Alternative Algorithm for Traversals of the Control Flow Graph in GaudiHive

Illya Shapoval[1,2,3,4]

[1] CERN, [2] KIPT, [3] UNIFE, [4] INFN SF

Thanks go to Benedikt Hegner[1]

CF4HEP meeting, CERN

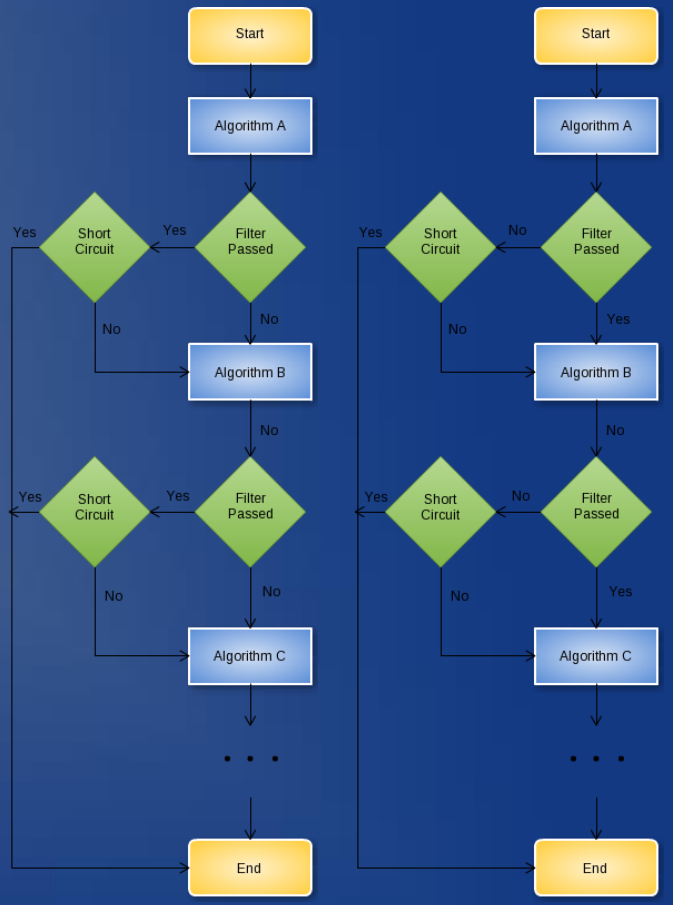
October 2013

Content

- Legacy algorithm in GaudiHive
- Alternative algorithm
- Comparison
- Further steps

Prehistory: Control Flow in Gaudi

- Control flow is supported through:
  - Algorithm *sequencers*
  - Algorithm *filter decisions*
- Two types of sequencers: OR, AND



