Parallel Interval Analysis for Chemical Process Modeling

Chao-Yang Gau and Mark A. Stadtherr*

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

SIAM CSE 2000
Washington, D.C.
September 21–24, 2000

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

Outline

- Motivation: Reliability in Computing
- Methodology: Interval Newton/Generalized Bisection
- Parallel Implementation on a Cluster of Workstations
- Some Performance Results

High Performance Computing

In chemical engineering and other areas of engineering and science, high performance computing is providing the capability to:

- Solve problems faster
- Solve larger problems
- Solve more complex problems

⇒ **Solve problems more reliably**

Motivation

- In process modeling and other applications, chemical engineers frequently need to solve nonlinear equation systems in which the variables are constrained physically within upper and lower bounds; that is, to solve:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$$

- These problems may:
  - Have multiple solutions
  - Have no solution
  - Be difficult to converge to any solution

Motivation (cont'd)

- There is also frequent interest in globally minimizing a nonlinear function subject to nonlinear equality and/or inequality constraints; that is, to solve (globally):

$$\min_{\mathbf{x}} \phi(\mathbf{x})$$

subject to

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$$

$$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$$

- These problems may:
  - Have multiple local minima (in some cases, it may be desirable to find them *all*)
  - Have no solution (infeasible NLP)
  - Be difficult to converge to any local minima

Interval Newton/Generalized Bisection

- Given initial bounds on each variable, IN/GB can:
  - Find (enclose) any and all solutions to a nonlinear equation system to a desired tolerance
  - Determine that there is no solution of a nonlinear equation system
  - Find the global optimum of a nonlinear objective function

- This methodology:
  - Provides a **mathematical guarantee** of reliability
  - Deals automatically with rounding error, and so also provide a **computational guarantee** of reliability
  - Represents a particular type of branch-and-prune algorithm (or branch-and-bound for optimization)

IN/GB (cont'd)

- For solving a nonlinear equation system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ the interval Newton method provides pruning conditions; IN/GB is a branch-and-prune scheme on a binary tree

- No strong assumptions about the function $\mathbf{f}(\mathbf{x})$ need be made

- The problem $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ must have a finite number of real roots in the given initial interval

- The method is not suitable if $\mathbf{f}(\mathbf{x})$ is a "black-box" function

- If there is a solution at a singular point, then existence and uniqueness cannot be confirmed—the eventual result of the IN/GB approach will be a very narrow enclosure that *may* contain one or more solutions

IN/GB (cont'd)

- Can be extended to global optimization problems
- For unconstrained problems, solve for stationary points
- For constrained problems, solve for KKT points (or more generally for Fritz-John points)
- Add an additional pruning condition:
  - Compute interval extension (bounds on range) of objective function
  - If its lower bound is greater than a known upper bound on the global minimum, prune this subinterval since it cannot contain the global minimum
- This is a branch-and-bound scheme on a binary tree

Some Types of Problems Solved

- Fluid phase stability and equilibrium (e.g. Hua et al., 1998)
- Location of azeotropes (Maier *et al.*, 1998, 1999, 2000)
- Location of mixture critical points (Stradi *et al.*, 2000)
- Solid-fluid equilibrium (Xu *et al.*, 2000)
- Parameter estimation (Gau and Stadtherr, 1999, 2000)
- Phase behavior in porous materials (Maier and Stadtherr, 2000)
- General process modeling problems—up to 163 equations (Schnepper and Stadtherr, 1996)

Parallel Branch-and-Bound Techniques

- 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
- For practical problems, the binary tree that needs to be searched in parallel 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)

Parallel BB (cont'd)

- Need an effective work scheduling and load balancing scheme to do parallel tree search efficiently
- Manager-worker schemes (centralized global stack management) are popular but may scale poorly due to communication expense and bottlenecks
- Many implementations of parallel BB have been studied (Kumar et al., 1994; Gendron and Crainic, 1994) for various target architectures
- There are various BB and BP schemes; we use an interval Newton/generalized bisection (IN/GB) method

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)

- Target architecture: Distributed computing on a networked cluster using message passing
  - Often relatively inexpensive
  - Uses widely available hardware

- Use distributed (multiple pool) load balancing

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

- Components of typical schemes
  - Workload state measurement
  - State information exchange
  - Transfer initiation
  - Workload placement
  - Global termination

Components

- Workload state measurement
  - Evaluate local workload using some “work index”
  - Use stack length: number of intervals (boxes) remaining to be processed

- State information exchange
  - Communicate local workload state to other “cooperating” processors
  - Selection of cooperating processors defines a virtual network
  - Virtual network: Global (all-to-all), 1-D torus, 2-D torus, etc.

