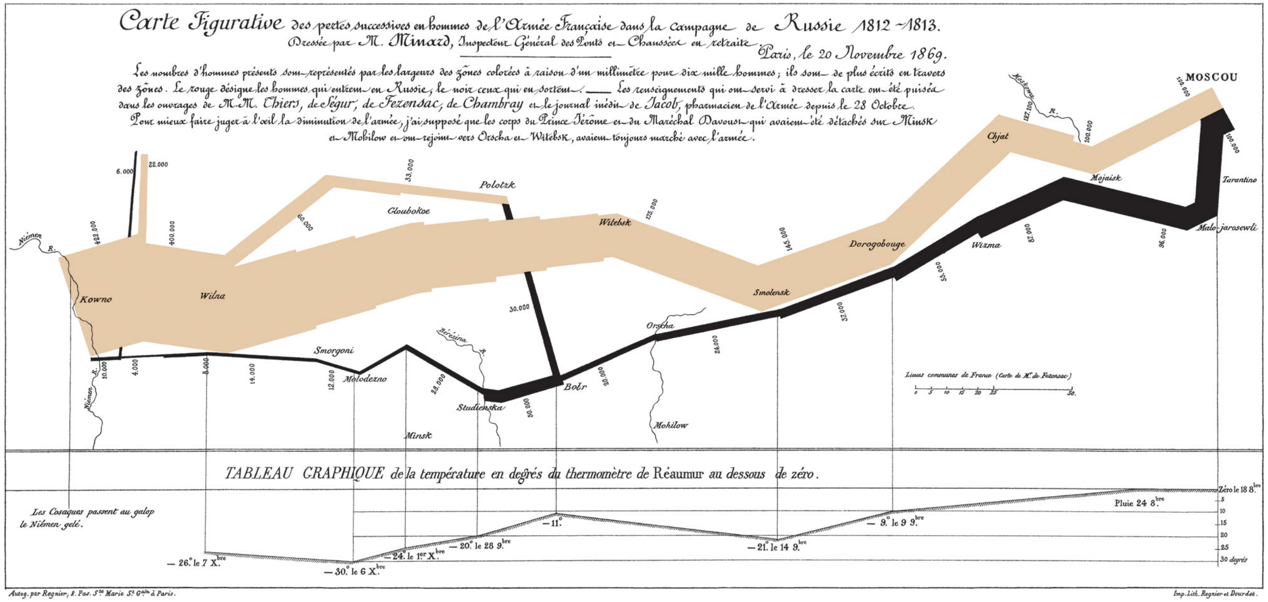
HPCs as Analysis Facilities

HEP Workloads:



production-like:

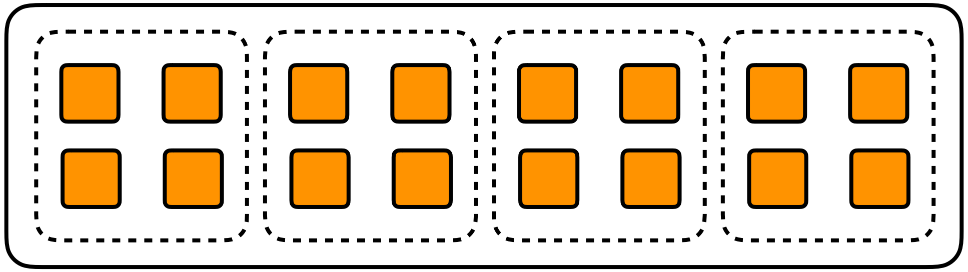
high throughput, ~stable software, schedulable. Experience on HPC.

analysis-like

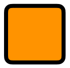
low(er) throughput, volatile software, fast turnaround / interactive (?):

How can we bring analysis onto HPCs?

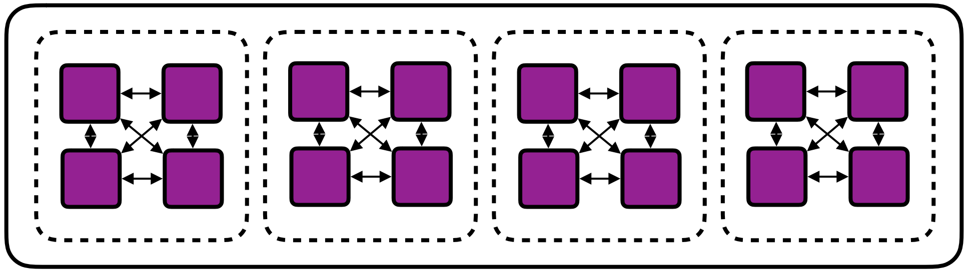
HPC Workloads



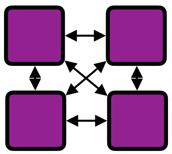
single-core, single-node



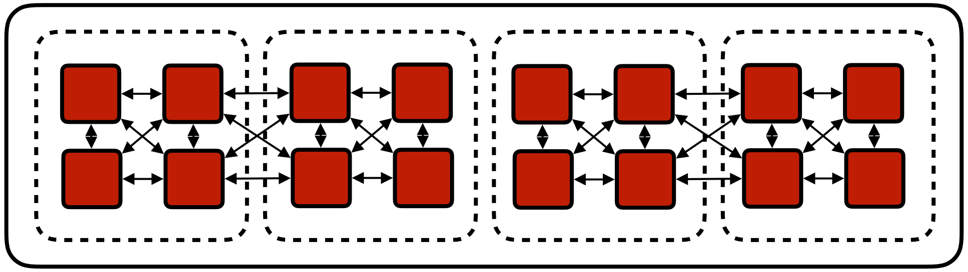
e.g. typical user job



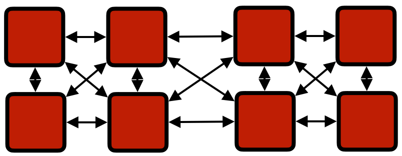
multi-core, single-node



e.g. simulation



multi-core, multi-node



e.g. OpenMPI

Challenges / Notes:

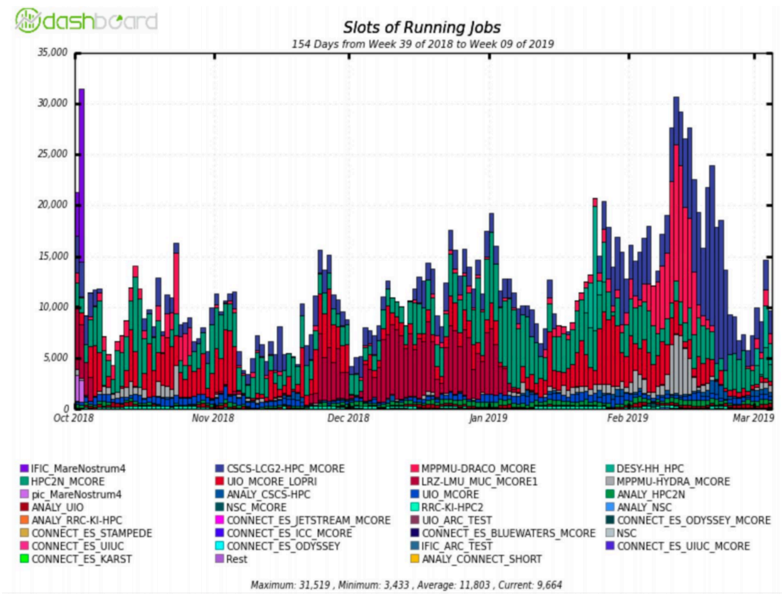
- Many traditional HEP workloads, including analyses are naturally embarassingly parallel.
  - Not classic HPC (i.e. OpenMPI) jobs.
- Usage patterns to HPCs differ from rest of distributed
  - user access:
    - direct access often restricted
    - requires integration into larger distributed compute infra
  - network access:
    - connectivity for software and data
- HPCs with hardware acceleration are coming online:
  - need to formulate workloads can make use of GPUs

