BEAM SUMMIT

Scaling machine learning to millions of users with Apache Beam

Tatiana Al-Chueyr
Principal Data Engineer @ BBC Datalab

Online, 4 August 2021

BBC

@tati_alchueyr

- **Brazilian** living in London **UK** since 2014
- Principal Data **Engineer** at the **BBC** (Datalab team)
- Graduated in **Computer Engineering** at Unicamp
- Software developer for **18 years**
- **Passionate** about **open-source**

Apache Beam user since **early 2019**

BBC.datalab.hummingbirds

The knowledge in this presentation is the result of lots of **teamwork** within **one squad** of a **larger team** and even **broader organisation**

Darren Mundy

Richard Bownes

Tatiana Al-Chueyr

current squad team members

previous squad team members

Bettina Hermant

David Hollands

Jana Eggink

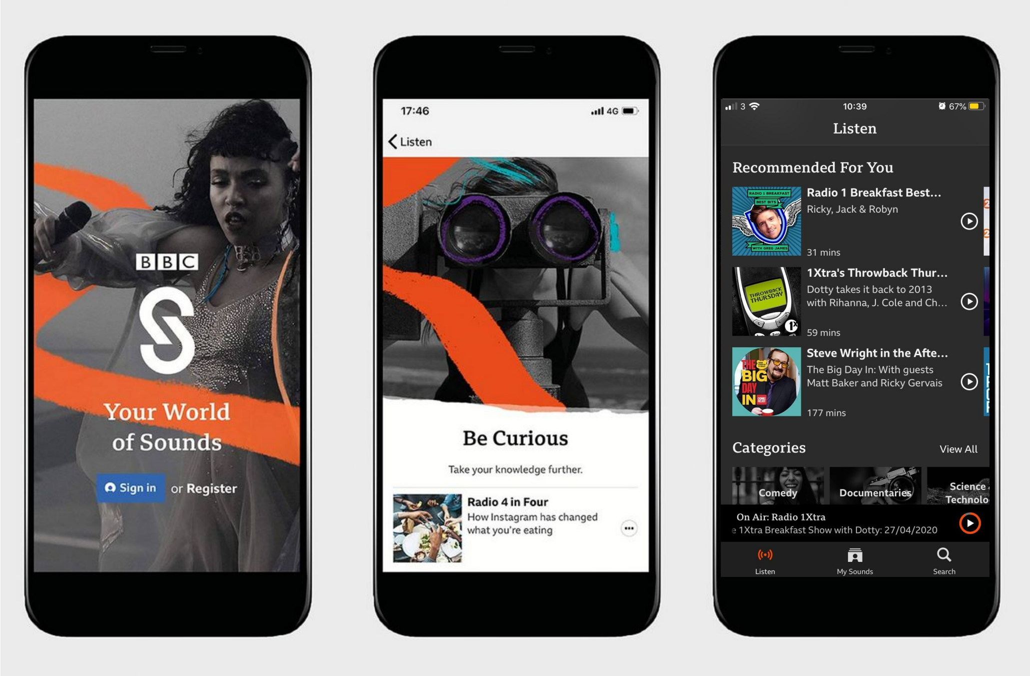
Marc Oppenheimer

some

business context

business context goal

to build a **replacement** for an **external third-party** recommendation engine



to **personalise** the experience of **millions of users** of **BBC Sounds**

business context numbers

BBC Sounds has approximately

- **200,000** podcast and music **episodes**
- **6.5 millions** of **users**

The **personalised** rails (eg. **Recommended for You**) display:

- **9 episodes** (smartphones) or
- **12 episodes** (web)

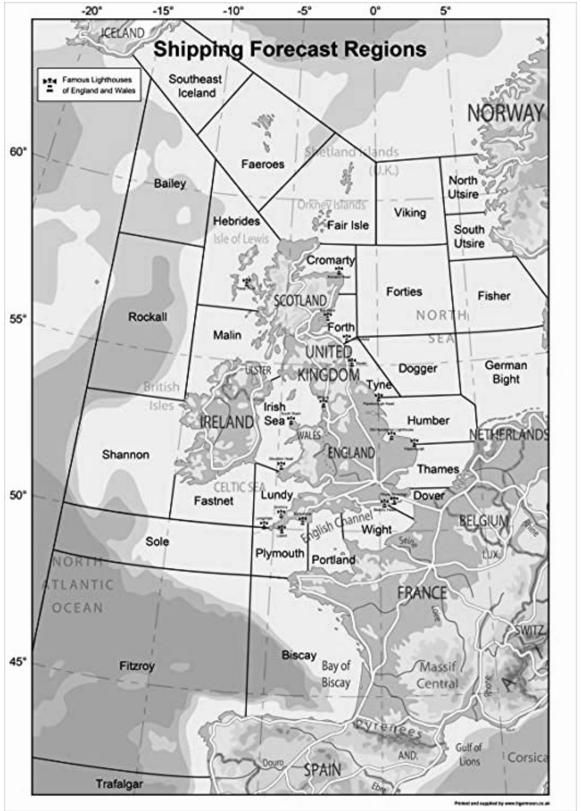
business context problem visualisation

it is similar to finding the **best match** among **20,000 items** per user x **65 million** **times**

business context product rules

The **recommendations** must also comply to the BBC product and editorial **rules, such as:**

- **Diversification:** no more than **one** item per **brand**
- **Recency:** no **news** episodes **older** than **24 hours**
- **Narrative arc:** next **drama series** episode
- **Language: Gaelic** items to **Gaelic listeners**
- **Availability:** only **available** content
- **Exclusion**: **shipping forecast** and **soap-opera**



