The High Cost of Communication

Hardware Implementation
vs.
Software Development

Guy E. Blelloch
Carnegie Mellon University

What is left to decide?

Two important unknowns about future "massively" parallel machines.

1. Single vs. multiple address space?
2. How good will interprocessor communication be?

Similar issues for the two questions:

1. Strong impact on programming.
   Simplifies software at cost of hardware.
2. Still in the research phase.
   Universities and Funding Agencies can have an important impact.

The cost of communication

Why does bandwidth effect the cost of software development?

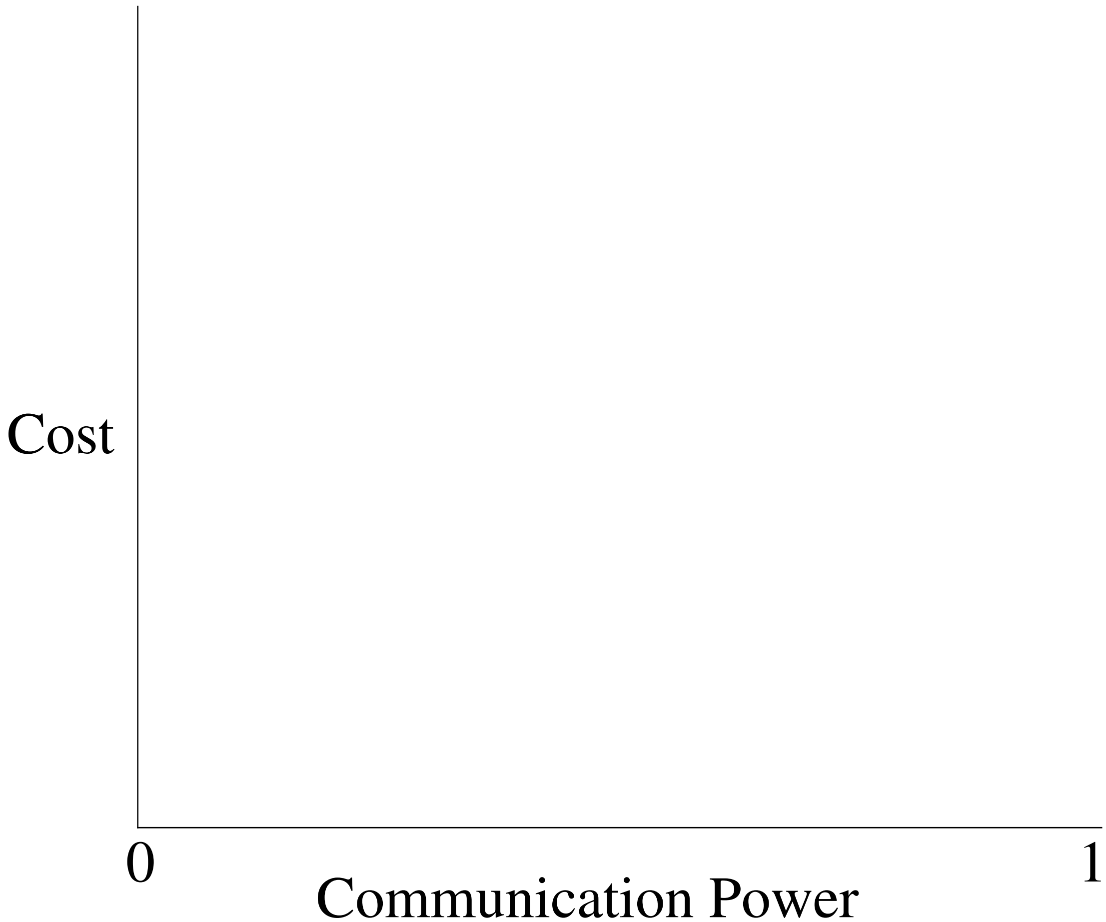
Particularly for applications on irregular data

1. Locality complicates algorithms, languages and compilers.
2. Different algorithms for different data.
3. Complicates abstraction boundaries.

