Data-Flow Awareness in Parallel Data Processing

D. Bednárek, J. Dokulil*, J. Yaghob, F. Zavoral

Charles University Prague, Czech Republic

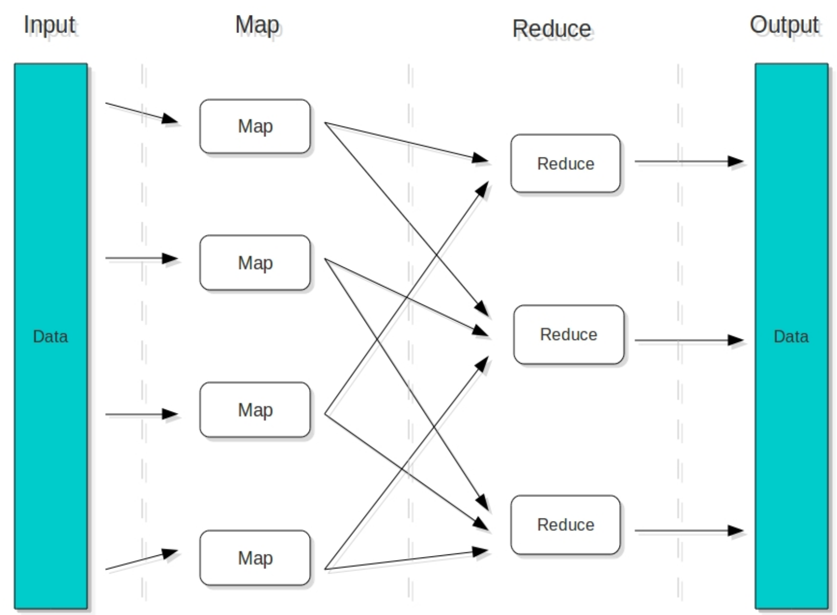
*University of Vienna, Austria

6th International Symposium on Intelligent Distributed Computing

September 24-26, 2012, Calabria

Problem

- Processing large data collections (GB, TB)
  - unstructured data, BigData
  - data streams



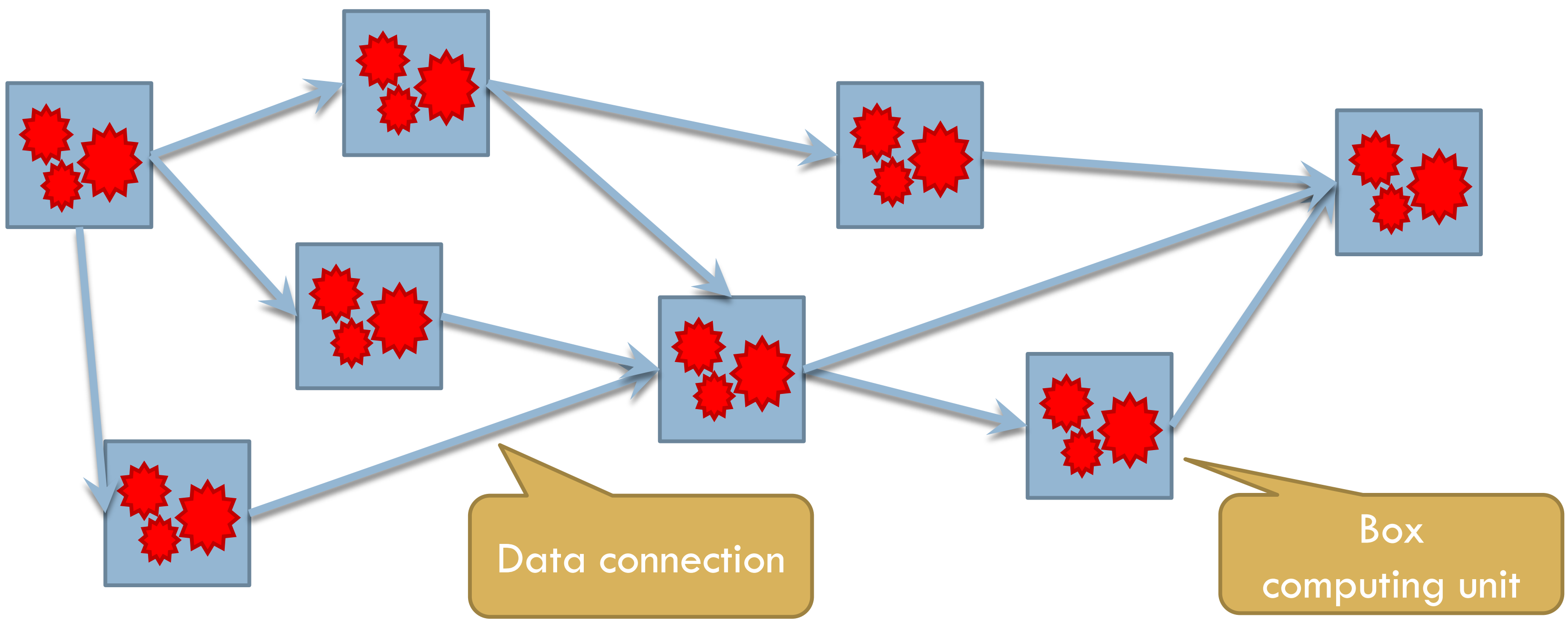
- Cloud Computing
  - Hadoop
  - Map-Reduce paradigm
  - not sufficient for nonlinear computations
    - matrix inversion, fluid dynamics equations, data joins

