AUTO SCALING

Date: 27.06.2019

AGENDA

- Intro
- Available Solutions
- Alert Based Scaling
- Auto Scaler Sokar
- Roadmap

INTRO

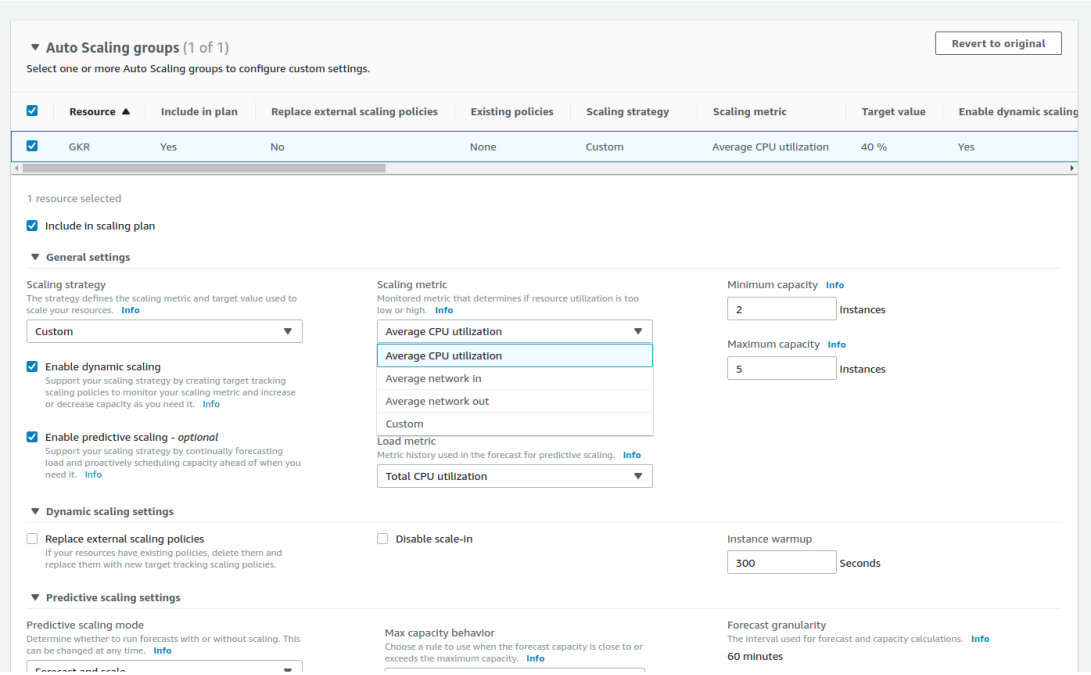
- Goal
  - Automatically adjust the number of instances to fulfill a defined SLA in a cost efficient way.
  - Dynamically scale in/out based on current load.

- Preconditions
  - Scalable Architecture + Implementation
  - Scalable resources (Platform)
  - Information about current System State

- What to be scaled
  - Storage/ Transport (DB‘s, Message Queues)
  - Networking Infrastructure (Loadbalancer, Router, Gateways, ..)
  - **Compute (EC2 Instances)**
  - **Mircoservices/ Components (Nomad Jobs)**

AVAILABLE SOLUTIONS

- AWS Dynamic Scaling (EC2 Instances/ Nomad Cluster)
  - https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html
  - AWS Autoscaling Policies
    - Predefined (availability, cost, performance) or custom Strategies
    - CPU, MEM and Custom metrics
    - Dynamic and Predictive Scaling



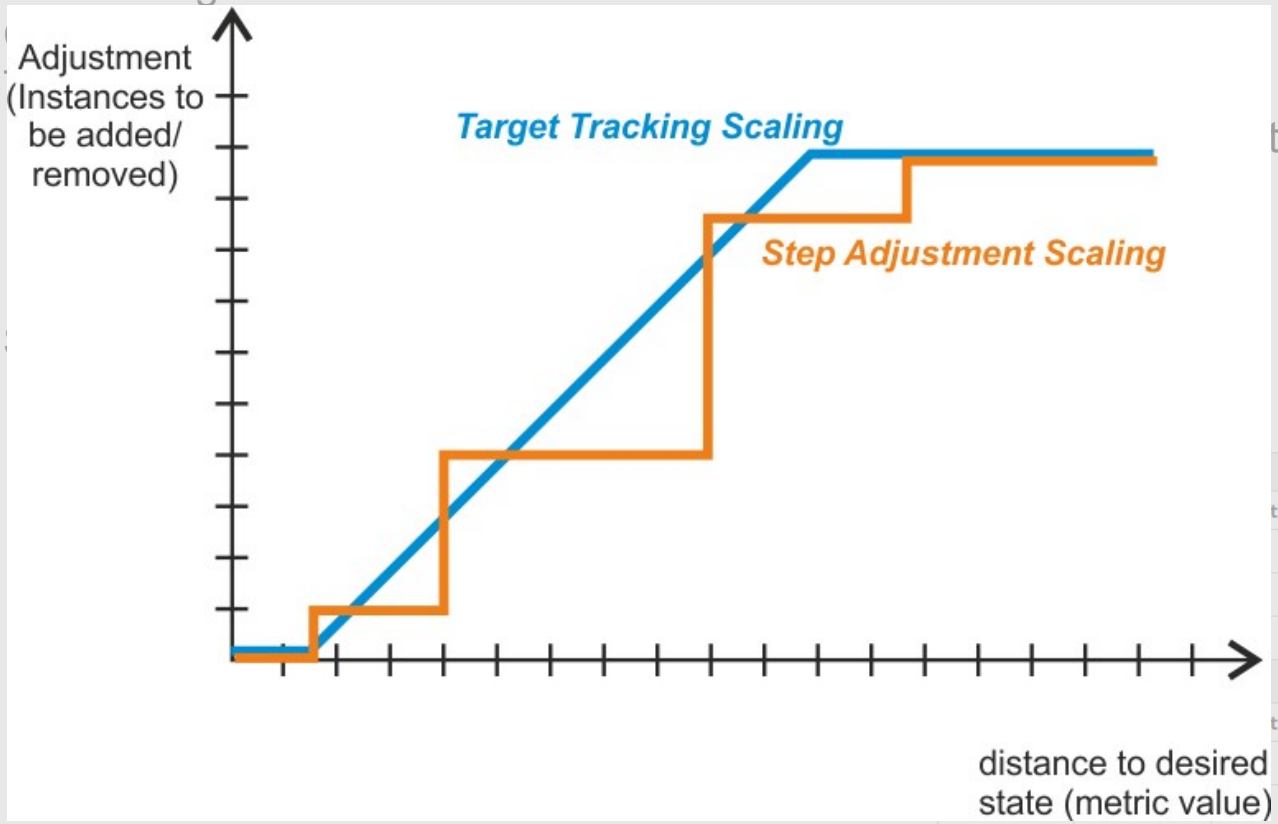
AVAILABLE SOLUTIONS

- AWS Autoscaling Policies
  - Custom metrics
  - Target Tracking Scaling Policiy
    - Scales to ensure that a defined target value for a metric is kept
    - Alarm is fired if the metric violates the defined target value
    - Calculates the needed instances based on difference between target and current metric value
  - Simple and Step Scaling Policy
    - Metric + Threshold
    - Alarm when thresholds are violated
    - Non-Linear, stepwise scaling adjustment

Scale out policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
0	10	0	50 <= *value* < 60
10	20	10	60 <= *value* < 70
20	null	30	70 <= *value* < +infinity
Scale in policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
-10	0	0	40 < *value* <= 50
-20	-10	-10	30 < *value* <= 40
null	-20	-30	-infinity < *value* <= 30

AVAILABLE SOLUTIONS

- AWS Autoscaling Policies
  -
  -
  -



t

target and

		tment	Metric value
			50 <= *value* < 60
			60 <= *value* < 70
			70 <= *value* < +infinity

		tment	Metric value
			40 < *value* <= 50
			30 < *value* <= 40
null	-20	-30	-infinity < *value* <= 30

