Enabling PerfKit Benchmarker for Apache Beam

Author: jasonkuster@google.com
Status: Final
Last Modified: Feb 2017

Overview

The Beam community has settled on PerfKit Benchmarker (PKB) as our tool for running Beam performance tests[1]. PerfKit Benchmarker has many of the utilities necessary for running Beam performance tests already (this was a strong factor in favor of its choice) but needs some modifications to enable greater functionality, increase ease of use, and support all the use cases we need to support.

Note: Most work items currently have placeholders for JIRA issues (tagged BEAM-XXXX) after them -- once we agree that this is the set of things we want to do, JIRA tickets will be created.

Goals

Enable Performance Tests On All Beam Runners

Beam runners require an instance of a data-processing backend which they can run on. PKB should support either bringing up the backend itself, or pointing to an already-running instance controlled by the user.

Support Beam IO Testing

As discussed in several email threads on the Beam list [2, 3, 4], we are in the process of solidifying how we test IO in Beam, both from a functionality perspective and in terms of performance. PKB already has a large amount of the framing in place to support the important aspects here: it has a provision phase and a tear-down phase which we can potentially hook to our IO provisioning system, and it has built-in utilities for capturing relevant metrics and publishing them to the performance UI.

Support Rich Beam- and Runner-Related Metrics

Benchmark wall time is the easiest metric to capture, but a mature set of performance benchmarks will capture more interesting data than wall time. Wall time is also affected by

cluster spin-up time, which in some cases can not be decoupled from actual job execution time. PerfKit has support for capturing arbitrary metrics from a benchmark run, and Beam has a metrics API[BEAM-147] that is approaching maturity. It is likely that these two systems will complement each other well. We will use the metrics API for capturing general Beam metrics, but will capture individual runner-related metrics as well to assist runner authors in tuning performance specific to their runner.

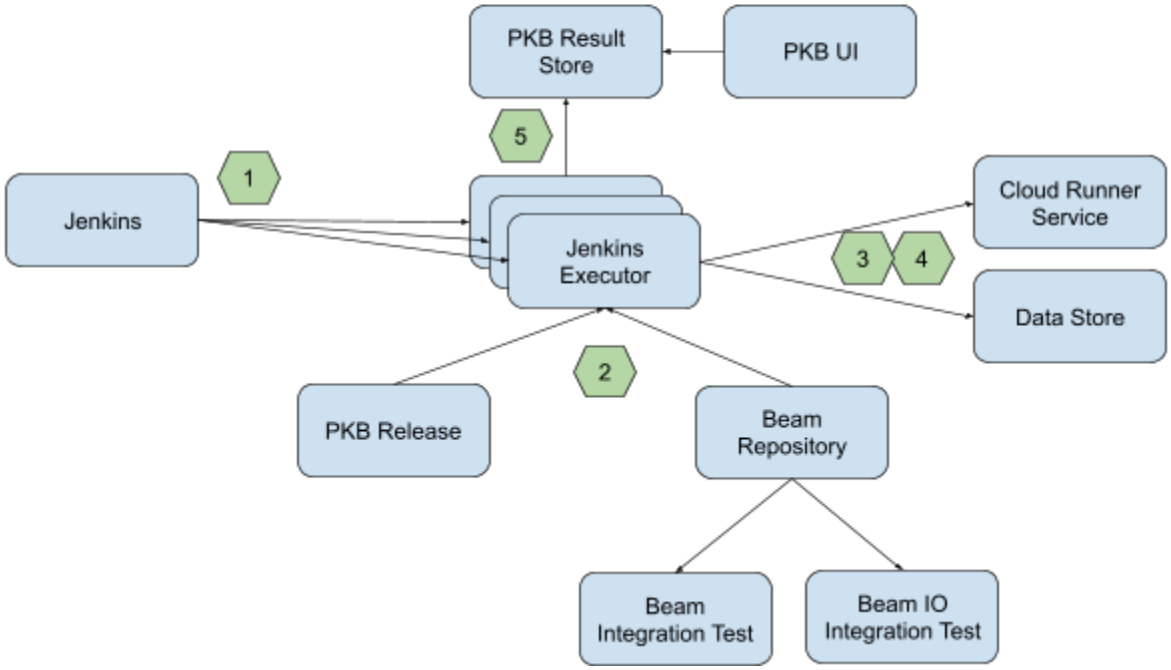
Easy To Start, Powerful With More Work

This goal has some similarities to other philosophies held in Beam. What this means is that, to start, it should be easy to write the initial version of a benchmark. There should be very few blockers in between a benchmark author and getting their benchmark to run against any runner with a set of sane defaults. This goal will be accomplished via the script and related infrastructure mentioned below.

