DISTRIBUTED ANALYSIS OF MEDICAL IMAGE DATA USING JAVASPACE TECHNOLOGY

J. Ruminski

Gdansk University of Technology, Department of Biomedical Engineering, Gdansk, Poland

jwr@biomed.eti.pg.gda.pl

Abstract: Application of distributed processing for analysis of medical image data is presented. JavaSpace technology is used to investigate potential benefits of distributed analysis in selected medical image processing tasks including: image content description and parametric imaging. Experiments performed on LAN-based distributed system proved that application of JavaSpace technology improves computational performance with limited source code modification. We conclude that in modern consortiums (virtual organizations) the complex problems or prototypes can be solved without new investments.

Introduction

Distributed systems - DS - (and computational GRIDs) are more and more popular today. They offer potential of supercomputer performance with high scalability and easy access. Assuming, that every institution has many desktop computers (usually hundreds or thousands in case of hospitals, universities, etc.) connected to network the distributed system can be constructed, supporting resources for intensive computation (e.g. simulations during night by all idle computers). Typical DS architecture uses the hub model, where each node imports a task from the server, performing operations and returning results to the server. In this work we presents results obtained using JavaSpace technology [1] applied to image content description and parametric imaging.

Image content description

Image content description is a very important step in image content-based retrieval. During last years image content-based retrieval have been a very active area of research [2][3]. The MPEG7 applications and dedicated medical applications have been proposed [4]. However the very important problem is to describe effectively image content for image libraries. Especially in a region-based content description the description time (usually including an image segmentation) is high. Taking into account the huge sets of images (as it is in medicine) the description engines requires higher performance. Fortunately, elementary image description does not depend on other conditions. Since the elementary description process is independent so the entire image libraries description task can be splited between nodes of a distributed processing system (fig.1).

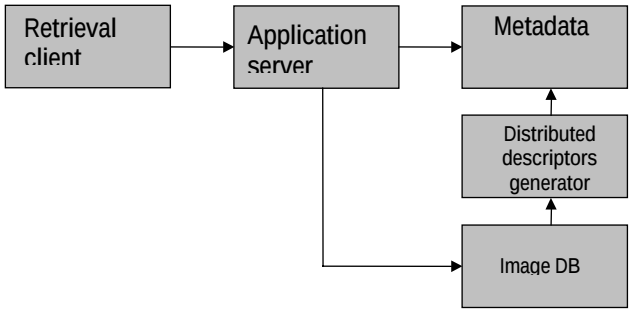

Figure 1: Architecture of content based retrieval system with distributed descriptor generator.

There are different sets of descriptors proposed in literature (e.g. MPEG7). In this research we used descriptors introduced in previously constructed content based retrieval system [4].
Run-length codes (RLC) are generated for a segmented image (e.g., with adaptive thresholding technique). With a one pass scanning of an image a table consisting of N (N- number of runs) runs is constructed. Each row of the table stores four run attributes: a value of the run V, the first pixel of the run FP (From Pixel), the last pixel of the run TP (To Pixel), and the region label which the run belongs to (RLC={V, FP, TP, ID}). As a result all unique regions are labelled, which produces a set of run-length encoded (RLE) regions. The whole algorithm requires only the previous image raw runs to test for adjacency and a single vector for equivalence analysis. It operates very fast since only iteration and comparison operations are required. With the RLE-regions a set o descriptors is proposed for intensity, regularity and shape features including:
- a set of first order statistical descriptors for intensities and ;
- a Minimum Bounding Rectangle (MBR) – min/max FP, TP values of a region runs;
-a total area of a region A (R – the total number of a region runs) - $A = \sum_{i=1}^{R} (TP_i - FP_i + 1)$,
- a region contour {B(n,m)} and a perimeter P – number of (FP, TP) pixels of vertical and horizontal runs (or No of pixels after application of morphological gradient Rp=R-E(R,N4),R–region, E-erosion, N4–struct. element [0 1 0][1 1 1][0 1 0]);

ISSN: 1727-1983

- a compactness $C = \frac{P^2}{A}$;

- a region centroid (a center of a region gravity) G(x,y) – according to statistical moments:

