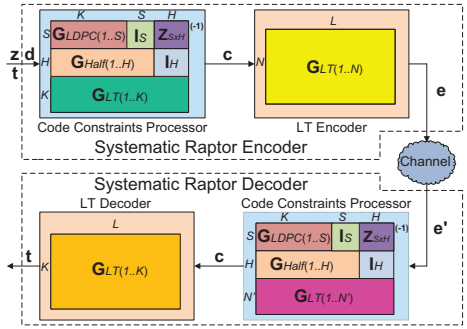
Implementation of Raptor Code on GPU

Linjia Hu[1] Todor Mladenov[2] Saeid Noshabadi[1]
[1]Department of Computer Science, Michigan Technological University, MI, USA
[2] Department of Information and Communication and Department of Nanobio Materials and Electronics, Gwangju Institute of Science and Technology, South Korea

Introduction

Raptor Code principle

Raptor Code come as an improvement to LT-Code, which performs as close as possible to the Shannon's channel limit and provides linear encoding and decoding time. It has been chosen for the forward error correction (FEC) scheme in 3GPP and DVB-H standards. The blow block diagram shows the systematic Raptor encoder and decoder.



At the encoding side, d denotes the input vector of the Raptor encoder, and contains K source symbols and (S + H) zero symbols.

$$d_{[0:L-1]} = [z^T \; t^T]^T \text{ , where } L = K + S + H$$

Code Constrains Processor multiplies d with the inverse of the preprocessor matrix A to produce the intermediate symbols c.

$$c_{[0:L-1]} = A^{-1}{}_{L\times L} \bullet d_{[0:L-1]}$$

LT Encoder can generate any number of encoded symbols e, such that

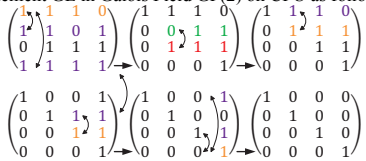
$$G_{LT} \bullet c = e_{[0:N-1]}$$

At the decoding side, we exchanges the positions of Code Constraints Processor and LT Encoder.

$$c_{[0:L-1]} = [z^T e'^T] \bullet A^{-1}{}_{M*L} \quad \text{and} \quad t_{[0:K-1]} = G_{LT} \bullet c_{[0:L-1]}$$

Matrix Inversion on CPU

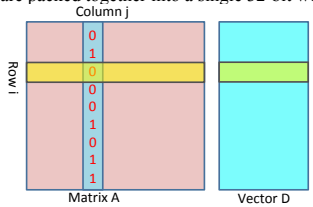
The code profiling of Raptor Code shows that the inversion of the preprocessor matrix A contributes more than 90% of the decoding time, so we concentrate on the optimization of matrix inversion algorithm. The most common matrix inversion algorithm is Gaussian Elimination (GE) , so we implement GE in Galois Field GF(2) on CPU as follows:



Implementation

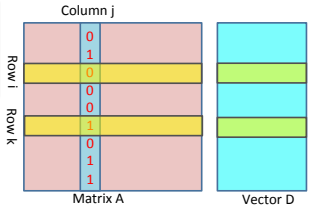
Matrix Inversion on GPU

We try to implement Raptor Codes on GPU for the purpose of processing large block and symbol size efficiently. Recommended by the 3GPP and DVB-H standard, the maximum block size is 8192, and maximum symbols size is 8192 bytes. For the large matrix, we use two types of data to memory mapping . One is "WORD," which uses 32-bit word to store 1 bit matrix element, and the other is "PACKED WORD," in which 32 matrix elements are packed together into a single 32-bit word.

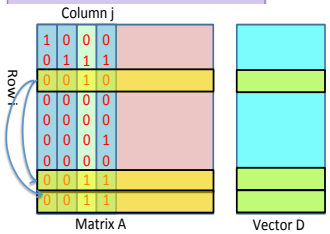


Step-1: (kernel 1)
- find pivot row for forward reduction.
- If A[i, j] == 1, go to Step-3;
- otherwise, go to search the pivot row and record its row index for row exchange , go to Step-2.

Step-2. (kernel 2)
- exchange row i and row k in both A and D for forward reduction.
- One thread processes one pair of elements' exchange.

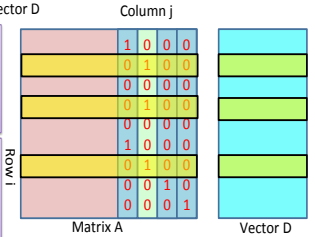


Step-3. (kernel 3)
- extract the current column and copy them to host
- host calculates the number of rows that need to do reduction, and their row indexes for forward reduction.



Step-4. (kernel 4)
- Forward reduction from row i to all the rows that need to do reduction in both matrix A and D
- Send the row i to shared memory of each blocks that do forward reduction.

Step 5: (kernel 5)
- extract the current column(j) to host.
- The host calculate the number of rows that need to be substituted and their row indexes.

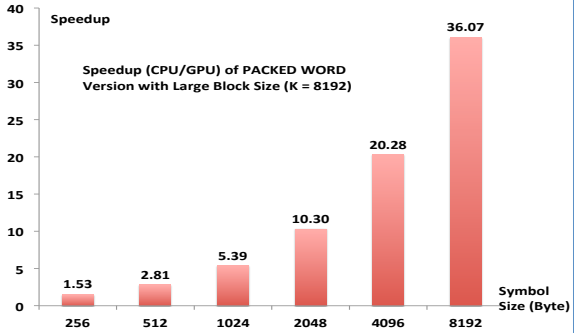


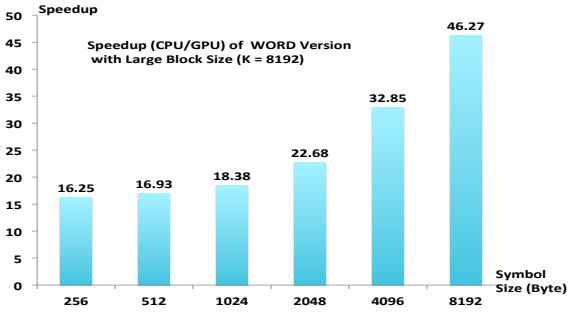
Step 6: (kernel 6)
- backward substitution from row i to all the rows above it that need to be substituted
- just update vector D, the final updated vector D is the solution

Conclusion

Performance

Our test platform uses a 3.20 GHz Intel Core i7 quad-core CPU, a GeForce GTX 570 graphic card with 2.5 GB video memory, CUDA 4.0 and the Fedora 13 operating system. For large block size and symbol size, the speedup of PACKED WORD version decoding can approach 36, and the speedup of WORD version decoding can approach 46.





Reference

[1]T. Mladenov, S. Nooshabadi, and K. Kim, "Implementation and Evaluation of Raptor Codes on Embedded Systems," IEEE Transactions on Computers, Dec. 2011.

[2]T. Mladenov, S. Nooshabadi, and K. Kim, "Strategies for the Design of Raptor Decoding in Broadcast/Multicast Delivery Systems," IEEE Transactions on Consumer Electronics, vol. 56, no. 2, pp. 423-428, May 2010