# High-Performance Computing Service Over the Internet for Intraoperative Image Processing

Yasuhiro Kawasaki, Fumihiko Ino, *Member, IEEE*, Yasuharu Mizutani, Noriyuki Fujimoto, *Member, IEEE*, Toshihiko Sasama, Yoshinobu Sato, *Member, IEEE*, Nobuhiko Sugano, Shinichi Tamura, *Member, IEEE*, and Kenichi Hagihara

***Abstract*—This paper presents a framework for a cluster system that is suited for high-resolution image processing over the Internet during surgery. The system realizes high-performance computing (HPC) assisted surgery, which allows surgeons to utilize HPC resources remote from the operating room. One application available in the system is an intraoperative estimator for the range of motion (ROM) adjustment in total hip replacement (THR) surgery. In order to perform this computation-intensive estimation during surgery, we parallelize the ROM estimator on a cluster of 64 PCs, each with two CPUs. Acceleration techniques such as dynamic load balancing and data compression methods are incorporated into the system. The system also provides a remote-access service over the Internet with a secure execution environment. We applied the system to an actual THR surgery performed at Osaka University Hospital and confirmed that it realizes intraoperative ROM estimation without degrading the resolution of images and limiting the area for estimations.**

***Index Terms*—Cluster computing, computer-assisted surgery, high-performance computing, medical image processing, message-passing program, range of motion (ROM) estimation.**

## I. Introduction

With the rapid advances in information technology, three-dimensional (3-D) image processing plays an increasingly important role in surgery. 3-D image processing is a key technique for preoperative surgical planning and intraoperative guidance systems [1]–[3] that assist surgeons in developing surgical plans and performing surgery according to preoperative plans. For example, range of motion (ROM) estimators [4]–[6] support surgeons in selecting and aligning the optimal components of changeable artificial joint in total hip replacement (THR) surgery. However, such image processing requires a large amount of computation because recent X-ray computed tomography (CT) scans produce high-resolution 3-D images. Therefore, high-performance computing (HPC) approaches are necessary for intraoperative image processing, which requires real-time processing.

One emerging platform in HPC is the cluster system [7], [8], or a collection of interconnected computers. Although, as compared to shared-memory systems, cluster systems are loosely coupled systems, rapid advances in hardware technology have made such systems attractive because processors become much faster and interconnections offer higher bandwidth with lower latency. Some cluster systems consist of off-the-shelf hardware, which allows the utilization of HPC resources at a lower cost as compared to vendors' supercomputer systems. In addition, because cluster systems enable high-speed and large-scale data processing by multiprocessor and distributed memory architecture, developing medical image processing systems on clusters realizes *the HPC-assisted surgery* with both real-time and high-resolution image processing at a low cost. Moreover, cluster systems have the flexibility to extend their performance for the increasing computational cost of medical image processing.

In order to realize this novel type of surgery, cluster-enabled medical systems need to possess several extra facilities in addition to those on single systems. These facilities can be classified into two groups based on whether they are intended for preoperative or intraoperative assistances. For preoperative supports, cluster-enabled medical systems require 1) a secure execution environment to keep patients' confidential information and 2) necessary data distribution for data-intensive applications. For intraoperative assistances, in addition to the above facilities 1) and 2), also 3) parallel processing is needed for real-time processing of compute-intensive applications.

To realize the HPC-assisted surgery and share this novel procedure with several hospitals, we developed a testbed, named Medical Image Processing Cluster (MIP-Cluster), composed of 64 off-the-shelf symmetric multiprocessor (SMP) PCs with two CPUs per node. MIP-Cluster allows surgeons to utilize remote HPC resources via the Internet and resolves facility issues 1)–3) outlined above in the following medical applications: an accurate estimation for the ROM adjustment in THR surgery, a rigid/nonrigid registration [9], [10] for surgical planning, and real-time volume rendering [11] for the 3-D visualization of invisible parts inside patients.

In the present paper, we report our work as follows. Section II reviews the related work concerning the use of HPC during surgery. Section III describes the design and implementation of

Manuscript received June 7, 2003; revised October 23, 2003. This work was supported in part by JSPS Research for the Future Program JSPS-RFTF99I00903, by JSPS Grants-in-Aid for Scientific Research (C)(2)(14580374) and Young Researchers (B)(15700030), and by Network Development Laboratories, NEC.

Y. Kawasaki, F. Ino, Y. Mizutani, N. Fujimoto, and K. Hagihara are with the Department of Computer Science, Graduate School of Information Science and Technology, Osaka University, Osaka 560-8531, Japan (e-mail: ino@ist.osaka-u.ac.jp).

T. Sasama is with the Department of Information and Knowledge Engineering, Graduate School of Engineering, Tottori University, Tottori 680-8552, Japan.

Y. Sato and S. Tamura are with the Department of Medical Robotics and Image Sciences, Graduate School of Medicine, Osaka University, Osaka 565-0871, Japan.

N. Sugano is with the Department of Orthopaedic Surgery, Graduate School of Medicine, Osaka University, Osaka 565-0871, Japan.

Digital Object Identifier 10.1109/TITB.2004.824740

MIP-Cluster. Section IV presents a ROM estimator as an example of medical applications available on MIP-Cluster. Section V shows experimental results obtained at Osaka University Hospital using our systems. And finally, Section VI summarizes the main conclusions.

## II. Related Work

Several recent works [12]–[17] incorporated HPC approaches into operating room. In [12], Warfield *et al.* describe a real-time intraoperative image segmentation algorithm for image-guided surgery. The algorithm classifies the brain from intraoperatively scanned volumetric data and shows surface rendering of the classified brain with transparently rendered skin surface. It dynamically balances processor workloads by using a master/slave (M/S) decomposition [18] with the number of tasks greatly exceeding the number of CPUs. Their parallel implementation on a 20 CPU Sun Ultra HPC 6000 classifies a $256 \times 256 \times 60$ voxel image in approximately 20 s, which is rapid enough to realize intraoperative assistances. By contrast, sequential implementation requests more than 2 min. They also present an intersubject affine registration algorithm [13] that is capable of aligning an anatomical atlas to a subject in less than 45 s on a 12 CPU SunFire 6800. In [14], a nonrigid registration algorithm that simulates the biomechanical properties of the brain and its deformations during surgery is described. Its implementation on the Sun Ultra HPC 6000 accelerates the time of this estimation to less than 10 s. Although this algorithm uses a static load balancing strategy with equally sized domains being assigned to each CPU, the load imbalance issue remains unaddressed. Nevertheless, their HPC approach has successfully assisted surgeons during surgery.

Rohlfing *et al.* [15] demonstrate a parallel implementation of a fast nonrigid registration algorithm for intraoperative imaging such as a brain deformation during cranial surgery. On a 128 CPU SGI Origin 3800, their parallel implementation reduces the execution time from hours to minutes with the maximal speedup of a factor of 44.

In addition to these vendors' shared-memory multiprocessors, clusters of off-the-shelf PCs also exhibit HPC benefits. In [16], Ourselin *et al.* introduce parallel affine registration, yielding a speedup of 6 on 10 CPUs. Liao *et al.* [17] develop a high-resolution stereoscopic display driven by a parallel rendering system on a cluster. Warfield *et al.* [14] also demonstrate the greatest performance with a cluster of fast CPUs, exceeding that of a shared-memory multiprocessors which utilizes higher bandwidth lower latency backplane but with slower CPUs.

Another emerging HPC platform is the Grid [19] whose concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. Hastings *et al.* [20] show a toolkit that allows rapid and efficient development of biomedical image analysis applications in a distributed environment. They are currently developing interfaces to the Globus toolkit [21], which provides standard Grid mechanisms such as resource, security, and file management. Mizuno-Matsumoto *et al.* [22] develop a Grid-enabled system for analyzing the functional state of the brain. Their system running on a 16 CPU Exemplar V2200/N reduces the analysis time from 285 to 40 s. Norton *et al.* [23] introduce a method that enables volume visualization of data from the Grid. In order to overcome the problem of low bandwidth, their method limits the requirement for data transmission by combining wavelet compression with visibility detection. This enables responsible visualization with a delay of 2 s or less between frames.

