Adaptive and Secure Application Partitioning for Offloading in Mobile Cloud Computing

N.M. Dhanya[1(✉)] and G. Kousalya[2]

[1] Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore 641112, India
dhanyahari07@gmail.com

[2] Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore 641014, India
kousir@gmail.com

Abstract. Smart phones are capable of providing smart services to the users very similar to laptops and desktop computers. Despite of all these capabilities battery life and computational capabilities are still lacking. By combining mobiles with cloud will reduce all these disadvantages because cloud is having infinite resources for processing. But in cloud security is a major concern. Since mobile devices contain private data a secure offloading of application is necessary. In this paper we are proposing a secure partitioning of application so that the most sensitive or vulnerable part of the application can be kept in the mobile and rest of the application can be offloaded to the cloud.

1 Introduction

The introduction of cloud computing leads to a new era in utility computing, where the computing power is offered as a service [1]. The combination of Mobile technology with cloud computing radically changed the perspective of distributed application processing in mobile devices. The benefits that can be utilized by the end user are reduced cost, improved performance and higher scalability. The mobile devices having resource intensive components which will affect the battery charge. But we can't even offload the entire application to the cloud also. The total application offloading is not possible because majority of the applications are using the local resources like global positioning system, camera and other sensors [2]. So for offloading the application to cloud the application should be spitted into different groups according to some granularity levels. The parts which are highly resource intensive are offloaded to the cloud and system dependent parts are executed at the mobile device itself. The implementation of above model introduces the problem of communication cost between the components that reside at the mobile side and the components at the cloud side.

The deployment should be optimized based on both the mobile user and cloud provider perspective. All the remote components should be deployed to a virtual machine with enough CPU power while reducing the communication cost between the local and the remote components. One more constraint that is introduced is the security of the parts which are offloading. The partitioning should be such a way that the most

J.H. Abawajy et al. (Eds): SSCC 2015, CCIS 536, pp. 45–53, 2015.
DOI: 10.1007/978-3-319-22915-7_5

vulnerable parts are retained back in the mobile and the rest of the parts are offloaded to the cloud for remote execution. The major contribution of this paper include

1. We present algorithms to partition an application consisting of interconnected components which will reduce the communication cost between cloud and the mobile and at the same time reduces the security risk incurred due to offloading.
2. Considering the bandwidth as a dynamic variable an adaptive factor is included in the algorithm

2 Related Work

Graph partitioning is one of the fundamental issues in many domain such as task allocation on grid, network partitioning and VLSI design. The graph partitioning deals with cutting the graph into k parts while minimizing the number of edges that are cut. If the k = 2 it is called bi-partitioning which is needed in our problem. Finding a solution for this problem is NP-Hard [3] in nature. Kumar et al. [4] describes a series of offloading strategy and has done an extensive survey on the offloading strategy existing. Kernighan-Lin [5] One of the most popular local graph partitioning algorithm is Kernighan-Lin algorithm. The K-L algorithm starts with the partitioning of vertices in a balance manner that is each partition contain equal number of nodes. It proceeds in a series of passes. During each pass, the algorithm improves the initial solution by swapping pairs of vertices to create a new solution which is having smaller edge cuts if any. This process is repeated until no more subsets are found. This is the local minimum and no improvements can be done to this solution.

A simple min-cut algorithm by Stoer-Wagner [6] proposes a graph cutting those edges with minimum weights. This is one of the fastest algorithm. It works in different phases and at each phase it identifies a pair of vertices for min-cut. Multilevel k-way Partitioning Scheme for Irregular Graphs - Karypis and Kumar [7] introduced a new multilevel graph partitioning technique. It consists of three phases like coarsening, partitioning and un-coarsening. A graph G is initially coarsened into a smaller graph and then bisection is done on that graph. Then the coarsened graph is expanded back to the original graph by periodically refining the partition. Since this decreases the edge cut this is one of the popular methods used.

Graph partitioning - A survey- Ulrich Elsner [8]: this paper describes in detail about all the graph partitioning strategies. Hunt and Scott introduced an automatic distributed partitioning system called Coign, [12] which automatically transforms a program into distributed applications without accessing the source codes. Coign constructs a graph model of the application's inter-component communication through scenario-based profiling to find the best distribution. Later a graph cutting algorithm is executed to partition the application across the network. One of the advantages of this method is that an end user without source code can change the application into distributed applications.

