GaudiHive:
towards intra-event scalability
Status Report

Illya Shapoval

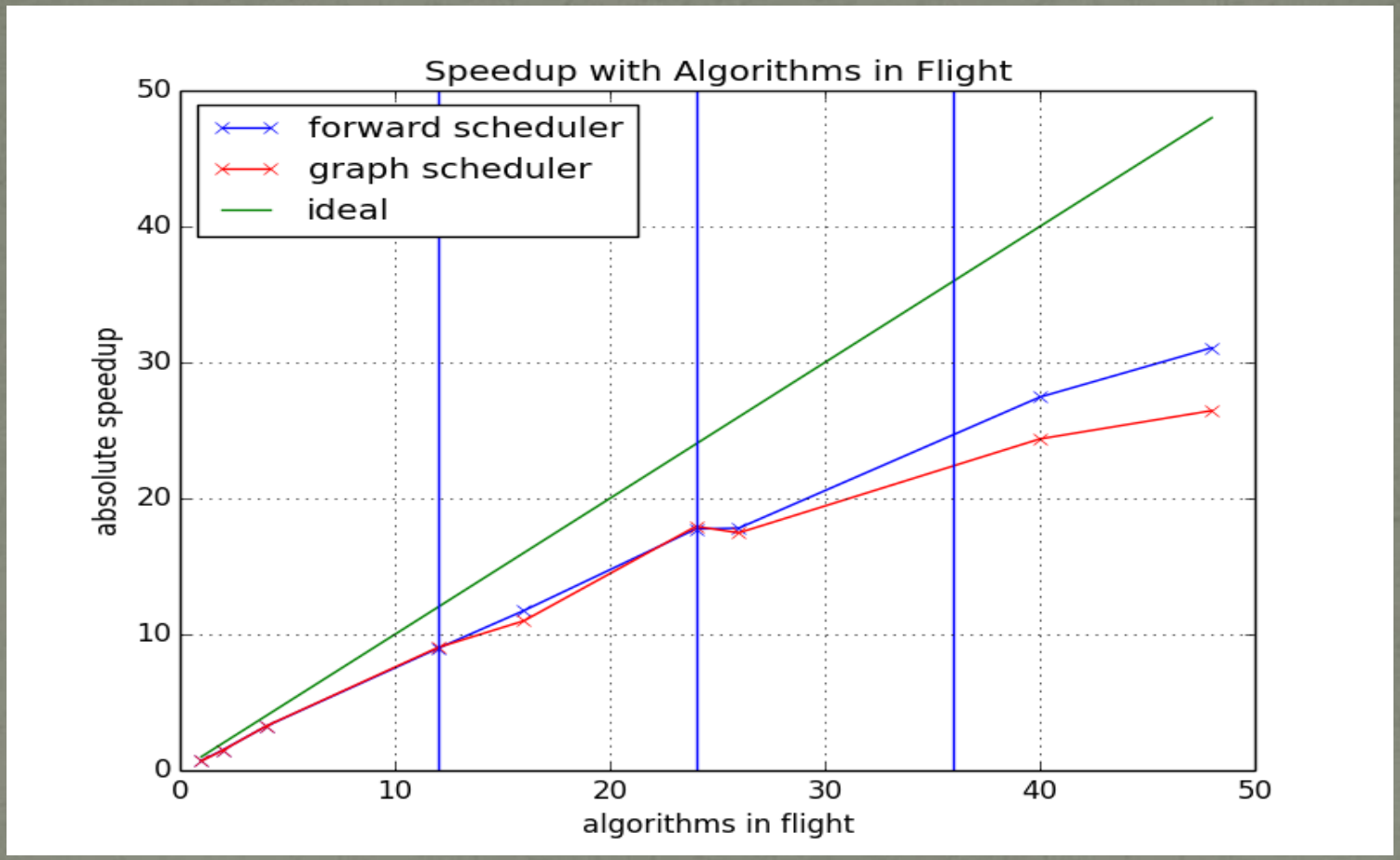
CERN, UNIFE, KIPT, INFN-FE

CF4HEP Meeting
CERN
8 May 2014

Contents

- Forward vs. graph-based scheduling: scalability mismatch problem
- Aspects of intra-event concurrency nature
- Plans

