On Utilization of the Cluster Computing Technology for Video Conversion*

Chao-Tung Yang†, Chin-Ming Chi, and Ping-I Chen

High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University, Taichung, 40704, Taiwan R.O.C.
email: {ctyang, g932834}@thu.edu.tw

Abstract

The popularization of broadband network and the development of MPEG-4 compression technology urge to the multiplexing development of Internet information. Among these techniques, Video-on-Demand is the most popular service. However, it takes extremely long compression time to convert audio and video data into MPEG-4 format. Although MPEG-4 enhances the compression ratio, it still needs massive storage equipment to deposit the audio and video data. The price of MPEG-4 related hardware equipment still stays at a high level currently. Thus, these problems can easily be solved by using the cluster computing technology, or PC Clusters. In this paper, we use the Linux PC cluster to achieve the high performance video conversion. Moreover, we use diskless cluster computing technology to make it more convenient in the system administration. In video conversion aspect, we use software tool called "dvd::rip" to perform in parallel the video compression, with the goal to achieve the best execution time, by enabling that each node to perform in its best processing potency.

Keywords. *Cluster computing, Linux PC Cluster, Video conversion, MPEG-4.*

1. Introduction

Advances in media technology permit to store the content of complete DVDs in a CD-ROM, without any noticeable loss of quality. This makes buying an expensive DVD burner with limited record capacity obsolete. To copy a video with up to 9 GB from a DVD to a CD-ROM requires large amount of computing power and time. All the data volume must be reduced to about a 12th of its original size to accommodate the 700 MB of limited storage capacity of the CD-ROM. A data compression of this magnitude for digital video is only possible with the new video compression standard MPEG-4. Generally speaking, MPEG-4 is an extension of the MPEG-2 technology, but MPEG-4 can be used more universally, with additional extensions. Generally, when we want to convert a DVD title to a MPEG-4 format on a single PC, we should perform the following steps as shown in Figure 1.

It is difficult to reduce the transfer time from the DVD to a storage device, unless upgrading the transfer bus to the SCSI speed or to use a RAID storage system. The Video Conversion time is the key to reduce the total time. The sequential processing machine needs a conversion time of 5 hours. If we divide the video file and then submit the sub-file to the different conversion computing nodes, it's possible to reduce the video conversion time. It is performed the broker in the cluster computing to find an available resource then to complete the video conversion job. After each node completes his conversion job, the result is sent to the master node of the cluster environment and then combined. It can save quite a large amount of time. The references listed show this need and technique usage [4, 8, 9, 18, 21, 22]. See Figure 2 for additional details.

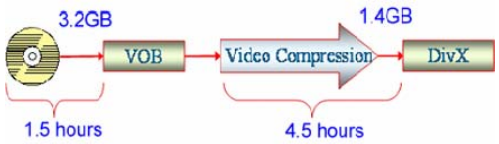

Fig. 1. DVD conversion – single stream.

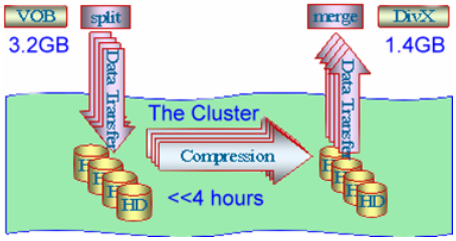

Fig. 2. Video conversion – using cluster computing technology.

In this paper, we propose a Linux PC cluster platform to achieve high-throughput DVD transcoding, which is, we investigate the use of

* This work is supported in part by National Science Council Taiwan, under grant no. NSC94-2622-E-029-002-CC3.
† The corresponding author

cluster computing to perform the DVD conversion, by using dvd::rip application running in cluster computing testbed. The dvd::rip is a full featured DVD copy program written in Perl. It provides an easy to use but feature-rich Gtk+ GUI to control almost all aspects of the ripping and transcoding process, and it uses the widely known video processing swissknife transcode, as also many other Open Source tools. dvd::rip itself is licensed under GPL / Perl Artistic License. Thus, we utilize different test models to shorten the waiting time between nodes, which will cause the execution time as short as possible.

The remaining of this paper is organized as follows. In section 2, some background reviews are discussed, while in section 3 experimental platforms and conversion applications are introduced. In section 4, experimental results are discussed, and finally, in section 5, some conclusion remarks and future works are described.

2. Background Review

2.1. Video Format

The history of MPEG goes back to the year 1987. MPEG stands for Motion Pictures Expert Group, a worldwide organization that develops manufacturer and platform independent standards for video compression. The first result was introduced as MPEG-1 in 1992. It was the basis for the less successful European Video-CD. Because of its limited resolution of 352×288 pixels, MPEG-1 is only suitable for the home environment, and the achievable video quality in relation to the data rate is rather low from today's point of view. MPEG-2 was introduced in 1995 and is mainly based on MPEG-1. The higher resolution with a maximum of 720×576 pixels is a major improvement enabling a significantly better video quality. The MPEG-4 was released by the MPEG group in December 1999. The detail information about MPEG-1, MPEG-2, and MPEG-4 is listed in Table 1.

