REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* MAY 2013	**2. REPORT TYPE** CONFERENCE PAPER (Post Print)	**3. DATES COVERED** *(From - To)* JUN 2010 – MAY 2013

4. TITLE AND SUBTITLE INTEGRATION, DEVELOPMENT AND PERFORMANCE OF THE 500 TFLOPS HETEROGENEOUS CLUSTER (CONDOR)	**5a. CONTRACT NUMBER** IN-HOUSE
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER N/A
6. AUTHOR(S) Mark Barnell, Qing Wu, Richard Linderman	**5d. PROJECT NUMBER** HPCC
	5e. TASK NUMBER IN
	5f. WORK UNIT NUMBER HO
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rome Research Site/RITB 525 Brooks Road Rome NY 13441-4505	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate Rome Research Site/RITB 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TP-2013-024

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA Case Number: 88ABW-2012-2709
DATE CLEARED: 8 May 2012

13. SUPPLEMENTARY NOTES
Proceedings ASME 2012 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE) Aug 12-15, 2012, Chicago, Illinois, USA. This is a work of the United States Government and is not subject to copyright protection in the United States.

14. ABSTRACT
The Air Force Research Laboratory Information Directorate Advanced Computing Division (AFRL/RIT) High Performance Computing Affiliated Resource Center (HPC-ARC) is the host to a very large scale interactive computing cluster consisting of about 1800 nodes. Condor, the largest interactive Cell cluster in the world, consists of integrated heterogeneous processors of IBM Cell Broadband Engine (Cell BE) multicore CPUs, NVIDIA General Purpose Graphic Processing Units (GPGPUs) and Intel x86 server nodes in a 10Gb Ethernet Star Hub network and 20Gb/s Infiniband Mesh, with a combined capability of 500 trillion floating operations per second (TFLOPS). Applications developed and running on CONDOR include large-scale computational intelligence models, video synthetic aperture radar (SAR) back-projection, Space Situational Awareness (SSA), video target tracking, linear algebra and others. This presentation will discuss the design and integration of the system. It will also show progress on performance optimization efforts and lessons learned on algorithm scalability on a heterogeneous architecture.

15. SUBJECT TERMS
cluster computing, heterogeneous computing, GPGPU

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Courtney Usmail
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	9	**19b. TELEPHONE NUMBER (*Include area code*)** N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

ASME 2012 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE)
August 12-15, 2012, Chicago, Illinois, USA

DETC2012-70083

Integration, Development and Performance of the 500 TFLOPS Heterogeneous Cluster (Condor)

Mark Barnell, Qing Wu and Richard Linderman
Air Force Research Laboratory
Information Directorate
Rome, New York, USA

ABSTRACT

The Air Force Research Laboratory Information Directorate Advanced Computing Division (AFRL/RIT) High Performance Computing Affiliated Resource Center (HPC-ARC) is the host to a very large scale interactive computing cluster consisting of about 1800 nodes. Condor, the largest interactive Cell cluster in the world, consists of integrated heterogeneous processors of IBM Cell Broadband Engine (Cell BE) multicore CPUs, NVIDIA General Purpose Graphic Processing Units (GPGPUs) and Intel x86 server nodes in a 10Gb Ethernet Star Hub network and 20Gb/s Infiniband Mesh, with a combined capability of 500 trillion floating operations per second (TFLOPS). Applications developed and running on CONDOR include large-scale computational intelligence models, video synthetic aperture radar (SAR) back-projection, Space Situational Awareness (SSA), video target tracking, linear algebra and others. This presentation will discuss the design and integration of the system. It will also show progress on performance optimization efforts and lessons learned on algorithm scalability on a heterogeneous architecture.

INTRODUCTION

The Affiliated Resource Centers (ARCs) are Department of Defense (DoD) Laboratories and Test Centers that acquire and manage High Performance Computing (HPC) resources as a part of their local infrastructure, but share their HPC resources with the broader DoD HPC user community via the High Performance Computing Modernization Program (HPCMP) which coordinates allocation of their HPC resources. In order to provide tomorrow's Air Force with massively parallel and scalable HPC applications, the software must be developed on large clusters. Unlike typical HPC clusters, all AFRL/RI clusters allow for interactive development and testing. In 2010, the AFRL Information Directorate won a two-million-dollar project, sponsored by the HPCMP, and built the Condor Cluster, which is DoD's largest interactive super computer as of November 2011. The Condor cluster consists of 84 Servers (2U Dual six-core Intel Westmere 5660, 24 or 48 GB RAM) each with 2 GPGPUs (NVIDIA C1060, C2050 or C2070s) [2]. The heterogeneous cluster has 22 Play Station 3s (PS3s) connected to each of the 78 server nodes (1716 PS3s in total).

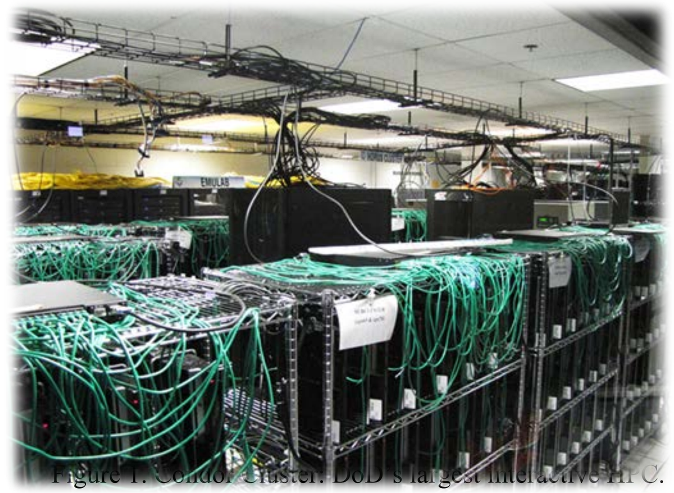


Figure 1. Condor Cluster: DoD's largest interactive HPC.