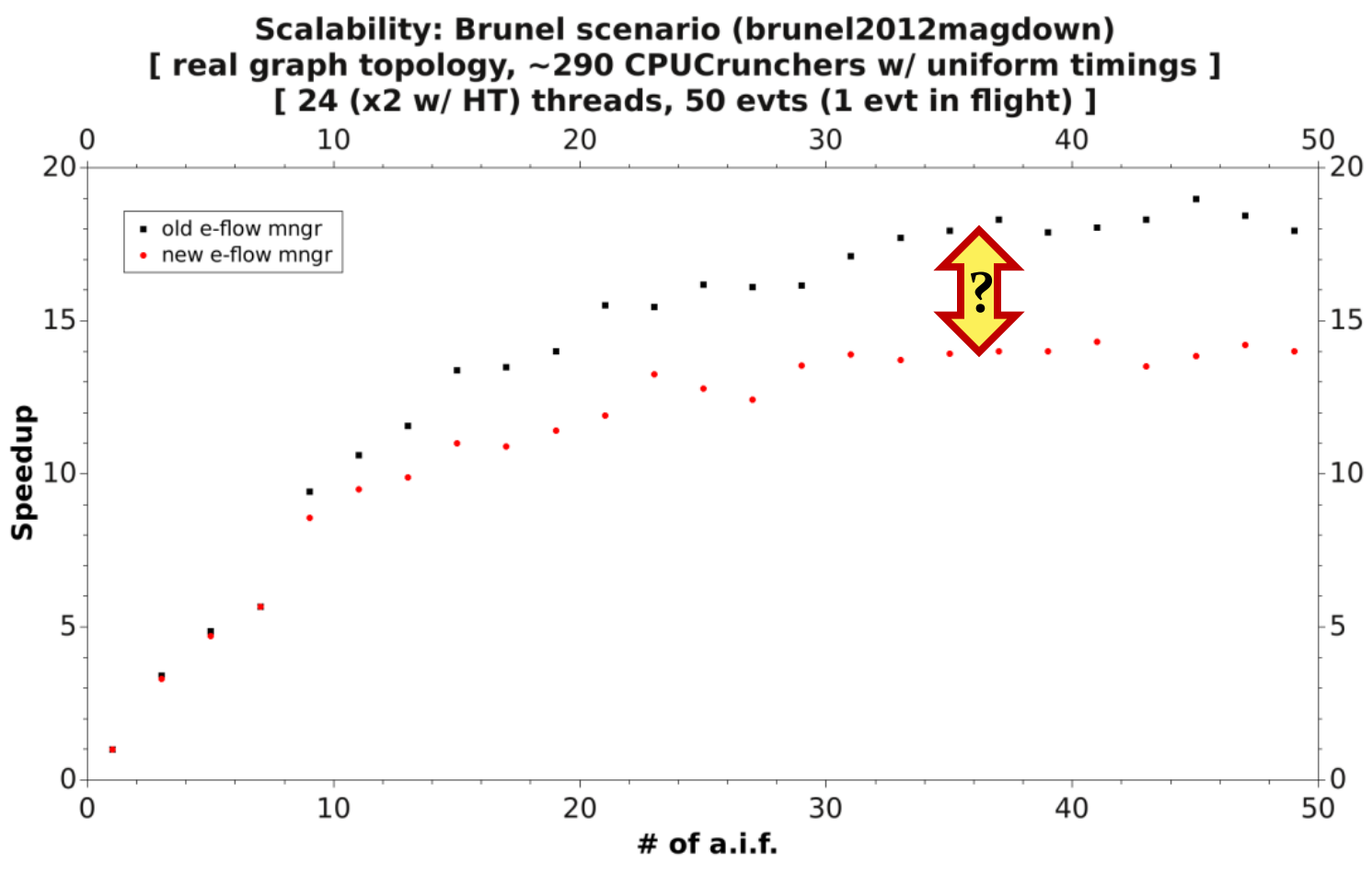
Fwd vs Graph-based scheduling: scalability mismatch (with uniform timing of CPUCrunchers)



