How to benchmark your Beam pipelines for performance and cost

Roy Arsan, Solutions Architect, Google Cloud
About me

- Cloud Solutions Architect @ Google Cloud
  - Team: Data Analytics, Solutions Engineering
  - Industries
    - Financial services, Healthcare & Life Sciences, Manufacturing & Retail
  - Use cases
    - Data movement, log analytics, security analytics
  - Products
    - BigQuery, Dataflow, Pub/Sub
Why benchmark thy pipeline

- Will my pipeline meet expected SLOs?
  - Job/event e2e latency, Event throughput (EPS)
- Will my pipeline operate with optimum performance/cost ratio
  - Maximize resource utilization
- Is my pipeline properly sized and configured?
Disclaimers

- Results presented are for demo purposes only.
  - No performance or cost guarantees.
  - Your mileage may vary.
    - Test your pipeline using your own real data and environment

- No pipeline harmed in this benchmarking process
Why benchmarking is so hard

- Heterogeneous data types, pipeline types, stages, cloud providers
- Selecting appropriate benchmark tests
- Configuring nontrivial environments
- Variety of Dataflow user-facing performance metrics
- Variability in performance due to number of moving parts
- Analyze data and get actionable insights
Benchmarking toolset
PerfKit Benchmarker
Open Source Benchmark Framework

- A canonical set of public benchmarks to measure and compare cloud offerings.
- PKB takes care all the work from provisioning, executing, cleaning up and publishing results through CLI.
- All benchmarks are running with default settings. (Not tuned to in favor of any providers).
- Supported providers: GCP, AWS, Azure, Digital Ocean, Rackspace, OpenStack
- Contribution from Stanford, MIT, CloudHarmony, CloudSpectator, etc.
Example 1: Wordcount

- Apache Beam \texttt{WordCount example}
- PKB \texttt{wordcount benchmark}
- Running on different machine types

```java
flags: &myflags
dpb_service: service_type: dataflow, worker_count: 1, worker_group: *eight_core 
  flags: *myflags

benchmarks:
  - dpb_wordcount_benchmark: {}
    dpb_service: service_type: dataflow, worker_count: 1, worker_group: *four_core 
      flags: *myflags
  - dpb_wordcount_benchmark: {}
    dpb_service: service_type: dataflow, worker_count: 1, worker_group: *two_core 
      flags: *myflags
```
Demo
Example 1: Wordcount perf results
Benchmark Example 2

Custom or Google-provided Dataflow template
Example 2: Dataflow Template
Pub/Sub Subscription to BigQuery

Orchestrator state machine:
1. Deploy test bed and seed input data
2. Restore Pub/Sub subscription snapshot
3. Run pipeline with configuration A
4. Wait until all messages are processed
5. Collect stats into BQ: pipeline utilization, cost, latency, and throughput
6. Repeat steps 2-5 for N different pipeline configurations
7. Synthesize all tests results into optimal unit worker size (for perf/$) and horizontal scaling guideline for pipeline to meet SLO
Example 2: Dataflow Template

- Dataflow template: Pub/Sub Subscription to BigQuery
- PKB dataflow template benchmark
- Run across different recent versions

```python
dpb_df_template_benchmark:
description: Run Dataflow template
dpb_service:
  service_type: dataflow
  worker_count: 1
  worker_group: *four_core
flag_matrix: recent_versions
flag_matrix_defs:
  recent_versions:
    dpb_df_template_gcs_location:
    - gs://dataflow-templates,2022-07-04-00_RC00/PubSub_Subscription_to_BigQuery
    - gs://dataflow-templates,2022-06-27-00_RC00/PubSub_Subscription_to_BigQuery
    - gs://dataflow-templates,2022-06-21-00_RC00/PubSub_Subscription_to_BigQuery
    - gs://dataflow-templates,2022-06-06-00_RC00/PubSub_Subscription_to_BigQuery
    - gs://dataflow-templates,2022-05-30-00_RC00/PubSub_Subscription_to_BigQuery
```
Example 2: Dataflow Template

- Dataflow template: Pub/Sub Subscription to BigQuery
- PKB dataflow template benchmark
- Run across different input sizes
Example 2: Pipeline perf results

<table>
<thead>
<tr>
<th>vCPUs*</th>
<th>Mem (GB)*</th>
<th>Max egress (Gbps)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1-standard-2</td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>n1-standard-4</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>n1-standard-8</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>n1-standard-16</td>
<td>16</td>
<td>60</td>
</tr>
</tbody>
</table>

*See Compute Engine docs for latest specs and factors
Example 2: Utilization & cost results

Average CPU Utilization (%)

<table>
<thead>
<tr>
<th>Instance</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1-standard-16</td>
<td>6.18</td>
</tr>
<tr>
<td>n1-standard-8</td>
<td>12.36</td>
</tr>
<tr>
<td>n1-standard-4</td>
<td>29.89</td>
</tr>
<tr>
<td>n1-standard-2</td>
<td>46.47</td>
</tr>
</tbody>
</table>

Total Cost

<table>
<thead>
<tr>
<th>Instance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1-standard-2</td>
<td>0.027</td>
</tr>
<tr>
<td>n1-standard-4</td>
<td>0.039</td>
</tr>
<tr>
<td>n1-standard-8</td>
<td>0.067</td>
</tr>
<tr>
<td>n1-standard-16</td>
<td>0.136</td>
</tr>
</tbody>
</table>
Caveats and next steps

1. Variability in pipeline performance
   ○ Run time, spin up/down time, failures
     ■ Iterate each test configuration N times
     ■ Pre-warm pipeline before running each test
     ■ Mark test complete when all events written to sink

2. Dependency on testbed setup, and source/sink characteristics
   ○ Networking, Cloud NAT
   ○ Limited to upstream and downstream performance
     ■ e.g. BQ write throughput quota per project
Caveats and next steps

3. Sensitivity to input workload

- Realistic benchmark and stress testing
  - Backlog
  - Steady-state
  - Steady-state with bursts
  - Steady-state with step function
- Realistic data sets
  - Run different types of data
Your mileage may vary…

based on your data, environment and workload profile
Call to Action

- **Benchmark** your own pipelines
  - Estimate costs, plan capacity to meet SLOs, and avoid performance regressions
- **Use** battle-tested PerfKit Benchmarker
- **Share** feedback and suggestions
- **Extend** PerfKit Benchmarker for benefit of Dataflow & Beam community

Wiki: [https://googlecloudplatform.github.io/PerfKitBenchmarker/](https://googlecloudplatform.github.io/PerfKitBenchmarker/)
GitHub: [https://github.com/rarsan/PerfKitBenchmarker/](https://github.com/rarsan/PerfKitBenchmarker/) (to be merged upstream)
Acknowledgements

- **Diego Orellana**, Software Engineer @ Google, PerfKit Benchmarker
- **Sergei Lilichenko**, Cloud Solutions Architect @ Google, Data Analytics
- **Rodd Zurcher**, Cloud Solutions Architect @ Google, App/Infra Modernization
Questions?

twitter.com/RoyArsan
linkedin.com/in/arsan
github.com/rarsan