J Korea Industr Inf Syst Res Volume 17 Number 5
http://dx.doi.org/10.9723/jksiis.2012.17.5.029

클라우드 컴퓨팅에서 결정테이블을 이용한 워크플로우 스케줄링

(A workflow scheduling based on decision table for cloud computing)

김 정 원*
(Jeong Won Kim)

요 약 클라우드 컴퓨팅은 자원을 사용자 스스로 관리하지 않고 클라우드 공간내에서 서비스 제공자에 의해 제공되고 이질적인 자원을 가상화하여 자원 사용의 효율성을 제공하여 최근 각광을 받고 있다. 한편 클라우드 컴퓨팅에서 다양한 워크플로우들이 실행되고 서버는 이질적인 특성을 가지므로 워크플로우 효율적인 스케줄링은 사용자 응답성과 자원 이용률에 중요한 요소이다. 본 연구에서는 워크플로우의 중요도에 의해 스케줄링하여 비용대비 응답성을 향상시키고 자원 할당시 결정 테이블을 사용하여 워크플로우의 특성에 부합하도록 스케줄링하여 비용대비 가용성을 향상시키는 2단계 스케줄링 알고리즘을 제안한다. 제안하는 기법의 성능 검증을 위해 다양한 실험을 수행하였는데 비교 기법 대비 성능 향상을 확인할 수 있었다.

핵심주제어 : 워크플로우, 클라우드 컴퓨팅, 응답성, 가용성

Abstract Cloud computing has gained great popularity because users don't need to install any softwares as well as maintain hardwares and service providers also can utilize its resources through virtualization of servers. As workflows feature variableness and servers are heterogenous, efficient scheduling of workflows in cloud computing is important factor in view of responsibility and resource utilization. In this paper, we propose a new workflow scheduling named 2-step scheduling which prioritizes each workflow through its significance degree and allocates resources to workflows through decision table. The goal of this 2-step scheduling is to improve responsibility as well as availability versus cost. Simulation results show that the proposed scheme in contrast of counterparts can improve the responsibility as well as availability of resource.

Key Words : workflow, cloud computing, responsibility, availability

1. 서 론

클라우드 컴퓨팅은 인터넷 기반의 컴퓨팅으로 클라우드 시스템이 제공하는 자원, 즉 소프트웨어, 하드웨어, 정보를 마치 가스, 전기 등의 공공 유틸리티처럼 사용하는 환경이다. 사용자는 개인 소유의 컴퓨터에 소프트웨어를 설치할 필요 없이 클라우드 서비스 제공자의 컴퓨팅 환경에 인터넷으로 접속하여 개개인의 자료나 정보, 그리고 각종 소프트웨어를 사용할 수 있다. 이것은 개인이나 기업이 구축하면 많은 비용이 발생하지만 스토리지, 메모리, 처리 하드웨어 및 대역폭을 중앙집중하므로 저비용의 효율적인 컴퓨팅 환경을

* 신라대학교 컴퓨터공학과, 제1저자 (jwkim@silla.ac.kr)

제공하는 기술이다[1].

이러한 자원들이 클라우드라는 단일 환경으로 통합되는데 일반적으로 다양한 성능을 가지는 자원이 클라우드내에 존재할 것이다. 또한 사용자의 작업 내용도 단일 태스크로 구성되는 경우도 있겠지만 대부분 다수의 데이터가 통합되거나 여러 개의 작업의 흐름이 통합되어 실행되는 경우도 많다. 이것을 워크플로우라고 하는데 WfMC[2]에서는 문서, 정보, 또는 태스크가 하나의 노드에서 다른 노드로 전체 또는 부분이 자동으로 일련의 규칙에 의해 전달되는 흐름으로 정의하고 있다. 따라서 클라우드 컴퓨팅에서는 다수의 워크플로우 들이 존재하고 워크플로우를 구성하는 태스크들이 클라우드 컴퓨팅의 자원에 효율적으로 할당하는 것은 성능에 상당한 영향을 미칠 것이다.

