Is Teaching Parallel Algorithmic Thinking to High School Students Possible? One Teacher's Experience

March 12, 2010

Contact Information

- Shane Torbert, smtorbert@fcps.edu

  Thomas Jefferson High School for Science and Technology

  Fairfax County Public Schools, Fairfax County, VA

- Uzi Vishkin, vishkin@umd.edu

  The University of Maryland Institute for Advanced Computer Studies (UMIACS)

  College Park, MD 20742

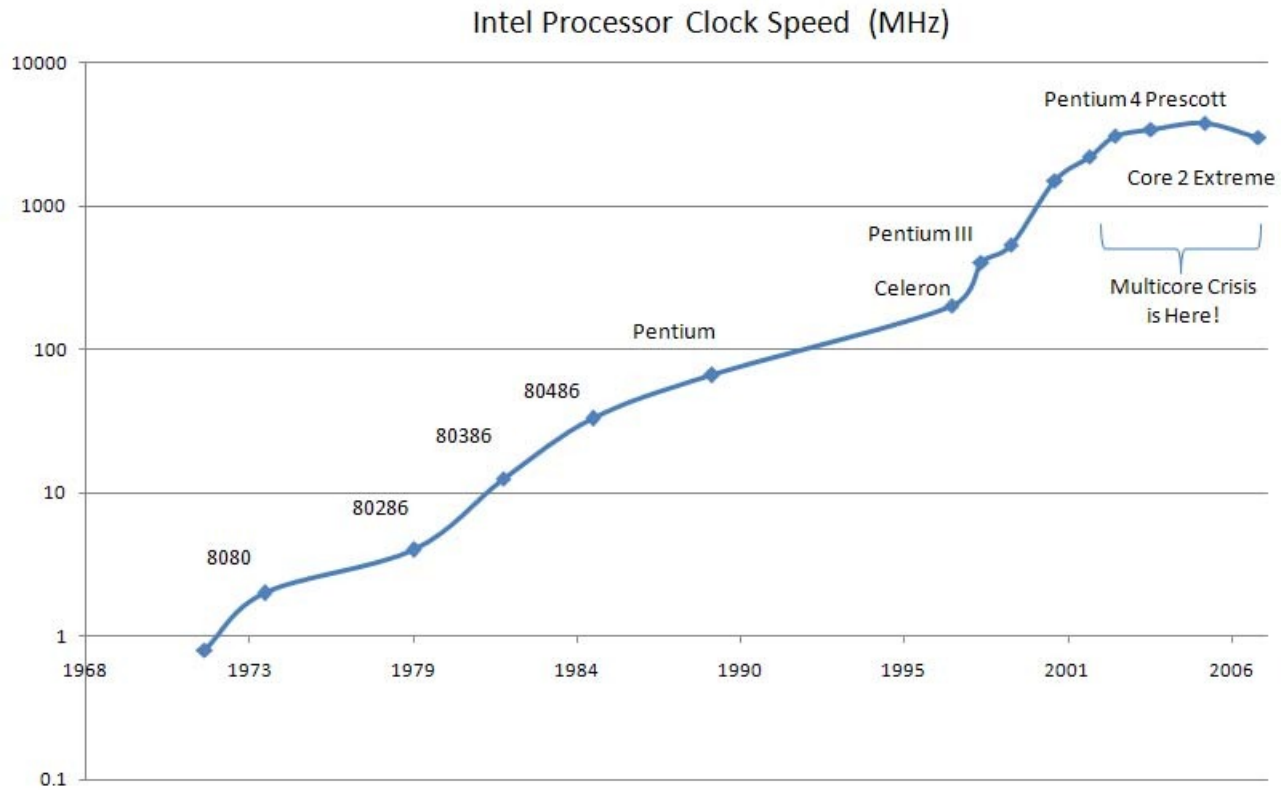
- Ron Tzur, ron.tzur@ucdenver.edu

  University of Colorado Denver

- David J. Ellison, djelliso@indiana.edu

  Indiana University Bloomington

At Issue



http://smoothspan.wordpress.com/2007/09/06

Future Outlook

- What does the future of Computer Science look like?
  - How are students attracted to the field?
- Broadly, what kinds of problems can computing solve?
  - Science and engineering.
  - Databases and media.
  - Communication and productivity.
- Narrowly, what activities will the students engage in?
  - Designing algorithms and writing code.
  - Implementation for large-scale cases $\Rightarrow$ **parallel**.

