Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 96 (2016) 540 – 549

Procedia Computer Science

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016, 5-7 September 2016, York, United Kingdom

OaaS based on temporal partitioning with minimum energy consumption

Emna Hosni[a*], Zaki Brahmi[b]

[a]*Manouba University,RIADI-GD Laboratory, Manouba, Tunisia*
[b]*Manouba University,RIADI-GD Laboratory, Manouba, Tunisia*

Abstract

Cloud computing is an economical solution for industry which is highly scalable and useful of virtualized resources that can be used on demand. It will have a significant impact on companies with the introduction of orchestration platforms as a Service (OaaS) to perform the services that support a variety of business processes such as BPEL. It's becoming an adoptable technology for many of the organizations, thanks to its flexibility and because it reduces total cost of ownership. Thus, an effective OaaS must meet several requests simultaneously; ensuring scalability and optimizing the use of shared resources in order to minimize energy consumption. In this paper, we will investigate three issues i) exploiting the minimum of resources to execute a maximum number of processes, ii) Preventing possible overload to the server, and iii) minimizing dynamic energy consumption which becomes one of the main challenges for large-scale computing, such as in cloud data center. As a solution for these challenges, we propose to use Workflow partitioning technique and this based on temporal dynamic reconfiguration approach. Our work aims to reduce the dynamic energy consumption; especially in communication buffers between partitions of BPEL process during partitioning. The proposed approach is based on two main steps: 1) Estimate the energy consumption of BPEL processes 2) Temporal and dynamic partitioning of BPEL process based on reconfigurable architecture in order to minimize overall energy consumption on each BPEL process.

Peer-review under responsibility of KES International

Keywords: OaaS; BPEL process; Dynamic energy consumption; Temporal partitioning; Buffers.

1. Introduction

With the rapid development of processing and storage technologies, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called cloud computing [12]. This model has been raised as a cost effective solution for

* **Corresponding author. Tel.: +216-95-966-999**
E-mail address: **emnahos@gmail.com**

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2016.08.232

businesses, especially small and medium enterprises (SMEs). Its main features are scalability, resources on demand, virtualization, pay-per use model and multi-tenancy. The integration of different information systems, services and cost saver have been a crucial challenge for companies. However, Cloud Computing provides a solution that satisfies this need; OaaS. The orchestration as a service is a new term that allows outsourcing Information Technology (IT) systems and services into Cloud infrastructures that can guarantee a quality of services that the original enterprises could not provide. Hence, enterprises can deploy its workfows and execute on demand without sacrificing the functionality and reliability they are accustomed to from traditional architecture. The main advantage is that companies can concentrate on their core business and they do not have to pay for infrastructure, its installation and maintenance. OaaS requires an efficient scalable orchestration engines; the less the customers must pay, the more they can be served, and it is an economical solution for company thanks to reducing total cost of ownership. There are few studies in which the authors have paid attention to the problem of efficient orchestration engines like the Bis-Grid Engine [6], the Guarana RT [5] and TPBPEL [3]. In this paper we propose an OaaS based on temporal dynamic recon- figuration approach. The proposed approach is based on two main steps: 1) the estimation of energy consumption of a BPEL process: That determines the energy consumption of a process which refers to the total power consumed by hardware resources. 2) Apply the temporal partitioning algorithm based on energy efficiency: This step allows partitioning process while minimizing the energy produced by communication buffers during the partitioning process. The temporal partitioning algorithm used in this work is based on dynamic reconfigurable architecture that has gained much popularity among the scientists because of its flexibility and high performance. This algorithm is based on the method of max-flow/min-cut that provides the best cut with the minimum energy consumption between partitions. This temporal partitioning algorithm can be very useful in Cloud field, since it can adapt to real-time applications in the reconfigurable architecture.
The remainder of this paper is organized as follows: in Section 2, we make a study of the related work. We focus in Section 3 on the basic concepts used and the formulation of the problem of temporal dynamic reconfiguration. Then, we present our orchestration engine in Section 4. Section 5 shows the evaluation of our approach. Finally, we conclude.

