Comparison of GPU and FPGA Implementation of SVM Algorithm for Fast Image Segmentation

Marcin Pietron, Maciej Wielgosz, Dominik Zurek,
Ernest Jamro, and Kazimierz Wiatr

AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Krakow
ACK Cyfronet AGH
ul. Nawojki 11, 30-950 Krakow
{pietron,wielgosz,jamro,wiatr}@agh.edu.pl
dominik.zurek1102@gmail.com

Abstract. This paper presents preliminary implementation results of the SVM (Support Vector Machine) algorithm. SVM is a dedicated mathematical formula which allows us to extract selective objects from a picture and assign them to an appropriate class. Consequently, a black and white images reflecting an occurrence of the desired feature is derived from an original picture fed into the classifier. This work is primarily focused on the FPGA and GPU implementations aspects of the algorithm as well as on comparison of the hardware and software performance. A human skin classifier was used as an example and implemented both on Intel Xeon E5645.40 GHz, Xilinx Virtex-5 LX220 and Nvidia Tesla m2090. It is worth emphasizing that in case of FPGA implementation the critical hardware components were designed using HDL (Hardware Description Language), whereas the less demanding or standard ones such as communication interfaces, FIFO, FSMs were implemented in Impulse C. Such an approach allowed us both to cut a design time and preserve a high performance of the hardware classification module. In case of GPU implementation whole algorithm is implemented in CUDA.

Keywords: SVM, image segmentation, FPGA, GPU, CUDA.

1 Introduction

This work is part of the Synat project embracing several initiatives aiming to create a repository of images which are assigned a descriptive name according to their contents. Such a database of tagged images will significantly reduce search time since only picture tags will be processed instead of images so the process will involve simple string operations rather than image recognition.

The project is a huge challenge due to an immense volume of data collected over the past years denoted today as the Internet resources. Therefore the core part of the undertaking is to design and implement a classification system which should be both reliable and fast. In order to achieve the high performance of a search engine the most computationally intensive operations are to be ported to hardware. Thus FPGAs and GPUs due to their strongly parallel structure and growing processing speed [1] seem to be the best choice.

Image segmentation is a process which aims to separate a picture into several regions based on objects or features of interest. A single SVM system may embrace several modules trained to recognize different features so the unit as a whole is capable of tracing multidimensional objects in terms of a number of features.

It is worth emphasizing that a segmentation may also be regarded as a form of data compression, the classfier accepts images and yields information regarding objects which usually occupies much less memory resources than corresponding original data.

There are plantiful image segmentation algorithms [2,3,4] and their number is still growing to meet constantly rising demands of data analysis systems. However, reliability and data processing speed are the factors which are at a premium when it comes to a real life application of a given algorithm. SVM meets both those criterions and therefore was chosen as a classification algorithm for the project.

2 SVM Classifiers

Support vector machines were originally devised and described by Vapnik [5,6]. They are used for binary classification which means that there are exactly two classes of objects (e.g. black and white rectangles) and a classification formula is found in a training process of the classifier.

The SVM algorithm can be envisioned as a process of creating a hyperplane which separates data in an n-dimensional space. It is conducted in an iterative manner in which a selected plane is gradually adjusted to provide the optimal so-called generalization margin. The following cases may occur:

- Input data is linearly separable and the SVM method guaranties that at least one plane of the best separation margin exists and will be adopted
- A dimension incrementation is used to bring data to a space of more dimensions so a separation plane can be found

A feature space for 2D can be modeled as a sphere with its center and radius (see Fig. 1).

The sphere is build upon a set of supportive vectors which constitute its structure. A classification process of a priorly trained SVM maybe perceived as probing whether a given point (input data) belongs to the sphere or it's located outside of it. In the first case a point is positively classified whereas in the second one it's considered to be an outliner.

