Multi-Objective Computation Offloading for Mobile Biosensors via LTE

Pascal Libuschewski*, Dennis Kaulbars†, Björn Dusza†, Dominic Siedhoff‡, Frank Weichert‡, Heinrich Müller‡, Christian Wietfeld† and Peter Marwedel*
*TU Dortmund University, Department of Computer Science XII, Dortmund, Germany
†TU Dortmund University, Communication Networks Institute, Dortmund, Germany
‡TU Dortmund University, Department of Computer Science VII, Dortmund, Germany

***Abstract*—For a rapid identification of viral epidemics a mobile virus detection is needed, which can process samples without a laboratory. The application of medical biosensors, at key positions with a high passenger volume (e.g. airports) became increasingly important as epidemic early warning systems. As mobile biosensors have to fulfill various demands, like a rapid analysis and a long battery lifetime, in this study we present a multi-objective computation offloading for mobile sensors. The decision whether it is beneficial to offload work to a server, using the Long Term Evolution (LTE) wireless network, can be made automatically and dynamically on the basis of conflicting objectives and several constraints.**

I. Introduction

In the face of diseases spreading fast all over the world from airport to airport, a rapid mobile virus detection is needed for a successful containment of epidemics [1]. A medical biosensor which can detect viruses, is the PAMONO (Plasmon-Assisted Microscopy of Nano-Objects) sensor [2]. It is a modified SPR (Surface Plasmon Resonance) sensor, which is able to detect individual viruses within less than three minutes. As is shown in Figure 1, viruses are detected while a liquid or air sample is passed through a flow cell and the viruses attach to the antibodies on the sensor surface, resulting in a small increase in brightness on the processed camera images. An automatic detection software [3] is used to count the number of viruses.

These sensors can be used in different places, like airports, railway stations or in areas far away from urban areas for a rapid virus test. The mobile application of the sensor assumes an even distribution of resources between the devices and a central server, as can be seen in Figure 2. The distributed sensors form a network, where single devices can cooperate with other devices in the network to provide a reliable and rapid detection of viruses. With this new paradigm of cooperative (virus-) detection, the concept of offloading becomes important for energy consumption and to improve the performance - retaining a constant quality of the detection [4], [5]. In the context of offloading, processing time is important, particularly the adherence of strict time limits, if several samples have to be analyzed in a short period of time. The presented study focuses on an evaluation of the trade-off between different resources (e.g. energy, time) for a sustainable, cooperative use of biosensors. Therefore a novel approach for an energy efficient offloading for mobile biosensors is presented.

The computation to count the viruses can either be done on the mobile device with a mobile Graphics Processing Unit (GPU) or offloaded by using the Long Term Evolution (LTE) wireless network to transfer all sensor images or partial results of the local computation to a server. The server executes the remaining calculation and transfers the final result back to the mobile device. The decision, whether and at what point the computation should be offloaded, is based on a multi-objective optimization. Within the optimization, two different simulators are used to calculate the energy consumption and run time of the GPU and the energy consumption and transfer time of the LTE device.

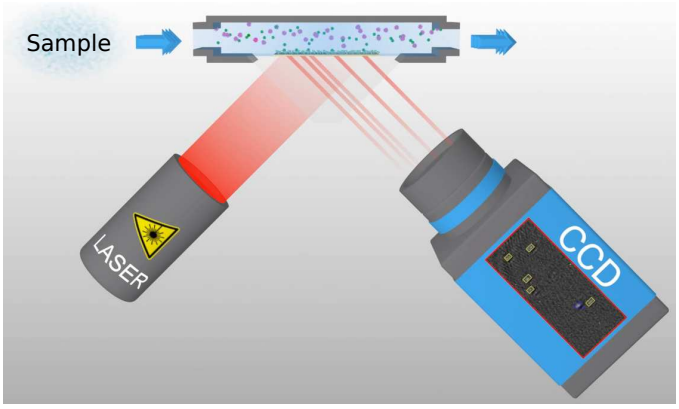

Fig. 1. PAMONO biosensor. A sample with viruses is inserted into a flow cell. A laser illuminates a gold layer with antibodies on top and a camera records the reflected light. The viruses individually attach to the antibodies, and the attached viruses are counted automatically.

