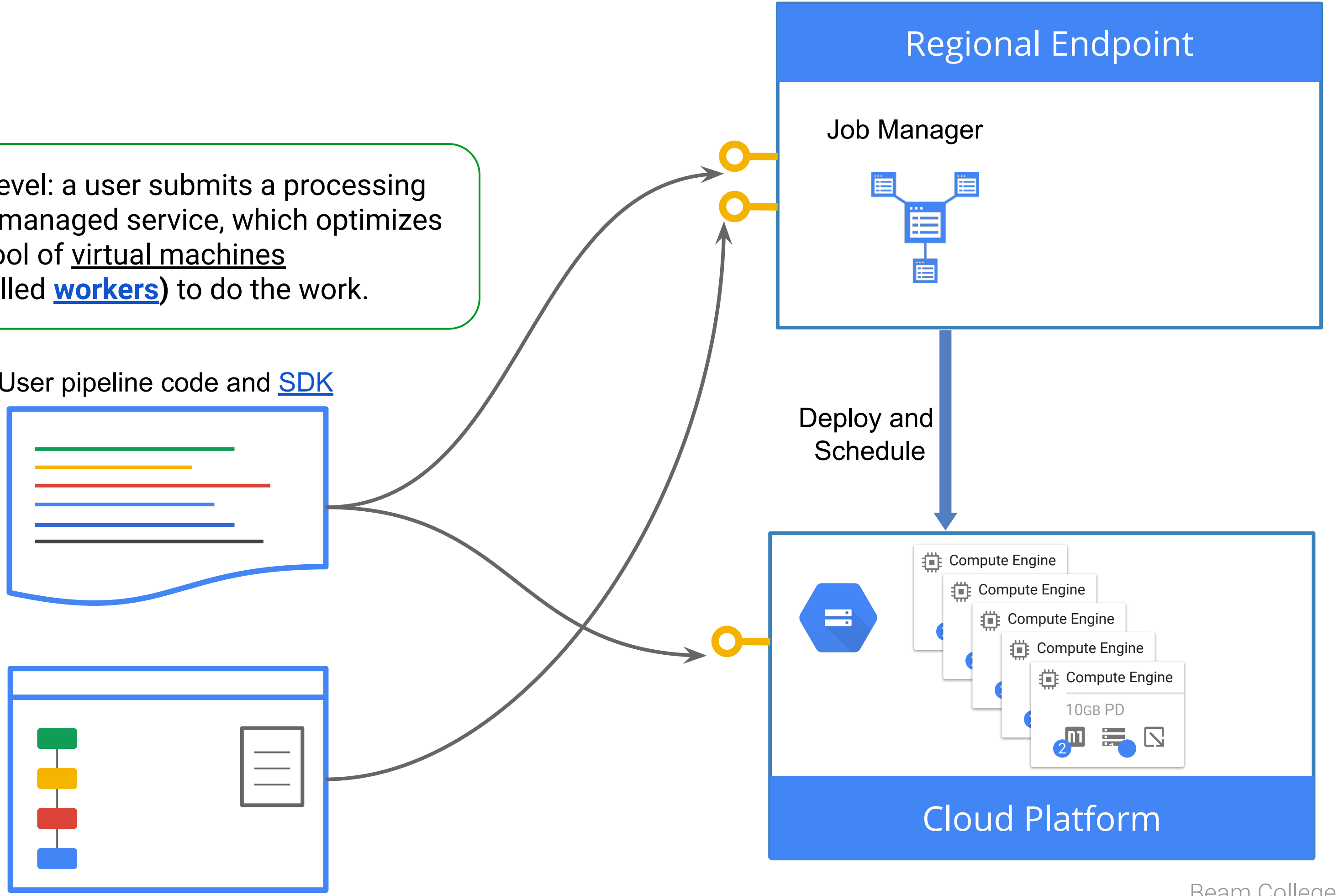
Configuring your Dataflow job

Israel Herraiz

Strategic Cloud Engineer, Google
http://twitter.com/herraiz

The Dataflow runner

At a very high level: a user submits a processing pipeline to our managed service, which optimizes it and runs a pool of virtual machines (sometimes called **workers**) to do the work.



