Parallel Numerics '05, 129-139 M. Vajteršic, R. Trobec, P. Zinterhof, A. Uhl (Eds.)
Chapter 6: Multimedia ISBN 961-6303-67-8

Limitations of Cluster Computing in a Communication Intensive Multimedia Application

Florian Tischler, Andreas Uhl *

Salzburg University, Department of Scientific Computing
Jakob Haringer-Str. 2, A-5020 Salzburg, Austria
http://www.scicomp.sbg.ac.at/

Block-Matching motion compensation is one of the key technologies in current video coding standards and requires about 70% of the overall execution time. Therefore, parallelization of video coding always involves parallelization of block-matching motion compensation. In this work we use this communication intensive application to show the limitations of cluster computing as compared to shared memory architectures.

1 Introduction

The widespread use of digital video in various environments has caused a high demand for efficient compression techniques. Unfortunately, many compression algorithms have prohibitive execution times using a single serial microprocessor [1], which leads to the use of high performance computing systems for such tasks. Software based approaches are becoming more popular in this area because of the rapid evolution of multimedia techniques which has dramatically shortened the time available to come up with a new hardware design for each improved standard. In this context, several papers have been published describing real-time video coding on general purpose parallel architectures – see for example MPEG-1,2,4 [2, 3, 4], H.261/3/L [5, 6, 7], and wavelet-based video compression [8, 9, 10].

Block-matching motion compensation is the most demanding part of current video coding standards and requires 60 – 80 % of the total computations involved. A significant amount of work discusses dedicated hardware for

*Corresponding author. E-mail: uhl@cosy.sbg.ac.at

block-matching (e.g. [11]), some software based block-matching approaches have also been investigated [12]. However, in most cases, only one approach (in particular one parallelization granularity) is discussed in detail and it is left to the reader to compare the results to other research in this area. In recent work [13], we have systematically compared different levels of parallelization granularity from the scalability point of view, we have investigated special communication patterns for these algorithms [14], and we have also proposed to switch among granularities in a dynamic way [15]. We have as well compared these different granularity levels in the context of a complete wavelet packet based parallel video codec [16].

In this work, parallel block-matching algorithms with different granularities are used as example application with high communication demand in order to assess the effectiveness of cluster computing for this scenario. In section 2 we shortly review block-matching motion compensation in the video coding context. Section 3 discusses three granularity levels for parallel block-matching which are experimentally evaluated on three architectures using MPI based message passing. In section 4 we review the concept of switching granularities and provide experimental results to show the performance problems of the cluster implementation.

2 Block-Matching Motion Compensation in Video Coding

The main idea of motion compensated video coding is to use the temporal and spatial correlation between frames in a video sequence (Fig. 1) for predicting the current frame from previously (de)coded ones. Since this prediction fails in some regions (e.g., due to occlusion), the residual between this prediction and the current frame being processed is computed and additionally stored after lossy compression.

Because of its simplicity and effectiveness block-matching algorithms are widely used to remove temporal correlation [17]. In block-matching motion compensation, the scene (i.e. video frame) is classically divided into non-overlapping "block" regions. For estimating the motion, each block in the current frame is compared against the blocks in the search area in the reference frame (i.e. previously encoded and subsequently decoded frame) and the motion vector (d_1, d_2) corresponding to the best match is returned (see Fig. 2). The "best" match of the blocks is identified to be that match giving the

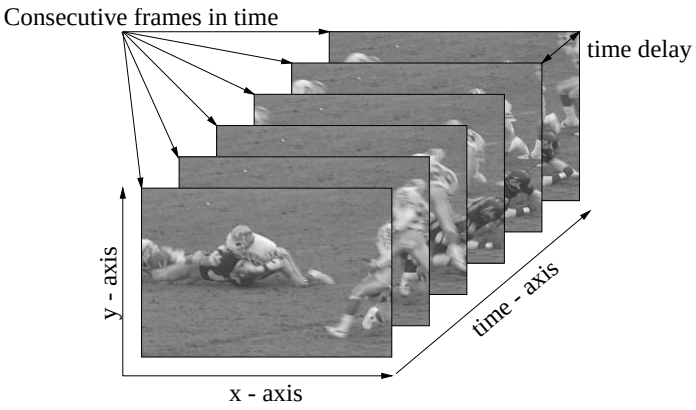


Figure 1: Frame-structure of video (football sequence)

