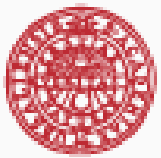


UPPSALA
UNIVERSITET

Prototype of an online track and event reconstruction scheme for the $\overline{\text{P}}$ANDA experiment at FAIR

Project in Computational Science (10 weeks)

Björn Andersson Johan Nordström
Supervisors: Michael Papenbrock, Karin Schönning

March 6, 2017

Uppsala University

Table of contents

1. Project introduction
2. Event reconstruction algorithms
3. Implementation details
4. Performance analysis
5. Summary and outlook

Project introduction

Project introduction

- $\overline{\text{P}}$ANDA experiment: event rates up to 20MHz, event size 10kB

Project introduction

- $\overline{\text{P}}$ANDA experiment: event rates up to 20MHz, event size 10kB
- Data rate: 200GB/s

Project introduction

- $\overline{\text{P}}$ANDA experiment: event rates up to 20MHz, event size 10kB
- Data rate: 200GB/s
- Reduce data rate with a completely software based trigger

Background

- **Project goal**: Prototype of an online track and event reconstruction scheme for the $\overline{\text{P}}$ANDA experiment.

Background

- **Project goal**: Prototype of an online track and event reconstruction scheme for the $\overline{\text{P}}$ANDA experiment.
- Requirements:

Background

- **Project goal**: Prototype of an online track and event reconstruction scheme for the $\overline{\text{P}}$ANDA experiment.
- Requirements:
  - Fast and efficient

Background

- **Project goal**: Prototype of an online track and event reconstruction scheme for the $\overline{\text{P}}$ANDA experiment.
- Requirements:
  - Fast and efficient
  - Scalable

Methodology

- Generation of semi-realistic data

Methodology

- Generation of semi-realistic data
- Prototype event reconstruction algorithms

Methodology

- Generation of semi-realistic data
- Prototype event reconstruction algorithms
- Benchmarking of algorithms

