

Gigabit Ethernet-based Parallel Video Processing

Horst Eidenberger

*Vienna University of Technology, Institute of Software Technology and Interactive Systems
Favoritenstrasse 9-11, A-1040 Vienna, Austria
eidenberger@ims.tuwien.ac.at*

Abstract

This paper describes solutions for parallel video processing based on LAN-connected PC-like workstations. We outline application scenarios for the processing of video with broadcast TV resolution and promising areas for future research. Additionally, we describe a prototype system implemented in our workgroup. This Network-of-Workstations is based on Gigabit Ethernet, free Java software and standard network protocols. Video data is streamed to and from processing hosts using IP multicast and the Real-time Transfer Protocol. Control mechanisms are based on network file sharing. In comparison to related approaches, our system makes use of high-performance network hardware and open source software. In consequence, our system is easier to implement, cheaper and more flexible than, for example, highly integrated commercial solutions.

1. Introduction

We have implemented a prototypical multimedia middleware for parallel video processing. The software layer is based on a Network-of-Workstations (NOW) connected by Gigabit Ethernet switches. We used this prototype to evaluate the principle feasibility of LAN-based processing of streamed media data. The advantages of such a setup over specialised hardware configurations are obvious: Using of-the-shelf hardware and free software reduces the costs enormously. Additionally, network clusters can easily be up-scaled (in the worst case by adding/exchanging network devices). If the software layer is based on free software and is itself implemented as open source, it can easily be adapted to new application scenarios. Finally, since every single component can be exchanged without affecting the rest of the installation, network solutions allow for greater flexibility and, in

consequence, potentially higher return of investment.

The presented idea is not that new. Papers [4, 5, 6] describe PSVP, a similar approach for Internet video processing based on hardware and software that were state-of-the-art in the late nineties. Section 2 describes the PSVP system. Subsection 4.2 discusses the differences of our approach to PSVP.

We have four reasons for rising this topic again. Since for the reasons given above, NOW-based video processing may become an important area of future multimedia research we wanted to stress the principle, its potentials and open research issues again. Especially, Subsection 3.3 lists (encouraged by the conference call) relevant areas requiring further research. Secondly, recent years' technological advances have lead to changed infrastructure parameters. On the hardware level, Gigabit Ethernet and increasing power of PC-like workstations as well as free software development kits for multimedia programming enable greater flexibility and allow us aiming at significantly more demanding applications. Thirdly, we propose an approach for spatial parallelisation only that is based on a simpler architecture for processing control and implemented by exclusively using standard operating system tools. Finally, technical limitations (mainly network bandwidth) forced restricting the earlier implementation on Internet video with lower spatial and temporal frequency. In contrast, our prototype is intended for manipulation of live video streams of television resolution (NTSC, PAL, SECAM). It allows for performing analysis and effect processing tasks as well as sophisticated compositing tasks involving multiple parallel streams. In conclusion, new developments have lead to a substantially changed situation that justifies making NOW-based video processing a multimedia research issue again.

The paper is organised as follows: Section 2 gives background information on earlier approaches (including PSVP). Section 3 discusses application

scenarios for parallel video processing and promising research areas. Section 4 sketches the architecture of our prototype. In particular, Subsection 4.2 lists the major differences to PSVP. Finally, Section 5 discusses implementation issues and relevant future developments in the area of multimedia software middleware.

2. Related work

For related work we concentrated on approaches using local area networks for video transport from host to host. We excluded DSP and VLSI chip design approaches as well as work on processing units and network design for MIMD architectures. Surprisingly, only a handful relevant publications could be identified. The earliest paper [2] exploits connecting video processing elements by an ATM-based LAN. The authors' focus is on broadcasting (including media capture, display, storage and retrieval), not video processing. They investigate the suitability of ATM technology for parallel video processing. In [1], principle solutions for spatial partitioning for video processing are discussed. The authors introduce novel heuristic algorithms for frame-wise spatial media splitting. Unfortunately, the paper does not contain experimental results. The interesting paper [7] describes pipelining strategies for spatially and temporally parallelised video processing. It describes a scheduler that schedules tasks at compile time. Two optimisation criteria are used: minimising the number of required processors for a task and maximising the throughput for a fixed number of processors. Parallel processing algorithms are automatically generated from high-level descriptions. All algorithms are implemented in a NOW and experimentally evaluated.

