OSG All-Hands Meeting, July 12, 2023

Exploring PB-scale data processing with GlueX on shared OSG resources

Richard Jones, University of Connecticut

- the promise
- the challenge
- a feasibility study

This work is supported by the U.S. National Science Foundation under grant 2209480

Background: *the Gluex experiment*

GlueX at Jefferson Lab

- 9 GeV photons, fixed target
- *Scientific program*
    - search for exotic mesons
    - threshold charmionium production
    - dark matter searches, rare η decays
    - Primakov, pion polarizability

installed: 2012-2014

commissioned: 2014-2016

approved running: 2017-2025

This work is supported by the National Science Foundation under grant 2209480

Evolution: *beyond simulation?*

Open Science Grid

GlueX offline computing resource needs *(GlueX-doc-3813)*

- 35 Mcore-hr/yr - detector simulation, primarily on OSG
- 130 Mcore-hr/yr - experimental data reconstruction
  - *Jefferson Lab compute facility (total 70 Mcore-hr/yr, all experiments)*
  - *NERSC (proven option, but allocations are limited)*
  - *other ??*

In the future, maybe OSG can contribute to the greater need here

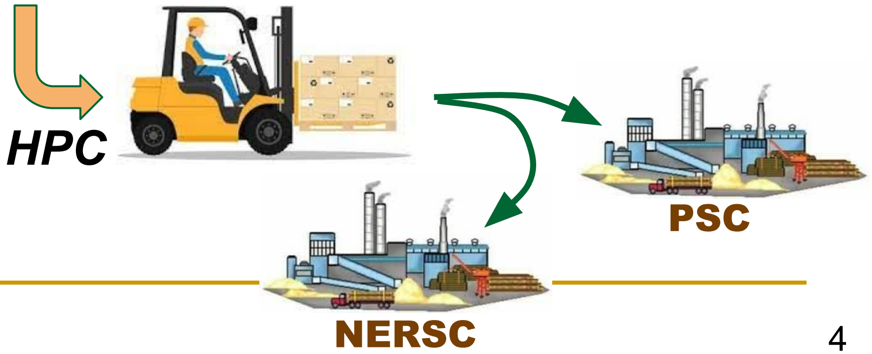
- This is intrinsically a **HTC problem**
- To solve it we are *relying primarily on HPC resources (NERSC, PSC, IU)*
- Why not expand capacity using OSG resources?

This work is supported by the National Science Foundation under grant 2209480

raw data reconstruction on shared HTC

the challenge of data distribution:

- 1.0GB/s staged on tape at Jlab
- 150MB / core-hr reconstruction rate
- 65k cores in steady-state to keep up
- *Is this scale feasible on shared HTC?*



This work is supported by the National Science Foundation under grant 2209480

Gluex challenge: dataflow choreography

Goal: *port an application optimized for HPC to run on HTC*

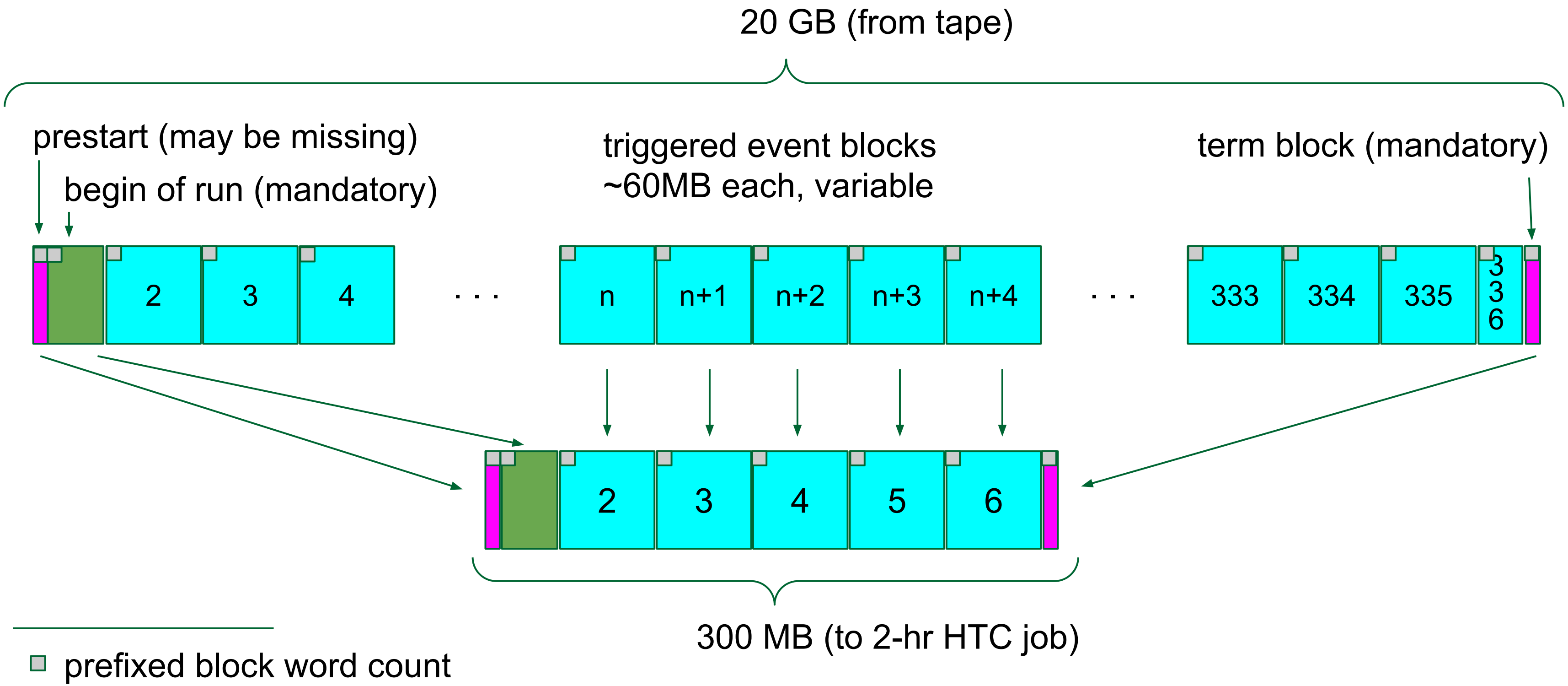
- output must be byte-for-byte identical to HPC for same input data
- source code must be identical to HPC, binary compatibility not required

