Self-Adapting, Self-Optimizing Runtime Management of Grid Applications using PRAGMA

S. Chandra, X. Li, H. Zhu and M. Parashar
The Applied Software Systems Laboratory
ECE/CAIP, Rutgers University

J. Yang, Y. Zhang, S. Rao and S. Hariri
HPDC Laboratory
ECE, University of Arizona
Outline

• Realistic scientific and engineering simulations on the Grid.
  – adaptive mesh refinement (AMR)
• PRAGMA: proactive and reactive runtime management of SAMR applications
  – context aware, reactive/proactive partitioning, load-balancing and scheduling
    • application sensitive partitioning
    • system sensitive partitioning
    • performance functions
• Towards autonomic computational science and engineering
  – context aware, self-configuring, adapting, optimizing, healing, anticipatory, ...
  – vGrid – Autonomic runtime management framework for Grid applications
    • AutoMate, Autonomia
Computational Modeling of Physical Phenomenon

- Realistic, physically accurate computational modeling
  - Large computation requirements
    - e.g. simulation of the core-collapse of supernovae in 3D with reasonable resolution \( (500^3) \) would require \( \sim 10-20 \) teraflops for 1.5 months (i.e. \( \sim 100 \) Million CPUs!) and about 200 terabytes of storage
    - e.g. turbulent flow simulations using active flow control in aerospace and biomedical engineering requires \( 5000 \times 1000 \times 500 = 2.5 \times 10^9 \) points and approximately 107 time steps, i.e. with 1GFlop processors requires a runtime of \( \sim 7 \cdot 10^6 \) CPU hours, or about one month on 10,000 CPUs! (with perfect speedup). Also with 700B/pt the memory requirement is \( \sim 1.75TB \) of run time memory and \( \sim 800TB \) of storage.
  - Coupled, multiphase, heterogeneous, dynamic
    - multi-physics, multi-model, multi-resolution, 
  - Complex interactions
    - application – application, application – resource, application – data, application – user, 
  - Software/systems engineering/programmability
    - volume and complexity of code, community of developers, 
      - scores of models, hundreds of components, millions of lines of code, 

Computational Modeling and the Grid

• The Computational Grid
  – Potential for aggregating resources
    • computational requirements
  – Potential for seamless interactions
    • new applications formulations

• Developing application to utilize and exploit the Grid remains a significant challenge
  – The problem: a level of complexity, heterogeneity, and dynamism for which our programming environments and infrastructure are becoming unmanageable, brittle and insecure
    • System size, heterogeneity, dynamics, reliability, availability, usability
    • Currently (typically) proof-of-concept demos by “hero programmers”
  – Requires fundamental changes in how applications are formulated, composed and managed
    • Breaks current paradigms based on passive components and static compositions
    • autonomic components and their dynamic composition, opportunistic interactions, virtual runtime, …
    – Resonance - heterogeneity and dynamics must match and exploit the heterogeneous and dynamic nature of the Grid

• Autonomic, adaptive, interactive simulations and the Grid offer the potential for such simulations
  – Autonomic: context aware, self configuring, self adapting, self optimizing, self healing,…
  – Adaptive: resolution, algorithms, execution, scheduling, …
  – Interactive: peer interactions between computational objects and users, data, resources, …
PRAGMA Research Objectives

- An adaptive runtime infrastructure for enabling dynamically adaptive simulations on distributed, heterogeneous and dynamic execution environments
  - reactively and proactively manage and optimize application execution using current system and application state, predictive models for system behavior and application performance, and an agent based control network

- Research components include:
  - models/mechanisms for monitoring and characterizing the state of adaptive applications and abstracting their current computational, communication and storage requirements
  - performance predictions functions and function composition to anticipate the operations and expected performance of applications for a given workload and system configuration
  - active control network combining application sensors and actuators and application management agents for managing and optimizing application execution
  - software design methodology to support the development of adaptive software systems capable of actively managing and exploiting system/network heterogeneity and dynamism

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Self-adapting, Self-optimizing Runtime Management in PRAGMA

- **Monitoring**
  - Detecting conditions affecting application execution, e.g. increased computational/network load, low available memory, or software/hardware failures

- **Analysis**
  - Encapsulates the adaptation policy, application reconfiguration strategy, and required system resources

- **Adaptation**
  - Once the appropriate adaptation/optimization strategy is identified, an adaptation process is initiated

