开源力量

Open Source Force

Nutch：从搜索引擎到网络爬虫

杨尚川
独立咨询顾问，专注于大数据和搜索引擎
2013年度优秀开源项目**APDPlat**发起人
ysc@apdplat.org
QQ: 281032878

大纲：

1、Nutch是什么

Nutch是Apache旗下的Java开源项目，最初是一个搜索引擎，现在是一个网络爬虫。下图为发起人Doug Cutting

Doug Cutting同时也是Lucene和Hadoop的发起人

Nutch的特性

插件架构，高度模块化

大多数功能都可以通过插件来实现和改变

易扩展，极强的伸缩性

增加机器即可，不用修改代码，从一台可扩展到成千上万台

高可用性，健壮容错

容忍宕机情况的出现

灵活可配置

提供了162个配置参数

Nutch的不足

- 所有文件都是只能写一次
- 批量处理架构导致无实时性
- 没有用户管理图形界面，只有命令行接口
- web2.0的普及导致的js分析和身份认证等问题

Nutch和其他项目的关系

Lucene Core（全文检索库）
Solr（企业搜索平台）
ElasticSearch（分布式的支持RESTFULL的实时搜索和实时分析）

Hadoop（分布式计算和分布式存储）
Tika（MIME类型检测、语言检测、元数据和文本自动提取）
Gora（对象到NOSQL的映射）

2、Nutch可以做什么

站内搜索引擎 Oregon State University
全网搜索引擎 即刻搜索(可能)
http://www.csdn.net/article/2010-09-13/279345?1284517620
垂直搜索引擎 Internet Archive,Creative Commons
数据采集
……

3、为什么要学习Nutch

Hadoop是大数据的核心技术之一，而Nutch集Hadoop之大成，是Hadoop的源头。学习Hadoop，没有数据怎么办？用Nutch抓！学了Hadoop的Map Reduce以及HDFS，没有实用案例怎么办？学习Nutch！Nutch的很多代码是用Map Reduce和HDFS写的，哪里还能找到比Nutch更好的Hadoop应用案例呢？

大数据这个术语最早的引用可追溯到**Nutch**。当时，大数据用来描述为更新网络搜索索引需要同时进行批量处理或分析的大量数据集。现在，大数据的含义已经被极大地发展了，业界将大数据的特性归纳为4个“V”。Volume数据体量巨大，Variety数据类型繁多，Value价值密度低，商业价值高，Velocity处理速度快。

4、Nutch的设计初衷

商业搜索引擎不开源，搜索结果不纯粹是根据网页本身的价值进行排序，而是有众多商业利益考虑。Nutch提供了开源的解决方案，帮助人们很容易地建立一个搜索引擎，为用户提供优质的搜索结果，并能从一台机器扩展到成百上千台。

5、Nutch的设计目标

并行运行在成千上万台服务器上
每个月抓取几十亿网页
为这些网页维护一个索引
对索引文件执行每秒上千次的搜索
提供高质量的搜索结果
以最小的成本运作

如何才能达成这样的目标呢？
1、需要一个分布式文件系统
2、需要一个分布式计算平台

对分布式文件系统的要求：

- 单一共享的命名空间
- 容错（磁盘及节点）
- 使用用于计算的节点上面的磁盘（计算和数据放到一起，减小网络开销）
- 不用重启系统就能增加移除磁盘及节点

Google 发布了GFS 论文

满足所有要求

一个NameNode

- 文件名和块的映射：name → <blockId>*
- 块和主机及端口的映射：blockId → <host:port>replication_level

多个DataNode

- 块和字节的映射：blockId → <byte>*
- 轮询NameNode进行复制,删除等请求

客户端同时和NameNode和DataNode进行交互

于是有了
Nutch Distributed File System(NDFS)

- GFS的JAVA开源实现
- 2003年由Mike Cafarella完成
- 作为Nutch的一部分

对分布式计算平台的要求：

把一项工作(job)分解为多个任务(task)

对于每一个任务：

- 分配主机(host)
- 启动任务
- 监控任务执行
- 宕机的时候终止任务
- 任务执行失败后能重新启动任务

顺序执行

一项工作(job)完成后执行下一项工作(job)

Google 发布了MapReduce 论文

MapReduce是一个可靠，可扩展的分布式计算平台

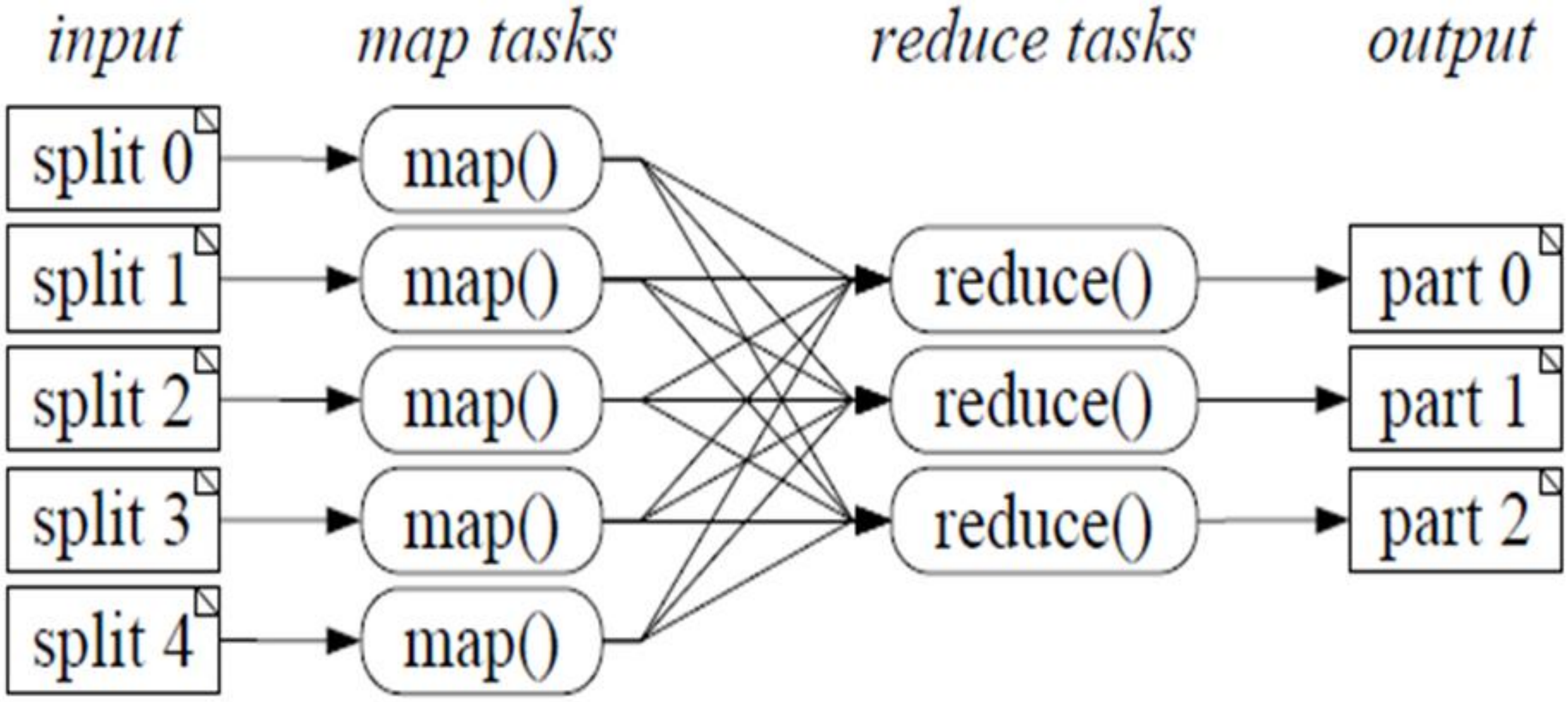
所有的数据都是顺序的<key,value>对

程序员只需要实现两个特定的方法

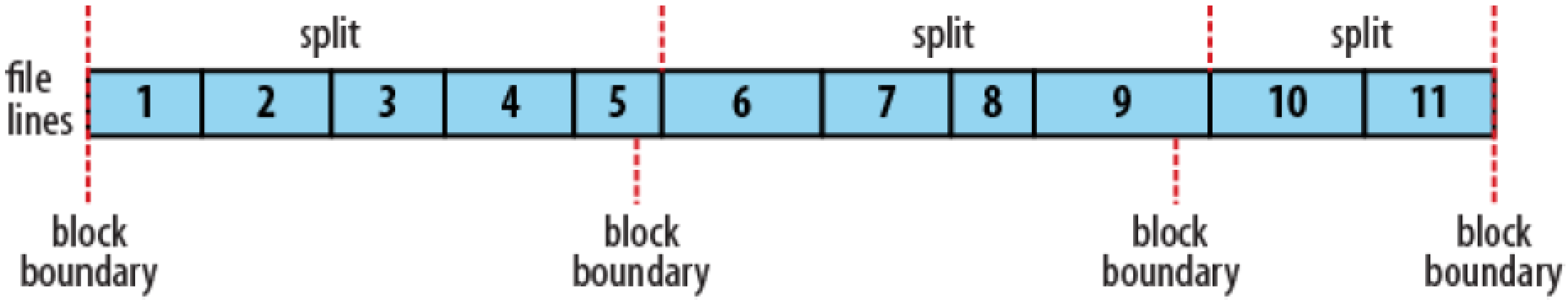
- map(k, v) → <k', v'>*
- reduce(k', <v'>*) → <k', v'>*

所有有相同k'的v'都有序地聚集在一起

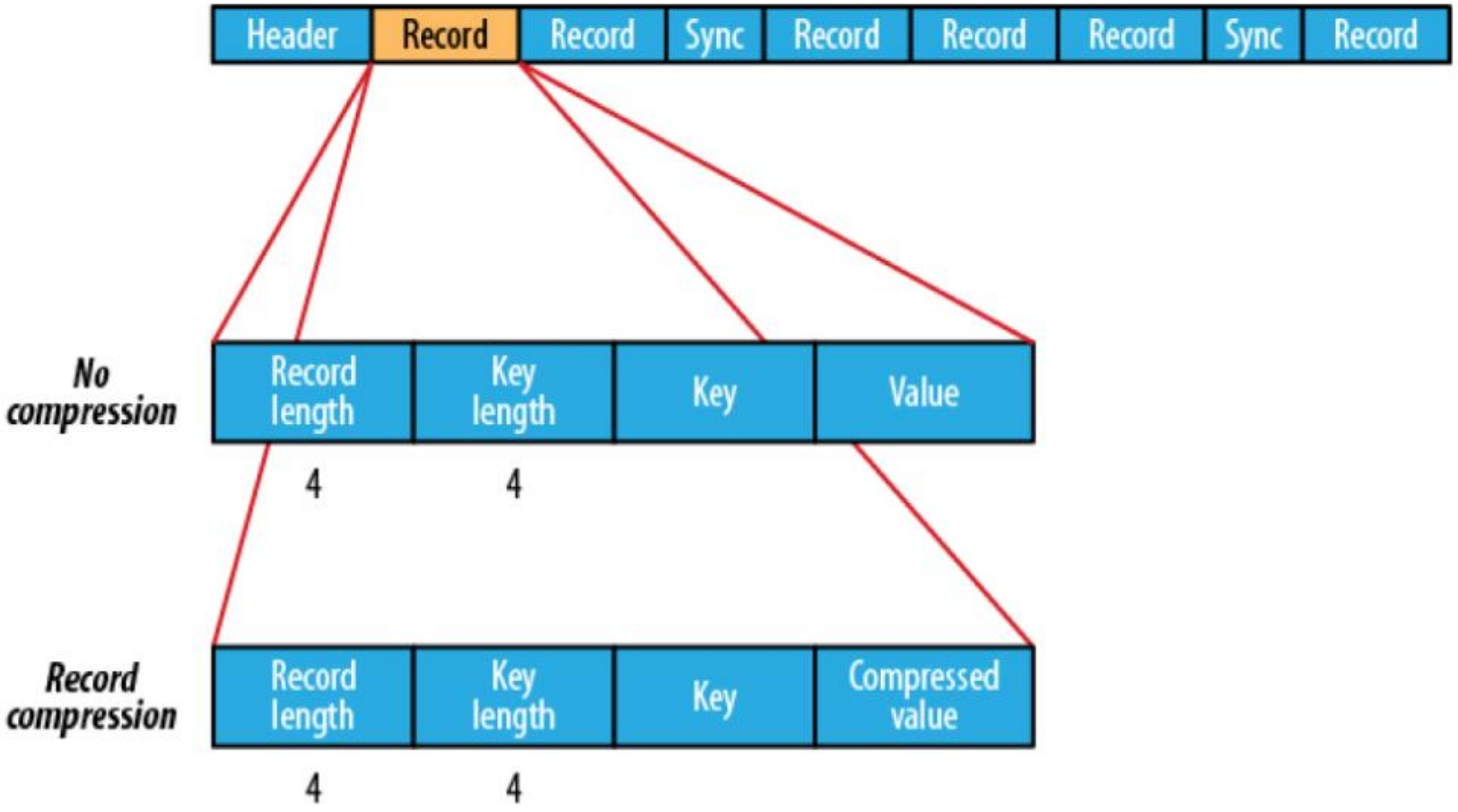
于是Nutch开始采用MapReduce

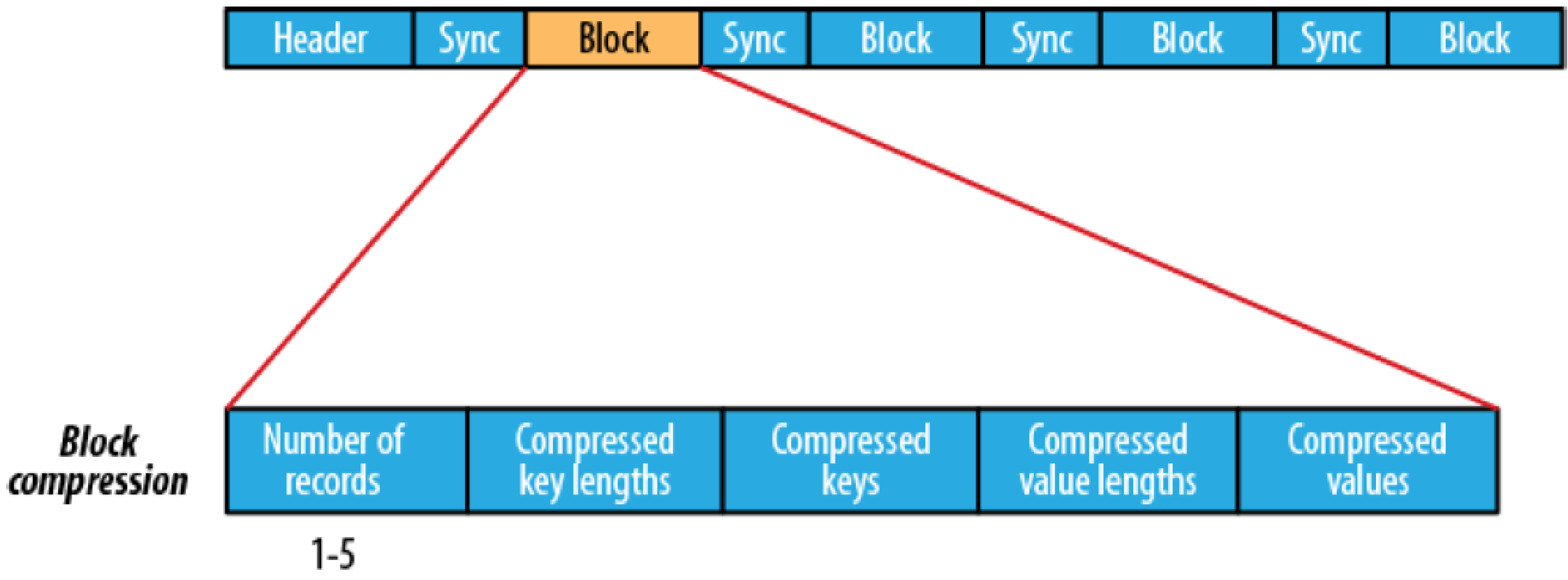

input
map tasks
reduce tasks
output
split 0
split 1
split 2
split 3
split 4
map()
map()
map()
map()
map()
reduce()
reduce()
reduce()
part 0
part 1
part 2

如何并行处理同一个大文件？



SequenceFile的结构




Header
Sync
Block
Sync
Block
Sync
Block
Sync
Block
Block compression
Number of records
Compressed key lengths
Compressed keys
Compressed value lengths
Compressed values
1-5

MapFile？

可以当成SequenceFile来使用

对KEY进行了排序和索引

6、Nutch的发展历程

2002年8月由Doug Cutting发起，托管于Sourceforge，之后发布了0.4、0.5、0.6三个版本
2004年9月Oregon State University（俄勒冈州立大学）采用Nutch
2004年9月Creative Commons（知识共享）推出基于Nutch的搜索服务
2005年1月加入Apache的孵化器
2005年6月孵化结束成为Lucene的子项目
2005年8月发布版本**0.7**（ Apache Lucene sub-project）
2005年10月发布版本**0.7.1**
2006年3月发布版本**0.7.2**
2006年7月发布版本**0.8**（全新的架构，基于Hadoop 0.4， **Hadoop诞生**）
2006年9月发布版本**0.8.1**
2007年4月发布版本**0.9**
2009年3月发布版本**1.0**（**Tika诞生，**0.1-incubating）
2010年4月Nutch成为Apache顶级项目
2010年6月发布版本**1.1**
2010年9月发布版本**1.2**
2011年6月发布版本**1.3**（ **从搜索引擎到网络爬虫**）

2011年11月发布版本**1.4**
2012年6月发布版本**1.5**
2012年7月发布版本**2.0**（ 2.X **Gora诞生，** table-based architecture ）
2012年7月发布版本**1.5.1**
2012年8月**Nutch诞生十周年**
2012年10月发布版本**2.1**（ 2.X开始支持elastic search ）
2012年12月发布版本**1.6**
2013年6月发布版本**2.2**（**crawler-commons 诞生**）
2013年6月发布版本**1.7**（**crawler-commons 诞生**）
2013年7月发布版本**2.2.1**

11年发展历程，3大分支版本
强调重用，诞生了Java开源项目Hadoop、Tika、Gora
不重新发明轮子，使用了大量第三方开源项目

Nutch1.3是从搜索引擎到网络爬虫的转折点

- Removed Lucene-based indexing and search webapp
  - delegate indexing / search remotely to SOLR
  - change of focus : “Web search application” → “Crawler”
- Removed deprecated parse plugins
  - delegate most parsing to Tika
- Separate local / distributed runtimes
- Ivy-based dependency management

Release 0.4

1. Http class refactored. (Kevin Smith via Tom Pierce)
2. Add Finnish translation. (Sampo Syreeni via Doug Cutting)
3. Added Japanese translation. (Yukio Andoh via Doug Cutting)
4. Updated Dutch translation. (Ype Kingma via Doug Cutting)
5. Initial version of Distributed DB code. (Mike Cafarella)
6. Make things more tolerant of crashed fetcher output files. (Doug Cutting)
7. New skin for website. (Frank Henze via Doug Cutting)
8. Added Spanish translation. (Diego Basch via Doug Cutting)
9. Add FTP support to fetcher. (John Xing via Doug Cutting)
10. Added Thai translation. (Pichai Ongvasith via Doug Cutting)
11. Added Robots.txt & throttling support to Fetcher.java. (Mike Cafarella)
12. Added nightly build. (Doug Cutting)
13. Default all link scores to 1.0. (Doug Cutting)
14. Permit one to keep internal links. (Doug Cutting)
15. Fixed dedup to select shortest URL. (Doug Cutting)

16. Changed index merger so that merged index is written to named directory, rather than to a generated name in that directory. (Doug Cutting)

17. Disable coordination weighting of query clauses and other minor scoring improvements. (Doug Cutting)

18. Added a new command, crawl, that constructs a database, injects a url file and performs a few rounds of generate/fetch/updatedb. This simplifies use for intranet sites. Changed some defaults to be more intranet friendly. (Doug Cutting)

19. Fixed a bug where Fetcher.java didn't construct correct relative links when a page was redirected. (Doug Cutting)

20. Fixed a query parser problem with lookahead over plusses and minuses. (Doug Cutting)

21. Add support for HTTP proxy servers. (Sami Siren via Doug Cutting)

22. Permit searching while fetching and/or indexing. (Sami Siren via Doug Cutting)

23. Fix a bug when throttling is disabled. (Sami Siren via Doug Cutting)

24. Updated Bahasa Malaysia translation. (Michael Lim via Doug Cutting)

25. Added Catalan translation. (Xavier Guardiola via Doug Cutting)

```
26. Added brazilian portuguese translation.
    (A. Moreir via Doug Cutting)

27. Added a french translation.  (Julien Nioche via Doug Cutting)

28. Updated to Lucene 1.4RC3.  (Doug Cutting)

29. Add capability to boost by link count & use it in crawl tool.
    (Doug Cutting)

30. Added plugin system.  (Stefan Groschupf via Doug Cutting)

31. Add this change log file, for recording significant changes to
    Nutch.  Populate it with changes from the last few months.
```

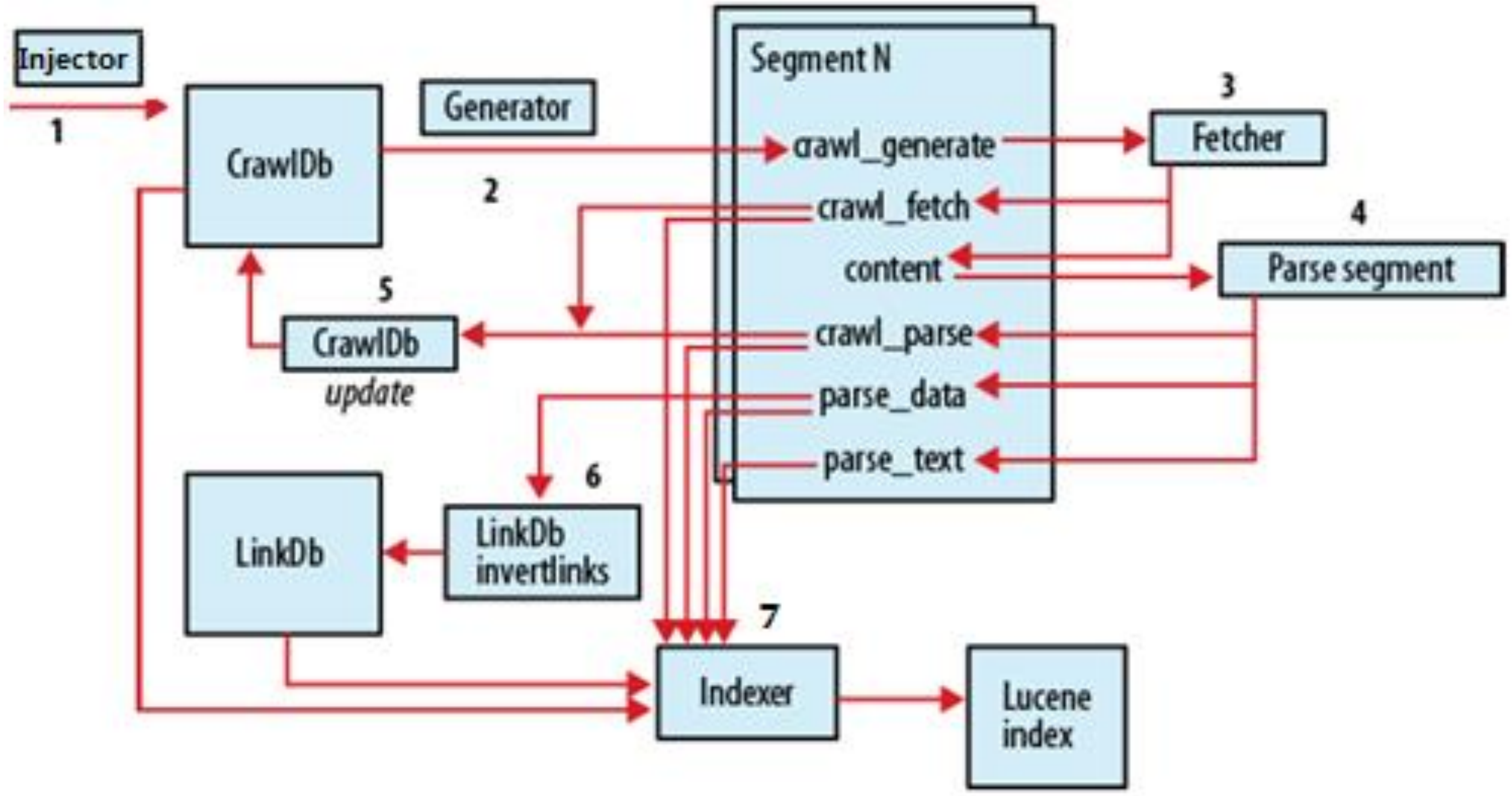
```
24. Added new config parameter fetcher.threads.per.host.  This is used
    by the Http protocol.  When this is one behavior is as before.
    When this is greater than one then multiple threads are permitted
    to access a host at once.  Note that fetcher.server.delay is no
    longer consistently observed when this is greater than one.
    (Luke Baker via Doug Cutting)
```

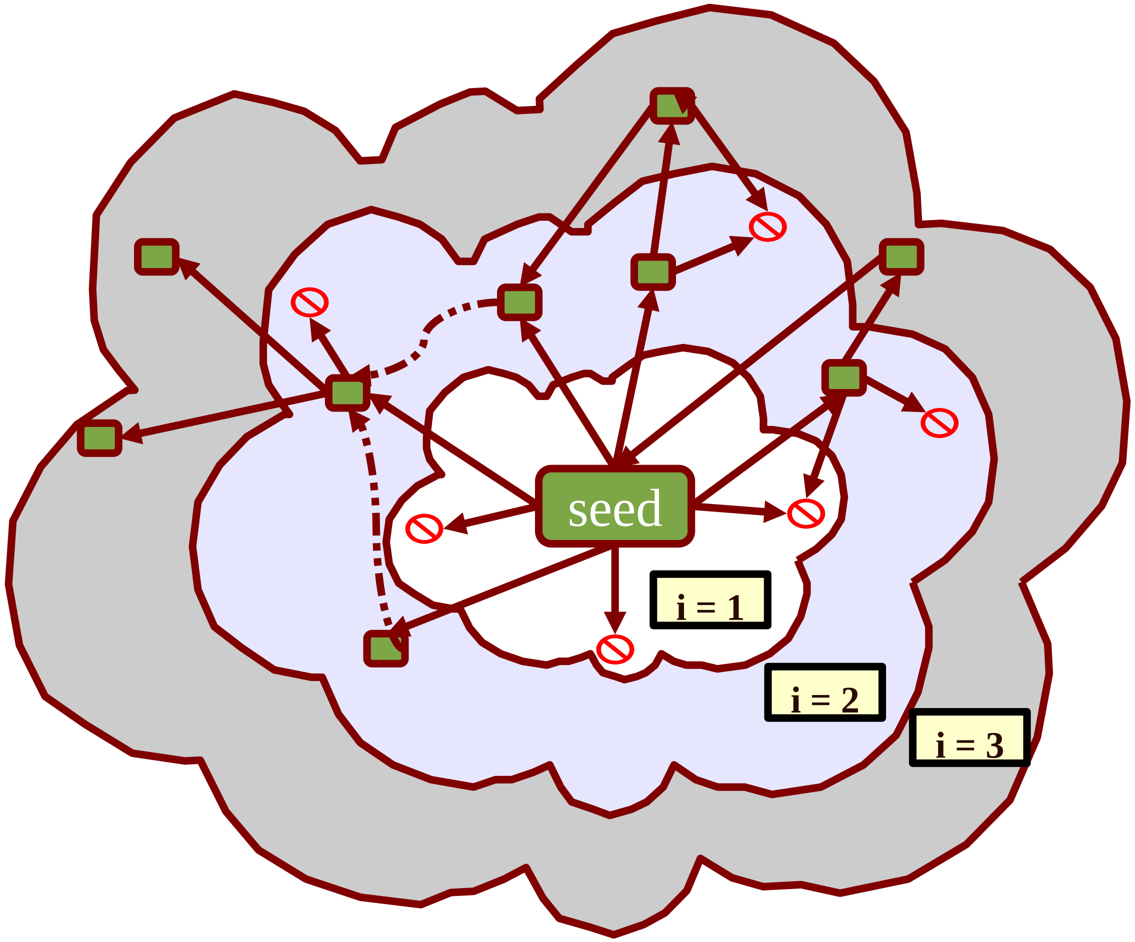
```
Release 0.7 - 2005-08-17

 1. Added support for "type:" in queries. Search results are limited/qualified
    by mimetype or its primary type or sub type. For example,
    (1) searching with "type:application/pdf" restricts results
    to pages which were identified to be of mimetype "application/pdf".
    (2) with "type:application", nutch will return pages of
    primary type "application".
    (3) with "type:pdf", only pages of sub type "pdf" will be listed.
    (John Xing, 20050120)

 2. Added support for "date:" in queries. Last-Modified is indexed.
    Search results are restricted by lower and upper date (inclusive)
    as date:yyyymmdd-yyyymmdd. For example, date:20040101-20041231
    only returns pages with Last-Modified in year 2004.
    (John Xing, 20050122)

 3. Add URLFilter plugin interface and convert existing url filters into
    plugins. (John Xing, 20050206)

 4. Add UpdateSegmentsFromDb tool, which updates the scores and
    anchors of existing segments with the current values in the web
    db.  This is used by CrawlTool, so that pages are now only fetched
    once per crawl.  (Doug Cutting, 20050221)

 5. Moved code into org.apache.nutch sub-packages.  Changed license to
    Apache 2.0.  Removed jar files whose licenses do not permit
    redistribution by Apache.  Disabled compilation of plugins which
    require these libraries.  (Doug Cutting 20050301)
```

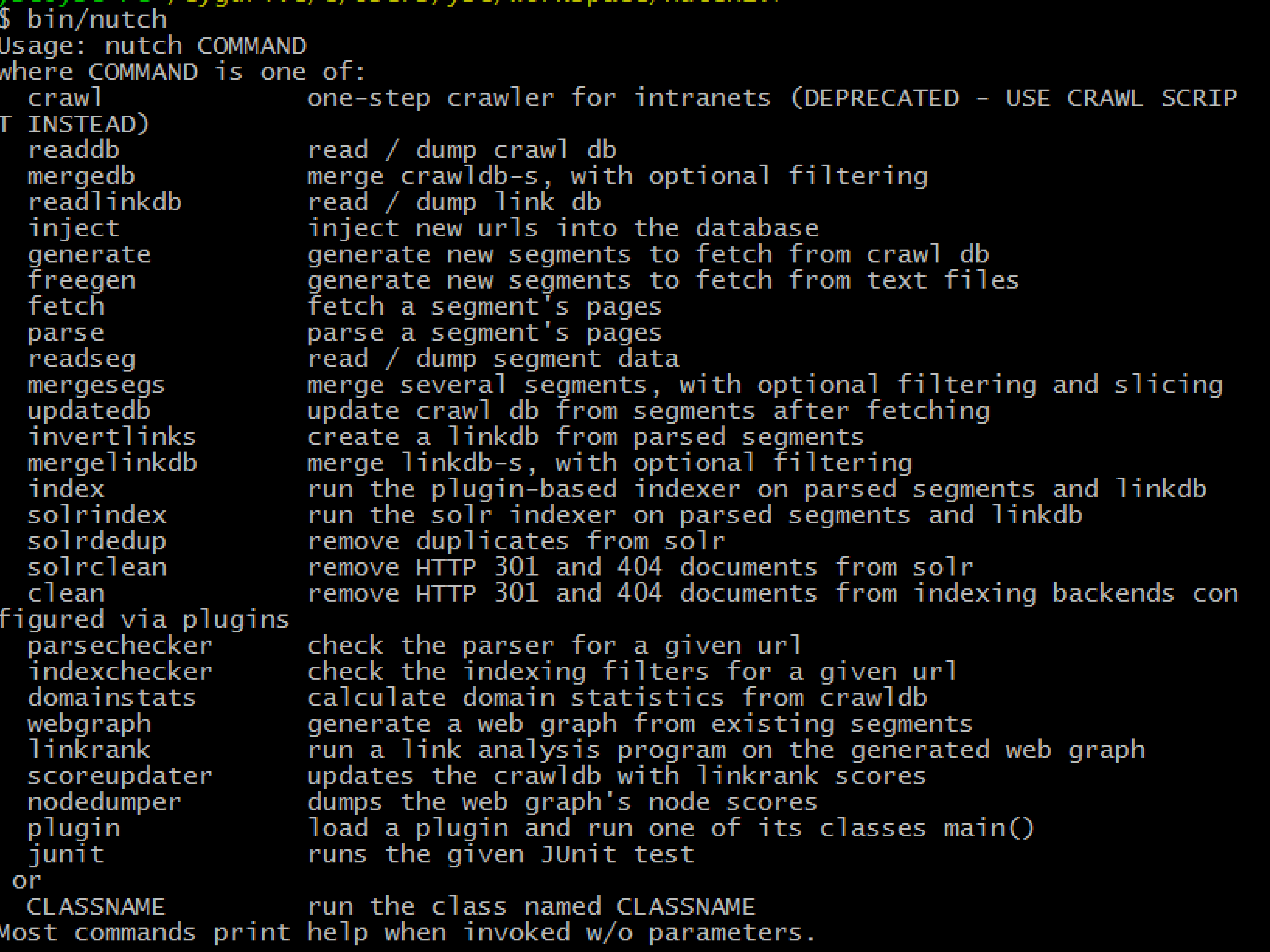
7、Nutch 3大分支版本

Nutch1.2是一个完整的搜索引擎
Nutch1.7是一个基于HDFS的网络爬虫
Nutch2.2.1是一个基于Gora的网络爬虫
1.X系列可用于生产环境、2.X系列还不成熟

