A Common Application Platform for SURAgrid (CAP)

-Mahantesh Halappanavar, John-Paul Robinson, Enis Afgane, Mary Fran Yafchalt and Purushotham Bangalore.
Contents:

- Introduction: Patterns
- Scientific Applications: Building Blocks
- Common Application Platform (CAP)
- Issues and Tools: GridMPI, MPICH-G2
- Conclusions & Future Work
Introduction: “Patterns”

- Algorithm Structures
- Support Structures
- Relationships
Introduction

- Problem Statement:
  
  “How to quickly grid-enable scientific applications to exploit SURAgrid infrastructure?”

- Our Goals for Grid-enablement:
  
  - Dynamic resource discovery
  - Collective resource utilization
  - Simple Job Management and Accounting
  - [NO] Minimal Programming Efforts
Basic Process

Problems → Algorithms → Programs → Implementation

- Algorithm Structures
- Supporting Structures
- Programming Environments
## Relationships: AS and SS

<table>
<thead>
<tr>
<th>SS</th>
<th>AS</th>
<th>Tasks</th>
<th>Data</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Task Parallelism</td>
<td>Divide &amp; Conquer</td>
<td>Geometric Decomposition</td>
</tr>
<tr>
<td>SPMD</td>
<td></td>
<td>** ****</td>
<td>***</td>
<td>****</td>
</tr>
<tr>
<td>Loop Parallelism</td>
<td>** ****</td>
<td>**</td>
<td>****</td>
<td></td>
</tr>
<tr>
<td>Master/Worker</td>
<td>** ****</td>
<td>**</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Fork/Join</td>
<td>**</td>
<td>** ****</td>
<td>**</td>
<td></td>
</tr>
</tbody>
</table>

Relationship between Supporting Structures (SS) patterns and Algorithm Structure (AS) patterns.
## Relationships: SS and PE

<table>
<thead>
<tr>
<th>SS</th>
<th>PE</th>
<th>OpenMP</th>
<th>MPI</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPMD</td>
<td>** ***</td>
<td>****</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Loop Parallelism</td>
<td>****</td>
<td>*</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Master-Worker</td>
<td>** **</td>
<td>***</td>
<td>***</td>
<td></td>
</tr>
<tr>
<td>Fork-Join</td>
<td>** ***</td>
<td>****</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationship between *Supporting Structures* (SS) patterns and *Programming Environments* (PE).
Important Observation

“This (SPMD) pattern is by far the most commonly used pattern for structuring parallel programs. It is particularly relevant for MPI programmers and problems using the Task Parallelism and Geometric Decomposition patterns. It has also proved effective for problems using the Divide and Conquer, and Recursive Data patterns.”

Single Program, Multiple Data (SPMD). This is the most common way to organize a parallel program, especially on MIMD computers. The idea is that a single program is written and loaded onto each node of a parallel computer. Each copy of the single program runs independently (aside from coordination events), so the instruction streams executed on each node can be completely different. The specific path through the code is in part selected by the node ID.
Scientific Applications

- What are they?
- How are they built?
Scientific Applications

- **Life Sciences:**
  - Bioinformatics, BioChemistry, …

- **Engineering:**
  - Aerospace, Civil, Mechanical, Environmental

- **Physics:**
  - QCD, Black Holes, …

The SCaLeS Reports: http://www.pnl.gov/scales/
Building Blocks

- OpenMP; MPI; BLACS, …
- Metis/ParMetis, Zoltan, Chaco, …
- BLAS and LAPACK, …
- ScaLapack, MUMPS, SuperLU, …
- PETSc, Aztec, …

“A core requirement of many engineering and scientific applications is the need to solve linear and non-linear systems of equations, eigensystems and other related problems.” – The Trilinos Project
ScaLAPACK

BLAS

LAPACK

PBLAS

BLACS

MPI/PVM/...

Global

Local

Communication routines targeting linear algebra operations.

Parallel BLAS.

Communication layer (message passing).

