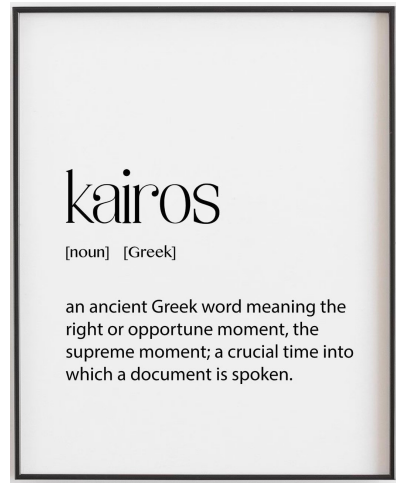
Kairos: Building Cost-Efficient Machine Learning Inference Systems with Heterogeneous Cloud Resources



Baolin Li, Siddharth Samsi, Vijay Gadepally, Devesh Tiwari

LVX VERITAS VIRTVS
Northeastern University

MIT LINCOLN LABORATORY

Kairos Executive Summary

A high-throughput ML inference system that is effective under QoS and cost budget constraints

Explores Two Important Questions

Is heterogeneity in hardware always beneficial for building high-performance ML inference services?

How to provision an effective heterogenous ML inference system and distribute ML inference queries on them?

ML-based services are deployed in cloud datacenters with heterogeneous resources

NVIDIA to Bring AI to Every Industry, CEO Says

From AI training to deployment, semiconductors to software libraries, systems to
Jensen Huang outlined how a new generation of breakthroughs will be put at the

Meta AI

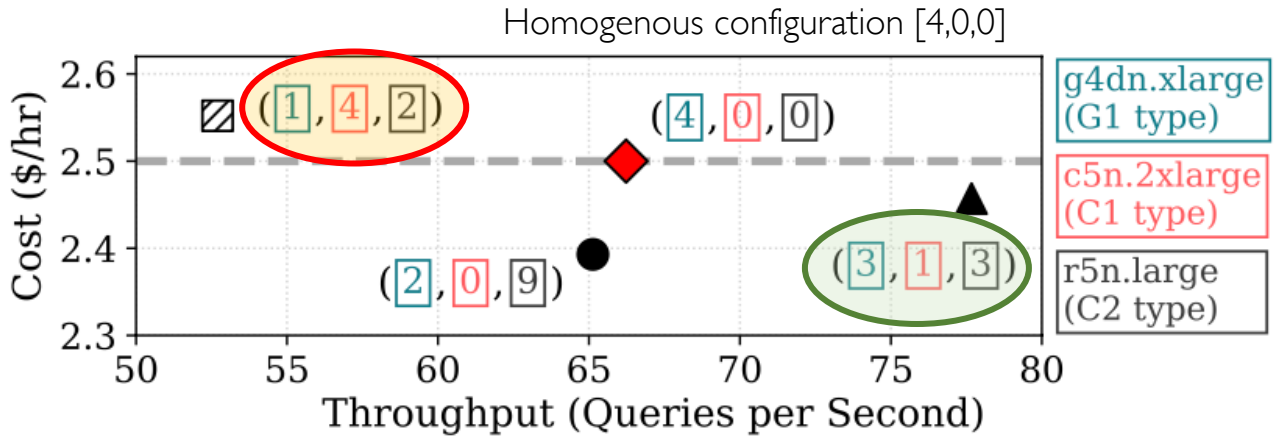
Hardware

MTIA v1: Meta's first-generation AI inference accelerator

Google Cloud unveils world's largest publicly available ML hub with Cloud TPU v4, 90% carbon-free energy

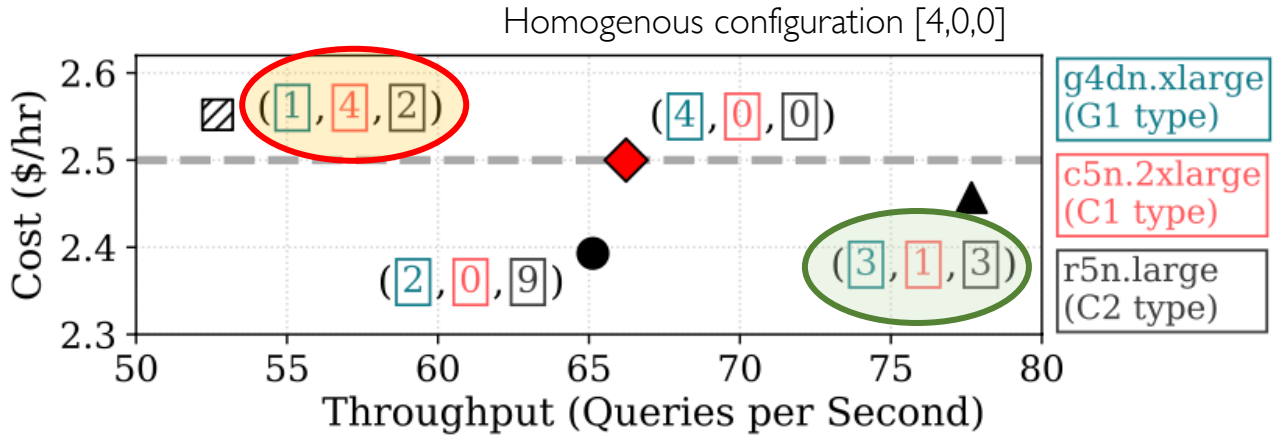
Announcing AWS Inferentia: Machine Learning Inference Chip

… but, exploiting heterogeneity optimally for ML inference serving is challenging!