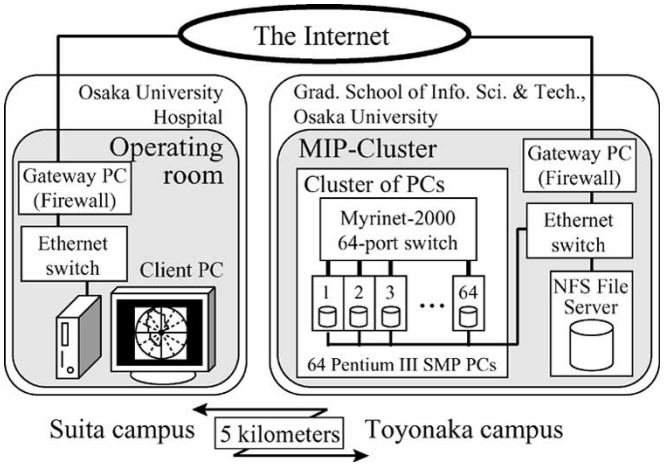


Fig. 1. Overview of MIP-Cluster. Each node in the cluster has two Pentium III 1-GHz processors, 2 GB of registered main memory with ECC, and 40 GB of local disk storage. Local storage is used for clinical data while NFS mounted storage is used for application development.

Although many researchers have realized medical image processing on shared-memory multiprocessors as well as distributed-memory multiprocessors, as summarized above, to the best of our knowledge, work on developing a framework to utilize HPC resources remote from the operating room during surgery is lacking. Only a few publications [24]–[26] describe the intraoperative use of an HPC resource distant from the operating room where surgery is performed. However, the access to the HPC resource from the operating room in these works is apparently restricted via a local-area network (LAN), because security issues are unaddressed.

The present paper aims to tackle the following issues related to HPC resources remote from the operating room for its sharing by several hospitals: a) physical proximity, b) security environment, c) network bandwidth, and d) network latency between the operating room and the remote HPC system, and e) scheduling of high-priority jobs on the shared system.

## III. MIP-Cluster: A Medical Image Processing System for Remote Parallel Processing

The design and implementation of MIP-Cluster that addresses issues a)–e) by hardware and software approaches are described as follows.

### A. Hardware Architecture of MIP-Cluster

Fig. 1 illustrates hardware overview of the MIP-Cluster system, including a client PC in the operating room at Osaka University Hospital, located approximately 5 km from the MIP-Cluster. The Internet connection from the client to the MIP-Cluster solves a) physical proximity issue because it provides HPC resources for any hospital at any time, such

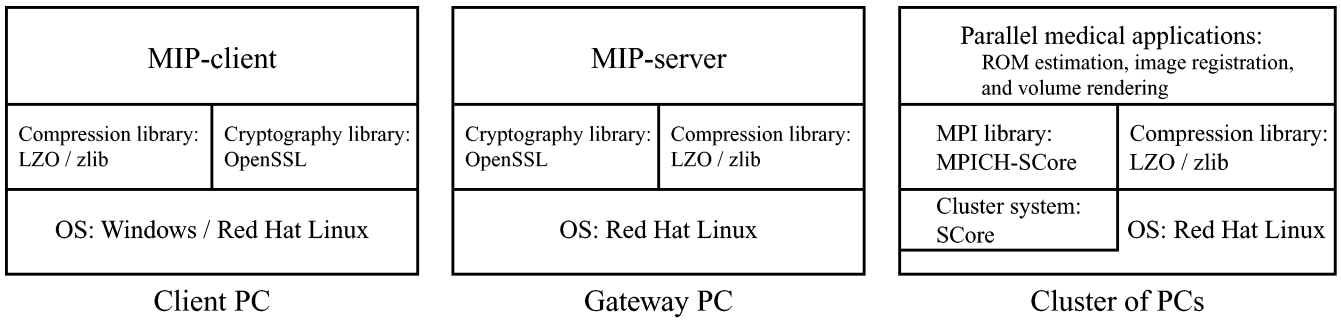


Fig. 2. Software architecture of MIP-Cluster. MIP-client and MIP-server provide an automation framework for remote parallel processing with a secure execution environment and a high-speed data transmission mechanism.

as with ubiquitous computing [27]. Ubiquitous computing is a key factor for preparing emergency surgeries. However, because the Internet is a public network, this connection raises b) the security issue. To address this issue in terms of hardware architecture, the system is divided into two components: a cluster of node PCs that executes parallel medical applications and a gateway PC that connects the cluster with the Internet. The gateway also serves as a network firewall that protects the cluster from malicious accesses and provides the cluster's HPC resources for remote client PCs in operating rooms. Thus, any malicious access to the system is limited to the gateway, and this makes the system secure and frees administrators from managing many node PCs.

The cluster has 64 SMP nodes in total and each node consists of off-the-shelf components such as two Pentium III 1-GHz processors, 2 GB of registered main memory with error-correction code (ECC), and 40 GB of local disk storage. Because the packets for node management such as network file system (NFS) and network information service (NIS) [28] can abandon the performance of parallel medical applications due to network congestion, the cluster is also connected to two networks, a Myrinet network [29] for parallel processing and a Fast Ethernet network for node management, yielding full-duplex bandwidth of 2 Gb/s and 100 Mb/s with the minimum latency of 10 $\mu$s and 50 $\mu$s, respectively.

Two types of storage are available in the cluster, a local storage for clinical data and a 144-GB NFS mounted storage for application development. Because all nodes share NFS mounted storage, local storage offers better read/write performance if every node simultaneously performs disk accesses during parallel execution. For example, every node can independently read preoperative data from its local storage, because this data can be distributed to nodes before surgery. On the other hand, NFS mounted storage suits for application development, where program code is frequently modified, compiled, and executed for debugging.

It should be noted that the cluster system must be tested for endurance in order to confirm the reliability of hardware components. Because parallel applications generally perform calculation and communication, memory and network components must behave in a correct manner at least during surgery. We tested them by using a program that performs read/write accesses for every address of main memory and transmits data between every pair of processors. Furthermore, we confirmed cooling facility be acceptable for heavy computation and communication running for 24 h.

In addition to reliability, the cluster system requires fault-tolerance capability that enables applications to continue to process without losing the time benefits of HPC. Because off-the-shelf components generally lack this capability, a software approach is required to enable this. One simple method is to duplicate the entire cluster system and automatically switch them with a timeout mechanism.

### *B. Software Architecture of MIP-Cluster*

Fig. 2 presents the software architecture of MIP-Cluster. Each node in MIP-Cluster runs on free software: the Linux operating system and the SCore cluster system software [30]. The licensing scheme of free software suits for the cluster system, because the cost for employed software should be independent of the number of processors in order to construct a larger system at a lower cost.

SCore provides a multiuser environment with fault tolerance based on checkpointing. This environment addresses issue e), because it uses a gang scheduling mechanism [31] with priority queues, which enables time-shared scheduling and provides an interactive parallel programming environment. In an emergency case, surgeons can submit parallel jobs with the highest priority, and these jobs are always executed immediately. If the number of processors is insufficient for all the queued jobs, then any job with a lower priority is suspended until the highest priority job terminates. If two or more jobs are submitted with the highest priority, these jobs are executed simultaneously.

