Organização e Recuperação de Informação: Projeto de um buscador

Marcelo K. Albertini

Faculdade de Computação, Universidade Federal de Uberlândia

Projeto de um buscador

The anatomy of a Large-Scale Hypertextual Web Search Engine

- Protótipo por Sergey Brin e Lawrence Page, em Stanford
- `http://google.stanford.edu`
- 24 milhões de páginas iniciais
- Desafios:
  - número de documentos
  - número de termos
  - número de consultas
  - uso de informação de hipertextos para melhorar qualidade

Cenário da época: 1997

- Pessoas iniciavam busca em índices mantidos por humanos
  - somente tópicos mais populares
  - caros de construir e manter
  - lentos para melhorar
  - subjetivos
- Depois recorriam a motores de busca automáticos
  - baixa qualidade
  - sensível a spam

Google: motivações

- Em 1994, WWWW (World Wide Web Worm) indexava 110 mil documentos
- Em 1997, WebCrawler indexava de 2 a 100 milhões de documentos
- Em 1994, WWWW recebia em média 1500 consultas por dia
- Em 1997, Altavista recebia na ordem de 20 milhões de consultas por dia
- Em 1997, Buscadores tinham baixa precisão: apenas 1 dos 4 principais conseguiam encontrar eles mesmos

Google em 1997: objetivo

- Estimavam em 2000, centenas de milhões de consultas por dia
- Em 2001, Google indexava mais de 1.5 bilhão de documentos
- Em 2009, mais de 1 bilhão de consultas e 24 petabytes de dados gerados por usuários por dia
- Em 2009, mais de 1 milhão de servidores
- Em setembro de 2013, Google indexa de 20 a 40 bilhões de documentos
- Se `the` aparece em 67.61% dos documentos e Google retorna que encontra `the` em $25,270$ milhões de documentos, então estimamos que o Google tem em seu índice cerca de 37.37 bilhões de documentos

Objetivo

"Abordar muitos dos problemas de qualidade e escalabilidade introduzidos pelo aumento da magnitude a tais extraordinários números"

Crescendo com a Web

- Crawling rápido
- Otimização do uso de espaço em disco para índices: centenas de gigabytes de índice
- Processamento de centenas a milhares de consultas por segundo
- Desempenho e custo de hardware melhorou
- Porém, tempo de seek em disco e confiabilidade de sistema operacional não mudou
- Esperança do custo para indexar e armazenar texto ou HTML
- Projeto original do Google: centralizado
- Nome Google: googol $= 10^{100}$ – ambição de ser um motor de busca extremamente grande

Metas de projeto

- Em 1994: “um buscador completo tornaria possível achar qualquer coisa muito facilmente”
- Em 1997: spam – apenas 1 de 4 buscadores comerciais conseguem encontrar resultados sobre si mesmo (no top-10) usando o seu próprio nome
- Número de documentos aumentou muito, mas habilidade de pessoas em ver documentos não mudou – “pessoas ainda querem olhar apenar para as algumas dezenas de resultados”
- Deve-se ter precisão muito alta, mesmo ao custo de taxa de recuperação
- Uso de estrutura de links e textos âncora da web para relevância e filtro de qualidade

Meta: buscador acadêmico

- Em 1993: 1.5% dos servidores web eram `.com`
- Em 1997: 60% eram `.com`
- Tecnologias de ORI eram consideradas segredos privados e de difícil domínio
- Objetivo declarado do Google: divulgar dados capturados a partir das buscas dos usuários a fim de apoiar atividades de pesquisa

Google: cacterísticas do sistema

Duas características principais

- Ranking de qualidade: PageRank
- Uso de texto âncora de links

PageRank

- Mapas de 518 milhões de links
- PageRank: medida objetiva da importância de citação que corresponde bem com a ideia subjetiva de importância
- Abordagem: Busca restrita para título de páginas e ordenada com o PageRank
- Inspiração em métodos de análise de literatura de citação acadêmica

Texto âncora

- Texto âncora de links são associados às páginas linkadas
  - Usado pelo World Wide Web Worm
- Texto âncora apresentam descrições acuradas da página linkada
- Texto âncora pode descrever documentos não indexáveis: videos, imagens, audio, applets
  - Possível retornar documentos não obtidos pelo crawler
- Melhoria de qualidade
- Em 1997: de 24 milhões de páginas obtidas, indexação de 259 milhões de âncoras

