iris hep
Institute for Research & Innovation in Software for High Energy Physics

IRIS-HEP View

G. WATTS
2019-07-21
CERN: COMPUTING INFRASTRUCTURES FOR FUTURE DATA ANALYSIS

Buffet Dinner

G. Watts (UW/Seattle)

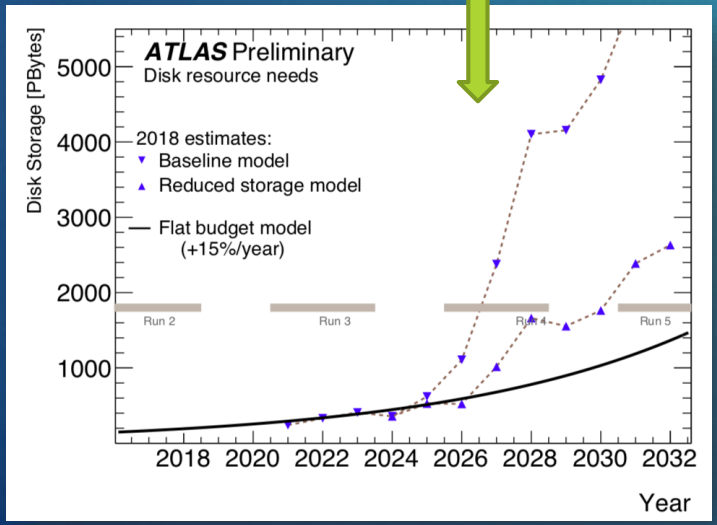
1. The ATLAS Run 3 and Run 4 Plan
2. ~~iDDS and ServiceX~~
3. func_adl progress (infrastructure)
4. ~~The Bigger (analysis) Picture~~

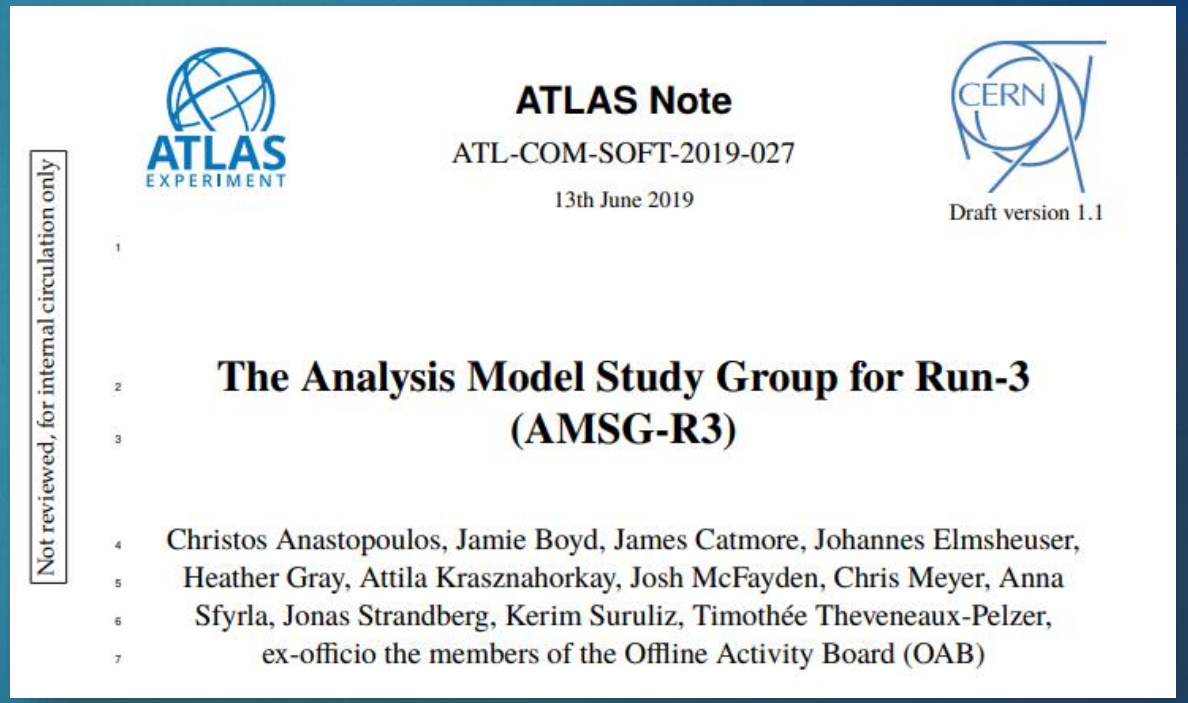
The ATLAS Computing Model

The ATLAS AMSG-R3

Task force assembled to review and make recommendations on the ATLAS analysis model.

Can we improve this?





