Performance and Energy Aware Kubernetes Scheduler

Han Dong, *Hamilton College, USA*
Parul Singh, *Red Hat Inc, Netherlands*
Yara Awad, *Boston University, USA*
Felix George, *IBM Research, India*
Krishnasuri Narayanam, *IBM Research, India*
Sanjay Arora, *Red Hat Inc, USA*
Jonathan Appavoo, *Boston University, USA*

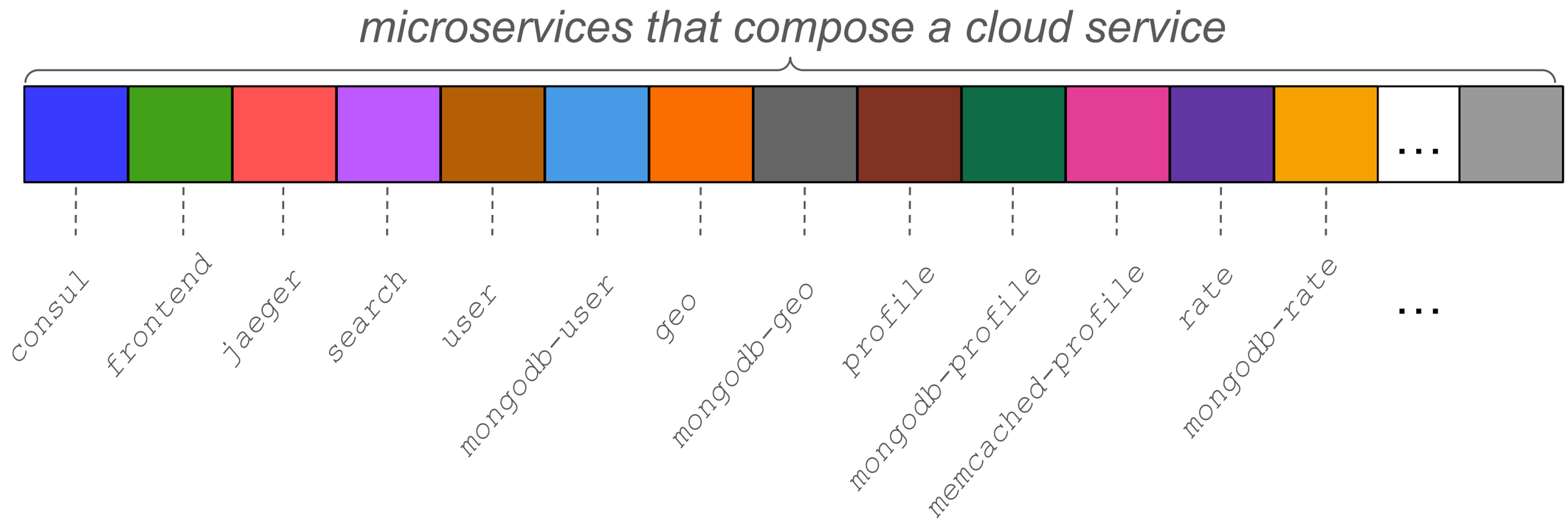
Performance and Energy Aware Kubernetes Scheduler

Advocating for better hardware awareness in cloud-scale deployments..

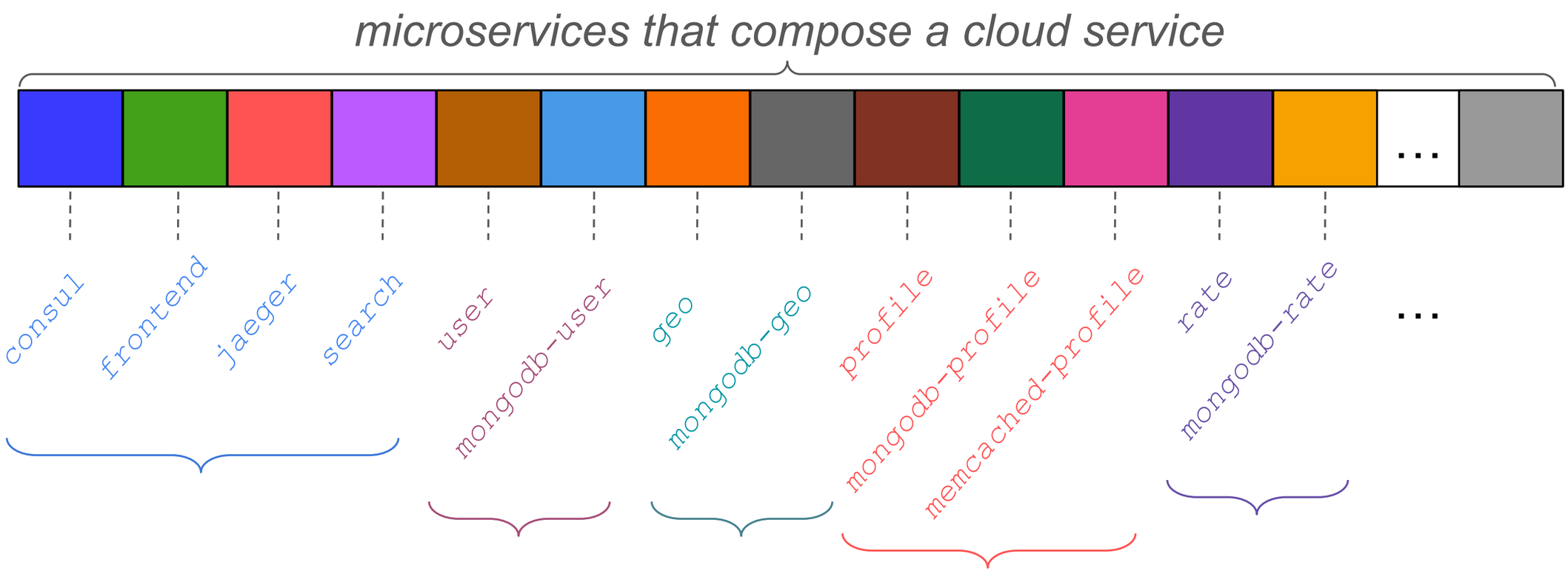
Exposing and exploiting hardware uniqueness..

- Challenges of managing microservices in the cloud
  - Explosive configuration space
- State-of-the-art configuration approaches
  - Kubernetes HPA, Cliantro Research System
- PAX approach and evaluation
  - Configuration via Black-Box Bayesian Optimization

