Project AutoMate
GridARM: Autonomic Runtime
Management of Grid Applications

S. Chandra, The AutoMate Group
The Applied Software Systems Laboratory
Rutgers, The State University of New Jersey
http://automate.rutgers.edu

CAIP Autonomic Computing Tutorial/Workshop
June, 2003

Acknowledgements:
Xiaolin Li, Shweta Sinha, Hailan Zhu – TASSL, Rutgers University
Johan Steensland, Jaideep Ray – Sandia National Laboratory
Ravi Sambasivan – CACR, California Institute of Technology

GridARM: Motivation

• Realistic, physically accurate computational modeling
  – Large computational requirements
  – Multiphase, heterogeneous, dynamic ⇒ complex interactions
  – Software/systems engineering/programmability

• Adaptive and interactive simulations on the “Grid”
  – Adaptive: resolution, algorithms, scheduling, execution
  – Interactive: peer interactions between computational objects and users, data, resources

• Developing applications to utilize and exploit the Grid remains a significant challenge
  – Complexity, heterogeneity, dynamism making programming environments unmanageable, brittle and insecure
  – Requires fundamental changes in how applications are formulated, composed and managed
  – Resonance: heterogeneity and dynamics must match and exploit the heterogeneous and dynamic nature of the Grid

• Alternative approach – autonomic computing
  – Based on strategies used by biological systems
  – Self-defining, contextually aware, self-healing, self-configuring, self-optimizing, open
GridARM: Overview

- **Autonomic runtime management framework**
  - Reactively and proactively optimize SAMR execution using current system and application state, online predictive performance models, and agent based control network
  - Builds on “vGrid” concept: abstraction of a virtual Grid, similar in convenience as “virtual memory”

- **GridARM infrastructure services**
  - Collect and characterize operational, functional, and control aspects of SAMR application to define autonomic components
  - Decompose application into “natural regions” (NRs) and NRs into virtual computational units (VCUs)
  - Apply allocation and scheduling strategies to map VCUs onto virtual resource units (VRUs) and then to physical Grid resources

GridARM: Overview

- **Terminology**
  - Natural Region: region of relatively homogeneous activity in the application domain and may span various SAMR hierarchy levels
  - VCU: application work unit consisting of computation patches at one or multiple levels that is scheduled by GridARM framework
  - VRU: schedulable unit in the virtual Grid to which a VCU can be mapped – individual resource or collection of Grid resources

- **GridARM components**
  - Context-aware services: monitor system and application state
  - Deduction engine: characterize state and define objective function
  - Autonomic runtime manager: hierarchical partitioning based on objective function, scheduling and mapping of VCUs onto VRUs, managing and tuning application execution
GridARM: Autonomic Application Management

Grid Resource Hierarchy
- Application Domain Hierarchy
- Virtual Grid Resource
- Autonomic Runtime Manager (ARM)

Loop for each level of Grid/Application hierarchy

GridARM Monitoring
- Self-observation, Context-awareness
- System states (CPU, Memory, Bandwidth, Availability etc.)
- Application states (Computation/Communication, Ratio, Nature of Applications, Application Dynamics)

GridARM Deduction
- Self-adaptation, Self-optimization, Self-healing
- Identify and characterize natural regions
- Define objective functions and management strategy
- Define VCU

GridARM Execution
- Partition, Map and Tune

NR1
NR2
NR5
VCU1
VCU2
VCUn
VCUi
VCUj
Self-learning

GridARM Monitoring
- System states (CPU, Memory, Bandwidth, Availability etc.)
- Application states (Computation/Communication, Ratio, Nature of Applications, Application Dynamics)

Structured Adaptive Mesh-Refinement

Adaptive Mesh Refinement
- Start with a base coarse grid with minimum acceptable resolution
- Tag regions in the domain requiring additional resolution, cluster the tagged cells, and fit finer grids over these clusters
- Proceed recursively so that regions on the finer grid requiring more resolution are similarly tagged and even finer grids are overlaid on these regions
- Resulting grid structure is a dynamic adaptive grid hierarchy

