Introduction to

# Information Retrieval

## فصل چهارم: ساخت شاخص

علی قنبری سرخی

# مبانی سخت افزاری

- هنگام ساخت سیستم بازیابی اطلاعات، تصمیمات بر اساس ویژگی‌های سخت افزاری که سیستم روی آن اجرا می‌شود، اتخاذ می‌گردد.
- پارامترهای یک سیستم متعارف در سال ۲۰۰۷ در جدول زیر آمده است.

| مقدار | توصیف |
|---|---|
| 5 ms = $5 \times 10^{-3}$ s | متوسط زمان پیگرد (seek) |
| 0.02 µs = $2 \times 10^{-8}$ s | زمان انتقال به ازای هر بایت |
| $10^9$ $s^{-1}$ | نرخ ساعت پردازنده |
| 0.01 µs = $10^{-8}$ s | عملیات سطح پایین (مانند مقایسه و جا به جایی یک کلمه) |
| several GB | اندازه حافظه اصلی |
| 1 TB or more | اندازه حافظه ثانویه |


2

# مبانی سخت افزاری (ادامه)

- دسترسی به داده در حافظه بسیار سریع‌تر از دستیابی به داده روی دیسک است.
- به عبارت دیگر، زمان کمتری برای دستیابی به داده در حافظه (حدود $5\ ns$) به نسبت دستیابی به داده روی دیسک (حدود $20\ ns$) صرف می‌شود.
- بنابراین می‌خواهیم تا جای ممکن تعداد زیادی داده را روی حافظه نگهداریم، به ویژه داده‌هایی که باید به طور مکرر به آن دستیابی انجام شود.
- روش نگهداری داده در دیسک (که مکررا استفاده می‌شود) بر روی حافظه اصلی، کش کردن نام دارد.


3

# مبانی سخت افزاری (ادامه)

- هنگامی که خواندن یا نوشتن دیسک را انجام می‌دهیم، مدتی طول می‌کشد کد هد دیسک به قسمتی از دیسک که داده در آن واقع شده است، برسد. این زمان، زمان پیگرد (حدود $5\ ms$) است.
- هیچ داده ای در زمان پیگرد نمی‌تواند منتقل شود.
- برای بیشینه کردن نرخ انتقال داده، قطعات داده‌ای که با هم خوانده می‌شوند باید روی دیسک به صورت مجاور نوشته شوند.


4

# مبانی سخت افزاری (ادامه)

- برای انتقال ۱۰ مگابایت داده از دیسک به حافظه اصلی در صورتی که به صورت مجاور باشند چقدر زمان نیاز است؟

- برای انتقال ۱۰ مگابایت داده از دیسک به حافظه اصلی در صورتی که در ۱۰۰ قطعه غیرهمجوار باشند، چقدر زمان نیاز است؟


5

# مبانی سخت افزاری (ادامه)

- سیستم‌های عامل معمولا داده‌های دیسک را به صورت بلاک بلاک می‌خوانند(یک بلاک را بافر می‌کند).
- بنابراین زمان خواندن تک بایت با یک بلاک برابر است
- اندازه بلاک‌ها معمولا ۸، ۱۶، ۳۲ و ۶۴ بایت است.


6

# مبانی سخت افزاری (ادامه)

- انتقال داده از دیسک به حافظه توسط گذرگاه سیستم انجام می‌شود. بنابراین پردازنده در این مدت زمانی در دسترس است.
- با توجه به این موضوع، با ذخیره داده‌های فشرده روی دیسک، زمان انتقال داده‌ها را کاهش داد.
- با استفاده از یک الگوریتم فشرده سازی مناسب، زمان خواندن داده فشرده و خارج کردن آن از حالت فشرده معمولا کمتر از زمان خواندن داده غیرفشرده است.


7

# مجموعه Reuters-RCV1

- مجموعه CV1 از رویترز را در نظر بگیرید.
- این مجموعه شامل اسناد روی مجله آنلاین رویترز در طول یک ساله بین ۲۰ آگوست ۱۹۹۶ تا ۱۹ آگوست ۱۹۹۷ است.

REUTERS


You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly En

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprin

[-] Text [-

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian

- این مجموعه شامل اطلاعات چند رسانه‌ای نیز هست که ما با آنها کاری نداریم.


8

# مجموعه Reuters-RCV1 (ادامه)

- متون این مجموعه حدودا چقدر حافظه را اشغال می‌کند؟

| | |
|---|---|
| 800,000 | اسناد |
| 200 | متوسط تعداد نشانه‌ها در هر سند |
| 400,000 | عبارات |
| 6 | متوسط تعداد بایت در هر نشانه (با فضاها و علامتگذاری) |
| 4.5 | متوسط تعداد بایت در هر نشانه (بدون فضاها و علامتگذاری) |
| 7.5 | متوسط تعداد بایت در هر عبارت |
| 100,000,000 | نشانه‌ها |


9

# شاخص گذاری مجموعه‌های بزرگ

- گام‌های اصلی شاخص گذاری

I. ایجاد تمام جفت‌های "شناسه سند – عبارت"

II. مرتب‌سازی تمام جفت‌ها بر اساس عبارت و سپس بر اساس شناسه سند

III. سازماندهی لیست پست‌ها

- برای مجموعه‌های کوچک می‌توان تمامی این گام‌ها را در حافظه انجام داد. برای مجموعه‌های بزرگ این کار ممکن نیست.

# شاخص گذاری مجموعه‌های بزرگ (ادامه)

- با ناکافی بودن حافظه اصلی، ما باید یک "الگوریتم مرتب سازی خارجی" را که از دیسک استفاده می‌کند، به کار ببریم.
- برای دستیابی به سرعت قابل قبول، چنین الگوریتمی باید تعداد پیگردهای تصادفی دیسک را طول مرتب سازی کمینه کند. چرا؟


11

# شاخص گذاری بلوکی مبتنی بر مرتب سازی (BSBI)

- BSBI مخفف الگوریتم Block Sort- Based Indexing است.
- الگوریتم BSBI به صورت زیر است.

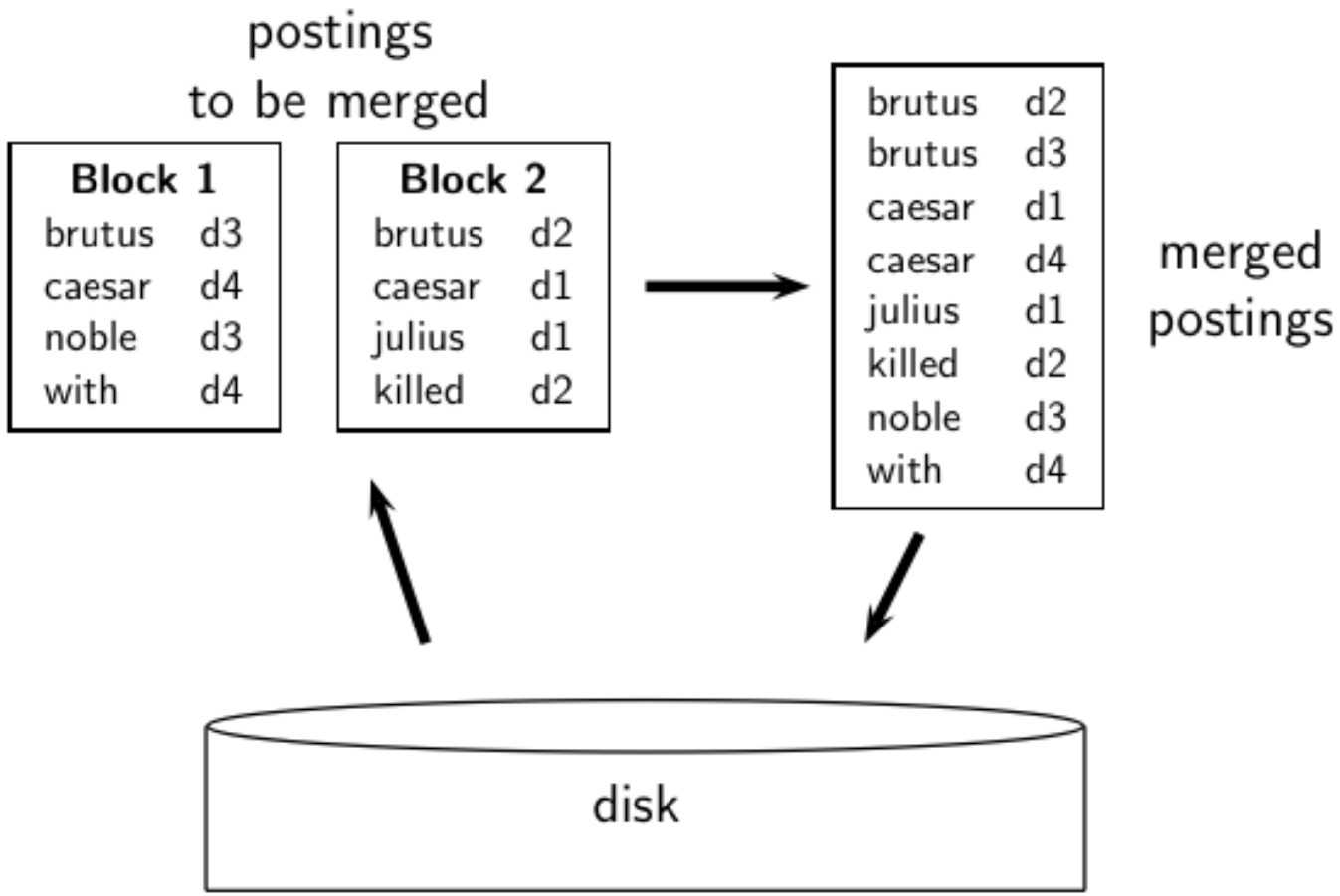
I. مجموعه را به قسمت‌های با اندازه مساوی قطعه بندی می‌کند.

II. جفت‌های "شناسه سند – شناسه عبارت" از هر قطعه را در حافظه مرتب می‌کند و سپس نتایج در حافظه ذخیره می‌شود.

III. تمامی نتایج میانی در شاخص نهایی ادغام می‌شود.


12

# شاخص گذاری بلوکی مبتنی بر مرتب سازی (BSBI) (ادامه)





13

# شاخص گذاری بلوکی مبتنی بر مرتب سازی (BSBI) (ادامه)

```
BSBIndexConstruction()
1  n ← 0
2  while (all documents have not been processed)
3  do n ← n + 1
4      block ← ParseNextBlock()
5      BSBI-Invert(block)
6      WriteBlockToDisk(block, f_n)
7  MergeBlocks(f_1, ..., f_n; f_merged)
```


14

# شاخص گذاری درون حافظه تک گذره

- شاخص‌گذاری بلوکی مبتنی بر مرتب سازی، مقیاس پذیری خوبی دارد، اما نیاز به ساختمان داده‌ای برای نگاشت عبارات به شناسه های آنها دارد.
- برای مجموعه های بزرگ، این ساختمان داده در حافظه گنجانده نمی شود. یک راه مقیاس پذیرتر، شاخص گذاری درون حافظه ای تک گذره (Single-Pass In-Memory Indexing) یا (SPIMI) است.
- SPIMI عبارات را به جای شناسه های آنها به کار می برد، هر لغت‌نامه یک بلوک را روی دیسک می نویسد، و سپس یک لغت نامه جدید را برای بلوک بعدی در نظر می‌گیرد.
- SPIMI می‌تواند مجموعه های با هر اندازه را تا زمانیکه فضای کافی در دیسک موجود باشد شاخص کند.


15

# شاخص گذاری درون حافظه تک گذره

- در این الگوریتم از بخشی که اسناد را تجزیه و آنها را به جریان جفت های (عبارت–شناسه سند) تبدیل میکند صرف نظر میکنیم.
- در اینجا زوج (عبارت–شناسه سند) نشانه نامیده می شود.
- الگوریتم روی جریان نشانه‌ها، تا زمانیکه کل مجموعه پردازش شود مکرراً فراخوانی می شود.
- نشانه ها یکی یکی در طول هر فراخوانی متوالی از الگوریتم پردازش می شوند. (خط ۴)
- هنگامی که یک عبارت برای اولین بار اتفاق می افتد، به لغت نامه اضافه می شود. (بهترین پیاده سازی به صورت یک درهم سازی است) و یک لیست پست جدید ایجاد می شود (خط ۶).
- فراخوانی در خط (۷)، این لیست پست ها را برای وقوع بعدی عبارت بر میگرداند.


16

# شاخص گذاری درون حافظه تک گذره

- یک تفاوت بین دو الگوریتم BSBI و SPIMI این است که SPIMI یک پستر ا مستقیم به لیست پست آن می افزاید (خط ۱۰). بجای اینکه اول جفت‌های (شناسه سند- شناسه عبارت) را جمع آوری کرده و سپس آنها را مرتبط کند (همانطور که در الگوریتتم BSBI انجام می شد)، هر لیست پست پویا یوده (یعنی اندازه ی آن با رشد آن تنظیم می شد) و برای جمع آوری پستها فورا در دسترس است.
  - مزیت این روش: سریعتر است چون به مرتب سازی نیاز ندارد
  - به حافظه کمتری نیاز دارد. چون به دنبال عبارتی که یک لیست پست به آن تعلق دارد هستیم، بنابراین نیازی به ذخیره شناسههای عبارت پست ها نیست
  - در نتیجه اندازه بلوک هایی که در SPIMI می تواند پردازش کند بزرگتر بوده و فرآیند ساخت شاخص به طور کلی کارامدتر است.

# شاخص گذاری درون حافظه تک گذره

- به دلیل اینکه نمی دانیم لیست پستها یک عبارت زمانیکه اولین بار با آن مواجعه می شویم چقدر بزرگتر خواهد شد، از ابتدا فضایی به اندازه یک لیست پست کوتاه تخصیص داده و هر زمانی که پر شد فضا را دو برابر می کنیم. (خط ۸-۹)
- هنگامی که حافظه مصرف شد شاخص بلوک را (که شامل لغت نامه و لیست پست ها است) در دیسک می نویسیم (خط ۱۲)
- ما باید عبارات را (خط ۱۱) قبل از انجام اینکار مرتب کنیم. چون می خواهیم برای بهبود گام نهایی ادغام (در اینجا نوشته نشده است) لیست ها را به ترتیب لغت نامه ای بنویسیم.
- اگر لیست پست های هر بلوک به صورت نامرتب نوشته شده باشند ادغام بلوک ها نمی تواند توسط اسکن خطی ساده در طول هر بلوک انجام شود.


18

# شاخص گذاری درون حافظه تک گذره

```
SPIMI-Invert(token_stream)
 1  output_file ← NewFile()
 2  dictionary ← NewHash()
 3  while (free memory available)
 4  do token ← next(token_stream)
 5     if term(token) ∉ dictionary
 6        then postings_list ← AddToDictionary(dictionary, term(token))
 7        else postings_list ← GetPostingsList(dictionary, term(token))
 8     if full(postings_list)
 9        then postings_list ← DoublePostingsList(dictionary, term(token))
10     AddToPostingsList(postings_list, docID(token))
11  sorted_terms ← SortTerms(dictionary)
12  WriteBlockToDisk(sorted_terms, dictionary, output_file)
13  return output_file
```

Merging of blocks is analogous to BSBI.

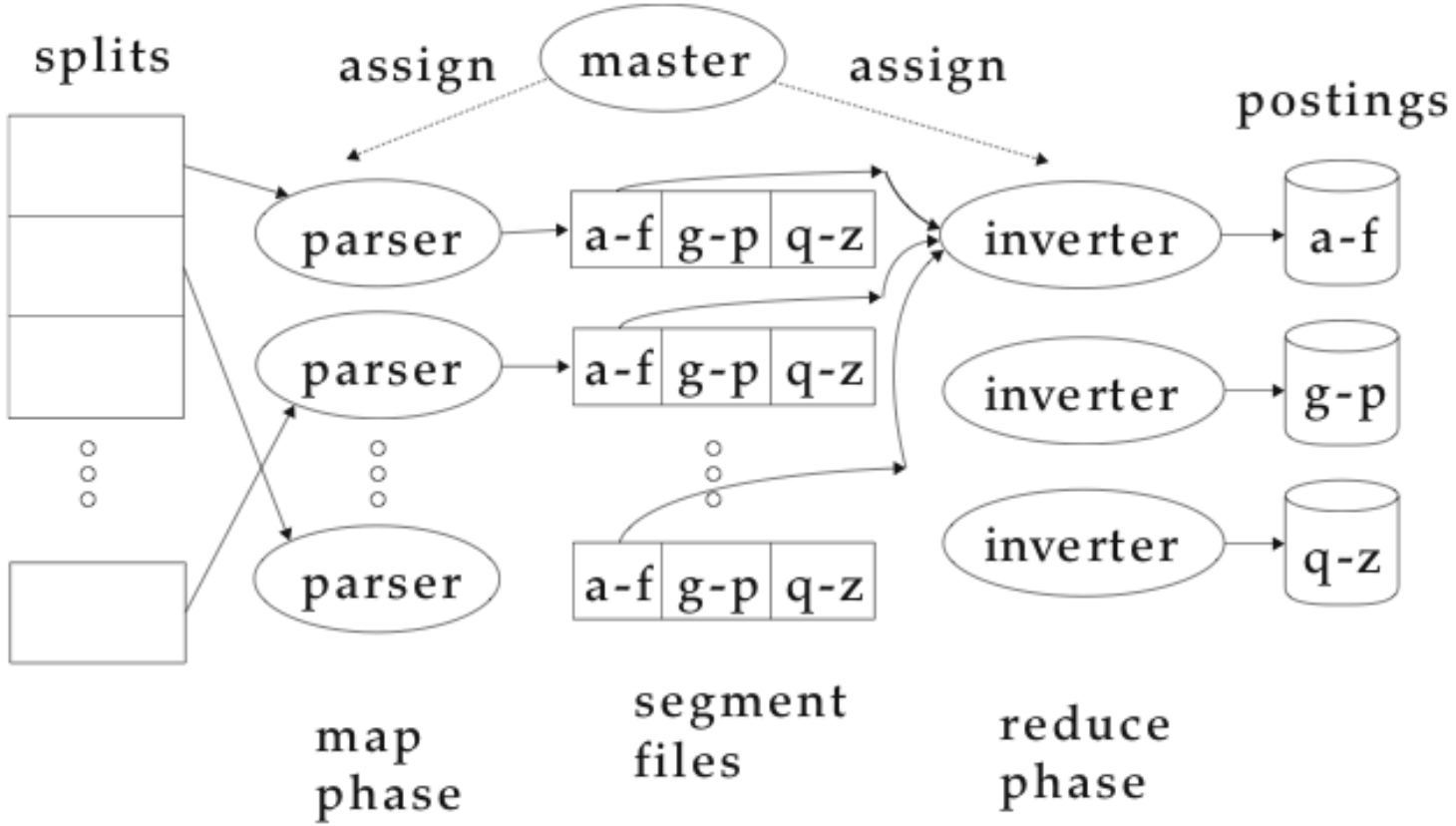

19

# شاخص گذاری توزیع شده

- اغلب مجموعه‌ها آنقدر بزرگ هستند که نمی‌توانیم شاخص را به طور کارآمد روی یک ماشین بسازیم. این شرایط به ویژه در مورد شبکه جهانی وب صادق است.
- در این بخش روشی بر پایه معماری MapReduce معرفی شده است.
- در مرحله نگاشت (map phase) داده ورودی به جفت‌های "مقدار – کلید" نگاشت می‌شود.
- در مرحله کاهش (reduce phase)، شاخص وارونه ساخته می‌شود.


20

# شاخص گذاری توزیع شده (ادامه)





21

# Index construction in MapReduce

**Schema of map and reduce functions**

| | | |
|---|---|---|
| map: | input | $\rightarrow$ list($k$, $v$) |
| reduce: | ($k$,list($v$)) | $\rightarrow$ output |

**Instantiation of the schema for index construction**

| | | |
|---|---|---|
| map: | web collection | $\rightarrow$ list(termID, docID) |
| reduce: | ($\langle$termID$_1$, list(docID)$\rangle$, $\langle$termID$_2$, list(docID)$\rangle$, ...) | $\rightarrow$ (postings_list$_1$, postings_list$_2$, ...) |

**Example for index construction**

| | | |
|---|---|---|
| map: | $d_2$ : C DIED. $d_1$ : C CAME, C C'ED. | $\rightarrow$ ($\langle$C, $d_2\rangle$, $\langle$DIED,$d_2\rangle$, $\langle$C,$d_1\rangle$, $\langle$CAME,$d_1\rangle$, $\langle$C,$d_1\rangle$, $\langle$C'ED,$d_1\rangle$) |
| reduce: | ($\langle$C,($d_2$,$d_1$,$d_1$)$\rangle$,$\langle$DIED,($d_2$)$\rangle$,$\langle$CAME,($d_1$)$\rangle$,$\langle$C'ED,($d_1$)$\rangle$) | $\rightarrow$ ($\langle$C,($d_1$:2,$d_2$:1)$\rangle$,$\langle$DIED,($d_2$:1)$\rangle$,$\langle$CAME,($d_1$:1)$\rangle$,$\langle$C'ED,($d_1$:1)$\rangle$) |



22

# شاخص گذاری توزیع شده (ادامه)

- یک "گره اصلی"(master) فرایند انتساب و انتساب مجدد وظایف را به "گره‌های کارگر"(worker) منفرد اداره می‌کند.
  - اگر وظیفه یک کارگر به اتمام برسد، گره اصلی وظیفه جدیدی را به او محول می‌کند.
  - اگر یک کارگر دچار مشکلی شود، گره اصلی وظیفه او را به کارگر دیگری محول می‌کند.


23

# شاخص گذاری پویا

- تاکنون فرض کردیم که مجموعه اسناد ایستا هستند. این فرض برای مجموعه‌هایی که به ندرت و یا هرگز تغییر نمی‌کنند، مناسب است.(مثل کتاب شکسپیر یا قرآن)
- اکثر مجموعه‌ها با اضافه کردن، حذف کردن و یا به روز رسانی اسناد تغییر می‌کنند.
- ساده‌ترین روش این است که به طور دوره‌ای شاخص را از ابتدا بسازیم.
- این کار در شرایطی مناسب است که تعداد تغییرات در طول زمان اندک باشد و تاخیر در ایجاد قابلیت جستجوی اسناد جدید قابل قبول باشد.


24

# شاخص گذاری پویا (ادامه)

- اگر الزامی وجود داشته باشد که اسناد جدید باید به سرعت قابل جستجو شوند، یک راه حل استفاده از دو شاخص است. یک شاخص اصلی بزرگ و یک شاخص کمکی کوچک(برای اسناد جدید).
- جستجوها روی هر دو شاخص انجام می‌شود و نتایج ادغام می‌گردد.
- می‌توان حذفیات را نیز ذخیره کرد و سپس آنها را از نتایج قبلی حذف نمود.
- هر زمان که شاخص کمکی به قدر کافی بزرگ شد، آنرا با شاخص اصلی ادغام می‌کنیم.


25