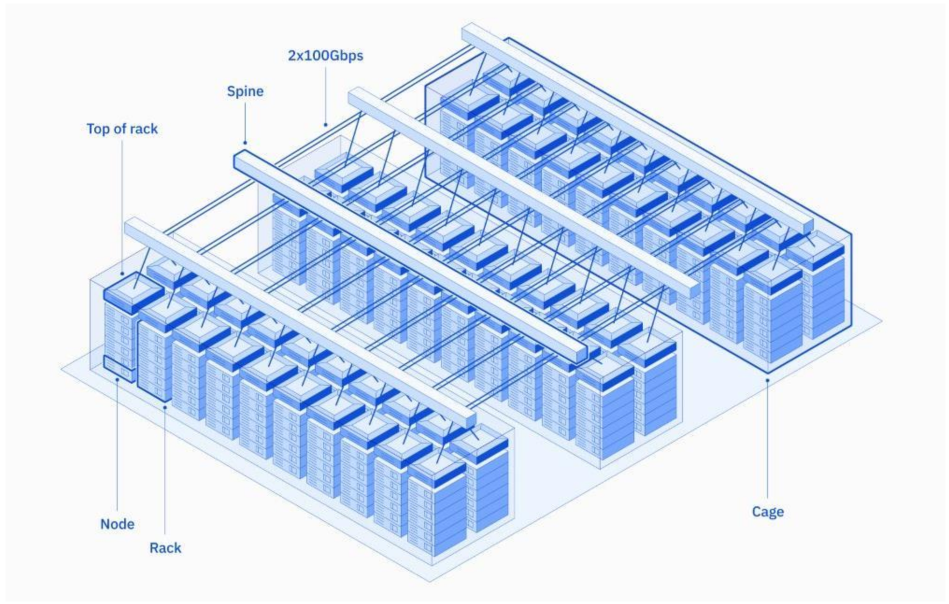
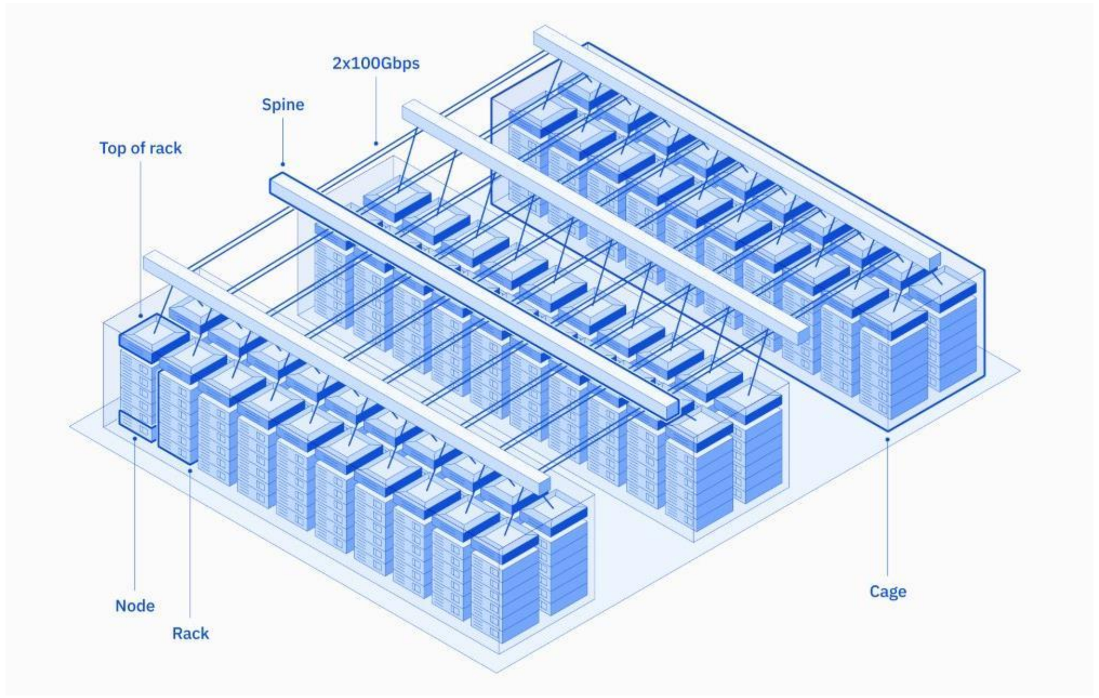


Sakkara

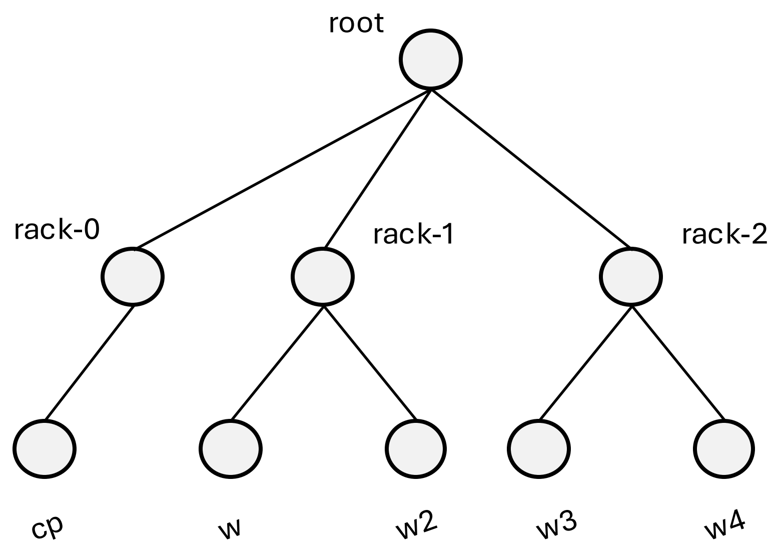
A hierarchical cluster topology group scheduler

