Challenges in Web Information Retrieval

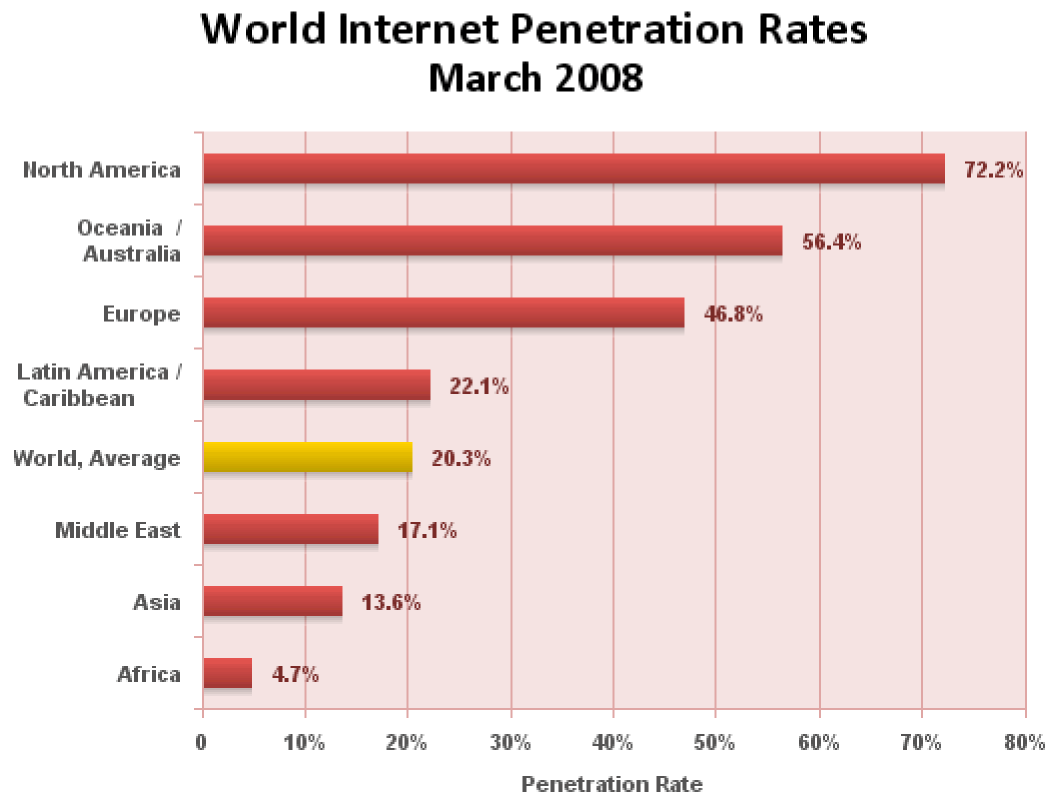
Monika Henzinger
Ecole Polytechnique Federale de Lausanne (EPFL) & Google Switzerland

Statistics for March 2008

20% of the world population uses the internet [internetworldstats.com]

~300 million searches per day [Nielsen NetRatings]

