SCALLOP: A Highly Scalable Parallel Poisson Solver in Three Dimensions

Gregory T. Balls, Scott B. Baden
Dept. of Computer Science and Engineering, UCSD

Phillip Colella
Applied Numerical Algorithms Group, LBNL

http://www.cs.ucsd.edu/∼gballs/scallop/

Motivation

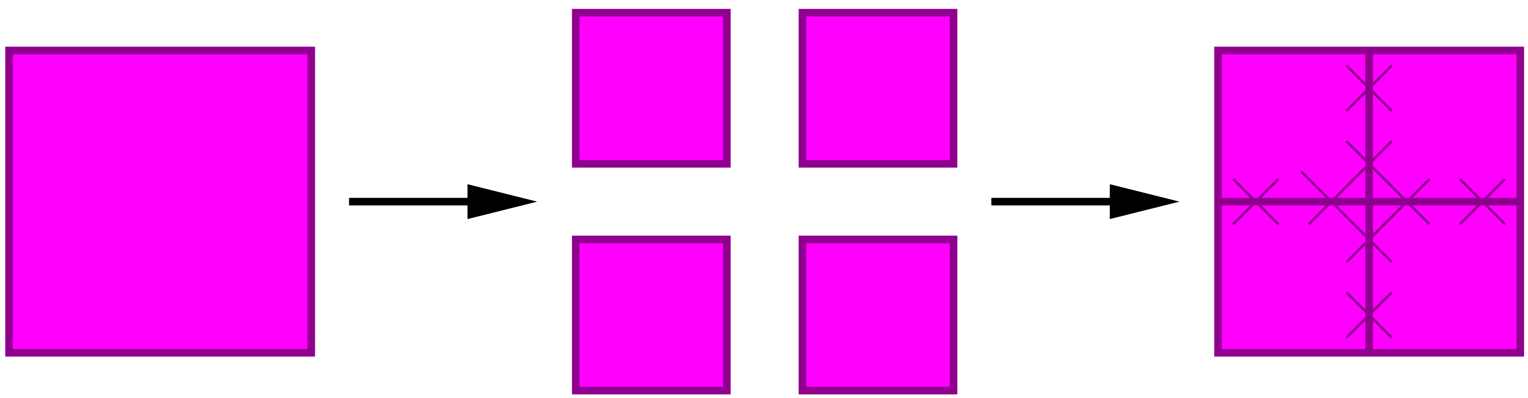
- Parallel solvers suffer from scalability problems due to communication overheads.
  - technological trends are increasing the relative cost of communication
- Elliptic regularity provides an opportunity to reduce communication costs significantly.
  - Barnes-Hut (1986), MLC (1986), FMM (1987), Bank-Holst (2000)
- Our contribution: extension of these ideas to finite difference problems.

Infinite Domain Poisson Equation

- These boundary conditions can be used for
  - modeling the human heart [Yelick, Peskin, and McQueen]
  - astrophysics
- Often these BCs are a better match than
  - extending the domain
  - using periodic boundary conditions
- Computing these boundary conditions is expensive, esp. on a parallel computer