Problem

- Parallel processing
  - Threading Building Blocks
    - shared memory parallel processing
  - OpenMP
    - mathematical computations
  - MPI
    - distributed computing
- Significant knowledge required
    - synchronization, cache hierarchy, technical details

Bobox

- Framework for parallel programming
- Main goal: simplify writing parallel programs
- Targeted to data-intensive applications
- Technical details handled by the framework



Data flow awareness

- Data-flow awareness in parallel data processing

- Explicit specification of data flow
  - part of the application algorithm
  - gives the task scheduler more precise info
  - improves the quality of task scheduling

$\Rightarrow$ better performance

Scheduler

- Scheduler - selection of the next task
- TBB, OpenMP, or MPI
  - no information on data flow
  - difficult to optimize their strategy
- Bobox model
  - abstraction of data flow known to the scheduler
    - decisions are data-flow aware
  - box - basic computational component
    - zero or more inputs and outputs in the pipeline
  - via - one link in the pipeline
    - connects one output to one or more inputs of other boxes
  - envelope - the smallest unit of data

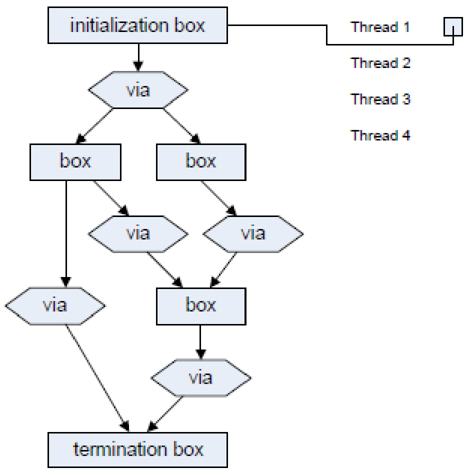
Scheduling in Bobox

- Fixed number of threads
- Separate scheduler for each thread
  - one central task scheduler would create a bottleneck
  - immediate queue
    - tasks to be scheduled immediately
    - data are hot in cache
  - stealing queue
    - tasks which are not tightly bound to the thread
- Scheduling algorithm
  - execute the first task in the immediate queue
  - if empty - head of the stealing queue
  - if empty - steals tasks from another scheduler

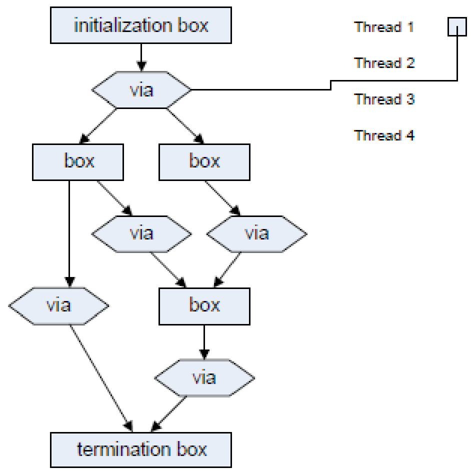
Task enqueueing

- Task enqueueing
  - immediate task - head of the immediate queue
  - relaxed task - end of the stealing queue
  - hard-wired into a box and via framework
- Main reason: cache-awareness of the framework
  - head of the immediate queue should be scheduled immediately by the same thread on the same CPU
  - the data bound to the task is hot in the cache
  - number of memory accesses is minimized
- relaxed task is placed on the tail
  - since the data are not needed to be hot

