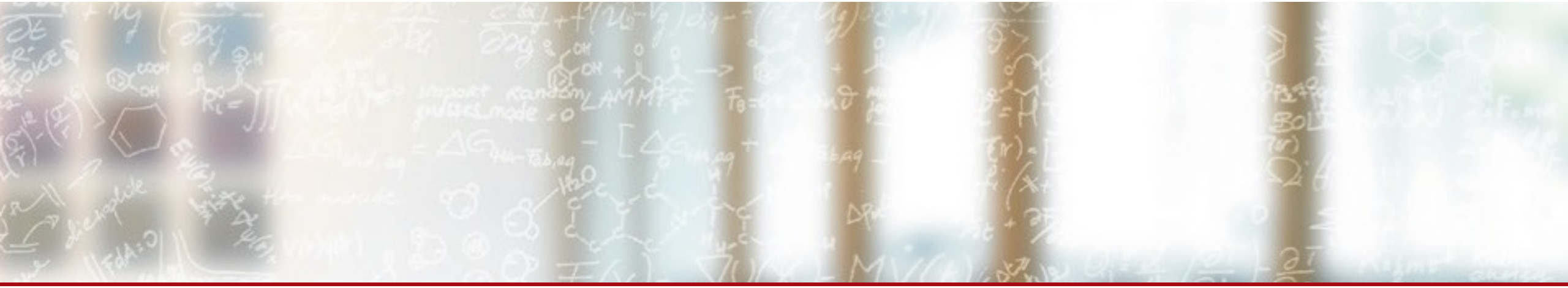
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

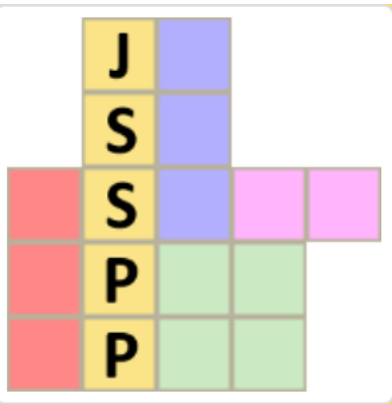


Resource elasticity for scientific platforms on HPC infrastructure

IPDPS – JSSPP workshop

Maxime Martinasso, ETH Zurich / CSCS

June 3rd, 2025



The Swiss National Supercomputing Centre (CSCS)

- A unit of the Swiss Federal Institute of Technology Zurich (ETH Zurich)
- Data centre based in Lugano

Our mission

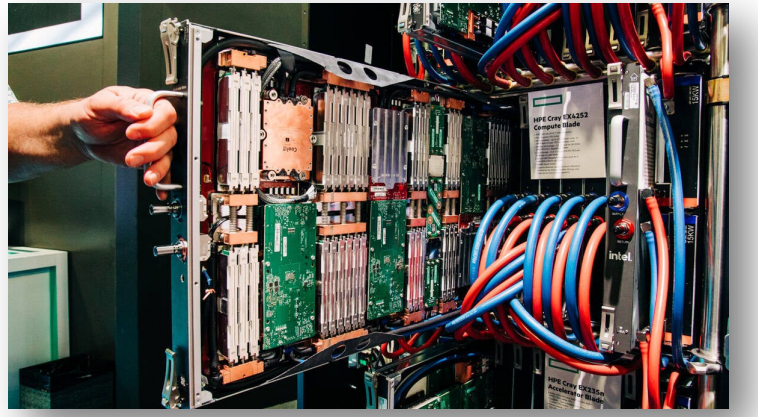
We develop and operate a High-Performance Computing and data research infrastructure that supports world-class science in Switzerland.

Alps heterogeneous hardware

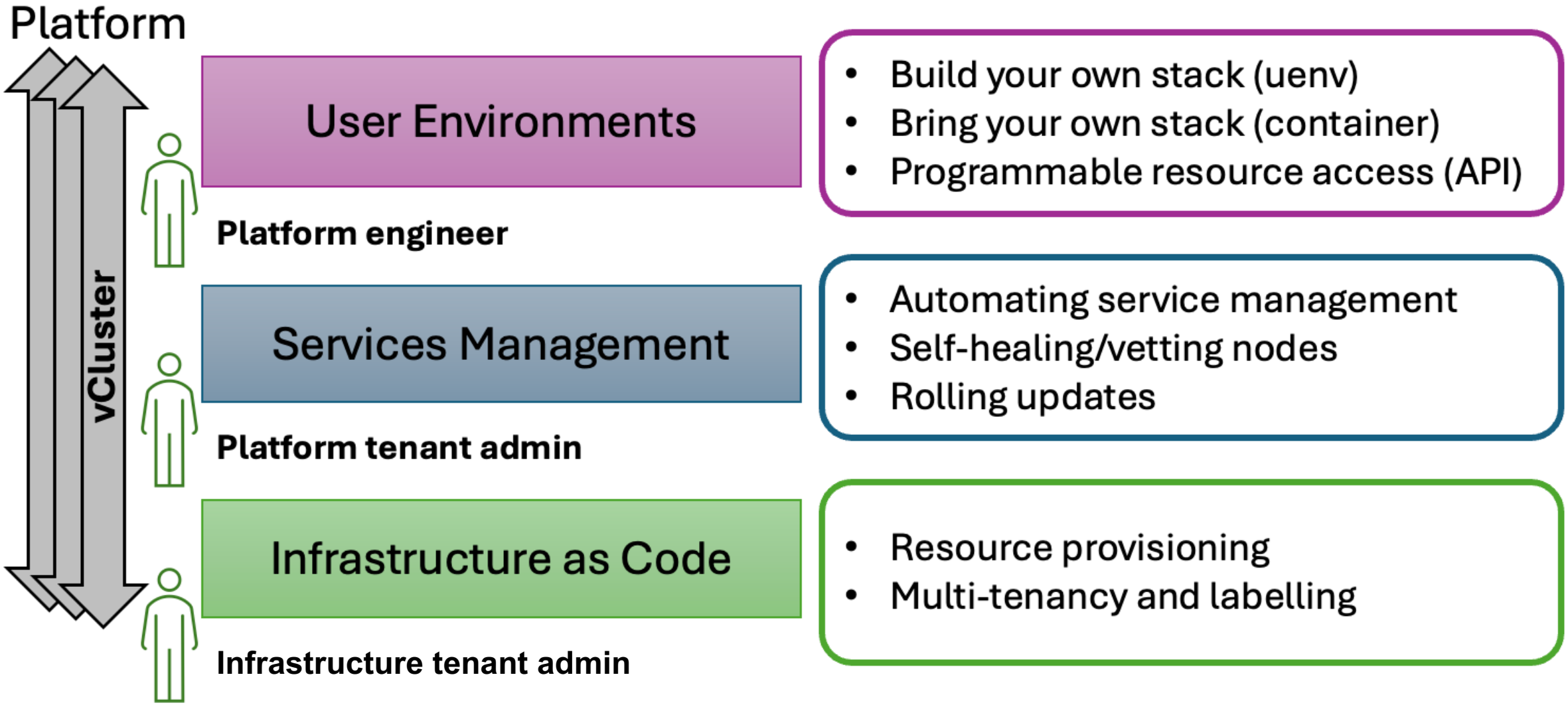
- Alps is an HPE Cray EX supercomputer being our new flagship infrastructure

