The rise of multicore CPUs introduced new challenges in the process of design of concurrent data structures: in addition to traditional requirements like correctness and progress, the scalability is of paramount importance. It is a common opinion that these demands are partially in conflict each other, so that it is necessary to relax the requirements about a behavior identical to the corresponding sequential data structures in order to achieve high performances.

In this work we introduce a relaxed approach for the management of heap based priority queues on multicore CPUs, with the aim to realize a tradeoff between efficiency and sequential correctness. The approach is based on dynamic arrangement of the data structure among the cores, with a step-by-step redistribution procedure that shares information only among cores directly connected in a virtual mesh.

**A HIGH PERFORMANCE DATA STRUCTURE**

Local redistribution of highest priority items (relaxed approach)

No centralized data structure!

N threads are logically organized according to a 2-dimensional periodic mesh $M_2$.

We define a High Performance Heap: a collection of partially ordered binary trees $H_i$ with the max-heap property, where the nodes are connected among them according the mesh $M_2$.

Each thread $P_i$ manages a private sub-structure $H_i$. If $H_i > H_{i'}$, then the item with largest priority $i^1 > i^1'$ is moved forward to a connected thread in the mesh $H_i$ according a producer-consumer protocol, alternatively in the two directions. In this way the critical items with highest priority are passed from thread to thread, an iteration after the other, through all nodes of the mesh, with a better distribution.

**HEAP MANAGEMENT IN MULTICORE ENVIRONMENTS**

Two traditional approaches

Centralized approach: Each thread manages a private sub-structure. Access to the data without synchronization.

Distributed approach: All threads $P_i$ access a single data structure. All basic operations must be carried out in a critical section.

**HEAPS IN COMPUTATIONAL SCIENCE**

- A heap is a dynamical data structures used when there is need to process high priority items (e.g. with large numerical error) produced at run time, with an order depending on the application data and that cannot be predicted.
- Each node is tagged with a priority higher than its children. The node with highest priority $e^*$ is in the root.
- Two efficient operations are defined on a heap: remove(root) and insert(item).
- In many scientific applications, the heap is periodically updated in an iterative section of the algorithms.

**SCALABILITY ANALYSIS AND TEST RESULTS**

In the proposed algorithm, at each iteration, there are no global synchronizations among threads $P_i$ and each of them exchanges data only with the two threads $P_{i-1}$ and $P_{i+1}$, so that the synchronization overhead is $T_0$ = $O(1)$ = constant, because it does not depend on the number of threads $N$.

**Linearizability !!**

**Scalability !!**

**Abstract**

The rise of multicore CPUs introduced new challenges in the process of design of concurrent data structures: in addition to traditional requirements like correctness and progress, the scalability is of paramount importance. It is a common opinion that these demands are partially in conflict each other, so that it is necessary to relax the requirements about a behavior identical to the corresponding sequential data structures in order to achieve high performances.

In this work we introduce a relaxed approach for the management of heap based priority queues on multicore CPUs, with the aim to realize a tradeoff between efficiency and sequential correctness. The approach is based on dynamic arrangement of the data structure among the cores, with a step-by-step redistribution procedure that shares information only among cores directly connected in a virtual mesh.

**A HIGH PERFORMANCE DATA STRUCTURE**

Local redistribution of highest priority items (relaxed approach)

No centralized data structure!

N threads are logically organized according to a 2-dimensional periodic mesh $M_2$.

We define a High Performance Heap: a collection of partially ordered binary trees $H_i$ with the max-heap property, where the nodes are connected among them according the mesh $M_2$.

Each thread $P_i$ manages a private sub-structure $H_i$. If $H_i > H_{i'}$, then the item with largest priority $i^1 > i^1'$ is moved forward to a connected thread in the mesh $H_i$ according a producer-consumer protocol, alternatively in the two directions. In this way the critical items with highest priority are passed from thread to thread, an iteration after the other, through all nodes of the mesh, with a better distribution.

**HEAP MANAGEMENT IN MULTICORE ENVIRONMENTS**

Two traditional approaches

Centralized approach: Each thread manages a private sub-structure. Access to the data without synchronization.

Distributed approach: All threads $P_i$ access a single data structure. All basic operations must be carried out in a critical section.

**HEAPS IN COMPUTATIONAL SCIENCE**

- A heap is a dynamical data structures used when there is need to process high priority items (e.g. with large numerical error) produced at run time, with an order depending on the application data and that cannot be predicted.
- Each node is tagged with a priority higher than its children. The node with highest priority $e^*$ is in the root.
- Two efficient operations are defined on a heap: remove(root) and insert(item).
- In many scientific applications, the heap is periodically updated in an iterative section of the algorithms.

**SCALABILITY ANALYSIS AND TEST RESULTS**

In the proposed algorithm, at each iteration, there are no global synchronizations among threads $P_i$ and each of them exchanges data only with the two threads $P_{i-1}$ and $P_{i+1}$, so that the synchronization overhead is $T_0 = O(1) = constant$, because it does not depend on the number of threads $N$.

**Linearizability !!**

**Scalability !!**