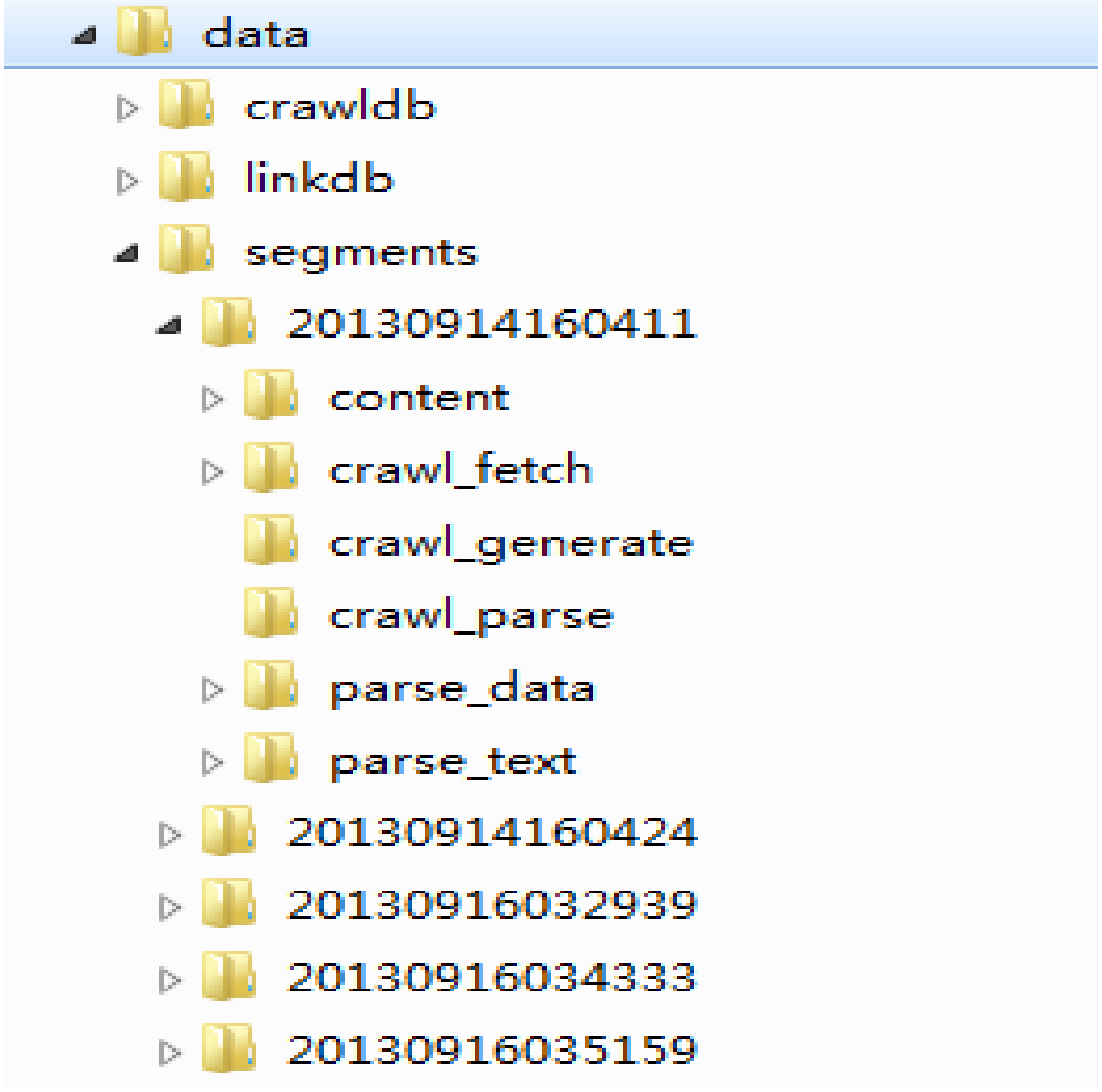
8、Nutch的整体架构



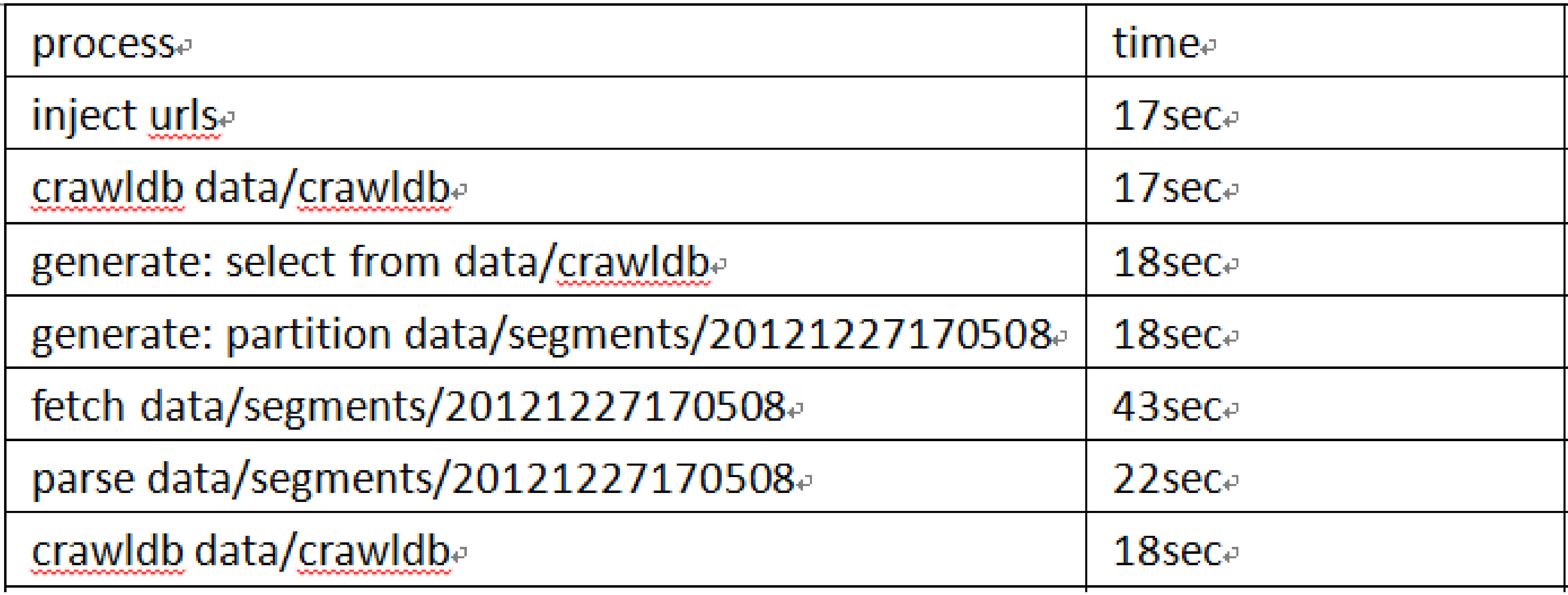

seed
i = 1
i = 2
i = 3



```
$ bin/nutch
Usage: nutch COMMAND
where COMMAND is one of:
  crawl             one-step crawler for intranets (DEPRECATED - USE CRAWL SCRIP
T INSTEAD)
  readdb            read / dump crawl db
  mergedb           merge crawldb-s, with optional filtering
  readlinkdb        read / dump link db
  inject            inject new urls into the database
  generate          generate new segments to fetch from crawl db
  freegen           generate new segments to fetch from text files
  fetch             fetch a segment's pages
  parse             parse a segment's pages
  readseg           read / dump segment data
  mergesegs         merge several segments, with optional filtering and slicing
  updatedb          update crawl db from segments after fetching
  invertlinks       create a linkdb from parsed segments
  mergelinkdb       merge linkdb-s, with optional filtering
  index             run the plugin-based indexer on parsed segments and linkdb
  solrindex         run the solr indexer on parsed segments and linkdb
  solrdedup         remove duplicates from solr
  solrclean         remove HTTP 301 and 404 documents from solr
  clean             remove HTTP 301 and 404 documents from indexing backends con
figured via plugins
  parsechecker      check the parser for a given url
  indexchecker      check the indexing filters for a given url
  domainstats       calculate domain statistics from crawldb
  webgraph          generate a web graph from existing segments
  linkrank          run a link analysis program on the generated web graph
  scoreupdater      updates the crawldb with linkrank scores
  nodedumper        dumps the web graph's node scores
  plugin            load a plugin and run one of its classes main()
  junit             runs the given JUnit test
 or
  CLASSNAME         run the class named CLASSNAME
Most commands print help when invoked w/o parameters.
```

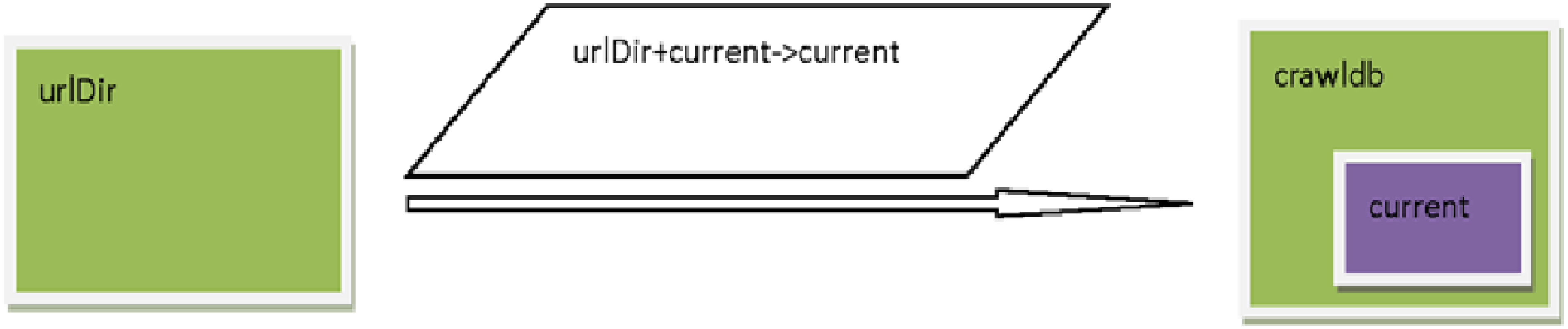


```
data
    crawldb
    linkdb
    segments
        20130914160411
            content
            crawl_fetch
            crawl_generate
            crawl_parse
            parse_data
            parse_text
        20130914160424
        20130916032939
        20130916034333
        20130916035159
```

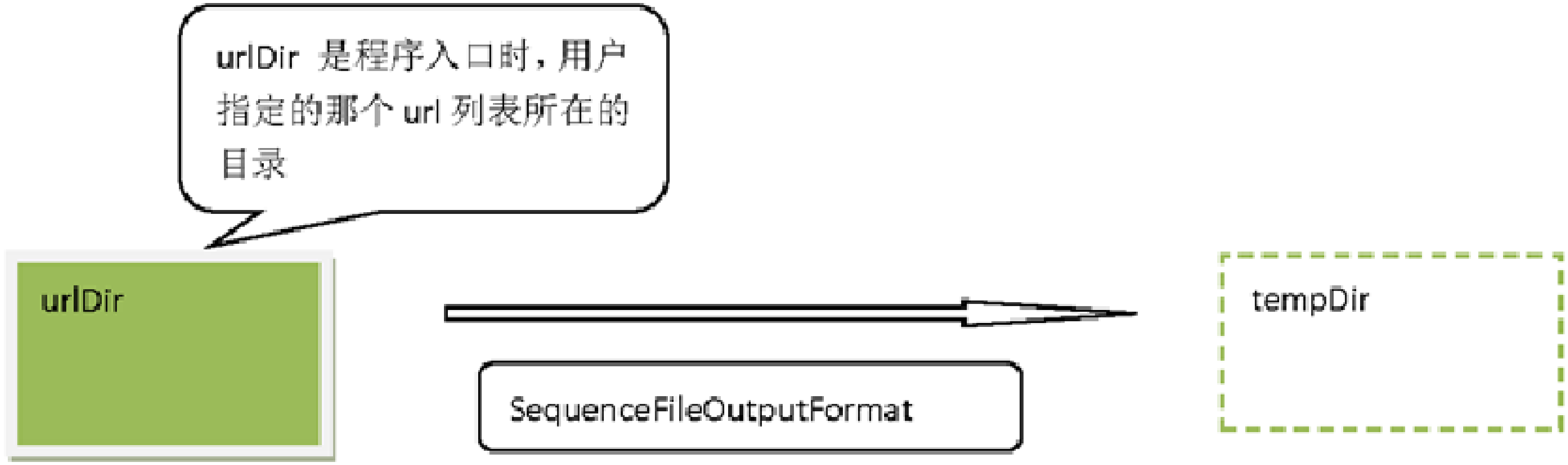


process	time
inject urls	17sec
crawldb data/crawldb	17sec
generate: select from data/crawldb	18sec
generate: partition data/segments/20121227170508	18sec
fetch data/segments/20121227170508	43sec
parse data/segments/20121227170508	22sec
crawldb data/crawldb	18sec

8.1 Injector(注入)

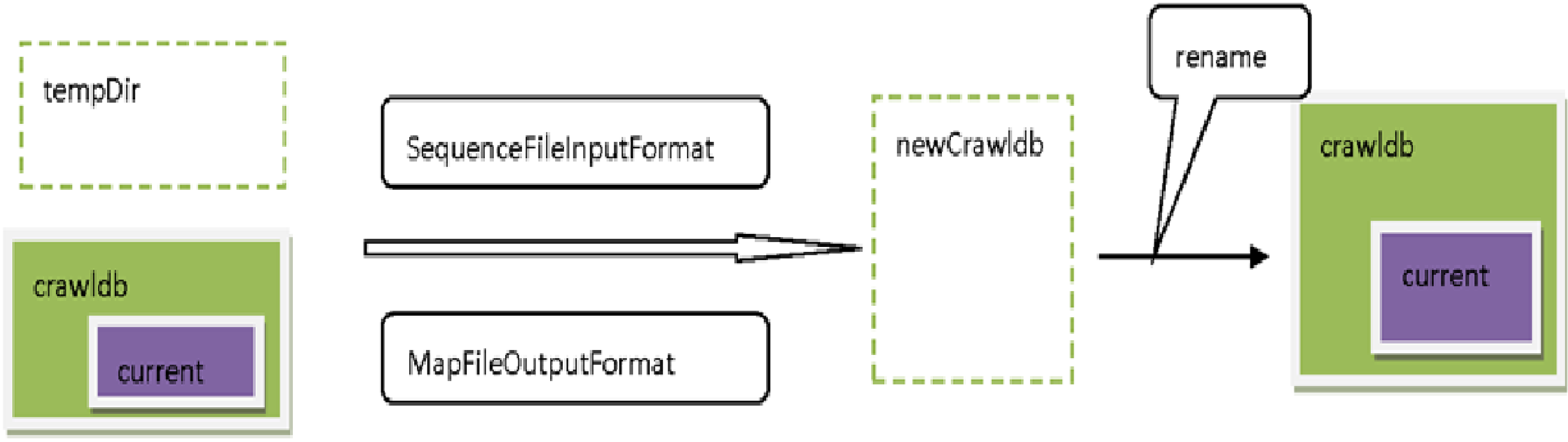


第一步：

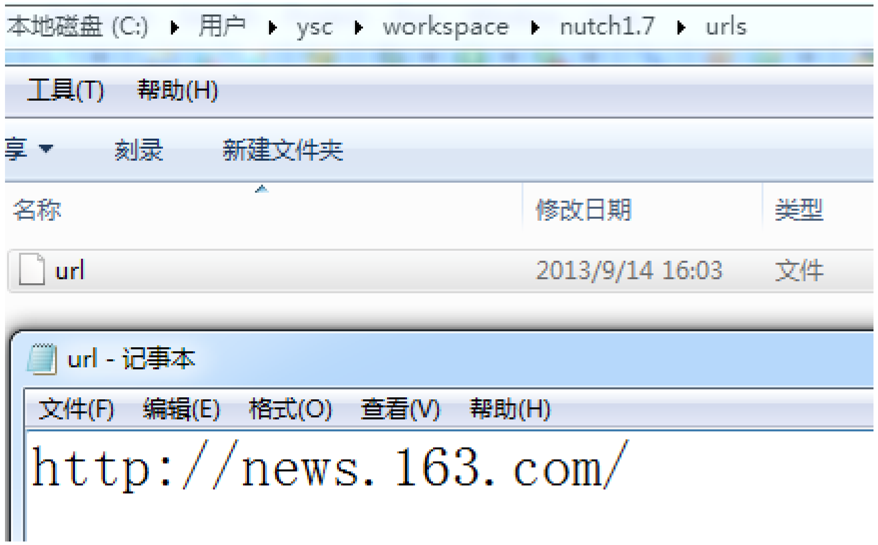


```
JobConf sortJob = new NutchJob(getConf());
sortJob.setJobName("inject " + urlDir);
FileInputFormat.addInputPath(sortJob, urlDir);
sortJob.setMapperClass(InjectMapper.class);

FileOutputFormat.setOutputPath(sortJob, tempDir);
sortJob.setOutputFormat(SequenceFileOutputFormat.class);
sortJob.setOutputKeyClass(Text.class);
sortJob.setOutputValueClass(CrawlDatum.class);
sortJob.setLong("injector.current.time", System.currentTimeMillis());
RunningJob mapJob = JobClient.runJob(sortJob);
```

第二步:



```
JobConf mergeJob = CrawlDb.createJob(getConf(), crawlDb);
FileInputFormat.addInputPath(mergeJob, tempDir);
mergeJob.setReducerClass(InjectReducer.class);
JobClient.runJob(mergeJob);
CrawlDb.install(mergeJob, crawlDb);

// clean up
FileSystem fs = FileSystem.get(getConf());
fs.delete(tempDir, true);
```

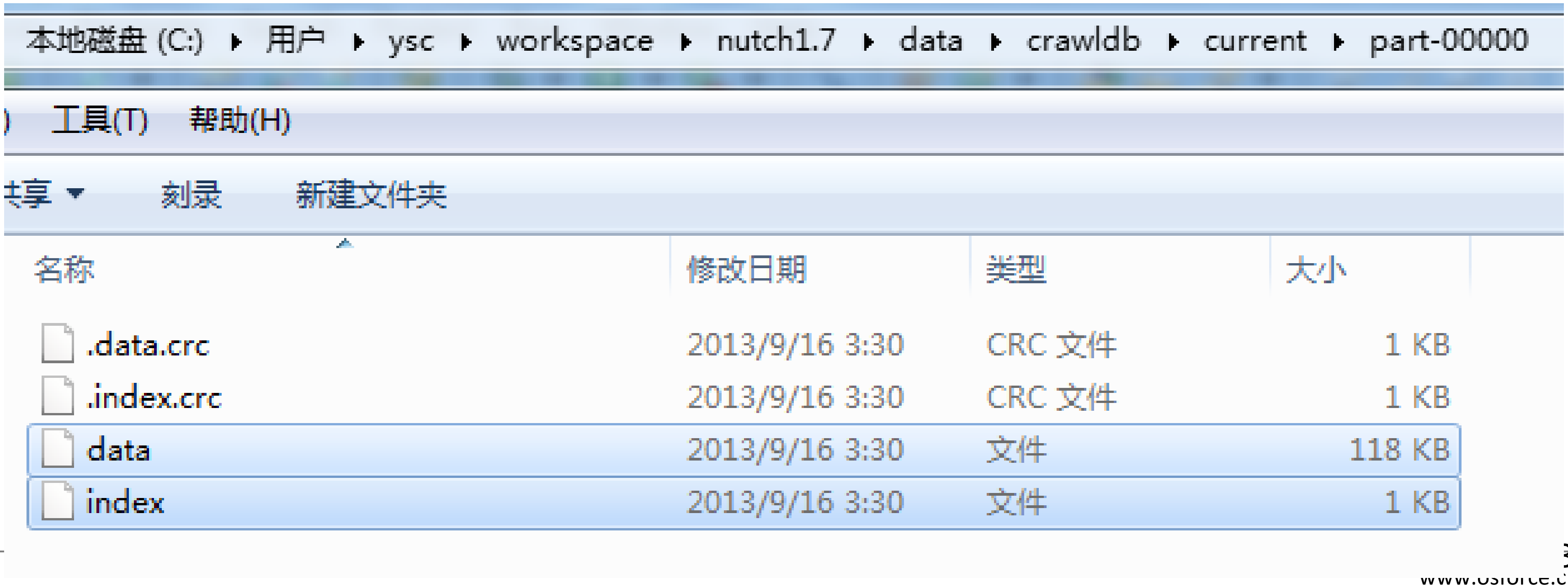


本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ urls

名称	修改日期	类型
url	2013/9/14 16:03	文件

url - 记事本

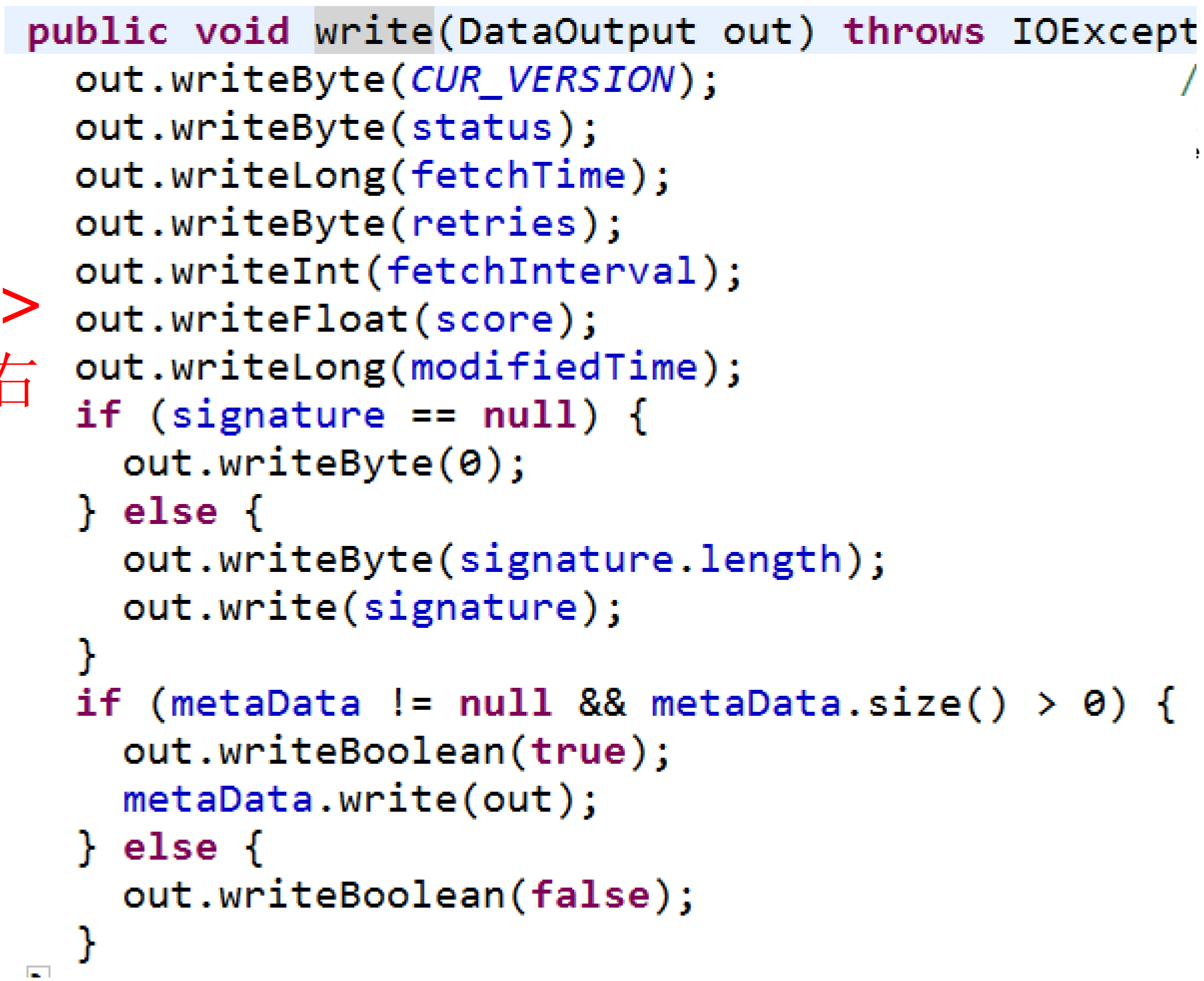
```
http://news.163.com/
```



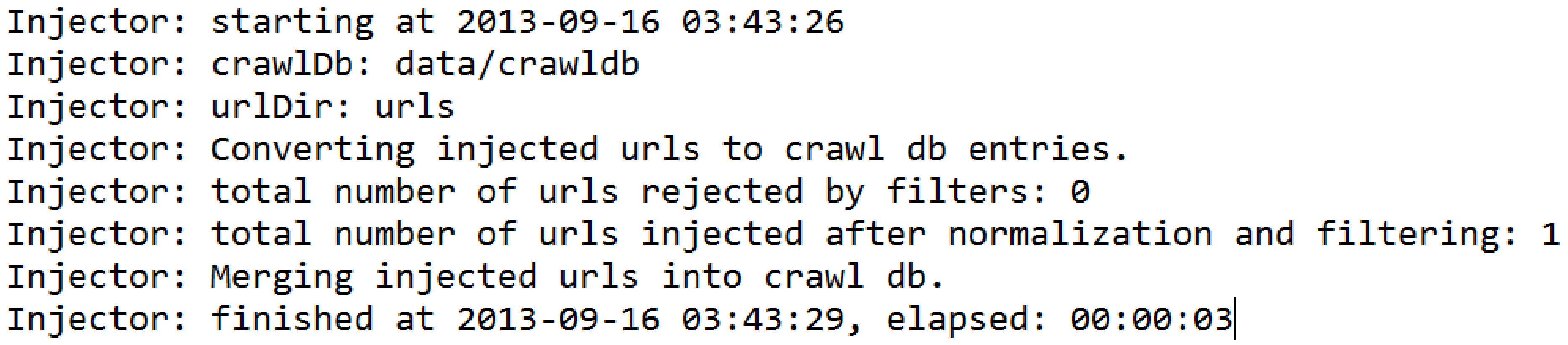
本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ crawldb ▸ current ▸ part-00000

名称	修改日期	类型	大小
.data.crc	2013/9/16 3:30	CRC 文件	1 KB
.index.crc	2013/9/16 3:30	CRC 文件	1 KB
data	2013/9/16 3:30	文件	118 KB
index	2013/9/16 3:30	文件	1 KB

CrawlDB是MapFile
<Text,CrawlDatum>
CrawlDatum结构如右图所示



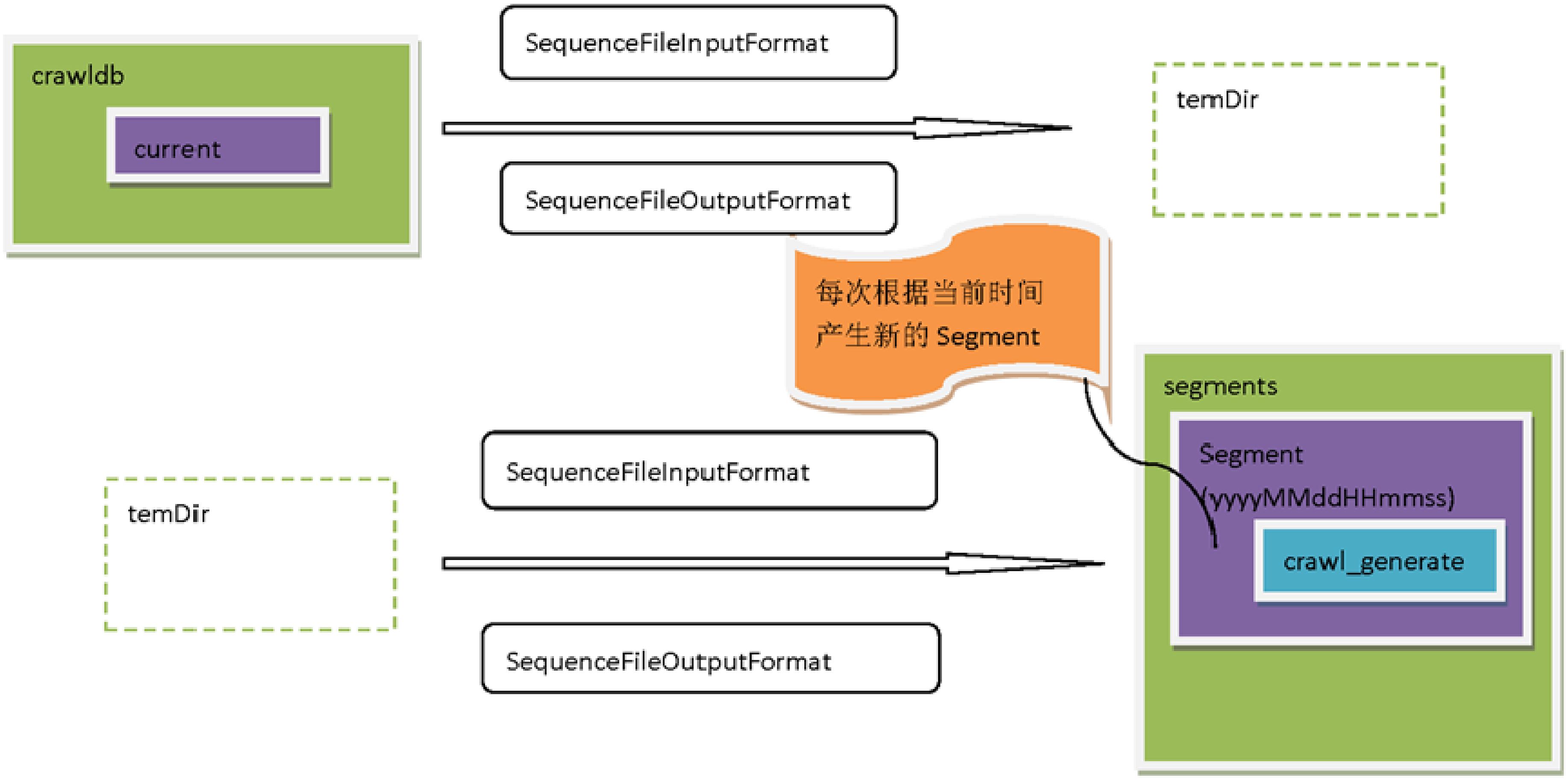
```
public void write(DataOutput out) throws IOExcept
  out.writeByte(CUR_VERSION);                    /
  out.writeByte(status);
  out.writeLong(fetchTime);
  out.writeByte(retries);
  out.writeInt(fetchInterval);
  out.writeFloat(score);
  out.writeLong(modifiedTime);
  if (signature == null) {
    out.writeByte(0);
  } else {
    out.writeByte(signature.length);
    out.write(signature);
  }
  if (metaData != null && metaData.size() > 0) {
    out.writeBoolean(true);
    metaData.write(out);
  } else {
    out.writeBoolean(false);
  }
```

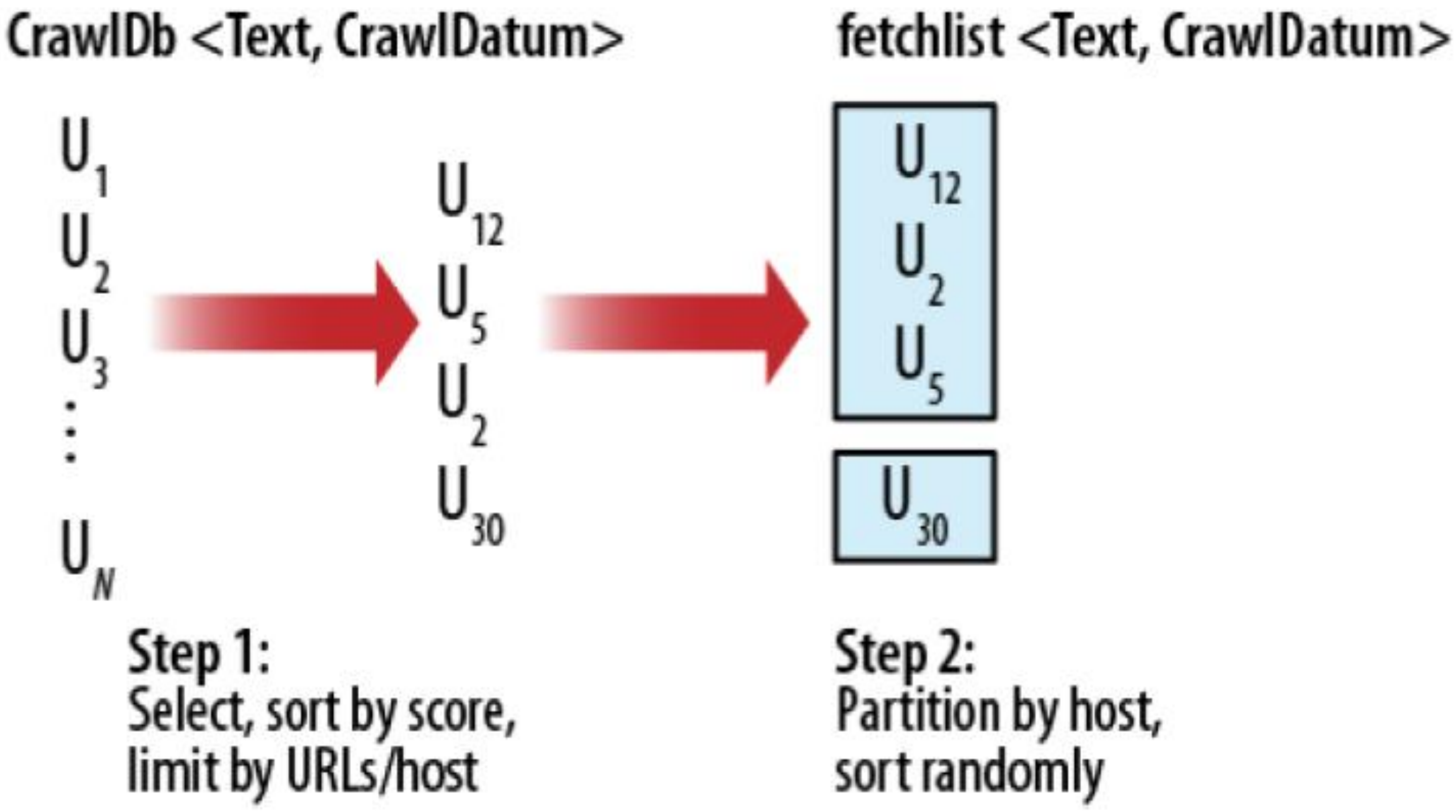


```
Injector: starting at 2013-09-16 03:43:26
Injector: crawlDb: data/crawldb
Injector: urlDir: urls
Injector: Converting injected urls to crawl db entries.
Injector: total number of urls rejected by filters: 0
Injector: total number of urls injected after normalization and filtering: 1
Injector: Merging injected urls into crawl db.
Injector: finished at 2013-09-16 03:43:29, elapsed: 00:00:03
```

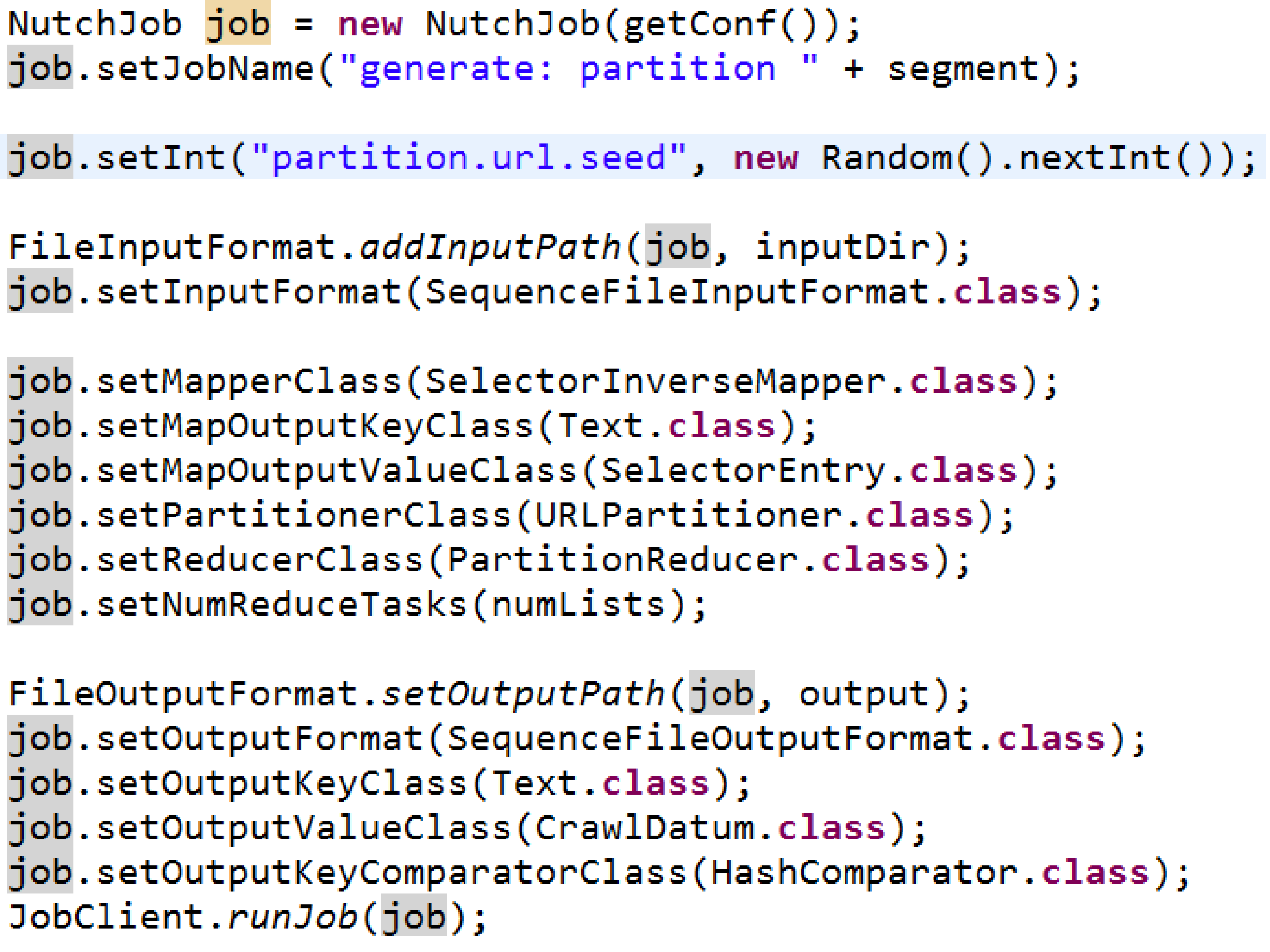
8.2 Generator(产生抓取列表)

generator.generate(crawlDb, segments, -1, topN, System.currentTimeMillis());

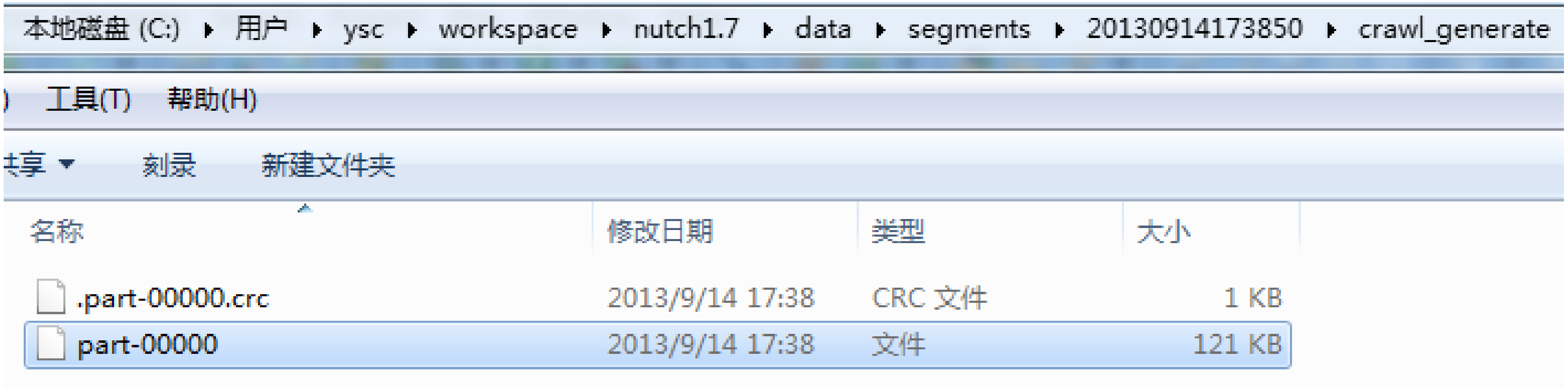



CrawlDb <Text, CrawlDatum>
fetchlist <Text, CrawlDatum>
U1
U2
U3
⋮
UN
U12
U5
U2
U30
U12
U2
U5
U30
Step 1:
Select, sort by score,
limit by URLs/host
Step 2:
Partition by host,
sort randomly

```
// map to inverted subset due for fetch, sort by score
JobConf job = new NutchJob(getConf());
job.setJobName("generate: select from " + dbDir);

if (numLists == -1) { // for politeness make
  numLists = job.getNumMapTasks(); // a partition per fetch task
}
if ("local".equals(job.get("mapred.job.tracker")) && numLists != 1) {
  // override
  LOG.info("Generator: jobtracker is 'local', generating exactly one partition.");
  numLists = 1;
}
job.setLong(GENERATOR_CUR_TIME, curTime);
// record real generation time
long generateTime = System.currentTimeMillis();
job.setLong(Nutch.GENERATE_TIME_KEY, generateTime);
job.setLong(GENERATOR_TOP_N, topN);
job.setBoolean(GENERATOR_FILTER, filter);
job.setBoolean(GENERATOR_NORMALISE, norm);
job.setInt(GENERATOR_MAX_NUM_SEGMENTS, maxNumSegments);
FileInputFormat.addInputPath(job, new Path(dbDir, CrawlDb.CURRENT_NAME));
job.setInputFormat(SequenceFileInputFormat.class);
job.setMapperClass(Selector.class);
job.setPartitionerClass(Selector.class);
job.setReducerClass(Selector.class);
FileOutputFormat.setOutputPath(job, tempDir);
job.setOutputFormat(SequenceFileOutputFormat.class);
job.setOutputKeyClass(FloatWritable.class);
job.setOutputKeyComparatorClass(DecreasingFloatComparator.class);
job.setOutputValueClass(SelectorEntry.class);
job.setOutputFormat(GeneratorOutputFormat.class);
try {
  JobClient.runJob(job);
} catch (IOException e) {
  throw e;
}
```



