International Journal of Hybrid Information Technology
Vol.6, No.6 (2013), pp.217-224
http://dx.doi.org/10.14257/ijhit.2013.6.6.19

An Efficiency Optimization Method for Mobile Application with Reconfigurable Software

Yonghua Zhu[1] and Congqi Xia[2]

[1]*Computing Center, Shanghai University, No.99, ShangDa Road, Shanghai, China*
[2]*School of Computer Engineering and Science, Shanghai University, No.99, ShangDa Road, Shanghai, China*

zyh@shu.edu.cn, silverbelial@gmail.com

Abstract

This paper discusses the efficiency optimization for mobile application. By adopting reconfigurable software architecture, a dynamic reschedule method is proposed. Modeling the overheads of single operation, the uncertainty caused by control structure is represented with experienced data and branch prediction. Several criteria are proposed considering different situation.

Keywords: *Mobile Application, Efficiency Optimization, Reconfigurable Software*

1. Introduction

With the rapid development of mobile communication network, the quality of service needs to improvise. At present, the size of the mobile communication network is substantial; improving the quality of network services in the future will mainly depend on the specific system of network optimization project. However, due to the nature of the mobile device itself, it is necessary to take the computing and storage capacity of the device into consideration when developing application on mobile device. In general, mobile devices have limited capacity of RAM, low-frequency CPU and the limited capacity of the non-volatile memory. More importantly, mobile device has limited battery capacity. Now there is a trend that the higher frequency CPU and larger memory is used in mobile devices, which makes the electricity become a crucial factor.

It has become popular to adopt customized mobile application. These customized applications make it possible and convenient that one can handle various businesses through his mobile device. However, in practical, due to the limitations of the mobile network environment and mobile device computing ability, data-intensive computing response time becomes very long and unbearable, which leads to poor user experience. However, data-intensive computing may be considered as their core business needs and functions which are commonly used.

Therefore, a strategy using reconfigurable software framework is proposed to optimize the efficiency of data-intensive mobile application based on C/S structure.

2. Reconfigurable Software

ISSN: 1738-9968 IJHIT
Copyright ⓒ 2013 SERSC

Performance and versatility are two related issues gathering increasing attention within the embedded device co-design community. Reconfigurable computing is considered by many as a viable solution to address these future mobile device demands [1].

According to architecture issued by Nitsch. C. and Lara. C., mobile device management needs a server architecture which is networked, modular and flexible [2]. Their architecture is built using Enterprise Java Bean (EJB) technology and uses a system management API constructed with XML. Their approach offers services such as application download on demand and on-the-fly update of hardware acceleration components.

An agent-based architectural framework supporting networked reconfigurable mobile devices has been explored in our work in [3]. The overall aim is to enable a mobile device to dynamically retrieve an optimal, automatically partitioned and individualized reconfigurable hardware-software based application solution from a provisioning server.

Many other frameworks can be learned from [4-7]. In [4], a programming framework based on a XML description of a set of available reconfiguration mechanisms is used to provide programmers with a development kit to write programs able to automatically reconfigure selected functionalities every time a hardware block involved into the computation is detected as faulty.

3. Optimization Strategy

In this chapter a strategy using reconfigurable software architecture is proposed to schedule the computing tasks according to the scheme that is most efficient in demand. Modeling and optimization objective is the key fracture of this strategy.

3.1. Modeling

3.1.1. Independent Logical Unit: In this section, the Independent Logical Unit (ILU) is introduced to express atomic tasks that can migrate independently. ILU has several properties as follows:

1) Indivisibility. ILU itself cannot be split into several ILU s, which means the internal logic in an ILU cannot be interrupted. Otherwise, the whole program would collapse when the dynamic migration occurred.

2) Context-free. ILU can be called in different devices and the required data should be transferred as input parameters to ensure correct operation.

3) Each ILU can specify a subsequent ILU , but it is the system's responsibility to invoke the next ILU , since the next ILU may not be at the same device with the previous one.

However, not all of the ILUs can be dynamically allocated to either server or client in the run-time, because each ILU may require certain physical resources situating the server side or the mobile device side, such as: ILUs that involve the database operations can be implemented only on the server side, because such ILUs allocated to the mobile device side will lead to a "database not find" exception; otherwise, ILUs to display specific data to the screen must be executed in the mobile device side.

