NESL: A Language for Teaching Parallel Algorithms

Guy Blelloch and Jonathan Hardwick

Carnegie Mellon University

SCANDAL group: Blelloch, Chatterjee, Greiner, Hardwick, Narlikar, Reid-Miller, Sipelstein, Zagha

Teaching Parallel Algorithms

Lots of parallel algorithms in the literature:

Mostly expressed in high-level models such as PRAM.

Good for teaching algorithms.

Lots of parallel programming languages:

Either too high level (performance implications are unclear),

or too low level (implementation details hide the algorithm).

We need a language that bridges the gap.

NESL: An Algorithmic Language

Easy to program high-level algorithms:

NESL code closely matches pseudocode for algorithm.

Well-defined performance model:

Language-based complexity model (work-time).

Interactive environment:

Mathematica-like environment, transparent use of machines.

(Reasonable performance:)

Happy if we're within a factor of 2 of machine-specific code.

Current Status

Language runs on:

Workstations, Cray , MasPar , CM-5, MPI (Paragon, SP-2, ...)

Environment:

X11 interface, program tracing and timing tools.

Sample algorithms:

Graph:	Connected components, MST , BFS
Geom.:	Delaunay triangulation, convex hull
Matrix:	Sparse matrix operations, FFT s
Sorting:	Quick, merge (bitonic, odd-even), sample, radix
N-body:	Greengard, Barnes-Hut, hybrid
...	

Language Features

Nested data-parallel:

Combines advantages of data-parallel and task-parallel:

Single thread of control (easy to understand and debug).

Easy to express algorithms operating on irregular data.

Complexity model:

"Work" and "depth" complexity for each language construct.

Rules for combining complexities to find algorithm complexity .

(Also functional:)

Side-effect free, memory management, first-class functions, ...

Data-Parallelism In NESL

Elementwise operations:

```
A = [3.0, 1.0, 2.0]

{x + 0.5 : x in A}  -> [3.5, 1.5, 2.5]


B = "Yoknapatawpah"

{c in B | c < 'n}   -> "kaaaah"
```

Aggregate operations:

```
sum(A)              -> 6.0

rank(A)             -> [2, 0, 1]
```

Nested Data Parallelism In NESL

Nested sequences and operations:

```
C = ["the", "rain", "in", "Spain"]
C[3]                  -> "Spain"
{#w : w in C}         -> [3, 4, 2, 5]

D = [[4, 5, 1, 6], [2], [8, 11, 3]]
{sum(s) : s in D}   -> [16, 2, 22]
permute(D,[2,0,1]) -> [[2], [8, 11, 3],[4, 5, 1, 6]]
```

Quicksort Pseudocode

```
procedure QUICKSORT(S);
if S contains at most one element then return S
else
  choose an element a randomly from S;
  let S1, S2, and S3 be the sequences of elements in
    S less than, equal to, and greater than a;
  return (QUICKSORT(S1) followed by S2 followed by
          QUICKSORT(S3))
end
```

From Aho, Hopcroft, and Ullman, “The Design and Analysis of Computer Algorithms”, Addison-Wesley, 1974.

Quicksort in NESL

```
function quicksort(S) =
if (#S <= 1) then S
else
  let a  = S[rand(#S)];
      S1 = {e in S | e < a};
      S2 = {e in S | e == a};
      S3 = {e in S | e > a};
      R  = {quicksort(v): v in [S1,S3]};
  in R[0] ++ S2 ++ R[1];
```

Complexity: Work and Depth

Work = serial complexity (time on one processor).

Depth = parallel complexity (time on unlimited processors).

Example: adding two vectors

Work = 16, Depth = 1

Example: summing a vector

Work = 15, Depth = 4

Combining Complexity

We use Brent's lemma for a PRAM to find total time:

$$T = O\left(\frac{W}{P} + D\right)$$

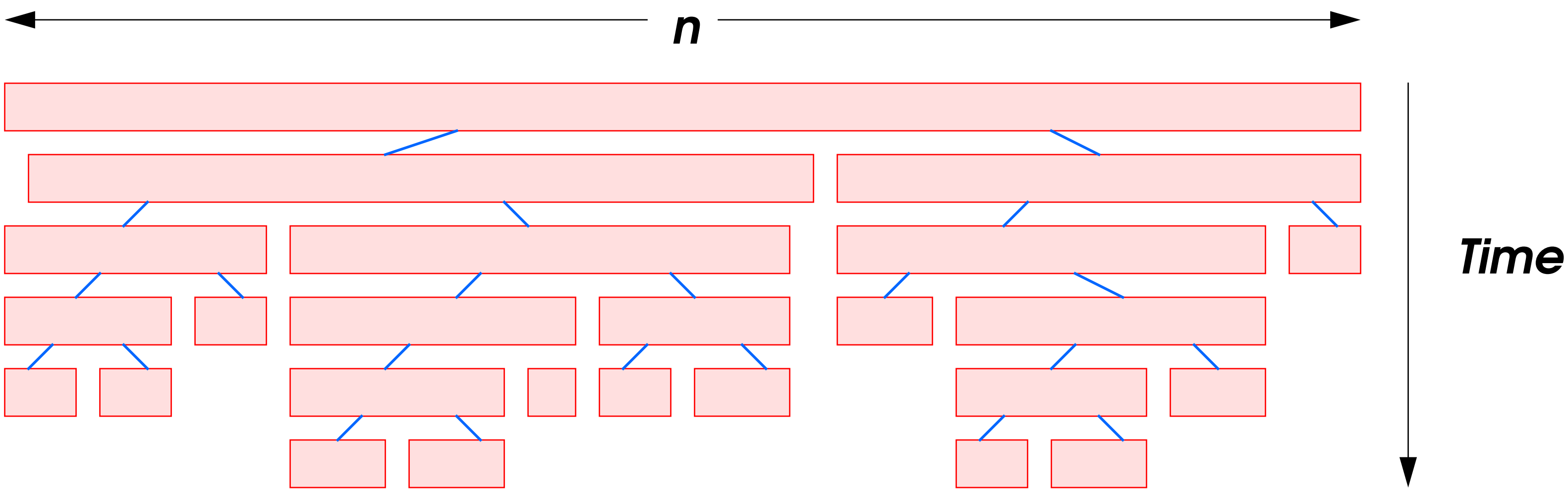
Statement complexities are combined using simple rules:

```
pexp = {exp(e): e in A}
```

$$W_{pexp}(A) = \sum_{i=0}^{n-1} W_{exp}(A_i)$$

$$D_{pexp}(A) = max_{i=0}^{n-1} D_{exp}(A_i)$$

Analysis of NESL Quicksort



Expected case:

$$
\begin{aligned}
\text{Depth} &= O(\log\ n) \\
\text{Work} &= O(n\ \log\ n)
\end{aligned}
$$

Teaching NESL

Used in CMU parallel algorithms classes for 3 years.

Emphasis on teaching work-efficient algorithms.

Sample student projects:

- Cole's mergesort
- Parallel expression evaluation
- Deterministic list ranking
- 3D volume rendering
- Comparison of connected component algorithms
- Image analysis
- Maximum-flow algorithms
- Speech recognition
- ...

Summary

Main conclusions of teaching NESL:

- Data parallel languages are easy to understand and teach.
- Nested data-parallelism greatly increases the number of algorithms that can be expressed.
- Language-based complexity model makes analysis easy .
- Interactive environment encourages experimentation.

Further Details

NESL system, demos, and papers are available via WWW :

`http://web.scandal.cs.cmu.edu/www/home.html`