Module-2
CSEN 3104
Lecture 20
26/08/2019

Dr. Debranjan Sarkar

SIMD Algorithms

Sorting on a mesh-connected parallel computer

Parallel Sorting on mesh

- Sorting of $N = n^2$ elements on an $n \times n$ mesh-type processor array
- Architecture (show figure) is similar to Illiac IV with exceptions
  - No wraparound connections, i.e.,
  - PEs at the perimeter have 2 or 3 rather than 4 neighbours
  - This simplifies the array sorting algorithm
- Two time measures are required to estimate the time complexity of the algorithm:
  - Routing time (t_R) to move one data item from a PE to one of its neighbours
  - Comparison time (t_C) for one comparison step (conditional interchange on the contents of two registers in each PE

Parallel Sorting on mesh

- Concurrent data routing is allowed
- Upto N numbers of concurrent comparisons may be performed
- This means that a comparison-interchange step between two items in adjacent processors can be done in time $2t_R + t_c$ *(route left, compare, and route* right)
- A number of these comparison-interchange steps may be performed concurrently in time ($2t_R + t_c$) *if* they are all between distinct, vertically adjacent processors
- A mixture of horizontal and vertical comparison-interchanges will require at least ($4t_R + t_c$) time unit

Parallel Sorting on mesh

- The PEs may be indexed by a bijection from {1, 2, …., n} × {1, 2, …., n} to {0, 1, …, N - 1}, where $N = n^2$
- N elements of a linearly ordered set are initially loaded in the N PEs
- Sorting problem is defined as the problem of moving the j^{th} smallest element to the processor indexed by j for all j = 0, 1 …., N – 1
- Example:
- The elements (N=16, n = 4) to be sorted are initially loaded in the 4 X 4 array of PEs (Show Figure)
- Three ways of indexing the processors
  - *Row-major indexing*
  - *Shuffled row-major indexing*
  - *Snake-like row-major indexing*

Parallel Sorting on mesh

- *Row-major indexing (Show diagram)*
- *Shuffled row-major indexing (Show diagram)*
  - Note that this indexing is obtained by shuffling the binary representation of the row-major index
  - For example, the row-major index 5 has the binary representation 0101
  - Shuffling the bits gives 0011 which is 3
  - In general, the shuffled binary number, say, "abcdefgh" is "aebfcgdh“
- *Snake-like row-major indexing (Show diagram)*
  - Obtained from the row-major indexing by reversing the ordering in even rows
- The choice of a particular indexing scheme depends upon how the sorted elements will be used
- We are interested in designing algorithms which minimize the time spent in routing and comparing

Parallel Sorting on mesh

- For any index scheme, there are situations where the two elements initially loaded at the opposite corner PEs, have to be transposed during the sorting (Show Figure)
- This transposition needs at least 4(n - 1) routing steps
- This implies that no algorithm can sort n^2 elements in time less than *O(n)*
- Thus an O(n) sorting algorithm is considered optimal on a mesh of n^2 PEs
- We shall show one such optimal sorting algorithm on the mesh-connected PEs

Odd-even Transposition Sort

- Different Sorting algorithms
  - Bubble SortComputational Complexity: $O(n^2)$ in average case
  - Merge Sort Computational Complexity: O(n log n) in worst case
  - Quick Sort Computational Complexity: O(n log n) in average case
- These algorithms are not easily parallelizable
  - Because the operations depend on the result of the previous operations
- Odd-even Transposition sort (or Brick Sort) is suitable for parallel computers and the time complexity is reduced to O(n)
- Examples of Odd-Even Transposition Sort

Review of Batcher's odd-even merge sort

- Sort the first half of a list, and sort the second half separately
- Sort the odd-indexed entries (first, third, fifth, …) and the even-indexed entries (second, fourth, sixth, …) separately
- Make only one more comparison-switch per pair of keys to completely sort the list
- List of numbers: 2 7 6 3 9 4 1 8
- We wish to sort it from least to greatest
- If we sort the first and second halves separately we obtain: 2 3 6 7 / 1 4 8 9
- Sorting the odd-indexed keys (2, 6, 1, 8) we get (1 2 6 8)
- Sorting the even-indexed keys (3, 7, 4, 9) we get (3 4 7 9)
- Leaving them in odd and even places respectively yields: 1 3 2 4 6 7 8 9
- This list is now almost sorted
- Doing a comparison switch between the keys in positions (2 and 3), (4 and 5) and (6 and 7) will finish the sort

Review of Batcher's odd-even merge sort

- Normally, the length of the list is a power of 2 (Here $2^3 = 8$)
- Two sorted sequences are loaded on a set of linearly connected PEs (Show Figure)
- In the first stage, the odd-indexed elements are placed in the left and then the even-indexed elements are placed in the right. This is basically unshuffle (or inverse shuffle) operation
- In the second stage, the odd sequences and the even sequences are merged
- The third stage is basically a perfect shuffle operation
- The fourth and final stage is a comparison-interchange operation of even-indexed elements with the next element
- Note that the perfect shuffle can be achieved by using the triangular interchange pattern (show figure)
- Similarly, an inverted triangular interchange pattern will do the unshuffle.
- The double-headed arrows indicate interchanges

Thank you