# Trends in Parallel Computing for Process Engineering

Chao-Yang Gau and Mark A. Stadtherr*

Department of Chemical Engineering
University of Notre Dame
Notre Dame, IN 46556 USA

AspenWorld 2000
Orlando, FL
February 6-11, 2000

*Phone: (219)631-9318; Fax: (219)631-8366; E-mail: markst@nd.edu

# Outline

- Trends in technology
- One application: Parallel branch-and-bound for chemical process modeling and optimization problems

# Trends – High End Computing

- Massively parallel
- Commodity processors (e.g., RISC, Pentium)
- Distributed memory
- Interconnect by very high speed switch
- Current (May 1999) example: IBM Blue Pacific
  - 5760 processors
  - 3.9 teraflops peak performance ($3.9 \times 10^{12}$ floating point operations per second)
- Soon: IBM ASCI White (ASCI = Accelerated Strategic Computing Initiative, a DOE program)

# ASCI White

- 10 teraflops peak performance; cost $100 million
- 8192 processors (375-MHz IBM POWER3-II; capable of four simultaneous floating point operations)
- 8 MB (at least) RAM cache per processor
- Processors organized into nodes of 16 processors each (512 nodes)
- Each node has at least 8 GB local memory plus two internal 18 GB disks
- System has 10,752 external disks (195 terabytes $\approx$ 20 LOCs)
- Multistar omega switching network (diameter two)
- Ultimate goal in ASCI series: 100 teraflops (0.1 petaflop) by 2004

# Trends – Commercial Mainstream

- Symmetric multiprocessor (SMP) systems (or some variation)
- Shared memory (may not be uniform access)
- Small (2) to moderate (64) number of commodity processors: desktop to enterprise scale
- SMP technology has become dominant in the commercial server market and is widely used in industrial and academic research
- Network-based systems: clusters of SMPs, clusters of workstations (COWs), metacomputing

# Metacomputing

- Heterogeneous network of computational resources, ranging from PCs to SMPs or beyond.

- The current fastest computer system is actually not IBM Blue Pacific, but is a metacomputing system:

- SETI@home
  - Over one million processors (home PCs)
  - Radio telescope data from Arecibo is processed, looking for signs of ETs.
  - Computation rate (Sept. 99) over 7 teraflops

- Virtual supercomputing
  - The network is the computer
  - Analogy: plug into a metacomputer for computing power as you would plug into the electrical power grid
  - Cycle scavenging

# Towards the Future ...

- Metacomputing (RefrigeratorNet??)
- Quantum computing
- Optical computing
- Hybrid technologies (e.g., quantum processors with optical interconnects)

# Parallel Branch-and-Bound Techniques

- Branch-and-Bound (BB) and branch-and-prune (BP) have important applications in engineering and science, especially when a $global$ solution to an optimization or equation solving problem is sought
  - process synthesis
  - analysis of phase behavior
  - molecular modeling
  - etc.

- BB and BP involve successive subdivision of the problem domain to create subproblems, thus requiring a tree search process
  - Applications are often computationally intense
  - Subproblems (tree nodes) are independent
  - A natural opportunity for use of parallel computing

- There are various BB and BP algorithms; we use an interval Newton/generalized bisection (IN/GB) method (session DT9, Wed. morning)

# Parallel BB (cont'd)

- For practical problems, the binary tree that needs to be searched may be quite large.

- The binary trees may be highly irregular, and can result in highly uneven distribution of work among processors and thus poor overall performance (e.g., idle processors).

- Need an effective load scheduling and load balancing scheme to do parallel tree search efficiently.

- Manager-worker schemes are popular but scale poorly due to communication bottleneck.

- Three types of algorithms, designed for network-based parallel computing (with MPI for message passing) were studied:
  - Synchronous Work Stealing (SWS)
  - Synchronous Diffusive Load Balancing (SDLB)
  - Asynchronous Diffusive Load Balancing (ADLB)

# Work Scheduling and Load Balancing

- Objective: Schedule the workload among processors to minimize communication delays and execution time, and maximize computing resource utilization.

