Development of a Compact Cluster with Embedded CPUs

Sritrusta Sukaridhoto[1,2], Yoshifumi Sasaki[3], Koichi Ito[2], and Takafumi Aoki[2]
[1]Electrical Engineering Polytechnic Institute of Surabaya (EEPIS),
Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia
[2]Graduate School of Information Sciences, Tohoku University, Aoba-yama 05, Sendai-shi 980-8579, Japan
Tel: +81-22-217-7169, Fax: +81-22-263-9308
[3]Faculty of Science and Engineering, Ishinomaki Senshu University, Ishinomaki-shi, 986-8580, Japan
E-mail: dhoto@aoki.ecei.tohoku.ac.jp

Abstract

This paper presents a compact cluster computer with embedded CPUs, called "UCC (Ubiquitous Computing Cluster)", in order to provide a cost-effective prototyping environment for design and test of ubiquitous computing applications. We achieve extremely small size, low power consumption and low cost by employing COTS (Commercial Off-The-Shelf) embedded products. As an application example, we also discuss implementation of a fingerprint matching algorithm using Phase-Only Correlation (POC) technique.

Keyword: *ubiquitous computing, cluster computer, embedded CPUs, fingerprint matching, phase-only correlation*

1. Introduction

Today, embedded processors (CPUs) can be found in a vast variety of products ranging from cellular phones, digital cameras and automobile navigation systems up to network-connected household appliances. Some of these embedded CPUs can run advanced operating systems, such as Linux, to achieve flexible network connectivity and to have logically same functionality as that of high-end CPUs designed for PCs and workstations. This trend accelerates the technology toward the age of "ubiquitous computing", that is to integrate computation into the environment enabling people to interact with computers more naturally [1]. In such situation, distributed parallel processing with network-connected embedded CPUs will become one of the most important technologies to realize a variety of pervasive applications.

One of the problems in managing R&D projects for ubiquitous/pervasive computing is the lack of cost-effective standardized platform for prototyping, testing and evaluating application programs on network-connected embedded CPUs. Addressing this problem, in this paper, we present a compact cluster computer with embedded CPUs, called a "Ubiquitous Computing Cluster (UCC)" [2], which provides a rapid-prototyping environment for ubiquitous/pervasive computing applications at very low cost compared with conventional PC clusters.

UCC consists of four computing nodes and a network switch (100Mbps Fast Ethernet) mounted together within a small skeleton rack. The key idea is to fully utilize COTS (Commercial Off-The-Shelf) embedded products to realize cost-effective prototyping environment. In this context, we carefully selected a commercially available Network Attached Storage (NAS) as a computing node for UCC, which consists of an embedded CPU, a memory, a hard disk drive, a network adapter and a USB interface. The four computing nodes run Linux with some daemons and libraries required to make inter-processor communication for parallel processing, where MPI (Message Passing Interface) [3], [4] and PVM (Parallel Virtual Machine) [5] could be employed for parallel programming.

By the use of COTS products, we achieve extremely compact size of 390mm x 280mm x 150mm, low power consumption of 60W (typical) and low cost. Thus, UCC can be easily introduced to educational programs in universities for Linux-based cluster computing. Another interesting feature of UCC is that every computing node has a USB interface, and hence UCC could be easily extended to various real-world application systems using USB-based sensors, such as USB cameras.

This paper is organized as follows: Section 2 gives the system overview of UCC. Section 3 describes performance test for basic data transfer bandwidth through MPI. Section 4 discusses an application of UCC to parallel fingerprint matching using the Phase-Only Correlation (POC) technique [6]. In Section 5, we end with some conclusion.

2. System Overview

Figure 1 shows the overall architecture of the compact cluster computer, called a "Ubiquitous Computing Cluster (UCC)" [2]. It consists of four computing nodes #0, #1, #2 and #3, where the node #0 works as a server node for various applications, such as NIS, NFS, ftp, telnet, etc., and is directly accessible from terminals outside. These four computing nodes and a network switch are mounted together within a small skeleton rack. The computing nodes are connected over a conventional 100Mbps Fast Ethernet.

We decided to use COTS (Commercial Off-The-Shelf) products with embedded CPUs in building UCC. A Network Attached Storage (NAS) is employed as a computing node, which consists of an embedded SH4 CPU (266MHz), a 64MB SDRAM, a 120GB HDD, a 100Mbps Fast Ethernet interface and a dual port USB 2.0 interface. The detailed specification of NAS used for a computing node is shown in Table 1.