  - 1024 AMD Rome-7742 nodes 256/512GB
  - 144 Nvidia A100 GPU nodes
  - 24 AMD MI250x GPU nodes (LUMI1 type)
  - **128 AMD 4xMI300A GPU nodes**
  - ***2688 4xGrace-Hopper nodes, 10752 GH200, ~430 PF***
  - Slingshot network (200 Gbps injection)
  - Two availability zones (HA, non-HA)
  - 100% liquid cooled
  - 100+10 PiB HDD
  - 5+1 PiB SSD (RAID10)
  - 100s of PiB tape library
  - ~10 MW (envelope for power and cooling)

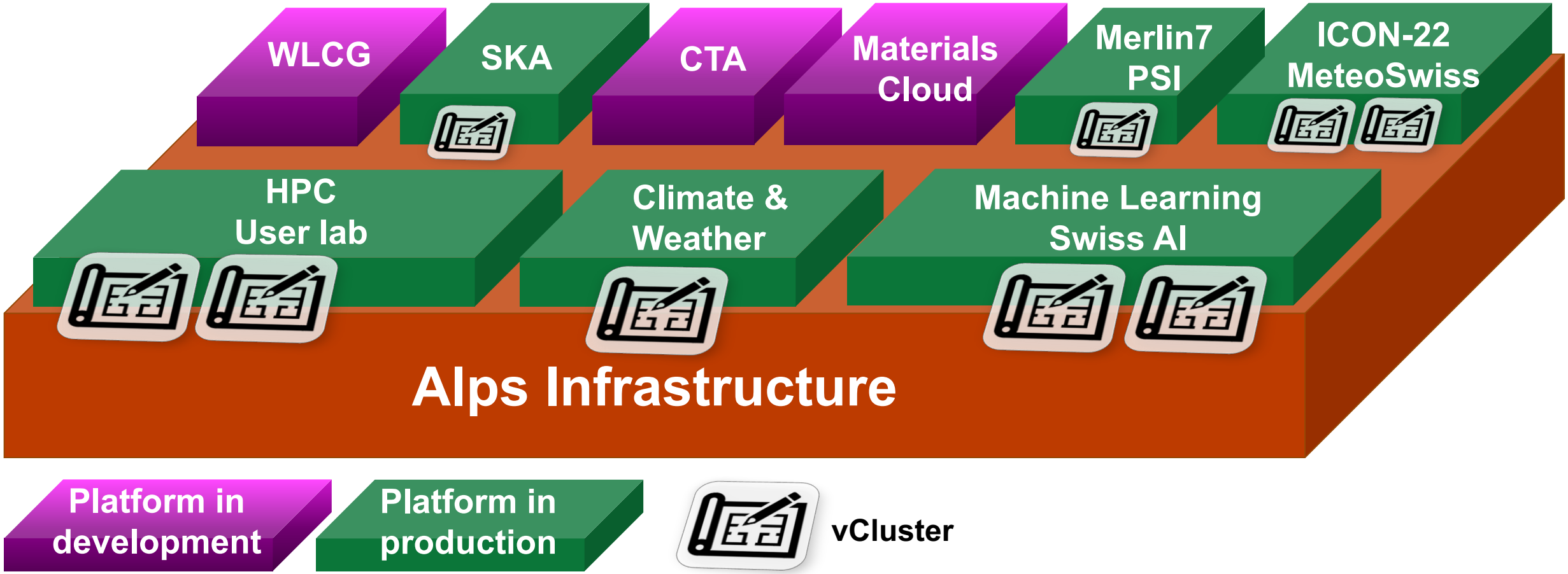
Water cooled blades



vCluster three layers **concept**