- Transfer initiation
  - Sender initiate
  - Receiver initiate
  - Symmetric (sender or receiver initiate)

Components (cont'd)

- Workload placement
  - Work-adjusting rule: How to distribute work (boxes) among cooperating processors and how much to transfer
    - Work stealing (e.g., Blumofe and Leiserson, 1994)
    - Diffusive propagation (e.g., Heirich and Taylor, 1995)
    - Etc.
  - Work-selection rule: Which boxes should be transferred
    - Breadth first
    - Best first (based on the lower bound value)
    - Depth first
    - Various heuristics
- Global termination
  - Easy to detect with synchronous, all-to-all communication
  - For local and/or asynchronous communication, use Dijkstra's token algorithm

Parallel Implementations

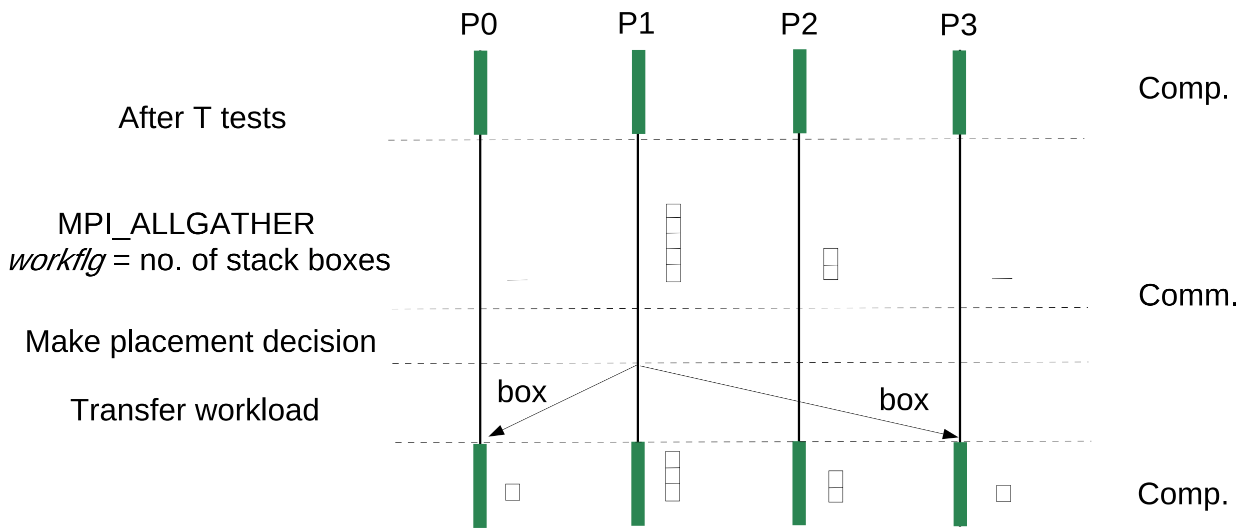
- Three types of strategies were implemented.
  - Synchronous Work Stealing (SWS)
  - Synchronous Diffusive Load Balancing (SDLB)
  - Asynchronous Diffusive Load Balancing (ADLB)

- These are listed in order of likely effectiveness.

- All were implemented in Fortran-77 using LAM (Local Area Multicomputer) MPI (Laboratory for Scientific Computing, University of Notre Dame)

Synchronous Work Stealing

- Periodically exchange 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 (receiver initiate)
- 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 work state and units of work with their immediate neighbors to maintain their workload

- Typical workload adjusting scheme (symmetric initiation):

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

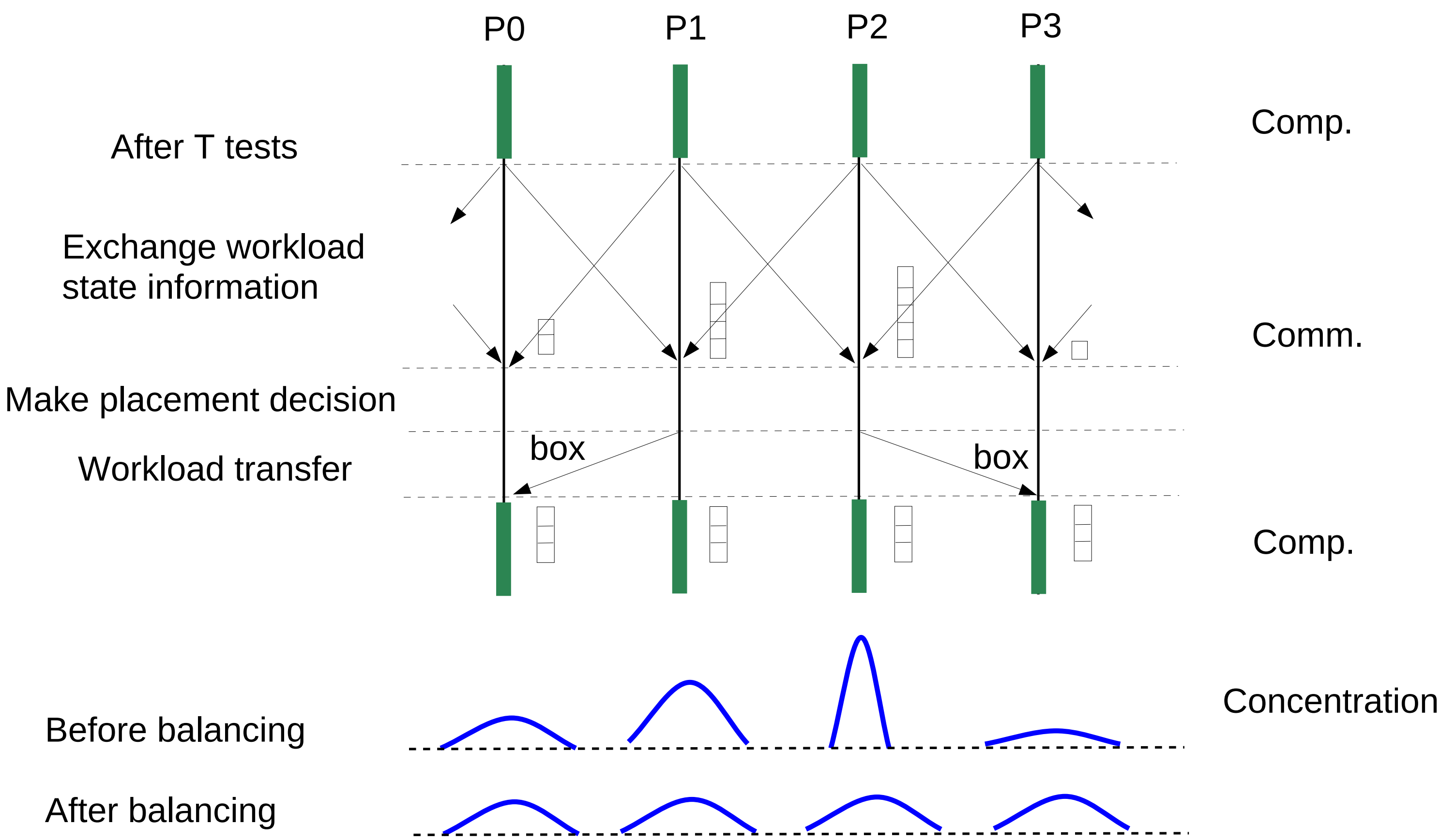
(i: local processor: j: neighbor processor)

