Using the Pyramid Library for Finite Element Mesh Adaptation

Charles D. Norton, John Z. Lou, and Thomas A. Cwik

National Aeronautics and Space Administration

Jet Propulsion Laboratory

California Institute of Technology

http://hpc.jpl.nasa.gov/APPS/AMR

HPCC/Earth and Space Sciences Project

PYRAMID: Parallel Unstructured Adaptive Mesh Refinement

PYRAMID

Modern... Simple... Efficient... Scalable...

Technology Description

An advanced software library supporting parallel adaptive mesh refinement in large-scale, adaptive scientific & engineering simulations.

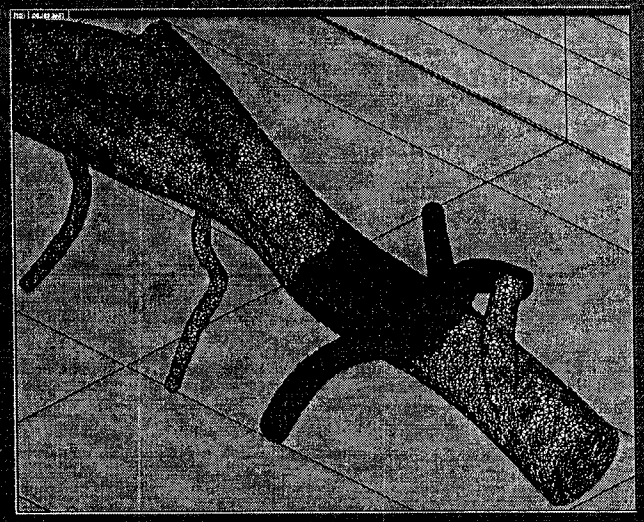
State-of-the-Art Design!

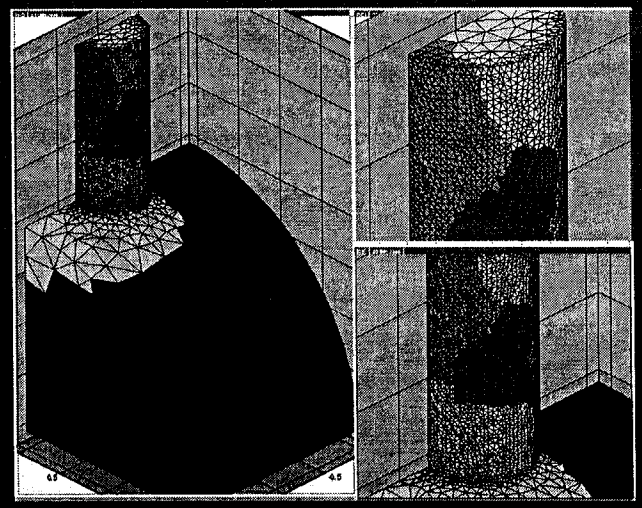
- Efficient object-oriented design in Fortran 90 and MPI
- Automatic mesh quality control & dynamic load balancing
- Scalable to hundreds of processors & millions of elements

Application Arena

- Computer Modeling & Simulation Applications with complex geometry
- Electromagnetic and semiconductor device modeling
- Structural/Mechanical/Fluid dynamics applications

John Z. Lou, Charles D. Norton, & Thomas A. Cwik
High Performance Computing Systems and Applications Group
http://hpc.jpl.nasa.gov/APPS/AMR

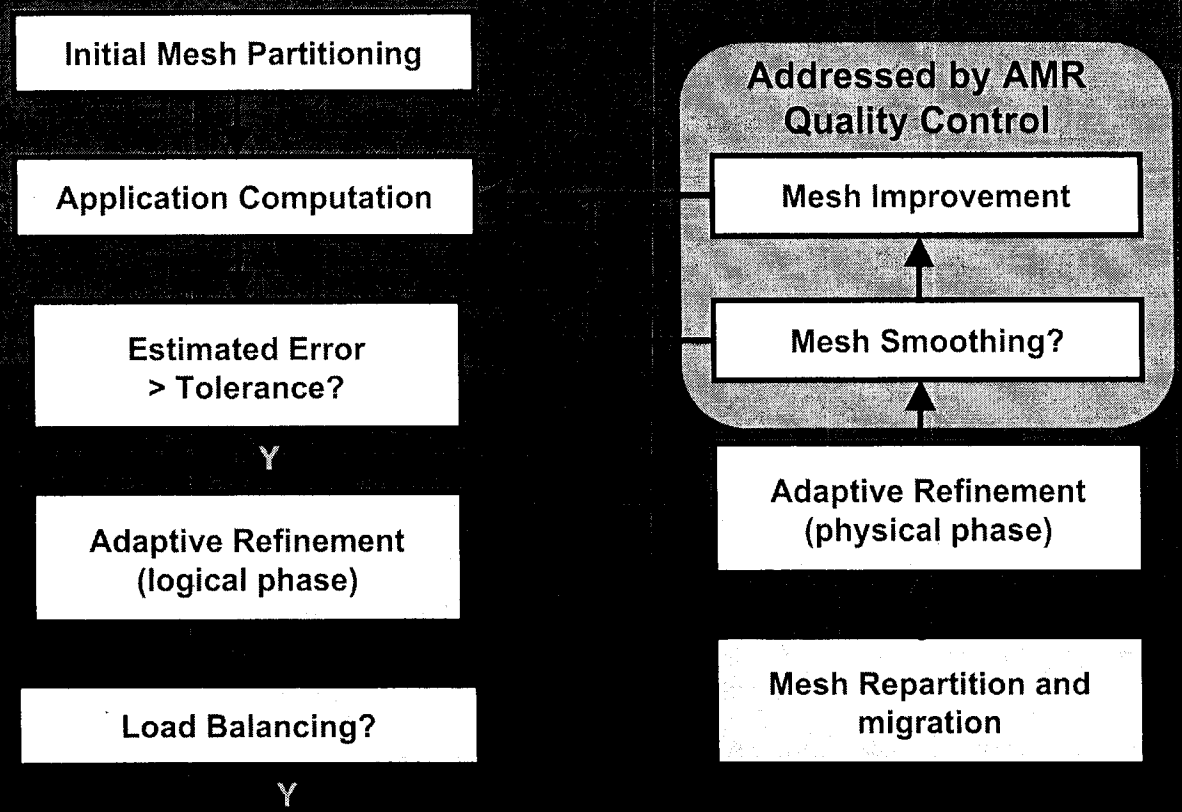




Initial geometry courtesy of SCOREC (Rensselaer)

Our Parallel AMR Process

- **Organization**
  - Partitioning, Adaptive Refinement, Load Balancing, Mesh Migration, and Element Quality Control



Technology

- **Fortran 90/95 Features Modernize Programming**

Modules
Encapsulate data and routines across program units

Generic Interfaces
One call can perform different actions based on types

Array Syntax
Simplifies whole array, and array subset, operations

Use-Association
Controls access to module content

Derived Types
User-defined types supporting abstractions in programming

Pointers/Allocatable Arrays
Supports flexible/dynamic data structures

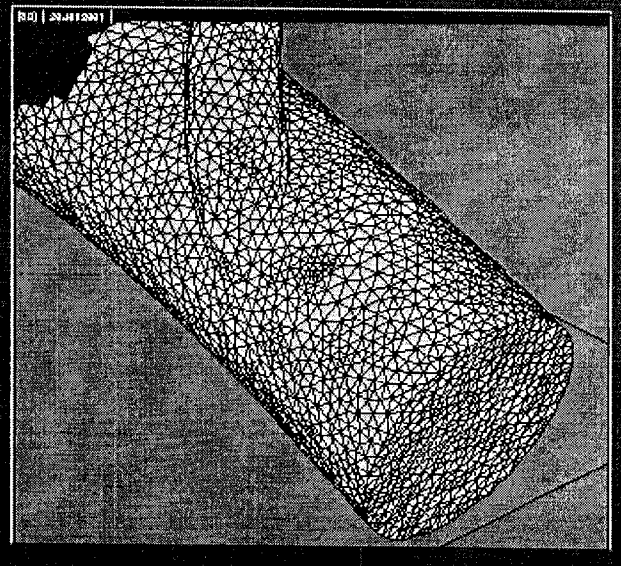
Backward compatible with Fortran 77

FOR MORE INFO...

