Global Nonlinear Parameter Estimation Using Interval Analysis: Parallel Computing Strategies

Chao-Yang Gau and Mark A. Stadtherr[1]
Department of Chemical Engineering
University of Notre Dame
Notre Dame, IN 46556 USA

AIChE Annual Meeting, Miami, FL, November 15–20, 1998
Session Number 244: High Performance Computing

[1]Author to whom all correspondence should be addressed. Phone: (219)631-9318; Fax: (219)631-8366; E-mail: markst@nd.edu

Outline

- Background
  - Parameter Estimation
  - Interval Newton Method
- Sequential Example
- Parallel Computing Strategies
- Parallel Examples

Background—Parameter Estimation

- Observations $y_{\mu i}$ of $i = 1, \ldots, q$ responses from $\mu = 1, \ldots, p$ experiments are available.

- Responses are to be fit to a model $y_{\mu i} = f_i(\mathbf{x}_\mu, \boldsymbol{\theta})$ with independent variables $\mathbf{x}_\mu = (x_{\mu 1}, \ldots, x_{\mu m})^T$ and parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^T$.

- Various objective functions $\phi(\boldsymbol{\theta})$ can be used to determine the parameter values that provide the "best" fit, e.g.
  - Relative least squares
  - Maximum likelihood

- Optimization problem to determine parameters can be formulated as either a constrained or unconstrained problem. In the unconstrained case, the experimental observations are substituted directly into the objective function. The unconstrained formulation is used here.

Background—Parameter Estimation (continued)

- Assuming a relative least squares objective and using an unconstrained formulation, the problem is

$$\min_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}) = \sum_{i=1}^{q} \sum_{\mu=1}^{p} \left[\frac{y_{\mu i} - f_i(\mathbf{x}_\mu, \boldsymbol{\theta})}{y_{\mu i}} \right]^2$$

- Equation-solving approach:
  - A common approach for solving this problem is to use the gradient of $\phi(\boldsymbol{\theta})$ and to seek the stationary points of $\phi(\boldsymbol{\theta})$ by solving $\mathbf{g}(\boldsymbol{\theta}) \equiv \nabla\phi(\boldsymbol{\theta}) = \mathbf{0}$.
  - Interval Newton technique can provide the capacity to find all roots, and insure that the global minimum is found.

- Optimization approach:
  - Interval Newton can be combined with an upper bound test so that roots of $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ that cannot be the global minimum need not be found.

Interval Newton Method

- For the system of nonlinear equations $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$, find (enclose) **with mathematical and computational certainty** all roots in a given initial interval $\boldsymbol{\Theta}^{(0)}$ or determine that there are none.

- At iteration k, given the interval $\boldsymbol{\Theta}^{(k)}$, if $0 \in \mathbf{G}(\boldsymbol{\Theta}^{(k)})$ solve the linear interval equation system

$$G'(\boldsymbol{\Theta}^{(k)})(\mathbf{N}^{(k)} - \boldsymbol{\theta}^{(k)}) = -\mathbf{g}(\boldsymbol{\theta}^{(k)})$$

  for the "image" $\mathbf{N}^{(k)}$, where $\mathbf{G}(\boldsymbol{\Theta}^{(k)})$ is an interval extension of $\mathbf{g}(\boldsymbol{\theta})$ and $G'(\boldsymbol{\Theta}^{(k)})$ an interval extension of its Jacobian over the current interval $\boldsymbol{\Theta}^{(k)}$, and $\boldsymbol{\theta}^{(k)}$ is a point inside $\boldsymbol{\Theta}^{(k)}$.

- Any root $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}^{(k)}$ is also contained in the image $\mathbf{N}^{(k)}$, suggesting the iteration scheme $\boldsymbol{\Theta}^{(k+1)} = \boldsymbol{\Theta}^{(k)} \cap \mathbf{N}^{(k)}$ (Moore, 1966).

- Interval Newton also provides an existence and uniqueness test:

Interval Newton Method (continued)

- *True*: If $\mathbf{N}^{(k)} \subset \mathbf{\Theta}^{(k)}$, then there is a **unique** zero of $\mathbf{g}(\boldsymbol{\theta})$ in $\mathbf{\Theta}^{(k)}$, and the point Newton method will converge quadratically to the root starting from any point in $\mathbf{\Theta}^{(k)}$.

- *False*: If $\mathbf{\Theta}^{(k)} \ \cap \ \mathbf{N}^{(k)} = \emptyset$ or $0 \notin \mathbf{G}(\mathbf{\Theta}^{(k)})$, then there is no root in $\mathbf{\Theta}^{(k)}$.

- *Unknown*: Otherwise, then either:
  - Continue with the next iterate $\mathbf{\Theta}^{(k+1)}$ if it is sufficiently smaller than $\mathbf{N}^{(k)}$, or
  - **Bisect** $\mathbf{\Theta}^{(k+1)}$ and perform interval Newton on the resulting intervals.

