分布式索引构建

恨少

一淘搜索技术部

2012 年 3 月 19 日

1. 引擎基础知识
2. Hadoop 工具链介绍
3. abuild 原理
4. kbuild 原理
5. Hadoop 相关配置及 Job 优化

索引种类

- 检索
  使用倒排索引,(“手机”, (宝贝 0, 宝贝 10, 宝贝 15)).
- 过滤和统计.
  使用正排索引,((宝贝 0, (价格:1500));(宝贝 1, (价格:1800))).
- 返回宝贝
  使用宝贝原始数据.

I'm Feeling Lucky

真实的 query

q= 韩版女装春装
&atype=b
&isnew=2
&advsort=advtaobao
&psweight=1&mlrfield=category
&ps=ends&ss=ends
&dfsort=16&pgnum=44
&!et=1332562892&n=44
&userloc= 浙江
&distfield=user_id:pid&distcnt=2:4
&filter=_level_one_cat:50023717
&statistic=field=catmap,count=500,parentcatid=1,percent=0
&mmfilter=reserve_pricediscount:promotions[100,200]
&src=ss-srp-s006020.cm8

iSearch 索引

- 支持多个 area, 每个 area 是一份完整的索引.
- 索引分段.
- 每个字段的正排索引存储在不同的文件中.
- 文档原始数据不分离.

倒排: shop.user_id.idx.seg_247 shop.user_id.doclist.seg_247

正排: shop.user_id.idx.pfl.seg_247

原始数据: shop.idx.dtl.seg_247 shop.dat.dtl.seg_247

Kingso 索引

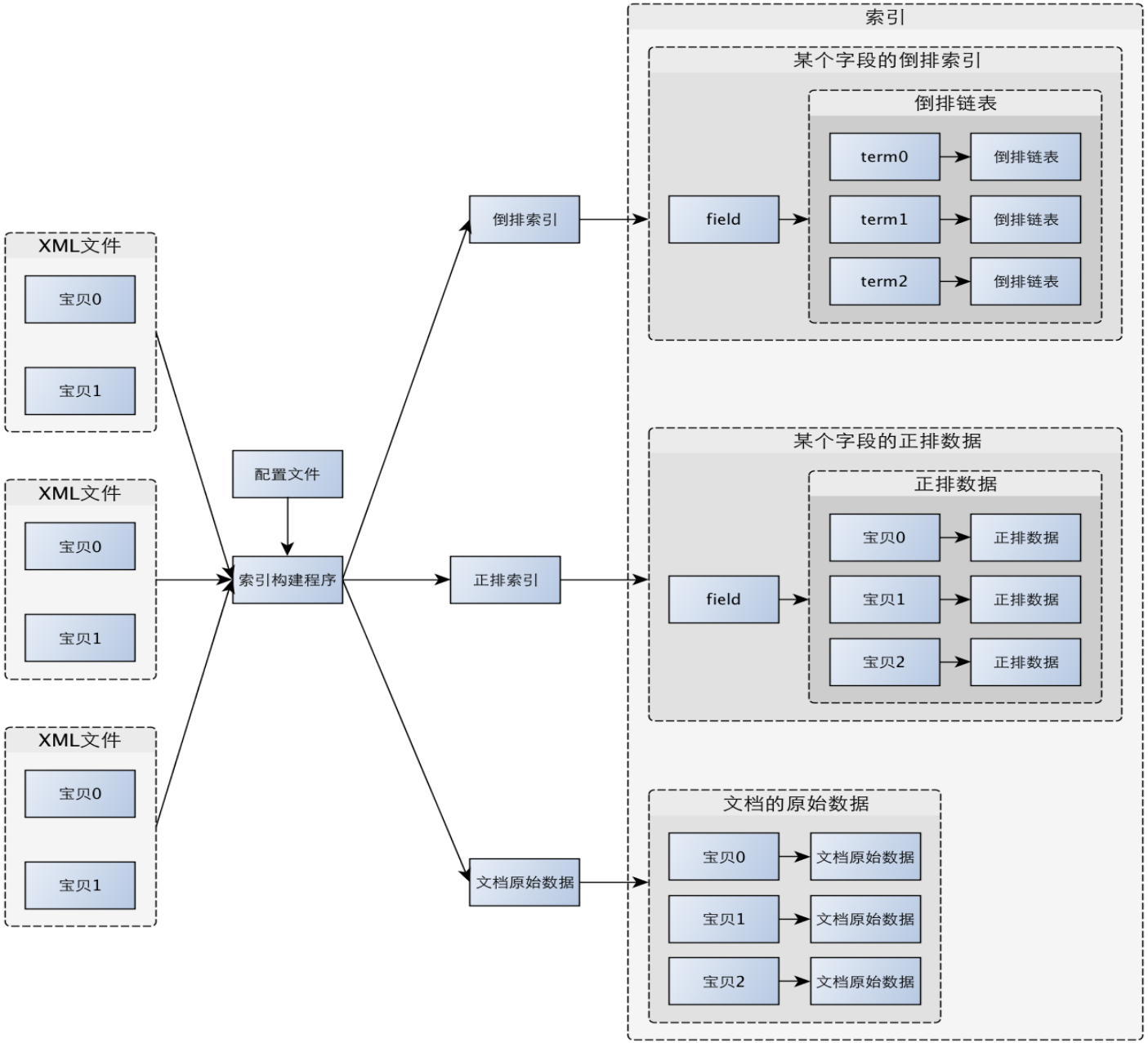
- 索引不分段.
- 每个文档的正排数据集中存储.
- 分离文档原始数据, 有专门的服务提供查询.

