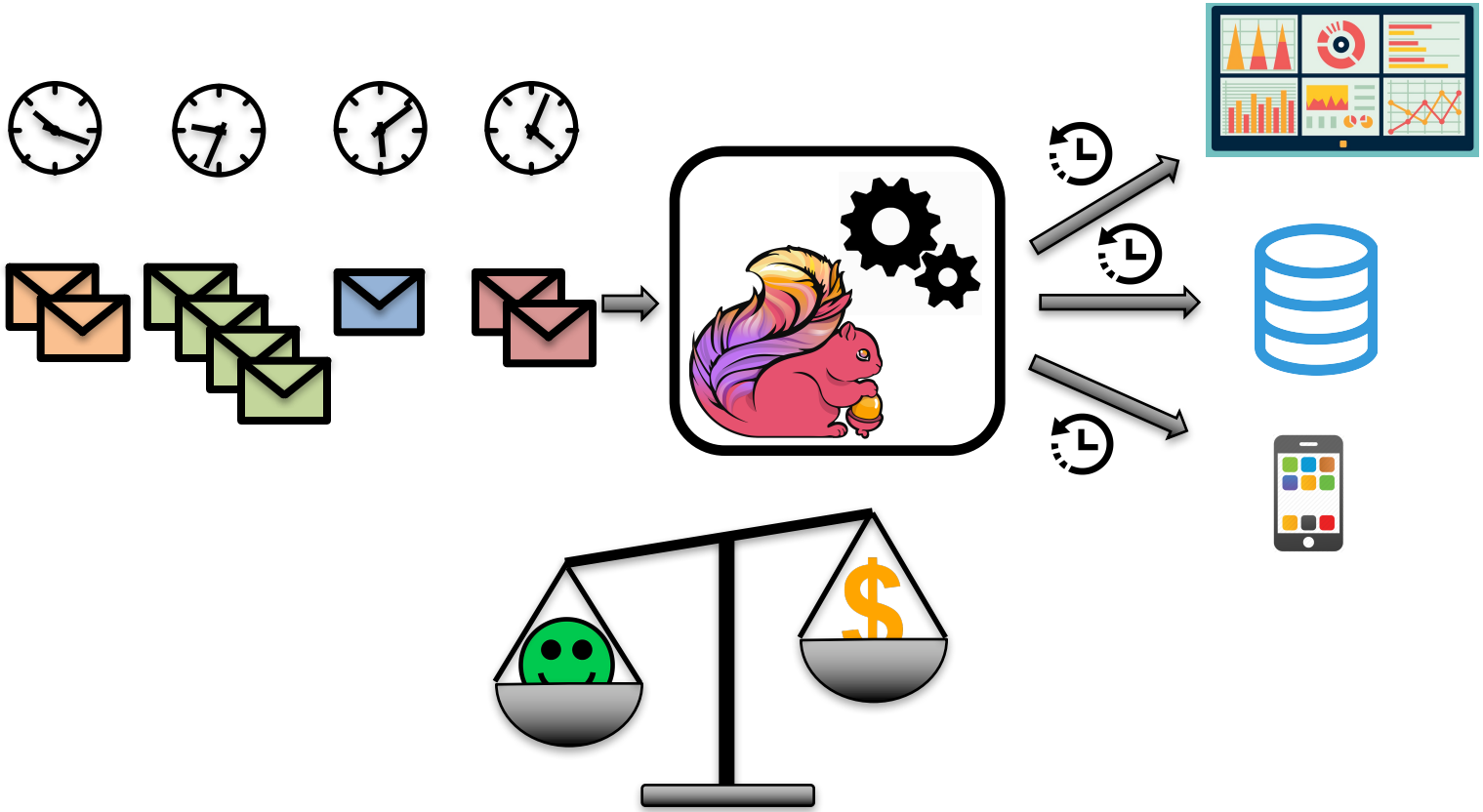
Dynamic Scaling: How Apache Flink® Adapts to Changing Workloads

