Usage-based scheduling in Kubernetes

What

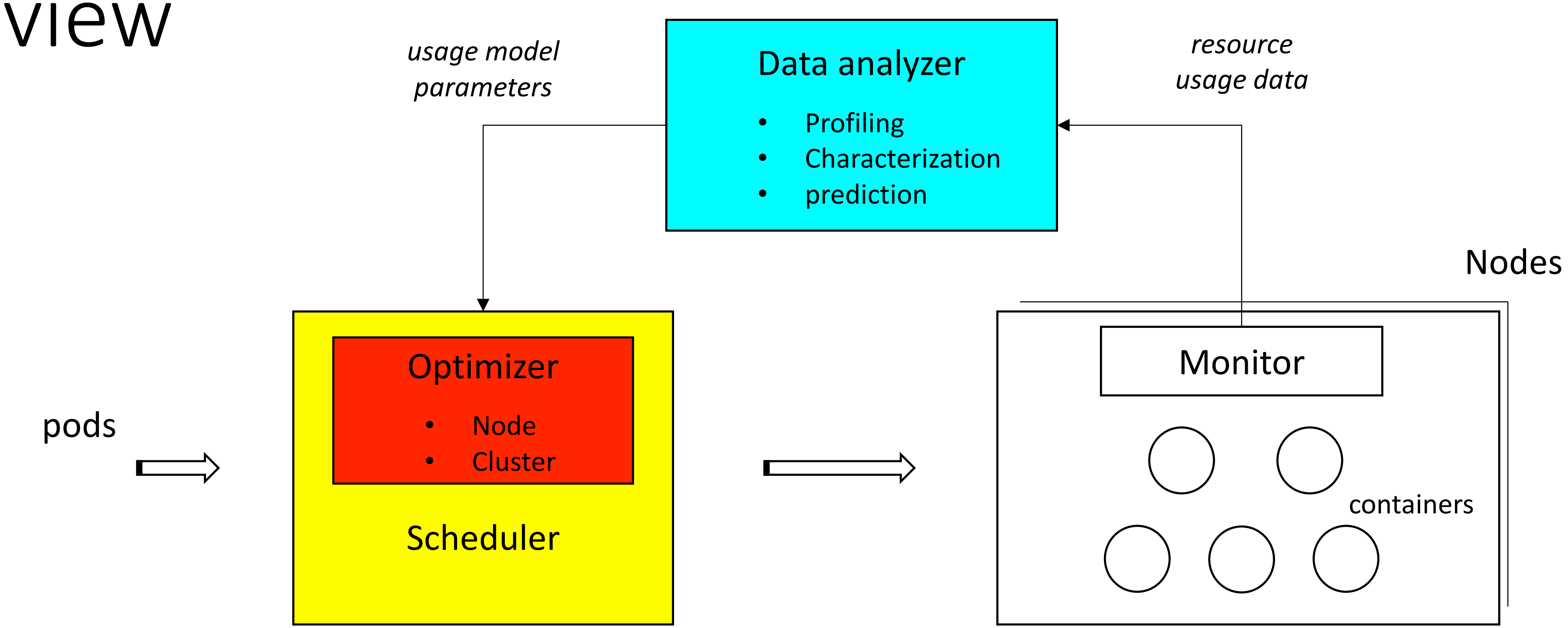
- Usage-Data Aware Scheduling

  Schedule pods based on actual usage data, leading to higher acceptance rates and better resource utilization, while limiting the risk of overloading due to usage variability

Why

- Resource request and limit specifications may not be accurate, leading to excessive over-allocation of resources
- Observing and analyzing actual resource usage data would help in determining
  - a good mix of different types of pods on a node
  - an optimal scheme for loading nodes in a cluster

Overview



usage model parameters

Data analyzer

- Profiling
- Characterization
- prediction

resource usage data

pods

Optimizer

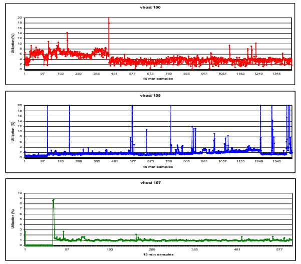
- Node
- Cluster

Scheduler

Nodes

Monitor

containers



Container data

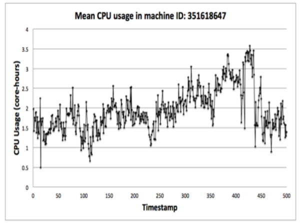
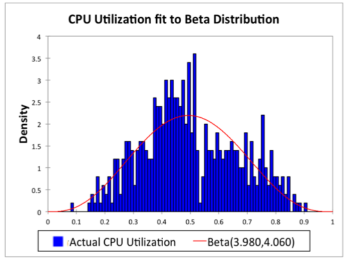