Scientific platforms



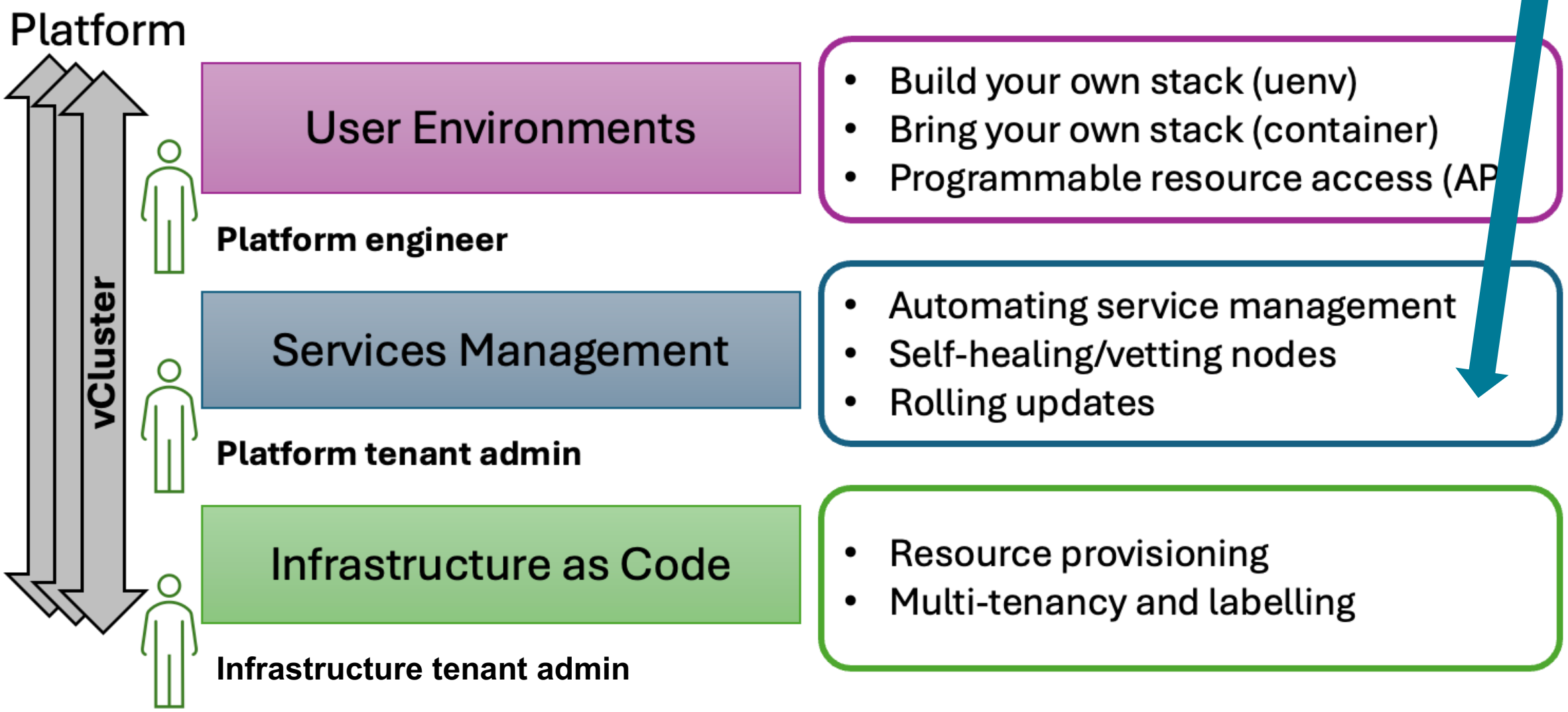
Elasticity requirements

- Platforms limit the job size
  - A platform user might want to scale outside of the platform resources
- Platforms limit the overall utilisation
  - HPC providers might want to maximize the system utilization
- Platforms should provide specific QoS

Need of elasticty among platforms, exchange of resources (nodes)

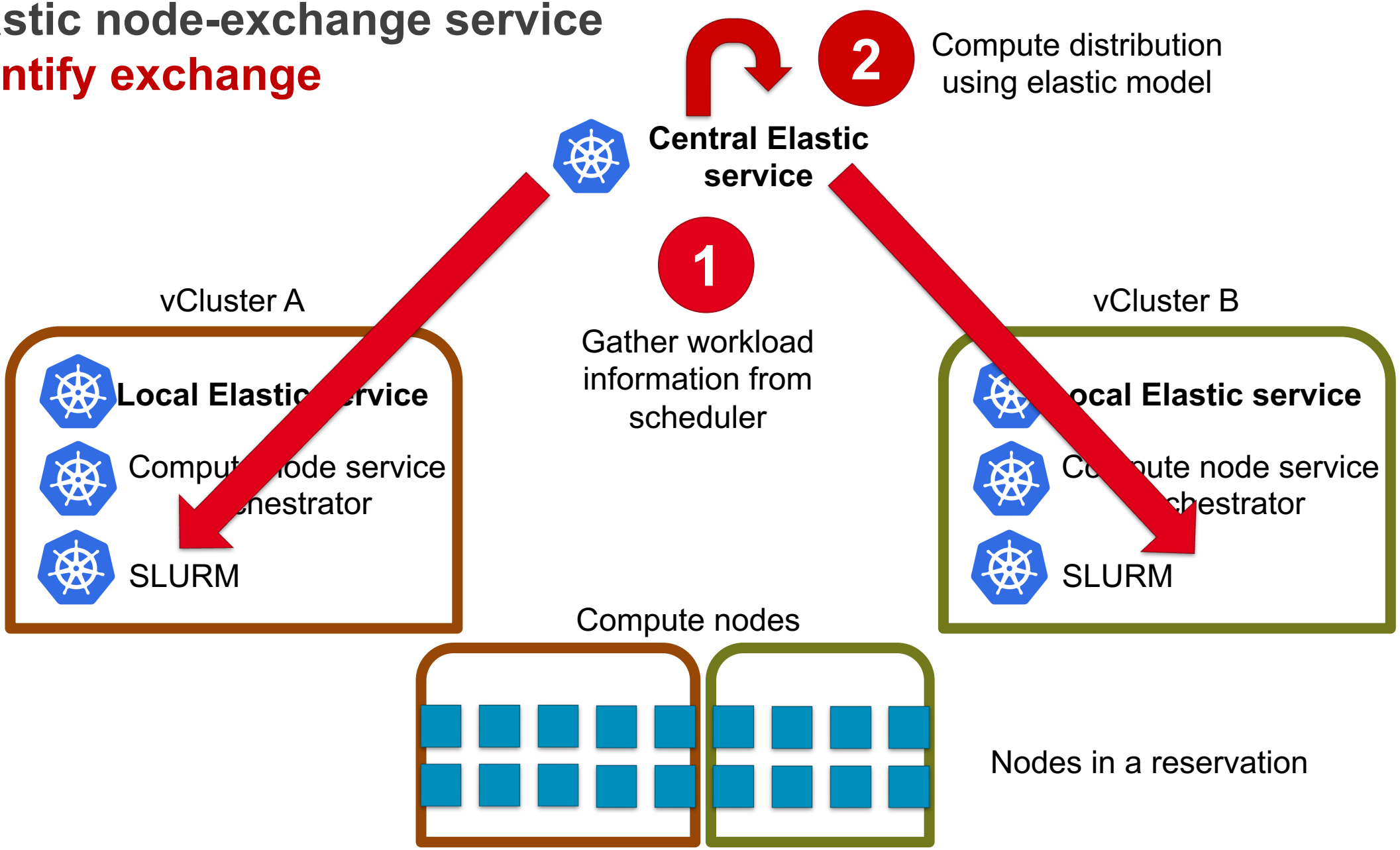
Elastic service: concepts and implementation

