GRAPH-BASED DECISION MAKING FOR TASK SCHEDULING IN CONCURRENT GAUDI

I. SHAPOVAL[1,2,3,4], M. CLEMENCIC[1], B. HEGNER[1], D. FUNKE[1,5], D. PIPARO[1], P. MATO[1]

CERN[1], SWITZERLAND

KIPT[2], UKRAINE

UNIFE[3], **INFN-FE**[4], ITALY

KIT[5], GERMANY

IEEE NSS/MIC 2015, San Diego, US

CONTENT

- Introduction
- Concurrency control: reactive scheduling

- Speedup and scalability with reactive scheduling

- Concurrency control: predictive scheduling

- Generic analysis of speedup constraints

GAUDI FRAMEWORK

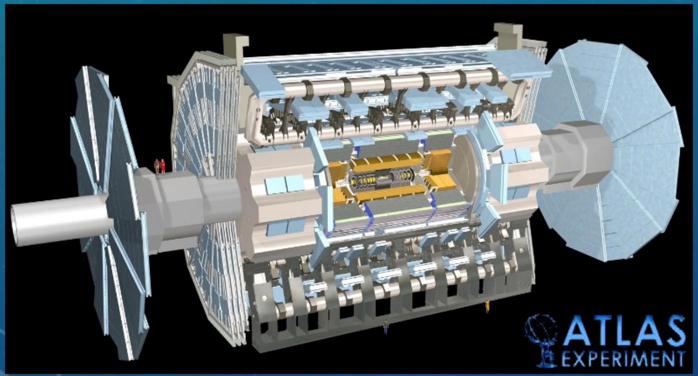
An object-oriented software architecture for event data processing applications in high energy physics domain.

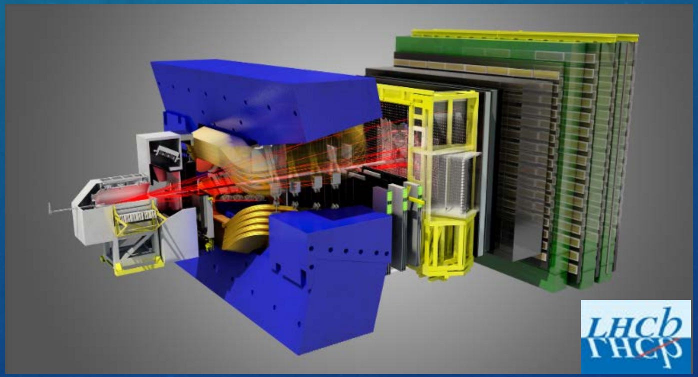
- ❖ Designed on principles of:
  - ✓ Separation of algorithms and data
  - ✓ Composability and reusability (via abstract interfaces)
- ❖ Written in C++ and Python
- ❖ ~150k SLOC

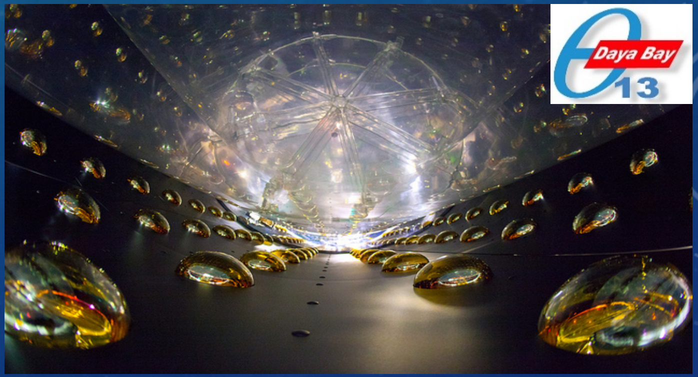
GAUDI FRAMEWORK

An object-oriented software architecture for event data processing applications in high energy physics domain.

- Designed on principles of:
  - ✓ Separation of algorithms and data
  - ✓ Composability and reusability (via abstract interfaces)
- Written in C++ and Python
- ~150k SLOC







GAUDI FRAMEWORK

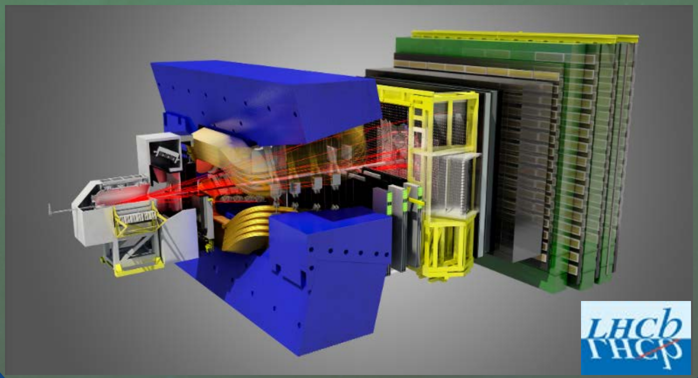
An object-oriented software architecture for event data processing applications in high energy physics domain.

- ❖ Designed on principles of:
  - ✓ Separation of algorithms and data
  - ✓ Composability and reusability (via abstract interfaces)
- ❖ Written in C++ and Python
- ❖ ~150k SLOC

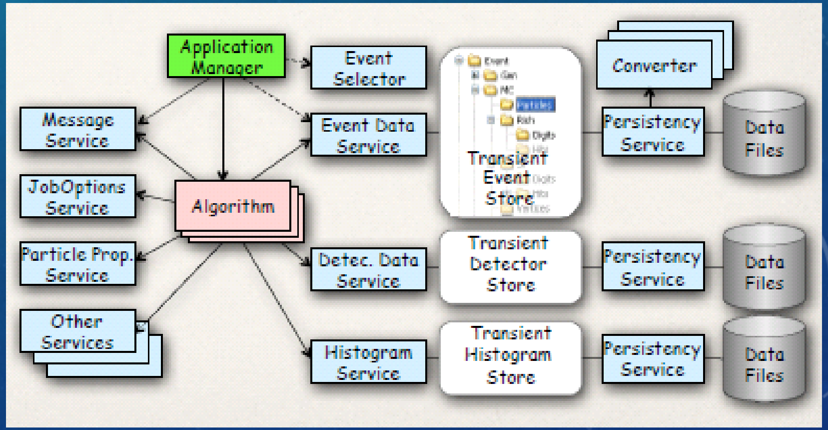
Gaudi-based

- HL Triggering
- Reconstruction
- Analyses
- Detector simulation
- Event display

>3.6M SLOC (C++ & Python)



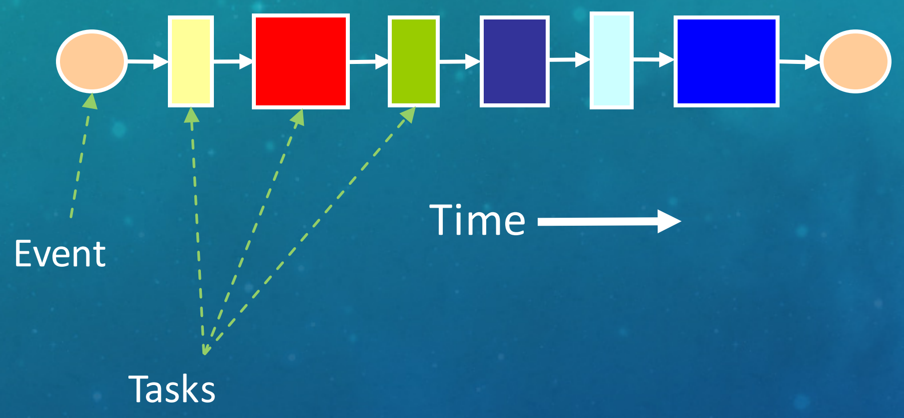
ATLAS and LHCb customization of Gaudi



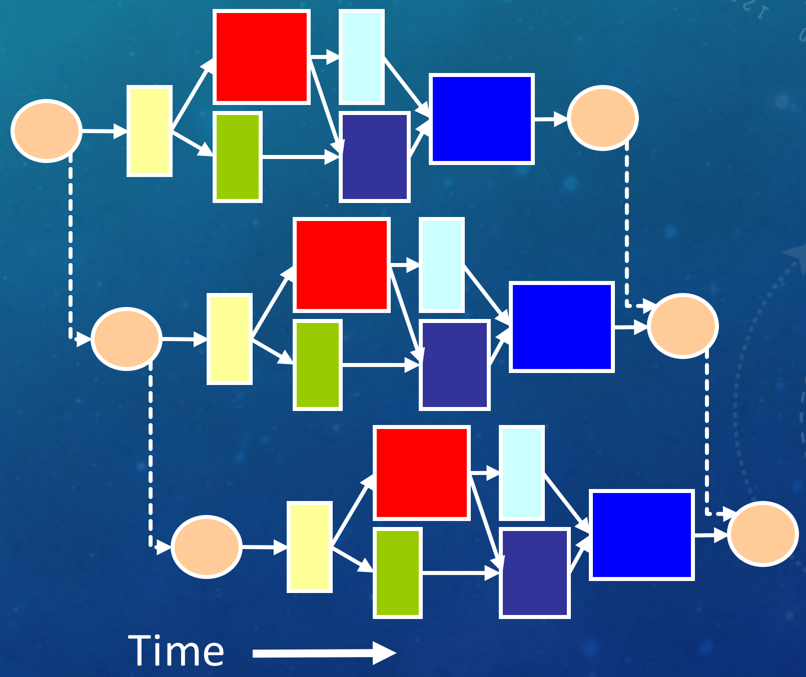
CONCURRENT GAUDI (A.K.A. GAUDI HIVE)

A prototype of a multithreaded task-based incarnation of Gaudi.

Sequential Gaudi job



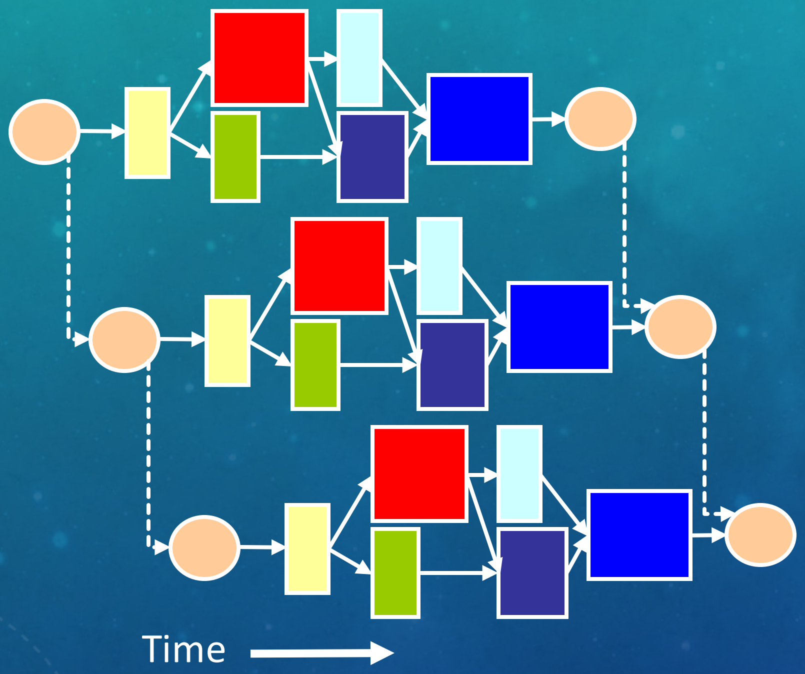
Concurrent Gaudi (Hive) job

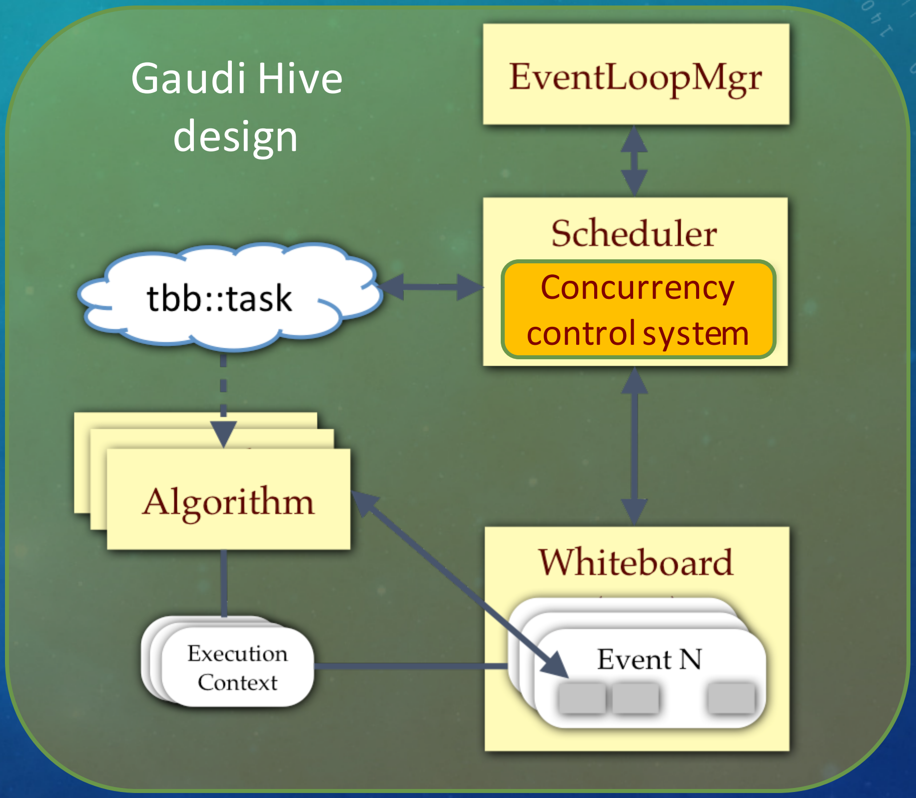


CONCURRENT GAUDI (A.K.A. GAUDI HIVE)

A prototype of a multithreaded task-based incarnation of Gaudi.

