COTTON_LACE: Non-blocking Split Deque for Work-Stealing

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Motivation

- The work stealing runtime of the Cotton++ library can be optimized further to achieve better performance.
- The various areas with scope of improvement are-
  - Task granularity specific optimizations
  - Locality specific improvements (NUMA aware/hierarchical work stealing)
  - **Improvement of the task queue**
- We focussed on the area 3 for this project.

Problem: The Inefficient Task Deque

- **Steal** and **pop** both need locks in COTTON++
- In short, the **victim** and **thieves** need to synchronize amongst themselves to operate on the deque
- These synchronizations are **expensive** as we need memory fences for them

Solution: LACE
(Tom van Dijk, Jaco C. van de Pol, 2014)

- non blocking **split** deque
- Each deque is split into **shared** and **private** regions
- The split point is **dynamic**
- Memory fences required only when **shrinking** the shared region or **popping** tasks

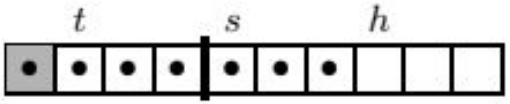


Fig. 2. The split deque, with tail t, split point s and head h. A task at position x is **stolen** if $x < t$. It is **shared** if $x < s$, and **private** otherwise. Of the 7 tasks in this example, 4 are **shared** and 1 is **stolen**.

Algorithm : Key concepts

- Thieves access **tail** and **split**, modify only **tail**

- Owner accesses **head** and **split** for push and pop
  - Only owner can modify **split**
  - Owner only accesses **tail** when pushing the first task

- Thieves set the **splitreq** flag when no stealable tasks are left

- Owner sets the **allstolen** flag when it finds the entire queue empty

```
steal()
{
    if(allstolen) return NULL
    acquire_lock()
    if(tail<split)
            task = steal_task()
    else if(!splitreq)
            splitreq=1
    release_lock()
    return task
}
```

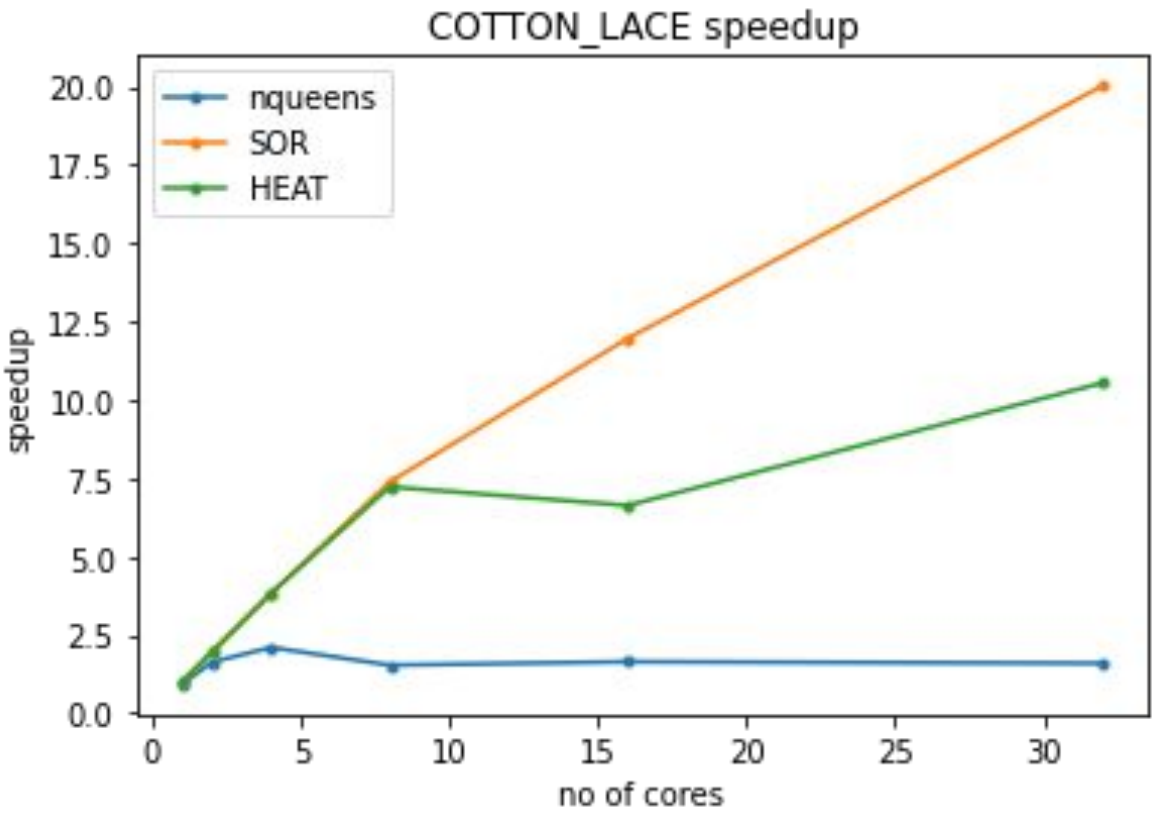
```
push()
{
    if (head==QUEUE_SIZE) return false
    push_the_task()
    if(allstolen)
                acquire_lock()
                tail=head-1
                split=head
                release_lock()
                allstolen=0
                splitreq=0
    else if(splitreq)
                grow_shared()
    return true;
}
```

```
pop()
{
   if(head==0) return NULL
    if(head==split)
          shrink_shared()
    task=steal_task()
    return task
}
```