2.2. MPEG4 Conversion

In 1999, the MPEG-4 specifications for the encoding of audio and video sequences were completed [5, 6, 18]. They define a system that is much more complex and requires much more computational power for the encoding than former MPEG specifications [16]. This work led to the need for efficient tools and mechanisms that help the implementation of systems based on this new specification.

The MPEG-4 specification is the first encoding specification that represents contents as a set of audiovisual objects that compose a scene, and have a defined behavior both in time and space. There are a number of tools that can be used to describe a scene, and each will give rise to a different class of objects. This paper focuses exclusively on natural video objects. These objects are the result of the evolution and extension of the MPEG-1 and MPEG-2 specifications.

Table 1. Comparison of MPEG-1, MPEG-2, and MPEG-4

	MPEG-1	MPEG-2	MPEG-4
Available since	1992	1995	1999
Max. video resolution	352×288	1920×1152	720×576
Default video resolution (NTSC)	352×288	640×480	640×480
Max. audio frequency range	48 KHz	96 KHz	96 KHz
Max. number of audio channels	2	8	8
Max. data rate	3 Mb/sec	80 Mb/sec	5 to 10 Mb/sec
Regular data rate used	1380 Kb/sec (352×288)	6500 Kb/sec (720×576)	880 Kb/sec (720×576)
Frames per second (NTSC)	30	30	30
Video quality	Satisfactory	Very good	Good to very good
Encoding hardware requirements	Low	High	Very high
Decoding hardware requirements	Very low	Medium	High

The central concept defined by the video section of the MPEG-4 specification is the video object (VO). It is the building block of the object-based representation. Such representation is appropriate for interactive applications, since it allows direct access to the objects that compose a scene. Video Objects can be natural—textures, image and video—or synthetic—facial animation, body animation and animated 2D or 3D meshes.

A video object can be made up of several layers (VOL), in order to support spatial or temporal scalability. The scalable syntax allows the reconstruction of an object using a layer model, beginning with a base layer and adding enhancement layers. This way, applications generate a single data stream that can be used in different bandwidth and/or computational power conditions.

An MPEG-4 scene can be made up of one or more VOs. Each VO is defined by information about its temporal and spatial features, that is, about shape, movement and texture. In some applications the VO encoding may not be desired, whether by the associate overhead or by the difficulty in creating the objects.

For these cases, the MPEG-4 specification allows the encoding of rectangular images that represent a degenerate case of an arbitrary shape object. The hierarchy description of a scene provided by an MPEG-4 bit is illustrated in Figure 3.

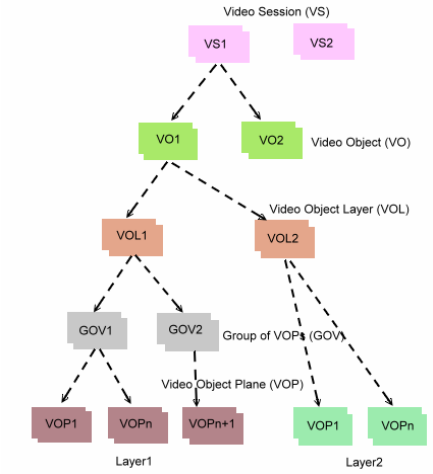

Fig. 3. Logic structure of an MPEG-4 Stream.

The hierarchy levels that directly describe a scene are:

- **Video Object Scene (VS)**: The whole MPEG-4 scene. It may contain 2D, 3D, natural or synthetic objects,
- **Video Object (VO)**: A video object is an object present in the scene. In the simplest case it can be a rectangular image,
- **Video Object Layer (VOL)**: Each object can be encoded in a scalable fashion (multi-layer) or nonscalable (single layer). This scalability can be spatial and/or temporal, thus allowing different resolutions and frame rates,
- **Group of Video Objects (GOV)**: A GOV is a set of VOPs. The GOV start codes mark positions in the bit stream where the VOPs are encoded independently of each other, thus allowing a random access to the bit stream. GOVs are optional,
- **Video Object Plane (VOP)**: A VOP is a sample of a VO. It can be encoded based on other VOPs, using motion compensation (see below).

The MPEG-4 specification defines three VOP encoding modes, as can be seen in Figure 4.

- **I-VOP (intra VOP)** - the encoding is independent of any other VOP,
- **P-VOP (predicted VOP)** - the encoding is based on the nearest past I-VOP,
- **B-VOP (bidirectional VOP)** - the encoding is based on both past and future I-VOPs and P-VOPs[11].

Fig. 4. VOP encoding modes.

2.3. Parallel Programming for MPEG

The parallel encoder developed explores the data parallelism found in the MPEG-4 specification. In order to establish the best data-partitioning model, some approaches where considered:

2.3.1 VO partitioning

As seen in the previous section the MPEG-4 bit stream is composed of independent objects. The first and most natural partitioning approach would be to attribute a different object to each processor [21]. However,

- Different object may have very different computational needs, thus being very difficult to have some balance in the workload of the different processors,
- The maximum number of processors that would be possible to use would be the same as the number of objects, which could be unity.