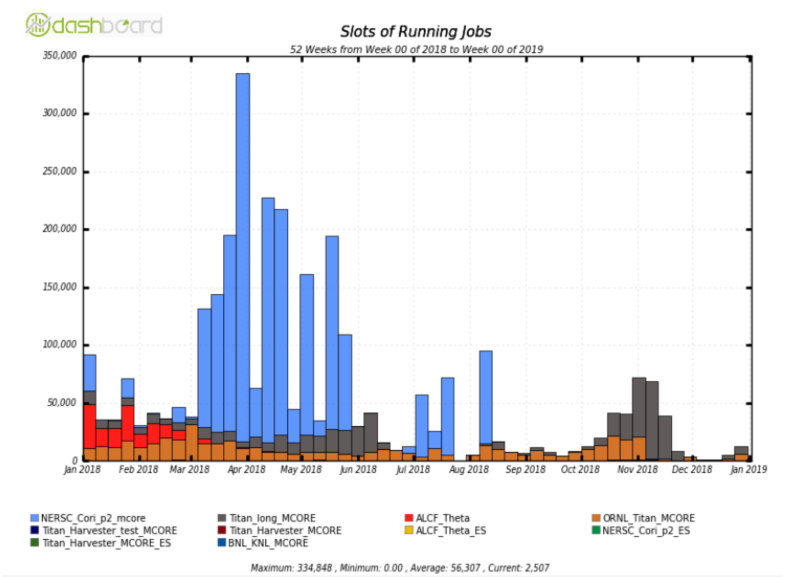
The Baseline: scheduling <u>existing</u> workloads into HPCs:

Harvester / Pilot2 provide the backbone for job submission to heterogeneous resources

- **Harvester**: Interfaces ATLAS distributed analysis infrastructure
- **Pilot2**: payload execution on host.

Expected to work well in the mid-term





Baseline - Analysis

Containerized Workloads:

- well-accepted now within HPC.
  - singularity, shifter, sarus
  - does not require network connectivity

- increasingly standards-compliant (OCI)
  - singularity 3, sarus
  - helps to **transparently move** workloads from user/cloud to HPC
- containerized job definition **integrated into Pilot**

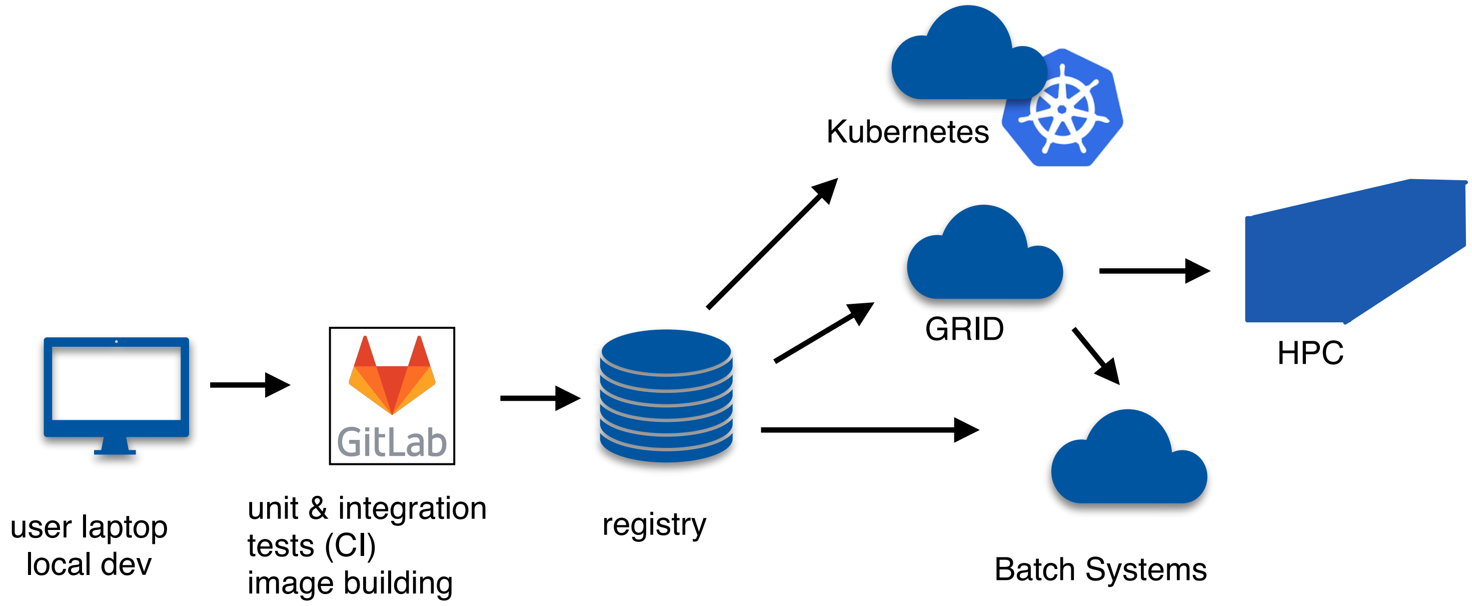
Solvable: Image distribution

- sync on HPC edge
- cvmfs
- crfs

Baseline - Analysis

Dedicated user cli (pcontainer) and transform (runcontainer)

Prodsys task parameters	
allowInputLAN	use
architecture	
cliParams	pcontainer --loadJson=/tmp/tmpvLR9nc --noBuild --containerImage='docker://alpine' --site='ANALY_MWT2_SL7' --outDS='user.aforti.test.20190318204327' --exec='echo 'Hello World''
cloud	US
countryGroup	uk



Baseline - Analysis with GPUs

also starting to integrate hardware accelerators

```
pcontainer --containerImage docker://... --architecture nvidia-gpu
```

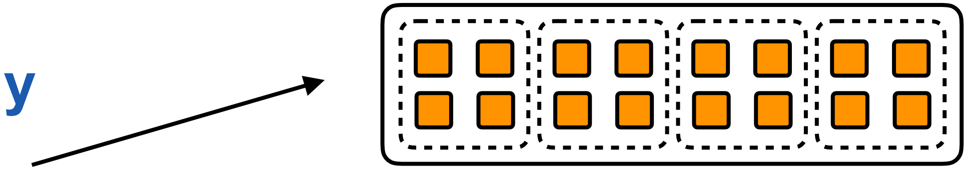
Addressing paradox that hardware accelerators are both plentiful (as a community) and scarce (as a user)

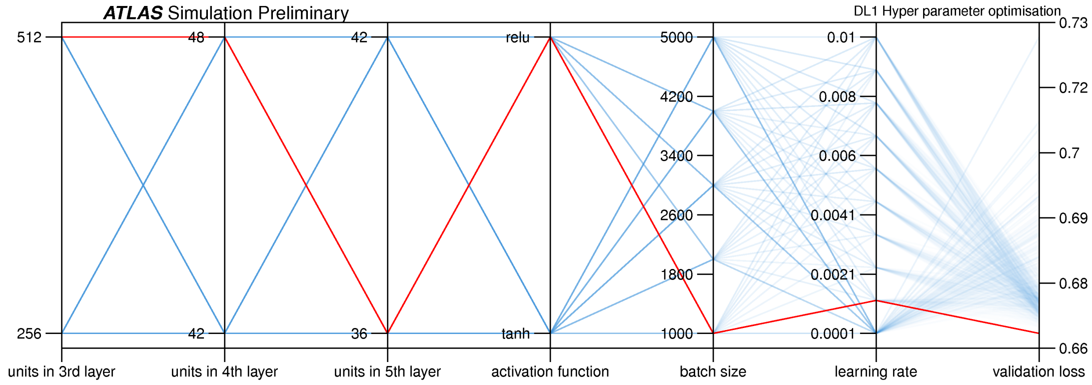
- **making GPUs more widely accessible through standard HEP interfaces (WMFS) essential for development and deployment of hardware accelerated workloads**