In [4, 5, 6] the authors describe the architecture of the Parallel Software-only Video effects Processing system (PSVP). [6] discusses methods and algorithms for temporal parallelisation. Similarly, [5] describes strategies and protocols needed for spatial parallelisation. In [4] the authors deal with control aspects of parallel video processing: selection of parallelisation method, distribution of algorithms and control of data flow. Below, the major aspects and components of the PSVP are described. See Subsection 4.2 for differences of our proposal to PSVP.

The PSVP was implemented in a NOW of 100 SUN Sparc workstations connected by 10 Mbps Ethernet. Two types of workstation roles are distinguished: effects server and effects processor. The effects server is responsible for setup and control of effects processors. Effects processors are used for parallel video processing. The application

communicates with the NOW through the effects server. Video streaming is based on UDP/IP and the Real-time Transfer Protocol (RTP) using MotionJPEG and self-defined proprietary formats as payload types.

Three software components are needed to run the system: FX compiler, FX mapper and FX processor. FX compiler translates high-level descriptions of video effects to binary code that can be executed on effect processor hosts. Effects are described as directed graphs by using the vocabulary of the Dali scripting language. Dali defines high-level operators for image processing. FX processor is a shell for effect program execution on effects processor hosts. FX processor handles configuration calls, video data reception and transmission. FX mapper maps spatio-temporal video segments to effects processor hosts. It implements a recursive partitioning process: Since both spatial and temporal parallelisation have their advantages, it would be desirable combining both methods in one video processing task. FX mapper is able to define a layered structure in which spatial and temporal parallelisation strategies are employed on each layer independently of strategies employed on other layers.

PSVP uses IP multicast for video transmission. The entire media stream is sent to all processing hosts. This approach guarantees that processors have all information they need and takes advantage of the network infrastructure by passing on the data multiplication process to the subnet. Recombination of processed frames is based on an intermediate format that allows easy merging. Similarly to the DV format, YCrCb channels are block-wise DCT encoded and transmitted using RTP. Temporal parallelisation is based on a selector function that redirects frames to effects processors and an interleaver function that recreates the video stream from processed frames.

3. Applications and problems of Parallel Video Processing

The three following sections are dedicated to the video processor we implemented. We will focus on the principal architecture and stress major differences to PSVP. This section lists use cases for parallel video processing and areas for future research. Section 4 sketches video processor design including control and data flow. Section 5 deals with implementation issues.

3.1 Basic use cases

Using a LAN-connected cluster of workstations for video processing is a generally appealing idea but in order to design a practically usable system, use cases

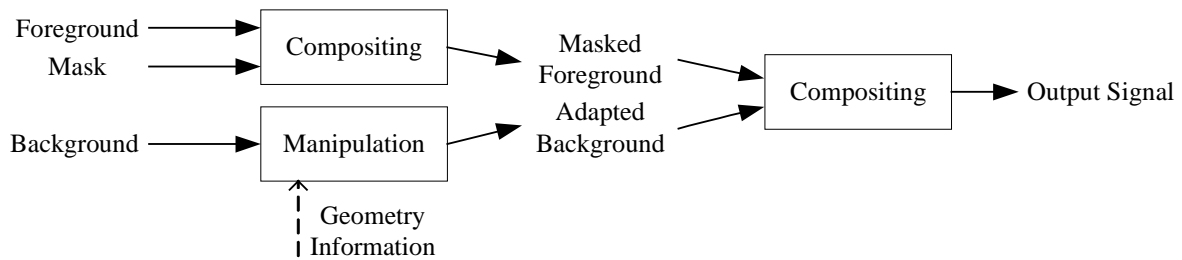


Figure 2. Parallel virtual studio application.