  This is the interval Newton/generalized bisection (IN/GB) approach.

- Basically, it follows a **branch-and-prune** scheme :
  - If test is true or false, then prune node.
  - If test is unknown and bisect, then branch (bisect node), generating a binary tree structure.

Interval Newton Method (continued)

- For optimization problems, a node is pruned if the interval extension $\mathbf{\Phi}(\mathbf{\Theta})$ of the objective $\phi(\boldsymbol{\theta})$ has a lower bound greater than the current best (least) upper bound.

- Best upper bound is determined and updated by:
  - Upper bound of $\mathbf{\Phi}(\mathbf{\Theta})$, and/or
  - Point function evaluations with interval arithmetic in each interval tested, and/or
  - Running a local optimizer.
  - Verify local methods with interval arithmetic.

Sequential Example Problem : Parameter Estimation in VLE Modeling

- Goal: Determine energy parameter values in the Wilson model for the binary system water(1) and formic acid(2) using the relative least squares objective (Gau and Stadtherr, FOCAPO 98).

- Twelve problems, each a different data set from the DECHEMA VLE Data Collection (Gmehling *et al.*, 1977-1990) were solved using IN/GB approach to determine the globally optimal value of parameters.

- Using the interval approach, the global minimum was found for all problems.

- For several problems the result presented in DECHEMA represents a local not global minimum, and does not achieve the globally best fit.

TABLE 1: IN/GB results vs. DECHEMA values

Data Set	P (mm Hg)	DECHEMA			IN/GB			No. of Minima
		θ_1	θ_2	$\phi(\boldsymbol{\theta})$	θ_1	θ_2	$\phi(\boldsymbol{\theta})$	
1	760	-195	759	0.0342	-168	759	0.0342	2
2	760	-278	1038	0.0106	-278	1038	0.0106	2
3	760	-310	1181	0.0151	-308	1167	0.0151	2
4	760	-282	985	0.353	-282	984	0.353	2
5	760	-366	1513	0.0257	-365	1509	0.0257	3
6	760	1067	-1122	0.0708	1065	-1120	0.0708	2
7*	200	892	-985	0.141	**-331**	**1250**	**0.0914***	2
8*	200	370	-608	0.0459	**-340**	**1404**	**0.0342***	3
9*	100	539	-718	0.165	**-285**	**996**	**0.111***	2
10*	100	450	-663	0.151	**-329**	**1394**	**0.0819***	3
11*	70	558	-762	0.0399	**-330**	**1519**	**0.0372***	3
12	25	812	-1058	0.0502	807	-1055	0.0502	2

*New globally optimal parameters found!

Parallel Computing and IN/GB

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

- Multiple processors can be used to concurrently perform IN/GB in disjoint parts (intervals) of the binary tree.

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

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

- Three types of algorithms designed for network-based parallel computing were studied.
  - Synchronous Work Stealing (SWS)
  - Synchronous Diffusive Load Balancing (SDLB)
  - Asynchronous Diffusive Load Balancing (ADLB)

Work Scheduling and Load Balancing

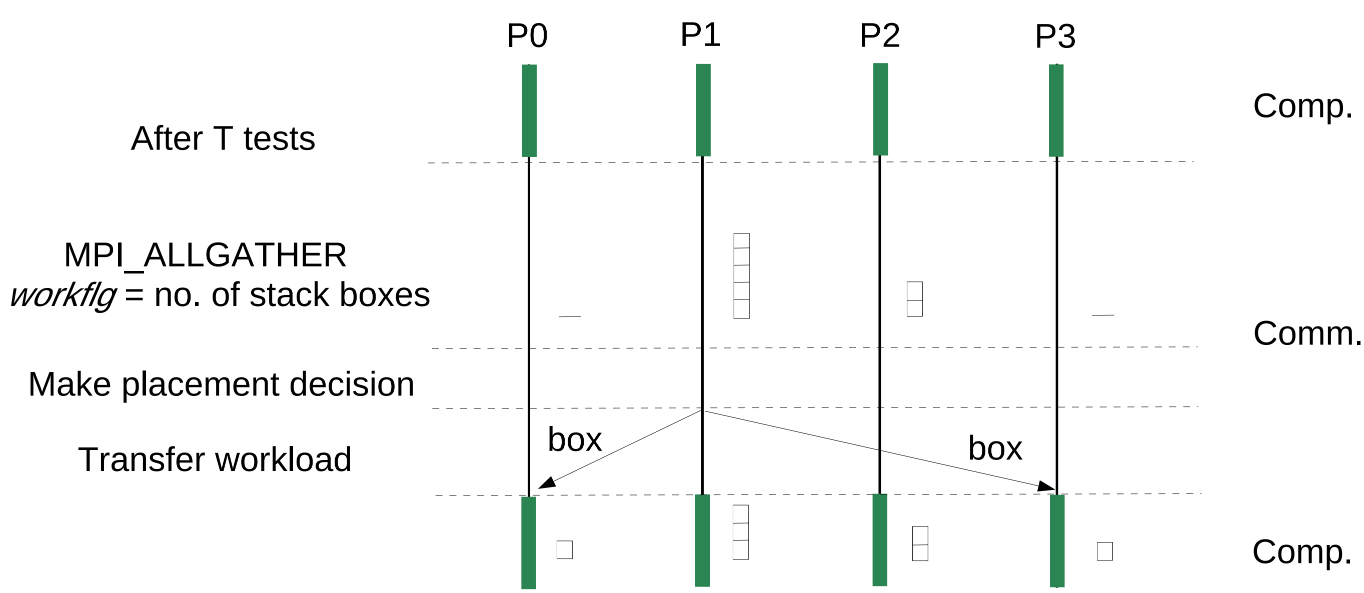
- Objective: Schedule the workload among processors to minimize communication delays and execution, and maximize computing resource utilization.
- Use Dynamic Scheduling
  - Redistribute workload concurrently at runtime.
  - Transfer workload from a heavily loaded processor to a lightly loaded one, performing load balancing.
- Use Distributed Load Balancing
  - Each processor locally makes a workload placement decision to maintain a local interval stack and prevent itself from becoming idle.
  - Alleviate bottleneck effects presented in the centralized policy (manager/worker).
  - Improvements in communication overhead could provide high scalability for the multi-processor 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.

