OFFLOADING TO NEIGHBOURING NODES IN SMART CAMERA NETWORK SECURITY

D Bikshapathi, SIRISHA MOTURI, PUJITHA MANDAPATI
4.THAMBI VINOD KUMAR, 5.SHRUTI
Assistant Professor, Department of Computer Engineering, Ellenki college of Engineering and Technonlogy, patelguda (vi), near BHEL ameenpur (m), Sangareddy Dist. Telangana 502319

.

ABSTRACT

Abstract—Mobile Cloud Computing refers to offloading com- putationally intensive algorithms from a mobile device to a cloud in order to save resources (time and energy) in the mobile device. But when the connection to the cloud is non-existent or limited, as in battle-space scenarios, exploiting neighbouring devices could be an alternative. In this paper we have developed a framework to offload computationally intensive algorithms to neighbours in order to minimise the algorithm completion time. We propose resource allocation algorithms to maximize the performance of these systems in real-time computer vision applications (drop less targets). Results show significant performance improvement at the cost of using some extra energy resource. Finally we define a new performance metric which also incorporates the energy consumed and is used to compare the offloading algorithms.

Index Terms—Offloading, Mobile Cloud Computing, Energy

I. INTRODUCTION

Off-the-shelf smartphones are becoming ubiquitous and powerful, making them an interesting prospect to form a smart networked camera. However, they are not powerful enough for many applications, especially if the results are required in real-time. We have previously considered their pros and cons for distributed person re-identification [1]. If smartphones are deployed to carry out computationally intensive computer vision tasks, such as person tracking and re-identification between multiple cameras, they may not always be able to process everything within a user specified time. As such we can define performance of a system as the ratio of number of jobs processed to the number of jobs available.

Conventionally, computationally intensive algorithms have been offloaded to the “cloud” and it has been shown in some cases to save time and energy [2] [3]. In this paper however, we present a novel framework to offload these tasks to neigh- bouring mobile nodes which can significantly increase the performance without substantially depleting battery resource compared to the non-offloading case. We also present a single metric called Efficiency Score (ES) which also incorporates the energy consumption along with the performance.

unreachable. Even if there is a connection, the Network Bandwidth (BW) may be too low or intermittent. Neighbouring devices lack the computational power and energy of the cloud but may be readily available with high BW connections. However, we need to consider their available energy resources before offloading a job as when in the field, charging may not be readily available. In this paper, we discuss its feasibility in terms of time and energy. Magurawalage et al. have considered offloading to intermediate cloud like entity called cloudlets [6] but to the best of our knowledge, battery powered neighbours have not been used by anyone as an offloading candidate.

Computing platform types

If there is no network connectivity, the only option is to do on-board processing. However, if there is some connectivity, we have the option to offload. We use the term “onloader”

for the system which the "offloader" offloads its workload to. Some on-board processing can reduce the amount of data to be communicated while freeing up the onloader's resources. For example, a background subtraction algorithm can limit the sensor from sending images with little or no activity. This saves communication cost for the sensor and the onloader has fewer jobs to perform. However, when the algorithms areonloader. Then, X can offload (P N) targets to the cloud and as Kumar et. al explains, it can save time and energy [2]. If the cloud is not available, conventional system would simply drop the targets. However, device Y has no target in its FOV at this moment. We show that neighbour devices like Y can be good alternative and may be used for offloading.

We now describe the simulator we developed to study smartphones with cellular and Wi-Fi communication capabili- ties. Then in section III, we describe how we use the simulator for a person re-identification system. Then we describe exper- iments in section IV and show results. Finally in section V, we present conclusions to the paper.

II. MODELLING NETWORK OF SMART CAMERAS

The simulator allows us to use a simplified model of the algorithm flow for the target platform and update components easily as required. Wu et al. used queuing model theory to simulate workload on distributed nodes [3]. However, theirassumption that when there is no workload the nodes do not consume any energy is not valid in real life. The major elements of our simulator relate to the algorithmic tasks, the sensor architecture, communication links and the targets. We go through each one in detail below.

