클라우드 컴퓨팅의 과금 체계

최승일*, 박동진*, 이동주*
*공주대학교 산업시스템공학과
e-mail:sichoi@kongju.ac.kr

Billing System of Cloud Computing

Seungil Choi*, Dong-Jin Park*, Dong-Ju Lee*
*Dept of Industrial & Systems Engineering, Kongju National University

요 약

본 연구에서는 클라우드 서비스 확산에 필요한 유틸리티 컴퓨팅의 과금체계에 대해 살펴보았다. 클라우드 컴퓨팅에서 요구하는 조건들을 만족하는 과금정책을 제시한 후, 다양한 과금정책들을 비교 평가할 수 있는 지표로 작업 서비스 품질 만족도와 클러스터 수익성을 제시한다.

1. 서론

클라우드 컴퓨팅은 2006년 구글의 연구원인 비시 글리아에 의해 제안된 것으로 인터넷상의 서버를 통하여 데이터 저장, 네트워크, 콘텐츠 사용 등 IT 서비스를 모두 사용할 수 있는 컴퓨팅환경을 의미한다. 최근에는 클라우드 컴퓨팅을 기반으로 한 비즈니스 모델이 확산되고, 이에 따라 클라우드 컴퓨팅 시장 규모가 빠르게 증가하고 있다[1].

클라우드 컴퓨팅이 활성화되면서 하드웨어 장비는 보유하는 개념에서 수도와 전기처럼 빌려서 사용하는 개념인 유틸리티 컴퓨팅이 도입되고 있다. 이러한 유틸리티 컴퓨팅이 기업이나 조직에서 활성화되기 위해서는 다음과 같은 문제들에 대한 해결책이 요구된다.

첫째로 정보기술 거버넌스 구축이 필요하다. 정보기술 거버넌스 구축은 비즈니스의 효과성을 목적으로 하게 되며 비즈니스의 전략 변화를 정보기술로 효과적으로 지원하기 위해서는 기업의 민첩성을 높일 수 있는 유틸리티 기반이 바람직하다.

둘째로는 보안성의 강화가 필요하다. 대부분의 기업에서 내부에서 혹은 외부에서 유틸리티 서비스를 받게 된다. 두가지 형태의 서비스 모두 기업의 핵심 정보 누출의 위험성에서 자유로울 수가 없으므로 보안성 강화가 중요하다. 이미 인프라 측면에서 유틸리티의 안정성은 고가용성 등으로 강화되어 있고, 자가치료(self healing) 등의 가상화 지원과 비즈니스 연속성 지원이 이루어진다.

셋째로는 새로운 과금 체계의 등장이다. 궁극적인 유틸리티 컴퓨팅은 하나의 공공설비와 같이 서비스하는 것으로, 이러한 서비스에서의 과금은 주문형 용량확장방식(capacity on demand) 등에서 월별 할부제, 사용량 기준 요금제로 발전하고 있다. 주문형 용량 확장은 고객이 필요로 하는 정보기술자원을 무상으로 설치한 후에 시스템에 부하가 생기거나 서비스가 증대될 때 자원을 활용하는 방식으로 사용한 시간만큼 지불하는 방식으로 제공된다. 월별 할부제는 정보기술자원을 구매할 때 비용을 지불하는 방식이 아니라 월별 과금제를 적용하는 것으로 실시간의 사용량을 근간으로 지불하게 되는 방식이다.

넷째로는 유틸리티에 대한 모니터링 서비스의 발달이 필요하다. 서비스된 유틸리티 컴퓨팅을 단지 사용만 하는 것이 아니라 서비스 수준의 구현과 환경을 늘 실시간으로 모니터링 할 수 있는 환경 구현이 요구된다.

본 연구에서는 클라우드 컴퓨팅이 활성화되기 위한 4가지 선결요건 중 과금체계에 대해 자세히 살펴보고자 한다.

2. 과금 체계

2.1 과금 정책

1) 기본 요건

클라우드 컴퓨팅에서 과금 정책이 만족해야 할 네 가지 요건은 다음과 같다. 첫째, 과금 공식은 유연성을 지니고 있어 공유 수준을 결정하기 위해 자원에

대한 과금을 쉽게 조정할 수 있어야 한다. 둘째, 과금 공식은 공정성을 지녀야 한다. 자원에 대한 과금은 사용자들의 실제 사용량에 근거하여 이루어져야 한다. 많은 자원을 사용하는 사용자는 적은 자원을 사용하는 사용자보다 많은 비용을 부담한다. 또한 서비스 수준에 있어서는 높은 서비스 수준(짧은 처리 시한)을 원하는 사용자가 낮은 서비스 수준(긴 처리 시한)을 원하는 사용자보다 많은 비용을 부담한다. 셋째, 과금 공식은 개별 자원의 가격이 정적이지 않고 클러스터 운영 조건에 따라 변화할 수 있도록 동적이어야 한다. 넷째, 과금 공식은 자원에 대한 수요와 공급에 따라 적절한 가격이 산정되도록 적응할 수 있어야 한다. 예를 들어 자원에 대한 수요가 높아지면 자원에 대한 과부하를 막기 위해 자원에 대한 단가가 상승해야 한다[3].