Fortran 90 Programming. Ellis, Philips, & Lahey; Addison Wesley, 1994
http://hpc.jpl.nasa.gov/PEP/nortonc/oof90.html

Technology

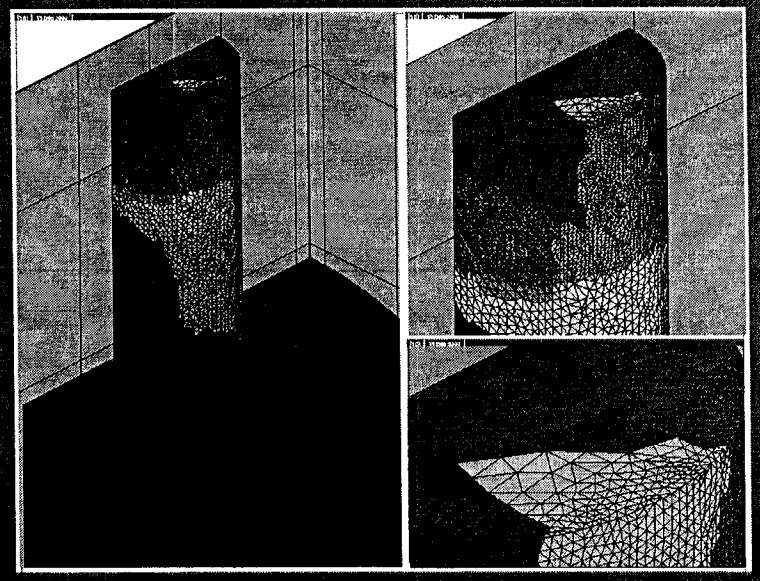
- **A Minimal PYRAMID Program**
  - Initialization Section
    - Optional arguments override defaults



```
PROGRAM pyramid_example
USE pyramid_module
implicit none
   ! Statements omitted...
   type (mesh), dimension(2) :: meshes
   call PAMR_INIT()
   call PAMR_LOAD_MESH_PARALLEL( meshes(1), in_file )
   call PAMR_REPARTITION( meshes(1) )
   ! Adaptive refinement loop…
   call PAMR_ELEMENT_COUNT( meshes(2) )
   call PAMR_VISUALIZE( meshes(2), "visfile.plt" )
   call PAMR_FINALIZE( mpi_active = .true. )
END PROGRAM pyramid_example
```

Technology

- **A Minimal PYRAMID Program**
  - Adaptive Refinement



```
PROGRAM pyramid_example
! Adaptive refinement loop...
   do i = 1, refinement_level
      call PAMR_ERROR_EST( meshes(1), &
                           meshes(2) )
      call PAMR_LOGICAL_AMR( meshes(1) )
      call PAMR_REPARTITION( meshes(1) )
      call PAMR_PHYSICAL_AMR( meshes(1), meshes(2) )
   end do
END PROGRAM pyramid_example
```

  - Users must specify their error estimation method
  - Mesh hierarchies can be defined

Technology

Object-Based Access to Data Structure

- Explicit reference to element coordinates is complicated

```
type (mesh) :: this
real, dimension(3) :: xyz_pos
xyz_pos = this%nodes(this%elements(2)%node_indx(1))%coord
```

- PYRAMID simplifies such references

```
type (mesh) :: this
real, dimension(3) :: xyz_pos
real, dimension(3,4) :: all_pos
real, dimension(3,3,4) :: n_normal
xyz_pos  = PAMR_ELEMENT_COORD(this, element_indx=2, &
                              node_indx=1)
all_pos  = PAMR_ELEMENT_COORD(this, element_indx=2)

! Access signed local normal basis for all faces
n_normal = PAMR_FACE_NORMALBASIS(this, element_indx=3)
```

Technology

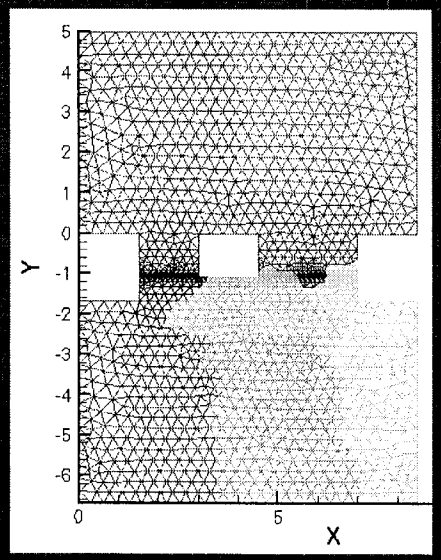
- **Numerous User-Driven Commands Are Included**
  - Initialization, Mesh I/O, Termination, Adaptive Refinement, Repartitioning, Data Migration, Visualization, Data Structure Access, Mathematical, and Auxiliary
  - Almost every command contains optional arguments for use customization

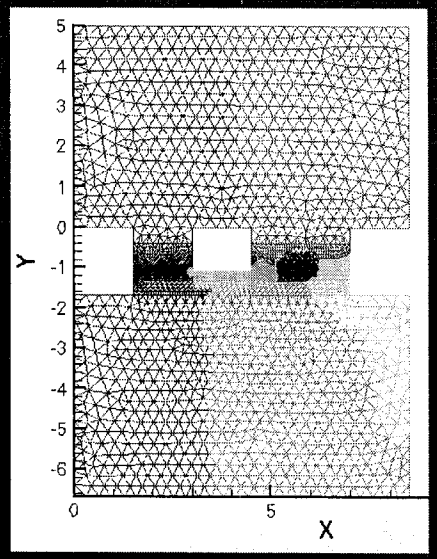
```
PAMR_CURRENT_TIME
PAMR_DEFINE_MESH_TERMS
PAMR_ELAPSED_TIME
PAMR_ELEMENT_CENTROID
PAMR_ELEMENT_COORD
PAMR_ELEMENT_COUNT
PAMR_ELEMENT_ID
PAMR_ELEMENT_VOLUME
PAMR_ERROR_EST
PAMR_FACE_AREA
PAMR_FACE_CENTROID
PAMR_FACE_COORD
PAMR_FACE_INDX
PAMR_FACEEDGE_ID
PAMR_FACEEDGE_INDX
PAMR_FACE_NORMALBASIS
PAMR_FACE_UNITNORMAL
PAMR_FINALIZE
PAMR_GET_EDGE_TERMS
PAMR_GET_ELEMENT_TERMS
PAMR_GET_FACE_TERMS
PAMR_GET_NODE_TERMS
PAMR_INIT
PAMR_LOAD_MESH_COMP
PAMR_LOAD_MESH_SERIAL
PAMR_LOAD_MESH_PARALLEL
PAMR_LOGICAL_AMR
PAMR_MAP_MESH_TERMS
PAMR_PHYSICAL_AMR
PAMR_REPARTITION
PAMR_SAVE_MESH
PAMR_SET_EDGE_TERMS
PAMR_SET_ELEMENT_TERMS
PAMR_SET_FACE_TERMS
PAMR_SET_NODE_TERMS
PAMR_VISUALIZE
```

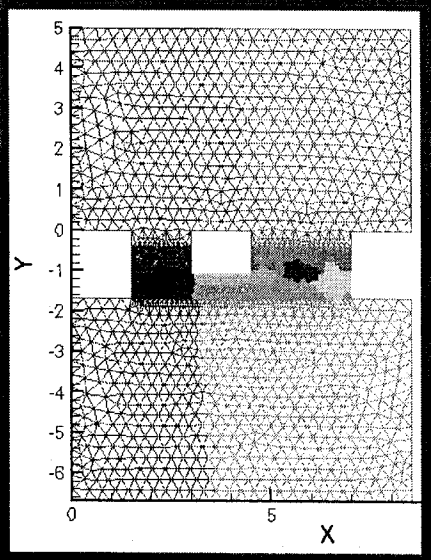
  - Most commands are generic based on the mesh component applied

Technology

- **Dynamic Load Balancing with ParMetis**
  - ParMetis gives partitioning, PYRAMID performs migration
  - Migration handles irregular communication patterns with a scalable and efficient non-blocking algorithm





