Dynamic Resource Allocation for Distributed Dataflows

Lauritz Thamsen

Technische Universität Berlin

04.05.2018

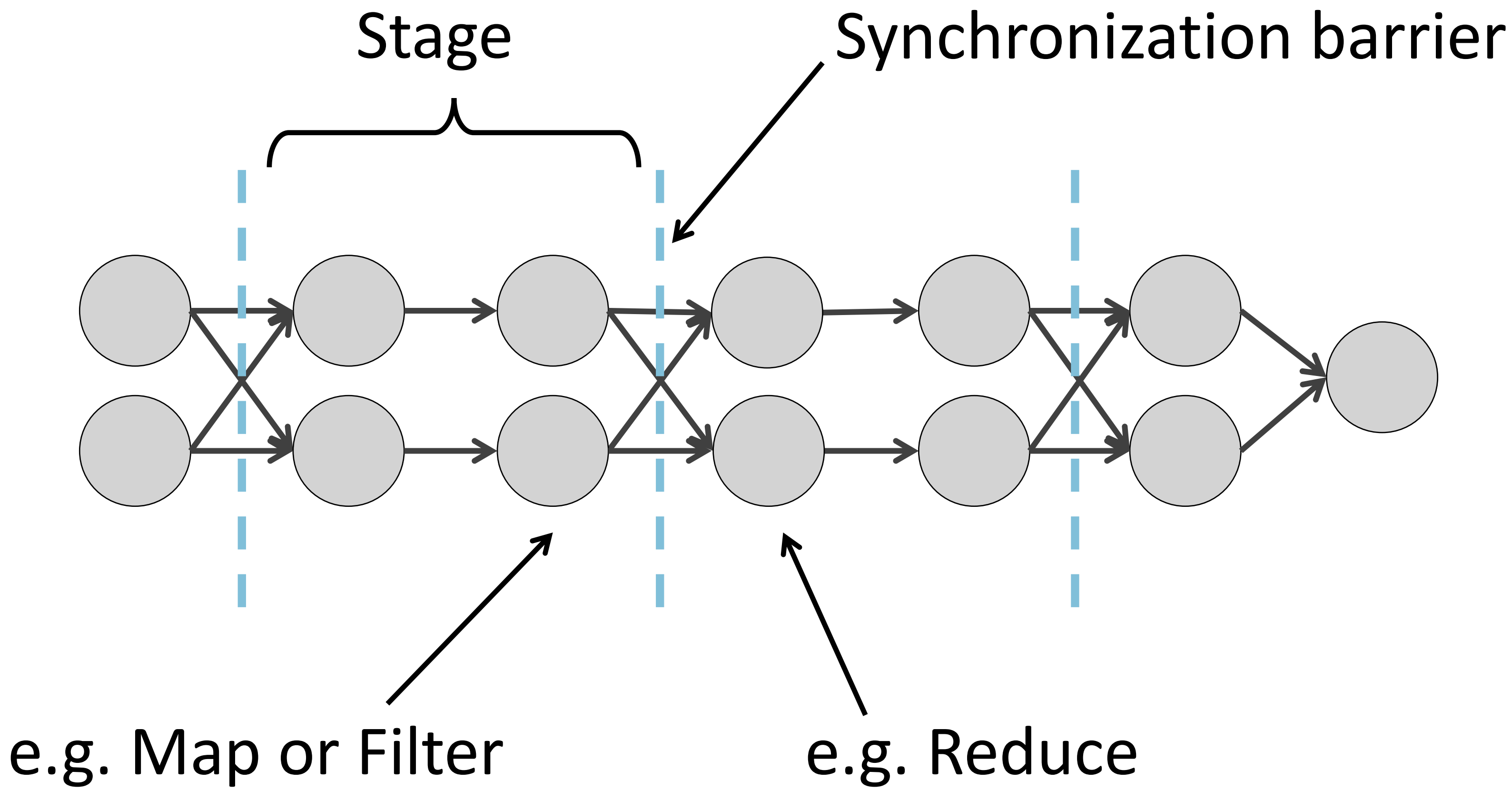
Technische Universität Berlin

Complex and Distributed IT Systems

Distributed Dataflows

- E.g. MapReduce, SCOPE, Spark, and Flink
- Used for scalable processing in many domains, e.g.
  - Search and data aggregation
  - Relational processing
  - Graph analysis
  - Machine learning

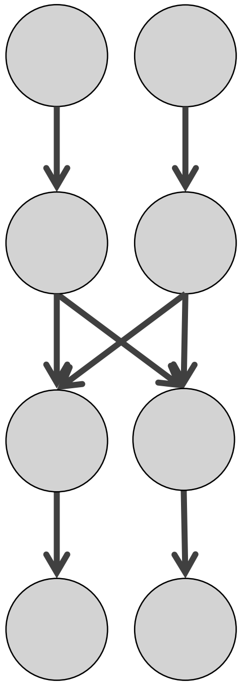
Distributed Dataflow Jobs



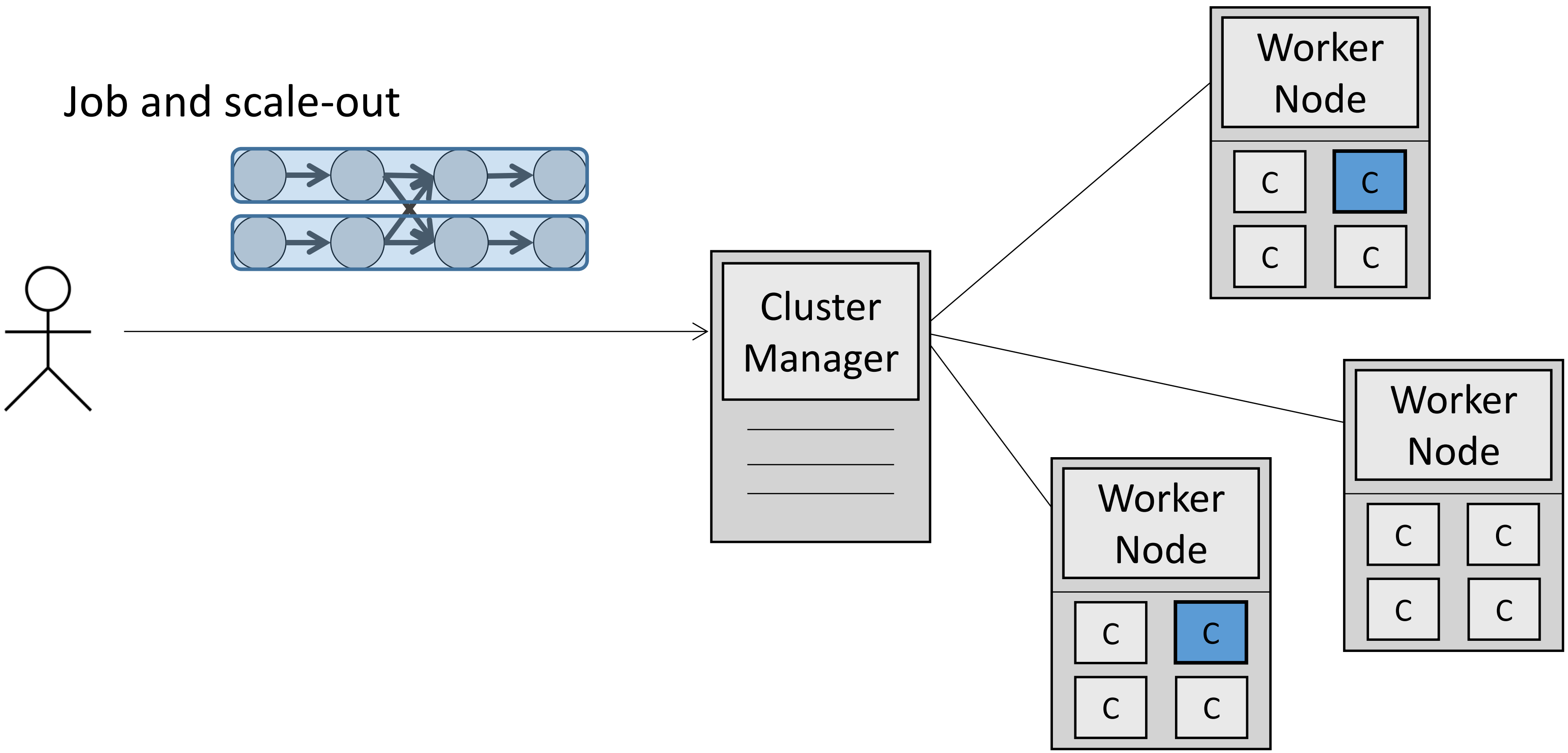
Compared to HPC

- High-level programming abstractions
- Comprehensive frameworks and distributed execution engines

- Usable even without extensive knowledge of parallel and distributed programming



Yet Users Need to Allocate Resources



Constraints of Production Jobs

- Users need to reserve resources for runtime targets:
  - Usability constraints,
  - Service level agreements,
  - Cluster reservation only valid for a certain time

