A Systematic Analysis of Dynamic Load Balancing Strategies for Parallel Interval Analysis

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

AIChE Annual Meeting, Dallas, TX, Oct. 31– Nov. 5, 1999
Session Number 10D02: Application of Parallel Computing Strategies in Engineering Systems

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

Outline

- Parallel Branch-and-Prune (and/or Bound)
  - Interval Newton/Generalized Bisection (IN/GB)
- Asynchronous Diffusive Load Balancing
- Analysis
  - Virtual Connectivity Analysis
  - Scalability Analysis (Equation-Solving Problems)
  - Speedup Anomaly Analysis (Global Optimization)
- Experiments and Results
- Concluding Remarks

Parallel Branch-and-Prune (Bound)

- Branch-and-Prune (and/or Bound)
  - A tree search algorithm is often used in intelligent search.
  - Successive decompositions into smaller disjoint (independent) subproblems.
  - Capability of finding all solutions or the globally optimal solution.
  - Many applications: nonlinear mixed-integer and global optimization, combinatorial problems, interval analysis etc ...

- Parallel Processing
  - An additional source of improvement in search efficiency.
  - Implementation with dynamic load balancing and work distribution.

Interval Newton Method

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

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

$$F'(\mathbf{X}^{(k)})(\mathbf{N}^{(k)} - \boldsymbol{x}^{(k)}) = -\mathbf{f}(\boldsymbol{x}^{(k)})$$

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

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

- Interval Newton also provides an existence and uniqueness test:

Interval Newton Method (continued)

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

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

- *Unknown*: Otherwise, then either:
  - Continue with the next iterate $\mathbf{X}^{(k+1)}$ if it is sufficiently smaller than $\mathbf{N}^{(k)}$, or
  - **Bisect** $\mathbf{X}^{(k+1)}$ 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.

Asynchronous Diffusive Load Balancing

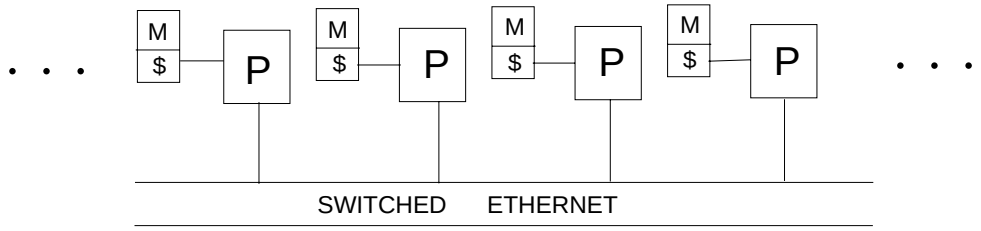
- Irregular parallel search trees need load balancing to redistribute workload concurrently at runtime.

- Distributed System: coordinate all processors to maintain distributed interval stacks and prevent idle states through workload transfer over network.

- Asynchronous Diffusive Load Balancing (ADLB) [Gau and Stadtherr, 1998]
  - Use asynchronous nonblocking and persistent communication to update workload information and transfer workload.
  - Overlap communication and computation and reduce idle state.
  - Receiver Initiate and Local Communication: Exchange workload information with their immediate neighbors, once local workload less than certain threshold, request neighbors for stack boxes.

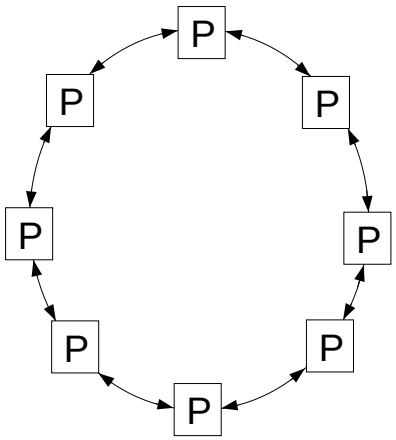
Parallel System

- Physical Architecture:
  Network-based system - Sun Ultra Enterprise 2 workstations connected by switched Ethernet

