cesnet
metacentrum

KUBERNETES CONTAINER ORCHESTRATOR
SCHEDULING PROBLEMS AND CHALLENGES

Dalibor Klusáček
CESNET

April 2022
Brno

Past and Future Kubernetes Tutorials

RNDr. Lukáš Hejtmánek, Ph.D.

- 4. 5. + 11. 5. 2022 - Kubernetes Tutorial (1&2) here at Sitola
- Online Webinar (past)
  - https://metavo.metacentrum.cz/cs/seminars/Webinar_2022/kubernetes2022.html

Platforma Kubernetes — Úvodní seznámení

Lukáš Hejtmánek, CERIT-SC

What is the Big Deal with K8s?

- **Containerized applications are popular**
  - Containers hide the complexities of modern SW
- **K8s is “container orchestrator”**
  - Deploys containers (in so called “Pods”)
  - Handles network, storage access
  - Checks their status (availability and scalability)
  - Organizes them wrt. given rules (Pod-to-node mapping)
  - Kills/restarts Pods when needed
- **CERIT-SC K8s installation**
  - 2,560 CPUs in 20 nodes (128 cores, 512 GB RAM, 1 GPU, 7TB local SSD)
  - Web and interactive applications
    - Jupiter Hub, Binder Hub, Ansys, Matlab, RStudio, Wordpress…

Scheduling Challenges in K8s

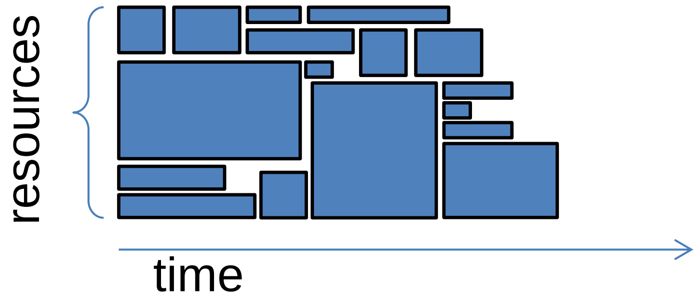
- We know standard batch-oriented HPC scheduling
- We cannot reuse same techniques in K8s easily
- Examples, Comparisons & Discussion

HPC vs. K8s SCHEDULING

Batch vs K8s Workloads

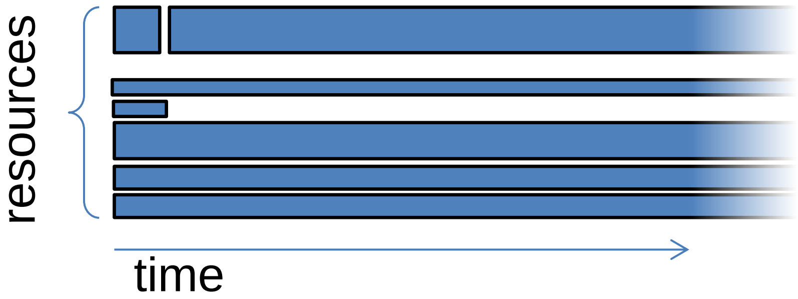
Batch workloads

- Scripted executables
- Non-interactive (mostly)
- **Waiting in queue is OK**
- Resource intensive
- Rather accurate resource requirements
- Strict **maximum runtime limit**



K8s workloads

- Interactive usage is common
- GUI-based work
- Long running services
- **Waiting is not OK**
- Overestimated resource requirements
- Usually **not limited runtime**



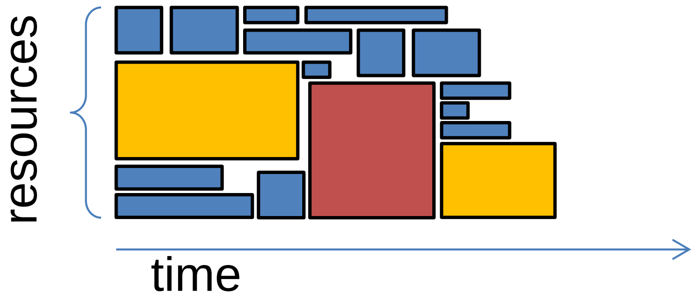
Batch vs K8s Scheduling Concepts

Batch scheduling basics

- **The organization owns the resources**
- Resources are **provided for free**
- So fairness is important

How does the scheduler work?