Working on porting / submitting Summit

Baseline - Analysis with GPUs

Obvious payload that already fits into current paradigm Hyper-Parameter Scans

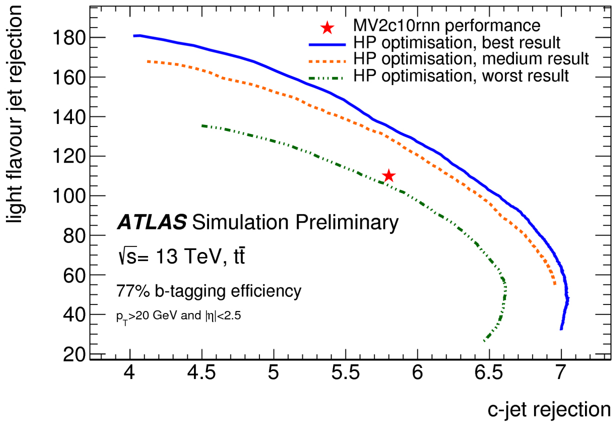


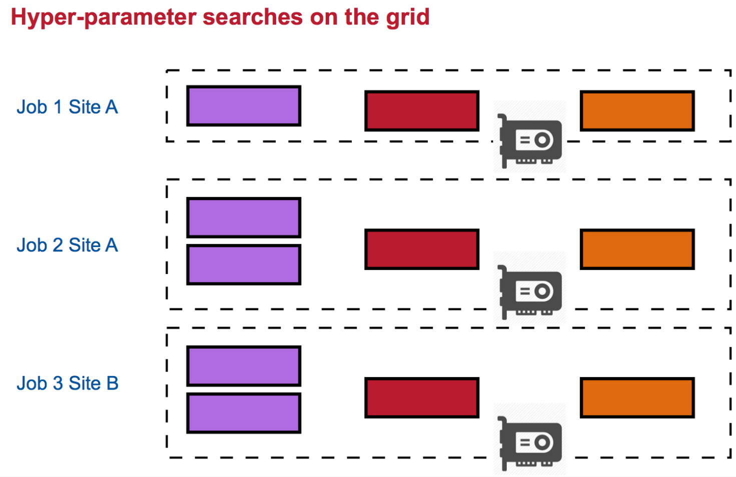


low-hanging fruit for both

- physics performance
- computing

Extends to networks w/ O(days) training time

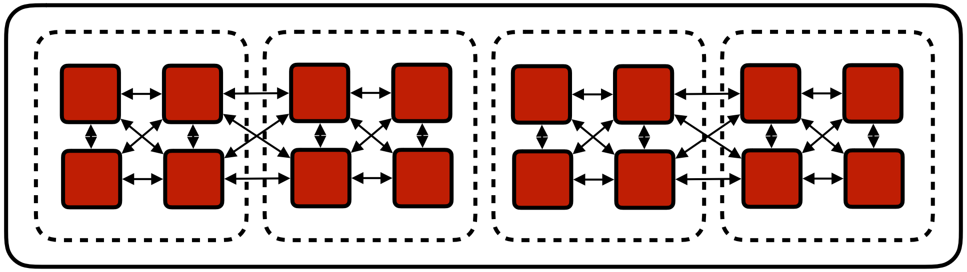




Beyond the Baseline:

Distributed Training: the obvious candidate to check all boxes

Applies to range of architectures CPU, GPU,.. in future perhaps emerging architectures (dataflow engines, spatial ..)



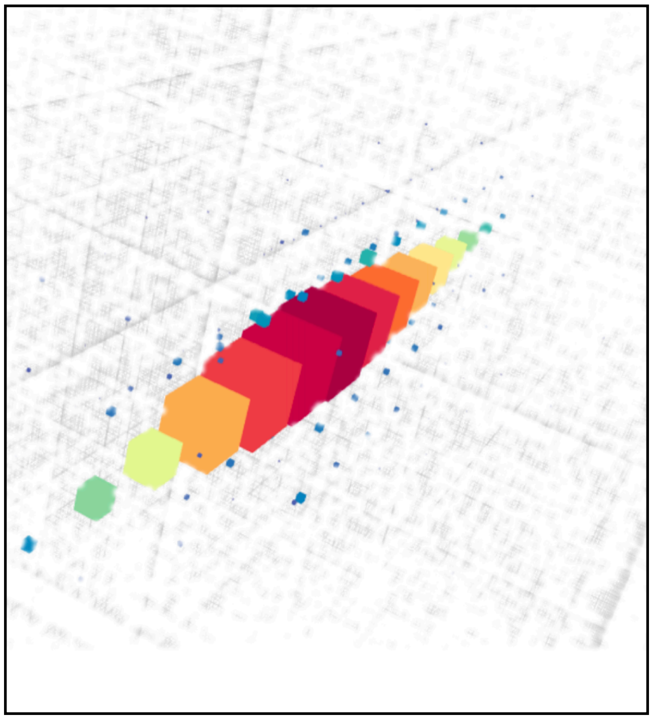
Various ML frameworks - some based on MPI

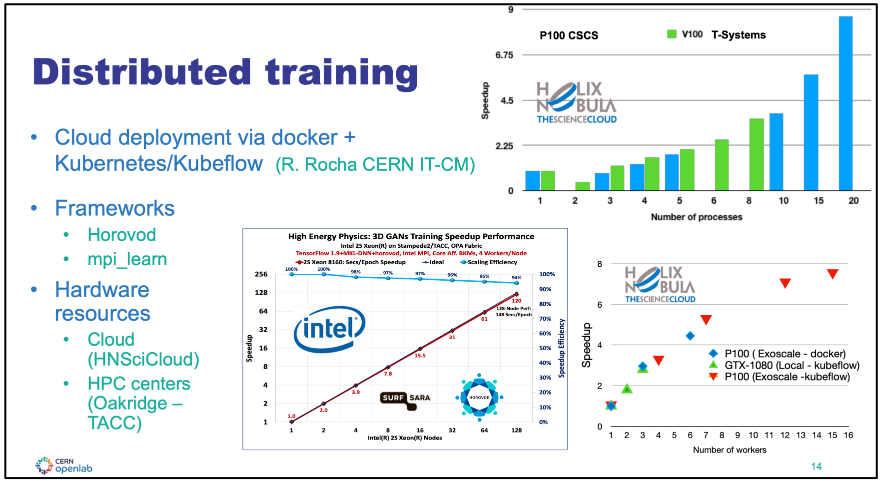
- Horovod (keras or torch)
- Distributed TF
- BigDL
- TF-Replicator
- PyTorch Distributed

