A Large Scale Fault-Tolerant Grid Information Service

Francisco Brasileiro, Lauro Beltrão
Alisson Andrade, Walfredo Cirne
{fubica,lauro,alisson,walfredo}@dsc.ufcg.edu.br

Universidade Federal de Campina Grande, Brazil

Sujoy Basu, Sujata Banerjee
{Sujoy.Basu,Sujata.Banerjee}@hp.com

Hewlett-Packard Laboratories, USA

Motivation

– OurGrid is a P2P grid middleware developed mainly at LSD/UFCG that envisions to gather hundreds of thousands of resources in tens of thousands of peers (see http://www.ourgrid.org/)
Motivation

• **How to discover** “entities” with particular attributes in a large P2P grid system?
  – A scheduler might want to locate suitable resources
    • OS=linux && RAM ≥ 1G && clock > 4GHz && load < 0.5
  – A user may want to locate a dataset that contains particular data items
    • rainfall && -37°52’ < long < -37°46’ && 144°54’ < lat < 145°03’
  – …

Motivation

• **To support this service a large scale** Grid Information Service must
  – **Scale** to large amounts of meta-data and to high operation load
  – Be able to execute **rich queries** that encompass not only multiple attributes, but also range operators
  – Tolerate **faults**
Centralized/static hierarchical GISs

- Support rich queries
- Fault tolerance may be implemented via the use of redundant servers
- **Scale poorly** (at a reasonable cost)

P2P GISs

- DHT-based
  - Scale well
  - Highly reliable
  - Do not support rich queries in a natural way
- Distributed tree-based
  - Scale well
  - Support rich queries
  - Do not tolerate faults
Our Approach

- **Extend** an existent kd-tree based GIS with fault tolerance mechanisms
- Do that **without impacting scalability** and the ability to **support rich queries**

Outline of the rest of this presentation

- Explain the functioning of NodeWiz
- Analyze the impact of faults in NodeWiz
- Explain the fault tolerance mechanism proposed
- Show the results of an experiment run with the fault-tolerant version of NodeWiz
- Present conclusions and directions for future research
NodeWiz’s Rationale

- Providers advert their service attributes
- Each NodeWiz node is responsible for a portion of the attribute subspace
- Clients query for services with particular attributes
- Adverts have a TTL and should be renewed
- Peers maintain routing tables to route operations

Performing an operation

- Query sent to A: load < 0.4 && Mem >= 2GB && Clock >= 2 Ghz

<table>
<thead>
<tr>
<th>Peer A</th>
<th>Level</th>
<th>Attr</th>
<th>Range</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Load</td>
<td>0.6 - INF</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mem</td>
<td>2 - INF</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Load</td>
<td>0.3 - 0.6</td>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Peer C</th>
<th>Level</th>
<th>Attr</th>
<th>Range</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Load</td>
<td>0.6 - INF</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mem</td>
<td>0 - 2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Clock</td>
<td>2 - INF</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Peer F</th>
<th>Level</th>
<th>Attr</th>
<th>Range</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Load</td>
<td>0.6 - INF</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mem</td>
<td>0 - 2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Clock</td>
<td>0 - 2</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>
Joining a NodeWiz Network

• Identify the most **overloaded peer**

• Balance load with it

• Identify attribute subspace that can **improve load balance**

Say, node H wants to join the system

- It asks one node it knows, which is the most overloaded node
- It then joins the system at this node
On the consequences of faults

- For the sake of analysis we assume
  - System is well-formed
    - all routing tables have the same size (L=\(\log_2 N\))
  - Load is well-balanced
    - all peers receive the same load
    - All queries match a single peer
  - Probability of unsuccessful operation issued to \(P\):
    - \(POU(P) = \sum_{i=0}^{L} \frac{R_i(P)}{N} \cdot [1 - (1 - f)^{i+1}]\)
The impact of faults

The probability of unsuccessful operations as a function of the number of peers for different failure rates (f) is shown in the graph. The graph illustrates how the probability increases with the number of peers for all failure rates tested (f=10.0%, f=2.5%, f=1.0%).
Voluntary leaves

• When a peer L leaves, its attribute subspace must be reclaimed by another peer
• L’s replacement - R(L) - is the last peer in L’s routing table
• L contacts R(L) informing about its departure
• R(L) stores L’s state and propagates L’s request for other peers that have L in their routing table

Voluntary leaves

• During leaving process, routing tables may become inconsistent
• L is kept in the system until it receives authorization to leave from R(L)
• L also forwards new adverts or queries to R(L)
• R(L) waits for acknowledges from other peers before authorizing L’s departure
Voluntary leaves: B’s departure

\[
\text{Load} \geq 0.6
\]

\[
\text{BW} < 1
\]

\[
\text{Mem} < 1
\]

\[
\text{BW} \geq 1
\]

\[
\text{Mem} \geq 1
\]
Dealing with involuntary leaves

- We are not concerned with the state of faulty peers
- Our approach is: $R(L)$ must detect that $L$ has failed, so $R(L)$ monitors $L$
- When $L$ fails, $R(L)$ creates a virtual peer $S(L)$ that assumes $L$'s identity
- $S(L)$ initiates a voluntary leave with $R(L)$

Dealing with simultaneous faults

- What if both $L$ and $R(L)$ fail at “the same time”?
Dealing with simultaneous faults

- A peer $P$ monitors a set of peers where each set has a leader that may be replaced by $P$
- $P$ monitors $L$ sets, where $L$ is the size of $P$’s routing table
  - Each peer in its routing table is a set leader
- Each set is composed by every peer that can replace the set leader before $P$ replaces it

- Example:

<table>
<thead>
<tr>
<th>Level</th>
<th>Attr</th>
<th>Range</th>
<th>Route/Set leader</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Load</td>
<td>0.6 - INF</td>
<td>B</td>
<td>{D,G,H}</td>
</tr>
<tr>
<td>1</td>
<td>Mem</td>
<td>2 - INF</td>
<td>C</td>
<td>(F)</td>
</tr>
<tr>
<td>2</td>
<td>Load</td>
<td>0.3 – 0.6</td>
<td>E</td>
<td>{}</td>
</tr>
</tbody>
</table>

Experiments

- Performed in PlanetLab in a system with up to 64 peers
- Adverts came from PlanetLab data collected by Ganglia
- Experiments have a start up phase and a performance collection phase
  - In the start up phase peers join the system while adverts are stored (no TTL)
  - In the collection phase queries are sent randomly to peers in the system
  - Half of data is used in each phase
Conclusions and future work

• Faults have a great impact on the QoS of tree-based GIS
• The fault tolerance mechanisms proposed solve the problem without impacting scalability
• OurGrid 4.0 will incorporate the fault-tolerant version of NodeWiz to perform resource discovery
• We intend to perform experiments to compare FTNodeWiz with DHT-based GISs in fault-prone scenarios
Questions?

• Contact information:
  – Francisco Brasileiro
    • fubica@dsc.ufcg.edu.br
  – LSD/UFCG
    • http://lsd.ufcg.edu.br/
  – OurGrid project
    • http://www.ourgrid.org/
  – For a real-time snapshot of the running system access http://status.ourgrid.org/