  - 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)

- Messages have higher granularity

- Synchronism and blocking communication still cause inefficiencies

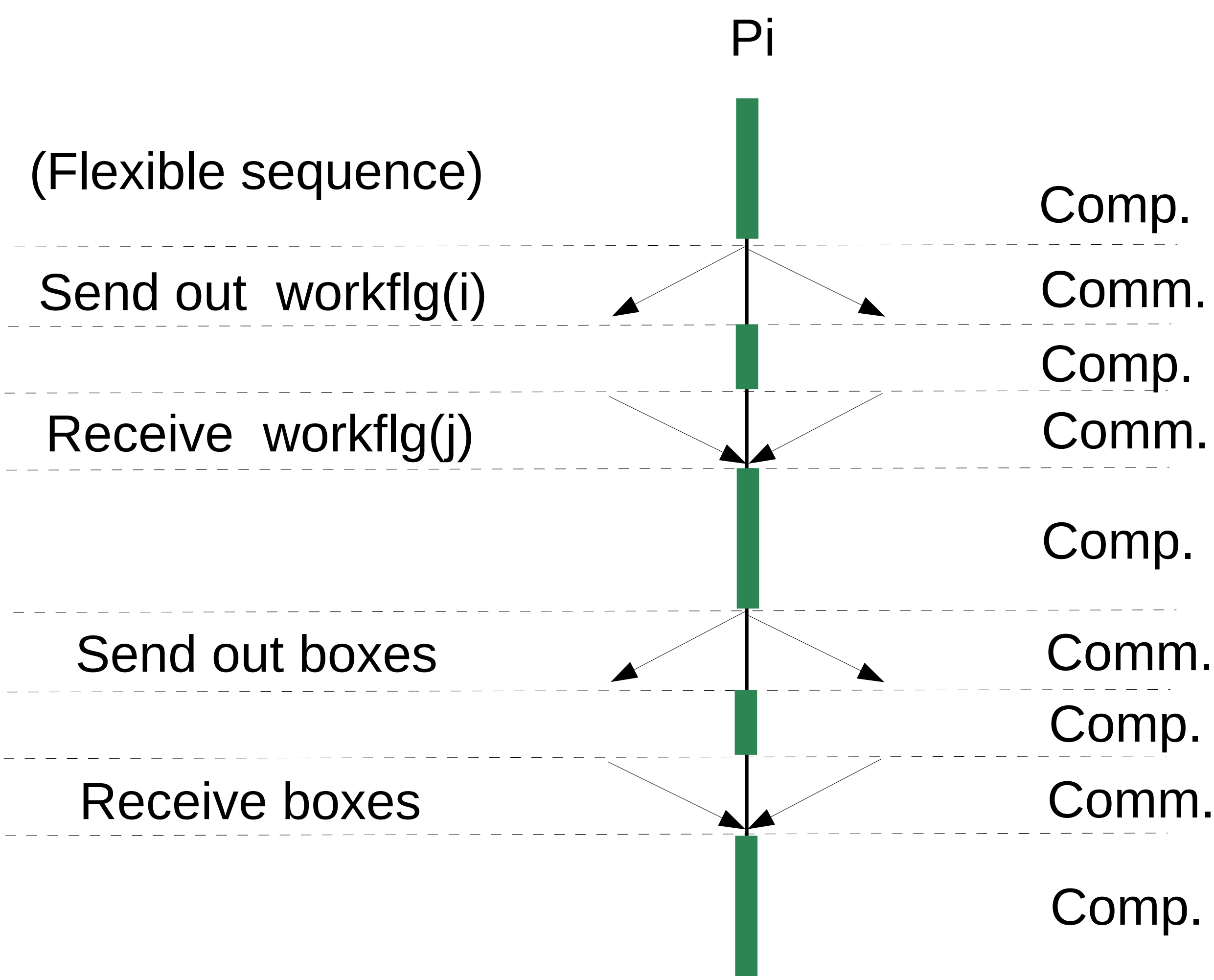
Synchronous Diffusive Load Balancing



Asynchronous Diffusive Load Balancing

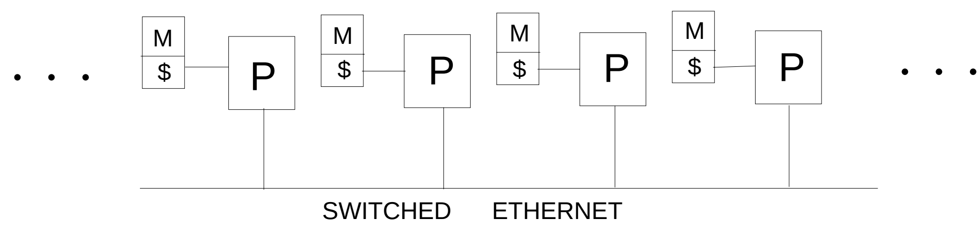
- Use asynchronous nonblocking communication to send workload information and transfer workload
- Overlaps communication and computation
- Receiver-initiated diffusive workload transfer scheme:
  - Send out work state information only if it falls below some threshold