- Major problems
  - Large network overhead (global, all-to-all)
  - Large idle time from process synchronism and blocking communication



Synchronous Diffusive Load Balancing

- Use Local Communication : Processors periodically exchange workload infomation and units of work with their immediate neighbors to maintain a moderate workload , not too heavy or too light.

- Reduce the appearance of idle states.

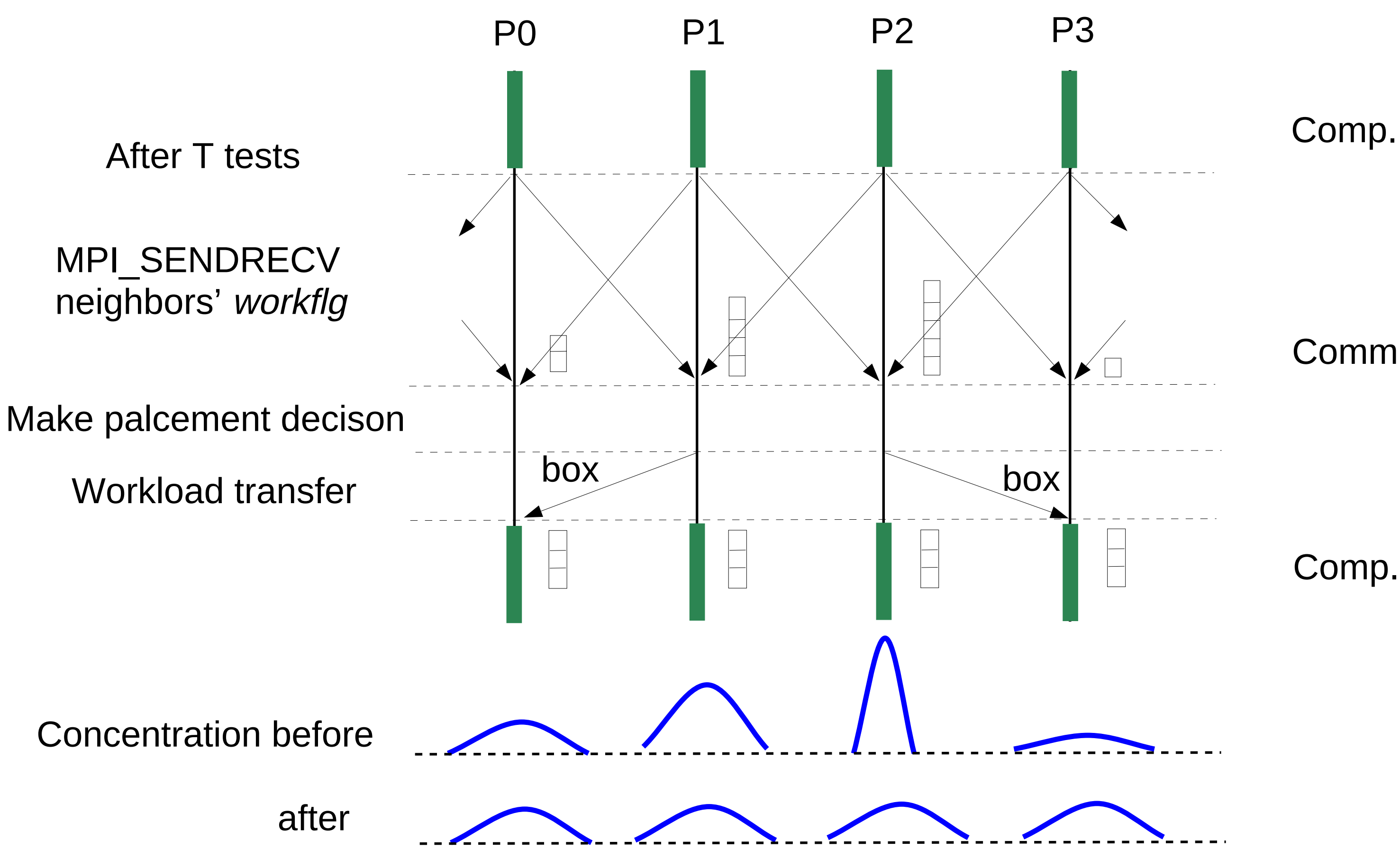
- Workload adjusting scheme:

$$u(j) = 1/2(workflg(i) - workflg(j))$$

  i: local processor, j neighbor processor
  (a) Positive $u(j)$: send intervals(boxes).
  (b) Negative $u(j)$: receive intervals (boxes).

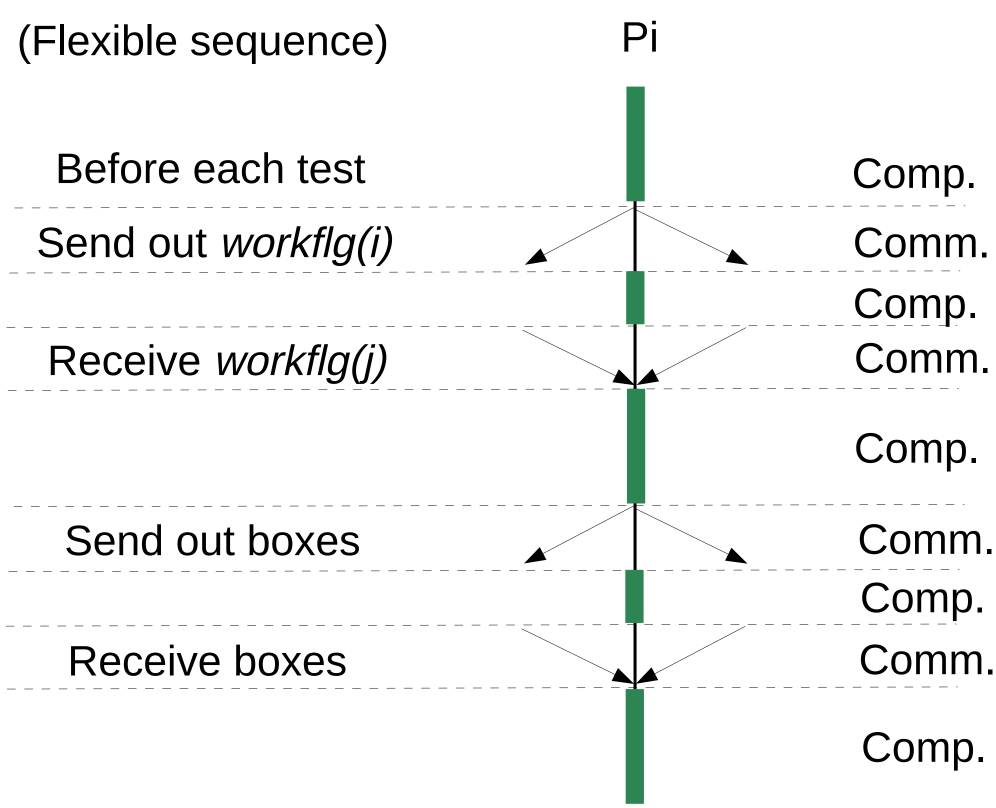
- Major problems
  - Synchronism inefficiency.
  - Termination effects arising from local communication strategy and diffusive message propagation.

Synchronous Diffusive Load Balancing (Continued)



Asynchronous Diffusive Load Balancing

- Use asynchronous nonblocking comm., MPI_ISEND, to update workload info. and transfer workload, and break process synchronization.
- Overlap communication and computation
- Just maintain the local workload (number of stack boxes) higher than some threshold.



Test Problem for Parallel Computation

- Parameter estimation for data set-10 of the water and formic acid system using the maximum likehood estimator as the objective function:

$$\phi(\boldsymbol{\theta}, \boldsymbol{V}) = (n + \mu + 1) \log V_1 V_2$$

$$+ \sum_{i=1}^{n} \sum_{\mu=1}^{p} \left[\frac{\gamma_{\mu i,\mathrm{calc}}(\boldsymbol{\theta}) - \gamma_{\mu i,\mathrm{exp}}}{V_\mu} \right]^2,$$

  where $\boldsymbol{V}$ is a diagonal covariance matrix with unknown elements V_μ.

