Scalable Data Analysis and Query Processing

Kamlakar Tiwari

April 8, 2022

Abstract

Many developing technologies require large amounts of scalable data. Query processing is crucial in achieving scalability. The goal is to maximise output while reducing execution time. This could be done with any data and any platform using modern algorithms.

1 Introduction

Query management is crucial to the success of a database. It completes the process of taking query input, processing it, and producing output connected to it. It's not as straightforward as a one-line description suggests. It entails a wide range of resources as well as a comprehensive procedure for ensuring that the process runs successfully.

2 Data analysis on clouds

Clouds provide elastic services, scalable performance, and scalable data storage to a huge and growing number of users and applications on a daily basis. Clouds, in reality, have broadened the scope of distributed computing systems by delivering advanced Internet services that complement and complete the distributed computing capabilities offered by the Web, Grid systems, and peer-to-peer networks. Most cloud computing applications, in particular, leverage big data repositories within the cloud itself, therefore enormous datasets are processed with minimal latency to successfully extract data analysis models in those circumstances.

Big data is a buzzword for enormous, heterogeneous, and sometimes unstructured digital stuff that is challenging to process using typical data management tools and approaches. The word refers to the complexity and variety of data and data kinds, as well as the requirement for real-time data gathering and processing, as well as the value that smart analytics may provide. However, we must acknowledge that facts are not always valuable in and of themselves; rather, they become valuable when we are able to extract value from them,

i.e., when we can use them to produce discoveries. Smart and scalable analytics techniques, services, programming tools, and applications are required to extract meaningful insights from large digital datasets.

3 Exascale and big data analysis

Because of the vast number of datasets available and the constant evolution of methods and algorithms for extracting knowledge from them, data analysis has taken centre stage. Data analysis solutions are transforming various scientific and industrial fields by using the potential of data mining and machine learning approaches. For example, the amount of data generated by social media on a daily basis is staggering and never-ending. Every day, hundreds of terabytes of data are uploaded to Facebook and Twitter, including hundreds of millions of images.

4 Extreme data sources and scientific computing

Traditional data storage, file systems, and database management systems are being challenged by scalability and performance needs. Because they were not designed to scale beyond a certain threshold, such systems' architectures have reached their limits in handling very big processing workloads requiring petabytes of data. This circumstance necessitates the development of new architectures and analytics platform solutions to analyse large amounts of data in order to extract complicated predictive and descriptive models. Exascale systems, both in terms of hardware and software, have the potential to aid in the development of solutions to these issues.

In the e-health area, vast amounts of patient data are available and can be used for improving medicines, forecasting and tracking health data, and hospital and health centre management. Exascale computing is still potential in other scientific domains where scalable storages and databases are not used/required, but very sophisticated data analysis in this area will necessitate unique hardware/software solutions. Quantum chromodynamics, materials simulation, molecular dynamics, materials design, earthquake simulations, subsurface geophysics, climate forecasting, nuclear energy, and combustion are examples of scientific disciplines where future Exascale computing will be widely used. All of these applications necessitate the employment of advanced models and algorithms to solve complex equation systems, which will be aided by the usage of Exascale systems.

5 Programming models features for exascale data analysis

Scalable data analysis applications in Exascale computing systems are a difficult task that necessitates high-level fine-grain parallel models, appropriate programming tools, and parallel and distributed programming skills. Techniques and skills are required, in particular, for describing task dependencies and inter-task parallelism, creating synchronisation and load balancing mechanisms, dealing with failures, and appropriately managing distributed memory and concurrent communication among a large number of processes. The programming difficulties become considerably more complicated when the target computing infrastructures are heterogeneous and require different libraries and tools to build applications on them. Different scalable programming methods have been proposed to address some of these challenges in data-intensive applications.
Scalable programming models may be categorized by:
1. Their level of abstraction: expressing high-level and low-level programming mechanisms, and
2. How they allow programmers to develop applications: using visual or script-based formalisms.

A programmer can specify only the high-level logic of an application while hiding the low-level elements that aren't necessary for application design, such as infrastructure-dependent execution details, using high-level scalable models. The compiler, which examines the application code and optimises its execution on the underlying infrastructure, aids the programmer in the definition of the application, and the performance of the application is determined by the compiler. Low-level scalable models, on the other hand, allow programmers to engage directly with the computational and storage pieces that make up the underlying infrastructure, allowing them to directly determine the parallelism of their programmes.

