J Supercomput (2011) 58:151–159
DOI 10.1007/s11227-009-0350-1

Automatic tuning of iterative computation on heterogeneous multiprocessors with ADITHE

J.A. Martínez · E.M. Garzón · A. Plaza · I. García

Published online: 6 November 2009

Abstract This work studies the problem of balancing the workload of iterative algorithms on heterogeneous multiprocessors. An approach, called ADITHE, is proposed and evaluated. Its main features are: (1) using a homogeneous distribution of the workload on the heterogeneous system, the speed of every node is estimated during the first iterations of the algorithm; (2) according to the speed of every node, a new workload distribution is carried out; (3) the remaining iterations of the algorithm are executed. The result of this workload redistribution is that the execution times for every iteration at every node are similar and, consequently, the penalties due to synchronization between nodes at every iteration are mostly eliminated. This approach is appropriate for iterative algorithms with similar workload at every iteration, and with a relevant number of iterations. The high portability of ADITHE is guaranteed because the estimation of speed of nodes is included in the execution of the parallel algorithm. There is a wide variety of iterative algorithms related to science and engineering which can take advantage of ADITHE. An example of this kind of algorithms (morphological processing of hyperspectral images) is considered in this work to evaluate its performance when ADITHE is applied. The analysis of the re-

J.A. Martínez · E.M. Garzón (✉) · I. García
Department of Computer Architecture and Electronics, University of Almería, Ctra Sacramento S/N, 04120 Almería, Spain
e-mail: gmartin@ual.es

J.A. Martínez
e-mail: jamartine@ual.es

I. García
e-mail: igarcia@ual.es

A. Plaza
Department of Technology of Computers and Communications, University of Extremadura, Avda. de la Universidad S/N, 10071 Cáceres, Spain
e-mail: aplaza@unex.es

sults shows that ADITHE significantly improves the performance of parallel iterative algorithms on heterogeneous platforms.

Keywords Parallel iterative algorithms · Heterogeneous multiprocessors

1 Introduction

High performance computing is the field of computational science dealing with engineering and scientific applications of cluster-based computing. It concerns the development of models and strategies which allow hard computing applications to be solved using the most advanced computing platforms. Up to now most of the applications requiring large computing resources have been solved on supercomputers or clusters of shared and distributed memory multiprocessors composed by a large number of identical processors. Nowadays, the popularity and availability of multicore processors, which combine two or more independent cores into a single package, has modified (enlarged) the concept of cluster computing. Currently, a cluster can be seen as a flexible reconfigurable computing system which is composed of nodes with different characteristics and performance. These computing environments are known as heterogeneous computing systems, and there exists a strong demand for developing new strategies to face the problems appearing when a specific application has to be solved on this kind of heterogeneous environments.

In this paper we describe our experiences when trying to efficiently implement iterative algorithms on fully heterogeneous computing systems [6]. This type of algorithms can be found in a wide set of scientific and engineering problems such as partial differential equations solvers (PDEs) or image processing algorithms, among others; their common feature is their iterative behavior. Our main goal consists of designing a strategy that will allow to dynamically analyze the computational power of the processors involved in the heterogeneous system and to determine the computational burden (set of data) that must be located at each processor. This strategy is called ADITHE (ADaptive ITeration on HEterogeneous) and its main features are described in Sect. 2. Recently, other proposals of load balancing for heterogeneous systems have been developed, however hybrid parallel programming models have not been considered in their implementation [4]. The rest of the paper is organized as follows. Section 3 describes an iterative application coming from the field of image processing; it is used to evaluate the performance of ADITHE. In Sect. 4 we describe the characteristics of the computing platform and the performance estimators used to evaluate the efficiency of ADITHE. Section 5 discusses experimental results. This paper ends with some conclusions and future work.

2 The keys of ADITHE

It is well known that to exploit a heterogeneous system composed by a set of P nodes, the workload on each node i has to be balanced. For a heterogeneous system this condition means that the workload of every node has to be proportional to

its computational power or speed [1, 3]. Then, to optimize the parallel implementation of an algorithm on heterogeneous multiprocessors, it is necessary to know: (1) the computational power or speed of each node of multiprocessor; (2) a method for decomposing the problem into smaller subproblems (data partition/distribution); and moreover (3) data dependencies among subproblems and synchronization points. However, the approach to determine the values of the speed of each node can be inaccurate since the selected benchmarks can exploit the architectures in a different way than the specific algorithm, because the memory management, fine grain parallelism exploitation, and so on, depend on the specific program.