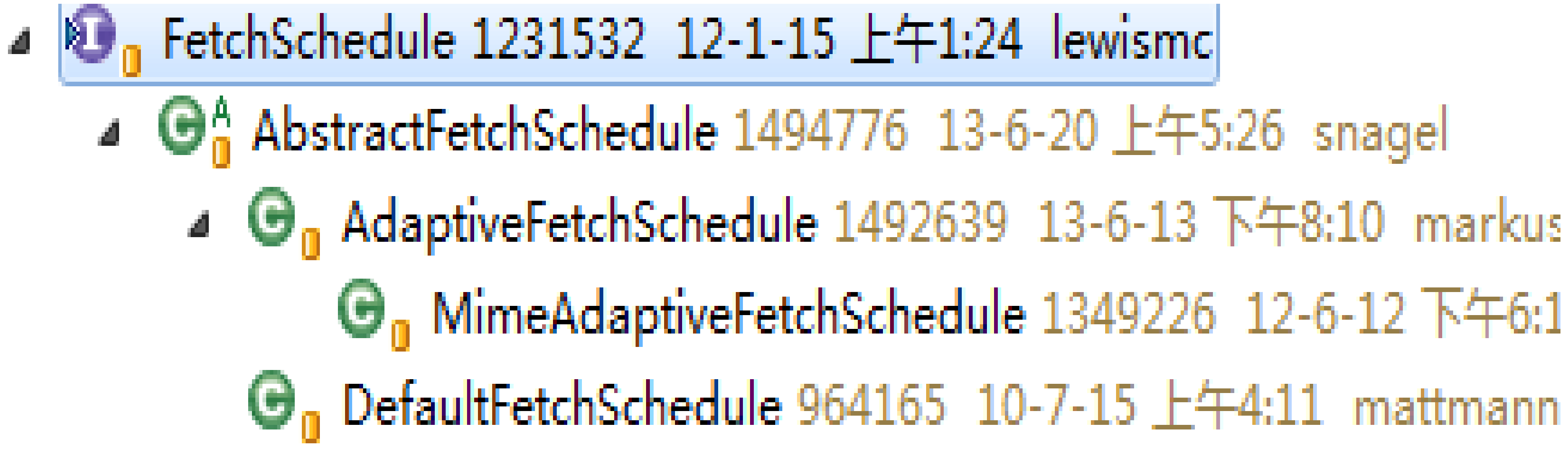
```
NutchJob job = new NutchJob(getConf());
job.setJobName("generate: partition " + segment);

job.setInt("partition.url.seed", new Random().nextInt());

FileInputFormat.addInputPath(job, inputDir);
job.setInputFormat(SequenceFileInputFormat.class);

job.setMapperClass(SelectorInverseMapper.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(SelectorEntry.class);
job.setPartitionerClass(URLPartitioner.class);
job.setReducerClass(PartitionReducer.class);
job.setNumReduceTasks(numLists);

FileOutputFormat.setOutputPath(job, output);
job.setOutputFormat(SequenceFileOutputFormat.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(CrawlDatum.class);
job.setOutputKeyComparatorClass(HashComparator.class);
JobClient.runJob(job);
```



本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ segments ▸ 20130914173850 ▸ crawl_generate

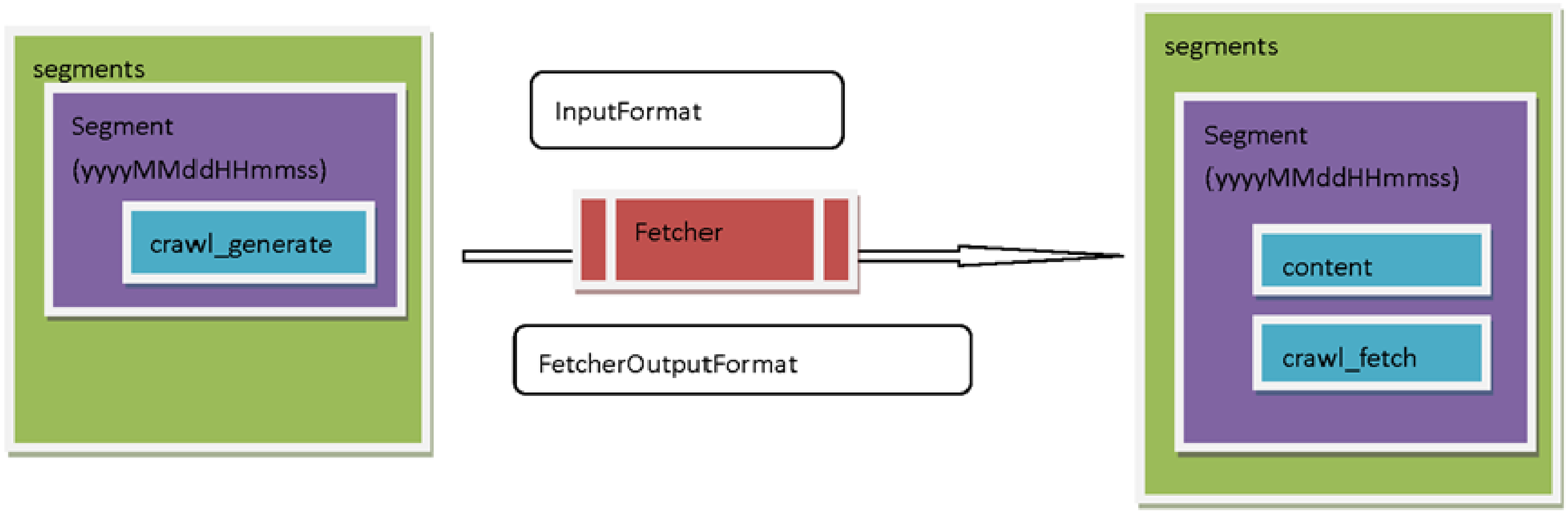
名称	修改日期	类型	大小
.part-00000.crc	2013/9/14 17:38	CRC 文件	1 KB
part-00000	2013/9/14 17:38	文件	121 KB

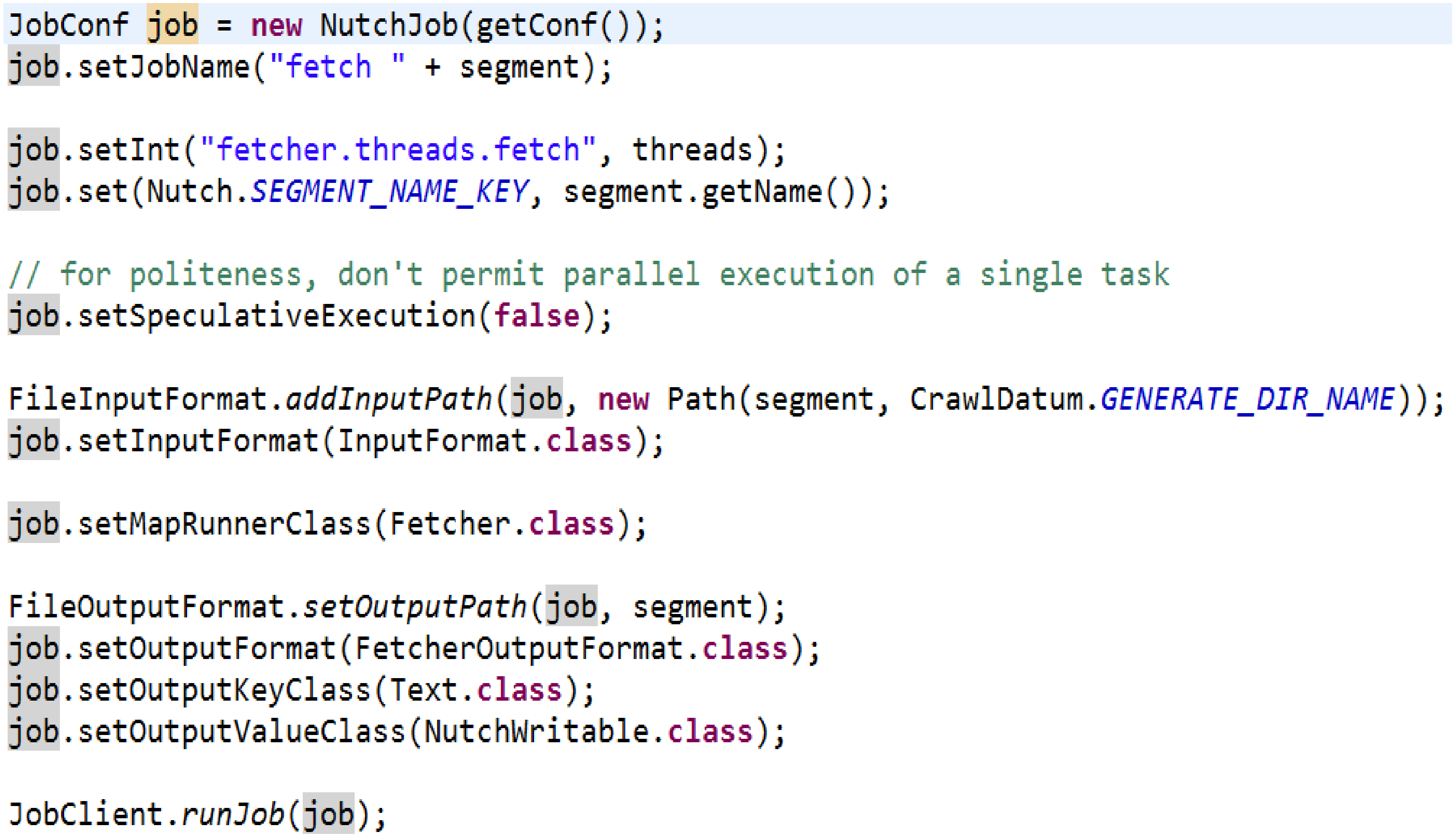
```
Generator: starting at 2013-09-16 03:43:29
Generator: Selecting best-scoring urls due for fetch.
Generator: filtering: true
Generator: normalizing: true
Generator: topN: 1
Generator: jobtracker is 'local', generating exactly one partition.
Generator: Partitioning selected urls for politeness.
Generator: segment: data/segments/20130916034333
Generator: finished at 2013-09-16 03:43:34, elapsed: 00:00:04
```



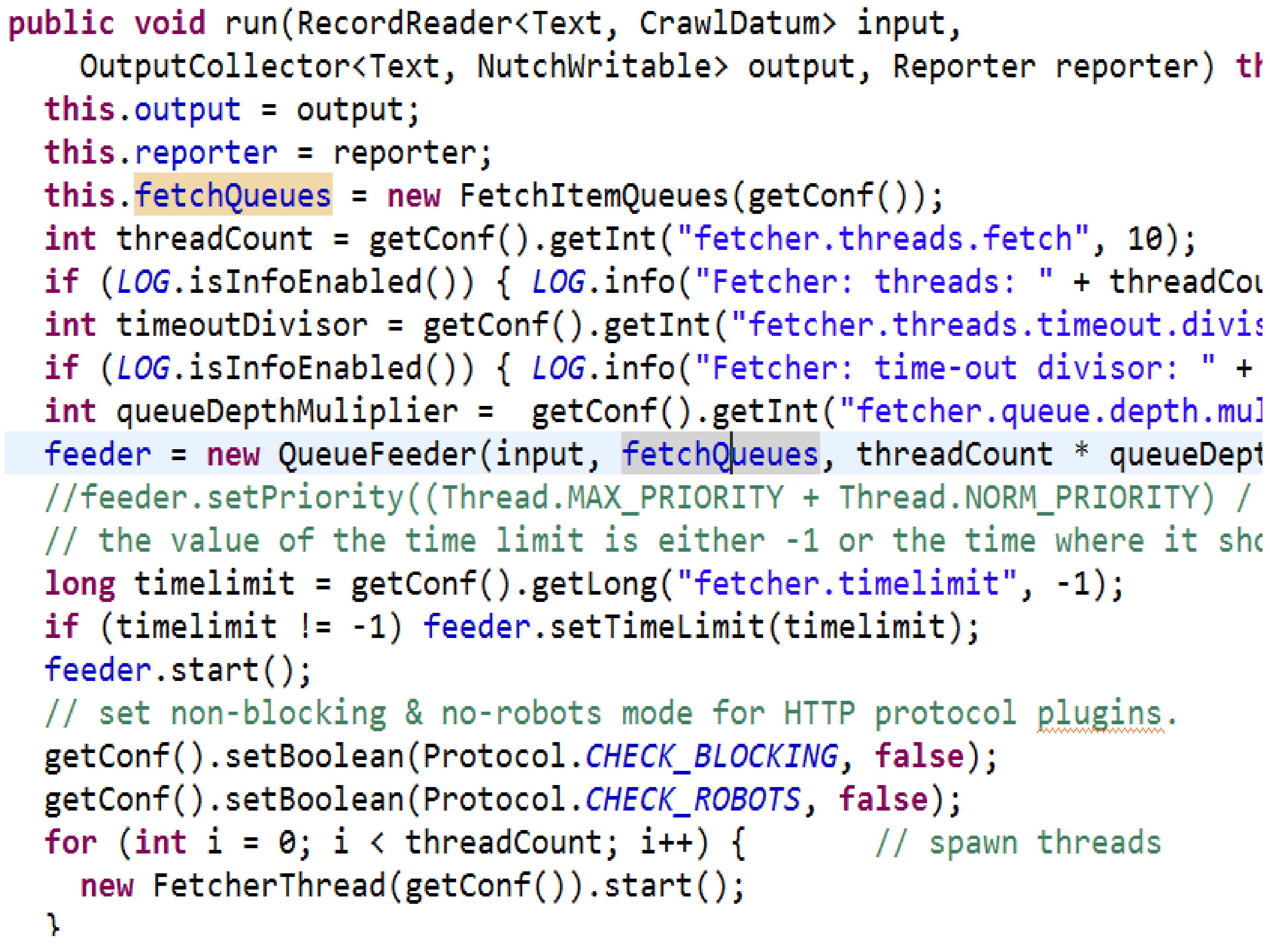
```
FetchSchedule 1231532  12-1-15 上午1:24  lewismc
  AbstractFetchSchedule 1494776  13-6-20 上午5:26  snagel
    AdaptiveFetchSchedule 1492639  13-6-13 下午8:10  markus
      MimeAdaptiveFetchSchedule 1349226  12-6-12 下午6:1
    DefaultFetchSchedule 964165  10-7-15 上午4:11  mattmann
```

8.3 Fetcher(抓取网页)

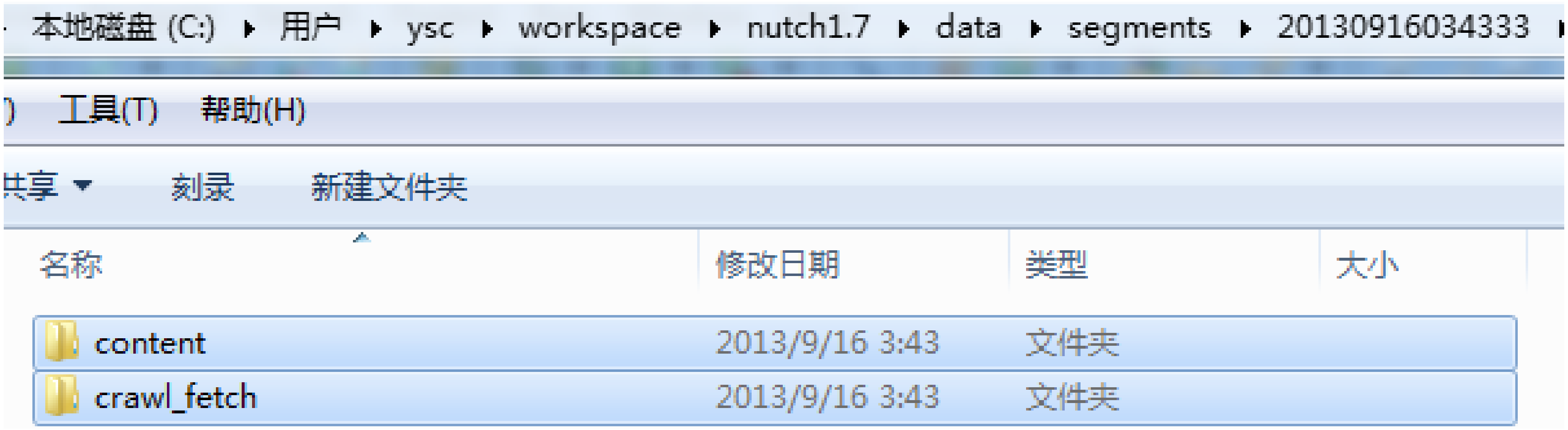




```java
JobConf job = new NutchJob(getConf());
job.setJobName("fetch " + segment);

job.setInt("fetcher.threads.fetch", threads);
job.set(Nutch.SEGMENT_NAME_KEY, segment.getName());

// for politeness, don't permit parallel execution of a single task
job.setSpeculativeExecution(false);

FileInputFormat.addInputPath(job, new Path(segment, CrawlDatum.GENERATE_DIR_NAME));
job.setInputFormat(InputFormat.class);

job.setMapRunnerClass(Fetcher.class);

FileOutputFormat.setOutputPath(job, segment);
job.setOutputFormat(FetcherOutputFormat.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(NutchWritable.class);

JobClient.runJob(job);
```

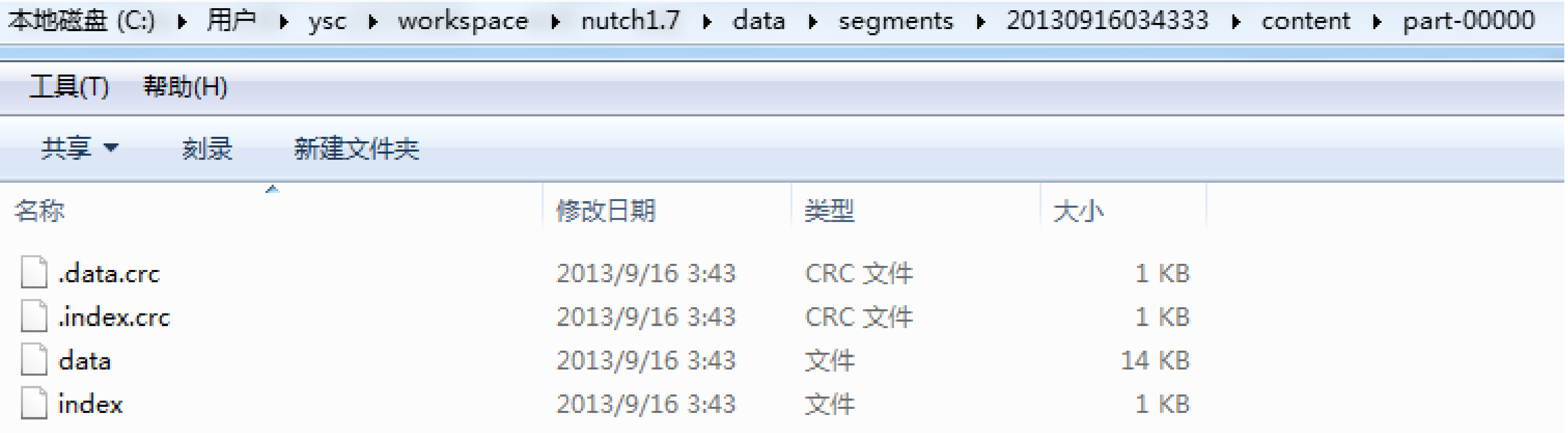


```
public void run(RecordReader<Text, CrawlDatum> input,
    OutputCollector<Text, NutchWritable> output, Reporter reporter) th
  this.output = output;
  this.reporter = reporter;
  this.fetchQueues = new FetchItemQueues(getConf());
  int threadCount = getConf().getInt("fetcher.threads.fetch", 10);
  if (LOG.isInfoEnabled()) { LOG.info("Fetcher: threads: " + threadCou
  int timeoutDivisor = getConf().getInt("fetcher.threads.timeout.divis
  if (LOG.isInfoEnabled()) { LOG.info("Fetcher: time-out divisor: " +
  int queueDepthMuliplier =  getConf().getInt("fetcher.queue.depth.mul
  feeder = new QueueFeeder(input, fetchQueues, threadCount * queueDept
  //feeder.setPriority((Thread.MAX_PRIORITY + Thread.NORM_PRIORITY) /
  // the value of the time limit is either -1 or the time where it sho
  long timelimit = getConf().getLong("fetcher.timelimit", -1);
  if (timelimit != -1) feeder.setTimeLimit(timelimit);
  feeder.start();
  // set non-blocking & no-robots mode for HTTP protocol plugins.
  getConf().setBoolean(Protocol.CHECK_BLOCKING, false);
  getConf().setBoolean(Protocol.CHECK_ROBOTS, false);
  for (int i = 0; i < threadCount; i++) {       // spawn threads
    new FetcherThread(getConf()).start();
  }
```



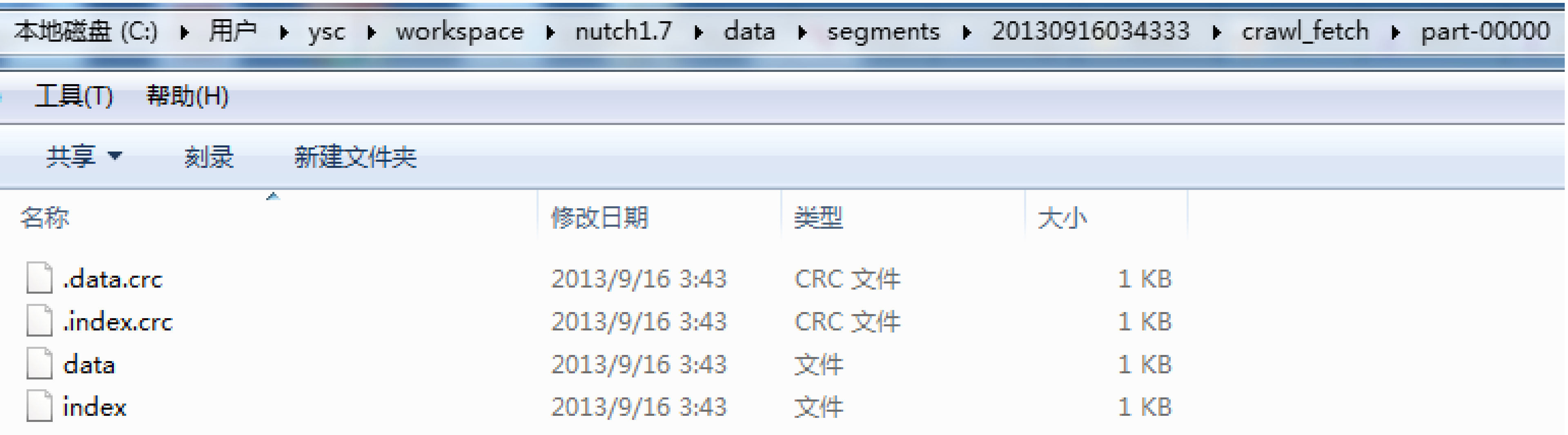
本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ segments ▸ 20130916034333

名称	修改日期	类型	大小
content	2013/9/16 3:43	文件夹	
crawl_fetch	2013/9/16 3:43	文件夹	



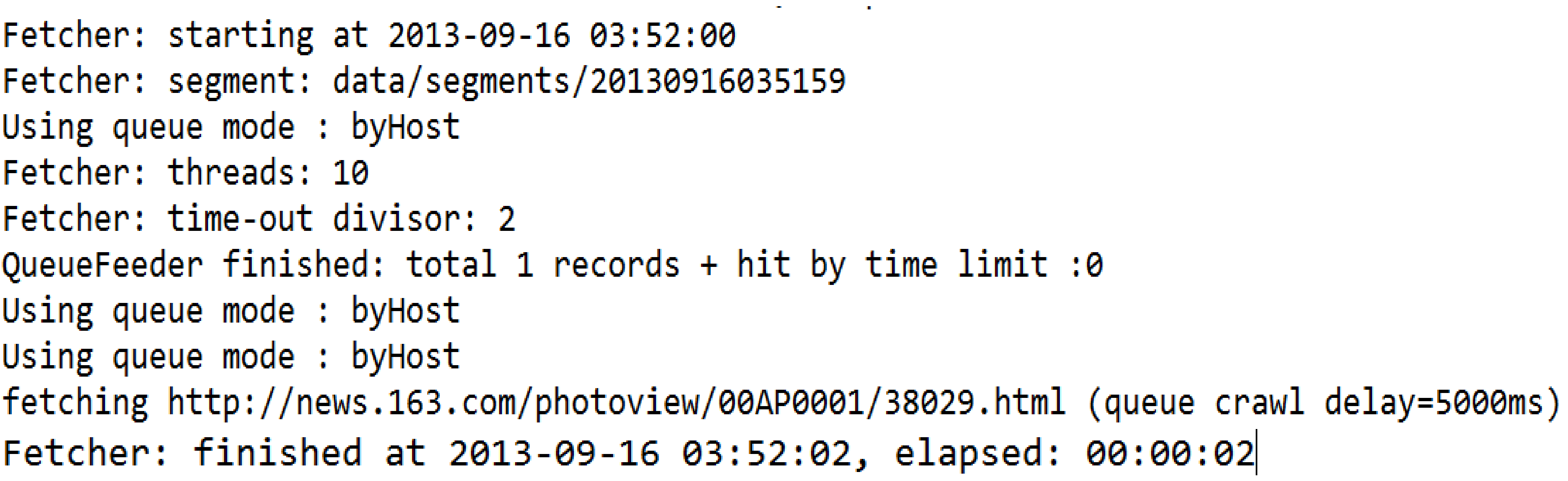
本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ segments ▸ 20130916034333 ▸ content ▸ part-00000

名称	修改日期	类型	大小
.data.crc	2013/9/16 3:43	CRC 文件	1 KB
.index.crc	2013/9/16 3:43	CRC 文件	1 KB
data	2013/9/16 3:43	文件	14 KB
index	2013/9/16 3:43	文件	1 KB



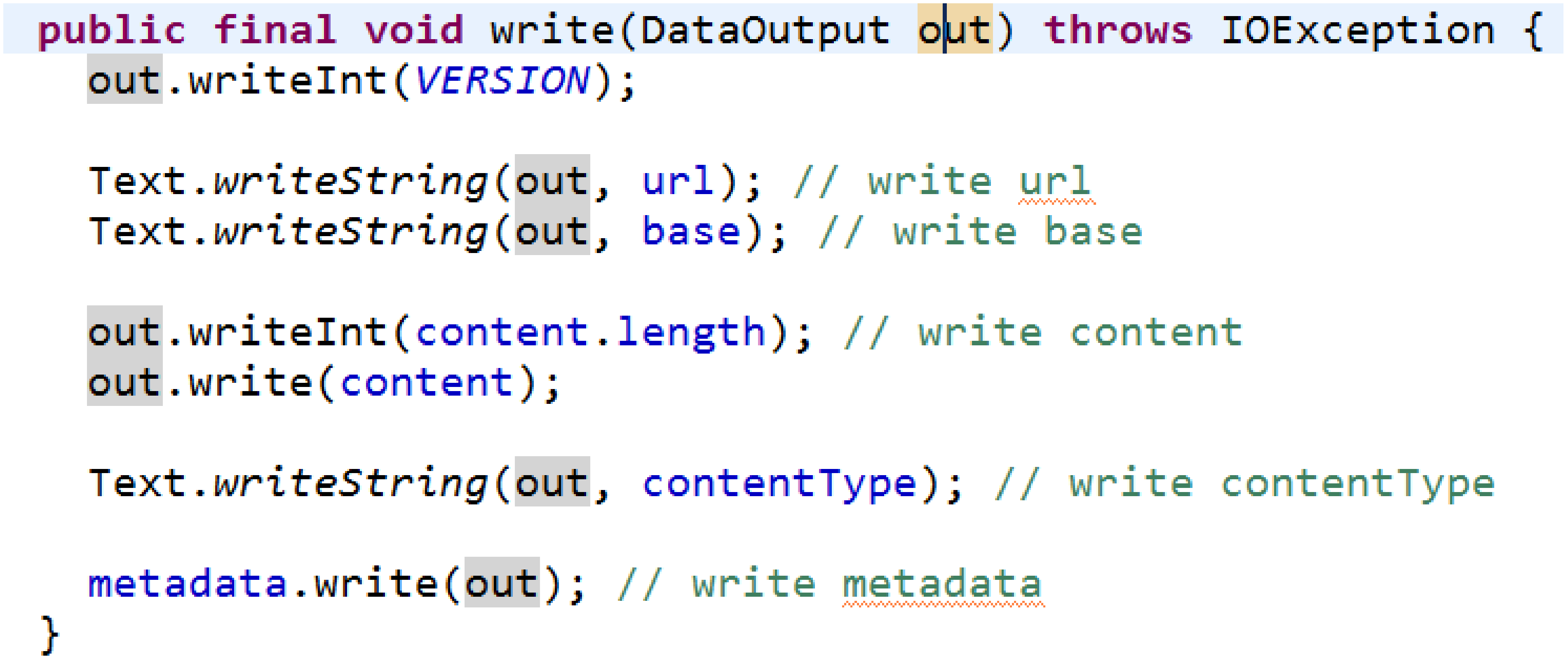
本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ segments ▸ 20130916034333 ▸ crawl_fetch ▸ part-00000

名称	修改日期	类型	大小
.data.crc	2013/9/16 3:43	CRC 文件	1 KB
.index.crc	2013/9/16 3:43	CRC 文件	1 KB
data	2013/9/16 3:43	文件	1 KB
index	2013/9/16 3:43	文件	1 KB



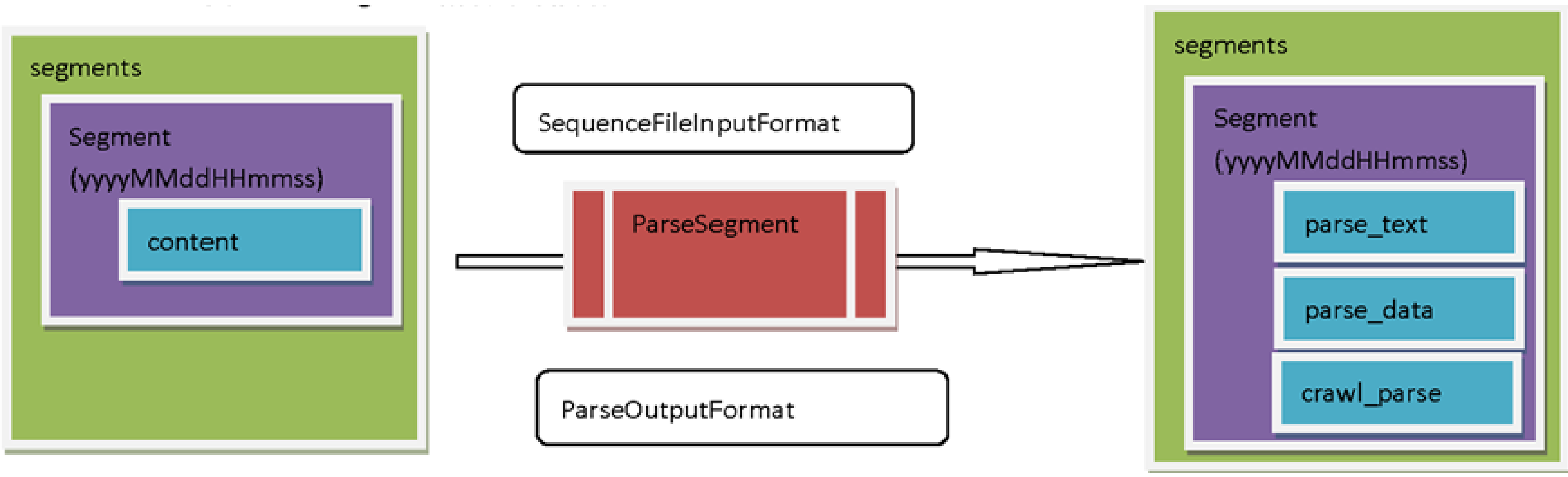
```
Fetcher: starting at 2013-09-16 03:52:00
Fetcher: segment: data/segments/20130916035159
Using queue mode : byHost
Fetcher: threads: 10
Fetcher: time-out divisor: 2
QueueFeeder finished: total 1 records + hit by time limit :0
Using queue mode : byHost
Using queue mode : byHost
fetching http://news.163.com/photoview/00AP0001/38029.html (queue crawl delay=5000ms)
Fetcher: finished at 2013-09-16 03:52:02, elapsed: 00:00:02
```

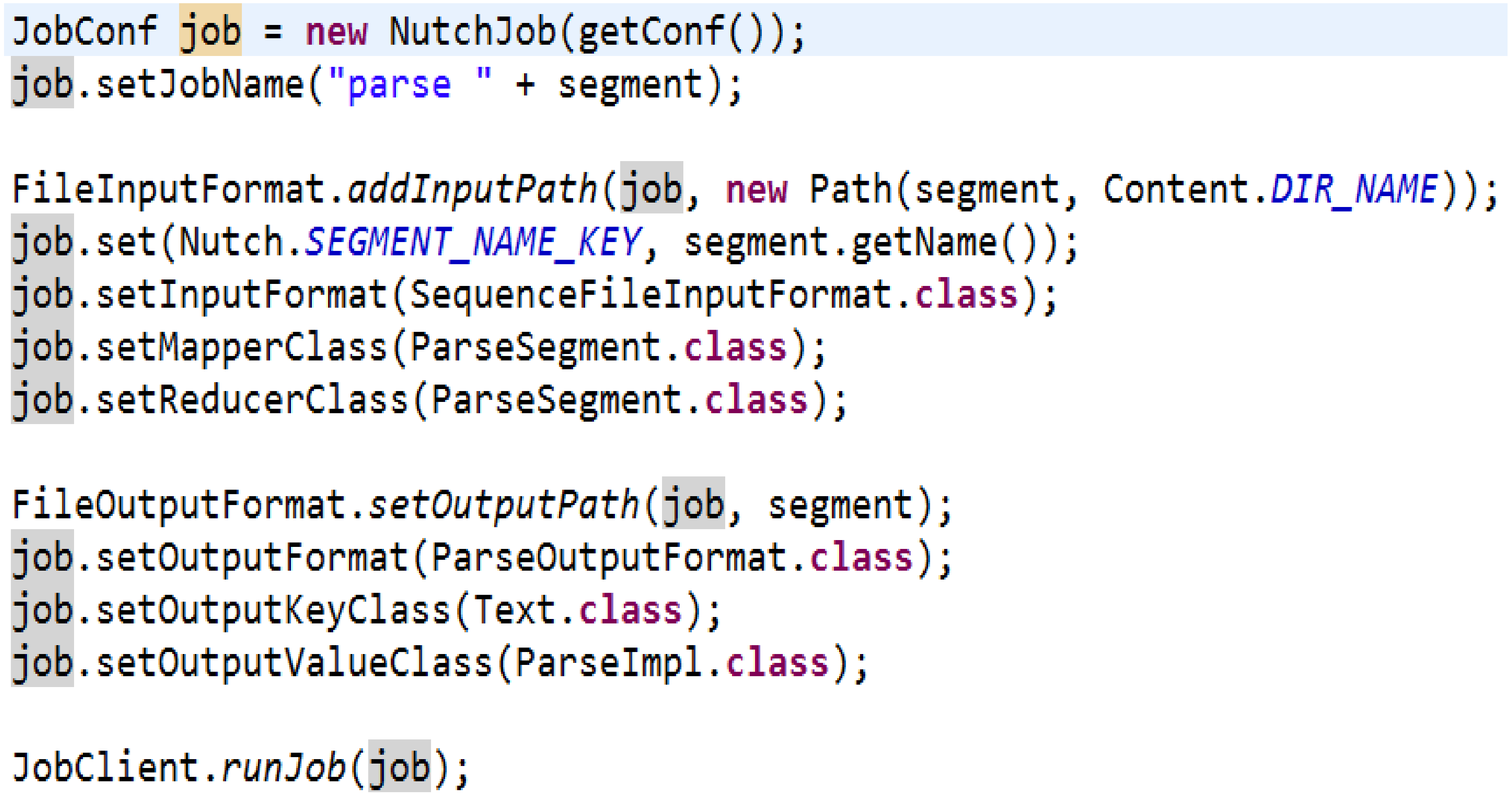
content是MapFile <Text, Content > Content结构如右图所示