Not reviewed, for internal circulation only

ATLAS EXPERIMENT

ATLAS Note

ATL-COM-SOFT-2019-027

13th June 2019

CERN

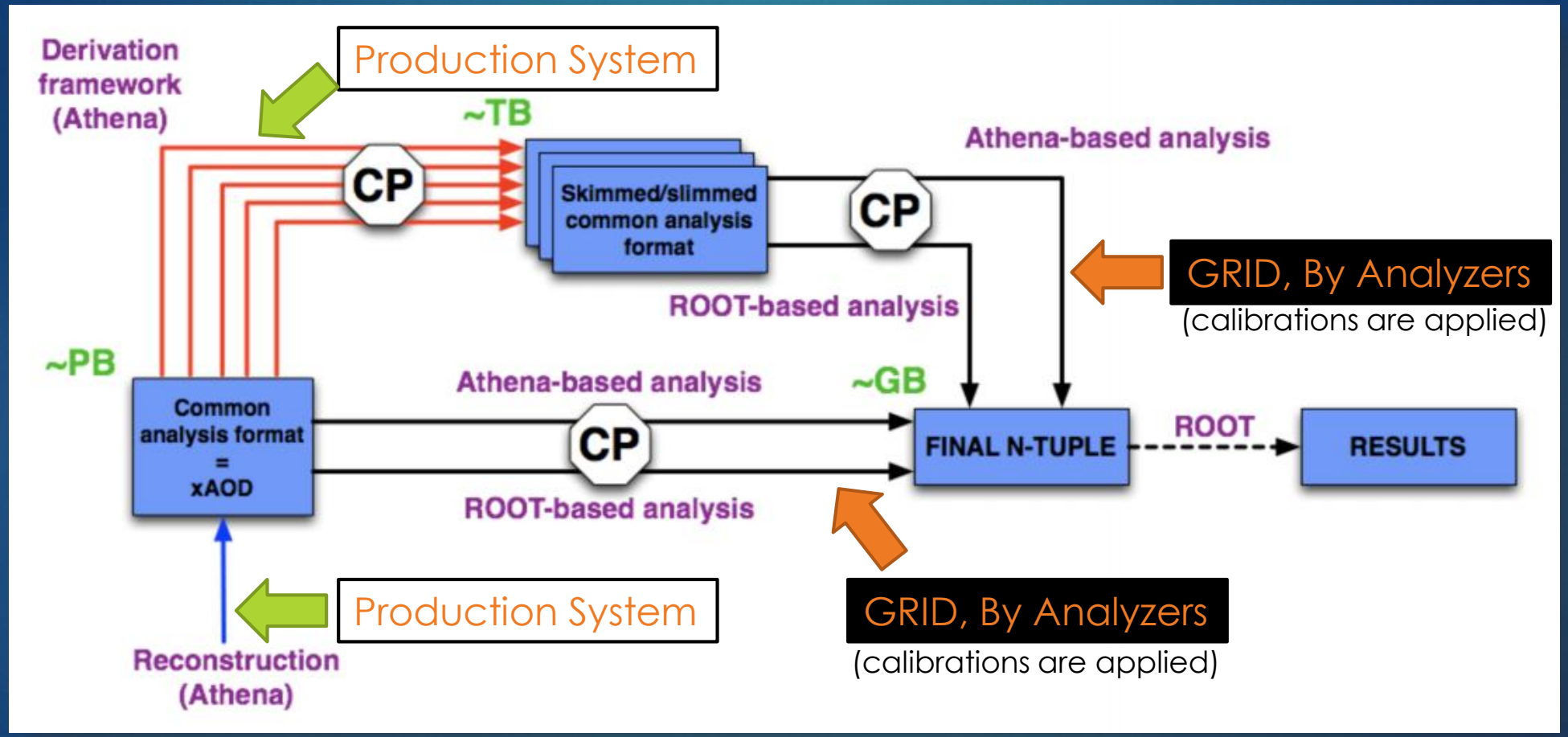
Draft version 1.1

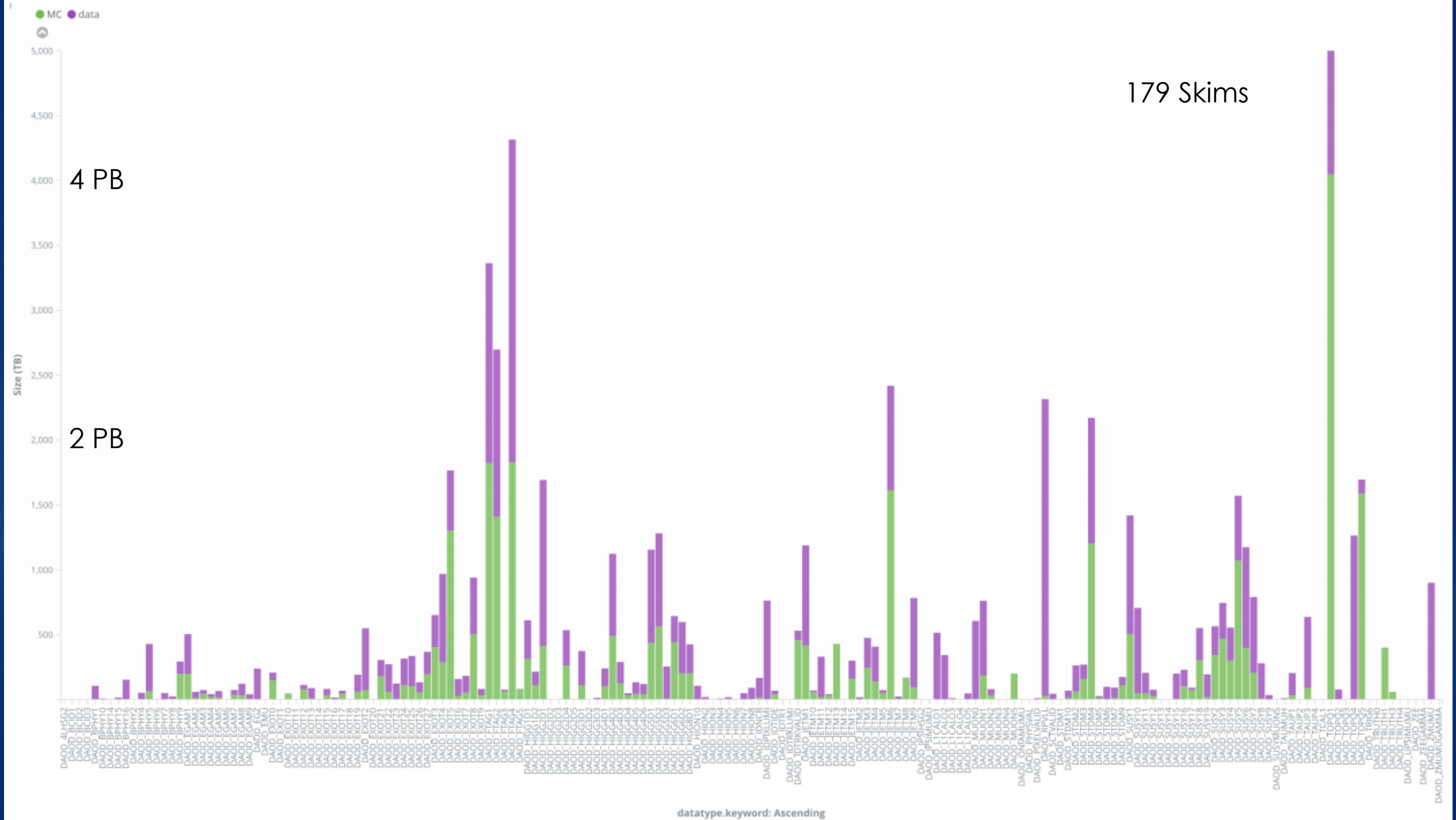
The Analysis Model Study Group for Run-3 (AMSG-R3)

Christos Anastopoulos, Jamie Boyd, James Catmore, Johannes Elmsheuser, Heather Gray, Attila Krasznahorkay, Josh McFayden, Chris Meyer, Anna Sfyrla, Jonas Strandberg, Kerim Suruliz, Timothée Theveneaux-Pelzer, ex-officio the members of the Offline Activity Board (OAB)

Currently using ~200 PB of data for everything
(68,267 HD Game Of Thrones Episodes)

Current System

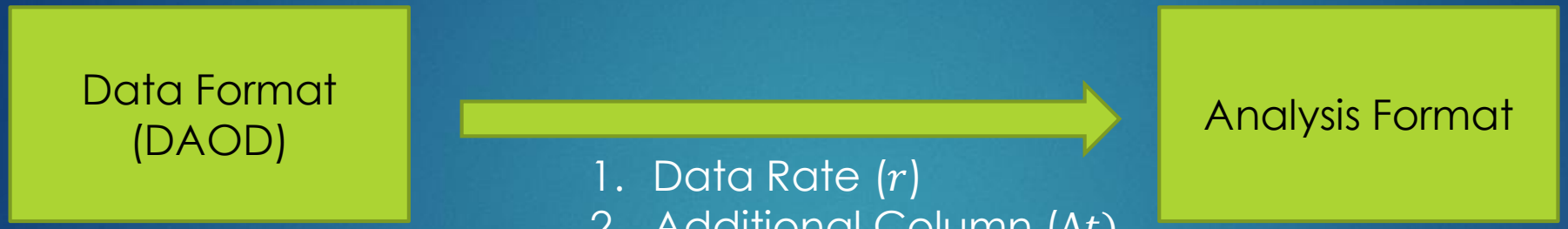



MC
data
179 Skims
4 PB
2 PB
Size (TB)
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
datatype.keyword: Ascending

(not to mention the extra variables required)

The Cost Of Slow Calibrations

G. Watts (UW/Seattle)



Scenario 1:

$r = 1$ Hz
$\Delta t = 0.01$ Hz

1. One new column is same time as 100 columns
2. Each run takes a long time

Run and dump everything if you think you need it or not. Add by re-running all columns + new ones