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
PRAGMA Research Activities

• Adaptive runtime infrastructure that supports self-managing, self-adapting and self-optimizing applications on the Grid
  – *Runtime sensing of current system and application state*
  – *Anticipate system and application behavior*
  – *Proactively and reactively adapt application execution*

• Application-sensitive adaptation
  – *Characterizes and uses current application state*
  – *Determines resource allocation, partitioning/mapping of application components, load-balancing and communication mechanisms*

• System-sensitive adaptation
  – *Driven by system state and system performance predictions*
  – *Enables setting of application granularity, communication strategies based on bandwidth, and nature of refinements based on availability and “health” of computing elements*
Structured Adaptive Mesh-Refinement (SAMR)

**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

**The Berger-Oliger Algorithm**

Recursive Procedure `Integrate(level)`

- If `(RegridTime)` Regrid
- Step $\Delta t$ on all grids at level “level”
- If `(level + 1) exists`
  - Integrate `(level + 1)`
  - Update`(level, level + 1)`

End if

End Recursion

level = 0
Integrate(level)

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Application Challenges

- Applications exhibit multiple scales of length and time
  - use Adaptive Mesh Refinement (AMR) with multiple independent time steps (MIT)

- Computational requirements vary dynamically at runtime
  - Dynamic Partitioning Support
  - Adaptive Communication Support
  - Dynamic Application Configuration Support

- Richtmyer Meshkov Instability (3D) (R. Samtaney, ASCI, CIT)
  - Air-SF6 interface with single harmonic perturbation.
  - Incident shock Mach number (M=1.5), 1024x128x128, 3 levels, 2K PE’s
  - Advantages of adaptivity: Time: ~ 15% Memory: ~25%
SAMR Grid Structures: Illustrations

Time Step

0
40
80
120
160
182

Legend
Level 0: Level 1: Level 2: Level 3: Level 4:
A Selection of SAMR Application Enabled

- Blast wave in the presence of a uniform magnetic field – 3 levels of refinement. (Zeus + GrACE + Cactus, P. Li, NCSA, UCSD)
- Multi-block grid structure and oil concentrations contours (IPARS, M. Peszynska, UT Austin)
- Mixture of H2 and Air in stoichiometric proportions with a non-uniform temperature field (GrACE + CGA, Jaideep Ray, SNL, Livermore)
- Richtmyer-Meshkov - detonation in a deforming tube - 3 levels. Z=0 plane visualized on the right (VTF + GrACE, R. Samtaney, CIT)
Run-time Management and Optimization for Dynamic (SAMR) Applications

• Partitioning, load-balancing and scheduling of SAMR applications.
  – **Partitioning Scheme**
    • “Best” partitioning based on application/system configuration and current application/system state
      – G-MISP+SP, pBD-ISP, SFC (Vampire, GrACE, Zoltan, ParMetis, …)
  – **Granularity**
    • patch size, AMR efficiency, comm./comp. ratio, overhead, node-performance, load-balance, …
  – **Number of processors/Load per processor**
    • Dynamic allocations/configuration/management
      – 1000+ processor from the beginning or “on-demand”
  – **Hierarchical decomposition using dynamics processor groups**
  – **Communication optimizations/latency tolerance/multithreading**
  – **Availability, capabilities, and state of system resources**
    • SNMP, NWS
PRAGMA Runtime Management Framework

- **System Characterization and Abstraction Component**
  - Models/mechanisms for monitoring and characterizing the state of adaptive applications and abstracting their current computational, communication and storage requirements
  - Runtime decisions on selection and configuration of computing elements, distribution and load-balancing schemes, communication mechanisms, application parameters and algorithms

- **Performance Analysis Module**
  - Built on Performance prediction Functions (PF) and PF composition
  - Anticipate operation and expected performance of applications for a given workload and system configuration

- **Active Control Network**
  - Composed of sensors, actuators, and management agents
  - Sensors/actuators define interfaces and mechanics for adaptation
  - Agents configure and deploy application components at runtime for managing and optimizing application execution
Application-sensitive Management

- PAC tuple, 5-component metric
- Octant approach: app. runtime state
- GrACE (ISP), Vampire (pBD-ISP, GMISP+SP), Zoltan, Metis, partitioners
- ARMaDA framework
  - Computation/communication
  - Application dynamics
  - Nature of adaptation
- RM3D, 64 procs on “Blue Horizon”
  - 100 steps, base grid 128*32*32
  - 3 levels, RF = 2, regrid 4 steps