倒排: user_id.bitmap_index user_id.bitmap_term user_id.term

正排: profile_group_0.seg_0 algfield.encode_cnt algfield.encode_idx

原始数据: detail.idx detail.dat

索引构建示意图


索引
某个字段的倒排索引
倒排链表
term0
倒排链表
term1
倒排链表
term2
倒排链表
field
某个字段的正排数据
正排数据
宝贝0
正排数据
宝贝1
正排数据
宝贝2
正排数据
field
文档的原始数据
宝贝0
文档原始数据
宝贝1
文档原始数据
宝贝2
文档原始数据
倒排索引
正排索引
文档原始数据
XML文件
宝贝0
宝贝1
XML文件
宝贝0
宝贝1
XML文件
宝贝0
宝贝1
配置文件
索引构建程序

主搜索索引构建问题

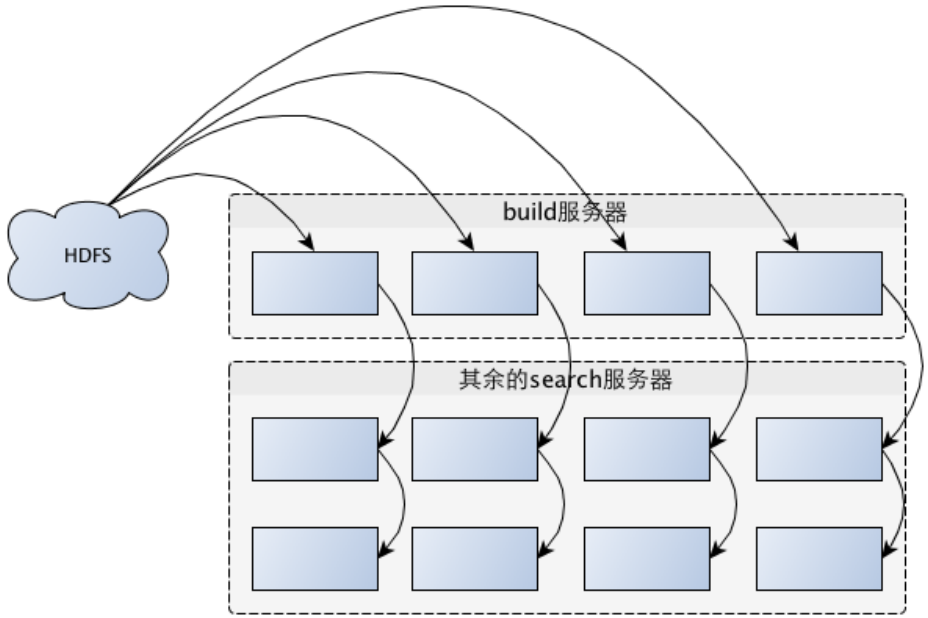
商品数量巨大, 索引构建是个难题.

- 数亿商品
- 数百 GB 存放在 Hadoop 上的压缩 XML 数据

现有索引构建流程

目前单机建索引的流程:

- 拉数据.
- 建索引.
- 链式分发索引.



单机构建缺点

- 建索引时间长
  对 1900w 商品建索引耗时 40 分钟.
- 分发索引时间长
  将所有索引分发完, 需要 60 分钟.
- 容错能力差
  索引构建时遇到硬件故障, 目前需要人工进行干预.

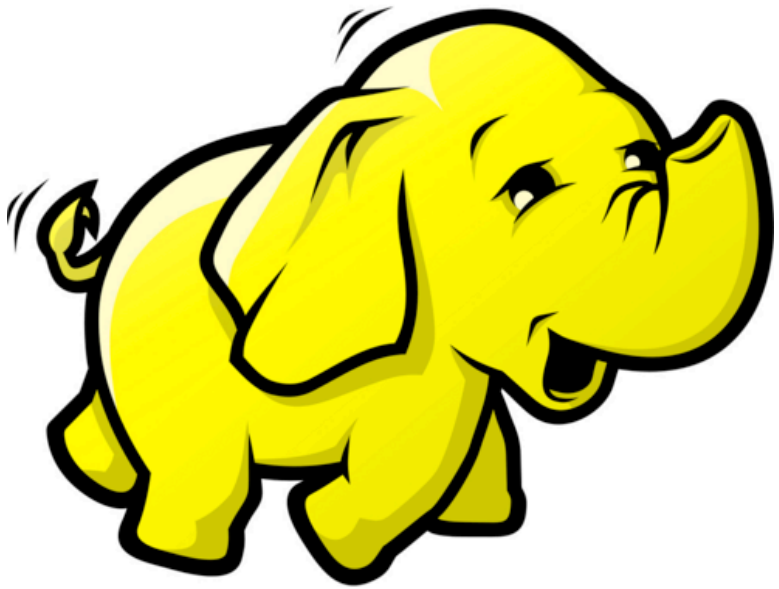
更好的构建方式?

更好的构建方式!

分布式索引构建的优点:

- XML 数据是分布式平台产出的, 处理起来更加方便.
  节省 6-10 分钟拉数据的时间.
- 将构建任务进行分解, 分布式建索引, 速度更快.
  将索引构建的时间减少一半.
- 利用 HDFS 多份拷贝, 加快索引分发速度.
  将集群分成几个独立的行组, 分别进行链式拷贝.
- 可靠性有保障, 不会因为单台服务器硬件故障而失败.
  再也不担心硬件故障了.