Explicit Multi-Threading (XMT)

- Programming in parallel by itself is not hard.
  - What is hard is programming for performance in parallel.
- Our insight is that coding in XMT as a parallel system is a more natural transition from serial than other alternatives:
  - MPI
  - PVM
  - OpenMP
  - pthreads
  - CUDA
- http://www.umiacs.umd.edu/~vishkin/XMT/

Background: Equipment at TJ

- Early 1990s, ETA supercomputer
- Late 1990s, Beowulf cluster
- 2003, Cray SV-1 supercomputer
- Mid 2000s, Linux workstations
- Currently
  - Itanium/Infiniband cluster
  - Intel I7 cluster with NVidia's CUDA
  - Sun SPARC Enterprise M4000 (4 CPUs x 2 cores x 2 threads)
  - Prototype Mac mini cluster

Background: Courses at TJ

- 1990s, one semester elective
  - C and C++
  - MPI and PVM
  - OpenGL and POV-Ray
- 2000s
  - C, Fortran, MPI, OpenGL
- 2008, two semesters
  - C, MPI, OpenGL
  - **XMT**, OpenMP, pthreads, CUDA, sockets

Programming Assignments

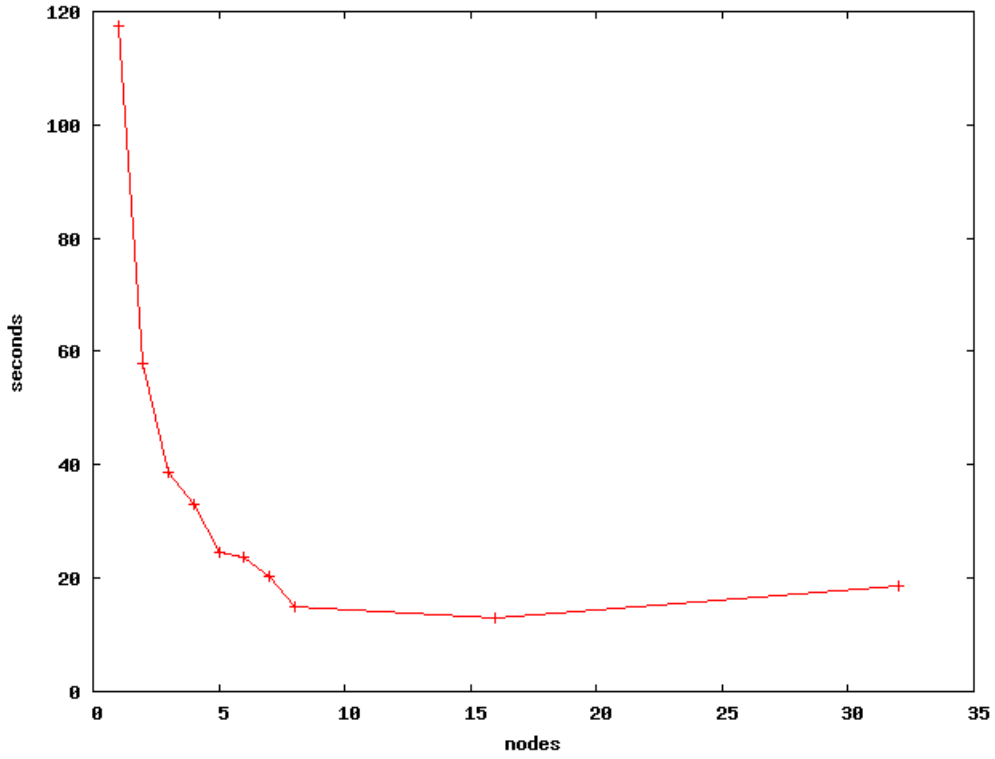
- Projectile Motion, air resistance
- Mandelbrot and Julia Sets, interactive zooming
- Diffusion Limited Aggregation, crystal growth
- Conway's Game of Life, cellular automata
- N-Body Gravitation, Newtonian mechanics
- Discrete Cosine Transform
  - Image Processing and JPEG lossy compression
- Solving Matrix Equations, iterative methods
- Ray Tracing

Communication Schemes

- Manager-Worker
  - Embarrassingly Parallel, independent operations
  - Parameter Search, ballistics vary wind and angle
  - Particle Diffusion, motion without collisions
- Nearest Neighbor
  - Conway's Game of Life, spatial decomposition
  - Heat Equation Simulation, simple averaging
- Round Robin
  - All pairs Particle-Particle interaction

