# 下载

Apache 分发目录

<http://www.apache.org/dist/>

nutch 为顶级工程

<http://www.apache.org/dist/>nutch

# How to start

bin/crawl urls crawl -depth 3 -topN 5

# build & deploy

## 目录

|  |  |  |
| --- | --- | --- |
|  |  |  |
| build | 构建根目录， |  |
| build/classes | nutch core 代码编译输出 |  |
| build/lib | nutch core第三方依赖jar目录 |  |
| build/apache-nutch-{version}jar | nutch core 构建jar包 |  |
| build/apache-nutch-{version}.job | 一个jar包，能在hadoop上运行 |  |
|  |  |  |
| build/plugins | nutch 所有plugin发布预备目录 |  |
| build/plugins/{pluginName} | nutch plugin 发布目录 |  |
| build/plugins/{pluginName}.jar | nuch plugin jar包 |  |
| build/plugins/plugin.xml | nuch plugin 描述文件 |  |
| build/plugins/\*.jar | nutch plugin 依赖第三方jar文件 |  |
|  |  |  |
| build/{pluginName} | nutch plugin 编译输出根目录 |  |
| build/{pluginName}/classes | nutch plugin 编译结果目录 |  |
| build/{pluginName}/{pluginName.jar} | nutch plugin 构建jar 包 |  |
|  |  |  |
|  |  |  |
|  |  |  |

## build All

|  |  |  |
| --- | --- | --- |
| 目标 | 依赖 | 描述 |
| runtime | jar,job | 缺省构建目标，将构建的产copy到runtime目录 |
| jar | compile-core | build nutch jar |
| compile-core | init,resolve-default | compile java core files |
| init | ivy-init | 建立build目录，test的build目录 |
| ivy-init | ivy-probe-antlib, ivy-init-antlib | 初始化Ivy settings |
| ivy-probe-antlib |  | 检查ivy是否存在 |
| ivy-init-antlib | ivy-download |  |
| ivy-download | 直接调用ivy-download-unchecked |  |
| resolve-default | clean-lib，最终调用copy-libs | resolve and retrieve dependencies with ivy  将依赖提取到build.lib.dir |
| clean-lib |  | 清除build.lib.dir及test.build.lib.dir |
| copy-libs |  | 将lib目录copy到build.lib.dir |
| job | compile | 生成nutch.job jar文件，此文件会提交到hadoop运行。其包含nutch所有的package及plugin package  (将build.lib.dir 和 build.plugins 及build.classes打包) |
| compile | compile-core, compile-plugins |  |
| compile-plugins | init, resolve-default  之后调用<ant dir="src/plugin" target="deploy" inheritAll="false"/> | 构建插件 |
|  |  |  |
|  |  |  |

Compile-core classPath指定${build.dir}/lib中所有jar

Ivy 使用示例： resolve 依赖和clear cache在 setting之后执行

<target name="init">

<ivy:settings file="../../ivysettings.xml"/>

</target>

<target name="resolve" depends="init">

<ivy:resolve/>

</target>

<target name="clean" description="Cleanup build directory">

<delete dir="${build.dir}"/>

</target>

<target name="clean-all" depends="init,clean" description="Clean and purge caches">

<!-- Purge the ivy cache -->

<ivy:cleancache/>

</target>

## Build plugin

执行完deploy后，所有插件均会在build目录中准备好， plugin会由根build中的runtime目标统一拷贝到runtime的local目录

|  |  |  |
| --- | --- | --- |
| 目标 | 依赖 | 描述 |
| deploy | 构建发布所有插件 | plugin/build.xml |
| 所有插件构建均依赖此文件 |  | plugin/build-plugin.xml |
| deploy | jar, deps-test | 拷贝jar及plugin.xml到${deploy.dir} |
| jar | compile | ${build.classes}打jar包 |
| compile | init,deps-jar, resolve-default | 编译源码到${build.classes} |
| init |  | 生成build目录 |
| resolve-default |  | 解析、下载依赖包 |
|  |  |  |
|  |  |  |

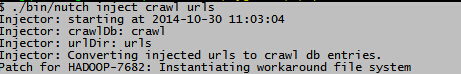
# 命令

如何调试：

在cygwin环境下，建立urls 目录，写入种子url。

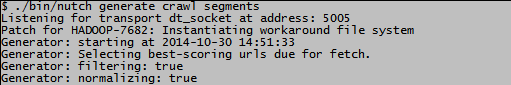
Inject

./bin/nutch inject crawl urls



Generate

./bin/nutch generate crawl segments



Fetch

./bin/nutch fetch segments/YYYYMMDDhhmmss

$ ./bin/nutch fetch segments/20141031163926 -threads 1

使用一个线程方便调试。

Fetch的输入目录需指定到 segment/YYYYMMDDHHmmss级别

C:\Users\wangqiaodong581\AppData\Roaming\Tencent\Users\2242703215\QQ\WinTemp\RichOle\I[CP39Z$KG~`CZAV_2IL(A1.png

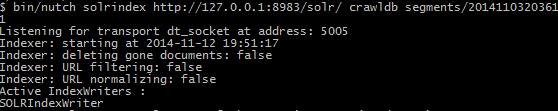
Parse

Parse插件。所有插件的加载均由nutch-default.xml中

./bin/nutch parse segments/ YYYYMMDDhhmmss

C:\Users\wangqiaodong581\AppData\Roaming\Tencent\Users\2242703215\QQ\WinTemp\RichOle\Y9ZW0OGKPZ6IMAQSD}R]ZCC.png

./bin/nutch index –D solr.server.url=http://127.0.0.1:8983/solr/ crawldb segments/YYYYMMDDHHmmss



Updatedb

./bin/nutch update crawl segments/ YYYYMMDDhhmmss

将segments/ YYYYMMDDhhmmss下面的crawl\_fetch、crawl\_parse 和 crawl/current作为输入目录，合并产生输出。