Data distribution strategy:

1. ***Split*** up the input data into schedulable chunks (2 core-hr)
   - 20GB evio file = 1.4M events = 130 core-hr
   - 340 evio blocks / file
   - 5 evio blocks / job
   - 70 jobs / evio file
2. ***Merge*** results after processing
   - much smaller data volume
   - can be performed on the SE as part of post-job validation
   - utilities already exist for merging (eg. *hadd, eviocat,* etc.)

This work is supported by the National Science Foundation under grant 2209480

chunking a GlueX raw data stream over xrootd



This work is supported by the National Science Foundation under grant 2209480

how <u>NOT</u> to subset a 20GB raw data file

- ✘ Split data to 300MB files on the SE prior to job submission
  - ✘ introduces extra step in processing
  - ✘ doubles load on the SE
  - ✘ <u>*not necessary*</u>

- ✘ Tell event reader to skip forward *N* events before start of processing
  - ✘ increases total data transfer load by factor ~35
  - ✘ large cpu load increase from block unpacking
  - ✘ assumes we know in advance how many events per file

- ✘ Tell event reader to skip forward *N* blocks before start of processing
  - ✘ still need to know when to stop
  - ✘ requires modification of the reconstruction software
  - ✘ violates goal to run ***the same software release on all production sites***

This work is supported by the National Science Foundation under grant 2209480

an *effective, light-weight* solution

customize the XRootD Posix preload client library

- no modifications to the Gluex software stack
- no modifications to the xrootd protocol, OSDF infrastructure
- changes are local to the OSG singularity container
- correctness can be verified prior to running production
- maintainable as a fork of the XRootD client github repo

This work is supported by the National Science Foundation under grant 2209480

an *effective, light-weight* solution

what changed in the xrootd client interface?

- all existing preload library functionality retained
- added: recognition of a <u>specially formatted xrootd url</u> at open
- overlap of processing, fetching next block

To access the full input file, use the standard xrootd url

root://cn440.storrs.hpc.uconn.edu/gluex/rawdata/Run071728/hd_rawdata_071728_000.evio

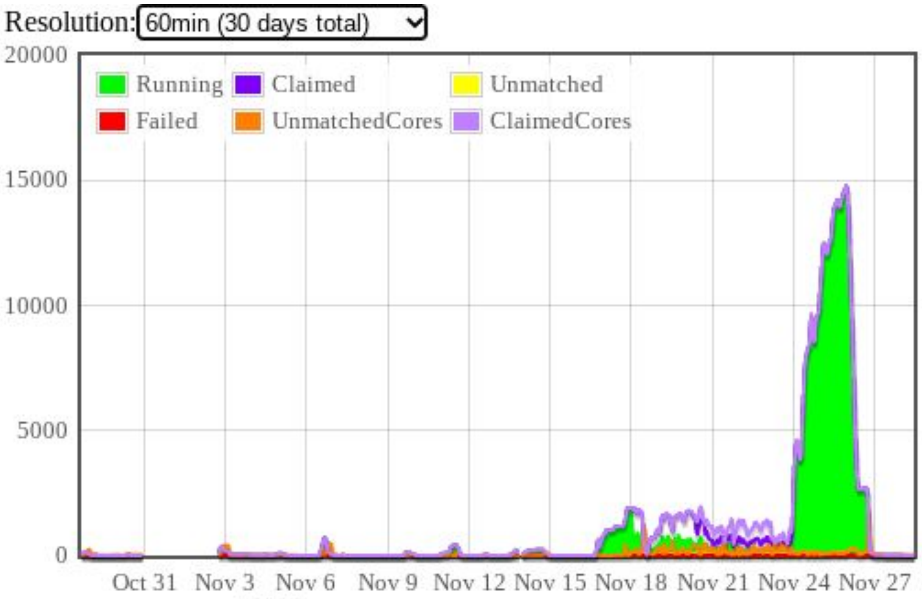
To get a subset with just 2 evio blocks + BOR + term

root://cn440.storrs.hpc.uconn.edu/gluex/rawdata/Run071728/hd_rawdata_071728_000.evio **:14,16**

