논문 13-38B-03-01

한국통신학회논문지 '13-03 Vol.38B No.03
http://dx.doi.org/10.7840/kics.2013.38B.3.163

Heterogeneous 멀티 코어 환경의 Thick Client에서 VDI 성능 최적화를 위한 혼합 병렬 처리 기법 연구

김 명 섭◆, 허 의 남°

VDI Performance Optimization with Hybrid Parallel Processing in Thick Client System under Heterogeneous Multi-Core Environment

Myeong-seob Kim◆ Eui-Nam Huh°

요 약

최근 HD급 동영상이나 3D 어플리케이션과 같은 이전보다 저사양, 모바일 단말에서는 구동하기 힘든 프로그램들에 대한 이용 요구가 확대되면서 처리해야 할 콘텐츠 데이터들이 고용량화 되고 있다. 클라우드 기반의 VDI(Virtual Desktop Infrastructure) 서비스는 이를 처리하기 위해 효율적인 데이터 처리 능력이 필요해졌으며 QoE(Quality of Experience) 보장을 위한 성능 개선 연구가 이슈가 되고 있다. 본 논문에서는 H/W 성능이 향상되어 CPU와 GPU를 탑재한 Thick Client기반의 3가지 Thick-Thin간 VDI 자원 공유 및 위임이 가능한 VDI 서비스에 대해 제안하며, VDI 서비스 성능의 개선을 위해 CPU와 GPU가 혼합된 Heterogeneous 멀티코어 환경에서 CPU와 GPU 병렬 처리 기법인 OpenMP와 CUDA를 활용하여 VDI 서비스 최적화 방안을 제안하고 기존의 VDI와 비교한 성능을 거론한다.

Key Words : Thin Client, Cloud Computing, OpenMP, CUDA, Hybrid Parallel Processing

ABSTRACT

Recently, the requirement of processing High Definition (HD) video or 3D application on low,mobile devices has been expanded and content data has been increased as well. It is becoming a major issue in Cloud computing where a Virtual Desktop Infrastructure (VDI) Service needs efficient data processing ability to provide Quality of Experience (QoE) in Cloud computing. In this paper, we propose three kind of Thick-Thin VDI Service which can share and delegate VDI service based on Thick Client using CPU and GPU. Furthermore, we propose and discuss the VDI Service Optimization Method in mixed CPU and GPU Heterogeneous Environment using CPU Parallel Processing OpenMP and GPU Parallel Processing CUDA.

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No. 2012-0006418).
◆ 주저자 : 경희대학교 컴퓨터공학과 클라우드컴퓨팅 & 보안 연구실, kms1205@khu.ac.kr, 학생회원
° 교신저자 : 경희대학교 컴퓨터공학과 클라우드컴퓨팅 & 보안 연구실, johnhuh@khu.ac.kr, 정회원
논문번호 : KICS2013-01-004, 접수일자 : 2013년 01월 03일, 최종논문접수일자 : 2013년 3월 4일

Ⅰ. 서 론

iOS, Android Windows 8과 같은 모바일 플랫폼들이 발전하면서 최근 모바일 기기들은 PC를 대체하고 있으며, 자신의 모바일 단말을 사용하는 BYOD(Bring Your Own Device) 근무 환경은 언제 어디에서든 자신이 가지고 있는 모바일 단말을 사용하여 일을 할 수 있는 Smart Working을 가능케 하였다.

이러한 모바일 기기의 급격한 발전은 클라우드와 연계되어 모바일 VDI[11] 서비스로 이어졌으며 이에 모바일 환경에 적합한 VDI 기술들이 연구되고 있다. 대표적인 VDI 기술들로는 RDP나 VNC, RemoteFX, PCoIP와 같은 기술들이 있다[4,5].

최근 HD급 동영상이나 3D 어플리케이션과 같은 이전보다 Thin Client 단말에서는 구동하기 힘든 프로그램들에 대한 이용 요구가 확대되고 있으며 처리해야 할 콘텐츠 데이터들이 고용량화 되고 있다. 이를 처리하기 위해 클라우드는 효율적인 데이터 처리 능력이 필요해졌으며 VDI 서비스 QoE(Quality of Experience) 보장을 위한 성능 개선에 대한 연구가 이슈가 되고 있다.

본 논문에서는 Thin Client보다 H/W 자원이 좋아진 Thick Client를 바탕으로 VDI Agent 서비스를 제안하고 VDI 서비스의 QoE 보장을 위해 혼합 병렬 처리를 이용한 VDI 최적화 기술을 제안하고 그 성능에 대해 분석할 것이다. 2장에서는 관련연구로써 효율적인 VDI 그래픽 처리를 위한 GPU 가상화 기술과 고품질 클라우드 서비스 중 하나인 클라우드 게임에 관해 설명하며 3장에서는 Heterogeneous CPU, GPU 환경을 갖춘 Thick Client 기반의 VDI Agent 시나리오를 제안하면서 Thick-Thin Client 모델을 정립할 것이다. 그 후 4장에서는 VDI 서비스 성능 개선을 위한 혼합 병렬 처리 적용 방법에 대해 소개한 후 5장 성능 평가를 통해 성능 개선 사항에 대해 분석하여 6장에서 결론을 맺는다.

