Wikipedia Mirror

Chris Perivolaropoulos

Sunday 21 February 2016

Contents

1 The xerces bug

At the time of writing mwdumper a strange, semi-random bug. While `make sql-dump-parts` is running the following is encountered:

```
...

376,000 pages (14,460.426/sec), 376,000 revs (14,460.426/sec)
377,000 pages (14,458.848/sec), 377,000 revs (14,458.848/sec)
Exception in thread "main" java.lang.
    ArrayIndexOutOfBoundsException: 2048
         at org.apache.xerces.impl.io.UTF8Reader.read(Unknown
             Source)
         at org.apache.xerces.impl.XMLEntityScanner.load(Unknown
              Source)
         at org.apache.xerces.impl.XMLEntityScanner.scanContent(
             Unknown Source)
         at org.apache.xerces.impl.
             XMLDocumentFragmentScannerImpl.scanContent(Unknown
             Source)
         at org.apache.xerces.impl.
             XMLDocumentFragmentScannerImpl$FragmentContentDispatcher
             .dispatch(Unknown Source)
         at org.apache.xerces.impl.
             XMLDocumentFragmentScannerImpl.scanDocument(Unknown
              Source)
         at org.apache.xerces.parsers.XML11Configuration.parse(
             Unknown Source)
         at org.apache.xerces.parsers.XML11Configuration.parse(
             Unknown Source)
         at org.apache.xerces.parsers.XMLParser.parse(Unknown
             Source)
         at org.apache.xerces.parsers.AbstractSAXParser.parse(
             Unknown Source)
         at org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser.
             parse(Unknown Source)
         at javax.xml.parsers.SAXParser.parse(SAXParser.java
             :392)
         at javax.xml.parsers.SAXParser.parse(SAXParser.java
             :195)
         at org.mediawiki.importer.XmlDumpReader.readDump(
             XmlDumpReader.java:88)
         at org.mediawiki.dumper.Dumper.main(Dumper.java:142)
make: *** [/scratch/cperivol/wikipedia-mirror/drafts/wikipedia-
    parts/enwiki-20131202-pages-articles20.xml-
    p011125004p013324998.sql] Error 1
```

Inspecting the makefules and running `make --just-print sql-dump-parts` we find out that the failing command is:

```
$ java -jar /scratch/cperivol/wikipedia-mirror/tools/mwdumper.
    jar   --format=sql:1.5 /scratch/cperivol/wikipedia-mirror/
    drafts/wikipedia-parts/enwiki-20131202-pages-articles20.xml
    -p011125004p013324998.fix.xml > /root/path/wikipedia-parts
    //enwiki-20131202-pages-articles20.xml-p011125004p013324998
    .sql
```

Fortunately this does not run for too long so we can safely experiment. Here is the `time` output:

```
26.65s user 1.73s system 78% cpu 35.949 total
```

The error seems to be during reading of the XML dump so it is not specific to SQL output. This could be useful for figuring out which article causes the error, removing which will hopefully resolve the error. To find that out we first try exporting to XML:

```
$ java -jar /scratch/cperivol/wikipedia-mirror/tools/mwdumper.
    jar   --format=xml /scratch/cperivol/wikipedia-mirror/
    drafts/wikipedia-parts/enwiki-20131202-pages-articles20.xml
    -p011125004p013324998.fix.xml > /tmp/just-a-copy.xml
```

As expected the same error as above is yielded. We then look for the last article two it tried to export by printing in reverse order the output xml file, finding the last two occurances of *<title>* with `grep` and reverse again to print them in the original order (note that `tac` is like `cat`, only that yields lines in reverse order):

```
$ tac /tmp/just-a-copy.xml | grep "<title>" -m 2 | tac
      <title>The roaring 20s</title>
      <title>Cranopsis bocourti</title> # <- This is the last
          one
```

