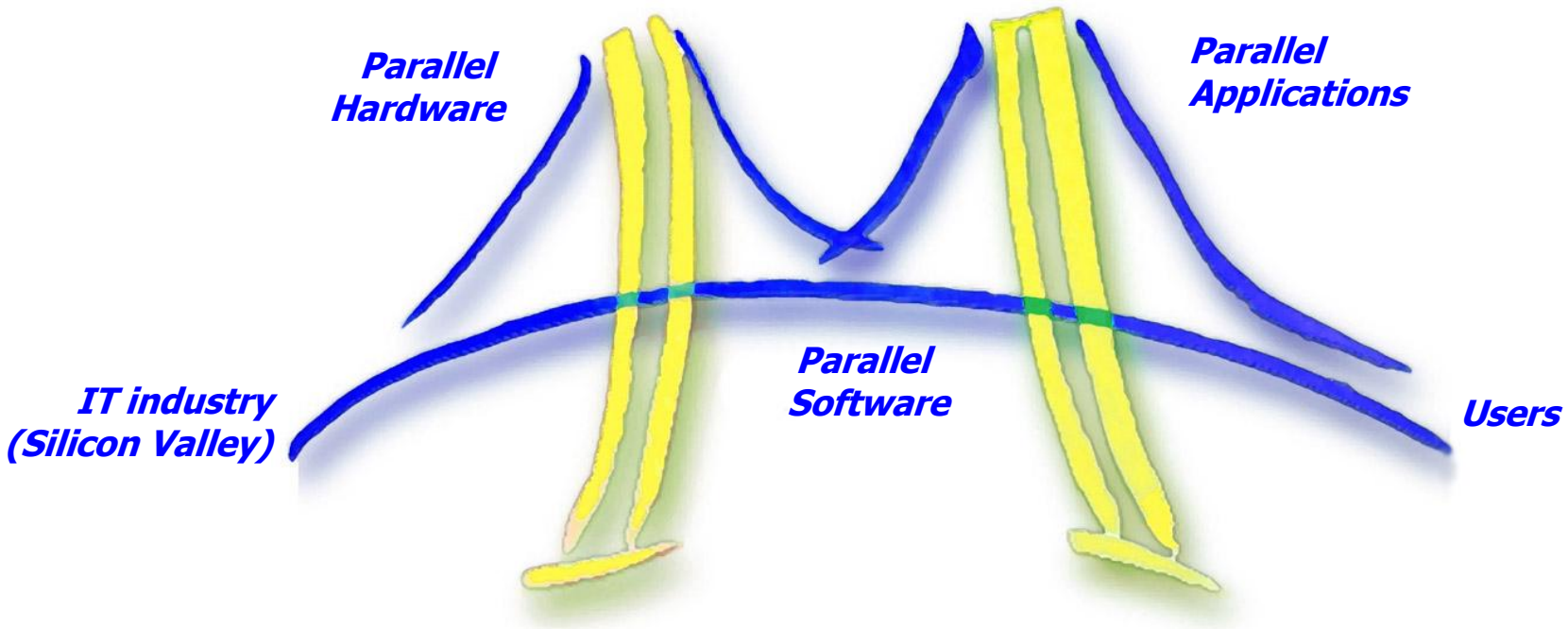
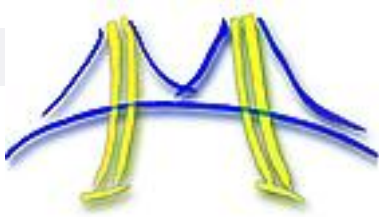


Parallel Layout

Automatic Generation and Optimization

01.14.2011
Leo Meyerovich
Adam Jiang
Rastislav Bodik

Why Generate a Layout Engine



Many and Growing Layout Languages

HTML, CSS, SMIL, XUL, Ext, jQuery, YUI OpenOffice, JavaFX, Swing, Flex,

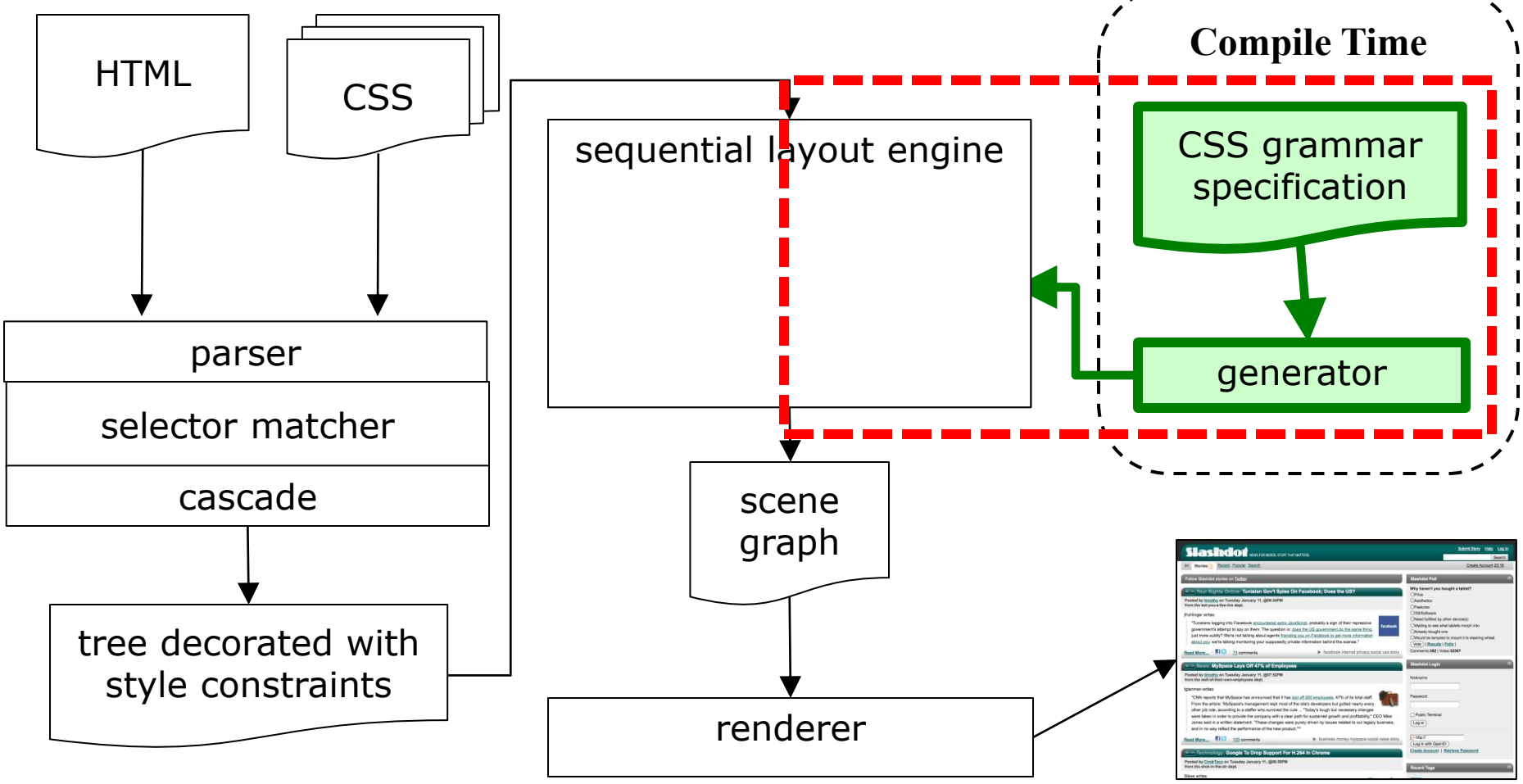
Adam & Eve, Thermo, XAML/WPF, Word, WinForms, Qt, LaTeX, music,