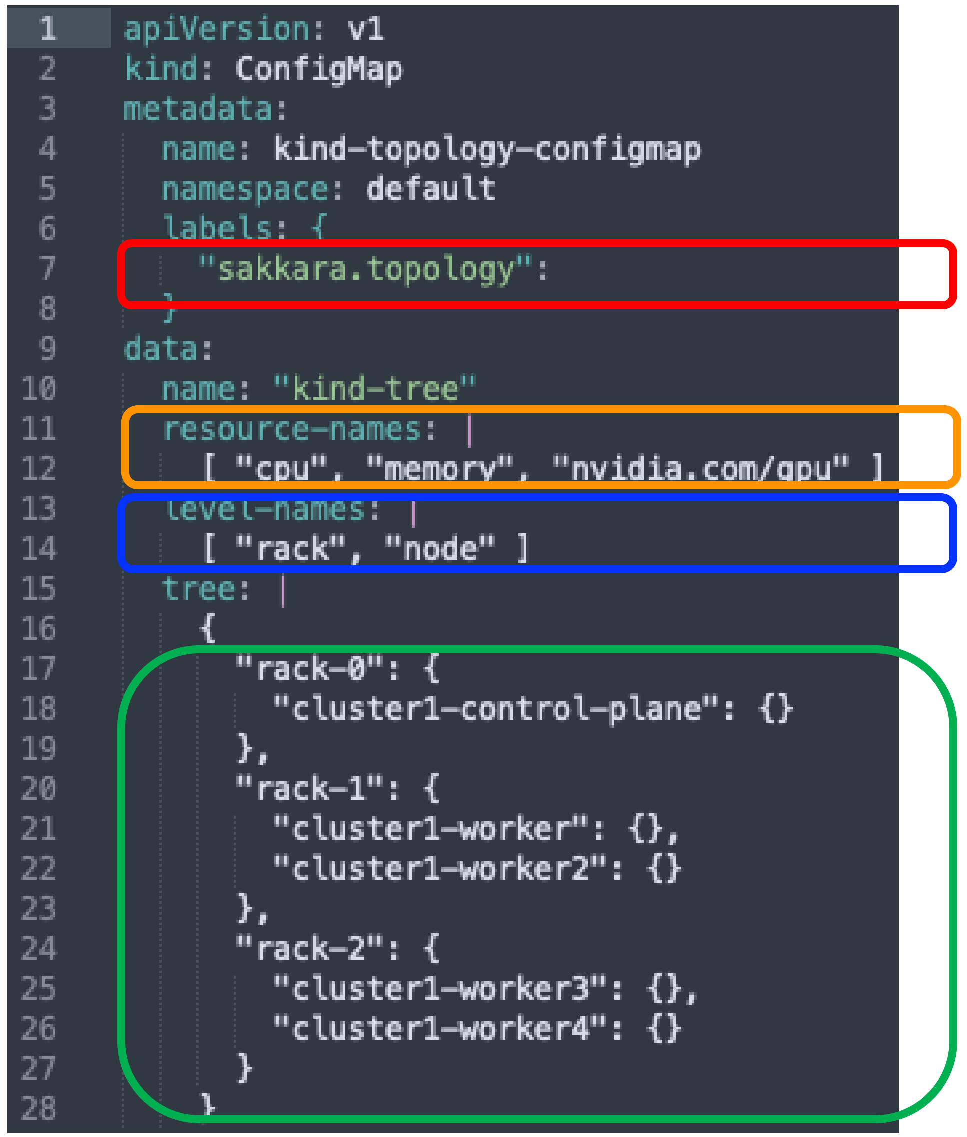
Topology-Aware Workload Placement

- Placing a workload impacts its performance
- Sakkara
  - solves the placement problem for all pods at once accounting for
    - resource requirements
    - topology constraints
  - place each pod accordingly



topology specification





```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kind-topology-configmap
  namespace: default
  labels: {
    "sakkara.topology":
  }
data:
  name: "kind-tree"
  resource-names: |
    [ "cpu", "memory", "nvidia.com/gpu" ]
  level-names: |
    [ "rack", "node" ]
  tree: |
    {
      "rack-0": {
        "cluster1-control-plane": {}
      },
      "rack-1": {
        "cluster1-worker": {},
        "cluster1-worker2": {}
      },
      "rack-2": {
        "cluster1-worker3": {},
        "cluster1-worker4": {}
      }
    }
```

label used for filtering

ordered resource names

level names top down

tree hierarchical topology

Sakkara: Highlights

- Scheduler plugin
    - Hierarchical cluster topology
        - configmap -> tree specs
    - Group (gang) scheduling
        - configmap -> group specs, placement results
        - or PodGroup
    - Logical application topology generation (pods ranking)
    - Uses chic-sched as core group placer (solver)
- Supports
    - Dynamic tree cluster topology
    - Multiple resources
    - Static, homogeneous groups
    - Group placement constraints
        - multi-level, multi-type (spread, pack, partition, range, factor), hard/soft
    - Group preemption (all or none)
    - Weighted nodes

