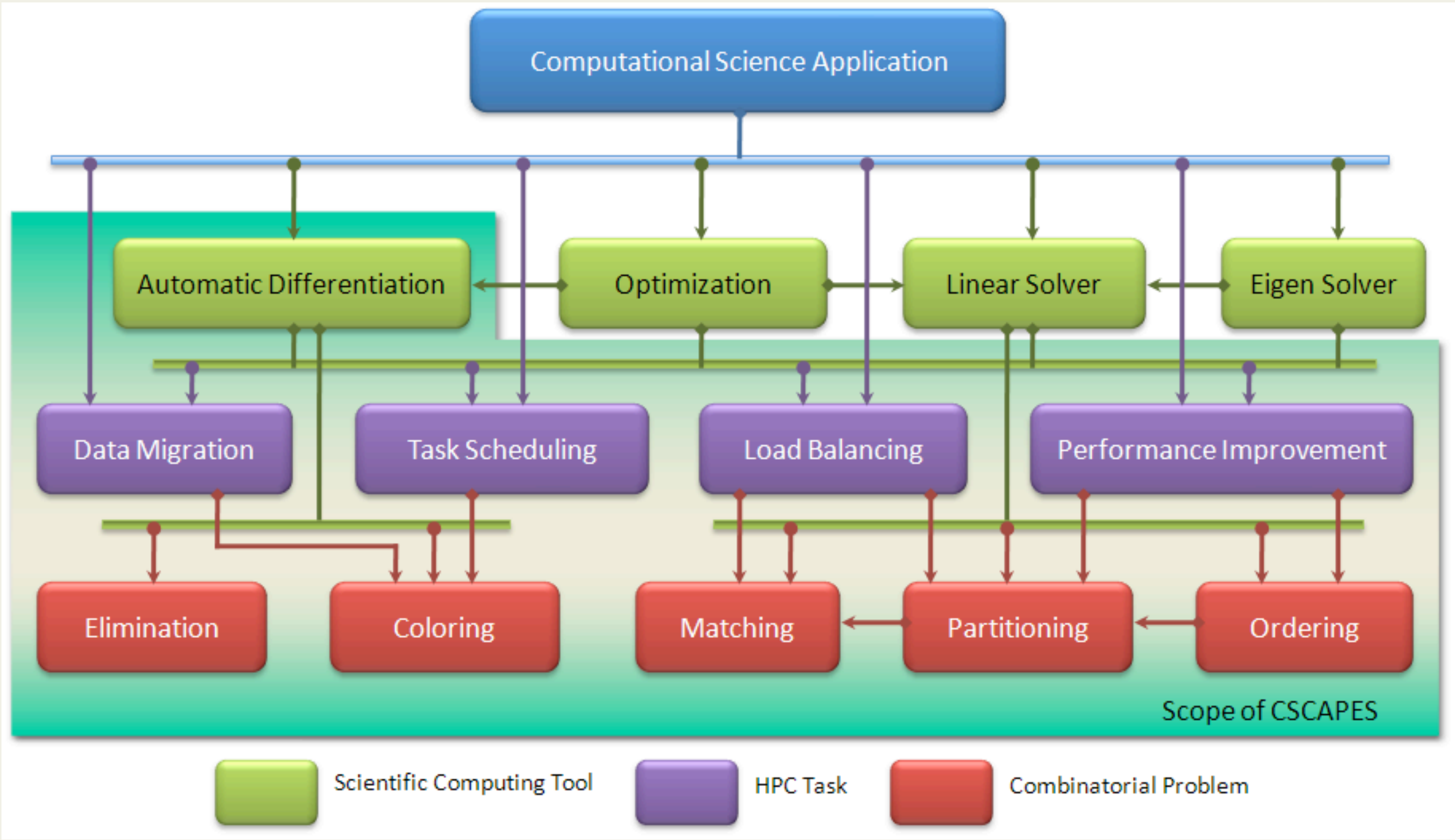
Multithreaded Graph Coloring Algorithms for Scientific Computing on Many-core Architectures

Assefaw Gebremedhin
agebreme@purdue.edu
Purdue University

ICCS Workshop on Manycore and Accelerator-based High-Performance Scientific Computing
Berkeley, January 28, 2011

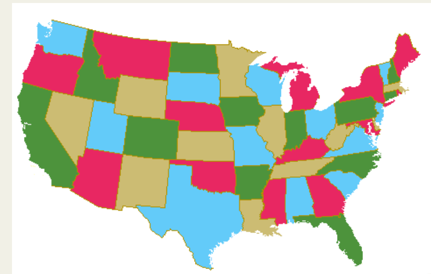
CSCAPES



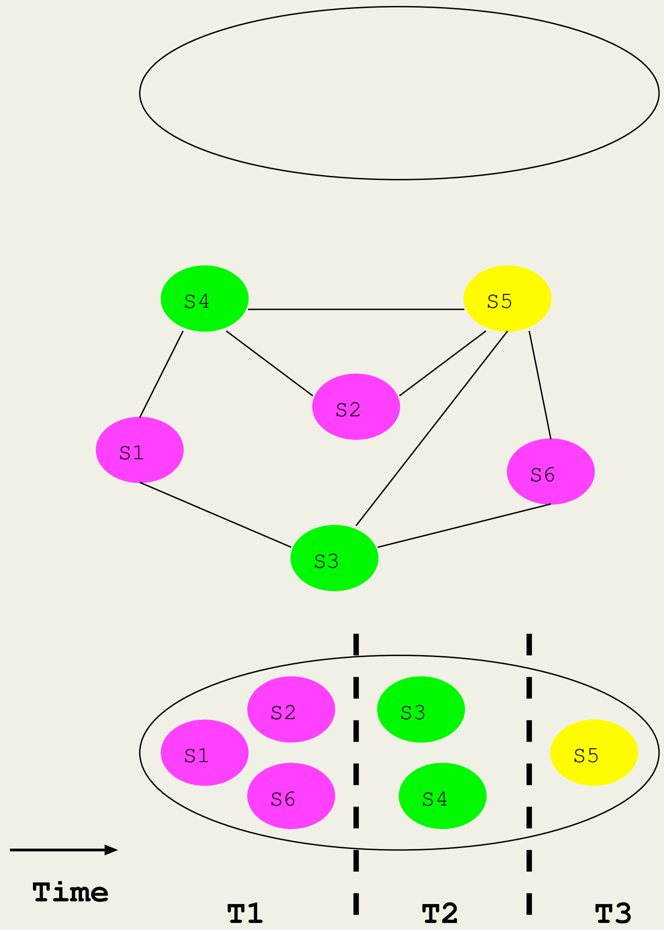
www.cscapes.org

Coloring and its applications

- Graph coloring is an abstraction for partitioning a set of binary-related objects into few “independent sets”
- Coloring contributed to the growth of much of Graph Theory
- Our work on coloring is motivated by its practical applications:
  - Concurrency discovery in parallel (scientific) computing
  - Sparse derivative matrix computation
  - Scheduling
  - Frequency Assignment
  - Facility Location
  - Register Allocation, etc