AVAILABLE SOLUTIONS

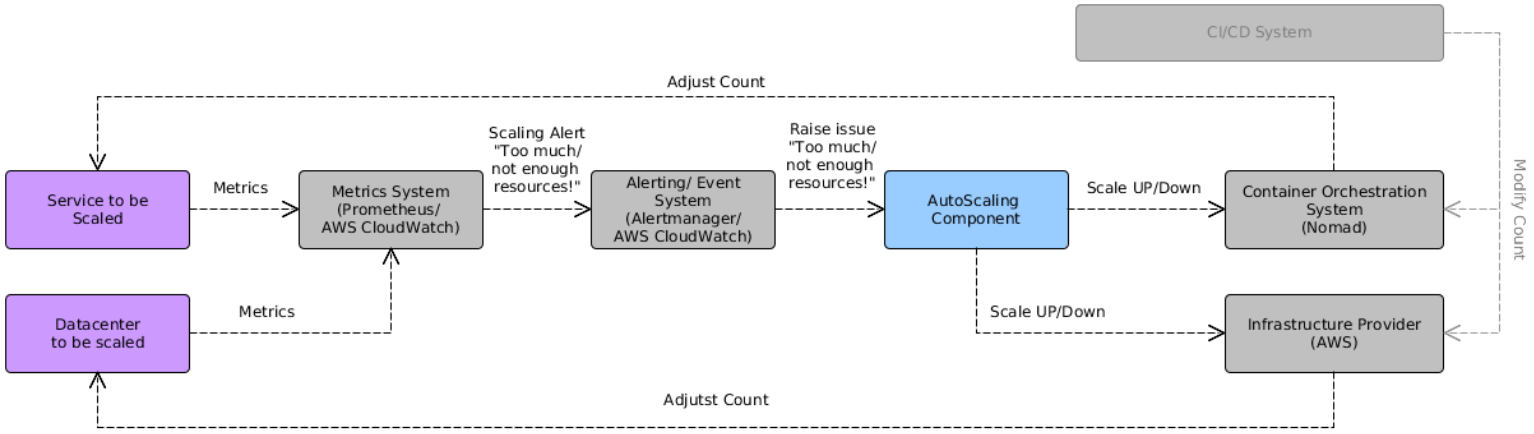
- Horizontal Pod Autoscaler (Kubernetes: job/pod)
  - https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
  - **Target Tracking Scaling**
  - Based on CPU or a custom metric
  - *desiredReplicas = ceil[currentReplicas * (currentMetricValue / desiredMetricValue)]*
  - regards pod state (initial-readiness-delay, cpu-initialization-period)
  - conservative downscaling (downscale-stabilization-window = 5m)

AVAILABLE SOLUTIONS

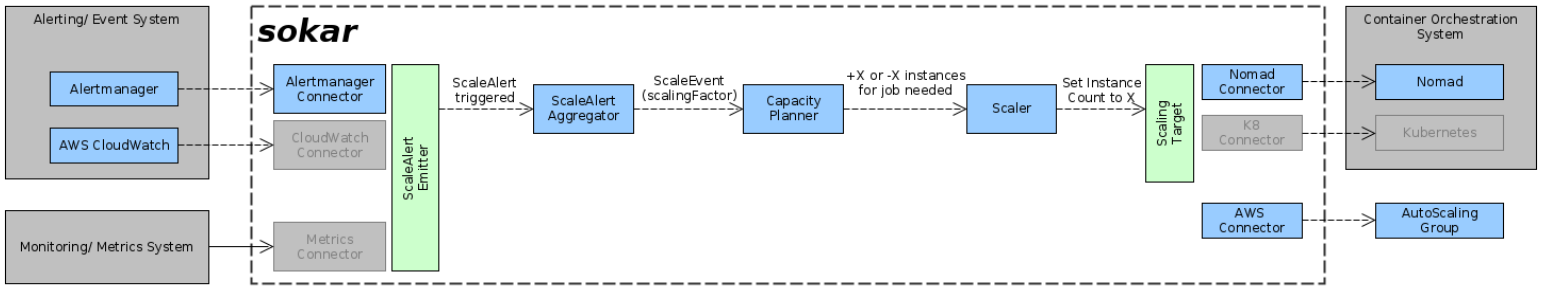
- Replicator (Nomad: cluster and job)
  - https://github.com/elsevier-core-engineering/replicator
  - Cluster:
    - Calculates the needed nodes in „worker pool“ to handle the currently running jobs
    - Considers available (capacity) and used resources (CPU, MEM and disk)
    - Reserves space for jobs running by guessing they would scale by 1
    - Supports draining and selection of least used node (while scaling down)
  - Job:
    - **Step Scaling**
    - Based on CPU and MEM
    - Separate thresholds for CPU/MEM scale out/ in
    - Scale Up/ Down by 1 if threshold is violated
    - Max/ Min job count can be specified
    - Fixed cooldown between scale actions (replicator_cooldown = 60)

ALERT BASED SCALING

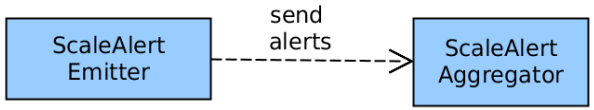
- Use Metrics for scaling (information about system state)
- Use Alerts to define the situation where scaling is needed
- Sequence
  - Service to be scaled exposes its state through metrics
  - Metrics are evaluated against scaling alert rules
    - Multiple scaling alerts per service
    - Can be defined based on different metrics
    - Separate alerts for scaling up and scaling down
  - On rule match, a scale alert is created and routed to the auto scaling component
  - Auto scaling component aggregates the available alerts and decides what to do
  - If a scale up/ down was decided, the container orchestration system (nomad, K8, …) or the Compute Instance Scaler (i.e. AWS ASG) is triggered accordingly



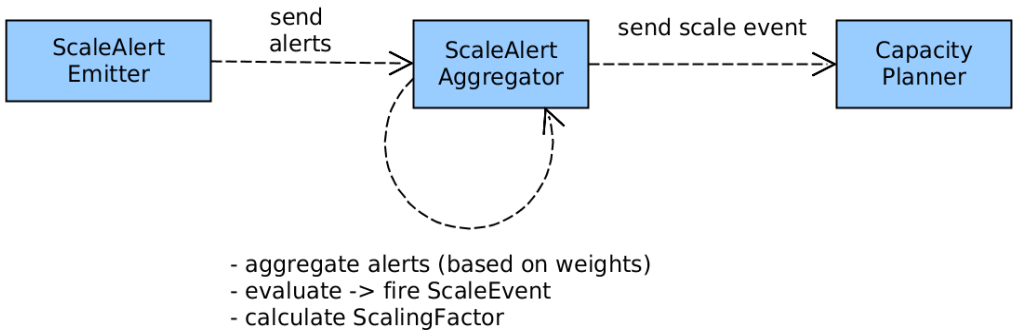
SOKAR



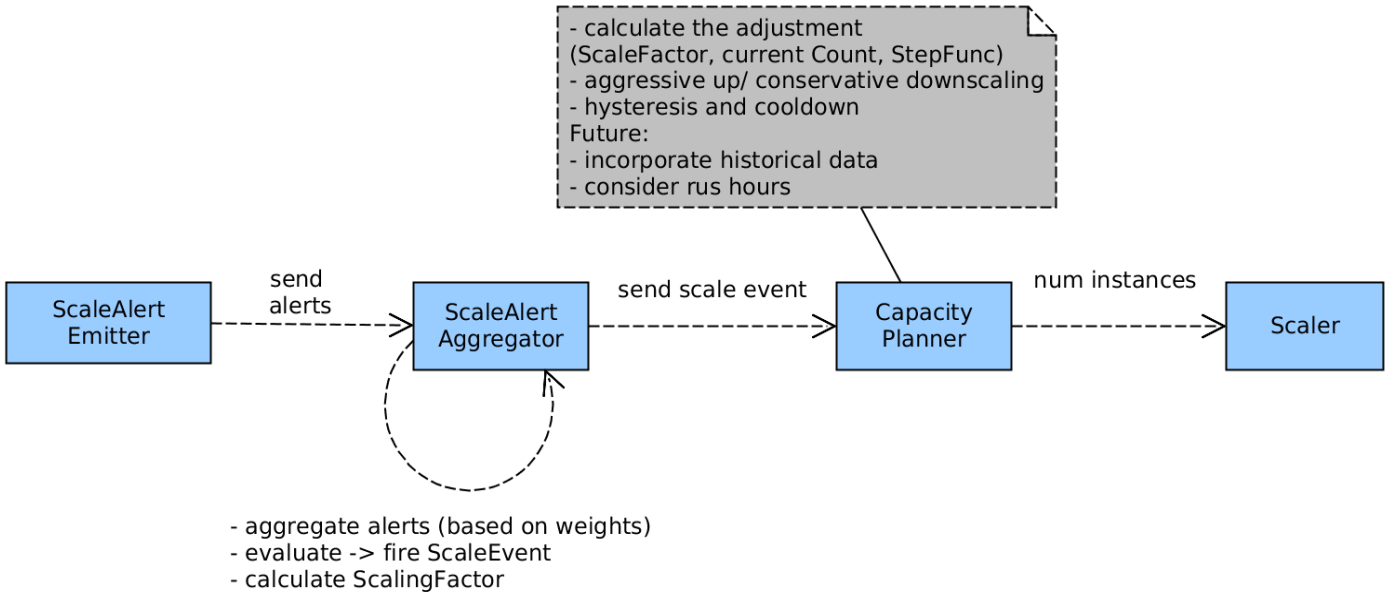
- Constant-, Linear- and Step Scaling
- Weighted Scaling Alerts
- One Auto Scaler per service



