Approximation Algorithms for Data Distribution with Load Balancing of Web Servers

Li-Chuan Chen

lichen@mitre.org

The MITRE Corporation

Co-author: Hyeong-Ah Choi

(George Washington University)

Outline of the Talk

- Research Motivation and Goals
- Prior Literature
- Problem Formulation
- Research Results
- Conclusions and Future Directions

Research Motivation

With the increased popularity of World-Wide-Web (WWW or Web) there are a number of problems:

– Servers overloaded.

– Internet backbone congestion.

– Slow Web services access.

Background

Approaches to Reduce Server Load:

- Mirror Web Sites: Replicate web server contents throughout network. (User must select server.)
- Web Caching: Stores frequently requested Web documents closer to the users (Cache coherence).
- Distributed Web Server: Web documents are distributed among a cluster of servers acting as a single server (Load balancing).

Research Goals

To reduce Web server load and to increase efficiency and reliability of Web system performance via:

Load Balancing: Balancing the load among a set of distributed Web document servers.

Research Goals: (Continued)

We will consider the design of optimization algorithms for achieving these objectives.

- Note that most formulations of these problems are NP-hard.
- We consider special cases and approximation algorithms for Load Balancing.

Load Balancing for Web Servers

Assume:

- A cluster of back-end Web servers working together as a single Web server.
- A set of Web documents are to be allocated among these servers.
- A mechanism to redirect HyperText Transfer Protocol (HTTP) requests to one of the back-end servers.