The long-term goal of AFRL/RI's high performance computing research is to provide the warfighters with Secure Embedded HPC (SEHPC) of the highest computing performance, under the Size-Weight-and-Power (SWaP) constraints. At the time when it was built, Condor was the largest, fastest and most energy-efficient interactive HPC in the Department of Defense.

The Condor HPC integrates the vast majority of the state-of-the-art HPC processing and networking architectures into one coherent functional system. This provides great R&D potentials and opportunities for the users so that they can explore and experiment with not only any single parallel computing architecture, but also any combinations of architectures, and evaluate their computing/communication performance and SWaP efficiencies under different programming and application scenarios. For processing architectures, the Intel Xeon server represents the multi-processor, super-scalar architecture; the NVIDIA Tesla GPGPU combines architectures of many-core, single-instruction-multiple-thread (SIMT, similar to SIMD), and streaming processing; the PlayStation 3 uses the IBM Cell BE processor, which adopts the multi-processor, single-instruction-multiple-data (SIMD, or vector processing) architecture. These three processors represent most of the modern high-performance processor architectures and cover a wide range of trade-offs among performance, power, size and weight.

DESIGN IMPLEMENTATION AND CONSTRAINTS

The Condor application development focuses on two related ongoing programs, one applied research effort and one basic research effort. The applied research focuses on voluminous generation of synthetic aperture radar (SAR) images providing persistent surveillance of city-sized areas with 1Hz update rate yielding a previously unachievable "video SAR capability" previously unachievable. The basic research effort investigates massively parallel neuromorphic architectures that can exploit the video SAR outputs, or alternative high resolution video cameras, to deliver robust perception, anticipation, and focus of attention.

The scalability and parallelism required to achieve sustained high computational throughputs demand low latency high bandwidth networking architectures. The Condor server nodes (custom built 2U X86 servers) were designed with both 20 Gb/s Infiniband and dual 10GbE network interface cards. This required the motherboard to support 48 PCI-E Gen2 (two Intel 5200 chipsets, 2x IOH-36D), allowing for four 16x Gen 2 slots. This supports maximum data throughput to all four PCI-E devices: two NVIDIA GPGPUs and the two network cards.

In a star-hub topology, 39 IBM BLADE RackSwitch G8000 Gigabit Ethernet spoke switches are connected to the PS3 compute nodes and aggregated to 12 RackSwitch G8100 10 Gigabit Ethernet switches. Dual 10 Gigabit Ethernet links are bonded for high-bandwidth switch-to-switch communications. The IBM BLADE RackSwitch G8100s are connected to the Condor server nodes. The IBM BLADE RackSwitch G8100's CX4 transceivers ensure low transmission latency with an average of 60 to 70 microseconds even when going through three switches.

The condor server nodes can also communicate between each of the 78 nodes through an Infiniband mesh. This allows for very low latency and high bandwidth when applications only require the x86 processors and GPGPUs. While running bench mark tests and network OpenMPI applications, we routinely achieved a sustained 25-28 Gb/s performance across the entire network.

Figure 2. Condor server node.

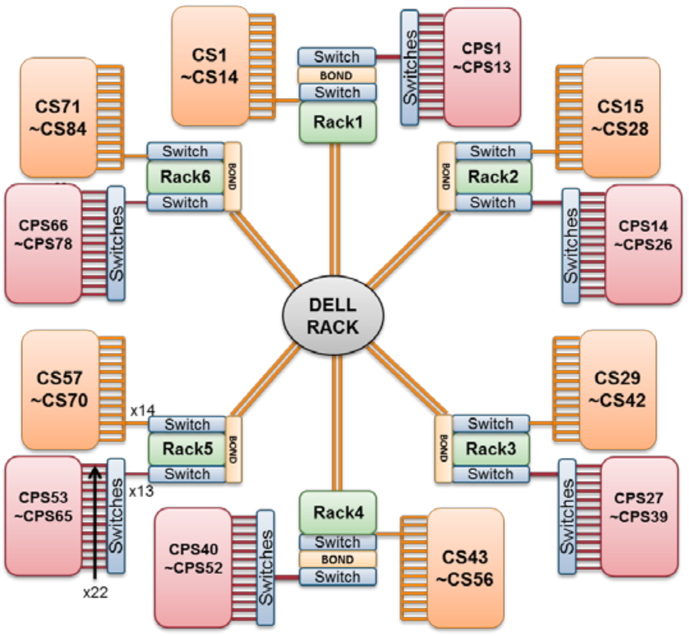


Figure 3. Bonded 10Gb Ethernet Blade switches.