$x = \frac{M_{1,0}}{M_{0,0}}; y = \frac{M_{0,1}}{M_{0,0}}$; $M_{0,0}$ =A and $M_{1,0}$, $M_{0,1}$ can be calculated as the sum of horizontal ($M_{1,0}$)/vertical ($M_{0,1}$) runs indexes: $M_{1,0} = \sum_{i=FP}^{TP} i$, $M_{0,1} = \sum_{i=FP}^{TP} i$;

- a set of first order statistical descriptors based on a histogram of the region contour to the centre of gravity distances distribution {B(n,m)-G(x,y)},

$$h_k\left[\sqrt{(x_i - x)^2 + (y_j - y)^2}\right],\ (x_i, y_j) \in B;$$

- a set of first order statistical descriptors based on horizontal/vertical run-length distributions,
histograms: $h_h[(PK_i\text{-}PP_i+1)]$, $h_v[(PK_i\text{-}PP_i+1)]$;
normalized energy E_{hv} and entropy H_{hv}:

$$E_{hv} = \frac{E_h + E_v}{\mu_h + \mu_v},\ H_{hv} = \frac{H_h + H_v}{\mu_h + \mu_v};$$

- spot density descriptor,

$D_S = \frac{N_S}{A}$, Ns – No of runs with length = 1.

Additionally a set of functions for region topology analysis like distance, adjacency, overlay, neighbourhood, etc. Defined functions use RLC description of regions and centre of gravity. Concluding, RLC based descriptors for shape, structure and values construct the full region representation for content based retrieval purposes and are used to test image description in the distributed system.

Parametric imaging

Synthesis of parametric images uses dynamically measured data fitted to the model (a curve). In this study we used parametric images for dynamic, active thermography (AT) and dynamic susceptibility contrast (DSC) MRI.

AT parametric images

In the pulsed thermography a target object is excited (optical heating used in the reported study) during a given time period t_1. As a result of the heating the object temperature varies in time according to thermal properties of the object. Taking into account a single point of the object its temperature can be represented as (fig. 2)

$$S_i^p = \left\{s_1^p, s_2^p, s_2^p, .., s_i^p\right\}; \quad (1)$$

where: i – total number of samples measured during cooling, p – index of a measurement point.

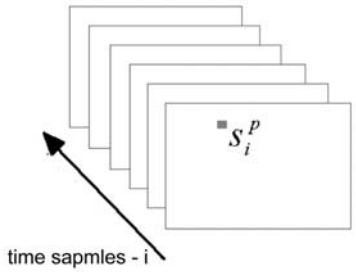

Figure 2: Sequence of images as a set of samples (S) for each point (p) in time (i).

Sequence of images was measured to calculate a set of samples for each point in time. Images were recorded by thermal camera with frequency adjusted to the object properties and heating conditions. In the applied pulsed thermography the optical heating (a set of lamps - 1000W) was usually lasting 30s. Thermal images were captured every 1s during 180 seconds (180 images, 180 samples for a one pixel). The character of sample values distribution is exponential and can be described by the formula

$$s_i^p = B_0^p + \sum_{j=1}^{N} B_j^p * \exp\left(-\frac{t_{i-1}}{\tau_j^p}\right); \mathrm{i} = 1..\mathrm{c}; \quad (2)$$

where: B – amplitudes, s_i^p - analytical value of a sample, τ_j^p - time constant for layer j. We explore one and two layers models (two time constants A1= B_1^p, A2=1/τ_1^p, A3= B_2^p, A4=1/τ_2^p). The equation describes also a simple RC electrical circuit with the time constant τ^p=R*C. For thermal studies the model and its parameters represent thermal properties of the object (conductivity, heat capacity). Quantitative description for the thermography can based on such parameters. Parametric image reconstruction method is looking for parameters of the equation (2) which minimize fitting errors to the measured set of samples. This is achieved by the application of the χ^2 test, and the fitting algorithm. We used Marquardt method with the modification given by Bevington [5].

AT material

Images were collected during phantom and in-vivo (domestic pig) studies using procedure presented above. Three different phantoms and three domestic pigs. The Local Ethics Commission approved the plan of experiments conducted on young pigs, each weighing about 20 kg. The animals were subjected to reference burns of different depth, inflicted following the procedure adopted from Singer [6]. In the course of healing burn wounds were left without dressings and medicaments. The wounds mechanically damaged have been excluded from evaluation. Finally 23 symmetric wounds were subjected to the analysis. Time sequences

ISSN: 1727-1983

were measured during three days following the burn procedure. Total number of image sequences was about 70.