워크플로우를 스케줄링하는 문제는 그리드 컴퓨팅의 주된 연구 주제로 일반적으로 단일 워크플로우에서 최적의 성능을 얻기 위해 정적 또는 동적 알고리즘들이 제안되었는데 주로 스케줄링 완료시간 보장, 대기시간 감소 등의 기법들이 제시되었다. 이 기법들은 주로 최선 방식(best-effort)의 스케줄링 알고리즘으로 일정 수준의 자원이 얼마나 신속하게 할당되는가와 어느 정도의 자원이 워크플로우에 전용될 수 있는가가 주된 관심인 클라우드에서는 다른 관점의 알고리즘이 필요하다[3, 4].

한편 클라우드 컴퓨팅은 광범위한 확장성을 가지고 사용자의 서비스 수준이 다양하며, 가상화를 통해 서비스 환경이 동적으로 설정되므로 그리드 컴퓨팅과는 다른 관점에서 워크플로우를 스케줄링할 필요가 있다. 특히 데이터 처리 위주의 워크플로우의 경우는 스토리지와 데이터 전송 비용이 시간에 따라 기하급수적으로 증가하는 것이 고려되어야 한다. 따라서 본 연구에서는 워크플로우가 가지는 특성에 기반한 새로운 스케줄링 알고리즘을 제안하고자 한다.

본 연구에서는 두 단계로 구분하여 실행하는데 먼저 워크플로우의 중요도에 의해 우선순위를 분류하는 단계와 분류된 워크플로우의 태스크들을 자원에 할당하는 단계이다. 첫 번째 단계에서는 심각한 사업상의 손실과 휴유증을 초래할 수 있는 경우는 자원을 전용(Resource provisioning)하고 소규모의 사용자 불편과 미미한 기능적 장애를 초래할 수 있는 경우는 최선(best-effort) 방식으로 스케줄링한다. 두 번째 단계에서는 사용자의 QoS와 클라우드 컴퓨팅의 효율을 최대화할 수 있도록 결정 테이블을 활용한 자원 할당 알고리즘을 사용한다.

2장에서는 관련 연구를 살펴보고 3장에서는 제안하는 알고리즘을 구체적으로 소개한다. 4장에서는 시뮬레이션을 통해 알고리즘을 효율성을 검증하고 5장에서는 결론과 향후 연구 과제에 대해 논의한다.

2. 관련 연구

워크플로우 스케줄링은 일종의 전역 태스크 스케줄링으로서 직접적으로 제어하지 않는 공유 자원상에서 상호 독립적인 태스크를 자원에 할당하고 실행을 관리하는 것으로 정의될 수 있다. 그리드 컴퓨팅 환경에서 단일 워크플로우의 최적의 성능을 얻기 위해 정적 또는 동적 알고리즘들이 제안되었다[5]. 이 연구들을 분류해 보면 워크플로우의 상속자는 입구노드를 만들고 피상속자는 출구노드를 만드는 복합 그래프 방식, 복합 그래프 방식이지만 레벨 단위로 분류하여 스케줄링하는 방식, 복합그래프를 만들지만 라우드로빈 방법으로 스케줄링하는 방식, 그리고 태스크의 실행시간을 고려하여 짧은 실행시간을 가진 노드가 긴 워크플로우의 태스크로 링크되는 랭크 기반의 스케줄링 방식으로 분류될 수 있다.

다음은 클라우드 환경에서 워크플로우 스케줄링의 주요 연구 결과이다. [6]에서는 비용기반 워크플로우 스케줄링 알고리즘을 제안하였는데 데드라인을 만족하면서도 실행 비용의 최소화를 고려하고자 하였다. [7]에서는 워크플로우 결과의 비용을 만족시키면서 실행시간의 최소화를 얻고자하는 스케줄링 기법을 제안하였는데 스케줄링 최적화 문제를 해결하기 위하여 유전 알고리즘을 적용하였고 그리드환경에서 실험 결과를 제시하였다.

