AutoSys: The Design and Operation of Learning-Augmented Systems

Chieh-Jan Mike Liang, Hui Xue, Mao Yang, Lidong Zhou, Lifei Zhu, Zhao Lucis Li,
Zibo Wang, Qi Chen, Quanlu Zhang, Chuanjie Liu, Wenjun Dai

Microsoft Research, Peking University, USTC, Bing Platform, Bing Ads

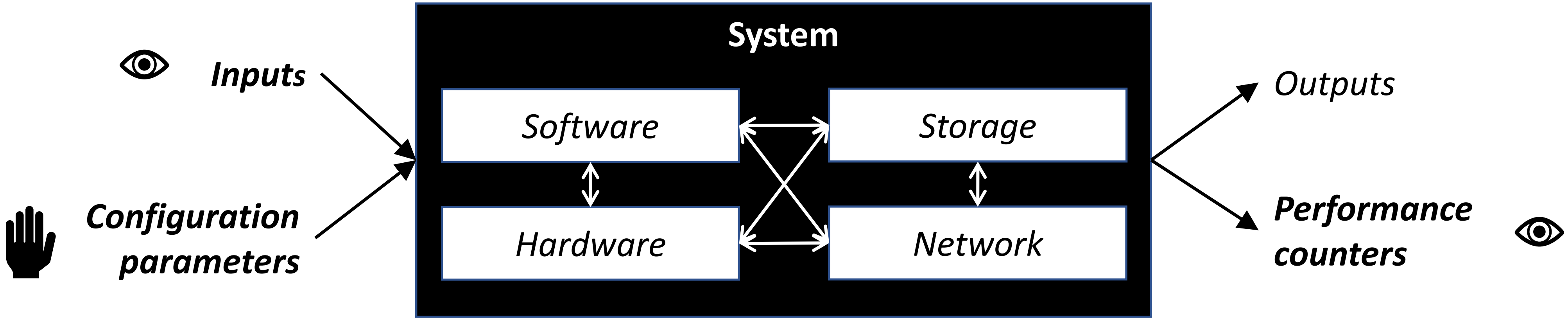
Learning-Augmented Systems

- **Systems whose design methodology or control logic is at the intersection of traditional heuristics and machine learning**
  - Not a stranger to academic communities: “Workshop on ML for Systems”, “MLSys Conference”, …

- **This work reports our years of experience in designing and operating learning-augmented systems in production**
  1. AutoSys framework
  2. Long-term operation lessons

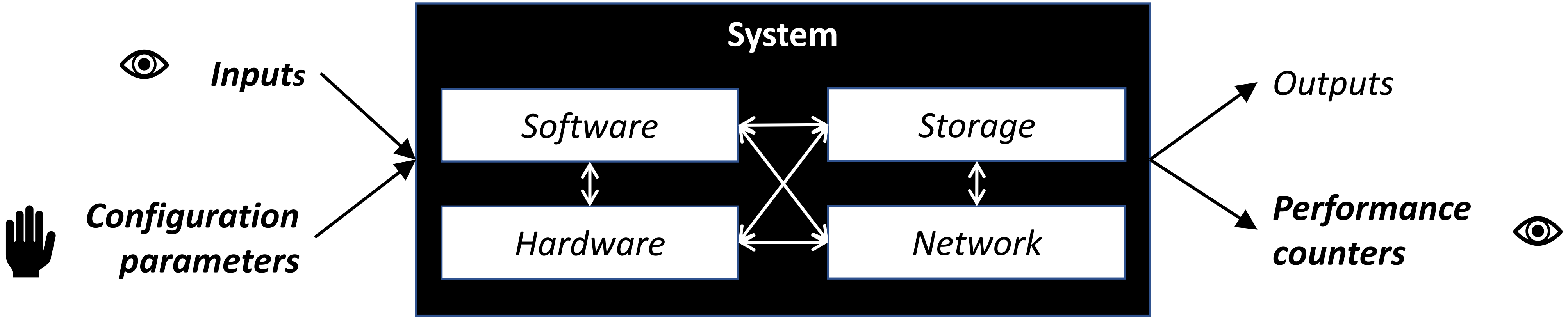
Our Scope in This Paper: Auto-tuning System Config Parameters

