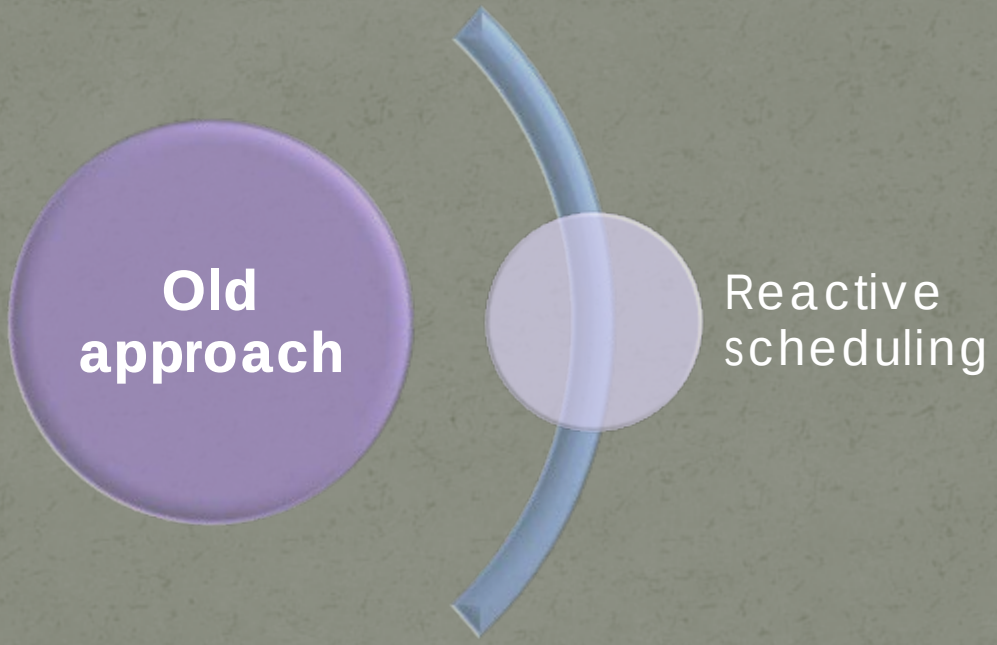
Graph-based Scheduling in GaudiHive

Illya Shapoval, Marco Clemencic

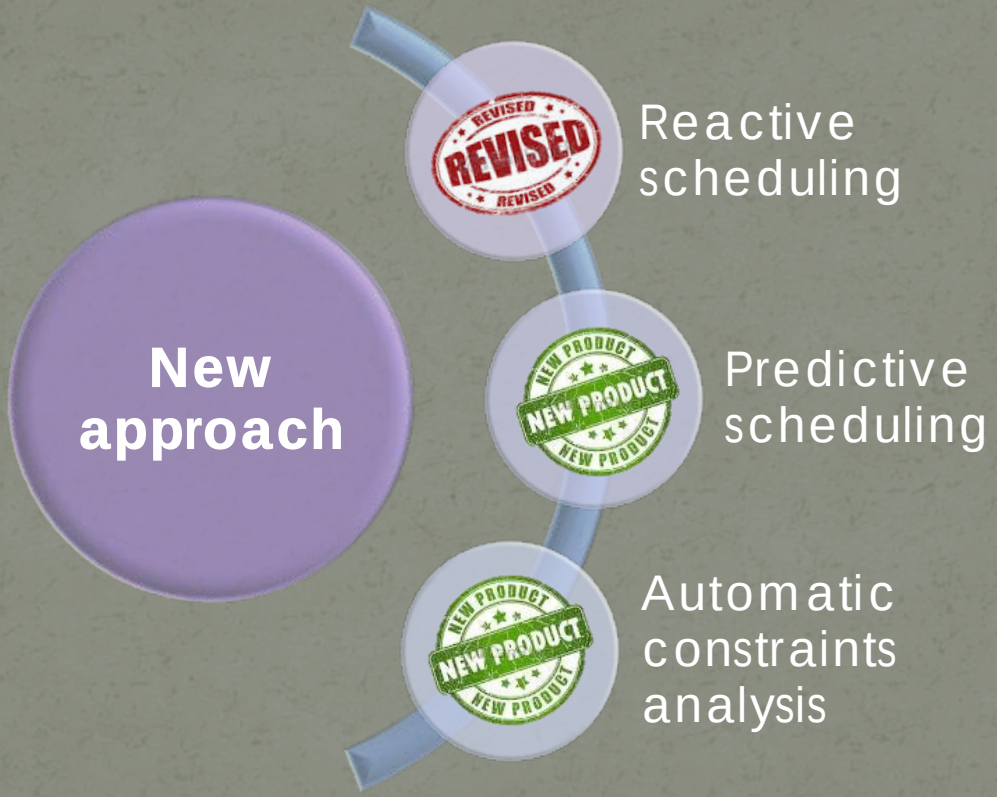
CERN, UNIFE, KIPT, INFN-FE

CERN
11 February 2015
Forum on Concurrent Programming Models and Frameworks

Subject



Subject


New approach
Reactive scheduling
Predictive scheduling
Automatic constraints analysis

Contents

- Introduction
  - Legacy approach to decision making
  - New approach to decision making 

- Concurrency control: reactive scheduling 

- GaudiHive scalability on close to real workflow topologies 

- Concurrency control: predictive scheduling 

- Generic analysis of speedup constraints 

- Interplay of speedup with algorithm's timing 

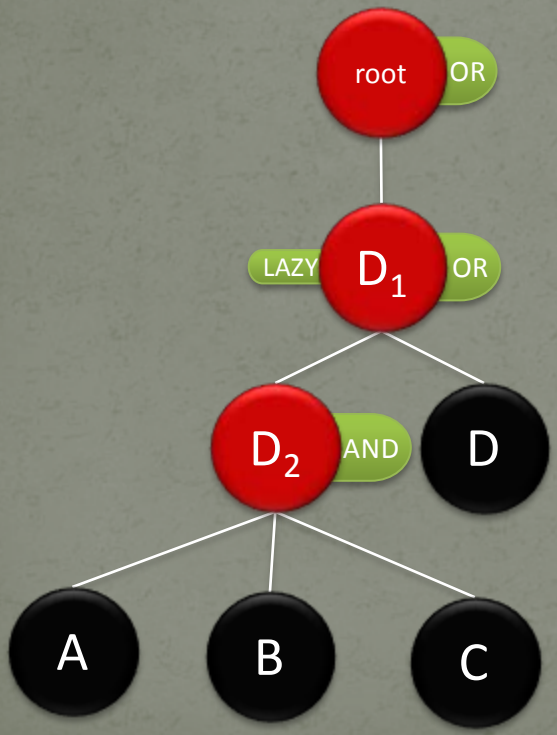
Intra-event algorithm precedence rules

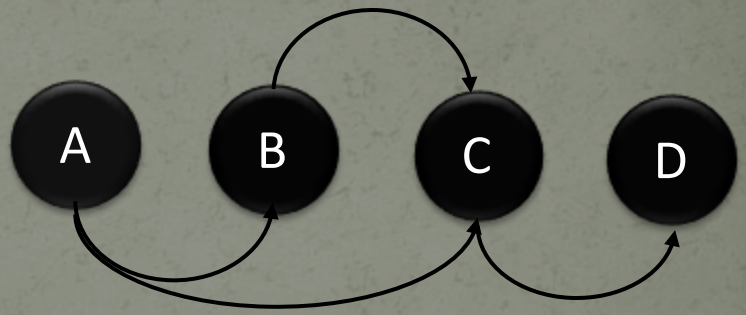
Control flow (CF) rules

- matching algorithms with events

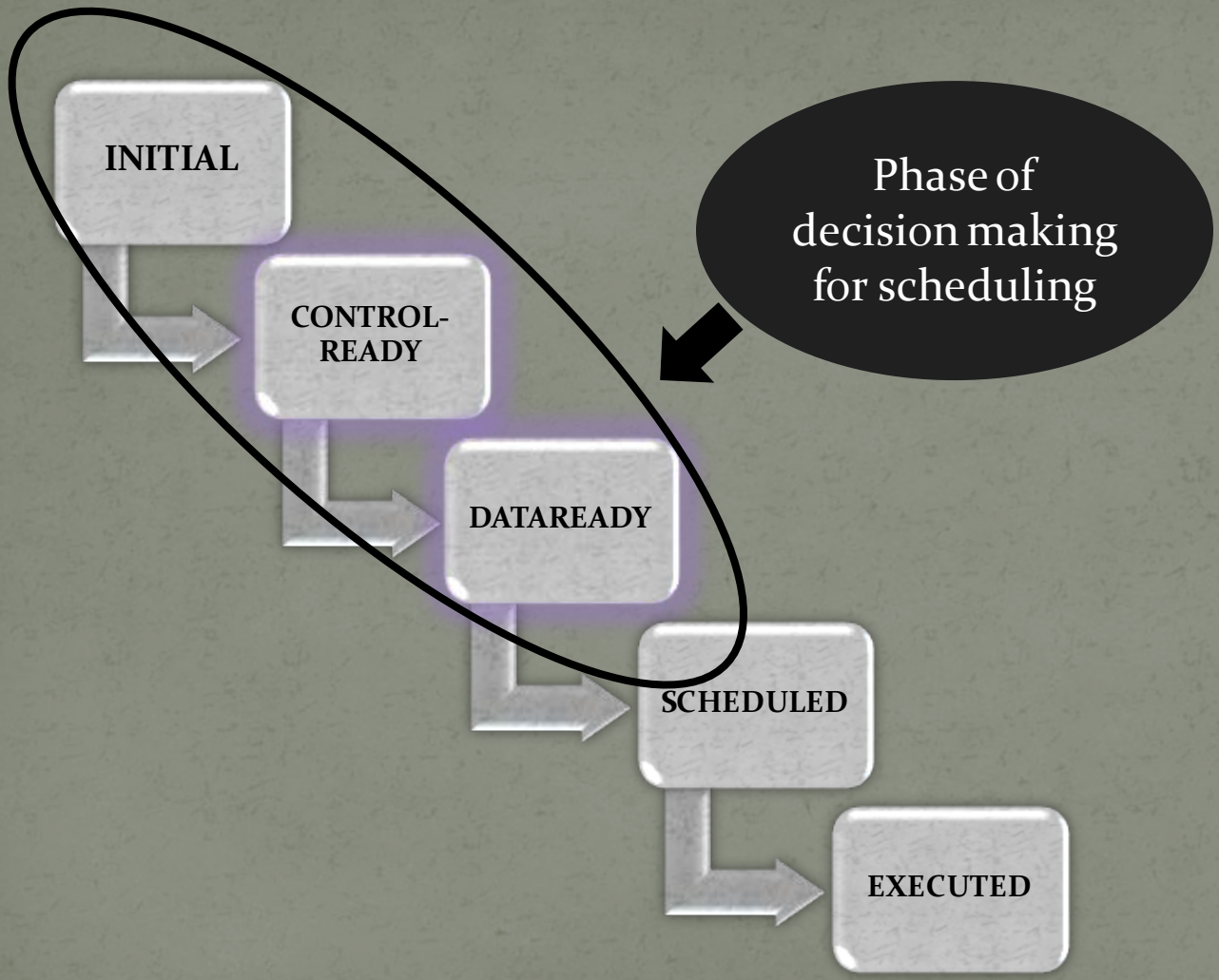
Data flow (DF) rules

- matching algorithms with their data inputs

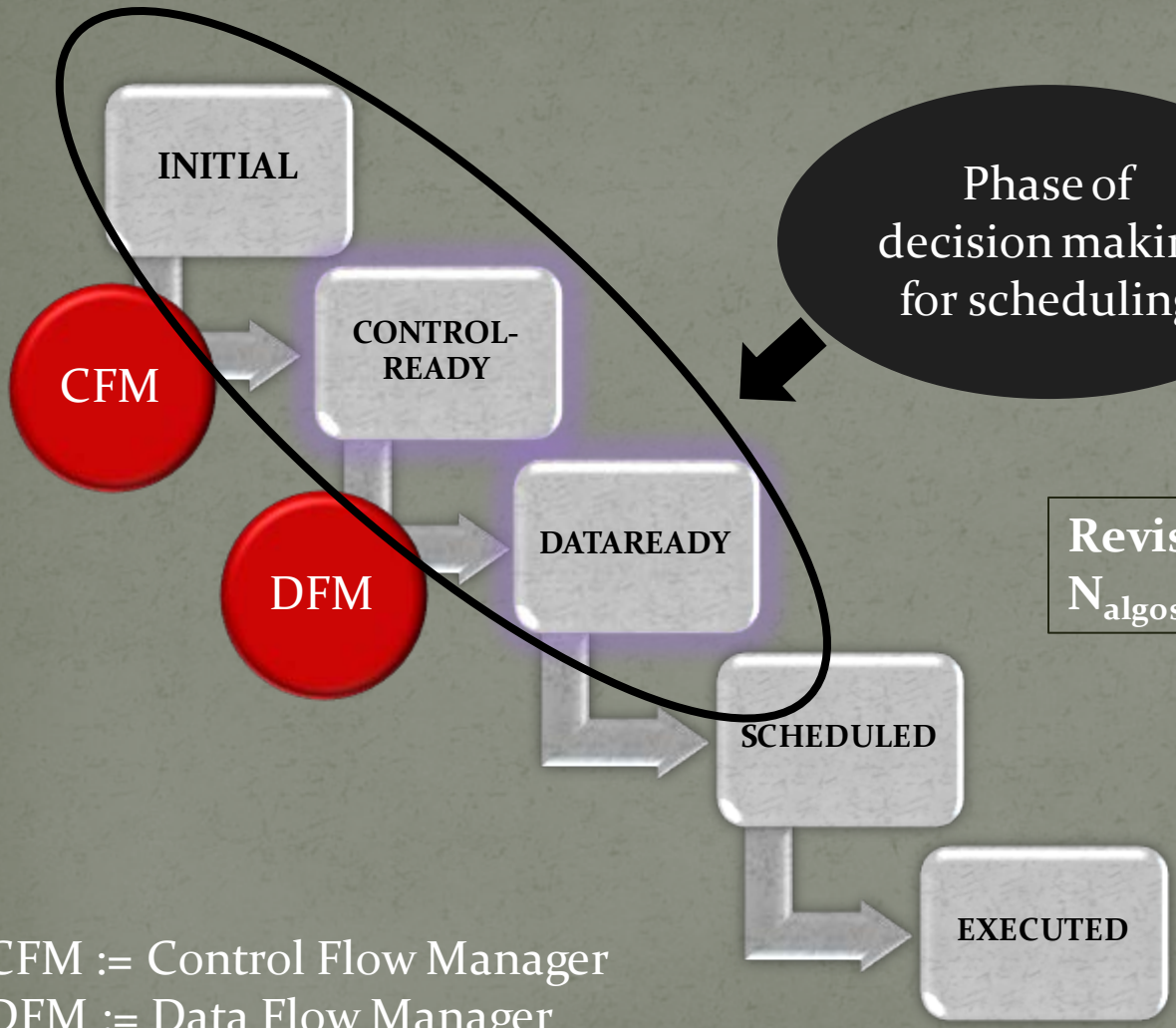




GaudiHive: finite state machine



GaudiHive: finite state machine

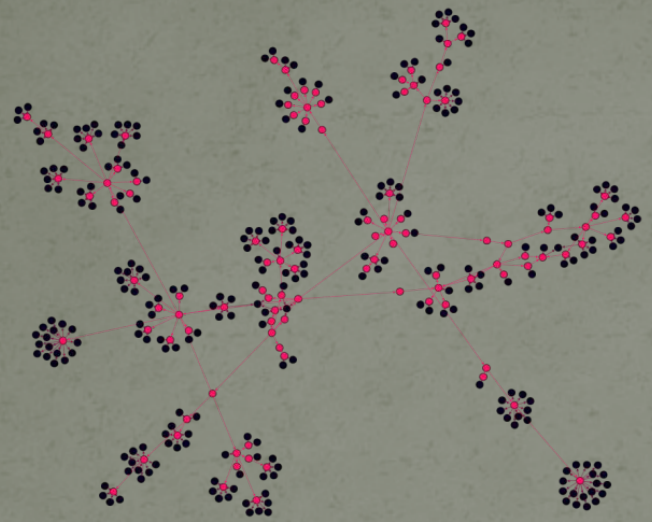


Revision of state is engaged $N_{algos} \times N_{trials} \times N_{events}$ times!