DSC-MRI parametric images

In the DSC-MRI brain studies, after injection of a bolus of contrast agent (Gd-DTPA), a series of images are measured. This time-sequence data presents local voxel activity of contrast (blood) flow and distribution. It is assumed, that measured MRI signal values are proportional to the contrast concentration. Contrast concentration as a function of time is measured for brain supported arteries. This function can be estimated as the arterial input function (AIF). Assuming ideal conditions this function should be an ideal impulse function, so measuring the output function (impulse response) one can specify properties of the object under study, including mass flow, mass volume, and mean transfer time. Since AIF is not an ideal impulse function (dispersion and delay) and because in DSC-MRI measurements are done from a volume of interest (VOI), deconvolution should be used to calculate VOI impulse response *F*R(t)* [7]:

$$C_t(t) = \frac{\rho}{Kh}\int_0^t C_a(\tau)\cdot\big(F\cdot R(t-\tau)\big)d\tau \;; \tag{3}$$

where: $C_a(t)$ - contrast concentration in the artery (e.g., Middle Cerebral Artery) – Arterial Input Function AIF,
- $C_t(t)$ - contrast concentration in the tissue,
- $\frac{\rho}{Kh}$ - scaling factor (quantitative description) ρ – mean tissue density of a brain ρ=1,04 g/mol, Kh – hematocrit ration (large to small arteries) Kh=(1-Hd)/(1-Hm); Hd=0.45; Hm=0.25,
- F*R(t) – scaled impulse response (residue function) inside VOI, R(t) - represents fractional tissue concentration:

$$R(t) = 1 - H(t) = 1 - \int_0^t h(\tau)\cdot d\tau \;; \tag{4}$$

- h(t) - transport function - impulse response (an ideal instantaneous unit bolus injection). Distribution of transit times through the voxel; depends on the vascular structure and flow.
The model is based on tracer kinetics for nondiffusable tracers – contrast material remains intravascular [8].
Since the first pass of the contrast bolus should be analyzed the recirculation need to be eliminated from the concentration function. It could be performed by the concentration function model fitting to measured data. The following model is usually used to describe the concentration function [9]:

$$C(t) = \begin{cases} K(t-t_0)^{\beta}\cdot e^{-\alpha(t-t_0)}, t > t_0 \\ 0, t \le t_0 \end{cases} \;; \tag{5}$$

where: K, α, β model parameters, t_0 - bolus arrival time (BAT). Model parameters were calculated using model to data fitting.
Scaled impulse response could be calculated using Fourier transforms (FFT) or matrix algebra (with matrix decomposition SVD to eliminate singularities). Since R(t=0) should be equal to 1, then

$F \cdot R(t=0) = F = CBF$ (Cerebral Blood Flow).

Cerebral blood volume (proportional to the normalized total amount of tracer) can be calculated as

$$CBV = \frac{\int_0^{\infty} C_t(\tau)d\tau}{\frac{\rho}{Kh}\int_0^{\infty} C_a(\tau)d\tau}. \tag{6}$$

Based on central volume theorem, mean transit time (average time required for any given particle of tracer to pass through the tissue after an ideal bolus injection) can be estimated as

$$\text{MTT=CBV/CBF.} \tag{7}$$

Three types of quantitative parametric images (CBF, CBV, MTT), synthesized under strictly controlled procedure, offer additional space to construct new descriptors for content based retrieval.

DSC-MRI material

We collected images for in-vivo measurements (1.5T MRI SE-EPI with: 12 slices, 50 samples, TR=1.25-1.61s; TE=32-53ms; slice thickness 5-10 mm; 60 series - 3000 images). Images were collected for typical cases and for cases where Blood-Brain Barrier was damaged. Using own, created software we extracted signals and concentration curves used in further processing.

