# Tetris: Predictive Pod Placement Strategy for Kubernetes

CSC 724 Spring 2019

Project Proposal

Vasudev Bongale

Kyle Martin

Sohail Shaikh

21-Feb-2019

# Agenda

- Problem Statement
- Idea
- Kubernetes Overview
- Related Work
- Prediction Strategies
- Experiment and Evaluations
- Project Roadmap

# Problem Statement

Kubernetes does not consider I/O utilization when provisioning pods. This could lead to performance degradation on an I/O bottleneck K8s Node. We propose to introduce a predictive pod placement strategy to avoid resource bottlenecks.

# Idea

- Manually categorizing pods by workload type.
  - CPU intensive
  - Memory intensive
  - IO intensive
- Policy based predictive modeling.
  - ARIMA, Gaussian - SVM
- Intelligent placement.
  - Choose a node for a pod

# Kubernetes (K8s)

- Automates container orchestration tasks
- Master-Slave Architecture
- Master
  - Scheduler
  - API Server
  - etcd
- Node
  - cAdvisor
  - Kubelet
  - Pods

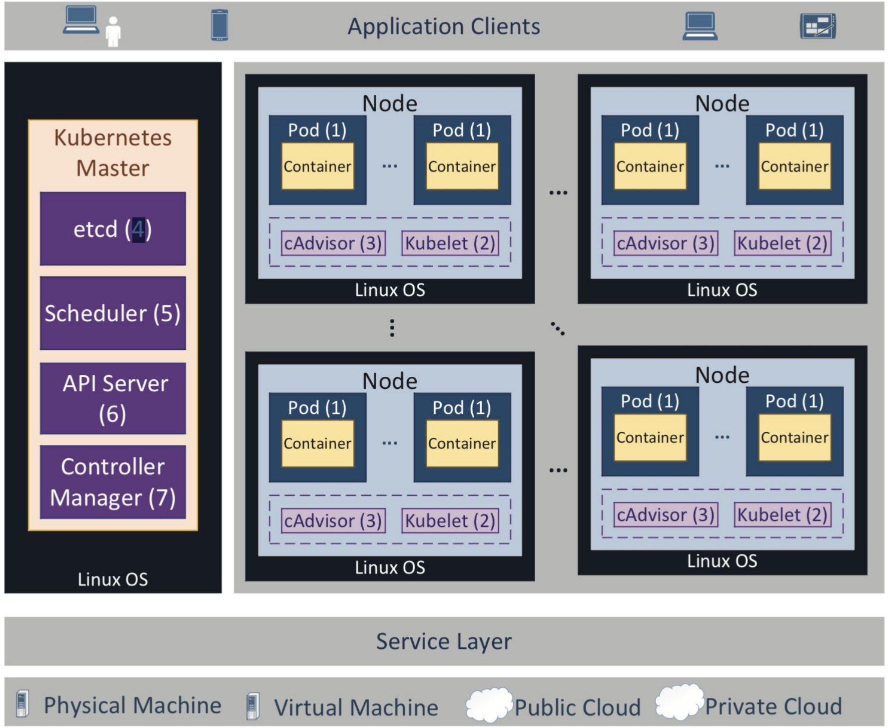



Figure from “A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning” [3]

# Monitoring Frameworks

- Monitoring is hard - dynamic cluster
- cAdvisor: built-in monitoring solution
- K8s is now designed to behave as a platform
- Core-metrics-pipeline
  - Metrics-server: does not store historical metrics
- Monitoring-pipeline
  - custom-metrics-API
  - Full monitoring solution: Ex: Prometheus, Datadog

# IO-RE

- Load balance decisions
  1. $Load_{IO}$ at $node_i$ updated by $Task_j$
  2. $Threshold_{IO}$ calculated for $node_i$
  3. *If* $Load_{IO}$ > $Threshold_{IO}$ *Then* move $task_j$
  4. *If* Load I/O < $Threshold_{IO}$ $Task_j$ *Then* execute at $node_i$

$$load_{IO}(i) = \sum_{j \in N_i} page(i,j) + \sum_{j \in N_i} IO(j)$$

Equation 1

$$threshold_{IO}(i) = \frac{D_i}{\sum_{j=1}^{n} D_j} \times \sum_{j=1}^{n} load_{IO}(j)$$

Equation 2

Equations from “Dynamic load balancing for I/O-intensive tasks on heterogeneous clusters” [1]

# IOCM-RE

Algorithm: IO-CPU-Memory based load balancing (IOCM-RE):
/* Assume that a task j newly arrives at node i */
if $IO(j) + \sum_{j \in N_i} IO(k) > 0$ then
  The IO-RE policy is used to balance the system node; /* see Section 3.2 */
else  if $page(i,j) + \sum_{j \in N_i} page(i,k) > 0$ then /* see Section 3.1(2) */
    The memory-based policy is utilized for load balancing;
  else   /* see Section 3.1(1) */
    The CPU-based policy makes the load balancing decision;

**Fig. 1.** Pseudocode of the IO-CPU-Memory based load balancing

Pseudocode  from "Dynamic load balancing for I/O-intensive tasks on heterogeneous clusters" [1]

# PAC

- PAC: Pattern driven Application Consolidation [2]
- Extract resource-usage patterns from VMs using FFT - *signatures*
- DTW (Dynamic Time warping) for signature matching.

