Future Generation Computer Systems 54 (2016) 326–335

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

High-performance forward error correction: Enabling multi-gigabit flows and beyond on commodity GPU and CPU hardware in presence of packet loss

Milan Kabát [b,*], Vojtěch David [b], Petr Holub [a,b,1], Martin Pulec [a]

[a] CESNET z. s. p. o., Zikova 4, 160 00 Praha, Czech Republic

[b] Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic

HIGHLIGHTS

- We propose parallel design of FEC coding applicable to both CPU and GPU architectures.
- Our scheme can manage multi-gigabit flows.
- We deploy our scheme in low-latency real-time multimedia transmissions over IP networks.
- We provide extensive evaluation in realistic scenarios.

ARTICLE INFO

Article history:
Received 1 May 2014
Received in revised form
15 January 2015
Accepted 18 April 2015
Available online 18 May 2015

Keywords:
Graphics processing unit (GPU)
Forward error correction
Low-density generation matrix
High-definition video
Low latency transmissions
UltraGrid

ABSTRACT

In demanding real-time multimedia transmissions, even a small packet loss might significantly degrade the visual quality. As retransmission is not an option in real-time transfers especially when transmitting the data over long distances, it is necessary to employ mechanisms of Forward Error Correction (FEC). Low-Density Generator Matrix (LDGM) codes are known to be suitable for coding on large block sizes, however, high bitrates of currently used video formats (FullHD, 4K) also require high throughput of FEC coding and decoding. We propose a parallel design of LDGM encoding and decoding algorithms suitable for off-the-shelf, (massively) parallel platforms, such CPUs with vector units or GPUs, and evaluate our approach in real-world scenarios of high-definition and 4K video transmissions. Our results show that offloading FEC computation to such platform is beneficial for low-latency, high-quality multimedia transmissions and may even enable transmissions beyond 10Gbps once the commodity network interfaces reach this speed.

1. Introduction

High-performance media transmissions are becoming more ubiquitous in various fields ranging from scientific visualization environments, medicine, broadcasting or advanced collaborative environments in general. Low-latency high-definition, UltraHD (4K), and even 8K video transmissions have been successfully implemented in the past [1–5], but their practical availability has been limited by available bandwidth in the common networks. Recent advances in efficient implementation of compression on commodity high-performance parallel architectures such as processors (CPUs) with powerful vector units [6], graphics processing units (GPUs) supporting general-purpose computations [7,8], or even dedicated hardware encoders/decoders on the GPUs (mostly H.264) have substantially facilitated adoption of these systems by wider user community. Thus the UltraHD/4K video has recently spread beyond traditional cinematography and entertainment, to fields like large data visualization or medical imaging, which combines various modalities: such as angiology combining X-ray, FFR, OCT, and echo into a complex visualization to support endoscopic heart operations in real time, as discussed in [9].

For low-latency video transmissions, even slight packet loss may impair the quality of the perception substantially as low-latency transmissions have often no means for retransmission especially for long-distance networks. This effect is visible in uncompressed media and becomes even more pronounced for com-

* Corresponding author.
E-mail addresses: kabat@ics.muni.cz (M. Kabát), vojta.david@gmail.com (V. David), hopet@ics.muni.cz (P. Holub), pulec@cesnet.cz (M. Pulec).
[1] Institute of Computer Science, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic.

http://dx.doi.org/10.1016/j.future.2015.04.007
0167-739X/

pressed media, unless some error resiliency scheme is employed. This can be part of the video coding or compression format itself (e.g., Redundant Slices in H.264), or it can be implemented independently as forward error correction (FEC) in the underlying application or network layer.

In this paper, we focus on the FEC for low-latency high-bandwidth media transmission systems, where the FEC is implemented on the application level. This gives us independence on the network types that are being involved in practice. We propose and evaluate a novel fine-grained parallelization of the Low-Density Generation Matrix (LDGM) code, which is a subset of Low-Density Parity Check (LDPC) codes. As we demonstrate, this approach is suitable for coding and decoding on commodity CPU and GPU hardware as of now, with data flows at least up to 10 Gbps. This covers a broad range of applications and media types for now and probably also at least for the near future. When comparing performance of CPUs including their vector processing support and GPUs, it turns out that our parallelization approach is very critical namely for decoding of multi-gigabit flows and the GPUs, which require fine-grained data parallelism, are the only viable option. Compared to more traditional Reed–Solomon FEC codes, which are known to be unsuitable for high-bandwidth video transmissions due to the small block size, the LDGM relies on large blocks [10,11]. In practice, Reed–Solomon coding is used over Galois field of size 2^8, as coding over larger fields leads to high computational complexity. However, the size of the field also poses a limit on the size of the coding block, i.e., the input of the FEC encoder. With $GF(2^8)$, this limit is approximately 128 kB [11] and large input data have to be segmented. However, this approach leads to parity data inefficiency as one parity packet is useful only for recovering data in the relevant coding block. On the other hand, coding over large blocks (e.g., several megabytes long) eliminates this inefficiency. It also helps to mitigate different levels of packet loss burst characteristics (otherwise also known as loss correlation) in the network transmissions, modeled by Gilbert–Elliot model [12] or higher-order Markov models [13,14]. With small block sizes, error burst might eliminate large section of the coding block, even the whole block. In such case, there is no way to reconstruct the lost data. Again, large coding blocks remove this drawback.