Dataflow & Compute Engine

Region endpoint

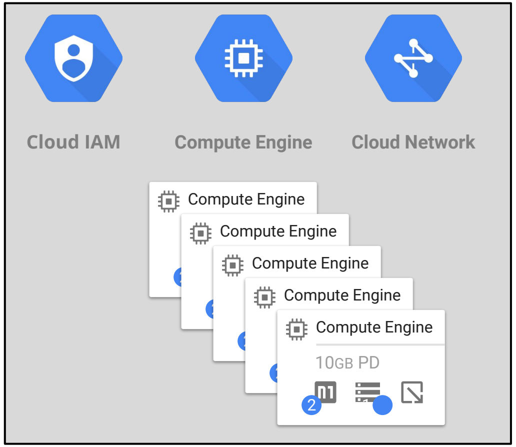
- Deploys and controls Dataflow workers and stores Job Metadata
- Region is **us-central1** by default, unless explicitly set using the region parameter

Zone

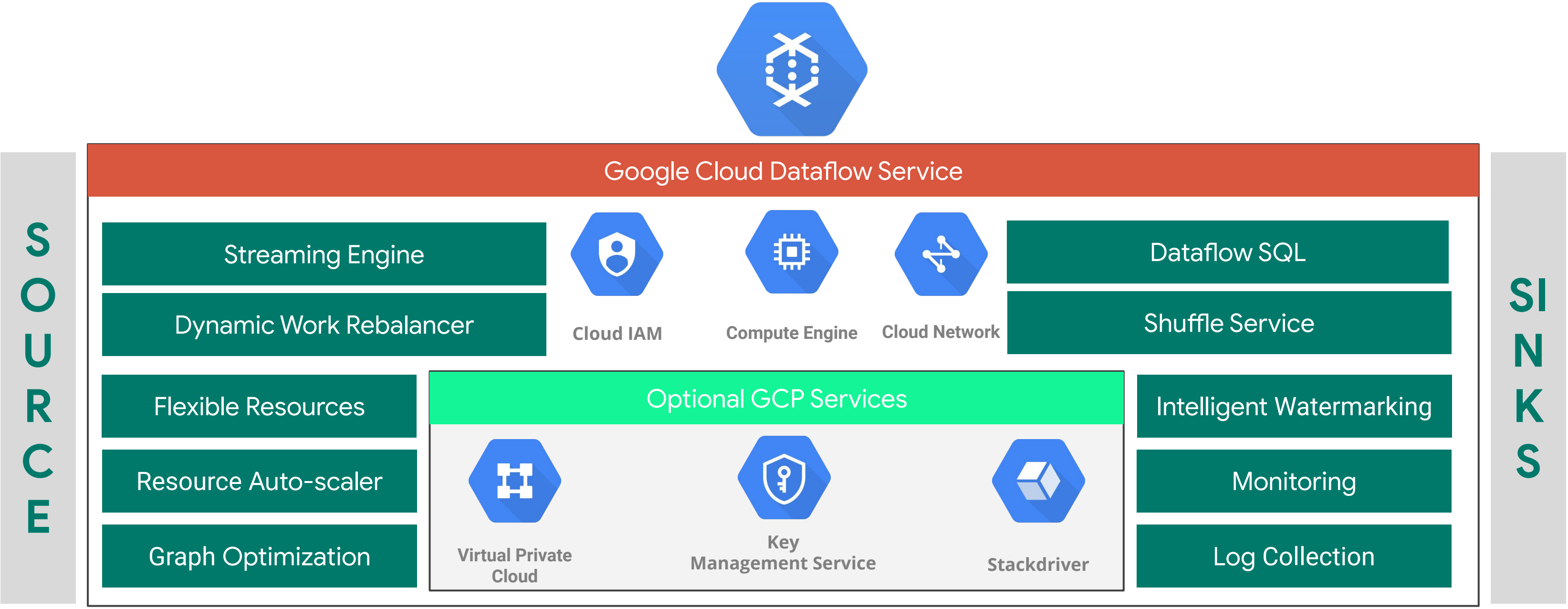
- Defines the locations of the Dataflow workers
- Defaults to a zone in the region selected based on available zone capacity. It can be overridden using the zone parameter.