- VM placement to provide global application consolidation and load-balancing

- Predicts future resource demands with 50-90% less error

# Why is workload prediction necessary?

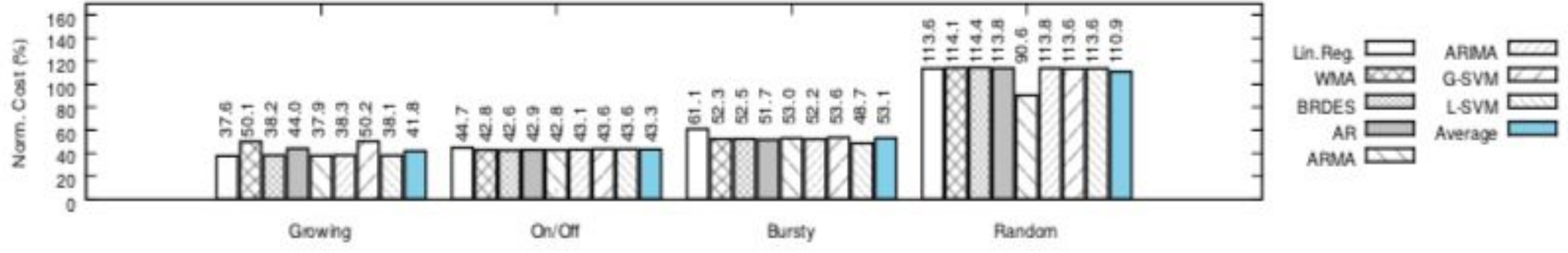
- Traditionally threshold based policies have been used.
- It's impossible to know the threshold in advance because workload patterns change over time.
- The threshold values are largely dependent on stability of a system.

# Predictive Approaches

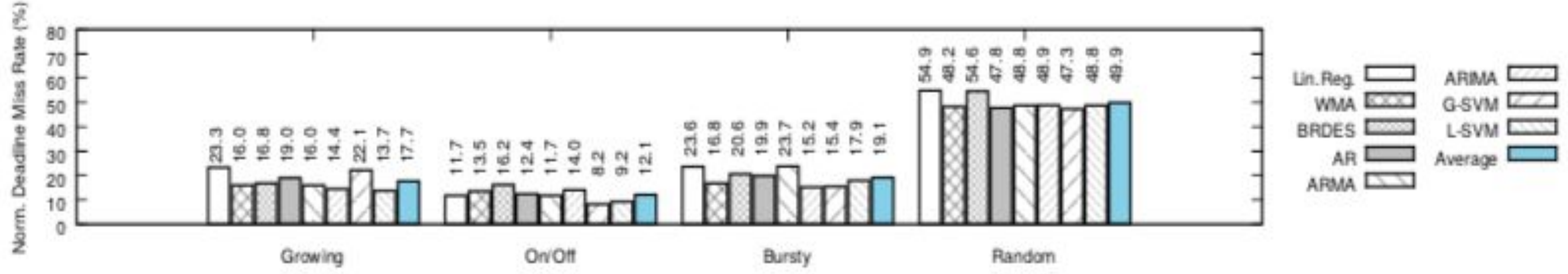
- Naive
  - Mean, KNN
- Regression
- Temporal
  - ARMA, ARIMA
- Non-temporal
  - SVM, decision trees
  - Least chosen for scaling

NO one predictive algorithm to rule them all!
(Image source: Wikipedia)

