

Reperforming a Nobel Prize discovery on Kubernetes

Lukas Heinrich, Ricardo Rocha





ATLAS and CMS discovered the Higgs boson in 2012.

Since then the LHC has released some of their data publicly.

In this presentation we'll try to reproduce one of the main results from CMS Open Data using modern cloud tools.

11/22/2013 5:55:18 p.m.

Running jobs: 244151
Transfer rate: 40.08 GiB/sec



US Dept of State Geographer
© 2013 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image Landsat

Google earth

Fecha de las imágenes: 4/10/2013 66°43'26,18" N 8°52'37,10" O alt. ojo 16085.50 km

Kubernetes

Spun out of Google as an open source
container orchestration project

Built on lessons from Borg and Omega



Borg, Omega, and Kubernetes

BRENDAN BURNS,
BRIAN GRANT,
DAVID OPPENHEIMER,
ERIC BREWER, AND
JOHN WILKES,
GOOGLE INC.

Though widespread interest in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than ten years and built three different container-management systems in that time. Each system was heavily

**LESSONS
LEARNED FROM
THREE CONTAINER-
MANAGEMENT
SYSTEMS OVER
A DECADE**

Loosely coupled collection of components to deploy, maintain and scale workloads

Declarative, Load Balancing, Self Healing, Auto Scaling

Service and Batch Workloads

Kubernetes

Largest open source project after kernel

35.000 contributors, **148.000** code commits

83.000 pull requests, **1.1M** contributions

2000+ contributing companies

Google, RedHat, VMware, Huawei, Microsoft, IBM, Fujitsu, ...

Open community welcome to contributions

Special Interest Groups (SIGs) : Auto-Scaling, Multi-Cluster, Scheduling, ...



Borg, Omega, and Kubernetes

LESSONS LEARNED FROM THREE CONTAINER- MANAGEMENT SYSTEMS OVER A DECADE

BRENDAN BURNS,
BRIAN GRANT,
DAVID OPPENHEIMER,
ERIC BREWER, AND
JOHN WILKES,
GOOGLE INC.

Though widespread interest in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than ten years and built three different container-management systems in that time. Each system was heavily

Kubernetes

Lingua franca of the cloud

Managed services offered by all major public clouds

Multiple options for on-premise or self-managed deployments

Common declarative API for basic infrastructure : compute, storage, networking

Healthy ecosystem of tools offering extended functionality



OPENSIFT



Rancher
Kubernetes Engine



MAGNUM
an OpenStack Community Project



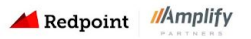
kubeadm





github.com/cncf/landscape

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.



Greyed logos are not open source

Kubernetes and containers at CERN

Started offering Mesos/DCOS, Swarm and Kubernetes

Now only Swarm and Kubernetes

Kubernetes by far the most popular solution

Spark as a Service, WebLogic, JIRA

INSPIRE-HEP, REANA/RECAST, Jupyter

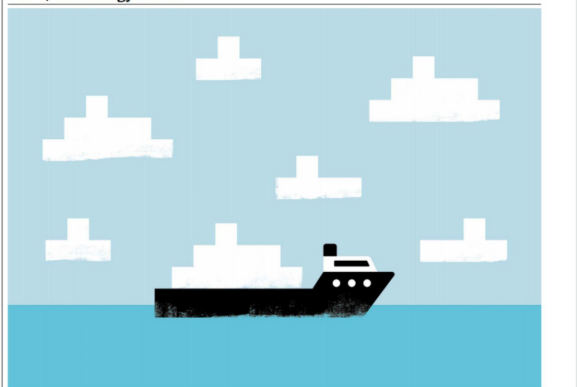
OpenStack, Batch / Condor

And many others

Clusters 468	Nodes 2988
Kubernetes 412	Swarm 47
Mesos 2	DCOS 7

Containers & Open in Science

Work / Technology & tools



CONTAINERS IN THE CLOUD

Standardized platforms allow researchers to run each other's software — no installation required. By Jeffrey M. Perkel

Murphy's law for the digital age: anything that can go wrong, will go wrong during a live demonstration. For Ben Marwick, that happened in front of a roomful of landscape architecture students in Berlin. The topic: computational reproducibility using Docker. Docker is a software tool that generates 'containers' — standardized computational environments that can be shared and reused. Containers ensure that computational analyses always run on the same underlying infrastructure, fostering reproducibility. Docker thereby insulates researchers from the challenges of installing and updating research software. However, it can be difficult to use. Marwick, an archaeologist at the University of Washington in Seattle, had become

proficient in migrating Docker configuration files ('Dockerfiles') from one project to the next, making minor tweaks and getting them to work. Colleagues in Germany invited him to teach their students how to follow suit. But because every student had a slightly different set of hardware and software installed, each one required a customized configuration. The demo "was a complete disaster", Marwick says. Today, a growing collection of services allows researchers to sidestep such confusion. Using these services — which include Binder, Code Ocean, Colaboratory, Gigantum and NextJournal — researchers can run code in the cloud without needing to install more software. They can click down their software configurations, migrate those environments from laptops to high-performance computing

clusters and share them with colleagues. Educators can create and share course materials with students, and journals can improve the reproducibility of results in published articles. It's never been easier to understand, evaluate, adopt and page the computational methods on which modern science depends. William Cono, a sleep researcher at Harvard Medical School in Boston, Massachusetts, spent weeks writing and debugging an algorithm, only to discover that a colleague's containerized code could have saved a lot of time. "I could have just gotten up and running, using all of the debugging work that he had already done, at the click of a button," he says. Scientific software often requires installing, navigating and troubleshooting a byzantine network of computational 'dependencies'

Work / Technology & tools

— the code libraries and tools on which each software module relies. Some have to be compiled from source code or configured just so, and an example of our code running on all five platforms can degenerate into a frustrating online odyssey through websites such as Stack Overflow and GitHub. "One of the hardest parts of reproducibility is getting your computer set up in exactly the same way as somebody else's computer is set up. That is just ridiculously difficult," says Kristin Whitaker, a neuroscientist at the Alan Turing Institute in London.