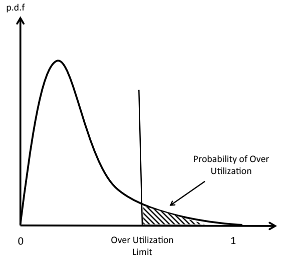
Fig. 13. CPU usage for average of 18 tasks, mean = 1.95, variance= 0.3

Node aggregated data

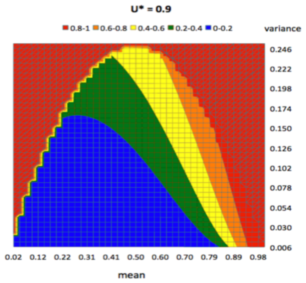


(c) CPU Utilization fit to Beta Distribution for average of 49 tasks

Data modeling



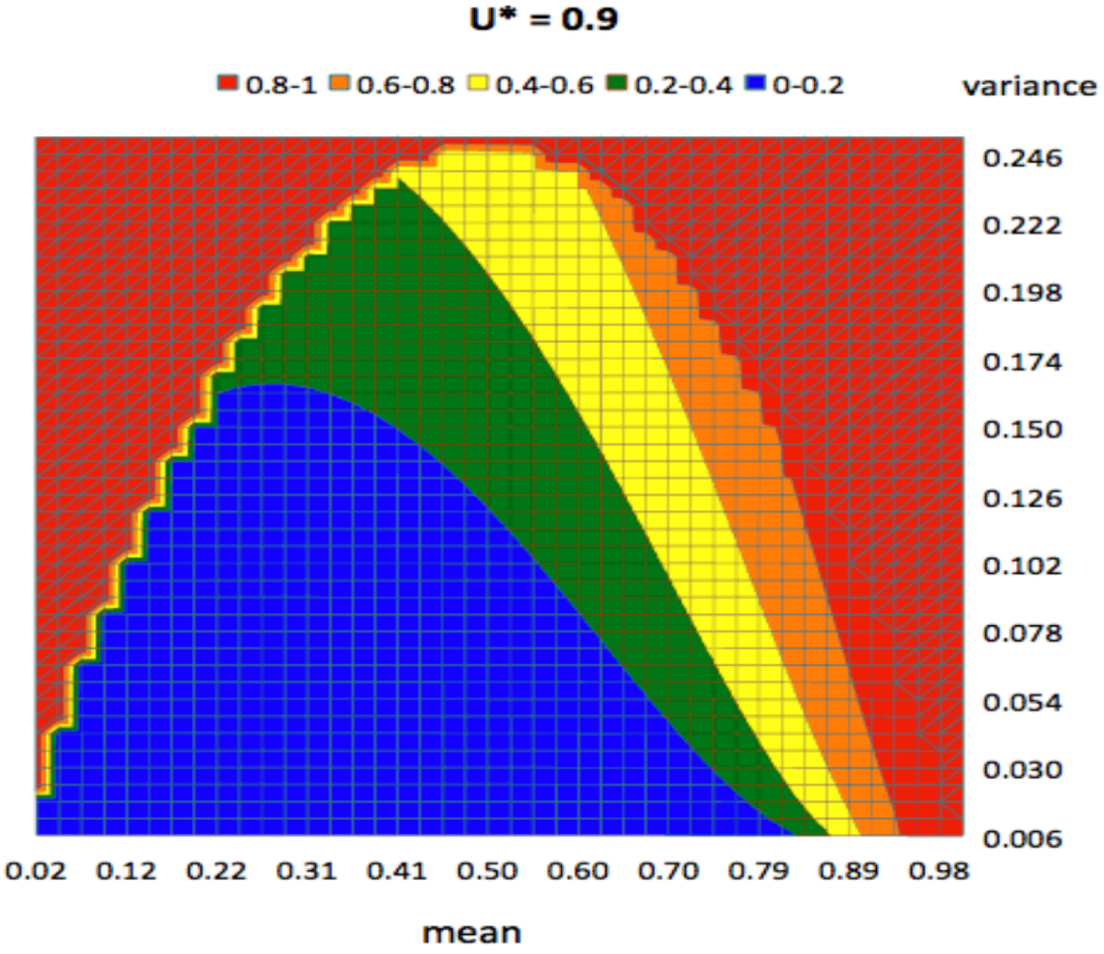
Effective availability



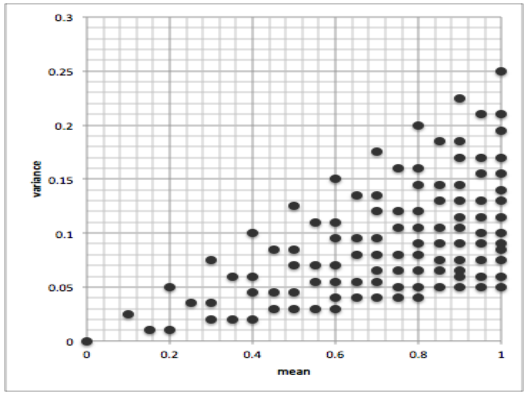
Optimization

Over-subscription policy

