Lumberyard Demo Setup Instructions

Author: Josh Patterson
josh@cloudera.com

Instructions

- **Hadoop / HBase Quick Setup**
  - Download VMWare
  - Download a VM image
    - http://www.thoughtpolice.co.uk/wmware
    - Centos 5.4 is a great choice, 64bit
    - You may want to crank up memory to 2GB for the instance.
  - Start your VM in VMWare
  - Install Hadoop 0.20 on your VM in pseudo-distributed mode.
    - To make it easy, we'll use CDH3
    - https://wiki.cloudera.com/display/DOC/Hadoop+%28CDH3%29+Quick+Start+Guide
    - Easy way to do this:
      - yum -y install hadoop-0.20-conf-pseudo
    - You will need to upgrade Java (as talked about in the instructions)
    - Will also need to add Cloudera's repository
  - Install HBase 0.90
    - To make it easy, we'll use Cloudera's repository
      - https://wiki.cloudera.com/display/DOC/HBase+Installation
      - Follow instructions closely
- **JMotif setup**
  - Build jmotif from trunk at
    - http://code.google.com/p/jmotif/
    - A build of this is already included in the lib directory
    - We need this for the latest iSAX indexing commits
      - provides the bulk of the mathematics of the indexing scheme.
    - If you want to build jmotif yourself, you'll need at least ant 1.7 and you'll need to set the following env vars
      - export JUNIT_HOME=<junit_home>
      - export FINDBUGS_HOME=<findbugs_home>
      - export WEKA_HOME=<weka_home>
        - you'll need a later weka, preferably 3.7.2
    - To build the source:
      - ant
    - To build the jmotif jar (already one pre-built in the lib dir)

- ant -f jar.build.xml

- Build Lumberyard
    - Use Git to pull down Lumberyard from github
        - git clone git://github.com/jpatanooga/Lumberyard
        - Build
            - use ant to build the Lumberyard jar
            - ant
    - Once we have both jmotif.lib.jar and Lumberyard.jar built, copy both jars into the VM onto the hbase path somewhere.

Run The Lumberyard Demo

- To execute the Lumberyard shell, type:
    - **hbase tv.floe.lumberyard.hbase.isax.index.shell.Shell** <command>
        - To view the help
            - **hbase tv.floe.lumberyard.hbase.isax.index.shell.Shell**
- **Let's create an index of some genomic data**
    - to make this simple, I've included the following data hardcoded in the actual source code
        - To create an index
            - **hbase tv.floe.lumberyard.hbase.isax.index.shell.Shell** CreateIndex "my_index"
                - -base_card 4
                - -dim_split 1
                - -base_word_len 8
                - -base_ts_sample_size 16
                - -split_threshold 100
            - What do these settings mean?
                - -base_card
                    - base cardinality
                - -dim_split
                    - number of dimensions to use per split
                - -base_word_len
                    - the base word length for the index, meaning "iSAX word"
                - -base_ts_sample_size
                    - how wide of a window in samples that we should use to scan through
                - -split_threshold
                    - in the original paper, this param corresponds to "th", which is the threshold for which we split nodes when they get too full

- If you want to play with some more genomic data, take a look at one of the two DNA files in the data subdir:
    - in the data subdirectory, we have the mitochondrial DNA of both the polar bear and the Hippo.
    - These samples were downloaded from http://www.cs.ucr.edu/~eamonn/iSAX/iSAX.html

- For a simple technical demo, let's search our sample index:
    - **hbase tv.floe.lumberyard.hbase.isax.index.shell.Shell** SearchGenomeIndex "my_index" "TTAGAACAGTACTCTG"
        - which should find two results:
        - `Found:`
            - `For 'ttagaacagtactctg' we found >`
                - `--- Occurence in source 'chimp+16' at offset 16`
                - `--- Occurence in source 'human+16' at offset 16`
    - If we search for "TTAGAACAGTACTCTA", which is one letter off, **we find the very same results** due to how the iSAX word is generated (becoming the same iSAX word for both queries)
        - this is because both sequences are converted into numbers and then converted to PAA and iSAX representations. both end in the same node key, at least in this example (the node could split in larger collections of data)
- If we search for a slightly different pattern, we'll see its not found:
    - **hbase tv.floe.lumberyard.hbase.isax.index.shell.Shell** SearchGenomeIndex "my_index" "TGGACTAAATGACTAG"

- The current demo
    - is built to demonstrate iSAX indexing and iSAX indexing persisted in HBase.
    - is meant to build indexes from a single thread
        - parallel index construction from Map Reduce is coming, requires some distributed lock coordination
    - should not be counted on for more than prototyping purposes
    - is a proof of concept for scalable index construction and search

Appendix: Bugs, Quirks, and Stuff

- Missing Jars
    - Hackstat?
        - copy the hackstat stuff from the Jmotif project to a location on the hbase path
- “Too many open files”
    - hadoop issue
    - resolve (generally) by setting the Ulimit in linux
        - ulimit -n 32768

References

- Based on the original paper by Shieh and Keogh:
    - http://www.cs.ucr.edu/~eamonn/iSAX.pdf
- iSAX homepage:
    - http://www.cs.ucr.edu/~eamonn/iSAX/iSAX.html