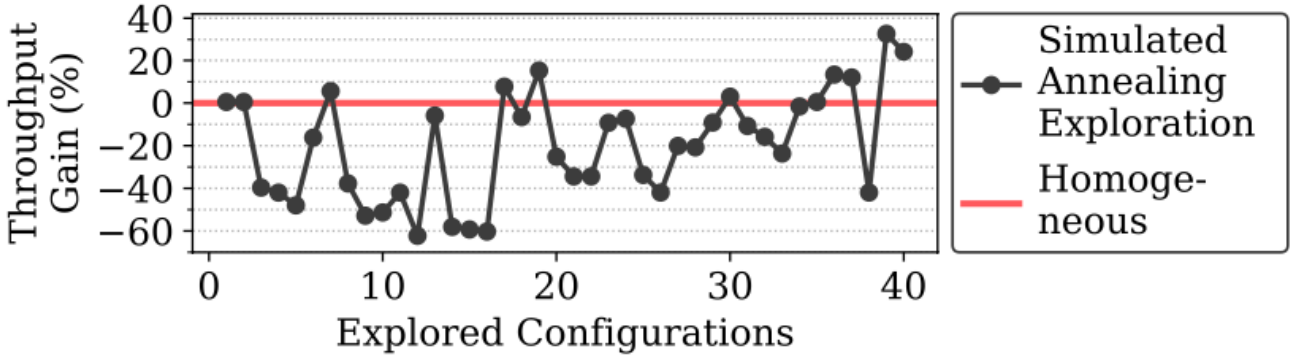


Heterogeneity can be worse than homogeneity.

… but, exploiting heterogeneity optimally for ML inference serving is challenging!

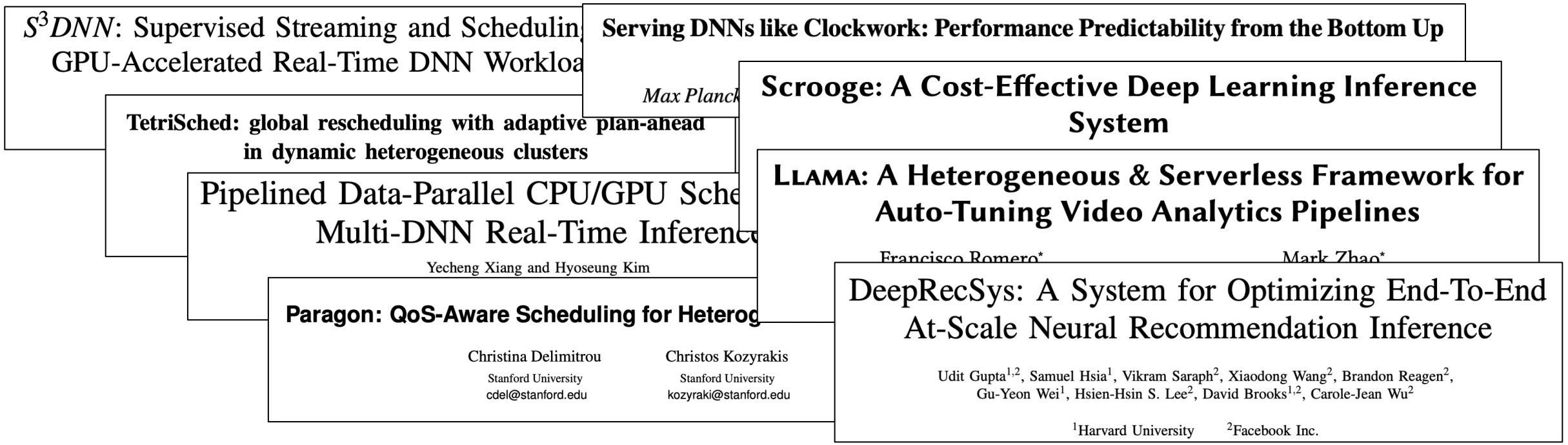


Heterogeneity can be worse than homogeneity.



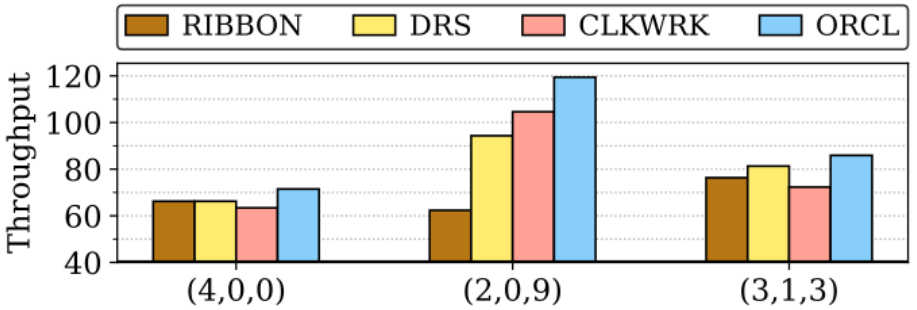
Finding effective heterogenous configuration requires expensive exploration of configurations.

Rich Literature of ML Inference Serving



Why are existing approaches not sufficient or optimal?

