Journal of The Korea Society of Computer and Information
Vol. 17 No. 9, September 2012
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci/2012.17.9.001

클라우드 컴퓨팅에서 워크플로우의 특성을 고려한 스케줄링 기법

김 정 원*

Scheduling Scheme for Cloud Computing based on Workflow Characteristics

Jeong-Won Kim*

요 약

클라우드 컴퓨팅은 컴퓨팅 자원의 효율적 사용과 수월한 접근이라는 관점에서 사용자 및 서비스 제공자에게 모두 각광받고 있다. 한편, 클라우드 컴퓨팅의 자원은 이질적 환경으로 구성되는 경우가 많고 사용자의 워크플로우는 다양한 특성을 가진다. 따라서 본 연구의 목적은 이러한 환경에서 워크플로우의 응답성 요구에 따라 우선순위를 부여하고 워크플로우를 구성하는 태스크의 자원 할당시 응답성, 비용, 그리고 부하 균등으로 구성되는 QoS 메트릭을 구성하여 효율적으로 워크플로우를 스케줄링하는 것이다. 제안기법의 성능 평가를 위해 다양한 실험을 수행한 결과 응답성과 가용성이 향상되었음을 확인할 수 있었다.

▸ Keywords : 클라우드 컴퓨팅, 워크플로우, 스케줄링, 응답성, 가용성

Abstract

Cloud computing has got great popularity in recent times because users can easily access its resources as well as service providers can use efficiently use its resources. By the way, cloud computing are composed of heterogeneous resources and workflows of user application have various characteristics. So, the main goal of this paper is to design new efficient workflow scheduling algorithm, which classifies workflows through their importance degree and allocates resources to each workflow based on QoS metrics such as responsibility, cost and load balancing. Simulation results show that the proposed scheme can improve the responsibility as well as availability of resource.

▸ Keywords : cloud computing, workflow, scheduling, responsibility, availability

* 제1저자 : 김정원
* 투고일 : 2012. 07. 03, 심사일 : 2012. 07. 31, 게재확정일 : 2012. 08. 23.
* 신라대학교 컴퓨터공학과(Dept. of Computer Engineering, Silla University)

I. 서 론

클라우드 컴퓨팅은 인터넷 기반의 환경으로 전기, 가스 등과 같은 공공 유틸리티처럼 사용자에게 주문형으로 자원을 제공한다. 이를 위해 클라우드 컴퓨팅은 중앙 집중식으로 데이터와 응용프로그램을 유지하고 인터넷으로 이를 서비스해야 한다. 클라우드 컴퓨팅 사용자는 개별적으로 소프트웨어를 설치하거나 데이터를 별도로 관리할 필요 없이 인터넷상에서 이들 자원에 접근할 수 있으므로 저장 공간, 메모리, 대역폭을 집중하여 자원을 효율적으로 이용할 수 있다.

이와 같이 클라우드 컴퓨팅은 다수의 사용자와 다수의 자원으로 구성되는 복합 환경으로 사용자 데이터의 흐름인 워크플로우(Workflow)를 적절한 자원에 할당하여 자원을 효율적으로 사용해야 함과 동시에 사용자의 QoS(Quality of Service)를 보장해야 한다. 이 워크플로우는 WfMC[1]에서는 문서, 정보, 또는 태스크가 하나의 노드에서 다른 노드로 전체 또는 부분이 자동으로 일련의 규칙에 의해 전달되는 흐름으로 정의하고 있다. 클라우드 컴퓨팅에서는 다수의 워크플로우 들이 존재하고 워크플로우를 구성하는 태스크들이 클라우드 컴퓨팅의 자원에 효율적인 할당은 성능에 상당한 영향을 미칠 것이다.

이 WfMC에서도 워크플로우를 효율적으로 처리하기 위하여 참조모델을 발표하였는데 Workflow Engine, Process Definition, Workflow Interoperability, Invoked Applications, Workflow Client Applications, Administration and Monitoring으로 구성된다. 이중 워크플로우 자원 할당과 우선순위를 결정하는 워크플로우 엔진은 가장 중요한 부분으로 정의하고 있다. 본 논문에서는 다중 워크플로우를 효율적으로 서비스하고 클라우드 컴퓨팅의 자원 이용을 최대화하는 스케줄링 알고리즘을 제안하고자 한다.