Graph coloring in concurrency discovery

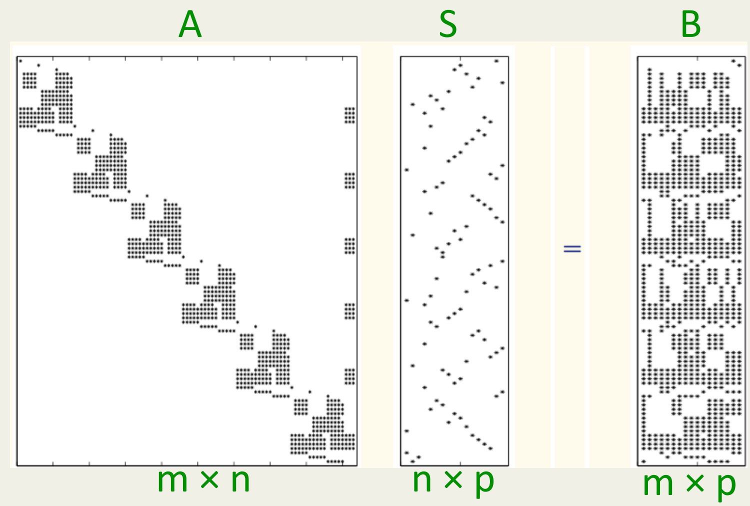


- Adaptive mesh refinement
- Iterative methods for sparse linear systems
- Full sparse tiling

Coloring models in derivative computation: overview

4-step procedure for computing a sparse derivative matrix A using Automatic Differentiation:

- *S1: Determine the sparsity structure of A*
- *S2: Obtain a seed matrix S by coloring the graph of A*
- *S3: Compute a compressed matrix B=AS*
- *S4: Recover entries of A from B*



	Unidirectional partition	Bidirectional partition	
Jacobian	***distance-2 coloring***	***star bicoloring***	**Direct**
Hessian	***star coloring***	**NA**	**Direct**
Jacobian	**NA**	***acyclic bicoloring***	**Substitution**
Hessian	***acyclic coloring***	**NA**	**Substitution**

Distance-2 coloring:
an archetypal model in direct methods

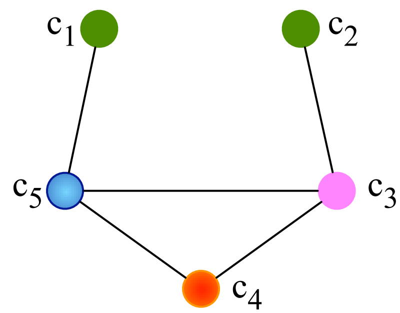
structurally orthogonal partition

distance-2 coloring

symmetric case

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 & a_{15} \\ 0 & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ a_{51} & 0 & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

$\boldsymbol{A}$

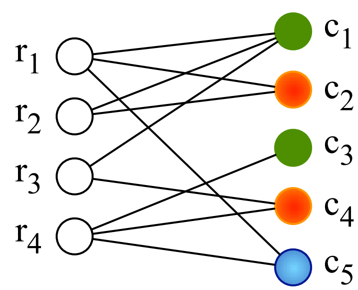


$\mathcal{G}_a$

nonsymmetric case

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & a_{15} \\ a_{21} & a_{22} & 0 & 0 & 0 \\ a_{31} & 0 & 0 & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \end{bmatrix}$$

$\boldsymbol{A}$



$\mathcal{G}_b$

Coloring models in derivative computation revisited

	Unidirectional partition	Bidirectional partition	
Jacobian	**distance-2 coloring** G, Manne and Pothen (05)	**star bicoloring** Coleman and Verma (98) Hossain and Steihaug (98)	**Direct**
Hessian	**star coloring** Coleman and More (84) **restricted star coloring*** Powell and Toint (79)	**NA**	**Direct**
Jacobian	**NA**	**acyclic bicoloring** Coleman and Verma (98)	**Substitution**
Hessian	**acyclic coloring** Coleman and Cai (86) **triangular coloring*** Coleman and More (84)	**NA**	**Substitution**

* Less accurate models

Jacobian: bipartite graph
Hessian: adjacency graph

ColPack

SIAM Review 47(4):629—705, 2005.