Microservices Cloud Deployments

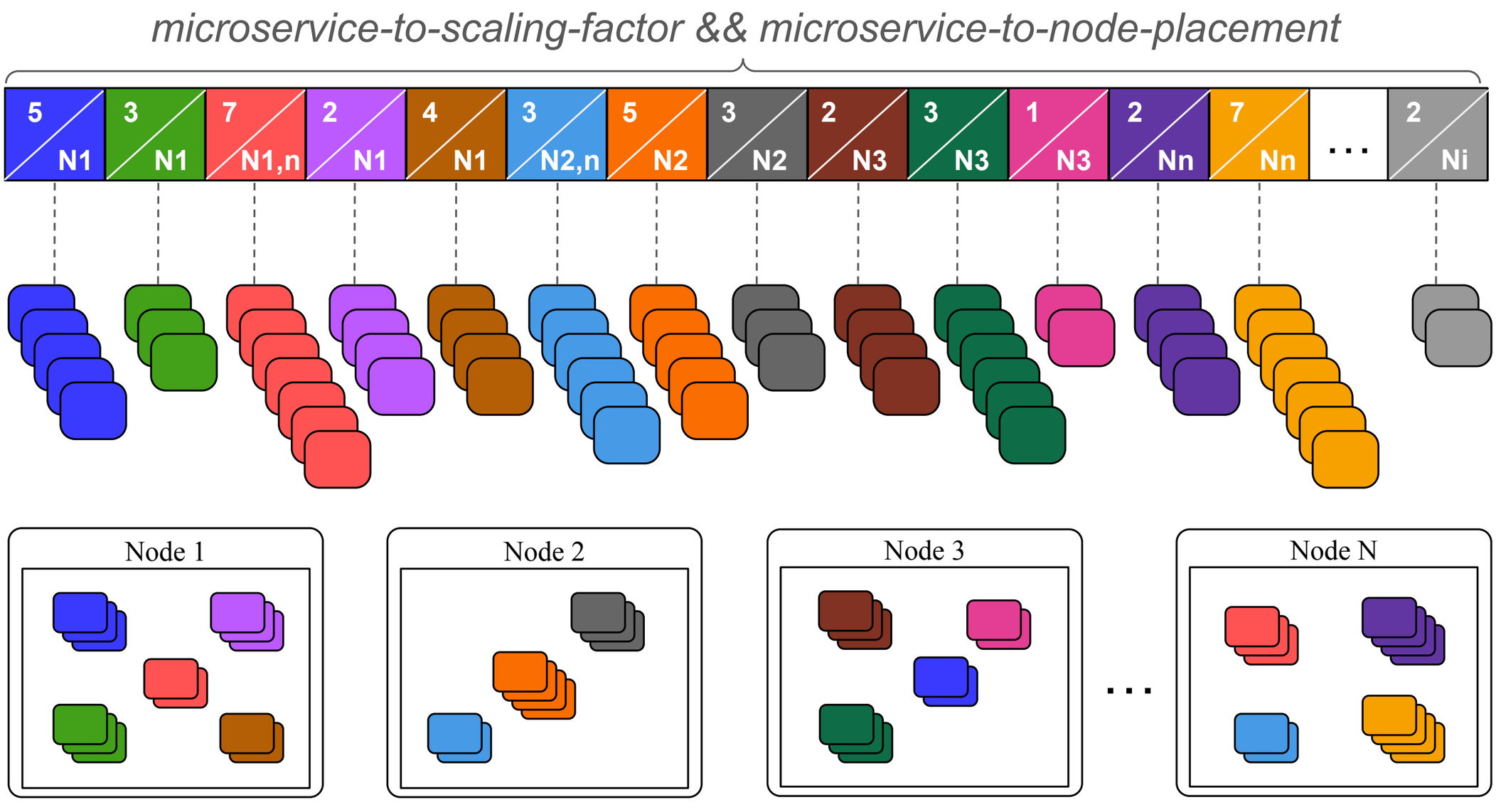


Microservices Cloud Deployments

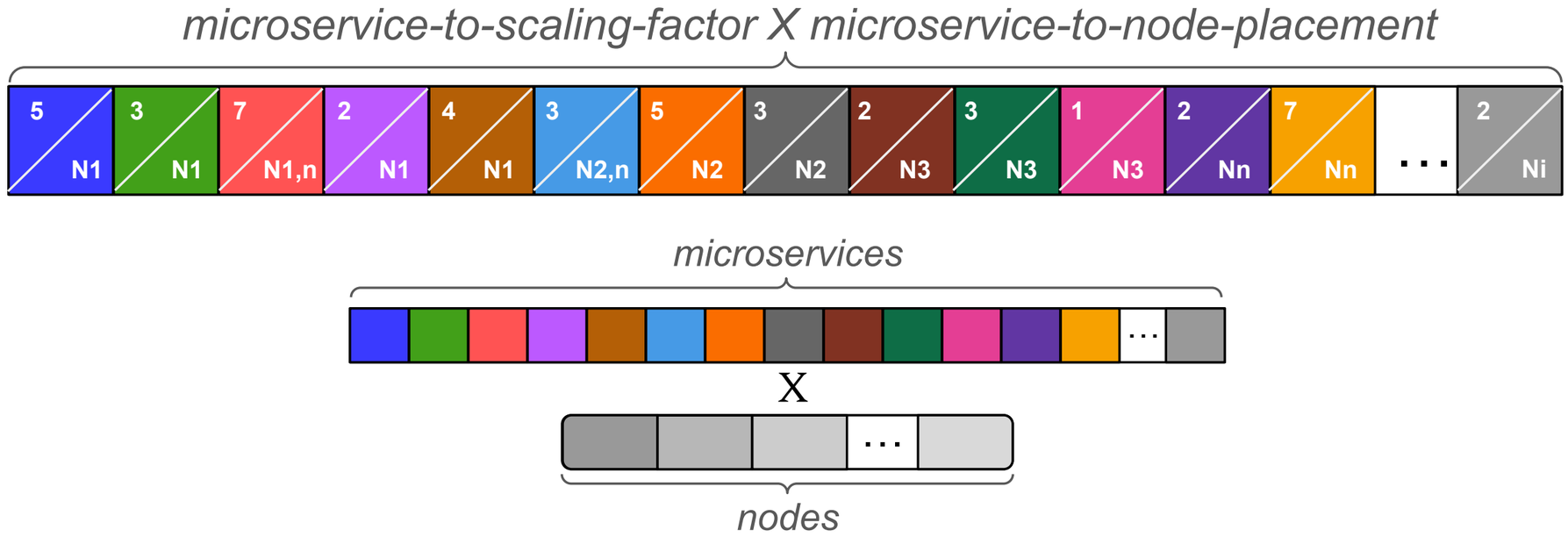


Interdependence *between microservices forces configuration to be the* ***joint configuration of all microservices****.*

Microservices Configuration: Scaling + Placement



Configuration Space Explodes



	5	3	1	2	4	0	0	0	0	0	0	0	0	…	0
	0	0	0	0	0	2	5	3	0	0	0	0	0	…	0
	0	0	0	0	0	0	0	0	2	3	1	0	0	…	0
⋮	..	..	..	..	..	..	..	..	..	..	..	..	..		..
	0	0	6	0	0	1	0	0	0	0	0	2	7	…	0

Constraints (s_i , p_i)

$1 \le s_i \le CPU_{max}$

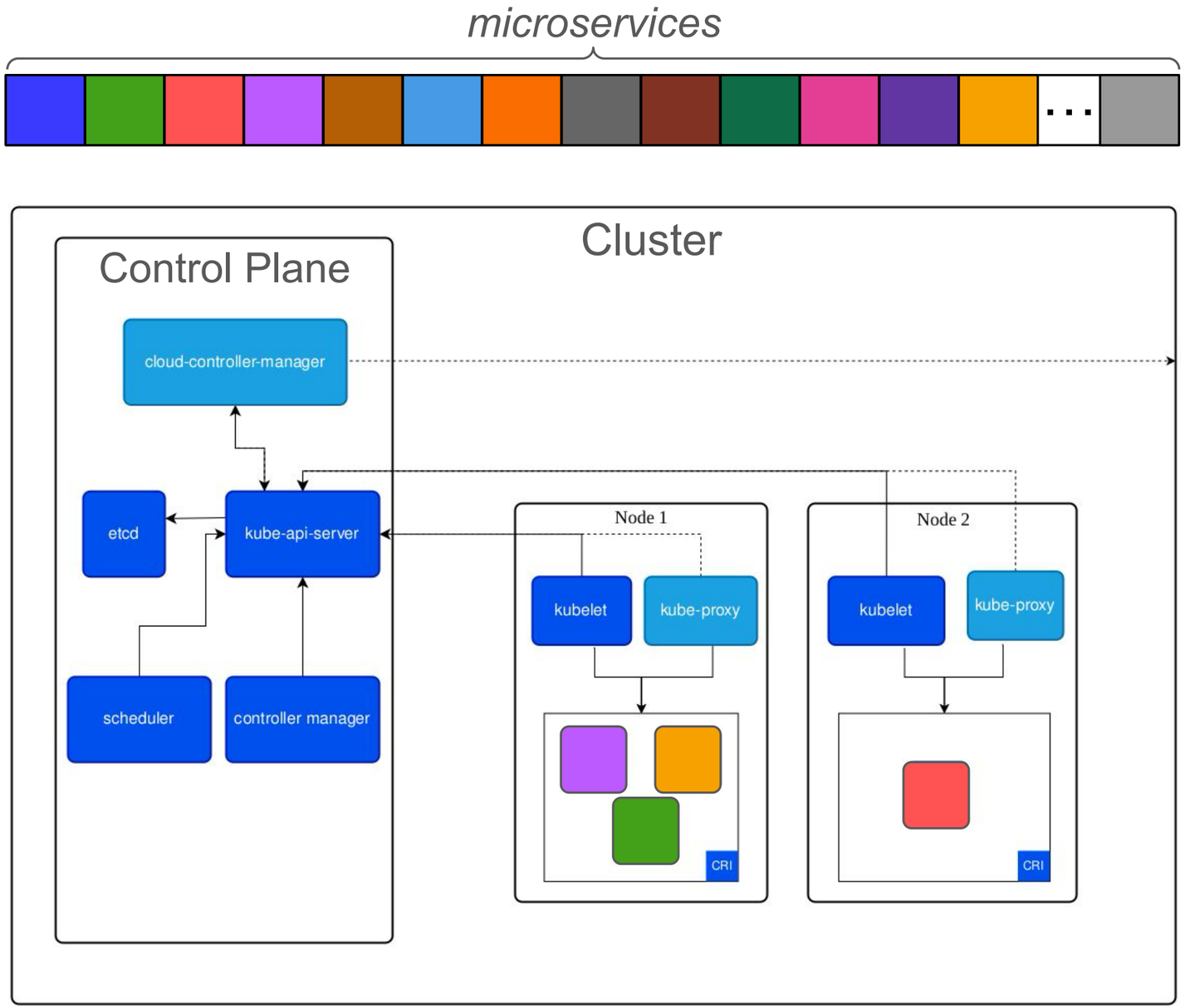
$1 \le p_i \le NODE_{max}$

Kubernetes Deployment

Kubernetes: a container-based cloud deployment platform

pod: set of processes belonging to one microservice

pod-replica: instance (or copy) of a microservice (created or destroyed as a result of scaling)