Parallel Speedup

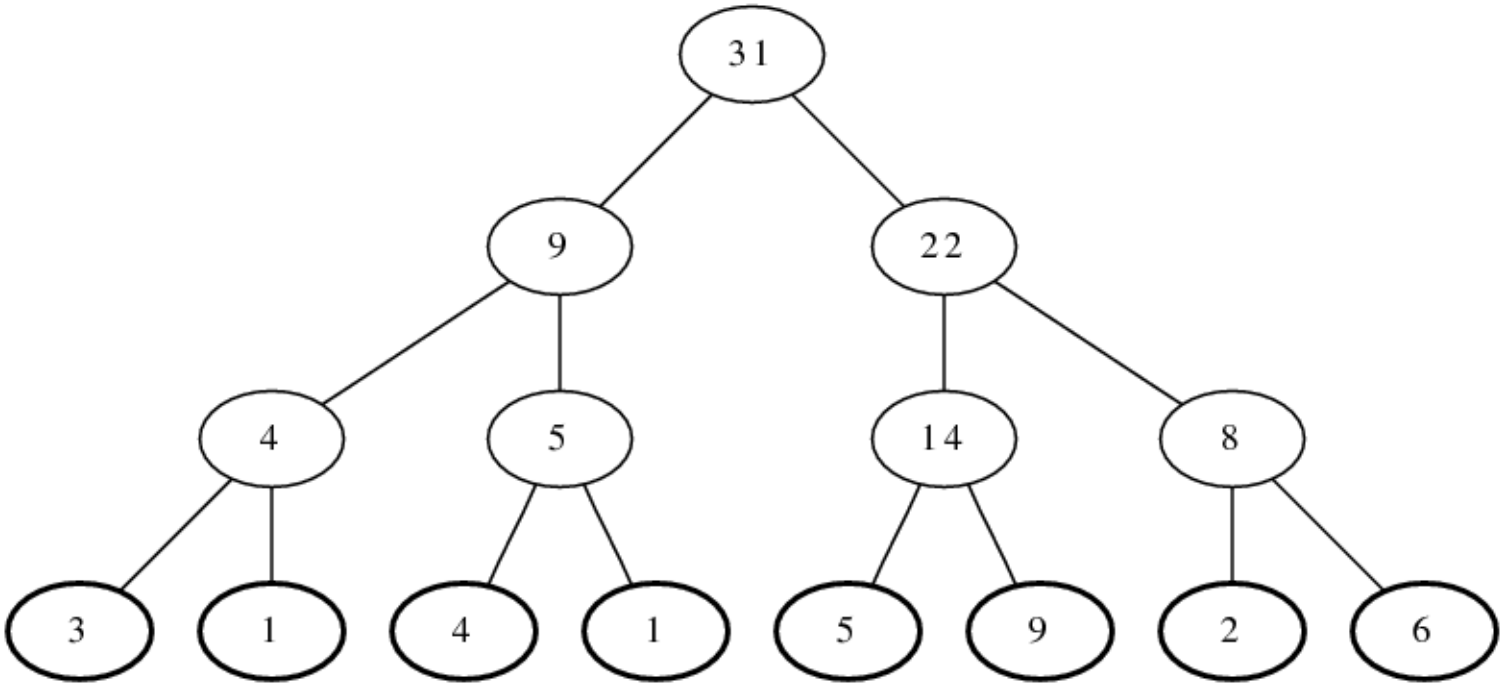
- Typical results on 100 Mbps non-dedicated system.



From Speedup to Formal Analysis with XMT

- Traditional Metrics
  - Timing data
  - Scale up the size of the problem
- Large-Scale Behavior
  - Fixed communication cost overhead
  - Availability of shared resources
- Big-Oh Analysis
  - Continuation from AP Computer Science relies on XMT
  - Measure both time and work (total operations) in parallel

Example: Simple Sum (1)



Example: Simple Sum (2)

```
// copying elements of an array to be summed A
// into the leaves of a balanced binary tree B

spawn(1,n)                  // for i, 1<=i<=n pardo
{
   int i;                   // index for left-to-right
   i=$;                     // XMT-C uses dollar sign
   B[0][i]=A[i];
}
```