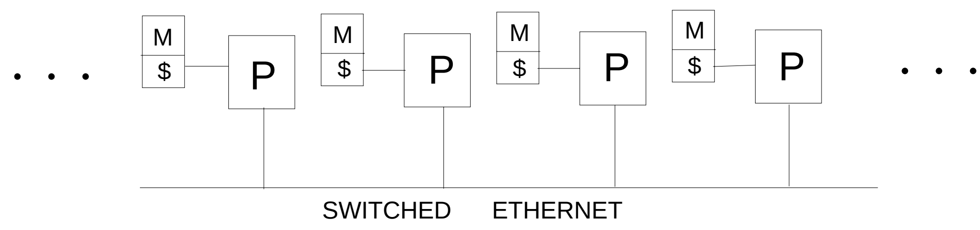
- This four-variable problem can also be treated as either an equation-solving or global optimization problem.

- This is a difficult problem with five stationary points.

Testing Environment

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

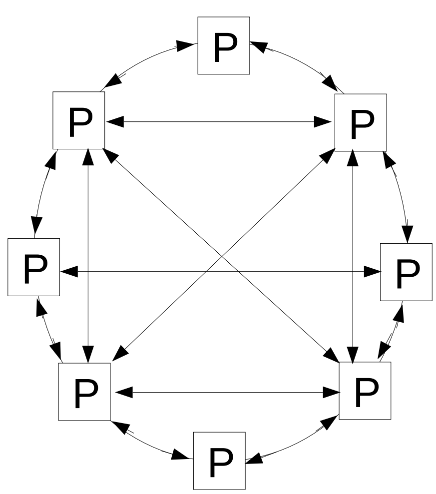
- Physical Hardware: Sun Ultra 1/140e workstations connected by switched Ethernet



- Virtual Network:

Global Communication

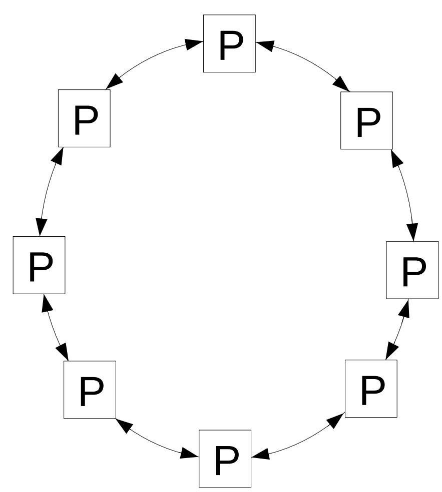
Star Network



Used for SWS

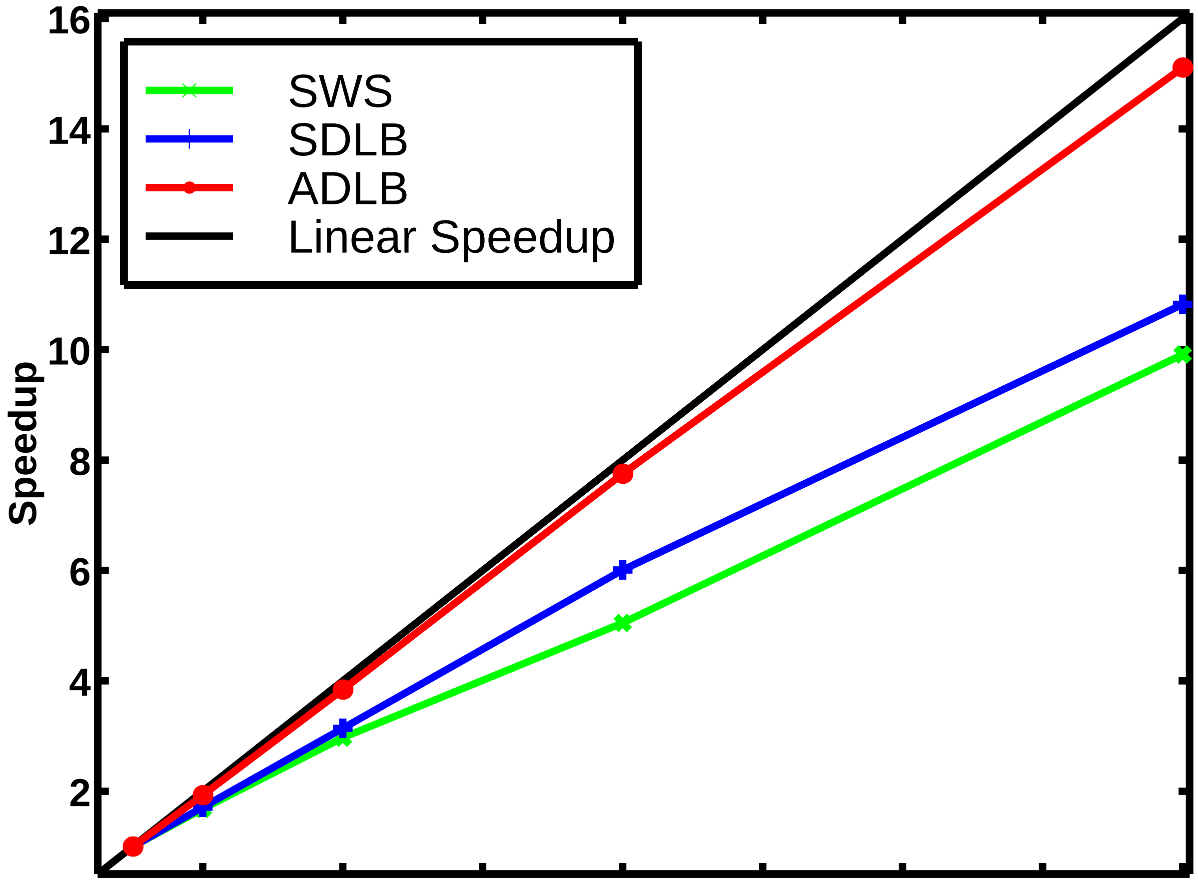
Local Communication

1-D Torus Network

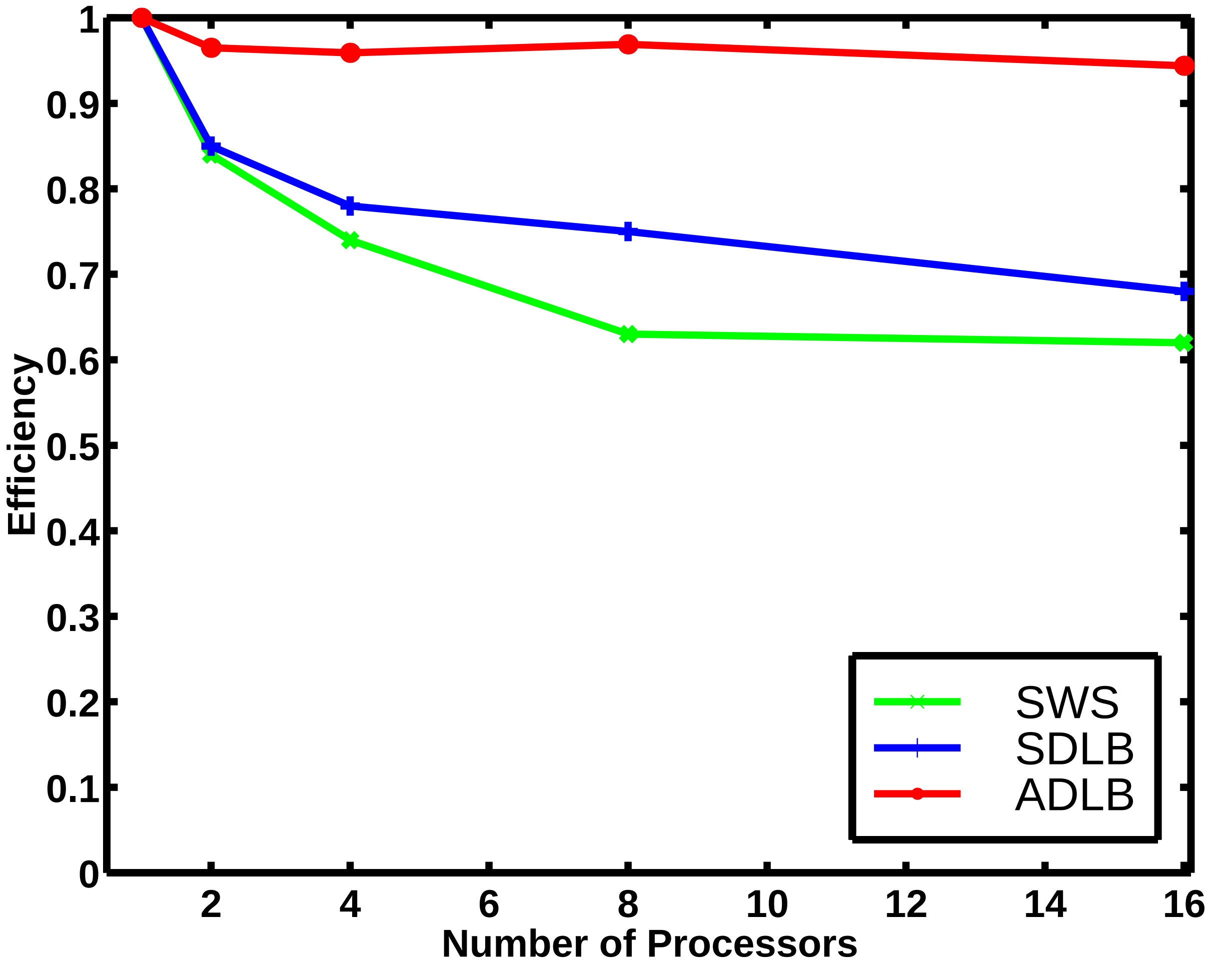


Used for SDLB and ADLB

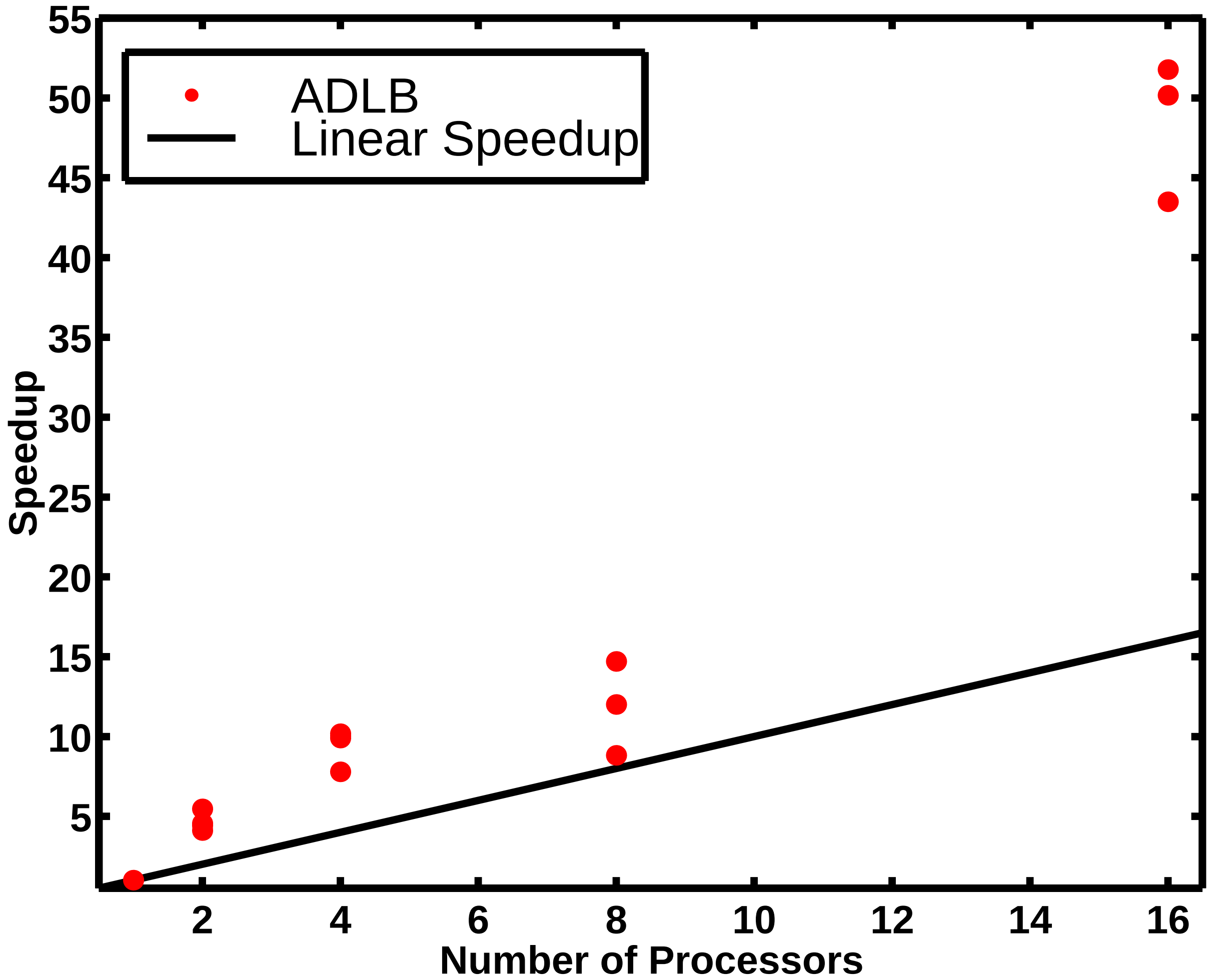
Comparison of Three Algorithms for Equation-Solving Problem - Speedup



Comparison of Three Algorithms for Equation-Solving Problem - Efficiency



Results for Optimization Problems Using ADLB



Results for Optimization Problems Using ADLB (continued)

- Superlinear Speedup: Broadcast of least upper bounds results in less work to do in parallel case.

- Speedup Anomaly: Results vary from run to run because of different timing in finding and broadcasting improved upper bound.

