A Parallel Neuromorphic Text Recognition System and Its Implementation on a Heterogeneous High Performance Computing Cluster

Qinru Qiu, Qing Wu, Morgan Bishop, Robinson Pino, *Members, IEEE*, and Richard Linderman, *Fellow, IEEE*

Abstract— Given the recent progress in the evolution of high performance computing (HPC) technologies, the research in computational intelligence has entered a new era. In this paper, we present a HPC-based context-aware Intelligent Text Recognition System (ITRS) that serves as the physical layer of machine reading. A parallel computing architecture is adopted that incorporates the HPC technologies with advances in neuromorphic computing models. The algorithm learns from what has been read and, based on the obtained knowledge, it forms anticipations of the word and sentence level context. The information processing flow of the ITRS imitates the function of the neocortex system. It incorporates large number of simple pattern detection modules with advanced information association layer to achieve perception and recognition. Such architecture provides robust performance to images with large noise. The implemented ITRS software is able to process about 16 to 20 scanned pages per second on the 500 TFLOPS (trillion floating point operations per second) AFRL/RI Condor HPC after performance optimization.

Index Terms — Heterogeneous (hybrid) systems, distributed architecture, natural language interfaces, machine learning.

1 Introduction

With the rapid development in high performance computing (HPC) technologies, the research in machine intelligence has entered a new era. How to harness the huge amount of computing power and memory storage provided by the modern HPC clusters and convert it to useful computations that assist or even replace the human cognition process? Will the performance of current neuromorphic computing models scale as the hardware resource increases? What is the bottleneck of current HPC architectures when applied to cognitive computing and how can this be addressed by future computing tools? The research work at Syracuse University and the Air Force Research Laboratory (AFRL) Information Directorate (RI) makes a preliminary effort in answering these questions.

Research discoveries in human psychology suggest that human information processing is a multi-level process [1] that mostly relies on pattern matching and sensory association rather than calculation and logic inference. Information is first processed by the sensory cortex where the complex data is reduced to abstract representations. The abstract representation is compared to stored patterns in massively parallel neural networks in the basal ganglia and neocortex to generate a quick reaction. If more sophisticated processing such as reasoning is needed then relatively slower sequential process will occur in the prefrontal cortex. To cope with this information process procedure, the neocortex of human brain consists of the primary sensory area, the association area and the higher order association area [2]. The primary sensory cortex detects the basic dimensions of the external stimuli to the five sensory systems. The sensory cortex is further divided into cortical columns which could detect a specific input pattern (such as contour, color, or pitch, etc.) in a specific area. Each sensory system has its own association area that combines information from the primary sensory cortex to produce perception (i.e. cognition). The higher order sensory system carries out complex mental processes by combining information from several sensory association areas. Sensory association is the most important step in perception. The association area is by far the most developed part of the cerebral cortex.

The above analysis partly reveals the answers to the first question that we previously raised. In order to harness the modern computer to imitate the human cognition process, we believe that the following architecture should be considered:

- *Received and approved for public release by AFRL on 03/29/2011, case number 88ABW-2011-1860.*

- *Q. Qiu is with the Syracuse University, Syracuse, NY 13244. E-mail: qiqiu@syr.edu.*
- *Q. Wu is with the Air Force Research Laboratory, Rome, NY 13441. E-mail: Qing.Wu@rl.af.mil.*
- *M. Bishop is with the Air Force Research Laboratory, Rome, NY 13441. E-mail: Morgan.Bishop@rl.af.mil.*
- *R. Pino is with the Air Force Research Laboratory, Rome, NY 13441. E-mail: Robinson.Pino@rl.af.mil.*
- *R. Linderman is with the Air Force Research Laboratory, Rome, NY 13441. E-mail: Richard.Linderman@rl.af.mil.*

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

1. Both the hardware and software should follow the hierarchy of neocortex system, with the lower level dedicated for pattern detection of the raw external input and the upper level dedicated for information association based perception.
2. The same input should be processed by multiple function modules corresponding to different primary sensory cortical columns for the detection of different patterns. In this way the complexity of each function module is reduced. Furthermore, all of the function modules are independent to each other and can be implemented in parallel.
3. Advanced and fast information association is more important than accurate detection. With the help of information association, relatively simple pattern matching algorithm can be used to achieve accurate perception.

Many algorithms have been proposed for pattern detection and information association. Clearly, different algorithms favor different hardware configurations. In general, pattern matching algorithms such as neural networks and support vector machines (SVM) are dominated by matrix and vector operations while information association models such as Bayesian network and probabilistic graph model require large storage space to capture complex relations. If we try to replicate the human information processing flow, it naturally requires machines with massively parallel processing capability and high computation speed at the bottom layer for pattern detection and machines with large memory space and high memory access speed at the upper layer.

Such hierarchical architecture can be found in the 1,800-node 500 TFLOPS (trillion floating point operations per second) Condor HPC cluster that has been built at AFRL/RI in 2010. The Condor HPC consists of 78 subclusters and each sub-cluster is composed of dual Intel Xeon six-core processors as the head node, 22 Sony PlayStation3 (PS3) computers based on the IBM Cell Broadband Engine (Cell-BE) processor [12], and 2 NVIDIA general purpose graphic processing unit (GPGPU) cards. Each Cell-BE processor has one PowerPC processor and 6 synergistic processing elements (SPE). Each SPE is a self-contained vector processor that peaks at eight floating point operations per clock cycle at 3.2 GHz. With 6 of these SPEs, a cell processor provides 153 GFLOPS (billion floating point operations per second) peak performance. The vector processing capability of the SPE makes it suitable for matrix and vector operations used in pattern matching algorithms such as neural networks and SVMs. Overall, the 1,716 Cell-BE processors deliver 262 TFLOPs computing power and form the first layer hardware of a neuromorphic computing system. The second layer is naturally the head nodes, each of which has 12 cores and 24GB memory. The memory access speed is up to 2GB/s per core.

We believe that such brain inspired signal processing flow could generally be applied to many cognitive applications, from image processing, to intruder detection, etc. To investigate the software and hardware requirements of this new information processing approach, a proof-of-concept prototype of context-aware Intelligence Text Recognition [13][14][15] software (ITRS) is developed on the Condor HPC. Its architecture incorporates the Condor HPC technologies with advances in neuromorphic computing models. The lower layer of the ITRS performs pattern matching of the input image using a simple nonlinear autoassociative neural network model called Brain-State-in-a-Box (BSB) [6]. It matches the input image with the stored alphabet. Each BSB model is analogy to a cortical column in the primary sensory area that performs the preliminary detection. Sometimes, multiple matching patterns may be found for one input character image. The upper layer of the ITRS performs information association using the cogent confabulation model [11]. It enhances those BSB outputs that have strong correlations in the context of word and sentence and suppresses those BSB outputs that are weakly related. In this way, it selects those characters that form meaningful words and sentences. Each confabulation model is analogy to a cortical column in the sensory association area that associates the primary detections to form high level cognition. Compared to the existing optical character recognition (OCR) system such as OCRopus [3][4][5], Tesseract [15][16], and Microsoft OneNote, the proposed ITRS system has the following uniqueness.

1. It has a much simpler bottom layer for image processing and pattern matching. The BSB model is a simple and weak associative memory compared to some more powerful networks using complex learning rules [6]. However, we propose a novel racing mechanism that enables the BSB to generate fuzzy pattern matching result which retain rich information that could be processed by the upper association layer. By contrast, most of the existing text recognition systems heavily rely on image processing and pattern matching, which require complex algorithms and intensive computation. They provide deterministic result and cannot be integrated with an information association layer.
2. The text recognition of ITRS is mainly achieved by a powerful information association layer. The more than 6GB knowledgebase of the information association layer contains information extracted from English literature. It is trained by "reading' more than 70 classical texts. Our hash based technique enables us to update and query the knowledgebase efficiently [17]. Although many of the existing text recognition systems also have integrated dictionary and language models, compared to the one used in the ITRS, they are rather preliminary.
3. The powerful information association technology and extensive knowledge in English language enables the ITRS to perform text recognition using information beyond the input image. The experimental results show that the ITRS system is capable of recognizing more than 85% of words correctly when each word has 30% characters occluded. Using the OCR function of the Microsoft OneNote, less than 5% of words can be read accurately.

4. The ITRS is intrinsically parallel and its software structure fits nicely to the architecture of Condor HPC. To recognize a sentence requires up to 10,400 BSB models, 20 confabulation models at word level and 1 confabulation models at sentence level. With medium optimization effort in double buffering and latency hiding, these models can be operated in parallel. The Cell-BE in the cluster is efficient in processing the BSB models due to its vector processing capability and abundance. The headnode in the cluster is the natural candidate for the processing of the confabulation model, due to its large storage space and fast memory access speed. Compared to the ITRS, the existing text recognition system does not have good scalability to distributed and parallel computing hardware. Because they do not have an intrinsic parallel structure, they usually focus on the parallelization of a specific pattern matching [18] or image processing [19] algorithm, which requires high design effort.

To implement the ITRS system on the Condor cluster is not trivial. The major challenge is how to balance the workload and hide the communication latency for better performance. This paper will introduce the software and hardware architecture of the ITRS system as well as some implementation details. Its accuracy and performance will be discussed based on the experiment data. The impact of the available hardware resource on the system throughput will also be analyzed.

The remainder of the paper is organized as follows. A brief introduction of related works in text recognition is provided in Section 2. In Section 3 we introduce the basics of the two neuromorphic models used for the ITRS software. Section 4 describes the overall system model and the algorithms in different layers. Section 5 gives the details of implementation on the Condor HPC. The experimental results and discussions are presented in Section 6. Section 7 summarizes the work.