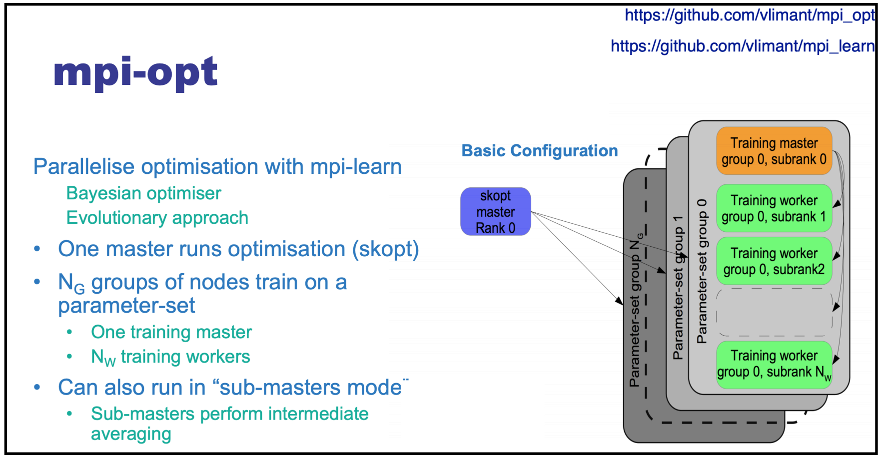
Example: Learning the Simulator

GANs: 3D convolutional GAN, ATLAS GAN, etc

Motivation: Fast Calo Simulation







slides: S. Vellecorsa ACAT19

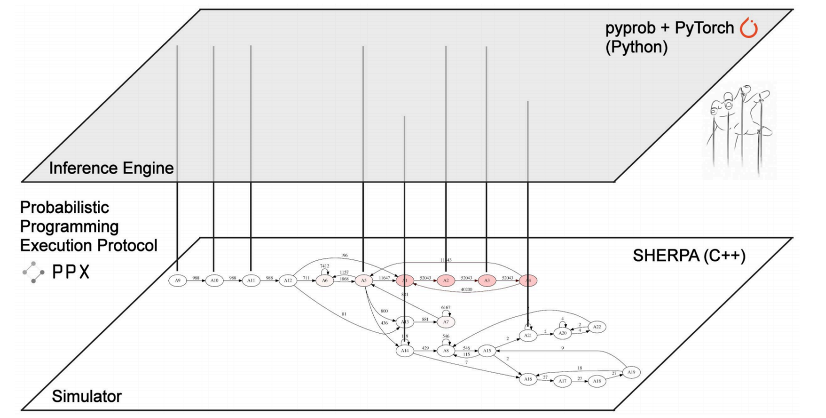
Example: Probabilistic Programming - using the Simulator

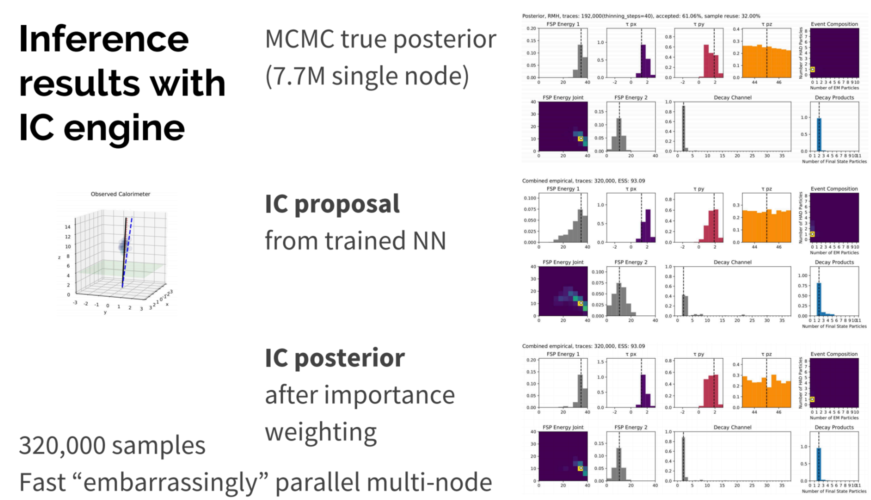
- ML-"steered" simulator to sample possible posterior interpretations of a given observation
- "universal inference"

Both paradigms:

- massively distributed training on HPCs (HSW @ Cori)
- embarassingly parallel inference to obtain posterior samples

PPX





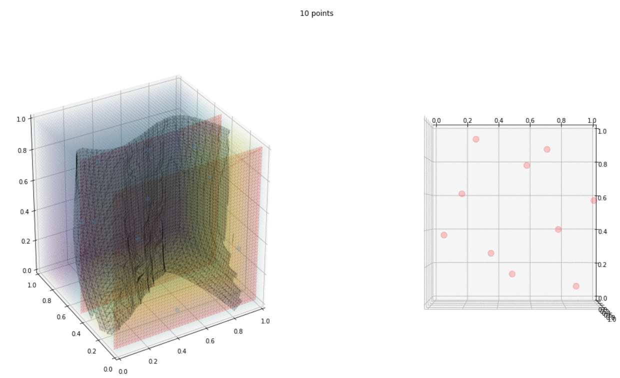
slides: G. Baydin ACAT19

Example: Active Learning

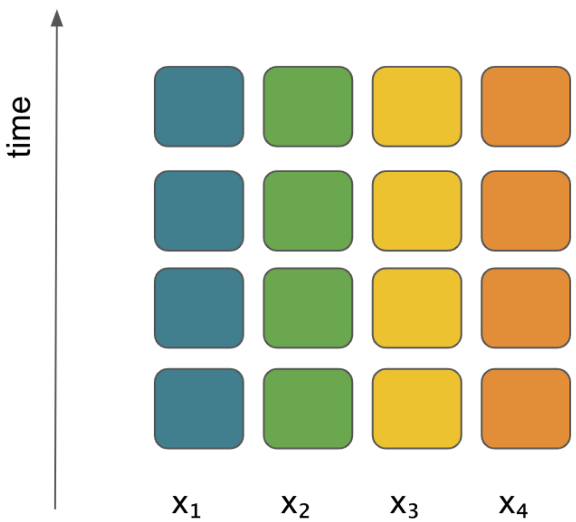
Active Learning is a good fit for parallelizable tasks and HPCs

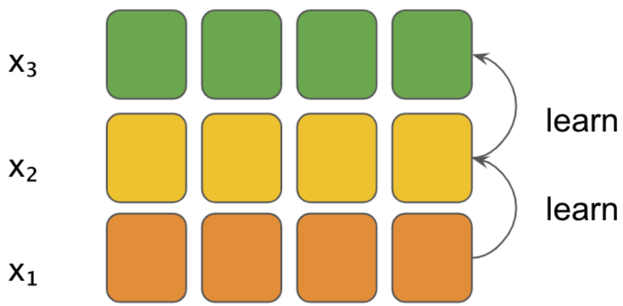
- parallelization neutralizes sequential nature of active learning
- overall reduction of used cycles