- Independent Operations: Work is $O(N)$ but Time is only $O(1)$

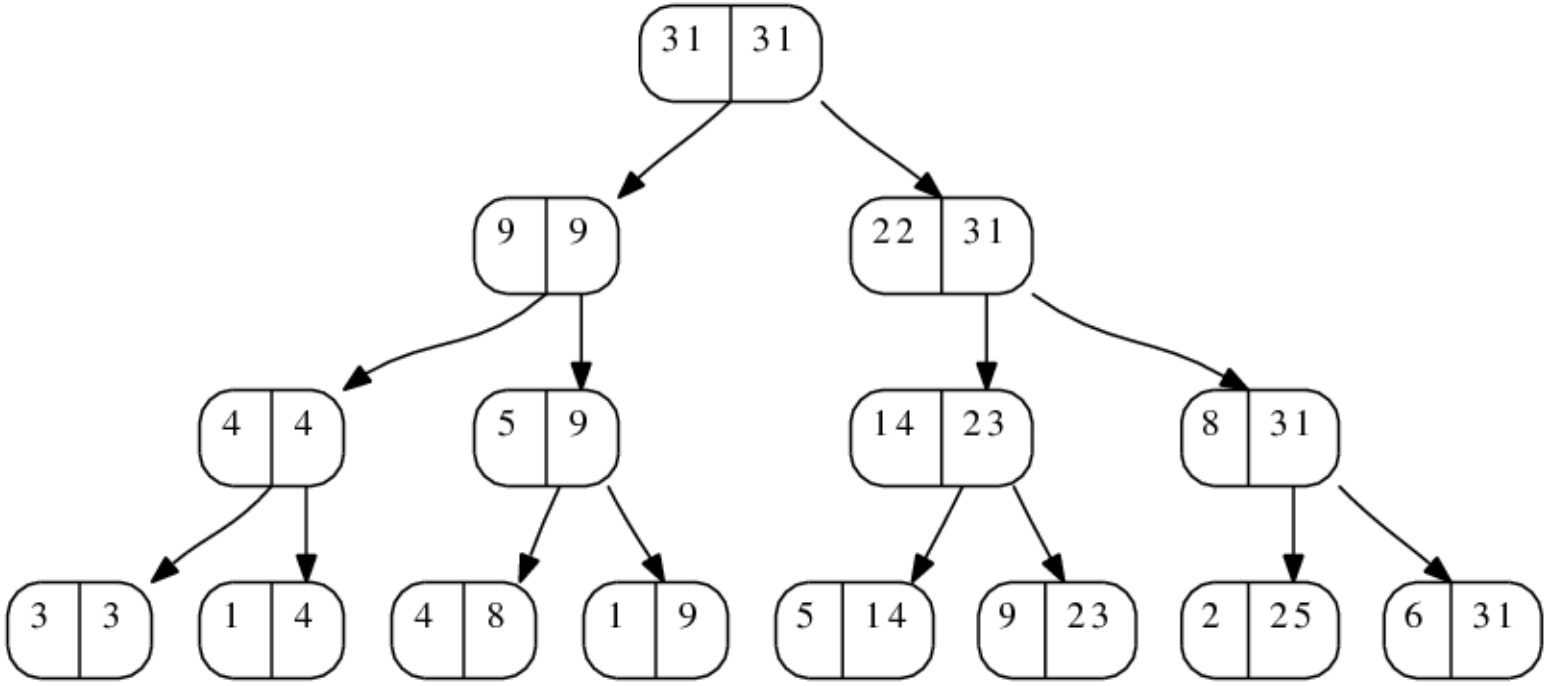
Example: Simple Sum (3)

```
// at each point in time p=2^h and n/p gives the
// number of nodes at level h of the binary tree

h=1;
for(p=2;p<=n;p*=2)   // move up the tree
{
   spawn(1,n/p)
   {
      B[h][$]=B[h-1][2*$-1]+B[h-1][2*$];
   }
   h+=1;             // h goes from 1 to log_n
}
```

