Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 63 (2015) 16 – 23

Procedia Computer Science

The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015)

Pervasive Data Processing

Ichiro Satoh

National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan

Abstract

This paper presents a data processing framework for enabling MapReduce approach to be available in pervasive networks, including sensor networks and Internet of Things (IoT). It is unique among other existing MapReduce-based approaches, because it can locally process data maintained on nodes in pervasive networks. It dynamically deploys programs for data processing at the nodes that have the target data as a map step and executes the programs with the local data. Finally, it aggregates the results of the programs to certain nodes as a reduce step. The paper proposes the architecture of the framework and describes its basic performance and application.

Peer-review under responsibility of the Program Chairs

Keywords: Data processing, distributed systems, sensor network

1. Introduction

Pervasive networks connects a variety of devices such as everyday consumer objects and industrial equipment onto the network, enabling information gathering and management of these devices via software to increase efficiency, enable new services, or achieve other health, safety, or environmental benefits. The networks generate large quantities of data that need to be processed and analyzed in real time. They assumed to transfer massive amounts of small message sensor data to data centers or cloud computing environments for processing, because the computational resources of their devices have assumed to be limited. Several approaches to processing a large amount of data at data centers. Among them, MapReduce is one of the most typical and modern computing models for processing large data sets in distributed systems. It was originally studied by Google[2] and inspired by the *map* and *reduce* functions commonly used in parallel list processing (LISP) or functional programming paradigms. *Hadoop*, is one of the most popular implementations of MapReduce and was developed and named by Yahoo!.

* Corresponding author. Tel: +81-3-4212-2546.
E-mail address: ichiro@nii.ac.jp

Peer-review under responsibility of the Program Chairs
doi:10.1016/j.procs.2015.08.307

Processing large quantities of data generated from devices in real time will increase as a proportion of inbound traffics and workloads in networks from pervasive networks to data centers. However, bandwidth of networks between pervasive networks and data centers tend to be slow and unreliable. Pervasive networks generate massive amounts of input data from nodes. Transferring the entirety of that data to a single location for processing will not be technically and economically viable,

However, modern pervasive devices tend to have certain amounts of computational resources. For example, a Raspberry Pi computer, which has been one of the most popular embedded computers, has 32 bit processor (700 MHz), 512 MB memory, and Ethernet port. Therefore, such pervasive devices have potential capabilities to execute a small amount of data processing. In fact, we have already installed and evaluated Hadoop on Raspberry Pi computers with Linux, but its performance is not practical even when the size of the target data is small, e.g., less than 10MB.

Hadoop has been essentially designed for be executed on high performance servers and it is complicated so that it is almost impossible to redesign Hadoop for pervasive devices, e.g., embedded computers. Therefore, this paper is to propose a MapReduce framework available at limited computers and network, e.g., Raspberry Pi computers, independently of Hadoop. The framework has three key ideas. to save computational resources at node. The first is to deploy and execute programs for data processing at nodes that has the target data. The second is to introduce management functions into programs for data processing. The third is to provide KVS for MapReduce processing available with limited memory.

The author proposed another MapReduce framework based on mobile agent technology[6], independently of the previous one except for the notion of the deployment of programs for data processing. The framework proposed in this paper is constructed based on our previous framework but it is designed for executing on embedded computers or IoT devices.

2. Related Work

The tremendous opportunities to gain new and exciting value from big data are compelling for most organizations, but the challenge of managing and transforming it into insights requires new approaches, such as MapReduce processing. It originally supported *map* and *reduce* processes[2]. The first is invoked dividing a large scale data into smaller sub-problems and assigning them to worker nodes. Each worker node processed the smaller sub-problems. The second involves collecting the answers to all the sub-problems and aggregates them as the answer to the original problem it was trying to solve. There have been many attempts to improve Hadoop, which is an implementation of MapReduce by Yahoo! in academic or commercial projects. However, there have been few attempts to implement MapReduce itself except for Hadoop. For example, the Phoenix system[8] and the MATE system[4] supported multiple core processors with shared memory. Also, several researchers have focused on iteratively executing MapReduce efficiently, e.g., Twister[3], Haloop[1], MRAP[7], and SSS[5]. These implementations, except for SSS, assume data in progress to be stored at temporal files rather than key-value stores in data nodes and SSS executes data stored in a key-value store shared from task nodes and then its results the key-value store. They assume data to be stored in high-performance servers for MapReduce processing, instead of in the edges.

Google's MapReduce, Hadoop, and other existing MapReduce implementations have assumed their own distributed file systems, e.g., the Google file system (GFS) and Hadoop file system (HDFS), or shared memory between processors. For example, Hadoop needs to move target data from the external storage systems to HDFS via networks before processing these.

Our MapReduce system does not move data between nodes. Instead, it deploys program codes for defining processing tasks to nodes that have data by using the deployment of components corresponding to the tasks and it executes the codes with their current local data. Hadoop and its extensions are unsuitable in sensor networks and embedded computers, because its file system, HDFS, tends to become a serious bottlenecks in the operation of Hadoop and it often requires wide band networks, which may not be available in sensor nodes or embedded computers. In the literature on sensor networks, The Internet of Things (IoT), and machine-to-machine (M2M) communications, several academic or commercial projects have attempted to support data at the edge, e.g., at sensor nodes and embedded computers. For example, Cisco's *Flog Computing* and EMC's computing intend to integrate cloud computing over the Internet and peripheral computers. However, most of them do not support the aggregation of data generated and processed at the edge.