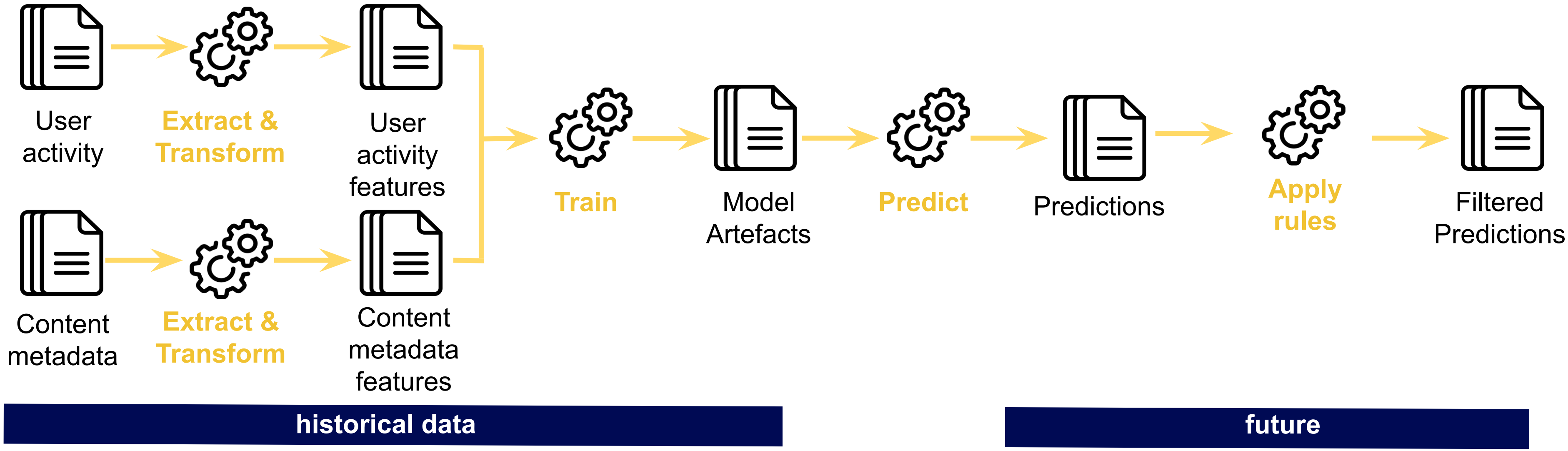
technology & architecture

overview

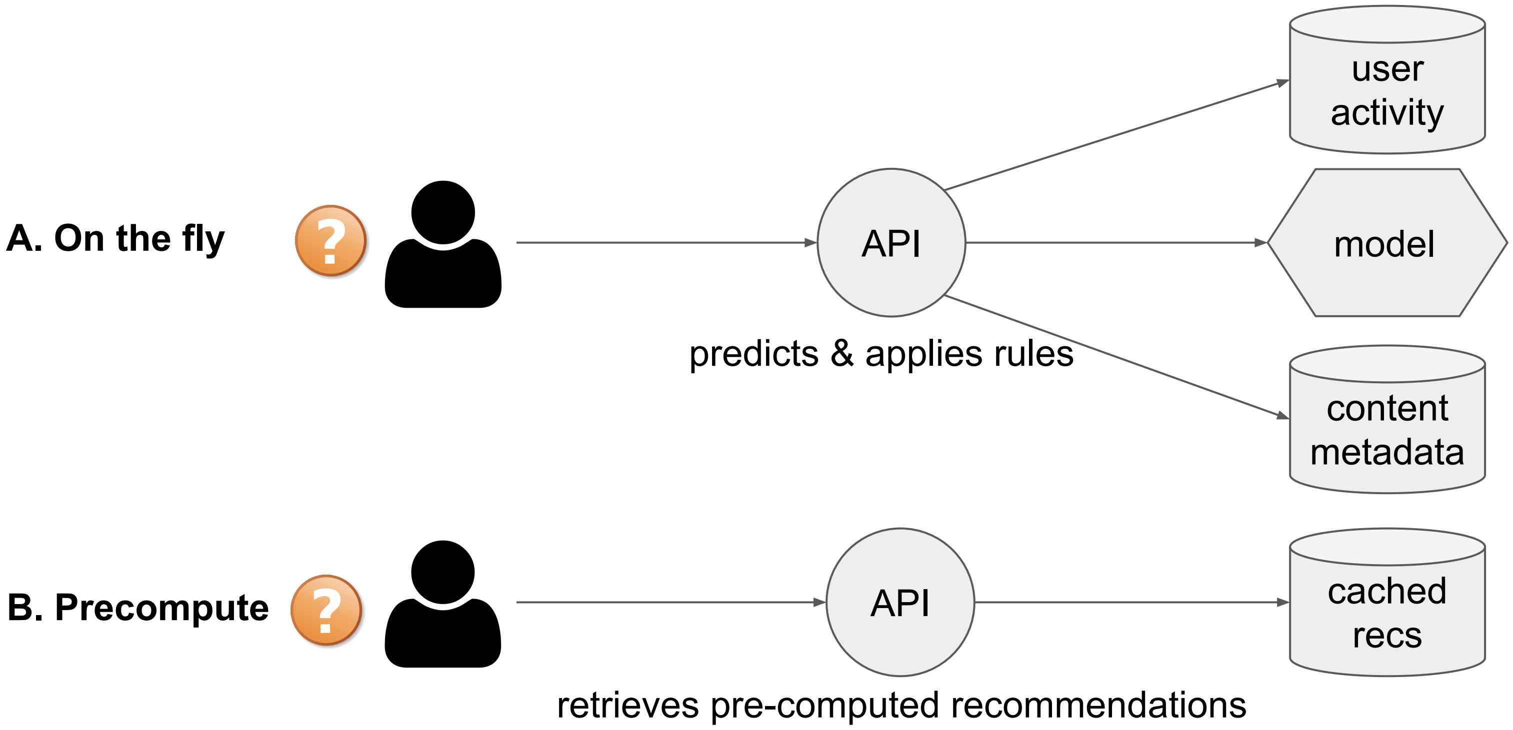
technology overview

- Python
- Google Cloud Platform
- Apache Airflow
- Apache Beam (Dataflow Runner)
- LightFM Factorisation Machine model

architecture overview



risk **analysis** predict on the fly



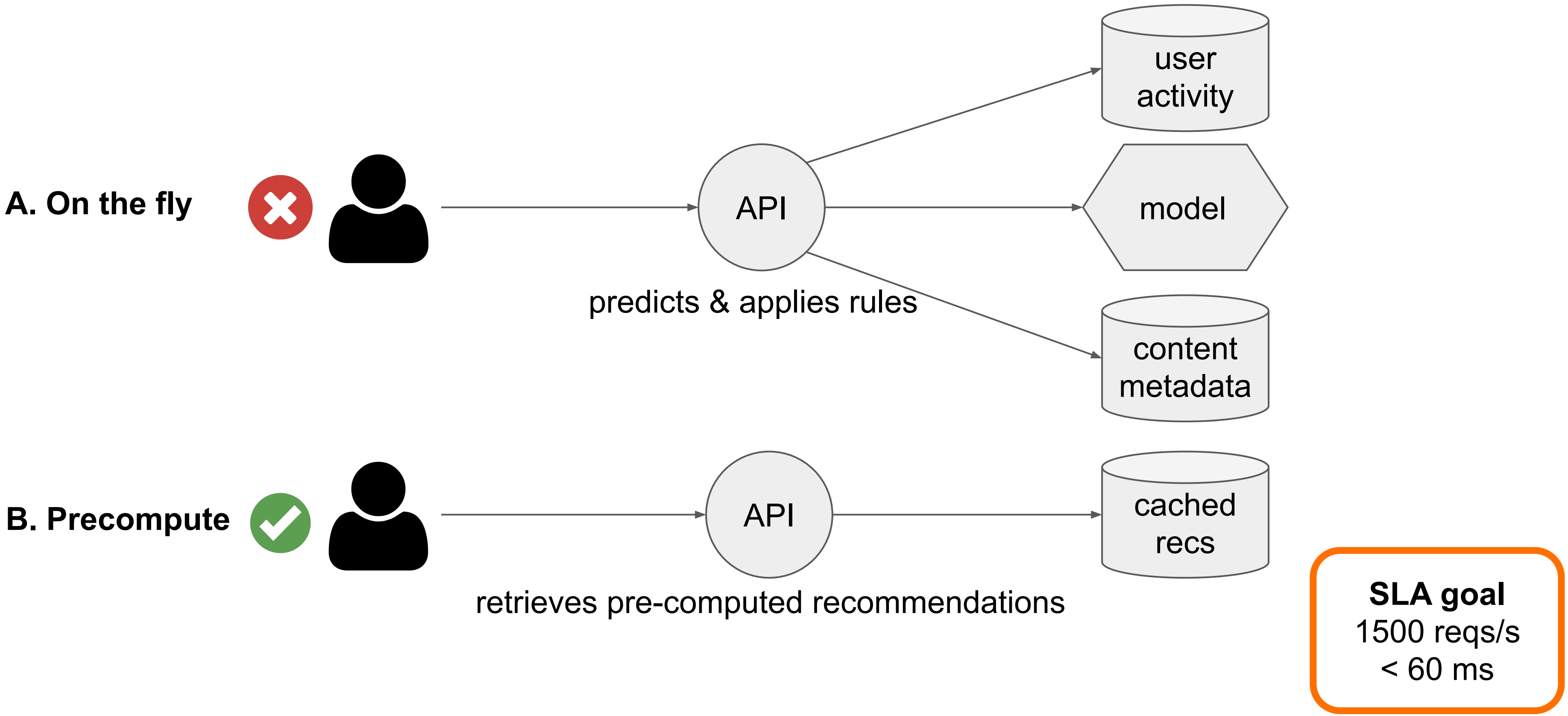
SLA goal
1500 reqs/s
< 60 ms

risk **analysis** predict on the fly

	On the fly	Precomputed	Precomputed
Concurrent load tests requests/s	50	50	1500
Success percentage	63.88%	100%	100%
Latency of p50 (success)	323.78 ms	1.68 ms	4.75 ms
Latency of p95 (success)	939.28 ms	3.21 ms	57.53 ms
Latency of p99 (success)	979.24 ms	4.51 ms	97.49 ms
Maximum successful requests per second	23	50	1500