Ⅱ. 관 련 연 구

GPU 가속기능을 활용한 가상 데스크탑 환경(VDI)은 최근 은행, 기업, 공공기관에서 스마트워크 환경으로서 활용되고 있으며, 모바일 환경에서도 빠르게 확산되고 있다. 또한, VDI 외에 고성능을 요구하는 게임이나 CAD와 같은 그래픽 분야에서도 VM-to-GPU consolidation ratio를 개선하여 경제성을 높이기 위해 GPU 가상화를 필요로 하고 있다[6,7].

클라우드 게임은 대표적인 고품질 클라우드 서비스로 대규모의 클라우드 노드를 구성하고, 게임 콘솔이나 고성능 PC에서 수행하던 컴퓨터 게임을 각 클라우드 노드 상에서 실행하는 기술이다. 노드에서 구성된 게임 영상은 유무선 네트워크를 통하여 스트리밍 방식으로 사용자 단말기에 전송되어 재생되며, 사용자 단말기에서 발생하는 사용자 입력은 클라우드 노드에 즉시 전송되어 게임에 반영된다.

클라우드 게임은 추가적인 장비 설치 없이 모바일 단말기나 디지털 TV에서 고품질 3D 애플리케이션을 사용할 수 있으며 다수의 플랫폼에서 서비스할 수 있으므로 개발 비용이 절감된다. 요구사항으로는 입력에 대한 높은 반응성과 영상 품질과 이를 유지하는 것이며, 마지막으로 적은 비용으로 시스템을 구축하는 것이다[3].

Ⅲ. Thick Client VDI 서비스

3.1. Thick-Thin Client VDI 서비스 시나리오

H/W기술이 급속한 발전을 보임에 따라 데스크톱, 스마트폰, 타블렛 PC와 같은 단말들의 사양이 높아지고 있다. 특히 단말의 성능에 가장 영향을 많이 끼치는 CPU 및 메모리는 발전 주기가 단축되고 있어 해를 거듭할수록 보다 풍부한 자원(CPU, Memory, HDD 등)을 갖춘 Thick Client가 되었다.

반면 최소한의 자원으로 운영 되는 Thin Client는 일반 가정에 보급되는 데스크톱과 달리 스마트폰이나 타블렛 PC에 탑재되는 저사양의 프로세서를 사용하여 Thin Client 데스크톱을 구성하고 있다. 본 논문에서는 이런 Thick-Thin Client를 기반으로 한 VDI 서비스를 제안한다.

제안하는 VDI 서비스는 Thick-Thin Client들이 네트워크를 구성하여 클라우드 VDI 서비스를 공유 또는 위임하게 된다. 공유와 자원 위임이란 점에서 DLNA(Digital Living Network Alliance)와 유사하다. DLNA에서 사용자는 홈 네트워크를 구성하고 구성된 홈 네트워크 안의 단말들끼리의 파일 공유를 통해 동영상이나 사진, 음악을 재생할 수 있다[8]. Thick-Thin Client VDI 서비스의 구성요소인 VDI Server, Agent, Client의 역할은 다음과 같다.

- VDI Server: VDI 서비스를 위한 Application

을 수행하고 화면을 인코딩 하여 VDI Agent 나 VDI Client로 전송한다.

- VDI Agent: H/W 성능이 높은 Thick Client 단말로 Cloud의 VDI Server에서 제공하는 서비스를 제공받는다. 또한 수신한 서비스를 VDI Client에 공유 또는 위임 할 수 있다.
- VDI Client: 저 사양 Thin Client의 단말로 VDI 서비스를 제공받는다.

Thick-Thin Client VDI 서비스는 3가지 모델인 1) Direct VDI Client 구조, 2) VM VDI Agent 구조, 3) Physical VDI Agent로 나눌 수 있으며 각 모델의 특징들을 설명한다.

3.2. Direct VDI Client 구조

Direct VDI Client구조는 VDI Agent 없이 모든 단말 디바이스들이 VDI Client가 되어 VDI Server와 1:1 혹은 1:N 으로 직접 통신하는 모델이다. 그림 1은 Direct VDI Client 구조를 나타낸다.

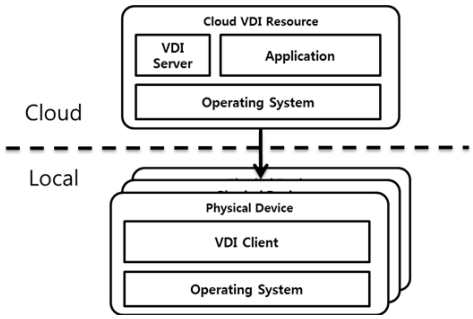

그림 1. Direct VDI Client 구조도
Fig. 1. Direct VDI Client Architecture

Direct VDI Client 구조의 특징으로는 VDI Server와 Client간 직접적인 통신을 통해 최소한의 딜레이로 서비스가 가능하다는 점과 VDI Agent, VDI Client 생성시 비용이 발생하지 않는 점이 있다. 하지만 VDI Client를 관리함에 있어 VDI Server는 추가적인 부하가 발생하게 되며 VDI 특성상 서로 다른 VDI Client에서 입력 이벤트가 발생 시 VDI Server에서 혼잡이 발생할 수 있다.

