Distributed Programming Abstractions

http://wiki.esi.ac.uk/Distributed_Programming_Abstractions

Shantenu Jha\textsuperscript{12}, Murray Cole\textsuperscript{3}, Daniel Katz\textsuperscript{1}, Manish Parashar\textsuperscript{4}, Omer Rana\textsuperscript{5} \& Jon Weissman\textsuperscript{6}

\textsuperscript{1}Louisiana State University
\textsuperscript{2}e-Science Institute, Edinburgh
\textsuperscript{3}University of Edinburgh
\textsuperscript{4}Rutgers University
\textsuperscript{5}Cardiff University
\textsuperscript{6}University of Minnesota
First Generation of Grid Applications

- Demonstrated the potential of the Grid with “proof-of-concept” applications
- Addressed distributed computing issues from an HPC perspective (mostly)
  - Applications, programming systems, tools extended/adapted to address distributed computing issues
    - E.g., mpichg, ft-mpi, gridsolve
- Ad hoc and application/system specific solutions
- Scientific output not spectacular; can the next generation be made easier..
Developing Grid/Distributed Applications is (still) Hard!

- The many challenges of Grids/distributed systems
  - HPC *meets* Distributed Computing
    - A union of design and runtime issues -> unique challenges!
  - Performance, scale, data volumes, distribution, heterogeneity, dynamism, failures, ...

- Evolution of computational infrastructure
  - Increasing complexity, new requirements
    - Coupled, dynamic data-driven, data-intensive, end-to-end, application-level scheduling & related features
  - TeraGrid, PetaCloud(?), DEISA, PRACE..
Objectives of DPA Research Theme

• Study the Landscape of Distributed Applications
  • Understand requirements and characteristics (vectors)
    • What makes distributed applications hard to develop?
    • What makes distributed infrastructure hard to use?
  • Extract recurring patterns and best practices
    • How have Grid/DC issues been addressed at different levels?
    • Problem formulation, algorithm, programming system, middleware, infrastructure, etc.
  • How does/can infrastructure support attributes?
Objectives of DPA Research Theme - II

- Identify gaps, propose a strategy and next steps
  - Gaps in programming system, implementation mechanisms and run-time support
  - Effective Development & Deployment of applications
    - Extending existing approaches: Message passing, data parallel, shared memory, etc.
    - Defining new formulations: Asynchronous algorithms, bio-inspired, gossip based messaging, etc.
- Better resource utilisation?
- Shield applications from evolving infrastructure?
DPA: Application Driven Research

- **Integrated Approach:**
  - By studying real scientific applications to find models and abstractions that are used today,
  - Identify models and abstractions that could help make writing distributed applications easier
  - Commonly occurring modes are *patterns*
    - *e.g.*, Master-Worker
  - Support for a specific pattern is an *abstraction*
    - *e.g.*, Condor supports MW Framework
- To do this, need to study representative set of current and future real-world applications
Characterizing Applications

• An iterative process that involves describing and classifying applications
  1. Describe initial set of applications
  2. Determine vectors (axes and values on the axes) that describe the applications (want minimal spanning set of axes)
  3. Find clusters of vectors – define as application classes, with example applications
  4. Add/subtract applications and continue with step 2; repeat until changes are minimal
Representative Applications..

