Competing with the Best

Using Auto-tuning to Refine the Performance of Chapel

SC13 Chapel Lightning Talks

Ray Chen <rchen@cs.umd.edu>

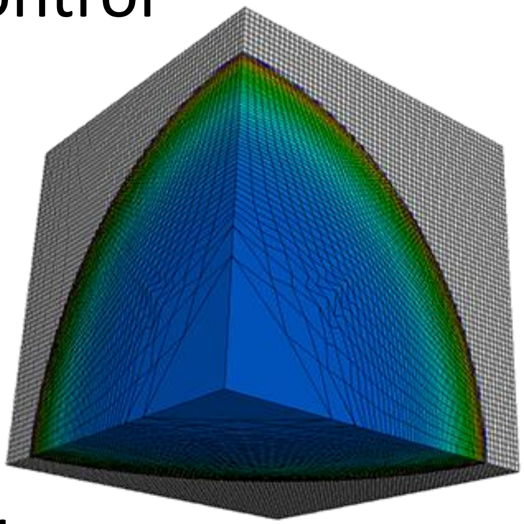
University of Maryland

UMIACS University of Maryland Institute for Advanced Computer Studies

LTS Laboratory for Telecommunications Sciences

Brief Background

- Prior study of HPC languages [1]
  - Compared emerging languages along with mature
  - Used a proxy application as the control
  - Awarded IPDPS 2013 best paper

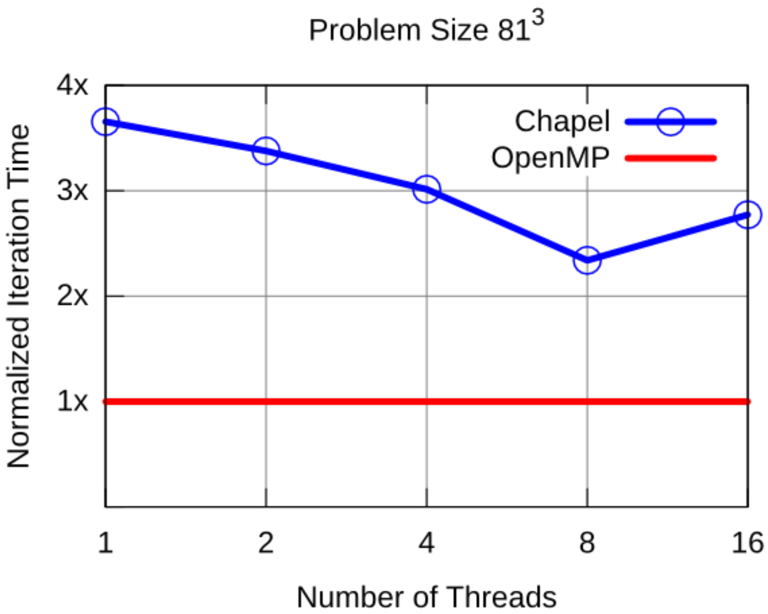


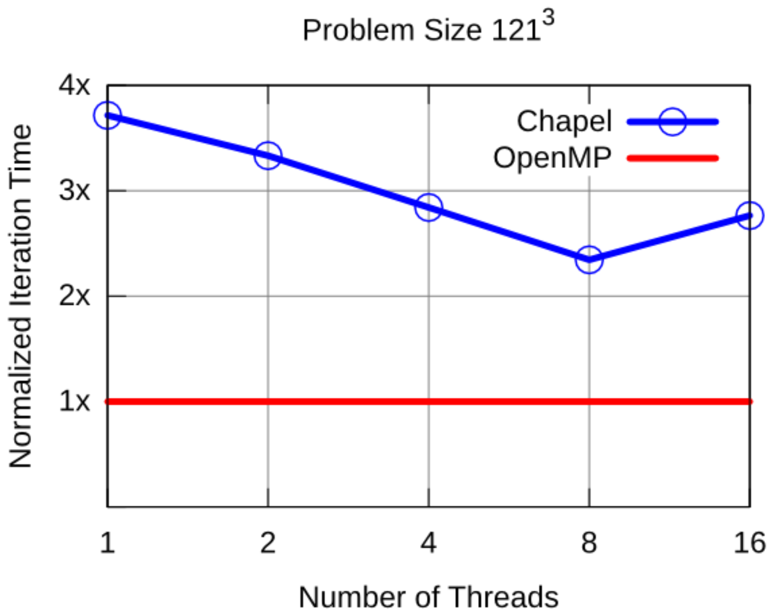
- Proxy Application: LULESH
  - Solves a Sedov blast problem
  - Typical of HPC hydrodynamics codes
  - Indirection arrays to create an unstructured mesh