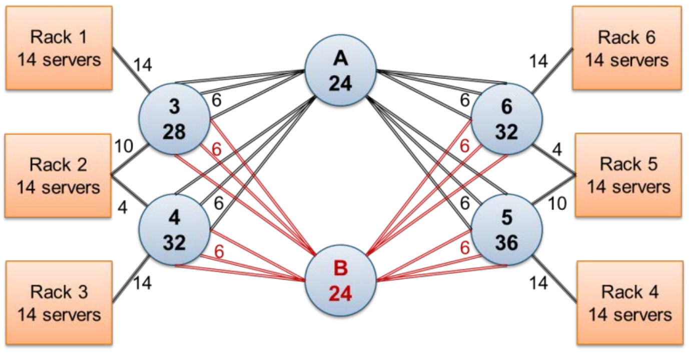


Figure 4. Infiniband mesh non-blocking 20Gb/s.

The design of the Condor HPC system had physical constraints and limitations. As shown in Figure 5, the actual footprint of the system, layout, power and cable trays were chosen carefully to allow for maximum cooling and minimum cable lengths.

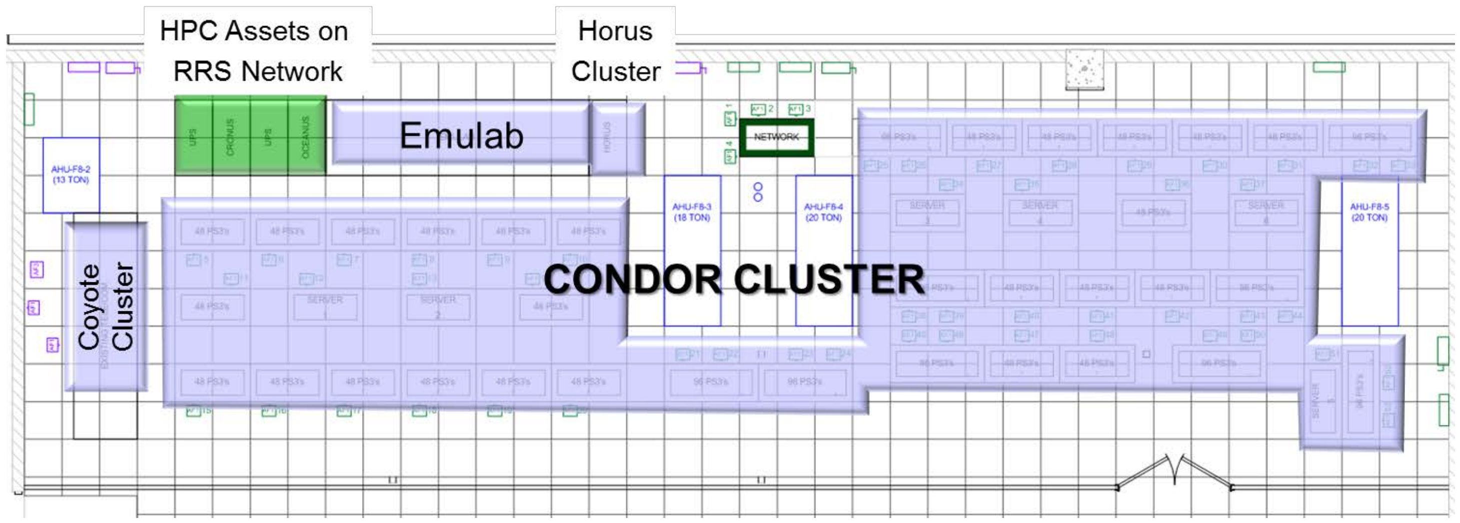


Figure 5. Condor physical layout.

PRIMARY APPICATIONS

IMAGE PROCESSING: This applied research work focuses on voluminous generation of synthetic aperture radar (SAR) images providing persistent surveillance of city-sized areas with a 1 Hz update rate. The signal and image processing application that produces the best quality formed image is the Backprojection algorithm. This algorithm in its purest form requires N^3 computations and, when processing billions of pixels, requires a sustained performance of hundreds of teraflops.

Implementation of Backprojection on the Playstations: The original Backprojection algorithm processes each range vector completely before moving on to the next. Since each range vector contributes energy to every pixel in the scene, this method requires the reading and writing of all pixel memory locations for each range vector processed. To optimize this process, the algorithm was improved to calculate each pixel value over a number of range vectors, reducing the total number of memory accesses.

Because of the limited amount of memory available on the PS3, the number of pixels and the amount of RADAR data had to be reduced in order to fit. An additional procedure was designed to divide the pixel matrix into smaller rectangular sections that can be assigned to the available processors. Because the range data required for calculating the pixel values for each of these sections exists as a contiguous segment of each range vector, code has been written to extract only the necessary data needed by each processor. The combination of reducing the pixel area and the required data allows the algorithm to be scaled to fit within the available memory.

The original code was written to store variables and perform all calculations using double precision data. Since the PS3 performs single precision operations about 10X faster than double precision [3,5], most of the calculations are performed in single precision to maximize performance. Single precision operations also reduce the memory size requirements.

The pulse compression portion of the process is performed on the Xeon processors. The resulting data was then transmitted to the PS3s for image formation. The data storage format difference was also taken care of, with the Xeon in little endian format and the PS3 in big endian format. The Xeon processors perform the byte rearrangement to accommodate the PS3s.