A. Algorithmic tasks

The simulator's model for the algorithmic task is characterised by its number of Operations (OP), input and outputdata size. For example, a person detection algorithm takes an image of size M N as the input, requires approximately C OP per image and outputs the number of persons in theimage. Assuming one OP per clock cycle, we can estimate the execution time on the device using the clock frequency

$$T_{exec} \propto \text{Clock Frequency} \quad (1)$$

We are aware that in different processors some OP take more than one cycle and multiple OP can be possible in one cycle, however this approximation (Eqn. (1)) gives an estimate of time required without detailed execution information. If desired, algorithms could be executed on the Device Under Test (DUT) to measure the execution time more precisely.

The number of OP required for an algorithm can change. For instance, in the Mixture of Gradients (MOG) algorithm for background subtraction, it depends on how quickly a matching Gaussian distribution is detected for the particular pixel [7]. To make calculations easier for the simulation, we take the worst case scenario where the matching Gaussian is not found.

B. Component Based Sensors

In order to realistically emulate its behaviour, a sensor is divided into its components such as the Central Processing Unit (CPU) and cellular radio. We do not consider the energy consumption by the display as it can be turned off by theapplication. We use the utilisation based model by Jung et al. to calculate the energy consumption [8] and our parameters are based on a Google Nexus I phone which was one of theirDUTs. However if desired, the simulator can be calibrated for a different DUT in a straightforward manner.

TABLE I. CPU PARAMETERS

Frequency	245.0	384.0	460.8	499.2	576.0	614.4	652.8	691.2	768.0	806.4	844.8	998.4
β^{cpu}_{freq}	201.0	257.2	286.0	303.7	332.7	356.3	378.4	400.3	443.4	470.7	493.1	559.5
β^{cpu}_{idle}	35.1	39.5	35.2	36.5	39.5	38.5	36.7	39.6	40.2	38.4	43.5	45.6

1) Image Sensor: The image sensor consumes significant energy in a mobile device when used continuously. According to Likamwa et al., the energy consumption per frame of the image sensor can be modelled as follows [9].

$$E_{camera} = P_{idle} \times (T_{frame} - T_{active}) + P_{active} \times T_{active} \quad (2)$$

where $T_{active} =$ Number of Pixels . Based on Eqn. 2 , we can either reduce the image resolution, thereby reducing Tactive or reduce the acquisition rate to save the energy consumption.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-10, 2016
DOI: 10.21276/ijcesr.2016.3.10.12

2) Application Processor (AP): The CPU power is made up of two parts, idle power and the running power, as follows:

$$pcpu = \beta cpu \times u + \beta cpu, \quad (3)$$

where u is the utilisation and βcpu are the CPUparameters listed in Table I. We calculate the utilisation as the ratio of the CPU time used vs the time available per frame. However, the CPU is also used by the Operating System (OS) and other running applications. Dargie used normal and exponential distributions to simulate workload in [10]. We also used a random variable (r) sampled from a Gaussian distribution to simulate these other activities. By adjusting the mean of r we can a simulate busy sensor and idle sensor. The total utilisation is calculated as shown below.

$$u = \frac{\sum_{i=1}^{N} T_{execi}}{T_{Frame}} + r \quad (4)$$

where N is the number of algorithms to be processed T_{execi} is the execution time for ith algorithm (see Table II forexecution times for all algorithms) and TFrame = 1 is the time available for each frame. In the situation where Texeci > TFrame which is very likely in the case of algorithms for person re-identification; we only run the CPU to 100% load and run the remainder of the algorithm in the next frame and so on.