- **The problem is simple…**
  - A great application of black-box optimization
  - Find the configuration that best optimizes the performance counters

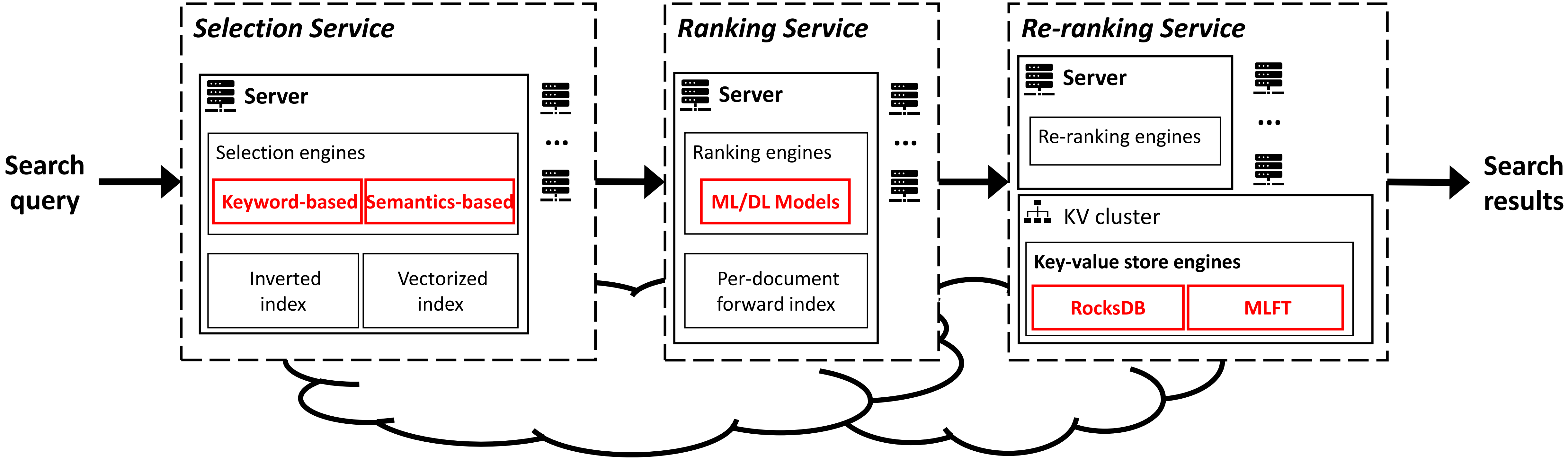


Our Scope in This Paper: Auto-tuning System Config Parameters

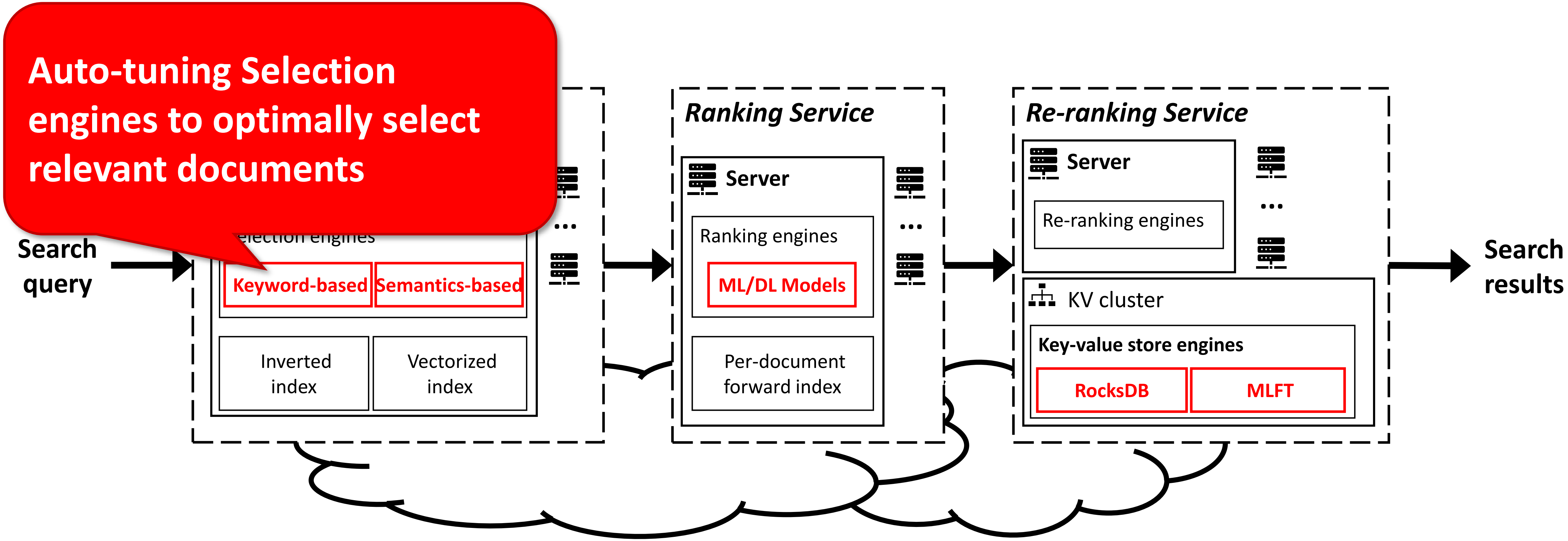