Easier evaluation

Docker reduces that to a single command. "Docker really provides reduced friction for that stage of the cycle of reproducing somebody else's work, in which you have to build the software from source and combine it with other external libraries," says Lorena Barba, a mechanical and aerospace engineer at George Washington University in Washington DC. "It facilitates that part, making it less error-prone, making it less onerous in researcher time." Barba's team does most of its work in Docker containers. But that is a computationally savvy research group; others might find the process daunting. A text-based 'command-line' application, Docker has dozens of options, and building a working Dockerfile can be an exercise in frustration.

That's where the cloud-based services come in. Binder is an open-source project that allows users to test-drive computational notebooks — documents such as Jupyter or R Markdown notebooks, which blend code, figures and text. Code Ocean, Gigantum and NextJournal (the latter three have free and paid tiers) let users write code in the cloud as well as, in some cases, bundle it with the data to be processed. These platforms also allow users to modify the code and apply it to other data sets, and provide version-control features for reviewing changes.

Such tools make it easier for researchers to evaluate their colleagues' work. "With Binder, you have taken that barrier [of software installation away], says Karthik Ram, a computational ecologist at the University of California, Berkeley. "If I can click that button and be dropped into a notebook where everything is installed, the environment is exactly the way you intended to be, then you may be more likely to take a look and give your feedback." Identifying required dependencies, and where to find them, varies with the platform. "Code Ocean and Gigantum, for example, have a one-click operation, whereas Binder requires a list of dependencies in a GitHub repository. Whitaker's advice: codify your computing environment as early as possible in a project, and stick with it. "If you try and do it at the end, then you are basically doing archaeology on your code, and it's really, really hard," she says. Ram developed a tool called Holopunch for

projects that use the statistical programming language R. Holopunch relieves the process of setting up Binder into four simple commands. (See <https://github.com/Gigantum/holopunch> on all five platforms at go.nature.com/2p9se1c.) The easiest way to try Binder is at mybinder.org, a free, albeit computationally limited, website. Or, for greater power and security, researchers can build private 'Binder hubs' instead. The Alan Turing Institute has two, including one called Hub23 (in reference to Hub 23 at the Second World War code-breaking facility at Bletchley Park, UK), that provides

"Researchers can be confident that their code will remain usable, whichever platform they choose."

greater computational resources and the ability to work with data sets that cannot be publicly shared, Whitaker says. The Rango community, which promotes open, reproducible and scalable geoscience, built a dedicated BinderHub so that researchers can explore climate modelling and satellite data sets that can amount to tens of terabytes, says Joe Hamman, a computational hydroclimatologist at the National Center for Atmospheric Research in Boulder, Colorado. (Whitaker's team is building a tutorial on building a BinderHub at go.nature.com/3P9J2C.)

Languages and clouds

Google's Colaboratory is basically a cross between a Jupyter notebook and Google Docs, meaning users can share, comment on and jointly edit notebooks, which are stored on Google Drive. Users execute their code in the Google cloud — only the Python language is officially supported — on a standard central processing unit (CPU), a graphics processing unit (GPU) or a tensor processing unit (TPU), a specialized chip optimized for Google's TensorFlow deep-learning software. "You can open your notebook or someone else's notebook from GitHub, start playing around with it, and use your own GPU and CPU and GPU and CPU, and so on later," says Jake VanderPlas, a member of the Colaboratory team in Google in Seattle.

NextJournal supports notebooks written in Python, R, Julia, Bash and Clojure, with more languages in development. According to Martin Kavaler, chief executive of NextJournal, which is based in Berlin, the company has registered nearly 3,000 users since it launched the platform on 3 May. Code Ocean, a beta version of which launched last year, features a browser-based client that users can install on their own system or remotely, for cloud-based coding and execution in the Jupyter and RStudio coding environments. Cono and WhoSaidGigantum to

machine-learning algorithms in the Amazon cloud, says the service makes it easy for collaborators to do the ground running. "I can get my code running on all five platforms, and this cloud compute infrastructure to the training and learning," he explains. Then there's Code Ocean, which supports both notebooks and conventional scripts in Python, R, Julia, Matlab and C, among other languages. Several journals now use Code Ocean for publication and to promote computational reproducibility, including titles from Taylor & Francis, De Gruyter and SAGE. In 2018, Nature Technology, Nature Machine Intelligence and Nature Methods launched a pilot programme to use Code Ocean for peer review. Nature, Nature Protocols and BMC Bioinformatics subsequently joined the trial. More than 95 papers have now been included in the trial, according to Erika Patrana, editorial director of Nature Research's applied science and chemistry journals, and more than 20 of those have been published.

Felicy Allen, a computer scientist at the Wellcome Sanger Institute in Hinxton, UK, co-authored one study in that trial, which analysed the types of mutation that can arise from CRISPR-based gene editing. (F. Allen et al. *Nature Biotechnol.* 37, 647–72, 2019). She estimates that it took a week to get the Code Ocean environment working. "I was surprised to really like it," Allen says. "And I think it was really nice that it made an example that someone could just press 'go' on and would work."