2.3.2. VOL partitioning

The next in hierarchy level is VOL. Its size and complexity are similar to those of the VO, so the inconvenience presented in the previous section is also valid here.

The encoding of the enhancement layers uses the reconstructed images from lower level layers, so there can be dependencies issues when encoding different VOLs. In order to work around this difficulty, it was decided not to implement multi-layer support. This way it is assumed that the encoding process only generates the base layer.

2.3.3. GOV partitioning

Each VOL is constituted by VOPs that can be grouped in GOVs (group of VOPs). The main characteristic of a GOV is that it starts in an I-VOP, and all the included VOPs are encoded based on others that are also part of the GOV. So, a GOV is a small set of VOPs (typically around 15) that have no external data dependencies.

A GOV based data partitioning will achieve a rather fine granularity, and a consequent load-balancing capability. Another relevant aspect is that the computational needs for successive GOVs are similar, because there is a high probability that the characteristics of an image in a sequence don't vary much with time.

The GOV division is not, however, mandatory in the MPEG-4 specification. Although it could be defined as a requisite for the parallel encoding, a slightly different solution was chosen.

2.3.4. Pseudo-GOV partitioning

From the VOP encoding definitions, it is known that the smallest set of VOPS without external dependencies is defined by the number of frames between two successive IVOPs (intra-period). This is also the size of the smallest GOV that can be defined. The data partitioning solution was built upon a virtual hierarchy entity that was called pseudo-GOV that is a GOV that has the smallest possible size, and only has meaning during the parallel encoding process. This way the work batch contains an integer number of pseudo-GOVs.

Figure 5 shows the data dependencies for each VOP type and the proposed data distribution. In this the batch contains a single pseudo-GOV, but the model is valid for any integer number of pseudo-GOVS.[11]

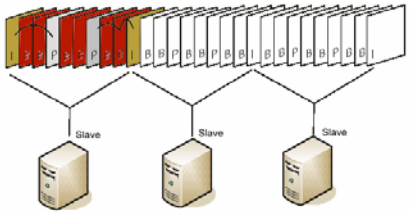

Fig. 5. VOP sequence and its distribution.

The first VOP of each pseudo-GOV is the same as the last VOP of the preceding pseudo-GOV. This is required because each I-VOP is the reference for the encoding of BVOPs on the previous pseudo-GOV and both P-VOPs and B-VOPs on the next pseudo-GOV.

3. Scheduling Approaches

In order to achieve an efficient parallelization, four scheduling algorithms were considered, namely Round-Robin, Adapted Batch Size Round Robin, Dynamic Scheduling and Adapted Batch Size Dynamic Scheduling. The algorithms are discussed with detail in next subsections.

3.1. Round-Robin

The first one used was the round robin scheduling. It is the usual starting point when developing scheduling algorithms. An equal number of work batches are sent to each slave and the processed data blocks are then received in the same order as they were sent. This way the master can become blocked waiting for a slave to finish its work and return the processed data, as can be seen in Figure 6.

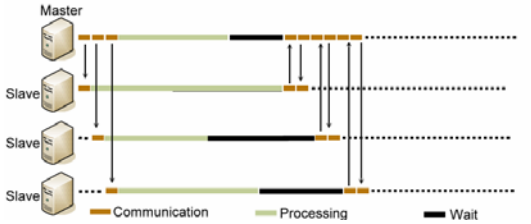

Fig. 6. Activity diagram for the Round-Robin scheduling.

In this example the workload/capacity of the computers is not uniform. If one of the computers takes more time than the others to process its data, it will cause time intervals when some computers are idle.

3.2. Adapted Batch Size Round-Robin

This is used to solve the problem of extra standby period caused by the Slaves due to their different processing time while performing Round-Robin test. This method is mainly used to guarantee that each Slave has same material processing time. In order to achieve the goal, the computation ability of each Slave must be pre-measured, after that the data material can be assigned to each Slave for data processing. Therefore, in the beginning, the Master assigns the same small amount material to each Slave for surveying and recording the computation time. Afterward, by the estimating result, the Master can distribute different suitable material size to each Slave. Comparing with Round-Robin, in the environment of Adapted Batch Size Round Robin, each Slave computation time is approximately same and the standby period of each Slave is also shortened. However, the server platoon regulation method is still Round-Robin, and each Slave still need massive time in waiting feedback and receiving material. Moreover, the Master server engine also needs extra computing time to figure out the computation ability of other server engines.

3.3. Dynamic Scheduling

This is used to solve the problem of the standby period while performing Round-Robin test in order. The procedures are shown in Figure 7 as follows:

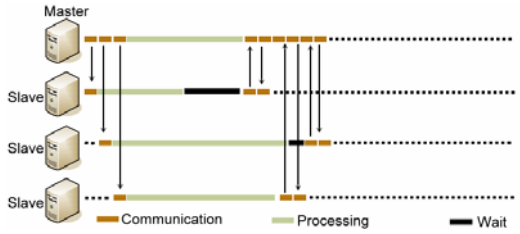

Fig. 7. Activity diagram for the Dynamic Scheduling

From this chart we can see that the Master no longer waits for the slave in order, but decide to

receive the feedback material when the Slave is able to complete the computation. Comparing with the Round-Robin method, dynamic scheduling has omitted big share of standby period. Because Master does not receive material in order, therefore preparation of an extra buffer is necessary for depositing the material in order to re-organize the computed result. To avoid the buffer oversized, Master must try to receive the feedback material in order occasionally.