2. Related Work

There are a few numbers of papers in which the authors have paid attention to the problem of designing efficient orchestration engines. In the next section, we will present some of these approaches but also we will consider the related works on reconfigurable partitioning approaches.

Orchestration Engines

The goal of Bis-Grid [6] is to combine BPEL (The Business Process Execution Language) standard for orchestration services and network technologies with comprehensive security mechanism to provide secure integration platforms that uses orchestration as a service. In [5], the authors propose an orchestration engine called Guarana RT. This sub- sequently assembled engine is an effective implementation that uses a configurable thread pool that works at the granularity of tasks; Instead of allocating threads to process instances, it allocates threads to individual tasks inside processes. We notice that the missing things in [6,5] is neglecting some important features in cloud field especially scalability, dynamicity and energy consumption. the Bis-Grid [5] creates an instance in the Active BPEL engine which affects in turns a thread for each instance so, the orchestration between various tasks of a process is centralized. The orchestrator can easily become a bottleneck when the number of instances increases in performance which is highly handled and hence, the response time becomes longer. Guarana RT [5] cannot be executed in a dynamic environment that means this orchestration engine is not able to maintain its functionality in case of high demand at the execution time of BPEL. Guarana RT is not effective for the cloud environment. However, TPBPEL [3] orchestration engine is based on the temporal reconfiguration through partitioning of BPEL process. This architecture requires buffers to store data between partitions during the partitioning. In the case of insufficient resources and high demand, TPBPEL produces extra energy to partition BPEL. In order to minimize this extra energy, we propose an OaaS that maintains the energy consumption produced during temporal partitioning. To achieve this goal, it's necessary to choose the right temporal partitioning algorithm.

The Temporal Reconfiguration Approach

The temporal partitioning problem based on reconfigurable architectures deserves more attention in reconfigurable computing. In [1,2], the authors showed a temporal partitioning algorithm by using the mathematical method based on the eigenvectors of graph to find the partitioning with a minimum number of partitions. The aim of partitioning algorithm in [1] is to reduce the communication cost between partitions for full reconfigurable architecture. The authors have not taken into account the energy consumption constraints. However, in [2] the authors have applied temporal partitioning in partially reconfigurable environment to optimize the latency in the reconfigurable unit. In [10], the authors proposed a temporal partitioning algorithm which is based on architectural synthesis method. This algorithm is used to reuse shared functional units. However, the results of this synthesis (Inter Partition Synthesis) can be different. If we consider a high level of granularity the synthesis process (IPS) is effective for optimizing the additional resources. This requires less additional resources. If we consider a low level of granularity, this needs more additional resources. However, the partitioning level based on architectural synthesis may not be effective, which generates an additional energy consumption. In [7], the authors used a list of minimum cuts of various sizes via the mixed min-cut graph method. This method aims to extend the partitioning process by iterative search for minimum weight paths. Indeed, this partitioning algorithm requires match time to find the final partitioning which produce extra energy. So, the authors do not focus on energy produced during partitioning. However, in [9] the authors present a temporal partitioning algorithm for an embedded system in which its energy consumption is correctly evaluated. The authors have used the temporal partitioning approach to divide the application into temporal partitions, which are configured one after another on the dynamically reconfigurable FPGA device (DRFPGA). This algorithm satisfies the temporal constraints for the reconfigurable architecture. This algorithm uses a sequence of bi-partitioning based on max-flow/min-cut [4] methods. This approach is effective in an elastic environment. Indeed, it's useful for real-time applications. So, this algorithm can maintain the changes without losing any time to resume the entire algorithm. Thus, the energy consumption constraint is a critical factor for the Cloud. This is our main motivation to use this algorithm in our context.

