클라우드 환경에서의 비용 절감을 위한 휴리스틱 기반의 워크플로우 스케쥴링 기법

강동기, 김성환, 김대순, 윤찬현
한국과학기술원 전기및전자공학과
e-mail:dkkang@kaist.ac.kr, s.h_kim@kaist.ac.kr, sundae21@kaist.ac.kr,
chyoun@kaist.ac.kr

Heuristic based Workflow Scheduling for Cost Saving in Cloud Computing Environment

Dong-Ki Kang, Seong-Hwan Kim, Dae-Sun Kim, Chan-Hyun Youn
*Dept of Electrical Engineering, KAIST

요 약

클라우드 컴퓨팅 환경에서는 가상화된 자원 인스턴스를 할당받아 응용을 처리할 때 적절한 서비스 질을 만족하면서도 자원 운용 비용을 최소화하는 것이 중요한 이슈로 연구되고 있다. 그리드 환경과는 달리 클라우드 환경에서는 모든 가상 자원 인스턴스는 작업의 처리시간이 아닌 고정된 자원 할당 시간을 기준으로 가격이 책정되므로 기존의 그리드 환경을 고려한 자원 스케쥴링 기법은 클라우드 환경에서는 부적절한 자원 낭비를 발생시킬 수 있다. 이러한 문제를 해결하기 위해 본 논문에서는 클라우드 가격 정책을 고려한 휴리스틱 기반의 자원 스케쥴링 기법을 제안하고 이를 워크플로우 응용에 적용시키고자 한다. 제안된 기법의 성능 평가를 위하여 Montage 프로젝트를 워크플로우 응용으로 채택하였고 이를 Openstack 기반 클라우드 플랫폼을 기반으로 실험을 수행하였다.

1. 서론

클라우드 컴퓨팅 환경에서는 모든 컴퓨팅 자원을 가상화된 자원 인스턴스(Virtual Machine instance) 로 클라우드 서비스 사용자에게 제공하므로 기존의 물리 자원을 설치하고 관리 및 운영하는 비용을 효과적으로 줄일 수 있도록 한다. 모든 가상 자원 인스턴스는 pay-as-you-go manner 기반으로 클라우드 서비스 사용자에게 제공되어 자원을 사용한 시간만큼 비용을 부과하도록 한다. 가상 자원 인스턴스의 사용 시간이 길수록 그리고 해당 인스턴스의 컴퓨팅 성능이 높을수록 사용 비용이 증가하므로 처리하고자 하는 응용 서비스의 질을 요구 수준으로 만족하면서도 자원 사용 비용은 최소화하는 기법이 요구된다.

그런데 아마존, 구글, Gogrid 와 같은 클라우드 서비스 제공자들은 가상 자원 인스턴스 할당 시 fine-grained 기반이 아닌 coarse-grained 기반의 자원 가격 정책을 채택한다[1]. 즉 한 시간, 한달 혹은 1 년 단위로서 기간을 고정하여 자원을 할당하게 되므로 클라우드 서비스 사용자는 실질적인 작업 처리 시간에 관계없이 무조건 할당 시간을 기준으로 자원 사용 비용을 지불하여야 한다. 이 경우 작업을 처리하고 남는 시간 시간 만큼의 비용 낭비가 발생하게 된다. 만약 가상 자원 인스턴스를 요구하는 각 작업들의 처리시간이 짧으면서도 개수가 많다면 이러한 낭비 문제는 더욱 심각해진다. 기존의 그리드 환경에 기반한 자원 관리 기법들은 이러한 클라우드 환경에서의 자원 가격 정책을 고려하지 않았으므로 이를 고려한 새로운 자원 관리 기법이 요구된다[2,3].