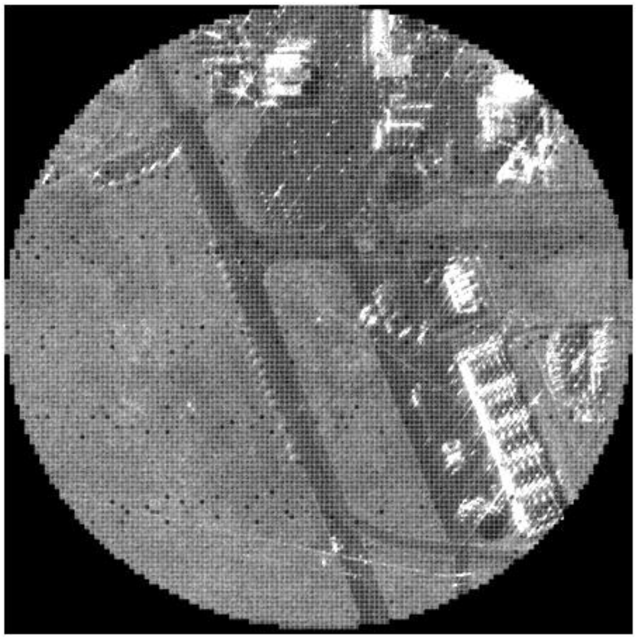


Figure 6. 1-km image illustrating processing distribution.

The image in Figure 6 was formed by a PS3 by dividing the area into 12,408 32x32 pixel blocks that were distributed to the SPE processors. By doing this, the number of range bins required to be in SPE's memory was about 80 of the overall 32k range bins that make up an entire vector. The memory required to contain the image is also reduced since each SPE only has to hold the 32x32 block that it's working on. Figure 6 also shows the formed image being reassembled with the grid lines to show the processing regions.

Performance & Optimizations: Performance was measured in compute time and FLoating Point Operations executed per second (FLOPs). A reduction in execution time along with an increase in FLOPs executed per second demonstrates improved performance.

There are 96 floating point operations for each pixel, for each pulse in the inner most loop. This loop is iterated over each pixel and each pulse. Removing only 1 FLOP removes 1600 FLOPs per pulse per (40 by 40) pixel chunk. Square Roots, divisions, sines and cosines are indeterminate and expensive calculations. Each platform has its own method of calculating these values. Some platforms have hardware instructions, while others rely on software libraries. In the case of the PS3, approximately 16 clock cycles are required to perform one double precision square root. To make an "apples to apples" comparison, the following FLOP assignments were made for each platform:

• Double precision square root	19 Flops
• Single Precision sin	25 Flops
• Single Precision cos	25 Flops

One of the computations in the inner loop is a distance calculation requiring a double precision square root. It is used to determine the difference, RANGEDIFF, between R (sensor to pixel) and R0 (sensor to spot center. R0 was given, but R needed to be calculated.

The following optimizations were aimed at reducing the requirement for the square root through approximation methods.

The first optimization attempt was to calculate the RANGEDIFF for the first and last pulse (512) in the buffer, then to perform one" Newton-Raphson" iteration for each pixel, for each pulse in between. Newton-Raphson is an iterative algorithm that converges on the final square root value. By using the close approximation described above as the starting point, it was hoped that one iteration would be required to return an accurate value. It turned out not to be accurate enough, even performing a complete square root every 32 pulses. The wavelength of about 3 cm requires accuracy to 3 decimal places; the algorithm was only accurate to 2 decimal places within in one row of 32 pixels.

While analyzing the data during the implementation of the previous optimization, it was noted that R appeared to change linearly. Five thousand pulses were placed in a spread sheet, and R was calculated and plotted. R for a given pixel does change linearly. This may not always be the case, but with a high PRF it may be likely. The PRF for this data was 7500 Hz; the change in sensor position over 512 pulses should be relatively small.

The second optimization approach leveraged the linearity of R. For each buffer of 512 pulses, and for each pixel in a chunk of pixels, R was calculated for the first pulse and the last pulse. The average difference was taken as an "increment" to the first R. This reduced the number of floating point operations in the inner loop from 96 to 71, while adding some overhead before entering the loop. Accuracy was maintained out to 4 decimal places.

Table 1 shows the FLOPs comparison before and after optimization for one chunk of pixels processed against 512 pulses. The optimized version executes about 257 fewer GFLOPs than the un-optimized version.

Table 1. FLOPs executed per pixel. 10 m by 10 m chunks.

Gflops Executed	Not Optimized	Optimized Flops
Compute Loop	78,669,504	58,163,200
+ Loop Overhead	0	86,484
Flops per Chunk per 512 Pulses	78,669,504	58,249,684
x Num chunks	210	210
	16,520,595,840	12,232,433,640
x Num SPEs	6	6
	99,123,575,040	73,394,601,840
x Num Pulse buffers	10	10
Flops Executed	991,235,750,400	733,946,018,400
Total Gflops Executed	**991.24**	**733.95**

Table 2 summarizes the performance results on a per chunk basis and compares the PS3 with the Xeon processor based on cost and energy usage per GFLOP.

Table 2. Performance metrics: 10 m by 10 m chunks.

Per Node	PS3	Optimized PS3	X86	Optimized X86
Gflops Executed	991.24	733.95	991.24	733.95
Compute Secs	32.10	20.79	16.17	13.08
Gflops/sec	31	35	61	56.10
Cost $ per Node	$380	$380	$3,000	$3,000
Cost $ per Gflop	$12.30	$10.76	$48.93	$53.48
Watts per Node	111	111	513	513
Watts/Gflops	3.59	3.14	8.37	9.14
Gflops/Watt	0.28	0.32	0.12	0.11

It can be seen from Table 2 that for this application the PS3 has the advantage over the Xeon for power and cost, while the Xeon has the advantage of speed. The PS3 relative to the Xeon is:

- ~ 1/5th the cost per GFLOP
- ~ 1/3rd the power per GFLOP
- ~ 1/8th the initial cost
- ~ 1 1/2 times slower

In order to process the entire 5 km image, 1664 PS3's would be required and each requires 32 MBs of data every 20.79 seconds.