3. Problem Statement

Definition 1 (BPEL process): is a quadratic BPEL = (PL, Var, O, C) where Var is used to define data relating to the internal state of the process and transform messages with web services invoked. Orchestration O gives the definition of the process in terms of the different types of activities that offer BPEL. The cost of energy consumption C represents the amount of resources requested by a process. We propose the following formula to calculate the cost of energy consumed by a BPEL process P_i.

$$CE(P_i) = \sum_{i=1}^{m} ni \times CE_{ai} \quad (1)$$

Where m denotes the different types of activities a_i in the process P_i. n is the number of activities having type m, CE_{ai} refers to the cost of energy consumption. The types of the structural activities are: {*Sequence, Flow, if, while, repeat until...*}. The types of the basic activities are: {*receive, invoke, reply,assign...*}.

Definition 2: The temporal reconfiguration problem of BPEL process (TRPP) is to execute a set of processes BPEL $P=\{P_1, P_2...P_n\}$ on a Cloud server S while respecting the maximum capacity of server U_{max} and the minimum capacity of server U_{min} at time t_i . Each sub-process admits a cost represented by w(x). Formally, we can present the TRPP problem as follows:

$$PRTP =< P, S , w(x) >, S = \{U_{max}, U_{min}\} \quad (2)$$

Definition 3 (Partitioning problem): An s-t cut $(X,\bar{X})$ of a flow network which is a partition of V into two separate parts such as $s \in X$ and $t \in \bar{X}$ then $C = s \cup t$. The capacity or size of a cut $(X,\bar{X})$ is the sum of the capacities on the forward cut-edges. The well-known max-flow/min-cut theorem [4] says that the value of a maximum flow is equal to the capacity of a minimum cut (or min-cut for short) on the flow network.

To deduce the additional power consumption caused by the buffers during the partitioning, we use the following theorem.
Theorem: The additional power consumption incurred by the buffers inserted in partitioning is equal to the total capacity of the forward cut-edges[9].
In our context, during partitioning of the workflow the communication buffers used (b) between partitions produce the extra energy that is equal to the total data capacity transfer of the previous partition which contain activities (a_1,...a_n). The energy consumption (EC) of buffers is presented in Fig.1. At this level, we present the

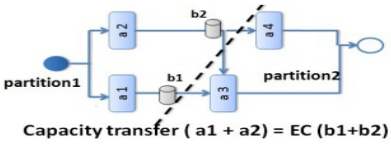

Fig. 1. **Energy consumption of buffers**

partitioning constraints used in this work.

- Precedence constraints[7]: Constraints define a temporal ordering on the node in G and are called precedence constraints. For two nodes v and u, we define P (v) ≤ P (u), v must be scheduled before u when partitioning.
- Area constraints: A temporal partitioning is feasible in accordance to a reconfigurable device if the following condition is verified:

$$\forall X_i \in X; \quad U_{min} \leq w(X_i) \leq U_{max} \tag{3}$$

Where $w(X_i)$ denoted the area of partition X_i, U_{max} denotes the maximum resource capacity of a reconfigurable unit and U_{min} denotes the minimum resource capacity of a reconfigurable unit. The area of partition X equals the area of vertices v_i belong to partition X_i and $w(v_i)$ denotes the area of v_i.

$$W(X_i) = \sum_{vi \in Xi} w(v_i) \tag{4}$$

Based on **(3)** and **(4)** to satisfy the area constraint.

$$\forall X_i \in X; \quad U_{min} \leq \sum_{vi \in Xi} w(v_i) \leq U_{max} \tag{5}$$

The balanced area of the device.

$$(U_{min} + U_{max})/2 \tag{6}$$

- Capacity constraint [9]: The capacity of a cut or *Size(c)* denotes the number of buffers inserted in the partition K. R_{max} is the upper bound on the size of partitions. The capacity constraint is defined when the following condition is satisfied:

$$Size(c) \leq R_{max} \tag{7}$$

4. Orchestration Engine as a service based on Temporal Reconfigurable Architecture

In this section, we propose an orchestration engine as a service in the Cloud (OaaS) based on temporal partitioning that minimizes energy consumption. Our approach is based on two ideas:

1. Estimation of the energy consumption of a BPEL process: this step determines that the energy consumption is the total power consumed by hardware resources.
2. Apply the temporal partitioning algorithm based on energy efficiency: This step allows the partitioning process while minimizing the energy caused by communication buffers in the server.