2 Related works

The research in text recognition has a long history [20]. It consists of three major thrusts, optical character recognition (OCR), pre-OCR image processing and post OCR words correction.

Before 2000, the research efforts focus on general OCR technology. Most of the works divides an OCR engine into five stages: scanning, segmentation, feature extraction, and character classification [22]. Some typical used feature extraction techniques for OCR include, template matching [23], zoning [24], moments extraction [25][26], contour information, etc. A detailed survey of feature extraction techniques for OCR is provided in [21].

Based on the classification techniques, the OCR can be divided into statistical identification [28], syntactic classification [28] and neural network based classification [22][29][30][31]. In [29], the authors recognize handwritten numbers based on the features extracted from the directional code histogram and gray scale transformation. A two stage neural network is developed to classify these features. The first layer consists of a 256 input single layer neural network that classifies the gray scale feature. The second layer consists of 45 64-input and 3-output neural networks that classify the directional features. The authors of [31] utilize a back propagation neural network to recognize the Japanese characters. The authors divide character image into small regions and extract blob information of these regions. These features are classified by a 4-layer neural network that has 2386 neurons and 185,580 connections.

In recent years, the research focus of text recognition shifts to pre-processing and post processing techniques. The former enhances the image quality for better OCR accuracy [32][33][34], while the later relies on dictionary of language information to correct OCR errors [35][37][36][38]. For example, both references [35] and [37] try to correct OCR error based on topic information. Their goal is to obtain a maximum likelihood estimate of the actual word t given the OCR output w by maximizing the posteriori $p(t|w) = p(w|t)p(t)/p(w)$. The word probability $p(t)$ is profiled for different topics, $p(w)$ (i.e. the probability of OCR output w) is assumed to be a uniform distribution. The error model $p(w|t)$ is assumed to be known by characterizing the OCR tools. This model assumes a constant $p(w\,|\,t)$ during the entire OCR process, which is not reasonable, because the signal noise ratio will change during OCR as the image quality changes. Furthermore, the assumption that w is uniformly distributed is also not realistic. The authors of [37] propose to verify the OCR output by sending a query to search engine such as Google for each recognized word, a correlation is built based on the number of results returned. This verification method requires no training and it is able to recognize popular made-up words. However, it does not consider any context information beyond word level. The authors of [38] use the n-gram model to capture character correlations at word level. A database is then developed for queries that search for the closest match. Again, this work only considers information at word level. None of the above mentioned work addresses the performance of OCR.

Our review shows that existing OCR technique usually requires complicated feature extraction and computation intensive pattern classification. It has a separate post-OCR correction stage which usually only relies on word level information. Our proposed ITRS overcomes these limitations by combining OCR and post-OCR correction, and utilizes context information at sentence level

3 Background

The neuromorphic model adopted by the ITRS software is mainly built based on the *Brain-State-in-a-Box* (BSB) attractor model [10] and the *Cogent Confabulation* model [11]. The BSB models provide the preprocessing of the image of each character seeking a matching pattern. The cogent confabulation algorithms combine information from the BSB model to form more complex objects such as words or sentences. During this procedure, it suppresses the inputs that does not have strong association with others and enhances the remaining inputs. In other words, the confabulation model eliminates those BSB results that do not form meaningful words and sentences.

3.1 Brain-State-in-a-Box model

The BSB model is a simple, auto-associative, nonlinear, energy-minimizing neural network [7][8][9][10]. A common application of the BSB model is to recognize a pattern from a given noisy version. It can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries.

There are two main operations in a BSB model, *Training* and *Recall*. In this work, we focus on the BSB recall operation. The mathematical model of a BSB ***recall operation*** can be represented in the following form:

$$\mathbf{x}(t+1) = S(\alpha * \mathbf{A} * \mathbf{x}(t) + \lambda * \mathbf{x}(t) + \gamma * \mathbf{x}(0)) \qquad (1)$$

where:

- $\mathbf{x}$ is an N dimensional real vector
- $\mathbf{A}$ is an N-by-N connection matrix
- $\mathbf{A}*\mathbf{x}(t)$ is a matrix-vector multiplication operation
- α is a scalar constant feedback factor
- λ is an inhibition decay constant
- γ is a nonzero constant if there is a need to maintain the input stimulation
- $S()$ is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1 & if \quad y \geq 1 \\ y & if \quad -1 < y < 1 \\ -1 & if \quad y \leq -1 \end{cases} \qquad (2)$$

Note that in the proposed algorithm, we choose α to be 0.1, λ to be 1.0 and γ to be 0.0. But they can be easily changed to other values during the implementation. Given an input pattern $\mathbf{x}(0)$, the recall process executes Equation (1) iteratively to reach convergence. A recall converges when all entries of $\mathbf{x}(t+1)$ are either "1.0" or "-1.0". In our implementation, it usually takes more than 10 iterations for recall to converge.

The BSB model is selected in the ITRS for two reasons. First, it is simple to operate compared to other complex neural network models [6]. Although it has lower accuracy, this can be compensated later in the information association stage. Second, its convergence roughly indicates the similarity between the input and the stored pattern. It is pointed out by [6] that the average convergence time of the BSB model increases as the input goes further away from the attractor. Such property enables the racing model in character recognition, which will be introduced in Section 3.2.

3.2 Cogent confabulation

Cogent confabulation is a connection-based cognitive computing model. It captures correlations between objects (or features) at the symbolic level and stores this information as a knowledge base. Given an observation, familiar information with high relevancy will be recalled from the knowledge base. Based on the theory, the cognitive information process consists of two steps: learning and recall. During learning, the knowledge links are established and strengthened as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-takes-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated and the winner neurons will activate other neurons through knowledge links. At the same time, those neurons that did not win in this procedure will be suppressed.

Fig. 1 shows an example of lexicons, symbols, and knowledge links. The three columns in Fig. 1 represent three lexicons for the concept of shape, object, and color with each box representing a neuron. Different combinations of neurons represent different symbols. For example, as shown in Fig. 1, the pink neurons in lexicon I represent the cylinder shape, the orange and yellow neurons in lexicon II represent a fire extinguisher and a cup, while the red neurons in lexicon III represent the red color. When a cylinder shaped object is perceived, the neurons that represent the concepts "fire extinguisher" and "cup" will be excited. However, if a cylinder shape and a red color are both perceived, the neurons associated with "fire extinguisher" receive more excitation and become activated while the neurons associated with the concept "cup" will be suppressed. At the same time, the neurons associated with "fire extinguisher" will further excite the neurons associated with its corresponding shape and color and eventually make those symbols stand out from other symbols in lexicons I and III.

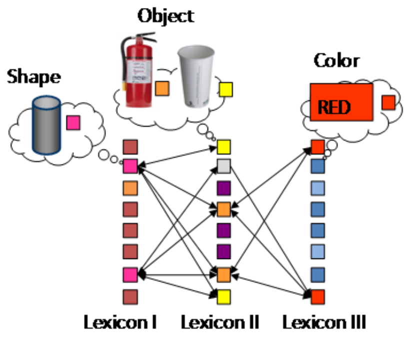


Fig. 1. A simple example of lexicons, symbols and knowledge links.

A computational model for cogent confabulation has been proposed by Hecht-Nielsen [11]. Based on this model, a *lexicon* is a collection of symbols. A *knowledge link* (*KL*) from lexicon I to II is a matrix with the row representing a source symbol in lexicon I and the column representing a target symbol in lexicon II. The (*i,j*)th entry of the matrix represents the strength of the synapse between the source symbol s_i and the target symbol t_j. It is quantified as the conditional probability $P(s_i \mid t_j)$. The collection of all knowledge links is called a *knowledge base* (*KB*). The knowledge bases are obtained during the learning procedure. During recall, the excitation level of all symbols in each lexicon is evaluated. Let l denote a lexicon, F_l denote the set of lexicons that have knowledge links going into lexicon l, and S_l denote the set of symbols that belong to lexicon l. The excitation level of a symbol t in lexicon l can be calculated as:

$$I(t) = \sum_{k \in F_l} \sum_{s \in S_k} I(s) \left[\ln\left(\frac{P(s|t)}{p_0}\right) + B \right], \quad t \in S_l. \qquad (3)$$

The function $I(s)$ is the excitation level of the source symbol s. Due to the "winner-takes-all" policy, the value of $I(s)$ is either "1" or "0". The parameter p_0 is the smallest meaningful value of $P(s_i \mid t_j)$. The parameter B is a positive global constant called the *band gap*. The purpose of introducing B in the function is to ensure that a symbol receiving N active knowledge links will always have a higher excitation level than a symbol receiving (N-1) active knowledge links, regardless of the strength of the knowledge links. For example, both symbols t_1 and t_2 belong to lexicon I. And both of them are activated by two symbols from other lexicons. However, t_1 is activated by two symbols that belong to the same lexicon, while t_2 is activated by two symbols that belong to different lexicons. We consider t_2 to be more highly activated than t_1 and the band gap B is introduced to increase the excitation level of t_2.

Compared to other information association models, such as the Bayesian network, the confabulation model is much simpler in training and recall due to its unique excitation mechanism and the adoption of posterior probability. For more information, please refer to [11].

