WebKit 모바일 웹 브라우저의 성능 향상을 위한 기법 연구

김정길* 정회원

A Study of High Performance WebKit Mobile Web Browser

Cheong Ghil Kim* *Regular Members*

요 약

스마트폰의 급속한 보급 확산에 따라 모바일 기기에서 웹 브라우저는 주요한 기능으로 자리 잡고 있으며 사용자는 모바일 기기에서 PC 수준의 고속화된 성능을 기대하는 현실이다. 웹키트(WebKit)는 구글의 안드로이드(Android) 플랫폼에 사용되고 있는 웹 브라우저를 만드는 데 기반을 제공하는 오픈 소스 응용 프로그램 프레임워크이다. 본 논문에서는 WebKit 라이브러리의 연산의 병렬처리를 통한 성능 향상 기법을 제안하였다. 제안된 병렬처리 기법은 JPEG 라이브러리에 적용되었으며, 성능 검증은 PC 기반의 실험을 통하여 이기종 컴퓨팅 기반의 모바일 임베디드 시스템 환경에서의 예측 방법을 사용하였다. 실험 결과는 제안된 병렬화 기법이 이기종 컴퓨팅 환경의 모바일 임베디드 기기로의 적용을 통한 성능 향상 가능성을 보여주었다.

Key Words : web browser; parallel programming; heterogeneous computing; Android; WebKit.

ABSTRACT

As the growing popularity of smartphones, mobile web browsing has become one of the most important and popular applications in mobile devices. Furthermore, it is clear that the demand for PC-like full browser performance on mobile devices is increasing greatly. WebKit is an open source web browser engine adopted by Google Android. This paper proposed a technique of increasing the performance of WebKit by paralleling its libraries. This method was applied to JPEG library and the performance evaluation was conducted in PC environment. The results was used to estimate the performance prediction on multi-core mobile embedded architecture and to show the feasibility of the proposed method to estimate the performance gain on heterogeneous multi-core embedded architecture.

I. Introduction

Nowadays, mobile computing is experiencing an unprecedented rise and the mobile industry already talks about 50 billion interconnected devices in 2020 [1]. In accordance with growing popularity of smartphones, mobile web browsing has become one of the most important and popular applications in smartphones. However, web browsers were designed for a PC application at the beginning; therefore they have to be reconsidered with great care for optimization on mobile devices having less computing resource and more power constraints than PCs.

Generally, webpage processing is known as CPU-bound work, and on the same wireless network the load time of a web page is 9x slower on the handheld [2]. However, the problem is that the demand for PC-like full browser performance on mobile devices is increasing greatly.

The important movement of future CPU architecture in mobile devices is that multi-core CPUs have become the dominant alternative solution for improving its performance due to power constraints, and dedicated accelerator processors have been integrated into one chip [3]. This movement already introduced dual core mobile devices and it is expected that mobile devices with up to 8 parallel hardware contexts will be introduced in the near future. Therefore, the importance of parallel computing techniques incorporating multiple processing cores and other acceleration technologies are increasing.

※ 이 본 논문은 2011년도 남서울대학교 교내연구비 지원에 의하여 연구되었음.
*남서울대학교 컴퓨터학과 (cgkim@nsu.ac.kr),
접수일자 : 2012년 3월 17일, 수정완료일자 : 2012년 3월 28일, 최종게재확정일자 : 2012년 6월 19일

To improve browser performance, several approaches have been introduced on the work of Berkeley Parallelism Lab [4], Their works are mostly concentrated on CSS selector matching, layout solving, and font rendering based on hardware parallelism [2]. In this paper, we exploit parallelism in processing WebKit [5] JPEG library by programming DCT with OpenCL and estimate the feasibility of possible performance improvement on mobile platforms.

This paper is structured as follows. Section 2 overviews the structure of WebKit and its parallel approach. Section 3 introduces our paralleling DCT for multi-core CPUs and GPGPUs, and the integration of it with WebKit JPEG library. Section 4 shows the simulation results with feasibility study. Finally, Section 5 covers the conclusion.

Ⅱ. Background

At the beginning, the function of web browsers was just rendering hyper-linked documents; and then JavaScript was introduced to enable scripting of simple animations and content transitions by dynamically modifying the document [2]. WebKit is an open source web browser engine and well-suited for mobile devices. Therefore, it is used as the default browser for mobile operating systems: iOS, Android, BlackBerry Tablet OS, and webOS. WebKit provides both layout and JavaScript engines for many systems. In general, browsers are large and complex with over 5 million lines of code.