- Jobs in queue(s)
- Queue is ordered by priority
- User-priority is dynamic (fairness)
  - User waiting = priority ↑
  - User computing = priority ↓

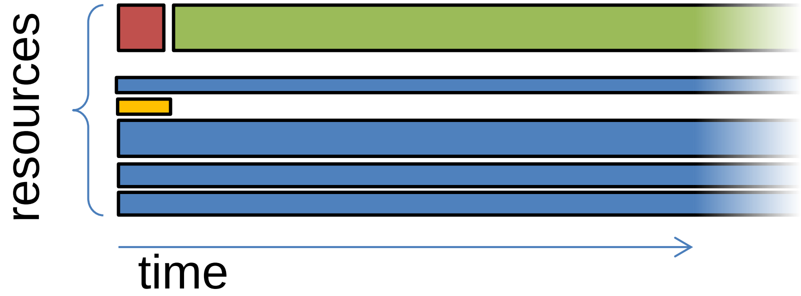


- Over long time period user's "share" is balanced with other active users

Scheduler decides who gets the resources and when

Batch vs K8s Scheduling Concepts

- **K8s workloads**
  - Interactive, no waiting, no maximum runtime…
- **Scheduling basics (cloud, K8s) in commercial world**
  - **Users "own" the infrastructure**
  - Pay-per-use model
  - **Perfect motivation to release resources**
  - Unused allocations? Overcommitted
    - Used by low QoS workloads
    - Can be terminated, if needed



- **There is no "scheduling" needed… You are the "scheduler"**
  - Instead, **"capacity planning"** is crucial

Batch vs K8s Scheduling Concepts

- **Scheduling = capacity planning**
  - Load prediction (Black Friday, Christmas, Superbowl, new season of Mandalorian…)
  - Clever aggregation of different workloads
  - Resource pool can be increased (thanks to the revenue)
- **Good scheduler/capacity planner = money**
  - You aggregate better
  - You sell more with less resources needed
- **The main difference between batch and K8s scheduling**
  - **The user who gives you the money is the "scheduler"**
  - **So there are no sophisticated schedulers available**

Scientific (non-commercial) Use of Kubernetes

- **We are not commercial providers**
  - **We have strictly limited resources**
  - Yet **our users expect similar experience** as in the commercial world
    - Partly because we advertise our installation in such way
- **K8s offers basic mechanisms for scheduling**
  - Resource quotas
    - Constraints that limit aggregate resource consumption per namespace
  - Pod resource requests and limits
    - Guaranteed *requests* + best effort upper bound *limits*
  - Static Priority Classes
    - Higher priority Pod evicts lower priority Pod if needed
  - Pods with limited runtime (called Jobs)