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

Decision making in scheduling

CF manager

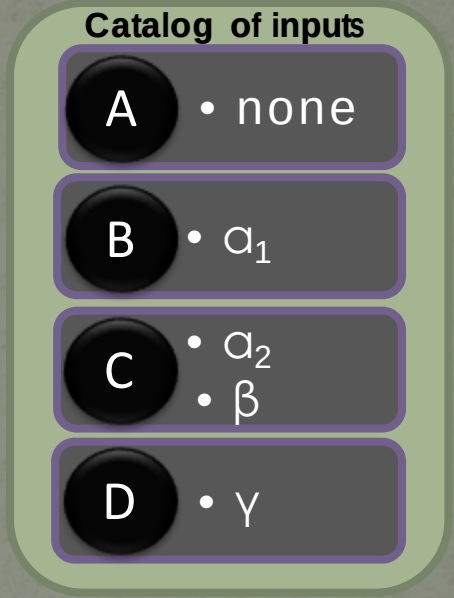


Operation:

- Global "waterfall" graph traversals

(each time a check or update of algorithm's state is needed)

DF manager



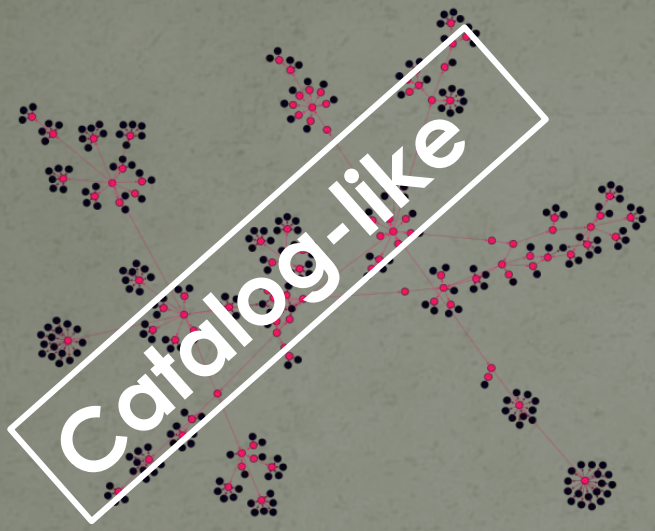
Operation:

- Catalog look-ups

(each time a check or update of algorithm's state is needed)

Decision making in scheduling

CF manager



Problems:

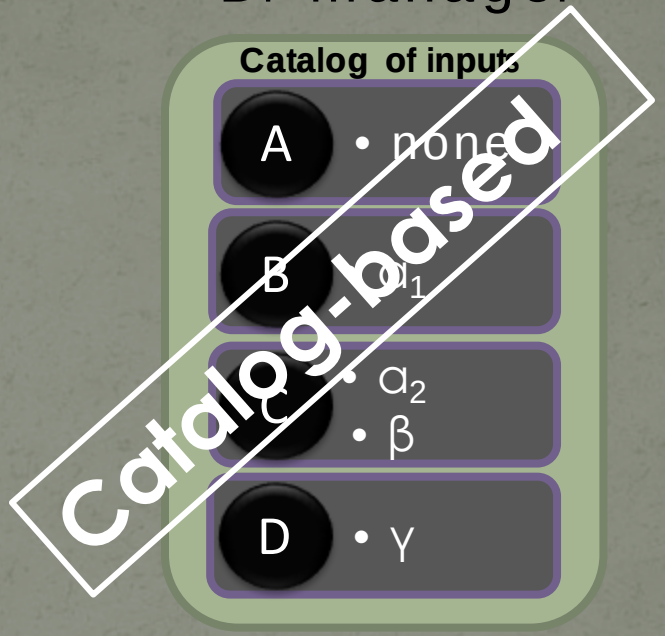
- Complexity:

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

- Timing:

Wasting CPU on unnecessary "blank-fire" computations

DF manager



Problems:

- Complexity:

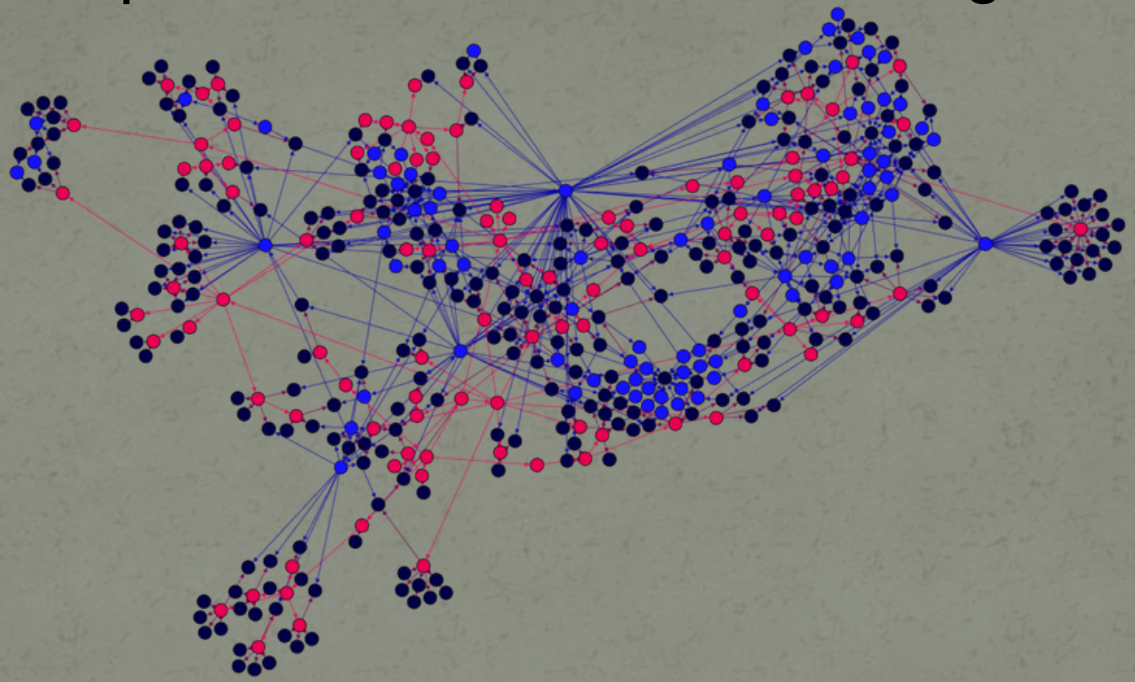
  Worst: $O(n_a)$/iter, Average: $O(1)$/iter

- Timing:

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

Decision making in scheduling

Graph-based decision making unit



- ✓ Ideal information partitioning
- ✓ Only one component for both CF and DF decisions
- ✓ Reach spectrum of insights on topology of the algorithms' precedence

Contents

- Introduction
  - Legacy approach to decision making
  - New approach to decision making 
- Concurrency control: reactive scheduling 
- GaudiHive scalability on close to real workflow topologies 
- Concurrency control: predictive scheduling 
- Generic analysis of speedup constraints 
- Interplay of speedup with algorithm's timing 

Decision making: complexity by the number of entities

	CF decisions	DF decisions
Catalog-based	Worst & Average: $\boldsymbol{O(n_a + n_d)}$	Worst: $\boldsymbol{O(n_a)}$, Average: $\boldsymbol{O(1)}$
Graph-based	Worst: $\boldsymbol{O(n_a)}$, Average: $\boldsymbol{O(1)}$	Worst & Average: $\boldsymbol{O(1)}$

n_a - number of algorithms
n_d - number of decision hubs

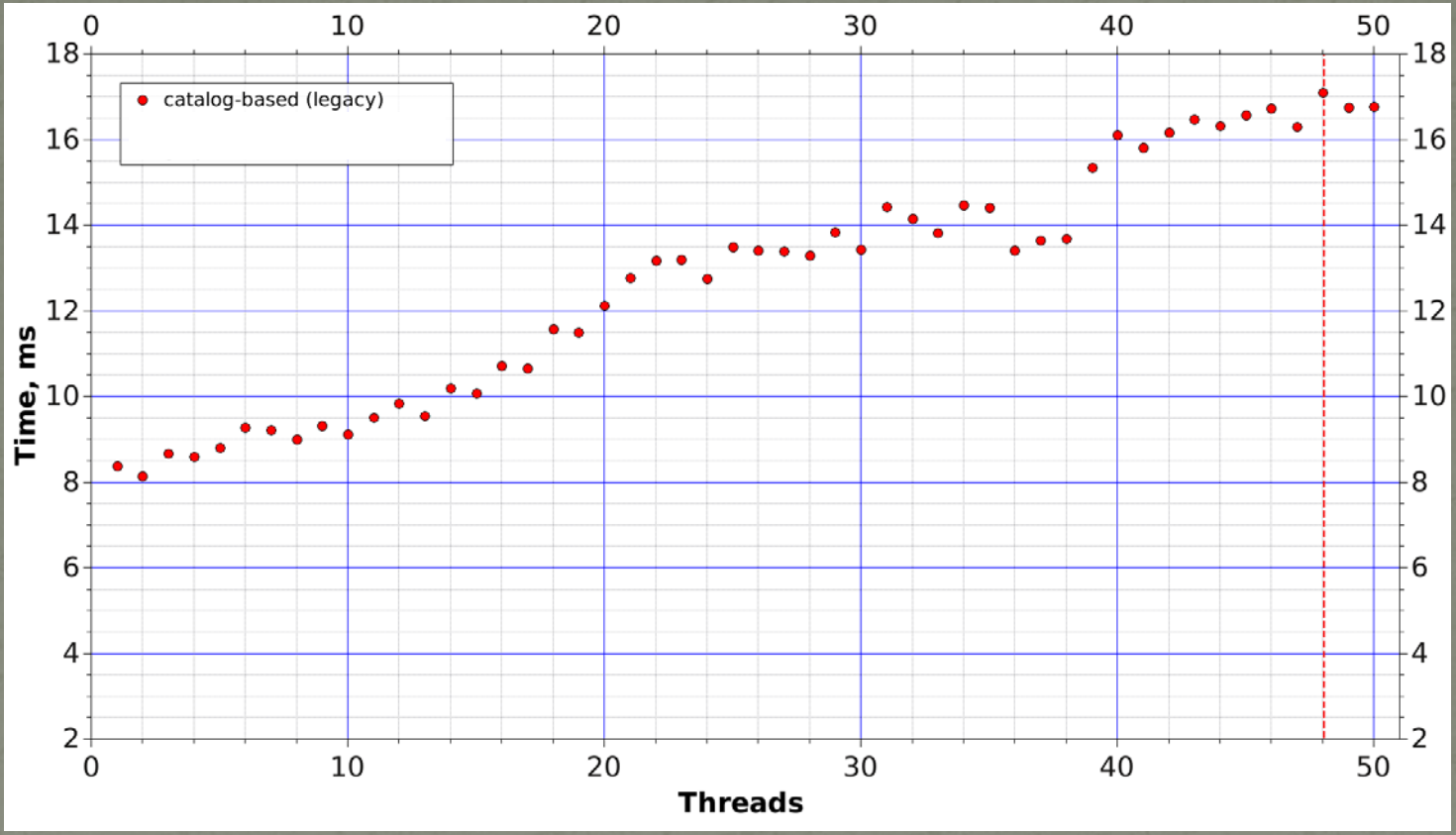
Testbed for benchmarking
(for all subsequent measurements in the talk)

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

Workflow configuration:

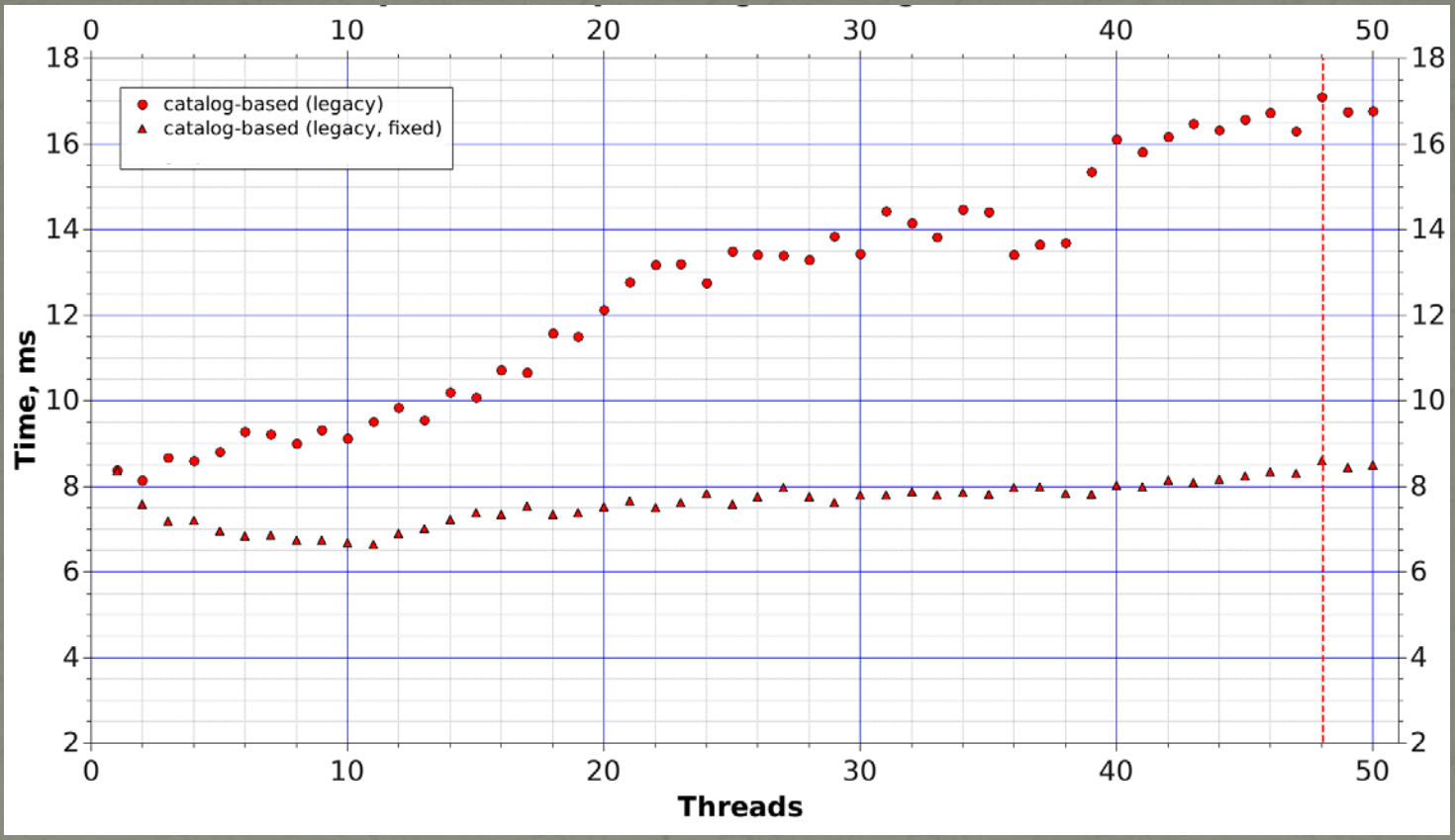
- Precedence graph of close to real size and topology (LHCb Brunel reconstruction case)
- CPUCrunchers as 'algorithms' ('modules' in CMS jargon)
- Real/uniform algorithms' timings