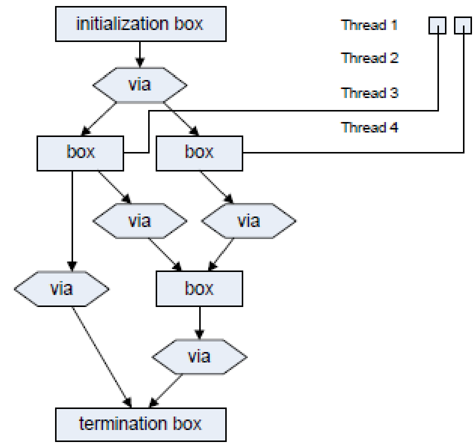
Scheduling example



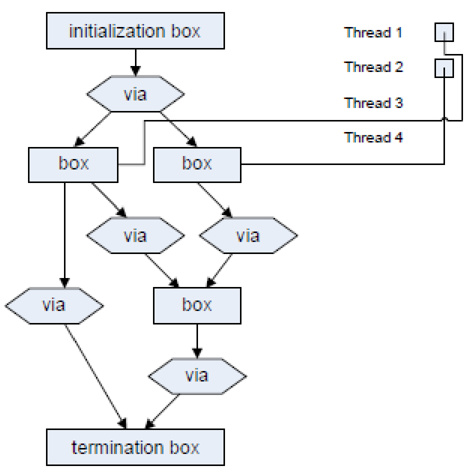
(a) Start



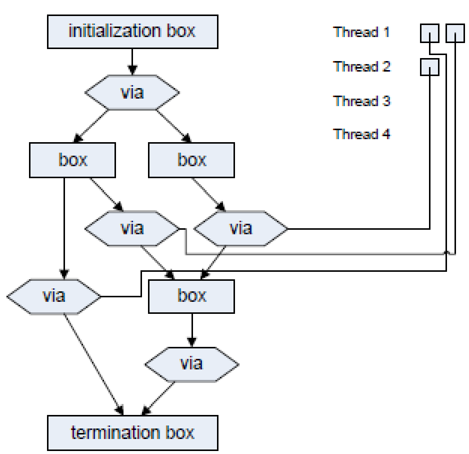
(b) Init finished



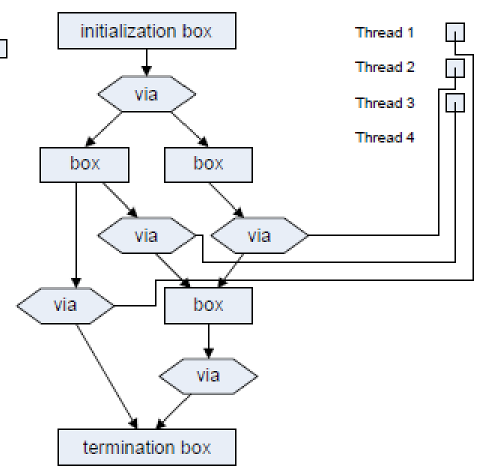
(c) Boxes scheduled



(d) Task stolen

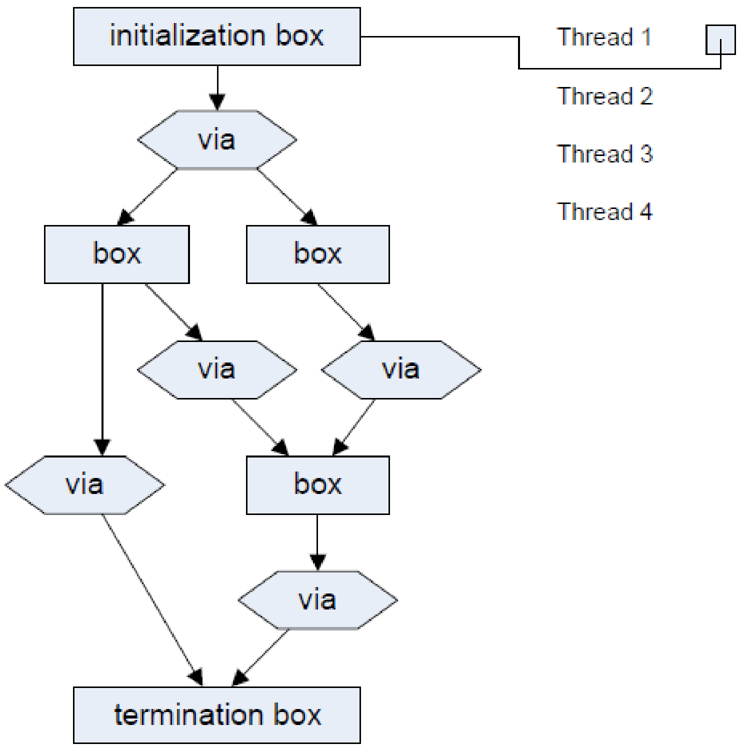


(e) Boxes finished



