CSCI 1010
Models of Computation

Lecture 24
Memory Hierarchies
Overview

• **Red** pebble game reviewed
• **Red-blue** pebble game on FFT and matrix multiplication models I/O operations
• Bounds on I/O operations for these problems
• For balanced computer systems, I/O and CPU time about the same.
• FFT harder to balance than matrix multiplication.
• Sketch of Hong-Kung I/O lower bound method
  – See Chapter 11 of Models of Computation by Savage
Motivation

• Large RAM simulated by a hierarchy of memories; from small, very fast to slow, very large.

• I/O time is the time to move data between two levels in the hierarchy.
Motivation

• **Goal**: To understand how I/O time at each level trades with the size of the memories.

• This is an OS issue. Page replacement algorithms are used to decide which cache lines or RAM pages to move to the next higher level when data must be moved from the higher to the lower level.

• We start with two-level memory model.
The Red Pebble Game

• Can pebble an input vertex at any time
• To pebble a non-input vertex, its predecessors must carry pebbles.
• Pebbles can slide.
• Can remove pebble any time
• Each output pebbled once
• Models CPU register allocation
  – Goal: Use registers efficiently
The **Red-Blue Pebble Game**

- **S** hot red pebbles simulate CPU registers.
- Unlimited number of cool **blue** pebbles simulate locations in secondary storage.
- Each **swap** of red and blue pebbles models an I/O operation.
- \( T_{I/O} = \text{number of I/O operations} \).
- **Goal:** Given \( S \) red pebbles, minimize \( T_{I/O} \).
  - Model problems where I/O ops are a bottleneck.
Rules of **Red-Blue Pebble Game**

1. (Computation) A *red* pebble can be put on a vertex if its predecessors have *red* pebbles.
2. (Deletion) A pebble can be deleted at any time.
3. (Initialization) **Blue** pebbles on inputs initially.
4. (Input from blue level) A *red* pebble can be placed on a vertex carrying a *blue* pebble.
5. (Output to blue level) A *blue* pebble can be placed on a vertex carrying a *red* pebble.
The Red-Blue Pebble Game

- **Definition** A pebbling strategy for a DAG is the sequential execution of the rules that places a blue pebble on each output. **Space** $S$ is the number of red pebbles used. **I/O time** is number of I/O operations used by the strategy. A **minimal pebbling strategy** has minimal I/O time for a given amount of space.
Red-Blue Game on FFT

• How many red pebbles suffice?
• What is an I/O efficient pebbling strategy?
  – For 9 red pebbles?
  – For a small number of pebbles?
• Can be shown that $T_{I/O} = \Theta(n \log n / \log S)$. 
RB Game on Matrix Multiplication

\[
\begin{bmatrix}
\text{\textbullet}\hspace{1cm}
\text{\textbullet}
\end{bmatrix}
= \left(\begin{array}{c}
\text{\textbullet}
\end{array}\right) \times \left(\begin{array}{c}
\text{\textbullet}
\end{array}\right)
\]

- \( C = [c_{i,j}] \), \( 1 \leq i,j \leq n \), is the **inner product**

\[
c_{i,j} = \sum_{1 \leq k \leq n} a_{i,k} \times b_{k,j}
\]

- Each \( c_{i,j} \) uses \( n \times \)'s, \( n-1 + \)'s, total = \( n^2(2n-1) \) I/O ops

- Write \( c_{i,j} = (\ldots(a_{i,1} \times b_{1,j})+a_{i,2} \times b_{2,j})+ a_{3,k} \times b_{3,j})+ \ldots) \)

  Can pebble with 3 red pebbles, \( 2n+1 \) I/O’s per \((i,j)\).

- MM can be done with \( S = 3 \), \( T_{I/O} = n^2(2n+1) \).
RB Game on Matrix Multiplication

• How to pebble with \( S \geq 3 \) pebbles?
  – Group entries into \( r \times r \) blocks, \( r = \sqrt{S/3} \)
• Let \( A = [A_{i,j}] \), \( B = [B_{i,j}] \), \( C = [C_{i,j}] \); \( A \), \( B \) \& \( C \) are \( (n/r) \times (n/r) \) matrices of \( r \times r \) submatrices \( A_{i,j} \), \( B_{i,j} \) and \( C_{i,j} \).
• For \( 1 \leq i, j \leq n/r \), define the \( r \times r \) submatrix \( C_{i,j} \) by
  \[
  C_{i,j} = \sum_{1 \leq k \leq n/r} A_{i,k} \times B_{k,j}
  \]
• Each of \( A_{i,k} \), \( B_{k,j} \) used in \( n/r \) inner products. Thus, \( n/r \) I/O ops done on each entry of \( A \), \( B \).
RB Game on Matrix Multiplication

• Group entries into $r \times r$ blocks, $r = \sqrt{S/3}$

$$C_{i,j} = \sum_{1 \leq k \leq n/r} A_{i,k} \times B_{k,j}$$

• Pebbling strategy mirrors the basic one:
  – Do I/O on inputs to $A_{i,k}, B_{k,j}$. Use $S$ red pebbles to form product $C_{i,j}$. (Why do $S$ red pebbles suffice?)
  – Each entry of $A$ and $B$ uses $(n/r)$ I/O ops