iPhone, Android, WAP,	MathML,	3 competing CSS *grid* proposals

A lot of code

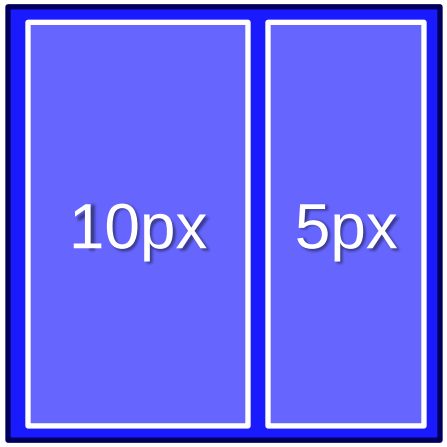
Firefox layout engine:
346111 lines

Approach

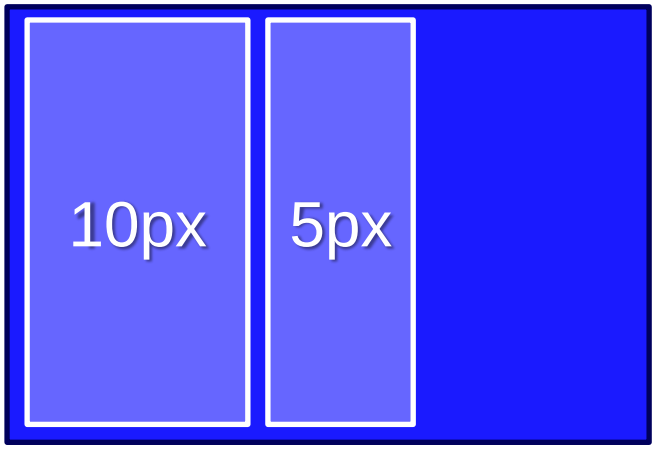

HTML
CSS
parser
selector matcher
cascade
tree decorated with style constraints
sequential layout engine
scene graph
renderer
Compile Time
CSS grammar specification
generator

HBox: Example of Specifying Layout

HBox with two **child nodes**



wInput = ShrinkToFit



wInput = 50px

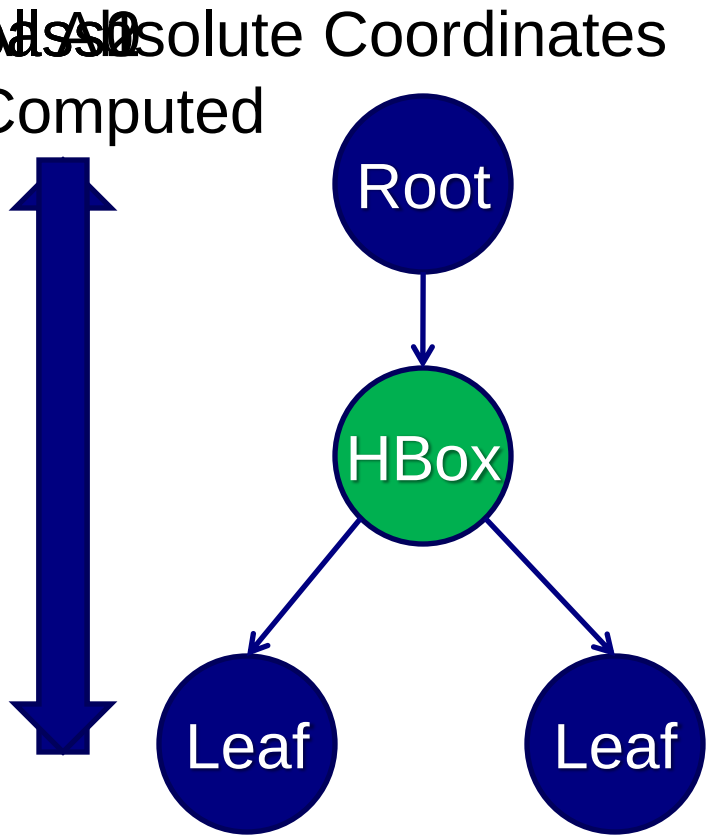
w := case wInput:

n px: n

shrinkToFit: sum (children.w)

HBox Traversal Functions

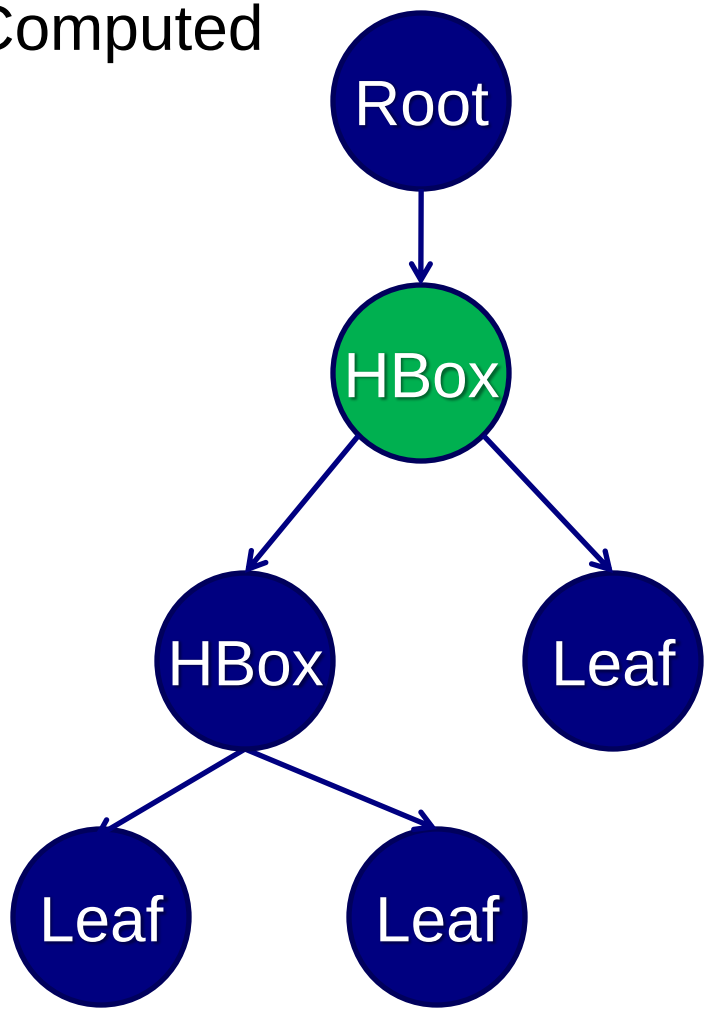
```
def pass0():
    child1.y = y
    child2.y = y
def pass1():
    cursor = child1.w
    w = if (wInput is shrink):
            child1.w + child2.w
        else:
            wInput
    h = max (child1.h, child2.h)
def pass2():
    child1.x = x;
    child2.x = x + cursor;
```