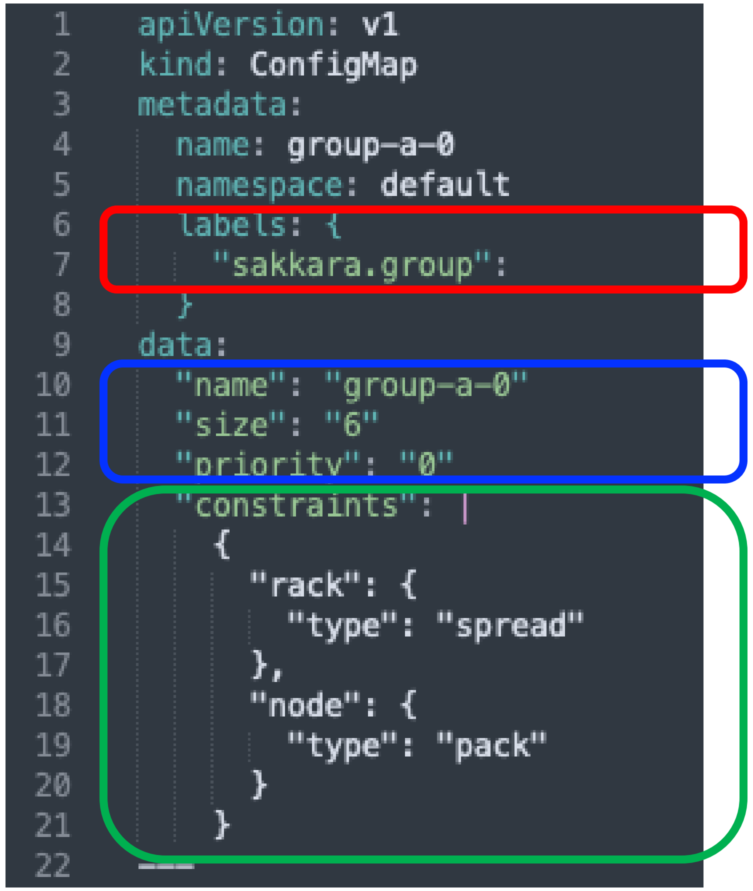
group specification

job specification

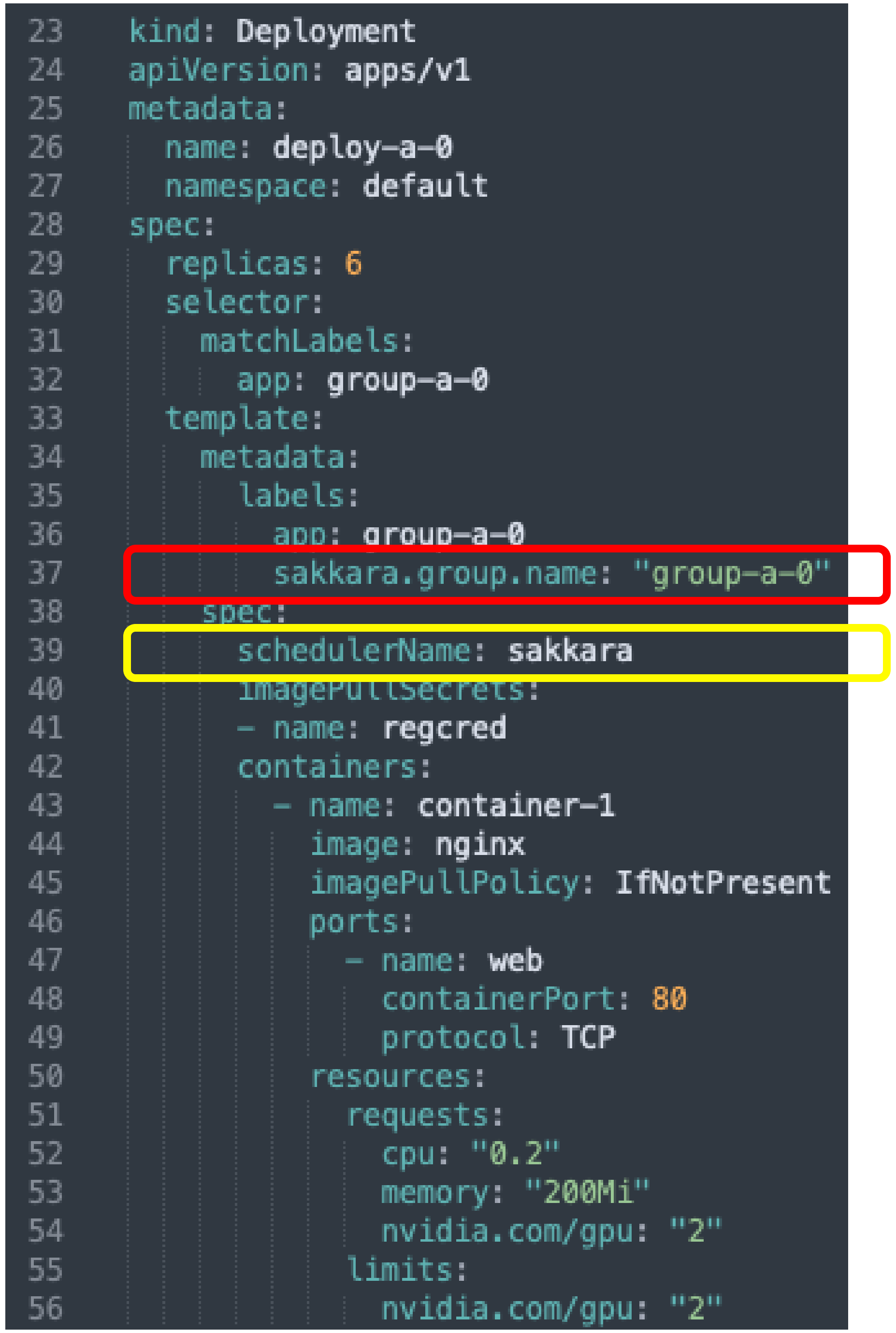
label used for filtering

group name, size, priority

group level constraints

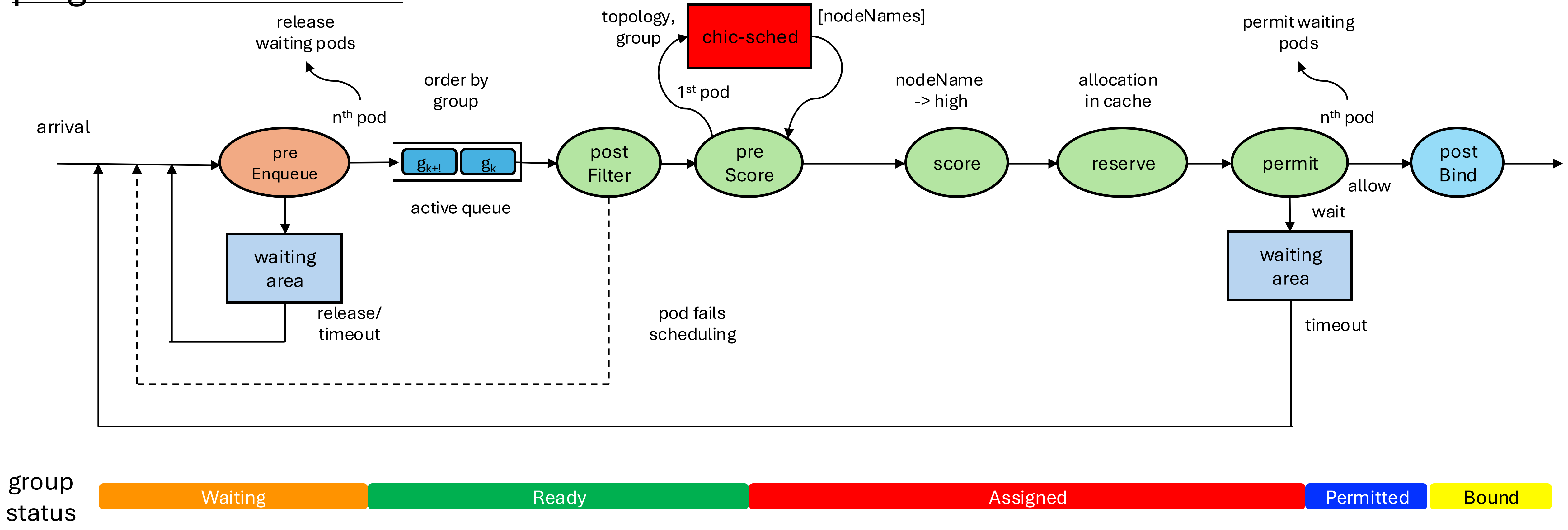


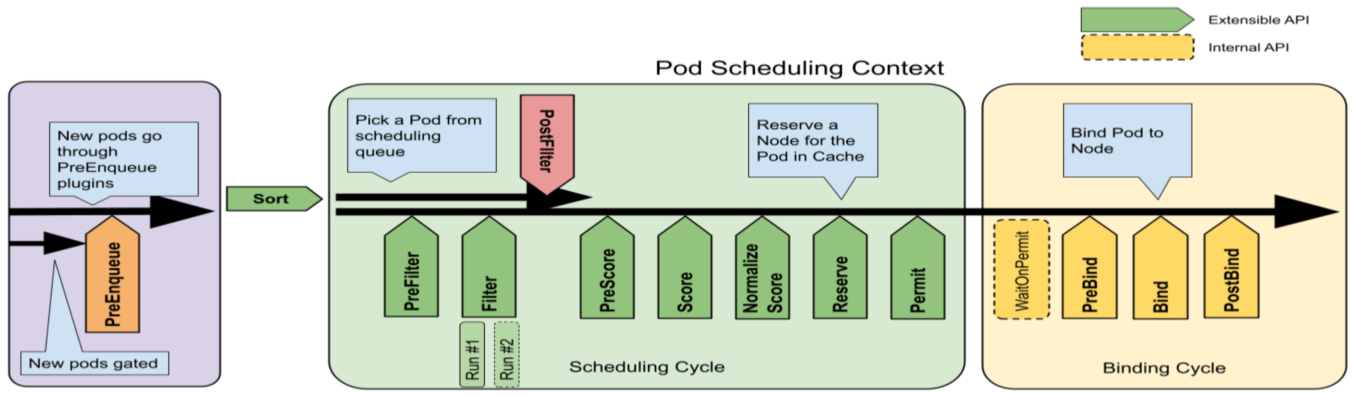
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: group-a-0
  namespace: default
  labels: {
    "sakkara.group":
  }
data:
  "name": "group-a-0"
  "size": "6"
  "priority": "0"
  "constraints": |
    {
      "rack": {