3.3. VM VDI Agent 구조

VM VDI Agent 구조는 클라우드에서 생성한 VM을 VDI Agent로 사용하는 모델이다. Direct VDI Client 구조와 달리 생성된 VM이 VDI Agent 역할을 수행하게 되며 VDI Client와 1:1 또는 1:N 통신을 하며 VDI Client에 전달될 VDI 서비스 공유와 위임에 대한 권한을 갖는다.

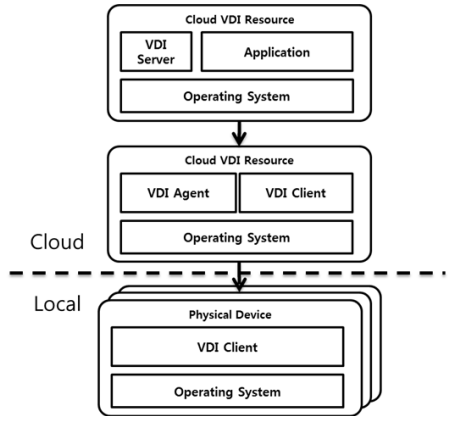

그림 2. VM VDI Agent 구조도
Fig. 2. VM VDI Agent Architecture

VM VDI Agent 구조는 VM 생성시마다 비용이 발생하여 클라우드에 자원 소모를 일으킬 수 있다. 또한 Server-Agent-Client 계층 구조로 인한 추가적인 딜레이가 발생 하며 Agent의 성능이VM 성능에 의존적이게 된다. 그림 2는 VM VDI Agent 구조를 나타낸다.

3.4. Physical VDI Agent 구조

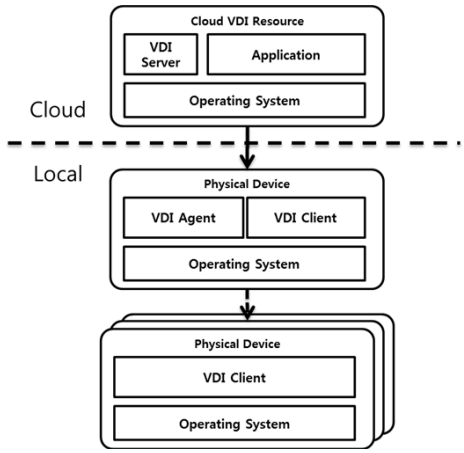

그림 3 Physical VDI Agent 구조도
Fig. 3. Physical VDI Agent Architecture

Physical VDI Agent 구조는 VM VDI Agent 구조와 달리 사용자 단말 중 하나가 VDI Agent 역할을 수행하는 모델이다. VDI Agent는 앞서 설명한 대로 VDI 서비스에 대한 공유와 위임에 대한 권한을 가지며, VDI Client를 관리한다. 또한 VM VDI Agent 구조와 달리 VDI Agent를 구성함에 있어 추가적인 비용이 발생하진 않지만 계층 구조로 인한 딜레이가 발생 하며 VDI Agent가 되는 단말의 H/W 성능에 의존적이게 된다.

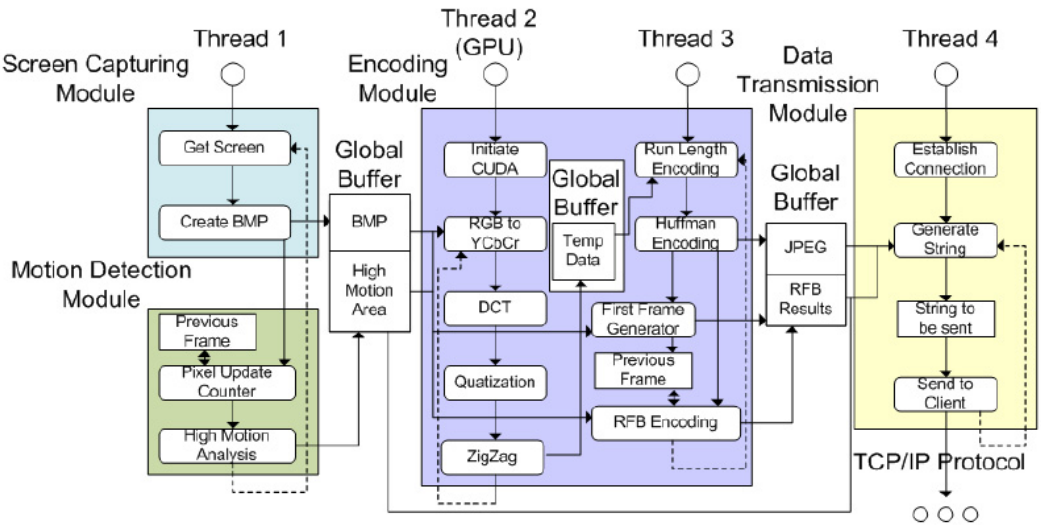

그림 4. VDI Agent 프로그램 [1]
Fig. 4. VDI Agent Program

3.5. Thick-Thin Client VDI 서비스 환경 구축

본 논문에서는 제안한 3가지 모델 중 추가적인 비용이 발생하지 않으며 VDI Server의 혼잡이 없는 Physical VDI Agent 모델을 바탕으로 다음 장에서 혼합 병렬 처리를 이용한 성능 최적화를 진행할 것이다. 4장에서는 본 모델을 바탕으로 테스트 환경을 구축하고 VDI Agent 성능 향상을 위한 혼합 병렬 처리 기법에 대해 서술할 것이다.