4 Architecture and Algorithms

4.1 System architecture

The ITRS is divided into three layers as shown in Fig. 2. The input of the system is a text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with stored images of the English alphabet. If there is noise in the image, multiple matching patterns may be found. The ambiguity can be removed by considering the word level and sentence level context, which is achieved by the information association in the second and third layer where word and sentence is formed using cogent confabulation models. Image processing front-end software is designed to read in the scanned images of text and separate them into blocks of smaller images of single characters. The ITRS system is evaluated using images of scanned text with missing information, i.e., texts with hard-to-recognize or missing characters. Its accuracy will be reported in Section 6.

In this work, we designed a new "racing" algorithm for BSB recalls. The algorithm is based on the observations that the convergence speed of the BSB recall process indicates the distance between the input and remembered patterns. For a given character image, we consider all patterns that converge within a certain number of iterations as potential candidates that may match the input image. Candidate BSB outputs will be activated and used to trigger the corresponding symbols in the confabulation model for information association. By using the racing algorithm, if there is noise in the image or the image is partially damaged, multiple matching patterns will be triggered for the same input character image. For example, a horizontal scratch will make the letter "T" look like the letter "F". In this case we have ambiguity in character recognition. The pattern that cannot form meaningful words and sentences will be eliminated in the later stages.

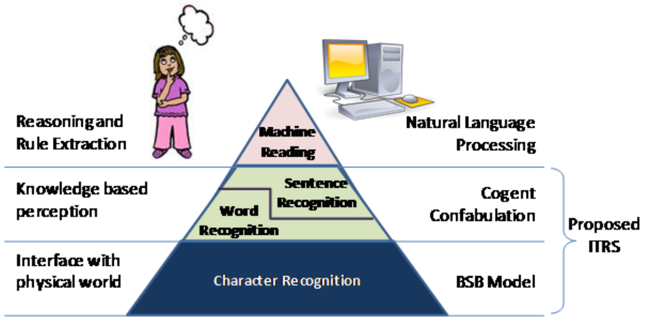


Fig. 2. Layered architecture of intelligent text recognition.

Fig. 3 shows an example of using the ITRS to read texts that have been occluded. The BSB algorithm recognizes text images with its best effort. The word level confabulation provides all possible words that associate with the recognized characters while the sentence level confabulation finds the combination among those words that gives the most meaningful sentence.

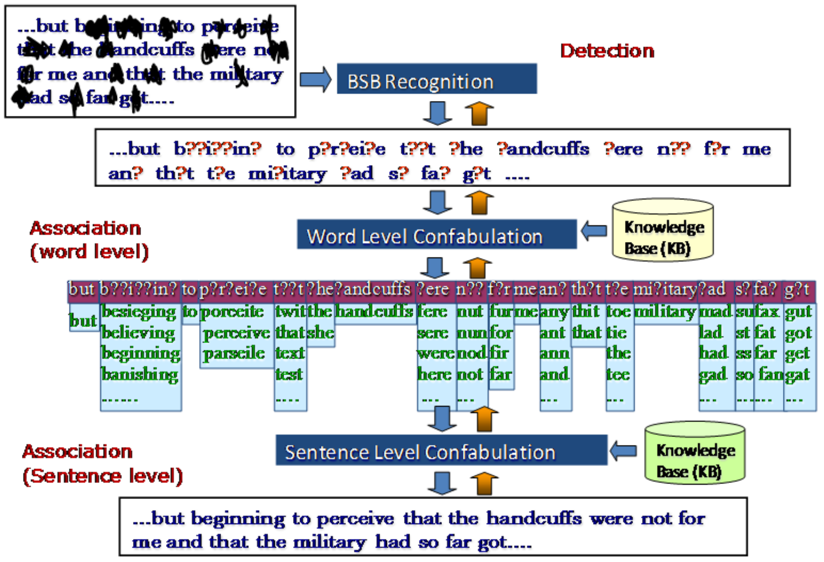

Detection
BSB Recognition
...but b??i??in? to p?r?ei?e t??t ?he ?andcuffs ?ere n?? f?r me an? th?t t?e mi?itary ?ad s? fa? g?t ....
Association (word level)
Word Level Confabulation
Knowledge Base (KB)
but b??i??in? to p?r?ei?e t??t ?he ?andcuffs ?ere n?? f?r me an? th?t t?e mi?itary ?ad s? fa? g?t
but besieging believing beginning banishing
to porceite perceive parseile
twit that text test
the she
handcuffs
fere sere were here
nut nun nod not
fur for fir far
me
any ant ann and
thit that
toe tie the tee
military
mad lad had gad
su st ss so
fax fat far fan
gut got get gat
Association (Sentence level)
Sentence Level Confabulation
Knowledge Base (KB)
...but beginning to perceive that the handcuffs were not for me and that the military had so far got....

Fig. 3. An example of occluded text recognition process.

4.2 The "racing" mechanism for BSB recalls

In this section we first describe the "racing" mechanism that we use to implement the multi-answer character recognition process.

Without loss of generality, assume that the set of characters we want to recognize from images consists of 52 characters, which are the upper and lower case characters of the English alphabet.

$$S = \{'a', 'b', \ldots, 'z', 'A', 'B', \ldots, 'Z'\}$$

We also assume that for each character in S, there are M typical variations in terms of different fonts, styles and sizes. For example, the set of images of character 'a' with different variations can be represented as:

$$S_a = \{a_1, a_2, \ldots, a_M\}$$

In terms of pattern recognition, there are a total of 52*M patterns to remember during training and to recognize during recall. If we follow the traditional application approaches of the BSB models, the solution is to train one BSB model to remember all the 52*M patterns. During recall, given an input image, this model will eventually converge to one of the remembered patterns (attractors) that represent the recognition result. The shortcomings of this approach is that, firstly it requires a BSB model with large dimensionality (N the dimension of vector X in Equation 1) to remember all the patterns. This increases the complexity ($\propto N^2$) of the computation and also reduces the scalability when implemented on parallel computing architectures. Secondly, this approach only provides one answer to the input image. The BSB recall process does not return the second or third closest attractor for the image. For recognizing damaged texts, providing only one answer is not adequate for the low-level pattern recognition model to work with high-level language models.

Therefore in our implementation, the primary goal is to design a process that provides multiple candidates for an input image. And the secondary goal is to have reasonably-sized BSB models to have good scalability and keep computation complexity under control.

The solution we designed is to use one BSB model for each character in S. Therefore there will be a set of 52 256-dimensional BSB models, that is:

$$S_{BSB} = \{BSB_a, BSB_b, \ldots, BSB_z, BSB_A, BSB_B, \ldots, BSB_Z\}$$

Each BSB model is trained for all variations of a character. For example, BSB_a is trained to remember all the variable patterns in S_a, BSB_b will remember patterns in S_b, and so forth. If we define the procedure "Recall(A, B)" as the recall process using model A with input image B, returning the number of iterations it takes to converge, the recall and candidate selection process can be described in Fig. 4.

In this algorithm, {K, Th_1, Th_2} are adjustable parameters based on overall reliability and robustness needs.

Generally speaking, in our multi-answer implementation, we utilize the BSB model's convergence speed to represent the "closeness" of an input image to the remembered characters (with variations). Then we pick up to K "closest" candidates (that satisfy conditions 3a and 3b) to work with high-level language models to determine the final output. On the AFRL/RI Condor HPC with 1,716 IBM Cell-BE processors, our implementation was able to execute the recall operations in parallel. Because each BSB model is small enough to fit on a single Cell-BE processor, the overall performance scales linearly with the number of Cell-BE processors used.

```
Input: character image X.
1. For each trained BSB model $BSB_i$ in $S_{BSB}$
        Conv[i] = Recall($BSB_i$, X);
2. Sort Conv[i] from low to high to form sorted list Conv_s[j];
3. Pick the first $K$ in Conv_s[j] as recognition candidates, if it
satisfies both conditions listed below:
   a. Conv_s[j] <= Th_1; // Convergence speed threshold
   b. Conv_s[j] – Conv_s[j-1] <= Th_2; // Separation threshold
```

Fig. 4. The BSB recall and candidate selection process.

4.3 Word and sentence confabulations

The inputs of word confabulation are characters with ambiguities referred as candidates. For each input image, one or multiple character level candidates will be generated by the BSB model. In this work, we assume that each word has less than 20 characters. Any word that is longer than this will be truncated. Currently, if a word has less than 20 characters, it will be padded with white spaces. In the future, the length of the word will be considered as an input to the confabulation model to avoid this type of padding and to speed up the process.

The word level confabulation model consists of three levels of lexicon units (LUs). There are 20 LUs in the first level and the ith LU in the first level represents the ith character in the word. There are 19 LUs in the second level and the ith LU in the second level represents a pair of adjacent characters at location i and i+1. Finally, there are 18 LUs in the third level and the ith LU in the third level represents a pair of characters located at i and i+2.

A *knowledge link* (*KL*) from lexicon I to II is an $M \times N$ matrix, where M and N are the cardinalities of symbol sets S_I and S_{II}. The (i,j)th entry of the knowledge link gives the conditional probability $P(i|j)$, where $i \in S_I$, and $j \in S_{II}$. Symbols i and j are referred to as the *source symbol* and *target symbol*. Between any two LUs, there is a knowledge link (*KL*). If we represent the lexicons as vertices and represent the knowledge link from lexicon I to lexicon II as a directed edge from vertex I to vertex II, then we will obtain a complete graph.