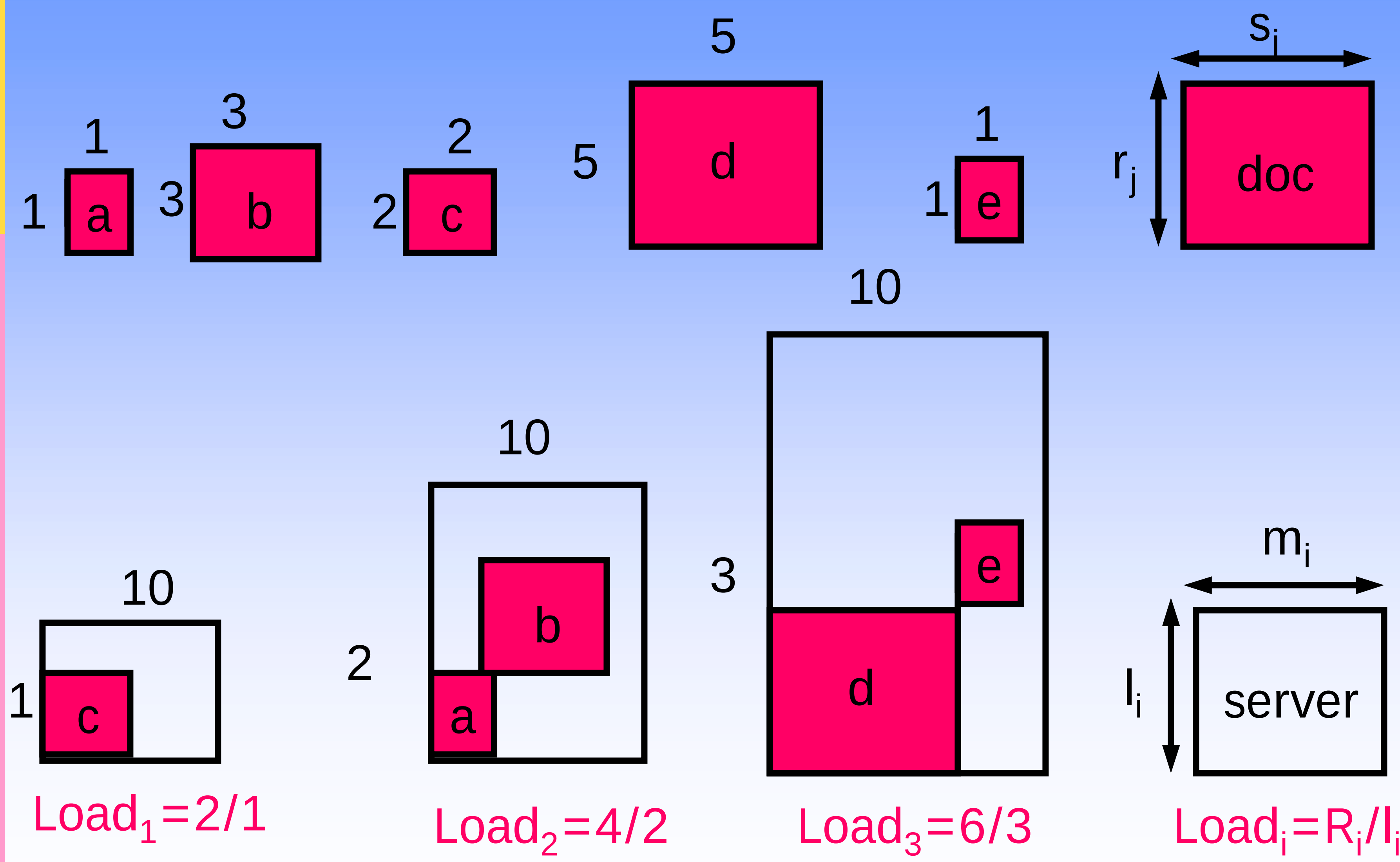
Prior Literature

- **Client-based load balancing**: A list of replicated server's performance is maintained at the client's proxy server and then a URL is mapped onto one of the servers. [MDZ99, SBSV98, LM97]
- **Server-based load balancing**: Front-end server dispatches incoming Web requests to one of the back-end servers via round-robin Domain Name Service (DNS) or server load. [SNCC96, AYHI96 GGMP95]
- **Hybrid approaches**: Combination of DNS round-robin, HTTP redirection, and document's access rate to balance the load. [NRY97]

Problem Formulation

- Consider M servers and N documents. Server i is associated with memory size m_i and number of simultaneous HTTP connections l_i. Document j is associated with document size s_j and access rate r_j.
- Given an allocation of documents to servers, let R_i denote the total access rate for server i.
- Define load of server i to be R_i / l_i . The objective is to minimize the maximum load over all servers.
- Input: Quadruple I = (r, l, s, m).
- Output: An allocation of documents to servers.

Example:



1
1
a
3
3
b
2
2
c
5
5
d
1
1
e
s_i
r_j
doc
10
1
c
10
2
b
a
10
3
e
d
m_i
l_i
server
$Load_1=2/1$
$Load_2=4/2$
$Load_3=6/3$
$Load_i=R_i/l_i$

Research Results

- **Lower bound** on the optimal load is r/l.
- **Optimal allocation is NP-hard**: Reduction from bin-packing. This means optimal allocation problem probably cannot be solved in polynomial time.

Research Results (Continued)

We present approximation algorithms for various special cases.

- No memory constraints: O(nm)-time factor-2 approximation of the optimal solution.
- Memory and load constraints: O(nlogn)-time factor-4 approximation of the optimal solution.
- Small documents: If a server can hold at least k documents, a 2(1 + 1/k)-factor approximation.

Lower Bounds

Input: Quadruple I = (r, l, s, m) with no memory constraint.

- Let r be the sum of access rates over all documents, and let l be the sum of connections over all servers.
- Let f^* be the optimal allocation cost. Then we have the following lower bound on f^*:

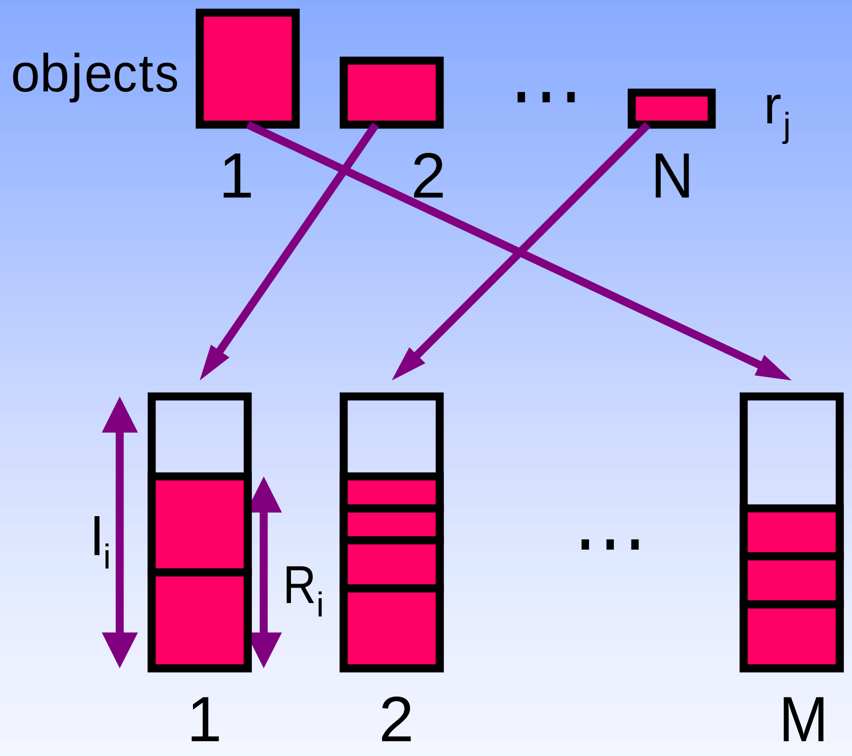
$$r_{\max} = \max_{1 \le j \le N} r_j,\ \ l_{\max} = \max_{1 \le i \le M} l_i$$

