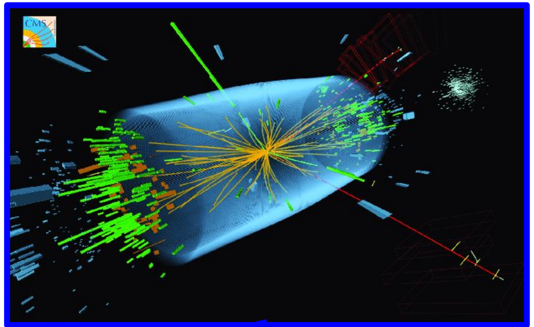
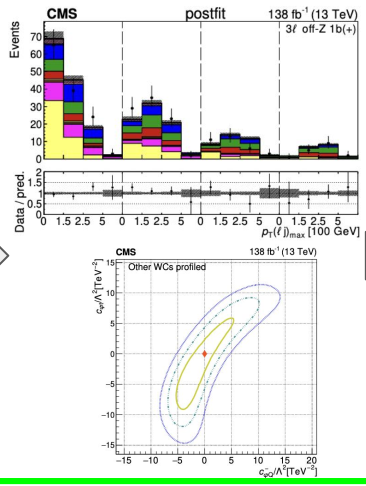
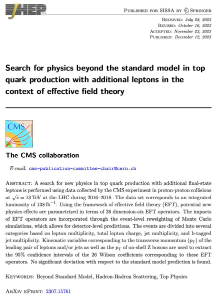
Reshaping Analysis for Fast Turnaround

Kevin Lannon, Connor Moore, Barry Sly-Delgado,
Douglas Thain, Benjamin Tovar, Austin Townsend, Jin Zhou

Analysis Computing






JHEP — Published for SISSA by Springer

Received: July 28, 2023
Revised: October 16, 2023
Accepted: November 23, 2023
Published: December 12, 2023

Search for physics beyond the standard model in top quark production with additional leptons in the context of effective field theory

The CMS collaboration

E-mail: cms-publication-committee-chair@cern.ch

Abstract: A search for new physics in top quark production with additional final-state leptons is performed using data collected by the CMS experiment in proton-proton collisions at $\sqrt{s} = 13$ TeV at the LHC during 2016–2018. The data set corresponds to an integrated luminosity of 138 fb^{-1}. Using the framework of effective field theory (EFT), potential new physics effects are parametrized in terms of 26 dimension-six EFT operators. The impacts of EFT operators are incorporated through the event-level reweighting of Monte Carlo simulations, which allows for detector-level predictions. The events are divided into several categories based on lepton multiplicity, total lepton charge, jet multiplicity, and b-tagged jet multiplicity. Kinematic variables corresponding to the transverse momentum (p_T) of the leading pair of leptons and/or jets as well as the p_T of on-shell Z bosons are used to extract the 95% confidence intervals of the 26 Wilson coefficients corresponding to these EFT operators. No significant deviation with respect to the standard model prediction is found.

Keywords: Beyond Standard Model, Hadron-Hadron Scattering, Top Physics

ArXiv ePrint: 2307.15761

Focus of this talk

Production:

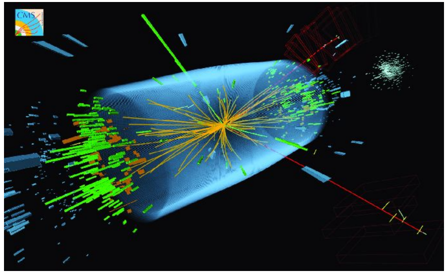
- High performance algorithms written in C++
- Input: Petabytes of data
- Output: Terabytes of analysis data (NanoAOD)
- Computing Scale: Millions of jobs running over long time scales operated by small team of experts

Analysis:

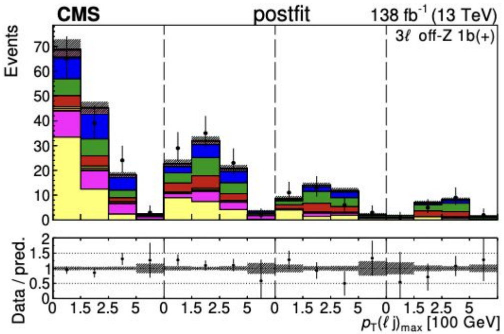
- User code written in Python and/or C++, includes histogram filling, fitting, and visualization
- Input: Terabytes of analysis data (NanoAOD)
- Output: Kilobytes of histograms, data tables, etc.
- Computing scale: Thousands of jobs running over relatively short time scales (hours) run by many different non-expert users (e.g. grad students)

