HEP Machine Learning on HPCs

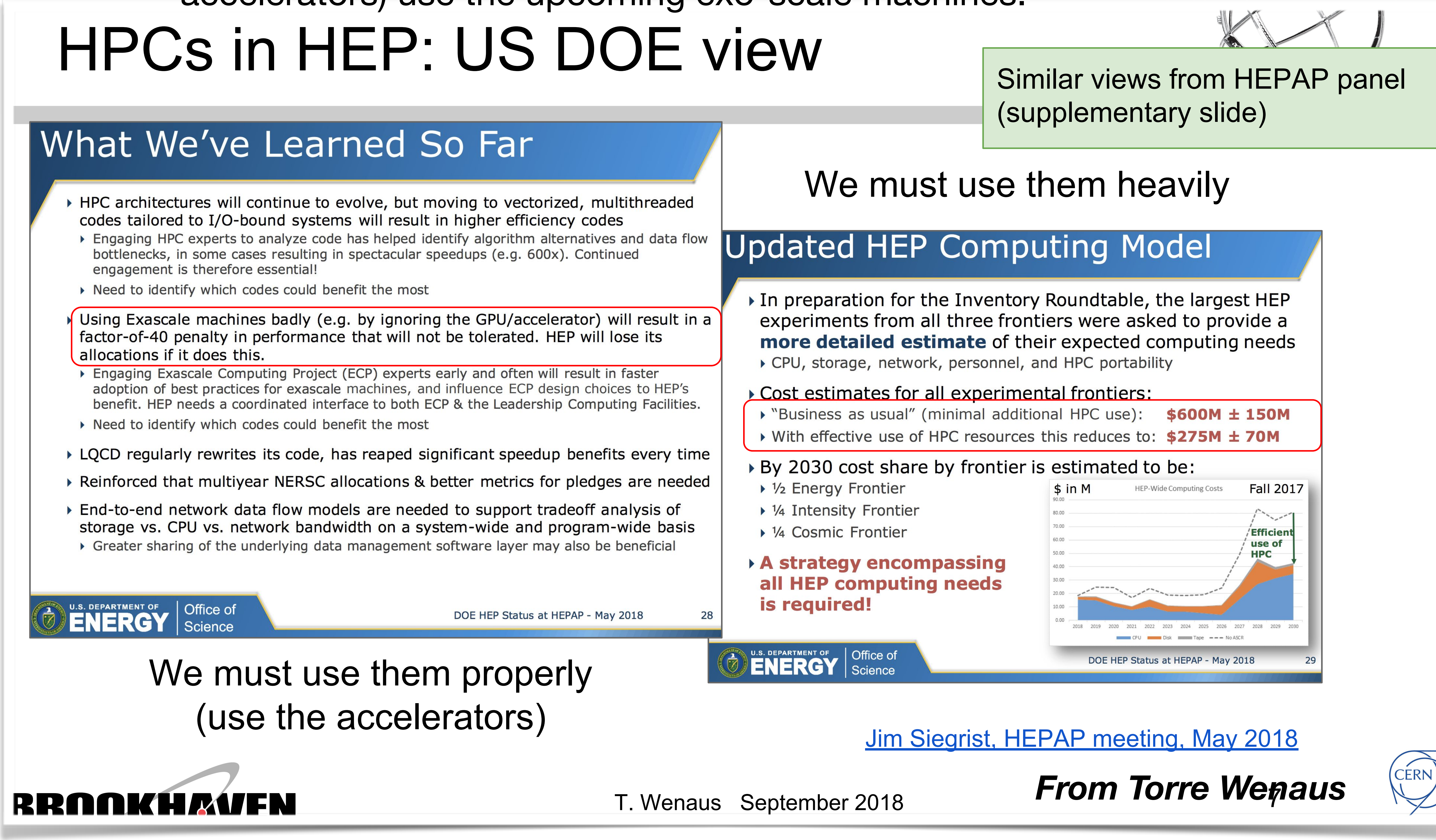
Amir Farbin
University of Texas Arlington

The Motivation

- Primary driver is Deep Learning training, but in principle other techniques are candidates too.
- Deep Learning training can be complex and time consuming:
  - Even with simple models that train relatively quickly (e.g. in a few hours), Hyper-parameter optimization can require training a lot of models.
  - Often GPUs provide > O(50) training acceleration.
  - For some problems, training times on single GPUs are O(week).
    - Realistic Reconstruction and Simulation Deep Learning Tasks
- Goal: a training service that provides necessary resources, manages the complexity, and utilizes parallelism to accelerate the training process for any users.

