Leveraging public cloud resources for the processing of CMS open data

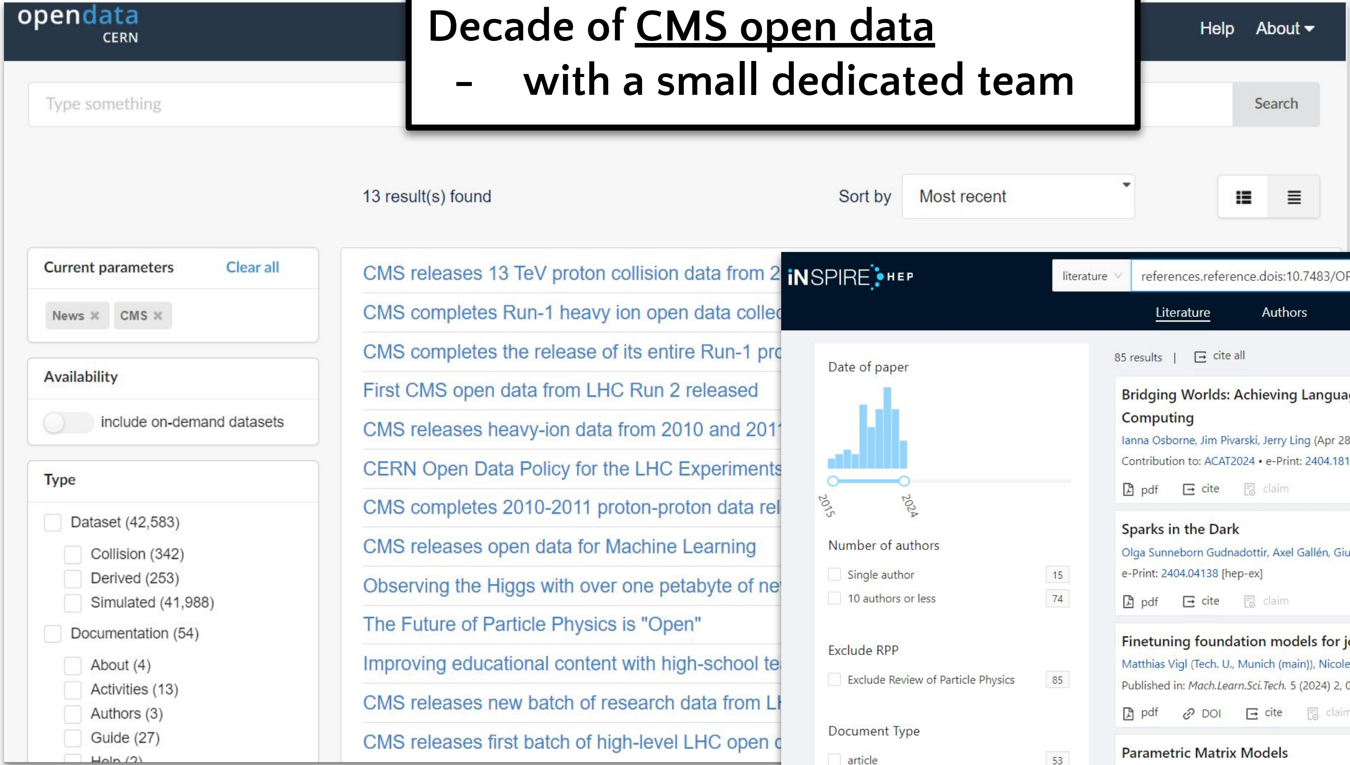
CHEP - October 19 - 25, 2024

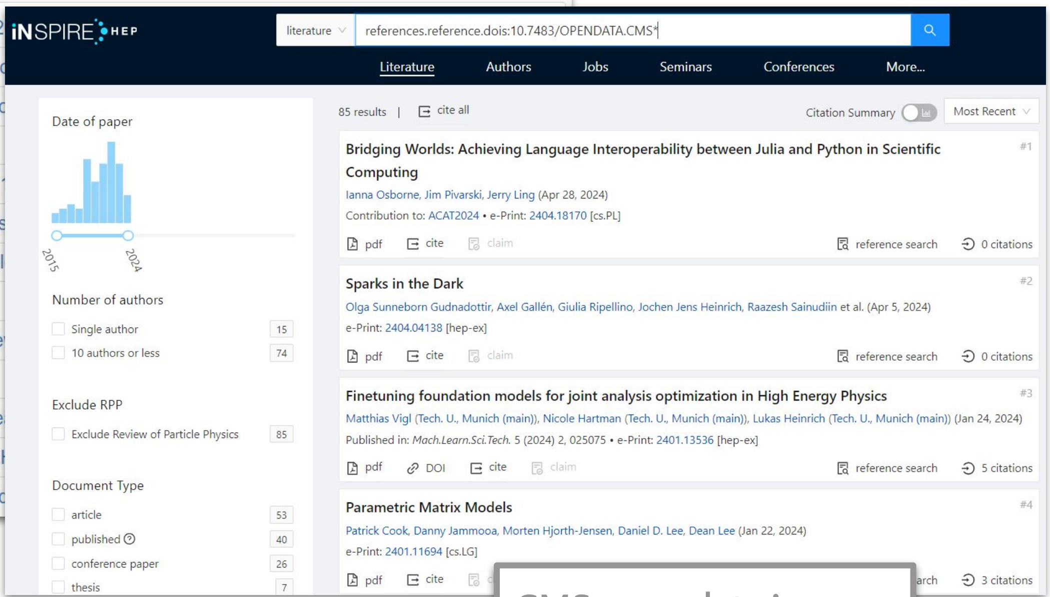
Kati Lassila-Perini
Helsinki Institute of Physics - Finland
Tom Cordruwisch, Subash Jayawardhana
Lapland University of Applied Sciences - Finland