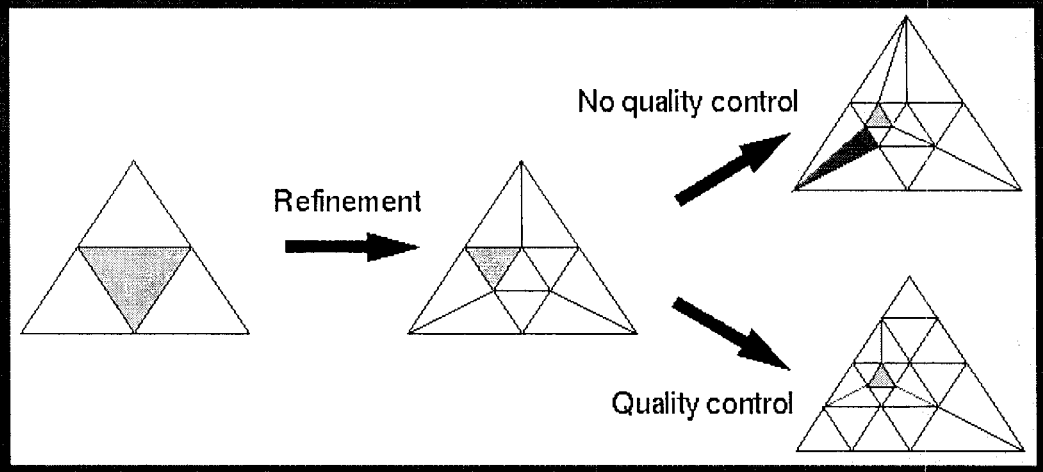


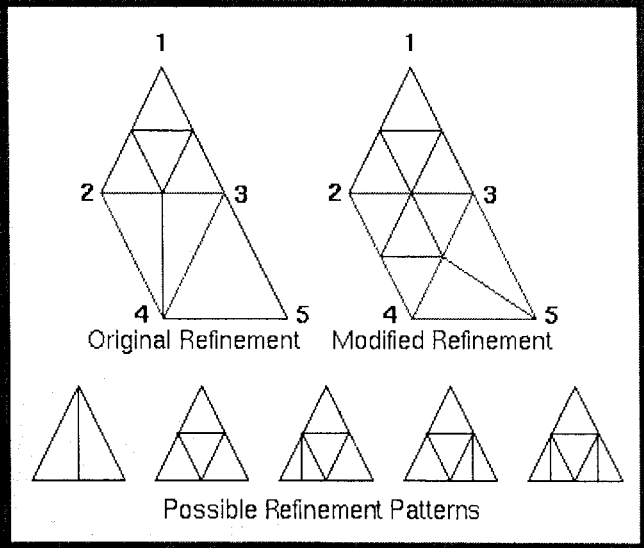
  - We are investigating Zoltan (Sandia National Labs) as an additional option for partitioning

Technology

Automatic Mesh Quality Control

- Modify coarse element refinement if successive refinements cause poor aspect ratios



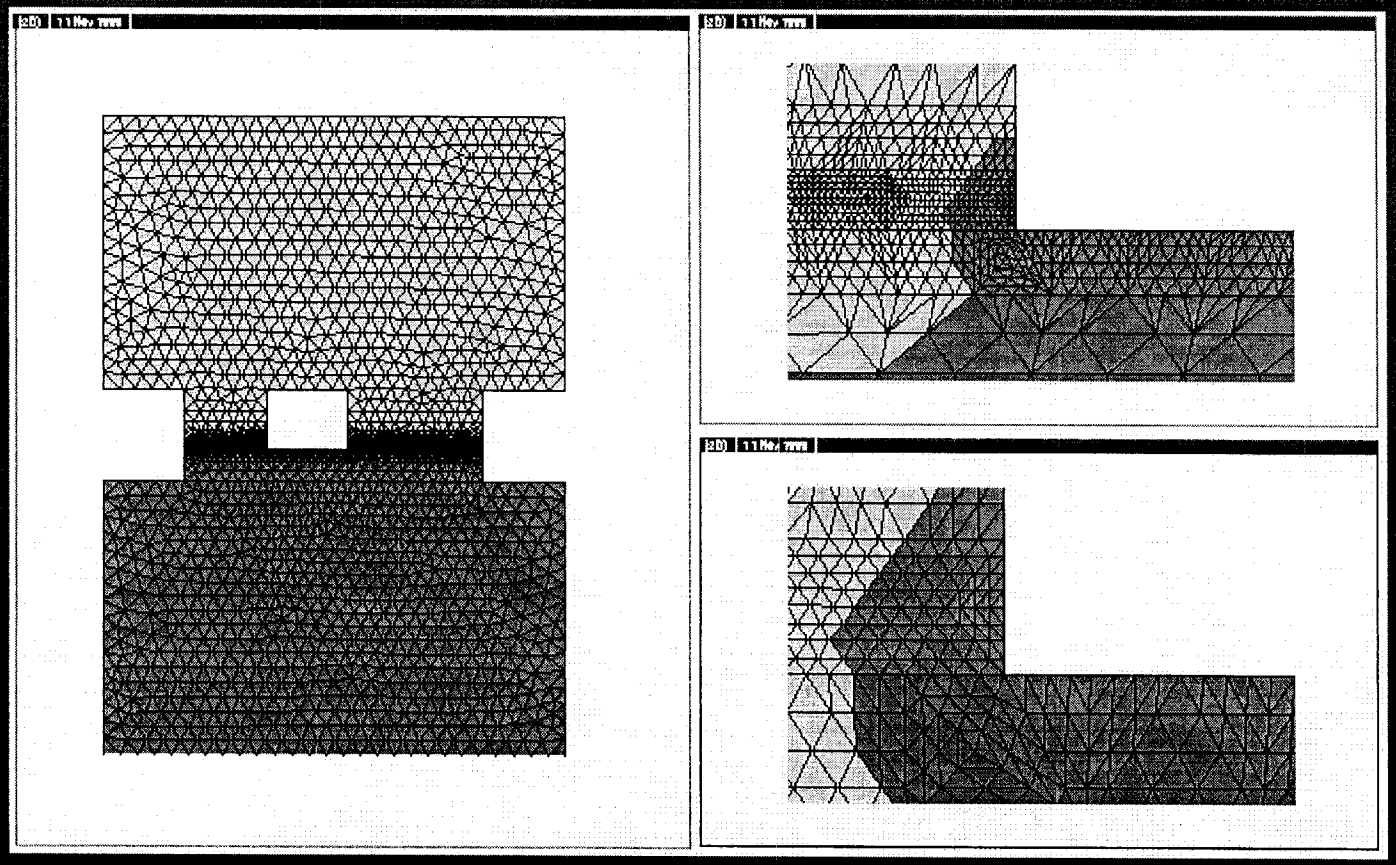


- Controls quality at the expense of additional elements

Technology

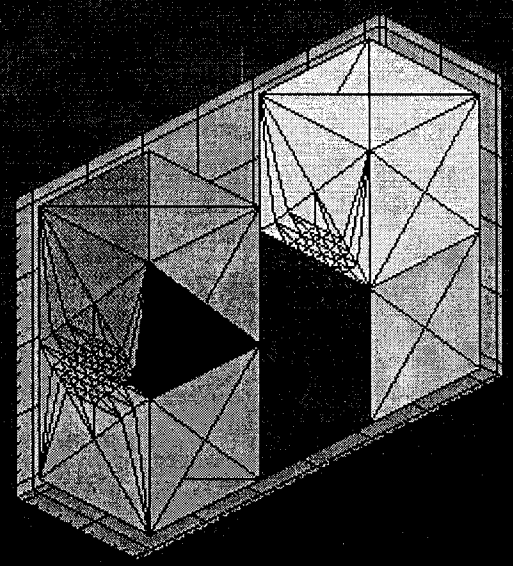
Automatic Mesh Quality Control

- Benefit of quality control applied to triangular elements

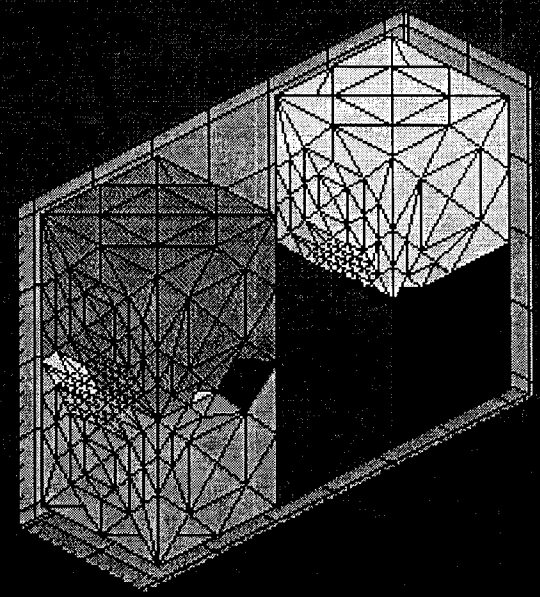


Technology

- **Automatic Mesh Quality Control**
  - Benefit of quality control applied to tetrahedral elements



