

Addressing Server Latency and Capacity to Enable Fast and Affordable Wireless Image Data Services

Dong-Gi Lee and Sujit Dey
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, California, USA
{synclee, dey}@ece.ucsd.edu

Abstract - With the widespread introduction of new wireless data services such as mobile multimedia email, ubiquitous Internet access, mobile commerce, home and medical monitoring services, and mobile conferencing, there will be a growing demand for multimedia-rich data services over wireless networks. A key challenge in enabling media-rich wireless data services is ensuring low and predictable end-to-end latency despite increasingly sophisticated algorithms deployed on wireless data servers to accommodate (i) the wide diversity in client hardware resources, service requirements, quality of connectivity, etc., and (ii) the need for customized/personalized client services. These algorithms may often cause servers to experience significant computational loads, leading to a potential degradation of server throughput, unacceptably high server response times, and poor end-to-end latency.

In this paper, we analyze the impact of content adaptation services on an image data server by examining server latency, capacity, and cost metrics. Our findings indicate that our proposed dynamic transcoding technique can increase server capacity by up to 3X and decrease server latency by up to 5X on average compared to conventional transcoding method when implemented in software. Even more significant gains are achieved through use of a *configurable Hardware/Software (HW/SW) architecture* that is capable of several different types of transcoding services while increasing the server capacity by up to 22X and decreasing the server latency by up to 30X. We present results demonstrating that by combining dynamic transcoding and our proposed configurable HW/SW architecture, image-based data services can be enabled at significantly reduced server costs, and thereby reduced overall wireless service cost.

1 INTRODUCTION

With the introduction of next-generation mobile data services and increasing user demand for ubiquitous Internet access, coupled with a proliferation of multimedia content on the Internet, there will be a drastic increase in the use of mobile handheld devices for multimedia data access over wireless networks. However, to enable multimedia data services over wireless networks requires more than a simple manifestation of the wireline data services on a mobile handheld device due to the specialized traits of wireless communication. For example, file sizes of the multimedia data must be small enough to minimize end-to-end latency, network traffic, and processing power in a mobile handheld device. Also, because mobile handheld devices vary significantly in their processing power, screen size, input mechanisms, battery capacity, and ability to present information (not all devices can support the image types or formats prevalent on the current Internet). Personalization is a key factor in making the wireless multimedia experience compelling for the end user. For example, standardization groups like ETSI [1] and 3GPP [2] that have been working towards the new generation of multimedia enabled wireless data services have both proposed standards for personalization of data to individual users. Therefore, a paradigm shift in how services are provided from a passive role - focused on the classical receive of the data services where data is "blindly"

provided to the all recipients in the same format, to a more active role - including user profiles, personalization, and the possibility to store and retrieve their own customized data is occurring.

One promising technique for providing active data services over the wireless network is image transcoding. Transcoding is used to serve variations of the same multimedia object with different formats and sizes, and usually trades off object fidelity (quality) for size. A significant amount of research work has been proposed using various forms of "transcoding" techniques to customize images for the prevailing network conditions or the destination target device characteristics [3,4,5,6]. These works can be classified into two main types (static and dynamic) according to when the different content variations are created [7]. Static transcoding is when different "pre-adapted" versions are created and stored in the server at content creation time, while dynamic transcoding creates a new version of the multimedia object according to dynamic requirements presented to the server. Most of the dynamic transcoding approaches focus on the tradeoff between image quality and size of wireless transmission. In our previous works [8,9], we further extended the meaning of dynamic transcoding by optimizing not only for image size and image quality, but also to affect the quality of service (service latency) and energy consumption during wireless communication. We characterized the effects of varying transcoding parameters on QoS (service latency) and energy consumption, and transcoded the image through runtime parameter selection depending upon the dynamic network conditions and appliance status.

1.1 Paper Overview and Contributions

While our previous work tried to address wireless network-side service issues using dynamic transcoding technique, in this paper, we move our attention to the server-side issues related with the dynamic transcoding technique. The proposed dynamic transcoding technique can be used to enable wireless multimedia communication, however, the implementation of dynamic transcoding requires significant computational resources at the image data server. This paper addresses the effect that implementing a dynamic transcoding technique has on an image data server. To satisfy end-to-end latency constraints, the time taken to dynamically transcode an image must be taken into account. Therefore, in this paper, we analyze the end-to-end latency achieved through the use of dynamic image transcoding. We also investigate the potential benefits and overheads of providing active image data services using various transcoding methods on the server performance in terms of server latency, server capacity, and end-user cost. We measure the server performance while varying the incoming service request rate (service workloads) and types of service requested (service features). Our experimental results demonstrate that by optimal parametric using the proposed dynamic transcoding technique, we can increase server capacity by 3X and decrease server latency by 5X on average.