Level 1/2/3

Linear systems, least squares, singular value decomposition, eigenvalues.

platform specific

http://acts.nersc.gov/scalapack
PETSc: Portable, Extensible Toolkit for Scientific Computation

- Computation and Communication Kernels
  - MPI, MPI-IO, BLAS, LAPACK
- Profiling Interface
- PETSc PDE Application Codes
  - ODE Integrators
  - Visualization
  - Nonlinear Solvers, Unconstrained Minimization
  - Interface
  - Linear Solvers
  - Preconditioners + Krylov Methods
  - Grid Management
  - Object-Oriented Matrices, Vectors, Indices
- Profiling Interface
  - Computation and Communication Kernels
    - MPI, MPI-IO, BLAS, LAPACK

http://acts.nersc.gov/petsc
“Explicit message passing will remain the dominant programming model for the foreseeable future because of the huge investment in application codes.”

-Jim Tomkis, Bob Balance, and Sue Kelly, ASC PI Meeting, Nevada, Feb 2007
Common Application Platform

- Basic Architecture
- Building Blocks
Building Blocks

GridWay

the globus alliance

MPI-*

AMD
Sun
IBM
Intel
Dell

Smarter Choice
microsystems

Leap ahead

Leap ahead

Problems and Tools

- Problems
- Grid-MPI
- MPICH-G2
What is homogenous?

- Guarantees that the result of a particular sequence of *floating point operations* is the same no matter which *processor* is used,
- Floating point numbers are communicated exactly between homogenous machines, and
- OS, compiler and compiler options (-O2) do not alter the representation of floating point values.

Source: http://www.netlib.org/scalapack/
Load Balancing

- Resource Performance
- Network (latency and bandwidth)
- How best to model the load balancing problem?
  - Bipartite graphs?
  - Hypergraphs*
- Adapt the current tools?

*http://www.cs.odu.edu/~pothen/Courses/CS695/lecture10-catalyurek.ppt
Communication:

- **Network Addressing:**
  - Private addresses for compute nodes
  - Physical connections

- **Network Traffic Control:**
  - Traffic pattern not conducive for long distance
Landscape: Problems & Tools

- Authentication
- Heterogeneity
- Network Addressing
- Network Traffic

- Globus
- IMPI
- RSIP

- MPICH-G2
- GridMPI-2
- IMPI-Relay
- PSPacer

- MPI-1 + ~MPI-2
- Topology Awareness
- Job Management
- IntraCluster (Vendor)
- Checkpointing
GridMPI™ version 2.0

- Introduction
- Assessment
- Benchmarks
GridMPI 2.0 (Nov 2007)

- *GridMPI/YAMPI*: is a new implementation of MPI-1.2 and part of MPI-2.0 standard features.
- *IMPI support*: for inter-cluster communication.
- *Vendor MPI*: support for local communication.
- *PSPacer 2*: For network bandwidth control and traffic smoothing over long-distance and wide-bandwidth communications via software.
Basic Architecture:

- IMPI Server
  - SSH/RSH
  - IMPI Protocol

- Cluster 1:
  - Rank 0
  - Rank 1
  - YAMPI Protocol

- Cluster 2:
  - Rank 2
  - Rank 3
  - YAMPI Protocol
Global Addresses for each node

STEP – 1:

Cluster 1

Cluster 2
Global Addresses for each node

STEP – 2:

Cluster 1
- Process 0
- Host A
- Process 1
- Host B
- Process 2
- Host C

Cluster 2
- Host D
- Process 3
- Host E
- Process 4
- Host F
- Process 5

IMPI Server
IMPI Relay:

- A forwarding mechanism to bridge nodes in a cluster private address to global address.
- Forwards only data in the IMPI protocol, and not a general mechanism for NAT (Network Address Translation).
Private Addresses for some nodes

STEP – 1:

Cluster 1

Process
0
1
2

Host
A

Cluster 2

Process
0
1
2

Host
D
E
F

IMPI Server

IMPI-Relay
Private Addresses for some nodes

STEP – 1:

Cluster 1

Process

0
1
2

Host

A
B
C

IMPI-Relay

Cluster 2

Host

D
E
F

Process

IMPI Server

3
4
5
GridMPI: IntraCluster–SendRecv

Comparison of GridMPI and MPICH–2 for IMB SendRecv tests

MegaBytes per second

Byte Size $2^n$

Semilogy plot

Mileva: dual core Opterons with GigE interconnect
GridMPI: IntraCluster–AllReduce