Machine type: c2-standard-8, Python 3.7, Sanic workers: 7, Prediction threads: 1, vCPU cores: 7, Memory: 15 Gi, Deployment Replicas: 1

risk **analysis** predict on the fly



risk analysis precompute recommendations

Estimate of time (seconds) to precompute recommendations

Number of threads	1	1	1	4	4	4	8	8	8	16	16	16	30	30	30
Candidate items \ Chunk of users	10k	60k	100k	10k	60k	100k	10k	60k	100k	10k	60k	100k	10k	60k	100k
1 user	0.01	0.05	0.05	0.01	0.01	0.02	0.02	0.02	0.03	0.01	0.02	0.03	0.01	0.01	0.03
100 users	0.32	2.45	4.21	0.16	1.11	1.86	0.14	0.86	1.45	0.12	0.76	1.28	0.11	0.73	1.20
1000 users	3.09	24.10	39.95	1.44	10.67	18.31	1.17	8.47	14.22	1.04	7.25	12.53	1.00	7.02	11.75
10000 users	31.05	231.13	409.41	14.24	109.31	184.96	11.40	84.50	141.85	9.99	73.69	125.3	9.75	69.73	116.58

analysis using c2-standard-30 (30 vCPU and 120 RAM) and LightFM

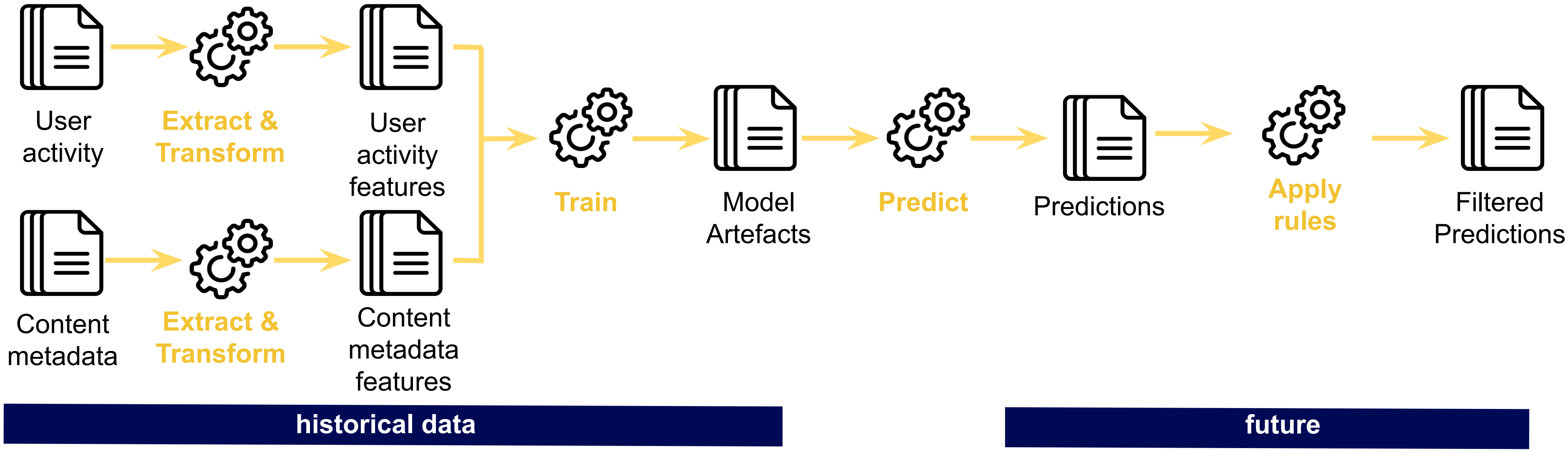
cost estimate: ~ US$ 10.00 run

risk analysis sorting recommendations

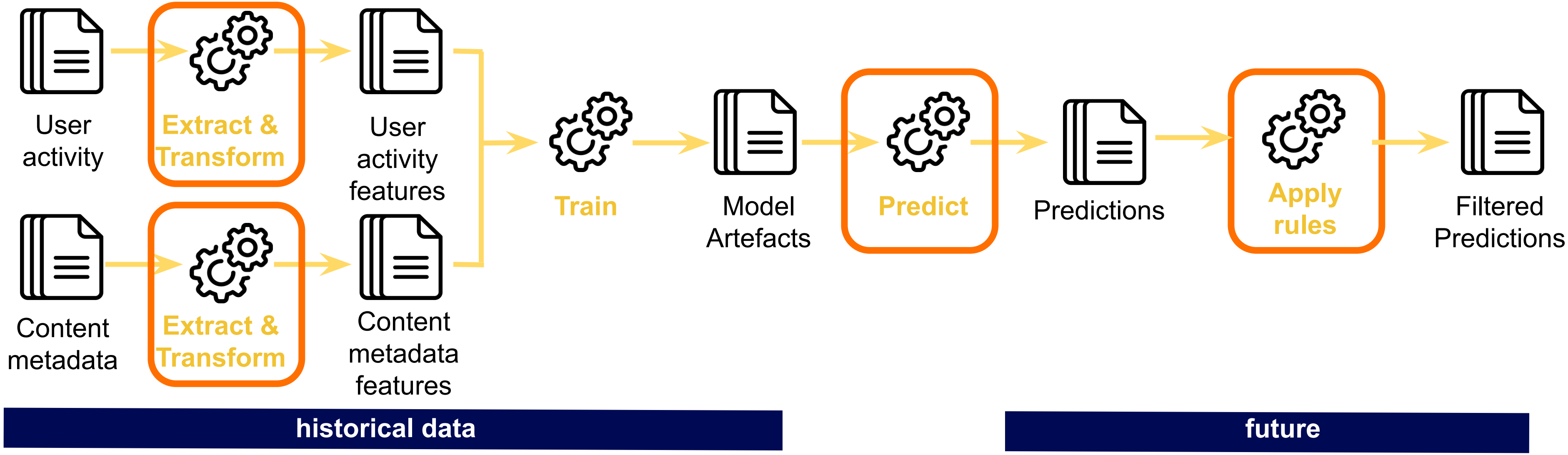
Sorting time (in seconds)			
Chunk of users \ Candidate items	10k	60k	100k
1 user	0.14	0.10	0.17
100 users	1.39	9.68	17.08
1000 users	16.07	105.35	180.60
10000 users	-	-	-