	Inference QoS	Through-put	Cost	Query Mapping	Proactive in Heterogeneity	No Online Exploration	Miscellaneous Notes
Paragon [10]	✘	✔	✘	✔	✘	✘	Requires prior data for training
TetriSched [11]	✘	✘	✘	✔	✘	✔	Supports user-based reservation
S^3DNN [13]	✔	✔	✘	✔	✘	✔	Uses supervised CUDA stream
DART [14]	✔	✔	✘	✔	✘	✘	Profiles layers and applies parallelism
Scrooge [15]	✔	✔	✔	✘	✘	✘	Chain execution of media applications
Ribbon [16]	✔	✔	✔	✘	✔	✘	Bayesian Optimization for allocation
DeepRecSys [17]	✔	✔	✘	✔	✘	✘	Schedules using profiled threshold
Clockwork [18]	✔	✔	✘	✔	✘	✔	Consolidates latency for predictability
KAIROS	✔	✔	✔	✔	✔	✔	Full heterogeneity support



Prior state-of-the-art solutions do not proactively exploit heterogeneity for cost and performance-effectiveness, incur high overhead during heterogeneity exploration, and suffer from sub-optimal inference query distribution/dispatching.

Kairos Goals and Key Ideas

Goals

Maximize throughput

Meet cost budget

Meet Quality-of-service (QoS)

Fast convergence

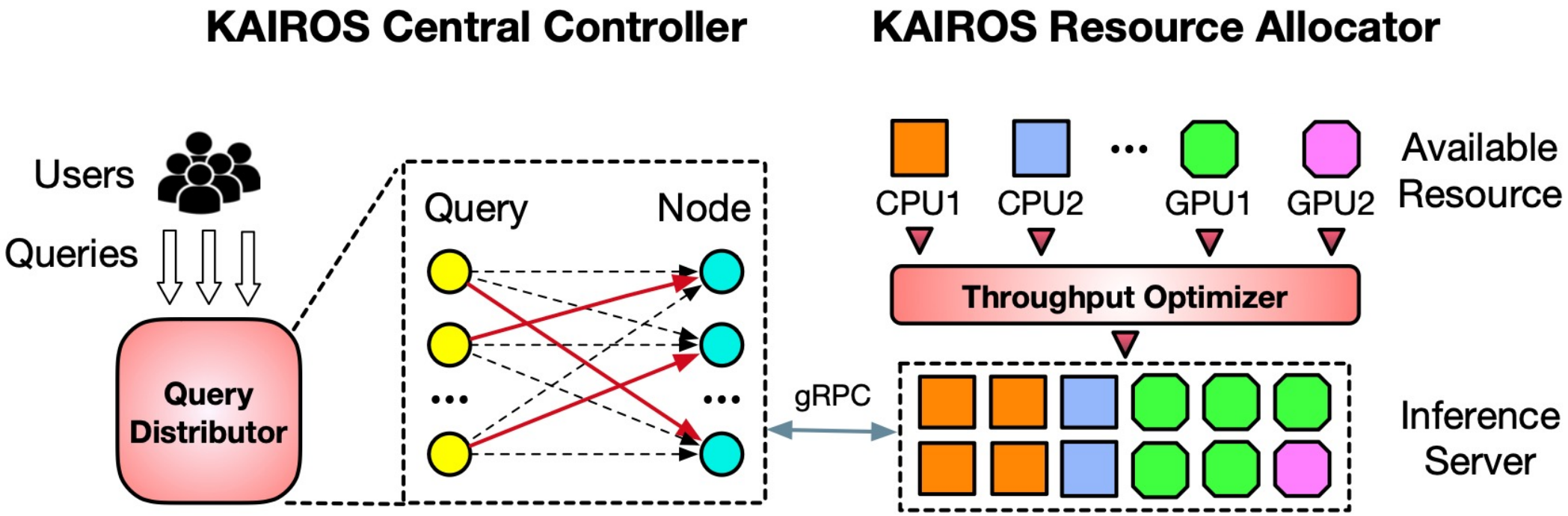
Heterogeneity-aware Query Dispatching Mechanism

Given heterogeneous hardware resources, a novel policy to optimally distribute the inference queries to the heterogenous hardware, in a QoS- and throughput-aware fashion

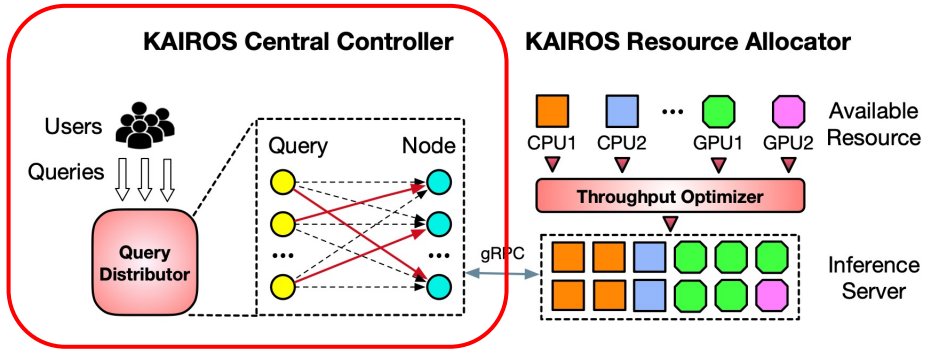
Finding Near-optimal Heterogenous Configuration without Online Exploration

Given a query distribution policy, design an optimizer to quickly find a near-optimal heterogeneous hardware configuration under cost budget

Kairos System Overview



Kairos' Inference Query Dispatcher



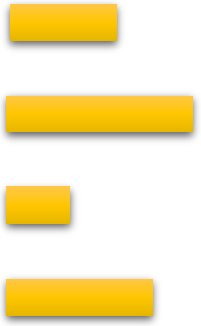
Processing time

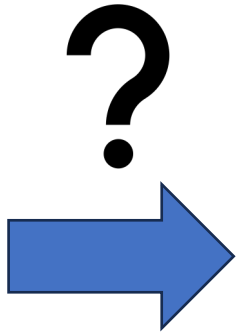
QoS constraint time

Waiting time

Queries

(with different batch sizes)