II. Related Work

Several approaches for computation offloading on mobile devices exist. Kumar and Lu [6] have shown that cloud computing can save energy on handheld devices, but not all applications lend themselves saving energy using the cloud. Li et al. [7] have shown how computation offloading can be used on handheld devices which are connected via WLAN to a server. An adaptive computation offloading for battery powered devices has been shown by Xian et al. [8]. Toma and Chen [9] have shown computation offloading for real-time tasks. Rudenko et al. [10] describe a framework to automatically migrate tasks to a server to receive the results.

In contrast to the existing approaches, here a multi-objective design-space exploration is conducted, which can be used for different hardware and software configurations.

Multiple and possibly conflicting objectives can be chosen, such as energy consumption, power consumption, number of core cycles, run time or the energy delay product. The presented optimization is done in a hardware/software co-design to identify the hardware and software configurations which meet the constraints.

The structure of the paper is as follows: Section III presents the automatic offloading method. In Section IV the results and discussion of the multi-objective evaluation are given. Finally, the Section V provides the conclusion and future work.

III. Methods

In order to decide whether a computation should be executed locally on the mobile device or offloaded to a server, the expected energy consumption of the mobile device has to be determined. This includes the energy costs for a local computation, as well as the energy costs for transferring data to the server and back to the device. The presented framework consists of three main parts: The computation of the energy consumption and run time of the GPU, the energy consumption and transfer time of the LTE device, and optionally, a genetic algorithm to explore larger design spaces, for which exhaustive sampling would require prohibitive computational effort.

The mobile processing device, that is used in conjunction with the mobile biosensor, is in this setup a laptop equipped with a mobile GPU and a LTE modem. Later on this setup will be replaced by a small handheld device, e.g. a tablet with an embedded GPU and an embedded LTE modem.

A. Determining Energy Consumption of the GPU

To calculate energy consumption and run time of the GPU, a modified version of GPUSimPow [11] is used, which is based on the (well known) simulator GPGPU-Sim [12] and gives accurate results for the energy consumption and run time.

The simulator can be configured with various parameters like number of streaming processors (SPs), core clock rate, DRAM type, number of raster operation processors (ROPs), different task schedulers and different mixing networks, to model the configuration of the desired architecture.

Any given GPU program, like the automatic virus detection, can be simulated in a cycle accurate manner. This gives accurate results of the actual utilization of the different parts of the GPU and the average power consumption P_{GPU}. As the number of simulated cycles and the core clock rate of the GPU are known, the run time T_{GPU} can be calculated. The energy consumption E_{GPU} can then be calculated by multiplying the average power consumption P_{GPU}, that was consumed for processing, with the run time.

B. Determining the Energy Consumption of the LTE Device

To model the energy that is consumed for transferring the full sensor data or partial results of the calculation via the LTE network to the server the Context-Aware Power Consumption Model (CoPoMo) [13] is used. CoPoMo is a highly accurate Markovian energy consumption model for LTE devices.

The model is based on the assumption that an end device enters different power states while transferring data to the base station. Within each state, the device has different values for its average power consumption, which depend on a number of system parameters like the specific device characteristic, the carrier frequency and the number of physical resource blocks allocated to the device. The actual device characteristics have been determined by extensive measurements.

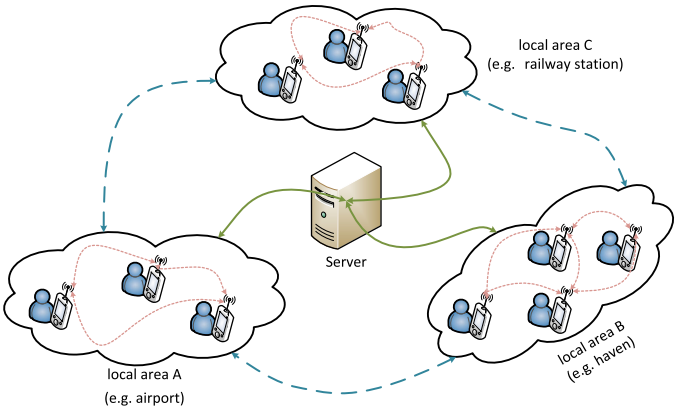

Fig. 2. Scenario of a network with several biosensors in different locations.

These power states can be used in a Markovian model, where each power state is mapped to a corresponding state within the Markov chain. The probability that the LTE device is within a specific state, as well as the transition probabilities between different states, depend on various context parameters like current radio channel conditions and traffic characteristics of the applications running on the device.

