Розпаралелювання алгоритмів відеопотоку в платформі Каскада

Адам Бржескі

Кафедра комп'ютерної архітектури, Гданський Університет Технології, ПОЛЬЩА, Гданськ, вул. Г. Нарутовіча 11\12,
E-mail: brzeski@eti.pg.gda.pl

Метою даної роботи є представлення різних технік розпаралелювання алгоритмів відеопотоку, надані платформою Каскада, що є новою системою роботи у суперкомп'ютерному середовищі, розробленому для обробки потоків мультимедія. Система сприяє як побудові алгоритмів так і забезпечує комунікаційний механізм для пишереної обробки даних, що дає змогу мати переваги з обчислювальної потужності суперкомп'ютеру.

Розглянуті методи розпаралелювання включають рамний рівень співпадіння, який досягається за рахунок рівночасної обробки послідовних рамок, багатозвязкових технік та обробки інформаційних даних. Для основного режима обробки даних було досліджено автономний аналіз збережених відеоматеріалів та обробки в мережі та потоків в реальному часі. Ефективність виконання в обох режимах було виміряно на чотирьох рудомістких алгоритмах розпізнання зображення, розроблена для підтримки медичного ендоскопічного обстеження. Підвищення виконання алгоритмів було дуже пріоритетною задачею, в зв'язку з тим, що повільне виконання перешкоджає успішному застосуванню в реальних медичних системах.

Для кожного метода розпаралелювання були проведені широкі часові тести. За допомогою запровадження алгоритмів розпаралелювання на різних ендоскопічних відеопослідовностей, були виміряні параметри, що впливають на можливості обробки даних автономно та в мережі, а саме рівень боробки даних та внесені затримки. Досягнуті результати підтверджують високі можливості платформи Каскада для реалізації паралельних алгоритмів. Ефективність виконання розглянутих медичних алгоритмів зросла до достатнього рівня у практичному застосуванні. Як показали часові вимірювання, рамковий рівень розпаралелювання був визначений як ефектине рішення для автономного аналізу, в той час коли конвеєрна обробка даних, що об'єднана з багато поточною обробкою, забезпечили високе виконання обробки даних в мережі.

Переклад зроблено Горьковою Н.Г., центр іноземних мов «Universal Talk», www.utalk.com.ua

Parallelization of video stream algorithms in Kaskada platform

Adam Brzeski

Departament of Computer Architecture, Gdańsk University of Technology, POLAND, Gdańsk, G. Narutowicza street 11/12,
E-mail: brzeski@eti.pg.gda.pl

The purpose of this work is to present different techniques of video stream algorithms parallelization provided by the Kaskada platform - a novel system working in a supercomputer environment designated for multimedia streams processing. Considered parallelization methods include frame-level concurrency, multithreading and pipeline processing. Execution performance was measured on four time-consuming image recognition algorithms, designed to support medical endoscopic examinations. The achieved results confirm high capabilities of Kaskada platform for executing parallel algorithms. Frame-level parallelization was proved to be a great solution for offline processing, while pipeline processing combined with multithreading provided high performance for online, real-time analysis

Keywords – parrallel processing, pipeline processing, multithreading, high-performance computing, endoscopy

I. Introduction

At the time of rapid development of high power computers, performing computation at low level, and many arising challenges associated with more abstract computer vision tasks, such as analysing videos from surveillence cameras or analysis of medical images, there appears to be a need for a solution effectively connecting the two areas, enabling succesful construction and execution of stream processing algorithms in the environment of a supercomputer. Kaskada platform[1], developed within Mayday 2012 project[2], is a novel approach in this field. Kaskada is a universal runtime platform for algorithms processing multimedia streams, e.g. videos and sound recordings. The platform operates in Galera cluster system, using its enormous computing power and making it available for executed algorithms. It is a perfect solution for algorithms presenting high demand on computational power, examples of which are image recognition algorithms supporting medical endoscopic examinations of gastrointestinal tract.

II. Kaskada platform

Except being a powerful execution environment for time-consuming algorithms, Kaskada also provides a universal external interface in the form of automatically created webservices, enabling launching algorithms from remote applications, e.g. from doctor's office. Kaskada is also a framework facilitating the construction of stream algorithms. Platform performs all video decoding tasks, passing raw frames to the algorithm. Also, extensive communication mechanisms are provided by the platform, enabling construction of highly parallelized, distributed algorithms in the form of computational *services* engaging multiple processors.