Although some worry about the long-term viability of commercial container-computing services, researchers do have options. Simon Adar, chief executive of Code Ocean, notes that Code Ocean 'compute capsules' are archived by the CLOCKSS project, which preserves digital copies of online scientific literature. And Code Ocean, Gigantum and NextJournal allow Dockerfiles to be exported for use on other platforms. All of which means that researchers can be confident that their code will remain usable, whichever platform they choose.

Benjamin Habes Kains, a computational pharmacogenomics researcher at the Princess Margaret Cancer Centre in Toronto, Canada, also uses open-source tools and is open to critiques of an analysis he published in *Nature* (B. Habes Kains et al. *Nature* 504, 389–393, 2013). For him, Code Ocean provides a way to ensure his code can be used and evaluated by his team, peer reviewers and the broader scientific community. "It's not so much that an analysis is correct or not," he says. "Nothing is really fully correct in this world. However, I've more transparent about it, you always communicate the level of confidence of criticism. You have nothing to hide — everything is there."

Jeffrey M. Perkel is technology Nature.

nature physics

Corrected: Publisher Correction

Open is not enough

Xiaoli Chen¹, Srinje Dalmeier-Tiessen^{1*}, Robin Dasler^{1,2}, Sebastian Feger^{1,3}, Parnfios Fokianos^{1,4}, Jose Benito Gonzalez¹, Harri Hirvonsalo^{1,5,6}, Dinos Koussis¹, Artemis Lavasa¹, Salvatore Mele¹, Diego Rodríguez Rodríguez¹, Tibor Šimko^{1,7}, Tim Smith^{1,8}, Ana Trisovic^{1,9}, Anna Trzcinska¹, Ioannis Tsanaktsidis¹, Markus Zimmermann¹, Kyle Cranmer¹, Lukas Heinrich¹, Gordon Watts¹, Michael Hildreth¹, Lara Lloret Iglesias¹, Kati Lassila-Perini¹ and Sebastian Neubert^{1,10}

The solutions adopted by the high-energy physics community to foster reproducible research are examples of best practices that could be embraced more widely. This first experience suggests that reproducibility requires going beyond openness.

Open science and reproducible research have become pervasive goals across research communities, political circles and funding bodies¹. The understanding is that open and reproducible research practices enable scientific success, accelerating future progress and discoveries in any discipline. In the struggle to take concrete steps in pursuit of these aims there has been much discussion and awareness-raising, often accompanied by a push to make research products and scientific results open quickly. Although these are laudable and necessary first steps, they are not sufficient to bring about the transformation that would allow us to reap the benefits of open and reproducible research. It is time to move beyond the rhetoric and the trust in quick fixes and start designing and implementing tools to power a more profound change.

Our own experience from opening up vast volumes of data in that opens cannot simply be taken on as an afterthought at the end of the scientific endeavour. In addition, openness alone does not guarantee reproducibility or reusability, so it should not be pursued as a goal in itself. Focusing on data is also not enough: it needs to be accompanied by software, workflows and explanations, all of which need to be captured throughout the usual iterative and closed research lifecycle, ready for a timely open release with the results. Thus, a graphics processing unit (GPU) or a graphics processing unit (GPU) is not having the reuse of research results as a goal requires the adoption of new research practices during the data analysis process. Such practices need to be tailored to the needs of each given discipline with its particular research environment, culture and idiosyncrasies. Services and tools should be developed with the idea of meshing seamlessly with existing research procedures, encouraging the pursuit of reusability as a natural part of researchers' daily work (Fig. 1). In this way, the generated research products are more likely to be useful when shared openly.

In tackling the challenge of enabling reusable research, we present these ideas as our guiding light when putting changes into practice in our community — high-energy physics (HEP). Here, we illustrate our approach particularly through our work at CERN, and present our community's requirements and rationale. We hope that the explanation of our challenges and solutions will stimulate discussions around the practical implementation of work-

flows for reproducible and reusable research more widely in other scientific disciplines.

Approaching reproducibility and reuse in HEP

To set the stage for the rest of this piece, we first construct a more nuanced spectrum in pursuit of this place the various challenges facing HEP allowing us to better frame our ambitions and solutions. We choose to build on the descriptions introduced by Carol Goble² and Lorena A. Barba³ shown in Table 1. These concepts assume a research environment in which multiple labs have the equipment necessary to duplicate an experiment, which essentially makes the experiments portable. In the particle physics context, however, the immense cost and complexity of the experimental set-up essentially make the independent and complete replication of HEP experiments unfeasible and unhelpful. HEP experiments are set up with unique capabilities, often being the only facility or instrument of their kind in the world; they are also constantly being upgraded to satisfy requirements for higher energy precision and level of accuracy. The experiments at the Large Hadron Collider (LHC) are prominent examples. It is this uniqueness that makes the experimental data valuable for preservation so that it can be later reused with other measurements for comparison, confirmation or inspiration.

Our considerations here really begin after gathering the data. This means that we are more concerned with repeating or verifying the computational analysis performed over a given dataset rather than with data collection. Therefore, in Table 2 we present a variation of these definitions that takes into account a research environment in which 'experimental set-up' refers to the implementation of a computational analysis of a defined dataset, and a 'lab' can be thought of as an experimental collaboration or an analysis group.