Based on the properties above, a request in a C/S architecture system can be modeled as a path composed by a number of ILUs . To abstract the computation and communication related factors, some sets and functions are introduced as follow:

ILUS is the set of all ILUs ;

 Copyright ⓒ 2013 SERSC

CPoints is the set of all devices;

R is the set of all real numbers;

CpM is a function mapping from ILUS to R, which stands for the computation amount of a specified ILU;

CpA is a function mapping from CPoints to R, which stands for the computation ability of a specified device;

CpT is a function mapping from CPoints × ILUS to R, which stands for the computation time to compute a ILU on the specified device;

$$\mathrm{CpT}(\mathrm{device}_i, \mathrm{ILU}_j)=\mathrm{CpM}(\mathrm{ILU}_j)/\mathrm{CpA}(\mathrm{device}_i) \quad (1)$$

CmM is a function mapping from ILUS × ILUS to R, which stands for the communication amount between two ILUs;

CmA is a function mapping from CPoints × CPoints to R, which stands for the communication speed between two devices; ($\mathrm{CmA}(device_i, device_i) = \infty$)

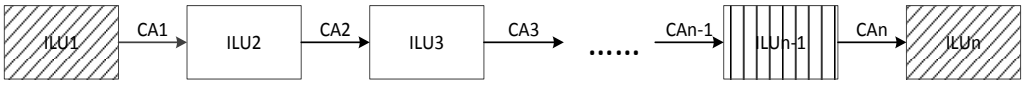

Figure 1. A linear ILU Sequence Example

In Figure 1, the diagonal striped ILUs (*e.g.*, $\mathrm{ILU}_1, \mathrm{ILU}_n$) must run in the mobile side, the vertical striped ILU (*e.g.*, ILU_{n-1}) must run on the server side and the other ILUs from ILU_2 to ILU_{n-2} have no requirements of physical resources which can be dispatched to both server side and client side.

A schedule scheme can be modeled as a function S mapping from ILUS to CPoints. If it have equation $\mathrm{S}(ILU_k) = device_l$, the ILU_k is scheduled to $device_l$.

3.1.2. Control structure of ILU: No matter how to express program logic, both the branch structure and loop structure will be inevitable. These two control structures will lead uncertainty when scheduling ILUs in reconfigurable software systems. In order to decide the schedule strategy before process a request, the cost of these two control structures must be estimated first.

Firstly, the branch structure is discussed. Due to the different weights of each branch, the ILU sequences that have branches may have different final weights. In most cases, it is not possible to determine how the process will be when receiving a request.

For such a situation, this paper presents several solutions as follows:

1) Branch Prediction. This solution is similar to the idea of branch prediction used in CPUs. A data structure is introduced to record all the former branch choices. According to the records statistics a prediction is made to determine which branch shall be taken. The corresponding factor is computed according to the generated path.

2) Branch Traversal. In order to prevent the rapid decrease of the efficiency when running into wrong prediction, branch traversal is introduced in this paper. The program explores all

Copyright ⓒ 2013 SERSC

the possibilities that the branch may take and record the corresponding schedule scheme in each path.

When considering the loop structure, operations in a loop control perform multiple times, so that a scheduling point within the cycle may lead to a proliferation of communication. In this paper, the whole loop body is treated as a special ILU , the cost is estimated based on the number of executions.

3.2. Factors in choosing the scheduling schemes

The previous section has modeled the reconfigurable program requests. It is easy to find an optimal schedule scheme in certain situation. But there is no silver bullet. This section will focus on discussing the influential factors in various circumstances to generate corresponding scheduling strategies.

3.2.1. Response time: The communication cost can be ignored when using local network, since it is commonly at a low level. As the main objective turns to optimize the response time, the following formula can be established:

$$\begin{aligned} Time_{response} = & \sum_{i=1}^{n} CpT(S(ILU_i), ILU_i) \\ & + \sum_{j=1}^{n-1} CmM(ILU_i, ILU_{i+1}) / CmA(S(ILU_i), S(ILU_{i+1})) \end{aligned} \tag{2}$$

The goal of the scheduling strategies is to achieve $\min(Time_{response})$ which means the shortest response time.

3.2.2. Communication Cost: As the mobile devices may be running in the commercial mobile network in which communication cost a lot of time and money, pursuit of reducing the response time at the cost of higher communication overhead will not be accepted by the users. In this case, the communicate cost turns to be the major optimization objective.

Communicate cost of a single request can be expressed by the following formula:

$$Cost_{com} = \sum_{j=1}^{n-1} CmM(ILU_i, ILU_{i+1}) \tag{3}$$

In this case, the goal of the scheduling strategy is to obtain $\min(Cost_{com})$.

3.3. Optimization Objective

In reality, too much communication will cause a lot of server pressure, while long response time will affect the user experience. Thus, in most cases, a balance is needed between the two factors.

$$C = \alpha Time_{response} + \beta Cost_{com} \tag{4}$$

In this case, the goal of the scheduling strategy is to obtain $\min(C)$.

As the values of the weight coefficients α and β will directly affect the schedule scheme, a set of suitable weight coefficients can adjust both the server pressure and the user experience to an appropriate level.