While software-based dynamic image transcoding can provide for improved server capacity and latency, even more significant gains can be achieved through the use of a novel transcoder

This work was supported by the Semiconductor Research Corporation under contract number 2001-HJ-900.

architecture. To this end, we explore different architectural options in a data server implementation and propose a *configurable Hardware/Software (HW/SW)* architecture for the image transcoder. The proposed architecture is capable of achieving significantly higher performance while accommodating a variety of transcoding services (flexibility). We specifically designed the hardware components for computation-oriented tasks in the transcoding process, while allowing the flexibility required to cover various transcoding services through parameterized hardware implementation. Our experimental results indicate that the dynamic image transcoding methodology, coupled with the proposed configurable HW/SW architecture can drastically reduce the server latency (by up to 30x) as well as increase server capacity (by up to 22x), leading to lower server costs as well as total service cost. In addition, our proposed architecture can be easily adapted to any kind of transcoding services without performance degradation.

The remainder of this paper is organized as follows. First, the overall service architecture for wireless image data services is described in the next section. In section 3, the performance evaluation method for service architectures is presented. Section 4 describes the exploration of optional server architectures for a wireless data server, together with our proposed architecture, and presents the impact of different architectures on the server performance. Finally, section 5 concludes the paper.

1.2 Related Work

1.2.1 Image-based Content Adaptation Techniques

Numerous approaches have been proposed to pursue efficient active data services using static or dynamic transcoding techniques to provide more flexibility and offer personalized versions of target images for the new generation of mobile users. For static transcoding [3,4,5,10,11], several “pre-adapted” versions of an image are created and stored in a data server at content creation time. During runtime, a version that best matches the desirable version in a request will be returned. The primary shortcoming of static adaptation, however, is that the management and maintenance of a number of content variants in the server results in considerable storage and other I/O costs. Therefore, static adaptation can be very efficient (with few content variants), at the cost of not being able to adapt to as many conditions, or it can adapt to many different conditions, at the price of efficiency.

In contrast, using dynamic transcoding approaches [6,10,11,12], the desired content is synthesized on the fly according to the dynamic service requirements and constraints. This approach can meet a wide range of varying needs, but the complexity and the time delay incurred in order to prepare content on the fly is a bottleneck to deployment. Most of the above approaches focused on achieving optimal compression with a minimal sacrifice of the image quality. They did not much talk about the extra computation overhead, its effect on the data server (server latency increase), and addressing technique for this overhead. In our previous works [8,9], however, we further extended the meaning of dynamic transcoding by showing that transcoding technique can be used to achieve not only the optimal compression and image quality, but also to affect the quality of service (service latency) and energy consumption during transcoding process.

1.2.2 Design Architecture for Adaptive Image-based Content Shaper

Past work has demonstrated various reconfiguration architectures at different levels of design, namely software-level, datapath-level, and logic-level, to dynamically reconfigure across different

algorithms. Software-level reconfigurability using general purpose or custom-fit processors [13] provides the highest amount of flexibility at the cost of poor performance in terms of execution time and power consumption. Datapath-level configurability like the RAPID architecture [14], as well as logic-level configurability like FPGA based systems [13,15,16], provide better execution time and power characteristics than general purpose processors, but still cannot match the performance of an ASIC implementation for a specific application. In addition, the overhead associated with run-time reconfiguration of an FPGA-based architecture (in terms of reconfiguration time and the storage of configurations in memory) can be significant. The adaptive image shaper proposed in this paper has to be low-energy, real-time, rapidly adapting to dynamically changing input conditions, and configurable in a parametric way. Consequently, the existing software, datapath, and logic-level reconfiguration architectures may not be sufficient for the adaptive image shaper proposed in this paper.

2 SERVICE ARCHITECTURE FOR WIRELESS IMAGE DATA SERVICES

The service architecture for wireless image data services that we envision is illustrated in Figure 1. The main components of this service architecture are 1) data servers, 2) wireless gateway servers, and 3) mobile clients. The data servers store the original image content and convert the content into a customized form for each mobile client, taking into account the client’s service requirements and associated constraints. The wireless gateway server typically resides at the base station, and mediates the data services between the mobile clients and the data servers.

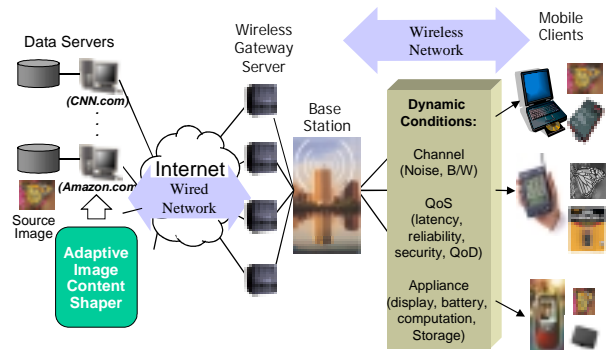


Figure 1. Service architecture for wireless image data services

To enable personalized image data services that adjust to each mobile client, an *adaptive image content shaper* is proposed, which resides at the data servers and controls the image content shaping (transcoding or adaptation) process. The adaptive image content shaper is implemented at the server-side (as opposed to at the gateway server or the client-side), as proximity to the original image data enables it to better understand the image content and configure it accordingly. The functional block diagram of the adaptive image content shaper is illustrated in Figure 2. It first monitors dynamic conditions such as wireless channel conditions, QoS (service latency) requirements, and appliance status (*Monitoring Dynamic Conditions*). It then selects the most efficient content adaptation technique based on the client’s request (*Algorithm Choice*), and optimizes the configuration of the algorithm selected for the client (*Optimize Configuration*). Finally

it shapes the requested image using the selected technique with the chosen optimal configuration.

