NGT Task 3.2

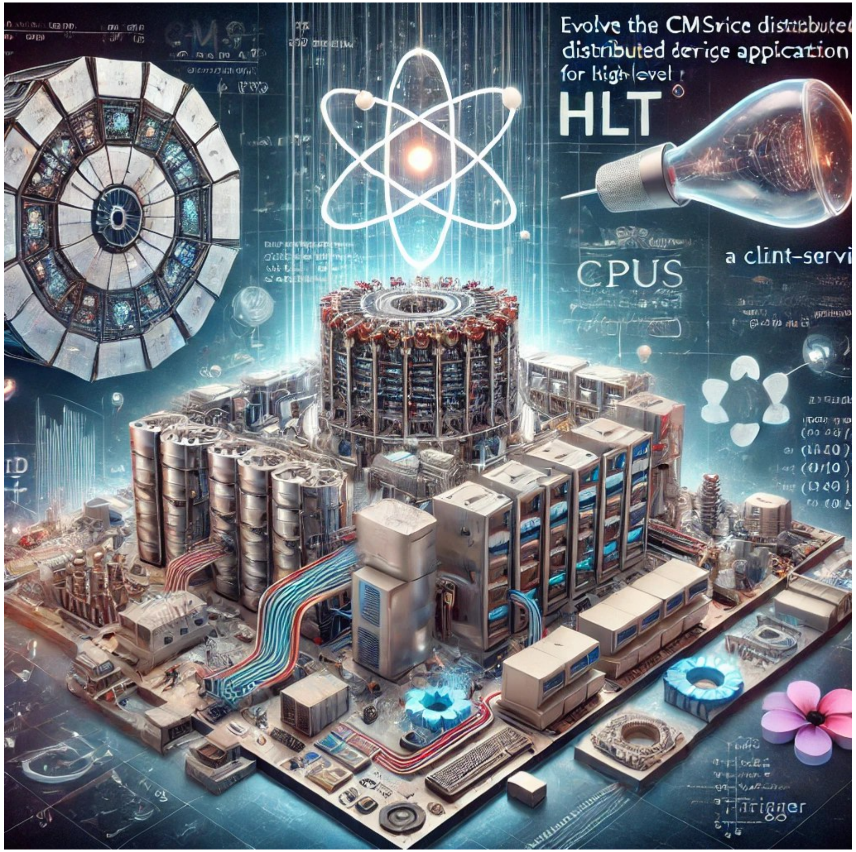
Evolving CMSSW into a client-server distributed application for HLT

A. Bocci

EP/CMD

CERN | NexTGen Next Generation Triggers

distributed CMS software



the CMS Software (CMSSW)

- modular
  - overall more than 5000 different “modules”
  - *e.g.* HLT configuration for Run-3 composed of 4600 instances of 380 different modules
- parallel
  - multithreaded, good scalability over 100 of threads
- heterogeneous
  - alpaka-based modules
  - single source, built for CPUs, NVIDIA GPUs, AMD GPUs
  - transparent backend choice at runtime
- R&D: extend to multi-process/multi-node
  - single logical application, multiple machines
  - ALICE O2 is multi-process via message passing
  - ATLAS uses MPI in HPC environments

use cases

original use case

- GPU-equipped HLT farm
  - balance the amount of memory and processing power available on CPU and GPU
  - fixed at time of procurement
  - the HLT configuration and code base evolves over the years
- alternative approach
  - offload part of the GPU-heavy computations to separate nodes
  - increase GPU processing power over time simply adding more nodes
  - leverage high-speed network interconnect to minimise the extra latecy

other use cases

- ease deployment at HPC sites
  - gradual transition to GPU usage, mixing CPU-only nodes and GPU-heavy nodes
  - use worker nodes with limited local disk space
    or outbound network access
- mix and match jobs with different parallelism requirements
  - multi-threaded “client” communicating with multiple single-threaded “servers”or vice-versa