3.4. Adapted Batch Size Dynamic Scheduling

This method is the combination of Adapted Batch Size Round-Robin and Dynamic Scheduling algorithm in order to reduce the waiting time that the Master spends to receive the feedback from the Slaves after finishing its own task. Such procedure in theoretically indeed is able to shorten the waiting time of the various server engines to the lowest. However, the idea cannot be achieved in reality resulted from that the Master is not able to calculate the computing time of each server engines precisely [11].

3.5. Parallel Video Transcoding (transcode)

The transcode is a Linux text-console utility for video stream processing, running on a platform that supports shared libraries and threads. Decoding and encoding were done by loading modules that are responsible for feeding transcode with raw video/audio streams (import modules) and encoding the frames (export modules). It supports elementary video and audio frame transformations, including de-interlacing or fast resizing of video frames and loading of external filters.

A number of modules are included to enable import of DVDs on-the-fly, MPEG elementary (ES) or program streams (VOB), MPEG video, Digital Video (DV), YUV4MPEG streams, Nuppel Video file format, AVI based codecs and raw or compressed (pass-through) video frames and export modules for writing DivX;-), XviD, DivX 4.xx/5.xx or uncompressed AVI and raw files with MPEG, AC3 (pass-through) or PCM audio. Additional export modules to write single frames (PPM) or YUV4MPEG streams are available, as well as an interface import module to the avi file library. It is modular concept intended to provide flexibility and easy user extensibility to include other video/audio codecs or file types. A set of tools are included to demux (tcdemux), extract (tcextract) and decode (tcdecode), while the sources into raw video/audio streams for import, probing (tcprobe) and scanning (tcscan) your sources. To enable post-processing of AVI files, fixing AVI file header information (avifix), merging multiple files (avimerge), splitting large AVI files (avisplit) to fit on a CD and avi sync to correct AV-offsyncs [19].

4. Experiments

4.1. Experimental Cluster Platform

We build a cluster computing testbed as shown in Figure 8 includes four Linux (Fedora Core 3) PC nodes:

- Master (FC3-01): Single Celeron 2000 processor, 512MB DDRAM and 3Com 3c905 1 interface.
- Client 1 (FC3-02): Single Celeron 2000 processor, 384MB DDRAM and 3Com 3c905 1 interface.
- Client 2 (FC3-03): Single Pentium 3 866 processor, 256MB SDRAM and 3Com 3c905 1 interface.
- Client 3 (FC3-04): Single Pentium 3 866 processor, 256MB SDRAM and 3Com 3c905 1 interface.

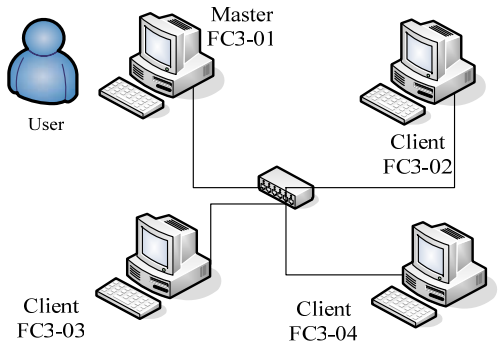

Fig. 8. Experimental Cluster computing testbed.

4.2. Software for Video Compression

This section gives an introduction of DVD to DivX compression in video conversion cluster. The related software installation on master and client nodes is listed in Table 2. As shown in Figure 9, the first step is to split VOB files into a number of chunks according to the number of nodes that the video conversion cluster system has. The sizes of divided files are based on their information available in dvd::rip (see Figure 10).

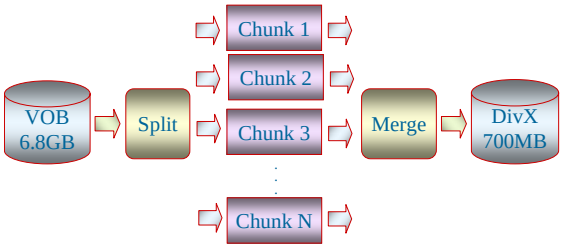

Fig. 9. Split and merge the video files.

By dvd::rip, a cluster consists of the following components:

1. A computer with a full dvd::rip and transcode installation, DVD access and local storage or access to a NFS server, where all files are stored.
2. A computer with a dvd::rip installation, but no GUI access and no transcode installation, where the cluster control daemon runs on. This may be the same computer as noted less than 1 (which is usually the case).
3. An arbitrary number of computers with a full transcode installation, dvd::rip are not necessary here. These are the transcode nodes of the cluster.
4. The GUI dvd::rip computer and the transcode nodes must all have access to the project directory, shared via NFS or something similar. It doesn't make any difference which computer on the network is the NFS server.
5. The communication between the cluster control daemon and the transcode nodes is done via SSH. All transcode commands are calculated by the cluster control daemon and executed via SSH on the transcode nodes. Dvd::rip assumes that the cluster control computer has user key authentication based access to the nodes. That means that no password needs to be given interactively.