The paper [13] deals with partitioning of programs in resources constrained devices to enable them to do distributed processing. This also deals with multiple constraints such as minimize energy and maximize performance. Here they are converting the application into Object Relational Graph (ORG) using static analysis and dynamic profiling. Later it is converted into Target Graph. This allows for uniform strategy for different constraints. Ou et al. [10] proposed a class level application partitioning strategy which improves the performance of offloading. It considers a multi-cost graph based model which is implemented in Java. It uses an online decision making strategy for splitting and offloading. the work is based on Heavy Edge Light Vertex Matching algorithm (HELVM). Expensive adaptation decision is the major disadvantage of this method. The work done by Tim et al. [14] proposes an algorithm which minimizes the time required for application execution. They are using Heavy Edge Matching (HEM) for coarsening and K-L for refinement. They uses Java and Adhoc framework for simulation.

3 Problem Statement

In our method the mobile application which is to be offloaded is partitioned into two for local and remote execution. First of all the application should be converted into graph called Weighted Object Relational Graph (WORG). Then taking the bandwidth changes into consideration the graph is broke dynamically. We are making the partition strategy as minimum data transfer and maximum security. So that the application is executed in a distributed manner with minimum transfer cost.

The Fig. 1 shows the architecture diagram for the proposed work. The Weighed object relational graph is calculated for the mobile application using the code analysis. Then the application is partitioned into two parts cloud partition and a local partition using genetic algorithm. The dynamic bandwidth due to mobility of the mobile node is also taken into consideration. Whenever there is a major change in the bandwidth correspondingly the partitions are changed.

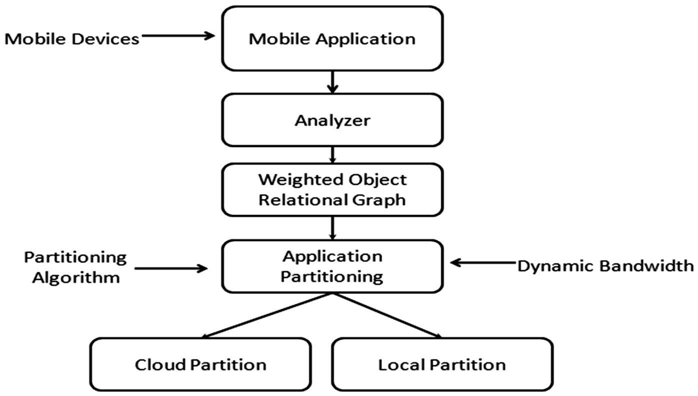

Fig. 1. System architecture

3.1 Weighted Object Relational Graph (WORG)

The application is converted into Weighted Object Relational Graph by using Soot Analysis frame work [9]. Figure 2 shows a sample WORG of a face recognition application. Each node is represented as a circle with weight which represents the memory cost for that node. En edge connecting two nodes represents the communication between the nodes with the corresponding costs. Static analysis and dynamic profiling techniques are used to create the WORG from the application. The example shows four classes Face preview, Image capture, Face Detection and Face detection library, where the face preview is the main class and from that image capture and face detection are called.

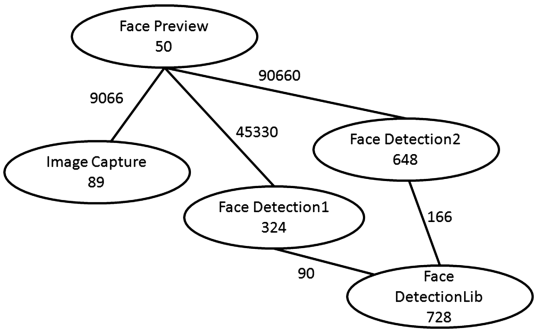

Fig. 2. Weighed object relational graph (WORG)