Relevant sub-detectors

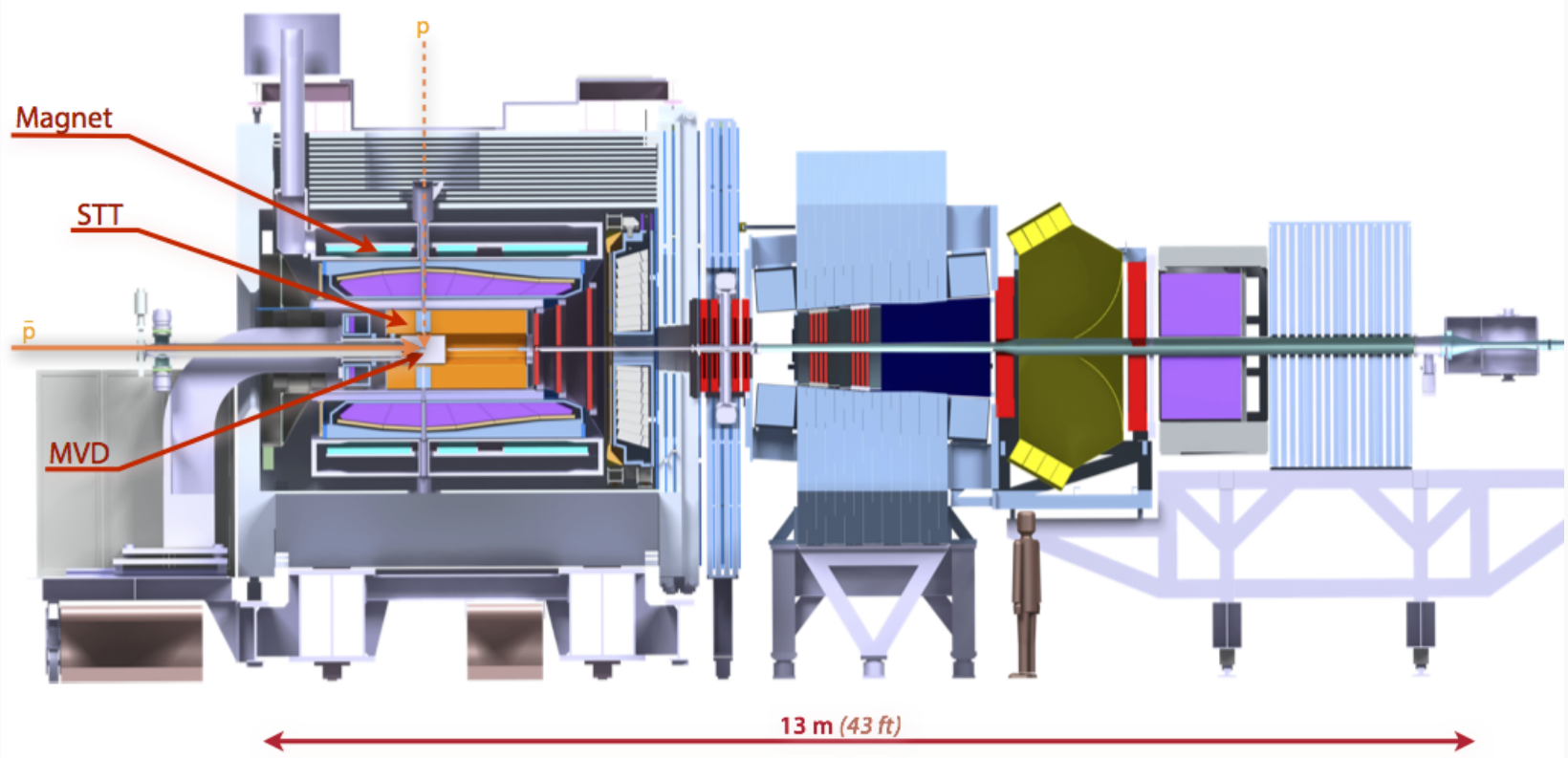


Figure 1: The $\overline{\text{P}}$ANDA Detector

Straw Tube Tracker

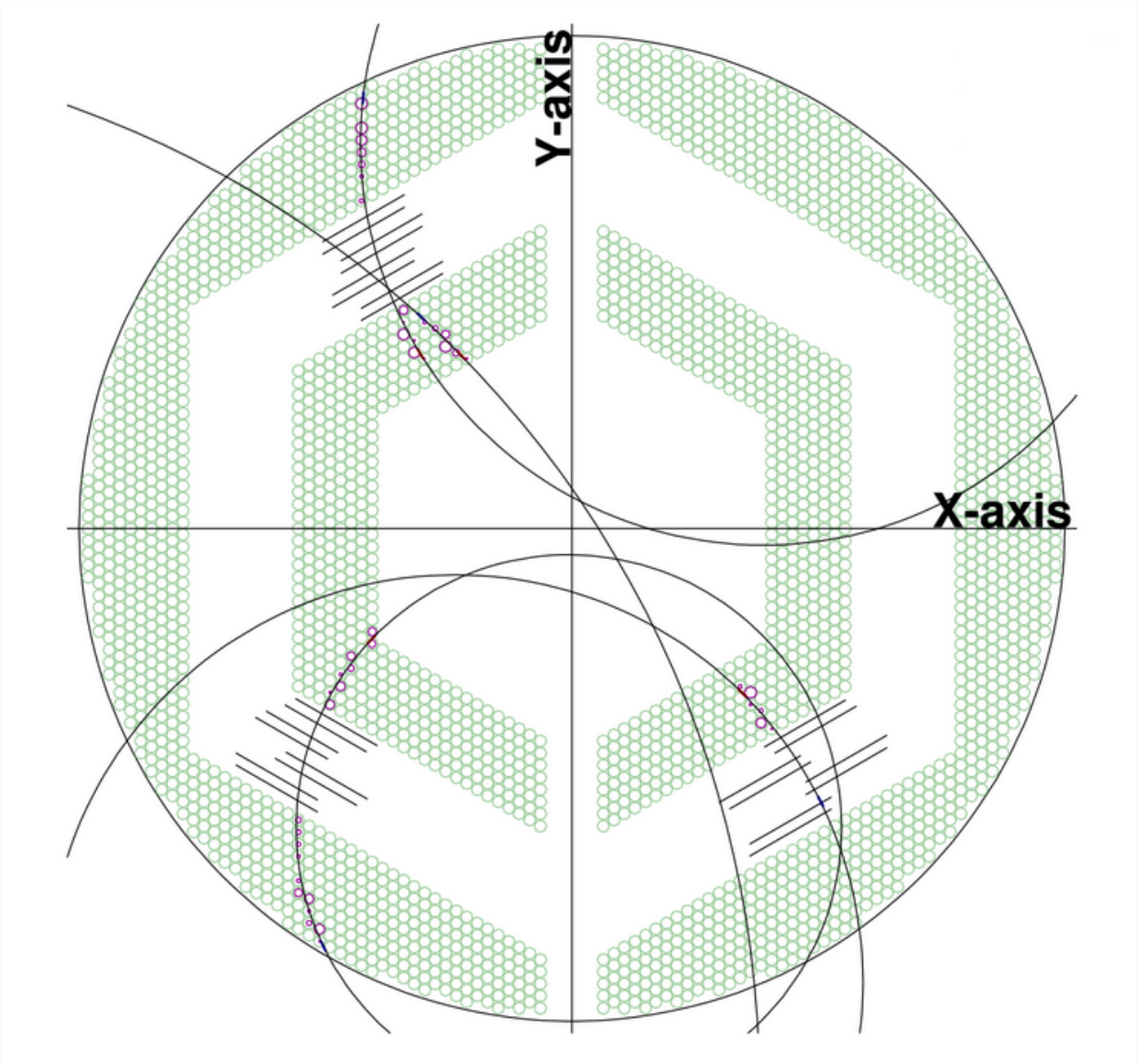


Figure 2: Cross section of STT with trajectories

Generated data structure

- Sequence of STT hits

```
class Hit {
        int strawID;
        float timeStamp;
};
```

Generated data structure

- Sequence of STT hits

```
class Hit {
        int strawID;
        float timeStamp;
};
```

- Hyperons $\rightarrow$ displaced vertices

Generated data structure

- Sequence of STT hits

```
class Hit {
        int strawID;
        float timeStamp;
};
```

- Hyperons $\rightarrow$ displaced vertices
- Hits generated in lines and V-shapes

Generated data structure

- Sequence of STT hits

```
class Hit {
        int strawID;
        float timeStamp;
};
```

- Hyperons $\rightarrow$ displaced vertices
- Hits generated in lines and V-shapes
- Additional noise hits.

Event intermixing

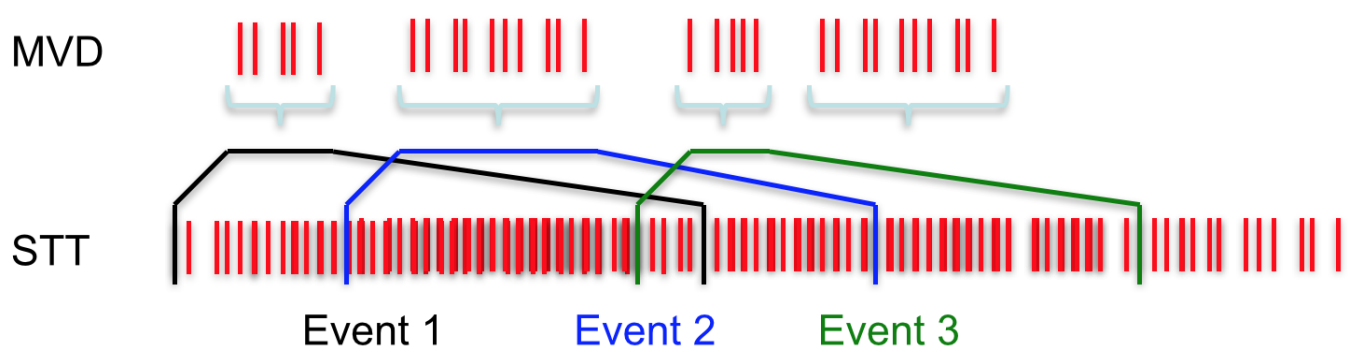


Figure 3: Illustration by T. Stockmanns.

Event intermixing is simulated by incrementing the timestamps with:

$t \rightarrow t + \Delta t + \xi, \quad \xi \sim \mathcal{N}(0, \sigma^2).$

Displaced vertices

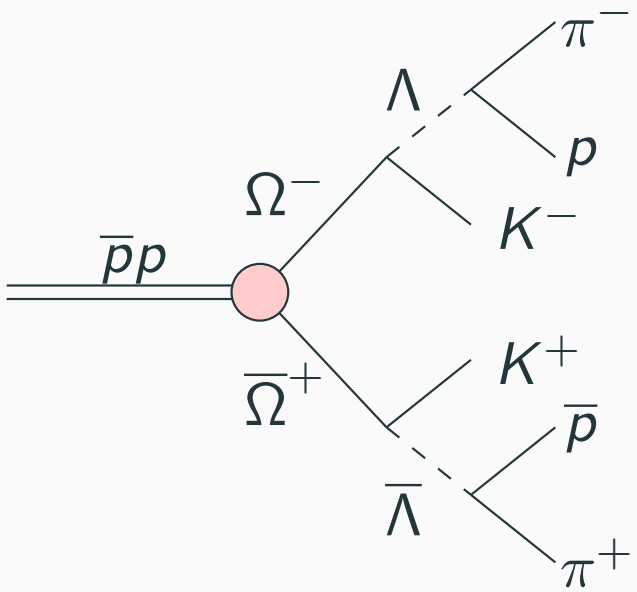


Figure 4: Proton Anti-proton reaction.

