Energy Aware Lossless Data Compression

Kenneth C. Barr and Krste Asanović

Introduction: Wireless transmission of a bit can require over 1000 times more energy than a single 32-bit computation. It would therefore seem desirable to perform significant computation to reduce the number of bits transmitted. If the energy required to compress data is less than the energy required to send it, there is a net energy savings and consequently, a longer battery life for portable computers. We report on the energy of lossless data compressors as measured on a StrongARM SA-110 system. We show that with several typical compression tools, there is a net energy *increase* when compression is applied before transmission. Reasons for this increase are explained, and hardware-aware programming optimizations are demonstrated.

Approach: We examine the energy requirements of several lossless data compression schemes using the "Skiff" platform developed by Compaq Cambridge Research Labs. Energy usage for the StrongARM CPU, memory, network card, and peripherals can be measured individually. The platform is similar to the popular Compaq iPAQ handheld computer, so the results are relevant to handheld hardware and developers of embedded software. Several families of compression algorithms are analyzed and characterized, and it is shown that carelessly applying compression prior to transmission may cause an overall energy increase. Behaviors and resource-usage patterns are highlighted which allow for energy-efficient lossless compression of data by applications or network drivers. We focus on situations in which the mixture of high energy network operations and low energy processor operations can be adjusted so that overall energy is lower. This is possible even if the number of total operations, or time to complete them, increases. Finally, a new energy-aware data compression strategy composed of an asymmetric compressor and decompressor is presented and measured. Complete results and analysis, including tests with compressible web data (HTML, Javascript, and CSS from popular websites) appear in [1] and [2].

Progress: To quantify the gap between wireless communication and computation, we have measured wireless idle, send, and receive energies on the Skiff platform. With the measured energy of the transmission and the size of data file, the energy required to send or receive a bit can be derived. The results of these network benchmarks appear in Table 1. Next, a microbenchmark is used to determine the minimum energy for an ADD instruction. From these initial network and ADD measurements, we can conclude that sending a single bit is roughly equivalent to performing 485–1267 ADD operations depending on the quality of the network link. This gap of 2–3 orders of magnitude suggests that much additional effort can be spent trying to reduce a file's size before it is sent or received. However, to acheive high compression, it is generally necessary to store large amounts of context in memory. Table 1 also shows the results of microbenchmarking the Skiff memory system: hitting in the cache requires more energy than an ADD, and a cache miss requires up to 145 times the energy of an ADD. Store misses are less expensive as the SA-110 has a store buffer to batch accesses to memory.

Category	Operation	System Energy Required (nJ)
CPU	32-bit ADD	0.86
Network	Send bit (near)	417.00
	Send bit (far)	1095.00
	Receive bit (near)	329.00
	Receive bit (far)	863.00
Memory	Load Hit	2.72
	Load Miss	125.00
	Load Miss + Writeback	181.00
	Store Hit	2.41
	Store Miss	78.34

Table 1: Energy per operation on the Skiff

Figure 1 shows the energy required for the Skiff to compress and send a 1MB text file. Popular compressors are examined with various parameters. Despite the large communication-to-computation energy ratio, in many cases the energy saved by reducing the amount of data to transmit is outweighed by the cost of performing the compression. Figure 2 shows the reverse operation: receiving data and uncompressing it. One sees that decompression is usually less energy intensive than compression, but does not always yield a net energy savings.

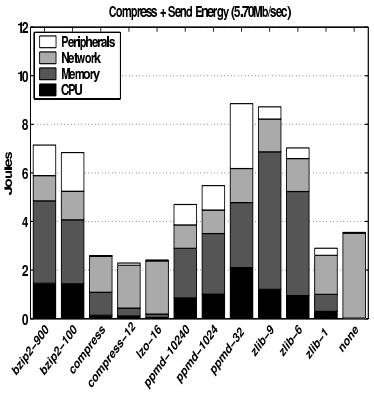

Figure 1: Energy required to transmit (send) 1MB text data using various compressors.

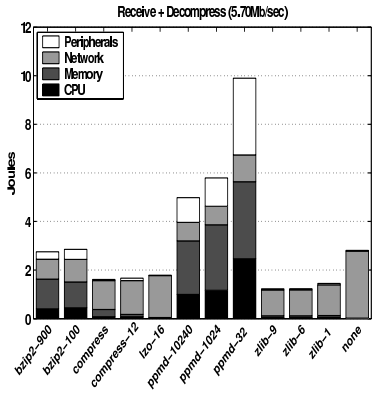

Figure 2: Energy required to transmit (recv.) 1MB text data using various decompressors.

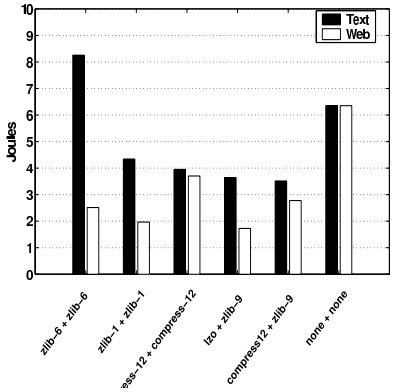

Figure 3: Asymmetric compression: energy to send and receive a compressible 1MB File.

On the Skiff, compression and decompression energy are roughly proportional to execution time. The Skiff requires lots of energy to work with aggressively compressed data due to the amount of high-latency/high-power memory references. However using the fastest-running compressor or decompressor is not necessarily the best choice to minimize *total* transmission energy. For example, during decompression both *zlib* and *compress* run slower than *LZO*, but they receive fewer bits due to better compression so total energy is less than *LZO*. These applications successfully walk the tightrope of computation versus communication cost. Despite the greater energy needed to decompress the data, the decrease in receive energy makes the net operation a win. More importantly, we have shown that reducing energy is not as simple as choosing the fastest or best-compressing program.

Optimizing the way a compressor uses memory can result in significant energy savings. For example, the same amount of data can be spread out across more memory to reduce collisions in a hash table thereby reducing the number of cache misses. Rearranging a structure in the *compress* benchmark to avoid wasted space permitted the program's dictionary to fit in the Skiff's cache. This resulted in 10X fewer cache misses. After a series of optimizations, we were able to reduce the energy of the *compress* benchmark by 51%. We also note that many usage models do not require the compression engine to match the decompression engine. By choosing the best compressor and decompressor for each side of a client-server connection (Figure 3), we were able to reduce energy of text transmission by 57% compared to the common *zlib-6* + *zlib-6* arrangement; a 31% reduction is seen for web data. Even when compared to the minimum symmetric arrangement, asymmetric compression saves up to 12% of energy.

Future: One may wish to examine the second order benefits of compression such as reduction in packet loss and less contention for the fixed wireless bandwidth. When looking at an entire system of wireless devices, it may be reasonable to allow some to individually use more energy in order to minimize the total energy used by the collection. We are also working on a portable, realtime hardware energy monitor, EProf, which could be used to create feedback-driven compression software.

Research Support: This work is supported by MIT Project Oxygen, DARPA PAC/C award F30602-00-2-0562, NSF CAREER award CCR-0093354, and an equipment grant from Intel.

References:

[1] K. Barr and K. Asanović, "Energy aware lossless data compression," M.S. thesis, Massachusetts Institute of Technology, Sept. 2002.

[2] K. Barr and K. Asanović, "Energy aware lossless data compression," in *The First International Conference on Mobile Systems, Applications, and Services*, May 2003.