1

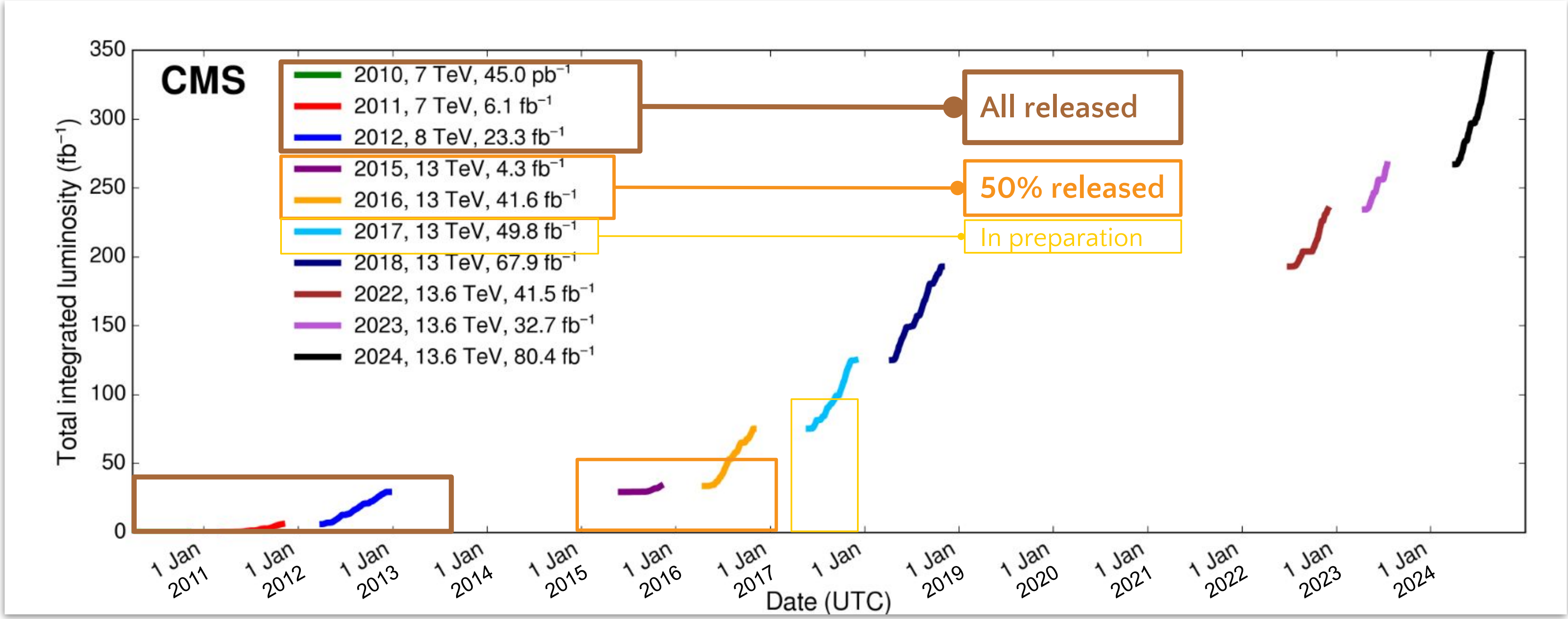
CMS Open Data

Decade of CMS open data – with a small dedicated team





CMS open data in use



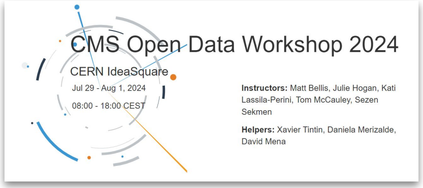
Open data quantity & time defined in the CMS open data policy

2

Public cloud and Kubernetes

Why and how?

Motivation: 1 - Why?



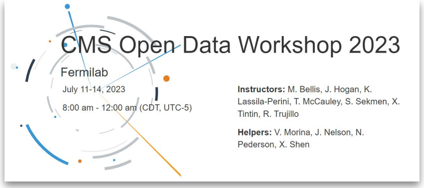
NanoAOD:
compact format
(1 kB/evt)
good for many analyses
no need for CMSSW

ease of use

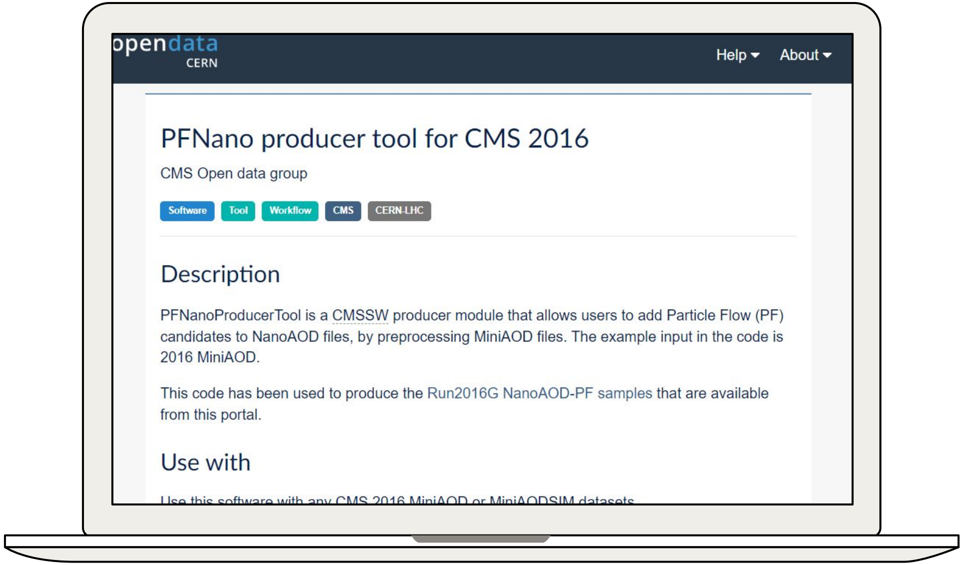
completeness

MiniAOD:
richest Run-2 OD format
(30-40 kB/evt)
good for ~ all analyses
requires CMSSW

Custom NanoAOD:
enrich NanoAOD with what you need



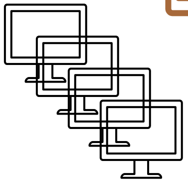
Motivation: 2 - How?



code: GitHub

env: docker

resources:



Custom NanoAOD: enrich NanoAOD with what you need

Why cloud and Kubernetes?

