Parallel Interval Newton Method on CUDA

Philip-Daniel Beck and Marco Nehmeier

Institute of Computer Science
University of Würzburg
Germany

PARA 2012

Outline

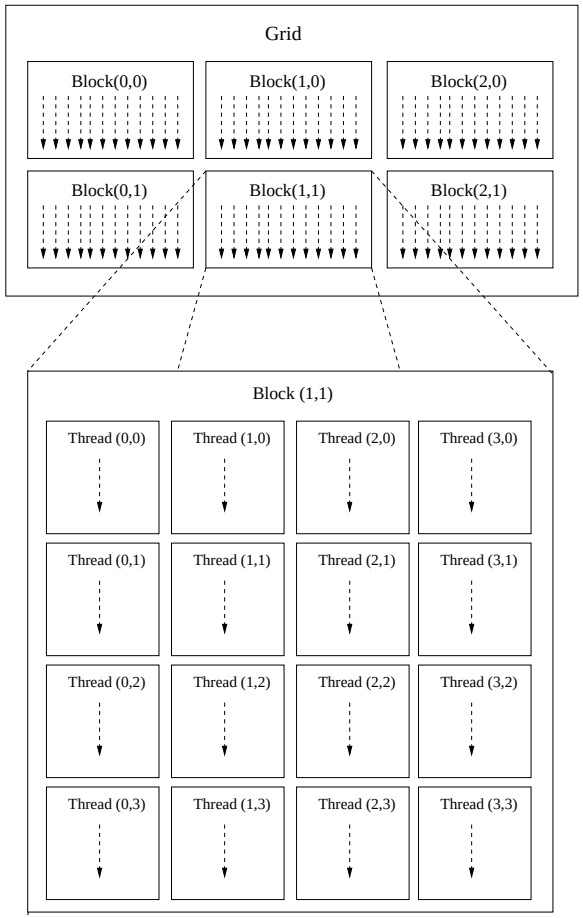
Outline

CUDA
Compute Unified Device Architecture

- GPU architecture for General Purpose Computing
- PCI Express add-in boards
- Highly parallel
  - Several streaming multiprocessors (SM)
    - Each consisting of several cores
- Compute capability = classification
- Usable like a many-core CPU
- CUDA C

CUDA Thread Hierarchy

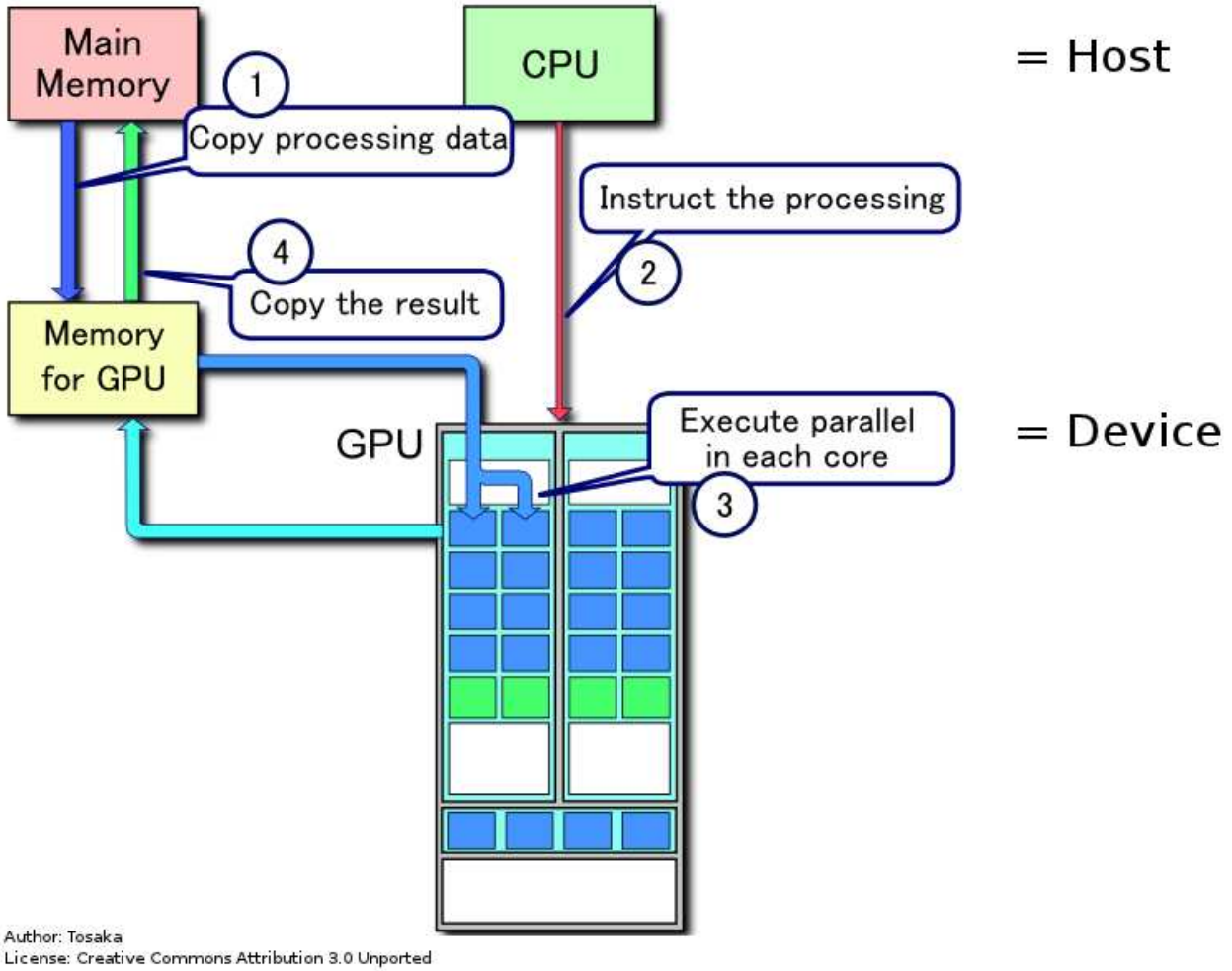
- Grid = group of thread blocks
- Thread block = group of threads running on one SM
- Warp = 32 threads of a thread block
  - SIMT (Single Instruction Multiple Threads)
  - Fundamental unit



CUDA Memory Model

Memory	Access	Scope	Lifetime
Register	R/W	1 thread	Thread
Local	R/W	1 thread	Thread
Shared	R/W	All threads in block	Block
Global	R/W	All threads + host	Host allocation
Constant	R	All threads + host	Host allocation
Texture	R	All threads + host	Host allocation

CUDA Processing flow



Outline

Root Finding
Interval Newton method

$$N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$$

$$X_{n+1} := X_n \cap N(X_n)$$

- Idea: Use all tangents
- Reduce diameter
  - Use bisection otherwise
- Nested intervals
  - Find all zeros

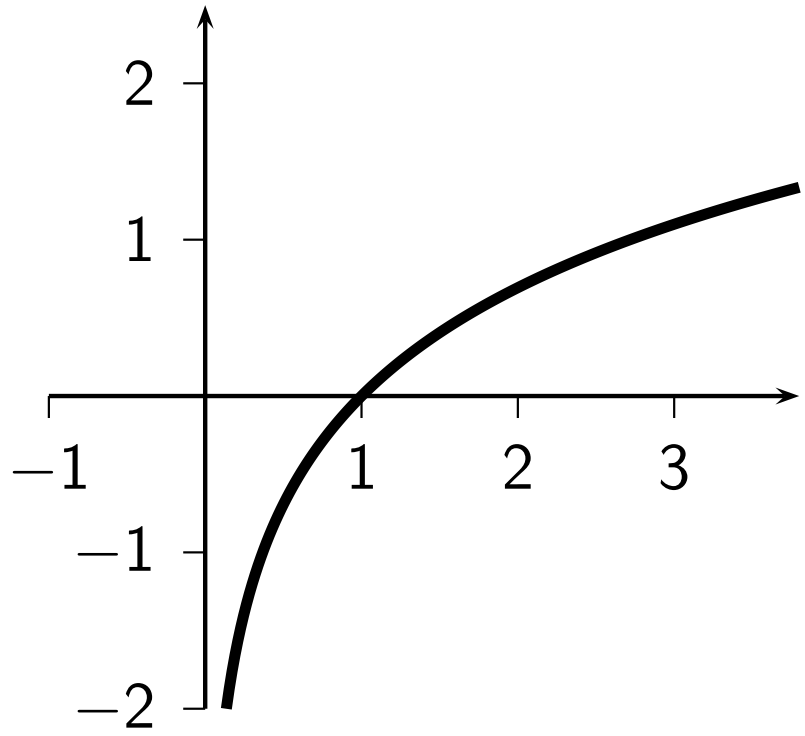
Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

$X_{n+1} \neq \emptyset$?
$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?
$\Rightarrow$ Bisection of X_{n+1}
$\Rightarrow$ Two rec. calls



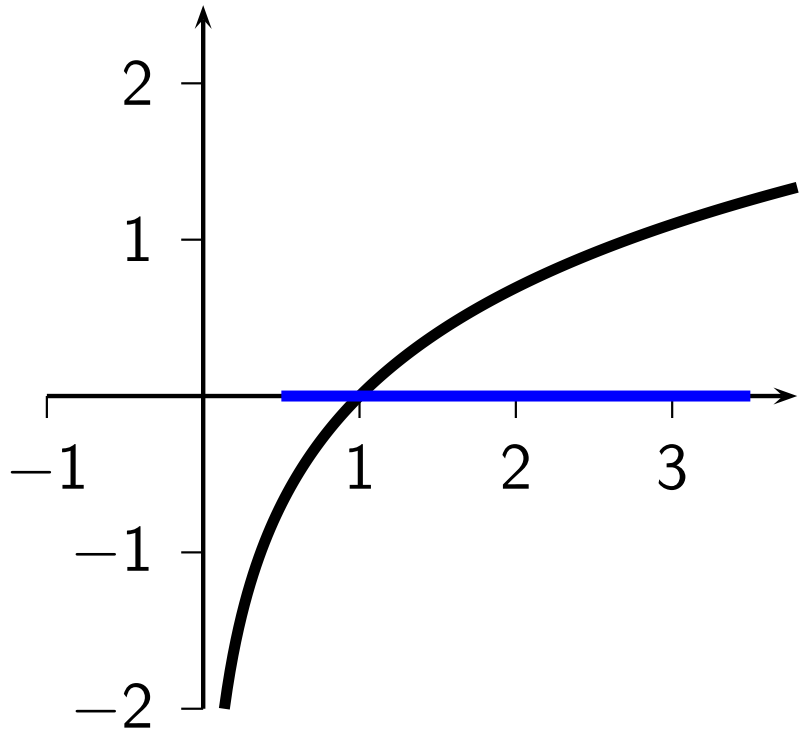
Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

$X_{n+1} \neq \emptyset$?
$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?
$\Rightarrow$ Bisection of X_{n+1}
$\Rightarrow$ Two rec. calls



