PRAM (4)
The relative power of different PRAM models
Topics

• Comparison of PRAM models
  – relative performance of EREW, CREW, and CRCW models
    • separated by lower bounds for key problems
    • related by simulation results

• Restricted CR/CW models

• Abstract computational complexity classes
  – inherently sequential problems?

• Work-Time and PRAM model
  – advantages
  – disadvantages
Comparison of PRAM memory models

- Some specific problems illustrate asymptotic advantage of CR and CW
  - Assume p processor PRAM (so not WT model)

  - Copy problem
    - given scalar y in shared memory, create vector R[1:p] with R[i] = y for 1 ≤ i ≤ p
      - EREW: \( T_C (p) = \Theta(lg p) \)
      - CREW, CRCW: \( T_C (p) = \Theta(1) \)

  - Maximum problem (comparison-based)
    - given X[1:p] in shared memory, find maximum value m in X
      - EREW, CREW: \( T_C (p) = \Theta(lg p) \)
      - arbitrary CRCW: \( T_C (p) = \Theta(lg lg p) \)
      - max-combining CRCW: \( T_C (p) = \Theta(1) \)

- These problems illustrate that
  - \( \text{EREW} \prec \text{CREW} \prec \text{arbitrary CRCW} \prec \text{combining CRCW} \)

  where \( A \prec B \) means that some problem can be solved asymptotically faster using model B than using model A
Power of concurrent reads: Copy problem

- **EREW Upper bound**: $O(\lg p)$ time
  - how?

- **EREW Lower bound**: $\Omega(\lg p)$ time
  - consider a step consisting of an EREW PRAM read and write operation.
    - **step 1**
      - at the start of step 1, only one copy of the value $y$ exists
      - Only one processor can read it and write it back to a new location
    - **step 2**
      - at the start of step 2, two copies of the value exist
      - two processors can each read one value simultaneously and write it back simultaneously to distinct locations
    - **step $(\lg p)$**
      - at most $2^{(\lg p)-1} = p/2$ copies exist, $p/2$ processors copy them simultaneously yielding $p$ copies.
    - therefore $\Omega(\lg p)$ steps are needed for $p$ processors to read a single value

- **CREW upper and lower bound**: $\Theta(1)$ time
  - trivial
Power of concurrent writes: Maximum problem

- Find maximum $m$ of sequence $X = <x_1, ..., x_n>$
  - EREW algorithm:
    - $S(n) = \Theta(\lg n)$, $W(n) = \Theta(n)$
  - CRCW algorithm
    - $S(n) =$
    - $W(n) =$

CRCW fast maximum - WT formulation

**Input:** $X[1:n]$

**Output:** $m = \max_{i \in 1:n} X[i]$

**Auxiliary:** $B[1:n]$

```plaintext
forall i in 1 : n do
    B[i] := 1
enddo

forall (i,j) in {1..n} \times {1..n} do
    if $X[i] < X[j]$ then
        B[i] := 0
    endif
enddo

forall i in 1 : n do
    if B[i] = 1 then
        m := X[i]
    endif
enddo
```
Work-efficient CRCW Maximum

1. Apply CRCW fast max to inputs organized into a very shallow tree
   place values at leaves of doubly-logarithmic depth tree
   - assume \( n = 2^{2^k} \) for some \( k > 0 \), so \( k = \log \log n \)
   - branching factor at level \( 1 \leq i \leq k \) is \( 2^{2^k-i} \) (add last level \( k+1 \) with bf 2)
   - apply fast maximum in parallel to all nodes in a level
     - total work at each level is \( O(n) \), number of levels is \( O(\log \log n) \)
     - \( S_1(n) = O(\log \log n), W_1(n) = O(n \log \log n) \) better but still not work efficient

2. In parallel apply sequential max to groups of \( (\log \log n) \) elements of input
   - let \( m = n / (\log \log n) \)
   - \( S_2(n) = O(\log \log n), W_2(n) = O(mS_2(n)) = O(n) \)

3. Cascade (1) and (2) to construct work-efficient fast maximum
   - \( S(n) = S_1(m)+S_2(n) = O(\log \log n), W(n) = W_1(m)+ W_2(n) = O(n) \)

4. CRCW PRAM-level upper bound for maximum
   - Brent’s theorem with \( n = p \): \( T_{\text{C}}(p,p) = O(p/p + \log \log p) = O(\log \log p) \)

CRCW PRAM-level lower bound for maximum: \( \Omega(\log \log p) \) [hard]
Quantifying relative power of CR/CW

• Lower bound: from example problems
  – $\Omega(lg p)$ slowdown of $p$-processor EREW PRAM over $p$-processor CREW or CRCW PRAM
  – guaranteed for some problems