## Sina 抓取

在BigP抓取中由于sina微博配置了robot.txt

实践中的几点修改

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | regex-urlfilter | 配置接收和拒绝的url |
|  |  |  |
| fetch | http.redirect.max | 默认中设置的是0，导致重定向需要在下个周期中解析&Parse |
| inject | http.agent.name | 必须指定一个值 |
| fs.file.impl | com.conga.services.hadoop.patch.HADOOP\_7682.WinLocalFileSystem | 由于cgywin的bug所致 |
| NUTCH\_OPTS | export NUTCH\_OPTS="-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=5005" | 调试需要 |
| fetcher.parse | false (在nutch-default.xml中默认指定) | fetch阶段会根据此参数决定是否要进行parse |
|  |  |  |

初始地址：

<http://weibo.com/234205288>

在抓取sina过程中

1. http会重定向 ，http状态为 ProtocolStatus.TEMP\_MOVED

定向地址<http://passport.weibo.com/visitor/visitor?a=enter&url=http%3A%2F%2Fweibo.com%2F234205288&_rand=1416483242.5706>

http://passport.weibo.com/js/visitor/mini.js

会返回script 代码段由浏览器解析后进行跳转。（需要开发解析js功能）

### 分析mini.js 功能

全局window 结构如下：

window.ufp

window.ufp.config

window.ufp.util

第一个function，主要为window.ufp.util增加如下功能函数

Q.getActiveXObject = \_;

Q.isDefined = I;

Q.isArray = N;

Q.isFunc = C;

Q.isString = R;

Q.isNum = E;

Q.isStrNum = F;

Q.getNum = A;

Q.formatNum = J;

Q.garbage = B;

Q.getAXOByClsid = P;

Q.postData = $;

Q.wload = G.wload = M;

第二个function，主要构造window.Store

### 抓取内容分析

GET /234205288?sudaref=passport.weibo.com HTTP/1.1

Request 如下：

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

Accept-Encoding: gzip, deflate, sdch

Accept-Language: zh-CN,zh;q=0.8

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Cookie值

SINAGLOBAL=1111314452718.9434.1405487917849; ULV=1417415083219:22:1:1:8264130479656.16.1417415083198:1416483072482; \_s\_tentry=bbs.ouou.cn; SSOLoginState=1417675017; YF-Ugrow-G0=d753b4e8c22a4258612a40425af01f8d; UOR=club.pchome.net,widget.weibo.com,bbs.ouou.cn; SUS=SID-1744669500-1417681828-GZ-w2zyt-0c23e98bf371374c69aec42cd78616c5; SUE=es%3Da8770dea6a9220428d79232867b56a98%26ev%3Dv1%26es2%3D38774f8fd7c2d830d5bfa55120adf17f%26rs0%3Ddf8Pgay%252FOJdAuw8dc6bmTu1L%252FbDGWiEUfKY8C%252BsPYRqWakDtBzcyBjcZfEUalMPn4Ojr8Gp1tk62%252FPULI2BUfhjEqaTBV8wk4laDWN%252F6oTqyfTs5m7HHvuQumBfQsGEsCAgSc1qrUJH0goec%252BTZOb6fgcDe6rpgbJOZfpZxaMJ4%253D%26rv%3D0; SUP=cv%3D1%26bt%3D1417681828%26et%3D1417768228%26d%3Dc909%26i%3D16c5%26us%3D1%26vf%3D0%26vt%3D0%26ac%3D0%26st%3D0%26uid%3D1744669500%26name%3Dnullwang%2540hotmail.com%26nick%3Dnullwang%26fmp%3D%26lcp%3D2012-04-13%252010%253A33%253A41; SUB=\_2A255hGv0DeTxGedJ71YX9ifJyzyIHXVa8wm8rDV8PUNbu9BeLVnCkW8h7jPiCUa7GGSfb8jNmQJ2TcHd7w..; SUBP=0033WrSXqPxfM725Ws9jqgMF55529P9D9WWbffhv2BHUEURLLk1skZhn5JpX5Kzt; ALF=1449217827

获取到了最终html 信息：

|  |  |
| --- | --- |
| 获取 html页面cookie 有下列不同 |  |
| SUS |  |
| SUE |  |
| SUP |  |
| SUB |  |
| SUBP |  |
| ALF |  |

***GET /234205288 HTTP/1.1***

HTTP/1.1 302 Moved Temporarily

Server: WeiBo

Date: Thu, 04 Dec 2014 10:54:01 GMT

Content-Type: text/html; charset=utf-8

Connection: close

Expires: Sat, 26 Jul 1997 05:00:00 GMT

P3P: CP="CURa ADMa DEVa PSAo PSDo OUR BUS UNI PUR INT DEM STA PRE COM NAV OTC NOI DSP COR"

Cache-Control: no-cache, no-store

Pragma: no-cache

Location: http://passport.weibo.com/visitor/visitor?a=enter&url=http%3A%2F%2Fweibo.com%2F234205288&\_rand=1417690441.8948