Claim: Harder to deal with locality than parallelism. (In a general way)

Can we build "scalable" machines with high bandwidth and low overhead communication?
(e.g. < 5 instructions per 1 word message)

Cost Trade-off

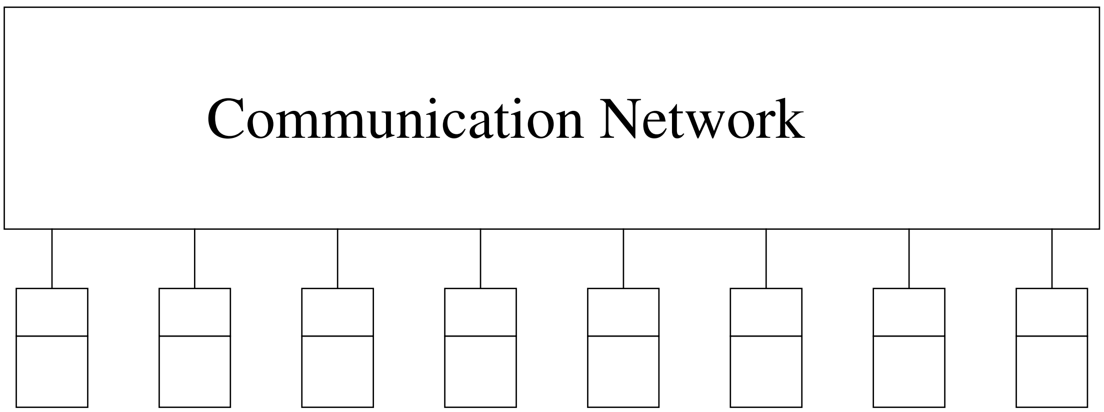


Myth: Scientists are willing to spend unlimited time getting the best performance.

Costs are a mixture of up-front and per-item expenses (true for both hardware and software).

Communication Parameters

The LOGP model



Overhead (in instructions)

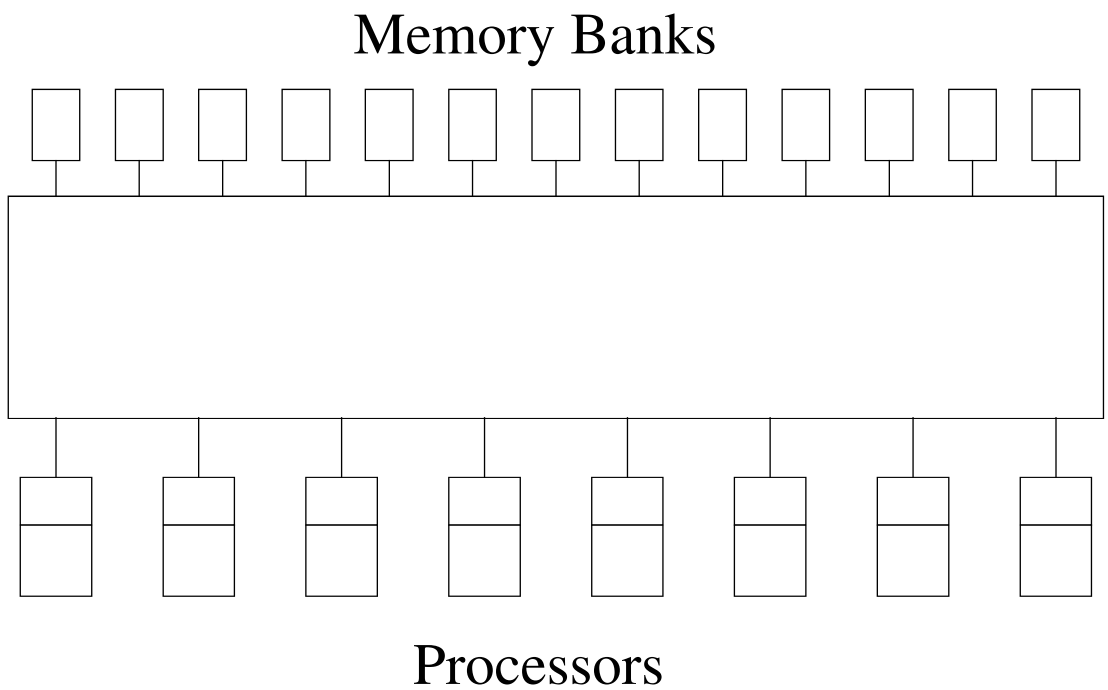
Latency (in instruction)