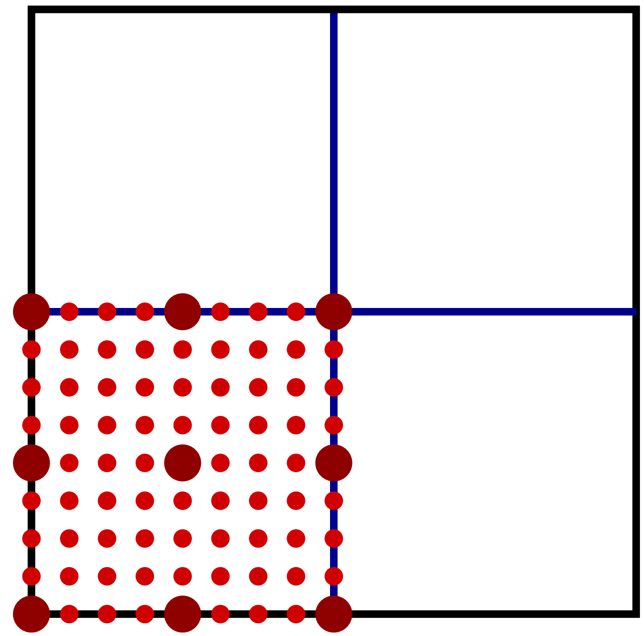
Domain Decomposition

- Divide problem into subdomains.
- Use a reduced description of far-field effects.
- Stitch solutions together.



Domain Decomposition Definitions

- Global problem of size N^3.
- Divide into q^3 subdomains.
- Corresponding coarse mesh of size $(\frac{N}{C})^3$, where C is the *coarsening factor*.
- In this 2-D slice, $N = 16$, $q = 2$, and $C = 4$.

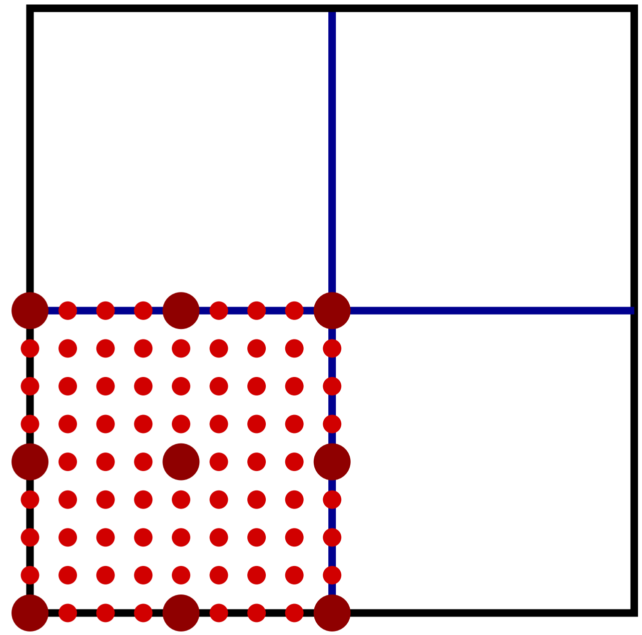


The Scallop Domain Decomposition Algorithm

- Five step algorithm, 2 communication steps

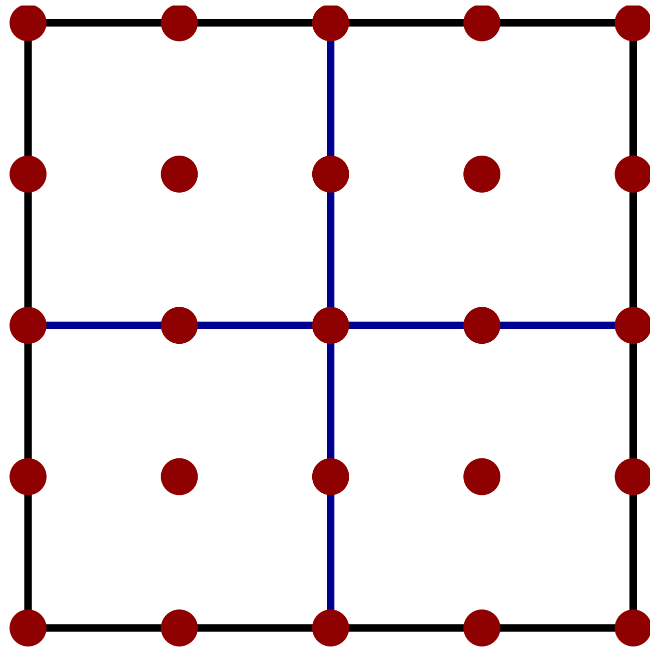
- Serial building blocks
  - Dirichlet solver:
    - Multigrid
    - FFT
  - Infinite domain solver (built on a Dirichlet solver):
    - two Dirichlet solves
    - infinite domain boundary calculation

