TreadMarks

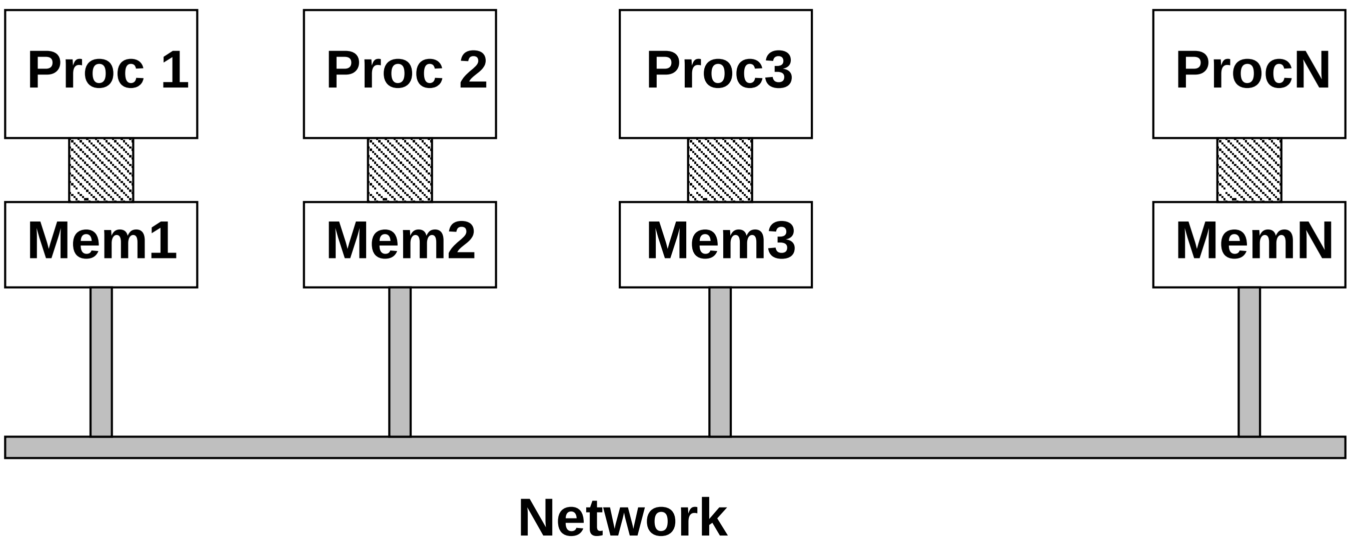
Shared Memory Computing on Networks of Workstations

Willy Zwaenepoel

ParallelTools LLC
and
Department of Computer Science
Rice University

Networks of Workstations

Parallel computing on networks of workstations



All commodity technology (including network), thus cheap

Performance?

Faster processors

Faster floating point

More memory

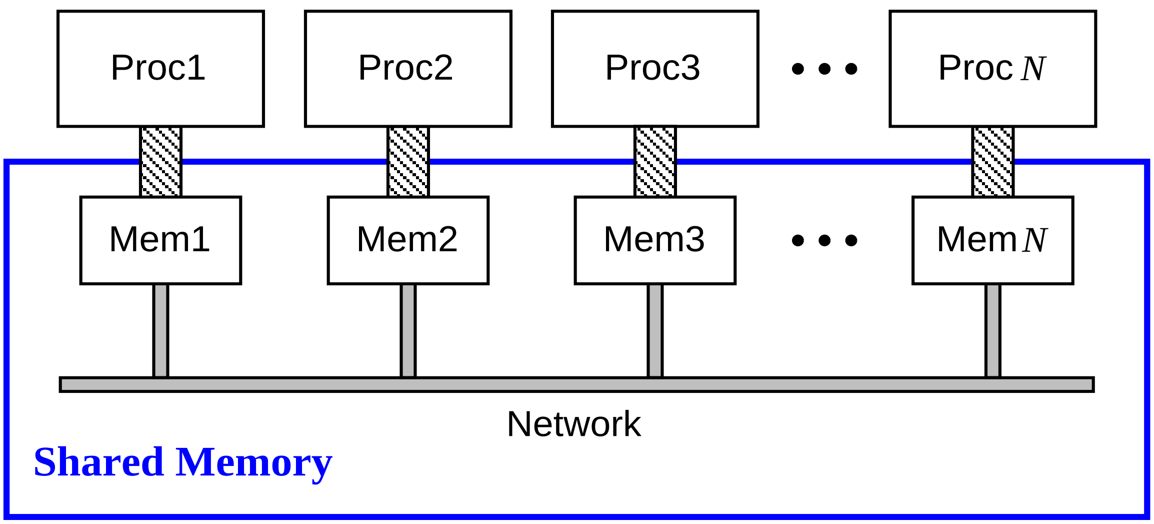
Faster networks (are coming ...)

Bottom line:

Good MIPS/FLOPS per $ ratio

Distributed Shared Memory (DSM)

Software provides shared memory image



Why Shared Memory?