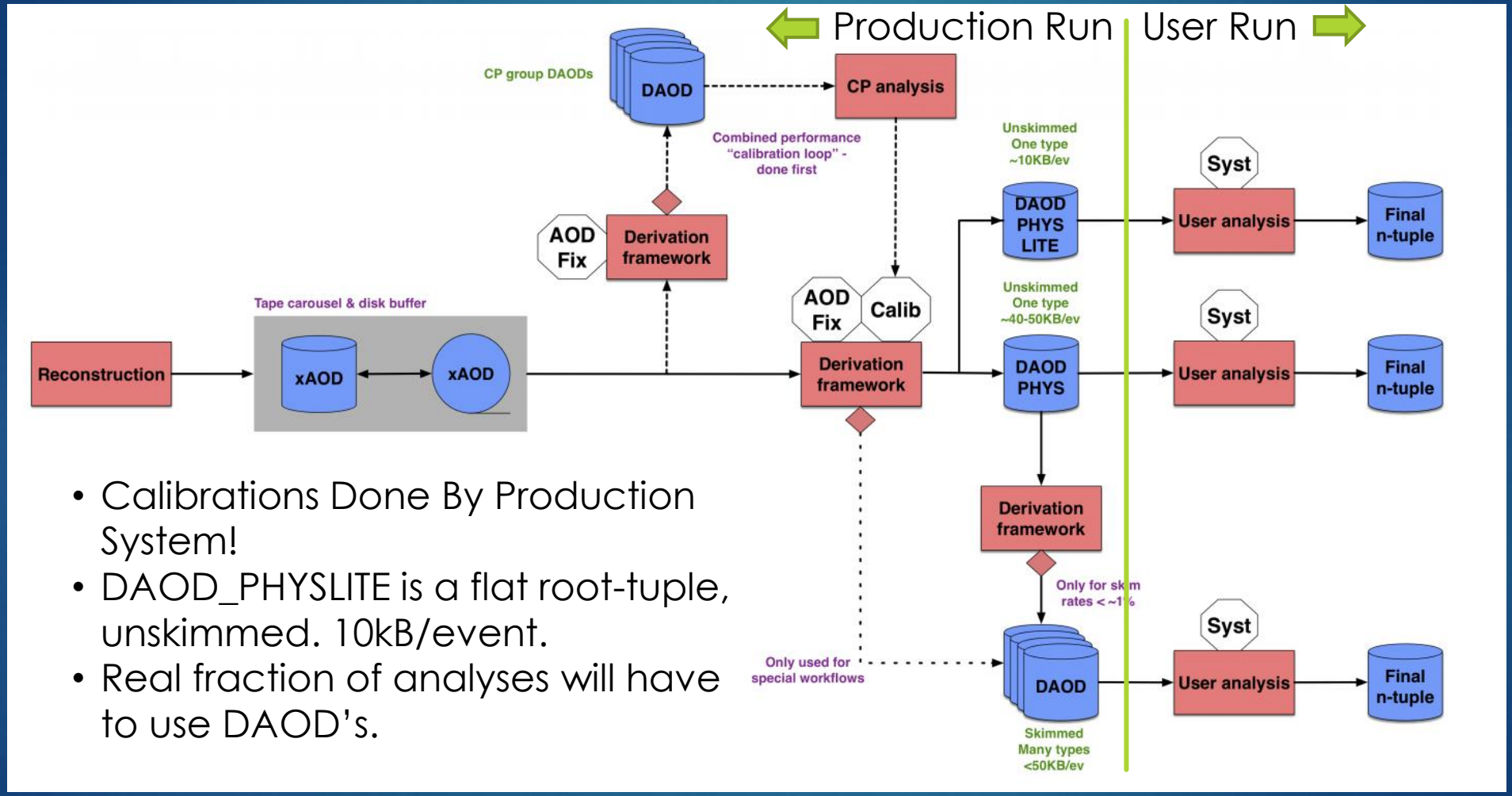
Scenario 2:

$r = 10{,}000$ Hz
$\Delta t = 0.01$ Hz

1. One new column is much faster than 100 columns
2. Runs are *short*

Run for just the columns you need, add more later

Modern Version



- Calibrations Done By Production System!
- DAOD_PHYSLITE is a flat root-tuple, unskimmed. 10kB/event.
- Real fraction of analyses will have to use DAOD's.

Run 3 Sizes

	MC				Data			
	AOD	DAOD	DAOD_PHYS	DAOD_PHYSLITE	AOD	DAOD	DAOD_PHYS	DAOD_PHYSLITE
events	3×10^{10}	1×10^{11}	3×10^{10}	3×10^{10}	2×10^{10}	1×10^{11}	2×10^{10}	2×10^{10}
size/event [kB]	600	100	70	10	400	50	40	10
disk space [PB]	18.0	10.0	2.1	0.3	8.0	5.0	0.8	0.2
"versions"	1.5	2	2	2	1.5	2	2	2
replication	0.5	1	4	4	0.5	2	4	4
sum [PB]	**13.5**	**20.0**	**16.8**	**2.4**	**6.0**	**20.0**	**6.4**	**1.6**

Run 4 is 3000 fb^{-1}
Run 3 is 300 fb^{-1}

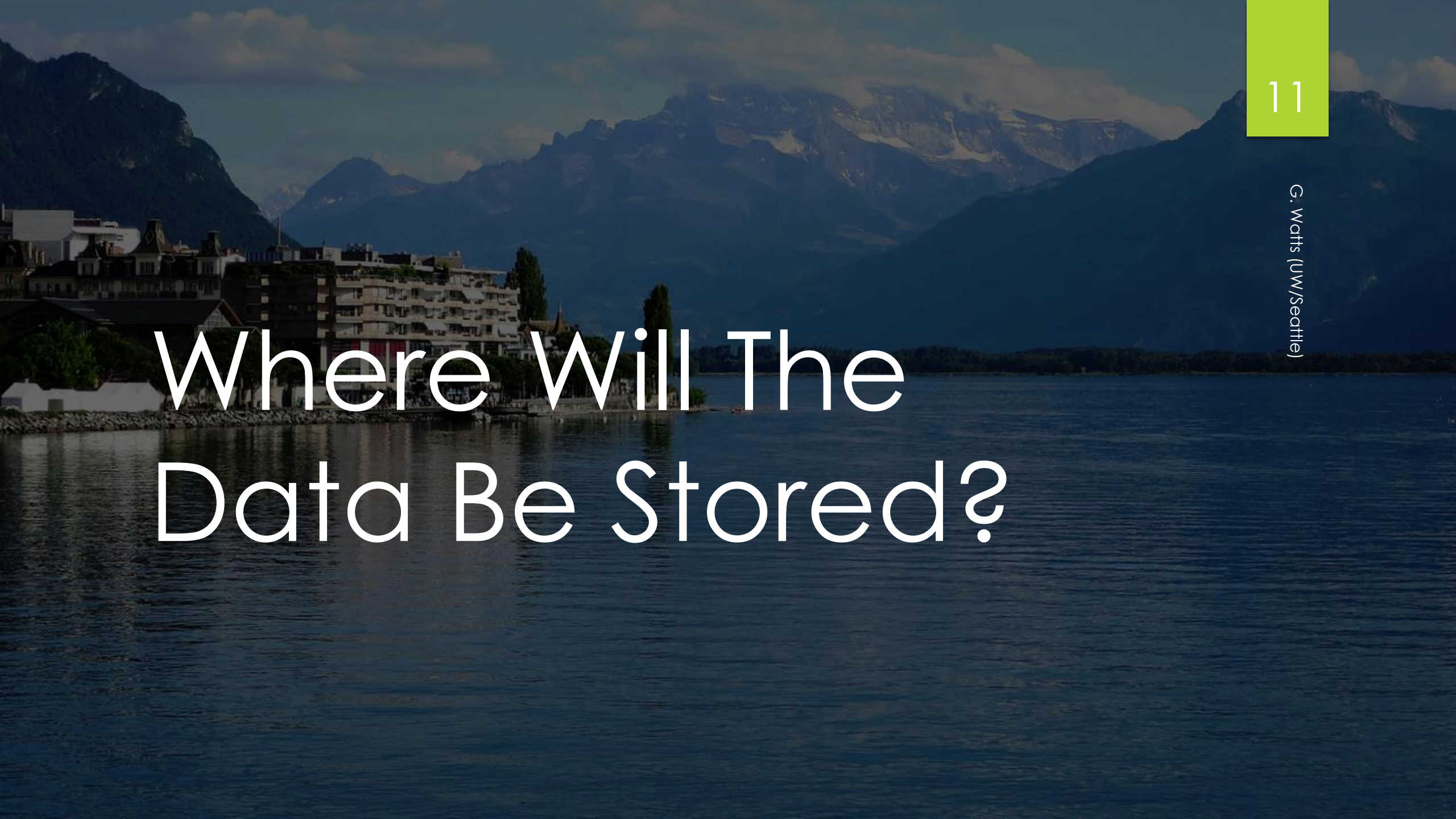
× 10 more data (16 PB for the PHYSLITE)

What Work?