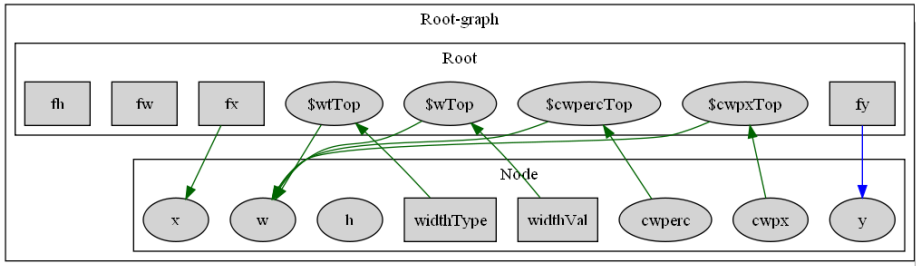
HBox Traversal Functions

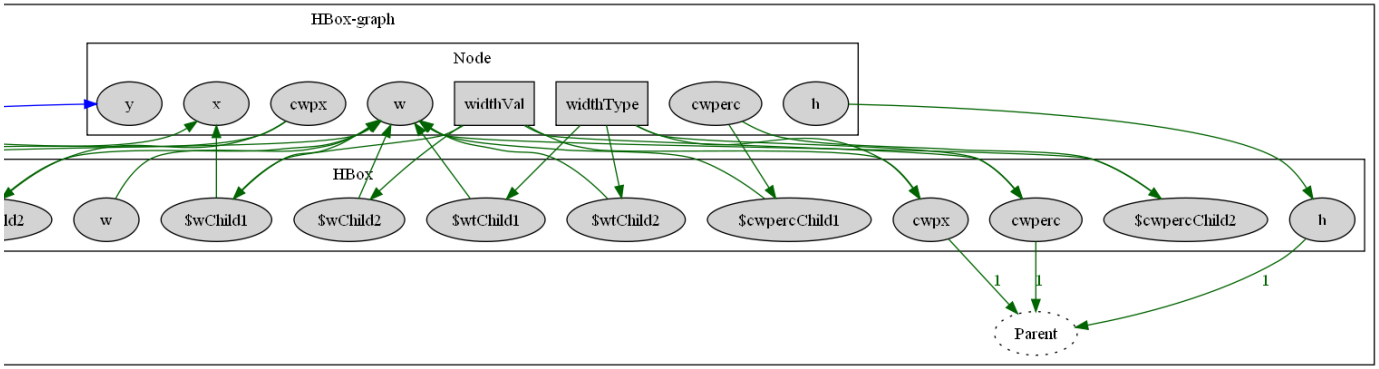
```
def pass0():
    child1.y = y
    child2.y = y
def pass1():
    cursor = child1.w
    w = if (wInput is shrink):
            child1.w + child2.w
        else:
            wInput
    h = max (child1.h, child2.h)
def pass2():
    child1.x = x;
    child2.x = x + cursor;
```

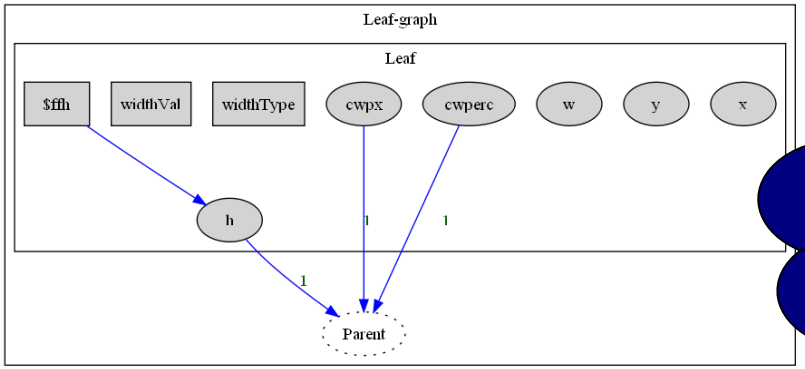
All Absolute Coordinates Computed



Scheduling Traversals

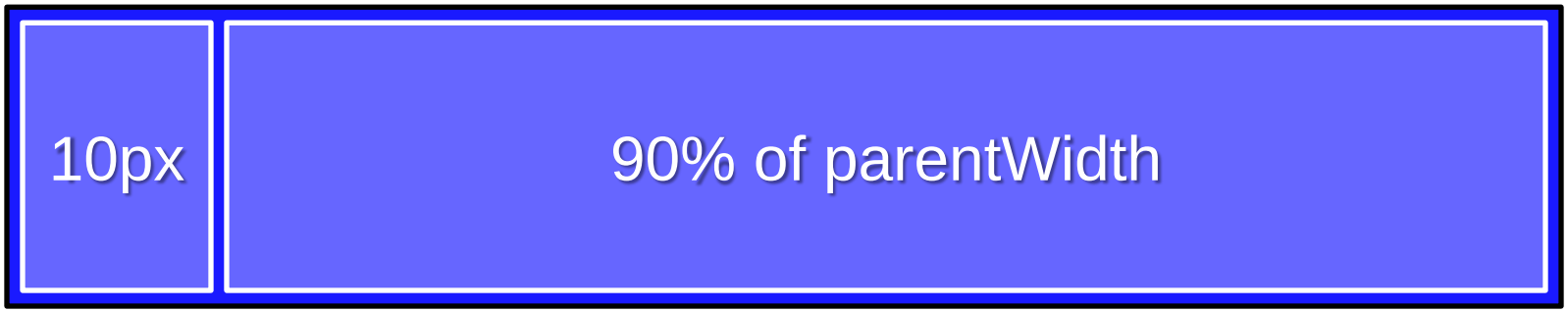




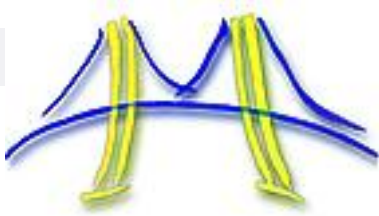


See Demo @ Poster Session

Leveraging Generation: % Width




```
w := case wInput:
        n px:          n
        n %:           parentWidth * n%
        shrinkToFit:   sum(children.w)
```



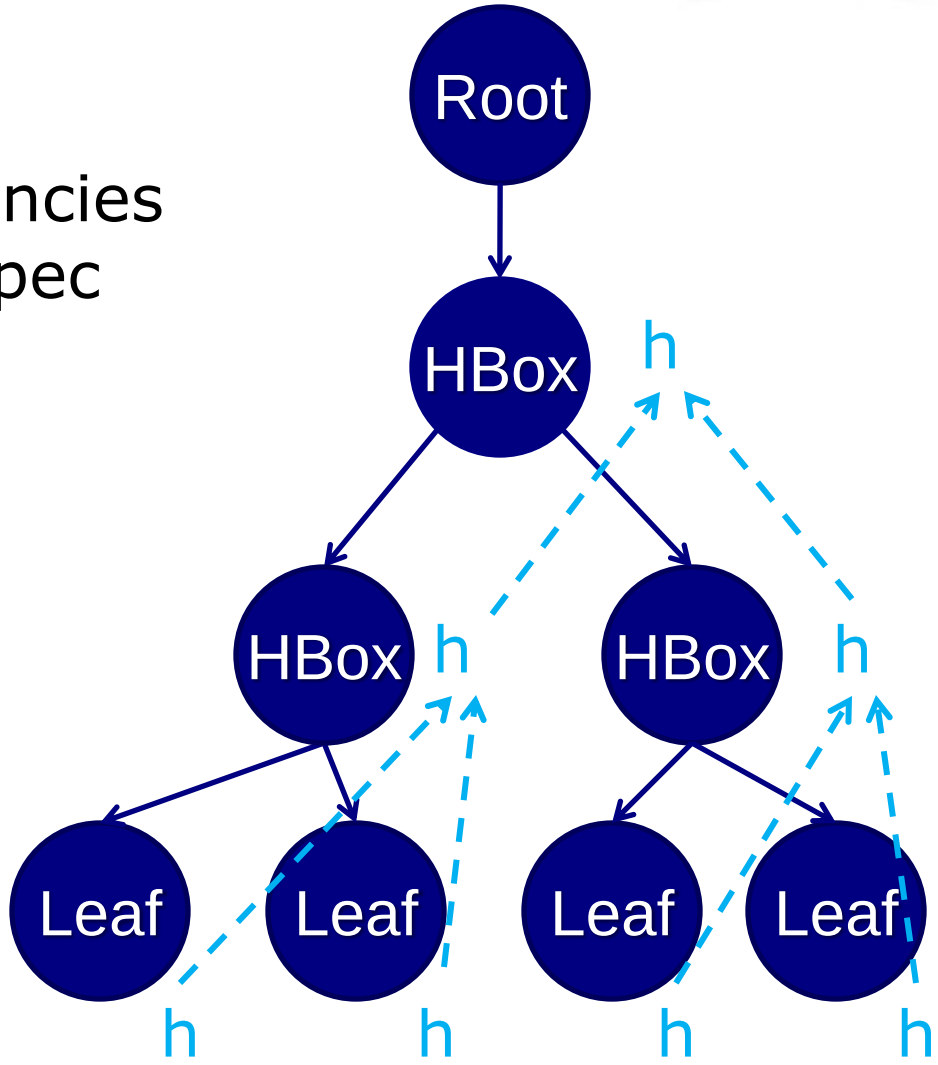
```
def pass0(): …
def pass1():
   cursor = child1.w
   w = calculateWidth(wInput,
            child1.w, child2.w)
   h = max (child1.h, child2.h)
```

```
def pass2():
    child1.x = x
    child2.x = x + cursor
```