⇨ search engines are the second largest application on the web



Outline of this talk

Search engine architecture

- Open problem: Loadbalancing

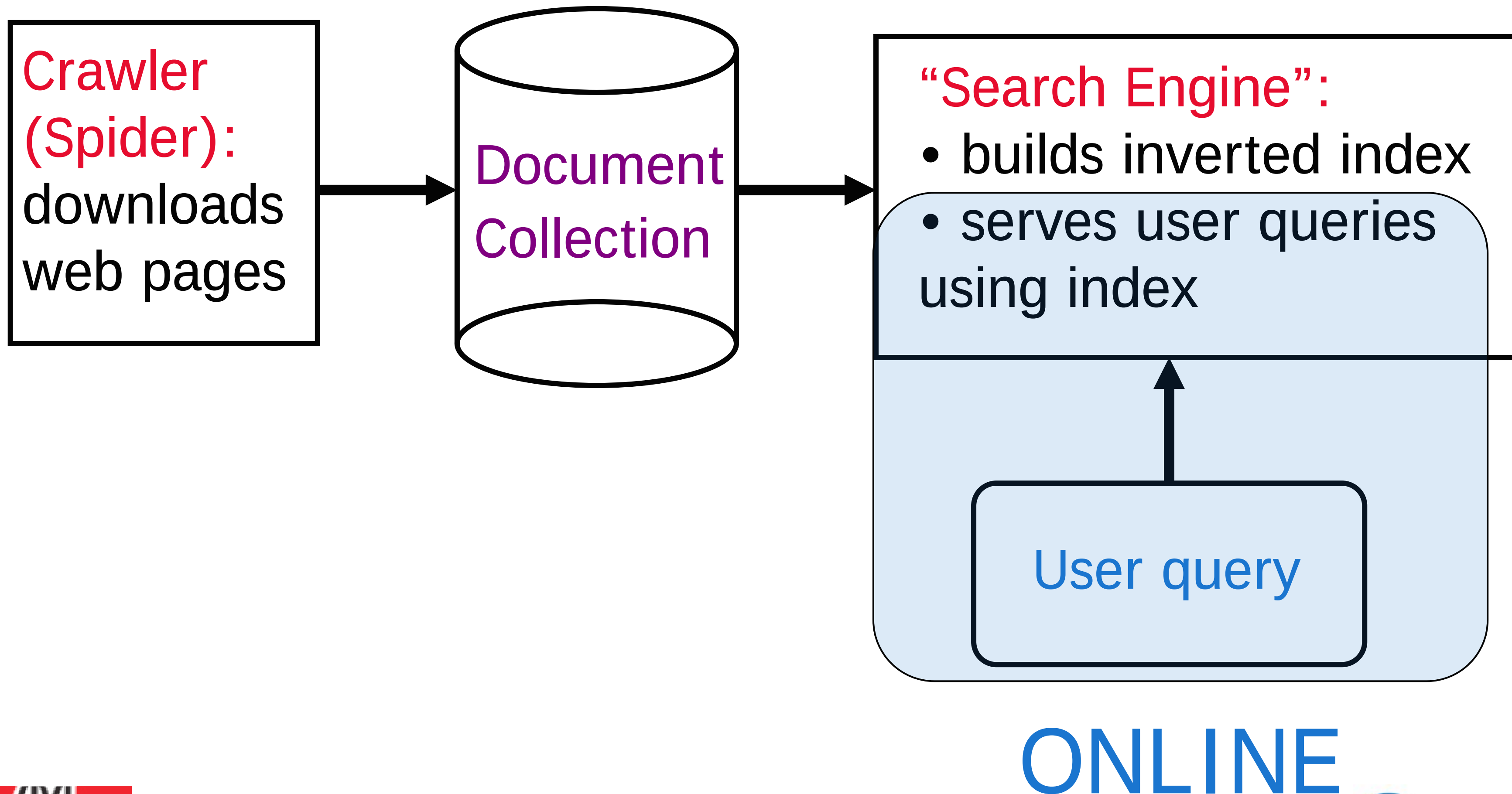
Large-scale distributed programming model

- Open problem: Relationship to data stream model

Sponsored search auctions

- Open problem: Realistic user modeling

Search Engine Architecture



Inverted Index

All web pages are numbered consecutively

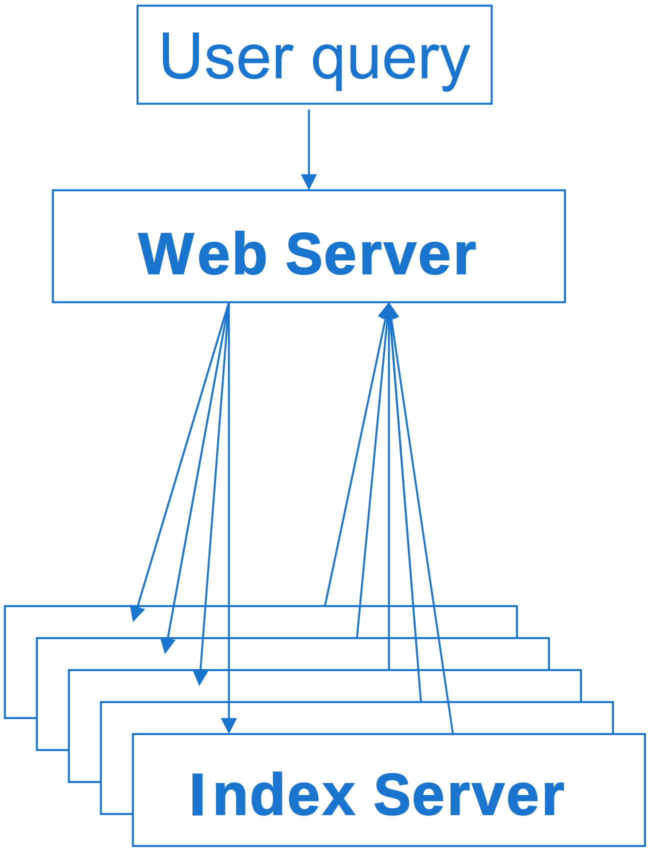
For each word keep an ordered list (posting list) of all positions in all document