Run 2	MC		Data		Sum
	AOD	DAOD	AOD	DAOD	
events	$3 \cdot 10^{10}$	$2 \cdot 10^{11}$	$2 \cdot 10^{10}$	$2 \cdot 10^{11}$	
size/event [kB]	600	100	400	50	
disk space [PB]	18.0	20.0	8.0	10.0	
other/older versions factor	1.5	2	1.5	2	
replication factor	1	1	2	2	
Sum [PB]	27.0	40.0	24.0	40	131.0

Table 16: Simple disk space model for the end of Run2 for AODs and DAODs based on Tab. 1.

- For most things it is an adaptation of our current EDM
- We have storage shims built in that allow us to mess with the data layout without affecting everyone's use of the data.
- Some real changes do need to happen in the EDM for some things.
- Penalty? About 35% more disk space used in Run 3, for Run 4 lethal.



Where Will The Data Be Stored?

Data Lakes & Infrastructure Feeding Analysis

G. Watts (UW/Seattle)

High Speed Analysis Skim

50-100's of TB's

Traditionally Run on the GRID

Currently fits on a Synology!
(1 TB SSD is $100 USD)
Will Intel Optane Memory change this equation?

Will only be stored on large GRID sites!

Some Conclusions

- Skimming will be a major operation but will be very fast!
  - 1-10K Events/second on current CPU's
  - Filter, Extract the data you need, etc.
  - Disk space requirements mean there will be a heavy requirement on keeping samples small
  - Your laptop will not be good enough to process 50-100 TB's of data
- Unlikely small sites will have 16 PB DAOD_PHYSLITE.
- What will we do for analyses that need more than PHYSLITE?
- Some question remain:
  - Can we have a system where analyzers only ask for what they need at the time they need it?
    - Further reducing disk space at the edges
  - Will flat formats be good enough (flat TTree's, awkward arrays, etc.)?
  - How similar will CMS and ATLAS and others be? Probably very now that everyone seems to be moving to rucio. That will be a huge thing.

Some Speculation

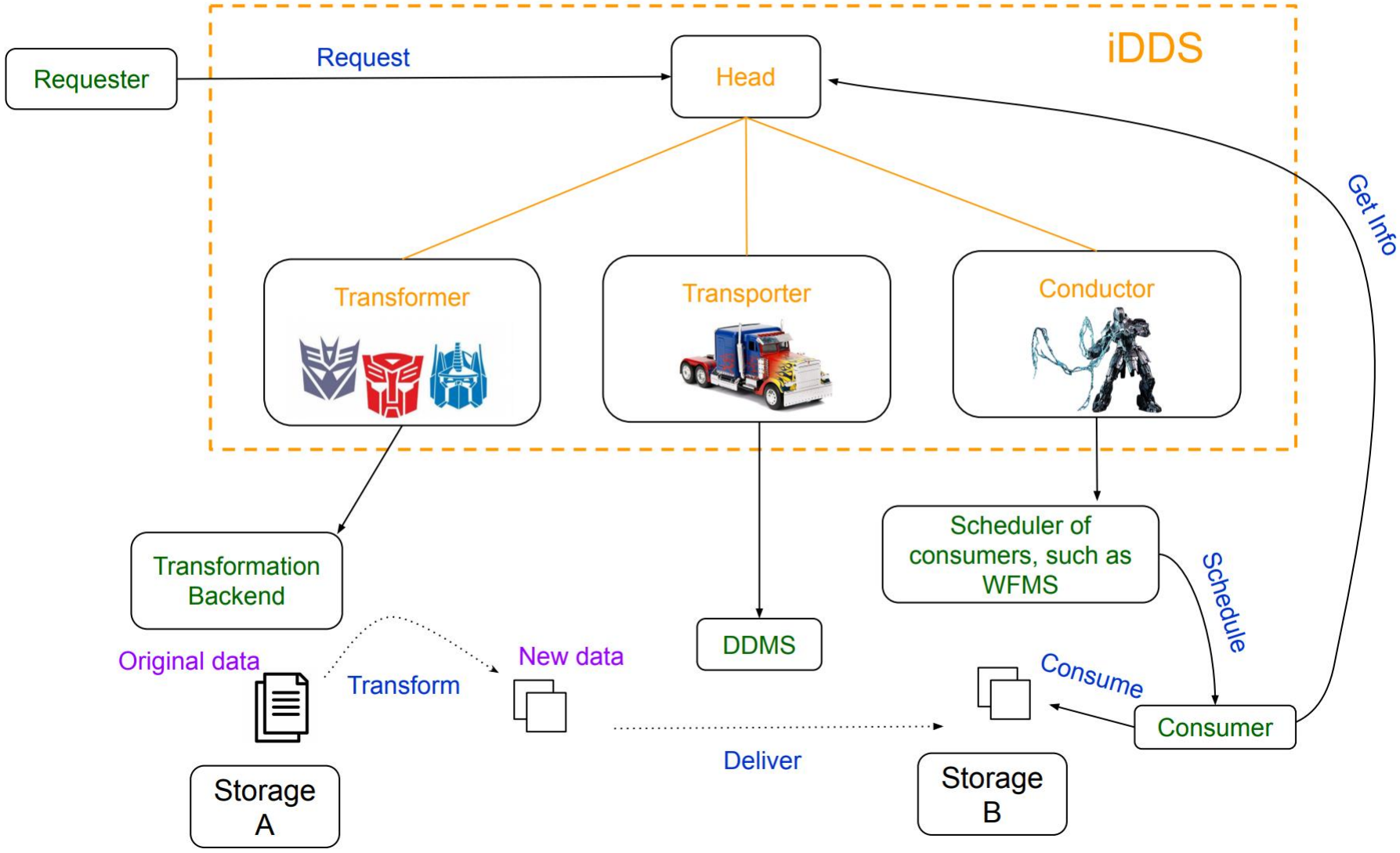
- Will more plot-making move to the GRID?
  - Where is the threshold for people to transition from a local machine?
  - My guess is about 2-3 hours to spin through the data
  - Normal running on the GRID for an initial skim can take ~days.
- Efforts to keep it down below several hours
  - People have made huge efforts to keep data sizes small
- The Future
  - GRID turn-around will need to be improved
  - New models that allow us to access the data quickly and in parallel
  - As we design Data Query languages, we need to keep this in mind.