cesnet metacentrum

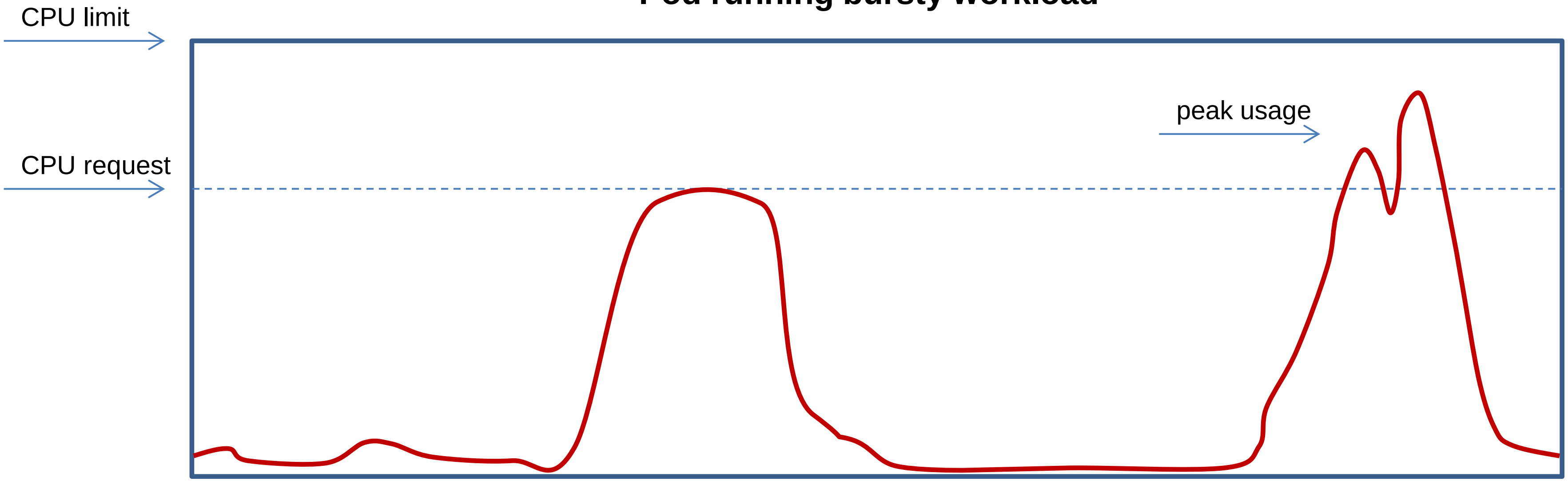
PROBLEMS?

Common Problems - Bursty Workloads

Bursty workloads

- E.g., long running services that scientists use “three times a week for 2 hours”
- Such services are mostly idle, but will have peaks
- **Overestimated requests**