DPOOL\_HEADER: balor202

Set-Cookie: YF-Page-G0=04608cddd2bbca9a376ef2efa085a43b;Path=/

LB\_HEADER: venus238

Content-Length: 0

***GET /visitor/visitor?a=enter&url=http%3A%2F%2Fweibo.com%2F234205288&\_rand=1417690441.8948 HTTP/1.1***

Response js min.js

***GET /js/visitor/mini.js HTTP/1.1***

***POST /visitor/genvisitor HTTP/1.1***

HTTP/1.1 200 OK

Server: nginx

Date: Thu, 04 Dec 2014 10:54:02 GMT

Content-Type: text/javascript; charset=utf-8

Connection: close

Cache-Control: no-cache, must-revalidate

Expires: Sat, 26 Jul 1997 05:00:00 GMT

Pragma: no-cache

DPOOL\_HEADER: dryad27

SINA-LB: aGEuMjM1LmcxLnF4Zy5sYi5zaW5hbm9kZS5jb20=

SINA-TS: ZWFjYTk0Y2UgMCAwIDAgMTggMTYwCg==

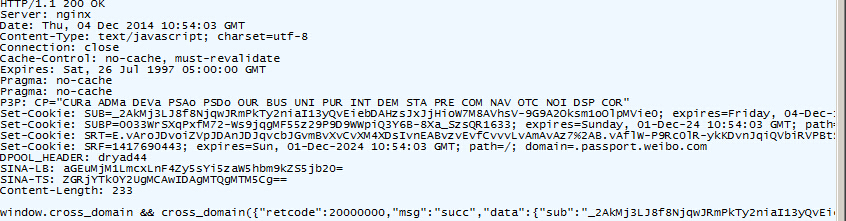
Content-Length: 165

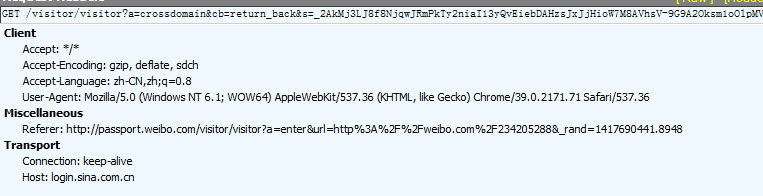
window.gen\_callback && gen\_callback({"retcode":20000000,"msg":"succ","data":{"tid":"7UbDhVDSg40Fx7ddsKZXzN+lavGRsG7NjiUWjeDm8QM=","new\_tid":false,"confidence":95}});

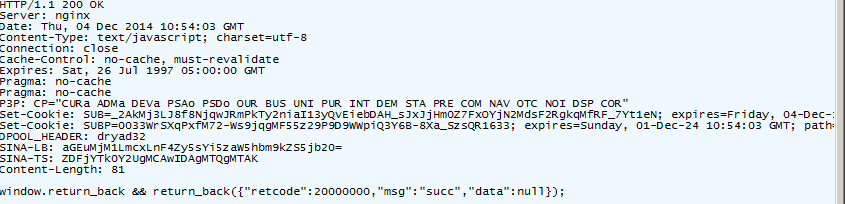
GET /visitor/visitor?a=incarnate&t=7UbDhVDSg40Fx7ddsKZXzN%2BlavGRsG7NjiUWjeDm8QM%3D&w=2&c=095&gc=&cb=cross\_domain&from=weibo&\_rand=0.29994952166453004 HTTP/1.1

Cookie

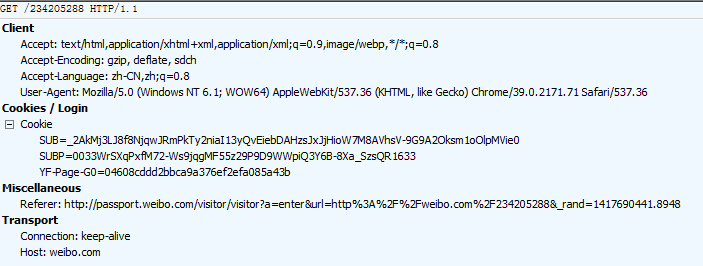








此请求返回真正的内容



## Baike抓取

http://baike.baidu.com/view/5081.htm

名称抓取分析

<span class="lemmaTitleH1">周星驰</span>

<span class="lemmaTitleH1">李修贤</span>

根据class 进行

<http://pivotallabs.com/xpath-css-class-matching/>

## 页面跳转方式

在抓取过程中，由于被抓取网站做了防御性机制，导致页面会跳转。

|  |  |  |
| --- | --- | --- |
| meta | <meta http-equiv="refresh" content="5;url=hello.html"> |  |
| JS | javascript:location.href='hello.html'  window.location.href='hello.html';  javascript:history.go(-1) |  |
|  |  |  |
| html file 使用 | <http://www.cs.tut.fi/~jkorpela/forms/file.html> |  |
|  |  |  |
|  |  |  |

## Js 文章参考

http://www.cnblogs.com/uedt/articles/1691443.html

<http://www.cnblogs.com/mountain-mist/articles/1600995.html>

http://www.blogjava.net/zkjbeyond/archive/2006/04/06/39514.html

## http 协议解析

http 消息： （请求消息、响应消息），【开始行+报头+正文】

|  |  |
| --- | --- |
| 请求行 | Method Request-URI HTTP-Version CRLF |
|  | method 分为：GET     请求获取Request-URI所标识的资源  POST    在Request-URI所标识的资源后附加新的数据  HEAD    请求获取由Request-URI所标识的资源的响应消息报头  PUT     请求服务器存储一个资源，并用Request-URI作为其标识 DELETE  请求服务器删除Request-URI所标识的资源  TRACE   请求服务器回送收到的请求信息，主要用于测试或诊断  CONNECT 保留将来使用  OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求 |
|  | POST /reg.jsp HTTP/ (CRLF) Accept:image/gif,image/x-xbit,... (CRLF) ... HOST:www.guet.edu.cn (CRLF) Content-Length:22 (CRLF) Connection:Keep-Alive (CRLF) Cache-Control:no-cache (CRLF) (CRLF)         //该CRLF表示消息报头已经结束，在此之前为消息报头 user=jeffrey&pwd=1234  //此行以下为提交的数据 |
|  |  |
|  |  |
| 响应 | HTTP-Version Status-Code Reason-Phrase CRLF |
|  | 1xx：指示信息--表示请求已接收，继续处理 2xx：成功--表示请求已被成功接收、理解、接受 3xx：重定向--要完成请求必须进行更进一步的操作 4xx：客户端错误--请求有语法错误或请求无法实现 5xx：服务器端错误--服务器未能实现合法的请求 |
|  |
|  |
|  |
|  |
|  | eg：HTTP/1.1 200 OK （CRLF） |
|  |  |