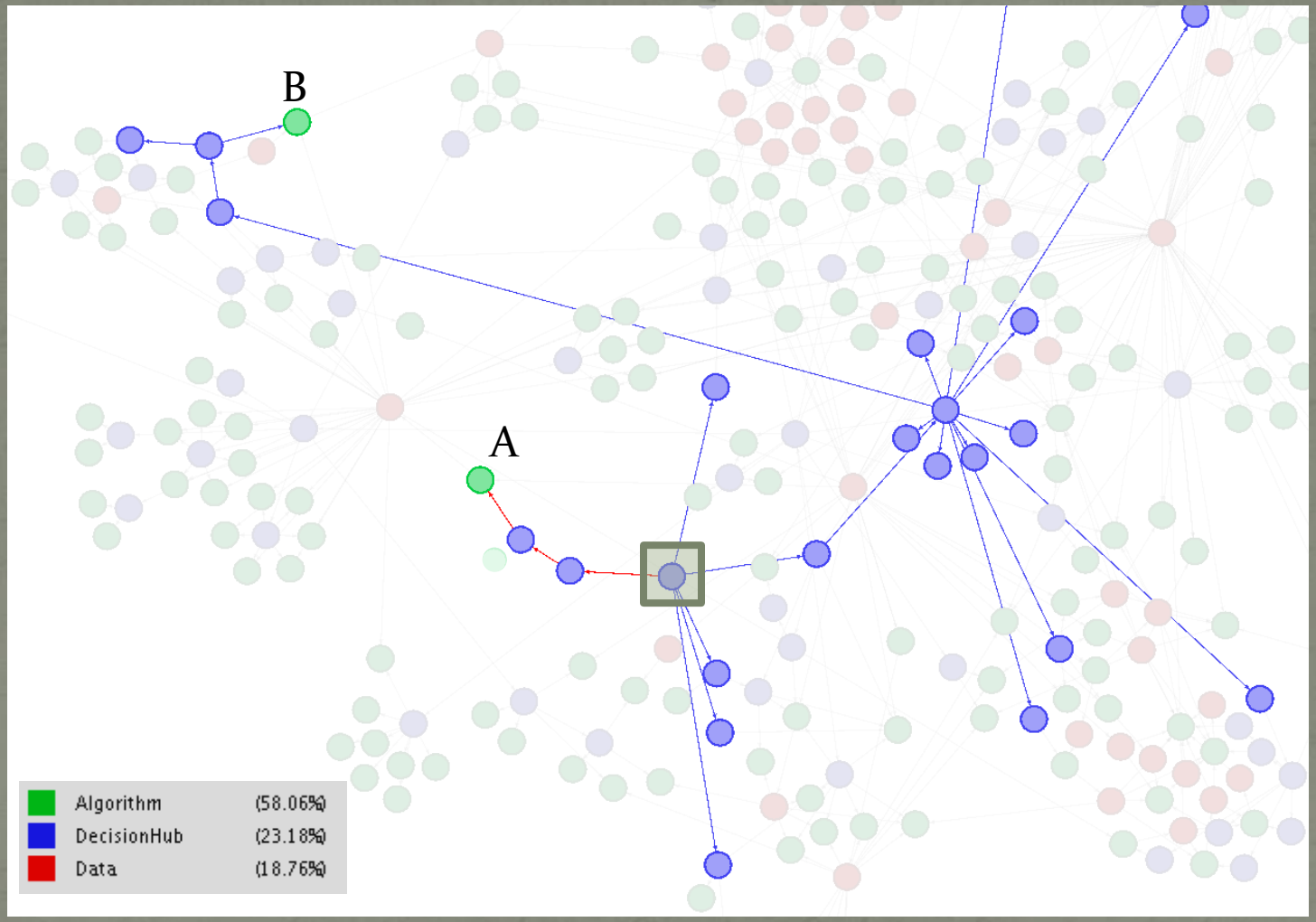
Measurements produced by Daniel F.

Fwd vs graph-based scheduling: scalability mismatch (2)

