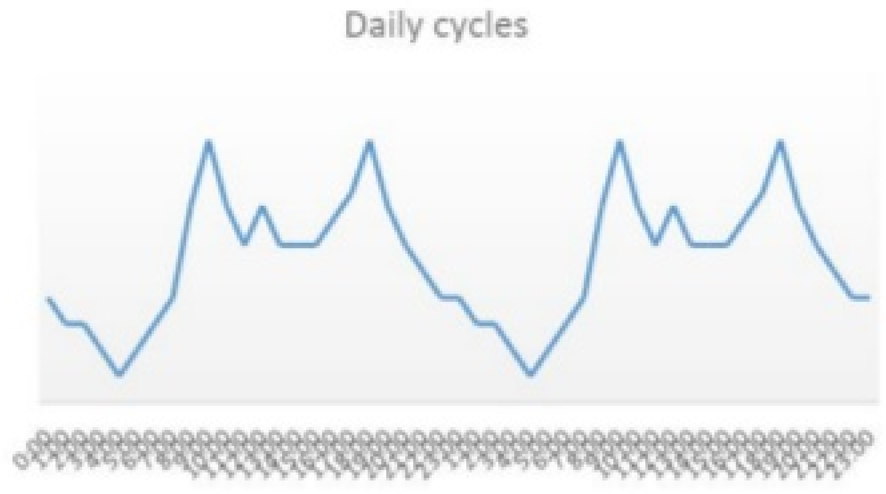
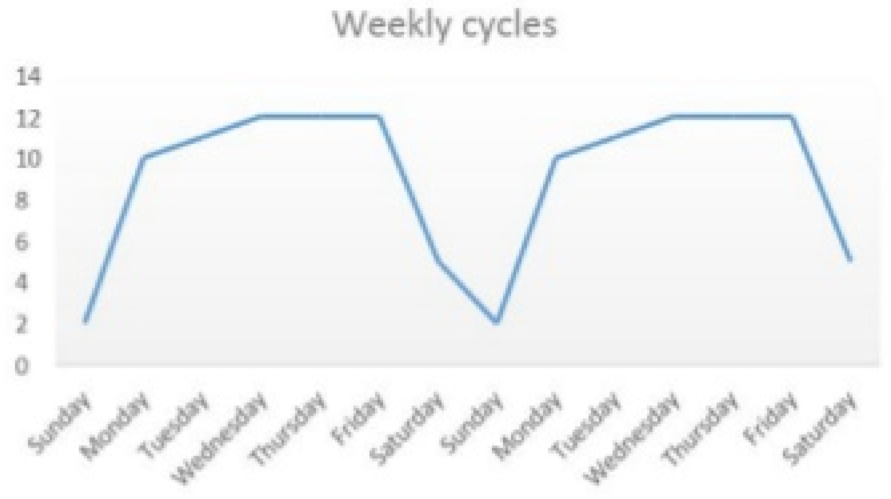
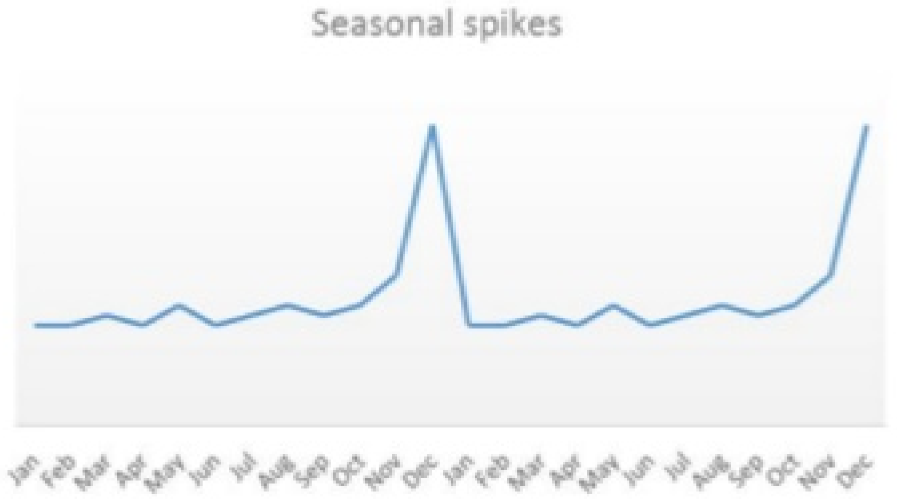
@stsffap @joerg_schad