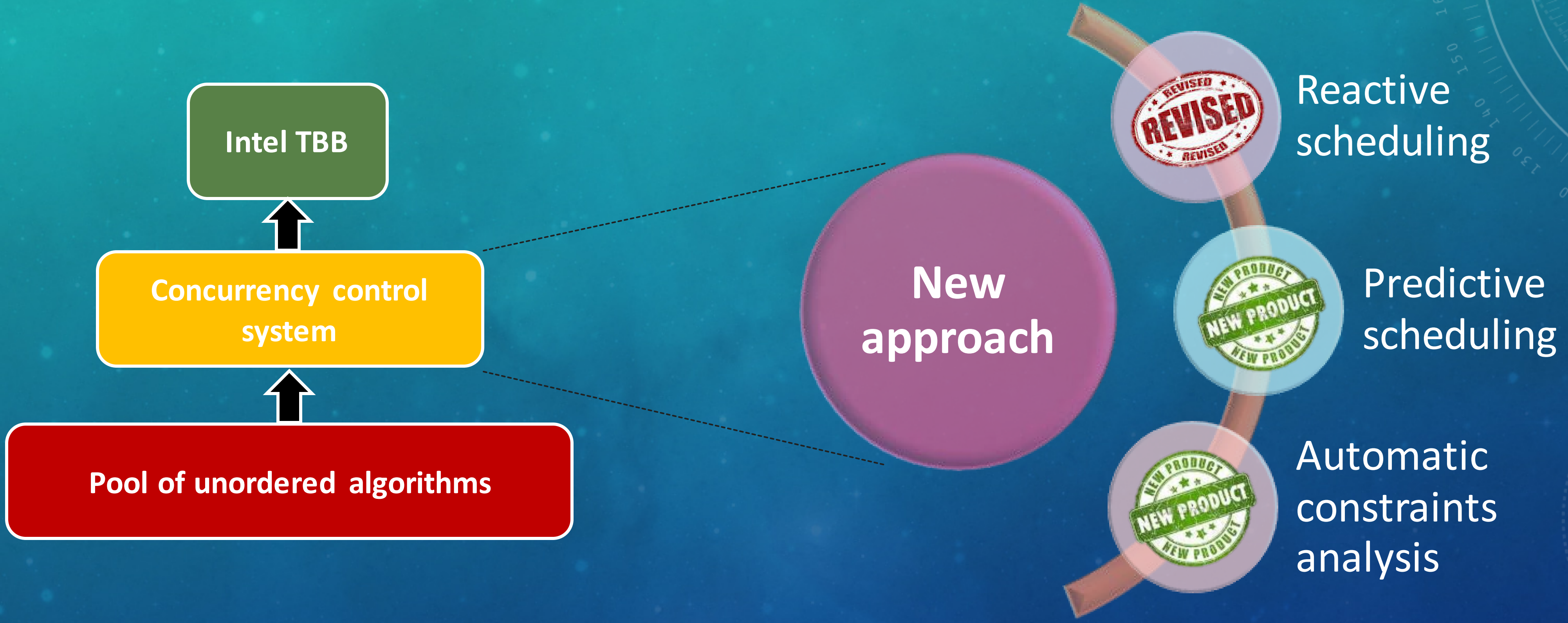
Concurrent Gaudi (Hive) job





GAUDI HIVE CONCURRENCY CONTROL

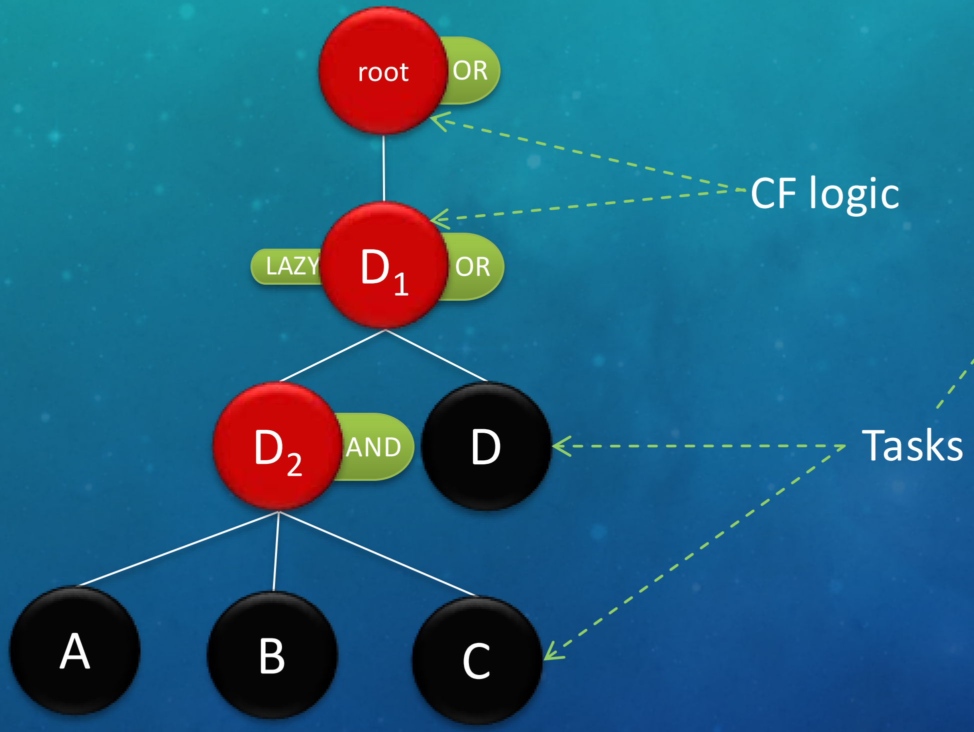
GAUDI HIVE CONCURRENCY CONTROL

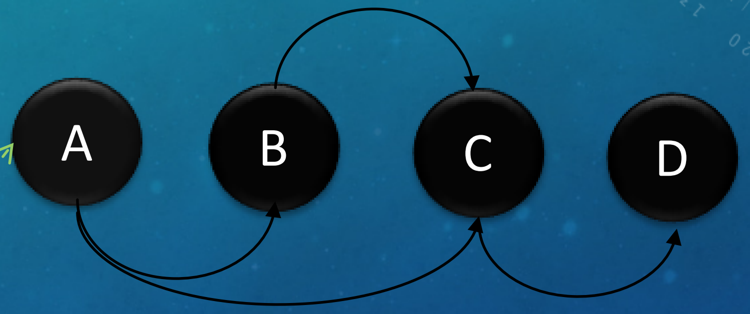


INTRA-EVENT TASK PRECEDENCE RULES

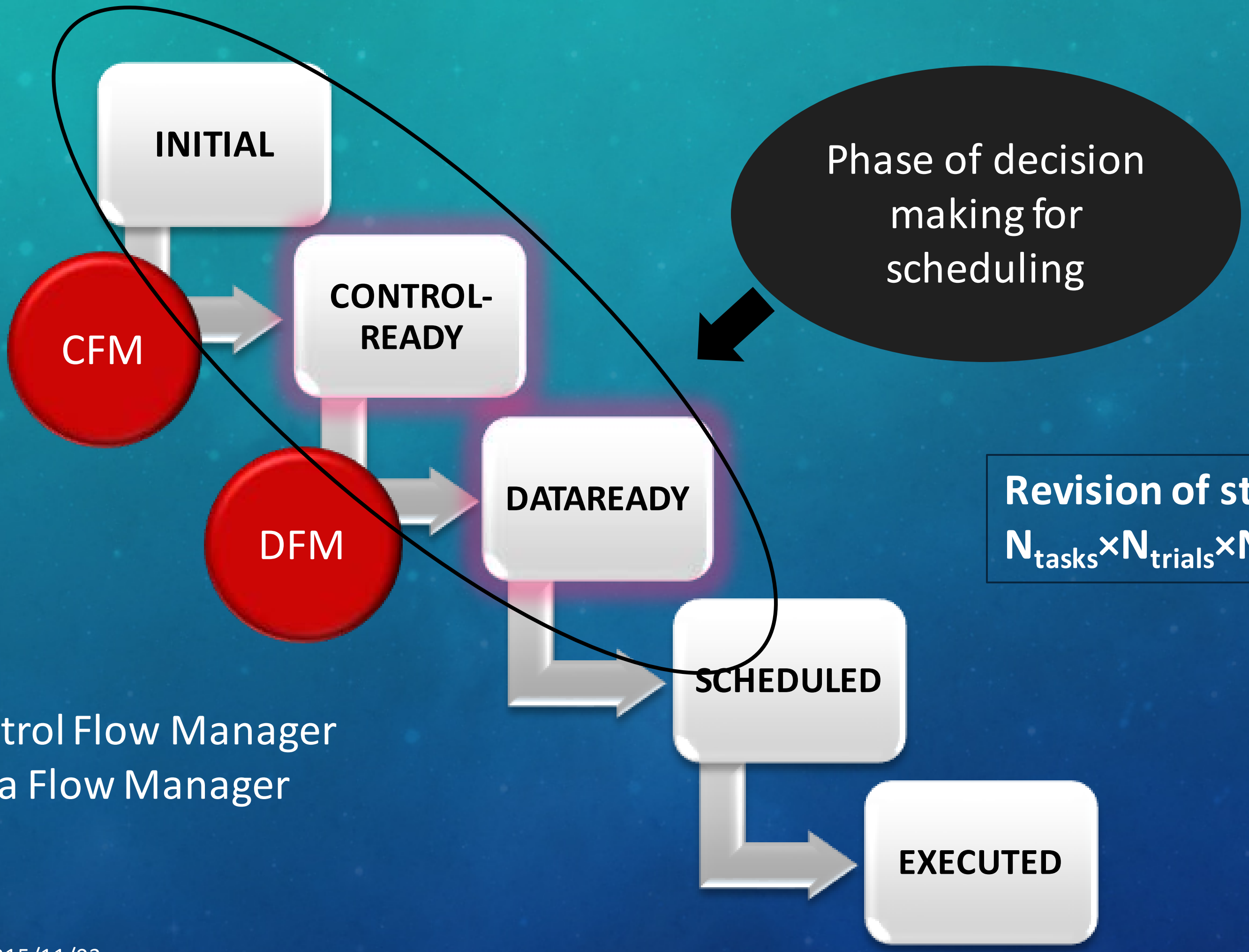
Control flow (CF) rules
- ✓ matching tasks with events

Data flow (DF) rules
- ✓ matching tasks with their data inputs





GAUDI HIVE: FINITE STATE MACHINE

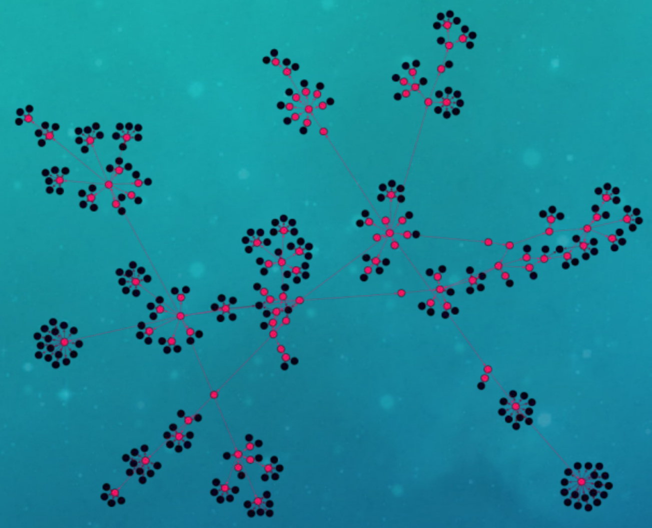


Revision of state is engaged $N_{tasks} \times N_{trials} \times N_{events}$ times!

