Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus

Gabrielle Allen, Thomas Dramlitsch, Ian Foster & al

Cristina Tugurlan

Center for Computation & Technology

AT LOUISIANA STATE UNIVERSITY

Outline

- Introduction
- Computational Grid Environment
- The Grid-Enabled Cactus Toolkit
- Experimental Results
- Summary
- References and Future Readings

Introduction

- Supercomputer centers were oversubscribed (SC2001).
- There was need of distributed techniques because:
  - increase accessibility ,
  - scale of large-scale simulation.
- Grid environments have a high degree of heterogeneity and dynamic behavior.
- A Grid enabled computational framework:
  - implement advance techniques
  - hide irrelevant complexities
  - present application developers with familiar abstractions.
  - such frameworks are ***Cactus*** and ***Globus*** based MPICH-G2 implementation of MPI ---- ***Cactus-G***

The Computational Grid Environment

Grid differs from parallel computing environment:

- nodes with different processor types, memory sizes
- highly heterogeneous and unbalance communication network :
    - mix of different intramachine networks
    - variety of Internet connections (bandwidth and latency)
- resource availability can vary over time and space
- integrates different operating systems and utilities

No conventional parallel program can run efficiently in such a Grid environment --- poor performances.

The Computational Grid Environment

Specialized techniques to overcome problems:

1. Irregular data distributions (load imbalances)
2. Grid-aware communication schedules
   - increase message sizes
   - organize data distributions
   - dedicated communication processors
3. Redundant computation (reduce comm. costs)
4. Protocol tuning (compress messages prior transmission over slow links)

The Grid-Enabled Cactus Toolkit
Cactus and MPICH-G2

Cactus

- an open source problem solving environment designed for scientists and engineers.
- originated in the academic research community
- developed and used by a large international collaboration of physicists and computational scientists
- its modular structure enables parallel computation across different architectures and collaborative code development between different groups.

Cactus Features

Powerful Application Programming Interface

- user modules (thorns) plug-into compact core (flesh)
- configurable interfaces, schedules and parameters