  - Donor processor follows diffusive scheme to determine amount of work to send (if any)
  - Recognizes that workload balance is less important than preventing idle states
- Dijkstra's token algorithm used to detect global termination

Asynchronous Diffusive Load Balancing



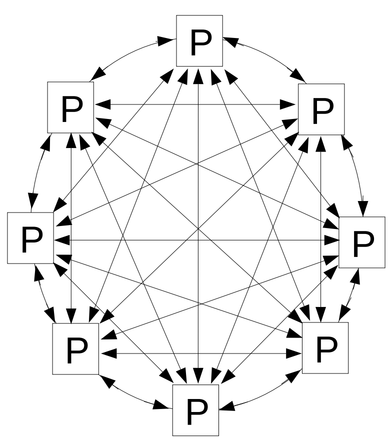
Testing Environment

- Physical hardware: Sun Ultra workstations connected by switched Ethernet (100Mbit)



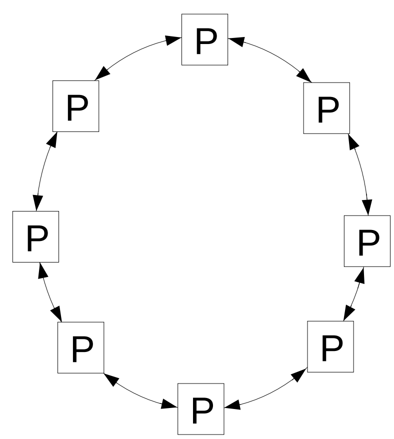
- Virtual Network:

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

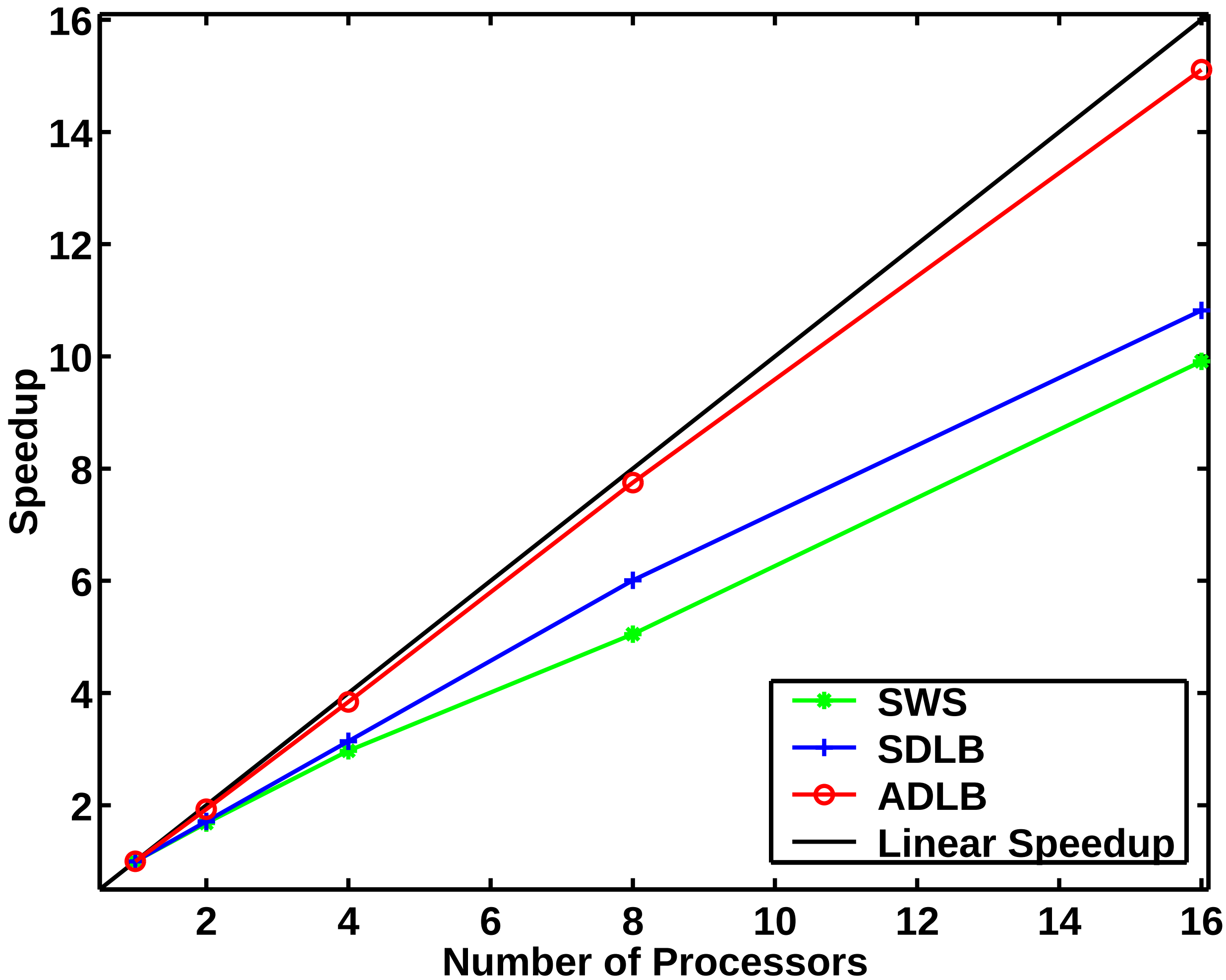
- Can be formulated as a nonlinear equation solving problem (which has five solutions)