Confabulation-based word level and sentence level prediction heavily relies on the quality of the *knowledge base* (*KB*). The training of the KB is the procedure to construct the probability matrix between source symbols and target symbols. The training program first scans through the training corpus and counts the number of co-occurrences of symbols in different lexicons. Then for each symbol pair it calculates their posterior probability.
The word level recall algorithm finds all words from possible combinations of input character candidates. For example, if the input candidates of a 3-letter word are: (w t s r p o k e c a) for the first letter, (h) for the second letter,

and (y t s r o m i h e a) for the third letter, then the word level confabulation program will find 24 words, including "why", "who", "wha", "thy", "thi", "the", "tha", "shy", "sho", "she", "rho", "phr", "ohs", "oho", "ohm", "kho", "eht", "cha", "aht", "ahs", "ahr", "ahm", "ahh", and "aha". Although some of them are not dictionary words, they appear at least once in the training corpus, which consists of more than 70 fiction books. Some of these non-dictionary words are names for special places or objects and some of them are used to represent specific sounds. A few of them are typos or errors in the training file.

For each input candidate in each lexicon, the recall algorithm sets the corresponding symbols to be active. A lexicon that has multiple symbols activated is referred to as an *ambiguious lexicon* and the goal of the word level confabulation is to eliminate such character level ambiguity as much as possible or to transform it into word level ambiguity which can be further eliminated by sentence level confabulation.

For each lexicon that has multiple symbols activated, we calculate the *excitation level* of each activated symbol. The excitation level of a symbol i in lexicon B is defined as:

$$EL_B[i] = \sum_{A\neq B} \sum_{j\in\{active\ symbols\ in\ A\}} kl_{AB}[j][i],$$

where $kl_{AB}[j][i]$ is the knowledge value from symbol j in lexicon A to symbol i in lexicon B. The N highest excited symbols in this lexicon are kept active. These symbols will further excite the symbols in other ambiguous lexicons. This procedure will continue until the activated symbols in all lexicons do not change anymore. If convergence cannot be reached after a given number of iterations, then we will force the procedure to stop.

For each word in a test sentence, the word level confabulation model generates one or multiple word candidates. They will be the input to the sentence level confabulation model.

The sentence level confabulation model is very similar to its word level counterpart except that there are only two levels of LUs. The first level LUs represent single words while the second level LUs represent adjacent word pairs. The training and recall functions of sentence level confabulation have the same principle as these functions at word level. It is important to point out that, each first level lexicon for word confabulation contains at most 26 symbols representing 26 letters in the alphabet and each second and third level lexicon for word confabulation contains at most $26 \times 26 = 676$ symbols; however, the maximum size of the symbols of each sentence level lexicon equals the total number of English words and word pairs. Without any compression, the sentence level knowledge base will be extremely large. For example, the English version of the book "Round the Moon" has about 47×10^3 words. Our analysis shows that it has 23×10^3 distinguished symbols (i.e. words and word pairs). As we mentioned earlier, each knowledge link is a $M \times N$ matrix, where M and N are the symbol size of the source and target lexicons. Without any compression, the trained knowledge base will have 2.3×10^9 entries which are equivalent to be 9.2 GBytes. Fortunately, the knowledge links are sparse matrices. Only less than 0.1% of the matrix has non-zero values. Therefore, an option to reduce the memory cost is to store the knowledge using the *list of list* (LIL) format, which has been widely used for sparse matrix storage. However, this leads to the second problem. As the size of the training corpus grows, the number of symbols of each lexicon can easily go up to hundreds of thousands. Even with the best search algorithm, the time to locate the entry in the compressed matrix grows logarithmically and soon the algorithm will become prohibitively slow. In this work, two-level hash functions are used to speed up the training and recall of the sentence level confabulation model. It provides 10 to 15X speed ups to locate a knowledge entry in a 6 GB compressed knowledge base. More details of sentence level confabulation can be found in our recent work [17].

We need to point out that, many state-of-the-art optical character recognition (OCR) systems also have integrated language model [3][16]. The most widely used existing language model is the n-gram model which captures the posterior probability $p(w_i | w)$ of the next n words w_i, $1 \leq i \leq n$, given the observation of the current word w. The confabulation based language model differs from the n-gram model in several ways. First, the confabulation model selects the ambiguous word in a way such that the likelihood of the observation of the rest of the sentence is maximized. In contrast, the n-gram model maximizes the likelihood of the selected words given the observation of the rest of the sentence. Although there is no definite advantage of one approach over another, they sometimes give different results.

Second, the n-gram model can only be applied in a sequential way. It analyzes the sentence and predicts words in a sequential order from left to right, while our knowledge base provides the relations between all words or word pairs in the sentence. This enables the software to confabulate ambiguous words at the beginning of the sentence based on the information provided later.

Finally, the recall function of the confabulation model mimics the information processing in the human neurology system, where neurons are exciting and being excited at the same time. Therefore, when ambiguity exists in multiple words, the selections of these entries evolve simultaneously. These differences indicate that the confabulation model is more suitable for information association than the n-gram model.

5 Implementation on AFRL/RI Condor HPC

5.1 Overall software architecture

The overview of the implementation of the ITRS software is shown in Fig. 5. It explores the parallelism in hardware and software to achieve a high throughput for the system. We partition the entire workload into pages. All sub-clusters run simultaneously and independently to process different pages. In this way the cluster level parallelism is achieved. There is a performance monitor that periodically checks the utilization of the processor cores in the cluster for performance characterization. Because each sub-cluster loads pages on-demand, at cluster level,

our system behaves asynchronously.
Upon receiving the page image, the head node first slices the image into small blocks, each of which contains one character. The blocks are dispatched to the PS3s, on which the BSB recalls are run for character recognition. The results are sent back to the head node for word level and sentence level confabulation. With a double buffering technique, the confabulation and BSB processes can be made parallel. Furthermore, all 132 SPEs in 22 PS3s are running simultaneously to process different characters. In this way we achieve processor level parallelism. At this level, the system is loosely synchronous because each SPE receives the same amount of image blocks and they perform the same amount of computation. Because of the limited buffer space, a periodic synchronization between the BSB and the confabulation is necessary. All inter-processor communication is implemented via the Message Passing Interface (MPI).

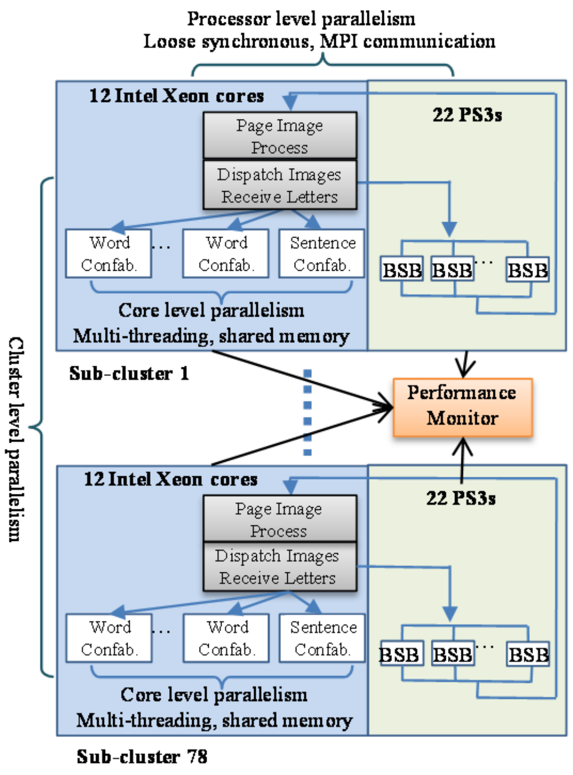


Fig. 5. Overview of the software architecture.

Based on the results from the BSB recalls, the host will fork multiple threads; each thread is a word level confabulation procedure. After all words in a sentence have been found, a sentence confabulation process is executed. The word level and sentence level confabulation threads are dispatched to different cores on the Intel Xeon processor, and in this way we achieve core level parallelism. The key reason that we choose thread level parallelism instead of process level parallelism is because it allows shared memory so that we do not have to duplicate the word-level knowledge base, which is more than 200MB in size. In order to avoid frequent context switching, which usually happens when the number of threads is greater than the number of cores, we adopt a token passing mechanism to control the number of threads. The program maintains a token pool. The number of tokens in the pool is less than or equal to the number of cores in the system. A token will be removed from pool when a thread is created and be returned when the thread ends. Because the threads are created on demand and complete dynamically, at this level, all cores work asynchronously.

5.2 Sub-cluster level task interactions

As we mentioned before, at the sub-cluster level, the system is loosely synchronous. At the beginning of each iteration, the head node processes the scanned page and sends 96 character images to each PS3. Without waiting for the PS3 to send back the BSB results, the head node continues to work on the BSB results received in the previous iteration. During the same time, all PS3s perform similar computations and they will complete the recalls at approximately the same time. The candidates for matching patterns are returned to the head node and stored in the MPI buffer. The head node will not process the MPI message until it has finished processing the previously returned results.

At the sub-cluster level, two techniques are used to increase the throughput of the system. Firstly we pipeline the BSB model and confabulation model on PS3s and the head node. Therefore the throughput of the system is determined by the maximum delay of BSB and confabulation instead of the total delay of these two. Secondly, by using the MPI for inter-processor communication, we implicitly use a double buffering technique to hide the communication latency.