It must be noted that the Kaskada platform supports two classes of algorithms: stream algorithms, which are built as a sequence of *computational tasks*, processing the input data in a pipeline model, and master-slave algorithms. This paper is entirely concentrated on streaming algorithms and methods of their parallelization.

III. Parallelized algorithms

For the purpose evaluating parallelization capabilities of Kaskada platform, implementions of endoscopy-supporting image recognition algorithms were used. Short description of each algorithm is presented below. A common feature of each algorithm is utilisation of artificial intelligence based classifiers.

- **Kodogiannis1** algorithm detecting disease tissues proposed by Kodogiannis et al, applying statistical features and unique NTU transformation[3],
- **Magoulas1** disease detection algorithm developed by Magoulas et al[4] basing on statistical features of images,
- **Magoulas2** extension of the previous algorithm, expanded with 2D-DWT transformations[5],
- **BaopuLi1** cancer detection algorithm proposed by Li and Meng. Incorporates multiple 2D-DWT and LBP transformations[6].

IV. Motivation for performance enhancement

Image recognition algorithms supporting gastrointestinal endoscopy examinations can be employed for both online and offline video analysis. In online mode videos must be processed in real-time, so algorithm's *processing rate* must be greater than video framerate. Moreover, it is required that the *delay* introduced by the algorithm is limited to about 0.2s. In offline analysis of stored videos the *delay* has no significance, but *processing rate* is expected to be considerably high in order to process long movies in short time.

However, computational complexity of algorithms described earlier is high. Time required for processing a typical frame of size 720x576 pixels measured on a single Galera's processor for the algorithm Kodogiannis1 is 0.4s, while for the other algorithms the time is above 1s. At the same time, a typical video stream has a framerate of 25 frames per second. Also, the *delay* implicated from single frame processing time exceeds the acceptable level. Hence, a significant increase in *processing rate* as well as shortening the *delay* are highly desired.

V. Parallelization methods and results

Several ways of parallelizing stream algorithms are enabled by Kaskada platform. The parallelization techniques described later in the article were applied on presented algorithms and. For each case, required parameters were measured: the *processing rate*, represented by the average number of frames processed per second, and the *delay*, being a single frame processing time. Tests were performed on 5 video sequences of 720x576 resolution, twice for each sequence, giving a total of 10 measurements for each case. Averaged results were presented on charts. Standard deviations of each sample were marked in the form of vertical bars at the average values.

A. Concurrent frame processing

Kaskada platform enables distributing consecutive frames of a video stream to different processors. In this way, multiple frames can processed concurrently, which can be denoted as a frame-level parallelization. Figure 1 presents a sample *service* implementing this case. A single node is desginated for distributing the frames among *computational tasks* organized in a layer, performing as separate instances of the algorithm. The last node in this scenario gathers computed results a generates the output of the service.

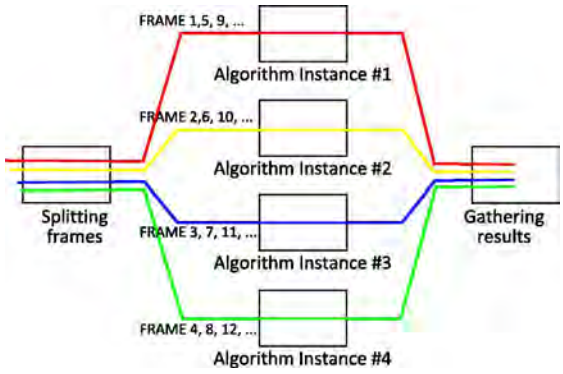

Fig.1 Distributing a frame sequence to 4 computational tasks

The advantages of this technique are high versatility and simplicity of implementation, since the mechanism is independent of the parallelized algorithm, provided that dependencies between frames are not considered. Moreover, this solution allows to effectively utlise the processors power and significantly reduce the overall processing time. Extension to any number of processors is possible providing high scalability. Unfortunately, the mechanism in no way reduces the processing time of a single frame, so that the *delay* remains unchanged comparing to sequential processing. The following charts present results achieved accordingly to the number of processors in the processing layer.

Eeach of the algorithm achieved stable *processing rate* growth. As expected, the *delay* remained unchanged, yielding only slight fluctuations.