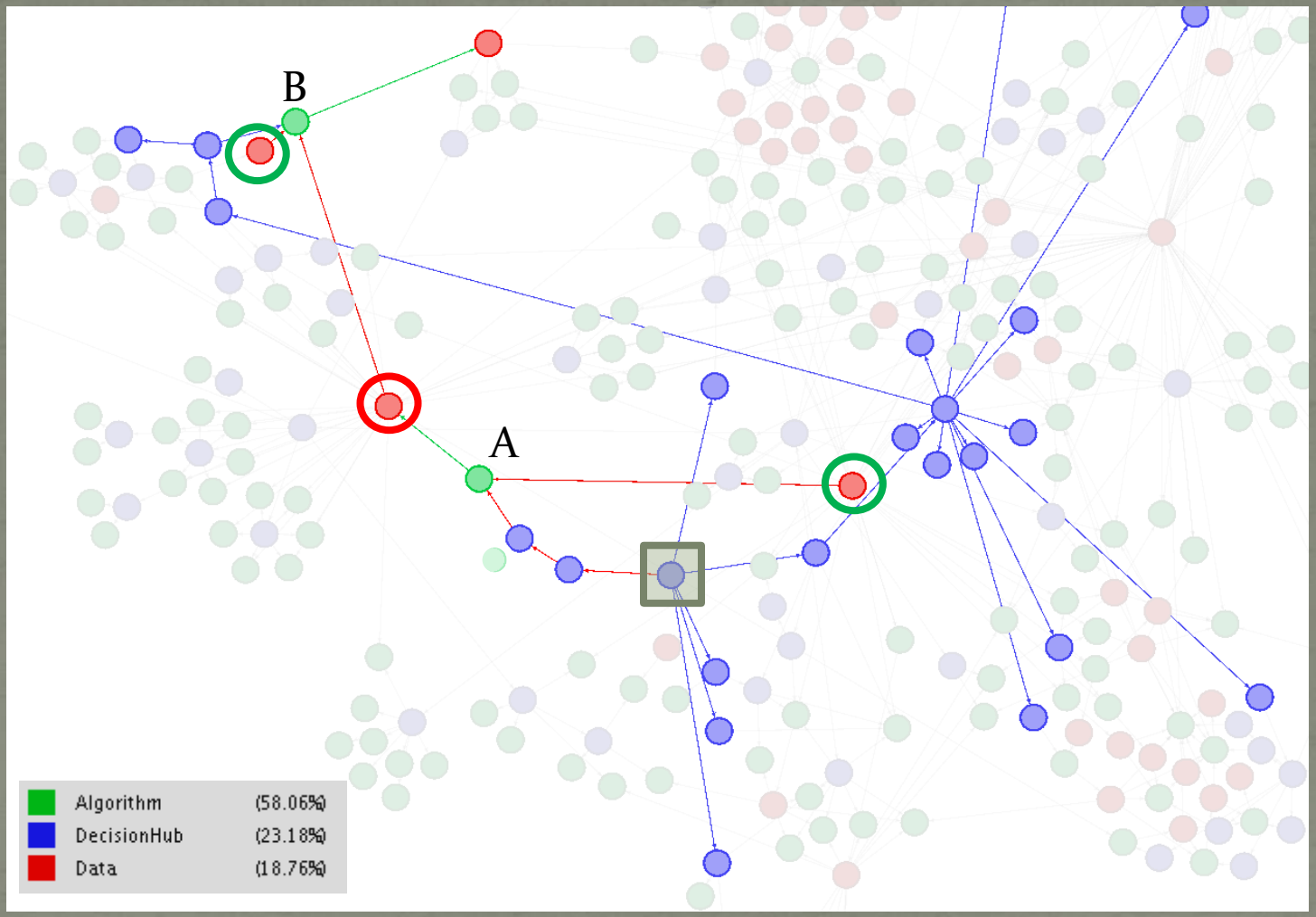
(re-measured with higher resolution)