Easier to go sequential → parallel

Sequential =
single thread + single address space

Shared memory =
multiple threads + single address space

Message passing =
multiple threads + multiple address spaces

Shared Memory API

Threads

Synchronization

- Locks
- Barriers
- Flags

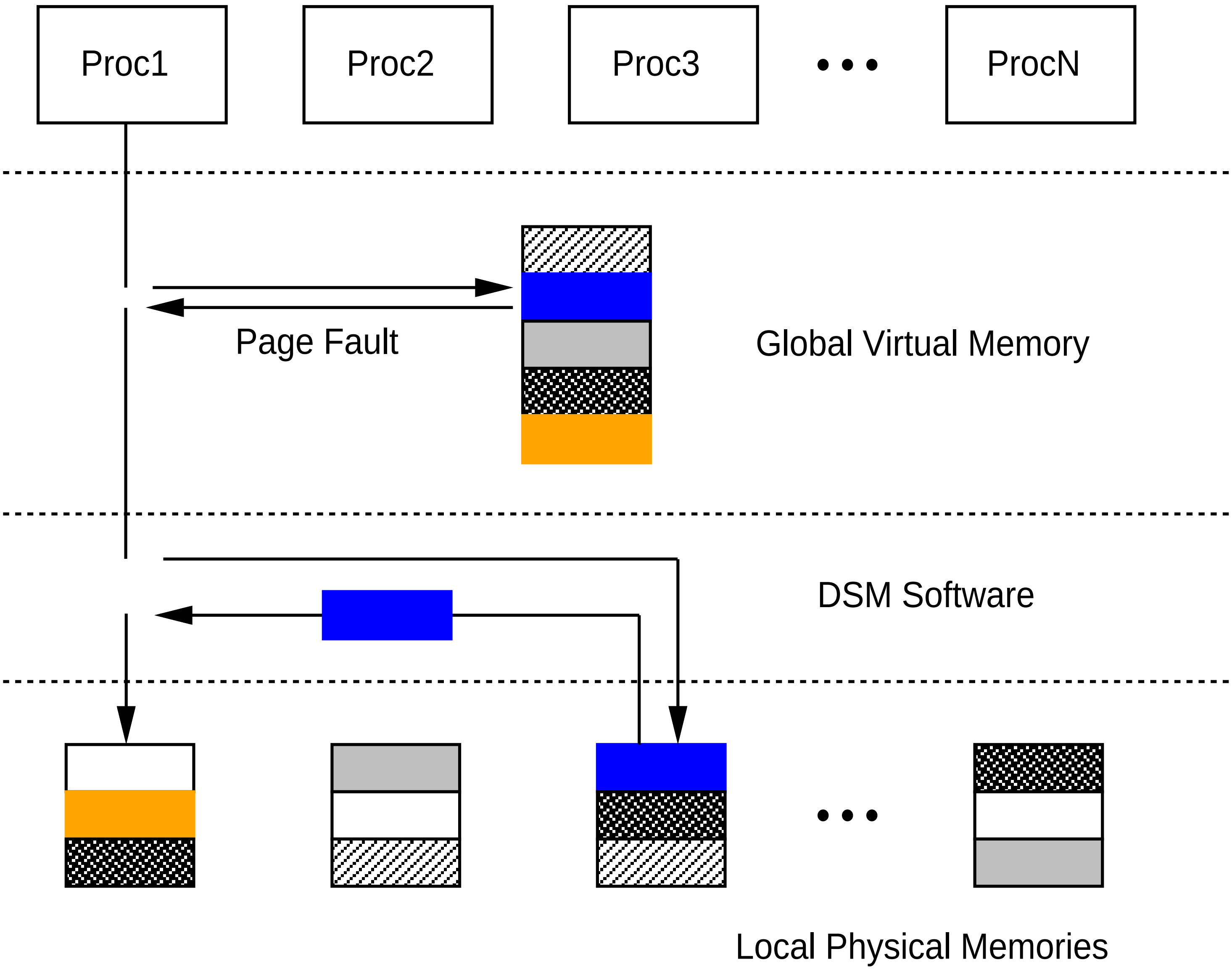
Shared memory allocation

Key Point

Distributed shared memory:

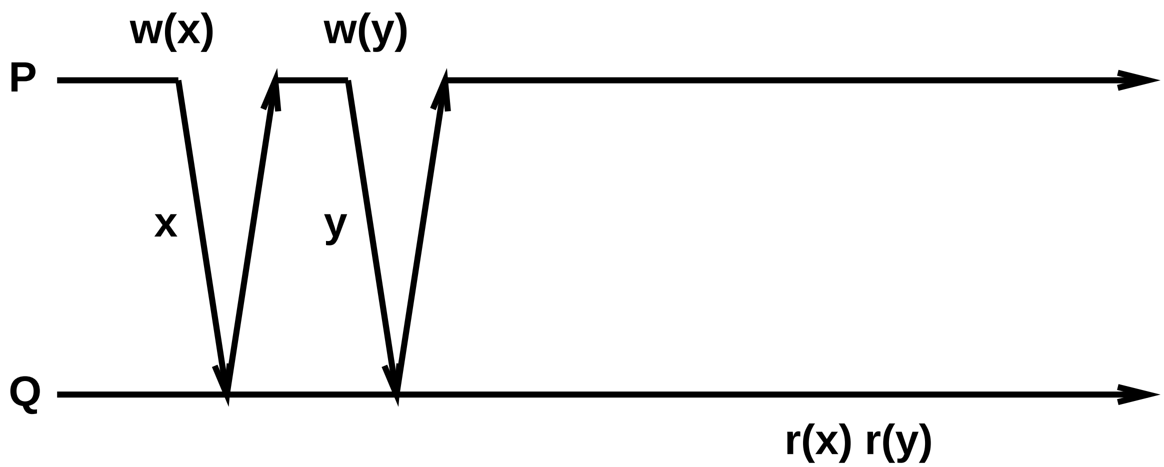
- support for parallel processing
- on networks of workstations
- for real problems
- with reasonable efficiency
- with reasonable programmer effort

Conventional DSM Implementation [Li 86]



Performance Problem: Sequential Consistency

Every write visible "immediately"

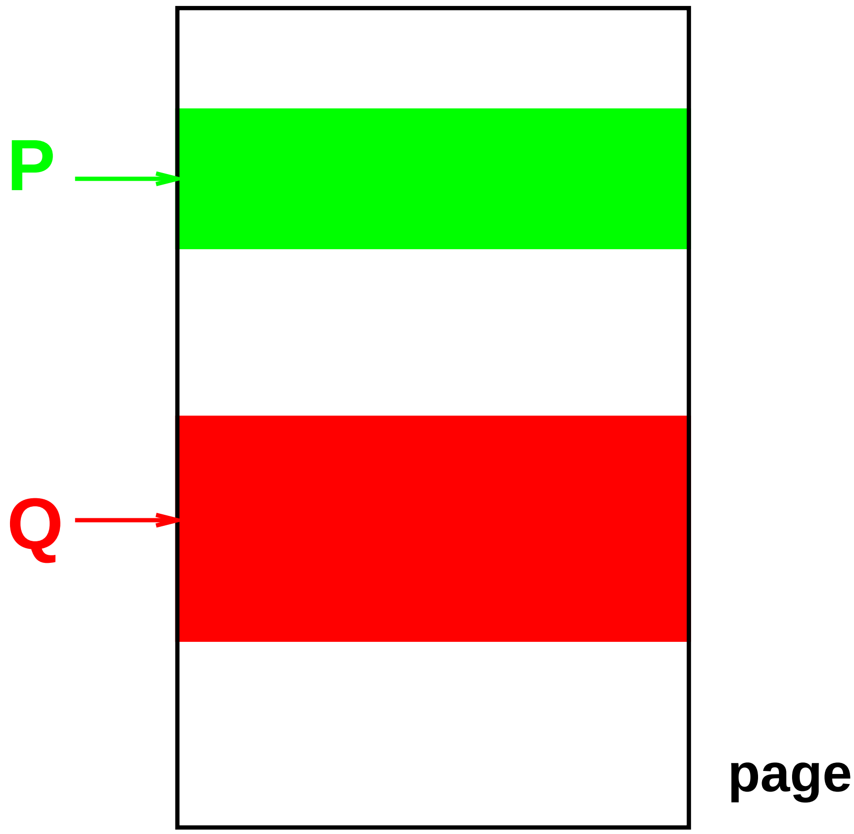


Problems:

- Number of messages
- Latency

Performance Problem: False Sharing

Pieces of the same page updated by different processors



Leads to "ping-pong" effect

Performance Problems: Solutions

Goal:

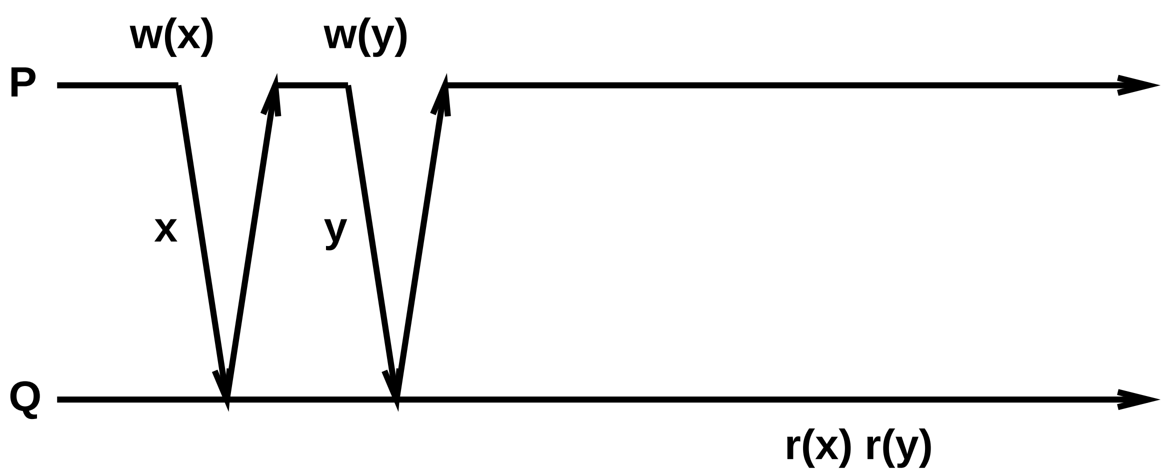
- Reduce communication
- Keep shared memory model

Techniques:

- Lazy release consistency [Keleher 92]
- Multiple writer protocol [Carter 91]

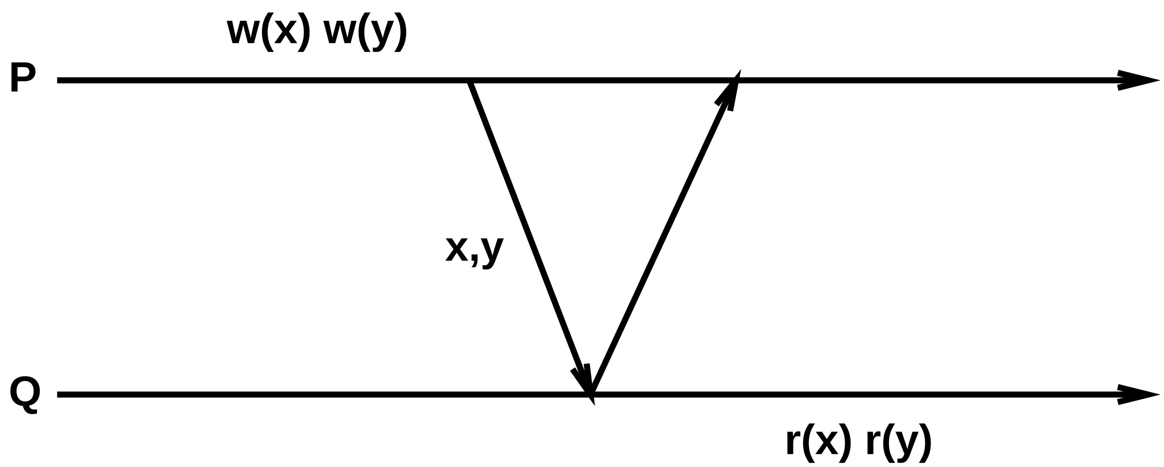
Sequential Consistency

Every write visible "immediately"



Relaxed Consistency Models

Delay making writes visible



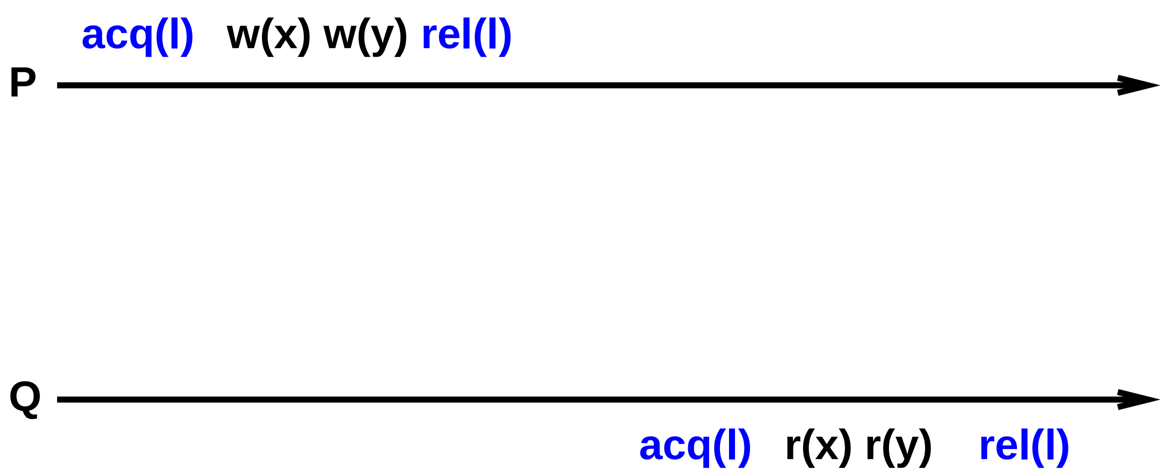
Goal:

- Reduce number of messages
- Hide latency

Delay until when?

There is more to this program ...

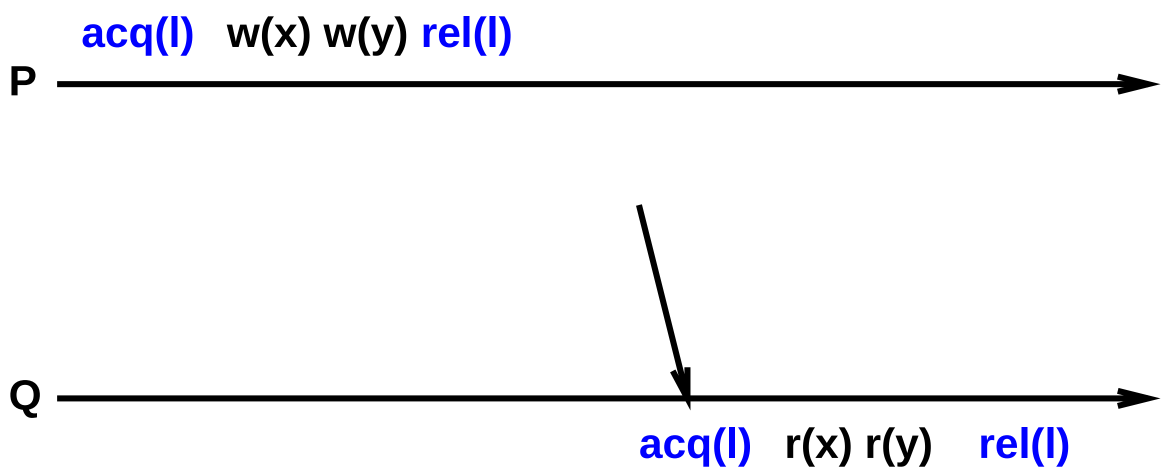
Program needs to be synchronized



Note: Synchronization is not added for RC, it was there already!

Release Consistency (RC)

Delay until Q synchronizes with P



If program is data-race-free, programmer won't notice!

RC Programming Model

Write data-race-free programs

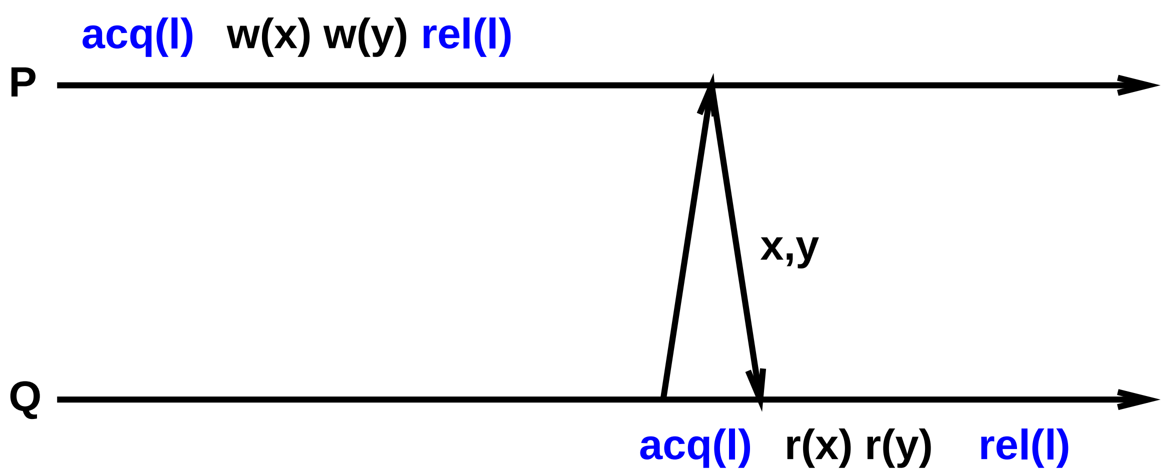
Synchronization through system primitives
(no spinlocks!)

Then, RC = SC, but with fewer messages

Lazy RC

Pull modifications at acquire

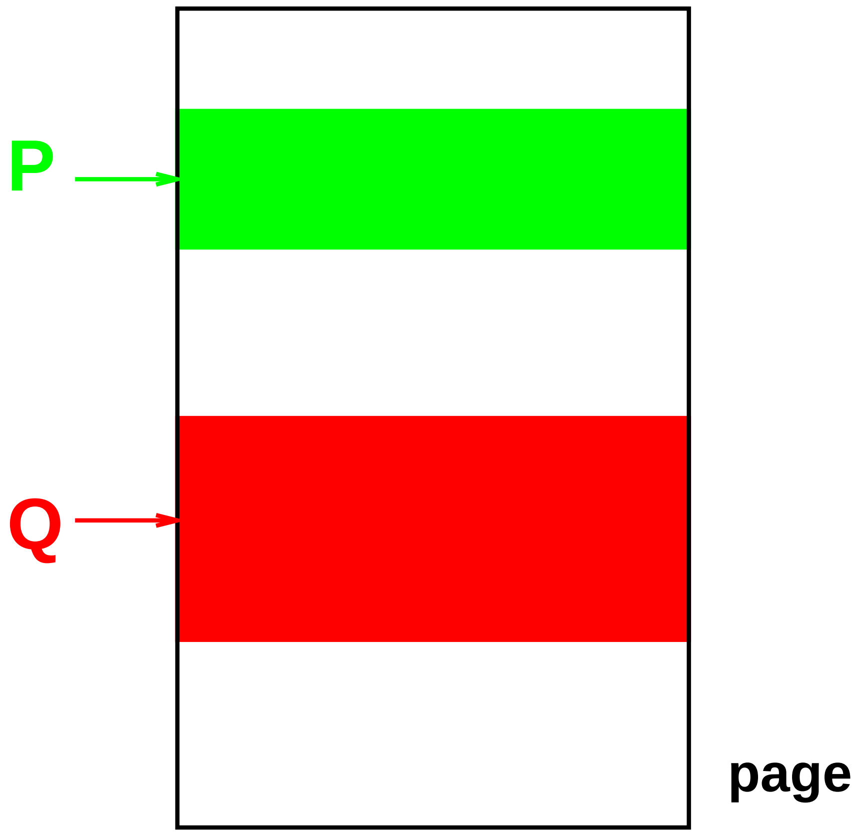
(rather than push them at release)



Fewer messages

False Sharing

Pieces of the same page updated by different processors



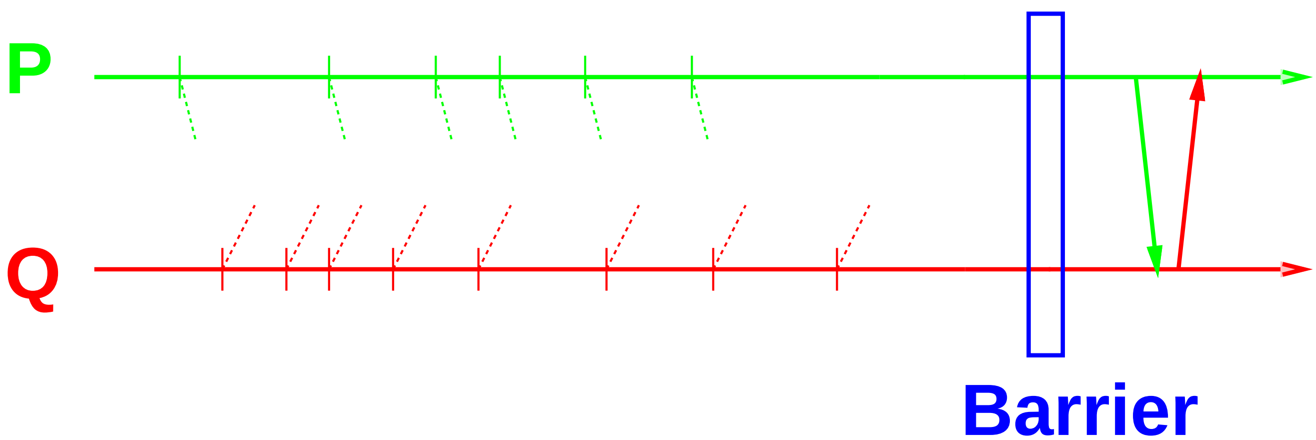
Multiple Writer Protocol

Addresses false sharing

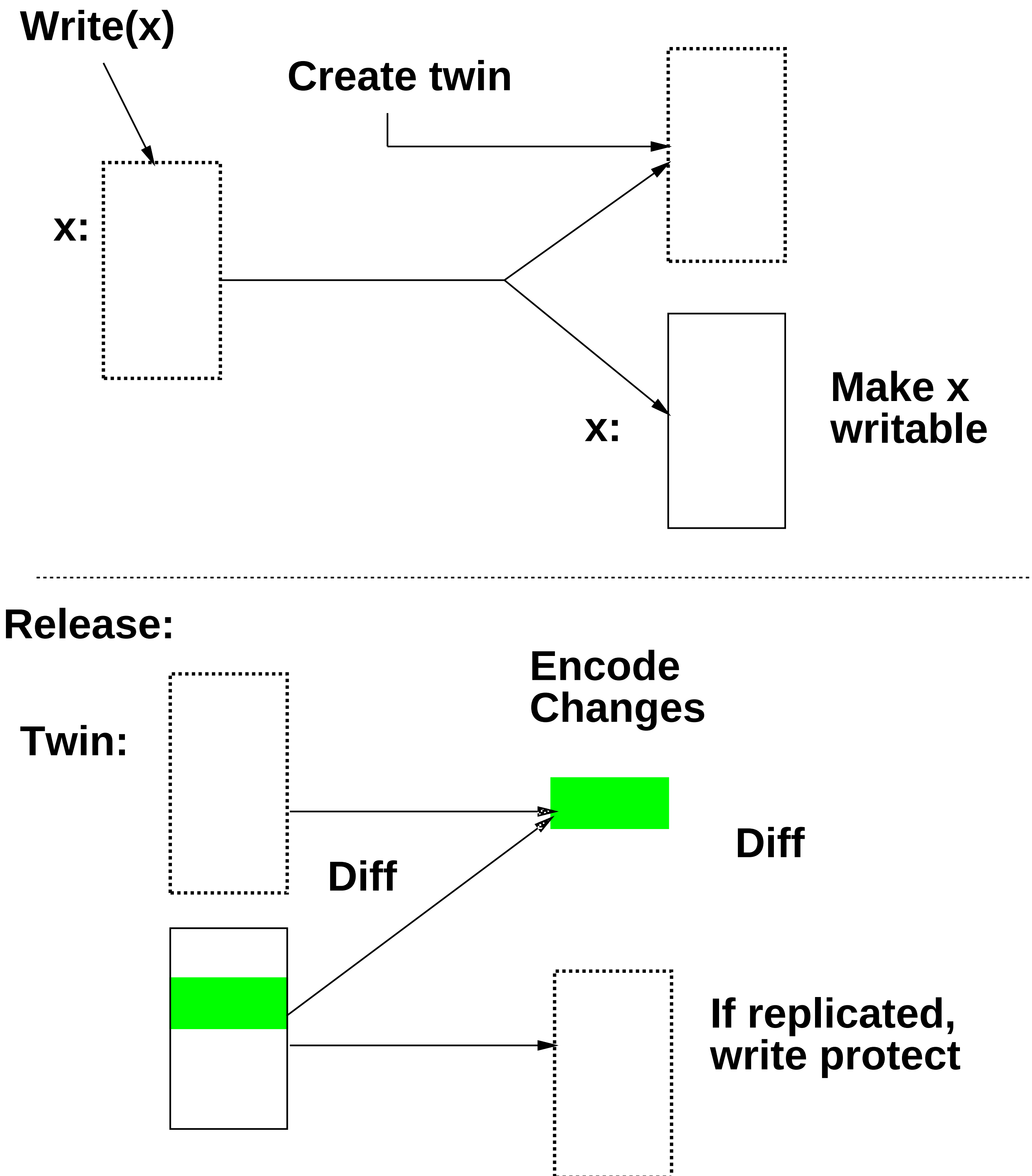
Buffer writes until synchronization

Create diffs

Synchronize → pull in modifications



Diff Creation



TreadMarks

Standard kernel and compilers

User-level library for C and Fortran

Implemented on

- DEC
- HP
- IBM
- Intel
- Sun
- SGI

Relatively portable

[Keleher et al. 94]

Two Applications

Mixed Integer Programming

Genetic Linkage

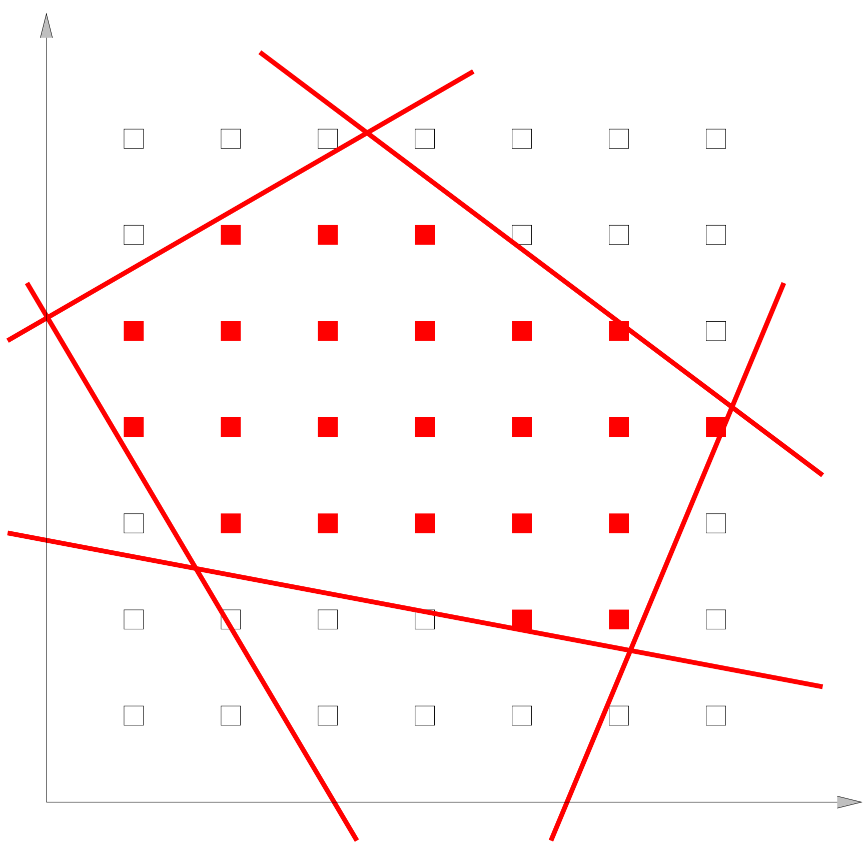
Mixed Integer Programming

Mixed Integer Programming =

Linear Programming +

Some of the variables are integers

A 2-dimensional example:



Mixed Integer Programming (continued)

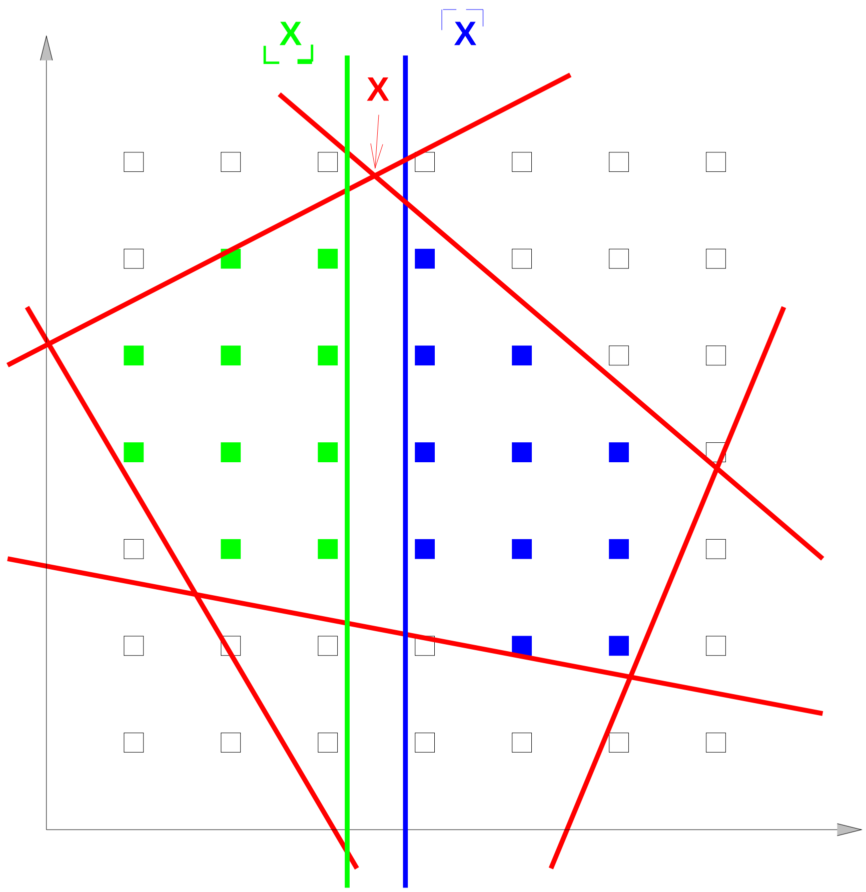
Used in many applications

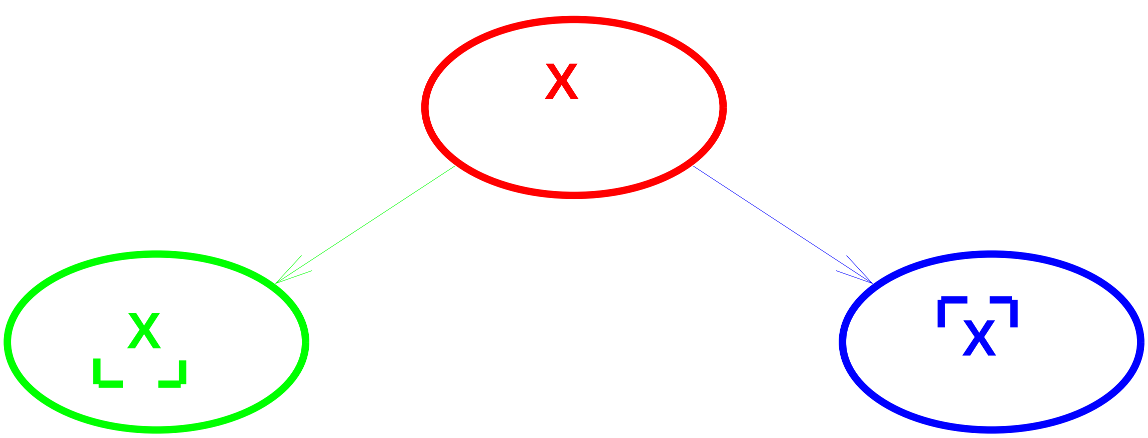
Hard in a theoretical sense

Hard in a practical sense:

real instances run for a long time

Branch-and-Bound





Algorithmic Smarts

Plunging

Pick the right variable

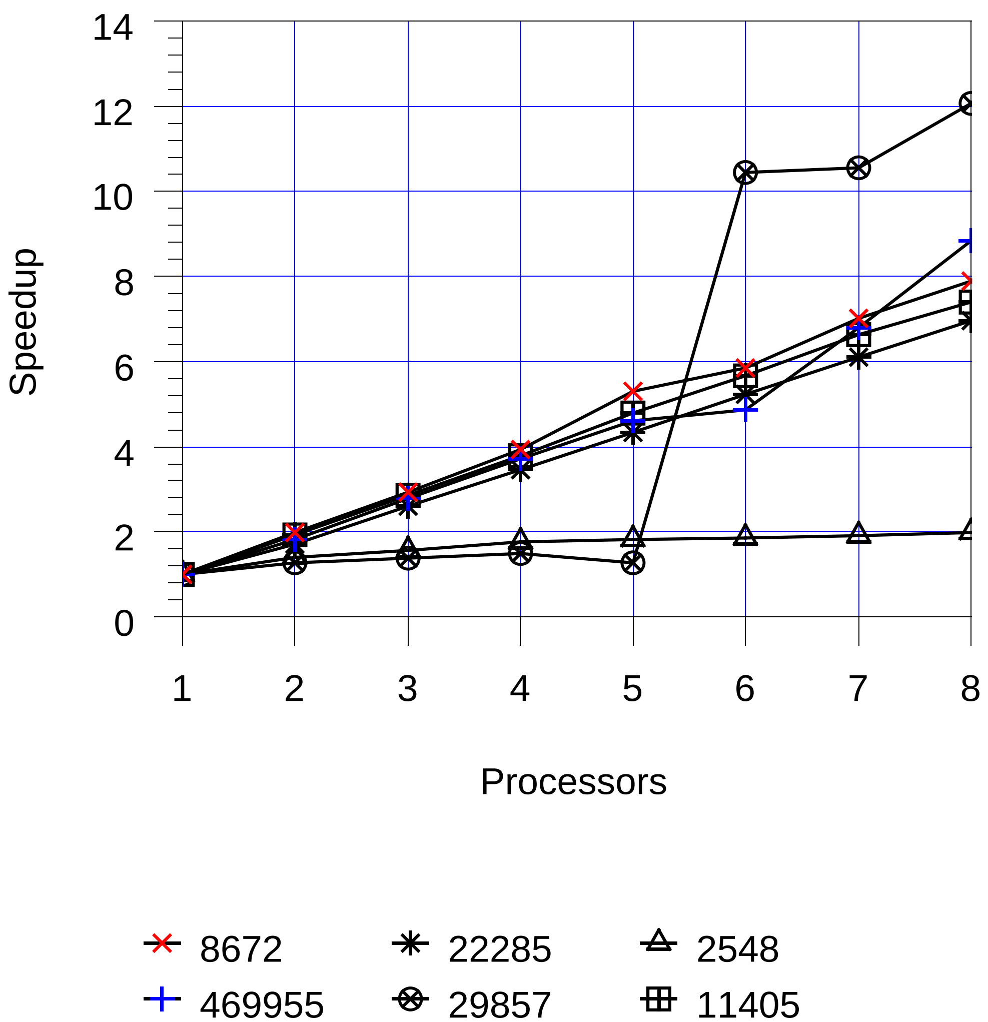
Pick the right node

Cutting planes (branch-and-cut)

10,000 lines of C code (excluding LP solver)

Results

MIPLIB problems longer than 2,000 seconds on 1 processor.



[Lee et al., 1995]

Neat Result

D. Bienstock and O. Gunluk, Lightwave network configuration (Bellcore), to appear in Mathematical Programming

521 variables, 56 0/1 variables

664 constraints

Previously unsolved

Solved on an 8-node IBM SP2 (3 1/2 days)

Genetic Linkage Analysis

Disease gene location:

- biological experiments
- computational steps (linkage analysis)

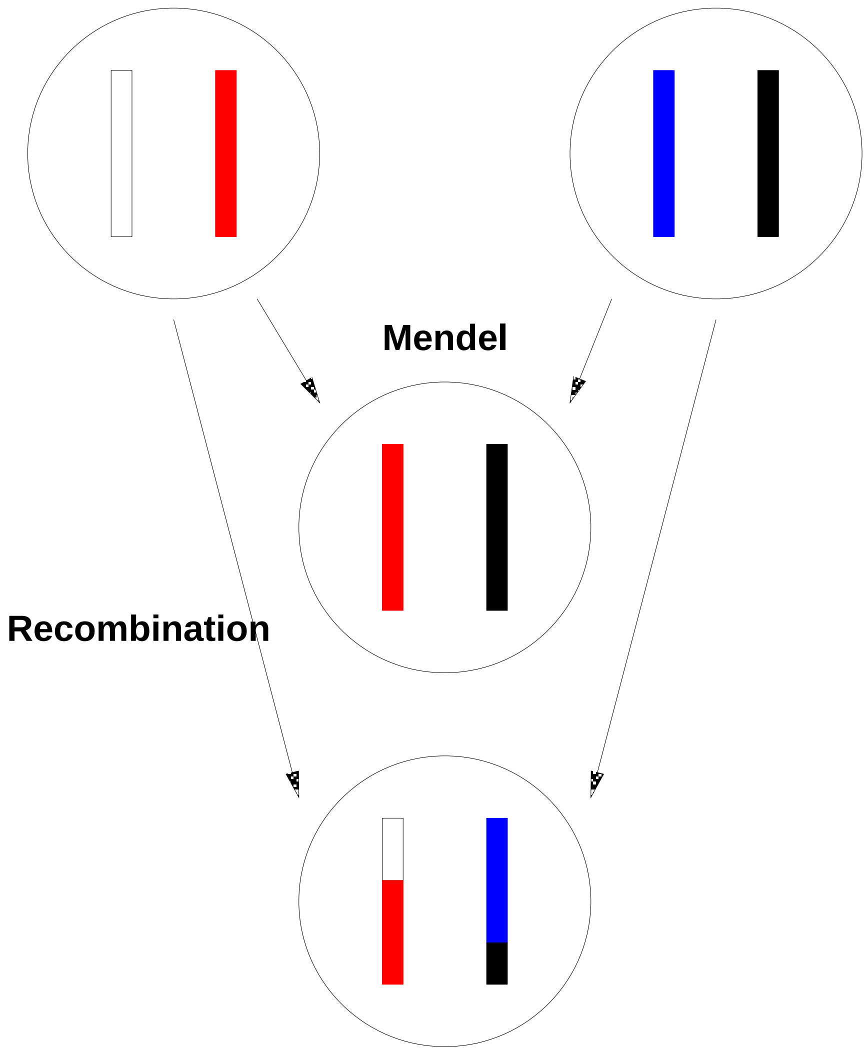
Computation is bottleneck

Hours to months is normal

Better accuracy desired

A 1-Minute Intro to Genetics

Probability of recombination θ



The Linkage Computation

Maximum likelihood optimization of θ

Linkage Parallelized

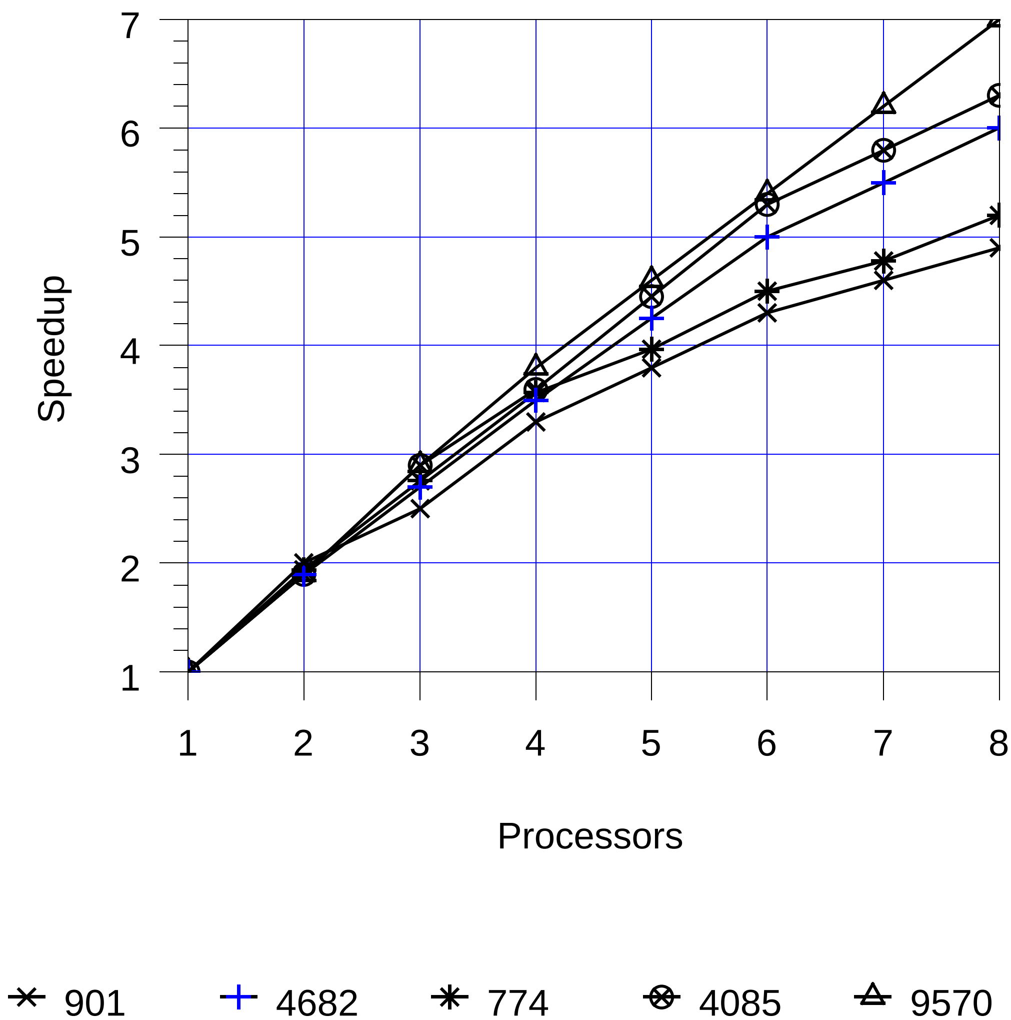
Optimize for θ

- For each nuclear family
  - Split up rows over processors
  - For each processor
    - Do updates for assigned rows
  - Synchronize

Load balancing in splitting

13,000 lines of C code

Results



[Gupta et al., 1995]

Parallel FASTLINK Sites

ANGIS, Sydney, Australia (SPARC SMP)

Columbia University, New York (Alpha SMP)

Fox Chase Cancer Center, Philadelphia (Alpha network)

Griffith University, Brisbane, Australia (IBM SP-2)

Human Genome Project, Hinxton, U.K. (SGI SMP)

Infobiogen, Paris, France (SPARC SMP)

MDC für Mol. Medizin, Berlin, Germany (SPARC SMP)

NIH (IBM SP-2 and SPARC network)

Ospedale San Raffaele, Milan, Italy (SPARC SMP)

Sequana Therapeutics, La Jolla (SPARC network)

University of Antwerp, Belgium (Alpha SMP)

Conclusion

Real problems can be solved

on networks of workstations

using distributed shared memory

with reasonable efficiency

with reasonable programmer effort

Further Work

Better support tools

- Compiler support
- Performance visualization

Multiprocessor support