Parallel medical applications are implemented using the C++ language and message passing interface (MPI) routines [32]. MPI is a widely used standard for writing message-passing programs and enables the development of highly portable and efficient parallel applications on distributed memory multiprocessors. We are currently using the MPICH-SCore library [33], a fast implementation of MPI, provided by SCore. It should be noted here that MPI assumes explicit parallelism coded by developers. Therefore, developers take the responsibility for identifying the bottleneck code of sequential applications and determining which code should be parallelized, according to the performance analysis of the applications.

The details of MIP-client and MIP-server in Fig. 2 are described in Section III-C.

### C. Framework for Remote Parallel Processing

MIP-client and MIP-server provide an automation framework for remote parallel processing with a secure execution environment and a high-speed data transmission mechanism. This framework employs a client–server paradigm and has the following two facilities in order to address issues b)–d) in terms of software architecture.

- *Secure execution environment*: The framework has to realize an environment that 1) protects patients' data transmitted over the Internet and 2) prevents any unauthorized/unauthenticated/uncertificated access to the cluster system. Requirement 1) is ethically obligatory and requirement 2) is required to prevent operating hijacking based on spoofing. These requirements are implemented by using the OpenSSL public key cryptography library [34], which encrypts transmitted data, authenticates users, and prevents tapping and spoofing. The authentication is achieved by giving the password of an account assigned for each surgeon. In our experiments, surgeons in the same operating room share one account authenticated only once immediately before surgery. The certification is based on X.509 certificates [35], which can be imported from public certificate authorities.
- *High-speed data transmission*: The framework has to provide a mechanism that efficiently transmits data over the Internet. This efficient mechanism is necessary for real-time remote processing, especially when a high-bandwidth network with a low latency is unavailable. For example, a $512 \times 512 \times 512$ voxel image in 16-b color becomes 256 MB in size, and thereby transmitting this raw image over a low-bandwidth network with a high latency can nullify the time benefits of HPC. Our framework satisfies this requirement by 1) compressing data as well as 2) avoiding retransmission of the same data. It compresses input/output data by using one of the following two libraries: LZO [36] for the high-bandwidth network and zlib [37] for the low-bandwidth network. Both are lossless data compression libraries, with LZO focusing on real-time compression and zlib specializing in general-purpose and high-rate compression. It should be noted here that the appropriate compression library is currently determined by application developers according to the network bandwidth between the client and the server. Retransmission mechanism is managed by comparing file information including file name, file size, and last modified date. Therefore, when data are modified at any location, the system detects the mismatched file information, and then decides retransmission.

In addition to the above facilities, our framework has the following useful facilities.

- *Progress report*: To inform distantly connected surgeons of the progress in parallel execution, the system reports the progress every half second and shows the expected completion time of the submitted job. This facility is useful when the submitted job is unexpectedly suspended due to other high-priority jobs.

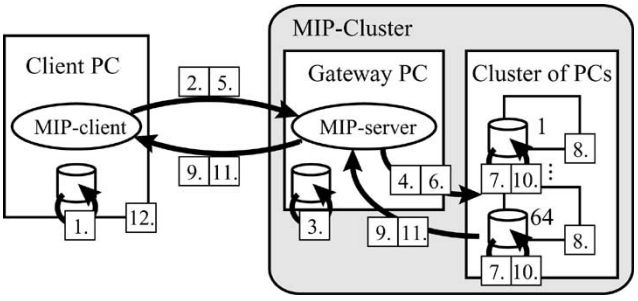


Fig. 3. Encapsulated procedure for remote parallel processing. The procedure consists of four major stages: (1.–4.) input-data transmission, (5.–7.) job submission, (8.–9.) parallel execution, and (10.–12.) output-data transmission. See text for details.

- *Automatic parallel execution*: The framework should enable users to submit their jobs in the same way as they do in sequential systems. This is necessary to prevent human error originated from the procedure for remote parallel execution. Thus, the procedure must be simplified in a mission-critical system, so that our framework hides it from the users. Fig. 3 shows the encapsulated procedure, which consists of four major stages as follows.
  1) Input-data transmission: The client (1.) compresses input data and (2.) sends it to the gateway. The gateway (3.) registers the received data and (4.) distributes it to the nodes in the cluster.
  2) Job submission: The client (5.) requests the gateway to execute a parallel job over the Internet. When the gateway receives a request from the client, it (6.) submits the requested job to SCore. Each node (7.) decompresses their input data.
  3) Parallel execution: Each node (8.) executes the submitted job in parallel and (9.) periodically reports the progress of execution to the client.
  4) Output-data transmission: Each node (10.) compresses output data and (11.) transmits it back to the client. The client (12.) decompresses the received data and obtains the result for the submitted job.

## IV. Parallelizing Range of Motion (ROM) Estimator

This section describes the background of ROM estimation and our strategy for parallelizing a ROM estimator [6].

### A. Background

ROM estimation, which computes the safe ROM, aims at assisting surgeons in selecting and aligning the optimal components of changeable artificial joint: the cup, head, neck, and stem components (ANCA-FIT, Cremascoli, Milan, Italy) as illustrated in Fig. 4(a). This assistance is important for both the surgeon and patient, because either inappropriate or improperly positioned components increase the risk of clinical problems such as dislocation, wear, and loosening [5], [6].

Existing ROM estimators [4]–[6] are useful in developing preoperative surgical plans, which determine the optimal components of an artificial joint, its implant position, and orientation. However, the preoperative plans may need to be altered

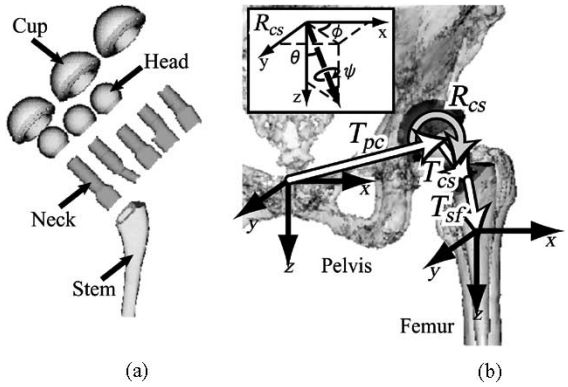


Fig. 4. Components of artificial joint and representation of hip joint motion. (a) Changeable cup, head, neck, and stem components and (b) hip joint motion defined over the components.

in view of unpredicted conditions revealed during surgery. For example, if the bone tissue around the preoperatively planned position is proven to be fragile, the surgeon has to realign the cup components in a stable position and to reselect the optimal combination of head and neck components for the new position. Therefore, in order to address this problem, a fast estimator based on the actual positions and orientations of the implanted cup and stem components is required.

The accuracy of intraoperative estimation depends on the following three factors.

1) *3-D surface models of surgical objects constructed from preoperative CT images:* In our navigation system [6], the CT images of the pelvis and femur are obtained by a helical scanner (HiSpeed Advantage, GE Medical Systems, Milwaukee, WI, USA). The image size is $512 \times 512$ pixels under a field of view (FOV) of 42 cm with 3-mm slice thickness and 1-mm reconstruction pitch. The surface models are reconstructed from these preoperative images by using the Marching Cubes algorithm [38] with a threshold value of 150 Hounsfield Units (HU).
2) *Tracking and measurement of the patient and operating tools during surgery:* The positions of the pelvis and the femur are tracked by 24 light-emitting diode (LED) markers attached to the bone using an optical 3-D position sensor (OPTOTRAK 3020 [39], Northern Digital Inc., Waterloo, ON, Canada). The placement of the cup and the stem components is guided and measured using cup and stem localizers with LED markers.
3) *Registration [40] of the model and the patient:* Surface-based registration is performed by using the iterative closest point (ICP) algorithm [41] with the least square method. 30 points are sampled from 1) a periarticular area that is usually exposed and is easily accessed through a posterolateral approach and 2) a peripheral area of 1) where some surface points can be digitized without additional skin incision.