The zone does not need to be in the same region as the endpoint. Reasons you may want to do this include:

- Security and Compliance
- Data locality
- Resilience and geographic separation



Google Cloud Dataflow Service



SOURCE

Google Cloud Dataflow Service

Streaming Engine

Dynamic Work Rebalancer

Cloud IAM

Compute Engine

Cloud Network

Dataflow SQL

Shuffle Service

Flexible Resources

Resource Auto-scaler

Graph Optimization

Optional GCP Services

Virtual Private Cloud

Key Management Service

Stackdriver

Intelligent Watermarking

Monitoring

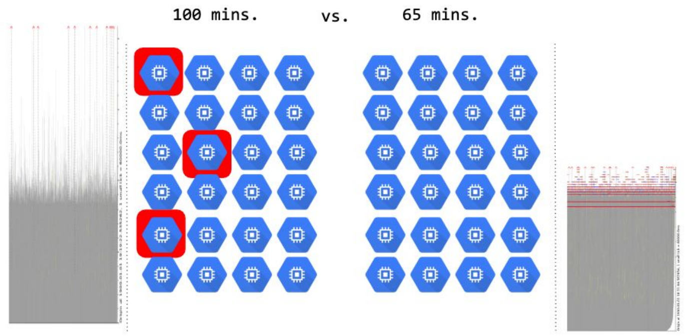
Log Collection

SINKS

Dataflow features

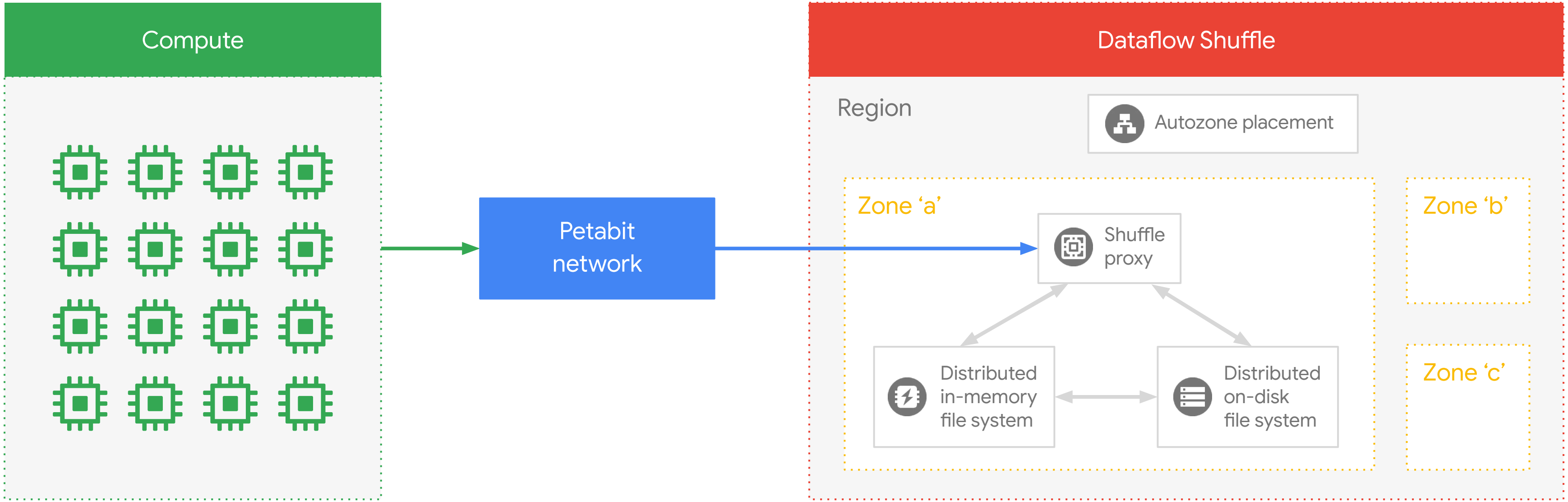
Batch Dynamic Work Redistribution

- Redistribute hot keys for more even workload distribution.
- Fully automated



https://cloud.google.com/dataflow/docs/guides/deploying-a-pipeline#autotuning-features