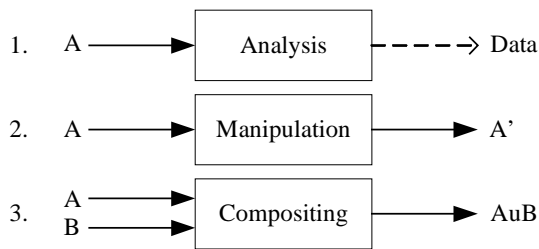


Figure 1. Basic video processing use cases.

have to be identified and requirements have to be derived. Such classic software engineering best practices are – even though being mandatory to estimate network, processing and software demands properly – hardly considered in PSVP and related projects. We have identified three basic application patterns (depicted in Figure 1). The *analysis* pattern takes one media stream as input and extracts media metadata. Exemplary applications are computer vision and visual information retrieval tasks. Analysis requires minimal network bandwidth at (likely) maximum processing power. *Manipulation* comprises all operations on one video stream (e.g. effect processing, format conversion). Manipulation network load is about twice as high as for analysis tasks. *Compositing* tasks combine two video streams (e.g. animation foreground and video background) to one output stream. Compositing induces the highest network load while processing is mostly simple alpha blending. Our video processor is able to deal with all three types of applications. We tested it, for example, with a tabletop soccer ball tracking application and a video effect that creates artefacts appearing in old movies.

3.2 Application use cases

More complex use cases can be assembled from these basic patterns. For example, a virtual studio application would require at least three input streams: foreground (including foreground mask), background and mask. Foreground mask and mask are replaced by the background. A virtual studio could be implemented

by firstly merging foreground and mask (compositing) and then, merging the result with the background video. Figure 2 illustrates the data flow. Geometry information is employed to guarantee that the (computer-generated) background scene is inserted from the correct perspective. If a camera matrix is used for tracking of foreground subjects/objects (e.g. speakers), analysis elements can be used to compute their coordinates.

As a second type of application, camera arrays could be implemented by series of analysis tasks (e.g. object tracking) and manipulation tasks (e.g. construction of panorama views). For 3D object reconstruction from video, input video signals could be merged pair-wise to stereo signals (by a compositing element). Then, depth information could be extracted from the stereo signals (analysis elements). The resulting data streams could be used matching, object recognition and reconstruction.

3.3 Problem areas

Several promising research directions can be derived from use cases and related work. Below, we consider four selected areas: parallelisation, setup and control, networking, software design.

Parallelisation can be performed functionally, spatially or temporally. Functional parallelisation is classical pipelining: chunks of media streams are manipulated in parallel using different operations in different steps. Since there is no difference to pipelining in other areas (e.g. microprocessor design), it is not relevant here. Spatial parallelisation raises two major problems: How/where should the media be split? Following the simplest solution would mean sending the entire media stream to all processing nodes. Obviously, this strategy puts high load on the network. For sophisticated application scenarios using multiple instances of the basic use cases, this strategy may be too resource-demanding. On the other hand, if the media is split, processing nodes get limited information that may turn out being insufficient (e.g. for analysis tasks). The second question is, how merging should be