Servers

(CPUs, GPUs)

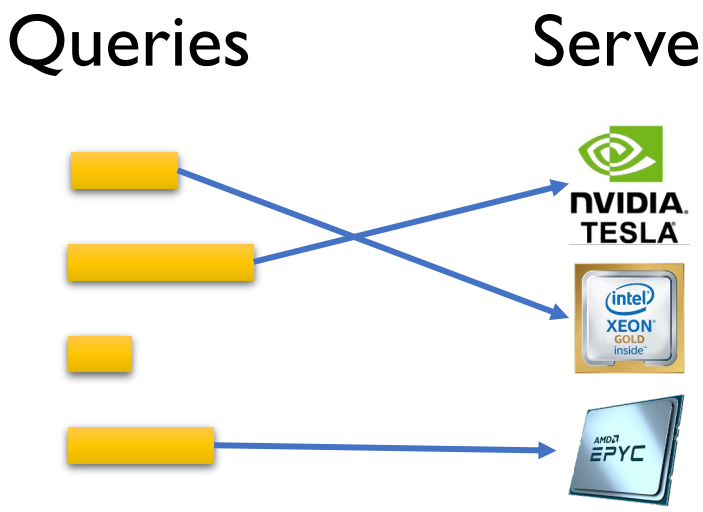
Time till availability

Performance

Heterogeneity coefficient

Heterogeneity coefficient: measures how important a hardware instance is to the system

Formulating Query Distribution as Bipartite Graph Matching Problem



Kairos minimizes resource usage to provide max slack time for the future queries

Table 2: Query distribution optimizer parameters.

List	Description
$L_{i,j}$	Time needed to finish serving Q_i on instance I_j from t_0.
m	Number of queries at time t_0.
n	Number of instances in the configuration.
C_j	Heterogeneity coefficient for instance I_j.
T_{qos}	QoS target latency.
W_i	Query Q_i's time spent waiting in queue before t_0.
$P_{i,j}$	Query-to-instance pairing/assignment matrix.

$$\min_{P} \sum_{i=1}^{m}\sum_{j=1}^{n} C_j(L_{i,j})P_{i,j}$$

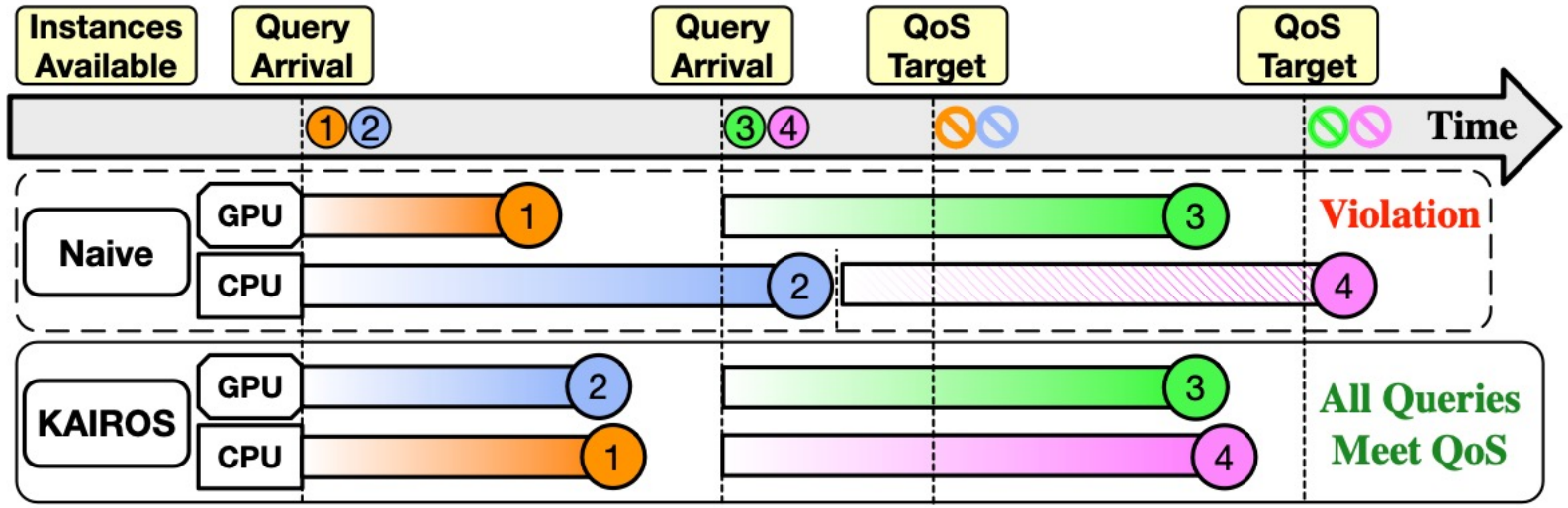
Edge cost

$$\text{s.t. } \forall i,j, \quad (L_{i,j}+W_i)P_{i,j} \le T_{qos},$$

$$\forall i,j, \quad \sum_{i=1}^{m} P_{i,j} \le 1, \sum_{j=1}^{n} P_{i,j} \le 1,$$

