TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing

The Potential of Diffusive Load Balancing at Large Scale

EuroMPI 2016, Edinburgh, 27 September 2016

Matthias Lieber, Kerstin Gößner, Wolfgang E. Nagel
matthias.lieber@tu-dresden.de

ZIH Center for Information Services & High Performance Computing

SPPEXA

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

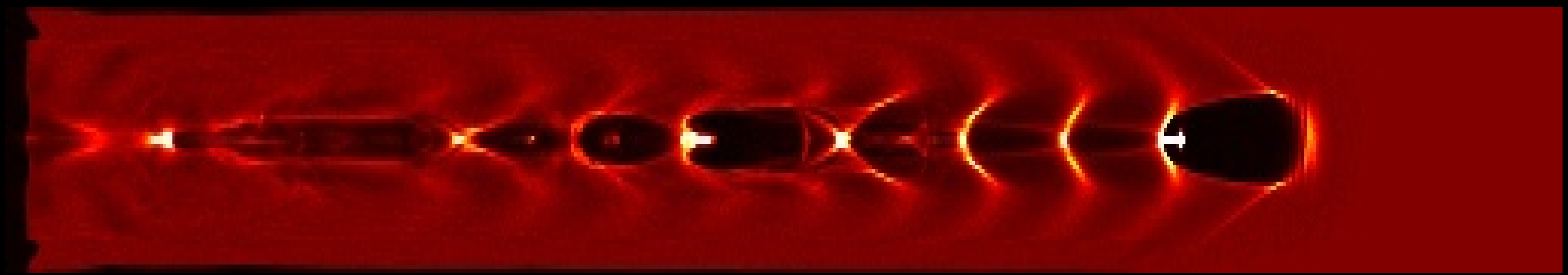
Motivation: Load Balancing

Load Balance

- A challenge for HPC at large scale
- Especially for applications with workload variations

Goals of load balancing

- Repartition application to balance workload
- Reduce comm. costs between partitions (edge cut)
- Reduce task migration costs
- Fast & scalable decision making

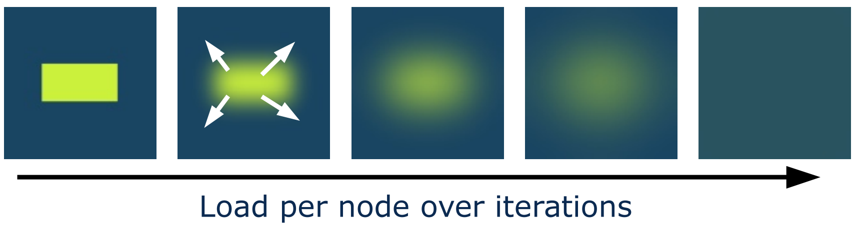


Particle density, laser wakefield acceleration simulation with particle-in-cell code PIConGPU

Motivation: Diffusive Load Balancing

Fully distributed method

- Local operations lead to global convergence



Cybenko,
Dynamic Load Balancing for Distributed Memory Multiprocessors,
J. Parallel Distr. Com. 7(2), 1989.

Practical application is rare

- Well described since the 1990's
- Only few papers show real use in HPC

Watts, Taylor,
IEEE T. Parall. Distr. 9, 1998.

Diekmann, Preis, Schlimbach, Walshaw,
Parallel Computing 26(12), 2000.

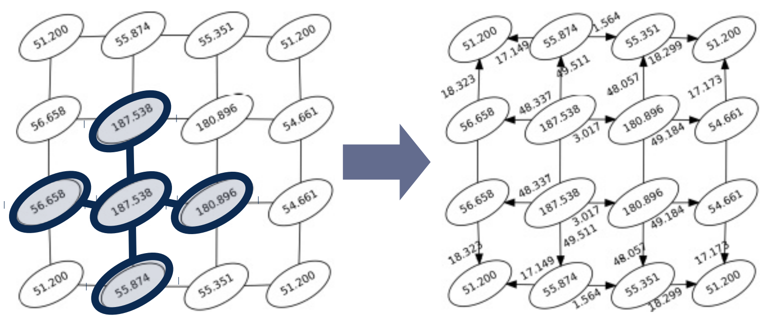
Schloegel, Karypis, Kumar,
SC 2000.