Highly Portable

- supported on most architectures
- sophisticated make system

Cactus Features

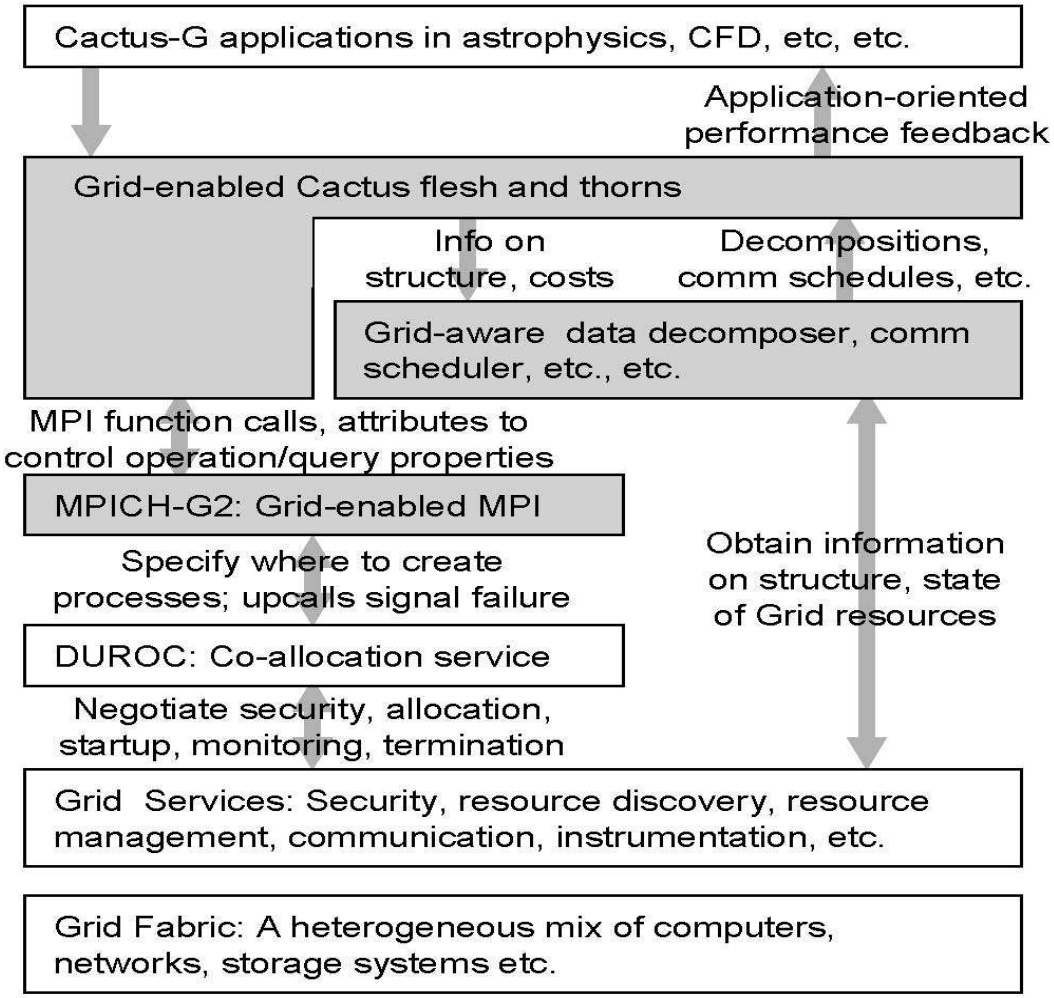
Advanced Computational Toolkit

- Accessible MPI-based parallelism for finite difference grids
- Access to a variety of supercomputing architectures and clusters
- Several parallel I/O layers
- Fixed and Adaptive mesh refinement under development
- Elliptic solvers
- Parallel interpolators and reductions
- Metacomputing and distributed computing

MPICH-G2

- an MPI implementation for exploiting heterogeneous collections of computers
- second version, first was MPICH-G
- exploits Globus services for:
  - resource discovery
  - resource allocation
  - startup
  - faster communications
  - authentication
  - executable staging
  - management and control
  - quality of service

Cactus Architecture for Grid



Cactus defines a set of layered abstractions and associates libraries

Cactus Architecture for Grid

- ***Grid-aware application layers***:
  - top ***Cactus application thorns*** for scientific computation (Fortran, C, C++)
  - how data is distributed across processors and how communication is done, is hidden from the programmer
- ***Grid-enabled communication library***: MPICH-G2
  - the abstraction presented is the MPI programming model
  - uses the *Dynamically Updated Resource Online Co-Allocator (DUROC)*

Cactus Architecture for Grid

- ***Grid-aware infrastructure*** thorns:
  - provides all features, layers, and drivers (parallel I/O, web interfaces, visualization, data mapping, communication)
  - Cactus driver thorn = ***PUGH*** provides MPI based parallelism. Improved and optimized for a heterogeneous Grid environment
- ***Grid services***:
  - Uniform protocols and APIs

Experimental Results

Performed a distributed run of a scientific Fortran application across 4 supercomputers at two sites NCSA Champaign Urbana and SDSC California.

