Morpheus: Towards Automated SLOs for Enterprise Clusters

Sangeetha Abdu Jyothi* Carlo Curino† Ishai Menache† Shravan Matthur Narayanamurthy† Alexey Tumanov^
Jonathan Yaniv** Ruslan Mavlyutov^^ Íñigo Goiri† Subru Krishnan† Janardhan Kulkarni† Sriram Rao†

* University of Illinois, Urbana-Champaign  † University of California, Berkeley
** Technion - Israel Institute of Technology  ‡ University of Fribourg
Operator/User tensions

- Run as many jobs as possible
- Fair-sharing

- Our focus is on batch jobs in big data enterprise clusters
- Periodic jobs should run predictably – output available by deadline
Roadblock: Unpredictability

 Sharing-induced
resource-sharing, queueing etc

Inherent
stragglers, failures, skew, hardware changes

25% of user tickets due to unpredictability
Current “solution”: Over-provisioning

50-k node COSMOS cluster

Users over-provision > 75% jobs
Utilization vs. Predictability
Towards automated SLOs

System focuses on periodic jobs

Empirically >60% are periodic

Our results:

5-13x reduction in deadline SLO violations

Reduce cluster size by 14-28%
Morpheus Overview

- Quantify user requirements
- Pack jobs efficiently
- Respond to unpredictabilities

Components:
- Automatic Inference Module
- Reservation Mechanism
- Dynamic Reprovisioning

Processes:
- Logs
- User sign-off
- SLO resource estimate
- Monitoring
Automatic Inference Module

Quantify user requirements

Derive deadline SLOs

Estimate job resource demands
Deadline SLOs

![Diagram showing processes X to Z with nodes A and B and a histogram showing job completion time of A and output consumption time of B. The deadline is indicated by a red dashed line at 14:00.]
Deadline SLO validation

\[ P(B_{\text{fail}} \mid A_{\text{missSLO}}) > 4 \times P(B_{\text{fail}} \mid A_{\text{meetSLO}}) \]

Valid estimate

~70% of jobs have high scheduling flexibility
Job Resource Demand

- Usage patterns (container skylines) of multiple instances of the same job
- Generate the best fitting model using Linear Program
- Fitting controlled by a parameter, \( \alpha \) (higher \( \alpha \) \( \rightarrow \) less resources)
- Other alternatives – Jockey [Eurosys‘12], PerfOrator [SoCC’16]
Reservation Mechanism

Automatic Inference Module → LCM → Dynamic Reprovisioning

User sign-off

SLO resource estimate

Pack jobs efficiently

Compact storage of jobs based on Least Common Multiple (LCM) of periods

LowCost Packing Algorithm
LCM Representation

Smallest repeating unit stored – Least Common Multiple (LCM) of periods

Efficient storage

Predictable allocation for users
Other key techniques (in the paper)

LowCost Packing Algorithm
Heuristic for achieving a balanced allocation
Load-aware online packing

Dynamic reprovisioning
Continuous monitoring of jobs
Allocate more resources when “progress” is slow
Experiments

Implementation:
Recurrent reservation mechanism, packing algorithm, and dynamic reprovisioning in Apache Hadoop/YARN
Stand-alone inference subsystem

Workload:

**Enterprise-trace**: Three-month trace from 50k-node COSMOS cluster

**Hadoop-trace**: One-month trace from 4k-node Hadoop cluster

**TPC-H**: Standard TPC-H benchmark
Evaluation – Scalability test

Morpheus can handle load in production clusters

2700-node cluster with 92 TB memory
Evaluation – Resource estimation

Morpheus provides more accurate resource estimates

Level of fitting controllable in the inference module of Morpheus

Higher $\alpha \rightarrow$ Tighter fitting $\rightarrow$ Less over-provisioning
Evaluation

\[ \alpha = 1\%, \text{ dynamic} \]

\[ \alpha = 1\%, \text{ static} \]

\[ \alpha = 5\%, \text{ static} \]

\[ \alpha = 5\%, \text{ dynamic} \]

\[ \alpha = 5\% \]

Normalized Cluster Resources

Normalized SLO Violations

Baseline (user provisioned)
Conclusion

• Predictable performance with lesser resources and higher utilization

• Three main ideas
  • Automatic inference
  • Recurrent reservations
  • Dynamic reprovisioning

• 5-13x reduction in SLO violations

• 14-28% reduction in cluster size
THANK YOU!