www.cscapes.org/coloringpage

An Example Application

Principle of Chromatography

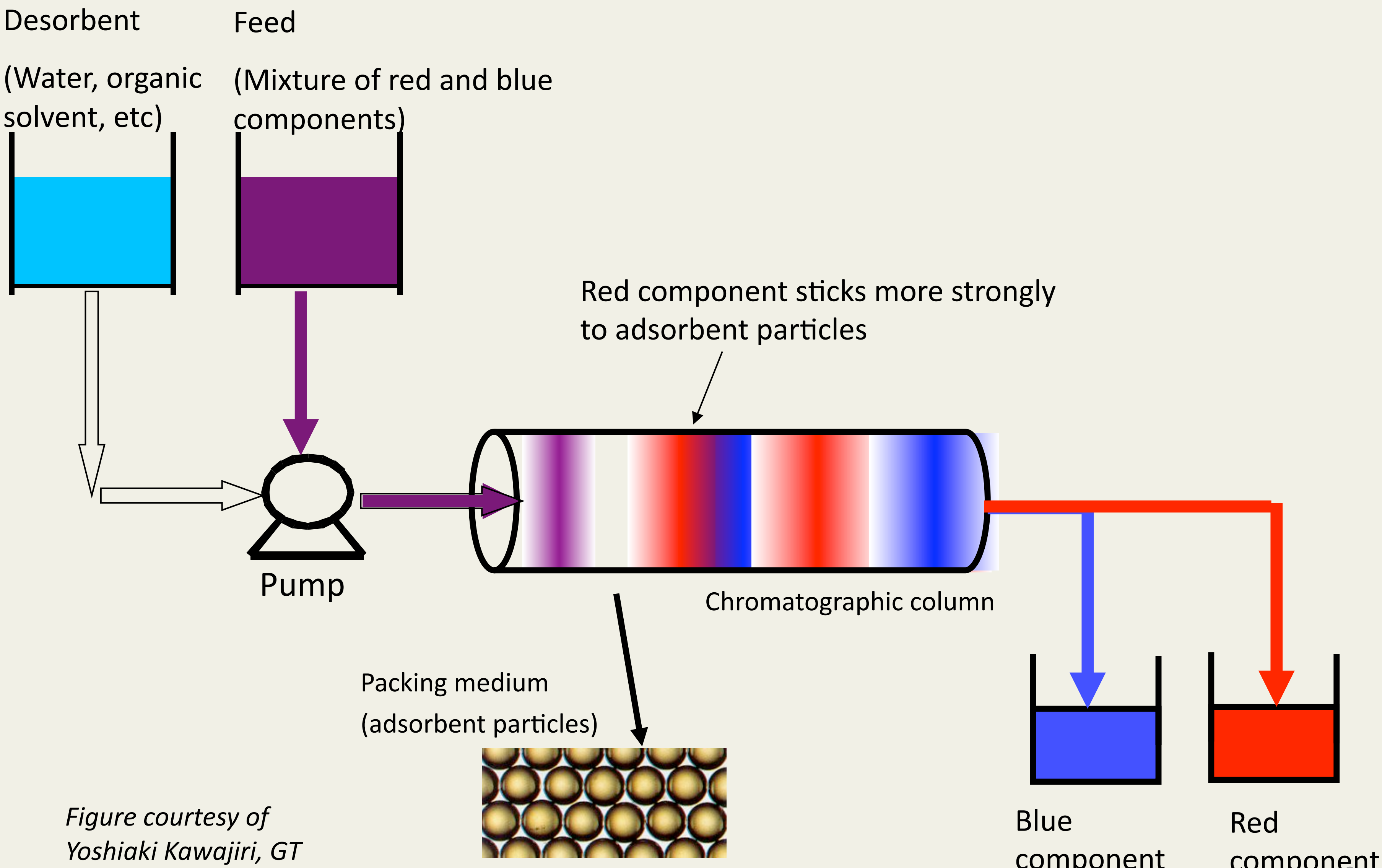


Figure courtesy of Yoshiaki Kawajiri, GT

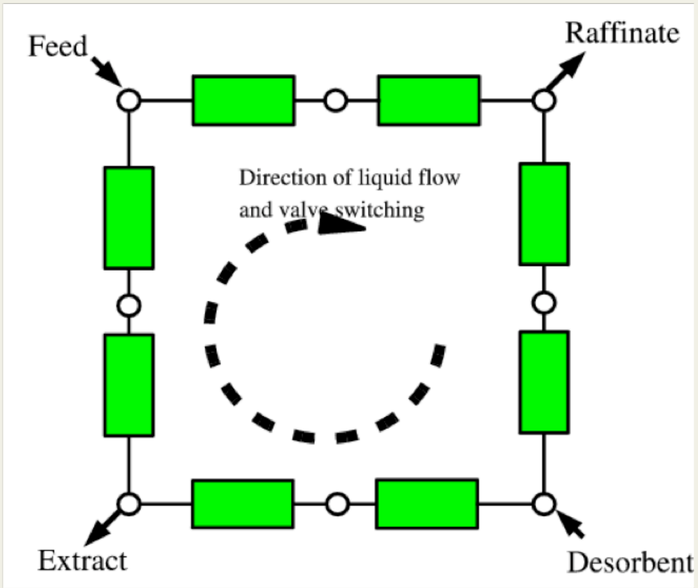
http://www.cwg.hu/english/r-wtcomp.html

Simulated Moving Bed process

- A psuedo counter-current process that mimics operation of TMB
- Reaches only Cyclic Steady State
- Various objectives to be maximized could be identified

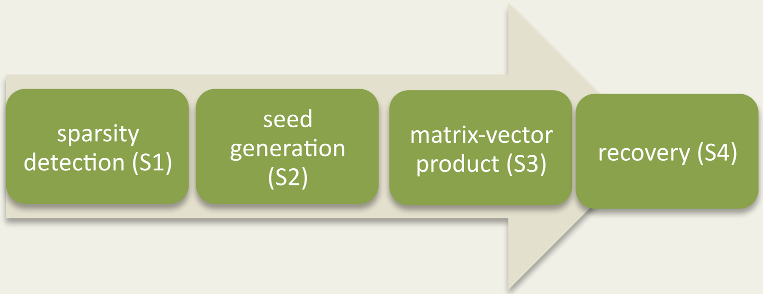
  E.g: product purity, product recovery, desorbent consumption, throughput

- We considered throughput maximization
- Objective modeled as an optimization problem with PDAEs as constraints
- Full discretization was used to solve the PDAEs ➔ sparse Jacobians



Results on Jacobian computation on SMB problem

• Tested efficacy of the 4-step procedure:



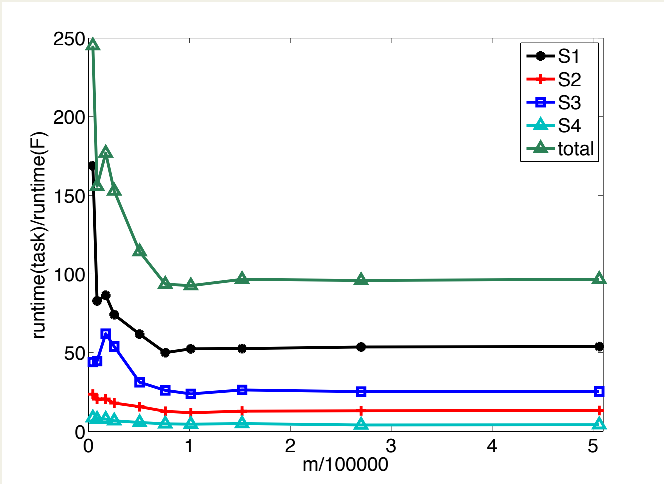
• Used ADOL-C for steps S1and S3, and ColPack for steps S2 and S4

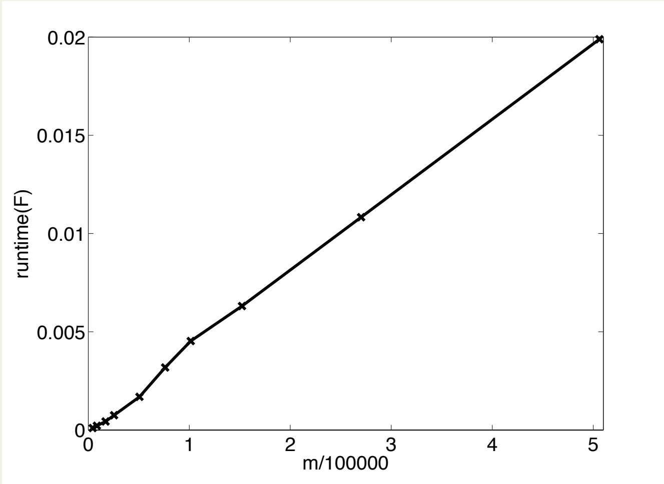
• Observed results for each step matched analytical results

• Techniques enabled huge savings in runtime

Time(Jacobian eval) ≈ 100×Time(function eval)

• Dense computation (without exploiting sparsity) was infeasible





G, Pothen and Walther: AD2008.

Complexity and algorithms

- Distance-k, star, and acyclic coloring are NP-hard (to even approximate)
  - Distance-1 coloring hard to approximate to within $n^{(1-e)}$ for all e>0 [Zuckerman'07]
- A greedy algorithm usually gives good solution

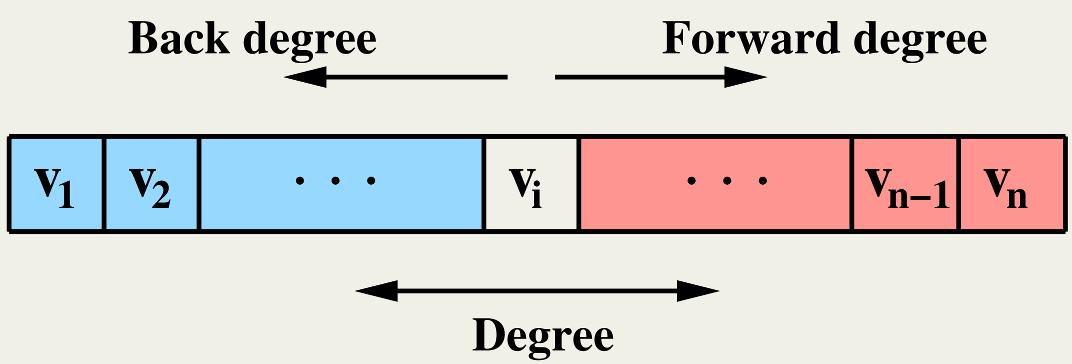
GREEDY(*G=(V,E)*)
    Order the vertices in *V*
    for i = 1 to */V/* **do**
        Determine forbidden colors to v_i
        Assign v_i the smallest permissible color
        [Update collection of induced subgraphs]
    end-for