6 Exascale programming systems

Exascale systems place additional demands on programming systems in order to support platforms with hundreds of homogenous and heterogeneous cores. Evolutionary approaches for Exascale programming have recently been developed, which extend or adapt classic parallel programming models like as MPI, OpenMP, and MapReduce. If a shared-memory model is utilised, these new frameworks restrict the communication overhead in message forwarding paradigms or the synchronisation control.

7 Exascale programming systems comparison

Several parallel programming paradigms, languages, and libraries are being developed to provide high-level programming interfaces and tools for high-performance application implementation on future Exascale systems. We'll go over the most important proposals and their primary characteristics here.

Distributed memory paradigms, in particular message passing systems, are candidate tools to be utilised as programming systems for Exascale systems because they will be built of millions of processing nodes. MPI is now the most widely used and researched system in this field. Various modifications of this well-known concept are currently being developed, such as Open MPI for Exascale. Pig Latin, Charm++, Legion, PaRSEC, Bulk Synchronous Parallel (BSP), AllScale API, and Enterprise Control Language are examples of distributed memory programming systems (ECL). We can see that some of Pig Latin's parallel operators, such as FILTER, which chooses a set of tuples from a relation based on a criteria, and SPLIT, are similar.

8 Requirements of exascale runtime for data analysis

The balance between sharing data among processing units and computing things locally to save communication and energy costs while maintaining performance and fault tolerance is one of the most essential aspects to consider in applications that run on Exascale systems and analyse enormous datasets. For data intensive/data-driven systems, a scalable programming model built on basic operations must have structures and processes for:

1. To fulfil high throughput requirements, parallel data access permits boosting data access bandwidth by splitting data into many chunks using various ways and accessing several data components in parallel.

2. Fault resiliency is becoming increasingly important as machines grow in size and complexity. Non-local communication must be prepared for a potential failure of one of the communication sides on Exascale systems with a large number of processes; runtimes must include failure handling procedures for recovering from node and communication faults.

3. Data-driven local communication is a technique for reducing data transfer overhead in massively parallel systems with multiple cores; in this situation, data availability among neighbour nodes controls the activities that those nodes do.

9 Concluding remarks and future work

On both the scientific and commercial fronts, cloud-based solutions for big data analysis tools and systems are nearing completion. New Exascale hardware/software solutions, on the other hand, must be researched and created to enable the mining of very large-scale datasets on those new platforms.

Exascale systems place additional demands on application developers and programming systems by requiring them to target architectures with a huge number of homogenous and heterogeneous cores. Other data-oriented challenges like data distribution and mapping, data access, data communication, and synchronisation must be handled alongside general difficulties like energy consumption, multitasking, scheduling, reproducibility, and robustness. Future data analysis programming models, runtime models, and hardware platforms will rely heavily on programming constructs and runtime systems to handle these issues and facilitate the scalable development of real large data analysis applications.

In particular, I've outlined a set of open design issues that are essential for creating Exascale programming systems and scaling them up. The following design considerations, among others, must be made:

1. Application reliability: Data analysis programming models must include constructs and/or mechanisms for handling task and data access failures and for recovering. As new data analysis platforms appear ever larger, the fully reliable operations cannot be implicit and this assumption becomes less credible, therefore explicit solutions must be proposed.

2. Data handling and sharing patterns: When subsets of data are kept in nearby processors and proximity is avoided when data must be transported, data locality mechanisms/constructs, such as near-data computing, must be built and evaluated in large data applications. Data affinity control, data querying (NoSQL technique), worldwide data dissemination, and sharing patterns are among the other challenges.

3. Communication patterns: A good paradigm design should contain communication patterns that allow application-dependent features and data access models, limit data transfer, and make Exascale runtimes and connections easier to manage.

4. Reproducibility requirements: Reproducibility is required for big data processing on massively parallel systems. To facilitate application reproducibility on large-scale computing systems, new data analysis programming frameworks must gather and generate metadata and provenance information regarding algorithm characteristics, software setup, and execution environment.