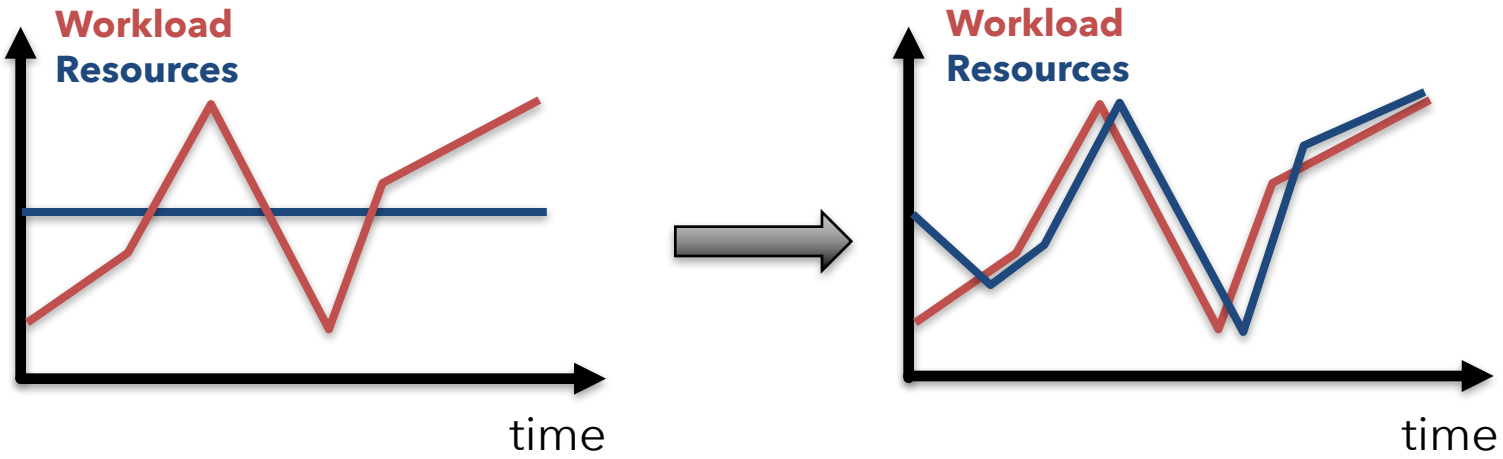
Till Rohrmann
trohrmann@apache.org
@stsffap