워크플로우의 스케줄링은 그리드 컴퓨팅에서는 주로 최선(best-effort)방식의 연구가 주류를 이루고 있는데 비해 클라우드 컴퓨팅의 경우는 자원이 태스크에 전용되므로 다른 방식의 알고리즘이 요구된다. 또한 클라우드 환경은 이질적으로 구성되는 경우가 많고 데이터 처리 위주의 워크플로우의 경우는 저장공간과 데이터 전송 비용이 시간에 따라 기하급수적으로 증가되는 것이 고려되어야 한다[2]. 따라서 본 논문은 이질적인 자원으로 구성된 클라우드 환경에서 워크플로우를 중요도에 따라 분류하고 이를 응답성, 비용, 부하균등으로 구성된 QoS 메트릭을 생성하여 워크플로우의 요구와 자원의 효율적 이용을 동시에 만족시키는 알고리즘을 제안하고자 한다.

논문의 구성은 관련 연구를 2장에서 분석하고 제안하는 알고리즘을 3장에서 제시하며. 4장에서는 제안한 알고리즘의 실험 결과를 소개하며 5장에서는 결론을 맺는다.

II. 관련 연구

클라우드 컴퓨팅 환경에서 워크플로우는 컴퓨팅 자원상에서 스케줄링의 주요 대상이다. 이 워크플로우 스케줄링은 일종의 전역 태스크 스케줄링으로서 직접적으로 제어하지 않는 공유 자원상에서 상호 독립적인 태스크를 자원에 할당하고 실행을 관리하는 것으로 정의 될 수 있다. 다음은 워크플로우 스케줄링의 주요 연구 결과들이다.

[3]에서는 웹을 통하여 발생된 워크플로우를 고려하고 이 워크플로우는 그리드에 존재하는 다수의 자원들을 접근하는 웹 응용의 일부분으로 처리된다. [4]의 Pegasus는 그리드와 같은 분산 환경에서 복잡한 과학 기반 워크플로우를 처리하는 프레임웍을 제안하였다. [5]에서는 워크플로우 스케줄링을 두 가지 부류로 분류하여 알고리즘을 개발하였는데 첫째는 큐의 대기시간이나 최단 실행 시간을 만족시켜야 하는 실시간 워크플로우와 평균 도착시간 또는 평균 실행시간을 만족시켜야하는 워크플로우이다. [6]에서는 최선형 서비스와 QoS 만족형 서비스의 두 가지 종류의 워크플로우 알고리즘을 제안하였는데 그리드 워크플로우 시스템에서 적합한 것으로 보인다.

[7]에서는 트랜잭션이 많은 워크플로우 환경에 적합한 처리율을 최대화시키는 알고리즘을 제안하였는데 트랜잭션에 치중한 나머지 다중 워크플로우 환경에는 적합하지 못한 방식으로 보인다. [8]에서는 요구사항이 중첩되는 다중 QoS와 다중 워크플로를 지원하는 스케줄링 알고리즘을 제안하였는데 논문에서는 이 기법을 통해 스케줄링 접근 비율을 향상시킬 수 있음을 보이고 있다. [9]에서는 다수의 소규모 클라우드를 구축하고 사용자에게는 하나의 클라우드로 보이고 성능면에서는 대규모 클라우드를 얻고자하는 시도를 하였다. [10]에서는 동시에 다수의 독립적 클라우드를 사용하기 위한 보안 기법을 제안하였다.

관련 연구의 대부분은 클라우드가 가지는 비용, 성능, 고장 감내성, 그리고 환경 구성 등의 평가 요소를 주로 고려하였으므로 워크플로우의 특성을 고려하여 스케줄링한다면 보다 효율적인 자원 이용을 획득할 수 있을 것으로 기대된다.