Fig. 6 shows the sub-cluster level task scheduling. When the confabulation delay is much greater than the BSB delay, the communication latency for the send and receive procedure as well as the computation latency of the BSB recalls are hidden. The initiation interval of the system is determined by the delay of image processing and confabulation, i.e. $T = T_{img} + T_{confab}$. When the confabulation delay is smaller than the BSB delay, the computation latency of the confabulation model is hidden, the system initiation interval is determined by the delay of image processing, the communication delay of sending and receiving MPI messages and the delay of the BSB recalls, i.e. $T = T_{img} + T_{send} + T_{bsb} + T_{rcv}$. It is important to point out that T_{bsb} is a constant. For each input image, the same number of BSB recalls is performed. Each BSB recall is run for the fixed number of iterations in order to check their convergence speed. The quality of the input image does not affect the BSB computation time. However, a lower image quality means that more candidates will be found by the character recognition process. Therefore it increases the workload and execution time for word and sentence confabulation processes.

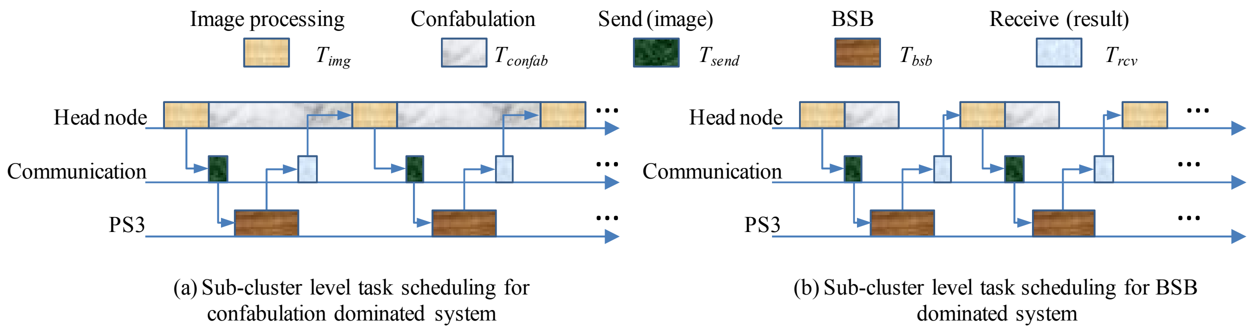


Image processing
T_{img}
Confabulation
T_{confab}
Send (image)
T_{send}
BSB
T_{bsb}
Receive (result)
T_{rcv}
Head node
Communication
PS3
(a) Sub-cluster level task scheduling for confabulation dominated system
(b) Sub-cluster level task scheduling for BSB dominated system

Fig. 6. Sub-cluster level task scheduling.

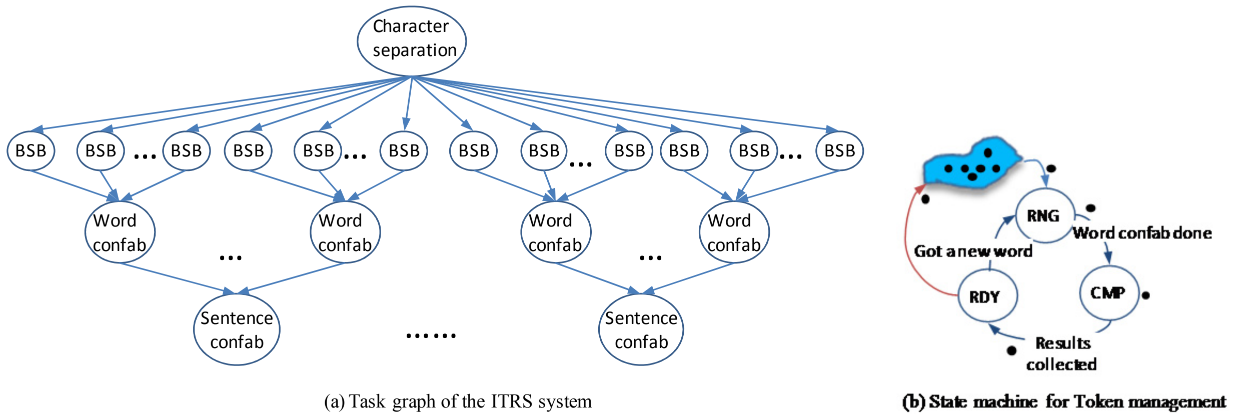


Fig. 7. Task graph of the ITRS and token management for multi-threading system.

5.3 Multi-threading confabulation

To fully utilize the multi-core architecture of the head node, layers 2 and 3 of the ITRS are implemented using multi-threading techniques. Fig. 7 shows the task and data dependency graph of the ITRS software. A word confabulation process will not be triggered until all the characters in that word have been processed by the BSB processes. Similarly, a sentence confabulation process cannot start until all the words in that sentence have been confabulated. Obviously the word confabulation process is triggered more frequently than the sentence confabulation process. Furthermore, each word confabulation takes a longer time than a sentence confabulation process. This is because we need to find all valid words from the combinations of the character candidates while only the most meaningful sentence from the combination of the word candidates.

In order to maximize the throughput, it is necessary to parallelize the word confabulation processes. Fig. 8 shows how the ITRS tasks are mapped to a sub-cluster with 22 Cell-BE processors and one N-core head node. At anytime, on the N-core head node we can run one thread of sentence confabulation and N-1 threads of word confabulation. Each word confabulation thread processes one word. The sentence confabulation thread is the main thread which is always active. Besides sentence confabulation, it also performs other tasks such as character seperation and communicates with the BSB processes. The word confabulation thread is dynamically created when all characters belonging to a word have been processed by the BSB processes. After the word confabulation completes, the thread will be deleted.

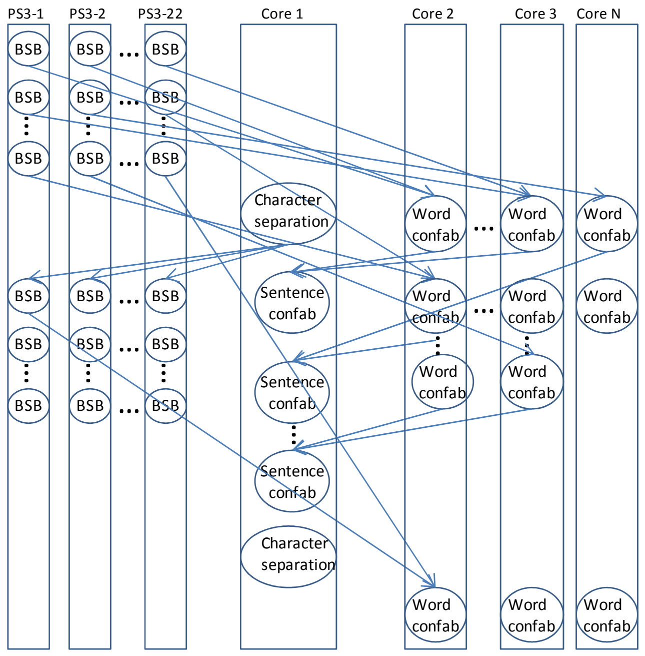


Fig. 8. Mapping of ITRS tasks to a sub-cluster.

We keep the number of threads to be exactly equal to the number of cores in the head node, in order to avoid excessive context switches. This is achieved by using a token passing mechanism. A token is used to represent the status of a core. It can be in 3 states: ready state, running state and completion state. A token pool is maintained in the main thread. The number of tokens in the pool equals to the number of cores in the system. All tokens in the pool are in the ready state. When all characters of a word are received and if there is a token in the pool, a new word confabulation thread is created and a token is removed from the pool. The state of that token is changed to "running". When the word confabulation completes, the thread is deleted and the token state is changed to "completion." Only after the results of the word confabulation are collected by the main thread, will the state of the corresponding token be changed to "ready" and the token will return to the pool. The token passing mechanism guarantees that at any time the number of active threads is no more than the number of processor cores.

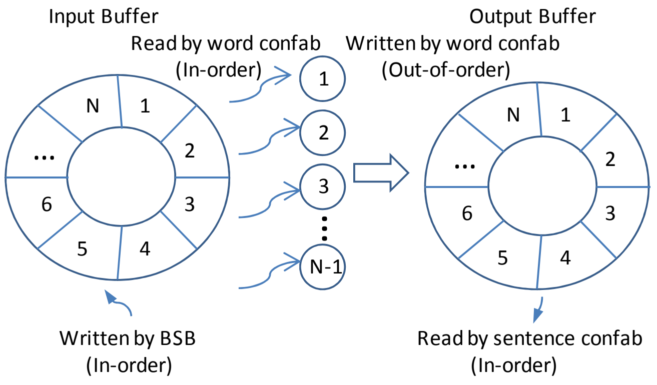


Fig. 9. In-order/out-of-order double buffering system.

The multi-threading architecture leads to an interesting synchronization problem. As shown in Fig. 9, two circular buffers are maintained in the main thread. The first buffer is referred as the "input buffer". It stores the outputs from the BSB processes that will be used as the inputs by the word confabulation thread. The input buffer is written in a sequential order. It is read by the word confabulation also in a sequential order. The results from word confabulation will be written into an "output buffer". The item in the *i*th location of the input buffer will be written into the *i*th location of the output buffer. There are up to *N*-1 threads of word confabulation working simultaneously on *N*-1 different words. Their processing speeds are different. Whenever a thread completes processing a word, it writes the result to the corresponding location in the output buffer and fetches another word from the input buffer. As a result, the output buffer will be written out-of-order. The output buffer will be read by the sentence confabulation process again in sequential order. A read pointer is used to indicate the starting word of the next sentence. When the next *M* entries from the read pointer have been filled (*M* is the number of the words in the next sentence), the sentence confabulation process will be started and those entries will be removed from the output buffer.