Princeton → (3,1) (3,10) (6,2) (9,4) (9,8) (10,1) (20,2)...

Tarjan → (3,2) (3,20) (7,4) (8,3) (9,2) (9,20) (104,2) ...

⇨ query running time linear in length of posting lists of query terms

Query Data Flow



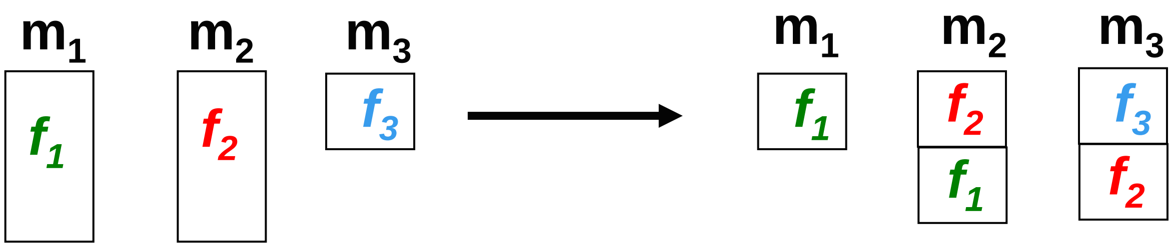
Split document set into subsets

Place complete index for one or more subsets on each index server

Problems:

- Some servers might have more indices than others
- Some indices have lower throughput than others causing their servers to become bottlenecks

Idea: Copy Indices



Questions:

Which indices to copy?

How to assign indices and copies to machines?

Where to send individual requests?

⇨ Offline file layout & online loadbalancing problem

Model

Offline layout phase:

- Set $m_1 \dots m_m$ of identical machines, each has s_i slots s.t. each indices fits into each slot
- Set $f_1 \dots f_n$ of indices
- Assign files and copies to machines

Online loadbalancing phase: A sequence of requests arrives s.t.

- every request t needs to access one index f_j and
- places a load of $l(t)$ on the machine that it is assigned to

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Google

Model (cont.)

Machine load ML_i = sum of loads placed on m_i

Goal: Minimize $\max_i ML_i$ (makespan)

- *A(s)* = maximum machine load on sequence s
- *OPT(s)* = maximum machine load on sequence s for optimum offline algorithm that might use a different file layout

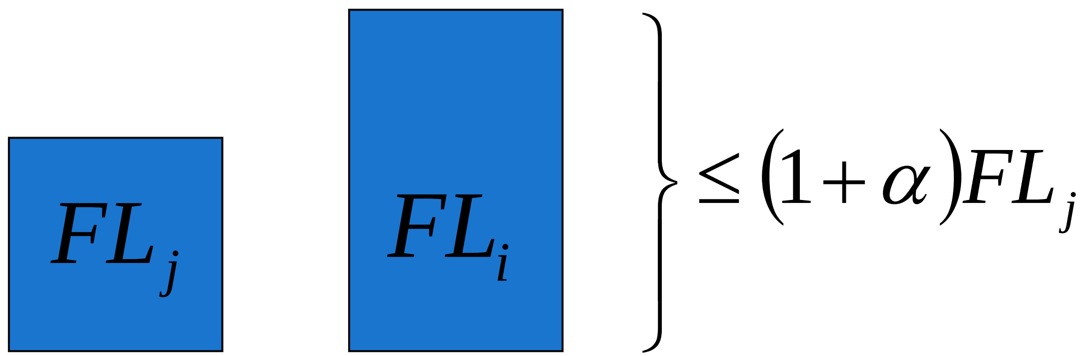
Competitive Analysis: An algorithm A is k-competitive if for any sequence s of requests

$$A(s) \leq kOPT(s) + O(1)$$

Goal: Study tradeoff between competitive ratio and number of used slots

Parameters

Set α s.t. $\forall i, j : FL_i \leq (1+\alpha) FL_j$
where FL_j = sum of loads of requests for index f_j