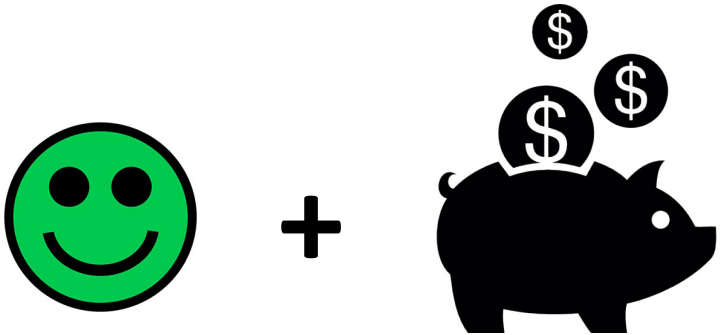
dataArtisans

Changing Workloads And SLAs



Resource Adaption



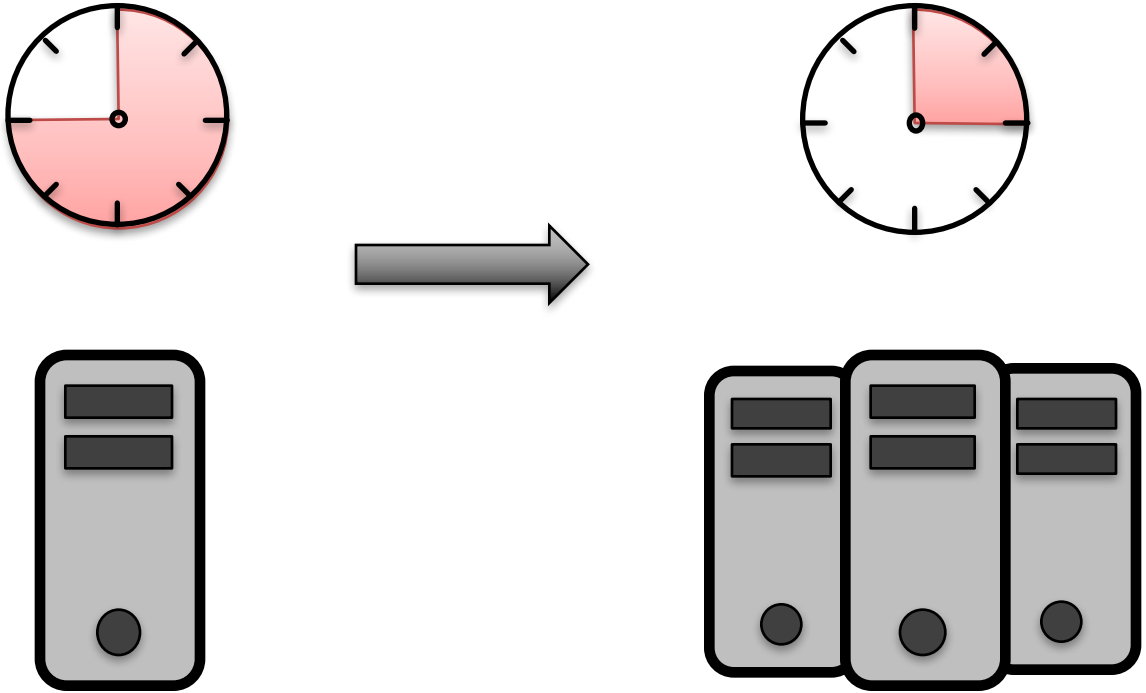


What Is This Talk About?

- Flink's approach to dynamic scaling
- Current state with demo
- Outlook on next development steps

