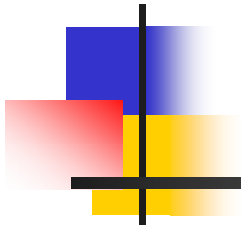
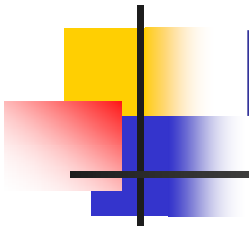


Predictive Block Dataflow Model for Parallel Computation

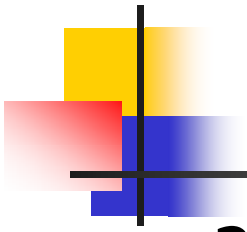
Vishal Mishra and Kursad Oney

29th April 2002



Motivation and Outline

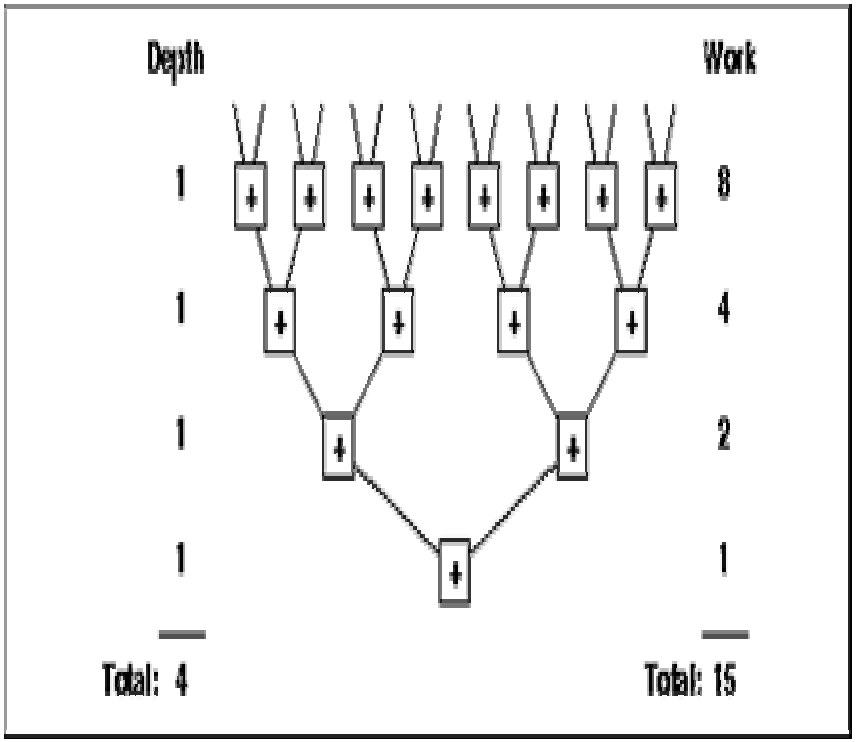
- Age of billion transistors on a chip
- Communication cost greatly reduced
- Programs written for sequential machines, don’t exploit inherent parallelism in programs
- Propose a new dataflow architecture
- Simulate the architecture for recursive data structures e.g. quicksort, Dijkstra’s shortest path
- Expose parallelism in these recursive algos.



Model of Computation

- 2-D array of processors with local memory on a single chip
- Token: [dest., context, instruction ptr, data]
- Blocks of instructions, coarse grained
- Data driven, owner computes rule
- Async operation or loosely sync
- Predictive: pre-fetching blocks of instructions
- Locality: block level
- Global address space: processor id & memory