On the other hand, creating a more sophisticated benchmark should be possible. In this case, the simple helpers which will be provided in Beam are not useful. PKB provides a large set of utilities for working with benchmarks, and the work below will expand those utilities to the point that a Beam benchmark author, with a bit more time, should be able to create a well-designed, repeatable, fair benchmark which provides useful information to the Beam community.

Usage

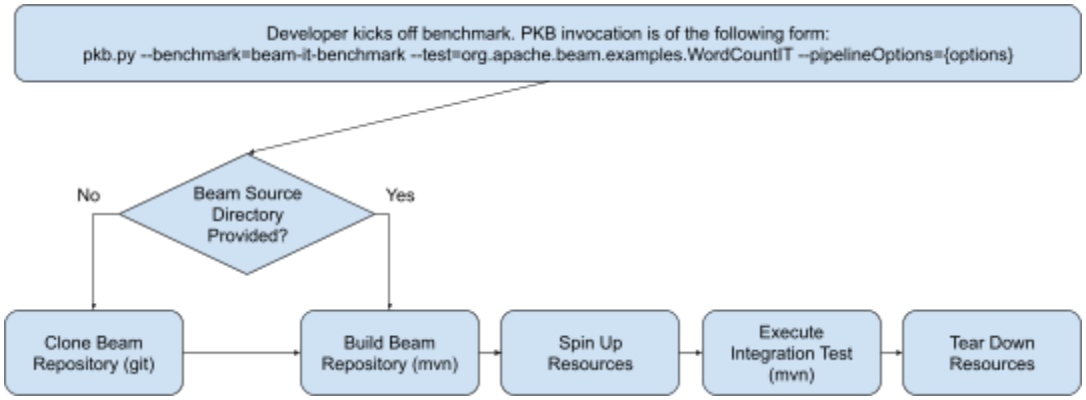
There are several components involved in this system. A diagram is provided below which describes how we envision these systems working together, with additional technical details below.



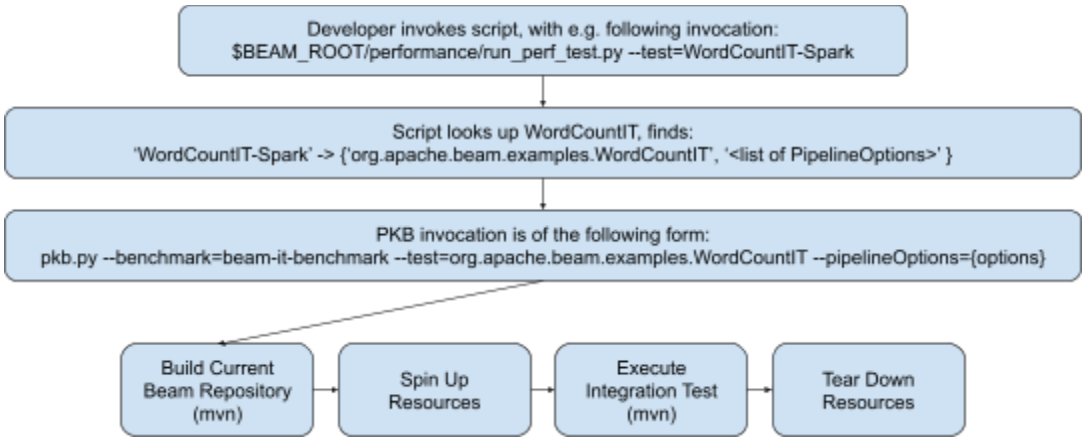
The overall idea here is to continue to use Jenkins and its executors as the workhorse for kicking off our test pipelines. We will use Beam's existing Integration Testing Framework for writing test pipelines, and will build support into PerfKit Benchmarker for running arbitrary Beam ITs. PKB and the IT running on the executor will be responsible for spinning up a runner as a service in the cloud, as well as any data stores on which the IT depends. PKB will then kick off the pipeline, which will run to completion, and metrics about the execution will be stored in a result store for eventual access by PKB's UI.