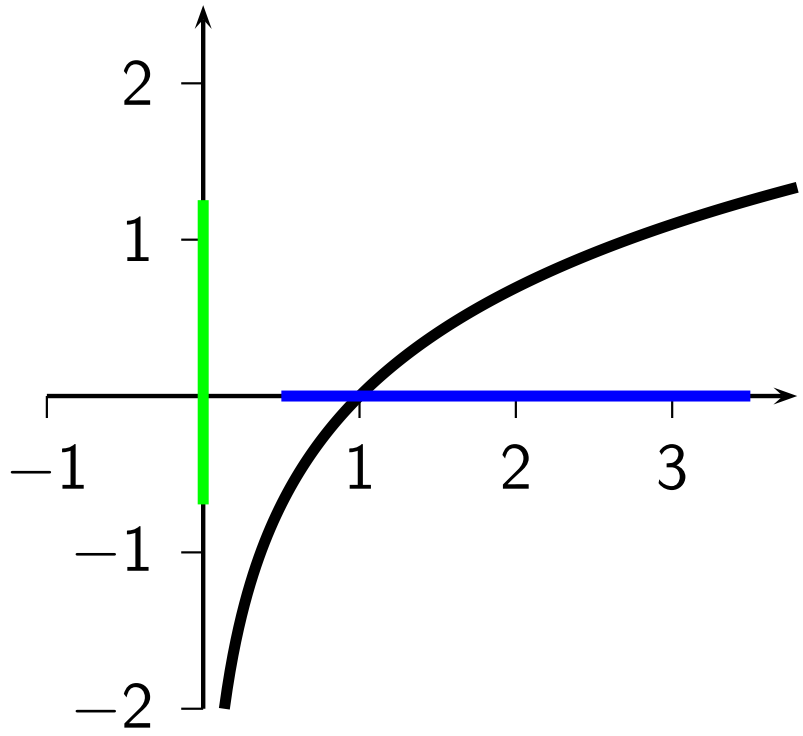
Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

$X_{n+1} \neq \emptyset$?
$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?
$\Rightarrow$ Bisection of X_{n+1}
$\Rightarrow$ Two rec. calls



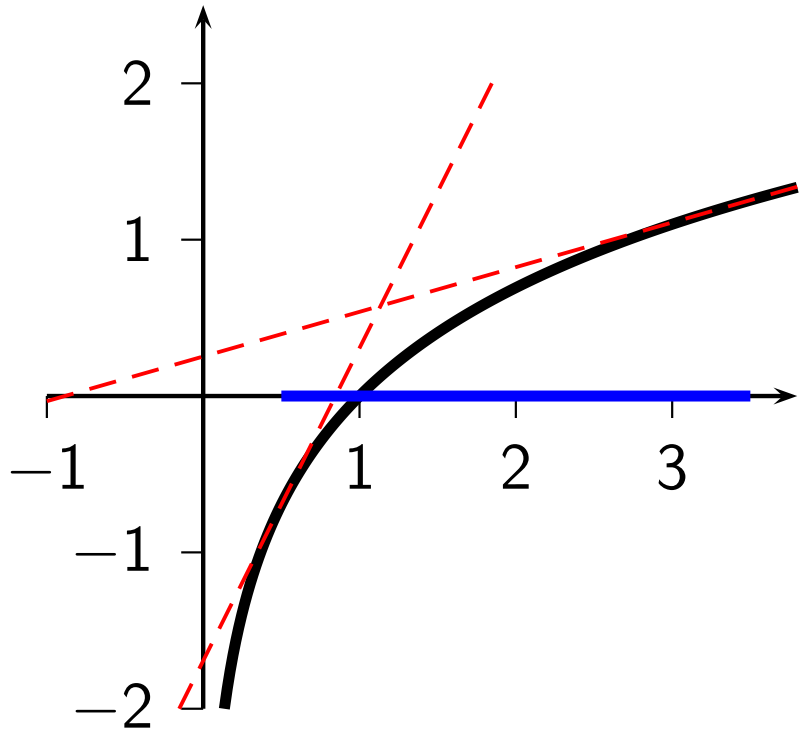
Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

$X_{n+1} \neq \emptyset$?
$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?
$\Rightarrow$ Bisection of X_{n+1}
$\Rightarrow$ Two rec. calls



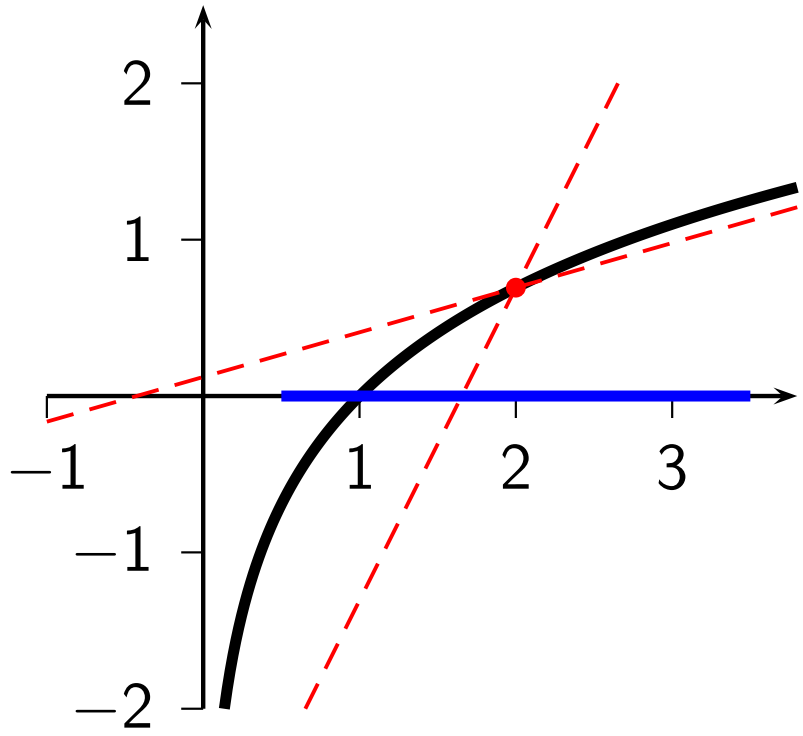
Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

$X_{n+1} \neq \emptyset$?
$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?
$\Rightarrow$ Bisection of X_{n+1}
$\Rightarrow$ Two rec. calls



Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

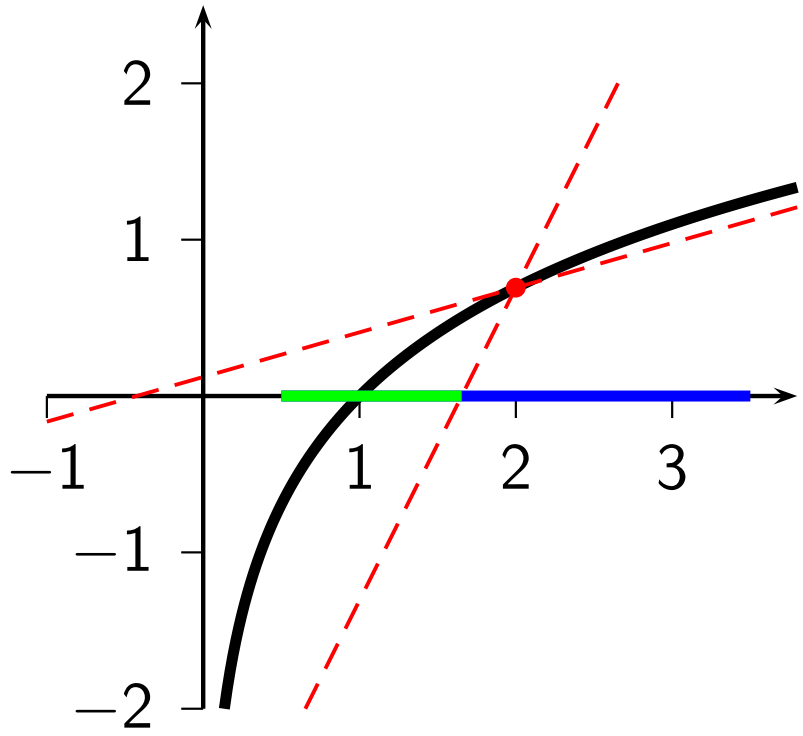
$X_{n+1} \neq \emptyset$?

$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?

$\Rightarrow$ Bisection of X_{n+1}

$\Rightarrow$ Two rec. calls



Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

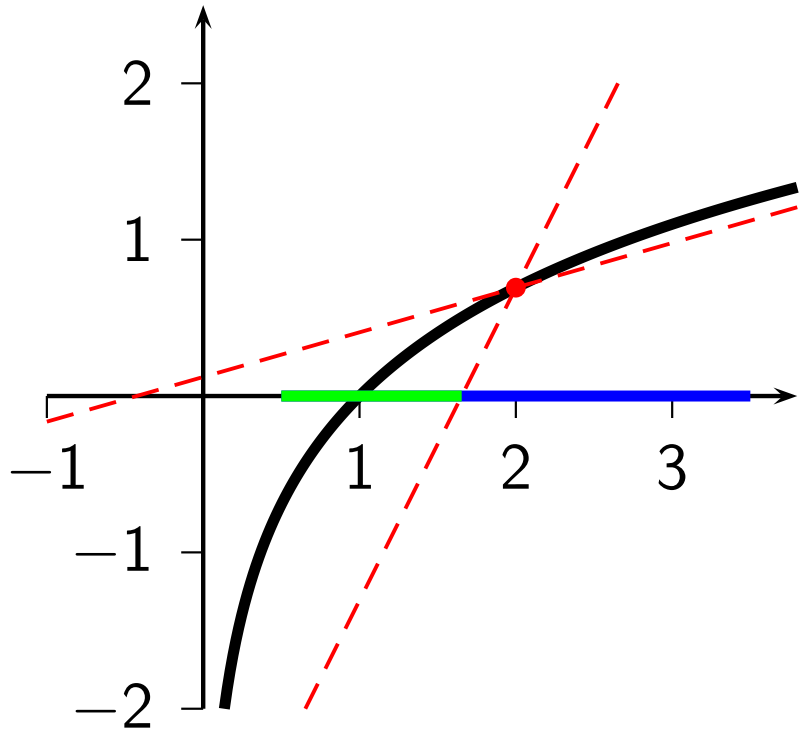
$X_{n+1} \neq \emptyset$?

$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?

$\Rightarrow$ Bisection of X_{n+1}

$\Rightarrow$ Two rec. calls



Interval Newton method

Illustration of the algorithm

1. Function f
2. Interval X_n
3. $0 \in F(X_n)$?
4. $N(X_n) := m(X_n) - \frac{f(m(X_n))}{F'(X_n)}$
5. $X_{n+1} := X_n \cap N(X_n)$

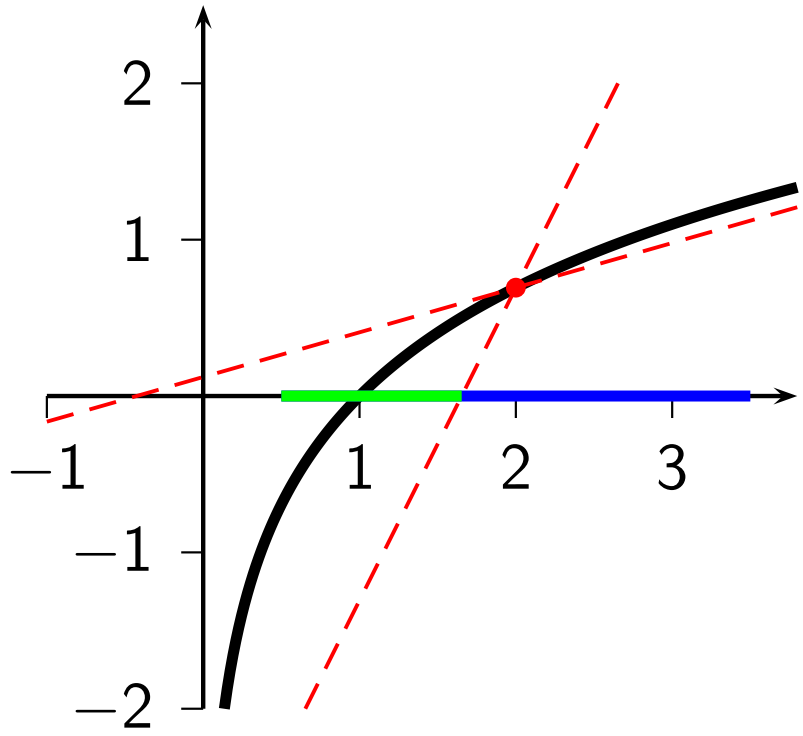
$X_{n+1} \neq \emptyset$?