sort 100k predictions per user with pure Python did not seem efficient

architecture overview

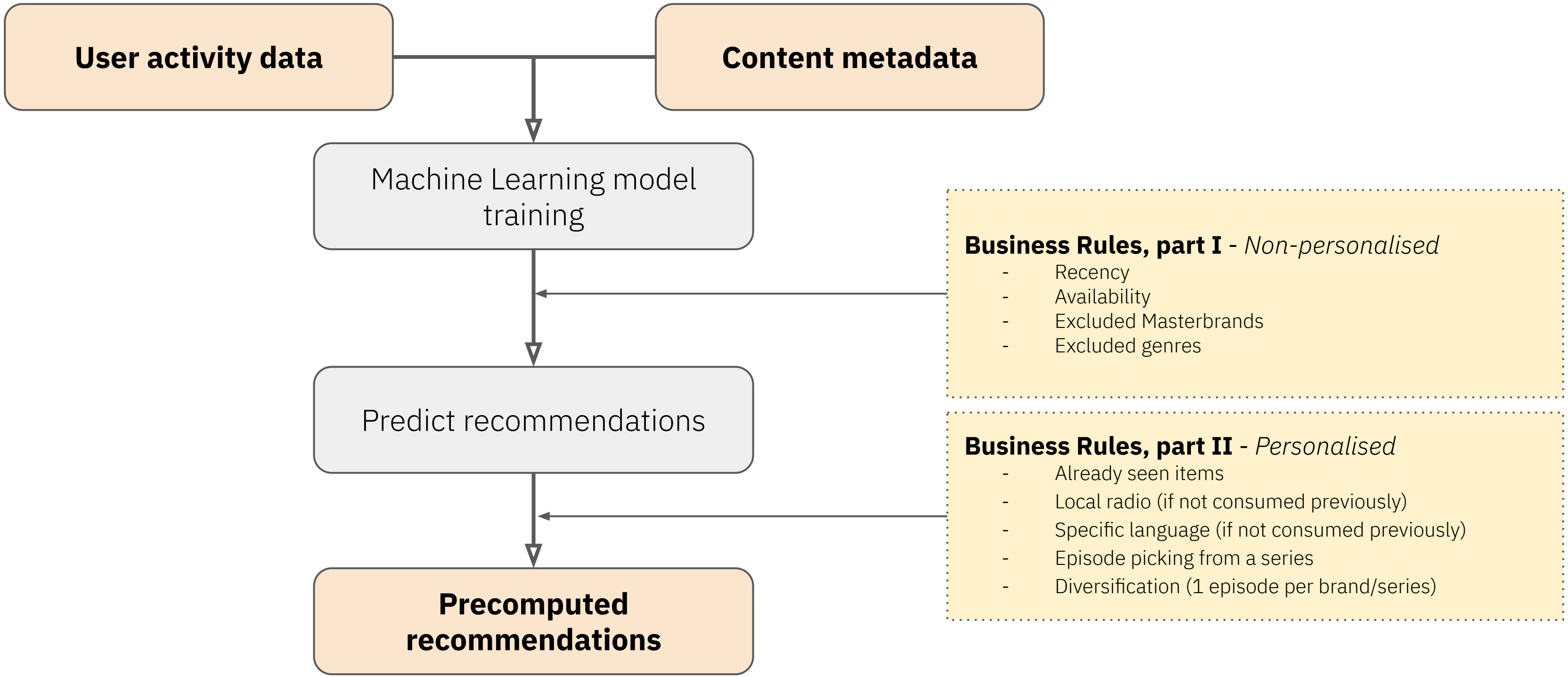


architecture overview



where we used Apache Beam

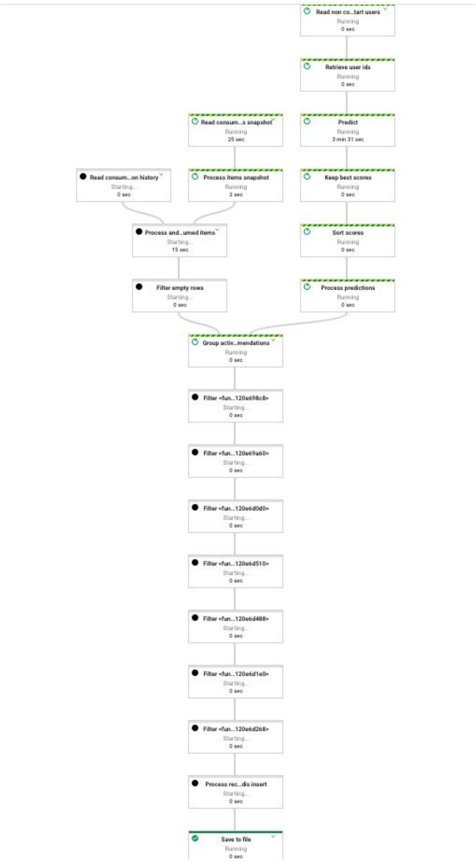
architecture overview



precompute recommendations

pipeline evolution

pipeline 1.0 **design & arguments**

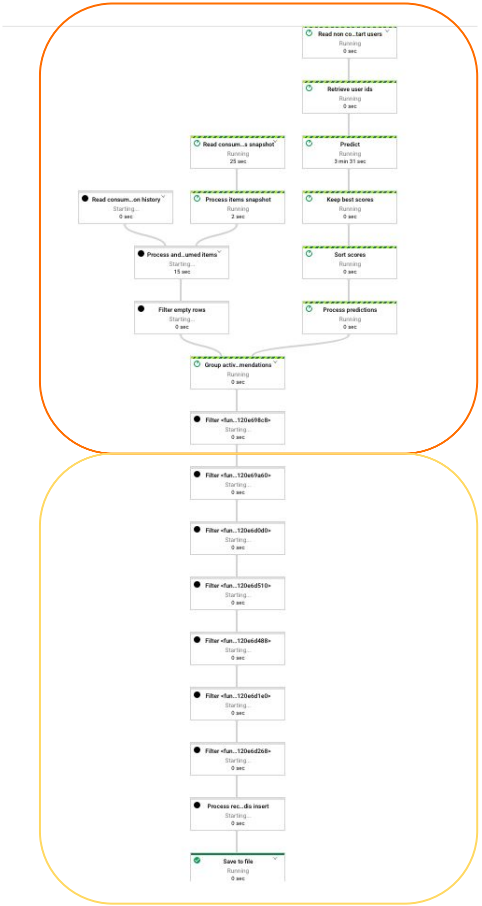


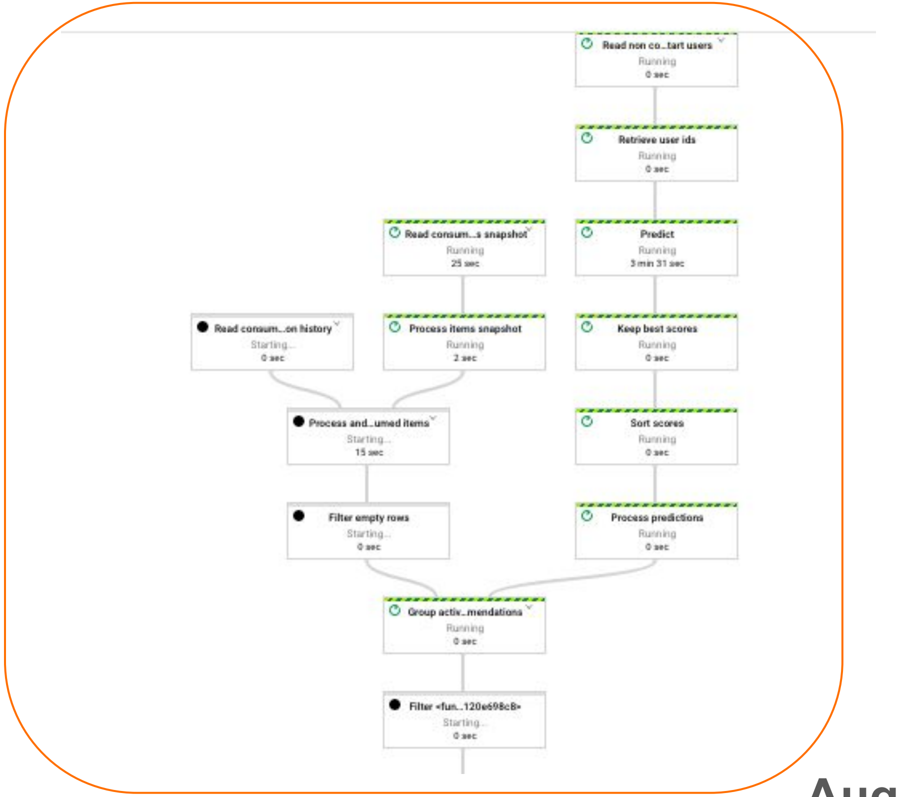

```
apache-beam[gcp]==2.15.0

--runner=DataflowRunner
--machine-type = n1-standard-1 (1 vCPU & 3.75 GB RAM)
--num_workers=10
--autoscaling_algorithm=NONE
```

