International Journal of Research (IJR) Vol-1, Issue-11 December 2014 **ISSN 2348-6848**

Implementation of Linear Network on OpenGL-enabled Cards

Devesh Kasturia ; Akshat Sharma ; Govind Rajput ; Kartik Singh
Students-Dronacharya College of Engineering, India

Abstract—

This paper describes the implementation of network writing on OpenGL-enabled graphics cards. Network writing is a stimulating approach to extend the capability and strength in multi-hop networks. Thisdownside is to implement random linear network writing on mobile devices thatarea unitrestricted in machine power, energy, and memory. Some mobile devices area unit equipped with a 3D graphics accelerator, thatmay well bewont to do most of the RLNC connected calculations. Such a cross-over have already been utilized in computationally hard to pleaseanalysis tasks as in physics or drugs. As a first step the paper focuses on the implementation of RLNC victimization the OpenGL library and NVidia's Cg toolkit on desktop PCs and laptops,manymeasuring results show that the implementation on the graphics accelerator is outperforming the mainframe by a significant margin. The OpenGL implementation performs comparativelyhigher with larger generation sizes because of the parallel nature of GPUs. thus the paper shows AN appealing answer for the longer term to perform network writing on mobile devices.

I. INTRODUCTION AND MOTIVATION

Network committal to writing is obtainingadditional and additional attention of late. once the introduction of the idea by Ahlswede in 2002 [1], an outsizedrange of analysis works has been allottedwanting into the various aspects of network committal to writing. [2] the bulk of analysis papers within the field deals with the development of network codes [13], [9] and therefore the application of network committal to writing to completely different communication situations [7], [11], specifically fixe d networks, meshed wireless network, underwater communica- tion, and plenty ofadditional concept behind network committal to writing is to requiremany original packets and mix them into one coded packet, that has the length of 1 of the initial packets. If a sender is combining N packets, the receiver has to receive a minimum of N coded packets with success to rewritethe initial packets. every coded packet is holding the allegedcoding vector to present the receiver the datathat packets are combined. Thus, as the othercommittal to writingtheme, network committal to writingwill bewont toalter erasures. additionallythereto, network committal to writing offers the chance to rearrange packets within the network. In distinction to differentcommittal to writing schemes that solely work finishto finish, every node has the chance to recombine coded packets into a brand new coded packet. This feature is of nicefacilitate whenever packet flows square measureintersectant as in fixed or meshed wireless networks.In order to know this feature we tend toprovidethe subsequent example. think about the communication topology of a wireless meshed network with five nodes given in Figure 1(a): MD1 is causing packets to MD2 mistreatment MD R as a relay as a result ofthe actual fact that an on the spot communication between MD1 and MD2 isn'tpotential. The communication of MD1 associated MD2 is a component of an in progress stream A, whereas the sender and supplydon't seem to bepictured here. an equivalent holds for MD3 and MD4 engaged on stream B. while not network committal to writing the relay would simplybecome the

bottleneck forwarding packets of stream A and B. just in caseeach streams square measureperforming at the capability limit, the info rate once the relay would be halfthe initial one. mistreatment network committal to writing, we might apply some type oflay flow committal to writingfor every stream. which means packets of stream A square measure coded largelymost likely already at the sender and decoded at the receiver. an equivalent applies for stream B. The interflow committal to writing assures the strength for the given stream. On prime of the lay flow committal to writing, we'll apply associate intra flow committal to writing. as a result of the given situation, MD2 willhear packets of MD3 and MD4 is doing an equivalent with MD1. Whenever MD1 and MD3 square measuretransmission packets of their given stream A and B, the relay MD R canmost likely receive each packets. additionally MD a pair of and MD threecan receive packets even if those packets don't belong to their stream. The relay MD R can take each packets and code these along into a brand new coded packet remarked as packet C. The committal to writingcan bea straightforward XOR of each packets. Packet C are going to be broadcast to MD2 and MD4. At every receiver the packet C are going to beXORedonce more with the packet A or B. E.g. MD4 has packet A before (which failed to belong to stream B), which is able to become packet B once the XORing with packet C . At MD2 the coded packet permits to retrieve packet A. Thus, with one packet transmission, the relay satisfied 2 nodes at an equivalent time.