HTTP消息报头包括普通报头、请求报头、响应报头、实体报头。  
每一个报头域都是由名字+“：”+空格+值 组成，消息报头域的名字是大小写无关的

|  |  |
| --- | --- |
| 请求头 | Accept |
|  | Accept-Charset |
|  | Accept-Encoding |
|  | Accept-Language |
|  | Authorization |
|  | Host |
|  | User-Agent |
|  |  |
|  | eg：GET /form.html HTTP/1.1 (CRLF) Accept:image/gif,image/x-xbitmap,image/jpeg,application/x-shockwave-flash,application/vnd.ms-excel,application/vnd.ms-powerpoint,application/msword,\*/\* (CRLF) Accept-Language:zh-cn (CRLF) Accept-Encoding:gzip,deflate (CRLF) If-Modified-Since:Wed,05 Jan 2007 11:21:25 GMT (CRLF) If-None-Match:W/"80b1a4c018f3c41:8317" (CRLF) User-Agent:Mozilla/4.0(compatible;MSIE6.0;Windows NT 5.0) (CRLF) Host:www.guet.edu.cn (CRLF) Connection:Keep-Alive (CRLF) (CRLF) |
|  |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 响应头 | Location 重定向时使用 |
|  | Server eg Server：Apache-Coyote/1.1 |
|  | WWW-Authenticate 与User-Agent请求报头域是相对应的 |
|  | eg：WWW-Authenticate:Basic realm="Basic Auth Test!" |
|  |  |

|  |  |
| --- | --- |
| 实体报头 | Content-Encoding eg：Content-Encoding：gzip |
|  | Content-Language eg：Content-Language:da |
|  | Content-Type eg: Content-Type:text/html;charset=ISO-8859-1 |
|  | Expires |
|  | Last-Modified |
|  |  |

参考

http://www.cnblogs.com/li0803/archive/2008/11/03/1324746.html

# 流程

Injector

Create

generate

fetch

parse

updatedb

dedup

index

clean

## 数据目录结构描述

整个处理链中，后面环节会使用前面环节的数据。

***url\_dir***

此目录为用户url数据种子文件所在目录， injector会读取此目录下所有文件进行处理。

***Crawldb***

此目录为所有已处理及未处理url数据文件所在目录，Nutch 的所有工作均基于此目录。其下包含

***--- Current*** 当前所有数据文件存放在此目录

***--- Old*** 上次数据文件需要被删除时备份在此目录

***segments***

generator的输出目录，下面子目录名称（根据系统当前时间获取）格式为yyyyMMddHHmmss ,每generate一次产生一个子目录。

Segments

--- yyyyMMddHHmmss

-------- crawl\_generate

-------- crawl\_fetch

-------- crawl\_parse

-------- content

-------- parse\_text

-------- parse\_data

目录分布情况：

|  |  |
| --- | --- |
| ***Crawl\_generate*** | generator会输出在此目录 |
| ***crawl\_fetch*** | fetch后的数据 存储的数据为<Text, CrawlDatum> |
| ***content*** | fetch 写入 <Text, Content> |
| ***crawl\_parse*** | 由parser 写入 |
| ***parse\_text*** | 由parser 写入 |
| ***parse\_data*** | 由parser 写入 |

***linkdb***

存储linkdb job从parse\_data中提取出来的反向链接的数据。其下包含两个目录

---current 当前数据

---old 上次备份

## Injector

***使用：***Injector <crawldb> <url\_dir>

***输入*** ：crawldb url目标数据库目录

***输入*** ：url\_dir url源输入目录,

***说明：将url dir中url数据注入到crawl db***

**数据格式：**url\_dir 中的文件数据格式， 由url和元数据组成。

例如：http://www.nutch.org/ \t nutch.score=10 \t nutch.fetchInterval=2592000 \t userType=open\_source

InjectMapper

**输入：** <url\_dir>

**输出：<crawl>/**current， 输出格式为 (url, CrawlDatum)

如果crawdb 不存在，则rename InjectMapper处理结果目录为crawdb。

存在，则产生InjectReducer 使用InjectMapper输出结果更新 crawdb内容。

sortJob

rename to

url\_dir

tempDir1

mergeJob

tempDir1

crawldb

tempDir2

tempDir2

crawDb

最终生成的数据位于**<crawdb>/current** 目录下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 阶段 | JOB name | 输入目录 | 读取格式 | 输出 | 输出格式 |
| Injector | sort | ${url\_dir} |  | ${mapred.temp.dir}/ inject-temp-$(randomNum) | < Text,CrawlDatum>  MapFileOutputFormat 或  SequenceFileOutputFormat |
|  | merge | ${sort.output}  ${crawldb}/current | SequenceFileInputFormat | ${crawdb}/${randomNum} | < Text,CrawlDatum> |
|  |  |  |  |  |  |
| Generator | select | ${crawldb}/current | SequenceFileInputFormat | ${mapred.temp.dir}/generate-temp-${UUID} | <FloatWritable, SelectorEntry>  GeneratorOutputFormat |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Generator

***使用：Generator <crawldb> <segments\_dir> [-force] [-topN N] [-numFetchers numFetchers] [-adddays numDays] [-noFilter] [-noNorm][-maxNumSegments num]***