$\Rightarrow$ Rec. call with X_{n+1}

$X_{n+1} = X_n$?

$\Rightarrow$ Bisection of X_{n+1}

$\Rightarrow$ Two rec. calls



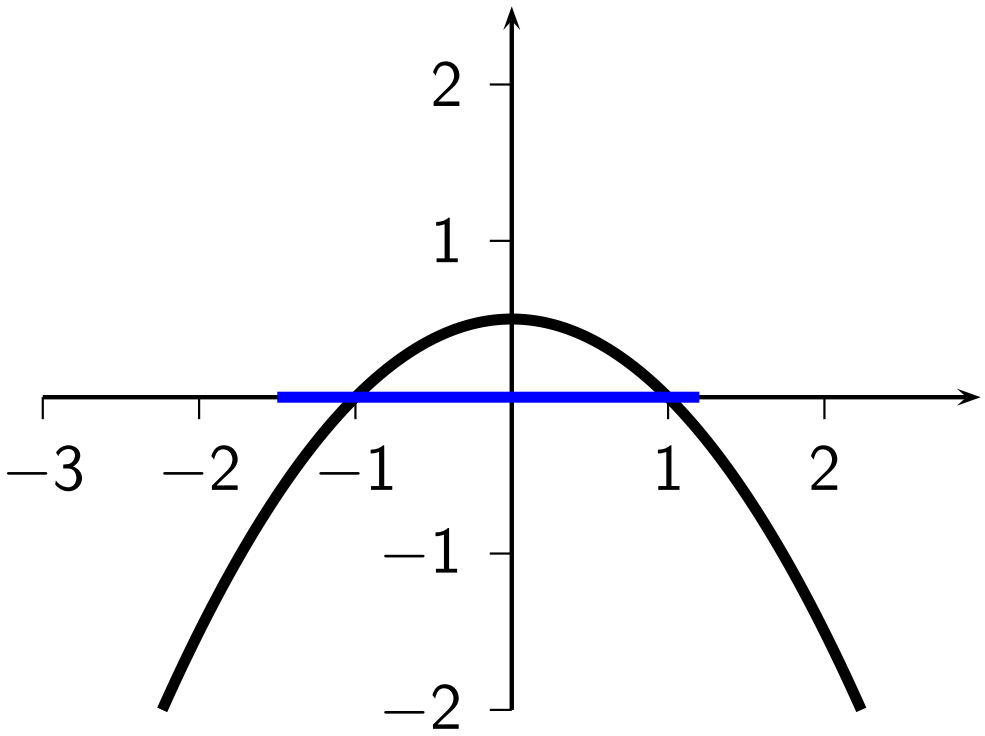
Interval Newton method

Illustration of the algorithm

Non monotonic ?

$\Rightarrow$ Extended interval arithmetic

$\Rightarrow$ Two recursive calls



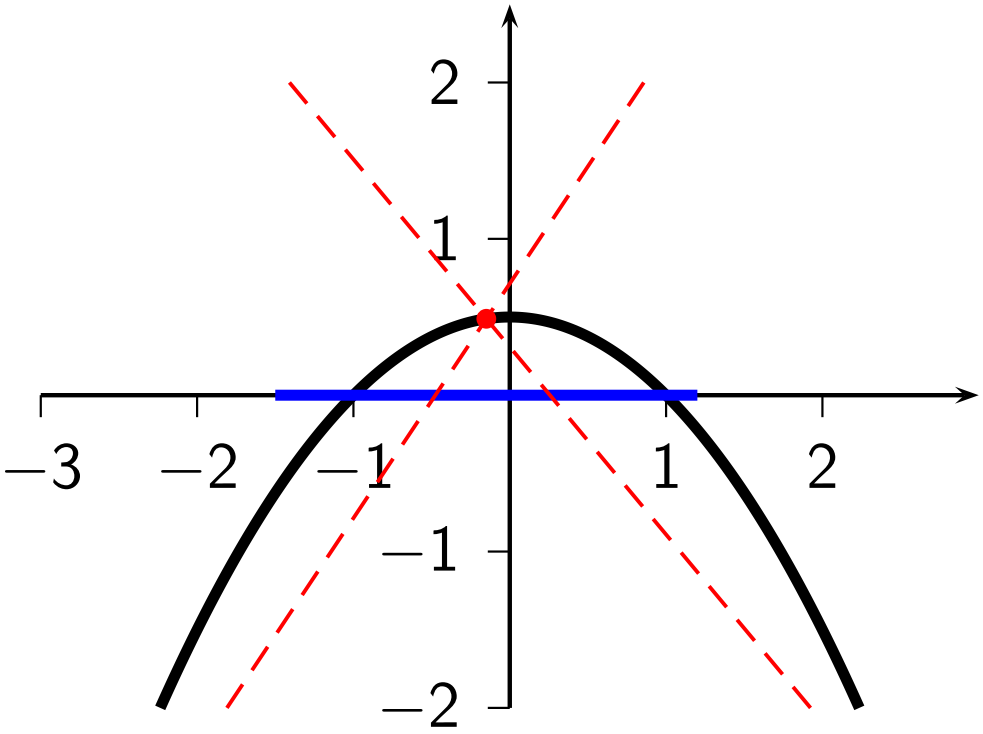
Interval Newton method

Illustration of the algorithm

Non monotonic ?

$\Rightarrow$ Extended interval arithmetic

$\Rightarrow$ Two recursive calls



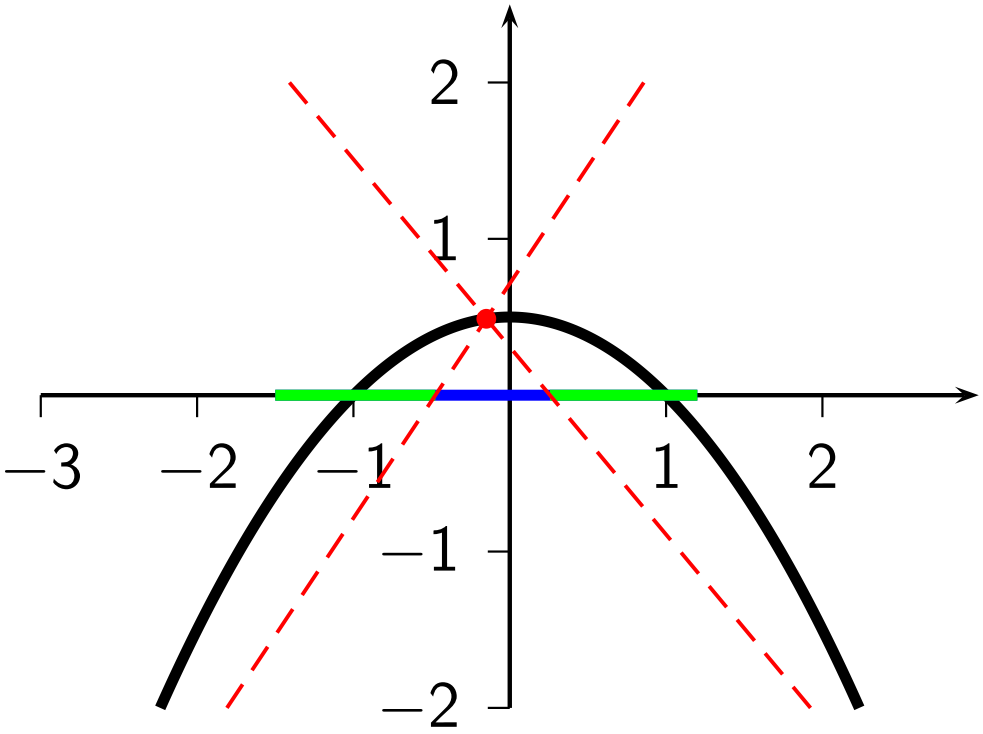
Interval Newton method

Illustration of the algorithm

Non monotonic ?

$\Rightarrow$ Extended interval arithmetic

$\Rightarrow$ Two recursive calls



Outline

Interval Newton on CUDA

Interval Newton vs. Parallel Computation

- Sequential method
- Static load balancing not advisable
  - Bisection of the work for the cases
    - $X_{n+1} = X_n$
    - $0 \in F'(X_n)$
  - Idleness for the cases
    - $0 \notin F(X_n)$
    - $X_{n+1} = \emptyset$

$\Rightarrow$ We need dynamic load balancing!

Interval Newton on CUDA

Interval Newton vs. Parallel Computation

- Sequential method
- Static load balancing not advisable
  - Bisection of the work for the cases
    - $X_{n+1} = X_n$
    - $0 \in F'(X_n)$
  - Idleness for the cases
    - $0 \notin F(X_n)$
    - $X_{n+1} = \emptyset$

$\Rightarrow$ We need dynamic load balancing!

Dynamic Load Balancing

- Blocking Queue
- Task Stealing
- Distributed Stacks
- Static Task List

Blocking Queue

- Global Queue
- Mutual exclusion
- Persistent threads

Task Stealing

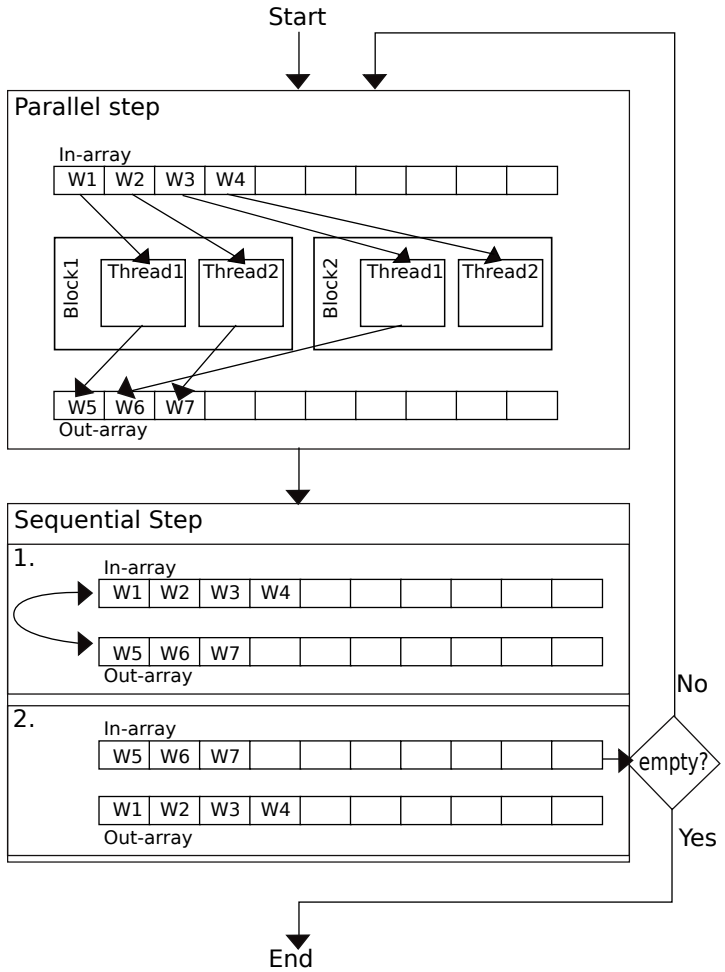
- Lock free
- Block local queue
- Steal tasks from other blocks in case of an empty queue
- Persistent threads

