现代计算机的体系结构

$$CPU \rightarrow Cache \rightarrow RAM \rightarrow Hard\ Disk \rightarrow OffLine\ Storage$$

小、快、贵 → 大、慢、廉

> Data Access Locality (More High More Efficient)

Problem 1

内存无法容纳所有的索引项

Solution

BSBI： Bolcked sorted-based indexing

BSBI

> 思想：
>
> - 同时使用内存和磁盘。
> - 把尽量多的工作放到内存内完成。
> - 对磁盘的访问尽量使用顺序读写。
>
> 方法：
>
> - 将文档集分块，使得每块都能被内存容纳。
> - 在内存中对每块的内容进行排序，并写回磁盘。
> - 将各块的排序进行合并，并创将posting lists.

BSBI算法复杂度：4SN 约为将文档读入内存四次的代价

```
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)
```

Figure 4.2 Blocked sort-based indexing. The algorithm stores inverted blocks in files $f_1, \ldots, f_n$ and the merged index in f_{merged}.

Problem 2

单个计算机无法容纳所有的索引项

Solution

分布式系统－Cluster

分布式的倒排索引

- Partition By Terms (有些节点会被密集访问，有的节点就浪费了 / 负载不平衡)
- Partition By Docs (Google Choose This Way)

MapReduce

1. Map（将一个大任务分解为多个小任务，解决每个小任务）
2. Reduce （将小任务大答案进行合并，以得到大任务的答案）

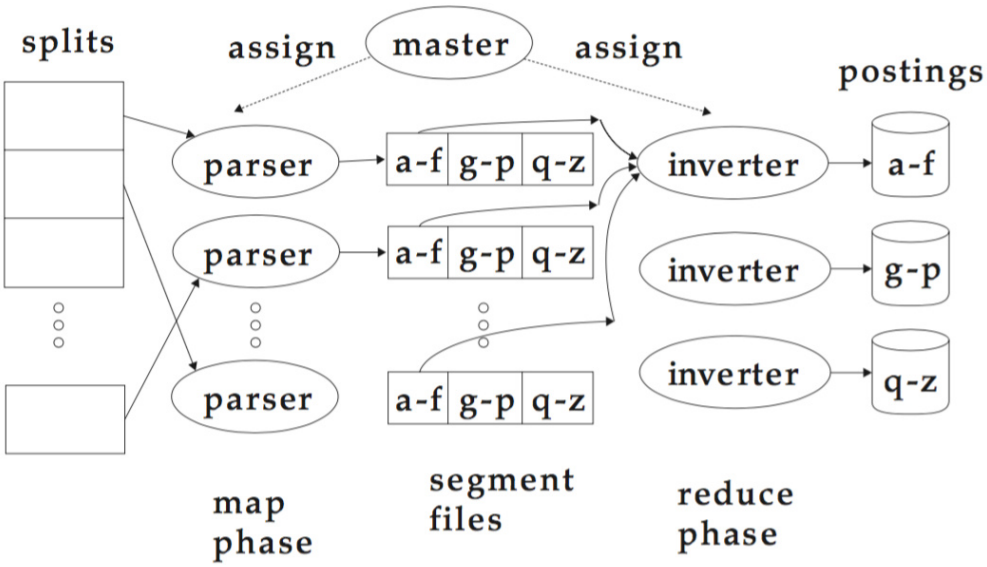

Figure 4.5 An example of distributed indexing with MapReduce. Adapted from Dean and Ghemawat (2004).

Problem 3

文档在不断增加，索引项在不断增加

Solution

动态索引

- 两个倒排索引：
  - 位于磁盘上的静态索引
  - 位于内存中的动态辅助索引
- 查询时合并两个索引结果
- 新来的文档加入辅助索引
- 当辅助索引过满时，合并到磁盘上的主索引，再清空
- 复杂度： $O(n^2)$

Logarithmic Merge

- 多个索引
  - 最底层主索引

    - 上层索引大小为下层的1 / 2
    - 最高层位于内存中，其余位于磁盘上
- 查询时合并多个索引的结果
- 当上层索引填满时，与下一层索引合并
- 复杂度： $O(nlogn)$

```
LMergeAddToken(indexes, Z_0, token)
 1  Z_0 ← Merge(Z_0, {token})
 2  if |Z_0| = n
 3     then for i ← 0 to ∞
 4          do if I_i ∈ indexes
 5                then Z_{i+1} ← Merge(I_i, Z_i)
 6                     (Z_{i+1} is a temporary index on disk.)
 7                     indexes ← indexes − {I_i}
 8                else I_i ← Z_i     (Z_i becomes the permanent index I_i.)
 9                     indexes ← indexes ∪ {I_i}
10                     Break
11          Z_0 ← ∅
```

```
LogarithmicMerge()
1  Z_0 ← ∅     (Z_0 is the in-memory index.)
2  indexes ← ∅
3  while true
4  do LMergeAddToken(indexes, Z_0, getNextToken())
```