system storage. Moreover, the Platform is in charge of on-demand shipping and deploying the required preservation tools on the cluster nodes that hold the data. This is being facilitated by employing a software packaging model and a corresponding repository. The SCAPE vision is that preservation workflows, based on assembling components using the Taverna graphical workbench, may be created on desktop computers by end-users (such as data curators). The Platform's execution system will support the transformation of these workflows into programs that can be executed on a distributed data processing environment.

Scalable Planning and Monitoring

Through its data-centric execution platform, SCAPE will substantially improve scalability for handling massive amounts of data and for ensuring quality assurance without human intervention. But fundamentally, for a system to be truly operational on a large scale, all components involved need to scale up. Only scalable monitoring and decision making enables automated, large-scale systems operation by scaling up the control structures, policies, and processes for monitoring and action. SCAPE will thus address the bottleneck of decision processes and information processing required for decision making. Based on well-established principles and methods, the project will automate now-manual aspects such as constraints modelling, requirements reuse, measurements, and continuous monitoring by integrating existing and evolving information sources and measurements.

Conclusions

At the end of the first project year, the SCAPE project can already offer real solutions to some of the big data challenges outlined above, in the form of a scalable platform design and infrastructure, initial tools and workflows for large-scale content analysis and quality assurance, and an architecture for scalable monitoring and control of large preservation operations. Initial results in the form of deliverables, reports and software are publicly available on the project website, wiki, and Github repository. We are confident that the project will have significant impact over the remaining two and one-half years.

This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).

Links:
http://www.scape-project.eu/
http://wiki.opf-labs.org/display/SP/Home
https://github.com/openplanets/scape

Please contact:
Ross King, AIT Austrian Institute of Technology GmbH
Tel: +43 (0) 50550 4271
E-mail: ross.king@ait.ac.at

Brute Force Information Retrieval Experiments using MapReduce

by Djoerd Hiemstra and Claudia Hauff

MIREX (MapReduce Information Retrieval Experiments) is a software library initially developed by the Database Group of the University of Twente for running large scale information retrieval experiments on clusters of machines. MIREX has been tested on web crawls of up to half a billion web pages, totaling about 12.5 TB of data uncompressed. MIREX shows that the execution of test queries by a brute force linear scan of pages, is a viable alternative to running the test queries on a search engine's inverted index. MIREX is open source and available for others.

Research in the field of information retrieval is often concerned with improving the quality of search systems. The quality of a search system crucially depends on ranking the documents that match a query. To get the best documents ranked in the top results for a query, search engines use numerous statistics on query terms and documents, such as the number of occurrences of a term in the document, the number of occurrences of a term in the collection, the number of hyperlinks pointing at a document, the number of occurrences of a term in the anchor texts of hyperlinks pointing at a document, etc. New ranking ideas are tested off-line on query sets with human rated documents. If such ideas are radically new, experimentally testing them might require a non-trivial amount of coding to change an existing search engine. If, for instance, a new idea requires information that is not currently in the search engine's inverted index, then the researcher has to re-index the data or even recode parts of the system's indexing facilities, and possibly recode the query processing facilities that access this information. If the new idea requires query processing techniques that are not supported by the search engine (for instance sliding windows, phrases, or structured query expansion) even more work has to be done.

Instead of using the indexing facilities of the search engine, we propose to use MapReduce to test new retrieval approaches by sequentially scanning all documents. Some of the advantages of this method are: 1) Researchers spend less time on coding and debugging new experimental retrieval approaches; 2) It is easy to include new information in the ranking algorithm, even if that information would not normally be included in the search engine's inverted index; 3) Researchers are able to oversee all or most of the code used in the experiment; 4) Large-scale experiments can be done in reasonable time.

MapReduce was developed at Google as a framework for batch processing of large data sets on clusters of commodity machines. Users of the framework implement a mapper function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reducer function that processes intermediate values associated with the same intermediate key. For the example of simply counting the number of terms occurring across the entire collection of documents, the mapper takes as input a document URL (key) and the document content (value) and outputs pairs of term and term count in the document. The reducer then aggregates all term counts of a term together and outputs the number of occurrences of each term in the collection. Our experiments are made of several such MapReduce programs: We extract anchor texts from web pages, we gather global statistics for terms that occur in our test queries, we remove spam pages, and we run a search experiment by reading web pages one at a time, and on each page we execute all test queries. Sequential scanning allows us to do almost anything we like, for instance sophisticated natural language processing. If the new approach is successful, it will have to be implemented in a search engine's indexing and querying facilities, but there is no point in making a new index if the experiment is unsuccessful. Researchers at Google and Microsoft have recently reported on similar experimental infrastructures.

When implementing a MapReduce program, users do not need to worry about partitioning of the input data, scheduling of tasks across the machines, machine failure, or interprocess communication and logging: All of this is automatically handled by the MapReduce runtime. We use Hadoop: an open source implementation of Google's file system and MapReduce. A small cluster of 15 low cost machines suffices to run experiments on about half a billion web pages, about 12.5 TB of data if uncompressed. To give the reader an idea of the complexity of such an experiment: An experiment that needs two sequential scans of the data requires about 350 lines of code. The experimental code does not need to be maintained: In fact, it should be retained in its original form to provide data provenance and reproducibility of research results. Once the experiment is done, the code is filed in a repository for future reference. We call our code repository MIREX (MapReduce Information Retrieval EXperiments), and it is available as open source software from http://mirex.sourceforge.net

Proud researchers and their cluster.

MIREX is sponsored by the Netherlands Organization for Scientific Research NWO, and Yahoo Research, Barcelona.

Links:
MIREX: http://mirex.sourceforge.net
Database Group:
http://db.cs.utwente.nl
Web Information Systems Group:
http://wis.ewi.tudelft.nl

Please contact:
Djoerd Hiemstra
University of Twente, The Netherlands
E-mail: hiemstra@cs.utwente.nl

A Big Data Platform for Large Scale Event Processing

by Vincenzo Gulisano, Ricardo Jimenez-Peris, Marta Patiño-Martinez, Claudio Soriente and Patrick Valduriez

To date, big data applications have focused on the store-and-process paradigm. In this paper we describe an initiative to deal with big data applications for continuous streams of events.

In many emerging applications, the volume of data being streamed is so large that the traditional 'store-then-process' paradigm is either not suitable or too inefficient. Moreover, soft-real time requirements might severely limit the engineering solutions. Many scenarios fit this description. In network security for cloud data centres, for instance, very high volumes of IP packets and events from sensors at firewalls, network switches and routers and servers need to be analyzed and should detect attacks in minimal time, in order to limit the effect of the malicious activity over the IT infrastructure. Similarly, in the fraud department of a credit card company, payment requests should be processed online and need to be processed as quickly as possible in order to provide meaningful results in real-time. An ideal system would detect fraud during the authorization process that lasts hundreds of milliseconds and deny the payment authorization, minimizing the damage to the user and the credit card company.