- Or can be formulated as a global optimization problem

Comparison of Algorithms on Equation-Solving Problem

Speedup vs. Number of Processors

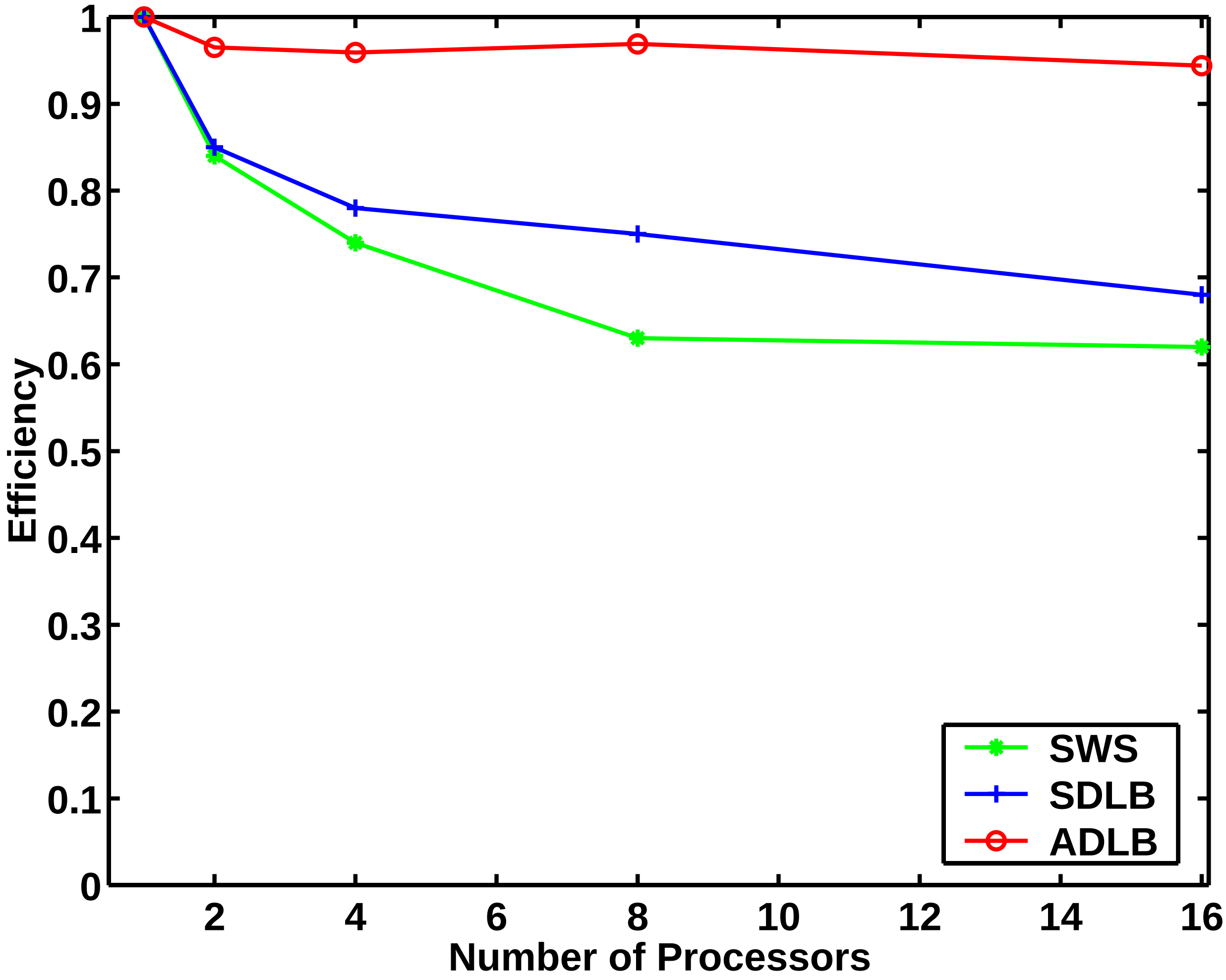
ADLB vs. SDLB vs. SWS



Comparison of Algorithms on Equation-Solving Problem

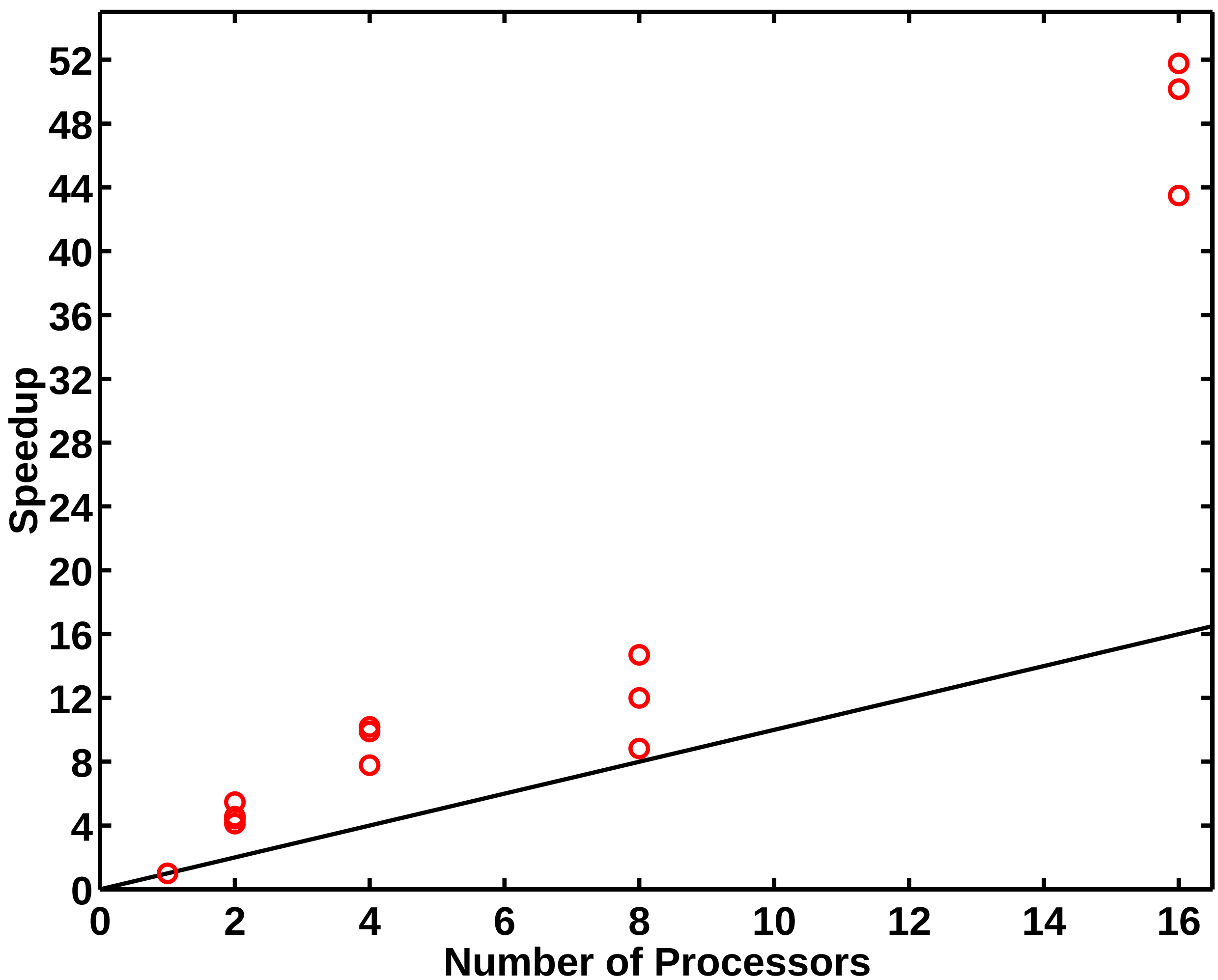
Efficiency vs. Number of Processors

ADLB vs. SDLB vs. SWS



Using ADLB on Optimization Problem

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



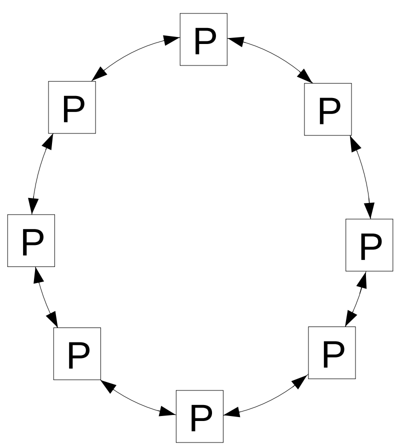
Using ADLB on Optimization Problem

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