- ***But*, the problem is very difficult for system operators in practice…**
  - Vast system-specific parameter search space
  - Continual optimization based on system-specific triggers



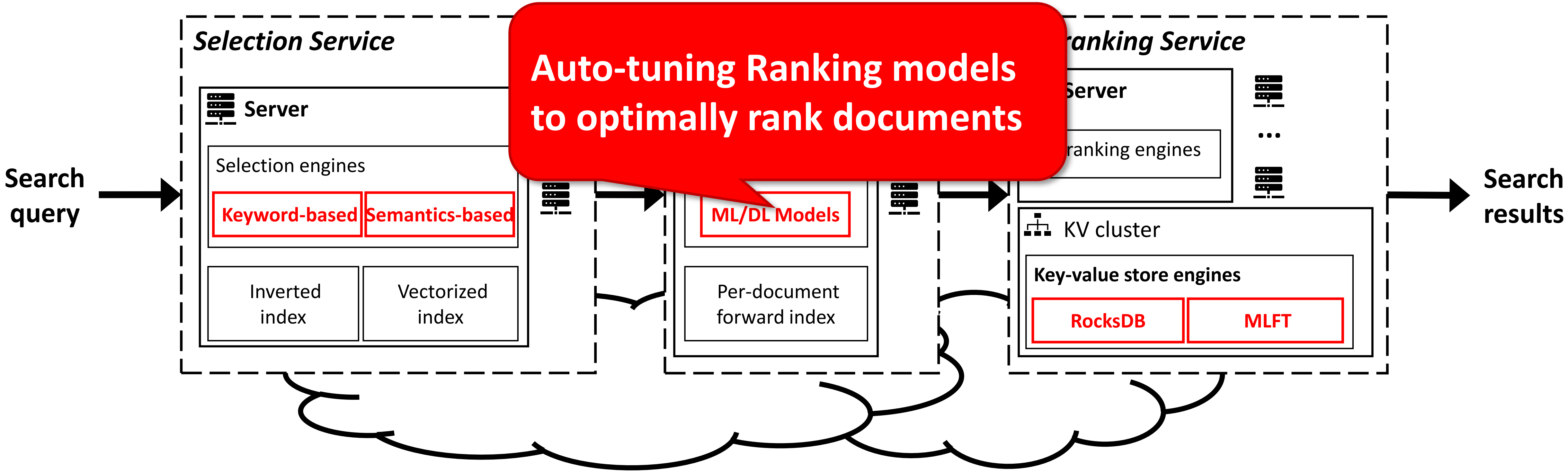
Our Scope in This Paper: Bing Web Search



