Data-Intensive Computing at the Intersection of Cloud Computing and Supercomputing

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

Who We Are, What We Are Seminar at Illinois Institute of Technology
September 23rd, 2013
• **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**
  – 1 Faculty Member
  – 6 PhD Students
  – 2 MS Students
  – 3 UG Students

• **More information**
• 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)
Some Challenges to Overcome at Exascale Computing

• Programming paradigms
  – HPC is dominated by MPI today
  – Will MPI scale another 3 orders of magnitude?
  – Other paradigms (including loosely coupled ones) might emerge to be more flexible, resilient, and scalable

• Storage systems will need to become more distributed to scale ➔ Critical for resilience of HPC

• Network topology must be used in job management, data management, compilers, etc
Fundamental Building Blocks (with a variety of resilience and consistency models)

- **Distributed hash tables (DHT)**
  - Also known as NoSQL data stores or key/value stores
  - Examples: Chord, Tapestry, memcached, Dynamo, MongoDB, Kademlia, CAN, Tapestry, Memcached, Cycloid, Ketama, RIAK, Maidsafe-dht, Cassandra and C-MPI, BigTable, HBase

- **Distributed message queues (DMQ)**
  - Example: SQS, RabbitMQ, Couch RQS, ActiveMQ, KAFKA, Hedwig
• Global File Systems and Storage
• Job Management Systems
• Workflow Systems
• Monitoring Systems
• Provenance Systems
• Data Indexing
• Relational Databases
MTC emphasizes:
• bridging HPC/HTC
• many resources
  o 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**
  - Active research project (CI UChicago & ANL)
  - Parallel Programming Framework
  - Throughput ~25k tasks/sec per process
  - Shown to scale to 128k cores

- Application Domains Supported
  - 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/
ZHT: Zero-Hope Distributed Hash Table

- ZHT: A distributed Key-Value store
  - Light-weighted
  - High performance
  - Scalable
  - Dynamic
  - Fault tolerant
  - Strong Consistency
  - Persistent
  - Versatile: works from clusters, to clouds, to supercomputers
A distributed file system co-locating storage and computations, while supporting POSIX.

Everything is decentralized and distributed.

Aims for millions of servers and clients scales.

Aims at orders of magnitude higher performance than current state of the art parallel file systems.
MATRIX - distributed MTC execution framework for distributed load balancing using Work Stealing algorithm

• Distributed scheduling is an efficient way to achieve load balancing, leading to high job throughput and system utilization

• Dynamic job scheduling system at the granularity of node/core levels for extreme scale applications
CloudKon: Cloud-Enabled Distributed Task Execution Framework

- Use Amazon services as building blocks
  - SQS, DynamoDB, and EC2
- Distributed load balancing
- Dynamic and Elastic
- Light-weight and Fast (2X+)
- Small codebase (1K-LOC, 5%)
GPU
- Streaming Multiprocessors (15 SMXs on Kepler K20)
- 192 warps * 32 threads

Coprocessors
- Intel Xeon Phi
- 60 cores * 4 threads per core = 240 hardware threads

GeMTC
- Efficient support for MTC on accelerators
• **Decentralization is critical**
  – Computational resource management
  – Storage systems

• **Preserving locality is critical!**
  – POSIX I/O on shared/parallel file systems ignore locality
  – Data-aware scheduling coupled with distributed file systems that expose locality is the key to scalability over the next decade

• **Co-locating storage and compute is GOOD**
  – Leverage the abundance of processing power, bisection bandwidth, and local I/O

Resource Management for Extreme-Scale Data-Intensive Computing
Coursework

- CS 451 Introduction to Distributed Computing
- CS 546 Parallel and Distributed Processing
- CS 550 Advanced Operating Systems
- CS 552 Distributed Real-Time Systems
- CS 553 Cloud Computing
- CS 554 Data-Intensive Distributed Computing
- CS 570 Advanced Computer Architecture
Specializations

• Undergraduate Level
  – Distributed and Cloud Computing
  – Data Science

• Master Level
  – Distributed and Cloud Computing
  – Cyber-Physical Systems
  – Data Analytics
Faculty

• Xian-He Sun

• Zhiling Lan

• Shangping Ren

• Ioan Raicu
<table>
<thead>
<tr>
<th>Employment Opportunities</th>
</tr>
</thead>
</table>

- Google  
- Yahoo  
- Microsoft  
- Amazon  
- IBM  
- Apple  
- VMWare  
- Netflix  
- Cray  
- Intel

- NVIDIA  
- Facebook  
- LinkedIn  
- Salesforce.com  
- Rackspace  
- Red Hat  
- Cleversafe  
- UnivaUD  
- Greenplum  
- AsterData

- Proprietary Trading Companies  
- Department of Energy Laboratories  
- NASA  
- Academic supercomputer centers  
- Many more…
• More information:
  – http://www.cs.iit.edu/~iraicu/
  – http://datasys.cs.iit.edu/
• Contact:
  – iraicu@cs.iit.edu
• Questions?