II. CONCEPT OF RANDOM LINEAR NETWORK CODING

The process of Network writingis divided into 2 separate components, the encoder is accountableto make a coded message from the initial one and therefore the decoder transforms these messages back to the initialformat.The data to be sent is divided into packets and a particularquantityof those packets forms a generation. the fullknowledgemay be divided into additional generations. The generation could be a series of packets that area unit encoded and decoded along. throughout the coding, linear combosof information packets area unitshapedsupported random coefficients. Let N be the quantity of packets in a very generation, and let P be the dimensions of oneknowledge packet. The header acts because the random coefficient matrix, and every encoded packet has the dimensions of N+P bytes.
The encoded packets from a similar generation area unit ag- gregatedalong, containing each the header knowledge, and therefore the encoded payload. once N encoded packets area unit received the encoded knowledgemay be decoded. there'sa smalllikelihood that the generated random coefficients don't seem to be linearly independent,thus the deciphermentdesiresextra encoded knowledge packets to be completed. The decipherment itself is done employing acustomaryGaussian elimination.

III. IMPLEMENTATION

We have enforceda similarruleeach on the processorand also the GPU. Our main goal was to prove that the GPU is capable of secret writing and cryptographyinformationvictimization Random Linear Network committal to writing. Moreover, we tend toneededto check the performance of those implementations. Note that the Evariste GaloisField(28) is employed in each implementations.

A. Reference implementation on the CPU

The C programming language is used to implement the following algorithms on the CPU. The field arithmetic on the Galois Field(28) is realized with two static byte arrays, one for multiplication and one for division. Addition and subtraction are the same operation in this field, and are identical

with th e exclusive OR (XOR) operation on the CPU.

B. The encoding mechanism

After reading N number of P sized messages, the encoder is ready to produce encoded messages for this generation. This N ∗ P bytes of data chunk is stored in a matrix of corresponding dimensions. For the coding it needs a random seed. In this paper we are not going into details about the connection between the entropy of this seed and the quality of the network coding. From this seed the generator gets an N byte long header for each encoded packet, and calculates the payload by multiplying the header as a vector with the data matrix. This operation is realized with simple array lookups and xor operations. The cost of getting a coded message is O(N P). At least N encoded message should be delivered to the decoder side to be able to decode the message. Therefore the total computing cost of transmitting N ∗ P bytes of data is O(N 2P), and the transmitted overload is N ∗ N bytes.

C. The decoding mechanism

Upon receiving a coded message, the received data is being interpreted by using the previous data. Basically the decoding algorithm is a step by step Gaussian eliminationover the Galois Field with back propagation. The elimination is based on the header part of the coded message, but the corresponding operations are also done on the payload part. The decoder stores the received, and partially decoded, data in an N ∗(N +P) size matrix. After the forward substitution part of the elimination each message which carries new information will have a leading column in the header part with a nonzero pivot element, let's mark this column with L. This row is then normalized by dividing all of its elements by the leading value. After this step the new row can be inserted into the decoding matrix to the corresponding row (row L). The last step is to propagate this row back to the existing nonzero rows. The algorithm stops when the matrix does not have any empty rows, thence the header part forms an echelon form, and the payload part of contains the decoded data in order. The cost of encoding a row is O(N (N + P)) for the forward substitution, O(N) for the pivot search, O(N + P) for the normalization and O(N (N + P)) for the backward substitution, therefore totally O(N (N + P)) for each row. The total computational cost is then O(N 3 + N 2P).

D. Basic Operations on the GPU

Porting such a poshformula to the GPU isn'tan easy task, as a result ofthe standard arithmetic operations and management structures don't seem to beaccessible,it'svitalto note that shadersdo notadd a serial manner, pixels square measure rendered in parallel by the presently activated shaders. Directly reading and writing memory in a veryshaderisn'tattainable, the solethanks to store and retrieve information is by mistreatment textures. From our purpose of read, textures behave as two-dimensional arrays of bytes,it'sattainable to browsesureparts of this array by performing arts texture operation operations. 2 dimensional vectors (called texture coordinates) square measureaccustomed specify that array component is to be browse. These texture coordinates square measureperpetually floating-point numbers, thusfurtherautomotive e should be taken to confirm that they're rounded to the properwhole number indices,the foremost serious limitation is that no informationmay be written into a texture in a veryshader. OpenGL solely offers functions to repeatinformation from traditional memory

to a texture and from the framebuffer to a texture

E. Encoder on the GPU The XOR texture is also used here to perform addition over the Galois Field.The encoded image is rendered row-by-row as GL_LINES primitives. Every row is rendered exactly N times using the encoder shader. This shader performs the following steps:

1)Get the multiplication coefficient from the random ma- trix

2)Multiply it with the next row of the original image

3)Xor this product with the previous results for this row

Of course, every row is rendered onto the framebuffer and the resulting pixels must be copied back to the result texture after completing Step 3. So these steps are done N times for each row. After this, the next row of the random matrix is selected to calculate the next line of the result texture.It means NxN line primitives are rendered to encode the whole image, and it would mean NxN OpenGL calls on the CPU. Fortunately OpenGL offers the possibility to pre-compile these API calls into a display list. All these instruc- tions can be executed by calling this pre-compiled display list. It means that the whole image can be encoded by a single API call on the CPU. But before calling this function, the shader and its parameters must be set up properly. The resulting encoded texture serves as the input for the decoder.

F. Decoder on the GPU

The decoder is composed of 3 different shaders that imple- ment the 3 consecutive phases of the Gaussian elimination:

1)phase: Forward substitution: reduce the new data row by the existing rows

2)phase: Find the pivot point in the reduced row

3)phase: Backward substitute this normalized row into the existing rows

The task in the 2nd phase is to find the pivot element (if any) in the reduced new packet. It is implemented using a simple shader which enumerates all elements of this packet, and stores the position and value of the first non-zero element into a special pixel. In this phase rendering is performed point-by- point, because data elements must be processed in a sequential manner. The pivot point position and value is used in the next phase.

G. GUI

We have developed a simple GUI to see the actual results of the computations. This GUI uses both the CPU and the GPU implementation to encode and decode the same data. Thereby it is possible to compare the resulting images and verify the GPU algorithm's correctness. The bottom row shows calculation results on the CPU, whereas the top row shows the results of the same calculations performed on the GPU.

IV. RESULTS

The following measurements are done on different plat- forms with different generation sizes.For simplicity, the packet size is always equal to 1024 (P = 1024). The CPU implementation runs only on one thread, so it utilizes only a single CPU core even if there are more available.The tendency is the same on all platforms: the throughput is approximately a first-order function of the generation size. This observation is in accordance with the cost

formulae presented previously, since the cost of encoding or decoding a single packet (O (N P) and O (N (N + P)) respectively) theoretically determines the achievable throughputs. Decoding performance is about 15-30% lower than encoding perfor- mance. This fact can also be justified theoretically by the same cost formulae, noticing that N + P is only slightly larger than P itself, as N << P in most cases.

V. CONCLUSION

Our work shows that it is possible to implement Random Linear Network Coding on state-of-the-art graphics cards. We intend to port this implementation onto OpenGL-enabled mobile devices. These devices also have a programmable GPU pipeline, but their capabilities are limited compared to graphics cards in the PC. Although these results seem very promising, a more native approach like NVIDIA's CUDA toolkit could yield much better results. Unfortunately, none of the state-of- the-art mobile devices support CUDA.

REFERENCES

[1]R. Ahlswede, NingCai, S.-Y.R. Li, and R.W. Yeung. Network informa- tion flow. Information Theory, IEEE Transactions on, 46(4):1204–1216, Jul 2000.

[2]Christina Fragouli, Jean-Yves Le Boudec, and JörgWidmer . Network coding: an instant primer. SIGCOMM Comput. Commun. Rev., 36(1):63– 68, 2006.

[3]Christina Fragouli and EminaSoljanin. Network Coding Applications. Now Publishers Inc, January 2008.

[4]Janus Heide, Morten V. Pedersen, Frank H.P. Fitzek, and Torben Larsen. Cautious view on network coding - from theory to practice. Journal of Communications and Networks (JCN), 2009.

[5]Tracey Ho and Desmond Lun. Network Coding: An Introduction. Cambridge University Press, 2008.

[6]R. Jacobsen, K. Jakobsen, P. Ingtoft, T. Madsen, and F.H.P. Fitzek. Practical Evaluation of Partial Network Coding in Wireless Sensor Networks. In 4th International Mobile Multimedia Communications Conference (MobiMedia 2008), Oulu, Finland, July 2008. ICTS/ACM.