Set β = $\max_t$ individual request load $l(t)$

Note: In web search engines: α is < 1, β is constant

Results

Assumption: Every machine has same number of slots

Slots	n	nm	$\frac{nm}{g(m)} \geq n$	$\frac{3n}{2}$
Competitive ratio deterministic	$1+\left(1-\frac{1+\alpha}{m+\alpha}\right)\alpha$	1	$1+\left(1-\frac{1+\alpha}{g(m)+\alpha}\right)\alpha$ *	
Competitive ratio randomized				$1+\frac{\alpha}{2}$ *

*: some additional conditions apply

Open questions

Lower bounds

Different models:

- Performance measures
- Machine properties:
  - Speeds (related/unrelated machines)
  - Slots per machine
- Arrival times and duration

Outline of this talk

Search engine architecture

- Open problem: Loadbalancing √

Large-scale distributed programming model: MapReduce

- Open problem: Relationship to data stream model

Sponsored search auctions

- Open problem: Realistic user modeling

What is MapReduce?

System for distributing batch operations over many data items over cluster of machines

Map phase:

- Extracts relevant information from each data item of the input
- Outputs (key, value) pairs

Aggregation phase:

- Sorts pairs by key

Reduce phase:

- Produces final output from sorted pairs list

User writes two simple functions: map and reduce. Underlying library takes care of *all* details

⇨ frequently used within Google (70k jobs in 1 month)

Model (Feldman et al. '08)

Massive unordered distributed (mud) model of computation:

A mud algorithm is a triple $(\Phi, +, \Gamma)$, where

- $\Phi: \Sigma \to Q$ maps an input item to message
- the aggregator $+: Q \to Q$ maps two message to a single message
- post-processing operator $\Gamma: Q \to \Sigma$ produces the final output

For input $\mathbf{x} = x_1, \ldots x_n$ it outputs

$$m(\mathbf{x}) = \Gamma(\Phi(x_1) + \Phi(x_2) + \ldots + \Phi(x_n))$$

A mud algorithm computes a function f if for all **x** and all possible topologies of + operations:

$$f(\mathbf{x}) = m(\mathbf{x})$$

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Relationship to streaming algorithms

Observation: Any mud algorithm can be computed by a streaming algorithm with the same time, space, and communication complexity.

Inverse:

- f must be order invariant on input, since mud works on unordered data

Theorem: For any order-invariant function f computed by a streaming algorithm with

- g(n)-space and c(n)-communication s. t. $g(n)=\Omega(\log n)$ and $c(n)=\Omega(\log n)$

there exists a mud algorithm with

- $O(g^2(n))$-space, $O(c(n))$-communication, and $\Omega(2^{\mathrm{polylog}(n)})$ time

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Open problems

More efficient mud algorithm

Multiple mud algorithms, running simultaneously over same input, each aggregating only values with same key

⇨ closer to MapReduce

Multiple iterations

- **Example**: Finding near-duplicate web pages using k fingerprints per page:
  - 1 MapReduce with space $O(k^2n)$
  - 2 MapReduces with space $O(kn)$