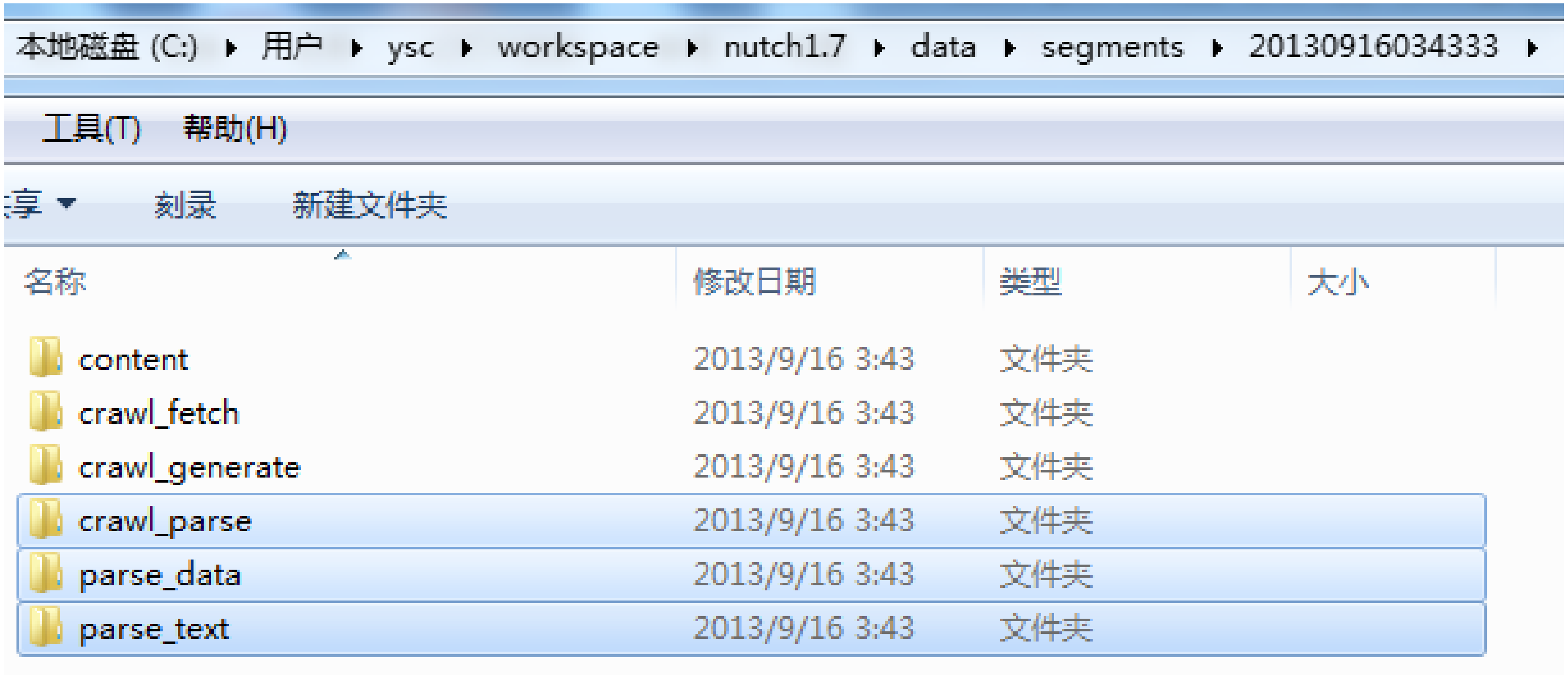
```
public final void write(DataOutput out) throws IOException {
    out.writeInt(VERSION);

    Text.writeString(out, url); // write url
    Text.writeString(out, base); // write base

    out.writeInt(content.length); // write content
    out.write(content);

    Text.writeString(out, contentType); // write contentType

    metadata.write(out); // write metadata
}
```

8.4 Parse segment(解析)



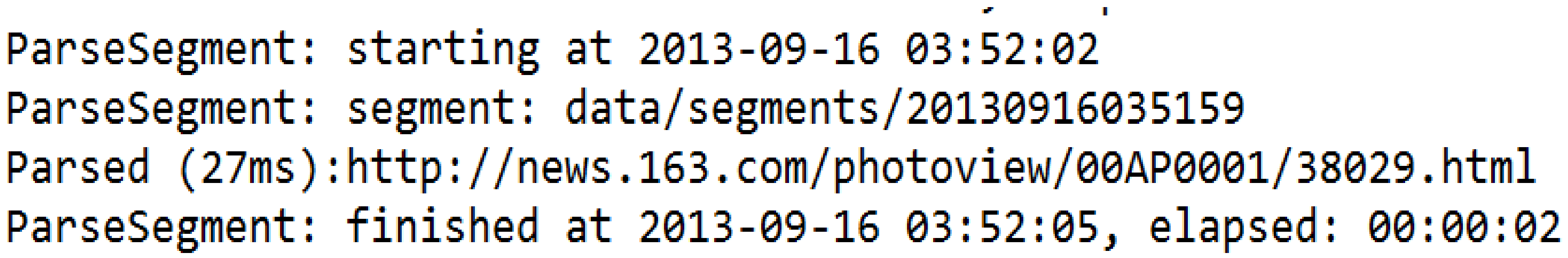


```
JobConf job = new NutchJob(getConf());
job.setJobName("parse " + segment);

FileInputFormat.addInputPath(job, new Path(segment, Content.DIR_NAME));
job.set(Nutch.SEGMENT_NAME_KEY, segment.getName());
job.setInputFormat(SequenceFileInputFormat.class);
job.setMapperClass(ParseSegment.class);
job.setReducerClass(ParseSegment.class);

FileOutputFormat.setOutputPath(job, segment);
job.setOutputFormat(ParseOutputFormat.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(ParseImpl.class);

JobClient.runJob(job);
```



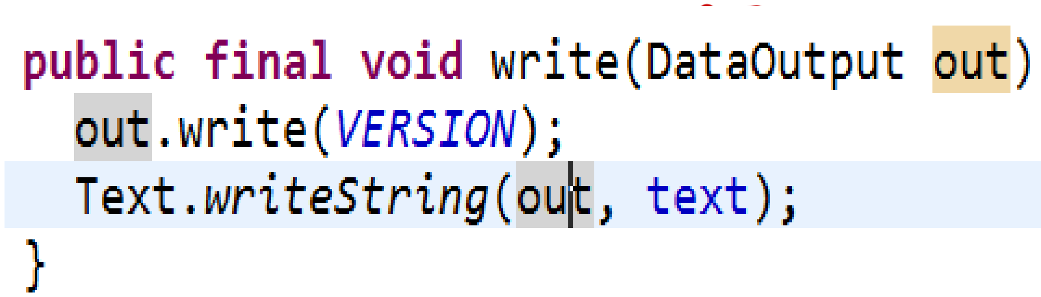
本地磁盘 (C:) ▸ 用户 ▸ ysc ▸ workspace ▸ nutch1.7 ▸ data ▸ segments ▸ 20130916034333 ▸

名称	修改日期	类型	大小
content	2013/9/16 3:43	文件夹	
crawl_fetch	2013/9/16 3:43	文件夹	
crawl_generate	2013/9/16 3:43	文件夹	
crawl_parse	2013/9/16 3:43	文件夹	
parse_data	2013/9/16 3:43	文件夹	
parse_text	2013/9/16 3:43	文件夹	



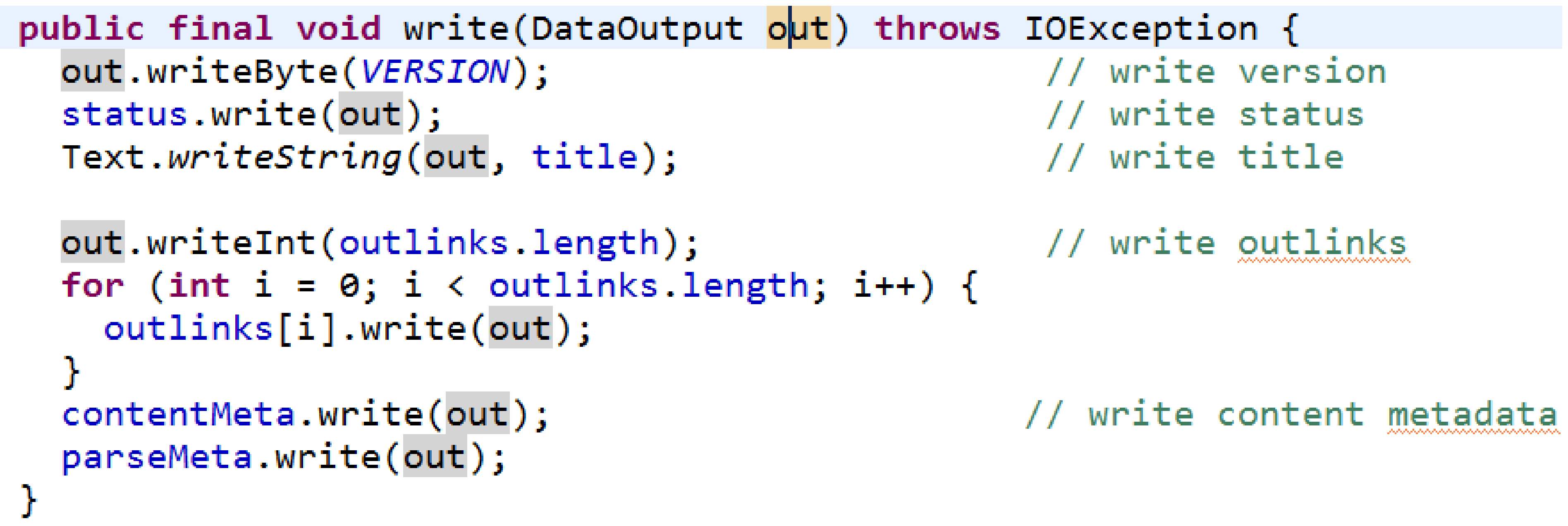
```
ParseSegment: starting at 2013-09-16 03:52:02
ParseSegment: segment: data/segments/20130916035159
Parsed (27ms):http://news.163.com/photoview/00AP0001/38029.html
ParseSegment: finished at 2013-09-16 03:52:05, elapsed: 00:00:02
```

parse_text是MapFile
<Text, ParseText>
ParseText结构如右图所示



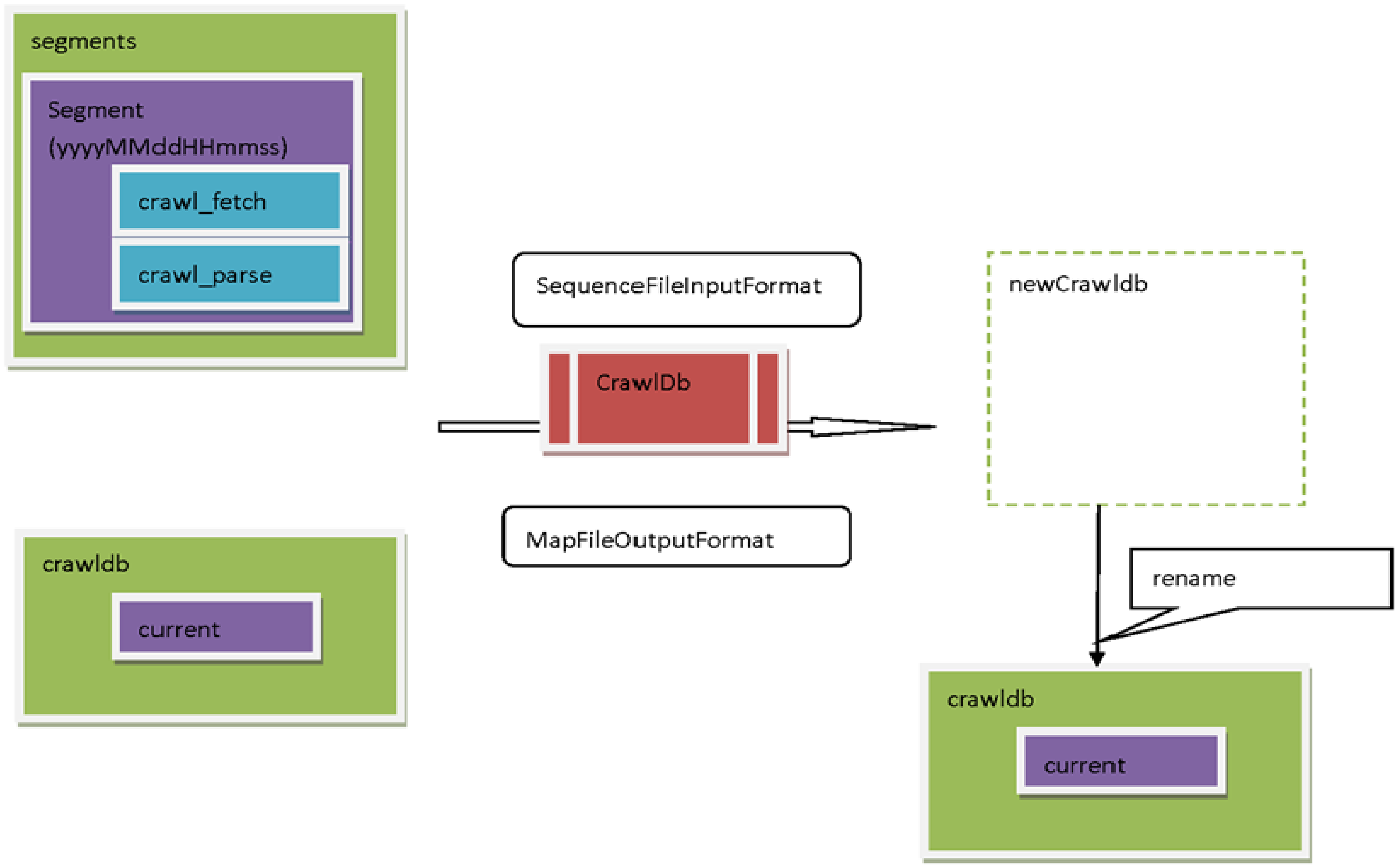
```
public final void write(DataOutput out)
  out.write(VERSION);
  Text.writeString(out, text);
}
```

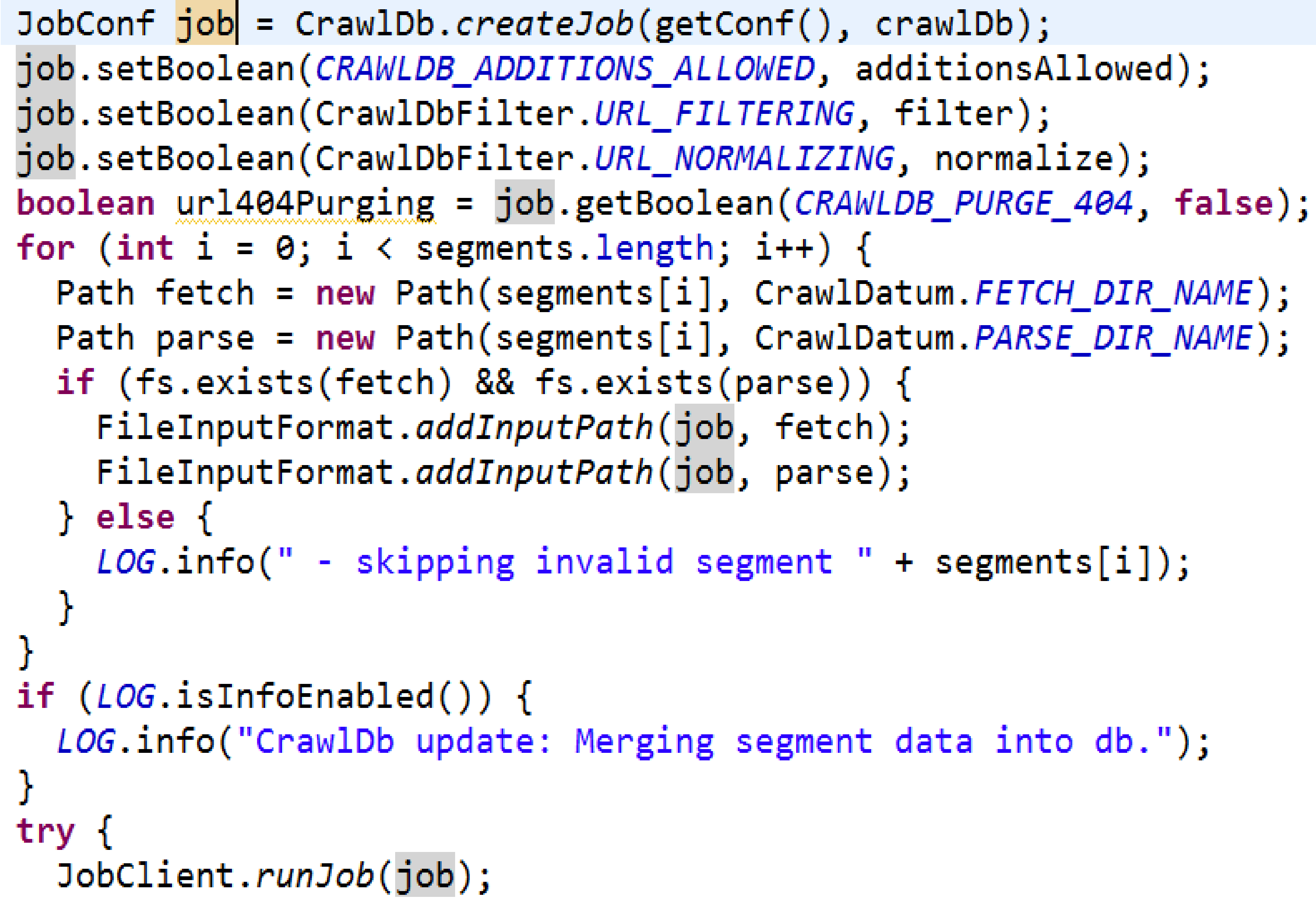
parse_data是MapFile
<Text, ParseData>
ParseData结构如下图所示



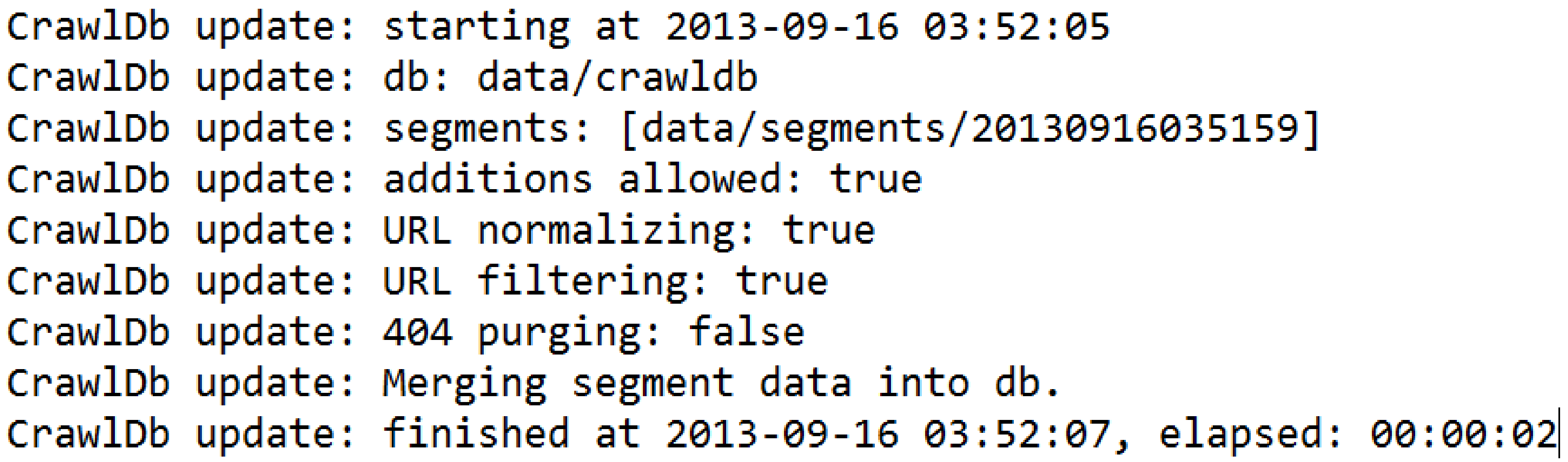
```
public final void write(DataOutput out) throws IOException {
  out.writeByte(VERSION);                          // write version
  status.write(out);                               // write status
  Text.writeString(out, title);                    // write title

  out.writeInt(outlinks.length);                   // write outlinks
  for (int i = 0; i < outlinks.length; i++) {
    outlinks[i].write(out);
  }
  contentMeta.write(out);                        // write content metadata
  parseMeta.write(out);
}
```

8.5 Update CrawlDB(更新crawldb)



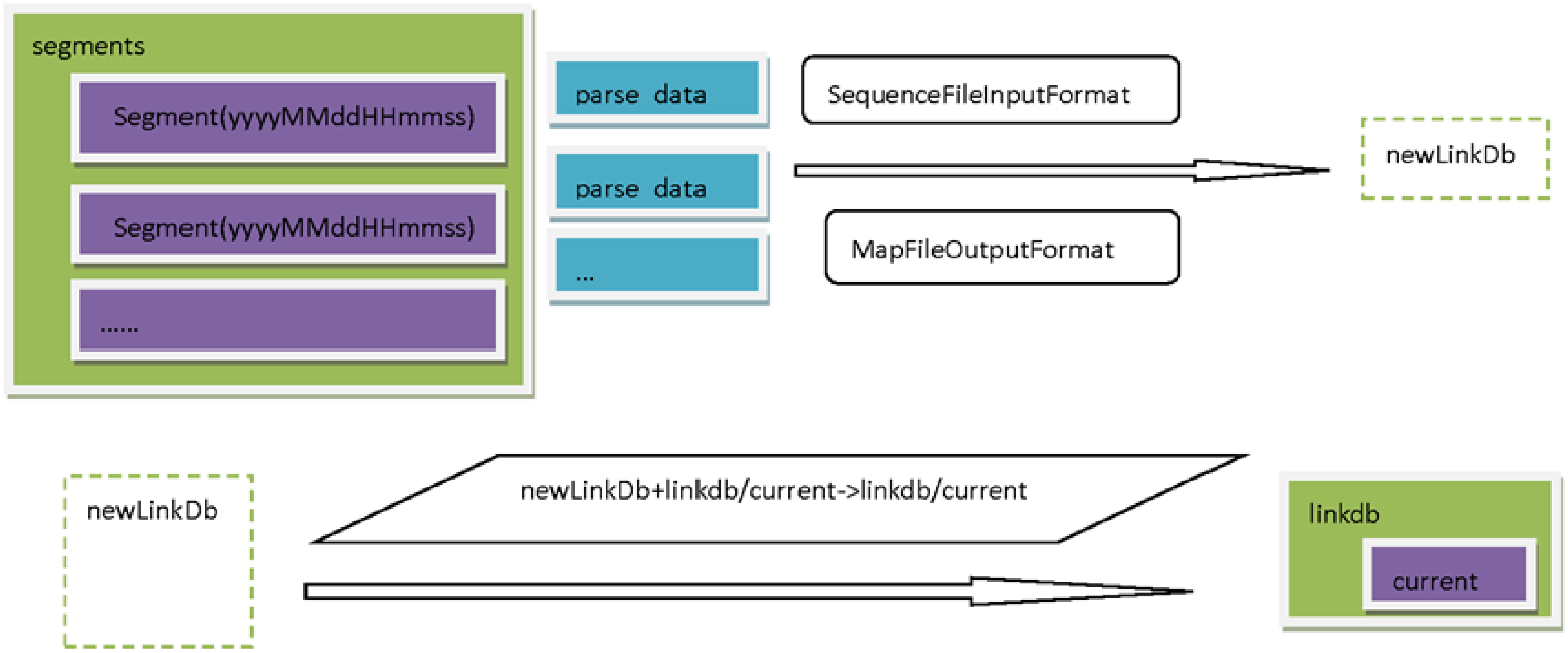


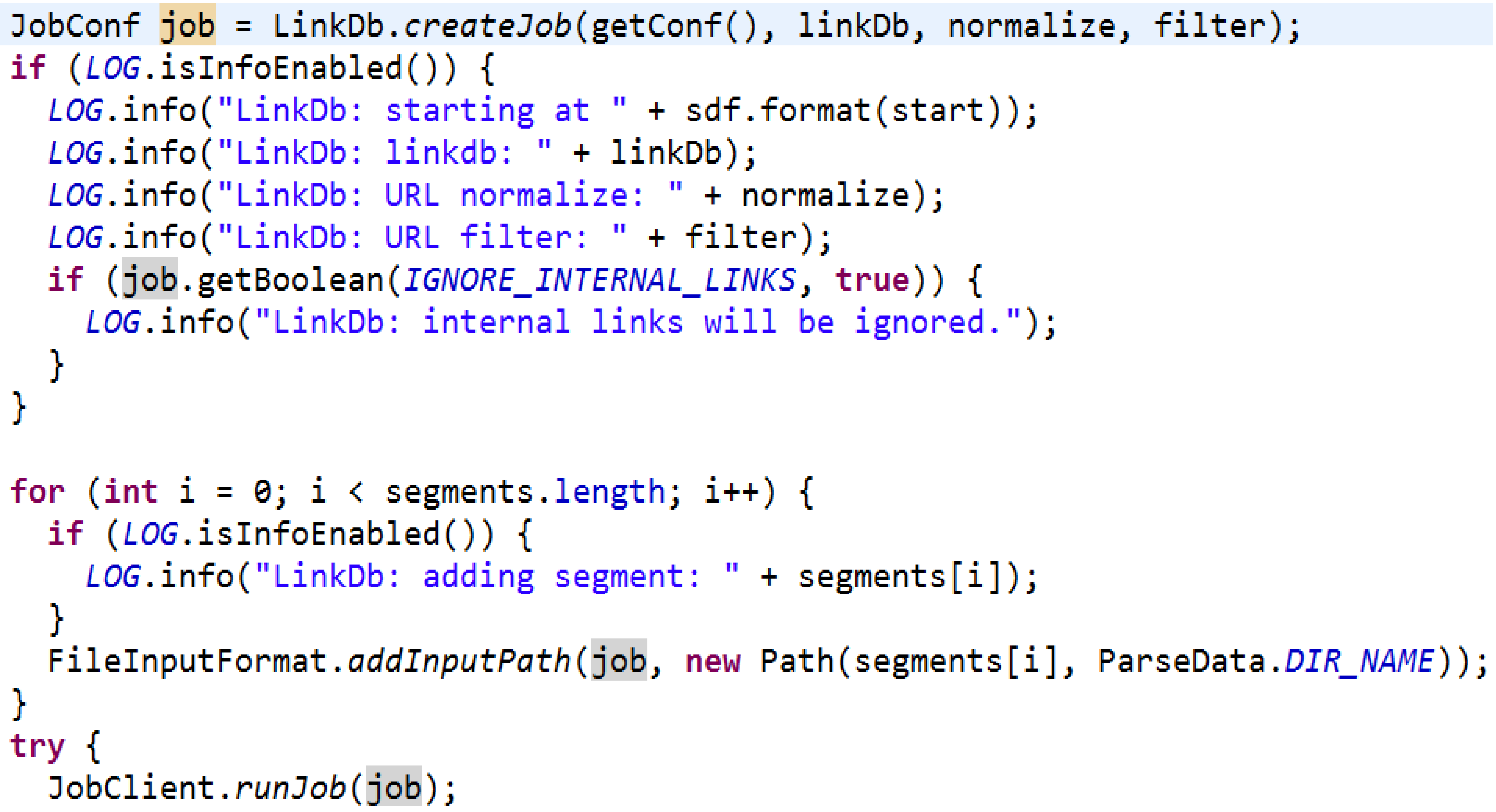
```java
JobConf job = CrawlDb.createJob(getConf(), crawlDb);
job.setBoolean(CRAWLDB_ADDITIONS_ALLOWED, additionsAllowed);
job.setBoolean(CrawlDbFilter.URL_FILTERING, filter);
job.setBoolean(CrawlDbFilter.URL_NORMALIZING, normalize);
boolean url404Purging = job.getBoolean(CRAWLDB_PURGE_404, false);
for (int i = 0; i < segments.length; i++) {
  Path fetch = new Path(segments[i], CrawlDatum.FETCH_DIR_NAME);
  Path parse = new Path(segments[i], CrawlDatum.PARSE_DIR_NAME);
  if (fs.exists(fetch) && fs.exists(parse)) {
    FileInputFormat.addInputPath(job, fetch);
    FileInputFormat.addInputPath(job, parse);
  } else {
    LOG.info(" - skipping invalid segment " + segments[i]);
  }
}
if (LOG.isInfoEnabled()) {
  LOG.info("CrawlDb update: Merging segment data into db.");
}
try {
  JobClient.runJob(job);
```



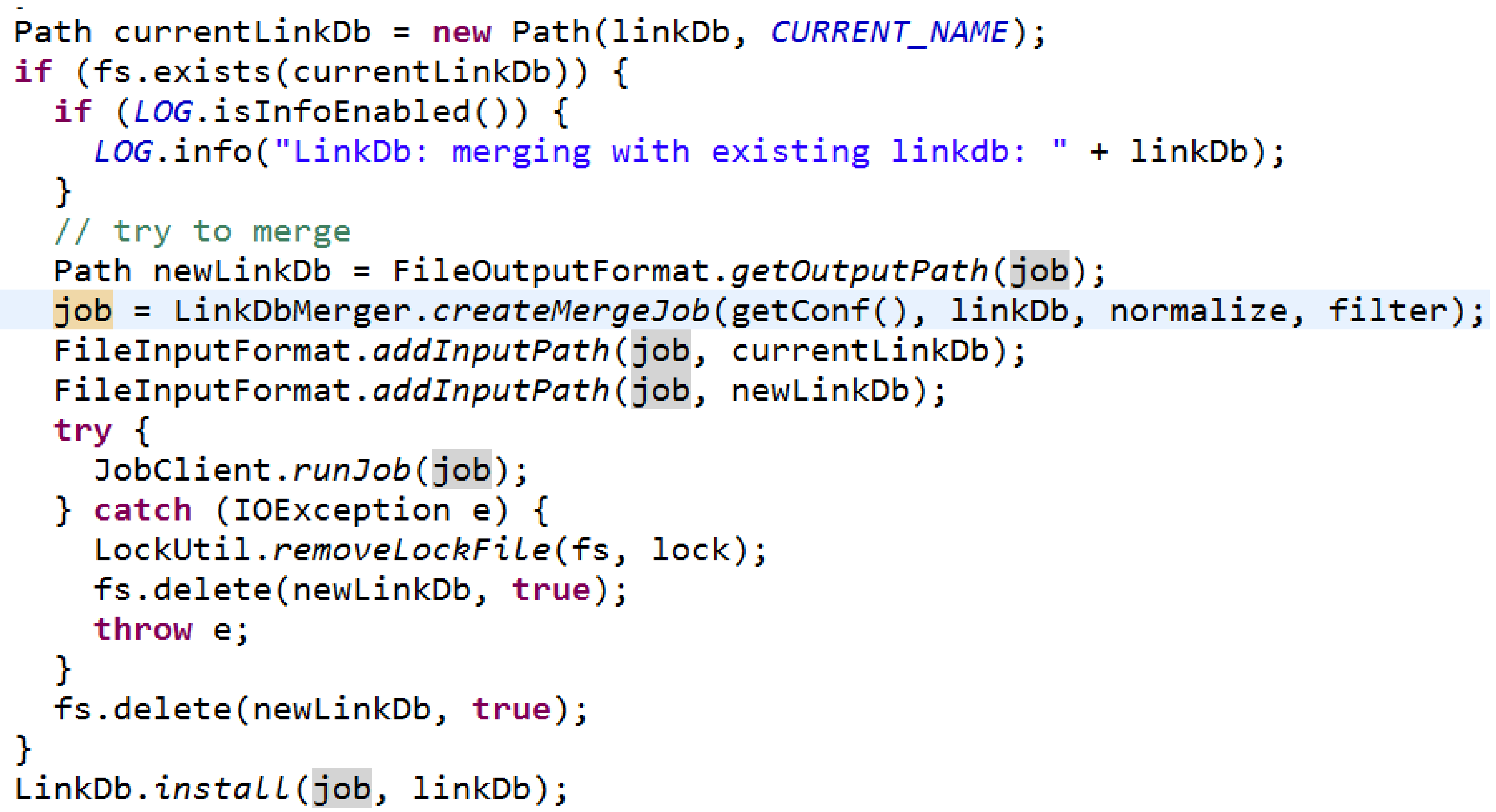
```
CrawlDb update: starting at 2013-09-16 03:52:05
CrawlDb update: db: data/crawldb
CrawlDb update: segments: [data/segments/20130916035159]
CrawlDb update: additions allowed: true
CrawlDb update: URL normalizing: true
CrawlDb update: URL filtering: true
CrawlDb update: 404 purging: false
CrawlDb update: Merging segment data into db.
CrawlDb update: finished at 2013-09-16 03:52:07, elapsed: 00:00:02
```

8.6 InvertLinks(链接反转)

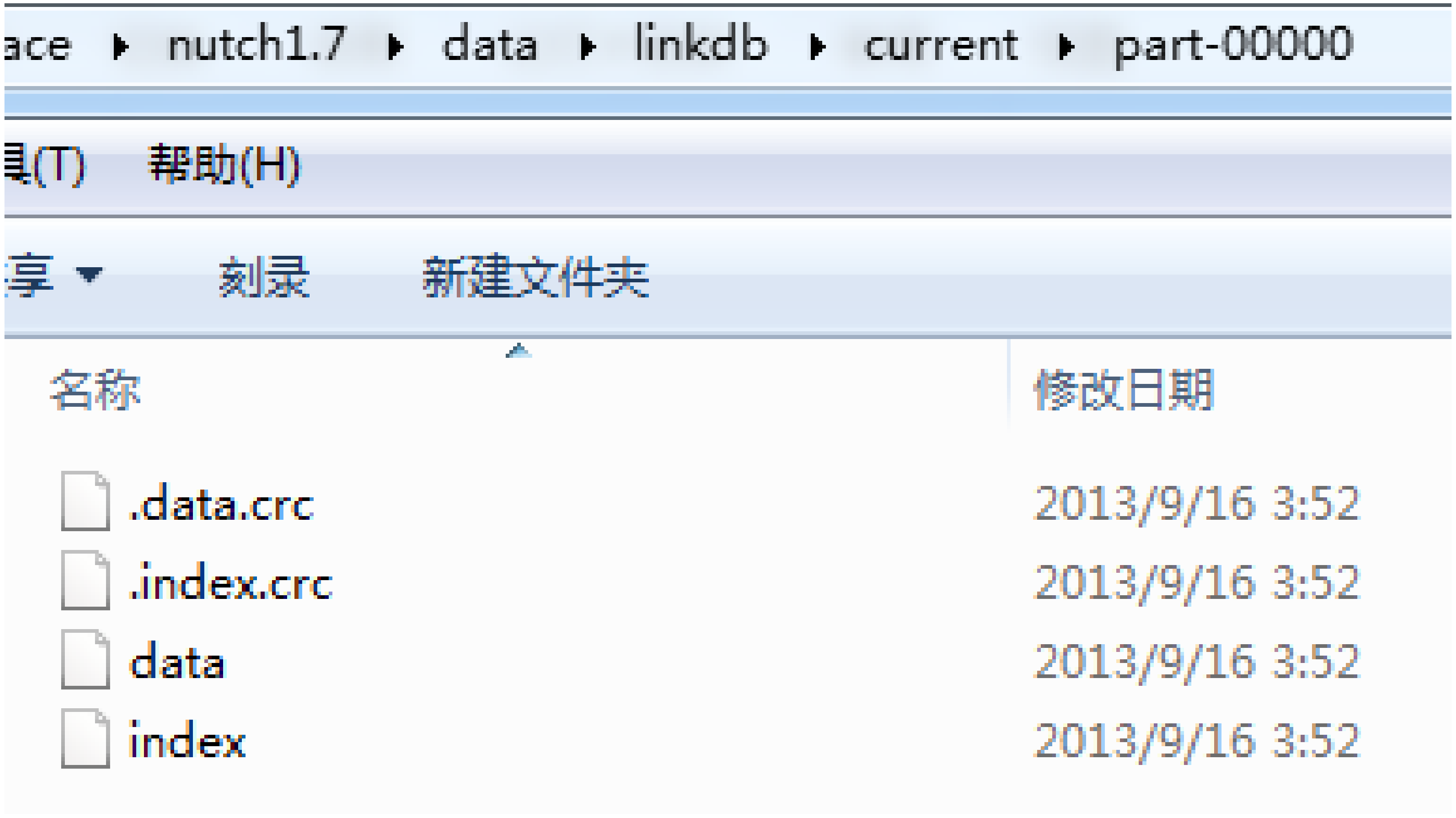




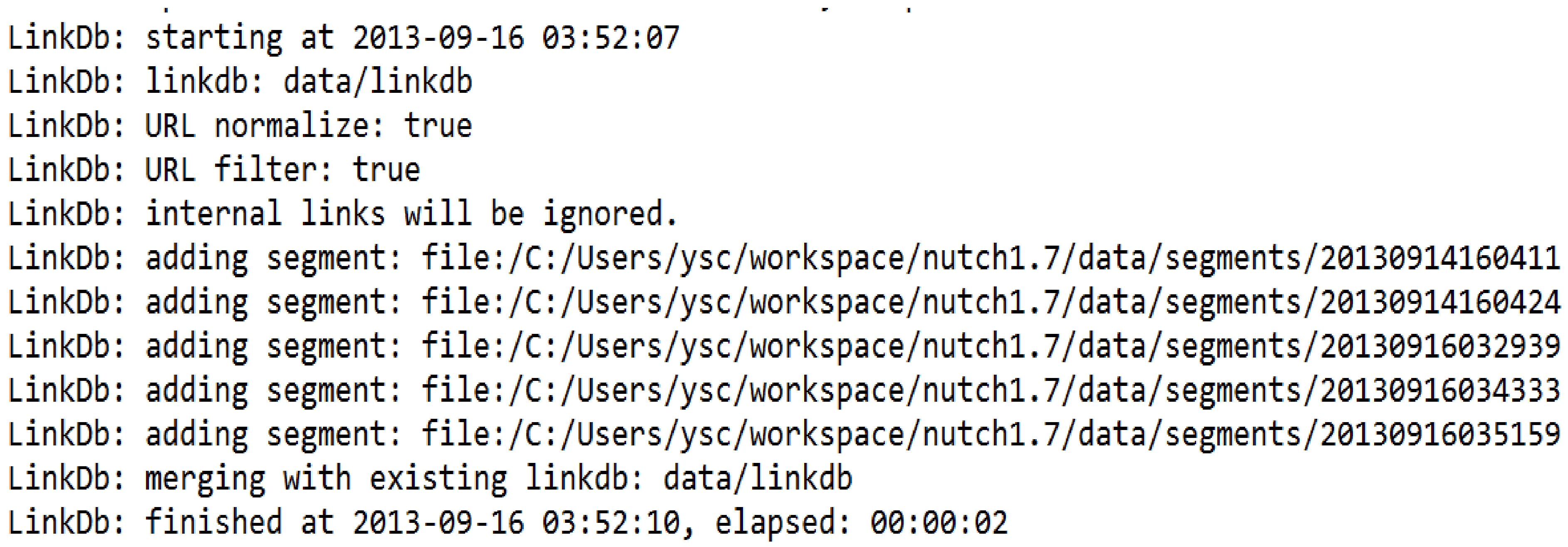
```java
JobConf job = LinkDb.createJob(getConf(), linkDb, normalize, filter);
if (LOG.isInfoEnabled()) {
  LOG.info("LinkDb: starting at " + sdf.format(start));
  LOG.info("LinkDb: linkdb: " + linkDb);
  LOG.info("LinkDb: URL normalize: " + normalize);
  LOG.info("LinkDb: URL filter: " + filter);
  if (job.getBoolean(IGNORE_INTERNAL_LINKS, true)) {
    LOG.info("LinkDb: internal links will be ignored.");
  }
}

for (int i = 0; i < segments.length; i++) {
  if (LOG.isInfoEnabled()) {
    LOG.info("LinkDb: adding segment: " + segments[i]);
  }
  FileInputFormat.addInputPath(job, new Path(segments[i], ParseData.DIR_NAME));
}
try {
  JobClient.runJob(job);
```



```
Path currentLinkDb = new Path(linkDb, CURRENT_NAME);
if (fs.exists(currentLinkDb)) {
  if (LOG.isInfoEnabled()) {
    LOG.info("LinkDb: merging with existing linkdb: " + linkDb);
  }
  // try to merge
  Path newLinkDb = FileOutputFormat.getOutputPath(job);
  job = LinkDbMerger.createMergeJob(getConf(), linkDb, normalize, filter);
  FileInputFormat.addInputPath(job, currentLinkDb);
  FileInputFormat.addInputPath(job, newLinkDb);
  try {
    JobClient.runJob(job);
  } catch (IOException e) {
    LockUtil.removeLockFile(fs, lock);
    fs.delete(newLinkDb, true);
    throw e;
  }
  fs.delete(newLinkDb, true);
}
LinkDb.install(job, linkDb);
```