UNIVERSITY OF NOTRE DAME

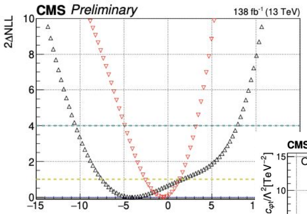
Drilling Down in Analysis

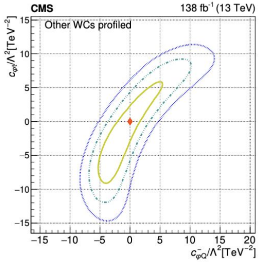


Analysis data format (nanoAOD)



Histograms





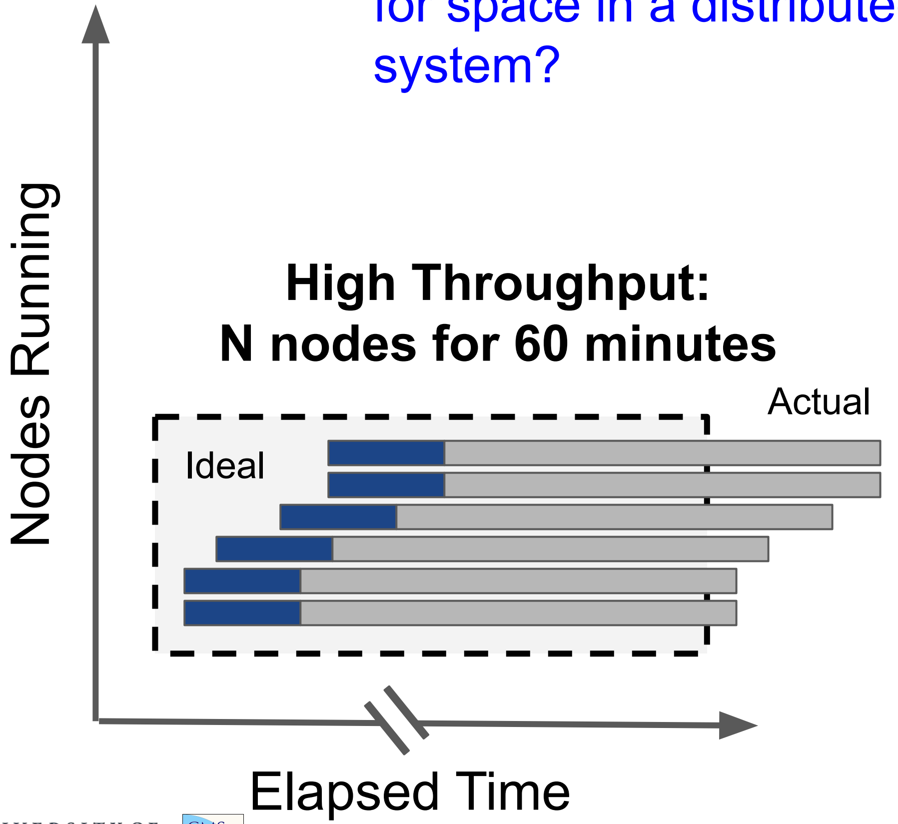
Statistical analysis & visualization

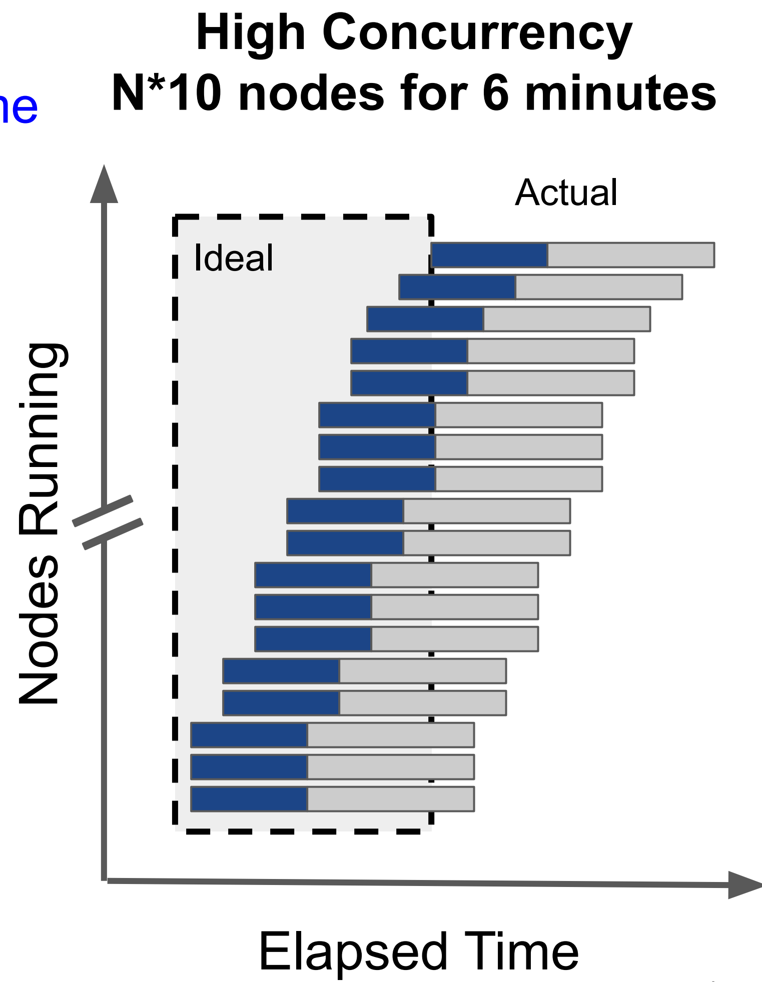
Focus on this step:

- Applying corrections and selecting events
- Calculating quantities of interest
- Filling histograms

Reshaping

How easily can you trade time for space in a distributed system?





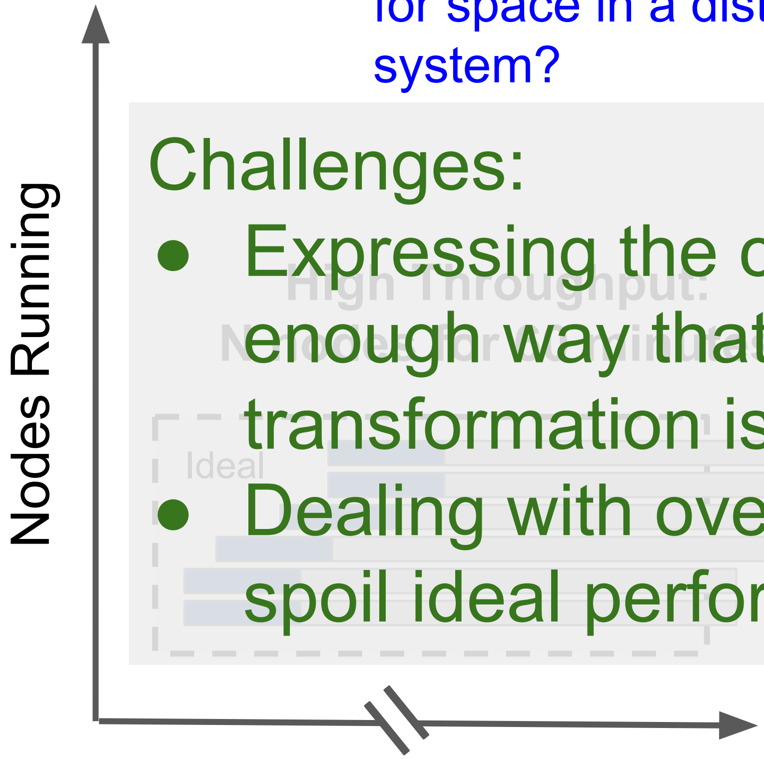
(Courtesy of Doug Thain)