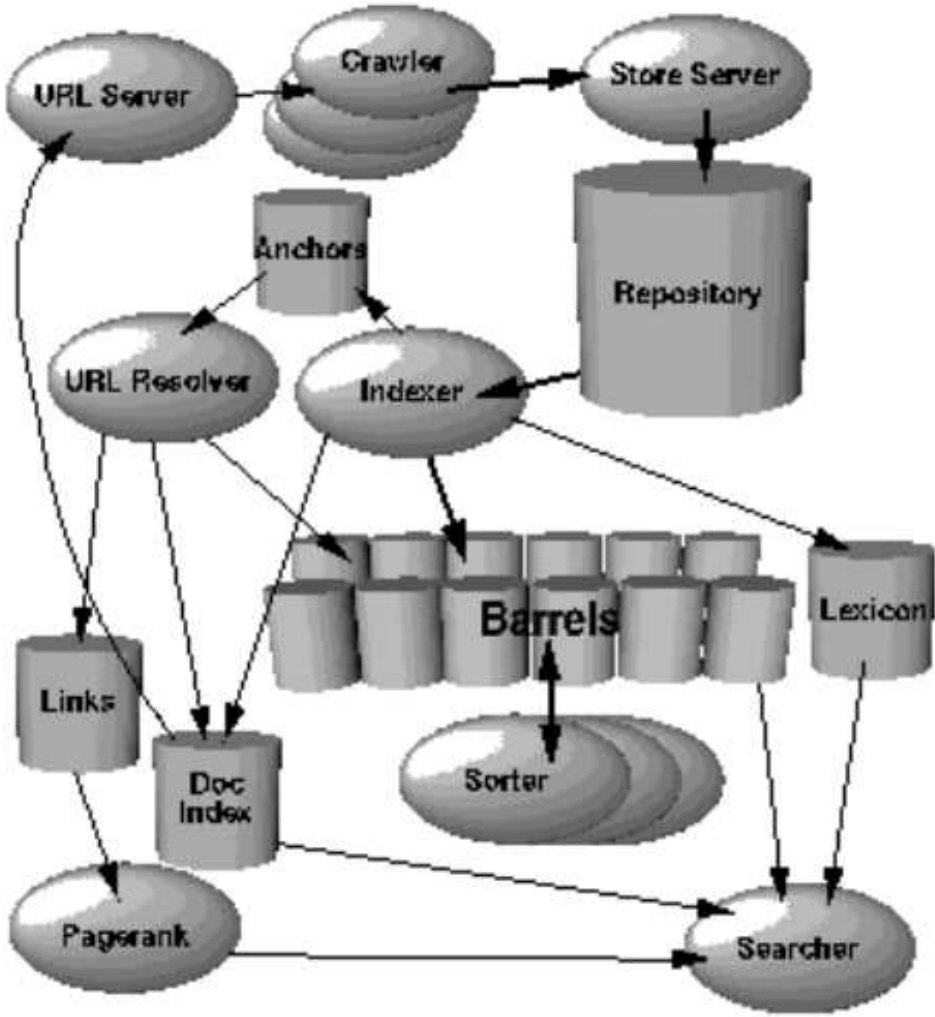
Outras características

- Google tem informação sobre posição de palavras nos docs para modificar busca
- Google usa informação sobre a apresentação visual: tamanho e cor de fontes
- Páginas completas não processadas são armazenadas no repositório
- Em 1997, 24 milhões de páginas ocupavam 147GB
- Uso do "Very Large Corpus" (TREC) de 20GB não foi suficiente
  - Modelo vetorial que funciona bem para o Very Large Corpus, não funciona para Web
  - Na web, modelo vetorial retorna documentos curtos mais algumas palavras (quer dizer, não usaram normalização de tamanho ou esse não foi efetivo)

Anatomia do sistema

- Estruturas de dados
- Crawling
- Indexação
- Busca

Implementação em C e C++.
Execução em sistemas Solaris e Linux.



Web crawling

- Crawing distribuído
- `URLserver` - envia URLs para crawlers
- Documentos capturados são enviados para o `storeserver`
- `storeserver` compacta e armazena docs em repositório
- Documentos tem `docID` atribuído ao obter um URL de uma página
- Indexação: `indexer` e `sorter`

Indexer

`Indexer`: organizador para indexação

- Lê doc a partir do repositório
- Descompacta documentos
- Processa o HTML (parser)
- Documento é convertido em conjunto de ocorrências de palavras: `hits`
- Índice posicional e de estilo (fonte)
- Distribuição de hits em `barrels` ("barris"): índice direto ordenado (ainda não invertido)
- Extrai links (origem, destino e texto âncora) e armazena informações em uma base sobre âncoras

URLresolver

`URLresolver`: organizador para cálculo do PageRank

- lê base de âncoras
- normaliza URL
- obtém novos `docIDs`
- coloca texto âncora em um indíce direto com `docID`
- gera base de links: pares de `docID` usados para o cálculo do PageRank

Sorter

`Sorter`: gerador do índice invertido

- Recebe os `barrels` (que estão ordenados por `docID`)
- Reorganiza docs e atribui `wordIDs` para gerar o índice invertido
  - feito no próprio espaço usado pelo índice direto
- Produz lista de `wordIDs` para o índice invertido
- Programa `DumpLexicon` usa lista de `wordIDs` para produzir para obter novo vocabulário para ser usado pelo buscador

O buscador

O buscador

- buscador é um servidor web
- usa, para responder às consultas,
  - o vocabulário produzido pelo `DumpLexicon`,
  - o índice invertido e
  - usa o PageRank

As estruturas de dados

- Seek em disco leva 10 ms
- Estruturas de dados projetadas para evitar seeks
  - `BigFiles`
  - `Repository`: guarda docs
  - Índice direto: documentos$\rightarrow$ palavras
  - Índice invertido: palavras $\rightarrow$ documentos
  - `Lexicon`: vocabulário
  - Listas de `Hits`: termos