August 2020

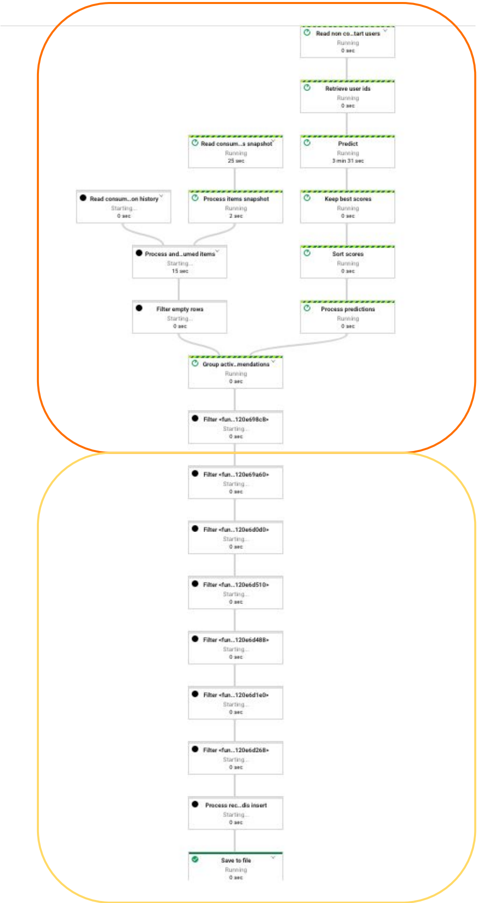
pipeline 1.0 **design**





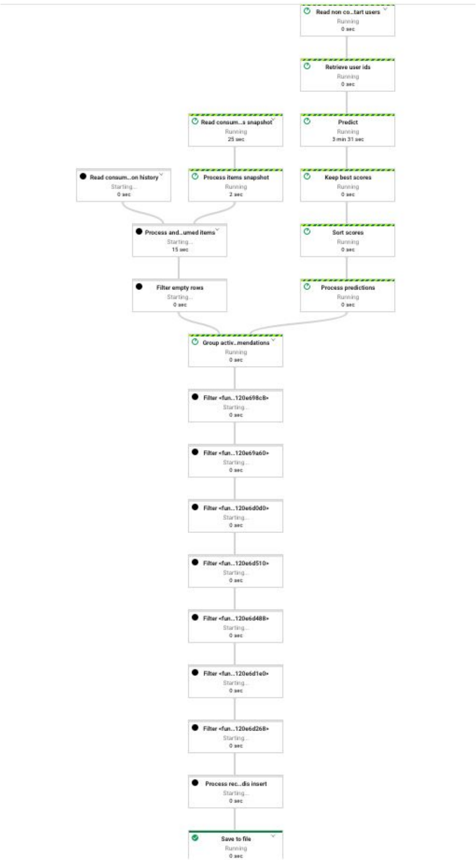
August 2020

pipeline 1.0 **design**



August 2020

pipeline 1.0 error when running in dev & prod




```
Workflow failed. Causes: S05:Read non-cold start
users/Read+Retrieve user ids+Predict+Keep best scores+Sort
scores+Process predictions+Group activity history and
recommendations/pair_with_recommendations+Group activity
history and recommendations/GroupByKey/Reify+Group activity
history and recommendations/GroupByKey/Write failed., The job
failed because a work item has failed 4 times. Look in previous log
entries for the cause of each one of the 4 failures. For more
information, see
https://cloud.google.com/dataflow/docs/guides/common-errors.
The work item was attempted on these workers:
  beamapp-al-cht01-08141052-08140353-1tqj-harness-0k4v
      Root cause: The worker lost contact with the service.,
  beamapp-al-cht01-08141052-08140353-1tqj-harness-0k4v
      Root cause: The worker lost contact with the service.,
  beamapp-al-cht01-08141052-08140353-1tqj-harness-ffqv
      Root cause: The worker lost contact with the service.,
  beamapp-al-cht01-08141052-08140353-1tqj-harness-cjht
      Root cause: The worker lost contact with the service.
```

August 2020

pipeline 1.0 data analysis

Data structure	(dev) size in disk	(prod) Size in disk		(dev) size in memory	(prod) size in memory
model	1.74 GiB	3.99 GiB		2.49 GB	4.50 GB
candidate items	22.58 MiB	23.95 MiB		0.19 GB	0.20 GB
item features	832.11 KiB	1.26 MiB		0.04 GB	0.05 GB
mapping	357 MiB	404 MiB		1.30 GB	2.79 GB
consumed items	20.04 MiB	48.16 MiB			
consumption history	299 MiB	1.45 GiB			
				4.02 GB	**7.54 GB**

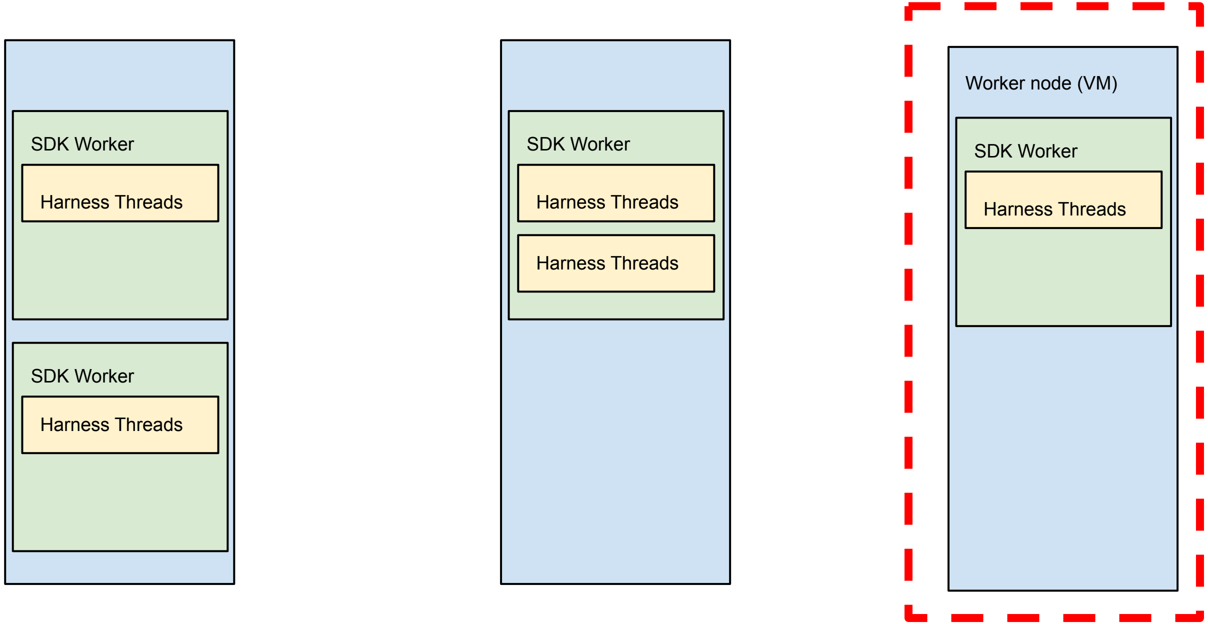
August 2020

pipeline 1.0 attempts to fix (i)

1. Change machine type to a larger one
   - --machine_type=custom-1-6656 (1 vCPU, 6.5 GB RM) - 6.5GB RAM /core
   - --machine_type=m1-ultramem-40 (40 vCPU, 961 GB RAM) - 24GB RAM/core
2. Refactor the pipeline
3. Reshuffle => too expensive for the operation we were doing
   - Shuffle service
   - Reshuffle function
4. Increase the amount of workers
   - --num_workers=40

pipeline 1.0 attempts to fix (ii)

5. Control the parallelism in Dataflow so the VM wouldn't starve out of memory

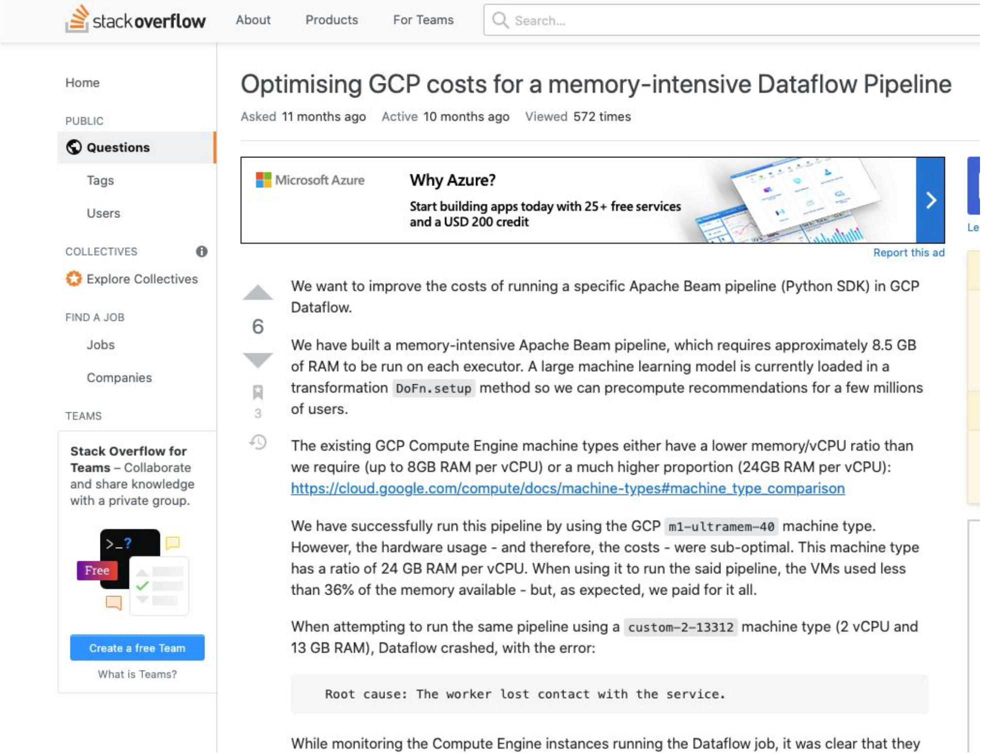


--number_of_worker_harness_threads=1
--experiments=use_runner_v2
(or)
--sdk_worker_parallelism

--experiments=no_use_multiple_sdk_containers
--experiments=beam_fn_api

September 2020

pipeline 1.0 attempts to fix (iii)



Optimising GCP costs for a memory-intensive Dataflow Pipeline

Asked 11 months ago Active 10 months ago Viewed 572 times

We want to improve the costs of running a specific Apache Beam pipeline (Python SDK) in GCP Dataflow.