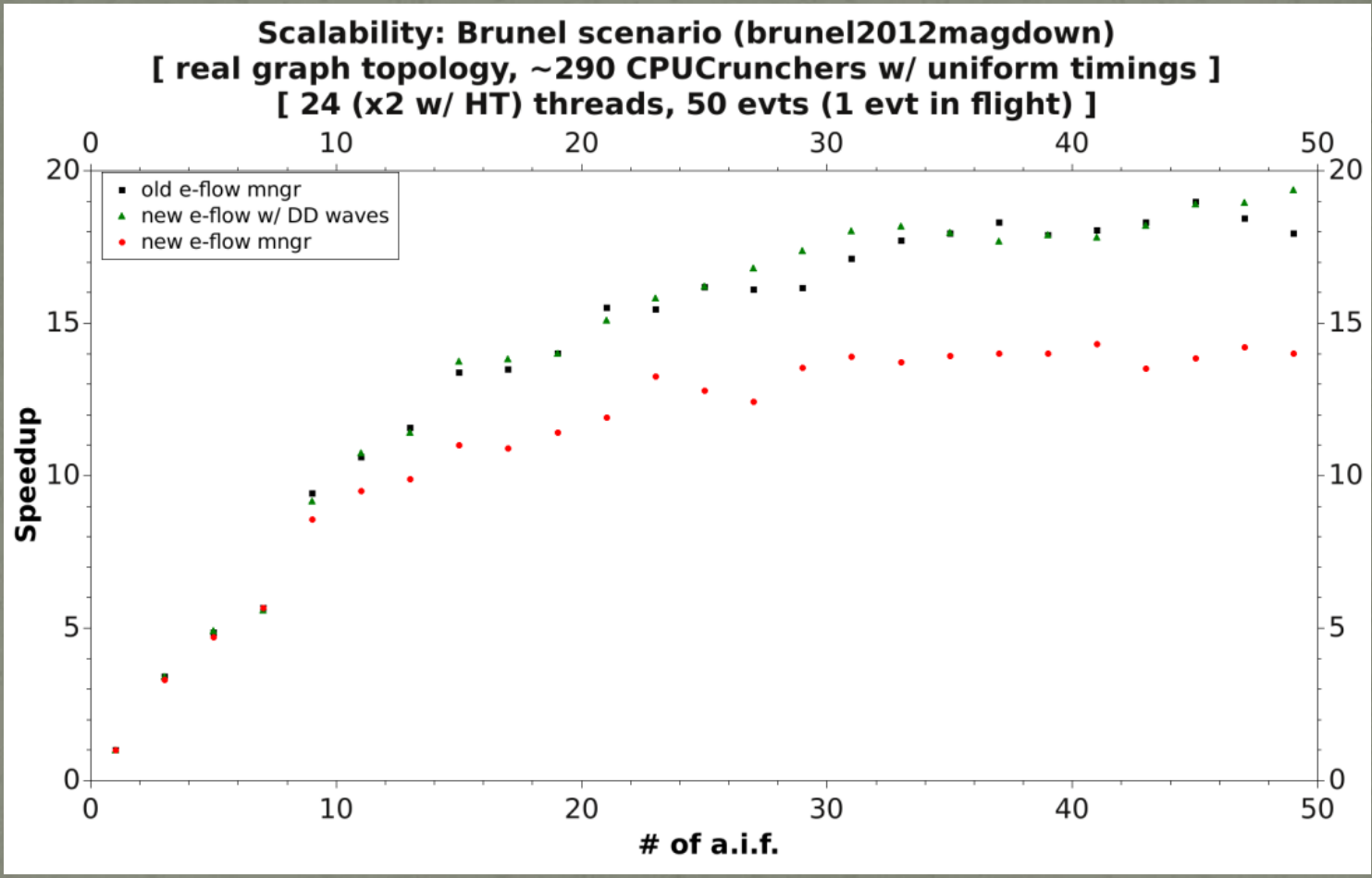
Graph-based e-flow: introducing DD short circuits