Outline of this talk

Search engine architecture

- Open problem: Loadbalancing √

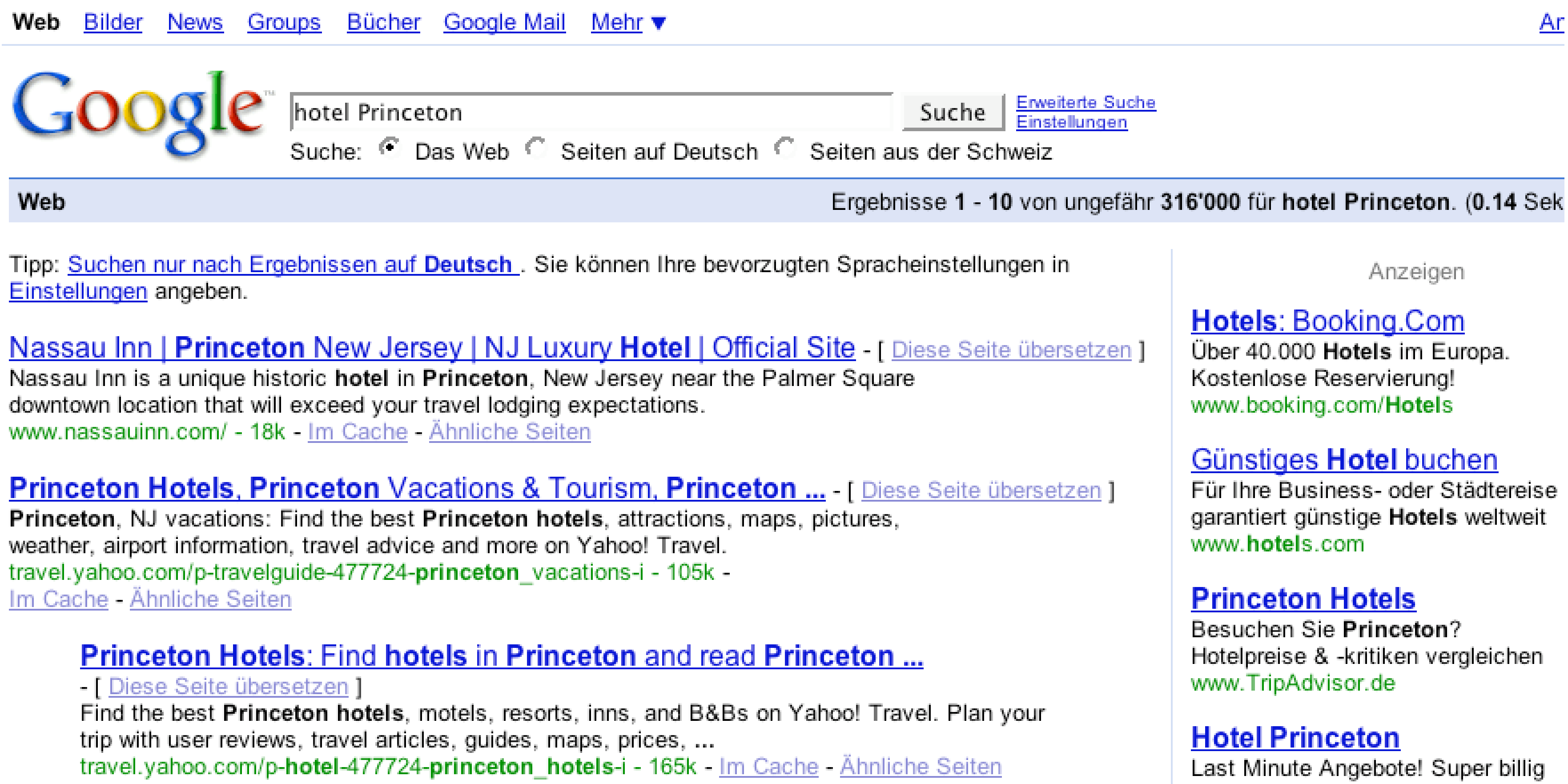
Large-scale distributed programming model: MapReduce

- Open problem: Relationship to data stream model √

Sponsored search auctions

- Open problem: Realistic user modeling

Search: hotel princeton



Web Bilder News Groups Bücher Google Mail Mehr ▼ Ar

Google

hotel Princeton Suche Erweiterte Suche Einstellungen

Suche: Das Web Seiten auf Deutsch Seiten aus der Schweiz

Web Ergebnisse **1** - **10** von ungefähr **316'000** für **hotel Princeton**. (**0.14** Sek

Tipp: Suchen nur nach Ergebnissen auf **Deutsch** . Sie können Ihre bevorzugten Spracheinstellungen in Einstellungen angeben.