Domain Decomposition Algorithm



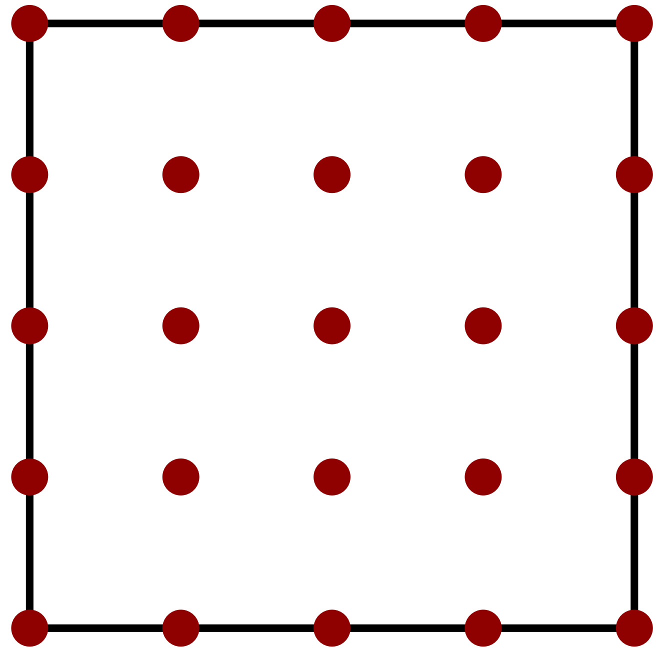
1 On each subdomain, solve an infinite domain problem, ignoring all other subdomains, and create a coarse representation of the charge.
[$O(N^3)$ work, no communication]

