The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems

Ioan Rａlcu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Motorola Labs
March 4th, 2009
Purpose

– On-demand “stacks” of random locations within ~10TB dataset

Challenge

– Processing Costs:
  • $O(100\text{ms})$ per object
– Data Intensive:
  • 40MB:1sec
– Rapid access to 10-10K “random” files
– Time-varying load
Challenges

1. Slow job dispatch rates
2. Long queue times
3. Poor shared/parallel file system scaling
Growing Storage/Compute Gap

- **Local Disk:**
  - 2002-2004: ANL/UC TG Site (70GB SCSI)
  - Today: PADS (RAID-0, 6 drives 750GB SATA)
- **Cluster:**
  - 2002-2004: ANL/UC TG Site (GPFS, 8 servers, 1Gb/s each)
  - Today: PADS (GPFS, SAN)
- **Supercomputer:**
  - 2002-2004: IBM Blue Gene/L (GPFS)
  - Today: IBM Blue Gene/P (GPFS)
High-Throughput Computing & High-Performance Computing

• HTC: High-Throughput Computing
  – Typically applied in clusters and grids
  – Loosely-coupled applications with sequential jobs
  – Large amounts of computing for long periods of times
  – Measured in operations per month or years

• HPC: High-Performance Computing
  – Synonymous with supercomputing
  – Tightly-coupled applications
  – Implemented using Message Passing Interface (MPI)
  – Large amounts of computing for short periods of time
  – Usually requires low latency interconnects
  – Measured in FLOPS
MTC: Many-Task Computing

- Bridge the gap between HPC and HTC
- Applied in clusters, grids, and supercomputers
- Loosely coupled apps with HPC orientations
- Many activities coupled by file system ops
- Many resources over short time periods
  - Large number of tasks, large quantity of computing, and large volumes of data

[MTAGS08] “Many-Task Computing for Grids and Supercomputers”
Many-Task Computing for Grids and Supercomputers

Problem Space

Input Data Size

Hi
Med
Low

Number of Tasks

1
1K
1M

MapReduce/MTC
(Data Analysis, Mining)

HPC
(Heroic MPI Tasks)

MTC
(Big Data and Many Tasks)

HTC/MTC
(Many Loosely Coupled Tasks)
[HPDC09] “The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems”, under review
[MTAGS08 Workshop] Workshop on Many-Task Computing on Grids and Supercomputers
[MTAGS08] “Many-Task Computing for Grids and Supercomputers”
[GCE08] “Cloud Computing and Grid Computing 360-Degree Compared”
[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
[TG08] “Data Intensive Scalable Computing on TeraGrid: A Comparison of MapReduce and Swift”
[GlobusWorld08] “Managing and Executing LooselyCoupled Large Scale Applications on Clusters, Grids, and Supercomputers”
[NOVA08] “Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments”
[UC07] “Harnessing Grid Resources with Data-Centric Task Farms”
[SC07] “Falkon: a Fast and Light-weight task execution framework”
[MSES07] “A Data Diffusion Approach to Large Scale Scientific Exploration”
[SWF07] “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”
[TG07] “Dynamic Resource Provisioning in Grid Environments”
[NASA06-08] “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”
[SC06] “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”
[TG06] “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”
[NSF06] “The Importance of Data Locality in Distributed Computing Applications”
“Significant performance improvements can be obtained in the analysis of large dataset by leveraging information about data analysis workloads rather than individual data analysis tasks.”

- **Important concepts related to the hypothesis**
  - **Workload**: a complex query (or set of queries) decomposable into simpler tasks to answer broader analysis questions
  - **Data locality** is crucial to the efficient use of large scale distributed systems for scientific and data-intensive applications
  - Allocate computational and caching storage resources, **co-scheduled** to optimize workload performance
Proposed Solution: Data Diffusion

- Resource acquired in response to demand
- Data diffuse from archival storage to newly acquired transient resources
- Resource “caching” allows faster responses to subsequent requests
- Resources are released when demand drops
- Optimizes performance by coscheduling data and computations
- Decrease dependency of a shared/parallel file systems
- Critical to support data intensive MTC

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
• Captures data diffusion properties
• Models the efficiency and speedup of entire workloads

• Base definitions
  – Data Stores (Persistent & Transient)
  – Compute resources (transient)
  – Data Objects
  – Tasks

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
[UC07] “Harnessing Grid Resources with Data-Centric Task Farms”
Data Diffusion: Execution Model