Comparison of GridMPI and MPICH-2 for Allreduce benchmark tests

Semilogy plot

Mileva: dual core Opterons with GigE interconnect
GridMPI: IntraCluster–Bcast

Semilogy plot

Mileva: dual core Opterons with GigE interconnect
GridMPI: InterCluster–PingPong

Avg RTT for 64 bytes: Head-to-head: 0.188 ms & Compute-to-head: 0.253 ms
GridMPI: InterCluster–SendRecv

Avg RTT for 64 bytes: Head-to-head: 0.188 ms & Compute-to-head: 0.253 ms
PSPacer: Network Traffic

Bursty traffic can degrade performance due to packet losses

PSPacer adjusts the interval between packets, and it produces smoothed and stable traffic

SOURCE: http://www.gridmpi.org/
NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB2.3)
NPROCS=16  CLASS=B

SOURCE: http://www.gridmpi.org/mpicomparison.html
Assessment:

- **Strengths:**
  - MPI 1 and ~MPI 2 std features
  - Support for heterogeneity with IMPI
  - **IMPI-Relay**: Support for private IP addresses
  - Vendor MPI: special network platforms
  - **PSPacer**: Long-distance/Wide-bandwidth

- **Weaknesses:**
  - **Authentication**: No features yet
    - How about GSI-SSH? (how to make it work?)
  - **Robustness**: (or just our trial runs?)
MPICH-G2

- MPICH-G2
- MPI-flavor for Globus
- RSIP
Traditional Approach:

- Treat each node in the cluster as an individual machine
  - Setup is painful

- Use cluster file systems (NFS/AFS):
  - Still need to generate individual host certificates with special naming conventions
  - Setup GSI on each node: cfengine?

- Now build MPICH-G2
MPI-flavor for Globus 4.x

- Built Globus 4.0.5 with MPICH 1.2.7.

**Documentation**?
  - Setup the paths to *mpi* libraries
  - MPICH-G2 does not work with MPICH-2?
  - `./configure --with-mpi --with-flavor=mpicc64`

- Experience: not entirely successful – trouble with *globusrun*
Not explored: MPI-G

- MPI-G is a successor to MPICH-G2
- Developer: Brian Toonan
- http://wiki.ngs.ac.uk/index.php?title=MPIg #MPIg_and_Globus
Assessment:

- **Pros:**
  - Support for Globus Toolkit: Authentication
  - Support for Vendor MPI: (IBM p575)
  - Support for heterogeneity

- **Cons:**
  - Relay: Private IP addresses
  - *Not straightforward* to make it work on the compute nodes of a cluster
RSIP: Realm-Specific IP

- Alternative to NAT
  - Shared public IP + TCP/UDP Ports

- Experiences:
  - Easy to install and run the server
  - Minimum changes to applications (/etc/hosts)
  - Reuse ports that are open for Globus

- Issues with Client:
  - Download denied for Python client
  - Kernel loadable client is compatible with Linux 2.4
Conclusions

- Not a trivial problem

- *Powershift*!
  - Onus is now on System Administrators

- **Load Balancing**
  - Minimize communication costs
  - Limits: ScaLAPACK (Latency<500 ms)
  - Dynamic Redistribution of Work

- **Heterogeneous Environment**
  - Issues with floating-point operations
Future Work:

- **GridMPI:**
  - Integrate with GSI-SSH (?)
  - Cross campus experiments (PSPacer)

- **MPICH-G2:**
  - RSIP
  - MPI-flavor for Globus

- **GridWay and other co-scheduling tools**

- **Applications:** BioSim; Multiple Genome Alignment
Landscape: Issues & Tools

- Authentication
- Heterogeneity
- Network Addressing
- Network Traffic

- MPICH-G2
- GridMPI-2
- IMPI-Relay
- PSPacer

- MPI-1 + ~MPI-2
- Topology Awareness
- Job Management
- IntraCluster (Vendor)
- Checkpointing