$$f^* \ge \max\left\{ \frac{r_{\max}}{l_{\max}}, \frac{r}{l} \right\}$$

- Proof is based on averaging and the pigeon-hole principal.

NP-Completeness

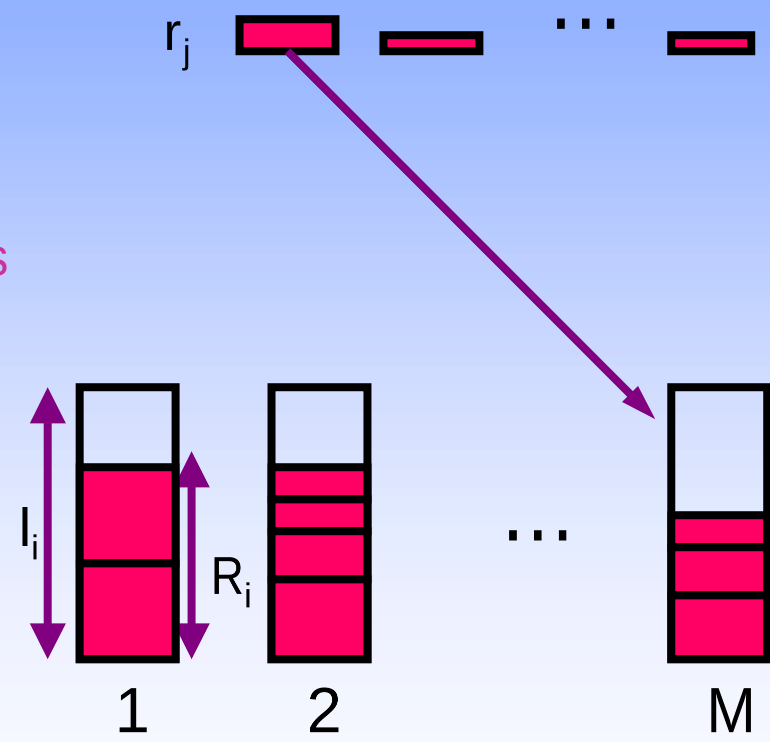
- 0-1 Allocation is NP-hard with no memory constraints by reduction from bin-packing, where r denotes size of the objects and bins are of size $l = l_1 = l_2 = \ldots = l_M$.
- Similarly for 0-1 Allocation with memory constraints, where objects are size of documents and bin sizes are size of memories.



Approximation Algorithm: No Memory

- Sort documents by decreasing order of access rate, r_j.
- For each document j
  - Place it in server i that minimizes $(R_i + r_j)/l_i$.
  - R_i += r_j.

Intuition: Put each document into the server with the greatest remaining load capacity.



Approximation Algorithm: No Memory (Continued)

- Approximation for no memory constraints is a O(nm)-time, factor-2 approximation of optimal load.
- Why Factor-2? Each server will be utilized to at least half its capacity. Optimal cannot utilize to more than full capacity.