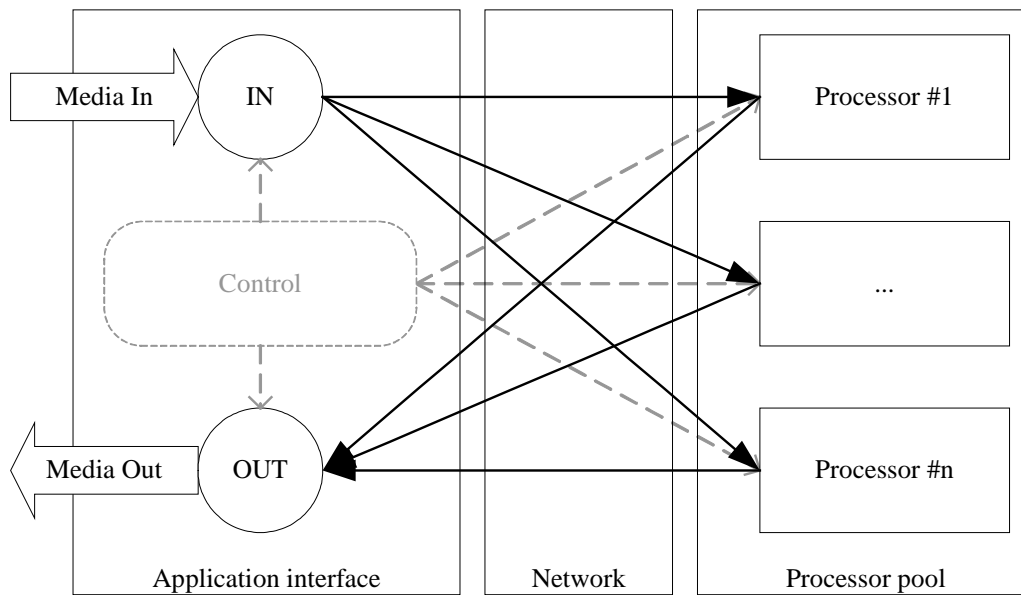


Figure 3. General video processor architecture.

performed. This includes issues from adequately designed media stream formats allowing fast, resource-saving merging to the definition of error handling procedures. Temporal parallelisation rises the questions, if streams with full or reduced frame rate should be redirected to processing nodes and how interleaving of processed streams should be performed. Answering the first question, again, a reasonable trade-off between resource consumption and availability of information has to be identified. Definition of interleaving strategies comprises intra-media synchronisation and handling of frame drops.

Setup and control problems include the avoidance of bottlenecks in splitter/merger components, network configuration and load balancing. Support of up-scaling of processing nodes is not sufficient to justify judging a NOW-based video processor being scalable. Performance bottlenecks can only be avoided, if splitting and merging tasks can be distributed over multiple nodes (e.g. by using hierarchies of hosts).

Network configuration includes distribution of processing scripts and parameters, and start/stop synchronisation. Performing configuration requires well-defined protocols and interfaces. For data transmission, suitable communication channels have to be defined. Closely related networking problems are bandwidth estimation and optimisation, selection of codecs as well as quality of service issues (estimation of availability of nodes, etc.). Furthermore, monitoring functionalities are required for controlling and fine-tuning of parallel processing.

The most relevant software design problem is providing a (visual) language for effect definition. As a major requirement the language has to allow easy parallelisation without the need for inter-process communication. Distribution of scripts to processing nodes must be as simple as possible. Distribution requires solving problems such as parameter distribution, synchronisation and interactive monitoring through distribution-transparent interfaces. Furthermore, quality of service issues (e.g. handling of package losses) have to be considered. The effect designer must be enabled to define error handling procedures tailor-made for his effect type. Finally, graph-based drag-and-drop user interfaces would be desirable for rapid prototyping of effect scripts. If standard components (e.g. colour transformations, edge operators) could be identified, user interfaces could easily be implemented that allow for assembling effect graphs from a palette of pre-defined components in a Visio-like style.

4. Prototype design

Our major goal in designing the video processor system (VP) was to make use of already existing (free) standard software whenever possible. The architecture should be simple and tailor-made for the described use cases. Open source software has to be considered as an important factor in software engineering and, today, even for multimedia applications remarkably powerful free tools do exist.