3.2 Application Partitioning Algorithm

Let a graph G = (V, E) is the application graph where G is the vertex and E is the edges between the vertices. Each vertex is represented by a cost V_i, which depicts the cost of the vertex including the CPU cost, Memory cost and vulnerability factor for each node. The vector is represented as W_v. The edge weight represents the communication cost between the components. The adjacency matrix represented by C_{ij}, the value of communication cost(bandwidth) between node V_i and V_j. If there is any communication cost between V_i and V_j, Cij represents the weight otherwise $C_{ij} = 0$. This is represented by a vector W_e. If in the graph we have n nodes and m vertices W_v is a vector of length *n and Cij* is a matrix of *n X n*. The vulnerability factor for each node can be represented as $S_{i.}$

The basic idea in security parameter is during offloading the app is divided in such a way that keep the components that have the biggest impact on the vulnerability to the local mobile itself. The vulnerability index depends on various features like call relationships, propagated vulnerabilities, cloud originated vulnerabilities and so on. Here we assume random values as vulnerability index for each node. In the optimized partitioning k modules are offloaded to the cloud. Our problem can be defined as an objective function *min (transmission cost)* + *max (security)*

Transmission cost: ***Minimize*** $\sum_{i=1}^{n}\sum_{j=1}^{k} C_{ij} \times \left(x_i\, XOR\, x_j\right) \times b$ where $x_i = 0$ if node i is executed on the local machine (mobile) and is equal to 1 if executed on remote machine.

Security cost: ***Minimize*** $\sum_{i=1}^{n} S_i \times (1 - x_i)$Which will reduce the security risk of moving the more vulnerable nodes to the cloud. Here the security is preserved by moving less vulnerable nodes to the cloud. The total cost function can be represented as **Minimize** $\sum_{i=1}^{n}\sum_{j=1}^{k} C_{ij} \times \left(x_i\, XOR\, x_j\right) \times b + \sum_{i=1}^{n} S_i \times (1 - x_i)$

The above equation can be solved by Genetic algorithm is one of the evolutionary methods which can be used for graph partitioning [11]. The graph partitioning can be formulated as a multi-constrained optimization problem where the objective function can be formulated as above equation. The inputs to the algorithm can be *V - Number of nodes n, E - Number of edges, W_v - Node weight vector, W_e - Edge weight vector and S_n - Vulnerability index vector* and the output is formulated as a vector of length n with 0's and 1's where 0 represents local and 1 represent remote execution. The basic genetic algorithm can be stated as below.

The above algorithm works as follows. The optimized partition using genetic algorithm is calculated for the current bandwidth and the previous bandwidth. The following facts are considered for change partition

1. If the bandwidth is increasing the partitions which are there in the cloud remains the same and some of the local components are offloaded to the remote site. That is no change in S and some nodes in C will be shifting to S
2. If the bandwidth is decreasing the partition which is there in the mobile remains the same and some of the remote components are shifted back to the local site. That is no change in C and some nodes in S will be shifting to C.

```
Basic Genetic Algorithm
Input : WORG ( V, E, $S_n$ ,$W_v$, $W_e$ )
Output : Optimal partitioning solution $\overline{X}$ where the local components are represented by 0 and
remote components are represented by 1
Algorithm :
        1. Initialize population $X_0$ with a random vector X with 0's and 1's
        2. Sort( $X_0$ )
        3. While (gen!=maxGen)
                a.  Use selection (Elitsm), mutation and crossover to generate children $Q_i$
                b.  $X_i = X_i \cup Q_i$
                c.  Sort( $X_i$ in the descending order depending on the cost function )
                d.  $X_i = X_i[1..popsize]$
                e.  $X_{i+1} = X_i$
                f.  i++
                g.  gen++
        4. End While
```

```
Bandwidth Adaptive Algorithm
Input : WORG ( V, E, $S_n$ ,$W_v$, $W_e$ ) , b, $b_p$
Output : Optimal partitioning solution $\overline{X}$ where the local components are represented by 0 and remote components are represented by 1
compute the optimized partition ($C_b$,$S_b$) using current bandwidth b using genetic algorithm
Compute the optimized partition ($C_{bp}$,$S_{bp}$) using previous bandwidth bp using genetic algorithm
$V'=V-C_b-S_{bp}$
$m=|V'|$
while $m>1$ do
        $\{V_1,V_2,...V_m\}$=order($V'$,WORG)
        $C=C_b \cup \{V_1,V_2,...V_{m-1}\}$
        $S=\{V_m\} \cup S_{bp}$
        $X=(C,S)$
        if ($b-b_p>threshold$ ) then //Checking condition for change partition
                $\overline{X}=X$
        end
        $V'=V'-V_m$
        $m=m-1$
end
```

The application is partitioned according to the above algorithm and the dynamic and adaptive nature is calculated at each iteration of a bandwidth change. If the bandwidth changes the results are calculated and decision is taken on whether to shift the nodes from mobile to cloud or cloud to mobile depending on the above two conditions.