[8]에서는 워크플로우가 요구하는 QoS를 만족시키고 자원 이용도를 높이기 위하여 응용들 사이에 자원 공유를 조정하고자하는 알고리즘을 제안하였다. [9]에서는 다중 워크플로우를 위해 계획기반 알고리즘을 제안하였는데 워크플로우 간의 순위를 미리 결정하고 이를 기반으로 스케줄 대상을 선정하는 방식이다. [10]에서는 요구사항이 중첩되는 다중 QoS와 다중 워크플로를 지원하는 스케줄링 알고리즘을 제안하였는데 논문에서는 이 기법을 통해 스케줄링 접근 비율을 향상

시킬 수 있음을 보이고 있다. [11]에서는 클라우드 자원내에서 실행되는 응용프로그램의 신뢰성과 확장성을 향상시키고자 클라우드 위상에 대한 다양한 형태를 제안하였다.

따라서 그리드 환경에서 워크플로우 시스템들은 사용자가 자원이나 서비스를 선택하는 경우가 많았으나 본 연구는 클라우드 환경에서 워크플로우의 특성을 반영하여 워크플로우 단위의 스케줄링이나 자원할당을 효율적으로 수행하는 알고리즘을 제안하고자 한다.

3. 2단계 워크플로우 스케줄링 알고리즘

클라우드 컴퓨팅의 성능은 사용자 입장과 서비스 제공자 입장에서의 평가가 있는데 사용자 측면에서는 지불 비용 대비 응답성, 신뢰성, 보안성 등이 주요한 평가 기준이고, 서비스 제공자 입장에서는 클라우드를 구성하는 노드의 부하 균등, 최대 사용자 지원, 확장성 등이 주요 평가 기준일 것이다. 본 연구에서는 이러한 평가 기준들 중에서 응답성, 비용, 부하균등 세 가지 기준을 고려하는 스케줄링을 위해 2 단계 알고리즘을 제시한다.

응답성과 비용 측면을 고려하기 위해 워크플로우의 중요도를 기준으로 우선 순위를 정하는 첫 번째 단계와 부하 균등과 비용대비 가용성을 높이기 위해 워크플로우의 태스크들을 자원에 할당하는 단계이다. 워크플로우의 우선순위를 결정하는 것은 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^n평면으로 표현 가능하다. X는 스케줄링 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 최적의 값이 스케줄링 대상으로 결정되고 이 q는 R^n 결정 공간 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)를 계산한다. 해결 중에서 최적의 득점을 획득한 노드가 자원 할당의 대상이 된다.

3.1 중요도에 의한 우선순위 분류

워크플로우가 클라우드의 스케줄러에 제출되면 어떤 워크플로우를 먼저 처리할 것인가를 결정해야 하는데 그리드 컴퓨팅이 경우 best-effort 방식으로 스케줄링되는데 비하여 클라우드의 경우는 자원이 전용되므로 자원을 미리 확보하여 스케줄링하는 것이 유리하다. 특히 데이터 처리 위주의 응용은 스토리지와 데이터 전송의 비용이 시간 경과에 따라 더욱 더 증가하므로 자원을 미리 확보하는 것이 성능향상의 중요한 요인이 될 것이다. 따라서 논문에서는 워크플로우의 요구 자원을 미리 확보하여 스케줄링하는 Resource Provisioning Job과 best-effort Job으로 구분하여 우선순위를 결정한다(표 1).

<Table 1> 중요도에 의한 워크플로우의 분류

우선순위	자원 할당	비고
level-1	Resource Provisioning	심각한 손실을 초래하는 경우
level-2	Resource Provisioning	상당한 휴유증을 유발하는 경우
level-3	best-effort	감내 가능한 불편을 초래하는 경우
level-4	best-effort	미미한 기능적 장애를 초래

Resource Provisioning 워크플로우는 비교적 높은 사용료를 지불한 사용자의 워크플로우와 중요도가 높은 워크플로우에 해당하며 마감시간을 만족하지 못할

경우 심각한 손실을 초래하는 경우가 level-1, 상당한 휴유증을 유발하는 경우는 level-2로 분류한다. best-effort는 저렴한 비용을 지불한 사용자나 중요도가 낮은 워크플로우에 해당하며 소규모 사용자의 불편을 초래하는 경우는 level-3, 미미한 기능적 장애를 초래한 경우는 level-4로 분류한다. 분류된 워크플로우들은 클라우드의 스케줄러로 순서대로 큐잉되고 한번 정해진 순서는 작업이 완료될 때 까지 변하지 않으며 비선점 방식으로 처리되며 동일한 우선순위가 발생할 경우 라운드로빈 방식으로 처리된다

.