		Do you have?	Want to get?	
1	Computing resources	✓	✗	• No need for cloud resources • Ask your IT support
2	Tools	✗	✓	• Read the CMS OD workshop tutorial (HTCondor) • Ask your IT support
3	Skills	✗	✓	• Read the CMS OD workshop tutorial (HTCondor) • Ask your IT support
4	Your custom CMS OD	✗	✓	• Process what you need • Download and store locally • Analyze on your own resources

Why cloud and Kubernetes?

		Do you have?	Want to get?	Why cloud / Kubernetes?
1	Computing resources	✗	✓	• Short-term, immediate resources • Pay what you use • Compatible with CMS OD environment
2	Tools	✗	✓	• Kubernetes as a basic tool • Terraform to deploy resources • Argo workflows to manage jobs • Open source and free
3	Skills	✗	✓	• Applicable within and beyond research • Attractive for early careers and young-minded • Example setup provided - we did it for you!
4	Your custom CMS OD	✗	✓	• Process what you need • Download and store locally • Analyze on your own resources

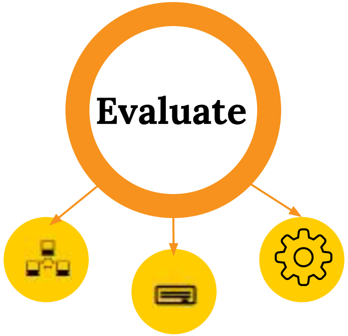
3

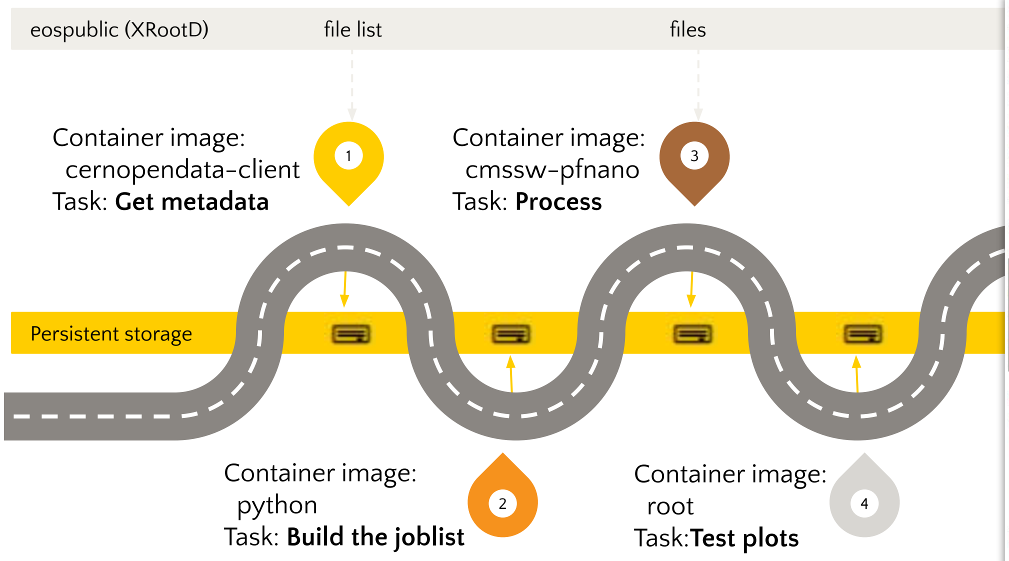
How much time and money?

Resources from a Google Cloud Research credit 303424260

Benchmarking use case

- Processing custom-NanoAOD:
  - input: MiniAOD data
  - full NanoAOD processing + Particle-Flow (PF) candidates
    - same as in the already-provided PF-Nano datasets on the portal
  - image: CMS OD image with PFNano processing code precompiled
  - using Argo workflow to run the processing
    - steps: get dataset metadata → make a joblist → process x N_{jobs} → (test plot)






```
$ argo get -n argo @latest
Name: pfnano-process-6rwwj
Namespace: argo
ServiceAccount: argo-service-account
Status: Running
Conditions:
PodRunning True
Created: Fri Sep 27 23:48:05 +0200 (1 minute ago)
Started: Fri Sep 27 23:48:05 +0200 (1 minute ago)
Duration: 1 minute 28 seconds
Progress: 2/355
ResourcesDuration: 1s*(1 cpu),25s*(100Mi memory)
Parameters:
nEvents: -1
recid: 30511
nJobs: 353
bucket: test-gcs-argo-bucket-regional

STEP TEMPLATE PODNAME DURATION MESSAGE
● pfnano-process-6rwwj cms-od-example
├─✓ get-metadata get-metadata-template pfnano-process-6rwwj-get-metadata-template-4259457820 17s
├─✓ joblist joblist-template pfnano-process-6rwwj-joblist-template-3232337233 11s
├─● runpfnano(0:eventsinjob:-1,firstfile:1,it:1,lastfile:1) runpfnano-template pfnano-process-6rwwj-runpfnano-templ
├─● runpfnano(1:eventsinjob:-1,firstfile:2,it:2,lastfile:2) runpfnano-template pfnano-process-6rwwj-runpfnano-templ
[...]
├─● runpfnano(348:eventsinjob:-1,firstfile:349,it:349,lastfile:349) runpfnano-template pfnano-process-6rwwj-runpfn
├─● runpfnano(349:eventsinjob:-1,firstfile:350,it:350,lastfile:350) runpfnano-template pfnano-process-6rwwj-runpfn
├─● runpfnano(350:eventsinjob:-1,firstfile:351,it:351,lastfile:351) runpfnano-template pfnano-process-6rwwj-runpfn
├─● runpfnano(351:eventsinjob:-1,firstfile:352,it:352,lastfile:352) runpfnano-template pfnano-process-6rwwj-runpfn
└─● runpfnano(352:eventsinjob:-1,firstfile:353,it:353,lastfile:353) runpfnano-template pfnano-process-6rwwj-runpfn
```