Therefore the Markovian model can be configured with context and system parameters. System parameters are for example the number of physical resource blocks used for a data transmission, as well as the assigned carrier frequency and the used modulation and coding scheme. The context parameters describe the cell environment, the file size of the data which has to be transmitted and the arrival rate of the data packets.

As result the average power consumption P_{LTE}, the average energy consumption E_{LTE} and the transmission time T_{LTE} can be determined for the LTE device.

C. Automatic Design Space Exploration

By combining the results from the different energy models, an overall energy consumption, run time and battery lifetime can be calculated. The overall energy consumption E is defined as

$$E = P_{\text{GPU}} \cdot T_{\text{GPU}} + P_{\text{CPU}} \cdot T_{\text{CPU}} + P_{\text{LTE}} \cdot T_{\text{LTE}} \qquad (1)$$

where P_{GPU}, P_{CPU} and P_{LTE} are the average power consumptions of the GPU, CPU (Central Processing Unit) and the LTE communication. The run time of the GPU and CPU and the communication time of the LTE device are given as T_{GPU}, T_{CPU} and T_{LTE}. The energy consumption of the server is not part of this model, as the objective is an energy efficient mobile device and not an energy efficient overall system. The energy consumption of the mobile CPU is also not part of this optimization, as the processing is either done on the GPU or on the server, hence this has no major effect on energy consumption.

The overall run time T is defined as

$$T = T_{\text{GPU}} + T_{\text{CPU}} + T_{\text{LTE}} + T_{\text{Server}} - T_{\text{Parallel}} \quad (2)$$

where T_{GPU}, T_{CPU} and T_{LTE} is the run time of the GPU and the CPU and the transfer time of the LTE device. The term T_{Parallel} is the time that can be saved by transferring data in parallel. In contrast to Equation 1, the run time of the calculation on the server T_{Server} is included, as the objective is the run time until the final result is calculated. The time to record the images (approximately two minutes) is not included, because no energy and run time can be saved on this task.

The corresponding expected battery lifetime B can be determined as

$$B = \frac{C_{\text{Batt}}}{P} \quad (3)$$

whereas C_{Batt} is the capacity of the battery and P the combined average power consumption of all parts of the device.

To show the generality of the method, four different mobile GPUs were simulated, which cover a wide range of performance levels. The evaluated GPUs are shown in Table I. The Geforce 520M GPU is the slowest of the four. It has 48 streaming processors (SPs), a core clock rate of 740 MHz and a GDDR-3 memory. The Geforce 580M GPU is the fastest. It has 384 streaming processors and a GDDR-5 memory, but has the slowest core clock rate with 620 MHz.

For reasons of simplicity, the LTE network was configured to use a fixed bandwidth and a fixed environment. If needed, the values for the LTE network can be modified easily, such that the design space exploration also takes into account a varying bandwidth or the environment.

The streaming algorithm [3], which automatically detects and counts viruses that appear in the sensor images was modified to cut off the calculation at several different steps in the pipeline. For example noise reduction and background elimination is done on the local device. Then the enhanced images, which can be compressed to a smaller size than the unmodified sensor images, are transferred to the server were the actual detection of the viruses is done.

A second possible scenario is the local device conducting the major part of the calculations until a text file with virus candidates and features is calculated. Only this text file is transferred to the server where the classification of the virus candidates is done. Afterward the result is transferred back to the device.

For each of the possible cut off points, the corresponding energy consumption and run time is calculated on all the considered GPUs. As a result, for each GPU a Pareto front with the trade-offs is received, which is used to decide whether and at which cut off point the calculation should be offloaded. The decision can be based on the current demands for run time and energy consumption on each individual mobile device.

To explore larger parameter spaces than the presented ones, an extensively modified version of ECJ (A Java-based Evolutionary Computation Research System) [14] can be used to optimize the parameters with an evolutionary algorithm. The multi-objective evaluation is done with SPEA2 [15], which is included in ECJ. Details of this approach can be found in [16].

TABLE I. EVALUATED MOBILE GPUS.

