

Ministry of Higher Education and Scientific Research  
University M'hamed BOUGARA Boumerdes  
Faculty of Engineering  
Department of Electrical Engineering and Electronics



## Report

Presented in Partial Fulfilment of the requirements of the

### **MAGISTER DEGREE**

In Electronics Systems Engineering

#### **Title**

### **APPLICATION OF PARALLEL PROCESSING TO MEDICAL IMAGING**

*Soutenu le: 27/09/2005*

By

Abbès Bouklachi

In front of the Examiner board

Pr. M. MEZGHICHE	Prof. at Univ. of Boumerdes	President
Pr. K. BADARI	Prof. at Univ. of Boumerdes	Supervisor
Dr. A. HERZELLAH	Maitre de conference at the Univ. of Boumerdes	Examiner
Dr. M. AHMED NACER	Maitre de conference at the Univ. of USTHB	Examiner

2007/2008

### المخلص

يعنى هذا البحث بتطبيقات البرمجة المتوازية في التصوير الطبي من خلال تطوير حلول ذات فعالية عالية لحساب بعض المعالجات التي تمر بها الصورة في أجهزة التصوير الطبي من حيث أداء التنفيذ و قابلية التسلق و الشمولية. لقد انتقينا معالجات شديدة التعقيد و الحمولة، ثم طورنا لها حلولاً ذات فعالية عالية. لقد اتبعنا في تطوير هذه الحلول على مبادئ البرمجة المتوازية التي تستغل التزامن المتضمن في الحساب. هذا من جهة و من جهة أخرى خزنا صورة عضو الإنسان في شفرات ترتب معطيات العضو المحصل عليها من أجهزة الفحص بحيث تمكننا من ربح الوقت في معالجة المناطق المتجانسة. كما اعتمدنا في بحثنا هذا على استعمال مختلف أشكال التنسيق و الجدولة فوق الشبكة لنضمن اتزان الحمولة و القدرة على التسلق. وأخيراً استعملنا تراكيب مثالية للشبكة لدراسة أداء الحسابات و إثارة الجدل حول فعاليتها بدون التقيد بتركيبة خاصة ترقباً لأنظمة الحاسب المستقبلية.

### Abstract

The objective behind this research work is to design efficient parallel solutions to selected medical imaging algorithms in terms of performance, scalability and generality. We have selected algorithms which present a considerable complexity and heavy load and designed solutions based on one hand, parallel programming *concepts to exploit the parallelism inherent in the algorithm*. On the other hand, encoded medical data sets with encoding schemes to enforce a structure so that we can identify coherent regions and save processing time for homogenous regions. Furthermore, we have used parallel models to exploit various forms of process synchronization and scheduling over network topologies to ensure load balancing and the ability to scale up. Finally, abstract machine architectures have been used to study the algorithms and argue about their efficiency without getting too much concerned with a specific hardware.

### Résumé

Ce travail de recherche consiste à la conception de solutions efficaces à des algorithmes d'imagerie médicale en termes de performance, de croissance et de généralité. Nous avons sélectionné des algorithmes qui présentent une complexité considérable et une importante charge et conçu des solutions basées sur deux concepts. Le premier consiste à exploiter le parallélisme inhérent dans l'algorithme tandis que le deuxième consiste à coder les données avec une structure qui nous permet d'identifier les régions cohérentes et par conséquent réduire le temps d'exécution pour les régions homogènes. En outre, nous avons utilisé les modèles parallèles pour exploiter une variété de formes de synchronisation et de planification sur les topologies de réseaux pour assurer la balance de charge et l'habilité de croissance. Finalement, nous avons utilisé des architectures de machine abstraite pour étudier les algorithmes et de débattre leurs efficacités sans se soucier d'un hardware spécifique.