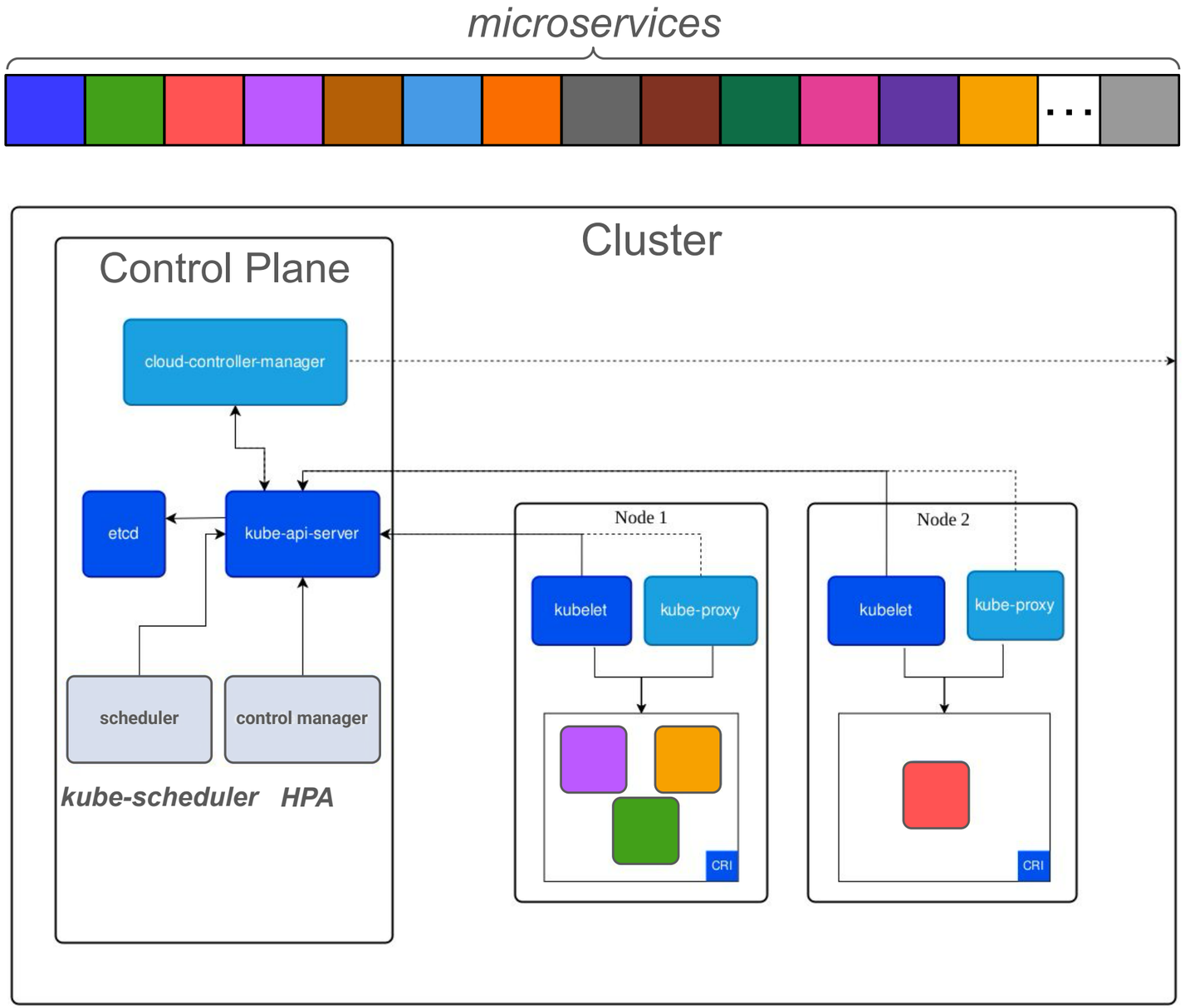


Kubernetes Deployment

Control plane manages the scaling and scheduling of microservices.

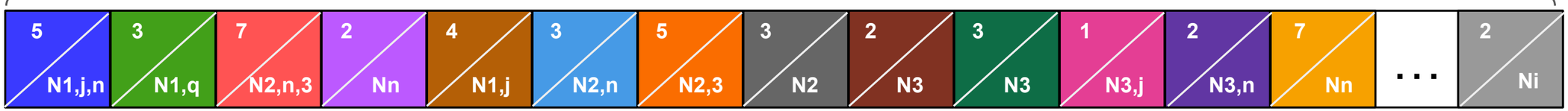
HPA: **h**orizontal **p**od **a**utoscaler - default pod-to-replica mechanism

Kube-Scheduler: default pod-to-node placement mechanism



Current Approaches: Decoupling Scaling and Placement

microservice-to-scaling-factor ~~X~~ + microservice-to-node-placement



Scaling Configuration (1xK vector)

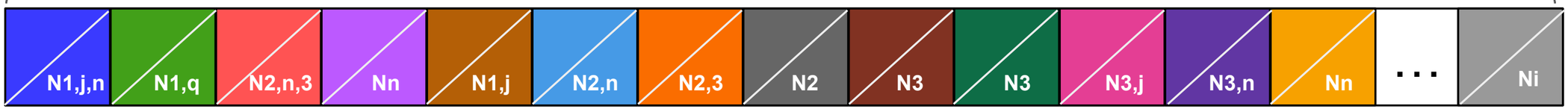
microservices (or pod-types)

Placement Configuration (1xK' vector)

instances (or pod replicas)

Current Approaches: Decoupling Scaling and Placement

microservice-to-scaling-factor ~~X~~ + microservice-to-node-placement



Scaling Configuration (1xK vector)

5	3	7	2	4	3	5	3	2	3	1	2	7	…	2

microservices (or pod-types)

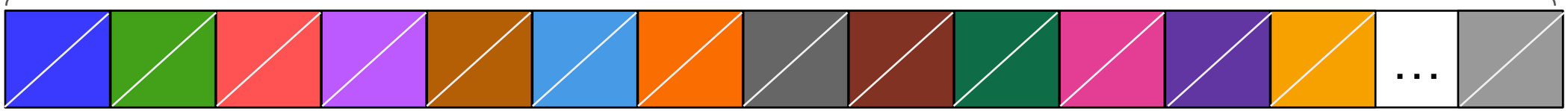
Placement Configuration (1xK' vector)

																…	

instances (or pod replicas)

Current Approaches: Decoupling Scaling and Placement

microservice-to-scaling-factor ~~X~~ + microservice-to-node-placement



Scaling Configuration (1xK vector)

5	3	7	2	4	3	5	3	2	3	1	2	7	…	2

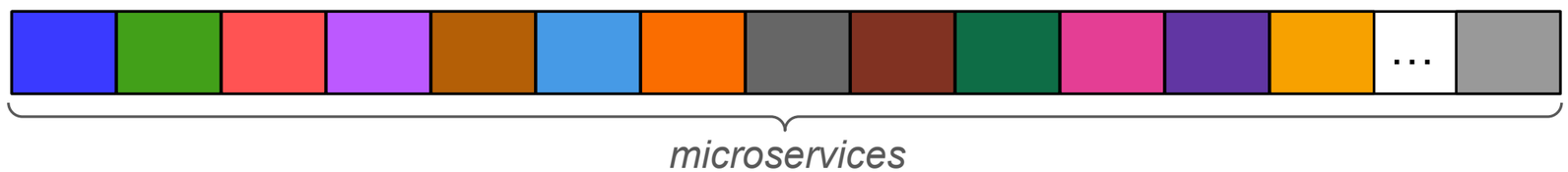
microservices (or pod-types)

Placement Configuration (1xK' vector)

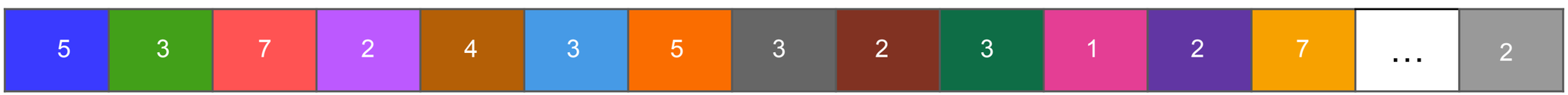
N1	N1	N1	Nj	Nn	N1	N1	Nq	N2	Nn	N2	N2	N3	N3	N3	Nn	Nn	…	Ni

instances (or pod replicas)

Current Approaches: Decoupling Scaling and Placement



Scaling Configuration (1xK vector)

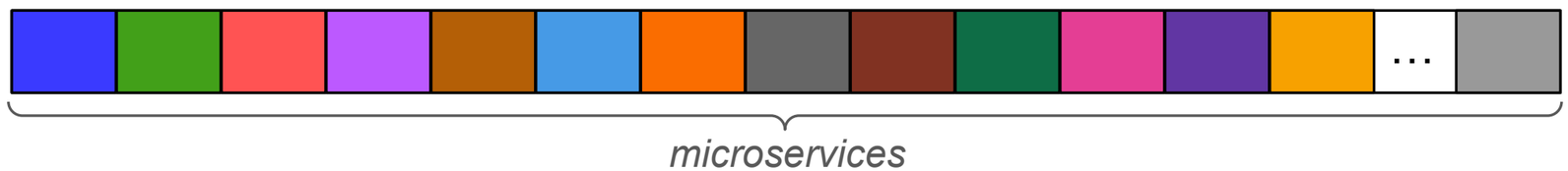


Integer Partitioning Problem

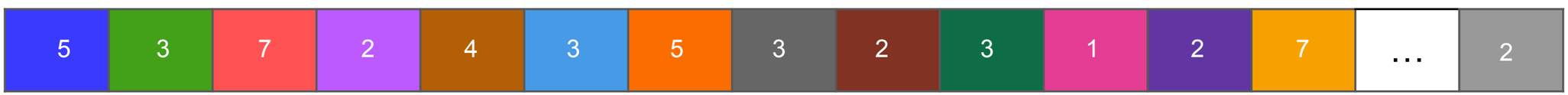
Placement Configuration (1xK' vector)

$O(N^{K'})$ pod-to-node configurations (N = number of nodes, K' = total number of pods)

Current Approaches: Decoupling Scaling and Placement



Scaling Configuration (1xK vector)



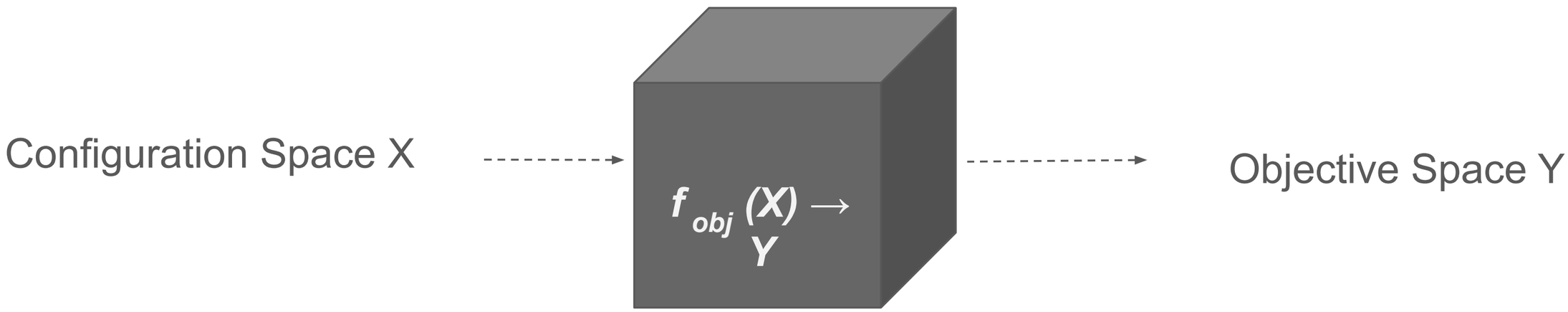
HPA: user-defined parameters + heuristic-based algorithms

Cilantro: **optimization policy searches configuration space** for optimal pod-to-CPUs configuration

Placement Configuration (1xK' vector)

