Structuring All-Pairs as a MapReduce Application

Cyril Briquet, Stéfan Sinclair
McMaster University, Hamilton, Ontario

http://voyeurtools.org/

1. Problem description

All-Pairs problem (Thain 2008):
<< All-Pairs(set A, set B, function F) returns matrix M:
Compare all elements of set A to all elements of set B
via function F, yielding matrix M,
such that M[i,j] = F(A[i],B[j]) >>

* challenge at scale: transferring large amounts of data

<< estimate document similarity for all 40 billion pairs of Old Bailey records >>

* each record (full text + id) = 4 kilobytes
* 40 billion of record pairs = 320 terabytes
(unoptimized approach)

2. Application scenario

Old Bailey corpus (London court, 17th-20th cent.):
* largest historical collection of judicial records
* 200,000 TEI-encoded XML documents

3. Cloud environment

* Voyeur Tools cloud-based text analytics
* web interface http://voyeurtools.org/

* compute cluster: 224 cores, 1 GB RAM per core
* multiple filesystems: HDFS, Lustre, local
* implementation: Java, Apache Hadoop

4. Dictionary data as the main input data transfer channel

* original insight: All-Pairs does not require MapReduce to handle lots of data, only lots of computations
* typical MapReduce app: partition input data into key/value MapReduce records
* our approach: provide MapReduce with data designation information only, transfer input data as dictionary data, using an external data transfer channel

shared storage
input records
mapper
shared storage
MapReduce daemon
local storage
shuffle
dictionary data
reducer
local storage
MapReduce daemon
shared storage
output records
shared storage

40 billion record id
* packaged into many MapReduce input files
(so that each map task runs ~20 minutes)
* ~5GB (448 files, ~10 MB each)
* transferred using HDFS

200,000 record full texts
* ~800 MB = 0.004% of unoptimized approach
* transferred using HDFS, Lustre or (soon...) BitTorrent, but not Hadoop distributed cache (does not scale)

* in practice, each mapper maintains an in-memory hierarchical cache to minimize accesses to the file system
* total order of the output data (comparisons of record pairs) guaranteed by passing an output shard index through the mapper and reducer keys
(each record pair preassigned to an output shard before invoking MapReduce)

5. Conclusion

* a structuring of All-Pairs as a MapReduce application has been presented
* summary: MapReduce transfers only the designation of input data, external channel transfers the actual input data
* approach valid because of: (1) problem structure, (2) availability of high performance file system / file sharing system
* All-Pairs for 20% of Old Bailey corpus is computed in 40 minutes on current cluster
* experiments needed to determine conditions under which to use HDFS, Lustre or BitTorrent

Acknowledgments
* Digging into Data (www.diggingintodata.org)
* Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca) and Compute/Calcul Canada
* Tango icons library (http://tango.freedesktop.org/), under Creative Commons Attribution Share-Alike license