云端小飞象



Hadoop Streaming

Hadoop streaming is a utility that comes with the Hadoop distribution. The utility allows you to create and run map/reduce jobs with any executable or script as the mapper and/or the reducer. For example:

```
\$HADOOP\_HOME/bin/hadoop
    jar \$HADOOP\_HOME/hadoop-streaming.jar \
    -input myInputDirs  \
    -output myOutputDir \
    -mapper /bin/cat    \
    -reducer /bin/wc
```

Hadoop Pipes

Hadoop Pipes allows C++ code to use Hadoop DFS and map/reduce.

Hadoop Pipes 主要缺点是:

- 出问题难于调试, 只能通过打印语句跟踪问题.
- 升级 Hadoop 时, 需要重新编译程序以链接新的 Pipes 库, 否则可能有不兼容情况.
- 效率比较低,map 和 reduce 得到的都是 string 字符串, 非字符类型需要进行一次转换.

```
void map(HadoopPipes::MapContext& context) {
    std::string strDoc = context.getInputValue();
}
```

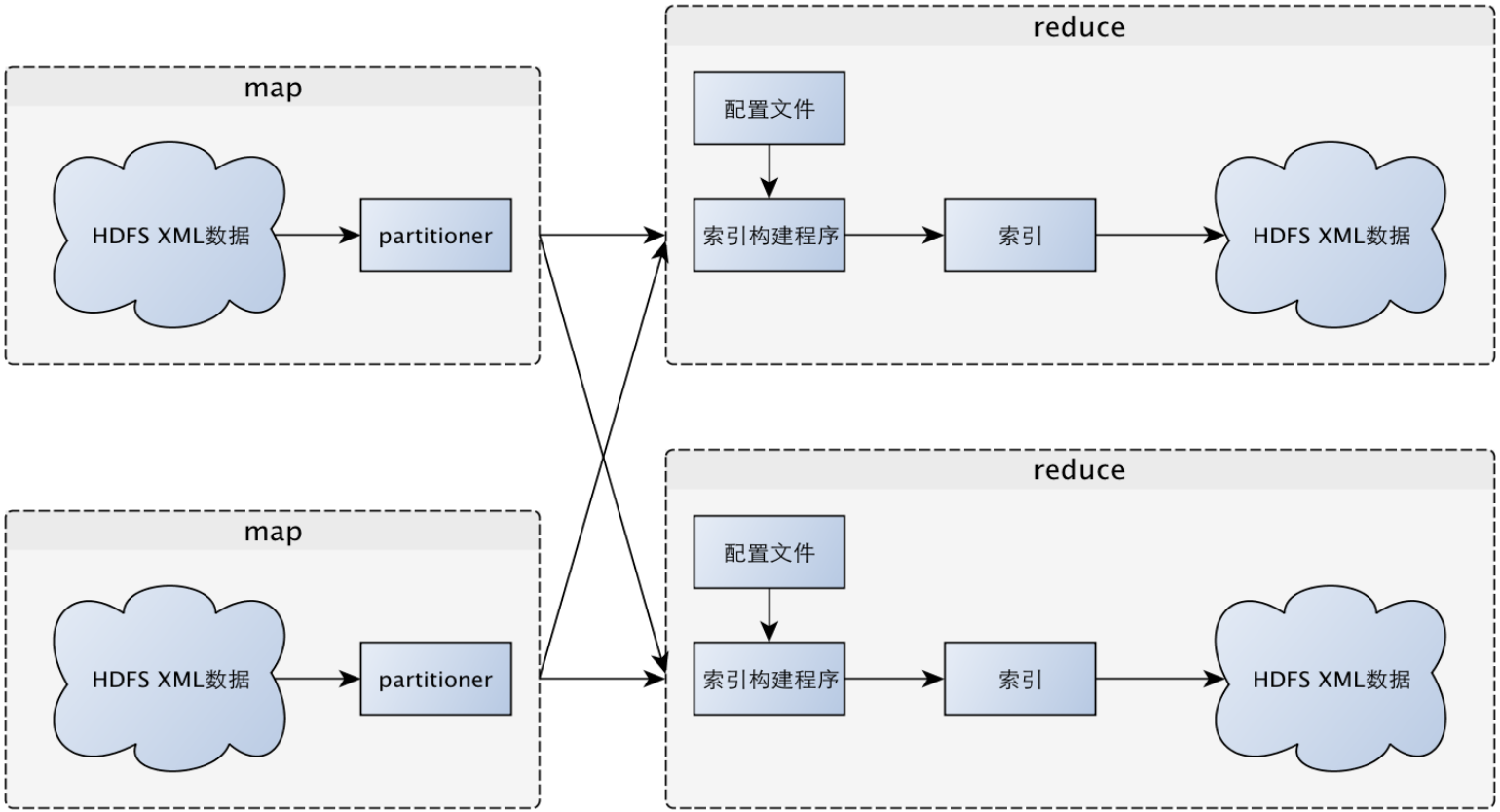
思路

abuild 利用 Hadoop Pipes 构建 iSearch 的索引, 被用于构建一淘的索引.abuild 索引构建分为两个过程:

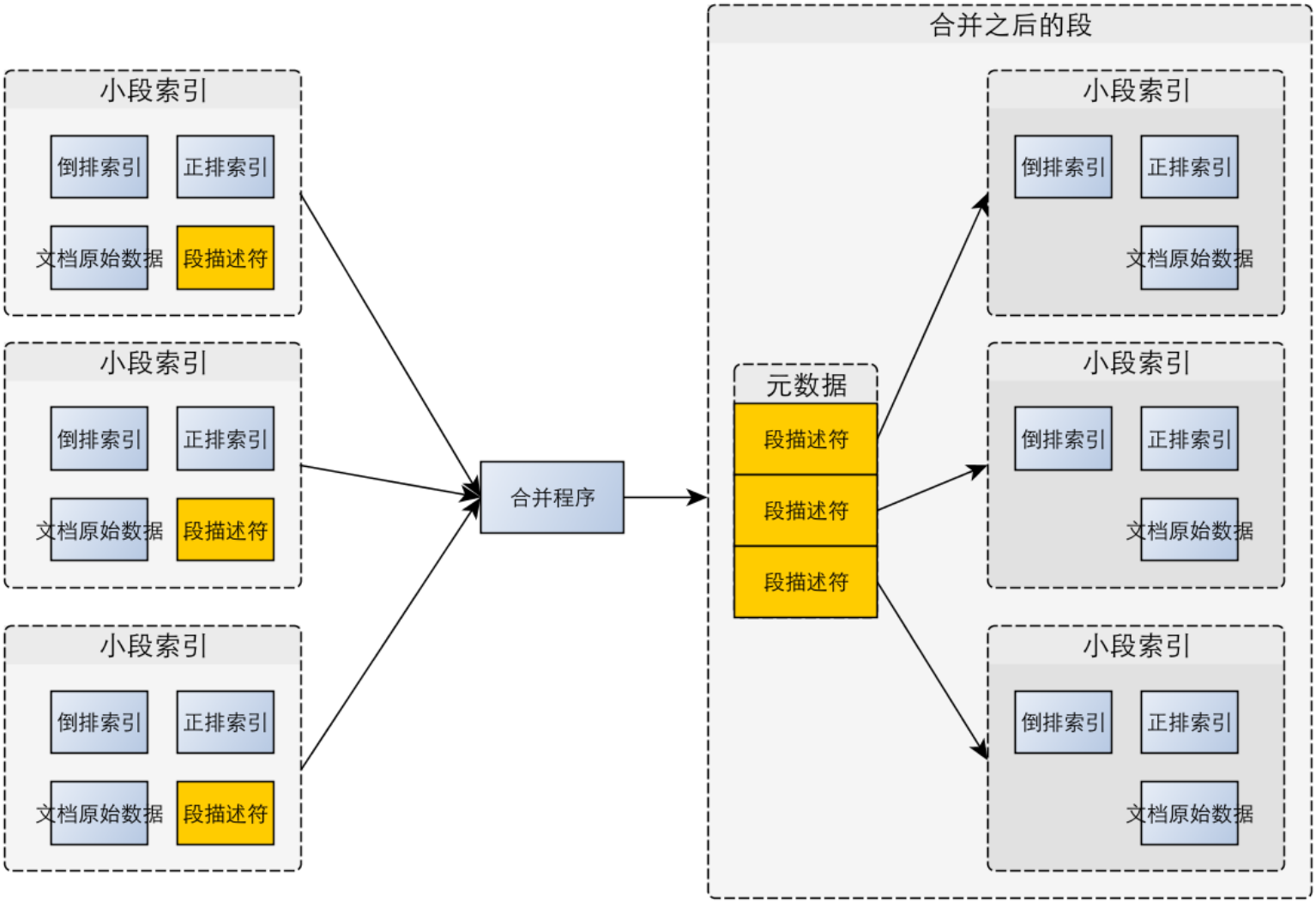
- 建小段索引.
  从 HDFS 上读入一个个文档, 建成许多小段索引.

- 合并小段索引的元数据.
  将第一个过程创建的小段索引进行合并, 比如将 384 个小段索引合并成 48 个, 合并后每份索引包含 8 个段. 由于只涉及元数据的合并, 所以速度非常快.

建小段索引示意图


map
HDFS XML数据
partitioner
map
HDFS XML数据
partitioner
reduce
配置文件
索引构建程序
索引
HDFS XML数据
reduce
配置文件
索引构建程序
索引
HDFS XML数据

合并示意图


小段索引
倒排索引
正排索引
文档原始数据
段描述符
小段索引
倒排索引
正排索引
文档原始数据
段描述符
小段索引
倒排索引
正排索引
文档原始数据
段描述符
合并程序
合并之后的段
元数据
段描述符
段描述符
段描述符
小段索引
倒排索引
正排索引
文档原始数据
小段索引
倒排索引
正排索引
文档原始数据
小段索引
倒排索引
正排索引
文档原始数据

一次 Hadoop 升级导致的问题

Hadoop 从 0.19.1 升级到 0.20.2

发现 abuild 每次运行都失败。最后发现是 Hadoop Pipes 新旧版本不兼容导致，需要修改项目的 configure.ac，让 abuild 在不同的环境下，链接不同版本的 Hadoop Pipes 库。

思路

kbuild 用于在 Hadoop 上构建 Kingso 的索引.kbuild 使用 Hadoop Streaming, 主要基于以下几个想法:

- 效率高.
  由于 HDFS 上的原始数据已经有规律,(nid/列数) 相同的宝贝会放在同一个 part 里面, 并不需要进行 map sort 和 partition 操作.

- 减少外部依赖.
  索引合并可以在非 Hadoop 环境运行, 既不依赖 Java, 也不依赖过时的 Hadoop Pipes.