3 A Choice of a Hardware Platform

It is very important to choose a proper hardware units and appropriate data transfer protocol since it affects the overall performance of the computational

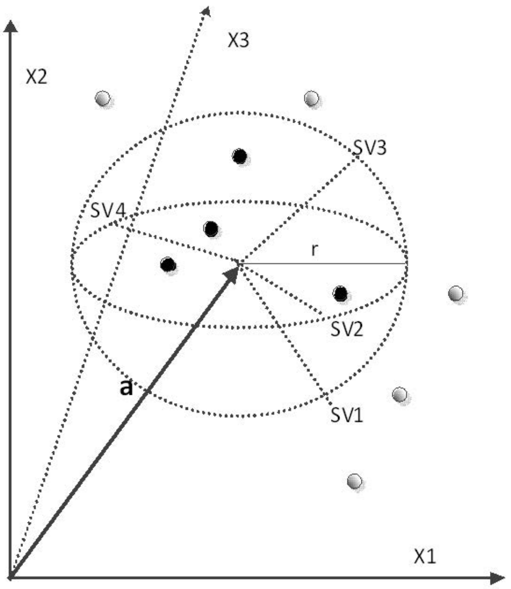

Fig. 1. N-sphere

system. It also may balk an effort invested in the development of the hardware algorithm. Therefore the authors decided to compare FPGA and GPU implementations in order to be able to choose the most appropriate solution for the complete system realization.

3.1 FPGA

FPGAs have been developed since late 1980s, and have a lot of advantages over processors. The most important ones are: massive parallel architecture, reconfigurability, low energy consumption, ability to shape freely its internal architecture.

Design and effective use of computing system based on FPGA is a difficult task, as evidenced by the long history of such trials. Existing HPRC (High Performance Reconfigurable Computing) solutions can be classified based on their integration with other computing nodes in the system. However, the authors have used DRC platform for the implementation of the SVM algorithm.

The architecture of Accelium (Xilinx Virtex-5) [12] is based on an idea of populating CPU sockets with FPGAs and the fast communication link between them. This architecture allows for equal access of processor and FPGA to the system resources.

3.2 GPU

The architecture of a GPU card is described in Fig. 2. The graphical processor unit has a multiprocessor structure. In Fig. 2 is shown N multiprocessor GPU with M cores each. The cores share an Instruction Unit with other cores in a multiprocessor. Multiprocessors have special memories which are much faster than global memory which is common for all multiprocessors. These memories are: read-only constant/texture memory and shared memory. The GPU cards are massive parallel devices. They enable thousands of parallel threads to run which are grouped in blocks with shared memory. The blocks are grouped in a grid (Fig. 3). CUDA is a software architecture that enables graphics processing unit (GPU), to be programmed using high-level languages such as C and C++. CUDA requires an NVIDIA GPU like Fermi, GeForce 8XXX/Tesla/Quadro, and so on. CUDA provides three key mechanisms to parallelize programs: thread group hierarchy, shared memories, and barrier synchronization. These mechanisms provide fine-grained parallelism nested within coarse-grained task parallelism.