NEUROMORPHIC COMPUTING: Brain-inspired signal processing algorithms and flow possess great potentials to be applied to many cognitive applications such as image processing, intrusion 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 Software (ITRS) was developed on the Condor HPC [1]. The software architecture of ITRS incorporates the Condor HPC technologies with advances in neuromorphic computing models.

The overview of the implementation of the ITRS is shown in Figure 7. 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 mostly 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 the cluster level, the system behaves asynchronously.

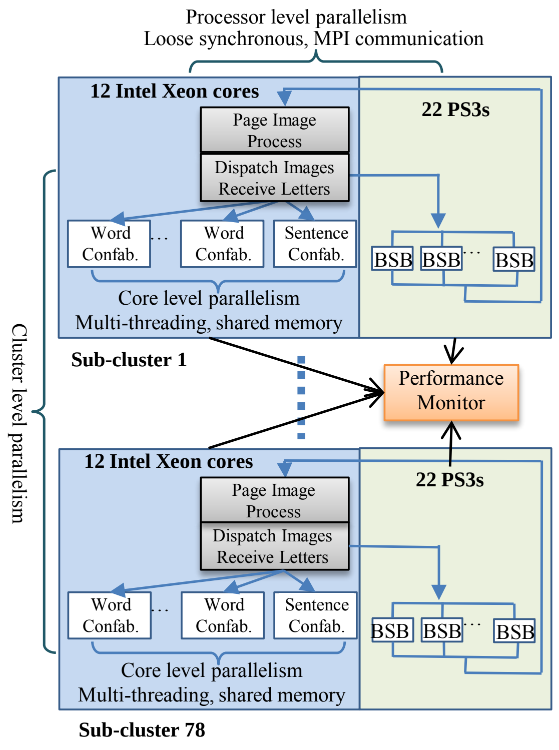


Figure 7. Overview of the ITRS implementation.

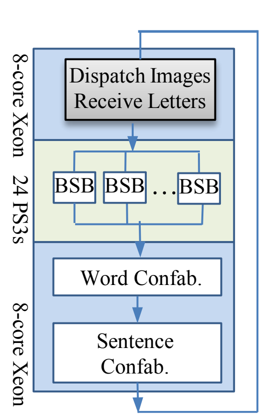
Upon receiving the page image, the head node first slices the image into small blocks, each of which contains one character. The blocks are dispatched to the PS3s, on which the BSB recalls are run for character recognition. The results are sent back to the head node for word level and sentence level 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).

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 200 MB 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.

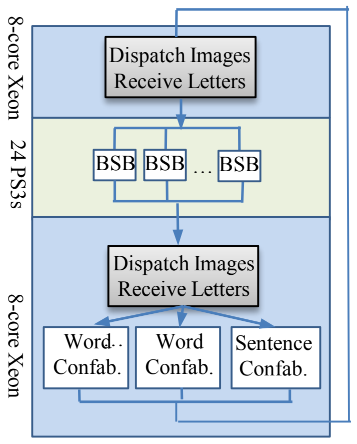
Overall, the implemented ITRS software is able to process about 16 to 20 scanned pages per second on the Condor HPC with reasonable efforts in performance optimization.

Figure 8 shows the evolution of the ITRS software architecture over time. We started with a baseline implementation as shown in Figure 8 (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 Figure 8 (b).

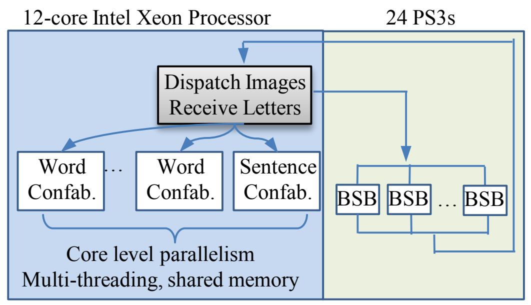
To evaluate the performance of the ITRS software, we carried out experiments on three different input test cases. In the first input file 20% of character images are scratched by 1-pixel-wide horizontal bars. Compared to the other two test cases, it has the highest image quality. The second input file 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 sizes of the input/output buffers in the double buffering system are set to be 100 sentences.



(a) Base line ITRS



(b) Multi-threading ITRS


12-core Intel Xeon Processor
24 PS3s
Dispatch Images Receive Letters
Word Confab.
Word Confab.
Sentence Confab.
BSB BSB … BSB
Core level parallelism
Multi-threading, shared memory

(c) Parallel ITRS

Figure 8. Evolution of the ITRS software architecture.

Figure 9 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%.

The results in Figure 9 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. Figure 10 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%.