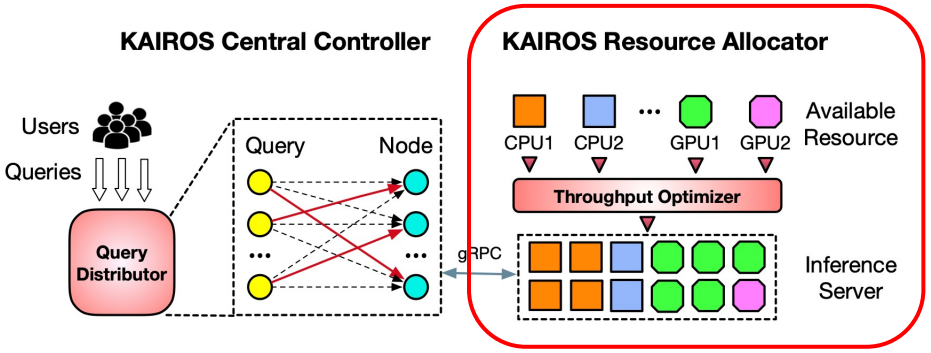
$$\sum_{i=1}^{m}\sum_{j=1}^{n} P_{i,j} \ge min\{m,n\}$$

Intuition behind Kairos' query dispatcher



Kairos matches higher speedup queries to more powerful devices to create more slack time for the future – resulting in lower chance of QoS violation

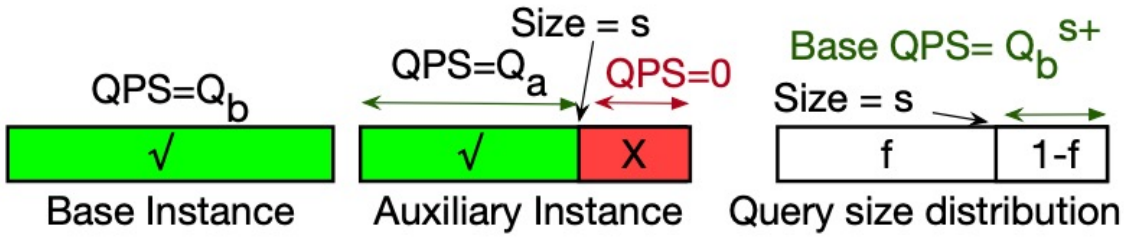
Kairos Resource Allocator



Key idea: rank heterogeneous resource configurations using approximation.
No online evaluation!

Classify the resources as base type or auxiliary type

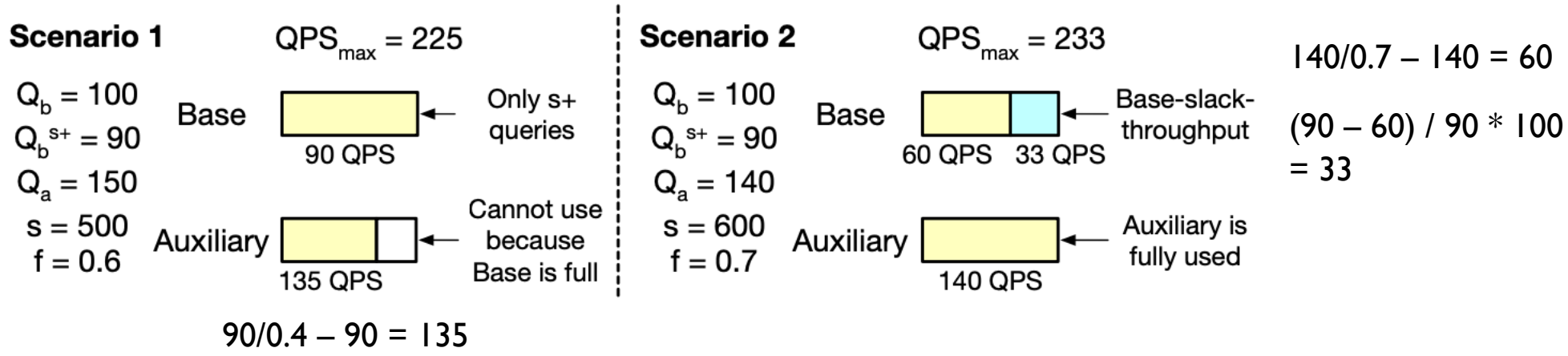
- Base instance type: most performant type, can meet QoS for all queries (usually most expensive)
- Auxiliary instance type: other types, can meet QoS for some queries smaller than certain batch sizes (usually cheaper)



One-Base-One-Auxiliary Example

Approximate the throughput upper bound based on whether base or auxiliary is the bottleneck

- Estimate maximum possible throughput in an unrealistic scenario where all queries are available to us at the beginning, and we can control when each query should arrive – then there is no need to worry about latency interactions with queuing.



Please refer to our paper for the detailed mathematical formulation

Exploiting Approximated Throughput Upper-bound

Approximate the throughput for all possible configurations

- Quick because no online evaluation is needed

Rank the configurations using the approximated throughput

- Kairos: takes the top-10 configuration as a cluster and pick the center (Euclidean distance)
- Kairos+: online evaluation and pruning. Always finds the optimal

Kairos+ algorithm

Algorithm 1: KAIROS+'s pruning-based algorithm for quickly finding optimal configuration.

```
UBs ← Sort all QPS_max high to low
curr_best = 0 // Highest throughput so far
best_config = None
configs ← list of all configs within cost budget
x ← variable representing one configuration
foreach UB(x) in UBs do
    if x ∈ configs then
        eval = f(x) // Actual QPS evaluation.
        if eval > curr_best then
            curr_best = eval
            best_config = x
            Filter all c out of configs that satisfies
              UB(c) ≤ curr_best
        end
        Prune away all sub-configs. of x from configs
    end
end
return curr_best, best_config
```

Experimental Methodology

Setup

- AWS cloud instances
- Mix of CPU and GPU instances

Metrics

- Maximize Throughput (Queries-Per-Second, QPS)
- Fixed cost budget

Instance Type	Instance Class	Price ($/hr)
g4dn.xlarge	GPU Accelerated Computing	0.526
c5n.2xlarge	Compute Optimized CPU	0.432
r5n.large	Memory Optimized CPU	0.149
t3.xlarge	General Purpose CPU	0.1664