Three different content adaptation techniques are used as candidates for the algorithm selection step. They are (1) simple transcoding, (2) static transcoding, and (3) dynamic transcoding. In simple transcoding, the multimedia server stores the original sample images and uses a standard image codec *e.g.*, JPEG [17] or JPEG2000 [18]) to compress the images. However, the configuration of the standard image codec is statically fixed, and so it can adapt the image only in a fixed way without taking the dynamic conditions into account. Alternatively, a server can store a number of different image versions, generated a-priori, so as to satisfy different client requirements and constraints, and select the most appropriate version for delivery depending on the dynamic conditions (“*static transcoding*”). We also add “*dynamic transcoding*” service for active content adaptation where the multimedia server stores just one version of each image, and synthesizes a customized version of the image on the fly (derived from the stored image) taking into account the dynamic requirements and constraints presented to the server. Compared to simple image transcoding, the image version generated in this approach must be the most appropriate version (in terms of size and quality) by considering the unique service characteristics. We choose a wavelet-based dynamic image transcoding technique introduced in our previous work [8,9] an example “*dynamic transcoding*” technique. For the “*static transcoding*” method, image versions for all possible service scenarios are generated a-priori using the wavelet-based image transcoding algorithm and stored on the data server.

To implement dynamic transcoding, we first identified available JPEG2000 image compression parameters and characterized the effects of varying JPEG2000 compression parameters on processing time (latency), energy consumption, compression ratio, and image quality. We next introduced a runtime parameter selection algorithm that selects the optimal image compression parameters depending upon the dynamic network conditions and appliance status. The effectiveness of these techniques for minimizing energy consumption and service latency while correctly balance image size and image quality has been previously demonstrated in the Palm.Net wireless data service environment [19] using a PalmVII handheld [20]. While our previous work tried to address wireless network-side service issues using dynamic transcoding technique, in this paper, we move our attention to the server-side issues related with the dynamic transcoding technique. We focus on addressing latency and capacity issues at the server. More detailed information about the dynamic image transcoding technique can be found in [8,9].

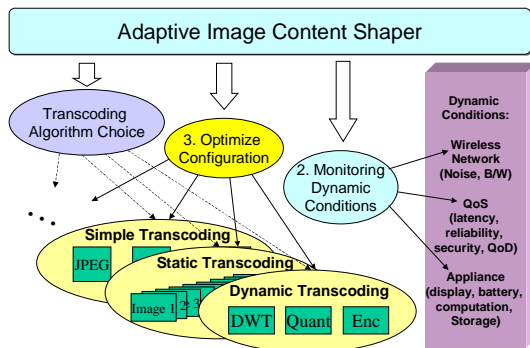


Figure 2. Functional block diagram for the proposed adaptive image content shaper.

Our proposed service architecture is quite generic and can be used with any wireless access technology and mobile data service architecture. In the next section, we present the performance evaluation framework and the metrics used to evaluate and compare server performance in deploying the above active image data services.

3 SERVER PERFORMANCE EVALUATION

Our proposed performance evaluation framework is a simulation-based framework that can generate random service workloads, execute the image transcoding process for each service workload, and measure the effects of image transcoding on server performance. It captures the server performance in terms of server latency, capacity, and server cost metrics. Figure 3 illustrates the functionality of our proposed simulation-based evaluation framework. It consists primarily of three major task modules: (1) generating random service workloads and mapping client requirements and surrounding constraints into each service workload, (2) executing image adaptation tasks for each service workload and measuring the server performance metrics, and (3) exploring different server architectures. In the next sub-section, we describe each task module in more detail.

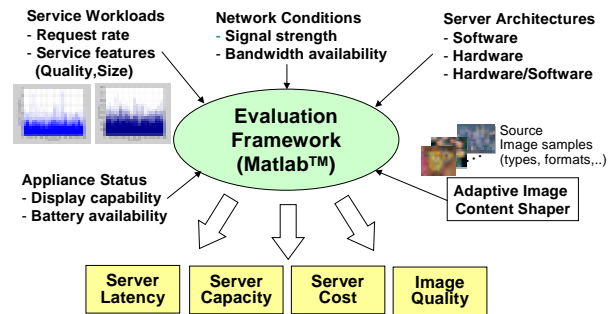


Figure 3. Evaluation and exploration framework for investigating server-side issues by active image data services.

3.1 Random Service Workloads Generation

Here we present how we generate image service workloads to evaluate server performance. Since our active image data services target next generation active image data services, there are currently no actual recorded service traces available. Hence, we generate analytical service workloads based on random distributions.

We first generate the arrival times for 10,000 service requests using a Poisson distribution [21], a common distribution used for representing network workloads. The inter-arrival times for the service requests follow an exponential distribution [21], dependent upon the average input service request rate (number of incoming service requests per unit second).