minimum mean square error (MSE) of all blocks in search area defined as

$$MSE(d_1, d_2) = \frac{1}{N_1 N_2} \sum_{(n_1,n_2)\in\mathcal{B}} [s_k(n_1, n_2) - \hat{s}_{k-l}(n_1 + d_1, n_2 + d_2)]^2$$

where $\mathcal{B}$ denotes a $N_1 * N_2$ block for a set of candidate motion vectors (d_1, d_2), s is the current frame and $\hat{s}$ the reference frame. Note that that (d_1, d_2) classically is restricted to integer pixel locations, however, it has turned out that an increase in accuracy to sub-pixel values (e.g. quater pixel) is able to increase prediction quality considerably.

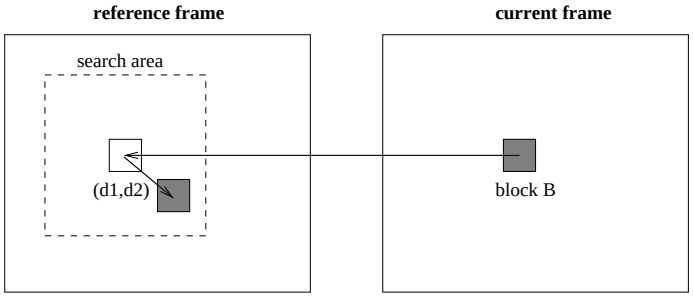


Figure 2: Block-matching motion estimation

The algorithm which visits all blocks in the search area to compute the minimum is called full search (FS). In order to speed up the search process, many techniques have been proposed to reduce the number of candidate blocks. The main idea is to introduce a specific search pattern which is recursively applied at the position of the minimal local error. The most popular algorithm

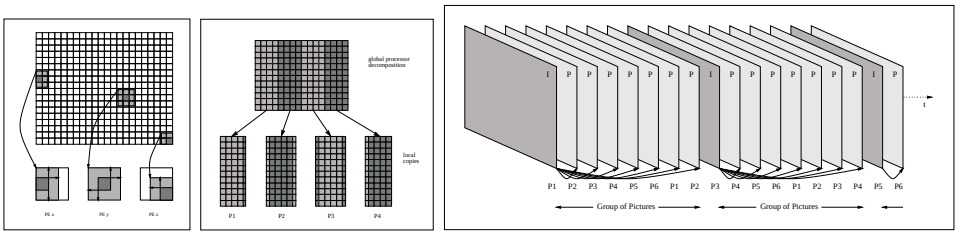
of this type is called "Three Step Search" (TSS) which reduces the computational amount significantly at the cost of a suboptimal solution (and therefore a residual with slightly more energy). In particular, a search pattern is applied in three recursions at the position of the lowest local error [17].

3 Granularity of Parallel Block-matching Algorithms

In general, granularity determines important characteristics of a parallelization approach. A coarse grained parallelization usually requires little communication, on the other hand, balanced load may be hard to achieve. Independent of the results corresponding to parallelization efficiency, granularity has also major impact with respect to hardware requirements and coding delay of a video compression system.

Examing the sequence of frames in the video stream to be encoded, we identify two main granularity levels: intra-frame granularity (fine grained parallelization) and inter-frame granularity (coarse grained parallelization).

As described in the previous section, the current frame and the reference frame are segmented into equal sized blocks. Based on this partitioning, two intra-frame granularities may be used (fine grained parallelization). In the *Blockbased* parallelization (BB) approach, the blocks of the current frame are distributed among the processing elements (PE), the computation of the motion vector for each block is done locally. To avoid interprocess communication, the entire search area surrounding the corresponding block in the reference frame (see Fig. 3.a) is sent to the PE in charge of the current block resulting in an overlapping data partition.