Control Flow in GaudiHive

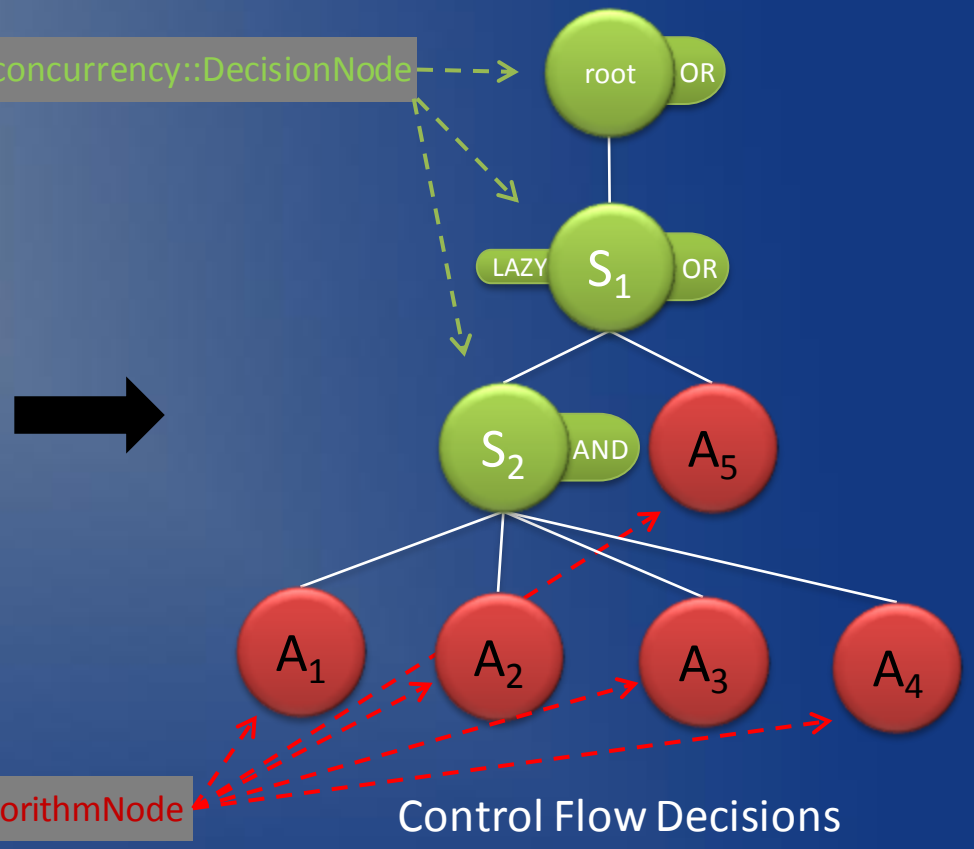
- The good old Gaudi control flow gears are not sufficient in GaudiHive:
  - Order of algorithm execution is not guaranteed

- New control flow machinery was/is needed
- Simple control flow graph is already used in GaudiHive:
  - Constructed during initialization and is static;
  - Shared between events;
  - Used to:
    - Promote algorithms to ControlReady state
    - Update decisions upon every execution of an algorithm

Control Flow: Graph Initialization

Configuration of sequencers:

a_1, a_2, a_3, a_4, a_5 = …
s_2 = GaudiSequencer("S_2")
s_2.ShortCircuit = False
s_2.ModeOR = False
s_2.Members += [a_1, a_2, a_3, a_4]
s_1 = GaudiSequencer("S_1")
s_1.Members += ([s_2, a_5])