We expect these parameters are acceptable for ROM estimations, because our prior work confirms that 1) two clinical cases of ROM estimations are consistent with actually measured motions [6] and 2) the error of registration is acceptable for clinical use [42]. The accuracy of registration is 1.2 mm and 0.9° of bias with 0.7 mm and 0.3° of root mean square (rms) in the pelvis, and 1.4 mm and 0.6° of bias with 1.3 mm and 0.3° of rms in the femur. See [6], [42] for more information of the accuracy of registration and cases of ROM estimations.

### B. ROM Estimation

Before describing the details of ROM estimation, it is necessary to briefly introduce a representation of hip joint motion presented in [6] (see Fig. 4(b)). Given the pelvis coordinate system (pelvis-CS) and the femur coordinate system (femur-CS), the hip joint motion is represented by the transformation from the pelvis-CS to the femur-CS $M_{pf}$, given by

$$M_{pf} = T_{pc} T_{cs} R_{cs} T_{sf} \tag{1}$$

where $T_{pc}$ is a $4 \times 4$ transformation matrix representing the orientation of the cup in the pelvis-CS, $T_{cs}$ is a fixed transformation matrix determined by the selected head and neck components, $R_{cs}$ is a variable transformation matrix constrained to the rotational motion, and $T_{sf}$ is a transformation matrix representing the reverse orientation of the stem in the femur-CS. Both $T_{pc}$ and $T_{sf}$ are determined by one of the following two methods. For preoperative assistances, the surgeon determines them by visualization and experience. For intraoperative assistances, optical position sensors provide the actual values of $T_{pc}$ and $T_{sf}$ by measuring implanted components.

Given $T_{pc}$, $T_{cs}$, and $T_{sf}$, the safe ROM is defined as a set of rotation transformation matrix $R_{cs}$ such that $R_{cs}$ avoids any implant–implant, bone–implant, and bone–bone impingements. Because $R_{cs}$ is defined as a 3-D rotation matrix, ROM estimation is a computing-intensive application. Therefore, a real-time estimator is necessary to reduce patient time in surgery in order to, for example, decrease the physical stress of patients and the need for blood transfusion. Furthermore, to obtain precise ROMs, an intraoperative estimation based on measured $T_{pc}$ and $T_{sf}$ is required.

In the following, we represent $R_{cs}$ by the Euler angles, $(\phi, \theta, \psi)$ $(0 \leq \phi < 360, 0 \leq \theta < 180, -180 \leq \psi < 180)$, where $\phi$, $\theta$, and $\psi$ represent yaw, pitch, and role angles, respectively (see Fig. 4(b)).

### C. Parallel Implementation

We now present how we parallelize the ROM estimator. The calculation in ROM estimation has two features as follows.

F1: *Coarse-grain parallelism with many parallel tasks.*
F2: *Nondeterministic time complexity.*

Feature F1 determines what should be calculated in parallel with respect to performance. This feature enables processors to investigate independently each rotation $(\phi, \theta, \psi)$ on whether it causes impingements. It leads to a reasonable acceleration, since there are a large number of rotations to be investigated. It should be noted that fine-grain parallelism should be exploited if the number of rotations is insufficient for that of processors. Thus, the balance between the number of parallel tasks and that of processors determines the choice of parallelism.

Feature F2 requires a dynamic load-balancing mechanism. If the rotations are equally assigned to processors in a static

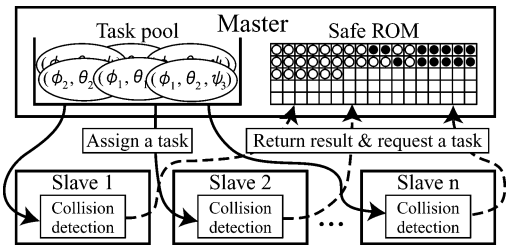


Fig. 5. M/S dynamic load balancing for ROM estimation.

manner, we fail to obtain a reasonable speedup due to this feature. In addition, it is not easy to statically predict the workload required for each rotation, because the workload depends on how complicated the objects intersect each other and this varies dynamically for individual rotations. Therefore, to realize dynamic load balancing, we employ the M/S paradigm [18], where the master manages the safe ROM and assigns tasks to the idle slaves (see Fig. 5). Here, a task corresponds to the collision detection for a set of rotations and the grain size of each task corresponds to the number of rotations in the set.

We implement our ROM estimator using the C++ language and the MPICH-SCore library. For each of collision detections, we use the V-COLLIDE library [43], [44], which rapidly detects precise collisions at $O(n+m)$ time, where $n$ and $m$ denote the number of polygonal objects and that of polygonal objects very close to each other, respectively. Because V-COLLIDE detects all of the collided polygons, a simple modification is made on this library so that it returns immediately upon detecting one collided polygon.

## V. Experimental Results

This section presents some experimental results obtained at Osaka University Hospital, consisting of 1) the speedup of our ROM estimator with different task grain sizes and 2) the turnaround time of remote parallel processing with different compression algorithms. We also discuss the possibility of the HPC-assisted surgery.

### A. Practical Use in THR Surgery

We applied our ROM estimator to a THR surgery performed at Osaka University Hospital. Fig. 6 shows the schema for this surgery assisted by both preoperative and intraoperative ROM estimations.

Before the surgery, we obtained CT images of the pelvis and the femur and constructed their 3-D surface models according to the method mentioned in Section IV-A. Pelvis and femur data were recorded in Visualization Toolkit (VTK) binary format [45] and were composed of 116 270 and 30 821 polygons (3.4 and 0.9 MB in size), respectively. Then, by using our automation framework, these data were compressed and transmitted to MIP-Cluster. Two types of estimation were subsequently performed: preoperative ROM estimation and preoperative limb length estimation [6] to minimize a limb length discrepancy between left and right legs. We repeated these preoperative estimations ten times with different positions, orientations, and combinations of artificial joints. Based on these estimation results, the surgeon developed a preoperative surgical plan establishing the optimal combination of the cup, head, neck, and stem components, and the implant position and orientation of the cup and stem components.

During the surgery, the surgeon first removed diseased bone tissue and temporarily implanted the cup and stem components according to the preoperative plan. As mentioned in Section IV-A, the placement of these components was guided and measured by localizers with LED markers. After this placement, we transmitted measured $T_{pc}$ and $T_{sf}$ (200 B in size) to MIP-Cluster, and then performed intraoperative limb length and ROM estimations to check whether the actual placed cup and stem components could cause serious clinical problems with the preoperatively selected head and neck components. These intraoperative estimations were repeated three times with various head and neck components in order to derive the final position at which the optimal components are implanted.

Fig. 7 shows the obtained safe ROM, where $\psi = -40, 0$, and $40$. On a sequential system, the number of rotations would have to be reduced to ensure real-time processing. For example, in order to limit the area for estimations, only $\phi$ and $\theta$ at $\psi = 0$ can be varied in order to obtain a two-dimensional safe ROM as shown in Fig. 7(a). Alternatively, the resolution of images has to be degraded in order to reduce the number of polygons. By contrast, MIP-Cluster enabled us to vary all of the three angles while still maintain real-time processing, the entire area for estimations, and the resolution of images, and thereby rendered a 3-D safe ROM as shown in Fig. 7(b).

The safe ROM is currently estimated based on implant–implant, bone–implant, and bone–bone impingements. Other impingements caused by muscles and soft tissues are not taken into consideration. Therefore, it is impossible to reproduce all simulated safe ROM intraoperatively. Nevertheless, our ROM estimator provides an upper bound for the safe ROM, which is useful for the surgeon to determine whether additional osteotomy on the pelvic bone is required. For example, if an upper bound is obviously small for a yaw angle, the surgeon can accordingly recognize that additional osteotomy is required to remove unwanted impingements. In this case, the surface model can be intraoperatively reconstructed from the bone shape resurfaced by the additional osteotomy based on a set of 3-D points obtained by digitizing the bone surface using the OPTORAK sensor [46].