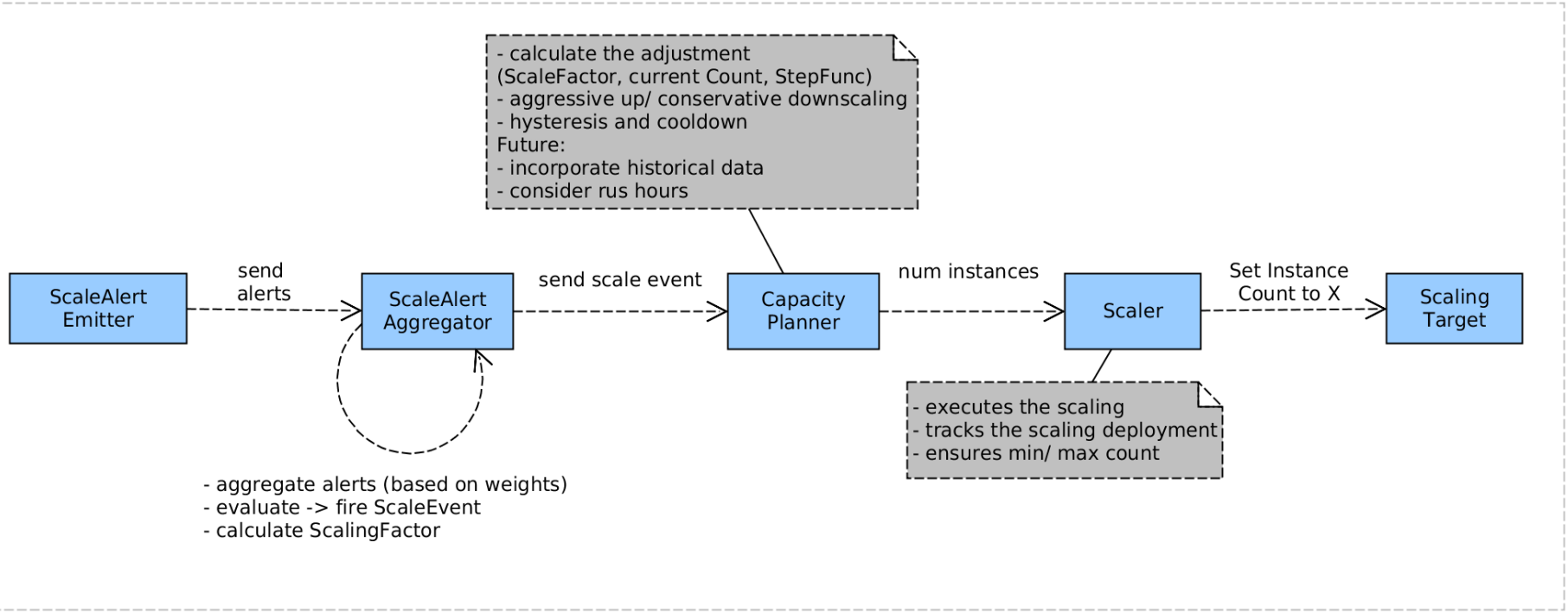
SOKAR



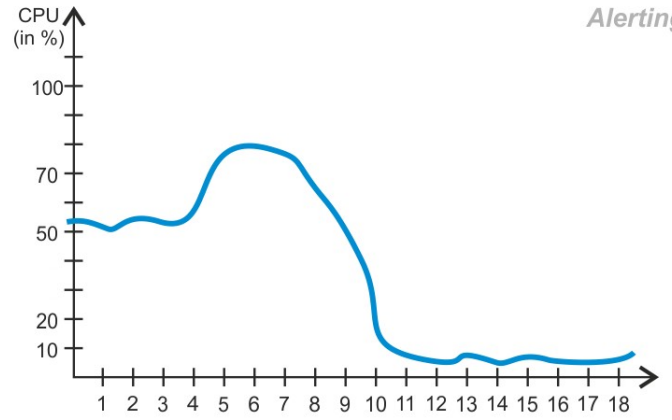
SOKAR



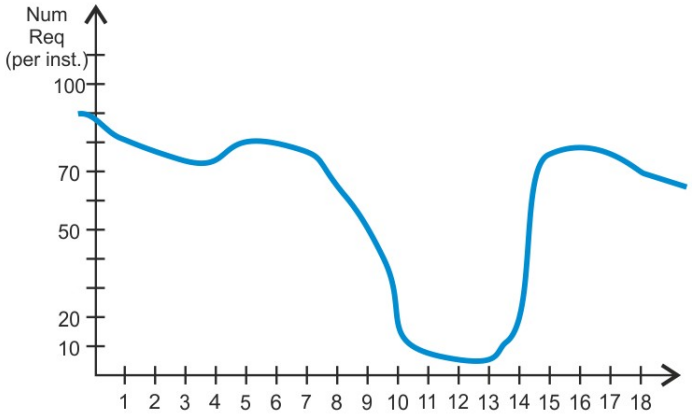
SOKAR

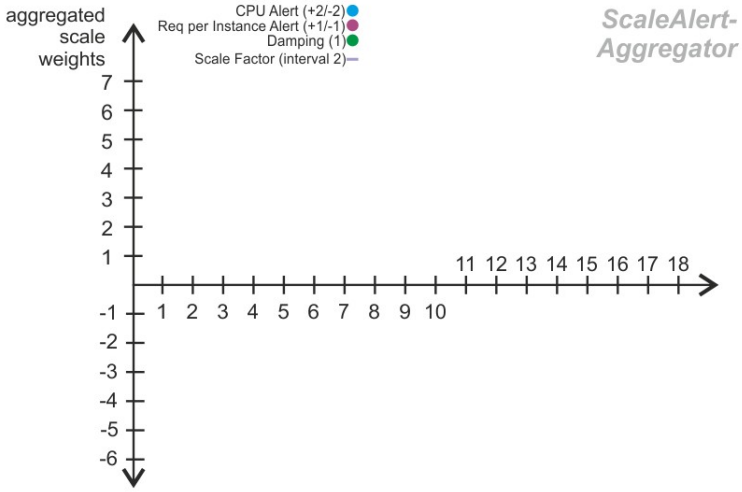


SOKAR

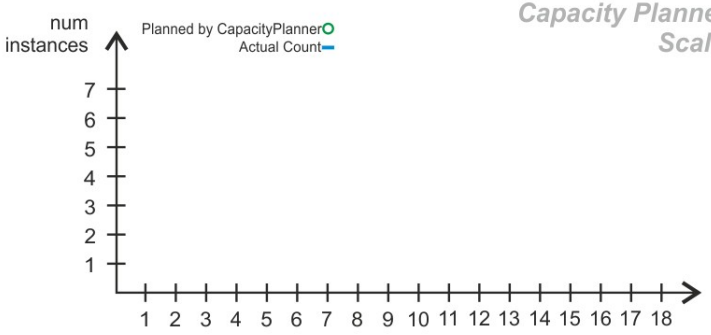


Alerting



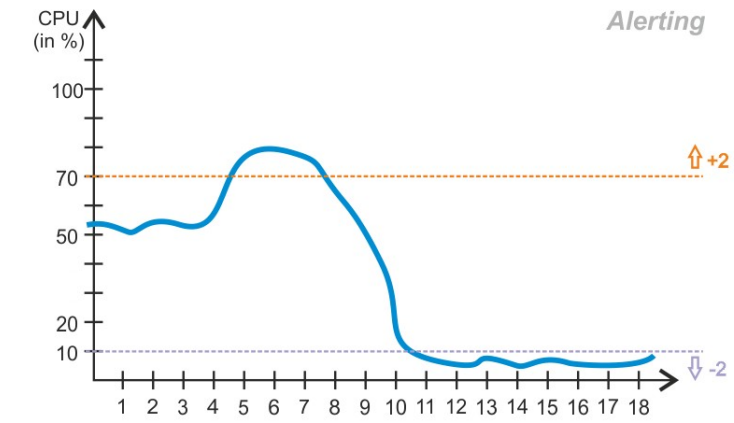


ScaleAlert-Aggregator