Time of decision making



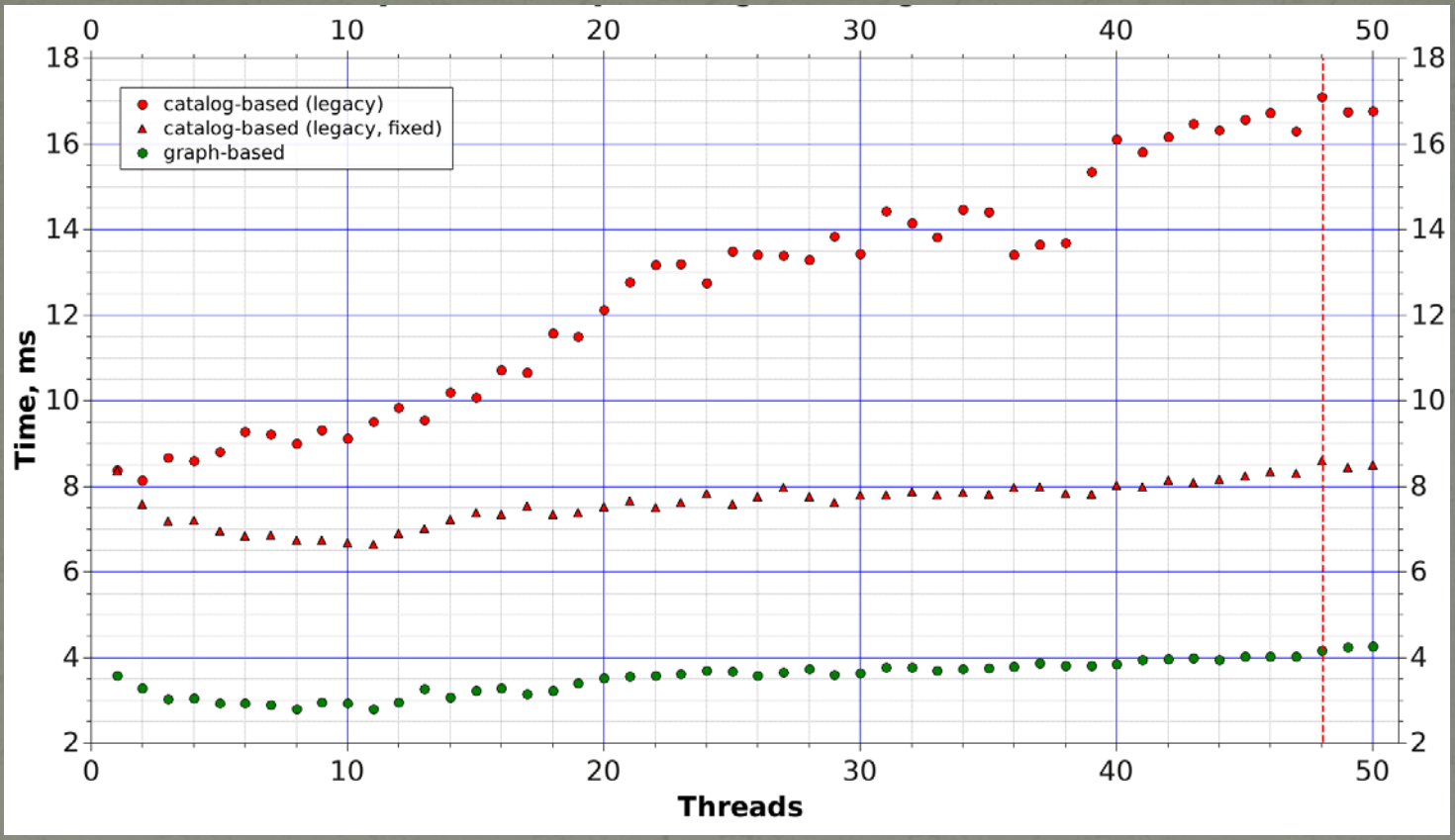
Total time spent to reason about precedence rules per event (spans 263 algorithms)

Time of decision making



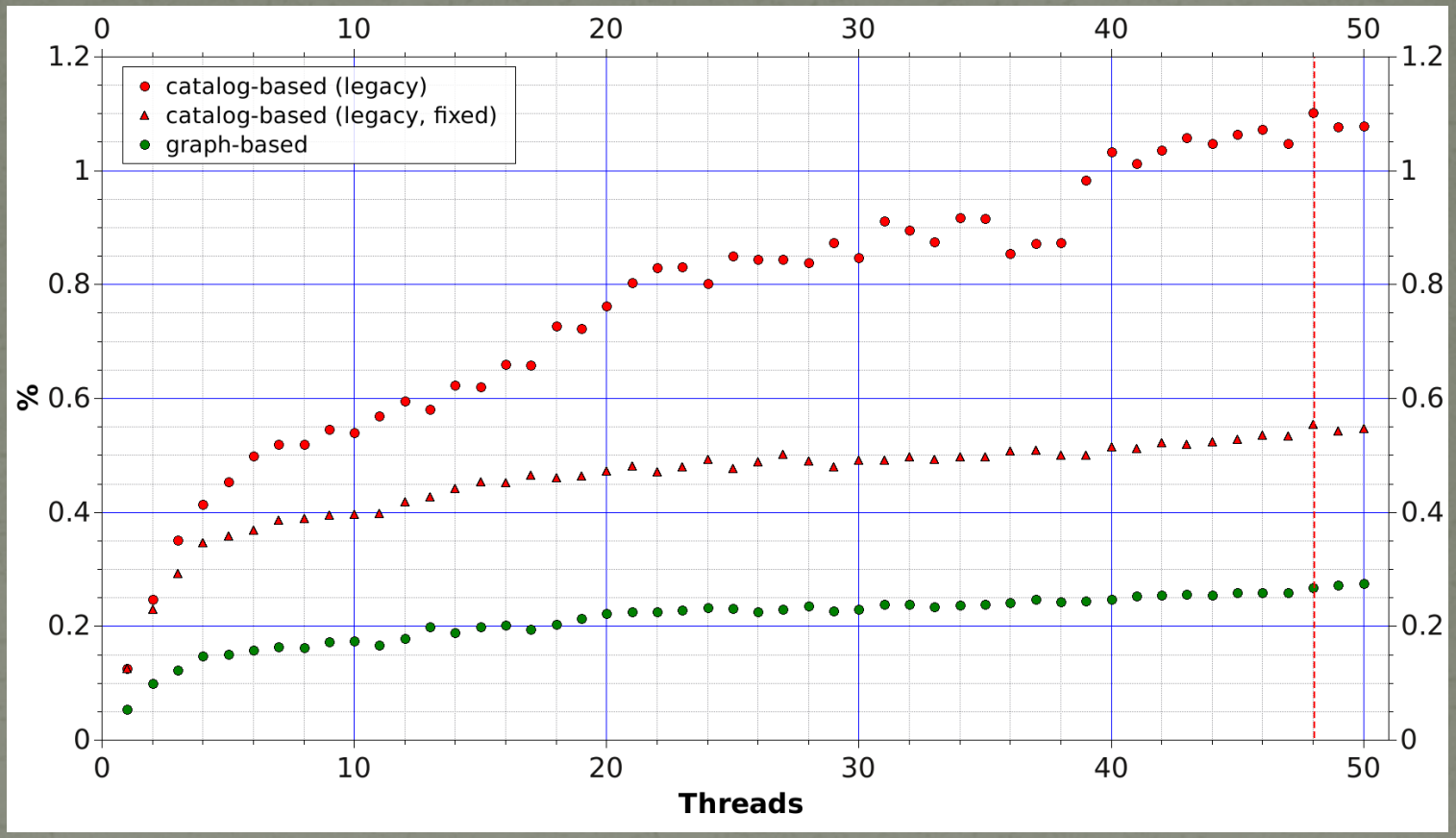
Total time spent to reason about precedence rules per event (spans 263 algorithms)

Time of decision making



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

Ratio of decision making time to event processing time

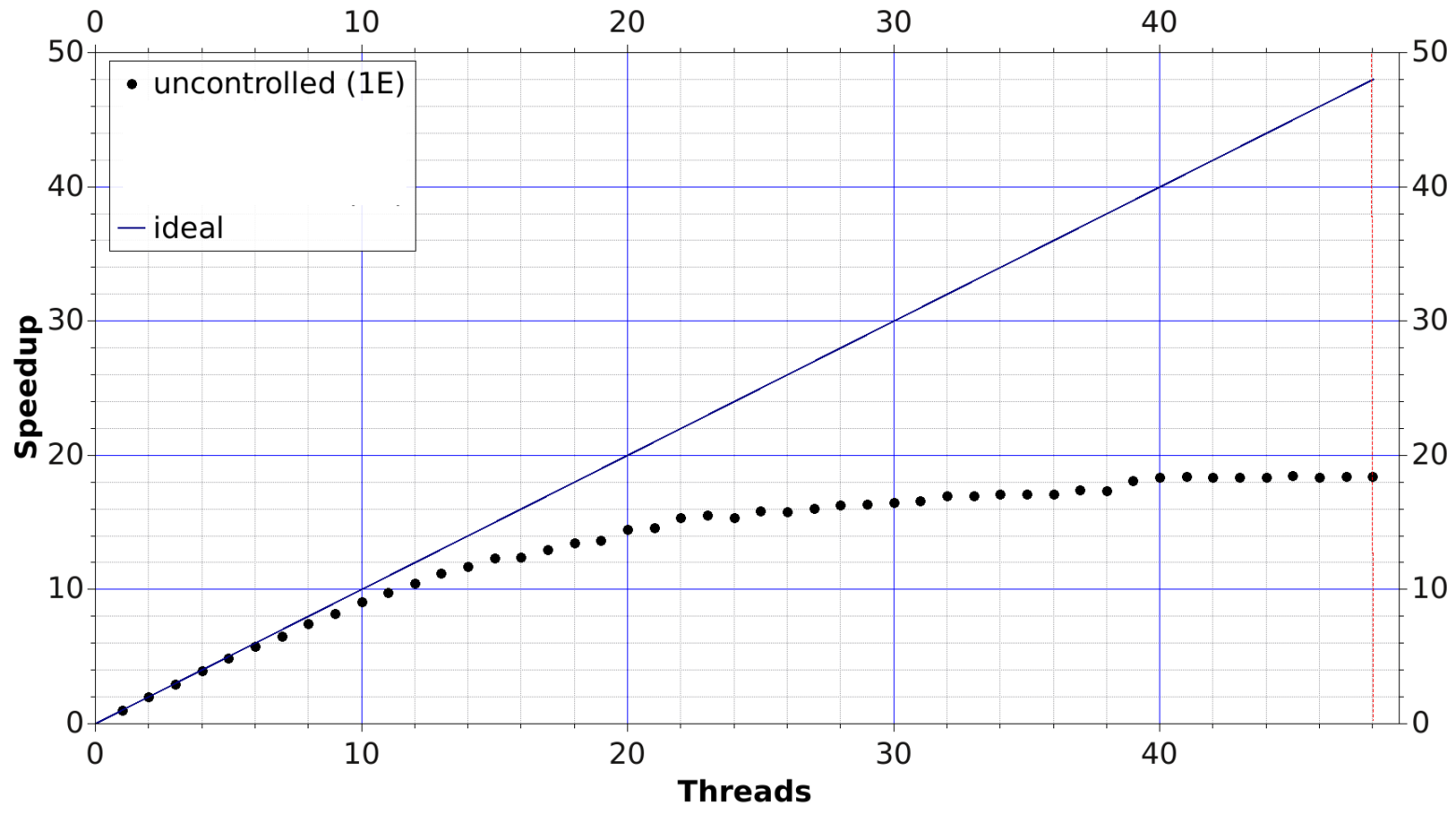


Chosen max. speedup of concurrent event processing is conservative: **4x** !

Contents

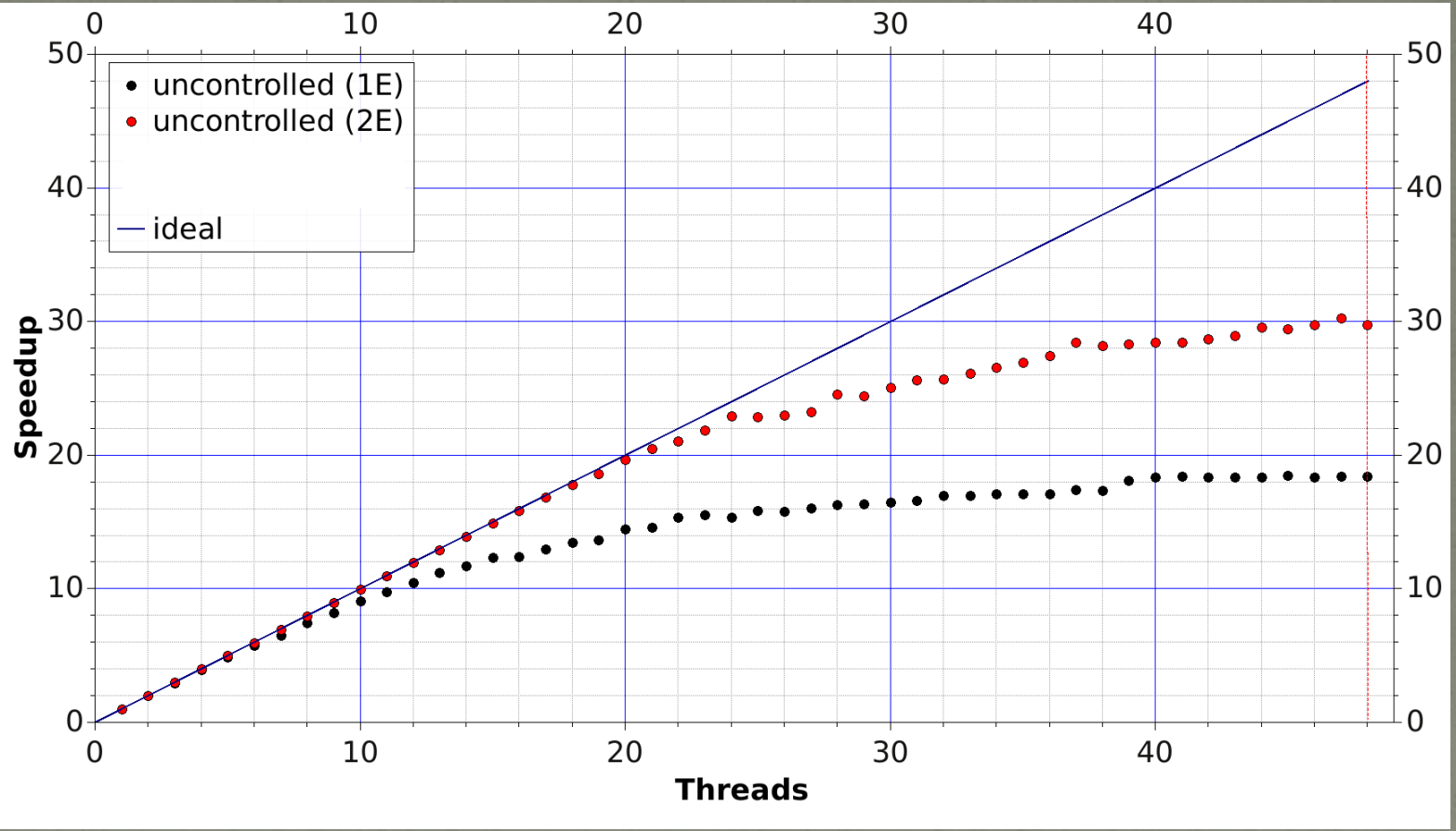
- Introduction
  - Legacy approach to decision making
  - New approach to decision making 
- Concurrency control: reactive scheduling 
- GaudiHive scalability on close to real workflow topologies 
- Concurrency control: predictive scheduling 
- Generic analysis of speedup constraints 
- Interplay of speedup with algorithm's timing 

