Load-Balancing in Content-Delivery Networks

Michel Goemans

Covering MSTs [G.-Vondrak '03]

- Complete graph K_n with distinct edge weights
  - $Q = \bigcup_{\{S: |S| \geq n-k\}} MST(G[S])$
    How large can |Q| be (as a function of n and k)?
  - S selected uniformly at random (Pr[v in S]=0.5)
    Find Q such that
    $$\Pr[Q \text{ contains } MST(G[S])] \geq 1 - 1/n^c$$
    How small can one choose |Q| ?

Problems with Centralized Content

Content-Delivery Networks

- Get content from server close to user
- **Fast:**
  - Eliminate long distances, peering issues
    http://www.lemonde.fr
- **Reliable**:
  - Less affected by failures in the internet
- **Scalable:**
  - Content can be massively accessed simultaneously

Akamai

- 13000 servers distributed across internet
- Delivery
  - Embedded objects — http://a177.ch1.akamai.net/...
  - Whole site — http://www.fbi.gov/
  - http/https — https://cardholder.paysystems.com
- All types:
  - html, pictures, software downloads, …
  - live and on-demand streaming
  - java servlets
- Reconstruction and processing at the edge

DNS based system

www.sonyericsson.com

↓ CNAME

a1538.g.akamaitech.net

↓ **MAPPING** to Akamai server

Time **T**o **L**ive: 20 sec

146.57.248.7 — On U. of M. campus

Mapping

- Mapping depends on
  - User location
    - IP space (dynamically) clustered in 10-50,000 blocks
  - Content type requested
    - web downloads, live streaming (WindowsMediaServer, QuickTime, Realaudio), secure content, java, …
  - Performance/congestion in internet
    - Latency, packet loss, …
  - Load on Akamai servers
- Mapping = Load Balancing

Multi-Objective

- Major goals:
  - User experience, quality mapping
  - Not overload servers
- Other goals:
  - Robustness against load spikes, internet failures, …
  - Bandwidth utilization

Mapping or Load-Balancing

- Two levels:
  - Toplevel
    - Mapping to a cluster of servers
    - Updated every < 1 min
  - Lowlevel:
    - Mapping within cluster
    - Constantly being updated
    - If a server goes down, other can seamlessly take over

LoadBal Complexity: Reaction Times

- Nameserver resolutions are cached by local nameservers (NS)
  - Impact of mapping changes not immediate
  - Actual load is smoothed
  - Harder to detect load spikes → need to anticipate
  - Stability issues

Load graph

- “Minor” issue since TTLs are small (20 secs)

LoadBal Complexity: Load Stickiness

- Once download/event starts, no way to shift load
- crucial issue for streaming events (connection times of over an hour)

→ “trial-and-error” approach unacceptable

Loadbal Complexity: Heterogeneity of Traffic

- Very different content types
  - http, https, live streaming (WMS, Real, QT, …), huge downloads, content with huge cache footprint,…
  - Not every machine can serve every request
- Customer constraints

→Same IP can be mapped at same time to many different machines for different contents

→Need to perform millions of assignments every 30 secs

Yesterday from here

- www.lemonde.fr, www.msnbc.com, www.bestbuy.com, www.logitech.com, www.monster.com
  146.57.248.* (University of Minnesota)
- a123.r.akareal.net
  63.240.15.177 (ATT – New York)
- https://cardholder.paysystems.com
  63.211.40.85 (L3 – New York)
- a177.ch1.akamai.net (usa.bmwfilms.com)
  64.241.238.153 (Savvis – Chicago)

LB Complexity: Multi-Dimensional Load

- Not a single constraining resource!
- Can be:
  - Bandwidth
  - CPU usage (e.g. key signing for https)
  - Disk usage (e.g. for cache misses, auction sites)
  - Memory (e.g. EdgeJava)
  - Threads (e.g. EdgeJava)
  - Number of licenses in realaudio
- Not necessarily linear
  - Live streaming: 0-1: whether cluster subscribes to stream

LoadBal Complexity: No load conservation

Multi-dimensional + non-linear

=

No load conservation

LoadBal Complexity: Contracts

- Network contracts
  E.g. Akamai machines on U. of M. campus can be used to serve only users from U. of M.