vCluster three layers **concept**



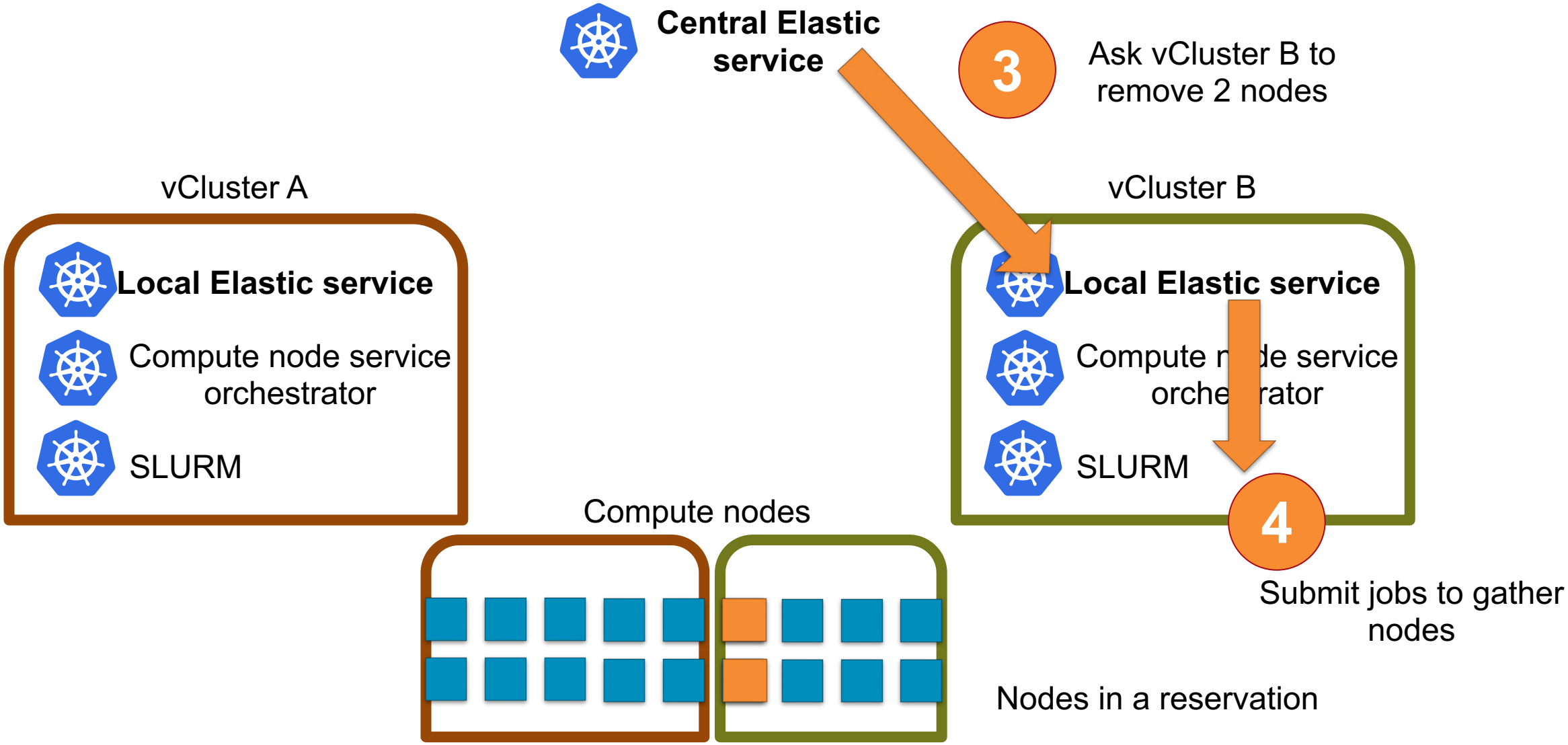
Elastic node-exchange service

Identify exchange



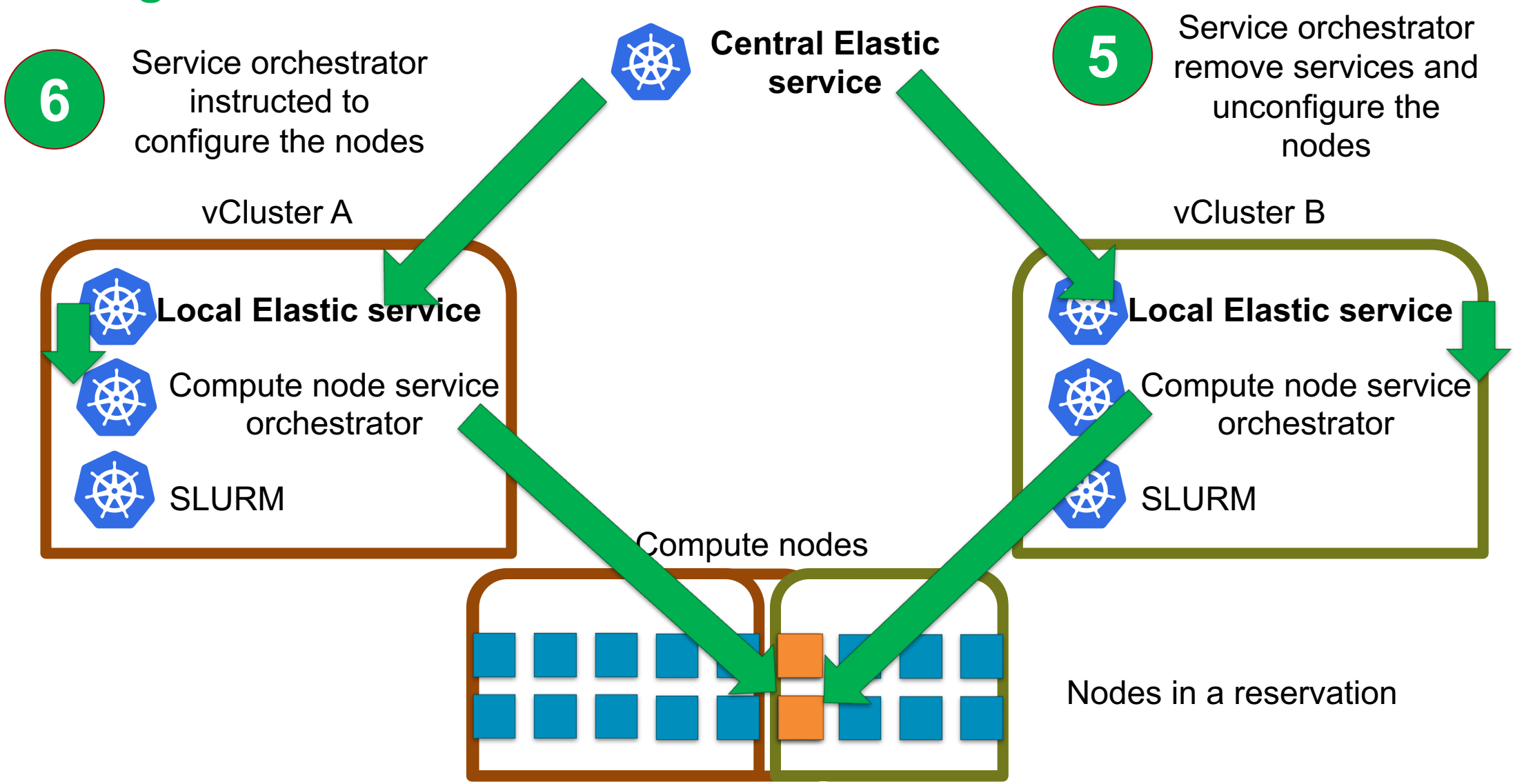
Elastic node-exchange service

Gather nodes



Elastic node-exchange service

Exchange nodes



Resource distribution model

example of a predicate function

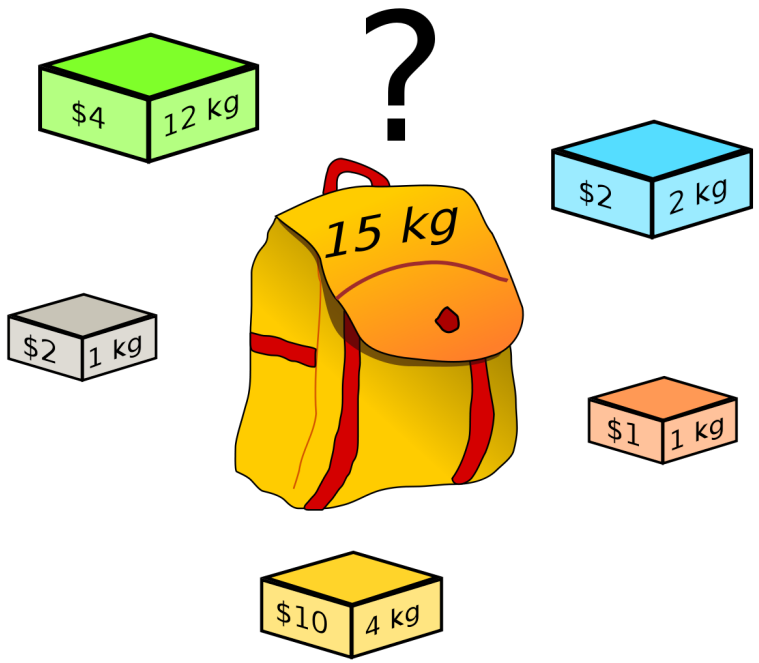
```
1  def AvgJobWaitTime(usage, n, values):
2      if 'avg_job_wait_time' not in usage:
3          return True
4      return usage['avg_job_wait_time'] < values[0]
```

Predicate function for QoS

- Define a predicate for each vCluster:
  - return **True** if the **vCluster offers nodes**
  - return **False** if the **vCluster requests nodes**
- Number of nodes based on load and the largest pending job

Resource distribution model

0-1 knapsack model



- Competing for resources, how to choose which vCluster will get nodes?
  - Weight in the bag = total offered resources
  - Weight on the boxes = specific vCluster resource request
  - Dollars on the boxes = priority among vCluster

- Priority is defined as the weighted sum of factors:
  - Fixed value (artificially manipulate priority)
  - Fair share factor
  - Request factor: the proportion of a request against the total number of requested nodes

Maximize utilization

- Remaining offered nodes are proportionally distributed based on the vClusters pending load