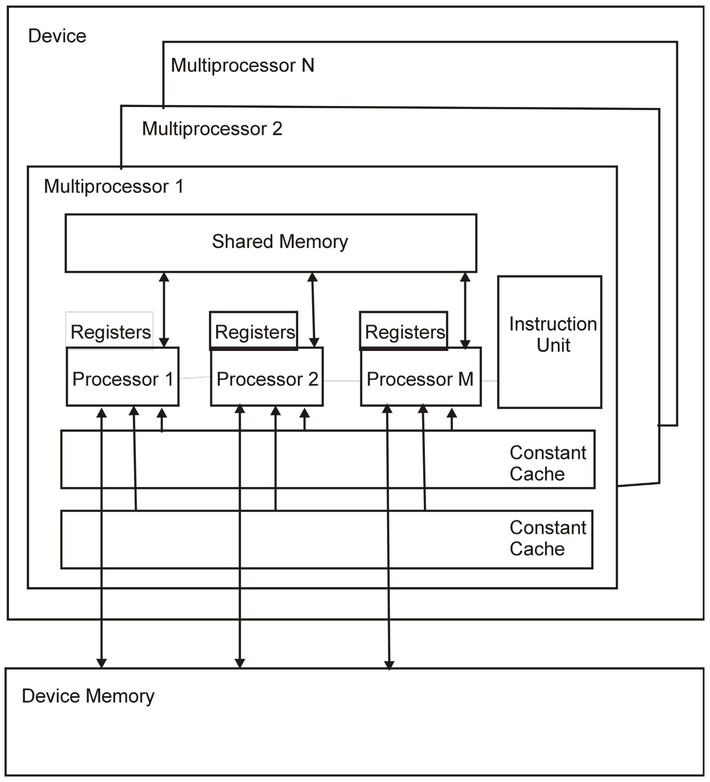

Fig. 2. GPU card architecture

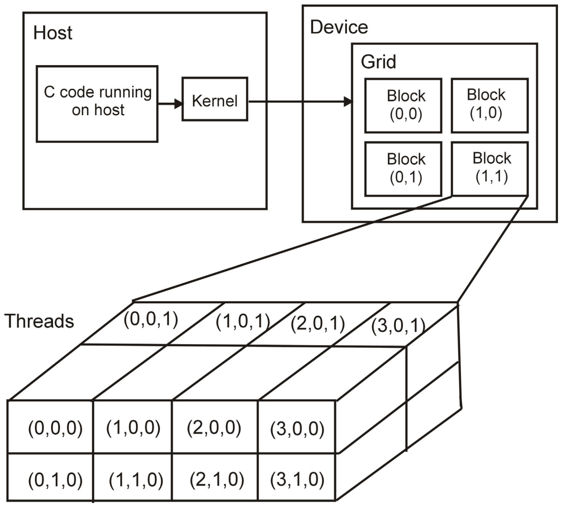

Fig. 3. Relation between grid, blocks, and threads

Creating the optimized code is not trivial and thorough knowledge about the GPUs architecture is needed. The main aspects are the usage of the memories, an efficient dividing code to parallel threads and thread communications. As was mentioned earlier constant/texture and shared memories are the fastest. Therefore programmers should optimally use them to speedup access to data on which an algorithm operates. Another important thing is to optimise synchronization and the communication of the threads. The synchronization of the threads between blocks is much slower than in a block. If it is not necessary it should be avoided.

4 System Overview

A human skin classifier OC-SVM (One Class Supportive Vector Machine) was implemented as a preliminary project which allows us to estimate performance and resource consumption for other classifiers. As a result of an experiment a black-and-white image is generated which reflect human skin location in the original picture which was fed into the classifier. A complete computational procedure is composed of several steps:

- SVM vectors and τ generation (training of the classifier)
- Input image fetch (the step is different for hardware and software implementation)
- Image resize and normalization
- Classification
- Noise and skin-like objects filtration

4.1 The Classification Algorithm

The classification algorithm is given by the following formula:

$$\sum_i \alpha_i K(X_x, X_i) \geq \sum_i \alpha_i K(X_s, X_i) = \tau \qquad (1)$$

where τ is the sphere radius, X_s and α_i are supportive vectors derived in a training process, X_x is an input pixel.

Regardless of a choice of vectors in right side of the equation 1 the result is constant and equals τ. Each pixel fed into the classifier is compared against all the support vectors in order to determine if it is located inside the sphere (see Fig.1).

In this implementation a Gaussian computational kernel was used:

$$K = e^{-\gamma \|X_i - X_j\|^2} \qquad (2)$$

where γ is a spread of the kernel.

If the (1) is met a given point is classified as belonging to the desired class. For SVM classifies the best results are achieved when input data is normalized (i.e. fall in the range [-1;1]).

4.2 Architecture of the Hardware Module in FPGA

The computationally intensive routines were ported to hardware to offload the GPP (General Purpose Processor) and to accelerate the computations. It was possible due to several features of the algorithm which makes it well suited for the FPGA implementation such as: fixed-point arithmetic, parallel structure (easy to pipeline), narrow-range input argument. Consequently a series of hardware units were designed which constitute the internal structure of computational module as presented in Fig.4. All the modules are parameterized and pipelined blocks which process a single input vector X every clock cycle. For a sake of the software compatibility the base data format employed in the application is 32 bit fixed-point (16 bits of both fractional and integer part) but it can be adjusted to meet different precision requirements in the future. Each module is equipped with the overflow signal which propagates across all the units composing the classification module. Such an approach allows us to avoid corruptions of the result just by simply examining the overflow output.

It is possible to connect several classification modules to form a parallel structure. Furthermore, it is worth noting that supportive vectors (denoted as SV in Fig. 5) are fetched from an external memory only once for the whole computations and therefore can be stored in the internal memory for all the computation. Moreover, the number of the supportive vectors as well as a is not large and usually not exceed tens, thus internal BRAM memory suffice to accommodate those coefficients (e.g. for the human skin classification only 16 supportive vectors are used). Increase of a number of supportive vectors improves the classifier accuracy at the expense of the accumulator throughput decrease (see Fig.4) which in turn affects an overall system performance.