The second parameter generated for our service workloads is the kinds of services demanded by the mobile clients and the associated conditions for each client (termed *service request types or features*). General service request features contain information about the current network conditions and the status of the appliance (battery power, screen size, *etc.*) when requesting the active data services. Image-based service request features will also include information on the requested image type (color or gray), allowable formats (JPEG, GIF, JPEG2000), image size, and image quality (PSNR) requirements. For each service request, an image

randomly selected from a set of 100 benchmark image samples is uniformly assigned to the request.

By combining the randomly chosen images and service request features (or types) with the arrival times for each service request, a service workload is finalized. This service workload is used to evaluate the server performance while performing image transcoding process. In the next sub-section, we discuss in more detail how the server performance is evaluated for each service workload generated.

3.2 Measuring Server Performance

Using our server performance evaluation framework, we evaluate the server performance for active image data services in terms of server latency, capacity, and server cost. In this section, we describe the process of evaluating each aspect of server performance in detail.

• Measuring Server Latency

Server latency is defined as the total time a data service request spends waiting for the server to customize the image content (in the case of simple or dynamic image transcoding) for the specific request. It includes the time spent waiting for the server to start processing the service request (*service wait time*), together with the time required to generate the appropriate image version (*server processing time*).

As described in section 2, the dynamic image transcoding technique imposes additional computational load on the data server to shape the image content optimally at run-time. We further divide the process of dynamic image transcoding into (1) a configuration task and (2) a coding task. The configuration task involves selecting the optimal transcoding algorithm and choosing its optimal configuration (configuration parameter selection) based on the given input requirements and client-associated conditions. The coding task involves performing the actual image adaptation using the selected algorithm with the chosen configuration.

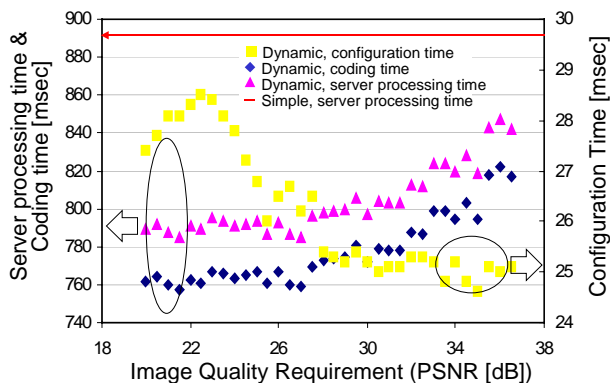


Figure 4. Profiled CPU execution time during adaptive image adaptation process (440MHz SUN UltraSPARC 10 Workstation)

Figure 4 illustrates the CPU execution time consumed (left y-axis) for different image quality requirements (x-axis) during the image transcoding process. The simple transcoding (using JPEG2000 [18]) and our proposed dynamic transcoding [9] tasks were implemented fully in software and the CPU execution time was profiled for performance comparison under different image service workloads (as described in section 3.1). The experiments were performed on a 440MHz SUN UltraSPARC 10 Workstation,

and assuming a Palm VII handheld as the client appliance used. The time required for the configuration task is also plotted (right y-axis), and is significantly lower than the time required for actual shaping an image. Note that the time required to actual shape the image (coding time) can be reduced by having a lower image quality requirement in our dynamic transcoding approach [8,9], while the simple image transcoding approach remains constant throughout all different input requirement (image quality requirement). The reason of this result is our dynamic transcoding technique was specifically designed to save latency/energy during transcoding process by changing its configuration (image compression parameter choice) according to input requirements, while the configuration of simple transcoding is fixed and could not be changed. In contrast, the configuration time plot shows reverse trends such that when a lower image quality is required, the configuration task takes longer as it explores more configuration options to find the most efficient configuration. Even so, this extra time required in the configuration task is minimal compared to the savings in time during actual image shaping process.

The other component of server latency that we consider is the service wait time in a data server queue. Due to the finite number of data servers, it is possible that a service request arrives before the previous service request is completed. We assume a First Come First Served (FCFS) service scenario where there is no service rejection (infinite queues are available) in a data server. To estimate the service wait time, we first use a service request arrival time assigned as described in section 3.1. To determine the service wait time for each service request, we begin by evaluating the service request with the earliest arrival time. The first request is evaluated to determine how long the configuration and shaping process takes (server processing time). If this server processing time does not complete before the arrival time of the next service request, the transcoding task of the next service request gets delayed. Once the processing of the current request ends, the processing of the next service request starts immediately. Using this (possibly) delayed start time (service waiting time), the time at which the computation completes for each service can be computed. This process is repeated until the computation start and end times of each service request have been completely measured. Using this method, we can capture the service wait time for each service request in the analytical service workload generated in section 3.1.