Gap (in instructions/word)

Bisection bandwidth (words)

Flops

$$\mathrm{Gap} \approx \frac{\mathrm{Flops} \times P}{\mathrm{Bisection}}$$

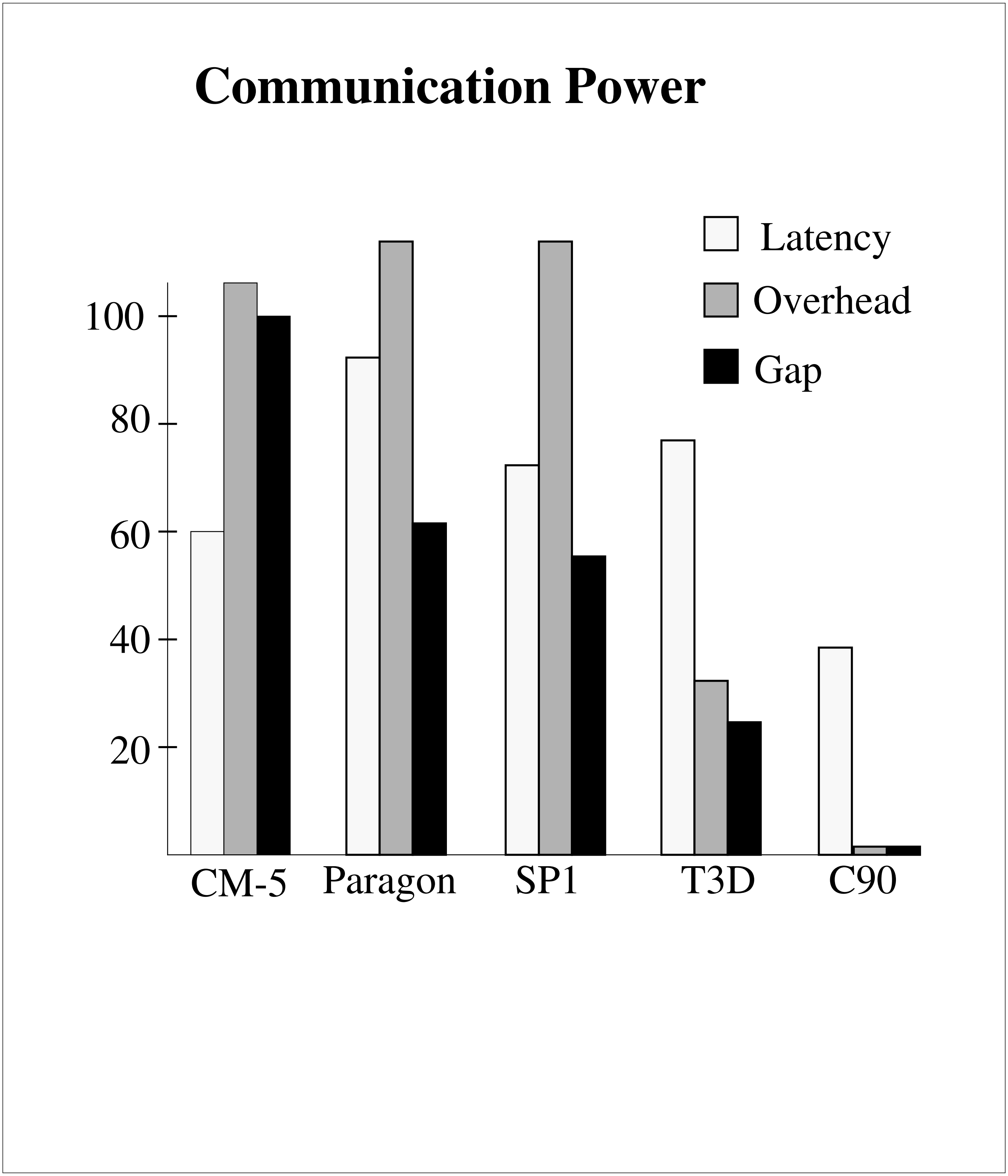

Memory Banks
Processors

Hiding Latency

Latency is not as important as the overhead and gap for "scalable" applications. Just requires more parallelism than processors.

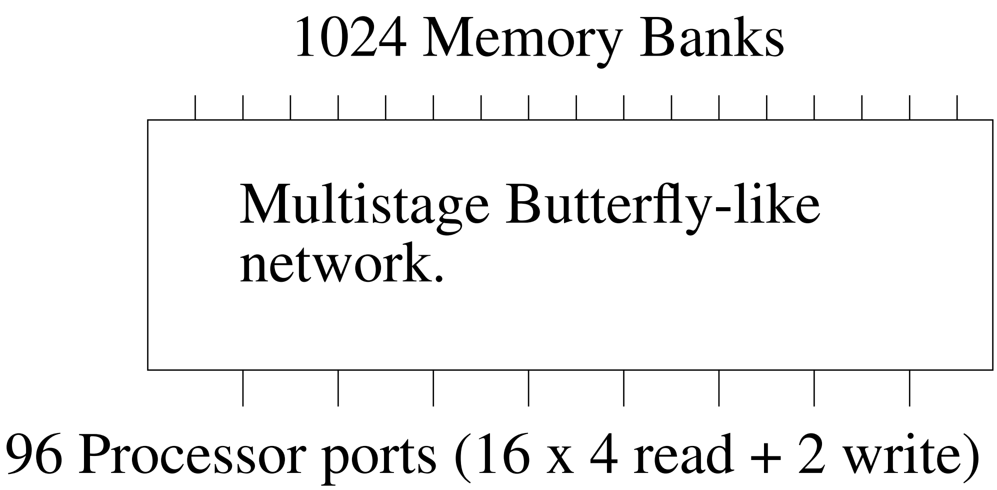
1. vectorizing
2. virtual processors
3. context switching
4. prefetching
5. split-phase assignment
6. bulk synchronous processing

