A Common Application Platform (CAP) for SURAgrid

-Mahantesh Halappanavar

Old Dominion University
Office of Computing and Communications Services

SURAggrid

SURAggrid Conference Call, 13 August, 2007
Grid-Enabling Applications

“True grid enabling should take advantage of the distributed resources by using the resources collectively, or by dynamically discovering resources.” – Purushotham Bangalore/Barry Wilkinson

“Build it, and they will come”
## Resources

<table>
<thead>
<tr>
<th>Circa</th>
<th>TFlops</th>
<th>CPUs</th>
<th>Members</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>09-2005</td>
<td>1</td>
<td>490</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>08-2006</td>
<td>3</td>
<td>912</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>07-2007</td>
<td>14</td>
<td>2,581</td>
<td>13</td>
<td>17</td>
</tr>
</tbody>
</table>

Source: Art Vandenberg, GSU.
Applications?

- ~10 all along….
- “Build it, and they will come ???”

How to attract more applications?

- Minimize effort to grid-enable apps.
Identifying Patterns

- Algorithm Structures
- Support Structures
- Relationships
Algorithm Structures

How to organize? (Linear and Recursive)

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

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

3. Organize By Flow of Data
   - *Pipeline*
   - *Event-Based Coordination*
Support Structures

Program Structures
1. **SPMD**
2. **Master/Worker**
3. **Loop Parallelism**
4. **Fork/Join**

Data Structures
1. **Shared Data**
2. **Shared Queue**
3. **Distributed Array**
### Relationships: AS and SS

<table>
<thead>
<tr>
<th>SS</th>
<th>Task Parallelism</th>
<th>Divide &amp; Conquer</th>
<th>Geometric Decomposition</th>
<th>Recursive Data</th>
<th>Pipeline</th>
<th>Event-based Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPMD</td>
<td>** ****</td>
<td>***</td>
<td>****</td>
<td>**</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Loop Parallelism</td>
<td>* ****</td>
<td>**</td>
<td>***</td>
<td>**</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Master/Worker</td>
<td>* ****</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Fork/Join</td>
<td>**</td>
<td>* ****</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).
“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.
The Computing Continuum

Loosely Coupled

Special Purpose
“SETI / Google”

“Grids”

Clusters

Highly Parallel

Tightly Coupled

Java

MPI

OpenMP

ICL
Scientific Applications

- What are they?
- How are they built?
The Usual Suspects

- Life Sciences:
  - Biology, Chemistry, ...

- 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

- Linear systems, least squares, singular value decomposition, eigenvalues.
- Communication routines targeting linear algebra operations.
- Parallel BLAS.
- Communication layer (message passing).
- Global and Local
- Platform specific

http://acts.nersc.gov/scalapack
PETSc

PORTABLE, EXTENSIBLE TOOLKIT FOR SCIENTIFIC COMPUTATION

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
“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 Meeing, Nevada, Feb 2007
SURAgrid

- Available Infrastructure
- Networking
SURAggrid Participants
(As of October 2006)
### Shared Computing Resources

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>System</th>
<th>CPUs</th>
<th>Peak GFlops</th>
<th>Memory GBytes</th>
<th>Disk GBytes</th>
<th>Test Status</th>
<th>Current Load</th>
<th>Current Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSrocks</td>
<td>Georgia State University</td>
<td>ACS Dell Dimension 3000 Rocks Cluster</td>
<td>6</td>
<td>37</td>
<td>6</td>
<td>240</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>AGLT2_LM</td>
<td>University of Michigan</td>
<td>Michigan SURAggrid Gatekeeper</td>
<td>1.6</td>
<td>32</td>
<td>8</td>
<td>80</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Banderia</td>
<td>The University of Texas at Austin</td>
<td>TACC DGC Cluster</td>
<td>8</td>
<td>48</td>
<td>8</td>
<td>144</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Calclab</td>
<td>Texas A&amp;M University</td>
<td>Calclab Optiplex Cluster</td>
<td>306</td>
<td>1971</td>
<td>306</td>
<td>3500</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>CANBlade</td>
<td>University of Louisiana at Lafayette</td>
<td>IBM BladesCenter HS20 Cluster</td>
<td>12</td>
<td>67</td>
<td>20</td>
<td>25</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>CISC</td>
<td>George Mason University</td>
<td>GMU CISC Condor Pool</td>
<td>14</td>
<td>20</td>
<td>22</td>
<td>400</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Grid04</td>
<td>University of Virginia</td>
<td>Dell Rocks Linux Cluster</td>
<td>40</td>
<td>240</td>
<td>60</td>
<td>400</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Grid11</td>
<td>University of Virginia</td>
<td>HP Linux Cluster</td>
<td>5</td>
<td>26</td>
<td>5</td>
<td>325</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Janus</td>
<td>Georgia State University</td>
<td>GSU IBM Power5+, p575</td>
<td>160</td>
<td>1082.6</td>
<td>320</td>
<td>4000</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>ksucsgrid01</td>
<td>Kennesaw State University</td>
<td>Kennesaw Dell PowerEdge Xeon Cluster x66_64</td>
<td>14</td>
<td>70</td>
<td>21</td>
<td>490</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Mileva</td>
<td>Old Dominion University</td>
<td>Sun Fire X2100 Dual Core Opteron Cluster</td>
<td>8</td>
<td>24</td>
<td>8</td>
<td>25</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Pelican</td>
<td>Louisiana State University</td>
<td>IBM SURAggrid Frame at LSU</td>
<td>368</td>
<td>2797</td>
<td>960</td>
<td>19000</td>
<td>Click Here</td>
<td></td>
<td>14R-0Q-10</td>
</tr>
<tr>
<td>Scoops</td>
<td>University of Alabama at Huntsville</td>
<td>ADCIRC-2006 Cluster</td>
<td>8</td>
<td>48</td>
<td>8</td>
<td>100</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
<tr>
<td>Tensor</td>
<td>Texas A&amp;M University</td>
<td>Tensor Opteron Cluster</td>
<td>256</td>
<td>718</td>
<td>384</td>
<td>3000</td>
<td>Click Here</td>
<td></td>
<td>R-Q-Q</td>
</tr>
</tbody>
</table>