- Dispatch Policy
  - first-available (FA), max-compute-util (MCU), max-cache-hit (MCH), good-cache-compute (GCC)
- Caching Policy
  - random, FIFO, LRU, LFU, 2
- Replay Policy
- Data Fetch Policy
- Resource Acquisition Policy
  - one-at-a-time, additive, exponential, all-at-once
- Resource Release Policy

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
[UC07] “Harnessing Grid Resources with Data-Centric Task Farms”
• Competitive ratio (worst case) between online algorithm and offline optimal
  – Measures the quality of the online algorithm, independent of data access patterns or workload characteristics
• The relation we prove to establish that 2Mark is $O(NM)$-competitive

\[
-2\text{Mark} (\sigma) \leq (NM + 2M / s + NM / (s + v)) \cdot \text{OPT} (\sigma)
\]

for all sequences $\sigma$

Philip Little, Amitabh Chaudhary,
University of Notre Dame

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
• What would data diffusion look like in practice?
• Extend the Falkon framework
**Scheduling Policies**

- **FA**: first-available
  - simple load balancing
- **MCH**: max-cache-hit
  - maximize cache hits
- **MCU**: max-compute-util
  - maximize processor utilization
- **GCC**: good-cache-compute
  - maximize both cache hit and processor utilization at the same time

---

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review

• 3GHz dual CPUs
• ANL/UC TG with 128 processors
• Scheduling window 2500 tasks
• Dataset
  • 100K files
  • 1 byte each
• Tasks
  • Read 1 file
  • Write 1 file

Workloads

• Monotonically Increasing Workload
  – Emphasizes increasing loads
• Sine-Wave Workload
  – Emphasizes varying loads
• All-Pairs Workload
  – Compare to best case model of active storage
• Image Stacking Workload (Astronomy)
  – Evaluate data diffusion on a real large-scale data-intensive application from astronomy domain

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
- 250K tasks
  - 10MB reads
  - 10ms compute
- Vary arrival rate:
  - Min: 1 task/sec
  - Increment function: CEILING(*1.3)
  - Max: 1000 tasks/sec
- 128 processors
- Ideal case:
  - 1415 sec
  - 80Gb/s peak throughput
Monotonically Increasing Workload
First-available (GPFS)

- GPFS vs. ideal: 5011 sec vs. 1415 sec

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
“The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
“Towards Data Intensive Many-Task Computing”, under review
Monotonically Increasing Workload

Good-cache-compute

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review

Data Diffusion vs. ideal: 1436 sec vs 1415 sec
Monotonically Increasing Workload
Throughput and Response Time

Throughput:
- Average: 14Gb/s vs 4Gb/s
- Peak: 81Gb/s vs. 6Gb/s

Response Time ➔
- 3 sec vs 1569 sec ➔ 506X

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
[DiDC09] “Towards Data Intensive Many-Task Computing”, under review
• Performance Index:
  – 34X higher
• Speedup
  – 3.5X faster than GPFS

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
[DiDC09] “Towards Data Intensive Many-Task Computing”, under review
• 2M tasks
  – 10MB reads
  – 10ms compute
• Vary arrival rate:
  – Min: 1 task/sec
  – Arrival rate function:
  – Max: 1000 tasks/sec
• 200 processors
• Ideal case:
  – 6505 sec
  – 80Gb/s peak throughput

\[ A = \left(\sin(\sqrt{\text{time} + 0.11}) \times 2.859678 + 1\right) \times (\text{time} + 0.11) \times 5.705 \]
GPFS \(\Rightarrow\) 5.7 hrs, \(~8\text{Gb/s}, 1138\text{ CPU hrs}\)

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review

Sine-Wave Workload
Good-cache-compute and SRP

- GPFS: 5.7 hrs, ~8Gb/s, 1138 CPU hrs
- GCC+SRP: 1.8 hrs, ~25Gb/s, 361 CPU hrs
Sine-Wave Workload
Good-cache-compute and DRP

- GPFS ➔ 5.7 hrs, ~8Gb/s, 1138 CPU hrs
- GCC+SRP ➔ 1.8 hrs, ~25Gb/s, 361 CPU hrs
- GCC+DRP ➔ 1.86 hrs, ~24Gb/s, 253 CPU hrs

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
All-Pairs Workload

• 500x500
  – 250K tasks
  – 24MB reads
  – 100ms compute
  – 200 CPUs

• 1000x1000
  • 1M tasks
  • 24MB reads
  • 4sec compute
  • 4096 CPUs

• Ideal case:
  – 6505 sec
  – 80Gb/s peak throughput

• All-Pairs( set A, set B, function F ) returns matrix M:
  • Compare all elements of set A to all elements of set B via function F, yielding matrix M, such that

\[
M[i,j] = F(A[i],B[j])
\]