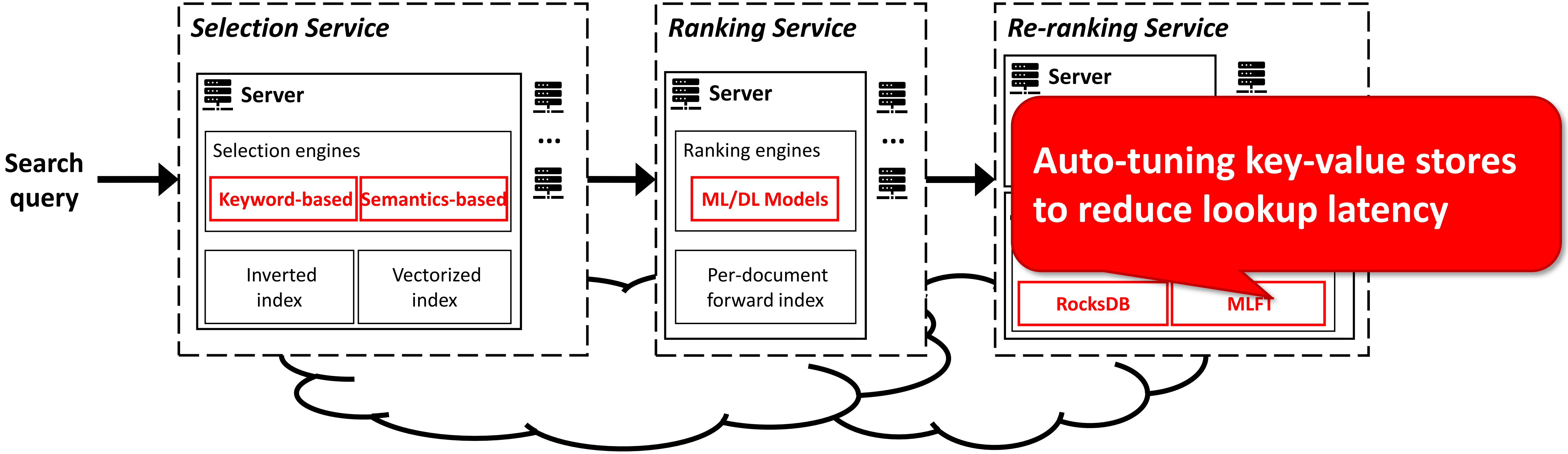
Our Scope in This Paper:
Bing Web Search