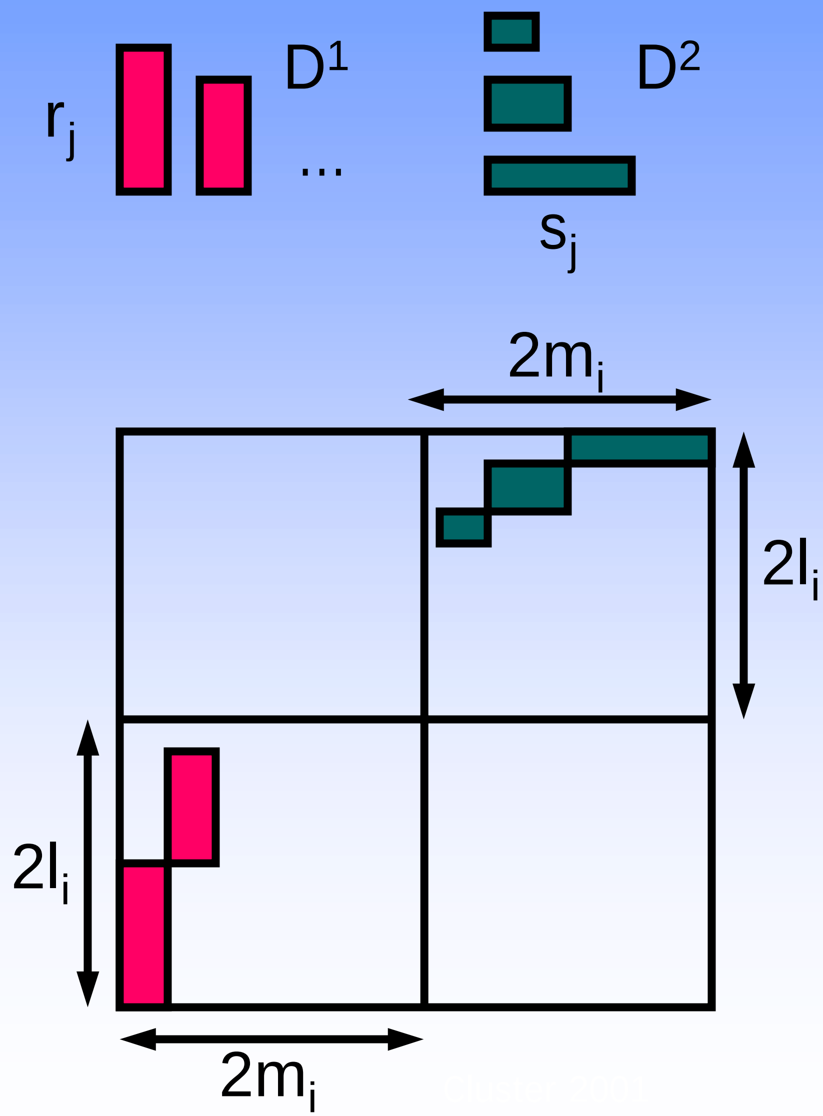
Approximation Algorithm: Memory and Load Constraints

- Split documents into 2 sets, D^1 , D^2, where D^1 consists documents whose access cost is bigger than document size and D^2 consists documents whose document size is bigger than access cost.
- Assign as many documents which are in D^1 as possible and then assign the remaining documents which are in D^2.
- If all documents have been assigned to some server then feasible allocation exist else no solution.

Approximation Algorithm: Memory and Load Constraints

Intuition:

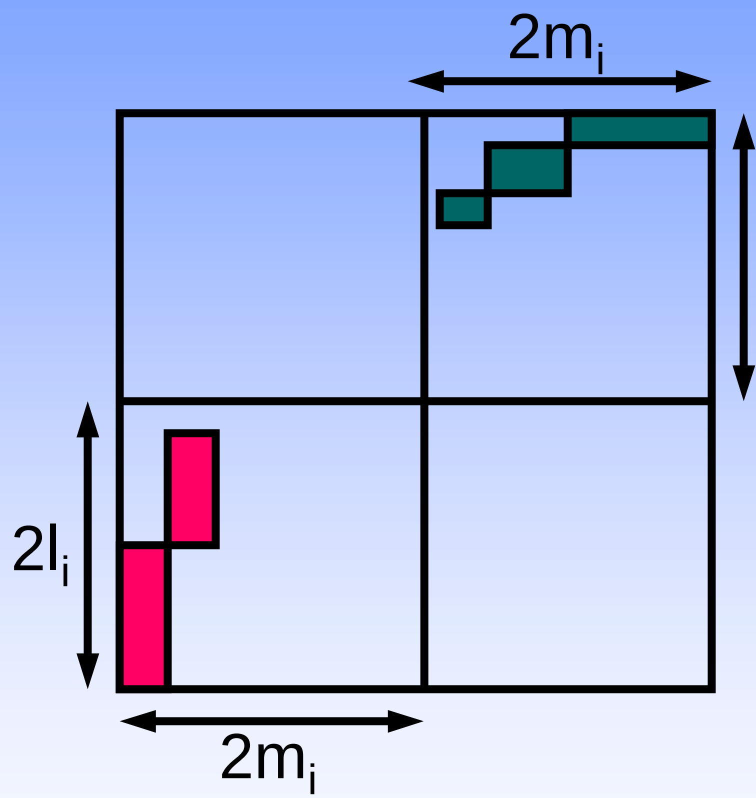
- Access dominant and size dominant documents are allocated separately.
- Thus each server is not too poorly utilized according to either criteria.