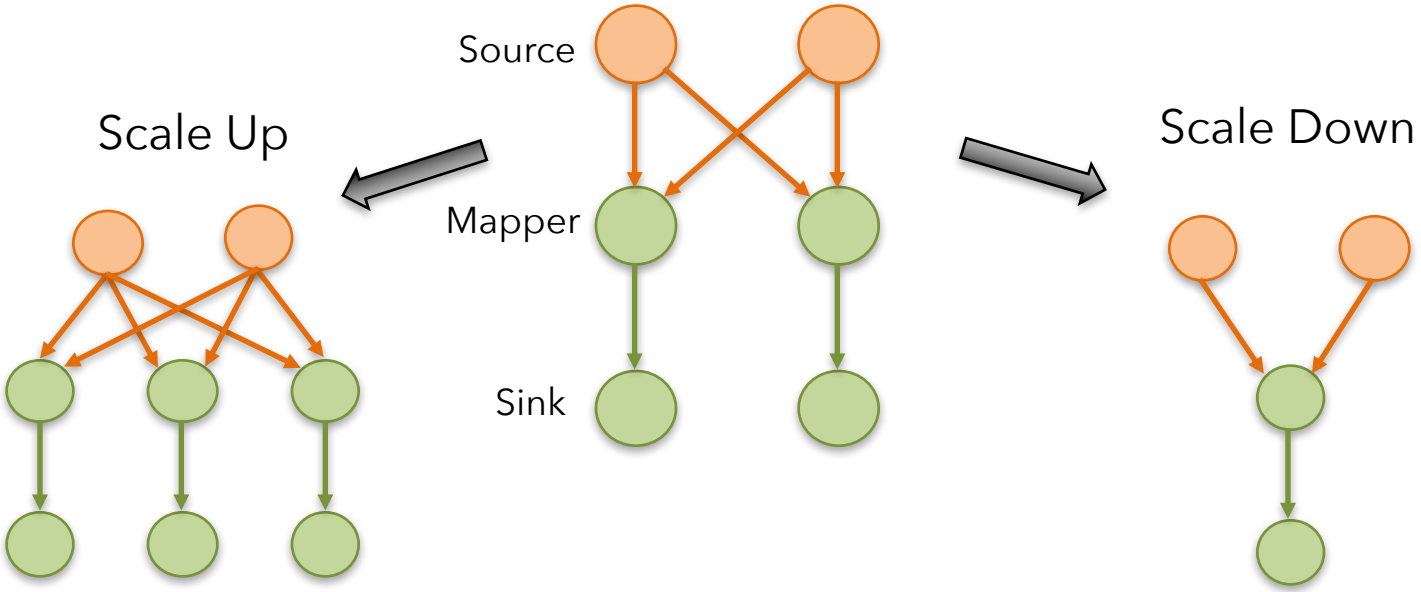
Dynamic Scaling

Basic Idea



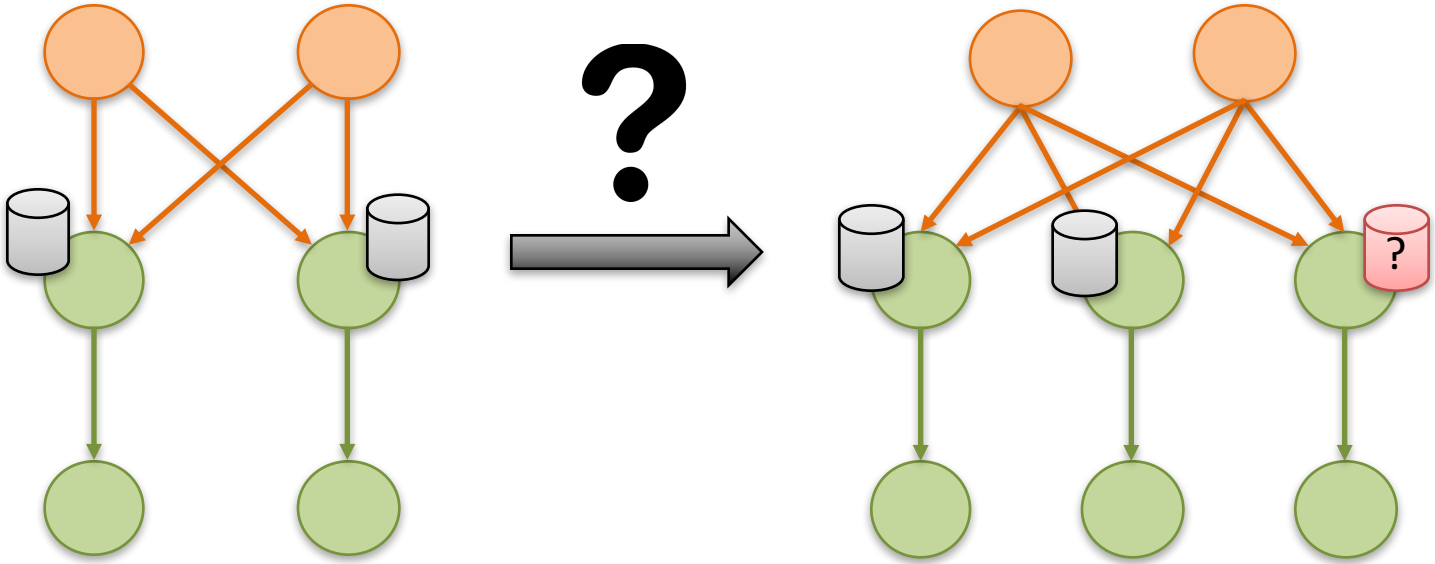
- Spread work across more workers to decrease workload

