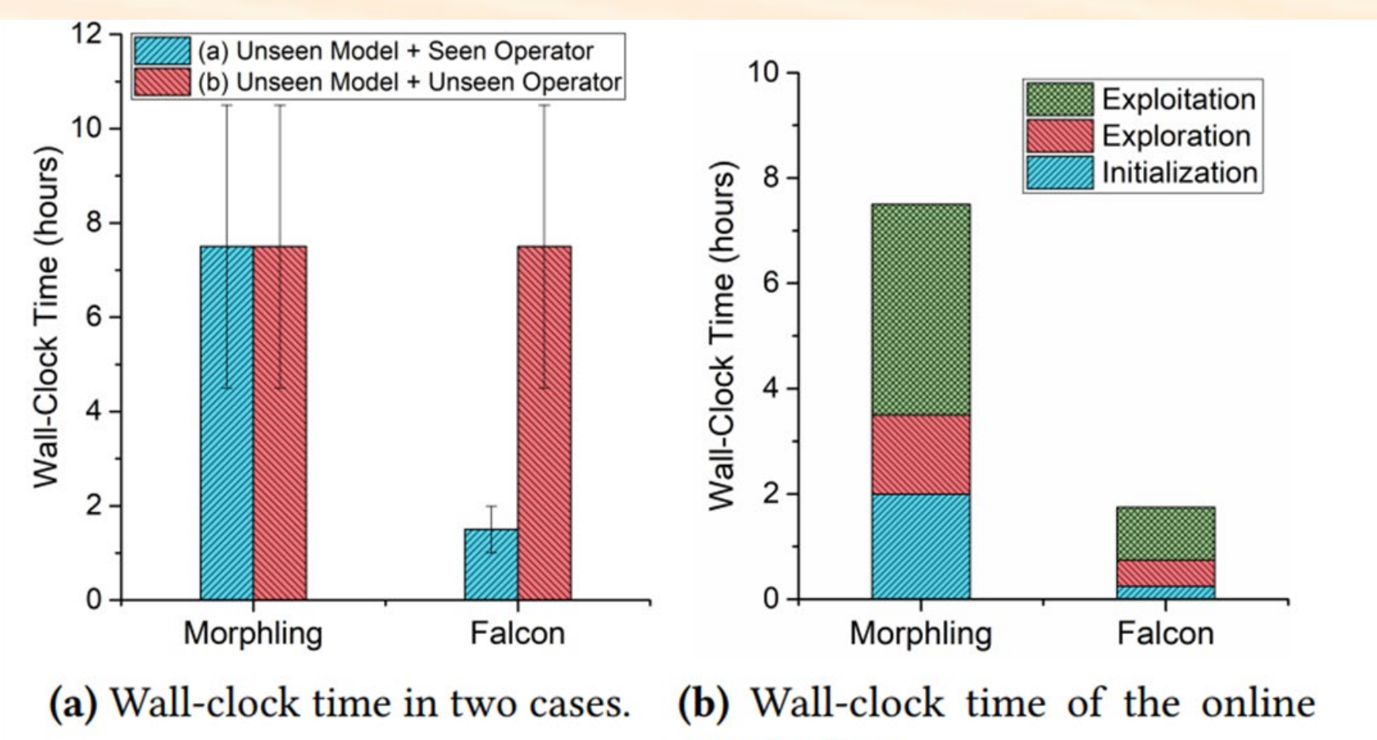

$\Omega_{nm}(3 \leq \text{CPU cores} \leq 5)$
$\Omega_{nm+1}(1.2 \leq \text{GPU mem} \leq 1.6)$
$\Omega_{nm+2}(1.4 \leq \text{GPU mem} \leq 1.6)$
$\Omega_{nm+2}(1.2 \leq \text{GPU mem} \leq 1.4)$
relu
Actual
MCTS-BO
Norm.(RPS/Budget)
GPU mem (GB)
A
B
C
D
E
F
G
Jump to search in another region

(c) KOP *relu* the 1st trial. (d) KOP *relu* the 2nd trial. (e) KOP *relu* the 3rd trial.