Example:
Optimizing BSM point generation



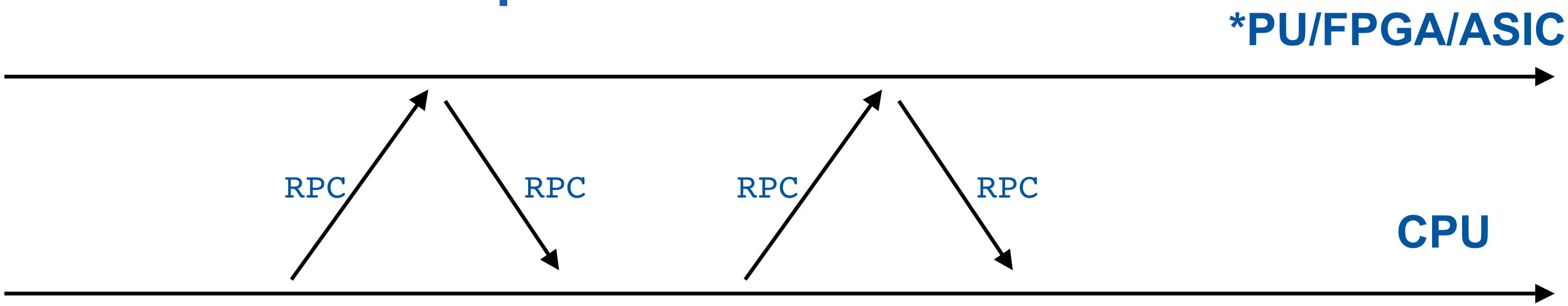
Animated ACAT Slides





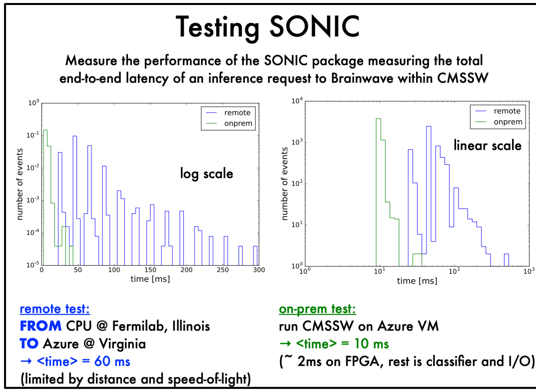
Example: Co-Processors

As ML applications become integrated into our main code-bases (reco but also analysis). May (?) be useful to offload inference to h/w-accelerated coprocessors.



Similarly for Training, preprocess data on CPU, train on GPU.

ACAT: Inference on FPGAs on-prem vs Cloud

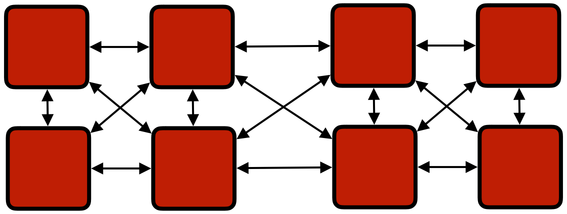


HPCs: possible advantage in fast links b/w CPU and coprocessors

slides: J. Ngadiuba ACAT19

Beyond MPI and ML:

Most workloads in HEP will not be MPI, but there are other distributed paradigms that might fit better.



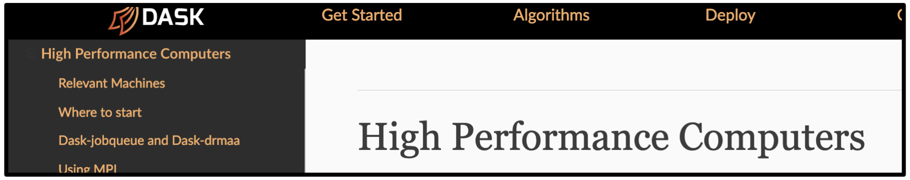
Generically, systems that gang-scheduled and communicate with each other

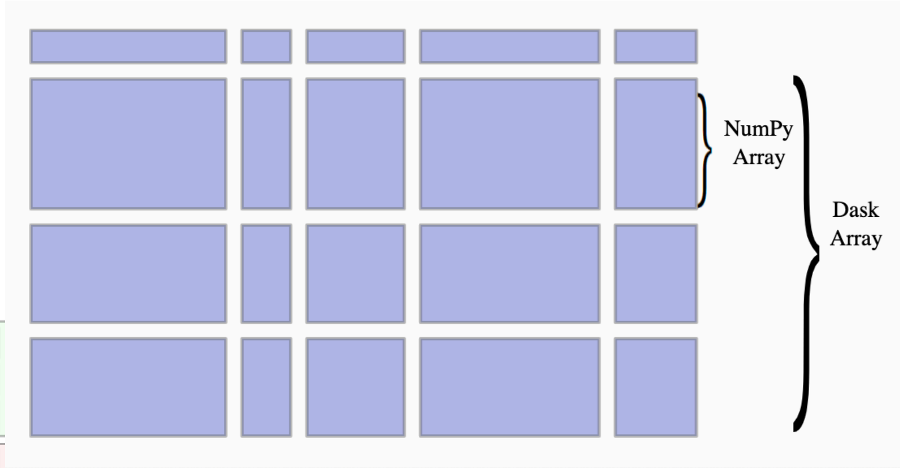
Out-of-Core Dataframes and Arrays:

Dataframe concept popular DS/DL, making its way into HEP.
Probably distributed analysis paragdim most aligned w/ analysis

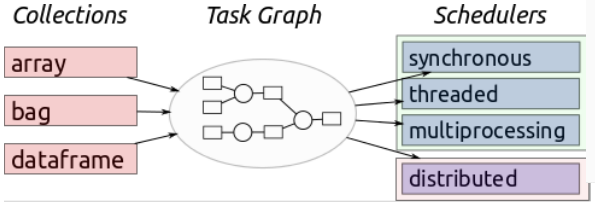
Multiple systems to handle TB-scale data-frames

- Dask
- Vaex
- Ray / Modin
- RDataFrame

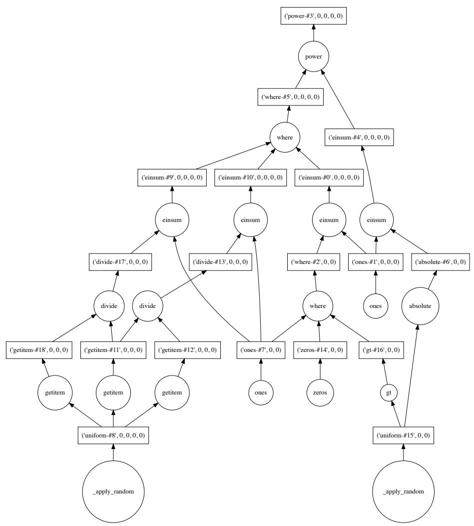


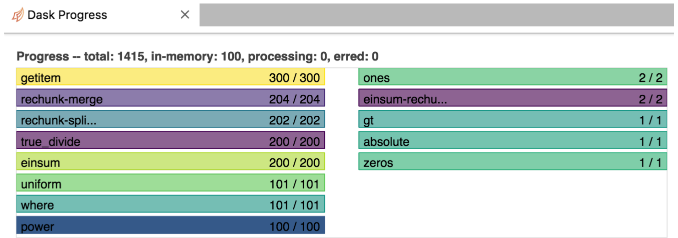


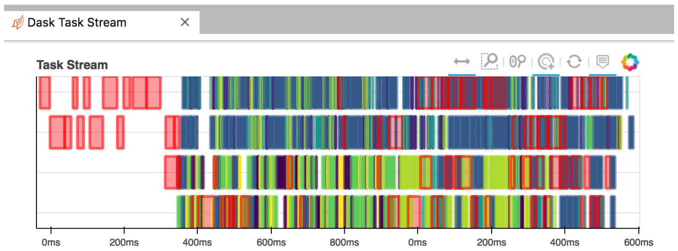
Some use MPI as a backend, but not the API exposed to users (**good!**)