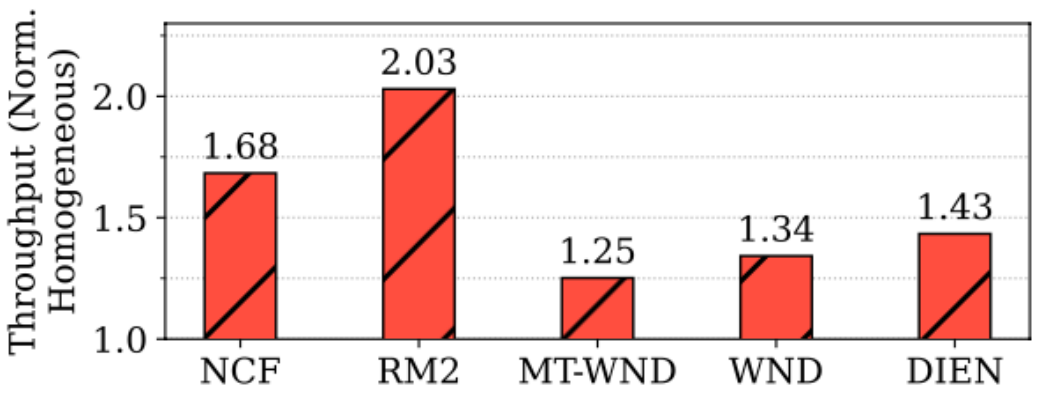
Workloads

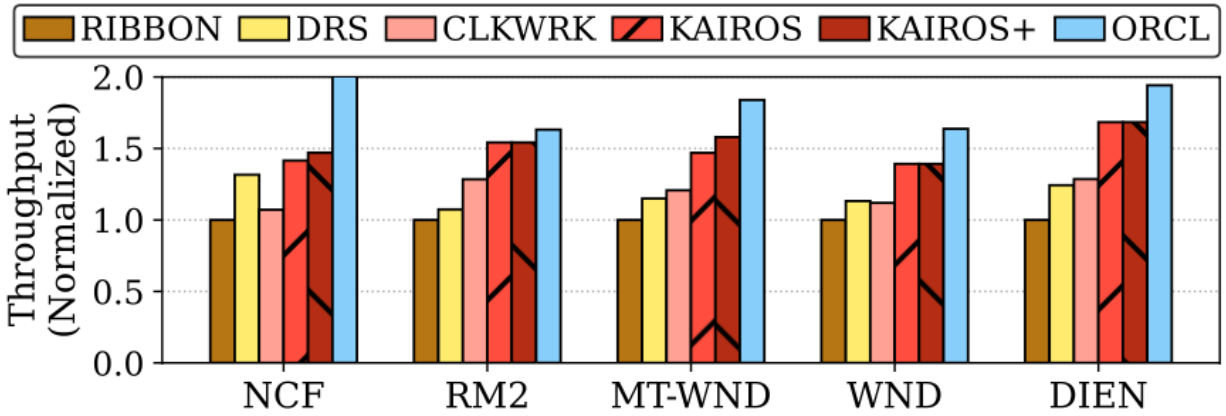
- NCF: movie recommendation (5ms)
- RM2: Facebook model (350ms)
- WND: Google App (25ms)
- MT-WND: YouTube Video (25ms)
- DIEN: Alibaba E-commerce (35ms)

Schemes

- Ribbon [SC'21]
- DeepRecSys (DRS) [ISCA'20]
- ClockWork [OSDI'20]
- Oracle

Kairos is significantly more effective than homogenous configurations and prior state of the art solutions.