CFM := Control Flow Manager
DFM := Data Flow Manager

DECISION MAKING IN SCHEDULING

CF manager

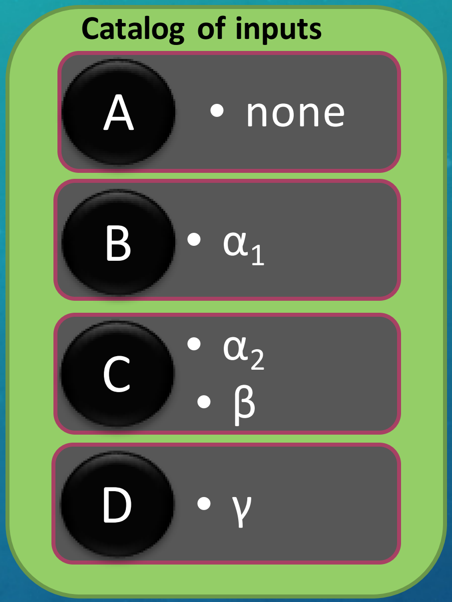


Operation:

- Global “waterfall” graph traversals

(each time a check or update of task’s state is requested)

DF manager



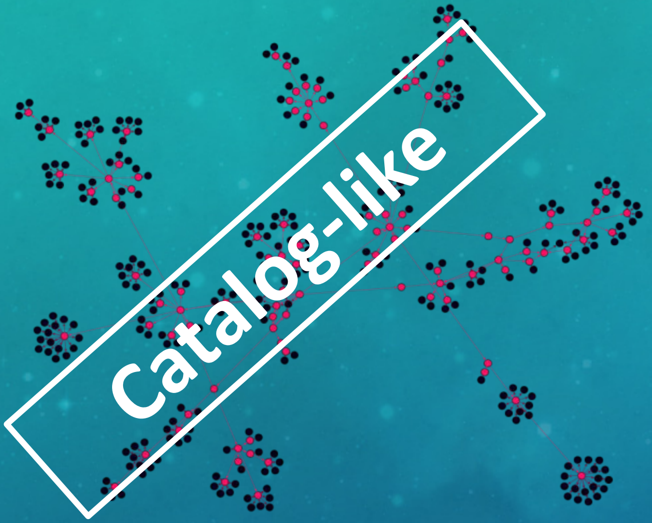
Operation:

- Catalog look-ups

(each time a check or update of task’s state is requested)

DECISION MAKING IN SCHEDULING

CF manager



Problems:

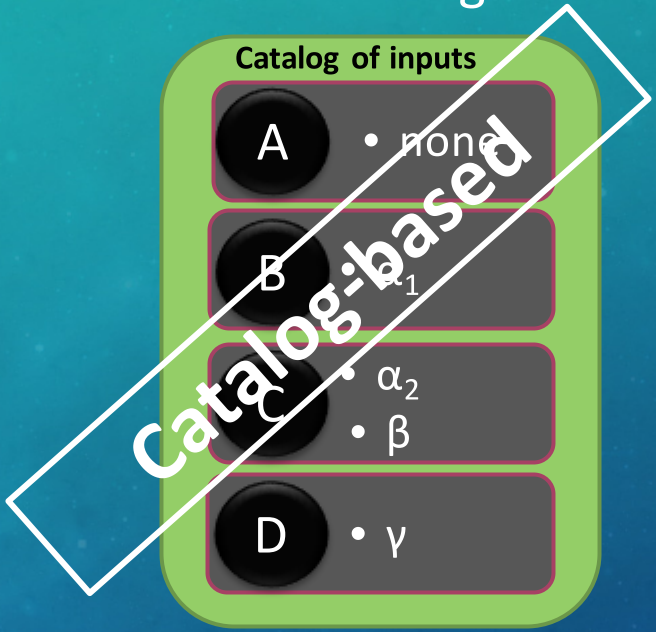
- Complexity:

Worst & Average: $O(n_a + n_d)$/iter

- Timing:

Wasting CPU on unnecessary “blank-fire” computations

DF manager



Problems:

- Complexity:

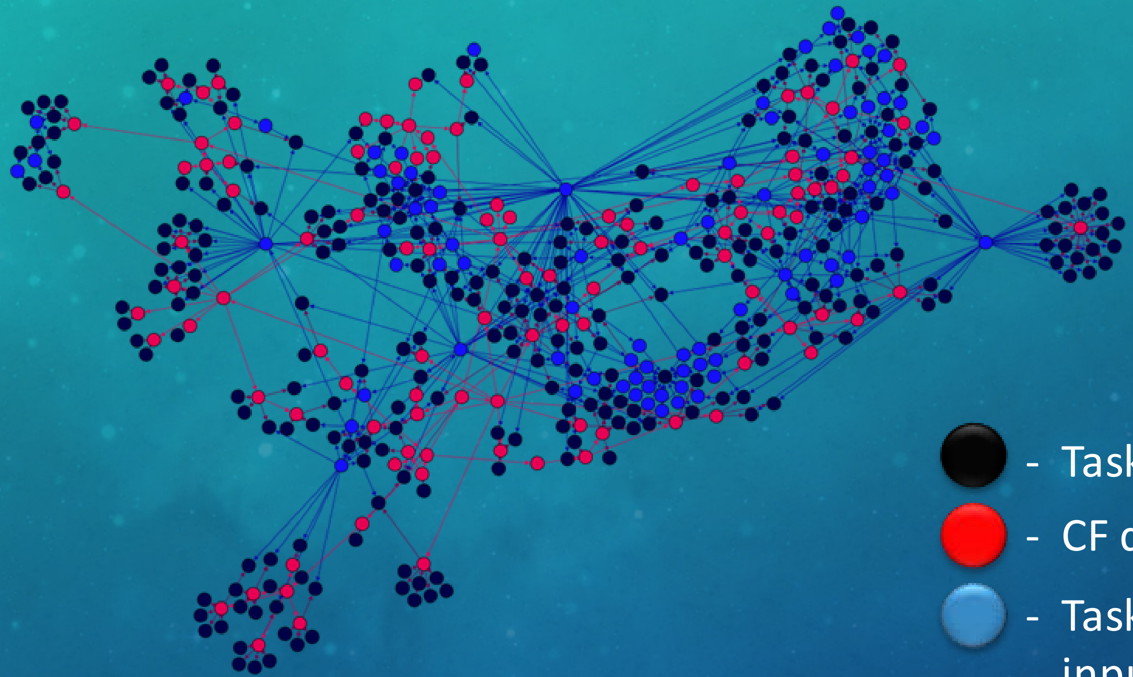
Worst: $O(n_a)$/iter, Average: $O(1)$/iter

- Timing:

Wasting CPU on necessary “blank-fire” computations: “blind-waiting-for-data” design

DECISION MAKING IN SCHEDULING

Graph-based decision making unit



- Tasks
- CF decision hubs
- Task data inputs/outputs

✓ Ideal information partitioning

✓ Only one component for both CF and DF decisions

✓ Reach spectrum of insights on topology of the algorithms' precedence

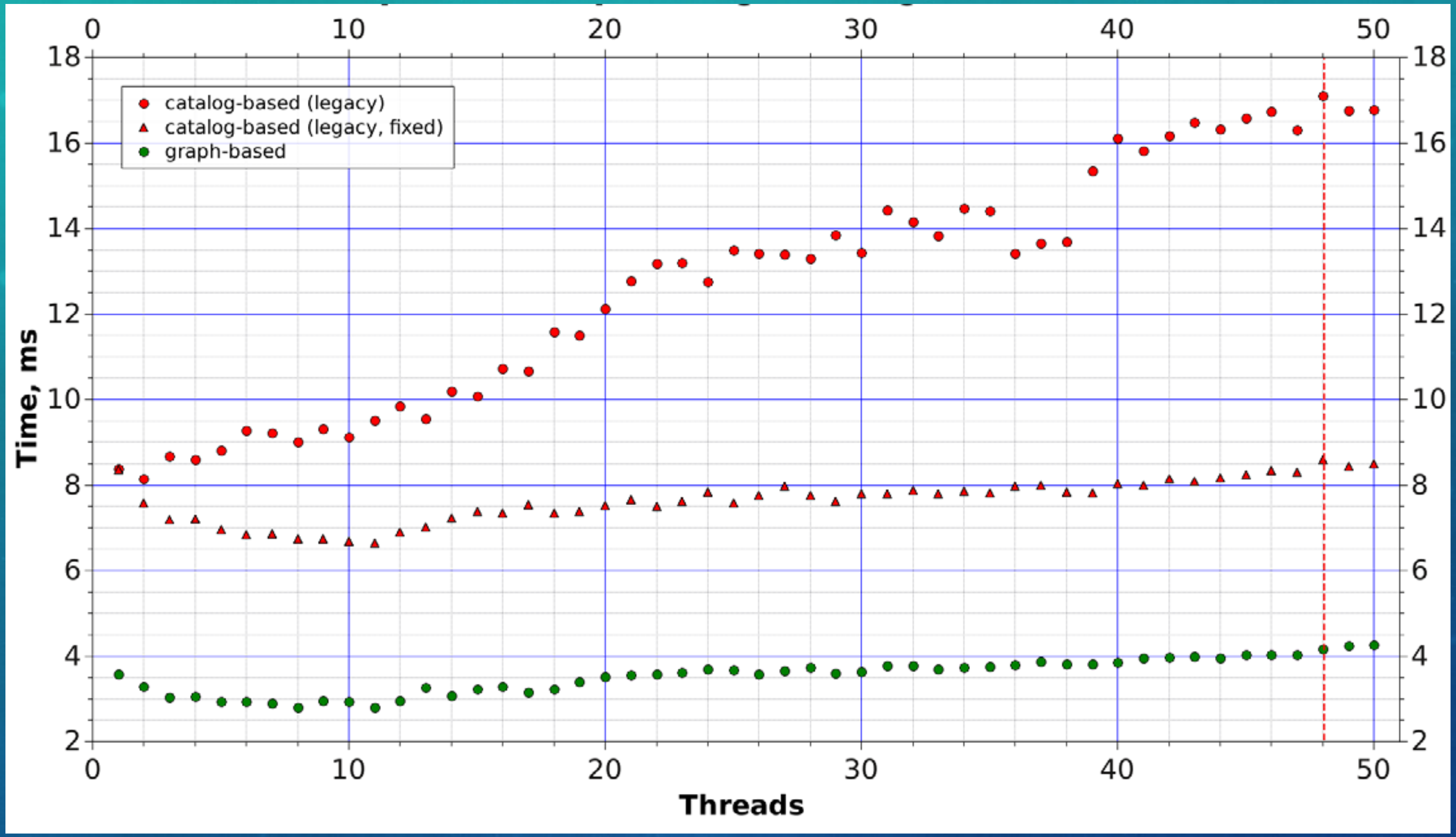
DECISION MAKING COMPLEXITY

	CF decisions	DF decisions
Catalog-based	Worst & Average: $O(n_t + n_d)$	Worst: $O(n_t)$, Average: $O(1)$
Graph-based	Worst: $O(n_t)$, Average: $O(1)$	Worst & Average: $O(1)$

n_t - number of tasks

n_d - number of decision hubs

TIME OF DECISION MAKING



Total time spent to reason about precedence rules per event (spans 263 tasks)