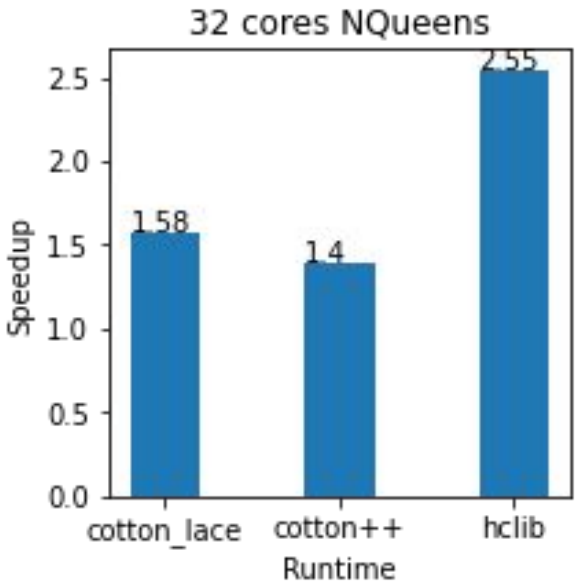
```
shrink_shared()
{
    acquire_lock()
    split=(split+tail)/2
    release_lock()
    if(unable_to_move_split)
                allstolen=1
}
```

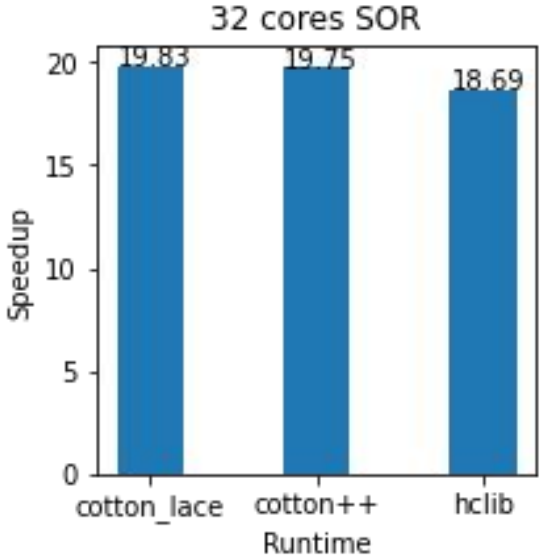
```
grow_shared()
{
    split=(split+head+1)/2
    splitreq = 0
}
```

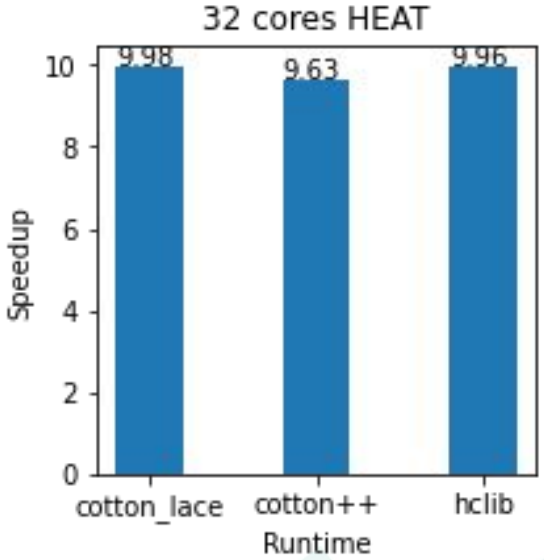
Speedup graph for COTTON_LACE



Comparison with HClib and Cotton++







In all 3 benchmarks cotton_lace outperforms cotton++

Number of Locks Taken on the Deque

Benchmark	COTTON++	COTTON_LACE
Nqueens	4,675,644	427
SOR	1,494,260	398,782
HEAT	3,446,984	1,942,058

Q/A