Fig. 1 shows the overall data flow in web browser [2]. Loading an HTML page sets off a cascade of events: the page is scanned, parsed, and compiled into a document object model (DOM), an abstract syntax tree of the document. Content referenced by URLs is fetched and added to the DOM tree. As the content necessary to display the page becomes available, the page layout is (incrementally) solved and drawn to the screen. After the initial page load, scripts respond to events generated by user input and server messages, typically modifying the DOM. This may, in turn, cause the page layout to be recomputed and redrawn [2].

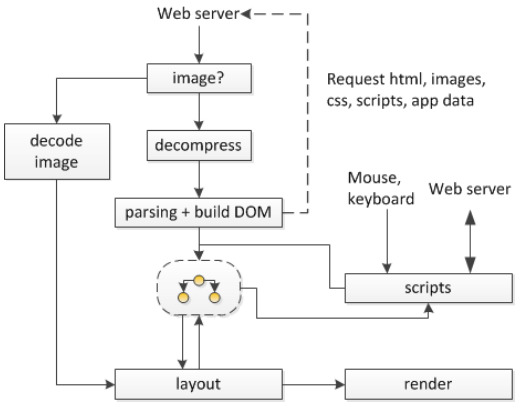

Fig. 1. Data flow of web browser

In the figure, we can understand the information on processing flow and functional components. The parallel web browser research [6] tries to enhance inefficiency of current web browser by exploiting parallelism in most core functional components like parsing, scripting, and layout [2]. As pointed out in [4], other parallel approaches on libraries of WebKit can be realized greatly when taking advantage of current modern CPU architecture.

Currently, the performance of modern mobile devices are increasing greatly; Especially, the big change comes form mobile CPU architecture. Single-core performance scaling is coming to an end because fundamental laws of physics limit further performance gains from uniprocessor architectures [3]. Therefore, all future computing platforms will provide multiple cores. The good news for embedded systems is that multi-core processors are more energy-efficient.

Under the new hardware architecture, there are several approaches on accelerating web browsers, sometimes using parallel languages such as OpenCL (Open Computing Language) [8] - an open standard for parallel programming of heterogeneous systems and WebCL - heterogeneous parallel computing in HTML5 web browsers. The major motivation is that the web browser is a CPU-intensive program. Especially on mobile devices, web pages load too slowly, expending significant time in processing a document's appearance.

Ⅲ. Accelerating Web Browser

To accelerate web browser on heterogeneous

multi-core architecture, it may be difficult to parallel every possible step of web browser engine because web browser execution pipeline causes dependency among each part of browser engine such as lexer, parser, layout, and so on. Furthermore, some of them are inherently sequential so it is nearly impossible to parallel [6].

In this paper, we tried to parallel WebKit JEPG library which could be one of computationally intensive modules of browser engine. As shown in Fig. 1, image decoding and rendering could slow down any web page loading time when there are too many or heavy images because the latency ratio is getting longer as the increased images overheads shown in Fig. 2. This figure shows the latency ratios of loading time of the web page with images of different size and number. Therefore, it could be a solution of improving WebKit performance by paralleling JPEG decoding library. For this purpose, this paper takes 2D DCT (two dimensional discrete cosine transform) algorithm, the most computational intensive part of JPEG decoding.

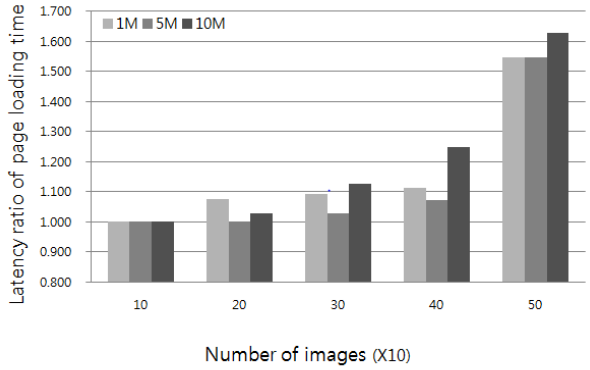

Fig. 2. Data flow of web browser

2D DCT can be described as a transform from a 2D matrix of pixels to that of spatial frequency information. The transformed matrix contains many small values or zero entries, such that the compression of such data using standard techniques becomes very straightforward. For an input matrix x(m, n) and an output matrix z(k, l) with {0 ≤ m, n, k, l ≤ N-1}, the forward N × N 2D DCT is defined as