Control Flow Decisions

root	S_1	S_2	A_5	A_1	A_2	A_3	A_4
-1	-1	-1	-1	-1	-1	-1	-1

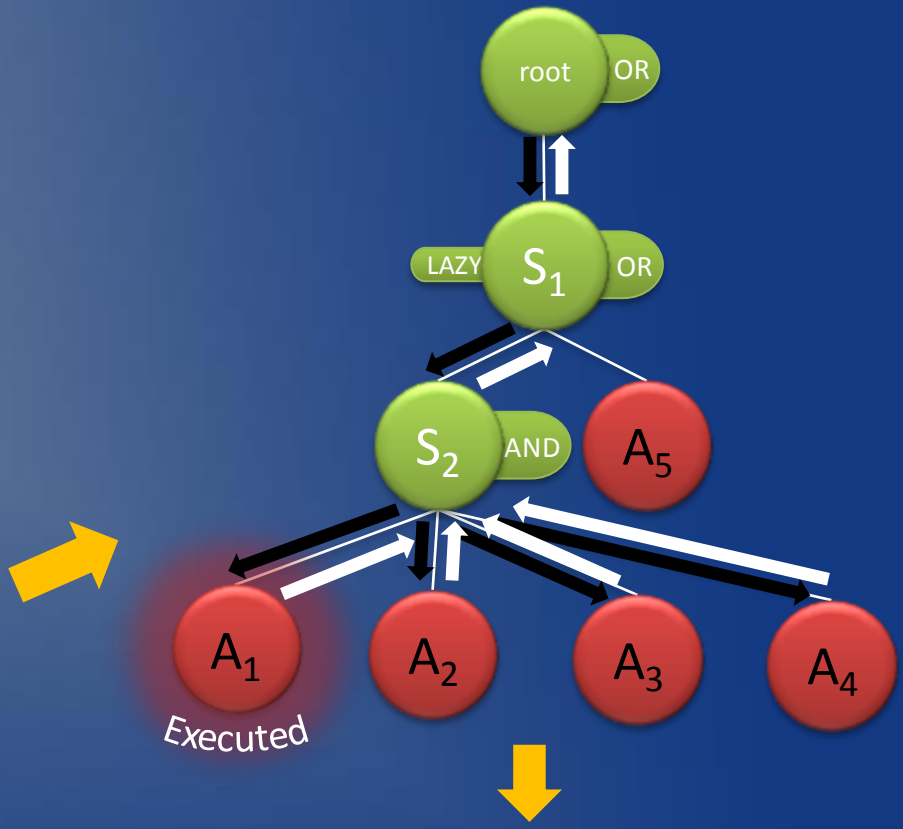
CF Graph: Legacy Traversal Strategy

The traversal:

- is triggered when algorithm's execution is completed;
- always starts at the root node;
- and always returns to the root;

E.g., first traversal within an event:

root → S_1 → S_2 → A_1 → A_2 → A_3 → A_4
↓
root ← S_1 ← S_2



Control Flow Decisions

root	S_1	S_2	A_5	A_1	A_2	A_3	A_4
-1	-1	-1	-1	1	-1	-1	-1

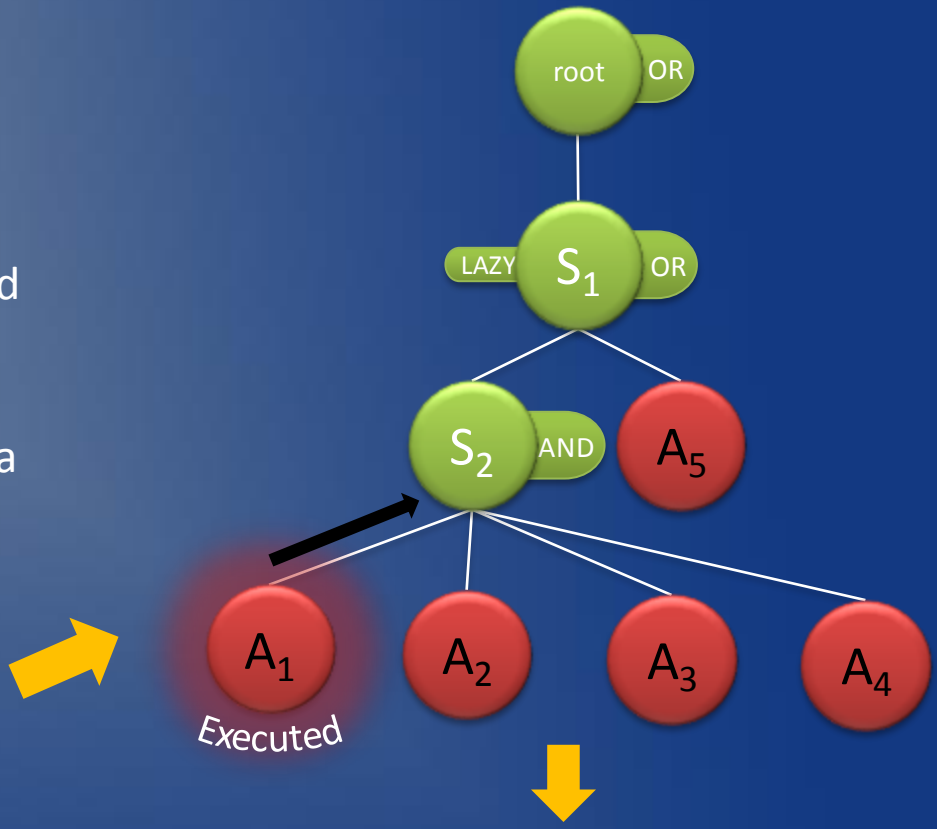
CF Graph: Alternative Traversal Strategy

The traversal:

- is triggered when algorithm's execution is completed;
- always starts at the node of executed algorithm;
- is continued from a particular node further to the top *iaoi* the node got a decision!