Poor Mesh Elements Without Quality Control

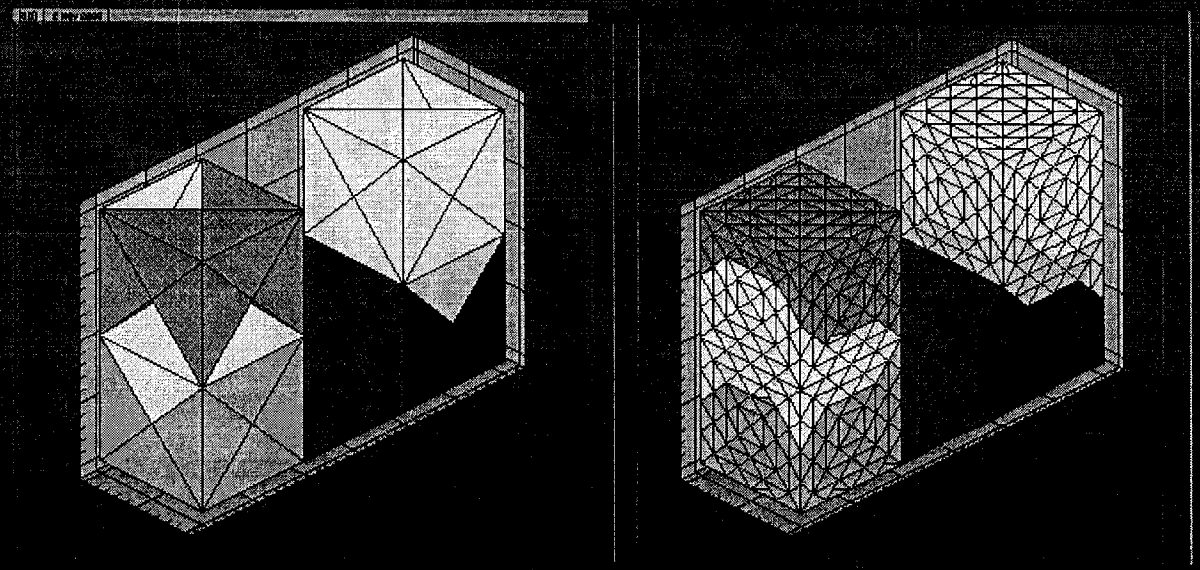


Good Mesh Elements With Quality Control

Note : Tecplot shows some edges in the backplane that do not exist in the mesh...

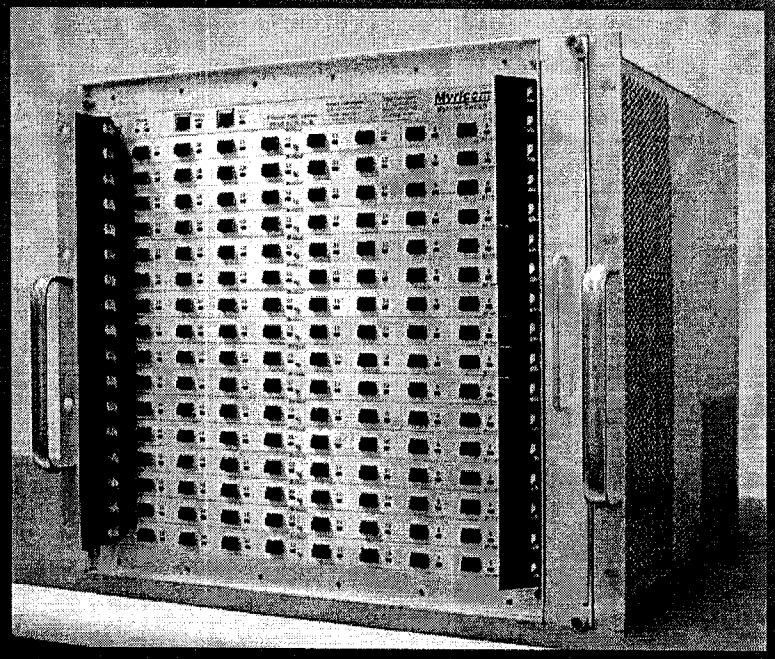
Technology

- **Large Scale Parallel Mesh Generation**
  - Specify uniform error for generation from coarse meshes



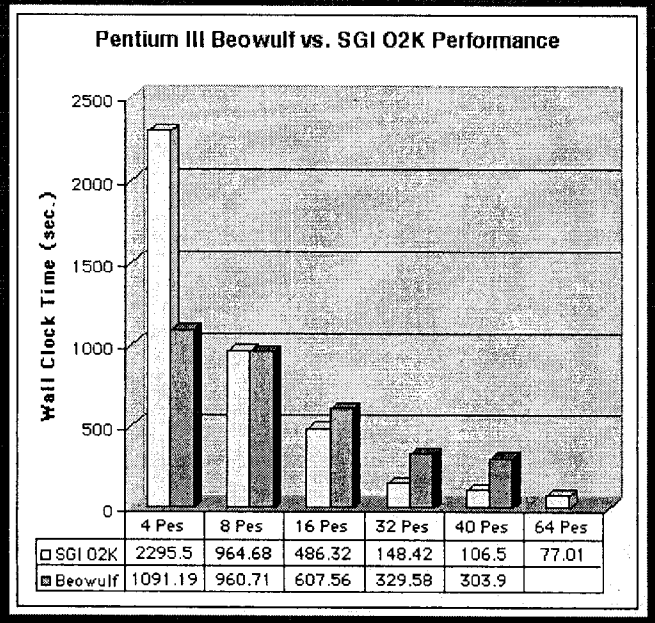
Parallel Uniform Refinement

Performance issues in Algorithms and Networking for Beowulf-Class Clusters

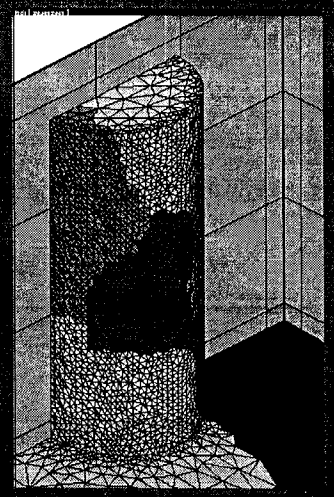


Performance

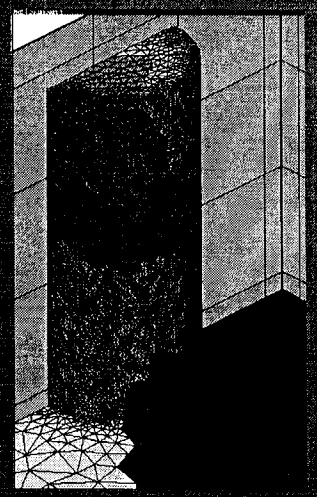
- **Pentium III Beowulf Cluster vs. SGI O2K Parallel AMR**
  - O2K scales well although the processor is slower than the 800 Mhz Beowulf PIII
  - Beowulf competes well, but performance is limited by 100 BaseT network



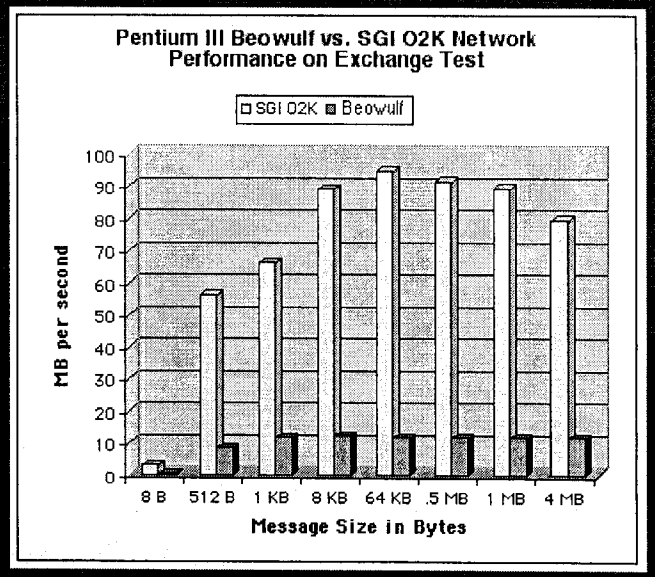
	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes	64 Pes
SGI O2K	2295.5	964.68	486.32	148.42	106.5	77.01
Beowulf	1091.19	960.71	607.56	329.58	303.9	



Repartitioned Muzzle Brake



After 3 Adaptive Refinements



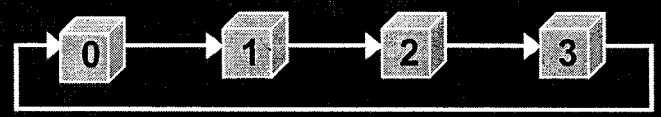
Performance

Irregular Data Communication

- Migration requires irregular, but predictable, data movement that varies in size and destination

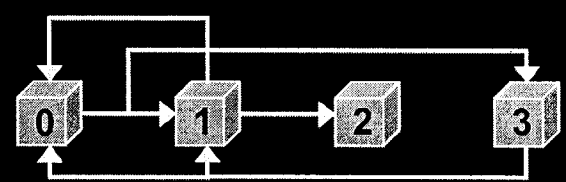
Circular-Shift "MPI_SENDRECV(...)"