• Upper bound: by simulation argument
  – $O(lg p)$ slowdown in $p$-proc EREW PRAM simulation of $p$-proc CRCW PRAM
    • Depends on existence of EREW algorithm to sort $p$ values in $O(lg p)$ steps using $p$ processors [Cole]
  – To simulate a single CR step using EREW PRAM
    • sort all addresses being read
    • identify unique addresses
    • EREW read of unique addresses
    • replicate (COPY) results to match number of reads (copy)
  – To simulate a single CW step using EREW PRAM
    • sort all (addr, new value) pairs
    • implement CW resolution strategy using parallel prefix
    • EREW write of surviving (addr, value) pairs
PRAM shared memory system

- PRAM model
  - assumes access latency is constant, regardless of value of $p$
  - includes CR and/or CW

- Physically impossible
  - processors and memory occupy finite volume $p$ and $m$
    - speed of light dictates increasing latency
      \[ L = \Omega\left((p + m)^{1/3}\right) \]
  - CR / CW must be reduced to ER / EW
    - requires $\Omega(\lg p)$ time in general case
Restricted CR / CW models

• unbounded CR and CW are expensive to implement
  – multi-stage combining and expansion network with $\lg p$ depth
    • NYU Ultracomputer, SB-PRAM

• restricted models with more efficient implementations
  – QRQW - queued read / queued write
    • cost of reference proportional to number of concurrent readers / writers
  – single-value broadcast
    • fundamental for SIMD operation
    • (simple) custom network
  – concurrent write of single shared location (bit)
    • fundamental for SIMD operation
    – $O(1)$ detection if any processor is enabled
    • (simple) custom network: write combining via logical OR
    • example: maximum problem in bit model
Bit-serial CRCW maximum algorithm

- \( S(n) = O(b) \)
- \( W(n) = O(bn) \)

**Bit-serial CRCW fast maximum**

**Input:** \( X[1:n] \)  \( b \)-bit unsigned int

**Output:** \( m = \max_{i \in 1:n} X[i] \)

**Auxiliary:** \( B[1:n] \) single-bit values
- \( c \) single-bit CRCW value

```plaintext
forall i in 1 : n do
    B[i] := 1
enddo

m := 0

for k := b-1 downto 0 do
    c := 0
    forall i in 1 : n do
        if (B[i] & (X[i] bit k)) then
            c := 1
        endif
        B[i] := B[i] & (c == (X[i] bit k))
    enddo

(m bit k) := c
enddo
```
Relation of (P)RAM complexity classes

• Complexity class P
  – problems with polynomial time complexity on RAM
    • $W(n) = S(n) = O(n^{O(1)})$ in W-T model

• Complexity class NC
  – problems in P with fast parallel algorithms
    • $W(n) = O(n^{O(1)})$ polynomial work
    • $S(n) = O(\lg^{O(1)} n)$ poly-logarithmic step complexity
    – very coarse form of work-efficiency

• (P – NC)
  – “inherently sequential” problems
Inherently Sequential Problems

- Polynomial Time complete (P-complete) problems
  - H is P-complete if
    - \( H \in P \)
    - for all \( A \in P \), \( A \) is log-space (RAM) reducible to \( H \)
  - P-complete problem
    - Circuit Value Problem (CVP)
  - P-complete by reduction to CVP
    - maximum flow in network, CFG parsing, (predicate logic) unification

- Can we find a fast parallel algorithm for a P-complete problem?
  - \( H \in NC \) and \( H \) is P-complete implies \( P = NC \)
  - no luck yet

- Conjecture: \( (P – NC) \neq \emptyset \)
  - if true
    - there exist “inherently sequential” problems
    - of limited consequence due to coarse definition of NC
W-T and PRAM models - conclusions

• Strengths of W-T and PRAM models
  – Ignore memory access costs
  – Source-level complexity metrics simplify analysis
  – Widely developed body of techniques
  – W-T programs are simple and expressive

• Liabilities of W-T and PRAM models
  – memory access cost is not constant in real life
    • already true with RAM model
      – random memory access time is $\Omega(\sqrt[3]{|\text{mem}|})$ in 3D space with speed of light restrictions
      – this is the reason for cache memories in modern processors
    • even less accurate for PRAM model
      – random memory access time is $\Omega(\sqrt[3]{p+|\text{mem}|})$
      – CR / CW implementations require $\Omega(\log p)$ time with present technologies
      – switching and bandwidth issues complicate situation further
W-T and PRAM model - conclusions

- Liabilities of W-T and PRAM models
  - Source-level complexity metrics oversimplify analysis
    - given two implementations
      - efficient sequential algorithm S on sequential computer
      - work-efficient and fast parallel algorithm C on PRAM-like parallel computer
    - for sufficiently large n, there exists p such that $T_c(n,p) < T_s(n)$
      - parallel algorithm is guaranteed to run faster
    - but p (and n) may be impractically large
      - p is not a truly scalable parameter in practice
  - Widely developed body of unrealistic techniques
    - extensive use of asymptotically efficient but impractical building blocks
      - fast and efficient sorting, efficient pointer jumping, etc.
  - W-T programs may not be able to fully express efficient implementations
    - homework problem 1