Scaling Stateless Jobs



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

Scaling Stateful Jobs

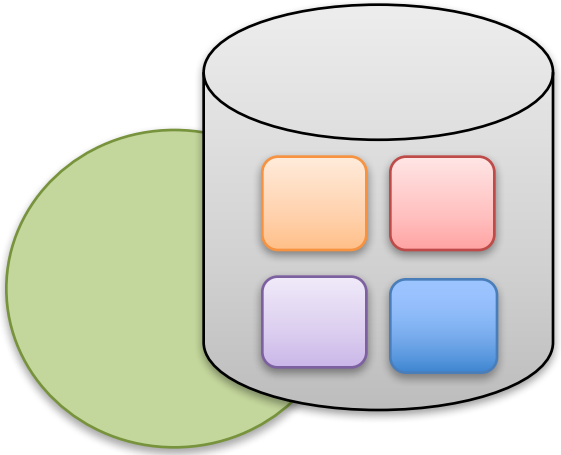


- Problem: Which state to assign to new task?

State in Apache Flink

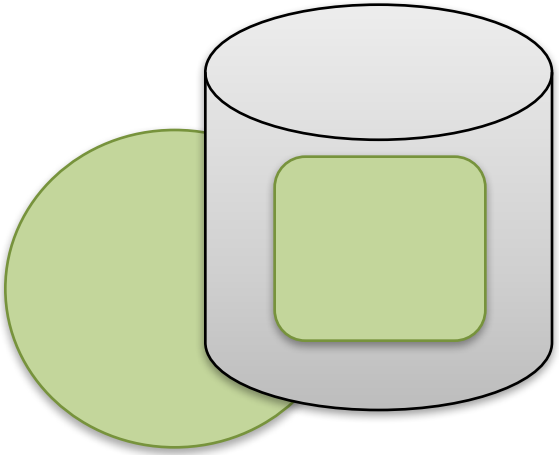
Keyed vs. Non-keyed State

Keyed



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

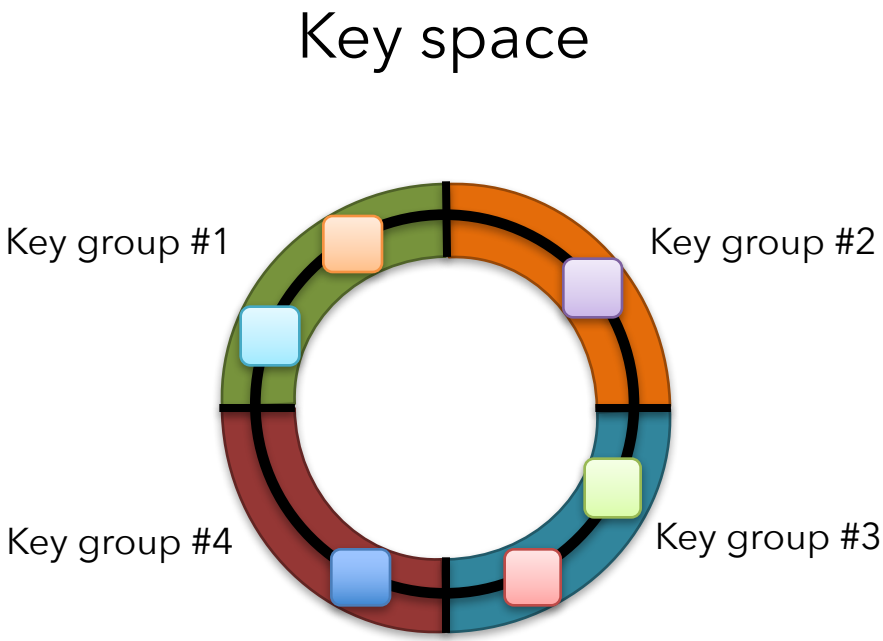
Non-keyed



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

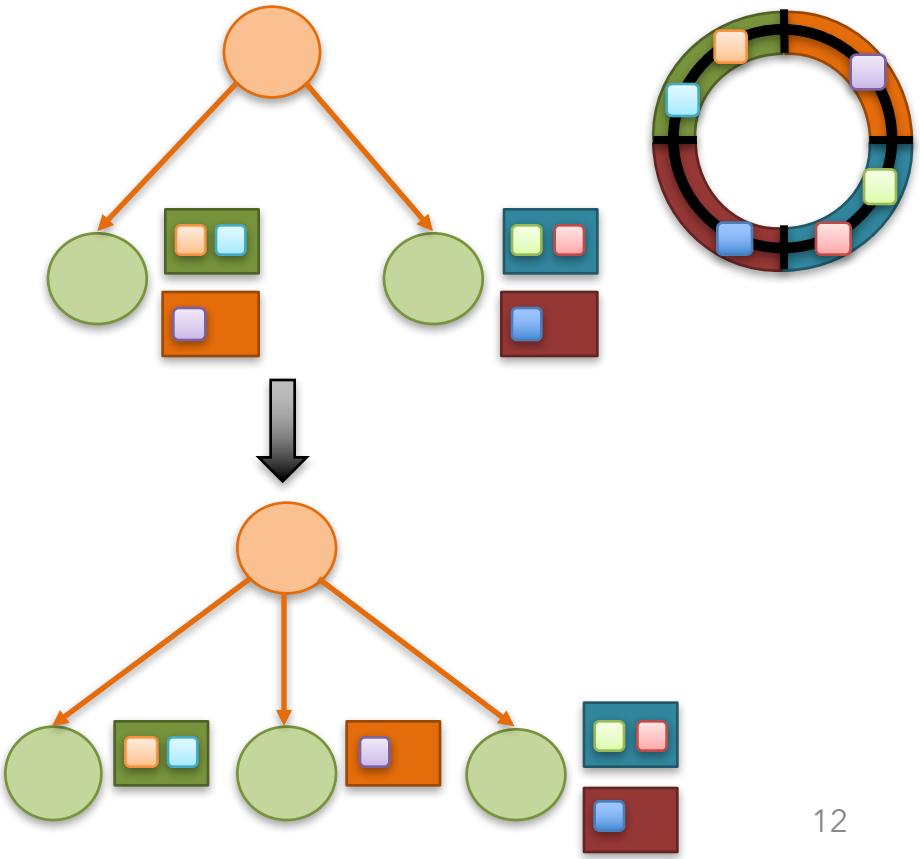