In the case of computational processes, physics analyses themselves are intrinsically complex due to the large data volume and algorithms involved. In addition, the analysts typically study more than one physics process and consider data collected under different running conditions. Although comprehensive documentation on the analysis methods is maintained, the complexity of the software implementations often hides minute but crucial details,

PERSPECTIVE

<https://doi.org/10.1038/s41567-018-0342-2>

OPEN

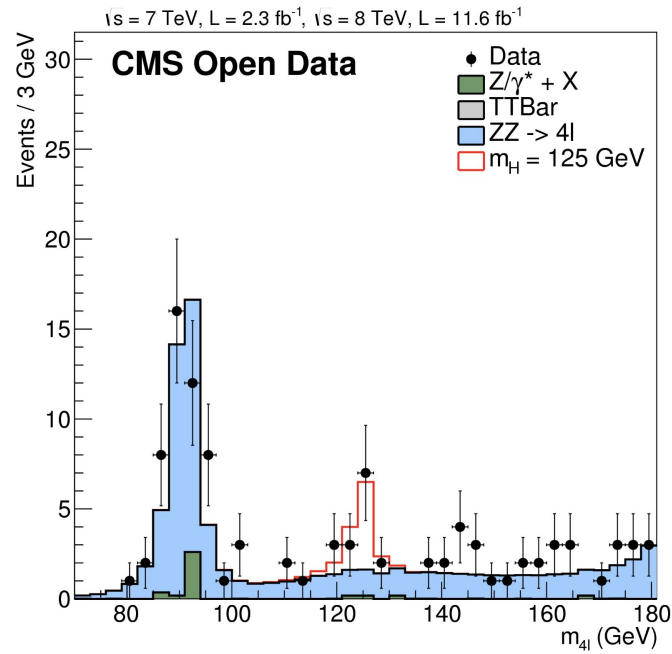
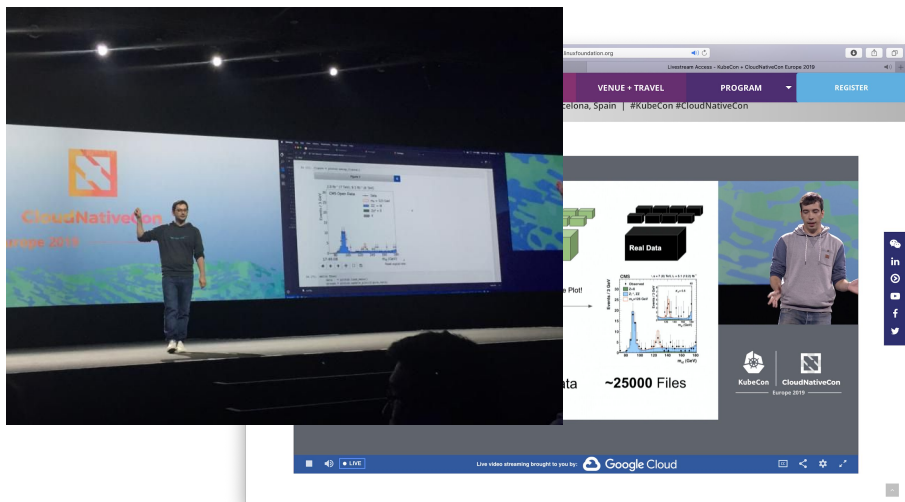
Challenge: $H \rightarrow 4l$ re-discovery on CMS Open Data

Benchmark analysis based on Open LHC Data.

Goal: Fit it within a live demo for 20-minute [Keynote at KubeCon EU 2019](#)

Learn something about cloud-native analysis, reproducibility, Open Data.

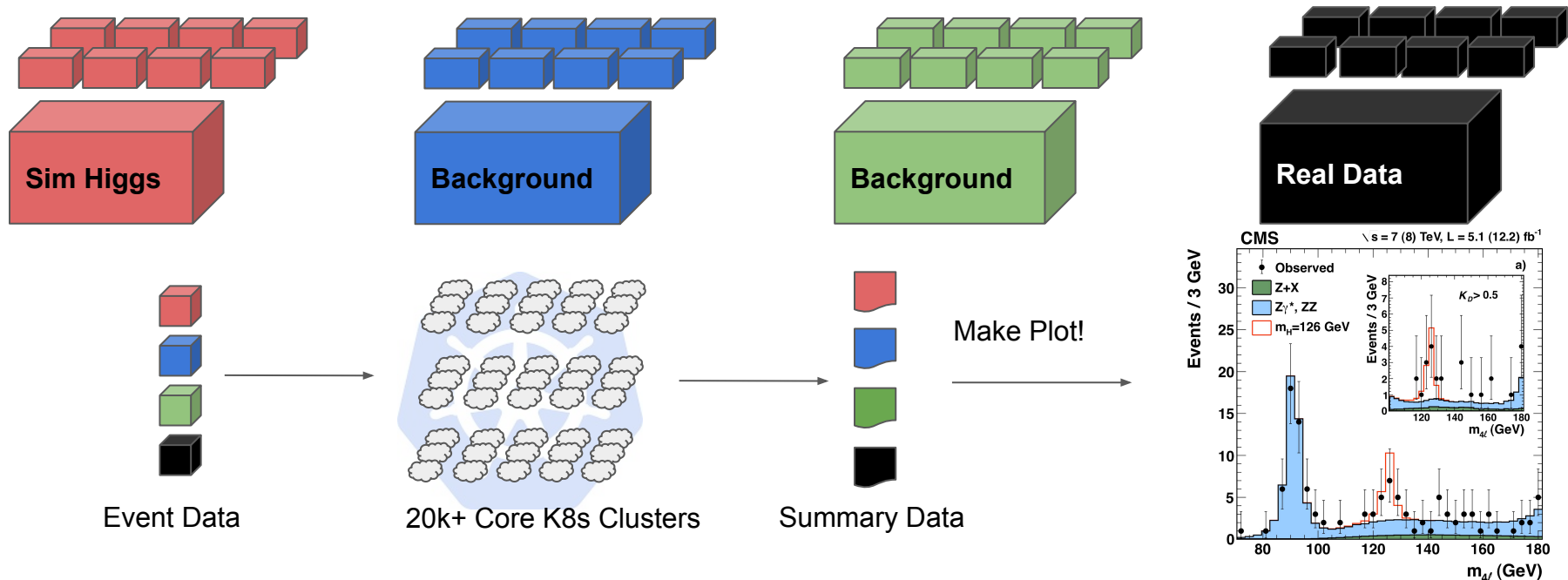
Have some Fun.



