Developing Extensible, Scalable Distributed Data-Intensive Scientific Applications

Shantenu Jha\textsuperscript{1,2}, Daniel Katz\textsuperscript{3}, Jon Weissman\textsuperscript{4}

\textsuperscript{1}CCT \& CS, LSU
\textsuperscript{2}e-Science Institute, Edinburgh
\textsuperscript{3}University of Chicago
\textsuperscript{4}University of Minnesota
The Case: Developing Extensible Scalable DDIA

- Data Inherently Distributed
  - Distributed Data Source/Sink:
    - c.f. Ian’s talk ("end-to-end problem")
    - Dynamically Data-Driven Applications e.g., Sensor-based
  - Collaborations:
    - Physically Distributed Usage
  - Multiple Applications/Services/Data, but each is "localized"
    - Ownership of Data Medical Data needs to be processed in-situ (no option but to move compute to data)
    - Distributed Data-Centers: Cost-effective, Space..
    - Multiple Mash-Up of Big-Data and Streaming Services

Distributed DI-Application is not just a simple sum of DI-Apps
The Case: Developing Extensible D-DIA (2)

• Multiple, Heterogeneous Infrastructure
  – Interoperation, e.g., Concurrently cross Grid-Clouds
  – Decouple Application Development from underlying infrastructure
    • Scale-out (and not just Scale-up)
• Frameworks: Support Runtime or Application Characteristics for multiple applications and different infrastructure
  – Support Multiple Programming Models
    • Master-Worker, but Irregular versus Regular Workload
  – Support Application-Level Patterns
    • MapReduce, File-based versus Stream-based
  – Support Distributed Affinities
A Distributed Data-Intensive Stack

- **Distributed Data Intensive Applications**
  - e.g. Particle Physics, Astronomy, Bio-Informatics

- **Programming Abstractions / Patterns/ Higher-level APIs**
  - e.g. MapReduce, All-Pairs

- **Common Runtime Support**
  - e.g. Distributed Affinity, Data Placement, Fault Tolerance

- **Physical Infrastructure**
  - e.g. TeraGrid/XD, Clouds, Future DoE DIS

**AF** = Application Framework, e.g., Dryad
**RF** = Runtime Framework, e.g., Nexus
Developing Distributed Applications Using SAGA
Examples: SAGA-based Frameworks

- SAGA MapReduce Framework:
  - Control the distribution of Tasks (workers)
  - Infrastructure-independent
  - Master-Worker: File-Based &/or Stream-Based
  - Data-locality optimization using SAGA’s replica API

- SAGA All-Pairs Framework:
  - Compute Matrix Elements, each is a Task
    - All-to-All comparison
  - Control the distribution of Tasks and Data
  - Data-locality optimization through external (runtime) modules
SAGA-MapReduce

- Interoperability: Use multiple infrastructure concurrently
- Control the $N_w$ placement
  - Simple staging of data
- SAGA-Sphere-Sector:
  - Open Cloud Consortium
- Stream processing model
  - Ongoing work
  - Apply to all elements (files) in a data-set (stream)

<table>
<thead>
<tr>
<th>Number-of-Workers</th>
<th>Size (MB)</th>
<th>$T_s$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG 1, AWS 2, Eucalyptus 1</td>
<td>10</td>
<td>1.5</td>
</tr>
<tr>
<td>TG 2, AWS 1, Eucalyptus 1</td>
<td>10</td>
<td>1.9</td>
</tr>
<tr>
<td>TG 1, AWS 2, Eucalyptus 1</td>
<td>100</td>
<td>2.9</td>
</tr>
<tr>
<td>TG 2, AWS 1, Eucalyptus 1</td>
<td>100</td>
<td>3.0</td>
</tr>
<tr>
<td>TG 2, AWS 2</td>
<td>10</td>
<td>1.4</td>
</tr>
<tr>
<td>TG 3, AWS 3</td>
<td>10</td>
<td>1.6</td>
</tr>
<tr>
<td>TG 4, AWS 4</td>
<td>10</td>
<td>2.1</td>
</tr>
<tr>
<td>TG 5, AWS 5</td>
<td>10</td>
<td>3.8</td>
</tr>
</tbody>
</table>

$T_s$: Time-to-solution, including data-staging for SAGA-MapReduce (simple file-based mechanism)
SAGA All-Pairs: Runtime Data Placement

- Classical: Place task on 4 LONI machines (512px Dell Clusters)
  - Simple data staging
- “Intelligent”: Map a task to a resource based upon Cost
  - Cost = Data Dependency + transfer times (latency)
- “Ignoring data dependencies is no longer an option”
  - Quote Chris Miceli 😊
Other Frameworks: Deployment & Coordination

Multiple Pilot-Jobs

BigJob::TeraGrid

BigJob::OSG
DDI Applications: Research Challenges

- Goal: Develop DDI scientific applications to utilize a broad range of distributed systems, without vendor lock-in, or disruption, yet with the flexibility and performance that scientific applications demand.
  - Frameworks as possible solutions
- Frameworks address some primary challenges in developing distributed DDI Applications
  - Coordination of distributed data & computing
  - Runtime (dynamic) scheduling, placement
  - Fault-tolerance
- Many Challenges in developing such Frameworks:
  - What are the components? How are they coupled? How is functionality expressed/exposed?
  - Layering, Ordering, Encapsulations of Components
Acknowledgements

Funding Agencies:
UK EPSRC (DPA, OMII-UK, OMII-UK PAL)
CCT Internal Resources

People:
SAGA D&D: Hartmut Kaiser, Ole Weidner, Andre Merzky, Joohyun Kim, Lukasz Lacinski, João Abecasis, Chris Miceli, Bety Rodriguez-Milla
Special Users: Andre Luckow, Yaakoub el-Khamra, Kate Stamou, Cybertools (Abhinav Thota, Jeff, N. Kim), Owain Kenway
Google SoC: Michael Miceli, Saurabh Sehgal, Miklos Erdelyi
Collaborators and Contributors: Steve Fisher & Group, Thilo Kielman & Co, Sylvain Renaud (JSAGA), Go Iwai & Yoshiyuki Watase (KEK)

http://saga.cct.lsu.edu
http://cct.lsu.edu/~sjha/select_publications