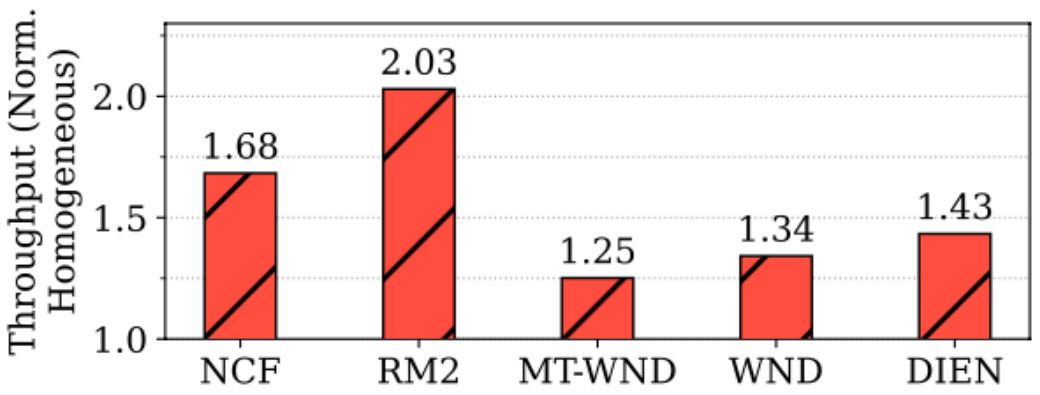


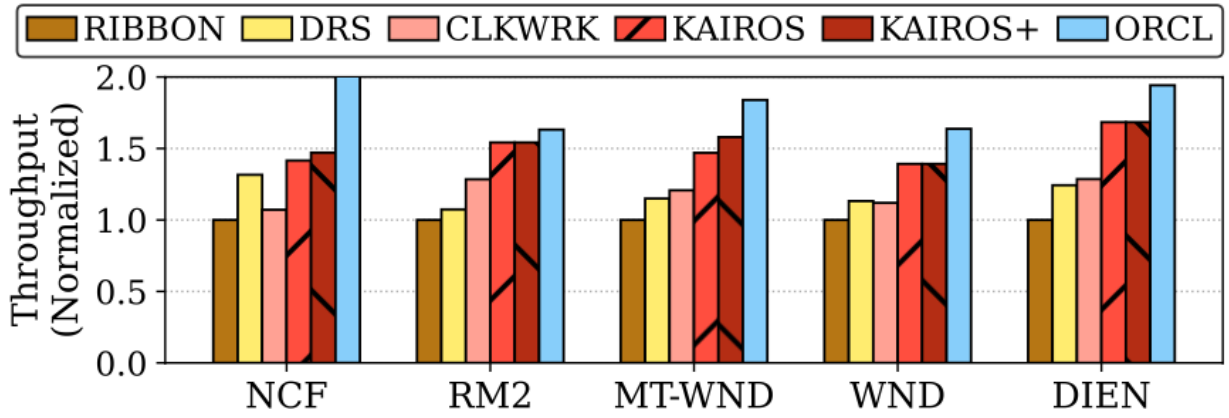


More than 1.25x throughput than the homogenous, QoS-honoring configurations under a cost budget.

Closet-to-optimal effectiveness, consistently across all models.

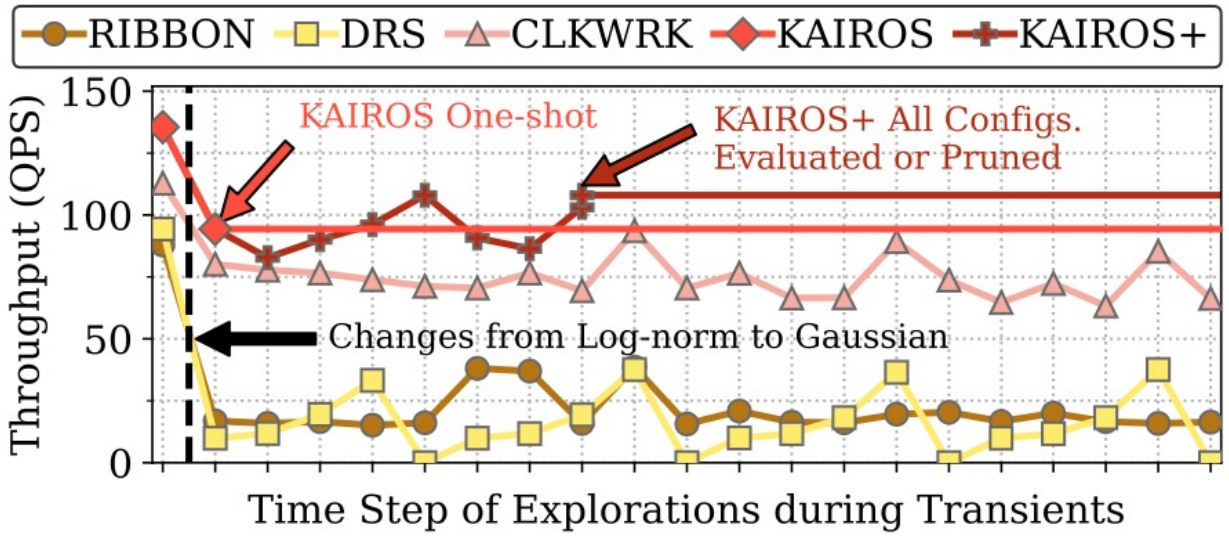
Kairos is significantly more effective than homogenous configurations and prior state of the art solutions.



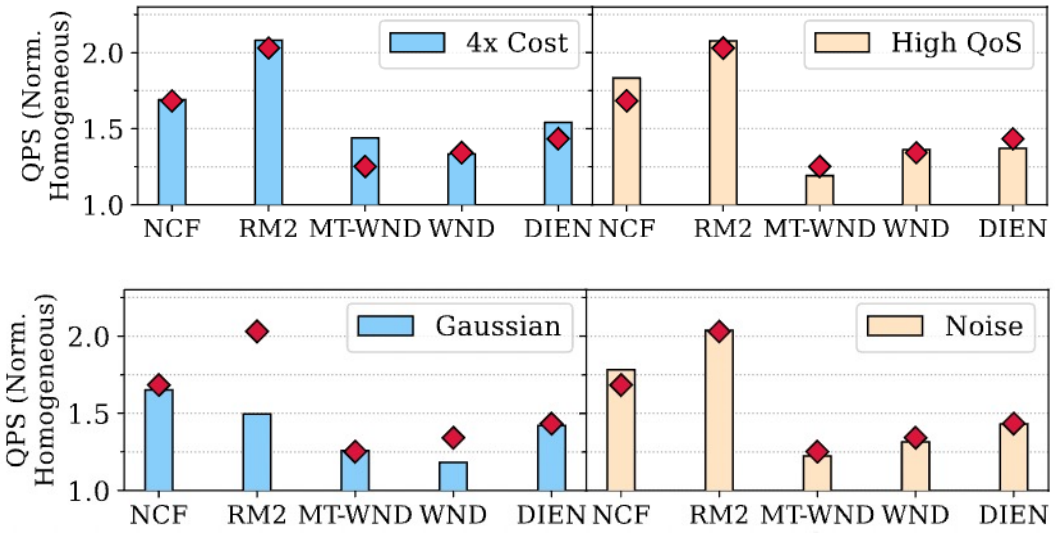


Kairos uses novel approximation method to find the near-optimal configuration in one shot. In contrast, prior methods are given competitive advantage to use the best configuration derived via an extensive offline search.