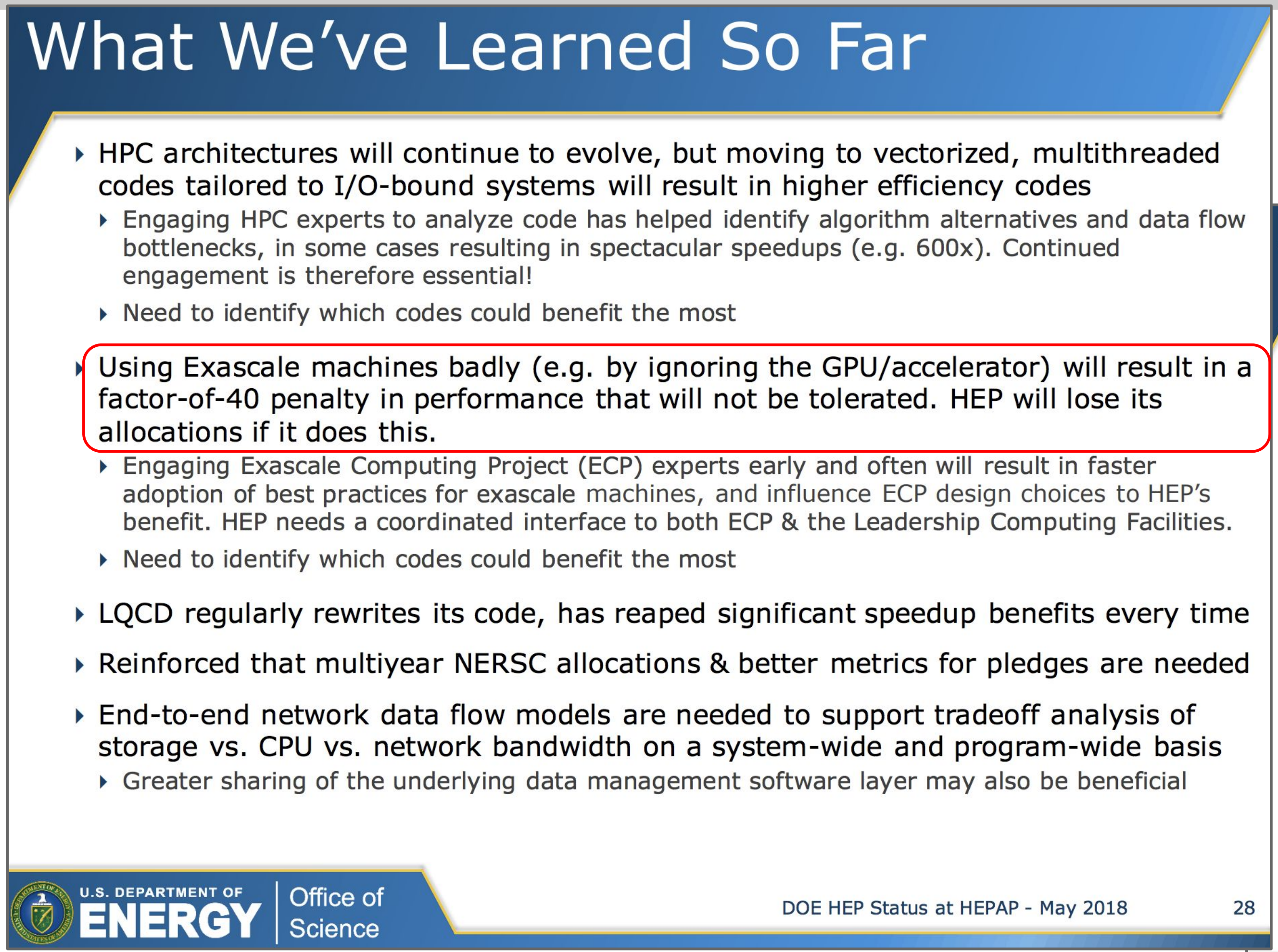
Motivation 2

- DOE wants the LHC experiments to meaningfully (i.e. using the accelerators) use the upcoming exo-scale machines.



HPCs in HEP: US DOE view

Similar views from HEPAP panel (supplementary slide)