(a) BB (b) IF (c) GOP

Figure 3: Partitioning schemes with different granularities

Another way of partitioning the current frame among the PE is the stripe subimage method (see [12] for a comparison of several flavours of this method) which we denote *Intra Frame* parallelization (IF). Using this method, the

current frame is split into horizontal or vertical stripes as shown in Fig. 3.b and the computations associated with the blocks contained in one contiguous stripe are assigned to one PE. Again, a search-overlap is used to avoid interprocess communication among the PEs.

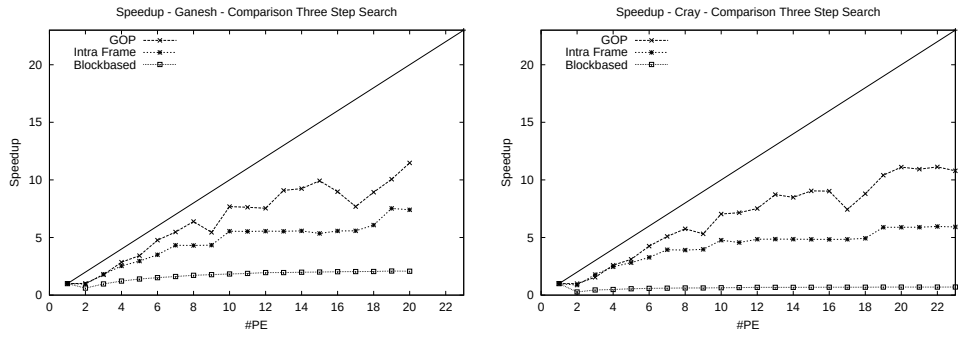
To preserve the quality reached by the sequential version, the smallest allowed unit for splitting the frame is limited by the blocksize. These units are combined according to the number of PEs to form stripes as uniformly sized as possible. Therefore, this method should perform well especially in the case where the units can be distributed evenly among the compute nodes. Additionally, two types of inter-frame granularity (coarse grained parallelization) may be considered. A group of pictures (GOP) is a collection of P-frames (predicted frames) which are all processed in relation to one reference frame, the I-frame. An obvious way to distribute the block-matching computations is to assign entire frames to single PEs. Fig. 3.c illustrates this technique which is denoted GOP parallelization. A second possibility is to assign complete GOPs to single PEs (denoted eGOP).

It is obvious that the lower communication demand (in terms of the number of messsages to be exchanged) of inter-frame granularity has to be paid with higher memory consumption and a possibly higher coding delay [13]. Consequently, coarse grained parallelization (GOP and eGOP) will not be suited for real-time and on-line applications like video conferencing, since especially the high coding delay is not acceptable for these types of applications. Additionally, the high memory requirements lead to high costs (especially for hardware solutions) and poor cache performance. Note that the eGOP approach is not investigated further since coding delay and memory requirements are too high for practical systems. Moreover, load imbalance is significant for short videos using this technique.

3.1 Experiments

All simulations use the standard test sequence "Football" in QCIF (quarter common intermediate format), having a size of 176×144 pixels. The blocksize is set to 8x8, motion vectors may point 7 pixels in each direction, the computations are performed on 40 successive frames of the sequence. Unless denoted otherwise, three step search (TSS) is employed for computing motion vectors. Three different architectures using native MPI versions are employed: a SGI Power Challenge (20 MIPS R10000 processors and 2.5 GB memory), a Cray T3E (DEC Alpha EV5 processors, 128 MB memory each, interconnected by a 3D torus with 3 GB/s bandwidth in each network node), and a Siemens HPCline cluster (compute nodes with 2 Pentium III @ 850 MHz and 512 MB memory each, interconnected by a 2D torus with 500 MB/s bandwidth in

each network node). Note that the three systems have opposite properties. Whereas the SGI has the slowest compute nodes and the cluster the fastest, the communication system is fastest on the SGI (shared memory) and slowest on the cluster (see above). Therefore, we may expect the best result with respect to scalability on the SGI and the worst results on the cluster, the Cray is expected to perform in-between. SPMD-style programming with non-blocking MPI communication commands is used.