GPU	#SPs	Core clock	DRAM	DRAM clock	#ROPs
520M	48	740 MHz	GDDR-3	800 MHz	4
540M	96	670 MHz	GDDR-3	900 MHz	8
560M	192	760 MHz	GDDR-5	1250 MHz	16
580M	384	620 MHz	GDDR-5	1500 MHz	32

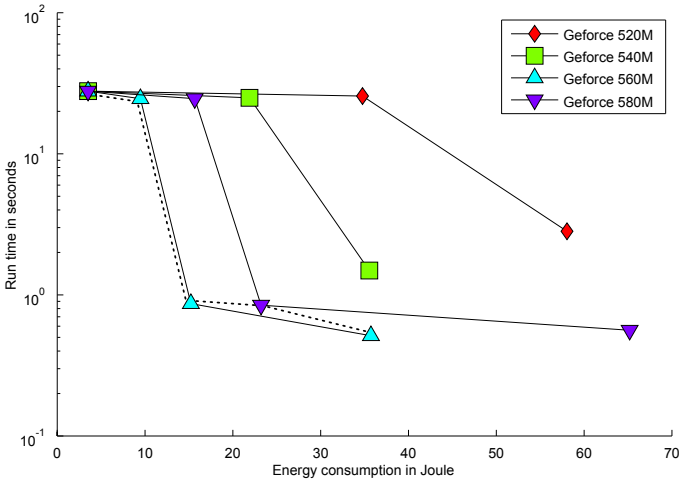

Fig. 3. Results of the multi-objective evaluation. Energy consumption versus the run time for four different GPUs, plotted as Pareto fronts. The overall Pareto front is plotted as a dotted line.

IV. RESULTS

The results for the multi-objective evaluation are presented in Figure 3 and Table II. The figure shows four Pareto fronts for the evaluated GPUs and the table shows the corresponding cut off points and values for energy consumption and run time. The overall Pareto front is indicated by the dotted line in the figure and by an italic font in the table.

The points of the overall Pareto front are the point where no mobile GPU is used with 3.5 Joule and a run time of 27.8 seconds, the three points of the Geforce 560M with an energy consumption of 9.5 Joule and 24.6 seconds, 15.2 Joule and 0.87 seconds and 35.7 Joule and 0.51 seconds, and the point of the Geforce 580M with an energy consumption of 23.2 Joule and a run time of 0.85 seconds. Accuracy of the simulators has already been shown in [11] and [13] and is therefore not within the scope of this paper.

Offloading all the processing and not using the mobile GPU is part of all Pareto fronts and is shown as the top left points in Figure 3 and the last line in Table II. This configuration has the lowest energy consumption but also the highest run time. Doing all the calculation on the mobile GPU and not using the server, was part of the Pareto fronts for the 560M and 580M but not for the 520M and 540M. These configurations result in a low run time, but also a high energy consumption as shown in the first two lines in Table II and the two bottom points in the figure.

Intermediate cut off points were identified for all GPUs as well. These correspond to the configurations where partial results are calculated locally, and the server is used to calculate the final result. The intermediate cut off points are of particular

TABLE II. ENERGY CONSUMPTION AND RUN TIME FOR DIFFERENT CUT OFF POINTS AND DIFFERENT GPUS. THE OVERALL PARETO FRONT IS HIGHLIGHTED IN ITALIC FONT.

Cut off point	GPU	Energy consumption	Run time
None, no offloading	*560M*	*35.72 Joule*	*0.51 s*
	580M	65.18 Joule	0.56 s
After pixel candidates are calculated	520M	58.0 Joule	2.82 s
	540M	35.54 Joule	1.49 s
	560M	*15.22 Joule*	*0.87 s*
	580M	*23.21 Joule*	*0.85 s*
After background and noise elimination is calculated	520M	34.76 Joule	25.67 s
	540M	21.89 Joule	24.94 s
	560M	*9.47 Joule*	*24.59 s*
	580M	15.66 Joule	24.6 s
Full offloading	*None*	*3.51 Joule*	*27.82 s*

interest, as energy can be saved with only a slight increase in run time and vice versa.

V. CONCLUSION AND FUTURE WORK

Using mobile biosensors for fast and reliable measurements, several conflicting hardware and software demands have to be met. The presented automatic design space exploration can be used to identify different efficient hardware/software configurations for the selected objectives. In the evaluated biosensor setup with energy consumption as the main objective, 90% energy could be saved compared to the fastest calculation. With overall time as the main objective, a speedup of 55 could be achieved compared to the most energy efficient configuration.