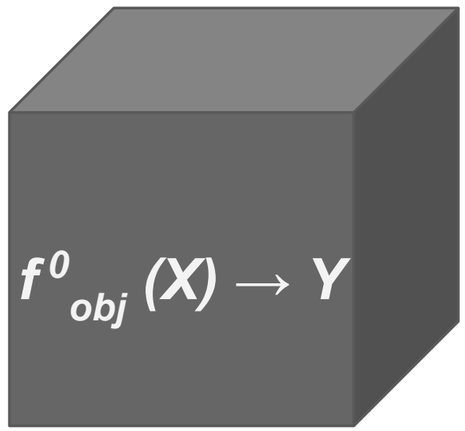
kube-scheduler places pod replicas on *suitable* and *available* nodes

Cilantro: Black Box Optimization *(Derivative Free Optimization)*



Cilantro: Black Box Optimization *(Derivative Free Optimization)*

Configuration Space X



Objective Space Y

μservice	# Replicas
A	3
B	1
…	…
K	7

X^0

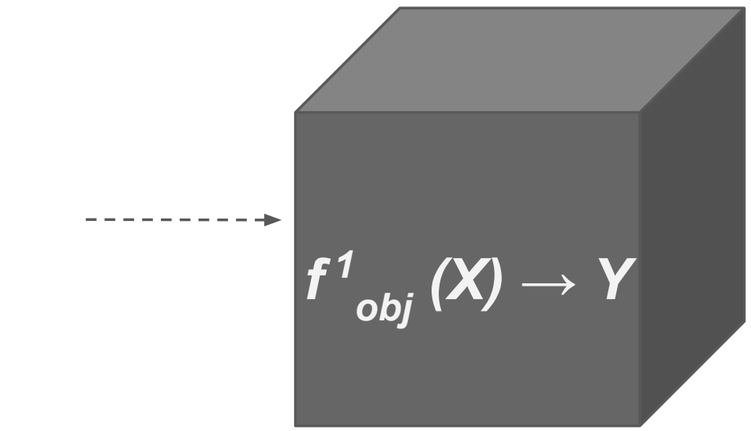
Objective Function
is updated at every iteration
until it yields an optimal
objective

Utility/ Performance Metric	Value
Average P99 Latency	1705 ms
…	…
Utilization	75%

Y^0

Cilantro: Black Box Optimization *(Derivative Free Optimization)*

Configuration Space X



Objective Space Y

μservice	# Replicas
A	4
B	1
…	…
K	3

X^1

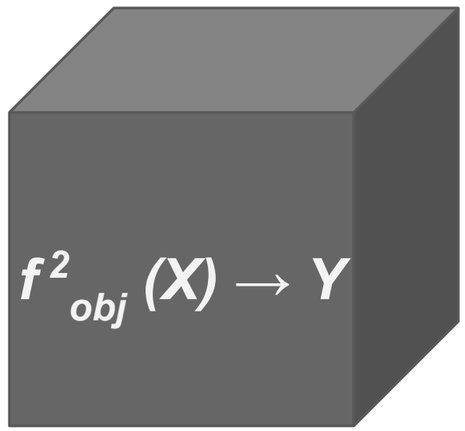
Objective Function is updated at every iteration until it yields an optimal objective

Utility/ Performance Metric	Value
Average P99 Latency	1930 ms
…	…
Utilization	70%

Y^1

Cilantro: Black Box Optimization *(Derivative Free Optimization)*

Configuration Space X



Objective Space Y

μservice	# Replicas
A	8
B	3
…	…
K	2

X^2

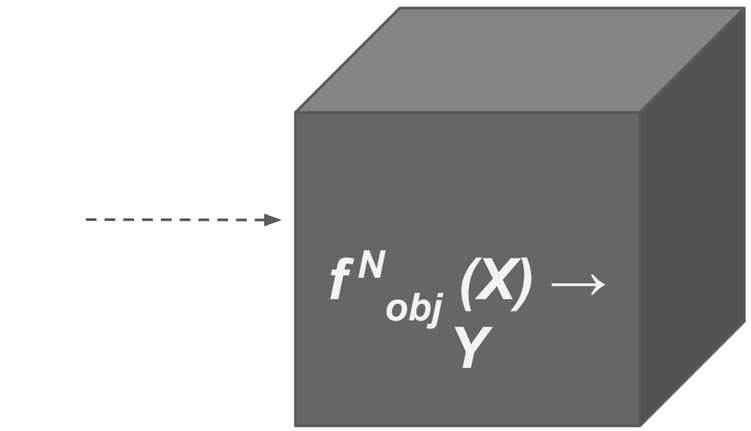
Objective Function
is updated at every iteration
until it yields an optimal
objective

Utility/ Performance Metric	Value
Average P99 Latency	1763 ms
…	…
Utilization	73%

Y^2

Cilantro: Black Box Optimization *(Derivative Free Optimization)*

Configuration Space X



Objective Space Y

μservice	# Replicas
A	4
B	2
…	…
K	6

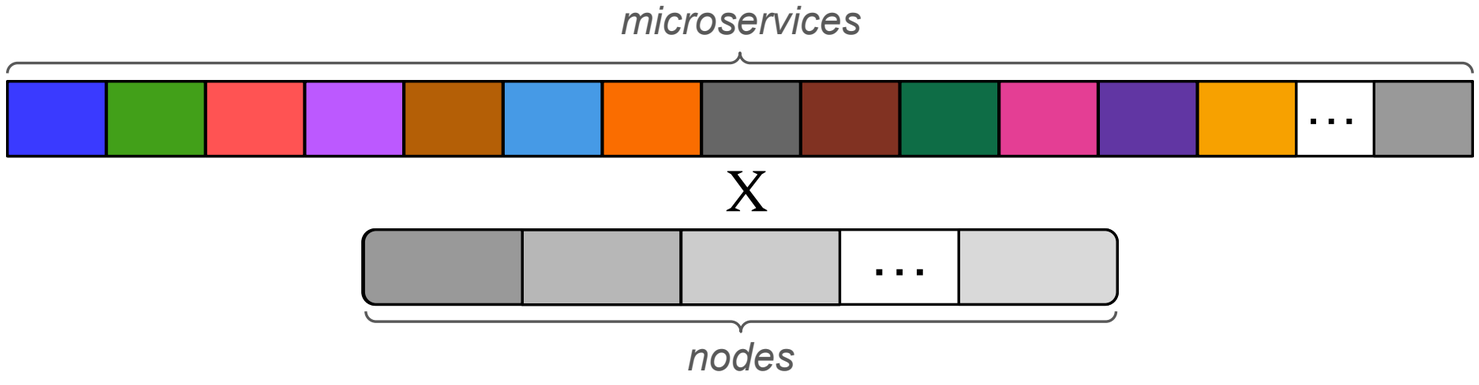
$\mathbf{X^N}$

f^N_{obj}

Utility/ Performance Metric	Value
Average P99 Latency	1590 ms
…	…
Utilization	82%

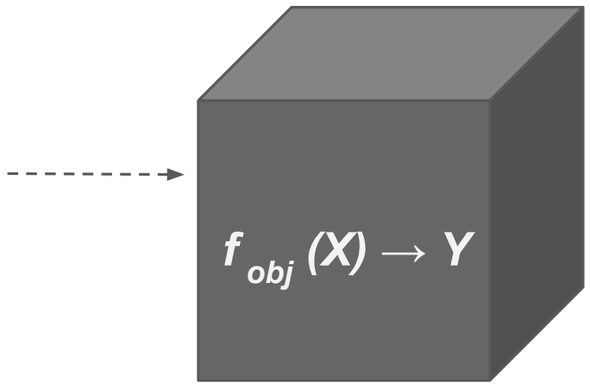
$\mathbf{Y^N}$

PAX Black Box Optimization: *Coupling Scaling and Placement*



μservice	# Replicas	Node
A	3	2
B	1	13
…	…	…
K	7	13

X



Utility/ Performance Metric	Value
Average P99 Latency	1705 ms
…	…

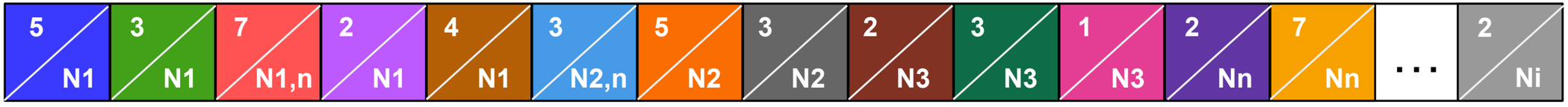
Y

Bayesian Optimization

sample-efficient black-box optimization

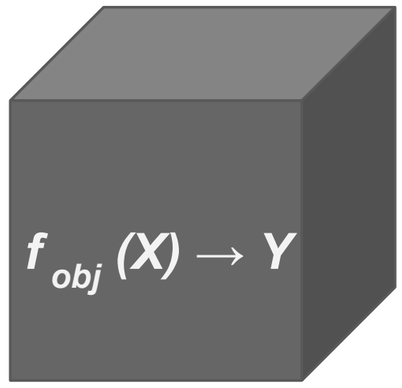
PAX Pod Scheduler: Coupling Scaling and Placement