Workflow structure

12

Cluster types

- Vocabulary
  - **GKE** = Google Kubernetes Engine
  - **Cluster** consists of nodes (machines with local boot disks)
  - Jobs run on **pods** (containerized applications) on the nodes
  - **Persistent storage** is a disk available for all nodes (and pods)
- Two types of GKE clusters: standard and auto-pilot:

Using: **GKE Standard clusters**

- allows defining all cluster components (type and number of nodes)
- auto-scales (nodes deleted automatically) if so configured
- cost goes with time and depends on the cluster setup
- can create nodes with the container image from a secondary boot disk

Not tested: GKE Auto-pilot clusters

- creates the cluster based on the resource request
- did not allow for a NFS disk server (container in the privileged mode)
- auto-scales (nodes deleted automatically)
- creating nodes with the image from the secondary boot disk did not work out of the box

Disk types and cost

- NFS disk
  - predefined size
  - the cost / disk size, not / usage
  - requires an NFS server on the cluster
- Google Cloud Storage bucket
  - size not predefined
  - the cost / actual usage
- Storage:
  - negligible (for a cluster lifetime 1-2d)
- Download ("egress"):
  - costly

→ Our choice: GCS bucket for the ease of use

europe-west4 (Netherlands)	NFS	GCS bucket
Storage /month	0.44$/10GB (22$/500GB) SSD: 1.87$/10GB	0.20$/10GB (10$/500GB)
Download cost	0.12$/GB ⚠️👉(60$/500GB)👈⚠️	
Download time	Local download speed Slight dependence on distance	Idem Independent of distance

Cluster and job configuration

- Node type
  - number of CPUs and amount of memory (“standard”/”highcpu”/”highmem”)
- Match with the job resource needs
  - use a test workflow running a single job / node to see the resource needs
  - for a quick, provider-independent check:

```
kubectl top node                (check node resources in use)
kubectl top pods -n argo        (check single pod usage)
```

```
$ kubectl top pods -n argo | grep runpfnano
pfnano-process-g2m2k-runpfnano-template-1439319107   998m         1394Mi
pfnano-process-g2m2k-runpfnano-template-2136346710   999m         1460Mi
```

  - resource requests define how argo distributes the jobs to the nodes
    - goal: close to full occupancy
      - lack of memory kills
      - lack of CPU slows down

```
resources:
  requests:
    cpu: "800m"
    memory: "1.8Gi"
    ephemeral-storage: "5Gi"
```

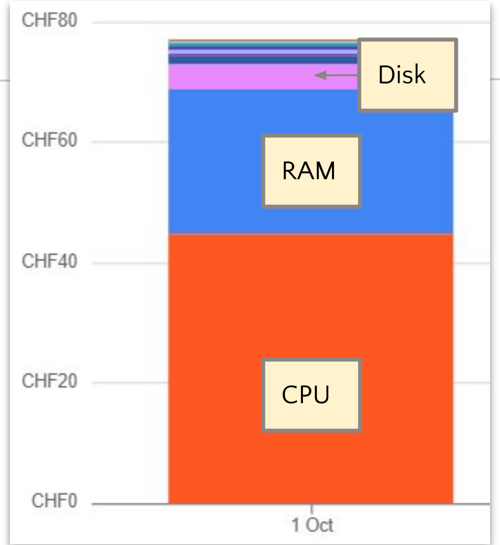
→ 1 job / vCPU

Cost of full dataset processing

▼	Compute Engine	CHF72.94
▲	Cloud Monitoring	CHF1.69
●	Networking	CHF1.65
■	Kubernetes Engine	CHF0.76
◆	Cloud Storage	CHF0.08

→

SKU	Service	SKU ID	Usage	Cost
E2 Instance Core running in Netherlands	Compute Engine	012A-5DBB-1352	2,191.27 hour	CHF44.69
E2 Instance Ram running in Netherlands	Compute Engine	D9EA-4FF0-E394	8,765.17 gibibyte hour	CHF23.96
Storage PD Capacity in Netherlands	Compute Engine	AE8C-46C3-4994	110.43 gibibyte month	CHF4.13
Prometheus Samples Ingested	Cloud Monitoring	A4E4-DF03-CDB6	33,221,907 Samples	CHF1.69
Regional Kubernetes Clusters	Kubernetes Engine	B561-BFBD-1264	8.91 hour	CHF0.76
Network Intelligence Center Network Analyzer Resource Hours	Networking	9BF8-CD36-F9B8	617 count	CHF0.58
Network Intelligence Center Topology and Google Cloud Performance Resource Hours	Networking	D9AD-28F8-05D8	617 count	CHF0.58
Network Intelligence Center Internet to Google Cloud Performance Resource Hours	Networking	BDBA-22FA-3925	617 count	CHF0.42
Network Internet Data Transfer Out from Netherlands to EMEA	Compute Engine	EF9D-7A2D-D06F	1.22 gibibyte	CHF0.12
Standard Storage Netherlands	Cloud Storage	89D8-0CF9-9F2E	4.7 gibibyte month	CHF0.08
Networking Private Service Connect Partner Select End Point	Networking	6882-BFB0-B7E5	8.82 hour	CHF0.07
Storage Image	Compute Engine	DAA2-253C-6680	0.35 gibibyte month	CHF0.01



An example job of 33M events, <u>MuonEG</u> (1.1 TB)