4 Application and Evaluation

The ORG constructed with the Dacapo Benchmark using soot framework and spark tool. The Table 1 specifies the graph which we considered for our experiments.

The ORG is constructed from the above data and adaptive splitting algorithm is applied onto the graph. The bandwidth is taken from a real time bandwidth trace obtained from a Samsung Galaxy phone.

Energy efficiency of graph partitioning algorithm The following is the evaluation of our algorithm with some standard methods. Stoer-Wagner is the basic graph

Table 1. - Sample application graph data

Programs	Classes	Methods	Nodes	Edges
avora	8812	22240	208	5430
H2	10264	14494	4912	11088
luindex	6530	8199	1027	6199
lusearch	7812	10148	1238	7148

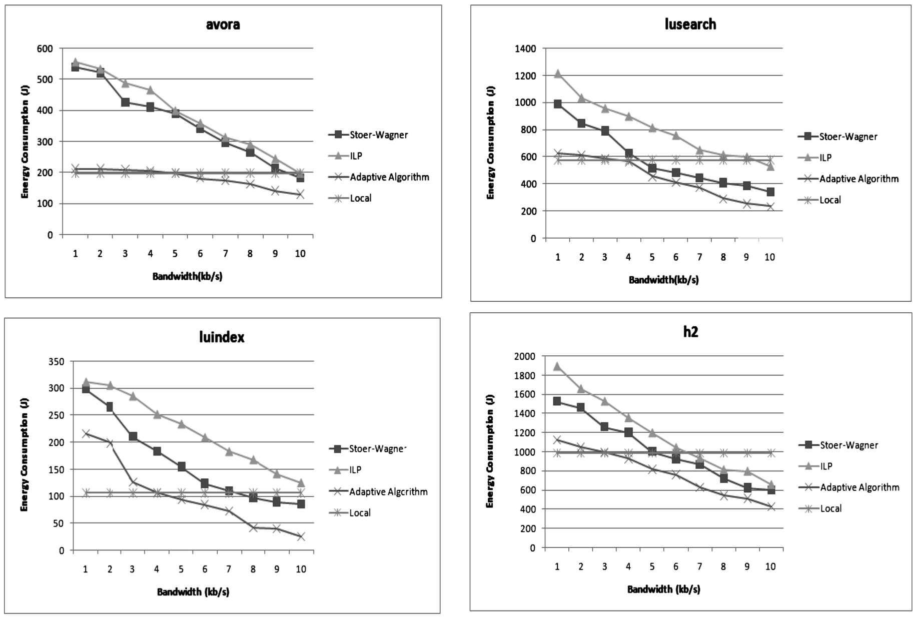

Fig. 3. Energy consumed for various graphs

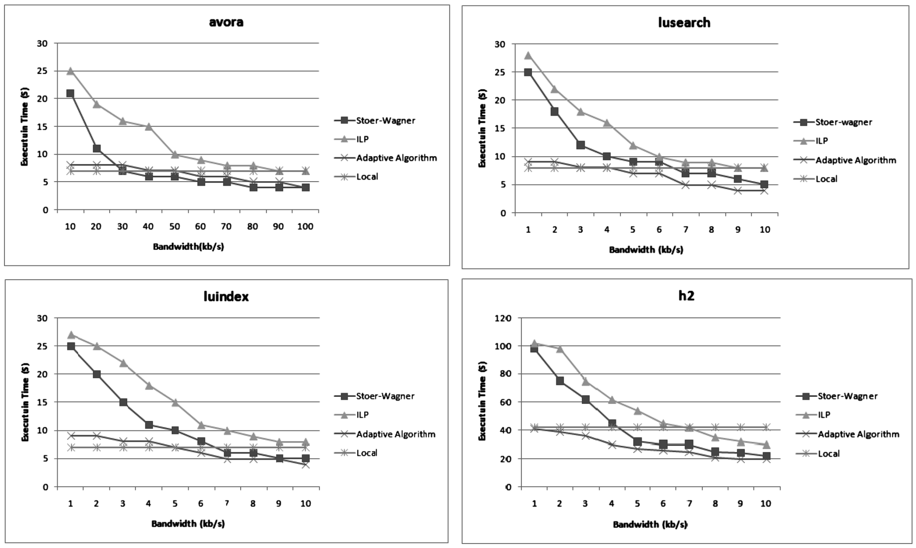

Fig. 4. Execution time for various graphs

bi-partitioning algorithm and an Integer Linear Programming based algorithm where all the constraints are specified. The whole evaluation is calculated on different bandwidth scenarios. The bandwidth parameter we varied from 10-100 and the execution time is plotted against the bandwidth. Local execution is done for all the sample graphs and the execution time is constant for any bandwidth. In all the sample graphs the ILP solution is taking more time because it is making the partition based on all the constraints. The Stoer-Wagner solution is also taking much larger time than the local execution. Energy efficiency is also calculated for the sample graph and our algorithm gives a better result than all other algorithms.

Our algorithm works reasonably well in all graphs at any bandwidth. Even if the adaptive algorithm is not showing better performance at lower bandwidth, higher bandwidth solutions are much better than the local solutions. At lower bandwidth the transmission cost is much larger that is the reason for lower performance. The Figs. 3 and 4 shows all these result .

5 Conclusion and Future Work

In this paper we used a genetic algorithm for graph partitioning and an adaptive algorithm for changing the fragments according to the variations in the bandwidth. Basically bandwidth is one of the most frequently changing components in offloading which will affect the decision a lot. Our algorithm is shifting the nodes dynamically depending on the current situations. Our experimental evaluation shows that the adaptive algorithm works reasonably good than the conventional liner programming approaches.

