A Concurrent Ubiquitous Video Streaming Platform Design Using SVC Techniques and Multi-core Parallelism Programming

Chung-Ming Huang, *IEEE Senior Member*, Wan-Ping Tsai and Chung-Wei Lin, *IEEE Student Member*
Laboratory of Multimedia Mobile Networking
Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan, R.O.C.
Correspondence: huangcm@locust.csie.ncku.edu.tw

***Abstract*—In the decade, CPU manufacture industries are highly prompting multi-core CPU products, e.g., Intel Core2 Duo and AMD Athlon64 x2. The merit of the multi-core CPU is able to dispatch multiple tasks in distinct CPU cores and boost the execution speed in parallel. In this paper, we design a Concurrent Ubiquitous Video streaming platform using Multi-Core parallelism programming and the scalable extension of H.264/AVC, which is called MC-CUV. The proposed streaming platform is able to serve lots of heterogeneous users who may utilize distinct terminals through distinct network interfaces to get on-demand videos from an ISP streaming server at meanwhile. In our experiments, distinct video's spatiotemporal resolutions and SNR quality are presented depending on the device capability and network condition. Comparing with the multi-core single-thread streaming approach, the average execution time can decrease effectively because multiple streaming tasks can be executed in parallel using a proposed task dispatcher through multiple CPU cores.**

***Index Terms*—Multi-core parallelism programming, the scalable extension of H.264/AVC, ubiquitous video streaming.**

1. INTRODUCTION

In 1965, Gordon Moore predicted the number of transistors on a chip would double every 12 months in the near future (a.k.a. Moore's Law) [1]. The Moore's Law depicted a roadmap for the product engineer when they design the layout of electronics transistors. In addition to the increased number of transistors, the CPU clock frequency also doubled every 18 months in the series of Intel Pentium processor (1993-2003) [2]. However, because of increased power consumption (heating) and design complexity, the CPU clock frequency has the bottleneck around 4GHz. To overcome the bottleneck of the processing speed, many CPU manufacture industries are highly promoting distinct processor architectures, e.g., CMP (Chip-level Multi Processing) and SMT (Simultaneous Multi Threading) [3]. It is noted that the emerging multi-core CPU belongs the CMP architecture. The merit of the multi-core CPU is able to assign multiple tasks to be executed in distinct CPU cores, such that the overall processing time can decrease effectively. Thus, emerging multi-core programming is able to make multiple tasks be executed in parallel, in which paralleled algorithms are frequently implemented using OpenMP [4]. Unfortunately, the most of current softwares, e.g., games, computer vision, and Internet applications [5][6][7], are still developed based on the single-core processing architecture, instead of the multi-core processing architecture. That is, no data parallelism or function parallelism are considered in the most of software development.

In this paper, we design a concurrent ubiquitous video streaming platform using multi-core parallelism programming and Scalable Video Coding (SVC) techniques, called MC-CUV. Considering the coming era of ubiquitous video streaming, lots of heterogeneous users may utilize distinct terminals, e.g., PDA, Laptop and desktop computer, through distinct network interfaces, e.g., wired, Wi-Fi, 3G and Wi-Max, to get on-demand videos from an ISP streaming server at meanwhile. In order to satisfy distinct users' quality requirements, one of the MC-CUV server functions is able to generate specific spatiotemporal resolutions and fidelity video bitstreams using novel scalable video coding, e.g., the scalable extension of H.264/AVC. Lots of heterogeneous users, routine maintenance tasks, session mobility management and complicated quality adaptation may result in heavy server loading in practical. Thus, in MC-CUV, distinct users' streaming tasks are designed to be processed by distinct CPU cores through a proposed task dispatcher.

The rest of this paper is organized as follows. Section 2 surveys the multi-core processor architecture, the usage of OpenMP and the scalable extension of H.264/AVC. Section 3 introduces the proposed MC-CUV architecture. Section 4 introduces algorithms of the MC-CUV streaming parallelism. Section 5 exhibits experiment results and Section 6 has concluding remarks.

2. RELATED WORKS

In this Section, we mainly survey the multi-core processor architecture, the usage of OpenMP and the scalable extension of H.264/AVC.