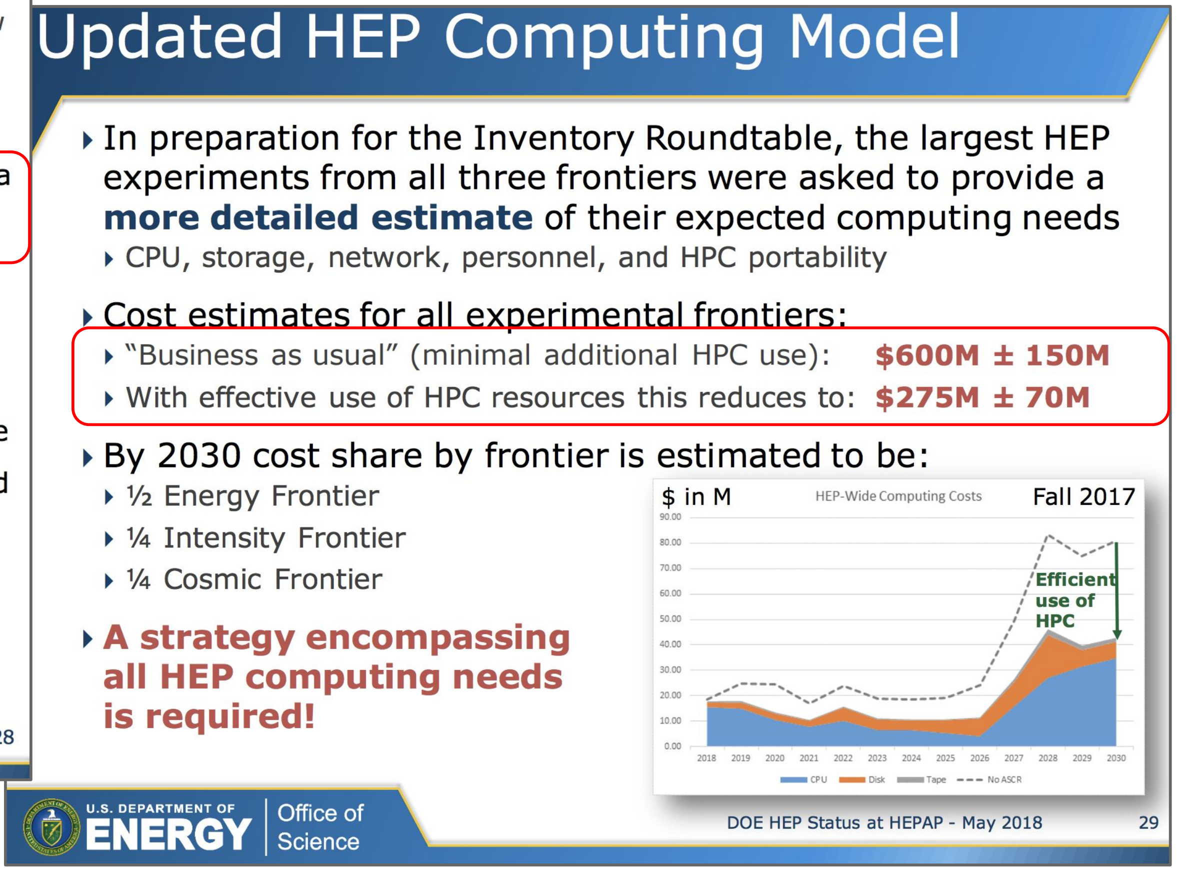
What We've Learned So Far

- HPC architectures will continue to evolve, but moving to vectorized, multithreaded codes tailored to I/O-bound systems will result in higher efficiency codes
  - Engaging HPC experts to analyze code has helped identify algorithm alternatives and data flow bottlenecks, in some cases resulting in spectacular speedups (e.g. 600x). Continued engagement is therefore essential!
  - Need to identify which codes could benefit the most
- Using Exascale machines badly (e.g. by ignoring the GPU/accelerator) will result in a factor-of-40 penalty in performance that will not be tolerated. HEP will lose its allocations if it does this.
  - Engaging Exascale Computing Project (ECP) experts early and often will result in faster adoption of best practices for exascale machines, and influence ECP design choices to HEP's benefit. HEP needs a coordinated interface to both ECP & the Leadership Computing Facilities.
  - Need to identify which codes could benefit the most
- LQCD regularly rewrites its code, has reaped significant speedup benefits every time
- Reinforced that multiyear NERSC allocations & better metrics for pledges are needed
- End-to-end network data flow models are needed to support tradeoff analysis of storage vs. CPU vs. network bandwidth on a system-wide and program-wide basis
  - Greater sharing of the underlying data management software layer may also be beneficial

U.S. DEPARTMENT OF ENERGY | Office of Science   DOE HEP Status at HEPAP - May 2018   28

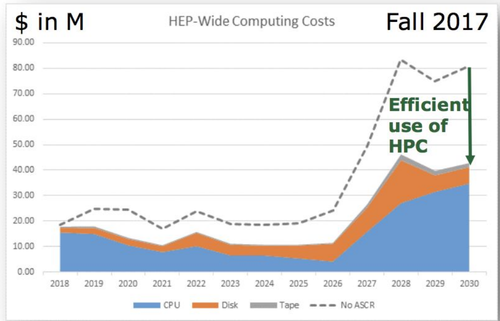
We must use them properly (use the accelerators)

We must use them heavily



Updated HEP Computing Model

- In preparation for the Inventory Roundtable, the largest HEP experiments from all three frontiers were asked to provide a **more detailed estimate** of their expected computing needs
  - CPU, storage, network, personnel, and HPC portability
- Cost estimates for all experimental frontiers:
  - "Business as usual" (minimal additional HPC use): **$600M ± 150M**
  - With effective use of HPC resources this reduces to: **$275M ± 70M**
- By 2030 cost share by frontier is estimated to be:
  - ½ Energy Frontier
  - ¼ Intensity Frontier
  - ¼ Cosmic Frontier
- **A strategy encompassing all HEP computing needs is required!**



U.S. DEPARTMENT OF ENERGY | Office of Science   DOE HEP Status at HEPAP - May 2018   29