Our future work is to expand the conditions to more sophisticated values which will consider all aspect of offloading scenario. A real android application will be taken and converted into graph and then partitioning can be done. The vulnerability index will be calculated based on all the parameters.

References

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
2. Pathak, A., Hu, Y.C, Zhang, M., Bahl, P., Wang, Y.M.: S.o.E. Engineering, computer, enabling automatic offloading of resource-intensive smartphone applications. Technical report. United States, Purdue University (2011)
3. Fjallstrom, P.-O.: Algorithms for graph partitioning: a survey. Comput. Inf. Sci. **3**(10), 1–37 (1998)
4. Kumar, K., Liu, J., Lu, Y.H., Bhargava, B.: A survey of computation offloading for mobile systems. J. Mobile Netw. Appl. **18**(1), 129–140 (2013)
5. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**, 291–307 (1970)

6. stoer, M., wagner, F.: A simple min-cut Algorithm. J. ACM (JACM) **44**(4), 585–591 (1997)
7. Karypis, G., kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. **48**(1), 96–129 (1998)
8. Elsner, U.: Graph partitioning - A survey
9. Einarsson, A., Nielsen, J.D.: Soot Framework: A Survivor's Guide to Java Program Analysis with Soot ′ BRICS. Department of Computer Science University of Aarhus, Denmark (2008)
10. Ou, S., Yang, K., Liotta, A.: An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2006), pp. 25–116. IEEE, Pisa, Italy (2006)
11. Farshbaf, M., Feizi-Derakhshi, M.-R.: Multi-objective Optimization of Graph Partitioning using Genetic Algorithm. In: Third International Conference on Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP 2009, 11–16 Oct 2009
12. Hunt, G.C., Scott, M.L.: The coign automatic distributed partitioning system. In: Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI), pp. 187–200, Feb 1999
13. Wang, L., Franz, M.: Automatic Partitioning of Object-Oriented Programs for Resource-Constrained Mobile Devices with Multiple Distribution Objectives Parallel and Distributed Systems. In: ICPADS 2008, 14th IEEE International Conference on 8–10 Dec. 2008, pp. 369 – 376 (2008)
14. Verbelen, T., Stevens, T., De Turck, F.: Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. Future Gener. Comput. Syst. **29**(2), 451–459 (2012)