**Total:** 1221 7148.6 2160 31729
Network Infrastructure

National LambdaRail Architecture
Common Application Platform

- Basic Architecture
- Building Blocks
- Issues
Basic Architecture

Browser

SURAgird Portal

Command-Line (GSISSH, etc.)

SiteA

Site1

MetaScheduler (MS)

Site2

Site3

A

B
Building Blocks

GridWay

MPICH-G2

g the globus alliance
Sun Microsystems
Intel
Linux
Sun Microsystems
AIX
Microsoft
AMD
Sun Microsystems
IBM
Intel
Dell
Issues: Load Balancing

- Non-uniform resources
- Non-uniform network (latency and bandwidth)
- How best to model the problem?
  - Bipartite graphs
  - Hypergraphs
Issues: Heterogeneous Platform

What is homogenous?

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

Source: http://www.netlib.org/scalapack/
Heterogeneous Platform

- Floating-point numbers depend on
  - Architecture (x86-64/PowerPC)
  - OS (Linux/AIX)
  - Compilers (GCC/Intel/Sun)

- Other issues?
  - We might not know until we have one!
Issues:

- Checkpointing
  - Detect errors
  - Automatic relocation?
  - Checkpointing and Rollbacks

- Compute Nodes:
  - Routing
  - Host certificates
Proposed Benchmark

- HP LinPack:
  - Hardware
  - OS
  - Compiler
  - Numerical libraries (BLAS/LAPACK)
  - Globus, MPICH-G2, GridWay
  - Performance
Conclusions

- **Powershift**: Onus on Sys Admins
- Load balancing
  - Minimize communication costs
  - ScaLAPACK: latency < 500 ms
- Dynamic redistribution of work
  - Zoltan
- Heterogeneous Environment
  - Issues
Thank You!

“There is no free lunch”

... well, maybe, there is?
ScaLAPACK: 2D Block-Cyclic Distribution

5x5 matrix partitioned in 2x2 blocks

\[
\begin{pmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
  a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
  a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
  a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\
  a_{51} & a_{52} & a_{53} & a_{54} & a_{55}
\end{pmatrix}
\]

2x2 process grid point of view

\[
\begin{pmatrix}
  a_{11} & a_{12} & a_{15} \\
  a_{21} & a_{25} \\
  a_{51} & a_{52} & a_{55} \\
  a_{41} & a_{45} \\
  a_{31} & a_{35} \\
  a_{23} & 1 & a_{24} \\
  a_{53} & a_{54} \\
  a_{43} & a_{44}
\end{pmatrix}
\]

http://acts.nersc.gov/scalapack/hands-on/datadist.html
2D Block-Cyclic Distribution

\[
\begin{bmatrix}
1.1 & 1.2 & 1.3 & 1.4 & 1.5 \\
-2.1 & 2.2 & 2.3 & 2.4 & 2.5 \\
-3.1 & -3.2 & 3.3 & 3.4 & 3.5 \\
-4.1 & -4.2 & -4.3 & 4.4 & 4.5 \\
-5.1 & -5.2 & -5.3 & -5.4 & 5.5
\end{bmatrix}
\]

CALL BLACS_GRIDINFO (ICTXT, NPROW, NPCOL, MYROW, MYCOL)

IF (MYROW.EQ.0 .AND. MYCOL.EQ.0) THEN
A(1) = 1.1; A(2) = -2.1; A(3) = -5.1;
A(1+LDA) = 1.2; A(2+LDA) = 2.2; A(3+LDA) = -5.2;
A(1+2*LDA) = 1.5; A(2+3*LDA) = 2.5; A(3+4*LDA) = -5.5;
ELSE IF (MYROW.EQ.0 .AND. MYCOL.EQ.1) THEN
A(1) = 1.3; A(2) = 2.3; A(3) = -5.3;
A(1+LDA) = 1.4; A(2+LDA) = 2.4; A(3+LDA) = -5.4;
ELSE IF (MYROW.EQ.1 .AND. MYCOL.EQ.0) THEN
A(1) = -3.1; A(2) = -4.1;
A(1+LDA) = -3.2; A(2+LDA) = -4.2;
A(1+2*LDA) = 3.5; A(2+3*LDA) = 4.5;
ELSE IF (MYROW.EQ.1 .AND. MYCOL.EQ.1) THEN
A(1) = 3.3; A(2) = -4.3;
A(1+LDA) = 3.4; A(2+LDA) = 4.4;
END IF

CALL PDGESVD (JOBU, JOBVT, M, N, A, IA, JA, DESCA, S, U, IU, JU, DESCU, VT, IVT, JVT, DESCVT, WORK, LWORK, INFO)

LDA is the leading dimension of the local array
Array descriptor for A (contains information about A)