최적의 IoT 서비스 제공을 위한 오픈스택 기반 스케줄러 비교 및 분석†

문영주*, 강지훈*, 유태묵*, 유헌창*, 정광식**, 길준민***††
*고려대학교 분산클라우드컴퓨팅 연구실
**한국방송통신대학교 컴퓨터과학과
***대구가톨릭대학교 IT공학부
*e-mail: {moonyj, k2j23h, lyoo12, yuhc}@korea.ac.kr
**e-mail: Kchung0825@knou.ac.kr ***e-mail: jmgil@cu.ac.kr

A Comparison and Analysis of the Openstack-based Scheduler for a IoT Service

YoungJu Moon*, JiHun Kang*, TaeMook Yu*, HeonChang Yu*,
KwangSik Chung**, JoonMin Gil***
*Distributed & Cloud Computing Lab., Korea University
**Department of Computer Science, Korea National Open University
***School of Information Technology Eng., Catholic University of Daegu

요 약

모든 사물에 인터넷이 연결되는 사물 인터넷(IoT: Internet of Things)시대가 열렸다. IoT 디바이스들을 연결하기 위해 클라우드 또한 더욱 관심이 높아지고 있다. IoT 디바이스를 연결한 클라우드는 작은 단위의 작업들을 다량으로 수행하게 된다. IoT 서비스에서 발생하는 작업들을 효율적으로 처리하기 위해서는 적합한 작업 스케줄링이 반드시 필요하다. 본 논문에서는 오픈소스 기반의 플랫폼인 오픈스택(OpenStack)에서 Filter 스케줄러와 Chance 스케줄러를 VM 개수에 따라 단위 시간동안 성능을 비교·분석한다. 이를 통해 오픈스택에서 IoT 서비스 사용자들을 위해 합리적인 스케줄러 방법을 도출해낼 수 있다.

1. 서론

최근 미래 사회의 중요한 키워드로 초연결 사회(Hyper-Connected Society)가 꼽히고 있다. 초연결 사회란, 사람과 사물, 자연, 사이버 세계가 네트워크를 통해 연결되어 상호호환 할 수 있는 환경이다. 스마트 폰이 대중화되면서 각종 센서의 대중화와 가격 하락으로 초연결 사회의 주요 이슈인 사물 인터넷(IoT: Internet of Things) 기술이 본격적으로 활용되고 있다[1]. IoT는 컴퓨터, 센서, 스마트 폰, 웨어러블 기기 등 무수히 많은 물체를 인터넷으로 연결하고 있다. 결국 수용되어야하는 디바이스나 처리되어야 하는 데이터의 양이 제한이 없어야하기 때문에 이를 모두 포괄할 수 있는 클라우드 컴퓨팅이 IoT와 함께 대두되고 있다[2].

IoT 디바이스에서 발생되는 데이터는 총량이 많고, 업데이트 속도가 잦으며, 다양한 속성을 포함하고 있다. 예를 들면, 자동차 GPS 시스템의 차 한 대에서는 차량의 ID, 차량의 번호, 차량의 종류, 차량이 이동한 거리, 차량이 이동한 시간, 차량의 위치 등 무수히 많은 특성들이 있으며, 기록되는 주기는 30초에서 60초사이가 될 것이다. 하나의 데이터는 바이트(Byte)의 작은 단위이지만, 많은 차량이 지속적으로 하루 동안 만드는 데이터는 매우 클 것이다[3]. 따라서 IoT-cloud 서비스는 다량의 디바이스들로부터 지속적으로 대량의 데이터가 발생하며, 클라우드에서는 이를 읽어 들여 지속적으로 데이터를 저장 및 관리를 해야 한다. 이로 인해 IoT 서비스를 적용하기 위한 오픈소스 클라우드 컴퓨팅 플랫폼은 단위 시간(초)당 작업 처리량과 디스크의 I/O 작업이 용이해야 한다.

본 논문에서는 오픈소스 클라우드로 가장 많은 개발자가 참여하고 있는 오픈스택(Openstack)의 작업 스케줄러인 Filter 스케줄러와 Chance 스케줄러를 가상머신 개수에 따라 단위 시간당 작업 처리량과 디스크 I/O 작업 성능을 비교·분석한다. 작업 성능 평가를 통해 IoT 서비스에 효율적인 스케줄링 방법을 찾을 수 있다.

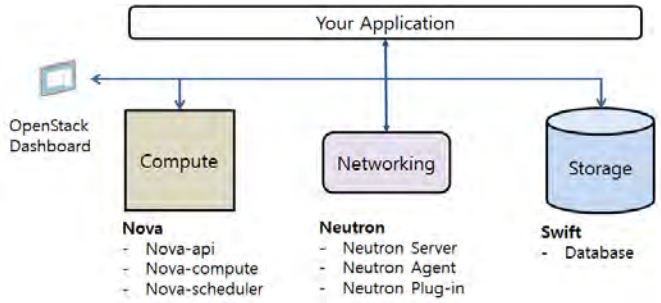
2장에서는 오픈스택의 구성요소와 스케줄러를 소개하고, 3장에서는 작업 단위시간당 성능을 비교 및 분석 한다. 마지막으로 4장에서는 분석 결과를 통한 결론과 향후 연구에 대한 서술로 마무리한다.

2. 오픈스택(OpenStack)

오픈스택은 세 개의 메인 컴포넌트로 구성이 되어 있다. (그림 1)에서 첫 번째로 제어 부분인 Nova, 두 번째는

† 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2014R1A1A2055463).
†† 교신저자

네트워크 부분인 Neutron(Grizzly 버전에서 Havana로 바뀌면서 Quantum에서 Neutron으로 이름이 변경), 세 번째는 스토리지 부분인 Swift이다.

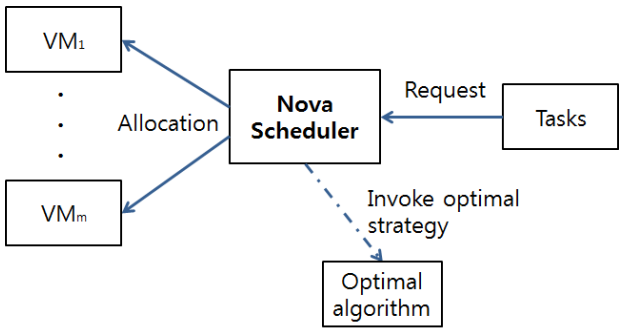


(그림 1) 오픈스택 아키텍처

1) 제어 부분인 Nova를 살펴보면, Nova는 오픈스택의 심장과 같은 부분으로 Python으로 작성되어 있으며, 대규모의 가상머신(VM)들의 네트워크들을 관리하고 스케줄링 한다.

2) 네트워크 부분인 Neutron은 유연한 플러그인 아키텍처와 풍부한 API를 통해 클라우드 네트워크를 지원한다. 구성요소로는 Neutron 서버, Neutron 에이전트, Neutron 플러그인, Neutron Database, Network provider, 메시지 큐로 이루어져 있다.

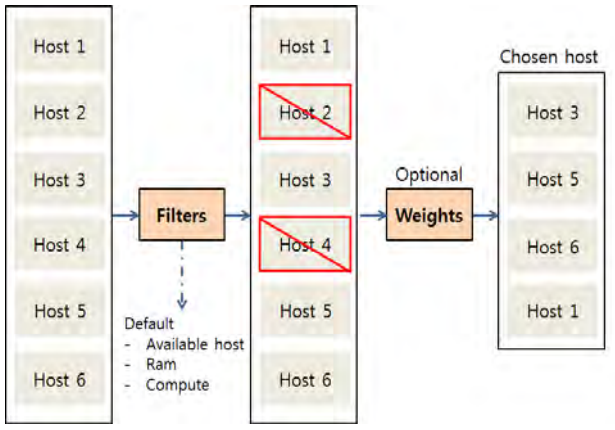
3) 스토리지 부분인 Swift는 자원을 관리하는데 사용되며, 클러스터에서 데이터의 복제 및 무결성을 보장할 책임이 있다. 따라서 스토리지 노드의 실패가 발생할 경우 데이터가 잃지 않도록 하는 프로세스가 존재한다[4][5].



(그림 2) 오픈스택 스케줄러 모델

오픈스택에서 스케줄러의 역할은 작업의 가상머신 할당이다. (그림 2)와 같이, 작업 요청 시 오픈 스택의 Nova Scheduler에서 Optimal algorithm으로 지정된 스케줄링 방법을 통해 작업에 가상머신이 할당된다[6]. Nova Scheduler에서는 기본 Optimal 알고리즘으로 Filter 스케줄러와 Chance 스케줄러를 제공하며, 가장 적은 부하의 가상머신에 작업을 할당하던 Simple 스케줄러는 현재 지원하지 않는다.

Filter 스케줄러는 여과되지 않은 호스트 사전의 필터 속성을 사용하여 필터의 마지막 인스턴스의 요청 수 (택한 호스트의 목록에 추가 할 때마다 가장 weight 값이 높은 호스트를 선택)에 대한 호스트를 선택한다. 다음의 후보를 찾을 수 없을 때는 스케줄링 할 수 있는 더 적절한 호스트가 없다는 것을 의미한다.



(그림 3) Filter 스케줄러 모델

Filter 스케줄러는 (그림 3)과 같이 중간 단계의 Filters와 Weights 부분을 타겟에 맞게 새로운 전략을 구축할 수 있어 유연하다[7].

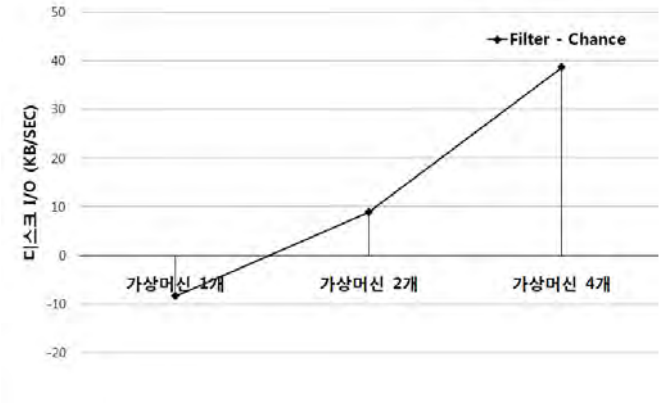
Chance 스케줄러는 작업의 유형에 구애받지 않기 위해 정의된 스케줄러로 사용가능한 가상머신에 무작위로 작업을 할당한다. 사용자 친화적인 Filter 스케줄러와는 달리 Chance 스케줄러는 직관적이지만 유연하지 못하다.

3. FIlter 스케줄러와 Chance 스케줄러의 성능 비교 및 분석

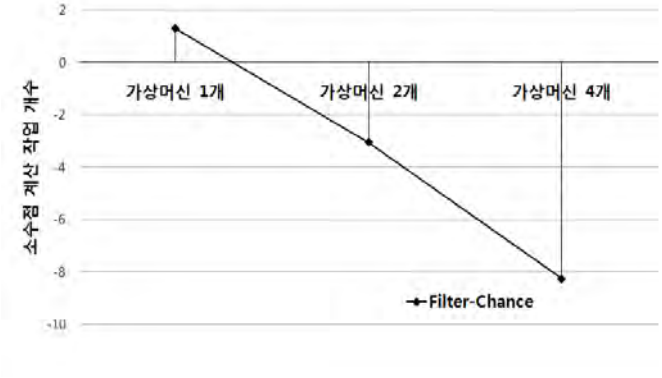
Filter 스케줄러와 Chance 스케줄러의 성능 비교를 위한 본 논문의 실험 환경은 다음과 같이 구성하였다. CPU는 Intel Core i5-4590T이고, 메모리는 8GM인 호스트 시스템에서 클라우드 환경을 구축하였다. 호스트 OS는 Ubuntu server 14.04이다. 각 가상머신은 VCPU는 1개와 2GB의 메모리, 20GB의 디스크로 구성을 하였다. 가상머신의 OS는 Ubuntu server 14.04이다.

각 가상머신에서는 디스크 I/O 성능을 총 1GB의 파일을 16개의 스레드로 나누어 디스크에 쓰는데 걸리는 시간을 측정한다. CPU 성능은 초당 소수점 연산 방법으로 측정한다.

(그림 4)에서 가상머신이 1개일 경우에는 Filter 스케줄러와 Chance 스케줄러의 디스크 I/O 속도 차이가 8.3kb/sec으로 근소하다. 그러나 가상머신이 증가할수록 Filter 스케줄러가 Chance 스케줄러보다 디스크 쓰기를 처리하는 속도가 더욱 빨라진다.



(그림 4) Filter 스케줄러와 Chance 스케줄러 간의 디스크 I/O 성능 차이



(그림 5) Filter 스케줄러와 Chance 스케줄러 간의 소수점 계산 작업 처리량 성능 차이

(그림 5)에서는 가상머신이 1개일 경우에는 Filter 스케줄러와 Chance의 스케줄러 단위 시간당 작업 처리량 차이가 1.3개로 근소하다. 하지만 VM 개수가 2개・4개로 늘어날수록 Chance 스케줄러가 Filter 스케줄러보다 단위 시간당 작업 처리량이 점점 증가하는 것을 알 수가 있다.

결론적으로 디스크 I/O와 작업 처리량에서 두 스케줄러가 다른 양상을 보인다. 먼저, 디스크 I/O 작업에서 가상머신의 개수가 2개 이상일 경우에는 Filter 스케줄러의 성능이 Chance 스케줄러의 성능보다 점점 좋아지는 것을 알 수가 있다. 가상머신이 1개일 경우에는 스케줄링의 영향이 적어 차이가 작지만, 가상머신의 개수가 늘어날수록 사용가능한 가상머신을 랜덤으로 할당하는 Chance 스케줄러와는 달리 Filter 스케줄러는 디폴트 필터에서 충분한 자원의 여유가 있는 머신에 작업을 할당하기 때문에 상대적으로 디스크 I/O 속도가 빨라졌다.

반대로 소수점 계산 작업 처리량에서 가상머신의 개수가 2개 이상일 경우에는 Chance 스케줄러의 성능이 Filter 스케줄러의 성능보다 더 좋았다. 가상머신이 1개일 경우에는 디스크 I/O 작업과 마찬가지로 스케줄링의 영향이 적어 차이가 작지만, 가상머신의 개수가 늘어날수록 Filter 스케줄러가 고려해야할 요인(RAM+CPU)으로 인해 작업량과 무관한 가상머신을 선택하게 되어 Chance 스케줄러와의 작업량 격차가 늘어나게 된다.

4. 결론

본 논문에서는 오픈스택에서 효율적으로 IoT 서비스를 제공할 수 있는 스케줄러를 검증하기 위해, 디폴트 스케줄러인 Filter 스케줄러와 Chance 스케줄러를 가상머신 개수에 따라 단위 시간(초)동안의 성능을 비교·분석했다. 디스크 I/O의 경우 가상머신 개수가 증가할수록 Filter 스케줄러가 더 나은 성능을 보였으며, CPU 성능 측정을 위한 소수점 계산 작업 처리량에서는 Chance 스케줄러가 더 나은 성능을 보였다. 이를 통해 데이터베이스의 접근이 잦은 IoT 서비스는 Filter 스케줄러가 유리하며, 단순 계산이 많은 IoT 서비스는 Chance 스케줄러가 유리하다는 것을 알 수 있다. 추후 연구로 클라우드에서 다양한 IoT 서비스를 지원하기 위해 Filter 스케줄러를 개선한다.

참고문헌

[1] 석왕헌, 송영근, 고순주 “통신환경 변화에 따른 M2M 산업 생태계 및 파급효과 분석” IT 이슈리포트 2013-7, ETRI, 2013.06.

[2] Xianrong Zheng Pressman "Cloud Service Negotiation in Internet of Things Environment: A Mixed Approach" Industrial Informatics, IEEE Transactions on (Volume:10 , Issue: 2)

[3] Youzhong Ma "An Efficient Index for Massive IOT Data in Cloud Environment" CIKM '12 Proceedings of the 21st ACM international conference on Information and knowledge management

[4] (2015) Openstack history. [Online]. Available: https://www.openstack.org/

[5] Amine Barkat, Alysson Diniz dos Santos, Thi Thao Nguyen Ho “OpenStack and CloudStack: Open Source Solutions for Building Public and Private Clouds” 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

[6] Hu, Bin, and Hong Yu. "Research of Scheduling Strategy on OpenStack." 2013 International Conference on Cloud Computing and Big Data (CloudCom-Asia). IEEE, 2013.

[7]] (2015)Filter scheduler. [Online]. Available http://docs.openstack.org/