Method – distributed analysis

Let's assume that the task can be represented by a universal interface which supports a simple method "execute":

public Entry execute(JavaSpace js);

where Entry is the interface representing functionality of the object which can be stored in logically defined space (JavaSpace) of objects (memory where objects are organized and stored). Constructing particular task to be solved (treated as elementary operation, even if it is a complex one) the class can be defined which implements the "execute" method. In the "execute"

ISSN: 1727-1983

implementation the set of instructions are programmed to solve the task (problem) and return a result.
Taking into account the parametric imaging problem (where solution uses model fitting to a measured data) the fitting task is described in Listing 1.

Listing 1. FitTask class – objects of this class are elementary tasks to be solved by execute() method in the distributed system.

```
public class FitTask extends TaskEntry{
(…)
public Entry execute(JavaSpace space) {

 double [] results;
 (…)
 setParameters();
 results=fit();
 (…)
 FitResult fr = new FitResult(results,noSeries);
 return fr;
}//end of execute()
}//end of class
```

In case of image content the only difference in the listing 1 is the change of the task class name (DescribeTask) and describe method *describe()*. Further we keep the parametric imaging problem as an ilustration for general distributed analysis of medical image data using JavaSpace technology.
The FitTask object receives the task data (1D signal), performs fitting procedure and creates a result object which is passed to the space. The result is an object of a Result class which implements the Entry interface. In the space there are always Entry objects which are either Tasks or Results. Applications can use the space in the way that clients (service users) write tasks to the space, and servers (Compute Servers [10]) check if there is any task in the space, if so, then the Task object is taken (moved from the space), the "execute" method is executed so the Result object is written back to the space. Compute servers check the space waiting for new tasks. In this same way the client (clients) checks if there is any result ready to be taken, if so, the Result object is taken (moved from the space) preparing the final solution (as a set of Result objects with or without data reorganization). In the Listing 2 the client functionality is presented.

Listing 2. FitMaster class – generates tasks to the space and collects results. FitMaster is a client software that creates a problem to be solved by a distributed processing system.

```
public class FitMaster{
(…)
public void generateTasks(){
 (…)
 while(dataAvailable()){
  getTimeSeries();
  FitTask task =
  new FitTask(params,time,series,taskNo,function);
  writeTask(task);
  (…)
 }
}//end of generateTasks()
public void collectResults(){
 Entry template;
 (…)
 template = space.snapshot(new FitResult());
 while(resultsAvailable()){
  FitResult result = (FitResult)takeTask(template);
  setParametricMaps(result);
 }
 }//end of collectResults()
}//end of class
```

The FitMaster/DescribeMaster class can include (as it was done in the reported study) a GUI (fig. 3) which allows a user to specify image series, a model, and other parameters (e.g., allowed number of task objects in the space).

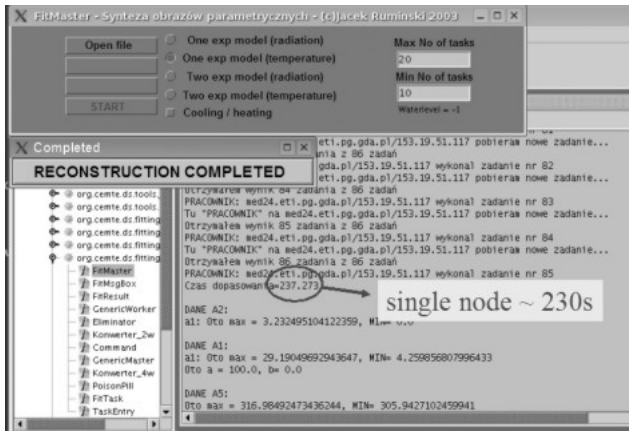

Figure 3: An example of the FitMaster GUI and results screen capture

The most important feature of the ComputeServer solution is that compute servers do not know anything about the task except the "execute" method (which knows how to solve the problem). The space and a set of compute servers can solve any task written by clients. In the Listing 3 the GenericWorkerer class is presented. It operates using three simple functions: take a task object from the space, solve it by calling execute() method, save a result object in the space.

Listing 3. GenericWorkerer class – the server side code, responsible for listening, taking and solving tasks. It's also generates results to the space.

```
public class GenericWorkerer{
 (…)
 public void run(){
  TaskEntry task = (TaskEntry)space.take(taskTmpl,
txn, Lease.FOREVER);
  Entry result = task.execute(space);
  if (result != null) {
      space.write(result, txn, task.resultLeaseTime());
  }
```


ISSN: 1727-1983

}//end of run()
}//end of class

The compute server can be implemented on any computer in any place having access to the computer network. Each compute server knows (by IP or by discovering) where is the space to be analysed. Within an institution the compute server can be started as a demon with dedicated resources (e.g. priority). This transparently support solving problems which require high computational power. In Fig. 4 distributed architecture of the parametric images synthesis system is presented.