- Use Dynamic Scheduling
  - Redistribute workload concurrently at runtime.
  - Transfer workload from a heavily loaded processor to a lightly loaded one (load balancing).

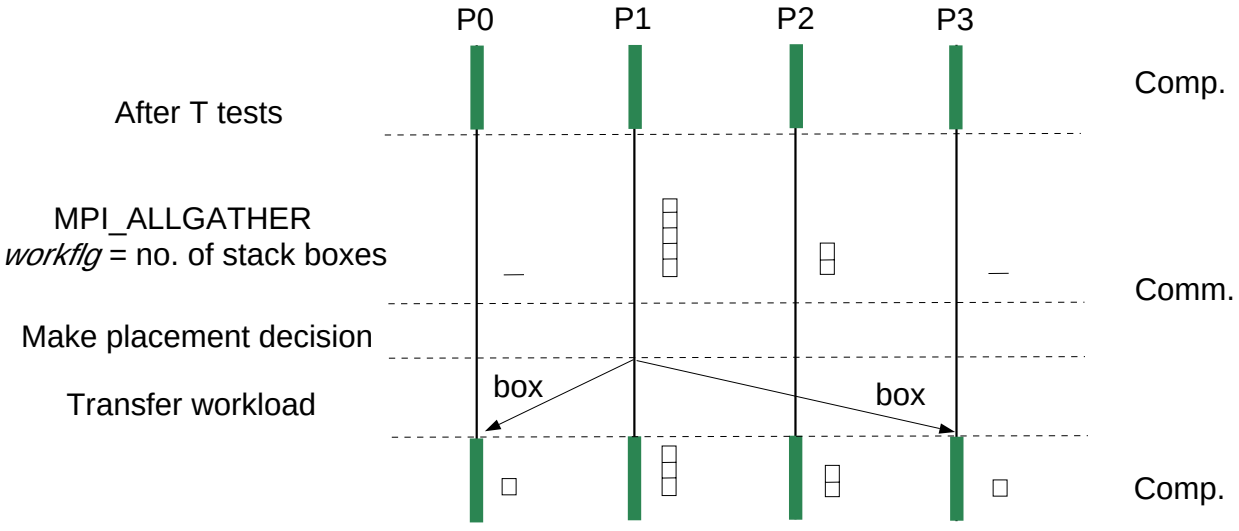
- Use Distributed Load Balancing
  - Each processor locally makes the workload placement decision to maintain the local interval stack and prevent itself from becoming idle.
  - Alleviates bottleneck effects from centralized load balancing policy (manager/worker).
  - Reduction of communication overhead could provide high scalability for the parallel computation.

# Synchronous Work Stealing

- Periodically update workload information, $workflg$, and any improved upper bound value (for optimization) using synchronous global (all-to-all) blocking communication.

- Once idle, steal one interval (box) from the processor with the heaviest work load.

- Difficulties
  - Large network overhead (global, all-to-all)
  - Idle time from process synchronism and blocking communication



# Synchronous Diffusive Load Balancing

- Use $local$ communication: Processors periodically exchange units of work with their immediate neighbors to maintain their workload.

- Reduces the appearance of idle states.

- Typical workload adjusting scheme:

