Scalability of Macroblock-level Parallelism for H.264 Decoding

Mauricio Alvarez Mesa[*,1], Alex Ramirez[†], Eduard Ayguade[†], Xavier Martorell[†], Mateo Valero[†,2]

[*] *Universitat Politecnica de Catalunya, Barcelona, Spain*
[†] *Barcelona Supercomputing Center, Barcelona, Spain*

ABSTRACT

This paper presents a study of the scalability of a macroblock-level parallelization of the H.264 decoder for High Definition applications. We have implemented this parallelization on a shared memory multiprocessor. Our implementation achieves a maximum of 9.5X speed-up using 24 processors. The limited scalability of this implementation comes from the overhead introduced by a centralized task queue. Our main conclusion is that macroblock-level parallelism is a fine-grain form of Thread-Level Parallelism that requires a fast and scalable work distribution and synchronization mechanisms in order to be efficient.

KEYWORDS: video codec, multicore, CMP, SMP, parallelization, h.264

1 Introduction

There is a paradigm shift in computer architecture towards chip multiprocessors (CMPs). These CMPs offer a huge amount of potential performance that can only be exploited efficiently if applications can be effectively parallelized. The scalability of performance on those architectures depends on the possibility of finding enough Thread Level Parallelism (TLP) in applications.

One application field that is very important in several computing domains is video processing. The trends towards High Definition (HD) applications and the use of advanced video codecs like H.264 have resulted in a significant increase in the computational requirements of video coding. An important question is whether video coding applications can benefit from the performance offered by CMP architectures. This translate into the question

[1]E-mail: {alvarez,xavim}@ac.upc.edu - {eduard.ayguade, alex.ramirez, mateo.valero}@bsc.es
[2]This work has been supported by the European Commission in the context of the SARC project (contract no. 27648), and the Spanish Ministry of Education (contract no. TIN2007-60625)

of how much TLP can be extracted from these applications and how effectively this TLP can be exploited in such architectures.

In this paper we evaluate the parallel scalability of the H.264 decoder by exploiting macroblock-level parallelism and measuring the resulting performance on an Shared Memory Multiprocessor (SMP). The paper is organized as follows. First, in section 2 we present the parallelization strategy that we have used. Then, in section 3 we present the evaluation and results. An finally, in section 4 we present the conclusions and future work.

2 Parallelization strategy

2.1 Macroblock-Level Parallelism. The 2D-wave strategy

In H.264 and other video codecs usually MacroBlocks (MBs) in a frame are processed in scan order, which means starting from the top left corner of the frame and moving to the right, row after row. MBs can be processed out of scan order provided these dependencies are satisfied. Processing MBs in a diagonal wavefront manner, as shown in Figure 2.1, satisfies all the dependencies and at the same time allows to exploit parallelism between MBs. We will refer to this parallelization as 2D-Wave [vdTJG03].

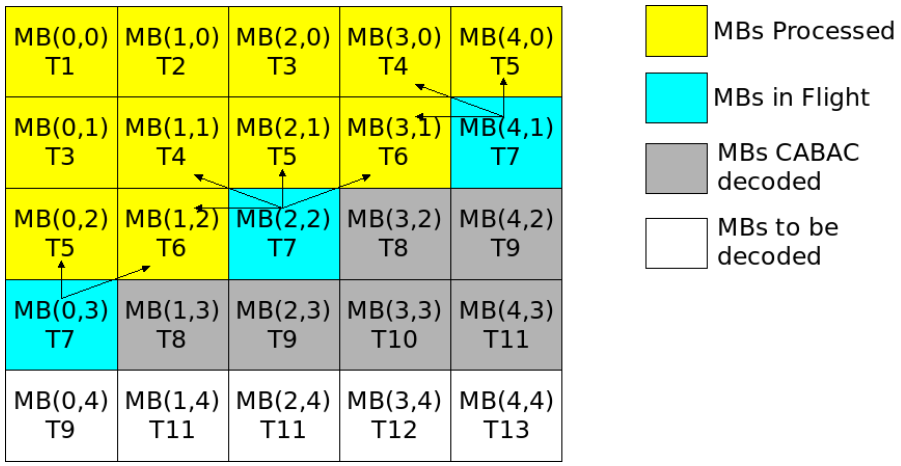

Figure 1: MB-level Parallelism Inside a Frame

With this strategy, the number of independent MBs in each frame depends on the resolution of the frame and changes during the frame processing time. For HD resolution (1920x1080 pixels) there are up to 60 independent MBs. Assuming that the time to decode a MB is constant and that there is not thread synchronization overhead the maximum speed-up for HD resolution is 32X (using 60 processors) [MAA+08]. These two assumptions are not true in a practical case: decoding time for each MB is not constant and there is some overhead for thread synchronization. In the next section we are going to analyse how to deal with these issues and show the effects of them on actual performance.

It is important to note that in our implementation entropy decoding is decoupled from the rest of macroblock decoding. First, CABAC entropy decoding is performed for all the MBs in a frame and the results are stored in a buffer. Once entropy decoding is performed, the decoding of MBs is executed in parallel using the 2D-wave strategy.