- But do not necessarily understand workload dynamics

Two-Fold Problem

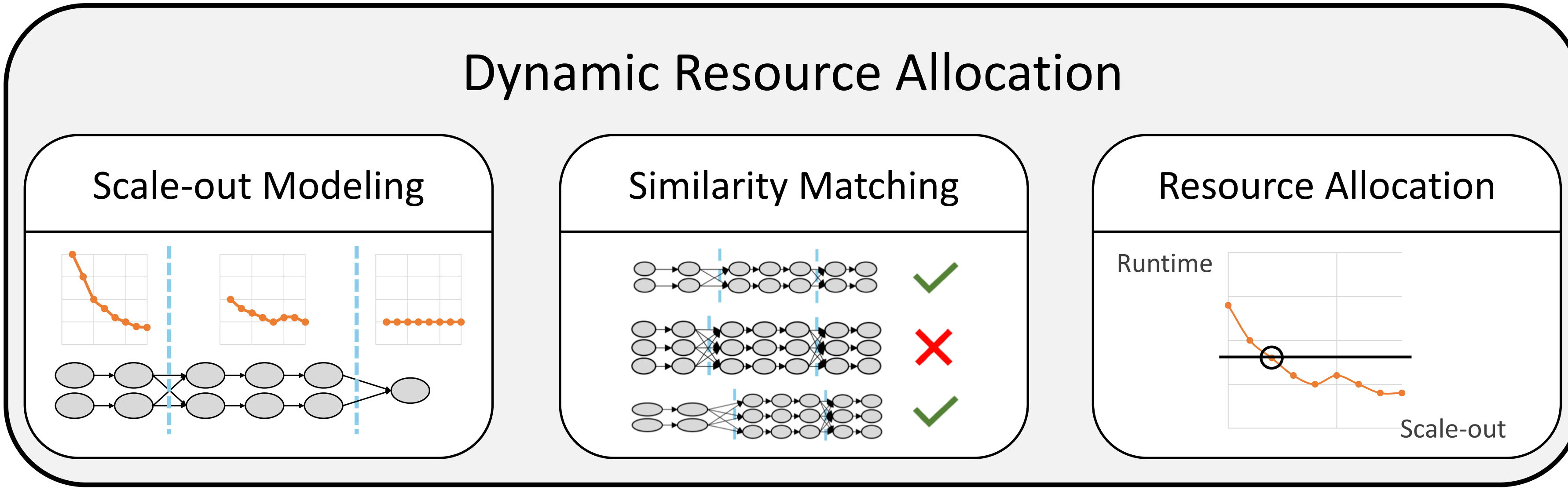
- Runtime Prediction: Estimating performance upfront is difficult as it depends on many factors, e.g. programs, data, systems, and infrastructure

- Runtime Variance: Significant performance variance due to data locality, interference, and failures

Given a runtime target for a recurring batch job,

how can minimal resources be allocated automatically?

Methods & Results

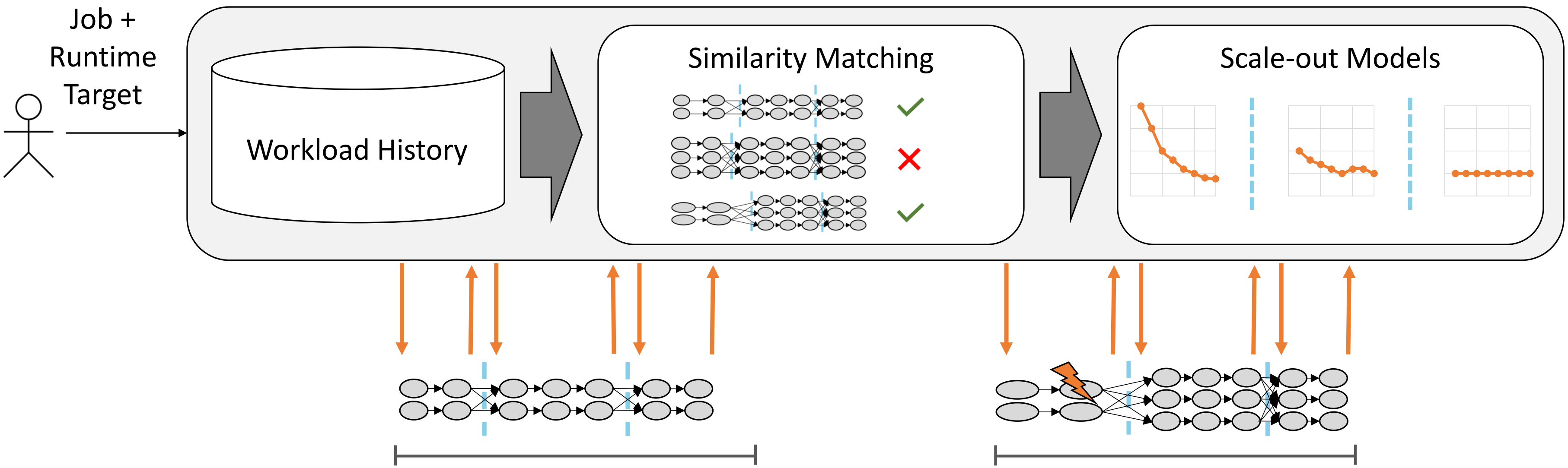


L. Thamsen, I. Verbitskiy, F. Schmidt, T. Renner, and O. Kao. “*Selecting Resources for Distributed Dataflow Systems According to Runtime Targets*”. IPCCC 2016. IEEE.

J. Koch, L. Thamsen, F. Schmidt, and O. Kao. “*SMiPE: Estimating the Progress of Recurring Iterative Distributed Dataflows*”. PDCAT 2017. IEEE.

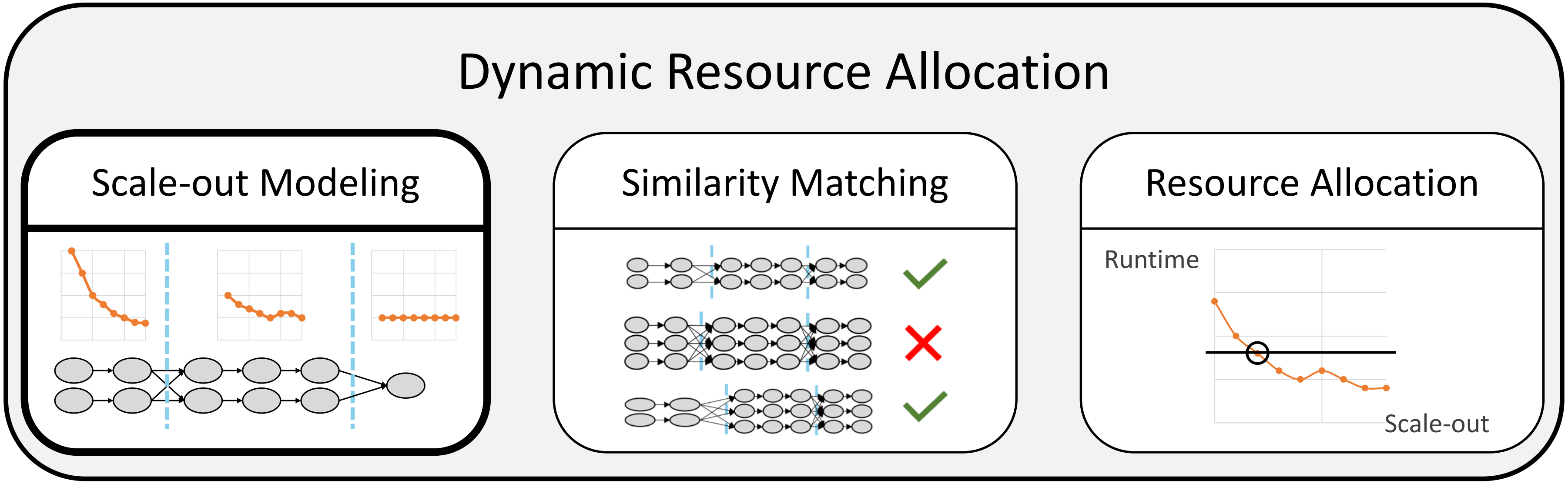
L. Thamsen, I. Verbitskiy, J. Beilharz, T. Renner, A. Polze, and O. Kao. “*Ellis: Dynamically Scaling Distributed Dataflows To Meet Runtime Targets*”. CloudCom 2017. IEEE.

