Development of a Connectionist Network Supercomputer[1]

Krste Asanović, James Beck, Jerry Feldman, Nelson Morgan, and John Wawrzynek

University of California and the
International Computer Science Institute
Berkeley, California

June 10, 1993

Abstract

This paper describes an effort at UC Berkeley and the International Computer Science Institute to develop a super-computer for artificial neural network applications. We describe our applications targets, machine goals, and the system architecture.

1 Introduction

The goals of the Connectionist Network Supercomputer-1 (CNS-1) project are to produce super-computer performance and flexible software for connectionist computation at moderate cost. This paper outlines why we have chosen this task and gives the current state of the design.

In a university-based research environment, there are two reasons for undertaking an ambitious development project: either the result is needed or one expects the effort to advance science and technology. Both motivations are equally important for CNS-1. Recent work in our lab and elsewhere has shown the practicality of connectionist systems for a range of important problems, but has also revealed needs for computational resources far exceeding those available to investigators in the field. Our own earlier development of the RAP [?] attached processor has played a crucial role in our research and is proving valuable to other groups who have acquired the system. CNS-1 will supply orders of magnitude more capability in a form that we already know how to exploit.

There are several features of connectionist computation that are exploited in the CNS-1 design. Experiments show that limited precision fixed point arithmetic suffices for almost all algorithms [?]. Many problems are highly regular and are well suited to parallel and pipelined execution. Connectionist networks are "embarassingly" parallel and map nicely to distributed memory machines. Communication is usually multi-cast and normally values only are sent, eliminating the read latency that plagues most distributed computing. Another major source of simplification in the CNS-1 design is that the machine will be used as a single-user, single-task attached processor; this eliminates many of the most complex hardware and software problems in parallel computing and fits our application requirements.

Our experience with connectionist speech research permits a fairly concrete consideration of these issues, particularly in the light of our recent experiences with our earlier machine, the Ring Array Processor. However, we have also considered a number of other appropriate applications such as language processing, auditory modeling, early and high level vision, and knowledge representation. In each of these cases we have good reasons to

[1]Research supported by The Office of Naval Research Grant URI-N00014-92-J-1617, International Computer Science Institute and the National Science Foundation Grant Nos. MIP-8922354 and MIP-8958568.

expect that we can benefit from an architecture that is slanted toward moderate-precision representation, both in raw performance and in price/performance.

A number of these applications require soft-real-time capabilities to handle real-world input such as speech, and many of them could benefit from system implementations using a relatively small number of nodes. In some cases we are interested in interfacing with specific analog interface systems that are part of our research [?]. These requirements provide a strong impetus for users to want a significant say in the design of the computing system.

In addition to the specific connectionist applications of CNS-1, we are very interested in the potential of CNS-1 as a general purpose connectionist accelerator. The connectionist model of computation differs in many ways from conventional models and we exploit these differences in the design. But the mismatch has meant that simulations of connectionist models have been sufficiently clumsy and inefficient to constitute a major barrier to advanced research and development. In some cases, such as speech recognition, we already have clear ideas on how to exploit the capabilities of CNS-1. There are other domains, such as early vision and auditory modeling, where the outlines of promising approaches are understood but the details remain to be worked out. Even more exciting to us are the prospects that a connectionist super-computer will encourage the formulation of problems and models that were not previously taken seriously because of impracticality.

The architectural keystone to supporting the full range of connectionist models is the treatment of sparsely connected and sparsely activated networks. All existing connectionist co-processors, including our own RAP system, have their hardware and software oriented towards densely connected feed forward networks. CNS-1 will still be most efficient for such nets, but considerable design effort has gone into support for general networks. A paradigm case has been human-like semantic memory where the activation of one concept automatically activates related concepts. Such a capability is an essential part of higher level vision and language understanding and could greatly extend the applications of CNS-1 in traditional AI domains. The semantic network is an extreme case of sparseness; many practical problems consist of a mixture of dense and sparse connections and CNS-1 should retain most of its performance benefits for such tasks.