Experiments and results

Setup and trace

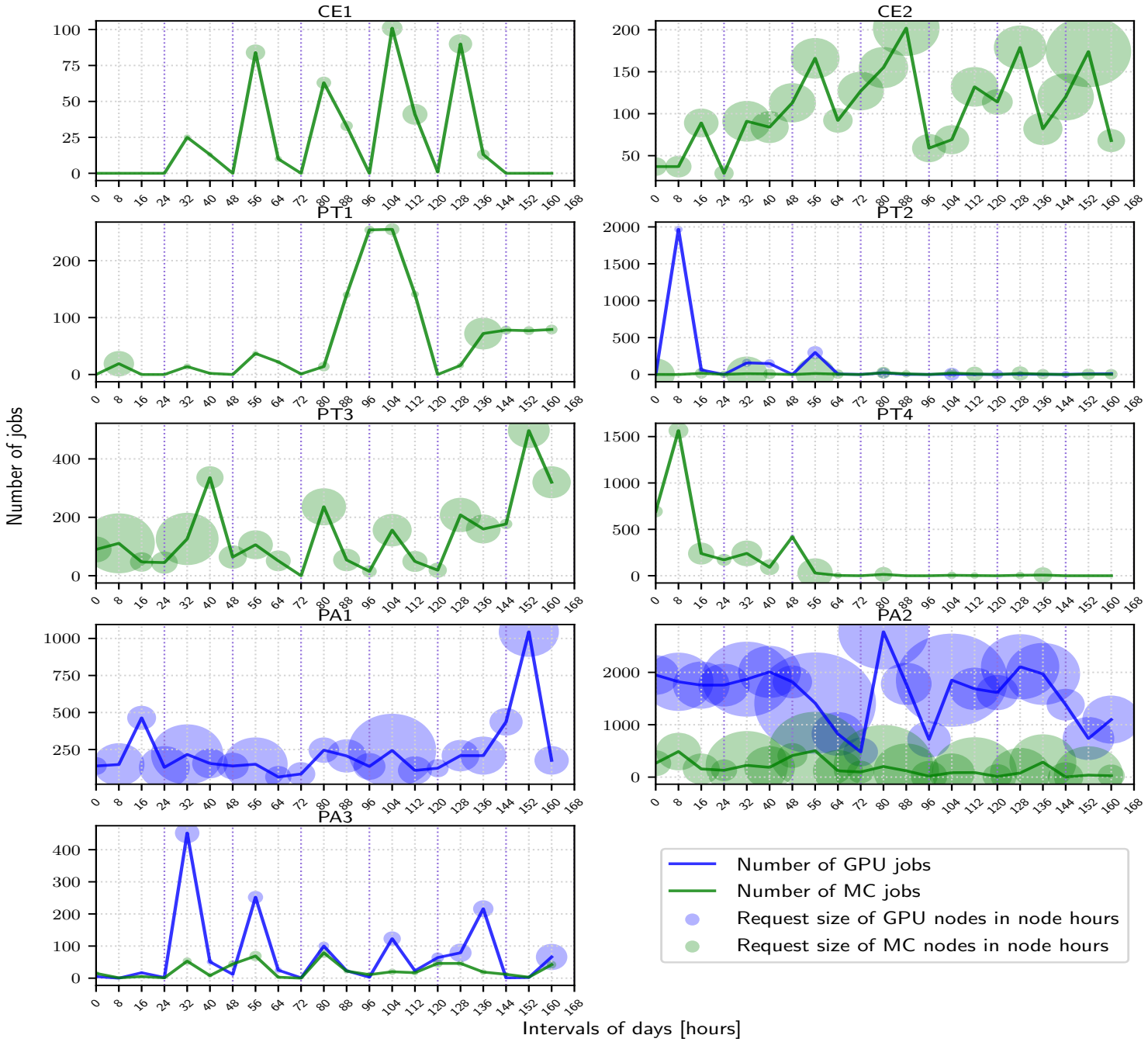
- Developed the elastic service components

- Use a Slurm-Replay[1] container for each vCluster
  - Allows to replay job submission with faster clock time (x16) and virtual nodes
  - Slurm is not modified, it is not a simulator!

- Define 9 vClusters based on business relationship
  - 2 Capital expenditure (CE): predicate function based on quota
  - 3 Project allocation (PA): predicate function always return True
  - 4 Partner institution (PT): predicate function based on average waiting time

- Different resource types depending on the vClusters

[1] Martinasso, M., et al.: RM-Replay: A High-Fidelity Tuning, Optimization and Exploration Tool for Resource Management. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 320–332. IEEE, Dallas, TX, USA (Nov 2018).



Trace – Piz Daint

- Use a real Piz Daint workload
  - 7 consecutive days
  - 43224 jobs using hybrid GPU nodes
  - 14886 jobs using multi-core nodes
- Piz Daint has 2 resource types
  - 1630 nodes Multicore (MC)
  - 5699 nodes GPU