The four computing nodes #0, #1, #2 and #3 run the Debian GNU Linux 2.4.21 configured for SH4 [7] and daemons of remote execution services such as rsh, rexec and rcp to make communications with each other in parallel processing. The server node #0 also runs NIS and NFS services to manage login IDs and shared file system of UCC. In the server node, also telnet and ftp services are prepared to allow login and file transfer from the terminals outside. As a parallel programming environment, C, C++ and Fortran compilers, vi and GNU Emacs editors, message passing interface (MPI) [3], [4] of mpich and parallel virtual machine (PVM) [5] are installed in advance.

Figure 2 shows a photograph of a prototype of the four-node UCC. We could achieve extremely compact size of 390mm x 280mm x 150mm, low power consumption of 60W (typical) and low cost by employing the COTS embedded devices. The detailed specification is shown in Table 2. It can be easily introduced to any situation such as educational programs for Linux-based cluster computing in universities and companies.

One of the valuable features of UCC is that every computing node has USB 2.0 interface and could be easily extended to various application systems using USB-based sensors. Figure 3 shows UCC connected with other devices using USB interface. This feature allows us to use UCC in various R&D projects for future ubiquitous computing applications employing a network of embedded CPUs. UCC may be useful as a prototyping environment for developing applications, such as ubiquitous computer vision systems using embedded cameras, ubiquitous personal identification/authentication systems using embedded biometrics sensors, ubiquitous speech recognition systems using embedded microphone arrays, etc.

UCC provided with special design box that makes UCC can easily connected each others. Figure 4 shows a photograph of several UCC connected together.

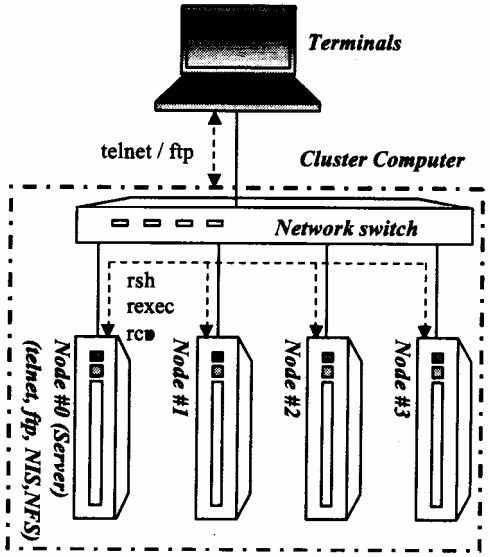

Figure 1: Architecture of the UCC.

Table 1: Specification of the NAS as a computing node

CPU	SH4 (SH7751R, 266MHz)
Memory	64MB SDRAM
HDD	120GB, ATA133, 5400rpm
NIC	10/100 BASE-T (RTL-8139C+)
I/F	USB 2.0×2port
OS	Debian GNULinux 2.4.21

Table 2: Overall specification of UCC.

Employed computing nodes		Embedded NAS
Number of computing nodes		4
Network interface		10/100 BASE-T
Power consumption		60W (TYP)
Size [mm]		W390 x D280 x H150
Software	OS	Debian GNU Linux 2.4.21
	Server functions(*)	NIS, NFS
	Communication functinos	telnet,ftp,rsh,rexec,rcp
	Development environment	GNU C-3.0.4, C++-3.0.4, F77
		vi, GNU Emacs
		MPI(mpich-1.2), PVM-3.0

(*)Only to the server node

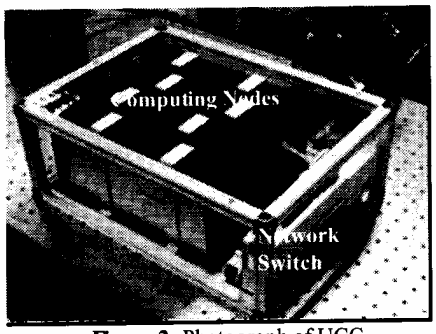

Figure 2: Photograph of UCC.

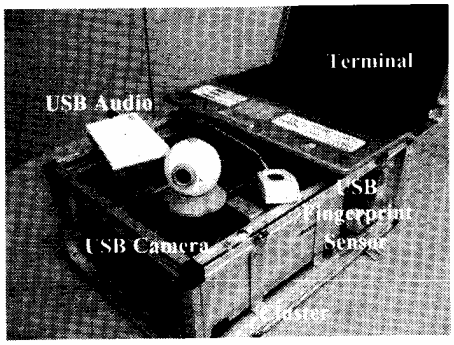

Figure 3: UCC connected with other devices.