E.g., **first** traversal within an event:

$A_1 \rightarrow S_2$



Control Flow Decisions

root	S_1	S_2	A_5	A_1	A_2	A_3	A_4
-1	-1	-1	-1	1	-1	-1	-1

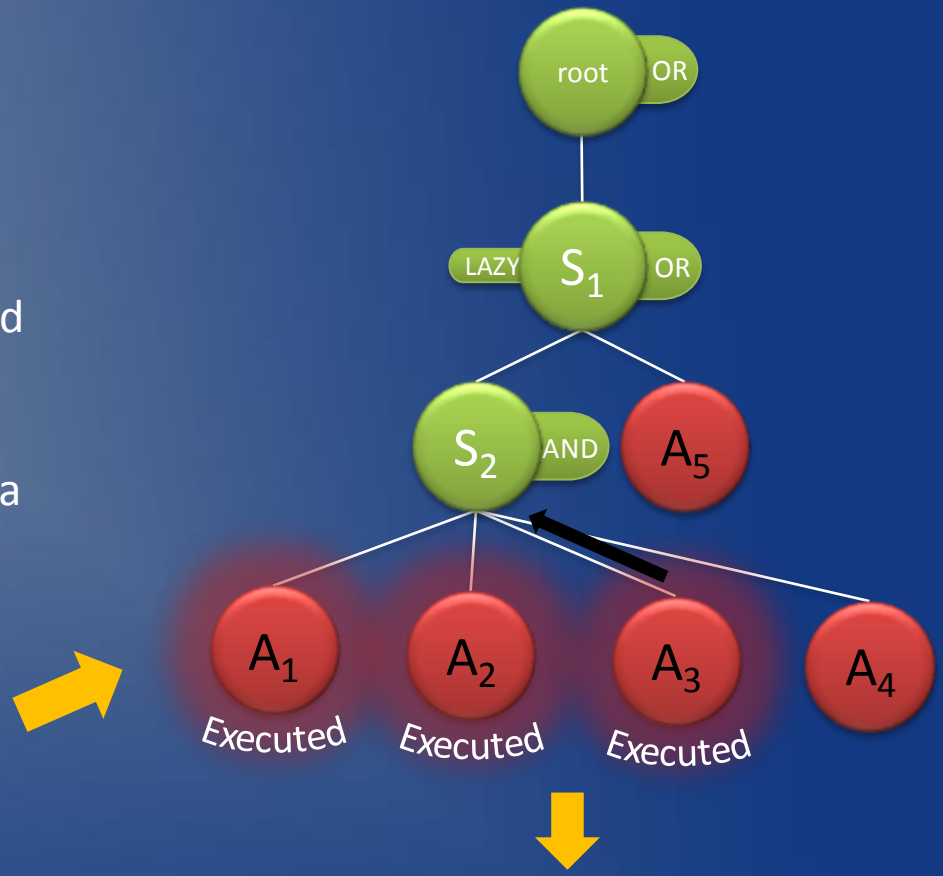
CF Graph: Alternative Traversal Strategy (2)

The traversal:

- is triggered when algorithm's execution is completed;
- always starts at the node of executed algorithm;
- is continued from a particular node further to the top *iaoi* the node got a decision!

Third traversal within an event:

$A_3 \rightarrow S_2$



Control Flow Decisions

root	S_1	S_2	A_5	A_1	A_2	A_3	A_4
-1	-1	-1	-1	1	1	1	-1

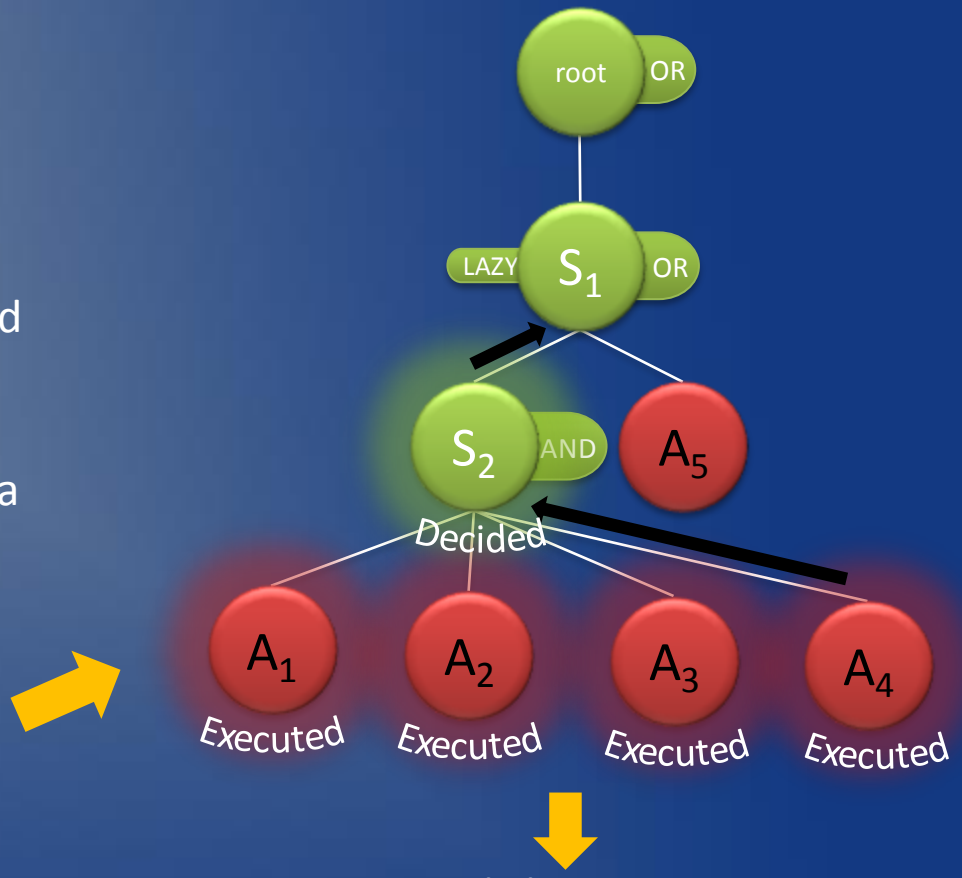
CF Graph: Alternative Traversal Strategy (3)

The traversal:

- is triggered when algorithm's execution is completed;
- always starts at the node of executed algorithm;
- is continued from a particular node further to the top *iaoi* the node got a decision!

Forth traversal within an event:

$A_4 \rightarrow S_2 \rightarrow S_1$



Control Flow Decisions

root	S_1	S_2	A_5	A_1	A_2	A_3	A_4
-1	-1	1	-1	1	1	1	1

Worst-case time complexities of the algorithms

Notations:

n_a - total number of algorithm nodes in the CF graph;

n_s - total number of sequencer nodes in the CF graph;

N := $n_a + n_s$ - size of the CF graph;

T – worst-case time complexity of an algorithm.