Example of logs

Gather workloads

Compute distribution

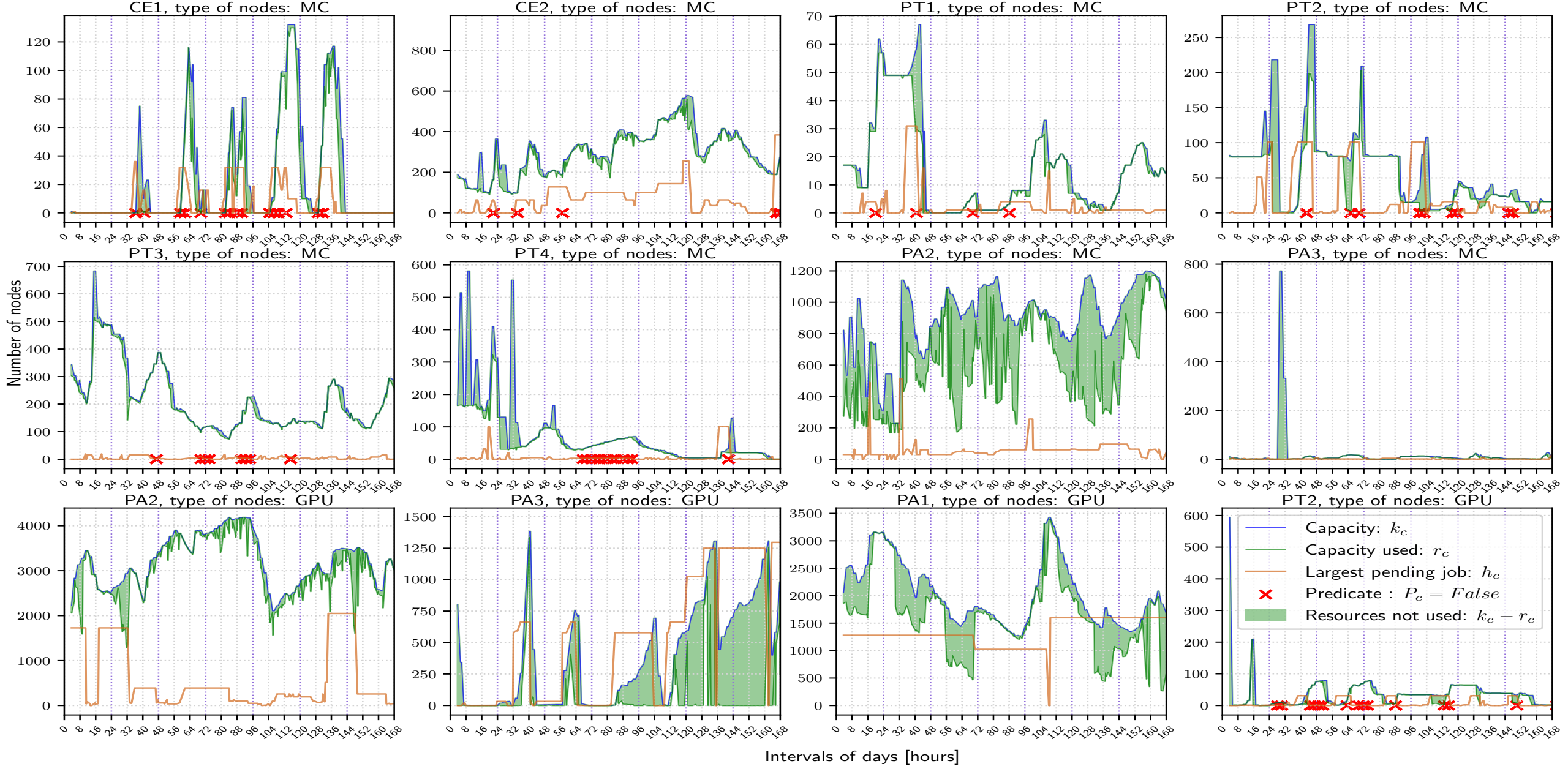
```
7] [IT82] elastic iteration=82
3] [IT82] - [2019-03-10 00:09:29] Clock div: 702 max clusters: ['pa_ul'], min clu
ity:50] [IT82] workload:
ity:52] ce_ich-success
ity:60]  mc cap=0
ity:52] ce_uzh-success
ity:60]  mc cap=275 running=263 pending=1760 max_pending=384 max_job_wait_time=13
ity:52] pa_pr-success
ity:60]  gpu cap=1687 running=563 pending=2218 max_pending=1600 max_job_wait_time
ity:52] pa_rest-success
ity:60]  mc cap=8 running=2 pending=0 max_pending=0 max_job_wait_time=0 avg_job_w
ity:60]  gpu cap=982 running=916 pending=7472 max_pending=1296 max_job_wait_time=
ity:52] pa_ul-success
ity:60]  mc cap=1031 running=807 pending=44 max_pending=40 max_job_wait_time=3417
ity:60]  gpu cap=3018 running=3004 pending=9032 max_pending=48 max_job_wait_time=
ity:52] pt_em-success
ity:60]  mc cap=14 running=14 pending=2 max_pending=1 max_job_wait_time=1423 avg_
ity:52] pt_eth-success
ity:60]  mc cap=16 running=16 pending=9 max_pending=3 max_job_wait_time=12793 avg
ity:60]  gpu cap=1 running=1 pending=2 max_pending=1 max_job_wait_time=9337 avg_j
ity:52] pt_mr-success
ity:60]  mc cap=286 running=244 pending=0 max_pending=0 max_job_wait_time=0 avg_j
ity:52] pt_usi-success
ity:60]  mc cap=0
ter:135] -#- resource allocation begin 82 -#-
ter:167] offers of clusters without a workload: mc [['pa_rest', 6], ['pt_mr', 42]
ter:218] predicate values: gpu {'pa_pr': True, 'pa_rest': True, 'pa_ul': True, 'p
ter:220] offers: gpu [['pa_pr', 253], ['pa_rest', 147], ['pa_ul', 452]]
ter:218] predicate values: mc {'ce_uzh': False, 'pa_ul': True, 'pt_em': True, 'pt
ter:220] offers: mc [['pa_rest', 6], ['pt_mr', 42], ['pa_ul', 184], ['pt_em', 2],
ter:222] requests: mc [['ce_uzh', 236]]
```

