SDM@HOME: A Web-Scale experimentation platform for Sparse Distributed Memory

Carl Kugblenu
ckugblenu@gmail.com

Evgeny Osipov
Evgeny.Osipov@ltu.se

Denis Kleyko
Denis.Kleyko@ltu.se

Luleå University of Technology, Sweden

Extended Abstract

Sparse distributed memory (SDM) is a neuro-inspired model of humans' long term memory developed by Kanerva in 1988 [1]. The applications of SDM can be applied for solving the Best Match problem (nearest neighbor search) in a highly parallel manner, i.e. finding the best match to an input word in a dataset of stored words, where a word is a high-dimensional binary vector. Other applications include transcription of speech, vision, and robotics [2]. Kanerva's SDM relies on the hypothesis that the biological memory is high-dimensional (in terms of words encoding the information) and sparse in relation to the whole representational space. High dimensionality (HD) in SDM refers to at least a thousand bits for encoding the data to be stored. The dimensionality of words on the one hand affects the capacity of the memory; on the other hand, it also affects the computational complexity of its implementation. There are several available implementations of SDM [3,4]. All of them are centralized and run on a standalone computer. There are also hardware implementations inclined towards in-memory computing [5]. In this work we present a distributed web-based computing platform for experimenting with truly large SDMs. We believe that this distributed SDM will be of a great interest for researchers working on practical applications as well as for theoretical studies, where a large scale SDM could be used as a model of memory and learning in biological organisms.

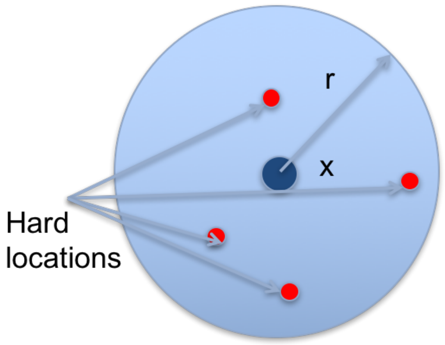

Figure 1. The major concept and storage principle of SDM.

While a comprehensive theoretical description of SDM principles is presented in [1], here we refresh the key operations in order to facilitate the introduction of the distributed implementation. The memory is defined by a tuple <*Addr, Content*>. The memory is autoassociative if *Addr=Content*, and the dimensionalities of *Addr* and *Content* are the same. Thus, SDM can be though as two matrices: one for address words and another for content words. In SDM (see Figure 1) many storage locations participate in a single write or read operation. The addresses of storage locations are fixed (also called *hard locations*) and often distributed randomly in the n-dimensional HD space $\{0,1\}^n$, where n is SDM's dimensionality. The number of storage locations is, however, a tiny fraction of the entire space therefore the memory is sparse. A hard location is initially empty. Once the memory was exposed to input data, the hard location contains a superposition of all words written to it. Writing a new word to an arbitrary address x means updating all hard locations in certain proximity to x. The proximity (or similarity) is measured by Hamming distance between two vectors. If the distance is lower that the threshold r then the location will be updated with the input word. Similarly to the dimensionality of words the value of the threshold affects SDM's capacity and accuracy. Pooling data from address x means pooling contents from all locations accessible from x. As a sensible example, consider an SDM operating with 10^3-dimensional words. For the correct operation, in order to write an input word to approximately 1000 locations – 10^6 hard locations are needed. Therefore, a single operation involves a Hamming-distance calculation through one million of vectors.

The proposed implementation of SDM is inspired by a popular platform for scientific computing – SETI@HOME [6]. Our system assumes a certain set of constantly present nodes for maintaining an operation of SDM with a baseline dimensionality of 10^3. The SDM@HOME platform assumes that parts of the memory will be stored on computing devices (either PCs or smart phones) of participating users. The larger is the number of users participating in distributed SDM the more powerful is the memory will be maintained. We also opt for simplicity of user experience while working with the platform. Therefore, no additional software needs to be installed on users' devices since the entire implementation is Web based. SDM@HOME is built as a hierarchical publish-subscribe Web application as illustrated in Figure 2.

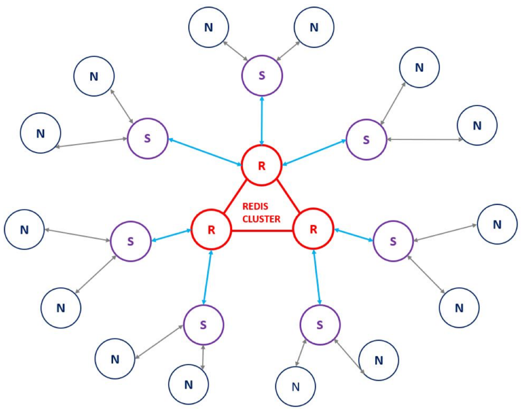

Figure 2. Architecture of SDM@HOME.