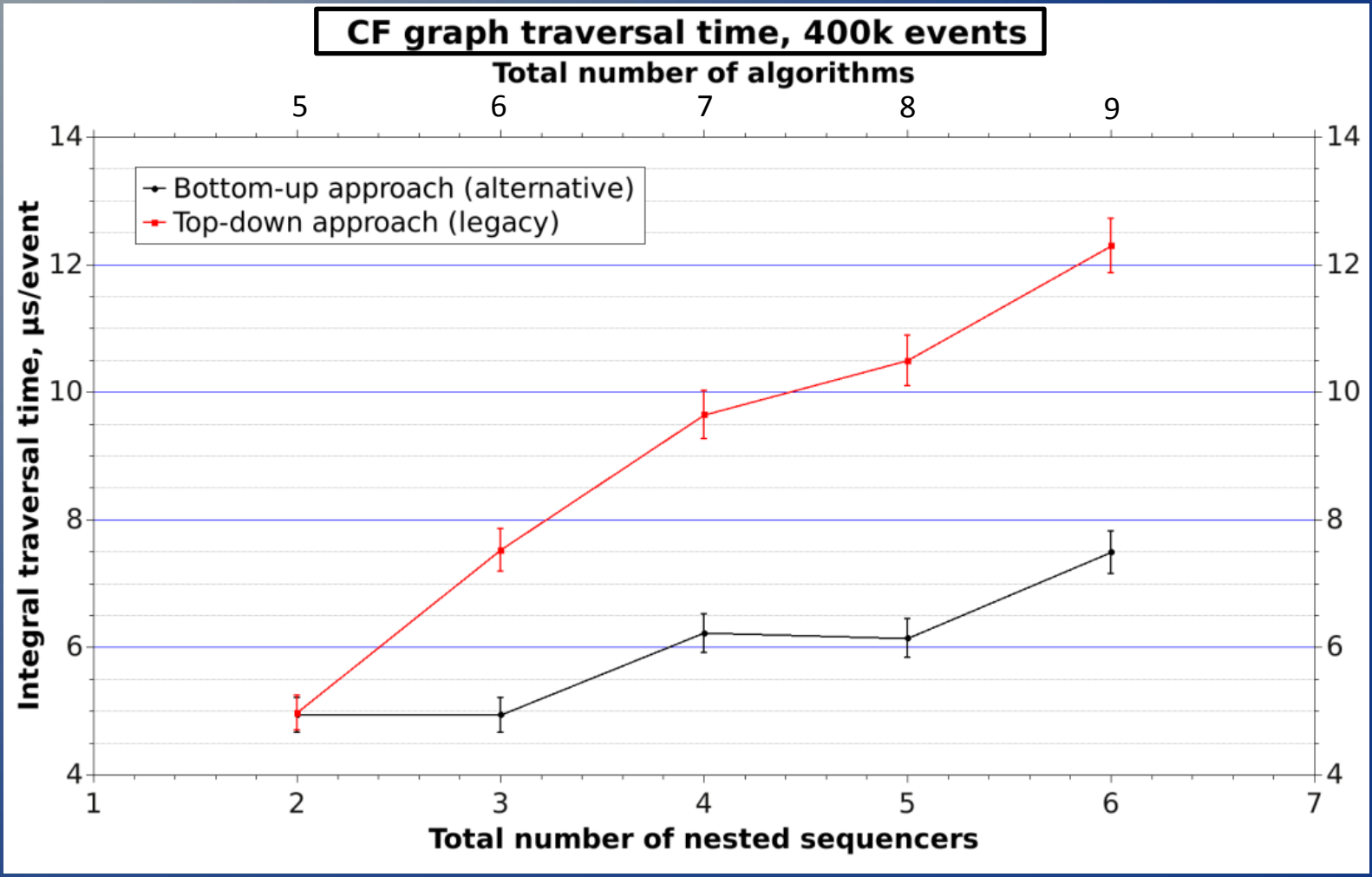
Use cases	Time complexity	
	Top-down approach	Bottom-up approach
$T(n_s, n_a) \in$	$O(n_a(n_a + n_s))$	$O(n_a + n_s)$
$T(n_s, n_a)\big\vert_{n_s=const} \in$	$O({n_a}^2)$	$O(n_a)$
$T(n_s, n_a)\big\vert_{n_a=const} \in$	$O(n_s)$	$O(n_s)$

Alternative CF graph traversal algorithm

- The alternative algorithm is implemented:
  - was completed in August;
  - committed to GaudiHive git;
  - exists side by side with the old algorithm;
  - can be activated by FSS configurable:

    ForwardSchedulerSvc(…, ControlFlowManagerNext=True)

Performance of the algorithms



Summary

- The new algorithm for CF graph traversals has been suggested
- The algorithm has been implemented
- Basic performance measurements have demonstrated
  - better performance is noticeable (e.g., 80% improvement already for 6 sequencers and 9 algorithms);
  - more scalable to graph size.

Further steps

- Making the new algorithm default (if we decide so)
  - Clean up of the current CF machinery code base;
- Movement towards the use of the Boost Graph Library
  - standardized generic interface for traversing graphs
- Proposal on an advanced machinery for CF management

BACKUP

Traversal performance: graph sample