In this work, the focus is on iterative algorithms; in this kind of algorithms, a process is iterated from the initial input data, until a stopping criterion is satisfied. Generally, parallelizing iterative algorithms is based on data partitioning among nodes, since data dependence between successive iterations prevents parallel execution of the iterative process. Consequently, the computational burden attached to each node is proportional to the size of the data subset processed.

Therefore, the optimization of parallel iterative algorithms on heterogeneous systems is based on the appropriate selection of data partitions, according to the speed of each node. To carry out this optimization we propose a strategy called ADITHE which splits the iterative process into three stages:

1. *Estimation of the nodes' speed, or benchmarking stage*. The first N_1 iterations of the algorithm are isolated from the iterative loop and executed in parallel using a homogeneous data partition. The CPU time of each node i is measured at this stage (CPU_time_i). Then, the relative speed of each node i is estimated as $\{\alpha_i = CPU\_Time_1/CPU\_Time_i;\ 1 \leq i \leq P\}$ where CPU_Time_1 refers to the fastest node and P is the number of nodes of the heterogeneous system. This estimation is expected to be accurate because it takes into account the specific combination architecture–algorithm.
2. *Adaptation of iterative algorithm to heterogeneous system*. To balance the workload on the heterogeneous system, the data partition has to verify the relation:

$$\frac{W_1}{\alpha_1} = \frac{W_2}{\alpha_2} = \cdots = \frac{W_P}{\alpha_P}; \quad \text{so} \quad W_i = \alpha_i \cdot \frac{\sum_{j=1}^{P} W_j}{\sum_{j=1}^{P} \alpha_j}, \tag{1}$$

   where W_i is the workload to be attached to node i, which is proportional to the size of the data subset defined by the new partition. At this stage, the data are redistributed according the condition expressed by (1).
3. *Parallel execution adapted to heterogeneous system*. This stage includes the execution of the remaining iterations of the algorithm, and can be considered as the optimized parallel execution of the iterative algorithm on the heterogeneous system.

Figure 1 illustrates the stages included in ADITHE. The success of ADITHE is guaranteed if the run-time of these first stages is short. ADITHE is specially appropriate for iterative algorithms which verify the following conditions: (1) the number of iterations included in stage 1, $N_1 \ll N$; (2) the workload of every iteration at the initial stage should be similar to the workload of the remaining iterations. There are

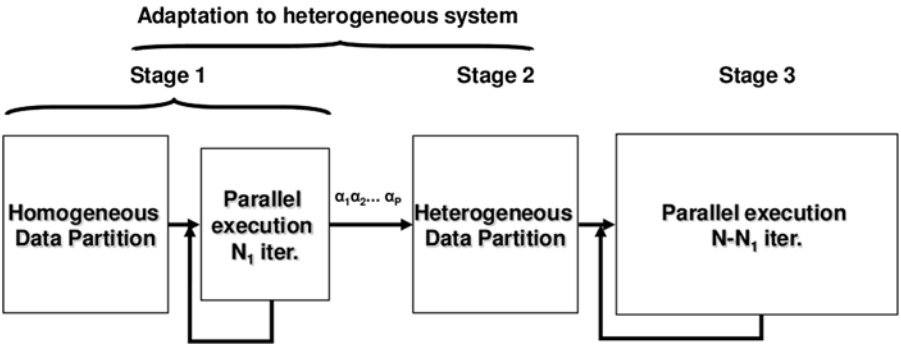

Fig. 1 Adaptive transformation of an iterative process by ADITHE approach

many examples of iterative algorithms which verify these conditions. The following section analyzes an example of iterative algorithm appropriate to ADITHE.

3 Case study: hyperspectral imaging

This section describes the application case study that will be used to illustrate the performance of the proposed framework in the context of a real application domain: the analysis of remotely sensed hyperspectral images, which can be seen as a stack of hundreds of images of the same area on the surface of the Earth collected at different wavelengths by an imaging instrument. In previous works, we have explored the application of morphological operations to integrate both spatial and spectral responses in hyperspectral data analysis [7]. Mathematical morphology is a theory for spatial structure analysis [10]. The original image is processed by a sliding window called structuring element (SE). The two basic operations of mathematical morphology are erosion and dilation.