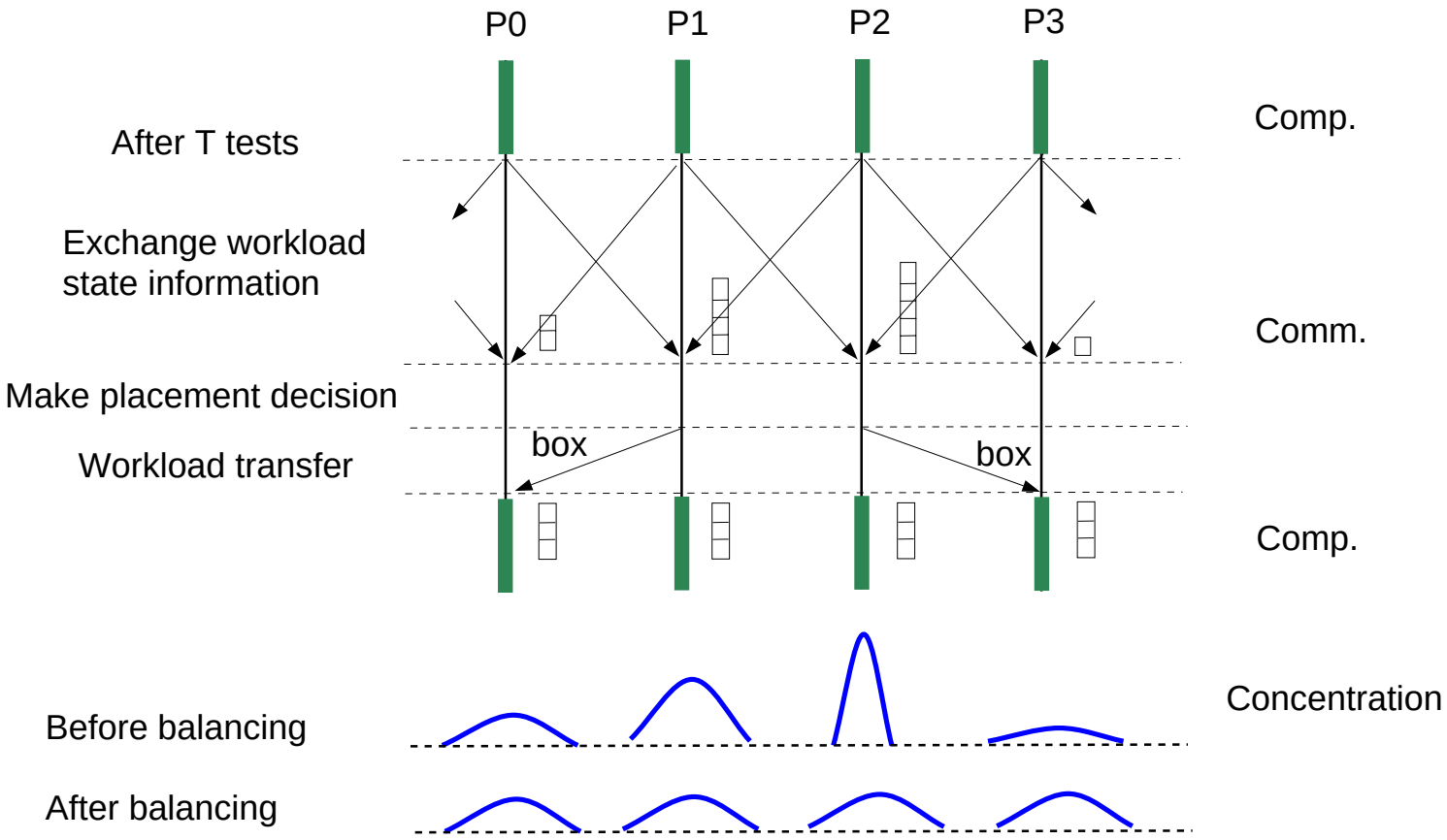
$$u(j) = 0.5(workflg(i) - workflg(j))$$

($i$: local processor: $j$: neighbor processor)

If $u(j)$ is positive and greater than some tolerance: send intervals (boxes). If $u(j)$ is negative and less than some tolerance: receive intervals (boxes).