        "type": "spread"
      },
      "node": {
        "type": "pack"
      }
    }
---
```



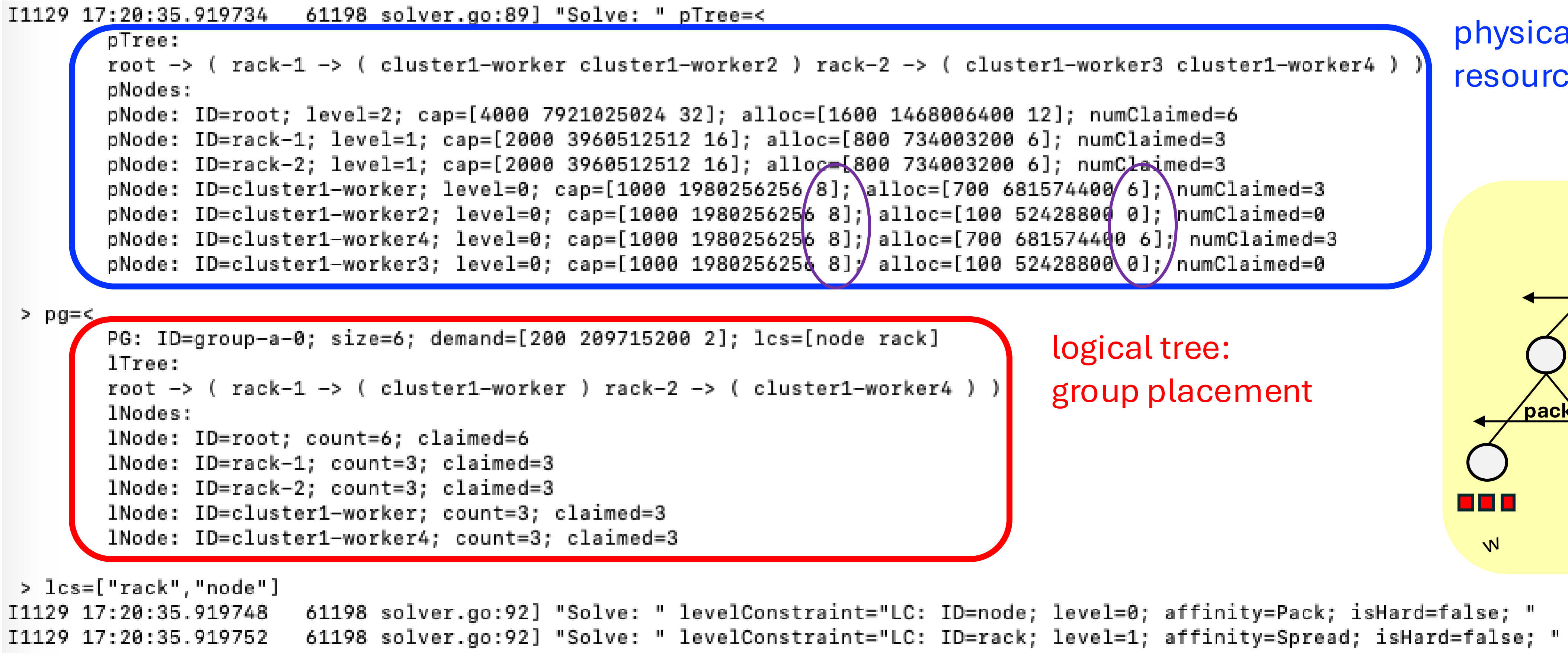
```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: deploy-a-0
  namespace: default
spec:
  replicas: 6
  selector:
    matchLabels:
      app: group-a-0
  template:
    metadata:
      labels:
        app: group-a-0
        sakkara.group.name: "group-a-0"
    spec:
      schedulerName: sakkara
      imagePullSecrets:
      - name: regcred
      containers:
        - name: container-1
          image: nginx
          imagePullPolicy: IfNotPresent
          ports:
            - name: web
              containerPort: 80
              protocol: TCP
          resources:
            requests:
              cpu: "0.2"
              memory: "200Mi"
              nvidia.com/gpu: "2"
            limits:
              nvidia.com/gpu: "2"
```

plugin state transitions





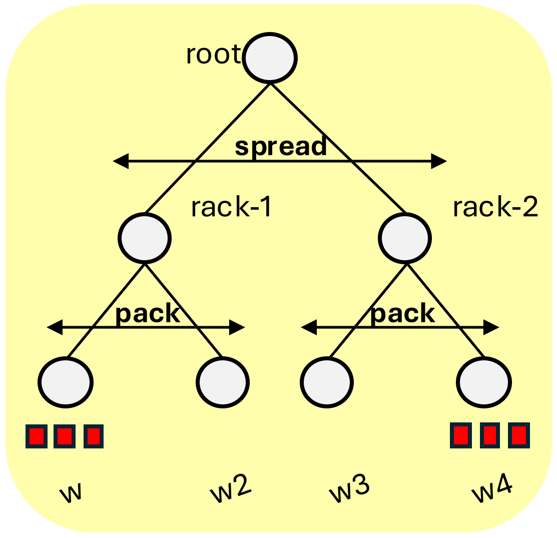
job scheduling results

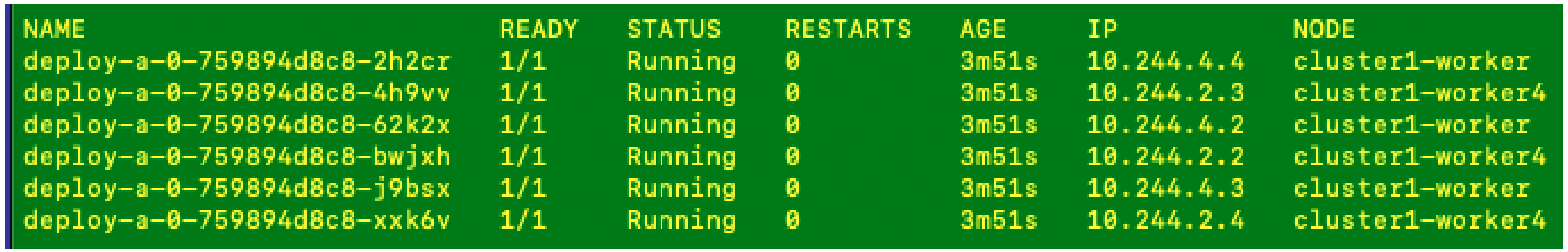