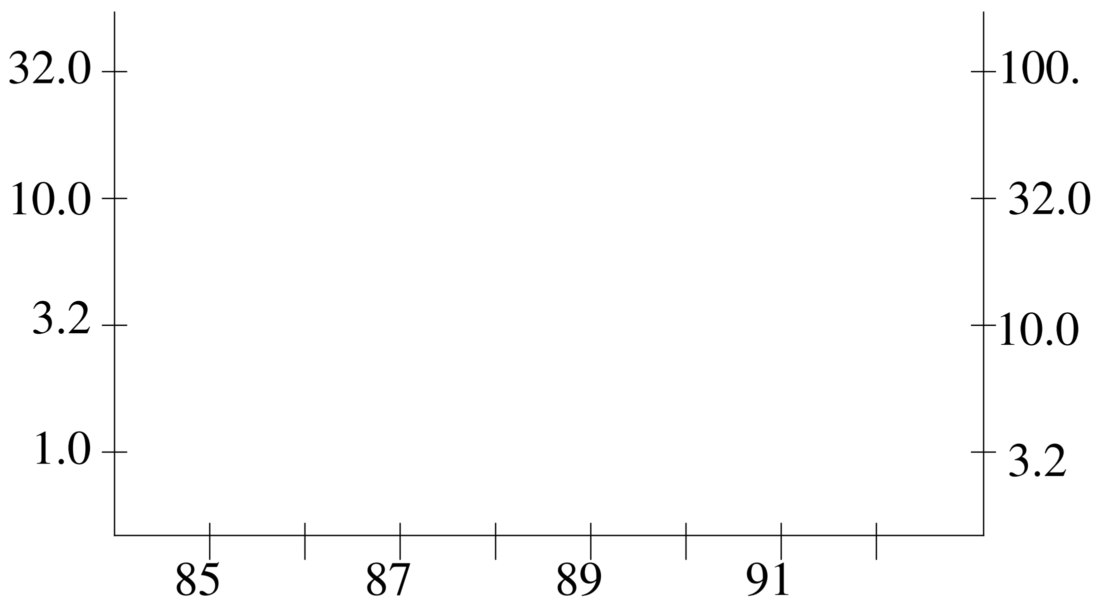
Same techniques can be used to hide memory latency on a serial processor .



Communication Power
Latency
Overhead
Gap
100
80
60
40
20
CM-5
Paragon
SP1
T3D
C90

Side-bar: The C90 Network





Programming for Locality

Claim: even for applications on “irregular” data one can usually get reasonable performance by carefully crafting the applications (most applications have some locality).

Problems:

- Often requires messy or complicated code.
- Often requires special purpose code.
