Overview
Difference to BFS
Input graphs
Implementations
Architecture
Results

Parallel Topological Sorting
Design of High Performance Computing, Fall 2015

Kevin Wallimann   Johannes Baum   Matthias Untergassmair

ETH Zürich

December 14, 2015

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
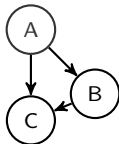ARCHITECTURE
RESULTS

OVERVIEW

- DAG defines partial order
- Topological sorting defines one total order on a DAG
- Parallel algorithm: finds one topological sorting of a given DAG

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

DIFFERENCE TO BFS

- BFS visits every node
- Topological sorting algorithm needs to visit every edge

Example:

Consider order A,C,B $\rightarrow$ valid in BFS, invalid in topological sorting
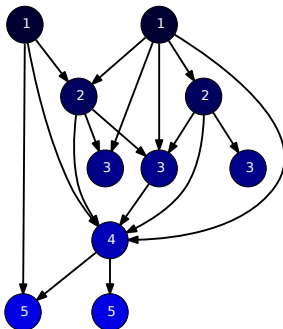
OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

RANDOM GRAPH

- Parameter:
  Average node
  degree

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS
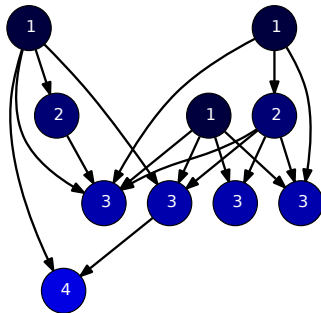
# SOFTWARE DEPENDENCY GRAPH

Nodes	Edges	Degree (Median)
100'000	266'680	2
10'000	27'416	2

Musco, V. et al. (2014) "A Generative Model of Software Dependency Graphs to Better Understand Software

Evolution."

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

PARALLEL ALGORITHM (SHARED MEMORY)

1. As a preparing step, initialize a counter for every child with the number of its parents.

2. Distribute parent nodes over threads and process them in parallel.

3. For every parent, append it to solution ($\rightarrow$ **lock**).

4. For every child of the parent, decrement its counter ($\rightarrow$ **lock**). If the counter is zero, make it a parent.

5. When all parents are processed ($\rightarrow$ **barrier**), distribute new parent nodes etc.

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS
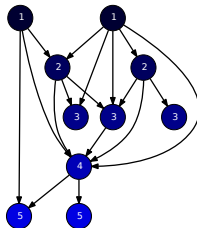
LOCAL LISTS APPROACH

- Parent nodes stored in a shared list.
- Distribution of parent nodes: Scatter the list among the threads. Each thread has now its local list.
- Add new parents to the end of local list.
- When all parents were processed, gather all local lists into the shared list.
- Repeat until there are no parents in the shared list.

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

BOOLEAN ARRAY APPROACH

- Array of length N. 1 if node i is a parent node, 0 otherwise.
- Distribution of parent nodes: Parallel for-loop through the array.
- Mark new parents by setting a 1 in a second array.
- When all parents were processed, swap arrays.
- Loop through the array until there are no new parents.

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

OPTIMISTIC COUNTER CHECK

- Decrement shared counter
- Return true if counter is zero

```
inline bool counterCheck() {
        bool lastone;
        #pragma omp critical
        {
          --parcount_;
          lastone = (parcount_ == 0);
        }
        return lastone;
}
```

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

OPTIMISTIC COUNTER CHECK

- Decrement shared counter
- Return true if counter is zero

```
inline bool counterCheck() {
        #pragma omp atomic
        --parcount_;
        return (parcount_ == 0);
}
```

- Multiple threads could return true, although only one thread should do so.

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

OPTIMISTIC COUNTER CHECK

```
inline bool counterCheck() {
        #pragma omp atomic
        --parcount_;
        return (parcount_ == 0);
}
```
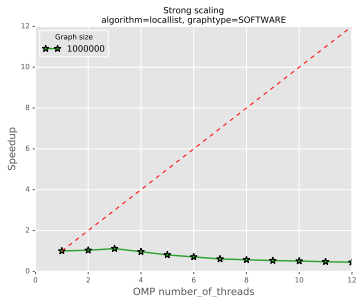
- List based approach: Child is inserted multiple times to solution $\Rightarrow$ Wrong.

- Array approach: Multiple threads write 1 to the array $\Rightarrow$ Ok, doesn't matter.

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

EULER

- Intel Xeon E5 on Euler cluster
- 2 processors per node
- 12 cores
- 30 MB shared last-level cache
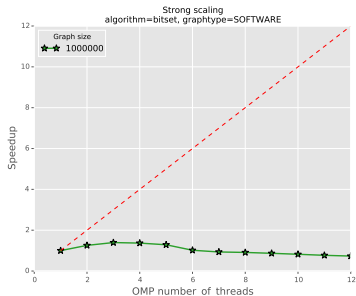- Software graph with around 1M nodes should fit into cache

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

STRONG SCALING SOFTWARE GRAPH

Local lists

Bool array

Absolute runtimes on 1 core	serial		bool array	local list
	0.45 s		0.58 s	0.48 s

OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

BOOL ARRAY & RANDOM GRAPH (NODE DEGREE 30)

Strong scaling

Weak scaling

	serial	bool array
Absolute runtimes on 1 core	2.76 s	3.51 s
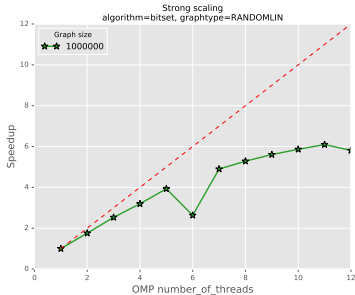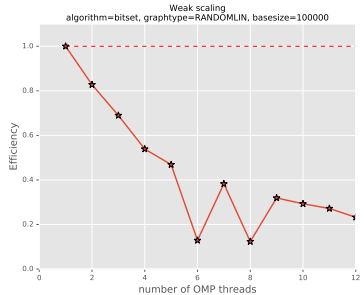
OVERVIEW
DIFFERENCE TO BFS
INPUT GRAPHS
IMPLEMENTATIONS
ARCHITECTURE
RESULTS

ABSOLUTE RUNTIME COMPOSITION

Local lists

Bool array

Overview
Difference to BFS
Input graphs
Implementations
Architecture
Results
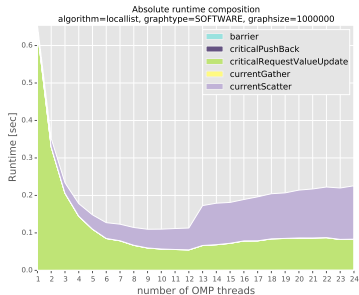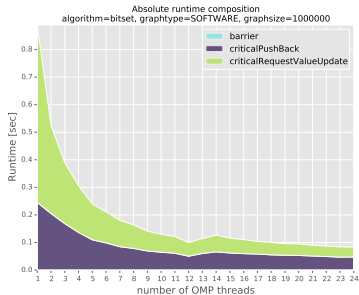
Remaining issues

- Local lists: Work stealing
- Pinpoint the reasons for bad scaling