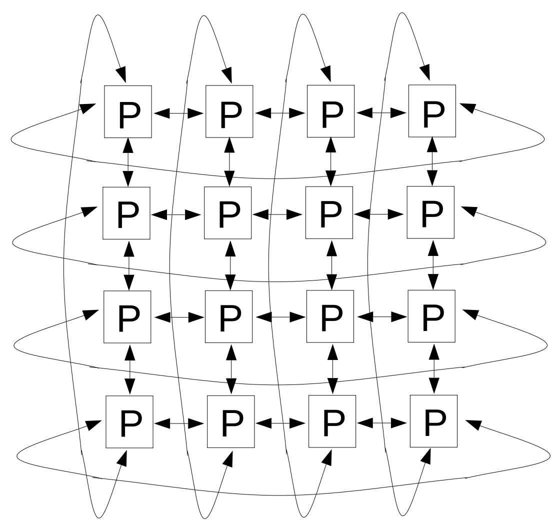
Effect of Virtual Network

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

1-D Torus Network



2-D Torus Network



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

Effect of Virtual Network

- ADLB algorithm was tested using both 1-D and 2-D virtual connectivity.

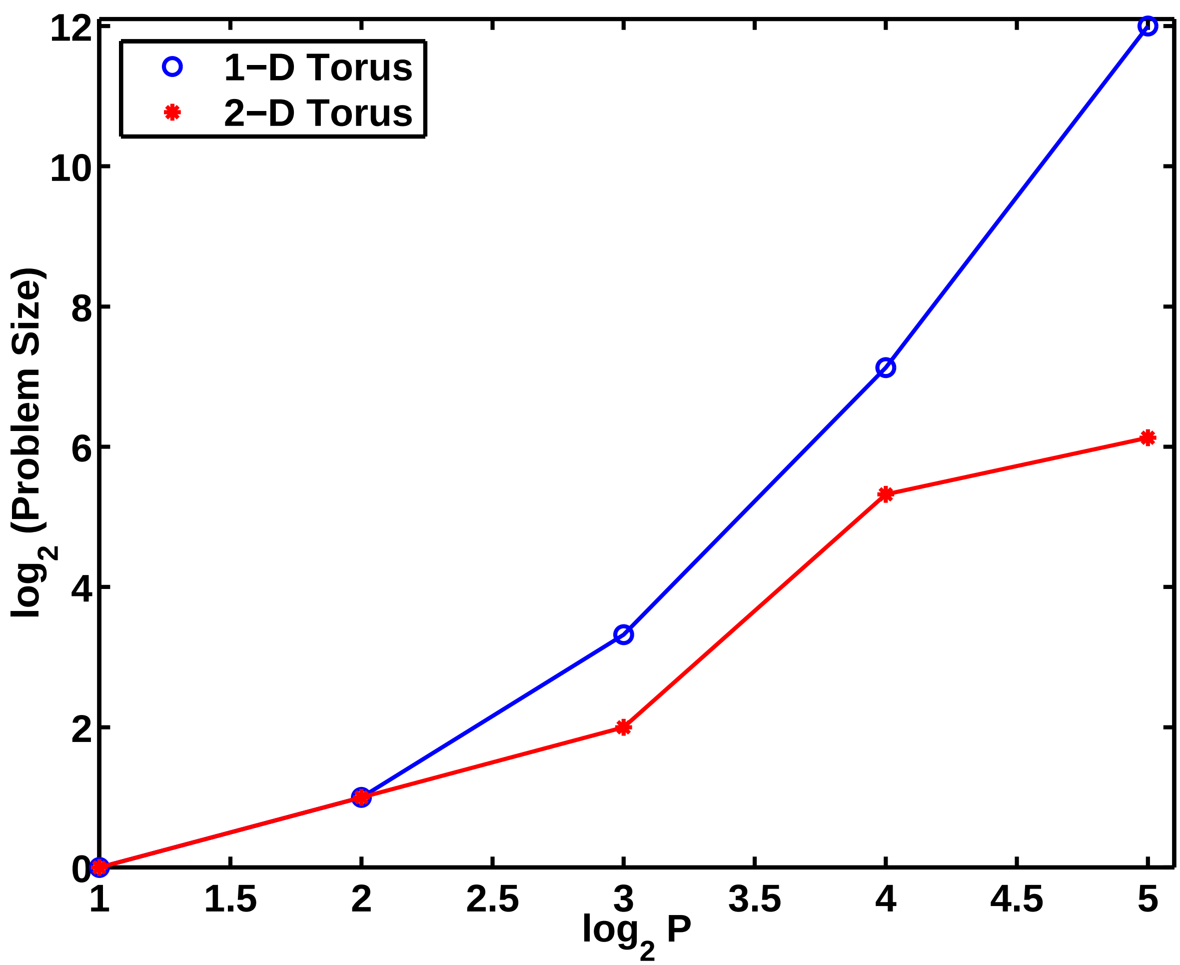
- The test problem is an equation solving problem: computation of critical points of mixtures

- Comparisons made using isoefficiency analysis: As number of processors is increased, determine problem size needed to maintain constant efficiency relative to best sequential algorithm

- Isoefficiency curves at 92% were determined up to 32 processors

Isoefficiency Curves (92%) for Equation-Solving Problem

2-D Torus vs. 1-D Torus
(Lower is better)



Stack Management for Workload Placement

- Especially for optimization problems, the selection rule for workload transfer can have a significant effect on performance

- With the goal of maintaining consistently high (superlinear) speedups on optimization (BB) problems, we have used a dual stack management scheme