Evaluation: reducing the search overhead for unseen DL models by up to 80%

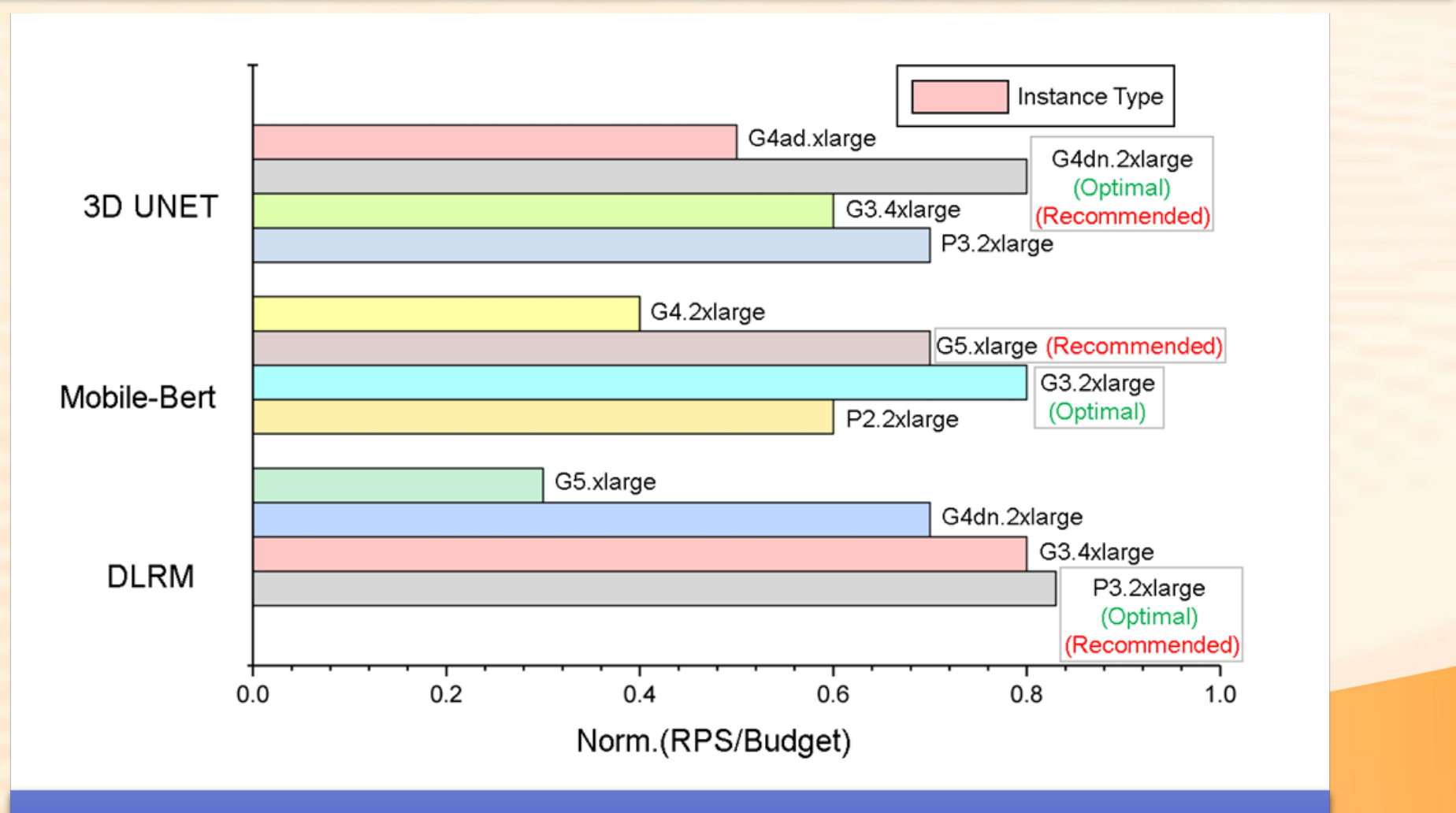


For unseen DL models, only Falcon can find near-optimal configurations after 6 *trials*



(a) Wall-clock time in two cases. (b) Wall-clock time of the online search phase.

Falcon can reduce up to 80% of search overhead by taking full advantage of KOPs and KOP-RCs



Real-world applications: Falcon can recommend a near-optimal configuration