In general, a new thread of word confabulation will start as long as the input buffer is not empty and a token is available. However, due to the variable confabulation speed of different words, it is possible that one of the threads is still working on a word that belongs to the sentence that is to be read out next, while other threads have already filled up the rest of the buffer. Because the output buffer must be read out in sequential order, no sentence can be read from the buffer before the current sentence is read. In this scenario, the output buffer is "full" and a stall happens. No new word will be fetched from the input buffer until the next sentence is removed from the output buffer. More strictly speaking, the stall happens when the following three conditions are true:

- The read pointer of the output buffer is at the *i*th location,
- There is one word confabulation thread working on the *j*th location, $j - i < M$, where M is the number of words in the next sentence,
- The current read pointer of the input buffer is at location *j*-1.

The stall is used to synchronize the speed of different word confabulation processes; therefore we refer to the delay that is introduced by the stall as synchronization delay. The synchronization delay increases when the variation of the word confabulation time gets larger.

6. Experimental Results

The ITRS software is implemented on the AFRL/RI Condor HPC and evaluated for accuracy and performance.

6.1 Recognition accuracy

In the first experiment, we test the software using text images with low level noise. Our test case is extracted from the book "Great Expectations" by Charles Dickens. The text consists of 96767 letters or 23912 words. The text has not been read during the training process. In order to explicitly control the noise in the input, we use generated bit maps of text images instead of scanned text images. Horizontal scratches are added to the images of letters selected randomly. The amount of noise in the input is controlled by two parameters: (1) the thickness of horizontal scratches varies from one pixel wide to three pixels wide. Fig. 10 shows examples of the three different types of horizontal scratches. Note that the scratches are located in the center of the text image, where most of the information to distinguish amongst various characters is found. Also note that each text image is 15x15 pixels, and a 1~3 pixel scratch across the image is equivalent to 7~20% missing information. (2) The probability that a character is scratched varies from 0.2, 0.4 to 0.6.

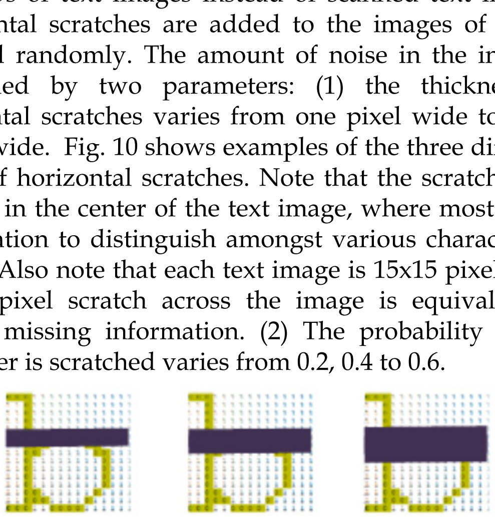


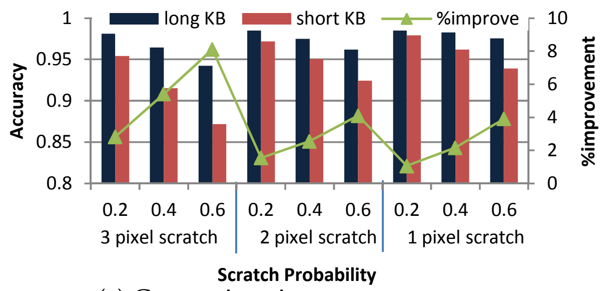
Fig. 10. Three different horizontal scratches.

The outputs of ITRS are compared against the original text. A sentence (or a word) is considered inaccurately recognized if there is one word (or one letter) mismatch from the original text. TABLE 1 gives the accuracy of word and sentence confabulation. They are calculated as the number of sentences (words) that have been correctly "read" divided by the number of sentence (word) confabulations that have been invoked. The same table also gives the percentage of correct words. It is calculated as the total number of correct words (including both confabulated and none-confabulated) divided by the total number of words in the text. As we can see, the rate of accurate word confabulation and the overall percentage of correct words are very close to each other. This is because the majority of words have at least one scratch. Therefore, they all need to go through the word confabulation process.

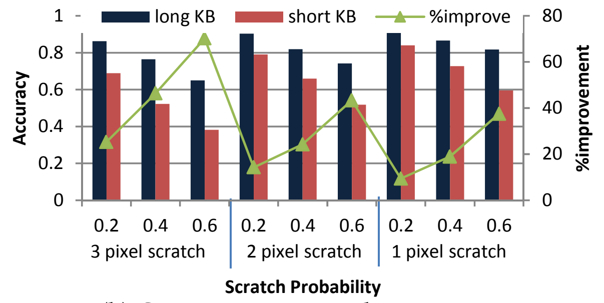
TABLE 1

RECOGNITION ACCURACY AT SENTENCE AND WORD LEVEL

Scratch prob.	Sentence confabulation accuracy			Word confabulation accuracy			Overall % correct words		
	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch
0.2	0.92	0.90	0.86	0.98	0.98	0.97	0.99	0.99	0.98
0.4	0.87	0.82	0.76	0.98	0.97	0.95	0.98	0.98	0.96
0.6	0.82	0.74	0. 65	0.97	0.95	0.93	0.98	0.97	0.94



(a) Comparison in sentence accuracy



(b) Comparison in word accuracy

Fig. 11. Impact of sentence knowledge base on the confabulation accuracy.

Fig. 11 shows the rate of correct sentences and correct words found by the ITRS when a well trained sentence level knowledge base (i.e. "long KB") is used and a poorly trained sentence level knowledge base (i.e. "short KB") is used. The size of the high quality knowledge base ("long KB") is more than 6 GB, while the size of the low quality knowledge base ("short KB") is only 2.7 MB. The data series of "% improve" gives the percentage improvement of the results obtained using "long KB" over the results obtained using "short KB". These charts indicate that better knowledge at the sentence level improves the sentence accuracy up to 80% and word accuracy up to 8%.

In the second experiment, we test the ITRS system using text images with completely occluded characters. The two test files are extractions from the book "Great Expectations" by Charles Dickens and the book "Lost World" by Arthur Conan Doyle. Neither of these books has been read during the training. We increase the percentage of occluded letters from 10% to 30%. TABLE 2 gives the sentence level and word level accuracy for different input files. As we can see, even with 30% of the characters missing, the ITRS can recognize more than 85% words correctly.

TABLE 2

ACCURACY WITH OCCLUDED CHARACTERS

Percentage of occluded letters	10%		20%		30%	
	Word accu.	Sentence accu.	Word accu.	Sentence accu.	Word accu.	Sentence accu.
Great Exp.	95.5%	60.2%	90.5%	33.6%	86.3%	23.9%
Lost World	97.0%	63.8%	92.7%	31.2%	86.4%	20.7%

The following two examples show the input text (with occluded letters) and the recognized text from ITRS:

Input: bu█ █ █new the s█unds by this t█me and co█ld █is█ociate th█m from the object o█ ██rs█it
Recognized sentence: but I knew the sounds by this time and could dissociate them from the object of pursuit

Input: gra█ious █o█dne█s █r█c█o█s me what█ █one wit█ th█ pi█
Recognized sentence: gracious goodness gracious me whats gone with the pig
Correct sentence: gracious goodness gracious me whats gone with the pie

Compared to existing text recognition system, the uniqueness of the proposed ITRS is its ability to achieve high accuracy from poor image detection and pattern matching results. For example, given a text image with about 20% characters occluded, the OCR function in Microsoft OneNote system outputs less than 20% correct words, which is much lower than the results from the ITRS.

6.2 System performance

Fig. 12 shows the evolution of the ITRS software architecture over time. We started with a base line implementation as shown in Fig. 12 (a), in which all the software components are connected sequentially except for the BSB engines that are running on 22 PS3s in parallel. Our first step is to improve the confabulation speed by multi-threading, as shown in Fig. 12 (b).

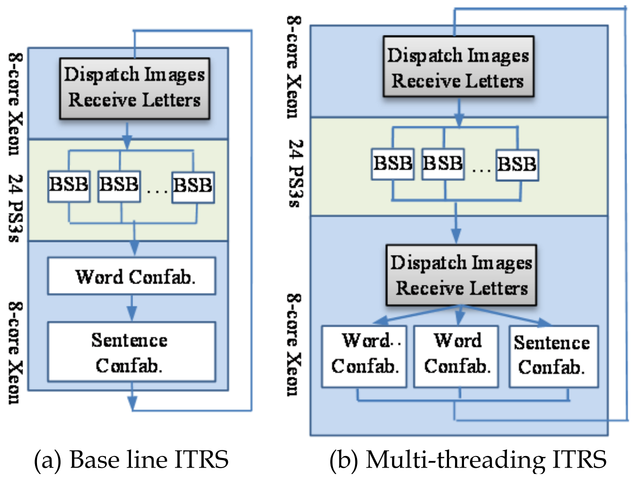
To evaluate the performance of the ITRS software, we carried out experiments on three different input test cases. In the first input file 20% of character images are scratched by 1-pixel-wide horizontal bars. Compared to the other two test cases, it has the highest image quality. The second input file has 40% of character images scratched by 2-pixel-wide horizontal bars. Compared to test cases one and three, it has the medium image quality. The last input file has 60% of character images scratched by 3-pixel-wide horizontal bars. It is the lowest quality input file. The number of word confabulation threads is varied from one to seven and denoted as *t*. The total runtime is broken down into BSB time, word confabulation time, sentence confabulation time and synchronization time. The concept of synchronization delay is introduced in Section 5.3. The sizes of the input/output buffers in the double buffering system are set to be 100 sentences.