Event reconstruction algorithms

Event reconstruction

- Three different algorithms
  - Clustering of STT hits based on position in space and time

Event reconstruction

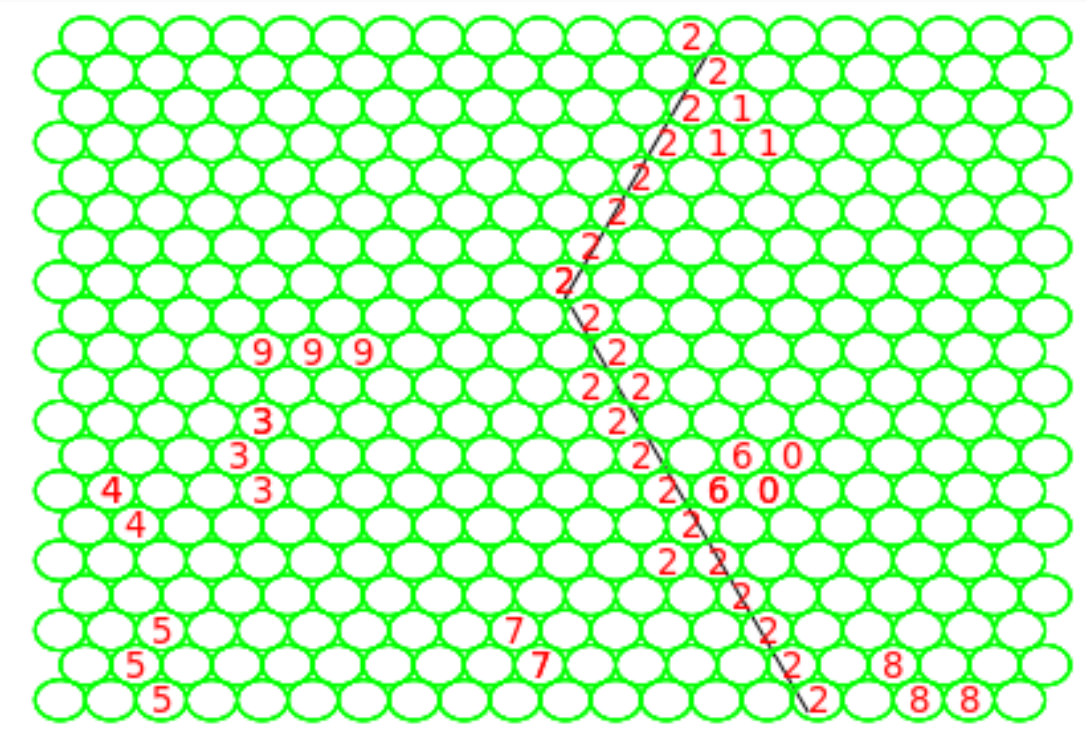
- Three different algorithms
  - Clustering of STT hits based on position in space and time
  - Track reconstruction

Event reconstruction

- Three different algorithms
  - Clustering of STT hits based on position in space and time
  - Track reconstruction
  - Displaced vertex detection (at clustering stage)

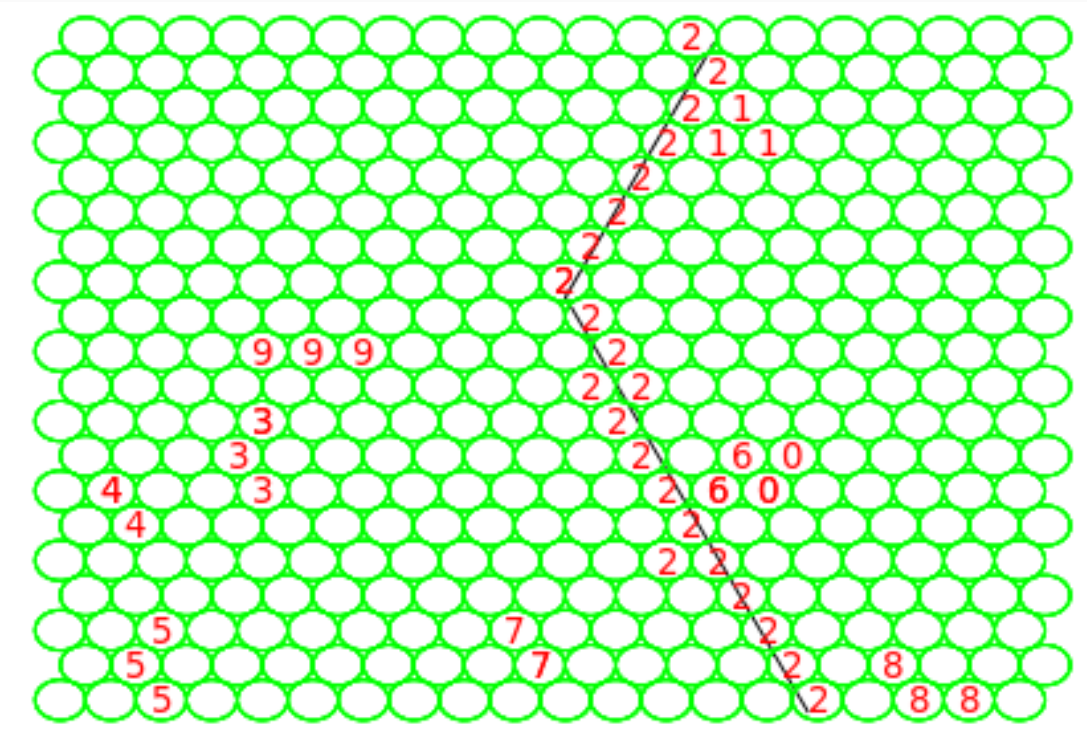
Clustering

- Neighbourhood relation in both space and time



Clustering

- Neighbourhood relation in both space and time
- Clustering N elements $\rightarrow$ time consuming for large N.



Parallel clustering

- Proposed solution: Split hit stream into bins and cluster separately

Parallel clustering

- Proposed solution: Split hit stream into bins and cluster separately
- Assumption (*):

Parallel clustering

- Proposed solution: Split hit stream into bins and cluster separately
- Assumption (*):
  - Time difference between two STT hits in the same event can not exceed STT response time ($\approx$ 250 ns)

Parallel clustering

- Proposed solution: Split hit stream into bins and cluster separately
- Assumption (*):
  - Time difference between two STT hits in the same event can not exceed STT response time ($\approx$ 250 ns)
- Place STT hits in multiple different bins so that (*) holds.

Track reconstruction

- Track reconstruction through modified version of STTCellTrackFinder (Schumann, FZ Jülich)

Track reconstruction

- Track reconstruction through modified version of STTCellTrackFinder (Schumann, FZ Jülich)
- Simplification: Straight particle trajectories (Line fitting)

