Resource prediction and quality control for parallel execution of heterogeneous medical imaging tasks

Citation for published version (APA):
Albers, A. H. R., Suijs, E., & With, de, P. H. N. (2009). Resource prediction and quality control for parallel execution of heterogeneous medical imaging tasks. In *Proceedings of the 16th IEEE International Conference on Image Processing, ICIP, 7-10 November, 2009, Cairo, Egypt* (pp. 2317-2320). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ICIP.2009.5414222

DOI:
10.1109/ICIP.2009.5414222

Document status and date:
Published: 01/01/2009

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

RESOURCE PREDICTION AND QUALITY CONTROL FOR PARALLEL EXECUTION OF HETEROGENEOUS MEDICAL IMAGING TASKS

Rob Albers [a,b], *Eric Suijs* [b] *and Peter H.N. de With* [a,c]

[a] Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands,
[b] Philips Healthcare, X-Ray R&D, PO Box 10.000, 5680 DA Best, The Netherlands,
[c] CycloMedia Technology, PO Box 68, 4180 BB Waardenburg, The Netherlands.

ABSTRACT

We have established a novel control system for combining the parallel execution of deterministic and non-deterministic medical imaging applications on a single platform, sharing the same constrained resources. The control system aims at avoiding resource overload and ensuring throughput and latency of critical applications, by means of accurate resource-usage prediction. Our approach is based on modeling the required computation tasks, by employing a combination of weighted moving-average filtering and scenario-based Markov chains to predict the execution. Experimental validation on medical image processing shows an accuracy of 97%. As a result, the latency variation within non-deterministic analysis applications is reduced by 70% by adaptively splitting/merging of tasks. Furthermore, the parallel execution of a deterministic live-viewing application features constant throughput and latency by dynamically switching between quality modes. Interestingly, our solution can successfully be reused for alternative applications with several parallel streams, like in surveillance.

Index Terms— Medical image processing, Object recognition, Software performance, Stochastic approximation, Multiprocessing.

1. INTRODUCTION & MOTIVATION

In advanced systems, multiple video applications are executed in parallel and share the constrained system resources. To optimize quality and fluent execution of tasks, management and quality control are required to avoid resource overload and guarantee throughput of critical applications. With dynamic video-processing applications, such as in image analysis, the computational complexity has become data dependent and memory usage is more irregular. Detailed know-how of specific application aspects, such as data-driven complexity and the corresponding memory requirements is relevant for optimal mapping of tasks on a computing platform and optimizing the performance during runtime. In this paper, we use modeling for runtime estimation of the resource usage with the aim to execute more functions in parallel on the same platform. With accurate model descriptions, at runtime, a resource-usage prediction can be made for resource planning, parallelization and possibly the corresponding Quality-of-Service (QoS) control [1][2].

The above problem statement has been addressed in several fields of applications, such as high-performance computing and multimedia. For multimedia computing, application analysis has revealed that algorithms have variable computational rates and memory sizes, which involves application tuning based on this variable nature [3][4]. In our case, the application field is professional medical imaging and processing. Since this involves a rather broad scope, we study a mixture of the previous features and new aspects. Furthermore, the application involves image analysis, which has a more dynamic nature and will be discussed below.

This paper is organized as follows. In Section 2, the architectural requirements are given. Section 3 presents the application under study. The method for resource-usage prediction is described in Section 4. For management of the resources and quality, in Section 5, we introduce scalability and runtime control of the application. Section 6 presents experimental results executing two applications in parallel on a multiprocessor system. Section 7 concludes the paper.

2. ARCHITECTURAL REQUIREMENTS

Let us start with a survey of the system architectural requirements that are important for professional medical imaging.
• *Low latency*. We explore a double-pipeline imaging application with the aim to (1) detect and enhance objects of interest under X-ray fluoroscopy and (2) denoise and increase the global image quality during a live interventional angiography procedure. Because physicians must see their actions directly on the screen (eye-hand coordination), a constant low latency is a key requirement for the real-time imaging application.
• *Variable image analysis*. A trend in medical imaging is the introduction of analysis and feature extraction techniques in the real-time video pipeline. As a consequence, the computational complexity has become data dependent. As an answer to the variable processing rates, performance prediction may be applied in the form of modeling to guide the mapping and to obtain efficient implementations.