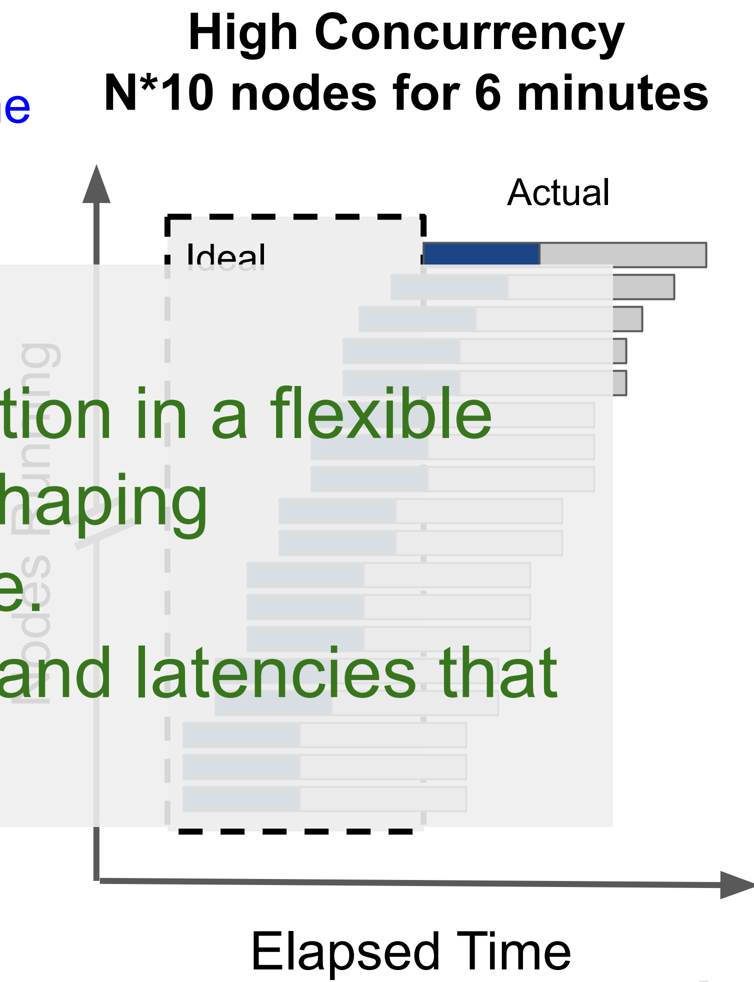
UNIVERSITY OF NOTRE DAME

CMS

Reshaping

How easily can you trade time for space in a distributed system?





(Courtesy of Doug Thain)

Challenges:

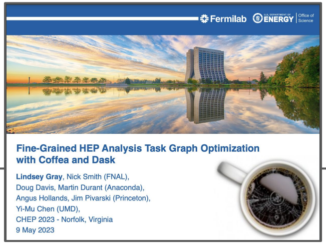
- Expressing the computation in a flexible enough way that the reshaping transformation is feasible.
- Dealing with overheads and latencies that spoil ideal performance

UNIVERSITY OF NOTRE DAME

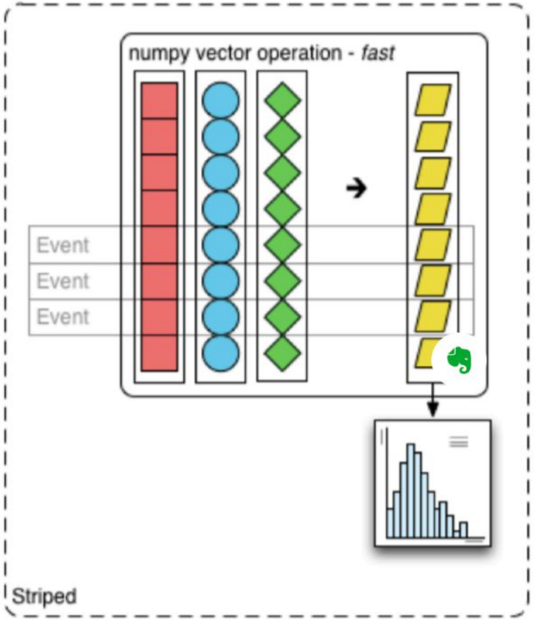
CMS

Analysis Software Paradigm

- Columnar analysis:
  - Load relevant values for many events into contiguous arrays
  - Evaluate several **array programming** expressions
    - Implicit *inner* loops
    - Plan analysis by composing data manipulations
  - Store derived values