Speedup saturation: uniform timing



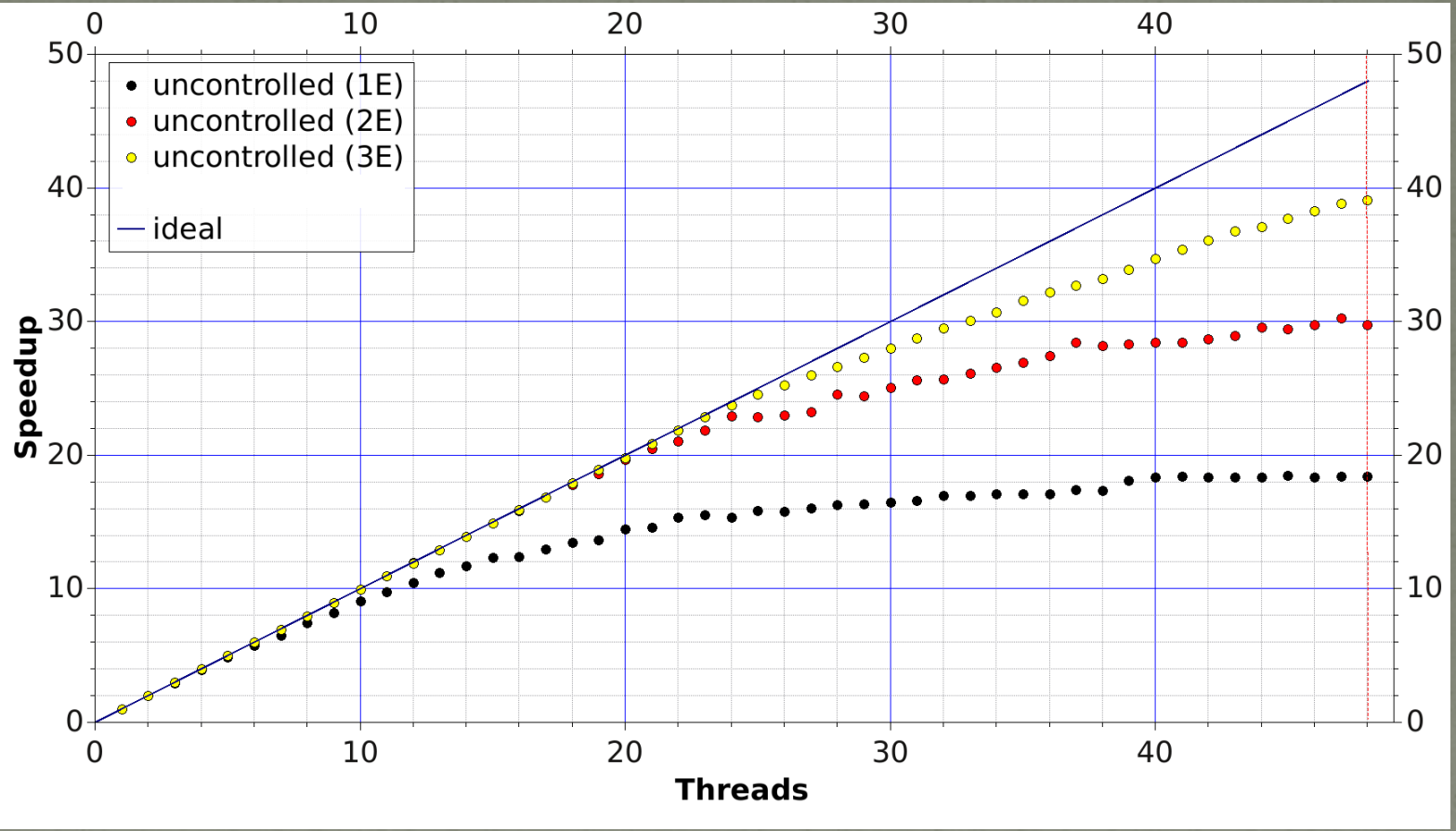
Intra-event mode only (algorithm timing ~10ms)

Speedup saturation: uniform timing



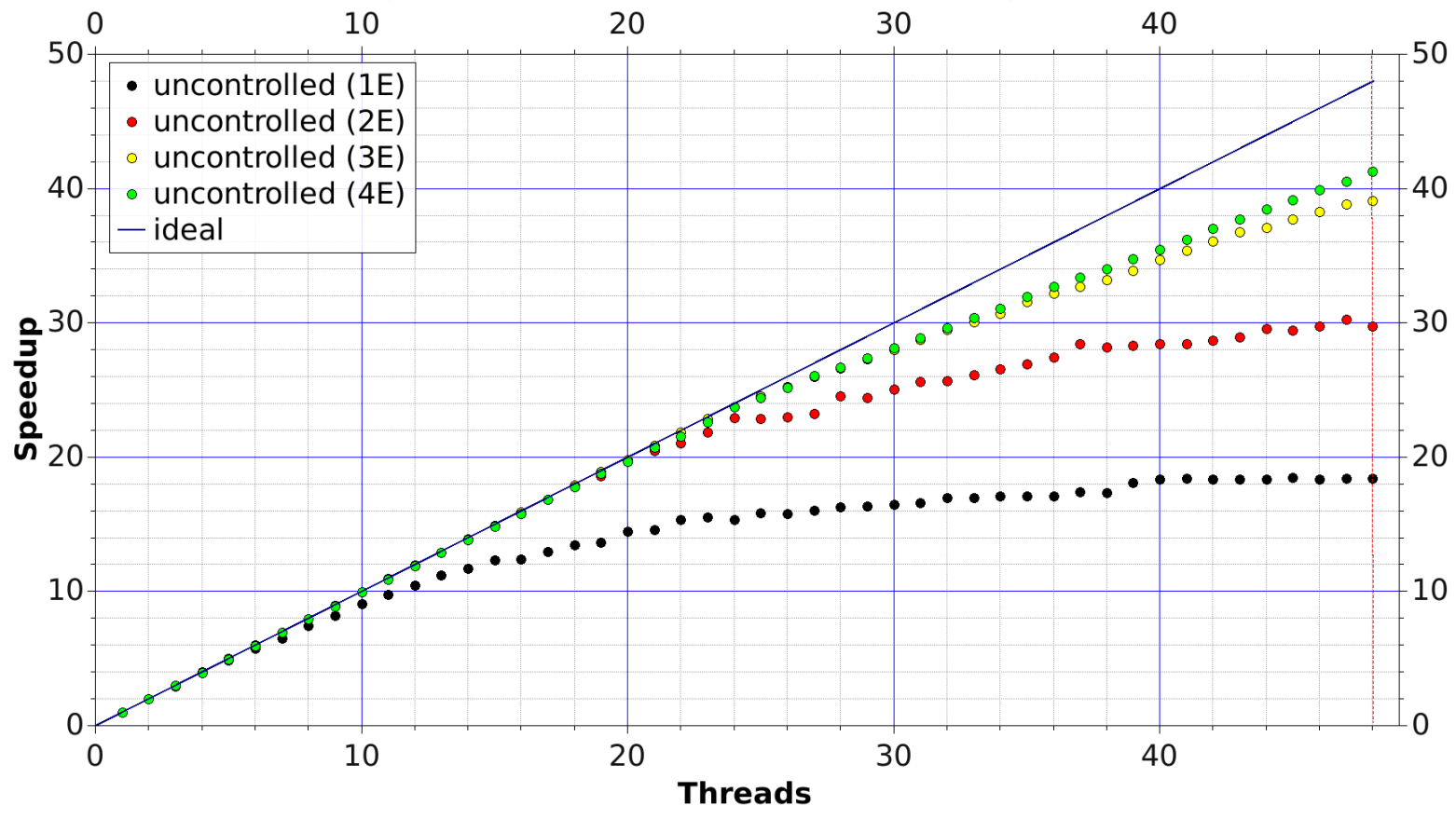
Intra-event + inter-event mode (algorithm timing ~10ms)

Speedup saturation: uniform timing



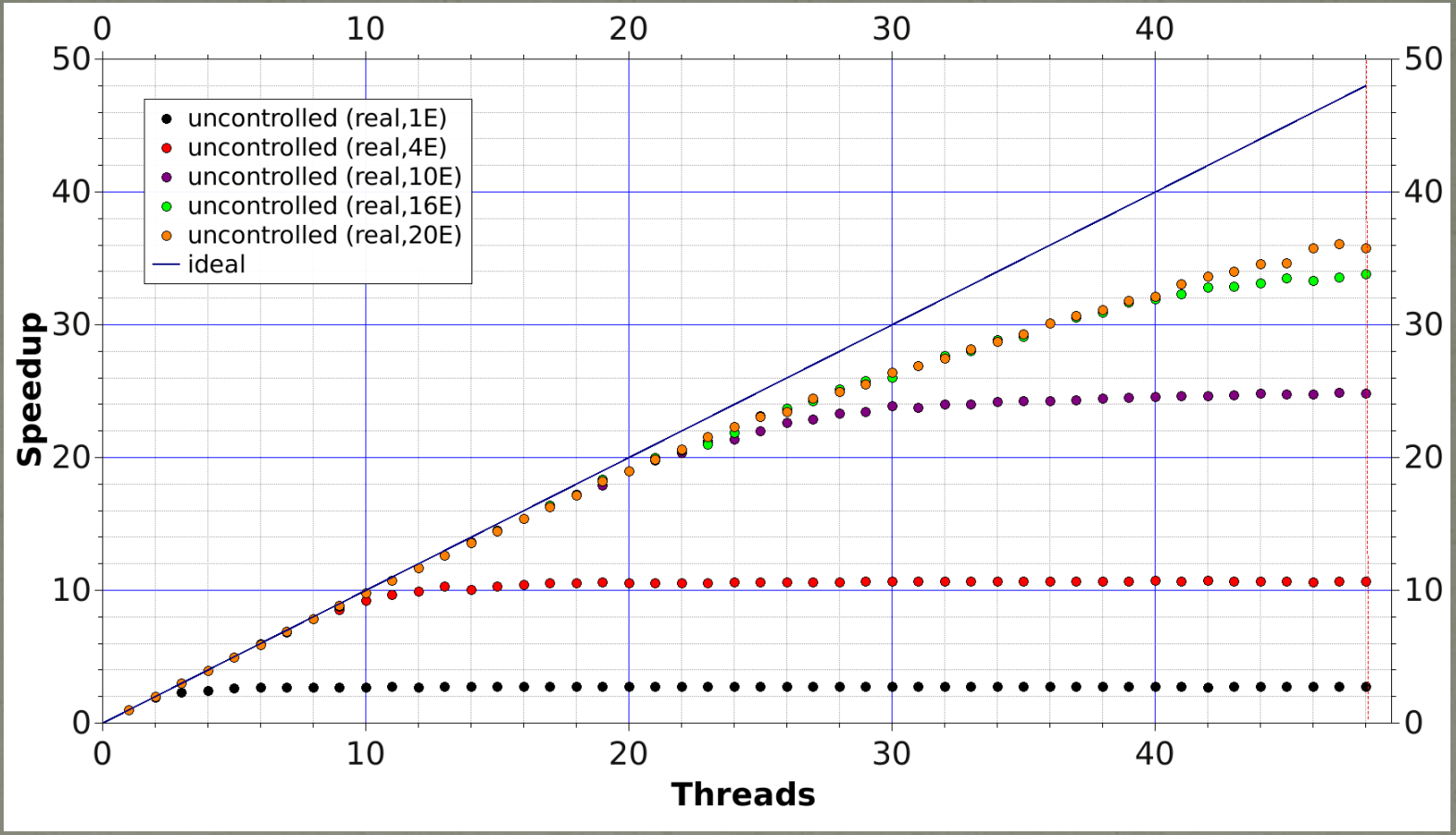
Intra-event + inter-event mode (algorithm timing ~10ms)

Speedup saturation: uniform timing



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

Speedup saturation: **real** timing



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

Measures to consider

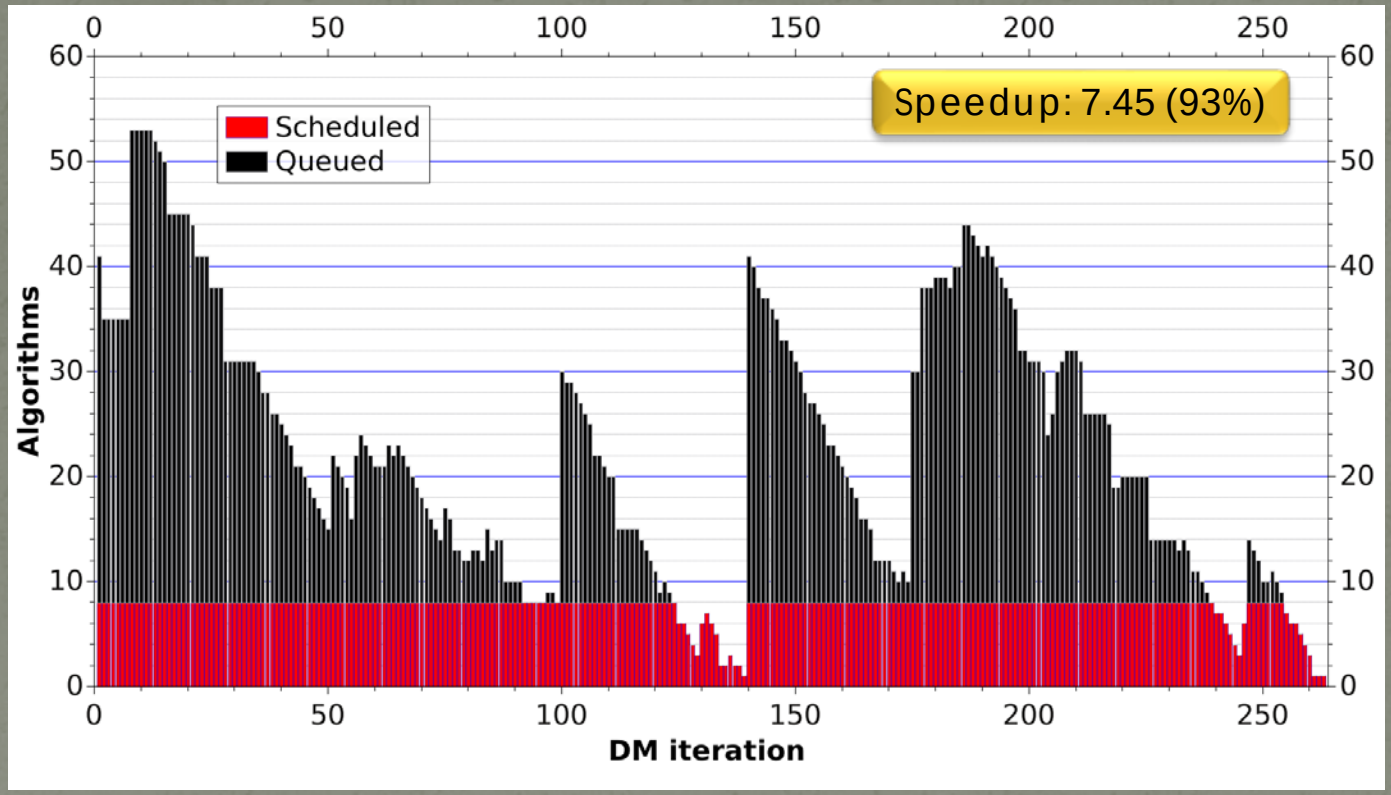
- Further reduce concurrency overhead
  (not discussed in this talk)

Measures to consider

- Further reduce concurrency overhead
  (not discussed in this talk)

