Scaling Social Nets

Pablo Rodriguez
Telefonica Research, Barcelona

Social Networks: Rapid Growth

- Facebook increased from 100M to 200M in less than 8 months
- Radio took 38 years to reach 50M, TV 13 years, Internet 4 years, iPod 3 and Facebook 2!
- Ashton Kutcher, first Twitter celebrity has more than 1M followers
- Nielsen research, says social networks are more popular than email
- Facebook is replacing Google as a tool to find content: my friends filter the Web for me.
Towards transparent scalability

Elastic resource allocation for the Presentation and the Logic layer:
More machines when load increases.
Components are stateless, therefore, independent and duplicable

Components are interdependent, therefore, non-duplicable on demand.
Scalability is a pain: the designers’ dilemma

If data is *local to a machine*
- Centralized programming paradigm can be used
- All read queries resolved locally without requests sent to other machines
- Management simple

If data is *not local* (distributed)
- Distributed programming – not trivial!
- Management costs increases
- *Performance degrades*

Why sharding is not enough for OSN?

Shards in OSN can never be disjoint because:
- Operations on user \(i\) require access to the data of other users
- at least one-hop away.

From graph theory:
- there is no partition that for all nodes all neighbors and itself are in the same partition if there is a single connected component.

Data locality cannot be maintained by partitioning a social network!!
### OSN’s operations 101

#### 1) Relational Databases

- Selects and joins across multiple shards of the database are possible but performance is poor (e.g. MySQL Cluster, Oracle Rack)

#### 2) Key-Value Stores (DHT)

- More efficient than relational databases: multi-get primitives to transparently fetch data from multiple servers.

- **But it’s not a silver bullet:**
  - lose SQL query language => programmatic queries
  - lose abstraction from data operations
  - suffer from high traffic, eventually affecting performance:
    - Incast issue
    - Multi-get hole
    - latency dominated by the worse performing server

### Maintaining Data Locality Semantics

Sketch of a Social Network to be split in two servers...
Maintaining Data Locality Semantics

Sketch of a Social Network to be split in two servers...

Full Replication (typical RDBMS)

Random Partition (typical Key-Value)
Performance problems...

- Network bandwidth is not an issue but Network I/O is, and CPU I/O too:
  - Network Latency increases
    - worse performing server produces delays
    - multiget hole
  - Memory hit ratio is decreased
    - random partition destroys correlations

Maintaining Data Locality Semantics

Sketch of a Social Network to be split in two servers...

- Full Replication (typical RDBMS)
- Random Partition (typical Key-Value)
- Random Partition + Replication
## Maintaining Data Locality Semantics

![Sketch of a Social Network to be split in two servers...](image)

- **Full Replication (typical RDBMS)**
- **Random Partition (typical Key-Value)**
- **Random Partition + Replication**
- **SPAR** (Social-based Partition + Replication)

### SPAR Algorithm from 10000 feet

<table>
<thead>
<tr>
<th>1) Online (incremental)</th>
<th>2) Fast (and simple)</th>
<th>3) Stable and Fair</th>
<th>4) Effective</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Dynamics of the SN (add/remove node, edge)</td>
<td>a) Local information</td>
<td>a) Avoid cascades</td>
<td>a) Optimize for MIN_REPLICA (<em>i.e. NP-Hard</em>)</td>
</tr>
<tr>
<td>b) Dynamics of the System (add/remove server)</td>
<td>b) Hill-climbing heuristic</td>
<td>b) Fair allocation of load</td>
<td>b) While maintaining a fixed number of replicas per user for redundancy (<em>e.g. K=2</em>)</td>
</tr>
<tr>
<td></td>
<td>c) Load-balancing via back-pressure</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Graph/Social partitioning algorithms fall short...