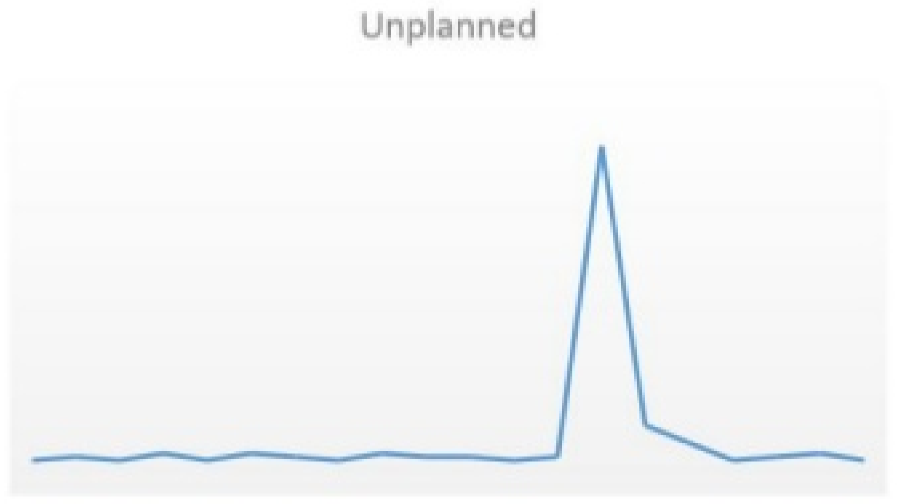
Elastic Streams at Scale

Workload cycles and spikes

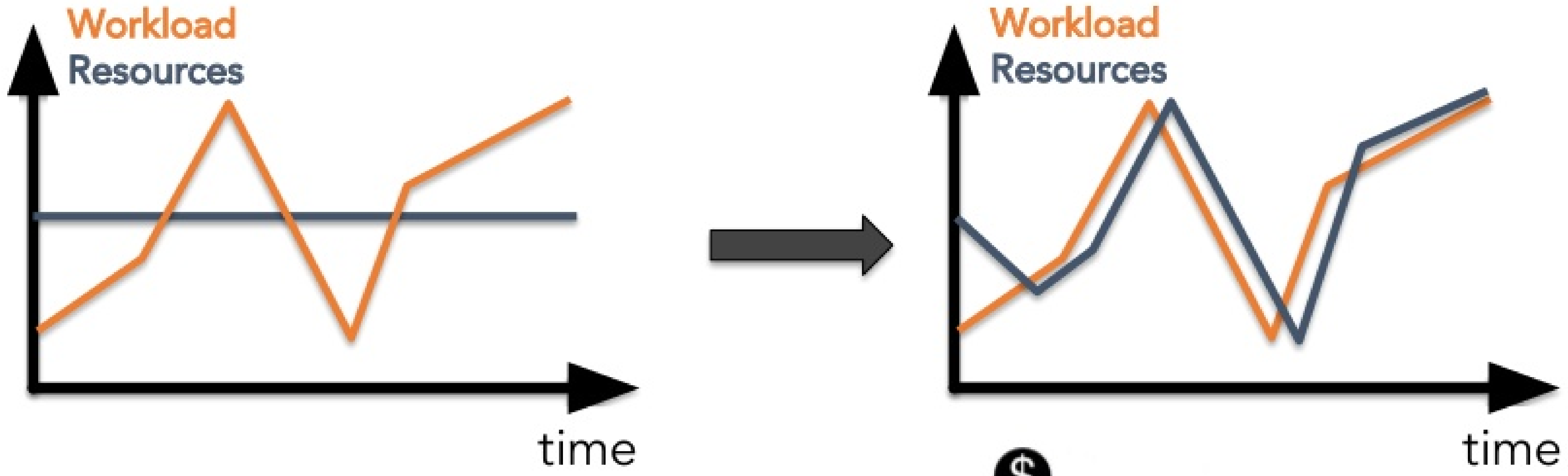


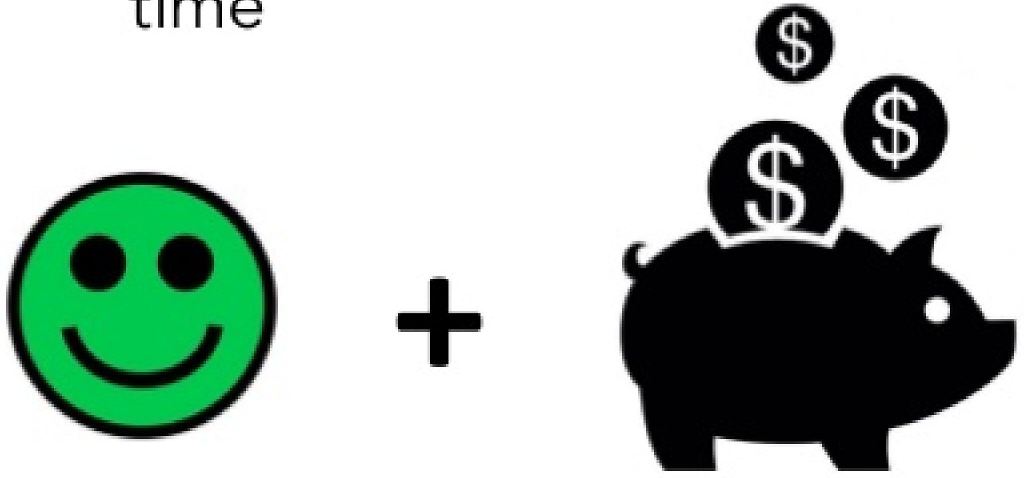






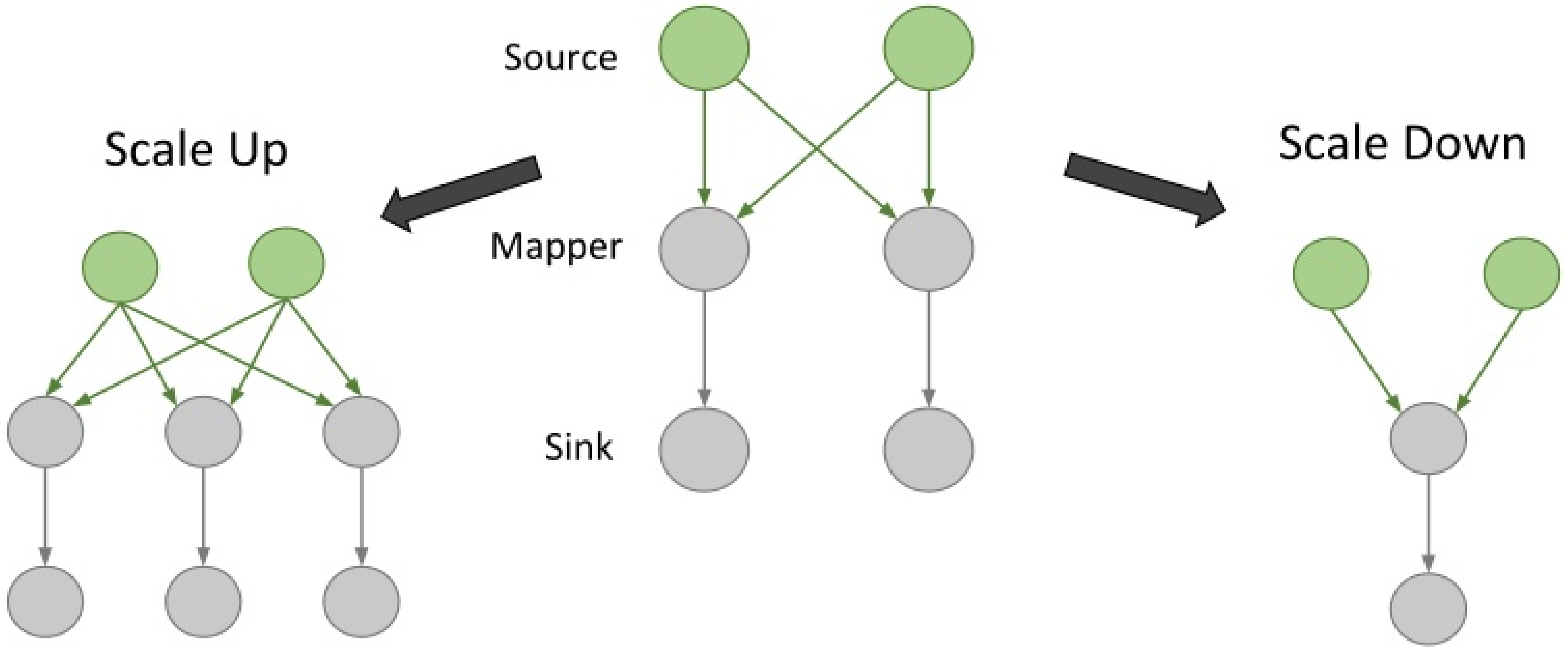
Resource adaption