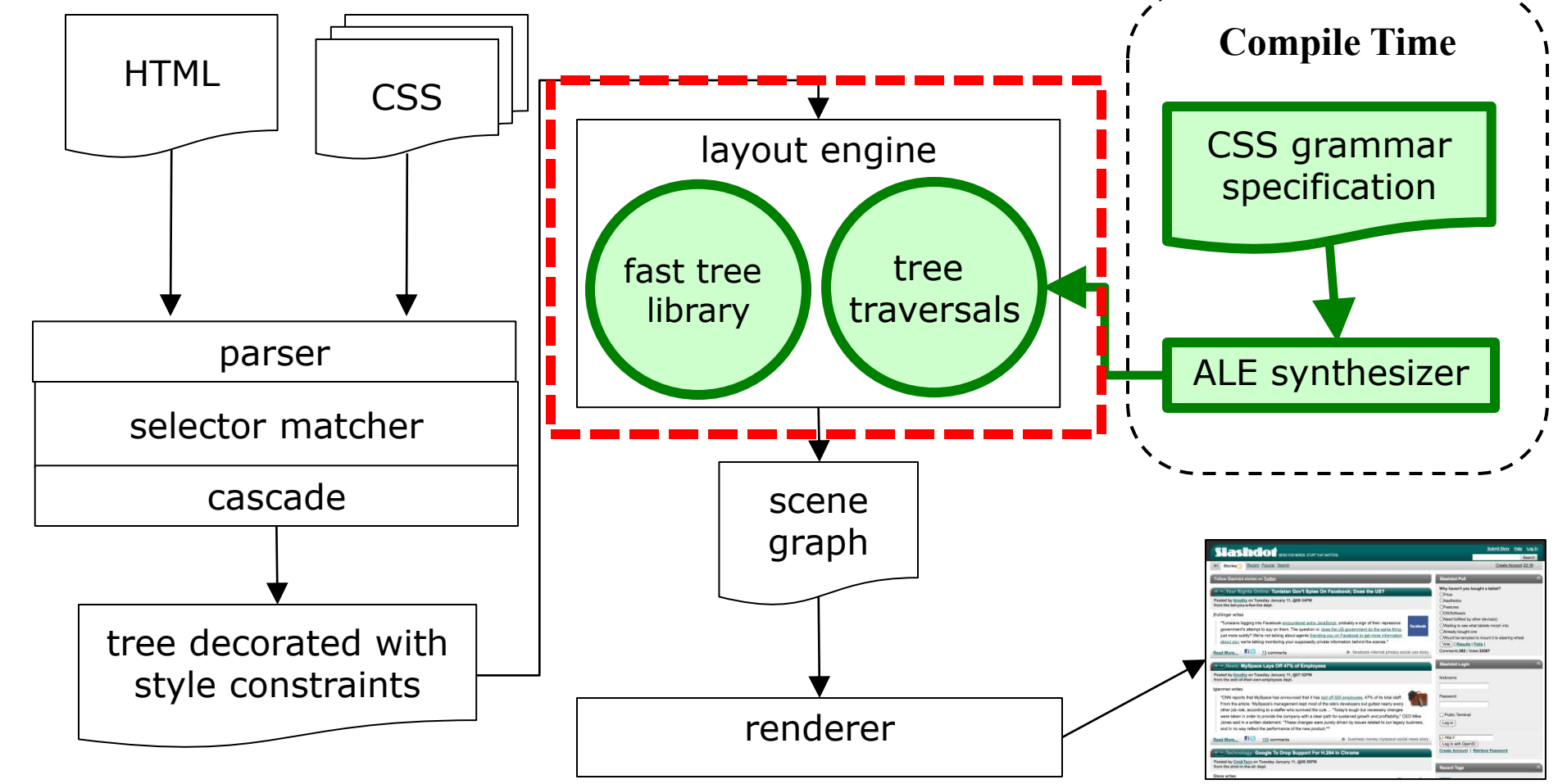
```
def pass0(): …
def pass1():
    cwpx =sumPx(children)
    cwperc = sumPercs(children)
    h = max (child1.h, child2.h)
def pass2():
    w = calculateWidth(wInput,
            cwpx, cwperc,
            parentWidth)
    child.parentWidth = w
def pass3():
    cursor = child1.w
def pass4():
    child1.x = x
    child2.x = x + cursor
```

Other Advantages

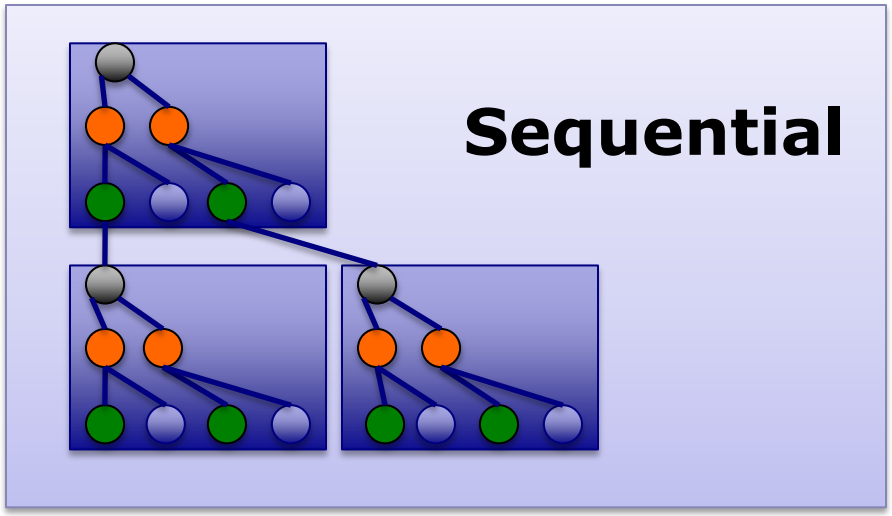
- Correctness Wins
  - Finds spec inconsistencies
  - Can visually debug spec
- Performance Wins
  - Optimal scheduling
  - Extract parallelism