***作用：***从crawldb中选取一部分数据。

***topN GENERATOR\_TOP\_N 指定每个segment 最多记录数***

***maxNumSegments 指定segment最大数***

Selector Job

根据maxNumSegments 参数将 crawldb中数据产生为多个segment，每个segment记录数受到topN限定。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | 输入通过scoringFilter获取到分值， | Text, CrawlDatum | <FloatWritable,SelectorEntry> |
| Partitioner | URLPartitioner 根据url所在机器将进行分区，（url指向同一主机的将分配到同一reducer） |  |  |
| Reducer | maxNumSegments：可产生的最多setments  limit 每个segment的最多记录数（来源与topN） | FloatWritable, Iterator<SelectorEntry> | <FloatWritable,SelectorEntry>  产生的文件列表：  fetchlist-${segNum}/${name} |
|  |  |  |  |

partitionSegment Job

其ruduceTask 数量由 参数 ***numFetchers 决定***

Select job 会针对每个segment产生相应的目录${mapred.temp.dir}-${UUID}/fetchlist-${segNum}，根据用户输入可产生多个目录输出。

Partition job

针对每个fetchlist-${segNum}都会产生一个Partition job， 此job的ruduceTask数量为***numFetchers***

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | SelectorInverseMapper | Select job 的output  FloatWritable key, SelectorEntry value | Text,SelectorEntry  文件如下：$<segments\_dir>/yyyyMMddHHmmss/crawl\_generate |
| Partitioner | URLPartitioner |  |  |
| Reducer | PartitionReducer | Text key, Iterator<SelectorEntry> values | OutputCollector<Text,CrawlDatum> |

SelectorEntry

|  |  |
| --- | --- |
| url | Text |
| datum | CrawlDatum |
| segnum | 数据所在的segment number |

***结果：***经过Partitioner处理后，数据将以Text,CrawlDatum格式存储在<segments\_dir>/<yyyyMMddHHmmss>/crawl\_generate 目录

Updatedb job

根据select产生的内容，对crawldb 进行合并更新。

更新URL的元数据中的WRITABLE\_GENERATE\_TIME\_KEY

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | CrawlDbUpdater | crawldb/current  $<segments\_dir>/yyyyMMddHHmmss/crawl\_generate |  |
| Reducer | CrawlDbUpdater |  | MapFileOutputFormat  <Text, CrawlDatum>  目录：${mapred.temp.dir}/generate-temp- ${UUID} |

## Fetcher

Fetcher会根据指定的threads 参数产生多个抓取线程(FetcherThread) ,及一个提供线程(QueueFeeder)。 这些线程运行于Mapper Job所在的机器上。FetchThread和QueueFeeder通过队列交互，QueueFeeder读取Mapper 的输入根据队列空满的情况酌情将数据放入队列。 FetcherThread抓取指定的数据，产生输出。

Fetcher 如果存在parse，会将parse的结果中的url存入队列。会根据参数fetcher.parse的设置来决定是否进行parse,默认为false（在nutch.default.xml中指定）。

关键性参数

|  |  |  |
| --- | --- | --- |
|  |  |  |
| fetcher.threads.fetch | 10， fetch 线程的数量 | configuration |
| -threads |  | 用户指定 |
| fetcher.timelimit.mins | -1秒单位， | 指定fetch线程结束时间 |
| fetcher.follow.outlinks.depth | -1 |  |
| fetcher.follow.outlinks.num.links | 4 |  |
| fetcher.follow.outlinks.depth.divisor | 2 |  |
| fetcher.queue.depth.multiplier | 50 线程队列因子，QueueFeeder 中使用  threadCount \* queueDepthMuliplier作为size来确定是否该放慢生产者 |  |
|  |  |  |
|  |  |  |

Fetch

其没有设置Mapper类，取而代之设置了MapperRunnerClass。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| MapperRunnerClass | Fetcher.class | $<segments\_dir>/yyyyMMddHHmmss/crawl\_generate | $<segments\_dir>/yyyyMMddHHmmss/  FetcherOutputFormat.class  <Text, NutchWritable> |
| FetcherOutputFormat | 自定义输出格式 | crawl\_fetch目录：  crawl\_parse目录： |  |

MapperRunnerClass的输入为RecordReader<Text, CrawlDatum>，输出为OutputCollector<Text, NutchWritable> 。

fetcherThreads.start

Feeder.start

fetchThread.count > 0

fetchItem的数据结构

|  |  |  |
| --- | --- | --- |
|  |  |  |
| url | text |  |
| datum | CrawlDatum |  |
| queueID | url.protocol + “://” + url.getHostName |  |
| outlinkDepth | int |  |

FetchItemQueue

|  |  |  |
| --- | --- | --- |
|  |  |  |
| maxThreads | queue支持最多读取线程，超过此数读取将return null。 |  |
| inProgress | 线程数 |  |
| addFetchItem | 生产者，将数据放入队列 | 非阻塞 |
| getFetchItem | 消费者获取 | 非阻塞 |
| finishFetchItem | 获取结束。获取完成后，调用此方法 |  |
|  |  |  |

FetchItemQueues

队列集合

|  |  |  |
| --- | --- | --- |
|  |  |  |
| queues | Map<String,FetchItemQueue> 其中key为queueId其从FetchItem中获取 |  |
| getFetchItem | 获取FetchItem，迭代队列获取一个可用的FetchItem | 非阻塞 |
| addFetchItem |  | 非阻塞 |
| finishFetchItem |  |  |

QueueFeeder

生产者，一个线程，通过Mapper读入数据放入队列。

FetcherThread

消费者，多个线程，由参数指定，当消费者从队列中取不到FetchItem时将会退出。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (key, new NutchWritable(datum) | 更新fetch 状态，输出 | $<segments\_dir>/yyyyMMddHHmmss//crawl\_fetch目录 |
| key, new NutchWritable(content) | 如有必要，输出fetch内容 | $<segments\_dir>/yyyyMMddHHmmss//content |
| key, new NutchWritable(new ParseImpl(new ParseText(parse.getText()),parseData, parse.isCanonical() |  | ParseOutputFormat |
|  |  |  |
|  |  |  |

