Journal of Network and Computer Applications 36 (2013) 293–305

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Benefits of using parallelized non-progressive network coding

Minwoo Kim[a], Karam Park[b], Won W. Ro[a,*]

[a] School of Electrical and Electronic Engineering, Yonsei University, 262 Seongsanno, Seodaemun-gu, Seoul, Korea
[b] Platform R&D Team, Mobile Communications, Samsung Electronics, 416, Maetan 3-dong, Yeongtong-gu, Suwon, Gyeonggi-do, Korea

A R T I C L E I N F O

Article history:
Received 1 September 2011
Received in revised form
23 February 2012
Accepted 29 May 2012
Available online 9 June 2012

Keywords:
Network coding
Parallel algorithm
Non-progressive decoder
Tiling algorithm
Matrix inversion
Matrix multiplication

A B S T R A C T

Network coding helps improve communication rate and save bandwidth by performing a special coding at the sending or intermediate nodes. However, encoding/decoding at the nodes creates computation overhead on large input data that causes coding delays. Therefore the progressive method which can hide decoding delay in waiting time is proposed in the previous works. However, the network speed has been greatly accelerated and progressive schemes are no longer the most efficient decoding method. Thus, we present non-progressive decoding algorithm that can be more aggressively parallelized than the progressive network coding, which can diminish the advantages of hidden decoding time of progressive methods by utilizing the multi-core processors. Moreover, the block algorithm implemented by non-progressive decoding helps to reduce cache misses. Through experiments, our scheme which relies on matrix inversion and multiplication shows 46.0% improved execution time and 89.2% last level cache miss reduction compared to the progressive method on multi-core systems.

1. Introduction

The idea of network coding has been first introduced in Ahlswede et al. (2000), which is a technique used to improve communication rate, network security, resource utilization, and save network bandwidth. In fact, the technique makes it possible to achieve maximum broadcast capacity by performing special coding at the nodes. Original data are coded at the source and the intermediate nodes, and the received messages are decoded at the destination nodes to recover the original data. The core idea of network coding is to combine multiple data in order to enhance the throughput of the entire network system (Wang and Li, 2006). That is, network coding seeks maximum possible information flow in a network environment. In other words, considering a specific network speed – fast or slow – performing high-speed network coding contributes in better data acquisition rate. By using network coding, more efficient multicast communication becomes possible even on lossy packet network environment, thus enhancing network performance such as throughput and reliability (Ahlswede et al., 2000; Ho et al., 2003, 2006; Maymounkov et al., 2006; Park et al., 2010; Shojania and Li, 2007; Chou et al., 2003; Koetter and Medard, 2003; Li et al., 2003).

However, despite these advantages of network coding, the computational delay problem arises at the receiving nodes. Since the data are encoded at the sending nodes, the receiving nodes should decode the received data in order to restore the original information (Ahlswede et al., 2000). With a popularity of random linear network coding (Ho et al., 2006; Maymounkov et al., 2006; Park et al., 2010; Shojania and Li, 2007), many decoding processes adopt a variation of Gauss–Jordan elimination, of which the computation complexity is high. Thus, the computation overhead is never negligible, and might produce huge delays especially when the size of data is large. The decoding delay overhead may cancel out the advantages brought by the network coding technique.

To reduce the computational overhead of the decoding process, many works have been done to find the best performing coding schemes (Ho et al., 2003, 2006; Maymounkov et al., 2006; Park et al., 2010; Shojania and Li, 2007; Chou et al., 2003; Koetter and Medard, 2003; Li et al., 2003). Among them, parallelization of decoding algorithms proposed in the previous research (Park et al., 2010; Shojania and Li, 2007) shows significant advances in reducing the decoding time. However, most of these studies rely on progressive schemes for the decoding methods. Progressive decoding scheme is very effective when the network speed is slow, because decoding operation can be hidden in the waiting time of the packets. However, we have found out two weaknesses of the progressive scheme. One is that the parallelization of progressive scheme produces significant amount of thread creation and destruction compared to a non-progressive method.

* Corresponding author. Tel.: +82 2 2123 5769; fax: +82 2 313 2879.
E-mail addresses: kenstars@yonsei.ac.kr (M. Kim), karam.park@samsung.com (K. Park), wro@yonsei.ac.kr (W.W. Ro).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.05.014

The other is that the implementation of aggressive parallelization and cache-aware parallelization of network coding is difficult for the progressive scheme. Since network coding is a time-sensitive application, frequent thread creation and low parallelization results in severe overall performance degradation. Moreover, performance gain of the progressive scheme is not that significant when the network speed is fast, since decoding time cannot be hidden in the waiting time any longer. We conclude that the progressive scheme is not the best way of decoding when the speed of the network is high.

Thus, in this paper we propose non-progressive schemes that wait until all the packets arrive before starting the decoding process. To demonstrate the advantage of non-progressive schemes first, we assume two cases: high network speed environment and lower network speed environment. The progressive decoding method proposed in Park et al. (2010) is analyzed and we compare it with non-progressive decoding schemes assuming different network speeds. For the realization of non-progressive decoding schemes, we focus mainly on parallelization method in order to accelerate the process by reducing the number of thread creation and destruction operations. We have found out that if the decoding time is decreased by aggressive parallelization algorithms, waiting time overhead of the arrival of all the packets can be canceled out due to short decoding delays. This is also very beneficial since the computing resources can be used for other applications until the actual decoding starts. We seek to improve the non-progressive decoding method so that it can diminish the advantages of progressive schemes.