Cellular automaton

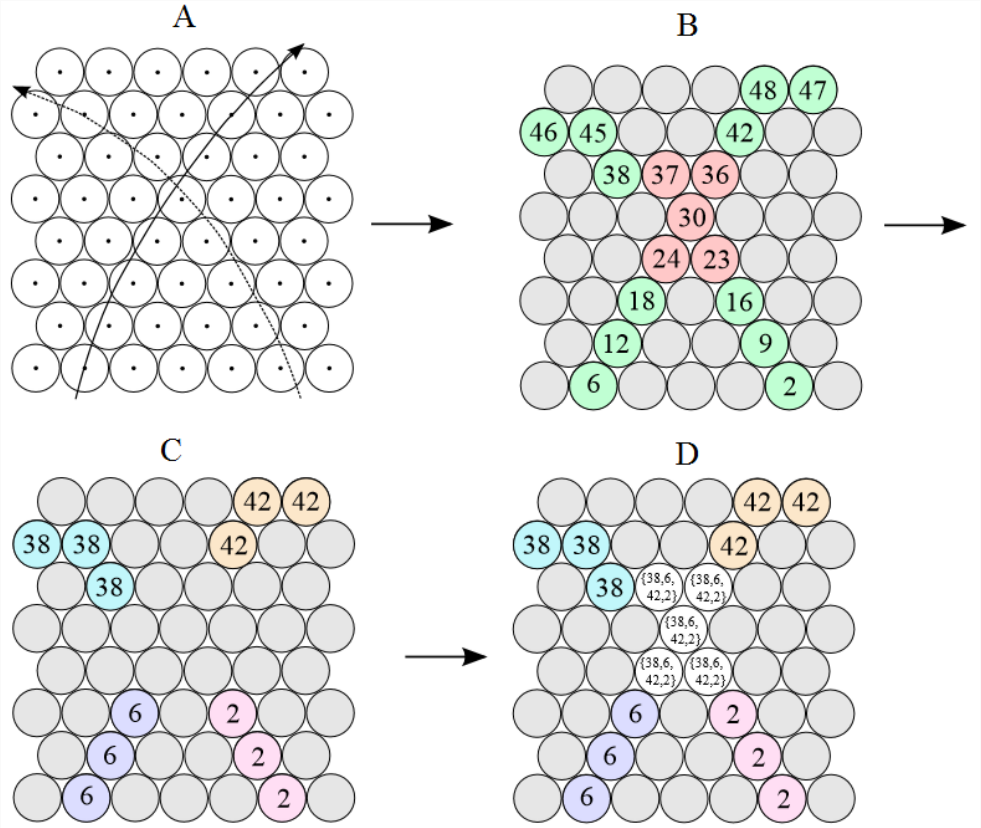
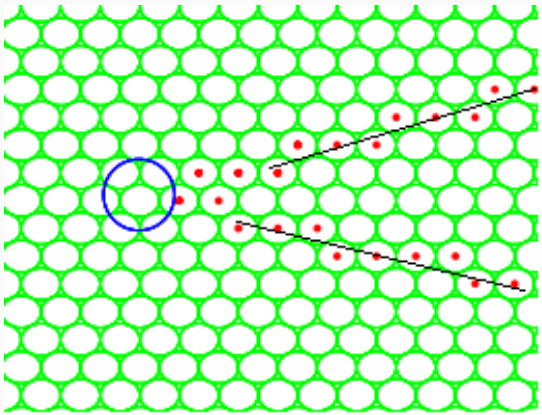


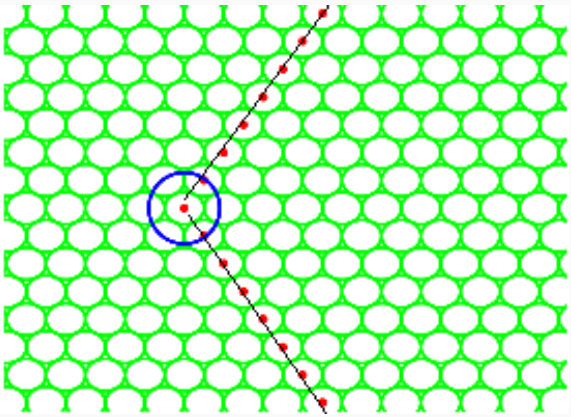
Figure 5: Illustration showing four steps in the track finding cellular automaton.

Displaced vertex detection

Two cases:

- An ambiguous node with exactly two possible IDs

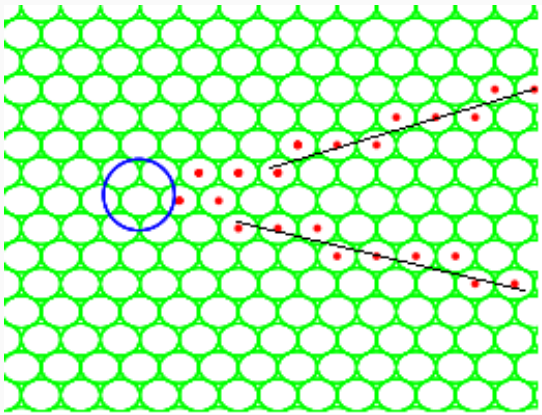


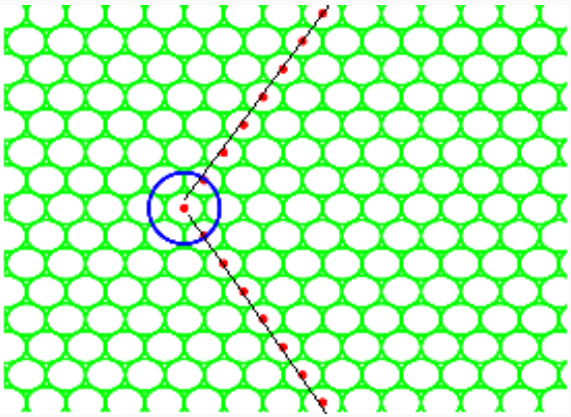


Displaced vertex detection

Two cases:

- An ambiguous node with exactly two possible IDs
- A tracklet with a high mean square error to the curve fit





Implementation details

Target hardware

- Various computing hardware is in consideration

Target hardware

- Various computing hardware is in consideration
- Prototype uses CPU cluster

Target hardware

- Various computing hardware is in consideration
- Prototype uses CPU cluster
  - Suitable for developed algorithms

Target hardware

- Various computing hardware is in consideration
- Prototype uses CPU cluster
  - Suitable for developed algorithms
  - Efficient to prototype in

Target hardware

- Various computing hardware is in consideration
- Prototype uses CPU cluster
  - Suitable for developed algorithms
  - Efficient to prototype in
  - Benchmarking hardware available

Software architecture

- Independent C++ project

Software architecture

- Independent C++ project
- Hybrid parallelization

Software architecture

- Independent C++ project
- Hybrid parallelization
  - Message Passing Interface (MPI) (non-shared memory)

Software architecture

- Independent C++ project
- Hybrid parallelization
  - Message Passing Interface (MPI) (non-shared memory)
  - OpenMP (shared memory)

Software architecture

- Independent C++ project
- Hybrid parallelization
  - Message Passing Interface (MPI) (non-shared memory)
  - OpenMP (shared memory)
- Applicable in multi-core CPU environments

Software architecture

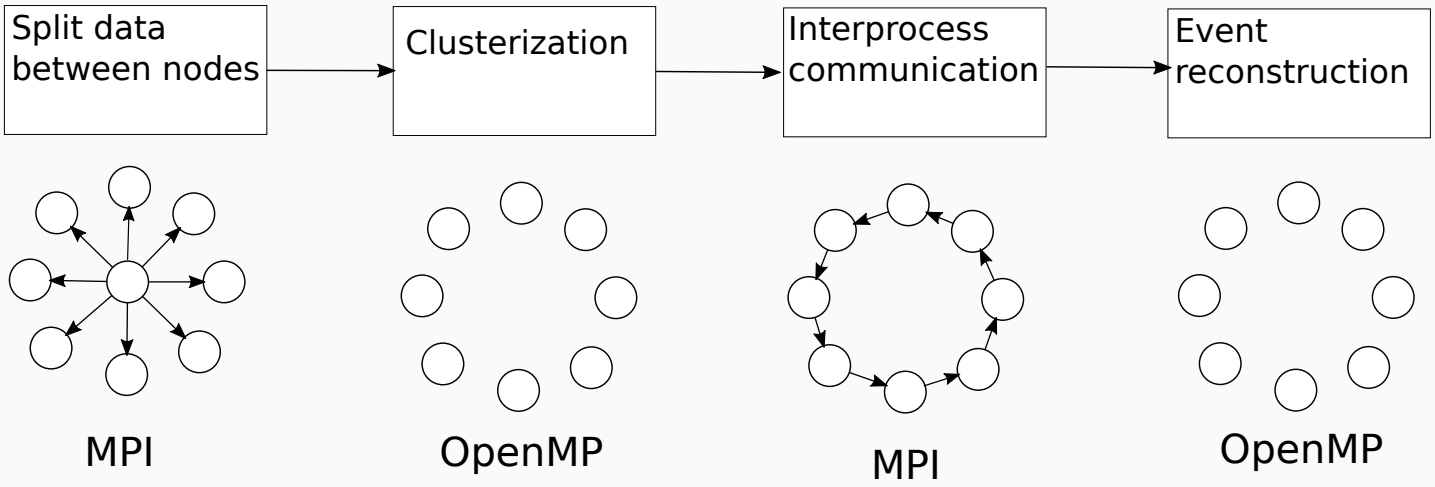
- Independent C++ project
- Hybrid parallelization
  - Message Passing Interface (MPI) (non-shared memory)
  - OpenMP (shared memory)
- Applicable in multi-core CPU environments
- Modular