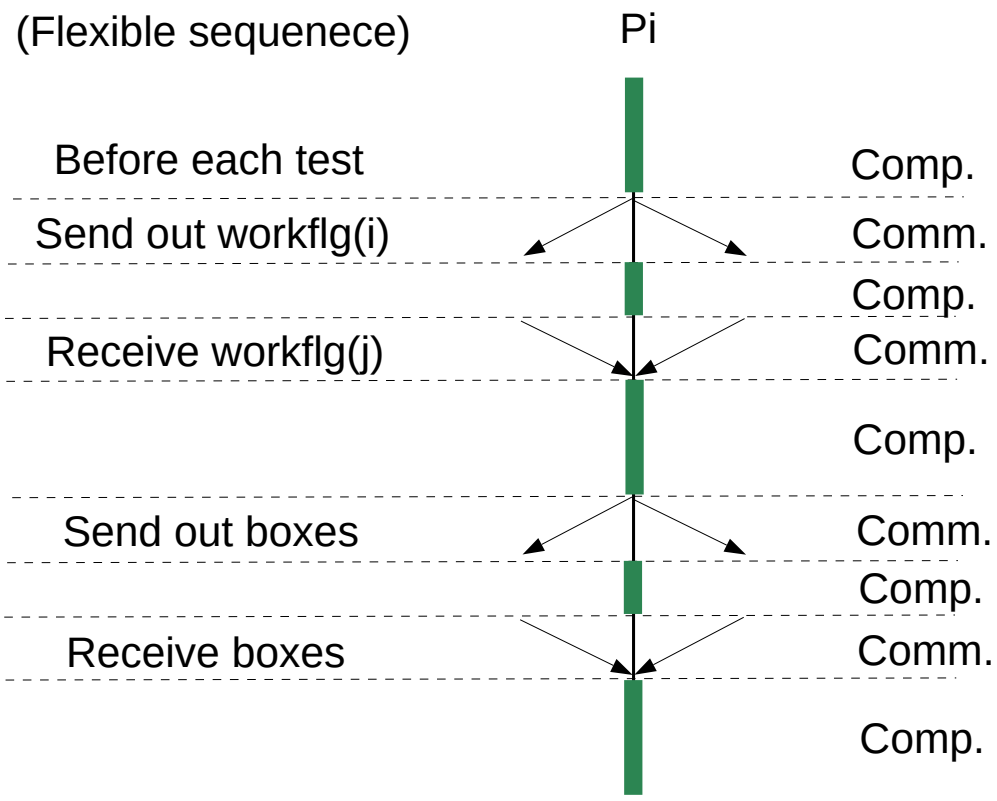
- Difficulties
  - Inefficiency due to synchronism.
  - Termination effects arising from local communication strategy and diffusive message propagation.

# Synchronous Diffusive Load Balancing (cont'd)



# Asynchronous Diffusive Load Balancing

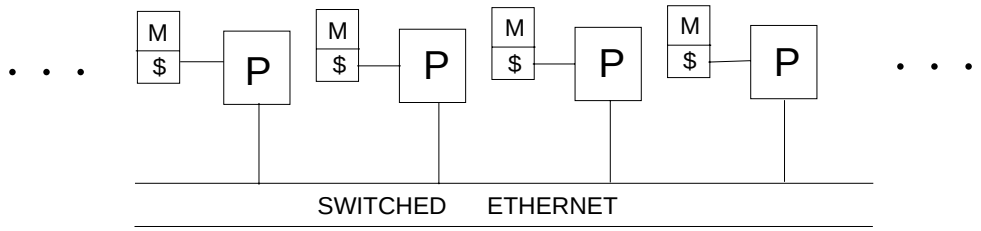
- Use asynchronous nonblocking communication, MPI_ISEND, to update workload information and transfer workload, breaking process synchronization.
- Overlap communication and computation.
- Maintain the workload (number of stack boxes) higher than some threshold.



# Testing Environment

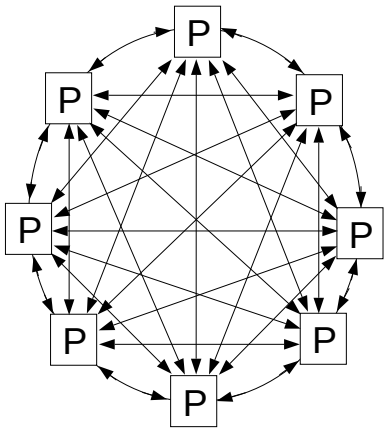
- Software: Implemented in Fortran 77 using the portable message-passing interface (MPI) protocol