Jim Siegrist, HEPAP meeting, May 2018

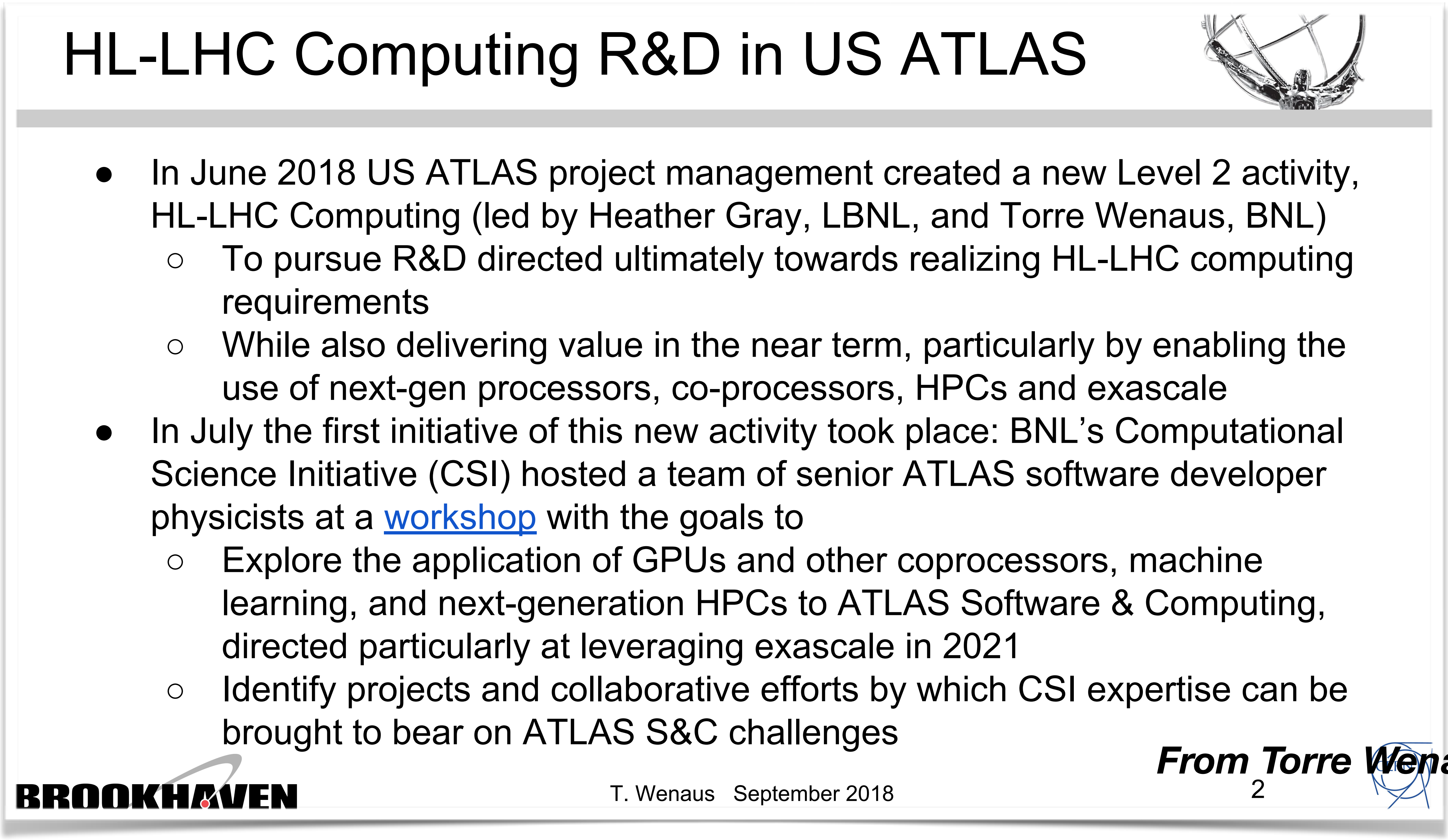
BROOKHAVEN   T. Wenaus September 2018

From Torre Wenaus

Accelerator utilization

- A big topic today and will be bigger in the future
- An example of why: the DOE tells us (LHC computing) **we must utilize the accelerators if we're to be allowed onto exascale machines**
  - Not unreasonably; most of their power is in the accelerators
- We're **not in a position today to use accelerators at large scale in offline**
  - ATLAS has no offline production applications today that utilize GPUs
  - ALICE will have GPU based online track reco for Run-3
  - CMS is partially rewriting tracking and calo sw to use GPUs, and studying larger scale adoption in the computing model studies underway (ECoM2X)
- The DOE position prompted new activity, again an ATLAS example…
  - In July 2018 the new "HL-LHC Computing" activity area in US ATLAS organized a workshop that brought together HPC experts from BNL's Computational Science Initiative (CSI) and a team of senior ATLAS software developer / physicists
    - Look for **GPU and ML applications for exascale** and identify projects

The Context



HL-LHC Computing R&D in US ATLAS

- In June 2018 US ATLAS project management created a new Level 2 activity, HL-LHC Computing (led by Heather Gray, LBNL, and Torre Wenaus, BNL)
  - To pursue R&D directed ultimately towards realizing HL-LHC computing requirements
  - While also delivering value in the near term, particularly by enabling the use of next-gen processors, co-processors, HPCs and exascale
- In July the first initiative of this new activity took place: BNL's Computational Science Initiative (CSI) hosted a team of senior ATLAS software developer physicists at a workshop with the goals to
  - Explore the application of GPUs and other coprocessors, machine learning, and next-generation HPCs to ATLAS Software & Computing, directed particularly at leveraging exascale in 2021
  - Identify projects and collaborative efforts by which CSI expertise can be brought to bear on ATLAS S&C challenges