Motivation of this work

- Performance comparison to other state-of-the-art methods at large scale

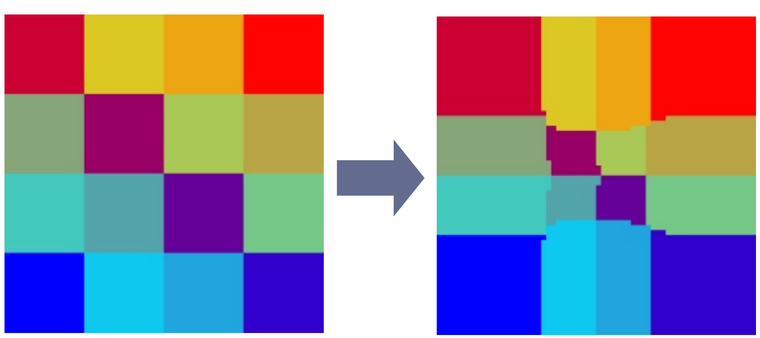
Contents

Short Diffusion Intro



Concept

- Arrange processes/nodes in a graph *G*, e.g. mesh
- Balance virtual load with neighbors for several iterations until global convergence
- Result: minimal* load flow between neighbors in *G* that leads to global balance



How to realize the flows?

- 2nd step required: task selection
- Satisfy flows best possible, keep edge cut and migration low (to reduce communication)

* Most methods minimize sum of squares of individual flows between nodes (two-norm)

Short Diffusion Intro: Algorithms

Original Diffusion Algorithm (Orig Diff)

Cybenko,
J. Parallel Distr. Com. 7(2),
1989.

- In each iteration i each node v updates its load: $l_v^{i+1} = l_v^i + \sum_{w \in N_v} \alpha_{vw} (l_w^i - l_v^i)$

  l_v ...load value of node v

  N_v ...neighbor nodes of node v

  α_{vw} ...diffusion parameter

Estimation:
One iteration should take few 10 µs only

Second Order Diffusion (SO Diff)

Muthukrishnan, Ghosh, Schultz,
Theory Comput. Sys. 31,
1998.

- Prev. iteration‘s transfer influences current

Improved Diffusion (Impr Diff)

Hu, Blake,
Parallel Computing 25(4),
1999.

- Update rule is adapted during iterations based on Laplacian matrix of graph *G*

Dimension Exchange (Dim Exch)

Cybenko, 1989.

Xu, Monien, Lüling, Lau,
Conc. Pract. E. 7, 1995.

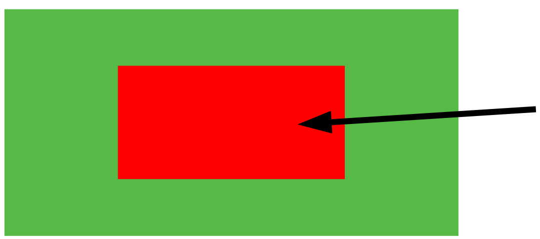
- Local load is updated immediately before exchanging with next neighbor

Contents

Performance Comparison: Diffusion Benchmark

Benchmark setup

- 3D task grid, 3D process mesh, 512 tasks per proc
- Artificial imbalanced workload data*
- Iterations terminate at target imbalance of 0.1%



2D grid example of BOX scenario

Red part is overloaded such that imbalance is 11% (i.e. max load / avg load - 1)

Simplifications

- Time measurement w/o checking termination criterion
- Simple task selection algorithm (single pass)

* in the paper we also use the particle-in-cell application szenario

Performance Comparison: Other Methods

Zoltan load balancing library

- MPI-based library implementations
- RCB: recursive coordinate bisection
- HSFC: Hilbert space-filling curve
- ParMetis graph partitioning via Zoltan

http://www.cs.sandia.gov/Zoltan

Boman, Catalyurek, Chevalier, Devine,
The Zoltan and Isorropia Parallel Toolkits for Combinatorial Scientific Computing: Partitioning, Ordering, and Coloring,
Scientific Programming, 20(2), 2012.