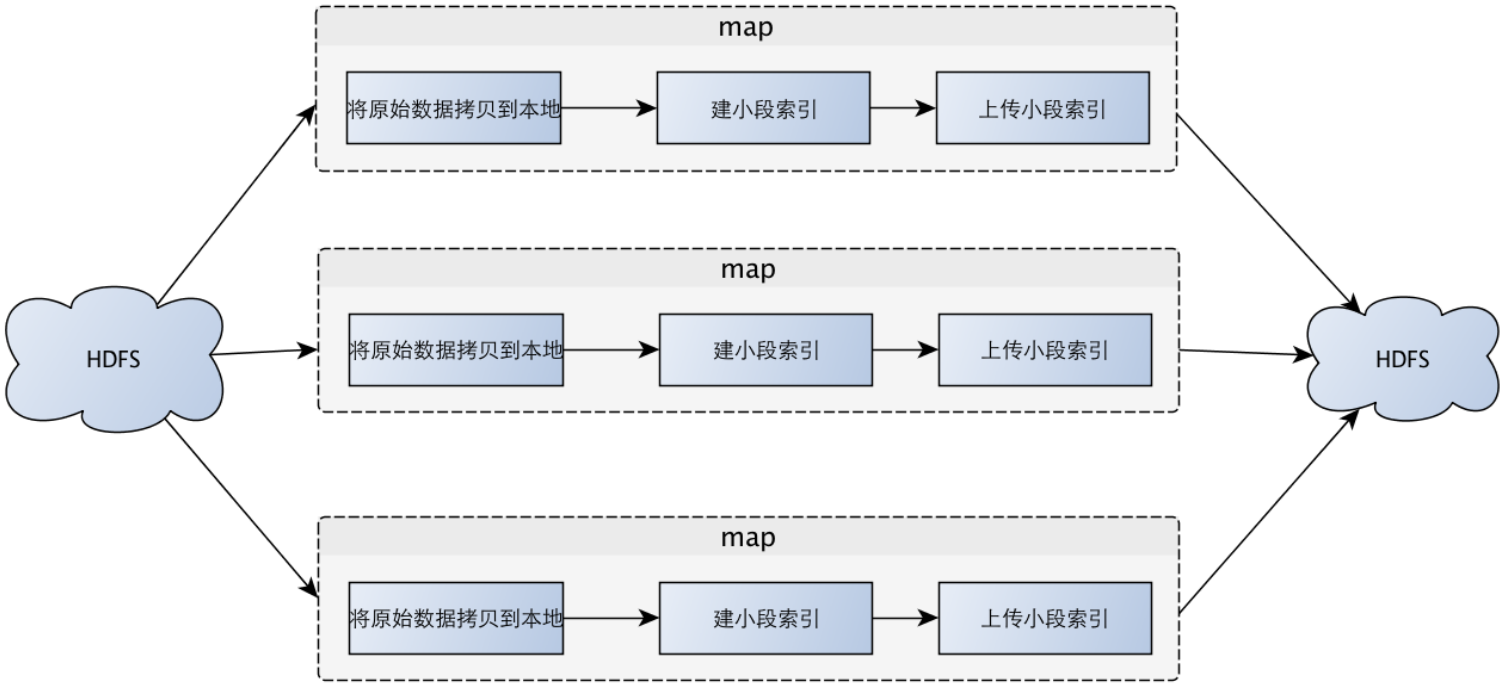
构建流程

索引构建同样分为两个过程:

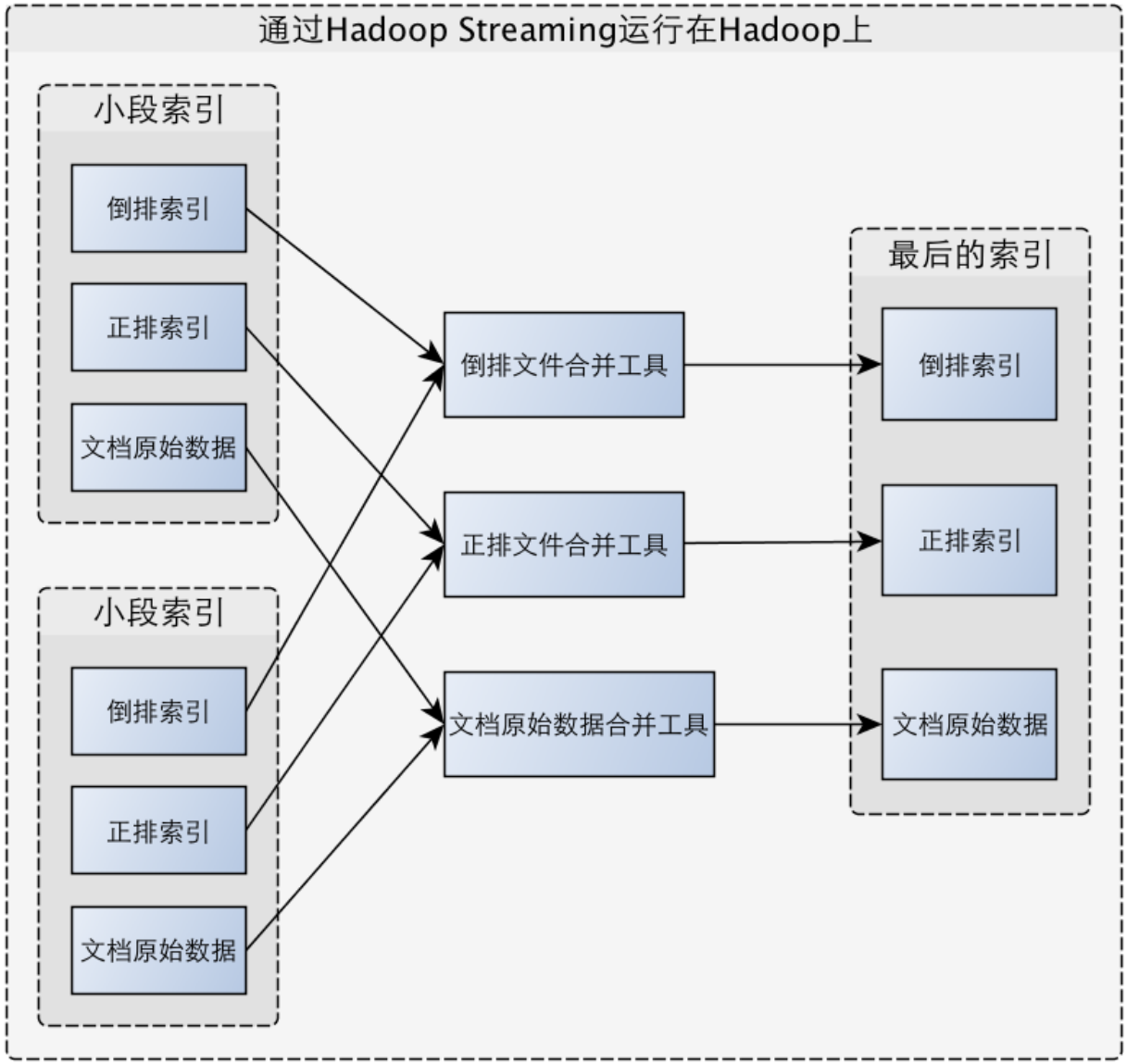
- 建小段索引
  每个 task 拖一部分 XML 数据到本地建索引, 完成之后将其放回到 HDFS 上.

- 对小段索引进行物理合并
  每个 task 拖若干个小段索引到本地进行合并.kbuild 的索引合并不同于 abuild, 它涉及到索引的物理合并, 比较耗时.

建小段索引示意图


map
将原始数据拷贝到本地
建小段索引
上传小段索引
map
将原始数据拷贝到本地
建小段索引
上传小段索引
map
将原始数据拷贝到本地
建小段索引
上传小段索引
HDFS
HDFS

合并示意图


通过Hadoop Streaming运行在Hadoop上
小段索引
倒排索引
正排索引
文档原始数据
小段索引
倒排索引
正排索引
文档原始数据
倒排文件合并工具
正排文件合并工具
文档原始数据合并工具
最后的索引
倒排索引
正排索引
文档原始数据

程序和配置文件的分发