### B. Speedup of Parallel ROM Estimator

To evaluate our parallel ROM estimator, we measured the speedup of the estimator with different task grain sizes. The results in this section concern only the parallelized part and Section V-C presents the entire turnaround time, including the execution time for data transmission. We varied the values of three angles $\phi$, $\theta$, and $\psi$ by $1^\circ$, $1^\circ$, and $20^\circ$, respectively, i.e., we investigated $360 \times 180 \times 18 = 1\,166\,400$ rotations.

In the following, let $s$ be the grain size of each task. Fig. 8 shows the execution time on 128 CPUs, where $100 \leq s \leq 9000$. On Fast Ethernet, we obtained the minimal speedup of $15.5$ when $s = 100$, and the maximal speedup reached a factor of $62.8$ when $s = 4000$. This maximal speedup is close to the result on Myrinet, which yields the minimal and maximal speedups of

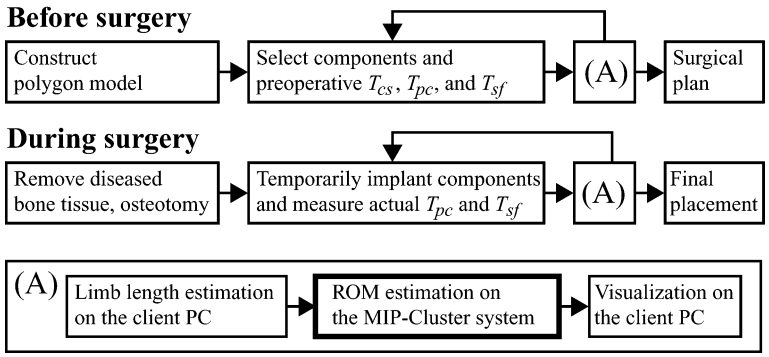


Fig. 6. Schema for total hip replacement surgery.

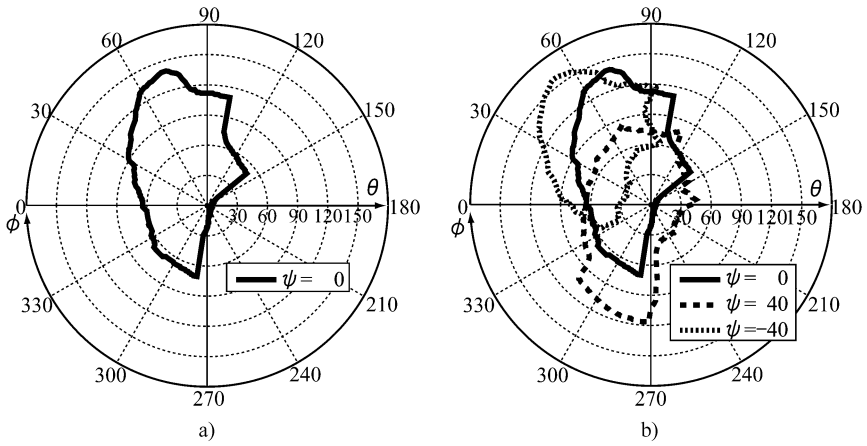


Fig. 7. ROM estimation results for (a) $\psi = 0$ and (b) $\psi = -40, 0,$ and $40$. An enclosed area corresponds to a slice of a safe ROM. $\phi$, $\theta$, and $\psi$ represent yaw, pitch, and role angles, respectively.

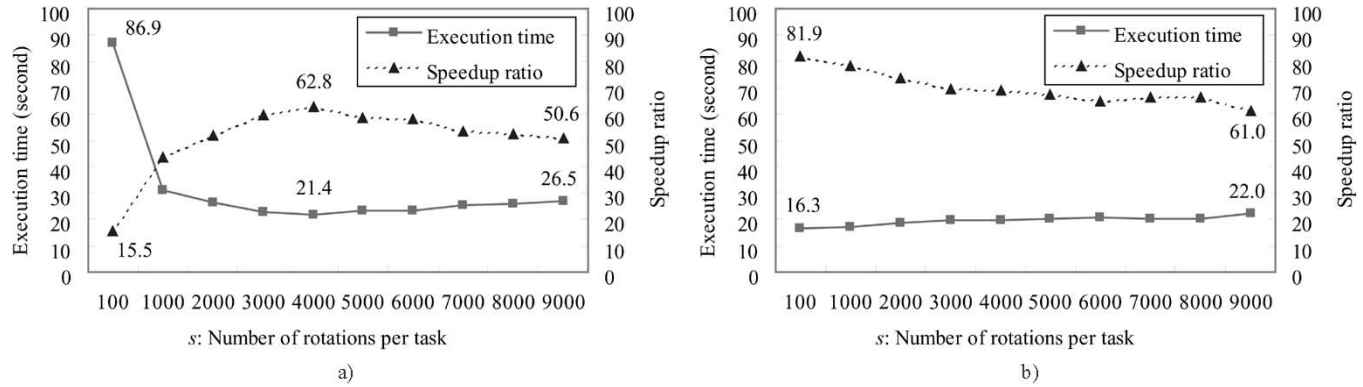


Fig. 8. Execution time on 128 CPUs with different grain size. Measured time on (a) Fast Ethernet and (b) Myrinet.

61.0 and 81.9, respectively. Thus, once the appropriate value for $s$ is selected, an intraoperative estimation can be performed on Fast Ethernet, a widely used LAN.

Fig. 9 shows the distribution maps of the execution time on Fast Ethernet, where $\psi = 0$ and $\psi = 40$. Based on these maps, the reason why the minimal speedup on Fast Ethernet becomes extremely small can be elucidated. In most cases, the slaves take below 2 ms to decide whether a rotation causes collisions. Therefore, the grain size of each task is too small for this high latency network. Under such circumstance, the master suffers from network congestion and the slaves easily become idle states. Actually, the total time of send routine (MPI _Send) called by the master accounts for 90% of the execution time (86.9 s) and the average total time of receive routine (MPI _Recv) called by the slaves accounts for 80%. Thus, the master becomes busy in assigning tasks while the slaves stay idle until they receive a task.

We also measured the execution time of a static implementation that statically assigns an equal number of rotations in a block distribution manner. The execution times on Fast Ethernet and Myrinet were 41.6 and 26.6 s, respectively. It should be noted here that the M/S implementation loses its load balancing facility with the increase of $s$. That is, the behavior of the M/S implementation becomes similar to that of the static implementation. However, there exists a gap between their execution time: $41.6 - 26.5 = 15.1$ s on Fast Ethernet and $26.6 - 22.0 = 4.6$ s

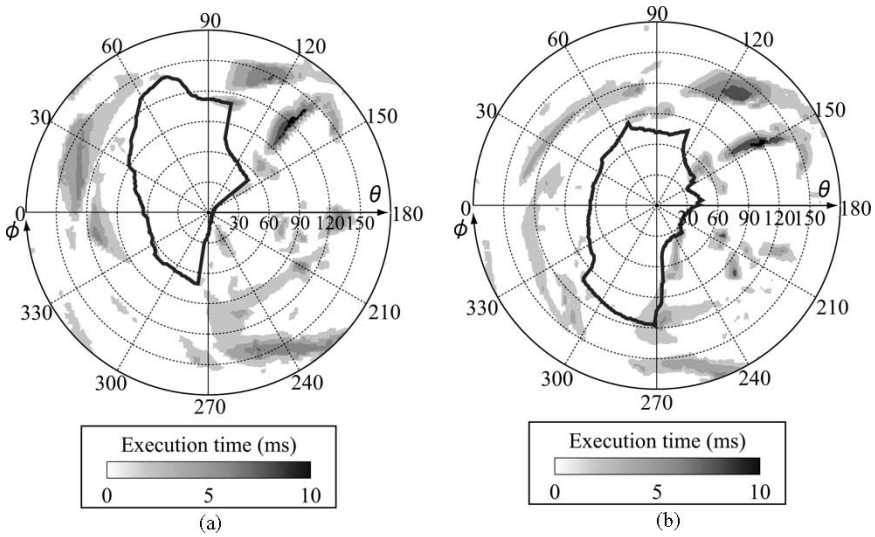


Fig. 9. Distribution maps of detection time on Fast Ethernet, where (a) $\psi = 0$ and (b) $\psi = 40$.