3.2 결정 테이블에 의한 자원 할당

우선순위가 결정되면 워크플로우를 구성하는 태스크를 자원에 할당하는 과정이 필요하므로 본 연구에서는 결정 테이블을 이용하여 최적의 노드를 결정한다. 후보 결정에는 응답성, 비용, 부하균등 등 다양한 기준이 있을 수 있으므로 이 방법은 각 기준마다 가중치를 부여하여 최대 득점을 받은 노드를 선택한다. 결정 기준이 C_i 이고 x가 자원들의 벡터, $f_k(x)$를 노드 k의 득점, 그리고 q를 모든 노드의 득점 벡터라면 q 는 $f_k(x)$의 함수이다. 이것을 정리하면 표 2와 같이 표현된다.

<Table 2> 결정 테이블

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 각 결정기준 C_n에 대한 노드 k의 득점으로서 W_i*$Rating$ 로 계산된다. 여기서 W_i는 각 결정기준에 대한 가중치로서 워크플로우의 특성에 따라 결정되며 스코어를 결정하는데 상당한 영향을 미친다. 예를 들면 Resource provisioning 워크플로우의 경우 응답성이 중요하고 Best-effort 워크플로우의 경우 부하 균등이 중요한 결정기준이 될 것이다. 이에 대한 완전한 해법이 없으므로 본 연구에서는 경험적으로 표 3과 같이 가중치를 결정한다.

<Table 3> QoS 메트릭에 대한 가중치

결정기준	*Weight*			
	Level-1	Level-2	Level-3	Level-4
C_1 (응답성)	5	4	2	3
C_2 (비용)	2	3	3	3
C_3 (부하균등)	2	2	4	3

득점을 결정하는 $Rating$은 노드에 자원을 할당하였을 때 각 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 최적의 $f_k(x)$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 응답성, 비용, 부하균등에 대한 점수를 계산하는 알고리즘이다.

그림 1의 함수 ratingByC1()에서 워크플로우의 자원 요구량을 먼저 계산하는데 CPU, memory, disk, network 자원을 고려한다. $R_{allocated}^{resources}$는 현재 사용중인 자원량의 벡터이고 $WF_i^{resources}$는 워크플로우가 요구하는 자원의 벡터이다. 식 3과 같이 $R_{allocated}^{resources}$와 $WF_i^{resources}$의 합을 노드의 최대자원량인 $R_{\max}^{resources}$으로 나누면 기대되는 자원 할당 정도가 계산된다. 본 연구에서는 이 값을 4단계로 구분하여 응답성을 결정하는데 0.25보다 작으면 자원이 여유가 있으므로 C_1에 의한 점수를 최대인 4로 결정하고 0.75보다 크면 자원이 여유가 없으므로 가장 낮은 1로 결정한다.

그림 1의 함수 ratingByC2()는 결정 기준 C_2(비용)에 의한 rating으로서 각 노드의 분당 사용 비용에 워크플로우의 총 실행 시간을 곱하여 비용을 계산한다.

그림 1의 함수 ratingByC3()는 결정기준 C_3(부하균등)에 의한 $Rating$으로서 워크플로우의 자원 요구량을 각 노드에 부과하였을 때의 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 최대값 4로 판단하고 7.5보다 크면 부하 불균등이 예상되므로 최소값인 1로 결정한다.

ratingByC1(WF_i,$Node_i$) {

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

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

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

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

if ($R_{usage}^{resouces} \le 0.25$) $Node_i^{rate}$ = 4;
else if ($R_{usage}^{resouces} \le 0.50$) $Node_i^{rate}$ = 3;
else if ($R_{usage}^{resouces} \le 0.75$) $Node_i^{rate}$ = 2;
else $Node_i^{rate}$ = 1;
}

ratingByC2(WF_i,$Node_i$) {
return $WF_i^{totalexecutiontime} * Node_i^{cost}$
}

ratingByC3(WF_i,$Node_i$) {

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

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

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

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

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

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

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;
}

<Fig 1> rating 계산 알고리즘

그림 2는 워크플로우를 노드에 할당하는 함수로 모든 노드에 대하여 각 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 구한다. 벡터 q의 요소중 최적의 값이 워크플로우를 실행할 노드로 선택된다.