2 Application Targets

We have developed a number of application targets for the CNS-1. These range from extensions of tasks we have already run on parallel computers through bona fide applications that have never been implemented on fast hardware. In addition, we have developed an abstract target application that is representative of classes of problems that we have previously been unable to handle. Together these applications imply a number of system goals for the CNS-1.

2.1 Speech Processing

Within our research group we have been doing large computations for connectionist speech research. In particular, we have been training large Multi-Layer Perceptrons (MLPs) to estimate phonetic probabilities for continuous speech recognition. These techniques are currently quite competitive with the best classical statistical approaches.

Training our networks for the standard DARPA Resource Management task requires on the order of 10^{14} arithmetic operations. Depending on the details of the run, this costs a few days of computing on our current parallel computer, the RAP. New tasks that we are attacking require significantly more computing due to larger databases and larger, more complex networks that are required to handle the more complex recognition tasks we are now working on. Additionally, we need to iterate on these training runs more often in order to test out new robust feature extraction methods. We estimate that we need computational power that is 2 to 3 orders of magnitude larger than we have with the RAP. This translates to a system that is 4 to 5 orders of magnitude more powerful than a Sparc 2 workstation, or perhaps 3–4 orders of magnitude more powerful than a 1995-vintage workstation.

In addition to raw network and training set size, we expect to require a more diversified instruction mix. In particular, there are many good reasons to tightly couple dynamic programming steps with the network evaluations. This suggests that dynamic programming and pointer bookkeeping must be possible within CNS-1.

Finally, as the speech networks (currently over a million connections) get larger, we tend more towards networks that are not fully connected. However, in our experience this sparseness is generally not in the form of random connection vacancies, but rather in the form of fully-connected subnets that are glued together in application-specific ways. Such "chunky" networks must be supported in the CNS-1 machine for the speech recognition research application.

2.2 Other Concrete Tasks

While our greatest experience with parallelizing large network applications is with speech, we have examined a number of other tasks during the design of the CNS-1. These tasks include:

Language processing: Incorporation of syntactic, semantic, and pragmatic knowledge into speech systems such as the one referred to above using connectionist language processing models.

Auditory modeling: instantiations of known physiological or psychological functions; Full implementations require an enormous amount of computation, but are potentially helpful for speech degraded by noise and reverberation.

Early vision: (e.g. for texture recognition) typically requiring no learning, but many convolutional operations.

High level vision: object recognition and analysis, requiring the sparse interconnection (fan-in and fan-out averaging 100-1000) of hundreds of thousands of units.

Large conceptual knowledge representation studies: simulation of large fully associative memories for tasks such as high level vision and language processing.

2.3 A Benchmark Problem

In addition to the concrete tasks described above, it is important to consider a more abstract problem that may clarify some of the costs and tradeoffs resulting from design decisions. To that end, we have defined the following problem description:

Evaluate a network with a million units and an average of a thousand connections per unit for a total of a billion connections. This should be done 100 times per second.

This description is deceptively simple, and does not actually specify the distribution of connections. However, this problem allows us to examine the relationship between this connectivity and the performance for alternate design choices. Since we do not yet know the relation between any form of this problem and the more concrete tasks mentioned earlier, we want to be sure that less common forms of this task are not impossibly inefficient on the CNS-1. However, we suspect that most tasks of interest will have some locality or "clumpiness" to the connection matrix, as we have observed such behavior on most real problems.

2.4 Machine Goals

Given the above target tasks and perspectives, in this section we present a summary of CNS-1 performance goals:

- Connectionist compute power. The minimum configuration should evaluate 100 billion connections per second, which therefore requires a weight-reading bandwidth of 200 billion bytes/second for 2-byte weights.
- Learning capability. Back-propagation learning including weight updates should be accomplished in at worst 1/5 of the evaluation rate.
- Communication capability. For the large abstract problem, we will require a broadcast of 10^6 activations 100 times per second. Each node must then read 10^8 B/sec from the network (the inputs from all units), but write out a much smaller amount (only the outputs from the units represented locally).
- Storage. Two gigabytes are needed to hold a billion 2-byte connections. Including activation tables and pointers for sparse networks, another 2 GB are required. This 4 GB is really a minimum, and larger systems will be required to simultaneously hold large data sets. We expect our initial system to be limited to the 4 GB figure since our current tasks actually use smaller networks (though they have very large data sets, e.g. 1–2 GB), and the large abstract problem does not at this time have a corresponding large input data set.
- Sparseness. A network with arbitrary sparseness should still run on CNS-1, although a network composed of fully connected subnets will be more efficient. The performance degradation with increasingly random sparseness should be gentle, so that a fully random interconnection pattern should be evaluated at a speed that is no worse than 1% of peak, and normally sparse networks should work at greater than 10% efficiency.
- Shared Weights. Many of the tasks described above incorporate shared or tied weights. While in some cases these weights have been tied to save storage, shared weights are a more useful way to enforce properties such as shift-invariance, as well as to reduce parameters for smooth estimators. Shared weight evaluation and learning must be supported.

- Non-connectionist processing. Since many applications require integrating operations that are not fixed point matrix-vector operations, the CNS-1 node must be able to perform general scalar operations at a rate that is only moderately reduced from the peak rates of the more common operations. Some common non-multiply-accumulate vector operations should also be supported.

- Numerical representations. We expect to take advantage of the fact that the variable we process, store, and communicate are short. For most purposes, 1-byte numbers are sufficient to represent unit activations and 2-byte numbers are sufficient to represent connection weights. These common choices generate far-reaching consequences in the reduction of resources to achieve the computational and communication goals described above.

- Floating point support. It is important to implement multiple precision and floating point representations. These will be important for the development stages of fixed point algorithms, when scaling or range properties of the variables may not be well understood. Additionally, the final form of coded algorithms may wish to use some floating point or double precision fixed point.

- Coding capabilities. CNS-1 is intended as a fully programmable computer, and is not just a fixed function back-propagation machine. Software must be developed to give CNS-1 the "look and feel" of a more general computer, if it is to be anything beyond an academic curiosity. The use of carefully hand-coded library routines will provide efficient operation for the common functions that we anticipate. However, another design goal must be graceful performance degradation for the general (non-parallel) operations included in the programmer's code. Provision for clean mechanisms of expression for these programming modes is another key requirement for the system.

- Soft real time capabilities. An important aspect of the machine's intended function is the ability to interface interactively with analog sensors and actuators. For instance, real-time streams of speech and video images must be accepted by the machine and processed roughly within the appropriate frame times. There should be a simple high-performance interface mechanism that can be used to connect CNS-1 to a variety of target external devices.

3 Previous Neurocomputers

A number of parallel computers have been built in the past with connectionist computations as the main target task. Many of these have used special-purpose architectures incorporating commercial DSP chips. Examples of this approach are the NeuroTurbo from Nagoya University[**?**] and our own Ring Array Processor (RAP) machine [**?**]. The RAP has been used since 1990 as an essential component in the development of connectionist algorithms for speech recognition. Implementations of this machine consisted of 4 to 40 TI-TMS320C30 floating-point DSP chips connected via a ring of Xilinx programmable gate arrays, each implementing a simple two-register data pipeline.

Several related efforts are underway to construct programmable digital neurocomputers, most notably the CNAPS chip from Adaptive Solutions [**?**] and the MA-16 chip from Siemens [**?**].

Adaptive Solutions provides a SIMD array with 64 processing elements per chip, in a system with four chips on a board controlled by a common microcode sequencer. As with the CNS-1, processing elements are similar to general purpose DSPs with reduced precision multipliers. Unlike our hardware, the Adaptive Solutions chip provides on-chip SRAM sufficient to hold 128K 16b weights but has no support for off-chip memory. Larger networks require additional processor chips to obtain the required storage, even if the extra processing power cannot be efficiently employed.

Like the CNS-1 hardware, the MA-16 leverages the high density and low cost of commercial memory parts. This chip is a direct realization of three general network formulae that are intended to summarize many connectionist computations. The system that is envisioned will consist of a 2D systolic array containing 256 of these chips, and the resulting system will provide impressive raw peak throughput. Multiple 68040s with additional integer ALUs are used as general-purpose processors to complement the systolic processing array.