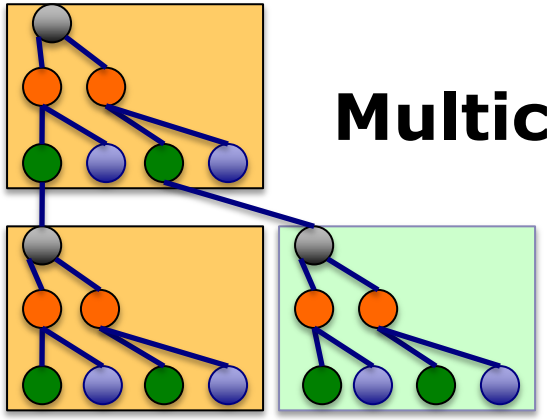


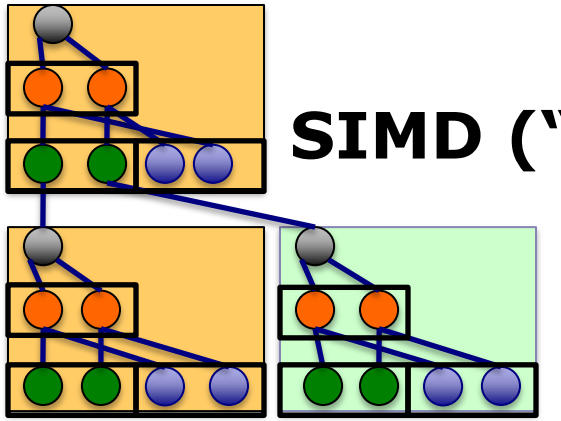
Fast Tree Library



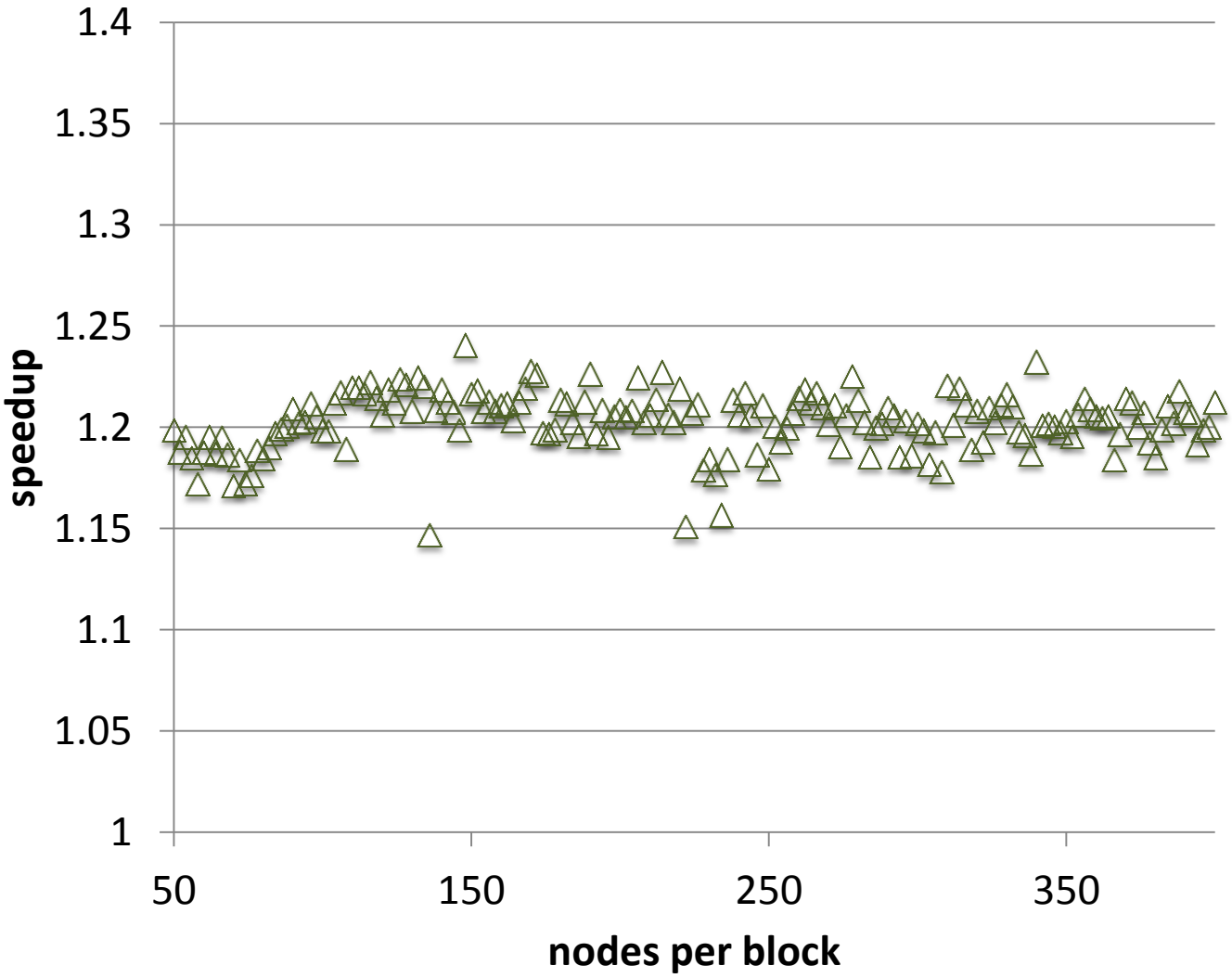
Overview of Tree Eval Strategies



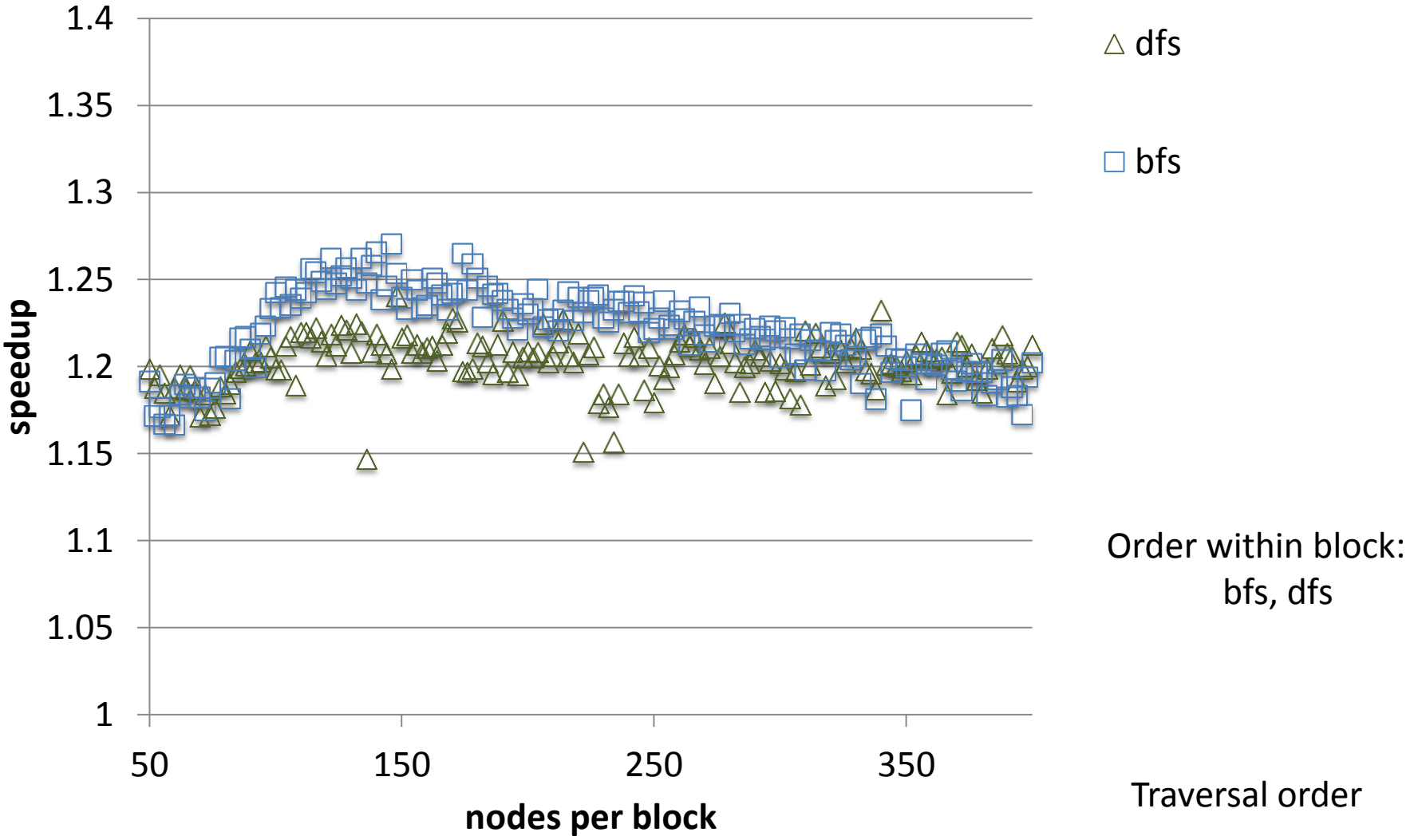




2. Optimizing memory