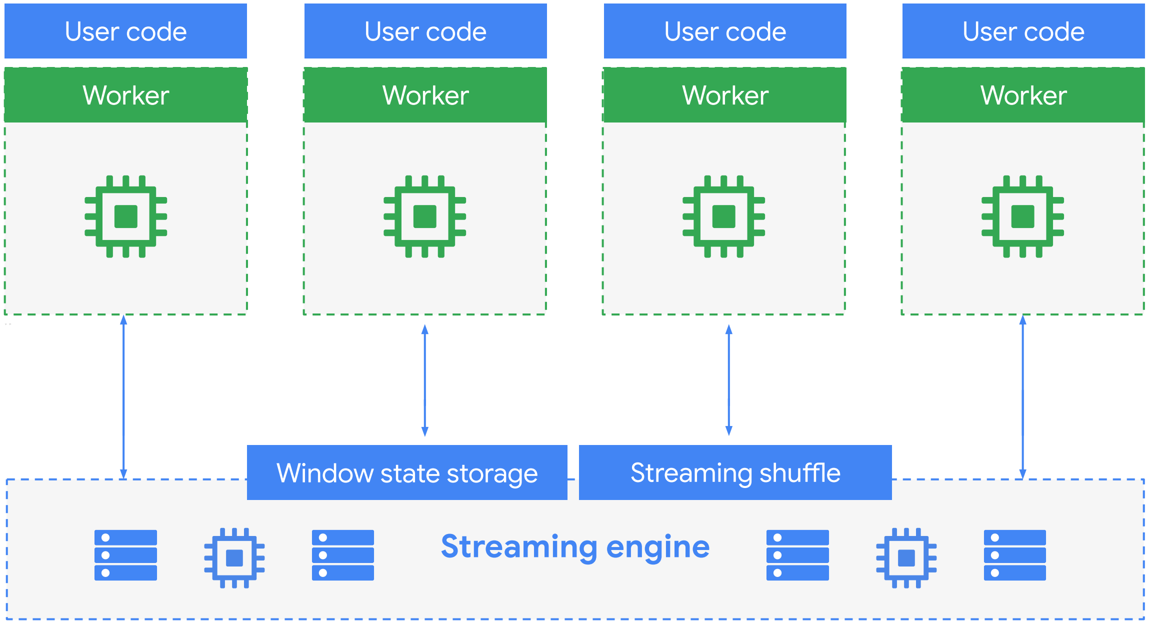
Dataflow Shuffle Service - Batch



Dataflow Streaming Engine

Benefits

- ✓ Smoother autoscaling
- ✓ Better supportability
- ✓ Less worker resources

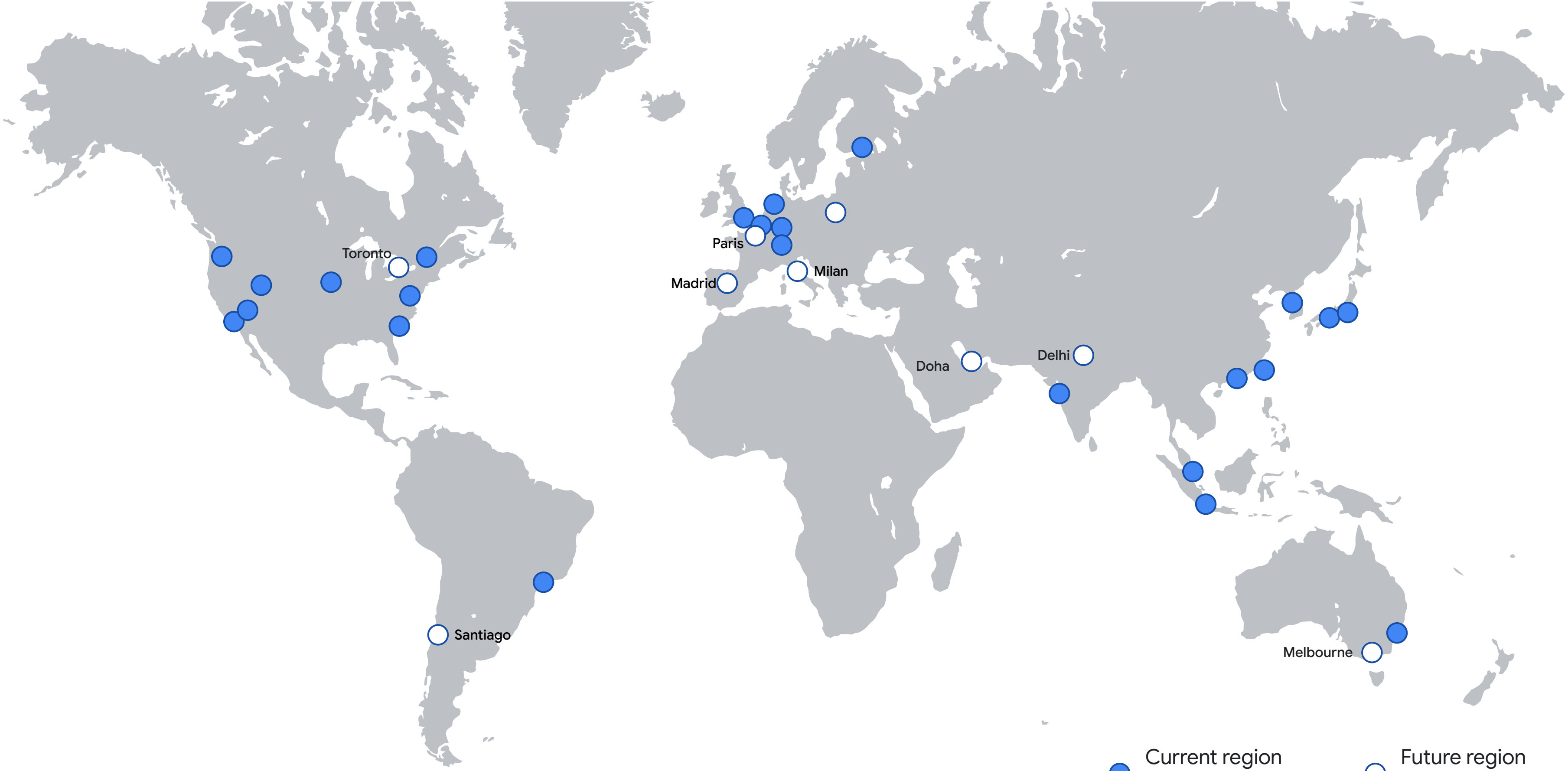


Configuring your Dataflow pipeline

Choosing the right parameters for your pipeline

- Region
- Autoscaling
- Worker type and size
- Public IPs
- FlexRS for batch pipelines
- Shuffle mode for batch pipelines
- Streaming Engine for streaming pipelines

Region



● Current region with 3 zones

○ Future region with 3 zones

Choosing the region

- You can run Dataflow in any region of GCP
  - https://cloud.google.com/dataflow/docs/resources/locations
- Run in the same region as where your data is
  - Choose your temp location in a GCS bucket in the same region

Autoscaling

Autoscaling

- Normally preferred
- However, for very small jobs, you may save some procesing time by setting a fixed number of workers
- Dataflow autoscales depending on the CPU usage and the backlog size
  - Set max_num_workers to limit the maximum number of workers created by autoscaling

Autoscaling

- Parameter ***autoscalingAlgorithm*** controls autoscale, options available
  - None
  - THROUGHPUT_BASED
- Enabled by default on all batch jobs