Domain Decomposition Algorithm



2 Aggregate all the coarse charge fields into one global charge.
[all-to-all communication]

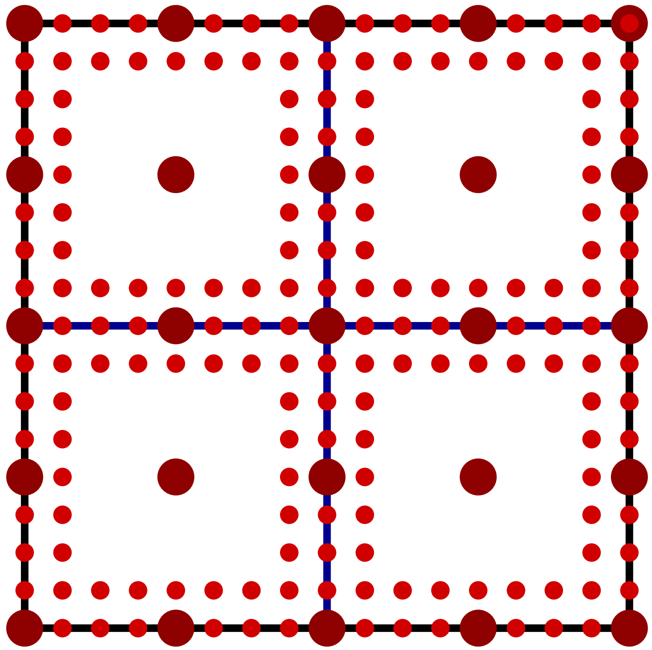
Domain Decomposition Algorithm



3 Calculate the global infinite domain solution.

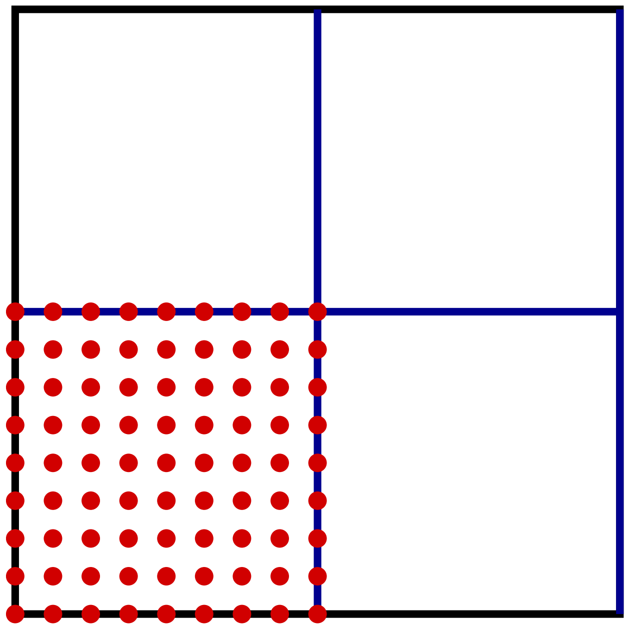
[$O(N^3)$ work, no communication]

Domain Decomposition Algorithm



4 Exchange boundary data between neighbors, and combine global coarse-grid and local fine-grids to get BCs.
[nearest-neighbor communication]

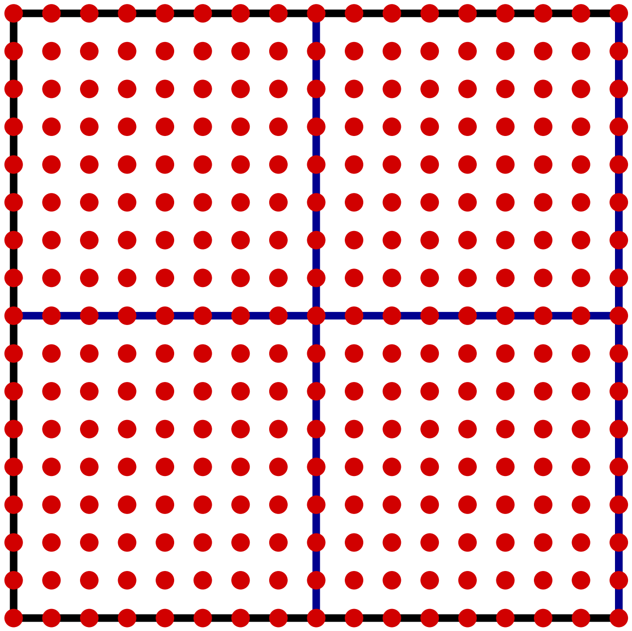
Domain Decomposition Algorithm



5 Compute final Dirichlet calculation on each subdomain using these BCs.
[$O(N^3)$ work, no communication]

Domain Decomposition Algorithm

1. Initial solution
2. Aggregation
3. Global coarse solution
4. Local correction
5. Final calculation



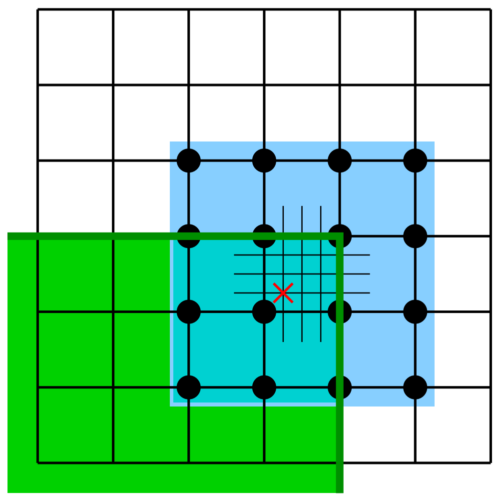
Overall: $O(N^3)$ work, two communication steps.

Subdomain Overlap

- In order to ensure smooth solutions and accurate interpolation, the local domains need to overlap.

- The overlap is measured in coarse grid spacing.

- For large refinement ratios, the overlap (in terms of fine grid points) gets very large.