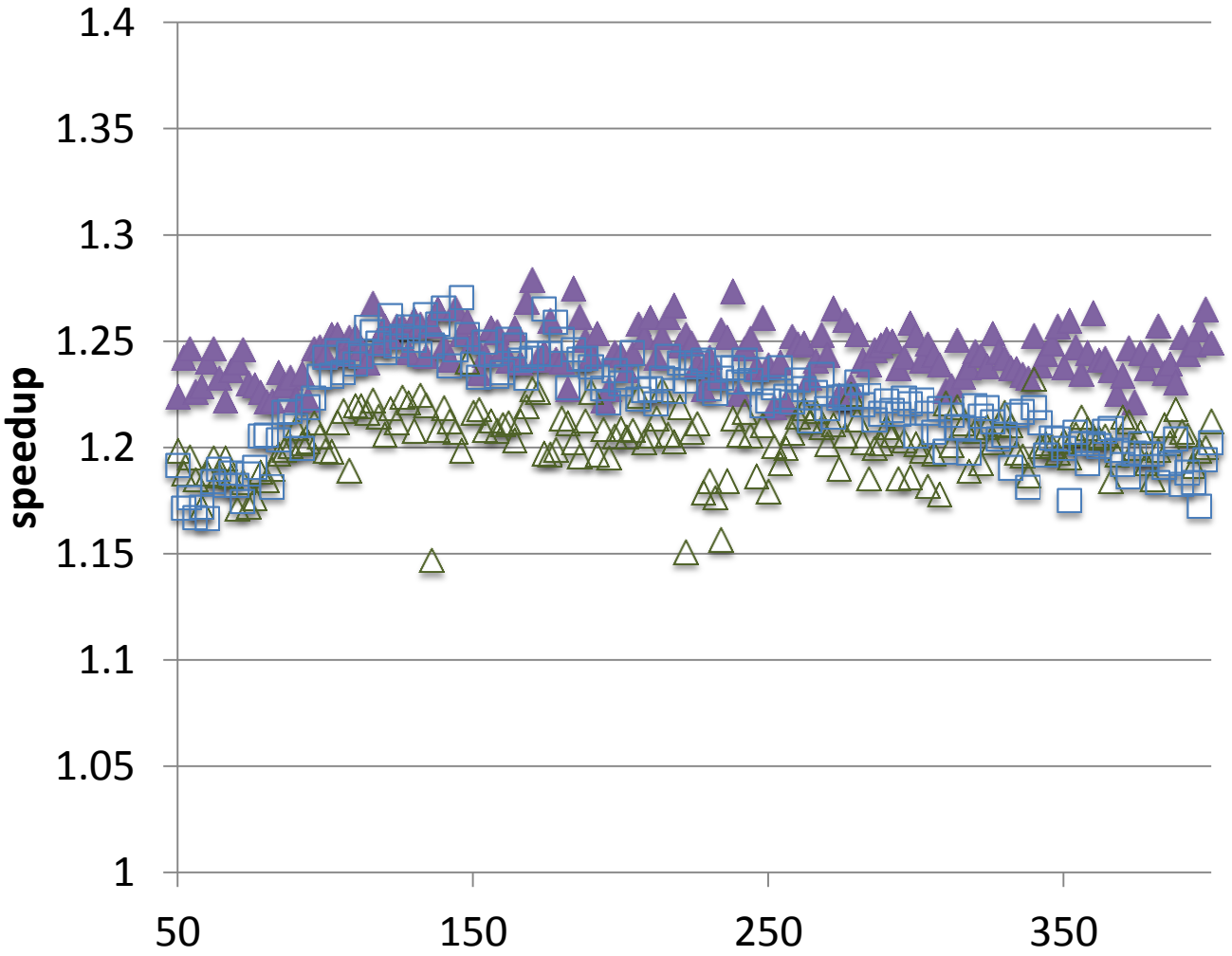
2. Optimizing memory



Order within block:
bfs, dfs

Traversal order

2. Optimizing memory



Order within block:
bfs, dfs

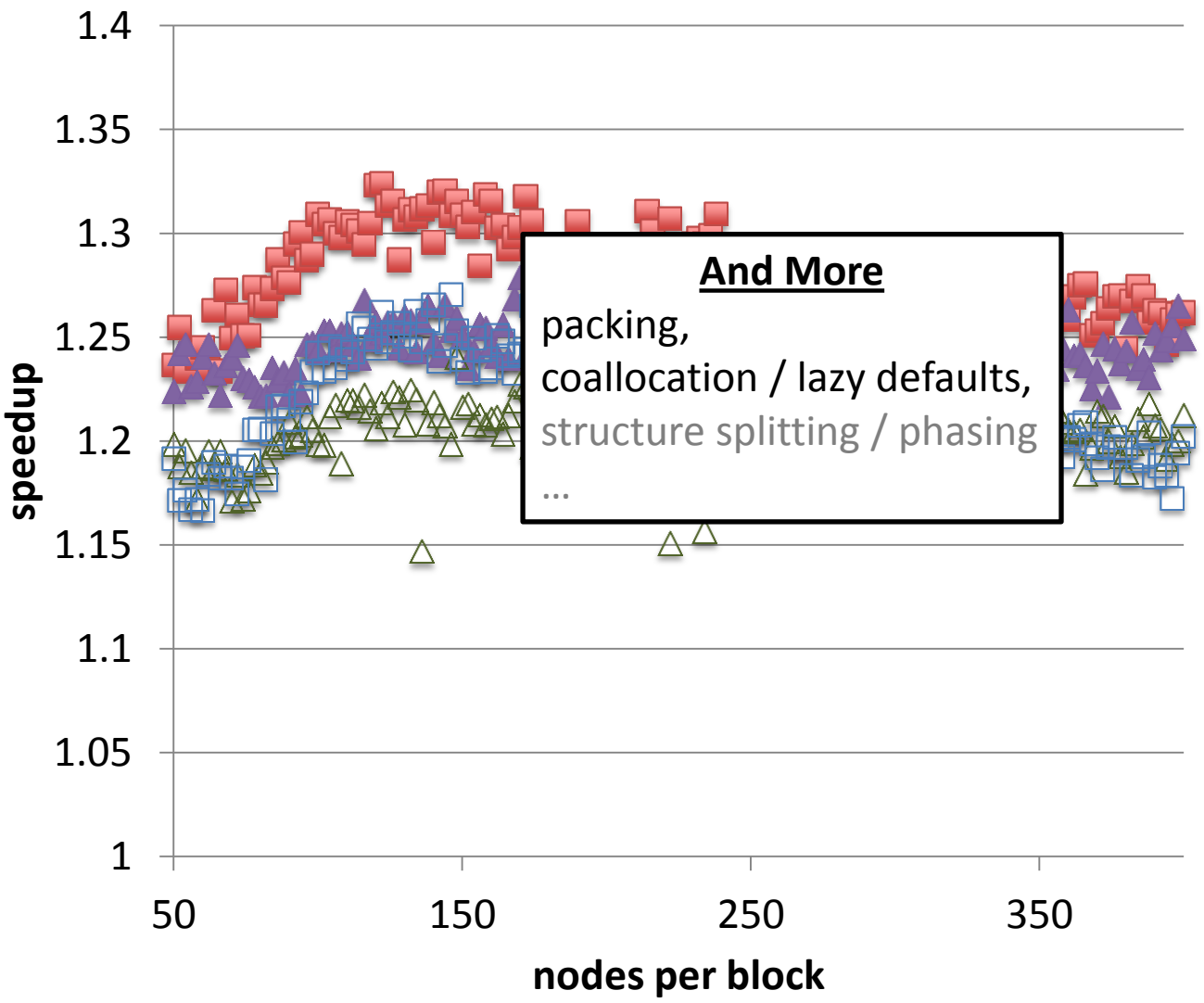
Pointer representation:
leftChild = 0x00ffaa00,
leftchild = 1200