Our Scope in This Paper: Bing Web Search



Our Scope in This Paper:
Bing Web Search



Towards A Unified Framework - *AutoSys*

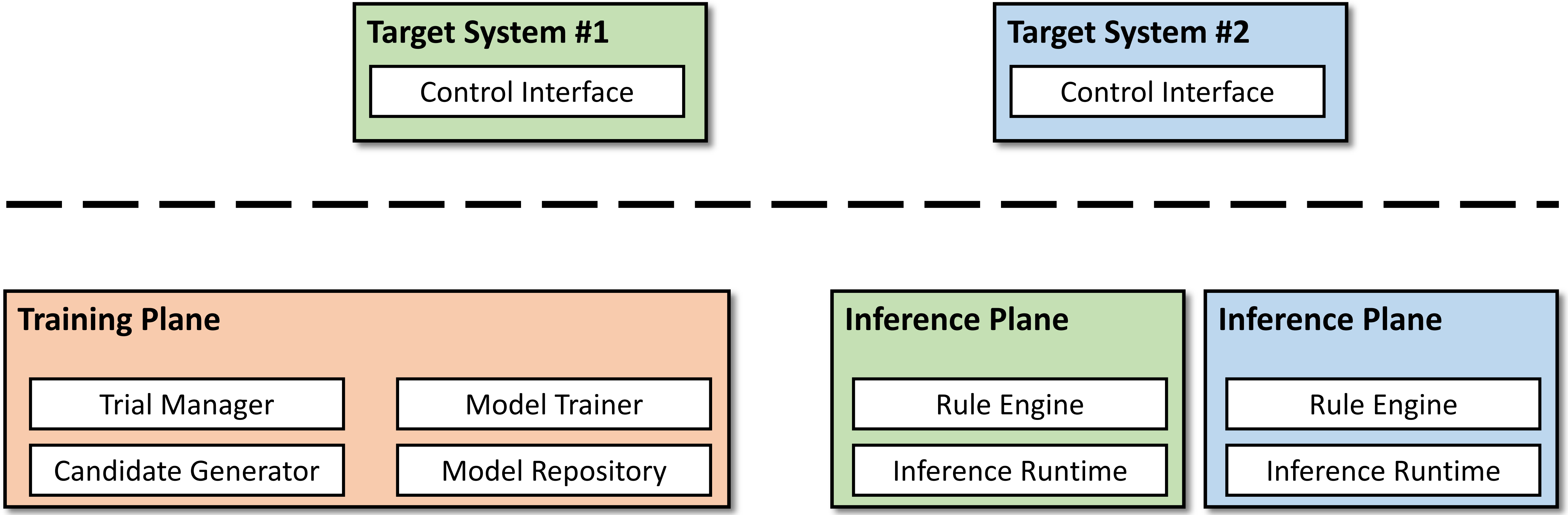
- **Addressing common pain points in building learning-augmented systems**
  - Job scheduling and prioritization for sequential optimization approaches
  - Handling learning-induced system failures (due to ML inference uncertainty)
  - Generality and extensibility
- **Lowering the cost of bootstrapping new scenarios, by sharing data and models**
  - System deployments typically contain replicated service instances
  - Different system deployments can contain the same service
- **Facilitating computation resource sharing**
  - Difficult to provision job resources
  - Jobs in AutoSys are ad-hoc and nondeterministic

Jobs Within AutoSys

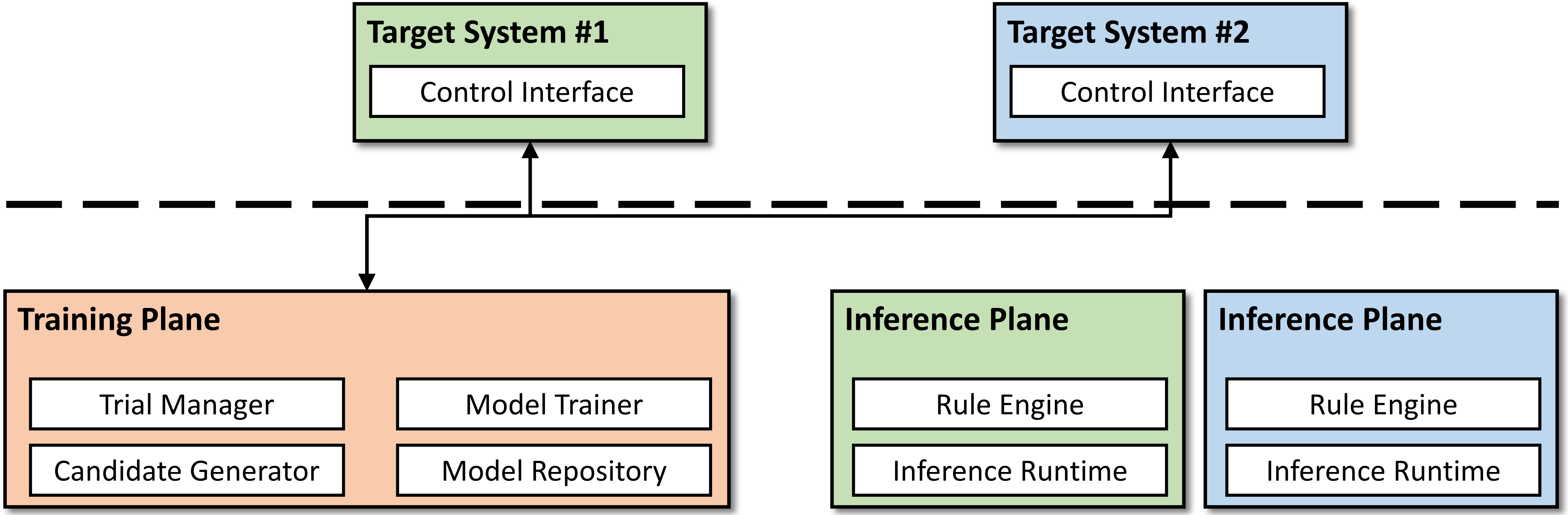
Types	Descriptions	Examples
Tuners	Executes **(1)** ML/DL model training and inferencing, and **(2)** optimization solver	Hyperband, TPE, SMAC, Metis, random search, …
Trials	Executes system explorations	RocksDB, …