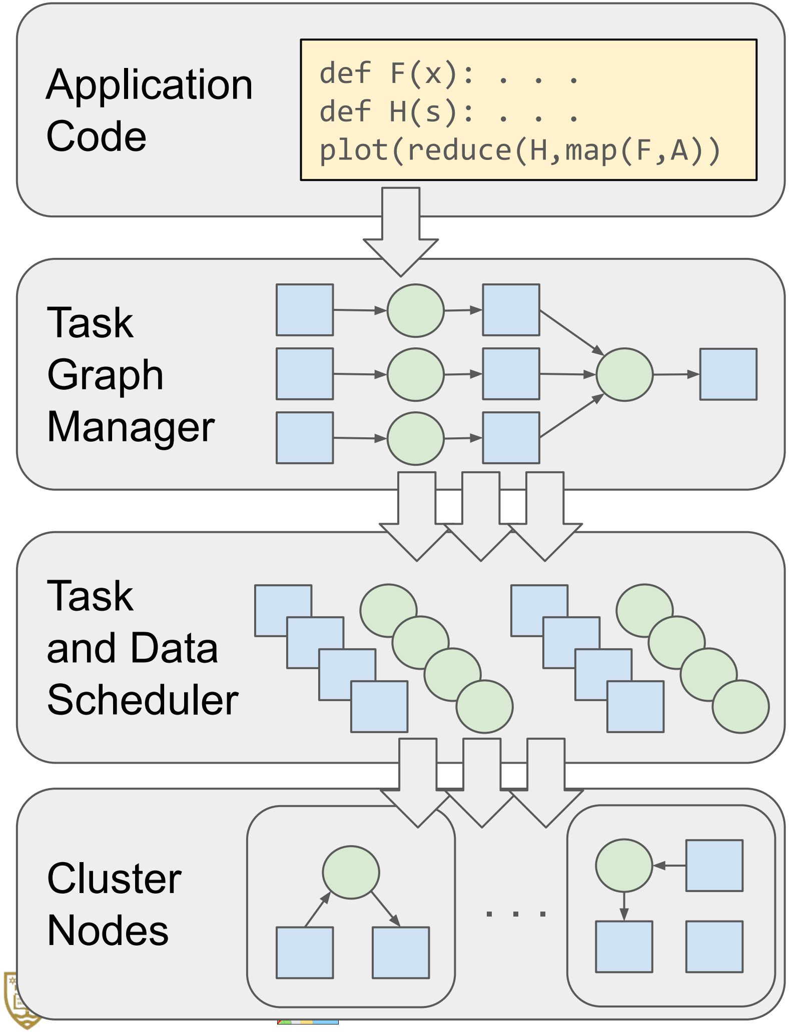


See CHEP23 talk on Coffea + Dask



User writes "numpy-like" code → converted into form easily executed on distributed resources.

UNIVERSITY OF NOTRE DAME
CMS

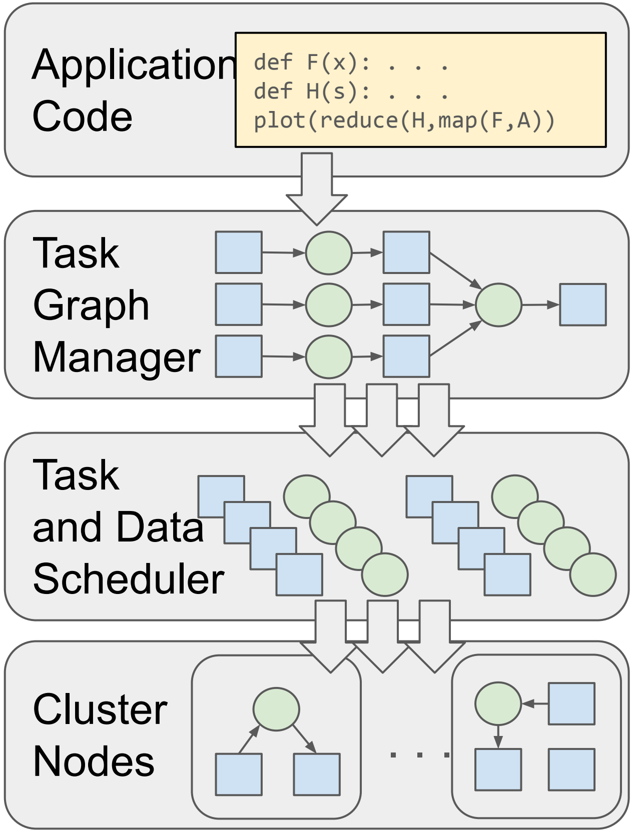


(Courtesy of Doug Thain)

Software Stack (conceptually)

- **Application code:** User writes "regular" numpy-like software expressing physics intent
- **Task Graph Manager:** Computations decomposed into graph form
- **Task and Data Schedule:** Graph is used to move data and schedule computation.
- **Cluster Nodes:** Computation is executed on data to produce results.