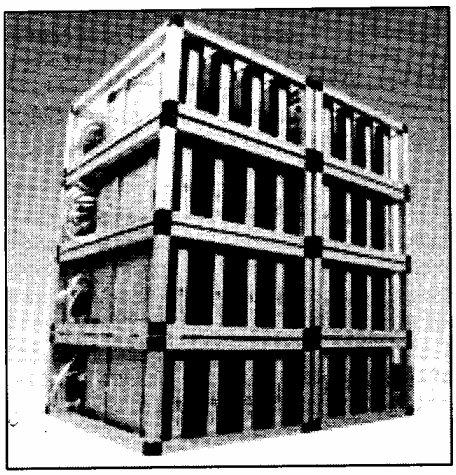

Figure 4: UCC connected with other UCC.

3. Message Passing Performance

In this section, we describe performance of basic MPI functions on UCC. Pallas MPI benchmark (PMB) [8] is employed to evaluate data transfer bandwidth and latency of basic ping-pong and broadcast communications. Figure 5 shows the result of ping-pong test, where two processors send and receive messages alternatively. While the data size is small, throughput of the ping-pong communication is low. As the data size increases, the bandwidth saturates around the data size of 512K bytes. Peak performance of ping-pong communication is estimated as about 9Mbytes/sec (~70Mbps) and this may be reasonable considering the maximum speed of a conventional 100Mbps Fast Ethernet.

Figure 6 shows the result of broadcast communication, where benchmark measures slower communication time for broadcasting a message from the server node #0 to the other nodes #1, #2 and #3. The behavior of broadcast is similar to that of ping-pong communication, but the peak performance is estimated as about 4.5Mbytes/sec, which is also reasonable performance. As a result of these benchmark tests, it seems that UCC has enough communication performance for practical applications of parallel processing.

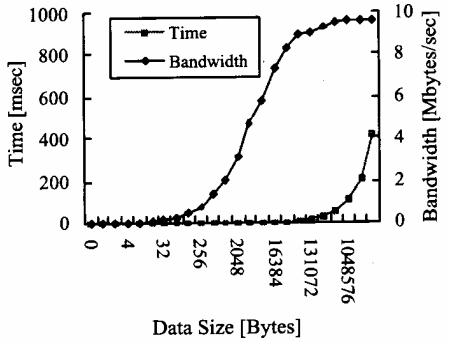

Figure 5: Benchmark for ping-pong communication.

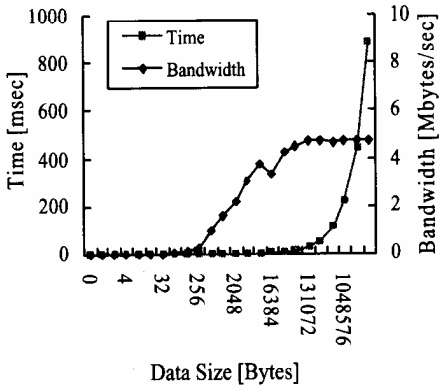

Figure 6: Benchmark for broadcast communication.

4. Application to Fingerprint Verification

In this section, we describe an application of UCC to fingerprint matching using Phase-Only Correlation (POC). The POC technique has been successfully applied to high-accuracy image registration tasks for computer vision applications [9-11], where estimation of sub-pixel image translation is a major concern. The same technique is also effective for fingerprint matching [6]. Figure 9 shows the fingerprint matching algorithm using POC, where the original algorithm [6] is simplified to be executed on UCC. In this experiment we use *Fujitsu MBF-200* fingerprint USB sensor.

Consider two *N1 x N2* images, $f(n_1,n_2)$ and $g(n_1,n_2)$, where we assume that the index ranges are $n_1 = -M_1 ... M_1$ (M_1>0) and

Figure 7: Photograph of experiment fingerprint verification system.