Table 2. The related software installation on our cluster computing testbed**.**

Master	aalib	divx41linux	faac
	ffmpeg	fping	Gtk-Perl
	lame	libdvbpsi	libdvdcss
	libdvdplay	libdvdread	libfame
	libmad	libmpeg3	libpostproc
	lirc	lzo	mjpegtools
	mpg321	mplayer	ogle
	perl-libintl	rar	subtitlerippe r
	a52dec	faad2	imlib2
	libdvdnav	libid3tag	libquicktime
	mpeg2dec	transcode	xine
	unrar	vcdimager	vobcopy
	xvidcore	perl-Video-DVDRip	
Cluster	transcode		

This may look confusing, but in fact all the different services described here can be distributed in arbitrary ways on your hardware. You can even use the cluster mode with one computer, which runs all services: dvd::rip GUI, cluster control daemon, transcode node (naturally using local data access), as in Figure 11. In this case you may “misuse” the cluster mode as a comfortable job controller, which is in fact a regular use case, because dvd::rip has no specific job features besides this.

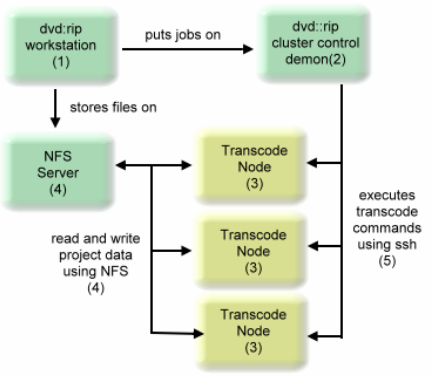

Fig. 10. Components of dvd::rip cluster.

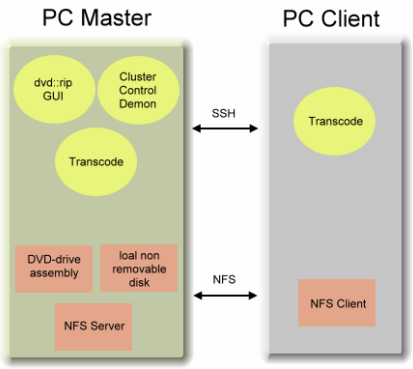

Fig. 11. System components.

A typical two-node installation may look like this: Master computer runs services:

- dvd:rip GUI
- dvd::rip cluster control daemon
- transcode node with local storage access
- NFS server

Client computer runs services:

- Transcode node with NFS access to the project data [2],

4.2.1. The dvd::rip Cluster Project Job

The job queue shows all tasks which must be completed as shown in Figure 12. Mainly the work is divided into four phases:

1. Transcode video: as many nodes as possible will be used in parallel for this phase. They will transcode different chunks of the video from MPEG to AVI, but without audio,
2. Transcode audio: due to technical reasons audio has to be transcoded independent from the video and it's not possible to break up the job into chunks which can be processed in parallel. If you selected more than one audio track, an appropriate number of audio transcoding jobs will appear,

3. Merge video + audio: The transcoded audio file of the first selected audio track and all video chunks are merged and multiplexed into one file. This is done preferably on the node with local hard disk access,
4. Split: If you decided to split the AVI afterwards, this is the final phase [2].

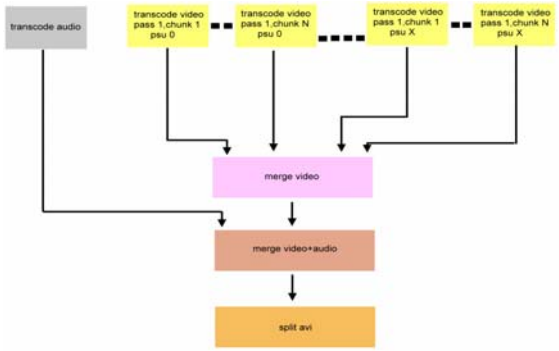

Fig. 12. Complete workflow one dvd::rip cluster project job.

4.3. Testing Model

In this experiment environment, we utilize different three testing models to shorten the waiting time between nodes, let execution time contract to get shortest. Let each node all to display the biggest operation potency. We use "Dynamic Scheduling" of Scheduling approach methods to plan our test model.

4.3.1. Static Processing Potency Model

In this testing model we calculate the chunks between each node the ratio according to /Proc/ in cpuinfo bogomips data in each node. The complete frames will sliver each ratio assembling. In this experiment environment we can get everyone node cpuinfo bogomips data x_1, x_2, x_3, x_4; x_1 is minimum value.

x_1 = 1695.74 for FC3-03 node.
x_2 = 1699.84 for FC3-04 node.
x_3 = 3948.54 for FC3-01 node.
x_4 = 3973.12 for FC3-02 node.

Calculates each between the proportions y_i as shown in Table 3:

$y_i = x_i / x_1$ for each i=1, 2, 3, 4.