Estimation of the energy consumption of a BPEL process

In this paper we estimate the aggregation rules to infer the energy consumed of BPEL process, inspired from the aggregation rules presented in [8]. For a sub-process (S p) that contains a sequence of successive activities, we calculate the sum of energy caused by (a1,a2), for a sub-process Flow which provides a parallelism among activities, we suppose that the request can only be routed to the a1,a2, or simultaneously, to both. Each routing occurrence has a known probability *por1*, *por2* and *por* // [8]. For each sub-process which contains an if-type activity, we assume a probability *pi* for a request to be routed towards one sub-process that contain (a^1, . . , a^n). For a loop sub-process, we presume that there is a maximal number of loops nl and a probability pli to loop i times. The energy consumption by process Pi respecting all types of activities is shown Algorithm **1**.

Algorithm 1 Algorithm CE_p^i

Input: $\mathbf{P_i}=\{\mathbf{a_1^m}, \mathbf{a_2^m} \ldots \mathbf{a_n^m}\}$

Output: CE_p^i = The energy consumed by a process p_i.

begin
for each activity a^m do
$a^{m-basic} \leftarrow$ *Receive* ∪ *invoke* ∪ *reply* ∪ *assign*
$a^{m-static} \leftarrow$ *Seq* ∪ *flow* ∪ *if* ∪ *while* ∪ *pick*
$CE_p^i \leftarrow 0$
if $a^m \in a^{m-basic}$ then
E (m-basic) ni × $\mathrm{CE_{m-basic}}$
$CE_p^i \leftarrow CE_p^i +$ CE(m-basic)
else if $a^m \in$ *Seq* then
E (Seq) ← ni × $\mathrm{CE_{Seq}}$
$CE_p^i \leftarrow CE_p^i +$ CE(Seq)
else if $a^m \in$ *flow* then
E (Seq) ← ni × ($Por1 \times CE_{ai} + Por2 \times CE_{ai} + Por \| \times CE_{ai}$
$CE_p^i \leftarrow CE_p^i +$ CE (*flow*)
else if $a^m \in$ *if* then
E (if) ← ni ×($P \times \mathrm{CE_{if}}$)
$CE_p^i \leftarrow CE_p^i +$ CE (*if*)
else if $a^m \in$ *while* then
E (*while*) ← ni × ($pli \times \mathrm{CE_{while}}$)
$CE_p^i \leftarrow CE_p^i +$ CE (*while*)
else if $a^m \in$ *pick* then
E (*pick*) ← ni × ($P \times \mathrm{CE_{pick}}$)
$CE_p^i \leftarrow CE_p^i +$ CE (*pick)*
end if
$CE_p^i = \sum_{i=1}^{am} CE_{ai}$
end for
end

Temporal partitioning based on constraint energy efficiency

In order to partition a BPEL process, we have inspired our approach from the work presented in [9], while respecting the specificity of our research work. Since this algorithm reduces the energy consumption in real time as respecting the area and capacity constraints. In [9], the authors use as input a Data Flow Graph. However, The BPEL process is a specific programming language. So, we have to use a BPEL Flow Graph (BFG) in order to treat effectively the characteristics of BPEL. We used the intermediate modeling (BFG) 11. Then, we analyze the energy caused by the buffers. Next, we apply the temporal partitioning algorithm which reduces the overall energy consumption without exceeding the buffer number limit. Finally we present an illustrative example.

The intermediate representation

BPEL Flow Graph is an extension of Control Flow Graph which is used to represent a BPEL program in a graphical mode. BFG does not only contain information structure, but also specifies all the information about control flow of BPEL program, data flow information and semantic information such as dead paths. This method is well modularized to support different phases of BPEL testing [11].