2.1. The multi-core processor architecture

In these years, multi-core processors are highly-promoted by Intel and AMD, and has rapidly grew when single-core processors reach physical bottleneck. To date, in the Top100 supercomputers, more and more processors fall into the multi-core processor family. The multi-core CPU (a.k.a. chip-level multiprocessing, CMP) is a single CPU package that consists of two or more independent processing cores, and then each program thread is executed on distinct CPU cores. Distinct

*This research is supported by the National Science Council of the Republic of China, Taiwan under the contract number NSC 97-2219-E-006-002, Information & Communications Research Labs (ICL), Industrial Technology Research Institute (ITRI), Taiwan, ROC, and Intel Microelectronics Asia Ltd., Taiwan Branch.

CMP package technologies can be classified as follows: (i) all cores on one die, e.g., Athlon 64 x2 and (ii) multiple dies each with several cores, e.g., Core 2 Quad. It is noted that caches, bus or memory are or are not communicated among distinct CPU cores.

In addition to CMP, SMT is the other feasible processor architecture that enables a single core to execute multiple threads simultaneously. Intel Pentium 4 3.06GHz model released in 2002 was the first desktop processor that support two-thread SMT engines (a.k.a. Hyper-Threading Technology (HT)). From the experiment results, up to a 30% speed improvement can be gained in comparison with other non-SMT Pentium 4 CPUs. In the future, combined CMP and SMT that allow an individual CPU core to execute multiple threads is the trend.

2.2. The introduction to OpenMP

The OpenMP (Open Multi-Processing) is a parallel multiprocessing API that is used for multi-platform shared-memory programming in C and Fortran. In the OpenMP parallelism, the master program is able to fork (split) several slave threads and allocate them to different processor cores using high-level instruments. The OpenMP instrument can be formatted commonly by

"#pragma omp *directive* [*clause*]".

where (1) *directive* provides links to directives used in the OpenMP API, e.g., *for, parallel, sections*, etc.; (2) *clause* provides links to clauses used in the OpenMP API, e.g., *if, nowait, num_threads*, etc. More detail information of OpenMP instruments can be found in MSDN [8].

It is noted that a good parallelism programming algorithm is able to improve the application performance as OpenMP is used. Data-domain and task-domain parallelism algorithms are two common approaches to parallelize programs. The data-domain parallelism algorithm is able to fragment data into several independent sub-data, which require similar computation cost; the task-domain parallelism algorithm is able to decompose the original computation into multiple thread-stages. D.E. Culler, et. al. considered that the data-domain parallelism is more scalable than the task-domain parallelism because less load-imbalance and shared-data synchronization occurred in the data-domain parallelism [9].

2.3. The scalable extension of H.264/AVC

In consideration of network heterogeneity and device diversity, streamed video quality should be adapted to distinct spatiotemporal resolutions and SNR quality. To achieve the goal, many approaches of scalable video coding were proposed, e.g., FGS (fine-granular scalability), FGST and the novel scalable extension of H.264/AVC. The reference software of the scalable extension of H.264/AVC named JSVM (Joint Scalable Video Model) aims at providing the combined scalabilities, error robustness and graceful degradation for a variety of network conditions and applications [10]. JSVM utilizes a 2-D spatial decimation that generates the lower spatial-resolution signal, e.g., from 4CIF to CIF/QCIF, to enable the spatial scalability. Using the technique of the inter-layer prediction, higher spatial-resolution motion information and texture signals can be predicted for the intra-block prediction and motion coding. JSVM utilizes the hierarchical B picture prediction structure to achieve the temporal scalability, in which the last picture of each GOP (Group of Picture) can be coded either as an I or P picture. Besides, FGS is realized by repeatedly decreasing the quantization parameter and applying the modified entropy coding process. Nowadays, SVC is currently being developed by the joint effort from both ITU-T VCEG and ISO/IEC MPEG.