BigFiles

- Arquivos virtuais
- Arquivo pode estar espalhado em múltiplos sistemas de arquivos
- Endereçáveis por inteiros de 64 bits
- Alocação entre diferentes sistemas de arquivos é automática
- Compactação

Repository

- Contém o HTML completo de cada página
- Compactação usando zlib
- Armazenamento contínuo de docs

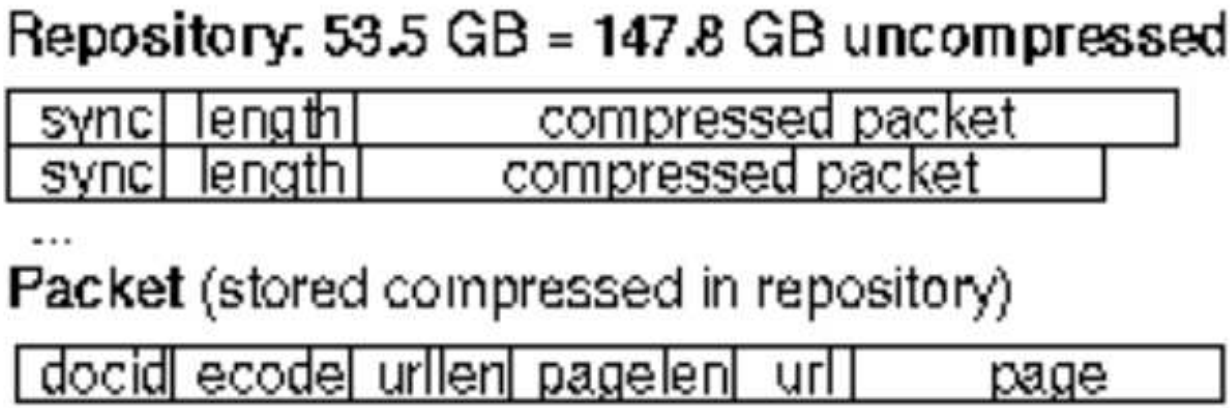


Figure 2. Repository Data Structure

Índice de documentos

Guarda informações sobre cada documento

- Índice ISAM (Index sequential acess mode)
  - tamanho fixo
  - Ordenado por `docID`
- Informações
  - status do documento
  - ponteiro para o repositório
  - checksum do documento
  - estatísticas
  - se doc foi capturado, tem ponteiro para um arquivo de tamanho variável `docinfo` com URL e título, senão aposta para URL
- mantém arquivo para converter URLs em `docIDs` em bloco, para eficiência

Lexicon

Vocabulário

- Cabe em memória principal: 256MB
- 14 milhões de palavras
- Palavras muito raras não são incluídas
- Implementação
  - Lista de palavras, concatenadas em memória separadas com NULL
  - Tabela hash de ponteiros para as palavras

Listas de `hits`

- Contém lista de ocorrências
- Inclui:
  - posição no documento (`imp` no tipo `plain`)
  - fonte usada
  - maiúsculas/minúsculas
- Ocupa maior parte do espaço usado para índice direto e invertido
- Uso de representação otimizada "à mão"
- Dois tipos de `hits`:
  - `fancy` – em URL, título, texto âncora, meta tag
  - `plain` – o resto