- Customer contracts
  E.g. maximum to serve, customer servers, …

LoadBal Complexity: Extreme Cases

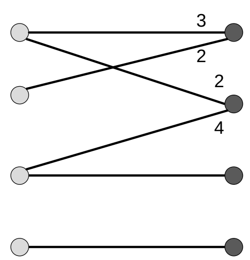
- 99% of time: Load-balancing easy
- 1%: extreme conditions
  - NEED to work under most incredible scenarios
    - Internet failures (or DOS attacks)
    - Only with part of input (since highly distributed environment)
    - Scheduled/unscheduled events
  - Load estimates unreliable
  - CRITICAL to run fast (avoid domino effect)
  - Reasonable mappings

LoadBal Complexity: Scalability

NEED
(sub)linear time algorithm
that can be
parallelized and distributed

Stable Marriages

- Assignment of men and women
  - Each man ranks each woman and vice versa
  - Marriage stable if no pair (m,w) unmatched where m prefers w to his "wife" and w prefers m to her "husband"



Beauty of Stable Marriages

- [Gale and Shapley '62]:
  - Stable marriage always exists!
  - Algorithm: ("men-propose, women-dispose")
    - Each unmatched man proposes to women in order of preference
    - A woman (tentatively) accepts if proposal came from a man she prefers to her tentative fiance
  - Stable marriage independent of order of proposals!:
    - "man-optimal" marriage
    - (lattice structure)
  - Running time linear in number of proposals ($\leq$ size of pref lists) [very fast]
  - Works also if incomplete preference lists

Residents-Hospitals Extension

- Residents-Hospitals
  - results + algorithm extends to case in which hospital j can accept c(j) residents
  - In use since 1951 by National Intern Matching Program

Stable Allocation Problem

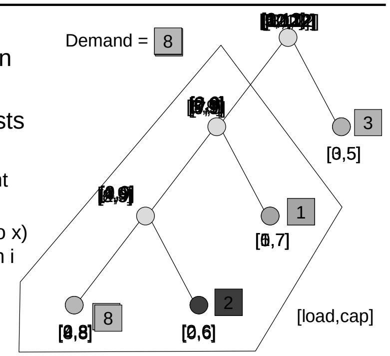
- [Baïou-Balinski ’98, G. ’00]
  - “demand item” i has demand s(i) and ranks every j
  - “supply item” j has capacity c(j) and ranks every i
  - Assignment (i,j) has capacity u(i,j)
  - (fractional) assignment π is stable if
    π(i,j)<min(s(i),u(i,j),c(j)) implies π(i,j’)= min(s(i),u(i,j’),c(j’))
    for every j’ preferred to j by i, and similarly for j
  - Gale-Shapley algorithm applies
    - Not polynomial, but weakly polynomial for integral inputs
    - [BB ’98] Strongly polynomial if used inductively

Stable Allocations With Tree Constraints

- [G ’00]:
  - resources 1,…,k
  - Supply item j has rooted tree T(j) of constraints
    - V(T(j))={1,…,k}
    - Every node v of T has capacity c(j,v)
  - Demand item i has basic resource b(i) and demand d(i)
    - When x units mapped to supply j, uses x units of each resource on path in T(j) from b(i) to root of T(j)
  - Stability as before

Algorithm for Tree Constraints

- Demand items request unassigned demands in order of preference
- When demand i requests x units from j, repeat:
  - Find lowest (in tree) tight constraint, say node v
  - Dispose demands (up to x) of lower preference than i and using resources in subtree rooted at v



Properties

- Algorithm
  - gives stable allocation
  - (man-optimal) maximizes amount of i^{th} demand allocated and allocates it to best possible supply node among all stable allocations
  - If tries to assign only ≥ 1-ε of each demand then linear in size of preference lists (used)
  - Easily parallelized and distributed (variety of ways)
  - If m demands, n supplies and k resources, then at most nk fractional demands assigned

Stable Allocations with Tree Constraints for Load Balancing

- Demand items: (groups of IPs, rule for mapping)
  m=millions
- Supply items: cluster of servers
  n=thousands
- (Incomplete) preference lists for demands based on performance + contract rules
- (Implicit) preference lists for supplies based on alternate choices, contract rules, …
- Tree of constraints model various resource constraints
- Almost integral assignment (at most a few thousands fractional)
- Just the core algorithm: many additional peripheral components
- Extremely fast!

Covering MSTs [G.-Vondrak '03]

- Complete graph K_n with distinct edge weights
  - $Q = \bigcup_{\{S: |S| \geq n-k\}} MST(G[S])$
    How large can |Q| be (as a function of n and k)?
    $$\leq e\,(k+1)\,n$$
  - S selected uniformly at random (Pr[v in S]=0.5)
    Find Q such that
    $$Pr\,[Q \text{ contains } MST(G[S])] \geq 1 - 1/n^c$$
    How small can one choose |Q| ?
    $$\leq e\,(c+1)\,n \log_2 n$$