We have built a memory-intensive Apache Beam pipeline, which requires approximately 8.5 GB of RAM to be run on each executor. A large machine learning model is currently loaded in a transformation `DoFn.setup` method so we can precompute recommendations for a few millions of users.

The existing GCP Compute Engine machine types either have a lower memory/vCPU ratio than we require (up to 8GB RAM per vCPU) or a much higher proportion (24GB RAM per vCPU): https://cloud.google.com/compute/docs/machine-types#machine_type_comparison

We have successfully run this pipeline by using the GCP `m1-ultramem-40` machine type. However, the hardware usage - and therefore, the costs - were sub-optimal. This machine type has a ratio of 24 GB RAM per vCPU. When using it to run the said pipeline, the VMs used less than 36% of the memory available - but, as expected, we paid for it all.

When attempting to run the same pipeline using a `custom-2-13312` machine type (2 vCPU and 13 GB RAM), Dataflow crashed, with the error:

```
Root cause: The worker lost contact with the service.
```

While monitoring the Compute Engine instances running the Dataflow job, it was clear that they

https://stackoverflow.com/questions/63705660/optimising-gcp-costs-for-a-memory-intensive-dataflow-pipeline

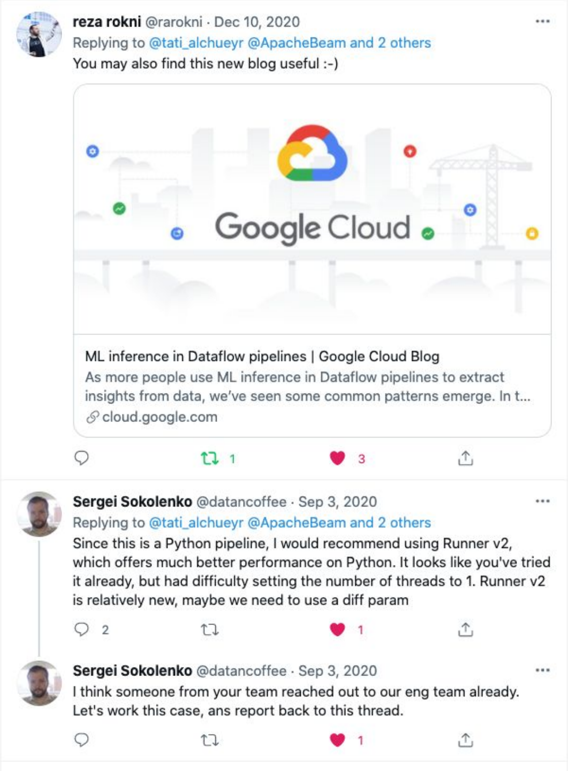
pipeline 1.0 attempts to fix (iii)

Tatiana Al-Chueyr
@tati_alchueyr

We've been struggling to reduce the costs of an @ApacheBeam pipeline in @googlecloud Dataflow: stackoverflow.com/questions/6370.... Any ideas @datancoffee @rarokni?

Optimising GCP costs for a memory-intensive Dataflow Pi...
We want to improve the costs of running a specific Apache Beam pipeline (Python SDK) in GCP Dataflow. We have bu...
stackoverflow.com

2:39 PM · Sep 2, 2020 · Twitter Web App



reza rokni @rarokni · Dec 10, 2020
Replying to @tati_alchueyr @ApacheBeam and 2 others
You may also find this new blog useful :-)

Google Cloud

ML inference in Dataflow pipelines | Google Cloud Blog
As more people use ML inference in Dataflow pipelines to extract insights from data, we've seen some common patterns emerge. In t...
cloud.google.com

1 3

Sergei Sokolenko @datancoffee · Sep 3, 2020
Replying to @tati_alchueyr @ApacheBeam and 2 others
Since this is a Python pipeline, I would recommend using Runner v2, which offers much better performance on Python. It looks like you've tried it already, but had difficulty setting the number of threads to 1. Runner v2 is relatively new, maybe we need to use a diff param

2 1

Sergei Sokolenko @datancoffee · Sep 3, 2020
I think someone from your team reached out to our eng team already. Let's work this case, ans report back to this thread.

1

https://twitter.com/tati_alchueyr/status/1301152715498758146
https://cloud.google.com/blog/products/data-analytics/ml-inference-in-dataflow-pipelines

pipeline 1.0 attempts to fix (iii)

2 Answers

Active | Oldest | Votes

3

We are working on long-term solutions to these problems, but here is a tactical fix that should prevent the model duplication that you saw in approaches 1 and 2:

Share the model in a VM across workers, to avoid it being duplicated in each worker. Use the following utility (https://github.com/apache/beam/blob/master/sdks/python/apache_beam/utils/shared.py), which is available out of the box in Beam 2.24 If you are using an earlier version of Beam, copy just the shared.py to your project and use it as user code.

Share Follow

answered Sep 9 '20 at 0:21
Sergei
201 •1 •4

Add a comment

https://stackoverflow.com/questions/63705660/optimising-gcp-costs-for-a-memory-intensive-dataflow-pipeline

pipeline 1.0 attempts to fix (iii)

2

I don't think that at this moment there's an option to control the number of executors per VM, it seems that the closest that you will get there is by using the option (1) and assume a Python executor per core.

> Option (1)
>
> `--number_of_worker_harness_threads=1 --experiments=use_runner_v2`

To compensate on the cpu-mem ratio you need, I'd suggest using custom machines with extended memory. This approach should be more cost-effective.

For example, the cost of a running a single executor and a single thread on a `n1-standard-4` machine (4 CPUs - 15GB) will be roughly around 30% more expensive than running the same workload using a `custom-1-15360-ext` (1 CPU - 15GB) custom machine.

Share Follow

edited Sep 3 '20 at 22:34

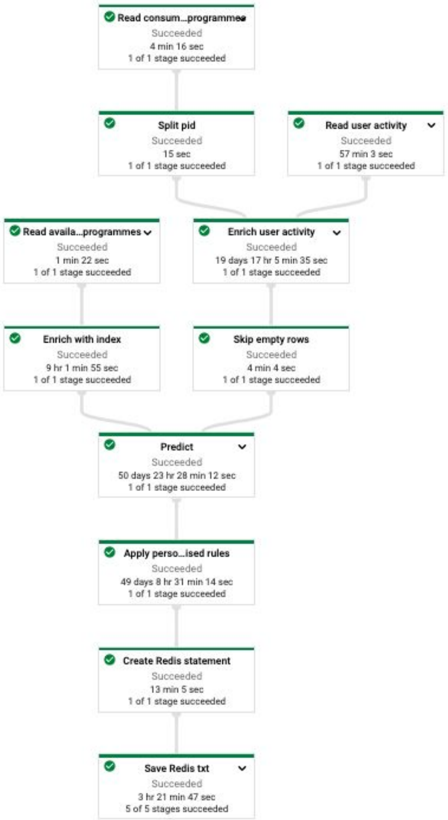
answered Sep 3 '20 at 21:00

Tlaquetzal

2,197 1 8 18

https://stackoverflow.com/questions/63705660/optimising-gcp-costs-for-a-memory-intensive-dataflow-pipeline

pipeline 2.0 design & arguments



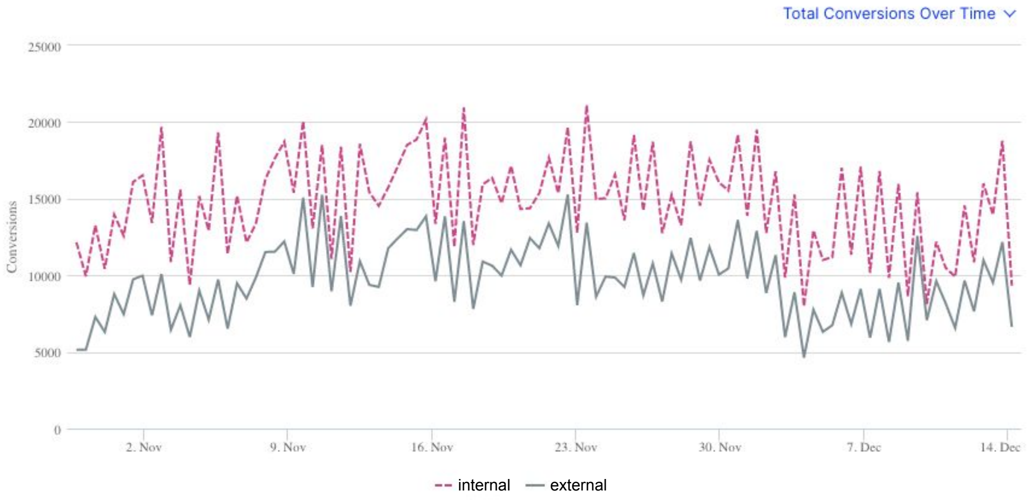
apache-beam== 2.24

--runner=DataflowRunner
--machine-type = custom-30-460800-ext
--num_workers= 40
--autoscaling_algorithm=NONE

September 2020

pipeline 2.0 business outcomes

- +59% increase in interactions in Recommended for You rail
- +103% increase in interactions for under 35s



September 2020

pipeline 2.0 issues

- but costs were high...