**ARMaDA evaluation for VectorWave-2D application on 32 processors on “Frea”**

<table>
<thead>
<tr>
<th>Partitioner</th>
<th>Execution time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFC</td>
<td>637.478</td>
</tr>
<tr>
<td>G-MISP+SP</td>
<td>611.749</td>
</tr>
<tr>
<td>pBD-ISP</td>
<td>592.05</td>
</tr>
<tr>
<td>ARMaDA with SFC start</td>
<td>470.531</td>
</tr>
</tbody>
</table>
Application-sensitive Management

• 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

![Graph showing run-times for ARMaDA partitioners for RM2D application on 64 processors on “Blue Horizon”](image)

  - 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

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Runtime Application State Characterization

• Computation/Communication requirements (“CCratio”)

\[
CCratio = \sum \frac{(Volume \ of \ bounding \ boxes)}{(Surface \ area \ of \ bounding \ boxes)}
\]

*high CCratio* ⇒ *more computation*, *low CCratio* ⇒ *more communication*

• Application Dynamics (“Dynamics”)

\[
Dynamics = \text{Size of} \ (\text{Current state boxes} \cap \text{Previous state boxes})
\]

*high value* ⇒ *lesser dynamics*, *low value* ⇒ *high activity dynamics*

• Nature of Adaptation (“Adapt”)

\[
Adapt = \frac{Volume \ of \ coarse \ level \ boxes}{Domain \ volume} \times \text{Number of coarse level boxes}
\]

*high Adapt* ⇒ *scattered adaptation*, *low Adapt* ⇒ *localized adaptation*
Octant Approach

- Used to classify runtime state with respect to:
  - Adaptation pattern (scattered or localized)
  - Runtime dominated by computation or communication
  - Activity dynamics in the solution
System-sensitive Management

- System characteristics using NWS
- RM3D compressible turbulence application
  - 128x64x64 base (coarse) grid
  - 3 levels, factor 2 refinement
- System/Environment
  - University of Texas at Austin (32 nodes), Rutgers (16 nodes)

<table>
<thead>
<tr>
<th>Procs</th>
<th>Dynamic Sensing (s)</th>
<th>Static Sensing (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>423.7</td>
<td>805.5</td>
</tr>
<tr>
<td>4</td>
<td>292</td>
<td>450</td>
</tr>
<tr>
<td>6</td>
<td>272</td>
<td>424</td>
</tr>
<tr>
<td>8</td>
<td>225</td>
<td>430</td>
</tr>
</tbody>
</table>

\[
C_k = w_p P_k + w_m M_k + w_b B_k
\]
System-sensitive Management

- **Application**
  - *RM3D (R. Samtaney, CIT)*
    - 128x32x32 base grid, 3 levels, refinement every 4 time-steps
- **System**
  - *Heterogeneous Linux cluster (64 nodes)*
  - *Synthetic load generators*
System Sensitive Management: Dynamic Load Assignment

- System sensitive partitioner can adapt to load dynamics (state sensing frequency of 20 iterations)
Proactive Runtime Management: Performance Functions

• Performance Functions (PFs)
  – *Describe the behavior of a system component, subsystem or compound system in terms of changes in one or more of its attributes*
  – *PFs characterize the operations and performance of any resource in a distributed environment*
  – *PFs can be composed to generate an overall end-to-end PF that characterizes and quantifies application performance*

• Generating PFs
  – *Identify the attributes that accurately express and quantify the operation and performance of a resource*
  – *Use experimental and analytical techniques to obtain the PFs over each system component*
  – *Use these PFs to generate an overall PF that can estimate the system performance at runtime*
Proactive Load Distribution Management

- Performance engine selects the appropriate performance function to predict the execution time of the application for next time step
  - The PF of RM3D on processor k for a given load $X_1$ and AMR level $X_2$ is empirically defined as:
    $t_k = a_0 + a_1X_1 + a_2X_2 + a_3X_1X_2 + a_4X_1^2 + a_5X_2^2 + a_6X_1^2X_2 + a_7X_1X_2^2 + a_8X_1^2X_2^2$
- Predicted execution time adjusted for current system load
  $t'_k = t_k \times L$
- Tune and adjust the processor capacities
  - Adjustment factor for each processor
    $$F_k(t) = \frac{t_{\text{avg}}}{t_k(t)}$$
  - Capacity of processor $k$ for next iteration
    $$C_k(t) = C_k(t-1) \times F_k(t)$$
Proactive System Sensitive Runtime Partitioning