Dynamic Resource Allocation



Assumptions

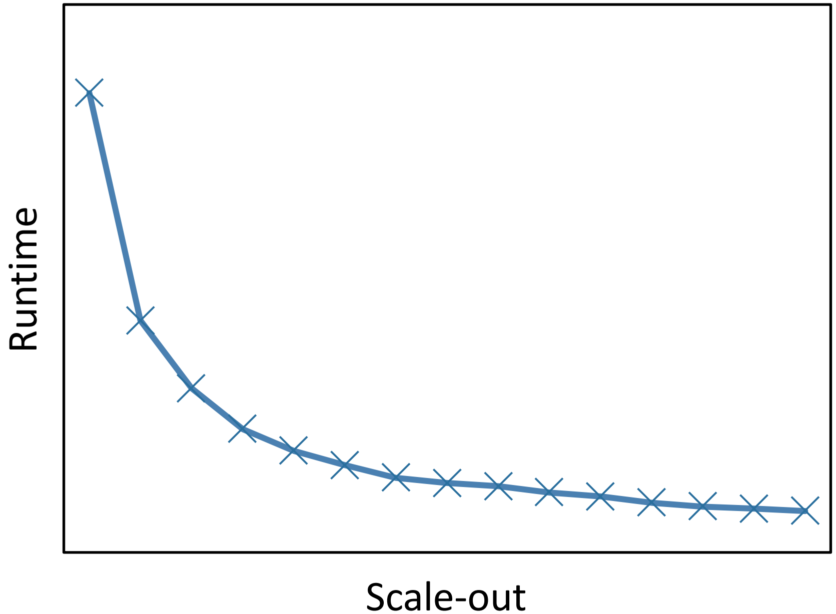
- Distributed dataflow jobs with multiple stages
- Dedicated homogeneous clusters
- Recurring batch processing jobs


Dynamic Resource Allocation
Scale-out Modeling
Similarity Matching
Resource Allocation
Runtime
Scale-out

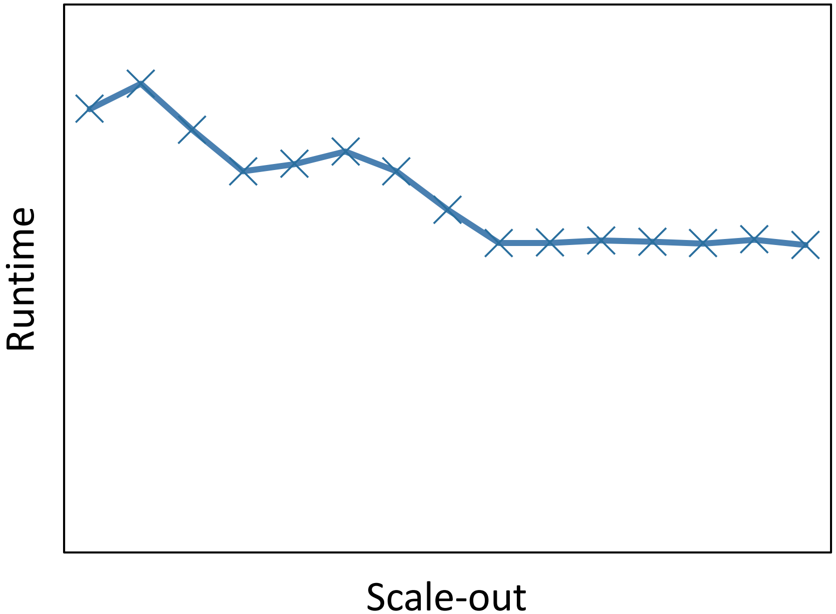
Scale-out Modeling

- Find a function that predicts the runtimes of a job based on previous runs

Example Job A:



Example Job B:



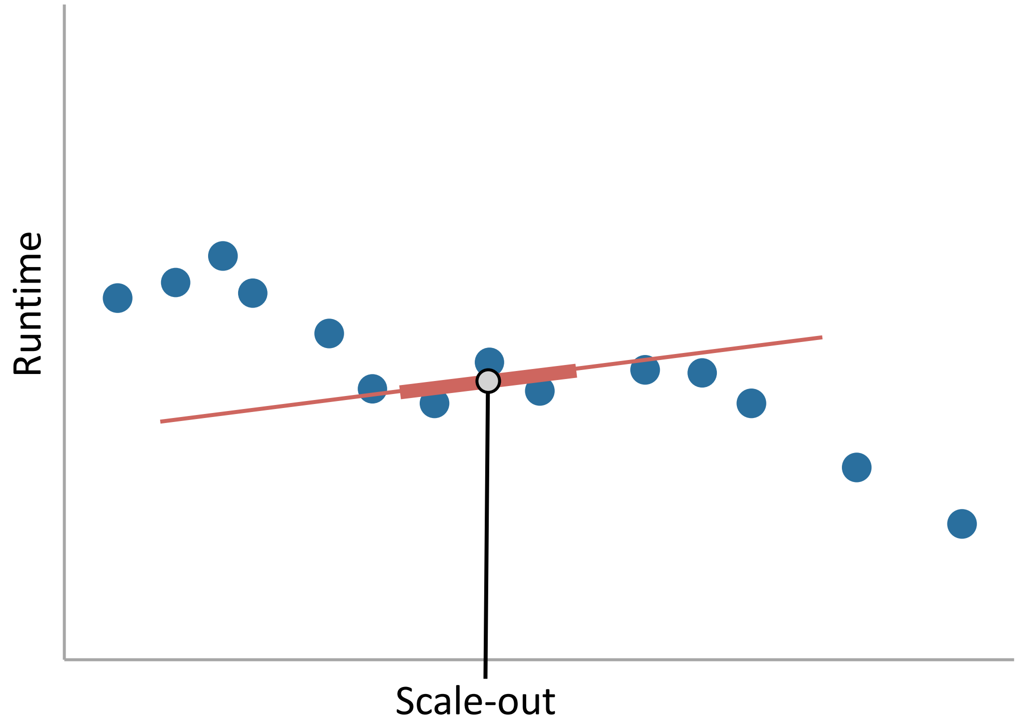
Parametric Regression

- Non-Negative Least Squares (NNLS) for a model of distributed data-parallel processing [Ernest]

$$runtime = \underbrace{\theta_0}_{\text{Sequential portion}} + \overbrace{\theta_1 \cdot \left(\frac{1}{containers}\right)}^{\text{Parallel portion}} + \underbrace{\theta_2 \cdot \log(containers)}_{\text{Tree-like communication patterns}} + \overbrace{\theta_3 \cdot containers}^{\text{Collect-like communication patterns}}$$

Non-parametric Regression

- Assuming locally defined behavior with Local Linear Regression (LLR)



Automatic Model Selection

- Goal: model with high accuracy when possible, otherwise use a robust fallback

- Approach:
  - Use parametric model for extrapolation
  - Dynamically select prediction model for interpolation with k-fold cross-validation

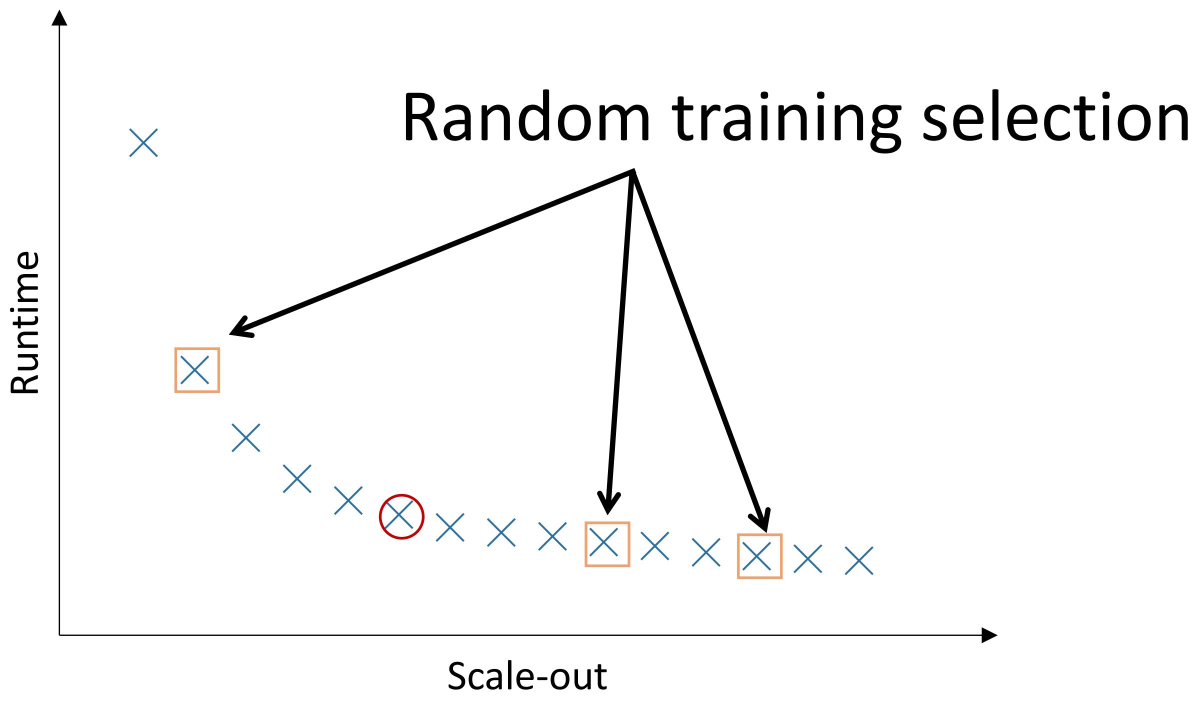
Evaluation Setup

- Using up to 60 commodity nodes (8 cores, 16 GB RAM) and exemplary distributed dataflow jobs

Job	System	Dataset	Input Size
WordCount	Flink	Wiki	250 GB
TPC-H Query 10	Flink	Tables	200 GB
K-Means	Flink	Points	50 GB
Grep	Spark	Wiki	250 GB
SGD	Spark	Features	10 GB
PageRank	Spark	Graph	3.4 GB