microservice-to-scaling-factor X microservice-to-node-placement



μservice	# Replicas	Node
A	3	2
B	1	13
…	…	…
K	7	13

X



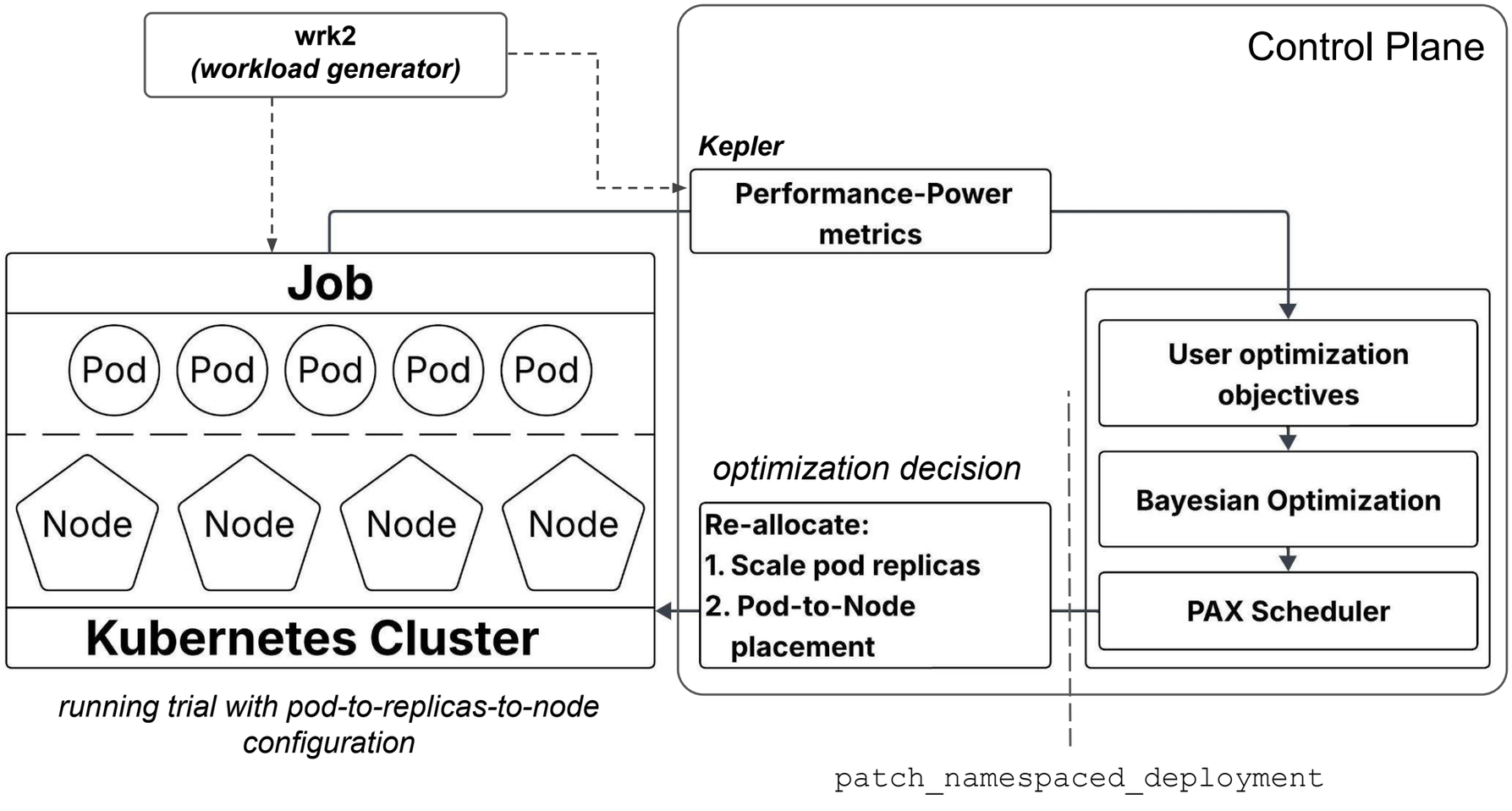
Utility/ Performance Metric	Value
Average P99 Latency	1705 ms
…	…

Y

Bayesian Optimization

sample-efficient black-box optimization

PAX Experimental Setup



PAX, HPA, Cilantro Comparison

Benchmark: HotelReservation (from DeathStarBench)
Hardware Cluster: 3 different clusters each with a total of 128 allocatable CPU

Cluster A	Cluster B	Cluster C
4x c220g2 nodes	2x sm220u nodes	2x c220g2 + 1x sm220u nodes

Name	Processor (Intel)	Node	Release	CPUs	TDP (W)	NIC	RAM	SSD	CO2 (kg)	Cost
c220g2	E5-2630 v3	22 nm	Q3'14	2 x 16	2 x 85	10GbE	128GB	480 GB	118.4	$599 [21]
sm220u	Xeon Silver 4314	10 nm	Q2'21	2 x 32	2 x 135	40GbE	256GB	960 GB	221.9	$6080 [5]

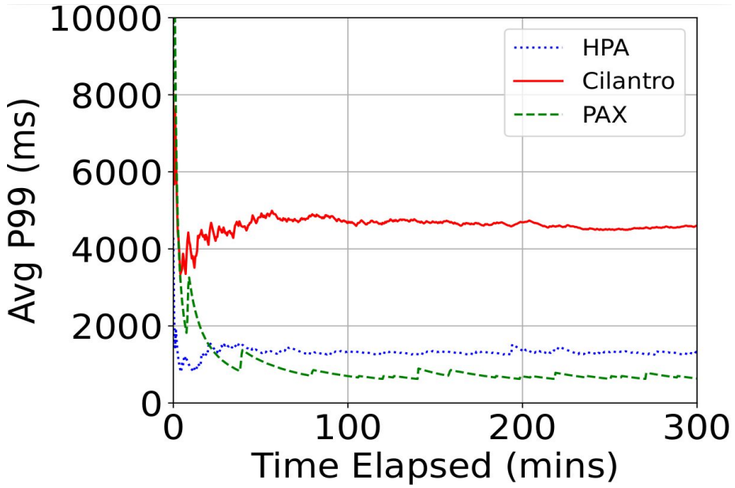
Table 1. Different hardware explored.

PAX, HPA, Cilantro Comparison

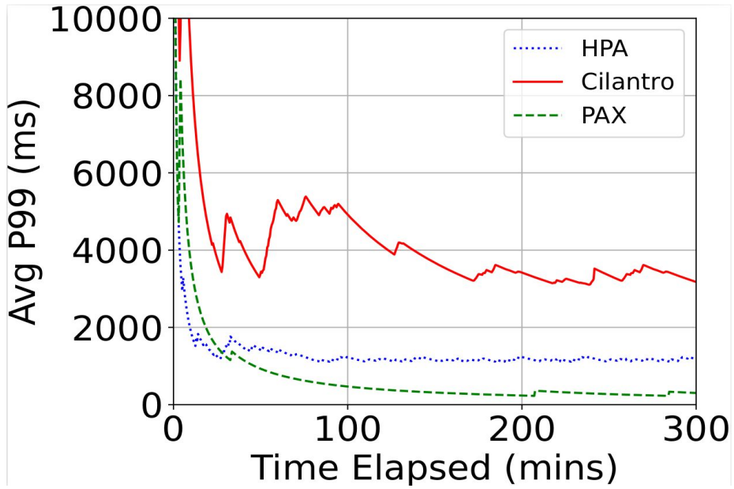
Benchmark: HotelReservation (from DeathStarBench)

Hardware Cluster: 3 different clusters each with a total of 128 allocatable CPU

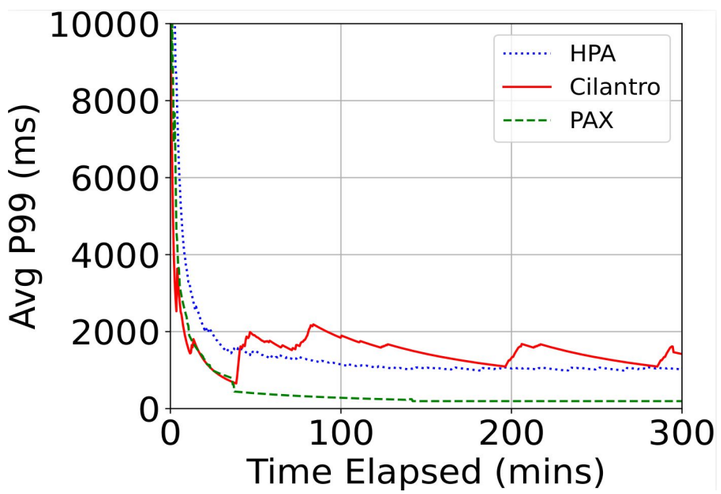
Cluster A (*older*)



Cluster B (*newer*)



Cluster C (*hybrid*)

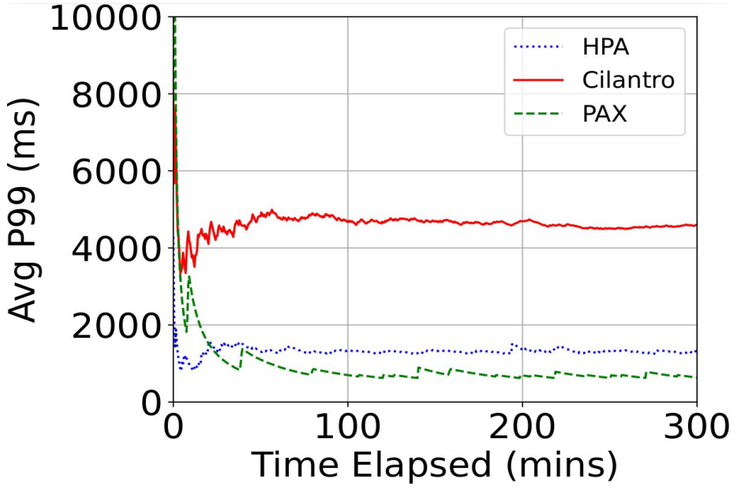


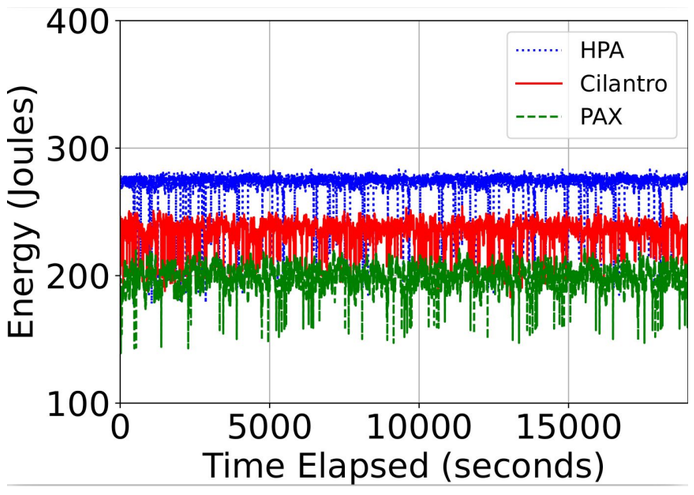
Name	Processor (Intel)	Node	Release	CPUs	TDP (W)	NIC	RAM	SSD	CO2 (kg)	Cost
c220g2	E5-2630 v3	22 nm	Q3'14	2 x 16	2 x 85	10GbE	128GB	480 GB	118.4	$599 [21]
sm220u	Xeon Silver 4314	10 nm	Q2'21	2 x 32	2 x 135	40GbE	256GB	960 GB	221.9	$6080 [5]

Table 1. Different hardware explored.