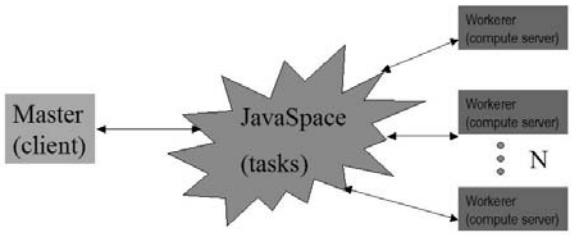

Figure 4: Distributed architecture of the parametric images synthesis system based on the JavaSpace technology and ComputeServer methodology

Using JavaSpace technology and Compute Server method we constructed distributed system build with 18 computers (Intel Celeron/ AMD Athlon XP) running under Linux Debian operating system. All computers were connected by Gigabit Ethernet switch. On each computer the GeneralWorkerer (compute server) has been installed as a demon. The one node has been used to operate required Jini services (lookup, transaction manager, JavaSpace). We used this configuration by connecting laptop computer (client) to the network (outside the LAN). Using the client software it was possibly to specify the input data (image library, image series) and other parameters. In the applied tests we used real medical images in DICOM format (1320 T1 and T2 weighted MRI images and 1560 CT images, AT raw-sequences and DSC-MRI data).

Results and conclusions

First experiment (a reference one) was to find out the time required to calculate the solution (descriptors and a set of parametric images) using a single node. In case of content description it took 0.1-1s depending on image matrix size (134kB-1MB) and an image content. Using AT and DSC-MRI image series (DSC-MRI clinical data: 256x256x50 and AT domestic pig studies: 204x128x147) the single node computation (AMD Athlon XP 1800, 512MB RAM) took usually 160 - 237s. The experiment consumed more time in case of AT image series (up to 237 seconds) which was a result of the number of time samples (more data in an elementary fitting task).

In case of introducing of next nodes (compute servers) we achieved linear scalability up to five (content description) and seven (parametric images) nodes. In case of image description the task object had implemented the image matrix so its transmission is very costly in terms of time. However the only matrix delivery is required: the result object has only descriptors. I case of image content description the application of JavaSpace technology outside the LAN (i.e. using low transmission speed) is not useful. In digital radiology department there are usually many computers working within a LAN (PACS) so it could be easily used to implement distributed image content description task (e.g. during night time).
In case of parametric image synthesis an average time required to solve DSC-MRI problem in distributed (7 nodes) environment was 20 seconds, while in AT case it was about 34s (with about 10% of the standard deviation). Taking more than 10 nodes did not improve the computational performance (it depends on the construction of the "execute" method and how the global problem is partitioned to Task objects). In the Listing 4 some messages from GeneralWorkerer nodes are presented.

Listing 4. Some messages from GeneralWorkerer nodes. It noticed that messages are coming asynchronously.

```
(...)
WORKERER: med6.eti.pg.gda.pl/153.19.51.83
has just completed task No 81
WORKERER: med27/127.0.0.1
has just completed task No 80
WORKERER: med5/127.0.0.1
has just completed task No 75
WORKERER: med8.eti.pg.gda.pl/153.19.51.85
has just completed task No 83
WORKERER: bio4.eti.pg.gda.pl/153.19.51.100
has just completed task No 82
WORKERER: med28.eti.pg.gda.pl/153.19.51.122
has just completed task No 84
WORKERER: med29.eti.pg.gda.pl/153.19.51.123
has just completed task No 85
(...)
The work completed in 30.953 seconds.
```

The potential value of the JavaSpace solution is a simplicity of its application. The JavaSpace offers only a limited set of functions focused on: writing objects to the space and reading/taking objects from the space. In the presented applications (image content description and parametric imaging) the single node solution was modified to fulfil the JavaSpace requirements. In case of required code modification it took about an hour (for each problem). An example of the parametric images generated using JavaSpace technology is presented in Fig. 5.