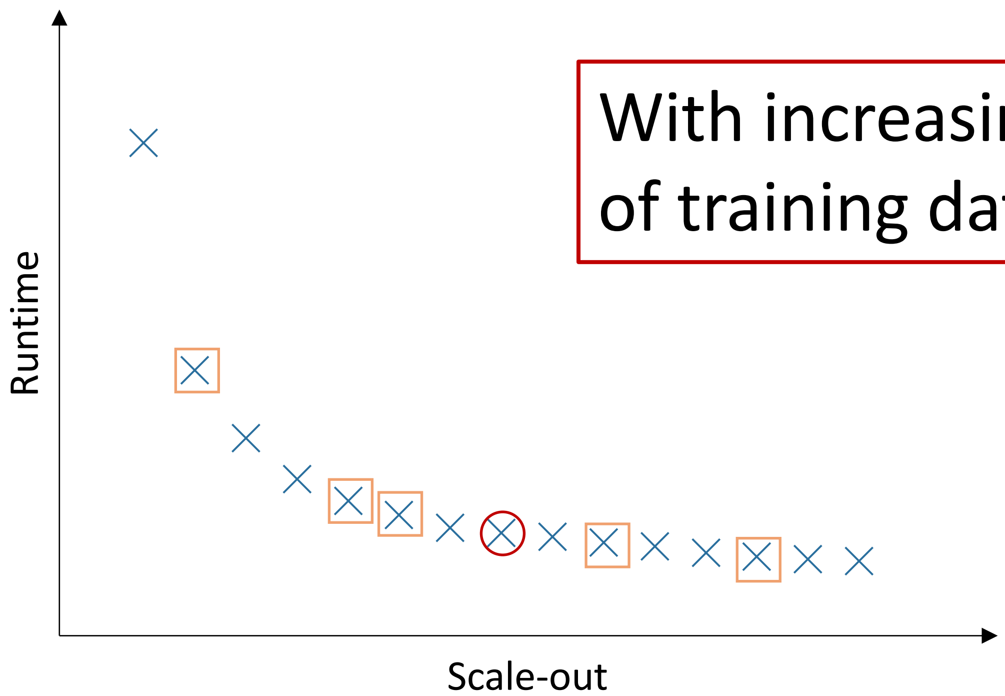
Cross-Validation of Models

- Repeated random sub-sampling



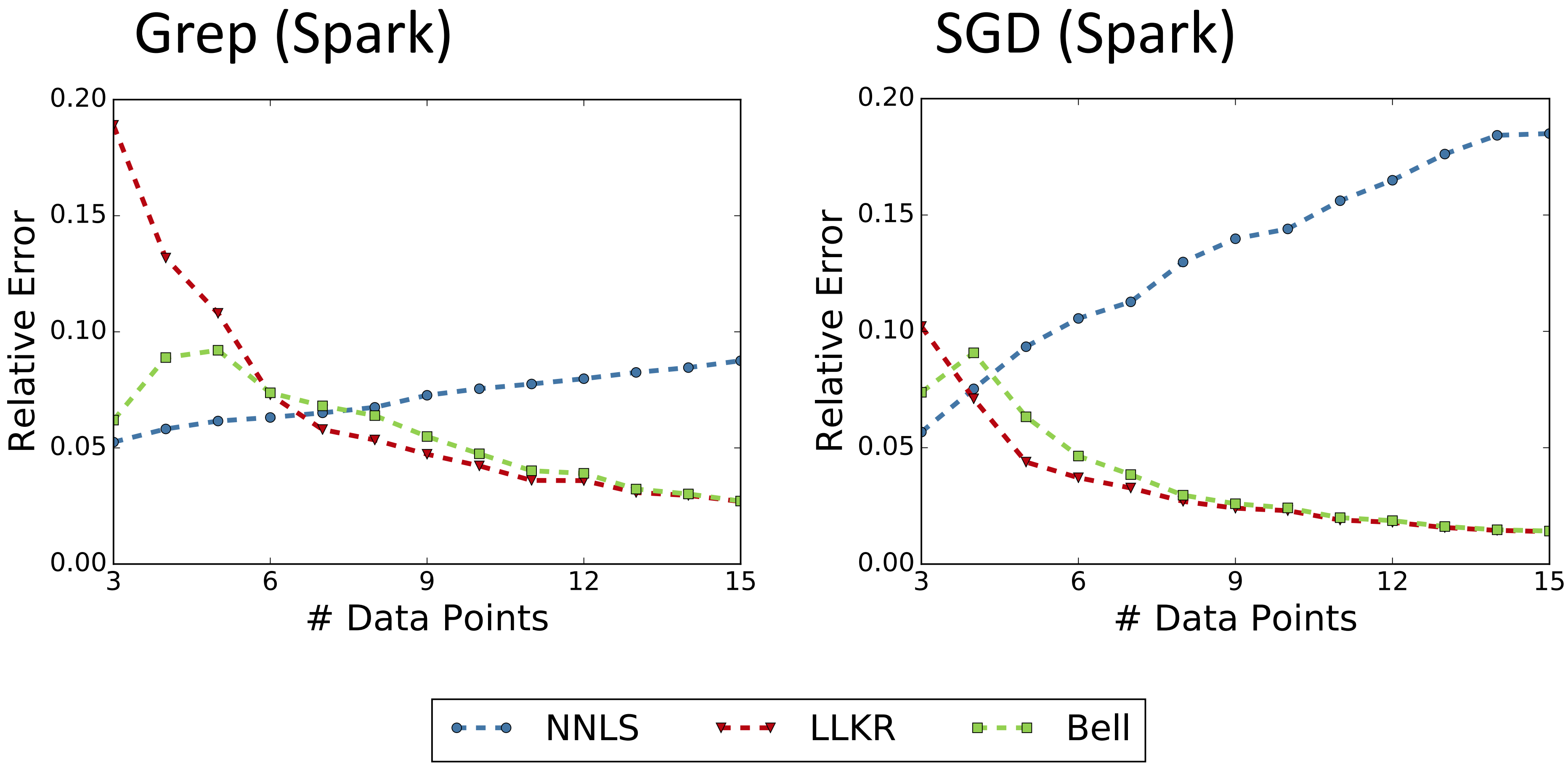
Cross-Validation of Models

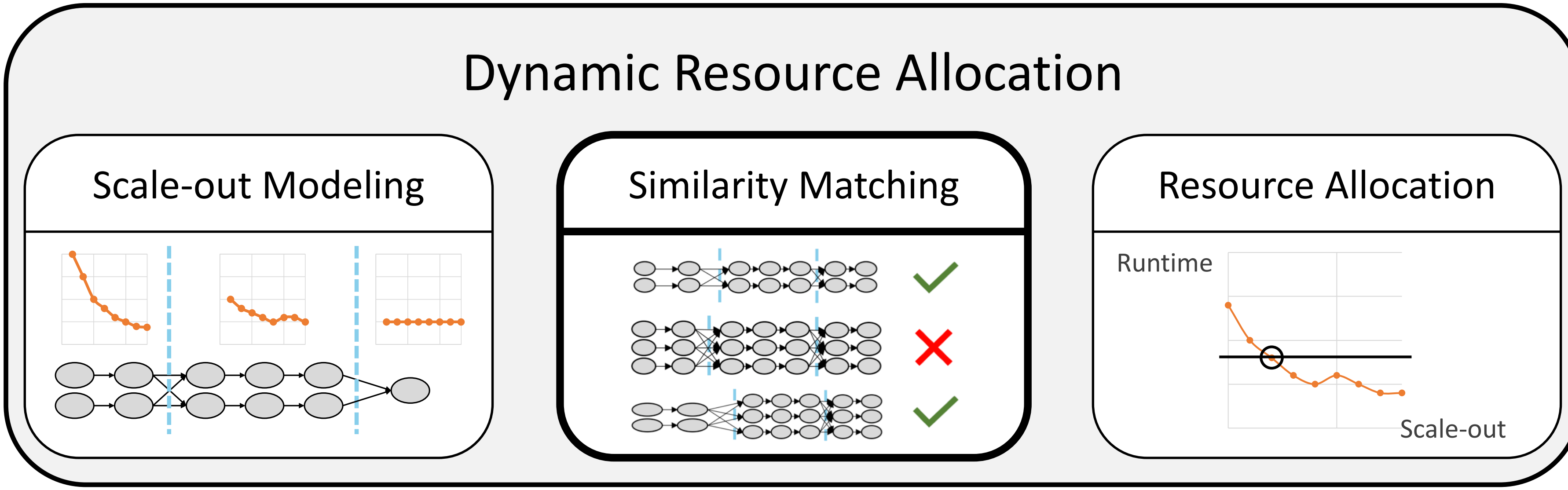
- Repeated random sub-sampling



With increasing amounts of training data, n = …

Relative Prediction Error





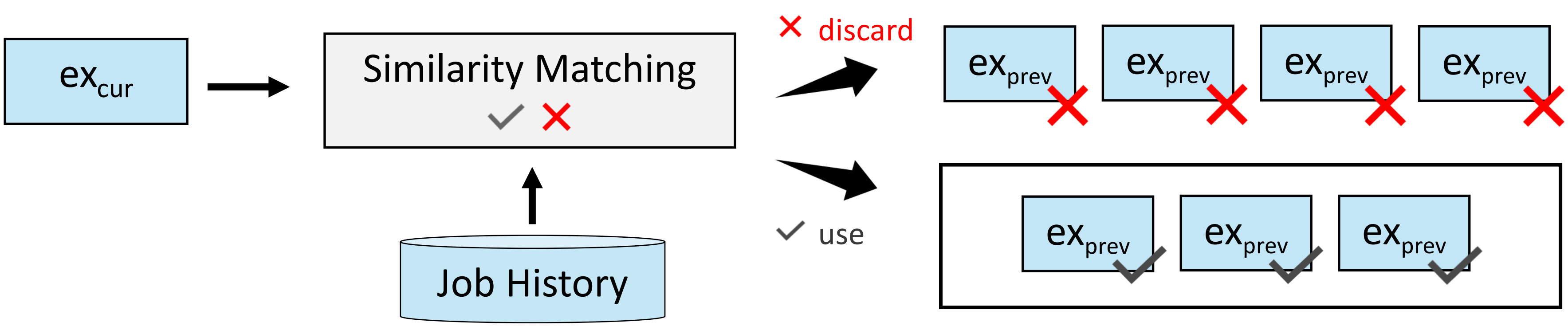
Addresses

Problem of Runtime Prediction

Problem of Runtime Variance

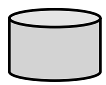
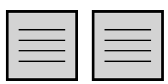
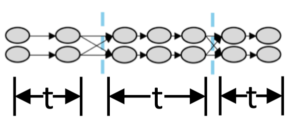