- 使用 mapred.cache.files 来 cache 大文件.
  分词包 aliws 的数据文件超过 1GB, 使用 mapred.cache.files 将其缓存在 tasktracker, 第一次启动比较慢, 以后会非常快.
- 小文件直接使用 file 上传.
  使用 file 上传小文件不需要额外的 put 操作, 更新比较方便.

独占一个节点的资源

- 使用 capacity scheduler.
  capacity scheduler 将资源分为 queue, 支持独占一个节点的 map/reduce 资源.

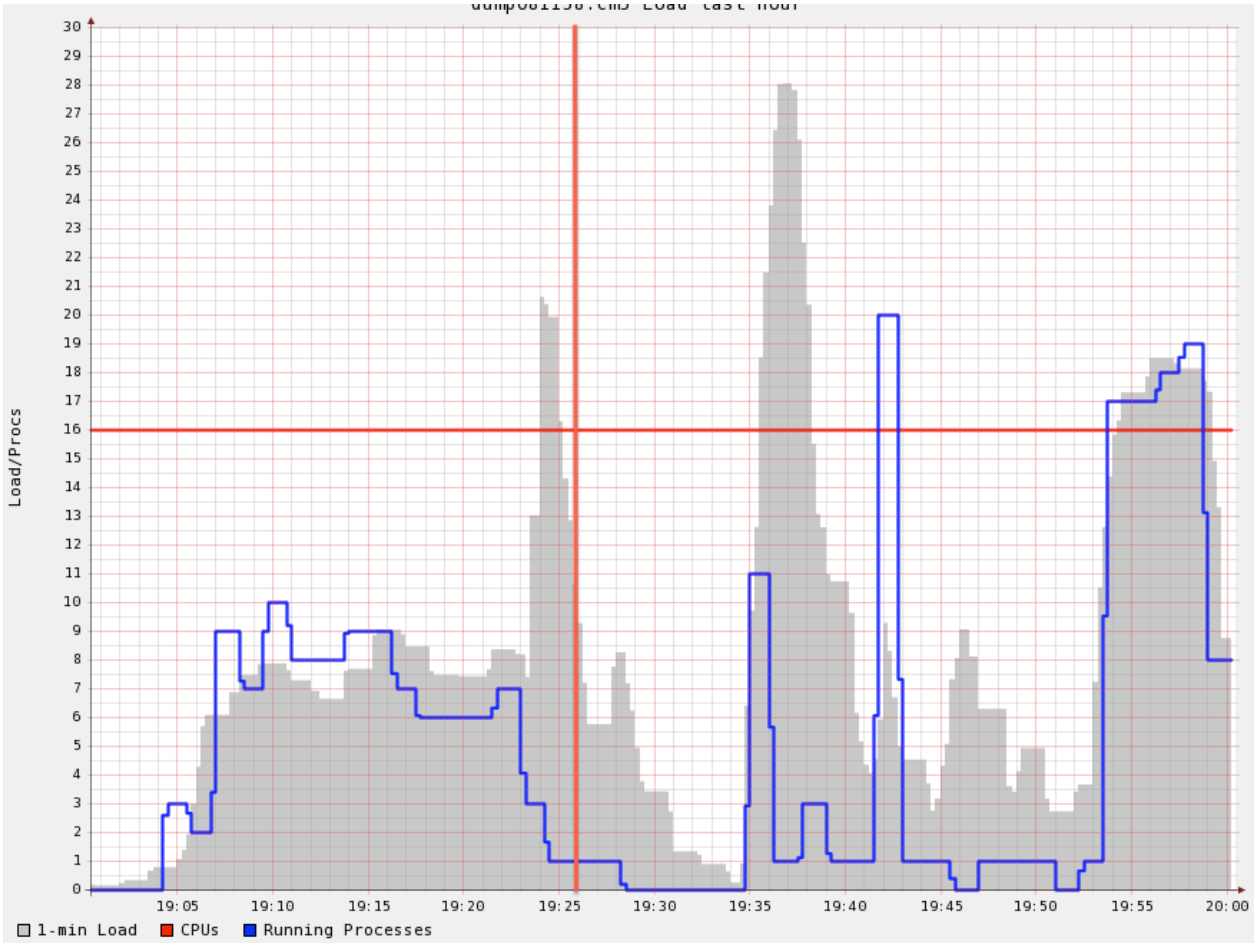
- 配置 mapred.job.map.memory.mb.
  如果一个节点配置 10 个 map 任务, 每个 map 任务可用 2GB 内存, 那么需要设置该值为 20GB, 这样就可以独占一个节点的 map 资源.