Distributed Stacks

- Lock free
- Block local stack
- Load balancing only between threads of a block
- Static distribution of the workload at the beginning
- Persistent threads

Static Task List

- Lock free
- Two Arrays
  - In-array (read)
  - Out- array (write)
- Swap arrays after each iteration
  - Terminate kernel
  - Initialize new kernel with swapped arrays
    - Number of blocks depends on array size



Outline

Test System

- Intel Xeon E5504 2.00 GHz
- 8 GB RAM
- Debian 64 Bit Linux
- NVIDIA Tesla C2070
  - Compute capability 2.0
  - 14 streaming multiprocessors
    - 32 cores
  - 6 GB RAM
  - 515 GigaFLOPS (double)
  - 1.03 TFLOPS (single)
- NVCC 3.2
- GNU gcc 4.4.5
- Boost 1.46.1
- filib++ 3.0.1

Block - Grid Ratio

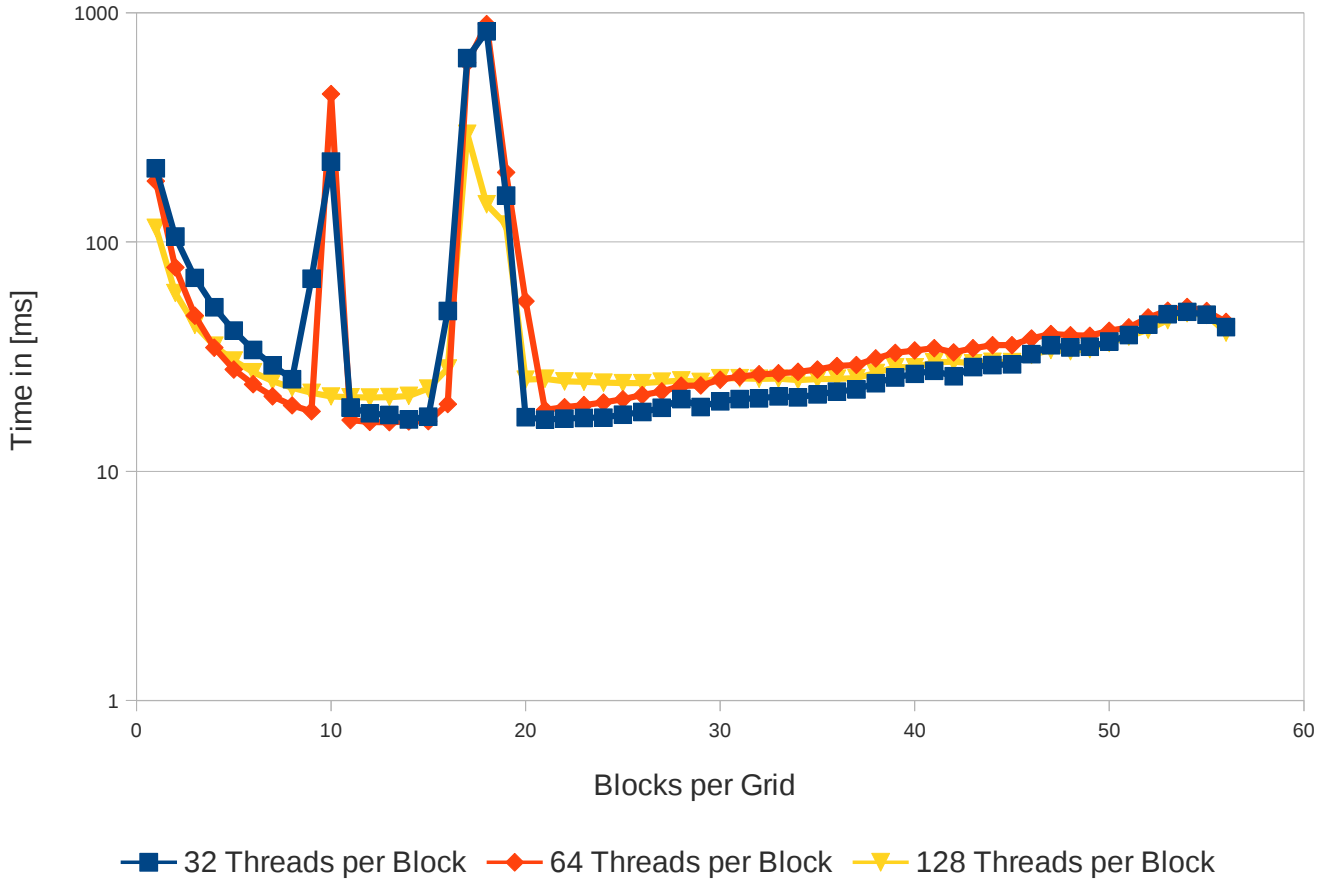
- NVCC `-ptxas-options=-v`

Method	Register/Thread	Shared Memory/Block
BlockingQueue	63	8240 Byte
TaskStealing	63	8240 Byte
DistributedStacks	63	32784 Byte
StaticTaskList	59	0 Byte

