A Lightweight Java Taskspaces Framework for Scientific Computing on Computational Grids

H. De Sterck\textsuperscript{1}, R.S. Markel\textsuperscript{2}, T. Pohl\textsuperscript{3}, and U. Rüde\textsuperscript{3}

\textsuperscript{1}Department of Applied Mathematics, University of Colorado at Boulder, USA
\textsuperscript{2}Advansys, Inc., Boulder, CO, USA
\textsuperscript{3}Department of Computer Science, University of Erlangen-Nürnberg, Germany

SAC 2003 - PDSN
Tuesday, March 11, 2003

(desterck@colorado.edu)
Introduction

Taskspaces grid computing framework:

- **Objectives:**
  1. **heterogeneous grids:** use isolated workstations, Beowulf clusters, parallel supercomputers, connected over the internet, “all” operating systems
  2. **lightweight, usability** (user – administrator): copy one single small executable file to worker and execute one single command to start worker; all application code downloaded
  3. **applications:** taskfarming, but also “parallel scientific computing” with intertask communication

- **Other approaches:**
Outline

1. Design and implementation
2. Parallel scaling tests
   – Jacobi iteration
3. Other applications
4. Future work
1. Design and implementation

- design principles

1. decentralization through the use of tuple spaces

2. distributed tuple space for intertask communication

3. platform independence, object orientation, code downloading, security mechanisms \(\Rightarrow\) Java

4. mobile agent objects for grid administration
Tuple spaces and bag-of-tasks paradigm

- a Tuple Space contains Task Objects
- Task Object = data + methods (code)
- decoupled in space and time
- self-configuring, no central coordination
- fault-tolerance (transaction-based)
- Tuple Spaces pioneered in the late 70s (Gelernter, “Linda”)
Communication – global tuple space

worker 17

\textit{task 5}

\ldots

worker 312

\textit{task 6}

\text{communication bag}

\text{comm 5\rightarrow 6}

\text{comm 6\rightarrow 5}

\text{comm 5\rightarrow 4}

\ldots

global communication space = \text{bottleneck}
Communication – distributed tuple space

- distribute the communication tuple space over the workers
- two stages
  (1) set up communication pattern through global tuple space (only one time for fixed communication pattern)
  (2) direct communication through local communication spaces
Java implementation – class diagram

- all classes extend “Communicator”
- socket connections (TCP/IP), object streams, serializable objects
- security: digitally signed jar files
- event notification in stead of polling mechanism
Taskspaces deployment diagram

- Runner.class < 2 kb
- task agent / reset agent
2. Parallel scaling tests – Jacobi iteration

- numerical linear algebra: Jacobi iterative method
- parallel: in every iteration neighbor–neighbor communication (exchange of boundary rows)

- constant problem size per processor \((1500^2 \text{ grid points})\)

\[ \Rightarrow \text{perfect parallel scaling} = \text{constant total time} \]
Direct communication

<table>
<thead>
<tr>
<th># of processors</th>
<th># of grid points</th>
<th>direct communication</th>
<th>communication bag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1536^2$</td>
<td>58s</td>
<td>58s</td>
</tr>
<tr>
<td>2x2</td>
<td>$3072^2$</td>
<td>74s</td>
<td>87s</td>
</tr>
<tr>
<td>4x4</td>
<td>$6144^2$</td>
<td>83s</td>
<td>216s</td>
</tr>
<tr>
<td>8x8</td>
<td>$12288^2$</td>
<td>89s</td>
<td>1504s</td>
</tr>
</tbody>
</table>

Direct communication versus communication through a central communication bag (200 Jacobi iterations on Blue Horizon, IBM SP, SDSC).

\[ \rightarrow \text{direct communication scales well (constant time)} \]
Grid experiment (200 Jacobi iterations, $1500^2$ grid points per processor). Number of processors (1 worker process per processor) and total execution times are shown.

$\Rightarrow$ near-perfect scaling for problems with neighbor-neighbor communication over internet
High throughput grid experiment

<table>
<thead>
<tr>
<th>System</th>
<th>Workers/Processor</th>
<th>Processes</th>
<th>Total Workers</th>
<th>Total Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Horizon, SDSC, San Diego, CA</td>
<td>64</td>
<td>128</td>
<td>240</td>
<td>105s</td>
</tr>
<tr>
<td>P4 Linux, CU Boulder, CO</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>103s</td>
</tr>
<tr>
<td>Itanium Linux, CU Boulder, CO</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>101s</td>
</tr>
<tr>
<td>forseti1, NCSA, Urbana, IL</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>hermod, NCSA, Urbana, IL</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Total number of workers: 104, 168, 280
Total execution time: 105s, 103s, 101s

High throughput grid experiment (50 Jacobi iterations, $500^2$ grid points per worker). Number of worker processes and total execution times are shown. Problem size is constant per worker process.

→ Taskspaces framework: parallel scientific computing with intertask communication is scalable on the internet!
3. Other applications

(1) engineering design collaboration

- Navier-Stokes gas dynamics, Lattice-Boltzmann
- “driven cavity”
- neighbor-neighbor communication in every step
- obstacles can be added interactively
- versatility of framework!
(2) Bioinformatics: RNA origin of life

with Rob Knight, MCDB, CU Boulder

- fold very large numbers ($\sim 10^9$) of random RNA sequences (length 30-500)
- look for “functional motifs” in the RNA structures that catalyze biochemical reactions
- main question: can life have originated from a small pool of random RNA molecules?
Origin of life: from random molecules!

- initial approach: 46 million RNA sequences, \( \sim \) 1000 tasks, 8 hours/task on Platinum (1GHz P3 linux cluster at NCSA, 968 compute processors)
- for folding, use C-code “Vienna” compiled on workers, called from Java
- using Taskspaces, one of the missing links in the chain of life is being established!
4. Future work

- we seek collaborators and government/corporate funding to extend our framework as follows:

(1) Optimal collective communication trees

- Taskspaces scales for neighbor-neighbor communication
- collective communication: broadcast from one to all, reduce, ...
- within parallel computers: optimal strategy = binomial tree
Collective communication

- high-latency/low-bandwidth connections: flat tree optimal

- multiple clusters, heterogeneous grids: optimal tree is somewhere in between
Collective communication: optimal trees

- we propose:
  - dynamically test communication time between nodes
  - grid discovers its network topology dynamically
  - determine optimal collective communication trees using clustering algorithm

⇒ scalable collective communication!
(2) Self-migrating parallel workflows

- heterogeneous collection of machines
- agent sets up parallel workflow (e.g., proteomics data processing)
- upon partial or total failure (catastrophic events), parallel workflow migrates
  ⇒ e.g., homeland security applications
A Lightweight Java Taskspaces Framework for Scientific Computing on Computational Grids

H. De Sterck\textsuperscript{1}, R.S. Markel\textsuperscript{2}, T. Pohl\textsuperscript{3}, and U. R"ude\textsuperscript{3}

\textsuperscript{1}Department of Applied Mathematics, University of Colorado at Boulder, USA
\textsuperscript{2}Advansys, Inc., Boulder, CO, USA
\textsuperscript{3}Department of Computer Science, University of Erlangen-N"urnberg, Germany

SAC 2003 - PDSN
Tuesday, March 11, 2003

(desterck@colorado.edu)