Important goals in the CNS-1 design are to achieve high performance on large, irregular network structures and to tightly couple the scalar non-connectionist portions of real applications with efficient connectionist computation. Both the CNAPS and the MA-16 are designed to be cascaded into larger SIMD processor arrays with only auxiliary scalar capabilities; the CNS-1 is MIMD with integrated scalar units.

4 CNS-1 System Overview

4.1 Architecture

The connectionist network supercomputer (CNS-1) is a multiprocessing system designed for connectionist network calculations and other moderate precision integer calculations. The architecture of the CNS-1 system is similar to that of other massively parallel computers; major differences arise in the architectural details of the processing nodes and the communication mechanisms. The hardware is illustrated in Figure **??**. Custom VLSI digital processing nodes are tailored for neural-network calculations. Processing nodes are connected in a mesh topology and operate independently in a MIMD style. Each node contains a private memory space and communicates with others through a simple message passing scheme. The initial implementation will be built with 128 processing nodes giving a maximum computational capacity of 256 billion integer operations per second and a total of 4GB of storage. The design is scalable to 1024 processing nodes for a total of up to two TeraOps and 32GB of RAM. One edge of the communications mesh is reserved for attaching I/O devices, allowing up to 8GB/s of I/O bandwidth. The CNS-1 array is attached as a compute server to a host workstation.

The processor, named Torrent, includes a MIPS CPU with a vector coprocessor running at 125MHz, a Rambus external memory interface with on-chip instruction and data caches, and a network interface.

The vector coprocessor accelerates neural computation. It contains multiple moderate precision fixed-point datapaths that complete up to 16 operations per cycle, yielding up to two billion operations per second. Torrent includes a 32b RISC scalar unit; we find it important to provide tightly coupled scalar processing since many applications have a significant non-neural component. The MIPS CPU provides general scalar processing and supports the vector coprocessor by providing scalar operands, address generation, and loop control.