Parallel structure

- Hit data sequence is divided into stacks

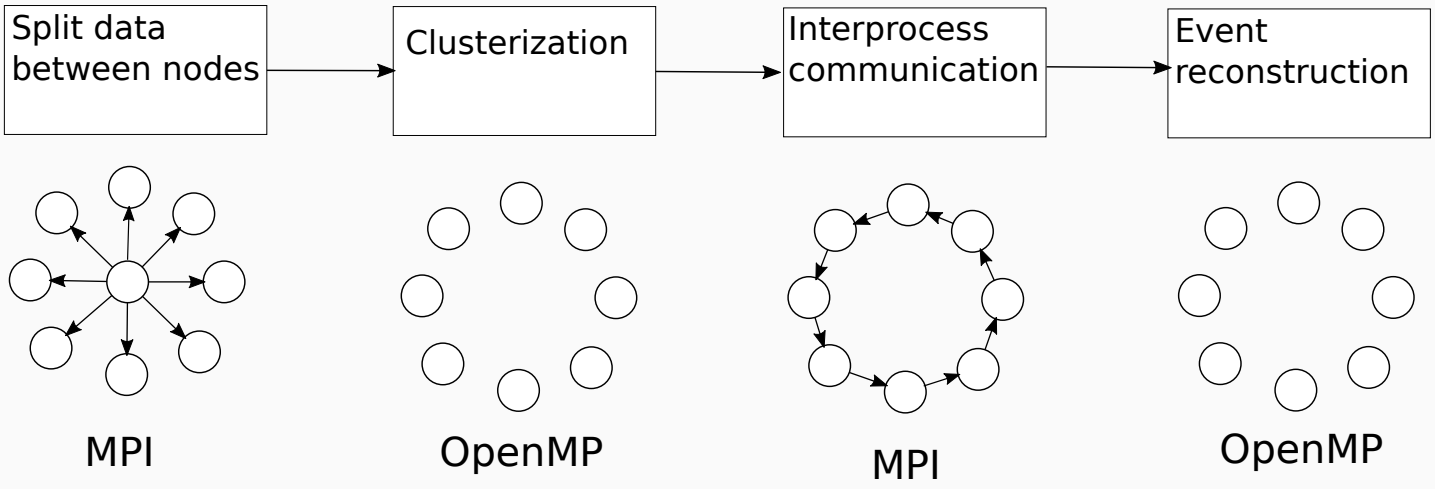
Parallel structure

- Hit data sequence is divided into stacks
- Stacks processed in four main stages



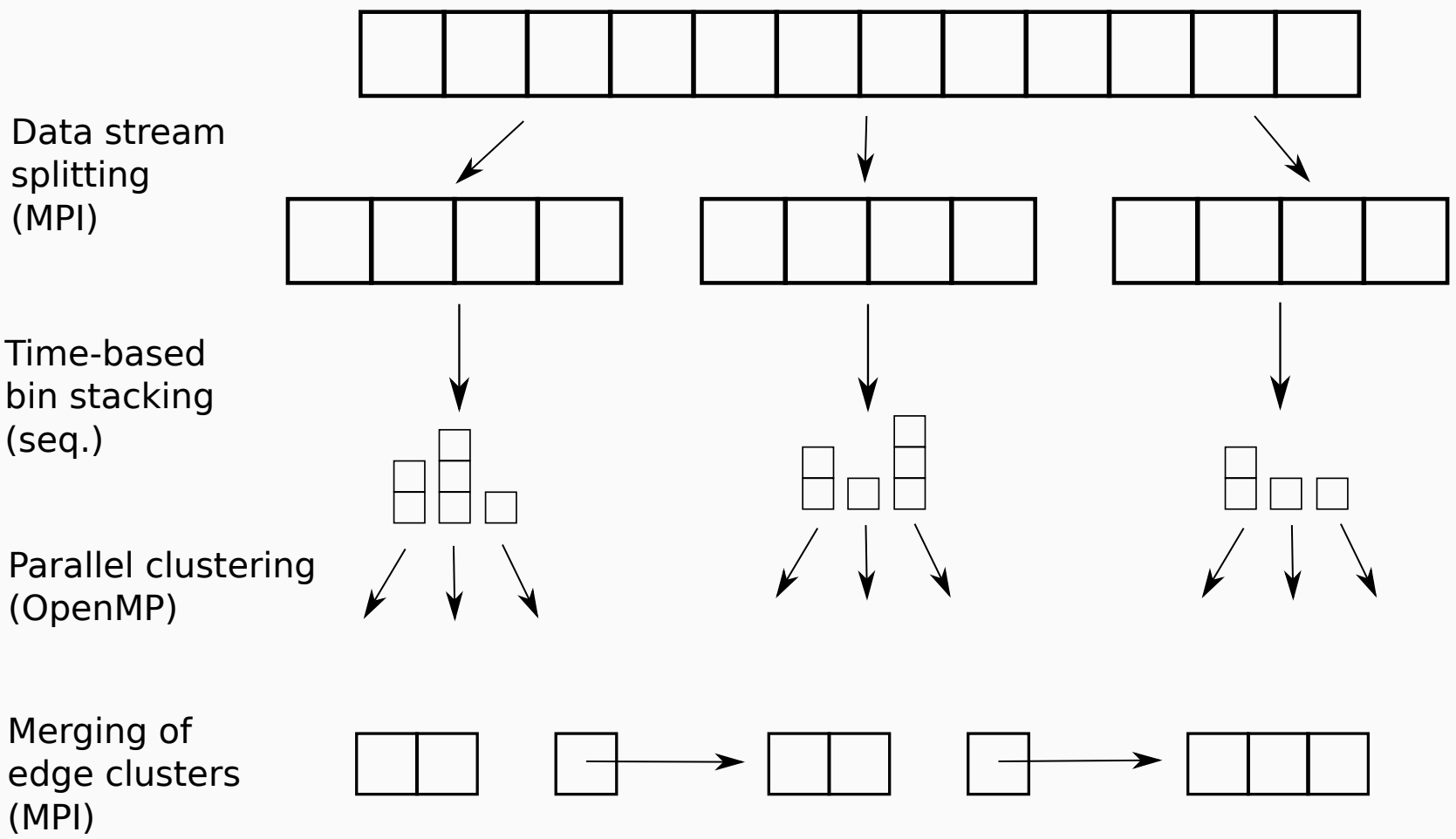
Parallel structure

- Hit data sequence is divided into stacks
- Stacks processed in four main stages