# Comparing Predictive Approaches



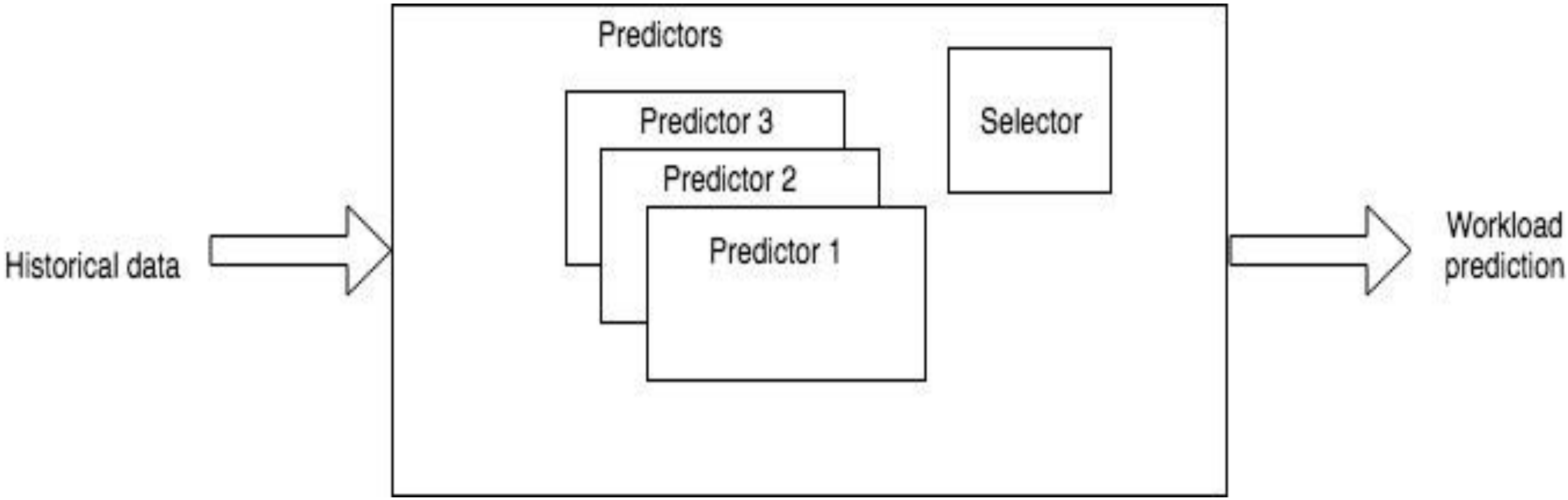

(a) Normalized Cost.




(b) Normalized Job Deadline Miss Rate.

Figure from “Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling [4]

# Our Predictive Approach



# Experiment

- Provision Resources:
    - AWS EC2 (2 A1.xlarge)
    - VCL
- Deploy workload generator applications.
    - IOmeter
    - stress-ng
- Collect core system metrics to generate historic data.
- Train predictive algorithms to predict node workload.
- Analyze pod placement decisions.

# Evaluations

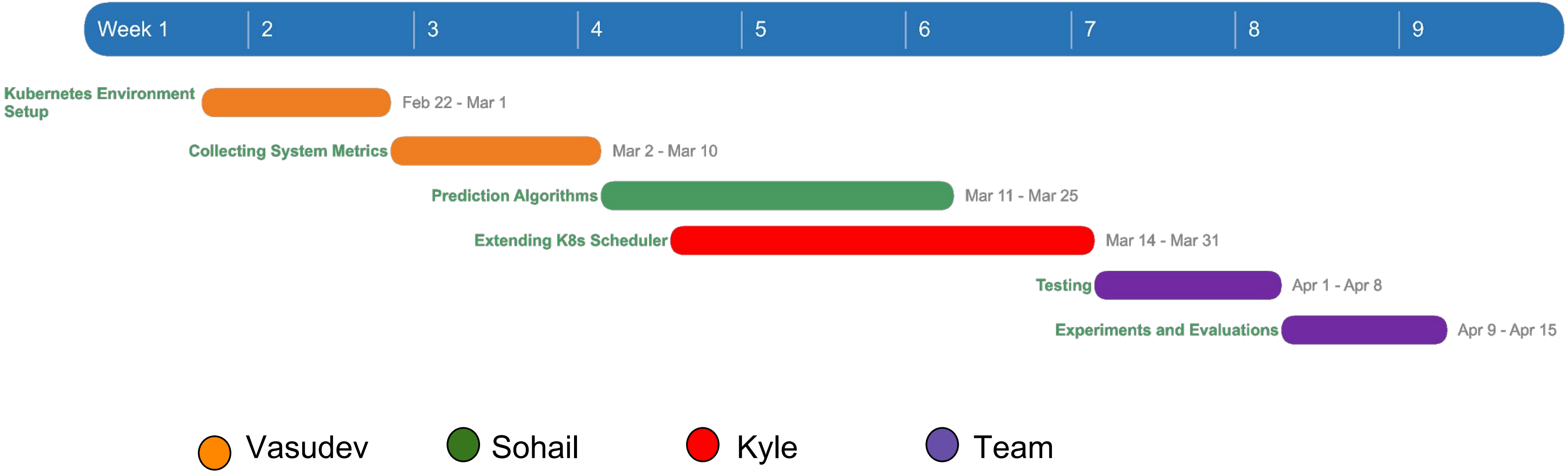
Native implementation

- No historical data involved for placement decisions
- May result in bad placement choices
- Not workload aware

Tetris

- Show improved resource utilization of entire cluster
- Placement decisions based on its workload

# Project Roadmap



# References

1. Xiao Qin, Hong Jiang, Yifeng Zhu, and David R Swanson. 2003. Dynamic load balancing for I/O-intensive tasks on heterogeneous clusters. In International Conference on High-Performance Computing Springer, 300–309.

2. Zhenhuan Gong and Xiaohui Gu. 2010. Pac: Pattern-driven application consolidation for efficient cloud computing. In18th IEEE/ACM International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems(MASCOTS 2010). IEEE, 24–33

3. Chia-Chen Chang, Shun-Ren Yang, En-Hau Yeh, Phone Lin, and Jeu-Yih Jeng.2017. A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning. In GLOBECOM 2017 - 2017 IEEE Global Communications Conference.IEEE, Singapore, 1–6

4. In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. 2016. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In2016 IEEE 9th International Conference on Cloud Computing (CLOUD).IEEE, San Francisco, CA, USA, 1–10.

# Questions?