  For a given problem, one will need different algorithms for different classes of inputs.
- Makes a mess of pointer-based algorithms and languages.

Conjugate Gradient

(one of the NAS benchmarks)

Inner loop is a sparse-matrix vector multiply (96% of the work). Rest of code is almost all local.

Representation of sparse matrix (sparse compressed column:

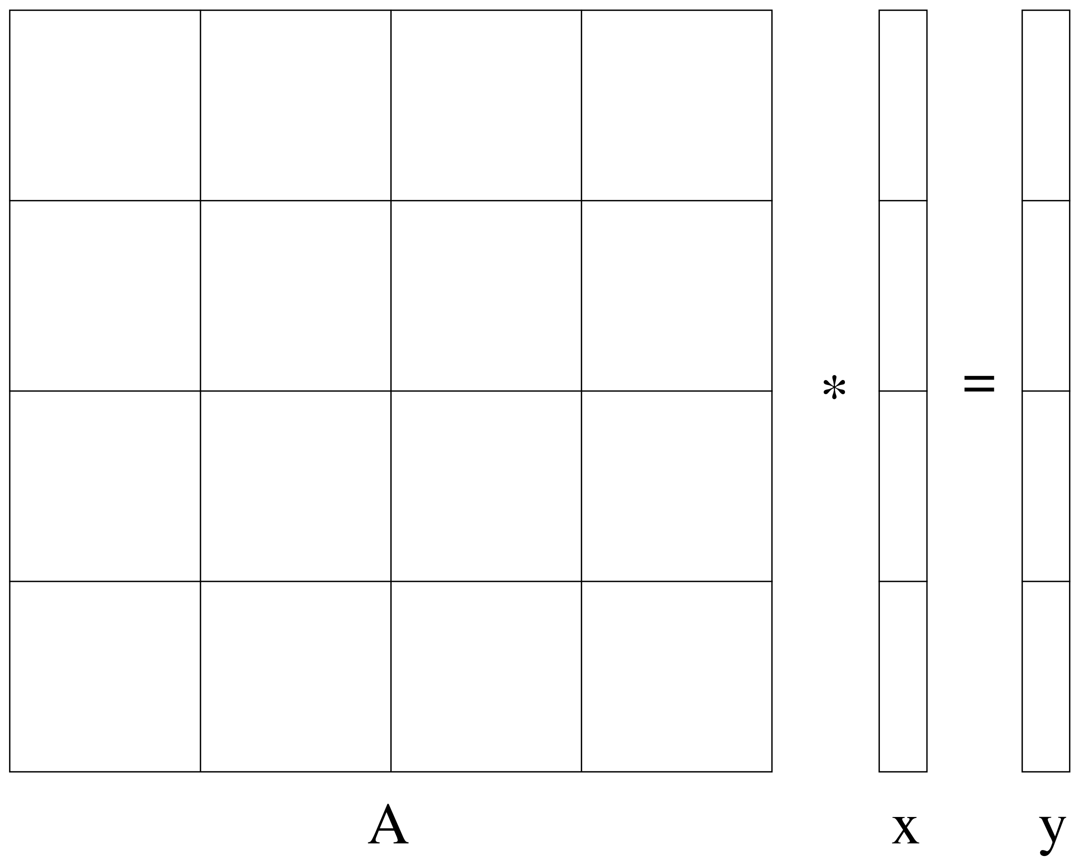
```
colstr = [1, 5, 8, 10, 13]
rowidx = [1,2,4,5,1,3,5,2,4,2,4,5]
a      = [5,7,9,2,......]
```