$$z(k,l) = \frac{2}{N}\alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\chi(m,n)\cos\frac{(2m+1)\pi k}{2N}cos\frac{(2+1)\pi l}{2N} \quad (1)$$

where

Equation (1) can be rewritten in matrix form as

$$Z = AXA^T \quad (2)$$

where X is the source pixel (spatial domain) data, Z is the DCT output coefficients (frequency domain), and A is an orthogonal matrix defined as

$$\alpha(u,v) = \sqrt{\frac{2}{N}}\alpha(u)\cos\frac{(2v+1)\pi u}{2N} \quad (3)$$

There are many fast 2D DCT algorithms already studied. We implemented several algorithms and estimated its runtime on multi-core CPU and GPU environments. Experiments show that data parallelism can be fully exploited on CPU and GPU architecture [8,9]. After that we integrated GPU empowered DCT with WebKit JPEG library. Data structure and procedure routines in JPEG library are quite optimized well on sequential processor because of performance issues. It computes DCT one row of blocks in image each step and it is not very good idea on multi-processor programming. Especially GPU has huge data transfer overhead so it is recommended to deliver all data to GPU.

Ⅳ. Simulation Result

본 Future heterogeneous embedded architecture may consist of ARM multi-core CPU and other dedicated processors including GPU for cost, power-consumption, and performance. For feasibility simulation, we first took a high level approach. Because there is no openCL enabled embedded device, we have to predict possible performance achievement. Our experimental environment with three different configuration of desktop PC, single-core embedded hardware, and multi-core embedded hardware are summarized in Table 1.

The simulation results of OpenCL DCT execution time are shown in Table 2. Because there is no openCL support for embedded hardware, we have to predict execution time of openCL DCT benchmark. Desktop GPU showed 5x - 10x faster performance than embedded GPU. If we apply this fact to predict execution time of openCL benchmark, dual-core embedded hardware's execution time will be 0.135 sec, which is 185x faster. Measured execution time

shows that desktop GPU is 5x-10x faster than embedded GPUs. If we apply 5 times slower performance on S5PV310, we can guess that execution time will be about 0.012 seconds.

Table 1. Hardware configuration

Desktop PC			
Hardware		Software	
CPU	Intel Xeon Quad Core 3.07GHz	OS	Ubuntu Linux 10.10 Linux Kernel : 2.6.35
GPU	NVIDIA Quadro 60, 1GHz, 96 Cuda cores Maximum Power Consumption : 40W GPU Memory Specs: 1GB DDR3, 128-bit, 25.6GB/s		
RAM	4GB	File System	EXT4
Single-core embedded hardware			
Hardware		Software	
CPU	Cortex-A8 Single-Core	OS	Android 2.2 Froyo
GPU	PowerVR SGX 540		
RAM	More than 128MB	File System	NFS
Multi-core embedded hardware			
Hardware		Software	
CPU	Cortex-A9 Dual-core	OS	Android 2.3 Ginzerbread
GPU	Mali 400		
RAM	More than 128MB	File System	FAT/EXT4 (SD/MMC)

Table 2. OpenCL DCT execution time

	Serial	OpenCL	OpenCL Benchmark
Desktop PC	5.625	0.0027	0.0074
S5PV210	59.356	N/A	0.071
S5PV310	24.998	N/A	0.04

Fig. 2 shows simulation results with various sizes of images in only DCT computation part of JPEG library and the performance achieved 6x - 12x speed up. The algorithm used in JPEG library is fast DCT algorithm using row and column scanning; therefore speed up we make is reasonable when we compare it to previous DCT experimental result. Here, the second parallel DCT runtime measurement is from graphic hardware profiler that is more accurate and measure time spent only on GPU.

Fig. 2 shows simulation results with various sizes of images in only DCT computation part of JPEG library and the performance achieved 6x - 12x speed up. The algorithm used in JPEG library is fast DCT algorithm using row and column scanning; therefore, the speed up we make is reasonable when we compare it to previous DCT experimental result. Here, the second parallel DCT runtime measurement is from graphic hardware profiler that is more accurate and measure time spent only on GPU.