```
I1129 17:20:35.919734   61198 solver.go:89] "Solve: " pTree=<
        pTree:
        root -> ( rack-1 -> ( cluster1-worker cluster1-worker2 ) rack-2 -> ( cluster1-worker3 cluster1-worker4 ) )
        pNodes:
        pNode: ID=root; level=2; cap=[4000 7921025024 32]; alloc=[1600 1468006400 12]; numClaimed=6
        pNode: ID=rack-1; level=1; cap=[2000 3960512512 16]; alloc=[800 734003200 6]; numClaimed=3
        pNode: ID=rack-2; level=1; cap=[2000 3960512512 16]; alloc=[800 734003200 6]; numClaimed=3
        pNode: ID=cluster1-worker; level=0; cap=[1000 1980256256 8]; alloc=[700 681574400 6]; numClaimed=3
        pNode: ID=cluster1-worker2; level=0; cap=[1000 1980256256 8]; alloc=[100 52428800 0]; numClaimed=0
        pNode: ID=cluster1-worker4; level=0; cap=[1000 1980256256 8]; alloc=[700 681574400 6]; numClaimed=3
        pNode: ID=cluster1-worker3; level=0; cap=[1000 1980256256 8]; alloc=[100 52428800 0]; numClaimed=0
 > pg=<
        PG: ID=group-a-0; size=6; demand=[200 209715200 2]; lcs=[node rack]
        lTree:
        root -> ( rack-1 -> ( cluster1-worker ) rack-2 -> ( cluster1-worker4 ) )
        lNodes:
        lNode: ID=root; count=6; claimed=6
        lNode: ID=rack-1; count=3; claimed=3
        lNode: ID=rack-2; count=3; claimed=3
        lNode: ID=cluster1-worker; count=3; claimed=3
        lNode: ID=cluster1-worker4; count=3; claimed=3
 > lcs=["rack","node"]
I1129 17:20:35.919748   61198 solver.go:92] "Solve: " levelConstraint="LC: ID=node; level=0; affinity=Pack; isHard=false; "
I1129 17:20:35.919752   61198 solver.go:92] "Solve: " levelConstraint="LC: ID=rack; level=1; affinity=Spread; isHard=false; "
```

physical tree:
resource allocation

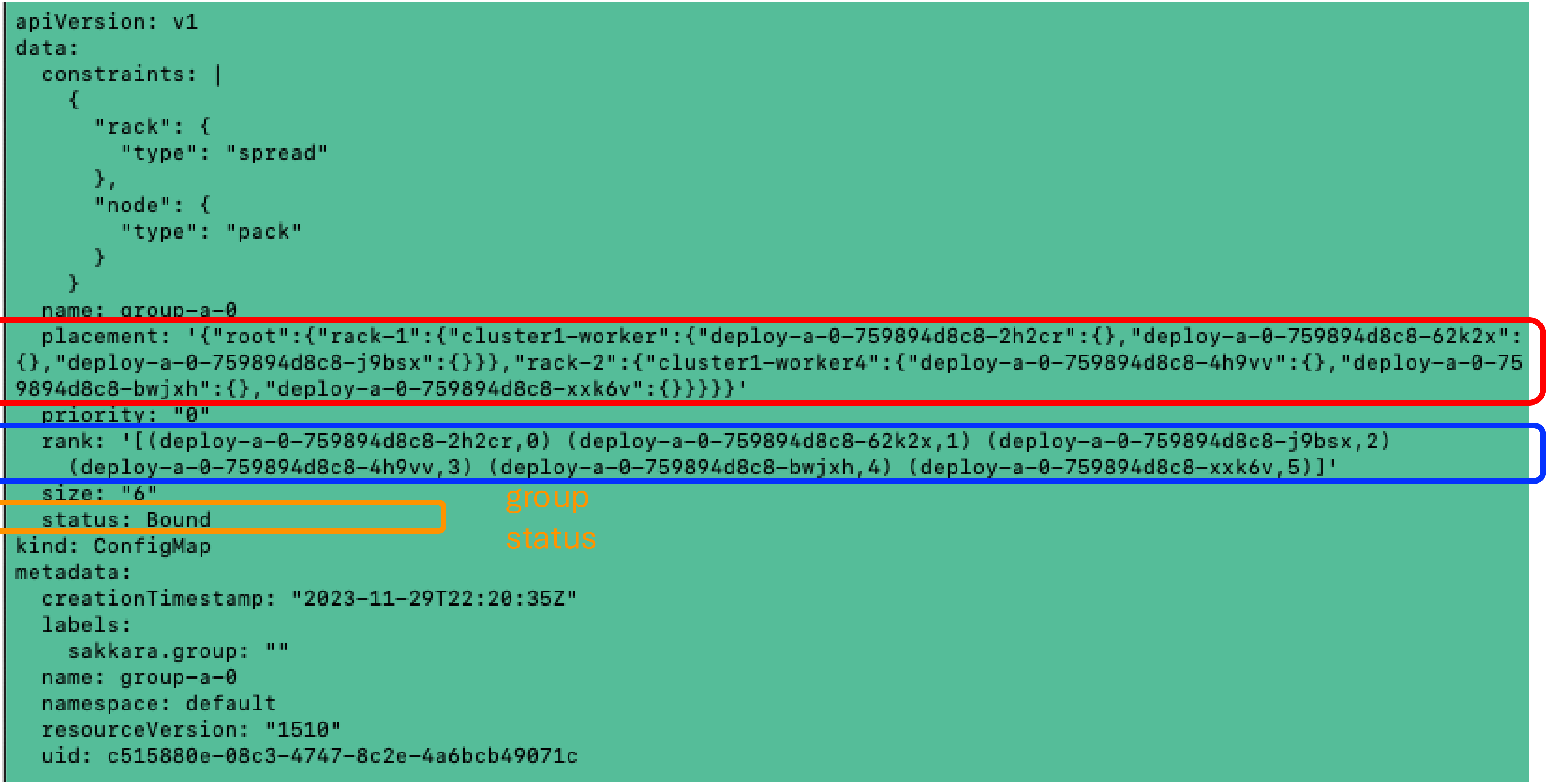
logical tree:
group placement