Demo

Challenge: $H \rightarrow 4l$ re-discovery on CMS Open Data

what would this look like in a cloud-native approach?



70 TB of Physics Data

~25000 Files



70 TB Dataset



OpenStack Magnum

25000 Kubernetes Jobs



Job Results

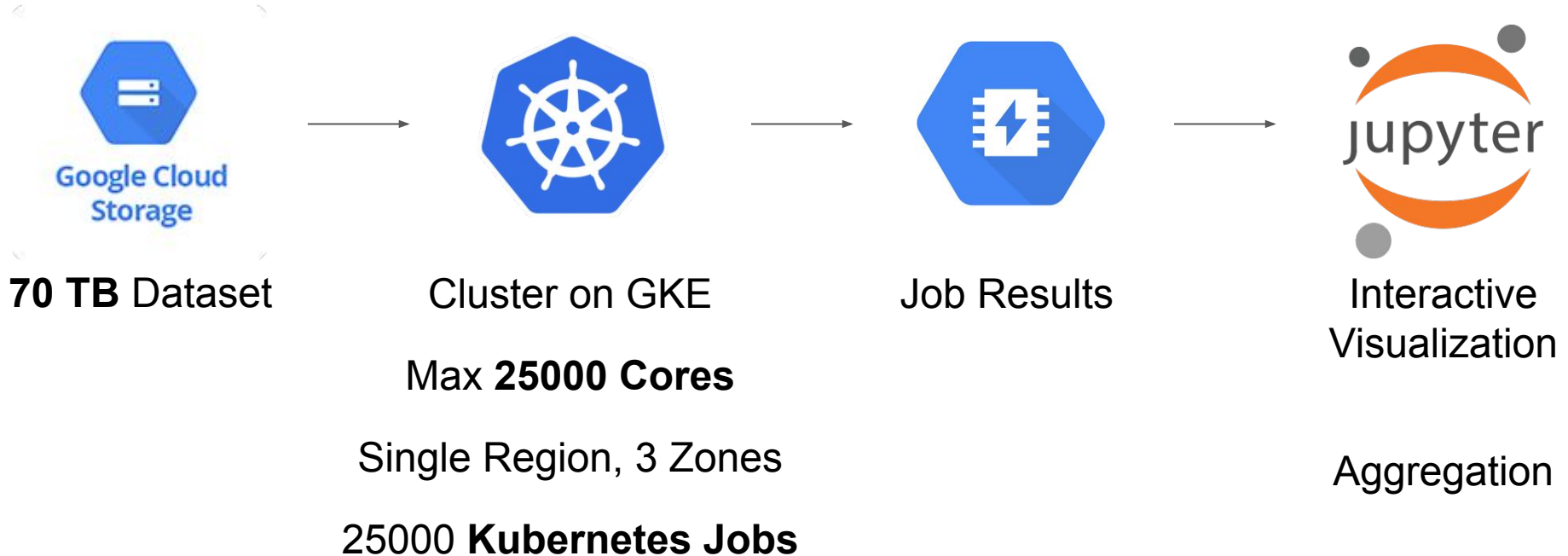


**Interactive
Visualization**

Aggregation

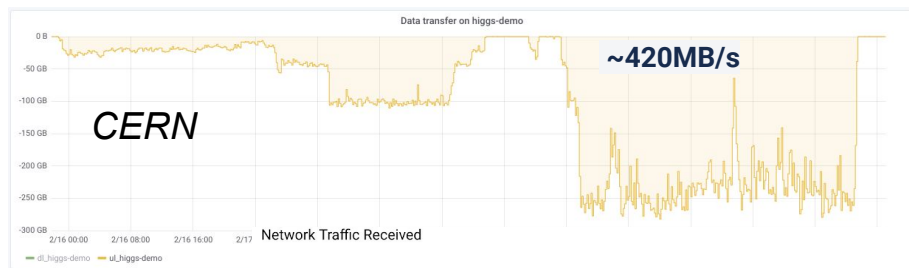


Google Cloud



Data Upload

Initial dataset (opendata) available on /eos

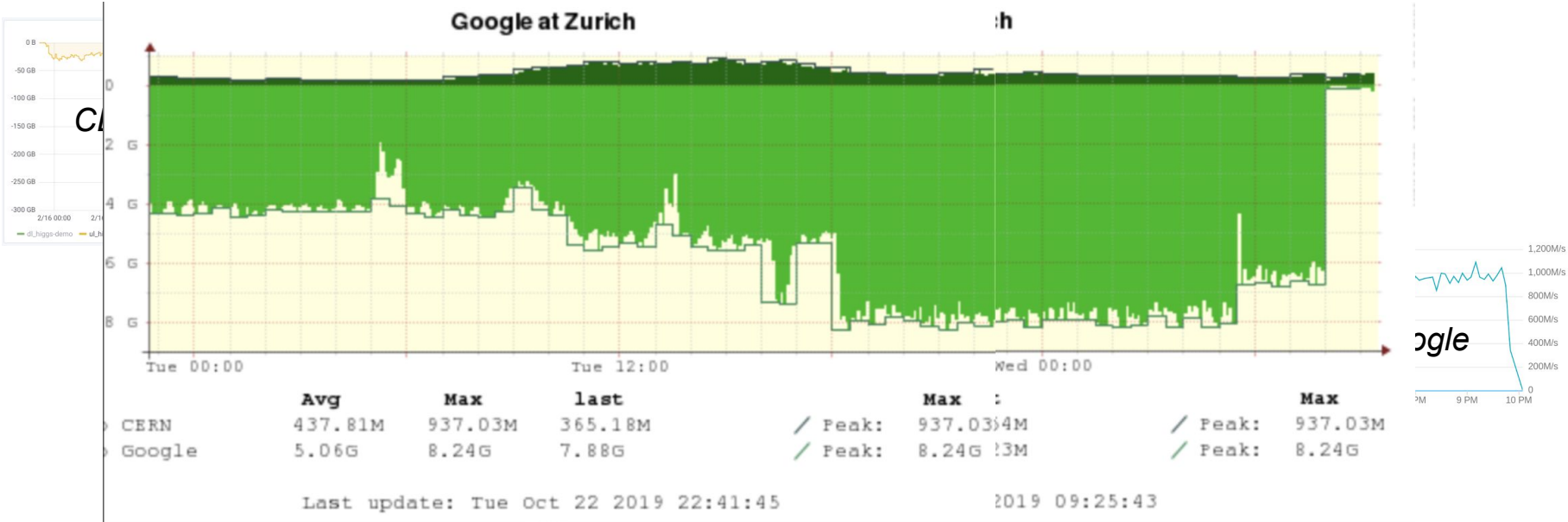