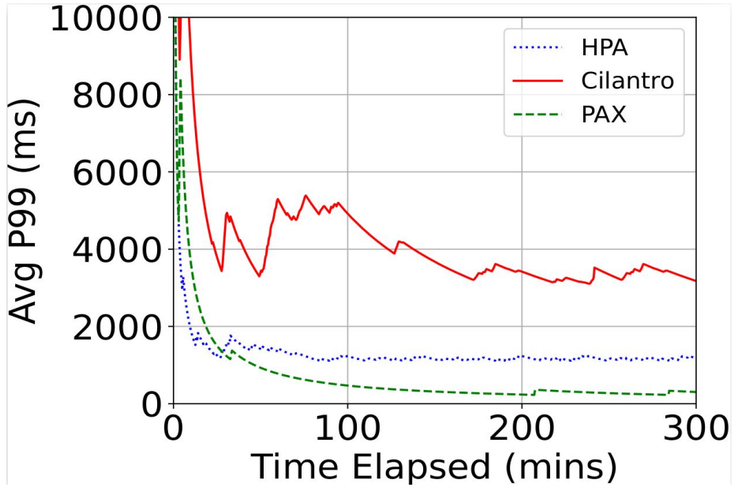
PAX, HPA, Cilantro Comparison

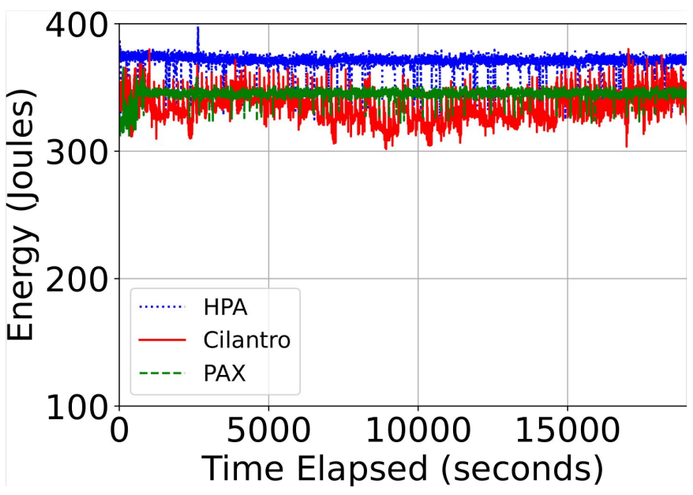
Cluster A (*older*)



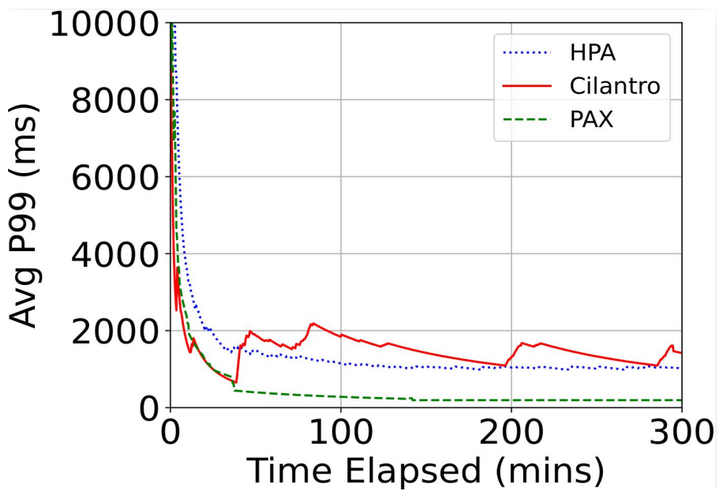


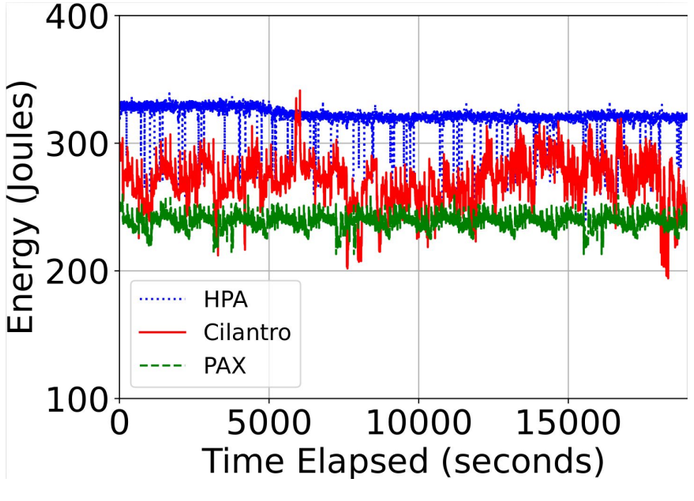
Cluster B (*newer*)





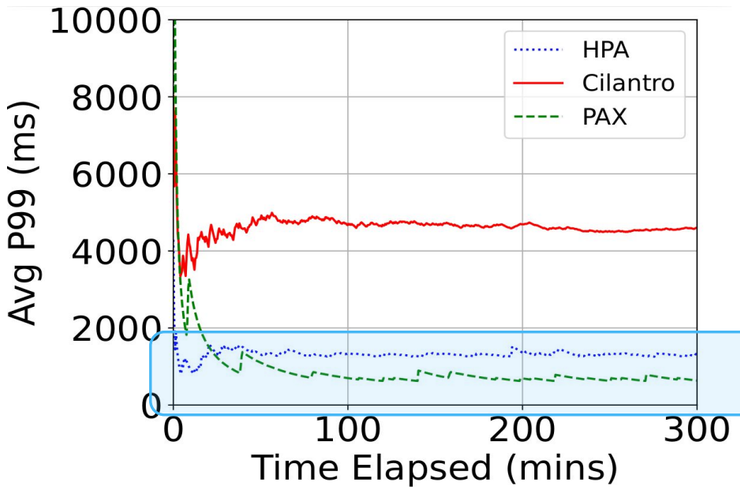
Cluster C (*hybrid*)



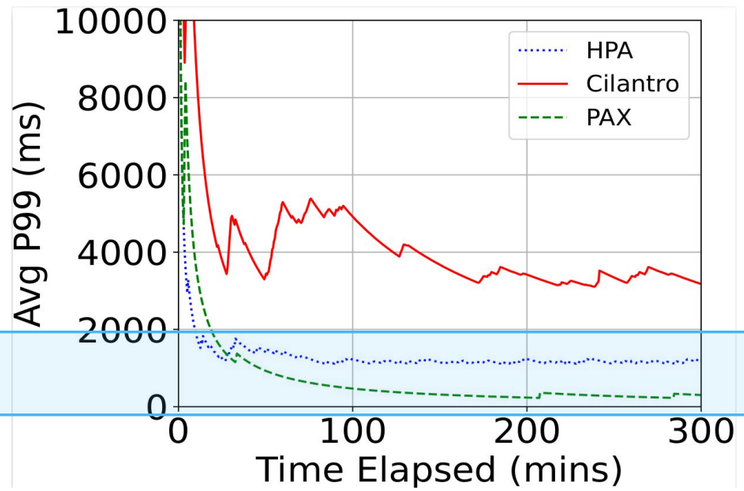


PAX, HPA, Cilantro Comparison

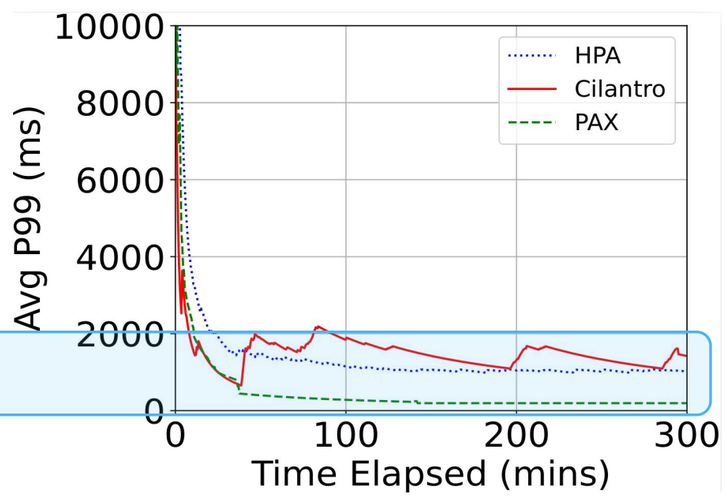
Cluster A (*older*)