TABLE I
TURNAROUND TIME OF SEQUENTIAL AND REMOTE PARALLEL PROCESSING. FE AND MY REPRESENT FAST ETHERNET AND MYRINET, RESPECTIVELY

| | Breakdown of turnaround time | Execution time in second | | | |
|---|---|---|---|---|---|
| | | 1CPU | 128CPU | | |
| | | | Compression algorithm | | |
| | | | No<br>FE / MY | LZO<br>FE / MY | zlib<br>FE / MY |
| Before surgery | $t_1$: Compress polygon data (1.) | — | — | 0.2 | 1.8 |
| | $t_2$: Transmit polygon data (2.-3.) | — | 3.4 | 2.1 | 1.5 |
| | $t_3$: Distribute polygon data (4.) | — | 10.1 / 2.4 | 6.1 / 2.0 | 4.2 / 1.9 |
| | $t_b$: Total (preoperative) | — | 13.5 / 5.8 | 8.4 / 4.3 | 7.5 / 5.2 |
| During surgery | $t_4$: Compress $T_{pc}$ and $T_{sf}$ (1.) | — | — | 0.1 | 0.1 |
| | $t_5$: Transmit and distribute $T_{pc}$ and $T_{sf}$ (2.-4.) | — | 0.5 / 0.5 | 0.5 / 0.5 | 0.5 / 0.5 |
| | $t_6$: Invoke parallel processes (5.-6.) | — | 4.8 / 4.8 | | |
| | $t_7$: Decompress all input data (7.) | — | — | 0.4 | 0.4 |
| | $t_8$: ROM simulation (8.-9.) | 1354 | 21.4 / 16.3 | | |
| | $t_9$: Compress the safe ROM (10.) | — | — | 0.4 | 1.2 |
| | $t_{10}$: Transmit the safe ROM (11.) | — | 13.0 | 0.5 | 0.1 |
| | $t_{11}$: Decompress the safe ROM (12.) | — | — | 0.4 | 0.4 |
| | $t_d$: Total (intraoperative) | 1354 | 39.7 / 34.6 | 28.5 / 23.4 | 28.9 / 23.8 |

on Myrinet, where $s = 9185$ ($\equiv \lfloor 1\,166\,400/127 \rfloor$). This gap is also caused by the network congestion mentioned above. While the static implementation gathers the distributed safe ROMs all at once, the M/S implementation allows every slave to return their safe ROMs as soon as they complete their assigned task. Therefore, comparing to the M/S implementation, the static implementation can easily cause network congestion when gathering the safe ROM.

### C. Turnaround Time of Remote Parallel Processing

Table I lists the turnaround times of ROM estimations when processing in sequential and remote parallel. To clarify the contribution of data compression, we compared three implementations: LZO, zlib, and no compression.

Although our system required additional times for remote parallel processing, times $t_1, t_2, \ldots, t_7$ and $t_9$, $t_{10}$, and $t_{11}$, it reduced the turnaround time from 1354 s on a single system to the minimum of 23.4 s on Myrinet and to that of 28.5 s on Fast Ethernet. These turnaround times of less than one half of a minute are speedy enough for intraoperative processing.

TABLE II
RESULTS ON DATA COMPRESSION. NUMBERS IN BRACKETS REPRESENT COMPRESSION RATE

| Data | Size in KB | | |
|---|---|---|---|
| | No | LZO | zlib |
| Pelvis polygon | 3 398 | 2 072 (39%) | 1 547 (54%) |
| Femur polygon | 910 | 591 (35%) | 453 (50%) |
| Safe ROM | 22 781 | 379 (98%) | 146 (99%) |

It was also evident that both data-compression algorithms successfully reduced the turnaround time by approximately 33% on Fast Ethernet and 37% on Myrinet. This improvement is achieved by the high compressibility of the safe ROM, because the safe ROMs are represented as a sequence of Boolean values and the same value repeatedly appears in the sequence. Therefore, as shown in Table II, both algorithms yield high compression rate, 99% by zlib and 98% by LZO, and its data size is accordingly reduced from 22.8 to 500 kB.

A significant gap between LZO and zlib appeared only at time $t_b$. That is, as compared to LZO, zlib takes more time to compress data but generates 25% smaller data. Therefore, zlib takes an advantage over LZO when transmitting data via a low-bandwidth network and when broadcasting data to the nodes

in the cluster. By contrast, LZO takes an advantage over zlib when transmitting data via a high-bandwidth network and when compressing data that achieve high-rate compression such as the safe ROM.

### D. Discussions on the HPC-Assisted Surgery

Our results indicate that HPC-assisted surgery can be successfully performed under the following conditions.

- *On parallel method*: The M/S paradigm allows easy utilization of sequential implementations, because it exploits coarse-grain parallelism. Therefore, this paradigm enables parallelizing many existing applications with ease. Furthermore, because the M/S paradigm realizes a dynamic load balancing mechanism, it provides an adaptive performance on heterogeneous clusters. In addition, incorporating a timeout mechanism into M/S applications realizes fault tolerance capability that tolerates hardware/software faults on the slaves. The shared-memory paradigm also allows parallelizing sequential applications easily. For example, OpenMP [47] automatically parallelizes sequential programs according to simple compiler directives inserted into the programs by developers. However, it requires expensive shared-memory multiprocessors. Thus, we believe that developing M/S programs on off-the-shelf clusters is a good solution in today's technology with balanced consideration on performance, cost, and effort for parallelization.
- *On data transmission*: High-speed data transmission is a key factor for real-time processing. To achieve this, an appropriate compression algorithm has to be selected for every transmission. One solution is to employ LZO for high-speed LANs and zlib for low-speed wide-area networks (WANs). Although this solution results in a good performance for our current applications, more sophisticated techniques, such as streaming, will deliver better performance for future applications that require a high frame rate with video quality.
- *On secure execution*: When addressing security issues, it is quite important to 1) clarify which objects have to be protected from what kind of attacks, and to 2) evaluate the loss and damage sustained from attacks. For point 1), the gateway is useful for protecting other vulnerable nodes from malicious accesses. Furthermore, OpenSSL is also useful for protecting MIP server from unauthorized/unauthenticated/uncertificated accesses. For point 2), a secondary gateway is needed to maintain HPC services in case the primary gateway stops the services. In addition, any data except on the client must be anonymous. This is an effective approach if anonymous data have no significance for attackers. Assuming that the client has i) personal information, ii) patient's anonymous data, and iii) database that binds i) and ii), the cluster is allowed to have only anonymous data, if the client manages the database and the cluster has the same anonymous data as the client. The key idea is that the binding information, which gives significance to attackers, is kept inside the client.

## VI. Conclusion

We present a remote parallel-processing system for medical image processing, the MIP-Cluster system, developed for the HPC-assisted surgery. Our new system provides an intraoperative ROM estimation by parallel processing based on the M/S dynamic load balancing. It also realizes an automation framework for remote parallel processing, which includes a secure execution environment by public key cryptography and high-speed data transmission by lossless data compression algorithms.

As a result, as compared to a sequential system which takes one half of an hour for the estimation, MIP-Cluster accelerates the process to less than one half of a minute, and thereby realizes intraoperative surgical planning without degrading the resolution of images and limiting the area for estimations.