This work is supported by the National Science Foundation under grant 2209480

a feasibility study

- 10 runs reconstructed on OSG
  - 1880 raw 20GB files
  - 35 TB, 600M events
  - 500k core-hr
- mapping onto osg user jobs
  - 4 cpu-hr / 1-core job
  - 68 jobs / evio file
  - 126,000 jobs
- highlights
  - completed in 3 days, see plot
  - 15,000 running cores, peak
  - job completion rate >90%



This work is supported by the National Science Foundation under grant 2209480

what happened in this time?

- data had already been staged from JLab to UConn *(staging time not included)*
- data distribution governed by a postgres database
    a. jobs “checked out” chunks of data via database transaction at job start
    b. successful completion of chunks registered to database at job exit
    c. reservations expired periodically, uncompleted chunks recycled
- *all jobs were identical, no predefined job-data association*

what happened in this time?

1. jobs were submitted from UConn submit node gluex.phys.uconn.edu
   a. shared same gluex factory with regular GlueX simulation jobs
   b. standard gluex queue, no special boosted privileges
   c. input data pulled during processing by osg worker
   d. output copied back to UConn SE at job exit

2. job outputs automatically merged
   a. performed by cron job running on the UConn-HTC cluster (colocated with SE)
   b. one-step merging, verification, and push of merged results to Jlab

3. not everything worked perfectly...

This work is supported by the National Science Foundation under grant 2209480

what worked well?

1. postgres database as a management hub
   a. jobs requested their work from the database when they started up
   b. jobs returned their exit code and completion time when they finished
   c. no problem with scaling
      i. ~5 jobs starting/finishing per second
      ii. merger cron job multiplicity scaled to keep up with job output
      iii. management was hands-off -- ***watch but do not touch!***
   d. only failed slices were reprocessed (1% of a full raw data evio file)
   e. recycling unprocessed slices was immediate -- within seconds!

2. output merging + verification + push to Jlab done in one step

This work is supported by the National Science Foundation under grant 2209480

where were the difficulties?

- ★ reconstruction app. memory footprint: RSS = 5.5GB @ 1 worker thread !!
  - reconstruction library alone = 1.4GB **-- *good!***
  - there are a lot of plugins -- 29 of them
  - most of the bloat comes from root histograms -- 17,000 mostly TH2I
  - remove the 5 worst offenders = 2.5GB -- ***kinda ok, not ideal***
- ★ thread explosion
  - reconstruction code had a strange behavior, required intervention
- ★ hanging transfers pushing output back to JLab
  - enagaged with UConn NetOps to resolve this
  - very difficult to diagnose problems at the JLab firewall from outside
  - connections were being unpredictably dropped -- ***hanging sockets***
  - investigated further with JLab NetOps -- ***hanging communication***

Results and assessment

- robust performance by redundant dcache xrootd server “doors”:
  - 6 instances: cn440...445, but only needed one
  - *door* functions only to forward connections to “movers” on “pool” hosts
  - input + output storage spread equally over 38 hosts
  - internal network 100Gb/s infiniband
  - external network 10Gb/s ethernet
- at peak, load remained reasonable (50 simultaneous transfers, 2 min. max)
- no long-lived sockets, all connections fetch just one block and disconnect
- no problems observed with database transaction rates, all good!
- ***BUT*** results were *not* byte-by-byte identical to HPC production -- *unrelated to OSG*

This work is supported by the National Science Foundation under grant 2209480

concluding comment: enabling role of OSDF

- UConn dcache SE is a OSDF origin (UConn-HPC_StashCache_origin)
  - 750TB of shared network-visible storage
  - used for staging input + output to/from tape @ Jlab
- data are read-once during reconstruction
  - raw data not suitable for caching
  - *nevertheless, OSDF data access was a major advantage!*
- xrootd access via OSDF worked well
  - many sites have limited direct network access to offsite
  - accessing through the OSDF url often factor 10 x faster than direct
  - sites have configured special routes for OSDF access -- *enables this capability*

This work is supported by the National Science Foundation under grant 2209480

Backup slides

This work is supported by the National Science Foundation under grant 2209480

an *effective, light-weight* solution

what XRootD components needed to be touched?

- github fork rjones30 / xrootd from master on 11/12/2020
- updates enclosed in #ifdef EVIO_BLOCK_SUBSET_EXTENSION
  - src/XrdPosix/XrdPosixFile.hh (40 added lines)
  - src/XrdPosix/XrdPosixFile.cc (37 added lines)
  - src/XrdPosix/XrdPosixXrootd.cc (237 added lines)