Failure-Awareness and Dynamic Adaptation in Data Scheduling

Mehmet Balman
MS Thesis
Department of Computer Science
Louisiana State University

Research Goal

“Reliability and Efficiency” for wide-area Data Access
Outline

• The Data Placement Challenge
• Lessons Learned from Computer Architecture
• Adaptive Data Scheduling
• Failure-Aware Data Placement
• Conclusion

Large Scale Applications

• Science
  • Astronomy - SuperNova, LSST (Large Synoptic Survey Telescope)
  • Biology (bimolecular computing)
  • Climate research
  • High Energy Physics (Cern)
• Business
  • Credit Card Fraud detection
    • (historical data, analyze transactions)
  • Data mining for brokerage and customer services
  • Oil and electronic design companies
    • (long term batch processes)
  • Medical institutions
    • (computational network, large image transfers)
**Data Deluge**

- Scientific and Business applications becoming more data-intensive
- Huge Computational requirements
- Immense data sets (real time processing of data)

---

**Data-intensive Computing**

- Using Distributed Resources to satisfy excessive computation requirements
- Data to be shared between geographically distributed sites
- Complex workflow characteristics
- High capacity, fast storage systems
Data Scheduling

• Make data placement a first class citizen

• Orchestrating data placement jobs

Stork  [www.storkproject.org](http://www.storkproject.org)

---

Data-Aware System Model
Key Attributes affecting Data Placement Performance

<table>
<thead>
<tr>
<th></th>
<th>In Single Host</th>
<th>Between a Pair of Hosts</th>
<th>Multiple Servers to Single Server</th>
<th>Between Distributed Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage Space</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Load and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Usage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Parallel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Bandwidth and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Concurrent Operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ordering of</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Placement Tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Contribution

- Failure-Aware Data Placement Paradigm for increased Fault-Tolerance
- Adaptive Scheduling of Data Placement Tasks
Outline

• The Data Placement Challenge
• Lessons Learned from Computer Architecture
• Adaptive Data Scheduling
• Failure-Aware Data Placement
• Conclusion

Generic Model
Outline

- The Data Placement Challenge
- Lessons Learned from Computer Architecture
- Adaptive Data Scheduling
- Failure-Aware Data Placement
- Conclusion
Adaptive Scheduling

- Dynamic Parameter Tuning
  - Parallel Stream
    - Aggregate TCP connections
  - Concurrent Jobs

- Aggregation of Data Placement Job
  - Source/Destination pair

Impact of Parallelism
Concurrent Jobs

- Low integration cost (no external profilers)
- Adapt to changing network conditions
- No high level predictors
- Increase level of parallelism gradually
  - Can we set the number of parallel streams while transfer is in progress?
Job Aggregation

- Aggregate data transfer jobs into a single job
- Eliminate the cost of connection for each transfer
- Major performance improvement
  - Especially with small files

![Job Aggregation Graph](image)
Outline

• The Data Placement Challenge
• Lessons Learned from Computer Architecture
• Adaptive Data Scheduling
• Failure-Aware Data Placement
• Conclusion

Failure-Awareness

• Early Error Detection
  – Network Exploration
• Error Classification and Reporting
• Adapt to Failures (Retry?)
Error Reporting Framework

Data Transfer Operations

Data Transfer Life Cycle

Tracing Data Transfer Operations
Integration

Structural Failure Detection and Error Reporting Framework

Operation
- Identify causal triggering	Simple
- Identify causal triggering	Detailed
- Report
- Error Message

Data Transfer Operations
- Initialize protocol handles
- Select features supported by the protocol
- Tune-up and configure protocol specific parameters
- Get status information about a data resource
- Perform data transmission operation
- Execute accuracy of the data transfer
- Finishes and clear handles

Data Transfer Life Cycle

Failure-Awareness

Graph showing the number of jobs submitted and completed over time.
Outline

• The Data Placement Challenge
• Lessons Learned from Computer Architecture
• Adaptive Data Scheduling
• Failure-Aware Data Placement
• Conclusion

Conclusion

• An Adaptive Approach for Parameter Tuning
• Early Error Detection and Error Classification
• Failure-Awareness in Scheduling
• Aggregation of Data Placement Jobs
Broader Impact

- **Stork**
  - [http://www.storkproject.org/](http://www.storkproject.org/)
- **Petashare (petaFS & petaShell)**
- I/O aggregation
  - **IRODS FUSE** and **IRODS Parrot** clients
    - 3-fold performance increase
- **Stork.globus-url-copy**
  - Extending **globus-url-copy**
    - New features:
      - Checkpointing (rescue file for restart)
      - Network explorations
      - Checksum verification
      - Auto Tuning the number of Parallel Streams

Future Research Problems

- **Semantic Compression**
  - For better end-to-end performance
- **Utilizing Replicated Data**
- **Distributed Scheduling**
  - Job delegation