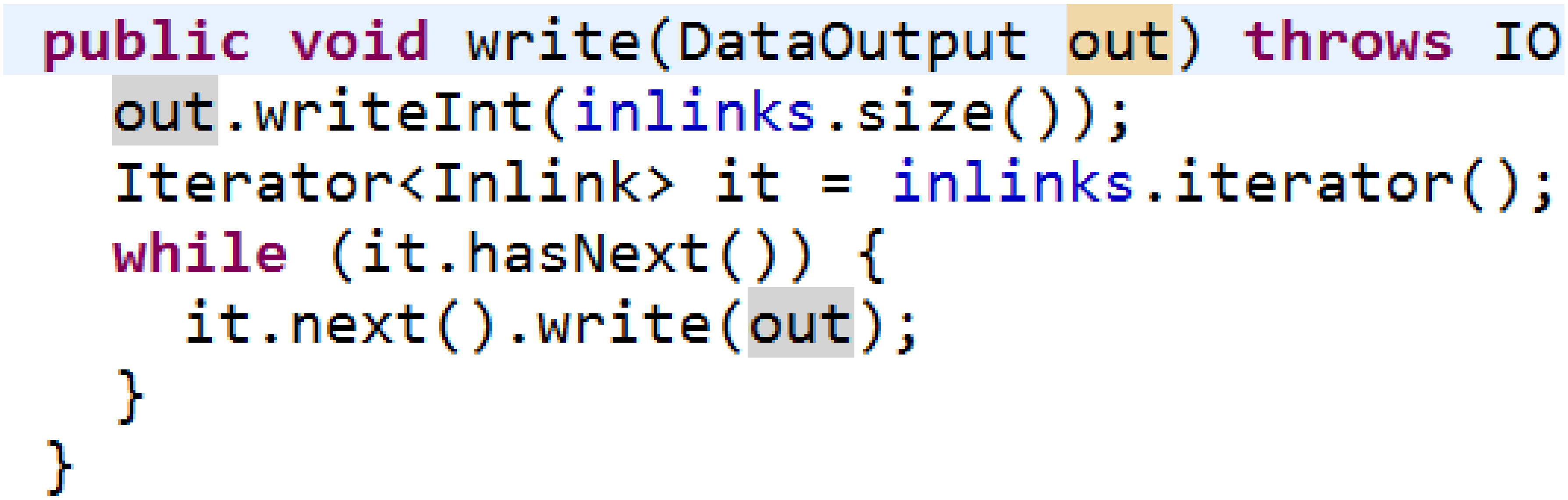


名称	修改日期
.data.crc	2013/9/16 3:52
.index.crc	2013/9/16 3:52
data	2013/9/16 3:52
index	2013/9/16 3:52

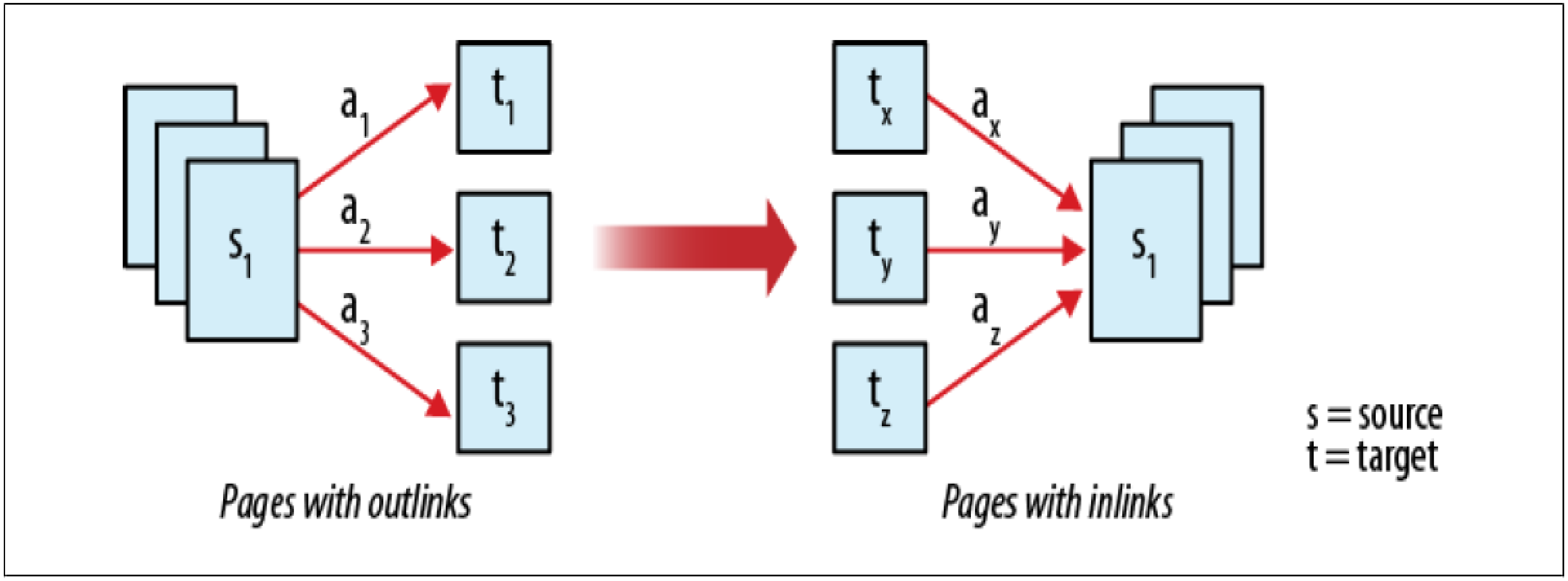


```
LinkDb: starting at 2013-09-16 03:52:07
LinkDb: linkdb: data/linkdb
LinkDb: URL normalize: true
LinkDb: URL filter: true
LinkDb: internal links will be ignored.
LinkDb: adding segment: file:/C:/Users/ysc/workspace/nutch1.7/data/segments/20130914160411
LinkDb: adding segment: file:/C:/Users/ysc/workspace/nutch1.7/data/segments/20130914160424
LinkDb: adding segment: file:/C:/Users/ysc/workspace/nutch1.7/data/segments/20130916032939
LinkDb: adding segment: file:/C:/Users/ysc/workspace/nutch1.7/data/segments/20130916034333
LinkDb: adding segment: file:/C:/Users/ysc/workspace/nutch1.7/data/segments/20130916035159
LinkDb: merging with existing linkdb: data/linkdb
LinkDb: finished at 2013-09-16 03:52:10, elapsed: 00:00:02
```

LinkDB是MapFile
<Text, Inlinks>
Inlinks结构如下图所示



```
public void write(DataOutput out) throws IO
   out.writeInt(inlinks.size());
   Iterator<Inlink> it = inlinks.iterator();
   while (it.hasNext()) {
     it.next().write(out);
   }
}
```



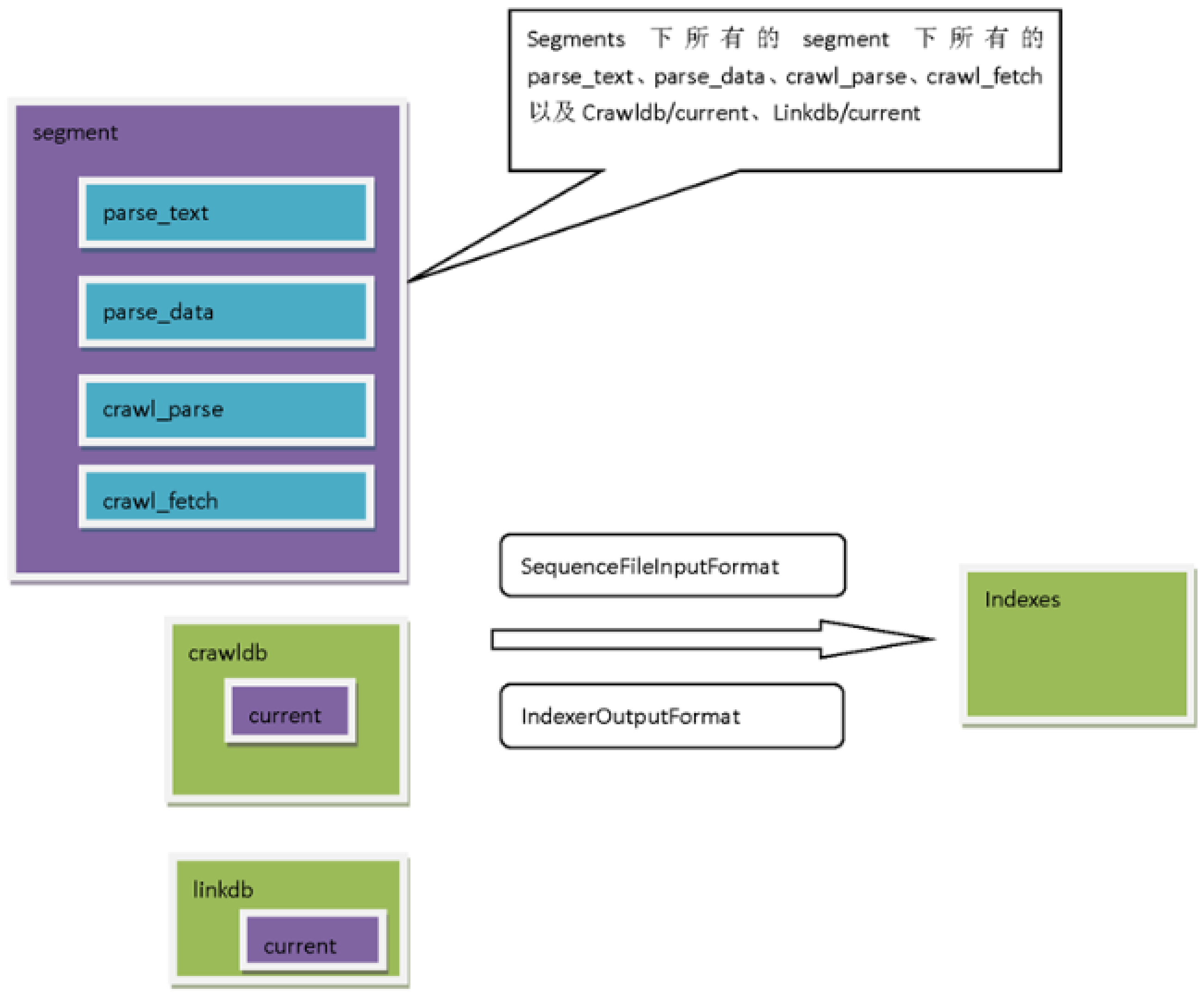
map输入<url, outlinks>
链接反转, map输出:

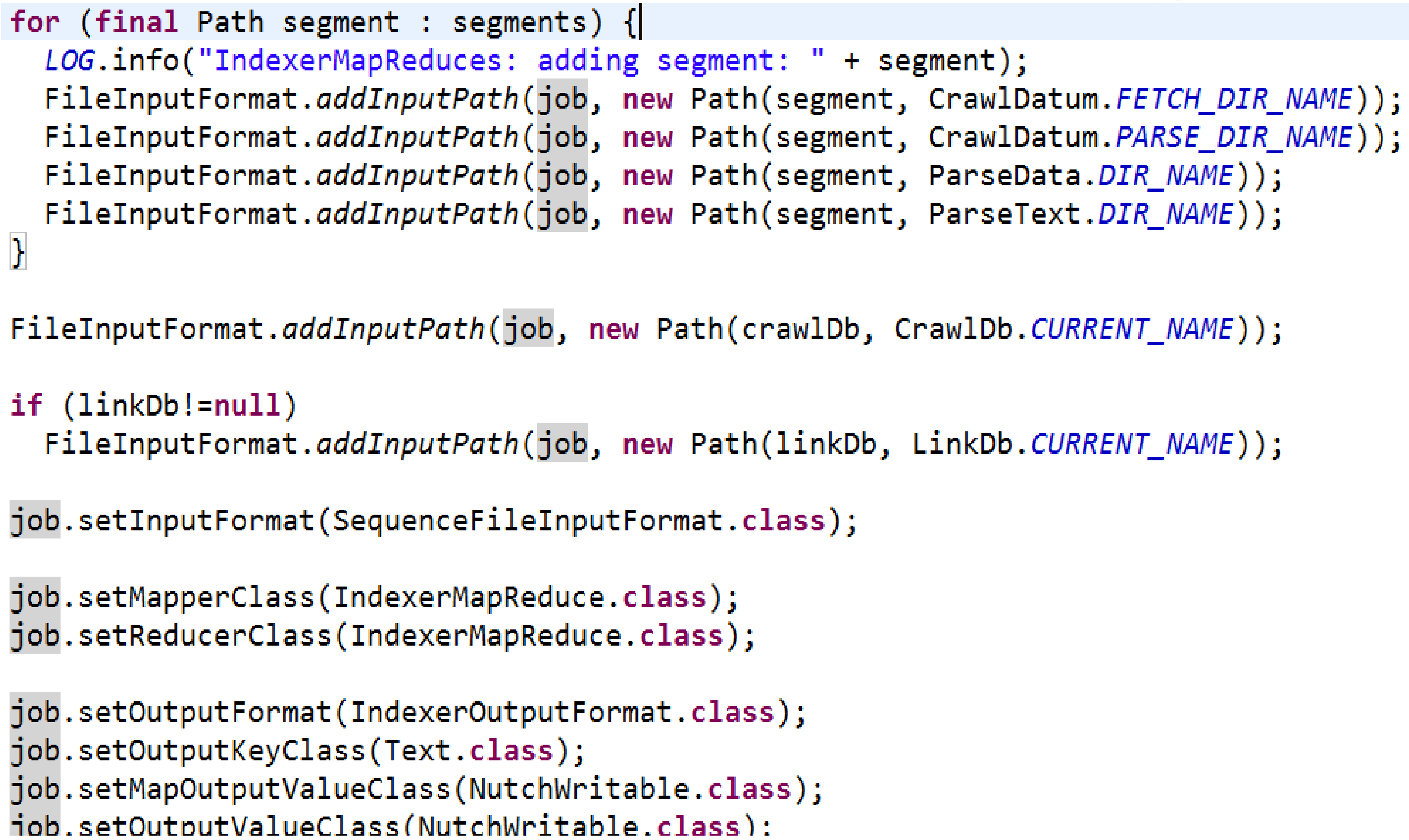
t1 -> s1
t2 -> s1
t3 -> s1
s1 -> tx
s1 -> ty
s1 -> tz

map的输出作为reduce的输入
ruduce输入:

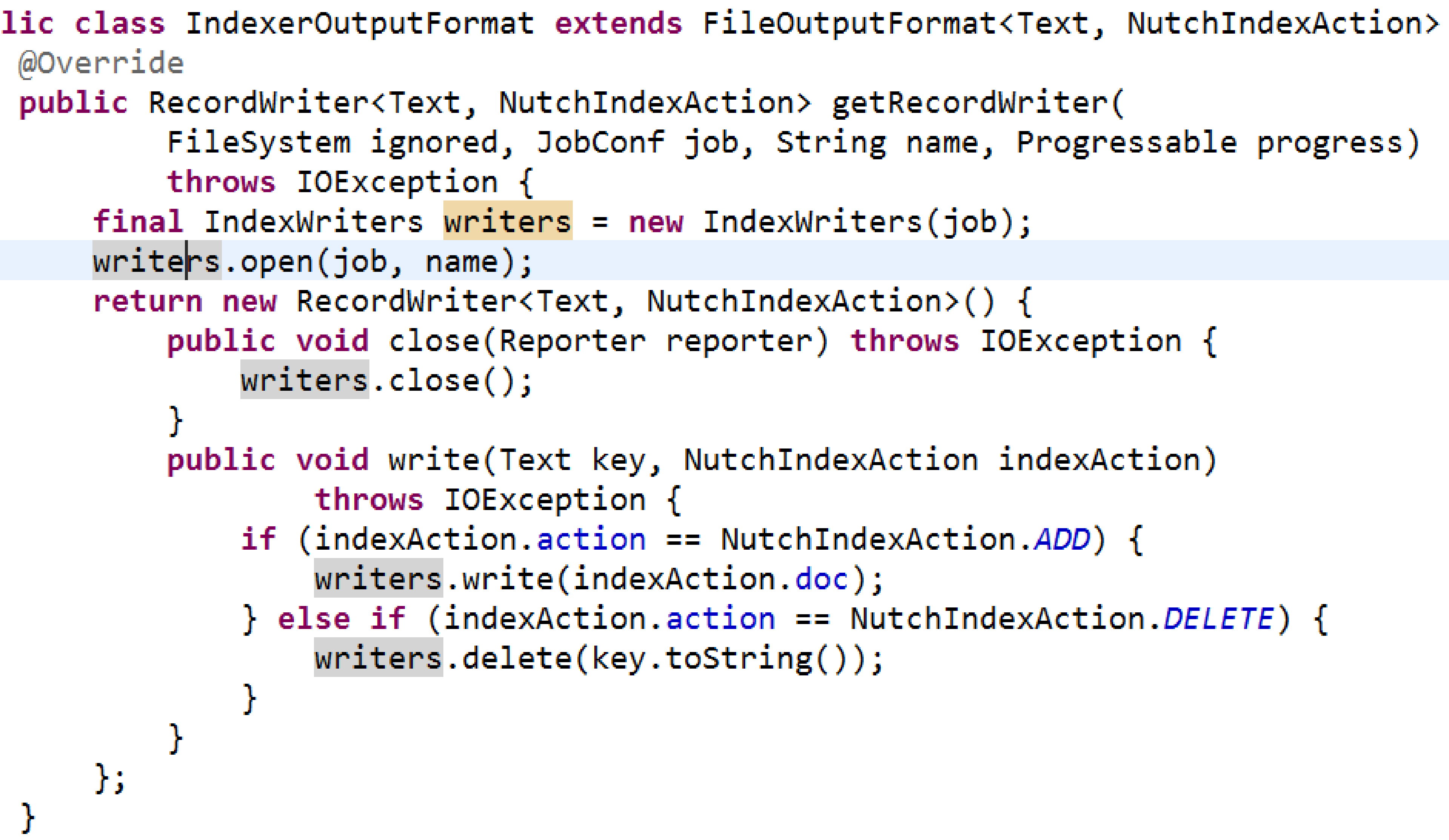
t1 -> s1
t2 -> s1
t3 -> s1
s1 -> tx, ty, tz

8.7 Indexer (建立索引)



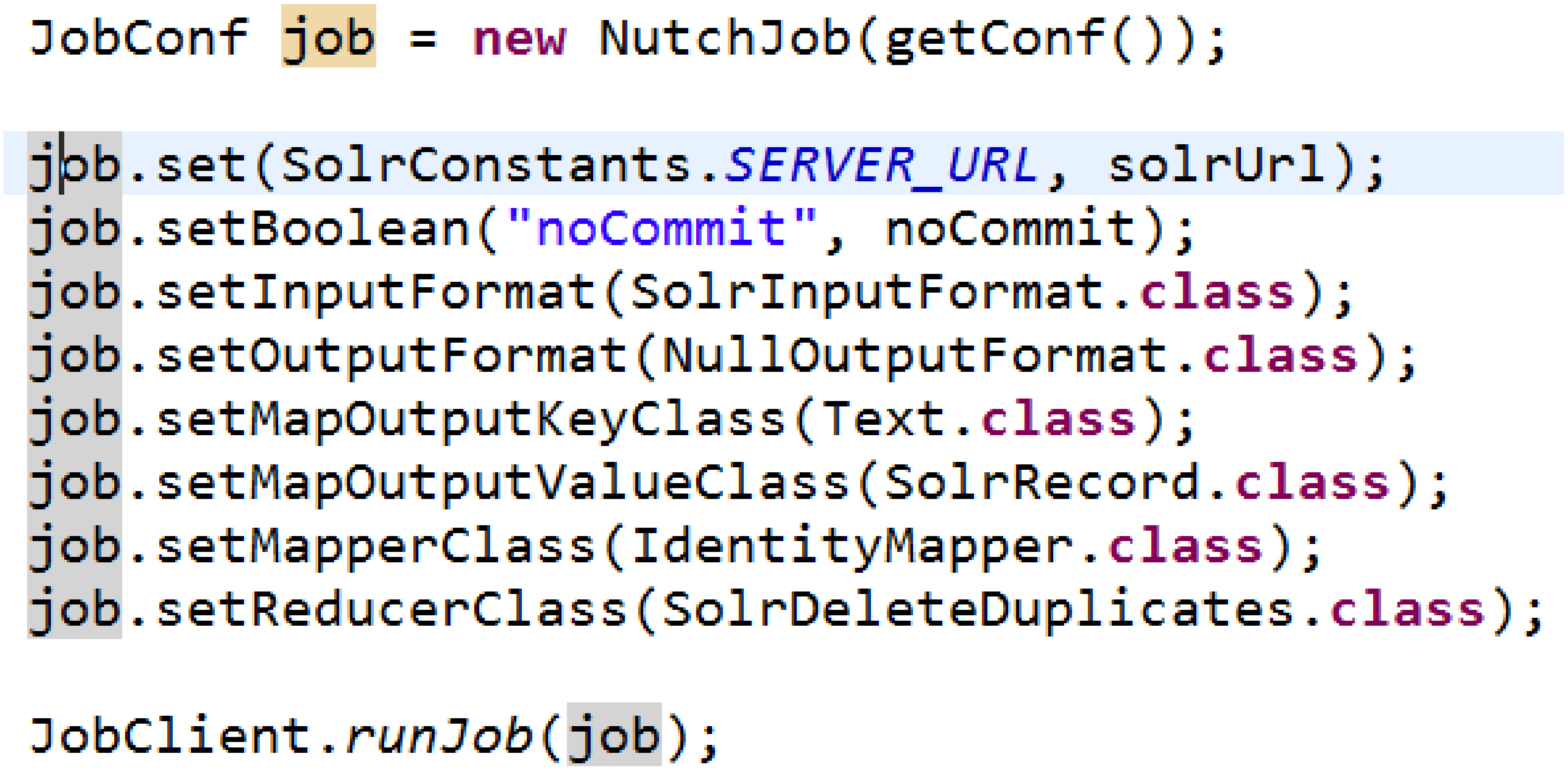


```
for (final Path segment : segments) {
  LOG.info("IndexerMapReduces: adding segment: " + segment);
  FileInputFormat.addInputPath(job, new Path(segment, CrawlDatum.FETCH_DIR_NAME));
  FileInputFormat.addInputPath(job, new Path(segment, CrawlDatum.PARSE_DIR_NAME));
  FileInputFormat.addInputPath(job, new Path(segment, ParseData.DIR_NAME));
  FileInputFormat.addInputPath(job, new Path(segment, ParseText.DIR_NAME));
}

FileInputFormat.addInputPath(job, new Path(crawlDb, CrawlDb.CURRENT_NAME));

if (linkDb!=null)
  FileInputFormat.addInputPath(job, new Path(linkDb, LinkDb.CURRENT_NAME));

job.setInputFormat(SequenceFileInputFormat.class);

job.setMapperClass(IndexerMapReduce.class);
job.setReducerClass(IndexerMapReduce.class);

job.setOutputFormat(IndexerOutputFormat.class);
job.setOutputKeyClass(Text.class);
job.setMapOutputValueClass(NutchWritable.class);
job.setOutputValueClass(NutchWritable.class);
```



```
lic class IndexerOutputFormat extends FileOutputFormat<Text, NutchIndexAction>
 @Override
 public RecordWriter<Text, NutchIndexAction> getRecordWriter(
         FileSystem ignored, JobConf job, String name, Progressable progress)
         throws IOException {
     final IndexWriters writers = new IndexWriters(job);
     writers.open(job, name);
     return new RecordWriter<Text, NutchIndexAction>() {
         public void close(Reporter reporter) throws IOException {
             writers.close();
         }
         public void write(Text key, NutchIndexAction indexAction)
                 throws IOException {
             if (indexAction.action == NutchIndexAction.ADD) {
                 writers.write(indexAction.doc);
             } else if (indexAction.action == NutchIndexAction.DELETE) {
                 writers.delete(key.toString());
             }
         }
     };
 }
```

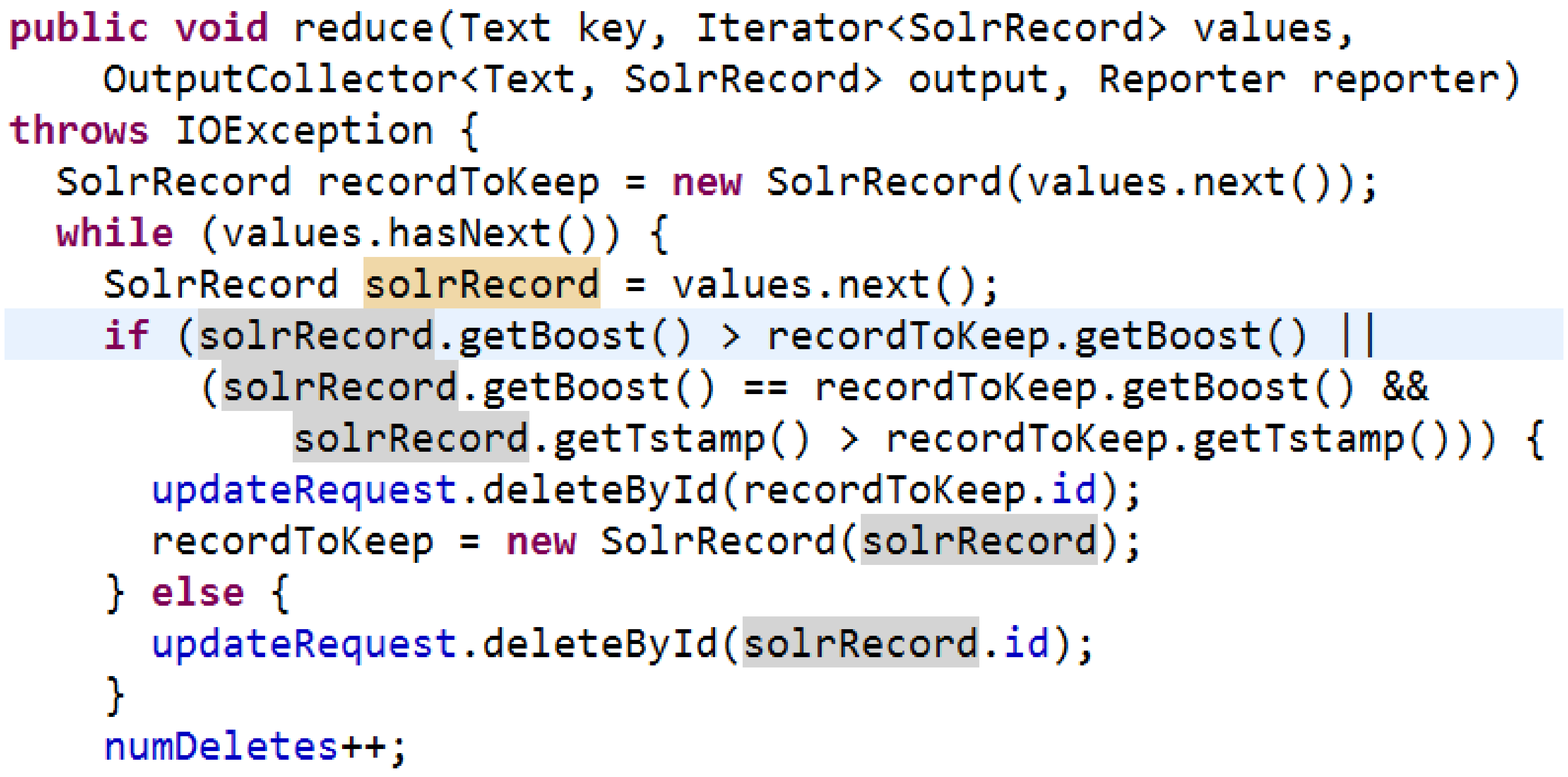
8.8 Index Dedup (索引去重)



```
JobConf job = new NutchJob(getConf());

job.set(SolrConstants.SERVER_URL, solrUrl);
job.setBoolean("noCommit", noCommit);
job.setInputFormat(SolrInputFormat.class);
job.setOutputFormat(NullOutputFormat.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(SolrRecord.class);
job.setMapperClass(IdentityMapper.class);
job.setReducerClass(SolrDeleteDuplicates.class);

JobClient.runJob(job);
```

注意：这里的KEY是页面的数字签名(digest)，因此这种去重方法会存在一个网页的多个版本

开源力量 Open Source Force



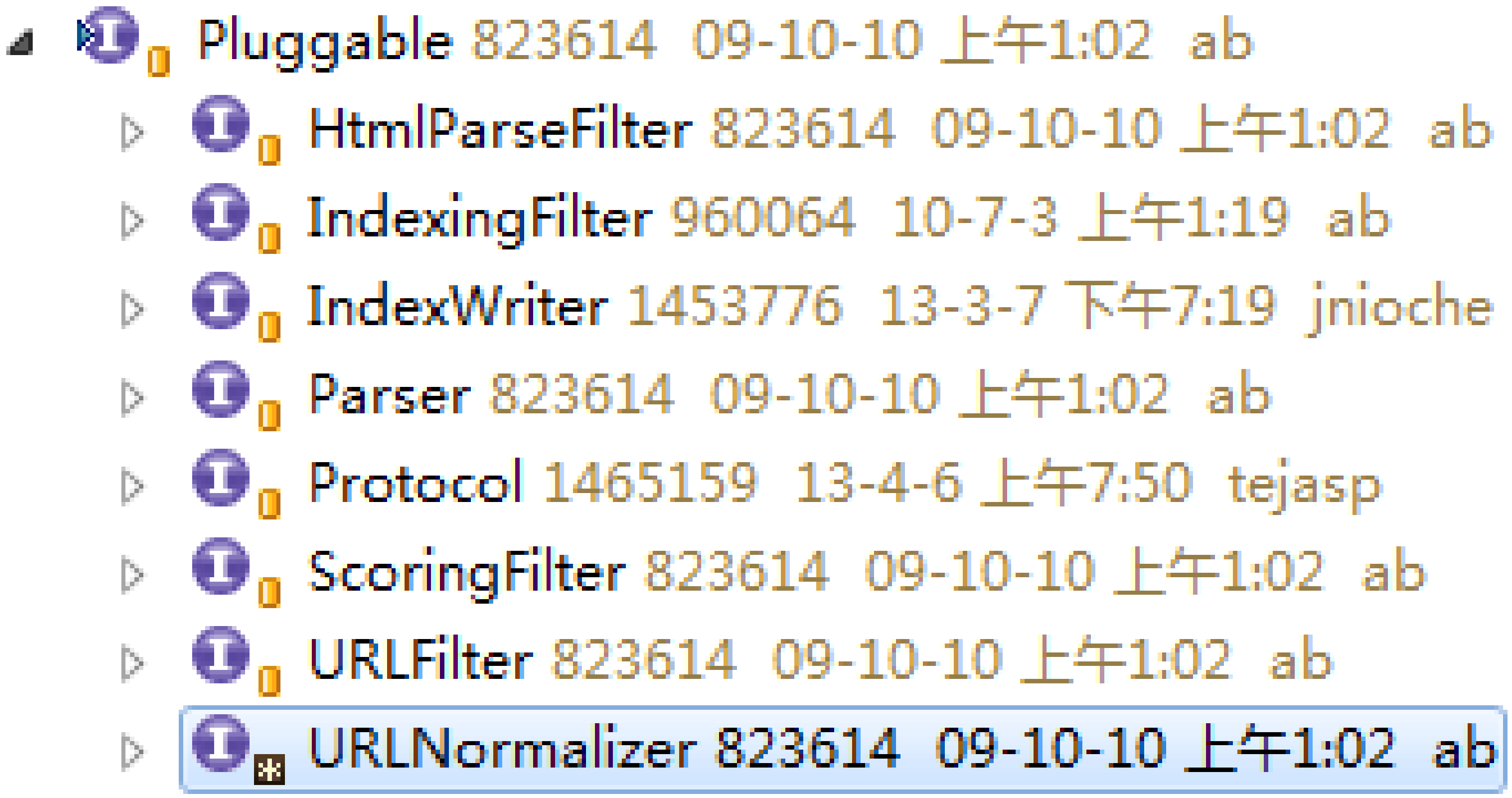
```
public void reduce(Text key, Iterator<SolrRecord> values,
    OutputCollector<Text, SolrRecord> output, Reporter reporter)
throws IOException {
  SolrRecord recordToKeep = new SolrRecord(values.next());
  while (values.hasNext()) {
    SolrRecord solrRecord = values.next();
    if (solrRecord.getBoost() > recordToKeep.getBoost() ||
        (solrRecord.getBoost() == recordToKeep.getBoost() &&
            solrRecord.getTstamp() > recordToKeep.getTstamp())) {
      updateRequest.deleteById(recordToKeep.id);
      recordToKeep = new SolrRecord(solrRecord);
    } else {
      updateRequest.deleteById(solrRecord.id);
    }
    numDeletes++;
```

8.9 插件机制

默认使用的插件如下图所示：

```
<name>plugin.includes</name>
<value>
    protocol-http| 多协议支持
    urlfilter-regex|挨个调用
    parse-(html|tika)|parse-html成功就不会调用pase-tika
    index-(basic|anchor)|挨个调用
    indexer-solr|挨个调用，可以同时写到多个索引服务器
    scoring-opic|挨个调用
    urlnormalizer-(pass|regex|basic)
</value>                                挨个调用
```