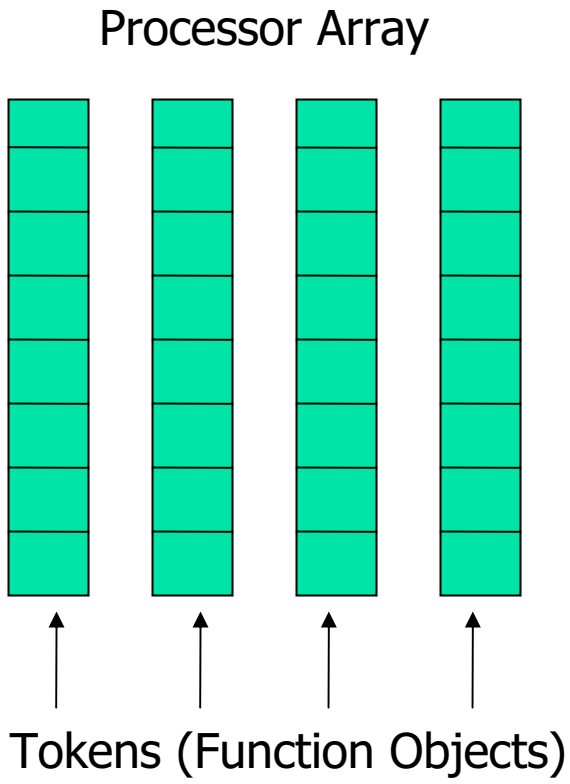
Simulation: Modified Work-Depth



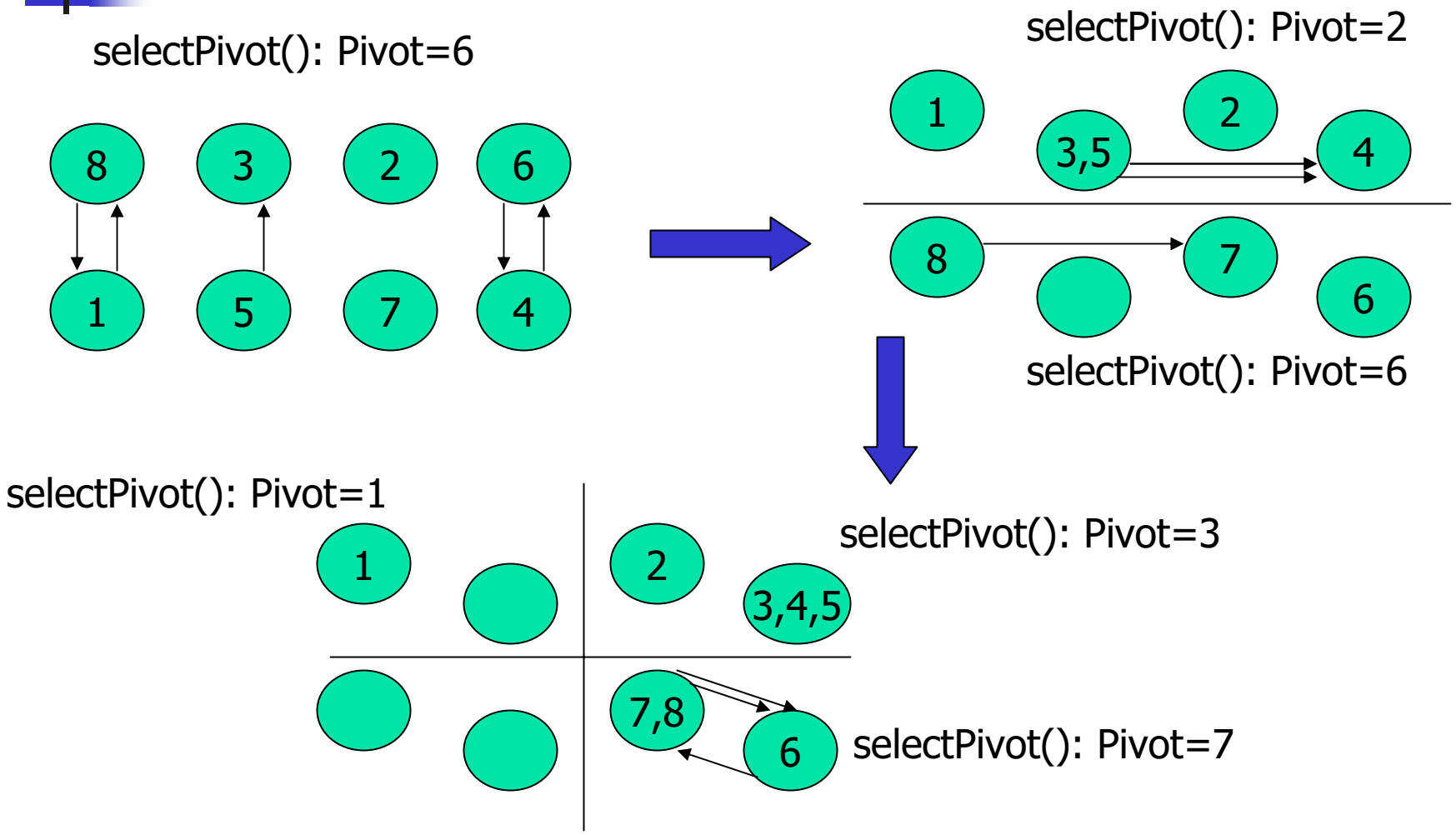
- Formal Virtual Model
- Work: Total No. of Ops
- Depth: Longest Chain of Sequential Dependencies
- Work – 15, $(n-1)$
- Depth – 4, $(\log_2 n)$
- Captures notion of parallelism
- Communication costs not directly accounted

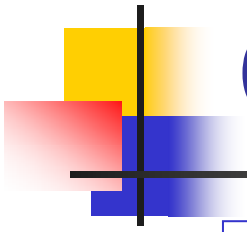
Simulation Model

- Processor: FIFO queue of tokens, Fixed Address Space
- Processor Array: vector of processor objects
- Tokens: function objects
- Communication Delays modeled by random token placement



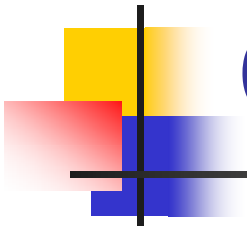
Quicksort Simulation





Quicksort Results

- Expected
  - Work = n log(n) = 24
  - Depth = log(n) = 3
  - Degree of Parallelism = 24/3 = 8
- Actual (Preliminary)
  - Work = 22
  - Avg Degree of Parallelism = 7
  - Max DOP = 8



Conclusions

- Proposed the Model of Computation
- Simulated the parallel architecture in C++ using function objects
- Implemented “quicksort” a highly recursive algorithm and evaluated the degree of parallelism
- Presently working on Dijkstra’s shortest path and more analysis