III. 결정 테이블을 사용한 워크플로우 스케줄링

클라우드의 스케줄러가 워크플로우를 스케줄링 할 때 두 가지가 고려되어야 하는데 워크플로우간의 우선 순위를 정하는 것과 각 워크플로우의 태스크들을 어떤 노드의 자원에 할당할 것인가이다. 워크플로우의 우선순위를 결정하는 것은 다양한 기준이 존재하는데 사용자 입장에서는 지불 비용 대비 응답성, 신뢰성, 보안성 등이 주요한 고려대상이다. 반면 서비스 제공자 입자에서는 클라우드를 구성하는 노드의 부하 균등, 최대 사용자 지원, 확장성 등이 주요한 관심사이다.

따라서 하나 이상의 결정 기준이 존재하고 또한 다수의 해결이 존재하는 MCDM(Multi-Criteria Design Making)으로 정의할 수 있다. 본 논문에서는 다음의 식 1과 같이 문제를 정의한다.

$$\max q = f(x) = (f_1(x), \ldots (f_k(x)), \quad \text{(식 1)}$$
$$\text{subject to}$$
$$q \in Q = f(x) : x \in X, x \subseteq R^n$$

수학적으로 MCDM 문제는 위의 수식처럼 결정공간으로 표현 가능한데 q는 k개의 스케줄링 기준 함수의 벡터이고 Q는 가능한 집합이며 R^n 평면으로 표현 가능하다. X는 가능한 집합이고 x는 크기 n을 가지는 결정 벡터이다. x는 클라우드의 노드가 가지는 자원(CPU, 메모리, 저장공간, 네트워크 등)들의 벡터가 될 것이고 $f_k(x)$는 이들 자원을 스케줄링 기준이 되는 함수에 적용했을 때의 값이 된다. 따라서 $f_k(x)$중 최대 q 값이 스케줄링 대상으로 결정되고 이 q는 R^n 결정공간 내에 존재하며 가능한 집합은 Q로 표현된다.

본 논문에서는 우선순위와 자원 매핑을 결정하기 위한 기준(Criteria)을 정하고, 워크플로우가 제출되었을 때 가능한 해결(Options)들의 점수(Rating)를 계산한다. 그리고 제출되는 워크플로우의 특성에 따라 각 결정기준에 가중치(Weights)를 휴리스틱하게 적용한다. 최종적으로 가중치와 각 해결(Option)들의 점수를 곱하여 최종 득점(Scores)을 계산한다. 해결 중에서 최대 득점(Scores)을 획득한 노드가 자원 할당의 대상이 된다.

3.1 워크플로우 특성에 기반한 우선순위 분류

일반적으로 그리드 컴퓨팅의 경우 워크플로우가 시스템의 스케줄러에 제출되면 최선 방식으로 스케줄링하여 자원의 효율성만 고려하는 측면이 강하다. 클라우드 컴퓨팅의 경우는 자원이 전용되므로 자원을 미리 확보하여 스케줄링하는 것이 유리하다. 특히 데이터 처리 위주의 응용은 저장공간과 데이터 전송의 비용이 시간 경과에 따라 더욱 더 증가하므로 자원을 미리 확보하는 것이 성능향상의 중요한 요인이 될 것이다. 따라서 본 논문에서는 워크플로우의 요구 자원을 미리 확보하여 스케줄링하는 Resource Provisioning 워크플로우와 best-effort 워크플로우로 구분하여 표 1과 같이 우선순위를 결정한다.

표 1. 워크플로우의 분류
Table 1. workflow classification

우선순위	자원 할당	비고
1st level	Resource Provisioning	상당한 사업상 손실을 유발하는 경우
2st level	Resource Provisioning	다수 사용자의 불편을 초래하는 경우
3st level	best-effort	소규모 사용자의 불편을 초래하는 경우
4st level	best-effort	가벼운 장애의 경우

Resource Provisioning의 경우 사용자가 비용을 많이 지불하거나 중요도가 높은 워크플로우에 해당하며 만약 스케줄링이 실패할 경우 심각한 손실을 초래하는 경우가 1st level, 상당한 휴유증을 유발하는 경우는 2st level로 분류한다. best-effort는 저렴한 비용을 지불한 사용자나 중요도가 낮은 워크플로우에 해당하며 소규모 사용자의 불편을 초래하는 경우는 3st level, 미미한 기능적 장애를 초래한 경우는 4st level로 분류한다. 분류된 워크플로우들은 클라우드의 스케줄러로 순서대로 삽입되고 한번 정해진 순서는 작업이 완료될 때 까지 변하지 않으며 비선점 방식으로 처리된다.