IRIS-HEP: iDDS and ServiceX

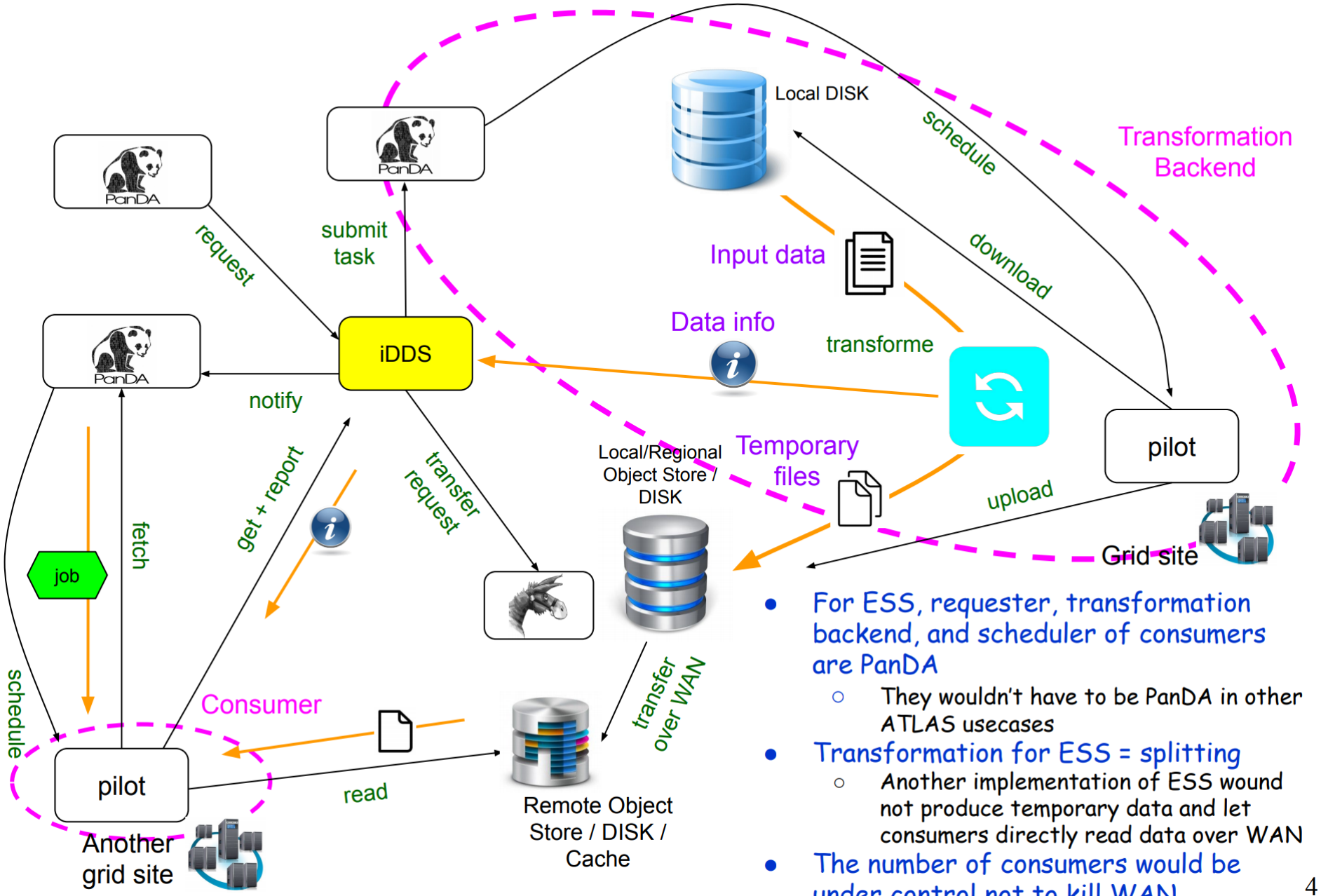
Introduction

- iDDS : intelligent Data Delivery Service
- An intelligent service orchestrating workflow management system (WFMS) and data management system (DDMS) to transform if needed and deliver collections of data to consumers
  - Not a storage, WFMS, or DDMS
  - Delegation of many functions to WFMS, DDMS and infrastructure like network and cache
- Joint project between ATLAS and IRIS-HEP
- Requirements
  - Experiment agnostic
  - Flexibility to support many use-cases and backend systems
  - Easy and cheaper deployment model
- Key functions
  - Transformation
  - Delivery
  - Orchestration

Architecture Overview of iDDS



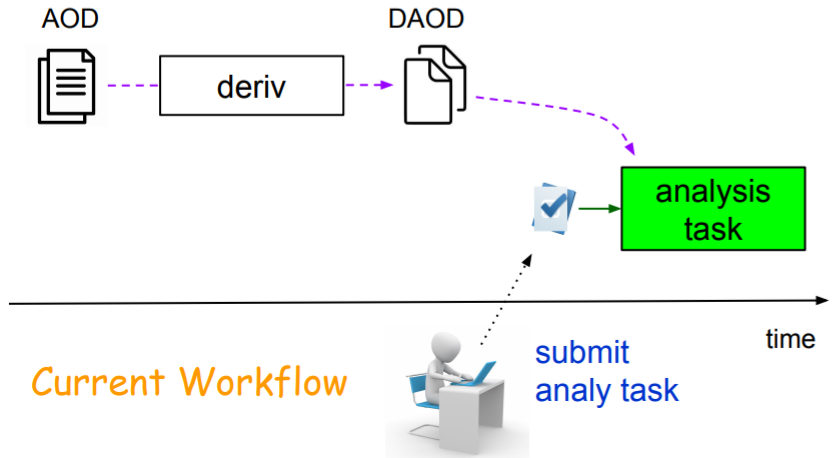
ATLAS implementation of iDDS for ESS + WAN

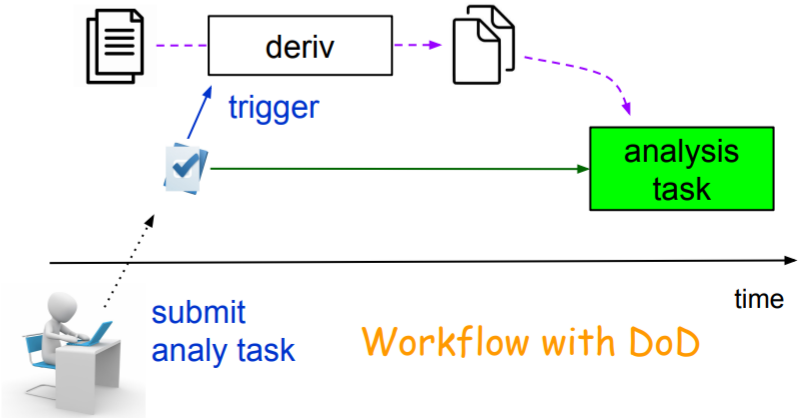


- For ESS, requester, transformation backend, and scheduler of consumers are PanDA
  - They wouldn't have to be PanDA in other ATLAS usecases
- Transformation for ESS = splitting
  - Another implementation of ESS wound not produce temporary data and let consumers directly read data over WAN
- The number of consumers would be under control not to kill WAN

A Possible Usecase for Cost Saving – Derivation on Demand (DoD) –

- Passive creation of DAOD rather than proactively keeping them on DISK
  - Run derivation when analysis tasks are submitted
  - Analysis tasks = user tasks running on DAOD != final analysis running on NTuple etc
  - Very short lifetime for DAOD
- Users wouldn't be able to run arbitrary derivation
  - Carriages, input datasets, and DAOD dataset names still to be centrally and officially defined in the derivation framework
- Essentially DAOD would be created only when users would really need these data for analysis → No unused DAOD in principle





Delivering data to new analysis platforms

First introduced in Feb 2018 whitepaper: delivery of data from lakes to clients (insulate from upstream systems, remove latencies, reformat, filter & possibly accelerate)