90-node e2-standard-4 regional cluster

(4 vCPUs, 16GB memory / node) - auto-scale

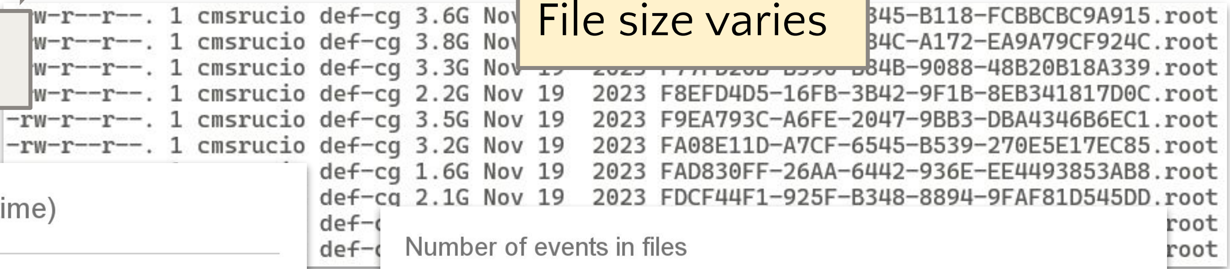
80 CHF - 9 hours

⚠️👉Add data download👈⚠️

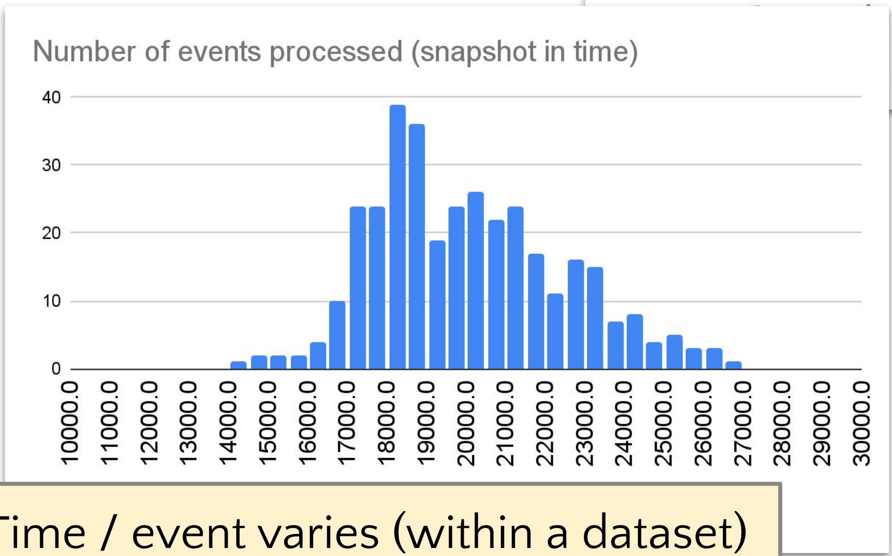
⚠️👉~ 40$ / 350GB 👈⚠️

Auto-scaling - input files and events are not equal!

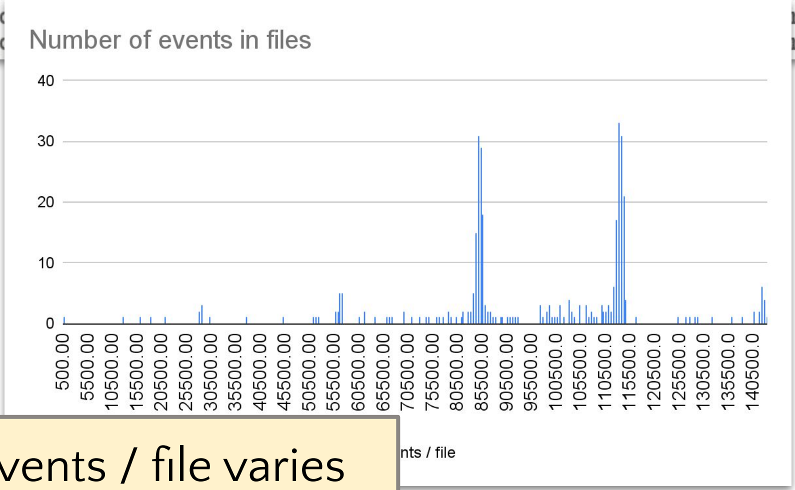
Jobs are not equal in time!


```
-rw-r--r--. 1 cmsrucio def-cg 3.6G Nov                345-B118-FCBBCBC9A915.root
-rw-r--r--. 1 cmsrucio def-cg 3.8G Nov                34C-A172-EA9A79CF924C.root
-rw-r--r--. 1 cmsrucio def-cg 3.3G Nov                84B-9088-48B20B18A339.root
-rw-r--r--. 1 cmsrucio def-cg 2.2G Nov 19  2023 F8EFD4D5-16FB-3B42-9F1B-8EB341817D0C.root
-rw-r--r--. 1 cmsrucio def-cg 3.5G Nov 19  2023 F9EA793C-A6FE-2047-9BB3-DBA4346B6EC1.root
-rw-r--r--. 1 cmsrucio def-cg 3.2G Nov 19  2023 FA08E11D-A7CF-6545-B539-270E5E17EC85.root
                       def-cg 1.6G Nov 19  2023 FAD830FF-26AA-6442-936E-EE4493853AB8.root
                       def-cg 2.1G Nov 19  2023 FDCF44F1-925F-B348-8894-9FAF81D545DD.root
                       def-                                                           root
                       def-                                                           root
```