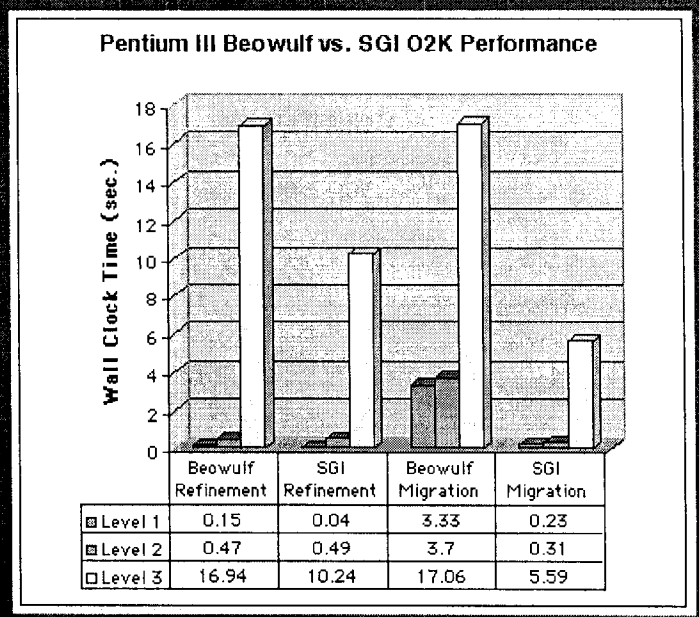
- All processors inspect all of the data
- "Guarantees" handling of cyclic deadlock dependencies
- Irregular data sizes affect pipelined flow performance
- MPI implicit buffering, due to poor pipeline structure, leads to poor performance



Direct Data Transfers

- Processors send/receive specific messages
- Send continuously while checking for receives
- "Arbitrary" message ordering can flood the network switch, leading to poor performance





	Beowulf Refinement	SGI Refinement	Beowulf Migration	SGI Migration
Level 1	0.15	0.04	3.33	0.23
Level 2	0.47	0.49	3.7	0.31
Level 3	16.94	10.24	17.06	5.59

Earthquake Mesh Refinement and Migration on 32 Processors

Performance

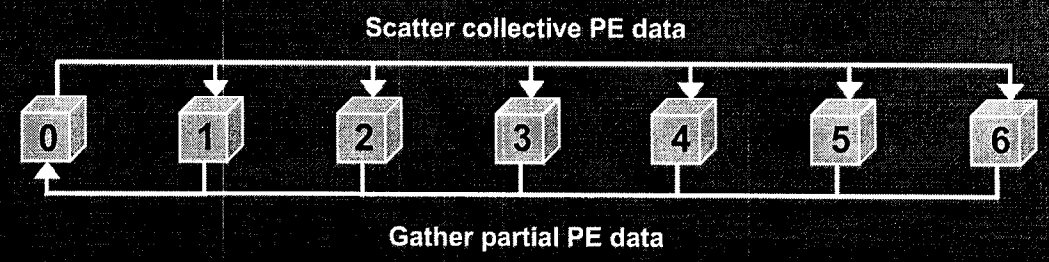
Irregular Data Communication

- Reduction schemes can improve performance, if implemented with care...

Note : Broadcasts simplify handling cases where the number of processors is not a power of two

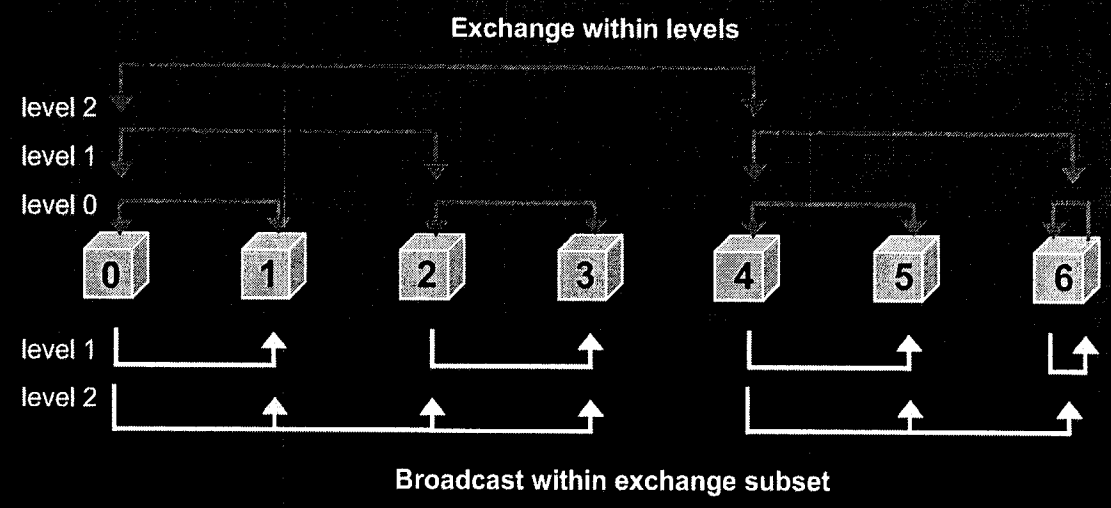
Reduce to 0 with Broadcast Scheme

- Not scalable and very inefficient for large data sets



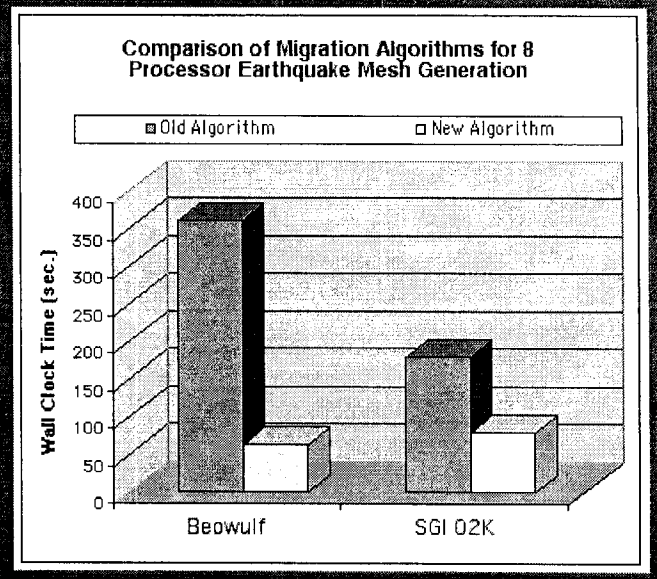
Reduce (or Exchange) to Leader with Subset Broadcast Scheme

- More scalable and efficient, but still requires multiple broadcasts at each tree level



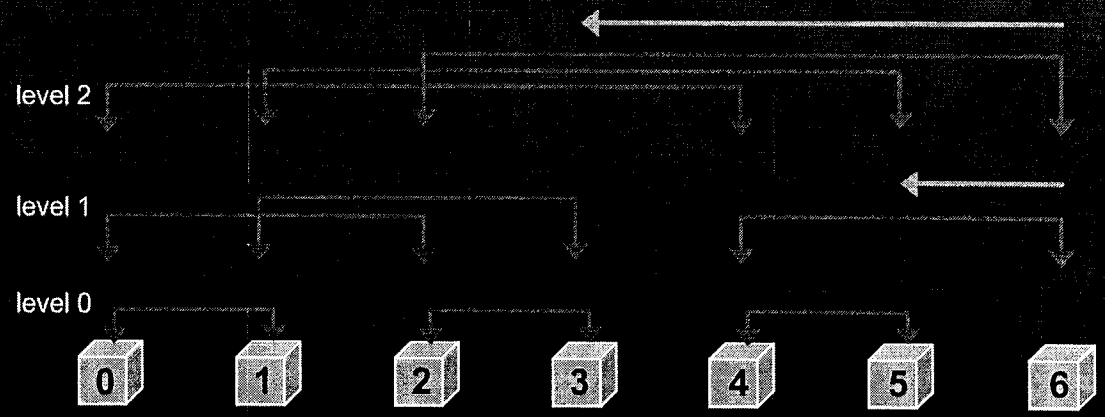
Performance

- **Irregular Data Communication**
  - Reduction schemes can improve performance, if implemented with care...
  - Our algorithm is non-blocking, scalable, and uses full-duplex communication