To determine the ratio of these two coefficients, some corresponding experiments are carried out and the numeric results are listed in following tables:

 Copyright ⓒ 2013 SERSC

Table 1. Experiment results under 0.1Mb/S network

$\alpha:\beta$	Average Response Time(ms)	Average Transmission Amount(Bytes)
10	921	9210
5	1213	8416
2	1421	5277
1	1596	2108
0.5	1627	1728
0.2	2245	928
0.1	3210	871
0.01	3521	854

Table 2. Experiment results under 1Mb/s network

$\alpha:\beta$	Average Response Time(ms)	Average Transmission Amount(Bytes)
10	872	8922
5	1421	7124
2	1523	4312
1	1621	1572
0.5	3752	1128
0.2	3661	927
0.1	3521	887
0.01	3576	887

Table 3. Experiment results under 10Mb/s network

$\alpha:\beta$	Average Response Time(ms)	Average Transmission Amount(Bytes)
10	821	9421
5	927	7125
2	1124	2472
1	1536	1927
0.5	2721	1275
0.2	2572	721
0.1	2917	823
0.01	2871	792

Table 4. Experiment results under 100Mb/s network

$\alpha:\beta$	Average Response Time(ms)	Average Transmission Amount(Bytes)
10	891	9121
5	1252	7927
2	1623	5332
1	1821	2342
0.5	2028	1821
0.2	2125	1029
0.1	2252	972
0.01	2262	962

Copyright ⓒ 2013 SERSC

According to the results, the recommended ratio of the coefficients is:

- $\alpha:\beta=0.5$ under 0.1 Mb/s network;
- $\alpha:\beta=1$ under 1 Mb/s network;
- $\alpha:\beta=2$ under 10 Mb/s network;
- $\alpha:\beta=1$ under 100 Mb/s network;

3.4. Optimization Algorithm

To improve the response time and reduce the communication amount, the algorithm which produces a schedule scheme is crucial.

Usually, to find the schedule scheme which produce the minimal value of the C, all the schedule schemes shall be generated and the corresponding C shall be computed. However, the time complexity of this kind of method is $O(m^n)$; m is the size of ILUS and n is the size of $CPoints$. This time complexity is totally unacceptable in a real time reschedule.

After processing the best schedule scheme samples, it is obvious that most best schedule scheme has few "gap points" (A gap point means an ILU_i is scheduled to $device_x$ and ILU_{i+1} is scheduled to $device_y$ and $x \neq y$), thus the method to find the local optimal solution is proposed.

Table 5. The Gap points' samples

High Speed network(100Mb/s)		Low speed network(<1Mb/s)	
Gap points	Best schedules	Gap points	Best schedules
2	9	2	12
4	8	4	7
6	2	6	1
>=8	1	>=8	0

The pseudo-code of the algorithm is stated as follow:

1. Convert the ILUS is a undirected graph;
2. Apply Tarjan algorithm to find the cutting edges and put them into a set named *CuttingEdges*
3. Compute the all the subsets of the *CuttingEdges* which size is no greater than 2
4. Compute the C value and find the best schedule scheme.

Therefore, a local optimal solution is generated. The time complexity of the algorithm above is reduced to $O(m^2)$.

4. Summary

This paper proposed a method to improve performance of mobile application in different situation. When modeling the overhead, certain prediction technique is used while different prediction technique may leads various schedule scheme. Some recommended coefficients ratio of response time and communication amount is listed as well. A locally optimally algorithm is proposed to avoid the high time complexity of the optimization.

 Copyright ⓒ 2013 SERSC

Acknowledgements

The authors would like to thank Yi Shen, Zhiguo Wu, Ju Lin and Lu Xiao for their assistance. The research was conducted with financial from Innovation Program of Shanghai Municipal Education Commission B.10-0108-08-002.

References

[1] T. O'Sullivan and R. Studdert, "Agent Technology and Reconfigurable Computing for Mobile Devices", 2005 ACM Symposium on Applied Computing, **(2005)**.
[2] C. Nitsch, C. Lara and U. Kebschull, "A Novel Design Technology for Next Generation Ubiquitous Computing Architectures", In Reconfigurable Architectures Workshop (RAW-03), **(2003)**.
[3] T. O' Sullivan and R. Studdert, "Mobile Agent Technology and Networked Reconfigurable Embedded Devices", In Proceedings of International Conference on Pervasive Computing and Communications, **(2004)**.
[4] D. Carlo, S. Prinetto and P. Scionti, "A FPGA-Based Reconfigurable Software Architecture for Highly Dependable Systems", *J+ Control & Comput. Eng. Dept., Politec. di Torino, Torino, Italy, **(2009)** November.
[5] A. W. Mast, "Reconfigurable Software Defined Payload architecture that reduces cost and risk for various missions", *J+ Gov. Commun. Syst. Div., Harris Corp., Melbourne, FL, USA, **(2011)**.
[6] C. Xia and Y. Zhu, "An Efficiency Optimization Strategy for Huge-Scale Data Handling", *J+ Recent Advaces in Computer Science and Information Engineering, vol. 125, **(2012)**, pp. 393-398.
[7] I.-Y. Chen and C. -C. Huang, "A Reconfigurable Software Distribution Framework for Smart Living Environments", *J+ Nat. Taipei Univ. of Technol., Taipei, **(2007)**.

Copyright ⓒ 2013 SERSC

Copyright ⓒ 2013 SERSC