- **AutoSys jobs are ad-hoc:**
  - Jobs are triggered in response to system and workload dynamics
- **AutoSys jobs are nondeterministic:**
  - Jobs are spawned as necessary, according to optimization progress at runtime
  - Job completion time depends on system benchmarks and runtime (e.g., cache warmup)

Overview


Target System #1
Control Interface
Target System #2
Control Interface
Training Plane
Trial Manager
Model Trainer
Candidate Generator
Model Repository
Inference Plane
Rule Engine
Inference Runtime
Inference Plane
Rule Engine
Inference Runtime

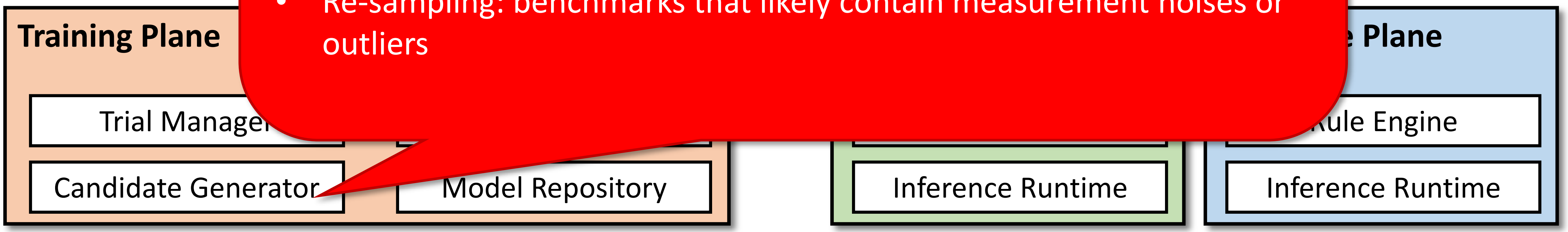
Overview – Learning



Overview – Learning

1.) From assessing current model progress, AutoSys generates benchmark candidates to iteratively improve the model