3. THE MC-CUV SYSTEM ARCHITECTURE

In the proposed MC-CUV, multi-core parallelism programming is used to increase the processing speed when lots of heterogeneous users' requests are concurrently coming and requesting streaming services; the scalable extension of H.264/AVC is used to provide distinct spatiotemporal resolutions and bitrates of transmitted videos for satisfying device diversity and network conditions. The complicated tasks of MC-CUV contain SVC video data delivering, network congestion prediction and session management. Referring to Fig. 1, major components of the MC-CUV streaming platform are as follows:

(1) The MC-CUV management system: The MC-CUV system provides a web portal (user interface) that manages uploaded videos and compresses these videos to SVC bitstreams based on the user's encoding configuration. The encoded SVC data are stored in the large movie repository. Partial management functions of MC-CUV, e.g., SVC encoding and session mobility, were appeared in our previous paper [11].

(2) The socket interface: In MC-CUV, socket protocols, i.e., SIP/SDP and TCP, are used for session signaling and data delivering, respectively. SIP is mainly responsible for creating/managing/terminating sessions and delivering the movie menu, which is implmented using the eXosip library [12].

(3) The network-congestion detection module: The network-congestion detection module mainly contains U-BEKF (Bandwidth Estimation Using Kalman Filter for Ubiquitous Video Streaming) for predicting the currently available bandwidth. U-BEKF is our previous investigation w.r.t. the bandwidth estimation using the Kalman filter [13]. According to the packet RTT and loss-rate reported in the RTCP packet, the precise available bandwidth (AvB) can be predicted based on the results of the prediction and measurement models. In the proposed MC-CUV task dispatcher, AvB is a critical factor for determining each user's transmission priority.

(4) The quality adaptation module: The quality adaptation module mainly contains two kernels: (i) the Quality-of-Presentation (QoP)-decision kernel and (ii) the real-time SVC bitstream extractor. The QoP-decision kernel is responsible for determining the proper spatial resolution, frame rate and transmission bitrate. Referring to Fig. 2, the spatial resolution depends on the hardware monitor, the frame rate depends on the memory capacity/CPU core, and the bitrate is truncated based on the available bandwidth (AvB). According to the

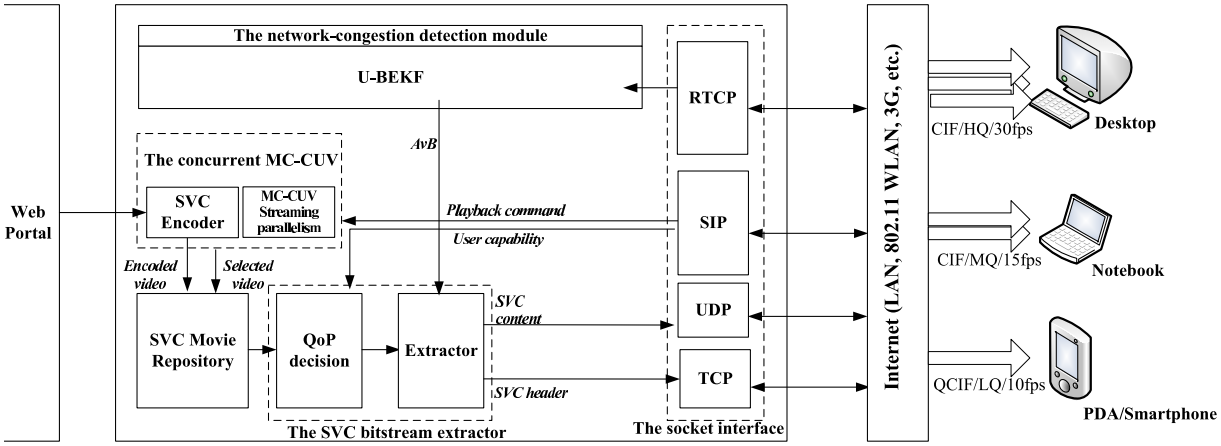
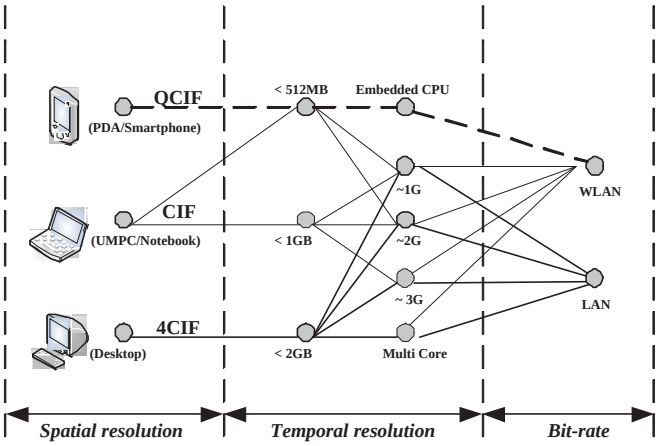


Fig. 1. The system architecture of MC-CUV.