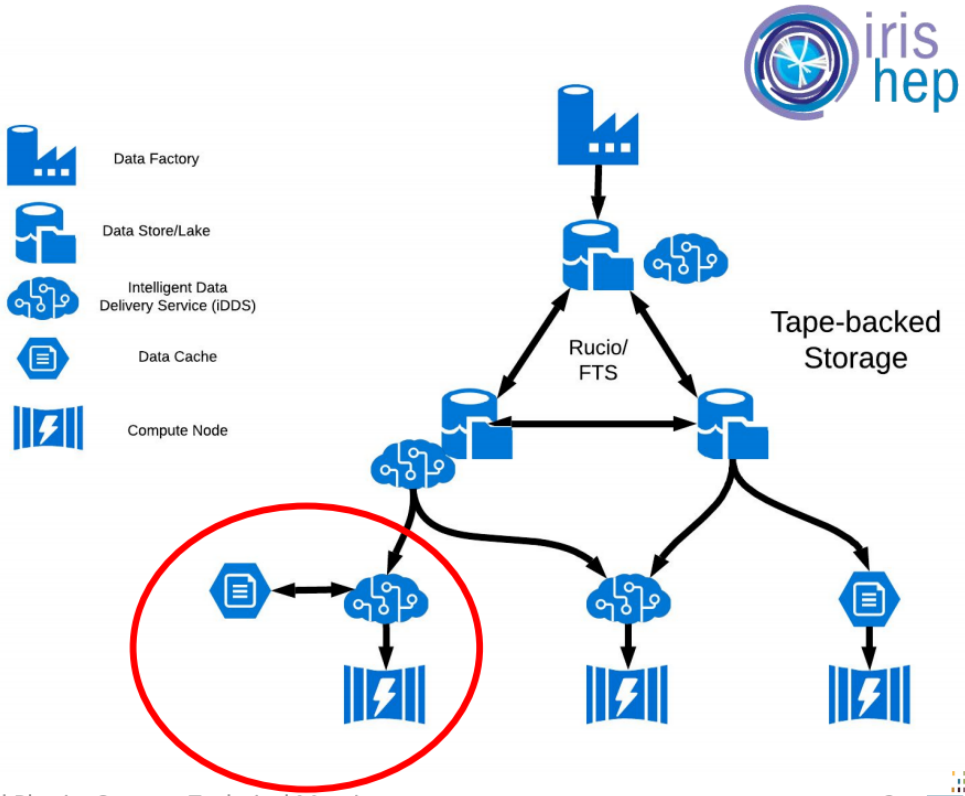
Nicely aligned with existing Event Service/ESS concepts

iDDS introduced by IRIS-HEP as framework to support development of both ideas

iDDS/ESS – (BNL, Wisconsin) – focuses on integration with PanDA WMS

iDDS/ServiceX – (Chicago, Illinois) – focuses on integration with Rucio and reformatting for pythonic tools & **endstage analysis systems**

Goal: reusable, **containerized service components** (e.g. caching, filters, reformatters), **interchangeable** between PanDA & generic analysis platforms (Coffea, Spark, Dask, ..)



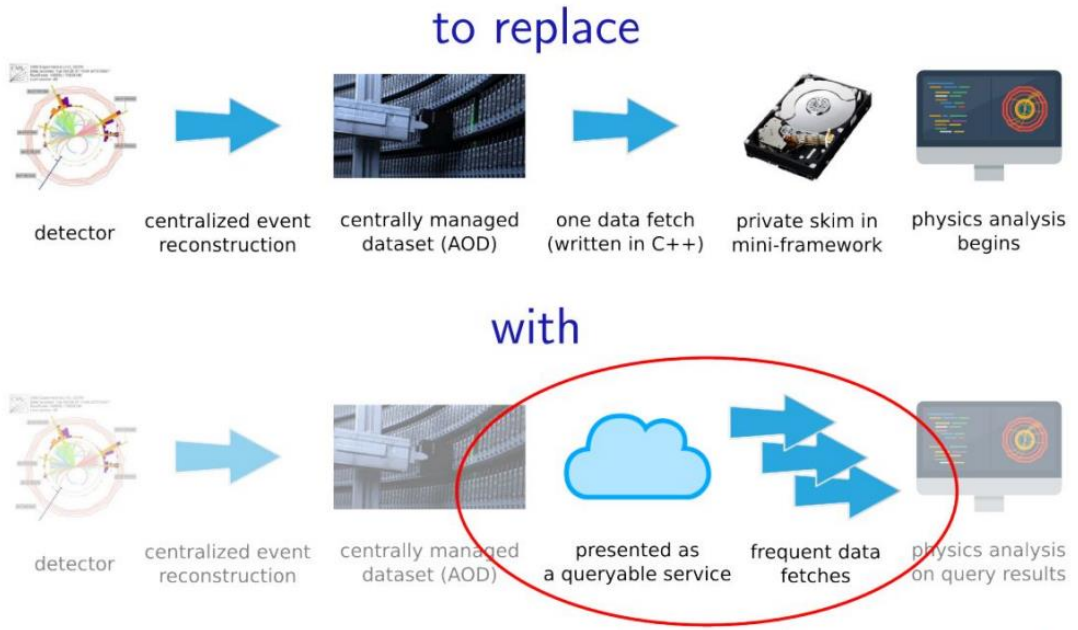
Classic analysis workflow

xAOD → DAOD → flat ntuples → skimmed ntuples → histograms/plots

- First two formats are prescribed, but enormous variation in after that

Standard analysis might have "primary ntuple"

- Write ntuplization code to dump xAOD into flat trees with specialized objects
- Submit jobs to HTCondor by hand
- Primary ntuple then skimmed/trimmed; some data replicated (multiple times)
- Selections/cutflows baked into analysis
- Adding new variables means throwing previous skim, replicating everything



ServiceX goals

Adding components to an overall iDDS ecosystem. Input-agnostic service to enable on-demand data delivery.

Tailored for **nearly-interactive,** high-performance **array-based analyses**

- Provide uniform interface to data storage services; users don't need to know how or where data is stored
- Capable of **on-the-fly data transformations** into variety of formats (ROOT files, HDF5, Arrow buffers, Parquet files, …)
- Pre-processing functionality: Unpack compressed formats, filter events in place, project data columns

Support for columnar analyses. Start from any format, extract only needed columns

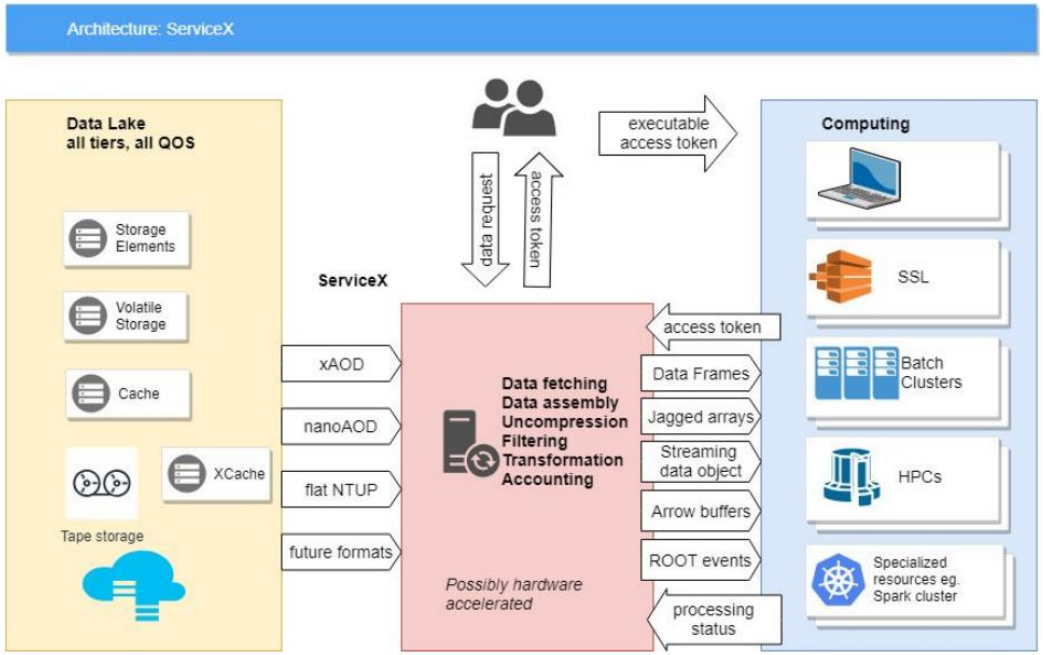
ServiceX components

Users specify needed events/columns and desired output format

- Use metadata tags (real/sim data, year, energy, run number, …)
- Any required preselection

