A Result-Data Offloading Service for HPC Centers

Henry Monti, Ali R. Butt

Sudharshan S. Vazhkudai

[Logos for Virginia Tech and Oak Ridge National Laboratory]
HPC Center Data Offload Problem

- Supercomputer serviceability affected by data offloading errors
  - Offloading is a large data job prone to failure
    - End resource unavailability
    - Transfer errors
  - Delayed offloading
    - From a center standpoint
      - Wastes scratch space
      - Renders result data vulnerable to purging
    - From a user job standpoint
      - Increased turnaround time if part of the job workflow depends on offloaded data
      - Potential resubmits due to purging

- Upshot: Timely offloading can help improve center performance
  - HPC acquisition solicitations are asking for stringent uptime and resubmission rates (NSF06-573)
Current Methods For Data Offloading

- Home grown solutions
  - Every center has its own

- Utilize point-to-point transfer tools:
  - GridFTP
  - HSI
  - scp
  - ...

Virginia Tech
Invent the Future
Limitations of Direct Transfers

- Require end resources to be available
- Do not exploit orthogonal bandwidth
- Do not consider SLAs or purge times
- Not an ideal solution for data-offloading
Our Contribution:
Decentralized Data-Offloading Service

- Utilize army of intermediary storage locations
- Offload data to nearby nodes
- Support multi-hop data migration to end user
- Allow end user to retrieve data as necessary

- Provide multiple fault-tolerant data flow paths from the center to the end user
Challenges Faced in Our Approach

- Discovering intermediary nodes
- Addressing insufficient participants
- Adapting to dynamic network behavior
- Ensuring data reliability and availability
Overlay Networks

P2P networks are self-organizing overlay networks without central control
Structured P2P Overlays

- Overlays with imposed structure
  - Each node has a unique random `nodeId`
  - Each message has a key
  - The `nodeId` and key reside in the same namespace

- Routing: Takes a message with a key and sends it to a unique node

- Implements Distributed Hash Table (DHT) abstraction
  - DHT abstraction is preserved in the presence of node failure/departure
  - Many implementations available, e.g. Pastry, Tapestry, Chord, CAN …
Intermediary Node Discovery

- Utilize DHT abstraction
- Nodes advertise their availability to others
- Receiving nodes *discovers* the advertiser
- Discovered nodes utilized as necessary

$2^{128} - 10$
What if there aren’t enough participants?

- **Use Landmark Nodes**
  - Nodes that are always available
  - Willing to store data

- **Leverage out-of-band agreements**
  - Other researchers who are also interested in the data
  - Data warehouses
    - cheaper option than storing at the HPC center

- **These nodes are a safety net!**
Adapting Data Distribution To Dynamic Network Behavior

- Available bandwidth can change
  - A simple random distribution may not be effective
  - Utilize network monitoring

- Network Weather Service (NWS)
  - Provides bandwidth Measurement
  - Predicts future bandwidth

- Choose dynamically changing data paths
- Select enough nodes to satisfy a given SLA
Protecting Data from Intermediate Storage Location Failure

- Use data replication
  - Achieved through multiple data flow paths

- Employ Erasure coding
  - Can be done at the Center or intermediaries
  - End user may pay for coding at the Center
Evaluation: Experimental Setup

- PlanetLab test bed
  - 22 PlanetLab nodes
    center + end user + 20 intermediary nodes

- Experiments:
  Compare point-to-point with the proposed method
  1. Random distribution
  2. Bandwidth measurement based
  3. Bandwidth forecasts based
## Results: Data Transfer Times

Times are in seconds
Transfer of a 95 MB file

<table>
<thead>
<tr>
<th></th>
<th>Direct</th>
<th>Random</th>
<th>Measurement Based</th>
<th>Forecast Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offload</td>
<td>739</td>
<td>245</td>
<td>214</td>
<td>210</td>
</tr>
<tr>
<td>Push</td>
<td>N/A</td>
<td>431</td>
<td>393</td>
<td>370</td>
</tr>
<tr>
<td>Pull</td>
<td>739</td>
<td>665</td>
<td>663</td>
<td>663</td>
</tr>
</tbody>
</table>
Conclusion

- A fresh look at Offloading
  - Decentralized approach
  - Monitoring-based adaptation
- Considers SLAs and purge policies
- Provides high reliability for data
- Outperforms direct transfer by 72%
Future Work

- Strategically placed Landmark nodes
- Schedule offload to coincide job completion
- Eager offloading
- Integration with job script

Contact
- Virginia Tech.
  - Distributed Systems and Storage Lab.
    http://research.cs.vt.edu/dssl/
  - {hmonti, butta}@cs.vt.edu
- ORNL
  - vazhkudaiss@ornl.gov