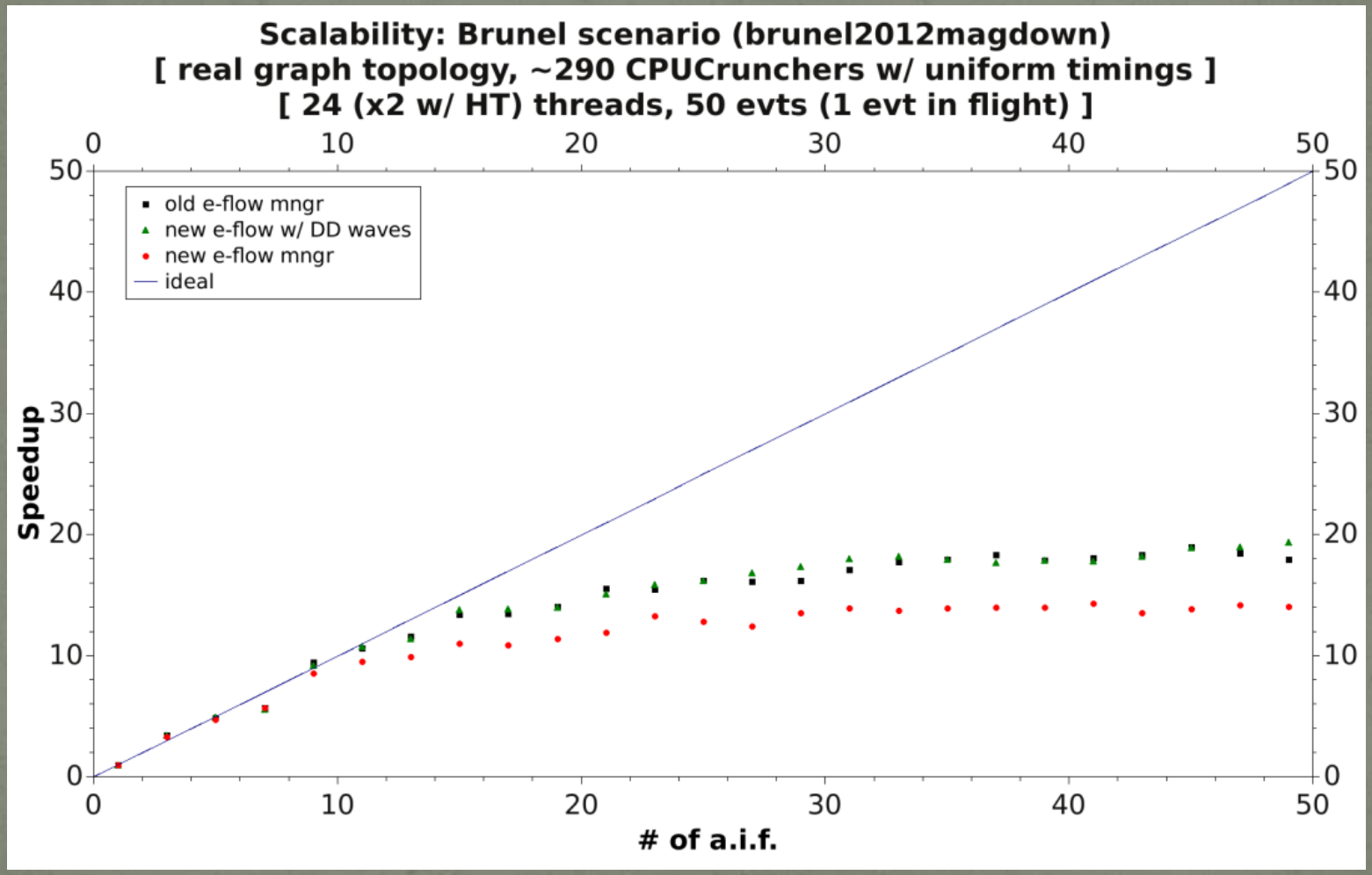
Graph-based e-flow: introducing DD short circuits (2)