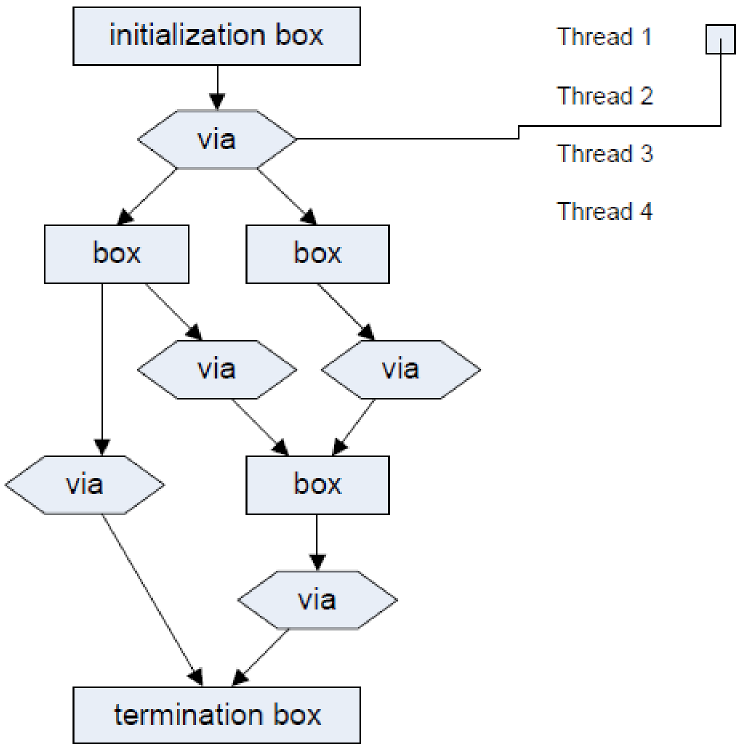
(f) Another task stolen

1. start



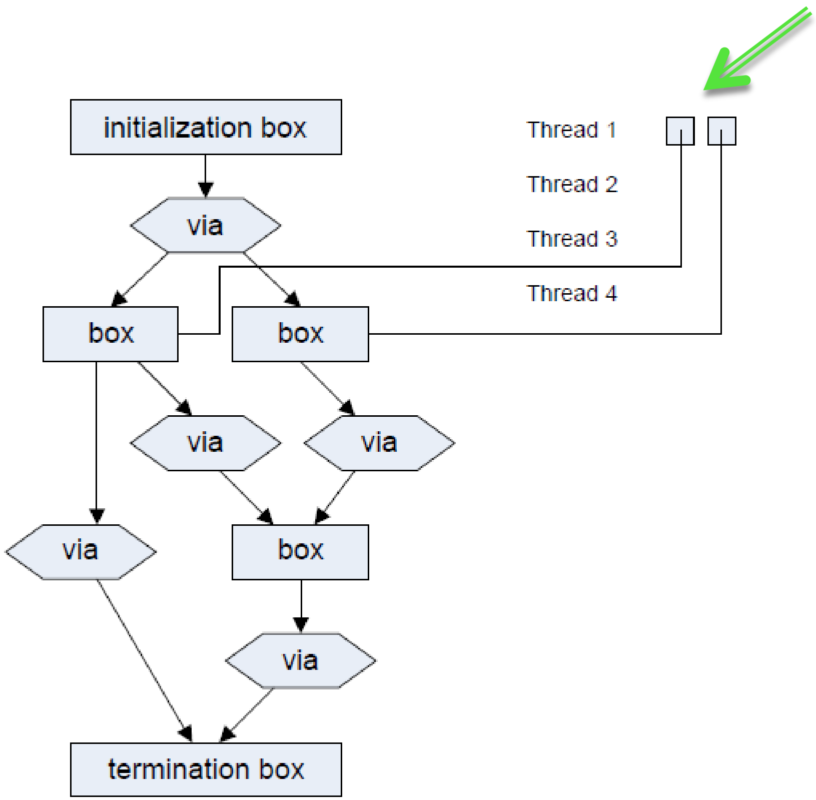
- Bobox model - pipeline
- 4 threads, task queues
- initial box is enqueued

2. initialization finished



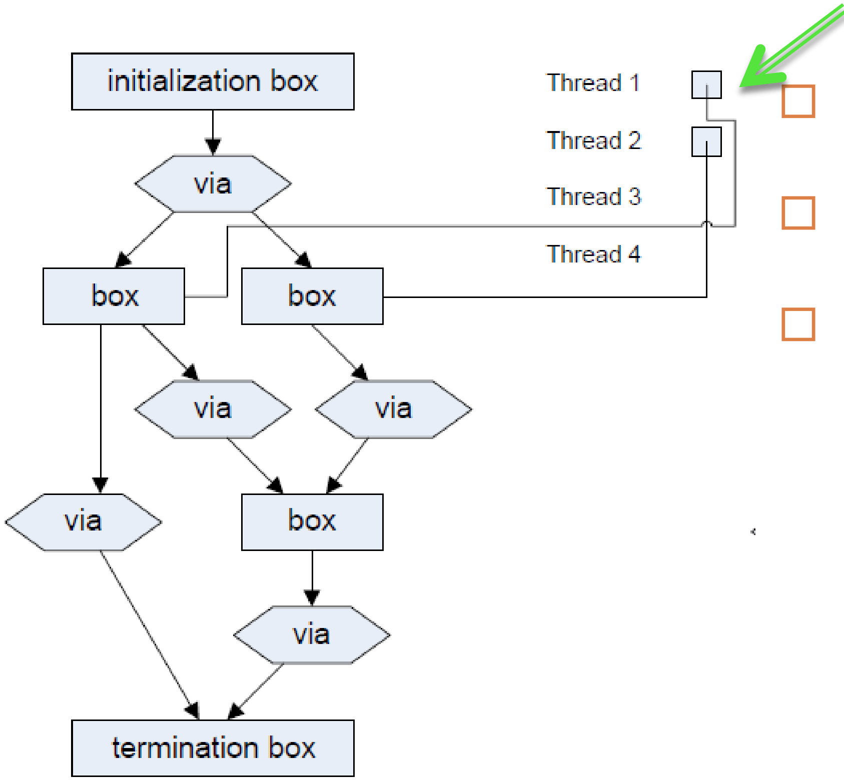
- box produces data
- sends it to the first via
- enqueues the via