• Thus, $T_{I/O} = O(n^3/r) = O(n^3/\sqrt{S})$.
  – Can’t improve bound by more than constant factor.
Balanced Computer Systems

• **Goal:** neither CPU nor I/O (disk) dominates
  – \( t_{\text{cpu}} \) = duration of a CPU cycle
  – \( t_{\text{I/O}} \) = duration of an I/O op (**typically** \( 100t_{\text{cpu}} \))
  – \( C_{\text{cpu}} = \frac{1}{t_{\text{cpu}}} \) is CPU rate (cycles/time)
  – \( C_{\text{I/O}} = \frac{1}{t_{\text{I/O}}} \) is I/O rate (ops/time) (**typically** \( 10^{-2} C_{\text{cpu}} \))
  – \( T_{\text{cpu}}(P) \) = # computation steps for problem \( P \)
  – \( T_{\text{I/O}}(P) \) = # I/O ops for problem \( P \)

• Elapsed time = \( t_{\text{cpu}} T_{\text{cpu}}(P) + t_{\text{I/O}} T_{\text{I/O}}(P) \)
Balanced Computer Systems

• Elapsed time = $t_{cpu} T_{cpu}(P) + t_{I/O} T_{I/O}(P)$

• System balanced if $t_{cpu} T_{cpu}(P) \equiv t_{I/O} T_{I/O}(P)$ or
  $\rho = C_{cpu}/C_{I/O} \equiv T_{cpu}(P)/T_{I/O}(P)$

• $\rho$, the relative speed of CPU vs I/O, is large

• No I/O bottleneck if $t_{cpu} T_{cpu}(P) \geq t_{I/O} T_{I/O}(P)$
  $\rho \leq T_{cpu}(P)/T_{I/O}(P)$
Balanced Systems for MM

• Matrix multiplication:
  – $T_{cpu}(P) = n^3$, $T_{I/O}(P) = \Theta(n^3/\sqrt{S})$
  – The system is in balance when
    \[
    \rho = \frac{C_{cpu}}{C_{I/O}} = \Theta(\sqrt{S}).
    \]

• No I/O bottleneck if $\rho \leq \Theta(\sqrt{S})$ (if $\rho = 10^2$, $S > 10^4$)
Balanced System for FFT

- Fast Fourier Transform:
  - \( T_{cpu}(P) = \Theta(n \log n) \), \( T_{I/O} = \Theta(n \log n / \log S) \).
  - The system is balance when
    \[
    \rho = \frac{C_{cpu}}{C_{I/O}} = \Theta(\log S)
    \]

- No I/O bottleneck if \( \rho \leq \Theta(\log S) \) (if \( \rho = 10^2 \), \( S > 10^{30} \))
Balanced Computer Systems

• **Conclusion:** Achieving balance is much more difficult for FFT than MM.
  
  – For a given amount of fast memory, $S$, I/O efficient computations require much smaller $p$ for FFT than for MM.
Lower Bound Method

• We describe a method to derive lower bounds on the number of I/O operations to pebble a DAG when we limit ourselves to S red pebbles.

• The approach is to derive an upper bound on the \textbf{S-span} of a graph G, \( \rho(S,G) \), the number of vertices that a can be pebbled from some initial placement of red pebbles when the initialization rule is disallowed.
Hong-Kung Lower Bounds

**Theorem** For every pebbling $P$ of $G = (V,E)$ in the red-blue pebble game with $S$ red pebbles, the I/O time used, $T_{I/O}(S,G,P)$, satisfies

$$T_{I/O}(S,G,P) \rho(2S,G) \geq |V| - |\text{In}(G)|$$

Where $\text{In}(G)$ is the number of input vertices.

**Proof** Divide the pebbling steps $P$ into subsets $P_1, ..., P_h$ where each $P_j$ consists of steps containing $S$ I/O ops except the last which may have fewer. Then, $h = \lceil T_{I/O}(S,G,P)/S \rceil$
Hong-Kung Lower Bound

Proof (cont) Let Q be an upper bound to number of vertices pebbled by red pebblings in $P_j$. Thus, $Qh$ is at least $|V| - |\text{In}(G)|$.

With $S$ additional red pebbles we can move all I/O ops in an interval $P_j$ to beginning or end of the interval. Thus, we now have $2S$ pebbles and can pebble at most $\rho(2S,G)$ vertices in $P_j$ and at most $h\rho(2S,G)$ in all $h$ intervals.
Hong-Kung Lower Bound

- **Proof (cont.)** Consider a vertex carrying a red (blue) pebble at beginning of $P_j$ that is pebbled blue (red) in $P_j$.
- In 1\textsuperscript{st} case use a new red pebble to postpone blue pebbling to the end of $P_j$. In 2\textsuperscript{nd} case use new red pebble to make the blue-t-red transition at the beginning of $P_j$.
- All blue pebblings are at ends of $P_j$. All pebblings inside $P_j$ are done with red pebbles. But at most $\rho(2S,G)$ of these can be done per interval. QED
Summary

- Red pebble game reviewed
- Red-blue pebble game on FFT and matrix multiplication models I/O operations
- Bounds on I/O operations for these problems
- For balanced computer systems, I/O and CPU time about the same.
- FFT harder to balance than matrix multiplication.
- Sketch of Hong-Kung I/O lower bound method
  - See Chapter 11 of Models of Computation by Savage