Class A memory < 512MB	1.875 fps for Embedded CPU
	3.75 fps for 1G CPU
	7.5 fps for 2G CPU
Class B memory 512MB- 1G	3.75 fps for 1G CPU
	7.5 fps for 2G CPU
	15 fps for 3G CPU
Class C memory 1G - 2G	7.5 fps for 1G CPU
	15 fps for 2G CPU
	30 fps for multi-core CPU

Fig. 2. The proposed hybrid QoP-decision rule.

QoP-decision result, the real-time SVC bitstream extractor extracts the proper quality of SVC bitstreams, including base-layer and partial enhancement-layer bitstreams. With the advances of SVC, QCIF/CIF/4CIF (spatial resolution), 1.875fps-30fps (temporal resolution) and any transmission bitrate can be combined arbitrarily for the user QoP requirement.

4. The MC-CUV Streaming Parallelism Algorithm

As mentioned before, a good parallelism algorithm is able to increase the multi-core system performance. Thus, MC-CUV tasks should be firstly identified and classified, e.g., SIP session management, TCP/UDP data transmission and quality adaptation. Fig. 3 shows the execution flowchart of MC-CUV, which mainly consists of 'Menu', 'Extract' and 'Streaming' stages. The 'Menu' stage is responsible for delivering the on-demand movie menu to multiple clients using the SIP protocol. The 'Extract' stage (quality adaptation) is responsible for extracting SVC bitstreams based on the user preference (manually) or the result of the proposed QoP-decision rule (automatically). The 'Streaming' stage is responsible for transmitting SVC bitstreams to multiple clients through a task dispatcher in parallel. It is noted that 'Menu', 'Extract' and 'Streaming' are regarded as task independecy because more time-consumption of SVC streaming stage are required than 'Menu' and 'Extract' stages in our experience. In MC-CUV, each client's data delivery is independent in the 'Streaming' stage.

Thus, the major obstacle that should be addressed is how to design the concurrent ubiquitous video streaming platform to make multiple streaming tasks be performed in parallel in distinct CPU cores. The proposed MC-CUV streaming parallelism algorithm contains a task dispatcher and OpenMP programming, which are detailed as follows:

(1) Task dispatcher design: The task dispatcher is able to select multiple coming user connections to deliver SVC video data they preferred based on a weighted priority (κ). The higher connection speed (τ), smaller video quantity (ς) and longer waiting time (γ) will result in the higher weighted transmission priority, i.e., $\kappa \propto \tau \ \propto \gamma \propto 1/\varsigma$. In the operating system, the CPU scheduling theory points out the shortest-job-first (SJF) algorithm is an optimal solution to schedule the process queued in the CPU and decrease the waiting time of each process. Like SJF, the proposed task dispatcher is able to select user connection requests with top-n transmission priorities from a socket queue.

(2) OpenMP programming: when multiple clients' connections are selected, they will enter into parallelized