Our Algorithm Improvements

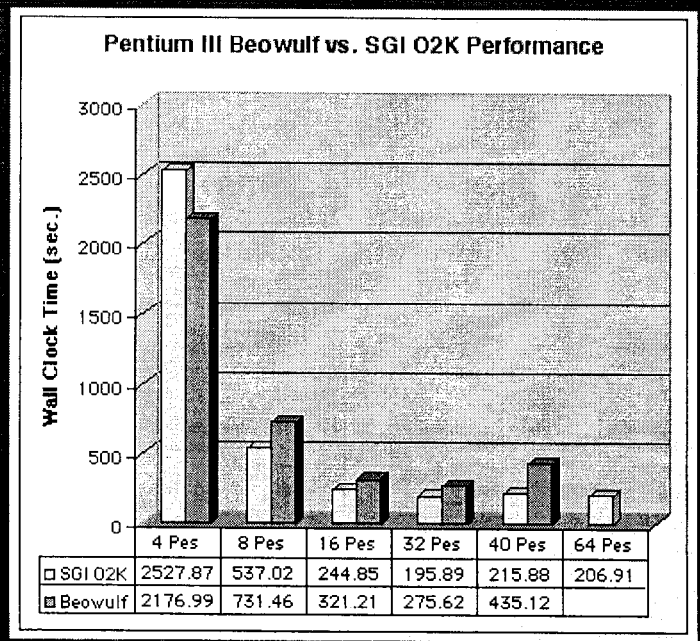
- Maximize exchanges at each level without repeated calculations
- Reduce data volume at each level with full-duplex communication
- Minimum number of broadcasts are required to support an arbitrary number of processors
- Processors which do not contribute to the calculation at a given level are idle (not shown in this example)



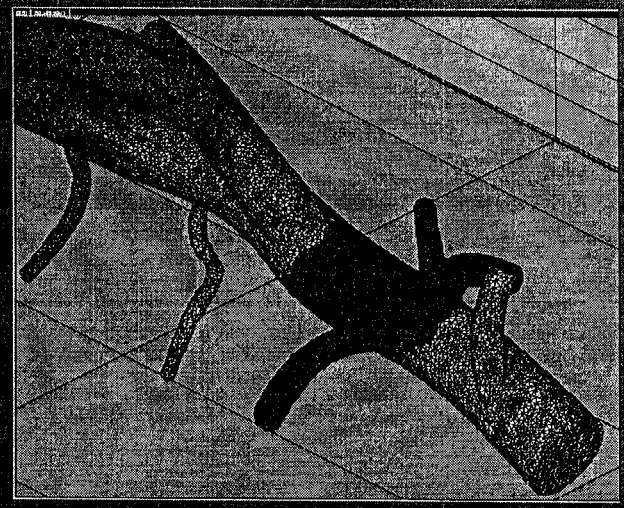
Maximum number of pairwise exchanges are performed

Performance

- **Pentium III Beowulf Cluster vs. SGI O2K Migration**
  - Beowulf performs well, but network still dominates with increasing processors
  - O2K is also affected for large (>30MB) messages



	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes	64 Pes
SGI O2K	2527.87	537.02	244.85	195.89	215.88	206.91
Beowulf	2176.99	731.46	321.21	275.62	435.12	



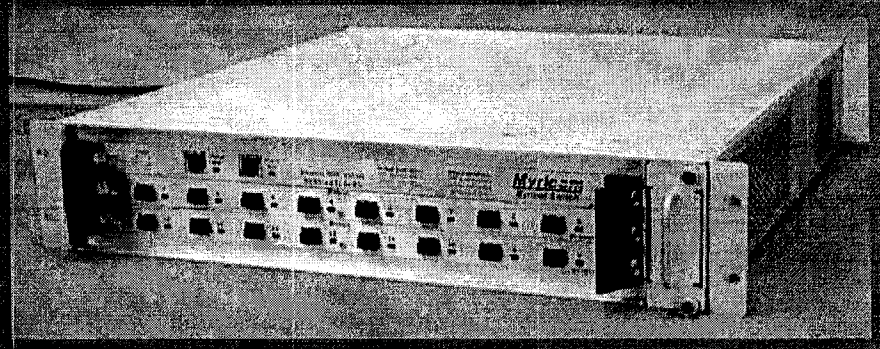
Artery Segment with 1.1 Million Elements

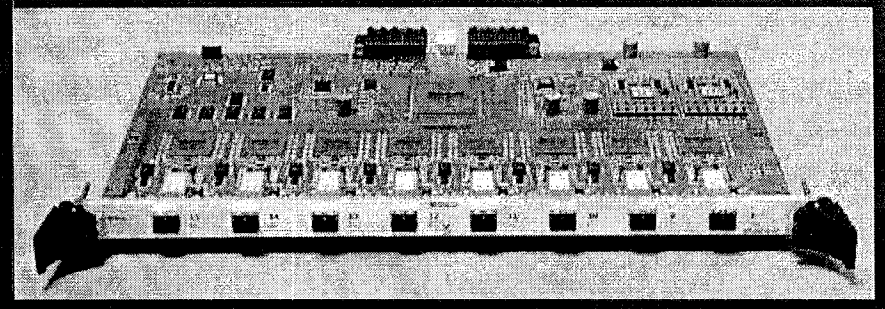
Adaptively Refined Artery Segment with 2 Million Elements

Note : New migration algorithms are applied

Myrinet 2000

- **2 Gbit/s Clos Topology**
- **Our Configuration**
  - 32 port M3-E32 Switch
  - PCI 64B, Lanai 9, SuperMicro 370 DLE Motherboard Serverworks LE rev 5 chipset
  - RedHat linux 6.2 Kernel 2.2.19 SMP
  - MPICH-GM 1.2.4 (latest 1.2.8)
  - Dual-PE 800 Mhz Pentium-III 52 Node System



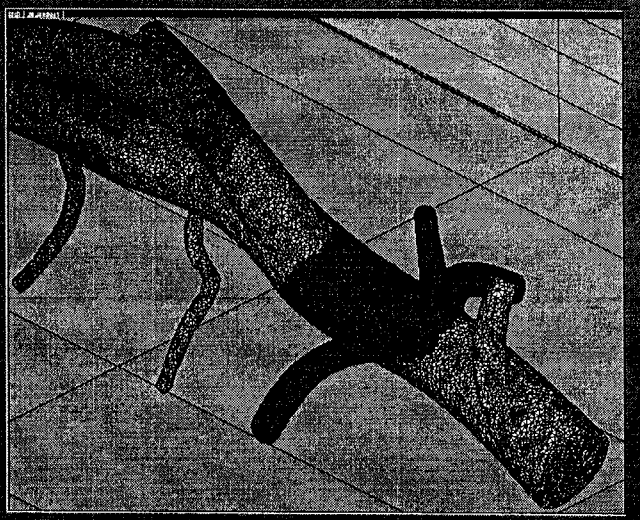


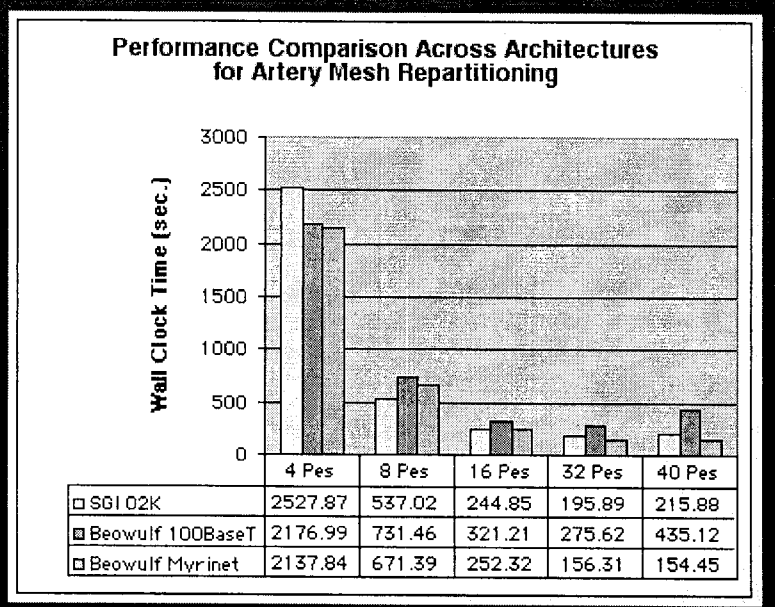
MPICH-GM Performance

- 225 MBytes/s (1.8 Gbit/s) 9.3 μsec latency
- 130 MBytes/s (1.0 Gbit/s) 1.5 μsec latency shared-memory communication