ServiceX

- Queries backend (Rucio) to find data
- Gives unique token to identify request
- Access data from storage through XCache
- Validates request
- Perform data transformations
- Keeps track of data delivered and processed; ensures all events processed



ServiceX implementation

System designed to be modular

- Can switch out modules to transform different types of input data, swap schedulers, …

Implemented as central service in Kubernetes cluster on GCP

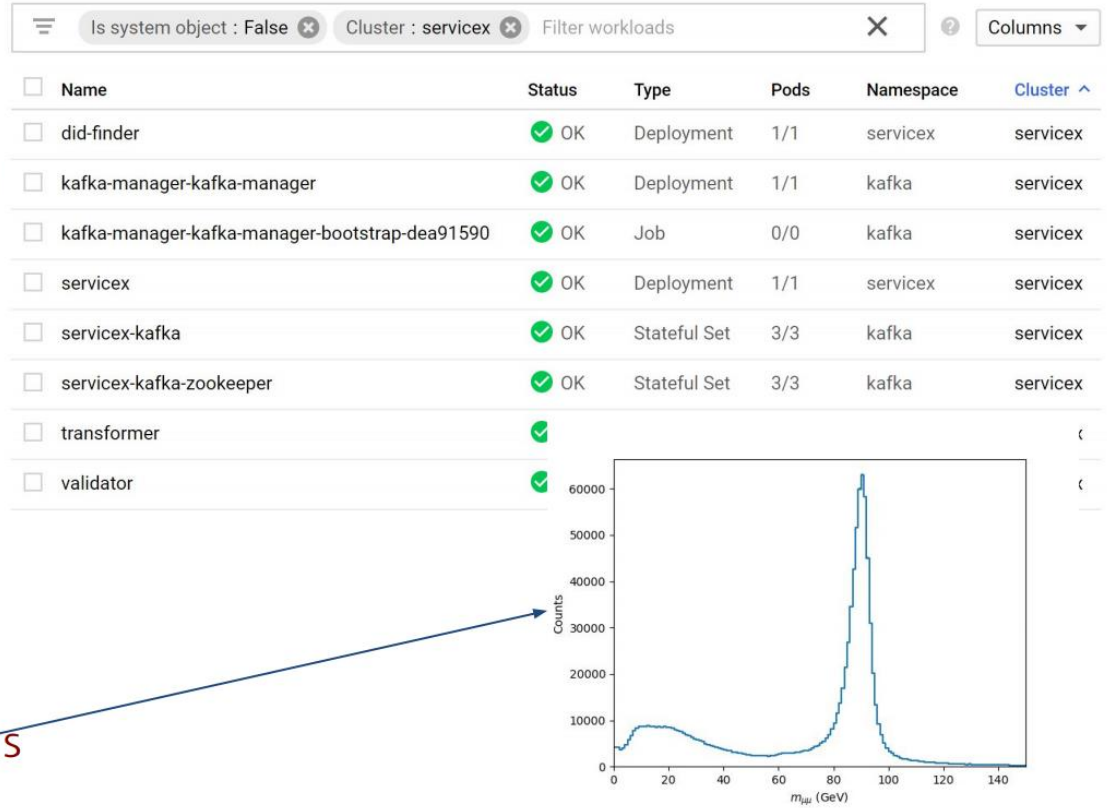
- Easy to deploy: Just use Helm chart to define plugins to run
- Will deploy on k8s-based IRIS-HEP Scalable Systems Lab (SSL) when available
- **Reproducible pattern** for deployment on Kubernetes clusters (e.g. **Tier2s**, **institutional k8s T3**?)

Composed of multiple deployments: API server, DID finder, transformer, Kafka manager, …

- API server: Interface for users, manages requests via DB
- DID finder: Queries data lake via Rucio, writes ROOT files to XCache
- Transformer: Takes input files from Xcache, outputs in various formats (flat trees, Awkward arrays, Parquet, Arrow tables, …)
- Kafka manager: Receives input from producer (currently transformer) and makes topics available to consumers (analysis jobs)

Early prototype up and running – backend

- Currently deployed on Kubernetes clusters on GKE
- Independent auto-scaling of components based on load
- Caching of inputs using XCache
- Transformation product is also persistified by Kafka
- Implemented backpressure for large requests
- Additional K8s cluster running Spark to perform analysis step.
- Uses Awkward array tools to perform calculations on columns



func_adl and Infrastructure

EATING YOUR DOGFOOD

Jet p_T for 57 Long Lived Particle Datasets

G. Watts (UW/Seattle)

```
In [2]: datasets = pd.read_csv(df_loc)
```

```
In [3]: datasets
```

Out[3]:

	mH	mS	Lifetime	MCCampaign	RucioDSName
0	60	5	5	mc16a	mc16_13TeV.311417.MGPy8EG_A14NNPDF23_NNPDF31ME...
1	60	5	5	mc16d	mc16_13TeV.311417.MGPy8EG_A14NNPDF23_NNPDF31ME...
2	60	5	5	mc16e	mc16_13TeV.311417.MGPy8EG_A14NNPDF23_NNPDF31ME...
3	60	15	5	mc16a	mc16_13TeV.311418.MGPy8EG_A14NNPDF23_NNPDF31ME...
4	60	15	5	mc16d	mc16_13TeV.311418.MGPy8EG_A14NNPDF23_NNPDF31ME...
5	60	15	5	mc16e	mc16_13TeV.311418.MGPy8EG_A14NNPDF23_NNPDF31ME...

1

57 rucio xAOD datasets

```
In [4]: async def get_jet_pt_distro(df_name:str):
            'Return a future that will be a pandas array of the number of jets'
            return await EventDataset(f'localds://{df_name}') \
                .SelectMany('lambda e: e.Jets("AntiKt4EMTopoJets")') \
                .Where('lambda j: j.pt()/1000.0 > 40.0') \
                .Select('lambda j: j.pt()/1000.0') \
                .AsPandasDF(['JetPt']) \
                .future_value(use_exe_func_adl_server)
```

Physics

```
In [7]: DSInfo = namedtuple('DSInfo', 'mH mS lt cp data')
        async def info_tuple (mH, mS, lt, cp, ds):
            return DSInfo(mH, mS, lt, cp, await get_jet_pt_distro(ds))
        pd_arrays = [info_tuple(mH, mS, lt, cp, ds) for index, (mH, mS, lt, cp, ds) in datasets.iterrows()]
```

Boilerplate

```
In [8]: %time
        results = await asyncio.gather(*pd_arrays)
```

2

Get the Jet p_T for all jets with $p_T > 40\ GeV$ (all at once)

This notebook, some simplified examples.