• Measuring Server Capacity

Server capacity is defined as the number of service requests per second that a data server can process without dramatically increasing the server latency due to server overload. Under normal load conditions, when the rate of incoming requests is below the capacity of server, the server can serve all incoming service requests without introducing large delays. However, when high load conditions exist, *i.e.*, when the incoming rate of service request exceeds the server capacity, the server cannot keep up with the arrival rate of incoming service requests, and the service wait time in a data server queue becomes a bottleneck. Thus the server latency theoretically increases to infinity. This is depicted by the sudden increase of the server latency. While a client's tolerance for total service response delay is application dependent, typically a total delay of 1 ~ 10 seconds for web-based applications is considered acceptable [22]. If the total service response delay is longer than this threshold, the clients tend to lose interest. For our experiments in this paper, we assume that the maximum user

endurance server latency is 1 second and use this value as a threshold for extracting the server capacity for performance evaluation. Assuming a threshold of 1 second allows the remaining time (approximately 0~9 second) to be consumed during wireless communication.

• **Measuring Server Cost**

The server cost is determined in terms of storage space required for a given transcoding technique and how many servers are required to satisfy the incoming service request rates under user acceptable server latency. For example, assuming user acceptable server latency is 1 second, and it takes 0.5 seconds to process each service request, a total of 2 identical servers are required to satisfy 4 incoming service requests within 1 second. This metric can be used to measure the server cost savings achieved in different service methodologies (simple versus dynamic image transcoding techniques) and architectures. For the case of storage space requirement for static transcoding method, we generate all possible versions of image sample a-priori based on total different image samples can be provided using dynamic transcoding method, and measure the total storage space occupied.

In the next section, we evaluate the effect and overhead of active image data services with these server performance metrics.

3.3 Impact of Image Transcoding Techniques on Server Cost, Latency, and Capacity

To evaluate the impact of image transcoding techniques on a data server, we use the server performance evaluation methods outlined in sections 3.1 and 3.2. We compare the result of dynamic image transcoding approach with the result of static and simple image transcoding cases.

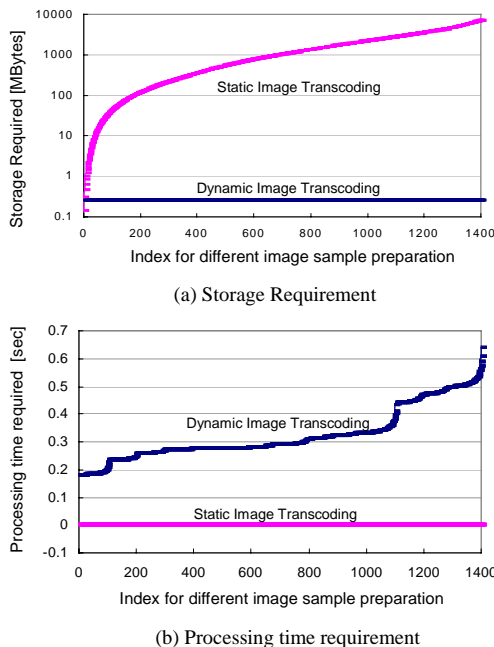


Figure 5. Performance comparison between static image adaptation vs. dynamic image adaptation technique

We first compare the benefits and overheads of “static” vs. “dynamic” image transcoding methods. We measured the storage space requirement and extra computation overhead. Figure 5 illustrates total amount of storage required and computation

overheads for two image transcoding techniques. We assumed the total number of service differentiation required for “static image transcoding” technique is the same as possible image provision using “dynamic image transcoding” (we found that total 1400 service differentiation can be provided to satisfy the dynamically varying service environment and user requirements). The total amount of storage is represented as the accumulated storage space for this preparation assuming 512 x 512 gray LENA benchmark image sample used. Note that as expected, in contrast to the extra computation overhead of dynamic image transcoding, there is no computation overhead for static image transcoding as all image samples are prepared and stored a-priori. We ignored the extra time overhead for searching and finding the most optimal image sample for static case. However, in our dynamic image transcoding, as we presented in section 3.2, note that depending upon different parameter selections (configuration choice), the server processing time varies significantly. We sort the server processing time from lowest to highest after profiling.

Next, we compare the benefits and overheads of “simple” vs. “dynamic” image transcoding methods. We extract the server performance metrics for this comparison. Figure 6 illustrates the different latencies incurred to provide active wireless image data service using image transcoding techniques. The top 2 graphs represent the server processing time (time consuming in configuration and image shaping to process a service request) and the service wait time (time consuming in a data server queue before image shaping process begins), respectively. The bottom graph shows the time delay required to transmit the image across the wireless channel assuming a 9.6 kbps Palm.Net environment used [19]. As shown in all three graphs, the latency incurred using a dynamic image transcoding is significantly lower than a simple image transcoding approach.