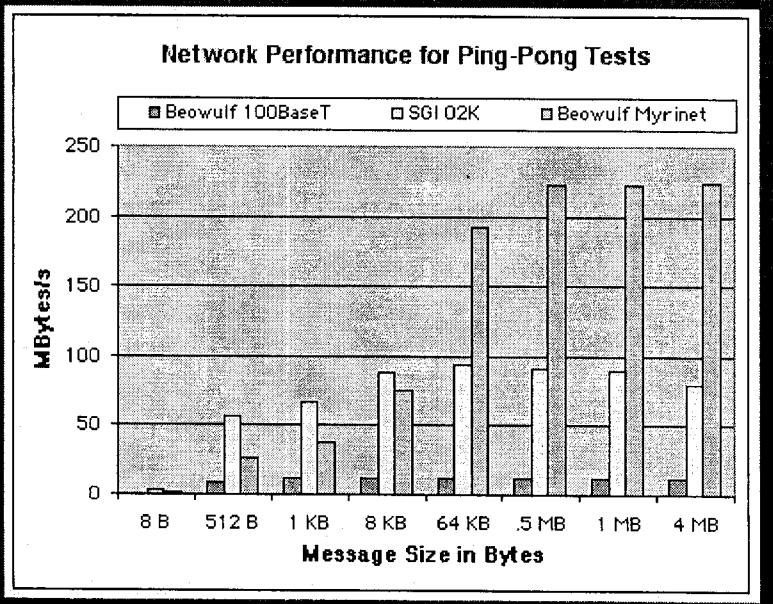
Myrinet Performance

- **Artery Mesh Repartitioning**
  - Improvement for larger numbers of processors compared to 100-BaseT Ethernet
  - The is essentially a communication benchmark



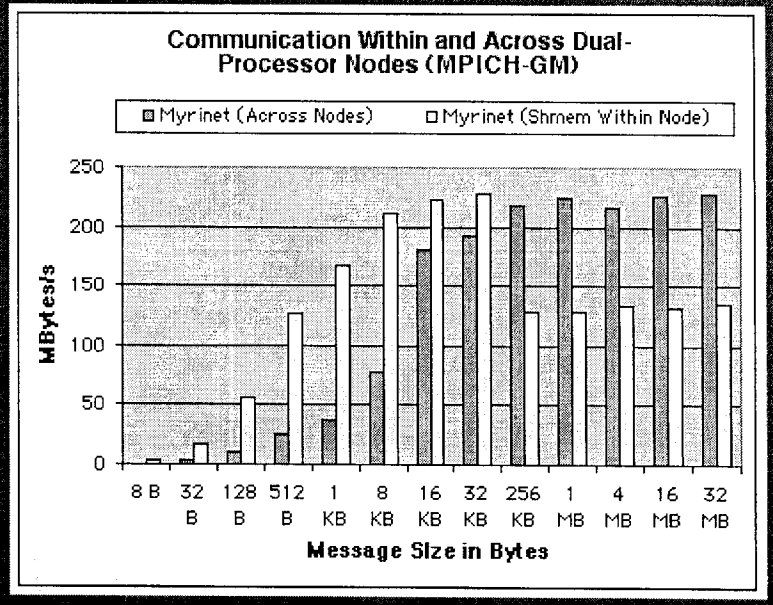


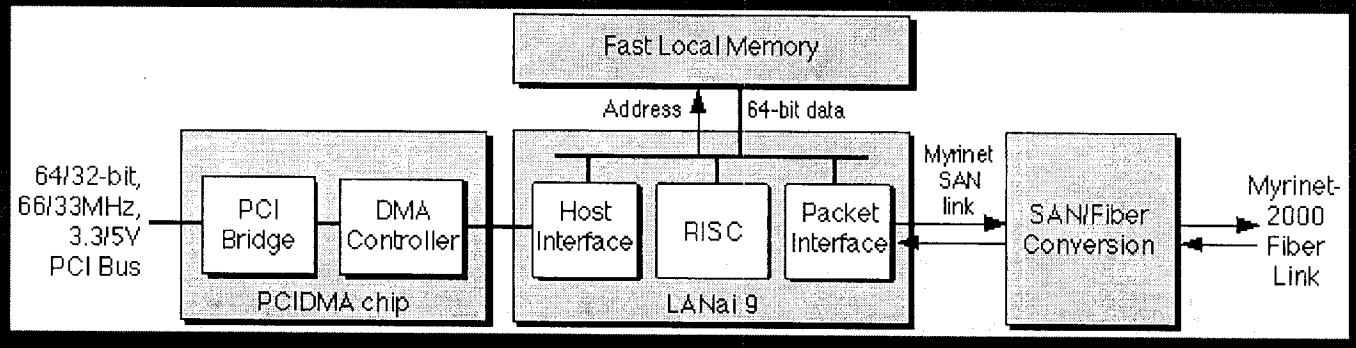
	4 Pes	8 Pes	16 Pes	32 Pes	40 Pes
SGI 02K	2527.87	537.02	244.85	195.89	215.88
Beowulf 100BaseT	2176.99	731.46	321.21	275.62	435.12
Beowulf Myrinet	2137.84	671.39	252.32	156.31	154.45



Myrinet Performance

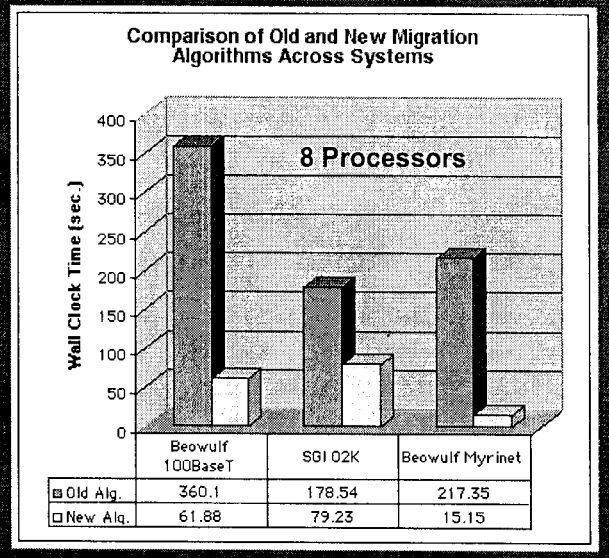
- **Effects of Shared-Memory for Dual-CPU Pentium III**
  - Shared-Memory processors access data from cache quickly for small messages, but performance is cache limited
  - New PCI chipsets reduce the need for shared-memory communication (Our PCI Bandwidth is 455 Mbytes/s)