Repartitioning Keyed State

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



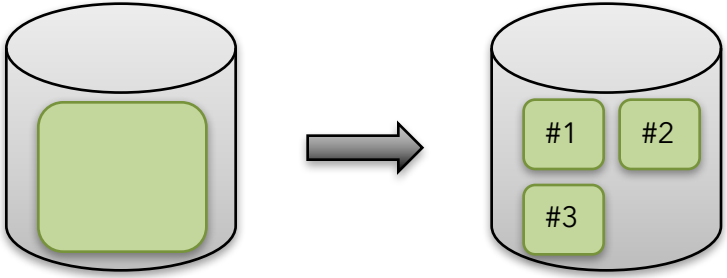
Repartitioning Keyed State contd.

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



Repartitioning Non-keyed state

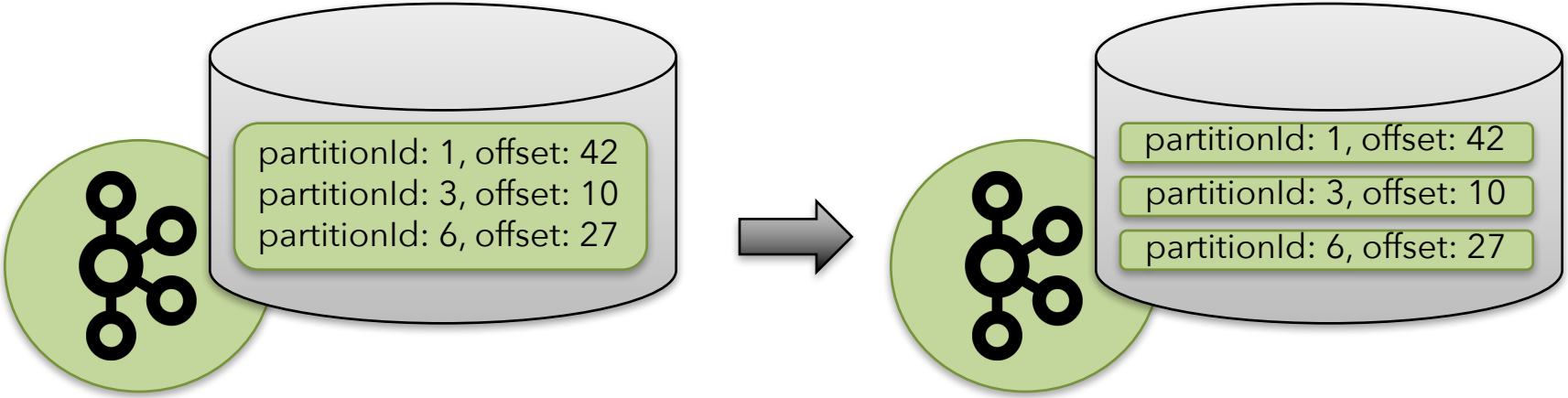
- User defined merge and split functions
  - Most general approach
- Breaking non-keyed state up into finer granularity
  - State has to contain multiple entries
  - Automatic repartitioning wrt granularity