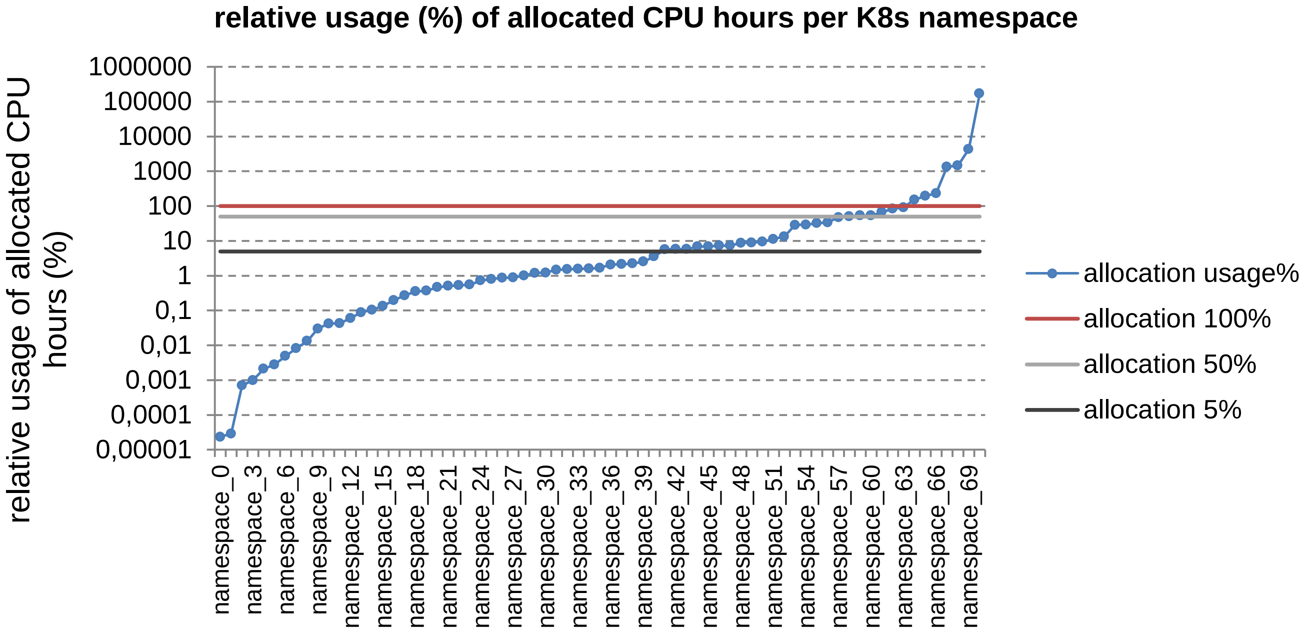
Pod running bursty workload



Common Problems – Resource Wasting

What is the problem?

- In general, overestimated requests (and zombies)
- Requests are guaranteed, thus overestimation means resource wasting

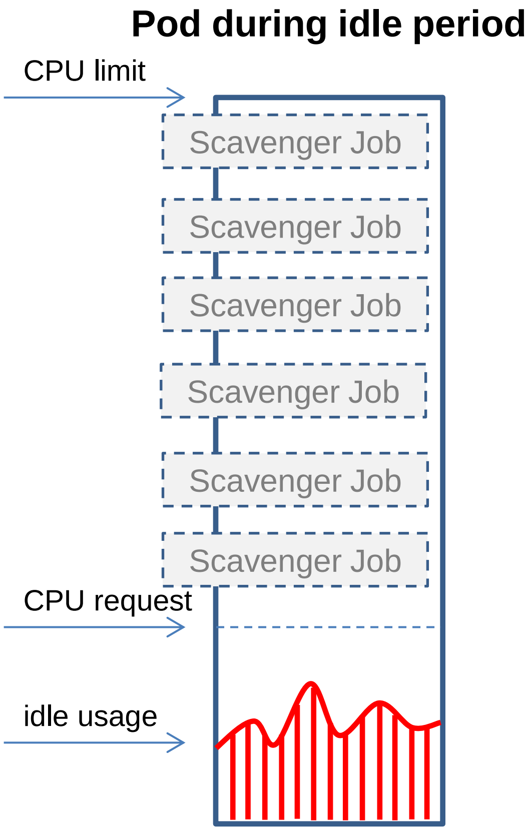


Some Good News

- Some problems can be addressed quite efficiently

- Free resources can be used by “scavenger” jobs
  - Jobs that are small and can be evicted/restarted easily
  - Help to utilize free resources

- Pod requests must be “low”
  - And we must **allow the affected Pod to “scale up”**