allocating_WF_to_Node(WF_i) {
for (all $Node_i$) {
$Rating_1$ = ratingByC1(WF_i,$Node_i$);
$Rating_2$ = ratingByC2(WF_i,$Node_i$);
$Rating_3$ = ratingByC3(WF_i,$Node_i$);

$$Score_i^j = Weight_j * Rating_j$$

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

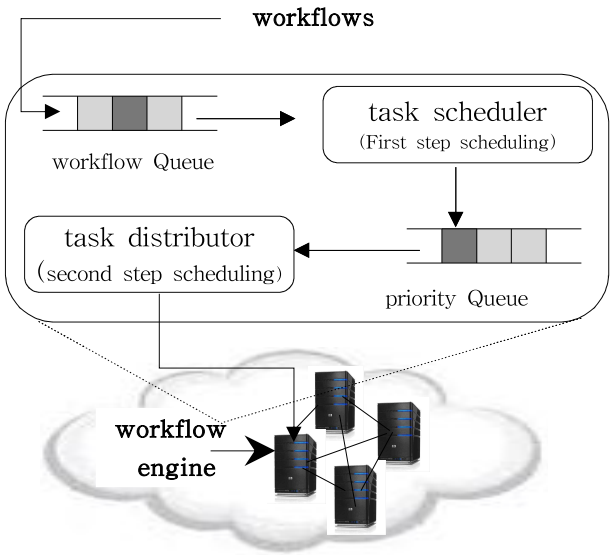
}
node = find Optimal $f_i(x)$;
return node;
}

<Fig 2> 노드 선택 함수

4. 실 험

WfMC의 참조 모델을 기반하여 워크플로우 엔진을 구현하고 제안 기법의 성능을 측정하였다. 그림 3은 실험 환경으로서 주요 기능은 워크플로우 인스턴스를 생성하고, 관리하며 실행하는 것으로 워크플로우를 실행하는 클라우드의 기능상 핵심을 차지한다. 제안하는 기법의 성능평가를 위해 메트랩과 시뮬링크를 사용하여 실험 환경을 구축하였다.

그림 3에서 보듯이 생성된 워크플로우는 큐에 도착 순서대로 삽입되고 스케줄링 주기마다 태스크 스케줄러는 워크플로우의 중요도에 기반한 우선순위를 결정하여 우선순위 큐에 삽입된다. 태스크 분배기(Task distributor)는 그림 4의 알고리즘에 의해 태스크가 실행할 노드를 결정하여 해당 노드로 태스크의 실행을 명령한다.



<Fig 3> 실험 환경

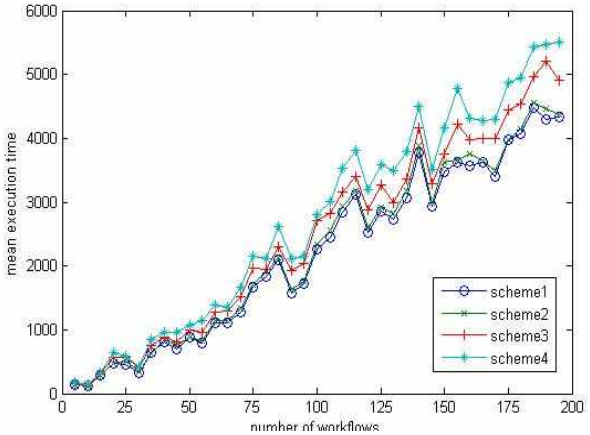
4.1 실험 환경

클라우드를 구성하는 노드는 확장성을 고려하여 저렴한 PC서버부터 워크스테이션을 고려하였는데 CPU는 워크플로우는 단일 태스크 및 다중 태스크를 가정한다. 실험의 목적은 논문에서 제안하는 2단계 스케줄링의 응답성과 가용성의 성능 향상을 검증하고 비교 대상 기법과의 성능 차이를 분석하는 것이다. 이전 연구들과 차이를 분석하는 것이 확실한 방법이지만 정확한 구현이 현실적으로 어려우므로 다음 기법들(표 4)과 비교한다.