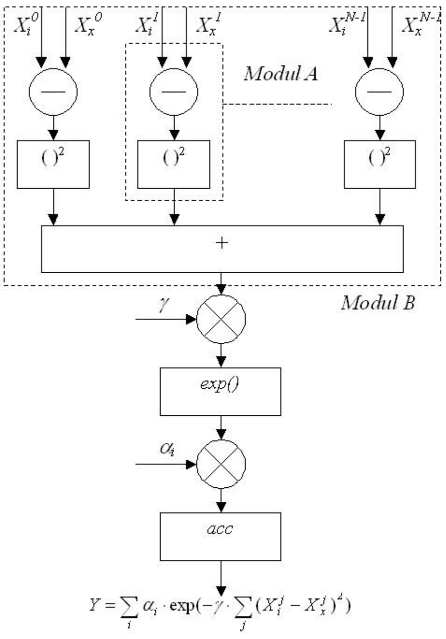

Fig. 4. Block diagram of the classification module

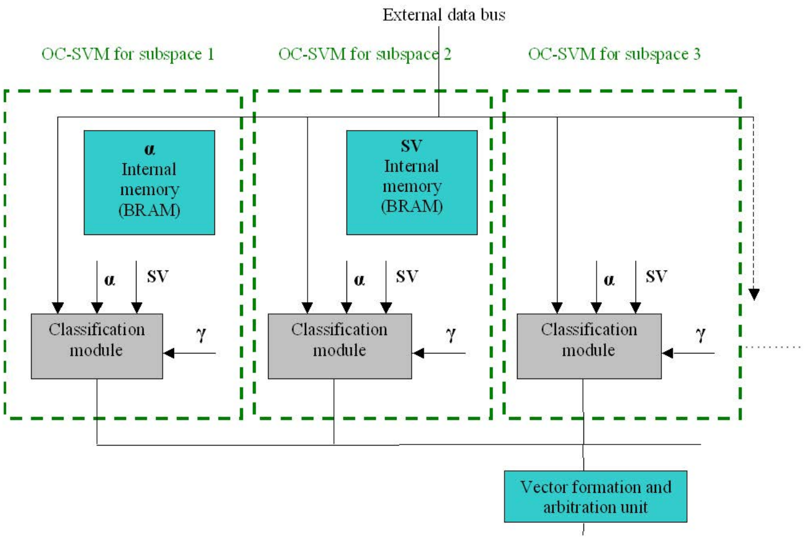

Fig. 5. Block diagram of the multimodule structure

The classifier (presented in Fig. 4) yields one bit results which reflects an occurrence of a feature of interest within an image. Therefore in order to take a full advantage of an external bus throughput, classification results are compacted into 32 bit bundles and sent to a host processor as such. Thereafter the GPP transforms those binary values into pixels to form a black-and-white image depicting the features of interest.

4.3 Architecture of the Implementation in GPU

In case of GPU implementation whole image is transfered to global memory. Each pixel is computed by single thread. Therefore as it is shown in Fig.6 in each block 512 pixels are processed. Each thread reads value of pixel form global memory, then computes SVM classifier formula (1) for one point of image and writes result back to shared memory (at the same location as pixel value). Apart from values of points of image shared memory stores sphere radius and supportive vectors derived from training process needed to classify each pixel. In this case each shared memory contains 512*3 bytes of image, 16*3 bytes of supportive vectors and 16 bytes of sphere radius (Fig.6). When the number of points exceeds 512 then they are divided to multiple blocks of GPU card.