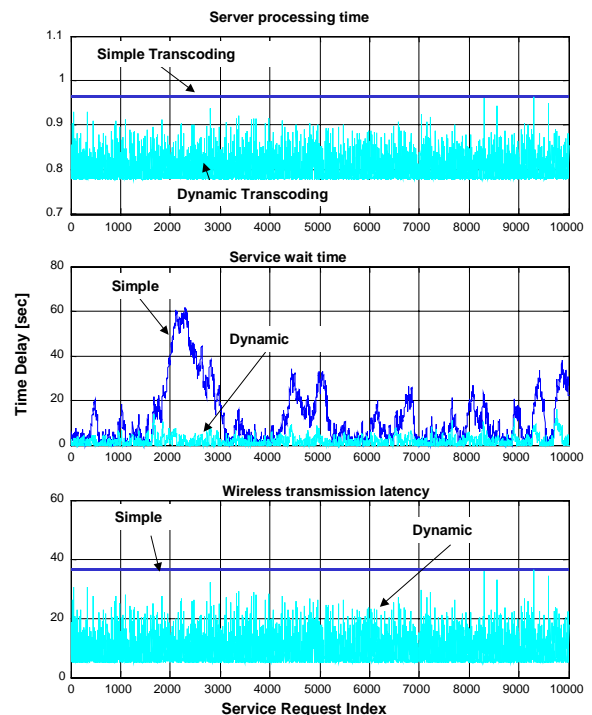


Figure 6. Impact of dynamic image transcoding on the different latencies incurred in image shaping process

On Figure 7, the total service latency, divided into server latency (the dark portion of the graph points) and wireless transmission delay (the light portion) is shown. Note that the worst-case delay in the simple image transcoding case is 99.2 seconds; while in the dynamic transcoding it is 44.9 seconds (a 54.3 second difference!). Also, average savings in total service latency from the simple image transcoding case to the dynamic case is 5x, a very significant savings. We can see similar trends on server capacity in Figure 9. It shows the server capacity can be increased (3x) compared to the simple conversion case. It is worth noting that the large latencies shown in Figure 7 are due primarily to the extremely low-bandwidth wireless channel (9.6 kbps), while the server latency still takes a large portion of total service latency (accounts for 18.5 % of the total service latency in average). However, in the case of a higher-bandwidth wireless channel, the server latency would become a much more significant portion of the total service latency, necessitating new ways to reduce the server latency. Therefore, in the next section, we develop novel server architecture to further reduce server latency and increase the server capacity in the dynamic image transcoding approach.

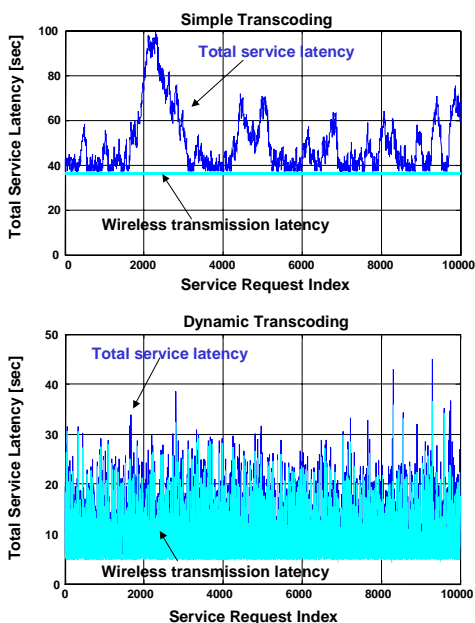


Figure 7. Impact of dynamic image transcoding on the total service latency

4 IMPROVING SERVER CAPACITY AND LATENCY USING CONFIGURABLE HW/SW ARCHITECTURE

In order to develop a server architecture for adaptive wireless image data services capable of delivering end-to-end service latencies within acceptable server latency and with affordable server capacity, we developed a configurable hardware/software (HW/SW) architecture. As illustrated in Figure 2, the functional structure of a data server for active image data services is composed of two main components: the configuration process and actual coding process. Therefore, both portions must be considered during the development of the HW/SW architecture. Traditional mappings of tasks into a HW/SW architecture is usually done with the objective of optimizing for performance (*i.e.*, time and power).

In our approach, in addition to the above objective, we further consider adaptability (or flexibility) as a new objective in mapping algorithm tasks to HW or SW. SW is easily adaptable (or flexible), whereas HW configuration has better performance and reduced energy consumption, but less flexibility.

In the next sub-sections, we present our architectural mapping process and its detailed impact on server performance.

4.1 Configurable Hardware/Software Architecture

Before mapping the proposed algorithm tasks to HW or SW, it is necessary to figure out which tasks are the most optimal for HW and which tasks are optimal for SW. For this task partitioning, we profiled the process of dynamic image transcoding by implementing in a fully SW configuration. As described in Section 3.2 (Figure 4), the coding (or compression) process takes up most of a CPU execution time during dynamic image transcoding, whereas the configuration process only takes 5 % of the CPU time. In addition, the configuration process requires more flexible controls (adaptability) to monitor the associated conditions and to decide the optimal algorithm and its configuration. Therefore, we can conclude that the configuration process can be categorized as a more control-oriented task (optimal for a SW mapping), while the coding process as a more computation-oriented task (optimal for a HW mapping). Next, we studied the coding process in detail in order to further explore the HW/SW mapping. We examined the main components of two image data preparation techniques such as simple image transcoding using JPEG2000 and our proposed dynamic image transcoding, as shown in Figure 8. We first found that the Discrete Wavelet Transformation (DWT) step is a common step and can be shared across two different algorithms without changing any configuration, whereas the quantization and encoding steps are different or need different configurations across the algorithms. We then found that the DWT step accounts for more than 50 % of the computation requirements across the algorithms. Therefore, we can conclude that the DWT step is the most computation-intensive process across the image compression processes and is the best candidate to obtain a significant performance improvement by mapping it to HW. In contrast, the quantization and encoding steps are different across the algorithms and requires more flexibility for dedicated algorithm, thereby, can be categorized as a more control-intensive process, leading to a SW mapping.