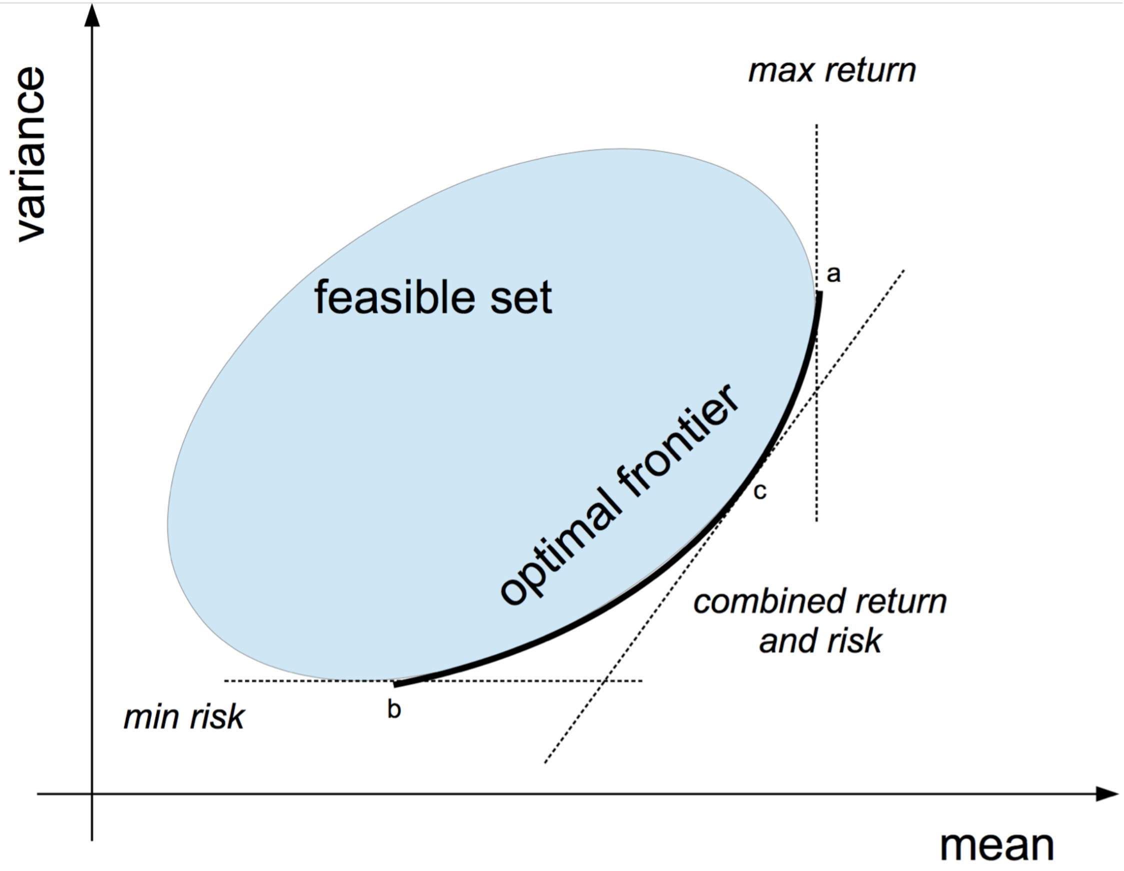
- Set parameters
    - Load threshold (U*)
    - Overload probability (epsilon)
- Explore Mean-Variance (MV) space
- Select operating point
    - Low load: afford variability
    - Medium load: calculated variability
    - High load: limited variability
- Results in an effective resource availability metric to be used in a scheduling predicate