Extension of morphological operators to multichannel data such as hyperspectral imagery with hundreds of spectral channels is not straightforward. Recently, we have developed an application-driven vector-ordering technique based on a spectral purity-based criterion [7, 8], where each pixel vector is ordered according to its spectral distance to other neighboring pixel vectors in the N-dimensional data set $\boldsymbol{f}$. More specifically, we adopt a distance-based technique which utilizes a cumulative distance between one particular pixel vector $\boldsymbol{f}(x, y)$, where (x, y) indicates the spatial coordinates, and all the pixel vectors in the spatial neighborhood given by a SE denoted by K as [9]:

$$C_K(\boldsymbol{f}(x, y)) = \sum_{(s,t)\in K} \mathrm{SAD}(\boldsymbol{f}(x, y), \boldsymbol{f}(s, t)), \tag{2}$$

where SAD is the spectral angle distance [2]. As a result, $C_K(\boldsymbol{f}(x, y))$ is given by the sum of SAD scores between $\boldsymbol{f}(x, y)$ and every other pixel vector in the K-neighborhood. Based on the above definitions, the extended erosion $\boldsymbol{f} \ominus K$ consists of selecting the K-neighborhood pixel vector that produces the minimum C_K value

Algorithm 1 Erosion & Dilation Sequential Algorithm using a 15×15 SE

1: $\boldsymbol{f} \leftarrow \boldsymbol{g} \leftarrow InputImage$
2: **for** $i = 1$ to 7 **do** {Erosion Process}
3: $\quad \boldsymbol{f} \leftarrow (\boldsymbol{f} \ominus K_3)(x, y)\ \forall (x, y) \in InputImage$
4: **end for**
5: **for** $i = 1$ to 7 **do** {Dilation Process}
6: $\quad \boldsymbol{g} \leftarrow (\boldsymbol{g} \oplus K_3)(x, y)\ \forall (x, y) \in InputImage$
7: **end for**

as follows [7]:

$$(\boldsymbol{f} \ominus K)(x, y) = \text{argmin}_{(s,t)\in K}\{C_K(\boldsymbol{f}(x + s, y + t)\}. \tag{3}$$

On the other hand, the extended dilation $\boldsymbol{f} \oplus K$ selects the K-neighborhood pixel that produces the maximum value for C_K as follows [7]:

$$(\boldsymbol{f} \oplus K)(x, y) = \text{argmax}_{(s,t)\in K}\{C_K(\boldsymbol{f}(x - s, y - t)\}. \tag{4}$$

In order to improve the quality of the results obtained through morphological processing, a large SE is generally recommended [9]. In this work, we take advantage of the fact that the results obtained using morphological operations with a large SE (say, a 15×15-pixel SE) are very similar to those obtained by applying morphological operations with a small SE (say, a 3×3-pixel SE) in iterative fashion. If we denote the latter SE as K_3, Algorithm 1 describes the approximate iterative algorithm to obtain the two morphological operations from the input data set $\boldsymbol{f}$ using a 15×15-pixel SE.

The computational load of Algorithm 1 is very large because of the dimensionality of the input data, since $\boldsymbol{f}$ is an image with hundreds of spectral bands. Then, it is appropriate to develop a parallel implementation, in order to accelerate the process. As mentioned in Sect. 1, the current prototype of multiprocessor is a cluster of P nodes with different computational power, i.e. heterogeneous multiprocessor. On the other hand, Algorithm 1 is iterative and the workload of every iteration is similar. Therefore, the ADITHE approach can be applied to Algorithm 1 with the purpose of exploiting heterogeneous multiprocessors. Algorithm 2 describes the parallel implementation of Algorithm 1 which is based on data partitioning, and the workload associated to the jth node, W_j, is proportional to the size of the sub-image at node j.