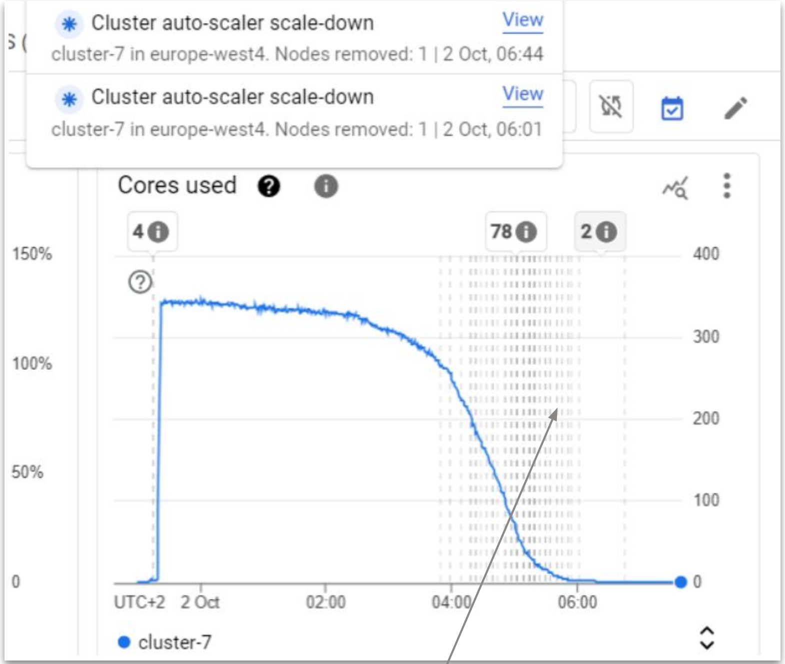
File size varies

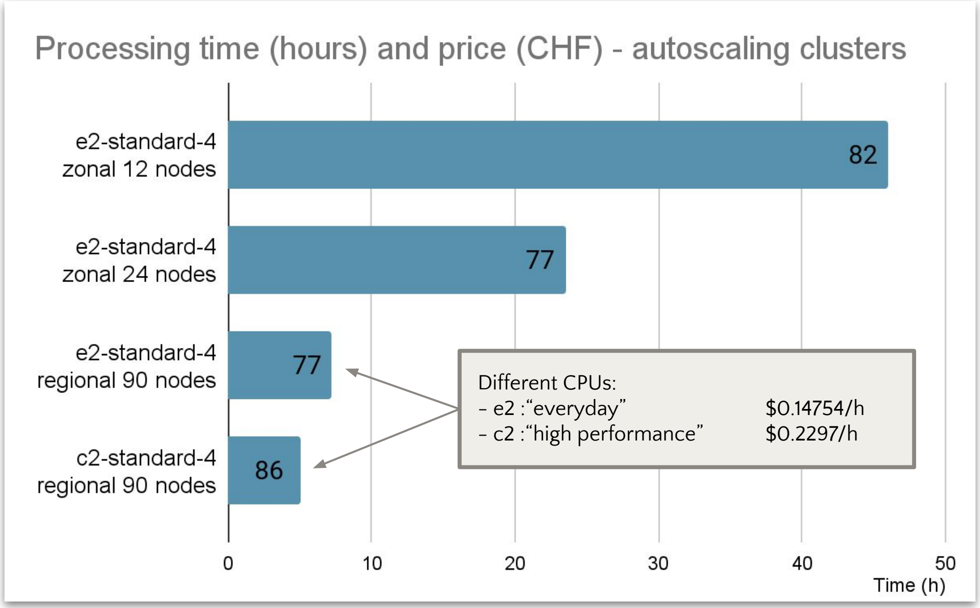


Time / event varies (within a dataset)



Events / file varies





Auto-scaling at work:
cost = N nodes x time

Optimal configuration:
a large cluster for a short time

80$ / 1TB

Approximate price of PFNano-type processing / 1 TB of input data

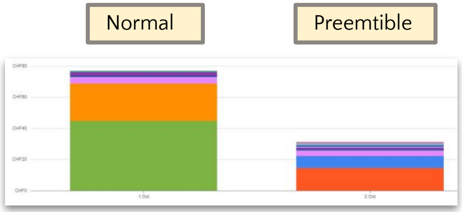
7 h / 1 TB

Time to process 1 TB

40$

Example download (PFNano-type content - 35% of the original size)

- Did the jobs fail and why?
  - Some XRootD timeouts (fixed on server-side), rare cluster networking timeouts.
- Can we gain in speed by uploading input files to the cluster?
  - No, not for this workflow. → See back-up
- How to handle the big CMSSW container image?
  - Use a secondary boot disk.→ See back-up
- Is there an overhead for Kubernetes and Argo services?
  - Nothing significant → See back-up
- What about spot / preembtible nodes (cheap but deletable)?
  - Cheaper but unreliable, definitely worth a follow-up → See back-up



- I/O can be expensive
  - Message logger every event increases run time 1-2 %.
- Mounted disk on kubernetes pods is shared to the persistent storage at the end of the step
  - Internal networking in the cluster can be surprisingly slow.
- Multithreading in kubernetes clusters is not obvious
  - CMSSW jobs can be configured to run in parallel threads within a job.
  - But: in a cluster, 4 single-thread jobs go faster than a 4-thread job in a 4 vCPU node.

FAQ and other observations

Is this difficult?

ADAPT FOR YOUR USE and OFF YOU GO!

Run a test job with *your* processing code.
Evaluate *your* output file size and adapt the disk size accordingly.
Adapt the cluster size and type or use our suggested values.

KNOW YOUR PHYSICS and CMS OD

Have your research idea!
Learn about CMS OD:
- Workshops / docs / support

Need more than NanoAOD?
- No? You could have skipped this talk.
- Yes? Adapt the example to your needs.

GET STARTED WITH OUR INSTRUCTIONS and EXAMPLES

Create a Google Cloud project. / Install: terraform, gcloud (or use Google Cloud shell), argo CLI, kubectl.
Deploy resources using example Terraform scripts.
Run our example job with Argo workflows.

Conclusion and outlook

CMS Open Data can be used without complications, NanoAOD format is a streamlined and condensed storage format that can be analyzed directly by open data users.

For analyses requiring detailed event content, we have demonstrated that using public cloud resources for custom NanoAOD processing is feasible both for time and cost.

We have shown how to optimize disposable cloud resources for a typical processing task.

Containerized CMS Open Data workflows can easily be run in a modern kubernetes environment.

Thank you!

Questions?