본 논문에서는 이러한 문제를 해결하기 위하여 클라우드 자원 비용 정책을 고려한 휴리스틱 기반의 워크플로우 스케쥴링 기법을 제안한다. 우리가 제안하는 기법은 크게 2 가지로 구성되는데 첫 번째는 가상 자원 패킹 (VM packing) 이며 두 번째는 단일 자원 기반 다중 요구 처리 (MRSR : Multi Requests to Single Resource) 과정이다. 가상 자원 패킹 단계에서는 이미 할당할 자원 flavor type 이 결정된 상황에서 각 task 들이 순차적으로 (sequentially) 가상 자원에 할당하도록 하며, MRSR 단계에서는 두 개의 task 가 한 쌍을 이루어 단일 가상 자원 인스턴스에서 동시에 실행될 수 있도록 통합하는 과정을 수행한다. 제안된 기법의 성능 평가를 위하여 오픈소스인 Openstack 기반의 클라우드 플랫폼을 구축하고 실험을 위한 응용으로서 Montage Project 를 워크플로우 응용으로 채택하여 성능을 측정하였다[4,5]. 실험 결과를 통해 우리가 제안하는 가상 자원 패킹 및 MRSR 기법을 통하여 클라우드 가격 정책하에서 발생하는 불필요한 자원 낭비를 효과적으로 줄이면서도 사용자가 요구하는 서비스 질을 유지할 수 있음을 확인한다.

2. 클라우드 기반 워크플로우 관리 시스템 구조

서론에서 언급했듯 클라우드 환경에서는 자원 할당 가

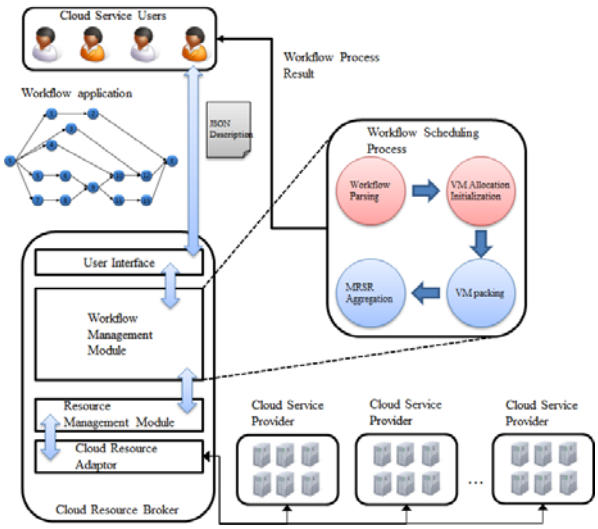


그림 1. 클라우드 기반 워크플로우 관리 시스템 구조

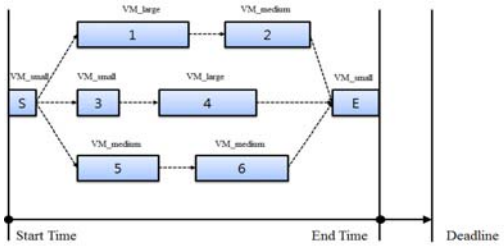
격이 작업 처리 시간이 아닌 고정된 자원 할당 시간에 의하여 결정된다. 일단 가상 자원 인스턴스가 사용자에게 할당되면 사용자의 작업이 끝나는 시간과 관계없이 미리 정해진 시간까지 활성화 상태를 유지하는 것이다. 이 경우 발생하는 자원 낭비 비용식은 다음과 같다.

$$Cost_{waste}(R,task_i) = Cost(R,p) \cdot \left(\left\lceil t^{p}_{R,task_i} \right\rceil - t^{p}_{R,task_i} \right) \quad (1)$$

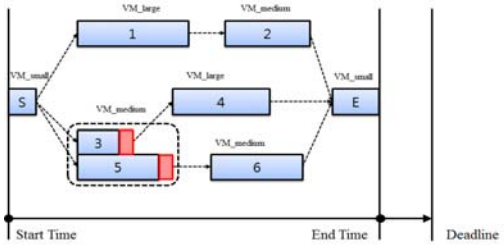
이때 $t^{p}_{R,task}$ 는 flavor type 이 R 인 가상 자원 인스턴스 위에서 워크플로우의 i 번째 task 가 실행될 때의 프로세싱 시간을 의미하며 $Cost_{waste}(R,task_i)$ 는 이 때 발생하는 자원 낭비 비용을 의미한다. $Cost(R,p)$ 는 flavor type R 인 가상 자원 인스턴스의 자원 할당 주기가 p 일 때 단위 시간당 지불해야하는 비용을 나타낸다. 결국 $Cost_{waste}(R,task_i)$ 을 최소화하는 것이 본 논문에서 제안하는 기법의 목표라 할 수 있다.