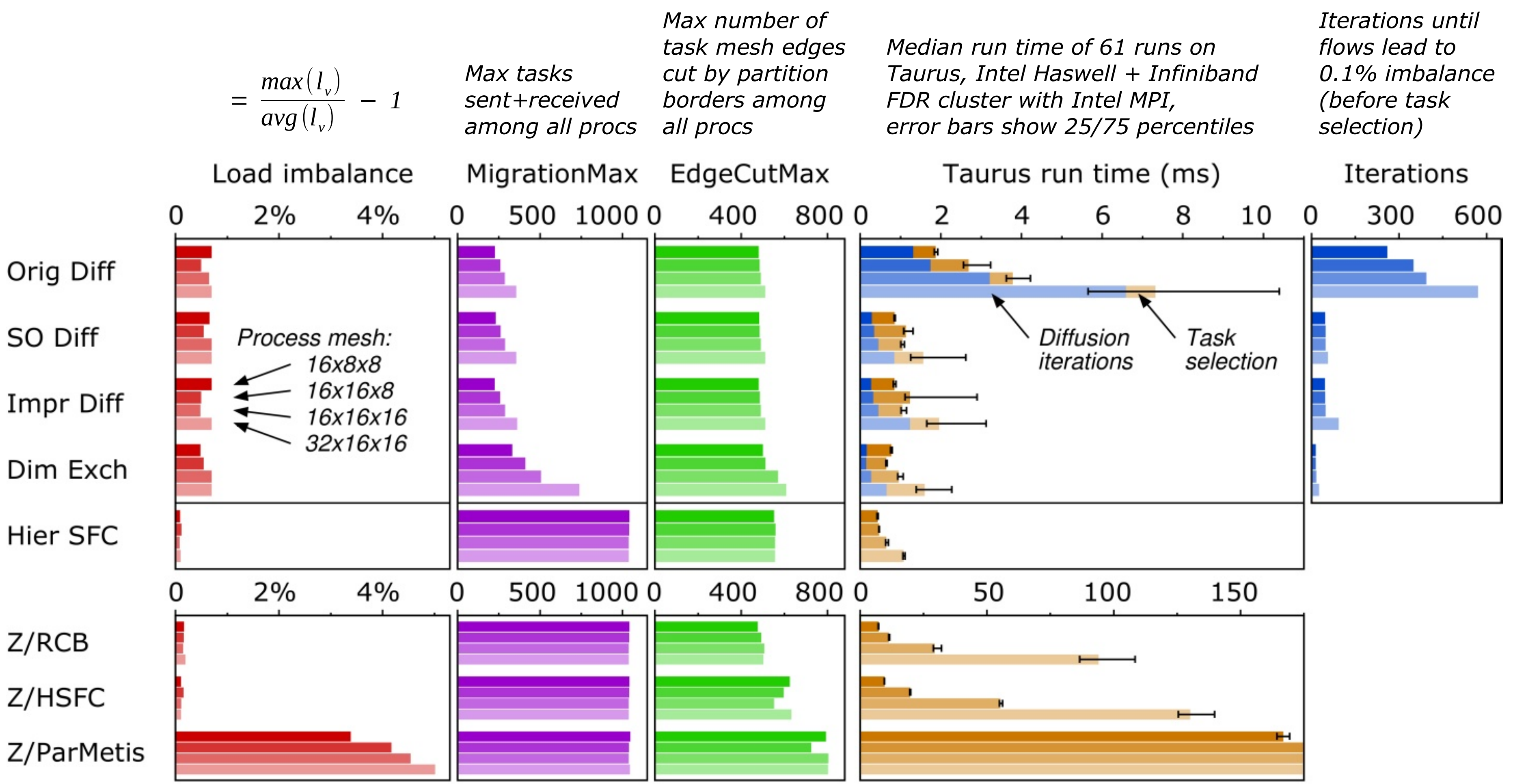
Schloegel, Karypis, Kumar,
A Unified Algorithm for Load-balancing Adaptive Scientific Simulations,
SC 2000.