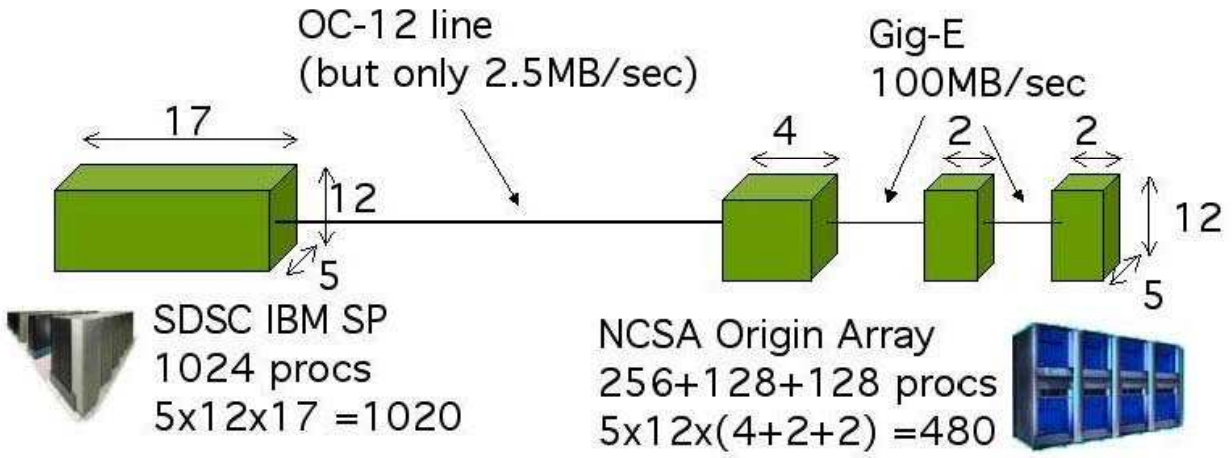


Figure 2: Processor topology across machines: Aegir [$5 \times 12 \times 2 = 120$], Forseti [$5 \times 12 \times 2 = 120$], Balder [$5 \times 12 \times 4 = 240$], and Blue Horizon [$5 \times 12 \times 17 = 1020$].