ISSN: 1727-1983

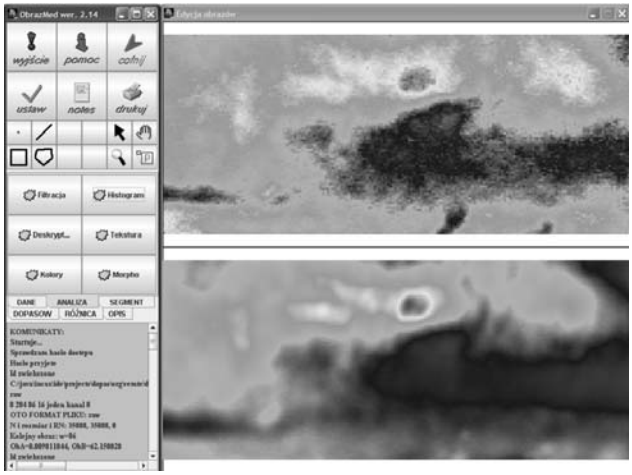

Figure 5: Parametric images (cooling rate - τ) in AT thermography used in burn depth assessment: τ value (colour) distribution indicates regions with different burn depth

We tested only two domains: image content description and parametric imaging. However the same methodology can be used in other domains like functional MRI, dynamic PET studies (with a rich set of compartmental models for different isotopes experiments), etc.
Application of distributed analysis of medical image data using JavaSpace technology offer cost effective solution of complex problems. The required source code modification is minimal. Since most problems in medical imaging and image analysis can be broken into smaller tasks then different medical applications are possible (e.g. real time 3D rendering in the operating room, intensive image reconstruction, etc.). This is especially attractive in cost efficient Service Oriented Architecture of systems, where computational resources can be offered by distributed institutions (e.g. universities). In modern consortiums (virtual organizations) the complex problems or prototypes can be solved without new investments (e.g. using existing infrastructure of computer laboratories). The different levels of security problems can be solved using as well configurable services (e.g. access control list using java policy) as Java available solutions (e.g. encryption, secure transport layers, etc.).

Acknowledgements

The work was partially supported by the Polish State Committee for Scientific Research, grant No 4 T11E 042 25, 2003-2006).

References

[1] FREEMAN E., HUPFER S., ARNOLD K. (1999), JavaSpaces Principles, Patterns and Practice, Addison Wesley.
[2] SMITH J.R. (1997), Integrated Spatial and feature image systems: retrieval, analysis and compression, PhD dissertation, Columbia University.
[3] CAI W., FENG D. D., FULTON R. (2000), Content based retrieval of dynamic PET functional images, IEEE Transactions on Information Technology in Biomedicine 4 (2)152-158.
[4] RUMINSKI J. (1999), Content-based medical image retrieval. Medical a. Biolog. Eng. a. Computing incorporat. Cellular Eng., 1999, vol. 37 Suppl. 2 s. 680-681, 1999.
[5] BEVINGTON P.R., ROBINSON D.K. (1991): Data Reduction and Error Analysis for The Physical Sciences, McGraw-Hill Higher Education.
[6] SINGER AJ, BERRUTI L, THODE HC, MCCLAIN S.A (2000), Standardized burn model using a multiparametric histologic analysis of burn depth. *Acad. Emerg. Med*; **7:** 1-6.
[7] ØSTERGAARD L., WEISSKOFF R.M., CHESLER D.A., GYLDENSTED C., ROSEN B.R. (1996), High resolution measurement of cerebral blood flow using intravascular tracer bolus passages: I. Mathematical approach and statistical analysis, II. Experimental comparison and preliminary results *Magn. Reson. Med.* 36, 715–736.
[8] ZIERLER KL (1962). Theoretical Basis of Indicator-Dilution Methods for Measuring Flow and Volume. *Circ.Res.* **10**:393-407.
[9] SORENSEN A.G., REIMER P. (2000), Cerebral MR Perfusion Imaging, Principles and Current Applications, Georg Thieme Verlag, Stuttgart.
[10] FREEMAN E., HUPFER S., (2000), Make room for JavaSpaces, Part 2, Build a compute server with JavaSpaces, http://www.javaworld.com/jw-01-2000/jw-01-jiniology.html

ISSN: 1727-1983