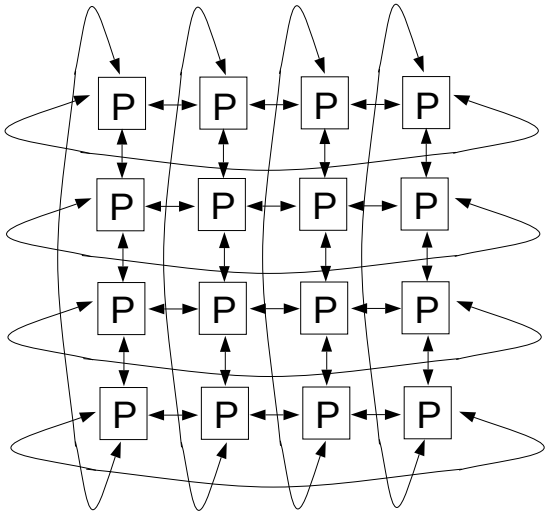


- Virtual Algorithm
  - Software: implemented in Fortran 77 using message-passing interface (MPI) protocol
  - Virtual Network: local communication

1-D Torus Network



2-D Torus Network



Comparison on Virtual Connectivity

- Virtually connectivity dominate propagation of messages and distribution of workload among processors.

- Comparison of 1-D vs. 2-D torus
  - Communication overhead:
    2 neighbors vs. 4 neighbors
  - Message diffusion distance:
    $P/2$ vs. $\sqrt{P}/2$
  - Mechanism for workload diffusion:
    uni-directional flow vs. bi-directional flow

- As number of processors increases, 2-D torus might have an edge over 1-D torus.

- Examined by scalability analysis over instances with sequential best algorithm (equation-solving problems)

Scalability Analysis - Definition

- Phenomena in parallel computing due to Amdahl's law:
  - Efficiency declines when problem size is fixed and processor number increases.
  - Efficiency climbs when processor number is fixed and problem size increases.

- Scalable parallel system: maintain constant performance (efficiency) as problem size and computer size increase

- Isoefficiency function [Kumar, 1994]:
  Relate problem size to processor number necessary for increase in speedup in proportion to processor number *using sequential best algorithm.*

- Small isoefficiency function represents highly scalable parallel algorithm.

- Applications: select best parallel algorithm and predict performance of specific parallel algorithm.

Scalability Analysis - Experiments and Results

- Parallel algorithms with 1-D and 2-D virtual connectivity, respectively, were examined by scalability analysis.

- An equation-solving problem, computation of critical points in mixtures [Stradi,1999], was selected as the test example.

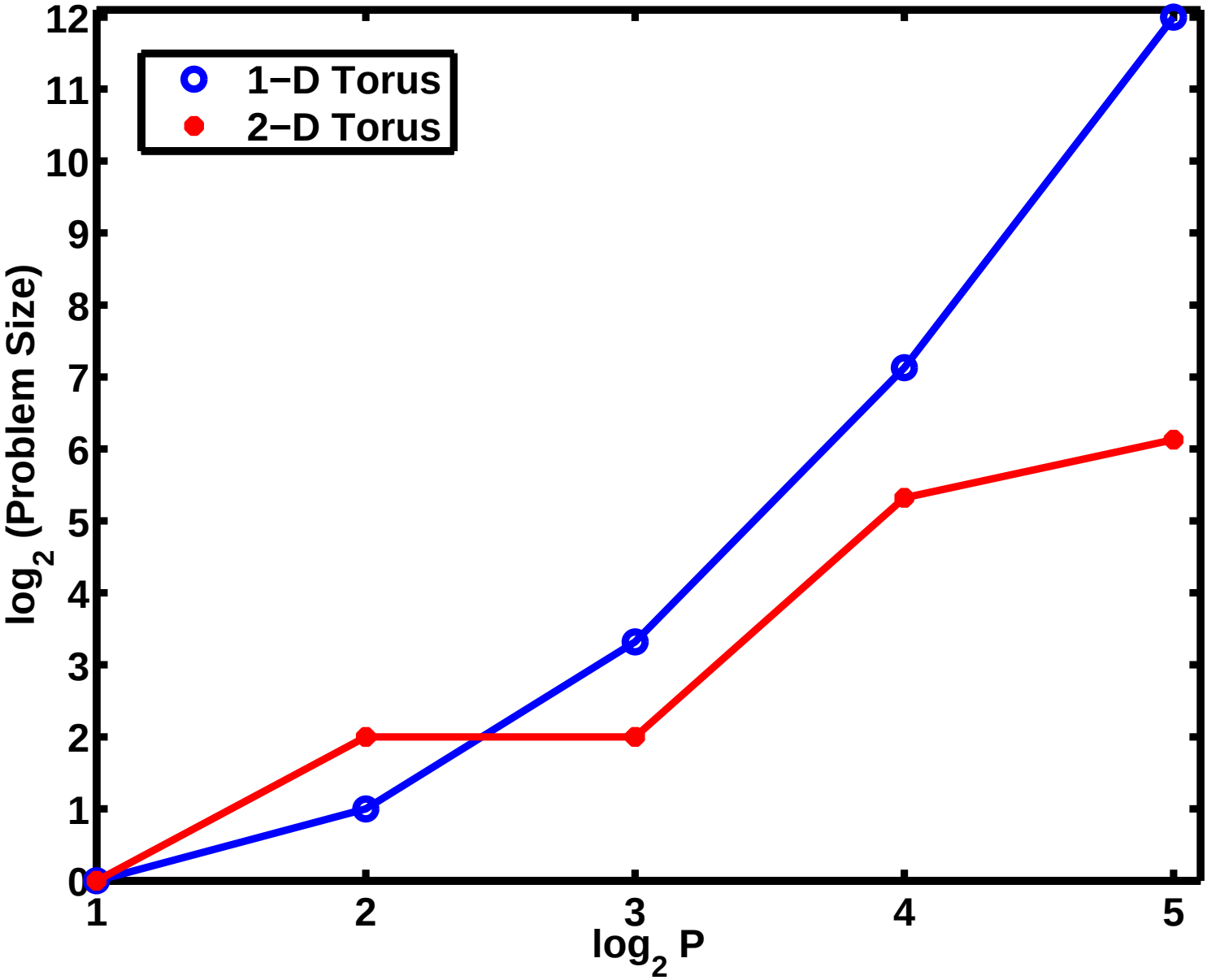
- Problem size was increased stepwisely by assigning multiple times of identical initial intervals to an underlying instance.