Beam IT -> PKB - BEAM-1595

Work has already started to create a PKB benchmark which can run arbitrary Beam Integration Tests. This benchmark uses Maven and the Failsafe plugin under the hood. Execution is diagrammed below.



For ease of use for Beam developers, we will create a performance testing directory with a configuration file which contains, essentially, a map of `IT Name` -> `IT PipelineOptions`. We will then write a helper script which, given the name of an IT, will compose the appropriate command line to run that IT with PKB. Modified diagram below.



PKB -> Jenkins - BEAM-1263

Once we have the ability to use PKB to kick off ITs, we can create a suite of Jenkins jobs which run different sets of performance tests in a continuous basis. These tests will publish official data to the PKB result store. Some example suites are:
JdbcIOPerf: JdbcTestRead, JdbcTestWrite
WordCountPerf: ApexWordCount, DataflowWordCount, FlinkWordCount, SparkWordCount

PKB UI - BEAM-1596

The PerfKit team also produces a UI (PerfKit Explorer) which knows how to read the results that PerfKit Benchmarker publishes to its result store. We will take up this UI in the Beam Testing project for use by the Beam community in understanding Beam's ongoing performance characteristics.

PerfKit Explorer is written as an App Engine app. Beam currently uses a GCP project, apache-beam-testing for its testing; we will create an App Engine site in this project for our instance of PerfKit Explorer. By default the website will be available at apache-beam-testing.appspot.com. PKB dumps its data to BigQuery; providing a reference to the table in our PerfKit Explorer dashboard will allow for automatic population of the data when new performance data becomes available.

Additional Work Items

Fair, Well-Understood Benchmarks - BEAM-1597

Design a comprehensive suite of benchmarks which provide fair comparisons between metrics and useful signal for regression testing. Provide all runner authors ample time to validate benchmarks before numbers are published.

Support For Beam IO Testing

PKB Enables Spin-Up/Spin-Down of Beam IO - BEAM-1598

Beam has settled on using Kubernetes for hosting IO data stores. PKB has basic support for Kubernetes; evaluate its capabilities and expand support to enable IO testing. This will involve passing the startup script as an argument to the benchmark and having PKB coordinate with a Kubernetes cluster. Given that our executors run on GCP, using Google Container Engine (hosted Kubernetes) will ensure that we don't need to handle the details of credentials or access; it will just work.

Notes:

- Loading data into the data stores for testing is another issue to tackle, but one which will not be dealt with here. How that is done does not block any of the work here and there are still open questions which prevent resolution of that problem.

All Runners Support Execution on PKB

- Apex - BEAM-1599
- Dataflow - BEAM-1600
- Flink - BEAM-1601
- Spark - BEAM-1602

Notes:

- PKB has the concept of Providers, i.e. cloud services which can be used for spinning up an instance of a data-processing system. It has support for Google Cloud Dataproc and Amazon Elastic MapReduce, which both have support for running YARN. Past that, both Dataproc and EMR have support for Spark out of the box, and EMR has support for Flink as well.
- Given that we will be running benchmarks via integration tests, it is important that all runners have the ability to kick off an IT against a cluster. As of this writing, Spark does not have this ability, tracked by BEAM-1603.
- Each of the non-Dataflow services listed above can be self-hosted. We should probably build a provider which doesn't spin up a service but acts as a passthrough for commands to either the Flink, Spark, or Apex runners. Tracked in BEAM-1604.

Beam Exports Metrics to PKB

- Runners
    - Apex - BEAM-1605
    - Dataflow - BEAM-1606
    - Flink - BEAM-1607
    - Spark - BEAM-1608
- Beam Metrics API - BEAM-1609

Notes:

- Beam Metrics API is currently tracked by BEAM-147.
- Runner metrics should include runner-specific measures which, while not necessarily able to be compared to those of other runners, are good to measure over time.