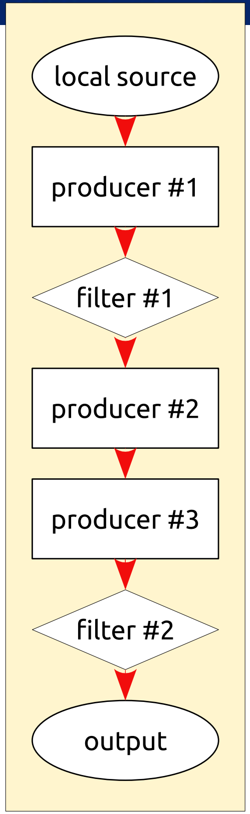
an ambitious goal

- design and implement a distributed application, with support for
  - sending and receiving arbitrary collections, including support inter-object references and provenance
  - multiple threads and multiple concurrent events (streams)
  - multiple senders and receivers per application
  - multiple clients and multiple servers with a non-trivial topology
  - efficient memory transfers to and from GPUs
  - event filtering
  - fault tolernce and error recovery

- with minimal impact on the existing code base
  - leverage the modular architecture of CMSSW and the Event Data Model approach
  - extend the framework capabilities
  - avoid rewriting and maintaining a dedicated reimplementation of “remote” algorithms

simplified diagram

from

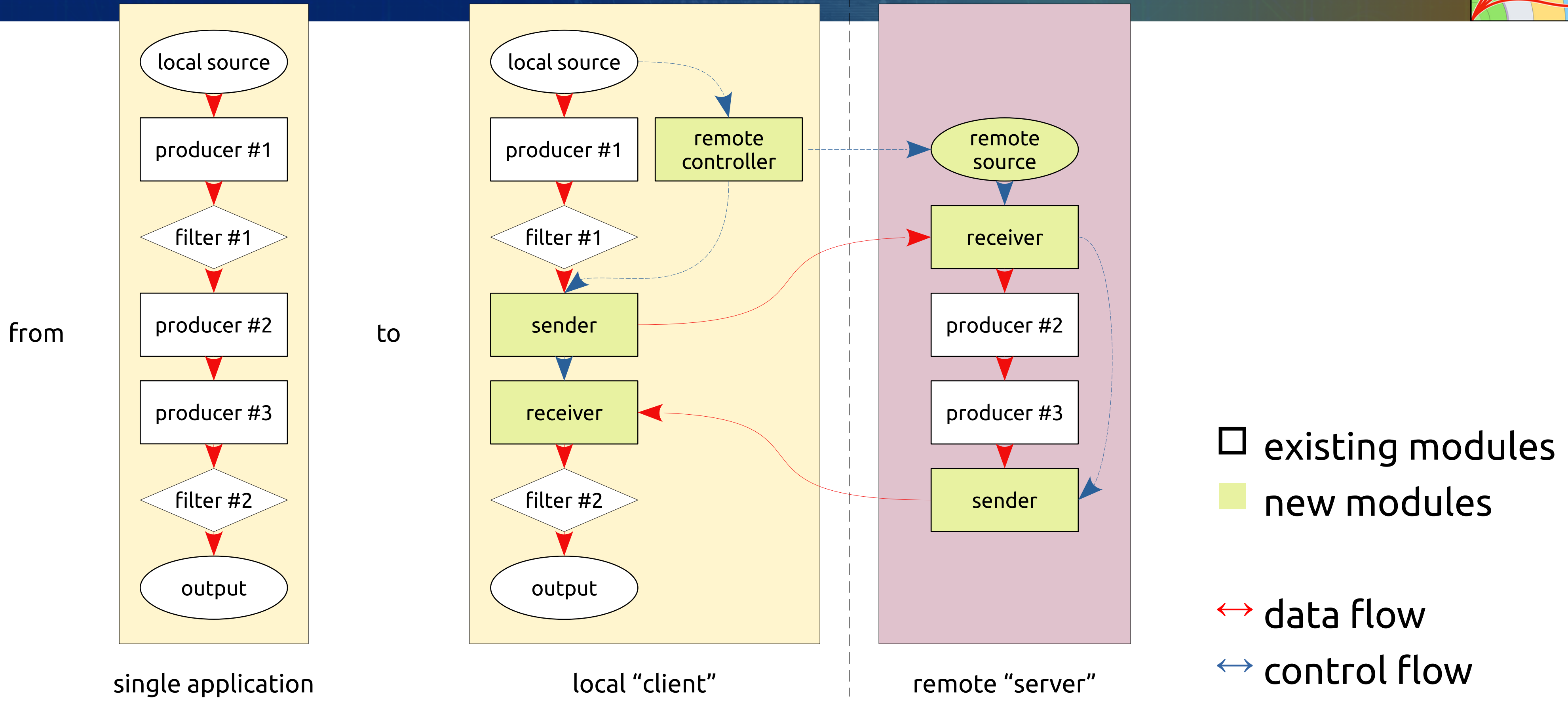


single application

☐ existing modules

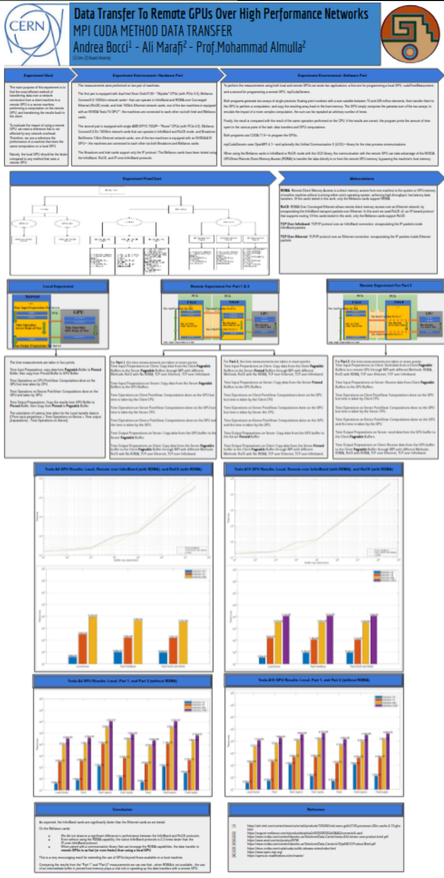
↔ data flow

simplified diagram



technology choices

- modular approach under study and evaluation since late 2020
    - limited personpower before NGT project
    - designed evolved during this year along with the first concrete implementation
        - thanks to Prof. Fawaz Alazemi (Kuwait University) and Andrea Valenzuela (CERN)
- choice to use MPI for the prototype
    - widely used in HPC (not so much in HEP)
    - efficient data movement
        - within a single node (shared memory) and over high speed interconnects (InfiniBand, RoCE, *etc.*)
    - support RDMA to and from GPU memory
        - demonstrated close-to-local data transfer peroformance (Ali Marafi, ACAT 2022)
    - work on fault tolerance integrated in the MPI 4.0 standard
        - "User Level Fault Mitigation" available in OpenMPI 5.0
- validate these choices in the coming year(s)
    - measure the performance of the modules, library, and network solutions
    - evaluate different paradigms like RPC