1 foreach $i$ in A
2   foreach $j$ in B
3     submit_job F $i$ $j$
4   end
5 end

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
All-Pairs Workload
500x500 on 200 CPUs

Efficiency: 75%

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
All-Pairs Workload
1000x1000 on 4K emulated CPUs

Efficiency: 86%

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
All-Pairs Workload

Data Diffusion vs. Active Storage

- Pull vs. Push
  - Data Diffusion
    - Pulls *task* working set
    - Incremental spanning forest
  - Active Storage:
    - Pushes *workload* working set to all nodes
    - Static spanning tree

Christopher Moretti, Douglas Thain, University of Notre Dame

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review

All-Pairs Workload Data Diffusion vs. Active Storage

- Best to use active storage if
  - Slow data source
  - Workload working set fits on local node storage

- Best to use data diffusion if
  - Medium to fast data source
  - Task working set $<$ workload working set
  - Task working set fits on local node storage

- If task working set does not fit on local node storage
  - Use parallel file system (i.e. GPFS, Lustre, PVFS, etc)

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review
Data Diffusion vs. Others

- [Ghemawat03,Dean04]: MapReduce+GFS
- [Bialecki05]: Hadoop+HDFS
- [Gu06]: Sphere+Sector
- [Tatebe04]: Gfarm
- [Chervenak04]: RLS, DRS
- [Kosar06]: Stork

Conclusions

- None focused on the co-location of storage and generic black box computations with data-aware scheduling while operating in a dynamic elastic environment
- Swift + Falkon + Data Diffusion is arguably a more generic and powerful solution than MapReduce
Image Stacking Workload
Astronomy Application

- **Purpose**
  - On-demand “stacks” of random locations within ~10TB dataset

- **Challenge**
  - Processing Costs:
    - O(100ms) per object
  - Data Intensive:
    - 40MB:1sec
  - Rapid access to 10-10K “random” files
  - Time-varying load

---

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
[TG06] “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”
Image Stacking Workload Profiling

Filesystem and Image Format

GPFS GZ
LOCAL GZ
GPFS FIT
LOCAL FIT

open
radec2xy
readHDU+getTile+curl+convertArray
calibration+interpolation+doStacking
writeStacking

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
Low data locality ➔
- Similar (but better) performance to GPFS

High data locality
- Near perfect scalability

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
• Aggregate throughput:
  – 39Gb/s
  – 10X higher than GPFS
• Reduced load on GPFS
  – 0.49Gb/s
  – 1/10 of the original load

• Big performance gains as locality increases

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
Stacking service (large scale astronomy application)

- 92 experiments, 558K files
  - Compressed: 2MB each → 1.1TB
  - Un-compressed: 6MB each → 3.3TB

[HPDC09] “The Quest for Scalable Support of Data Intensive Applications in Distributed Systems”, under review

Limitations of Data Diffusion

- Data access patterns: write once, read many
- Task definition must include input/output files metadata
- Per task working set must fit in local storage
- Needs IP connectivity between hosts
- Needs local storage (disk, memory, etc)
- Needs Java 1.4+
Contributions

• Identified that data locality is crucial to the efficient use of large scale distributed systems for data-intensive applications ➔ Data Diffusion
  – Integrated streamlined task dispatching with data aware scheduling policies
  – Heuristics to maximize real world performance
  – Suitable for varying, data-intensive workloads
  – Proof of $O(NM)$ Competitive Caching

• There is more to HPC than tightly coupled MPI, and more to HTC than embarrassingly parallel long jobs ➔ Many-Task Computing
• Does data diffusion apply to CDNs?
• Can we harness data locality for workloads that target data retrieval?
• …
More Information

• More information: http://people.cs.uchicago.edu/~iraicu/
• Related Projects:
  – Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  – Swift: http://www.ci.uchicago.edu/swift/index.php
• Dissertation Committee:
  – Ian Foster, The University of Chicago & Argonne National Laboratory
  – Rick Stevens, The University of Chicago & Argonne National Laboratory
  – Alex Szalay, The Johns Hopkins University
• Funding:
  – NASA: Ames Research Center, Graduate Student Research Program
    • Jerry C. Yan, NASA GSRP Research Advisor
  – NSF: TeraGrid