This operation finishes quickly despite `/tmp/just-a-copy.xml` being fairly large because `tac` seeks to the end of the file and reads backwards until `grep` finds the 2 occurances it is looking for and quits. On ext3 the seek operation does not traverse the entire file. Indeed from the `tac` source code:

```
if (lseek (input_fd, file_pos, SEEK_SET) < 0)
    error (0, errno, _("%s:␣seek␣failed"), quotef (file));
/* Shift the pending record data right to make room for the new
    .
   The source and destination regions probably overlap.  */
memmove (G_buffer + read_size, G_buffer, saved_record_size);
past_end = G_buffer + read_size + saved_record_size;
/* For non-regexp searches, avoid unnecessary scanning. */
if (sentinel_length)
    match_start = G_buffer + read_size;
else
```

```
    match_start = past_end;

if (safe_read (input_fd, G_buffer, read_size) != read_size)
{
    error (0, errno, _("%s:␣read␣error"), quotef (file));
    return false;
}
```

Let's save the path of the original xml file in a variable as we will be using it a lot. So from now on `$ORIGINAL_XML` will be the path of the original xml.

```
$ export ORIGINAL_XML=/scratch/cperivol/wikipedia-mirror/drafts
    /wikipedia-parts/enwiki-20131202-pages-articles20.xml-
    p011125004p013324998.fix.xml
```

First let's see if there is anything strange going on in the xml file:

```
$ grep "<title>Cranopsis␣bocourti</title>" -A 200 -B 100
    $ORIGINAL_XML | less
```

`| less` is to browse and `-A 200 -B 100` means *"show 200 lines after and 100 before the matching line"*. Nothing peculiar was found, so we can't really fix the problem in-place, we will try crudely removing the entire article and hope it works (spoiler alert: it does).

We will try to inspect the parents of the `title` of the breaking article. Fortunately the generated xml is indented so we can find the parents based on that. We count 6 spaces of indentation so we will search backwards from there on each level of indentation. The first line we find on each case will be a direct parent of the article.

```
$ for i in {0..6}; do \
    echo "Level␣$i:"; \
    tac /tmp/just-a-copy.xml | grep "^␣\{$i\}<[^/]" -m 1 -n |
        tac; \
done

Level 0:
17564960:<mediawiki xmlns="http://www.mediawiki.org/xml/export
    -0.3/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    " xsi:schemaLocation="http://www.mediawiki.org/xml/export
    -0.3/␣http://www.mediawiki.org/xml/export-0.3.xsd" version=
    "0.3" xml:lang="en">
Level 1:
Level 2:
38:  <page>
Level 3:
Level 4:
35:    <revision>
Level 5:
```

```
Level 6:
26:      <text xml:space="preserve">&lt;!-- This article was
    auto-generated by [[User:Polbot]]. --&gt;
```

Looks like the xml is just `page` s thrown in a grand domain called `mediawiki`. We could have seen that from the java source too but as expensive as this is, it is much faster than dealing with the source of `mwdumper`.

The easiest way to cut off this article would be `awk` but that will take ages and we want to optimize and automate this entire process. First let's try just plain comparing the articles:

```
$ cmp /tmp/just-a-copy.xml $ORIGINAL_XML
/tmp/just-a-copy.xml /scratch/cperivol/wikipedia-mirror/drafts/
    wikipedia-parts/enwiki-20131202-pages-articles20.xml-
    p011125004p013324998.fix.xml differ: byte 2, line 1
```

That was fast... Let's see what went wrong:

```
$ head $ORIGINAL_XML
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.8/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    schemaLocation="http://www.mediawiki.org/xml/export-0.8/␣
    http://www.mediawiki.org/xml/export-0.8.xsd" version="0.8"
    xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.23wmf4</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Special</
          namespace>
      <namespace key="0" case="first-letter" />

$ head /tmp/just-a-copy.xml
<?xml version="1.0" encoding="utf-8" ?>
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.3/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    schemaLocation="http://www.mediawiki.org/xml/export-0.3/␣
    http://www.mediawiki.org/xml/export-0.3.xsd" version="0.3"
    xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.23wmf4</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2">Media</namespace>
```