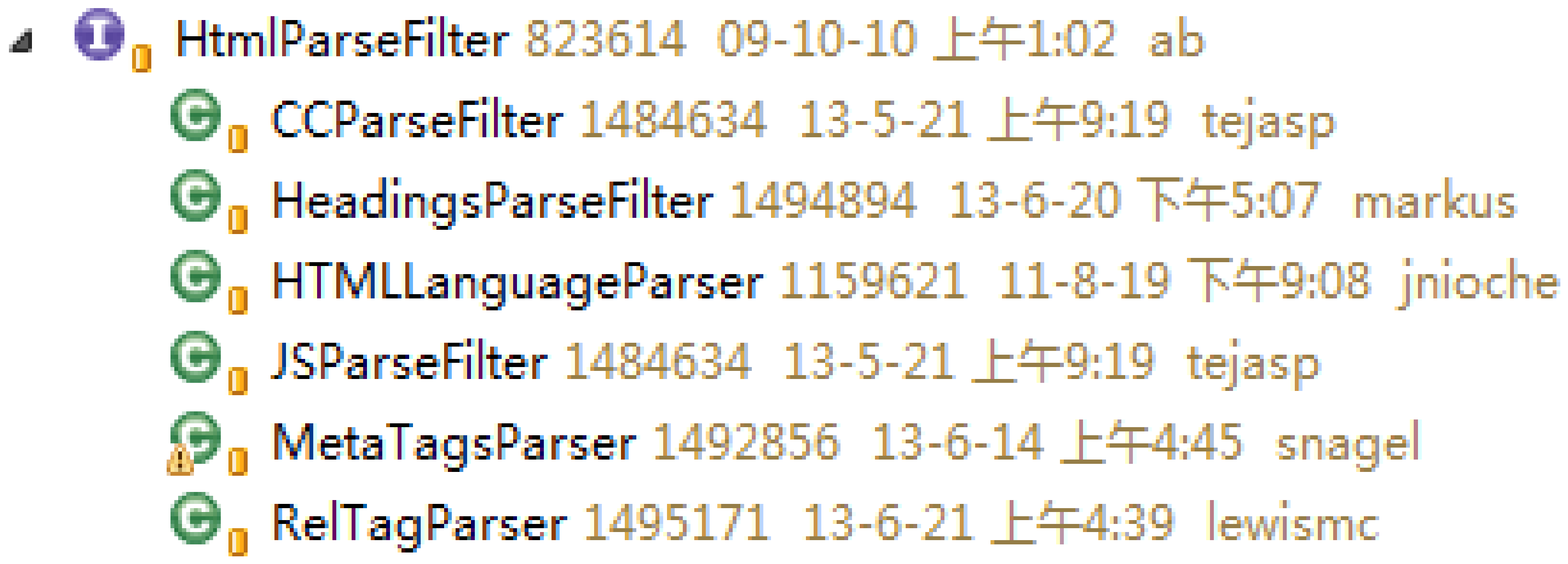
系统提供的可插接点：



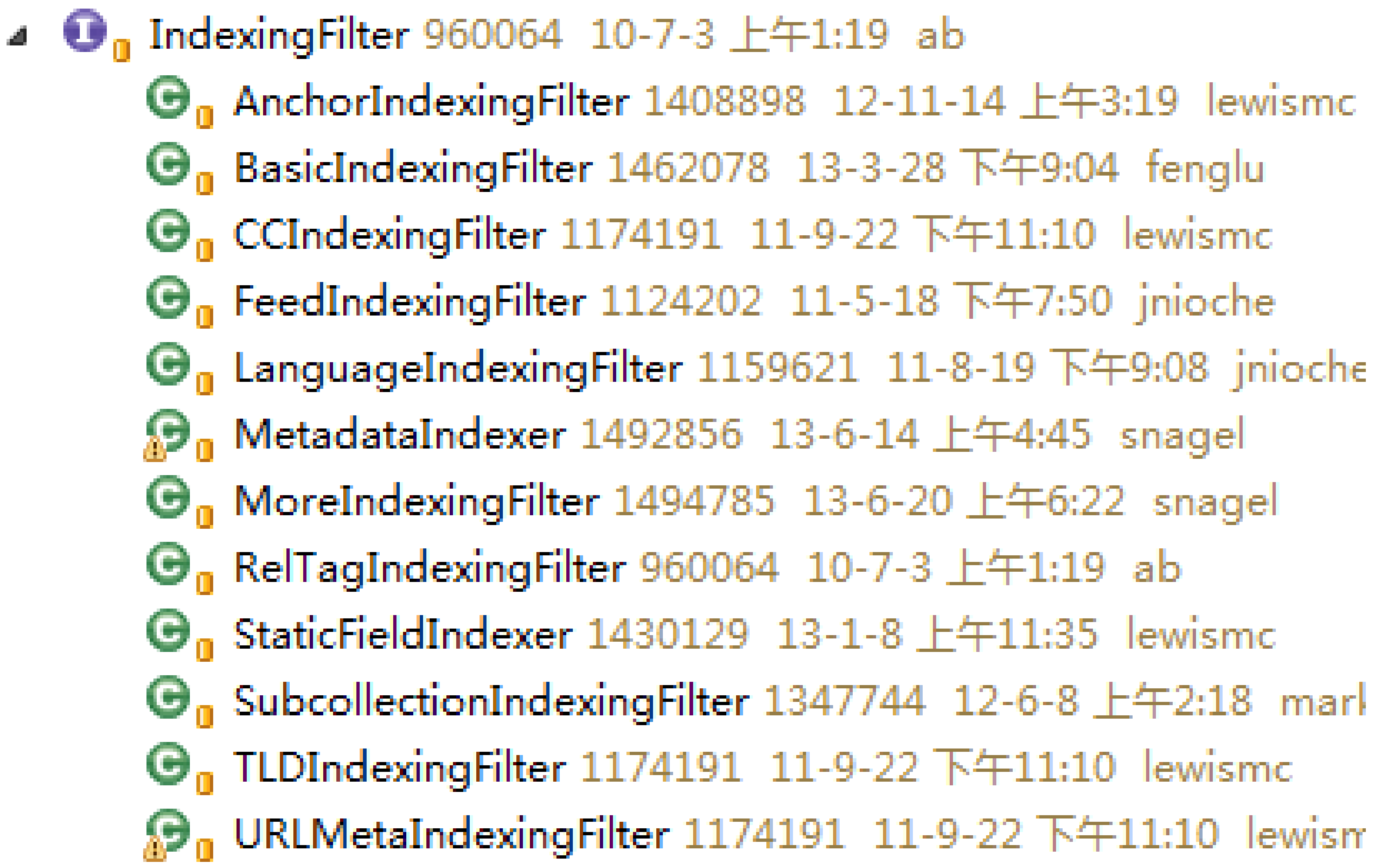
```
Pluggable 823614  09-10-10 上午1:02  ab
    HtmlParseFilter 823614  09-10-10 上午1:02  ab
    IndexingFilter 960064  10-7-3 上午1:19  ab
    IndexWriter 1453776  13-3-7 下午7:19  jnioche
    Parser 823614  09-10-10 上午1:02  ab
    Protocol 1465159  13-4-6 上午7:50  tejasp
    ScoringFilter 823614  09-10-10 上午1:02  ab
    URLFilter 823614  09-10-10 上午1:02  ab
    URLNormalizer 823614  09-10-10 上午1:02  ab
```

開源力量
Open Source Force

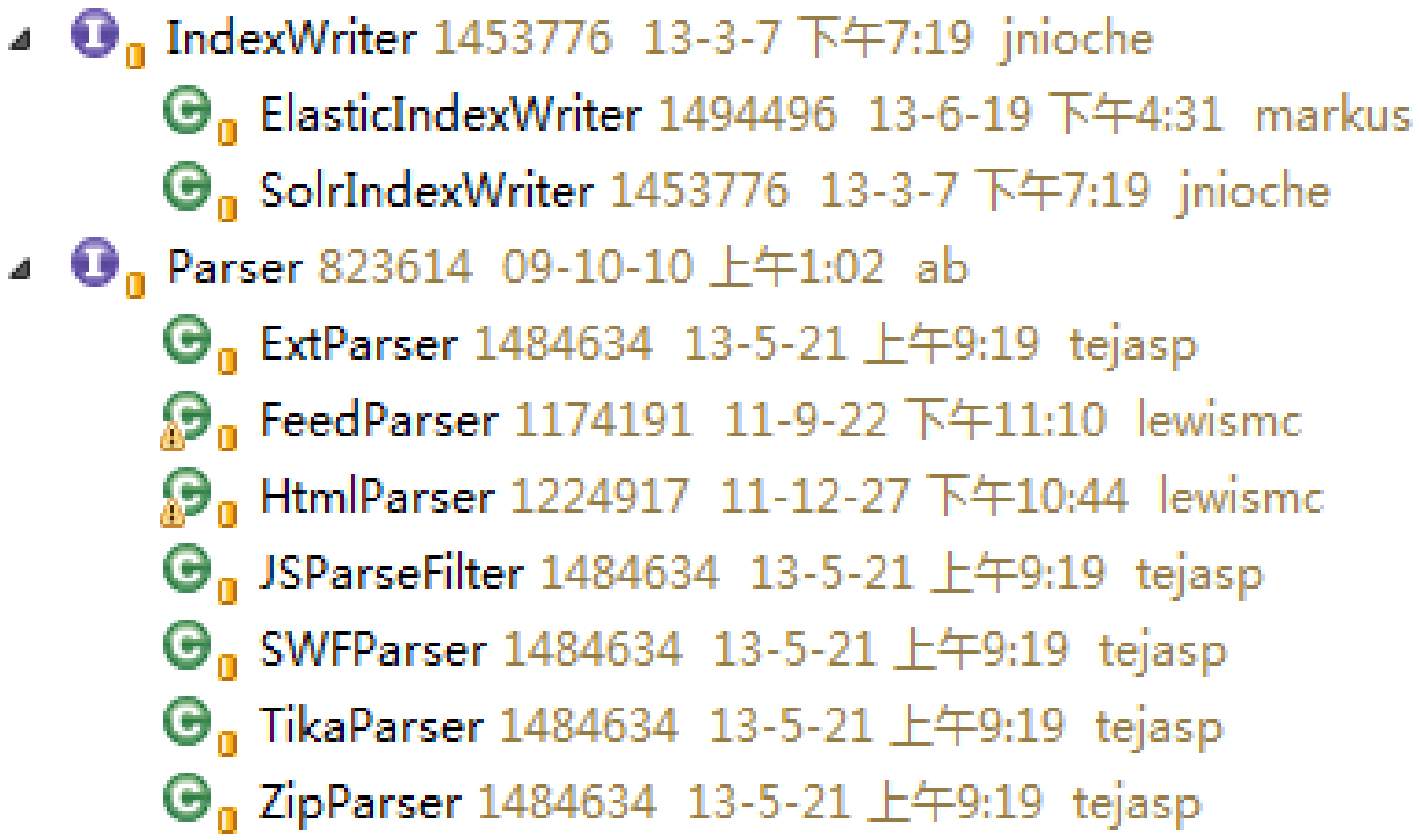
在parse-html和parse-tika插件中调用



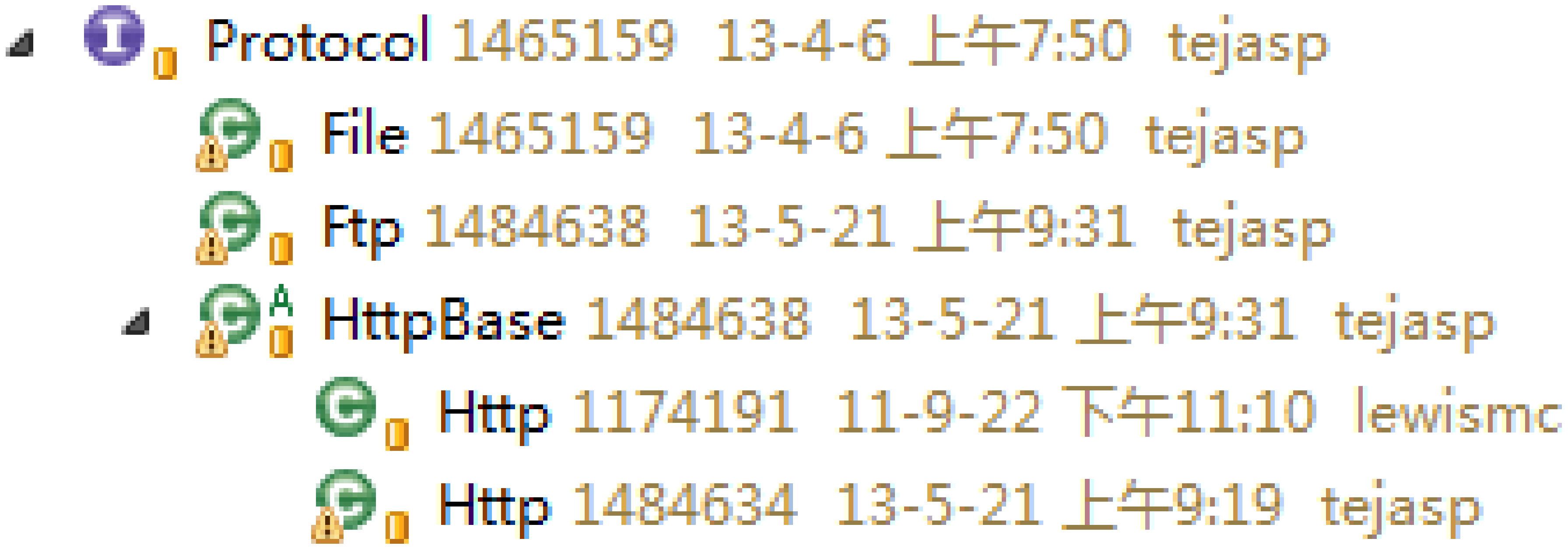
```
HtmlParseFilter 823614  09-10-10 上午1:02  ab
    CCParseFilter 1484634  13-5-21 上午9:19  tejasp
    HeadingsParseFilter 1494894  13-6-20 下午5:07  markus
    HTMLLanguageParser 1159621  11-8-19 下午9:08  jnioche
    JSParseFilter 1484634  13-5-21 上午9:19  tejasp
    MetaTagsParser 1492856  13-6-14 上午4:45  snagel
    RelTagParser 1495171  13-6-21 上午4:39  lewismc
```

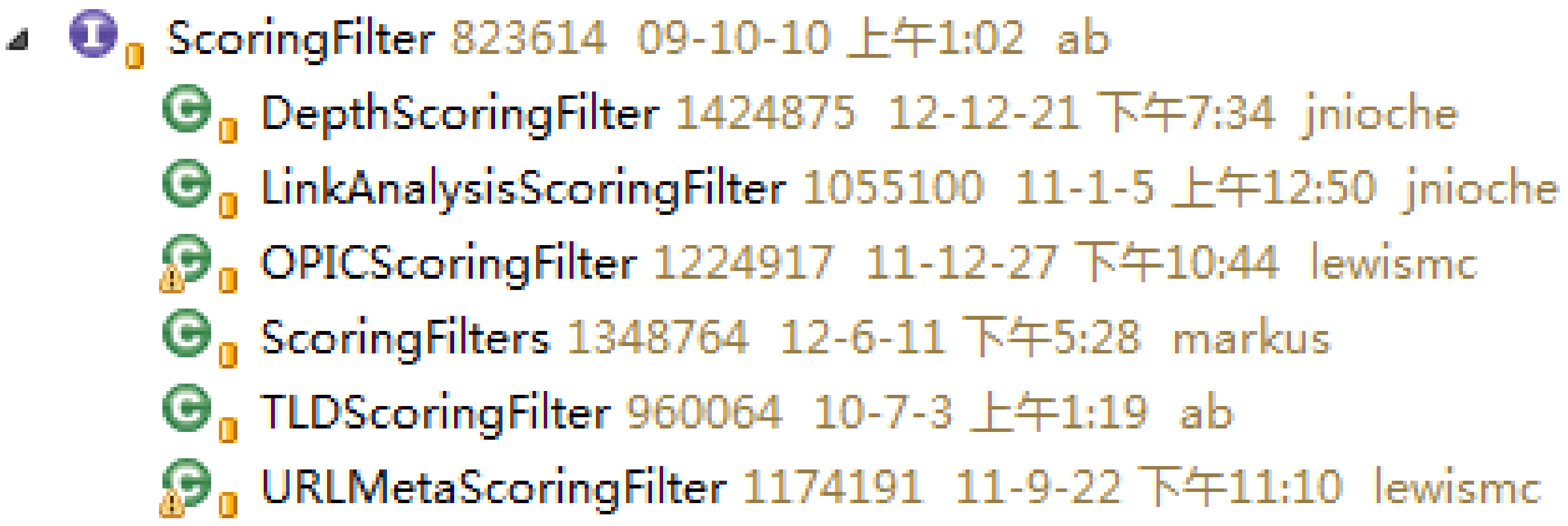


```
IndexingFilter 960064  10-7-3 上午1:19  ab
    AnchorIndexingFilter 1408898  12-11-14 上午3:19  lewismc
    BasicIndexingFilter 1462078  13-3-28 下午9:04  fenglu
    CCIndexingFilter 1174191  11-9-22 下午11:10  lewismc
    FeedIndexingFilter 1124202  11-5-18 下午7:50  jnioche
    LanguageIndexingFilter 1159621  11-8-19 下午9:08  jnioche
    MetadataIndexer 1492856  13-6-14 上午4:45  snagel
    MoreIndexingFilter 1494785  13-6-20 上午6:22  snagel
    RelTagIndexingFilter 960064  10-7-3 上午1:19  ab
    StaticFieldIndexer 1430129  13-1-8 上午11:35  lewismc
    SubcollectionIndexingFilter 1347744  12-6-8 上午2:18  marl
    TLDIndexingFilter 1174191  11-9-22 下午11:10  lewismc
    URLMetaIndexingFilter 1174191  11-9-22 下午11:10  lewism
```

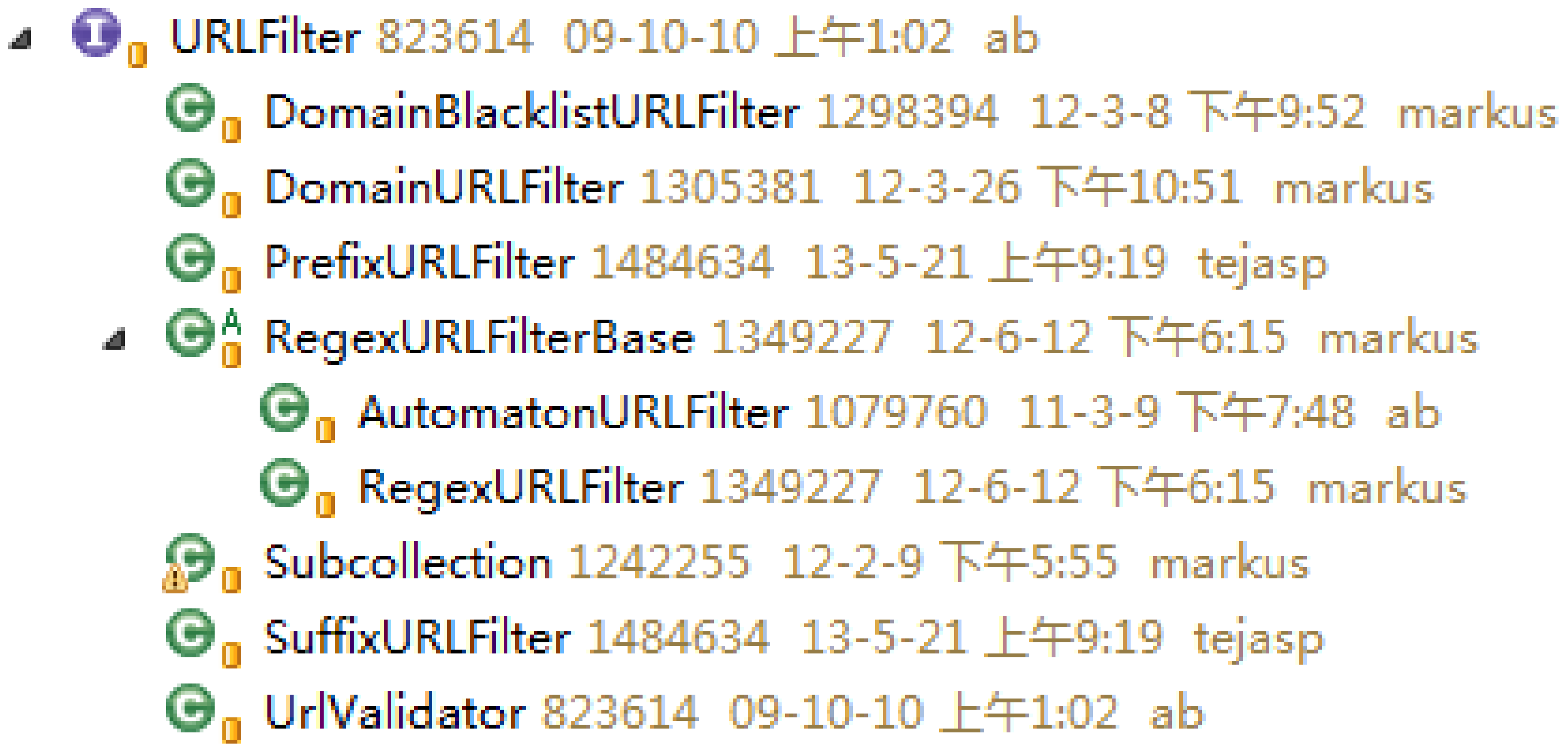


```
◢ IndexWriter 1453776  13-3-7 下午7:19  jnioche
    ElasticIndexWriter 1494496  13-6-19 下午4:31  markus
    SolrIndexWriter 1453776  13-3-7 下午7:19  jnioche
◢ Parser 823614  09-10-10 上午1:02  ab
    ExtParser 1484634  13-5-21 上午9:19  tejasp
    FeedParser 1174191  11-9-22 下午11:10  lewismc
    HtmlParser 1224917  11-12-27 下午10:44  lewismc
    JSParseFilter 1484634  13-5-21 上午9:19  tejasp
    SWFParser 1484634  13-5-21 上午9:19  tejasp
    TikaParser 1484634  13-5-21 上午9:19  tejasp
    ZipParser 1484634  13-5-21 上午9:19  tejasp
```

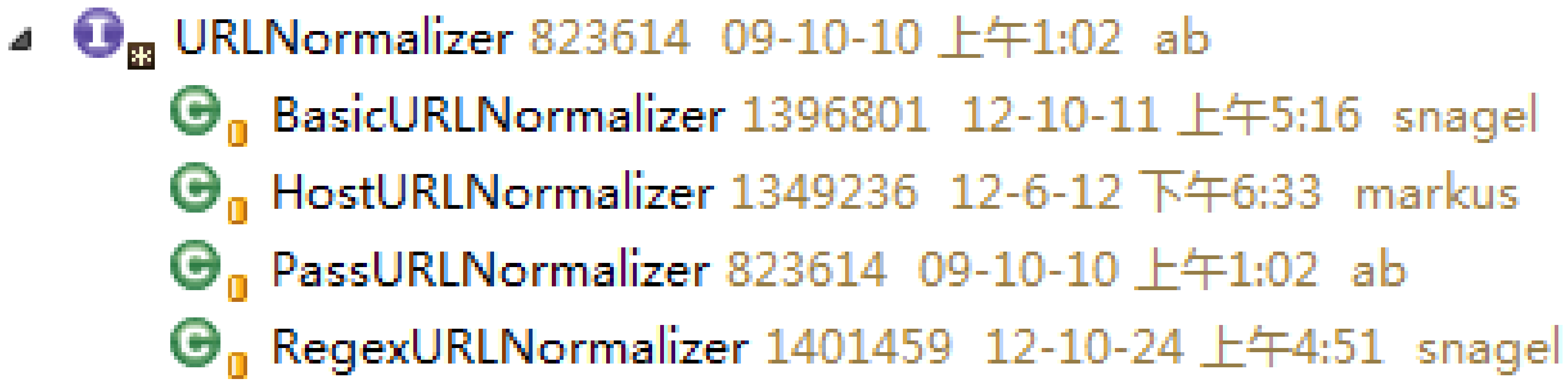

```
◢ Protocol 1465159  13-4-6 上午7:50  tejasp
      File 1465159  13-4-6 上午7:50  tejasp
      Ftp 1484638  13-5-21 上午9:31  tejasp
   ◢  HttpBase 1484638  13-5-21 上午9:31  tejasp
           Http 1174191  11-9-22 下午11:10  lewismc
           Http 1484634  13-5-21 上午9:19  tejasp
```



```
ScoringFilter 823614  09-10-10 上午1:02  ab
    DepthScoringFilter 1424875  12-12-21 下午7:34  jnioche
    LinkAnalysisScoringFilter 1055100  11-1-5 上午12:50  jnioche
    OPICScoringFilter 1224917  11-12-27 下午10:44  lewismc
    ScoringFilters 1348764  12-6-11 下午5:28  markus
    TLDScoringFilter 960064  10-7-3 上午1:19  ab
    URLMetaScoringFilter 1174191  11-9-22 下午11:10  lewismc
```

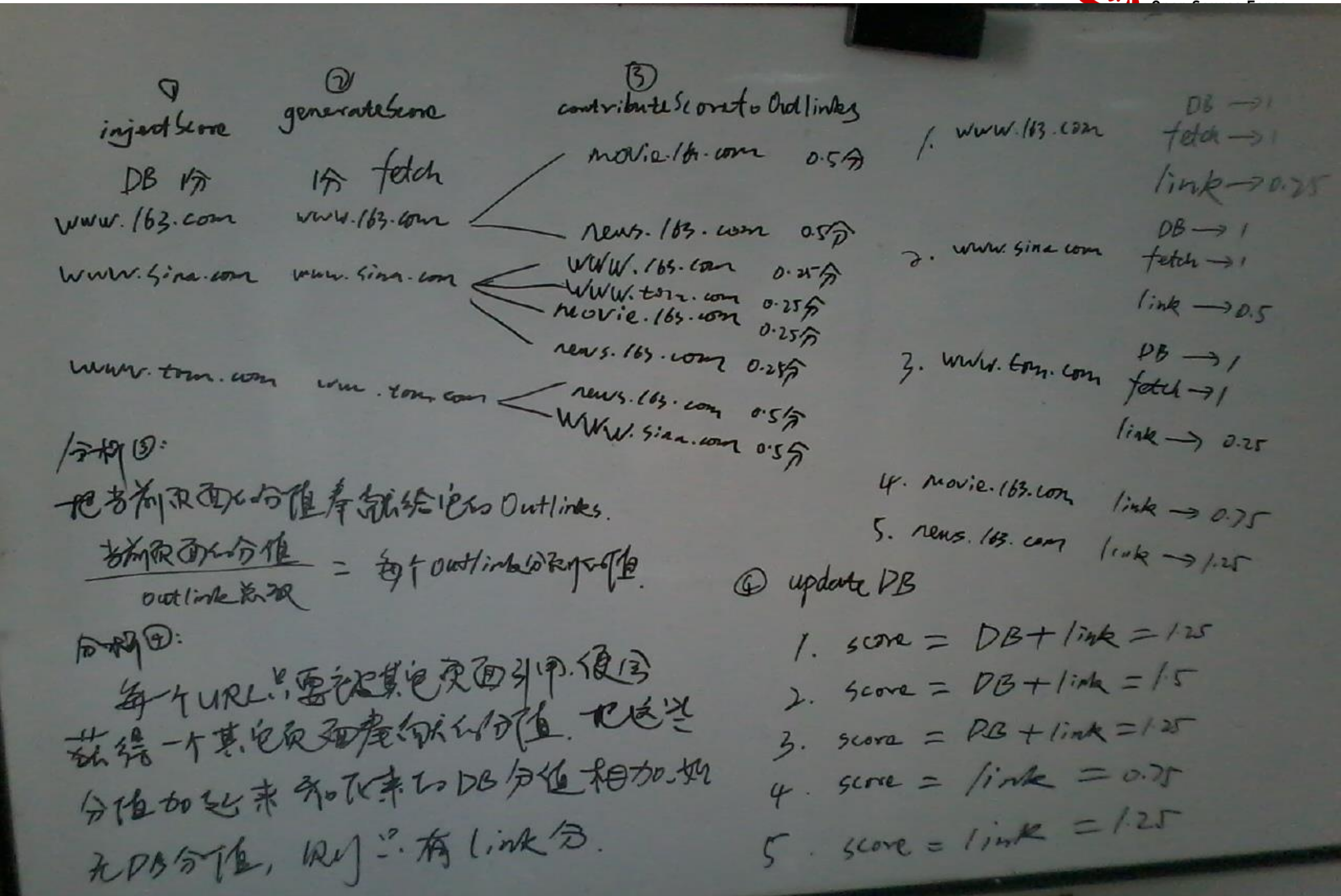


```
◢ URLFilter 823614  09-10-10 上午1:02  ab
    DomainBlacklistURLFilter 1298394  12-3-8 下午9:52  markus
    DomainURLFilter 1305381  12-3-26 下午10:51  markus
    PrefixURLFilter 1484634  13-5-21 上午9:19  tejasp
  ◢ RegexURLFilterBase 1349227  12-6-12 下午6:15  markus
        AutomatonURLFilter 1079760  11-3-9 下午7:48  ab
        RegexURLFilter 1349227  12-6-12 下午6:15  markus
    Subcollection 1242255  12-2-9 下午5:55  markus
    SuffixURLFilter 1484634  13-5-21 上午9:19  tejasp
    UrlValidator 823614  09-10-10 上午1:02  ab
```



```
URLNormalizer 823614  09-10-10 上午1:02  ab
    BasicURLNormalizer 1396801  12-10-11 上午5:16  snagel
    HostURLNormalizer 1349236  12-6-12 下午6:33  markus
    PassURLNormalizer 823614  09-10-10 上午1:02  ab
    RegexURLNormalizer 1401459  12-10-24 上午4:51  snagel
```

8.10 OPIC(On-line Page Importance Calculation)



① injectScore

DB 1分

www.163.com

www.sina.com

www.tom.com

② generateScore

1分 fetch

www.163.com

www.sina.com

www.tom.com

③ contributeScoretoOutlinks

www.163.com	movie.163.com	0.5分
	news.163.com	0.5分
www.sina.com	www.163.com	0.25分
	www.tom.com	0.25分
	movie.163.com	0.25分
	news.163.com	0.25分
www.tom.com	news.163.com	0.5分
	www.sina.com	0.5分

1. www.163.com	DB → 1	
	fetch → 1	
	link → 0.25	
2. www.sina.com	DB → 1	
	fetch → 1	
	link → 0.5	
3. www.tom.com	DB → 1	
	fetch → 1	
	link → 0.25	
4. movie.163.com	link → 0.75	
5. news.163.com	link → 1.25	

分析③:

把当前页面的分值平均分给它的Outlinks.

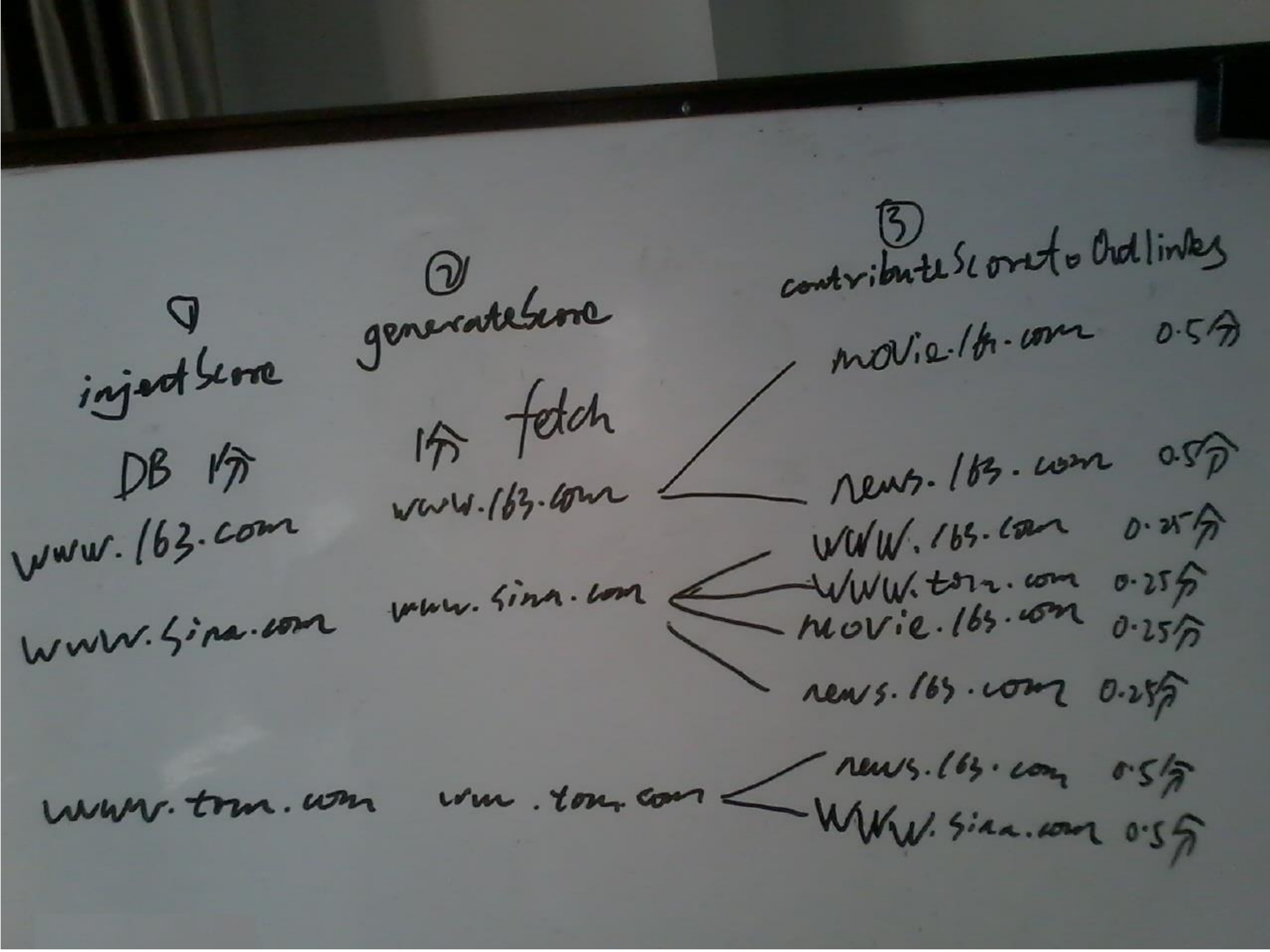
$$\frac{\text{当前页面的分值}}{\text{outlink总数}} = \text{每个outlink分配的分值}.$$

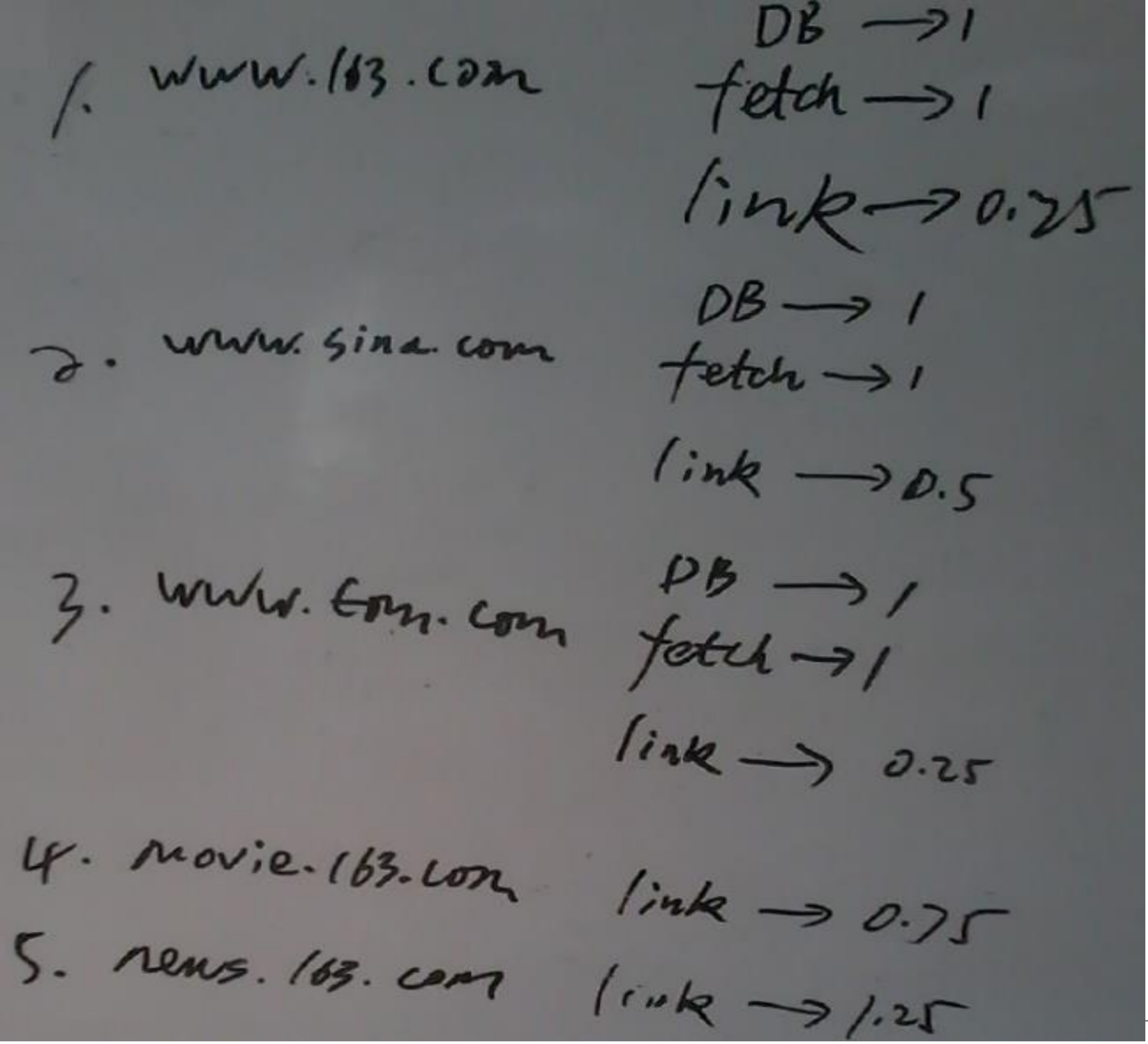
分析④:

每个URL只要在其它页面引用.便会获得一个其它页面平均分配的分值. 把这些分值加起来和原来的DB分值相加. 如无DB分值, 则只有link分.