3. boxes scheduled



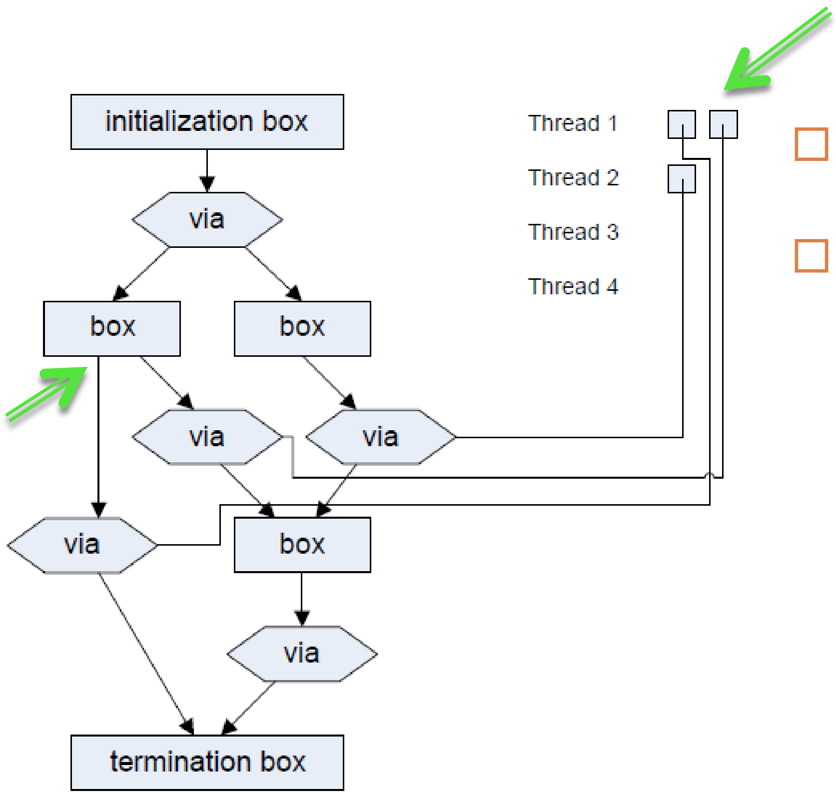
- the via duplicates data
- sends it to the boxes
- enqueues them

4. task stolen



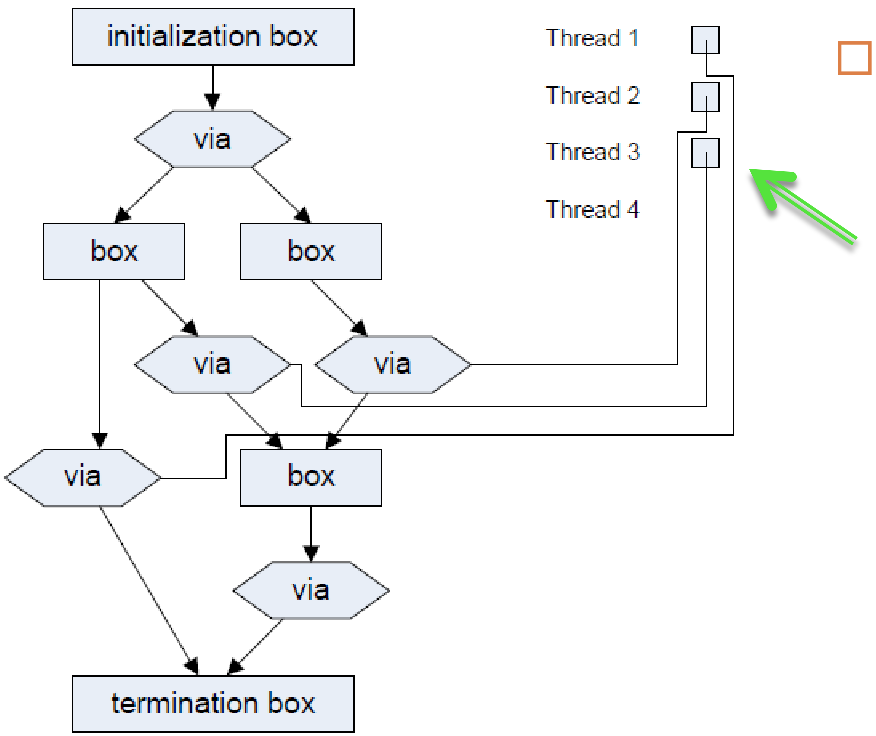
- thread 2 steals one task
- both tasks are invoked
- boxes produce their results