One remaining issue is to achieve fault tolerance, which enables highly reliable systems. This issue can be easily addressed by duplicating the entire system including the network between the system and the operating room, and automatically switch them with a timeout mechanism. Another issue on the ROM estimator is to reduce the amount of computation. To address this, we are currently incorporating an adaptive ROM refinement technique into the estimator. This technique manages the safe ROM in a hierarchy, where processors perform collision detections in a coarse-to-fine manner.

## Acknowledgment

The authors are grateful to the reviewers for their valuable comments that greatly improved this paper. The authors would also like to thank Dr. H. Wang, European Science Foundation, Strasbourg, France, for his valuable advice on this paper.

## References

[1] S. Martelli, R. E. Ellis, M. Marcacci, and S. Zaffagnini, "Total knee replacement kinematics: Computer simulation and intraoperative evaluation," *J. Arthroplasty*, vol. 13, no. 2, pp. 145–155, Feb. 1998.

[2] R. E. Ellis, C. Y. Tso, J. F. Rudan, and M. M. Harrison, "A surgical planning and guidance system for high tibial osteotomy," *Comput. Aided Surgery*, vol. 4, no. 5, pp. 264–274, 1999.

[3] B. Ma and R. E. Ellis, "Robust registration for computer-integrated orthopedic surgery: Laboratory validation and clinical experience," *Med. Image Anal.*, vol. 7, no. 3, pp. 237–250, Sept. 2003.

[4] A. M. DiGioia, B. Jaramaz, M. Blackwell, D. A. Simon, F. Morgan, J. E. Moody, C. Nikou, B. D. Colgan, C. A. Astion, R. S. Labarca, E. Kischell, and T. Kanade, "Image guided navigation system to measure intraoperatively acetabular implant alignment," *Clin. Orthop. Related Res.*, vol. 355, pp. 8–22, Oct. 1998.

[5] B. Jaramaz, A. M. DiGioia, M. Blackwell, and C. Nikou, "Computer assisted measurement of cup placement in total hip replacement," *Clin. Orthop. Related Res.*, vol. 354, pp. 70–81, Sept. 1998.

[6] Y. Sato, T. Sasama, N. Sugano, K. Nakahodo, T. Nishii, K. Ohzono, K. Yonenobu, T. Ochi, and S. Tamura, "Intraoperative simulation and planning using a combined acetabular and femoral (CAF) navigation system for total hip replacement," in *Proc. 3rd Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'00)*, Oct. 2000, pp. 1114–1125.

[7] T. E. Anderson, D. E. Culler, and D. A. Patterson, "A case for NOW (Networks of workstations)," *IEEE Micro*, vol. 15, pp. 54–64, Jan. 1995.

[8] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer, "BEOWULF: A parallel workstation for scientific computation," in *Proc. 24th Int. Conf. Parallel Processing (ICPP'95)*, Aug. 1995, pp. 11–14.

[9] F. Ino, K. Ooyama, Y. Kawasaki, A. Takeuchi, Y. Mizutani, J. Masumoto, Y. Sato, N. Sugano, T. Nishii, H. Miki, H. Yoshikawa, K. Yonenobu, S. Tamura, T. Ochi, and K. Hagihara, "A high performance computing service over the internet for nonrigid image registration," in *Proc. Computer Assisted Radiology and Surgery: 17th Int. Congr. and Exhibition (CARS'03)*, June 2003, pp. 193–199.

[10] F. Ino, K. Ooyama, A. Takeuchi, and K. Hagihara, "Design and implementation of parallel nonrigid image registration using off-the-shelf supercomputers," in *Proc. 6th Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'03), Pt. I*, Nov. 2003, pp. 327–334.

[11] A. Takeuchi, F. Ino, and K. Hagihara, "An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors," *Parallel Comput.*, vol. 29, no. 11/12, pp. 1745–1762, Nov. 2003.

[12] S. K. Warfield, F. A. Jolesz, and R. Kikinis, "Real-time image segmentation for image-guided surgery," in *Proc. High Performance Networking and Computing Conf. (SC98)*, Nov. 1998.

[13] ——, "A high performance computing approach to the registration of medical imaging data," *Parallel Comput.*, vol. 24, no. 9/10, pp. 1345–1368, 1998.

[14] S. K. Warfield, M. Ferrant, X. Gallez, A. Nabavi, F. A. Jolesz, and R. Kikinis, "Real-time biomechanical simulation of volumetric brain deformation for image guided neurosurgery," in *Proc. High Performance Networking and Computing Conf. (SC2000)*, Nov. 2000.

[15] T. Rohlfing and C. R. Maurer, "Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees," *IEEE Trans. Inform. Technol. Biomed.*, vol. 7, pp. 16–25, Mar. 2003.

[16] S. Ourselin, R. Stefanescu, and X. Pennec, "Robust registration of multimodal images: Toward real-time clinical applications," in *Proc. 5th Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'02), Pt. II*, Sept. 2002, pp. 140–147.

[17] H. Liao, N. Hata, M. Iwahara, S. Nakajima, I. Sakuma, and T. Dohi, "High-resolution stereoscopic surgical display using parallel integral videography and multi-projector," in *Proc. 5th Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI'02), Pt. II*, Sept. 2002, pp. 85–92.

[18] R. Buyya, Ed., *High Performance Cluster Computing*. Englewood Cliffs, NJ: Prentice-Hall PTR, 1999.

[19] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint of a New Computing Infrastructure*. San Mateo, CA: Morgan Kaufmann, 1998.

[20] S. Hastings, T. Kurc, S. Langella, U. Catalyurek, T. Pan, and J. Saltz, "Image processing for the grid: A toolkit for building grid-enabled image processing applications," in *Proc. 3rd IEEE/ACM Int. Symp. Cluster Computing and the Grid (CCGrid'03)*, May 2003, pp. 36–43.

[21] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *Int. J. Supercomput, Applic. High Performance Comput.*, vol. 11, no. 2, pp. 115–128, 1997.

[22] Y. Mizuno-Matsumoto, S. Date, Y. Tabuchi, S. Tamura, Y. Sato, R. A. Zoroofi, S. Shimojo, Y. Kadobayashi, H. Tatsumi, H. Nogawa, K. Shinosaki, M. Takeda, T. Inouye, and H. Miyahara, "Telemedicine for evaluation of brain function by a metacomputer," *IEEE Trans. Inform. Technol. Biomed.*, vol. 4, pp. 165–172, June 2000.

[23] A. Norton and A. Rockwood, "Enabling view-dependent progressive volume visualization on the grid," *IEEE Comput. Graph. Appl.*, vol. 23, pp. 22–31, Mar. 2003.

[24] M. Ferrant, A. Nabavi, B. Macq, P. M. Black, F. A. Jolesz, R. Kikinis, and S. K. Warfield, "Serial registration of intraoperative MR images of the brain," *Med. Image Anal.*, vol. 6, no. 4, pp. 337–359, 2002.

[25] M. Ferrant, A. Nabavi, B. Macq, F. A. Jolesz, R. Kikinis, and S. K. Warfield, "Registration of 3-D intraoperative MR images of the brain using a finite-element biomechanical model," *IEEE Trans. Med. Imag.*, vol. 20, pp. 1384–1397, Dec. 2001.

[26] S. K. Warfield, F. Talos, A. Tei, A. Bharatha, A. Nabavi, M. Ferrant, P. M. Black, F. A. Jolesz, and R. Kikinis, "Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided surgery," *Comput. Visualiz. in Sci.*, vol. 5, no. 1, pp. 3–11, July 2002.

[27] M. Weiser, "The computer for the twenty-first century," *Scientific Amer.*, pp. 94–104, Sept. 1991.

[28] H. Stern, M. Eisler, and R. Labiaga, *Managing NFS and NIS*, 2nd ed. Sebastopol, CA: O'Reilly Assoc. Inc., 2001.