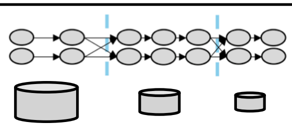
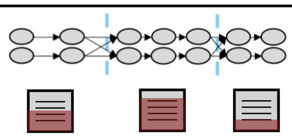
Select Executions for Model Training

- Recurring jobs that run on updated data, code, and parameters, yielding different runtimes
- Idea: Similar executions → similar runtimes

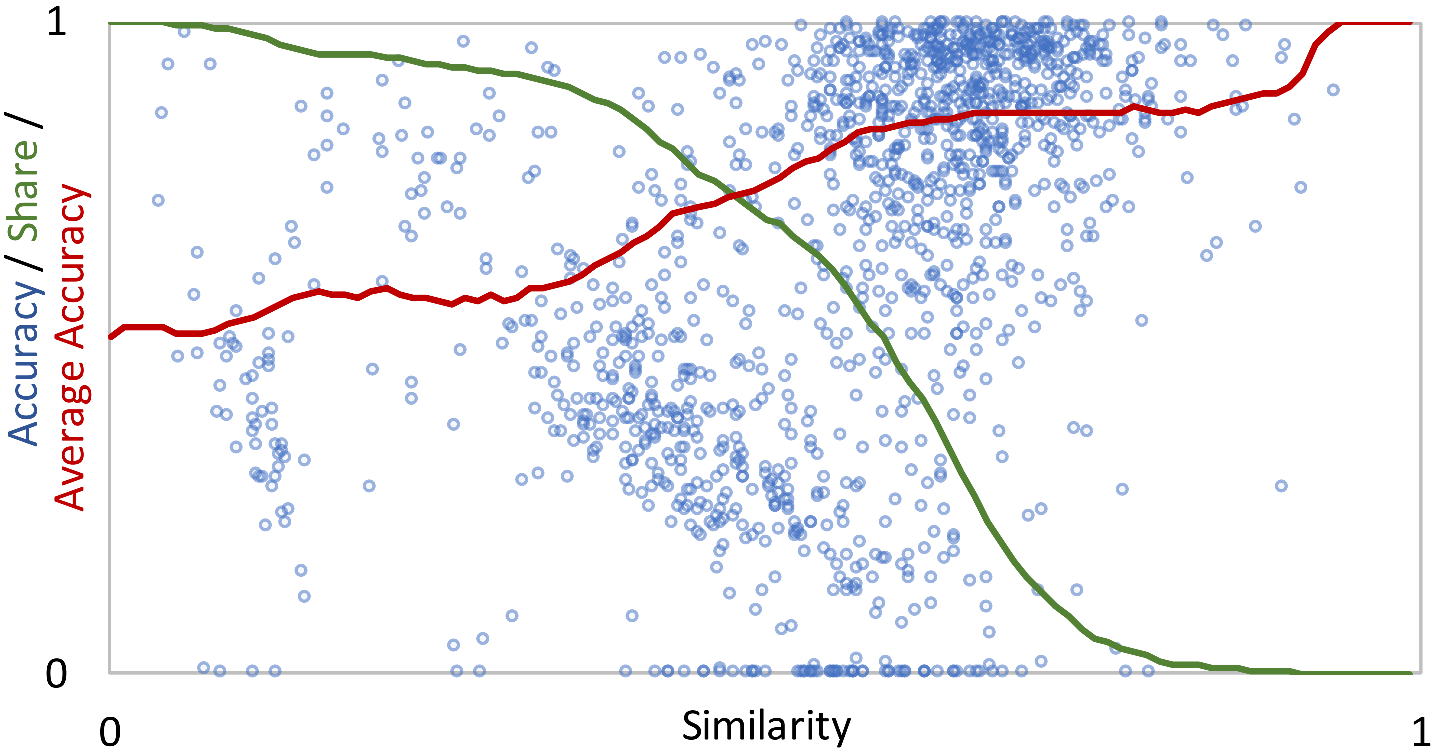


Different Similarity Measures

- Functions s $(ex_{current}, ex_{previous}) \in [0,1]$

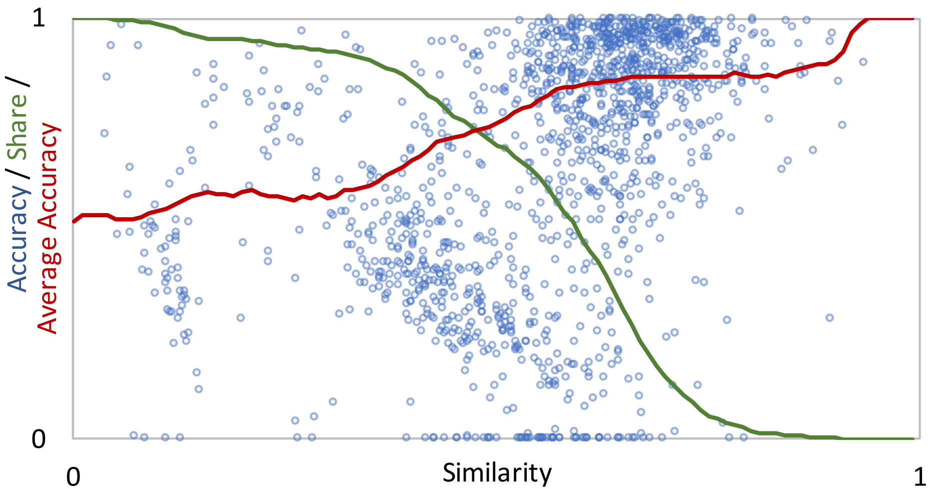
Measure		Type
	Input size	Offline
	Scale-out	
	Runtime of stages	Online
	Convergence	
	Resource utilization	

