KIPS Trans. Softw. and Data Eng.
Vol.8, No.1 pp.45~50 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2019.8.1.45

Resource Clustering Simulator for Desktop Virtualization Based on Intra Cloud

Hyun-Woo Kim†

ABSTRACT

With the gradual advancement of IT, passive work processes are automated and the overall quality of life has greatly improved. This is made possible by the formation of an organic topology between a wide variety of real-life smart devices. To serve these diverse smart devices, businesses or users are using the cloud. The services in the cloud are divided into Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). SaaS runs on PaaS, and PaaS runs on IaaS. Since IaaS is the basis of all services, an algorithm is required to operate virtualization resources efficiently. Among them, desktop resource virtualization is used for resource high availability of unused state time of existing desktop PC. Clustering of hierarchical structures is important for high availability of these resources. In addition, it is very important to select a suitable algorithm because many clustering algorithms are mainly used depending on the distribution ratio and environment of the desktop PC. If various attempts are made to find an algorithm suitable for desktop resource virtualization in an operating environment, a great deal of power, time, and manpower will be incurred. Therefore, this paper proposes a resource clustering simulator for cluster selection of desktop virtualization. This provides a clustering simulation to properly select clustering algorithms and apply elements in different environments of desktop PCs.

Keywords : Intra Cloud, Infrastructure as a Service, Resource Clustering Simulator

인트라 클라우드 기반 데스크탑 가상화를 위한 리소스 클러스터링 시뮬레이터

김 현 우†

요 약

IT의 점진적 진보에 따라 수동적인 작업 처리가 자동화되고 이를 통해 전반적인 삶의 질이 대폭 발전되었다. 이는 실생활에 접목된 다양하고 많은 스마트 디바이스간 유기적인 토폴로지가 형성됨으로써 가능하다. 이러한 다양한 스마트 디바이스에 서비스를 제공하기 위해서 기업 또는 사용자들은 클라우드를 이용하고 있다. 클라우드에서의 서비스는 크게 Infrastructure as a Service(IaaS), Platform as a Service(PaaS), Software as a Service(SaaS)로 나뉜다. SaaS는 PaaS 위에서 동작되고, PaaS는 IaaS 위에서 동작한다. 이와 같이 IaaS는 모든 서비스의 기반이기 때문에 가상화하는 자원을 효율적으로 운용하기 위한 알고리즘이 요구된다. 이 중에 데스크탑 자원 가상화는 기존 데스크탑 PC의 비가용 상태 시간의 자원 고가용성을 위해 사용된다. 이러한 자원의 고가용성을 위해서는 계층적 구조에 대한 클러스터링이 중요시된다. 또한 많은 클러스터링 알고리즘 중에서 데스크탑 PC의 분포율 및 환경에 따라 주로 사용되는 자원 비중이 다르기 때문에 적합한 알고리즘을 선정하는 것이 매우 중요하다. 만일 동작 환경의 데스크탑 자원 가상화에 적합한 알고리즘을 찾기 위해 다양한 시도를 한다면 이에 대한 전력적, 시간적, 인력에 대한 막대한 비용이 초래된다. 따라서 본 논문에서는 데스크탑 가상화의 클러스터 선정을 위한 리소스 클러스터링 시뮬레이터인 RCS를 제안한다. RCS에 클러스터 수, 호스트 수를 증가하여 동작하는 과정의 시각화 및 수행 시간을 비교 분석한다. 이를 통하여 데스크탑 PC들의 서로 다른 환경에서 클러스터링 알고리즘 선정 및 요소를 올바르게 적용할 수 있도록 클러스터링 시뮬레이션을 제공한다.

키워드 : 인트라 클라우드, 인프라스트럭처 서비스, 자원 클러스터링 시뮬레이터

1. 서 론

최근 IT의 진보적인 컴퓨팅 기술에 따라 많은 인력이 투입되던 수동적인 작업 처리가 자동화 되었다. 또한 다양한 스마트 디바이스(예를 들어, 디지털 카메라, 스마트 TV, 태블릿 PC, 스마트 폰) 등의 고성능화와 편리한 휴대성으로 여가시간 및 업무의 효율성이 증가되었다. 이는 실생활에 다양하고 많은 스마트 디바이스간 유기적인 토폴로지가 형성됨으로써 가능하다. 이러한 스마트 디바이스는 한정된 자원(예를 들어,

† 정 회 원 : 동국대학교 멀티미디어공학과 연구교수
Manuscript Received : September 27, 2018
Accepted : October 10, 2018
* Corresponding Author : Hyun-Woo Kim(hwkim@dongguk.edu)

CPU의 성능, 메모리의 크기, 스토리지의 크기)으로 인해 클라우드 서비스를 이용하고 있다. 일반적으로 클라우드는 서비스의 제공 형태에 따라 Infrastructure as a Service(IaaS), Platform as a Service(PaaS), Software as a Service(SaaS) 등으로 나뉜다[1-5]. 이러한 클라우드 서비스는 내부적으로 가상화를 이용하여 사용자에게 제공된다. 데스크탑 자원 가상화는 기존의 많은 데스크탑 PC를 가상화를 이용하여 클라우드 서비스가 가능하다. 이러한 데스크탑 PC 자원의 고가용성을 위해서는 계층적 구조를 위한 클러스터링이 매우 중요하다[3-9]. 이에 클러스터링 관점에 따라 Centroid-based Clustering[6], Distribution-based Clustering[7], Density-based Clustering[8], Connectivity based Clustering[9] 등의 많은 알고리즘이 개발되었지만 환경에 따라 최적화된 알고리즘을 선택하기가 매우 어렵다. 또한 클러스터링을 위한 요소의 정의 및 비율 등을 물리적 환경에 직접 테스트하기에는 많은 시간이 소요된다. 이러한 이유로 본 논문에서는 데스크탑 가상화의 클러스터 선정을 위한 효율적인 RCS(Resource Clustering Simulator)를 제안한다. RCS는 호스트의 효율적인 클러스터링을 위해 XML 기반의 호스트 정보를 읽어 들여 클러스터링 시뮬레이션을 제공한다. 클러스터링 동작과정을 2D 뷰 또는 3D 뷰로 시각화함으로써 클러스터링 알고리즘에 적용성 테스트가 가능하다. 또한 다양한 클러스터링 알고리즘을 시뮬레이션할 수 있도록 사용자가 추가할 수 있는 인터페이스를 제공하여 환경에 따른 능동적인 클러스터링 알고리즘 선택이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 자원 관리, 스케쥴링, 클러스터링 등의 시각화 및 시뮬레이터에 대한 연구를 살펴본다. 3장에서는 본 논문에서 제안하는 RCS의 시뮬레이션을 위한 2D, 3D 구성 방식을 설명한다. 4장에서는 RCS의 설계, 5장에서는 RCS의 구현 및 데스크탑 PC의 수에 따라 시뮬레이션 동작 속도를 평가한다. 마지막으로 6장에서는 전반적인 결론 요약 및 향후 연구로 구성되어 있다.

2. 관련 연구

자원 관리, 스케쥴링, 클러스터링의 시각화 및 시뮬레이터에 대한 연구를 살펴보면 다음과 같다.

GHSOM[10]은 유사한 제조부품을 클러스터링하고 이를 생산하기 위한 기계를 클러스터링하는 것으로 self-organizing map(SOM)기반 cellular manufacturing system에 대한 시각화 연구를 하였다. 시각화를 위해 growing hierarchical self-organizing map(GHSOM) 알고리즘 수행에 대한 수렴 기준, 계산 시간, 양자화 오차 등을 고려하였다. 이러한 고려 방법은 고정적이기 때문에 본 논문에서는 다양한 실험을 시각화하기 위한 사용자 설정 인터페이스를 제공한다.

VCS 시뮬레이터[11]는 윈도우 시스템에서만 동작 가능하며 추가적인 하드웨어가 필요하지 않다. 사용자의 클러스터를 구성, 서비스 그룹 및 자원 구성과 failover에 대해 시뮬레이션을 제공한다. 그러나 Command Line Interface(CLI) 기반의 인터페이스를 제공하고 텍스트 기반의 결과를 보여줌으로써 분석하는 데 어려움이 존재한다. 이에 본 논문에서는 Graphic User Interface(GUI) 기반의 인터페이스와 분석 용이를 위한 시각화를 제공한다.

GridSim[12, 13]은 스케줄링 알고리즘을 시뮬레이션 할 수 있는 Java 기반의 시뮬레이터이다. GridSim의 자원 엔티티 인터페이스는 멀티 프로세서인인 각 자원의 프로세서의 수, 프로세스 비용, 프로세싱 속도, 내부 프로세스 스케줄링 정책을 다르게 표현 가능하다. GridSim을 통해 가상 환경을 쉽게 시뮬레이션 할 수 있으나, 자원 분석에 필요한 다양한 변수를 조작하기 위해 사용자가 프로그램 소스코드를 작성해야 하고 이에 대한 결과를 차트 등으로 변화해야 하는 어려움이 존재한다. 따라서 본 논문에서는 GUI 기반의 자원 설정, 동작과정 및 결과에 대한 시각화를 제공한다.

ClusterSim[14]은 자바 기반의 병렬 이산 이벤트 시뮬레이션 도구이다. ClusterSim은 클러스터의 워크로드 및 시각적 모델링을 지원한다. 또한 시뮬레이터는 클러스터를 사용자가 지정하고 클러스터간에 Message Passing Interface(MPI), 병렬 작업 스케쥴링 알고리즘 등의 시뮬레이션이 가능하다. 그러나 이기종 또는 동종의 데스크탑 PC로 구성된 환경에서 클러스터를 선정하기 위한 기능이 없으며, 사용자가 클러스터를 수동적으로 지정해야 하는 번거로움이 있다. 이로 인해 클러스터 선정을 위한 알고리즘 동작 파악이 쉽지 않다.

CloudSim[15]은 클라우드 시스템 구성으로 데이터 센터, 가상 머신, 자원 프로비저닝 정책 등의 행동 모델링을 지원한다. 또한 사용자 정의 인터페이스를 통해 네트워크 간에 가상 머신의 할당이 가능하다. 그러나 결과가 텍스트 기반으로 도출되기 때문에 이를 문제의 원인 및 동작 과정 등을 파악하는데 어려움이 있다. 또한 데이터 센터의 구성에 있어서 데스크탑 자원 가상화에 대한 설정이 고려되지 않는다. 따라서 본 논문에서는 데스크탑 자원을 효율적으로 운영하기 위해 호스트 기반 클러스터링 동작과정의 시각화를 제공한다.

3. RCS 스킴

본 논문에서 제안하는 리소스 클러스터링 시뮬레이터인 RCS는 데스크탑 자원의 클러스터링을 위해 데스크탑 PC 정보를 Fig. 1과 같은 xml 형태로 입력 받는다. 또한 이질적인 데스크탑 PC의 클러스터링을 시뮬레이션하기 위해 임의의 데스크탑 정보를 생성하는 사용자 인터페이스도 제공한다.

RCS는 기본적인 클러스터링 요소로 distance와 performance로 나뉘며 상세 설명은 Table 1과 같다.

RCS는 2D 시뮬레이션을 위해서 호스트의 정보를 이용하여 기본적으로 x축은 distance(예를 들어, 홉수, 응답 시간 등), y축은 performance(예를 들어, CPU, 메모리, 스토리지의 크기)를 나타낸다. x축의 정규화는 클러스터링 시뮬레이션을 위한 호스트 중에 라우터의 홉수 또는 응답 시간을 기준으로

```
<?xml version="1.0"?>
- <host id="">
    <name/>
      <!-- desktop pc name -->
    - <performance>
      - <dynamic>
          <!-- dynamic performance factors -->
          <cpu/>
          <memory/>
          <storage/>
        </dynamic>
      - <static>
          <!-- static performance factors -->
          <cpu/>
          <memroy/>
          <storage/>
        </static>
    </performance>
  - <distance>
      <ping/>
      <request_time/>
    </distance>
</host>
```

Fig. 1. XML Scheme of Host in RCS

Table 1. RCS Clustering Elements

Factor	Description
Distance	- ping: time measurement by ping - request time: the time to connect individual hosts in which the RCS is operated and the time to respond to individual hosts in cases where time measurement by ping is impossible
Performance	- dynamic performance factors: idle CPU performance, size or residual memory and storage - static performance factors: performance of the basic CPU installed, maximum size of the memory and storage

측정값이 가장 높은 호스트인 $host_{MAXdistance}$를 정한다. 이를 기준으로 n번째 호스트인 $hostN_{distance}$는 Equation (1)을 통해 x축 좌표인 $hostN_{xvalue}$를 산출한다.

$$hostN_{xvaule} = \frac{hostN_{distance}}{host_{MAXdistance}} \times 100 \quad (1)$$

n개의 호스트를 Equation 1과 같이 수행함으로써 모든 호스트의 정규화된 x축 좌표값을 산출한다.

y축의 정규화는 호스트의 성능을 나타내는 CPU, 메모리, 스토리지별로 측정값이 가장 높은 호스트인 $host_{MAXcpu}$, $host_{MAXmemory}$, $host_{MAXstorage}$를 정한다. 다음으로 백분율을 통해 CPU의 비중을 나타내는 α, 메모리의 비중을 나타내는 β, 스토리지의 비중을 나타내는 γ를 정한다. 이 때, α, β, γ의 합은 100을 초과할 수 없으며, 합이 100이 되어야 한다. n번째 호스트의 성능인 는 Equation (2)를 통해 산출한다.

$$hostN_{performance} = \frac{hostN_{cpu}}{host_{MAXcpu}} \times \alpha + \frac{hostN_{memory}}{host_{MAXmemory}} \times \beta + \frac{hostN_{storage}}{host_{MAXstorage}} \times \gamma \quad (2)$$

이 때 $hostN_{yvalue}$는 $hostN_{performance}$와 같으며 n개의 호스트를 Equation (2)와 같이 수행함으로써 모든 호스트의 정규화된 y축 좌표값을 산출한다. RCS의 3D 시뮬레이션은 2D 시뮬레이션인 x축의 CPU, 메모리, 스토리지에 대한 성능 비율을 x축에 CPU와 메모리, z축에 스토리지, y축은 distance로 시각화한다.

4. RCS 설계

본 논문에서 제안하는 RCS는 시뮬레이션의 시각화 및 호스트 정보를 사용자로부터 입력 받는 User Interface, 사용자로부터 입력 받은 xml 형태의 호스트 정보를 분석하고 관리하는 Host Manager, 호스트 정보를 기반으로 수행할 클러스터링 알고리즘을 관리하는 CA Manager, Viewer에 시뮬레이션을 나타내기 위해 데이터를 가공하는 Coordinate Converter, 시뮬레이션 상태를 시각적으로 보여주는 Viewer로 구성되어 있다. RCS에 대한 기능적인 전체 구조도는 Fig. 2와 같다.

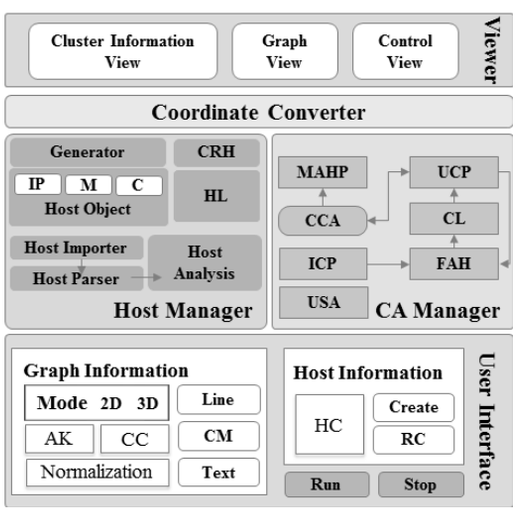

Fig. 2. RCS Architecture

User Interface는 시뮬레이션의 시각화를 제어하기 위한 graph information, 시뮬레이션 할 호스트를 설정하는 host information, 시뮬레이션 수행을 위한 run, 중지를 위한 stop으로 구성되어 있다. Graph information은 알고리즘을 선택하기 위한 AK(Algorithm Kind), 시뮬레이션 할 클러스터 수를 입력 받기 위한 CC(Cluster Count), 호스트를 그래프에 나타내기 위한 normalization으로 구성되어 있다. 또한 Mode를 통해 시뮬레이션 시각화에 따른 2D, 3D를 선택할 수 있다. Host information은 시뮬레이션 할 호스트의 수를 입력하는 HC(Host Count), 입력된 호스트 수만큼 생성하는 Create, 시스템 내부적으로 임의의 수를 생성하는 RC(Random Create)로 구성되어 있다.

Host Manager는 실제 환경에 설치된 호스트를 시뮬레이션하기 위해 XML 문서를 읽어 들이는 Host Importer, 읽어 들인 XML 문서를 분석하는 Host Parser, normalization을 통

해 호스트를 분석하는 Host Analysis가 구성되어 있다. 또한 임의의 호스트를 생성하는 Generator, 다양한 호스트 정보의 분포를 위해 비교하는 CRH(Compare Random Host)가 구성되어 있다. CRH를 통해 호스트 정보가 겹치지 않음을 판별하게 되면 HL(Host List)에 추가된다.

CA Manager(Cluster Algorithm Manager)는 사용자가 선택한 클러스터링 알고리즘을 나타내는 USA(User Selected Algorithm), 사용자가 설정한 클러스터 개수에 맞게 임의의 클러스터를 생성하는 ICP(Initialization Center Point), 생성된 임의의 클러스터를 기준으로 인접한 호스트를 찾는 FAH (Find Adjacent Host)가 수행된다. FAH를 통해 찾은 호스트는 CL(Cluster List)에 추가된다. 모든 호스트를 수행하게 되면 UCP(Update Center Point)를 통해 각 클러스터에 추가된 호스트를 기준으로 새로운 중심점으로 이동한다. 이 후에 FAH를 반복 수행함으로써 최적의 중심점으로 이동하게 되고 CCA(Check Clustering Availability)를 통해 추가 수행 여부를 판단한다. CCA를 통해 추가 수행이 필요하지 않게 되면 MAHP(Move Adjacent Host Point)를 통해 중심점으로 인접한 호스트를 클러스터로 선정한다.

Coordinate Converter는 Host Manager, CA Manager의 동작 상태를 Viewer가 시각화할 수 있도록 데이터를 가공하여 전달하는 브로커 역할을 한다.

Viewer는 RCS를 제어하기 위한 Control View, 시뮬레이션을 시각화하기 위한 Graph View, 클러스터별로 클러스터링된 호스트 정보를 사용자에게 제공해주기 위한 Cluster Information View로 구성되어 있다.

5. RCS 구현 및 성능평가

본 논문에서 제안하는 RCS의 초기화면은 Fig. 3과 같다. 자바 기반으로 JDK 1.8을 이용하여 구현하였으며, CPU i7, 메모리 16GB 환경에서 실행하였다. Fig. 3내의 ①은 데스크탑 자원 가상화를 위한 호스트 클러스터링 시뮬레이터를 사용자의 뷰 모드 선택에 따라 2D 또는 3D로 시각화한다. 또한 ①에서 클러스터를 선택하게 되면 클러스터링된 호스트의 정보를 보여주는 프레임이 활성화된다. Fig. 3내의 ②는 시뮬레이션할 데스크탑 PC 정보를 테이블 형태로 나타낸다. 이 테이블을 통해 사용자가 시뮬레이션할 호스트 정보의 직접 수정이 가능하다. Fig. 3내의 ③은 ①에서 시뮬레이션할 시각화를 2D 또는 3D로 나타낼지 선택하기 위한 인터페이스를 제공한다. 또한 클러스터링을 위한 알고리즘 선택과 2D, 3D 뷰에 맵핑하기 위한 정규화, ①에 정규화 상태를 나타내기 위한 뷰, 클러스터링하기 위한 클러스터의 수 등을 입력하기 위한 인터페이스를 제공한다. 시뮬레이션 상에서 클러스터링되는 상태를 보여주긴 위한 Line, 클러스터의 위치 변화를 파악하기 위한 Cluster Mark, 클러스터의 구별을 위한 Text 등의 인터페이스를 제공한다. Fig. 3내의 ④는 임의의 호스트에 대해 클러스터링 알고리즘 및 성능 비율 변화를 시뮬레이션하기 위한 호스트 수를 정의하는 인터페이스를 제공한다.

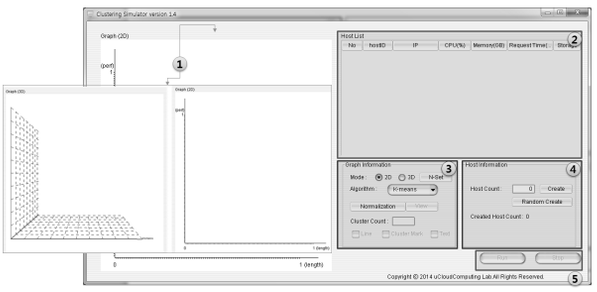

Fig. 3. Initial Configuration View of RCS

Fig. 4는 RCS에서 1,000개의 호스트를 생성하여 시뮬레이션하는 동작화면으로 좌측은 2D 시뮬레이션, 우측은 3D 시뮬레이션을 나타낸다. Fig. 4내의 ①은 호스트 개수를 1,000으로 입력하고 create를 통해 랜덤한 호스트 정보를 생성하고 이를 테이블에 나타낸다. Fig. 4내의 ②는 생성된 임의의 호스트를 그래프로 표현하기 위해 정규화를 수행하고 시뮬레이션 모드에 따라 x축, y축, z축의 값을 산출한다. view를 통해 각 호스트는 ②처럼 2D인 경우에는 하나의 포인트로 표현되고 3D인 경우에는 구 형태로 표현된다. Fig. 4내의 ③은 클러스터링을 하기 위해 클러스터의 수를 10으로 입력하고 Run 인터페이스를 통해 수행한 경우이다. 이 때 입력된 클러스터의 수만큼 클러스터링 시뮬레이션이 동작된다. Fig. 4내의 ④는 동작중인 시뮬레이션 상에서 Line과 Cluster Mark, Text를 활성화 경우로 현재 클러스터의 위치 및 클러스터링된 호스트가 연결된 모습을 나타낸다. 또한 각 클러스터의 호스트 ID를 그래프에 나타내줌으로써 클러스터 상태 파악이 가능하다.

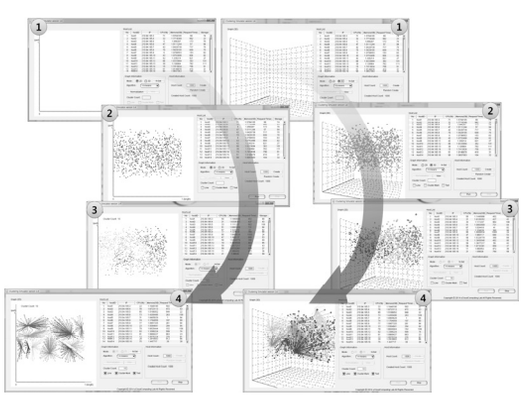

Fig. 4. 2D and 3D Simulation of RCS

Fig. 5는 Fig. 4에서 호스트 1,000개에 대한 클러스터 10개의 클러스터링 시뮬레이션 결과를 가지고 클러스터링된 호스트 정보를 나타낸다. Fig. 5내의 ①에서 클러스터를 사용자가 선택하게 되면 선택된 클러스터에 포함되어 있는 호스트 정보를 테이블 형태로 나타낸다. Fig. 5내의 ②는 선택된 클러스터를 IP 기반으로 식별할 수 있도록 사용자에게 제공해준다. Fig. 5내의 ③은 선택된 클러스터에 포함되어 있는 호스트 정보를 사용자가 파악할 수 있도록 제공해줌으로써 능동적인 파악이 가능하다.

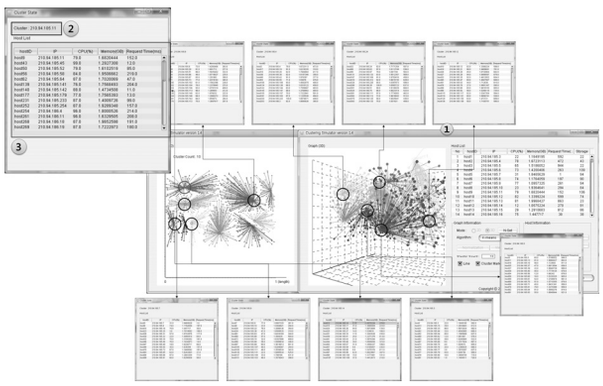

Fig. 5. Host Clustering and Monitoring through the RCS

RCS의 host와 클러스터 증가에 따른 클러스터링 워크로드 시간을 측정한 결과로 Fig. 6과 같다. 호스트의 수를 100, 200, 300, 400, 500, 1,000, 2,000, 3,000, 4,000, 5,000으로 증가시키고 각 증가시점에서 클러스터의 수를 5, 10, 15, 20, 25, 30으로 증가시켜 클러스터링을 수행하였다. 이 때 각 증가시점의 수행 속도는 50번씩 수행하여 얻은 수행 속도의 평균을 구하였다. Fig. 6을 통해서 호스트 수가 500개 미만인 경우에는 클러스터링 수행이 5초이내에 완료되는 것을 볼 수 있다. 또한 호스트 수가 1,000개 이상부터는 10초 이상의 시간이 소요되었다. 5,000개의 호스트에 클러스터 수를 30개로 적용하여 클러스터링을 한 경우에는 21초 정도 소요되었다. 실제로 5,000개의 호스트에 클러스터링 테스트를 위한 인력, 전력, 시간적 소모비용보다 효율적임을 알 수 있다.

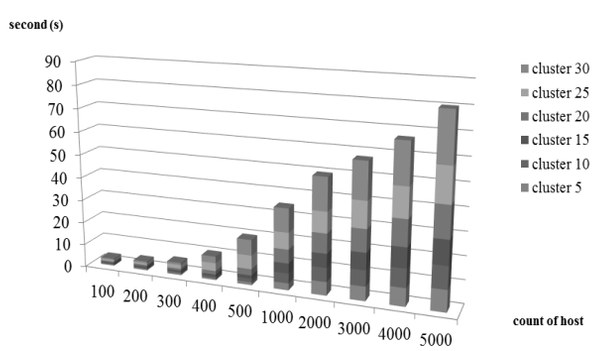

Fig. 6. RCS Architecture

6. 결 론

사용자들은 데스크탑 PC나 스마트 디바이스의 한정된 자원의 성능으로 인해 클라우드 서비스의 사용이 급증하고 있다. 이러한 클라우드 서비스를 위한 가상화 중에 데스크탑 자원 가상화는 기존 데스크탑 PC의 자원을 활용하기 때문에 클러스터링을 위한 계층적 구조가 매우 중요시된다. 따라서 본 논문에서는 데스크탑 가상화의 클러스터 선정을 위해 리소스 클러스터링 시뮬레이터인 RCS를 제안하였다. RCS는 동작중인 클라우드 인프라내의 호스트를 시뮬레이션 할 수 있도록 XML 기반의 인터페이스를 제공하였다. 또한 다양한 클러스터링 알고리즘 중에 적합한 클러스터링 알고리즘을 선정하기 위해 사용자가 클러스터링 요소를 결정할 수 있도록 사용자 인터페이스를 제공하였다.

향후에는 호스트의 정보를 CPU, 메모리, 스토리지뿐만 아니라 자원 가용 상태에 따른 시간과 네트워크 인프라도 고려할 수 있도록 사용자가 정의 인터페이스를 제공하여 다양한 응용 분야에 활용 가능하도록 연구하고자 한다.

References

[1] Nitesh Shrivastava, and Ganesh Kumar, “A survey on cost effective multi-cloud storage in cloud computing,” *International Journal of Advanced Research in Computer Engineering and Technology*, Vol.2, Issue. 4, Apr. 2013.

[2] So-Yeon Kim, Hong-Chan Roh, Chi-Hyun Park, and Sang-Hyun Park, “Analysis of Metadata Server on Clustered File Systems,” in *Proceedings of the Korea Computer Congress 2009*, KCC, Vol.36, No.1, Jul. 2009.

[3] Pradnya Eknath Gaonkar, Sachin Bojewar, and Jayesh Ajit Das, “A Survey: Data Storage Technologies,” *International Journal of Engineering Science and Innovative Technology*, Vol.2, No.2, pp.547-554, Mar. 2013.

[4] Garth A. Gibson, and Rodney Van Meter, “Network attached storage architecture,” *Communications of the ACM*, Vol.43, No.11, pp.37-45, Nov. 2000.

[5] B. Dong, Q. Zheng, F. Tian, K. Chao, R. Ma, and R. Anane, “An optimized approach for storing and accessing small files on cloud storage,” *Journal of Network and Computer Applications*, Vol.35, No.6, pp.1847-1862, Nov. 2012.

[6] Zhanquan Sun, Geoffrey Fox, Weidong Gu, and Zhao Li, “A parallel clustering method combined information bottleneck theory and centroid-based clustering,” *Journal of Supercomputing*, Vol.69, No.1, pp.452-467, Jul. 2014.

[7] Sarah P. Preheim, Allison R. Perrotta, Antonio M. Martin-Platero, Anika Gupta, and Eric J. Alm, “Distribution-Based Clustering: Using Ecology To Refine the Operational Taxonomic Unit,” *Applied and Environmental Microbiology*, Vol.79, No.21, pp.6593-6603, Nov. 2013.

[8] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, Arthur Zimek, “Density-based clustering,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol.1, No.3, pp. 231-240, May/Jun. 2011.

[9] Hartuv Erez, and Shamir Ron, “A clustering algorithm based on graph connectivity,” *Information Processing Letters*, Vol.76, No.4, pp.175-181, Dec. 2000.

[10] Manojit Chattopadhyay, Pranab K. Dan, and Sitanath Mazumdar, “Comparison of visualization of optimal clustering using self-organizing map and growing hierarchical self-organizing map in cellular manufacturing system,” *Applied Soft Computing*, Vol.22, pp.528-543, Sep. 2014.

[11] Naidila Sadashiv, and S. M Dilip Kumar, “Cluster, Grid and Cloud Computing: A Detailed Comparison,” in *Proceedings of the 6th International Conference on Computer Science and Education*, ICCSE 2011, pp.477-482, Aug. 2011.

[12] Anthony Sulistio, Uros Cibej, Srikumar Venugopal, Borut Robic, Rajkumar Buyya, “A toolkit for modeling and simulating data Grids: an extension to GridSim,” *Concurrency and Computation: Practice and Experience*, Vol.20, No.13, pp.1591-1609, Sep. 2008.

[13] Rajkumar Buyya, and Manzur Murshed, “GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing,” *Concurrency and Computation: Practice and Experience*, Vol.14, No.13-15, pp.1175-1220, Nov. 2002.

[14] Luís F. W. Góes, Luiz E. S. Ramos, and Carlos A. P. S. Martins, “ClusterSim: A Java-Based Parallel Discrete-Event Simulation Tool for Cluster Computing,” in *Proceedings of the 2004 IEEE International Conference on Cluster Computing*, pp.401-410, Sep. 2004.

[15] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation or resource provisioning algorithms,” *Software: Practice and Experience*, Vol.41, No.1, pp.23-50, Jan. 2011.

김 현 우

https://orcid.org/0000-0001-8295-2598

e-mail : hwkim@dongguk.edu

2005년 원광대학교 전기전자 및 정보공학부(공학사)

2016년 동국대학교 멀티미디어공학과 (공학박사)

2018년~현 재 동국대학교 멀티미디어공학과 연구교수

관심분야 : Cloud Computing, Internet of Things, Mobile Edge Computing, Information Security