~1 day for full dataset transfer, done first to Zurich then to NL

Ingress is free, Ingress is free...

Data Upload

Initial dataset (opendata) available on /eos



Ingress is free, Ingress is free...

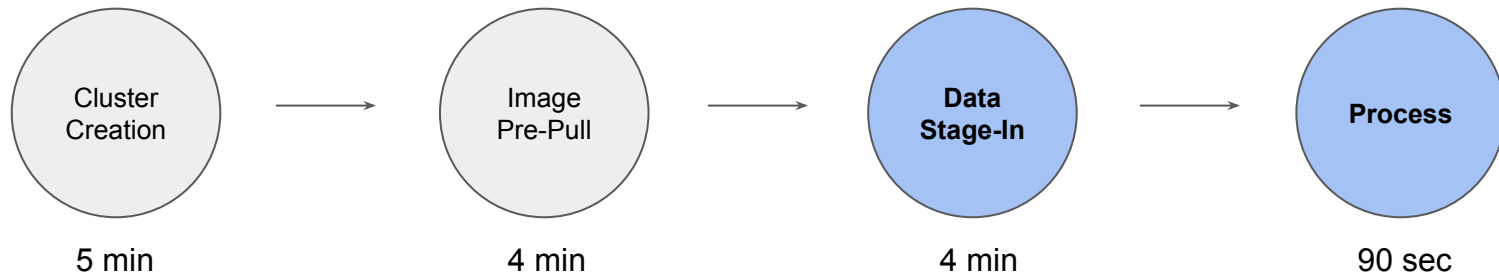
GCP Analysis Run

Kubernetes clusters on GKE (Managed Kubernetes service on GCP)

Today's run included

660 nodes: n1-highmem-16, 104 GB RAM

10560 cores, 69 TB RAM



GCP Analysis Run

Network guarantees 2Gb/core up to 16 core nodes (**32 Gbit per VM !**)

