DONAR
Decentralized Server Selection for Cloud Services

Patrick Wendell, Princeton University

Joint work with Joe Wenjie Jiang, Michael J. Freedman, and Jennifer Rexford
Outline

• Server selection background

• Constraint-based policy interface

• Scalable optimization algorithm

• Production deployment
User Facing Services are Geo-Replicated
Reasoning About Server Selection

Client Requests

Mapping Nodes

Service Replicas
Example: Distributed DNS

Clients

DNS Resolvers

- Client 1
- Client 2
- Client C

Mapping Nodes

Auth. Nameservers

- DNS 1
- DNS 2
- DNS 10

Service Replicas

Servers

Mapping Nodes

- DNS 1
- DNS 2
- DNS 10

Service Replicas
Example: HTTP Redir/Proxying

Clients

HTTP Clients
- Client 1
- Client 2
- Client C

Mapping Nodes

HTTP Proxies
- Proxy 1
- Proxy 2
- Proxy 500

Service Replicas

Datacenters
Reasoning About Server Selection
Reasoning About Server Selection

Client Requests

Mapping Nodes

Outsource to DONAR

Service Replicas
Outline

• Server selection background

• Constraint-based policy interface

• Scalable optimization algorithm

• Production deployment
Naïve Policy Choices
Load-Aware: “Round Robin”
Naïve Policy Choices
Location-Aware: “Closest Node”

Goal: support complex policies across many nodes.
Policies as Constraints

DONAR Nodes

- bandwidth_cap = 10,000 req/m

split_ratio = 10%
allowed_dev = ± 5%

Replicas
Eg. 10-Server Deployment

How to describe policy with constraints?
No Constraints
Equivalent to “Closest Node”

Requests per Replica

- 28%
- 35%
- 2%
- 6%
- 10%
- 1%
- 1%
- 7%
- 2%
- 9%
No Constraints
Equivalent to “Closest Node”

Impose 20% Cap

Requests per Replica

2% 6% 10% 1% 7% 2% 28% 35%
Cap as Overload Protection

Requests per Replica

2% 6% 10% 1% 1% 7% 14% 20% 20% 20%
12 Hours Later...

Requests per Replica

- 29% in green
- 16% in red
- 16% in yellow
- 12% in maroon
- 10% in orange
- 5% in blue
- 4% in light blue
- 3% in black
- 3% in purple
- 3% in maroon
"Load Balance"
(split = 10%, tolerance = 5%)

<table>
<thead>
<tr>
<th>Requests per Replica</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
</tr>
<tr>
<td>5%</td>
</tr>
<tr>
<td>5%</td>
</tr>
<tr>
<td>5%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>15%</td>
</tr>
</tbody>
</table>
“Load Balance”
(split = 10%, tolerance = 5%)

Trade-off network proximity & load distribution

Requests per Replica

5% 5% 5% 5% 5% 15% 15% 15% 15% 15%
12 Hours Later...

Large range of policies by varying cap/weight

Requests per Replica

7% 15% 15% 15% 5% 13% 5% 10% 10% 5%
Outline

• Server selection background

• Constraint-based policy interface

• Scalable optimization algorithm

• Production deployment
Optimization: Policy Realization

- Global LP describing “optimal” pairing

Minimize network cost

\[
\min \sum_{c \in C} \sum_{i \in I} \alpha_c \cdot R_{ci} \cdot \text{cost}(c, i)
\]

s.t.

- Server loads within tolerance

\[
|P_i - \omega_i| \leq \varepsilon_i
\]

- Bandwidth caps met

\[
B_i < B \cdot P_i
\]
Optimization Workflow

Measure Traffic  Track Replica Set  Calculate Optimal Assignment
Optimization Workflow

Measure Traffic

Track Replica Set

Calculate Optimal Assignment

Per-customer!
Optimization Workflow

- Measure Traffic
- Track Replica Set
- Calculate Optimal Assignment

Continuously!

(respond to underlying traffic)
## By The Numbers

<table>
<thead>
<tr>
<th></th>
<th>$10^1$</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DONAR Nodes</td>
<td></td>
<td></td>
<td>⬤</td>
<td></td>
</tr>
<tr>
<td>Customers</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicas/customer</td>
<td></td>
<td>⬤</td>
<td></td>
<td></td>
</tr>
<tr>
<td>client groups/customer</td>
<td></td>
<td></td>
<td></td>
<td>⬤</td>
</tr>
</tbody>
</table>

**Problem for each customer:**

$10^2 \times 10^4 = 10^6$
Measure Traffic & Optimize Locally?
Not Accurate!