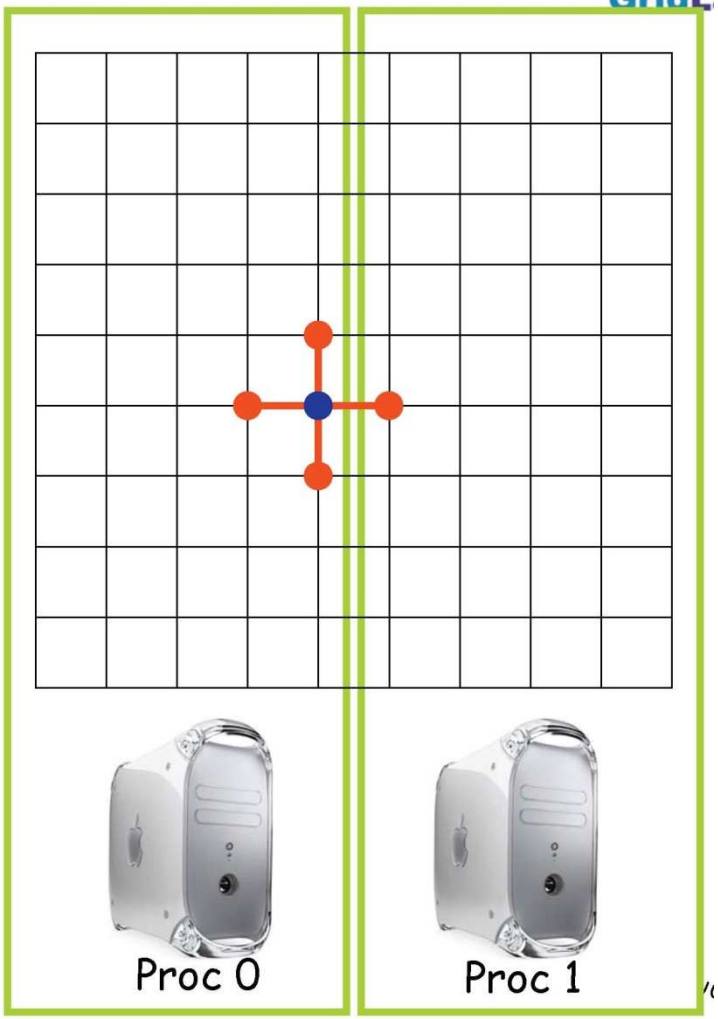
Distributes Terascale Computing

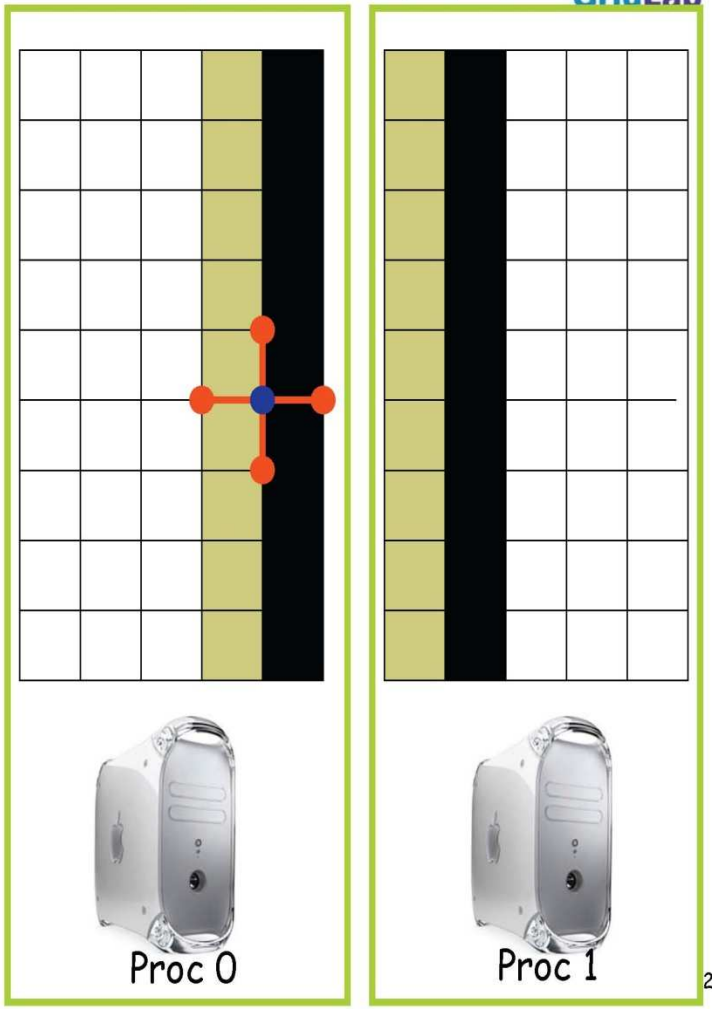
- Code involves finite differencing techniques (780 floating-point operations per grid point at each iteration)
- Techniques:
  - Ghostzones larger: increase communications granularity, memory usage
  - Compressed data before transferring it for bandwidth constraints
  - Overlapped communications to reduce latency effects

Ghost-zones

- are additional (duplicated) grid points which are added to each processor containing the required data to advance to the next iteration
- the data on each grid must be periodically synchronized
- the ghostzones are updated with current data

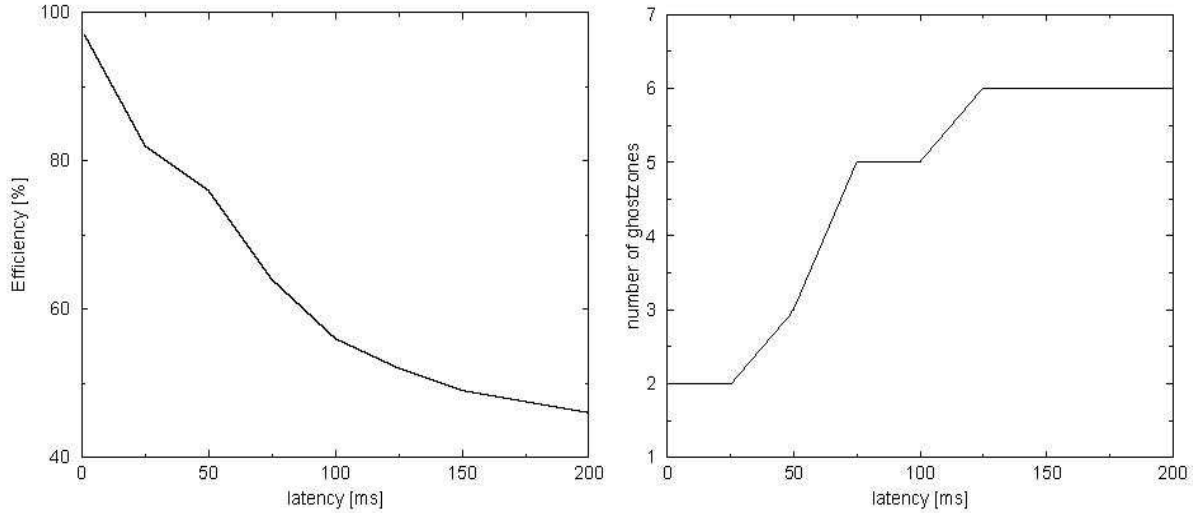
Ghost-zones





Ghost-zones

- Efficiency goes down as latency increases
- Graph shows the optimal number of ghostzones against the latency (4 ghost –efficiency 92%)