- For constant efficiency at 92%, 2-D torus averagely has smaller isoefficiency function, thus more scalable than 1-D torus.

- On four processors, 1-D torus has an advantage with lower communication cost over 2-D torus. However, for a larger number of processors this advantage is overweighted by poor message distribution

Isoefficiency Curve for Equation-Solving Problems

(Lower is better)



Parallel Global Optimization

- A node can also be pruned earlier if the interval extension $\boldsymbol{\Phi}(\boldsymbol{X})$ of the objective $\phi(\boldsymbol{x})$ has a lower bound greater than the current best (least) upper bound. This follows a **branch-and-bound** scheme.

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

- Once a better upper bound is generated, the value is diffusively broadcast to all processors.

- The way an interval box is transfered to and is examined by various processors is determined dynamically in parallel processing.

- The update of best upper bound in sequence and in parallel is performed in different manner, and can be considered as a nondeterministic process.

Speedup Anomaly and Search Overhead

- Due to nondeterministic process of parallel global optimization, speedup might vary greatly from run to run. This is called speedup anomaly.

- Search Overhead (SO) factor [Kumar, 1994]:

$$\text{SO} = \frac{\text{work done by parallel formulation}}{\text{work done by sequential formulation}}$$

- SO > 1 : deceleration anomaly and might result in sublinear speedup
  SO < 1 : acceleration anomaly and might result in superlinear speedup

- Dual-Stack Management:
  - Global random stack and local ordered stack.
  - Concentrate parallel search in acceleration anomaly through proper management of box stacks.

Speedup Analysis - Experiments and Results

- The effect on speedup anomaly was investigated when using single-stack management as well as dual-stack managements over 2-D torus.