Table. Self-optimizing performance gain for different base grid sizes on 4 processors cluster.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Execution time without self-optimization</th>
<th>Execution time with self-optimization</th>
<th>Percentage improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>64x16x16</td>
<td>438.83</td>
<td>362.97</td>
<td>17.29</td>
</tr>
<tr>
<td>64x32x16</td>
<td>780.95</td>
<td>660.67</td>
<td>15.40</td>
</tr>
<tr>
<td>64x32x32</td>
<td>2326.1</td>
<td>1681.68</td>
<td>27.70</td>
</tr>
<tr>
<td>64x64x32</td>
<td>4165.28</td>
<td>3535.3</td>
<td>15.12</td>
</tr>
</tbody>
</table>

Table. Self-optimizing performance gain for base grid size 64x64x32 on 8 processors

| Execution time without self-optimization | 2109.43 |
| Execution time with self-optimization   | 1651.33 |
| Percentage improvement                  | 21.72   |
Dynamic Load Redistribution

- PFs used to estimate the cost of dynamic redistribution and load balancing.
- PRAGMA determines at runtime when it is cost-effective to redistribute the grid hierarchy.
  - performance degradation due to load imbalance exceeds cost of repartitioning and data movement.
- Active runtime management reduce dynamic redistribution overheads by about 75% for the RM3D application.
Reactive Adaptations to Memory Availability

- Adapt SAMR distribution base match current memory availability
  - Monitor current memory availability using NWS
  - Tune patch size and mapping based on memory capacity
    - Tradeoff – cache locality, communication overheads

- Experimental setup
  - RM3D with 3 levels, mesh size is 128*32*32
  - 8 nodes are configured with NWS
  - Synthetically generate different memory consumption on processors P1, P3, P5, P7 with higher memory availability
Memory Adaptive Partitioner

• Partitioning Algorithm
  – Current memory availability of each processor is captured by NWS
  – Processors are sorted by memory availability (small → large)
  – Workload (i.e., list of patches) is sorted by size (small → large)
  – Assign smaller patches to processor with less memory

Fig. partitioning with memory adaptation

Fig. partitioning without memory adaptation

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Adaptation to Memory Availability: Results

Comparison of max patch size with different partitioning methods

Comparison of execution time of RM3D, 3 levels, 128*32*32

<table>
<thead>
<tr>
<th></th>
<th>With memory adaptation</th>
<th>Without memory adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Time (s)</td>
<td>395.939</td>
<td>672.564</td>
</tr>
<tr>
<td>Comp Time (s)</td>
<td>286.213</td>
<td>407.584</td>
</tr>
<tr>
<td>Regrid Time (s)</td>
<td>209.463</td>
<td>113.067</td>
</tr>
<tr>
<td>Sync Time (s)</td>
<td>196.747</td>
<td>400.79</td>
</tr>
<tr>
<td>Recompose Time (s)</td>
<td>218.299</td>
<td>249.771</td>
</tr>
</tbody>
</table>
RM3D+AMR Scalability (SDSC’s IBM SP - Blue Horizon)

- Base grid of 256x64x64
- 3 levels of factor 2 refinement
- Regridding every 4 steps
- 400 iterations
- AMR efficiency* > 85%
- Load imbalance < 11%
- Adaptive blocking
  - 4, 8, 16, 32
- Overlap of comm./comp.

\[ \text{AMR Efficiency} = 1 - \left( \frac{\text{# of points with AMR}}{\text{# of points with uniform grids}} \right) \]

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
• An autonomic computing system is one that has the capabilities of being self-defining, self-healing, self-configuring, self-optimizing, self-protecting, contextually aware, and open.