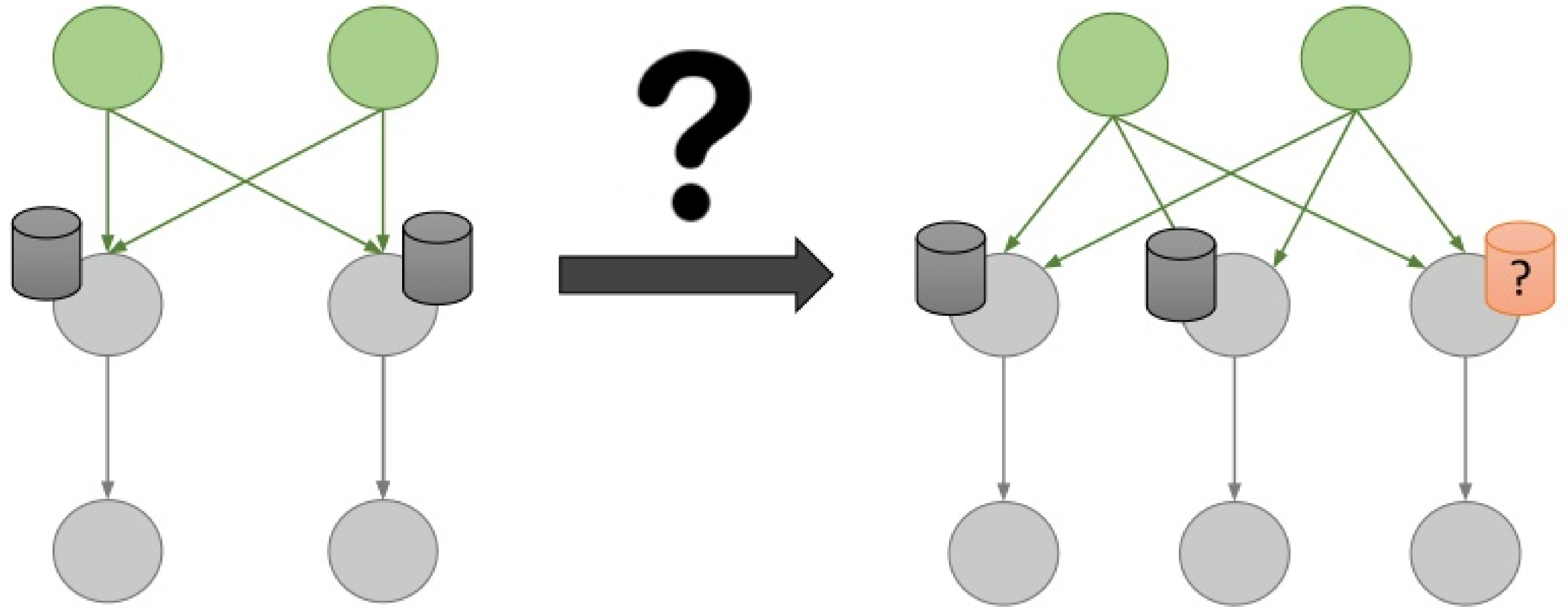
Scalable Flink applications

Scaling stateless jobs



- Scale up: Deploy new tasks
- Scale down: Cancel running tasks

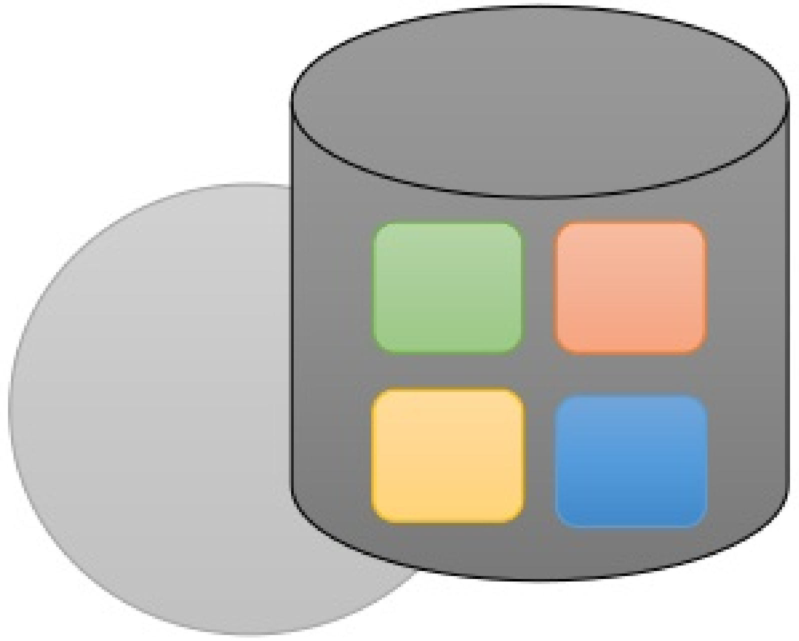
Scaling stateful jobs



- Problem: Which state to assign to new task?