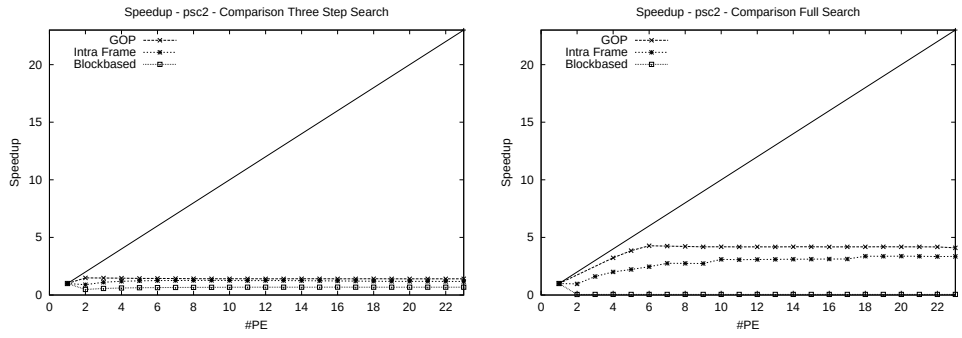


(a) SGI (b) Cray

Figure 4: Speedup results of parallel block-matching (TSS).

Figure 4 shows that block-based parallelization does not show satisfactory speedup, neither on the SGI nor on the Cray. The communication demand is too high to deliver reasonable results, even though the load may be balanced perfectly. Note that this contrasts to the results if FS is employed as motion vector search algorithm [13], where at least on the SGI reasonalble results are obtained. Intra Frame parallelization performs better but exhibits several plateaus in the speedup plots. These plateaus are due to the unequal distribution of the limited number of stripes in each frame among the PEs. See [13] for detailed explanations of this phenomenon. The overall performance of GOP parallelization is the best compared to the other two approaches. Also, GOP parallelization leads to plateaus in the speedup plots. The reason for these plateaus is that a fixed number of frames can not be distributed evenly among an arbitrary number of PEs, therefore several PEs may remain idle when the last frames are processed. Note, that these plateaus disappear for long video sequences since unbalanced load in the last scheduling round is not important for a large number of scheduling rounds. This is not the case for Intra Frame parallelization since these plateaus appear on a per frame basis and can only be avoided by skipping synchronization at the frame level, which is difficult

from the application point of view and reduces the corresponding advantages of this approach. When comparing the two architectures, the performance matches exactly the prediction. Reasonable performance is achieved on the Cray and the SGI using GOP and Intra Frame parallelization, the results on the SGI are slightly better due to the faster "communication" (exploiting the shared memory).



(a) TSS (b) FS

Figure 5: Speedup results of parallel block-matching on the cluster.

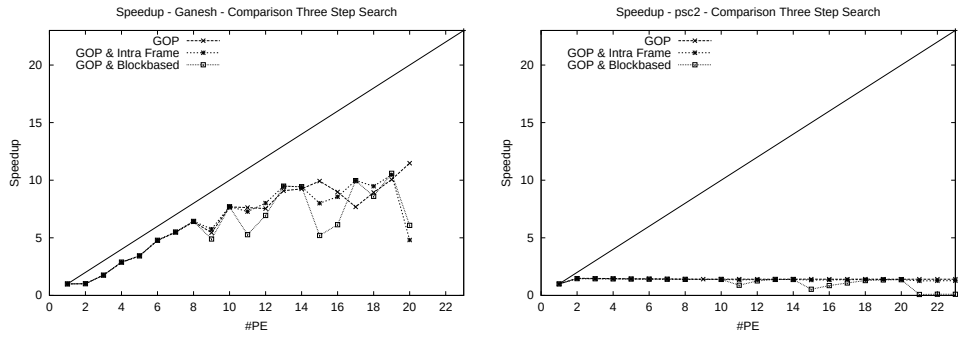
Turning to the Cluster implementation, the results differ dramatically (Fig. 5). None of the three granularity levels is able to produce any reasonable speedup. When considering block-matching with the full search (FS) algorithm for motion vector computation the results are slightly better but far from being satisfactory (see Fig. 5.b). As a matter of fact, the Cluster architecture turns out to be of no use for this communication intensive algorithm.