<table>
<thead>
<tr>
<th>Application Example</th>
<th>Execution Unit</th>
<th>Data Exchange</th>
<th>Coordination Environment (for data exchange and execution units)</th>
<th>Other Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Montage</td>
<td>Individual sequential and parallel executables</td>
<td>files - mostly one-to-one, some group-to-group, one-to-all, and all-to-one; files transferred by DAG enactor</td>
<td>DAG</td>
<td>(i) Message exchange is Event-driven and is tolerant to delays (ii) Centralized coordinator (iii) No control on execution ownership but component ownership is not important; ownership of resources.</td>
</tr>
<tr>
<td>Single Numerical Relativity simulation</td>
<td>multiple instances of single executable</td>
<td>concurrent coupling via messaging, bidirectional (MPI)</td>
<td>SPMD</td>
<td>MPI, co-scheduling</td>
</tr>
<tr>
<td>Home-based patient monitoring and data analysis. Involves period monitoring of patient data and data analysis at server</td>
<td>execution units include: (i) sensor-based data capture; (ii) data analysis of time series data</td>
<td>DAG</td>
<td>event-based – through the use of a workflow enactor</td>
<td>Web Services and Workflow (WebSphere)</td>
</tr>
<tr>
<td>Coupled Fusion Simulations</td>
<td>multiple simultaneously running parallel executables</td>
<td>loose coupling via data exchanged (streamed) between running executables</td>
<td>data</td>
<td>coupled simulations running in multiple systems</td>
</tr>
<tr>
<td>Autonomic Oil Reservoir Optimization (AORO)</td>
<td>parallel executables</td>
<td>data/event triggered dynamic workflow execution</td>
<td>data/event</td>
<td>decoupled coordination, content (pub/sub) messaging, dynamic discovery</td>
</tr>
</tbody>
</table>
# Application Classes

<table>
<thead>
<tr>
<th>Application Class (Cluster)</th>
<th>Properties (Values of Application Vectors)</th>
<th>Application Examples</th>
<th>General Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loosely Coupled Applications</td>
<td>Independent execution units; loosely coupled; data transfer through files.</td>
<td>Montage</td>
<td>Each execution unit is independently developed, and only exposes an I/O. Only an executable instance of each execution unit may be available. This application class involves a loose coupling between such executable instances.</td>
</tr>
<tr>
<td>Tightly Coupled Homogeneous Applications</td>
<td>Execution units interact via messaging or other mechanisms; strong dependency between units</td>
<td>Single Num. Rel. Simulation</td>
<td></td>
</tr>
<tr>
<td>Tightly Coupled Heterogeneous Applications</td>
<td>Execution units interact via files; strong dependency between units; units may be implemented using different programming libraries</td>
<td>Molecular Dynamics Simulation (with separate components implementing different capabilities — such as force and velocity calculations, position of particles, etc)</td>
<td>Trivial generalization of MPI to WAN using MPICH-G2 and variants thereof, should also be considered here.</td>
</tr>
<tr>
<td>Loose Coupling of Tightly Coupled Applications</td>
<td>Many applications need to be used in conjunction with other applications; they may not necessarily have been designed for this ab initio. Alternatively, certain commonly used algorithms call for replacing single long-time running simulations with multiple shorter time-duration simulations — but with possible infrequent communication between the individual simulations.</td>
<td>Replica Exchange simulations for protein folding, multi-physics scientific applications</td>
<td>There are two levels of communication; the first is internal to a single job/task (think monolithic MPI job), the second level is communication between the jobs/talks. The latter is less frequent and thereby more tolerant of latency and delays. Coordination of the many tasks/jobs is challenging.</td>
</tr>
<tr>
<td>Event-Oriented Applications</td>
<td>Execution units employ a publish/subscribe mechanism to coordinate; generally loose coupling between units</td>
<td>Distributed database search</td>
<td></td>
</tr>
</tbody>
</table>

---
Montage

• Application: Montage – Astronomical image mosaicking – takes multiple images from telescope(s) and combines them into a single scientifically-accurate image

• Vectors
  • Execution units are individual sequential and parallel tasks
  • Data exchange is through files; generally one-to-one, but sometimes one-to-all, all-to-one, group-to-group
  • Coordination is via a DAG that is generated when the problem is specified
  • Execution environment is a workflow enactor that takes runs the tasks in an order consistent with the DAG and moves files as needed for the tasks

• Classification: loosely coupled; also represents LIGO data analysis, many other complex parallel data processing tasks
  • Also includes subsets such as one-level workflows (e.g., mpiBLAST)
Patterns and DPA - I

• Distributed (and parallel) applications can be clustered around adherence to one or more underlying architectural *patterns*

• Benefits of understanding, abstracting, and building applications *explicitly* around patterns:
  • *Foresight* enables *optimisation*  
    “We’ll need this later over there”
  • *Constraints* enable *verification*  
    “Are you sure you mean that?”
  • *Abstraction* enables *portability*  
    “A farm is just a farm”
Useful DPA patterns may abstract concerns (and solutions) from both application and infrastructure domains.

