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.

In this context, researchers have proposed a new computing paradigm called Complex Event Processing. A complex event processor (CEP) is a system designed to process continuous streams of data in near real-time. Data flows in streams that are not stored, but are rather processed on-the-fly.

Similar to database management systems (DBMS), a CEP processes queries over tuples. However, while in the context of DMBS the set of tuples to be processed is fairly static, CEP deals with an infinite sequence of events. Data processing is performed through continuous queries based on the sliding window model. This approach differs from queries in traditional DBMS because a continuous query is constantly 'standing' over the streaming events and results are output any time the actual data satisfies the query predicate. A continuous query is modelled as a graph where edges identify data flows and nodes represent operators that process input data.

Centralized CEPs suffered from single node bottlenecks and were quickly replaced by distributed CEPs where the query was distributed across several nodes, in order to decrease the per-node tuple processing time and increase the overall throughput. Nevertheless, each node of a distributed CEP must process the whole input flow, which severely limits scalability and application scope.

The real research challenge is how to build a parallel-distributed CEP where data is partitioned across processing nodes that (i) does not require any node to process the whole input and (ii) provides the same results of an ideal centralized execution (ie without any delay due to input tuples queuing up).

The gist of the problem is how to distribute input tuples, so that tuples that must be aggregated or joined together are actually received by the same processing node.

Moreover, a parallel CEP should also feature elasticity in order to adapt the amount of computing resources to the actual workload and achieve cost-effectiveness. Indeed, any parallel system with a static number of processing nodes might experience under-provisioning (ie the overall computing power is not enough to handle the input load) or over-provisioning (ie the current load is lower than the system maximum throughput and some nodes are running below their capacity).

With those goals in mind, we are developing StreamCloud, a parallel-distributed and elastic CEP that delivers unmatched performance in terms of throughput and allows for cost-effective resource utilization. The StreamCloud project is carried out by the Distributed System Lab at Universidad Politecnica de Madrid in collaboration with the Zenith team at Inria and LIRMM, Montpellier. The system is being exercised for a Security Information and Event Management system in the MASSIF project.

StreamCloud leverages a novel parallelization strategy that allows splitting the logical input stream in multiple physical streams that are pushed towards processing nodes. The logical stream is never concentrated in a single node, in order to avoid bottlenecks. Communication between nodes is minimized and only used to guarantee semantic transparency, ie that the outcome of the computation matches the one of a traditional centralized solution. With this parallelization, StreamCloud is able to aggregate the computing power of hundreds of nodes to process millions of events per second.

Further, StreamCloud is elastic and continuously monitors its processing nodes and makes autonomous decisions on whether to add or remove nodes to cope with the incoming load with the minimal set of resources. This is crucial in cloud environments with a pay-per-use model. Node provisioning and decommissioning is complemented by dynamic load balancing so that StreamCloud can re-distribute the load among processing node in case of uneven load distribution.

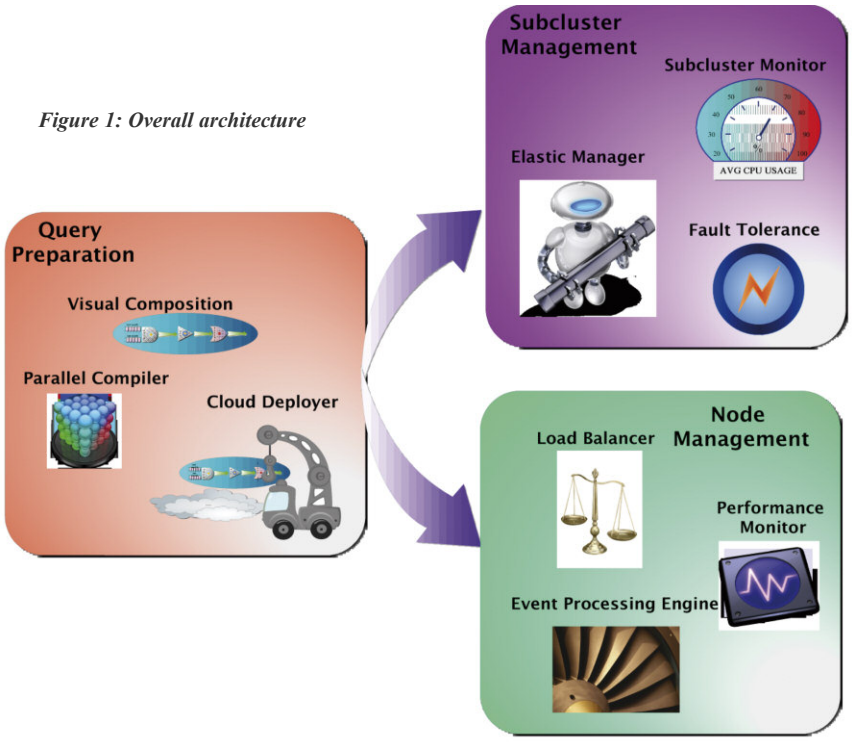

Figure 1: Overall architecture

Link:
http://www.massif-project.eu

Please contact:
Vicenzo Gulisano, Ricardo Jimenez-Peris, Marta Patiño-Martinez, Claudio Soriente, Universidad Politécnica de Madrid, Spain
Tel: +34 913367452
E-mail: vgulisano@fi.upm.es, rjimenez@fi.upm.es, mpatino@fi.upm.es, csoriente@fi.upm.es

Patrick Valduriez
Inria, LIRMM, France
Tel: +33 467149726
E-mail: Patrick.Valduriez@inria.fr