그림 1 에서는 본 논문에서 제안하는 기법을 적용한 클라우드 기반의 워크플로우 관리 시스템 구조를 보인다.

먼저 클라우드 서비스 사용자가 처리하고자 하는 워크플로우 응용 description 을 JSON (JavaScript Object Notation) 형태로 클라우드 자원 브로커에 전송하면 이를 User Interface 를 통해 받은 다음 워크플로우 관리 모듈 내의 4 가지 프로세스 – 워크플로우 파싱, 가상 자원 할당 초기화, 가상 자원 패킹 및 MRSR 병합 – 를 거쳐 자원 할당을 완료한다. 먼저 워크플로우 파싱 단계에서는 입력된 전체 워크플로우 description 을 각각 서브 태스크로 분할하며, 가상 자원 할당 초기화 과정에서는 기존의 워크플로우 스케쥴링 기법중의 하나인 GAIN/LOSS 알고리즘을 통하여 각 서브 태스크에 가상 자원 flavor type 을 할당한다. 가상 자원 패킹 과정에서는 동일한 flavor type 을 요구하는 서브 태스크들이 순차적으로 하나의 가상 자원 인스턴스에 할당된다. 즉 태스크들의 처리 시간이 서로



(a) MRSR 처리 전 워크플로우 그래프



(b) MRSR 처리 후 워크플로우 그래프

그림 2. 워크플로우 MRSR 병합 프로세스

중첩되지 않는 조건하에서 가상 자원 인스턴스에 패킹 되어 유휴 비율 (idle ratio) 을 최소화하는 것이다.

MRSR 병합 단계에서는 가상 자원 패킹을 통해 재 할당 과정을 거친 서브 태스크들을 병렬적으로 다시 통합하여 하나의 가상 자원 인스턴스 위에서 두 개의 서브 태스크에 동시에 실행될 수 있도록 한다. MRSR 기법은 멀티프로그래밍에 의하여, 태스크를 순차적으로 실행할 때의 처리시간보다 병렬적으로 실행할 때 처리시간이 더 짧은 것을 이용하여 자원 이용률을 최대화하면서도 사용자가 요구하는 서비스 질 – 본 논문에서는 워크플로우 데드라인으로 정의한다 – 을 만족할 수 있도록 한다.

사용자가 가상 자원을 통해 처리하고자 하는 워크플로우 g 는 다음과 같이 정의된다.

$$Workflow\, g = \{N(n_1, n_2, ..., n_k), E, d\}, k = of\, nodes \in g \quad (2)$$

여기서 n 은 워크플로우 g 의 서브 태스크이며, E 는 각 노드의 연결성을 나타내는 엣지 정보를, d 는 사용자가 요구하는 데드라인을 의미한다. 이 때, 워크플로우 g 의 임의의 노드 i 부터 j 까지 처리하는 데 필요한 총 프로세싱 시간은 다음과 같이 구할 수 있다.

$$tpt^{g}(i,j) = pt(n_i^g) + \max(tpt^{g}(N^{g}_{chi(i)}, j)) \quad (3)$$

여기서 $pt(n_i^g)$ 는 워크플로우 g 의 i 번째 태스크의 프로세싱 시간을 의미하며 $N^{g}_{chi(i)}$ 는 i 번째 task 의 child task 집합을 의미한다. 식 (3) 을 통해 노드 i 부터 j 까지의 critical path 길이를 구할 수 있고 만약 노드 i 가 시작 태스크, 노드 j 가 종료 태스크라면 이는 워크플로우의