Distinguish DA requirements into three levels:
- Programming, Deployment and Usage

Some DPA patterns are already supported:
- Programming: MapReduce, AllPairs,
- Deployment: @Home
- Usage: Master-Worker (Condor),
The Power of Abstractions

- Investigate the commonly occurring patterns associated with DA – programming, deployment and usage
  - How have these patterns been supported, i.e., what are the abstractions?
  - “The design space of programming tools & development methods reflects a snapshot of infrastructure status”
  - *Integrate application development, programming tools & infrastructure*
- By understanding these patterns and providing abstractions that support these patterns, can we improve:
  - Development & deployment of applications?
  - Better resource utilisation?
  - Shield applications from evolving infrastructure?
- Theme: Attempt to answer these (and more) question
Workshop in Relation to Theme

- Overlap with CoreGrid PM
- Common Set of Questions:
  - What Abstractions (programming and system) does your approach/tool/model provide?
  - Applications (science problem) developed?
  - Some applications that can/could use these (scope of applications, application classes)
  - Challenges?
- Europar08 Proceedings + Book on, “Abstractions for Distributed Applications and Systems”..
Additional Questions

• What are the main DA types in your project?
  • What are the main characteristics of these DA

• What are the common patterns – programming, deployment and usage – associated with these distributed applications?

• What abstractions for these commonly occurring modes (patterns) are available on the distributed infrastructure that you either use or are affiliated with?

• Gap Analysis of the abstractions and infrastructure:
  • Are the patterns well supported? Not supported? What further abstractions should be considered?
    • What abstractions have worked for you? (two examples)
    • What abstractions do you feel you need? (two examples)
  • How well will abstractions work with the next generation of infrastructure that your project will use?
<table>
<thead>
<tr>
<th>TIME</th>
<th>NAME</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:30 - 14:45</td>
<td>Shantenu Jha</td>
<td>Distributed Programming Abstractions</td>
</tr>
<tr>
<td>14:45 - 15:10</td>
<td>Marco Danelutto</td>
<td>Behavioural Skeletons: A Programming Abstraction Relieving Programmers of Non-functional Concerns</td>
</tr>
<tr>
<td>15:10 - 15:35</td>
<td>Domenico Talia</td>
<td>Distributed Data Mining Patterns as Services in Grids and Distributed Infrastructures</td>
</tr>
<tr>
<td>15:35 - 16:00</td>
<td>Marian Bubak</td>
<td>Methods and Tools for Development and Running Collaborative Applications</td>
</tr>
<tr>
<td>16:00 - 16:30</td>
<td>-</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:30 - 16:55</td>
<td>Denis Caromel, Mario Leyton</td>
<td>ProActive Parallel Suite: From Active Objects-Skeletons-Components to Environment &amp; Deployment</td>
</tr>
<tr>
<td>16:55 - 17:20</td>
<td>Christian Perez, Thierry Priol</td>
<td>On the Abstraction of Software Component Models</td>
</tr>
<tr>
<td>17:20 - 17:45</td>
<td>Jose Cunha</td>
<td>Abstractions for Organising Dynamic Distributed Systems</td>
</tr>
<tr>
<td>17:45 - 18:15</td>
<td>Discussion and Lessons Learned</td>
<td>Omer Rana</td>
</tr>
</tbody>
</table>

Schedule