RSVP: Ridiculously Scalable Video Playback on Clustered Tiled Displays

Jason Kimball*, Kevin Ponto†, Tom Wypych*, Falko Kuester‡
*Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA USA
Email: jkimball@ucsd.edu, twypych@ucsd.edu
†Wisconsin Institute for Discovery
University of Wisconsin, Madison
Madison, WI USA
Email: kbponto@wisc.edu
‡Department of Structural Engineering
University of California, San Diego
San Diego, CA USA
Email: fkuester@ucsd.edu

***Abstract*—This paper introduces a distributed approach for playback of video content at resolutions of 4K (digital cinema) and well beyond. This approach is designed for scalable, high-resolution, multi-tile display environments, which are controlled by a cluster of machines, with each node driving one or multiple displays. A preparatory tiling pass separates the original video into a user definable n-by-m array of equally sized video tiles, each of which is individually compressed. By only reading and rendering the video tiles that correspond to a given node's viewpoint, the computation power required for video playback can be distributed over multiple machines, resulting in a highly scalable video playback system. This approach exploits the computational parallelism of the display cluster while only using minimal network resources in order to maintain software-level synchronization of the video playback. While network constraints limit the maximum resolution of other high-resolution video playback approaches, this algorithm is able to scale to video at resolutions of tens of millions of pixels and beyond. Furthermore the system allows for flexible control of the video characteristics, allowing content to be interactively reorganized while maintaining smooth playback. This approach scales well for concurrent playback of multiple videos and does not require any specialized video decoding hardware to achieve ultra-high resolution video playback.**

***Keywords*-tiled video; tiled display walls; high resolution video; video playback;**

I. Introduction

Tiled display environments offer a superb workspace for ultra-high-resolution media content. These kinds of systems offer not only a large amount of pixel real-estate, but also an environment for distributed computation. As thin-bezel LCD displays and desktop PC hardware have decreased in cost, the world has seen a proliferation of these kind of systems, as seen in the OptiPuter project [1]. Unfortunately, many approaches to playing video content on these types of display systems does not scale well beyond HD resolution video. The current software systems to deliver this video are currently lacking in capability due to a number of factors including network bandwidth limitations and CPU processing power.

CineGrid has presented motivations and techniques for 4k video playback aimed at creating and distributing this high-resolution media from different locations all across the globe [2]. However this paper by Herr shows that streaming of video beyond 4k resolution to display nodes requires significant network resources and exceeds the current limitations of the CineGrid approach. In this regard, the playback technology currently lags behind imaging technology, as cameras which support 8k resolution video have already been successfully prototyped [3]. The high bit-rate intrinsic to video restricts the resolution of video delivery in uncompressed, streaming-based tiled visualization approaches.

Pre-compressed video reduces the network limitations for reading video content, but the challenge for playing back ultra-high-resolution compressed media comes from the CPU work required to decode the video and then upload it to the graphics card for display. The associated cost function can be formulated as:

$$T_{Frame} = T_{Read} + T_{Decode} + T_{Upload} \quad (1)$$

Each of these time costs (T) are a function of the attributes of the video and the performance characteristics of the playback system, defined by the speeds of the CPU, GPU, etc. For the purposes of this paper, we take the system performance as a constant and focus instead on distributing workload across the entire tiled display system to reduce the time it takes to process each frame.

While distributed decoding approaches such as Chen [4] can leverage the distributed processing power of display clusters to decode higher resolution videos, the encapsulation overhead, node-to-node communication requirements,

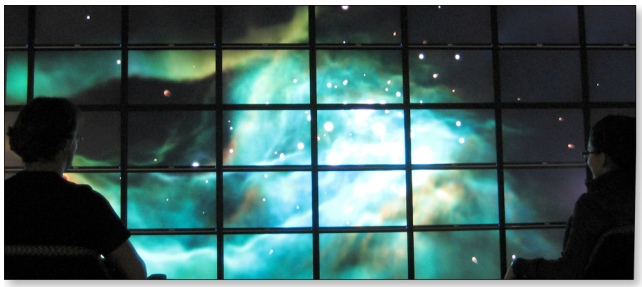

Figure 1. Users viewing a 20 megapixel video on a tiled display system.

and synchronization challenges limit the scalability when using a large number of display nodes.

To alleviate both the network limitations and CPU decoding performance issues, we propose a video tiling system which splits the source video into an n-by-m grid of equally sized video tiles, each of which is independently compressed and written to file. Each display node can read and decode the subset of video tiles in its view and ignore the rest, and multiple video tiles on a display node can be decoded in parallel. This allows video playback to scale to very high resolutions, well beyond 4k, without incurring any additional overhead dependent on the resolution of the videos or requiring extra communication between display nodes.

While tiled video approaches have been criticized as naive methods because they reduce motion prediction efficiency and incur a high time penalty for preprocessing [5], in this paper we refute these criticisms. We are able to demonstrate that tiling does not significantly impact the quality of the encoded videos and the additional overhead created by tiling videos is significantly less than the overhead incurred by non-tiled parallel decoding approaches. Furthermore we describe a tiled encoder and demonstrate that the encoding time to create tiled videos is not significantly more than the time to create a single non-tiled video.

The rest of this paper is as follows: we discuss previous high-resolution video playback approaches and their limitations, describe the implementation of a tiled video playback system, discuss tiled video creation, and give results for ultra-high-resolution video playback on a tiled display wall.

II. Related Works

A. Specialized Hardware

A standard approach to playing 4k content requires segmentation of each frame into four 1080p subcomponents [6]. Each of these sub-components is encoded using a JPEG2000 encoder and stored on a RAID array for rapid access. When the video is played, media content is streamed via a gigabit network interface to a render node with a JPEG2000 decoder card. While this system works well for 4k data, it does not scale, as Shirai et al. restrict the maximum output from their system to 3,840x2,160 [6]. For this system to work on a tiled display environment, each render node would need a JPEG2000 decoding card, in combination with a substantial amount of network resources, as JPEG2000 data needs to be streamed into each decoder card, thus imposing network scaling limitations. Additionally, no functionality exists for changing video location within a tiled display workspace, as decoding and display is fixed to each node's decoder card in the geometry that the JPEG2000 segments were encoded.

B. Uncompressed Pixel Streaming

Another technique to deliver multimedia content on tiled display systems is based on pixel streaming, as implemented by the Scalable Adaptive Graphics Environment (SAGE) [7], [8], and [9]. In this approach, a single system decodes and renders video content into a buffer which is subsequently mapped to the tiled display environment. This buffer is segmented such that it considers the viewpoints of each node in the tiled display system. The pixel information for each display is then streamed out via the network.

In the cost function illustrated in equation 1, the streaming of uncompressed pixel data adds a new term, $T_{Transport}$, for cost of transporting pixels over the network, as shown in equation 2. While T_{Upload} can be reduced as each display node only receives and uploads the part of the video it needs, since SAGE does not address parallel video decoding, T_{Read} and T_{Decode} are not reduced, which can limit the total resolution of video that can be decoded. Furthermore the network transport of the video pixels requires a network bisection bandwidth proportional to the total resolution of the video and the frame rate. This can also be a limiting factor in scalability.

$$T_{Frame} = T_{Read} + T_{Decode} + T_{Transport} + T_{Upload} \quad (2)$$

C. Synchronous Distributed Decoding

In an attempt to remove the network constrained performance limitations, approaches such as Choe [10] and Ponto [11] distributes compressed video to each node in the cluster. In this approach, a single video file containing the video content is replicated to each node, allowing each video file to be decoded concurrently on each node. This approach allows for video playback with minimal network bandwidth, empowering real-time interaction and filtering.

In the cost paradigm of equation (1), these approaches offer no advantage for a tiled display system compared to a conventional playback system, as each node must read, decode, and upload each video frame in its entirety. As a result, ultra-high-resolution video playback is not possible in these architectures, as they do not address the challenges of decoding video at or beyond 4k resolution.

D. Macro-Block Forwarding

To address the computational bottleneck of decoding a 4k resolution video on a single node, Chen in papers [4]

[12] and [13] present a clever method for distributing the decoding of 4k content on tiled display environments by utilizing key features of the MPEG2 codec which allows video frames to be segmented into smaller sections for decoding. The described system employs a layer of redirection nodes which interface the head node and the render nodes driving the tiled display environment. These nodes extract individual MPEG2 encoded macro-block data and distribute them to the required rendering nodes to decode and display.

In this way, the cost, T_{Decode}, is distributed throughout the entire tiled display environment and T_{Upload} is reduced as in the streaming approach described above.

While this approach demonstrates decoding scalability, it also imposes a number of restrictions. First and foremost, videos must be encoded in the MPEG2 format. The MPEG2 codec is a somewhat antiquated video codec, partially for the reasons which Chen is able to exploit in [4] and [13], and new codecs have been developed which can produce much higher quality results at much lower bitrates. Furthermore, the MPEG2 video format, by specification, can not have frame sizes greater than 1,920x1,152 [14]. This means that encoding videos of greater resolution must be done via custom software and can not be done using common MPEG2 encoders.

Additionally, this approach requires a second level of nodes between the head node and render nodes in order to negotiate macro-block forwarding. These routing nodes must receive and resend information, including header data which incurs an additional 20% bandwidth cost [4]. The published testing results demonstrate a decrease in scaling performance as more decoding nodes were used; this behavior is attributed to increasing dependency on inter-node communication, which imposes a limit on total scalability.

E. Tiling Video

In an effort to facilitate collaborative visualization, Renambot et. al introduce Sage Bridge [15], which segments SAGE visualization streams into blocks for easier transmission and display on multiple display walls concurrently. While SAGE works by streaming exact pixel geometry to each node, on heterogenous display environments this means that the pixel geometry must be recalculated and the source imagery must be streamed separately for each display wall. Sage Bridge acts as an intermediary and segments the pixel geometry into smaller blocks and streams these blocks to the appropriate display nodes. Each display receives the subset of blocks that it needs to display its pixel geometry, and truncates any excess pixels beyond the border of display. By using small enough blocks, the excess pixels do not overburden network or CPU resources for the display nodes; however, video decoding performance is still bounded by the computational capabilities of the single decoding node.

The approach presented in this paper is inspired by a similar idea: by breaking down a high resolution video into smaller blocks, the algorithm allows each node to cull the non-displayed blocks and only spend network and cpu resources on the blocks it needs for display. However, because the solution uses compressed video instead of uncompressed pixels, it is able to handle a significantly higher resolution video source without exhausting network bandwidth resources in transmission of the video to the display nodes. We contend that this approach adequately avoids the immediate performance limitations of both network and processing requirements associated with existing systems.

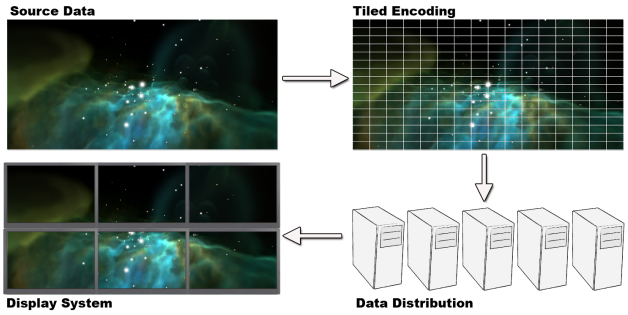

Figure 2. Overview of the process for the scalable multimedia system derived in this paper.

III. System Overview

The goal of this approach is to improve the scalability of video playback resolution for a given system by parallelizing the workload of video decoding and display across multiple machines. To accomplish this, a three step process is performed, as shown in Figure 2. First the original video is separated into a user definable n-by-m array of equally sized video tiles, each of which is individually compressed. This segmentation allows each rendering node to read, decode, and upload subsections of the original video frame as opposed to the entire frame as seen in other approaches like [11]. Next this tiled video content is distributed to the rendering nodes either through pre-distribution to local disks or through a network mounted file system. Finally, rendering nodes are tasked with playing the video tiles corresponding to their given viewpoint.

The video system provides the user with the ability to display one or more tiled videos on a tiled display system. The videos can be loaded and unloaded at will. Once loaded, each video or group can be resized and moved to any location on the display wall, or can be rendered as a part of any 2D or 3D scene, as a 2D texture applied to arbitrary surface geometry. The user can skip forward or backwards to any point in the video.

IV. Preprocessing Implementation/Performance

A. Content Preparation (Tiling)

As demonstrated by tiled display image viewers, the tiling of content is a practical method for distributing work loads

across tiled display systems [16] [17]. While it is possible to create video tiles using existing software by simply changing input cropping parameters, this approach may be inefficient as the time required to complete the encoding process increases linearly with the number of video tiles.

With low-latency encoding in mind, supporting near-realtime processing and delivery of ultra-high resolution video content, a custom video encoder application was specifically developed for generating tiled videos. This tiled-encoder creates a video encoding context for each of the video tiles. From the input stream, the tiled-encoder reads and decodes each input frame and passes it to each of the tiled-encoders. Each of these processes subsequently uses pointer-based operations to determine the corresponding region which to encode from the input frame, removing the need for unnecessary data replication. The advantage of this approach is that each input frame is only read and decoded once, regardless of the number of output video tiles generated. As demonstrated in the results section, reading the source image is a significant portion of video encoding, and removing that redundancy significantly improve speed of the video tiling process. For this reason, the encoding of tiles using this method adds minimal overhead compared to encoding a video without tiling.

B. Tile Creation Costs

Chen also dismisses the idea of tiling video data a priori for reasons of time and computational cost, stating video tiling incurs a "tremendous amount of offline computation" [5]. As the approach of Chen only works for MPEG2 video streams, any other input (via an image sequence, online stream or video encoded in a different codec) would need to be re-encoded before it could be used on their system. As mentioned above, frame sizes which are greater than HD reside outside the MPEG2 specification [14] meaning that standard capture devices are unlikely to encode data in the MPEG2 format. As a result, re-encoding most forms of input data would likely be necessary.

It is therefore important to compare the time required to encode a single video to the time required to encode tiled content. A brute-force approach for tiled video encoding entails running a video encoder for each of the video tilings, each time focusing on a different region of the original input. Given this approach, one could propose that if the major bottleneck comes from the encoding of video data, encoding N number of tiles would take approximately N times longer than encoding a single movie.

As shown in Figure 3, the tiled encoding paradigm described in Section IV-A required very little additional time for extra tilings. In fact, the MPEG2 encoding for an 8x8 tiled output took less time to create than that of its non-tiled counterpart. Furthermore, the encoding of 256 individual videos (16x16) for each of the codecs took less than 50% more time to that of its non-tiled counterpart. Consequently,

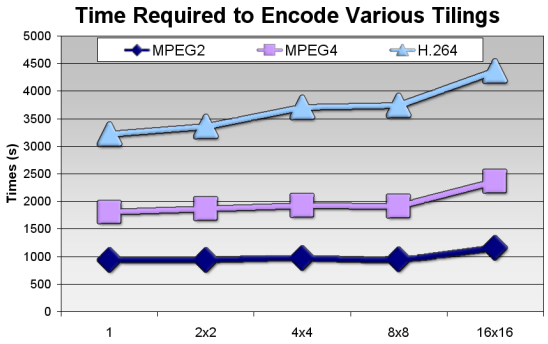

Figure 3. The total time required to encode various video tilings in our tiled encoding application.

given input data which is not already in the MPEG2 format, creating a tiled video does not take significantly more time than creating the MPEG2 file needed for the approach proposed by Chen.

The tiled encoding framework could be further optimized by encoding multiple tiles simultaneously. As the current version encodes the video tiles serially after a frame is read from disk, this could result in a substantial improvement in encoding time of tiled videos for multi-core systems, given a disk optimized for parallel read/write operations. As standard disks were used, disk I/O contributed a major bottleneck (approximately 700 ms per read independent of encoding and tiling), this optimization has not yet been implemented.

C. PSNR Quality Difference

Chen rejects the idea of pre-tiling video, stating "The re-encoding process introduces additional quantization error and limits the ranges of motion vectors, thus reducing the video quality" [5]. To test this assertion we compare the peak-signal-to-noise-ratio (PSNR) of the various tiled video encodings against the original input files.

We selected two different test video segments, a 4k video from the 70-mm film Baraka [18] and a 20 megapixel video from the Orion Nebula Visualization [19]. The scene from Baraka is a slow walk-through of a room filled with crystal detailing. It provides a lot of dynamic detail which changes from frame to frame due to the specular highlights from the crystals. This provides a challenge for the encoding as many motion vectors are poorly matched between frames. The scene from Orion is a much smoother and more consistent scene, with objects slowly flying through space.

Both scenes were encoded using H.264 as a full-size video and as 2-by-2, 4-by-4, and 8-by-8 tiles. A constant quantization value of 19 was used to give similar encodings, with only the performance of the motion vector prediction varying. Adaptive I-Frame and B/P-Frame generation was turned off, so that the frame type of both the non-tiled video and all of the different tiled video configurations was

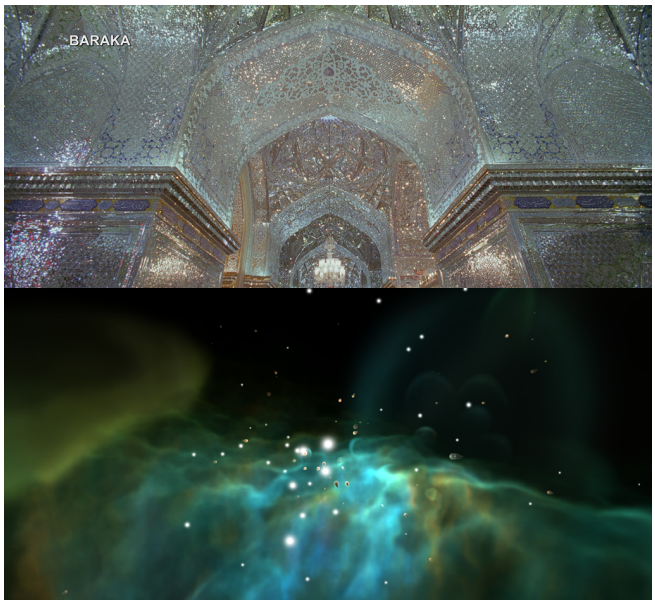

Figure 4. Baraka (top) and Orion (bottom) sample videos.

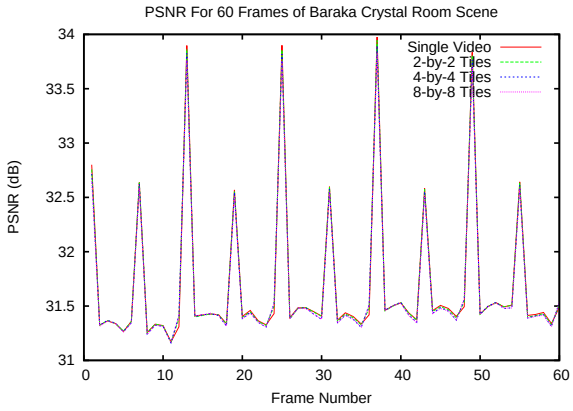

Figure 5. PSNR for Baraka Test Video.

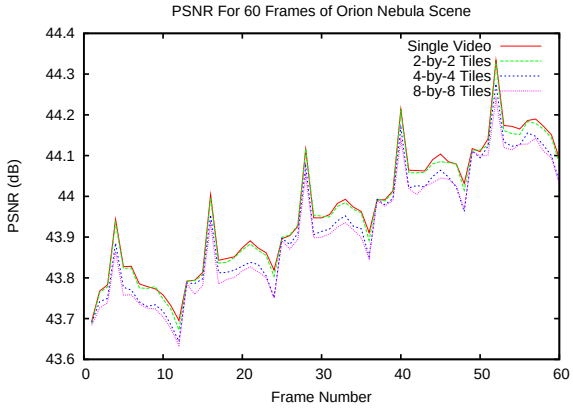

Figure 6. PSNR for Orion Test Video.

consistent, allowing a fair comparison of per-frame encoding error. The PSNR (peak signal-to-noise ratio) of each encoded frame was computed in relation to the original source image frames.

The results for Baraka 5 and Orion 6 both show that the PSNR is extremely close for the original and all of the different tile configurations. While in general the PSNR was slightly better for the non-tiled video, the differences are extremely small. Furthermore, the file size increases for tiling the video are also small. Figure 8 shows a comparison.

An additional encoding of the Baraka 2-by-2, 4-by-4, and 8-by-8 tiles were made at a quantization value of 18 to compare the additional file size gained while increasing the quality of the tiled videos to beyond that of the non-tiled video. Figure 7 compares the PSNR values of these tiled videos to the non-tiled video with a quantization value of 19. The file size difference for both quantization values are shown in Figure 9 and the percentage increased values are in relation to the non-tiled video encoded with a quantization value of 19.

V. Distributed Playback

A. Data Distribution

Regardless of the selected tiled or non-tiled format, the video has to be made accessible to all of the playback nodes. This can be done by replicating data on the machines locally, or preferably by providing fast network-centric data access, for example via a network mounted file system. While pre-distribution of content yields lower network bandwidth, and potentially faster data access, networked data servers often provide a greater ease of use. Ponto et al. demonstrated the network requirements for various means of data distribution in a distributed tiled display environment [11].

B. Synchronized Distributed Decoding

The video decoder is a multi-threaded distributed decoding system built on top of the ffmpeg suite of libraries and can handle a wide range of both container formats and codecs for audio and video. Decoding and playback is synchronized across the display wall so that all tiled videos play together as a seamless high-resolution video. Intelligent video culling allows a large number of simultaneous videos to play across the wall, with each display node only processing the minimum subset of videos required to update their displays. As each node in the tiled display system dynamically determines which videos are within its viewpoint, videos can be repositioned on the fly. This system enables users to zoom and pan through video information which is even larger than the display workspace.

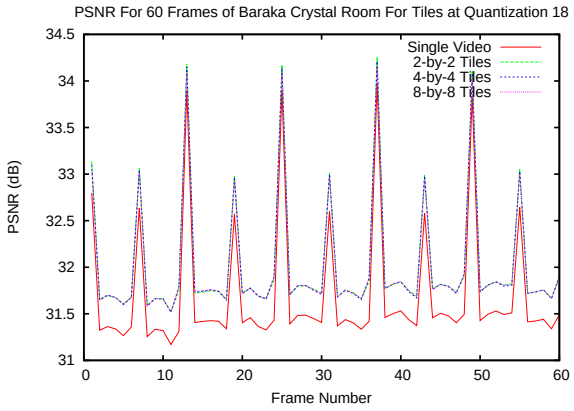

Figure 7. PSNR comparison between non-tiled video with a quantization value of 19 and tiled videos with a quantization value of 18.

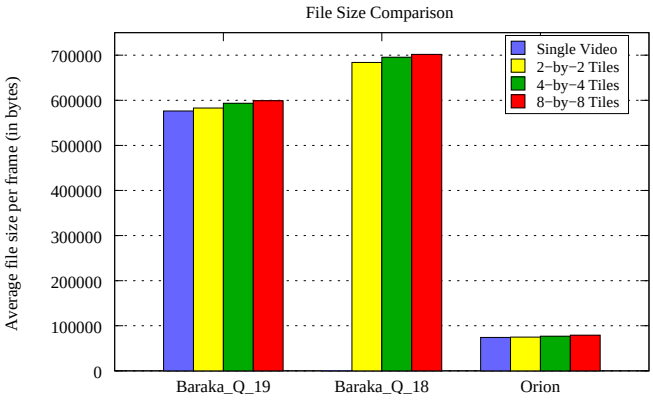

Figure 8. Filesize comparisons between a single video and tiled videos, including two different quantization values for the Baraka test video.

1) Video Display in Visualization Middleware: CGLX is an OpenGL based middleware which provides a windowing environment which implements a unified display-context across multiple displays on a visualization cluster [20]. It synchronizes the display and user input event loops across all of the display nodes using a network barrier. We use this library to facilitate the display of our decoded video data as well as the transport of cluster-wide messages for synchronization events.

2) Frame Synchronized Playback: The distributed decoding and playback operation is orchestrated by a synchronization mechanism running on the head node. The head node keeps track of a video progress timer which is used to advance each new frame in the video sequence. The timer is either tied to the progression of audio packets played back by the sound card or to a high-performance system clock in the absence of an audio track.

While the head node does not need to display video, it does have to keep track of the PTS (presentation time stamp) to know when to advance to the next frame. The PTS values are identical for each video tile, so only one video tile needs to be loaded to get video frame PTS values and also any audio track to be played. This requires the video frames to be read from disk, but not decoded, and therefore very little processing power is needed for the head node's synchronization.

Encoding Sequence	Tilings			
Baraka Q = 19	1	2x2	4x4	8x8
KB Per Frame	576	583	593	599
Size Increase	NA	1.12%	2.98%	3.94%
Baraka Q = 18	1	2x2	4x4	8x8
KB Per Frame	NA	684	695	702
Size Increase	NA	18.67%	20.67%	21.77%
Orion	1	2x2	4x4	8x8
KB Per Frame	74.2	74.9	76.9	79.2
Size Increase	NA	0.93%	3.65%	6.71%

Figure 9. File size comparison for different tile configurations. Size increase is in comparison to the non-tiled video with a quantization value of 19.

During each cycle of the display loop the video progress timer is updated and compared against the PTS of the next video frame to be displayed. When the time is greater or equal to that frame's PTS, a CGLX event message with updated the PTS value is sent to all of the display nodes.

3) Visibility in Initialization and Interaction: The massive scalability of the system is derived from an architecture which allows each node to only decode and display individual video tiles which are contributing content to their portion of the display canvas. In so doing, each node only uses computational resources to processing video tiles currently in view.

Video tile geometry is loaded from an XML file which provides the file path for each video tile and describes the orientation tile in an n-by-m grid. Upon initialization, each node utilizes global display scene information and reads the geometry description of the tiled video to check tile visibility using the culling system. For each visible video tile a video decoding worker process is created.

Nodes which do not display culled video tiles do not need to read, decode, or display expend resources processing culled tiles. However, when videos are moved in the global scene, individual tile visibility changes from the perspective of render nodes. As such, tiles no longer necessary to the view of an individual render node become culled, and new tiles which become visible must be able to start displaying. Upon identifying the need for a new tile on a render node, playback begins by seeking to the most recent I-Frame before the current display PTS value and decoding frames until the current PTS value is reached and normal playback resumes. Decoded frames below the current PTS are discarded without being uploaded to the video card, to speed the process. This fast forward happens quickly enough to not significantly impact the user.

Figure 10. Three high resolution videos playing.

Time (ms)	Tilings				
MPEG2	1	2x2	4x4	8x8	16x16
Time to Read	NA	0.17	0.10	0.03	0.02
Time to Decode	NA	6.64	2.60	0.61	0.21
Time to Upload	NA	1.55	0.57	0.10	0.06
Total Time	NA	8.36	3.27	0.74	0.29
MPEG4	1	2x2	4x4	8x8	16x16
Time to Read	0.15	0.10	0.08	0.05	0.01
Time to Decode	15.96	6.50	2.57	0.71	0.25
Time to Upload	5.66	1.70	0.25	0.08	0.06
Total Time	21.77	8.3	2.90	0.84	0.32
H.264	1	2x2	4x4	8x8	16x16
Time to Read	0.97	0.35	0.22	0.19	0.10
Time to Decode	30.6	9.10	4.55	2.61	0.74
Time to Upload	5.70	1.60	0.521	0.10	0.07
Total Time	37.27	11.05	5.29	2.89	0.91

Figure 11. The average time in each frame for read, decode and upload various for video tlings.

For both initialization and run-time changes to the scene, the process of loading and unloading video tiles is local to each rendering node, as the operation does not require any coordination from the head node to select tiles, nor do they need to communicate with any neighboring nodes. Globally synchronized information about the scene geometry is sufficient to allow each node to compute which video files are visible within its portion of the global scene.

As the described approach can take full advantage of computational resources driving multi-tile display environments with minimal network overhead, we argue the implemented architecture can scale well beyond existing techniques.

VI. Scalability Analysis

Quantifying video playback performance is a difficult endeavor. In many ways, playback is a binary measure, either the system can play the specified video or it can not. Figure 10 demonstrates three videos playing on an eight tile display wall, driven by four display nodes. The two videos in the foreground are at 4k resolution and the video in the background is approximately 20 mega pixels (6400-by-3072).

Since the purpose of the video tiling is to distribute the processing of large videos among many computers, it is important to evaluate how splitting videos into various sub components reduces T_{Frame} (as shown in Equation 1). Figure 11 shows the average performance characteristics of T_{Read}, T_{Decode}, and T_{Upload} for a single video tile

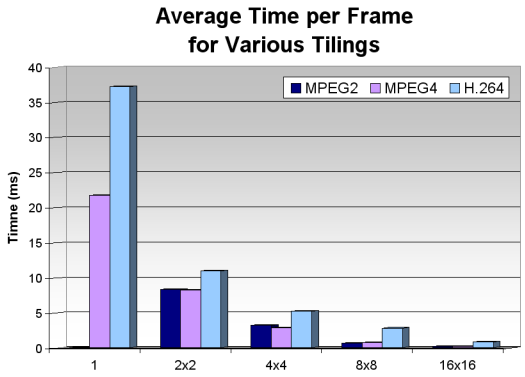

Figure 12. The time required to read, decode and upload a frame in each of the different video tile configurations.

across different tile configurations and Figure 12 shows the reduction of the overall T_{Frame} for a single video tile as tiling increases. Unfortunately due to the frame size exceeding the MPEG2 standard [14], the non-tiled result could not be verified for the MPEG2 format.

Chen attempts to measure the performance of their system with the metric of mega-pixels decoded per second [4]. While this metric is somewhat relevant, it is also ambiguous. As stated above, decoding is dependent on the video codec used in the encoding, the parameters given to this encoder (such as bitrate), and the content of the individual frames. On top of this, the performance characteristics of the machines used will greatly vary this number. While the individual results may not be representative, Chen demonstrates that decoding can be parallelized, resulting in substantial performance benefits [4].

Instead of measuring the direct performance of any given system, we believe it is more informative to analyze the scalability of the methods used for ultra-high-resultion video playback. As the approach in this paper allows video frame reads, decodes, and uploads to be fully parallelized for a cluster of machines, the limiting factor preventing infinite scaling is due to network data transfer. Many of the other approaches have network bandwidth requirements which scale by

$$Bandwidth \approx framerate \times framesize \quad (3)$$

In the method presented in this paper, the bandwidth is proportional to

$$Bandwidth \approx framerate \times numberofnodes \quad (4)$$

As the number of nodes is significantly smaller than the number of pixels, the presented approach results in a massive savings for network overhead. To demonstrate this disparity, we measure the difference between the maximum theoretical video frame size possible for the approaches of SAGE [7], the Macro-Block forwarding method used by Chen [4], and

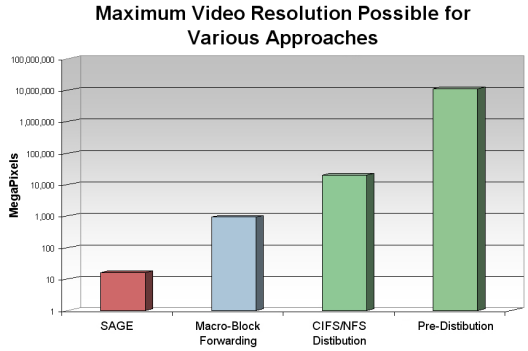

Figure 13. The theoretical maximum video resolution given only the constraints of a 10 GbE network card.

the method presented in this paper, using data that is either pulled locally to save network bandwidth or via a network filesystem. In order to determine the theoretical maximum video resolution possible, we will assume the maximum data transfer to or from any machine is 10 gigabits per second as a baseline. This metric is useful as it is not dependent on system performance and can therefore evolve with changing hardware.

For the approaches of SAGE and this approach we assume a system consisting of a single head or streaming node and 16 render nodes. For the Macro-Block forwarding method we will assume an extra 4 splitter nodes are used, matching the 1-4-(4,4) method shown in [4]. We will assume a video quality of .310 bits per pixels at 24 fps, matching the exact parameters shown for Stream 16 in [4].

As shown in Figure 13, the SAGE approach is able to scale to 16 megapixels before the network interface is fully saturated by the raw pixel data. The Macro-Block approach scales much more effectively and will be saturated at just under a gigapixel from the compressed video data. The presented approach reduces the amount of information routed through the network even when pulling the data off of a network mounted drive. As this approach does not incur the 20% overhead seen in the Maco-Block forwarding method and the data being pulled is distributed amongst the rendering nodes, the approach is able to scale to a maximum frame size of 20 gigapixels. By distributing the data to the nodes a priori, the network requirements are reduced substantially. In this approach the only information passed via the network interface are control signals, allowing the system to scale to a theoretical limit of 10 terapixels.

VII. Conclusion

This paper presents a scalable approach for arbitrary sized video playback on tiled display systems. Playback of ultra-high-resolution video is made possible through a preprocessing step, creating tiled content. By tiling data, the workload of reading, decoding, and uploading video frames is distributed throughout the entire display environment. Due to the low network bandwidth used, this approach scales incredibly effectively, far exceeding previous methods. We see the proposed method as relevant in both the academic and the entertainment communities, as it provides a framework for next generation, scalable multimedia technology.

References

[1] T. A. DeFanti, J. Leigh, L. Renambot, B. Jeong, A. Verlo, L. Long, M. Brown, D. J. Sandin, V. Vishwanath, Q. Liu, M. J. Katz, P. Papadopoulos, J. P. Keefe, G. R. Hidley, G. L. Dawe, I. Kaufman, B. Glogowski, K.-U. Doerr, R. Singh, J. Girado, J. P. Schulze, F. Kuester, and L. Smarr, "The optiportal, a scalable visualization, storage, and computing interface device for the optiputer," *Future Gener. Comput. Syst.*, vol. 25, no. 2, pp. 114–123, 2009.

[2] L. Herr, "Creation and Distribution of 4 K Content," *Television Goes Digital*, p. 99, 2008.

[3] H. Shimamoto, T. Yamashita, N. Koga, K. Mitani, M. Sugawara, F. Okano, M. Matsuoka, J. Shimura, I. Yamamoto, T. Tsukamoto *et al.*, "An Ultrahigh-Definition Color Video Camera With 1.25-inch Optics and 8k x 4k Pixels," *SMPTE Motion Imaging Journal*, pp. 3–11, 2005.

[4] H. Chen, "A parallel ultra-high resolution mpeg-2 video decoder for pc cluster based tiled display system. to appear," in *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS), IEEE CS*. Press, 2002, p. 30.

[5] H. Chen, G. Wallace, A. Gupta, K. Li, T. Funkhouser, and P. Cook, "Experiences with scalability of display walls," in *Proceedings of the immersive projection technology (IPT) workshop*, 2002.

[6] D. Shirai, T. Yamaguchi, T. Shimizu, T. Murooka, and T. Fujii, "4k shd real-time video streaming system with jpeg 2000 parallel codec," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, Dec. 2006, pp. 1855–1858.

[7] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh, "High-performance dynamic graphics streaming for scalable adaptive graphics environment," in *SC 2006 Conference, Proceedings of the ACM/IEEE*, 11-17 2006, pp. 24 –24.

[8] L. Renambot, A. Johnson, and J. Leigh, "Lambdavision: Building a 100 megapixel display," in *NSF CISE/CNS Infrastructure Experience Workshop, Champaign, IL*, 2005.

[9] B. Jeong, J. Leigh, A. Johnson, L. Renambot, M. Brown, R. Jagodic, S. Nam, and H. Hur, "Ultrascale collaborative visualization using a display-rich global cyberinfrastructure." *IEEE Computer Graphics and Applications*, vol. 30, no. 3, pp. 71–83, 2010.

[10] G. Choe, J. Yu, J. Choi, and J. Nang, "Design and implementation of a real-time video player on tiled-display system," in *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, oct. 2007, pp. 621 –626.

[11] K. Ponto, T. Wypych, K. Doerr, S. Yamaoka, J. Kimball, and F. Kuester, "Videoblaster: A distributed, low-network bandwidth method for multimedia playback on tiled display systems," in *IEEE International Symposium on Multimedia*, December 2009, pp. 201 –206.

[12] G. Wallace, O. Anshus, P. Bi, H. Chen, Y. Chen, D. Clark, P. Cook, A. Finkelstein, T. Funkhouser, A. Gupta, M. Hibbs, K. Li, Z. Liu, R. Samanta, R. Sukthankar, and O. Troyanskaya, "Tools and applications for large-scale display walls," *Computer Graphics and Applications, IEEE*, vol. 25, no. 4, pp. 24–33, 2005.

[13] H. Chen, "Scalable and Ultra-High Resolution MPEG Video Delivery on Tiled Displays," Ph.D. dissertation, Princeton University, 2003.

[14] J. Mitchell, *MPEG video compression standard*. Kluwer Academic Publishers, 1997.

[15] L. Renambot, B. Jeong, H. Hur, a. Johnson, and J. Leigh, "Enabling high resolution collaborative visualization in display rich virtual organizations," *Future Generation Computer Systems*, vol. 25, no. 2, pp. 161–168, Feb. 2009.

[16] D. Svistula, J. Leigh, A. Johnson, and P. Morin, "MagicCarpet: a high-resolution image viewer for tiled displays," 2008.

[17] K. Ponto, K. Doerr, and F. Kuester, "Giga-stack: A method for visualizing giga-pixel layered imagery on massively tiled displays," *Future Generation Computer Systems*, vol. 26, no. 5, pp. 693 – 700, 2010.

[18] R. Fricke, "Baraka," Magidson Films Inc, 1992, film.

[19] D. Nadeau, J. Genetti, C. Emmart, E. Wesselak, and B. O'Dell, "Orion Nebula Visualization," San Diego Supercomputer Center, 1999.

[20] K.-U. Doerr and F. Kuester, "CGLX: A scalable, high-performance visualization framework for networked display environments." *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 3, pp. 320–332, Apr. 2010.