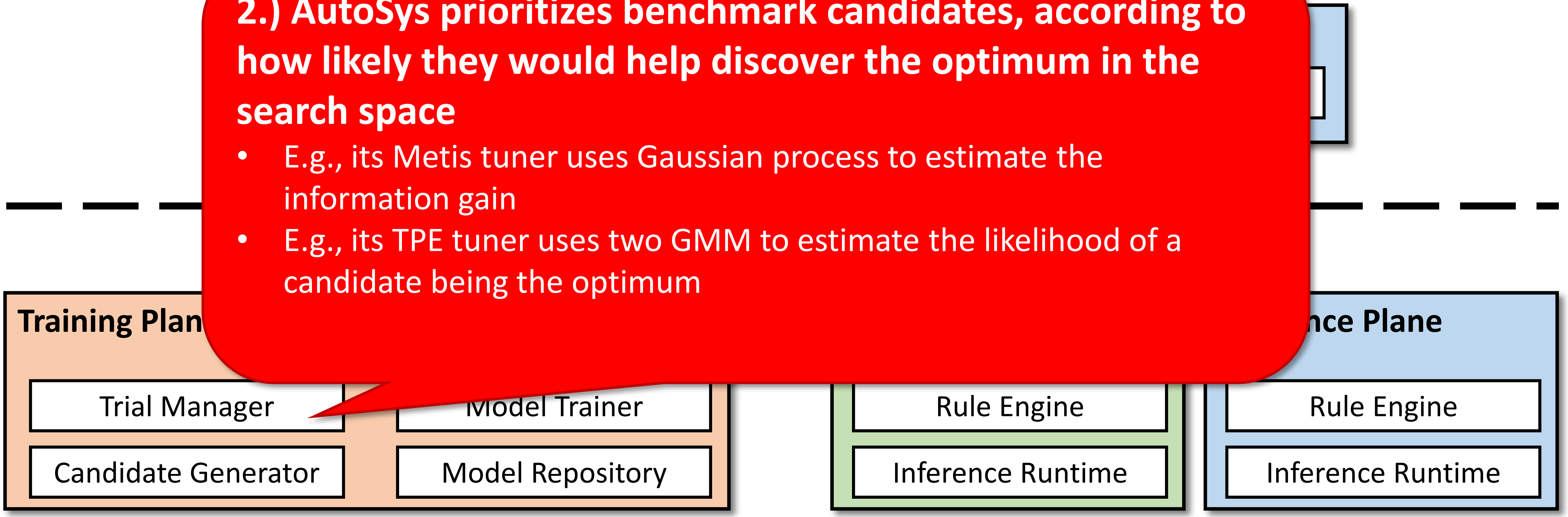
- Exploration: benchmarks that are of high uncertainty
- Exploitation: benchmarks that are likely being optimal
- Re-sampling: benchmarks that likely contain measurement noises or outliers