- Load balancing by master/slave model

Parallel clustering pipeline



Performance analysis

Performance metrics

- MPI and OpenMP components are analyzed separately

Performance metrics

- MPI and OpenMP components are analyzed separately
- Performance metrics:

Performance metrics

- MPI and OpenMP components are analyzed separately
- Performance metrics:
  - Efficiency: $E(N) = \frac{\tau_1}{N\tau_N}$

Performance metrics

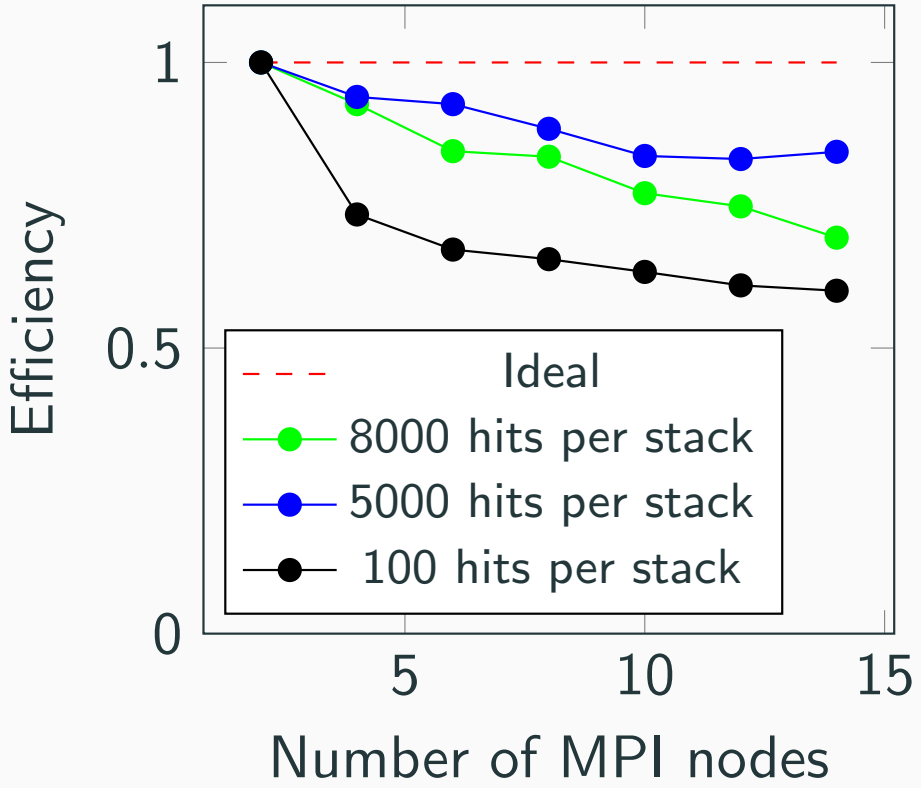
- MPI and OpenMP components are analyzed separately
- Performance metrics:
  - Efficiency: $E(N) = \frac{\tau_1}{N\tau_N}$
  - Speedup: $S(N) = \frac{\tau_1}{\tau_N}$.

Performance metrics

- MPI and OpenMP components are analyzed separately
- Performance metrics:
  - Efficiency: $E(N) = \frac{\tau_1}{N\tau_N}$
  - Speedup: $S(N) = \frac{\tau_1}{\tau_N}$.
  - τ_1 and τ_N are the execution times using 1 and N computing nodes respectively for a fixed problem size.

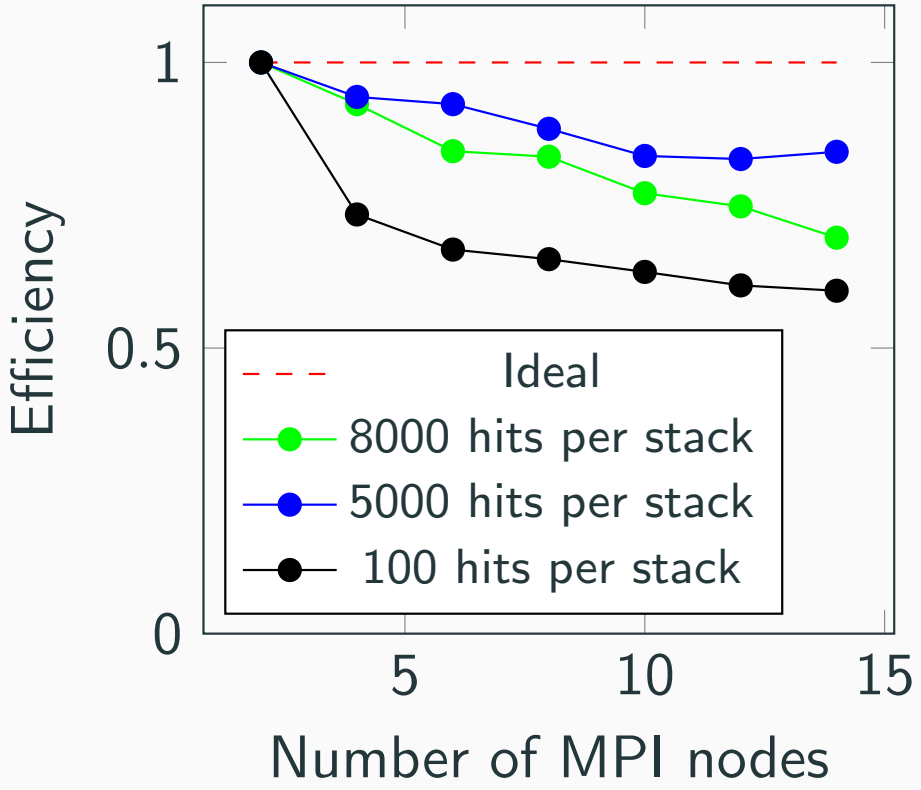
Non-shared memory efficiency (MPI)