Software Stack



Coffea

Awkward Array

dask

dask-awkward

TaskVine

HTCondor Software Suite

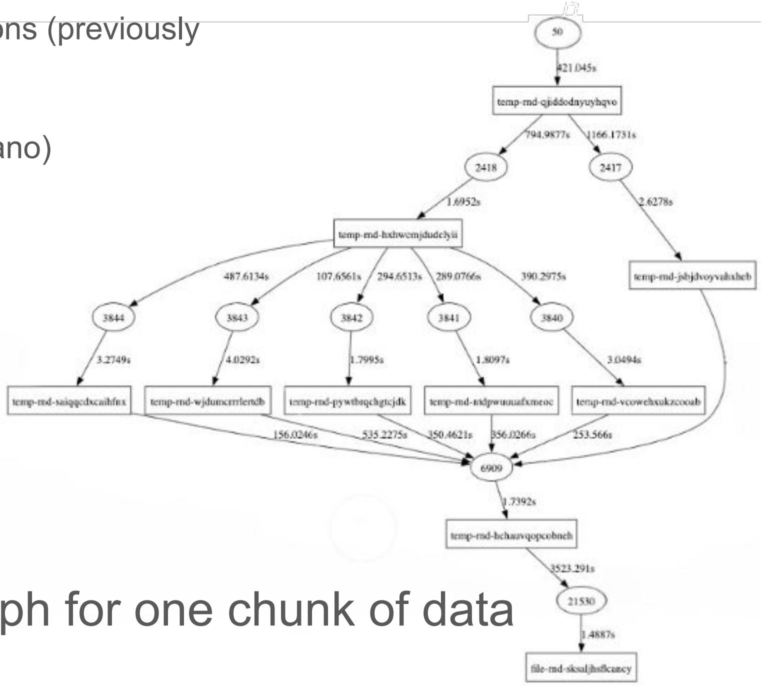
UNIVERSITY OF NOTRE DAME

CMS

Example Application: DV

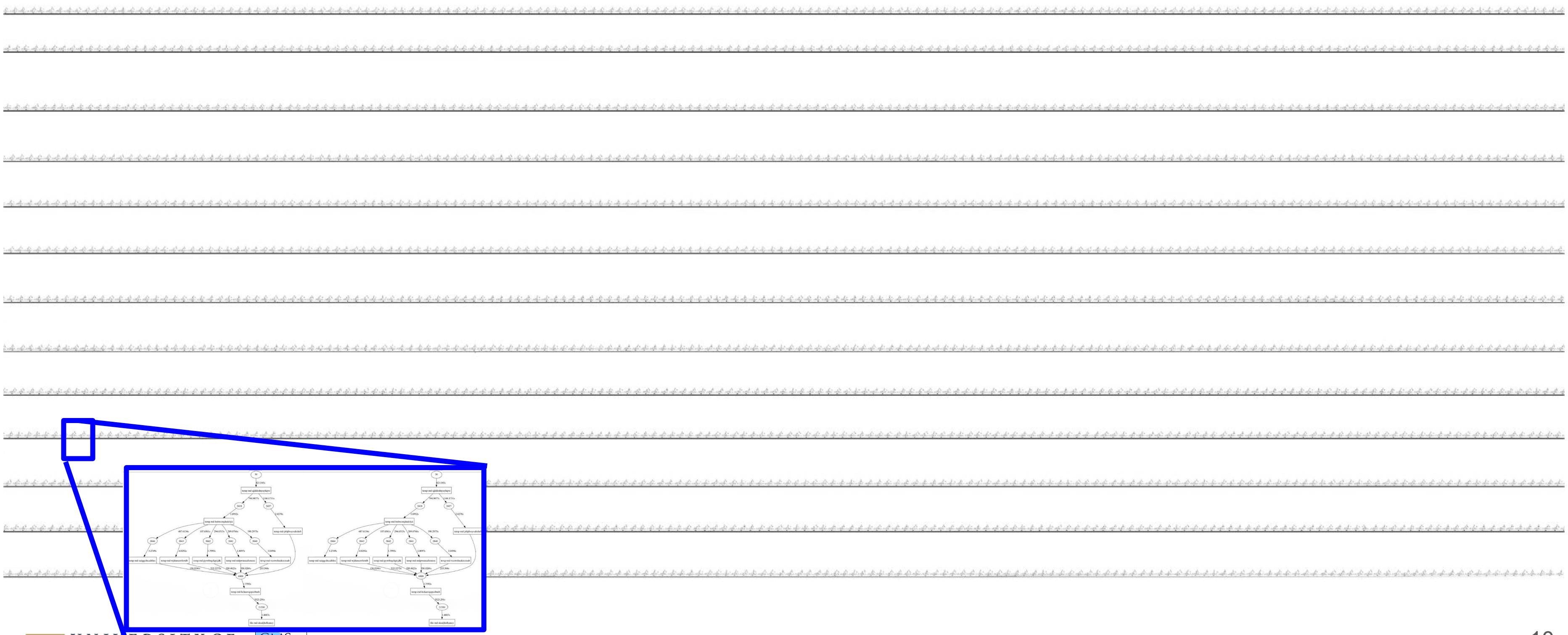
Connor Moore

- DV application calculates energy correlation functions (ECFs) on jets
  - ECFs probe jet substructure
  - Calculated using jet constituents (PFCandidates)
  - Computationally heavy–calculating up to 5-point correlations (previously considered infeasible)
- Input:
  - Modified analysis format that stores jet constituents (PFnano)
  - ~20 million (160 GB) – one dataset
- Output:
  - 5.7 million events (7.6 GB)
  - ~160 ECFs stored in parquet files
  - To be used for ML training
- Resources
  - 4-6k CPU cores
  - 400 GB disk
  - 2 GB/core memory
  - 6-8 hours