CONTENT

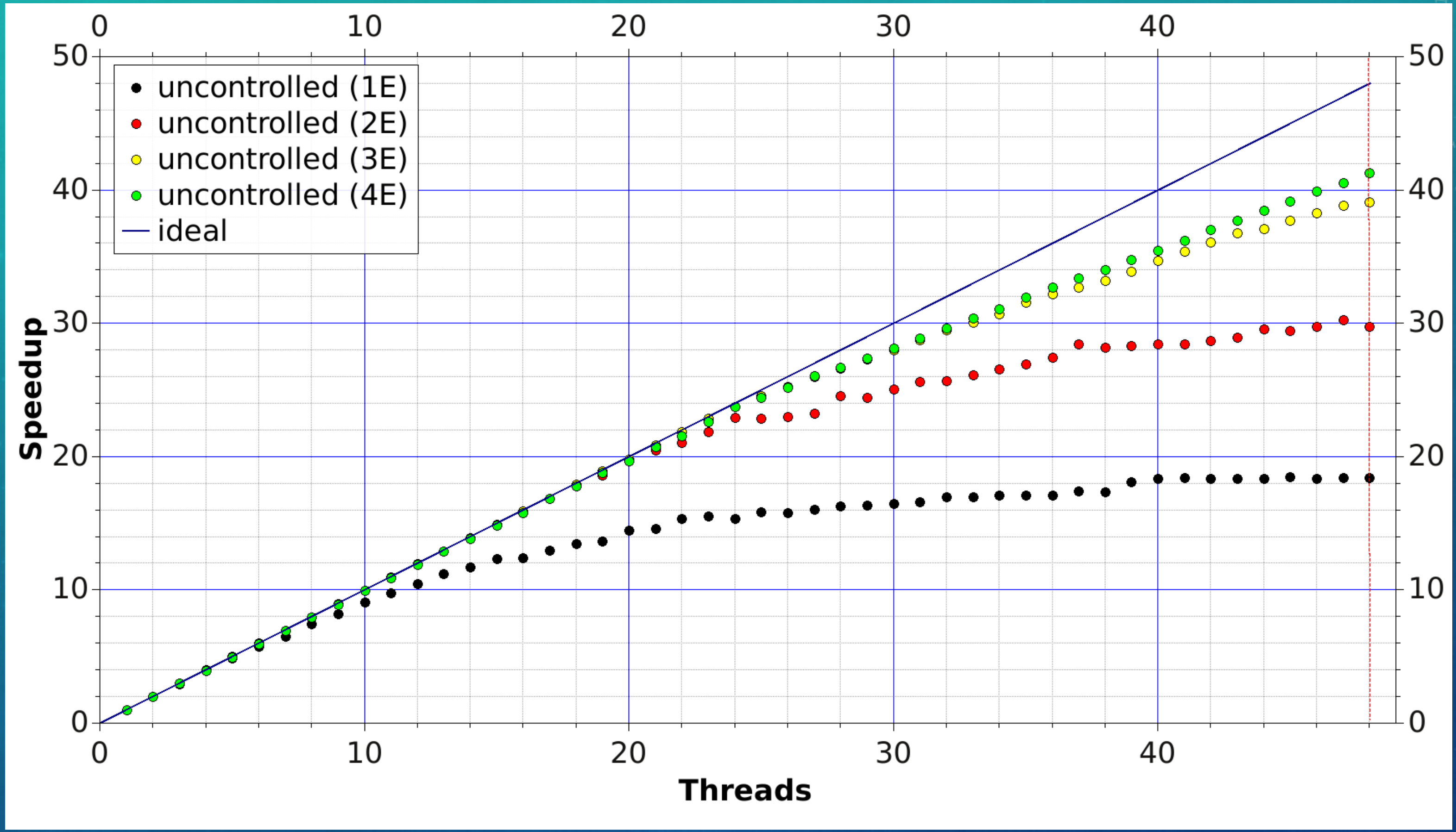
- Introduction
- Concurrency control: reactive scheduling 

- Speedup and scalability with reactive scheduling 

- Concurrency control: predictive scheduling 

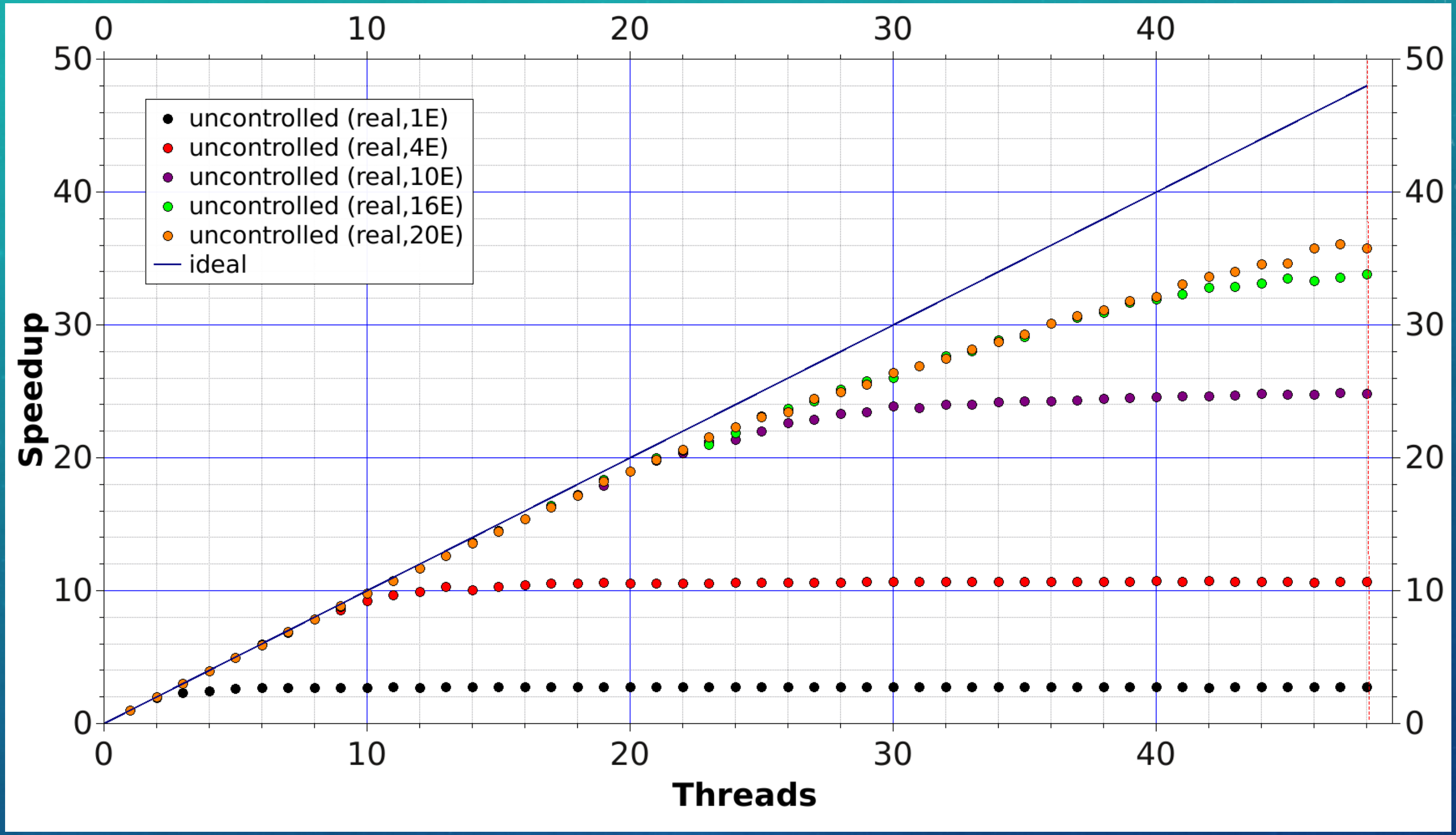
- Generic analysis of speedup constraints 

SPEEDUP SATURATION: OPTIMISTIC TASK TIMING MAP



Intra-event + inter-event mode (uniform task timing ~10ms)

SPEEDUP SATURATION: PESSIMISTIC TASK TIMING MAP



Intra-event + inter-event mode (real task timing)

WAYS TO FACILITATE SCALABILITY

- Further reduce framework-level overhead

  (not discussed in this talk)

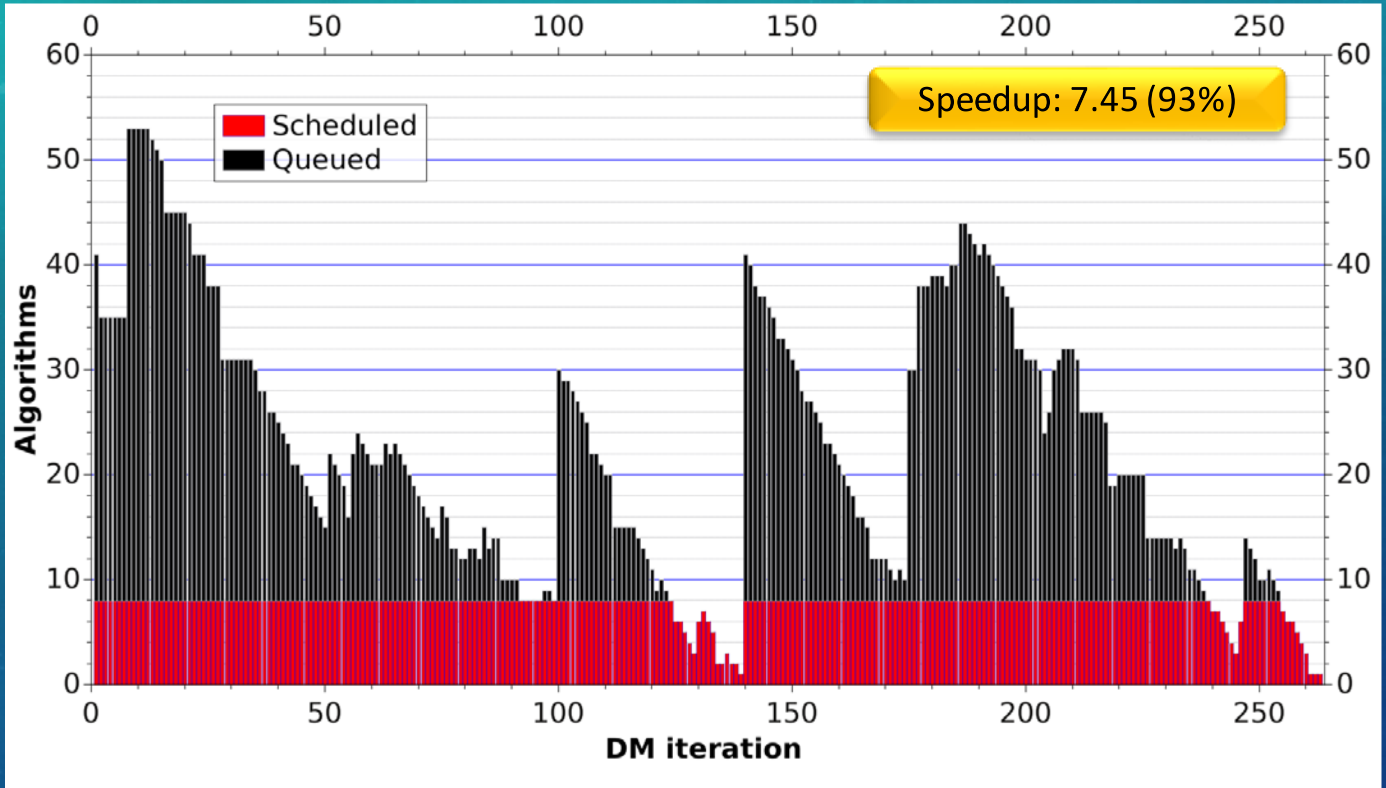
WAYS TO FACILITATE SCALABILITY

- Further reduce framework-level overhead

  (not discussed in this talk)