- Compute Capability 2.0
  - 32768 Registers
  - 48 KB Shared Memory

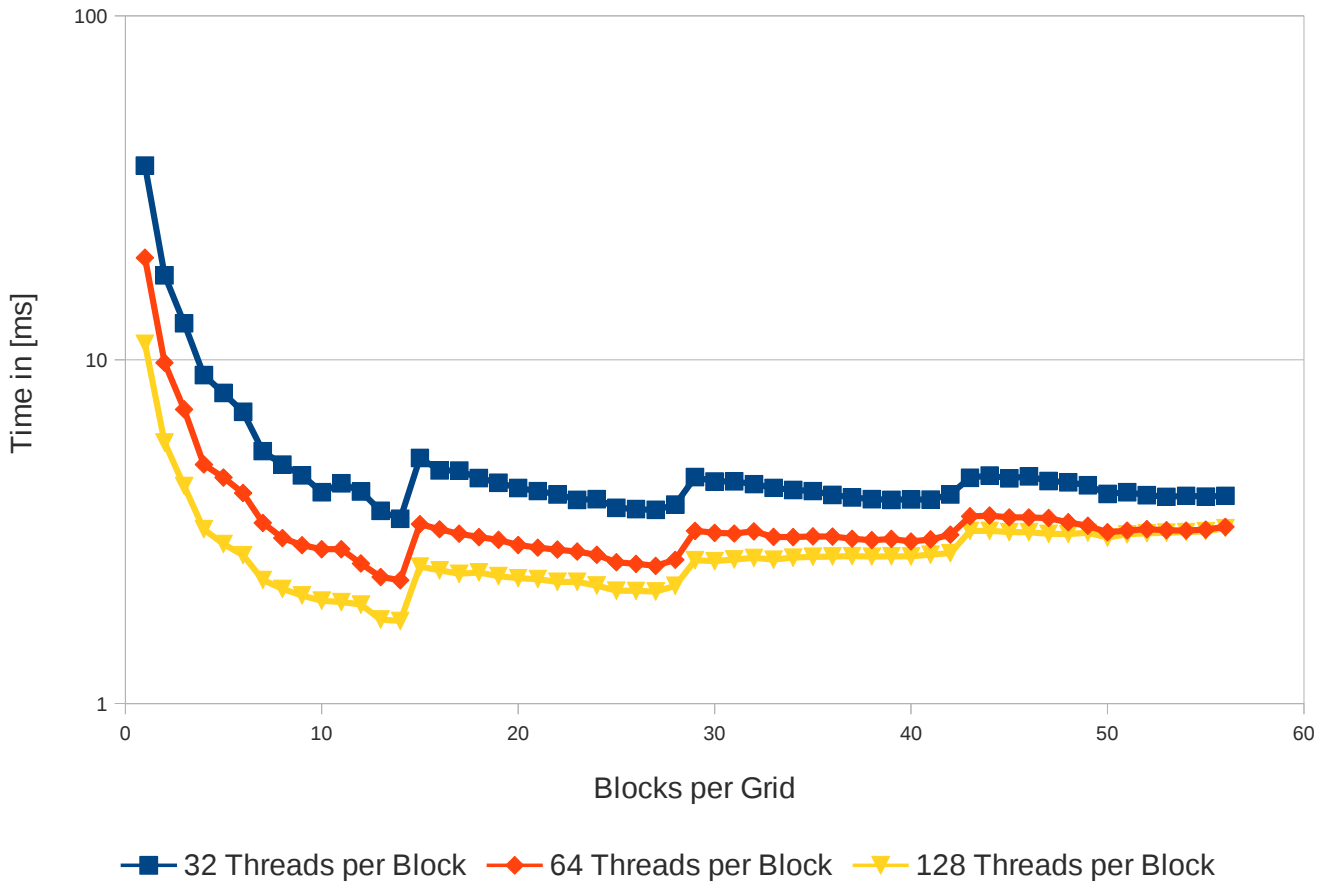
Block - Grid Ratio

Blocking Queue



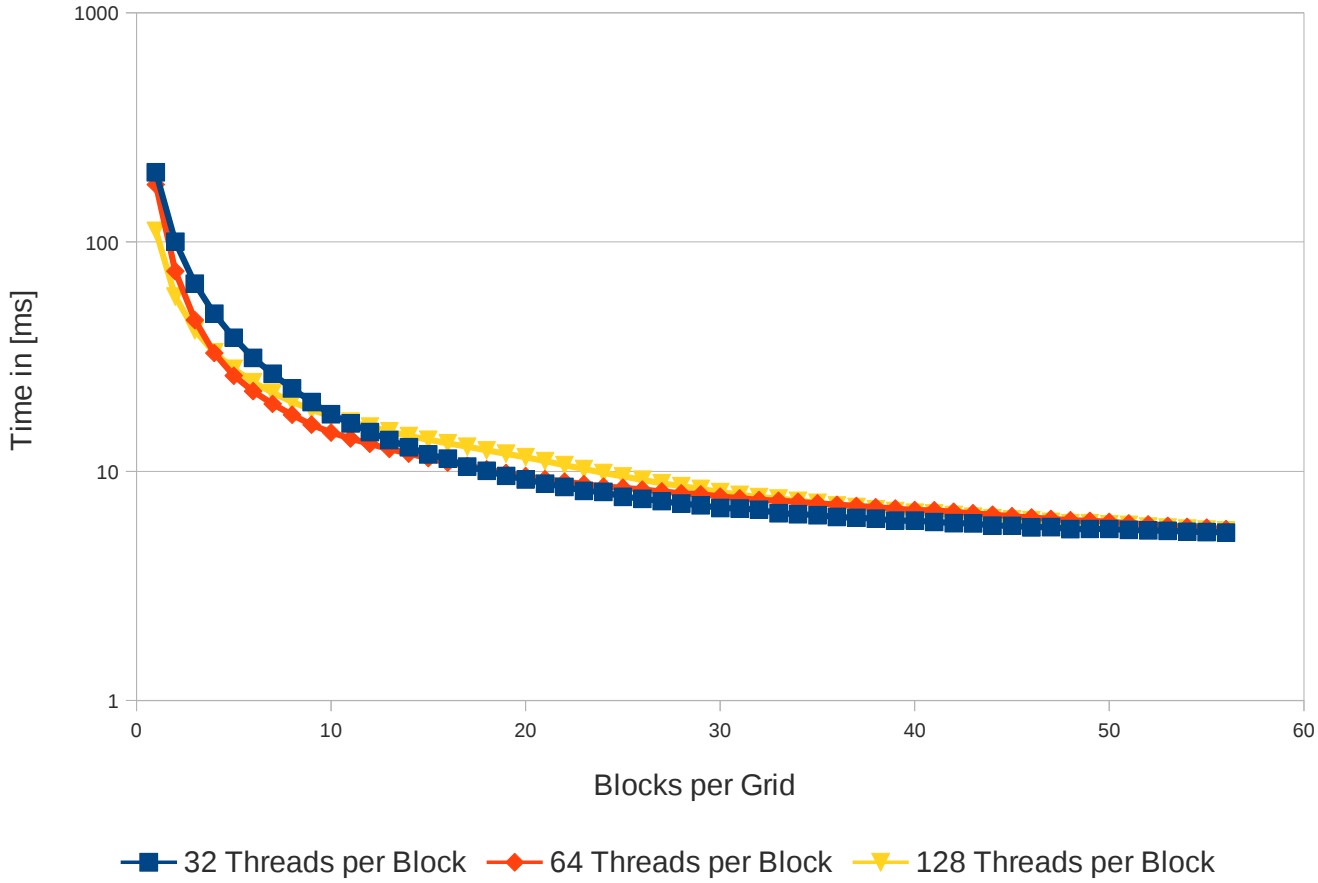
Block - Grid Ratio

Distributed Stacks



Block - Grid Ratio

Task Stealing



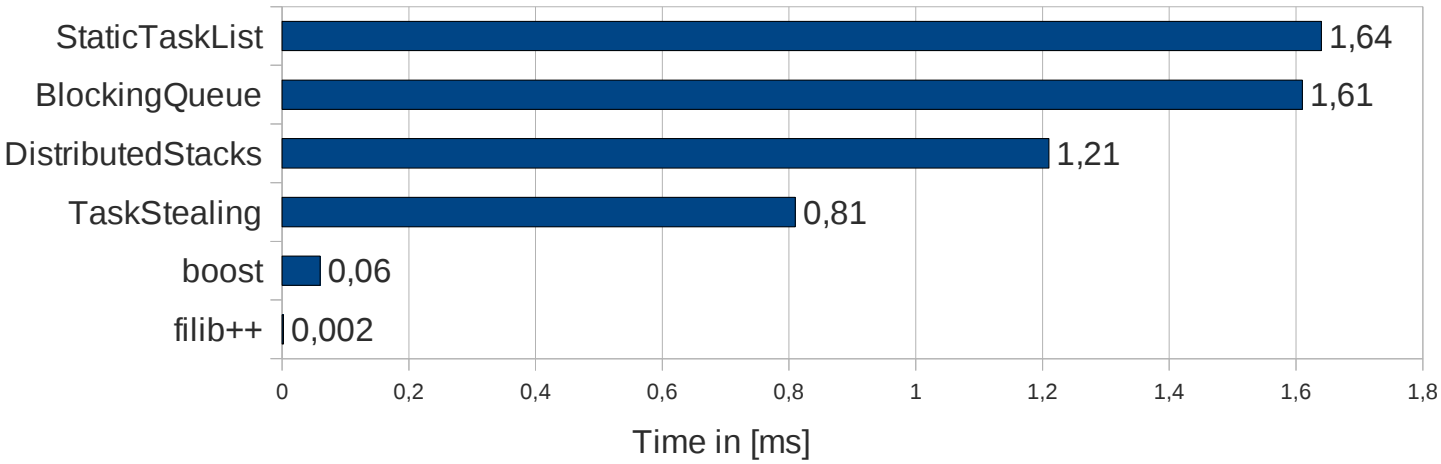
Block - Grid Ratio

Method	Blocks per grid	Threads per block
BlockingQueue	14	64
TaskStealing	28	64
DistributedStacks	14	128
StaticTaskList	-	32

Run Time

$\sinh x$

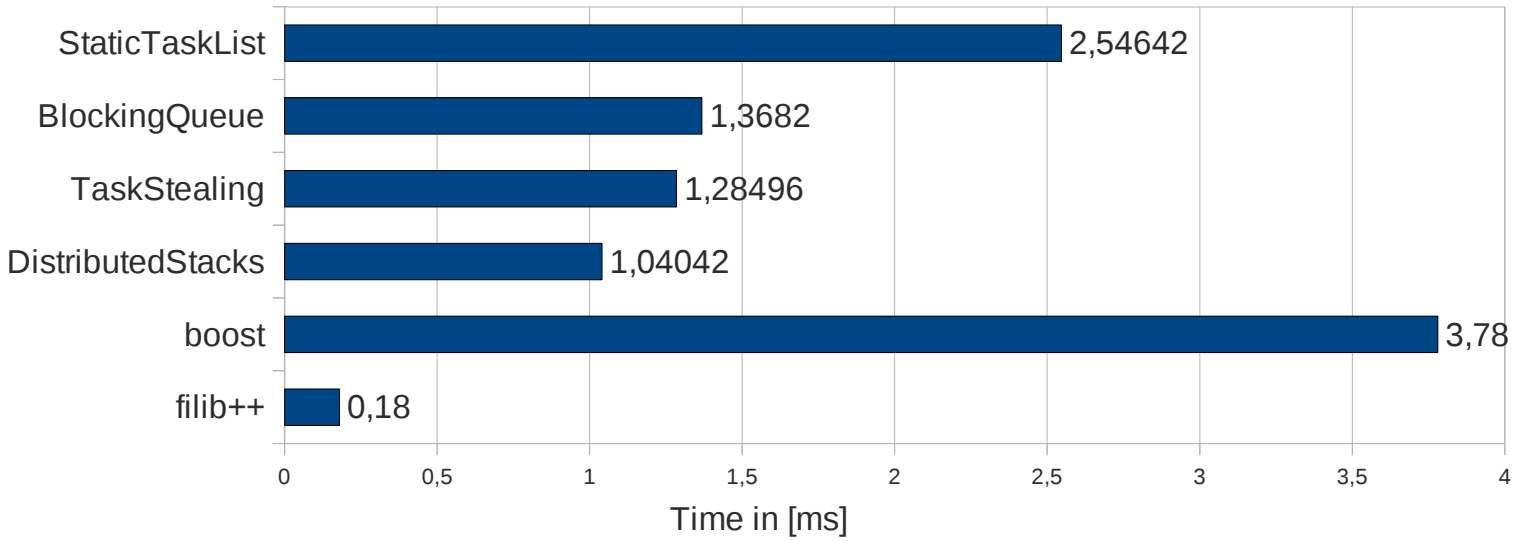
$X_0 = [-1, 1]$, $\epsilon = 10^{-12}$



Run Time

$$\sin x - \frac{x}{100}$$

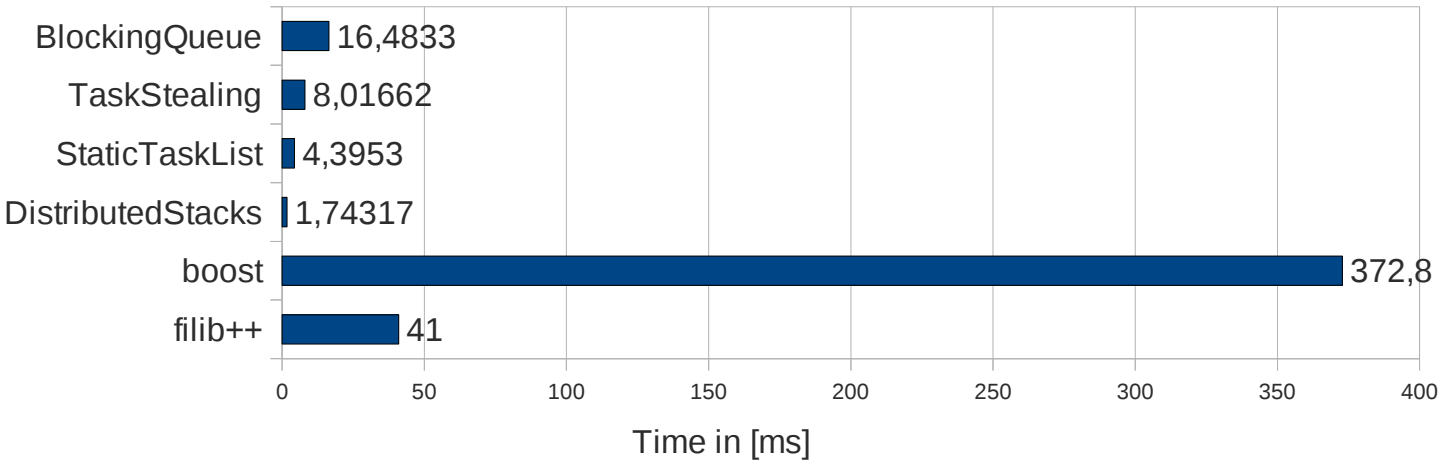
$X_0 = [0, 100]$, $\epsilon = 10^{-12}$



Run Time

$$\sin x - \frac{x}{10000}$$

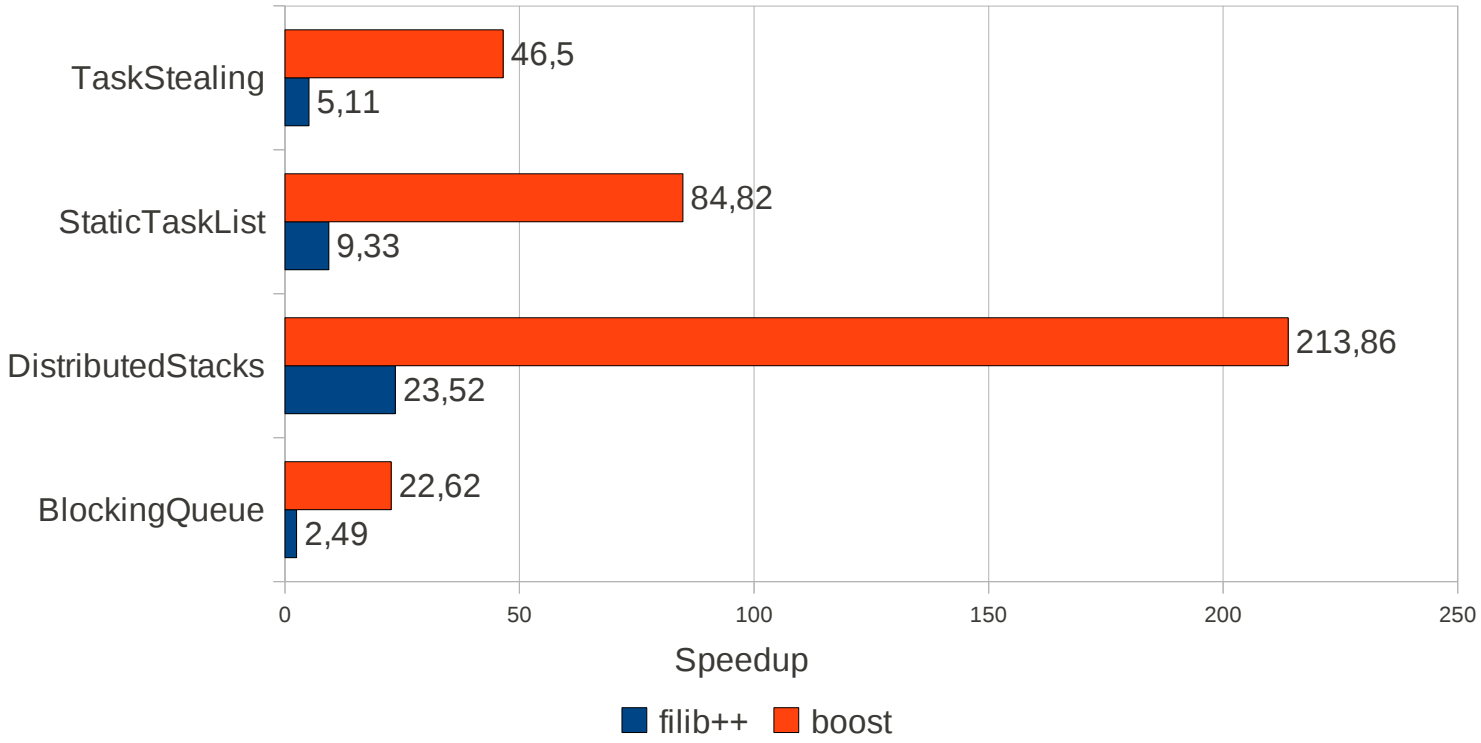
$X_0 = [0, 10000]$, $\epsilon = 10^{-12}$