5. boxes finished



- newly created tasks
- enqueued with the same thread

6. another task stolen

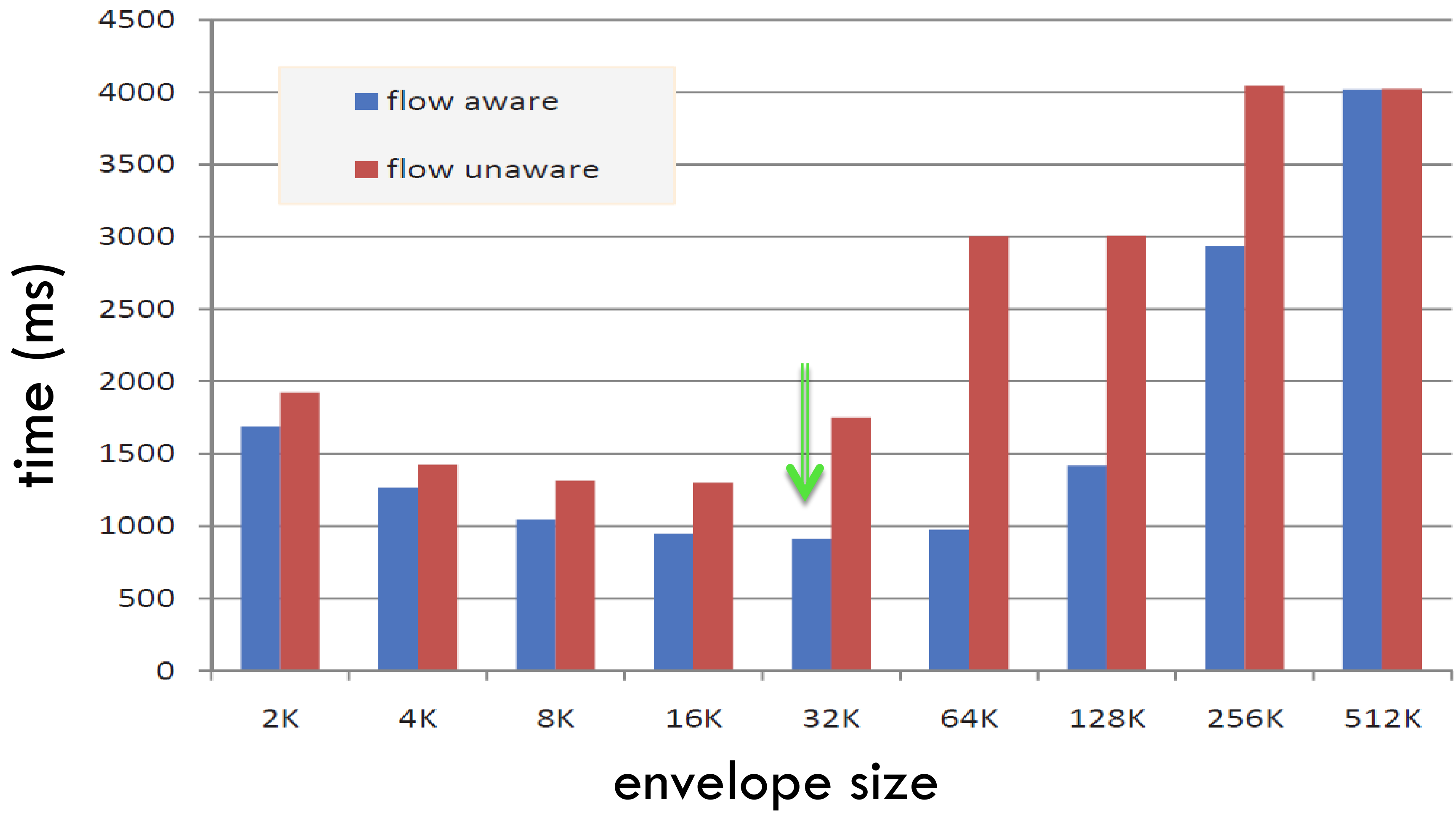


- one task stolen by the thread 3

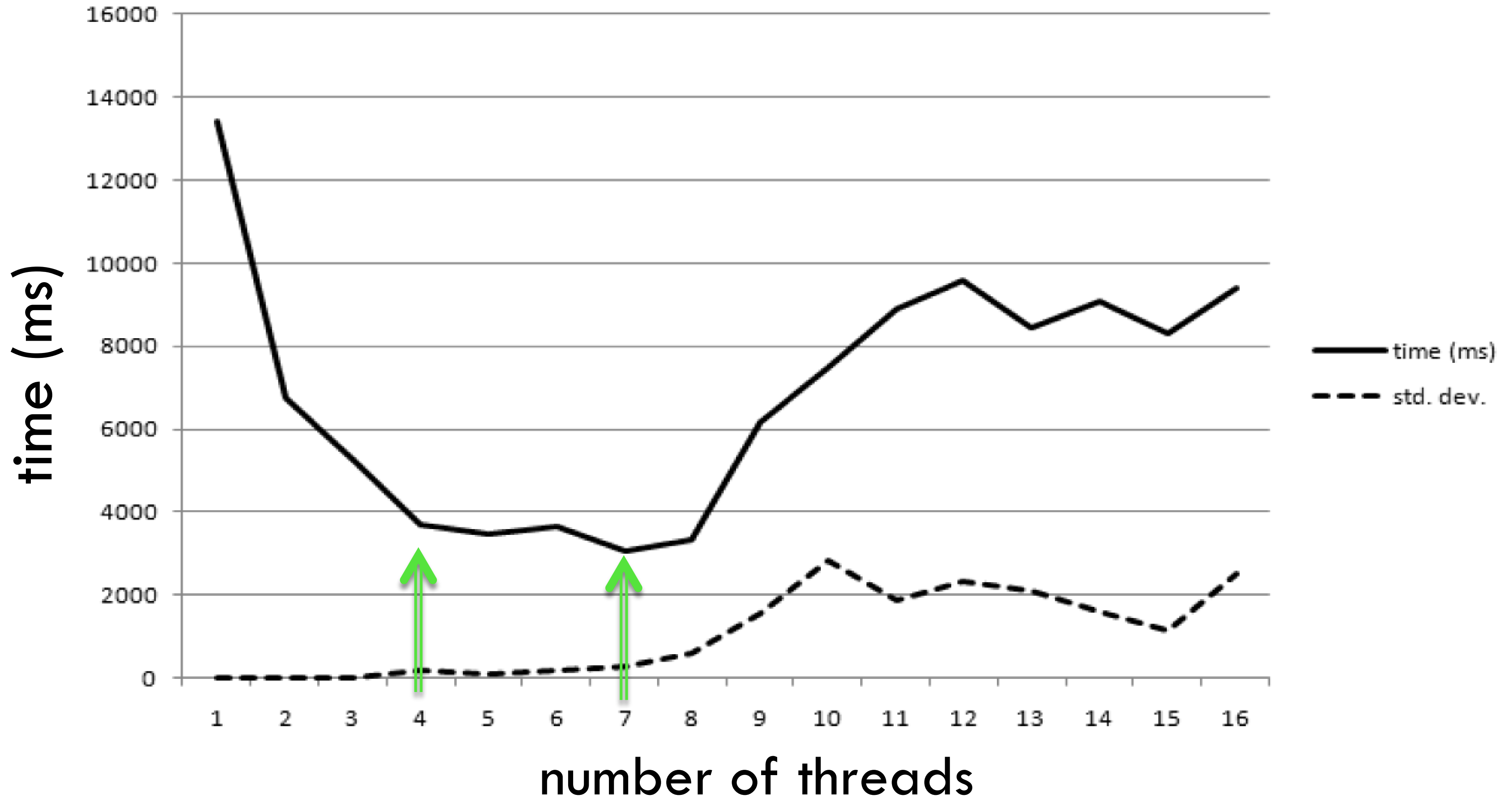
Performance

- DFA and non-DFA (referential) scheduler
- Performance is affected by envelope (data) size
  - data larger than cache $\Rightarrow$ performance decreases
  - several levels of caches $\Rightarrow$ several drops
- Number of threads
  - CPU: 4 cores w/ hyperthreading
  - 2 to 4 threads significantly better than single-threaded
  - best results: 7 threads

Performance / envelope size



Performance / number of threads



Related projects

- Bobox is used in several related projects
  - SPARQL compiler
  - model visualization (graph drawing)
  - semantic processing
  - query optimization
  - data stream processing (astro-informatics)
- Participating universities
  - Charles University Prague (cz)
  - Comenius University Bratislava (sk)
  - University of Vienna (at)

