Supporting Data-Intensive Distributed Computing

Ioan Raicu
Computer Science Department, Illinois Institute of Technology
Math and Computer Science Division, Argonne National Laboratory

July 18th, 2013
DePaul University
• **Research Focus**
  – Emphasize designing, implementing, and evaluating systems, protocols, and middleware with the goal of supporting *data-intensive applications on extreme scale distributed systems*, from many-core systems, clusters, grids, clouds, and supercomputers

• **People**
  – Dr. Ioan Raicu (Director)
  – 6 PhD Students
  – 2 MS Students
  – 4 UG Students

• **Contact**
  – iraicu@cs.iit.edu
Today (2013): Multicore Computing
- O(10) cores commodity architectures
- O(100) cores proprietary architectures
- O(1000) GPU hardware threads

Near future (~2019): Manycore Computing
- ~1000 cores/threads commodity architectures
• Today (2013): Petascale Computing
  – O(100K) nodes
  – O(1M) cores

• Near future (~2018): Exascale Computing
  – ~1M nodes (10X)
  – ~1B processor-cores/threads (1000X)
Proposed Software Stack in Large-Scale Distributed Systems

Applications

Many-Task Computing
(SwiftScript, Charm++, MapReduce)

High-Performance Computing
(MPI)

Resource Manager
(Cobalt, SLURM)

Parallel File Systems
(GPFS, PVFS)

Distributed Execution Fabric
(MATRIX)

Simulator
(SimMatrix)

Persistent Distributed Hash Tables (ZHT)

Distributed File Systems (FusionFS)

High-End Computing Hardware
(Petascale to Exascale Systems)

Hardware
(Terascale)

& (GeMTC)
Many-Task Computing (MTC)

MTC emphasizes:
- bridging HPC/HTC
- many resources
  - short period of time
- many computational tasks
- dependent/independent tasks
- tasks organized as DAGs
- primary metrics are seconds

Advantages:
- Improve fault tolerant
- Maintain efficiency
- Programmability & Portability
- support embarrassingly parallel and parallel applications
Swift/T and Applications

• Swift/T
  o Active research project (CI UChicago & ANL)
  o Parallel Programming Framework
  o Throughput ~25k tasks/sec per process
  o Shown to scale to 128k cores

• Application Domains Supported
  o Astronomy, Biochemistry, Bioinformatics, Economics, Climate

Swift lets you write parallel scripts that run many copies of ordinary programs concurrently, using statements like this:

```swift
foreach protein in proteinList {
    runBLAST(protein);
}
```

Images from Swift Case Studies -
http://www.ci.uchicago.edu/swift/case_studies/
## Swift Applications

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Characteristics</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Astronomy</td>
<td>Creation of montages from many digital images</td>
<td>Many 1-core tasks, much communication, complex dependencies</td>
<td>E</td>
</tr>
<tr>
<td>Astronomy</td>
<td>Stacking of cutouts from digital sky surveys</td>
<td>Many 1-core tasks, much communication</td>
<td>E (Falkon)</td>
</tr>
<tr>
<td>Biochemistry</td>
<td>Analysis of mass-spec data for post-translational protein modifications</td>
<td>10,000 – 100,000 K jobs for proteomic searches using custom serial codes</td>
<td>D</td>
</tr>
<tr>
<td>Biochemistry</td>
<td>Protein folding using iterative fixing algorithm, also exploring other biomolecule interactions</td>
<td>100s to 1000s of 1-1000 core simulations &amp; data analysis</td>
<td>O</td>
</tr>
<tr>
<td>Biochemistry</td>
<td>Identification of drug targets via computational screening</td>
<td>Up to 1M x 1 core</td>
<td>O (Falkon)</td>
</tr>
<tr>
<td>Bioinformatics</td>
<td>Metagenome modeling</td>
<td>1000’s of 1-core integer programming problems</td>
<td>D</td>
</tr>
<tr>
<td>Business economics</td>
<td>Mining of large text corpora to study media bias</td>
<td>Analysis and comparison of 70M+ text files of news articles</td>
<td>D</td>
</tr>
<tr>
<td>Climate</td>
<td>Ensemble climate model runs and analysis of output data</td>
<td>10s to 100s of 100-1000 core simulations</td>
<td>E</td>
</tr>
<tr>
<td>Economics</td>
<td>Generation of response surfaces for various economic models</td>
<td>1K to 1M 1-core runs (10K typical), then data analysis</td>
<td>O</td>
</tr>
<tr>
<td>Neuroscience</td>
<td>Analysis of functional MRI datasets</td>
<td>Comparison of images; connectivity analysis with SEM, many tasks (100K+)</td>
<td>O</td>
</tr>
<tr>
<td>Radiology</td>
<td>Training of computer aided diagnosis algorithms</td>
<td>Comparison of images; many tasks, much communication</td>
<td>D</td>
</tr>
<tr>
<td>Radiology</td>
<td>Image processing and brain mapping for neurosurgical planning research</td>
<td>1000’s of MPI application executions</td>
<td>O</td>
</tr>
</tbody>
</table>
Accelerator Architecture

GPU
- Streaming Multiprocessors (15 SMXs on Kepler K20)
- Warps
  - 32 threads in a warp
  - 192 warps
    - i. hardware available
    - ii. ind. compute

Coprocessors
- Intel Xeon Phi
  - 60 cores * 4 threads per core = 240 hardware threads
GPU Block Diagram - Highlighting SMX
Highlighting SMX and Warps
How do you program GPUs?

C / C++ - Fortran - Swift/T

User Runtime

CUDA/OpenCL/OACC → GPU Code

Operating System / Device Driver

NVIDIA Graphics Processing Unit
• Collaborations with groups seeking interesting applications

• Collaborative proposals to NSF or NIH combining Medical Imaging and Distributed Systems (Clouds, Big Data, and/or parallelism)
• More information:
  – http://www.cs.iit.edu/~iraicu/
  – http://datasys.cs.iit.edu/

• Contact:
  – iraicu@cs.iit.edu

• Questions?