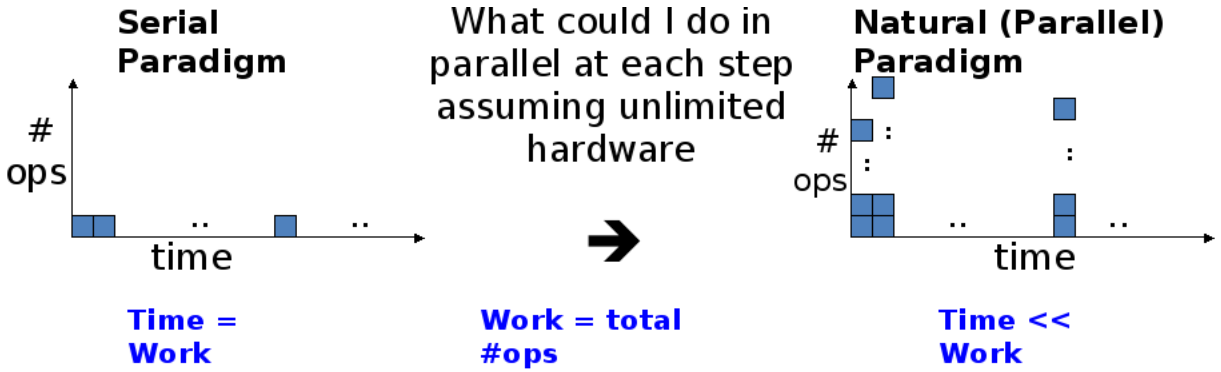
First Assignment: Prefix-Sum

- Overall the Work is $O(N)$ but Time is only $O(\log N)$.
- Add a top-to-bottom pass to also calculate prefix-sums.



Second Paradigm Shift

- First paradigm shift is from serial to parallel.
  - But, parallel paradigm of "decomposition first" is painful.
- Now shift within parallel to improve teachability $\Rightarrow$ **XMT**.
  - Express "what can be done in parallel" in a natural way.
  - Build both hardware and software around this model.



Other XMT-Enabled Programming Assignments

- Matrix Multiplication
  - Work is $O\left(N^3\right)$ but Time is only $O(\log N)$. Wow!!!
- Prefix-Min, find the smallest value up to each index.
- Compaction, given an auxiliary list of zeros and ones, keep only those values marked with a corresponding one.
- Nearest-One, find the closest one, but only looking backward.
- Segmented Prefix-Sum, auxiliary list again, reset the accumulating sum at each corresponding one.
- Student term: **micro-parallelism**

Project: Rank-Merge (1)

- Setup for a parallel mergesort requiring $O(N \log N)$ Work but only $O\left(\log^2 N\right)$ Time.
- Serial
  - Zipper merge requires $O(N)$ operations at each level.
  - Number of levels is $O(\log N)$, so Big-Oh is $O(N \log N)$.
- Parallel
  - Given two sorted lists, combine them into one sorted list.
  - First idea is to find the rank of each element in the other list.
  - Then, index in final list is current index plus this rank.

Project: Rank-Merge (2)

Index	Value	Rank	Index	Value	Rank
0	**1**	0	0	**2**	1
1	**4**	1	1	**6**	3
2	**5**	1	2	**8**	4
3	**7**	2	3	**9**	4

Final Index:	0	1	2	3	4	5	6	7
Sorted Value:	**1**	**2**	**4**	**5**	**6**	**7**	**8**	**9**

Project: Rank-Merge (3)

- Partial Solution
  - To find rank, binary search for each element in the other list.
- Binary Search
  - Independent Operations: $O(N \log N)$ work, $O(\log N)$ time
  - Once rank is known, move elements into final sorted list in only $O(N)$ work and $O(1)$ time.
  - For $O(\log N)$ total levels: $O(N \log^2 N)$ work, $O(\log^2 N)$ time
- Analysis
  - Time is much better than serial but work is worse, increased by a factor of $O(\log N)$ over original zipper merge version.

Project: Rank-Merge (4)

Index	Value	Rank	Index	Value	Rank
0	**1**	0	0	**2**	1
1	4	-	1	6	-
2	5	-	2	8	-
3	**7**	2	3	**9**	4

Final Index:	0	1	2	3	4	5	6	7
Sorted Value:	**1**	**2**				**7**		**9**

Project: Rank-Merge (5)

- Better Solution
  - Combine serial zipper merge with parallel binary search idea.
- Partitioning
  - Form $O(N/\log N)$ partitions each of size $O(\log N)$ elements.
  - Use binary search to rank the endpoints of these partitions.
- This takes only $O(N/\log N \times \log N) = O(N)$ work.
  - Then, zipper merge the other elements between the endpoints.
- This takes only $O(2 \times \log N) = O(\log N)$ time because, now, elements from independent partitions may be merged in parallel.

Teachability For All (1)

- Parallel rank-merge successful with XMT but not feasible to even try with MPI for high school students and time constraints.
- Elective freshman course at the University of Maryland in Spring 2009. Most of the 19 students were not Computer Science or Computer Engineering majors. Nearly all students were able to produce correct working XMT-C code that achieved satisfactory speed-ups for each of the assignments.
- Montgomery Co. Public Schools, middle-school summer camp for underrepresented students. The class met nine mornings and students designed parallel algorithms using XMT.

Teachability For All (2)

- Baltimore Polytechnic Institute, a majority African-American high school. The class met bi-weekly over two months and was offered to 11th grade students. Activities included constructing parallel algorithms and quantifying their time complexity.

Thank you.