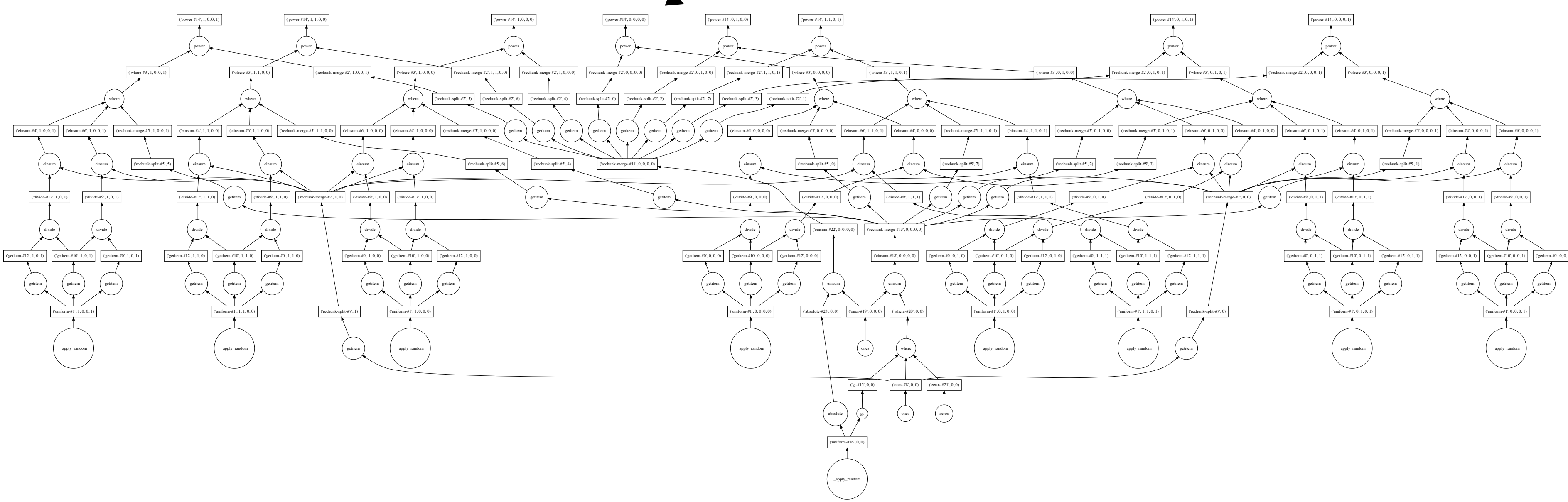
Example: Distributed Statistical Analysis using





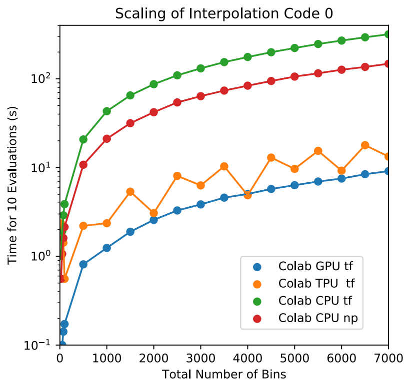


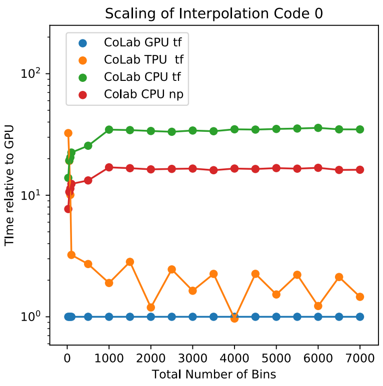
explode computational graph to multiple workers

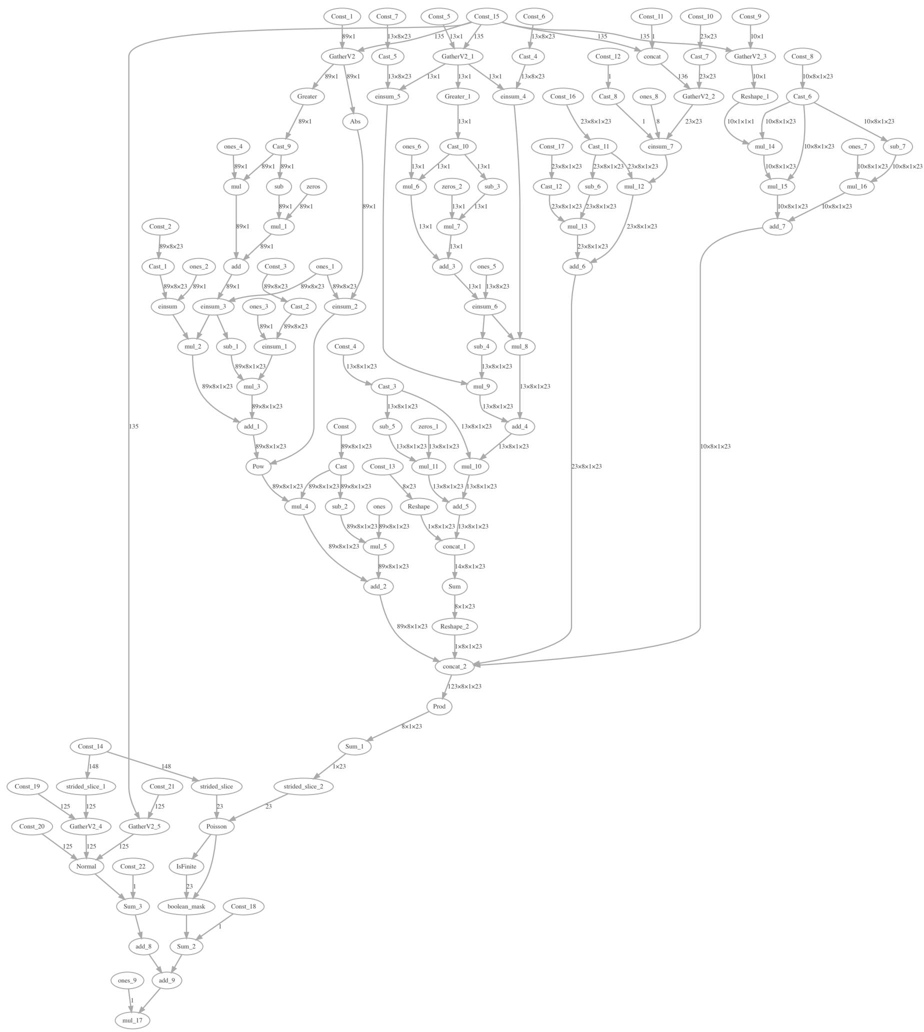


Example: Statistical Analysis on Hardware Accelerators

Standard HEP Fits (HistFactory) on GPUs and TPUs







Possible that once a computation is expressed within DL-type computational graphs, porting to new architectures not too hard.