<표1> 가상 자원 패킹 Pseudo code

```
input : Workflow g <- GAIN/LOSS scheduling
output : Scheduled Workflow g with VM packing

1: taskList N, N' <- extract sub-tasks from workflow g
2: while(true)
3:   i = earliest start task(N), f = requiredFlavor(i)
4:   for(each VM j ∈ allocated VMs)
5:     if(flavor(j)=f & state(j, start time(i))=idle)
6:       N' <- allocateVM(i, j)
7:       break
8:     end if
9:   end for
10: if(cannot find packable VM)
11:   N' <- allocateVM(i, new VM k)
12: end if
13: remove i from N
14: if(N=null) break
15:end while
```

<표2> MRSR 병합 Pseudo code

```
input : Workflow g <- VM packing
output : Scheduled Workflow g with MRSR aggregation

1: while(true)
2:   for(each task i in workflow g)
3:     for(each VM j ∈ allocated VMs)
4:       if(startTime(VM j) ≤ startTime(task i) &
5:           OverlapCount(VM j, task i)<3)
6:         MRSRpair p = (VM j, task i)
       calculate total processing time t of workflow g
       after MRSR aggregation with p
9:         if(t < deadline(g))
10:          candidateMRSRLists C <- insert p
11:        end if
12:      end if
13:    end for
14: end for
15: if(C = null)
16:   break
17: end if
18: workflow g <- apply min cost MRSRpair in p
19: empty C
20:end while
```

총 프로세싱 시간으로 간주할 수 있다.

그림 2 에서는 워크플로우를 시간 흐름에 따라 그래프의 형태로 보이고 있다. 2(a) 에서는 가상 자원 초기 할당 및 패킹에 의해서 각 태스크에 할당하고자 하는 가상 자원 인스턴스가 결정된 워크플로우 그래프를 보이며, 2(b) 에서는 MRSR 병합 기법을 통해 태스크 3 과 태스크 5 가 태스크 5 에 할당된 medium 타입의 가상 자원 인스턴스로 병렬 병합 된 그래프를 보이고 있다. 즉 태스크 3 과 태스크 5 가 하나의 인스턴스에서 동시에 실행될 때의 추가 지연 시간이 워크플로우의 critical path 길이에 영향을 주지 않거나 혹은 영향을 주더라도 워크플로우의 데드라인을 만족하면 이는 허용이 가능하다. 이를 고려하여 MRSR 을 수행하기 위한 조건식은 다음과 같다.

$$tdt^{g(n)} = tdt^{g(n-1)} + t^{delay}_{merge(i,j)} \leq d^g \tag{4}$$

$$mergePair^{g(n)}_{cost} = \\ \arg max_{(n_i,n_j)}(c_{(af_i,p)} \cdot \lceil t^p_{i,af_i} \rceil + c_{(af_j,p)} \cdot \lceil t^p_{j,af_j} \rceil - \\ c_{(af_i,p)} \cdot \lceil t^p_{(ij),af_i} \rceil), \forall n_i,n_j \in N^{g(n)}, \forall n_i,n_j \notin N^{g(n)}_{merge} \tag{5}$$

식 (4) 에서 $tdt^{g(n)}$ 은 워크플로우 g 에 대하여 MRSR 을 n 번 적용했을때 원래의 프로세싱 시간에 추가적으로 더해지는 총 지연 시간을 나타내며, $t^{delay}_{merge(i,j)}$ 는 서브 태스크 i, j 에 대해 MRSR 을 수행했을 때의 지연 시간을 나타낸다. 식 (5) 는 워크플로우 g 로부터 MRSR 을 수행했을 때 최대로 절감할 수 있는 자원 절감 비용 크기를 나타내며 그 때의 서브 태스크 i, j 가 MRSR 을 수행하는 태스크로 선택된다. 그리고 한번 MRSR 을 수행한 태스크 i, j 는 이후의 MRSR 수행 후보 태스크 목록에서 제외된다.