And thanks to SlidesCarnival for this free presentation template

Back-up

2014, Nov — **2010 pp, 50%**

First release, virtual machine environment

2016, Apr — **2011 pp, 50%**

Simulated samples, validation examples, basic tools

2017, Dec — **2012 pp, 50%**

More usage examples (Higgs), Jupyter notebooks

2019, Jul — **2010 pp, 100%**

ML samples, special datasets, docker containers, simulated data generation tools

2020, Aug — **2011 pp, 100%**

First examples of automated workflows, improved tools

2020, Dec — **2010-11 HI, 100%**

First heavy-ion data release

2021, Dec — **2015 pp, 99%**

First Run-2 data release, slimmer data format

2022, Dec — **2012 pp, 100%**

Full Run-1 pp data release, improved usage examples

2023, Sept — **2013 HI, 100%**

Full Run-1 heavy-ion data, extended usage examples

2024, Apr — **2016 pp, 50%**

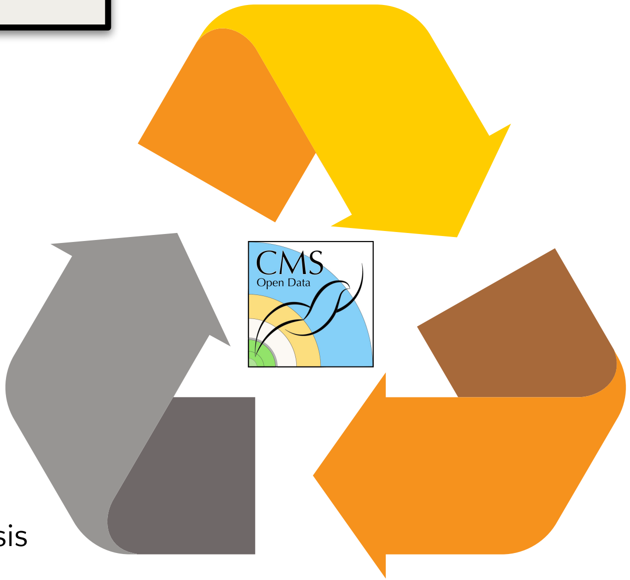
First data format not requiring CMS software,, updated usage instructions

CMS Open Data release timeline releases since 2014

CMS open data:
full research-level data
– not an “open-data” reduction

Tools:

- software
- environments
- interfaces



Data:

- collision data
- simulations
- additional data for analysis

Knowledge:

- instructions
- actionable examples
- understanding of experimental data

Input data streaming vs upload

- Data is streamed with xrootd protocol from eospublic at CERN
  - No significant difference between locations close to of far from CERN
  - Processing time dominates over data access time.
- Any faster if input data uploaded?
  - Uploaded files to the container local disk before processing
    - Fairly fast upload with xrdcp (upload faster close to CERN)
  - But:
    - No speed-up for the processing time (even slightly slower from local files)
    - Explored differences with file:<filepath> (normal local file access) and root://<xrootdserver>/<filepath> (local xrootd server in the container): no significant difference
    - xrootd server version on eospublic more recent that in the container
- No significant gain.

Processing time: streaming vs local

Compare:

same 4 jobs

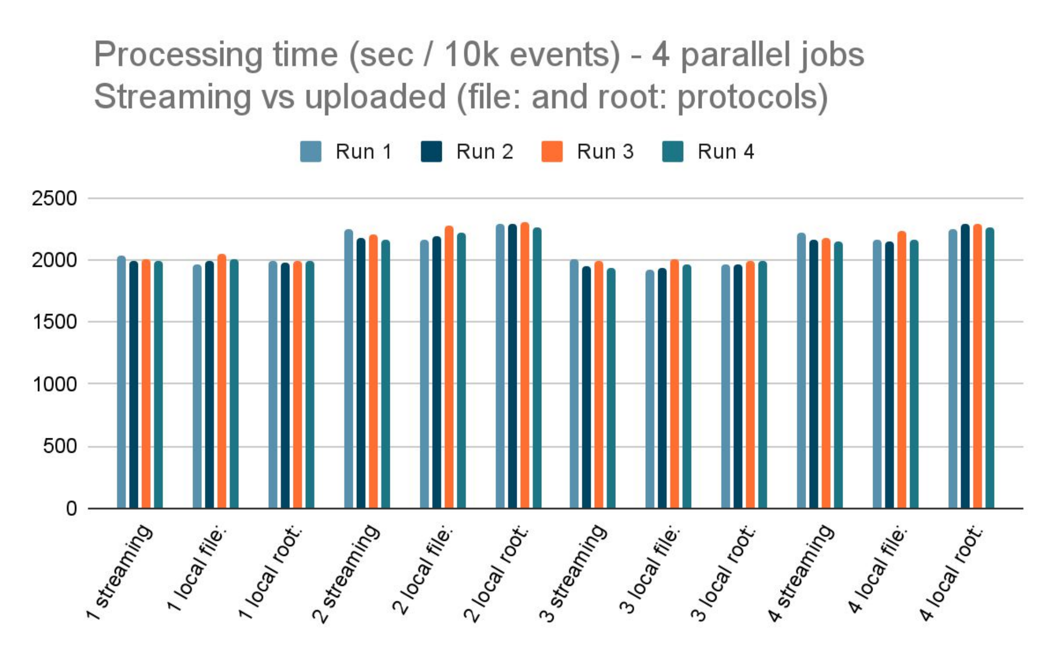
several times (Run 1–4)

– streaming

– local (read with file;)

– local (read with root:)

No gain observed.