T. Wenaus September 2018

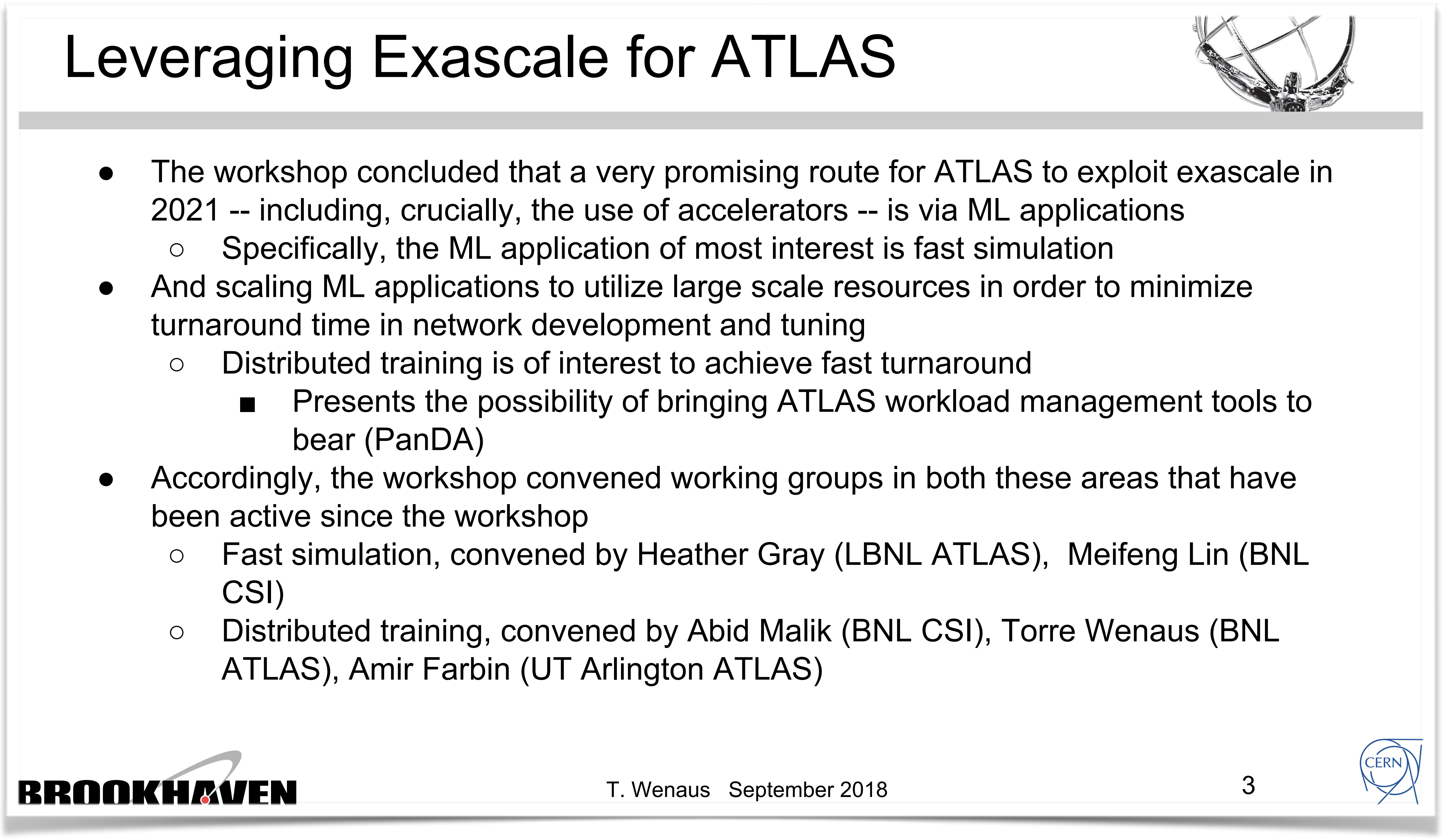
2

From Torre Wenaus

Also see:

https://indico.cern.ch/event/646942/contributions/2974809/attachments/1635055/2686755/ATLASSWCWeek-Wells_FINAL.pdf

The Plan



Leveraging Exascale for ATLAS

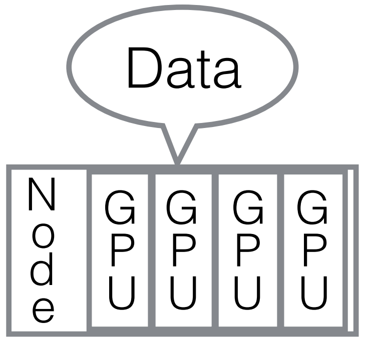
- The workshop concluded that a very promising route for ATLAS to exploit exascale in 2021 -- including, crucially, the use of accelerators -- is via ML applications
  - Specifically, the ML application of most interest is fast simulation
- And scaling ML applications to utilize large scale resources in order to minimize turnaround time in network development and tuning
  - Distributed training is of interest to achieve fast turnaround
    - Presents the possibility of bringing ATLAS workload management tools to bear (PanDA)
- Accordingly, the workshop convened working groups in both these areas that have been active since the workshop
  - Fast simulation, convened by Heather Gray (LBNL ATLAS), Meifeng Lin (BNL CSI)
  - Distributed training, convened by Abid Malik (BNL CSI), Torre Wenaus (BNL ATLAS), Amir Farbin (UT Arlington ATLAS)

BROOKHAVEN T. Wenaus September 2018 3 CERN

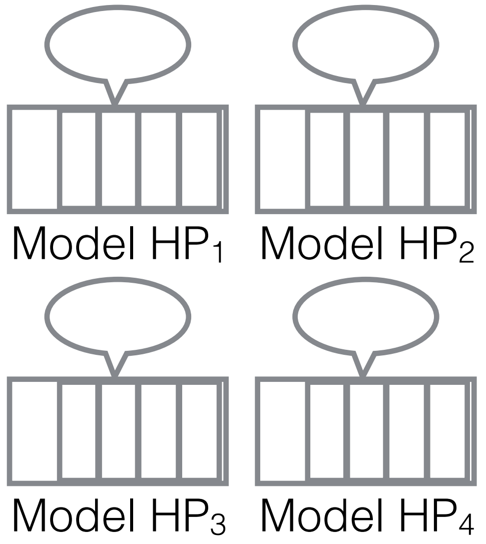
- Distributed Training and Hyper-parameter optimization on PANDA.
- PANDA already interfaces to HPCs.
- Goal is to be experiment agnostic.