The energy consumption caused by the buffers at temporal partitioning

Our paper is mainly concerned with the dynamic power consumption model which corresponds to the energy consumed during a transition of data at time t_i. Indeed, we have adopted the same philosophy presented in [9] in order to reduce extra energy during partitioning. To minimize energy consumption, we have taken into account the capacity transfer of each activity when partitioning in the reconfigurable architecture. The additional power consumption is proportional to the capacity of transferring data of activities that feeds the output signal to the inserted buffer that is associated with cut $(X,\bar{X})$. We suppose that the energy consumed by the activity is $E(ai)$, the capacity transfer is $cap_{(ai,j)}$ and the capacity of the inserted buffers is $cap_{(bi)}$. For each activity, we associate a bi-dimensional 0-1 variable y_{ij} such that:

$\boldsymbol{y_{ij}} \in \{\mathbf{0, 1}\}$, $1 if a_i$ is an executed activity**, 0** otherwise.

We assume that the additional energy consumption caused by the buffer is proportional to the transfer capacity between activities (a_i, a_j) such that$(a_j \in X)$ and $(a_j \in \bar{X})$. The buffers b_i receiving the output data of a partition consume energy as follows:

$$CE^{+} = [E_{(ai)} \times cap_{(bi)} \times \sum_{i=1}^{n} cap_{(ai,j)}] \cdot \boldsymbol{y_{ij}} \quad \textbf{(8)}$$

Temporal Partitioning BPEL

The temporal partitioning algorithm is used to find a partitioning with an optimal number of partitions (K) where the energy consumption has the lowest value with a minimum number of buffers used. The algorithm is composed of two steps: The first is to find an initial partition of the graph. This step gives an optimal solution in terms of energy consumption using ***max flow/min-cut*** method. Then, if the area constraint and the upper bound on the size of a partition are satisfied, we adapt the initial partitioning. The aim of the second step is to find the final partitioning of the graph that satisfies both the area and capacity constraints. This step is recursively partitioned until the number of sub-process is equal to K. Every sub-process is loaded into one stage scheduled. If the second step cannot find a feasible scheduling then we relax the number of partitions by one and the algorithm goes to the first step, and then we restart to find a feasible solution with the new number of partitions. The function of ***transformBFG(P_i)*** is to transforms each process P_i into a BPEL Flow Graph G_i. ***CE^{+}()*** aims to calculate the additional energy consumption caused by the inserted buffers between partitions while respecting the types of activities (see Algorithm1). The function ***Sum(Size(c))*** returns the number of buffers obtained by cutting. ***Nb- buffer*** is used to store the total number of buffers that have been calculated by ***Sum(Size(c))***. $\boldsymbol{R_{max}}$ is the upper bound on the cut size in partition; $\boldsymbol{R_{max}}$ denotes the great number of buffers that can be inserted between two partitions. K is the minimum number of partitions. $\boldsymbol{C_{min}}$ is used to store all the graphs that have been partitioned with minimum energy consumption**. w(x)** denotes is the total area of all nodes in X. The temporal partitioning algorithm based on constraint energy efficiency is shown Algorithm **2**.

Algorithm 2 Algorithm PT-MinEC-BPEL

Input: $P = \{P_1, P_2 \ldots P_n\}$ U_{max}: maximum capacity of server, U_{min}: minimum capacity of server, R_{max}: maximum size of a partition.