978-1-4244-5654-3/09/$26.00 ©2009 IEEE

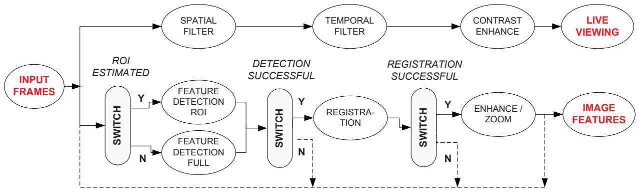

Fig. 1. Flow graph with deterministic and non-deterministic tasks.

As dedicated hardware lacks flexibility, using of-the-shelf multi-core processors as a platform for image processing seems to be a valid choice. With multiple cores to provide the required processing power, runtime programmability and flexibility is maintained. Co-processors can be added in the future for cost-efficiency reasons. It is important to note that the approach we employ in this paper is essentially different from considering only the worst-case state of the system. Unlike the worst-case approach, our approach is dynamic, i.e. it makes use of runtime characteristics of the input data and the environment of the application.

3. MEDICAL IMAGING APPLICATION

This section presents the key application used in this paper. Coronary angioplasty is a catheter-based procedure performed by an interventional cardiologist in order to open up a blocked coronary artery and restore blood flow to the heart muscle. Image analysis and motion-compensation techniques can improve the visualization and measurement of objects of interest (such as stents) in X-ray angiography, thereby making it easier to realize optimum and complete stent placement with an X-ray intervention. Additionally, we explore a medical-imaging application to detect and enhance moving features, combined with several tasks for increasing the image quality during a live interventional angioplasty procedure. We assume a double-pipeline processing approach, see Fig. 1. Spatial filtering uses directional filter kernels that preserve the local edges while removing noise (anisotropic filtering). For temporal filtering, a recursive filter is employed and the contrast enhancement function is based on look-up tables. In a second processing branch, image analysis and motion-compensation tasks detect and enhance specific objects of interest. Whereas the first branch contains *stream-oriented* tasks having a *deterministic* nature in terms of computations and memory, the second branch contains *data-dependent non-deterministic* processing tasks.

The dynamics in the application come from three major aspects: (1) At the start, a Region-Of-Interest (ROI) of variable data-dependent size is chosen for further analysis, and (2) at every stage, switch functions internally select a specific flow graph, depending on the previous stage(s). Moreover (3), some of the internal flow graphs require intrinsically a variable processing time. Tasks in the image analysis cannot be easily switched off, since that would lead to an incomplete or unacceptable result. The presented flow graph is based on a cascade of stages which are individually described in [5, 6, 7, 8]. Summarizing, our application contains two parallel branches of different nature and the complete infrastructure should operate in real time. In the next section, we discuss the prediction model for the computation time, including the estimation technique coping with the dynamic behavior.

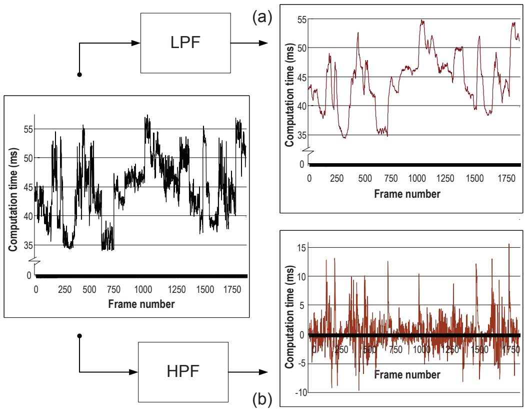

Fig. 2. Splitting the computational statistics in structural (a) and short-term (b) behavior.