Speedup

$$\sin x - \frac{x}{10000}$$

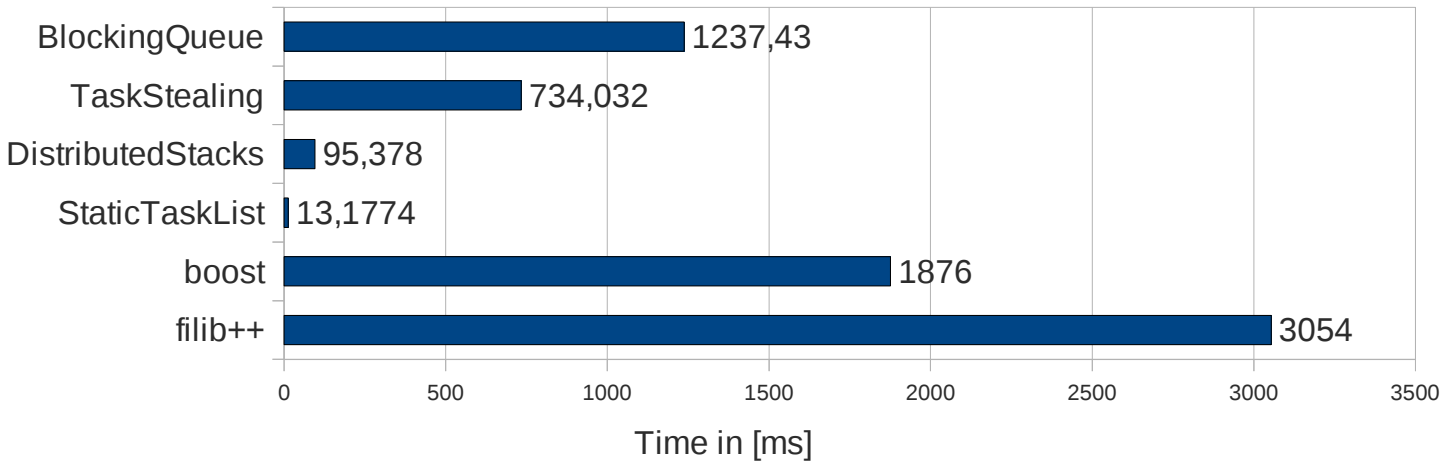
$X_0 = [0, 10000]$, $\epsilon = 10^{-12}$



Run Time

$$\sin \frac{1}{x}$$

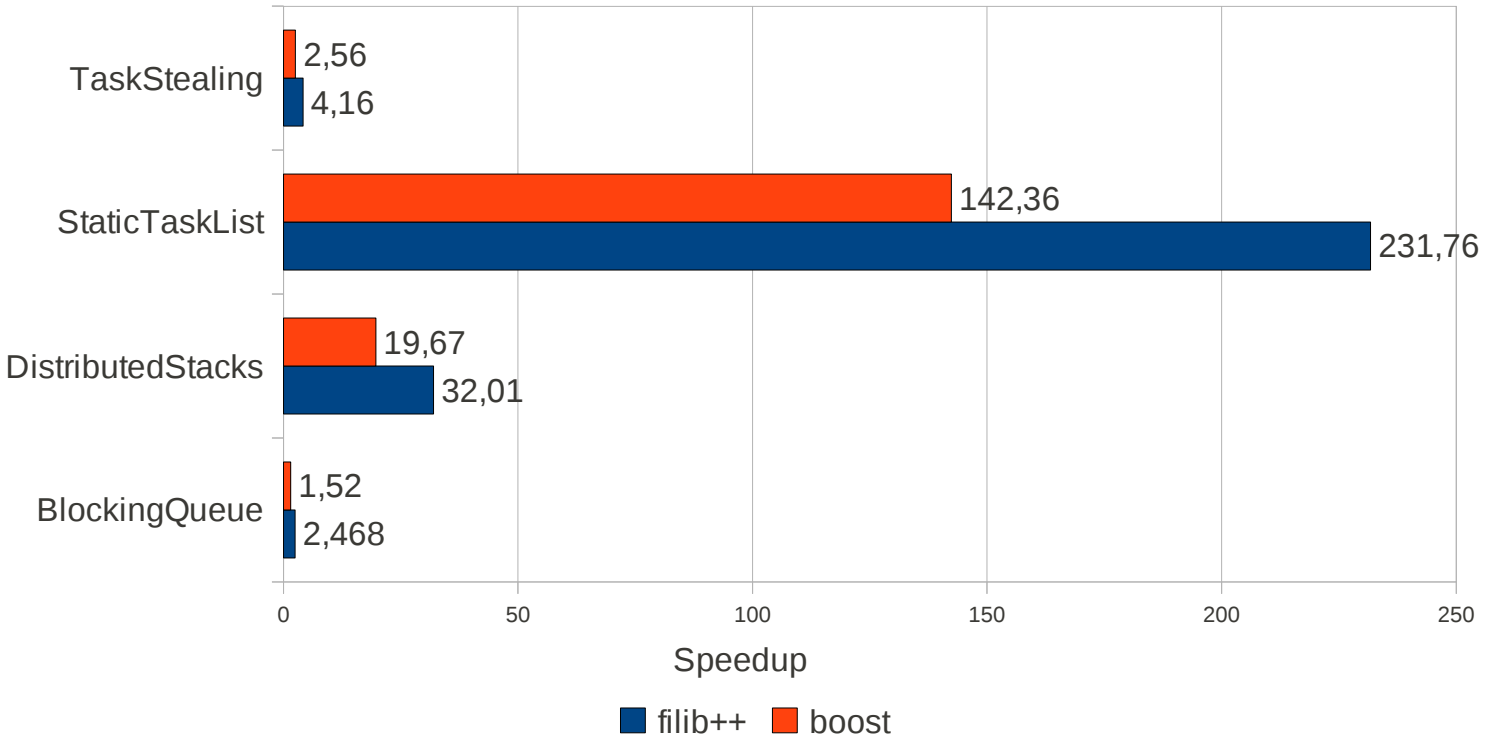
$X_0 = [0.00001, 15]$, $\epsilon = 10^{-12}$



Speedup

$$\sin \frac{1}{x}$$

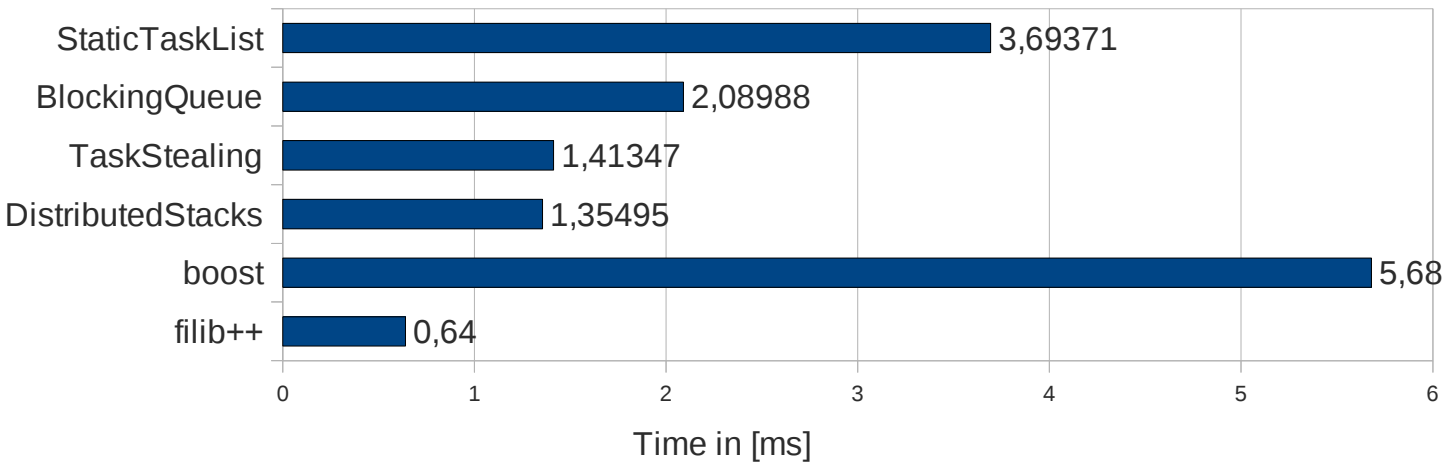
$X_0 = [0.00001, 15]$, $\epsilon = 10^{-12}$



Run Time

$$(3x^3 - 5x + 2) \cdot \sin^2 x + (x^3 + 5 \cdot x) \cdot \sin x - 2x^2 - x - 2$$

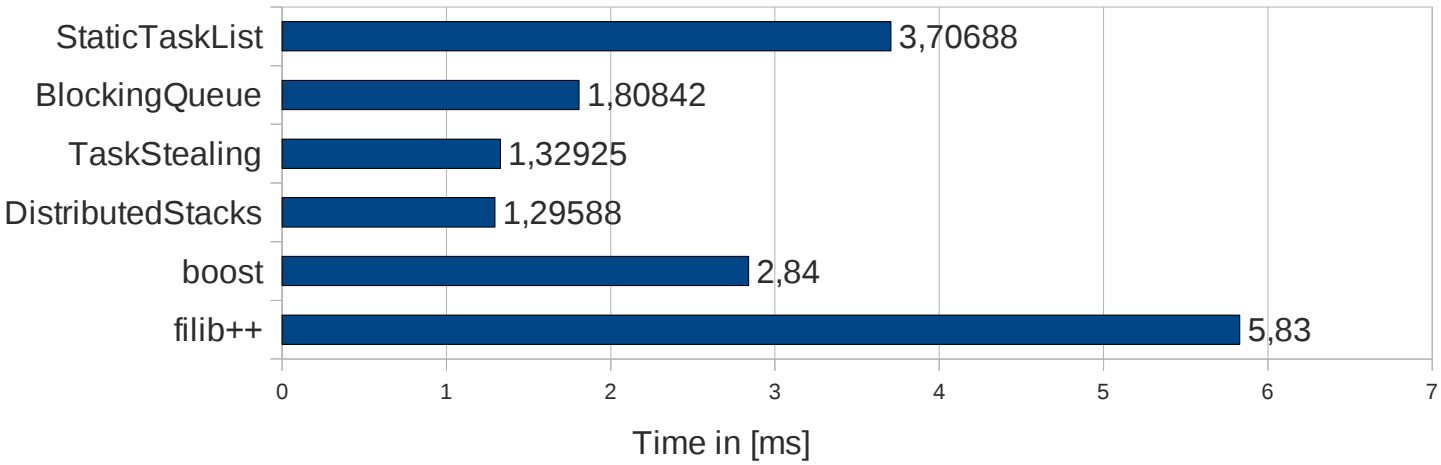
$X_0 = [-10, 10]$, $\epsilon = 10^{-12}$