Output: C_{min}

begin

Cmin $\leftarrow \emptyset$

Nb_buffer $\leftarrow 0$

1.**For each process** $P \in P_i$ do

Gi $\leftarrow trans\_for\_BFG(Pi)$

2.For each G_i do

K$\leftarrow (CE_p^i(\text{Gi}) / ((U_{min} + U_{max})/2)$

3. **repeat**

Cmin $\leftarrow max\_flow/mincut(G_i)$

K $\leftarrow K - 1$

if $(Size(c) > R_{max})$ **then**

Return ***Cmin***

if $(U_{min} \leq w(x) \leq U_{max})$ **then**

if $(CE^+(C) < CE^+(C_{min}))$ **then**

$C_{min} \leftarrow C$

end if

$Nb\_buffer \leftarrow Sum(Size(c))$

Until $(K \leq 1)$

$C_{min} \leftarrow C_{min} \cup C_{min}(G_i)$

return at the stage 2

end for

end for

end

Example

The following example explains the proposed engine mechanism. We assume that the server S allows a maximum resource capacity 5000. At time t_0, the server is running 10 processes. A new BPEL process is executed in the server S. But at this moment, the server does allow a maximum capacity U_{max} which is equal to 3000 and a minimum capacity is U_{min} =10. The new process is shown in fig.2.This process

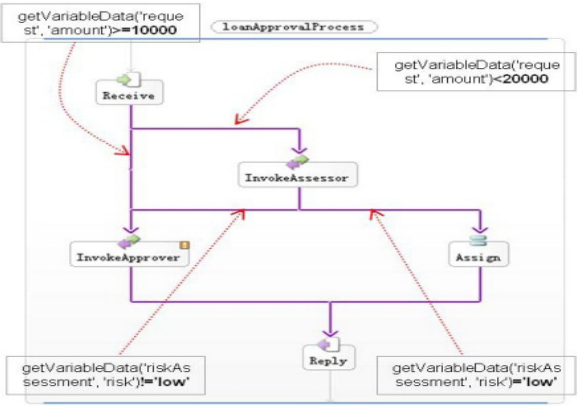

Fig. 2. Example of BPEL program of Loan approval process [11].

begins with the receipt of a loan application. For small amounts (less than $ 20,000) and those at low risk, approval is automatic. For large amounts or high-risk individuals, each credit request must be examined in more detail. The use of risk assessment

and loan approval services are represented by elements of invoke. Note that the transition conditions granted to determine the links active links, all join conditions using the default setting. Finally, the process responds with a"loan approved" message or a"loan rejected". For this, at time t_1 server capacity is insufficient to run the new process. To apply our algorithm, the new process should be transformed into a BPEL Flow Graph. In order to check the effectiveness of our approach we presented two partitioning models illustrating BFG as shown in Fig.3.

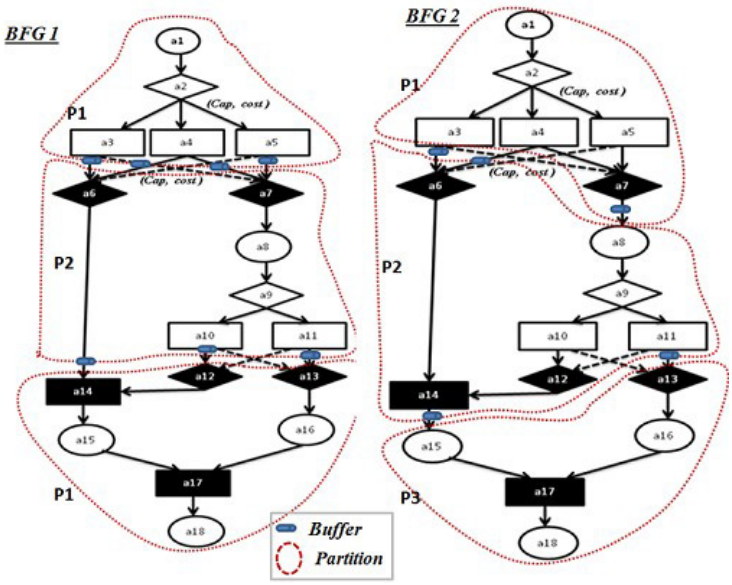

Fig. 3. **Temporal partitioning of Bpel Flow Graph (BFG) during the temporal partitioning.**

The cost of each partition must be less or equal to 3000. We assume for BFG1 the energy consumption *(EC)* of partitions P_1, P_2, P_3 is equal to 800, 1000, 3200 respectively, and for BFG2 energy consumption of the partitions P_1, P_2, P_3 is equal to 1100, 3000, 900 respectively. For BFG1, we need 7 buffers for sending data between partitions. However, for BFG2 only 5 buffers are needed when partitioning. Therefore, to reduce the extra energy caused by buffers, the partitioning associated with BFG2 is the best one because it reduces the energy consumption with a minimum number of buffers.