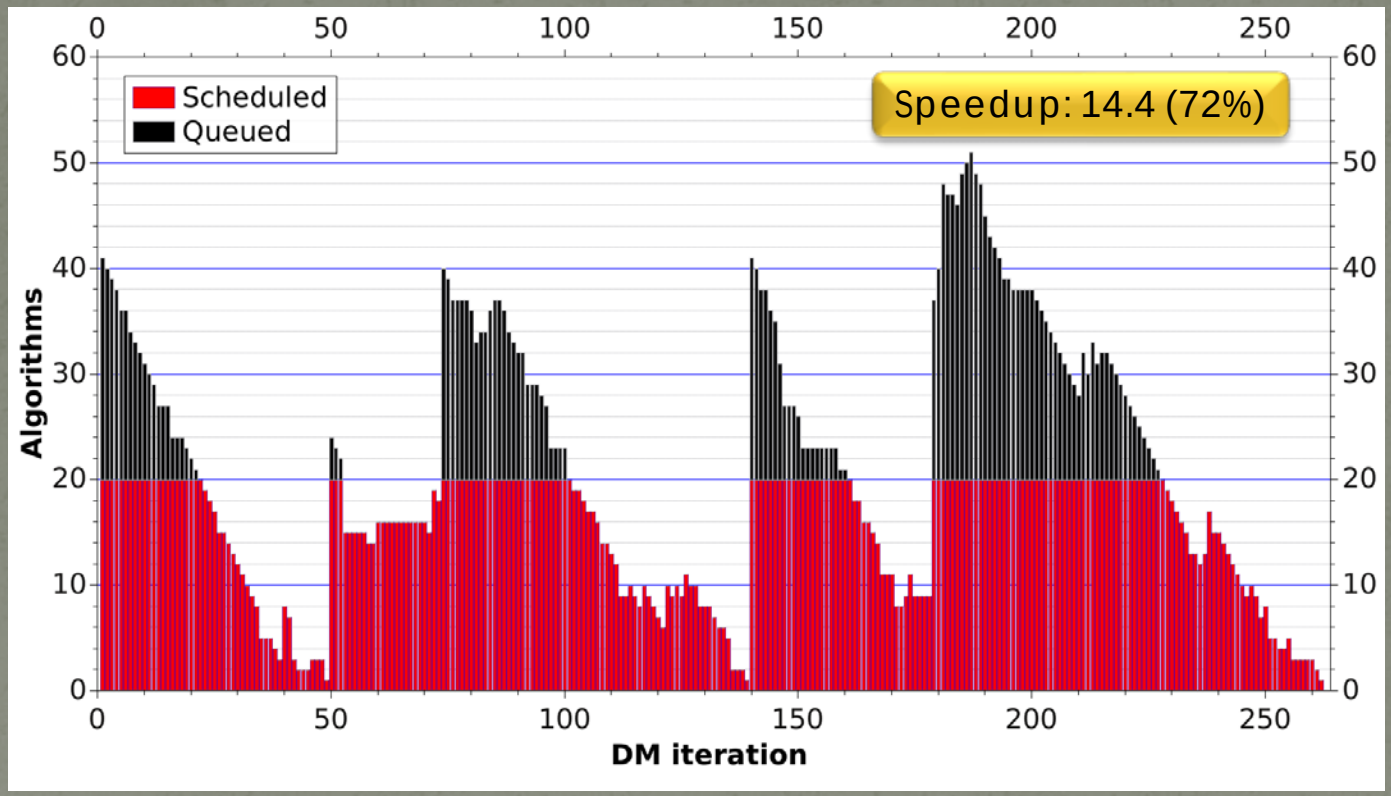
- Improve intra-event concurrency
  - its low level pushes to overuse the inter-event concurrency
  - better use of data locality (?)

Intra-event concurrency dynamics



Reactive scheduling only (**8 threads**, 263 algorithms)

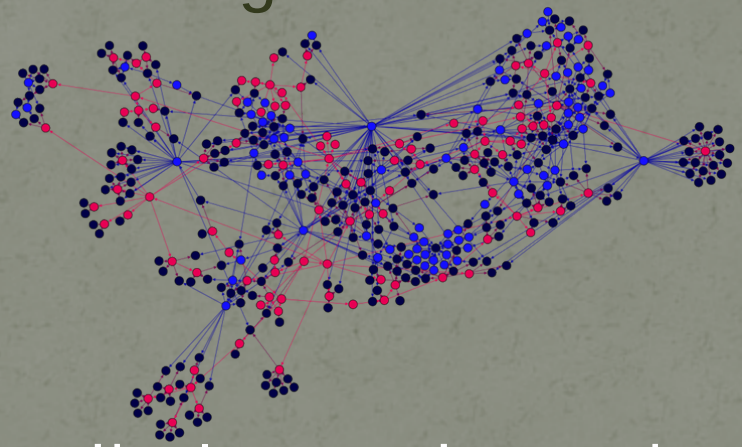
Intra-event concurrency dynamics



Reactive scheduling only (**20 threads**, 263 algorithms)

Harmful degree of freedom…

- Typical precedence graphs (in LHCb) are significantly heterogeneous



- Concurrency disclosure dynamics dependent on execution path
  - uncontrolled in GaudiHive reactive scheduling

Contents

- Introduction
  - Legacy approach to decision making
  - New approach to decision making 
- Concurrency control: reactive scheduling 
- GaudiHive scalability on close to real workflow topologies 
- Concurrency control: predictive scheduling 
- Generic analysis of speedup constraints 
- Interplay of speedup with algorithm's timing 

Predictive scheduling in GaudiHive

What:

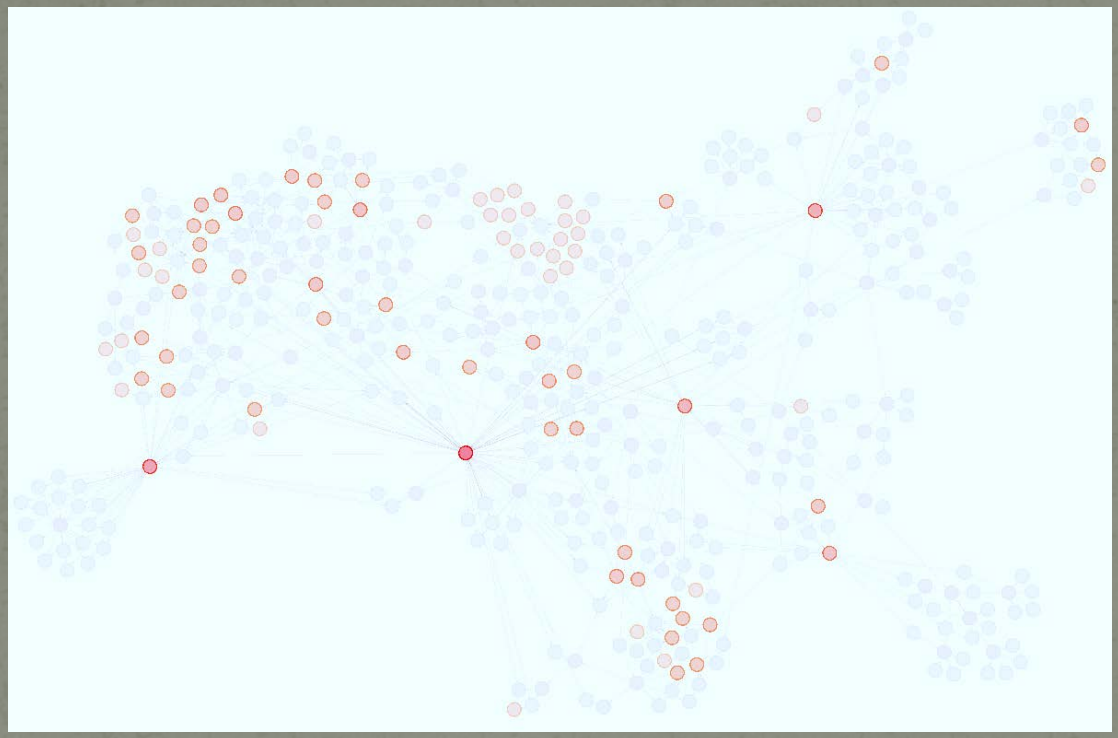
- maximize concurrency disclosure dynamics
  - or at least create facilitating pressure towards it

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

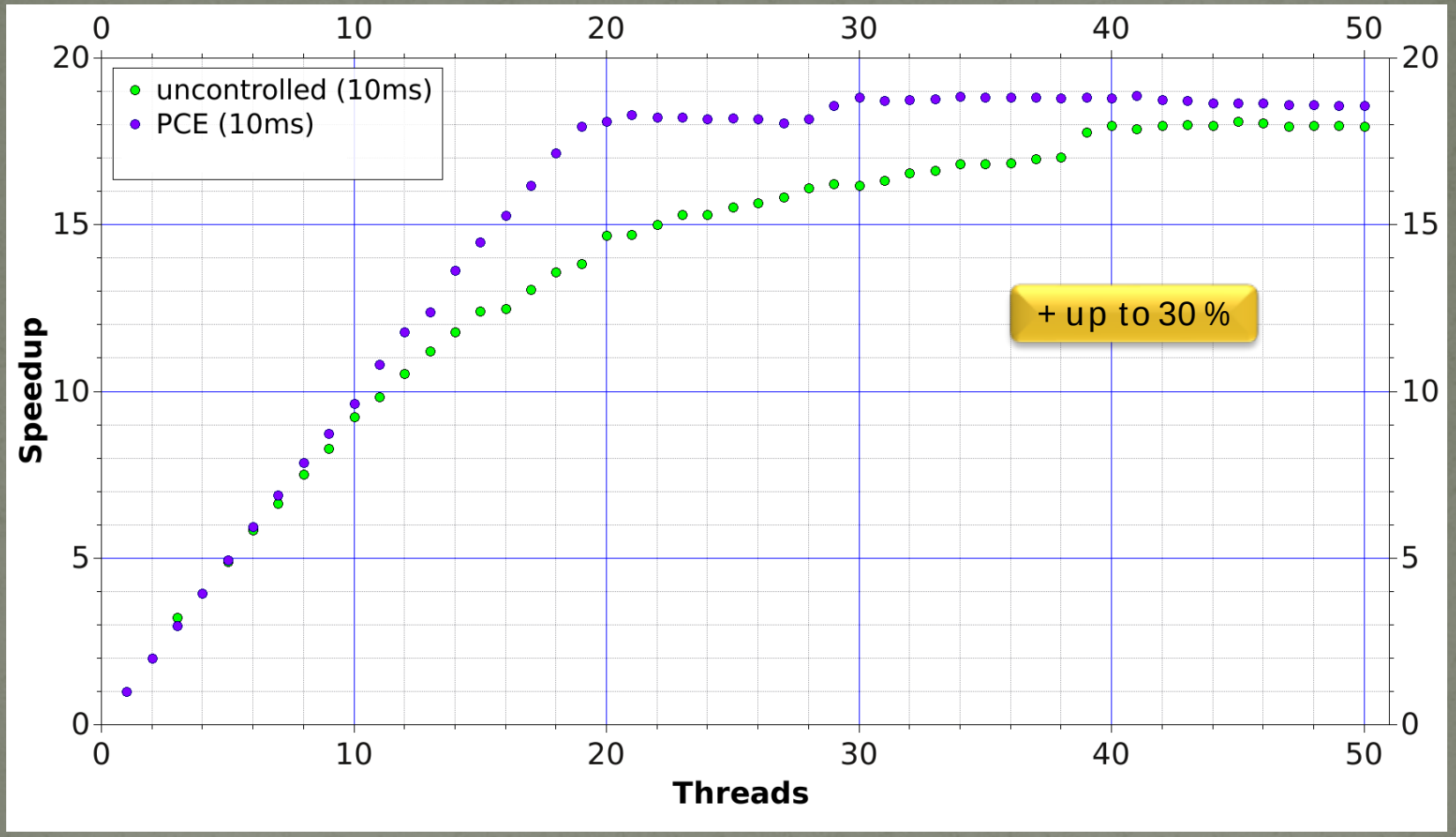
Asymmetry of products consumption

- rank algorithm by its products consumption extent



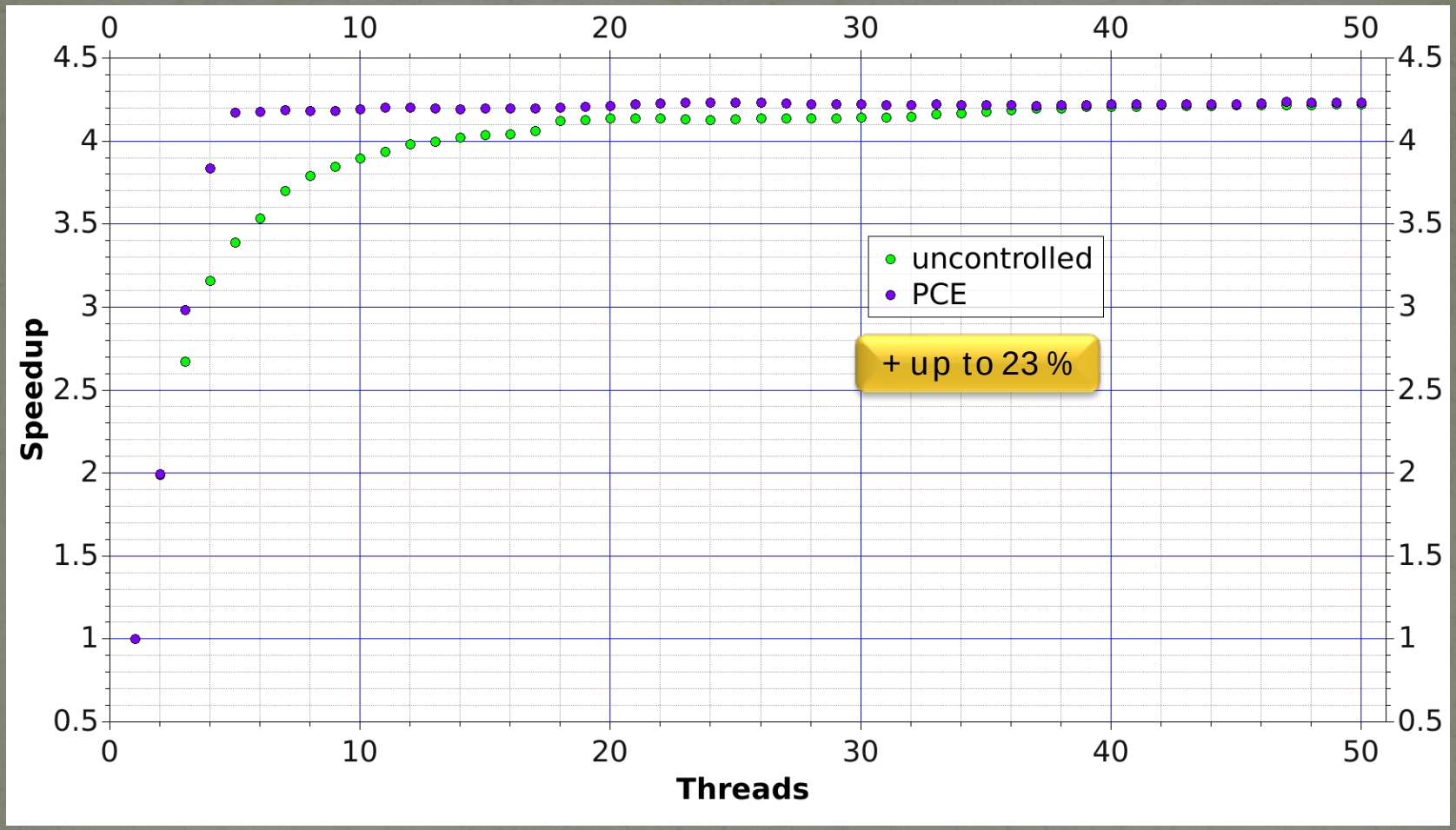
Precedence graph with all, but data nodes, greyed out. Color intensity of a data node represent number of its consumers

Predictive scheduling: products consumption extent (PCE)



Uniform algorithm timing (~10ms)