Run Time

$$x^{14} - 539.25 \cdot x^{12} + 60033.8 \cdot x^{10} - 1.77574e^{6} \cdot x^{8} + 1.70316e^{7} \cdot x^{6} - 5.50378e^{7} \cdot x^{4} + 4.87225e^{7} \cdot x^{2} - 9.0e^{6}$$

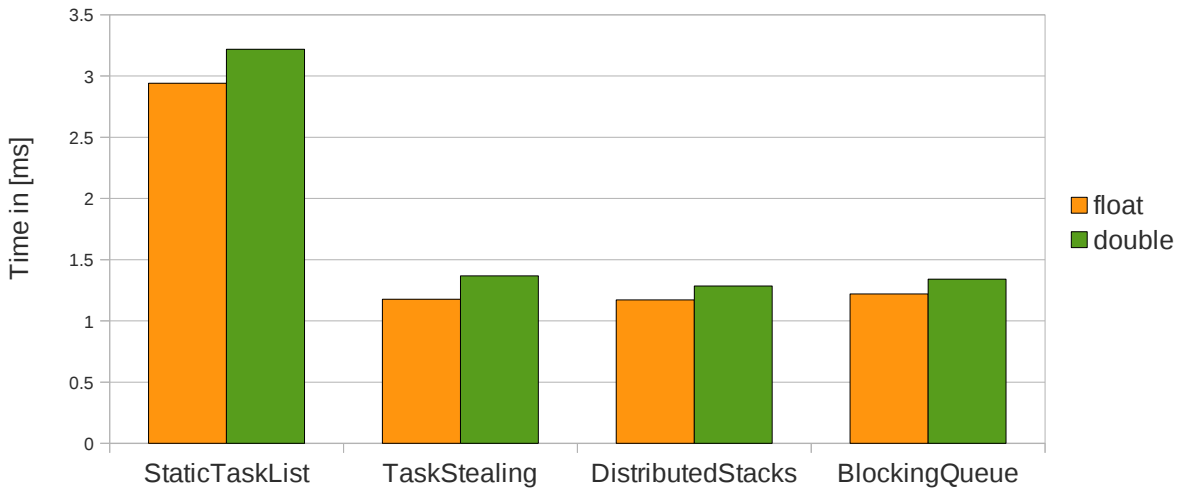
$X_0 = [-20, 20]$, $\epsilon = 10^{-12}$



Precision vs. Run Time

$$(3x^3 - 5x + 2) \cdot \sin^2 x + (x^3 + 5 \cdot x) \cdot \sin x - 2x^2 - x - 2$$