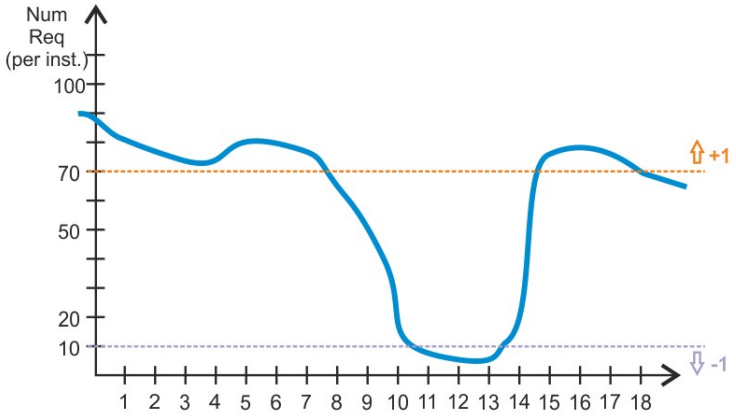


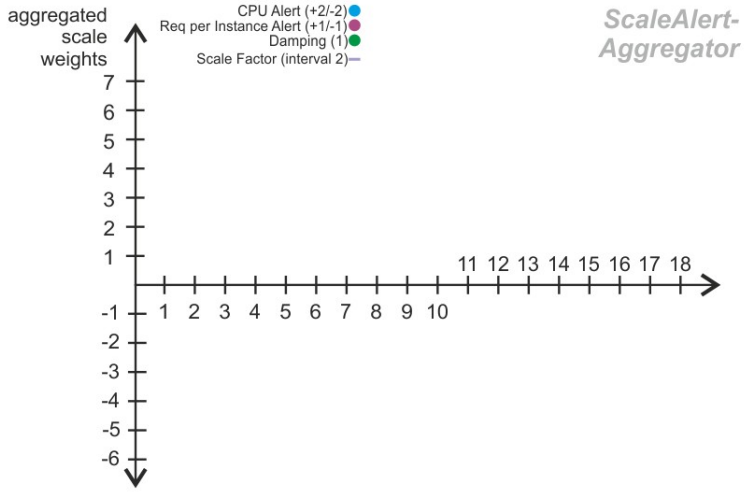
Capacity Planner/ Scaler

SOKAR

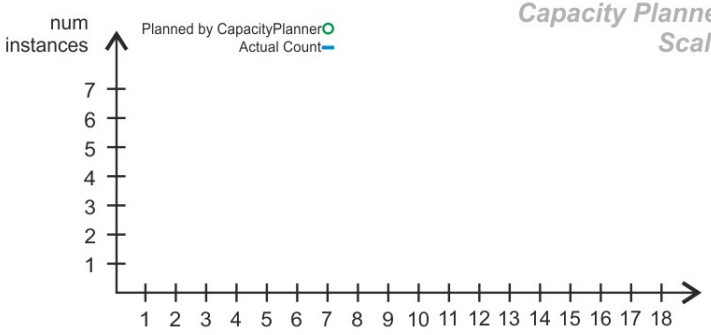


Alerting



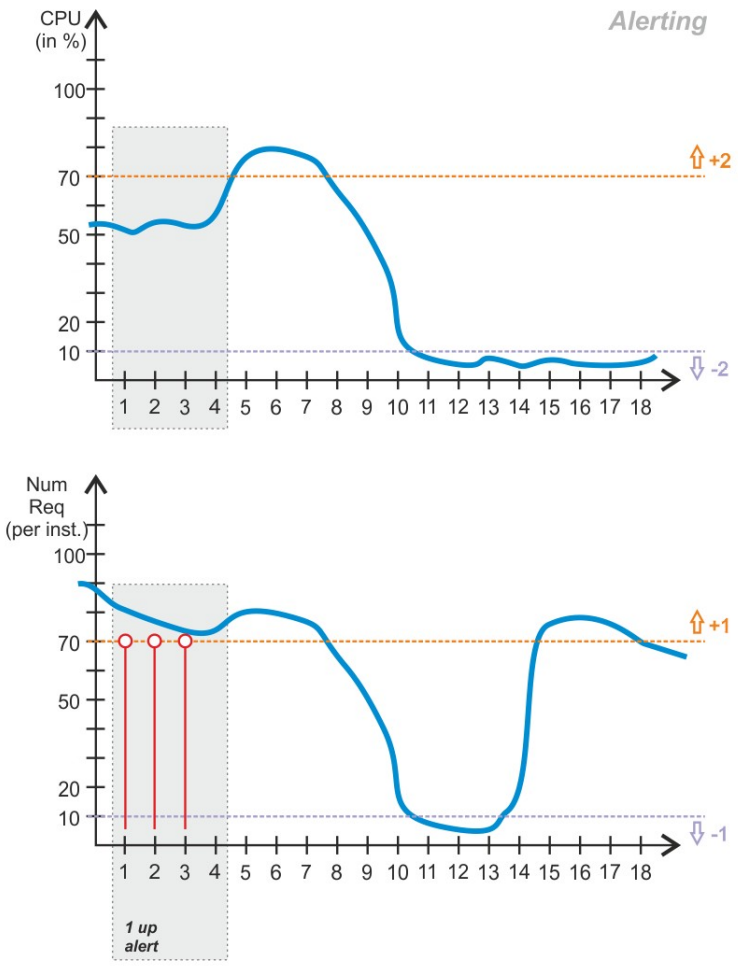


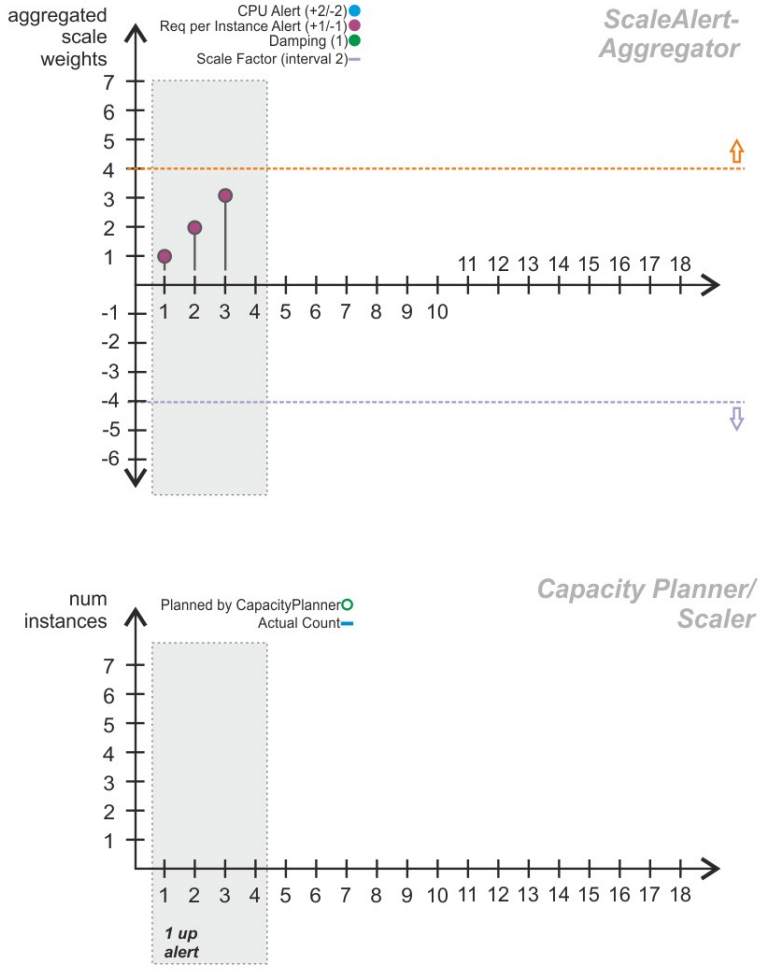
ScaleAlert-Aggregator



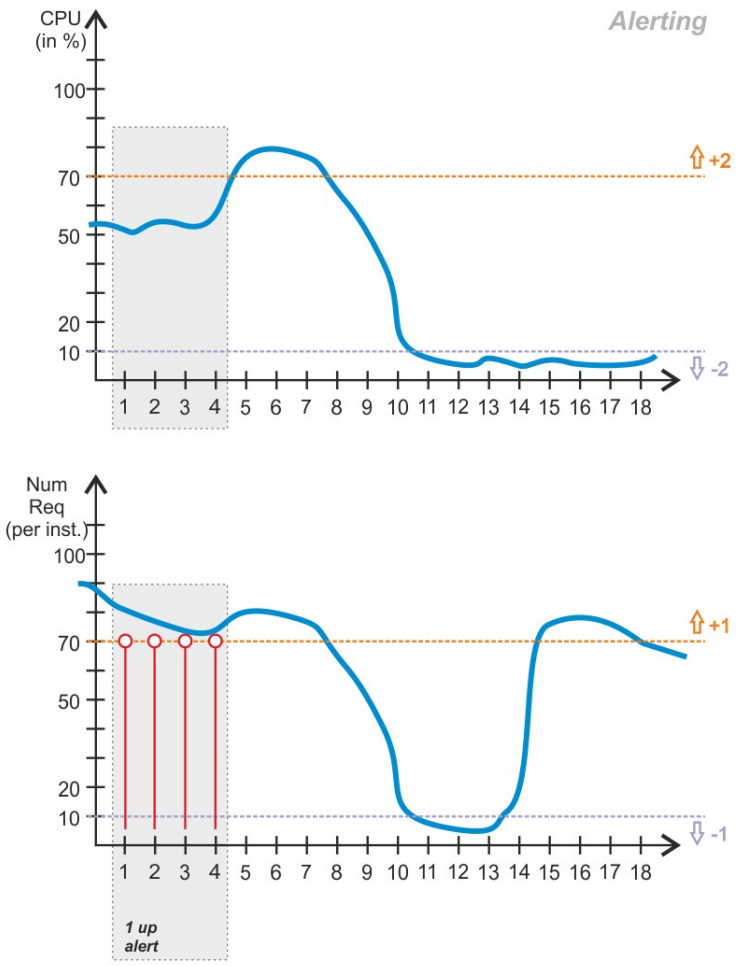
Capacity Planner/ Scaler