3.2 결정 테이블을 이용한 자원 할당 알고리즘

워크플로우의 중요도에 의해 우선순위가 정해지면 워크플로우를 구성하는 태스크를 클라우드내의 자원에 할당하는 과정이 요구된다. 본 연구에서는 결정 테이블(Decision table)을 이용하여 여러 후보 중 최적의 후보를 결정한다. 결정의 기준이 C_i 이고 q 는 $f_k(x)$의 함수이다. x가 자원들의 벡터

라면 $f_k(x)$는 노드 k의 득점(Scores)이 된다. 그러므로 q는 모든 노드의 득점을 가지는 벡터이다. 이것을 일반화 하면 다음의 결정 테이블이 생성된다.

표 2. decision table의 예
Table 2. example of decision table

criteria	weight	$node_1$	$node_2$	⋯	$node_k$
C_1	W_1	$score_1^1$	$score_2^1$	⋯	$score_k^1$
C_2	W_2	$score_1^2$	$score_2^2$	⋯	$score_k^2$
⋯	⋯	⋯	⋯	⋯	⋯
C_n	W_n	$score_1^n$	$score_2^n$	⋯	$score_k^n$
scores(q)		$f_1(x)$	$f_2(x)$	⋯	$f_k(x)$

표 2에서 $score_k^n$ 는 노드 k의 각 결정기준 C_n에 대한 득점으로서 W_i*$Rating$ 에 의해 값이 결정된다. W_i는 각 결정기준에 대한 가중치로서 워크플로우의 특성에 따라 결정된다. 이 가중치의 결정이 스코어를 결정하는데 상당한 영향을 미친다. Resouce provisioning 워크플로우의 경우 응답성이 중요하고 Best-effort 워크플로우의 경우 부하 균등이 중요한 결정기준이 될 것이다. 모든 결정 기준의 중요도는 얻고자하는 목적에 따라 달리 결정되는데 일반적으로 최적해는 존재하지 않으므로 일반적으로 휴리스틱하게 부여하여 최적해에 근접함을 목표로 한다[11]. 따라서 본 연구에서는 경험적으로 표 3과 같이 가중치를 결정한다.

표 3. QoS 메트릭에 대한 가중치
Table 3. Weight for QoS metrics

Criteria	$Weight$		
	1st Level	2st Level	3st, 4st Level
C_1(응답성)	3	3	2
C_2(비용)	2	2	3
C_3(부하균등)	1	2	4

각 노드의 $Rating$은 임의의 노드에 자원을 할당하였을 때 결정기준에 대한 점수가 된다. 자원 벡터 x에 의해 노드의 점수가 계산되면 가중치를 곱하여 워크플로우의 특성에 맞는 득점을 계산할 수 있다. 마지막으로 모든 결정기준에 대한 점수를 모두 합산하여 $f_k(x)$를 식 2와 같이 구하여 최대의 $f_k(x)$가 자원 할당 노드로 선택된다.

$$f_k(x) = \sum_{i=1}^{n} score_k^i \qquad \text{(식 2)}$$

본 논문에서는 결정기준에 대한 각 노드의 점수($Rating$)를 계산하는데 그림 1은 응답성에 대한 점수를 계산하는 알고리즘이다. 먼저 워크플로우의 자원요구량을 계산하는데 자원은 CPU, memory, disk, network를 고려한다. $WF_i^{resources}$는 워크플로우의 자원요구량이고 $R_{\max}^{resources}$는 최대자원요구량이다. 식 3과 같이 각 자원에 대한 요구 자원 비율을 계산한다. 이것이 2.5보다 작으면 자원이 여유가 있다는 것이고 7.5보다 크면 자원이 여유가 없다는 것을 의미한다. 본 연구에서는 계산의 수월성을 위해 4구간으로 나누고 $Rating$을 4에서 1까지로 지정하였다.

rating_Criteria_1(WF_i, $Node_i$) {

$$WF_i^{mem} = \sum_{j=1}^{n} T_{mem_j} \quad WF_i^{cpu} = \sum_{j=1}^{n} T_{cpu_j}$$

$$WF_i^{disk} = \sum_{j=1}^{n} T_{disk_j} \quad WF_i^{innet} = \sum_{j=1}^{n} T_{innet_j}$$

$$WF_i^{outnet} = \sum_{j=1}^{n} T_{outnet_j}$$

$$R_{usage}^{resouces} = \frac{R_{allocated}^{resouces} + WF_i^{resources}}{R_{\max}^{resouces}} \qquad \text{(식 3)}$$

if ($R_{usage}^{resouces} \le 2.5$) $Node_i^{rate}$ = 4;
else if ($R_{usage}^{resouces} \le 5.0$) $Node_i^{rate}$ = 3;
else if ($R_{usage}^{resouces} \le 7.5$) $Node_i^{rate}$ = 2;
else $Node_i^{rate}$ = 1;
}

그림 1. 결정 기준 C_1(응답성)에 의한 rating
Fig. 1. Rating of Criteria 1(Responsibility)

그림 2는 결정 기준 C_2(비용)에 의한 rating으로서 각 노드의 분당 사용 비용에 워크플로우의 총 실행 시간을 곱하여 비용을 계산한다.

rating_Criteria_2(WF_i, $Node_i$) {
return $WF_i^{totalexecutiontime} * Node_i^{cost}$
}

그림 2. 결정 기준 C_2(비용)에 의한 rating
Fig. 2. Rating of Criteria (Cost)

그림 3은 결정기준 C_3(부하균등)에 의한 $Rating$으로서 워크플로우의 자원 요구량을 각 노드에 부과하였을때의 자원 요구량의 기대값 X를 $R_{usage}^{resources}$로 두고 평균 μ와 분산 σ^2을 구한다. 이 분산을 최대값 10으로 하는 구간에 정규화시켜서 2.5보다 작으면 부하균등이 최대값 4로 두고 7.5보다 크면 부하 불균등이 상당한 구간으로 최소값인 1로 지정하여 $Rating$을 계산한다.

그림 4는 매핑함수로서 모든 노드에 대하여 모든 결정기준에 대한 $Rating$을 계산하고 결정기준별 가중치를 곱하여 득점 $Score_i^j$을 구한다. 그리고 모든 결정기준에 대한 득점 $Score_i^j$들의 합을 계산하여 벡터 q의 요소인 $f(x)$를 구하고 최대 득점 노드를 자원 할당 노드로 결정한다.

rating_Criteria_3($WF_i, Node_i$) {

$$WF_i^{mem} = \sum_{j=1}^{n} T_{mem_j} \quad WF_i^{cpu} = \sum_{j=1}^{n} T_{cpu_j}$$

$$WF_i^{disk} = \sum_{j=1}^{n} T_{disk_j} \quad WF_i^{innet} = \sum_{j=1}^{n} T_{innet_j}$$

$$WF_i^{outnet} = \sum_{j=1}^{n} T_{outnet_j}$$

$$R_{usage}^{resouces} = \frac{R_{allocated}^{resouces} + WF_i^{resources}}{R_{\max}^{resouces}}$$

$$\mu = E(\sum_{node}^{all} R_{usage}^{resources}),\ Let\ X = R_{usage}^{resources}$$

$$var(X) = E((X-u)^2) = \sigma^2$$

if ($\sigma^2 \le 2.5$) $Node_i^{rate}$ = 4;

else if ($\sigma^2 \le 5.0$) $Node_i^{rate}$ = 3;

else if ($\sigma^2 \le 7.5$) $Node_i^{rate}$ = 2;

else $Node_i^{rate}$ = 1;