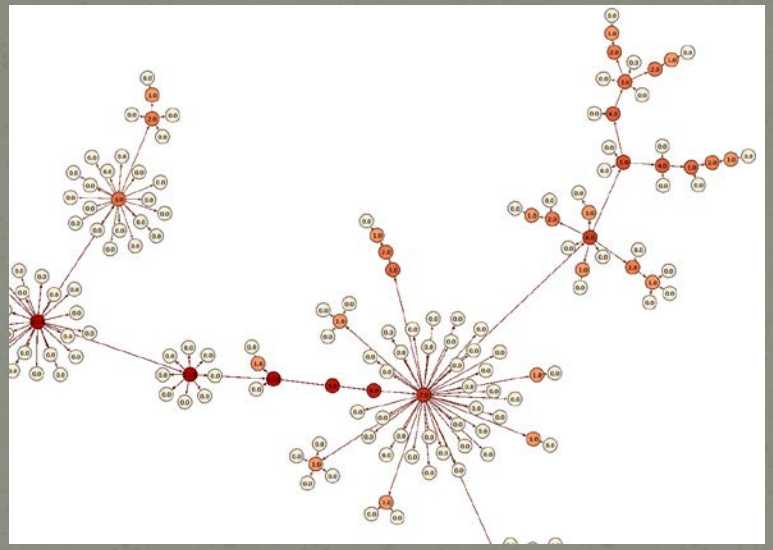
Predictive scheduling: products consumption extent (PCE)



Real algorithm timing

Predictive scheduling: data realm eccentricity (DRE)

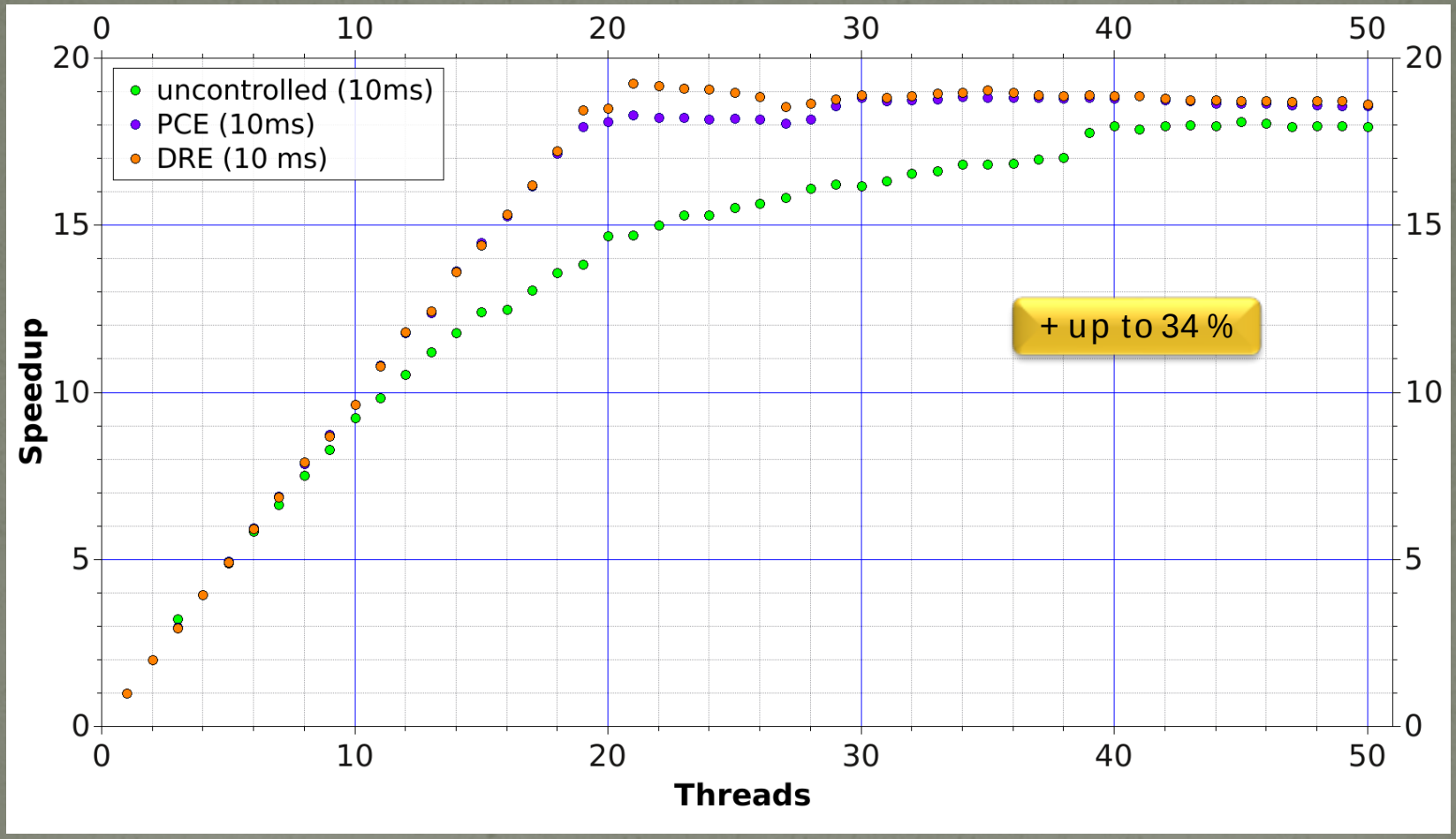
- rank algorithm by its eccentricity in data realm



Color intensity represents eccentricity-based rank

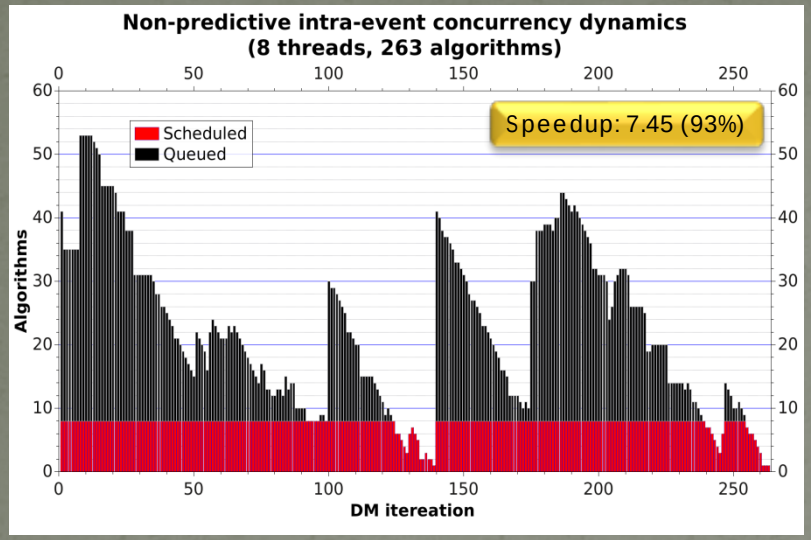
- implements critical path lookup technique in case of uniform algorithm timings
- note: not only graph diameter is tracked, but also all other sub-critical paths.

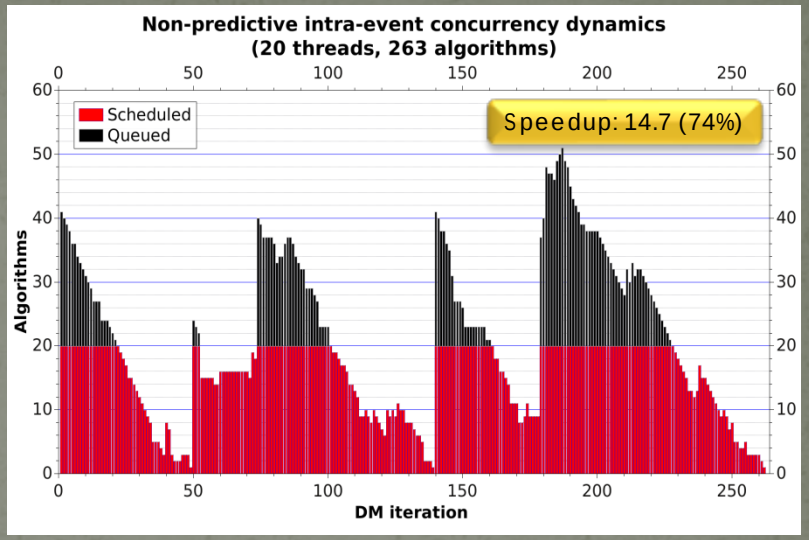
Predictive scheduling: data realm eccentricity (DRE)



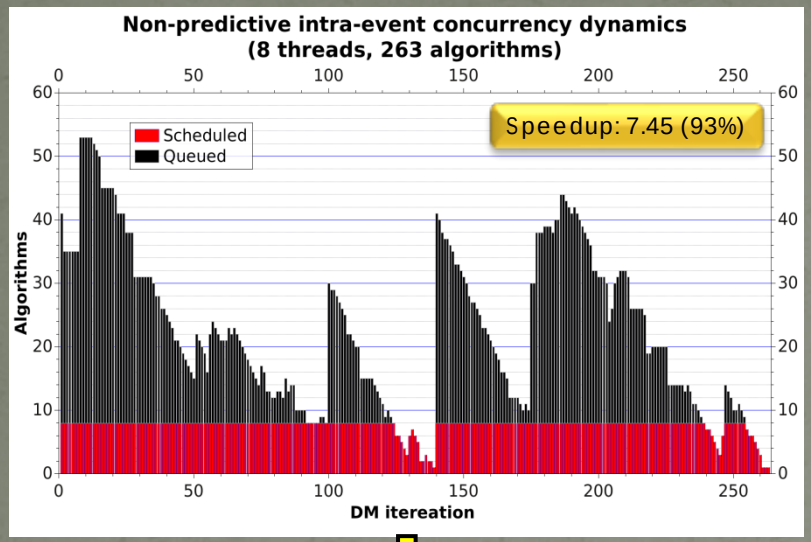
Uniform algorithm timing (~10ms)