<Table 4> 실험에서 사용된 스케줄링 기법들

기법	우선순위 기준	자원할당 방법
scheme1	importance	decision table
scheme2	importance	first fit
scheme3	arrival time	first fit
scheme4	round robin	random

scheme1은 본 논문에서 제안하는 기법이고 scheme2는 자원할당에서 최초 적합 노드를 선택하여 scheme1의 성능을 평가하고자 한다. scheme3, scheme4는 워크플로우의 특성을 고려하지 않는 것으로 도착순서나 라운드 로빈 방식으로 스케줄링한다.



<Fig 4> 응답성 실험 결과

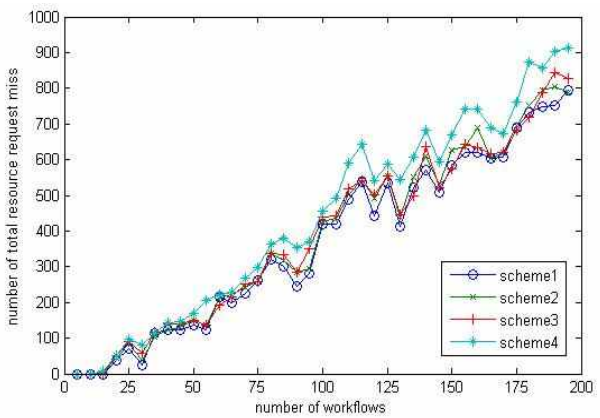
구체적으로 클라우드 자원은 10대의 노드로 구성되며 노드의 자원은 {cpu, memory, storage, network, cost}의 쌍으로 표현되며 CPU는 10kMIPS, memory는 GB, storage는 100GB, network는 100Mbps, cost는 10달러의 배수이다. 계산의 수월성을 위해 {1, 1, 1, 3, 100}부터 {5, 4, 5, 3, 500}까지의 정수로 표현한다. 각 노드는 30개의 서비스로 구성되며 모든 서비스는 하나의 태스크를 동시에 실행할 수 있다. 워크플로우는 1개에서 200개까지 동시에 발생될 수 있으며 포아송 분포로 도착하여 서비스를 요청한다. 또한 워크플로우는 데이터 처리 위주 또는 계산 위주의 특성을 랜덤하게 설정하며 각 워크플로우의 태스크는 1개에서 20개까지의 태스크로 구성되며 각 태스크의 실행시간은 10에서 100단위까지로 설정하고 1단위 실행하는데 1초의 시간을 설정하여 태스크의 실행시간을 계산하였다.

4.2 실험 결과

그림 4는 워크플로우의 수가 증가함에 따라 워크플로우의 평균 실행 시간을 표현한 것이다. 전체적으로 워크플로우의 수가 증가함에 따라 실행 시간이 증가하는 것은 당연하며 중요도가 높은 태스크를 먼저 수행한 scheme1, scheme2가 나머지 기법에 비해 나은 성능을 보이고 있다. 이것은 scheme3, scheme4가 워크플로우의 긴급성을 무시하고 도착시간이나 라운드 로빈 방식으로 수행하기 때문으로 분석된다. 또한 scheme1이 scheme2에 비해 낮은 실행 시간을 보이고 있는데 후보 자원 중에서 최적의 응답성을 제공하는

노드를 선택하기 때문이다.

그림 5는 가용성 평가에 대한 실험으로 워크플로우의 수가 증가함에 따른 자원 요구 실패 회수를 표현한 것이다. 자원의 부족은 클라우드 전체에서 부족한 것도 있지만 자원 선택에 있어 비효율이 원인이 될 수 있으므로 가용성 평가에 기준으로 채택하였다. 워크플로우의 수가 작을 때는 실패회수가 크게 차이를 보이고 있지 않지만 증가할수록 scheme1, scheme2가 scheme3, scheme4보다 낮은 빈도를 보이고 있다. 가용 자원 후보 중 임의로 선택하는 scheme4는 노드들의 부하를 적절하게 반영하지 못하여 최악의 성능을 보이고 있다. 결정 테이블을 사용한 scheme1 기법이 최초 적합 노드를 선택하는 scheme2, scheme3보다 나은 성능을 유지하는 것은 부하 균등 유지에 최적의 노드를 선택하기 때문인 것으로 분석된다.