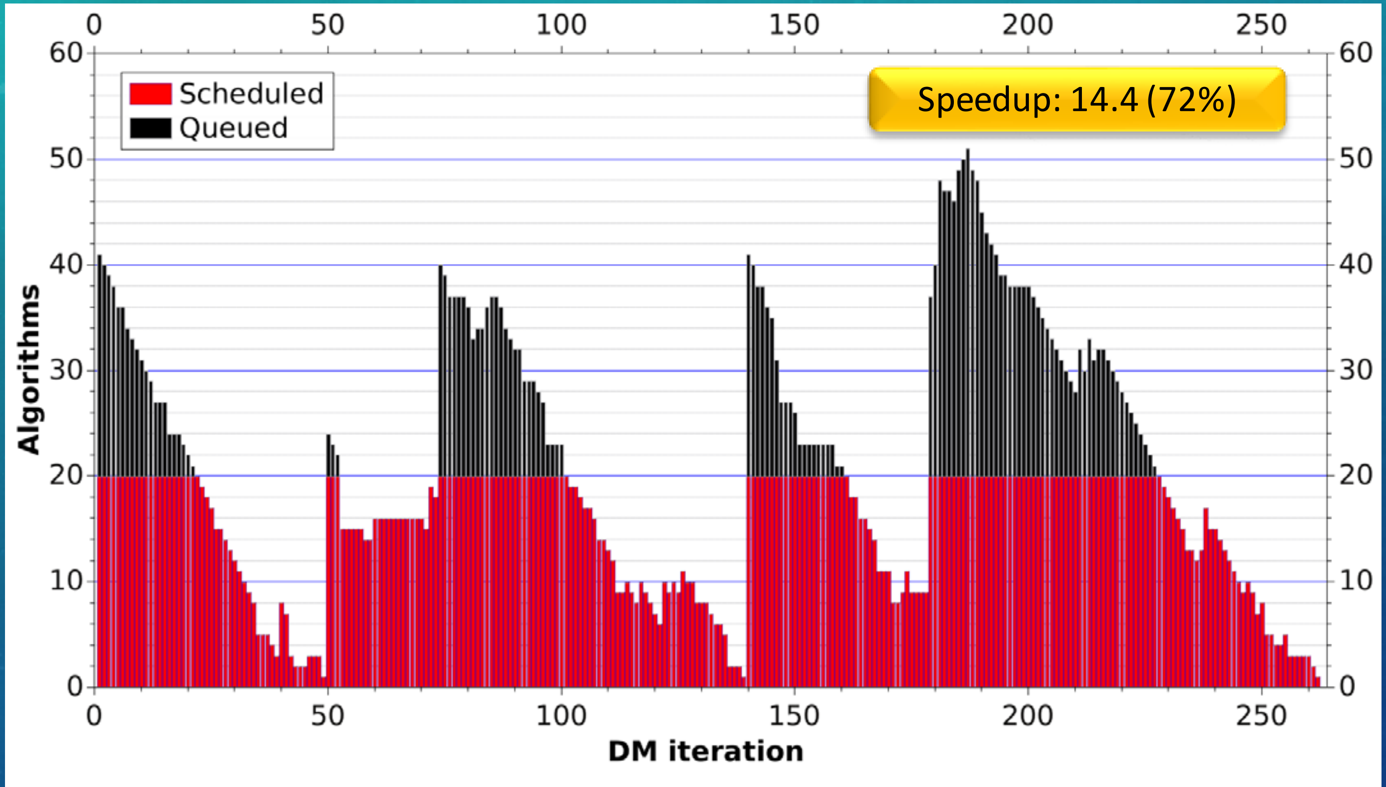
- Improve intra-event concurrency dynamics
  - its low level pushes to overuse the inter-event concurrency
  - may help to better utilize data locality

INTRA-EVENT CONCURRENCY DYNAMICS



Reactive scheduling only (**8 threads**, 263 tasks per event)

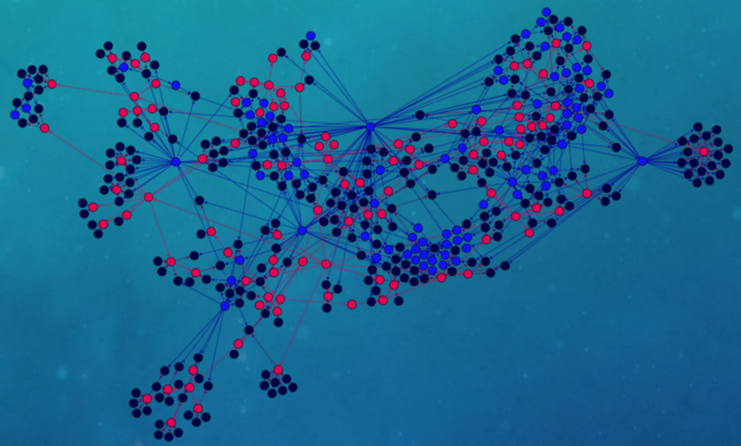
INTRA-EVENT CONCURRENCY DYNAMICS



Reactive scheduling only (**20 threads**, 263 tasks per event)

HARMFUL DEGREES OF FREEDOM...

- Typical task precedence graphs (in LHCb) are significantly heterogeneous



- Concurrency disclosure dynamics is drastically dependent on execution front
  - uncontrolled in Gaudi Hive minimalistic reactive scheduling

CONTENT

- Introduction
- Concurrency control: reactive scheduling 

- Speedup and scalability with reactive scheduling 

- Concurrency control: predictive scheduling 

- Generic analysis of speedup constraints 

PREDICTIVE SCHEDULING IN GAUDI HIVE

What:

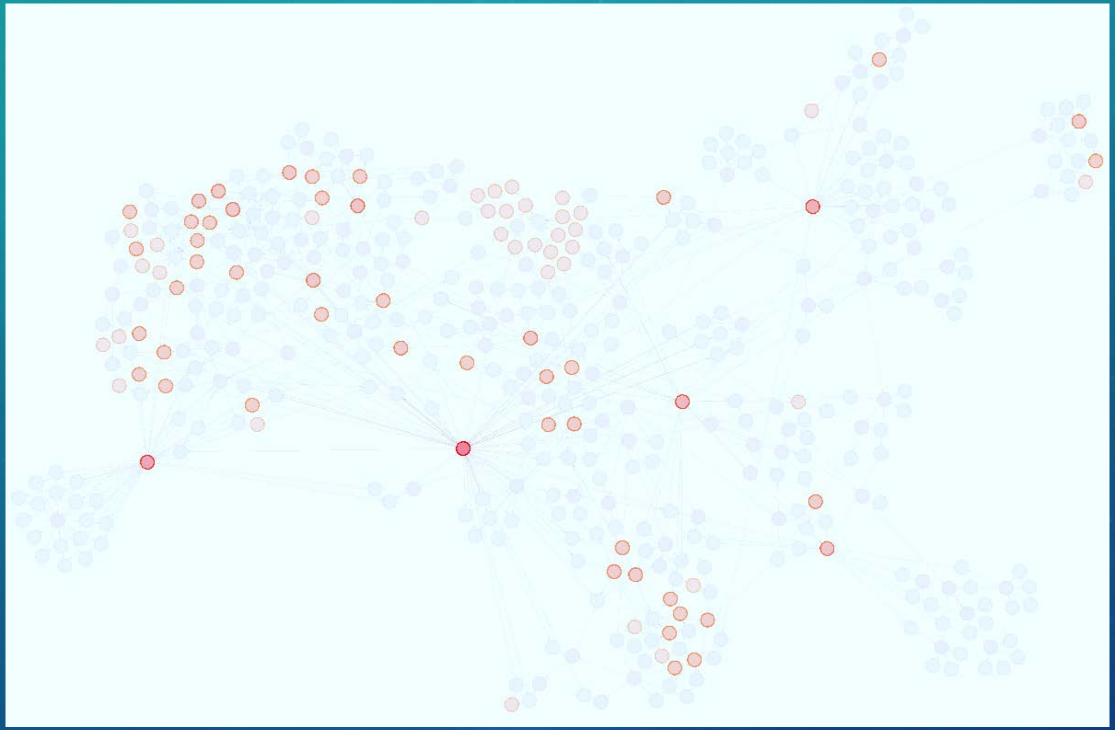
- maximize concurrency disclosure dynamics
  - or at least create facilitating pressure towards it

How:

- rank algorithms reflecting their ‘importance’ within precedence graph
  - plenty of ranking strategies studied elsewhere
- prioritize the queue of ready-to-run algorithms following each reactive iteration

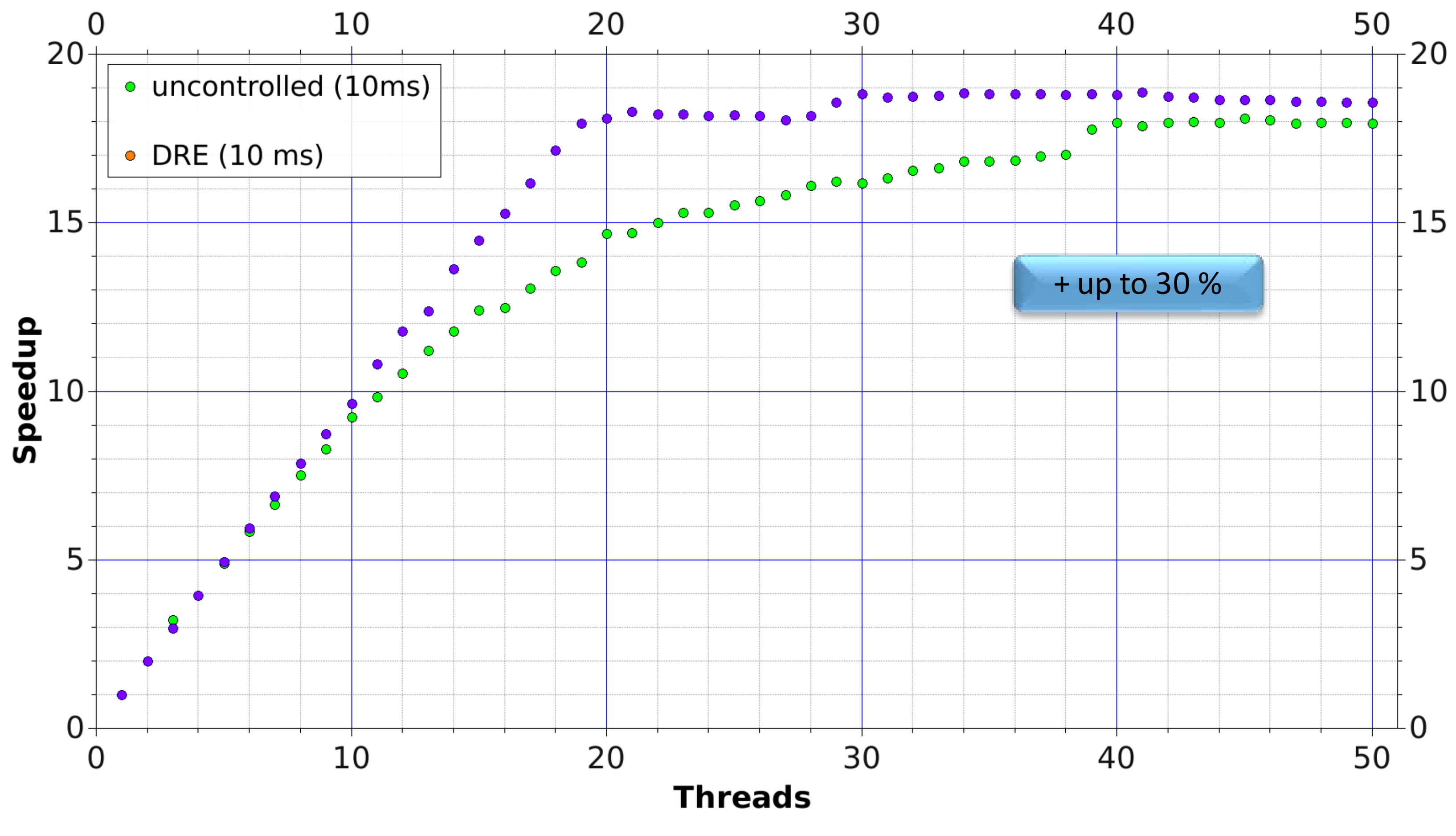
ASYMMETRY OF PRODUCTS CONSUMPTION

Rank algorithm by its products consumption extent



Precedence graph with all, but data nodes, faded out. Color intensity of a data node represents the number of its consumers