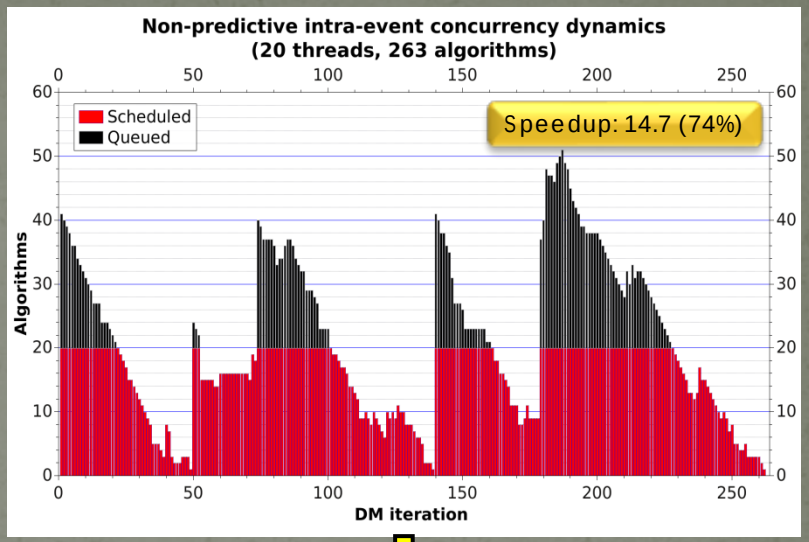
Predictive scheduling: DRE mode

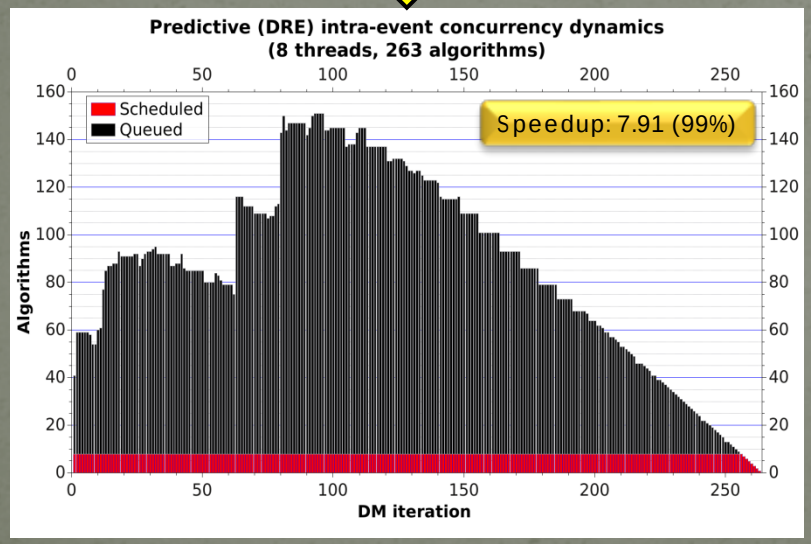


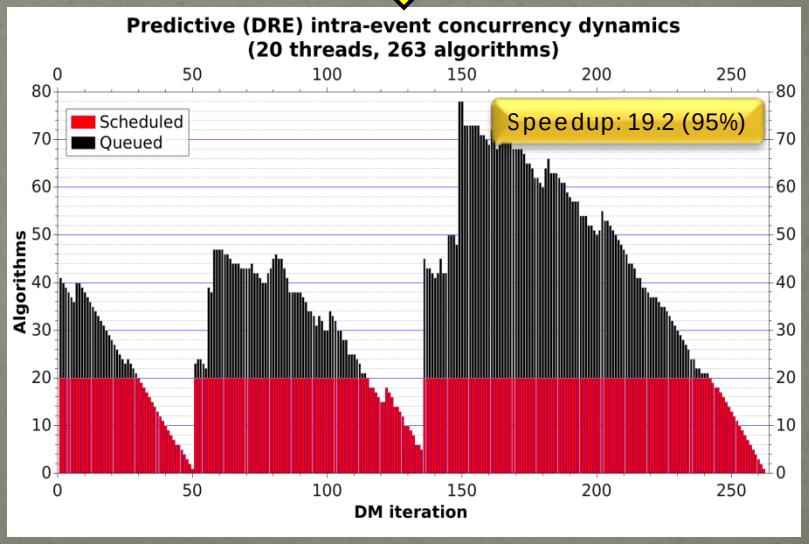


Predictive scheduling: DRE mode

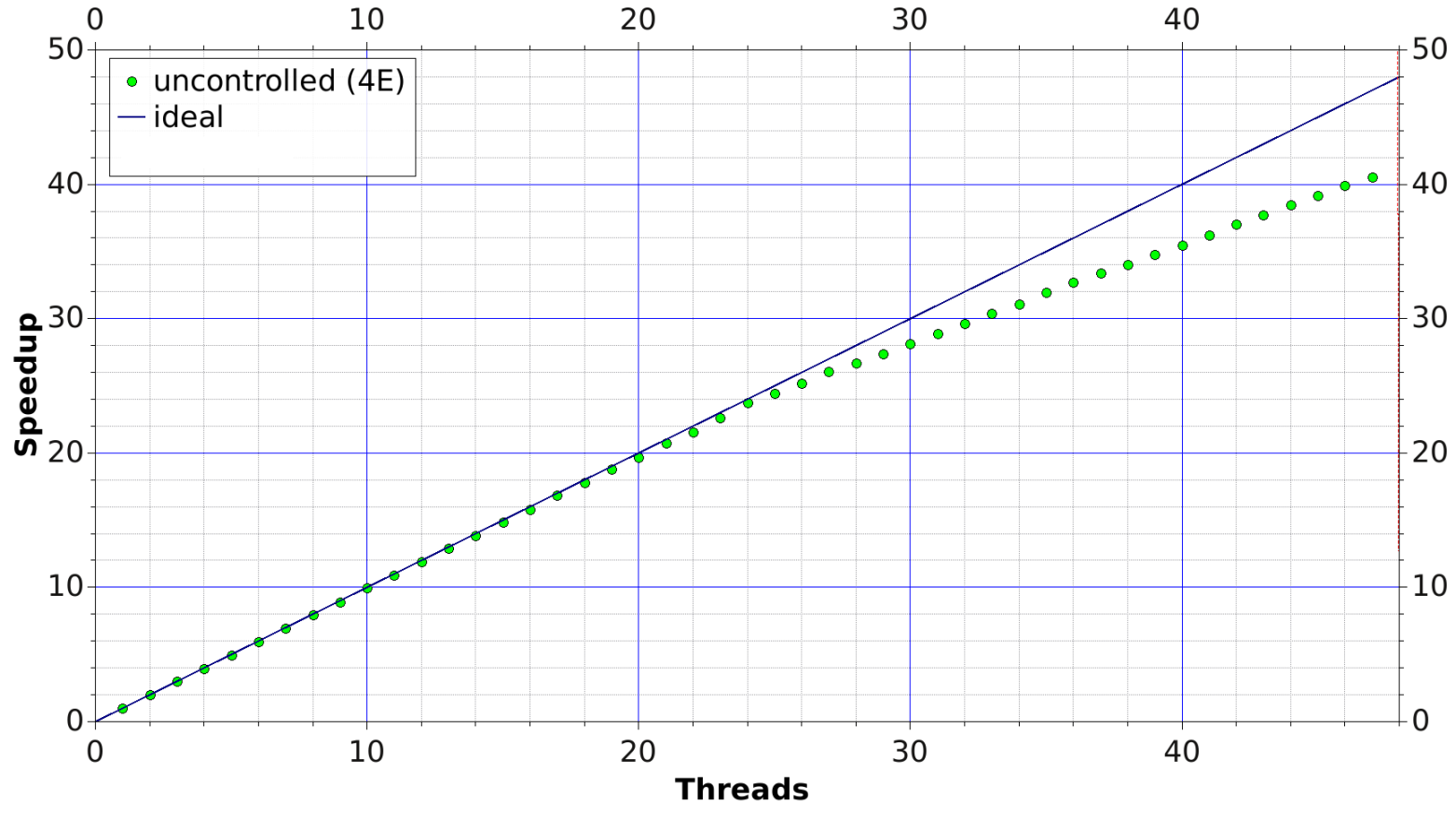






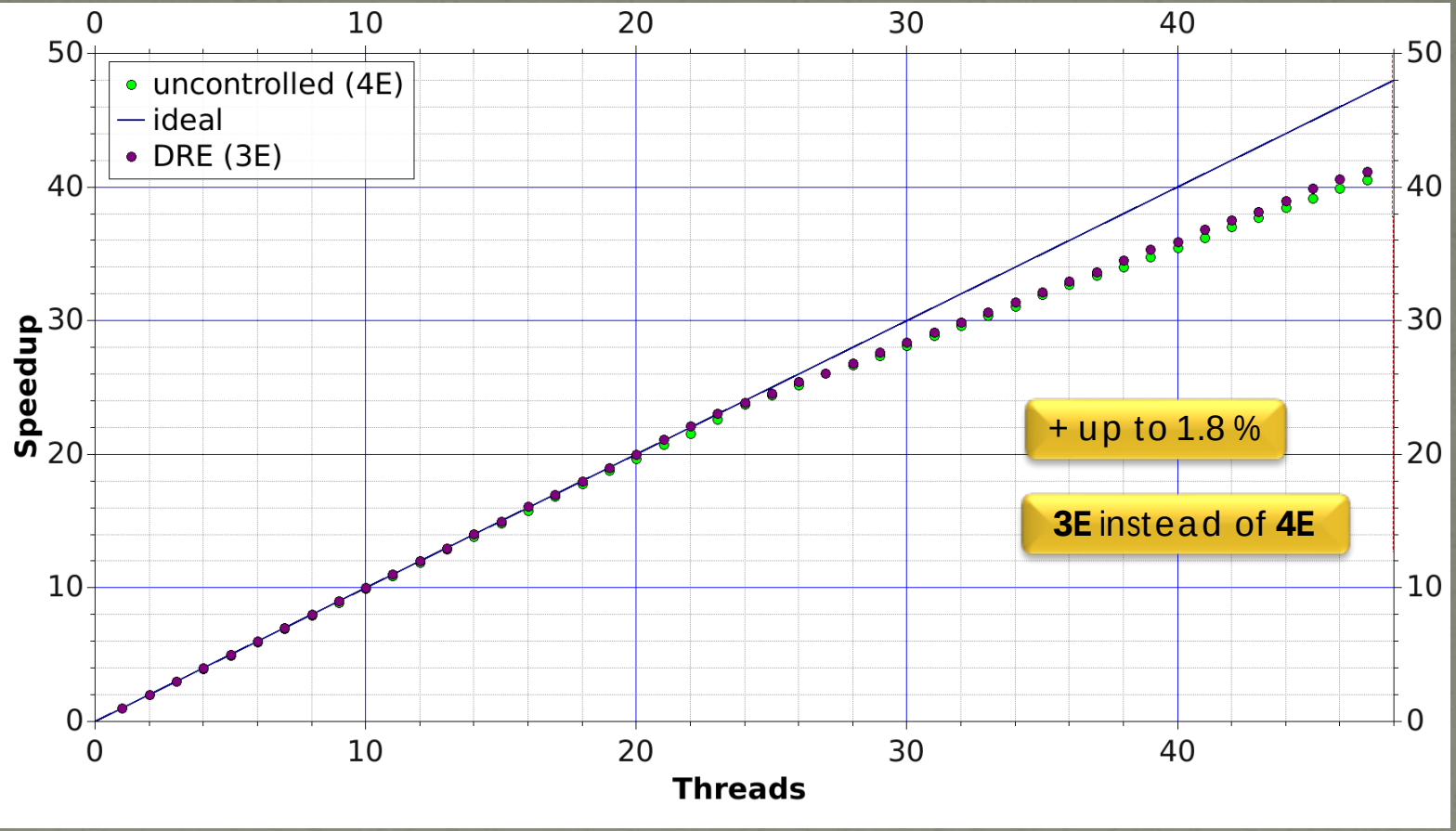


Saturated speedup, boosted by predictive scheduling



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

Saturated speedup, boosted by predictive scheduling



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

Considered prediction strategies

- Product consumption extent (PCE)
  - accumulates breadth-first consumption up to 1 level
- Data realm eccentricity (DRE)
  - simulates critical and sub-critical paths lookup

Implemented, but not discussed in this talk:

- Algorithm's cumulative out-degree (ACOD)
  - extends of the PCE mode to n-level
- Algorithm eccentricity on materialized views (AE)
  - learns materialized views of precedence graphs and detects (sub)critical paths at runtime

Contents

- Introduction
  - Legacy approach to decision making
  - New approach to decision making 

- Concurrency control: reactive scheduling 

- GaudiHive scalability on close to real workflow topologies 

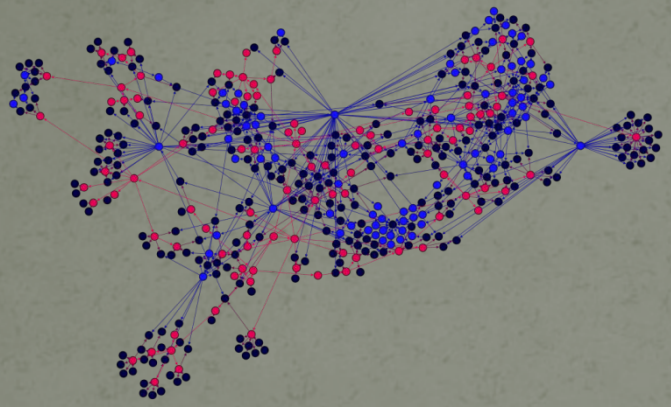
- Concurrency control: predictive scheduling 

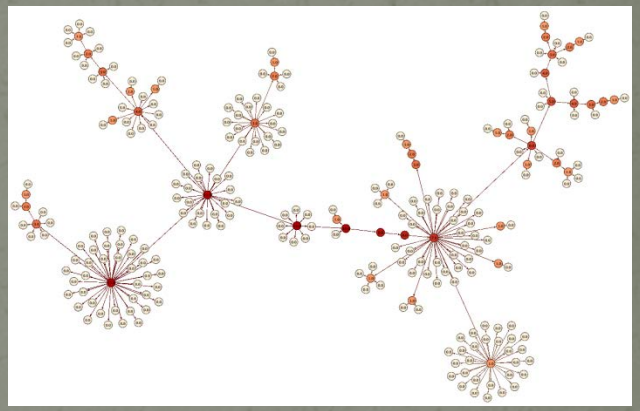
- Generic analysis of speedup constraints 

- Interplay of speedup with algorithm's timing 

Generic constraints analysis

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





- provides, for a given event and comp. resources:
  - critical and sub-critical paths visualization
  - theoretical intra-event speedup limit (⬆ 20)
  - expertize on how to increase the speedup

Contents

- Introduction
  - Legacy approach to decision making
  - New approach to decision making 

- Concurrency control: reactive scheduling 

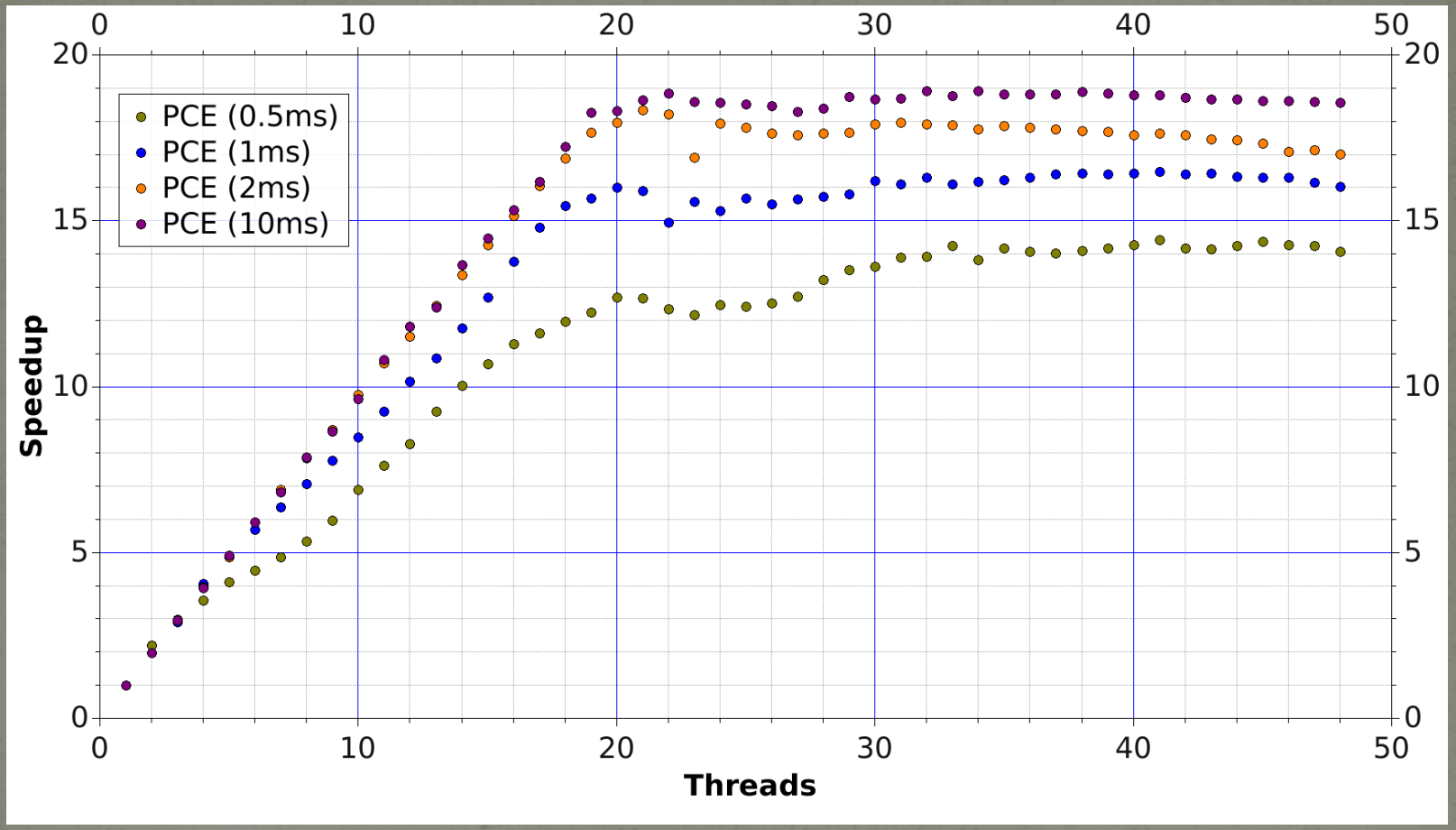
- GaudiHive scalability on close to real workflow topologies 

- Concurrency control: predictive scheduling 

- Generic analysis of speedup constraints 

- Interplay of speedup with algorithm's timing 

Algorithm timing granularity and speedup



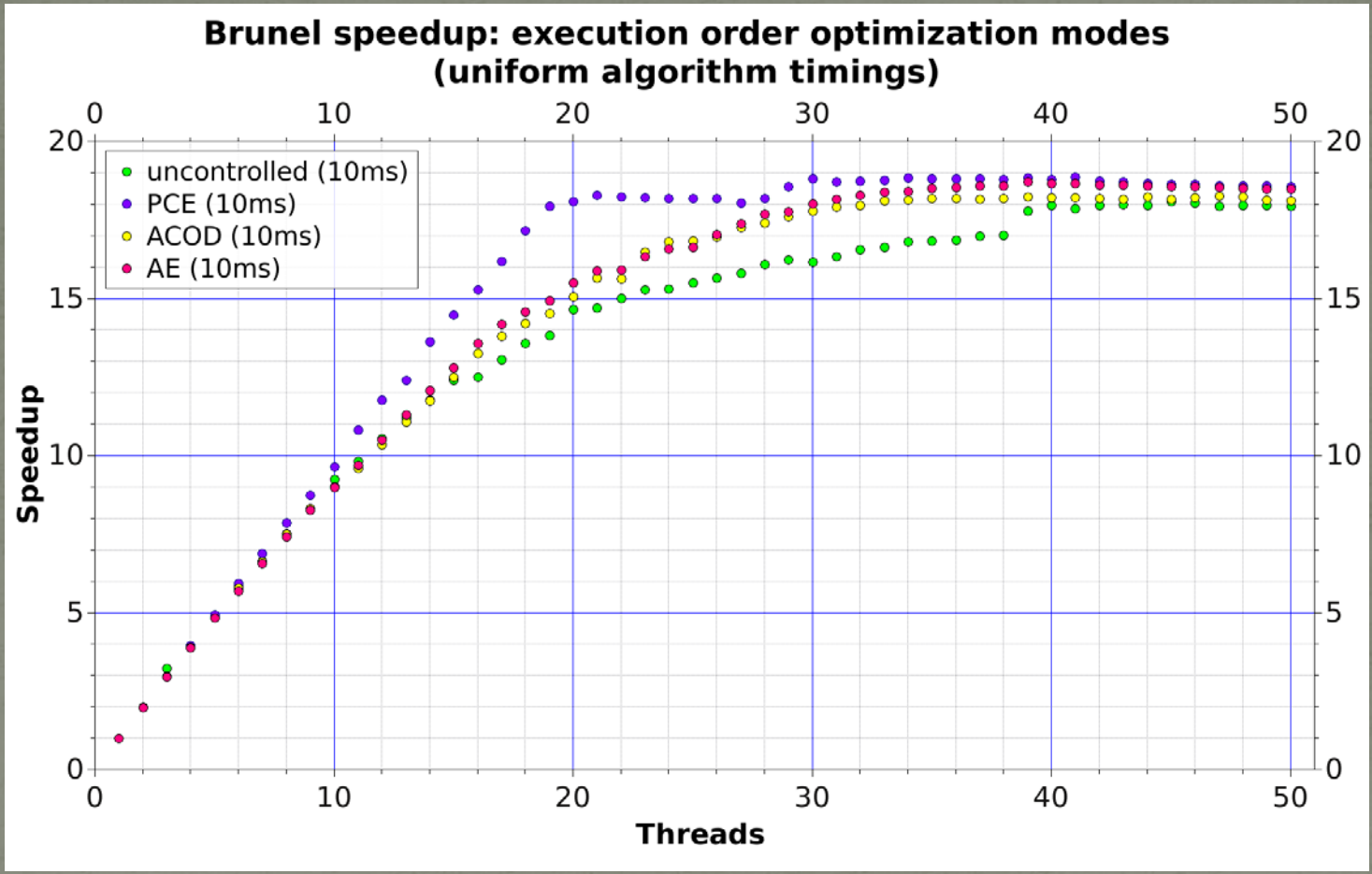
Intra-event concurrency only (uniform algorithm timing)

