

Plenty of Parallelism

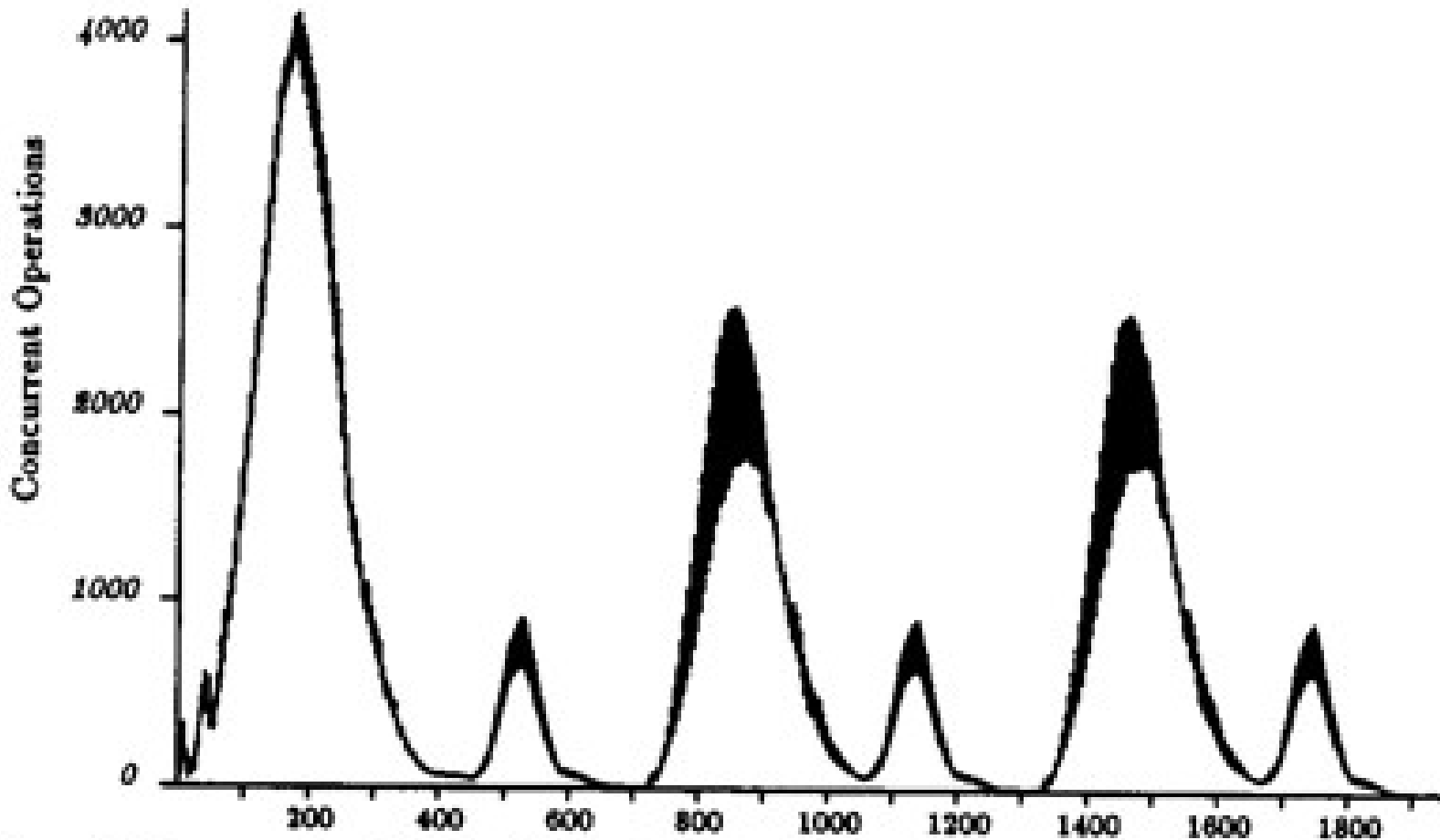


Figure 7: ($p = \infty, l = 0$) Parallelism Profile for SIMPLE (3 iterations, 20×20).

[From Arvind, Maa, Culler, "Parallelism in Dataflow Programs", Dec 1987]

The Secret: Use a Parallel Language!

```

for i←1:m, j←1:n do
  a[i,j] := b[i] c[j]
end

```

```

a[i,j] := b[i] c[j],   i←1:m, j←1:n

```

```

{ x3 | x ← fibs, x < 1000 }
[ (x,y) ↦ x(y+1)/2 | x ← 1#100, y ← 0#50 ]

```

```

Σ[k←1:n] a[k] xk

```

```

U[k←1:n] S[k]

```

```

BIG MIN[k←1:n] a[k]

```

$$\sum_{k=1}^n a_k x^k$$

Execution Environment

- Don't have the machine to yourself
 - > Must share with other programs
 - The OS at the very least!
 - > Load will vary over time
 - > If it's running it's important
- One program = Many components
 - > Every one is parallel
 - > Layers upon layers of parallelism

Slackness [Valiant CACM 1990]

- Expose extra parallelism
- Make task creation cheap
- Balance the load

- Cover latency with computation
- Adapt to changes in load

- Balancing temporal, spatial locality

Problem: Isolation [Burton Smith]

- Sequencing isolates in time
- Parallelism isolates in space
 - > Prevent updates that conflict
- Easiest: Eschew Updates
- Partition the world
- Detect need for isolation
 - > Strongly atomic nested transactions
 - > Isolated \neq Sequential!
 - > Transactional *data*, not *memory*

Parallel = Higher-Order

- Capture computational patterns
 - > map/reduce, stencil, ...
 - > skeletons community
 - > Framework? Library? Who cares!
- “Insert computation here” flavor
- That computation may be parallel
- It might even use the abstracted patterns

Generator Example

```

generate[R](r:Reduction[R], body:Z64→R): R =
  if size ≤ block then
    a : R = r.zero()
    i : Z64 = lo
    while i ≤ hi do
      a := r.join(a,body(i))
      i += 1
    end
    a
  else
    mid = ⌊(lo + hi) / 2⌋
    r.join(
      BlockedRange(lo,mid,b).generate(r,body),
      BlockedRange(mid+1,hi,b).generate(r,body))
  end

```

Hoist with my own petard

- What's the right amount of parallelism to expose in a framework?
- That rather depends upon:
 - > What computation we plug in
 - > The context in which we run
- Wish: no need to set block sizes or slice up recursions into “parallel at the top” and “sequential at the bottom”

Programmers: A Plea

- Patronizing attitude to Joe Coder
 - > Can't write a better FFT or malloc
 - > Can't write great parallel code
- But Joe Coder writes most code
 - > Will write parallel code
 - > If we make it easy
 - > And it solves the problem
- Right model for imperfect code?

projectfortress.sun.com

Wiki with Fortress documentation, Trac/svn repository

Put the feature in
a library.

Don't build it into
the compiler.

* Whenever possible

A language for building other languages