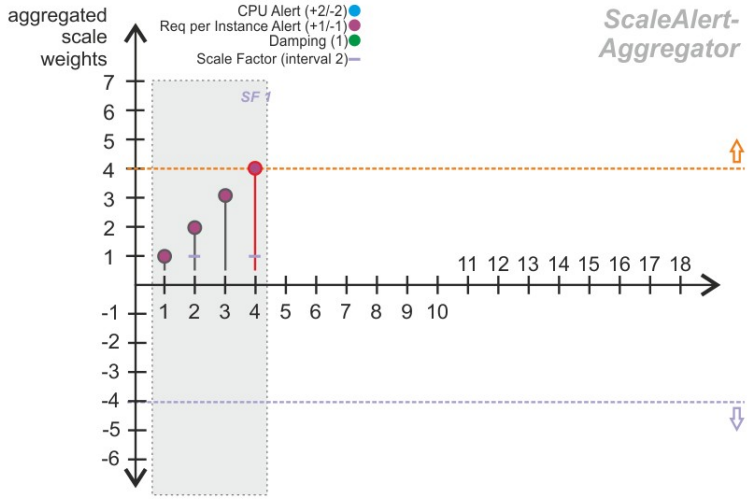
SOKAR

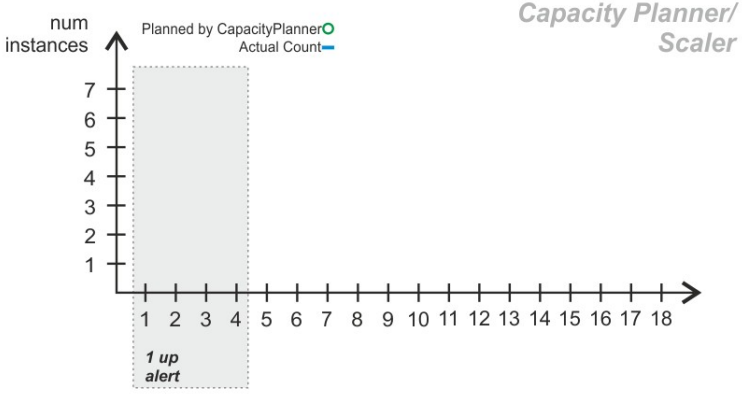




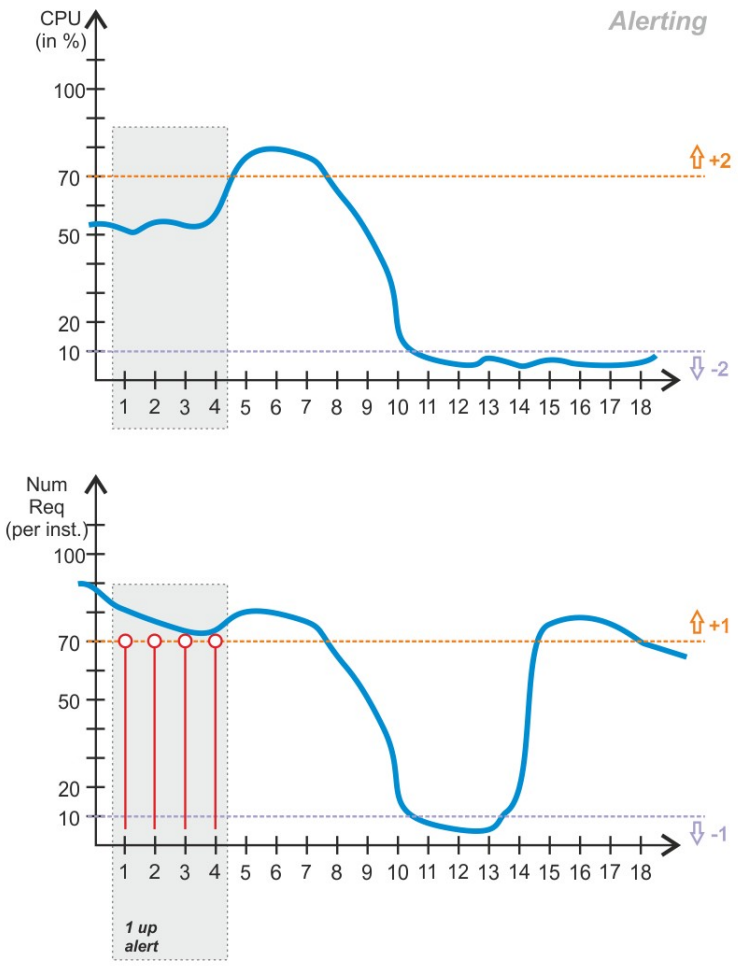
SOKAR

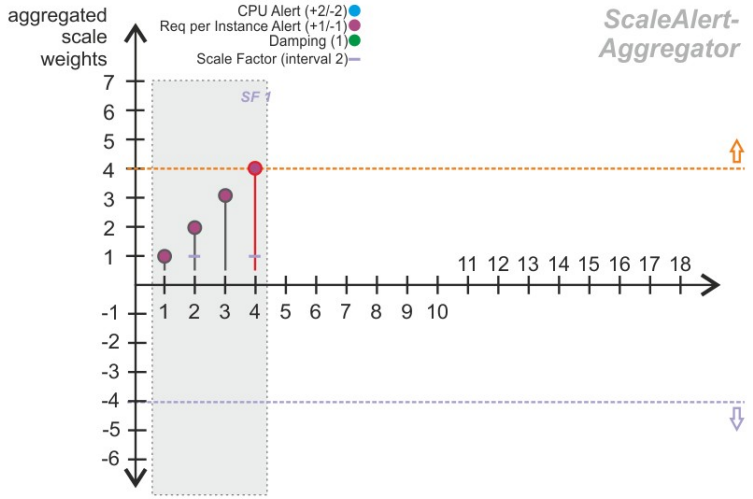


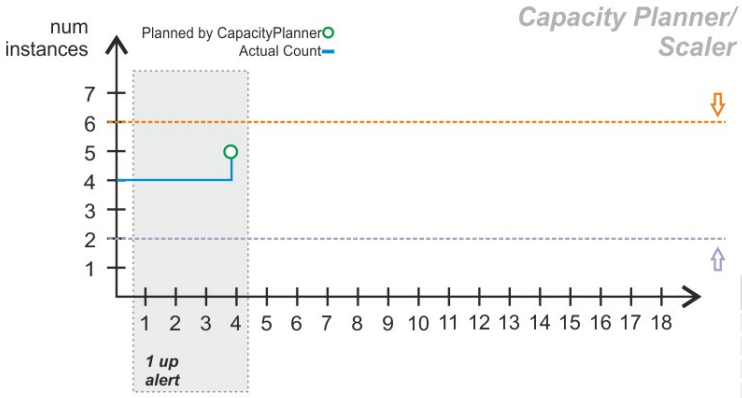




SOKAR

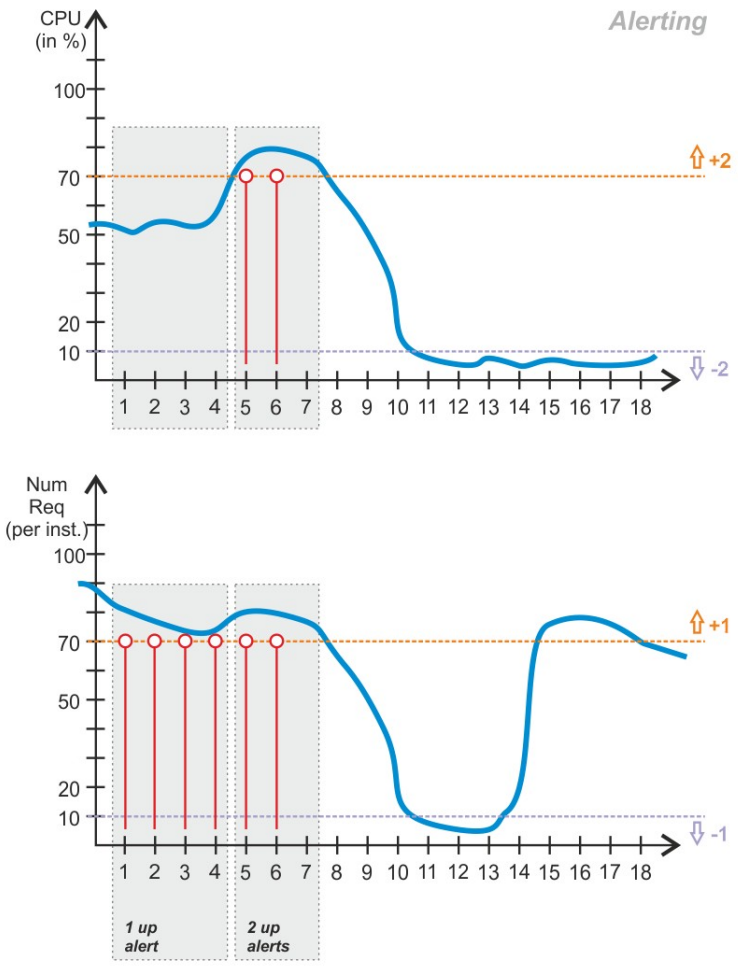


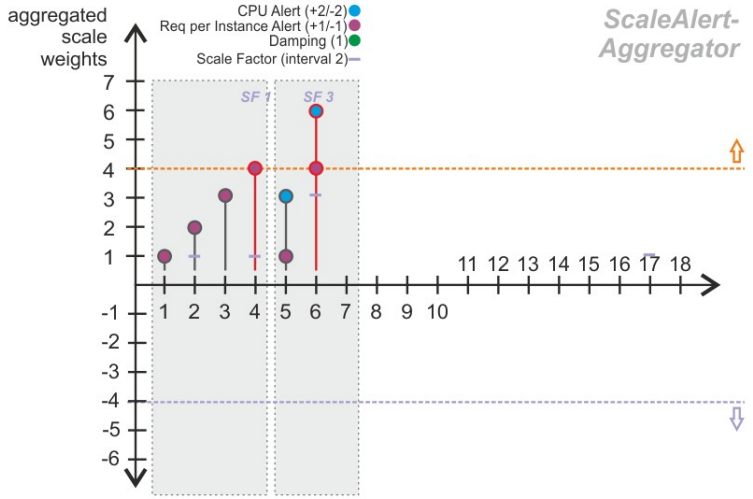