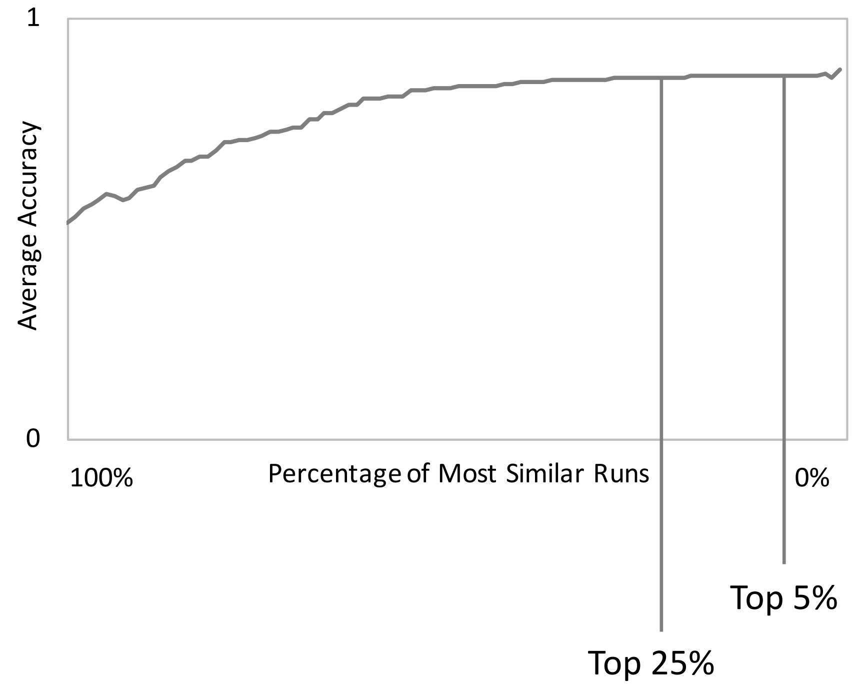
Evaluation of Similarity Measures



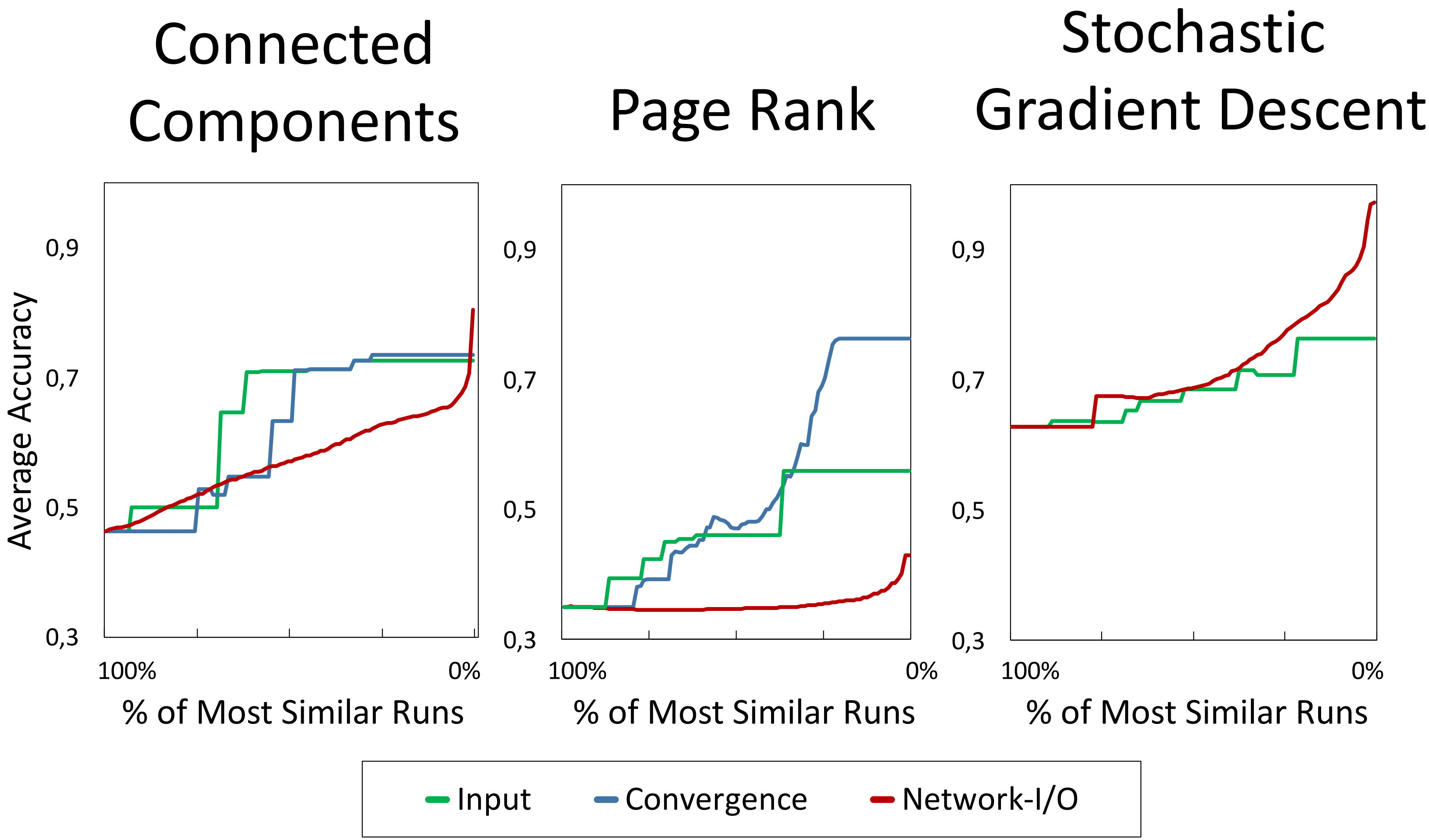
Normalized Similarity Quality

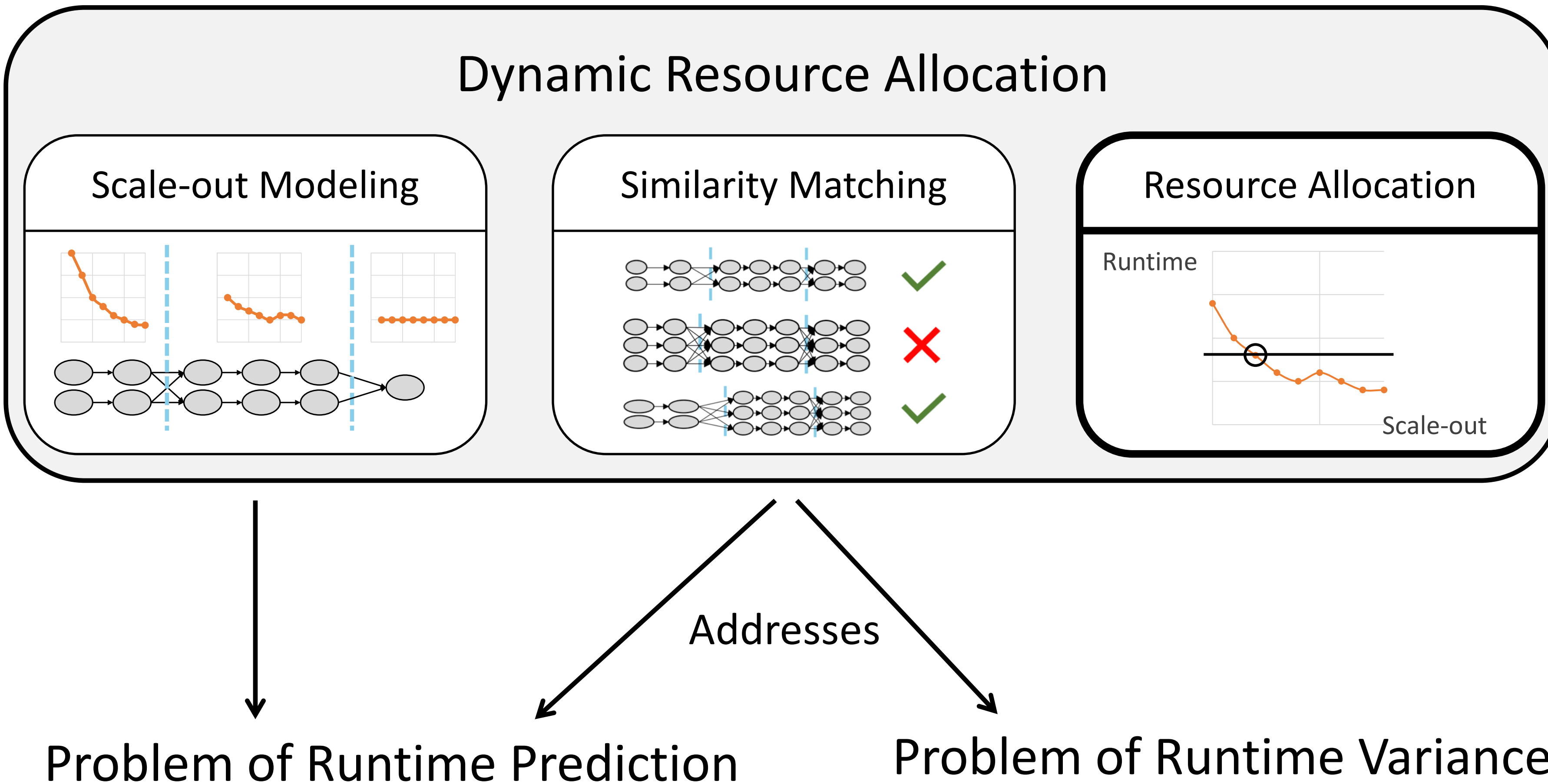
- Accuracy normalized over share of selected points