[29] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su, "Myrinet: A gigabit-per-second local-area network," *IEEE Micro*, vol. 15, pp. 29–36, Feb. 1995.

[30] PC Cluster Consortium. (2003) SCore Cluster System Software. [Online]. Available: http://www.pccluster.org/

[31] A. Hori, H. Tezuka, and Y. Ishikawa, "Highly efficient gang scheduling implementation," in *Proc. High Performance Networking and Computing Conf. (SC98)*, Nov. 1998.

[32] "MPI: A message-passing interface standard," *Int. J. Supercomput. Applic.*, vol. 8, no. 3/4, pp. 159–416, 1994.

[33] F. O'Carroll, H. Tezuka, A. Hori, and Y. Ishikawa, "The design and implementation of zero copy MPI using commodity hardware with a high performance network," in *Proc. 12th ACM Int. Conf. Supercomputing (ICS'98)*, July 1998, pp. 243–250.

[34] OpenSSL Project. (2003) OpenSSL: The Open Source Toolkit for SSL/TSL. [Online]. Available: http://www.openssl.org/

[35] R. Housley, W. Ford, W. Polk, and D. Solo. (1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF. [Online] RFC 2459

[36] M. F. X. J. Oberhumer. (2003) LZO Data Compression Library. [Online]. Available: http://www.oberhumer.com/opensource/lzo/

[37] J.-L. Gailly and M. Adler. (2003) Zlib General Purpose Compression Library. [Online]. Available: http://www.gzip.org/zlib/

[38] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Computer Graphics (Proc. SIGGRAPH'87)*, vol. 21, July 1987, pp. 163–169.

[39] Northern Digital, Inc. (2003) OPTOTRAK. [Online]. Available: http://www.ndigital.com/optotrak.html

[40] J. V. Hajnal, D. L. Hill, and D. J. Hawkes, Eds., *Medical Image Registration* . Boca Raton, FL: CRC, 2001.

[41] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 239–256, Feb. 1992.

[42] N. Sugano, T. Sasama, Y. Sato, Y. Nakajima, T. Nishii, K. Yonenobu, S. Tamura, and T. Ochi, "Accuracy evaluation of surface-based registration methods in a computer navigation system for hip surgery performed through a posterolateral approach," *Comput. Aided Surgery*, vol. 6, no. 4, pp. 195–203, 2001.

[43] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha, "V-COLLIDE: Accelerated collision detection for VRML," in *Proc. 2nd Symp. Virtual Reality Modeling Language (VRML'97)*, Feb. 1997, pp. 117–124.

[44] M. K. Ponamgi, D. Manocha, and M. C. Lin, "Incremental algorithms for collision detection between polygonal models," *IEEE Trans. Visual. Comput. Graph.*, vol. 3, pp. 51–64, Jan. 1997.

[45] W. Schroeder, K. Martin, and B. Lorensen, Eds., *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall PTR, 1998.

[46] K. Nakahodo, T. Sasama, Y. Sato, N. Sugano, K. Ohzono, T. Nishii, S. Nishihara, K. Yonenobu, T. Ochi, and S. Tamura, "Intraoperative update of 3-D bone model during computer navigation of pelvic osteotomies using real-time 3-D position data," in *Proc. Computer Assisted Radiology and Surgery: 14th Int. Congr. and Exhibition (CARS'00)*, June 2000, pp. 252–256.

[47] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon, *Parallel Programming in OpenMP*. San Mateo, CA: Morgan Kaufmann, 2000.

**Yasuhiro Kawasaki** received the B.E. degree in information and computer sciences from Osaka University, Osaka, Japan, in 2002.

He is currently working toward the M.E. degree in the Department of Computer Science, Graduate School of Information Science and Technology, Osaka University. His current research interests include high-performance computing, Grid computing, and systems architecture and design.

**Fumihiko Ino** (S'01–A'03–M'04) received the B.E. and M.E. degrees in information and computer sciences from Osaka University, Osaka, Japan, in 1998 and 2000, respectively.

He is currently an Assistant Professor with the Graduate School of Information Science and Technology at Osaka University. His research interests include parallel and distributed systems, software development tools, and performance evaluation.

**Yasuharu Mizutani** received the M.E. degree in information and computer sciences from Osaka University, Osaka, Japan, in 2001.

From 2001 to 2002, he was a Software Engineer at Mitsubishi Electric, Kobe, Japan. He is currently working toward the Ph.D. degree in the Department of Computer Science, Graduate School of Information Science and Technology, Osaka University. His research interests include parallel programming language, dynamic load balancing algorithms, and performance evaluation.

**Noriyuki Fujimoto** (M'01) received the B.E., M.E., and Ph.D. degrees from Osaka University, Osaka, Japan.

He is an Associate Professor of the Graduate School of Information Science and Technology at Osaka University. His research interests include parallel computing, Internet computing, combinatorial optimization, and approximation algorithms. In particular, his research efforts now focus on Grid computing and web search.

Dr. Fujimoto is a Member of the ACM.

**Toshihiko Sasama** received the B.E. and M.E. degrees in information and knowledge engineering from Tottori University, Tottori, Japan, in 1995, 1997, respectively, and the Ph.D. degree in information and computer sciences from Osaka University, Osaka, Japan, in 2001.

From 2001 to 2003, he was a Research Associate at Osaka University. Since 2003, he has been an Assistant Professor in the Department of Information and Knowledge Engineering, Faculty of Engineering, Tottori University. His research interests include computer network architectures.

**Yoshinobu Sato** (M'90) received the B.S., M.S., and Ph.D. degrees in information and computer sciences from Osaka University, Osaka, Japan, in 1982, 1984, and 1988, respectively.

From 1988 to 1992, he was a Research Engineer at the NTT Human Interface Laboratories, Yokosuka, Japan. In 1992, he joined the Division of Functional Diagnostic Imaging of Osaka University Medical School as a Faculty Member. From 1996 to 1997, he was a Research Fellow in the Surgical Planning Laboratory, Harvard Medical School and Brigham and Women's Hospital, Boston, MA. He is currently an Associate Professor in the Graduate School of Medicine and Graduate School of Engineering Science at Osaka University, where he leads a group conducting research on 3-D image analysis and surgical navigation systems in the Division of Interdisciplinary Image Analysis.

**Nobuhiko Sugano** graduated and received the M.D. degree from the Osaka University Medical School, Osaka, Japan, in 1985.

He is currently an Associate Professor and the Chief of Hip Reconstruction of the Department of Orthpaedic Surgery of Osaka University Graduate School of Medicine.

**Shinichi Tamura** (S'68–M'71) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Osaka University, Osaka, Japan, in 1966, 1968, and 1971, respectively.

He is currently a Professor of Graduate School of Medicine and Graduate School of Engineering Science of Osaka University. He has published over 200 papers in scientific journals. His current research activities include works in the field of medical image processing and its applications.

Dr. Tamura is currently an Associate Editor of *Pattern Recognition* and a Vice Editor-in-Chief of *Medical Imaging Technology*. He received several paper awards from journals including *Pattern Recognition* and *Investigative Radiology*. He is a Member of the Institute of Electronics, Information and Communication Engineers of Japan, the Information Processing Society of Japan, the Japanese Society of Medical Imaging Technology, and the Japan Radiological Society.

**Kenichi Hagihara** received the B.E., M.E., and Ph.D. degrees in information and computer sciences from Osaka University, Osaka, Japan, in 1974, 1976, and 1979, respectively.

From 1994 to 2002, he was a Professor in the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University. Since 2002, he has been a Professor in the Department of Computer Science, Graduate School of Information Science and Technology, Osaka University. From 1992 to 1993, he was a Visiting Researcher at the University of Maryland, College Park. His research interests include the fundamentals and practical application of parallel processing.