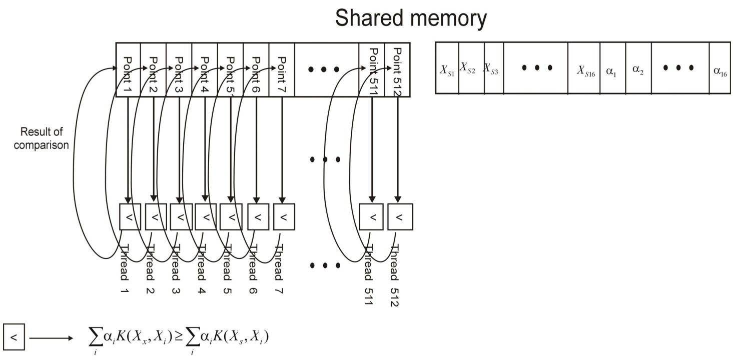

Fig. 6. Diagram of the SVM implementation in GPU

5 Implementation Results

The classification algorithm was initially implemented on GPP in C++ and the OpenCV library was used. A set of 16 support vectors was generated which described a human skin, each of which are 32bit RGB colors.

Fig. 7 and 8 represent experimental results for the randomly chosen images. It can be noticed that the system wrongly classified some parts of the image. Unfortunately the system often confuses bright objects with a human skin. One way to improve the accuracy is increasing the contrast between an object and a background. Similar result of accuracy improvement may be achieved when a

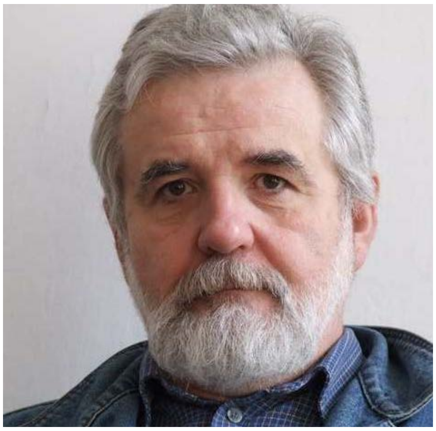

Fig. 7. Original image (before segmentation)

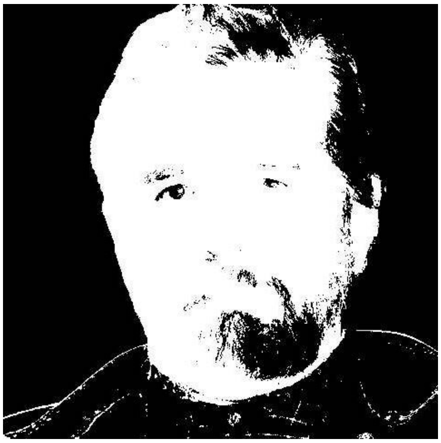

Fig. 8. Results of human skin segmentation

larger number of supportive vectors is employed but it is done at a expanse of a loss of classifier's generalization feature.

Time required to execute the following algorithm on GPP, FPGA and GPU is presented in Tab. 2 and Fig.9 . They show that in case of less number of pixels FPGA platform is faster than CPU and GPU. When more points are processed the GPU card is fastest and CPU is the slowest execution unit. In case of more points GPU cards takes advantage of its massive parallel architecture. Therefore it is faster than pipelined FPGA architecture. FPGA hardware implementation according to the formula (1) (assuming that no input data fetch delay is introduced) can be calculated as follows: $16(SVM) \times 480(pixels) \times 480(pixels) \cong 4 \times 10^6$ clock cycles. Consequently theoretical processing time for 200 MHz equals 0.02s. Due to a low resources consumption a single FPGA can accommodate several modules which boost a performance several times. The power consumption in case of FPGA (Virtex-5 LX220) is about 15 watts, GPU (Nvidia Tesla m2090) consumption is 250 watts. The implementation results of the module on DRC AC2020 [12] were presented in Tab. 1. Tranfer times are described in Tab. 3.

It is worth noting that a number of coefficients has a large impact on the resources occupation in case of FPGA. On the GPU platform coefficients occupied very small part of shared memory. In this particular implementation the number of the coefficient is three (R, G, B).