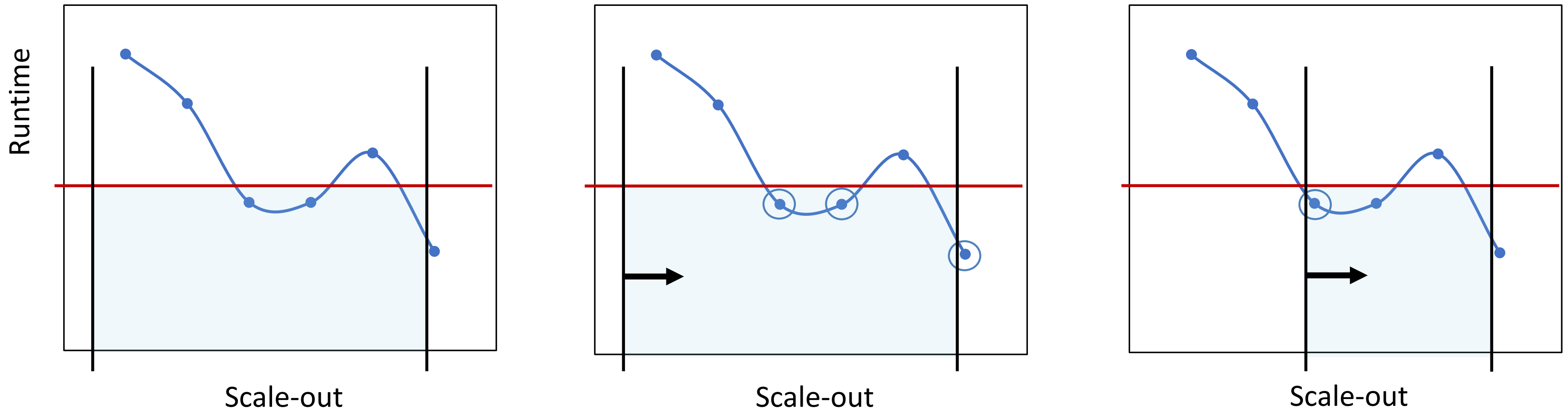
Evaluation of Similarity Measures



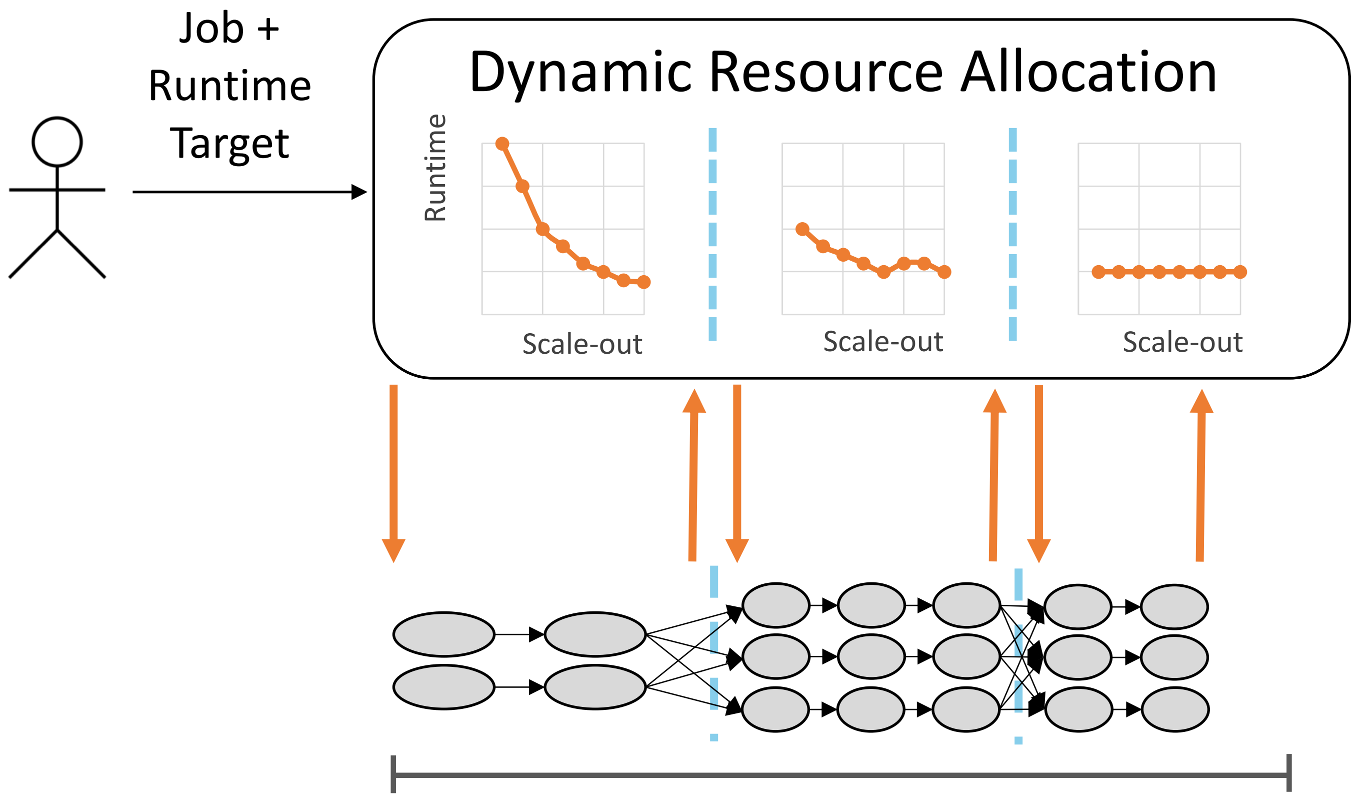

Dynamic Resource Allocation
Scale-out Modeling
Similarity Matching
Resource Allocation
Runtime
Scale-out
Addresses
Problem of Runtime Prediction
Problem of Runtime Variance

Resource Allocation

- Search for minimal scale-out for a runtime target

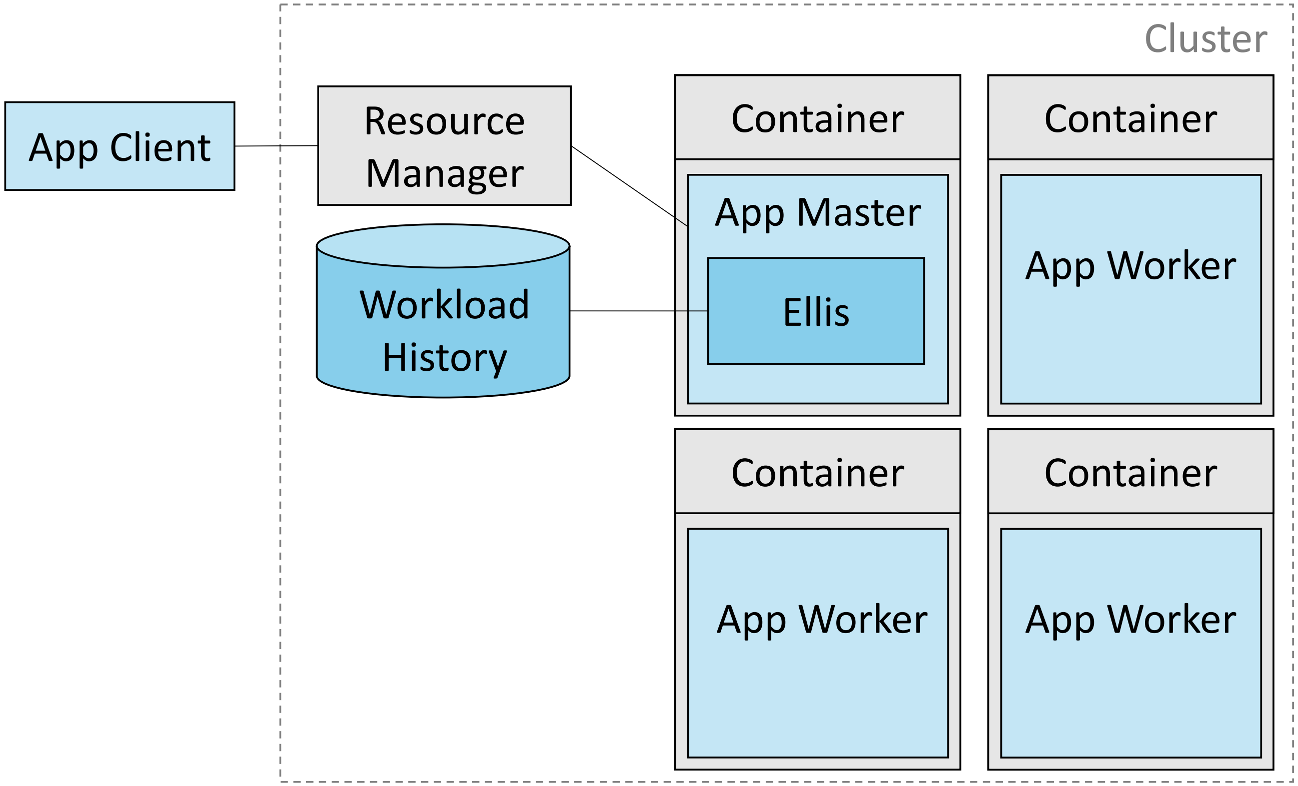


Stage-wise Dynamic Allocations



Prototype System: *Ellis*

- Integrated and usable on a per-job basis with YARN



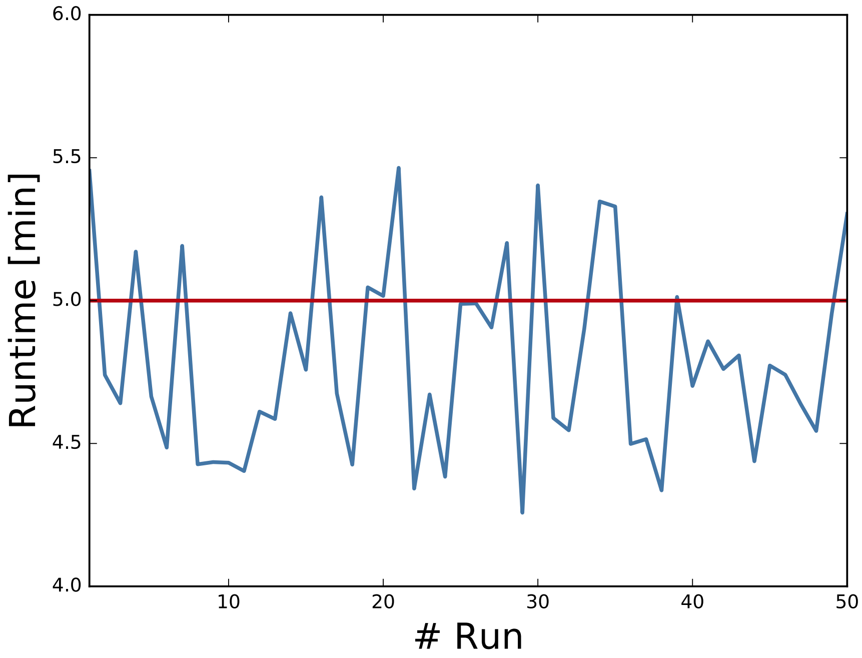
Evaluation Setup

- Using up to 40 commodity nodes (8 cores, 16 GB RAM) and exemplary iterative Spark jobs (MLlib)