```
NAME                           READY   STATUS    RESTARTS   AGE     IP            NODE
deploy-a-0-759894d8c8-2h2cr    1/1     Running   0          3m51s   10.244.4.4    cluster1-worker
deploy-a-0-759894d8c8-4h9vv    1/1     Running   0          3m51s   10.244.2.3    cluster1-worker4
deploy-a-0-759894d8c8-62k2x    1/1     Running   0          3m51s   10.244.4.2    cluster1-worker
deploy-a-0-759894d8c8-bwjxh    1/1     Running   0          3m51s   10.244.2.2    cluster1-worker4
deploy-a-0-759894d8c8-j9bsx    1/1     Running   0          3m51s   10.244.4.3    cluster1-worker
deploy-a-0-759894d8c8-xxk6v    1/1     Running   0          3m51s   10.244.2.4    cluster1-worker4
```

group configmap



```
apiVersion: v1
data:
  constraints: |
    {
      "rack": {
        "type": "spread"
      },
      "node": {
        "type": "pack"
      }
    }
  name: group-a-0
  placement: '{"root":{"rack-1":{"cluster1-worker":{"deploy-a-0-759894d8c8-2h2cr":{},"deploy-a-0-759894d8c8-62k2x":
{},"deploy-a-0-759894d8c8-j9bsx":{}}},"rack-2":{"cluster1-worker4":{"deploy-a-0-759894d8c8-4h9vv":{},"deploy-a-0-75
9894d8c8-bwjxh":{},"deploy-a-0-759894d8c8-xxk6v":{}}}}}'
  priority: "0"
  rank: '[(deploy-a-0-759894d8c8-2h2cr,0) (deploy-a-0-759894d8c8-62k2x,1) (deploy-a-0-759894d8c8-j9bsx,2)
    (deploy-a-0-759894d8c8-4h9vv,3) (deploy-a-0-759894d8c8-bwjxh,4) (deploy-a-0-759894d8c8-xxk6v,5)]'
  size: "6"
  status: Bound
kind: ConfigMap
metadata:
  creationTimestamp: "2023-11-29T22:20:35Z"
  labels:
    sakkara.group: ""
  name: group-a-0
  namespace: default
  resourceVersion: "1510"
  uid: c515880e-08c3-4747-8c2e-4a6bcb49071c
```

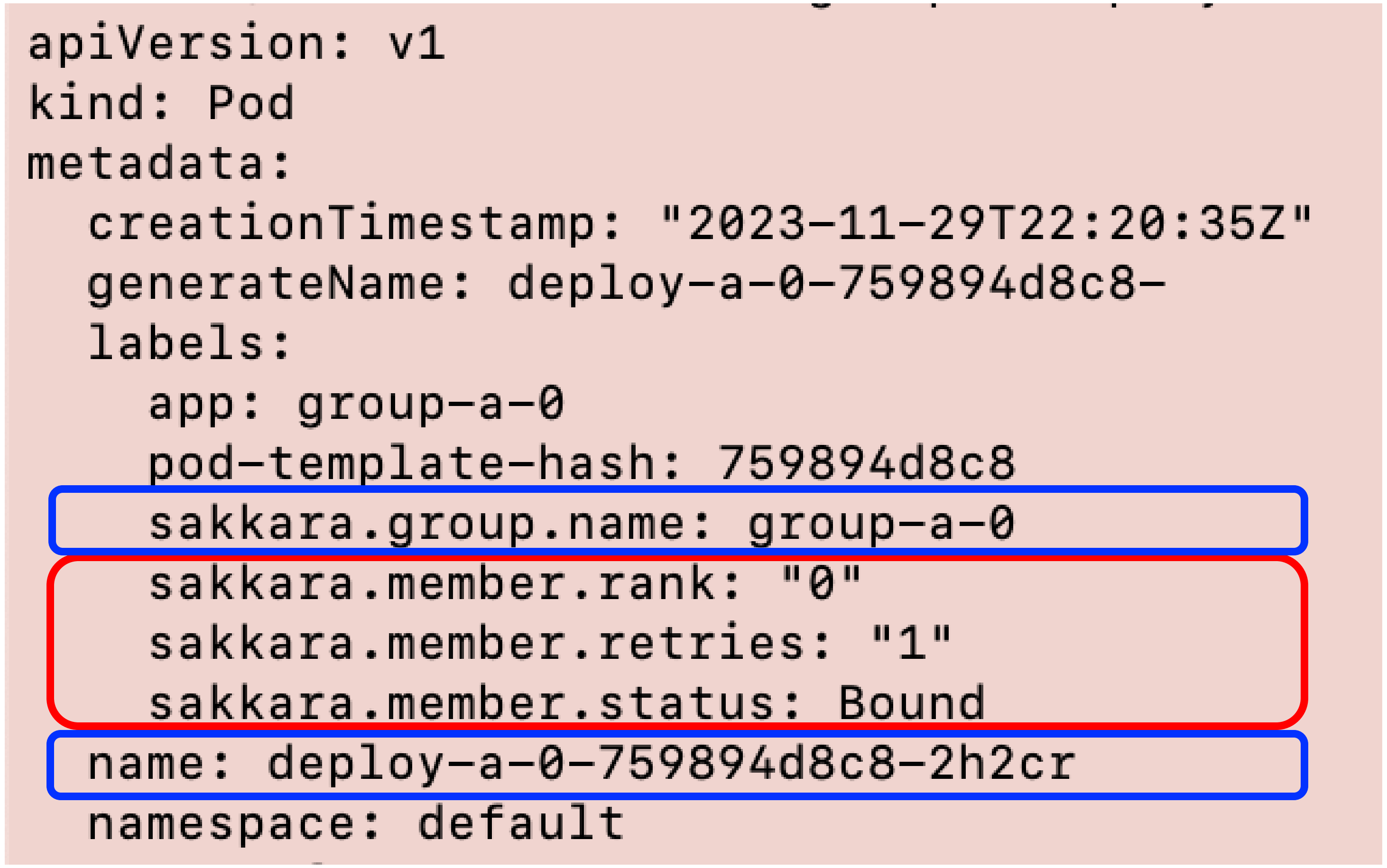
pods placement

pods ranking

group status

rank: order based on (hierarchical) distance from master pod (rank 0)

pod labels



```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2023-11-29T22:20:35Z"
  generateName: deploy-a-0-759894d8c8-
  labels:
    app: group-a-0
    pod-template-hash: 759894d8c8
    sakkara.group.name: group-a-0
    sakkara.member.rank: "0"
    sakkara.member.retries: "1"
    sakkara.member.status: Bound
  name: deploy-a-0-759894d8c8-2h2cr
  namespace: default
```

group name

sakkara-generated attributes

pod name

Sakkara: Algorithm

chic-sched: A scheduler for HPC placement groups

https://github.ibm.com/chic/chic-sched

- Heuristics-based and topology-aware group placement algorithm
  - physical and logical trees (infrastructure-concealed, application topology)
  - multiple-level constraints (pack racks, spread servers)
  - tree traversal without retrials, suboptimal, fast, depends on heuristics
  - modeling and analysis driven heuristics design
- Dynamic (elastic) placement groups
  - optimal addition/deletion
  - provisional scheduling