}

그림 3. 결정기준 C_3(부하균등)에 의한 rating
Fig. 3. Rating of Criteria (Load balancing)

mapping_WF_to_Node(WF_i) {

for (all $Node_i$) {

$Rating_1$ = rating_Criteria_1($WF_i, Node_i$);

$Rating_2$ = rating_Criteria_2($WF_i, Node_i$);

$Rating_3$ = rating_Criteria_3($WF_i, Node_i$);

$$Score_i^j = Weight_j * Rating_j$$

$$f_i(x) = \sum_{j=1}^{n} Score_i^j$$

}

node = find maximum $f_i(x)$;

return node;

}

그림 4. 매핑 함수
Fig. 4. Mapping function

Ⅳ. 실험

본 논문에서 제안하는 알고리즘의 성능 평가를 위하여 시뮬레이션 환경을 구축하였는데 그림 5는 제안하는 알고리즘의 성능 평가를 위한 워크플로우 스케줄링 시스템(WSS(Workflow scheduling system)의 구조이다. 다양한 워크플로우들이 클라우드 시스템에 도착하면 Workflow pool에 큐잉된다. Task scheduler 는 각 워크플로우의 우선순위를 결정하고, Task distributor 는 태스크들을 제안하는 알고리즘에 의해 자원에 할당한다.

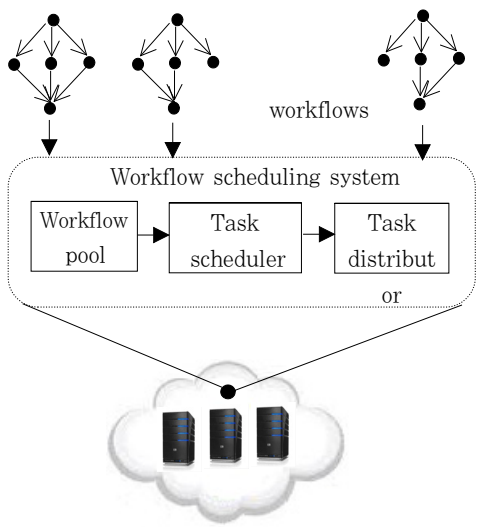

그림 5. 워크플로우 스케줄링 시스템
Fig. 5. Workflow scheduling system

4.1 실험 환경

제안하는 기법의 효율성을 검증하기 위하여 실험 모델은 WfMC[1]에 기반하여 구성하였다. 실험 모델은 클라우드를 구성하는 자원과 다양한 요구로 구성되는 다중 워크플로우를 고려하는데 클라우드 자원의 경우는 다양한 성능으로 구성되는 컴퓨팅 자원을 의미하며 워크플로우의 경우 단일 워크플로우 뿐만 아니라 다중 워크플로우도 고려한다.

구체적으로 클라우드 자원은 10대의 호스트로 구성되며 각각의 호스트는 {cpu, memory, storage, network, cost}의 쌍으로 표현되며 계산의 수월성을 위해 {1, 1, 1, 1, 100}부터 {3, 4, 4, 3, 300}까지로 표현한다. 또한 클라우드는 30개의 서비스로 구성되며 모든 서비스는 하나의 태스크를 동시에 실행할 수 있다. 워크플로우는 1개에서 200개로 구성되고 포아송 분포로 도착하여 서비스를 요청하며 각 워크플로우는 단일 또는 다중 워크플로우로 구성된다. 또한 워크플로우는 데이터 처리 위주 또는 계산 위주의 특성을 랜덤하게 설정하며 각 워크플로우의 태스크는 1개에서 10개까지의 태스크로 구성되며 각 태스크의 실행시간은 10에서 100단위까지로 설정하고 1단위 실행하는데 1초의 시간을 설정하여 태스크의 실행시간을 계산하였다. 이 실험은 매트랩과 시뮬링크상에서 실험되었다.