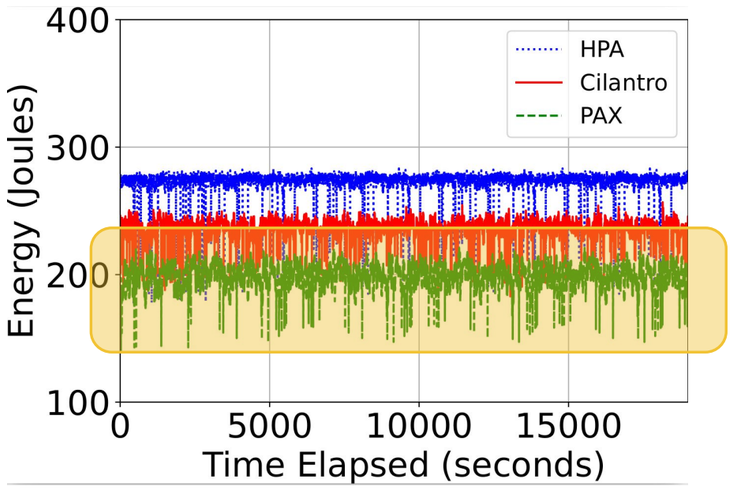


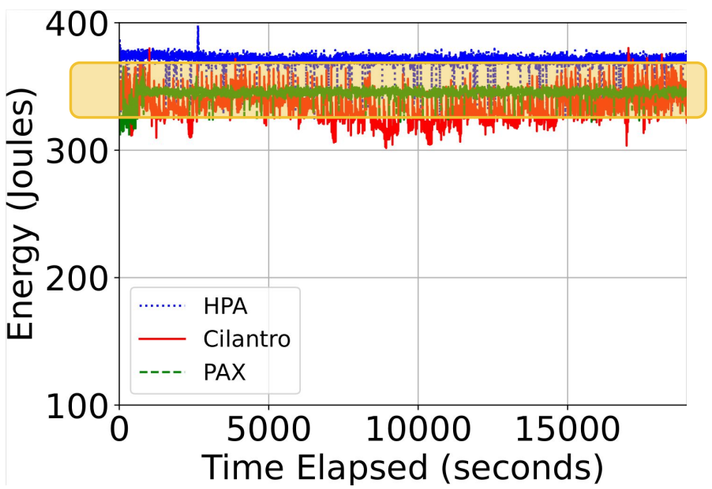
Cluster B (*newer*)

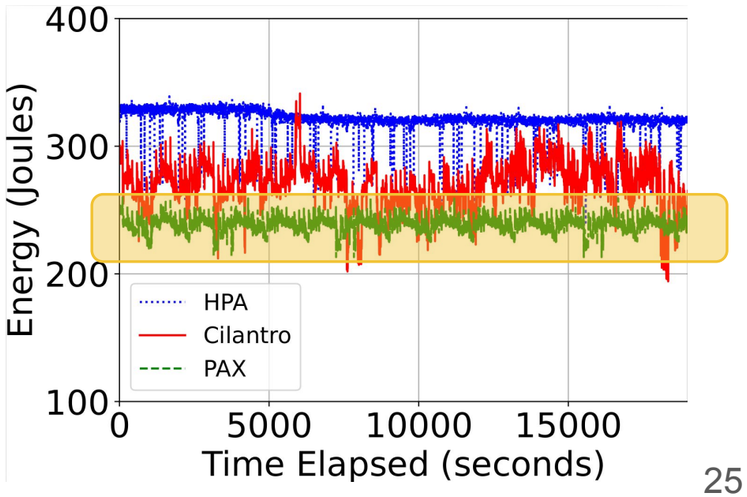


Cluster C (*hybrid*)



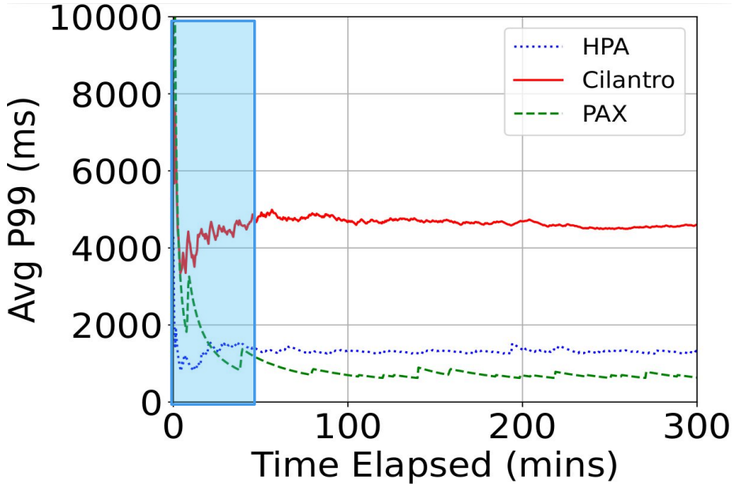


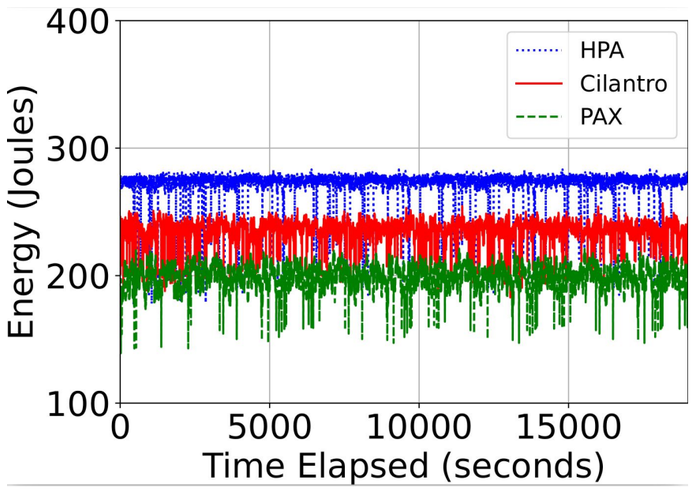




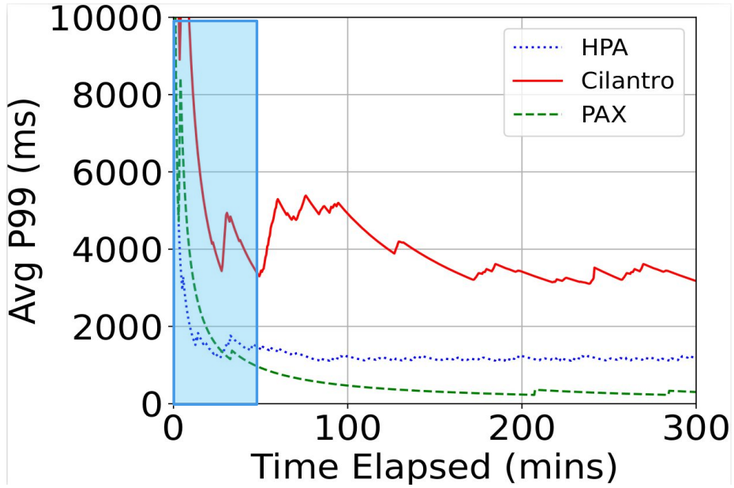
PAX, HPA, Cilantro Comparison

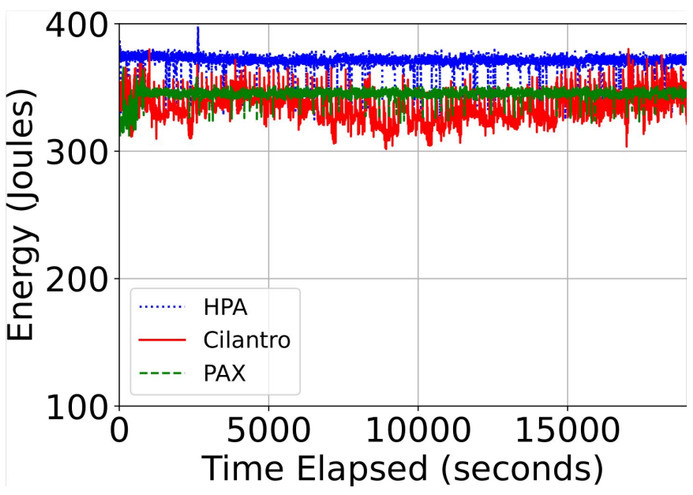
Cluster A (*older*)



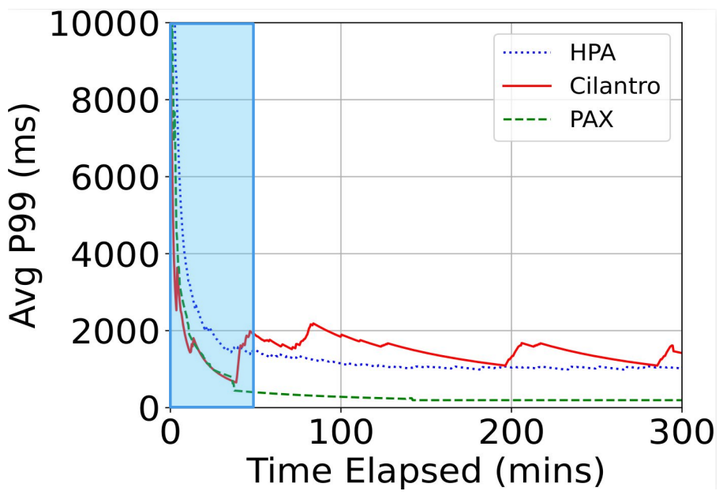


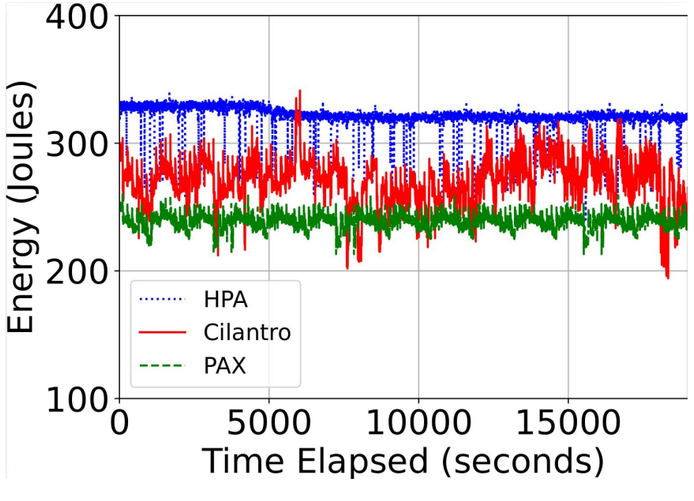
Cluster B (*newer*)