The paper is organized as follows: Section 3 summarizes other approaches in using LDGM codes for forward error correction and provides supporting arguments for our approach. Section 2 gives a theoretical overview of LDGM representation and coding algorithms. In Section 4, we introduce our parallel design which is suitable for massively parallel platforms, with its implementation on GPU described in Section 5. The evaluation of our coding scheme under various conditions is provided in Section 6.

2. Theoretical background

In this section, we describe LDGM codes, individual steps in LDGM encoding process, and also introduce GPU architectures and vector processing CPU instructions to provide better insight into our design of the parallel LDGM algorithm in the following section.

2.1. Basic overview of LDGM codes

The Low-Density Generator Matrix code [15] is a linear (n, k)-code with linear encoding complexity and it is a subclass of Low Density Parity Check Codes which were discovered by Gallagher [16]. These codes operate on blocks of data. In one such block, there are k source symbols and during the encoding process, $n - k = m$ parity symbols are created. The ratio between the number of source symbols and the number of *all* symbols created (i.e., source plus parity symbols) is called the *code rate*. LDGM codes can operate on several channels, e.g., binary symmetric channel, binary erasure channel or packet erasure channel. Because the real computer network can be naturally modeled as packet erasure channels when observed from the application-level perspective, we focus on this channel type only in paper.

There are two common representations of the LDGM code: either we can use parity check matrix or a special bipartite graph. The parity check matrix of an (n, k)-code has n columns and m rows and is composed of two parts as can be seen in Fig. 1(a), where $m = n - k$. The left part, i.e., the first k columns, forms a sparse matrix, while the rest is an $m \times m$ identity matrix. Note that the matrix in this figure is clearly not sparse, as we use low values of n and k for clarity reasons.

In the graph representation of LDGM codes, two kinds of nodes are used: *message nodes* (corresponding to source and parity symbols) and *check nodes* (corresponding to rows of parity matrix). For an (n, k)-code, there are n message nodes and $n - k$ check nodes. If the parity check matrix has 1 at the row i and the column j, there is an edge between the message node j and the check node i in the corresponding bipartite graph. The graph representation for the parity check matrix in Fig. 1(a) is depicted in Fig. 1(b), with check nodes in the upper part of the picture.

Standard LDGM was defined using the identity matrix on the right side of the parity matrix. There are two alternative specifications called the LDGM Staircase Code [17] and the LDGM Triangle Code [10], which use staircase matrix (Fig. 2) and lower triangle matrix instead of the identity matrix. Staircase matrix, which is further used in this paper, substantially improves protection of the parity as shown in [10]. This follows from the design of the staircase matrix, where the value of previous parity symbol is added to the next one, except for the first symbol. This way not only a source data symbol, but also a parity symbol might also help in recovering other parity symbols. Triangle matrix further improves protection by a small margin as there are more 1's per row, but at the cost of increased amount of computation for the very same reason.

In [11], LDGM Triangle is suggested as a better option than the Staircase code in cases when channel characteristic is not known. In the paper, the codes were evaluated according to the *average inefficiency ratio*, i.e., the average number of packets necessary to decode. However, this metric is not directly applicable in our scenario—we perform FEC decoding on all data received during a fixed time period, and we are not aware of a Staircase–Triangle comparison for such cases. Hence, taking low latency into account, we chose to use the Staircase version, which should perform better in terms of speed according to [10].

2.2. Encoding process

With LDGM codes, the parity check matrix can be used for encoding. To create the parity symbols, we follow a simple rule: the sum of all symbols connected to a particular variable node must be zero. The staircase matrix contained in the parity check matrix ensures that each constraint node is connected to exactly two parity check nodes (except for the first constraint node, which is only connected to one parity check node). Together with the previous encoding rule, this means that a parity symbol can be created as the sum of all symbols connected to the same check node. In modulo 2 arithmetics, the summation of values corresponds to XOR operation. For example, for a vector of 5 source symbols, the first parity symbol in our example from Fig. 1(a) would be computed as the XOR value of symbols in positions 0 and 1.

2.3. Decoding process

After transmission of the encoded frame, the packet loss is projected into the loss of symbols—each received non-duplicate