The attributes of the xml tags are quite different. Our best chance is if the line numbers match up. We count the numbers of lines in `/tmp/just-a-copy.xml` and hope that the corresponding line number in `$ORIGINAL_XML` will be the same line. If that is so we can ignore the the contextual xml information and just blank out the problematic article. We will use `wc` which is also quite fast.

```
$ wc -l /tmp/just-a-copy.xml
17564961 /tmp/just-a-copy.xml
```

And the corresponding line in `$ORIGINAL_XML` would be about:

```
$ sed "17564960q;d" $ORIGINAL_XML
[[Willie Jones (American football)|Willie Jones]],
```

Football... nothing to do with frogs. Looks like there is no avoiding some level of parsing.

1.1 Parsing

We will make the following assumptions to avoid properly parsing the document:

- The XML in the original file is valid
- Any XML within the articles is HTML escaped

First off working with lines is slow because user space code needs to look for newlines. Working bytes delegates work to the kernel, speeding things up considerably. So the `dd` is the right tool for the job. So we will first find at which byte is the article I am interested in.

```
$ grep -b "<title>Cranopsis␣bocourti</title>" -m 1
    $ORIGINAL_XML
1197420547:     <title>Cranopsis bocourti</title>
```

This may take a little while but you are stuck with it unfortunately. Our stategy is to make two files: `/tmp/original_tail.xml` that will contain all the data *after* the page we want to remove and `/tmp/original_head.xml` that will contain all the data *before* the page we want to remove.

Now we will use `sed` to look for `</page>` after byte 1197420547 which will be point x we will and dump the contents of `$ORIGINAL_XML` after point x:

```
$ dd if=$ORIGINAL_XML skip=1197420547 ibs=1 | sed '0,/<\/page>/
    d' > /tmp/original_tail.xml
```

Great, that worked! `dd` does not copy in reverse so we will need to do something more complex to construct `/tmo/original_head.xml`. Let's say the position where we found the title of the page we want to remove is $\alpha = 1197420547$ and the point where the page starts is point β. It is fairly safe to assume that $\beta > \alpha - 1000$ (we can calibrate the constant 1000 if that assumption is wrong, but it turns out that it isn't). This way we only need to search into 1Kb for `<page>`. Effectively instead of copying the bytes in range $[0, \beta)$ we are concatenating two ranges $[0, \alpha - 1000] \cup (\alpha - 1000, \beta)$ by making a subshell that will first output the first range and then output $(\alpha - 1000, \alpha)$ stopping when it finds `<page>`. Here is the one liner:

```
$ (dd count=$((1197420547-1000)) ibs=1 if=$ORIGINAL_XML; \
   dd if=$ORIGINAL_XML count=1000 skip=$((1197420547-1000)) ibs
       =1 \
        | tac | sed '/<page>/,$d' | tac) > /tmp/original_head.
            xml
```

1.2 Covering up with spaces

1.3 The sed command

Above we kind of glazed over our use the `sed` command but it might be interesting to spend some time on it.

2 mediawiki stack overview

Wikipedia-mirror builds upon the mediawiki stack provided by bitnami. A service that builds the entire server within the confines of a directory. This is useful because we avoided the overhead of dealing with container or VM technologies and we had direct access to the filesystem of the stack while still having bitnami's build system do the tedious job of orchestrating the various components and separating our sever from the rest of the system.

The stack is comprised of

- An http server, in our case apache
- The web application runtime, in our case PHP
- A database, in our cas MySQL
- The web application itself, in our case mediawiki

All of the above are provided by the the bitnami mediawiki stack. Xampp used to be go-to for that but it is unmaintained so we decided to go with bitnami which works pretty well.

Once the stack is set up properly the wikipedia dump xml is downloaded and then turned into an sql dump with mwdumper. Could be piped directly to MySQL? but extracting can take time and things tend to go wrong during the dumping step.