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Autonomic Computational Science and Engineering

- Simulation of Active Flow Control of Turbulent Flows
- Virtual Groundwater Basin Model

vGrid Application Development

- Thermonuclear Combustion Supernovae
- Non-Born - Oppenheimer Molecular Quantum Mechanics

vGrid Runtime Management

vGrid Grid Virtualization Services

Temporal Scheduling  Spatial Scheduling

Work Work Work

Grid Middleware

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Application Runtime Management in V-Grid

Grid Resource Hierarchy

Virtual Grid Resource
Autonomic Runtime Manager (ARM)

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

V-Grid Deduction
( Self-adaptation, Self-optimization, Self-healing)
Identify and characterize natural regions
Define objective functions and management strategy
Define VCU's

V-Grid Execution
Partition, Map and Tune

NR: Application Natural Regions
VCU: Virtual Computational Unit

Application Domain Hierarchy

Loop for each level of Grid/Application hierarchy
Runtime Management Architecture

Dynamic Driver Application

Computation/Communication

Application Dynamics

Nature of Adaptation

Application State Characterization

Self-Observation & Analysis

Self-Optimization & Execution

Autonomic Partitioning

Partition/Compose

Repartition/Recompose

Virtual Computation Unit (VCU)

Virtual Resource Unit (VRU)

Dynamic Driver Application

Monitoring & Context-Aware Services

Application Monitoring Service

Resource Monitoring Service

Heterogeneous, Dynamic Computational Environment

Natural Region Characterization

CPU

Memory

Bandwidth

Availability

Access Policy

System Capability Module

Resource History Module

System State Synthesizer

Performance Prediction Module

Objective Function Synthesizer

Deduction Engine

Prescriptions

Decision Space

Normalized Work Metric

Normalized Resource Metric

Autonomic Scheduling

Virtual Grid Autonomic Runtime Manager

Local Grid Scheduling

Global Grid Scheduling

Virtual Grid Time Scheduling (VGTS)

Virtual Grid Space Scheduling (VGSS)

Adaptation Overheads

Data Migration

Adaptive Partitioning

Memory Requirement

Application Locality

Load Balancing

Granularity Control

Autonomic Scheduling

Self-Optimization & Execution

Current System State

Current Application State

Deduction Engine

Deduction Engine

Deduction Engine

Objective Function Synthesizer

Resource History Module

Performance Prediction Module

Virtual Resource Unit

Mapping Distribution Redistributor

Execution

Natural Region Characterization

Heterogeneous, Dynamic Workload

VGTS VGSS VGTS VGSS

VGTS: Virtual Grid Time Scheduling

VGSS: Virtual Grid Space Scheduling

Navigation:

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
AutoMate: Enabling Autonomic Grid Applications

AutoMate Application Layer
- Autonomic Application Composition
- Opportunistic Interactions
- Composition/Context Agents
- Autonomic Applications

AutoMate Component Layer
- Autonomic Component
- Component Access Control Agent
- Component Rule/Context Agent
- Control Aspect
- Operational Aspect
- Functional Aspect

AutoMate System Layer
- Discovery, Factory, Lifecycle, Metadata, Monitoring, Interaction, Context Services
- Component Services
- Semantic P2P Messaging, Events, Notification
- System/Context Agents
- Grid Middleware (OGSA)
Autonomia: Autonomic Computing Infrastructure

MAS: Mobile Agent System
- Component Service Item
- Component Entry
Node Entry

MAS A
Host A
Component Monitor

MAS B
Host B
Component Monitor

CS1
CS2

CS: Computer Server
- Component
- Mobile Agent
- Component Entry

AMS
Java Spaces
- Self protection
- Self Optimization
- Self healing
- Self configuration

XML Parser

Shared DB

Resource Repository

Components Repository

Application Delegated Manager (ADM)
- Agent Interface
- Schedule/Launch
- Monitoring Server

User’s Application

Application Management Editor (XML)
Summary

- Adaptive and interactive simulations can enable accurate solutions of physically realistic models of complex phenomena
- PRAGMA: Proactive and Reactive Grid Application Management
  - Application monitoring and characterization
    • sensors, actuators, control network
  - Predictive performance functions
  - Active application control and management framework
    • reactive/proactive, application/system sensitive
- vGrid: Towards Autonomic Computational Science and Engineering
- Information, publications, software
  - www.caip.rutgers.edu/TASSL
  - www.ece.arizona.edu/~hpdc

M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03
Autonomic Computing Workshop (Sponsored by NSF)

- Autonomic Computing Workshop will be held in conjunction with the Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), June 25th, 2003 in Seattle Washington.


M. Parashar & S. Hariri, NGS Workshop, IPDPS, Nice 04/22/03