2.2 Implementation of 2D-wave for a SMP

Our implementation is based on a dynamic task model in which a set of threads is activated when a parallel region is encountered. In our context a parallel region means the decoding of all MBs in a frame. Each parallel region is controlled by a frame manager, which consist of a thread pool, a task queue, a dependence table and a control thread. The generation of work on the task queue is dynamic and asynchronous. When all the dependencies for a MB are resolved a new task is inserted on the task queue. Any thread from the thread pool can take this task and process it. When a thread finish the processing of a MB it updates the table of dependencies and if it founds that another MB has resolved its dependencies it can insert a new task into the task queue. As a further optimization step a tail submit method has been applied where each worker process MBs directly without passing through the task queue.

3 Performance analysis

3.1 Evaluation Platform

For these experiments we have used a modified version of the FFmpeg H.264 decoder with HD video inputs taken from HD-VideoBench [ASRV07]. The application was tested on a SGI Altix which is a shared memory machine, with a cc-NUMA architecture with 64 dual core IA-64 processors. Each one of the 128 cores works at 1,6 GHz, with a 8MB L3 cache and 533 MHz Bus, and the system has a total 512 GB RAM. The compiler used was gcc 4.1.0 and the operating system was Linux with kernel version 2.6.16.27.

3.2 Execution time

We used a higly optimized version of the 2D-wave H.264 decoder, which includes: dynamic and distributed scheduling, tail submit optimization and a lock-free mechanism for updating the table of dependencies. We executed the application for a different number of processors and take the time for the MB decoding loop

In Figure 3.2 it is shown the speed-up for the parallel region of the application compared to the sequential execution. The speed-up ranges from 1.85X for 2 processors up to 9.52X for 24 processors. Going beyond 24 processors results in diminishing benefits, with 32 processors the speed-up is 8.7X. The efficiency of the parallelization decreases with the number of processors: for 2 processors it is 92.5%, for 24 processors 39.7% and for 32 processors is 27.2%. Although this experiment reveals a significant amount of parallelism, the speed-up is far from the theoretical maximum of 32X for 60 processors. The main factor that inhibits the exploitation of all the available TLP is the overhead for accesing the centralized task queue. In average, it takes more time to add a new element to the task queue than to process it.

4 Conclusions

Although the theoretical analysis shows the 2D-wave parallelization of the H.264 decoder exhibits a lot of TLP the implementation on a real multiprocessor presents some scalability problems. With our implementation we have achieved a maximum 9.5X speed-up for the parallel portion of the code when using 24 processors. Using more processors does not result

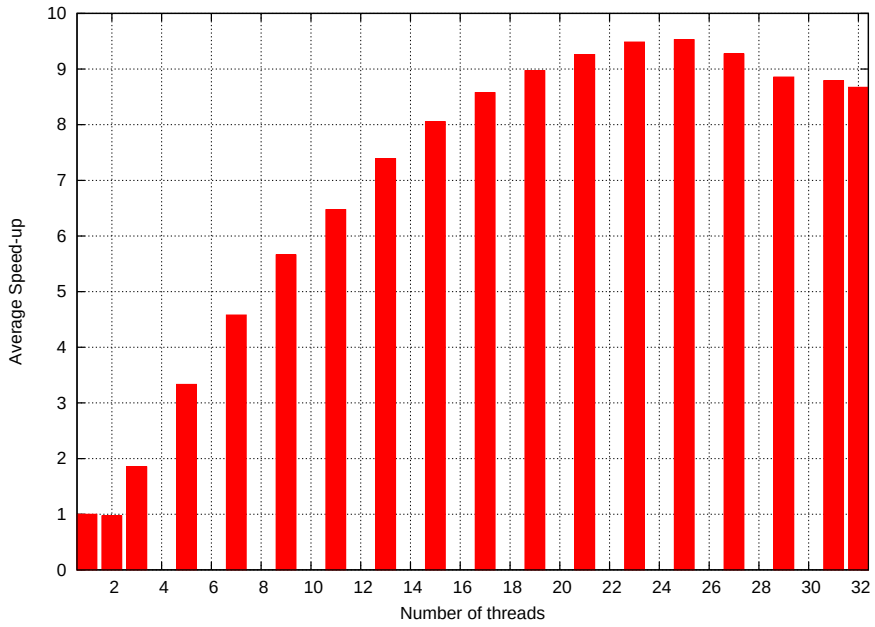

Figure 2: Average Speed-up for Decoding a Frame using the 2D-Wave Parallelization

in more performance gains. That is due to the overhead introduced by the central task queue structure. Getting and submitting elements from/to the task queue is expensive compared to the time required to decode a MB. The main conclusion from our implementation, in terms of scalability, is that is necessary to go for different implementations of the task queue or other synchronizations mechanisms that can reduce the overhead.

Currently we are working on alternative implementations of the task queue in order to reduce the synchronization overhead. Also we want to compare the same parallelization on heterogeneous multicore architectures, specifically in the Cell/BE processor. And finally we are working on the use of emerging programming models that can allow to express efficiently the kind of parallelism that we have implemented by hand using threads and locks.

References

[ASRV07] M. Alvarez, E. Salami, A. Ramirez, and M. Valero. HD-VideoBench: A Benchmark for Evaluating High Definition Digital Video Applications. In *IEEE Int. Symp. on Workload Characterization*, 2007.

[MAA+08] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez. Parallel Scalability of H.264. In *Workshop on Programmability Issues for Multi-Core Computers (MULTIPROG) 2008*, January 2008.

[vdTJG03] E.B. van der Tol, E.G. Jaspers, and R.H. Gelderblom. Mapping of H.264 Decoding on a Multiprocessor Architecture. In *Proc. SPIE Conf. on Image and Video Communications and Processing*, 2003.