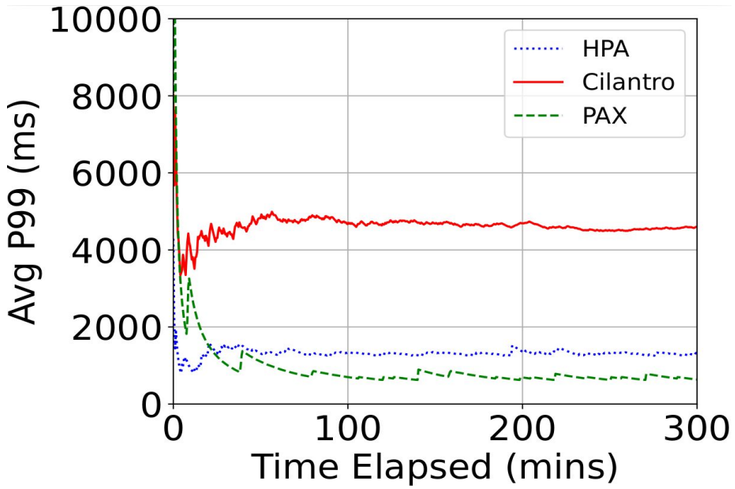
Cluster C (*hybrid*)

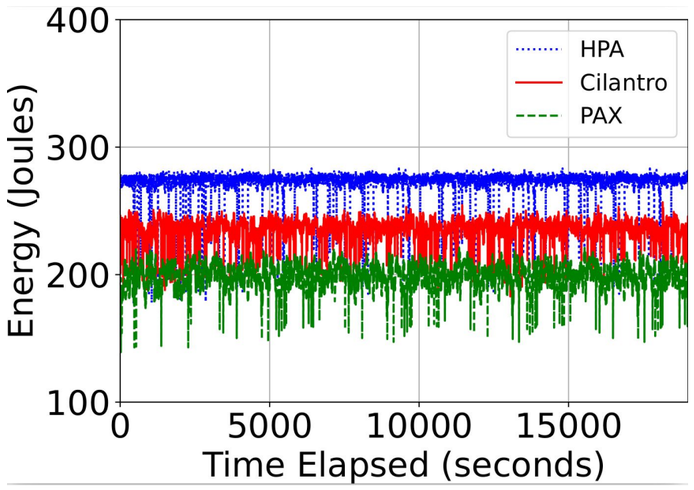




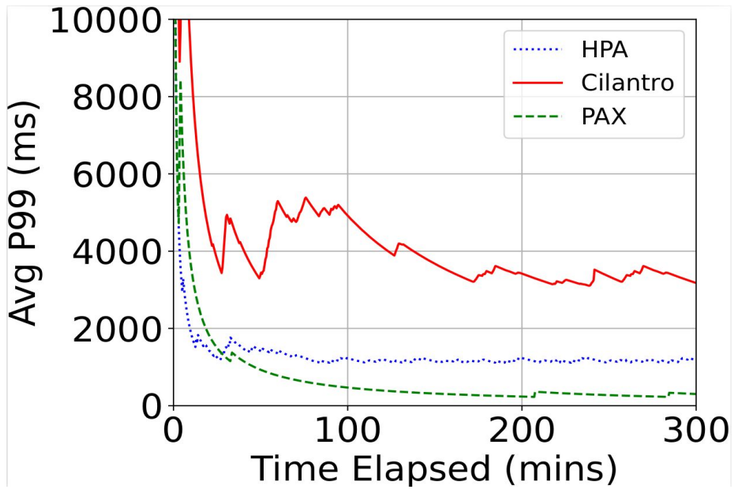
PAX, HPA, Cilantro Comparison

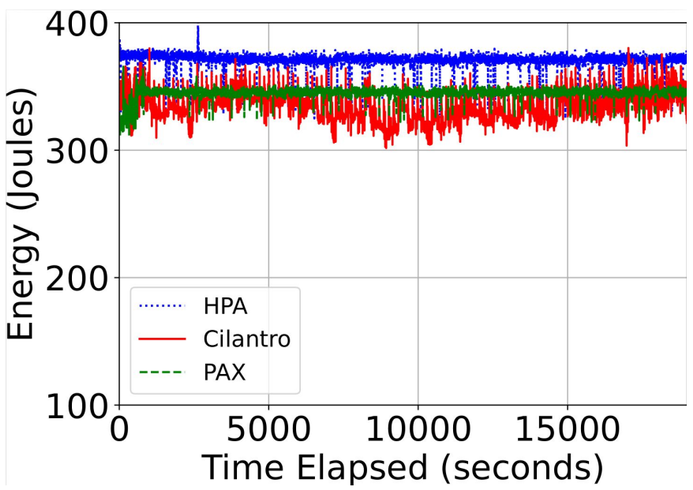
Cluster A (*older*)



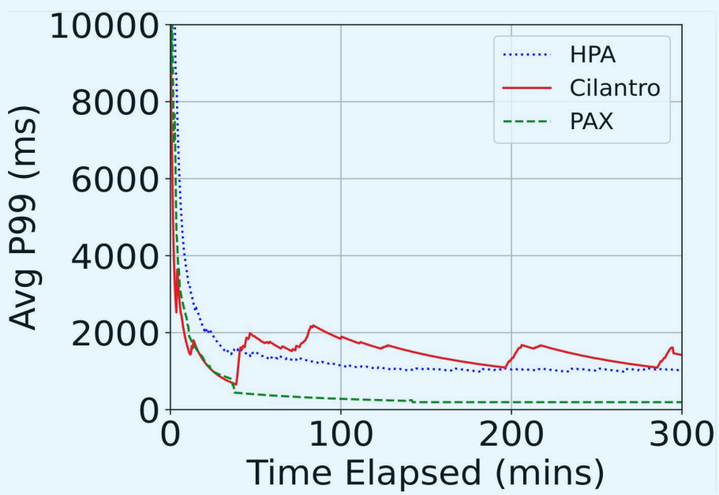


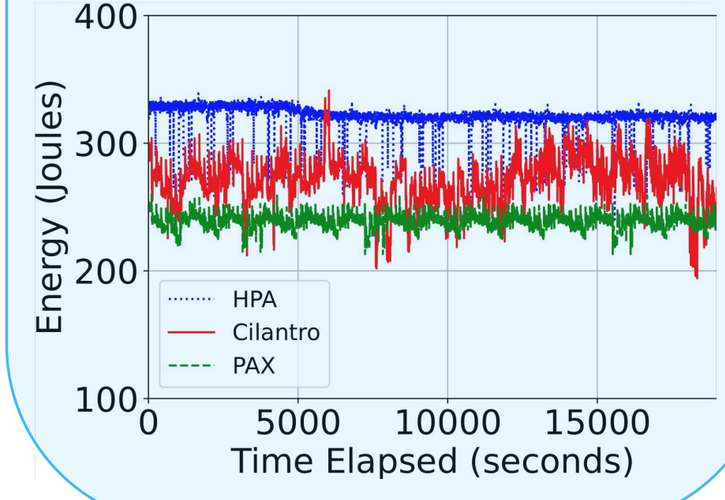
Cluster B (*newer*)





Cluster C (*hybrid*)





Examining Replica Count and Placement

	HPA		Cilantro		PAX	
Microservice	**Replicas**	**Node**	**Replicas**	**Node**	**Replicas**	**Node**
consul	1	2021-A	20	2014-A, 2014-B, 2021-A	10	2014-A
frontend	7	2014-A, 2014-B, 2021-A	10	2014-A, 2014-B, 2021-A	5	2014-A
jaeger	1	2021-A	9	2014-A, 2014-B, 2021-A	10	2014-B
search	6	2014-A, 2014-B, 2021-A	5	2014-A, 2014-B, 2021-A	1	2021-A
user	1	2021-A	5	2014-A, 2014-B, 2021-A	3	2014-B
mongodb-user	1	2014-A	3	2014-A, 2014-B, 2021-A	1	2014-B
geo	7	2014-A, 2014-B, 2021-A	7	2014-A, 2014-B, 2021-A	15	2014-B
mongodb-geo	1	2014-A	3	2014-A, 2014-B, 2021-A	1	2014-A
profile	7	2014-A, 2014-B, 2021-A	4	2014-B, 2021-A	1	2014-A
mongodb-profile	1	2021-A	3	2014-A, 2014-B, 2021-A	1	2014-A
memcached-profile	1	2021-A	7	2014-A, 2014-B	26	2021-A
rate	6	2014-A, 2014-B, 2021-A	4	2014-A, 2014-B, 2021-A	2	2014-A
mongodb-rate	1	2014-B	3	2014-A, 2014-B, 2021-A	1	2014-B
memcached-rate	2	2014-B, 2021-A	6	2014-A, 2014-B, 2021-A	19	2014-B
recommendation	6	2014-A, 2014-B, 2021-A	8	2014-A, 2014-B, 2021-A	12	2014-A
mongodb-recommendation	1	2014-A	3	2014-A, 2014-B, 2021-A	1	2014-B
reserve	6	2014-A, 2014-B, 2021-A	3	2014-A, 2014-B, 2021-A	8	2021-A
mongodb-reserve	1	2021-A	3	2014-A, 2014-B, 2021-A	1	2014-A
memcached-reserve	2	2014-B, 2021-A	4	2014-A, 2014-B	1	2014-A

Table 4. Pod replicas and their node placement in the 2X-Server-2014, 1X-Server-20221 cluster. 2014-A and 2014-B refer to distinct 2014 servers and 2021-A is the 2021 server.

Open Questions, Limitations, and Future Work

- Stabilizing configuration versus reconfiguring in response to an event
  - Rate of reconfiguration
- Colocating versus distributing replicas across nodes
- Configuring a larger node/CPU space
- Evaluating other benchmarks from DeathStarBench
- Running PAX dynamically in response to a changing world