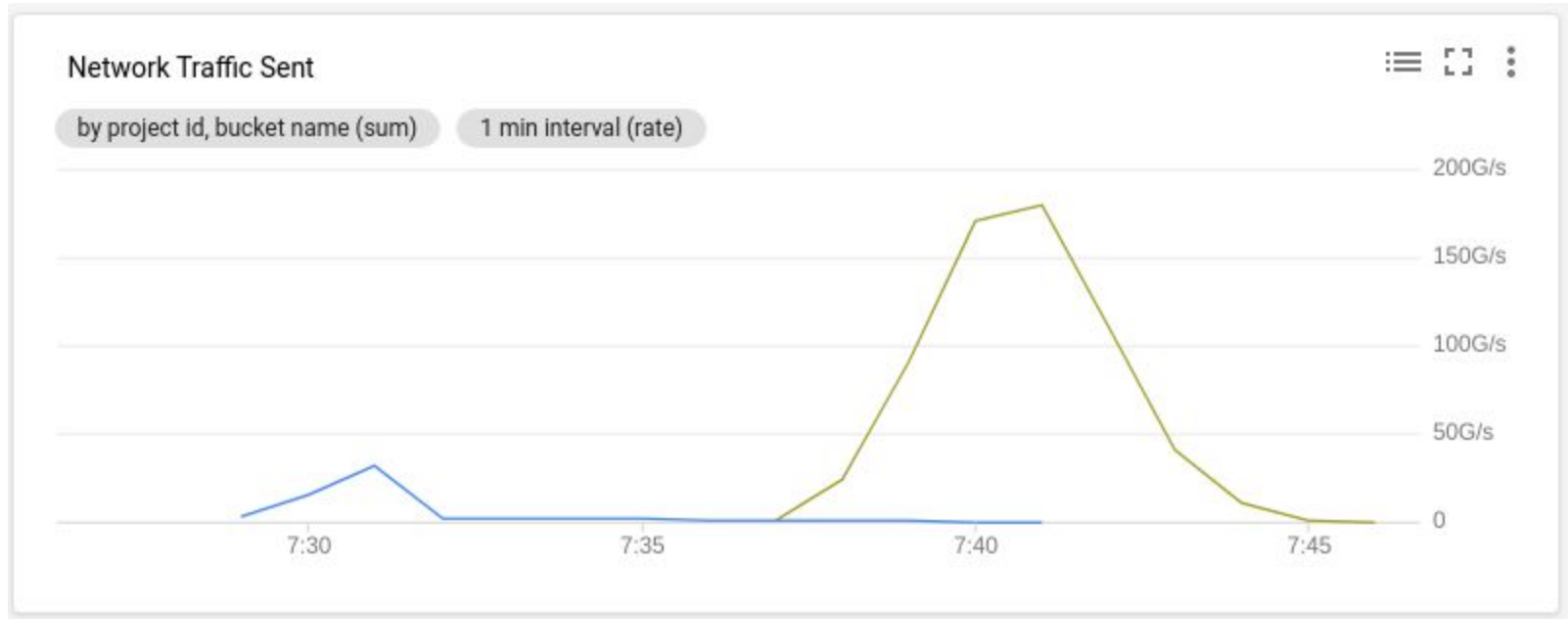
GCS can handle these rates somehow, and we end up bound by local i/o

Ended up using in-memory filesystems to go around this

	Zonal standard persistent disks	Regional persistent disks	Zonal SSD persistent disks	Regional SSD persistent disks	Local SSD (SCSI)	Local SSD (NVMe)
Maximum sustained IOPS						
Read IOPS per GB	0.75	0.75	30	30	266.7	453.3
Write IOPS per GB	1.5	1.5	30	30	186.7	240
Read IOPS per instance	3,000	3,000	15,000 - 60,000*	15,000 - 60,000*	400,000	680,000
Write IOPS per instance	15,000	15,000	15,000 - 30,000*	15,000 - 30,000*	280,000	360,000

GCP Analysis Run

Network guarantees 2Gb/core up to 16 core nodes (**32 Gbit per VM !**)



GCP Pricing

Billing is updated daily, though there are APIs to query for details

Considering a ~10 minutes run it implies (compute table prices, NL region)

$$\text{\$1.043} * 1530 / 6 = \text{\$260} \text{ (~5x cheaper if using pre-emptibles)}$$

Parking storage cost for the dataset (monthly cost, lots of room for creativity)

$$\text{\$0.020} * 70000 = \text{\$1400}$$

Total under \$300 usd

Running on credits, **no Committed Use or Sustained Compute discounts**

Open Questions

A stunt... or could we come up with a usable model?

Technically feasible. What do these technologies imply for LHC computing?

Analysis Models,
Infrastructure,
Funding Models,
....



Opportunities for Infrastructure

Simplified deployments (Federation), common APIs

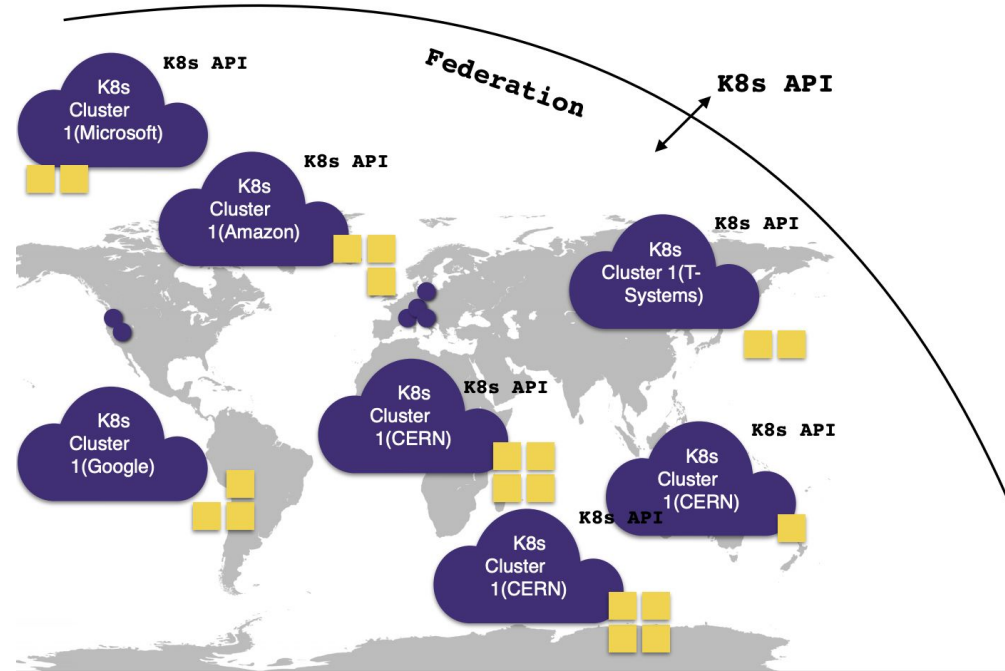
Bursting Scale-out to near-arbitrary scales