The code:

```
do 200 j = 1, n
  xj = x(j)
  do 100 k = colstr(j), colstr(j+1)-1
    y(rowidx(k))=y(rowidx(k))+a(k)*xj
```

Speeding up the CG benchmark

(Lewis and van de Geijn: Supercomputing93)
(also Hendrickson, Leland and Pimpton)



Takes advantage of:

- well distributed non-zeros
- matrix is not too sparse (120 elts/column)

Summary of example

- All algorithms do the same number of operations and give the same results. Differences are only a question of data allocation and movement.
- If communication was not a problem, then the parallel code would be trivial and general to all matrices.

Using sparse-compressed rows:
Preprocessing:

```
flag = false
flag(rowstr) = true
flag = parity_prefix(flag)
```

Inner loop:

```
products = val * x(colidx)
sums = sum_suffix(products,flag)
y = sums(rowstr)
```

What should the gap be?

Claim: If the gap is around 5 then:

1. Simple, easy to code, general-purpose algorithms would usually run within a factor of 2 of locality-minded special-purpose code.
2. The amount of work to get that extra factor of 2 would be relatively small.

Neither of these are true in general if the gap is greater than 10.

Conclusions

Non enough attention has been paid to the high software-development cost for *communication-impaired* machines.

Suggestion: The government should coordinate an effort to:

1. Standardized components/interfaces for low overhead, high throughput networks (cost sharing)
2. Develop hooks into standard microprocessors (coordinate with industry).

Hidden Agenda

How to give a talk promoting the PRAM without mentioning it once.

Some Observations

- Software is becoming a much larger part of the total cost of computer systems.
- Few applications have been ported to parallel computers.
- Irregular computations are becoming more common even in scientific computing.

To get wide spread use of parallel machines it is critical to make them easy to program.

Claims

- High bandwidth great simplifies programming
- Bandwidth/processor -speed is a question of cost.

	Procs	Gflops	Gwords	Gap	Overhead
C90	16	16	24	< 1	1
CM-5	1024	128	1.2	100	100
Paragon	1024	75	1.2	60	1000
T3D	1024	150	6	25	10?
IBM SP1	512	50	1.0	50	1000

The NAS Benchmarks

- Used by almost all vendors
- Broad set of applications
- Simple enough to study

	Benchmark	Communication	Regular?
EP	Embarrassingly Parallel	low	yes
MG	Multigrid	medium	yes
CG	Conjugate Gradient	high	no
FT	3D FFT	medium	yes
IS	Integer Sort	high	no
LU	Lower-upper Diagonal	high	yes
SP	Scalar Pentadiagonal	medium	yes
BT	Block Tridiagonal	low	yes