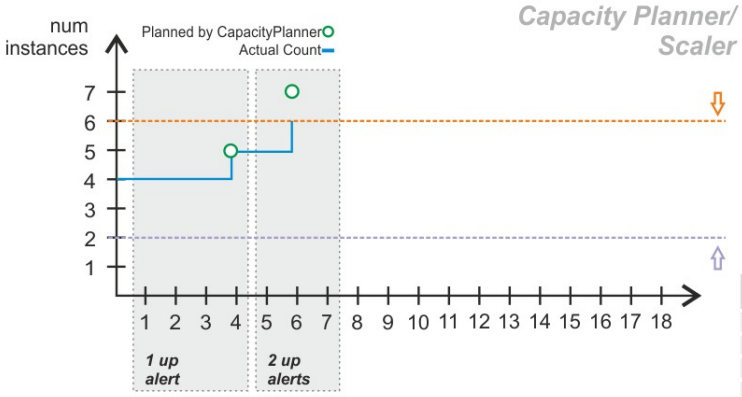


Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

SOKAR

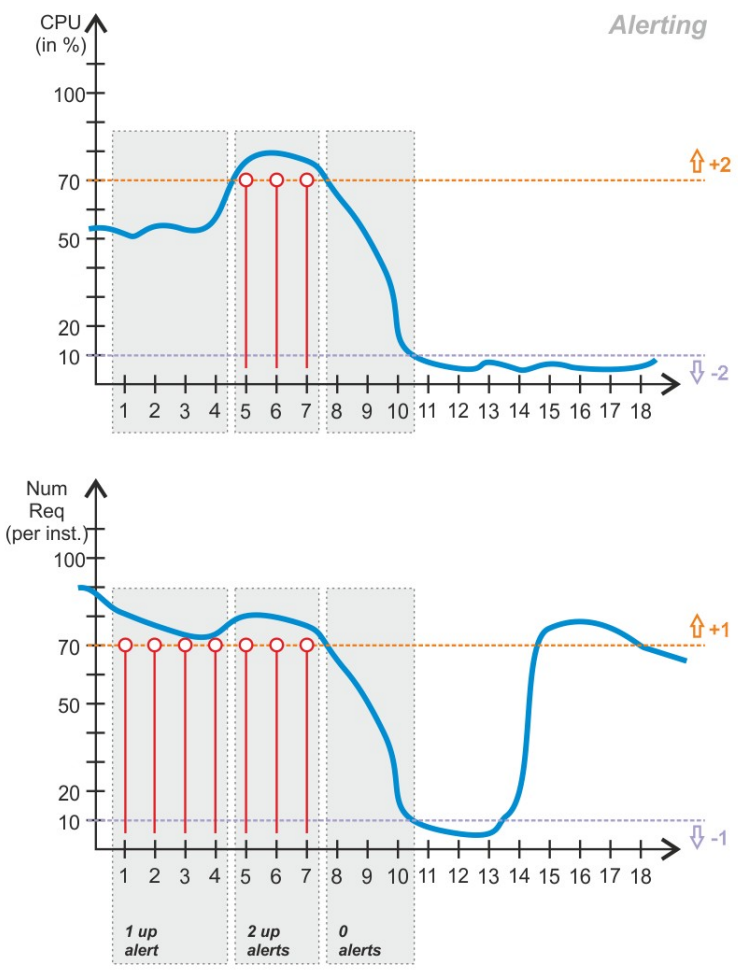




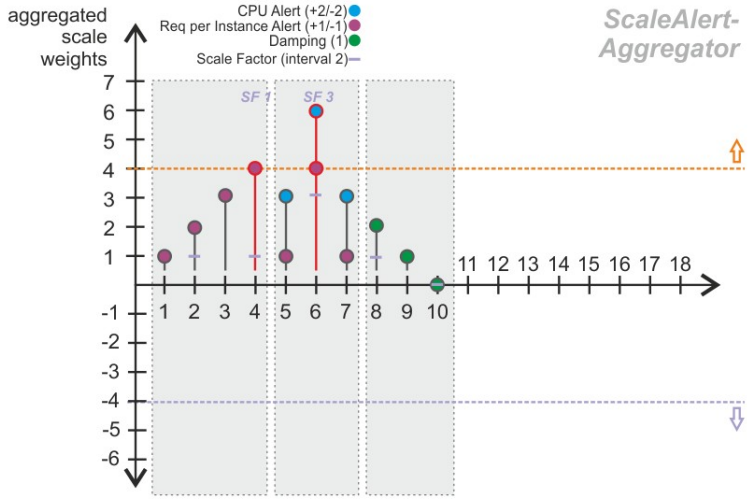


Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

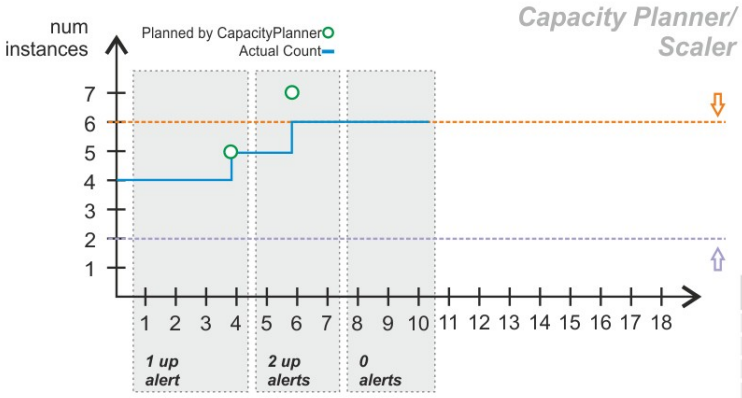
SOKAR



Alerting