当FetchThread会对取到的内容Parse，parse出outlink。这些outlink会加入到FetchItemQueues，同时output也会收集这些parse后的数据。

Fetch如何结束，

Fetch时会在mapper机器上运行多个线程（一个feeder 线程、多个fetcher线程、一个watcher线程）。 Fedder和fetcher均为daemon线程，Watcher线程为主线程。

## Parser Segment

Parse会将fetcher 的输出作为输入（fetcher 结果的content目录）

Parse job

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | ParseSegment | $<segments\_dir>/yyyyMMddHHmmss/content | $<segments\_dir>/yyyyMMddHHmmss/ |
| Reducer | ParseSegment |  | ParseOutputFormat |

Parse也可以在Fetch时完成。

ParseOutputFormat对应多个目录都在$<segments\_dir>/yyyyMMddHHmmss/下

|  |  |  |
| --- | --- | --- |
|  |  |  |
| parse\_text | 解析出来的文本 | parse.getText() |
| parse\_data | parse出来的data | 包括title，parseMeta，parseStatus，contentMeta，OutLink， |
| crawl\_parse | crawl之后生成的元数据 |  |

## Crawldb

用来将多个目录的数据更新进入crawldb

Update job

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper |  | $<segments\_dir>/yyyyMMddHHmmss/crawl\_fetch  $<segments\_dir>/yyyyMMddHHmmss/crawl\_parse  ${crawldb} | ${crawldb} |
| Reducer |  |  |  |

## LinkDb

Linkdb

输出url及其fromUrl

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | LinkDb | $<segments\_dir>/yyyyMMddHHmmss/parse\_data | $linkdb/linkdb-${randomInt} |
| Combiner | LinkDbMerger |  |  |
| Reducer | LinkDbMerger |  | MapFileOutputFormat  <Text, Inlinks> |
|  |  |  |  |

Merge Job

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper |  | $linkdb/current  linkDb job的输出 | linkdb-merge-${randomInt} |
| Reducer |  |  |  |
|  |  |  |  |

LinkDb.install 将job的输出安装到如下目录: linkDb/current

## Dedup

去重，由两个job构成，一个标记、一个去重。（因为数据量大，标记&去除分为2个步骤）

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| DeduplicationJob |  | ${crawldb}/current |  |
| DeduplicationMergeJob |  |  |  |
|  |  |  |  |

## Index

将获取到的数据提交到搜索服务器,通过插件进行提交。

Indexer job

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 输入 | 输出 |
| Mapper | IndexerMapReduce | ${crawlDb}/current  crawl\_fetch  crawl\_parse  parse\_data  parse\_text  ${linkdb}/current  <key, Writable> | <Text, NutchWritable > |
| Reducer | IndexerMapReduce | <Text, NutchWritable > | <Text, NutchIndexAction>  IndexerOutputFormat |

IndexerOutputFormat

适配器，调用IndexWriters将数据通过插件进行输出。

# 插件

PluginRepository 插件库，所有插件经在此登记。

plugin descriptor 代表一个插件，其包含插件所有信息， plugin使用lazy loadin机制，在需要的时候才进行插件加载。

Nutch-default.xml 中plugin.folders 指定了插件所在目录。插件通过classLoader定位资源的url然后通过java 的File来进行加载。

plugin.includes 指定加载那些插件

plugin.excludes 指定不加载的插件

由于MIME文件特性，不同类型的资源将使用不同的parse进行解析，通过nutch-default.xml中parse.plugin.file配置项指定资源与parser的对应关系(默认值为parse-plugins.xml文件)

## 插件机制

ExtensionPoint 扩展点

PluginDescriptor 插件描述符，代表每个插件。（通过解析plugin.xml）

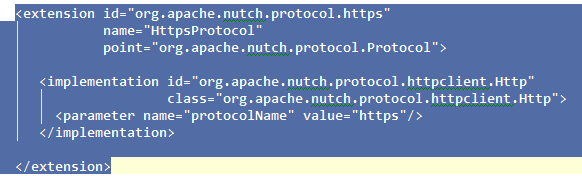
pluginRepository 插件库

插件库指定特定插件目录，获取该目录所有插件，



|  |  |
| --- | --- |
| plugin | id 唯一描述一个插件 |
| extension | 扩展，用了描述插件针对扩展的实施，可存在多个extension元素 |
| extension-point | 申明一个扩展点，nutch中扩展点的描述在插件nutch-extensionpoints中描述 |
| runtime | 根据export属性将指定的文件添加到exportLib或noExportLib |
| requires | 解析其依赖的插件，其中plugin为插件id |
|  |  |

Extension中实施还可以存在参数



### ClassLoader

每个pluginDescriptor有其单独的classLoader，其包含自身的exportedLib和 noExportedLib和其依赖的plugin的exportedLib（递归包含）

### URLNormalizer

HostURLNormalizer 规范主机名称，通过空格分开，将名称1映射为2

根据输入参数文件对主机进行映射，文件示例：

# Force all sub domains to www.

\*.example.com example.com

# Force no sub domain to www. URL's

www.example.net example.net

# Force www. sub domain when hitting link without sub domain

example.org www.example.org

### Protocol

***如何查找合适的Protocol？***

Nutch 提供ProtocolFactory通过其获取protocol，getProtocol(String protocolName) 通过匹配extension的属性protocolName的值与协议名称匹配来确定相应的插件，找到第一个能匹配的插件即可。

### Parser

***如何找到合适的Parser？***

ParseUtil通过ParserFactory获取parsers。使用分别使用每个Parser去解析内容，成功即返回，如果失败再使用下一个进行尝试。

与parser相关的配置 nutch-default.xml中的***parse.plugin.file***指定parse的插件文件***parse-plugins.xml*** 此文件指定了插件名称和mime类型的对应关系。