Hierarchical space-filling curve

- Own fast and scalable method
- Leads to high migration

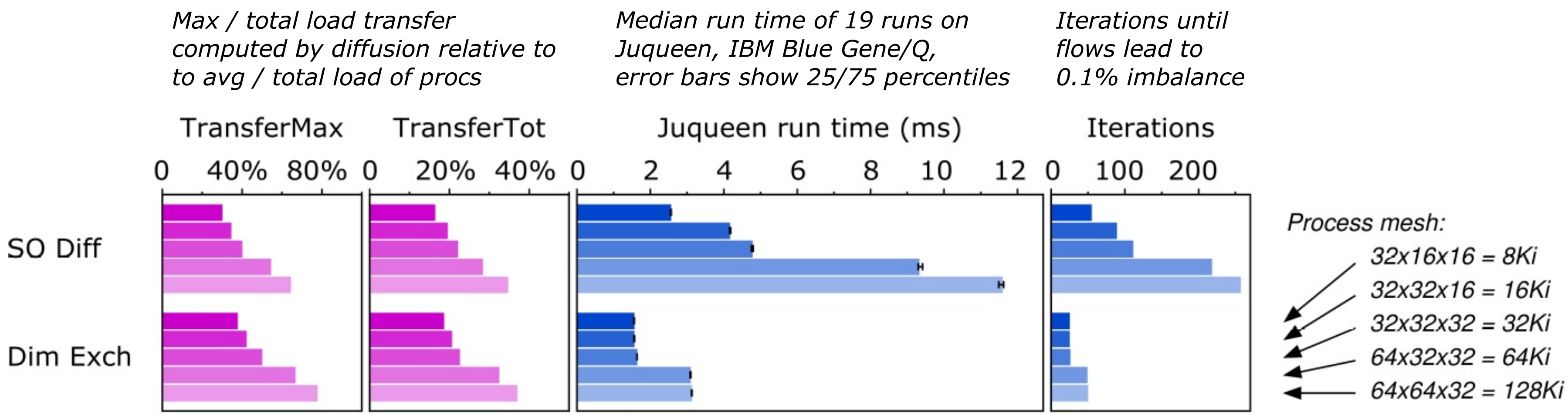
Lieber, Nagel,
Scalable High-Quality 1D Partitioning,
HPCS 2014.

Performance Comparison: 1Ki-8Ki weak scaling



- Diffusion leads to smallest migration
- Diffusion achieves very good edge cut
- Diffusion run time ca. 2 ms for 8192 processes, Zoltan much slower

Performance: 8Ki-128Ki, without task selection



- Dimension exchange scales better than second order diffusion
- Diffusion takes few ms even on 128k processes*

* task selection time does not depend on process count and takes few ms on Juqueen

Summary

Conclusion

Diffusive load balancing is attractive on large scale when overhead (time for decision making, task migration) has to be low, e.g. in case of frequent rebalancing.

Future work

- Improve task selection
- Scalable termination criterion:
  estimate required iterations or check convergence?
- Optimal process graph topology:
  match the hardware or the application?
- Add to Zoltan / Charm++ / application XYZ

Thank you very much for your attention

Acknowledgments / Funding:

VAMPIR

PIConGPU

SPPEXA

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN