DSV 기반 자원 고가용성을 위해 GPU를 이용한 신속한 자동 확장 기법

박부광*, 김현우*, 변휘림*, 허윤아*, 송은하**, 정영식*
*동국대학교 멀티미디어공학과
**원광대학교 교양교육대학
e-mail:pbg0517@dongguk.edu

Rapid Auto-scaling Mechanism using GPU for Resource High Availability based on DSV

Boo-Kwang Park*, Hyun-Woo Kim*, HwiRim Byun*, Yoon-A Heo*,
Eun-Ha Song**, Young-Sik Jeong*
*Dept. of Multimedia Engineering, Dongguk University
**Dept. of Liberal Arts, WonKwang University

요 약

IT 기술의 진보적 발전에 따라 클라우드 컴퓨팅 분야 연구들이 활발히 진행되고 있다. 클라우드 컴퓨팅은 가상화 기술을 이용하여 크게 인프라, 플랫폼, 소프트웨어 관점으로 나뉘어 사용자에게 다양한 서비스를 제공한다. 가상화 기술 중에 Desktop Storage Virtualization (DSV)은 분산된 레거시 데스크탑으로 구성되어 있기 때문에 비가용 상태 시간별 클러스터링 및 사용자 요청에 따른 자동 확장이 매우 중요시된다. 본 논문에서는 GPU의 many-core를 이용하여 분산된 데스크탑의 성능 상태 분석 및 자동 확장을 위해 스레드별로 호스트를 매핑하고 병렬적으로 처리하는 Rapid Auto Scaling Mechanism (RASM)을 제안한다.

1. 서론

IT의 급격한 발전에 따라 수 많은 스마트 디바이스가 생성되고 이러한 스마트 디바이스의 소형화 및 저조한 컴퓨팅 성능 보완을 위해 클라우드 컴퓨팅 서비스를 이용한다. 이 중 수많은 디바이스에서 생성되는 데이터 저장을 위해 STorage as a Service를 사용한다. 이러한 서비스 제공을 위해 클라우드는 가상화 기술을 이용한다. 가상화 기술 중에 Desktop Storage Virtualization (DSV)는 기존의 많은 데스크탑 PC를 가상화하여 클라우드 서비스가 가능하다. DSV는 분산된 수많은 데스크탑 환경으로 구성되기 때문에 비가용 상태 시간별 클러스터링 및 자동 확장이 매우 중요시된다. 기존의 클러스터링 및 자동 확장을 위한 많은 연구 및 알고리즘이 개발되어 있지만, DSV 환경에서 성능 효율을 위한 연구가 미흡하다[1, 2, 3, 4, 5].

이에 본 논문에서는 DSV 환경에서 클러스터링 및 자동 확장의 보다 빠른 처리를 위해 GPU의 many-core를 이용하는 Rapid Auto Scaling Mechanism (RASM)을 제안한다.

2. 관련 연구

기존의 자동 확장 연구로 자동으로 작업 정보 및 요구 성능 기반 자원 인스턴스를 계산하는 cloud auto-scaling 기법을 제안하였다[4]. 다른 연구로 on-demand 기반 가상 자원 할당을 위한 server-side auto-scaling 기법을 제안하였다[5]. 가상 자원을 요청한 실시간 작업에 따라 확장/축소가 가능하지만, DSV 환경에서의 분산된 컴퓨팅 자원의 통합과 실시간 정보 분석에는 적합하지 않다.

다른 연구에서는 가상 머신에 대한 할당 및 할당 해제의 불필요한 처리 시간이 발생됨에 따라 이를 위한 다양한 메커니즘을 소개하였다[6]. 그러나 자동 확장을 위한 고려사항으로 사용자 자원 요구 측면 고려하기 때문에 DSV 환경과 같은 독립된 데스크탑 수행 및 분산된 데스크탑 자원 통합 기반 자원 제공에 적합하지 않다.

본 논문에서는 GPU를 이용하여 빅 데이터 저장을 위한 데스크탑 스토리지 선정을 위해 신속하게 처리한다.

3. RASM 스킴

본 논문에서 제안하는 RASM은 자바 기반으로 만들어졌으며 기존 DSV 환경에 적용하여 Auto-scaling 처리 지연시간 최소화를 위해 GPU 모드를 제공한다.

GPU 모드를 사용하기 위해서는 몇 가지 고려 사항이 존재하는 데, 다음과 같다.

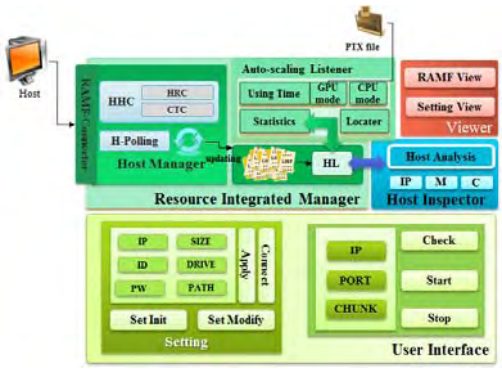
- 먼저, GPU를 이용하기 때문에 GPU를 사용할 수 있는지에 대한 동작 가능 여부를 체크해야 한다.
- 동작이 가능한 지에 대한 필수적 검사로는 그래픽 카드가 CUDA를 지원해주는 지에 대한 체크가 있다.

* 이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2014R1A1A2053564).

- 다음으로 Parallel Thread Execution (PTX) 파일을 실행 시킬 수 있는 지에 대한 여부 검사가 끝나면, 비로소 GPU 모드를 사용할 수 있도록 활성화 된다.
- 동작이 가능한 경우에는 선택적 GPU 모드가 활성화되고, 그렇지 않은 경우에는 비활성화된다.

RASM에서의 GPU 모드를 이용한 데스크탑 성능 분석을 위한 개수는 그래픽 카드에 의존적이다. 각 데스크탑의 정보 처리를 병렬적으로 실행하는 데 있어서 최대의 효율을 이끌기 위해서는 기본적으로 GPU의 각 스레드는 두 개 이상의 데스크탑 정보에 대응되지 않아야 한다. 즉, 각 스레드는 최대 하나의 데스크탑 정보 분석을 위한 계산을 수행해야 한다. 이외에도 뱅크 충돌, 레지스터, 로컬 메모리, 공유 메모리, 글로벌 메모리 등을 고려해야 한다.

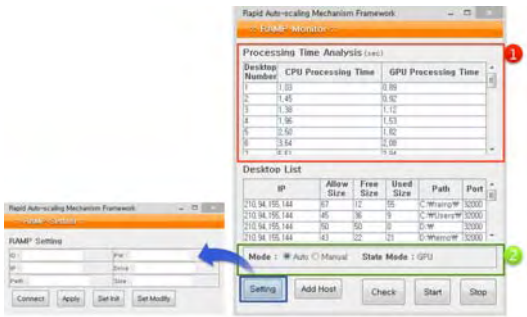
4. RASF 설계



(그림 1) RASF 구조

본 논문에서 제안하는 RASM이 동작하는 프레임워크를 Rapid Auto Scaling Framework (RASF)라 한다. RASF는 기본적으로 분산된 데스크탑의 스토리지 통합 정보를 입력 받기 위한 User Interface, 연결된 호스트의 정보를 분석 및 검사하는 Host Inspector, 연결된 데스크탑 자원을 관리하고 auto-scaling 기능을 내재하는 Resource Integrated Manager, 사용자에게 동작 상태 및 설정 상태를 시각화하는 Viewer로 나뉜다.

5. RASF 구현



(그림 2) RASF 실행 화면

그림 2는 RASF의 자원 자동 확장 동작화면을 나타낸다. 최초 실행시 자동 확장을 위한 처리 시간을 누적한다. 이후에 연결되는 데스크탑 수의 증가에 따라 CPU와 GPU처리 시간을 통계한다. 자동 확장을 수행하는 경우에 통계 자료 기반 연결된 데스크탑 수를 체크하고 CPU 및 GPU 모드를 통해 처리한다. 그림 2내의 ①은 데스크탑 수가 증가될 때 CPU 및 GPU의 처리 시간의 통계를 나타낸다. 그림 2내의 ②는 ①을 통해 CPU 및 GPU 모드로 동작되는 화면을 나타낸다.

6. 결론

본 논문에서는 Desktop Storage Virtualization 기반 클러스터링 및 자동 확장을 위한 지연처리시간의 최소화를 위해 GPU를 이용하여 병렬 처리하는 Rapid Auto Scaling Mechanism (RASM)을 제안하였다. RASM은 분산된 수많은 데스크탑인 Desktop Storage Node (DSN)를 관리하는 Desktop Manager Node (DMN)가 GPU를 이용하여 각 DSN을 GPU의 스레드에 매핑하고 병렬 처리를 수행하였다. 이를 통해 적은 시간에 클러스터링 및 자동 확장을 수행하였다. 이러한 수행은 스토리지 사용자에게 높은 QoS를 제공한다.

참고문헌

[1] Luis M. Vaquero, Luis Rodero-Merino, Rajkumar Buyya, “Dynamically scaling applications in the cloud,” ACM SIGCOMM Computer Communication Review, Vol. 41, No. 1, pp. 45-52, Jan. 2011.

[2] Jong Hyuk Park, Hyun-Woo Kim, Young-Sik Jeong, “Efficiency Sustainability Resource Visual Simulator for Clustered Desktop Virtualization Based on Cloud Infrastructure,” Sustainability, Vol. 6, No. 11, pp. 8079-8091, Nov. 2014.

[3] Cristóbal A. Navarro, Nancy Hitschfeld-Kahler and Luis Mateu, “A Survey on Parallel Computing and its Application in Data-Parallel Problems Using GPU Architectures,” Communications in Computational Physics, Vol. 15, No. 2, pp. 285-329, Jun. 2015.

[4] Ming Mao, Jie Li, Marty Humphrey, “Cloud Auto-scaling with Deadline and Budget Constraints,” In proceedings of 11th IEEE/ACM International Conferenceon Grid Computing, Brussels, Belgium, 25-28, Oct. 2010, pp. 41-48.

[5] Young Woon Ahn, Albert M. K. Cheng, Jinsuk Baek, Minho Jo, Hsiao-Hwa Chen, “An Auto-Scaling Mechanism for Virtual Resources to Support Mobile, Pervasive, Real-Time Healthcare Applications in Cloud Computing,” IEEE Network, Vol. 27, No. 5, pp. 62-68, Sep. 2013.

[6] Gropesh Banker, Gayatri Jain, “A Literature Survey on Cloud AutoScaling Mechanisms,” International Journal of Engineering Development and Research, Vol. 2, No. 4, pp. 3811-3817, 2014.