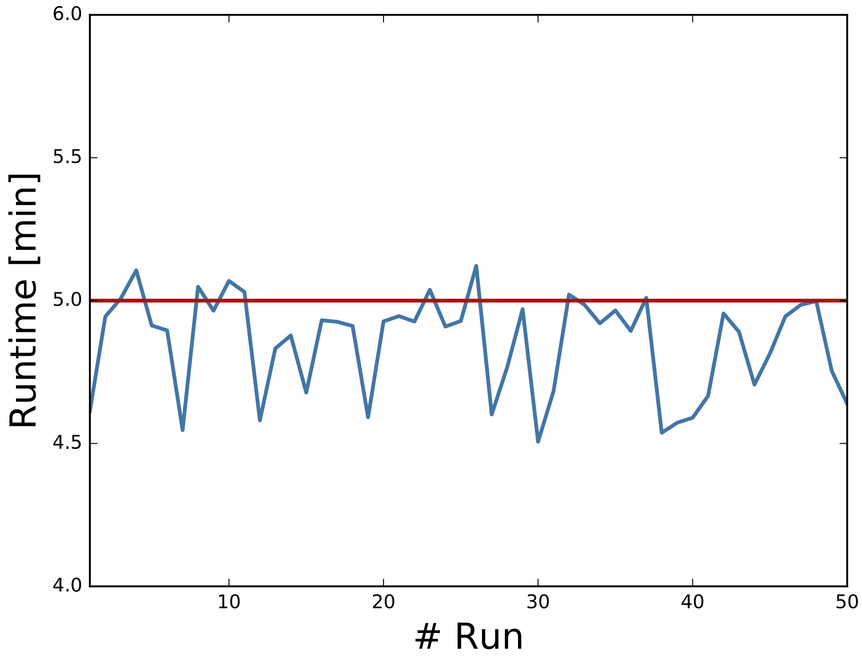
Job	**Dataset**	**Input Size**
Multi-Layer Perceptron (MLP)	Multiclass	29 GB
Gradient Boosted Trees (GBT)	Vandermonde	111 GB
Stochastic Gradient Descent (SGD)	Tables	37 GB
K-Means	Points	50 GB

Results for Gradient Boosted Trees

Ellis static (only initially):



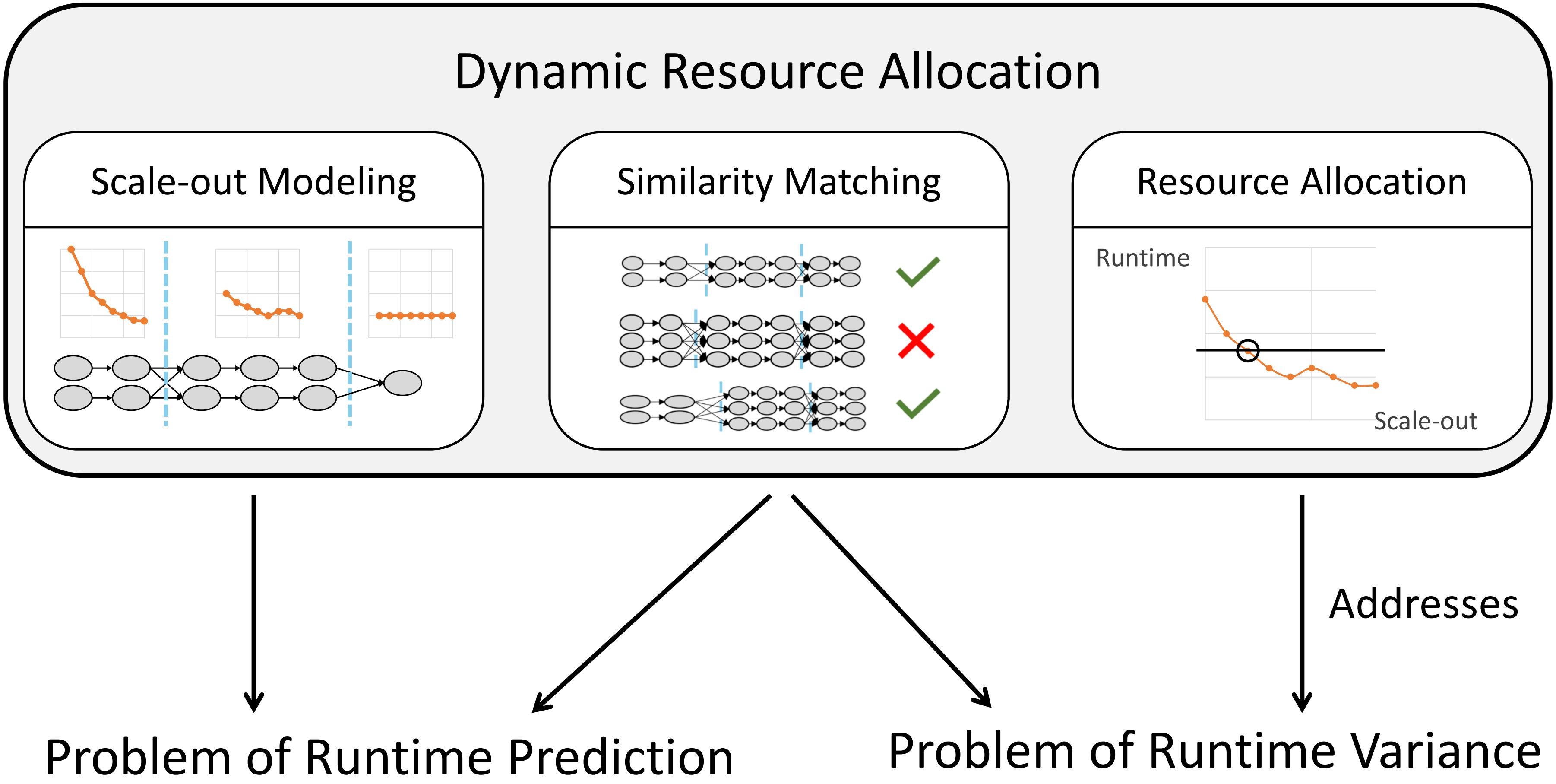
Ellis dynamic (per stage):



Results for All Benchmark Jobs

- Comparison of $\frac{Ellis\ dynamic}{Ellis\ static}$ in three metrics:

Job	Violation Count	Violation Sum	Resources
SGD	25%	7%	94%
K-Means	35%	5%	80%
GBT	47%	29%	100%
MLP	69%	13%	127%


Dynamic Resource Allocation
Scale-out Modeling
Similarity Matching
Resource Allocation
Runtime
Scale-out
Addresses
Problem of Runtime Prediction
Problem of Runtime Variance

Summary

- New methods for dynamic resource allocation that are applicable to distributed dataflow frameworks
- Prototype integrated and usable with YARN
- Promising evaluation results

References

- [GoogleMR] J. Dean and S. Ghemawat. "**MapReduce: Simplified Data Processing on Large Clusters**". *Commun. ACM* 51.1 (2008). ACM. 2008.
- [Twitter] G. Mishne, J. Dalton, Z. Li, A. Sharma, and J. Lin. "**Fast Data in the Era of Big Data: Twitter's Real-time Related Query Suggestion Architecture**". *2013 ACM SIGMOD International Conference on Management of Data* (**SIGMOD '13**). ACM. 2013.
- [Google Trace] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "**Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis**", *Third ACM Symposium on Cloud Computing* (**SoCC '12**). ACM. 2012.
- [Quasar] C. Delimitrou and C. Kozyrakis, "**Quasar: Resource-efficient and QoS- aware Cluster Management**", *19th International Conference on Architectural Support for Programming Languages and Operating Systems* (**ASPLOS '14**). ACM. 2014.
- [Morpheus] S. A. Jyothi, C. Curino, I. Menache, S. M. Narayanamurthy, A. Tumanov, J. Yaniv, R. Mavlyutov, I. n. Goiri, S. Krishnan, J. Kulkarni, and S. Rao, "**Morpheus: Towards Automated SLOs for Enterprise Clusters**", *12th USENIX Conference on Operating Systems Design and Implementation* (**OSDI'16**). USENIX. 2016.