ParserFactory首先通过content的contentType在parse-plugin.xml中寻找匹配的插件，再在插件中寻找相应的extensions，通过extension的属性contentType的值进行匹配。每个Plugin中仅找到第一匹配的Extension。如果通过contentType属性没有匹配到extension，则通过id进行匹配（plugin.id == extension.id）。

### IndexFilter

用于IndexJob中对NutchDocument进行filter，在文档中添加基础的field如：domain,host,url,content,title,tstamp等。

Nutch默认启用了index-basic,index-anchor两个filter。

### IndexWriter

IndexWriter将输入NutchDocument转为输出SolrInputDocument同时将输出结果提交到solr sever，文档的转换通过solrindex-mapping来变换名称。需要注意solr文档schema的设置。

### 与solr集成

安装solr

1. 下载solr 包 http://www.apache.org/dyn/closer.cgi/lucene/solr/

* 启动 java -Djetty.port=8983 –jar start.jar cd ${APACHE\_SOLR\_HOME}/example

Solr 自带查询界面

<http://127.0.0.1:8983/solr/collection1/browse>

参考 【http://wiki.apache.org/nutch/NutchTutorial】

启用插件 indexer-solr。

### 自定义插件

htmlParseFilters

多个插件之间的顺序通过conf 中的htmlparsefilter.order来指定。

接管 Parser 和 IndexWriter

Parser的结果ParseData中contentMeta用来保存entity，缺省使用第一个entity作为主要entity。在同一个ParseData中其它entity与主要entity存在关联，不同结果Parse通过Outlink[]存在关联

|  |  |  |
| --- | --- | --- |
|  |  |  |
| r |  |  |
| s |  |  |
| w |  |  |
| d |  |  |
| j |  |  |

# 构建插件

1. 在plugins 下增加工程
2. 修改plugins/build.xml文件，加入build 目标

# 调试

设置

export NUTCH\_OPTS="-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=5005"

export NUTCH\_OPTS=

或修改nutch文件加入

-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=5005)

针对sina weibo

<http://passport.weibo.com/visitor/visitor?a=enter&url=http%3A%2F%2Fweibo.com%2Fnullwang&_rand=1415068764.884>

## 插件没加载

发现自己开发的插件没有加载？

Configuration: 中加载了nutch-default.xml和nutch-site.xml

**Nutch-default.xml中默认使用了如下设置：**

|  |  |
| --- | --- |
|  |  |
| includes | protocol-http|urlfilter-regex|parse-(html|tika)|index-(basic|anchor)|indexer-solr|scoring-opic|urlnormalizer-(pass|regex|basic) |
| excludes |  |

## URL 被reject

Regex filter 将url过滤了。修改regex-urlfilter.txt文件设置接收的url pattern。

# Hadoop

## Job

|  |  |
| --- | --- |
| InputFormat |  |
| MapperClass |  |
| ReducerClass |  |
| OutputFormat |  |
| MapOutputValueClass |  |
| OutputKeyClass |  |
| OutputValueClass |  |

# Read 式抓取

Cobra toolkit

|  |  |  |
| --- | --- | --- |
| cobra | * XAMJ\_Project/Common * XAMJ\_Project/cssparser * XAMJ\_Project/HTML\_Renderer | 依赖项  **Cobra: Java HTML Renderer & Parser** |
| Additional modules required for the browser are | * XAMJ\_Project/Platform\_Public\_API * XAMJ\_Project/Platform\_Core * XAMJ\_Project/Primary\_Extension |  |
| Anyone interested in the JavaFX/Java rendering extension should check out: | * XAMJ\_Project/JWebExtension |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 参考

<http://stackoverflow.com/questions/4942169/once-again-how-to-get-dom-in-javascript-engine-has-this-somebody-done>

<http://programmers.stackexchange.com/questions/200247/a-few-clarifications-about-the-dom>

<http://lobobrowser.org/java-browser.jsp>

爬虫

<http://blog.csdn.net/zhouyong0/article/details/6460971>

# 问题

***？***java.io.IOException: Failed to set permissions of path: \tmp\hadoop-wangqiaodong581\mapred\staging\wangqiaodong5811149935579\.staging to 0700

在cygwin运行bin/crawl urls -dir crawl -depth 3 -topN 5出现错误

http://stackoverflow.com/questions/17208736/failed-to-set-permissions-of-path-tmp/18381218#18381218

参数用

-Dhadoop.tmp.dir=<A directory location with write permission>

相互引用问题，

A 引用 B, B引用 A. 在fetcher时，A 将被fetch多少次？

## [The temporary job-output directory doesn't exist!](http://blog.csdn.net/huojushou1128/article/details/23129915)

Nutch 在Inject时出现上述错误

解决方法：关闭hadoop的safe模式 hadoop dfsadmin -safemode leave。  
原因：有可能是在执行mapreduce任务时强制退出，迫使namenode进入safe模式。

Ggy 环境，

解决办法： 在nutch home目录下执行命令。

## Fetcher: java.io.IOException: Segment already fetched!

at org.apache.nutch.fetcher.FetcherOutputFormat.checkOutputSpecs(FetcherOutputFormat.java:58)

删除fetch的输出crawl\_fetch 目录，再次运行fetch

## ThreadPoolExecutor

设置SynchronousQueue同步队列后，如何退出？

getTask，方法将被阻塞。

Shutdown 和 shutdownNow 将会 interrupt idle task，从而使 getTask 线程返回。

## Unique ID 生成算法

简单&实用 like youtube

<http://kvz.io/blog/2009/06/10/create-short-ids-with-php-like-youtube-or-tinyurl/>