[1] Karlin, A. Bhatele, J. Keasler, B. L. Chamberlain, J. Cohen, Z DeVito, R. Haque, D. Laney, E. Luke, F. Wang, D. Richards, M. Schulz, C. H. Still, Exploring Traditional and Emerging Parallel Programming Models using a Proxy Application. In Proceedings of 27th IEEE International Parallel & Distributed Processing Symposium, Boston, MA, pages 1-14

Chapel vs. OpenMP

- Chapel wins for programmer productivity
  - 1108 SLOC vs. 2403 for OpenMP
- OpenMP still better for run-time performance





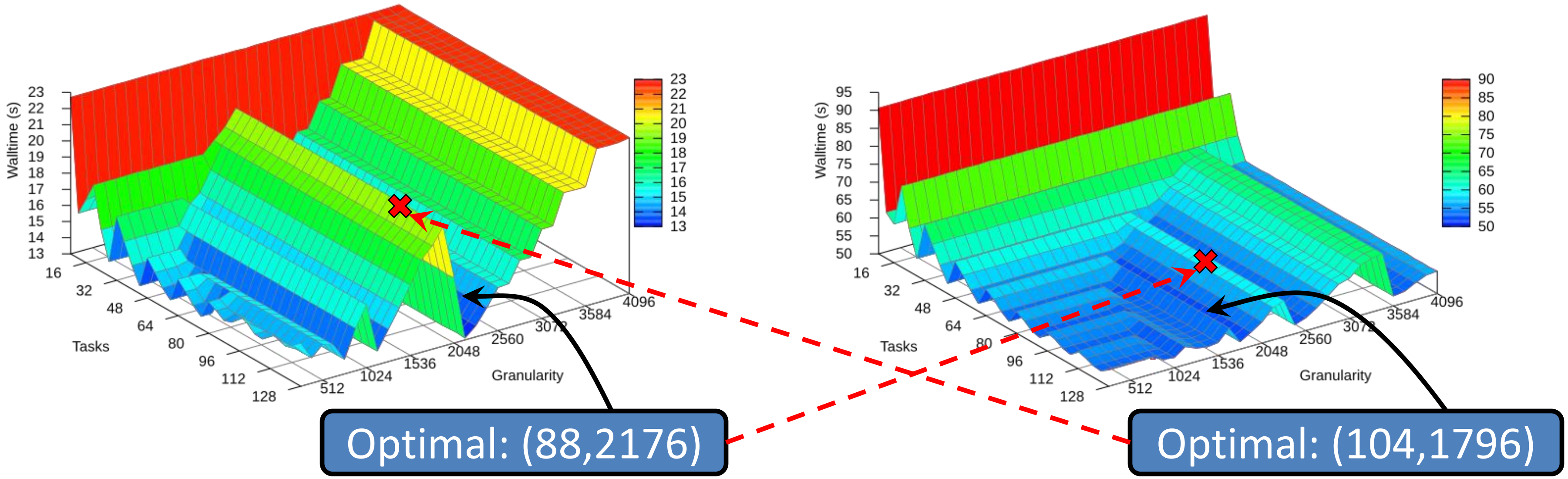
Controlling Parallelism

- Kernel vs. user-space threads
  - User-space threads dominate for Chapel's LULESH
    - Kernel-space threads always slower in our tests
  - Optimal thread count difficult to predict
- User-accessible knobs built into Chapel
  - Task count per data parallel loop
  - Data decomposition granularity