In other offloading approaches [6], [7], [8], [9], offloading tasks is only a binary decision. The task is either offloaded or not. Here multiple cut off points were inspected and it could be shown that it can be beneficial to offload only parts of the calculation. Of particular interest is the case, where a slight increase in run time from 0.51 to 0.87 seconds, by transferring a small amount of data, can reduce the energy consumption by 57% for the 560M GPU and with a similarly small increase in run time by 64% for the 580M GPU.

A somehow surprising result is, that the most power efficient GPUs in the example are not on the Pareto front at all. Although the faster GPUs use more power, the result is calculated faster and thus they need less energy. Another interesting result is, that the fastest GPU only provides one point to the overall Pareto front. This indicates that the program can not fully utilize the 384 cores of this GPU. By inspecting the calculated utilization of the GPU, the application designer can easily identify and remedy bottlenecks.

As future work, the offloading will be extended to use also other devices in the local area or devices in other areas to process data. If one device has low battery power, insufficient computing capacity or the server is currently unavailable, the work can be offloaded to another mobile device, which has more battery power and sufficient computing capacities. Finally, different data rates and location scenarios for the LTE communication will be evaluated, such that the offloading automatically adapts to the environment.

ACKNOWLEDGMENT

Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, projects A4 and B2. URL: http://sfb876.tu-dortmund.de

REFERENCES

[1] L. Baril, *Need for Biosensors in Infectious Disease Epidemiology*. John Wiley & Sons, Ltd, 2008.

[2] A. Zybin and et al., “Real-time detection of single immobilized nanoparticles by surface plasmon resonance imaging,” *Plasmonics*, vol. 5, pp. 31–35, 2010.

[3] P. Libuschewski, D. Siedhoff, C. Timm, A. Gelenberg, and F. Weichert, “Fuzzy-enhanced, real-time capable detection of biological viruses using a portable biosensor,” in *Biomedical Engineering Systems and Technologies, 2013. Proceedings of the International Joint Conference on*, February 2013.

[4] S. Ou, K. Yang, and A. Liotta, “An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems,” in *Pervasive Computing and Communications, 2006. Fourth Annual IEEE International Conference on*, March 2006, pp. 1–10.

[5] K. Yang, S. Ou, and H.-H. Chen, “On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications,” *Comm. Mag.*, vol. 46, no. 1, pp. 56–63, 2008.

[6] K. Kumar and Y.-H. Lu, “Cloud computing for mobile users: Can offloading computation save energy?” *Computer*, vol. 43, no. 4, pp. 51–56, April 2010.

[7] Z. Li, C. Wang, and R. Xu, “Computation offloading to save energy on handheld devices: A partition scheme,” in *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*. ACM, 2001, pp. 238–246.

[8] C. Xian, Y.-H. Lu, and Z. Li, “Adaptive computation offloading for energy conservation on battery-powered systems,” in *Parallel and Distributed Systems, 2007 International Conference on*, vol. 2, Dec 2007, pp. 1–8.

[9] A. Toma and J.-J. Chen, “Computation offloading for frame-based real-time tasks with resource reservation servers,” in *Real-Time Systems, 2013. 25th Euromicro Conference on*. IEEE, 2013, pp. 103–112.

[10] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, “The remote processing framework for portable computer power saving,” in *Applied Computing. Proceedings of the 1999 ACM Symposium on*, 1999, pp. 365–372.

[11] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, “How a single chip causes massive power bills GPUSimPow: A GPGPU power simulator,” in *Performance Analysis of Systems and Software, 2013. IEEE International Symposium on*. IEEE, 2013, pp. 97–106.

[12] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, “Analyzing CUDA workloads using a detailed GPU simulator,” in *Performance Analysis of Systems and Software, 2009. IEEE International Symposium on*, 2009, pp. 163–174.

[13] B. Dusza, C. Ide, L. Cheng, and C. Wietfeld, “CoPoMo: a context-aware power consumption model for LTE user equipment,” *Emerging Telecommunications Technologies. Transactions on*, vol. 24, no. 6, pp. 615–632, 2013. [Online]. Available: http://dx.doi.org/10.1002/ett.2702

[14] S. Luke and L. Panait, “A survey and comparison of tree generation algorithms,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 81–88.

[15] E. Zitzler, K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, T. Fogarty, E. Z. Ler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization,” in *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, 2001. Proceedings of the International Conference on*, 2001.

[16] P. Libuschewski, D. Siedhoff, and F. Weichert, “Energy-aware design space exploration for GPGPUs,” *Computer Science - Research and Development*, pp. 1–6, 2013.