# Pepperdata Capacity Optimizer:
# Real-Time, Automated Resource Optimization at Scale

More Pods Per Node for Kubernetes. More Executors Per Node for YARN.
No Manual Tuning. No Recommendations. No Code Changes.

## The Problem of Resource Overprovisioning

Most application resources are overprovisioned. However, overprovisioning leads to underutilized nodes and wasted spend. Instance rightsizing and manual tuning or applying recommendations cannot address underutilization.

## Real-Time, Automated Resource Optimization for Data Workloads Running on Kubernetes and YARN

**Dynamic, Real-Time Resource Adjustment Based on Actual Usage Levels**

**Capacity Optimizer** remediates resource overprovisioning by providing the system scheduler with real-time visibility into actual CPU, GPU, and memory utilization levels to **automatically deliver more containers per node for both Kubernetes and YARN environments**.

This automatic and continuous **resource rightsizing maximizes utilization levels** for both the peaks and valleys of application runtime **without the need for manual tuning, application code changes, or applying recommendations**.

### Supported Workloads and Environments

- Apache Spark, Apache Flink, Apache Airflow, Jobs, JobController, and CronJobs on Kubernetes, Amazon EKS, Google GKE, Microservices
- Apache Spark on Cloudera Data Engineering
- Amazon EMR, Google DataProc
- Cloudera Data Platform (CDP)

### Supported Schedulers

- Default scheduler on Amazon EMR and EKS and Google GKE
- Apache YuniKorn on Amazon EKS

### Supported Autoscalers

- Amazon EMR Managed Autoscaling and Custom Autoscaling Policy on Amazon EMR
- Cluster Autoscaler and Karpenter on Amazon EKS
- Cluster Autoscaler with and without Node Auto-Provisioning (NAP) on Google GKE

## Four Components of Intelligent Kubernetes Resource Optimization

With Capacity Optimizer installed:

1. **THE SCHEDULER** makes more accurate and efficient resource decisions with Pepperdata-provided real-time CPU and memory usage information.
2. **WORKLOADS ON NODES** are launched based on real-time physical utilization.
3. **PODS** are launched with optimized resource requests.
4. **THE AUTOSCALER** can scale up more efficiently since nodes are packed based on actual utilization.

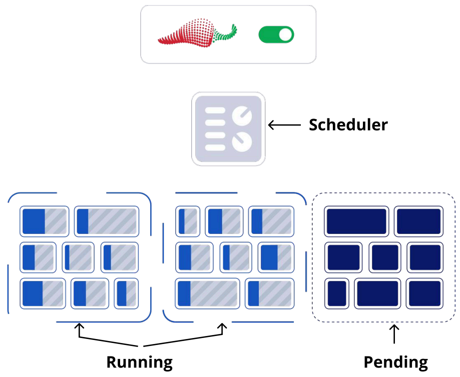
## Enhance Your Autoscaler's Efficiency

Capacity Optimizer also enhances the efficiency of your cloud autoscaler by ensuring new nodes are provisioned only when existing nodes are fully utilized. If there are any pods in the pending state, Capacity Optimizer enables the autoscaler (e.g. Karpenter) to add new nodes only when all existing nodes are fully utilized. This optimizes node scaling without altering downscaling behavior.

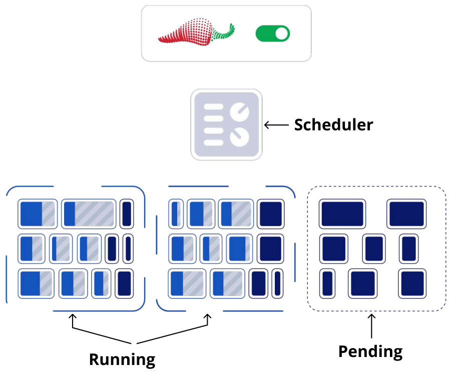
## With Capacity Optimizer Enabled

Before Capacity Optimizer is enabled, the scheduler sees all instances as full. The moment Capacity Optimizer is enabled, more pods automatically run per node since the scheduler now perceives available capacity and can launch pending pods based on actual utilization—resulting in nodes running at their greatest utilization and lowest cost.

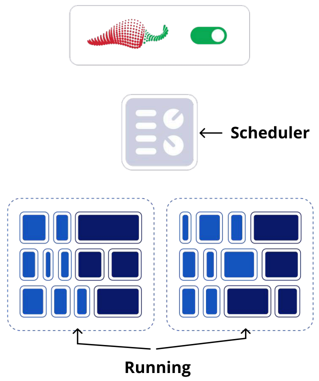
1. **The scheduler** immediately becomes aware of existing but unused node capacity.



2. **Pending pods** are now launched to fill actual available resources.



3. **Resource utilization is increased**, costs are reduced, and throughput performance is improved.



## Capacity Optimizer Benefits

### Improve Resource Utilization

By creating more containers per node, Capacity Optimizer increases utilization by up to 80 percent—eliminating waste and maximizing efficiency.

### Reduce Cost Across All Your Clusters

By optimizing resources automatically in real time, Capacity Optimizer cuts cloud spend by up to 75 percent—with no developer effort.

### Free Your Engineers: No Manual Tuning, No Recommendations, No Application Code Changes

Capacity Optimizer works continuously and automatically in the background with no tedious recommendations to implement and no need for developers to modify code or tune configs.

### Instantly Reclaim Wasted CPU, GPU, and Memory Resources

Ensure maximum throughput performance by continuously optimizing GPU, CPU, and memory in real time.

## YARN/Amazon EMR Environments

Similar to how it works in Kubernetes, Capacity Optimizer provides actual usage information to the YARN/Amazon EMR scheduler so that new containers are launched based on actual usage rather than on allocated resource levels. Capacity Optimizer also guides the Amazon EMR autoscaler to add new nodes only when existing ones are fully utilized. In this manner, Capacity Optimizer improves utilization, reduces instance hours, and lowers cloud costs in YARN, Amazon EMR, and Google Dataproc environments.

### About Pepperdata

Deployed on over 30,000+ clusters, Capacity Optimizer optimizes resources in some of the largest and most complex environments in the world, providing more pods per node in Kubernetes environments, and more executors per node in YARN environments. Since 2012 Pepperdata has helped companies ranging from startups and mid-sized ISVs to top enterprises such as Citibank, Autodesk, Magnite, Royal Bank of Canada, and members of the Fortune 10 save over $250 million. For more information, visit www.pepperdata.com.

aws partner network
Select
Technology
Partner

AICPA SOC
AICPA Service Organization Control Reports
aicpa.org/soc
Formerly SAS 70 Reports

100% ROI GUARANTEE

Pepperdata, Inc.
530 Lakeside Drive
Suite 170
Sunnyvale, CA 94085

**Start a Free Trial**
www.pepperdata.com

**Send an Email**
info@pepperdata.com