Ⅳ. 혼합 병렬 처리 기법 적용

4.1. Application Type 분류

본 절에서는VDI Agent서비스 성능 개선에 앞서 Thick/Thin Client VDI 서비스시 가능한 Application의 종류를 사용자의 입력과 출력되는 화면의 변화를 기준으로 다음과 같이 4가지로 분류하였다.

- $A_{browser}$: 사용자 입력과 화면 변화가 모두 낮음 (ex. 책, 인터넷)
- A_{word}: 사용자 입력은 많지만 화면 변환 부분이 작음 (ex. 워드 작업)
- A_{video}: 사용자 입력은 적지만 화면 변화가 큼 (ex. 동영상)
- A_{game}: 사용자 입력과 화면 변화가 모두 큼 (ex. 게임)

4.2. VDI Agent 프로그램 구조

그림 4는 본 연구에서 사용한 VDI Agent 프로그램 구조도이다. Thin Client 서비스의 핵심 중 하나는 클라이언트에서 네트워크를 통해 화면을 효율적으로 출력할 수 있도록 화면 이미지를 압축하는 것이다. 이를 위해 사용되는 코덱은 네트워크 대역폭 요구 사항을 줄이기 위한 효율적인 압축 및 원격 콘텐츠로 효율적인 상호 작용을 활성화하기 위해 낮은 지연 시간을 요구한다. 본 연구에서 사용되는 VDI Agent 프로그램은 전통적인 Thin Client 프로토콜인 RFB와 복잡한 그래픽 출력을 보낼 MJPEG 코덱을 혼합한 하이브리드 원격 디스플레이 프로토콜을 사용하였으며, MJPEG의 성능을 향상하기 위해 GPU를 사용하였다[1]. 또한 총 4개의 Thread를 사용하여 작업을 분할하고 기본적인 병렬 처리가 가능하도록 하였다. 각 Thread에 할당된 일은 다음과 같다.

- Thread1: Screen Capturing, Motion Detection
- Thread2: GPU MJPEG Encoding
- Thread3: RFB Encoding, Combine Data
- Thread4: Data Transmission

본 논문에서는 VDI Agent 프로그램의 최적화를 위해 Multi Core CPU 병렬처리를 위한 OpenMP[9]기법와 Multi Core GPU 병렬처리를 위한 CUDA 기법을 사용하여 VDI Agent에 적합한 혼합 병렬 처리 방법을 제시하고 개선된 성능을 분석할 것이다.

4.3. 대조군 VDI Agent 테스트 결과

표 1. Simulation 환경
Table 1. Simulation Environment

Name	VDI Server	VDI Agent	VDI Client
CPU	2.40GHz (Quad Core)	3.30GHz (Quad Core)	2.93Ghz (Quad Core)
GPU		GeForce GTX 460 2대	GeForce 9300 GS
OS	Android, Windows	Windows	Windows
Network	Ethernet	Ethernet	Ethernet
Display Screen	640 * 480		

본 절에서는 VDI Agent 프로그램의 기본적인 성능을 분석하기 위해 Application type에 따라 대조군 VDI Agent 성능 평가를 시도하였다. 테스트 환경은 3장에서 제안하였던 Physical VDI Agent 모델로 구성하기 위해 2대의 PC와 1대의 서버로 표 3과 같은 환경으로 구축하였다.

그림 5의 그래프는 최적화하기 전의 대조군 VDI Agent의 각 Thread별 소모 시간을 테스트 한 결과이다. 그래프 분석 결과 Screen Capturing, Motion Detection작업을 수행하는 Thread1에서 가장 많은 시간을 소모하는 것을 알 수 있었다. 또한 A_{video}의 경우 상대적으로 A_{word}와 에 비해 GPU MJPEG Encoding 작업을 수행하는 Thread2의 가 수행시간이 짧음을 볼 수 있었다.

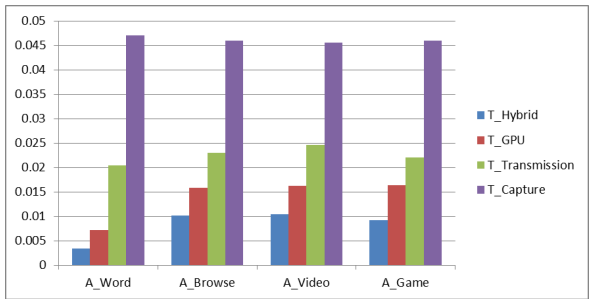

그림 5. 대조군VDI Agent 실험 결과 그래프
Fig. 5. VDI Agent Test Result Graph

4.4. 혼합 병렬 처리 기법

그림 6은 VDI Agent 프로그램의 Encoding 과정을 도식화 한 것이다. 그래픽 업데이트를 위한 데이터는 그래픽 카드 하드웨어 프레임 버퍼에서 후킹된다. 모션 디텍터 인코더 모듈은 비디오 인코딩에 대해 화면 변환이 많은 하이 모션(High motion) 부분을 결정한다.