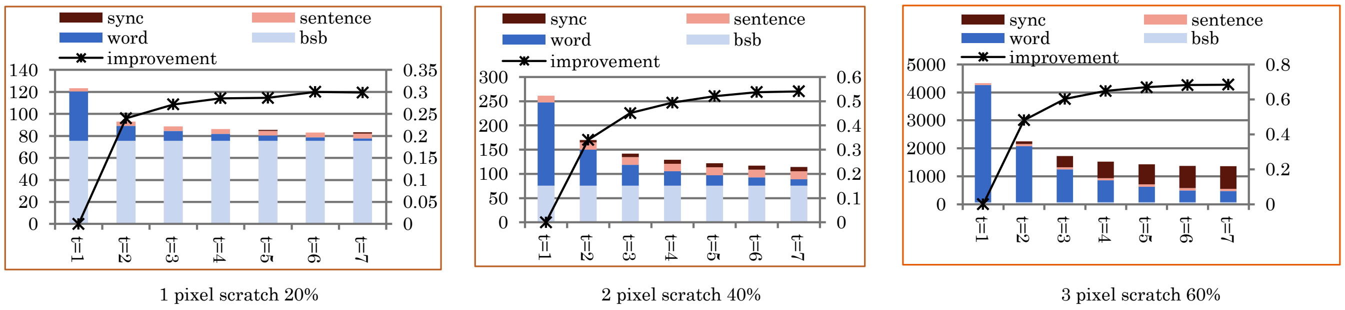


Figure 9. Performance improvement by multi-threading confabulation.

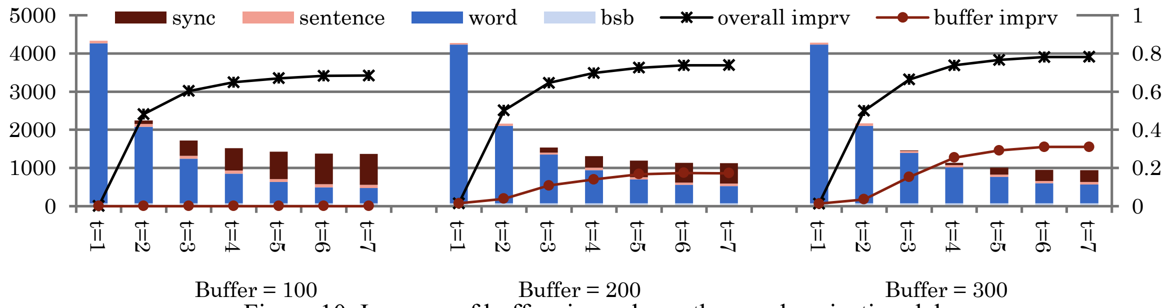
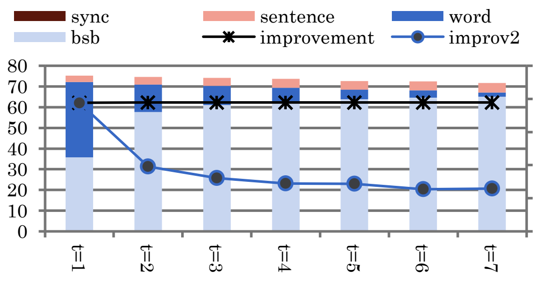
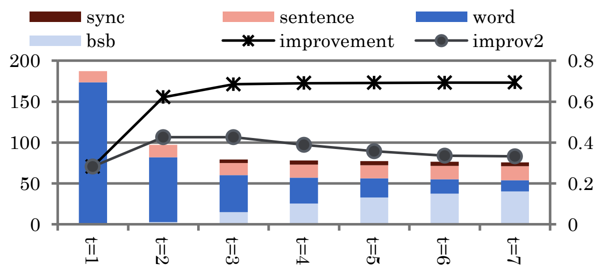


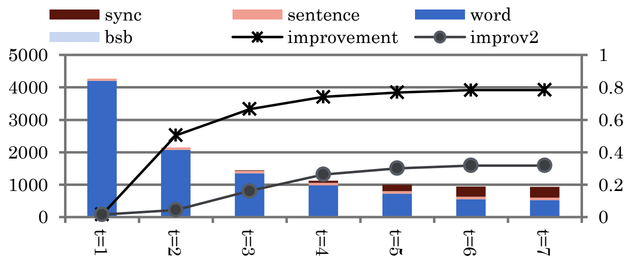
Figure 10. Increase of buffer size reduces the synchronization delay.



(a) Results for high quality test case



(b) Results for medium quality test case



(c) Results for low quality test case

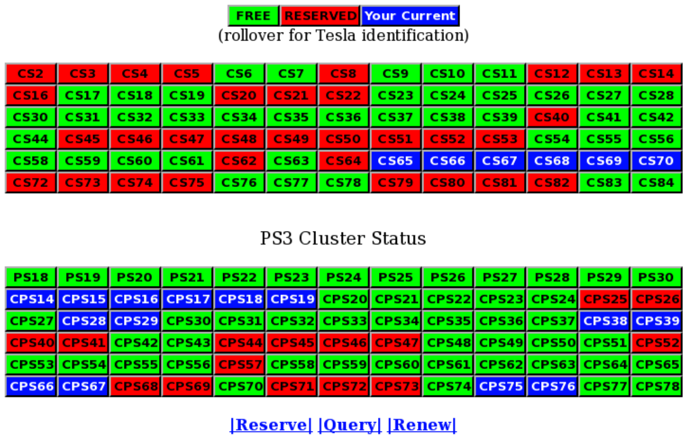
Figure 11. Performance improvement by parallelizing BSB and confabulation.

We further improve the ITRS software architecture by parallelizing the BSB and confabulation processes, as shown in Figure 8 (c). Figure 11 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%.