- ColPack has
  - $O(|V|d_k)$-time algorithms for distance-k coloring (d_k is average degree-k)
  - $O(|V|d_2)$-time algorithms for star and acyclic coloring

Key idea: exploit structure of two-colored induced subgraphs

Ordering techniques in ColPack: fresh formulation

Ordering	Property
Largest First	for $i = 1$ to n: v_i has **largest degree** in $V \setminus \{v_1, v_2, \ldots, v_{i-1}\}$
Incidence Degree	for $i = 1$ to n: v_i has **largest back degree** in $V \setminus \{v_1, v_2, \ldots, v_{i-1}\}$
Dynamic Largest First	for $i = 1$ to n: v_i has **largest forward degree** in $V \setminus \{v_1, v_2, \ldots, v_{i-1}\}$
Smallest Last	for $i = n$ to 1: v_i has **smallest back degree** in $V \setminus \{v_n, v_{n-1}, \ldots, v_{i+1}\}$



Formulation enables:

- *modular imp.*
- *linear time imp.*
- *discovery of use in other contexts*

Parallelization…

Challenges in parallelization in general (on contemporary platforms)

- Parallel Architectural Models?
  - Control mechanism; address space (memory) organization; interconnection network; etc
- Parallel Programming Models?
  - Shared memory; distributed memory; massive threading; etc
- Parallel Computational Models?
  - Wish: realistic yet reasonably simple abstractions

Challenges in parallelizing graph algorithms

- Low available concurrency
- Poor data locality
- Irregular memory access pattern
- Access pattern determined only at runtime
- High data access to computation ratio

Parallel Coloring Algorithms

- Independent-set based (previous approaches)
  - Find maximal independent set in parallel (Luby's algorithm)
  - Limited (or no) success
- Iteration and speculation

Iterative Algorithm (*G*=(*V*,*E*))

Order *V* in parallel

U* = *V

while *U* is not empty

1. Speculatively color vertices in *U* in parallel;

2. Check consistency of colors in *U* in parallel, store conflicts in *R*;

***U* = *R*;**

- Dataflow
  - Fine-grain (edge-level) synchronization; no iteration
  - Feasible when there is HW support for FGS (like the Cray XMT)

Enhancing the Iterative Algorithm

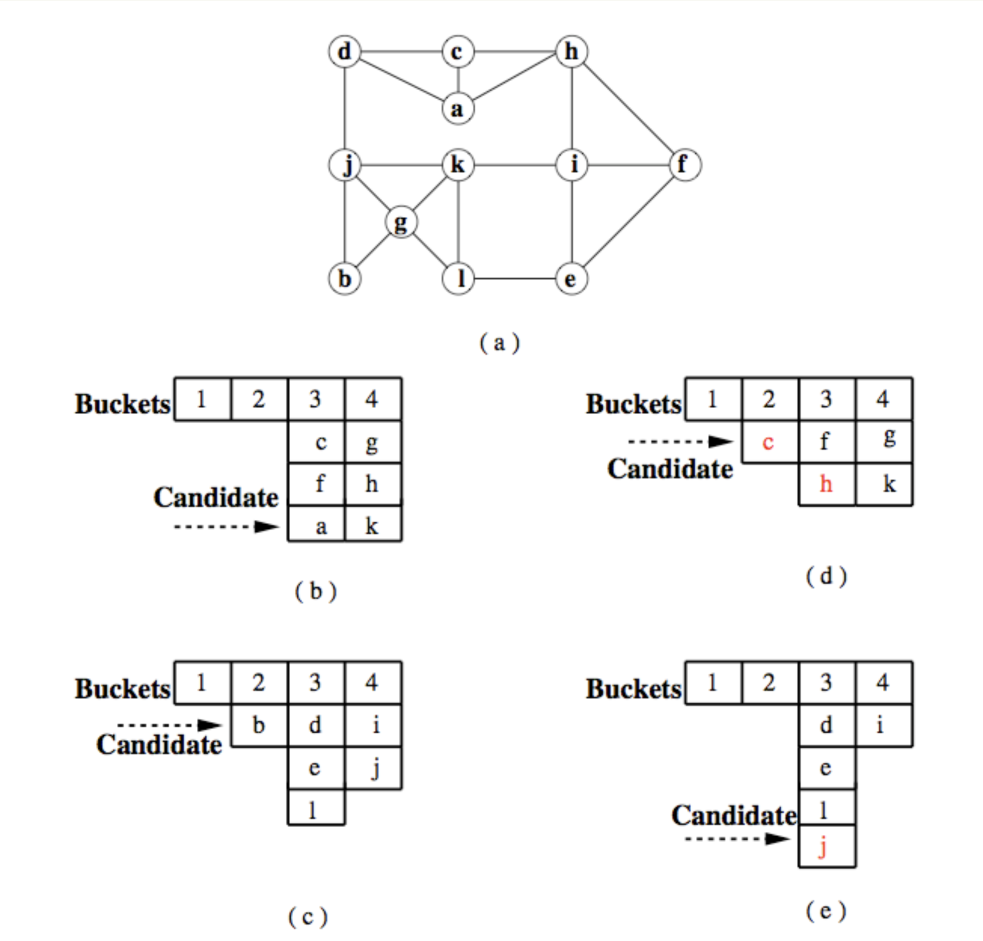
- Color choice
  - First Fit
  - Staggered First Fit
  - Least Used
  - Random
- Resolving a conflict
  - Randomization

Ordering is inherently sequential

Remedy: approximation

Illustration:

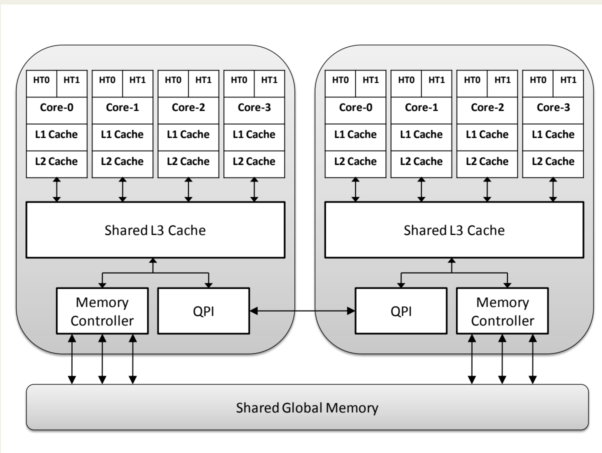
Smallest Last ordering



Experimental Results on Parallel Performance

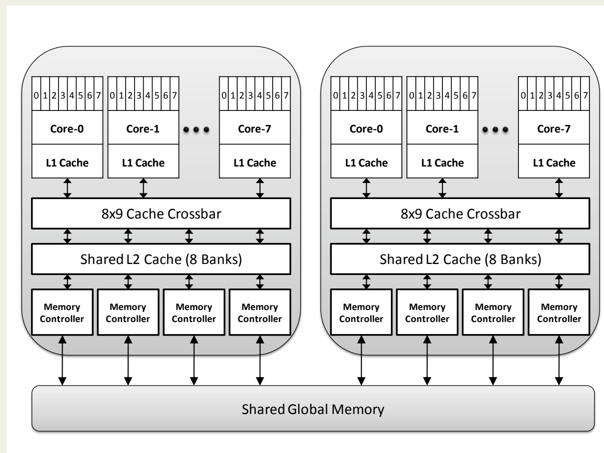
Test platforms

Intel Nehalem



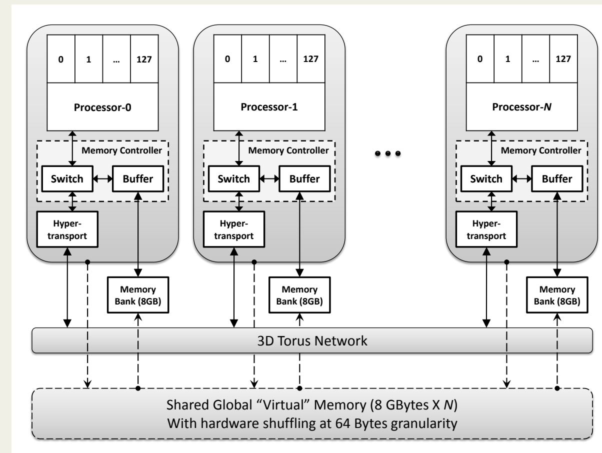
- two quad-core chips
- two hyperthreads per core
- private L1 and L2 cache, shared L3 cache

Sun Niagara 2



- two 8-core sockets
- 8 hardware threads per socket
- L1 cache on core, shared L2 cache

Cray XMT



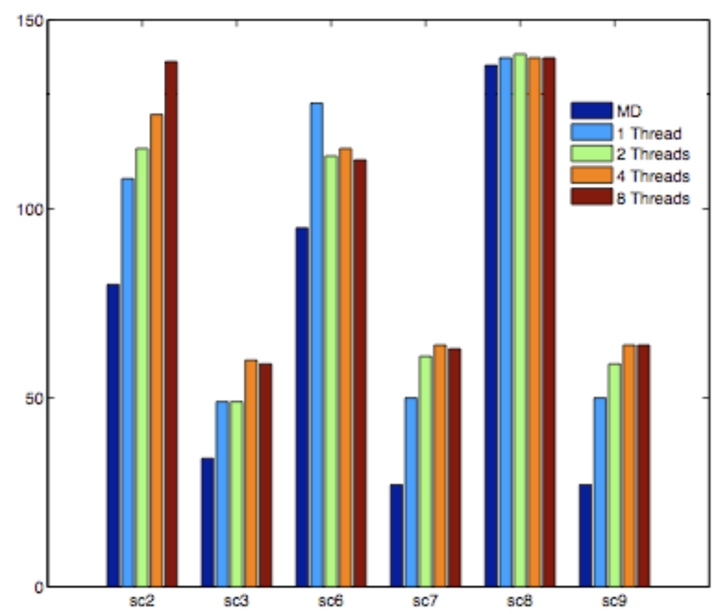
- 128 processors
- 128 hardware thread streams per processor
- cache-less, globally accessible shared memory
- hardware support for fine-grain synchronization

Test graphs

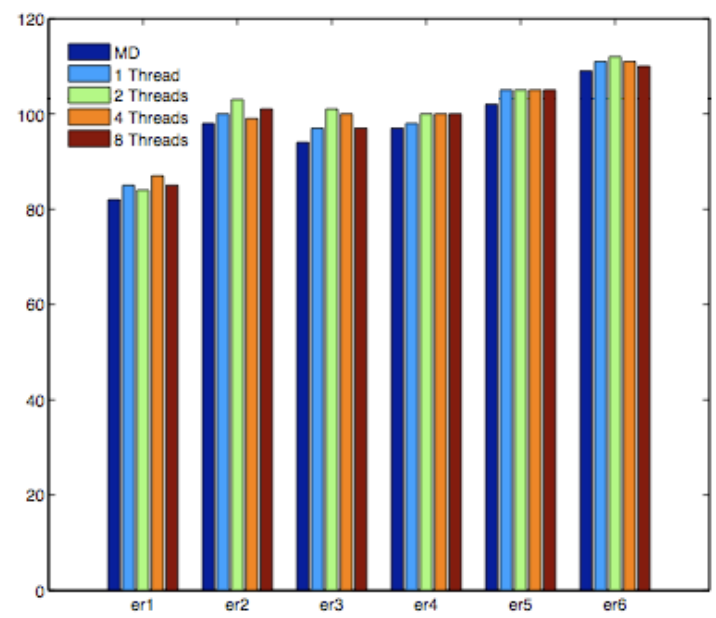
Name	$\|V\|$	$\|E\|$	Δ	Name	$\|V\|$	$\|E\|$	Δ
sc2 (bone010)	986,703	35,339,811	80	g1	131,072	1,046,384	407
sc3 (af_shell10)	1,508,065	25,582,130	34	g2	262,144	2,093,552	558
sc6 (kkt_power)	2,063,494	6,482,320	95	g3	524,288	4,190,376	618
sc7 (nlpkkt120)	3,542,400	46,651,696	27	g4	1,048,576	8,382,821	802
sc8 (er1)	16,777,216	134,217,651	138	g5	2,097,152	16,767,728	1,069
sc9 (nlpkkt160)	8,345,600	110,586,256	27	g6	4,194,304	33,541,979	1,251
er1	131,072	1,048,515	82	b1	131,072	1,032,634	2,980
er2	262,144	2,097,104	98	b2	262,144	2,067,860	4,493
er3	524,288	4,194,254	94	b3	524,288	4,153,043	6,342
er4	1,048,576	8,388,540	97	b4	1,048,576	8,318,004	9,453
er5	2,097,152	16,777,139	102	b5	2,097,152	16,645,183	14,066
er6	4,194,304	33,554,349	109	b6	4,194,304	33,340,584	20,607

sc : graphs from scientific computing apps
er : R-MAT (0.25, 0.25, 0.25, 0.25)
g : R-MAT (0.45, 0.15, 0.15, 0.25)
b : R-MAT (0.55, 0.15, 0.15, 0.15)