Approximation Algorithm: Memory and Load Constraints

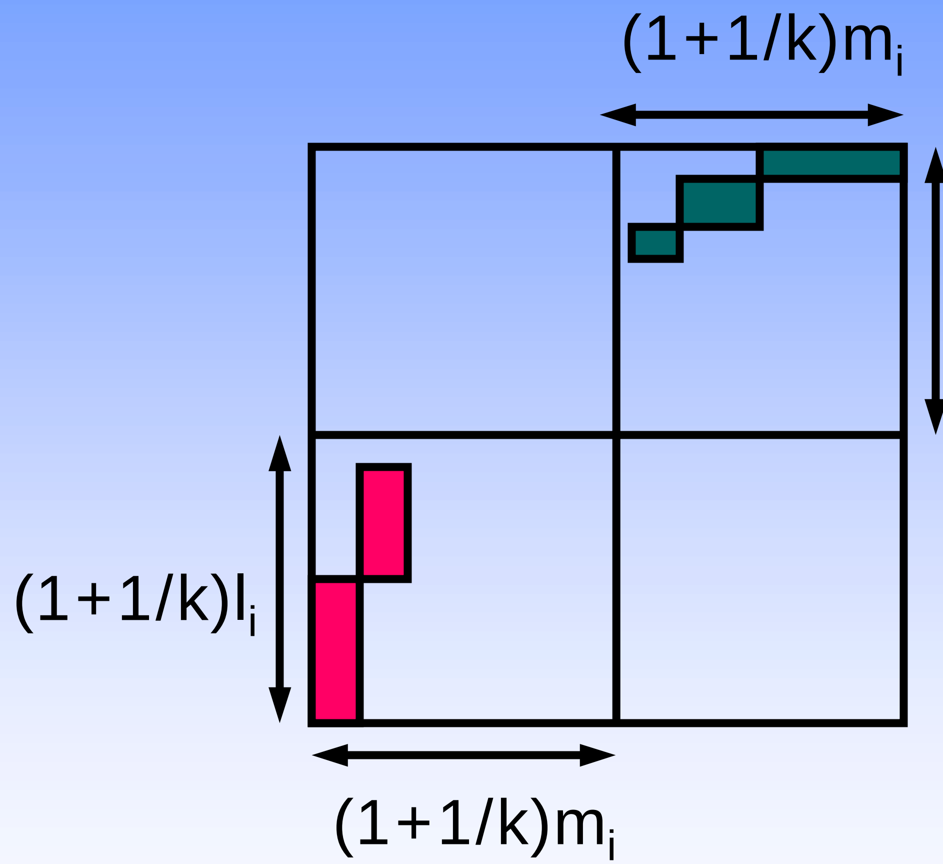
- For equal memory and load constraints, this is an O(n log n)-time, factor-4 approximation of optimal solution.

- Why? Intuitively each server is at least ¼ utilized in load or memory.



Approximation Algorithm: Small Documents

- Approximation for small documents is 2(1 + 1/k) time the optimal solution if each server can hold at least k documents.



Future Directions

Consider ways to strengthen our existing results either by improving efficiency of the algorithms or by eliminating some of the assumptions that are made.

- Dynamic Load Balancing: How to deal with server failures, access rate changes, and changes in server capacity.
- On-line algorithm for load balancing.