Resource metrics

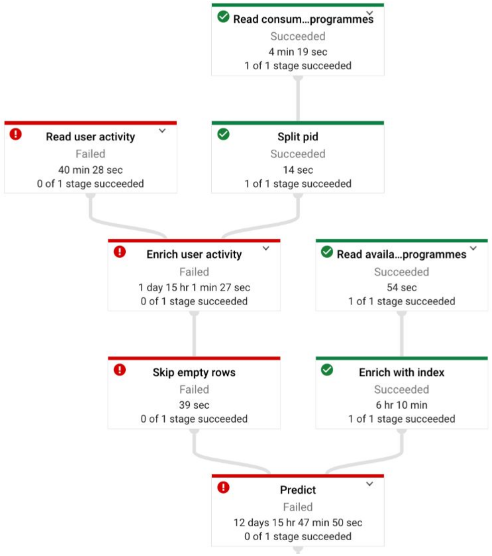
Current vCPUs	1,200
Total vCPU time	3,170.248 vCPU hr
Current memory	17.58 TB
Total memory time	47,553.721 GB hr
Current HDD PD	9.77 TB
Total HDD PD time	26,418.734 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr

Elapsed time	2 hr 42 min
Encryption type	Google-managed key

£ 279.31 per run

September 2020

pipeline 2.0 **issues**

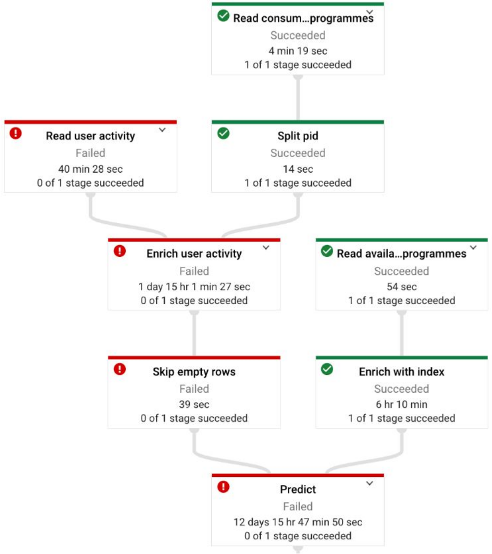



```
OSError: [Errno 28] No space left on device During handling
```

	successful compute-predictions-21628eba 2021-02-24_20_06_12-17726452853028084617 25 February 2021	unsuccessful compute-predictions-191557b2 2021-02-28_20_01_56-16233421279575525541 1st March 2021
consumption-history/	1.87 GiB	1.88 GiB
consumed-items/	55.63 MiB	55.68 MiB
candidate-items/rfy/	25.06 MiB	25.09 MiB
xantus.pkl	4.67 GiB	4.69 GiB
item_features.npz	1.73 MiB	1.73 MiB
mapping.json	473.84 MiB	476.01 MiB

March 2021

pipeline 2.0 issues



Resource metrics

Current vCPUs	1,200
Total vCPU time	1,025.553 vCPU hr
Current memory	17.58 TB
Total memory time	15,383.289 GB hr
Current HDD PD	1,000 GB
Total HDD PD time	854.627 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	241.01 KB
Billable Shuffle data processed	60.25 KB

If a batch job uses Dataflow Shuffle, then the default is 25 GB; otherwise, the default is 250 GB.

March 2021

pipeline 2.0 issues

apache-beam== 2.24

--runner=DataflowRunner
--machine-type = custom-30-460800-ext
--num_workers= 40
--autoscaling_algorithm=NONE
--experiments=shuffle_mode=appliance

March 2021

cost savings plan

1. Administer pain relief

- ➔ Attempt shared memory
- ➔ Attempt FlexRS

Timebox: 1 week

2. Hook up to bypass

- ➔ Mid week delta (only compute mid week for users with activity since Sunday's run)

Timebox: 2 weeks

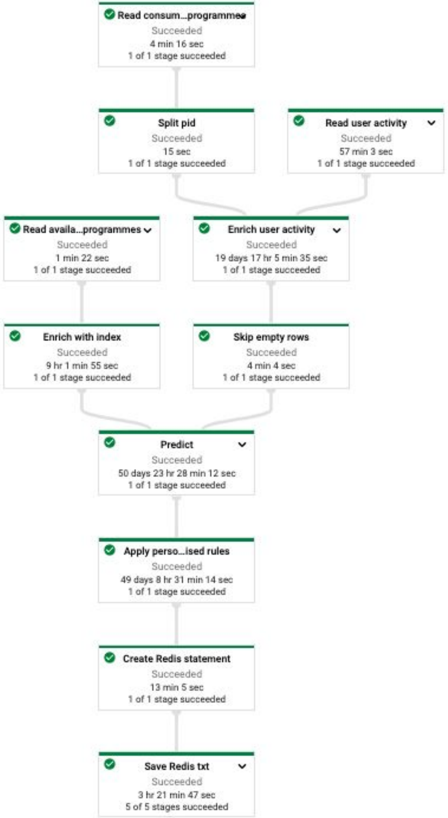
3. Heart surgery

- ➔ Split pipeline
- ➔ Major refactor
- ➔ SCANN vs LightFM.score()
- ➔ etc.

Timebox: 1 month

April 2021

pipeline 3.0 design




```
apache-beam== 2.24

--runner=DataflowRunner
--machine-type = custom-30-460800-ext
--num_workers= 40
--autoscaling_algorithm=NONE
--experiments=shuffle_mode=appliance
```

April 2021

pipeline 3.0 shared memory & FlexRS strategy

- Used production-representative data (model, auxiliary data structures)
- Ran the pipeline for 0.5% users, so the iterations would be cheap
  - 100% users: £ 266.74
  - 0.5% users: £ 80.54
- Attempts
  - Shared model using custom-30-460800-ext (15 GB/vCPU)
  - Shared model using custom-30-299520-ext (9.75 GB/vCPU)
  - Shared model using custom-6-50688-ext (8.25 GB/vCPU)
    - 0.5% users: £ 18.46 => -77.5% cost reduction!

May 2021

pipeline 3.0 shared memory & FlexRS results

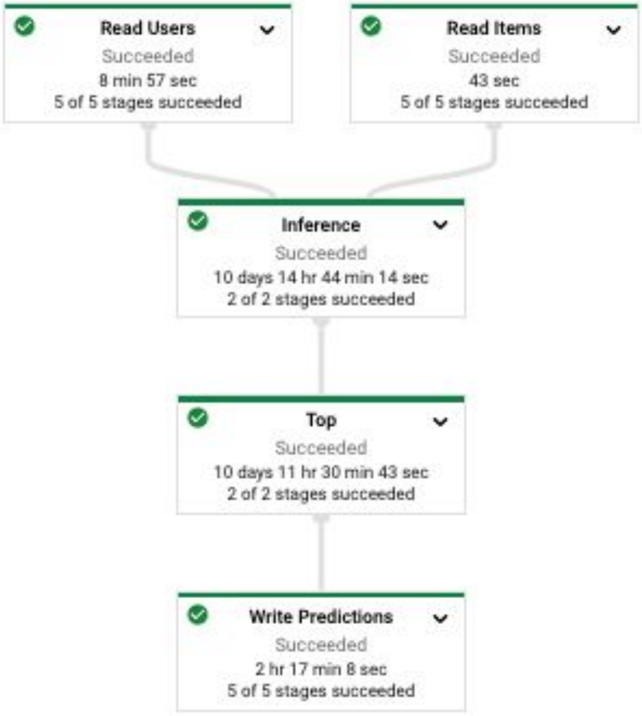
- However, when we tried to run the same pipeline for 100%, it would take hours and not complete.
- It was very inefficient and costed more than the initial implementation.