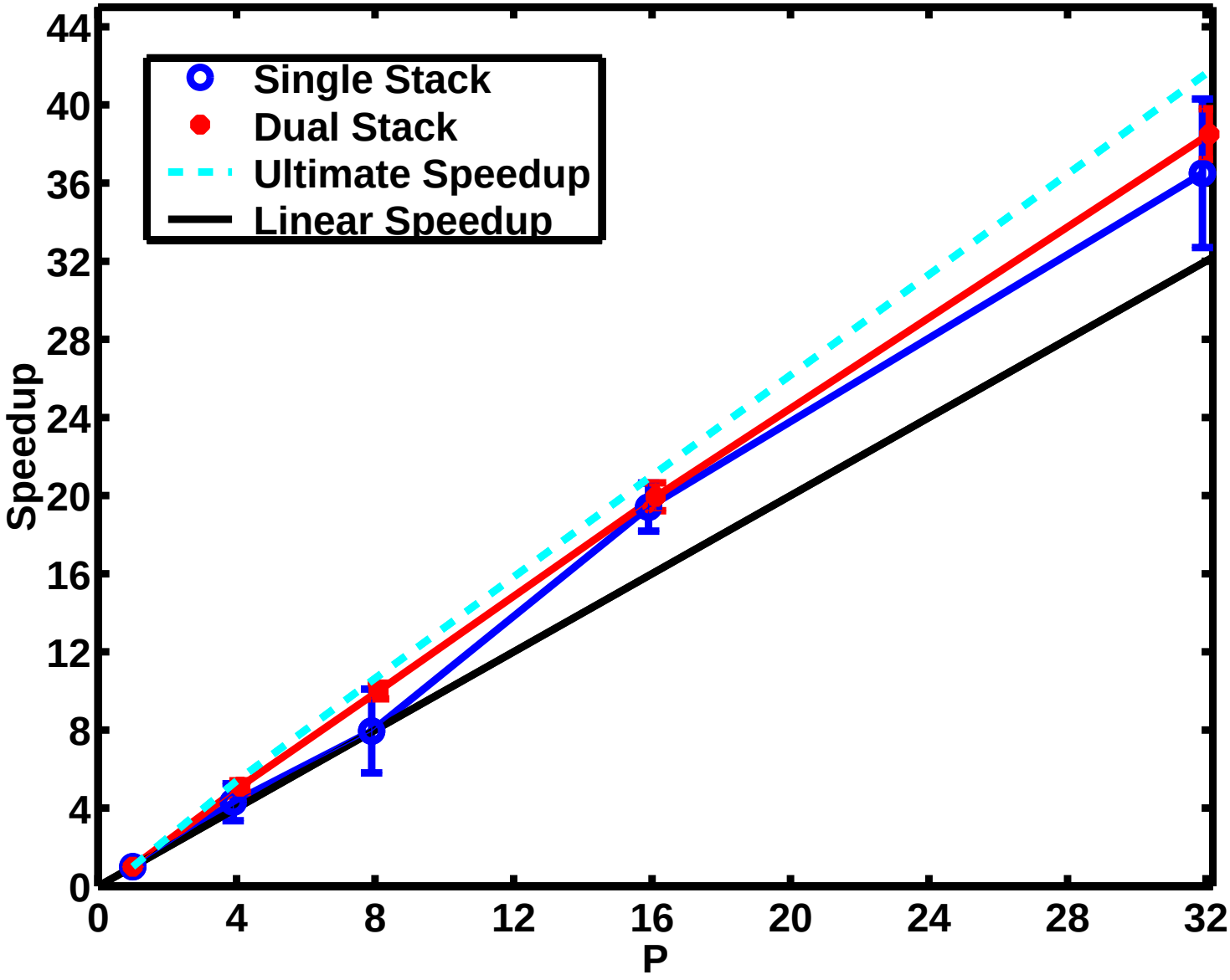
- An optimization type of problem, error-in-variable parameter estimation [Kim,1990; Esposito,1998], was used for testing.

- Ultimate speedups, which provide speedup upper bounds, can be obtained by initially setting best upper bound at globally minimal objective value.

- Experiments show that dual-stack management results in higher as well as more concentrated speedups, and also prevents deceleration anomaly.

- Random process, which finites virtual granuality, indeed alleviates irregularity of tree search on small number processors.

Speedup Anomaly for Optimization Problems



Concluding Remarks

- We have developed a parallel branch-and-prune (and/or bound) scheme embedded in IN/GB for equation-solving and global optimization problems.
  - High efficiency in branch-and-prune scheme
  - Good speedup in branch-and-bound scheme.

- We presented two performance metrics for parallel algorithm: isoefficiency function and speedup anomaly.

- In terms of underlying parallelism, 2-D torus as virtual connectivity has higher scalability.

- In terms of parallel tree search, dual-stack management achieves higher search efficiency on multiprocessor system.

- Parallel algorithms can provide some advantages which are not available by sequential algorithms.

- Acknowledgments
  - ACS PRF 30421-AC9
  - NSF CTS95-22835, DMI96-96110 and EEC97-00537-CRCD
  - EPA R824731-01-0 and R826734-01-0
  - DOE DE-FG07-96ER14691
  - US Army Research Office DAAG55-98-1-0091

- For more information:
  - Contact Prof. Stadtherr at markst@nd.edu
  - Copies of slides will be available next week at http://www.nd.edu/~markst/presentations.html