Table 1. Implementation results of the module building block in Impulse C (see Fig.4)

# 4-input LUT	# flip-flops	# BRAM
122,637[83%]	59,208[42%]	2,049[6%]

Table 2. Implementation results

Number of pixels	# GPU [ms]	# CPU [ms]	# FPGA [ms]
2048	0,6	1,82	0,16896
10240	0,74	36,1	0,8448
51200	1,58	180,7	4,224
204800	5,13	714	16,896
512000	12,15	1813,54	42,24

Table 3. Transfer time

Number of pixels	# GPU [ms]	# FPGA [ms]
2048	0,45	1,12
10240	0,49	1,22
51200	0,66	1,65
204800	1,3	3,25
512000	2,6	6,5

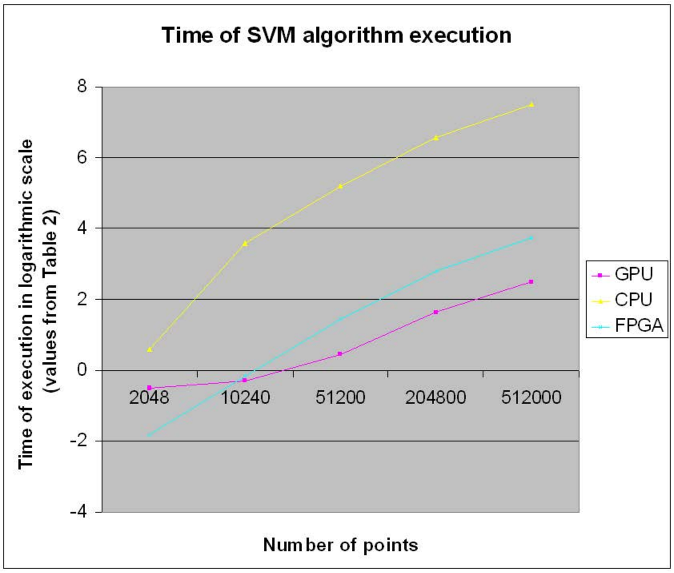

Fig. 9. Chart with implementation results

The classification module in case of FPGA is a fully pipelined structure and it is capable of working at the frequency of 200 MHz. Each module generates a single result every n clock cycles where n denotes number of the support vectors employed.

Support vectors along with α,γ,τ are generated on the host side (by GPP) and are sent to the FPGA and GPU only once for the whole computations. The both implementations perform the classification for all the X vectors and sends the results back to the host processor.

6 Summary

In this paper implementation results of the selected parts of the fast image segmentation were presented along with some performance analysis. Both FPGA and GPU implementation were discussed. GPU significantly surpasses FPGA and GPP in the high volume calculations. It is worth emphasizing that SVM can be easily parallelized due to its structure which makes it an ideal candidate for GPU implementation.

Acknowledgments. The work presented in this paper was financed through the research program - *Synat*.

References

1. Mueller, R., Teubner, J., Alonso, G.: Data Processing on FPGAs. In: Systems Group, Department of Computer Science, VLDB 2009, ETH Zurich, Switzerland, Lyon, France, August 24-28 (2009)
2. Jun, T.: A color image segmentation algorithm based on region growing. In: 2010 2nd International Conference on Computer Engineering and Technology (ICCET), April 16-18, vol. 6, pp.V6-634–V6-637 (2010)
3. Farmer, M.E., Jain, A.K.: A wrapper-based approach to image segmentation and classification. IEEE Transactions on Image Processing 14(12), 2060–2072 (2005)
4. Lan, Y., Li, C., Zhang, Y., Zhao, X.: A novel image segmentation method based on random walk. In: Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PACIIA 2009, November 28-29, vol. 1, pp. 207–210 (2009)
5. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer (2000)
6. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.N.: A support vector clustering method. In: Proceedings of the 15th International Conference on Pattern Recognition, vol. 2, pp. 724–727 (2000)
7. http://www.alpha-data.com/
8. http://www.nallatech.com/
9. http://www.picocomputing.com/
10. Baxter, R., Booth, S., Bull, M., Cawood, G., Perry, J., Parsons, M., Simpson, A., Trew, A., McCormick, A., Smart, G., Smart, R., Cantle, A., Chamberlain, R., Genest, G.: Maxwell - a 64 FPGA Supercomputer. In: Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), pp. 287–294 (2007)
11. www.silicongraphics.ru/pdf/rasc_data.pdf
12. www.drccomputer.com/pdfs/DRC_Accelium_Overview.pdf
13. www.vhdl.org/fphdl/
14. Wielgosz, M., Jamro, E., Wiatr, K.: Hardware Implementation of the Exponent Based Computational Core for an Exchange-Correlation Potential Matrix Generation. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2009, Part I. LNCS, vol. 6067, pp. 115–124. Springer, Heidelberg (2010)