Chapter 14. Parallel Algorithms.

1. To describe a parallel algorithm
   - Parallel Random Access Machine (PRAM)
   - Work – Depth (WD)
2. To resolve access conflicts
   - Exclusive – Read Exclusive - Write (EREW)
   - Concurrent – Read Exclusive – Write (CREW)
   - Concurrent – Read Concurrent – Write (CRCW)
3. concept
   - Work load – total number of operations: $W(n)$
   - Worst-case running time: $T(n)$
   - $P(n) = W(n)/T(n)$ processors and $T(n)$ time (on a PRAM)
   - $W(n)/p$ time using any number of $p \le W(n)/T(n)$ processors
   - $W(n)/p + T(n)$ time using any number of p processors.

WD presentation:

can be implemented by any $P(n)$ processors within $O(W(n)/P(n) + T(n))$ time.

Prefix sums:

$T(n) \overset{?}{=} O(\log n)$ $\quad W(n) = O(n)$

Merge:

binary Search: $T(n) = O(\log n)$ $\quad W(n) = O(n \log n)$

Serial Ranking: $T(n) = O(n+m)$ $\quad W(n) = O(n+m)$

Parallel Ranking:

Stage 1: Partitioning $p = n/\log n$ $\quad T = O(\log n)$ $\quad W = O(p \log n) = O(n)$

Stage 2: Actual Ranking $\quad T = O(\log n)$ $\quad W = O(p \log n) = O(n)$

$T = O(\log n)$, $\quad W = O(n)$

Maxium finding:

Replace "+" by "max": $T(n) = O(\log n)$, $W(n) = O(n)$

Compare all pairs: $T(n) = O(1)$ $\quad W(n) = O(n^2)$

Partition by $\sqrt{n}$:

$A_1 = A(1)$, $\quad A(\sqrt{n}) \Rightarrow M_1 \sim T(\sqrt{n}), W(\sqrt{n})$

$A_2 = A(\sqrt{n}+1)$, $\quad A(2\sqrt{n}) \Rightarrow M_2 \sim T(\sqrt{n}), W(\sqrt{n})$

$A_{\sqrt{n}} = A(n-\sqrt{n}+1)$, $\quad A(n) \Rightarrow M_{\sqrt{n}} \sim T(\sqrt{n}), W(\sqrt{n})$

$M_1, M_2, \cdots M_{\sqrt{n}} \Rightarrow A_{max} \sim T = O(1),\ W = O(\sqrt{n}^2) = O(n)$

$T(n) \le T(\sqrt{n}) + c_1$, $\quad W_n \le \sqrt{n}\, W(\sqrt{n}) + c_2 n \Rightarrow T(n) = O(\log \log n)$,

$W(n) = O(n \log \log n)$

Partition by $h = \log \log n$:

$A_1 = A(1)$, $\quad A(h) \Rightarrow M_1 \sim O(h)$

$A_2 = A(h+1)$, $\quad A(2h) \Rightarrow M_2 \sim O(h)$

$A_{n/h} = A(n-h+1)$, $\quad A(n) \Rightarrow M_{n/h} \sim O(h)$

$T(n) = O(h + \log\log(n/h)) = O(\log \log n)$

$W(n) = O(h \times (n/h) + (n/h) \log\log(n/h)) = O(n)$

Random Sampling:

$M(n^{\frac{7}{8}}) \sim T = O(1)$,

$W = O(n)$

Chapter 13 Randomized Algorithms

The world behaves randomly – randomly generated input solved by traditional algorithm.

Average-case Analysis

The algorithm behaves randomly – make random decisions as the algorithm processes the worst-case input

Randomized Algorithms. $\qquad$ permute array

$E[X] = \sum_{j=0}^{\infty} j \cdot Pr[X=j]$

The hiring problem

$$X_i = \begin{cases} 1 & \text{if candidate } i \text{ is hired} \\ 0 & \text{if candidate } i \text{ is not hired} \end{cases}$$

$$\Rightarrow X = \sum_{i=1}^{N} X_i$$

$$E[X] = E[\sum_{i=1}^{N} X_i] = \sum_{i=1}^{N} E[X_i] = \sum_{i=1}^{N} Pr[\text{candidate } i \text{ is hired}] = \sum_{i=1}^{N} \frac{1}{i} = \ln N + O(1)$$

$$\Rightarrow O(C_h \ln N + N C_i)$$

Online hiring Algorithm

$S_i :=$ the ith applicant is the best

$\{A_i :=$ the best one is at position $i\}$

$\cap \{B_i :=$ no one at position $k+1 \sim i-1$ are hired$\}$

$Pr[S_i] = Pr[A \cap B] = Pr[A] \cdot Pr[B] = \frac{k}{N(i-1)}$

$Pr[A] = \frac{1}{N}$, $\quad Pr[B] = \frac{k}{i-1}$

$$Pr[S] = \sum_{i=k+1}^{N} Pr[S_i] = \sum_{i=k+1}^{N} \frac{k}{N(i-1)} = \frac{k}{N} \sum_{i=k}^{N-1} \frac{1}{i}$$

$$\frac{k}{N} \ln\left(\frac{N}{k}\right) \le Pr[S] \le \frac{k}{N} \ln\left(\frac{N-1}{k-1}\right)$$

about $k = \frac{1}{e}$.

Quick Sort $\qquad$ ※ 数值随机化与过程随机化

Deterministic Quicksort

– $\Theta(N^2)$ worst-case running time

– $\Theta(N \log N)$ average case running time, assuming every input permutation is equally likely.

Central splitter := the pivot that divided the set so that each side contains at least $\frac{n}{4}$

Modified Quicksort := always select a central splitter before recursions.