The DD short circuits on the dance floor



The DD short circuits on the dance floor (2)

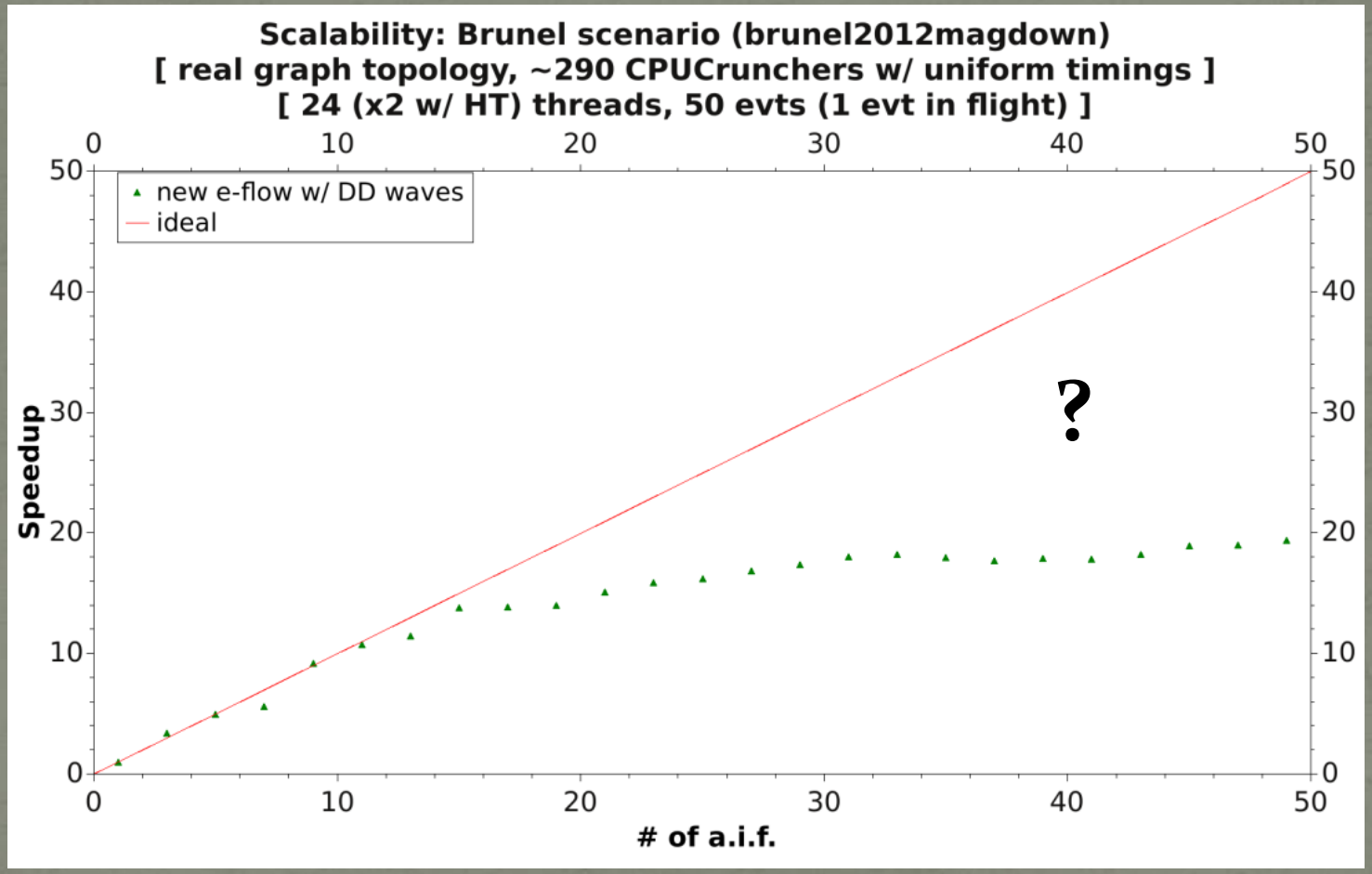


Contents

- Forward vs. graph-based scheduling: scalability mismatch problem
- Aspects of intra-event concurrency nature
- Plans

Theoretical limit for intra-event scalability in GaudiHive

What is our *max_speedup(x)*? well, yes, Amdahl's law.. so what then?

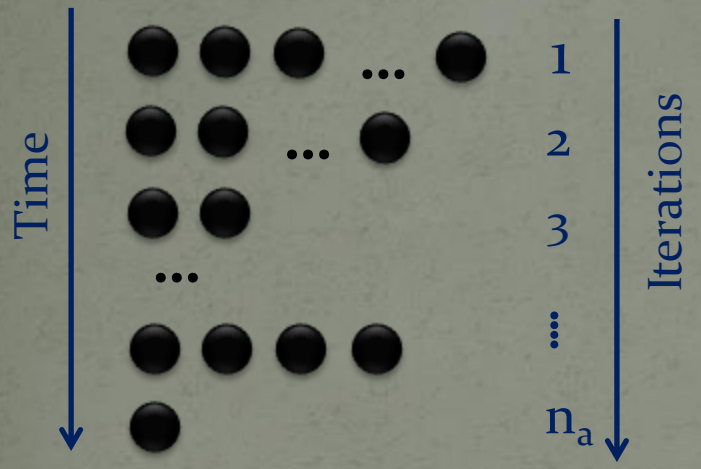


Asymptotical speedup

Assuming:

- Infinite computation resources
- Uniform algorithm timing

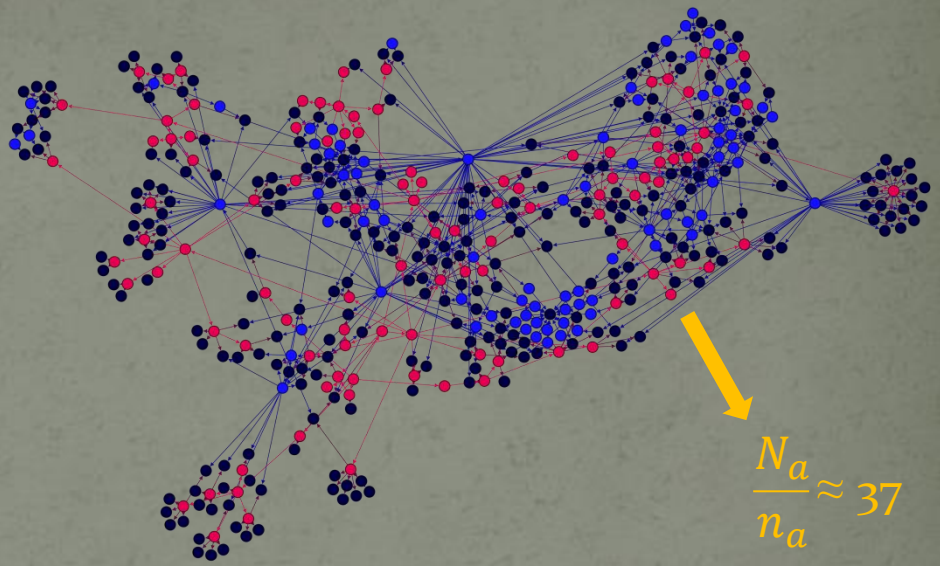
Execution plan:



$$\lim_{x \to \infty} \max\_speedup(x) = \frac{N_a}{n_a}$$

The formalism can be revised to drop the assumptions!

brunel2012magdown



$$\frac{N_a}{n_a} \approx 37$$

with all decisions being non-lazy

A simple execution flow ***simulator*** has been developed, which analyses given flow graph at configuration time to report the asymptotical speedup available.