Conclusion

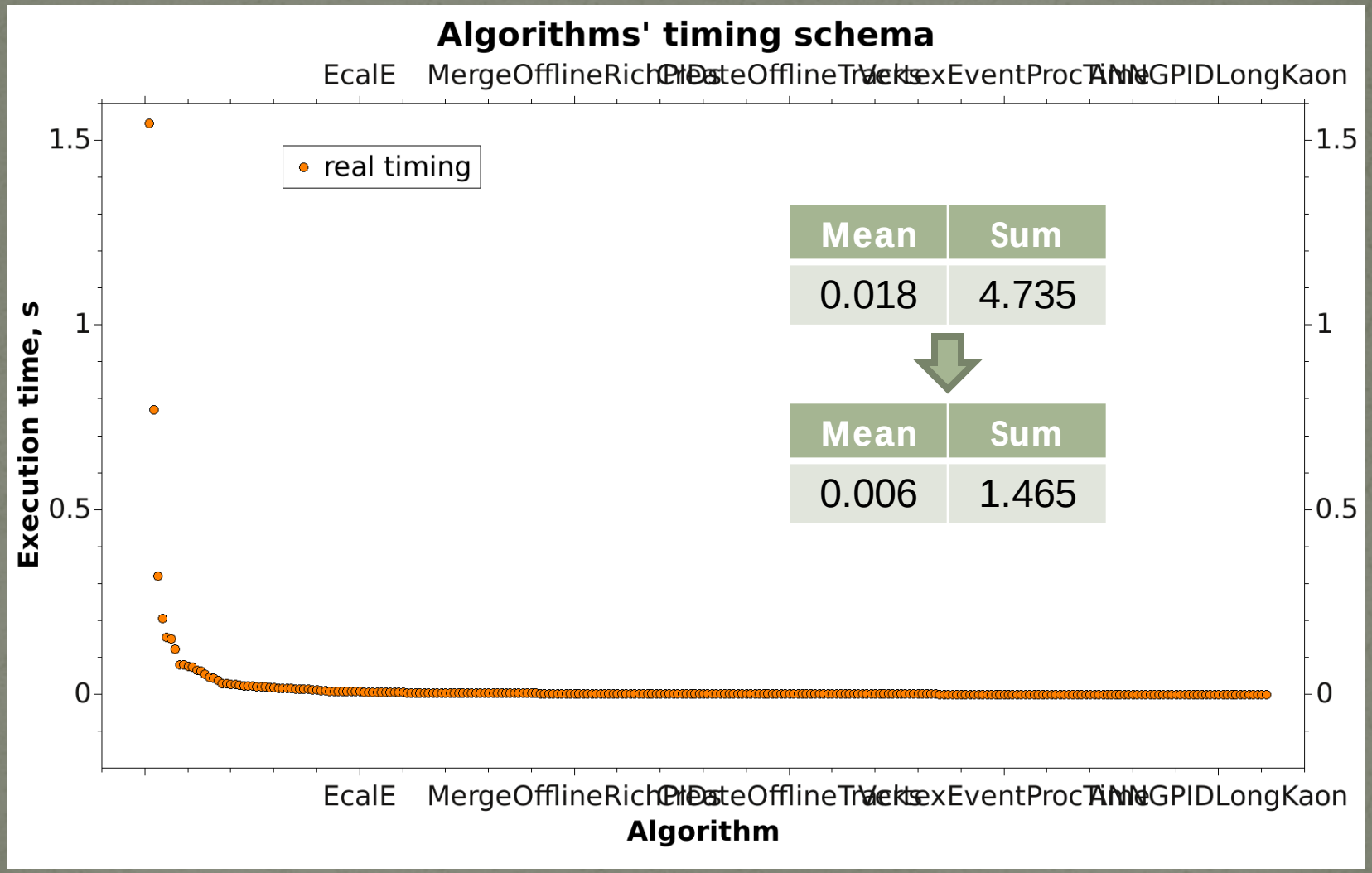
- Graph-based scheduler implemented:
  - Revised reactive scheduling
    - decision making time is 2x faster;
  - Added predictive scheduling
    - some ranking techniques found to be effective (with up to 30% improvement in intra-event speedup)
    - more control on the balance between intra- and inter-event concurrent processing
- All measurements were obtained on close to real precedence graphs (LHCb Brunel reconstruction workflow)

Spare slides

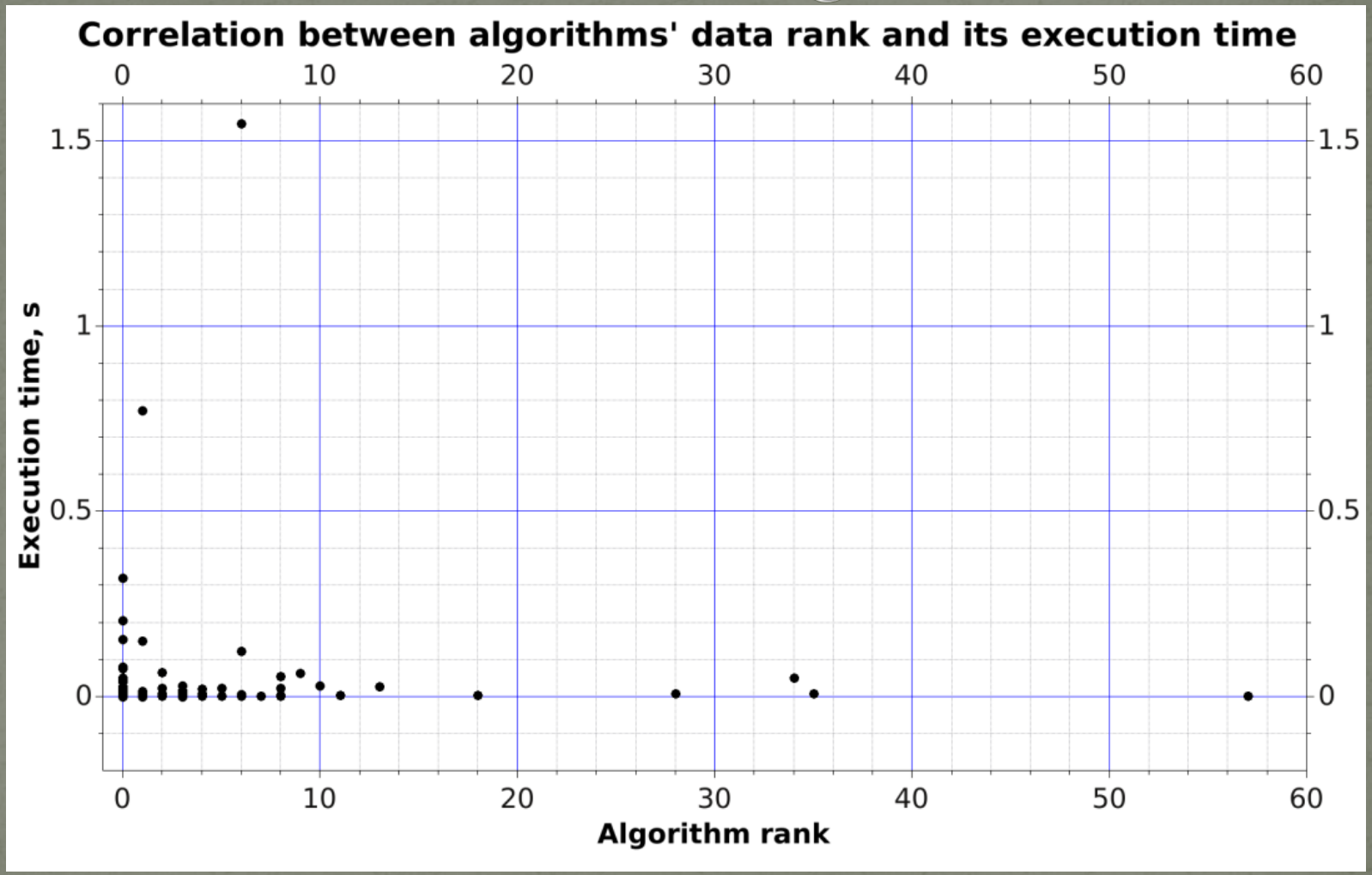
Brunel speedup: ACOD and AE modes



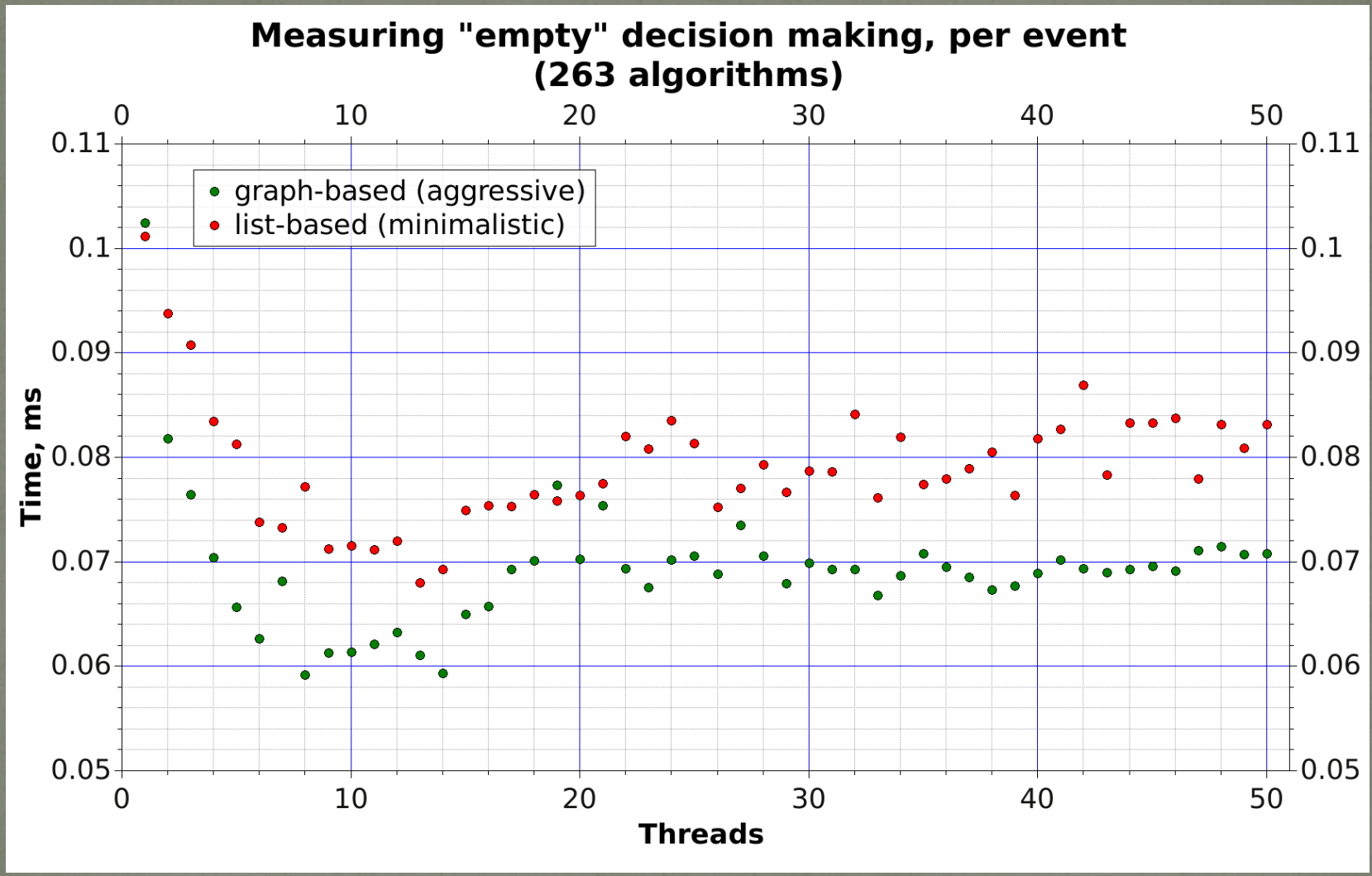
Algorithms' timing distribution



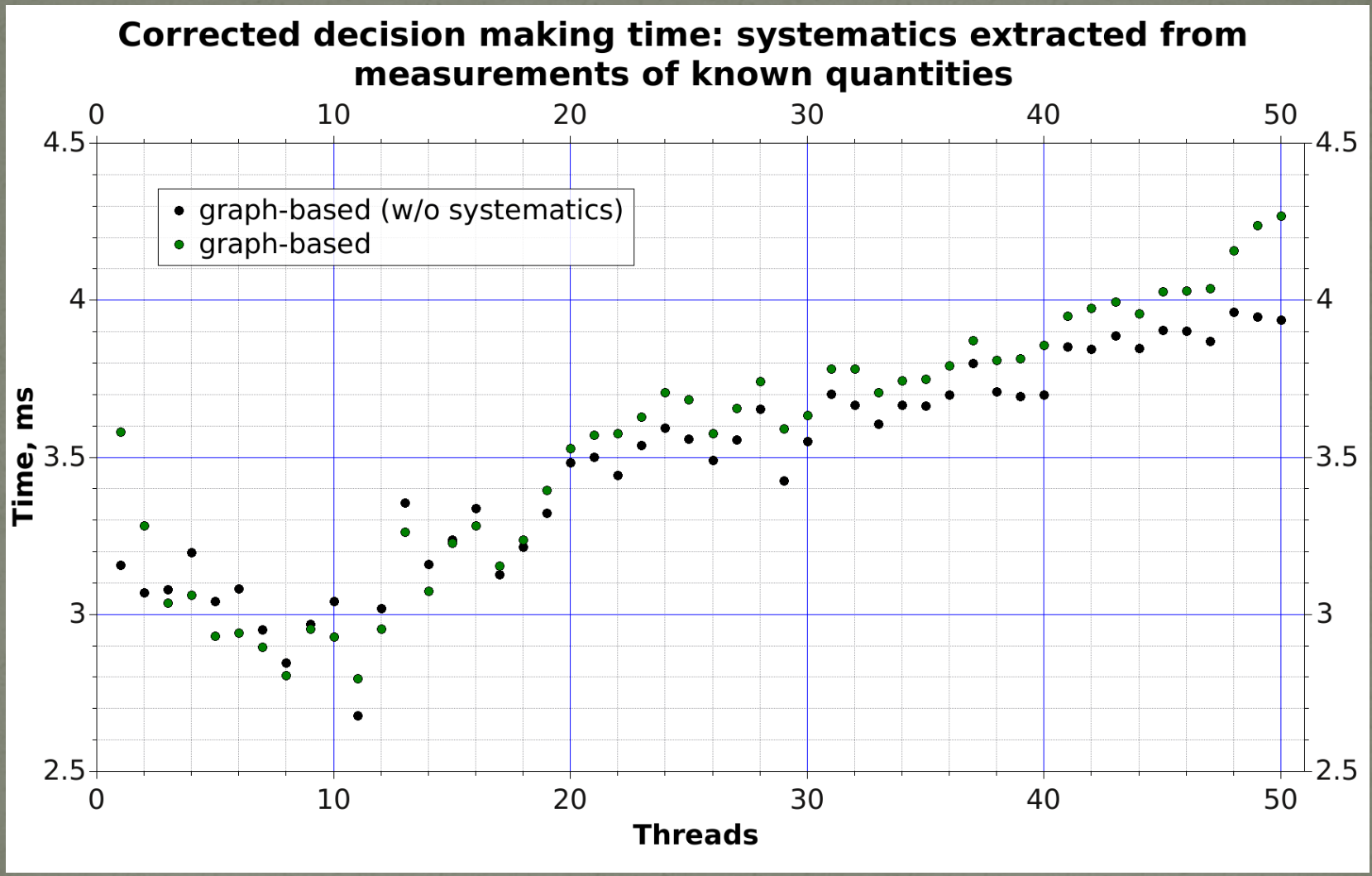
Correlation between algorithm rank and timing



Systematics: zero measurement



Systematics by measurement of known duration



Runaway of decision making