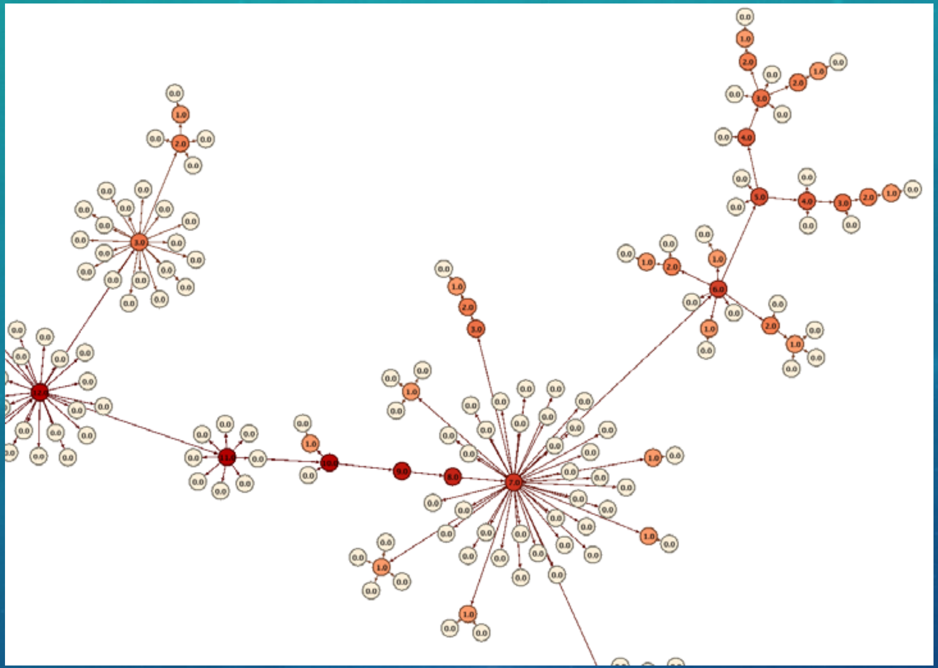
PREDICTIVE SCHEDULING: PRODUCTS CONSUMPTION EXTENT (PCE)



Uniform task timing map (~10ms)

PREDICTIVE SCHEDULING: DATA REALM ECCENTRICITY (DRE)

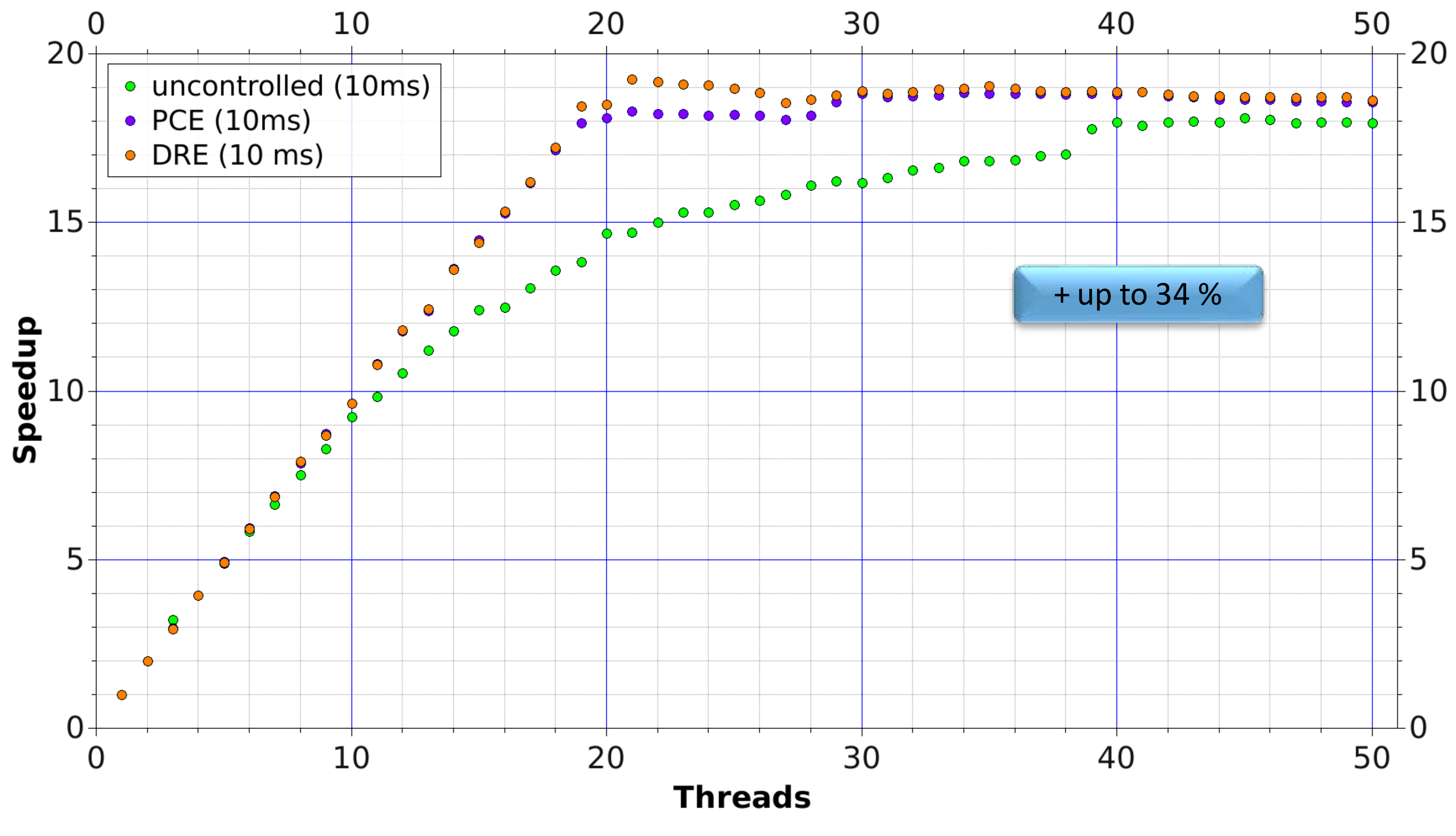
- rank algorithm by its eccentricity in data realm



Color intensity represents eccentricity-based rank

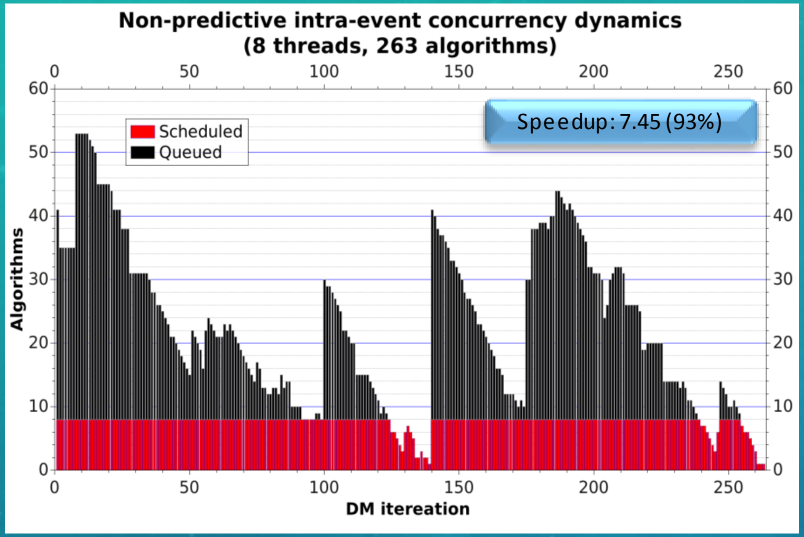
- implements critical path lookup technique in case of uniform task timing map
- note: not only graph diameter is tracked, but also all other sub-critical paths

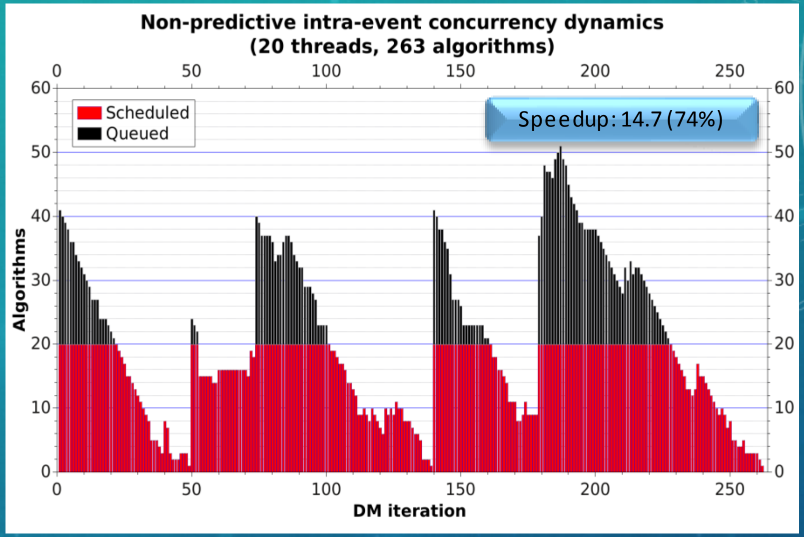
PREDICTIVE SCHEDULING: DATA REALM ECCENTRICITY (DRE)



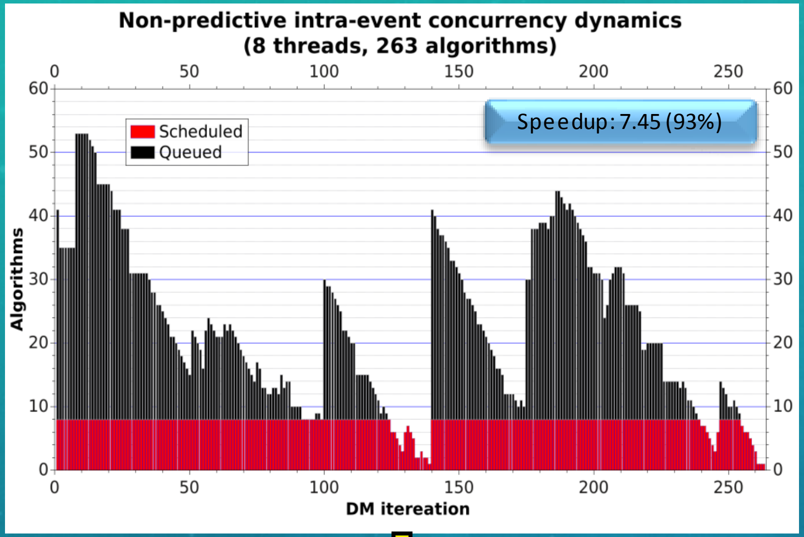
Uniform task timing map (~10ms)

