Cache-Efficient Parallel Partition Algorithms

Alek Westover

MIT PRIMES

October 20, 2019

Partition

Definition (Partitioned Array)

$A[i]$ *predecessor*, $A[j]$ *successor* $\implies i < j$

Definition (Array partitioned relative to pivot value p)

$A[i] \leq p, A[j] > p \implies i < j$

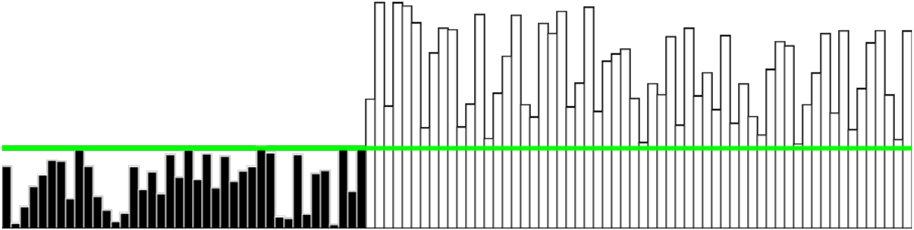
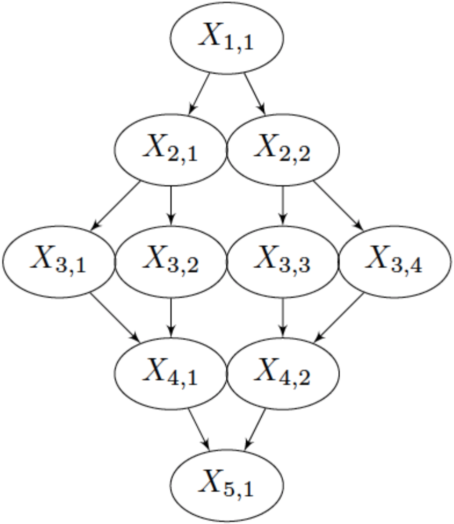


Figure: Black: Predecessor, White: Successor, Green: Pivot Value

Parallel Algorithm



Work and Span

Definition (T_p)

Running time on p processors. Note: $T_p \geq T_\infty$, $T_p \geq \frac{T_1}{p}$.

Definition (Work)

Running time on a single processor. $T_1 = \sum_i W_i$.

Definition (Span)

Running time on infinitely many processors. $T_\infty = \sum_i 1$.

Theorem (Brent's Theorem)

$$T_p = \sum_i \left\lceil \frac{W_i}{p} \right\rceil \leq \sum_i \left(\frac{W_i}{p} + 1 \right) = \frac{T_1}{p} + T_\infty.$$

Serial Partition

```
while low < high do
    while A[low] ≤ pivotValue do
        low ← low + 1
    end while
    while A[high] > pivotValue do
        high ← high − 1
    end while
    Swap A[low] with A[high]
end while
if A[low] ≤ pivotValue then
    low ← low + 1
end if
```

Randomized Algorithms

Definition (With high probability in n)

Probability of success is

$$1 - \frac{1}{n^c}$$

for c of our choice. i.e. the probability can be made arbitrarily close to 1.

Memory Bandwidth Bound

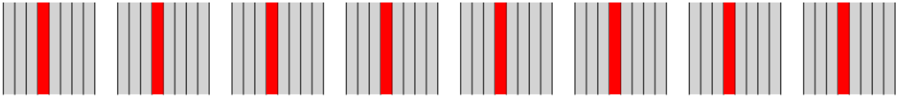
Definition (Cache miss)

A cache miss occurs when the algorithm must load a cache-line not stored in cache into cache.

- Being in-place is desirable
- Kuszmaul developed an in-place version of parallel partition.
- His in-place version outperformed the out-of-place standard algorithm.
- But was outperformed in practice by a lower span algorithm
- Experiments show the source of the problem: Memory bandwidth bound, i.e. incurring too many cache misses
- Note that both temporal and spatial cache efficiency are important.

Strided Algorithm citeFrancisPa92, Frias08 Description

Partition A into chunks $C_1, C_2, \ldots C_n/gb$ each consisting of g cache lines of size b. Let P_i be the union of the i-th cache-line from each chunk C_j.



Definition (Partially Partitioned Array)

$\exists u, l$ such that

$$i < u \implies A[i] \text{ is predecessor }, i \geq l \implies A[i] \text{ is successor}$$

- Perform serial partitions on all P_i in parallel.
- Let v_i be the position in A of the first successor in P_i. Perform a serial partition on $A[\min_i v_i], \ldots, A[\max_i v_i - 1]$.