제안기법의 검증을 위하여 두 가지 비교 대상을 실험에 구축하였는데 첫 번째는 워크플로우의 도착시간 순으로 스케줄링하되 자원 선택은 제안하는 기법을 적용하는 것이고 두 번째는 도착시간 순으로 스케줄링하고 자원의 선택을 임의로 결정하는 것이다. 이 두 가지 비교 대상 기법들에 의해 본 논문이 제안하는 스케줄링 기법의 효율성을 분석할 수 있다.

4.2 실험결과

사용자의 응답성과 시스템의 가용성은 클라우드 컴퓨팅의 성능 분석에 중요한 두 가지 요소이다. 그림 6은 워크플로우의 수가 1개에서 200개 까지 증가함에 따른 각 워크플로우의 평균 실행 시간을 나타내고 있다. 이 실행 시간은 워크플로우가 도착해서 각 태스크가 실행완료된 시간으로 지연 시간을 포함하고 있다. 전반적으로 워크플로우의 수가 증가할수록 평균 실행시간이 증가하고 있고 제안 기법은 첫 번째 비교대상 기법(by arrival time)에 비해 11%, 두 번째 비교대상 기법(by random)에 비해 18%의 성능 향상을 나타내었다. 이러한 결과는 제안 기법은 워크플로우의 긴급성에 따라 4단계로 구분하여 우선순위가 높은 워크플로우가 먼저 스케줄링되고 자원 할당에 있어서도 임의적으로 선택하기 보다는 워크플로우의 특성에 따라 자원을 선점하여 할당하기 때문인 것으로 분석된다.

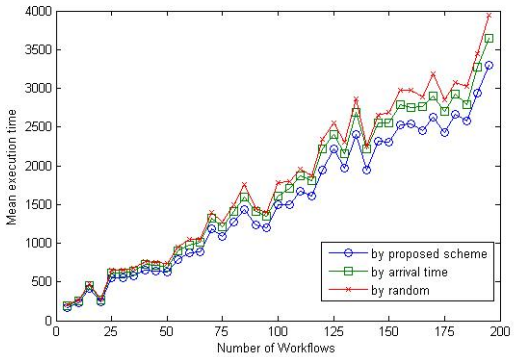

그림 6. 워크플로우의 수에 따른 평균 실행시간
Fig. 6. Mean execution time

그림 7은 워크플로우의 수에 따른 시스템의 가용성을 나타낸 것이다. 가용성은 클라우드에 소속된 호스트들의 평균 이용률을 의미하지만 본 논문에서는 실험의 간소화를 위해 태스크가 자원을 요구했을 때 실패 회수로 시스템의 가용성을 표현하고자한다. 이는 시스템의 가용성이 높으면 자원 요구 실패 회수도 낮다는 것에 기반을 둔다. 그림 7에서 보듯이 워크플로우의 수가 증가할수록 자원 요구 실패(Number of miss)의 수도 증가한다. 그러나 제안 기법은 첫 번째 비교 기법(by arrival time)에 비해 13%, 두 번 째 비교 기법(by random)에 비해 23%의 성능 향상을 나타내었다. 이것은 제안하는 기법이 표3에 의해 응답성, 비용, 부하균등 기준에 따라 워크플로우의 중요도를 구분하고 긴급성을 요구하는 1st, 2st 레벨은 응답성이 높도록 자원을 할당하고 3st, 4st 기법은 부하 균등에 가중치를 높게 하여 전반적인 시스템의 가용성을 향상시켰기 때문이다.