3) Cellular (3G): Cellular radio is modelled as a three state system: IDLE, Forward Access Channel (FACH) and Dedicated Channel (DCH). The IDLE mode is the non com- municating mode and has the lowest power consumption. In this mode, the User Equipment (UE) is turned on but has not established Radio Resource Control (RRC) connection with the Radio Network Controller (RNC). In DCH state the UE has a dedicated transport channel for data transmission in both directions, but this is 50 to 100% more expensive than FACH, where FACH is the intermediate state with reduced power consumption and low data rate. There is no dedicated channel allocated in this mode and it can only transmit user data through shared low speed channel that is typically less than 15kbps [11]. As we can see from Eqn. (5), power is only dependent on state but not on utilisation. Fig. 2 shows the state diagram with the inactivity timers which along with data buffer size controls the state promotions and demotions.

$$\begin{cases} \beta_{IDLE} & \text{if RRC state is IDLE} \\ \beta FACH & \text{if RRC state is FACH} \\ \beta DCH & \text{if RRC state is DCH} \end{cases} \quad (5)$$

where RRC is the current state of UE and βIDLE, βFACH and βDCH are based on [8].

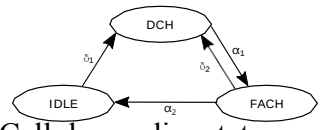

Fig. 2. Cellular radio states, α1 and α2 are inactivity timers whereas δ1 and δ2 are delay to get to DCH

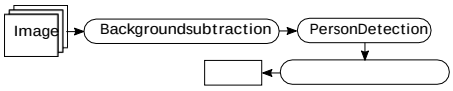

Fig. 3. Person Re-identification work-flow.

4) Wi-Fi: The Wi-Fi model calculates the time and energy of the Wi-Fi component in the connected mode. There are two modes depending upon the packet rate.

$$pwifi = \begin{cases} \beta LT \times p + \beta LT\ base & \text{if } p \leq \text{Threshold} \\ \beta HT \times p + \beta HT\ base & \text{if } p > \text{Threshold} \end{cases}$$

where p is the packet rate, βLT, βHT, βLT base and βHT base are the parameters of the DUT based on [8]. If the number of packets per second exceeds the threshold of 20 then Wi-Fi is in the high power state, else in the low power state. Unlike the cellular system, the power consumption is directly proportional to the data rate. Although Wi-Fi consumes energy in scanning mode, we ignore it as we assume a connection between the sensors as the basis of this research.

III. SYSTEM DESIGN

In this section, we show how offloading may be used toincrease the performance of a system. We consider a pedestrian re-identification system outlined in Fig. 3. It starts with imageacquisition from the image sensor. A background subtraction and a person detection algorithm is applied on the image to detect the

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-10, 2016
DOI: 10.21276/ijcesr.2016.3.10.12

number of people in the view. When there a pedestrian is detected, we apply a person re-identification algorithm to each detection such as [12]. Our goal is to identify as many detections as possible.

A. Application Partitioning

The algorithmic complexity of the person re-identification algorithm outweighs that of other algorithms in the chain (see Table II). So, the overall complexity of the system can be estimated as (N) where N is the number of people detected. To be realistic, we limit the number of people in an image (800 × 600) in the simulator to be fewer than 10.

B. Energy Saving Methods

We replicate following energy saving techniques to make the simulation realistic as much as possible.

1) Dynamic Frame per Second: We can save energy by decreasing the number of FPS of the system (see Eqn. (4)). However, very low FPS may mean some of the detections may be missed. We implemented an algorithm to vary the FPS of each individual sensor between 1 and 16 in the following way

$$\text{FPS (new)} = \begin{cases} \text{FPS(old)} \times 2 & \text{if } t < \tau \\ \text{FPS (old)} \div 2 & \text{if } t > \tau \end{cases} \quad (7)$$

where t is the time between target activities and τ is 5 seconds.

TABLE II. EXECUTION TIMES FOR CPU RUNNING @ 998.4 MHZ

Algorithm	ExecutionTime
BackgroundSubtraction	0.1
PersonDetector	0.2
PersonRe-identification	5.1
TotalTime	5.3

2) Dynamic Voltage and Frequency Scaling (DVFS): A simple algorithm controls the clock frequency of the sensor. When the CPU utilisation is below 0.4, the clock frequency is lowered according to Table I and it is scaled to maximumfrequency as soon as the utilisation is above 0.9.