Fig. 13 shows the runtime information for the three test cases when the number of word confabulation threads increases from one to seven. It also reports the performance improvements of the multi-threading implementations compared to the baseline implementation.

Several observations can be made from the results:

1. No matter how the image quality changes, the BSB time remains constant.
2. When the quality of the input text image deteriorates, the word/sentence confabulation time increases. This is because we rely on the confabulation to resolve the ambiguities in the input.
3. When the quality of the input text image deteriorates, the synchronization delay gets longer. This is because the variations in the word confabulation speed increases as the level of ambiguity rises, and the in-order/out-of-order circular buffer will be blocked more frequently.

With the multi-threading technique, we can improve the runtime by up to 70%.



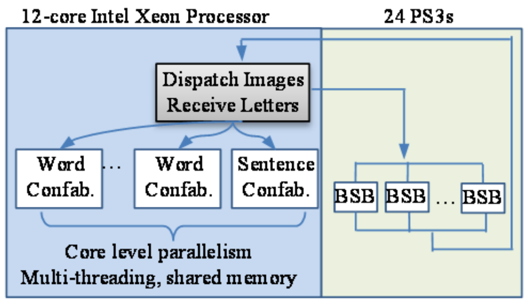


Fig. 12. Evolution of the ITRS software architecture.

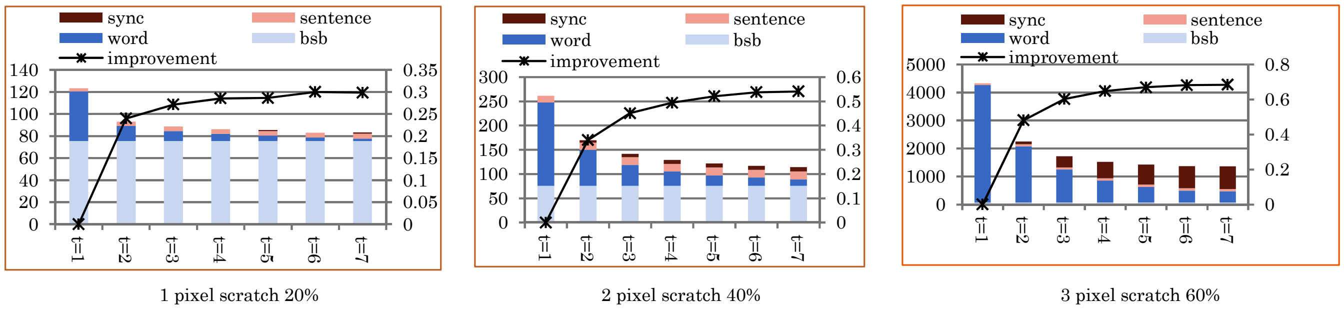


Fig. 13. Performance improvement by multi-threading confabulation.

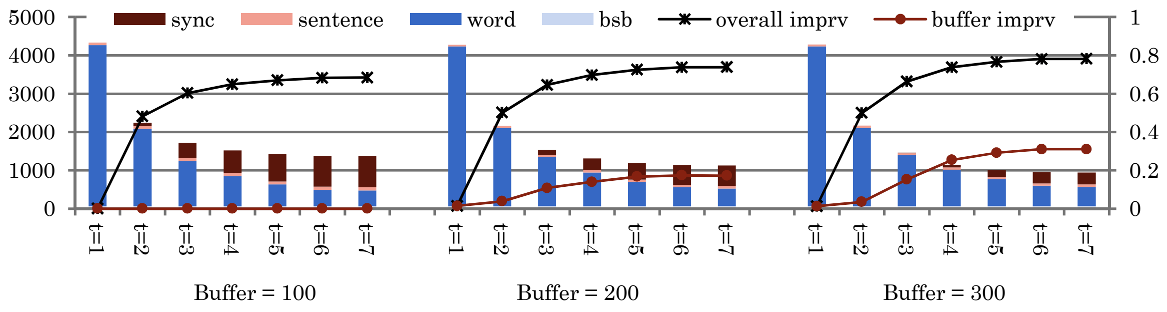
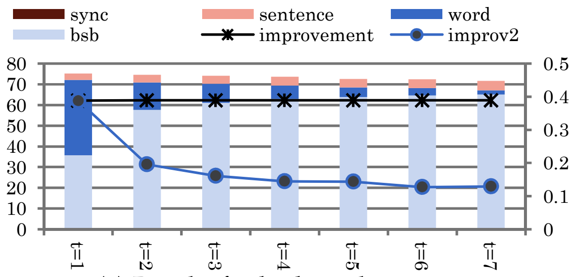
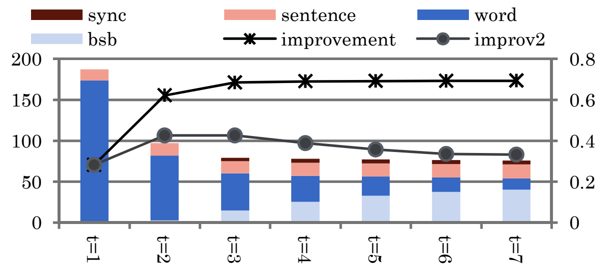


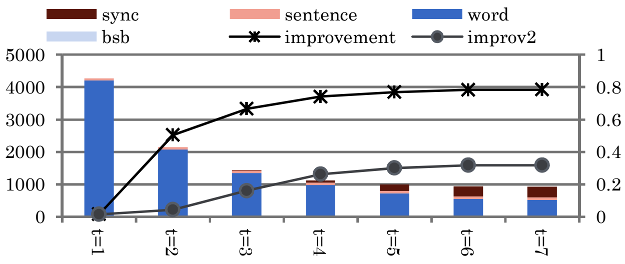
Fig. 14. Increase of buffer size reduces the synchronization delay.



(a) Results for high quality test case



(b) Results for medium quality test case



(c) Results for low quality test case

Fig. 15. Performance improvement by parallelizing BSB and confabulation.

The results in Fig. 13 show that with low quality input, the synchronization delay becomes the bottleneck that prevents us achieving linear speedups by using multi-threading techniques. One way to relieve this bottleneck is to increase the capacity of the double buffering system. We increase the buffer size from 100 sentences to 200 and 300 sentences and run the experiment again on the low quality input file. Fig. 14 gives the runtime information for the systems with three different buffer configurations. The last data series (i.e. "buffer imprv") gives the performance improvement due to the increased buffer size. The results show that with seven word confabulation threads, increasing the buffer size from 100 to 200 and 300, we reduce the runtime by 20% and 30%.

We further improve the ITRS software architecture by parallelizing the BSB and confabulation processes, as shown in Fig. 12(c). Fig. 15 shows the performance of the improved system on high quality, medium quality and low quality inputs. The buffer capacity is set to 300 sentences. The data series labeled "improvement" gives the performance improvement of the system over the base line implementation, while the data series labeled "improv2" gives the percentage speed improvement by comparing the parallel ITRS with multi-threading ITRS. The number of word confabulation threads and the buffer size of these two systems are kept the same. The results show that parallelizing the BSB and confabulation is most effective for the medium quality test cases, because the BSB time and confabulation time are approximately equal for this type of test cases and executing them simultaneously can reduce the total runtime by 50%.

7 Conclusion

We have presented a HPC-based context-aware Intelligent Text Recognition System (ITRS) that serves as the physical layer of machine reading. A parallel computing architecture is adopted that incorporates the HPC technologies with advances in neuromorphic computing models. The algorithm learns from what has been read and, based on the obtained knowledge, it forms anticipations at the word and sentence level. The knowledge helps to suppress the noise in during pattern detection. The implemented ITRS software is able to process about 16 to 20 scanned pages per second on the 500 TFLOPS AFRL/RI Condor HPC cluster with reasonable effort put toward performance optimization.

Acknowledgment and Disclaimer

Received and approved for public release by AFRL on 03/29/2011, case number 88ABW-2011-1860.

The contractor's work is supported by the Air Force Research Laboratory, under contract FA8750-09-2-0155.

Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

References

[1] R. Wray, C. Lebiere, P. Weinstein, K. Jha, J. Springer, T. Belding, B. Best, and V. Parunak, "Towards a Complete, Multi-level Cognitive Architecture," *Proc. of the International Conference for Cognitive Modeling*, 2007.
[2] R. S. Swenson, "Review of clinical and functional neuroscience," *Educational Review Manual in Neurology*, Castle Connolly Graduate Medical Publishing, 2006.
[3] "The OCRopus Open Source Document Analysis and OCR System," http://code.google.com/p/ocropus/.
[4] T. M. Breuel, "The OCRopus open source OCR system," *Proc. Of SPIE Document Recognition and Retrieval*, Jan. 2008.
[5] F. Shafait, "Document Image Analysis with OCRopus," *Proc. Of IEEE International Multitopic Conference*, 2009.
[6] J. A. Anderson, "An Introduction to Neural Networks," *The MIT Press*, 1995.
[7] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, probability learning: Some applications of a neural model," *The MIT Press*, 1989.
[8] M. H. Hassoun, *Associative Neural Memories: Theory and Implementation*, Oxford University Press, 1993.

[9] A. Schultz, "Collective recall via the Brain-State-in-a-Box network," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 580–587, July 1993.

[10] Q. Wu, P. Mukre, R. Linderman, T. Renz, D. Burns, M. Moore and Q. Qiu, "Performance Optimization for Pattern Recognition using Associative Neural Memory," *Proc. Of 2008 IEEE International Conference on Multimedia & Expo*, June 2008.