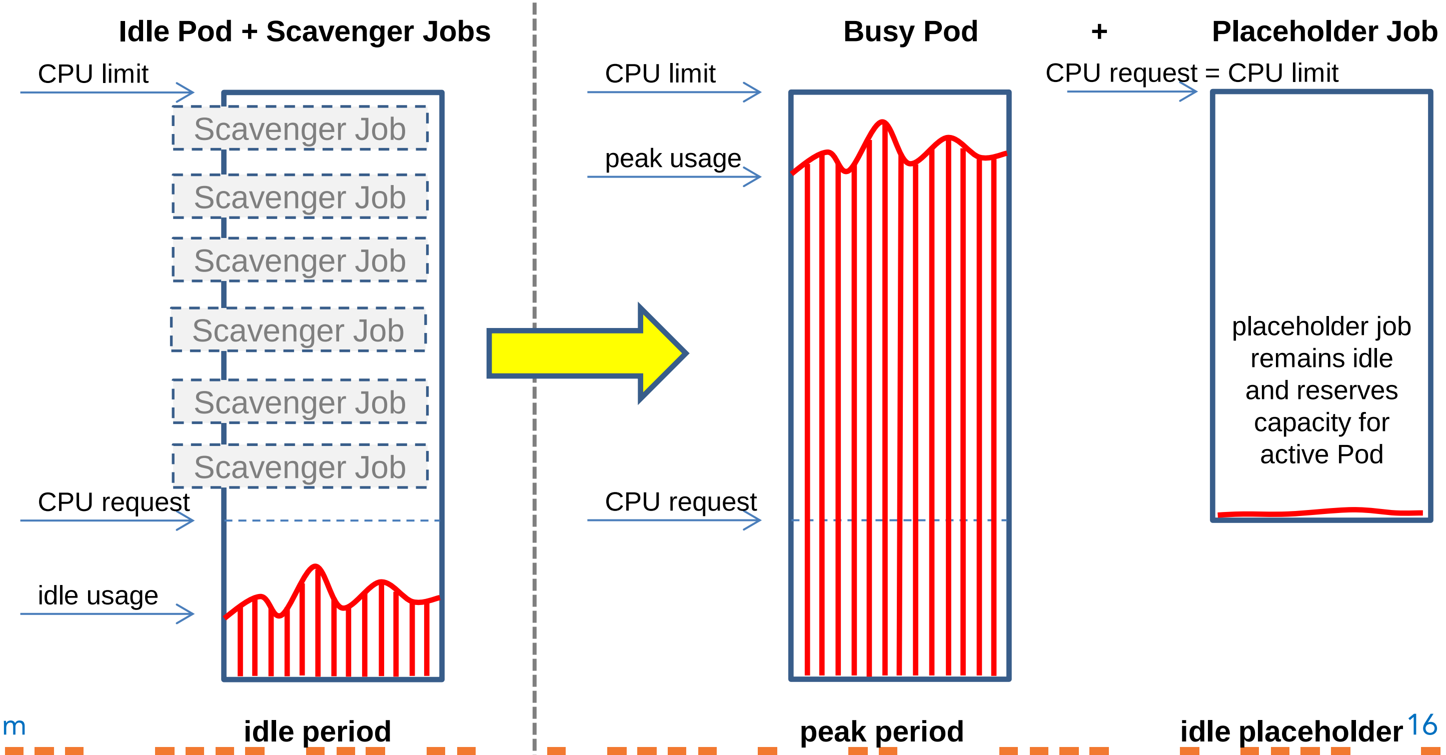
K8s Limitations

- **It is impossible to modify Pod priority dynamically**
- **Or adjust too generous/tight Pod allocations**
  - Pod restart is requested
  - No problem for “stateless” microservices
  - Usually bigger deal for “scientific computations”
- **There is a “workaround”**
  - Enables the Pod to use more/less resources

Workaround: Placeholder Jobs

Pod-scaling can be achieved by running "placeholder" job

- Placeholder evicts scavangers
- Best effort
- Manual process



OPEN PROBLEMS

Open Problems – HPC vs. K8s Comparison

Common HPC batch scheduler

- When the **system is full and new user arrives** you can always:
- Tell the user what is his/her priority
- And estimate (roughly) when the running jobs of other users will terminate
- Or even **provide him/her a non-destructive reservation**
- **This is all automatic**

In K8s…

- **Impossible to estimate Pod wait time** (when we are out of resources)
  - No guarantees – the Pod either starts immediately or… never?
  - Unless we “manually” adjust the priority of the new Pod to evict some running Pod(s)
- **Resource reclaiming is not solved** => no Pod life-cycle management
- There is no such thing as “fairshare” in K8s
- No automation

Alternate Solutions & Future Work

- Partition the infrastructure into clusters with different "rules"
  - E.g., a cluster with time-limited access only
  - Dedicated schedulers for each such cluster (either our own or third party)
- Still, infrastructure will suffer from fragmentation
- The need for long-term solution remains
  - How to offer the service?
  - What "QoS" we want to guarantee
- Definition of overall usage policy
  - **Define mechanisms to implement this policy**
  - Either "by hand" or through some automated scheduling policy

QUESTIONS?
SUGGESTIONS?

