Module-2

CSEN 3104

Lecture 21

27/08/2019

Dr. Debranjan Sarkar

SIMD Algorithms

Sorting on a mesh-connected parallel computer

Odd-even merge on a rectangular array of PEs

- Batcher's odd-even merge sort on a linear array can be generalized to a square array of PEs
- Let M(j,k) be the algorithm of merging two j-by-k/2 sorted adjacent subarrays to form a sorted j-by-k array, where j, k are powers of 2, and $k > 1$
- All the arrays are arranged in the snake-like row major ordering
- When $j = 1$ and $k = 2$, i.e., in case of M(1,2), a single comparison-interchange step is sufficient to sort two unit subarrays

Odd-even merge on a rectangular array of PEs

- Given two sorted columns of length $j \geq 2$, $M(j, 2)$ consists of the following steps:
- J1. Move all odds to the left column and all evens to the right. Time: $2t_R$
- J2. Use the "odd-even transposition sort" to sort each column

  Time: $j\,(2t_R + t_c)$
- J3. Interchange on even rows. Time: $2t_R$
- J4. One step of comparison-interchange (every "even" with the next "odd")

  Time: $2t_R + t_c$
- So total time required = $2t_R + j\,(2t_R + t_c) + 2t_R + (2t_R + t_c) = (6 + 2j)\,t_R + (1 + j)\,t_c$

- Show Figure to illustrate the algorithm $M(j, 2)$ for $j = 4$

Odd-even merge on a rectangular array of PEs

- For $j > 2$ and $k > 2$, $M(j, k)$ is defined recursively in the following way:
- M1. If $j > 2$, perform a single interchange step on even rows
  If $j = 2$, do nothing Time: $2t_R$
- M2. Unshuffle each row Time: $(k - 2)t_R$
- M3. Merge by calling $M(j, k/2)$ on each half Time: $T(j, k/2)$
- M4. Shuffle each row Time: $(k - 2)t_R$
- M5. Interchange on even rows Time: $2t_R$
- M6. Comparison-interchange of adjacent elements
  (every "even" with the next "odd“) Time: $4t_R + t_c$

Odd-even merge on a rectangular array of PEs

- Steps M1 and M2 unshuffle the elements
- Step M3 recursively merges the "odd sequences“ and the "even sequences”
- Steps M4 and M5 shuffle the "odds" and "evens" together
- Step M5 performs the final comparison-interchange
- Show figure to illustrate the algorithm M(4, 4), where the two given sorted 4-by-2 subarrays are initially stored in 16 processors

Odd-even merge on a rectangular array of PEs

- Let $T(j, k)$ be the time needed by $M(j, k)$. Then we have
- $T(j, 2) = (2j + 6)t_R + (j + 1)t_C$ for $k = 2$
- $T(j, k) = (2k + 4)t_R + t_C + T(j, k/2)$ for $k > 2$
- By repeated substitution, we have the following time bound:

  $T(j, k) \le (2j + 4k + 4\log_2 k)t_R + (j + \log_2 k)t_c$
- For n x n array of PEs, the $M(n,n)$ sort algorithm can be done in $T(n,n)$ time which is proportional to $O(n)$:

  $T(n,n) = (6n + 4\log_2 n)t_R + (n + \log_2 n)t_c = O(n)$ $[t_c \le t_R]$
- A speedup of $O(\log_2 n)$ achieved over the best sorting algorithm (Quicksort), which takes $O(n\log_2 n)$ steps on a uniprocessor system (in the best case and in the average case)

Thank you