1) SPAR
   - incremental (online)
   - stable (minimize migrations)
   - simple (and fast)

2) SPAR optimizes different goal: replicas rather than edges

MIN_REPLICA Problem

SPAR Online from 10,000 feet

MIN_REPLICA is NP-Hard 😊

Heuristic:
- Greedy optimization
- Local information
- Load balance constrain

Six events:
- Add/Remove

Status quo: replica of 1 in M3 and replica of 6 in M1.
SPAR in the wild

**Twitter clone** (Laconica, now Statusnet)
- Centralized architecture, PHP + MySQL/Postgres

**Twitter data**
- Twitter as of end of 2008, 2.4M users, 12M tweets in 15 days

**The little engines**
- 16 commodity desktops: Pentium Duo at 2.33Ghz, 2GB RAM connected with Gigabit-Ethernet switch

Test SPAR on top of:
- MySQL (v5.5)
Trying out various partitioning algorithms...

Real OSN data

Twitter
- 2.4M users, 48M edges, 12M tweets for 15 days (Twitter as of Dec08)

Orkut (MPI)
- 3M users, 223M edges

Facebook (MPI)
- 60K users, 500K edges

Partition algorithms

- Random (DHT)
- METIS
  - spectral clustering
- MO+
  - modularity optimization + recursive
- SPAR online

How many replicas are generated?

<table>
<thead>
<tr>
<th>Rep. overhead</th>
<th>% over K=2</th>
<th>Rep. overhead</th>
<th>% over K=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAR</td>
<td>2.44</td>
<td>+22%</td>
<td>3.69</td>
</tr>
<tr>
<td>MO+</td>
<td>3.04</td>
<td>+52%</td>
<td>5.14</td>
</tr>
<tr>
<td>METIS</td>
<td>3.44</td>
<td>+72%</td>
<td>8.03</td>
</tr>
<tr>
<td>Random</td>
<td>5.68</td>
<td>+184%</td>
<td>10.75</td>
</tr>
</tbody>
</table>
How many replicas are generated?

<table>
<thead>
<tr>
<th>Replication Method</th>
<th>Twitter 16 partitions</th>
<th>Rep. overhead</th>
<th>% over K=2</th>
<th>Twitter 128 partitions</th>
<th>Rep. overhead</th>
<th>% over K=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAR</td>
<td>2.44</td>
<td>+22%</td>
<td>3.69</td>
<td>+84%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO+</td>
<td>3.04</td>
<td>+52%</td>
<td>5.14</td>
<td>+157%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METIS</td>
<td>3.44</td>
<td>+72%</td>
<td>8.03</td>
<td>+302%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>5.68</td>
<td>+184%</td>
<td>10.75</td>
<td>+434%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.44 replicas per user (on average)
2 to guarantee redundancy
+ 0.44 to guarantee data locality (+22%)

Replication with random partitioning, too costly!
### How many replicas are generated?

<table>
<thead>
<tr>
<th></th>
<th>Twitter 16 partitions</th>
<th>Twitter 128 partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rep. overhead</td>
<td>% over K=2</td>
</tr>
<tr>
<td>SPAR</td>
<td>2.44</td>
<td>+22%</td>
</tr>
<tr>
<td>MO+</td>
<td>3.04</td>
<td>+52%</td>
</tr>
<tr>
<td>METIS</td>
<td>3.44</td>
<td>+72%</td>
</tr>
<tr>
<td>Random</td>
<td>5.68</td>
<td>+184%</td>
</tr>
</tbody>
</table>

Other social based partitioning algorithms have higher replication overhead than SPAR.

Replication overhead grows sub-linearly with the number of partitions.
SPAR on top of Cassandra

• Different read request rate (application level):

<table>
<thead>
<tr>
<th></th>
<th>Vanilla Cassandra</th>
<th>SPAR + Cassandra</th>
</tr>
</thead>
<tbody>
<tr>
<td>99th &lt; 100ms</td>
<td>200 req/s</td>
<td>800 req/s</td>
</tr>
</tbody>
</table>

Network bandwidth is not an issue but Network I/O and CPU I/O are:

- Network delays is reduced
- Worse performing server produces delays
- Memory hit ratio is increased
- Random partition destroys correlations

Conclusions

SPAR provides the means to achieve transparent scalability

1) For applications using RDBMS (not necessarily limited to OSN)
2) And, a performance boost for key-value stores due to the reduction of Network I/O
Questions? Thanks.