Distributed Training

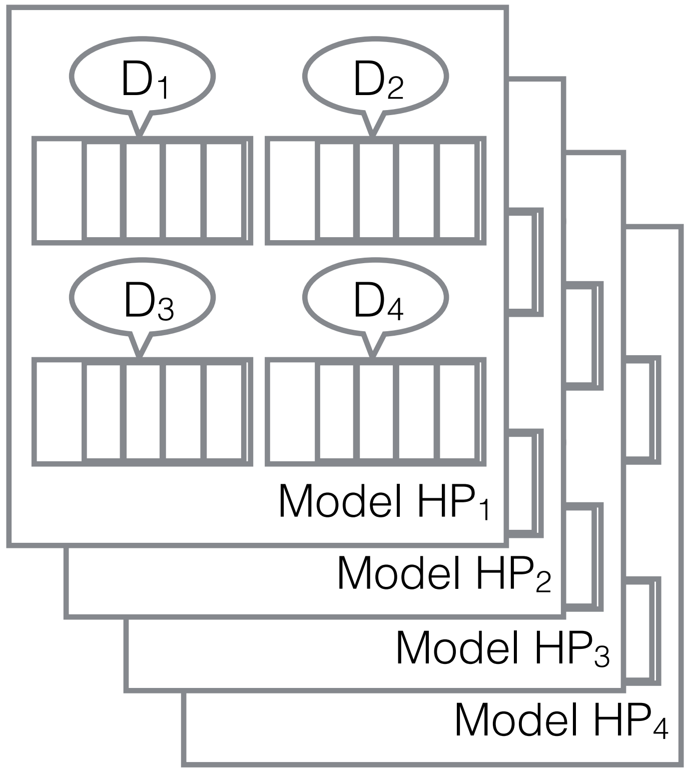
1. *Tensor operation parallelism*: GPUs, FPGA, and ASICs (Google's Tensor Processing Unit).
   - Note additional HN, Data, Model parallelism with multi-GPU

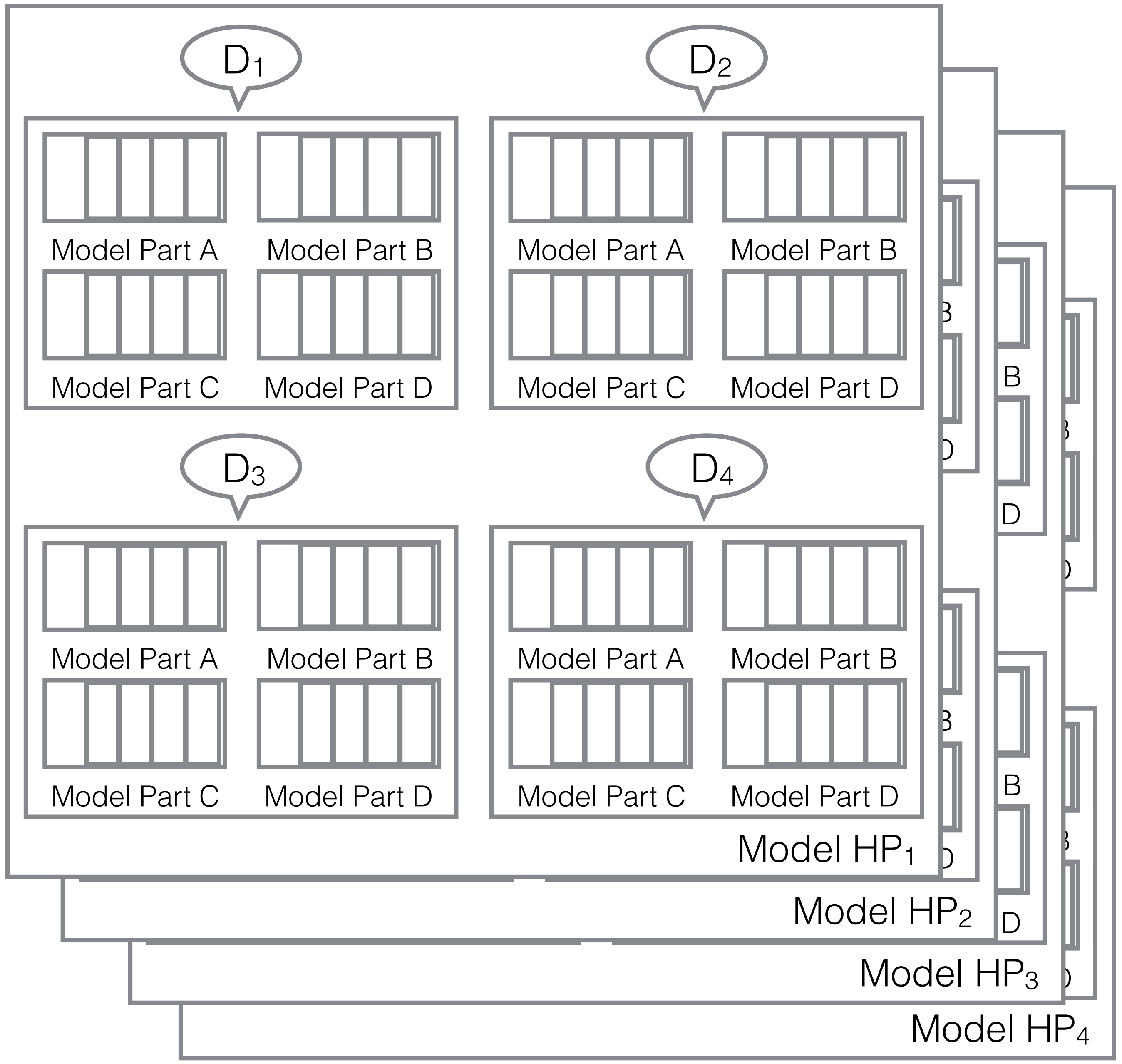


2. *Hyper-parameter scan*: simultaneously train multiple models. *e.g.* 1 model per GPU or node.