Table 3. For each y_i value.

y_i	y_1	y_2	y_3	y_4
	1.00	1.00	2.32	2.34

The computation cuts the Frame chunks z_j as shown in Table 4:

$$z_j = \sum_{i=1}^{4} y_i * j \quad \text{for each } j=1,2..,10.$$

Table 4. For each z_j value.

z_j	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}
	6	14	20	26	34	40	46	54	60	66

4.3.2. Processor Number Model

In this method the data material of the video frames intended for conversion is divided into the same quantities as the PCs without considering the different computing ability among the individual PCs. In this experimental environment, we use 4 nodes and therefore the data material was divided into 4 portions.

4.3.3. Dynamic Processing Potency Model

We first will test the material to make the operation to each node, and takes it actually operates the potency, again depends on the proportion to cut a frames. a_1, a_2, a_3, a_4 is actually operates the potency, a_1 is minimum value.

a_1 = 6.9fps for FC3-04 node.
a_2 = 7.0fps for FC3-03 node.
a_3 = 8.8fps for FC3-01 node.
a_4 = 9.0fps for FC3-02 node.

Calculates each between the proportions *bi* as shown in Table 5.

$b_i = \frac{a_i}{a_1}$ for each i=1, 2 , 3, 4.

Table 5. For each b_i value.

b_i	b_1	b_2	b_3	b_4
	1.00	1.01	1.27	1.30

The computation cuts the Frame chunks c_j as shown in Table 5.

$$c_j = \sum_{i=1}^{4} b_i * j \quad \text{for each } j=1, 2 .., 10.$$

Table 6. For each c_j value.

c_j	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
	4	10	14	18	23	28	32	36	41	45

5. Performance Results

We choose two DVD movies with different content and capacity as the source data for testing. Following table (Table 7) shows the detailed information of the source data.

Table 7. Source data of DVD.

Case	Video Length	Video Type	Ratio	Video Frames	Resolution	Video Size
Case 1	01:38:48	NTSC	16:9	143364	720x480	3.2 GB
Case 2	01:48:37	NTSC	16:9	197041	720x480	6.6 GB

Transcode Data:

- Format: Divx4; Video code: xvid.
- Target Size:1400M, 700M, 400M.

5.1. Static Processing Potency Model

We use aforementioned test model to measure the processing time and find it takes less in *j*. By this experiment we know when *j*=2, Case 1 and Case 2 both achieve the best performance as shown in Figures 13, 14 and 15.

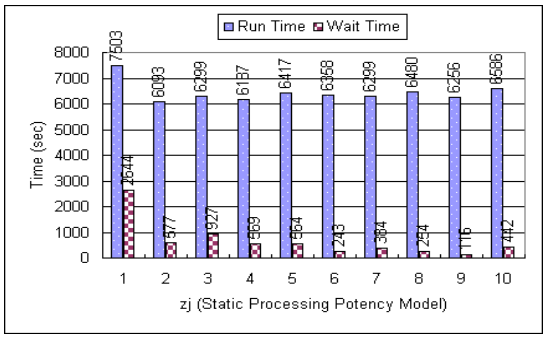

Fig. 13. Static Processing Time with Case 1.

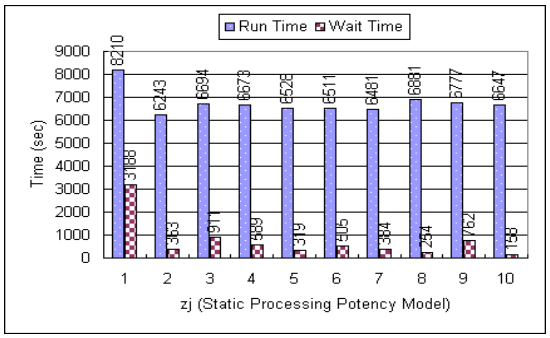

Fig. 14. Static Processing Time with Case 2.

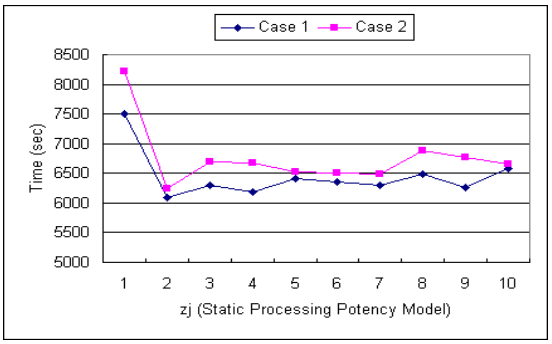

Fig. 15. Static Processing Time with Case 1 and Case 2.

The parallel performance ratio in each j can be achieved by 80% in average as shown in Figure 16.

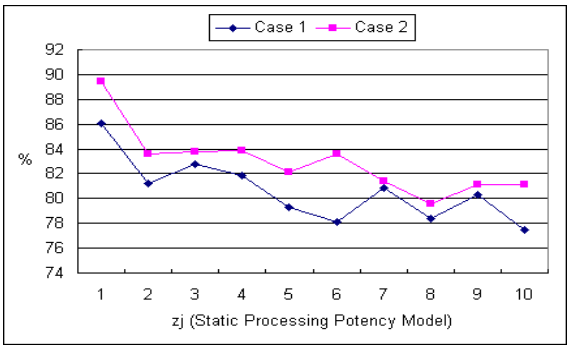

Fig. 16. Static Processing Parallel Performance Ratio with Case 1 and Case 2.

5.2. Dynamic Processing Potency Model

We use aforementioned test model to measure the processing time and found it takes less in *j*. By this experiment we know when *j*=3, case 1 and case 2 achieve the best performance as shown in Figures 17, 18 and 19.

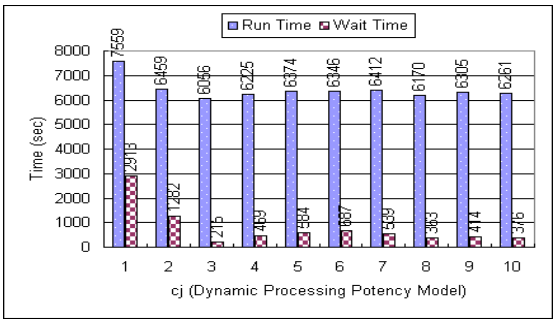

Fig. 17. Dynamic Processing Time with Case 1.

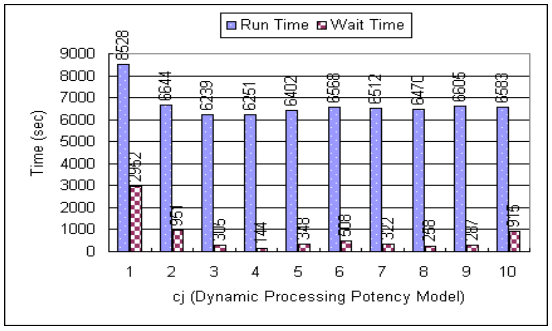

Fig. 18. Dynamic Processing Time with Case 2.

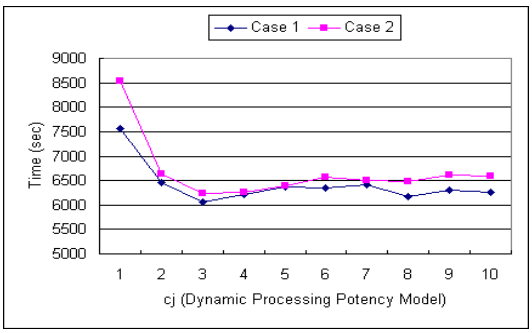

Fig. 19. Dynamic Processing Time with Case 1 and Case 2.

The parallel performance ratio in each j can be achieved by 80% in average as shown in Figure 20.

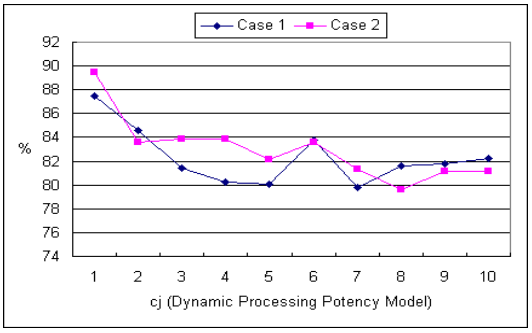

Fig. 20. Dynamic Processing Parallel Performance Ratio with Case 1 and Case 2.

5.3. Different Target Size

In here we compare video conversion time on different target size. We use test model "Dynamic Processing Potency Model", to do comparison processing time. By this experiment we know when target size is reduced the video conversion time is reduced accordingly. The results are shown in Figures 21 and 22.

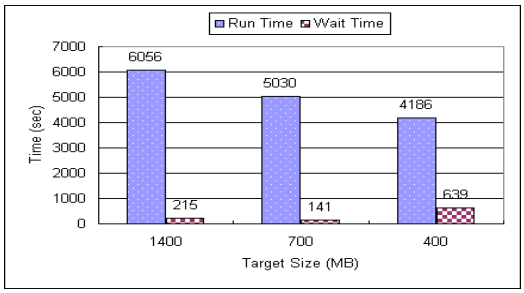

Fig. 21. Comparison of conversion time on different target size by Case 1.

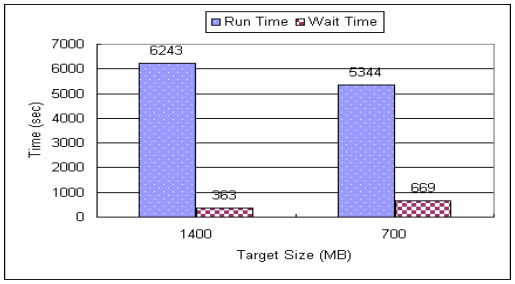

Fig. 22. Comparison of conversion time on different target size by Case 2.

5.4. Comparison between Single PC and Cluster system

Here we compare video conversion time from DVD to Divx4 of single PC and cluster system. Obviously the answer is positive as shown in Figure 23.

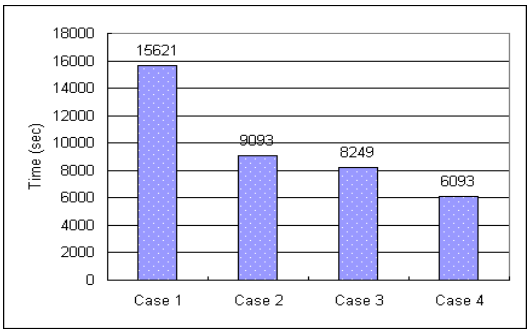

Fig. 23. The comparison of conversion time from DVD to Divx4 of single PC and cluster system.

6. Conclusions

Nowadays, single PC has processed mostly video conversion, which not only takes long time but also wastes the waiting period. Better performance can be achieved with powerful hardware, although the cost is higher. Using cluster system can certainly satisfy both.

In this paper, we use dvd::rip to assist in job assignment to nodes, and hope that cluster platform is capable to shorten the video conversion time and meanwhile obtain higher efficiency.

In this experimental research, we utilize different types of cluster systems, and therefore, the computing ability of each individual node must be considered when assigning job. Here we use "Dynamic Scheduling" of scheduling approaches methods to plan our test model.

In the experiment of "Static Processing Potency Model", when j=3, we obtain the best conversion time. The result is certain and proved by using two video movies, cases 1and 2, as shown in previous sections.

In the experiment of "Dynamic Processing Potency Model", when j=2, we obtain the best conversion time. We have used two additional video movies, cases1 and 2, to prove the result. It also shows the best performance when z_j and cj=14.

We also obtain the comparison result between traditional cluster and diskless cluster that both are similar in computing capability. However, it is more convenient and efficient to organize and administrate in the environment of diskless cluster system.

We are planning to use this testing model to link a lager cluster system, to see if we can obtain and prove that the best performance of conversion time also falls on j.

As future work, we will work on transforming this entire system into diskless system, to evaluate if these changes affect the computing ability of the overall system.

References

[1] B. Wilkinson and M. Allen, "Parallel Programming: Techniques and Applications Using Networked

Workstations and Parallel Computers", Prentice Hall PTR, NJ, 1999.
[2] "dvd::rip", http://www.exit1.org/dvdrip/.
[3] "Grid Computing", http://www.globus.org.
[4] Gunawan, T.S.; Cai Wen Tong, "Parallel motion estimation on SMP system and cluster of SMPs", 2002. APCCAS '02. 2002 Asia-Pacific Conference on Circuits and Systems, Volume: 2, 28-31 Oct. 2002.
[5] ISO/IEC, "MPEG-4 Overview -(Melbourne Version)", JTC1/SC29/WG11 N2995,Oct 1999.
[6] ISO/IEC, "Information Technology-Generic Coding of Audio-Visual Objects, Final Draft of International Standard", JTC1/SC29/WG11 N2502, Oct 1998
[7] **"**LAM/MPI Parallel Computing", http://www.lam-mpi.org.
[8] Lee, J.Y.B., "Parallel video servers: a tutorial", IEEE Multimedia, Volume: 5 Issue: 2, April-June 1998.
[9] "MPEG-4 - Copying a DVD Video to CD-ROM", http://www.tomshardware.com/video/20000913/.
[10] M. Wolfe, "High-Performance Compilers for Parallel Computing", Addison-Wesley Publishing,NY, 1996.
[11] Po-Kai Chiu., "Design and Implement a MP4 Video-on-Demand System based on PC Cluster", Volume: 19-24,2003.
[12] "PVM", http://www.epm.ornl.gov/pvm/pvm_home.html/.
[13] R. Buyya, "High Performance Cluster Computing: System and Architectures", Volume: 1, Prentice Hall PTR, NJ, 1999.
[14] R. Buyya, "High Performance Cluster Computing: Programming and Applications", Volume: 2, Prentice Hall PTR, NJ, 1999.
[15] Richard S. Morrison, "Cluster Computing", Revision Version 2.1, Saturday, 26 January 2002.
[16] "The MPEG home page", http://drogo.cselt.stet.it/mpeg/.
[17] Tierney, B.L.; Johnston, W.E.; Herzog, H.; Hoo, G.; Guojun Jin; Lee, J.; Chen, L.T.; Rotem, D., "Using high speed networks to enable distributed parallel image server systems", Supercomputing '94. Proceedings, Volume: 14-18, Nov. 1994.
[18] Touradj Ebrahimi, Caspar Home, "MPEG-4 natural video coding-An overview",Image Communication Journal 1999.
[19] "Transcode", http://zebra.fh-weingarten.de/~transcode/.
[20] Yong He; Ahmad, T.; Liou, M.L., "MPEG-4 based interactive video using parallel processing", 1998. Proceedings. 1998 International Conference on Parallel Processing, Volume: 10-14, Aug. 1998.
[21] Yong He; Ahmad, T.; Liou, M.L., "Real-Time Interactive MPEG-4 System Encoder Using a cluster of Workstations", IEEE Transactions on Multimedia, Volume: 1 Issue: 2, June 1999.
[22] Young He, Ishfaq Ahmad and Ming L. Liou, "Real-Time Interactive MPEG-4 System Encoder Using a cluster of Workstations", IEEE Transactions on Multimedia, Volume: 1 Issue: 2, June 1999.