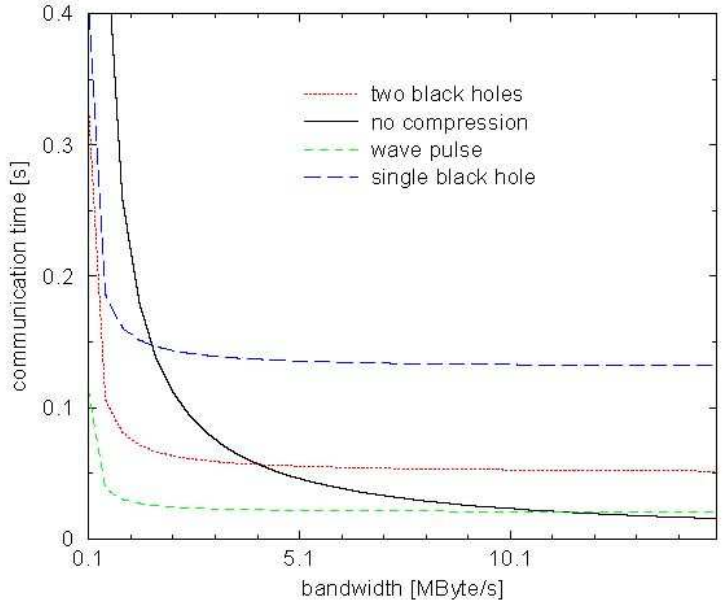
Compression

- Cactus has a thorn that compresses messages prior transmission across the network
- It is efficient, much better than 50% up to 90%
- Adaptive algorithm that adjust compression parameters on the fly

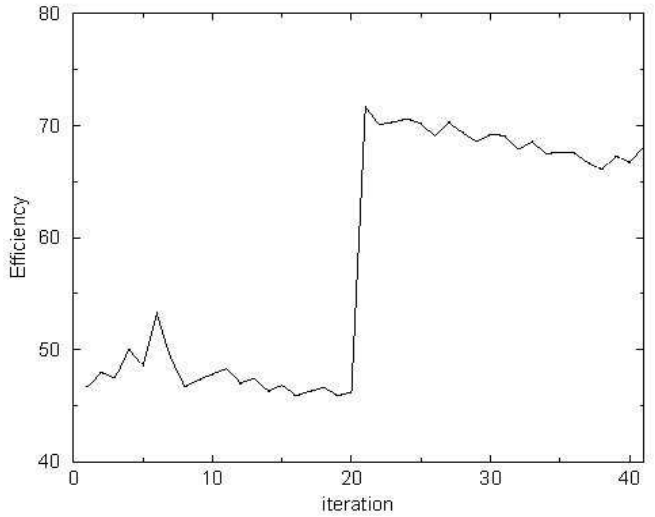
Communication Overlap

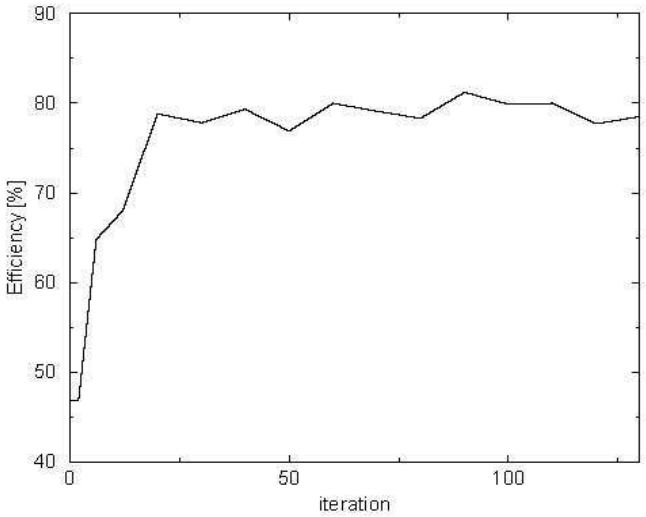
- Redistribute gridpoints so that the WAN communicating processors had fewer grid points and could overlap communication with computation on the other processors