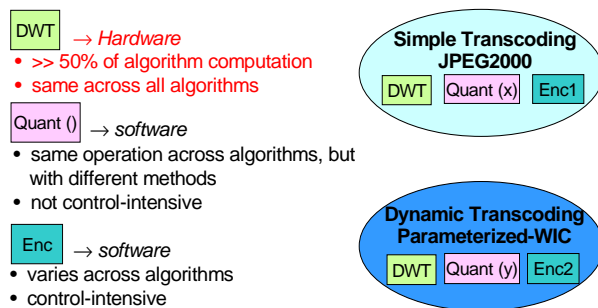


Figure 8. Illustration of mapping process of algorithm tasks into configurable HW/SW architecture

We also ensure adaptability for the case of dynamic image transcoding in the DWT hardware by creating a “parameterized HW” DWT block. Parameterized HW refers to a HW accelerator which can accept configuration parameters from outside (via bus

interface) and configures itself to execute the task based on the configuration parameters accepted. In the dynamic image transcoding process, these configuration parameters are given from the configuration task which is mapped into SW. In this manner, the functionality of DWT hardware block can be dynamically modified.

Note that our proposed configurable HW/SW architecture can be applied to multiple server architectures such as distributed or parallel server architecture by simple deployment.

4.2 Impact of Configurable HW/SW Architecture on Server Performance

In this section, we discuss the effects of using our configurable HW/SW architecture on the server performance in a data server. The top of Figure 9 illustrates the total service latency using our proposed HW/SW architecture. Note that compared to the total service latencies shown in Figure 6, significant latency savings are achieved using our proposed HW/SW architecture, including a server latency reduction of at least 30x. The bottom of Figure 9 demonstrates the increase in server capacity achieved with our configurable HW/SW architecture. The x-axis represents the average requests per second and the y-axis represents the average server latency, including service wait and server processing time. Using 1 second as the server latency cutoff for server overloading, we see that a simple image transcoding solution can handle an average of 0.1 requests per second, a dynamic software image transcoding solution 0.3 requests per second, while our configurable HW/SW architecture can handle 2.2 requests per second. This demonstrates that by adopting our configurable HW/SW architecture, a data server can increase the server capacity by 22x compared to the simple image transcoding case, and 7x over the dynamic software image transcoding case.

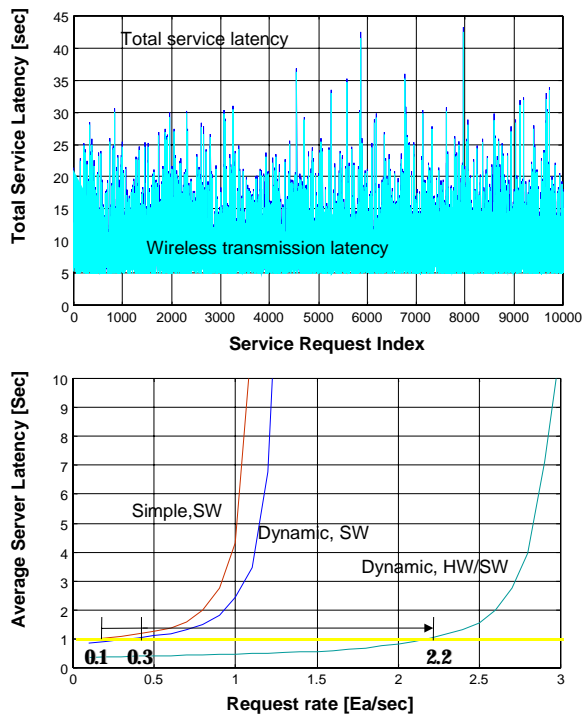


Figure 9. Impact of configurable HW/SW architecture on total service latency and server capacity (server performance)

Similarly, Figure 10 shows the server cost (as defined in section 3.2) required to meet a given average service request rate for three different architectural configurations (simple software, dynamic software image transcoding, and dynamic with configurable HW/SW.) Note that comparing to a fully software dynamic image transcoding case, using our configurable HW/SW architecture can reduce the required number of servers by 3 times to satisfy an average of 5 incoming service requests per second. This means we can drastically reduce the server cost by adding a small hardware block to implement DWT process, while still enabling the flexibility required to implement dynamic image transcoding technique.