- 5000 STT hits per stack
  $\rightarrow$ optimal efficiency

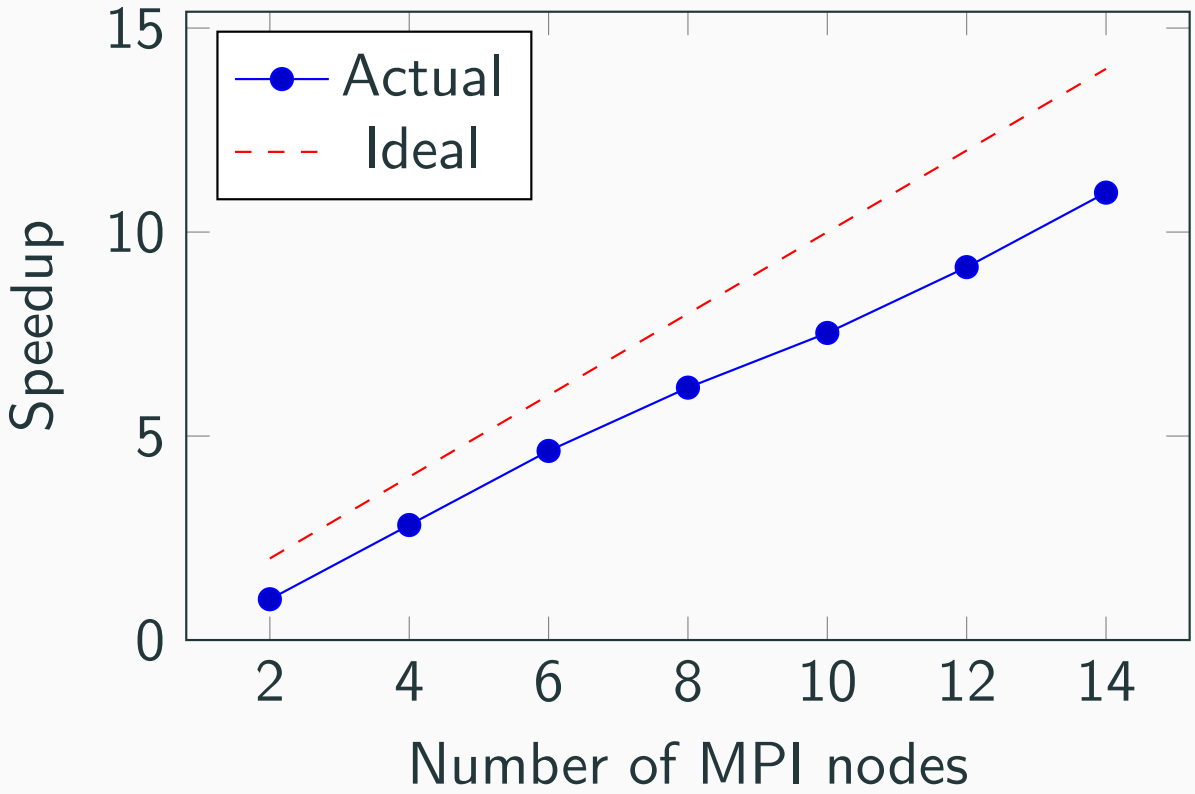


Non-shared memory efficiency (MPI)

- 5000 STT hits per stack
  $\rightarrow$ optimal efficiency
- Good efficiency!

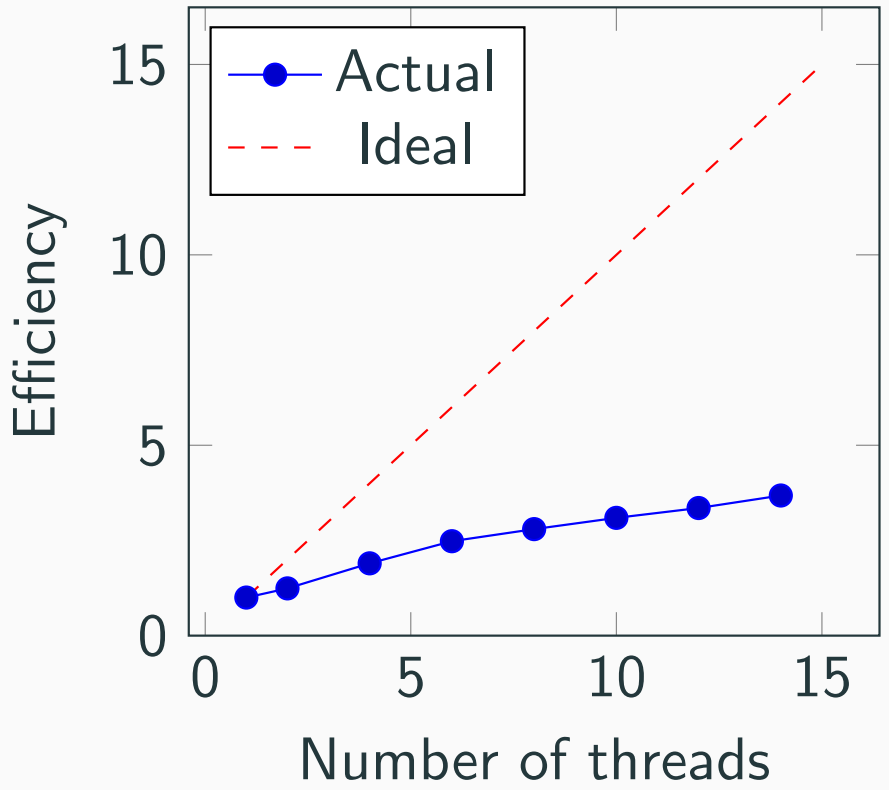


Non-shared memory speedup (MPI)



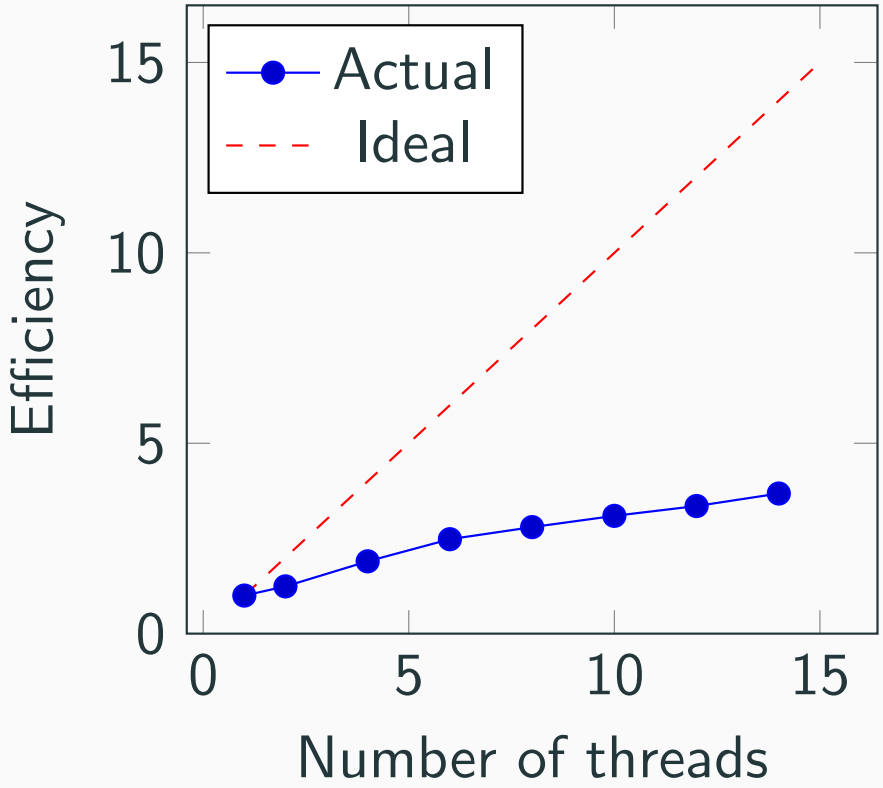
Shared memory speedup (OpenMP)

- Quite low speedup



Shared memory speedup (OpenMP)

- Quite low speedup
- Will improve with more realistic algorithms.