hardware

How much compression

Pointer compression

2. Optimizing memory



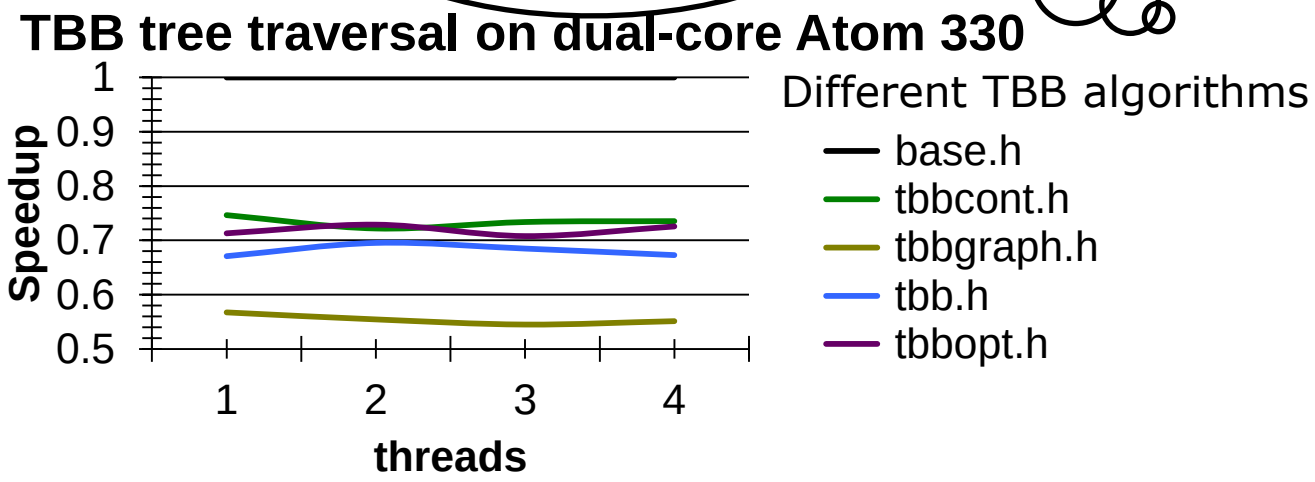
Order within block:
bfs, dfs

Pointer representation:
leftChild = 0x00ffaa00,
leftchild = 1200

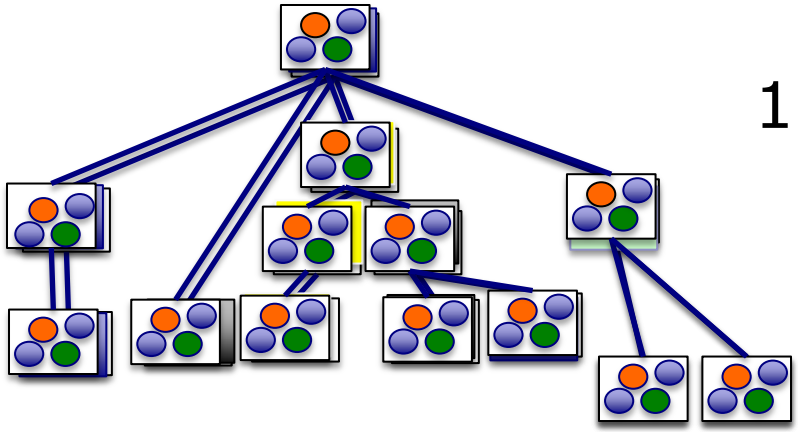
Challenge Problem for Task Parallelism?

- Se… …s

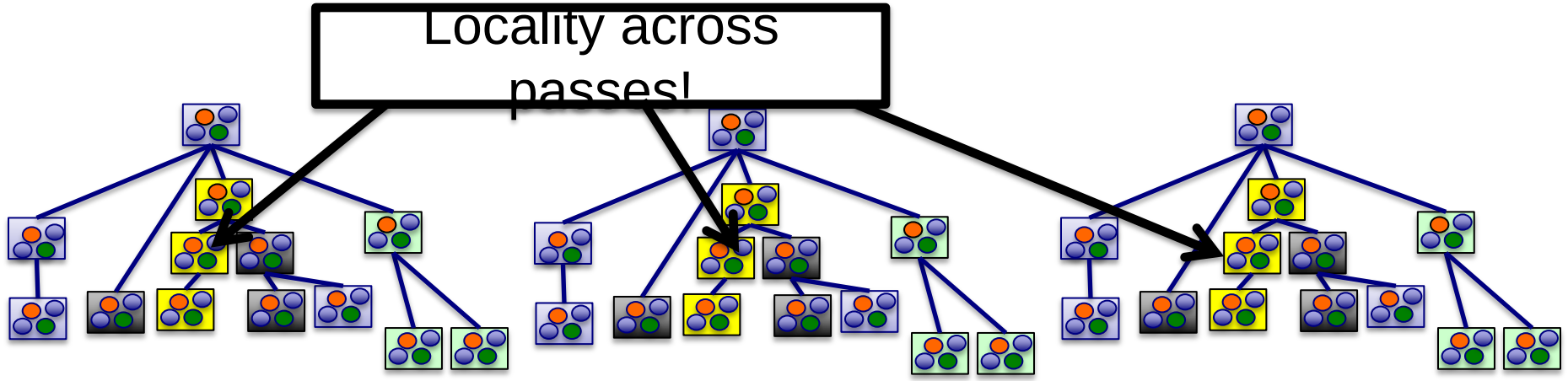
Dynamic task allocation?
Runtime queues?
Locality across traversals?