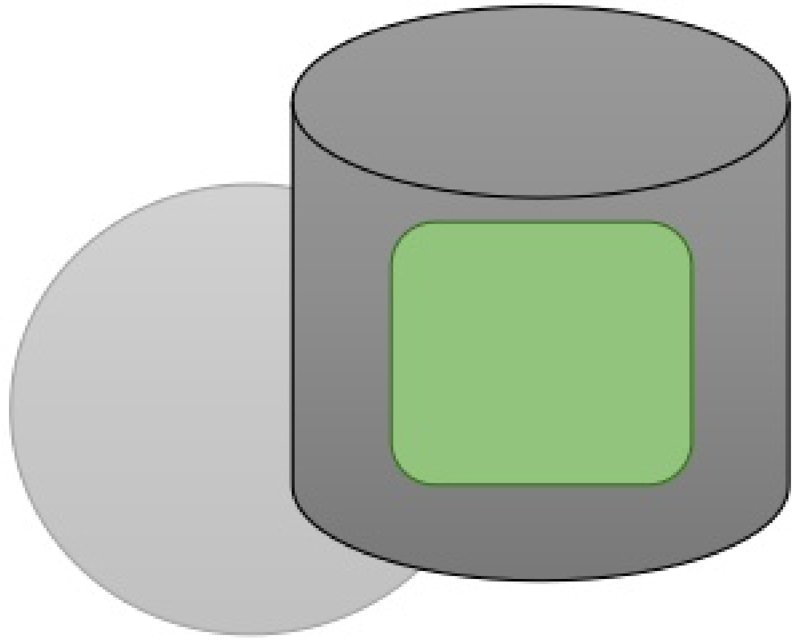
Keyed vs. operator state

Keyed



- State bound to a key
- E.g. Keyed UDF and window state

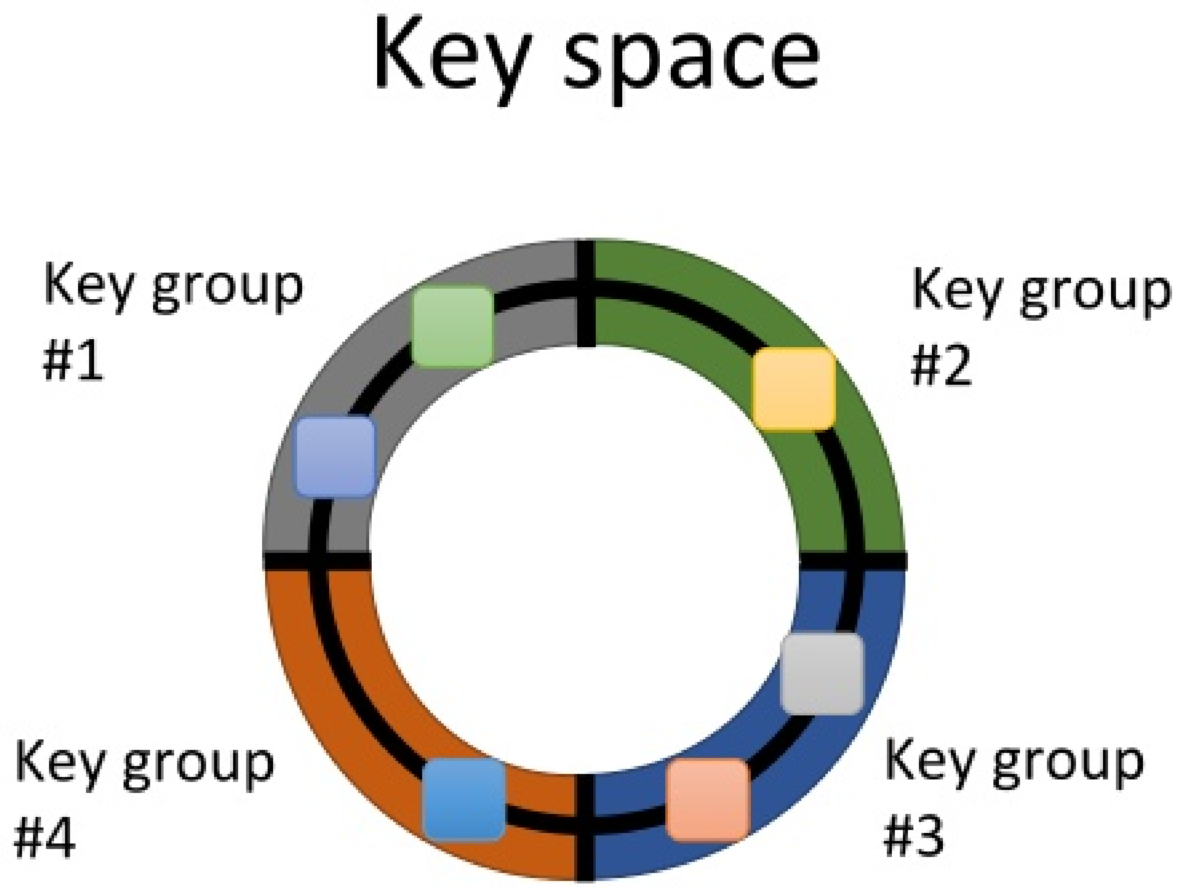
Operator



- State bound to a subtask
- E.g. Source state

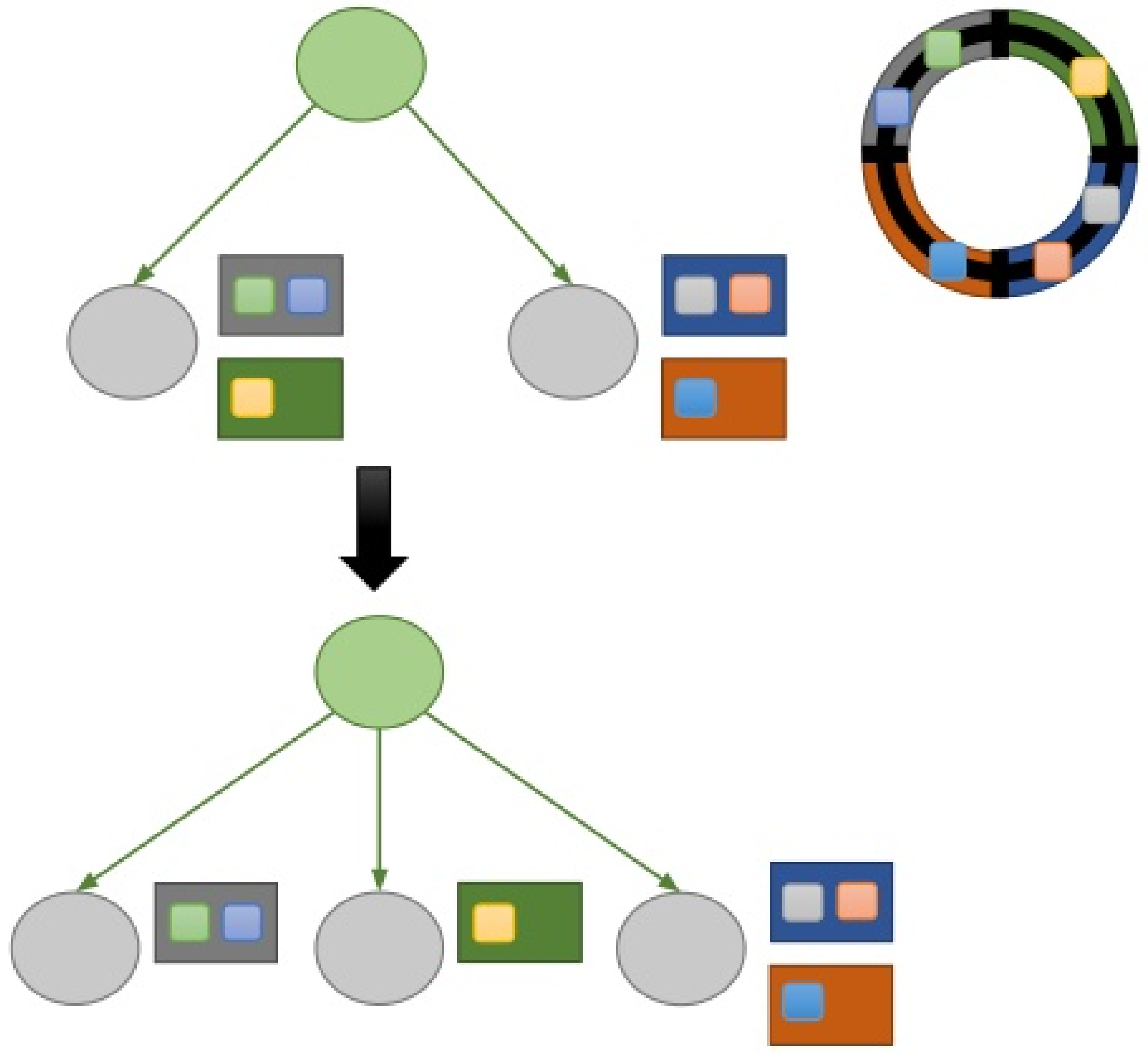
Repartitioning keyed state

- Similar to consistent hashing
- Split key space into key groups
- Assign key groups to tasks