4. RESOURCE-USAGE PREDICTION

To handle the variable processing rates of the application, performance prediction is applied in the form of *modeling* to guide the mapping and obtain an efficient implementation. Modeling of non-deterministic resource usage is complicated because depending on the image content and intermediate analysis result, the analysis algorithm may switch to a different group of processing tasks. We have considered several options for modeling of the computation time. As a first solution, we investigated literature on video traffic modeling [9]. Most of the papers deal with Markov-chain approaches, since the estimation of the model parameters is straightforward and there is a large number of analysis techniques available.

An alternative for modeling of the system behavior is to classify the timing statistics of the video frames in two categories, as a result from mapping the algorithms on a platform. Hence, we then investigate short-term and structural fluctuations in processing time on the platform. Short-term fluctuations can be caused by cache misses or the overhead imposed by task switching and control. Structural fluctuations in processing time are caused by the dependency on the video content itself over a sequence of images of several seconds.

As a consequence of the previous discussions, we have adopted a concept where the long-term statistics are decoupled from the short-term stochastic behavior, by employing different models for those statistics (Fig. 2). We consider the prediction model to consist of long-term low-frequency fluctuations, around which short-term high-frequency fluctuations can take place. Discriminating between the low and high-frequency part can be implemented with Finite Impulse

Response (FIR) or Infinite Impulse Response (IIR) filters. We apply the Exponentially Weighted Moving Average (EWMA) filter, as this IIR filter adapts more quickly to the input signal compared to FIR filters. The EWMA filter is defined by the well-known IIR filter equation:

$$y(t_k) = (1 - \alpha) \times y(t_{k-1}) + \alpha \times x(t_k). \qquad (1)$$

The prediction model for the short-term *data-dependent* tasks is described by a probabilistic process that can be covered with a finite-state Markov chain. A first-order Markov chain is by definition memoryless, where in the model it is implicitly assumed that the processing times of successive frames are independent. Based on the exponentially decaying autocorrelation function, we have concluded that the short-term fluctuations can be successfully modeled with Markov chains. The Markov state-space description can be generated by analyzing the *computation time* C over a long time period. The number of states M is C_{max}/σ_C, where C_{max} denotes the largest measured value and σ_C the standard deviation. We have experimentally evolved to a model with approximately $2M$ states to obtain sufficient accuracy. The quantization intervals are adaptively chosen such that each interval contains on the average the same amount of samples. The entries of the transition probability matrix $\{P_{ij}\}$ are estimated by

$$P_{ij} = n_{ij}/(\sum_{k=1}^{M} n_{ik}), \qquad (2)$$

where n_{ij} denotes the number of transitions from interval *i* to interval *j*. These entries are inserted into an $N \times N$ transition matrix, $Q = [P_{11}, P_{12}, ..., P_{NN}]$.

5. MANAGING RESOURCES AND QUALITY

Since our aim is to execute more functions on the same platform with a guaranteed throughput, we use the model descriptions as a prediction for parallelization and quality control. For quality control, several options exist. Within our application, data-dependent switch statements occur (Fig. 1), which can cause the total processing time to change rather abruptly. However, during a live interventional X-ray procedure, large latency differences between succeeding frames are not allowed for clinical reasons (eye-hand coordination of the physician). A straightforward solution is to employ an application task partitioning on the platform, based on worst-case resource usage with a delay function at the end of the pipeline. The main drawback is that for most of the time, the reserved resource budget is set too conservative. Moreover, it is impossible to exploit the difference between average-case and worst-case requirements without affecting the reliability.

Another approach for varying latency is to use the prediction models from Section 4 to dynamically switching on/off non-essential tasks at runtime, thereby preserving a fluent output rate of both pipelines at the expense of a variable quality.

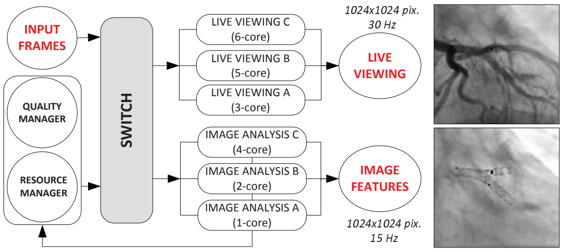

Fig. 3. Resource and quality control architecture for live viewing and feature analysis.