4 Hybrid Granularity in Parallel Block-Matching

Due to its good efficiency, the GOP parallelization approach is an interesting candidate for video compression applications and other applications where large quantities of still image data need to be compressed. As we have noted, due to its high coding delay and memory demand it is not suited for on-line applications. However, for off-line applications where large amounts of data need to be processed in reasonable time, coding delay and memory demand are not critical issues. For example, we mention large surveillance systems and all types of storage applications in general. In an environment where long videos are processed (e.g. for insertion into a video-on-demand server)

the load-imbalance phenomenon as discussed in the previous section does not pose any problem. However, in case of many short video sequences to be compressed this behaviour degrades execution efficiency significantly. Coming back to a surveillance application as an example, this would be the case if the video cameras only record if triggered by a sensor measurement, e.g. an infrared sensor. Here, many short videos are delivered to the storage system and need to be compressed as fast as possible. The aim of this section is to introduce a methodology to improve the efficiency of the GOP parallelization approach in such an environment.

Since the load-balancing problems are intrinsic to the GOP approach and both other approaches show limited scalability, we propose to extend the GOP approach in a way where we can keep its good efficiency and incorporate the good load-balancing facilities of the other two schemes (see [15] for more details and comprehensive results).



(a) SGI (b) Cluster

Figure 6: Speedup results of the hybrid granularity modes (TSS).

In particular we suggest to start the computations according to the GOP parallelization and switch to a finer granularity as soon as the efficiency of the first scheme degrades. Note that switching granularity implies a data redistribution procedure which is costly in terms of communication and synchronization demand. Of course, the location of the granularity switch needs to be determined in advance (i.e. to avoid a performance degradation) and should not react to an already poor behaviour. The GOP approach is restricted to a number of frames which is an integer multiple of the number of PEs. Frames exceeding this set are processed according to an approach with finer granularity (BB or IF) and the data are distributed accordingly.

Fig. 6 shows the speedup results of this approach on the SGI and the

Cluster, respectively. Contrasting to the results in [15] for FS, even on the SGI the hybrid granularity algorithms may improve the GOP results only for few selected parameters (i.e. GOP is improved by GOP+IF with 12, 17, and 18 PEs), in most cases the results are worse. Again we face dramatically poor results for the Cluster where almost no speedup is produced across the entire range of PEs (see Fig. 6.b). Finally when considering again the results for FS on the cluster (Fig. 7), we notice some speedup but both types of hybrid algorithms decrease the performance of the GOP algorithm due to the poor performance of the intra frame granularities at the end of the computations.

5 Conclusion

We have selected a communication intensive multimedia application for showing the limitations of cluster computing employing the message passing paradigm using MPI – parallel block matching motion compensation of QCIF video sequences employing TSS as well as FS as motion vector search algorithms can not be performed in a reasonable way on a cluster whereas acceptable results with respect to scalability are obtained on shared memory and dedicated HPC architectures with high bandwidth interconnections.

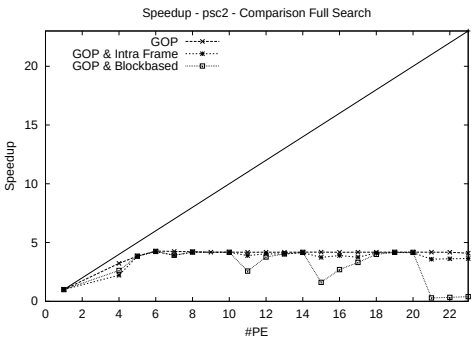


Figure 7: Speedup results of the hybrid granularity modes on the Cluster (FS).

Acknowledgements

The authors have been partially supported by the Austrian Science Fund FWF, project no. P13903.

References

[1] K. Shen, G.W. Cook, L.H. Jamieson, and E.J. Delp. An overview of parallel processing approaches to image and video compression. In M. Rabbani, editor, *Image and Video Compression*, volume 2186 of *SPIE Proceedings*, pages 197–208, 1994.