5. Complexity Analysis

The complexity of our approach depends on:

- The complexity of temporal partitioning algorithm:
  In [9], the authors have a time complexity equal to $\boldsymbol{O(|V||E|)}$ in order to find a feasible cut with minimum energy consumption.
- The complexity of the BPEL transformation BFG, for this transformation method, the authors in [11] can estimate the worst case complexity of node explosion in BFG, which depends on the activities that have more than one outgoing link. Assume in a given BPEL file, there are $\boldsymbol{m}$ activities that have more than one outgoing link and each activity has Ai outgoing links, then the worst-case complexity of nodes number explosion is:
  $O\ (2^{A1+A2+\cdots Am})$
  But in most situations, the worst case will not hold and the transformed result BFG will not have so many nodes for two reasons; 1) in the worst case, each BPEL activity is assumed to be transformed to execution scenarios. But in most cases, the scenarios can be reduced greatly by removing those invalid ones. 2) In the

worst case, the nodes number is calculated by multiplication. But in most cases, the number is calculated by addition [11]. The following equation calculates the complexity of our approach :

$$C = O\,(O(|V||E|) + O(2^{NA})) \times n = 2^{NA}$$

NA represents the *m* activities that have more than one outgoing link.

6. Implementation and Evaluation

The experiments conducted in this work were performed on a 6 GB of RAM with an Intel Core CPU frequency I5- 4200U 2.30 GHz Windows 7. We used our temporal partitioning algorithm (PT MINEC BPEL) to solve every tests problem. The size of the problem is depended on the number of activities in the BPEL process. We vary the size of BPEL program in order to generate a necessary number of buffers for each BPEL at run time. Fig. A shows the evolution of our approach based on the number of buffers. The extra

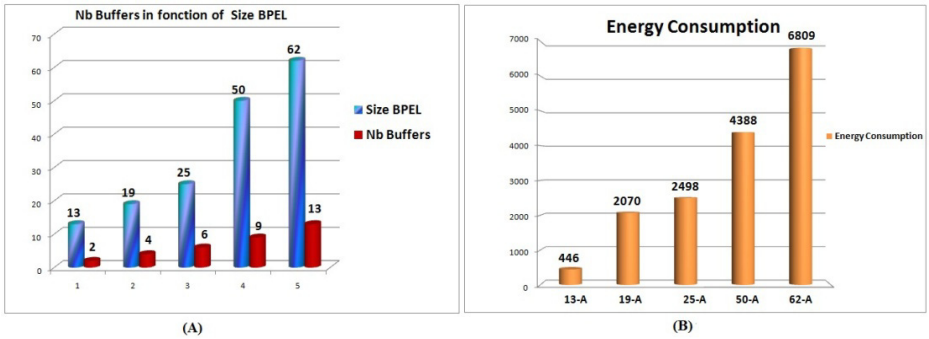

Fig. 4. **(A) The number of buffers in size function of BPEL program during the temporal partitioning, (B) The Energy consumption produced during the temporal partitioning of each size BPEL.**