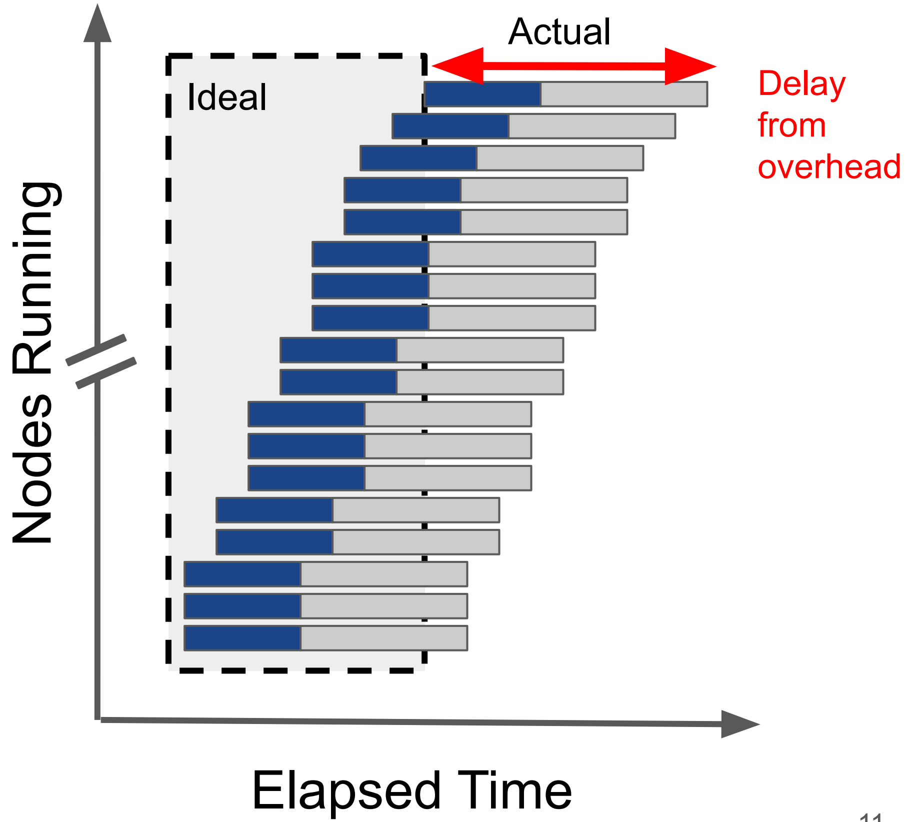
Graph for one chunk of data

Full Graph for DV (All Chunks)



Performance requires on dealing with overhead

- Key is being able to start many tasks quickly with low overhead
- Overhead and latency starting tasks can dramatically slow performance, limiting the benefits of high concurrency

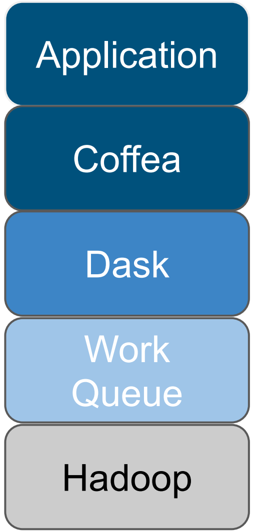


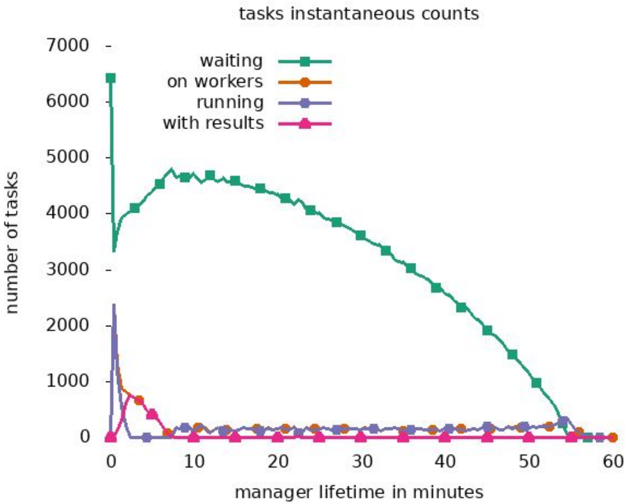
(Courtesy of Doug Thain)