2.1 Elements of the stack

We present each of the elements of the stack in more detail below.

1. Apache

   As per wikipedia:

   ```
   The Apache HTTP Server, colloquially called Apache, is the world's
   most used web server software. Originally based on the NCSA HTTPd
   server, development of Apache began in early 1995 after work on the
   NCSA code stalled. Apache played a key role in the initial growth of
   the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP
   server, and has remained most popular since April 1996. In 2009, it
   became the first web server software to serve more than 100 million
   websites.

   Apache is developed and maintained by an open community of developers
   under the auspices of the Apache Software Foundation. Most commonly
   used on a Unix-like system (usually Linux), the software is available
   for a wide variety of operating systems besides Unix, including
   eComStation, Microsoft Windows, NetWare, OpenVMS, OS/2, and
   TPF. Released under the Apache License, Apache is free and open-source
   software.
   ```

   it is fair to say that apache is at least one of the most popular web servers on the internet. wikipedia.org itself seems to be using a more complex stack involving varnish, an HTTP accelerator, and nginx, an alternative, also quite popular HTTP server. We arrive at this conclusion by inspecting the headers returned by wikipedia.org. In the `http://www.wikipedia.org` case we are redirected to the secure domain (pay attention to the `Server:` line):

```
$ curl -s -D - http://www.wikipedia.org -o /dev/null
HTTP/1.1 301 TLS Redirect
Server: Varnish
[...]
```

And if we directly ask for `https://www.wikipedia.org` nginx seems to be handling our request:

```
$ curl -s -D - https://www.wikipedia.org -o /dev/null
HTTP/1.1 200 OK
Server: nginx/1.9.4
[...]
```

However it is beyond the scope of the project to precisely replicate wikipedia's infrastructure. We focus on the functionality. Therefore due to the popularity, familiarity and by virtue of apace being part of the automatically installable bitnami mediawiki stack, we use it as our server.

2. PHP

   Mediawiki, which is discussed later, is written entirely in PHP, a popular server side, dynamically typed, object oriented scripting language. PHP is essential and is installed along the bitnami mediawiki stack. PHP is popular among web developers partly due to it's support for multiple relational database libraries (including PostgreSQL, MySQL, Microsoft SQL Server and SQLite) and it essentially being structred as a template language generating HTML.

3. MySQL

   Mediawiki can use a number of different SQL database backends:

   - **MSSQL:** An SQL database by Microsoft
   - **MySQL:** Using the standard PHP library for MySQL.
   - **MySQLi:** An extension to the MySQL backend
   - **Oracle:** A propertiary SQL database by Oracle.
   - **SQLite:** An SQL database that is typically accessed as a library rather than over a client-server scheme as is the case with the other options on the list.

   Wikipedia provides multiple dump files for SQL tables of secondary importance in MySQL format (eg. page redirects, categories etc) and suggests `mwdumper` which parses the XML dumpls of the wikipedia articles into MySQL. That and bitnami providing it as part of it's automatically built stack, make MySQL the obvious choice for the wikipedia-mirror stack.

4. MediaWiki

   Mediawiki is the beating heart of wikipedia.

2.2 Tools

A number of tools were developed in assisting the

1. page$_{\text{remover.c}}$
2. sql-clear.sh
3. utf8thread.c
4. webmonitor.py
5. xml-parse.sh

2.3 Setting up

Following are step by step instructions First, clone the git repo:

```
$ git clone https://github.com/fakedrake/wikipedia-mirror
$ cd wikipedia-mirror
```

At this point in theory one can run `make sql-load-dumps` which will take care of stting up everything needed to load the the database dumps into the working SQL database. Of course for that to happen first a couple of steps need to be carried out:

- Download the wikipedia database dumps in XML format.
- Transform them into a format that MySQL understands.
- Set up the bitnami stack that includes a local install of MySQL
- Load the MySQL dumps into MySQL

All of these steps are encoded as part of the a dependency hierarchy encoded into makefile targets and are in theory taken care of automatically, effectively yielding a functioning wikipedia mirror. However this process is extremely long fragile so it is advised that each of these steps be run individually by hand.