	Sequential DCT			Parallel DCT(H/W counter)				Parallel DCT(profiler)			
	CbCr	Y	Total	CbCr	Y	Total	Speedup	CbCr	Y	Total	Speedup
256x256	163	1409	3041	816	658	1474	2.06	303	173	476	6.39
512x512	6579	5449	12028	1714	1458	3172	3.79	758	523	1281	9.39
1200x1200	37412	30500	67912	4894	4670	9564	7.10	2936	2636	5572	12.19

Fig. 2. Simulations on various sizes of images

According to profile result [10] shown in Table 3, data transfer spends about 3x more time than kernel computation does. In desktop environment, communication between CPU and GPU is done via PCI channel which is quite slower than accessing main memory. In embedded system, CPU and GPU are usually on same ship and share memory so that we can mitigate data transfer overhead.

Table 3. Data transfer

Total Satge	256x256	512x512	1200x1200
clCreateBuffer	101	230	530
memcpy (temporary buffer)	49	171	542
clSerKernelArg	19	27	61
clEnqueueWriteBuffer(memcpyHtoD)	963	2766	7572
clEnqueueNDRangeKernel(CbCr space)	808	1766	5018
clEnqueueNDRangeKernel(Y space)	694	1462	4692
clEnqueueReadBuffer	2158	3547	12602
encode_mcu(writing DCT result)	1881	4636	20892
Total time	6673	14605	51909

Also we can speed up current version of JPEG library on desktop hardware environment which has large amount of data transfer overhead. Current graphic hardware support 2 simultaneous data transfer. When we apply pipeline data transfer, we can reduce data communication overhead to half. We pipeline data write, kernel computation and data read. Like our case, if kernel computation is smaller than data transfer, we can add more computation for free like quantization step in JPEG library. Also CPU computing resource can be exploited during GPU computation.

V. Conclusion

This paper explores the performance model of parallel programming of WebKit JPEG for accelerating web browser by measuring execution time. For our simulation, we implemented the fast DCT algorithm on various computing environments. The evaluation results show the feasibility of the proposed method to estimate the performance gain on heterogeneous embedded architecture.

참 고 문 헌

[1] http://news.cnet.com/8301-13506_3-20051610-17 .html.

[2] L. A. Meyerovich and R. Bodik, "Fast and parallel webpage layout," Proceedings of the 19th international conference on World wide web, pp. 711-720, 2010.

[3] C. G. Kim, D. H. Lee, and J. Kim, "Optimizing Image Processing on Multi-core CPUs with Intel Parallel Programming Technologies," Multimedia Tools and Applications, DOI: 10.1007/s11042-011-0906-y, Nov. 2011.

[4] Leo Meyerovich, "Rethinking Browser Performance, USENIX, login, vol 34, no. 4, pp. 14-20, Aug. 2009.

[5] http://www.webkit.org

[6] C. G. Jones, R. Liu, L. Meyerovich, K. Asanovic, and R. Bodik, "Parallelizing the Web Browser," HotPar'09 Proceedings of the First USENIX conference on Hot topics in parallelism, pp.7-7 2009.

[7] C. Lemuet, J. Sampson, J. Francois, and N. Jouppi, "The potential energy efficiency of vector acceleration," Proceedings of the 2006 ACM/IEEE conference on Supercomputing, pp. 0-1, 2006.

[8] C. G. Kim and Y. S. Choi, "A High Performance Parallel DCT with OpenCL on Heterogeneous Computing Environment," Multimedia Tools and Applications, DOI 10.1007/s11042-012-1028-x . Feb. 2012.

[9] J. G. Hong, J. S. Wook, C. G. Kim, and B. Burgstaller "Accelerating 2D DCT in Multi-core and Many-core Environments," In Proc. Of the 35th Conference of Korea Information Processing Society, May 2011.

[10] J. Leskela, J. Nikula, and M. Salmela, "OpenCL embedded profile prototype in mobile device," IEEE Workshop on Signal Processing Systems, 2009. SiPS 2009. pp. 279-284, Oct. 2009.

저자

김 정 길 (Cheong Ghil Kim) **정회원**

- 1987년 : Univ. of Redlands, U.S.A. 컴퓨터과학 학사졸업
- 2003년 : 연세대학교 컴퓨터과학 공학석사
- 2006년 : 연세대학교 컴퓨터과학 공학박사
- 2006년~2007년 : 연세대학교 컴퓨터과학과 박사후연구원
- 2007년~2008년 : 연세대학교 컴퓨터과학과 연구교수
- 2008년~현재 : 남서울대학교 컴퓨터학과 조교수

<관심분야> : 멀티미디어 임베디드 시스템, 이기종 컴퓨팅, 모바일 AR