4.1 Architecture

Figure 3 sketches the VP architecture. Video streams are read and distributed to processing nodes by an IN-node. An OUT-node is responsible for collecting and reassembling the processed media streams. IN and OUT are software components that can either reside on a single host, on two dedicated hosts or be distributed over a hierarchy of hosts. This flexible concept allows scalability for video stream splitting and merging and avoids the major bottleneck of PSVP.

The Control component is responsible for network setup and configuration of processing nodes. Currently, VP supports only spatial parallelisation. Media data is transported through the network using the Real-time Transfer Protocol, RTCP for quality control and a Photo JPEG codec with lossless compression. We selected this codec because it provides effective compression at minimum encoding/decoding costs. The input media stream is sent to all processing nodes in its entirety using IP multicast on the router level. Only processed video data is sent back using the same codec and RTP unicast. Since the VP network is based on Gigabit Ethernet, sufficient bandwidth is available to support all basic use cases for TV resolution videos (NTSC, PAL, SECAM) at minimum compression (see Subsection 5.2 for performance evaluations).

Effects can be defined in a high-level programming language. Concrete implementations are derived from a template that guarantees that spatial parallelisation can easily be accomplished. Basically, the template dictates frame processing in loops over rows and columns. The Control component distributes binary effect programs to processing nodes using a standard network file sharing protocol. A processor shell runs on every processing node that is able to execute digital effect scripts. Before processing can be started, Control configures IN and OUT by associating spatial regions with the IP addresses of processing nodes.

As mentioned above, IN and OUT are software-only components and may exist in multiple instances. Associations can be made arbitrarily. Therefore, it is, for example, possible to define a configuration with a shadow fail-back system. Providing such features makes the VP architecture suitable for high availability applications (TV broadcasting, etc.)

Video processing starts with the first frame distributed by IN. Since we use RTP for streaming, there is no way but providing sufficient network bandwidth to guarantee quality of service. Luckily, in our experiments Gigabit Ethernet turned out to be sufficiently fast. The OUT component manages output synchronisation using a large cache memory.

4.2 Differences of VP to PSVP

The architecture of VP is generally simpler: VP is assembled of less components. All software components used are released under open source licenses. The hardware components involved are cheap PC-like workstations and of-the-shelf network switches. Still, VP supports processing of video with full TV-resolution. The PSVP was designed to be used on Internet video streams with a resolution of 320 by 240 pixels.

VP components can be arbitrarily combined. It is even possible to assemble circuits from multiple VP instances. This feature allows for implementing sophisticated application scenarios (e.g. virtual studios) from basic use cases. Furthermore, since video stream input and output operations can be distributed over multiple machines, the input/output bottleneck is avoided in VP.

Generally, VP is based on simpler control mechanisms: Standard protocols are used for media and control data transmission. Therefore, standard network tools can be used for configuration and monitoring. All components use the same communication interfaces and the same codecs on all transmission channels. Hence, VP supports highest configuration flexibility. On the other hand, configuration is performed on a technically lower level than in the PSVP. Especially, currently VP supports only configuration for spatial parallelisation.

5. Implementation

5.1 Software

VP is an ongoing open source project. Currently, control mechanisms are only partially available and the OUT component has only alpha status. All other components are at least on a beta quality level. All media processing software components (including effect templates) are implemented in the Java programming language. We are using the Java Media Framework [3] for video manipulation and streaming instead of Quicktime for Java, because it is open software, can be used free of charge and implements RTP, RTCP and the required codecs. At the current point in time, other options do not exist. The Control component is implemented using the Samba protocol and Perl scripts. All nodes are running Linux operating systems and identical Java Virtual Machines.

Even though Java performs excellently in our test environment, future implementations may have to be based on a lower but faster level. Unfortunately, currently state of the art multimedia environments as

DirectShow, Quicktime and OpenML do not support RTP-based streaming. Furthermore, they are only available for a few operating systems.