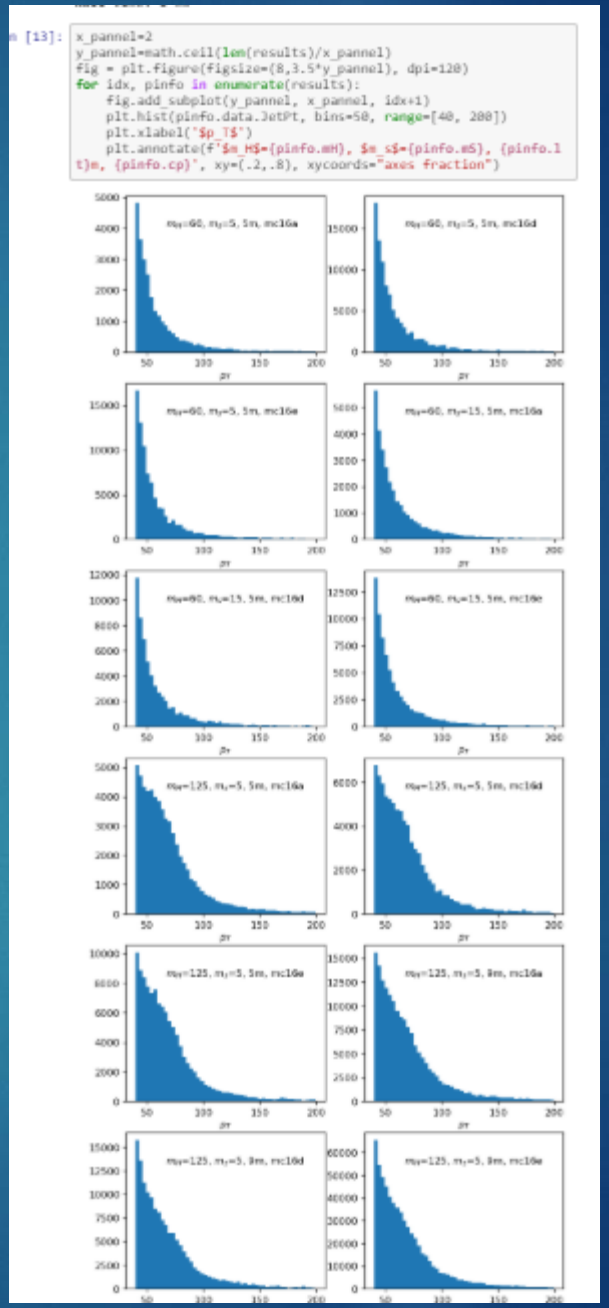
Plot

Plot!

The rucio MC datasets were 1.6 TB
The total data delivered to python was 84 MB

How close to the data source should we push cuts?

- Less data means
  - makes it easy to handle on the client
  - Bandwidth limited is possible
- OTOH
  - More complex cuts at data source are more expensive
  - Harder to maintain code
  - Harder to fall into pit of success



That took 20 minutes (excluding data download)
On a 5-year-old Core i5 machine at UW

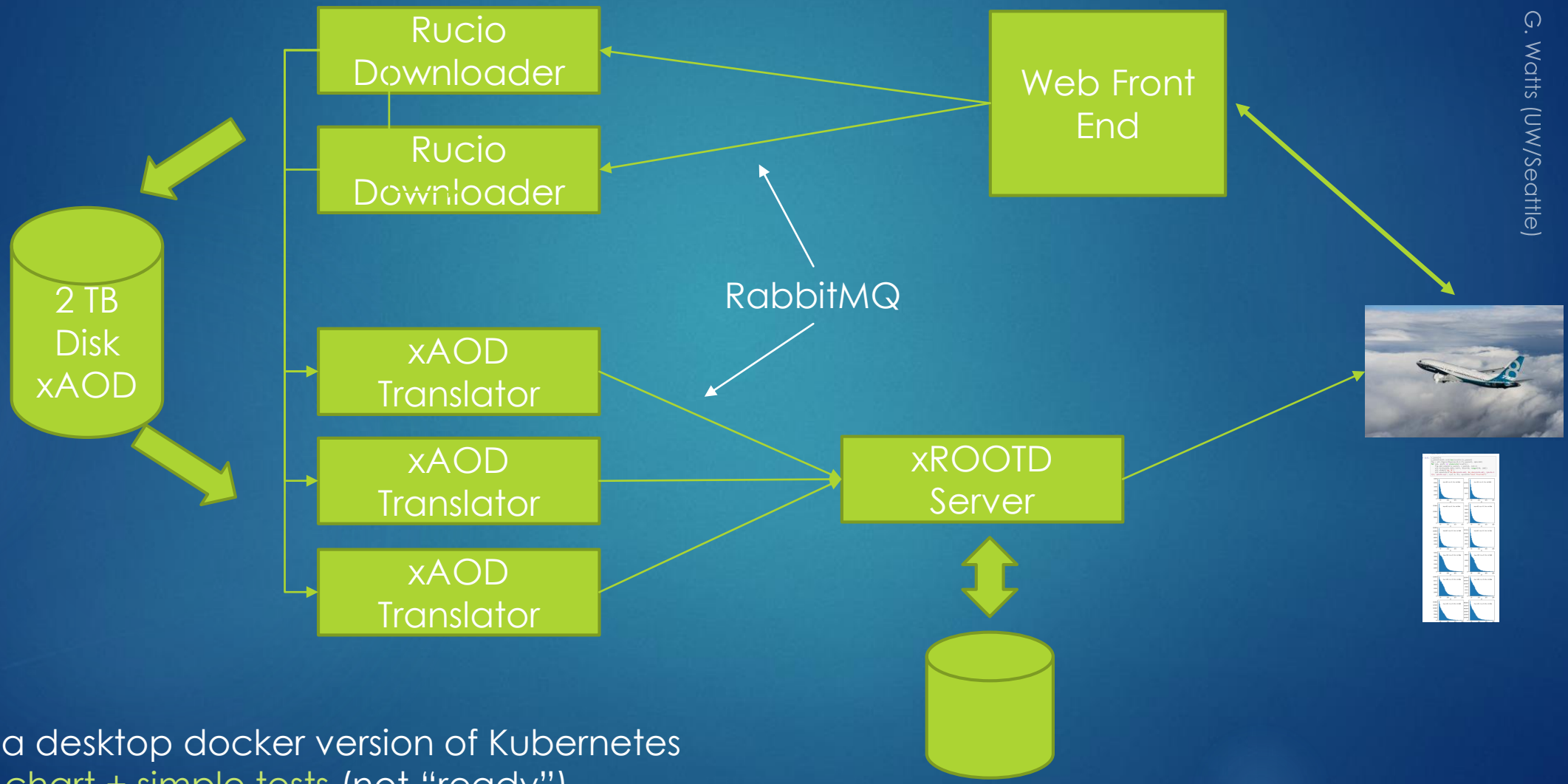
That took 20 minutes (excluding data download)
On a 5-year-old Core i5 machine at UW

To Slow

20 Minutes	Core i5 with 2 cores and hyper threading, spinning disk
30 Seconds	Would require ~200 cores and distributed disk (?)
< 30 Seconds	Requires changing the way we build software for ATLAS

Think of this as the cost of adding a new column from a binary format!

Design



Used a desktop docker version of Kubernetes
Helm chart + simple tests (not "ready")

Comments On The Tooling

- Containers
  - Here to stay.
  - Likely the primary mode of distributing environment by Run 4 timeframe.
    - ATLAS wants to move all its GRID workloads to these.
  - Anything requiring a custom software setup will require them (CMS, ATLAS, etc.)
  - Baked into almost every OS (except Mac?)
- Running Containers
  - Custom interfaces (e.g. GRID)
  - Kubernetes when you have a stable workflow.
- Workflow
  - Modern OSS tools are far beyond what we have in HEP
  - Most tools we depend on now would be written in a totally different way using RabbitMQ, etc.
  - Scale up for the previous page: need a bigger cluster and it should just work
- Distribution of a WorkFlow
  - K8 is almost the only game in town right now
  - Combined with helm.
  - Can bring up almost anywhere

Took me a month to learn these tools

Anything new should start with this!

Putting It All Together

- We are talking mostly about DataQuery here
- But there is so much more…