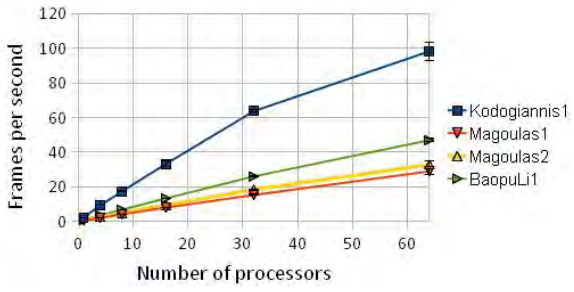

Fig.2 *Processing rate* of frame-level parallelized algorithms

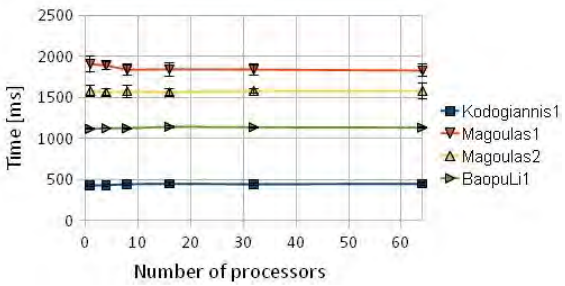

Fig.3 Delay of frame-level parallelized algorithms

B. Multithreading

Each *computational task* in the Kaskada platform is actually a process that can be executed using multiple threads. Since Galera's nodes are 8-core systems, it is a reasonable choice to split the execution into 8 threads, which could potentially result in 8 times speedup in ideal case. In practice, however, achieved speedup is usually much lower due to memory access conflicts and data synchronization between threads. The advantage of this solution is a possibility to shorten the single frame processing time, at the same time reducing the *delay*.

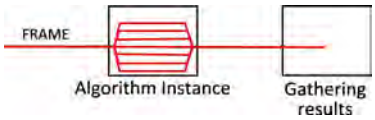

Fig.4 Conception of multithreaded frame processing

Multithreading was implemented using OpenMP mechanism[7]. To accomplish this it was required to identify time-consuming loops in the algorithms, which should be parallelized. Therefore, execution time measurements of particular stages of algorithm were carried out, which indicated code regions to be parallized.

The following charts present achieved results accordingly to the number of processor used for multithreading.

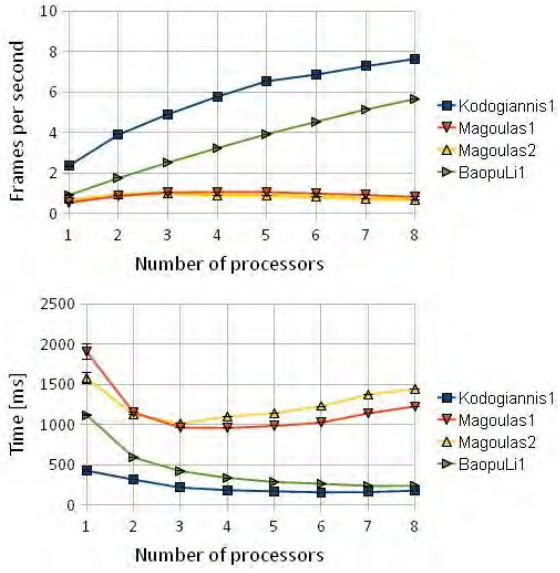

Fig.5 Processing rate (top) and delay (bottom) of algorithms parallelized using multithreading

As expected, the method results in much lower speedups than the previous one. While algorithms BaopuLi1and Kodogiannis1 gained satisfactory speedup with high efficiency, including shortenening the single frame time, algorithms Magoulas1 and Magoulas2 did not show significant performance improvement. The reason for this fact is the low capability of these algorithms for parallelization implicated from large data dependencies. The method therefore enables slight increase of *processing rate* and reduction of introduced *delay*, but in the case of less complex algorithms this technique may be sufficient. It can be also succesfully pull together with other methods like the previous concurrent frame processing or pipeline processing.

C. Pipeline processing with multithreading

The most interesting parallelization technique offered by the Kaskada platform is algorithm-level pipeline processing. The algorithms are divided into functional blocks to be executed by separate *computational tasks* in a form of a pipeline. Independent blocks can be put in a layer for concurrent execution. This allows to construct a service arranged adequately to a logic scheme of the algorithm. Therefore, each of the block can be distributed to different Galera's node and executed using multithreading, which enables to utlise high number of processors. Exemplary service implementing such scenario for the algorithm BaopuLi1 is shown in Figure 6.