Repartitioning Non-keyed State contd.

- Non-keyed state entries gathered at the job manager
- Repartitioning schemes
  - Repartition & send
  - Union & broadcast

Example: Kafka Source



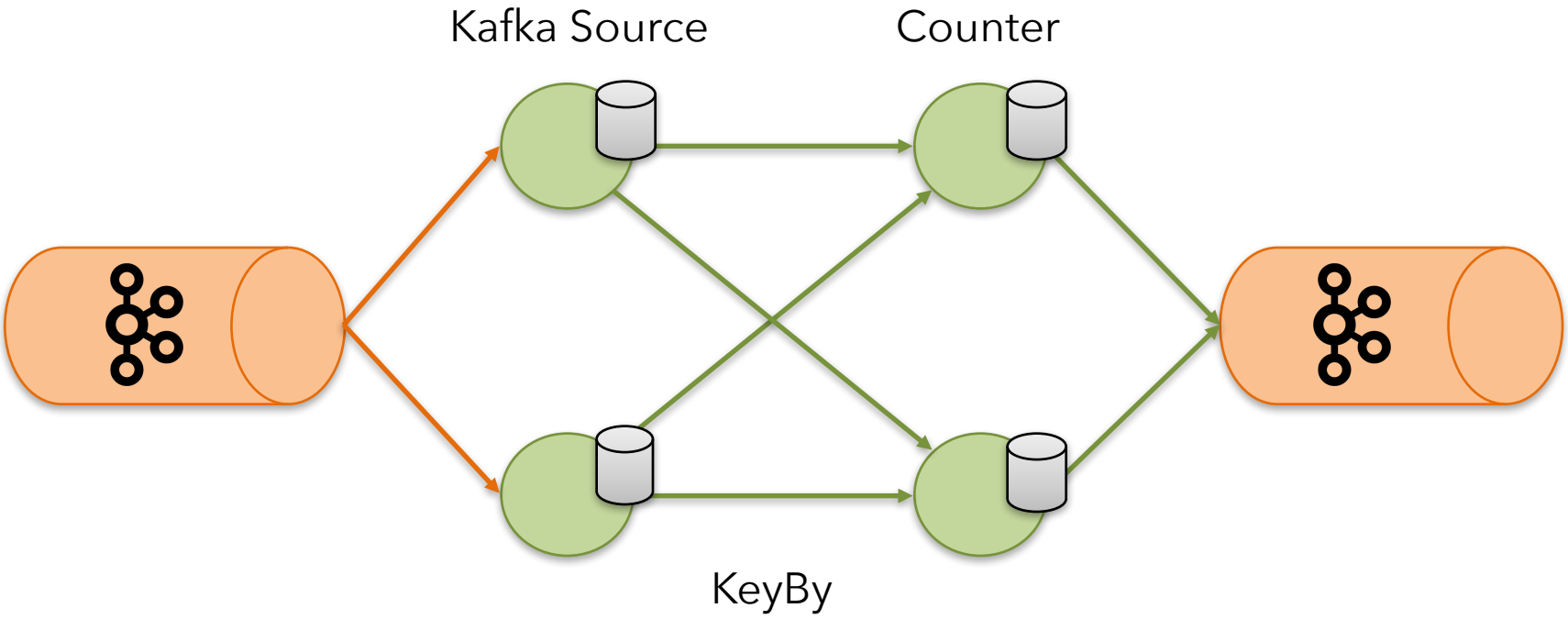
- Store offset for each partition
- Individual entries are repartitionable