<Fig 5> 가용성 실험 결과

종합적으로 응답성 측면에서 제안 기법인 scheme1은 나머지 기법 대비 각각 2.5%, 11.8%, 19.5%의 성능 향상을 보였고, 가용성 측면에서 각각 4.4%, 6.9%, 15.5%의 성능 향상을 나타내었다.

5. 결 론

본 연구에서는 클라우드 컴퓨팅에서 워크플로우를 효율적으로 스케줄링하는 기법을 제안하였다. 워크플로우를 스케줄링하는 기존의 연구들이 주로 마감시간의 보장, 큐내에서 대기시간 감소, 가용성 증가 등을 목적으로 한 반면 본 연구는 워크플로우 자체의 긴급성 및 중요도를 고려하고 비용대비 응답성과 가용성을 향상시키기 위해 2단계의 스케줄링 알고리즘을 고안하였다.

이 기법은 워크플로우의 응답성을 보장하기 위해 긴급성에 따라 우선순위를 분류하고 워크플로우의 특성과 자원의 상태를 반영하여 자원을 할당하기 위해 결정 테이블을 구축하여 최적의 자원을 선택함으로써 클라우드의 가용성을 향상시키는 알고리즘이다.

실험을 통해 제안 기법의 성능 향상을 확인할 수 있었고 향후 과제로는 고장 감내 가능한 기법의 개발과 실제 클라우드에 적용하는 것이다.

참 고 문 헌

[1] 유기동, “문서 자동요약 기술을 적용한 클라우드 스토리지 기반 지능적 아카이빙 시스템,” 한국산업정보학회논문지, Vol.17 No.13, 2012.

[2] Workflow Management Coalition, “Workflow Management Coalition Teminology & Glossary”, February, 1999.

[3] 나문성, 김승훈, 이재동, “클라우드 환경에서 대규모 콘텐츠를 위한 효율적인 자원처리 기법,” 한국산업정보학회논문지, Vol.15 No.4, 2010.

[4] Ewa Deelman, "Grids and Clouds:Making Workflow applications Work in Heterogeneous Distributed Environments," The Internation Journal of High Performance Computing Applications, Vol24, No.3. Fall 2010.

[5] Jablonski, S. and C. Bussler, "Workflow Management Systems: Modeling, Architecture and Implementation", Thomsom Press, 1996.

[6] Jia Yu, Rajkumar Buyya and Chen Khong Tham, “Cost-based Scheduling of Scientific Workflow Applications on Utility Grids”, In 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, Dec. 5-8, 2005.

[7] Jia Yu and Rajkumar Buyya, “A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms”,Proceedings of the 15th IEEE International symposium on

High Performance Distributed Computing (HPDC 2006), IEEE CS Press,Los Alamitos, CA, USA, June 19-23, 2006, Paris, France.
[8] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant and K. Salem, "Adaptive control of virtualized resources in utility computing environments," SIGOPS Oper. Syst. Rev., vol. 41, pp. 289-302, 2007.
[9] Zhifeng Yu and Weisong Shi, "A Planner-Guided Scheduling Strategy for Multiple Workflow Applications," icppw, pp.1-8, International Conference on Parallel Processing - Workshops, 2008.
[10] 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.
[11] J. Kosinska, J. Kosinski, K, Zielinski, "The Concept of Application Clustering in Cloud Computing Environments", 2010.

김 정 원 (Jeong Won Kim)

- 정회원
- 부산대학교 전자계산학과 학사
- 부산대학교 대학원 전자계산학과 석사
- 부산대학교 대학원 전자계산학과 박사
- 현재 : 신라대학교 컴퓨터정보공학부 부교수
- 관심분야 : 클라우드 컴퓨팅, 내장형시스템, 멀티미디어, 운영체제, 모바일 컴퓨팅

논 문 접 수 일 : 2012년 07월 05일
1차수정완료일 : 2012년 08월 21일
2차수정완료일 : 2012년 09월 10일
게 재 확 정 일 : 2012년 09월 22일