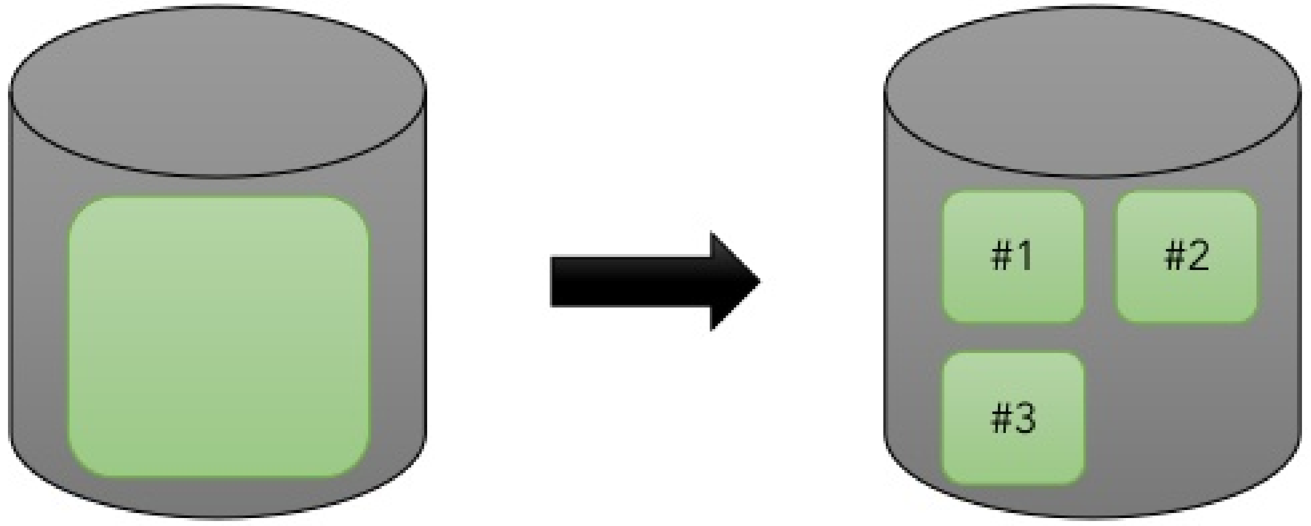
Repartitioning keyed state contd.

- Rescaling changes key group assignment
- Maximum parallelism defined by #key groups

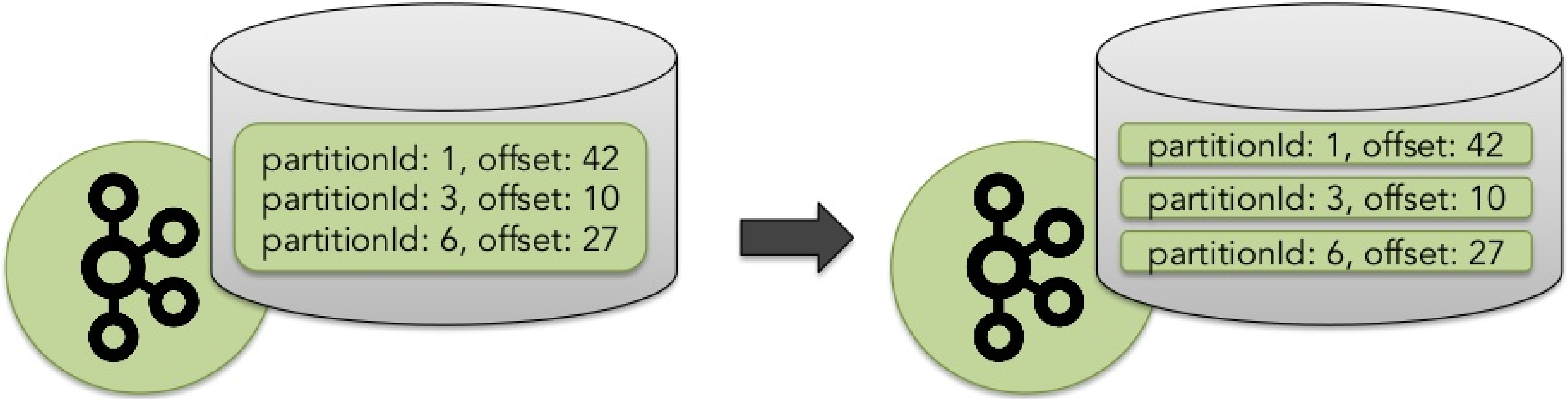


Repartitioning operator state

- Breaking operator state up into finer granularity
  - State has to contain multiple entries
  - Automatic re-partitioning wrt granularity