        - Anna Polova will join the project as a technical student in February



time line and milestones

Q4 2024

Implementation of a client-server, multithreaded, distributed test application, based on CMSSW, leveraging high-speed host-to-host or shared memory communication.

Q2 2025

Implementation of a small-scale demonstrator of a full HLT-like application.

Q2 2026

Support for optimal use of remote accelerators, *e.g.* using RDMA to/from GPU memory

Q4 2026

Support for multiple servers and distributed configurations.

Q4 2027

Compare different approaches to improve the resiliency of the system, such as server redundancy and client-side failure mitigation strategies.

Q2 2028

Evaluate the performance of different network interconnects and communication protocols.

Q4 2028

Large scale deployment and testing of the whole infrastructure in view of the HL-LHC data-taking in Run 4

time line and milestones

- ✔ Q4 2024
  - Implementation of a client-server, multithreaded, distributed test application, based on CMSSW, leveraging high-speed host-to-host or shared memory communication.
- Q2 2025
  - Implementation of a small-scale demonstrator of a full HLT-like application.
- Q2 2026
  - Support for optimal use of remote accelerators, *e.g.* using RDMA to/from GPU memory
- Q4 2026
  - Support for multiple servers and distributed configurations.
- Q4 2027
  - Compare different approaches to improve the resiliency of the system, such as server redundancy and client-side failure mitigation strategies.
- Q2 2028
  - Evaluate the performance of different network interconnects and communication protocols.
- Q4 2028
  - Large scale deployment and testing of the whole infrastructure in view of the HL-LHC data-taking in Run 4

2024 results

2024 results

☑ step 1

- controller/source
- no sender/receiver
- single thread, single stream
- single client, single server