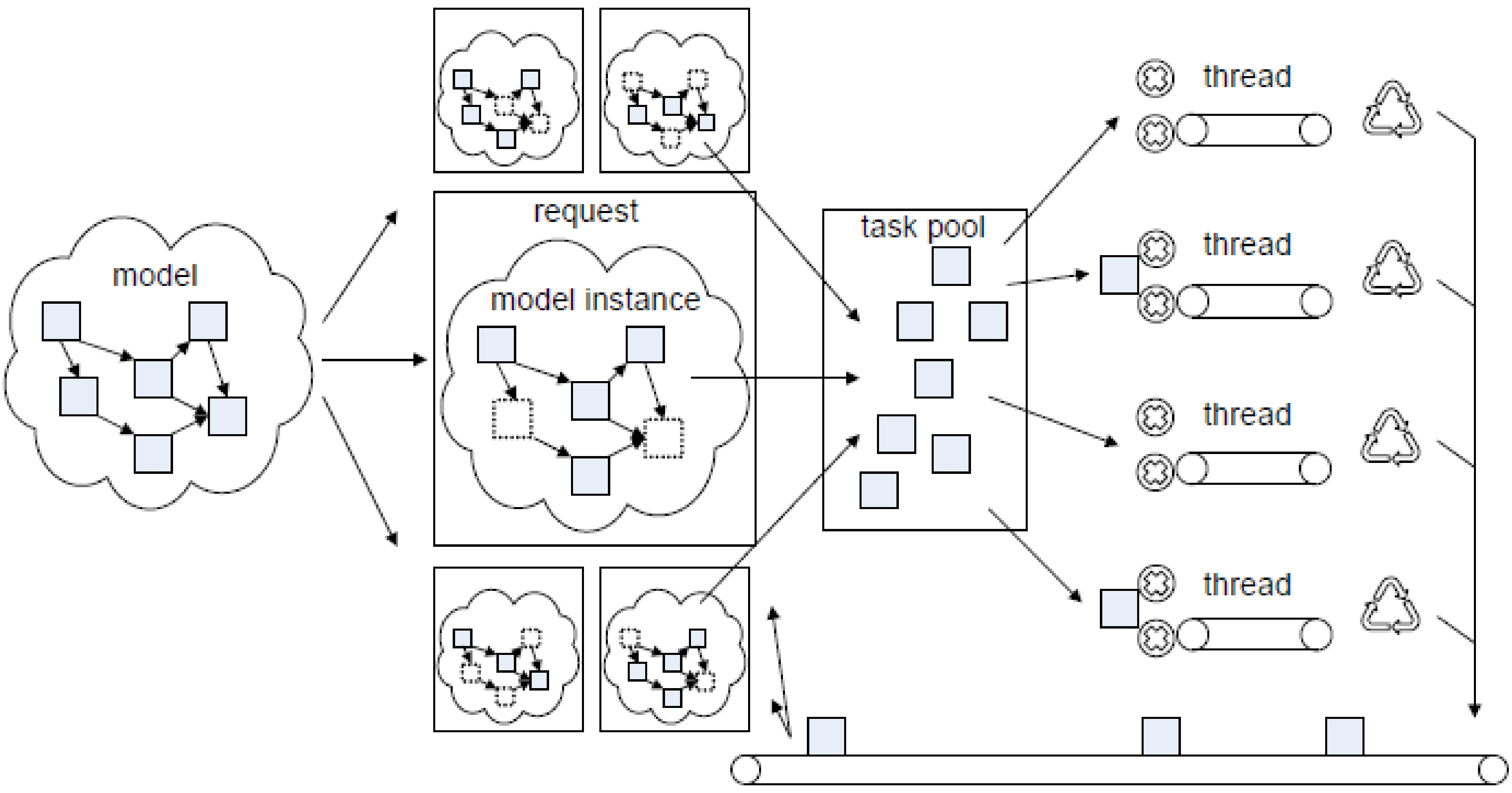
Conclusions

- We have demonstrated the impact of supplying dataflow information to a scheduler
- Advantage of Bobox architecture
  - data-flow information is a part of the application code
- Data-awareness
  - significant performance improvement
  - DFA scheduler required appx. 30% less time

The End

Thank you for your attention

Bobox runtime architecture



Performance / envelope size

□	2K	4K	8K	16K	32K	64K	128K	256K
DFA	1686	1267	1046	947	911	977	1420	2935
ref	1927	1423	1313	1297	1751	3002	3007	4044

Bobox

- Box
  - computation unit
  - basic paradigms
    - task parallelism, non-linear pipeline
  - high-performance messaging
    - communication, synchronization
  - technical details handled by the framework
    - NUMA, cache hierarchy, CPU architecture