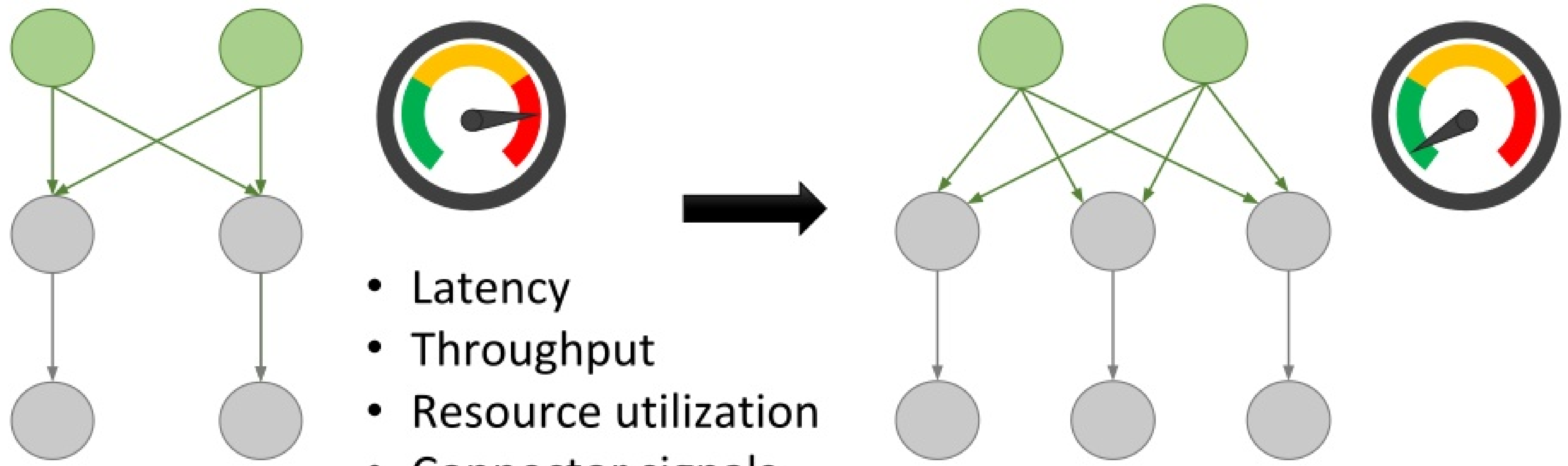


Example: Kafka Source


partitionId: 1, offset: 42
partitionId: 3, offset: 10
partitionId: 6, offset: 27
partitionId: 1, offset: 42
partitionId: 3, offset: 10
partitionId: 6, offset: 27

- Store offset for each partition
- Individual entries are repartitionable

Automatic scaling



- Latency
- Throughput
- Resource utilization
- Connector signals

Resource elasticity

Where to get the resources from?

- Scaling Flink applications changes required number of slots
- Flink needs to start and stop TaskManagers
- ClusterManager is required to start TaskManagers dynamically

Apache Mesos in a nutshell

- Cluster management framework
  - Dynamic resource allocation
  - Running multiple applications
  - 2-level scheduling
- Fault-tolerant, battle-tested
- Scalable to 10,000+ nodes
- Created by Mesosphere founder @ UC Berkeley, used in production by 100+ web-scale companies

Why Flink on Mesos?

- Mesos offers full functionality to implement fault tolerant and elastic distributed applications

- 30% of survey respondents were running Flink on Mesos (prior to proper Mesos support*, September 2016)

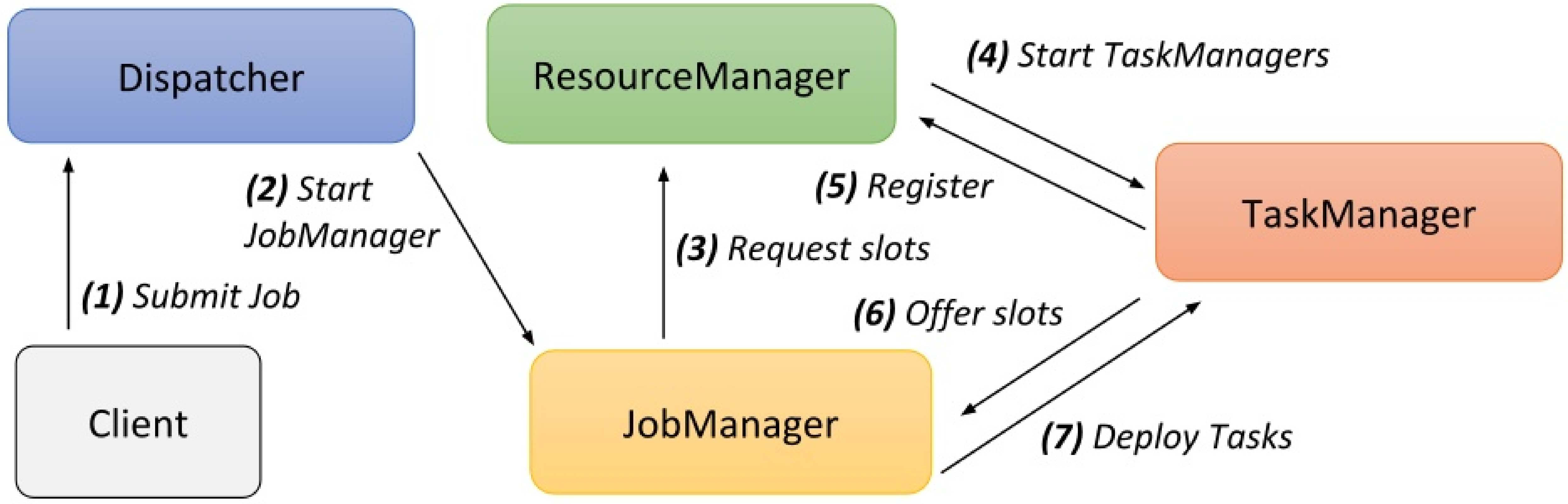
- Other Deployment Models
  - Standalone
  - Yarn
  - Kubernetes