The architecture includes Nodes (N), Supernodes (S) and a publish-subscribe system (R). A node N is a web client, which participates in SDM populated by itself and other users. It maintains a small chunk of hard locations. It answers read queries generated by other users by reporting the content of its hard location(s) matching the query. It can also generate and send own data words for storing, as well as generate and send own read queries. The node can view the statistics of SDM performance. The SDM functionality is implemented as a Web worker. It is a JavaScript file executed from an HTML page. It runs in the background, independently of other user-interface scripts that may also have been executed from the same HTML page. The local hard locations and their content are kept even if the node goes offline and are restored when joining the system back. In order to store significant amount of the structured data of the SDM state IndexedDB is used. Static web assets of the user interface are cached locally by a Service worker in order to decrease the load on the supernode on subsequent visits. Supernode is a web socket server for real-time bidirectional event-based communication with the directly connected nodes N. It runs the web server delivering the HTML user interface to the nodes. The publish-subscribe system handles communication and discovery of all supernodes with publish-subscribe message pattern. It is implemented with Amazon AWS ElastiCache service. The supernodes' tier is prepared for on-demand automatic scaling to accommodate dynamically growing population of users. To store a word, a node sends it to the supernode it is directly connected to. The supernode in turn: i) sends the word to other directly connected nodes, ii) publishes the word to the ElastiCache cluster. Other supernodes subscribed to the ElastiCache cluster receive the word and send it to their connected nodes. For the *read* operation, a node sends a query word and a unique query ID for backward routing purpose to its supernode. The supernode in turn: i) sends the word and ID to other connected nodes, ii) publishes the word to ElastiCache cluster for distribution to all nodes via other supernodes. The nodes, which find amongst their local entries one or several hard locations, similar by Hamming distance to the query word respond to the supernode with the content of the found locations. Supernodes use the query ID to find the originating supernode for direct forwarding of the response. The process repeats until it converges or diverges.

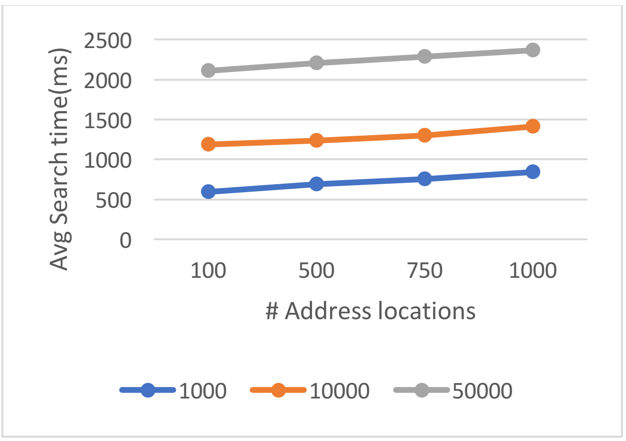

Figure 3. Read/Write time per node for dimensionalities 1000, 10000 and 50000 bits.

The major performance indicator of SDM@HOME is the write and read query times. They heavily depend on the size of the locally stored part of the global SDM as well as the communication delay across the Internet. The graph in Figure 3 depicts a typical processing time in a web browser running on a PC with 1GB RAM (t2.micro instance in AWS) for different dimensionalities of words and varying number of hard locations per node. It provides an indication that a single node can maintain a reasonable portion of hard locations without overloading the host device and to maintain the distributed SDM with realistic number of nodes. As an example: approximately 1000 nodes would be needed for 1000-dimensional SDM. In this SDM each node needs to store 1000 hard locations. In this configuration, the read/write time is less than a second. For higher dimensionalities, the read/write times are somewhat larger, however are still acceptable considering the size of the resulting SDM.

To conclude, we have developed a distributed implementation of Kanerva's Sparse Distributed Memory for experimentation on very large scale. This implementation is subject for future enhancements in terms of improved usability and observability of the global SDM. At NICE workshop, we will demonstrate a working prototype of SDM@HOME and discuss directions for practical experimentation.

References

[1] P. Kanerva, Sparse Distributed Memory, MIT Press, 1988.

[2] M. Allen and P. Fritzsche, "Reinforcement learning with adaptive Kanerva coding for Xpilot game AI," *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, 2011, pp. 1521-1528.

[3] M. Silva, A.Braga, W.Lacerda, “Reconfigurable co-processor for Kanerva's sparse distributed memory”, In Microprocessors and Microsystems, Volume 28, Issue 3, 2004, Pages 127-134

[4] Marcus Tadeu Pinheiro Silva, Antônio Pádua Braga, Wilian Soares Lacerda, Reconfigurable co-processor for Kanerva's sparse distributed memory, In Microprocessors and Microsystems, Volume 28, Issue 3, 2004, Pages 127-134

[5] M. Kang, N.R. Shanbhag W.Lacerda, “In-Memory Computing Architectures for Sparse Distributed Memory”, IEEE Transactions on Biomedical Circuits and Systems, Volume 10, Issue 4, 2016, Pages 855-863

[6] SETI@HOME. Online. https://setiathome.berkeley.edu/