C. Offloading

We classify only the re-identification algorithm as offload- able as for others, the communication costs and the time delay outweighs the benefits of offloading. Offloading an algorithmentails sending input data, waiting for the onloader to execute, and receiving output data. Before transmitting however, the data has to be formatted in packets and some overhead will beadded to the processor. These operations can be a few hundred per packet which needs to be added to the CPU workload.

Time Cost: The communication times are proportional to the data size to be communicated and inversely proportional to the network BW. We assumed the BW to be static. Waiting times can be estimated using Eqn. (1) for the onloader's clock frequency. But, it does not take into account the CPU load. If our onloader's AP is already busy, our estimation can be very far from reality. So we re-write the time calculation using onloader's average CPU utilisation.

$$T_{wait} = \frac{T_{exec}}{1 - E[u]} \quad (8)$$

where E[u] is the average of u from Eqn. (4). The total time cost which is also known as makespan, is shown below [13].

$$T_{total} = T_{packet} + T_{send} + T_{wait} + T_{receive} \quad (9)$$

where Tpacket is the time to format the data in a packet.

Energy Cost: There are two energy costs involved. The first is for the offloader (Eoff) and includes data packeting and the radio communication cost.

$$E_{off} = (T_{send} + T_{receive}) \times P_{radio} + P_{cpu} \times T_{packet} \quad (10)$$

where radio 3G, WiFi . Second is for the onloader which includes radio cost, execution cost and the packeting costs. So far, in literature, cost for the onloader is ignored as energy is not of major concern for the cloud. But while offloading to the neighbours, we need to consider it.

$$E_{on} = (T_{receive} + T_{send}) \times P_{radio} + P_{cpu} \times (T_{execute} + T_{packet}) \quad (11)$$

D. Multi-Objective Optimisation

The time and energy costs from Eqn. (9, 10 and 11) can be inferred as variables of a multi-objective optimisation problem.

$$Cost = w_{time} \times T_{total} + w_{off} \times E_{off} + w_{on} \times E_{on} \quad (12)$$

where wtime, woff and won are the weights for each objective. The cost function involves adding time and energy variables(i.e. different units), which requires careful selection of the weights. We avoid this situation by limiting one of the objec- tives to a threshold (ε) and

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-10, 2016
DOI: 10.21276/ijcesr.2016.3.10.12

optimising rest of the objectives [14]. Regarding the real-time nature of our problem, we limit the time and optimise the energy variables. We set $\varepsilon = 25$ seconds and leave out the nodes that do not satisfy this constraint (denoted by the light dots in Fig. 4). It is still a multi-objective problem but only with two variables of the same unit (Joules). We now study three methods to optimize offloading performance.

1) Minimize Energy Cost (MEC): In this method, we choose the node that satisfies the time constraint described above and incurs the minimum offloader and the onloader energy cost. The solution is pareto-optimal and denoted by nodes on the line in Fig. 4 [14].

$$\text{CostMEC} = w_{off} \times E_{off} + w_{on} \times E_{on} \quad (13)$$

2) Minimize Battery Impact (MBI): In a battery-powered device, using the least energy cost per job alone may not in- crease device lifetime. For example, say an algorithm requires 10 and 8 Joules on devices X and Y respectively. But X and Y have 500 and 50 Joules left in their battery respectively. Considering energy cost alone, Y is the best choice but when we consider the amount of energy left in the device clearly X is a better choice. We re-write Eqn. (13) as follows.

3) Offload Only if Busy (OOB): The previous methods try to find the global solution, but offloading has overhead costs. So, this method tries to offload only if on-board processing is estimated to be infeasible. To do so, we add all the operations in the execution queue and use Eqn. (1) to estimate the minimum remaining processing time. If this time is greater than the threshold (ε), offload the algorithm minimising the time and energy objectives defined in Eqn. (12).

IV. SIMULATION AND RESULTS

We simulated a number of sensors connected to each other by Wi-Fi and to the server (when available) by cellular link. For simplicity, we assume that resource information about all the nodes (remaining energy, current CPU load etc.) is avail- able and all the sensors have same computational capability but the server is 10 times more powerful. Also there is no energyand using the best representation for identification purposes.