Two types of non-progressive methods are presented in this paper: one relying purely on Gauss–Jordan elimination and another that relies on matrix inversion and multiplication. We adopt blockwise and tiling methods in order to improve data locality. Blockwise Gauss–Jordan elimination is used for the pure Gauss–Jordan decoding method and also for the matrix inversion (Melab et al., 2000; Petiton and Aouad, 2004; Vancea and Vancea, 2008), and tiling scheme is adopted by matrix multiplication (Hunold et al., 2004; Park et al., 2003). These methods modify the general data access pattern in a matrix into a more efficient manner. Then, parallelization of the non-progressive methods is considered. We seek for the processes that can run simultaneously due to the data independence. The improvement made on the decoding process of non-progressive network coding is done by adopting well-known parallelization schemes. Our main contribution of the research work is the development of high speed non-progressive decoder that outperforms progressive decoding methods.

Our proposed algorithm is compared with other decoding methods including purely progressive model (Park et al., 2010; Shojania and Li, 2007) and hybrid model, and its performance is evaluated via experiments on real machines. The execution time and the number of cache misses are measured for the comparison. Through the experiments on real machines, we have found out that the non-progressive decoding method that relies on matrix inversion and multiplication shows its effectiveness. On an average, we achieve 46.0% of performance improvement with the 8-core system compared to the implementation of the progressive decoding scheme which is also parallelized over the 8 cores. Moreover, our method improves cache performance by reducing 70.3% of the L1 cache misses and 89.2% of the L2 (last level) cache misses.

2. Background research

2.1. Advantages of network coding

Figure 1 illustrates a simple one-source two-sink network model with a simple network coding (Ahlswede et al., 2000).

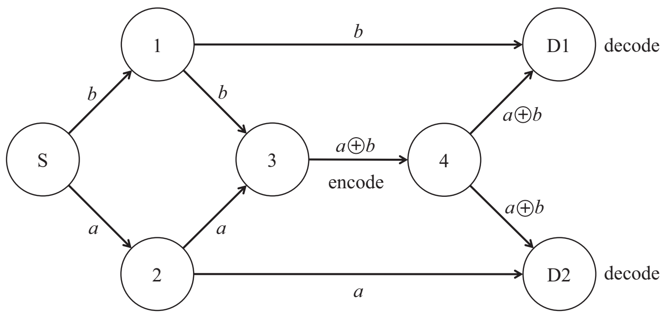

Fig. 1. One-source two-sink network.

Symbols a and b represent information bits which are sent from the source node (S) to both sinks (D1 and D2). As seen from the figure, both a and b arrive at node 3. If the capacity of each edge is the size of one bit, delay will occur since only one bit among $\{a, b\}$ can be sent from node 3 to node 4 at a time. To improve bandwidth efficiency between nodes 3 and 4, network coding can be adopted (Ahlswede et al., 2000).

In the example presented in Fig. 1, the XOR of a and b is calculated (encoding process) at node 3 and the result is sent to node 4. As a result, both bits a and b are received at node D1 with no extra delay, by recovering bit a with b and $a \oplus b$ (decoding process). Bit b can be recovered with the same manner at node D2. Thus, combining the data using network coding technique increases the bandwidth efficiency by enhanced data throughput (Wang and Li, 2006; Maymounkov et al., 2006).

2.2. Encoding and decoding schemes with random linear network coding

The network coding method used in this paper basically follows the schemes introduced in Ho et al. (2003, 2006); Chou et al. (2003); Koetter and Medard (2003); and Li et al. (2003). Especially, we adopt the random linear network coding introduced by Ho et al. (2006) where the sending nodes perform random linear mappings from input data onto outputs of the coded data. All the arithmetic calculation is performed by finite field operations. Addition and subtraction are done by simple XORs and multiplication and division are performed by Galois Field GF(2^8) operations. The reason for using GF operations is to prevent overflows, especially when multiplication is performed. Moreover, remainders that might be produced in division operation can be prevented by finite field operations.

Data that are sent from the source is represented by matrix X. Matrix X is encoded at the sending nodes with being multiplied by some random coefficient matrix A, which in result produces matrix B. Matrices A and B are put together and each row forms the unit of transfer called packet. The concept of encoding is presented in Fig. 2. After the packets have arrived at the receiving nodes, matrix X is decoded from matrices A and B. This basic principle of encoding and decoding is further researched in the other previous literatures (Ho et al., 2006; Chou et al., 2003). Decoding process will be the main issue handled in this paper, and we propose aggressively parallelized decoding algorithms in network coding.

As described in Fig. 2, A is an $n \times n$ matrix and X and B are $n \times m$ matrices. Throughout this paper, the value of m will be larger than n, in most cases $m \geq 4n$ where the value of n span from 64 to 1024. This is because decoding as much information as possible with a single coefficient matrix is more efficient.

To decode the arrived packets and obtain X, we can basically use two different approaches. The first method is a traditional approach which obtains the inverse matrix of A and multiplies it