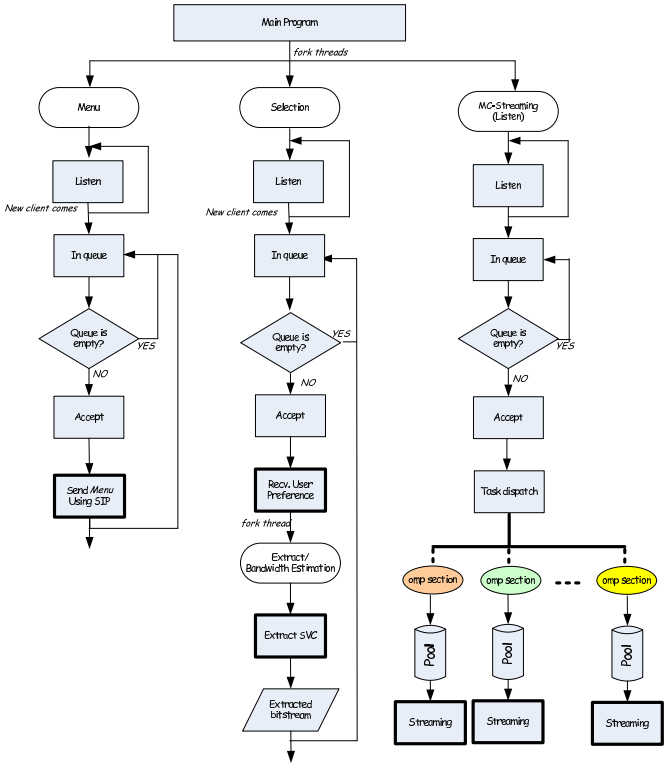


Fig. 3. Parallelized tasks in MC-CUV (black boxes: code sections that are able to be executed in parallel).

sections that is made using OpenMP. The directive of OpenMP *sections* is able to identify code section to be assigned among all threads, e.g.,

```
#pragma omp parallel sections num_threads(n)
{
  #pragma omp section
  {
    MC-CUV streaming (0)
  }
    ......
  #pragma omp section
  {
    MC-CUV streaming (n-1)
  }
}
```

Then, Operating System will assign these sections codes (multiple MC-CUV streaming tasks) into distinct CPU cores automatically, and thus these streaming tasks can be performed in parallel.

5. Experimental Results

In this Section, we will exhibit the snapshot of the MC-CUV execution among heterogeneous networks and devices, the SIP usage, and the performance of the MC-CUV streaming. The experimental equipments are HP Compaq Notebook (Intel Core2 Duo 2.4GHz, 2GB memory ram) and Acer desktop computers (Intel Core2 Quad 2.4GHz, 2GB memory ram).

In the MC-CUV streaming platform, when the quality of streamed videos are determined using the proposed QoP-decision rule, the SVC bitstream extractor is able to extract specific quality sub-bitstreams (base-layer plus partial enhancement-layers) accordingly. Referring to Fig. 4, distinct on-demand video qualities are decoded and presented for distinct devices, e.g., CIF (desktop computer) and QCIF (PDA). In addition, SIP is used for delivering the movie menu, and its SIP/SDP message can be observed by Ethereal, which is also shown in Fig. 4.

In addition, the proposed MC-CUV is compared with the multi-core single-thread application. In our simulations, 10, 100, 1000 user connections are performed concurrently, and the average execution times are compared when the same on-demand videos are requested under the same assumption of network conditions. Referring to Table I, when 100 user connections are coming concurrently, the average execution time is 815.1 seconds for the video transmission at the MC-CUV streaming platform, and the execution time is 1413.6 seconds at the multi-core single-thread streaming application. About 31%-43% gain can be obtained. Besides, balanced CPU loading (near to 100% utilization) is also observed in the MC-CUV streaming server side, which is shown in Fig. 5.

6. Conclusion

In this paper, we propose a Concurrent Ubiquitous Video (CUV) streaming platform using multi-core parallelism programming and the scalable extension of H.264/AVC (SVC). To achieve the ubiquitous video streaming among heterogeneous networks and devices, scalable layered on-demand videos that

Fig. 4. Distinct quality-of-presentation videos using the bitstream extractor and the SIP/SDP message capture.

