The External-Scheduler-Interface allows an external scheduler to control which cluster resources, Batch Applications like Apache Spark and Apache Flink will use for their TaskManager and Executor Pods. No Assumptions are made about the Driver and JobManager Pods. The Interface offers slots in the form of Testbeds to the External-Scheduler. The size (both in terms of the number of slots and resources) of Testbeds are controlled via the Testbeds manifest. Managing the Testbed is not exposed through the Interface, as they are not expected to be changed by an External-Scheduler, other than using its slots.

The concept of a Testbed allows the reservation of resources inside a cluster. It may be unreasonable to give an external scheduler all available resources in a multi-tenant cluster. The resources available to the external scheduler can be precisely controlled using Testbeds. Schedulings created by the external scheduler are always directed towards a Testbed. Currently, an active scheduling will claim its Testbed and prevent other Schedulings from using it.

The Scheduling resource creates a queue of Batch Jobs that will be run in the specified Testbed. Schedulings currently support a slot based strategy, where the external scheduler directly specifies which job should use which slot, and a queue based strategy, where only a queue of jobs is submitted and no specific slot is used.

Listing 1: Scheduling resource manifest with the queue based strategy

```
apiVersion: batchjob.gcr.io/v1alpha1
kind: Scheduling
metadata:
  name: profiler-scheduling-0
  namespace: default
spec:
  queueBased:
    - name: spark-crawler
      namespace: default
    - name: spark-crawler
      namespace: default
    - name: batchjob-spark
      namespace: default
    - name: batchjob-spark
      namespace: default
  slots:
    name: profiler-slots
    namespace: default
```

Naturally an external scheduler could interact with the Interface through the Kubernetes API by managing scheduling manifest. However, in case the external scheduler cannot directly access the Kubernetes API a thin HTTP layer is provided. The External-Scheduler-Interface offers endpoints for querying the current cluster situation in the form of the Testbeds slots occupation status and

the status of Schedulings or Batch Jobs. The Interface provides a REST API that allows the creation and deletion of Schedulings and the ability to update information stored inside the Batch Job manifest. An additional Stomp [**?**] (WebSocket) server is available to not enforce any polling for updates.

The functionality is demonstrated as part of the evaluation, either through the Manual Scheduler, which acts as a testing tool and as visualization, and the Example Scheduler, which uses multiple Testbeds to profile Batch Jobs for its scheduling decisions.