Nassau Inn | **Princeton** New Jersey | NJ Luxury **Hotel** | Official Site - [Diese Seite übersetzen]
Nassau Inn is a unique historic **hotel** in **Princeton**, New Jersey near the Palmer Square downtown location that will exceed your travel lodging expectations.
www.nassauinn.com/ - 18k - Im Cache - Ähnliche Seiten

Princeton Hotels, **Princeton** Vacations & Tourism, **Princeton ...** - [Diese Seite übersetzen]
Princeton, NJ vacations: Find the best **Princeton hotels**, attractions, maps, pictures, weather, airport information, travel advice and more on Yahoo! Travel.
travel.yahoo.com/p-travelguide-477724-**princeton**_vacations-i - 105k - Im Cache - Ähnliche Seiten

Princeton Hotels: Find **hotels** in **Princeton** and read **Princeton ...** - [Diese Seite übersetzen]
Find the best **Princeton hotels**, motels, resorts, inns, and B&Bs on Yahoo! Travel. Plan your trip with user reviews, travel articles, guides, maps, prices, **...**
travel.yahoo.com/p-**hotel**-477724-**princeton**_**hotels**-i - 165k - Im Cache - Ähnliche Seiten

Anzeigen

Hotels: Booking.Com
Über 40.000 **Hotels** im Europa.
Kostenlose Reservierung!
www.booking.com/**Hotels**

Günstiges **Hotel** buchen
Für Ihre Business- oder Städtereise garantiert günstige **Hotels** weltweit
www.**hotels**.com

Princeton Hotels
Besuchen Sie **Princeton**?
Hotelpreise & -kritiken vergleichen
www.TripAdvisor.de

Hotel Princeton
Last Minute Angebote! Super billig

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Google

Sponsored Search Auctions

Advertisers enter bids for keywords.

At query time:

Ranking Scheme: System ranks ads by

- Bid
- Effective bid = bid * click-through-rate

2. Payment Scheme: Charge advertisers only if users click on an ad.
   - Generalized First Price (GFP): Pay what you bid: Advertisers see-saw.
   - Generalized Second Price (GSP): Pay what the ad below you bid: stable

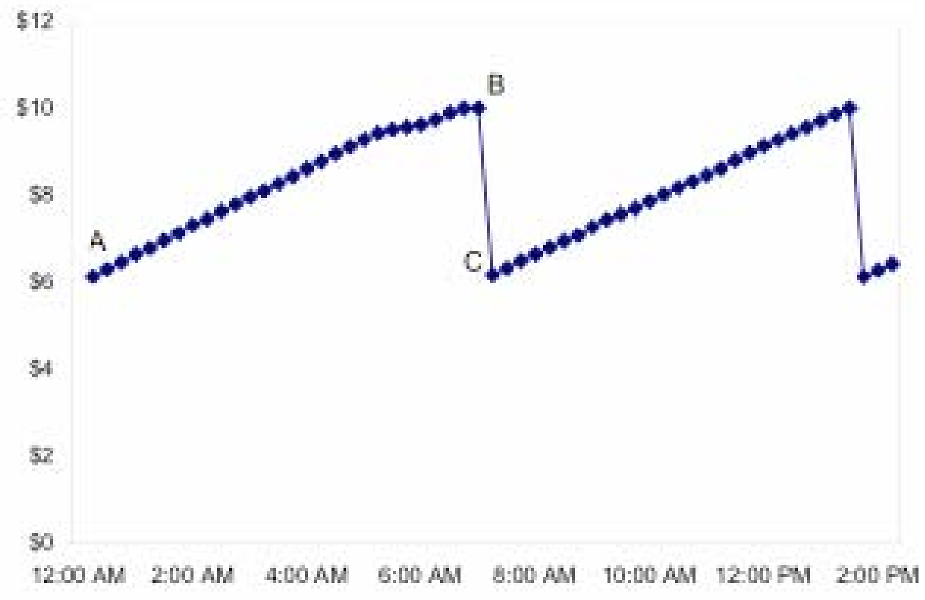
Adv	Bid	Price
Alice	$0.32	$0.24
Bob	$0.24	$0.17
Carol	$0.17	$0.14
David	$0.14	---

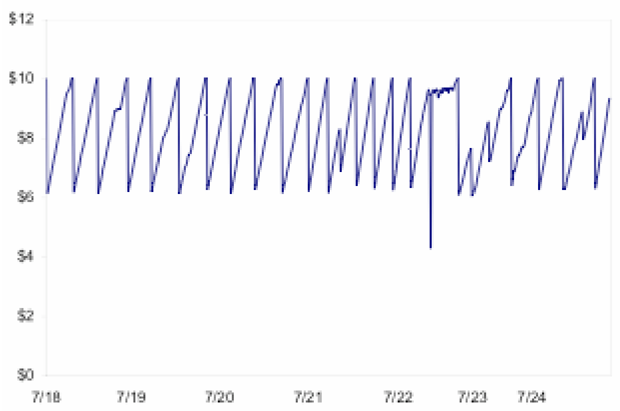
Goal: Design ranking and payment scheme that makes everybody "happy"

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Google

Pay what you bid: Non-stability





Source: Edelman, Ostrovsky, Schwarz: Internet Advertising and the Generalized Second Price Auction: Selling Billions of Dollars Worth of Keywords

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Google

Sponsored Search Auctions

Advertisers enter bids for keywords.