$n_2 = -M_2 ... M_2$ (M_2>0) for mathematical simplicity, and hence $N_1 = 2M_1 + 1$ and $N_2 = 2M_2 + 1$. Let $F(k_1,k_2)$ and $G(k1,k2)$ denote the 2D Discrete Fourier Transforms (2D DFTs) of the two images. $F(k1,k2)$ and $G(k1,k2)$ are given by

$$F(k_1,k_2) = \sum_{n_1,n_2} f(n_1,n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} = A_F(k_1,k_2) e^{j\theta_F(k_1,k_2)} \quad (1)$$

$$G(k_1,k_2) = \sum_{n_1,n_2} g(n_1,n_2) W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2} = AG(k_1,k_2) e^{j\theta_G(k_1,k_2)} \quad (2)$$

where $k_1 = -M_1 ... M_1$, $k_2 = -M_2 ... M_2$, $W_{N_1} = e^{-j\frac{2\pi}{N_1}}$, $W_{N_2} = e^{-j\frac{2\pi}{N_2}}$, and the operator $\sum_{n_1,n_2}$ denotes $\sum_{n_1=-M_1}^{M_1} \sum_{n_2=-M_2}^{M_2}$. $A_F = (k_1,k_2)$ and $A_G = (k_1,k_2)$ are amplitude components, and $e^{j\theta_F(k_1,k_2)}$ and $e^{j\theta_G(k_1,k_2)}$ are phase components. The cross spectrum $R_{FG}(k_1,k_2)$ between $F(k_1,k_2)$ and $G(k_1,k_2)$ is given by

$$R_{FG}(k_1,k_2) = F(k_1,k_2)\overline{G(k_1,k_2)} = A_F(k_1,k_2) A_G(k_1,k_2) e^{j\theta(k_1,k_2)} \quad (3)$$

where $\overline{G(k_1,k_2)}$ denotes the complex conjugate of $G(k_1,k_2)$ and $\theta(k_1,k_2)$ denotes the phase difference $\theta_F(k_1,k_2) - \theta_G(k_1,k_2)$. The ordinary correlation function $r_{fg}(n_1,n_2)$ is the 2D Inverse Discrete Fourier Transform (2D IDFT) of $R_{FG}(k_1,k_2)$ and given by

$$r_{fg}(n_1,n_2) = \frac{1}{N_1 N_2} \sum_{k_1,k_2} R_{FG}(n_1,n_2) W_{N_1}^{-k_1 n_1} W_{N_2}^{-k_2 n_2} \quad (4)$$

Where $\sum_{k_1,k_2}$ denotes $\sum_{k_1=-M_1}^{M_1} \sum_{k_2=-M_2}^{M_2}$.

On the other hand, the cross-phase spectrum (or normalized cross-spectrum) $\hat{R}_{FG}(k_1,k_2)$ is defined as

$$\hat{R}_{FG}(k_1,k_2) = \frac{F(k_1,k_2)\overline{G(k_1,k_2)}}{\left|F(k_1,k_2)\overline{G(k_1,k_2)}\right|} = e^{j\theta(k_1,k_2)} \quad (5)$$

The POC function $\hat{r}_{fg}(n_1,n_2)$ is 2D IDFT of $\hat{R}_{FG}(k_1,k_2)$ and given by

$$\hat{r}_{fg}(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1,k_2}\hat{R}_{FG}(n_1,n_2)W_{N_1}^{-k_1n_1}W_{N_2}^{-k_2n_2}$$

(6)

When $f(n_1,n_2)$ and $g(n_1,n_2)$ are the same image, i.e., $f(n_1,n_2) = g(n_1,n_2)$, the POC function will given by

$$\begin{aligned}\hat{r}_{fg}(n_1,n_2) &= \frac{1}{N_1N_2}\sum_{k_1,k_2}W_{N_1}^{-k_1n_1}W_{N_2}^{-k_2n_2}\\ &= \delta(n_1,n_2)\\ &= \begin{cases}1 & if n_1=n_2=0\\ 0 & otherwise,\end{cases}\end{aligned}$$

(7)

The above equation implies that the POC function between two identical images is the Kronecker's delta function $\delta(n_1,n_2)$.