Summary

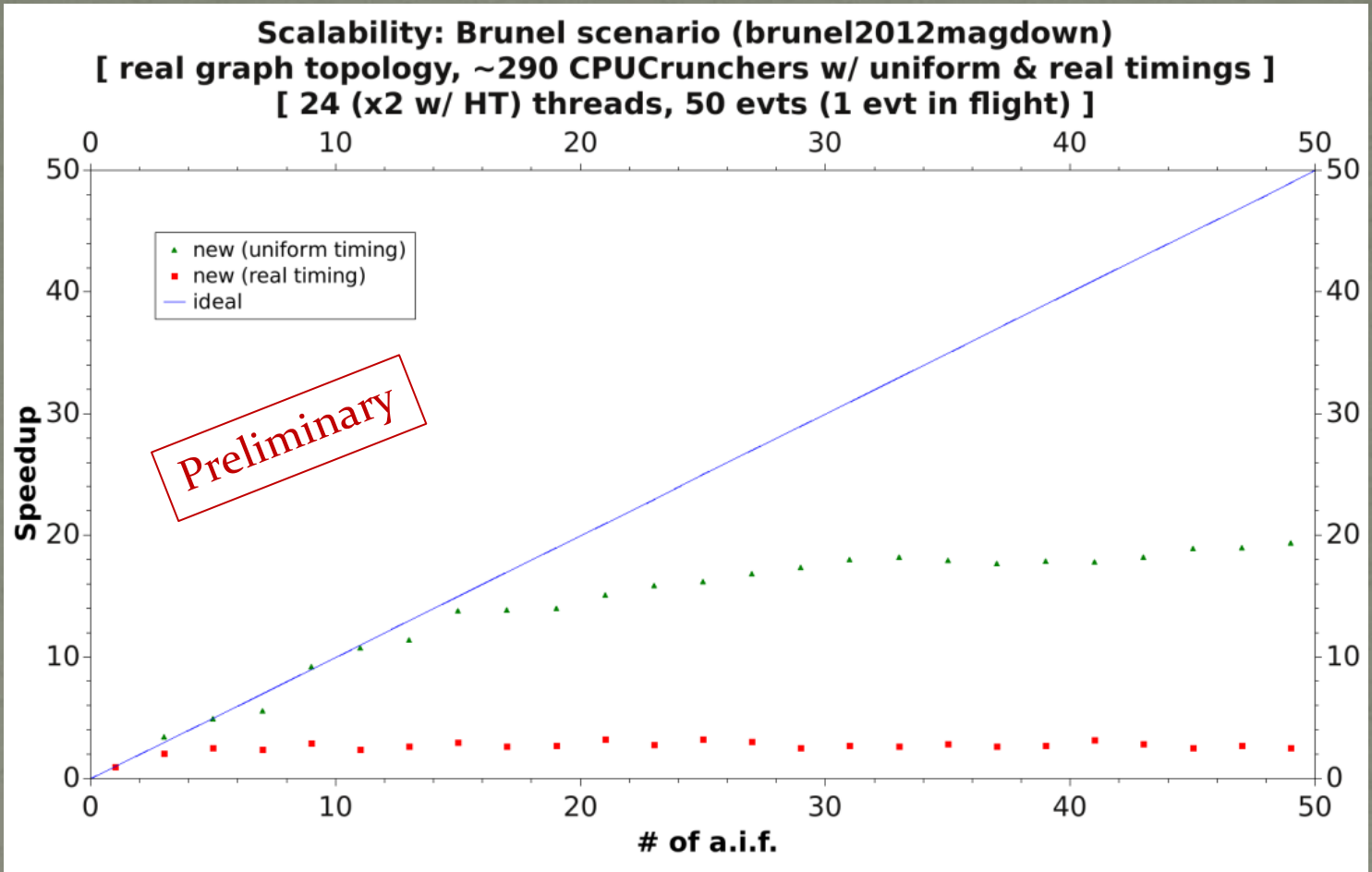
- The consistency in scalabilities of Forward and Graph-based scheduling technique is reached
  - DD short circuit notifications picked up by v0.6 milestone
- Noticeable amount of code reshuffling and improvements in the graph-based e-flow manager
  - Migration to hierarchical visitor pattern started
- Work has been started to better understand the limits of intra-event concurrency depth
  - E-flow simulator developed

(some of) Plans

- Make the graph-based scheduling default (with all the clean-up of the legacy code)?
- Try a couple of execution plan optimization tricks
  - Prioritized algorithm scheduling, based on ranking within the DD realm context;
  - Execution plan reshuffling to maximize the execution flow.
- A note on the graph-based e-flow scheduling description (pending)
- Change the recursive design pattern to hierarchical visitor pattern everywhere
- Migration to the boost graph library
  - better testing and debugging capabilities

Backup

Uniform vs. real* timing of algorithms



* Preliminary: real timing extracts need to be cross-checked