Parallel Overlapped Block-Matching Motion Compensation Using MPI and OpenMP

Elias Pschernig and Andreas Uhl
Salzburg University, Department of Scientific Computing
Jakob Haringer-Str. 2, A-5020 Salzburg, Austria

Digital video compression is the computationally most demanding algorithm in the area of multimedia signal processing. Block-matching motion compensation covers about 60-80% of the runtime of all standardized video coding schemes. Overlapped block-matching motion compensation (OBM) enhances the prediction results of classical non-overlapped block-matching at a high additional computational cost.

In this context, we investigate two OBM algorithms, the Raster Scan Algorithm (RAST) and an iterative algorithm, based on the Iterative Conditional Mode (ICM). We experimentally show that both approaches improve the prediction quality as compared to classical block-matching significantly.

Both algorithms were implemented using MPI and OpenMP and executed on SGI and HITACHI shared memory architectures. There exist several different granularity levels for parallel BM. In our context, we use intra-frame parallelization (distributing single blocks to the PEs) since this is the approach most suited for applications on a majority of hardware systems. This block based parallelism (BBP) works very well in the case of ICM, whereas for RAST this approach is not straightforward. The error for a block depends on the MVs of neighbour blocks, which are expected to be available in the sequential raster-scan order. This means, that simply working on blocks in parallel won't work. As a consequence, only few of the available PEs can be used in parallel for the majority of blocks. The maximum number of simultaneously processed blocks is half the number of columns of the frame.

For the experiments, we use the test-sequence "Mobile" with 720×576 pixels. Note that for MPI implementation each communication event needs to be explicitely stated whereas OpenMP facilitates parallleization by simple loop distribution using comiler directives (e.g. `#pragma parallel pfor`). As a consequence, the implementation effort is significantly higher in the MPI case. On the other hand, OpenMP is restricted to multiprocessors whereas MPI may be used on almost any high performance computing architecture.

Whereas for classical block-matching the advantage of the OpenMP implementation over MPI is clearly exhibited, the situation entirely changes for ICM. Due to the higher amount of computations required for each block, the computation/communication ratio changes and facilitates very good efficiency for both approaches on an almost identical level. RAST on the other hand again requires more communication and especially synchronization effort as explained before. For these reasons, again the OpenMP implementation outperforms the MPI message passing approach. However, the execution efficiency is on a satisfying level for both approaches since the scalability constraints of the RAST parallelization are valid for a higher number of PEs only in case of a large video sequence like Mobile.

In contrast to classical block-matching we achieved satisfying efficiency for the MPI implementation as well as for the OpenMP implementation for all two types of OBM. As a consequence, the decision which programming model has to be chosen does not depend on the required efficiency but mainly on the nature of the target environment. On multiprocessors, OpenMP will be chosen due to its low implementation costs. In case of heterogenous target hardware platforms (e.g. multiprocessors, multicomputers, clusters) MPI will be chosen.