Structured Adaptive Mesh-Refinement

The Berger-Oliger Algorithm
Recursive Procedure Integrate(level)
If (RegridTime) Regrid
Step:
1. Integrate (level)
2. Update(level, level + 1)
End if
End Recursion
level = 0
Integrate(level)

Sample Grid-based SAMR Application

- Simulation of flames
  - Combustion application at Sandia National Lab
  - Ignition of H₂-Air mixture in non-uniform temperature field with 3 “hot-spots” at 1000K
  - Mass-fraction plots of various radicals (OH, H, H₂O, HO₂)
  - High dynamism and space-time heterogeneity
• SAMR partitioning, load-balancing and scheduling
  – Partitioning Scheme
    • “Best” partitioning based on application/system configuration and current application/system state
  – Granularity
    • Patch size, AMR efficiency, comm./comp. ratio, overhead, node-performance, load-balance, ...
  – Number of processors/Load per processor
  – Dynamic allocations/configuration/management
  – Hierarchical decomposition using dynamics processor groups
  – Communication optimizations/latency tolerance/multithreading
  – Availability, capabilities, and state of system resources
  • SNMP, NWS

**ARMaDA: A GridARM Prototype**

**ARMaDA: Autonomic Runtime Management**

**Reactive System Sensitive Partitioning**

• Relative Capacity of node \( k \) represented as:

\[
C_k = w_p^P_k + w_m^M_k + w_b^B_k
\]

Where \( w_p, w_m, w_b \) are weights associated with relative CPU, Memory, and Bandwidth availabilities, respectively, where

\[
w_p^P + w_m^M + w_b^B = 1
\]

- **Capacity Calculator**
  -CPU
  -Memory
  -Bandwidth
- **Resource Monitoring Tool**
  -Partitions
  -Weights
- **Application**
  -Heterogeneous System Sensitive Partitioner
**System Sensitive Partitioning: Results**

- Application: RM3D (Richtmyer-Meshkov 3-D CFD)
  - 128x32x32 base grid, 3 levels, refinement every 4 time-steps
- System
  - Heterogeneous Linux cluster (64 nodes)
  - Synthetic load generators

![Graph showing time vs. total time, compile, regrid, sync, and recompose times for static and dynamic partitioning]

**Application Aware Partitioning**

- Application State Monitoring and Characterization
  - Implements mechanisms for abstracting current application state at runtime
  - State characterized in terms of pre-defined metric such as computational requirements, application dynamics, nature of adaptation, etc.
- Deductive Engine and Policy Repository
  - Provides an association for mapping application characteristics to metric-based state octants
  - Policies define the association of octants to partitioners
- Adaptation (Meta-partitioner)
  - Dynamically configures appropriate partitioner and associated partitioning parameters at runtime

**Application Runtime Characterization**

- Computation/Communication requirements ("CCratio")
  \[ \text{CCratio} = \frac{\sum \text{(Volume of horizons boxes)}}{\sum \text{(Surface area of bounding boxes)}} \]
  - high CCratio ⇒ more computation, low CCratio ⇒ more communication
- Application Dynamics ("Dynamics")
  - Dynamics = Size of (Current state boxes - Previous state boxes) / Domain volume
  - high value ⇒ lesser dynamics, low value ⇒ high activity dynamics
- Nature of Adaptation ("Adapt")
  - Adapt = Volume of coarse level boxes / Number of coarse level boxes
  - high Adapt ⇒ scattered adaptation, low Adapt ⇒ localized adaptation
Application Aware Partitioning: Results

- RM2D, 64 processors on “Blue Horizon”, 60 iterations
  - base grid 128*32, 3 levels, RF = 2, regrid every 4 time-steps
- 4 test cases for partitioning configurations

- Overhead minimal: 0.415 sec state sensing time
- Speedup improvement for ARMaDA adaptive partitioner
  - 4.66% over pBD-ISP, 11.32% over G-MISP+SP, 27.88% over SFC

Partitioning Variants and Optimizations

<table>
<thead>
<tr>
<th>Higher-level</th>
<th>HPA</th>
<th>LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower-level</td>
<td>GPA</td>
<td>BPA</td>
</tr>
<tr>
<td>Common Basis</td>
<td>SFC</td>
<td>CGDS</td>
</tr>
</tbody>
</table>