At query time:

Ranking Scheme: System ranks ads by

- Bid
- Effective bid = bid * click-through-rate

2. Payment Scheme: Charge advertisers only if users click on an ad.
   - Generalized First Price (GFP): Pay what you bid: Advertisers see-saw.
   - Generalized Second Price (GSP): Pay what the ad below you bid: stable

Adv	Bid	Price
Alice	$0.32	$0.24
Bob	$0.24	$0.17
Carol	$0.17	$0.14
David	$0.14	---

Goal: Design ranking and payment scheme that makes everybody "happy"

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Most desirable properties

Stability: Bidders reach an equilibrium where it's not in their interest to change bids

Simplicity: Bidders can understand how the price is derived from the bids

Monotonicity: Increasing bid does not decrease position and does not decrease click probability

Current Model

Assumptions:

- *ca(i)* = click-through rate for ad *i*
- *cp(j)* = click-through multiplier for position j, *cp(j)* < *cp(j-1)*
- Separability: Pr[click on ad *i* at pos *j*] = *ca(i) cp(j)*
- Each bidder *i* has internal value *v(i)*
  - Expected value at position *j*: *ca(i) cp(j) v(i)*
  - Expected utility at position *j: ca(i) cp(j) (v(i) – price(j))*
- If p_i is the position for bidder *i* then total expected value =

$$\sum_i ca(i) cp(p_i) v(i)$$

Goal: Maximize total expected value (efficient allocation)

Observation: Ranking by decreasing *ca(i) v(i)* maximizes total expected value

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Google

Current Model (cont.)

Observation: Ranking by decreasing *ca(i) v(i)* maximizes total expected value

Recall: System ranks by effective bid = *ca(i) b(i)*

System knows only *b(i)* not *v(i)*

Payment scheme:

- Vickrey-Clarke-Groves (VCG):
  - It's best for bidder *i* to bid *v(i)*
    ⇨ stable
    ⇨ ranking maximizes total expected value
  - Price depends on "damage caused to the other players" ⇨ not very simple
- GSP:
  - simple, monoton, stable,
  - but bidding *v(i)* is not usually best ⇨ ranking does not usually maximize total expected value

Separable user models

Above separable user model:

Pr[click on ad *i* at pos *j*] = *ca(i) cp(j)*

- "Pick position according to distribution *cp(j).* Click on the ad in that position with probability *ca(i)* ."

More realistic separable user model:

- "Scan from top down. When you reach an ad, click with probabilty *ca(i)*. Continue scanning with probability *q(i,j).*"

Different User Model: Markovian (Feldman et al.'08)

Markovian user model:

- Scans ads from top down.
- When reaches ad *i* in position *j*, clicks with probability *ca(i).*
- Continues scanning with probability *q(i,j).*

For *q(i,j)* does not depend on *j*, Feldman et al.

- give simple algorithm for finding best ranking of ads
  - monoton
- VCG payments resulting auction is stable and maximizes total expected value

Open problems

Markovian User Model

- Non-VCG pricing: Is there a simple, stable payment scheme in the Markovian User Model?
- User impatience: Analyze the case that q(i,j) depends on both i and j

Model budgets for bidder (Feldman et al, Borgs et al, Dobzinski et al.)

Consider a variety of advertiser preferences = utility functions

- I don't care how much I pay, but I always want slot 3.
- I'm willing to pay up to $5 per click, or up to $1 per impression.
- My margin is $1 per click. Give me position that maximizes my profit, i.e. value of clicks minus price paid.
- Maximize my profit, but never spend more than $0.50 per click.

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Summary

Search engine architecture

- Open problem: Loadbalancing √

Large-scale distributed programming model: MapReduce

- Open problem: Relationship to data stream model √

Sponsored search auctions

- Open problem: Realistic user modeling √

Happy Birthday, Bob!