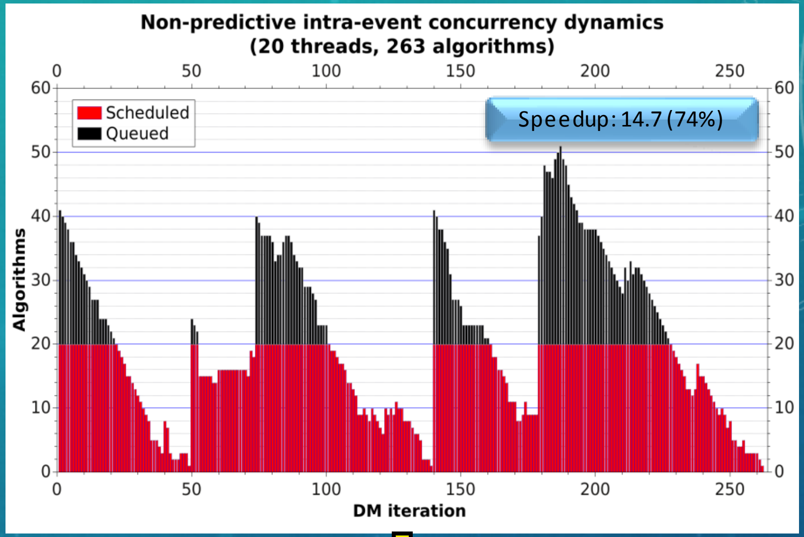
PREDICTIVE SCHEDULING: DRE MODE

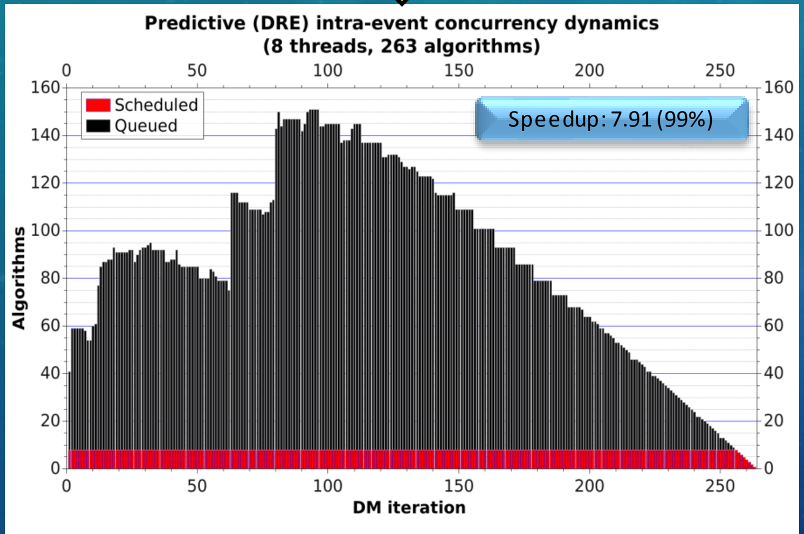


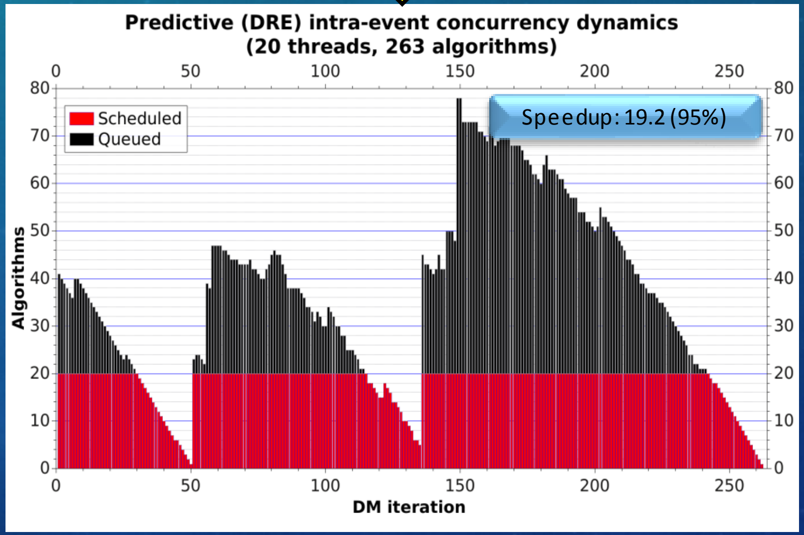


PREDICTIVE SCHEDULING: DRE MODE

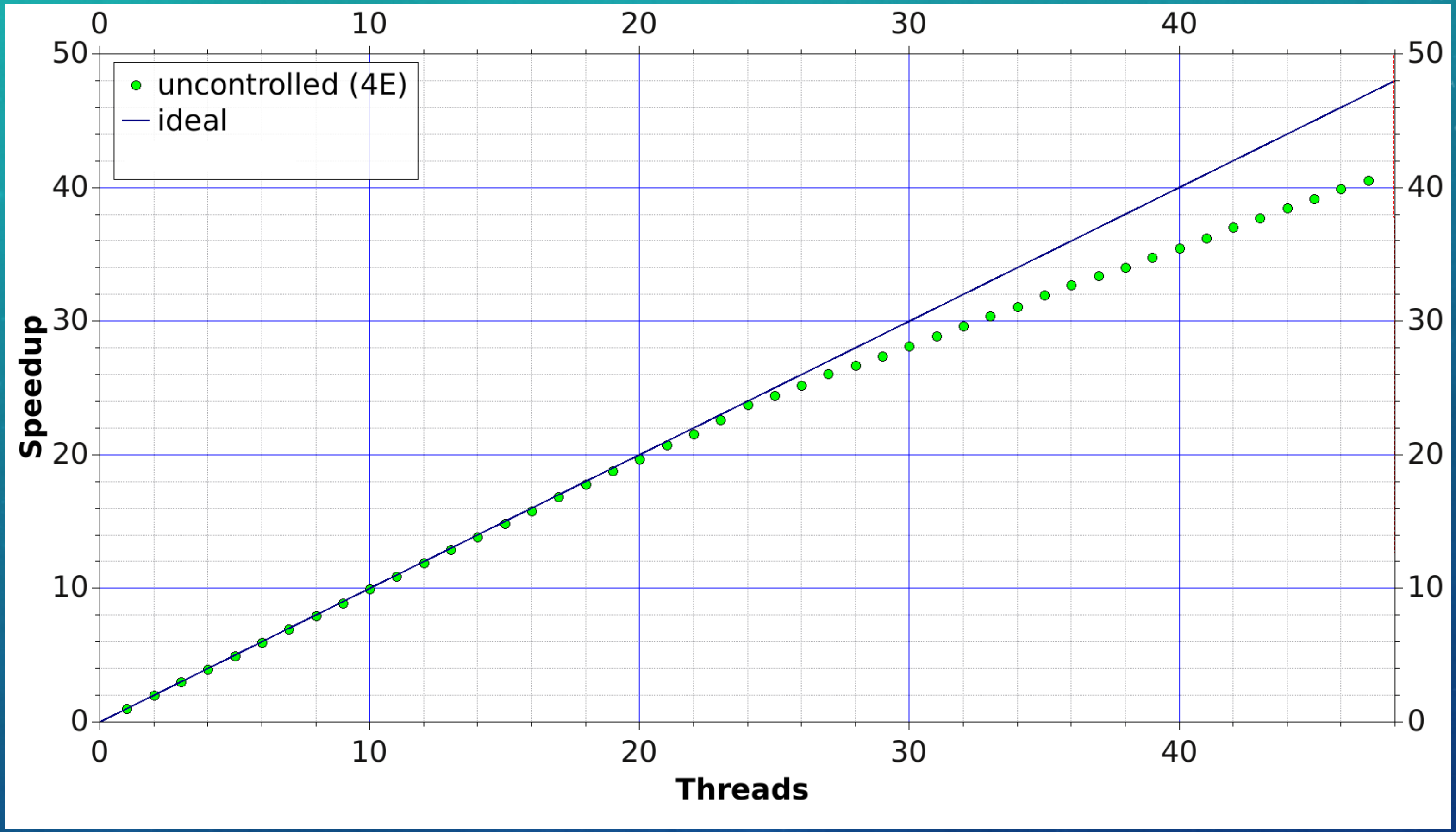






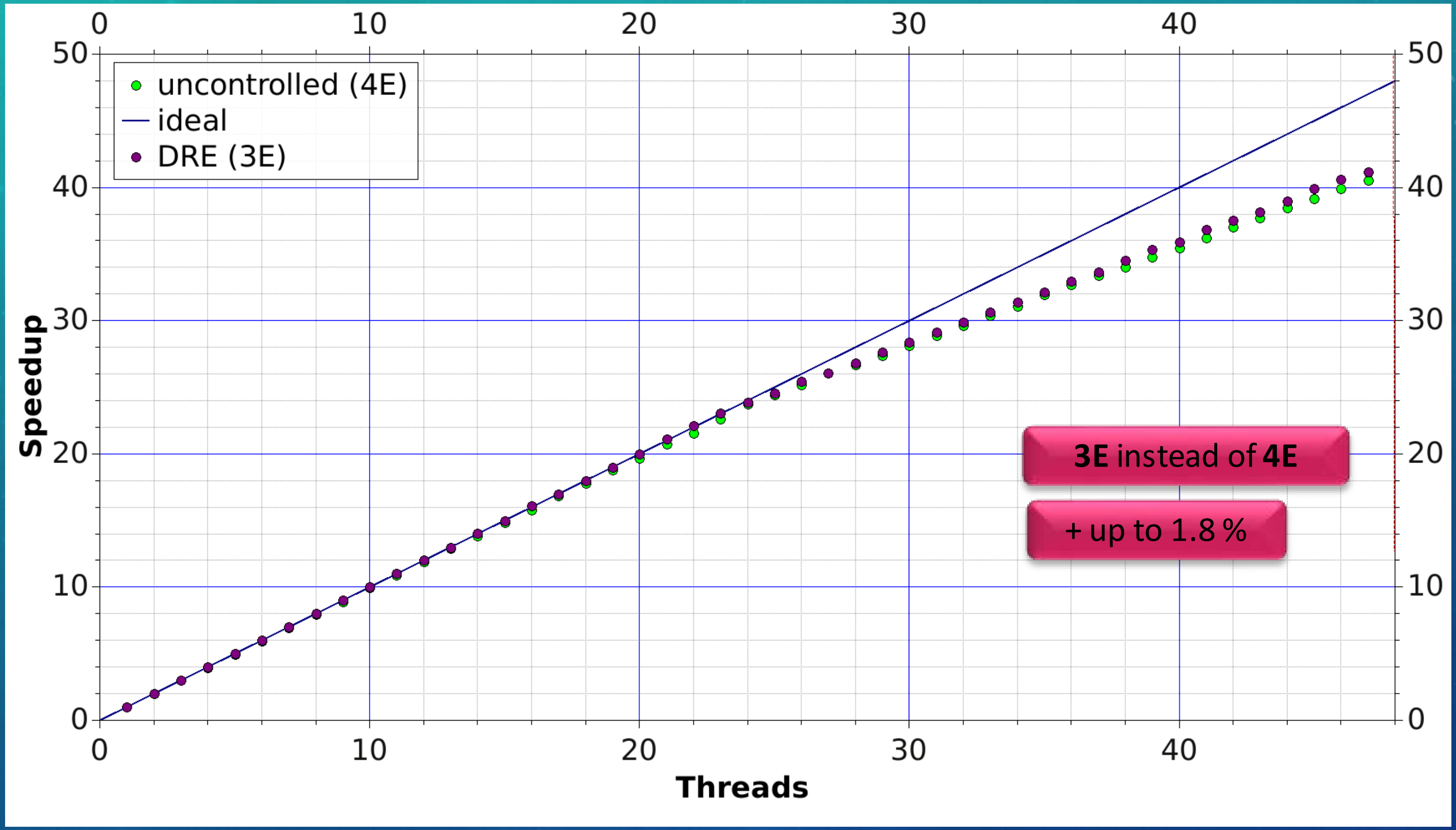


SPEEDUP SATURATION IN PREDICTIVE SCHEDULING



Intra-event + inter-event mode (uniform task timing map, ~10ms)

SPEEDUP SATURATION IN PREDICTIVE SCHEDULING



Intra-event + inter-event mode (uniform task timing map, ~10ms)

CONTENT

- Introduction
- Concurrency control: reactive scheduling 

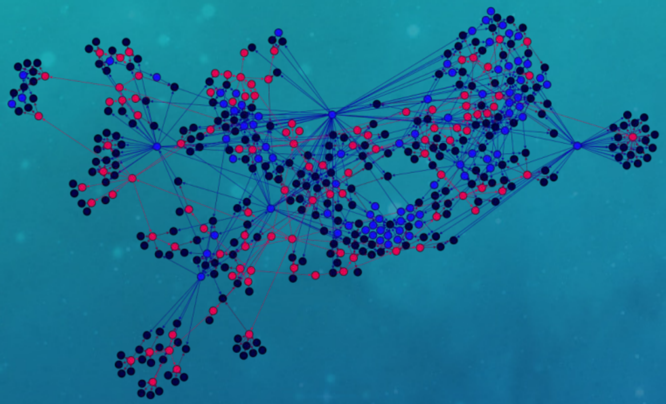
- Speedup and scalability with reactive scheduling 

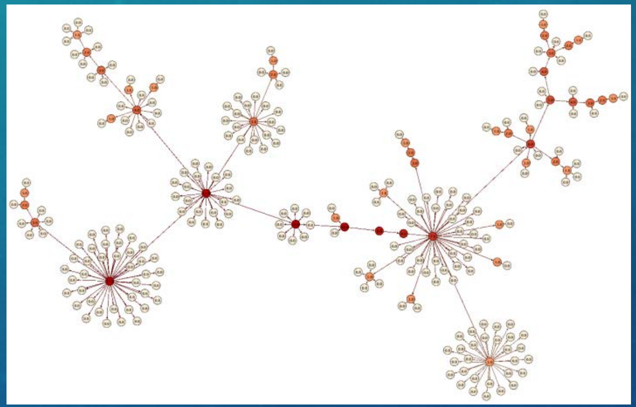
- Concurrency control: predictive scheduling 

- Generic analysis of speedup constraints 

GENERIC ANALYSIS OF PRECEDENCE CONSTRAINTS

- built-in tool available to create materialized views of polymorphous precedence graphs





- provides, for a given event and hardware platform:
  - visualization of critical and sub-critical paths
  - theoretical intra-event speedup limit
  - expertize on how to increase throughput in a given data processing workflow

CONCLUSION

Graph-based decision making in concurrency control allows to:

- (Algorithmically) Reduce decision making time by 2x, and improve its asymptotic complexity
- Implement predictive scheduling with diverse look-ahead strategies, which:
  - yield significant improvement in intra-event speedup (~30% in LHCb event reconstruction workflow)
  - allow to achieve higher throughput in harsh data processing conditions

SPARE SLIDES

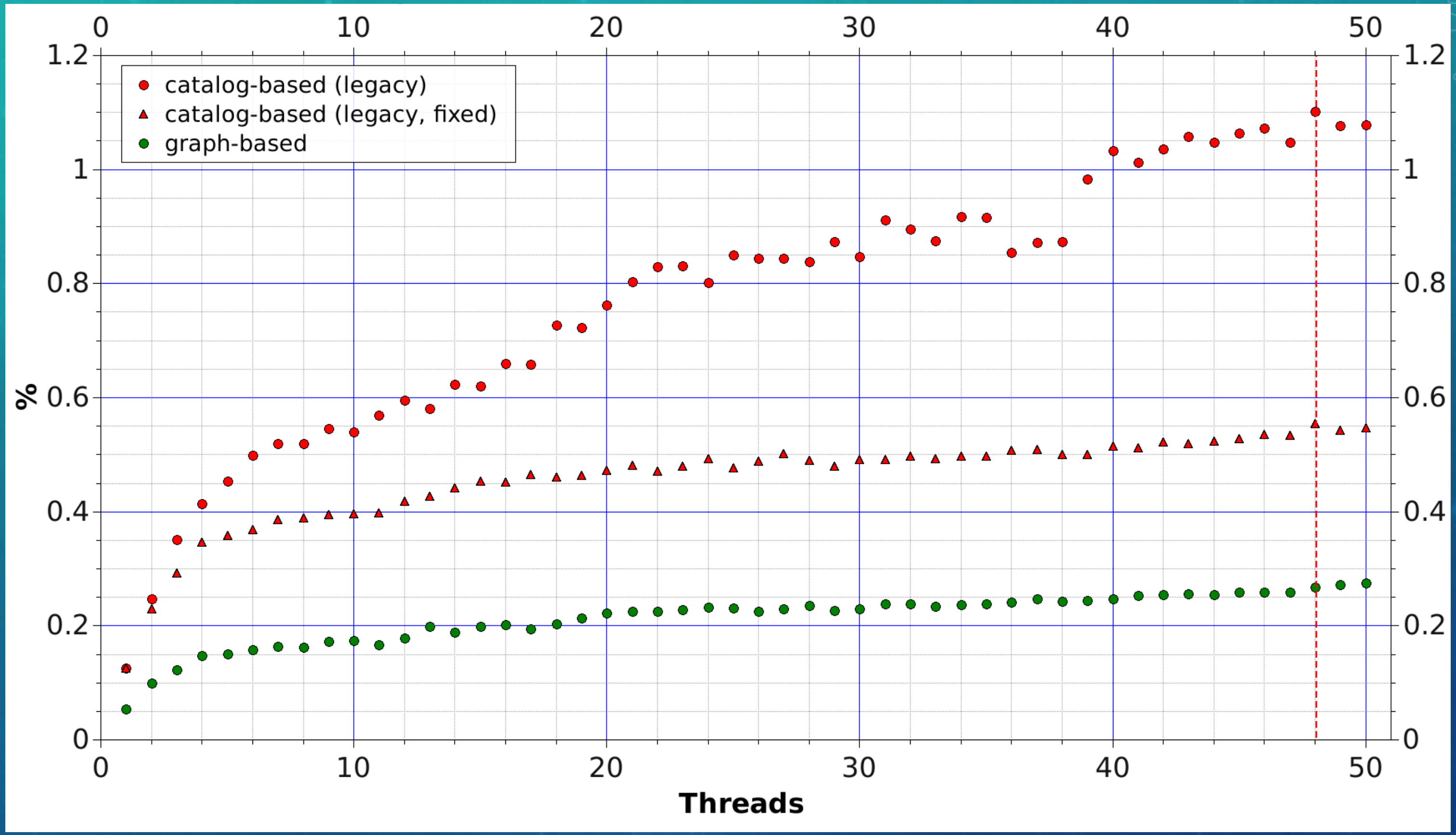
TESTBED FOR BENCHMARKING

- Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz
- 2 sockets: 24 + 24 HT
- L2 256KB, L3 30 MB

Data processing workflow configuration:

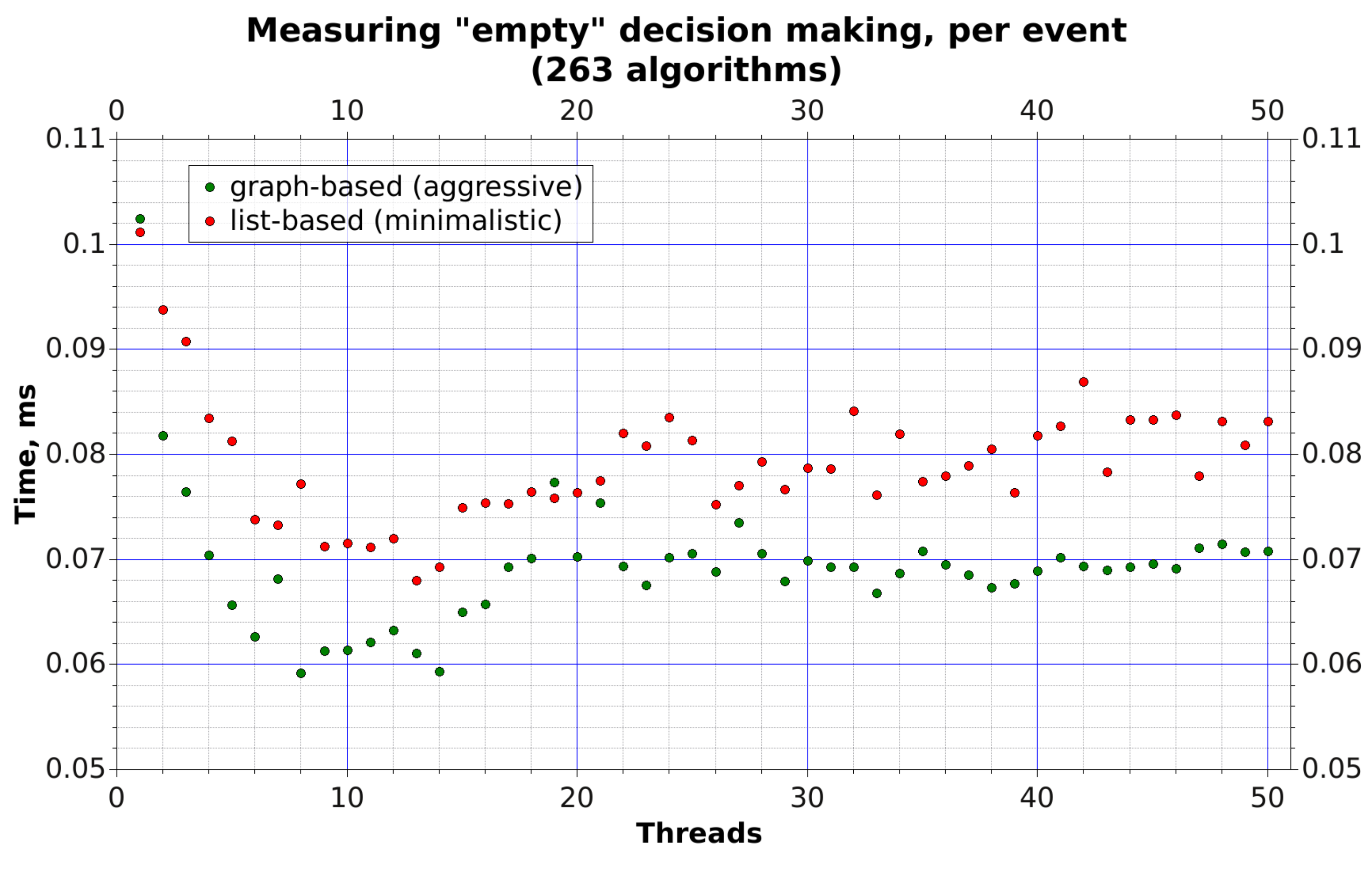
- Precedence graph of close to real size and topology (LHCb Brunel reconstruction case)
- CPUCrunchers as tasks
- Real/uniform tasks' timings

RATIO OF DECISION MAKING TIME TO EVENT PROCESSING TIME

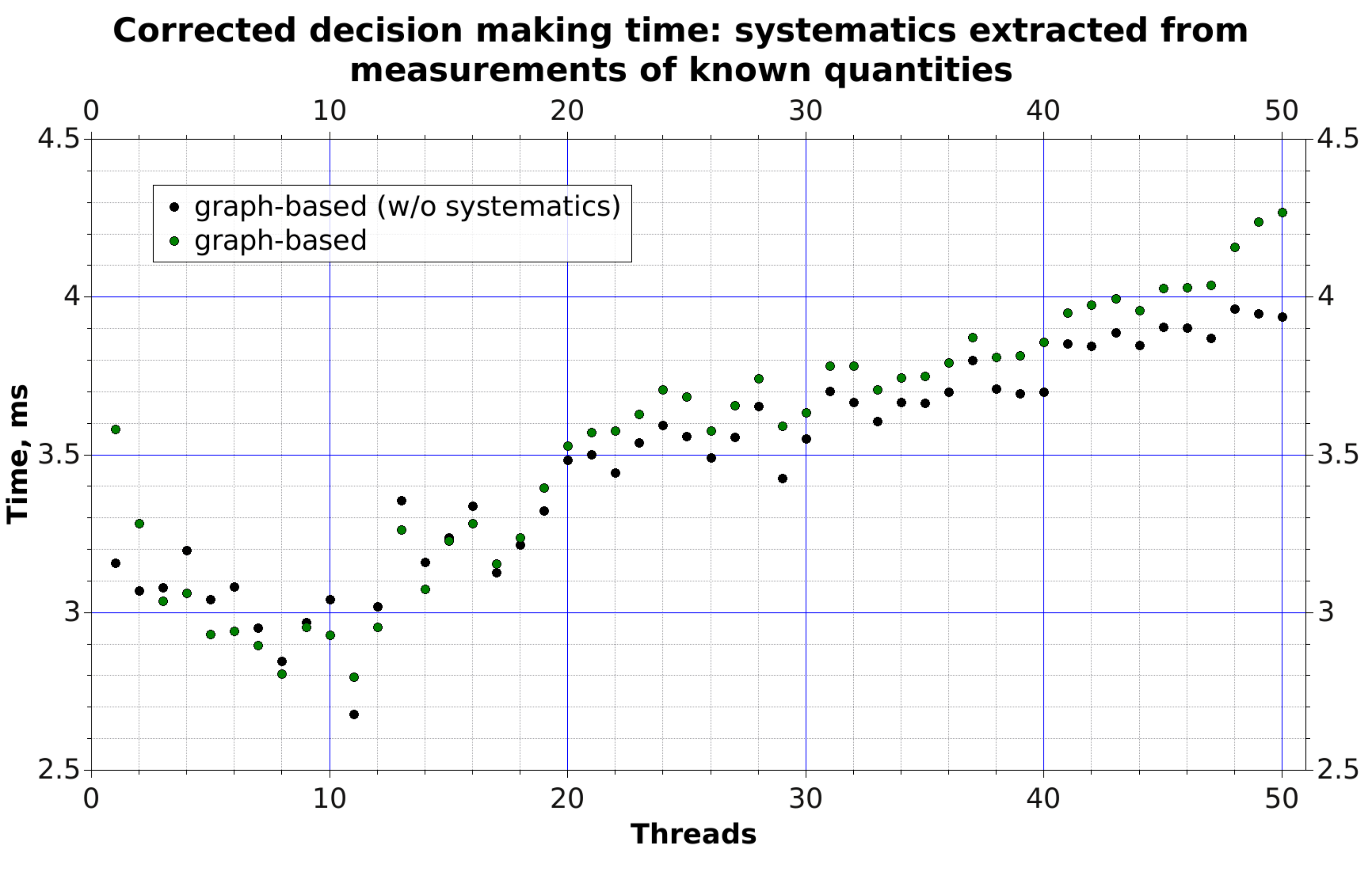


Chosen max. speedup of concurrent event processing is conservative: 4x !

SYSTEMATICS: ZERO MEASUREMENT



SYSTEMATICS BY MEASUREMENT OF KNOWN DURATION



RUNAWAY OF DECISION MAKING