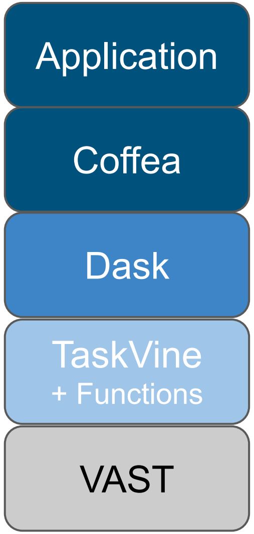
UNIVERSITY OF NOTRE DAME

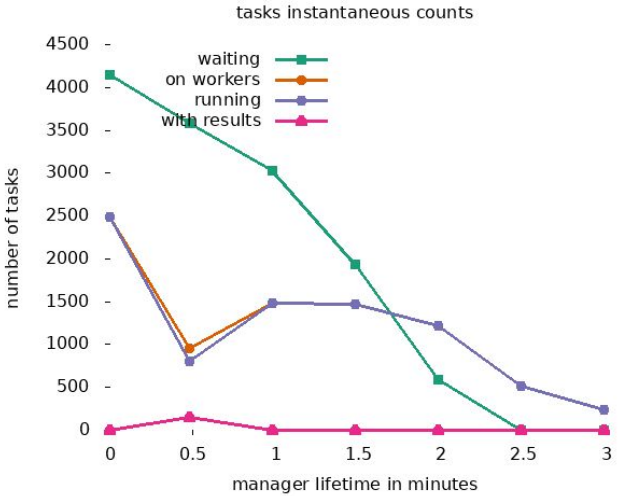
CMS

Performance optimization







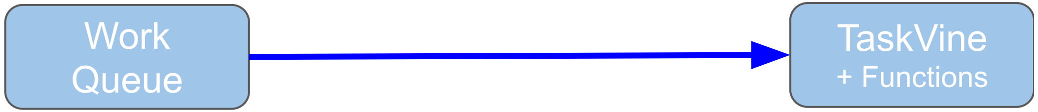


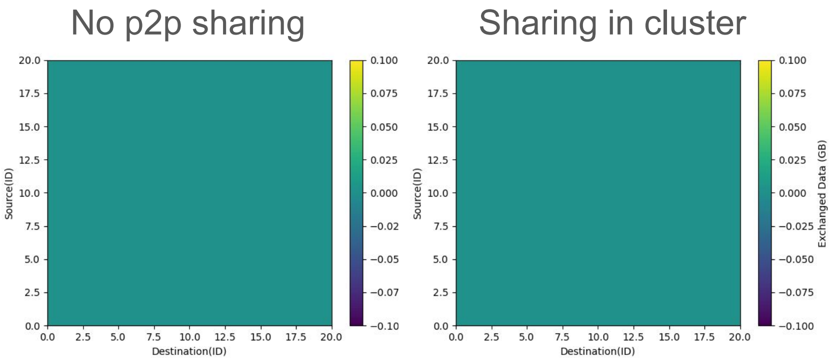
Significant performance improvement within same application by swapping out bottom two layers (task/data scheduler and file system

UNIVERSITY OF NOTRE DAME CMS

Talk at SC24 ⟵ Barry Sly-Delgado

Performance optimization

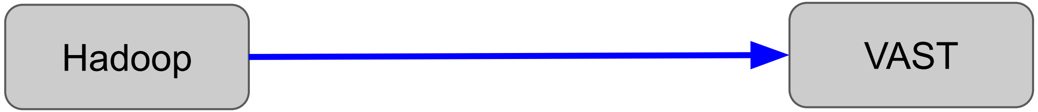




TaskVine + Functions:

- More efficient distribution of graph payloads
- Smarter data caching
- Peer-to-peer data sharing between workers in the cluster.

Reducing overhead and latency improved performance by 20x in this example!



Vast: high performance NVMe storage compared to Hadoop on spinning disks

UNIVERSITY OF NOTRE DAME

Potential for further optimization

- Having graph representation of tasks opens many possibilities
  - Analyze graph structure and reorganize for better performance (e.g. minimize I/O)
  - Intelligent checkpointing and caching accelerating graph evaluation under small variations in analysis code
  - Exploring alternative graph scheduling strategies (breadth first vs depth first): maximizing performance versus satisfying constraints on storage or memory
  - Improving performance under worker failure by caching some results in shared storage
  - Intelligently scheduling graph nodes on appropriate resources (GPU vs CPU)
- Only just beginning to tap potential of this approach!

Challenges to be tackled

- Intermediate data products transferred between task nodes via file system: can result in large temporary storage requirements and compression robs you of processing time
- With current tools, extremely difficult to correlate task failures with lines in source code. Need improved debugging capabilities to link graph to original source code statements
- Communicating the right information back to the users so that they can spot and resolve performance bottlenecks (I/O, memory, or storage limitations)

Conclusions

- Columnar analysis → Task graphs represents an exciting new paradigm in writing analysis software
- Task graphs offer rich and flexible expression of computing tasks that appear to be highly amenable to automated analysis and optimization, such as reshaping
- The effectiveness of reshaping relies on our ability to minimize overhead and latency at beginning and end of tasks
- Stay tuned for further developments!

The Team

Notre Dame CMS Graduate Students

Connor Moore

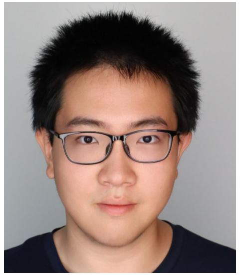
Austin Townsend

Notre Dame Cooperative Computing Lab

Doug Thain
(Director)

Ben Tovar
(Res. Software Eng.)

Barry Sly-Delgado
(Grad Student)



Jin Zhou
(Grad Student)