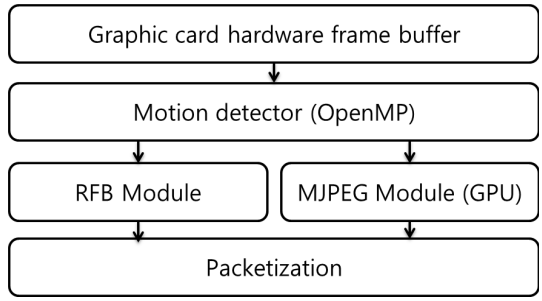

그림 6. VDI Agent Encoding 도식도
Fig. 6. VDI Agent Encoding Flow Diagram

화면 변환이 적은 로우 모션(Low motion)의 경우 서버에서 적은 자원을 소모하기 때문에 RFB 모듈을 사용한다. 반면 하이 모션의 경우 MJPEG 모듈을 통해 실시간으로 인코딩 한다. MPEG 인코딩과 비교하여 MJPEG 인코딩은 적은 컴퓨팅 리소스를 소모하므로 보다 효율적이라 할 수 있으며 어플리케이션이 실행 중인 동안에도 유연한 압축률을 제공할 수 있다는 장점이 있다.

전송 프로토콜은 RTP, RTCP와 TCP를 혼합하여 사용한다. MJPEG을 전송하기 위해 RTP, RTCP 프로토콜을 사용하였으며 스트리밍이 필요하지 않은 경우 기존 Thin Client 서비스와 같은 TCP 프로토콜을 사용하여 전송한다.

대조군 VDI Agent 테스트에서 확인하였듯이 Screen Capturing과 Motion Detection 작업을 수행하는 Thread1이 가장 많은 시간이 소모되는 것을 알 수 있었다. 다음 절에서는 Multi Core CPU 병렬 처리 언어인 OpenMP를 Thread1에 적용하여 성능 향상을 위한 방법을 제시할 것이다.

4.5. OpenMP 기법 적용

VDI Agent Screen Capturing Thread는 그림7 좌측과 같은 순서로 MJPEG과 RFB 하이브리드 인코딩을 위한 작업들을 수행한다. 그래픽 데이터는 하드웨어 프레임 버퍼에서 후킹되며 화면의 하이 모션 디텍팅을 위해 전 화면 영상의 그래픽 데이터

와 현 화면 영상의 그래픽 데이터를 비교한다. 생성된 차영상과 현 화면 영상을 Raw Data로 파일로 저장한 후 MJPEG 인코딩을 하기 위한 YCbCr 색상으로 변환한다.

OpenMP 기법 적용은 데이터가 반복적으로 수행되거나 독립적으로 수행되는 각각의 작업을 분석하여 적용하였다. 화면 비교와 Raw Data를 YCbCr 영상으로 바꾸는 작업은 데이터가 반복적인 잡업을 수행하게 되므로 CPU 코어의 개수만큼 Thread를 생성하여 작업을 병렬화 시켰다. 또한 차영상을 저장하는 작업과 현 화면을 저장하는 작업은 상호간에 데이터가 독립적이므로 OpenMP를 사용하여 두 작업을 병렬화 시켜 동시에 수행되도록 하였다.

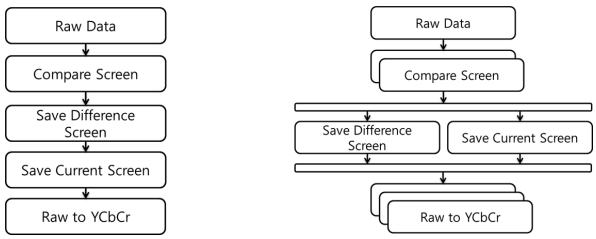

그림 7. OpenMP 적용 전(좌) 적용 후(우)
Fig. 7. OpenMP before applying (left) after applying (Right)

4.6. OpenMP 성능 예측

암달의 법칙은 컴퓨터 시스템의 일부를 개선할 때 전체적으로 얼마만큼의 최대 성능 향상이 있는지 계산하는 데 사용된다. 암달의 법칙에 따르면 어떤 시스템을 개선하여 P만큼의 부분에서 S 만큼의 성능 향상이 있을 때 전체 시스템에서 최대 성능 향상은 다음과 같다. 다음은 암달의 법칙을 나타낸다.

$$\frac{1}{(1-P)+\frac{P}{S}}$$

대조군 VDI Agent Screen Capturing Thread의 프로세서 백분율은 아래와 같다. 이 결과를 바탕으로 OpenMP 성능을 예측하면 1.19 배로 VDI Agent Screen Capturing Thread에서 약 19% 의 성능 향상을 예측할 수 있다.

