Chapter 21
Modeling Big Data Analytics with a Real-Time Executable Specification Language

Amir A. Khwaja
King Faisal University, Saudi Arabia

ABSTRACT

Big data explosion has already happened and the situation is only going to exacerbate with such a high number of data sources and high-end technology prevalent everywhere, generating data at a frantic pace. One of the most important aspects of big data is being able to capture, process, and analyze data as it is happening in real-time to allow real-time business decisions. Alternate approaches must be investigated especially consisting of highly parallel and real-time computations for big data processing. The chapter presents RealSpec real-time specification language that may be used for the modeling of big data analytics due to the inherent language features needed for real-time big data processing such as concurrent processes, multi-threading, resource modeling, timing constraints, and exception handling. The chapter provides an overview of RealSpec and applies the language to a detailed big data event recognition case study to demonstrate language applicability to big data framework and analytics modeling.

INTRODUCTION

Data is growing at a significant rate in the range of exabytes and beyond. This data is structured and unstructured, text based and more richer content in the form of videos, audios, and images, and from various sources such as sensor networks, government data holdings, company databases and public profiles on social network sites (Katina, 2013). Along with other various challenges related to big data such as storage, cost, security, ethics, or management, processing big data perhaps is even more challenging (Kaisler, 2013). One way to mitigate some of the processing challenges is to build prototypes or models of big data analytics that may allow understanding the underlying complexities and verifying functional correctness before actual implementation of the solutions. The sheer volume and velocity of big data is rendering our traditional systems incapable of performing analytics on the data which is constantly in

DOI: 10.4018/978-1-4666-9840-6.ch021

Copyright © 2016, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

motion (Katal, 2013). One of the most important aspects of big data is being able to capture, process, and analyze data as it is happening in real-time. Unlike traditional data stored in some database or data warehouse for later processing, the real-time data is handled and processed as it flows into the system along with timing constraints on the validity and business response to the flowing in data. Even though various approaches have been suggested and introduced to address the problem of analyzing big data in the cloud, it still has been a challenge achieving high performance, better parallelism and real-time efficiency due to the ubiquitous nature of big data as well as the complexity of analytic algorithms (Osman, 2013). Alternate approaches must be investigated especially consisting of highly parallel and real-time computations for big data processing. This chapter will present one such approach using an executable real-time specification language based on the dataflow programming model. The language features will be presented and the language applicability for modeling big data analytics will be demonstrated through a detailed case study.

BACKGROUND

This section provides an overview of the dataflow programming paradigm and introduces the RealSpec real-time specification language.

Dataflow Programming Model

Dataflow programming paradigm was originally motivated by the ability of exploitation of massive parallelism (Johnston, 2004). The dataflow approach has the potential to exploit large scale concurrency efficiently with maximum utilization of computer hardware and distributed networks (Herath, 1988). Dataflow paradigms can employ parallelism both at fine grain instruction level as well as at various coarse grain levels, however, fine grain parallelism has significant overhead (Ackerman, 1982). Due to the fine grain parallelism overhead and the need of big data analysis and processing, this chapter will focus on coarse grain level parallelism.

Dataflow programs are represented as directed graphs. Nodes on a dataflow graph represent logical processing blocks with no side effects and working independently that can be used to express parallelism (Sousa, 2012). Freedom from side effects is necessary for efficient parallel computation (Tesler, 1968). Directed arcs between the nodes represent data dependencies between the nodes. Arcs that flow toward a node are said to be input arcs to that node, while those that flow away are said to be output arcs from that node (Johnston, 2004). The status or intermediate results of each node is kept in a special node level memory that is capable of executing the node when all of the necessary data values have arrived (Ackerman, 1982). When a node in a dataflow network receives all required inputs, the node operation is executed. The node removes the data elements from each input, performs its operation, and places the transformed data on some or all of its output arcs. The node operation then halts and waits for the next set of input data to become executable again. This way instructions are scheduled for execution as soon as their operands are available as compared to the control flow execution model where instructions are only executed when the program counter reaches the instructions regardless of whether they are ready for execution before that. Hence, in a dataflow model, several instructions can potentially execute simultaneously providing the potential for massive instruction level parallelism. Figure 1 provides a simple

21 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/modeling-big-data-analytics-with-a-real-time-executable-specification-language/150177

Related Content

Approaches for Pattern Discovery Using Sequential Data Mining
Manish Guptaand Jiawei Han (2012). *Pattern Discovery Using Sequence Data Mining: Applications and Studies (pp. 137-154).*
www.irma-international.org/chapter/approaches-pattern-discovery-using-sequential/58677

Analysis and Integration of Biological Data: A Data Mining Approach using Neural Networks
Diego Milone, Georgina Stegmayer, Matías Gerard, Laura Kamenetzky, Mariana Lópezand Fernando Carrari (2013). *Data Mining: Concepts, Methodologies, Tools, and Applications (pp. 203-230).*
www.irma-international.org/chapter/analysis-integration-biological-data/73441

Usability Evaluation of Social Media Web Sites and Applications via Eye-Tracking Method
Duygu Mutlu-Bayraktar (2017). *Social Media Data Extraction and Content Analysis (pp. 85-112).*
www.irma-international.org/chapter/usability-evaluation-of-social-media-web-sites-and-applications-via-eye-tracking-method/161960

The Dynamics of Content Popularity in Social Media
Symeon Papadopoulos, Athena Vakaliand Ioannis Kompatsiaris (2010). *International Journal of Data Warehousing and Mining (pp. 20-37).*
www.irma-international.org/article/dynamics-content-popularity-social-media/38952

Mining Spatio-Temporal Trees
Wynne Hsu, Mong Li Leeand Junmei Wang (2008). *Temporal and Spatio-Temporal Data Mining (pp. 209-226).*
www.irma-international.org/chapter/mining-spatio-temporal-trees/30268