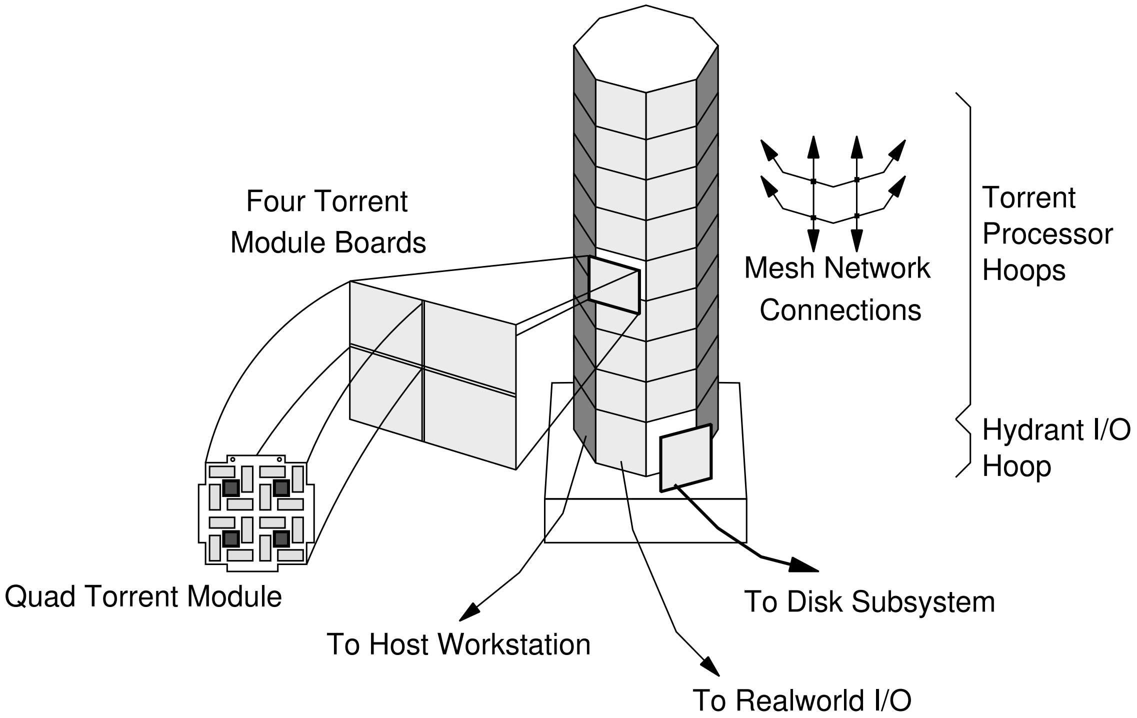

Figure 1: CNS-1 Hardware.

To provide the high memory bandwidth needed to feed the vector unit, the memory interface implements four Rambus channels [?] with an aggregate memory bandwidth of over 1GB/s. Each Torrent processor will have four 18Mb RDRAM chips per Rambus channel giving a per node total of 32MB of RDRAM. An on-chip data cache is included to reduce average memory access latencies, but may be bypassed when accessing vector data to avoid cache pollution. The instruction bandwidth is provided by an on-chip instruction cache.

Torrent includes an on-chip hardware router to handle communications in the CNS-1 mesh. A related and important feature of Torrent is a fast processor-network interface. CNS-1 has some unique requirements for data communication that are not well met by existing massively-parallel computer systems. Effective execution of neural network architectures with sparse interconnections and activations requires efficient communication of small messages. The communication and management of continuous sensory data streams require rapid processor response to asynchronous external events. Torrent has a fast and simple network interface with messages sent and received directly from processor registers. The active message model [?] is directly supported, where an arriving message triggers execution of a local event handler.

System design is simplified because each processing node comprises only a single chip processor and DRAM. No external logic is necessary to complete a node, increasing reliability and minimizing the board area required per processor.

The host and other devices will connect to the processors of the CNS-1 through custom VLSI I/O nodes attached to an edge of the communications mesh. The I/O node, named Hydrant, includes the same network router as Torrent together with a set of message send and receive buffers that can be accessed over a conventional parallel interface. Additional external components can be used to customize a Hydrant to provide a wide variety of I/O interfaces, including standard peripheral interfaces, such as HIPPI and SCSI, as well as custom interfaces to sensors and actuators. In early versions of the machine the host will provide access to disk storage. Later improvements will include direct interfaces to fast mass-storage subsystems without host intervention.

A mesh topology is used for the data network connecting Torrent and Hydrant nodes. A network link between neighboring nodes consists of 8b of data and an acknowledgment wire in each direction. The data network is synchronous with a clock rate of 125MHz, giving a peak data throughput of 125MB/s in each direction per link. The largest CNS-1 system with 1024 Torrent nodes has a bisection bandwidth of 8GB/s, or 8MB/s per processor.

A bit-serial diagnostic network is provided for hardware diagnostics, bootstrapping, and run-time monitoring. This diagnostic network is a high-performance extension of the industry standard JTAG and is controlled by the host workstation.

4.2 Software

A considerable amount of software development is required to make CNS-1 a usable system. At the lowest level we require diagnostic routines that will be used to detect hardware failures in the machine. These routines are written for the host system and take over CNS-1 for the duration of the diagnostic tests.

Users start their applications from the host by invoking a `cnsserver` and giving it the name of a CNS-1 executable. The `cnsserver` process first resets, then performs a bootstrap sequence which ends by downloading the executable to all processing nodes in the array. The application is started by a message sent from the `cnsserver` at this point. `cnsserver` then enters a monitoring loop, repeatedly scanning out error flags and profiling information across the diagnostic network. In the early CNS-1 system, the host is also used for file I/O, and so `cnsserver` may be interrupted with system I/O requests from the array. Finally, the application will send an exit message back to `cnsserver` which will then clean up on the host and release the machine for the next user.