ENERGY-EFFICIENT INTERACTIVE SYSTEM

Deployment and development of the Condor supercomputer was configured for two primary objectives: interactive (on-demand) and energy-efficient (green) computing. Interactive computing provides the users with direct access to the resources based on their schedule and scalability needs [4]. When the applications and software development activities use only a portion of Condor, the rest can be put in shutdown or put to sleep mode for significant energy savings. This has major impacts on the facility's infrastructure and costs.

The current 100+ Condor users can login into one of six login severs and begin by reserving server nodes and PS3 clusters. Figure 12 shows the Condor status and reservation system as web-based user interface.



🟩 FREE 🟥 RESERVED 🟦 Your Current

(rollover for Tesla identification)

🟥 CS2	🟥 CS3	🟥 CS4	🟥 CS5	🟩 CS6	🟩 CS7	🟥 CS8	🟩 CS9	🟩 CS10	🟩 CS11	🟥 CS12	🟥 CS13	🟥 CS14
🟥 CS16	🟩 CS17	🟩 CS18	🟩 CS19	🟥 CS20	🟥 CS21	🟥 CS22	🟩 CS23	🟩 CS24	🟩 CS25	🟩 CS26	🟩 CS27	🟩 CS28
🟩 CS30	🟩 CS31	🟩 CS32	🟩 CS33	🟩 CS34	🟩 CS35	🟩 CS36	🟩 CS37	🟩 CS38	🟩 CS39	🟥 CS40	🟩 CS41	🟩 CS42
🟩 CS44	🟥 CS45	🟥 CS46	🟥 CS47	🟥 CS48	🟥 CS49	🟥 CS50	🟥 CS51	🟥 CS52	🟥 CS53	🟩 CS54	🟩 CS55	🟩 CS56
🟩 CS58	🟩 CS59	🟩 CS60	🟩 CS61	🟥 CS62	🟩 CS63	🟥 CS64	🟦 CS65	🟦 CS66	🟦 CS67	🟦 CS68	🟦 CS69	🟦 CS70
🟥 CS72	🟥 CS73	🟥 CS74	🟥 CS75	🟩 CS76	🟩 CS77	🟩 CS78	🟥 CS79	🟥 CS80	🟥 CS81	🟥 CS82	🟩 CS83	🟩 CS84

PS3 Cluster Status

🟩 PS18	🟩 PS19	🟩 PS20	🟩 PS21	🟩 PS22	🟩 PS23	🟩 PS24	🟩 PS25	🟩 PS26	🟩 PS27	🟩 PS28	🟩 PS29	🟩 PS30
🟦 CPS14	🟦 CPS15	🟦 CPS16	🟦 CPS17	🟦 CPS18	🟦 CPS19	🟩 CPS20	🟩 CPS21	🟩 CPS22	🟩 CPS23	🟩 CPS24	🟥 CPS25	🟥 CPS26
🟩 CPS27	🟦 CPS28	🟦 CPS29	🟩 CPS30	🟩 CPS31	🟩 CPS32	🟩 CPS33	🟩 CPS34	🟩 CPS35	🟩 CPS36	🟩 CPS37	🟦 CPS38	🟦 CPS39
🟥 CPS40	🟥 CPS41	🟩 CPS42	🟩 CPS43	🟥 CPS44	🟥 CPS45	🟥 CPS46	🟥 CPS47	🟩 CPS48	🟩 CPS49	🟩 CPS50	🟩 CPS51	🟥 CPS52
🟩 CPS53	🟩 CPS54	🟩 CPS55	🟩 CPS56	🟥 CPS57	🟩 CPS58	🟩 CPS59	🟩 CPS60	🟩 CPS61	🟩 CPS62	🟩 CPS63	🟩 CPS64	🟩 CPS65
🟦 CPS66	🟦 CPS67	🟥 CPS68	🟥 CPS69	🟩 CPS70	🟥 CPS71	🟥 CPS72	🟥 CPS73	🟩 CPS74	🟦 CPS75	🟦 CPS76	🟩 CPS77	🟩 CPS78

|Reserve| |Query| |Renew|

Figure 12. Condor status and reservation page.

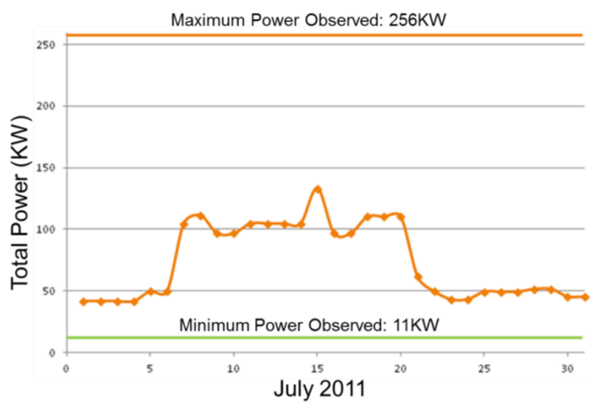


Figure 13. Condor power consumption.

The PS3s are configured with Fedora 9 or Yellow-Dog Linux (YDL) and included with the bootloader and operating system is the wake-on-LAN option. This option allows all 1716 PS3s to be put in a power savings mode (sleep). A PS3's typical idle power draw is 95 watts and 5 watts in sleep mode. The PS3s will consume 67 percent of the total 256 KWs when the entire Condor cluster is operational. The systems reservation mirrors the power draw is shown in Figure 13. The typical HPC system will run all of the nodes in idle mode, using up to 70% of the peak system power. Condor typically runs around 40% of peak during the work week, and 18% on the weekends. The estimated power cost saving is $219,964.00/yr and this achieves a reduction of 792 tons of carbon footprint on the environment [6].