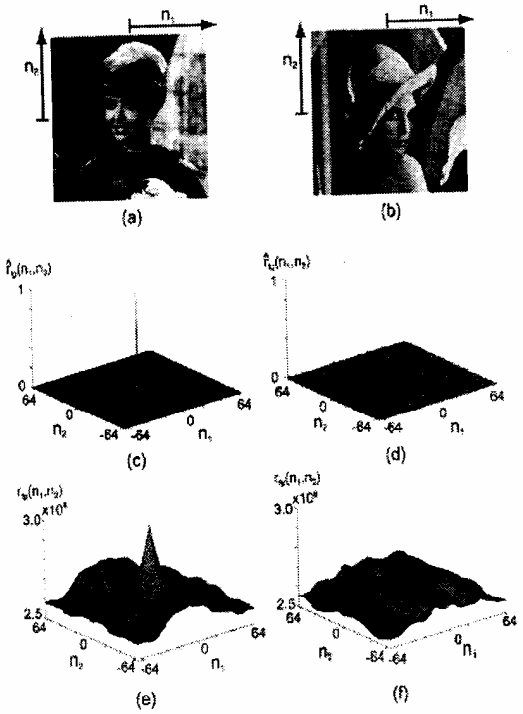

Figure 8: Example of the POC function $\hat{r}_{fg}(n_1,n_2)$ and the ordinary correlation function $r_{fg}(n_1,n_2)$: (a) image $f(n_1,n_2)$, (b) image $g(n_1,n_2)$, (c) POC function between the two identical images (the image $f(n_1,n_2)$), (d) POC function between $f(n_1,n_2)$ and $g(n_1,n_2)$, (e) ordinary correlation function between the two identical images (the image $f(n_1,n_2)$), and (f) ordinary correlation function between $f(n_1,n_2)$ and $g(n_1,n_2)$.

The most remarkable property of POC compared to the ordinary correlation is its accuracy in image matching. Figure 7 shows an example of image matching using POC function. When two images are similar, their POC function $\hat{r}(n_1,n_2)$ gives a distinct sharp peak. When two images are not similar, the peak drop significantly. Thus, the POC function exhibit much higher discrimination capability than the ordinary correlation function. The height of the peak can be used as a good similarity measure for image matching.

Parallel processing of the fingerprint matching is performed as follows: (i) remove the background from $f(n_1,n_2)$ and $g(n_1,n_2)$ on the server node #0, (ii) send $f(n_1,n_2)$ and $g(n_1,n_2)$ to every nodes, (iii) rotate $f(n_1,n_2)$ over independent angle ranges on the four nodes #0, #1, #2 and #3 in parallel, (iv) calculate the POC functions between the rotated images and $g(n_1,n_2)$, and compute similarity scores on the four nodes in parallel, (v) send the calculated similarity scores to the server node #0, and (vi) select the highest similarity score as an overall matching score on the server node #0.

procedure *Fingerprint Matching Algorithm Using POC Function for UCC*

Input:
$f(n_1,n_2)$: the registered fingerprint image,
$g(n_1,n_2)$: the fingerprint image to be verified;

Output:
matching score between $f(n_1,n_2)$ and $g(n_1,n_2)$;

1. **begin**
2. remove background from $f(n_1,n_2)$ and $g(n_1,n_2)$;
3. rotate $f(n_1,n_2)$ over the angular range -10° from 12° with an angle spacing 2° to generate a set of rotated images;
4. calculate the POC functions between the rotated images and $g(n_1,n_2)$, and compute the similarity scores, where the similarity score is defined as the sum of the highest two peaks of the POC function;
5. select the highest value of similarities as the matching score between $f(n_1,n_2)$ and $g(n_1,n_2)$
6. **end**

Figure 9: Fingerprint matching algorithm using POC function for UCC.

To implement the above algorithm, two libraries: FFTW [12] and ImageMagick [13] are used for Fast Fourier Transform (FFT) and image rotation, respectively. Table 3 shows the processing time of

fingerprint matching on UCC changing the number of available computing nodes. The processing time on a high-end workstation, Sun Blade 2000 (with 900MHz UltraSPARC III CPU and 1GB memory), is also shown in Table 3 for reference, where the same fingerprint matching algorithm is implemented using signal processing tools in MATLAB version 6. Using four computing nodes, we could achieve more than two times speed-up compared with a single-node implementation. Although further performance improvements based on detailed performance profiling are required, the result demonstrates a potential of the UCC platform for developing and prototyping practical application programs to be mapped on network-connected embedded CPUs.

Table 3: Processing time of fingerprint matching.

	Sun Blade 2000 (with MATLAB 6)	UCC		
Number of Processors	1	1	2	4
Averaged Processing Time [sec]	44.54	70.87	45.03	31.50

To make faster computation, we made register database image for each rotated fingerprint. We calculated 2D DFT for every fingerprint and then put it as database on every node. Figure 10 shows registered image on each node.