☑ step 2

- send/receive fixed types
- single sender/receiver

☑ step 3

- multiple threads, multiple streams

☑ step 4

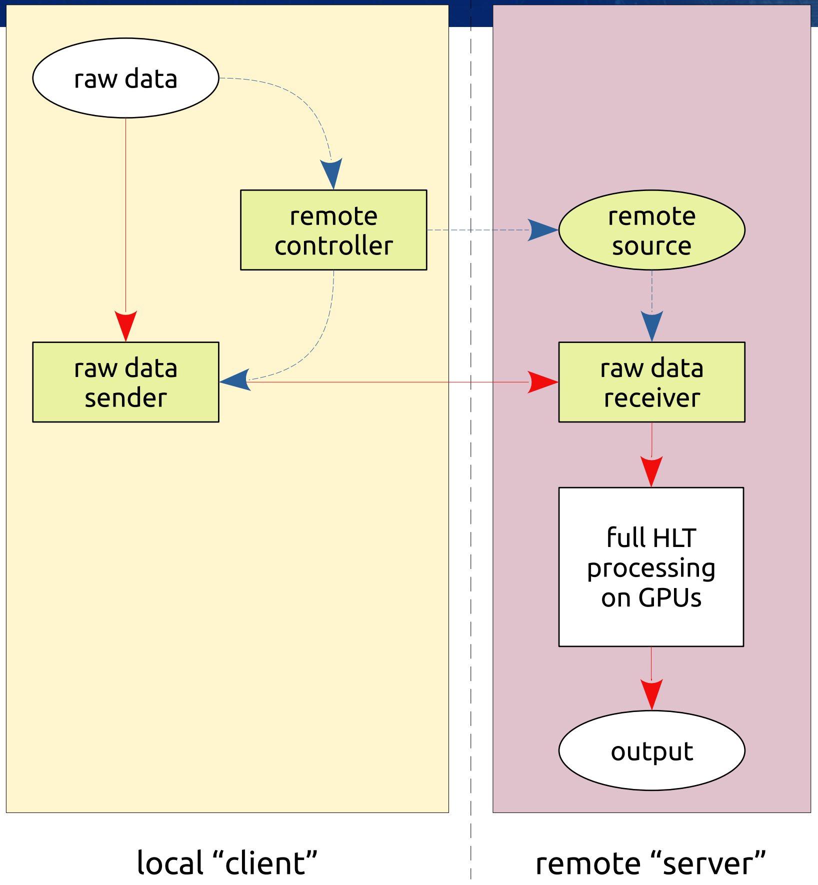
- multiple senders, multiple receivers

☑ 2024 demonstrator

- integrate steps 2, 3, 4
- controller/source
- send/receive fixed types
- multiple senders, multiple receivers
- multiple threads, multiple streams
- no support for `edm::Ref` and similar
- single client, single server