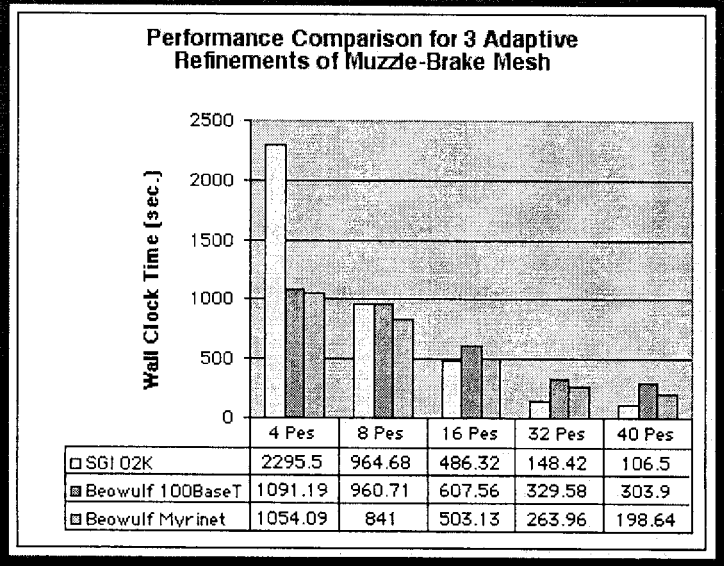


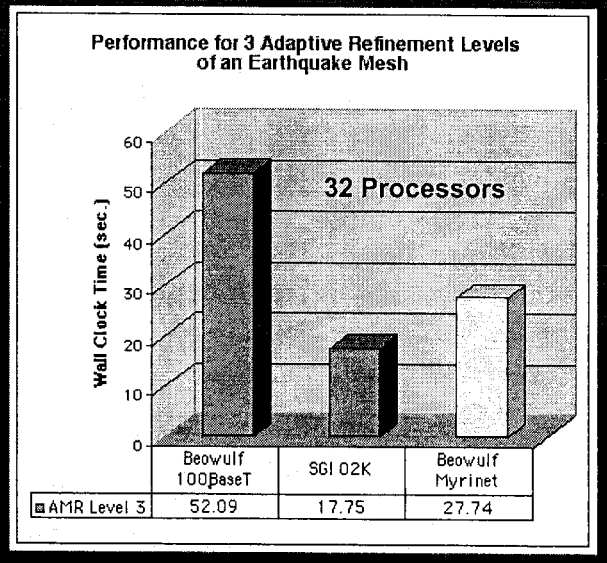


Myrinet Performance

- **Muzzle Refinement and Earthquake Generation Meshes**
  - Myrinet improved communication, but computation still holds back overall improvement
  - Perhaps Pentium III bandwidth to memory?

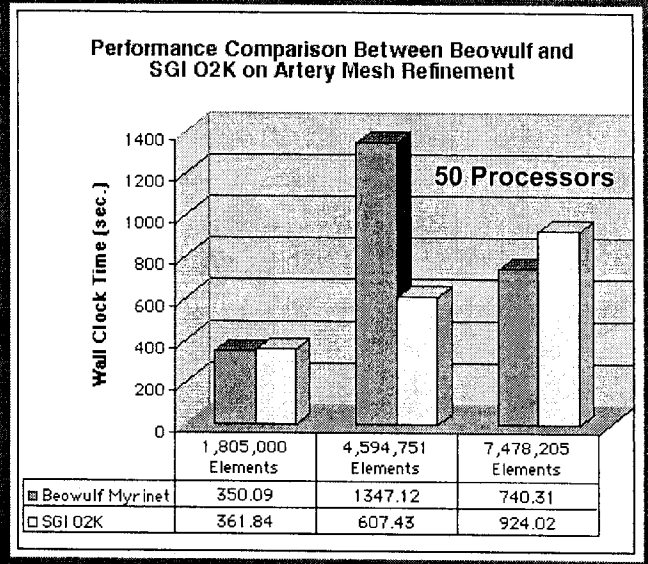


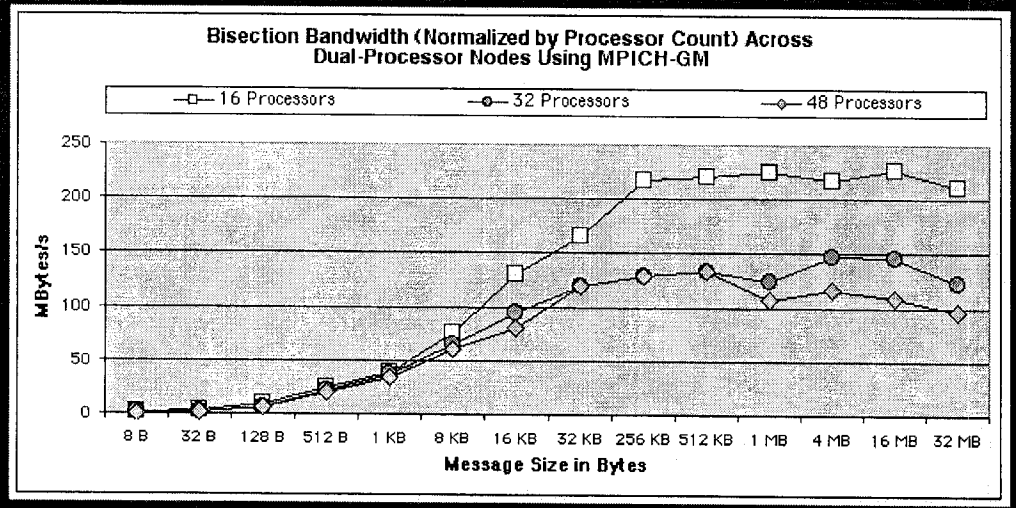


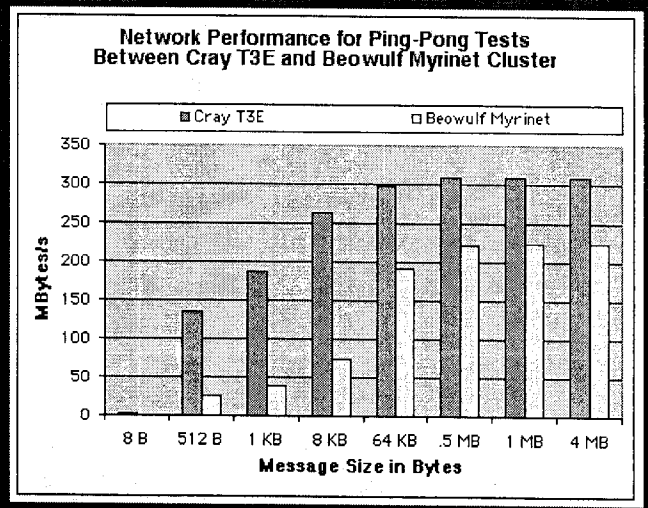


Myrinet Performance

- **Artery Refinement, Bisection, and the Cray T3E Network**
  - AMR performance varies based on partitioning, but Myrinet helps
  - Bisection bandwidth drops with increasing numbers of processors (no shared memory)
  - Cray T3E network remains the benchmark







Next Generation Features

Development is User-Driven

- Used for adaptive refinement of multi-scale meshes for active device modeling

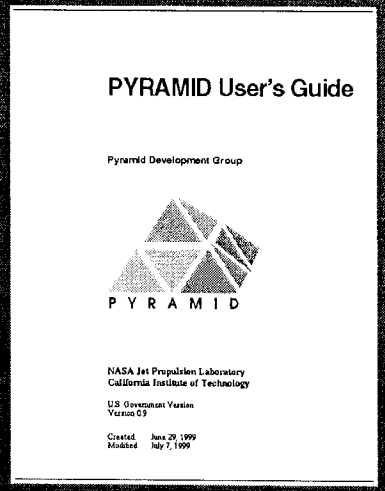
Additional Work Directions

- User-controllable boundary zone definition
- Interpolation methods among mesh levels
- Straightforward approaches for incorporating error estimation
- Coarsening

Demonstration Release

- hpc.jpl.nasa.gov/APPS/AMR

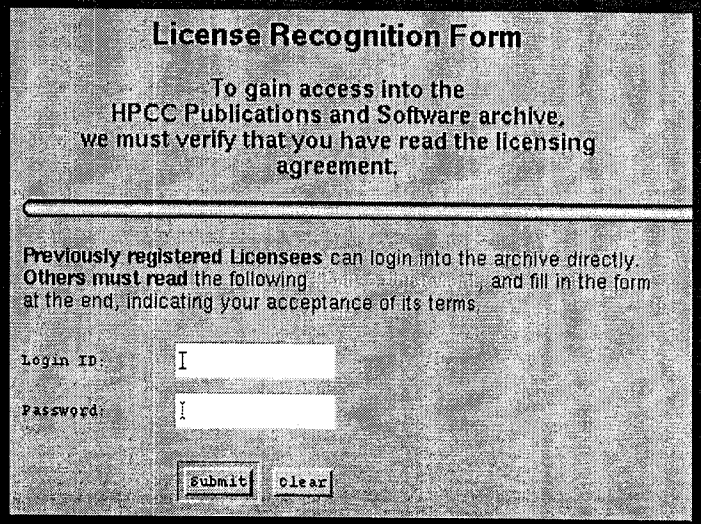
Note : Functionality is limited in demo release


PYRAMID User's Guide

Pyramid Development Group

PYRAMID

NASA Jet Propulsion Laboratory
California Institute of Technology

US Government Version
Version 0.9

Created June 29, 1999
Modified July 7, 1999