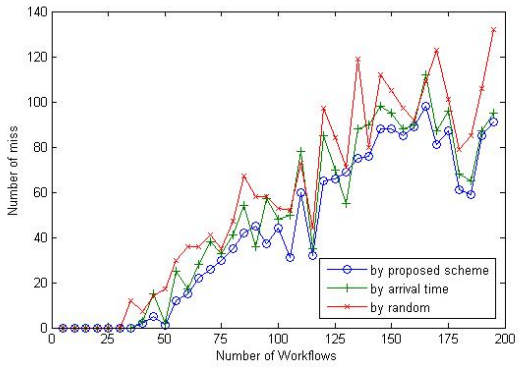

그림 7. 워크플로우의 수에 따른 자원 획득 실패
Fig. 7. Resource miss rates

V. 결론

본 논문에서는 클라우드 컴퓨팅에서 다중 워크플로우를 우선순위와 워크플로우의 특성을 고려하여 스케줄링할 수 있는 기법을 제안하였다. 클라우드 컴퓨팅은 사용자의 응답성과 시스템의 가용성, 그리고 비용이 중요한 성능 측정 요소로 작용하는데 본 논문에서는 사용자의 응답성을 고려하여 워크플로우를 4단계로 나누어 높은 비용을 지불하거나 긴급성을 요하는 워크플로우의 우선순위를 높여 스케줄링한다. 또한 시스템의 비용대비 가용성을 높이기 위해 워크플로우를 구성하는 태스크들을 클라우드내의 호스트에 할당할 때 응답성, 비용, 부하균등의 특성을 고려하여 분배한다.

제안된 기법의 성능평가를 위해 철저한 실험을 수행하였는데 성능 평가 요소로 워크플로우들의 평균 실행시간과 자원 할당 실패 회수를 분석하여 응답성과 가용성을 분석하였다. 실험결과 제안 기법은 도착순서에 따라 스케줄링하는 기법과 자원을 임의적으로 할당하는 기법에 비해 10%이상의 성능향상을 보였다.

현재의 연구는 자원 할당 결정 과정에서 평가 메트릭의 가중치를 임의로 고정하였는데 가중치의 변화에 따른 성능 분석과 다양한 워크플로우를 모델링하여 제안기법에 적용하고 신뢰성을 고려한 스케줄링 기법을 개발하는 것이 향후 과제이다.

참고문헌

[1] Workflow Management Coalition, "Workflow Management Coalition Teminology & Glossary", February, 1999.

[2] InSeong Kang, TaeHo Kim, HongChul Lee, "Data processing techniques applying data mining based on enterprise cloud computing ," Journal of The Korea Society of Computer and Information, v.16, no.8, pp.1-10, 2011.

[3] S. Elnikety, E. Nahum, J. Tracey and W. Zwaenepoel, "A method for transparent admission control and request scheduling in e-commerce web sites," Proceedings of the 13th International Conference on World Wide Web, pp. 276-286, 2004.

[4] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good,A. Laity, J. C. Jacob, and D. S. Katz. "Pegasus: A framework for mapping complex scientific workflows onto distributed systems", pp.219 - 237, 2005.

[5] Patel, Y. Darlington, J., "A novel stochastic algorithm for scheduling QoS-constrained workflows in a web service-oriented grid", WI-IAT 2006 Workshopspp, pp.437-442, 2006.

[6] J. Yu and R. Buyya, "Workflow Schdeduling Algorithms for Grid Computing", Technical Report, GRIDS-TR-2007-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, May 2007.

[7] Ke Liu, Jinjun Chen, Yun Yang and Hai Jin, "A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G", Concurrency and Computation:Practice and Experience, Wiley, pp.1807-1820, 2008.

[8] Boris Mejias, Peter Van Roy, "From Mini-clouds to Cloud Computing", 2010.

[9] Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing," ispa, pp.629-634, 2009 IEEE International

Symposium on Parallel and Distributed Processing with Applications, 2009.
〔10〕 M. Jensen, J. Schwenk, J.M. Bohli, L.L. Iacono, "Security prospects through cloud computing by adopting Multiple Clouds", 2011.
〔11〕 http://en.wikipedia.org/wiki/MCDA.

저 자 소 개

김 정 원
1995년 : 부산대학교 전자계산학과(학사)
1997년 : 부산대학교 대학원 전자계산학과 (석사)
2000년 : 부산대학교 대학원 전자계산학과 (박사)
2000년~2001년 : 기술신용보증기금 기술평가역(차장)
2002년~현재 : 신라대학교 컴퓨터정보공학부 교수
관심분야 : 내장형시스템, 멀티미디어, 운영체제
Email : jwkim@silla.ac.kr