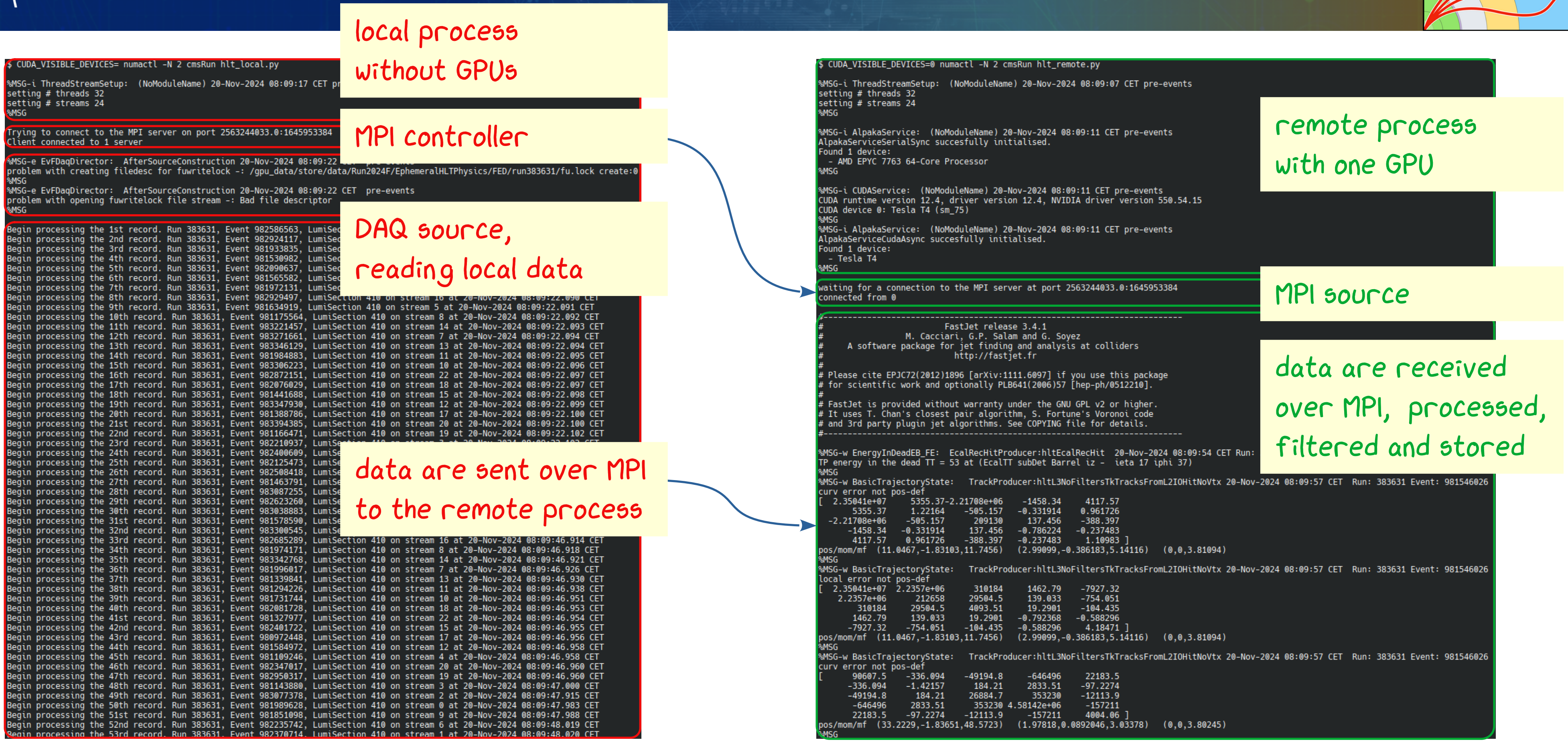
implemented in cms-sw/cmssw#32632

fully remote processing

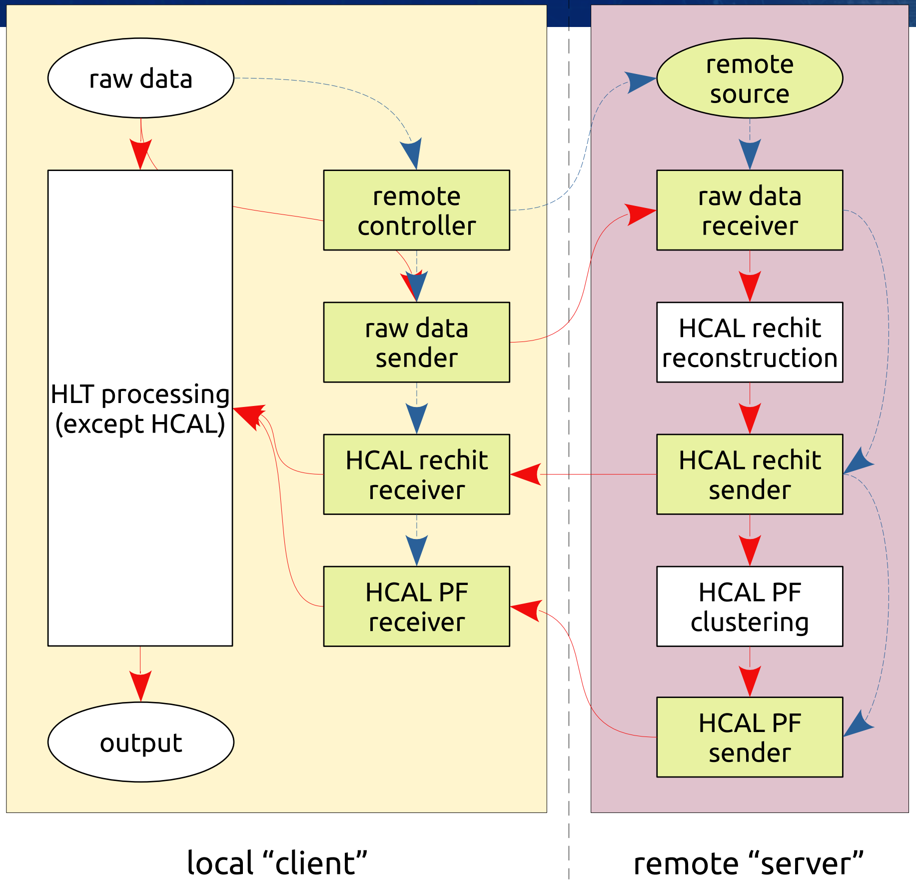


- single-sided communication
  - send RAW data from one process to another
  - full HLT reconstruction in the remote process
- demonstrate controller / follower pattern
  - estabilish communication
  - synchronise run, luminosity block and event transitions
- demonstrate data distribution
  - single collection type: RAW data
  - single sender
  - single receiver

HLT running over a simple MPI topology

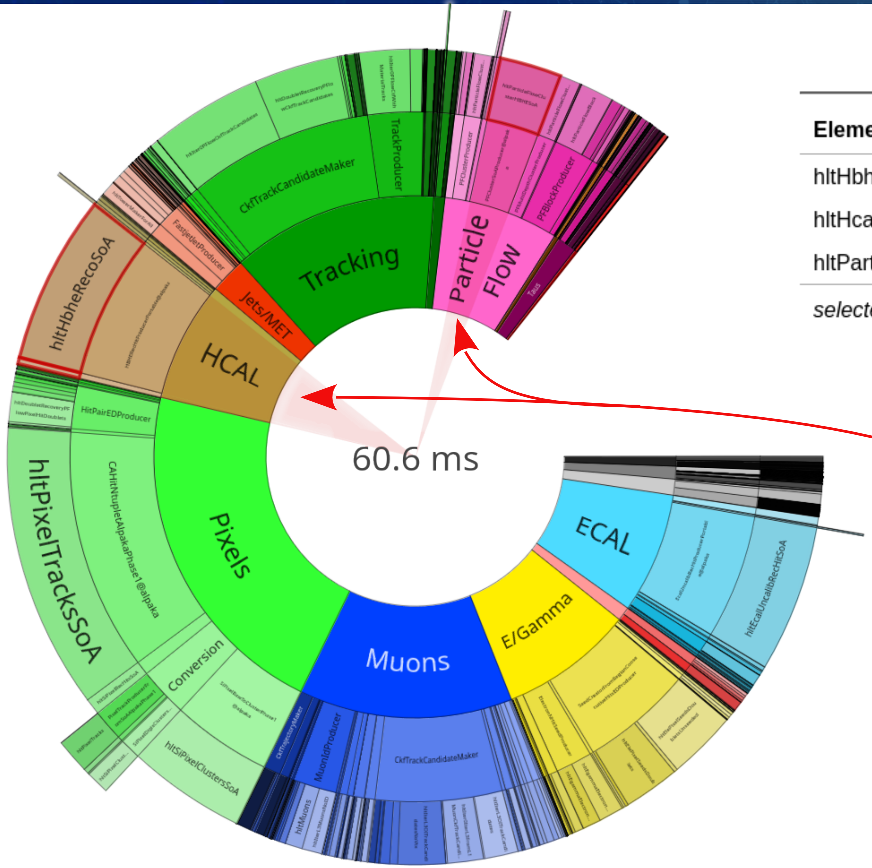


prototype of distributed processing



- multi-sided communication
  - send RAW data from one process to another
  - send back reconstructed objects
- demonstrate cooperative processing
  - run locally using only the CPU
  - offload part of processing to remote GPU node
- demonstrate data distribution
  - multiple collection types
    - RAW data
    - HCAL rechit SoA
    - HCAL PF cluster SoA
  - multiple senders, multiple receivers