Example of logs

Transfer requests

Execution of requests and effective node transfers

```
ns:121] [IT82] Transfer requests
ns:123] [IT82] [TR] pt_em->ce_uzh 2mc knapsack
ns:123] [IT82] [TR] pt_eth->ce_uzh 2mc knapsack
ns:123] [IT82] [TR] pa_rest->ce_uzh 6mc knapsack
ns:123] [IT82] [TR] pt_mr->ce_uzh 42mc knapsack
ns:123] [IT82] [TR] pa_ul->ce_uzh 184mc knapsack
ns:123] [IT82] [TR] pa_ul->pa_rest 42gpu proportional
ns:123] [IT82] [TR] pa_pr->pa_rest 150gpu proportional
arbiter:289] remaining_nodes: {'gpu': 852, 'mc': 0}
arbiter:314] -#- resource allocation end 82 -#-
transfer:11] -#- execute transfer begin 82 -#-
ec:25] Task 806a3a92f9984aee82c1379fef6d2913: started
ec:25] Task 41cfb528a24949e18b2e7bb7e02aa0b1: started
ec:25] Task 686e913d4f1943c6b32a27f636318095: started
ec:25] Task 6e7abe5191974c34840bf8c20f6a3df2: started
ec:25] Task cbc704552373459fa05ac49422af822d: started
ec:25] Task 5b09b10dbdba4bf5a57c2e266d508a44: started
ec:25] Task 39bc5cddf3f849eaaeb96b8bd49f3164: started
transfer:28] total submitted requests: 7
ec:25] Task 806a3a92f9984aee82c1379fef6d2913 is in progress, 0 over 2
ec:25] Task 41cfb528a24949e18b2e7bb7e02aa0b1 is in progress, 0 over 2
ec:25] Task 686e913d4f1943c6b32a27f636318095 is completed, 6 over 6
ec:25] nodes "nid[00007,00174-00175,00179,00382,00389]" sets to RESUME on ce_uzh
ec:25] Task 686e913d4f1943c6b32a27f636318095: nid[00007,00174-00175,00179,00382,00389
transfer:121] -- transfer pa_rest -> ce_uzh completed.
ty:23] Timing: transfer_swap_nodes = 0.3 s
ec:25] Task 6e7abe5191974c34840bf8c20f6a3df2 is in progress, 0 over 42
ec:25] Task cbc704552373459fa05ac49422af822d is in progress, 20 over 184
ec:25] Task 5b09b10dbdba4bf5a57c2e266d508a44 is in progress, 0 over 42
ec:25] Task 39bc5cddf3f849eaaeb96b8bd49f3164 is in progress, 20 over 150
transfer:56] check: it=0, n_completed=1/7, n_failed=0
ec:25] Task 806a3a92f9984aee82c1379fef6d2913 is in progress, 0 over 2
ec:25] Task 41cfb528a24949e18b2e7bb7e02aa0b1 is in progress, 0 over 2
ec:25] Task 6e7abe5191974c34840bf8c20f6a3df2 is completed, 42 over 42
ec:25] nodes "nid[00035,00050,00286,00355,00368,00499,00534,00584,00720,00772,00866,0
ec:25] Task 6e7abe5191974c34840bf8c20f6a3df2: nid[00035,00050,00286,00355,00368,00499
transfer:121] -- transfer pt_mr -> ce_uzh completed.
```



Utilisation GPU: **84.6% → 83.4%** Utilisation MC: **73.2% → 72.8 %** 6101 nodes exchanged per day

QoS for Partnership vCluster

COMPARING STATISTICS OF WAITING TIME IN HOURS OF JOBS WAITING MORE THAN 4 HOURS.

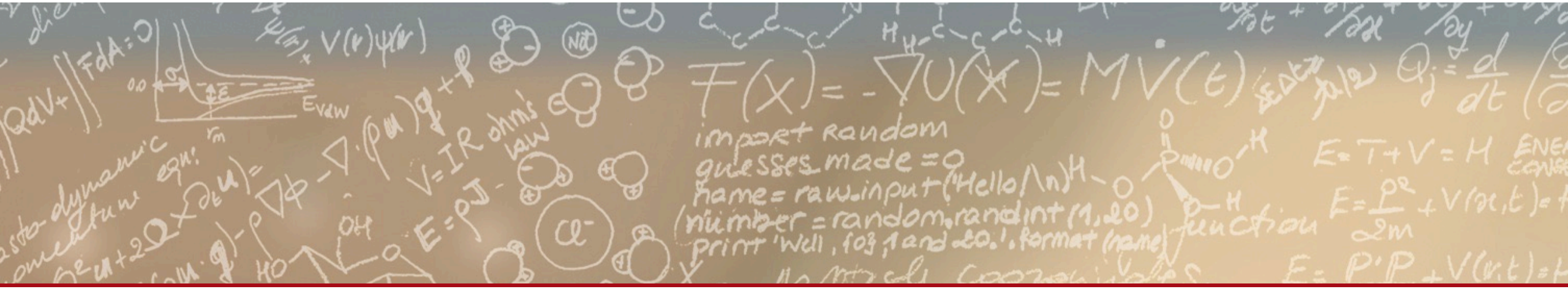
Cluster instance	count	mean	med	std	min	max
Original PT1	22	68.4	68.8	46.6	10.1	143.5
Replayed PT1	328	5.0	5.0	0.8	4.0	13.8
Original PT2	0	0.0	0.0	0.0	0.0	0.0
Replayed PT2	49	5.8	5.7	1.1	4.1	10.5
Original PT3	13	43.4	39.4	25.6	5.8	87.8
Replayed PT3	1133	5.9	5.6	1.3	4.0	9.2
Original PT4	98	16.5	8.7	12.2	4.4	52.2
Replayed PT4	59	9.4	7.2	5.3	4.6	22.1

Conclusion and future work

- Resource elasticity is possible among vClusters
  - Lead to a good overall utilisation
  - Enable jobs to scale outside of tenant resources
  - More reliable QoS for vClusters

- Explore improvements
  - Granularity of the number of nodes that can be exchange
  - Consider job runtime expectation in the model

- Integration into Alps
  - Find the right paramaters of the model and predicates
  - Improve reconfiguration time (now in ~10 minutes, target is 1-2 minutes)



Thank you for your attention.