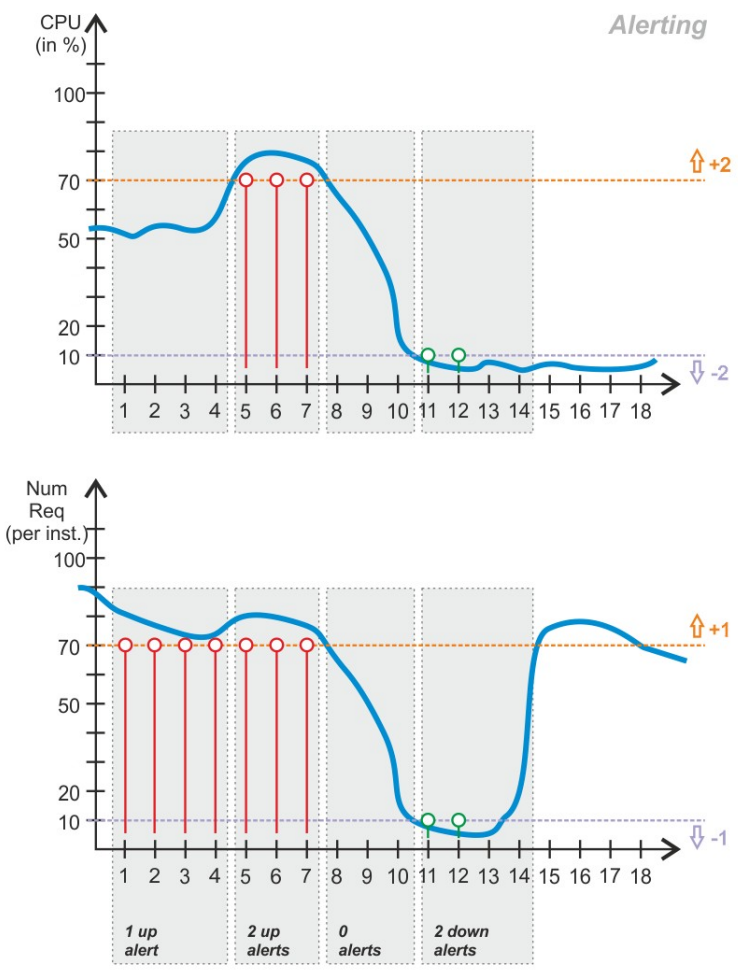
ScaleAlert-Aggregator

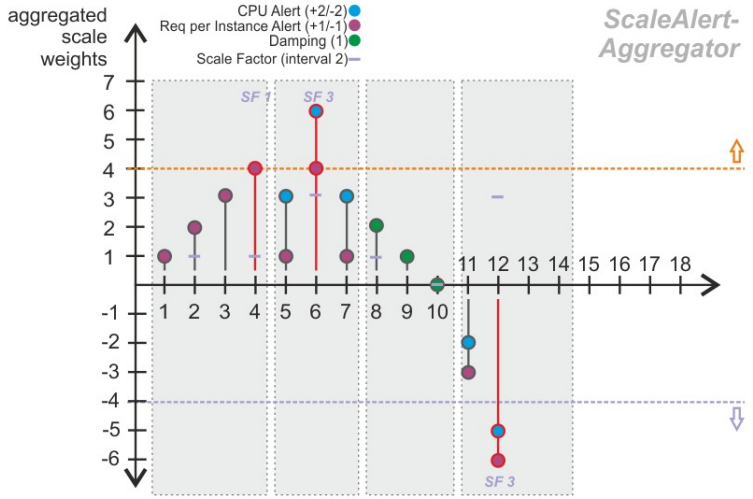


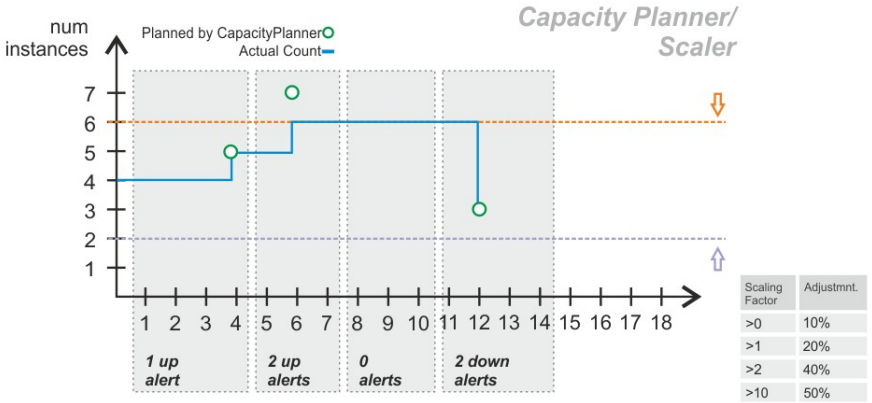
Capacity Planner/ Scaler

Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

SOKAR

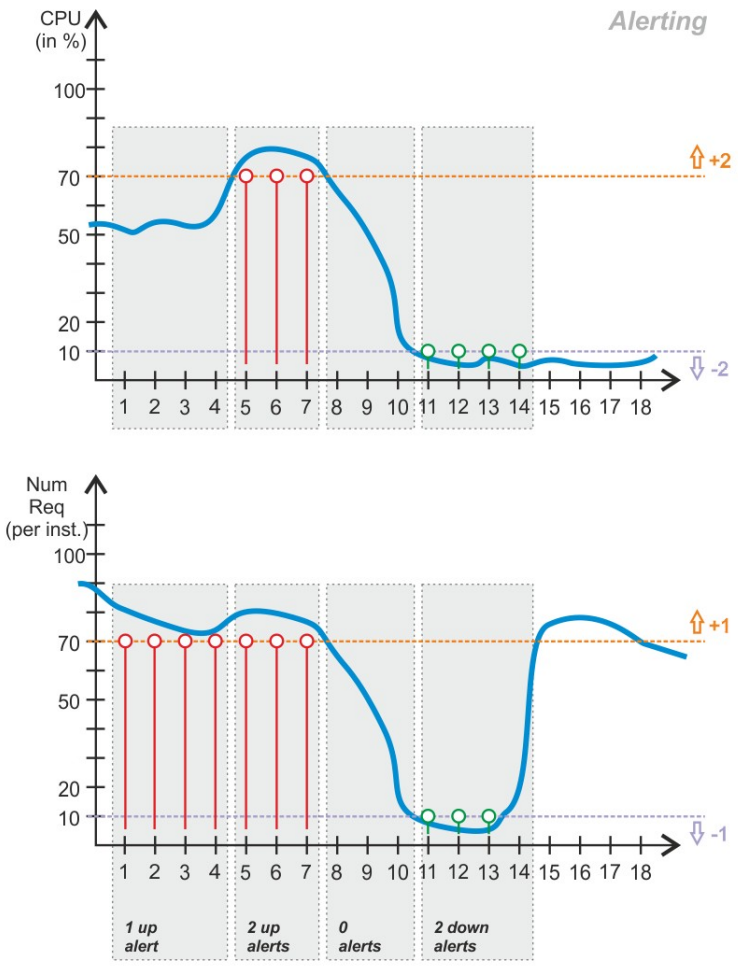


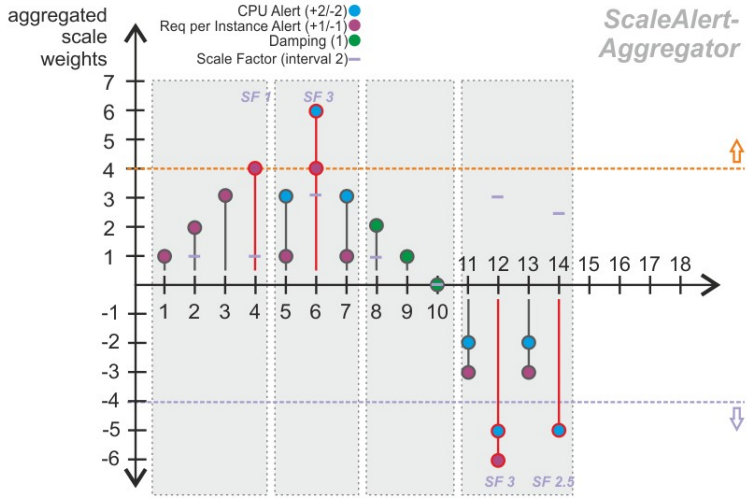


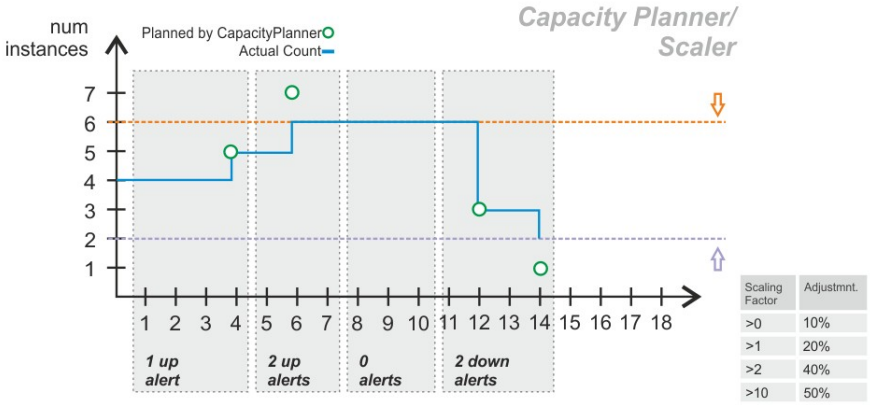


Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

SOKAR

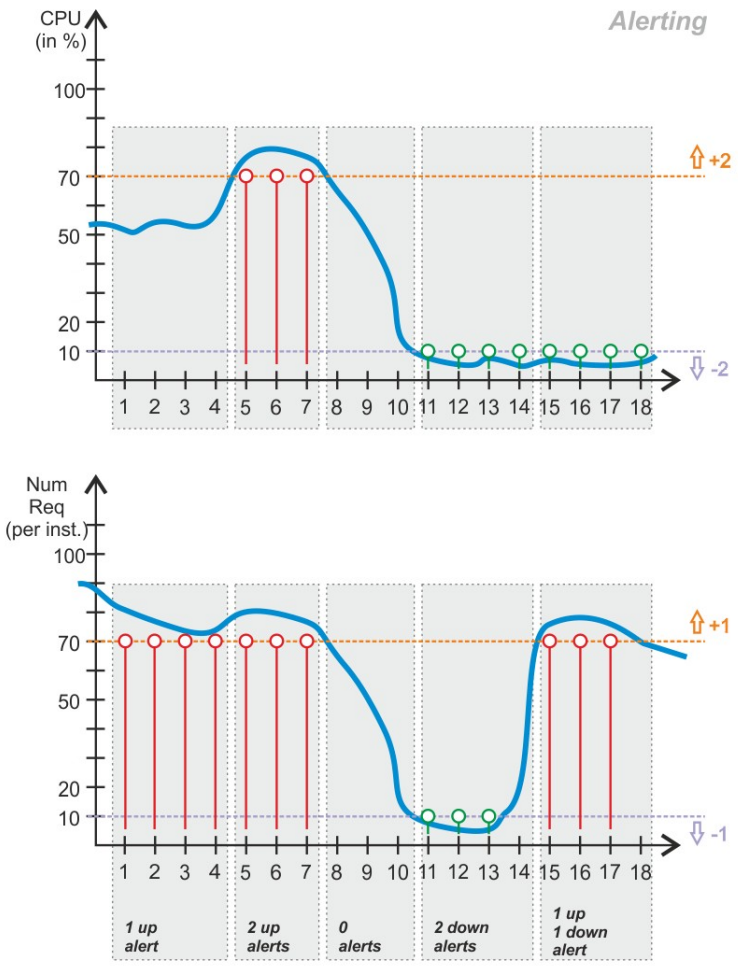


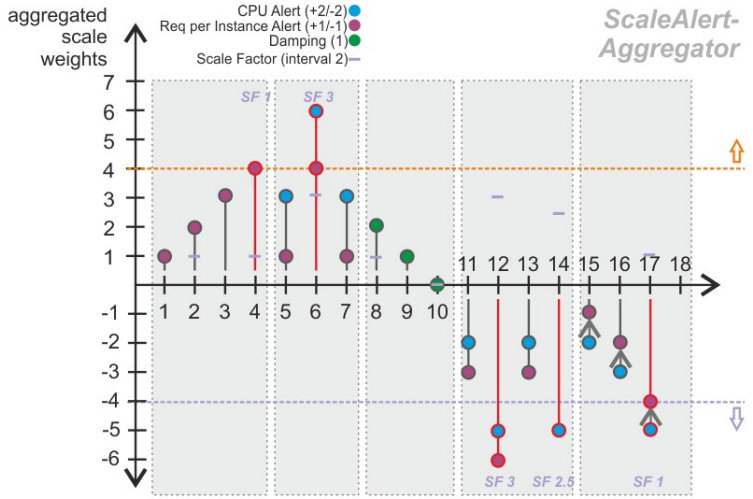


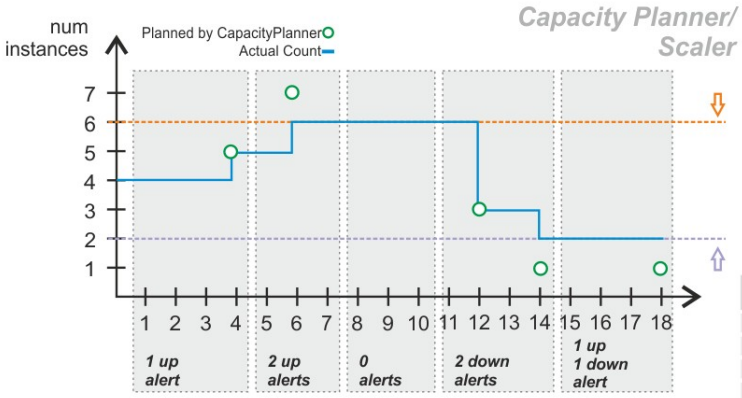


Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

SOKAR







Scaling Factor	Adjustmnt.
>0	10%
>1	20%
>2	40%
>10	50%