Best of both worlds: Dask + GPUs

README.md

Dask GPU Dataframes

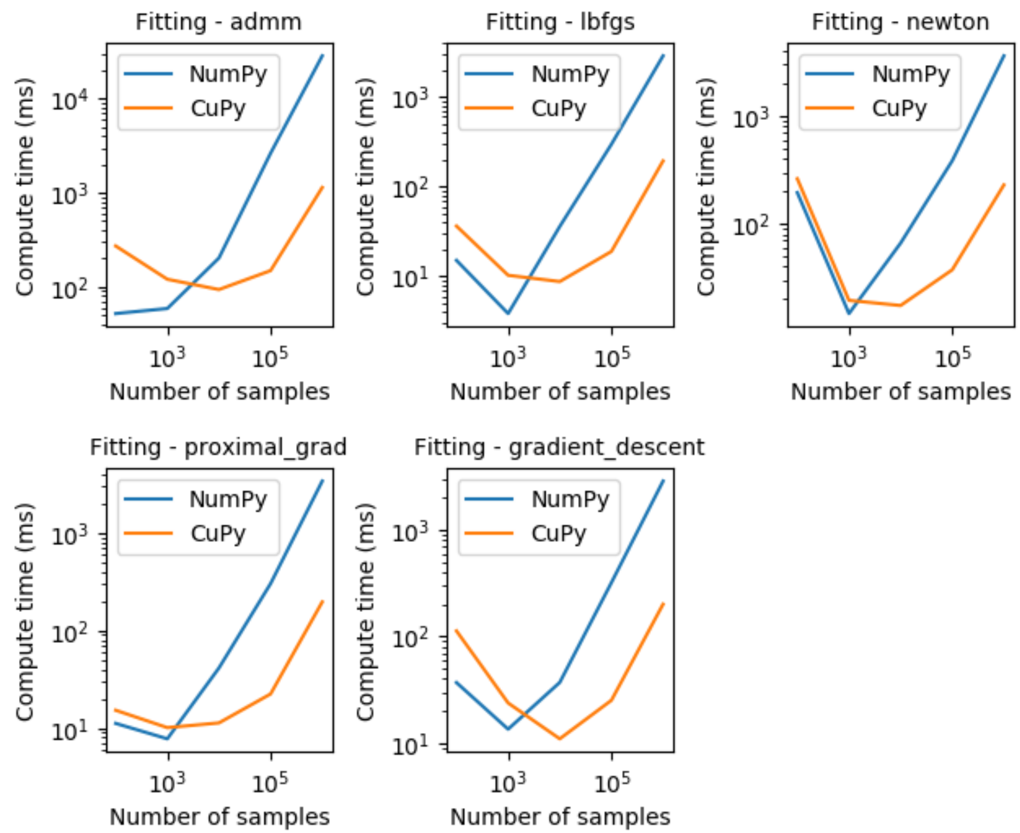
A partitioned gpu-backed dataframe, using Dask.

Setup from source

Setup from source repo:

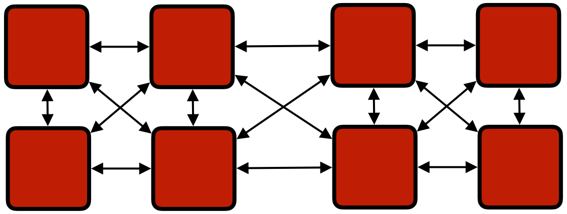
1. Install dependencies into a new conda environment

```
conda create -n dask-cudf \
   -c rapidsai -c numba -c conda-forge -c defaults \
   cudf dask cudatoolkit
```



good target for analysis on GPU-heavy HPCs?

Generic distributed system: Kubernetes



Brings thee things together we want on an HPC

- accelerator-aware workloads
- containers
- distributed processing beyond MPI
- UX parity to clouds: helps move workloads into HPC

use-cases: automated analysis pipelines (REANA, Parsl, yadage, luigi, snakemake, nextflow, etc...) on HPC

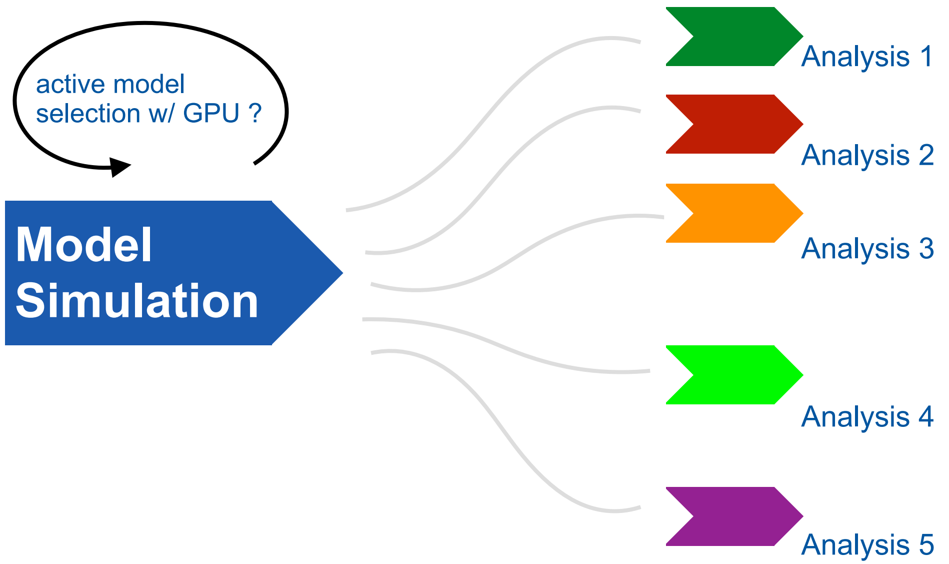
Example: Reinterpretations

conceivable to run full reinterpretation campaign tightly packed within HPC. E.g. (O(100M-1B) events)

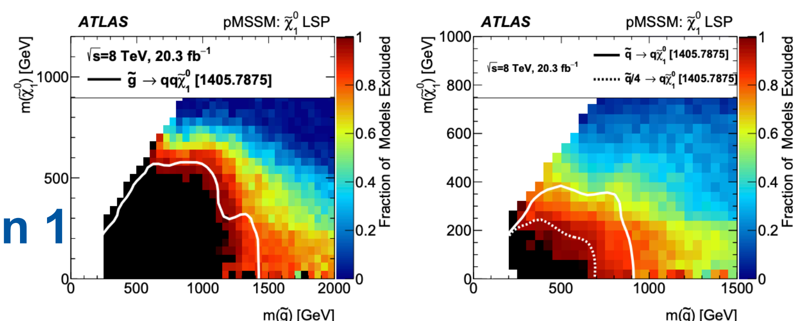
Related to "analysis train" idea.

- capture analysis as containerized computational graph
- events simulation / fast chain - go through standard HPC hooks
- analysis re-execution through containerized workflows on e.g. Kubernetes deployed on HPC

Exploits locality and scale of data and compute at HPC



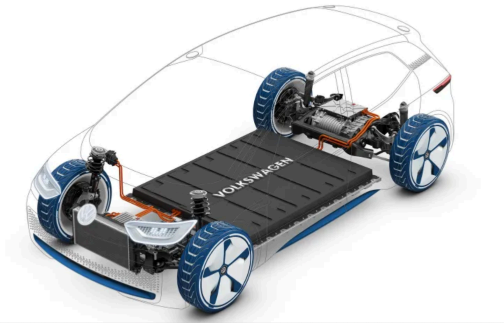
Run 1



Question: How to deploy distributed workloads through WMFS

- force MPI
  - not a natural API for analysis
- bespoke solutions per workload type
  - standard MPI Jobs
  - Dask, Ray, Vaex, Spark, etc..
  - Distributed Learning
- how to we describe distributed applications?

Perhaps optimal for some cases, but hard to scale to new analysis systems

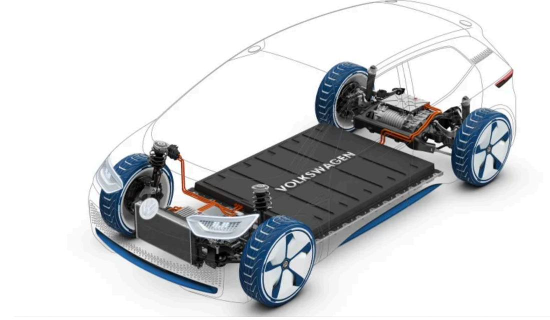


Revisit how we solved it in non-distributed setup

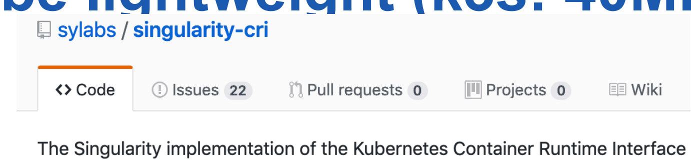
- **Pilot**: capture resources from "provider" generically, defines "blank slate" on which we can build/deploy our workload

What's the pilot in the distributed case

... perhaps it's Kubernetes.