- Enabled by default on streaming jobs using the streaming engine
  - Streaming Engine is the default mode for Python streaming pipelines using Beam >= 2.21
- When disabled the number of workers defaults to 3 or the value of numWorkers if set.
- When enabled maxNumWorkers allows the maximum number of workers to be defined**.
- When enabled setting numWorkers will set the initial number of workers the pipeline will start with.

Worker features

Public IPs

- Disable. Only required if you need access to the Internet.
- If using any other GCP service (GCS, BigQuery), enable Private Google Access in the VPC in the region where you are running
  - https://cloud.google.com/vpc/docs/configure-private-google-access
- For Java dependencies, use fat JARs
- For Python dependencies, explore using custom containers
  - https://cloud.google.com/dataflow/docs/guides/using-custom-containers

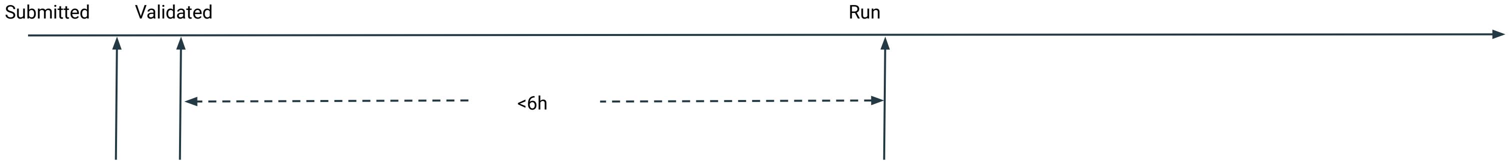
CPU, memory and disk size

- N2 machines can provide better CPU performance.
- Having OOM problems? Explore using highmem workers.
- Disk size: no need to normally change the default values
    - If using a lot of disk, better consider using shuffle mode (for batch) or streaming engine (for streaming)