No one node sees entire client population
Aggregate at Central Coordinator?

Mapping Nodes → Service Replicas
Aggregate at Central Coordinator?

Share Traffic Measurements ($10^6$)

Mapping Nodes

Service Replicas
Aggregate at Central Coordinator?

Optimize
Aggregate at Central Coordinator?

Return assignments \((10^6)\)
<table>
<thead>
<tr>
<th></th>
<th>Accurate</th>
<th>Efficient</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local only</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Central Coordinator</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Decomposing Objective Function

\[\min_{\alpha, \mathbf{c}} \sum_{i \in I} R_{ci} \cdot \cos(\mathbf{c}, i)\]

\[\sum_{c \in C} \sum_{i \in I} \alpha_{cn} \cdot R_{nci} \cdot \cos(\mathbf{c}, i)\]

∀ clients

∀ instances

Traffic from c
cost of mapping c to i

Traffic to this node

We also decompose constraints
(more complicated)
Decomposed Local Problem
For Some Node \((n^*)\)

\[ \text{load}_i = f(\text{prevailing load on each server} + \text{load I will impose on each server}) \]

\[
\min \forall_i \text{load}_i + s_{n^*} \sum_{c \in C} \sum_{i \in I} \alpha_{cn^*} \cdot R_{n^*ci} \cdot cost(c, i)
\]
DONAR Algorithm

Solve local problem

Mapping Nodes

Service Replicas
DONAR Algorithm

- Solve local problem
- Share summary data with others ($10^2$)

Mapping Nodes

Service Replicas
DONAR Algorithm

Solve local problem

Mapping Nodes

Service Replicas
DONAR Algorithm

- Share summary data w/ others ($10^2$)

Diagram:
- Mapping Nodes
- Service Replicas
DONAR Algorithm

- Provably converges to global optimum
- Requires no coordination
- Reduces message passing by $10^4$
### Better!

<table>
<thead>
<tr>
<th></th>
<th>Accurate</th>
<th>Efficient</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local only</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Central Coordinator</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DONAR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Outline

• Server selection background

• Constraint-based policy interface

• Scalable optimization algorithm

• Production deployment
Production and Deployment

• Publicly deployed 24/7 since November 2009

• IP2Geo data from Quova Inc.

• Production use:
  – All MeasurementLab Services (incl. FCC Broadband Testing)
  – CoralCDN

• Services around 1M DNS requests per day
Systems Challenges (See Paper!)

• Network availability
  *Anycast with BGP*

• Reliable data storage
  *Chain-Replication with Apportioned Queries*

• Secure, reliable updates
  *Self-Certifying Update Protocol*
CoralCDN Experimental Setup

Client Requests

DONAR Nodes

CoralCDN Replicas

split_weight = .1

tolerance = .02
Results: DONAR Curbs Volatility

“Closest Node” policy

DONAR “Equal Split” Policy
Results: DONAR Minimizes Distance

- **Requests per Replica**
- **Ranked Order from Closest**
- **Minimal (Closest Node)**
- **DONAR**
- **Round-Robin**
Conclusions

• Dynamic server selection is difficult
  – Global constraints
  – Distributed decision-making

• Services reap benefit of outsourcing to DONAR.
  – Flexible policies
  – General: Supports DNS & HTTP Proxying
  – Efficient distributed constraint optimization

• Interested in using? Contact me or visit http://www.donardns.org.
Questions?
Related Work (Academic and Industry)

• Academic
  – Improving network measurement
    • iPlane: An informationplane for distributed services
      H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson,
      A. Krishnamurthy, and A. Venkataramani, “,” in OSDI, Nov. 2006
  – “Application Layer Anycast”
    • OASIS: Anycast for Any Service
      Michael J. Freedman, Karthik Lakshminarayanan, and David Mazières
      Proc. 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation
      (NSDI ’06) San Jose, CA, May 2006.

• Proprietary
  – Amazon Elastic Load Balancing
  – UltraDNS
  – Akamai Global Traffic Management
Doesn’t [Akamai/UltraDNS/etc] Already Do This?

- Existing approaches use alternative, centralized formulations.
- Often restrict the set of nodes per-service.
- Lose benefit of large number of nodes (proxies/DNS servers/etc).