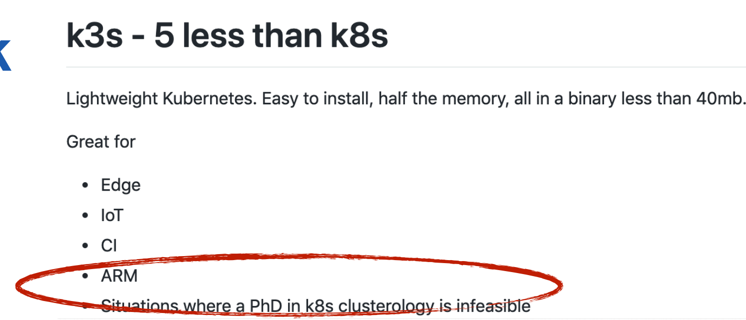
- captures generic multi-node resources
- blank canvas on which a wide range of distributed workloads can be deployed
- community is developing standards how to **describe distributed workloads**
  - **k8s popular target for data anlysis platforms anyways**
- scales to 10-100k cores, handles custom hardware
- increasing attention from HPC community (singularity CRI)
- good for heterogenous environment
  can move from HPC to Cloud and back
- testable by users outside of HPC
- can be lightweight (k3s: 40MB)



sylabs / singularity-cri

<> Code | Issues 22 | Pull requests 0 | Projects 0 | Wiki

The Singularity implementation of the Kubernetes Container Runtime Interface



k3s - 5 less than k8s

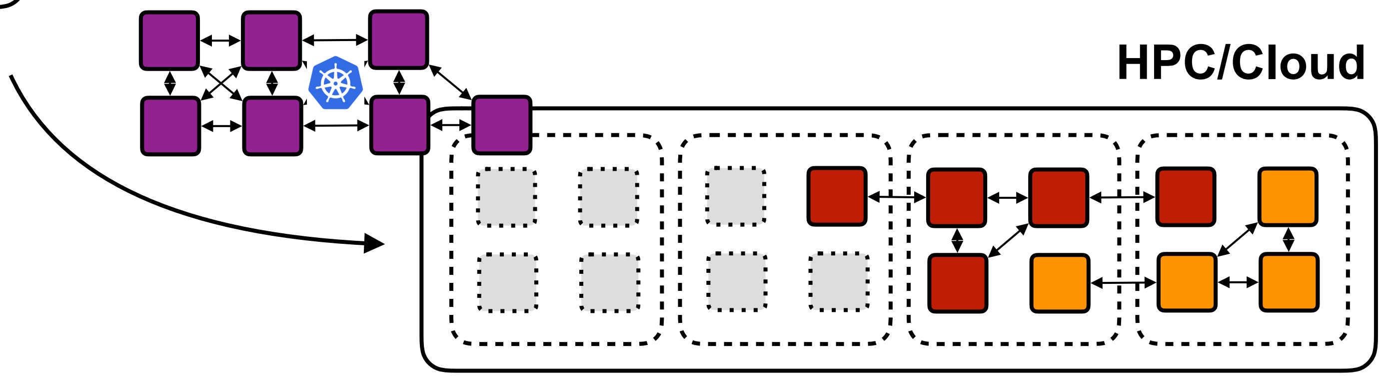
Lightweight Kubernetes. Easy to install, half the memory, all in a binary less than 40mb.

Great for

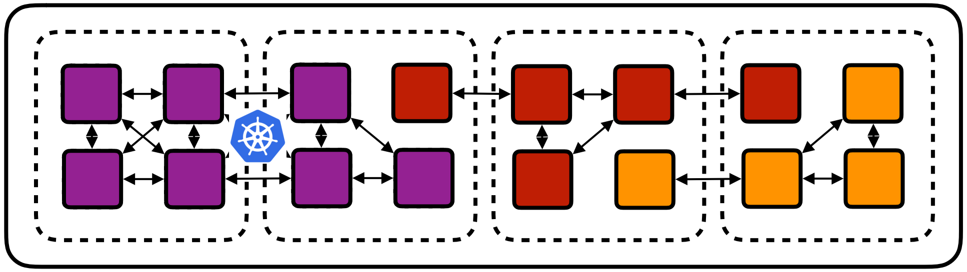
- Edge
- IoT
- CI
- ARM
- Situations where a PhD in k8s clusterology is infeasible

Harvester ?

`pdistributed --app dask/dask:1.0.0 ... ?`



- **grab resource via k8s**
- **deploy workload to k8s**
- **run to completion**
- **evict**



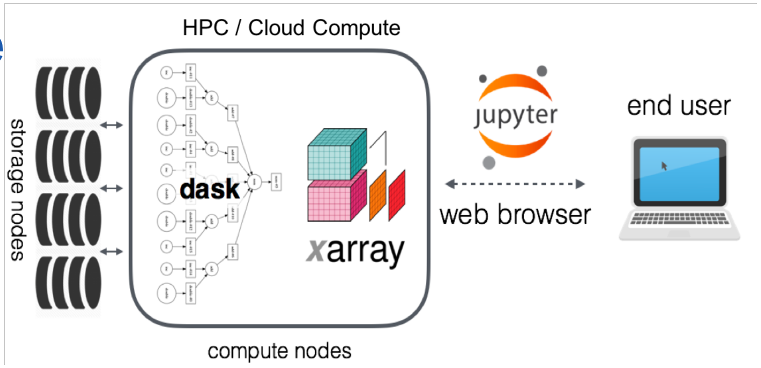
Interactive Workloads

- demand for interactive analysis, backed by large scale compute
- direct ssh access to HPCs not feasible for full VO
-

Pangeo: An Open Source Big Data Climate Science Platform

Example: pangeo.io

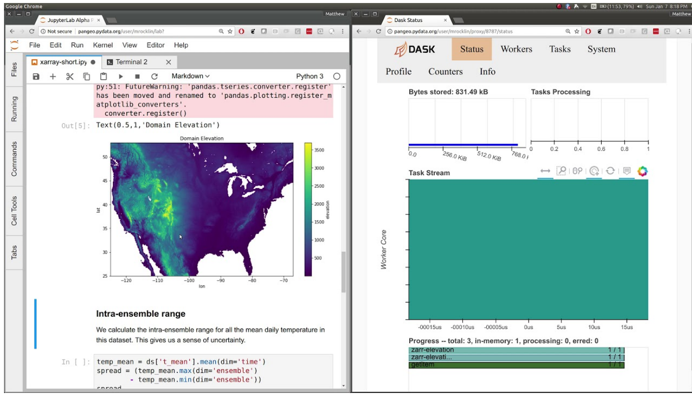
kubernetes + dask + xarray + jupyte



Can we bring this to HEP?

Move from shell to API-based access to distributed compute + advanced UIs

- e.g. JupyterHub at BNL
- atlas-ml.org via SLATE
- SWAN @ CERN



Co-locate with HPC, integrate with

Large number of cycles will be provided by HPC (esp US)

- hardware accelerators (GPU, CSA, ...) following general trend (TPU)
- need to figure out how to use them and integrate them into our existing workload management
- Machine Learning Applications are a natural fit
  - ML development / training
  - end-to-end ML analysis
  - co-processor inference

- For Analysis, forcing users into OpenMPI (as main API) probably too limiting
  - but many opportunities for higher level interfaces (e.g. data-frames, computational graphs) that internally adapt to h/w, distirbuted scenarios
- Kubernetes as Pilot?