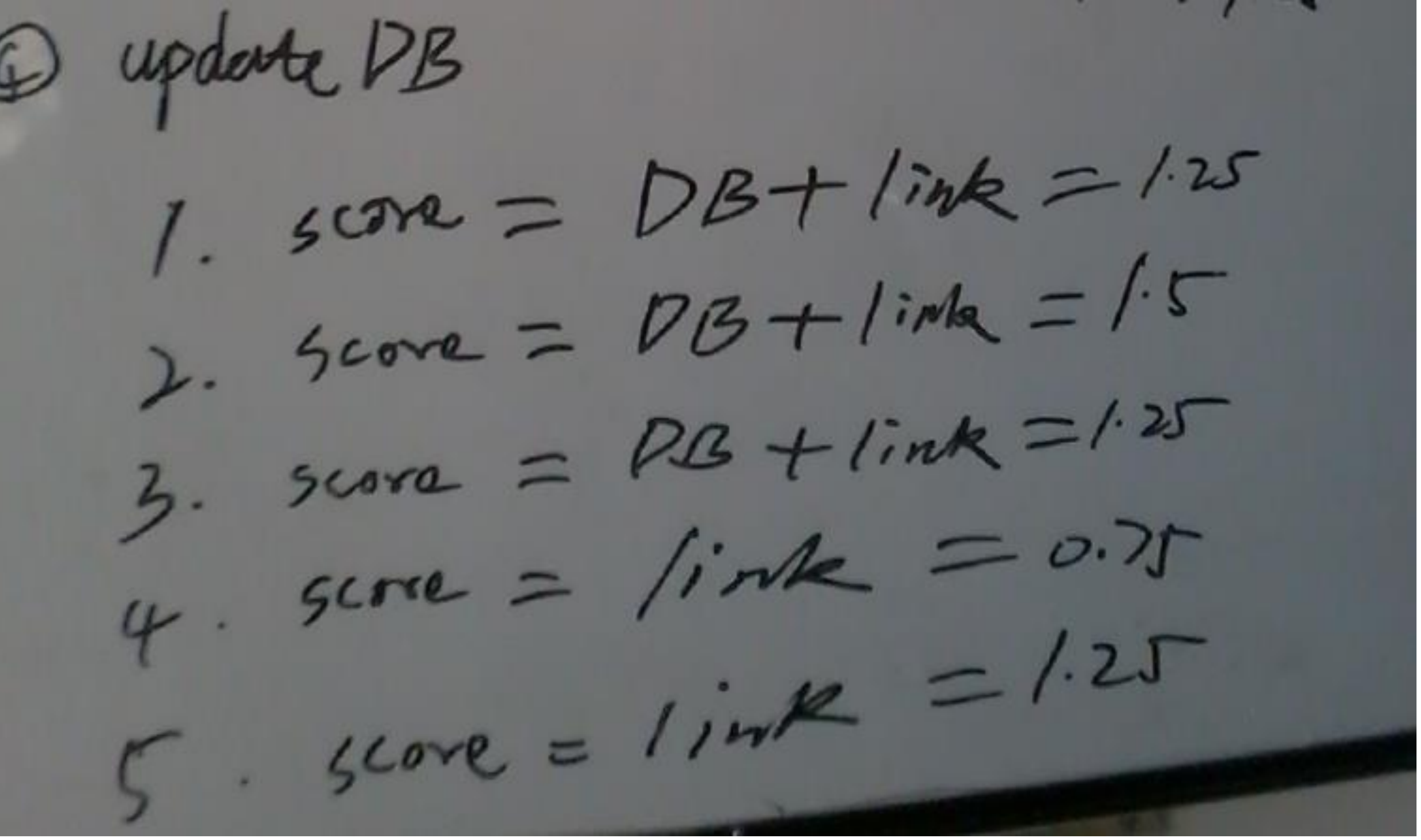
④ update DB

1. score = DB + link = 1.25
2. score = DB + link = 1.5
3. score = DB + link = 1.25
4. score = link = 0.75
5. score = link = 1.25


①
injectScore
DB 1分
www.163.com
www.sina.com
www.tom.com
②
generateScore
1分 fetch
www.163.com
www.sina.com
www .tom.com
③
contributeScoreToOutlinks
movie.163.com 0.5分
news.163.com 0.5分
www.163.com 0.25分
www.tom.com 0.25分
movie.163.com 0.25分
news.163.com 0.25分
news.163.com 0.5分
www.sina.com 0.5分



1. www.163.com
   - DB → 1
   - fetch → 1
   - link → 0.25
2. www.sina.com
   - DB → 1
   - fetch → 1
   - link → 0.5
3. www.tom.com
   - DB → 1
   - fetch → 1
   - link → 0.25
4. movie.163.com
   - link → 0.75
5. news.163.com
   - link → 1.25



④ update DB

1. score = DB + link = 1.25
2. score = DB + link = 1.5
3. score = DB + link = 1.25
4. score = link = 0.75
5. score = link = 1.25

分析③：

把当前页面的分值平均给它的Outlinks.

$$\frac{\text{当前页面的分值}}{\text{Outlink总数}} = \text{每个Outlink分配的值}.$$

分析④：

每一个URL只要被其它页面引用，便会获得一个其它页面平均下来的分值，把这些分值加起来和它本来的DB分值相加，如无DB分值，则只有link分.

9、Nutch的使用

一些具体的实践方法及演示

9.1、下载并解压eclipse（集成开发环境）

下载地址：http://www.eclipse.org/downloads/，下载Eclipse IDE for Java EE Developers

9.2、安装Subclipse插件（SVN客户端）

插件地址：http://subclipse.tigris.org/update_1.8.x，

9.3、安装IvyDE插件（下载依赖Jar）

插件地址：

http://www.apache.org/dist/ant/ivyde/updatesite/

开源力量
Open Source Force

9.4、签出代码

File > New > Project > SVN > 从SVN 检出项目

创建新的资源库位置 > URL：https://svn.apache.org/repos/asf/nutch/tags/release-1.7/ > 选中URL > Finish

弹出New Project向导，选择Java Project > Next，输入Project name：nutch1.7 > Finish

9.5、配置构建路径

在左部Package Explorer的 nutch1.7文件夹上单击右键 > Build Path > Configure Build Path...

> 选中**Source选项** > 选择src > Remove > Add Folder... > 选择src/bin, src/java, src/test 和 src/testresources *【对于插件，需要选中src/plugin目录下的每一个插件目录下的src/java，src/test文件夹】*

切换到**Libraries选项** >

Add Class Folder... > 选中nutch1.7/conf

Add Library... > IvyDE Managed Dependencies > Next > Main > Ivy File > Browse > ivy/ivy.xml > Finish *【如需要在开发环境中对插件进行开发调试，需要加入插件特定的ivy.xml】*

切换到**Order and Export选项**>

选中conf > Top > OK

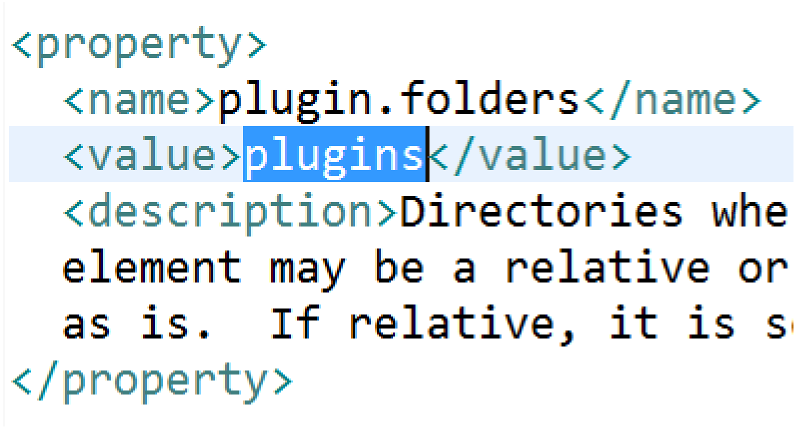
9.6、执行ANT

在左部Package Explorer的 nutch1.7文件夹下的 build.xml文件上单击右键 > Run As > Ant Build

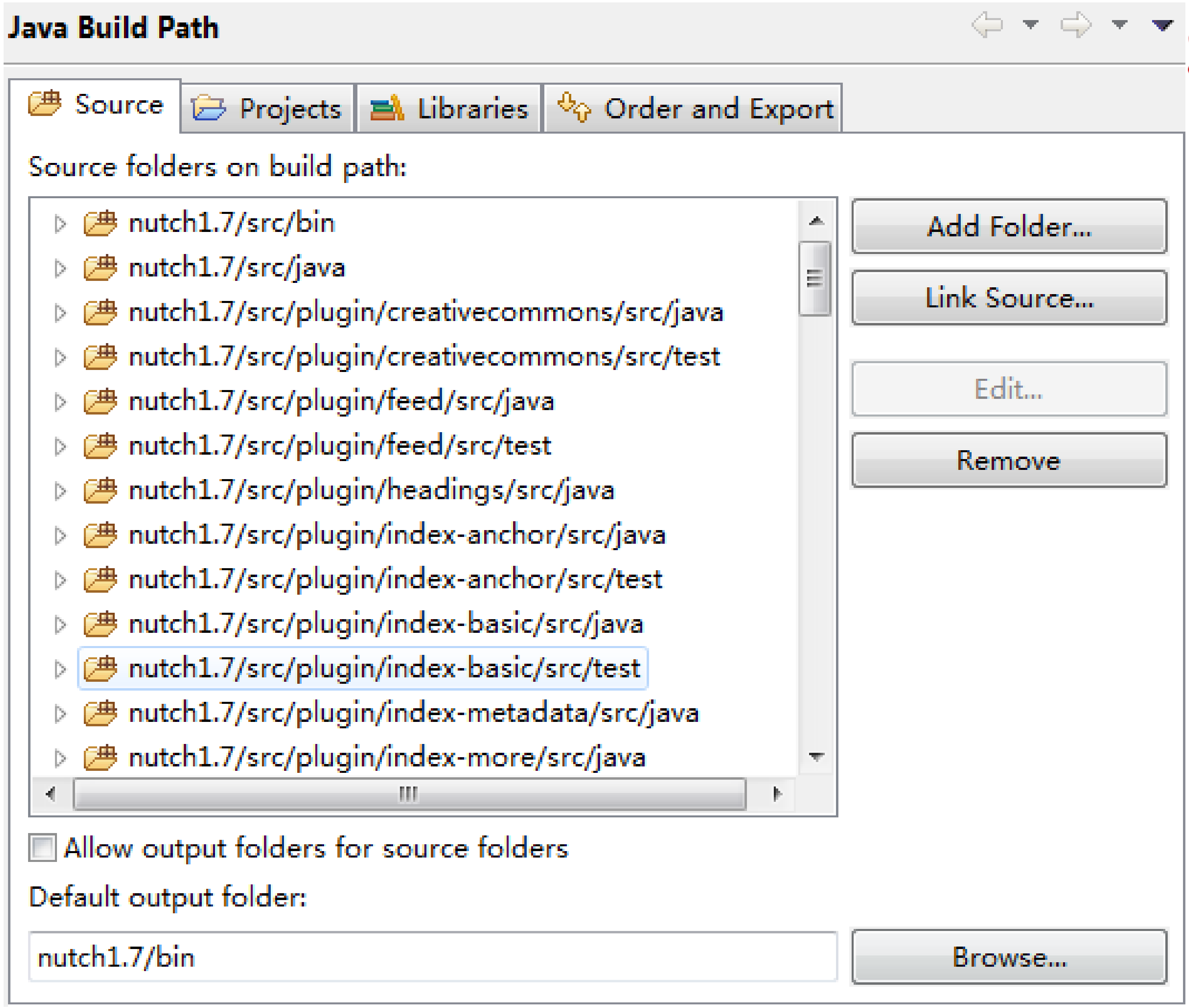
在左部Package Explorer的 nutch1.7文件夹上单击右键 > Refresh

在左部Package Explorer的 nutch1.7文件夹上单击右键 > Build Path > Configure Build Path... > 选中Libraries选项 > Add Class Folder... > 选中**build** > OK

为什么加入build?



```
<property>
  <name>plugin.folders</name>
  <value>plugins</value>
  <description>Directories whe
  element may be a relative or
  as is.  If relative, it is s
</property>
```



Java Build Path

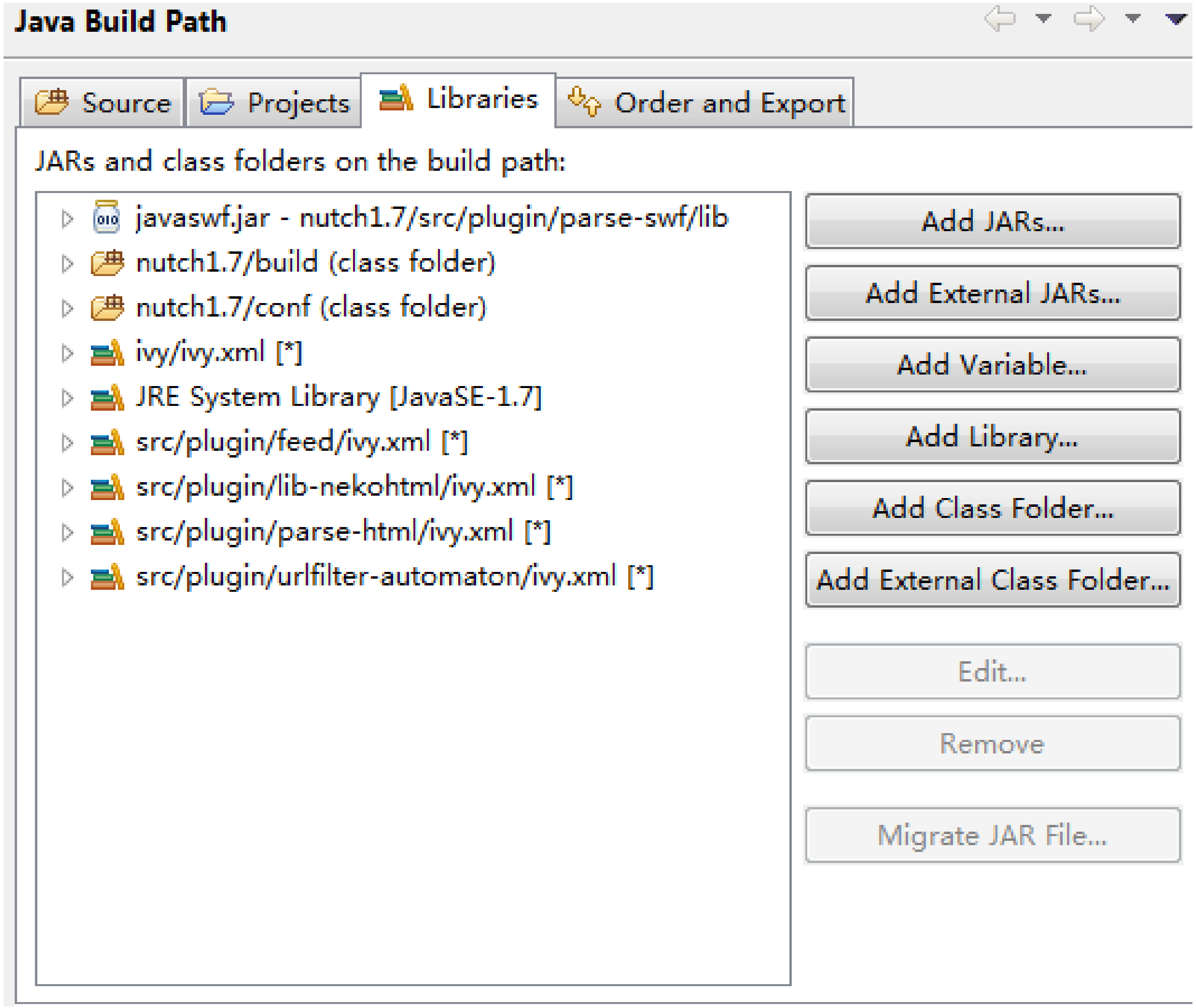
Source | Projects | Libraries | Order and Export

Source folders on build path:

```
nutch1.7/src/bin
nutch1.7/src/java
nutch1.7/src/plugin/creativecommons/src/java
nutch1.7/src/plugin/creativecommons/src/test
nutch1.7/src/plugin/feed/src/java
nutch1.7/src/plugin/feed/src/test
nutch1.7/src/plugin/headings/src/java
nutch1.7/src/plugin/index-anchor/src/java
nutch1.7/src/plugin/index-anchor/src/test
nutch1.7/src/plugin/index-basic/src/java
nutch1.7/src/plugin/index-basic/src/test
nutch1.7/src/plugin/index-metadata/src/java
nutch1.7/src/plugin/index-more/src/java
```

☐ Allow output folders for source folders

Default output folder: nutch1.7/bin



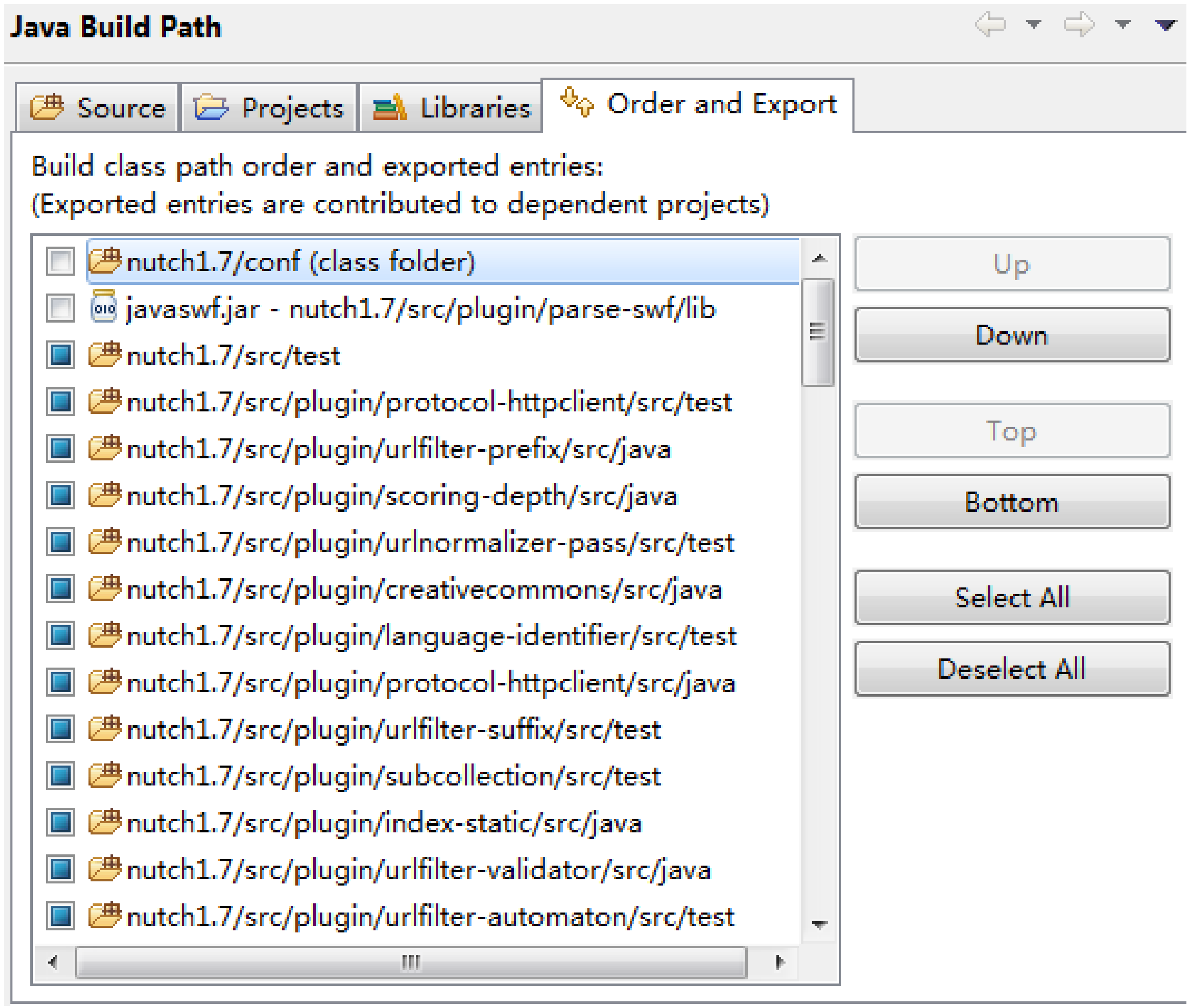
Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- javaswf.jar - nutch1.7/src/plugin/parse-swf/lib
- nutch1.7/build (class folder)
- nutch1.7/conf (class folder)
- ivy/ivy.xml [*]
- JRE System Library [JavaSE-1.7]
- src/plugin/feed/ivy.xml [*]
- src/plugin/lib-nekohtml/ivy.xml [*]
- src/plugin/parse-html/ivy.xml [*]
- src/plugin/urlfilter-automaton/ivy.xml [*]

开源力量
Open Source Force


Java Build Path

Source | Projects | Libraries | Order and Export

Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

- [ ] nutch1.7/conf (class folder)
- [ ] javaswf.jar - nutch1.7/src/plugin/parse-swf/lib
- [✓] nutch1.7/src/test
- [✓] nutch1.7/src/plugin/protocol-httpclient/src/test
- [✓] nutch1.7/src/plugin/urlfilter-prefix/src/java
- [✓] nutch1.7/src/plugin/scoring-depth/src/java
- [✓] nutch1.7/src/plugin/urlnormalizer-pass/src/test
- [✓] nutch1.7/src/plugin/creativecommons/src/java
- [✓] nutch1.7/src/plugin/language-identifier/src/test
- [✓] nutch1.7/src/plugin/protocol-httpclient/src/java
- [✓] nutch1.7/src/plugin/urlfilter-suffix/src/test
- [✓] nutch1.7/src/plugin/subcollection/src/test
- [✓] nutch1.7/src/plugin/index-static/src/java
- [✓] nutch1.7/src/plugin/urlfilter-validator/src/java
- [✓] nutch1.7/src/plugin/urlfilter-automaton/src/test

9.7、修改配置文件nutch-site.xml 和regex-urlfilter.txt

将nutch-site.xml.template改名为nutch-site.xml
将regex-urlfilter.txt.template改名为regex-urlfilter.txt
在左部Package Explorer的 nutch1.7文件夹上单击右键 > Refresh
将如下配置项加入文件nutch-site.xml：

```
<property>
  <name>http.agent.name</name>
  <value>nutch</value>
</property>
<property>
  <name>http.content.limit</name>
  <value>-1</value>
</property>
```

修改regex-urlfilter.txt，将

```
# accept anything else
+.
```

替换为：

```
+^http://([a-z0-9]*\.)*news.163.com/ (注意最后要有反斜杠且不能有空格)
-.
```

9.8、开发调试

在左部Package Explorer的 nutch1.7文件夹上单击右键 > New > Folder > Folder name: urls

在刚新建的urls目录下新建一个文本文件url，文本内容为：http://news.163.com/

打开src/java下的org.apache.nutch.crawl.Crawl.java类，单击右键Run As > Run Configurations > Arguments > 在Program arguments输入框中输入: urls -dir data -depth 3 > Run

在需要调试的地方打上断点Debug As > Java Applicaton

9.9、查看结果

查看segments目录：

打开src/java下的org.apache.nutch.segment.SegmentReader.java类

单击右键Run As > Java Applicaton，控制台会输出该命令的使用方法

单击右键Run As > Run Configurations > Arguments > 在Program arguments输入框中输入: -dump data/segments/* data/segments/dump

用文本编辑器打开文件data/segments/dump/dump查看segments中存储的信息

查看crawldb目录：

打开src/java下的
org.apache.nutch.crawl.CrawlDbReader.java类

单击右键Run As > Java Applicaton，控制台会输出该命令的使用方法

单击右键Run As > Run Configurations > Arguments > 在Program arguments输入框中输入: data/crawldb -stats

控制台会输出 crawldb统计信息

查看linkdb目录：

打开src/java下的org.apache.nutch.crawl.LinkDbReader.java类

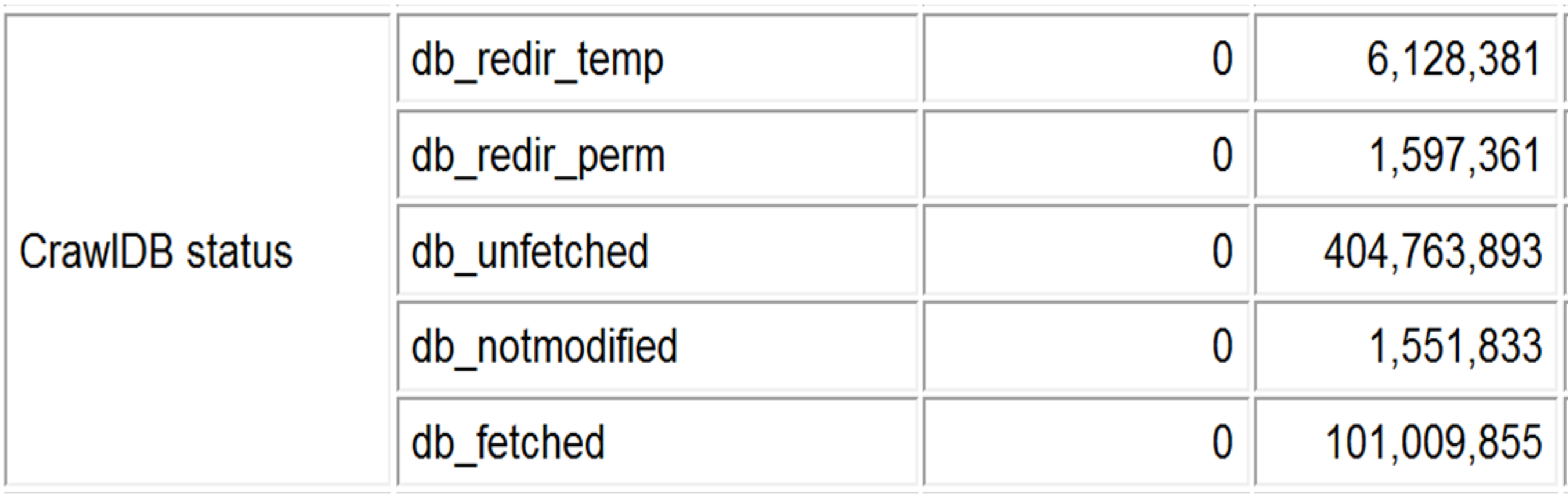
单击右键Run As > Java Applicaton，控制台会输出该命令的使用方法

单击右键Run As > Run Configurations > Arguments > 在Program arguments输入框中输入: data/linkdb -dump data/linkdb_dump

用文本编辑器打开文件data/linkdb_dump/part-00000查看linkdb中存储的信息

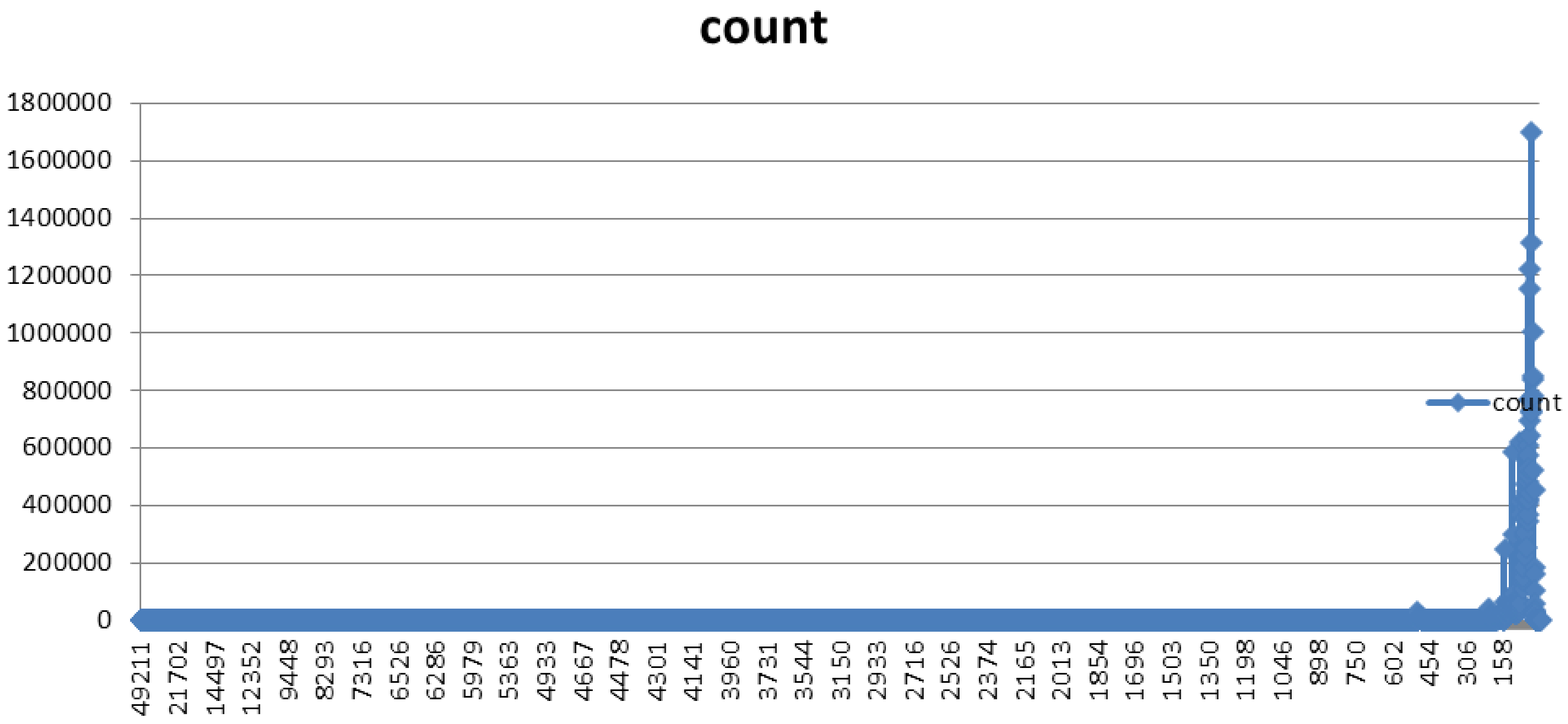
10、一些优化技巧

案例背景：IDC内部抓取托管网站

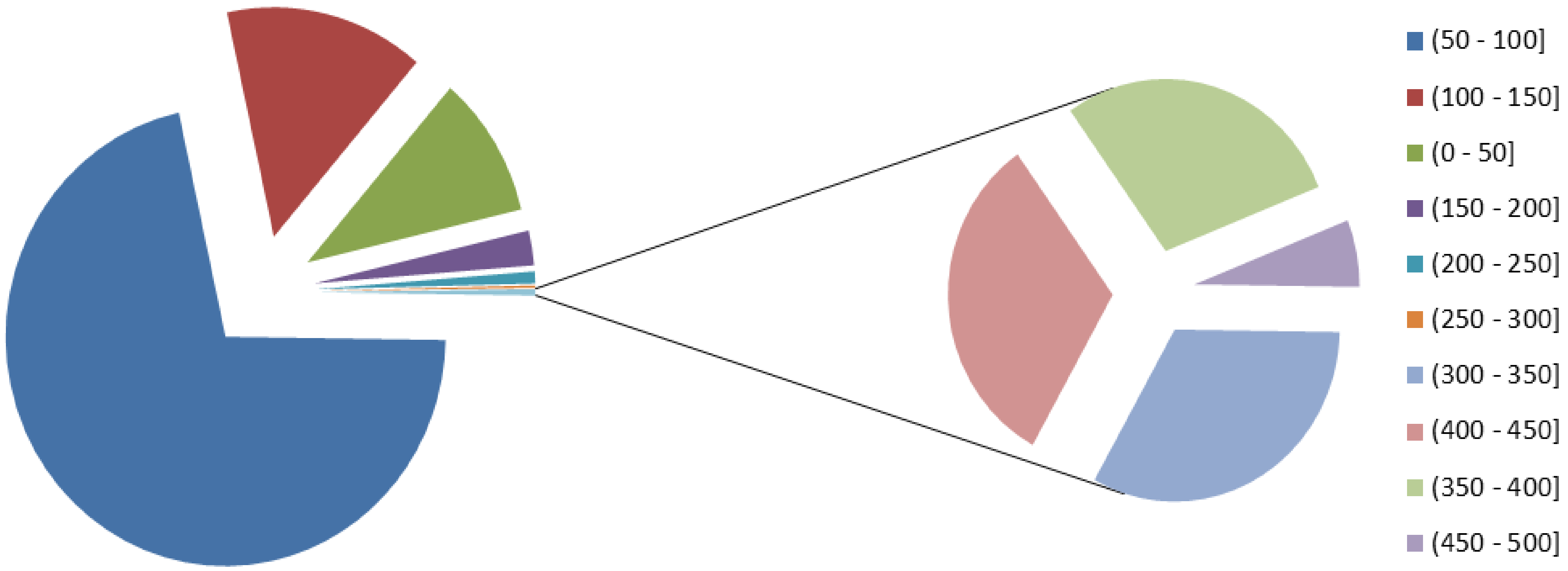


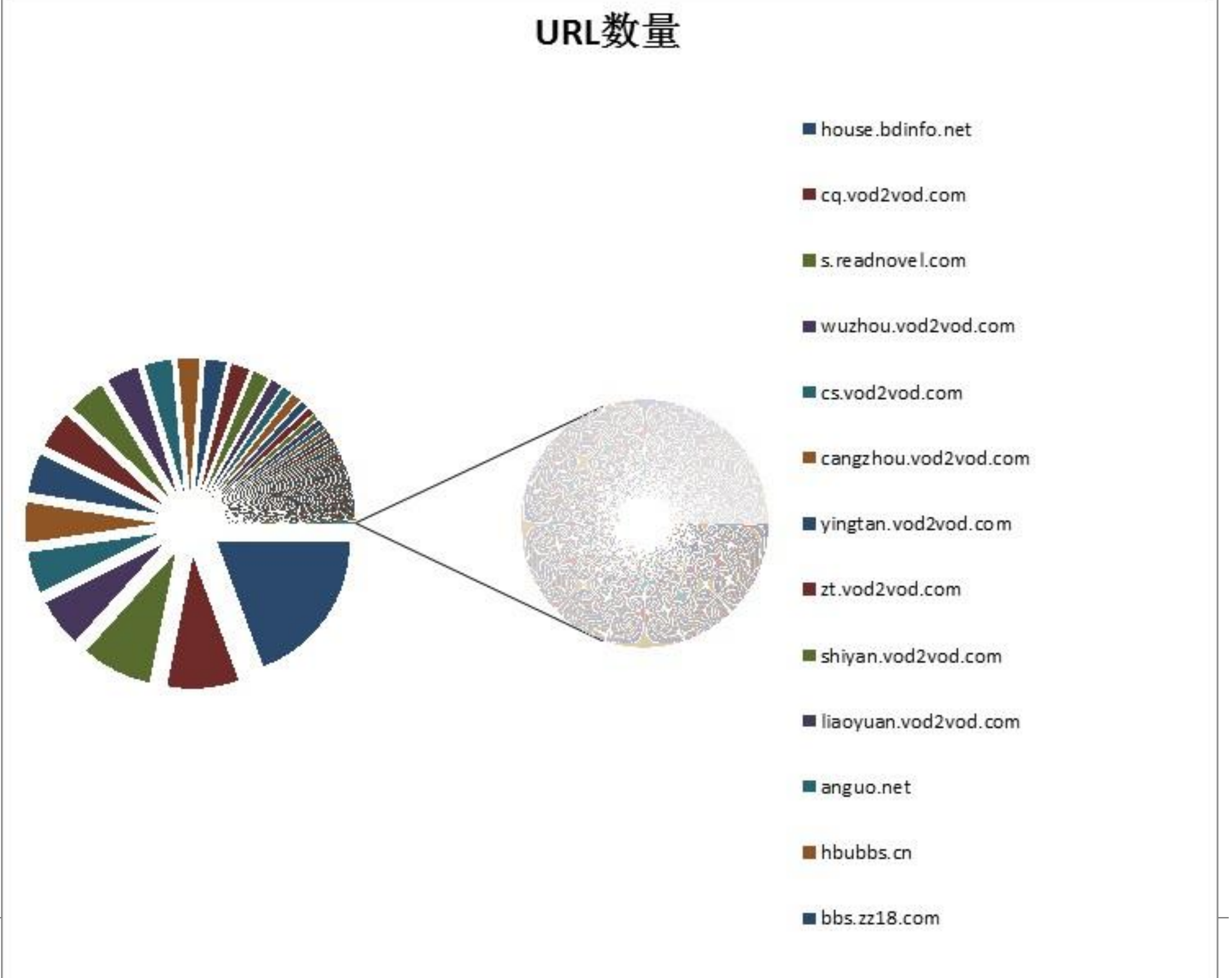
CrawlDB status	db_redir_temp	0	6,128,381
	db_redir_perm	0	1,597,361
	db_unfetched	0	404,763,893
	db_notmodified	0	1,551,833
	db_fetched	0	101,009,855

URL长度分布图


count
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
count
49211
21702
14497
12352
9448
8293
7316
6526
6286
5979
5363
4933
4667
4478
4301
4141
3960
3731
3544
3150
2933
2716
2526
2374
2165
2013
1854
1696
1503
1350
1198
1046
898
750
602
454
306
158

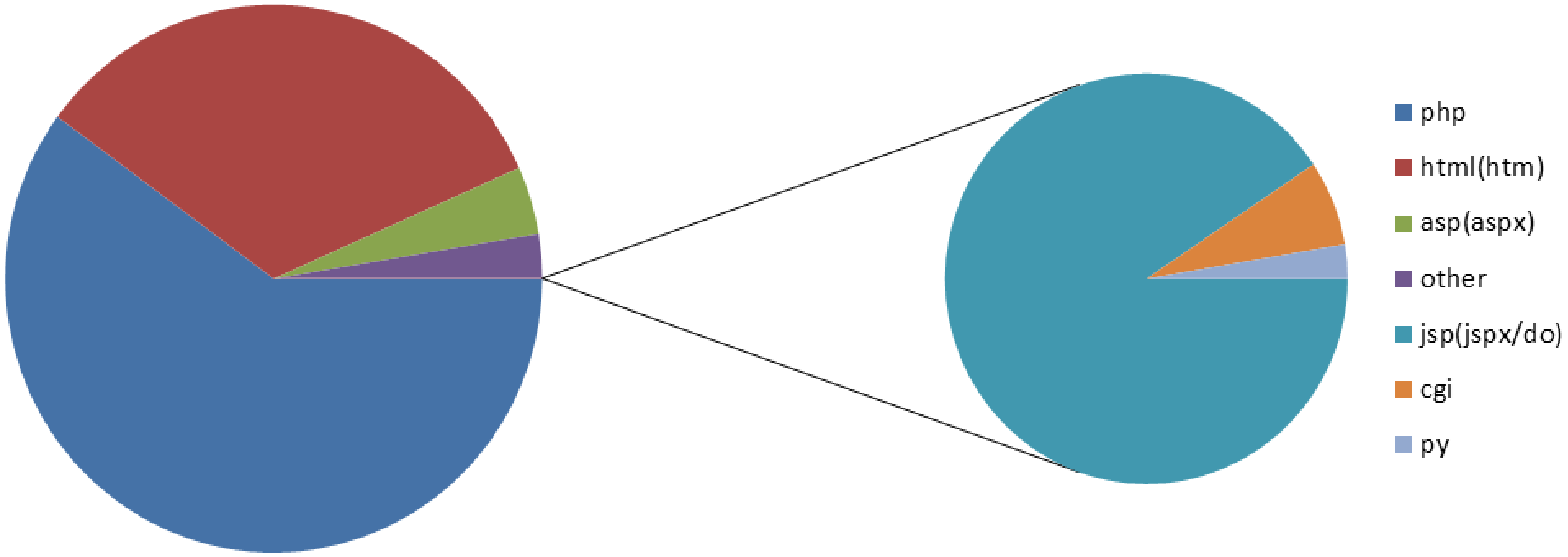
特定长度范围的URL数量




URL数量
house.bdinfo.net
cq.vod2vod.com
s.readnovel.com
wuzhou.vod2vod.com
cs.vod2vod.com
cangzhou.vod2vod.com
yingtan.vod2vod.com
zt.vod2vod.com
shiyan.vod2vod.com
liaoyuan.vod2vod.com
anguo.net
hbubbs.cn
bbs.zz18.com

开源力量 Open Source Force

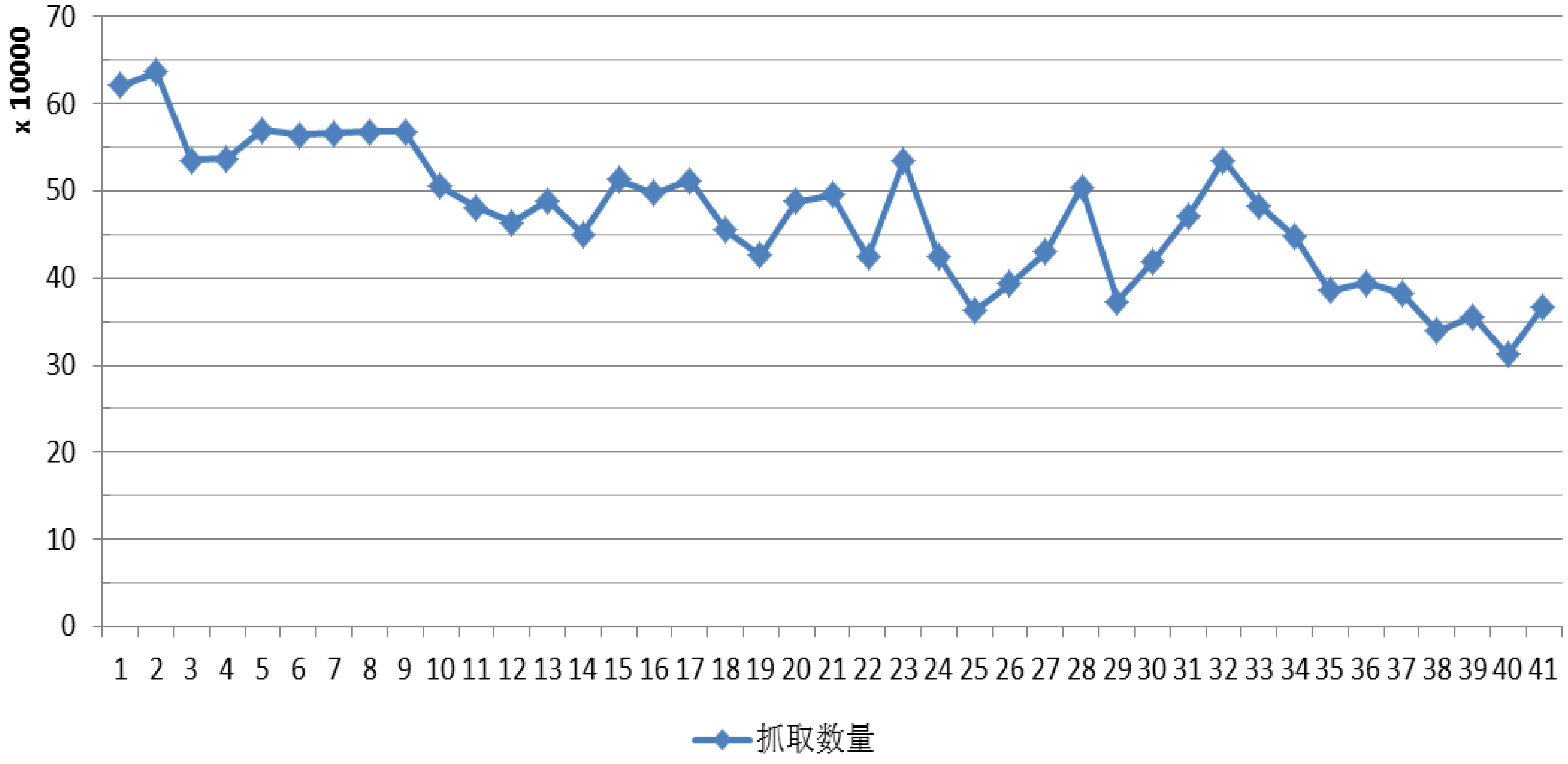
网页数量



下面是在5亿URL规模上的41轮抓取统计结果

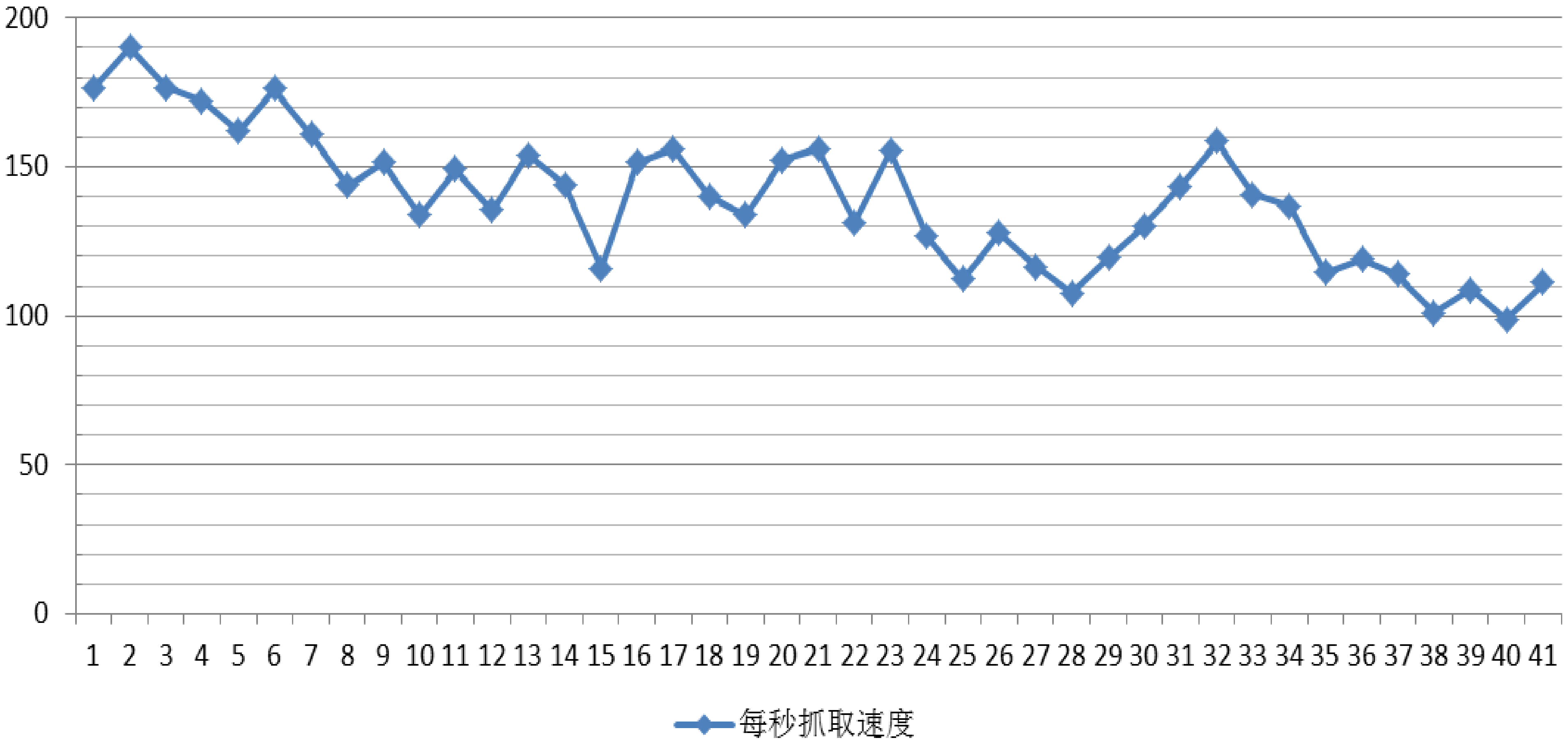
开源力量
Open Source Force

抓取数量

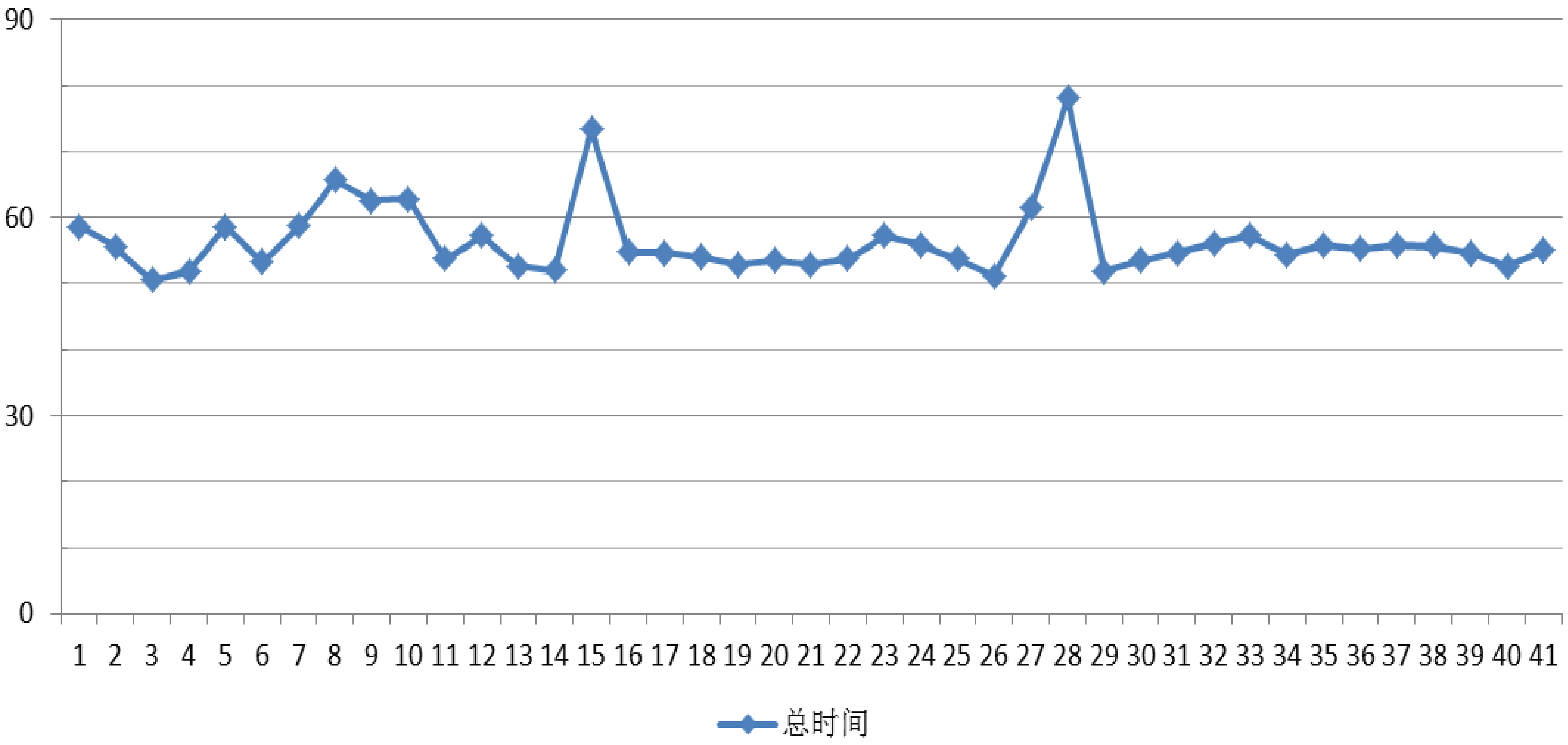


开源力量
Open Source Force

每秒抓取速度

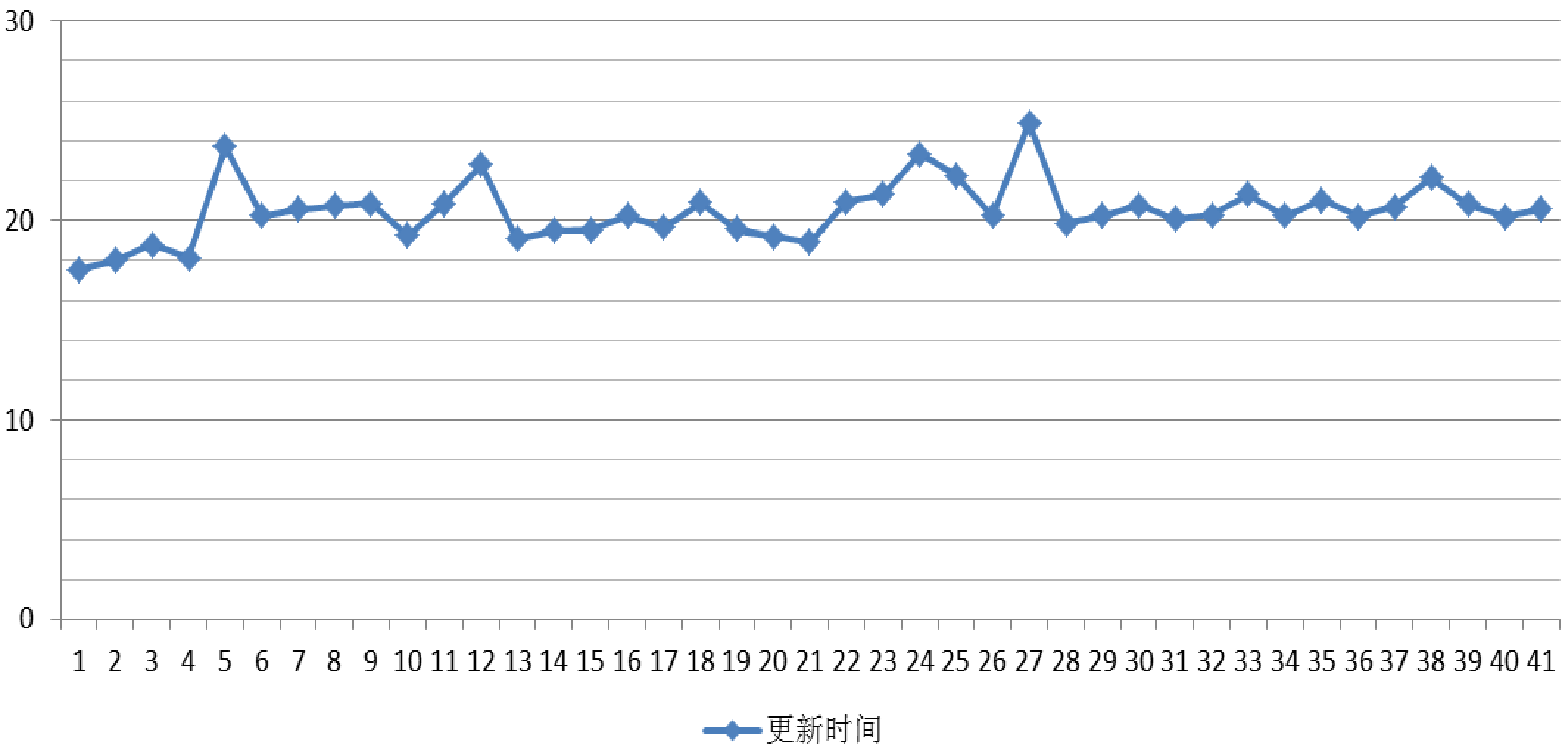


总时间

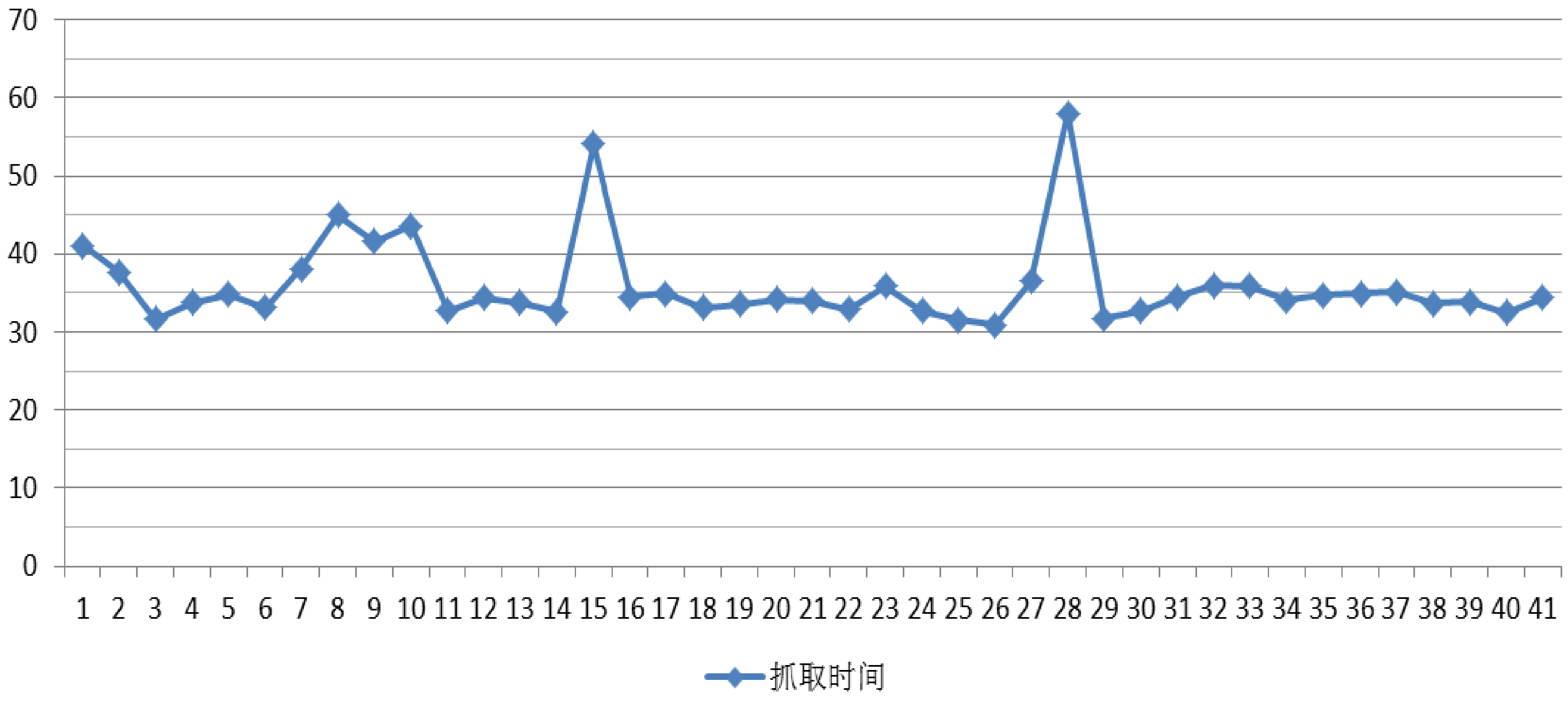


90
60
30
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
总时间

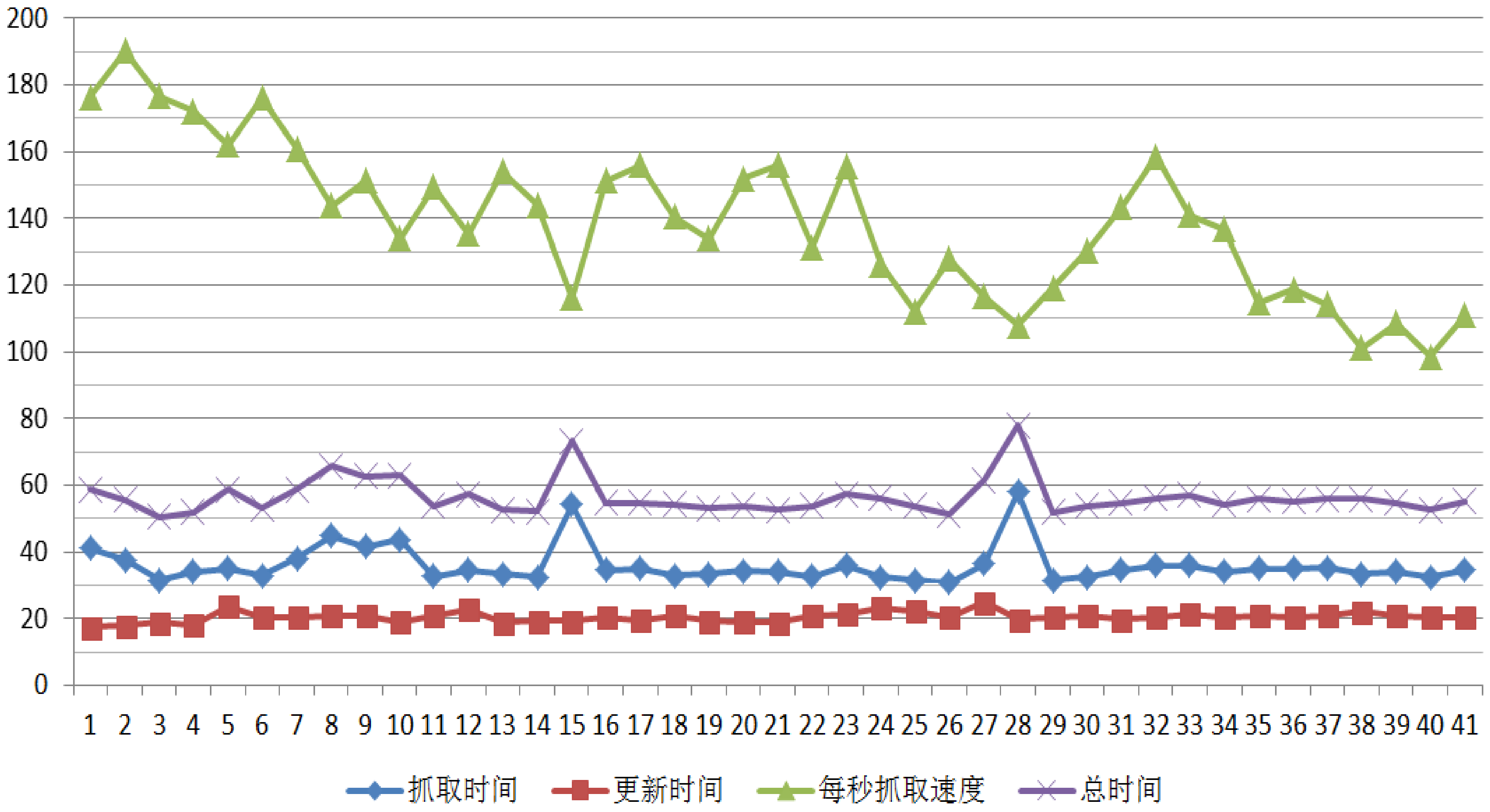
更新时间


30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
更新时间

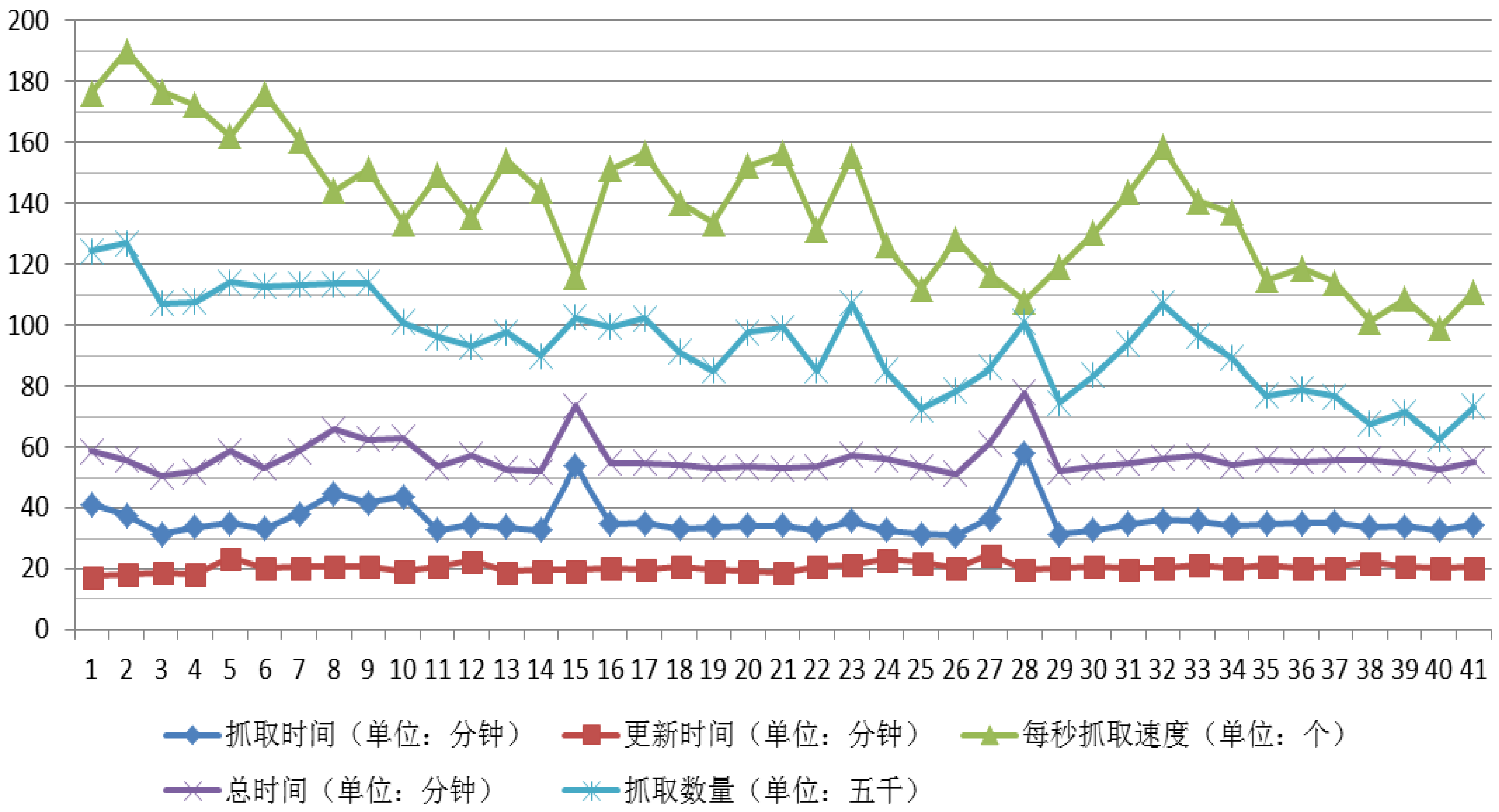
抓取时间


70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
抓取时间

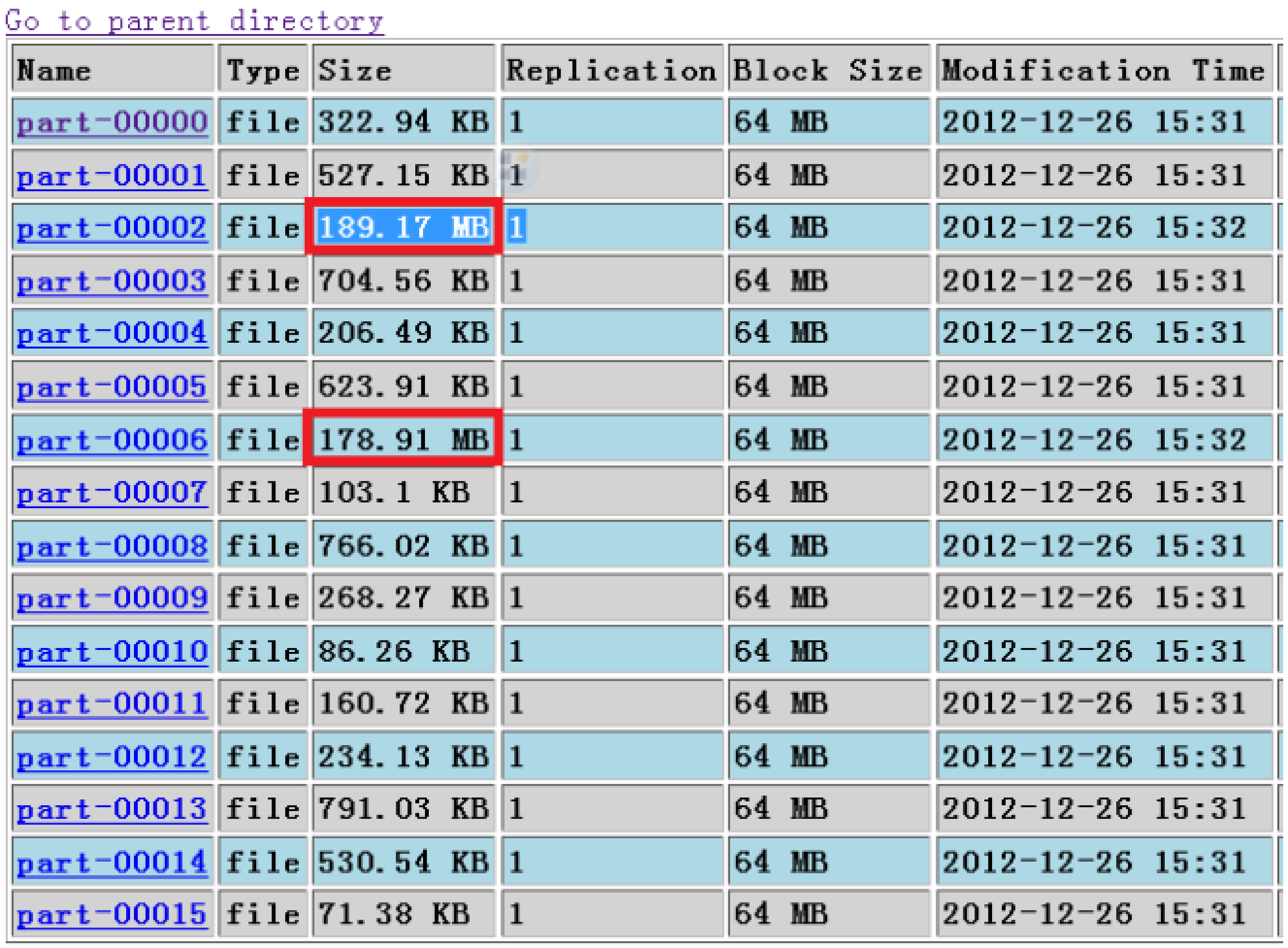
开源力量
Open Source Force



开源力量
Open Source Force

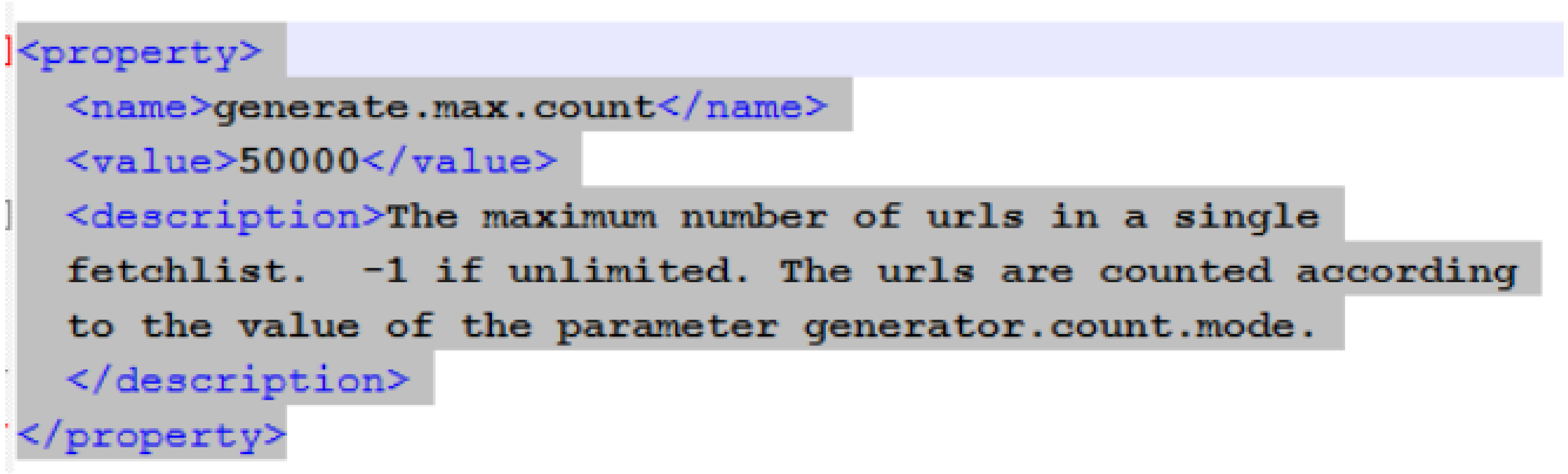


为什么会出现这么的差距？



Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time
part-00000	file	322.94 KB	1	64 MB	2012-12-26 15:31
part-00001	file	527.15 KB	1	64 MB	2012-12-26 15:31
part-00002	file	189.17 MB	1	64 MB	2012-12-26 15:32
part-00003	file	704.56 KB	1	64 MB	2012-12-26 15:31
part-00004	file	206.49 KB	1	64 MB	2012-12-26 15:31
part-00005	file	623.91 KB	1	64 MB	2012-12-26 15:31
part-00006	file	178.91 MB	1	64 MB	2012-12-26 15:32
part-00007	file	103.1 KB	1	64 MB	2012-12-26 15:31
part-00008	file	766.02 KB	1	64 MB	2012-12-26 15:31
part-00009	file	268.27 KB	1	64 MB	2012-12-26 15:31
part-00010	file	86.26 KB	1	64 MB	2012-12-26 15:31
part-00011	file	160.72 KB	1	64 MB	2012-12-26 15:31
part-00012	file	234.13 KB	1	64 MB	2012-12-26 15:31
part-00013	file	791.03 KB	1	64 MB	2012-12-26 15:31
part-00014	file	530.54 KB	1	64 MB	2012-12-26 15:31
part-00015	file	71.38 KB	1	64 MB	2012-12-26 15:31



```xml
<property>
  <name>generate.max.count</name>
  <value>50000</value>
  <description>The maximum number of urls in a single
  fetchlist.  -1 if unlimited. The urls are counted according
  to the value of the parameter generator.count.mode.
  </description>
</property>
```

单个站点网页太多，会拖慢整个Fetch的时间

参考资料：

- 官网：http://nutch.apache.org/
- Wiki ：http://wiki.apache.org/nutch/
- 书籍：Hadoop: The Definitive Guide, 3nd Edition
- 视频：http://yangshangchuan.iteye.com/blog/1837935

Questions

?

THANK YOU

开源力量
Open Source Force