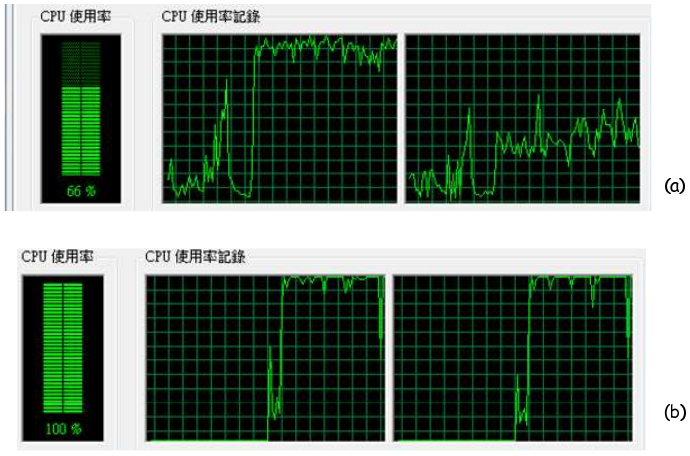


Fig. 5. The comparisons of the CPU utilization at (a) the multi-core single-thread streaming application (45%-70%) and (b) the multi-core multi-thread MC-CUV streaming platform (95%-100%).

TABLE I
THE COMPARISON OF EXECUTION TIME FOR THE PROPOSED MC-CUV AND SINGLE-THREAD STREAMING PROGRAMS.

Total concurrent connections	MC-CUV execution time	Single thread execution time	Gain
10	83.6	120.7	31%
100	815.1	1413.6	43%
1000	8512.3	13532.4	37%

are compressed using SVC are provided. Distinct Internet protocols, e.g., TCP/UDP and SIP, are used for data delivering and session message exchanges, respectively. In addition, to serve lots of heterogeneous users who may utilize distinct terminals through distinct network interfaces to get on-demand videos from an ISP streaming server at meanwhile, a parallelized MC-CUV streaming programming are developed using OpenMP. Different user tasks can be assigned into distinct CPU cores using the proposed task dispatcher in consideration of user connection speed, quantity of transmitted videos and its waiting time. In our experiments, the proposed MC-CUV is able to make CPU loading more balanced, increase the CPU utilization and decrease the execution time about 31%-43% effectively. In the future, a multi-core computer server cluster will be designed to increase the streaming efficiency and achieve the goal of the load-sharing further.

REFERENCES

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits", *Electronics*, April 19, 1965.
[2] J. Parkhurst, J. Darringer, B. Grundmann, "From single core to multi-core: preparing for a new exponential", *Proceedings of the IEEE/ACM International Conference on Computer-aided Design*, pp. 67-72, November 2006.
[3] R. Noronha and D.K. Panda, "Improving Scalability of OpenMP Applications on Multi-core Systems Using Large Page Support", *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pp. 1-8, March 2007.
[4] Ommi OpenMP Compiler Project. [on-line] http://phase.hpcc.jp/Omni/
[5] T. P. Chen, D. Budnikov, C. J. Hughes, Y. K. Chen, "Computer Vision on Multi-Core Processors: Articulated Body Tracking," *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 1862-1865, July 2007.
[6] Q. Zhang, Y. Chen, J. Li, Y. Zhang and Y. Xu, "Parallelization and Performance Analysis of Video Feature Extractions on Multi-Core Based Systems," *Proceedings of the IEEE International Conference on Parallel Processing*, September 2007.
[7] W.H. Li, A. M. Zhang, L and Kleeman, "Real Time Detection and Segmentation of Reflectionally Symmetric Objects in Digital Images," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4867-4873, October 2006.
[8] [on-line] http://msdn.microsoft.com/zh-tw/library/2kwb957d.aspx
[9] D.E. Culler, J. Pal Singh and A. Gupta, "Parallel Computer Architecture: A Hardware/Software Approach", Morgan Kaufmann Publishers, 1999.
[10] [on-line] The SVC reference software JSVM, http://ftp3.itu.ch/av-arch/jvt-site/2005
[11] C.M. Huang, T.Y. Wang, H.J. Lai and C.W. Lin, "An H.264 SVC-based Robust Video Streaming System Using the Adaptive Interleaving Coding Technique and Unequal Error Protection", *Proceedings of Workshop on Consumer Electronics and Signal Processing*, pp. 90-96, November 2007.
[12] [on-line] http://www.gnu.org/software/osip/
[13] C.M. Huang, C.W. Lin and X.Y. Lin, "A Predictive Video-on-Demand Bandwidth Management Using the Kalman Filter 4 over Heterogeneous Networks", *The Computer Journal*, doi:10.1093/comjnl/bxn011, March 2008.