We tested the algorithms with various parameters using 100 Monte-Carlo runs each representing a 10 minute period. The results are listed in Table III. When the cloud is not available, Successful Identifications (SI) (which is targets detected minus targets dropped) improved from 10.4 in the Non Offloading (NO) case to 12.7 for the MEC case and 13 for OOB case but degraded to 8.87 for MBI. MEC and OOB boosted the performance by more than 20% while only incurring around extra 10% energy consumption. MBI did not perform well because of the communication overhead. However, if the runs were longer, may be we would see some improvement in the device lifetime. All the algorithms did better than the NO case when the cloud was available, dropping almost no targets but the energy consumption was significantly higher. This shows that offloading to cloud blindly may increase the performance.

V. CONCLUSION

This paper presented a simulation model for offloading computationally intensive algorithms to neighbouring devices when the cloud is not available. The results show that among the three, OOB consistently achieved the best trade-off be- tween power and performance. It improved the performance by approx. 25% while costing about 10 20% more energy. The ES metrics suggests that energy is used more productively

REFERENCES

[1] S. Sthapit, J. Thompson, J. Hopgood, and N. Robertson, “Distributed implementation for person re-identification,” in Sensor Signal Process- ing for Defence, 2015, pp. 1–5, Sept 2015.

[2] K. Kumar and Y.-H. Lu, “Cloud Computing for Mobile Users: Can Of- floading Computation Save Energy?,” Computer (Long. Beach. Calif)., vol. 43, no. 4, pp. 51–56, 2010.

[3] H. Wu, W. Knottenbelt, and K. Wolter, “Analysis of the Energy- Response Time Tradeoff for Delayed Mobile Cloud Offloading,” in Teletraffic Congr. (ITC 27), 2015 27th Int., pp. 134–142, 2015.

[4] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” Futur. Gener. Comput. Syst., vol. 29, pp. 84–106, jan 2013.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-10, 2016
DOI: 10.21276/ijcesr.2016.3.10.12

[5] P. Natarajan, P. K. Atrey, and M. Kankanhalli, “Multi-Camera Coordi- nation and Control in Surveillance Systems : A Survey,” ACM Trans. Multimed. Comput. Commun. Appl., vol. 11, no. 4, 2015.

[6] C. M. Sarathchandra Magurawalage, K. Yang, L. Hu, and J. Zhang, “Energy-efficient and network-aware offloading algorithm for mobile cloud computing,” Comput. Networks, vol. 74, pp. 22–33, 2014.

[7] Z. Zivkovic, “Improved adaptive Gaussian mixture model for back- ground subtraction,” Proc. 17th Int. Conf. Pattern Recognition, 2004. ICPR 2004., no. 2, pp. 28–31 Vol.2, 2004.

[8] W. Jung, C. Kang, C. Yoon, D. D. Kim, and H. Cha, “DevScope: A Nonintrusive and Online Power Analysis Tool for Smartphone Hardware Components,” Proc. Eighth IEEE/ACM/IFIP Int. Conf. Hard- ware/Software Codesign Syst. Synth., pp. 353–362, 2012.

[11] C. W. Johnson, Radio Access Networks for UMTS: Principles and Practice. John Wiley and Sons, 2011.

[12] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, “Person re-identification by symmetry-driven accumulation of local fea- tures,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 2360–2367, 2010.

[13] Y. Jiang and S. Member, “A Survey of Task Allocation and Load Balancing in Distributed Systems,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 585–599, 2016.

[14] M. Caramia and P. DellOlmo, “Multi-objective Optimization,” in Multi- objective Manag. Freight Logist. Increasing Capacit. Serv. Lev. Saf. with Optim. Algorithms, pp. 11–37, Springer London, 2008.

[15] F. Bai and A. Helmy, “A Survey of Mobility Models in Wireless Adhoc Networks,” Wirel. Ad Hoc Sens. Networks, pp. 1–30, 2004.