Distance-2 coloring: # colors



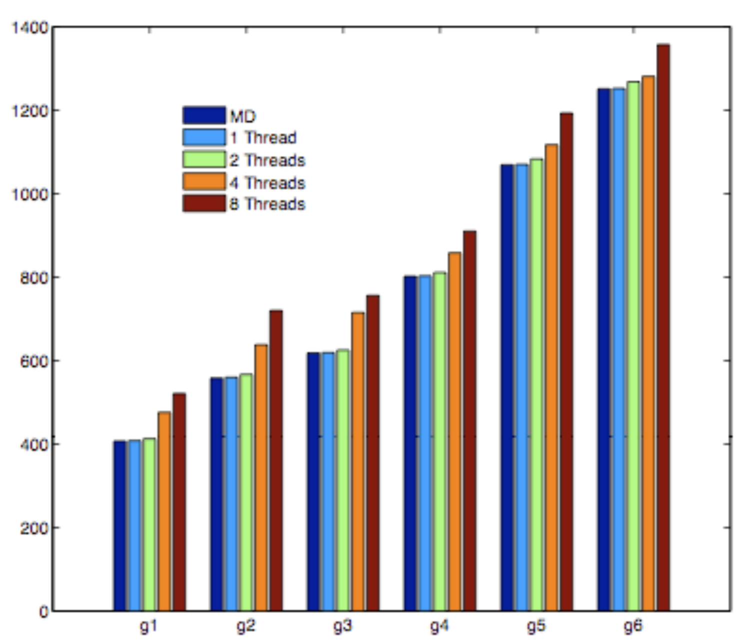
(a) Colors - SLE-FF-SC



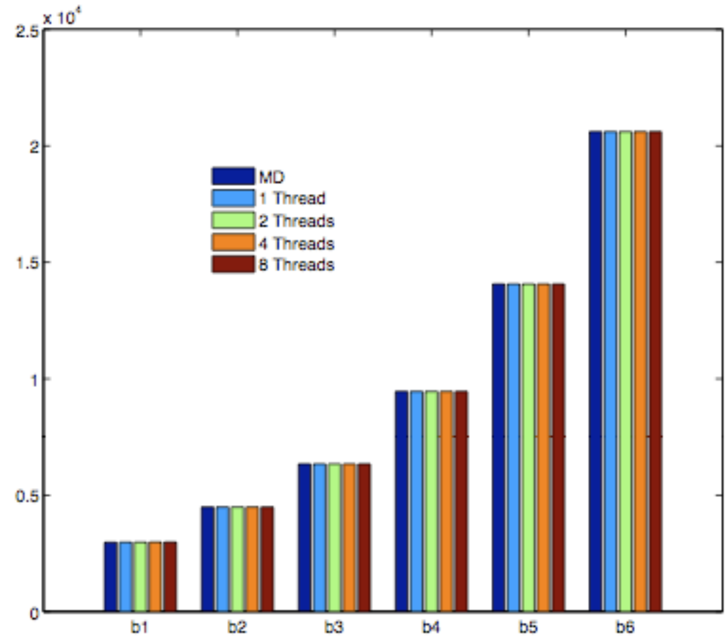
(b) Colors - SLE-FF-ER

Nehalem

Distance-2 coloring: # colors



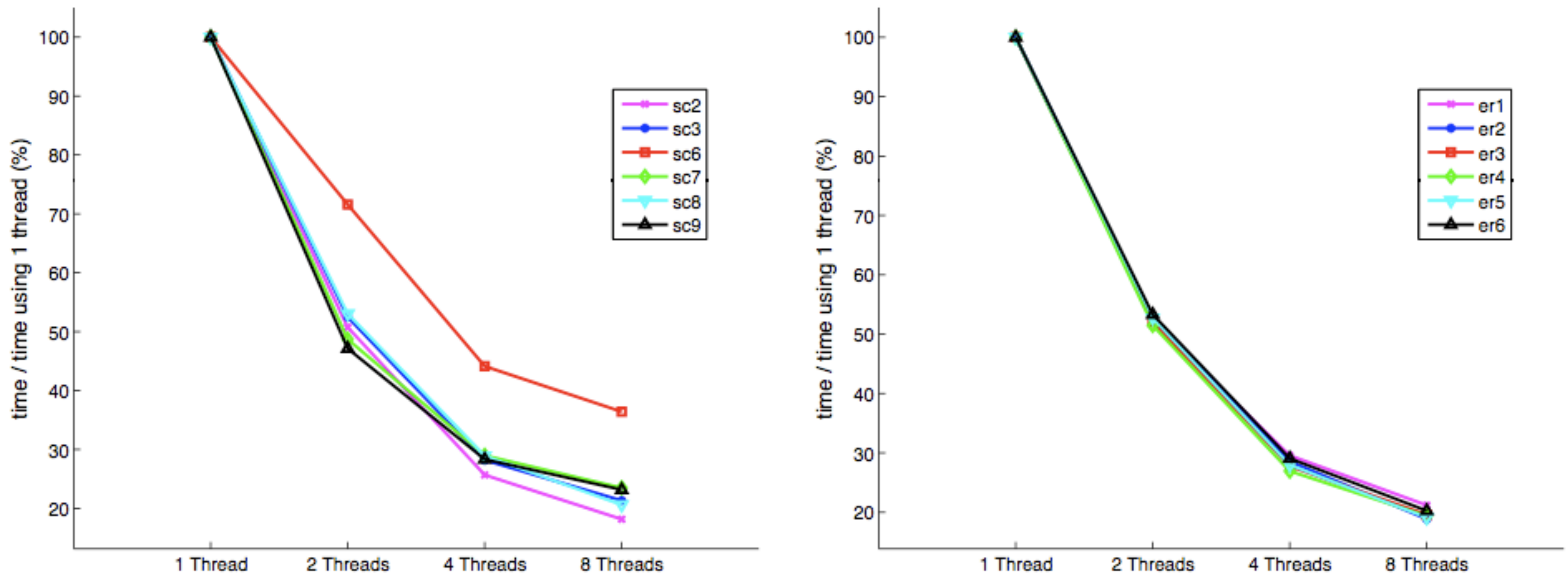
(c) Colors - SLE-FF-G



(d) Colors - SLE-FF-B

Nehalem

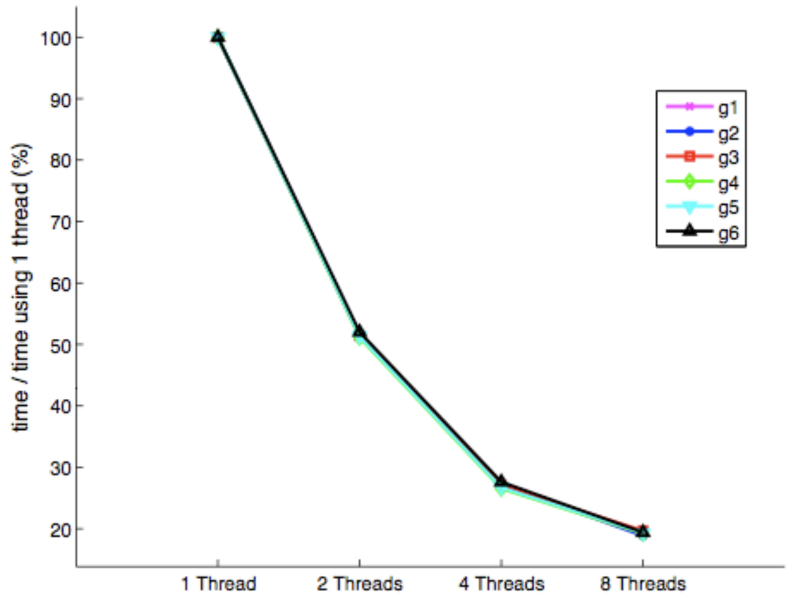
Distance-2 coloring: runtime



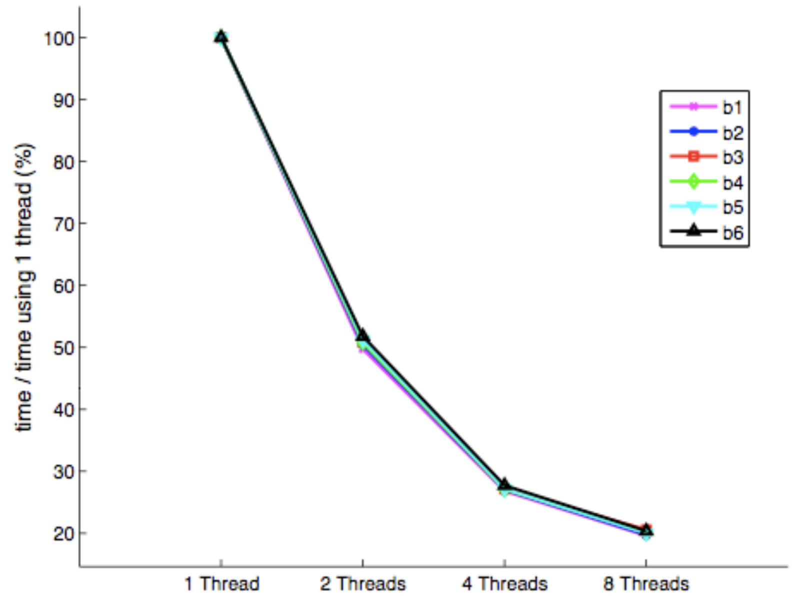
(a) TT - SLE-FF-SC (b) TT - SLE-FF-ER

Nehalem

Distance-2 coloring: runtime



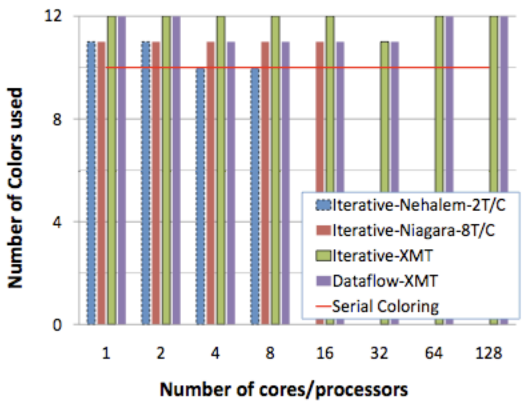
(c) TT - SLE-FF-G



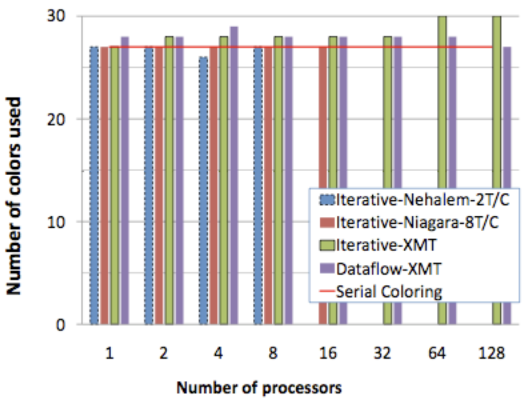
(d) TT - SLE-FF-B

Nehalem

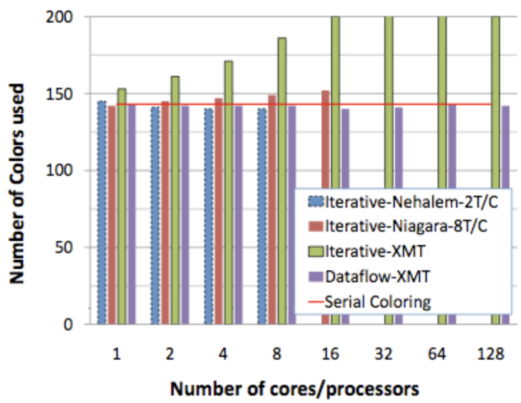
Distance-1 coloring: # colors



(a) RMAT-ER



(b) RMAT-G



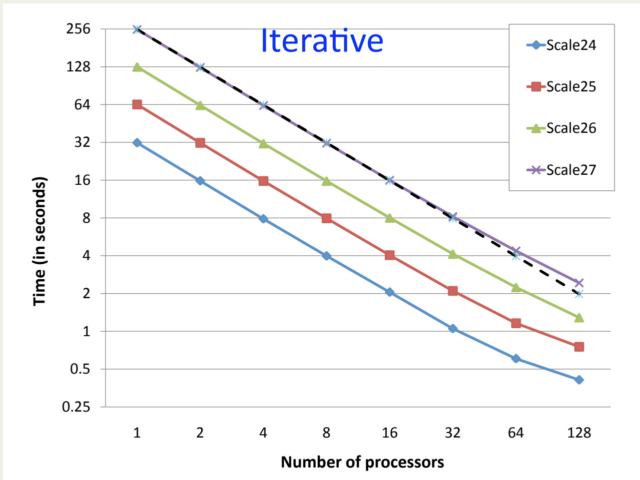
(c) RMAT-B

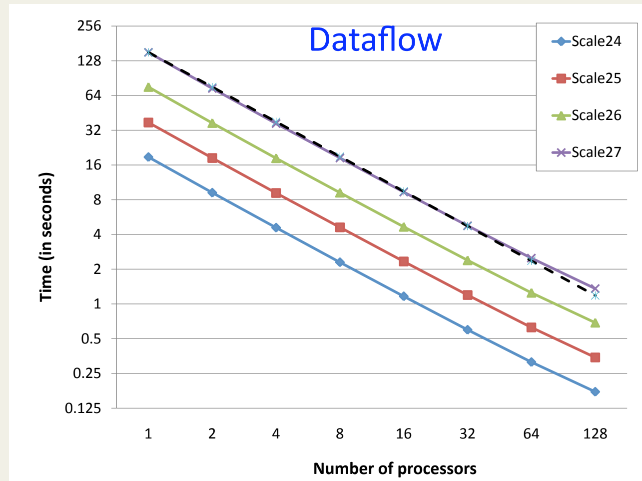
Nehalem, Niagara 2, Cray XMT

Distance-1 coloring : runtime

Small-world graphs with 2^{24}, …, 2^{27} vertices and 134M, …, 1B edges

Cray XMT

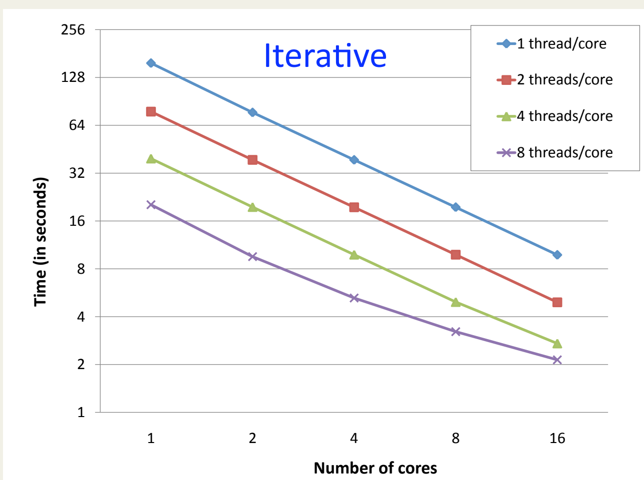


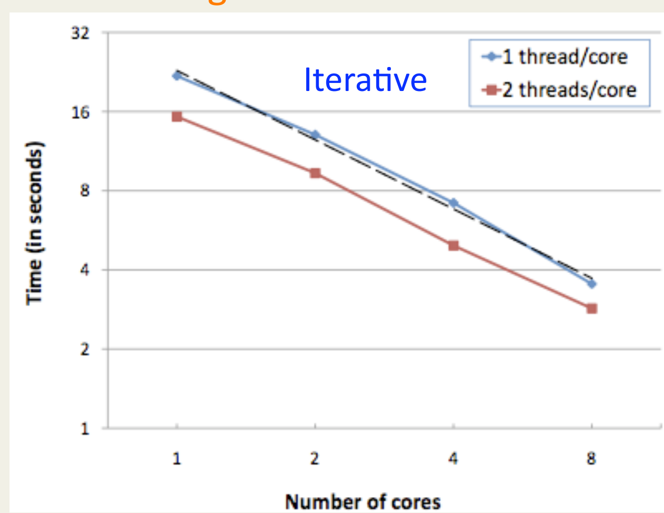


Cray XMT

Small-world graph with 2^{24} = 16M vertices and 134M edges