In Algorithm 2 the three stages of ADITHE can be distinguished. Lines 6–7 evaluate the parameters which allow us to establish the new heterogeneous partition data according to the values of α_j obtained in the stage 1 of ADITHE. The rest of Algorithm 2 is similar to Algorithm 1, just including the necessary communication points to develop its parallel implementation. Algorithm 2 includes the following communication points: (CP#1) The input image $\boldsymbol{f}$ is scattered across all nodes homogeneously; (CP#2) All *CPU_Times* measured in stage 1 are gathered to node 1, moreover, the updated image from the previous processing stages is also gathered to node 1; (CP#3) Node 1 scatters the updated image according to the values of α_j and the balance condition (1); (CP#4 and CP#5) These communication points are necessary to interchange the halo data between neighboring nodes due to the data

Algorithm 2 Parallel Algorithm using ADITHE; (j refers to every node $j = 1$ to P)

1: $\boldsymbol{f} \leftarrow \boldsymbol{g} \leftarrow InputImage$
2: $W_j \leftarrow W/P$ {**Stage 1: Homogeneous Data Partition. (CP#1)**}
3: **for** $i = 1$ to $N_1 = 1$ **do** {One iteration of the Erosion Process}
4: $\quad \boldsymbol{f} \leftarrow (\boldsymbol{f} \ominus K_3)(x, y)\ \forall (x, y) \in \text{Sub-image}_j$
5: **end for**
{**Gather image (CP#2)**}
6: $\alpha_j \leftarrow CPU\_Time_1/CPU\_Time_j$ {**Stage 2: Heterogeneous Data Partition**}
7: $W_j \leftarrow \alpha_j \cdot (W/\sum_{k=1}^{P} \alpha_k)$
{**Heterogeneous scatter image (CP#3); Stage 3: Main Parallel Execution**}
8: **for** $i = 2$ to 7 **do** {Erosion Process}
9: $\quad \boldsymbol{f} \leftarrow (\boldsymbol{f} \ominus K_3)(x, y)\ \forall (x, y) \in \text{Sub-image}_j$
$\quad$ {**Halo data interchange (CP#4)**}
10: **end for**
11: **for** $i = 1$ to 7 **do** {Dilation Process}
12: $\quad \boldsymbol{g} \leftarrow (\boldsymbol{g} \oplus K_3)\ \forall (x, y) \in \text{Sub-image}_j$
$\quad$ {**Halo data interchange (CP#5)**}
13: **end for**

dependence of the morphological operations, since an SE centered in a border pixel of a local partition requires information from adjacent data partitions to complete the local calculations.

4 Evaluation

There are different scalability and efficiency indicators for heterogeneous systems referred in bibliography [1, 3, 5]. Many references suggest that the node with more computational power should be taken as a reference. Our analysis is based on the so-called Heterogeneous Speed-up defined as: HSpeed(P) $= Runtime_1/Runtime_P$, where $Runtime_1$ is the run-time when only the fastest node computes the whole algorithm, and $Runtime_P$ is obtained when P nodes collaborate in the same computation. Consequently, the ideal value of HSpeed for a heterogeneous system with P nodes is HSpeed-Ideal(P)$= \sum_{j=1}^{j=P} \alpha_j$. Moreover, as $\alpha_1 = 1$ and $\alpha_j \leq 1$, then HSpeed-Ideal(P) $\leq P$, even when other penalties involved in parallel executions are not considered. However, under actual conditions, it is predictable that HSpeed(P) $<$ HSpeed-Ideal(P) due to communication and imbalance penalties.

The heterogeneous computing systems used to evaluate ADITHE are composed of a set of nodes described in Table 1. They are connected via a Gigabit Ethernet network. All the nodes run under Linux kernel 2.6. Communication among nodes is carried out under OpenMPI. In order to take advantage of the multicore architecture, the computation inside each node is developed by the POSIX-Thread library suitable for shared memory systems. As a result, a hybrid parallel programming model has been applied to exploit the two levels of parallelism, inside each node (with POSIX-Threads) and outside each node (with OpenMPI).