- Each processor maintains two workload stacks, a local stack and a global stack
  - The processor draws work from the local stack in the order in which it is generated (depth-first pattern)
  - The global stack provides work for transmission to other processors
  - The global stack is created by randomly removing boxes from the local stack, contributing breadth to the tree search process

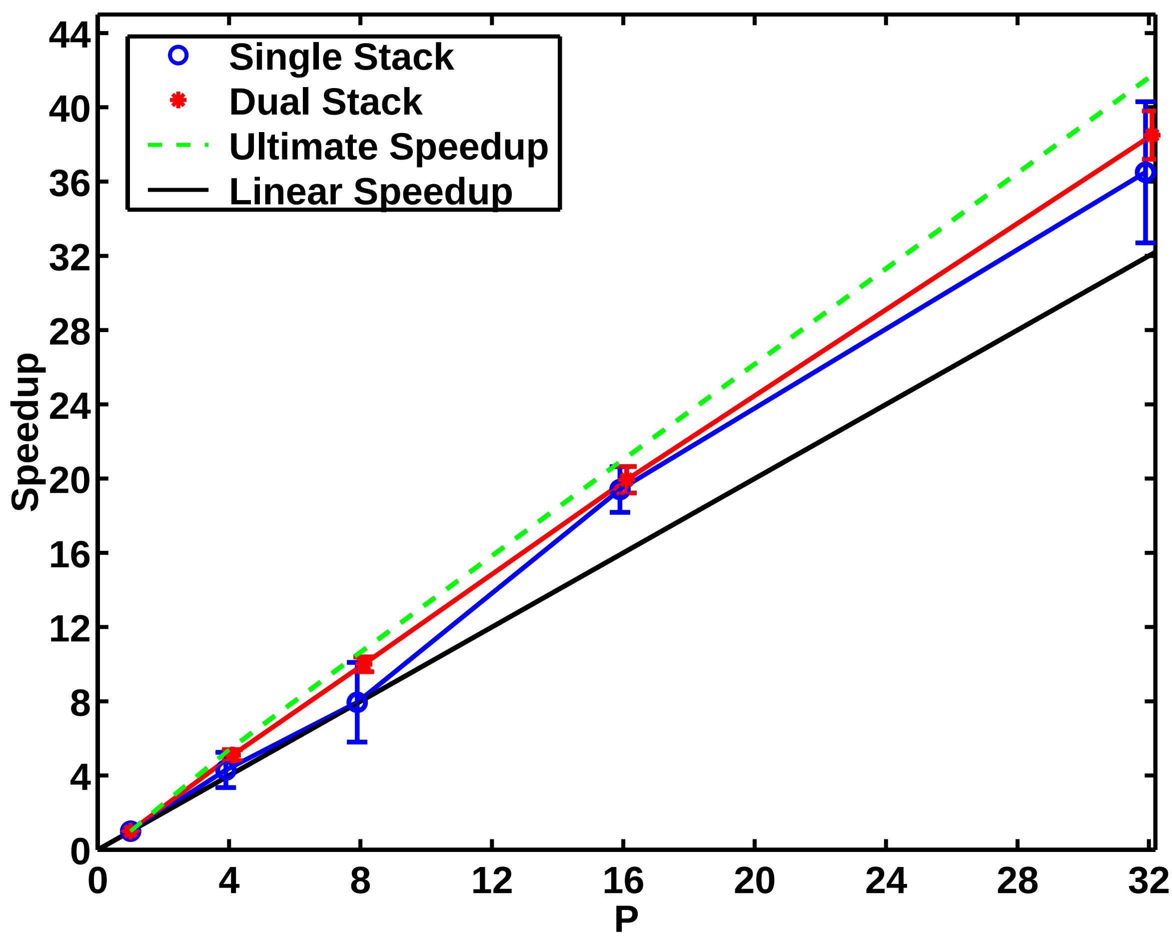
Workload Placement (cont'd)

- The dual stack strategy was tested using a 2-D torus virtual network up to 32 processors
- The test problem was an optimization problem: parameter estimation using an error-in-variable approach
- For comparisons, an “ultimate speedup” was determined by initially setting the best upper bound to the value of the global minimum
- Results indicate that the dual stack strategy leads to higher speedups and less variability from run to run (based on 10 runs of each case)

Workload Placement (cont'd)

Speedup vs. Number of Processors

Dual Stack vs. Single Stack vs. Ultimate



Concluding Remarks

- IN/GB is a powerful **general-purpose** and **model-independent** approach for solving a variety of process modeling problems, providing a **mathematical and computational guarantee** of reliability

- Continuing advances in computing hardware and software (e.g., compiler support for interval arithmetic, parallel computing) will make this approach even more attractive

- With effective load management strategies, parallel BB and BP problems (using IN/GB or other approaches) can be solved very efficiently using MPI on a networked cluster of workstations
  - Good scalability
  - Exploit potential for superlinear speedup in BB

- Parallel computing technology can be used not only to solve problems faster, but to **solve problems more reliably**

Acknowledgments

- ACS PRF 30421-AC9 and 35979-AC9
- NSF DMI96-96110 and EEC97-00537-CRCD
- US ARO DAAG55-98-1-0091
- Sun Microsystems, Inc.

For more information:

- Contact Prof. Stadtherr at markst@nd.edu
- See also
  http://www.nd.edu/˜markst