Other Applications

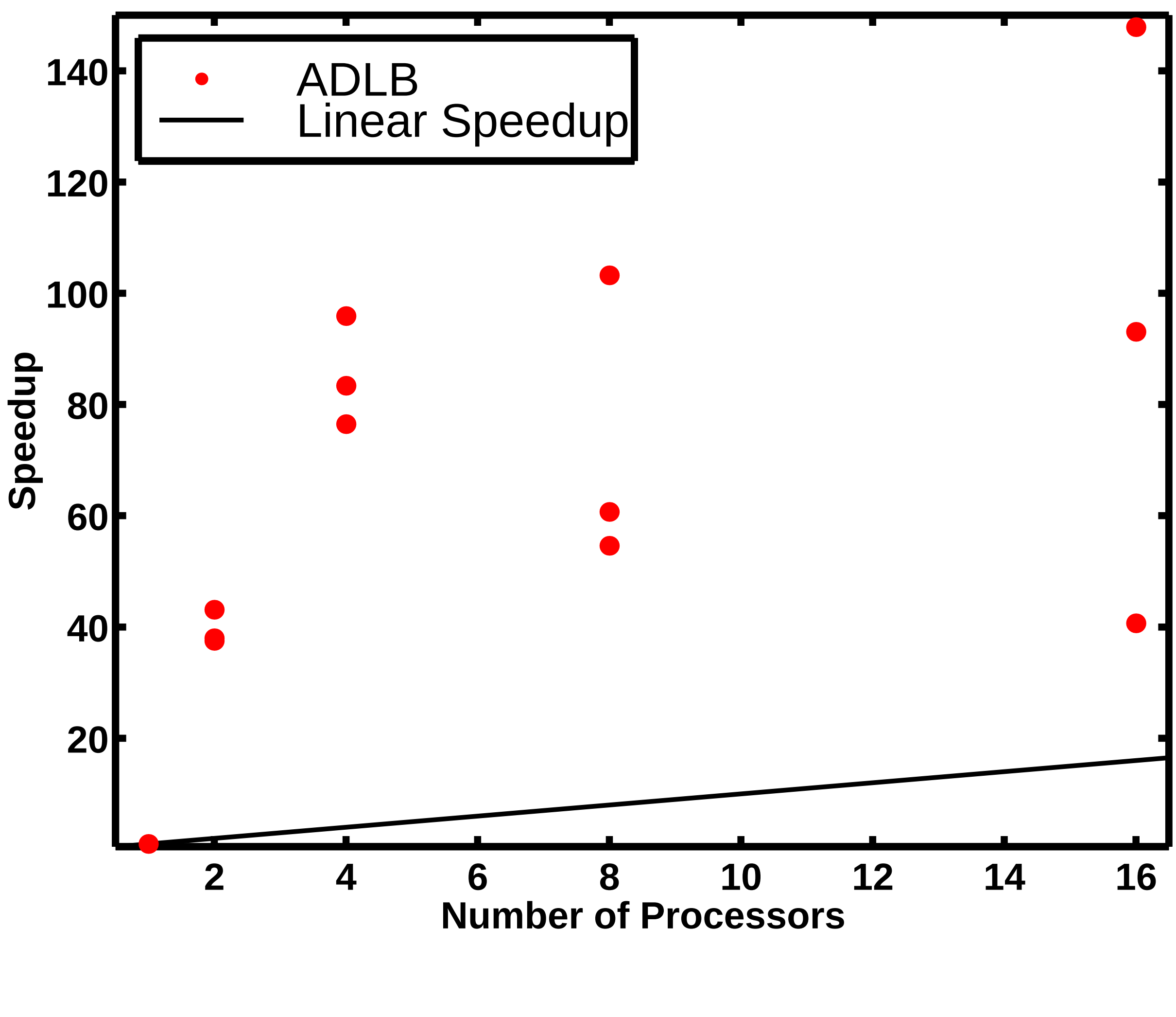
- The developed dynamic load balancing algorithms are general-purpose, and are straitforward to apply in solving other problems.

- Example: A six-component phase stability problem (Tessier *et. al.*, 1998) was set up and treated as a global optimization problem solved using IN/GB with upper bound test.

- Parallel runs done using ADLB algorithm.

- Superlinear speedup and speedup anomalies can also be observed in this kind of optimization problem.

Results for Phase Stability Optimization Problem



Concluding Remarks

- Interval analysis is a **general-purpose** and **model-independent** approach for solving parameter estimation problems, providing a **mathematical and computational guarantee** that the global optimum is found.
  - Other VLE models could be used.
  - Other objective functions (e.g., error-in-variables method) could be used.
  - Other types of data could be used.

- Three dynamic load balancing algorithms have been developed to enhance the parallel computing efficiency of interval approach.

- The best performance was obtained when using asynchronous diffusive load balancing algorithm (ADLB), resulting in nearly linear speedup, for equation-solving problems, and superlinear speedup for optimization problems.

- Acknowledgments
  - ACS PRF 30421-AC9
  - NSF CTS95-22835, DMI96-96110, EEC97-00537-CRCD
  - EPA R824731-01-0
  - DOE DE-FG07-96ER14691
  - Sun Microsystems, Inc.

- For more information:
  - Contact Prof. Stadtherr at markst@nd.edu
  - See also
    http://www.nd.edu/~markst