## Table of contents

<b>1 Introduction</b>	1
1.1 Research statement	2
1.1.1 Motivation	2
1.1.2 Research objectives	3
1.1.3 Research approach	4
1.2 Identification of medical imaging algorithms and case studies	5
1.2.1 Information flow in a medical imaging system	5
1.2.2 Data acquisition	5
1.2.3 Image processing	6
1.2.4 Model creation	7
1.2.5 Viewing operations	7
1.2.6 Generation of realistic images	7
1.3 Concepts used in the design of the parallel algorithms	9
1.3.1 Parallel models used to exploit the parallelism inherent in the algorithm	9
1.3.2 Data encoding used to exploit data parallelism	9
1.3.3 Synchronisation through messages (parallel models (MPS & BSP) ..	9
1.3.4 Distributed memory systems	10
1.3.5 Mating other models to the algorithms	10
1.4 Evaluation of performance	11
1.5 Thesis organisation	12
<b>2 Medical imaging</b> (Algorithms, software packages, computational requirements, medical imaging applications)	13
2.1 Medical imaging algorithms	14
2.1.1 Acquisition algorithms and their computational requirement	16
2.1.1.1 Sampling and reconstruction algorithms	16
2.1.1.2 Reconstruction algorithms	18
2.1.1.3 Computational requirements	19
2.1.2 Resampling and interpolation algorithms	20
2.1.3 Representation of 3D medical objects and encoding algorithms.....	21
2.1.3.1 Boundary representations	24
2.1.3.2 Volume representations	24
2.1.3.2.1 The VDE file format	26
2.1.3.3 Octrees and Quadtrees	27
2.1.3.3.1 Octree and Quadtree data structures	29
2.1.4 Segmentation algorithms	31
2.1.4.1 Image segmentation	31
2.1.4.2 Edge detection algorithms	33
2.1.4.3 The elastic matching algorithm (mssm)	34
2.1.5 Visualisation algorithms	36

2.1.5.1	Volume rendering .....	36
2.1.5.2	Octree visualisation method .....	40
2.2	Medical imaging systems, software tools and their computational requirements .....	42
2.2.1	Medical imaging computational requirements .....	42
2.2.2	Approaches to the use of 3D graphics in medical imaging .....	42
2.2.3	Medical graphics systems .....	44
2.2.3.1	ANALYSE software package .....	44
2.2.3.2	The VDE software .....	46
<b>3</b>	<b>Parallel computers , performance measures, models of parallel computations</b> .....	<b>48</b>
3.1	Parallel computers .....	49
3.1.1	Introduction to parallel computers and Distributed Memory Multicomputers .....	49
3.1.1.1	Pipeline computers .....	49
3.1.1.2	Vector computers .....	50
3.1.1.3	Multiprocessors .....	51
3.1.1.4	Distributed Memory Multicomputers .....	55
3.1.2	Transputer and related systems .....	58
3.1.2.1	Transputer architecture & support of concurrency.....	58
3.1.2.2	Support of concurrency .....	59
3.1.2.3	The INMOS link .....	61
3.1.2.4	Performance .....	63
3.1.2.5	Building systems with transputers .....	64
3.2	Performance measures .....	65
3.2.1	Introduction (Amdahl's law and other laws) .....	66
3.2.2	Definitions and laws .....	67
3.2.2.1	Crosch's law .....	67
3.2.2.2	Performance and technology .....	68
3.2.2.3	Von Neumann's Bottleneck .....	68
3.2.2.4	Amdahl's law .....	69
3.2.2.5	The Gustaffson-Barsis law .....	72
3.2.2.6	Fine grain vs coarse grain machines and R/C ratio .....	73
3.2.3	Performance models .....	75
3.2.3.1	Basic model: Two- processors with unoverlapped communications .....	75
3.2.3.2	Extension to N processors .....	77
3.3	Models of parallel computation .....	81
3.3.1	Introduction .....	81
3.3.2	Techniques for exploiting parallelism .....	82
3.3.2.1	The processor farm .....	82
3.3.2.2	Algorithmic parallelism .....	82
3.3.2.3	Geometric parallelism .....	83
3.4	Parallel languages .....	84
3.4.1	Introduction .....	84
3.4.2	The ANSI C toolset .....	84
3.4.2.1	Introduction .....	84
3.4.2.2	Support for earlier tools .....	85
3.4.2.3	ANSI C toolset .....	85
3.4.2.3.1	ANSI C compiler .....	85