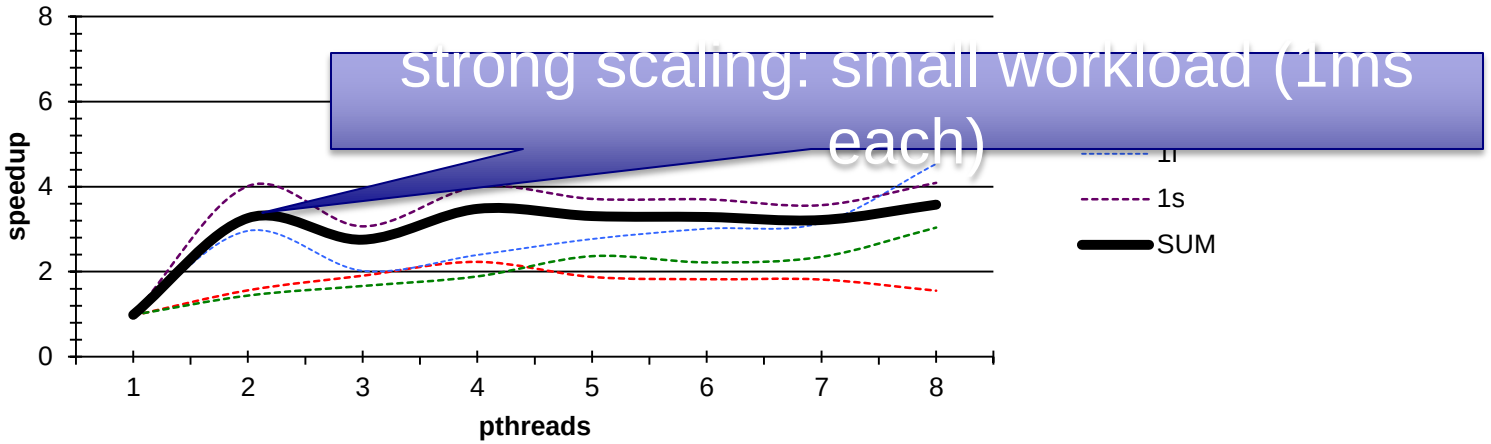
Semi-Static Work Stealing



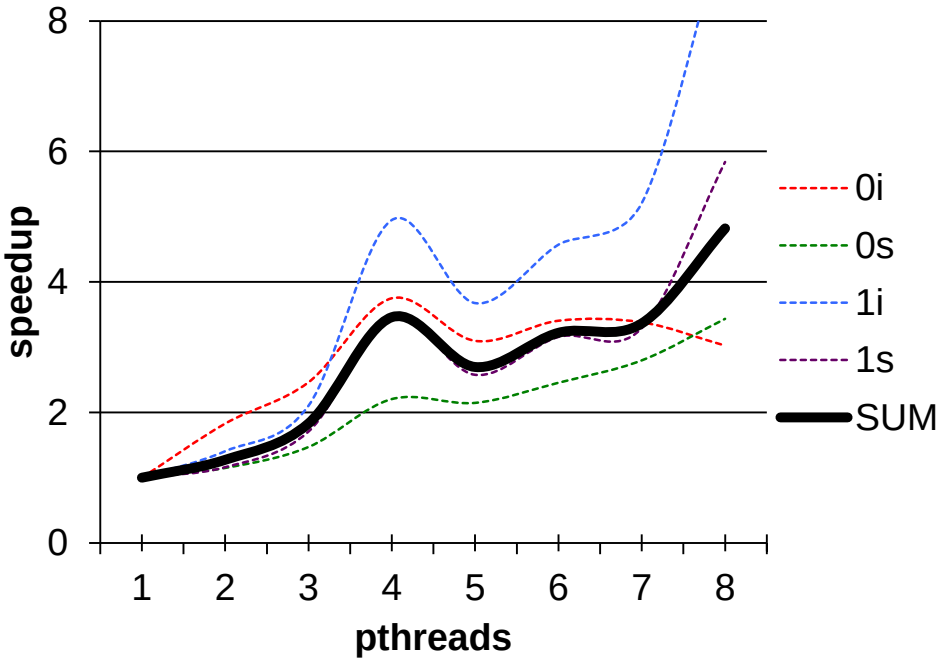
1. ***Before parallel traversal:*** approximate work stealing schedule
2. ***Traversal*:** reuse schedule tuned locking scheme



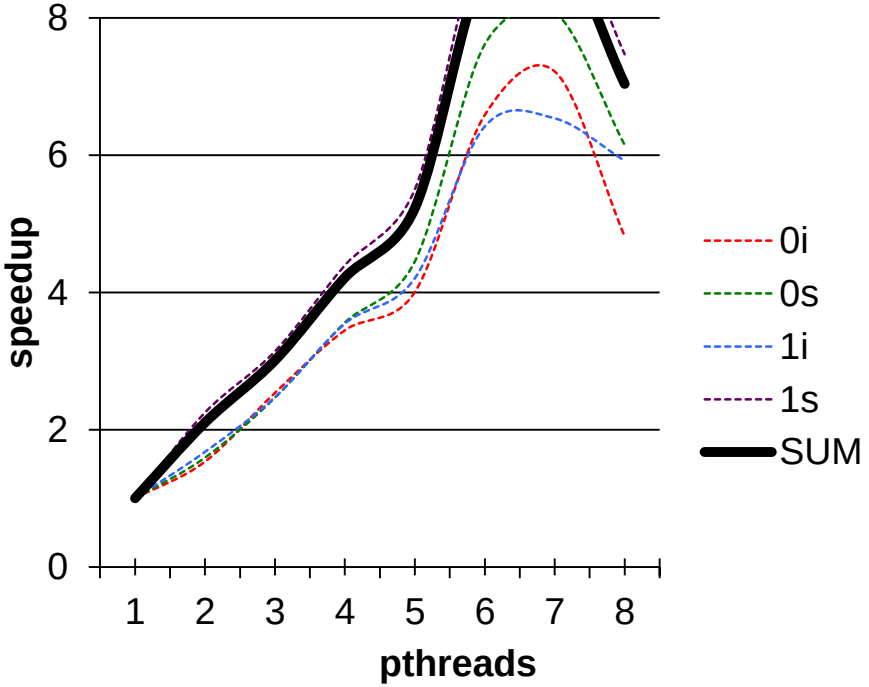
Opteron Speedup
(2 sockets x 4 cores); 1,000 nodes



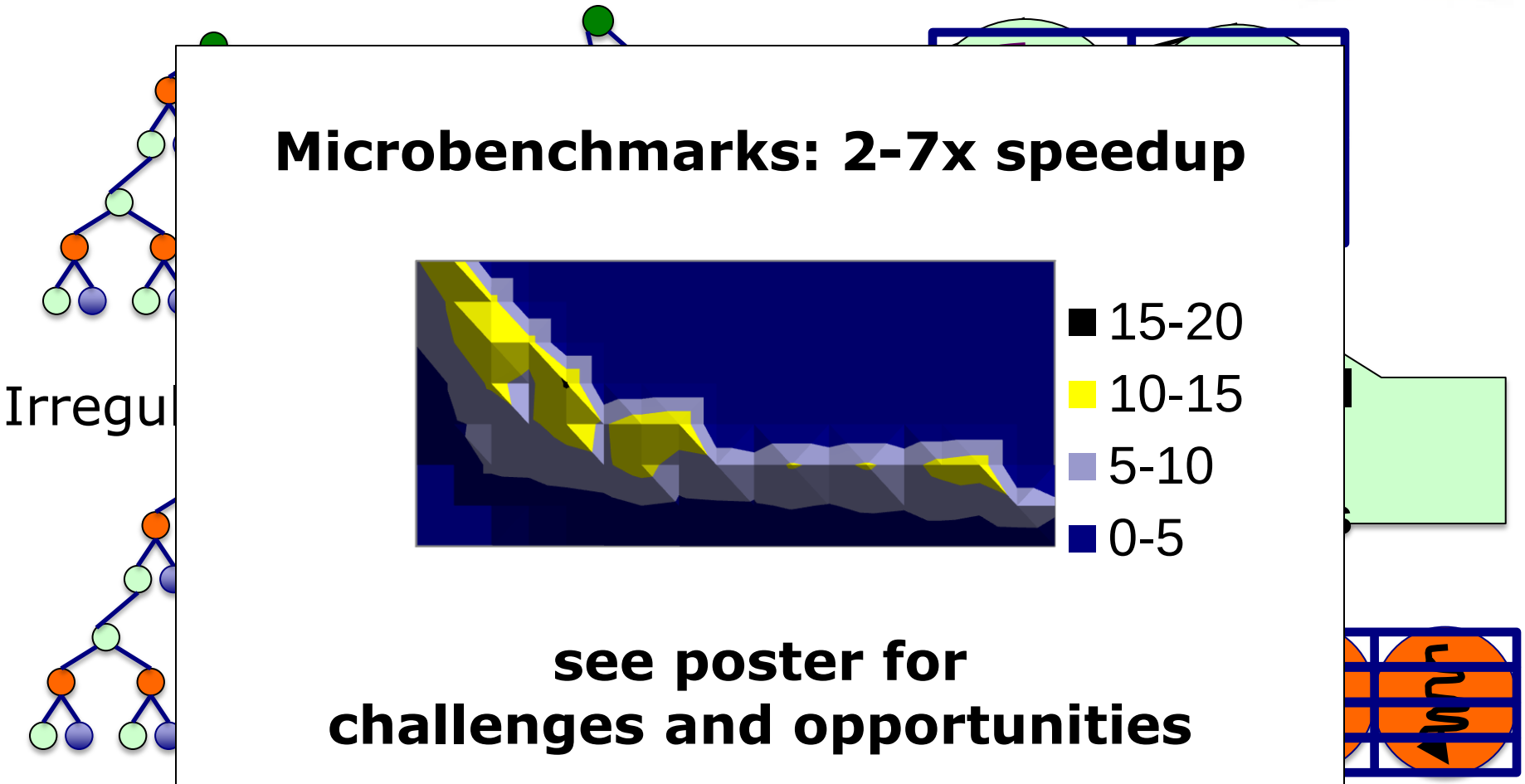
10,000 nodes



1,000 nodes; repeat each 10x



SIMD Task Evaluation (MSR)



Microbenchmarks: 2-7x speedup

see poster for challenges and opportunities

Demo

- Parallel layout

Status and Future Work