pipeline 4.0 heart surgery

- Split compute predictions from applying rules
- Keep the interfaces to a minimal
  - between these two pipelines
  - between steps within the same pipeline

June 2021

pipeline 4.1 precompute recommendations



apache-beam== 2.29

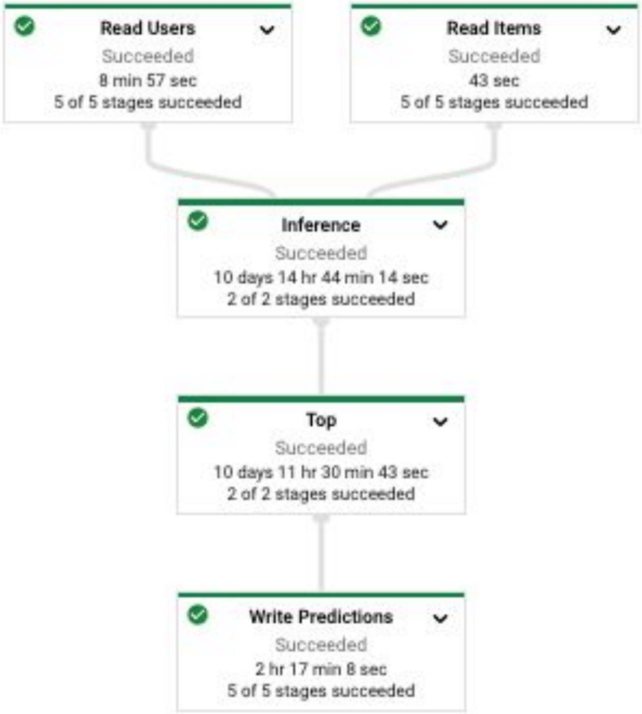
--runner=DataflowRunner
--machine-type = n1-highmem-16
--flexrs-goal = COST_OPTIMIZED
--max-num-workers= 64
--number-of-worker-harness-threads=7
--experiments=use_runner_v2

+ Batching
+ Shared memory

https://cloud.google.com/blog/products/data-analytics/ml-inference-in-dataflow-pipelines

July 2021

pipeline 4.1 precompute recommendations



Resource metrics

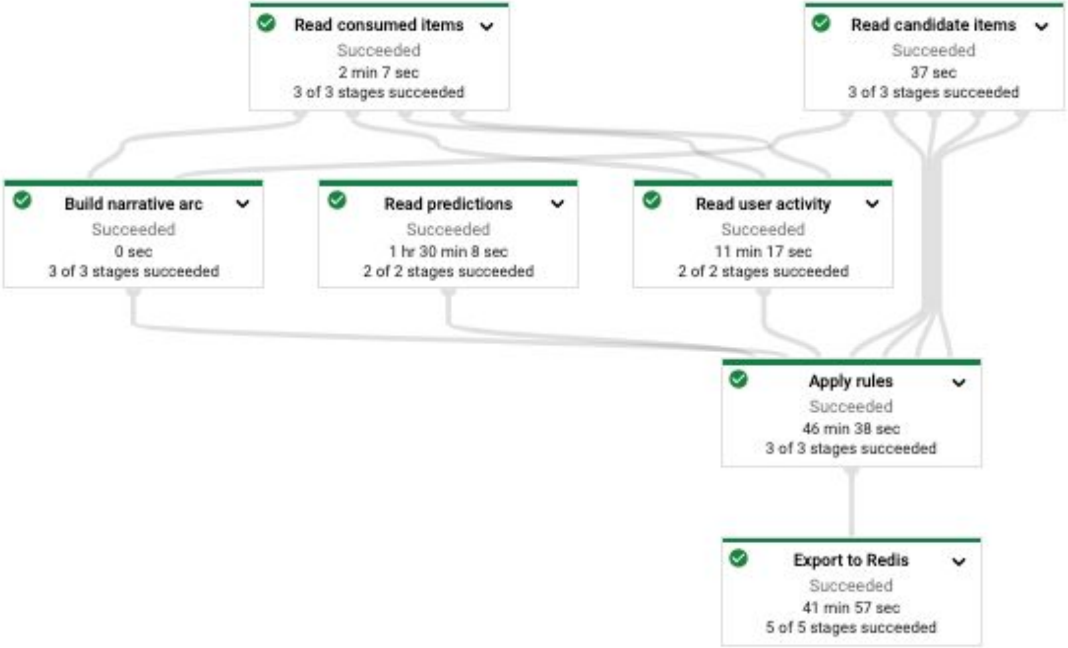
Current vCPUs	252
Total vCPU time	38.042 vCPU hr
Current memory	945 GB
Total memory time	142.658 GB hr
Current HDD PD	6.15 TB
Total HDD PD time	951.053 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	271.44 GB
Billable Shuffle data processed	73.22 GB

Cost to run for 3.5 million users:

- 100k episodes: £ 48.92 / run
- 300 episodes: £ 3.40
- 18 episodes: £0.74

July 2021

pipeline 4.2 apply business rules



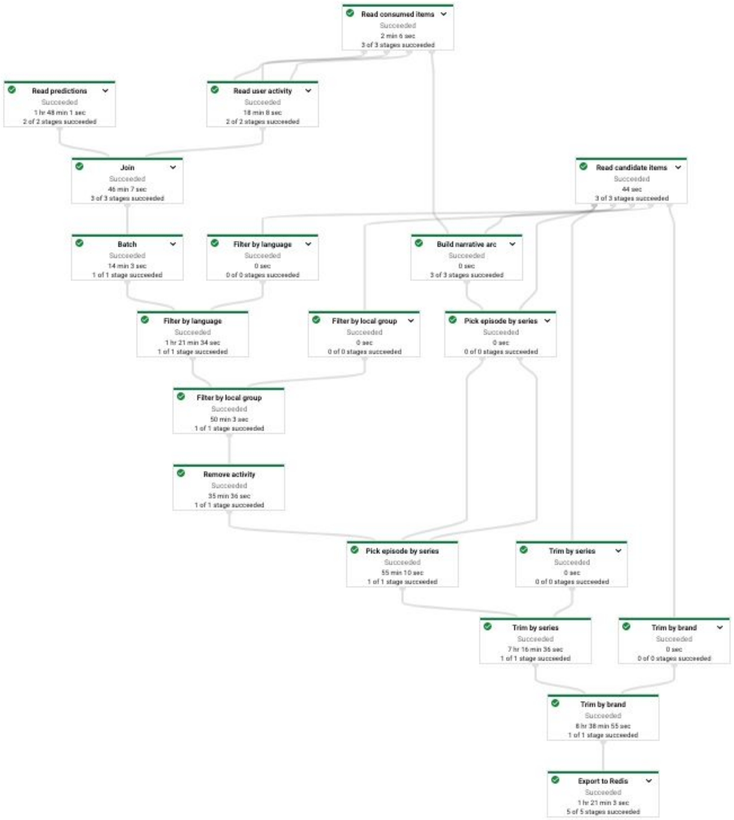
apache-beam== 2.29

--runner=DataflowRunner
--machine-type = n1-standard-1
--experiments=use_runner_v2

+ Implemented rules natively
+ Created minimal interfaces and views of the data

July 2021

pipeline 4.2 apply business rules



Resource metrics

Current vCPUs	252
Total vCPU time	38.042 vCPU hr
Current memory	945 GB
Total memory time	142.658 GB hr
Current HDD PD	6.15 TB
Total HDD PD time	951.053 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Total Shuffle data processed	271.44 GB
Billable Shuffle data processed	73.22 GB

Cost to run for 3.5 million users:

- £ 0.15 - 0.83 per run

July 2021

pipeline 4.0 heart surgery

- We were able to reduce the cost of the most expensive run of the pipeline from **£ 279.31** per run to less than **£ 50**
- Reduced the costs to -82%

July 2021

take**aways**

takeaways

1. plan based on your **data**
2. an **expensive** machine learning pipeline is better **than none**
3. reducing the **scope** is a good starting point to **saving money**
    - Apply non-personalised rules before iterating per user
    - Sort top 1k recommendations by user opposed to 100k
4. using **custom machine types** **might limit** other **cost savings**
    - Such as FlexRS (schedulable preemptible instances in Dataflow only work)
5. to use **shared memory** may not lead to cost savings
6. **minimal interfaces** lead to more **predictable behaviours** in Dataflow
7. **splitting the pipeline** can be **a solution to costs**

Thank you!

@tati_alchueyr