RSIP

Globus

IMPI

THANK YOU!
Algorithm Structures

How to organize? *(Linear and Recursive)*

- **Organize by Tasks**
  - Task Parallelism
  - Divide and Conquer

- **Organize by Data Decomposition**
  - Geometric Decomposition
  - Recursive Data

- **Organize By Flow of Data**
  - Pipeline
  - Event-Based Coordination
Program Structures
- SPMD
- Master/Worker
- Loop Parallelism
- Fork/Join

Data Structures
- Shared Data
- Shared Queue
- Distributed Array
The Computing Continuum
## Benchmark: Allreduce (4 proc)

<table>
<thead>
<tr>
<th>#bytes</th>
<th>t_avg[usec]</th>
<th>#bytes</th>
<th>t_avg[usec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>398.60</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>403.88</td>
<td>4</td>
<td>113.49</td>
</tr>
<tr>
<td>8</td>
<td>198.43</td>
<td>8</td>
<td>113.25</td>
</tr>
<tr>
<td>16</td>
<td>201.96</td>
<td>16</td>
<td>114.29</td>
</tr>
<tr>
<td>32</td>
<td>203.04</td>
<td>32</td>
<td>115.52</td>
</tr>
<tr>
<td>64</td>
<td>207.39</td>
<td>64</td>
<td>115.66</td>
</tr>
<tr>
<td>128</td>
<td>210.51</td>
<td>128</td>
<td>120.93</td>
</tr>
<tr>
<td>256</td>
<td>220.92</td>
<td>256</td>
<td>126.24</td>
</tr>
<tr>
<td>512</td>
<td>235.64</td>
<td>512</td>
<td>139.81</td>
</tr>
<tr>
<td>1024</td>
<td>368.36</td>
<td>1024</td>
<td>166.32</td>
</tr>
<tr>
<td>2048</td>
<td>2490.82</td>
<td>2048</td>
<td>233.27</td>
</tr>
<tr>
<td>4096</td>
<td>133139.39</td>
<td>4096</td>
<td>706.29</td>
</tr>
<tr>
<td>8192</td>
<td>1127.50</td>
<td>8192</td>
<td>593.81</td>
</tr>
<tr>
<td>16384</td>
<td>149183.78</td>
<td>16384</td>
<td>788.96</td>
</tr>
<tr>
<td>32768</td>
<td>3189.82</td>
<td>32768</td>
<td>1232.43</td>
</tr>
<tr>
<td>65536</td>
<td>56856.32</td>
<td>65536</td>
<td>2603.80</td>
</tr>
<tr>
<td>131072</td>
<td>7076.99</td>
<td>131072</td>
<td>6080.06</td>
</tr>
<tr>
<td>262144</td>
<td>20679.20</td>
<td>262144</td>
<td>11155.54</td>
</tr>
<tr>
<td>524288</td>
<td>24359.40</td>
<td>524288</td>
<td>25643.52</td>
</tr>
<tr>
<td>1048576</td>
<td>66875.02</td>
<td>1048576</td>
<td>39201.83</td>
</tr>
<tr>
<td>2097152</td>
<td>235458.39</td>
<td>2097152</td>
<td>81098.47</td>
</tr>
<tr>
<td>4194304</td>
<td>189229.39</td>
<td>4194304</td>
<td>154493.15</td>
</tr>
</tbody>
</table>

**GridMPI-2**

**MPICH-2**
# Benchmarking Barrier
# #processes = 2
# ( 7 additional processes waiting in MPI_Barrier)
#repetitions t_min[usec] t_max[usec] t_avg[usec]
1000 62.38  62.41  62.40

# Benchmarking Barrier
# #processes = 4
# ( 5 additional processes waiting in MPI_Barrier)
#repetitions t_min[usec] t_max[usec] t_avg[usec]
1000 115.19 115.22 115.20

# Benchmarking Barrier
# #processes = 8
# ( 1 additional process waiting in MPI_Barrier)
#repetitions t_min[usec] t_max[usec] t_avg[usec]
1000 173.61 173.71 173.65

# Benchmarking Barrier
# #processes = 9
#repetitions t_min[usec] t_max[usec] t_avg[usec]
1000 220.12 220.19 220.15

---

GridMPI-2

---

MPICH-2