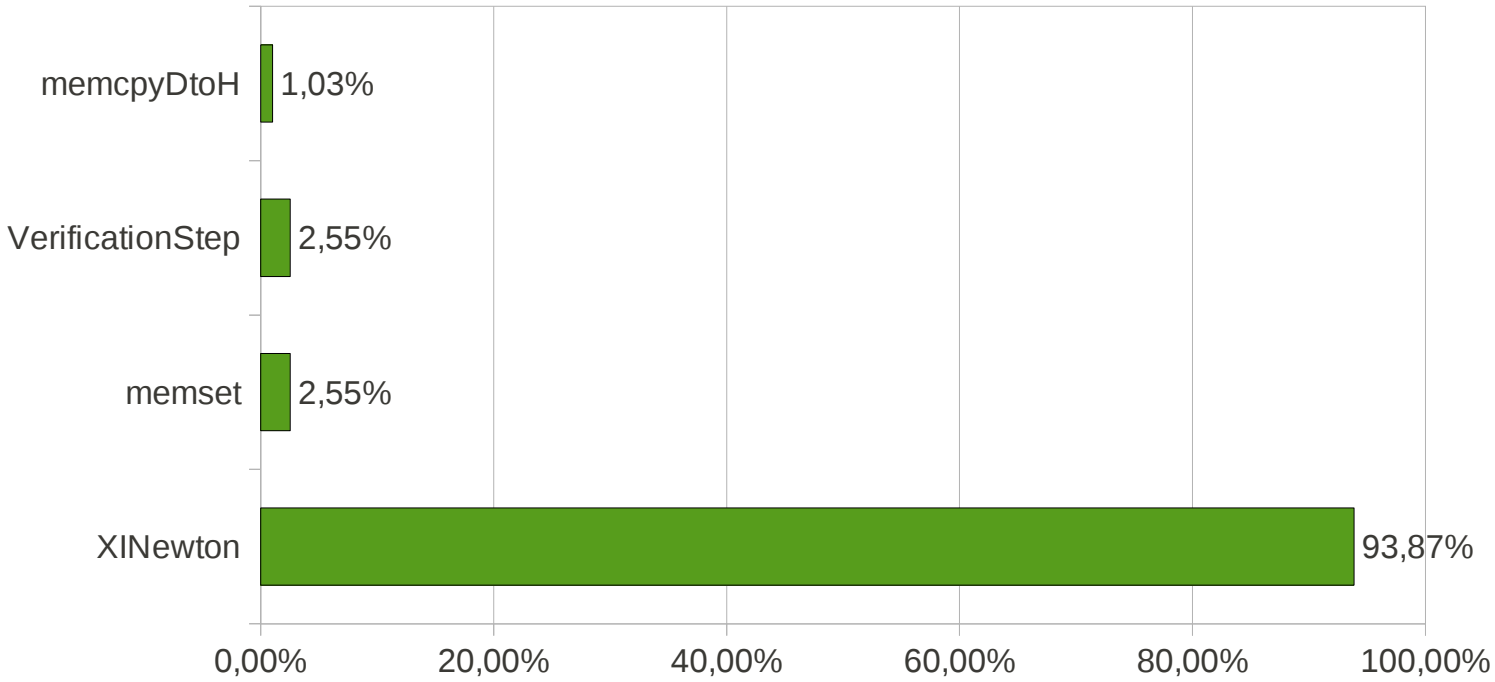
$X_0 = [-10, 10]$, $\epsilon = 10^{-6}$



Workload
Distributed Stacks

$\sinh x$

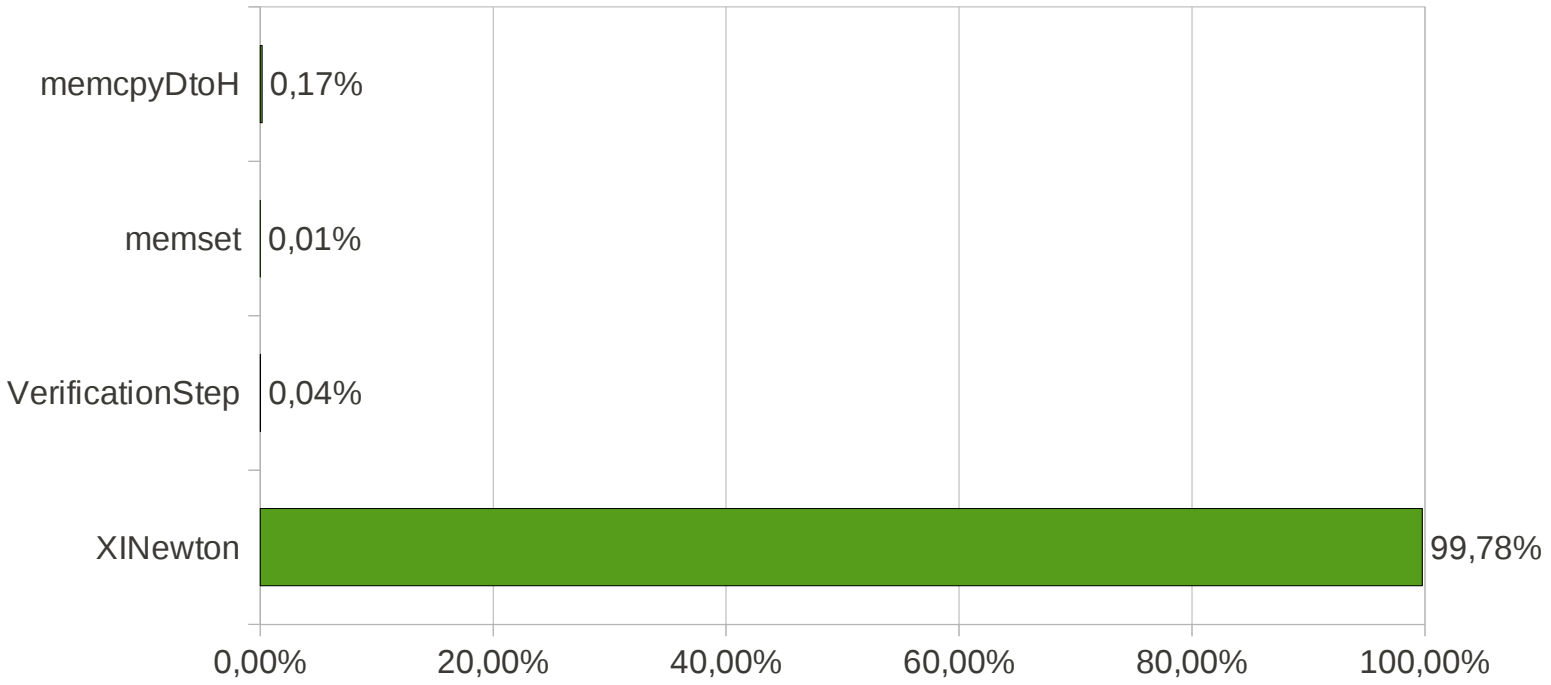
$X_0 = [-1, 1]$, $\epsilon = 10^{-12}$



Workload
Distributed Stacks

$\sin \frac{1}{x}$

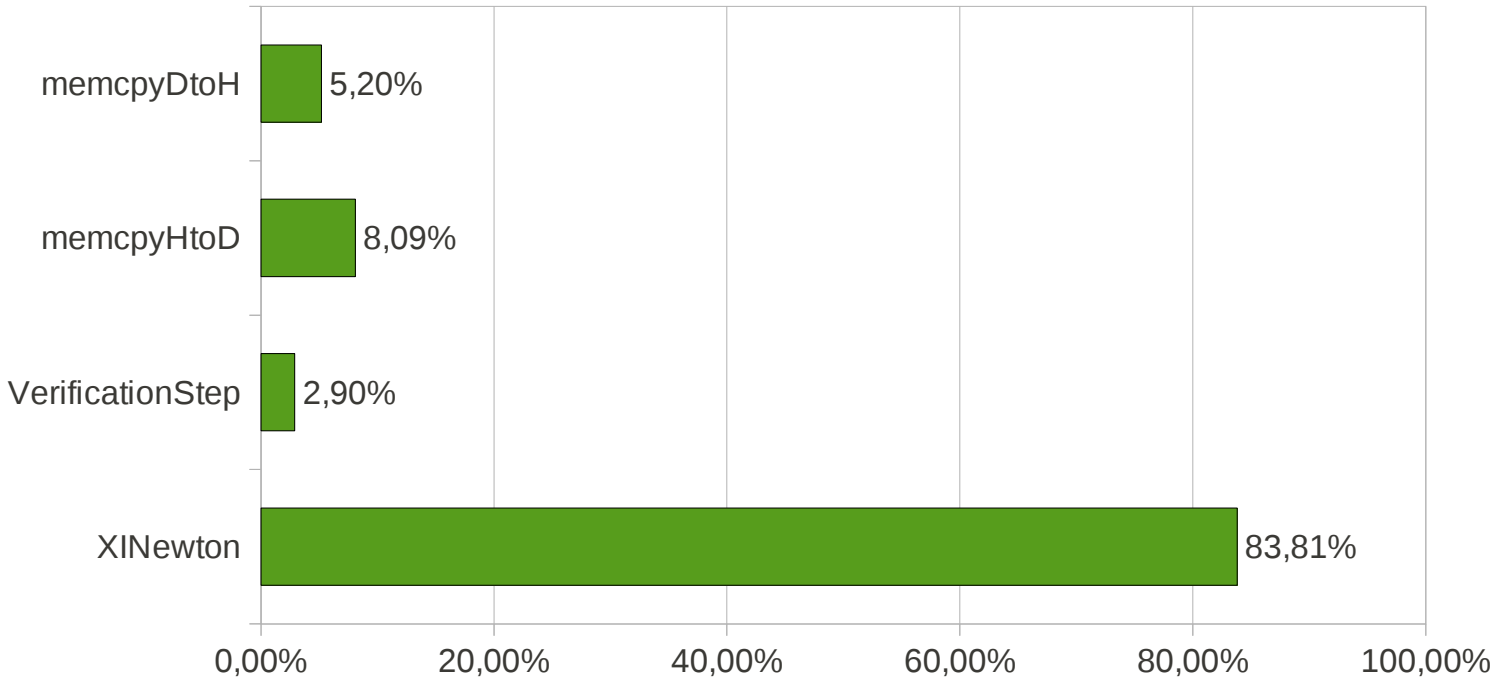
$X_0 = [0.00001, 15]$, $\epsilon = 10^{-12}$



Workload
Static Task List

$\sinh x$

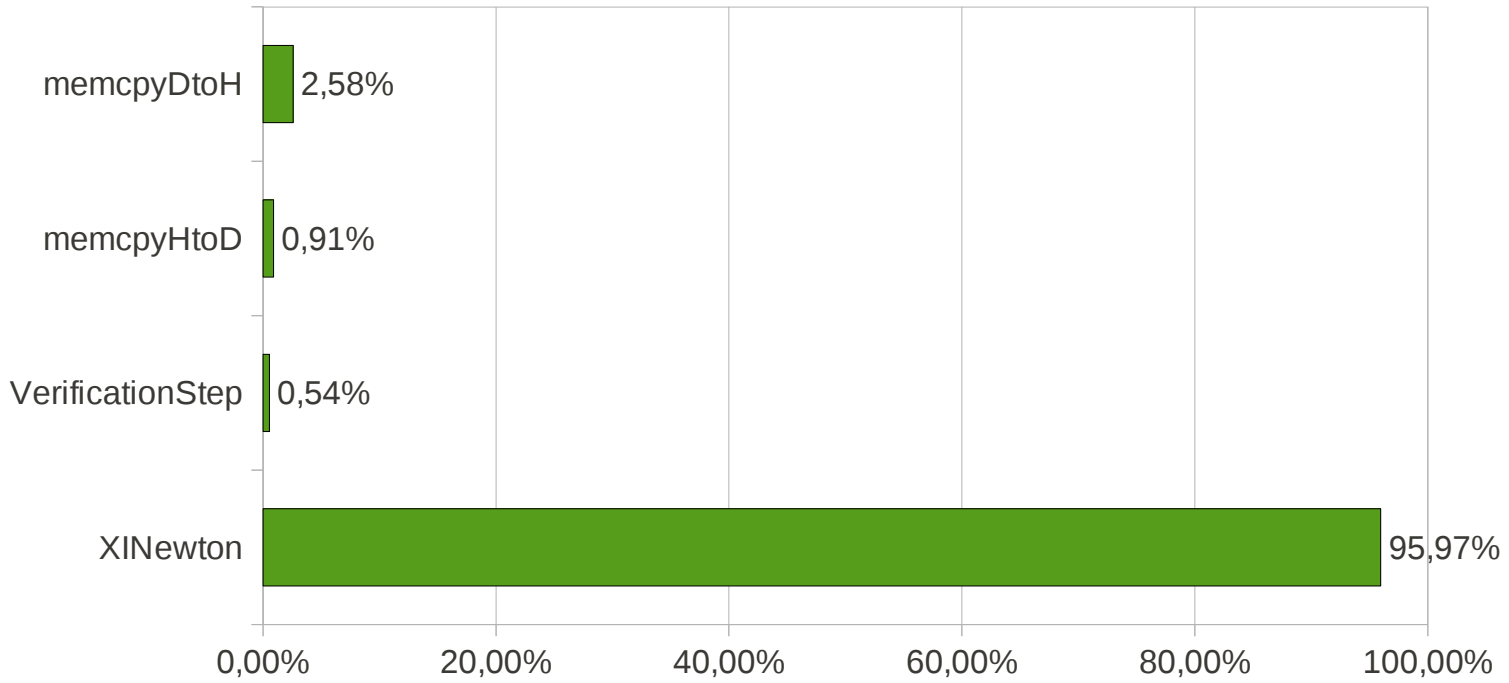
$X_0 = [-1, 1]$, $\epsilon = 10^{-12}$



Workload
Static Task List

$\sin \frac{1}{x}$

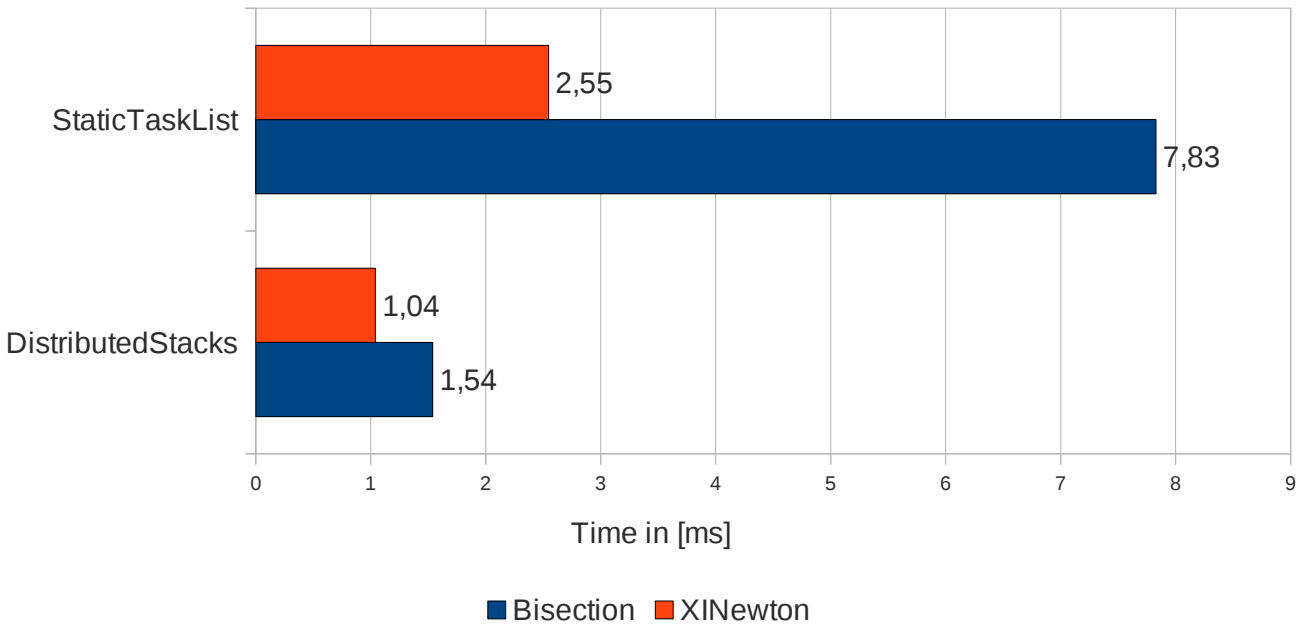
$X_0 = [0.00001, 15]$, $\epsilon = 10^{-12}$



Interval Newton vs. Bisection

$$\sin x - \frac{x}{100}$$

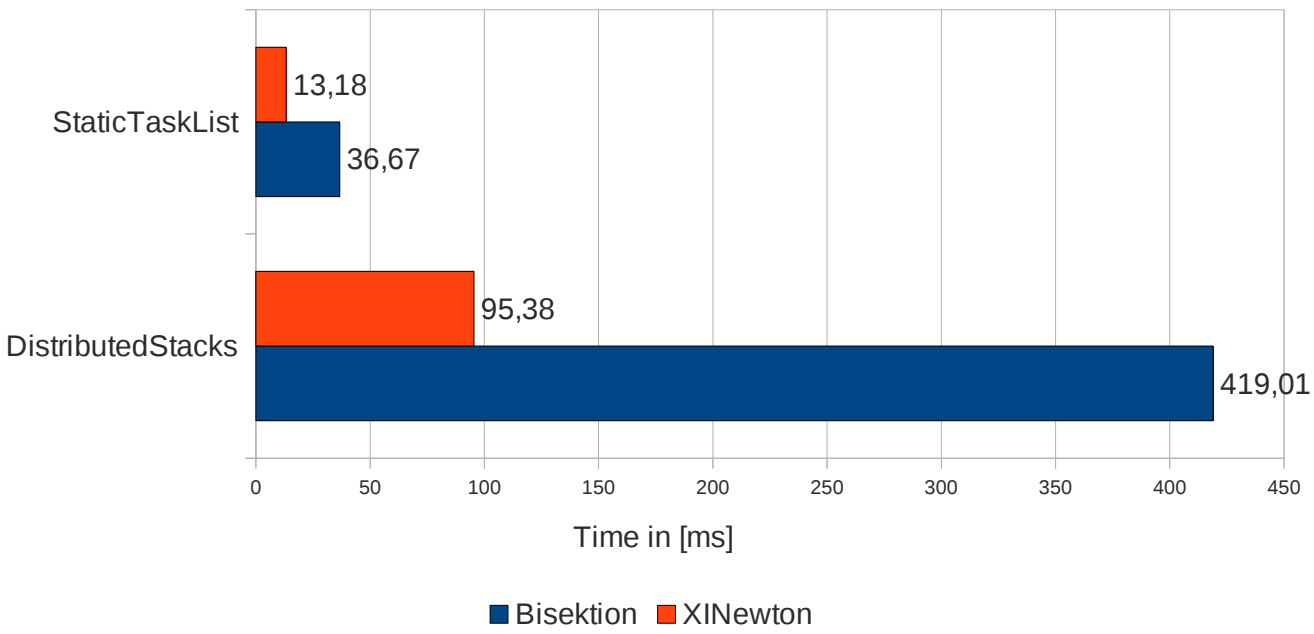
$X_0 = [0, 100]$, $\epsilon = 10^{-12}$



Interval Newton vs. Bisection

$$\sin \frac{1}{x}$$

$X_0 = [0.00001, 15]$, $\epsilon = 10^{-12}$



Outline

Summary

- Parallel interval Newton method
- Many roots
- Global memory not advisable
- Static Task List
  - Slow for simple functions
- Distributed Stacks

Questions ?

Marco Nehmeier

nehmeier@informatik.uni-wuerzburg.de