It should be possible to generate coarse points without calculating the fine-grid boundary region.

Computational Overhead

There are three sources of computational overhead:

1. Global coarse-grid calculation on a grid of size $(\frac{N}{C})^3$:
   - small if $\frac{N}{C} << \frac{N}{q}$
2. Extra computation due to overlap of the fine-grid domains:
   - small if C is reasonably small
3. Two fine-grid calculations (complete solutions, not just smoothing steps or V-cycles):
   - unavoidable, but the final Dirichlet solution is less costly than a full infinite domain solution

The KeLP Programming System

- Scallop is built with KeLP, a C++ class library
- KeLP simplifies the expression of coarse to fine grid communication
  - bookkeeping
  - domains of dependence
- KeLP provides useful abstractions
  - expression of communication in geometric terms
  - set operations on geometric domains
  - bookkeeping between fine and coarse meshes

Experiments

- Ran on two SP systems with Power 3 CPUs:
  - NPACI's Blue Horizon
  - NERSC's Seaborg
- Ran with two different serial solvers:
  - Multigrid
  - FFT (using FFTW)
- Compiled with `-O2`, standard environment.

Scaled Speed-up

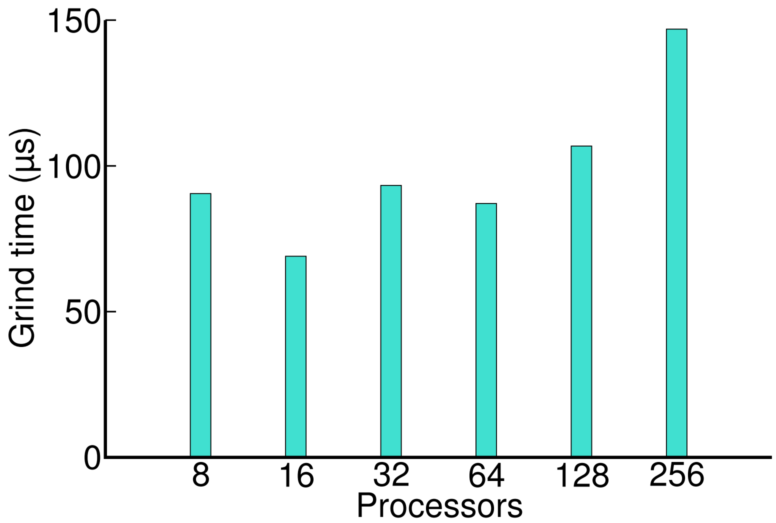
- try to keep work per processor constant
- $N^3 \propto q^3 \propto C^3 \propto P$
- We report performance in terms of grind time
  - ideally should be constant

Parameters – Multigrid / Blue Horizon

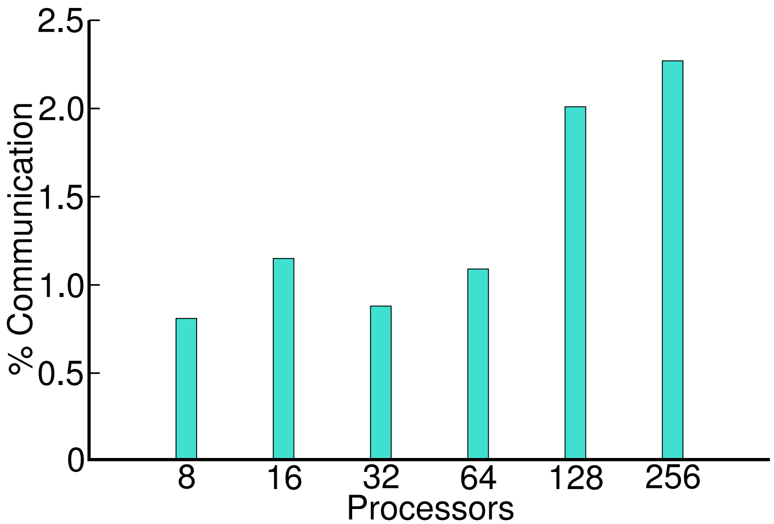
P	q	C	N
8	2	3	192^3
16	4	4	256^3
32	4	5	320^3
64	4	6	384^3
128	8	8	512^3
256	8	10	640^3