关闭推测执行

mapred.map.tasks.speculative.execution=false

由于索引构建是非典型的 Hadoop task, 所有相同的任务会往相同的目录下写数据, 导致冲突. 因此必须将该项置为 false.

性能问题


Load/Procs
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
19:05
19:10
19:15
19:20
19:25
19:30
19:35
19:40
19:45
19:50
19:55
20:00
1-min Load
CPUs
Running Processes

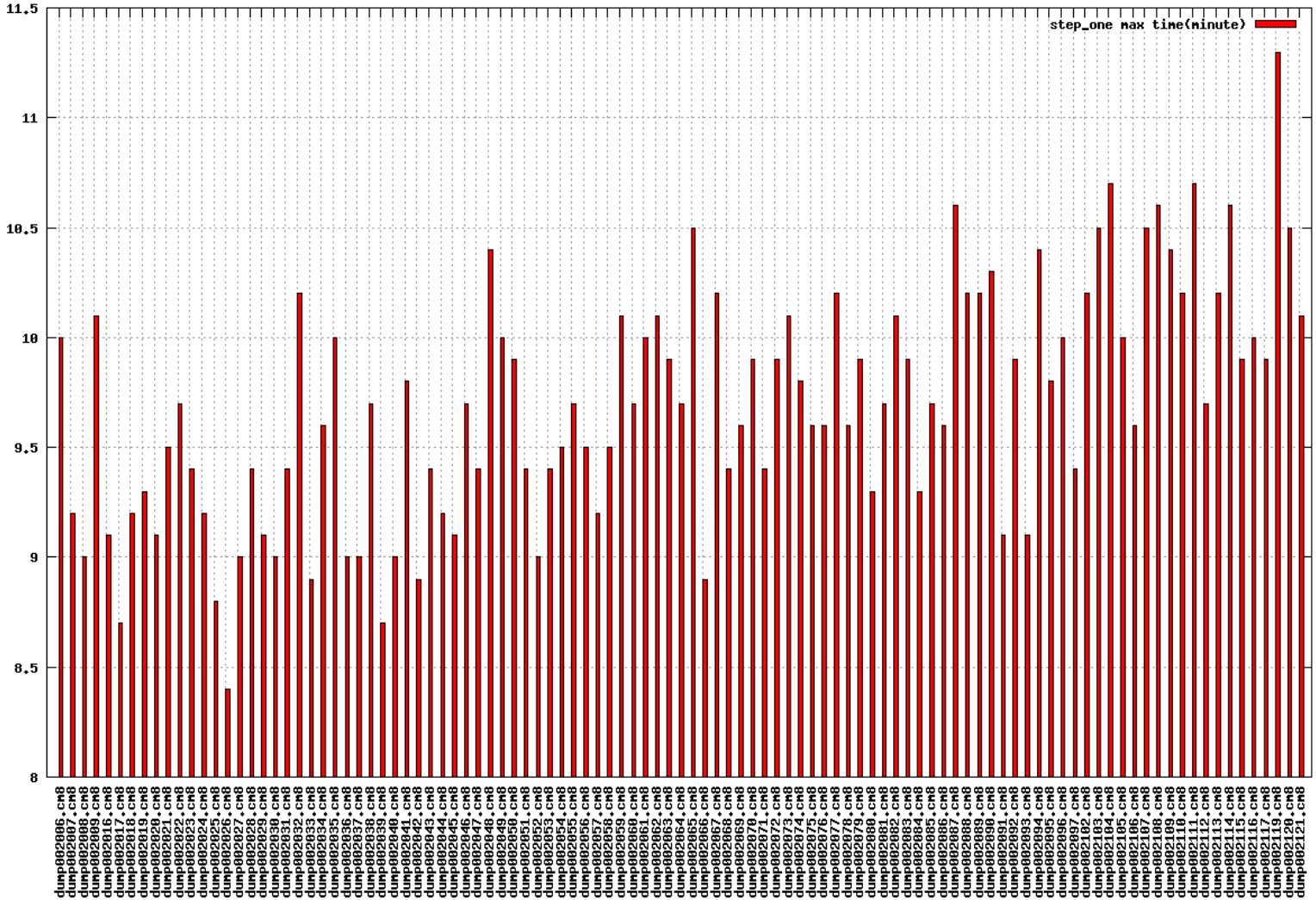
合理利用 IO 和网络资源

- 持续优化合并程序, 减少对内存的占用.
- 文件不落地，减少磁盘和 rm 的开销

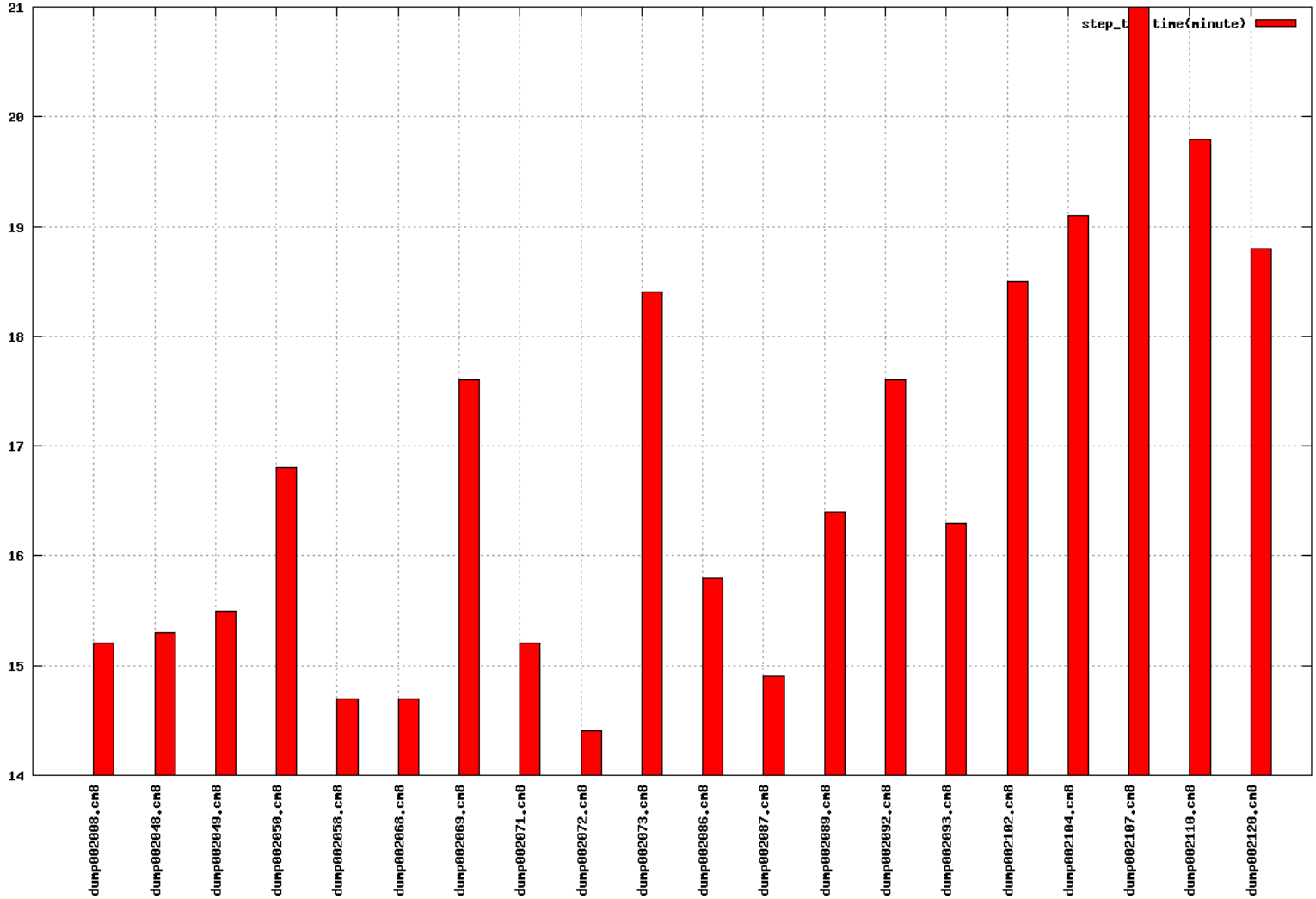
  hadoop fs -cat index.tar | tar xf - -C output
  tar -c index | hadoop fs -put - index.tar

- 分解任务, 尽量让任务并行起来.
  合并倒排索引的时候, 拖正排索引数据. 高频词截断时上传合并之后的正排索引数据.

kbuild 第一阶段耗时



kbuild 第二阶段耗时



它山之石

- 将现有程序应用在 Hadoop 上, 量大也不怕.
- 对 Java M/R, Streaming, Pipes 的抉择.
- 分发程序和配置文件的方法.
- Hadoop 集群上资源竞争很激烈, 注意效率.

The End

谢谢大家!