- Compare Screen : 0.3%
- Save Difference Screen : 4.7%
- Save Current Screen : 4.5%
- Raw to YCbCr : 19.7%

4.7. GPU 병렬 처리 기법

GPU의 강력한 성능은 그래픽 처리뿐만 아니라 일반적인 데이터를 처리하는 데에도 사용 할 수 있는데 이를 GPGPU (General Purpose Graphic Processing Unit)라고 한다. CUDA는 그래픽 전용 API 함수들이 GPGPU의 일반적인 용도로 사용하기에 무리가 있어 효율적인 GPGPU를 지원하기 위해 개발되었다.

CUDA는 그래픽 카드를 이용한 GPGPU의 통합 개발 환경을 제공하는 것을 목적으로 하며, GPU를 이용한 범용적인 프로그램 개발을 할 수 있도록 프로그램 모델, 프로그램 언어, 컴파일러, 라이브러리, 디버거, 프로파일러를 제공하는 통합 환경을 구축하였다.

스케일러블 링크 인터페이스(Scalable Link Interface, SLI[12])는 그래픽 칩셋 제조 업체인 엔비디아에서 여러 대의 그래픽 카드를 장착하기 위해 만든 기술이다. GPU 병렬 처리 기법에서는 SLI 기법과 Multi CUDA 기법을 사용하여 VDI Agent 성능을 측정할 것이다.

4.8. CUDA 기법 적용

VDI Agent 프로그램은 JPEG 압축 알고리즘의 퍼포먼스 향상을 위하여 GPU 처리 기술인 CUDA[10]를 사용하여 성능 향상을 시도하였다. JPEG 압축은 컴퓨팅 집약적이므로 CPU에서 인코딩을 사용할 경우 긴 압축 시간으로 인해 지연이 발생 할 수 있어 CPU 성능이 떨어질 수 있다.

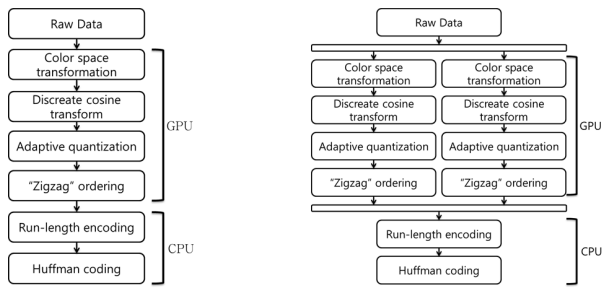

그림 8. Multi CUDA 적용 전(좌) 적용 후(우)
Fig. 8. OpenMP before applying (left) after applying (Right)

그림 8은 JPEG 압축 과정에서 GPU와 CPU의 흐름을 보여주는 플로우 다이어그램이다. JPEG 압축에는 데이터 병렬 처리가 가능한 색 공간 변환,

이산 코사인 변환, 양자화 및 지그재그 인코딩 과정을 거치게 되는데 이 과정들을 GPU에 할당하여 수행하였다. 반면 Run-length 인코딩과 호프만 코딩의 경우 GPU에 할당 시 성능이 크게 떨어지게 되므로 CPU에 할당하여 수행하였다.

본 연구에서는 VDI Agent의 2개의 그래픽 카드를 효율적으로 이용하기 위한 Multi CUDA와 SLI 기법을 사용하여 성능을 테스트 할 것이다. 데이터를 자동으로 병렬화 시켜 수행하는 SLI와는 달리 Multi CUDA를 적용하기 위해서 각각의 그래픽 카드를 관리해야 하므로 OpenMP를 사용하여 하나의 Thread를 더 생성하여 GPU Encoding 부분에 할당하였다.

V. 성능 평가

5.1. App type별 OpenMP 기법 성능 평가

본 절에서는OpenMP를 사용하여 개선한 VDI Agent의 성능 평가를 위해 4가지 타입의 대조군 실험과 동일한 어플리케이션을 사용하여 Screen Capture 시간을 비교하였다. 테스트 환경은 VDI 대조군 실험과 같은 환경에서 진행되었다.

테스트를 통해 OpenMP를 사용할 경우 그러지 않을 경우와 비교하여 약 5~15%가량의 성능이 향상되었음을 볼 수 있었다. Application별로 분리해 보면 화면 변화가 적은 Word나 Browse Application에서 OpenMP가 15%가량의 성능이 개선되었음을 볼 수 있었으며, 화면 변화가 큰 Game이나 Video Application의 경우 약 5%의 성능이 개선되었음을 볼 수 있었다.

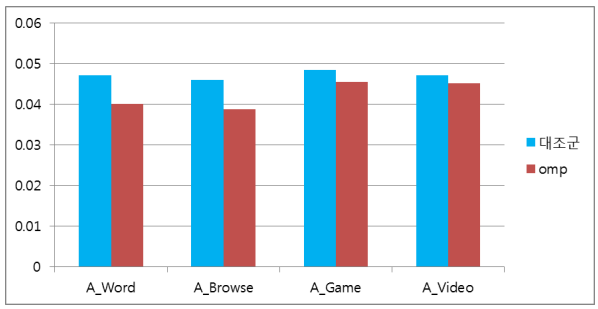

그림 9. Application별 OpenMP 적용 성능 평가
Fig. 9. OpenMP Performance Evaluation with Application type

다음으로는 Application 종류에 따라 성능의 차이가 큰 원인을 파악하기 위해 프로그램 시작부터 100번째 화면 Capturing까지의 데이터를 시간 순서대로 나열하여 비교하였다. 그래프는 시간 순서별 Video와 Word 작업에서의 Capturing 시간을 나타낸다.