[2] S.M. Akramullah, I. Ahmad, and M.L. Liou. Performance of software-based MPEG-2 video encoder on parallel and distributed systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):687–695, 1997.

[3] Y. He, I. Ahmad, and M.L. Liou. Modeling and scheduling for MPEG-4 based video encoder using a cluster of workstations. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Parallel Computation. Proceedings of ACPC'99*, volume 1557 of *Lecture Notes on Computer Science*, pages 306–316. Springer-Verlag, 1999.

[4] K. Shen, L.A. Rowe, and E.J. Delp. A parallel implementation of an MPEG1 encoder: faster than real-time ! In A.A. Rodriguez, R.J. Safranek, and E.J. Delp, editors, *Digital Video Compression: Algorithms and Technologies*, volume 2419 of *SPIE Proceedings*, pages 407–418, 1995.

[5] J. C. Fernandez and M. P. Malumbres. A parallel implemenatation of H.26L video encoder. In B. Monien and R. Feldmann, editors, *Parallel Processing. Proceedings of EuroPar'02*, volume 2400 of *Lecture Notes on Computer Science*, pages 830–833. Springer-Verlag, 2002.

[6] K.K. Leung, N.H.C. Yung, and P.Y.S. Cheung. Parallelization methodology for video coding – an implementation on the TMS320C80. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(10):1413–1423, 2000.

[7] N.H.C. Yung and K.K. Leung. Parallelization of the H.261 video coding algorithm on the IBM SP2 multiprocessor system. In *Proceedings of the IEEE International Conference on Algorithms, Architectures, and Applications for Parallel Processing*, pages 571–578. North Holland, 1997.

[8] M. Feil and A. Uhl. Efficient wavelet-based video coding. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium IPDPS'02 (Abstracts and CD-ROM), PDIVM 2002*, page 128. IEEE Computer Society Press, 2002.

[9] M. Feil and A. Uhl. Motion-compensated wavelet packet zerotree video coding on multicomputers. *Journal of Systems Architecture*, 49:75–87, 2003.

[10] R. Kutil. Approaches to zerotree image and video coding on MIMD architectures. *Parallel Computing*, 28(7–8):1095–1109, August 2002.

[11] S.-C. Cheng and H.-M. Hang. A comparison of block-matching algorithms mapped to systolic-array implementation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(5):741–757, October 1997.

[12] M. Tan, J. M. Siegel, and H. J. Siegel. Parallel implementation of block-based motion vector estimation for video compression on four parallel processing systems. *Internaltional Journal of Parallel Programming*, 27(3):195–225, 1999.

[13] F. Tischler and A. Uhl. Granularity levels in parallel block-matching motion compensation. In D. Kranzlmüller, P. Kacsuk, J. Dongarra, and J. Volkert, editors, *Recent advances in Parallel Virtual Machine and Message Passing Interface (EuroPVM/MPI) - 9th European PVM/MPI Users Group Meeting*, volume 2474 of *Lecture Notes on Computer Science*, pages 183 – 190. Springer-Verlag, September 2002.

[14] F. Tischler and A. Uhl. Communication patterns in MPI-based parallel block-matching. In R. Trobec, P. Zinterhof, M. Vajteršic, and A. Uhl, editors, *Parallel Numerics '02 – Theory and Applications (Proceedings of the International Workshop)*, pages 211–220, Bled, Slovenia, October 2002.

[15] F. Tischler and A. Uhl. Dynamic granularity switching in parallel block-matching motion compensation. In M. Danelutto, D. Laforenza, and M. Vanneschi, editors, *Parallel Processing. Proceedings of EuroPar'04*, volume 3149 of *Lecture Notes on Computer Science*, pages 768–775. Springer-Verlag, September 2004.

[16] M. Feil and A. Uhl. ParWave: Granularity in parallel wavelet packet video coding. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering 2002*, pages 479–490, Stuttgart, Germany, 2002. Springer-Verlag.

[17] B. Furht, J. Greenberg, and R. Westwater. *Motion estimation algorithms for video compression*. Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands, 1997.