Performance of `CodornicesRq` software package (first release)

Michael Luby, Lorenz Minder, Pooja Aggarwal
International Computer Science Institute

February 5, 2019

1 Introduction

We provide performance results for the first release of the `CodornicesRq` software package.[1] The test platform is an AMD Ryzen 5 2600 processor with a nominal maximum frequency of 3.4 GHz. All tests are run sequentially using the single threaded `CodornicesRq` library, and thus only one core is used. All tests use a symbol size of $1,280$ bytes $= 10,240$ bits, which is a typical payload size for packets sent over the internet. For example, if $K = 1,000$ for a source block, the source block size is $1,280,000$ bytes $= 10,240,000$ bits.

We consider three different types of operations:

- *Encode:* generate K repair symbols from a source block of K source symbols.
  - Input symbols: source symbols with identifiers $\langle 0, \ldots, K-1 \rangle$
  - Output symbols: repair symbols with identifiers $\langle K, \ldots, 2K-1 \rangle$
- *Decode with no overhead:* generate the K source symbols of a source block from K random repair symbols for that source block.
  - Input symbols: K repair symbols with randomly chosen identifiers
  - Output symbols: K source symbols
- *Decode with* 5% *overhead:* generate the K source symbols of a source block from $M = 1.05 \cdot K$ random repair symbols for that source block.
  - Input symbols: M repair symbols with randomly chosen identifiers
  - Output symbols: K source symbols

[1] We expect substantial performance improvements in future releases.

Decode performance improves as the overhead increases. Performance also improves as the amount of source symbols used to decode increases, as generally it is somewhat faster to decode when source symbols are available (in particular, there is no need to generate the already available source symbols). Thus, *Decode with no overhead* results are essentially worst case, whereas *Decode with 5% overhead* results are closer to what is expected in practice.

As expected, there are no decode failures in tests when decoding with 5% overhead, and there are less than 1% decode failures in tests when decoding with no overhead.

Compile time is defined as the time it takes to compile the program of symbol operations that will be used to generate output symbols from the input symbols for a source block. Thus, the compile time is the time to run `RqInterCompile` plus the time to run `RqOutCompile` for the source block, where `RqInterCompile` and `RqOutCompile` are described in [1]. (The time for the calls to `RqInterGetMemSizes`, `RqInterInit`, `RqInterAddIds`, `RqOutGetMemSizes`, `RqOutInit`, and `RqOutAddIds`, are essentially zero, and thus are not reported.)

Execute time is defined as the time is takes to execute the compiled program of symbol operations to generate the output symbols from the input symbols for a source block. Thus, the execute time is the time to run `RqInterExecute` plus the time to run `RqOutExecute` for the source block, where `RqInterExecute` and `RqOutExecute` are described in [1].

Execute speed, the ratio of the source block size to the execute time, is shown in Figure 1 for various values of K. Execute speed provides an indication of how much of the CPU is needed for execution on average, e.g., when a stream of data arriving at 10 Mbps, less than 0.6% of the CPU is used on average for encode execution if the stream is partitioned into source blocks using $K = 1,000$ and encoded to generate a 20 Mbps stream.

Execute speeds for *Encode* and *Decode with no overhead* are essentially identical, and execute speed for *Decode with 5% overhead* is somewhat faster. Generally, compile time is a small proportion of execute time, as shown in Figure 2. Furthermore, in many use cases, a program can be compiled once and executed multiple times on different source blocks, and thus the compile time can be amortized over many executions.

Figure 3 shows the compile time and execute time raw data from which Figure 1 and Figure 2 are derived. In Figure 3:

- The column labeled "Inter" in the "Compile Time" category is the time to execute `RqInterCompile` per source block.
- The column labeled "Out" in the "Compile Time" category is the time to execute `RqOutCompile` per source block.
- The column labeled "Inter" in the "Execute Time" category is the time to execute `RqInterExecute` per source block.
- The column labeled "Out" in the "Execute Time" category is the time to execute `RqOutExecute` per source block.

References

[1] Michael Luby, Lorenz Minder, Pooja Aggarwal. How to use the CodornicesRq software package. International Computer Science Institute, February 5, 2019.