Strided Algorithm Analysis

- Partial partiton step: work $O(n)$, span $\Theta(n/g)$.
- Serial cleanup step: span $\Theta(v_{\max} - v_{\min})$, which is $O(n)$ in general.
- If the number of predecessors in each P_i is similar, $v_{\max} - v_{\min}$ can be small.
- In particular, if $b \in \text{polylog}(n)$, and the array values are selected independently at random from some distribution, and g is chosen to optimize span ($g = n^{1/3}$), then with high probability in n,

$$v_{\max} - v_{\min} < \tilde{O}(n^{2/3}),$$

the span is

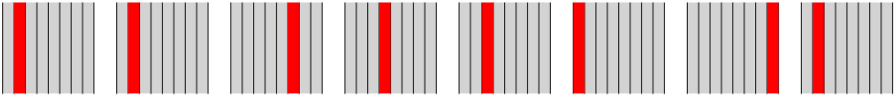
$$\tilde{O}(n^{2/3}),$$

and the number of cache misses is fewer than

$$\frac{n}{b} + \frac{\tilde{O}(n^{2/3})}{b}.$$

Smoothed Striding Algorithm Description

Let $X[1], \ldots, X[s]$ be chosen uniformly at random from $\{1, \ldots, g\}$. Let U_i be the union of the $(X[j] + i) \mod g$-th cache-line from each chunk C_j.



- Perform serial partitions on all U_i in parallel.
- The array is partially now partitioned with $A[i]$ a predecessor for all $i < v_{\text{min}}$ and $A[i]$ a successor for all $i \geq v_{\text{max}}$.

Note that we will make $s = \frac{n}{gb} < \text{polylog}(n)$ so the algorithm remains in-place.

Partial Partition Step Analysis

Proposition

Let $\epsilon \in (0, 1/2)$ *and* $\delta \in (0, 1/2)$ *such that* $\epsilon \geq \frac{1}{\text{poly}(n)}$ *and* $\delta \geq \frac{1}{\text{polylog}(n)}$*.*
Suppose $s > \frac{\ln(n/\epsilon)}{\delta^2}$*. Finally, suppose that each processor has a cache of size at least* $s + c$ *for a sufficiently large constant* c*.*
Then the Partial-Partition Algorithm achieves work $O(n)$*; achieves span* $O(b \cdot s)$*; incurs* $\frac{s+n}{b} + O(1)$ *cache misses; and guarantees with probability* $1 - \epsilon$ *that*

$$v_{max} - v_{min} < 4n\delta.$$

Hybrid Algorithm Analysis - General Theorem

Theorem

The Hybrid Smoothed Striding Algorithm algorithm using parameter $\delta \in (0, 1/2)$ *satisfying* $\delta \geq 1/\operatorname{polylog}(n)$*: has work* $O(n)$*; achieves span*

$$O\left(\log n \log \log n + \frac{b \log n}{\delta^2}\right),$$

with high probability in n; and incurs fewer than

$$(n + O(n\delta))/b$$

cache misses with high probability in n.

Hybrid Algorithm Analysis - Corollary for specific parameter settings

An interesting corollary of the above theorem concerns what happens when b is small (e.g., constant) and we choose δ to optimize span.

Corollary

Suppose $b \leq o(\log \log n)$. Then the Cache-Efficient Full-Partition Algorithm algorithm using $\delta = \Theta\left(\sqrt{b / \log \log n}\right)$, achieves work $O(n)$, and with high probability in n, achieves span $O(\log n \log \log n)$ and incurs fewer than $(n + o(n))/b$ cache misses.

Recursive Algorithm Analysis - General Theorem

Theorem

With high probability in n, the Recursive Smoothed Striding algorithm using parameter $\delta \in (0, 1/2)$ *satisfying* $\delta \geq 1/\operatorname{polylog}(n)$*: achieves work* $O(n)$*, attains span*

$$O\left(b\left(\log^2 n + \frac{\log n}{\delta^2}\right)\right),$$

and incurs $(n + O(n\delta))/b$ *cache misses.*

Recursive Algorithm Analysis - Corollary for specific parameter settings

A particularly natural parameter setting for the Recursive algorithm occurs at $\delta = 1/\sqrt{\log n}$.

Corollary

With high probability in n, the Recursive Smoothed Striding Algorithm using parameter $\delta = 1/\sqrt{\log n}$*: achieves work* $O(n)$*, attains span* $O(b \log^2 n)$*, and incurs* $n/b \cdot (1 + O(1/\sqrt{\log n}))$ *cache misses.*

Experiments

- Strided Algorithm vs Smoothed Striding algorithm
- Cache misses

Acknowledgments

I would like to thank

- The MIT PRIMES program
- William Kuszmaul, my PRIMES mentor
- My parents

References

Rhys S. Francis and LJH Pannan. 1992.
A parallel partition for enhanced parallel quicksort.
Parallel Comput. 18, 5 (1992), 543–550.