Input Parameter Sensitivity

- Exhaustive parameter sweep for two data sets

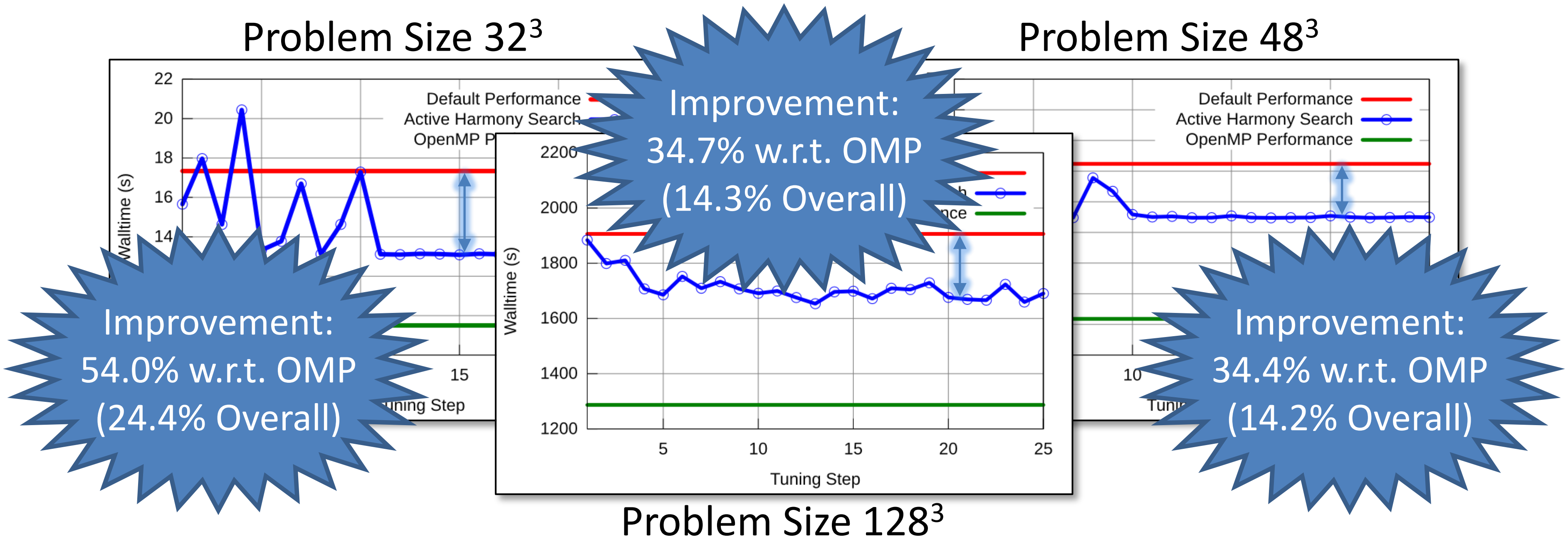
Problem Size 32^3 Problem Size 48^3



- Optimal points are not exchangeable
  - Results in 20% or 80% slowdown

Auto-tuning Results

- Search converges after 10 search steps
- Performance gap narrowed 34-54%
  - Overall performance improvement 13-24%



Conclusion

- Chapel can be within 29% of OpenMP
  - All from auto-tuning (no source code changes)
  - Improves upon 2-4x slowdowns of previous study
- On the horizon
  - Managing tasks among concurrent parallel loops
    - Complicated, if not impossible to do statically
    - Even worse for nested parallel loops
  - Auto-tuning as a solution
    - Dynamic problems call for dynamic solutions