- Physical hardware: Sun Ultra workstations connected by switched Ethernet



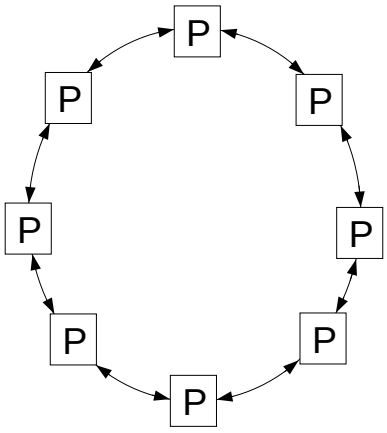

- Virtual Network:

Global Communication
All-to-All Network




Used for SWS

Local Communication
1-D Torus Network



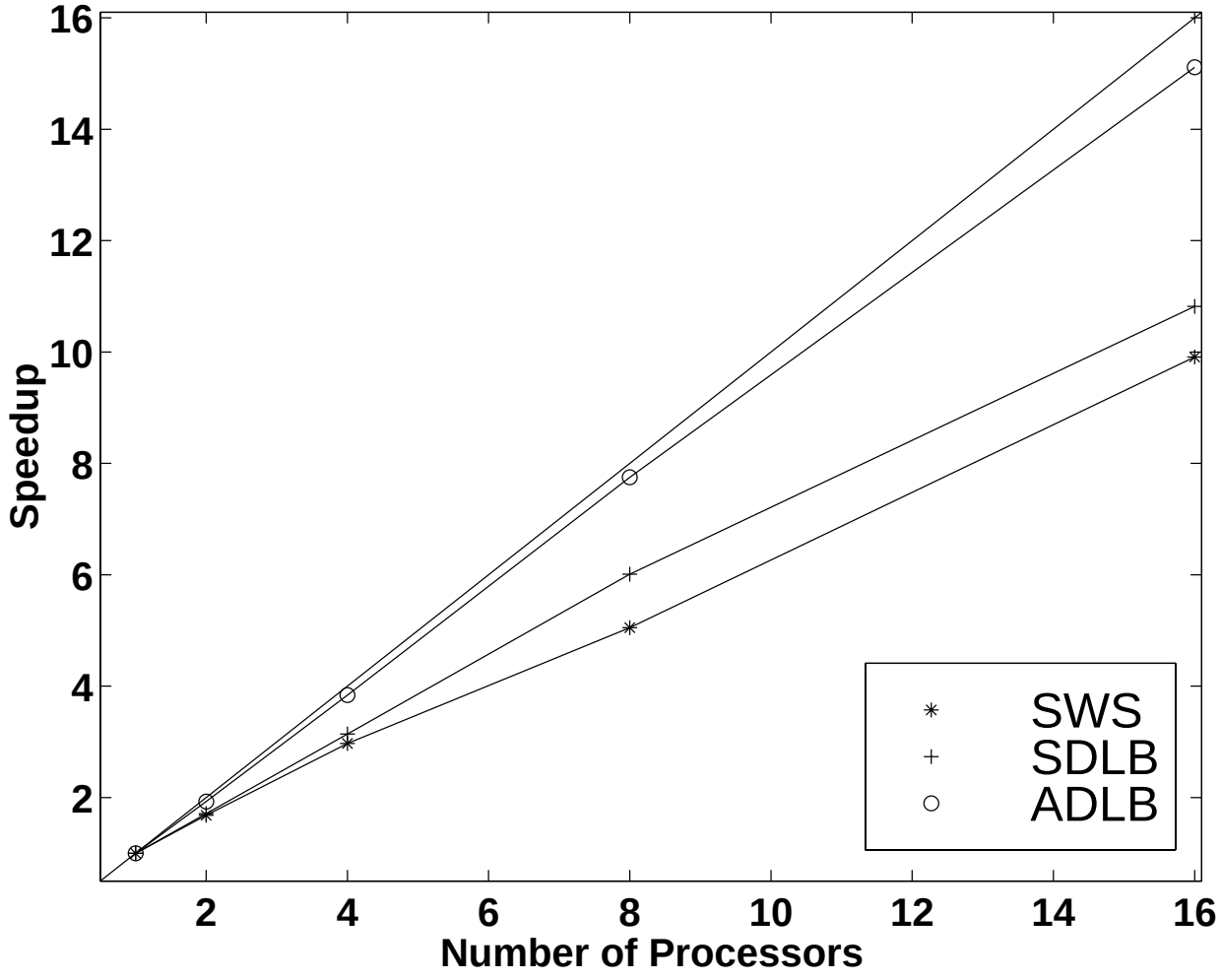

Used for SDLB and ADLB

# Test Problem

- Parameter estimation in a vapor-liquid equilibrium model.