Kairos adapts quickly and effectively to load changes, and is robust to parameters.



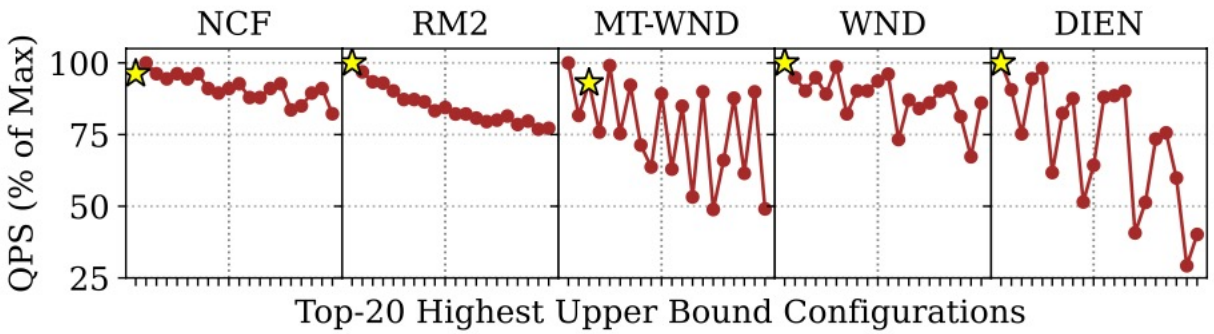
Adaption to load change distributions.



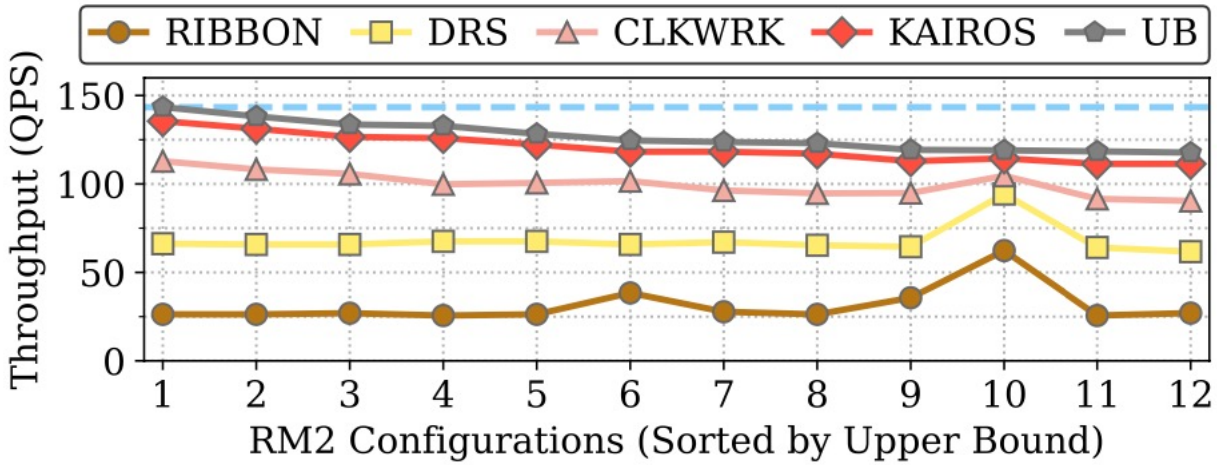
Sensitivity to cost budget, QoS target, noisy latency measurements, etc.

Why does Kairos work so effectively?

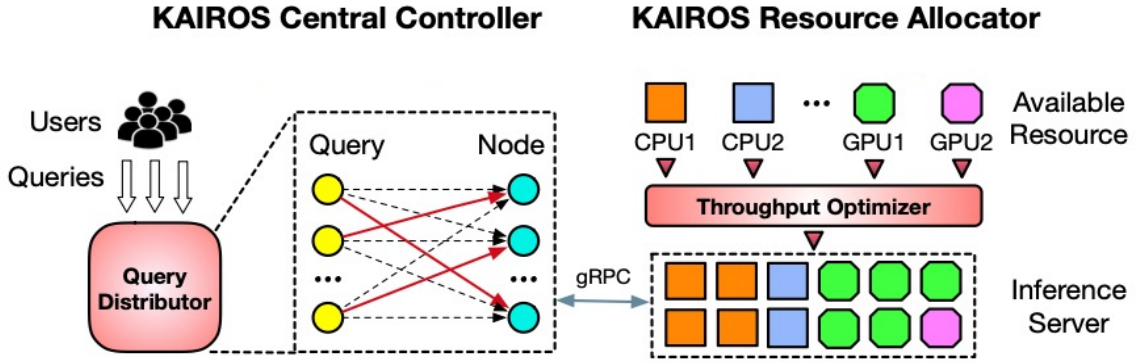
Kairos' novel approximation method provides near-optimal heterogeneous configuration



Kairos' query dispatching mechanism works effectively with approximated near-optimal heterogenous configuration.



Summary of Kairos' Contributions



Contact

Baolin Li

li.baol@northeastern.edu

An open-source ML inference system that achieves high inference throughput and meets QoS under a specified cost budget

A novel approximation method to determine heterogenous configuration without expensive online evaluation of different heterogenous hardware instances.

A novel query-distribution/dispatching mechanism by mapping the problem of query dispatching among heterogenous hardware as the bipartite graph matching problem.

Open-source Artifact

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001, and United States Air Force Research Laboratory Cooperative Agreement Number FA8750-19- 2-1000. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering, or the United States Air Force. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.