- Relatively small problem size per processor
- Significant overlap in 256-processor case

Results – Multigrid / Blue Horizon



Grind times



Communication percent

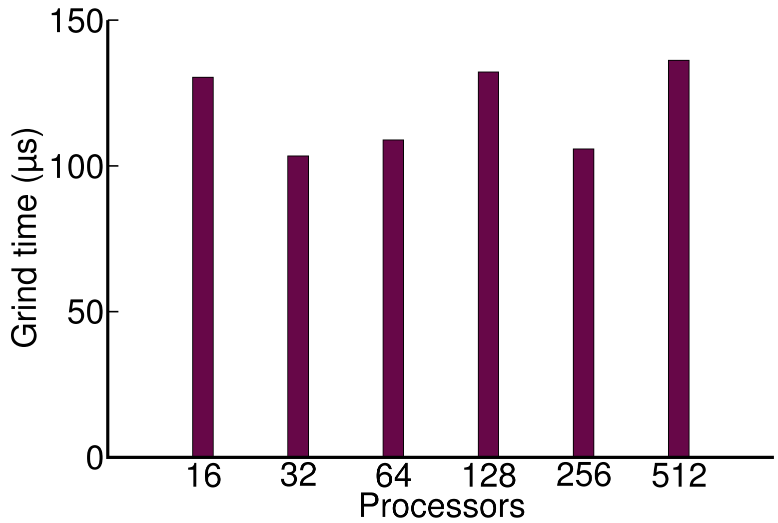
- Grind time increases by a factor of $\sim$ 1.6 - 2.1 over a range of 8 - 256 processors.
- Communication takes less than 2.5% of the running time.

Parameters – Multigrid / Seaborg

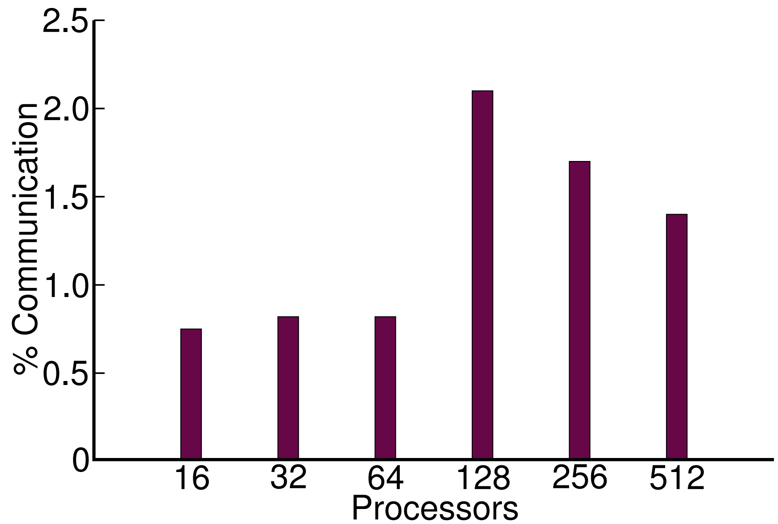
P	q	C	N
16	4	3	384^3
32	4	4	512^3
64	4	5	640^3
128	8	6	768^3
256	8	8	1024^3
512	8	10	1280^3

- More work per processor
- Relatively less overlap among domains

Results – Multigrid / Seaborg



Grind times



Communication percent

- Grind time increases by a factor of < 1.3 on Seaborg over a range of 16 - 512 processors.