Flink's Revamped Distributed Architecture

- Motivation
  - Resource elasticity
  - Support for different deployments
  - REST interface for client-cluster communication

- Introduce generic building blocks
- Compose blocks for different scenarios

The Building Blocks



The Building Blocks

ResourceManager

- ClusterManager-specific
- May live across jobs
- Manages available Containers/TaskManagers
- Used to acquire / release resources

Dispatcher

- Lives across jobs
- Touch-point for job submissions
- Spawns JobManagers

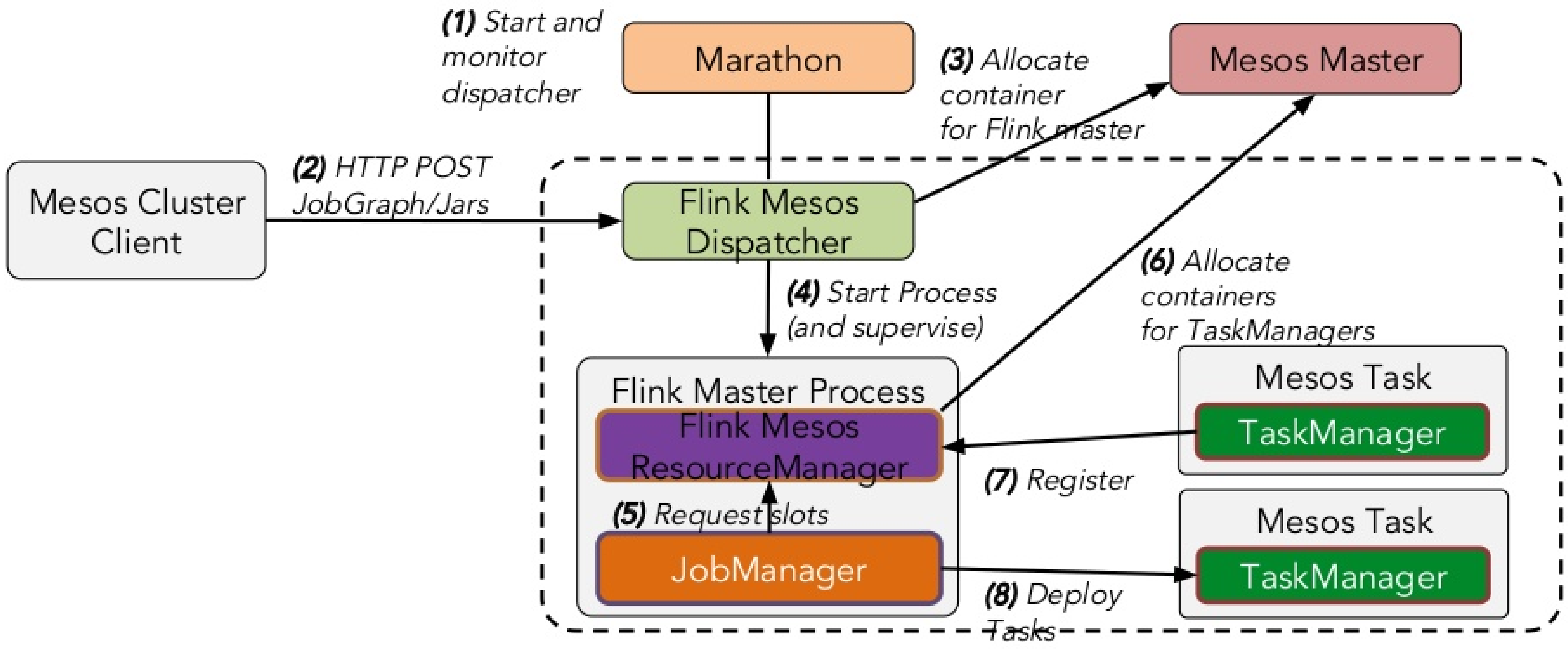
JobManager

- Single job only, started per job
- Thinks in terms of "task slots"
- Deploys and monitors job/task execution