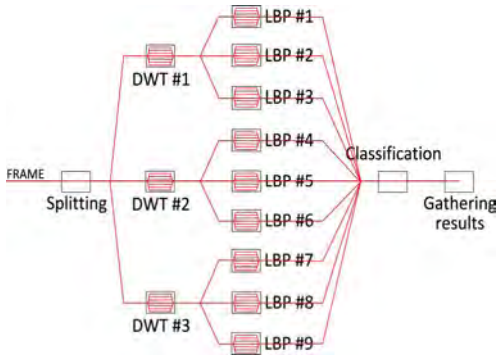

Fig.6 Conception of pipeline processing combined with multithreading for BaopuLi1 algorithm

Pipeline processing assures increase of *processing rate*, while concurrent execution of separate blocks, as well as multithreading, shortens the *delay*.

The charts show performance of algorithms measured for 8 variants utilising from 1 to 8 processors in each multithreaded node. Since the arrangement of the service is different for each of the algorithm, also the ranges of possibly used numbers of processors differ between them. Algorithm Magoulas1 was excluded from the test, since it's structure prevent efficient pipeline implementation.

BaopuLi1 algorithm showed relatively stable performance growth with the increasing number of processors. For 86 processors the *processing rate* exceeded 50, while the *delay* dropped below 0.1s. Kodogiannis1 algorithm achieved best performance for 32 processors. Marginal performance gain was achieved for the algorithm Magoulas2.

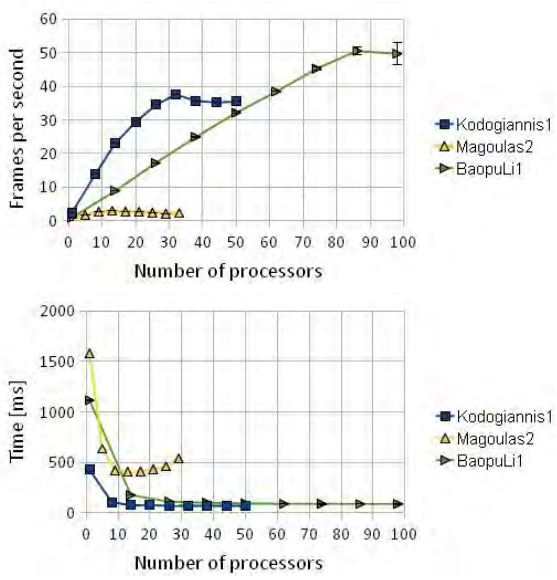

Fig.7 Processing rate (top) and delay (bottom) of algorithms parallelized using pipeline model with multithreading

Presented pipeline processing method therefore requires some sort of capabilty from the parallelized algorithm. In return, very good performance can be achieved for well divisible algorithms.

Conclusion

Performance achieved of the algorithms after parallelization in Kaskada platform can be considered satisfactory. For the purpose of offline processing, parallelization on the frame level showed best results, offering high *processing rate* along with high speedup efficiency. For certain algorithms, only a slightly lower efficiency was retained by the pipeline processing method combined with multithreading, which still remained high *processing rate*. In addition, the method significantly reduced a single frame processing time, and thus the introduced *delay*. Concluding, parallelization capabilities of Kaskada platform enabled considerable performance gain for the investigated algorithms. The presented medical recognition algorithms suffered from high computational complexity, resulting in long execution time. Utilising computational power of Galera supercomputer, Kaskada platform accelerated all the algorithms to perform fairly well in offline processing mode, providing efficient speedup with increasing number of processors. For suffiiciently divisible algorithms, also online processing became possible by utilising pipeline processing supported by multithreading.

Acknowledgements

Research funded within the project No. POIG.02.03.03-00-008/08, entitled ”MAYDAY EURO 2012- the supercomputer platform of context-depended analysis of multimedia data streams for identifying specified objects or safety threads”. The project is subsidized by the European regional development fund and by the Polish State budget”.

References

[1] J. Proficz, H. Krawczyk, “Kaskada – multimedia processing platform architecture”, 2010.

[2] M. Życzkowski, Mayday 2012 project site, http://mayday2012.gda.pl, [Online][Cited: 09.2011]

[3] V.S. Kodogiannis and M. Boulougoura, “An adaptive neurofuzzy approach for the diagnosis in wireless capsule endoscopy imaging”, International Journal of Information Technology, 13(1), 2007.

[4] G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis, “Neural network-based colonoscopic diagnosis using on-line learning and differential evolution”, Applied Soft Computing, 2004.

[5] G.D. Magoulas, “Neuronal networks and textural descriptors for automated tissue classification in endoscopy”, Oncology Reports, 15, 2006.

[6] B. Li and M. Meng, “Small bowel tumor detection for wireless capsule endoscopy images using textural features and support vector machine”, IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[7] Blaise Barney, ” OpenMP Tutorial” [Online] [Cited: 09.2011], http://computing.llnl.gov/tutorials/openM