PREPARING FOR THE FUTURE

Large scale computing systems provide the basis to investigate and implement solutions for C4ISR challenges. Fundamental for many of the Data-to-Decision problems is the ability to perceive, fuse, and exploit information within voluminous flows from increasingly capable and affordable sensors monitoring the air, space, and cyber domains. Signal and image processing, such as creating the video SAR capability, present significant computational loads near the sensor which then feed the even more challenging tasks of recognition, information fusion, tracking, and exploitation based upon this flood of imagery. HPC systems and the Condor cluster support basic research into massively parallel neuromorphic models at scales approaching that of the human neocortex for robust visual perception and recognition (Figure 14).

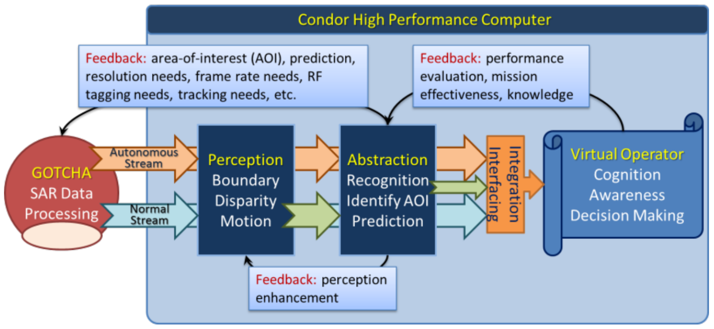

Condor High Performance Computer
Feedback: area-of-interest (AOI), prediction, resolution needs, frame rate needs, RF tagging needs, tracking needs, etc.
Feedback: performance evaluation, mission effectiveness, knowledge
GOTCHA SAR Data Processing
Autonomous Stream
Normal Stream
Perception Boundary Disparity Motion
Abstraction Recognition Identify AOI Prediction
Integration Interfacing
Virtual Operator Cognition Awareness Decision Making
Feedback: perception enhancement

Figure 14. C4ISR autonomous sensing framework.

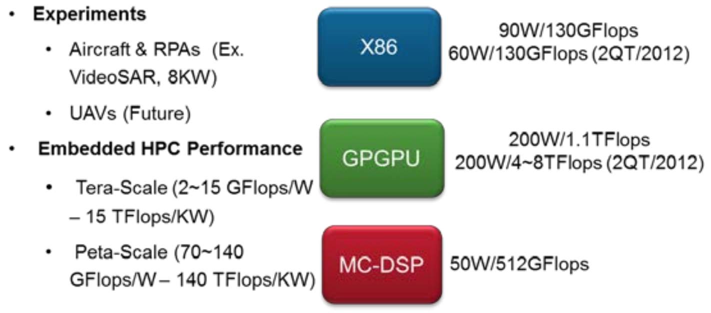

Experiments
Aircraft & RPAs (Ex. VideoSAR, 8KW)
UAVs (Future)
Embedded HPC Performance
Tera-Scale (2~15 GFlops/W – 15 TFlops/KW)
Peta-Scale (70~140 GFlops/W – 140 TFlops/KW)
X86
90W/130GFlops
60W/130GFlops (2QT/2012)
GPGPU
200W/1.1TFlops
200W/4~8TFlops (2QT/2012)
MC-DSP
50W/512GFlops

Figure 15. Plan of embedded HPC under SWaP constraints.

We continue to expand our HPC portfolio and relationships with HPCMP and tailor our capabilities to solve significant Air Force challenges. Embedded HPC systems will be developed and integrated close to the sensor, enabling processing of high volume data with greatly improved information content. We are developing hybrid scalable computing framework for imagery information exploitation, real-time and autonomous sensing and deciding technologies on our Condor cluster. The scalable computing framework will be robust enough to run on tomorrows HPC architectures (Figure 15).

CONCLUSION

We have presented an interactive HPC supercomputer, Condor, which has been developed and designed to be energy-efficient and interactive with users. Condor provides the Air Force and the DoD community the ability to prototype, develop and evaluate large-scale massively parallel HPC applications.

ACKNOWLEDGMENTS

The contractor’s work is supported by the Air Force Research Laboratory, under contract FA8750-10-C-0216.

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] 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.

[2] “Board Specfications Tesla C2050 and Tesla C2070 Computing Processor Board.” *NIVIDIA Corporation*, July 2010. http://www.nvidia.com/docs/IO/43395/BD-04983-001_v03.pdf.

[3] Buttari, A., J. Dongarra, and J. Kurzak, “Limitations of the PlayStation 3 for High Performance Cluster Computing” LAPACK Working Note 185, 2007.

[4] Feng, W., X. Feng, and R. Ge, “Green Supercomputing Comes of Age.’ *IT Professional*, 10, 1, pp. 17-23, 2008.

[5] IBM, “Cell Broadband Engine,” URL: https://www-01.ibm.com/chips/techlib/techlib.nsf/products/Cell_Broadband_Engine, last modified May 2009. Accessed April 27, 2012.

[6] “PUE and DCiE Data Center Efficiency Measurement and Benchmarking,” URL: http://www.42u.com/measurement/pue-dcie.htm, last modified March 2012. Accessed April 27, 2012.