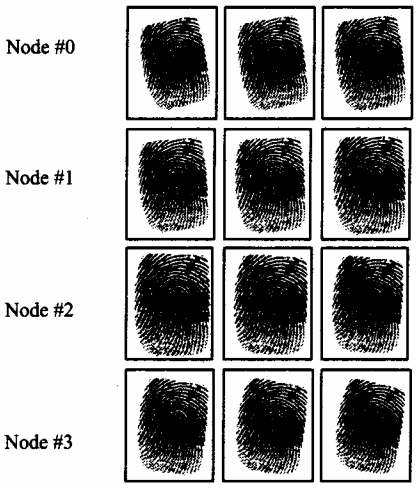

Figure 10: Registered database image of fingerprint on each node.

With database image, we can eliminate some processes according to algorithm. After that, we can achieve speed because in each node only calculate POC function and peak extraction as shown on figure 11.

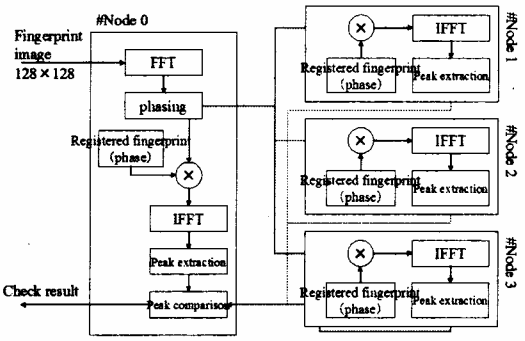

Figure 11: Simple algorithm for parallel fingerprint matching using POC.

Figure 12 shows the speedup factor for this experiment. By using 1 node computation we can achieve around 3 seconds to verify 1 fingerprint, by using 2 nodes we can achieve around 2 seconds and by 4 nodes we can achieve around 1 second.

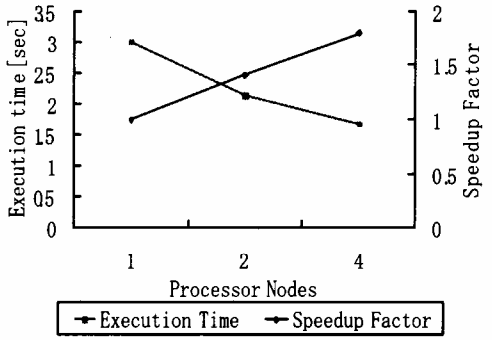

Figure 12: Speedup Factor for parallel fingerprint matching using POC.

5. Conclusion

In this paper, we presented a compact cluster computer with embedded CPUs, called "Ubiquitous Computing Cluster (UCC)", in order to provide a cost-effective prototyping environment for design and test of ubiquitous computing applications. UCC achieves extremely small size, low power consumption and low cost through the use of COTS embedded products, and can be easily introduced to educational programs and R&D projects in universities and companies. UCC is commercially available from Tokyo Electron Device Limited [14].

References

[1] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Communications of the ACM, vol. 36, No. 7, pp. 75-84, 1993.

[2] Ubiquitous Computing Cluster (UCC) http://www.aoki.ecei.tohoku.ac.jp/ucc/

[3] MPICH-A Portable Implementation of MPI http://www-unix.mcs.anl.gov/mpi/mpich/

[4] LAM/MPI Parallel Computing http://www.lam-mpi.org/

[5] PVM: Parallel Virtual Machine http://www.csm.ornl.gov/pvm/pvm_home.html

[6] K. Ito, H. Nakajima, K. Kobayashi, T. Aoki, and T. Higuchi, "A fingerprint matching algorithm using phase-only correlation," IEICE Trans. Fundamentals, vol. E87-A, no. 3, pp. 682-691, March 2004.

[7] Debian Distribution for SH3 and SH4 http://debian.dodes.org/index.en.html

[8] Pallas MPI Benchmark http://www.pallas.com/e/products/pmb/

[9] Q. Chen, M. Defrise, and F. Deconinck, "Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 16, no. 12, pp. 1156-1168, December 1994.

[10] T. Kenji, T. Aoki, Y. Sasaki, T. Higuchi, and K. Kobayashi, "High-accuracy subpixel image registration based on phase-only correlation," IEICE Trans. Fundamentals, vol. E86-A, no. 8, pp. 1925-1934, August 2003.

[11] T. Kenji, M. A. Muquit, T. Aoki, and T. Higuchi, "A sub-pixel correspondence search technique for computer vision applications," IEICE Trans. Fundamentals, vol. E87-A, no. 8, pp. 1913-1923, August 2004.

[12] FFTW (Fastest Fourier Transform in the West) http://www.fftw.org/

[13] ImageMagick http://www.imagemagick.org/

[14] Tokyo Electron Device Limited http://www.teldevice.co.jp/eng/index.html