List of Figures

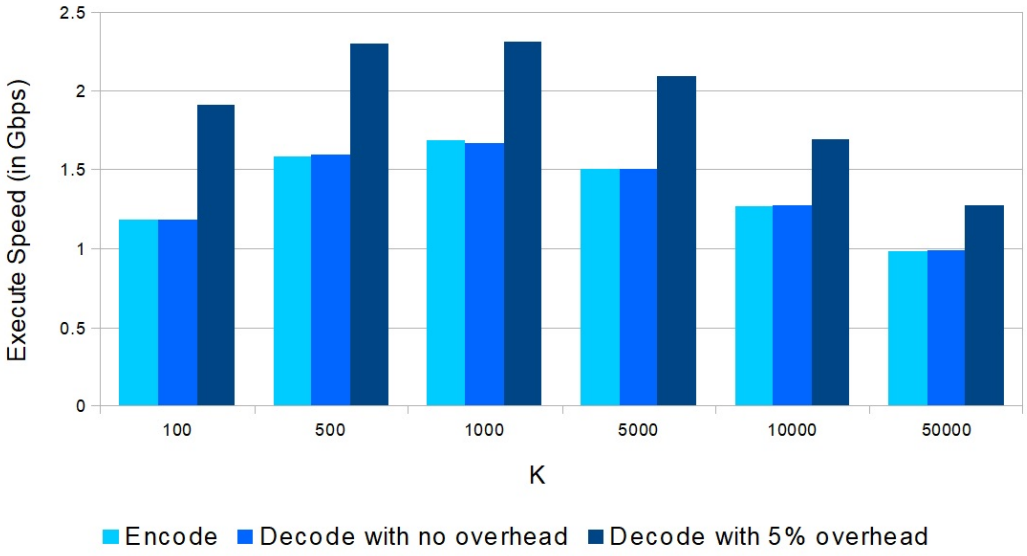

Figure 1: Execute speed for source blocks with K source symbols. The symbol size is $1,280$ bytes $= 10,240$ bits for each source block. The test platform is AMD Ryzen 5 2600 @ 3.4 GHz.

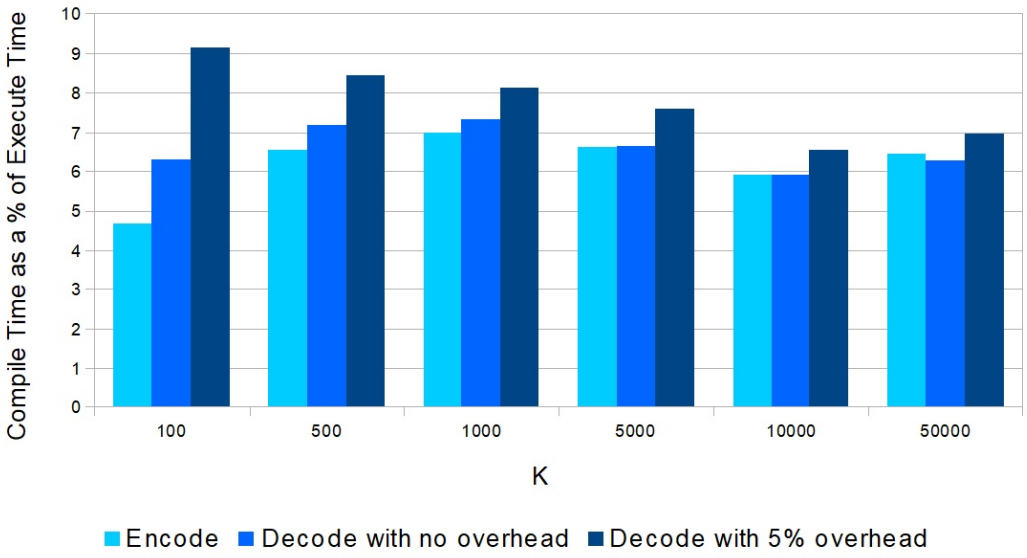

Figure 2: Compile time as a % of execute time for source blocks with K source symbols. The symbol size is $1,280$ bytes $= 10,240$ bits for each source block. The test platform is AMD Ryzen 5 2600 @ 3.4 GHz.

K		Compile Time (in ms)			Execute Time (in ms)		
		Inter	Out	Total	Inter	Out	Total
100	Encode	0.035	0.005	0.040	0.751	0.114	0.865
	Decode with no overhead	0.050	0.005	0.055	0.764	0.100	0.865
	Decode with 5% overhead	0.047	0.005	0.052	0.463	0.100	0.564
500	Encode	0.189	0.023	0.212	2.685	0.554	3.240
	Decode with no overhead	0.208	0.023	0.231	2.633	0.583	3.216
	Decode with 5% overhead	0.175	0.023	0.198	1.764	0.581	2.346
1000	Encode	0.377	0.046	0.423	4.956	1.111	6.068
	Decode with no overhead	0.403	0.046	0.449	4.972	1.162	6.136
	Decode with 5% overhead	0.333	0.045	0.378	3.485	1.164	4.649
5000	Encode	2.036	0.221	2.258	27.875	6.192	34.070
	Decode with no overhead	2.049	0.221	2.270	27.871	6.268	34.141
	Decode with 5% overhead	1.731	0.221	1.952	19.376	6.304	25.682
10000	Encode	4.328	0.439	4.767	64.738	16.030	80.771
	Decode with no overhead	4.323	0.440	4.763	64.355	16.129	80.486
	Decode with 5% overhead	3.722	0.441	4.163	47.619	15.914	63.535
50000	Encode	31.480	2.170	33.650	412.619	109.235	521.857
	Decode with no overhead	30.368	2.165	32.533	408.148	109.661	517.811
	Decode with 5% overhead	27.136	2.169	29.305	311.588	111.050	422.641

Figure 3: Raw timing data (in milliseconds) from which the information in Figure 1 and Figure 2 can be deduced. The symbol size is $1,280$ bytes $= 10,240$ bits for each source block. The test platform is AMD Ryzen 5 2600 @ 3.4 GHz.