Table 1 Description of the Heterogeneous Systems. (MNC: Maximum Number of Cores)

H. Sys.	1	2	3	4	5
Core	Opteron	Itanium	Xeon64	Opteron	Xeon32
	2.3 GHz 64 GB	1.5 GHz 64 GB	1.9 GHz 16 GB	2.3 GHz 64 GB	2.3 GHz 1 GB
MNC	16	16	8	32	2
HS-A	1	1	1	1	1
HS-B	8	4	4	2	2
HS-C	7	3	5	1	1
HS-D	16	16	8	32	2

Table 2 Values of the relative speed (α_i) and the parallel run-times (s)

H. Sys.	α_i					Parallel run-times				
	α_1	α_2	α_3	α_4	α_5	1	2	3	4	5
HS-A	1.00	0.80	0.59	0.99	0.65	238.76	135.04	100.33	71.29	59.95
HS-B	1.00	0.41	0.30	0.26	0.16	30.44	23.31	19.47	17.11	16.79
HS-C	1.00	0.34	0.44	0.14	0.09	34.38	26.39	20.75	19.71	18.55
HS-D	1.00	0.71	0.33	1.54	0.09	15.60	10.04	9.12	5.80	6.2

Each heterogeneous system (HS-X, $X \in \{A, B, C, D\}$) is composed of nodes (1–5). Each node has a Maximum Number of Cores given by MNC. In Table 1 the set of four heterogeneous systems used in our experiments is described, where each row specifies the number of cores used from each node (1–5). It means that the data in Table 1 gives the number of used cores of each node in each heterogeneous system. For instance, the heterogeneous system named HS-B is composed of 8 cores of node 1, 4 cores of nodes 2 and 3, and 2 cores of nodes 4 and 5, respectively. Performance evaluations are based on the execution times obtained from every Heterogeneous System (HS-A, ..., HS-D) using the first ith nodes described in Table 1. Each execution is labeled as HS-X(i), where i represents the set of nodes (from 1 to i) used in a particular execution.

In order to evaluate the performance of ADITHE on a heterogeneous system, the parallel run-times of Algorithm 2 have been measured when it is executed on the heterogeneous systems described in Table 1 with the input AVIRIS hyperspectral Cuprite radiance data.[1] Table 2 shows the relative speed of every node, α_i, for each heterogeneous system (HS-X), evaluated during stage 1 of ADITHE (Algorithm 2). The strong heterogeneity of the system is shown by the difference in values of parameter α_i.

The right side of Table 2 shows the run-times measured when Algorithm 2 is executed on the heterogeneous systems previously described. In general, a relevant decrease of the run-time values can be observed when the number of nodes increases.

[1] Available on-line: http://aviris.jpl.nasa.gov/html/aviris.freedata.html.

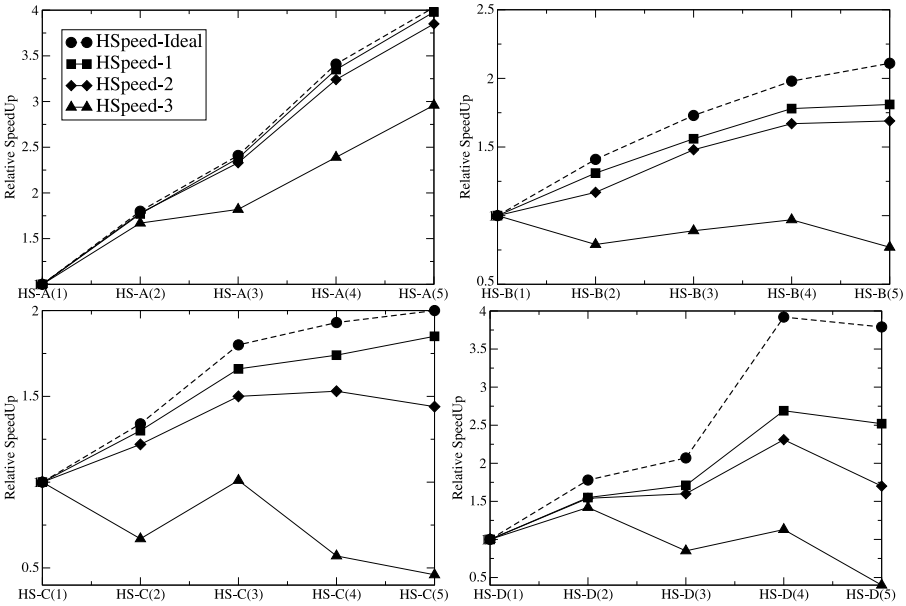

Fig. 2 Relative speed-up obtained on the considered heterogeneous systems

However, the shortest run-time is achieved for HS-D(4). When the last node (HS-D(5)) is added, the run-time increases due to the fact that communication penalties are more relevant than the increase of computational power of the system.