2) 과금 공식

과금 공식은 위의 네가지 기본 요건을 만족하도록 정의한다. 요청한 작업에 의해 사용되는 클러스터 자원들은 프로세서 사용 시간, 메모리 크기, 스토리지 크기 등이 있다. 작업 i가 컴퓨팅 노드 j에서 사용하는 클러스터 자원의 단가 P_{ij}는 다음과 같이 계산된다.

$$P_{ij} = (\alpha * PBase_j) + (\beta * PUtil_{ij})$$

단가 P_{ij}는 두가지 요소로 구성되는데 컴퓨팅 노드 j에서 자원을 사용하는 기본 단가 $PBase_j$에 기초한 정적인 부분과 작업 i를 처리하는 자원의 사용 단가 $PUtil_{ij}$에 기초한 동적인 부분으로 구성된다. α, β는 정적인 부분과 동적인 부분에 대한 인수로 전체 단가인 P_{ij}에서 정적인 부분과 동적인 부분의 가중치를 쉽게 구성하고 수정할 수 있도록 유연성을 제공한다. 클러스터 소유자는 클러스터 자원에 대해 고정 기본 단가를 지정한다. 예를 들어 $PBase_j$는 프로세서 사용 시간당 $0.1, 디스크 저장 공간 월 1GB당 $0.15 등으로 정할 수 있다. $PUtil_{ij}$는 컴퓨팅 노드 j에서 AT_i(작업 i가 클러스터에 도달한 시간)부터 DT_i(작업 i가 종료된 시간)까지의 자원 사용량을 근거로 $PBase_j$의 인수로 정의된다.

$$PUtil_{ij} = \frac{RESMax_j}{RESFree_{ij}} * PBase_j$$

$RESMax_j$는 컴퓨팅 노드 j에서 AT_i부터 DT_i까지 활용가능한 자원의 최대용량이다. $RESFree_{ij}$는 컴퓨팅 노드 j에서 AT_i부터 DT_i까지 자원의 사용되지 않는 잔여용량으로 현재 실행중인 다른 작업과 작업 i에 사용되는 자원 용량을 최대용량에서 차감하여 계산한다.

$$RESFree_{ij} = RESMax_j - \left(\sum_{k=1}^{n_{accept_j}} RES_k \right) - RES_i$$

클러스터에 제출된 n개의 작업 중에서 n_{accept}개의 작업만이 진입 통제를 통해 실행될 수 있도록 받아들여진다. 만약 진입 통제가 이루어지지 않으면 $n = n_{accept}$이다. 또한 n_{accept}개의 작업 중에서 n_{accept_j}는 컴퓨팅 노드 j에서 AT_i부터 DT_i까지 실행중인 작업의 수를 의미한다. 경제적 관점의 진입 통제와 자원 할당 메커니즘은 $RESFree_{ij}$가 항상 양수가 되도록 하기 위해 P_{ij}를 계산하기 전에 컴퓨팅 노드 j에 충분한 자원이 있는 먼저 확인한다.

과금 공식은 컴퓨팅 노드 j에서 작업 i에 사용되는 각각의 다른 자원에 대한 단가 P_{ij}를 산정한다. 따라서 컴퓨팅 노드 j에서 작업 i를 실행하는 전체 가격은 각각의 P_{ij}들로부터 산출된다. 이처럼 정교한 과금은 작업들이 각기 다른 자원들의 사용량에 근거하여 가격이 정해진다는 점에서 공정하다고 할 수 있다. 예를 들어 계산이 많이 필요한 작업은 데이터 집중적인 작업에 비해 많은 스토리지 용량을 필요로 하지 않아 디스크 스토리지 자원을 사용하는 것에 대해서는 가격이 낮게 책정된다.

과금 공식은 사용자 중심과 시스템 중심의 인수들을 동시에 고려하도록 한다. 사용자 중심의 인수는 작업 i에 요구되는 자원 용량 RES_i이다. 예를 들어 요구된 자원 용량이 적으면(low RES_i) 단가 P_{ij}는 낮아진다. 시스템 중심의 인수는 컴퓨팅 노드 j에서 활용가능한 용량인 $RESFree_{ij}$이다. 예를 들어 컴퓨팅 노드 j에서 수요가 적은 자원(high $RESFree_{ij}$)은 낮은 단가가 책정된다.