- Use the maximum likelihood estimator as the objective function to determine model parameters that give the "best" fit.
- Problem data and characteristics chosen to make this a particularly difficult problem.
- Can be formulated as a nonlinear equation solving problem (which has five solutions).
- Or can be formulated as a global optimization problem.

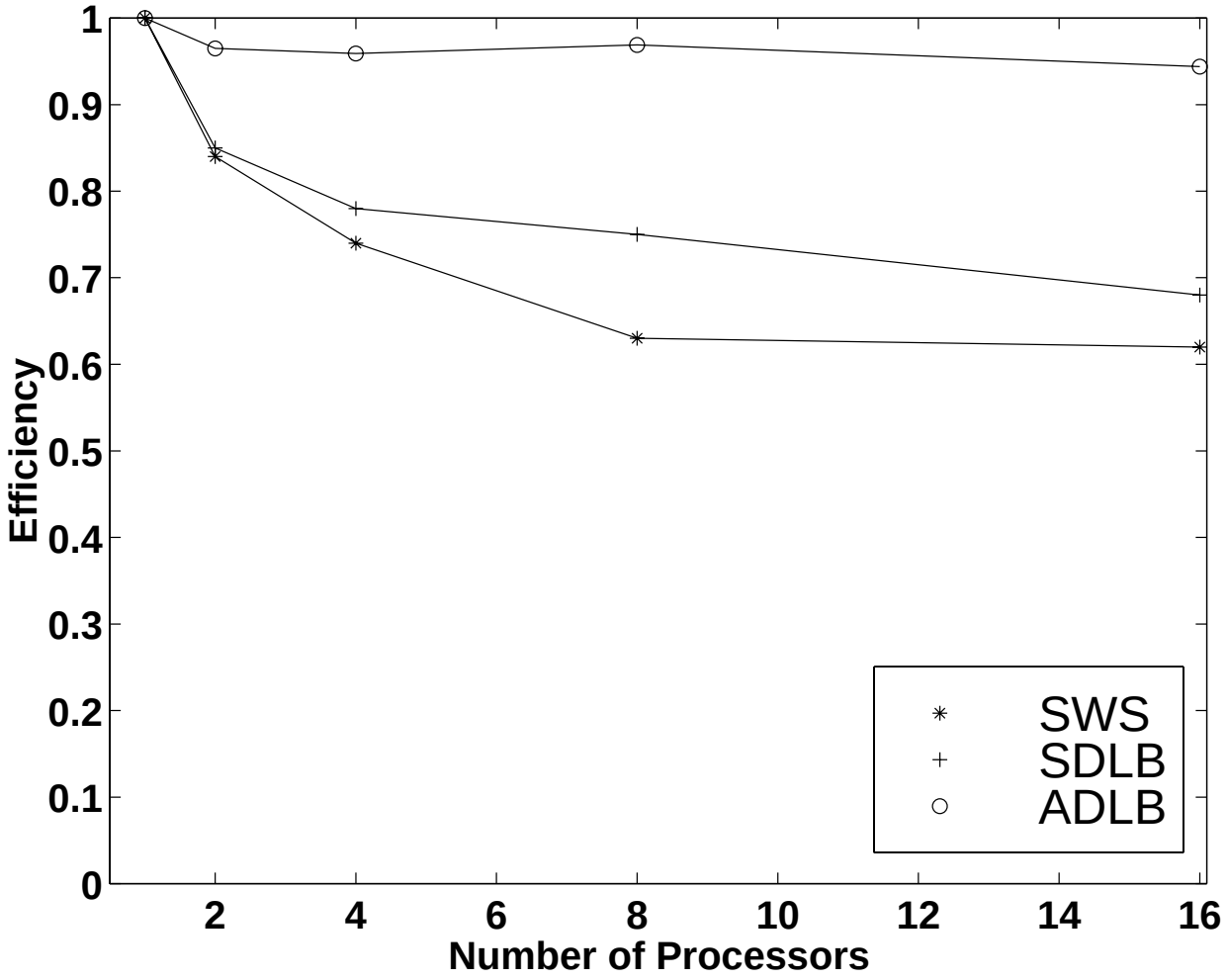
# Comparison of Algorithms on Equation-Solving Problem

Speedup vs. Number of Processors



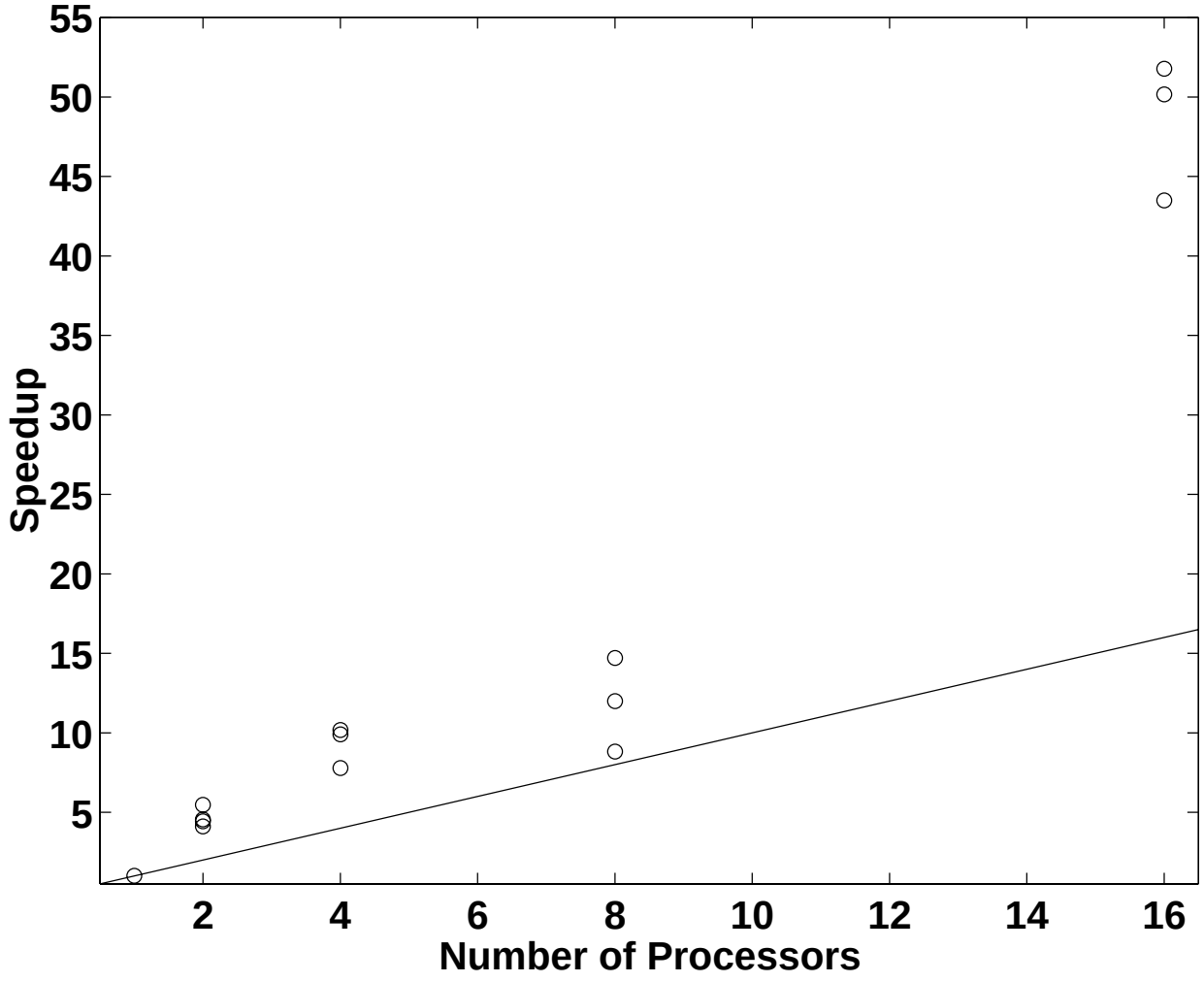
# Comparison of Algorithms on Equation-Solving Problem

Efficiency vs. Number of Processors



# Using ADLB on Optimization Problem

Speedup vs. Number of Processors (three different runs of same problem)



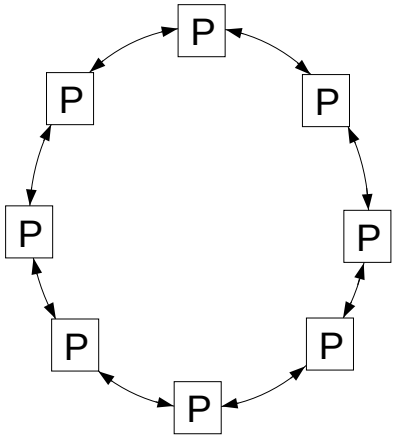
# Using ADLB on Optimization Problem (cont'd)

- Speedups around 50 on 16 processors: superlinear speedup
- Superlinear speedup is possible because of broadcast of least upper bounds, causing intervals to do discarded earlier than in the serial case. That is, there is less work to do in the parallel case than in the serial case.
- Speedup anomaly: Results vary from run to run because of different timing in finding and broadcasting improved upper bound.

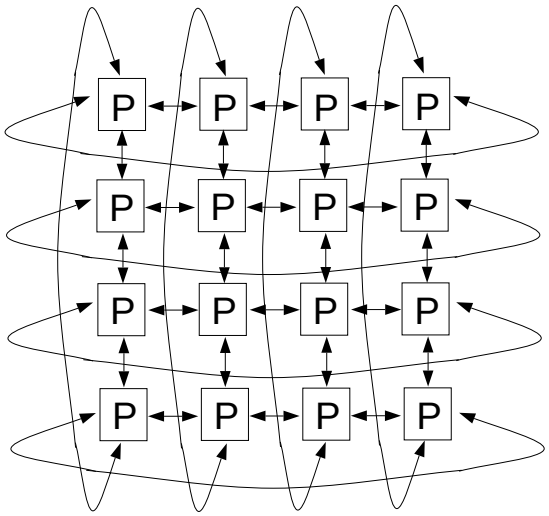
# Discussion

We have also considered performance in a 2-D torus virtual network.

1-D Torus Network




2-D Torus Network



# Discussion (cont'd)

- Comparison of 1-D vs. 2-D torus
  - 2-D has higher communication overhead (more neighbors)
  - 2-D has smaller network diameter (shorter message diffusion distance): $\lceil \sqrt{P}/2 \rceil$ vs. $\lfloor P/2 \rfloor$
  - Trade off may favor 2-D for large number of processors.

- Isoefficiency analysis with up to 32 processors demonstrated the better scalability of the 2-D torus on parallel BB and BP problems (Gau and Stadtherr, 1999)

- A dual stack management strategy can be used to reduce speedup anomaly and produce more consistently high superlinear speedups on optimization (BB) problems (Gau and Stadtherr, 1999).

# Concluding Remarks

- Effective load management strategies can be developed for solving BB and BP problems in parallel on a network-based system.

- Parallel computing technology can be used not only to solve problems faster, but to **solve problems more reliably**.
  - Find the **global** optimum in an optimization problem.
  - Find **all** the solutions of a system of nonlinear equations.

- These reliability issues are often overlooked: Are we just getting the wrong answers faster?

# Acknowledgments


- NSF DMI96-96110
- NSF EEC97-00537-CRCD
- U.S. Army Research Office DAAG55-98-1-0091


- For more information:
  - Contact Mark Stadtherr at markst@nd.edu