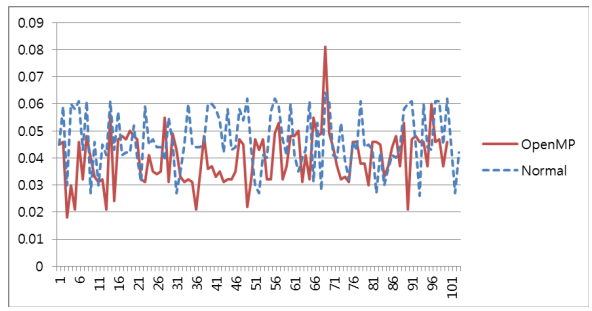

그림 10. Word Application 시간 순서별 Capturing 시간 그래프
Fig. 10. Word Application Capturing time Graph

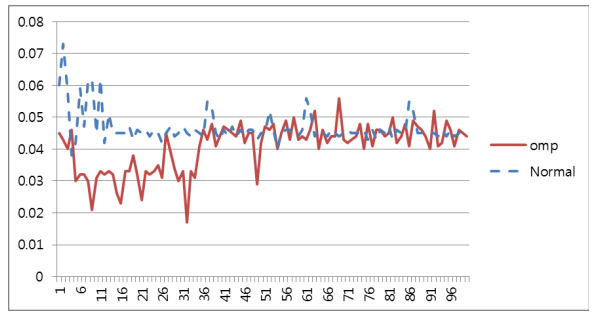

그림 11. Video Application 시간 순서별 Capturing 시간 그래프
Fig. 11. Video Application Capturing time Graph

Word Application의 경우 지속적으로 대조군 대비 OpenMP 에서의 Capturing의 성능이 향상되어있음을 볼 수 있었다. Video Application의 경우 약 40번째 Capturing까지 대조군보다 OpenMP를 적용하였을 경우 약 20% 가량의 성능이 향상되어 있음을 볼 수 있었다. 하지만 시간이 지나면서 OpenMP 모델의 성능이 떨어지면서 대조군과 비슷한 딜레이 시간을 유지하는 것을 볼 수 있었다. 이는 화면 전환이 높아 처리해야 할 데이터보다 CPU 자원이 부족하여 더 이상의 성능을 유지하지 못한 것으로 분석된다.

5.2. GPU 기법 성능 평가

본 절에서는 SLI와 Multi-CUDA를 사용하여 개선한 VDI Agent의 성능 평가를 위해 대조군 실험과 동일한 비디오 어플리케이션을 사용하여 GPU Encoding 시간을 비교하였다. 테스트 환경은 VDI

대조군 실험과 같은 환경이며 화면의 크기를 바꿔 가면서 진행하였다.

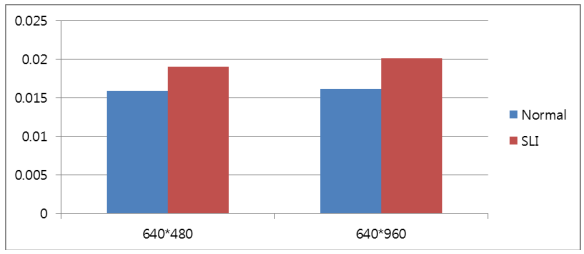

그림 12. Video 화면 크기에 따른 SLI 성능 평가 그래프
Fig. 12. SLI performance evaluation graph according to the size of the Video Screen

그림 12는 Video Application 수행시 GPU Encoding 시간을 측정한 것이다. SLI를 적용 시 성능이 좋아질 것이라는 예상과는 달리 성능이 약 20%가량 떨어지는 것을 확인할 수 있었다. 이는 VDI Agent GPU Encoding 알고리즘이 SLI를 사용하기에 적합하지 않은 것으로 분석된다. 화면의 크기가 1280*960인 경우 VDI Agent 프로그램 버퍼의 한계를 넘어서 데이터를 측정할 수 없었다.

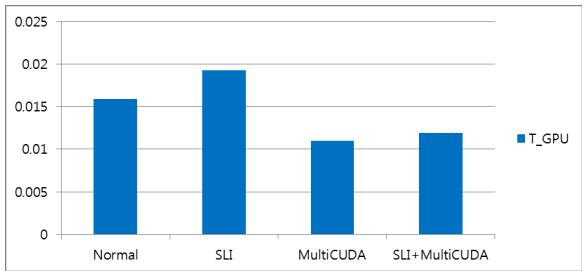

그림 13. Video Application GPU Encoding 성능 평가 그래프
Fig 13. GPU Encoding performance rating graph

그림 13은 같은 화면 크기의 Video Application 사용 시 대조군, SLI기법, Multi-CUDA, SLI와 Multi-CUDA동시에 적용한 경우 GPU Encoding 시간을 비교한 그래프이다. Multi-CUDA와 SLI와 Multi-CUDA 혼합 형식 모두 GPU Encoding에서 성능이 향상되었으며 Multi-CUDA의 경우 약 30% 성능이 향상된 것을 볼 수 있었다.