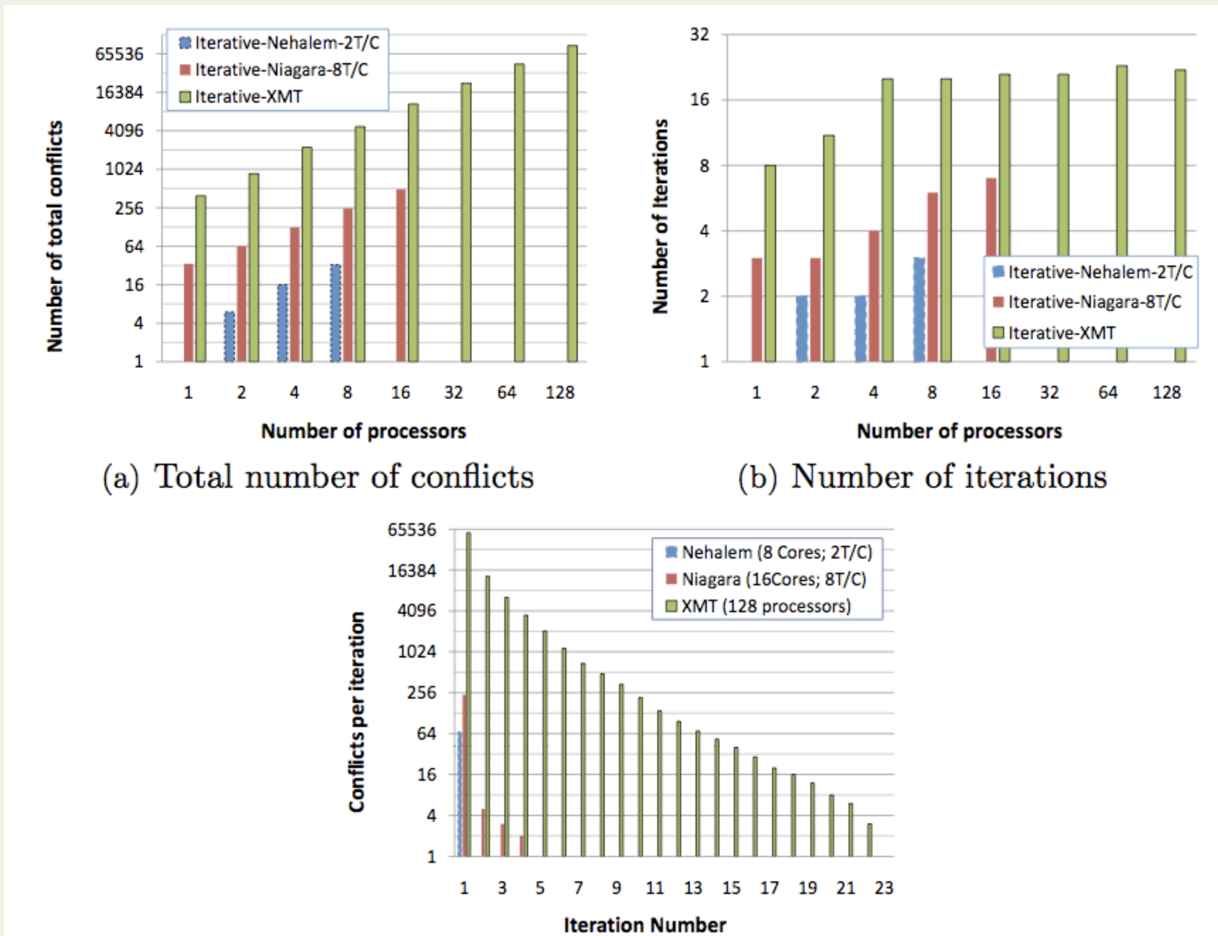
Niagara 2





Nehalem

Iterative: looking inside



(a) Total number of conflicts

(b) Number of iterations

Nehalem, Niagara 2, Cray XMT

A "generic" parallelization technique?

- "Standard" Partitioning
  - Break up the given problem into *p* independent subproblems of almost equal sizes
  - Solve the *p* subproblems concurrently
- "Relaxed" Partitioning
  - Break up the problem into *p*, not necessarily entirely independent, subproblems of almost equal sizes
  - Solve the *p* subproblems concurrently
  - Detect inconsistencies in the solutions concurrently
  - Resolve any inconsistencies

Can be used potentially successfully if the resolution in the fourth step involves only local adjustments

Thanks

- Erik Boman, Doruk Bozdag, Umit Catalyurek, John Feo, Mahantesh Halappanavar, Bruce Hendrickson, Paul Hovland, Fredrik Manne, Duc Nguyen, Mostafa Patwary, Alex Pothen, Arijit Tarafdar, Andrea Walther
- Financial Support: DOE, NSF

Some References

- Gebremedhin, Nguyen, Pothen and Patwary. ColPack: Graph Coloring Software for Derivative Computation and Beyond. *ACM Trans. Math. Software.* Submitted. 2010.
- Gebremedhin, Manne and Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review 47(4):627—705, 2005.*
- Gebremedhin, Tarafdar, Manne and Pothen. New acyclic and star coloring algorithms with applications to computing Hessians. *SIAM J. Sci. Comput. 29:1042—1072, 2007.*
- Gebremedhin, Pothen and Walther. Exploiting sparsity in Jacobian computation via coloring and automatic differentiation: a case study in a Simulated Moving Bed process. *AD2008, LNCSE 64:339---349, 2008.*
- Catalyurek, Feo, Gebremedhin, Halappanavar, Pothen. Multithreaded Algorithms for Graph Coloring. *In submission, 2011.*
- Bozdag, Catalyurek, Gebremedhin, Manne, Boman and Ozguner. Distributed-memory parallel algorithms for distance-2 coloring and related problems in derivative computation. *SIAM J. Sci. Comput. 32(4):2418--2446, 2010.*
- Bozdag, Gebremedhin, Manne, Boman and Catalyurek. A framework for scalable greedy coloring on distributed-memory parallel computers. *J. Parallel Distrib. Comput. 68(4):515—535, 2008.*

- For more information: www.cs.purdue.edu/homes/agebreme