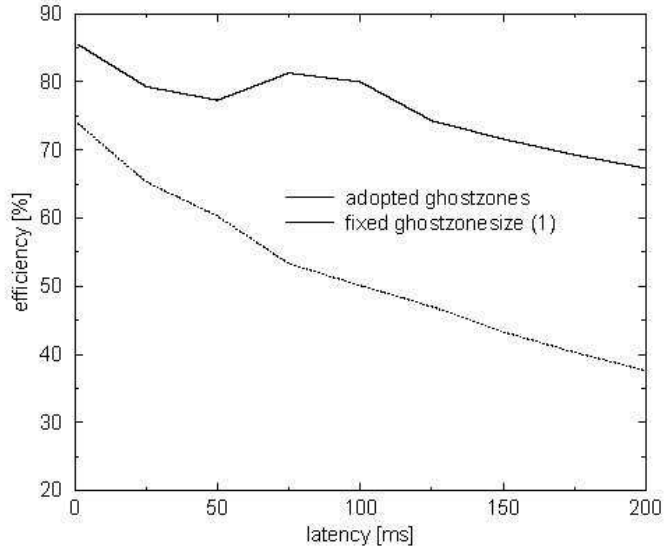
Communication time depending on bandwidth and compression

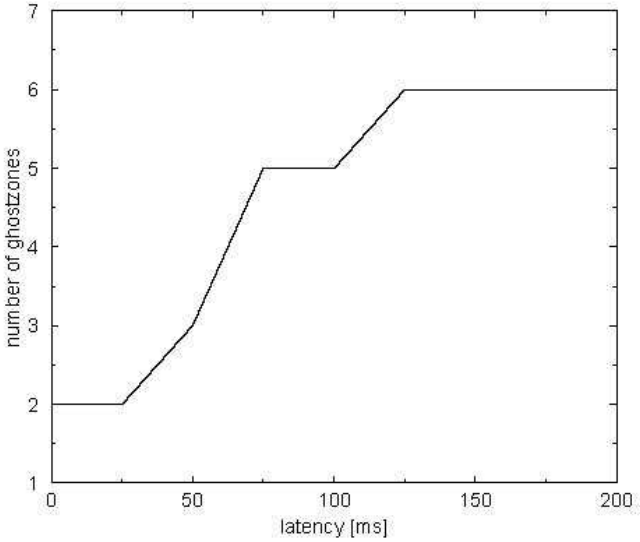


Adaptive Techniques









Summary

- The heterogeneous, dynamic and unreliable nature of Grid environments has major implications for application development
- Using special communication and latency reducing techniques, the efficiency is increased from 14% (without) to over 80% (with techniques)
- Want that adaptive techniques to find automatically the optimal communication parameters.

References and Further Readings

References

[1] G. Allen, W. Benger, C. Hege, J. Massó, A. Merzky, T. Radke, E. Seidel, and J. Shalf. Solving einstein's equations on supercomputers. *IEEE Computer*, 32(12), 1999.

[2] G. Allen, T. Goodale, and E. Seidel. The cactus computational collaboratory: Enabling technologies for relativistic astrophysics, and a toolkit for solving pdes by communities in science and engineering. In *7th Symposium on the Frontiers of Massively Parallel Computation-Frontiers 99*, New York, 1999. IEEE.

[3] F. Berman. High-performance schedulers. In *[11]*, pages 279–309.

[4] S. Brunett, D. Davis, T. Gottschalk, P. Messina, and C. Kesselman. Implementing distributed synthetic forces simulations in metacomputing environments. In *Proceedings of the Heterogeneous Computing Workshop*, pages 29–42. IEEE Computer Society Press, 1998.

[5] *http://www.cactuscode.org*.

[6] IMPI Steering Committee. IMPI - interoperable message-passing interface, 1998. `http://impi.nist.gov/IMPI/`.