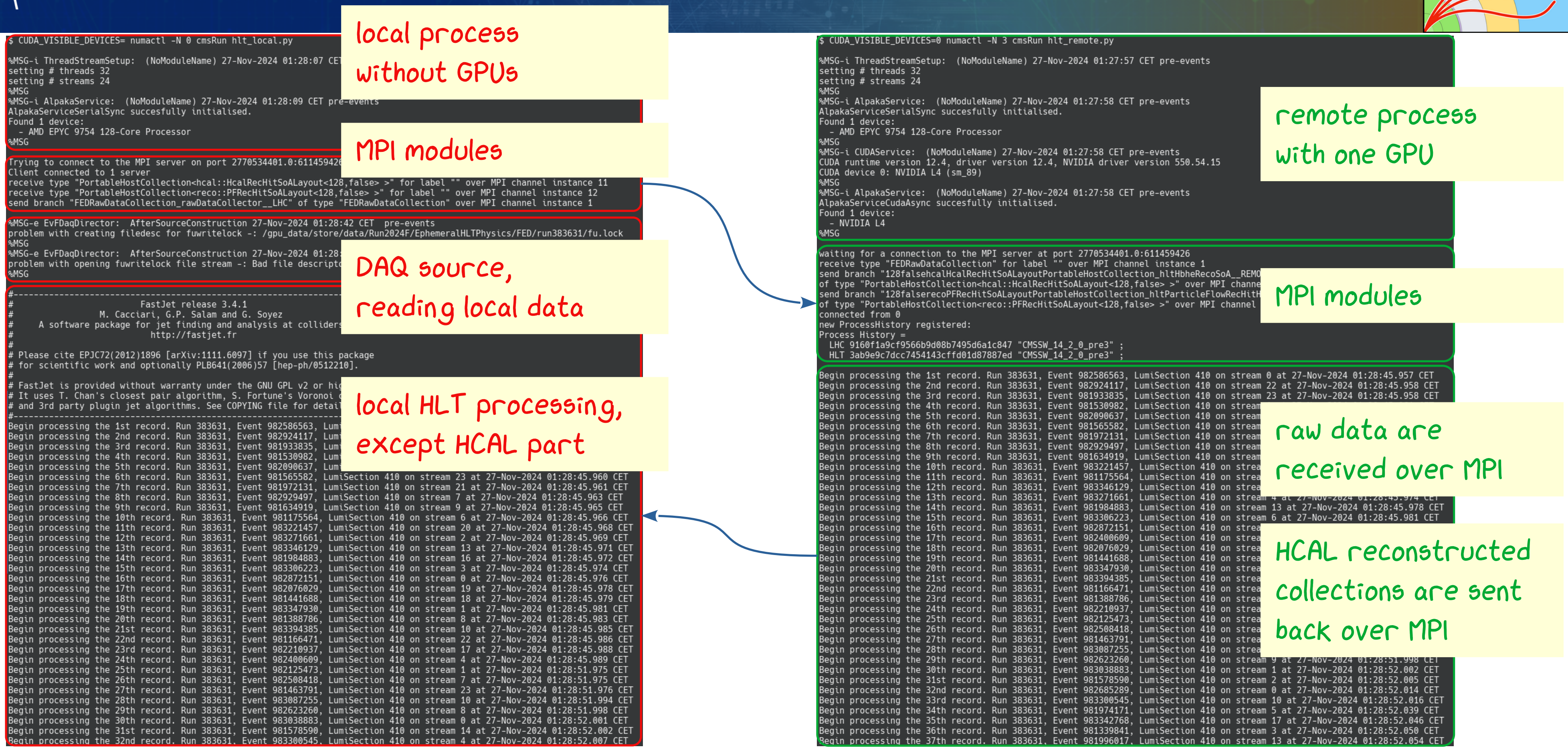
offload to a remote GPU



Element	Time	Fraction
hltHbheRecoSoA	43.2 ms	6.7 %
hltHcalDigis	1.6 ms	0.2 %
hltParticleFlowClusterHBHESoA	15.8 ms	2.4 %
selected	*60.6 ms*	*9.4 %*

- ~90% of the HLT runs locally on CPU
- ~10% runs remotly on GPU

a distributed HLT application over MPI



the road ahead

the road ahead

- 2025 deliverables
  - simplify the code base and support send/receive of arbitrary collections
  - efficient encoding and decoding of "Structure of Arrays" data types
    - bypass ROOT de/serialisation for types with a known layout
    - in collaboration with Task 3.1.2 and Task 1.7

already in progress !

- 2026 deliverables
  - send data directly from (local) GPU memory to (remote) GPU memory
  - non-linear topology with multiple clients and servers
- 2027 deliverables and contractual milestone
  - study various fault tolerance approaches
  - implementation of a client-server, multithreaded, distributed test application, based on the CMS software framework CMSSW, leveraging high-speed host-to-host or shared memory communication
- 2028 deliverables
  - study different hardware interconnects and software libraries
  - plan for large-scale deployment in Run 4

long-term goal, start next year

NexTGen

Next Generation Triggers