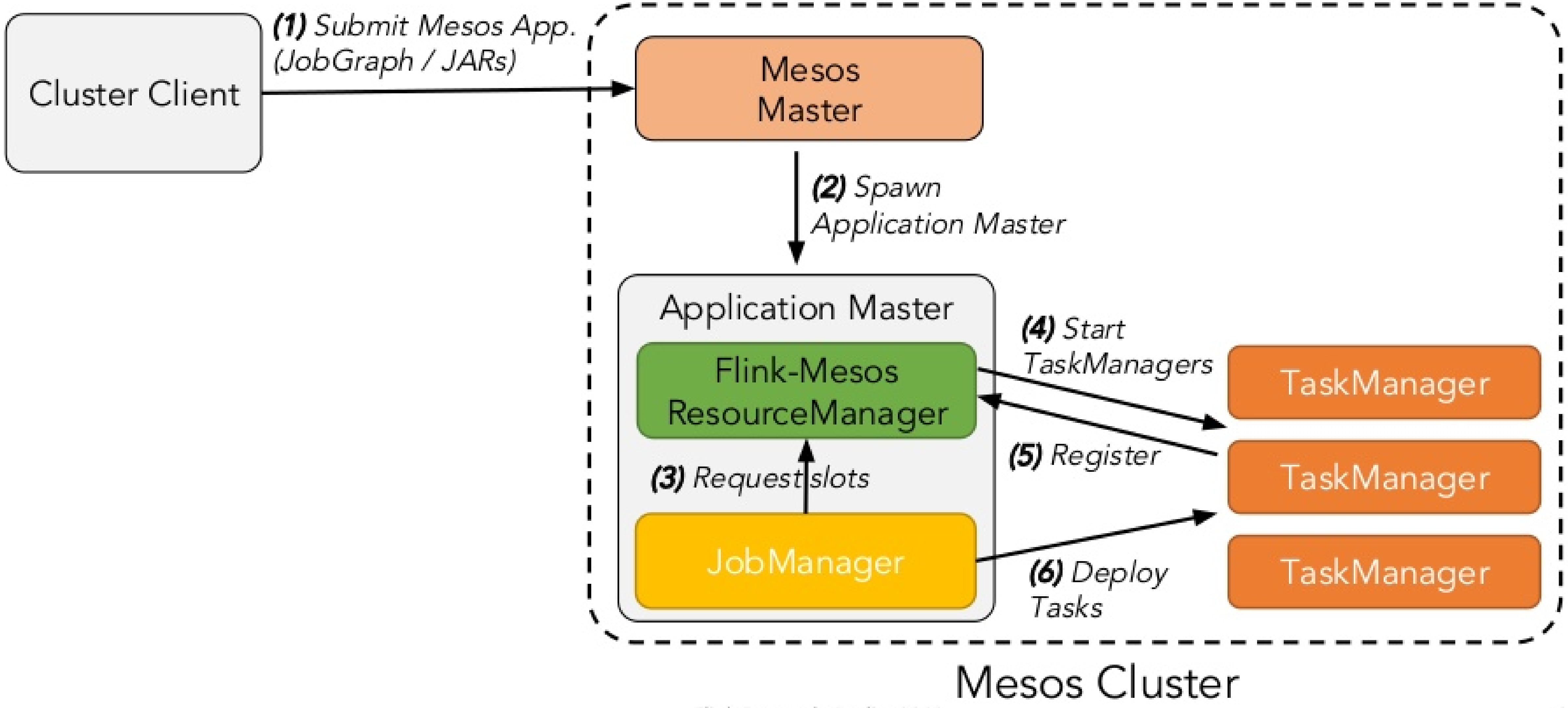
TaskManager

- Registers at ResourceManager
- Gets tasks from one or more JobManagers

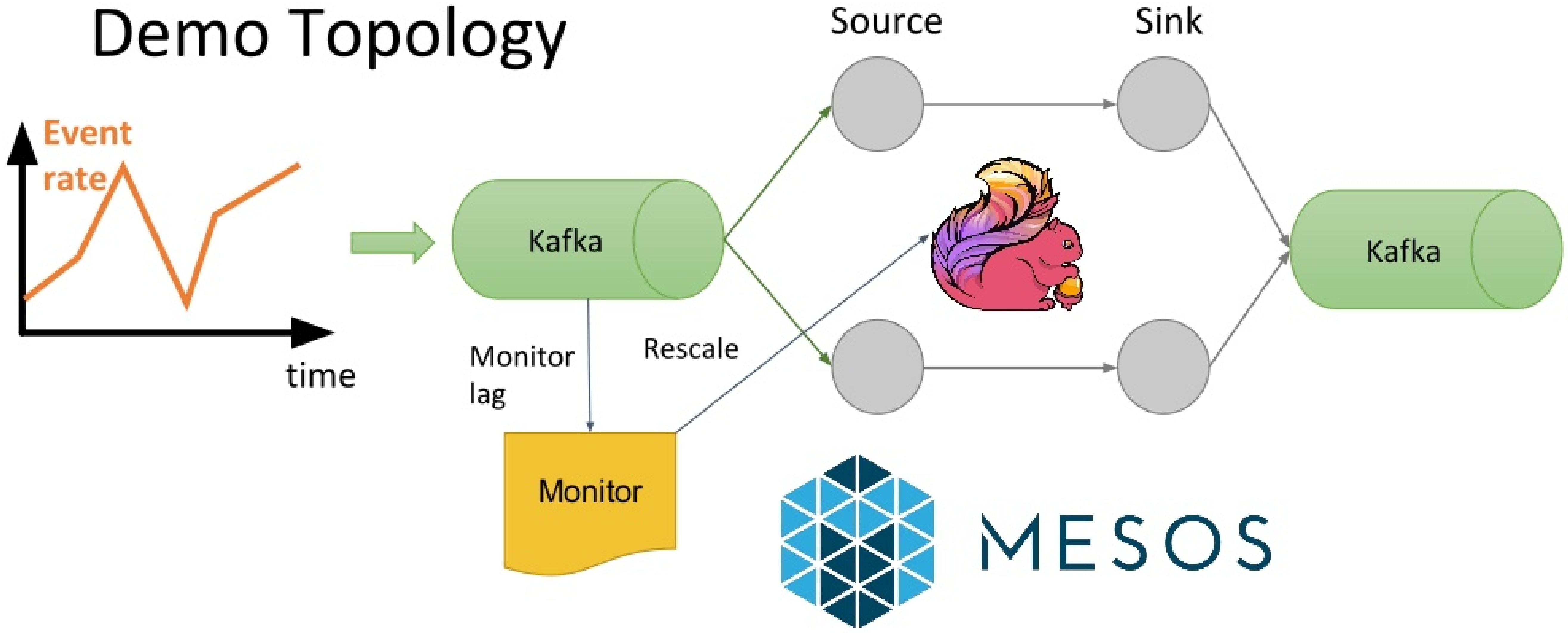
Flink Mesos Integration


(1) Start and monitor dispatcher
Marathon
(3) Allocate container for Flink master
Mesos Master
Mesos Cluster Client
(2) HTTP POST JobGraph/Jars
Flink Mesos Dispatcher
(4) Start Process (and supervise)
(6) Allocate containers for TaskManagers
Flink Master Process
Flink Mesos ResourceManager
(5) Request slots
JobManager
Mesos Task
TaskManager
(7) Register
Mesos Task
TaskManager
(8) Deploy Tasks

Building Flink-on-Mesos (job mode)



Demo Topology



- Executed on Mesos to support dynamic resource allocation

Wrap Up

- Flink supports resource elasticity
- Flink applications can adapt to changing workloads
  - Currently manually
  - Future automatically via re-scaling policies
- Flink integrates natively with Mesos

THANK YOU!

@stsffap
@dataArtisans
@ApacheFlink

WE ARE HIRING
data-artisans.com/careers

dataArtisans