Parallel clustering run time

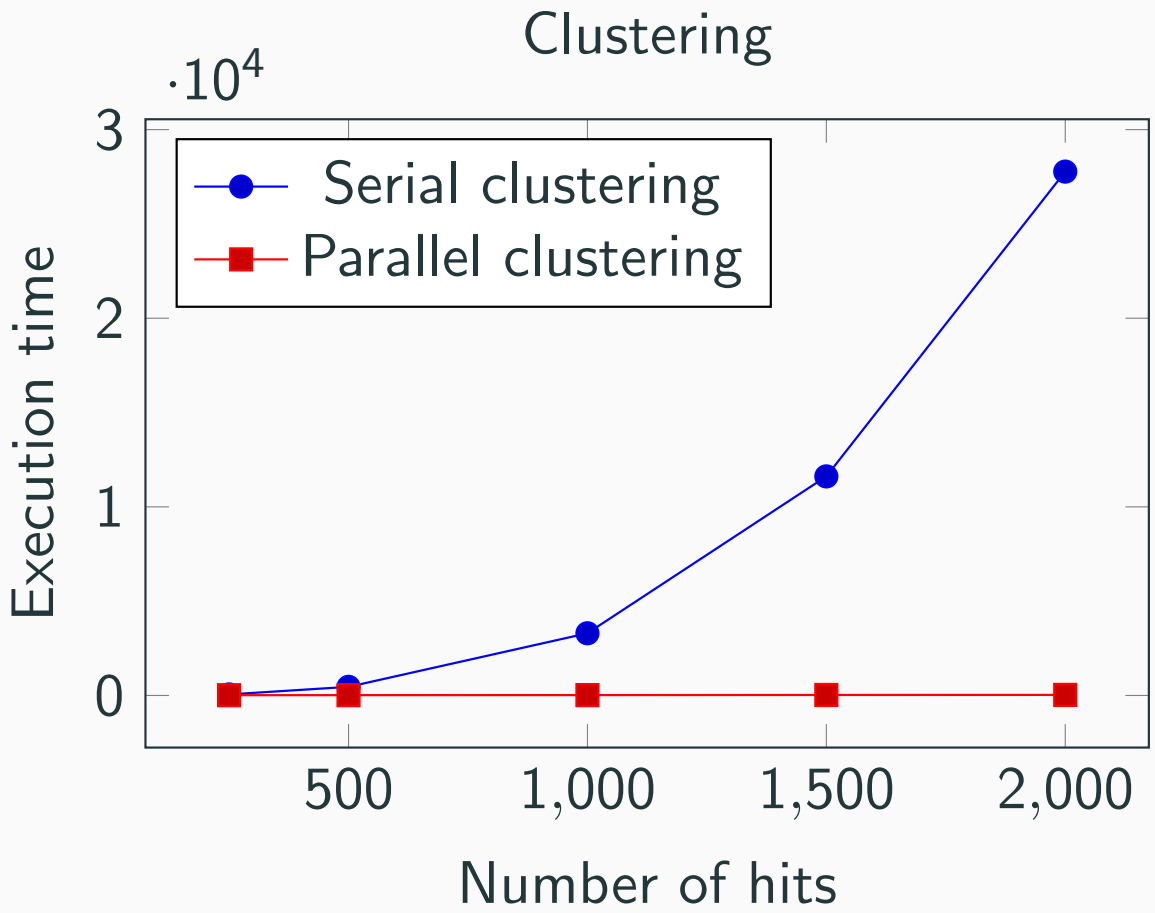


Figure 6: Execution time of the clustering algorithms.

Summary and outlook

Summary

- Prototype of an online track and event reconstruction scheme.

Summary

- Prototype of an online track and event reconstruction scheme.
- Hybrid parallelization model (MPI/OpenMP)

Summary

- Prototype of an online track and event reconstruction scheme.
- Hybrid parallelization model (MPI/OpenMP)
- Possible extension of STTCellTrackFinder

Summary

- Prototype of an online track and event reconstruction scheme.
- Hybrid parallelization model (MPI/OpenMP)
- Possible extension of STTCellTrackFinder
- Promising scaling.

Outlook

- Modify the event reconstruction algorithms to handle realistic data.

Outlook

- Modify the event reconstruction algorithms to handle realistic data.
- Integrate the system with the $\overline{\text{P}}$ANDA simulation framework (PandaRoot).

Outlook

- Modify the event reconstruction algorithms to handle realistic data.
- Integrate the system with the $\overline{\text{P}}$ANDA simulation framework (PandaRoot).
- Investigate the use of a dynamic load balancing scheme.

Outlook

- Modify the event reconstruction algorithms to handle realistic data.
- Integrate the system with the $\overline{\text{P}}$ANDA simulation framework (PandaRoot).
- Investigate the use of a dynamic load balancing scheme.
- More thorough performance analysis on larger scale systems

Outlook

- Modify the event reconstruction algorithms to handle realistic data.
- Integrate the system with the $\overline{\text{P}}$ANDA simulation framework (PandaRoot).
- Investigate the use of a dynamic load balancing scheme.
- More thorough performance analysis on larger scale systems
- Look into frameworks that allow for streaming data processing.

Outlook

- Modify the event reconstruction algorithms to handle realistic data.
- Integrate the system with the $\overline{\text{P}}$ANDA simulation framework (PandaRoot).
- Investigate the use of a dynamic load balancing scheme.
- More thorough performance analysis on larger scale systems
- Look into frameworks that allow for streaming data processing.
- The report is available as an internal $\overline{\text{P}}$ANDA document.

Questions

Questions?

Inter-process communication

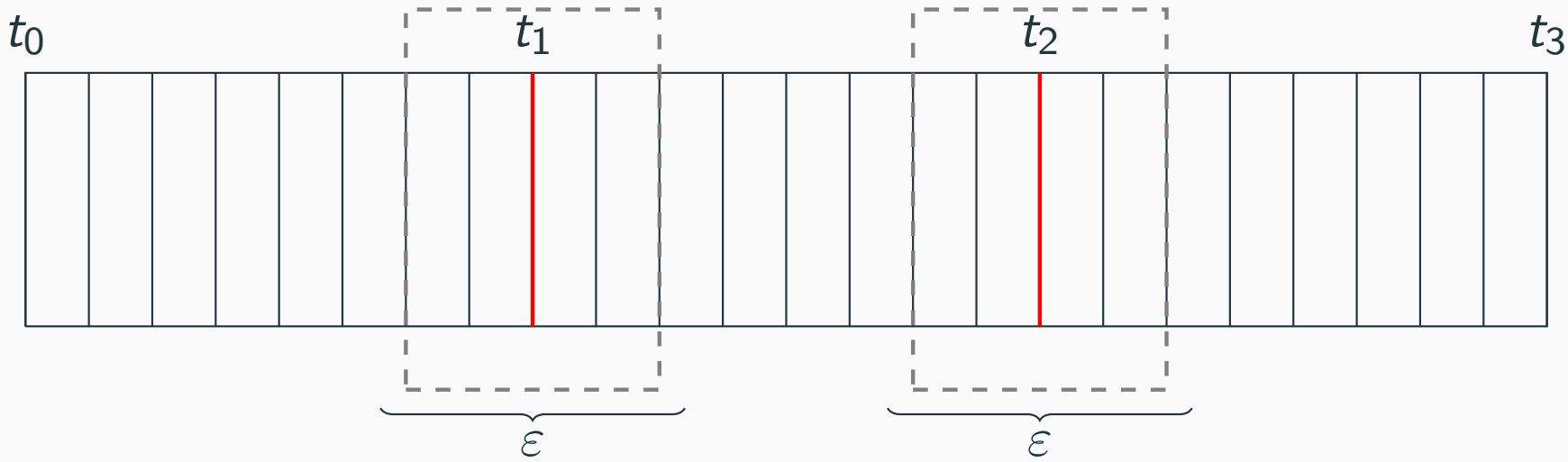


Figure 7: Ordered hit sequence by time.

- Stacks in different nodes $\rightarrow$ need interprocess-communication.
- Can be ignored if one allows to throw away a proportion of events.