Using preemptible workers (FlexRS)

FlexRS - Cost Effective Batch Processing

- https://cloud.google.com/dataflow/docs/guides/flexrs
- Delayed Scheduling + Shuffle Service + Preemptible VM
    - Jobs are validated and queued for up to 6 hours
    - Certain percentage of preemptible VMs are used.
    - ~60% CPU/Mem price
- FlexRS reduces batch processing costs by using advanced scheduling techniques, the Cloud Dataflow Shuffle service, and a combination of preemptible virtual machine (VM) instances and regular VMs.
- Jobs with FlexRS use service-based Cloud Dataflow Shuffle for joining and grouping.



Shuffle mode for batch pipelines

Shuffle mode (batch pipelines)

- Recommended for *complex* pipelines
    - Using GroupByKey, CoGroupByKey or Combine
- Shuffling is delegated to Google infrastructure
- Additional charge per gigabyte processed in the shuffle infra
- Overall cost normally better because running time is lower
    - You may also save by consuming less working resources
        - Smaller disks (default is 25 GB for shuffle mode and 250 GB for non-shuffle mode)
    - But if your pipeline is simple, the overall cost may be higher
    - In doubt? Experiment!

Shuffle mode and FlexRS

- FlexRS forces the use of shuffle mode
- Even if your pipeline is simple, using shuffle mode will not imply additional costs
    - Great savings thanks to the use of preemptible workers
- Temporary results are stored in the Dataflow Shuffle service
    - Your data is safe even if your workers are pre-empted by Compute Engine

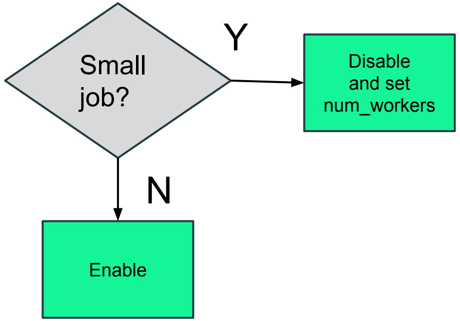
Streaming engine

Streaming engine (streaming pipelines)

- Offload shuffling calculations to the Dataflow service
- More responsive to autoscaling in reaction to variations of data volumes
- Less consumption of worker resources
  - Smaller workers
  - Less processing time
- Pay per GB of data processed in the Dataflow service
  - For complex pipelines, cost normally will be lower or similar to not using Streaming Engine
    - And processing time will be much lower
  - For very simple pipelines, cost can be higher with little gain in performance
- Default mode for Python streaming pipelines using Apache Beam >= 2.21

Summary of decisions

Autoscaling



Small job?