Using the models, we are able to accurately predict how many resources are required. This information can be used by a runtime manager for on-the-fly switching between groups of tasks. This approach is dynamic, i.e., it makes use of runtime characteristics of the input data and the environment of the application. The approach consists of several steps.

• *Initialization.* By processing the first frame of the sequence, we initialize the partitioning of the flow graph based on the image characteristics. The output latency is set to an initial value (close to the average case), which will be our latency *budget* during runtime.

• *Runtime adaptation.* Based on the outcome from the resource predictions for subsequent frames, the resource manager determines if for the *next* frame, more resources are required than available on the system. If so, the live-viewing graph is degraded to a lower quality and resource-demanding mode, by switching off (some of) the non-essential filtering tasks. This releases system resources that will be allocated to the image analysis tasks in order to maintain the latency target by splitting tasks and executing them in parallel.

• *Profiling.* The application can be profiled to gather statistical information of the differences between the actually consumed resources and the predicted values. The information can be used for on-line model training, or to give insight information about the prediction quality of the model.

Summarizing, the resource manager controls the firing of non-essential tasks and guarantees the essential tasks of both applications. The quality manager ensures that a combination of pipelines is chosen such that the application throughput is guaranteed with a certain quality.

6. EXPERIMENTAL RESULTS

For training the prediction models, we have used a data set of 37 video sequences of in total 1,921 video frames with different scenarios to create the dynamics in algorithmic adaptation and switching. For the test sequences, an average prediction accuracy of 97% is reached with sporadic excursions of the prediction error up to 20-30%. For the experiments, we have used a chip-multiprocessor system containing two quad-core processors [10]. In total, the system consists of eight processor cores of 2.33 GCycles/s, eight L1 caches (32 KB), four L2 caches (4 MB) and 4 GB of external memory.

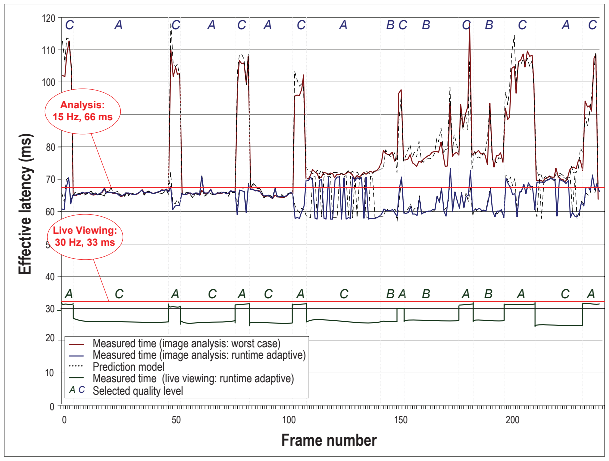

Fig. 4. Effective latency for worst-case vs. runtime adaptive control.

The resource manager can choose between three instances of live viewing, (a) low-quality, (b) medium-quality, or (c) high-quality viewing. For image analysis, a similar selection between different parallelization strategies can be made (See Fig. 3). The selection and reconfiguration of quality levels for both pipelines is done at an image level granularity. The mapping is fixed during the processing time of an image. Currently, the selection process is relatively coarse, and one processor core is dedicated for the operating system. The control can be easily made more scalable by adding intermediate quality levels. The actual selection is based on the resource demand for the analysis tasks, as estimated by the prediction model. As a case study, we process images (1024×1024 pixels, 30 Hz), where live viewing operates at the full frame rate, and image analysis at half the frame rate.