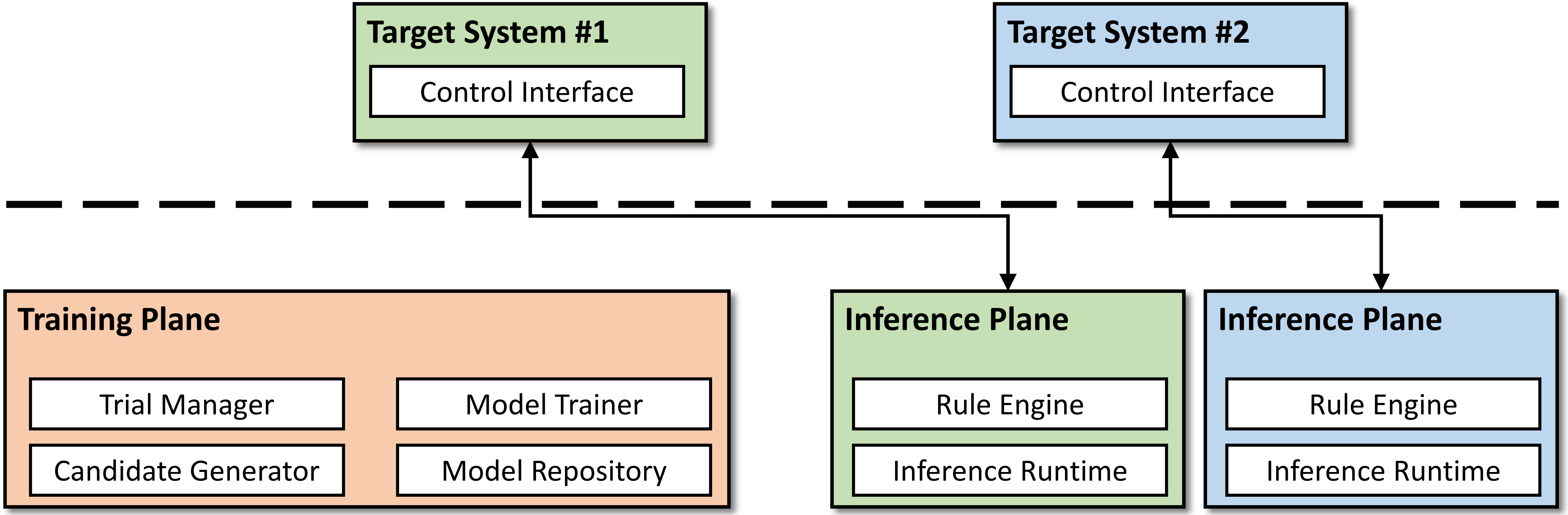
Overview – Learning

2.) AutoSys prioritizes benchmark candidates, according to how likely they would help discover the optimum in the search space

- E.g., its Metis tuner uses Gaussian process to estimate the information gain
- E.g., its TPE tuner uses two GMM to estimate the likelihood of a candidate being the optimum



Overview – Auto-Tuning Actuations



Overview – Auto-Tuning Actuations

3.) As it is difficult to formally verify ML/DL correctness, AutoSys opts to validate ML/DL outputs with a rule-based engine.

- Useful for validating parameter value constraints and dependencies
- Useful for preventing known bad configurations from be applied
- Useful for implementing triggers based on the system’s actuation feedback

Training Plane

Trial Manager

Model Trainer

Candidate Generator

Model Repository

Rule Engine

Inference Runtime

ane

Rule Engine

Inference Runtime

Summary of Production Deployments

	Tuning time	Key results (vs. long-term expert tuning)
Keyword-based Selection Engine (KSE)	1 week	Up to 33.5% and 11.5% reduction in 99-percentile latency and CPU utilization, respectively
Semantics-based Selection Engine (SSE)	1 week	Up to 20.0% reduction in average latency
Ranking Engine (RE)	1 week	3.4% improvement in NDCG@5
RocksDB key-value cluster (RocksDB)	2 days	Lookup latency on-par with years of expert tuning
Multi-level Time and Frequency-value cluster (MLTF)	1 week	16.8% reduction on avg in 99-percentile latency

Long-term Lessons Learned

Higher-than-expected learning costs

- **Various types of system dynamics can frequently trigger re-training**
  - System deployments can scale up/down over time
  - Workloads can drift over time
- **Learning large-scale system deployments can be costly**
  - Testbeds might not match the scale and fidelity of the production environment
  - It is typically infeasible to explore system behavior in the production environment

Long-term Lessons Learned

Pitfalls of human-in-the-Loop

- **Human experts can inject biases into training datasets**
  - E.g., human experts can provide labeled data points for certain search space regions
- **Human errors can prevent AutoSys from functioning correctly**
  - E.g., wrong parameter value ranges

Long-term Lessons Learned

System control interfaces should abstract system measurements and logs to facilitate learning

- **Many systems distribute configuration parameters and error messages over a set of not-well documented files and logs**
- **Many system feedbacks are not natively learnable, e.g., stack traces and core dump**
- **Some systems require customized measurement aggregation and cleaning**

Conclusion

- **This work reports our years of experience in designing and operating learning-augmented systems in production**
  1. AutoSys framework, for unifying the development at Microsoft
  2. Long-term operation lessons

- **Core components of AutoSys are publicly available at https://github.com/Microsoft/nni**

Mike Liang

Systems and Networking Research Group
Microsoft Research Asia

liang.mike@microsoft.com

www.microsoft.com/en-us/research/people/cmliang

Microsoft