HPA: Hierarchical Partitioning Algorithm
LPA: Level-based Partitioning Algorithm
GPA: Greedy Partitioning Algorithm
BPA: Binpack Partitioning Algorithm
IPA: Iterative Partitioning Algorithm
SFC: Space-filling Curve
CGDS: Composite Grid Distribution Strategy

Level-based Partitioning Algorithm (LPA)

- Patch workload at finer levels of SAMR grid hierarchy considerably greater than work at coarse level
  - \( W_{lev} = W_{coarse} \cdot RF^{{lift(DIM)}} \) for \( lev \leq L \)
  - \( W_{coarse} \) is a finer level out of total L levels in hierarchy with refinement factor RF and application rank DIM

- Unequal computation and synchronization delays for different patch level depths causes processor “wait times”
- LPA accumulates patches at same refinement level depth and partitions workload at each level across processors
- Results in more patches due to level-based partitioning but improves synchronization time for SAMR application
Binpack Partitioning Algorithm (BPA)

- Load-balancing scheme for SAMR partitioning phase
  - Initially allocates computational workload among processors as long as processor capacity does not exceed threshold
  - Patches with workload exceeding threshold split into smaller parts
  - For remaining work, BPA uses a "best-fit approach" to assign patch to appropriate processor
  - If "best-fit" fails, then patch assigned to processor with maximum available capacity

- Several patch divisions result in high number of patches
- Improved load-balance results in lower synchronization time and faster SAMR application execution

LPA and BPA: Results

- RM3D: 64 processors on IBM SP2 "Blue Horizon"
  - 128^3^2^2 base grid, 3 levels, 100 iterations, factor 2 space-time refinement with global clustering, regrid every 8 steps

Effect of Partitioning Optimizations on RM3D Performance for 64 processors on Blue Horizon

Hierarchical Partitioning Algorithm (HPA)

- A general hierarchical structure of processor groups
**Sequence Diagram for HPA**

- Process or 1 (Group 1 Master)
- Process or 2 (Group 1)
- Process or 3 (Group 2 Master)
- Process or 4 (Group 2)

**Synchronization in group**
- to get global view in group
- Synchronization among masters to exchange local domains to get global domain
- Partition among masters to get local domain

**Broadcast local domain to group**
- Computation

**HPA: Results**

**Execution Time of RM3D application**
- (100 steps, size=128x32x32)

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>GPA</th>
<th>LPA</th>
<th>HPA</th>
<th>HPA+LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Communication Time of RM3D application**
- (100 steps, size=128x32x32)

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>GPA</th>
<th>LPA</th>
<th>HPA</th>
<th>HPA+LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Experimental Setup:**
- IBM SP2 SMP cluster
- (8 processors per node)
- at San Diego Supercomputer Center

**Summary**

- Adaptive and interactive simulations enable accurate solutions of realistic models of complex phenomena
- GridARM: autonomic runtime management framework
  - Application/system monitoring and characterization
  - Adaptive application control and management framework
    - reactive/proactive, application/system sensitive
    - Integrated application software development
    - autonomous components and dynamic compositions
- Overall goal
  - Manage dynamism and space-time heterogeneity of SAMR applications and Grid resources
  - Support efficient and scalable execution for Grid-based SAMR
References

• Engineering an Autonomic Partitioning Framework for Grid-based SAMR Applications.
• Dynamic Load Partitioning Strategies for Managing Data of Space and Time Heterogeneity in Parallel SAMR Applications.
  - X. Li and M. Parashar, Euro-Par, August 2003 (to appear)
• vGrid: A Framework for Building Autonomic Applications.
  - B. Khargharia, S. Hariri and M. Parashar, CLADE, June 2003
• Self Adapting, Self Optimizing Runtime Management of GDI Applications using PRAGMIL.
• Adaptive Runtime Management of SAMR Applications.
  - S. Chandra, Y. Zhang, S. Soha, J. Yang, M. Parashar and S. Hariri, HPC, December 2002
  - S. Chandra and M. Parashar, PKCS, November 2002
• Information, publications – www.caip.rutgers.edu/TASSL