In Fig. 4, both the results of a worst-case mapping and a parallelized quality-controlled execution are shown. For the non-deterministic image analysis application, a worst-case execution (dark red curve, top) shows heavy excursions (85%) on the effective latency. The computation latency can vary between 60 and 120 ms, as the partitioning of tasks across processing cores is fixed. In Fig. 4(blue curve, middle), the results of our proposed runtime adaptive execution are shown as well. The variation on the latency is reduced significantly to only 20%. For the deterministic live-viewing application running in parallel, only the quality-controlled mapping is shown (green curve, bottom) where latency varies around 30 ms. This variation results from the switching of the resource manager in the parallelization degree of the non-deterministic analysis processing and at the same time switching live viewing between quality modes.

7. CONCLUSIONS

We have established a control system (according to Fig. 5) for combining the execution of deterministic and non-deterministic image-processing applications on a single platform.

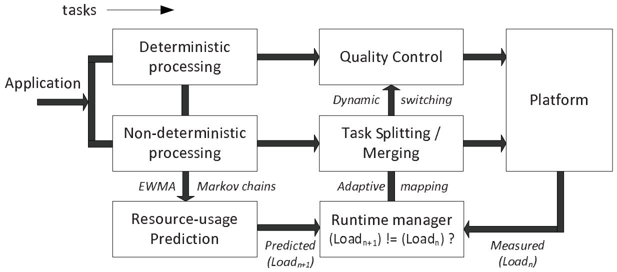

Fig. 5. Control system for parallel execution of heterogeneous imaging tasks.

The system is based on a modeling of the required computations of the processing tasks, by employing a combination of weighted moving-average filtering and scenario-based Markov chains to predict the execution. This research has been validated with a medical imaging case, executing two applications in parallel. We have shown that scenario-based Markov modeling is suited to describe the runtime resource usage of non-deterministic image analysis applications (97% accuracy), even if the flow graph dynamically switches between groups of tasks.

We have realized a runtime adaptive mapping of data and computations, such that the latency variation within image analysis is reduced by 70%. Furthermore, a live-viewing application is executed in parallel, where the runtime manager maintains constant throughput and latency by dynamically switching between quality modes. The techniques described in this paper can successfully be reused for alternative video applications using image analysis and stream-oriented tasks in parallel, like in surveillance systems.

8. REFERENCES

[1] C.C. Wüst *et al.*, “QoS control strategies for high-quality video processing,” *Real-Time Syst.*, vol. 30, no. 1-2, pp. 7–29, 2005.

[2] B. Li and K. Nahrstedt, “QualProbes: middleware QoS profiling services for configuring adaptive applications,” in *ACM Int. Conf. on Distr. systems platforms*, USA, 2000, pp. 256–272.

[3] P. Poplavko *et al.*, “Execution-time prediction for dynamic streaming applications with task-level parallelism,” in *Digital System Design Architectures*, 2007.

[4] M. Pastrnak and P.H.N. de With, “Data storage exploration and bandwidth analysis for distributed mpeg-4 decoding,” in *Cons. Elec., IEEE Int. Symp. on*, 2004.

[5] G.Z. Yang *et al.*, “Structure adaptive anisotropic image filtering,” *Image and Vision Computing*, vol. 14, no. 2, pp. 135–145, Mar. 1996.

[6] V. Auvray *et al.*, “Multiresolution parametric estimation of transparent motions,” *IEEE Int. Conf. on Image Proc.*, pp. I–141–4, Sept. 2005.

[7] P. Vuylsteke, “Image processing in computed radiography,” in *Proc. of the Int. Symp. on Computerized Tomography for Industrial Appl.*, Berlin, D, 1999, vol. 4, pp. 67–CD.

[8] V. Bismuth and R. Vaillant, “Elastic registration for stent enhancement in x-ray image sequences,” *IEEE Int. Conf. on Image Proc.*, pp. 2400–2403, Oct. 2008.

[9] V.S. Frost and B. Melamed, “Traffic modeling for telecommunications networks,” *Comm. Mag., IEEE*, vol. 32, no. 3, pp. 70–81, Mar 1994 .

[10] S. Radhakrishnan, S. Chinthamani, and Kai Cheng, “The blackford northbridge chipset for the intel 5000,” *Micro, IEEE*, vol. 27, no. 2, pp. 22–33, 2007.