First, download and install bitnami. The following command will fetch an executable from the bitnami website and make a local installation of the bitnami stack discussed above:

```
$ make bmw-install
```

Next step is to make sure `maven`, the java is a software project management and comprehension is installed, required to install and setup mwdumper (see below). You can do that by making sure the following succeeds:

```
$ mvn --version
```

Note: if running on Ubuntu 14.04, you may need to install Maven (for Java) using `sudo apt-get install maven`.

Now everything is installed to automatically download Wikipedia's XML dumps and then convert them to SQL using maven. First maven will be downloaded and built. Then the compressed XML dumps will be downloaded from the wikipedia, they will be uncompressed and finally converted to MySQL dumps using `mwdumper`. This is a fairly lengthy process taking 6 to 11 hours on a typical machine:

```
$ make sql-dump-parts
```

After that's done successfully you can load the SQL dumps to the MySQL database.

```
$ make sql-load-parts
```

Finally the

```
$ make mw-extensions
```

3 Mediawiki Extensions

For mediawiki to act like wikipedia a number of extensions are required. The installation process of such extensions is not automated or streamline. To automatically manage this complexity a mechanism is provided for declaratively installing extensions. To add support for an extension to wikipediabase one needs to add the following code in `Makefile.mwextnesions` (modifying accordingly):

```
MW_EXTENSIONS += newextension
mw-newextension-url = url/to/new/extnesion/package.tar.gz
mw-newextension-php = NewExtensionFile.php
mw-newextension-config = '$$phpConfigVariable = "value";'
```

And wikipedia-mirror will take care of checking if the extension is already installed and if not it will put the right files in the right place and edit the appropriate configuration files. The entry points for managing extensions are (provided that the name of the registered extension is newextension):

```
make mw-print-registered-extensions # Output a list of the
    registed extensions
make mw-newextension-enable          # Install and/or enable the
     extension
make mw-newextension-reinstall       # Reinstall an extension
make mw-newextension-disable         # Disable the extension
make mw-newextension-clean           # Remove the extension
```

All registered extensions will be installed and enabled when wikipedia-mirror is built.

4 Dumps

Wikipedia provides monthly dumps of all it's databases. The bulk of the dumps come in XML format and they need to be encoded into MySQL to be loaded into the wikipedia-mirror database. There are more than one ways to do that.

4.1 PHP script

Mediawiki ships with a utility for importing the XML dumps. However it's use for importing a full blown wikipedia mirror is discouraged due to performance tradeoffs. Instead other tools like mwdumper are recommended that transform the XML dump into MySQL queries that populate the database.

4.2 mwdumper

The recomended tool for translating the XML dumps into MySQL code is mwdumper. Mwdumper is written in java and is shipped separately from mediawiki.

1. Xml sanitizer

2. Article dropper

5 Automation

5.1 Makefiles / laziness

5.2 Shell scripts

5.3 Bitnami

6 Performance

6.1 Compile time

Compile time includes the time it takes for:

- Downloading all the components of a wikipedia server
- The bitnami stack
  - mwdumper
  - mediawiki-extensions
  - Installing and building those components (~1 min)
  - Downloading the wikipedia dumps
  - Preprocessing the dumps (~10 mins)
  - Populating the mysql database (~10 days)

    Builds were done on Infolab's Ashmore. The system's specs are quite high end but the bottleneck was the disk IO so less than 1% of the rest of the available resources were used during the MySQL database population.

1. Attempts to optimizing MySQL

6.2 Runtime

Runtime of wikipedia mirror turned out to be too slow to be useful and therefore the project was eventually abandoned. Namely for the full wikipedia dump of July 2014 the load time for the Barack Obama, not taking advantage of caching was at the order of ~30s.