[11] R. Hecht-Nielsen, *Confabulation Theory: The Mechanism of Thought*, Springer, August 2007.

[12] IBM, "IBM Cell Broadband Engine resource center," http://www-128.ibm.com/developerworks/power/cell/.

[13] J. Mantas, "An Overview of Character Recognition Methodologies," *Pattern Recognition*, 19(6); 425-430, 1986.

[14] G. Nagy, "Optical Character Recognition – Theory and Practice," *Handbook of Statistics*, Vol. 2, 621-649, 1982.

[15] Google, "Tesseract-OCR", http://code.google.com/p/tesseract-ocr/

[16] R. Smith, "An Overview of the Tesseract OCR Engine," *Proc. Of International Conference on Document Analysis and Recognition*, 2007.

[17] Q. Qiu, Q. Wu, D. Burns, M. Moore, M. Bishop, R. Pino, R. Linderman, "Confabulation Based Sentence Completion for Machine Reading," *Proc. Of IEEE Symposium Series on Computational Intelligence*, April 2011.

[18] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and Scalability of GPU-based Convolutional Neural Networks," *Proc. Of Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2010.

[19] L. Lam, C. Y. Suen, "An Evaluation of Parallel Thinning Algorithms for Character Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, Issue. 9, pp914-919, 2002.

[20] S. Mori, C. Y. Suen, K. Yamamoto, "Historical review of OCR research and development," *Proceedings of the IEEE*, Vol. 80, Issue. 7, pp. 1029-1058, 1992.

[21] O. D. Trier, A. K. Jain, and T. Taxt, "Feature Extraction Methods for Character Recognition – A Survey," *Pattern Recognition*, Vol. 29, No. 4, pp. 641-662, 1996.

[22] L. Fedorovici, E. Voisan, F. Dragan and D. Iercan, "Improved Neural Network OCR based on Preprocessed Blob Cases," *Proc. Of International Joint Conference on Computational Cybernetics and Technical Informatics* (ICCC-CONTI), 2010.

[23] W. K. Pratt, "Digital Image Processing," Wiley, New York, 1991.

[24] M. Bokser, "Omnidocument Technologies," *Proceedings of the IEEE*, Vol. 80, pp 1066-1078, 1992.

[25] A. Khotanzad and Y. H. Hong, "Invariant Image Recognition by Zernike Moments," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 12, Issue. 5, pp. 489-497, 1990.

[26] S. O. Belkasim, M. Shridhar and A. Ahmadi, "Pattern Recognition with Moment Invariants: A Comparative Study and New Results," *Pattern Recognition*, Vol. 24, pp. 1117-1138, December, 1991.

[27] L. Devroye, L. Gyorfi, G. Lugosi, "A Probabilistic Theory of Pattern Recognition," *Springer-Verlag*, 1996.

[28] S. Lucas, E. Vidal, A. Amiri, S. Hanlon and J. C. Amengual, "A Comparison of Syntactic and Statistical Techniques for Off-line OCR," *Proc. Of International Colloquium on Grammatical Inference*, 1994.

[29] J. Cao, M. Ahmadi and M. Shridhar, "Handwritten Numeral Recognition with Multiple Features and Multistage Classifiers," *Proc. Of IEEE International Symposium of Circuits Systems*, Vol. 6, 1994.

[30] P. Shah, S. Karamchandani, T. Nadkar, N. Gulechha, K. Koli, K. Lad, "OCR-based chassis-number recognition using artificial neural networks," *Proc. Of IEEE International Conference on Vehicular Electronics and Safety* (ICVES), 2009.

[31] S. D. Budiwati, J. Haryatno, E. M. Dharma, "Japanese character (Kana) pattern recognition application using neural network," *Proc. Of International Conference on Electrical Engineering and Informatics* (ICEEI), 2011.

[32] J. Yi, Y. Peng, J. Xiao, "Using Multiple Frame Integration for the Text Recognition of Video," *Proc. Of International Conference on Document Analysis and Recognition*, 2009.

[33] Y. Chiang, C. A. Knoblock, "Recognition of Multi-oriented, Multi-sized, and Curved Text," *Proc. Of International Conference on Document Analysis and Recognition*, 2011.

[34] D. Brodic, D. R. Milivojevic, V. Tasic, "Preprocessing of binary document images by morphological operators," *Proc. Of International Convention of MIPRO*, 2011.

[35] A. Bhardwaj, F. Farooq, H. Cao, V. Govindaraju, "Topic Based Language Models for OCR Correction," *Proceedings of the second workshop on Analytics for noisy unstructured text data*, 2008.

[36] R. Jin, A. Hauptmann, C. Zhai, "A Content-based Probabilistic Correction Model for OCR Document Retrieval," *Proceedings of International ACJI SIGIR Conference on Research and Development in Information Retrieval Workshop Program*, 2002.

[37] M. Donoser, H. Bischof, S. Wagner, "Using Web Search Engines to Improve Text Recognition," *Proceedings of International Conference on Pattern Recognition*, 2008.

[38] S. M. Harding, W. B. Croft and C. Weir, "Probabilistic Retrieval of OCR Degraded Text Using N-Grams," *Proceedings of European Conference on Digital Libraries*, 1997.

Qinru Qiu received her M.S. and Ph.D. degrees from the department of Electrical Engineering at University of Southern California in 1998 and 2001 respectively. She received her B.S. degree from the department of Information Science and Electronic Engineering at Zhejiang University, China in 1994. Dr. Qiu is currently an associate professor at the Department of Electrical Engineering and Computer Science in Syracuse University. Before joining Syracuse University, she has been an assistant professor and then an associate professor at the Department of Electrical and Computer Engineering in State University of New York, Binghamton. Her research areas are energy efficient computing systems, energy harvesting real-time embedded systems, and neuromorphic computing. She has published more than 50 research papers in referred journals and conferences. Her works are supported by NSF, DoD and Air Force Research Laboratory.

Richard W. Linderman received his B.S., M.S. and Ph.D. degrees from the department of Electrical Engineering at Cornell University in 1980, 1981 and 1984, respectively. Dr. Linderman, a member of the scientific and professional cadre of senior executives, is the Chief Scientist, Information Directorate, Air Force Research Laboratory, Rome, N.Y. The Information Directorate leads the discovery, development and integration of affordable warfighting information technologies for air, space and cyberspace forces. Dr. Linderman serves as the directorate's principal scientific and technical adviser and primary authority for the technical content of the science and technology portfolio. He provides principal technical oversight of a broad spectrum of information technologies including fusion and exploitation; command and control; advanced architectures; information management; communications and networking; defensive information warfare; and intelligent information systems technologies. Dr. Linderman was commissioned as a second lieutenant in May 1980. Upon completing four years of graduate studies, he entered active-duty, teaching computer architecture courses and leading related research at the Air Force Institute of Technology. He was assigned to Rome Air Development Center in 1988, where he led surveillance signal processing architecture activities. In 1991, he transitioned from active-duty to civil service as a senior electronics engineer at Rome Laboratory, becoming a principal engineer in 1997. During these years, he pioneered three dimensional packaging of embedded architectures and led the Department of Defense community exploring signal and image processing applications of high performance computers. Dr. Linderman holds six U.S. patents and has published more than 70 journal, conference and technical papers.

Qing Wu received his Ph.D. degree from the department of Electrical Engineering at University of Southern California in 2002. He received his B.S. and M.S. degrees from the department of Information Science and Electronic Engineering at Zhejiang University (Hangzhou, China) in 1993 and 1995, respectively. Dr. Wu is currently a Senior Electronics Engineer at the United States Air Force Research Laboratory (AFRL), Information Directorate (RI). Before joining AFRL, he was an Assistant Professor in the Department of Electrical and Computer Engineering at State University of New York, Binghamton. His research interests include large-scale computational intelligence models, high-performance computing architectures, circuits and systems for energy-efficient computing. He has published more than forty research papers in international journals and conferences.

Morgan Bishop received his B.A. in 2004 from the Computer Science department at the State University of New York at Geneseo. Mr. Bishop is currently a Computer Scientist at the United States Air Force Research Laboratory (AFRL), Information Directorate, located in Rome, New York. Prior to joining AFRL, he was the Lead Developer for Jeansee Corporation where he investigated DNA binding algorithms to achieve optimal DNA codes for use in parallel computing architectures. His research interests include scalable algorithm development for heterogeneous high performance computers, basic research in next-generation massively parallel systems, and the development of brain-inspired intelligence models for real-world application. He has published more than fifteen research papers in journals and conferences throughout the world.

Robison E. Pino received the B.E. degree in electrical engineering with honors, summa cum laude, from the City University of New York, City College, NY in 2002. Dr. Pino received the M.Sc. degree and Ph.D. degree in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY in 2003 and 2005 respectively. Since 2009, Dr. Pino is a Senior Electronics Engineer at the United States Air Force Research Laboratory (AFRL). Dr. Pino worked from 2005 to 2009 at IBM as an Advisory Scientist/Engineer Development enabling advanced CMOS technologies, and as a Business Analyst within IBM's Photomask business unit. He also served during 2006 to 2009 as an adjunct professor at the University of Vermont where he taught electrical engineering courses at the graduate and undergraduate levels. Dr. Pino was a Distinguished Lecturer of IEEE, EDS, in 2010, AFRL Information Directorate Scientist/Engineer of the year in 2011, and Top 200 Most Influential Hispanics in Technology by HE&IT Magazine in 2011. Dr. Pino holds 2 patents, 4 pending, and has published over 40 technical papers including one book.