3. *Data Parallelism*: Each GPU or Node computes gradient on sub-set of data. Sync'ing gradients bottlenecked by bus or network.





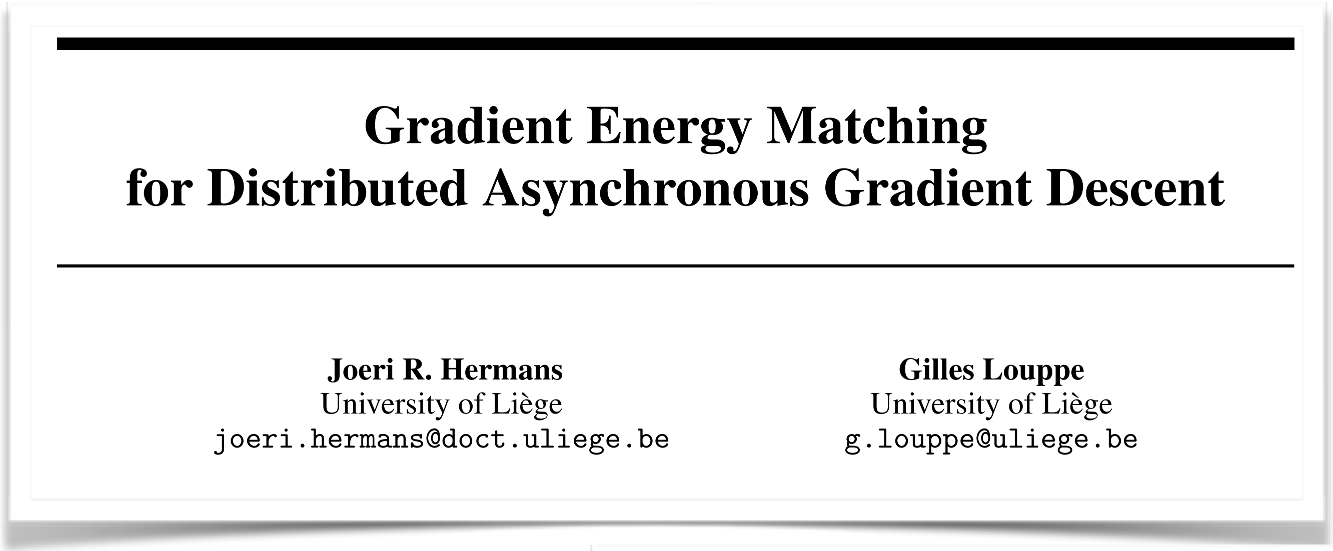
4. *Model Parallelism*: Large model spread over many GPUs or nodes. Less network traffic but only efficient for large models.

A Vision

- User sets up a training session in similar manner as current GRID submissions.
  - Define resources required. e.g. CPUs / GPUs per training job.
  - Define training data samples. Use Distributed Data Management system (i.e. Rucio).
- Hyper-parameters and measured optimization metrics on test/validation samples are book-kept and reported to PANDA via appropriate API.
- Hyper-parameter optimization can be either provided as part of service or run externally via the API.
- Processing performance metrics (e.g. time per epoch) are also reported to the system via API and monitored.
- Trained Models and any results (e.g. plots), are stored in DDM.

Distributed Training

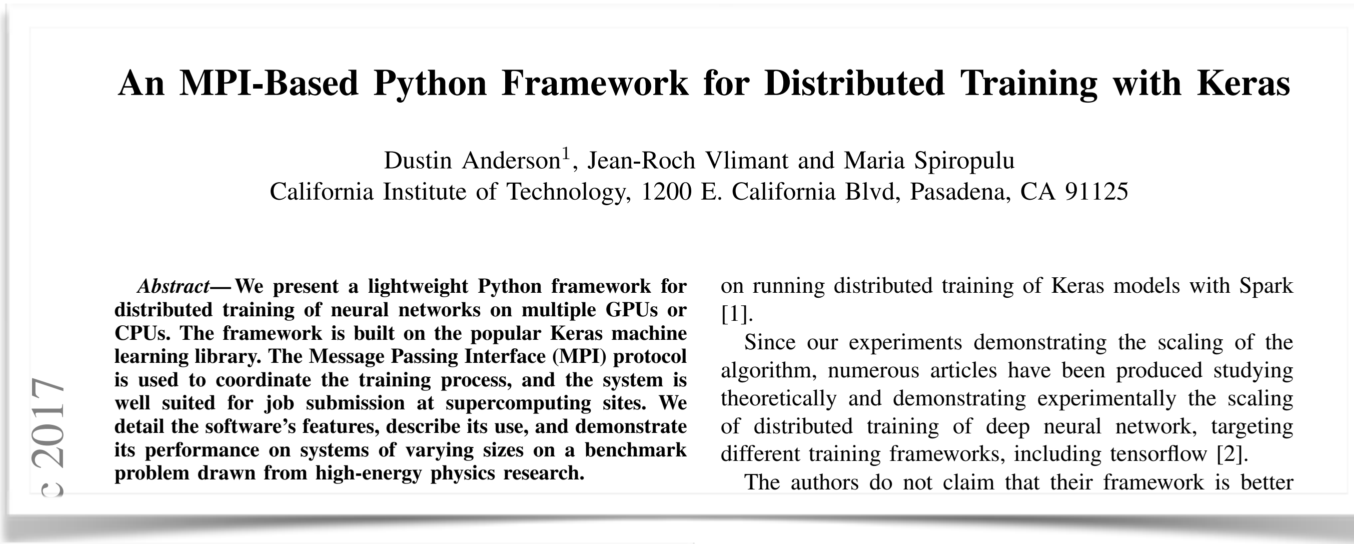
- Existing tools and techniques:
  - *Horovod:* from Uber.
  - *MPILearn:* from HEP colleagues at CalTech
  - From Giles' group: https://arxiv.org/pdf/1805.08469.pdf
  - From LBL/NERSC colleagues: https://arxiv.org/pdf/1708.05256.pdf
- In general, scaling to large number of nodes can be difficult.