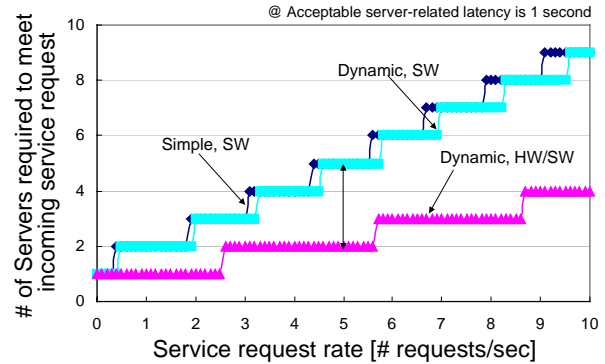


Figure 10. Impact of different architectures on server cost with different average request rates

5 CONCLUSION

Future deployment of wireless multimedia data services will require very customized/personalized services, creating very high workloads on a wireless data server. This paper presents a potential solution to the emerging problem by demonstrating a dynamic image transcoding methodology, coupled with a configurable HW/SW architecture, can be used for addressing server-side performance issues. The proposed configurable HW/SW architecture combined with the dynamic image transcoding technique can not only increase the server performance dramatically, but also reduces the cost of a data server, thereby enabling content-rich active image data services over the wireless networks at affordable cost. In the future, we will develop a complete algorithm which can address both wireless network-side issues as well as server-side issues simultaneously.

REFERENCES

- [1] The European Telecommunications Standards Institute (ETSI), <http://www.etsi.org>.
- [2] The 3rd Generation of Partnership Project (3GPP), <http://www.3gpp.org>.
- [3] R. Mohan, J.R. Smith, and C.S. Li, "Adapting multimedia Internet content for universal access", in *IEEE Transactions on Multimedia*, vol.1, no.1, pp.104-114, March 1999.
- [4] A. Ortega, F. Carignano, S. Ayer, and M. Vetterli, "Soft caching: Web cache management techniques for images," *Workshop on Multimedia Signal Processing in IEEE Signal Processing Society*, Jun. 1997.
- [5] B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, and K.R. Walker, "Agile Application-aware adaptation for mobility", *Proc. of the 16th ACM Symposium on Operating Systems and Principles*, Oct. 1997.

-
- [6] S. Chandra, A. Gehani, C.S. Ellis, and A. Vahdat, "Transcoding Characteristics of Web Images", *Conference on Multimedia Computing and Networking 2001, MMCN'01*, Jan. 2001.
- [7] Z. Lei and N.D. Georganas, "Context-based Media Adaptation in Pervasive Computing", in *Proc. of Canadian Conference on Electrical and Computer Engineering (CCECE)*, Toronto, May 2001.
- [8] D.G. Lee, S. Dey, "Adaptive and Energy Efficient Wavelet Image Compression For Mobile Multimedia Data Services", in *Proc. of IEEE International Conference on Communication*, Apr. 2002.
- [9] D.G. Lee, D. Panigrahi, S. Dey, "Network-aware Image Data Shaping for Low-Latency and Energy-efficient Data Services over the Palm Wireless Network", in *Proc. of World Wireless Congress*, May 2003.
- [10] T.W. Bickmore and B.N. Schilit, "Digerter: Device-independent Access to the World Wide Web", in *Proc. of the 6th World Wide Web Conference*, pp. 655-663, April 1997.
- [11] A. Fox, S.D.Gribble, E.A. Brewer, and E. Amir, "Adapting to network and client variability via on-demand dynamic distillation", *ACM SIGPLAN Notices* 31, vol. 9, pp 160-170, September 1996.
- [12] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V.Perret, and J. Rubas, "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing", in *IEEE Transactions on Personal Communications*, vol.5, issue 6,pp.8-17, Dec. 1998.
- [13] P.F. Joseph, A. Fisher, and G. Desoli, "Custom-Fit Processors: Letting Applications Define Architectures", in *Proc. of the 29th IEEE/ACM International Symposium on Microarchitecture*, 1996.
- [14] C. Ebeling, D.C. Cronquist, and P. Franklin, "RaPiD - Reconfigurable Pipelined Datapath", in *the 6th International Workshop on Field-Programmable Logic and Applications*, 1996.
- [15] R.D. Wittig and P. Chow, "OneChip: An FPGA Processor With Reconfigurable Logic", in *IEEE Symposium on FPGAs for Custom Computing Machines*, April, 1996.
- [16] Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J. Stockwood, "Hardware-Software Co-Design of Embedded Reconfigurable Architectures", in *Proc. of 37th Design Automation Conference*, June 2000.
- [17] G. K. Wallace, "The JPEG still picture compression standard", in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, June 1996.
- [18] O. K. Al-Shaykh, "JPEG-2000: A new still image compression standard", *Thirty-Second Asilomar Conference on Signals Systems and Computers*, vol. 1, pp. 99-103, 1998.
- [19] Bellsouth wireless data network services, <http://www.bellsouthwd.com/solutions/personalsolutions.html#palm>.
- [20] "Palm VII white paper", <http://www.palm.com/pr/palmvii/7whitepaper.pdf>, accessed December 2000.
- [21] S. Deng, "Empirical model of WWW document arrivals at access link", in *Proc. of the 1996 IEEE International Conference on Communication*, Jun. 1996.
- [22] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating user-perceived quality into web server design", in *9th International World Wide Web conference*, May 2000.