Optimal container mix on a node



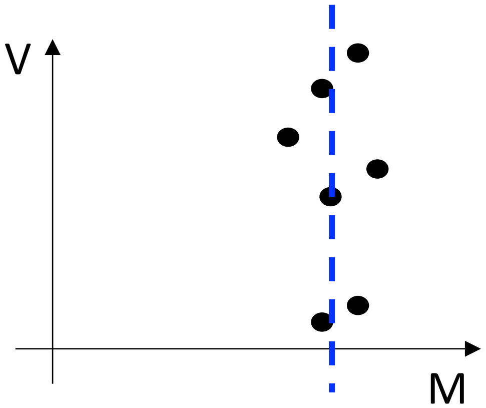
- Consider multiple container types, characterized by
  - Average resource usage
  - Variance in resource usage
- A given mix of container types leads to a point in the MV space
- Combinations of mix choices lead to a feasible set
- Optimal points result from a given compromise of loading and risk
- Related to Markowitz portfolio theory



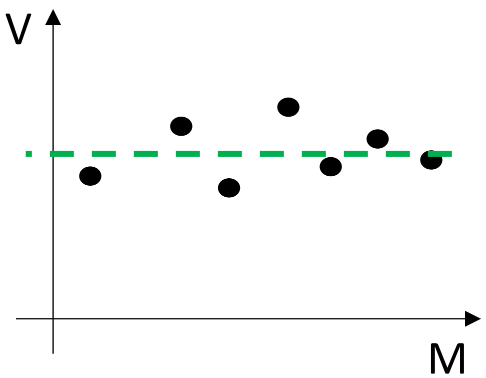
Balancing the cluster

- A point represents the loading of a node
- Collection of points represents the cluster

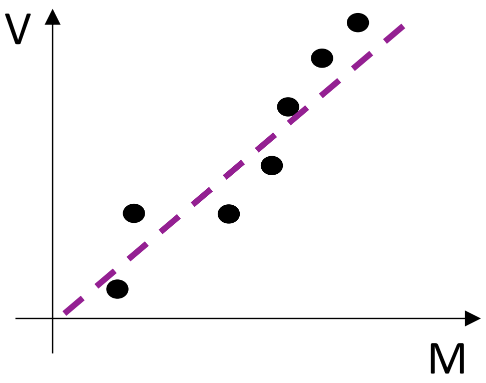
Balance load



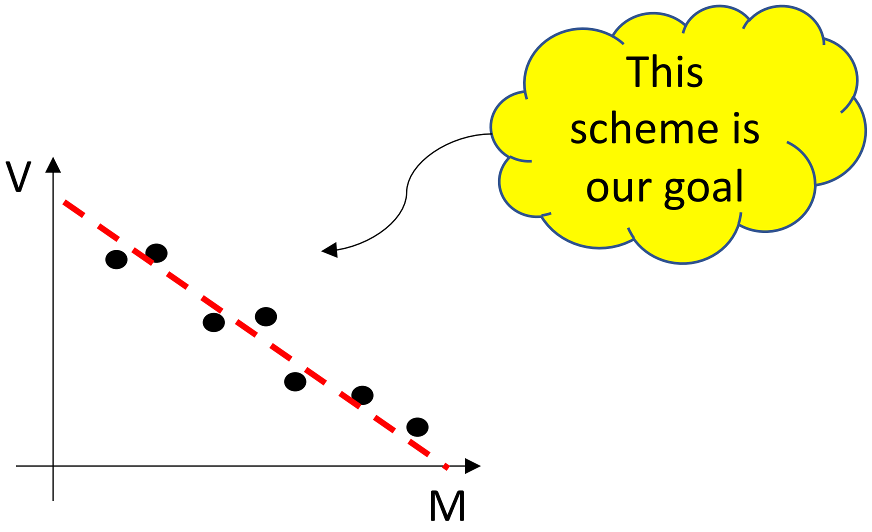
Balance variability



Balance relative variability



Balance risk



This scheme is our goal

Potentially enabling …

- Scheduling of a pod on a node using the effective availability metric
- Distribute load according to a cluster balancing policy
- When adding/deleting a pod to a replication set, schedule pod optimally
- When migrating a pod (e.g. during staging), select a node optimally
- Classify containers through learning their type, based on resource usage amount and variability