```
kind: GroupPlacement
spec:
  group:
    name: MyApp
    size: 20
    type: bx2-16x64
  constraints:
  - level: rack
    affinity: pack
    soft: true
```

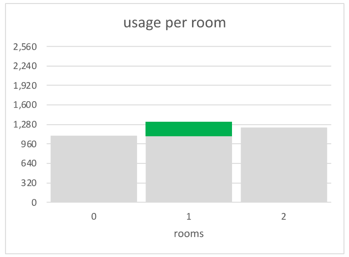
group size
- specified, fixed
- unspecified, estimated, dynamic

type
- homogeneous
- specifies resources requested

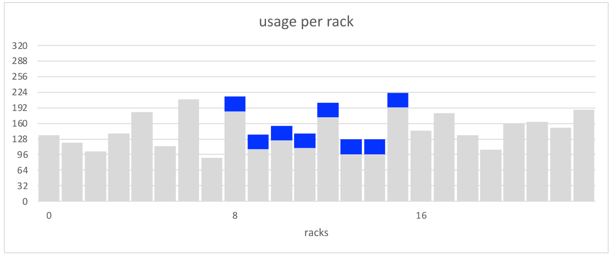
level: zone | room | rack | server
affinity: spread | pack
soft: true | false

Simulation Experiment I

Infrastructure:
1 zone
3 rooms/zone
8 racks/room
20 servers/rack
Placement Group:
120 VMs
1/8 server

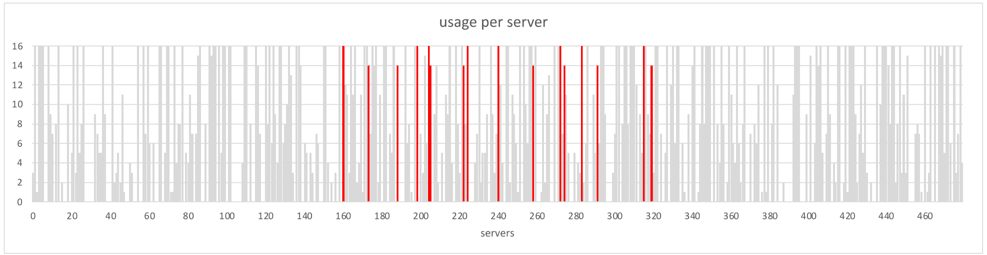


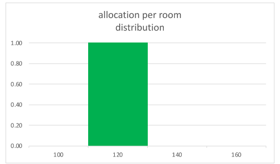
Pack **Room** (default)

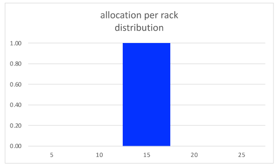


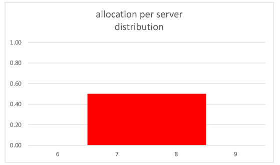
Spread **Rack** Soft

Pack **Server** Soft





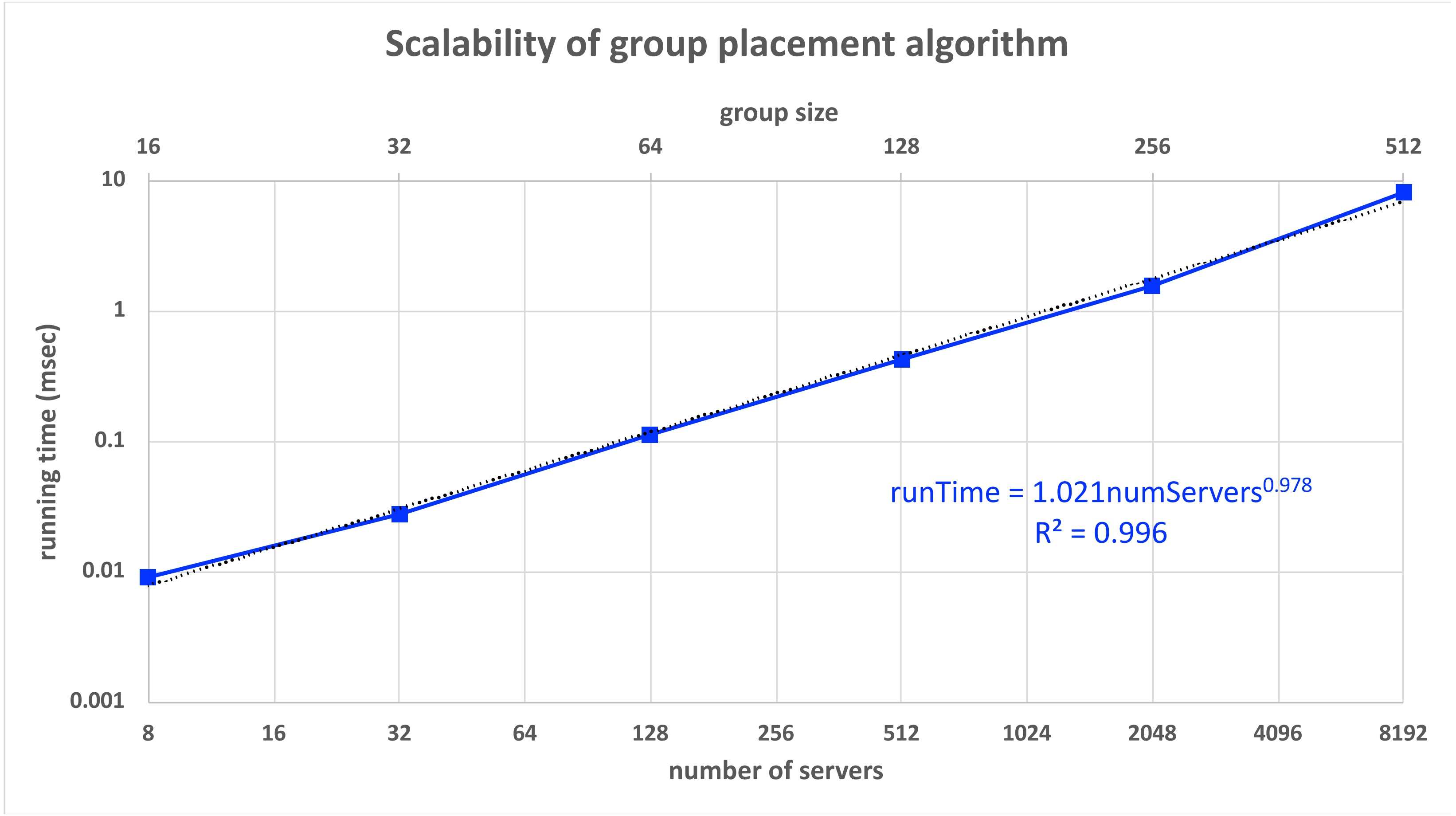




Placement algorithm

- Traverse physical tree in depth first order
- Solve placement subproblem when visiting node n
- Choice of heuristics
  - ordering of sibling nodes
  - determination of placement range
  - selection of best number to place at a node
- Return of logical tree, representing placement result
- Variations of algorithm
  - place a partially placed group
  - place dynamic group (size changes)

Performance



Configuration

- 1 zone, x racks, 2x servers/rack
- group size 8x
- resources
  - CPU
  - capacity = [16]
  - demand = [2]
- load = 40%
- skewed allocation
  - p0=1/4, p1=1/8, CoV=1.5
- constraints
  - rack, spread, soft
  - server, pack, soft
- 10,000 samples per point