Rescaling: Why is That so Hard?

- Handling of state
- Repartitioning of keyed & non-keyed state
- Unique among open source stream processors, afaik

Demo Time

Demo Topology



Current State and next Steps

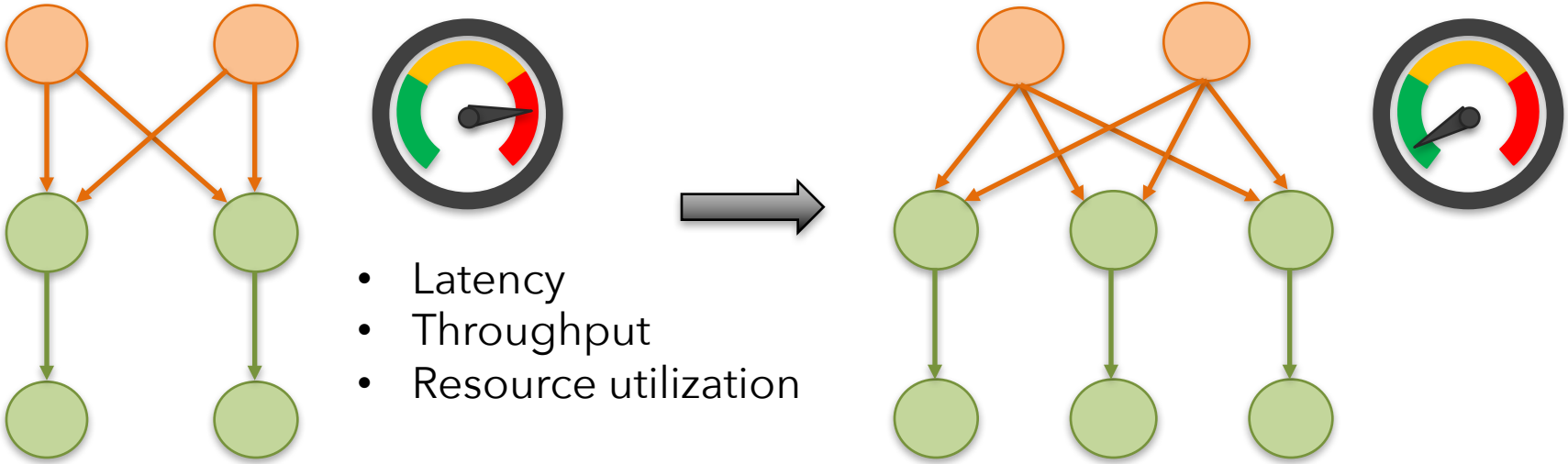
Current State

- Manual rescaling
  1. Take savepoint
  2. Stop the job
  3. Restart job with adjusted parallelism and savepoint

Next Steps

- Integrate savepoint with stop signal
- Rescaling individual operators w/o restart
- Dynamic container de-/allocation
  - *"Running Flink Everywhere" by Stephan Ewen, 16:45 at Kesselhaus*

Auto Scaling Policies



- Kubernetes on GCE, EC2 and Mesos (marathon-autoscale) already support auto-scaling

Conclusion

- Scaling of keyed and non-keyed state
- Flink supports manual rescaling with restart

  (WIP branch: https://github.com/tillrohrmann/flink/tree/partitionable-op-state)
- Future versions might support scaling on the fly and automatic rescaling policies