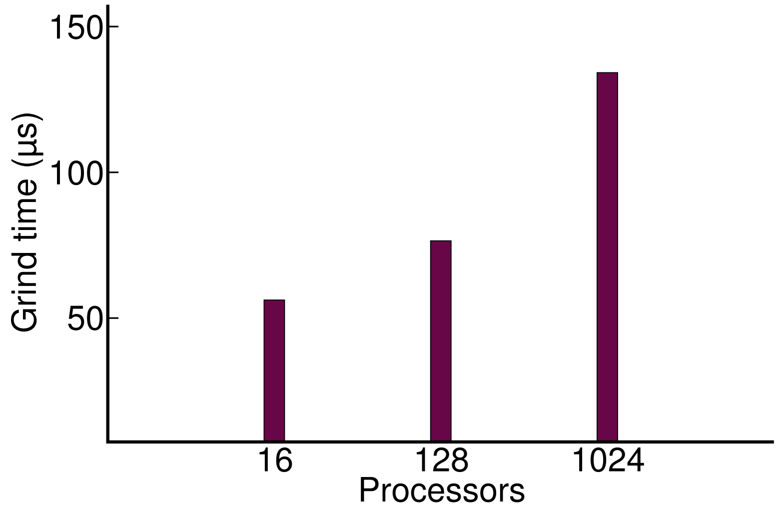
- Communication takes less than 2.5% of the running time.

Parameters – FFTW / Seaborg

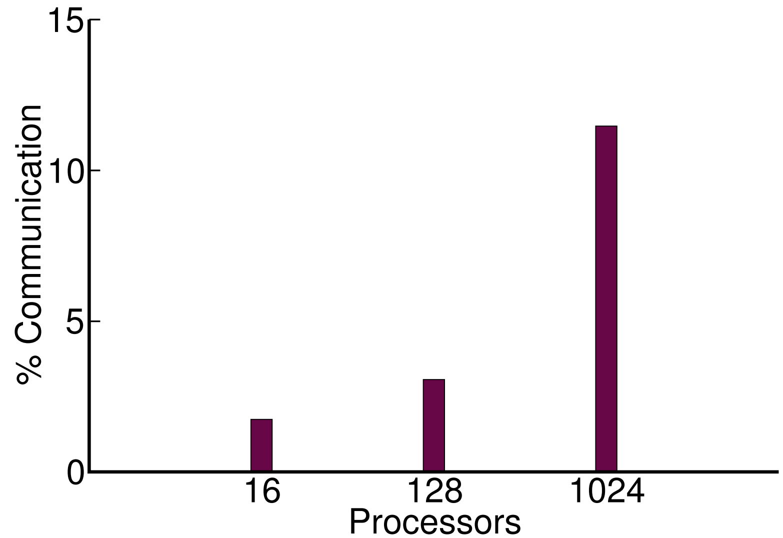
P	q	C	N
16	4	3	384^3
128	8	6	768^3
1024	16	12	1536^3

- 16 and 128 processor cases same as before
- Scaled up to 1024 processors

Results – FFT / Seaborg



Grind times



Communication percent

- Grind time increases by a factor of 2.4 over a range of 16 - 1024 processors on Seaborg.
- Communication takes less than 12% of the running time.

Conclusions and Future Work

- Communication cost is small: less than 12% on up to 1024 processors.
- Speed-up is fairly good up to 1024 processors.
  - Slowdown of a factor of 2.4 going from 16 to 1024 processors.
- Scaling to larger problems is underway
  - Reducing the effective cost of domain overlap.
  - Reducing the cost of the infinite domain boundary calculation.
- Extension to adaptive mesh refinement algorithm

Scallop

- For more information, visit

  http://www.cs.ucsd.edu/∼gballs/scallop/
  http://www.cs.ucsd.edu/groups/hpcl/scg/

- This work was supported by
  - National Partnership for Advanced Computational Infrastructure (NPACI)
  - National Science Foundation
  - U.S. Department of Energy