Libra는 클러스터 작업량 상태에 관계없이 짧은 종료 시한을 가지는 작업보다 긴 종료 시한을 가지는 작업에 인센티브를 부여한다[2]. 여기에 제시된 과금 공식은 $PUtil_{ij}$가 작업 i의 종료 시한(AT_i부터 DT_i까지)에 근거하여 자원의 사용정도를 고려하므로 클러스터 작업량을 고려하게 된다. 예를 들어 사

용자가 10 단위의 메모리 크기를 필요로 하는 작업 i를 수행하는데 2시간과 5시간의 종료 시한을 지정하는 경우를 생각해 보자. 컴퓨팅 노드 j에서 기본단가 $PBase_j$가 단위당 $1이며, 종료 시한까지 시간당 100 단위의 메모리 크기가 사용되지 않고, 종료 시한까지 90 단위의 메모리 크기를 사용하는 n_{accept_j}개의 작업이 있다고 가정한다.

예를 들어 2시간의 짧은 종료 시한에 대해서는 $PUtil_{ij}$=(200/(200-90-10))*1=$2 per unit이고, 5시간의 긴 종료 시한에 대해서는 $PUtil_{ij}$=(500/(500-90-10))*1=$1.25 per unit으로 단가가 낮아진다.

여기에 제시되는 과금 공식은 요구된 종료 시한 동안 컴퓨팅 노드에서 활용 가능한 자원들에 따라 작업의 전체 단가가 달라지므로 동적이라고 할 수 있다. 또한 전체 단가가 작업 제출을 권장하거나 단념하도록 하면서 현재의 수요와 공급에 따라 자동적으로 조정되므로 과금 공식이 적응력이 있다고 할 수 있다.

2.2 평가 지표

클러스터의 활용도 수준을 측정하기 위해 작업 서비스 품질 만족도(Job QoS Satisfaction)와 클러스터 수익성(Cluster Profitability)이라는 두 가지 사용자 중심의 성과 평가 지표를 정의한다. 먼저 작업 서비스 품질 만족도는 작업 요구를 만족시키는 활용도 수준을 측정한다. 서비스 품질 만족도가 높을수록 더 나은 성과를 나타낸다. 서비스 품질 만족도는 제출된 n개의 작업 중 요구된 서비스 품질(처리시한과 예상비용)을 만족한 n_{QoS}개작업의 비율로 정의된다. 두 가지의 기본 서비스 품질 매개변수로는 처리시한 D_i과 예상비용 B_i이 있다. 처리시한을 만족하려면 완료 시간이 처리시한보다 작아야 하며, 비용 조건을 만족하려면 실제비용(C_i)이 예상비용(B_i)보다 작아야 한다.

$$\text{Job QoS Satisfaction} = n_{QoS}/n$$

두 번째로 클러스터 수익성은 클러스터 소유자의 경제적 이익을 발생시키는 활용도 수준을 측정한다. 클러스터 수익성이 높을수록 더 나은 성과를 나타낸다. 클러스터 수익성은 제출된 작업의 총 예상비용 중 서비스 품질을 만족하는 클러스터에 의해 발생한 이익의 비율로 정의한다.

$$\text{Cluster Profitability} = \sum_{i=1}^{n_{QoS}} C_i \Big/ \sum_{i=1}^{n} B_i$$

3. 결론

가상화 기술을 기반으로 한 클라우드 컴퓨팅의 활성화를 위해서는 원천기술의 R&D도 중요하지만, 클라우드 컴퓨팅 도입 촉진을 위한 법과 제도 개선, 클라우드 컴퓨팅 서비스 신뢰성 제고를 위한 보안 및 인증체계 구축, 클라우드 컴퓨팅의 호환성 제공을 위한 표준화 추진 등이 필요하다. 본 연구에서는 클라우드 서비스가 하드웨어를 보유하는 개념에서 수도와 전기처럼 빌려서 사용하는 개념으로 제공되면서 필요한 유틸리티 컴퓨팅의 과금체계에 대해 살펴보았다.

클라우드 컴퓨팅에서의 과금정책은 공유수준에 따라 쉽게 조정할 수 있어야 하며, 실제 사용량과 서비스 수준에 따라 과금이 이루어지며, 가격은 클러스터 운영 조건에 따라 변화할 수 있도록 동적이어야 하며, 자원에 대한 수요와 공급에 따라 가격이 결정되어야 한다. 본 연구에서는 이러한 조건을 만족하는 과금정책을 제시하였고, 다양한 과금정책들을 평가할 수 있는 지표로 작업 서비스 품질 만족도와 클러스터 수익성이라는 사용자 중심의 지표를 제시하였다. 향후 과금체계에 대한 심도있는 연구가 클라우드 서비스 활성화에 일조를 할 수 있기를 기대한다.

참고문헌

[1] 민옥기, 김학영, 남궁한, 클라우드 컴퓨팅 기술 동향, 전자통신동향분석, 제24권 제4호, 2009.

[2] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya, “Libra: a computational economy-based job scheduling system for clusters”, Software: Practice and Experience, 34(6), pp. 573-590, 2004.

[3] C. S. Yeo and R. Buyya, “Pricing for Utility-Driven Resource Management and Allocation in Clusters”, The International Journal of High Performance Computing Applications, 21(4), pp. 405-418, 2007.