Ⅵ. 결론 및 추후 연구

본 논문에서는 고품질 클라우드 VDI 서비스의 QoE를 만족시키기 위해 Thick-Thin Client VDI 서비스를 위한 3가지 모델인 Direct VDI Client, VM VDI Agent, Physical VDI Agent를 제시하고 VDI Agent의 최적화를 위해 다수의 CPU를 사용하는 Multi Core CPU환경에서 공유 메모리 구조 병렬처리 언어인 OpenMP와 다수의 그래픽 카드를 사용하는 Multi GPU 환경에서 병렬 처리를 위한 Multi CUDA와 SLI를 사용한 혼합 병렬 처리 기법을 제안하고 성능을 평가하였다.

제안한 GPU 기법의 경우 화면의 크기에 상관없이 SLI 기법을 적용하였을 경우 경우 약 20% 가량 성능이 떨어지는 것을 확인할 수 있었다. 하지만 Multi CUDA를 사용할 경우 약 30% 가량 성능이 향상되었음을 볼 수 있었다.

OpenMP 기법의 경우 화면 변화가 작은 Word Processing이나 Bowser Application에서는 성능이 15%가량 향상됨을 확인할 수 있었지만 Video나 Game Application의 경우 VDI Agent 프로그램이 시작 후 일정 시간 동안만 성능이 향상 된 것을 확인 할 수 있었다.

일정 시간 후에는 혼합 병렬 처리를 적용하기 전과 차이가 없어지는 것을 확인할 수 있었으며 이에 추가적으로 Video와 Game Application에 적합한 최적화 병렬화 알고리즘을 연구하여 보완할 계획이다.

References

[1] B. Song, W. Tang, and E.N. Huh, "Novel isolation technology and remote display protocol for mobile thin client computing," in *Proc. ACM ICUIMC 2012*, pp. 41-47, Kuala Lumpur, Malaysia, Feb 2012

[3] C. G. Lim, S.S Kim, K.I. Kim, J.H. Won, and C.J. Park. "Technology trends of cloud computing-based game streaming," *Electron. and Telecommun. Trends. 26* vol 1. pp. 47-56, Feb. 2011.

[4] J.H Kim, I.H. Kim, C.W. Kim, and Y.I Eom. "Technology trends of mobile virtualization," *Korea Inf. Sci. Soc. review*. vol. 28 no. 6, June. 2010.

[5] W.O. Kwon and H.Y. Kim, "Technology trends of high performance VDI protocol," *NIPA Weekly Technology trends*. vol. 1546, May. 2012.

[6] K.W. Hong, J.W. Yoon, and W. Ryu, “Technology trends of game virtualization,” *NIPA Weekly Technology trends*. vol. 1588, Oct. 2012.

[7] nVIDIA Cloud computing. Retrieved Nov., 5, 2012. [Online] Available: http://www.nvidia.co.kr/object/cloud-computing-kr.html

[8] Wikipedia DLNA. Retrieved Sep., 26, 2012. [Online] Available: http://ko.wikipedia.org/wiki/DLNA

[9] OpenMp. OpenMP Retrieved Nov., 11, 2012. [Online] Available: http://openmp.org/wp/

[10] nVIDIA CUDA. Retrieved Nov., 19, 2012. [Online] Available: http://www.nvidia.com/object/cuda_home_new.html

[11] Wikipedia Desktop virtualization. Retrieved Sep., 30, 2012. [Online] Available: http://en.wikipedia.org/wiki/Desktop_virtualization#VDI

[12] nVIDIA SLI. Retrieved Nov., 25, 2012. [Online] Available: http://www.nvidia.co.kr/object/sli-technology-overview-kr.html

[13] M.S. Kim, J.Y. Park, S.J. Lee, J.H. Ku, and E. Huh, “VDI performance optimization with OpenMP based on multi core environment thick client system” in *Proc. KICS winter 2013 domestic conf.* Yongpyong Resort, pp. 324-325. Jan. 2013

[14] S.Y. Lee, Y.R. Shin, M.S. Kim, A.Y. Son, J.S. Bong, and E.N. Huh. “Parallel processing research for VDI optimization on multi GPU environment”, in *Proc. KICS winter 2013 domestic conf.* Yongpyong Resort, pp. 318-319. Jan. 2013

김 명 섭 (Myeong-seob Kim)

2007~2011년 2월 경희대학교 컴퓨터공학과 졸업(학사)
2011~2013년 2월 경희대학교 일반대학원 컴퓨터공학과 졸업(석사)
현재 경희대학교 컴퓨터공학과 박사과정
<관심분야> 클라우드 컴퓨팅, 모바일 클라우드, etc

허 의 남 (Eui-nam Huh)

2002. The Ohio University 전산학과 졸업(박사)
2005~2011년 경희대학교 컴퓨터공학과 교수
현재 경희대학교 컴퓨터공학과 교수
<관심분야> 클라우드/그리드 컴퓨팅, 센서 네트워크, 네트워크 보안, 모바일 컴퓨팅, etc