Figure 2 plots the heterogeneous speed-up obtained for the four systems, taking as reference HS-X(1). In this figure, HSpeed-Ideal represents the theoretical maximal heterogeneous speed-up according to the values of α_i (HSpeed-Ideal(P)= $\sum_{i=1}^{i=P} \alpha_i$). HSpeed-1 shows the heterogeneous speed-up achieved when Algorithm 2 is executed on every heterogeneous system without the stage 1 of ADITHE. In this case, the values of α_i are precomputed and introduced as inputs into the program. HSpeed-2 gives the heterogeneous speed-up obtained when Algorithm 2 is executed on the heterogeneous systems as proposed in this work. Finally, HSpeed-3 represents the heterogeneous speed-up achieved when Algorithm 1 is executed using a homogeneous data distribution.

The main penalties for the parallel execution on heterogeneous systems are the communication process and the idle time on every node due to the workload imbalance. We can estimate these penalties comparing the plots in Fig. 2. So, the difference between HSpeed-3 and HSpeed-2 shows how ADITHE improves the performance of the parallel algorithm, distributing the workload according to balance conditions given by (1). Notice that HSpeed-2 only includes penalties due to communications and stage 1 of ADITHE. As HSpeed-1 includes only communication penalties, then the difference between HSpeed-1 and HSpeed-2 is due to the estimation of the relative speed-up (α_i) and the subsequent redistribution of data. As can be seen in Fig. 2, these differences are not relevant. Consequently, these results show that ADITHE has a great impact on the performance of heterogeneous systems when iterative algorithms are executed in parallel.

5 Conclusions and future work

In this paper, we have proposed and evaluated an approach to balance the workload of iterative algorithms on heterogeneous multiprocessors. The evaluation results show that ADITHE is suitable for efficiently executing parallel iterative algorithms on heterogeneous multiprocessors. The load is redistributed according to the computational power of every node, measured during the execution of the first iterations. Moreover, the portability of ADITHE is higher than in other approaches from literature, since it does not need a previous benchmarking process. The advantages of ADITHE have been evaluated in the context of an application domain in which iterative algorithms follow a communication pattern typically from the field of image processing. The evaluation of ADITHE with other kinds of iterative algorithms with different communication patterns will be considered in our future work.

Acknowledgements This work has been funded by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117 and TIN2007-29664-E), Junta de Andalucía (P08-TIC-3518), in part financed by the European Regional Development Fund (ERDF) and the Marie Curie RTN (MRTN-CT-2006-035927).

References

1. Bosque JL, Pastor L (2006) A parallel computational model for heterogeneous clusters. IEEE Trans Parallel Distrib Syst 17(12):1390–1400
2. Chang C-I (2003) Hyperspectral imaging: techniques for spectral detection and classification. Kluwer, New York
3. Chen Y, Xian-He S, Wu M (2008) Algorithm-system scalability of heterogeneous computing. J Parallel Distrib Comput 68(11):1403–1412
4. Galindo I, Almeida F, Blanco V, Badía-Contelles JM (2008) Dynamic load balancing on dedicated heterogeneous systems. In: Recent advances in parallel virtual machine and message passing interface. LNCS, vol 5205. Springer, Berlin, pp 64–74
5. Kalinov A (2006) Scalability of heterogeneous parallel systems. Program Comput Softw 32(1):1–7
6. Lastovetsky A (2003) Parallel computing on heterogeneous networks. Wiley, Hoboken
7. Plaza A, Martinez P, Perez R, Plaza J (2002) Spatial/spectral endmember extraction by multidimensional morphological operations. IEEE Trans Geosci Remote Sens 40:2025–2041
8. Plaza A, Martinez P, Perez R, Plaza J (2004) A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. IEEE Trans Geosci Remote Sens 42:650–663
9. Plaza A, Martinez P, Plaza J, Perez R (2005) Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations. IEEE Trans Geosci Remote Sens 43:466–479
10. Serra J (1982) Image analysis and mathematical morphology. Academic Press, New York