Container image access

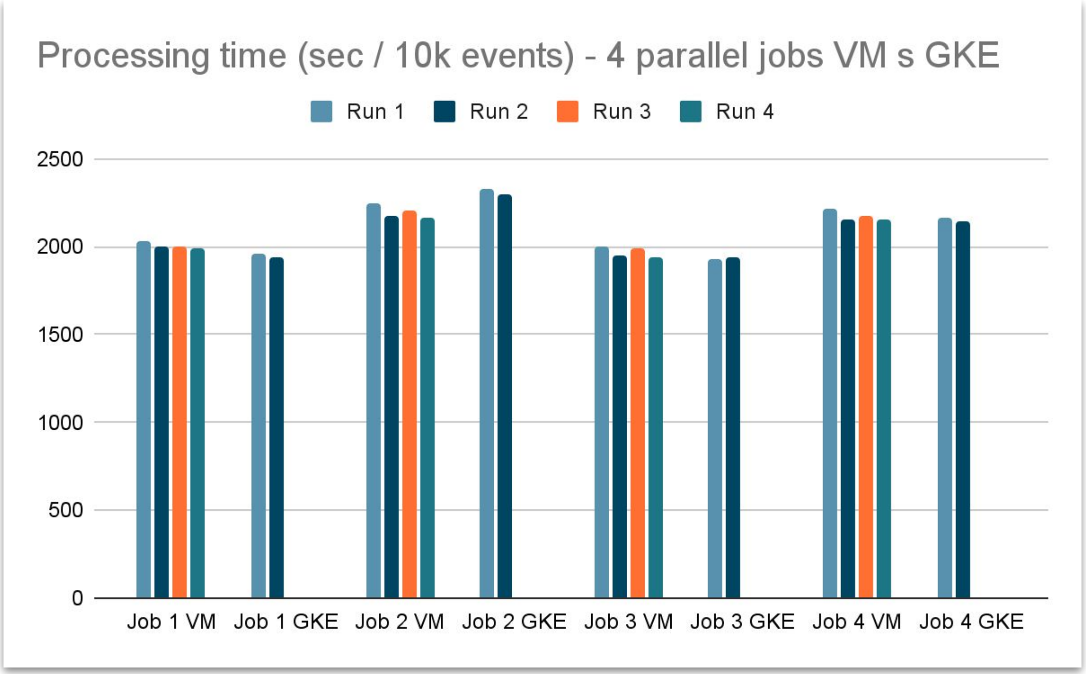
- CMSSW container image is big
  - Initial pull can take 30 mins.
  - Once pulled on the node, it is available to all pods.
  - Run a start job to pull the image to each node.
- Is there a better solution?
  - Uploading image to Google artifact registry and accessing from there is not significantly faster.
  - Use a new GCP feature: a secondary boot disk with container images preloaded:
    - Build a disk (tools exist), enable image streaming and define the disk in the node pool configuration so that it uses this secondary disk.
    - Immediate start of the jobs🚀!

Kubernetes / argo overhead

NAMESPACE	NAME	CPU(cores)	MEMORY(bytes)
argo	argo-server-5f7b589d6f-6jqcm	1m	18Mi
argo	workflow-controller-864c88655d-gwjcr	4m	26Mi
gke-managed-cim	kube-state-metrics-0	1m	72Mi
gmp-system	collector-2j5dw	5m	139Mi
[...]			
gmp-system	gmp-operator-57874fcf58-8h8w4	2m	62Mi
kube-system	event-exporter-gke-78fb679b7b-fdcr8	1m	83Mi
kube-system	fluentbit-gke-28sjh	12m	109Mi
[...]			
kube-system	gke-metrics-agent-x7rft	7m	113Mi
kube-system	konnectivity-agent-5f967456fc-2mc7x	1m	40Mi
[...]			
kube-system	konnectivity-agent-autoscaler-897d4f648-t97gb	1m	39Mi
kube-system	kube-dns-5fc99b87cb-dgzqj	3m	166Mi
kube-system	kube-dns-5fc99b87cb-nb25q	2m	138Mi
kube-system	kube-dns-autoscaler-6f896b6968-crn2z	1m	39Mi
kube-system	kube-proxy-gke-cluster-4-cluster-4-9248283b-00xm	1m	15Mi
[...]			
kube-system	l7-default-backend-6697bb6dfd-swb9h	1m	10Mi
kube-system	metrics-server-v1.30.3-7c8f6576cd-bdc85	4m	78Mi
kube-system	pdcsi-node-2fv5q	5m	47Mi

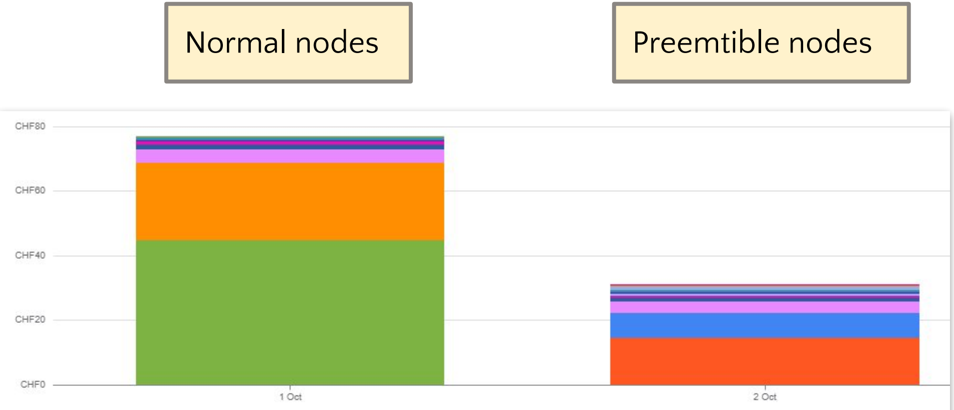
Compare:
- same jobs
- several times
- VM: no argo, no k8s
- vs GKE cluster

No significant overhead.



Preemtible / spot nodes

- Considerably cheaper
  - ¼ – ½ price
- Nodes can be deleted any time
- A trial with a 90-node e2-standard-4 cluster:
  - 13 / 90 nodes terminated
  - → 52 / 353 jobs failed
- Requires rerunning of the failed jobs.
  - The price advantage is worth some scripting for automated reruns.



SKU	Service	SKU ID	Usage	Cost
E2 Instance Core running in Netherlands	Compute Engine	012A-5DBB-1352	2,191.57 hour	CHF44.70
E2 Instance Ram running in Netherlands	Compute Engine	D9EA-4FF0-E394	8,766.35 gibibyte hour	CHF23.96
Spot Preemptible E2 Instance Core running in Netherlands	Compute Engine	D55E-4C77-A49B	1,779.48 hour	CHF14.51
Spot Preemptible E2 Instance Ram running in Netherlands	Compute Engine	7A8F-09E3-523C	7,118.05 gibibyte hour	CHF7.76