A number of languages will be supported by CNS-1. A Torrent assembler will be an important tool in the development of the optimized libraries. Both C and C++ will be made available by a port of the GNU `gcc` compiler to the Torrent processor. This C compiler will then be used as the target for a port of Sather [?]. Parallel language support for distributed memory machines is still very much a research issue. We expect to use experience gained in a pSather [?] port to the CM-5 to port pSather to CNS-1. The CNS-1 hardware is simple, fast, and flexible, and should prove an interesting vehicle for further research into parallel languages.

CNS-1 is an application specific computer, and an important component of the project is the development of software libraries for the applications we have in mind. These libraries should allow much of the peak performance of the machine to be made available to connectionist researchers in a straightforward and flexible manner. These connectionist libraries will be based on connectionist simulators that have evolved over several generations at ICSI, including parallel versions for the RAP.

The capabilities of CNS-1 might go unused unless a comfortable programming environment is developed. In particular, we see the need for powerful debugging tools to detect programming errors and accurate profiling tools to discover performance bottlenecks. The development of CNS-1 software is a major task that must be performed together with hardware design. Accurate hardware simulators will be made available for software development and feedback during hardware development. These should also be useful after hardware design as accurate performance predictors and debugging tools.

5 Acknowledgments

In addition to the authors, the primary CNS-1 design team members are Tim Callahan, Bertrand Irissou, Brian Kingsbury, Phil Kohn, John Lazzaro, Thomas Schwair, Carlo Séquin, and David Stoutamire. The National Science Foundation provided financial support for through Grant No. MIP-8922354, and with Graduate Fellowships. John Wawrzynek received support from the National Science Foundation through the Presidential Young Investigator (PYI) award, MIP-8958568. Primary funding for the project is from the ONR, URI No. N00014-92-J-1617 and the International Computer Science Institute.

References

[AM91] Krste Asanović and Nelson Morgan. Experimental Determination of Precision Requirements for Back-Propagation Training of Artificial Neural Networks. In *Proceedings 2nd International Conference on Microelectronics for Neural Networks*, Munich, October 1991.

[FLR92] Jerome A. Feldman, Chu-Cheow Lim, and Thomas Rauber. The Shared-Memory Langauge pSather on a Distributed-Memory Multiprocessor. In *Second Workshop on Languages, Compilers, and Run-Time Environments for Distributed-Memory Multiprocessors*, Boulder, Colorado, Sept 30 - Oct 2 1992.

[Ham90] D. Hammerstrom. A VLSI architecture for High-Performance, Low-Cost, On-Chip Learning. In *Proc. International Joint Conference on Neural Networks*, pages II–537–543, 1990.

[IYM+89] A. Iwata, Y. Yoshida, S. Matsuda, Y. Sato, and N. Suzumura. An Artificial Neural Network Accelerator using General Purpose Floating Point Digital Signal Processors. In *Proceeding JCNN*, pages II–171–175, 1989.

[LWM+92] John Lazzaro, John Wawrzynek, Misha Mahowald, Massimo Sivilotti, and David Gillespie. Silicon Auditory Processor as Computer Peripherals. In *Advances in Neural Information Processings Systems, Proceedings of the 1992 Conference*, December 1992.

[MBK+92] Nelson Morgan, James Beck, Phil Kohn, Jeff Bilmes, Eric Allman, and Jochaim Beer. The Ring Array Processor(RAP): A Multiprocessing Peripheral for Connectionist Applications. *Journal of Parallel and Distributed Computing, Special Issue on Neural Networks*, 14:248–259, 1992.

[Omo91] Stephen M. Omohundro. The Sather Language. Technical report, International Computer Science Institute, Berkeley, Ca., 1991.

[Ram92] Rambus Inc., 2465 Latham Street, Mountain View, California, USA. *Rambus Technology Guide, Preliminary Edition*, May 1992.

[RBR+91] U. Ramacher, J. Beichter, W. Raab, J. Anlauf, N. Bruls, M. Hachmann, and M. Wesseling. Design of a 1st Generation Neurocomputer. In *VLSI Design of Neural Networks*. Kluwer Academic, 1991.

[vECGS92] Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser. Active messages: a mechanism for integrated communication and computation. In *Proc. Tenth International Symposium on Computer Architecture*, May 1992.