Extreme Memoization: Everything in a LUT!

Pratyush Patel Luis Ceze

University of Washington

CPUs are not getting any faster, whereas memory and storage continue to scale with new technologies such as NVMs and DNA storage. Based on these trends, we propose that to prepare for an energy-efficient future, we should memoize everything* and share previously computed values at a global level rather than recomputing anything!

Well, *almost everything... We consider the question of when it is worthwhile to memoize: clearly, this should be when the energy consumed for storage and lookups is lower than that spent in computing the function. Although this is only true for large computations today, lookups are likely to get more and more efficient with on-going improvements in networking and storage, while compute remains bottlenecked.

To justify extreme memoization, we take inspiration from the human brain which performs "computation" using extremely energy-efficient memory lookups. We argue that memoization is a good idea by drawing parallels with the two ends of the computing spectrum. On the low-resource end, microprocessors and FPGAs already use lookup tables for floating point computation to improve energy efficiency; On a larger scale, memoization is widespread in the form of caches in CDNs and datacenters.

We describe our approach to create a global system to support large-scale caching of computation. Our plan is to build a worldwide lookup table with keys that represent language-level functions and their inputs, and values that represent their outputs. We expect to set a threshold on the number of function inputs to cache, as this will likely capture the most common inputs. We also plan to only cache functions that are long-running enough to justify the energy tradeoff. We argue that such a system can be supported by a centralized DNA-storage backend with an internet-esque architecture for low-latency memoized lookups.

Finally, we present new research questions that are opened up to support efficient extreme memoization. These range from new methods for data storage and lookups, programming language-agnostic function equivalence (e-graphs), approximate look-up tables, and large-scale hash functions.