Distributed Programming Models for Big Data Analytics

D

Rakhi Saxena
Deshbandhu College, University of Delhi, India

INTRODUCTION

The term "Big Data" has seen an explosion in interest in recent years. The major reasons for this attention are: extremely large volumes of data are being generated every second from varied sources such as the Internet, social media, mobile devices, sensors, and so on; cheap memory storage to store large data is abundantly available; and there is a sharp rise in computation capacity (Bell, Gray, & Szalay, 2006). Many organizations continuously store ever increasing large data resulting from interactions with customers, product sales, user recommendations, and so on. For example, at Facebook, as more users joined the social network and as more features were added to the site, the amount of compressed data stored each day went up from 5-6 TB to 10-15 TB in a matter of six months (Thusoo, Shao, Anthony, Borthakur, Jain, Sarma, Murthy, & Liu, 2010).

Timely and cost-effective analytics over this "Big Data" has emerged as a key ingredient for success in many businesses, scientific, and government endeavors (Cohen, Dolan, Dunlap, Hellerstein, & Welton, 2009). For example, Web search engines and social networks capture and analyze every user action on their sites to improve site design, spam and fraud detection, and advertising opportunities (Herodotou, Lim, Luo, Borisov, Dong, Cetin, & Babu, 2011). Other companies use big data analytics to identify the most profitable customers, to optimize their supply chains, and to establish pricing in real time – to dramatically boost their revenues and reputations. Analytics has allowed Amazon to dominate online retailing and turn a profit despite enormous investments in growth and infrastructure (Davenport, 2006).

DOI: 10.4018/978-1-4666-5202-6.ch071

However, analytics on Big Data scale is a non trivial task. The volume of Big Data far exceeds the processing capacity of traditional data mining methodologies. Traditional sequential algorithms that work well with data measured in megabytes and gigabytes fail when presented with large real world data measured in terabytes and petabytes. One solution for Big Data analytics is to develop and execute parallel distributed applications that process large amounts of data spread across many compute nodes. Therefore, a programmer designing and implementing distributed Big Data Analytics algorithms is burdened with additional complexities of distributed, parallel programming.

Inspired by functional programming languages. recent distributed parallel programming models hide the parallelization complexities inside libraries and runtime systems. A programmer is provided high level primitives (API) to express computation tasks; the model automatically parallelizes the tasks and executes them on a cluster of shared nothing commodity compute nodes. These models enhance the productivity of the programmer by permitting focus on the solution of the problem rather than the mundane tasks of parallelization, and hence are suitable for Big Data Analytics.

In this chapter, we present a survey of distributed programming models designed for general purpose computing on commodity clusters.

BACKGROUND

Given the large volume of data, applications that work on big data need to distribute data on a cluster of processors, and processing has to be

carried out in parallel for computation to complete in a reasonable amount of time (Dean, & Ghemawat, 2010). However, building and debugging distributed software remains extremely difficult. Distributed applications require a developer to orchestrate concurrent computation and communication across machines, in a manner that is robust to delays and failures. (Alvaro, Condie, Conway, Elmeleegy, Hellerstein, & Sears, 2010).

Traditional parallel applications are developed using communication-centric message passing frameworks such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) (Ekanayake, Li, Zhang, Gunarathne, Bae, Qiu, & Fox, 2010). In developing such programs, programmer has to deal with issues of concurrent programming such as - scheduling tasks among workers, transferring data between workers, load balancing, synchronization among the workers, and recovering from communication or processor failures. Parallel database systems have also been used for large scale data analytics, however they are expensive, difficult to administer and lack fault-tolerance for long-running queries (Pavlo, Paulson, Rasin, Abadi, DeWitt, Madden, & Stonebraker, 2009).

More recently, data-centric programming models have simplified the development of data parallel, distributed applications and have proven to be effective for implementation of large scale machine learning algorithms (Chu, Kim, Lin, Yu, Bradski, Ng, & Olukotun, 2007; He, Fang, Luo, Govindaraju, & Wang, 2008). These models present users with a simplified interface to express computations and automatically take over the onus of handling the parallelization/ concurrent programming complexities.

The landmark paper from Google introduced the Map Reduce programming model that provides an abstraction layer to simplify the design and implementation of parallel algorithms. Many other extensions of the Map Reduce paradigm intend to overcome limitations or extend functionality of Map Reduce. These models include Map-Reduce-Merge (Yang, Dasdan, Hsiao, & Parker, 2007), Twister (Ekanayake et al., 2010), and PACT (Alexandrov, Ewen, Heimel, Hueske, Kao, Markl, Nijkamp, & Warneke, 2011) among others. Interestingly, because of the simplicity of concept, Map Reduce has spawned a number of other models following similar philoso phies and these include Dryad (Isard, Budiu, Yu, Birrell, & Fetterly, 2007), Sphere (Gu, & Grossman, 2009), and Piccolo (Power, & Li, 2010) among others.

MAIN FOCUS

The growing demand for Big Data Analytics has resulted in the development of novel programming models to handle distributed processing of data. Applications such as log analytics, customer segmentation, and recommender systems work on large existing and historical data and an organization needs to respond quickly to changing market conditions. Such data analytic applications can benefit from a programming model that simplifies fast, efficient parallel processing of these terabytes of data.

Map Reduce is one such simple, yet powerful programming model that enables easy development of scalable applications. The abstraction of these models cuts down on application development time by freeing the programmer to focus on the solution of the problem to solve rather than on managing parallelization of the algorithm.

The original implementation of Map Reduce has severe limitations – Map and Reduce tasks are loosely synchronized, there is no communication between individual Map or Reduce tasks, no iteration or recursion is possible, there is no direct support for multiple job inputs, and it is assumed that different nodes perform jobs at about the same rate. These limitations were rectified by several follow up research works. We survey programming models that have been implemented based on the original idea of Map Reduce and others that are variants of the original concept. We categorize the models surveyed as i) Map Reduce based Models ii) Map Reduce Variants.