3.4.2.3.2	Generating executable code	85
3.4.2.3.3	Loading and running programs	85
3.4.2.3.4	Program development support	86
3.4.2.3.5	Runtime library	86
3.2.2.3.6	Environment variables	87
3.4.3	Parallel processing with ANSI C	87
3.4.3.1	Process, Channel, and Semaphore data types	87
3.4.3.2	Concurrency functions	88
3.4.3.3	Processes	88
3.4.3.4	Channel communication	92
3.4.3.5	Parallel programming examples	94-97
3.5	Parallel programming using ANSI C toolset on MIMD networks supported by the IMS B008 transputer board and The S708 software support	98
3.5.1	Introduction to the IMS B008 Board	99
3.5.1.1	The IMS B008 Transputer board	99
3.5.1.2	Configuration through switches	99
3.5.2	The MMS Module Motherboard Software (S708 device driver)	100
3.5.2.1	Installing the S708 device driver	100
3.5.2.2	Using the MMS to run application programs	100
3.5.2.3	Menu options	100
3.5.2.4	Description of the software configuration	100
3.5.2.5	Network mapper	101
3.5.3	Installing the IMS D7214 ANSI C toolset	101
3.5.3.1	Introduction	101
3.5.3.2	Installing the release	102
3.5.3.2.1	Installation	102
3.5.3.2.2	Setting up the toolset for use	102
3.5.3.3	confidence testing	103
<b>4</b>	<b>Design of two parallel solutions to the mssm algorithm with a logically fully connected network supporting message passing</b>	<b>104</b>
4.1	Parallel solution 1: (Task decomposition & Pipelined model)	105
4.1.1	The model	105
4.1.2	Programming with Cstools	107
4.1.3	Performance analysis	109
4.1.4	Conclusion	112
4.2	Parallel solution 2: (Data partitioning & Process farm)	113
4.2.1	The process farm model	114
4.2.2	Data partitioning and task granularity	115
4.2.3	Matching windows and data requirements	117
4.2.4	Data replication and locality of data access	117
4.2.5	Task scheduling (demand driven)	119
4.2.6	Synchronisation using buffered messages	119
4.2.7	Programming parallel solution 2 with Cstools	121
4.2.8	Performance of the parallel solution 2	122
4.3	Conclusion	124
<b>5</b>	<b>Mating LINDA, BSP and SM parallel programming models with the MSSM algorithm over distributed memory MIMD networks</b>	<b>127</b>

5.1 Introduction	128
5.2 Designing with Linda parallel programming model (Uncoupled programming)	129
5.2.1 The model	129
5.2.2 Uncoupled programming (The interface)	130
5.2.3 Programming with C-Linda the elastic matching problem	130
5.2.4 Performance of the model	132
5.3 Shared objects (abstract data types)	132
5.3.1 The model	132
5.3.1.1 Shared Objects	133
5.3.2 The interface	134
5.3.2.1 Data structures	134
5.3.3 Programming the elastic matching problem with SO model	135
5.3.4 Performance of the model	137
5.4 XPRAM programming model	137
5.4.1 The Underlying Machine Model	137
5.4.2 The XPRAM programming model	138
5.4.2.1 Process management	139
5.4.2.2 Memory access	139
5.4.2.3 Shared space	140
5.4.2.4 Process synchronisation	140
5.4.3 A programming interface for the XPRAM model	141
5.4.3.1 Process management	141
5.4.3.2 Shared memory	141
5.4.3.3 Process synchronisation	141
5.4.3.3.1 Futures	141
5.4.3.3.2 Bulk synchronous operation	142
5.4.4 Programming with XPRAM model	142
5.4.5 Performance of the model	145
<b>6 Parallelization of the Octree visualisation algorithm</b>	<b>146</b>
6.1 Introduction	147
6.2 The algorithm	148
6.2.1 The octree visualisation algorithm	148
6.2.2 The application software :Parallel Viewer Software (PVS)	150
6.3 The parallel model	151
6.4 Data granularity and task scheduling	152
6.5 Messages and their protocols	153
6.6 Process synchronization of the process farm model	154
6.6.1 Process spawning and channel links	154
6.6.1.1 CreateChannels and CreatePlist	154
6.6.2 The farmer process	155
6.6.3 The worker process	156
6.7 Design of the functions of the Parallel Viewer Software (PVS)	157
6.7.1 Acquisition and AdjustView	157
6.7.1.1 Acquisition	157
6.7.1.2 AdjustView	158
6.7.2 The octree visualization algorithm (The Vision Pipe)	159

6.7.2.1 Encoding: BuildOctree(), InitOctNode(), TestOctant() ....	159
6.7.2.2 Mapping: OctToQuad(), BuilQuadNode(), and Destroy.....	163
6.7.2.3 Display: TraverseQuadTree(), FillBuffer() .....	166
6.8 Implementation .....	168
6.8.1 The PVS (Parallel viewer software) .....	168
6.8.2 Global and functions prototypes (pvs.h) .....	173
6.8.3 Graphics library .....	175
6.8.4 Network library .....	175
6.8.5 Data partitioning library: SavePgrain() .....	175
6.8.6 Vision library .....	176
6.9 Performance of the parallel solution .....	176
6.10 Conclusion .....	177
<b>7 General conclusion</b> .....	178-179
<b>Templates</b> .....	180-185
Template 1:The interface window of the VDE software .....	180
Template 2:2D scanned slices of a head (Kennedy87] .....	181
Template 3: Low pass filtering (slice number 3 of template2) .....	182
Template 4: High pass filtering (slice number 3 of template 2) .....	183
Template 5: Visualization of the head .....	184
Template 6: Interface window of the PVS shows the software running with 4 processors	185
<b>References</b> .....	190-198