5.2 Gigabit Ethernet network

Gigabit Ethernet technology is the major advance that allows performing professional parallel video processing in LAN environments. To proof the concept, we evaluated the networking performance of a standard Gigabit Ethernet switch with 16 ports by connecting three processing nodes.

Table 1. TV format parameters and required network bandwidth for raw and JPEG-compressed streams.

Format	Resolution	Data rate (raw/JPEG)
NTSC	720x480 px, 30 fps	158,0 / 19,2 Mbps
PAL, SECAM	720x576 px, 25 fps	158,2 / 16,56 Mbps
PALplus	960x432 px, 25 fps	158,2 / 11,6 Mbps

We used the VP software for a compositing application on NTSC, PAL and PALplus (16:9 aspect ratio) video streams. Applying lossless JPEG compression, the NTSC stream turned out to require the highest bandwidth. See Table 1 for details. The depicted required bandwidth is without overhead for RTP- and other protocol headers and without considering RTCP communication bandwidth. In a compositing scenario, IN sends two video streams to the processing pool using multicast. The streams sent from processing nodes to OUT sum up to one unicast with full TV resolution. We implemented this scenario and tested it with a number of test streams over a duration of 20 minutes per test. Independent of the media format, we could *not* observe a *single* frame drop in any test.

These results prove that Gigabit Ethernet provides technically sufficient (and cheap) foundations for highly sophisticated parallel video processing tasks. Since the network turned out to be that reliable, using RTP based on unreliable UDP is also justified as being a sufficiently good solution.

6. Conclusions and future work

We present a system for parallel video processing that is based on of-the-shelf hardware (PC-like workstations and Gigabit Ethernet) and free software. The principle is not new: the PSVP system [6]

implements a similar idea. Major differences to our approach are that Gigabit Ethernet allows performing sophisticated video processing tasks in TV resolution and that all components of our system are free software. Still, there are many open research problems that should be tackled in the future. Our future work will include a reimplemention based on the OpenML framework as well as performance evaluations for larger networks.

7. Acknowledgements

The author would like to thank Christian Breiteneder and Axel Filipovic for their valuable comments and suggestions for improvement.

8. References

- [1] D.T. Altilar, and Y. Paker, "Minimum Overhead Data Partitioning Algorithms for Parallel Video Processing," *Proceedings Domain Decomposition Methods Conference*, Chiba Japan, Oct. 2001, pp. 251-258.
- [2] A. Guha, A., Pavan, J., Liu, A., Rastogi, and T. Steeves, "Supporting Real-Time and Multimedia Applications on the Mercuri Testbed," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, IEEE Press, May 1995, pp. 749-763.
- [3] SUN Microsystems, Java Media Framework Website, <http://java.sun.com/products/java-media/jmf/> (last visited: 2004-09-10).
- [4] K. Mayer-Patel, and L.A. Rowe, "A Multicast Control Scheme for Parallel Software-only Video Effects Processing," *Proceedings ACM Multimedia Conference*, ACM Press, Orlando FL, Nov. 1999, pp. 409-418.
- [5] K. Mayer-Patel, and L.A. Rowe, "Exploiting Spatial Parallelism for Software-only Video Effects Processing," *Proceedings SPIE Multimedia Computing and Networking Conference*, SPIE Press, San Jose CA, Jan. 1998, pp. 28-39.
- [6] K. Mayer-Patel, and L.A. Rowe, "Exploiting Temporal Parallelism for Software-only Video Effects Processing," *Proceedings ACM Multimedia Conference*, ACM Press, Bristol UK, Sep. 1998, pp. 161-169.
- [7] M.T. Yang, R. Kasturi, and A.A. Sivasubramaniam, "Pipeline-Based Approach for Scheduling Video Processing Algorithms on NOW," *IEEE Transactions on Parallel and Distributed Systems*, IEEE Press, vol. 14, no. 2, Feb. 2003, pp. 119-130.