将number转换为字符。

base62

基于[0-9][a-z][A-Z]构造62个数，log62 转换。

基于hash或其它算法对62个基字母进行排序。

## UUID

<https://kanru.info/blog/archives/2010/11/16/uuid-el/>

## Float

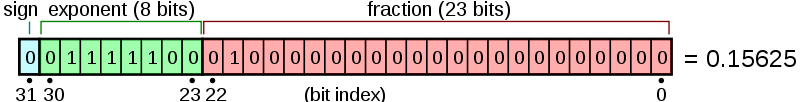
小数点后位数不固定 32-bit base 2

 IEEE 754 standard specifies a **binary32**

Sign 1bit

Exponent 8bits

[Significand](http://www.ask.com/wiki/Significand?qsrc=3044) [precision](http://www.ask.com/wiki/Precision_(arithmetic)?qsrc=3044) 24 bits (23 explicitly stored)



## Failed to create task or type antlib

使用intellij 的ant 直接构建时，出现上述错误

解决办法：自行安装apache 的ant，在ant的lib目录下放入ivy-2.2.0.jar文件。同时修改intellij中工作的构建属性(Ant Build->Build file properties->Execution)。

## Eclipse IVY plugin

为了解析ivy 依赖，需安装插件IvyIDEA 。由于nutch 工程的项目结构复杂，plugin的源码也在nutcch src目录下。由于各自包含Nutch core和nutch plugin可分别

## SSL/TLS

<https://sites.google.com/site/ssljavaguide/example-code>

使用java 默认的keystore 和truststore

[http://www.exampledepot.com/egs/javax.net.ssl/Client.html](https://108.61.220.71/browse.php?u=SETxSyEyIMMf2gcsm0yjX45sgdRqDt4kAQa7DHJpCjqKgSiZGxTJGV33Jt%2FDWp7oy%2BBHtI8%3D&b=14).

非对称加密

Keys : 秘钥对， 公钥和私钥。

Certificates：证书 ，由公钥，标示信息和过期时间组成。

CA ： certificate authorities （证书）认证机构

使用示例：

<http://www.herongyang.com/PKI/HTTPS-Java-javax-net-ssl-trustStore-System-Property.html>

[http://www.java2s.com/Code/Java/Servlets/javaxservletrequestX509Certificate.htm](https://108.61.220.71/browse.php?u=SETxSyEyIMwGzQtuhAekVZMstpVtBJxBBRepDEttDi2eyjKPQFDbHFCgFtbYSZz5kfpWqJa4DVD5A3lCHWVdm1lh0cabW8GJMRfp&b=14)

http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html

## Authentication

认证

## Cipher Suite

## Certificate

证书

## Cryptographic Hash Function

a small change in the original data produces a large change in the resulting hash; and it does not require a cryptographic key.

## cryptographic key.

秘钥对

## cryptographic algorithm

秘钥算法

## Digital Signature

数字签名

RSA 先hash 再用私钥 加密

RSA-based digital signature is calculated by first computing a cryptographic hash of the data and then encrypting the hash with the sender's private key

Ciphertext

Cleartext

encrypt and decrypt

secret key cryptography and public key cryptography.

## Key Agreement

## Key Exchange

一方产生对称key，用对方的pulic key 加密后发生给对方。

One side generates a symmetric key and encrypts it using the peer's public key (typically RSA). The data is then transmitted to the peer, who then decrypts the key using its corresponding private key

## Message Authentication Code

验证消息的完整性， 消息是否被修改过

## 3个问题

1. You cannot always be sure that the entity with whom you are communicating is really who you think it is.

authentication

2. Network data can be intercepted, so it is possible that it can be read by an unauthorized third party, sometimes known as an attacker.

Encrypting

3. If an attacker can intercept the data, the attacker may be able to modify the data before sending it on to the receiver.

Checksum

先认证 再 加密传输（用同一个secret key cryptography进行 encrypt and decrypt the messages）。

SSL uses public key cryptography to provide authentication ， secret key cryptography and digital signatures to provide for privacy and data integrity

Secret key cryptography is also called *symmetric cryptography*（私钥密码体制），包含如下算法

 Data Encryption Standard (DES)

 triple-strength DES (3DES),

Rivest Cipher 2 (RC2),

Rivest Cipher 4 (RC4).

Public key cryptography 公钥密码体制，包含如下算法 *asymmetric cryptography*

Rivest Shamir Adleman (RSA) algorithm

 Diffie-Hellman (DH) algorithm.

 It is therefore typically used only for encrypting small pieces of data, such as secret keys

其比较慢，仅

A digital signature is one of the components of a public key certificate，, and is used in SSL to authenticate a client or a server（公钥证书）

A public key certificate contains several fields, including:

* Issuer - The issuer is the CA that issued the certificate. If a user trusts the CA that issues a certificate, and if the certificate is valid, the user can trust the certificate.（颁发者）
* Period of validity - A certificate has an expiration date, and this date is one piece of information that should be checked when verifying the validity of a certificate.(有效期)
* Subject - The subject field includes information about the entity that the certificate represents.
* Subject's public key - The primary piece of information that the certificate provides is the subject's public key. All the other fields are provided to ensure the validity of this key.
* Signature - The certificate is digitally signed by the CA that issued the certificate. The signature is created using the CA's private key and ensures the validity of the certificate. Because only the certificate is signed, not the data sent in the SSL transaction, SSL does not provide for non-repudiation.（CA 签名）

密码hash 函数和checksum区别

**A cryptographic hash function** is similar to a checksum. The main difference is that while a checksum is designed to detect accidental alterations in data（检测偶然修改）, a cryptographic hash function is designed to detect deliberate alterations（检测故意修改）.

 A cryptographic hash function does not require a cryptographic key. The slightest change to the message typically makes a large change in the resulting hash. （小修改导致大变更）

经常使用：

Message Digest 5 (MD5)

Secure Hash Algorithm (SHA)

#### Message Authentication Code

和A cryptographic hash function类似，但需要 a secret key： HMAC.