Gradient Energy Matching for Distributed Asynchronous Gradient Descent

Joeri R. Hermans
University of Liège
joeri.hermans@doct.uliege.be

Gilles Louppe
University of Liège
g.louppe@uliege.be



An MPI-Based Python Framework for Distributed Training with Keras

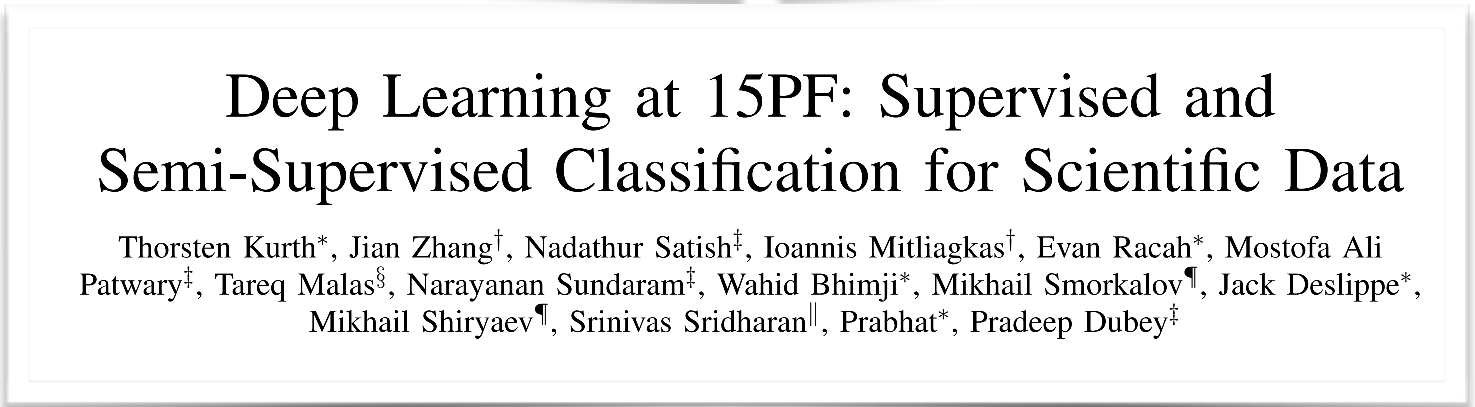
Dustin Anderson[1], Jean-Roch Vlimant and Maria Spiropulu
California Institute of Technology, 1200 E. California Blvd, Pasadena, CA 91125

Abstract—**We present a lightweight Python framework for distributed training of neural networks on multiple GPUs or CPUs. The framework is built on the popular Keras machine learning library. The Message Passing Interface (MPI) protocol is used to coordinate the training process, and the system is well suited for job submission at supercomputing sites. We detail the software's features, describe its use, and demonstrate its performance on systems of varying sizes on a benchmark problem drawn from high-energy physics research.**

on running distributed training of Keras models with Spark [1].

Since our experiments demonstrating the scaling of the algorithm, numerous articles have been produced studying theoretically and demonstrating experimentally the scaling of distributed training of deep neural network, targeting different training frameworks, including tensorflow [2].

The authors do not claim that their framework is better



Deep Learning at 15PF: Supervised and Semi-Supervised Classification for Scientific Data

Thorsten Kurth*, Jian Zhang†, Nadathur Satish‡, Ioannis Mitliagkas†, Evan Racah*, Mostofa Ali Patwary‡, Tareq Malas§, Narayanan Sundaram‡, Wahid Bhimji*, Mikhail Smorkalov¶, Jack Deslippe*, Mikhail Shiryaev¶, Srinivas Sridharan‖, Prabhat*, Pradeep Dubey‡

Short Term Plan

- Identify 2 types of problem:
    1. *Quick training:* few hours training times where large scale Hyper-parameter optimization is warranted.
        - Build out system for training submission, hyper-parameter management, and monitoring.
    2. *Long training:* days/weeks of training where distributed training is warranted.
        - Study scaling / speed up and integrate into system.
- Do you have a candidate problem?
    - Extensive Hyper-parameter optimization and/or long training times.
    - We already have some candidate problems in ATLAS… but yours may be better suited.
        - Can’t talk about it… this is a public session.
    - Once we have working solutions, will need people to try it.

Extras

Details

- 2 “use cases”. Factorizable…

  1. Hyper-parameter optimization
     - Run large number of single GPU training jobs, each with unique architecture.
     - Jobs “publish” metrics to database.
     - User or optimization suite, manually or automatically queries database, submits next set of training jobs.
  2. [Data Parallel] Distributed Training
     - Training data split into n sub-sets.
     - n GPUs within same facility (preferably same node), run simultaneously.
     - May designate one node as “parameter” server.
     - Replicate setup for hyper-parameter optimization.