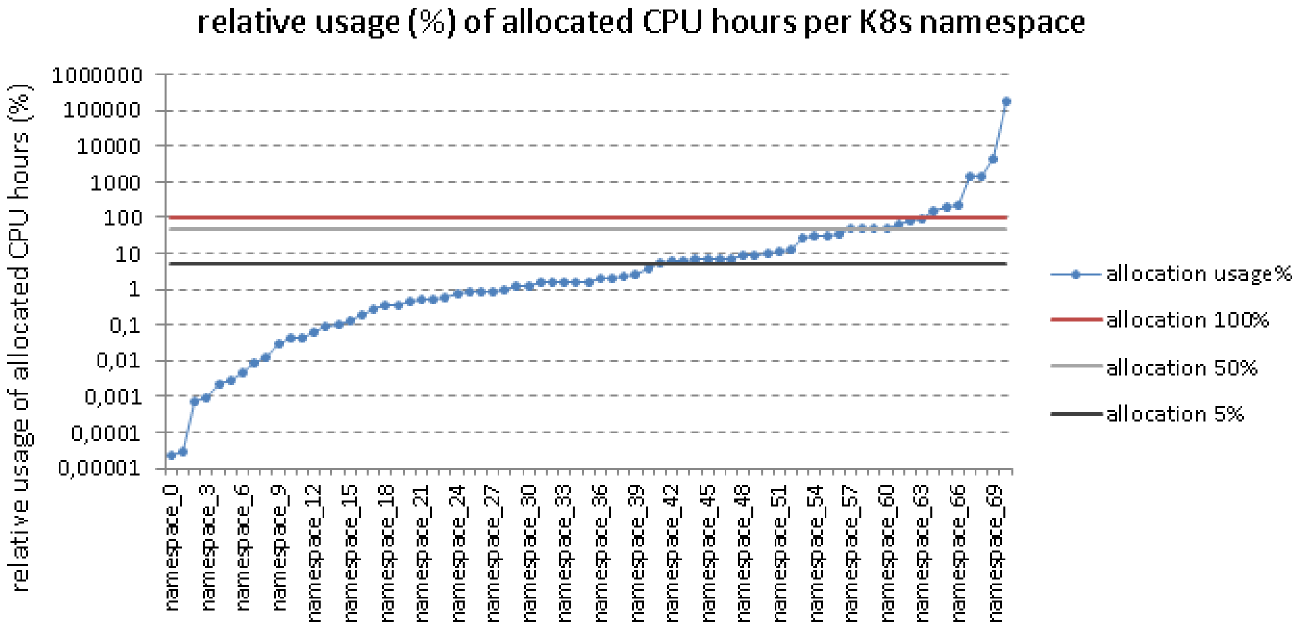
more info at:

- Sitola seminars in May (**4.+11. 5. 2022**)
- JSSPP 2022 paper ***"Using Kubernetes in Academic Environment: Problems and Approaches"***
- Future talk at ***Kubernetes batch + HPC day EU***

Specific Problems – Resource Wasting

What is the problem?

- In general, overestimated requests (and zombies)
- Requests are guaranteed, thus overestimation means resource wasting


relative usage (%) of allocated CPU hours per K8s namespace
relative usage of allocated CPU hours (%)
namespace_0
namespace_3
namespace_6
namespace_9
namespace_12
namespace_15
namespace_18
namespace_21
namespace_24
namespace_27
namespace_30
namespace_33
namespace_36
namespace_39
namespace_42
namespace_45
namespace_48
namespace_51
namespace_54
namespace_57
namespace_60
namespace_63
namespace_66
namespace_69
allocation usage%
allocation 100%
allocation 50%
allocation 5%