Auto-scaling

Access to special hardware

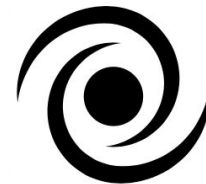
(FPGAs, TPUs, ...) easily

Integrated into LHC computing



Opportunities for Analysis Models

Rich gateway to scale-out, adaptive, on-demand computing



out-of-core dataframes

- Many systems natively integrate w/ k8s
- Rich real-time monitoring

How do you move smoothly
Between real-time analysis
and batch/scheduled work?

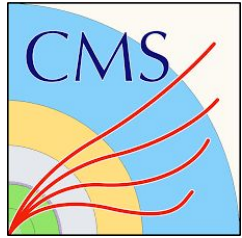
- k8s supports both well

The image is a collage of three screenshots related to data analysis and DASK:

- Top Left:** A JupyterLab interface showing Python code for generating data and performing computations. The code includes `import dask.array as da` and `z = (da.arcsin(x) + da.arcsin(y)).sum(axis=(1, 2))`. The output shows a large array of values.
- Top Right:** A DASK Profile dashboard showing a task graph with various colored nodes and a system metrics graph. The dashboard includes tabs for Status, Workers, Tasks, System, Profile, Graph, and Info.
- Bottom:** A Twitter post from @eggie5 with the text: "Multiple @SnorkelML labeling functions on 10s of millions of data points. This is 1 C4 @awscloud machine. A properly tuned #dask is a dangerous thing...". Below the tweet is a screenshot of a DASK dashboard showing a task stream and system metrics.

Open Data accessible to everyone at scale

LHC experiments part of growing list of experiments with complex open data problems: data complexity, data volume. large collaborations.



planck



ICECUBE
SOUTH POLE NEUTRINO OBSERVATORY



Open Data only useful, if it is feasible for external researchers to analyze it .
Demo goal: Show that public cloud can provide required scale on-demand.

```
[16:01:21] cmsusr@e6f7bea2253e /Users/lukasheinrich/Code/awesomedemo/higgs-demo/CMSSW_5_3_32/src $ \root -b
```

```
*****  
*                                     *  
*      W E L C O M E  to  R O O T      *  
*                                     *  
*   Version   5.32/00   2 December 2011 *  
*                                     *  
* You are welcome to visit our Web site *  
*      http://root.cern.ch              *  
*                                     *  
*****
```

```
ROOT 5.32/00 (branches/v5-32-00-patches@42372, Jun 10 2014, 18:26:00 on linuxx8664gcc)
```

```
CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
```

Moving to Cloud-based technologies:
analysis preservation as a by-product

Beyond a VM: Containerized CMSSW
~decade old software to reproduce results



cmsopendata/cmssw_5_3_32 ☆

By [cmsopendata](#) • Updated 4 months ago

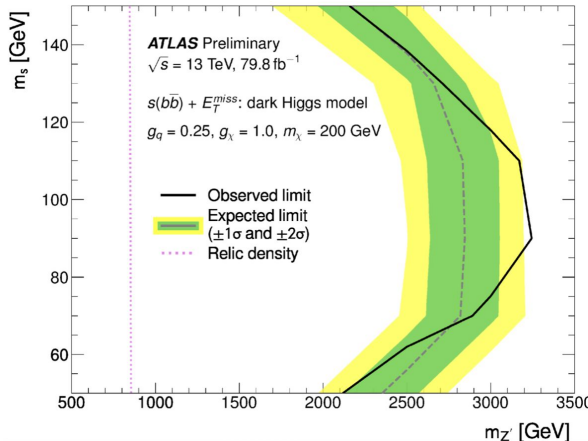
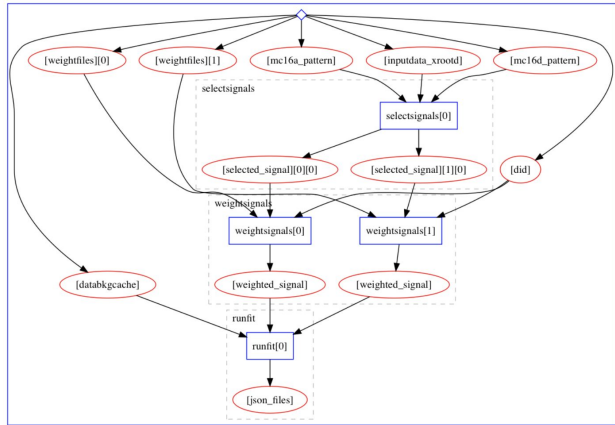
Container

Effective **re-use** of HEP analysis to generate new science results based on archived software.

Only possible through container-based workflows exposed to the user

reana


recast



Conclusions

Demonstrated Tbps analysis of CMS Open Data.

Modern cloud computing paradigms can give individuals scale to realistically analyze LHC data, foster reproducibility & reusability of LHC analyses.

Opens up exciting opportunities to evolve the LHC computing landscape as we look towards Run-4 / the HL-LHC era. Cross-team collaborations are crucial for R&D[*].

Thought expt: if we started today, what would our infrastructure look like?

We did learn a lot & had some fun.