energy depends on the number of buffers which are responsible for storing and transitioning the intermediate data when partitioning. Then, the data must be buffered from partition to another in order to minimize the data transfer cost and the cost of energy produced of BPEL program, our algorithm finds the appropriate partitioning which minimizes the energy consumption caused during the execution of BPEL. So the energy consumption of each size of problem is shown in Fig.B. In the test, it can be seen from the results in Fig. A and Fig.B that the process **50 A** and **62 A** generated a minimum number of buffers which varies between **2** and **4** respectively, with **4388** and **6809** as a cost of energy consumption. According to the results of the above experiments, we can conclude that the energy consumption slowly increases when the problem size increases. In addition, the reconfigurable architecture of our orchestration engine requires the use of buffers that dissipate the additional energy, which implies an increase in response time as well as data processing and storage data. For this we have partitioned the workflow with a minimum number of used buffers which reduces thereafter the energy produced during transition data. We show that the energy consumption and the data transport represent a significant percentage of energy consumption in Cloud Computing. Thus, we have reduced the total energy consumption that represents the high performance of our approach.

7. Conclusion and Future Works

In our work, we have investigated an orchestration as a cloud service (OaaS) that is based on a reconfigurable partitioning BPEL processes, and in order to exploit the available resources to execute a maximum number of processes and prevent possible overload to the server. The aim of our work is to reduce dynamic energy consumption caused by buffers during partitioning. Experimental results have shown that

the PT_Min_EC_BPEL algorithm can effectively reduce the power consumption without exceeding the buffer number limit, we can generalize our work to another type of workflow and especially intensive workflows. In our future works we will focus on the access control structure to save confidential data for the competing companies which execute their workflow in the same cloud server.

References

1. Ramzi Ayadi, Bouraoui Ouni, and Abdellatif Mtibaa, *A partitioning methodology that optimizes the communication cost for reconfigurable computing systems*, International Journal of Automation and Computing **9** (2012), no. 3, 280–287.
2. Ramzi Ayadi, Bouraoui Ouni, and Abdellatif Mtibaa . *Integrated temporal partitioning and partial reconfiguration techniques for design latency improvement*, Evolving Systems **5** (2014), no. 2, 133–141.
3. Zaki Brahmi and Chaima Gharbi, *Temporal reconfiguration-based orchestration engine in the cloud computing*, Business Information Systems, Springer, 2014, pp. 73–85.
4. LR Ford and Delbert Ray Fulkerson, *Flows in networks*, vol. 1962, Princeton Princeton University Press, 1962.
5. Rafael Z Frantz, Rafael Corchuelo, and Jose´ Luis Arjona, *An efficient orchestration engine for the cloud*, Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, IEEE, 2011, pp. 711–716.
6. André Höing, Guido Scherp, Stefan Gudenkauf, Dirk Meister, and André Brinkmann, *An orchestration as a service infrastruc- ture using grid technologies and ws-bpel*, Service-Oriented Computing, Springer, 2009, pp. 301–315.
7. Yung-Chuan Jiang and Jhing-Fa Wang, *Temporal partitioning data flow graphs for dynamically reconfigurable computing*, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **15** (2007), no. 12, 1351–1361.
8. Yanik Ngoko, Alfredo Goldman, and Dejan Milojicic, *Service selection in web service compositions optimizing energy consumption and service response time*, Journal of Internet Services and Applications **4** (2013), no. 1, 1–12.
9. Tzu-Chiang Tai and Yen-Tai Lai, *Power minimization for dynamically reconfigurable fpga partitioning*, ACM Transactions on Embedded Computing Systems (TECS) **12** (2013), no. 1s, 52.
10. LIU Ting, *Optimisation par synthese architecturale des methodes de partitionnement temporel pour les circuits reconfigurables*, Ph.D. thesis, These de doctorat, These de doctorat, UHP-Universite Henri Poincare, 2008.
11. Yuan Yuan, Zhongjie Li, and Wei Sun, *A graph-search based approach to bpel4ws test generation*, Software Engineering Advances, International Conference on, IEEE, 2006, pp. 14–14.
12. Qi Zhang, Lu Cheng, and Raouf Boutaba, *Cloud computing: state-of-the-art and research challenges*, Journal of internet services and applications **1** (2010), no. 1, 7–18.