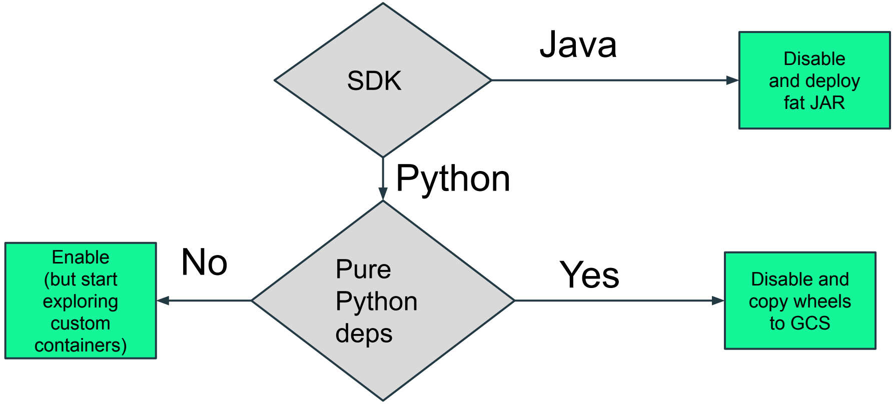
Y

Disable and set num_workers

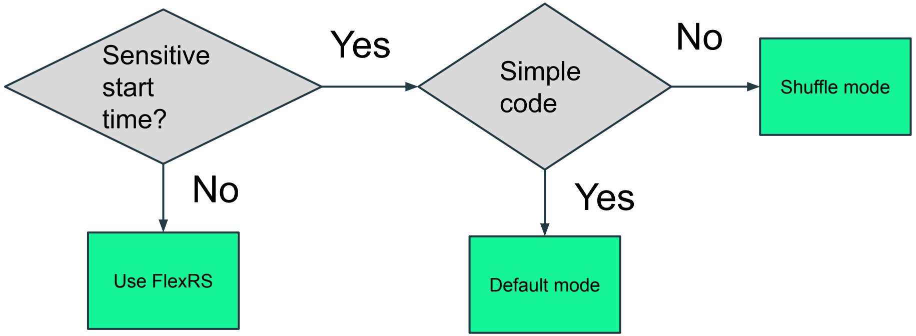
N

Enable

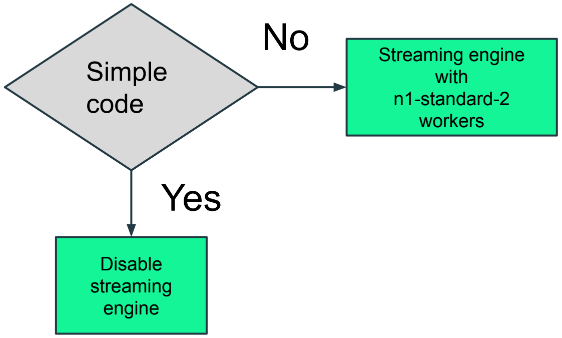
Public IPs


SDK
Java
Disable and deploy fat JAR
Python
Pure Python deps
Yes
Disable and copy wheels to GCS
No
Enable (but start exploring custom containers)

Batch pipeline



Streaming pipeline



Simple code

No

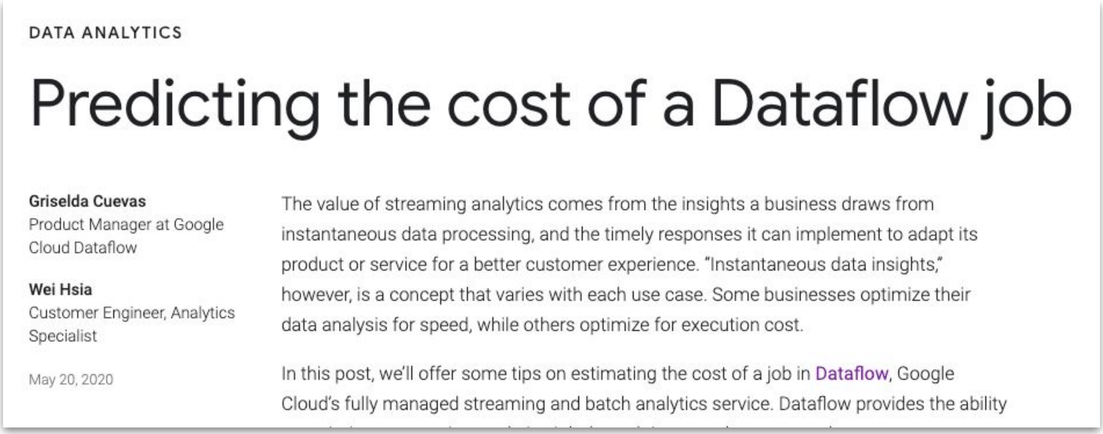
Streaming engine with n1-standard-2 workers

Yes

Disable streaming engine

In doubt about what options to chose?

Experiment and extrapolate



DATA ANALYTICS

Predicting the cost of a Dataflow job

Griselda Cuevas
Product Manager at Google Cloud Dataflow

Wei Hsia
Customer Engineer, Analytics Specialist

May 20, 2020

The value of streaming analytics comes from the insights a business draws from instantaneous data processing, and the timely responses it can implement to adapt its product or service for a better customer experience. "Instantaneous data insights," however, is a concept that varies with each use case. Some businesses optimize their data analysis for speed, while others optimize for execution cost.

In this post, we'll offer some tips on estimating the cost of a job in Dataflow, Google Cloud's fully managed streaming and batch analytics service. Dataflow provides the ability

https://cloud.google.com/blog/products/data-analytics/predicting-cost-dataflow-job

Thank you!

Questions?

BEAM
COLLEGE