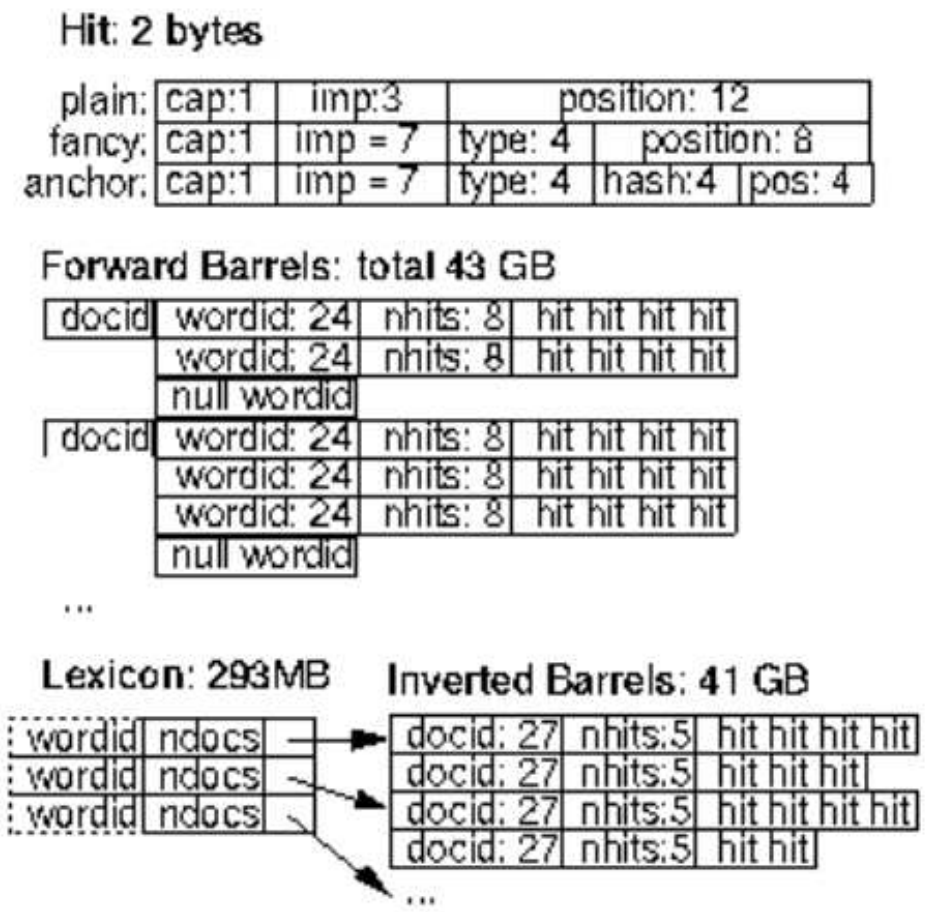


Figure 3. Forward and Reverse Indexes

Índice direto

- Índice direto é criado parcialmente ordenado
- Dividido em 64 barris (barrels)
- cada barrels recebe uma faixa de `wordID`

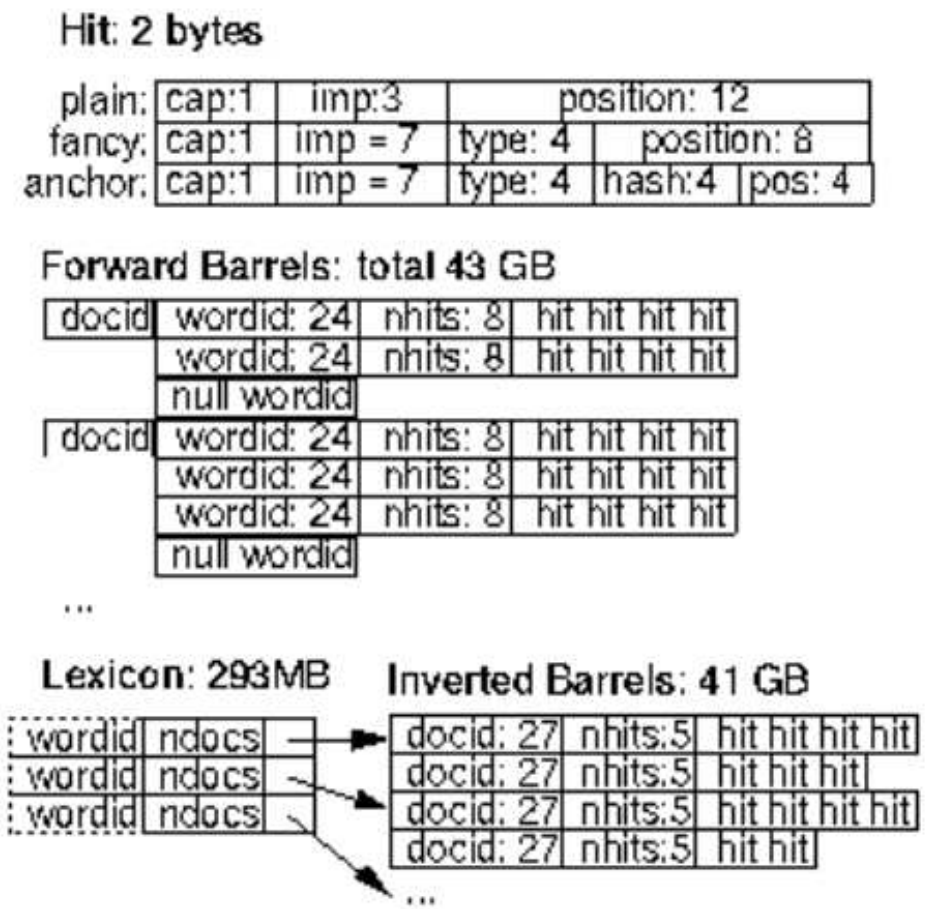


Figure 3. Forward and Reverse Indexes

Índice invertido

- Índice invertido consiste dos mesmos `barrels` que o índice direto
- Porém já foi processado pelo `sorter`
- `doclist` lista `docIDs` ordenados
- Mantém um índice invertido para o título, âncoras e outro índice invertido para o resto

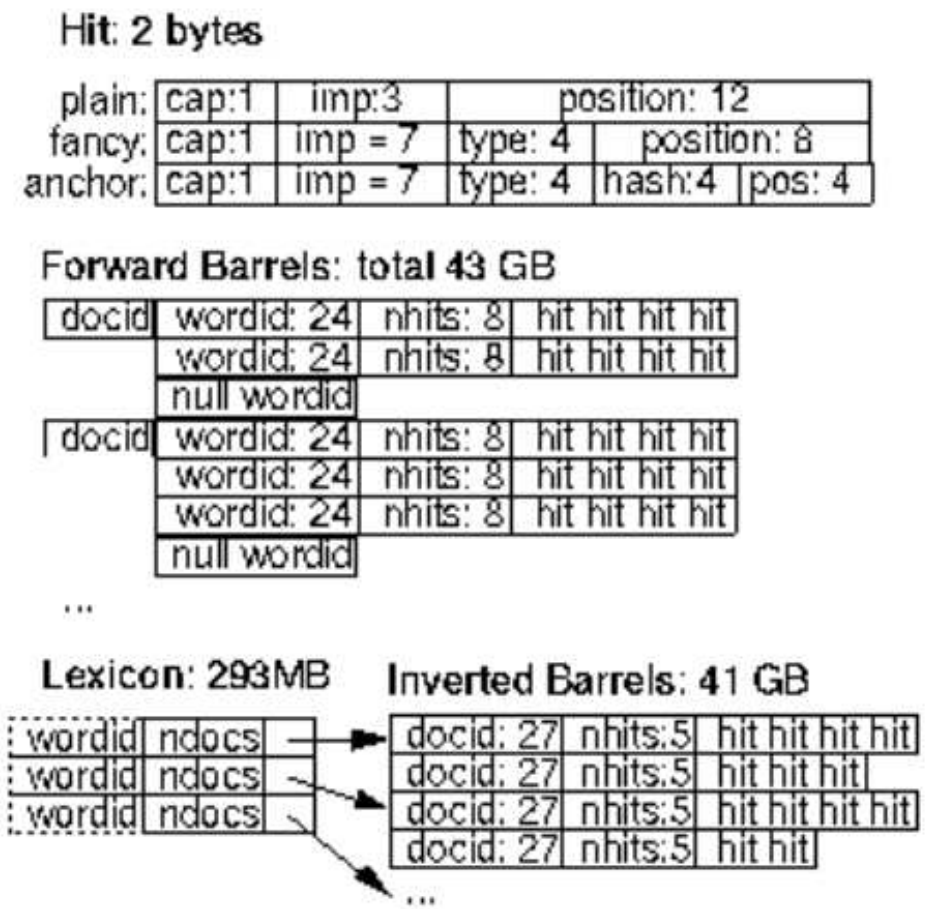


Figure 3. Forward and Reverse Indexes

Crawler

Crawling: tarefa desafiadora.

- questões de desempenho e confiablidade
- questões sociais
- "crawling é a aplicação mais frágil" devido à interação com muitos sistemas, o mais diversos possível
- objetivo: permitir crawling de centenas de milhões de docs

Crawler

Crawler + `URLServer`

- Um `URLserver` envia URLs para vários crawlers (usualmente 3)
- Implementação em Python
- Crawler mantém cerca de 300 conexões abertas simultâneamente
- No auge, captura mais de 100 docs por segundo usando 4 crawlers
- Uso de cerca de 600 kbytes por segundo
- DNS é gargalo – cache é mantido no crawler
- Geração de muitos emails e telefonemas de reclamação
- Muitos não conhecem o protocolo de exclusão de robôs e `robots.txt`

Crawling

- Parsing: implementação de analisador léxico com `flex`
- Erros
  - tags HTML erradas e/ou faltantes
  - bytes que não representam caracteres
  - encadeamente exagerado de tags

A busca

Processo de avaliação de consultas

1. Parsear a consulta
2. Converter palavras em `wordIDs`
3. Fazer consulta no `barrel` mais curto para a lista de documentos de cada palavra
4. Varrer lista de docs até encontrar um documento contendo todos os termos
5. Calcular ranking do documento para a consulta
6. Se terminamos `barrel` mais curto, começar consulta no `barrel` completo voltando ao passo 4
7. Se não estamos no fim de nenhuma lista de docs ir para passo 4
8. ordenar documentos encontrar pelo ranking e retornar os top k

Existe um limite máximo de 40 mil documentos para ranking, que quando alcançado interrompe o processo e retorna resultados

Retorno de relevância

Algoritmos usados no ranking tem parâmetros.

- Valores são obtidos por mecanismos de retorno de relevância.
- Uso de informações fornecidas por usuários confiáveis
- Ranking é modificado com base nessas informações

Requisitos de armazenamento

- Páginas capturadas: 147 GB
- Repositório (compactado): 53GB
- Índice invertido curto: 4 GB
- Índice completo: 37 GB
- Vocabulário: 293 MB
- Total (com outros): 108 GB

Desempenho do sistema

- Cerca de 9 dias para baixar 26 milhões de páginas
- Indexador processava 54 páginas por segundo
- 4 máquinas para o `sorter`: 24 horas
- Tempo de consulta: 1 a 10 segundos
  - Culpado: `BigFile` rodando no NFS

	Initial Query		Same Query Repeated (IO mostly cached)	
Query	**CPU Time(s)**	**Total Time(s)**	**CPU Time(s)**	**Total Time(s)**
al gore	0.09	2.13	0.06	0.06
vice president	1.77	3.84	1.66	1.80
hard disks	0.25	4.86	0.20	0.24
search engines	1.31	9.63	1.16	1.16

Table 2. Search Times

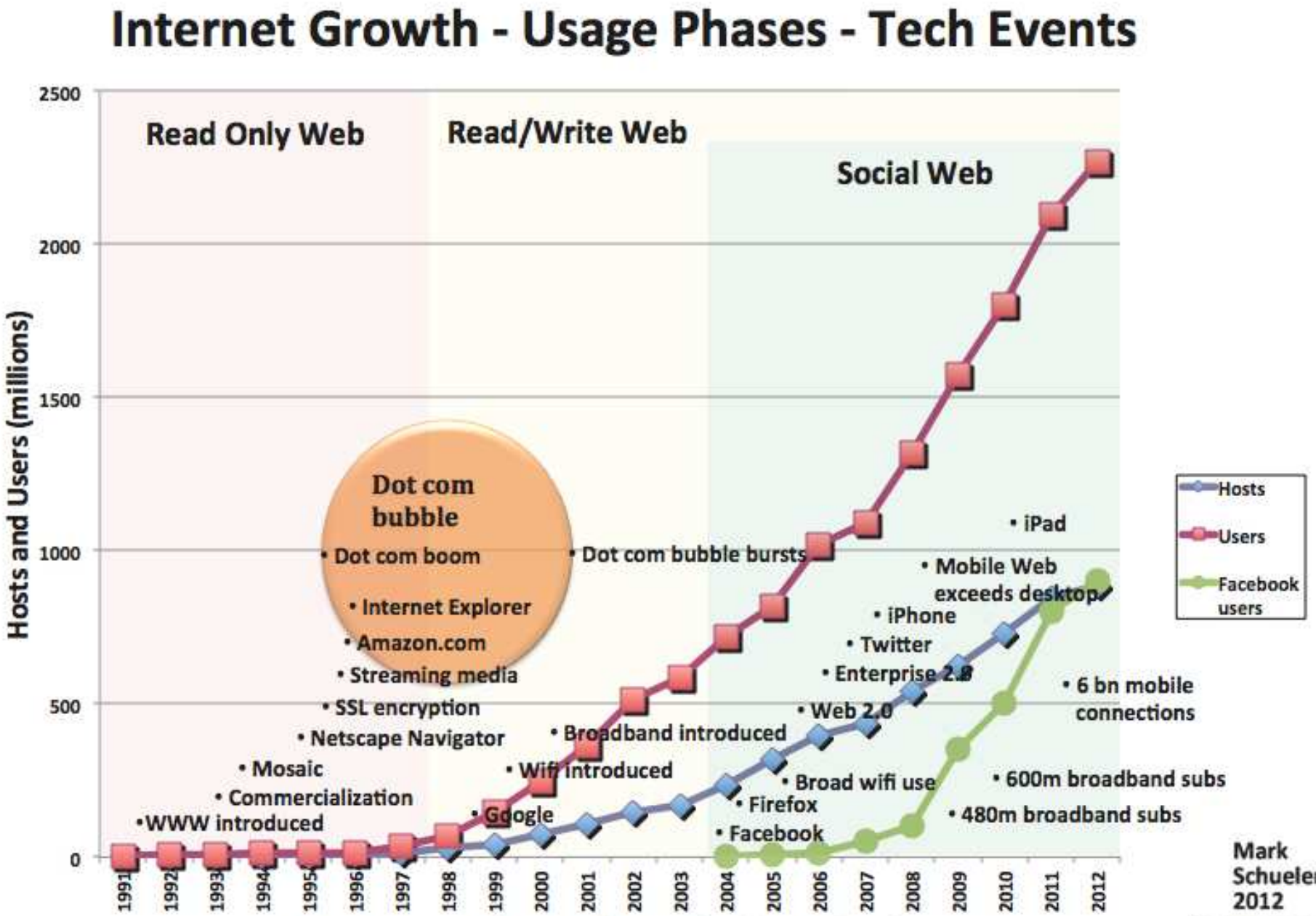
Trabalho futuro de 1997

- Meta imediata: aumentar capacidade para 100 milhões de docs
  - Em setembro de 2013, Google indexa aproximadamente 40 bilhões de documentos
- Cache de consultas
- Subindices
- Melhoria da atualização de documentos – recrawling
- Operadores booleanos
- Negação
- Stemming
- Obtenção de retorno de relevância
- Uso de clustering (no momento somente de nome do servidor)
- Uso da localidade do usuário
- Expansão da região do texto de âncora

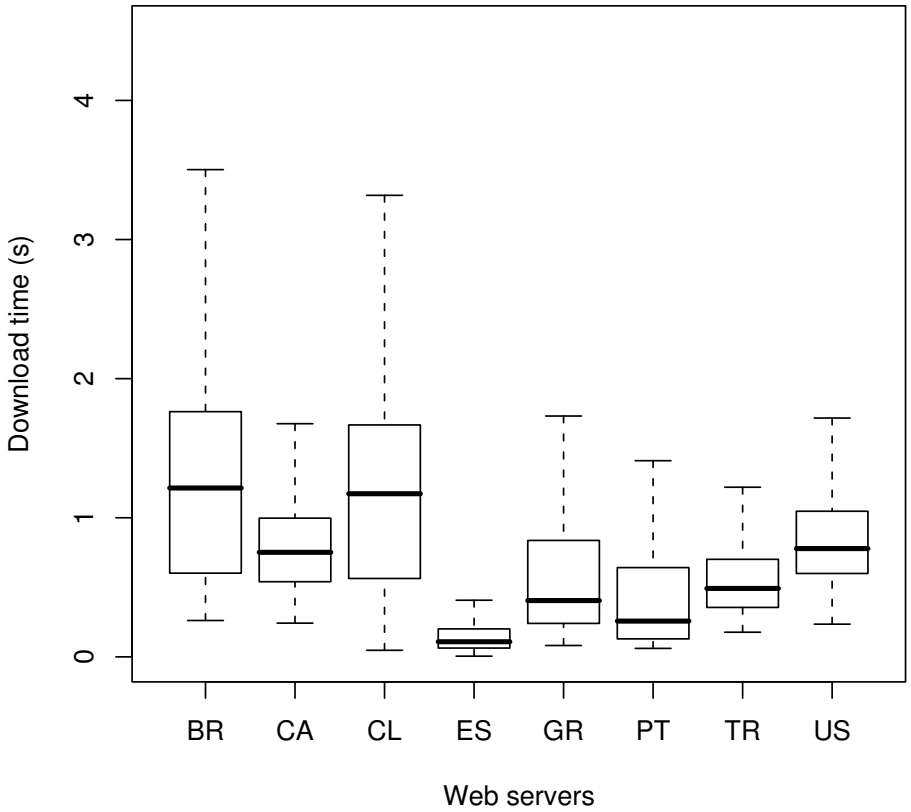
Desafios atuais: segundo o Yahoo Labs (SIGIR, Julho 2014)

Scalability and Efficiency Challenges in Large-Scale Web Search Engines, por Baeza-Yates e Cambazoglu

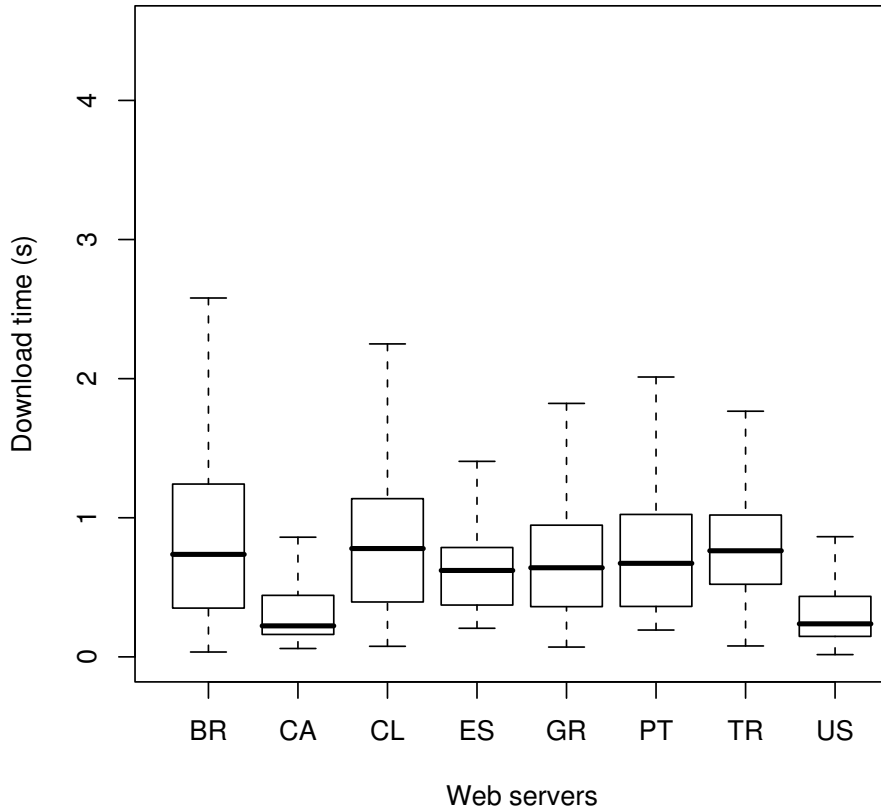
- Desafios = Tamanho + Diversidade + Dinamicidade
- Estimativa de 100 bilhões de documentos, 2.5 bilhões de usuários e 100s milhões de consultas por dia
- Consultas normalmente curtas
- Necessidades informacionais dinâmicas
- Usuários impaciantes (poucas consultas feitas e poucos documentos vistos)


Internet Growth - Usage Phases - Tech Events
Read Only Web
Read/Write Web
Social Web
Hosts and Users (millions)
2500
2000
1500
1000
500
0
Dot com bubble
• Dot com boom
• Internet Explorer
• Amazon.com
• Streaming media
• SSL encryption
• Netscape Navigator
• Mosaic
• Commercialization
• WWW introduced
• Google
• Wifi introduced
• Broadband introduced
• Dot com bubble bursts
• Firefox
• Facebook
• Broad wifi use
• Web 2.0
• Enterprise 2.0
• Twitter
• iPhone
• iPad
• Mobile Web exceeds desktop
• 6 bn mobile connections
• 600m broadband subs
• 480m broadband subs
Hosts
Users
Facebook users
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Mark Schuele
2012

On the Feasibility of Geographically Distributed Web Crawling, Cambazoglu et. al. 2008.



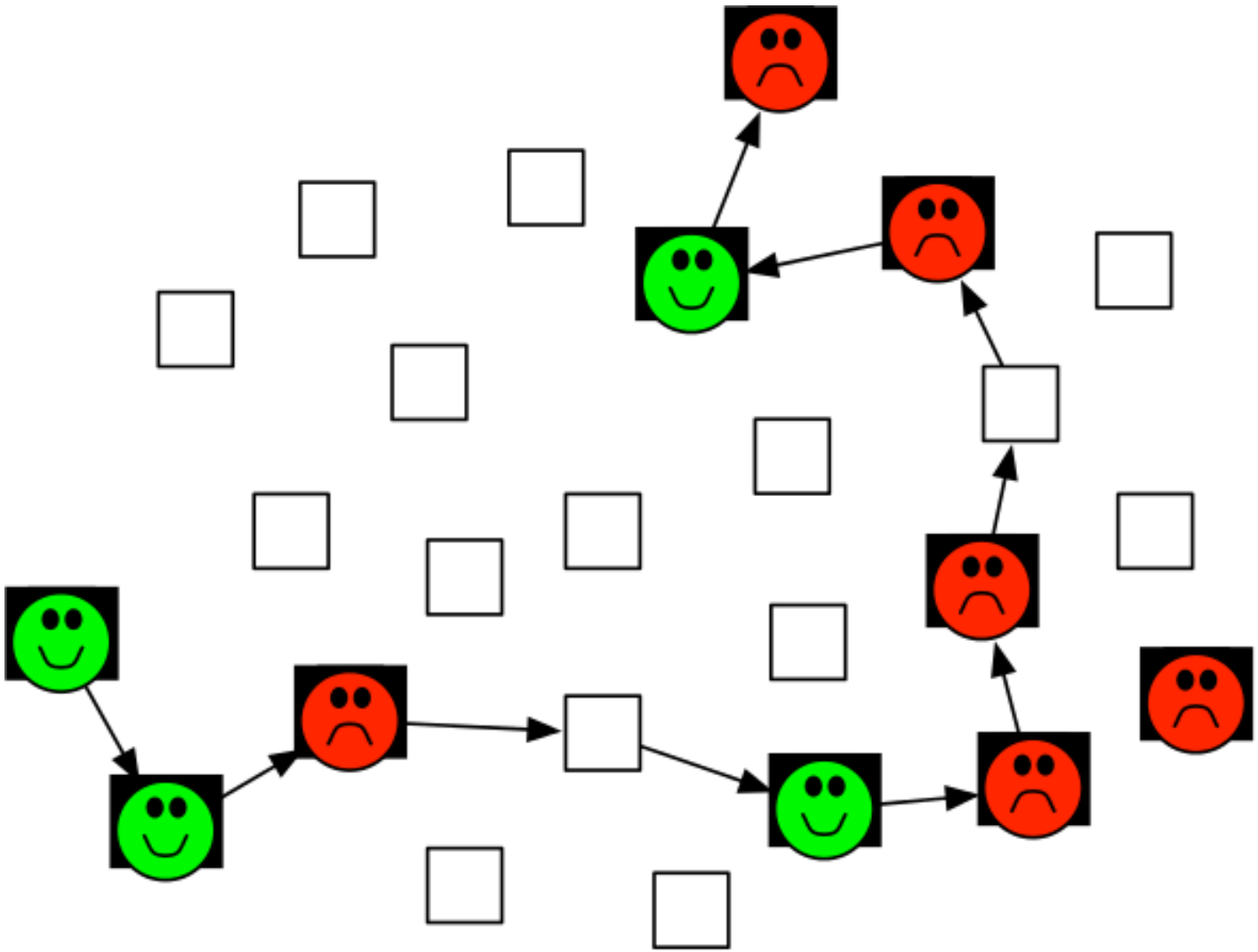
(a) ES crawler



(b) US crawler

Figure 5: Download times for two different crawlers.

Sentiment-focused web crawling. Vural et al. 2012.



A web escondida

Google's Deep Web crawl. Madhavan et al. 2008.

- Páginas não linkadas
- Conteúdo dinâmicos
- Sites privados
- Páginas dependentes de contexto

Crawling: outras questões

IRLbot: Scaling to 6 billion pages and beyond. Lee et al. 2009.

- Planejamento: inserção e remoção de conteúdo
- Redução de custos de crawling (IRLbot)
- Crawling dependente de consultas
- Crawling geográfico – particionamento da web

Indexação atual – relevância

- Pontuação de spam
- Pontuação de qualidade do domínio
- PageRank, TrustRank, SimRank
- Taxa de cliques e tempo de permanência

Indexação

- Compressão da lista de posições em um documento
  - Usando 36 bytes: 17 18 28 40 44 47 56 58
  - Usando saltos gasta-se 18 bytes: 17 1 10 12 4 3 9 2
  - Existem diversas técnicas
- Reordenação das listas de referências de termos
  - Ordenação alfabética e atribuição de ID por URL
  - Agrupamento de documentos com ids similares
  - IDs usando a travessia do grafo de similaridade entre documentos
- Manutenção do índice: criação índices "delta"
  - Não fazer junção com índices antigos
  - Indexação incremental: reservar espaço no fim das listas de referências
  - Junção: reescrever índice
- Particionamento do índice entre computadores
  - Por palavras
  - Por documentos (mais usado)
- Poda de índices para atender a maior parte das consultas

Outros tópicos

- Caching de itens (consultas e docs)
- Ranking com aprendizado de máquina
- Geração de resumos (snippets)
- Redução de latência a usuários
- Economia de energia (1 a 1.5% da energia mundial é consumida por data centers)