표 1 과 표 2 는 각각 가상 자원 패킹 및 MRSR 병합 단계에 대한 Pseudo code 를 보이고 있다. 먼저 표 1 의 가상 자원 패킹 기법을 살펴보면 2번 부터 9번 줄 까지 태스크를 가상 자원 인스턴스에 패킹하는 과정을 수행한다. 즉 태스크의 수행 시작 시간에 유휴 상태인 가상 자원 인스턴스에 할당하는 것이다. 만약 유휴 상태의 인스턴스를 찾지 못하면 새로운 인스턴스를 생성하여 할당하도록 한다. 표 2 의 MRSR 병합 기법을 살펴보면, 2번 부터 14 번 줄 까지는 MRSR 병합을 수행하기 위한 태스그 후보 쌍 집합을 수집하는 역할을 수행한다. 즉 MRSR 병합 조건식 (4) 을 만족하는 태스크 후보 쌍을 추출한다. 추출된 태스크 후보 쌍 집합에서 조건식 (5) 를 기반으로 18번 줄에서 자원 사용 절감 비용이 가장 큰 태스크 후보 쌍이 선택되어 MRSR 병합을 수행한다. 만약 MRSR 병합 태스크 후보 쌍이 없는 경우에는 알고리즘이 종료된다.

3. 실험 및 성능 평가

제안하는 가상 자원 패킹 및 MRSR 병합 기법의 성능 평가를 위하여 본 논문에서는 오픈스택 기반의 클라우드 플랫폼을 구축하였다. 구축된 시스템은 HP Xeon E5620 2.4G, Memory 16G, HDD 1T 성능의 노드 5 개로 구성되었으며 운영체제는 Ubuntu 12.04 를 기반으로 하였다. 제안 기법을 적용하기 위한 워크플로우 응용으로는 천문 관측 이미지 처리를 위한 Montage 프로젝트의 워크플로우 예제를 채택하였다[5]. 특별히 Montage 응용 중에서도 컴퓨팅 능력에 따라 수행 시간의 변동성이 큰 mProjExec,

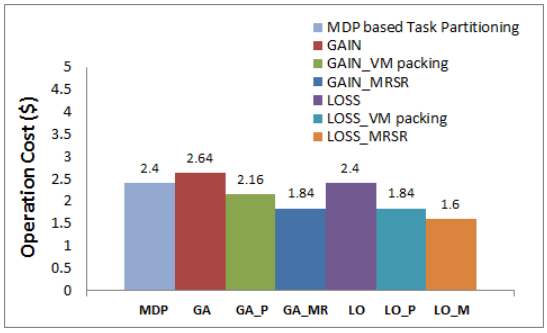


그림 4. 제안 기법의 자원 사용 비용 결과

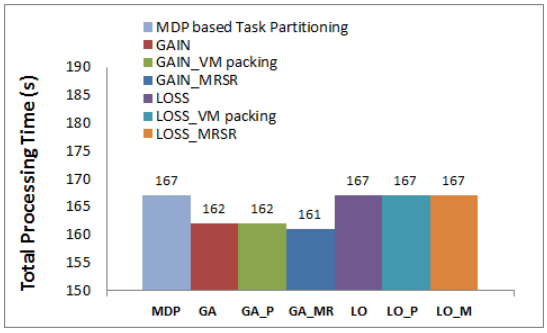


그림 5. 제안 기법의 프로세싱 시간 결과

mAdd, mJPEG 를 서브 태스크로 구성하여 실험을 수행하였다. 워크플로우 예제의 프로세싱 데드라인은 170 초로 설정되었으며 각 가상 자원 인스턴스의 가격은 GoGrid 의 클라우드 가격 정책을 채택하였다[1].

그림 4 는 기존의 MDP 기반의 워크플로우 스케쥴링 및 휴리스틱 기반 GAIN/LOSS 스케쥴링 기법과 비교하여 제안된 가상 자원 패킹 및 MRSR 병합 기법의 자원 사용 비용값을 그래프로 보이고 있다. 기존 그리드를 고려한 워크플로우 스케쥴링 기법들은 각 태스크에 인스턴스를 별개로 할당하므로 인스턴스의 잉여 시간에 의한 자원 낭비가 크지만 제안 기법은 각 서브 태스크들을 하나의 인스턴스에 순차적 및 병렬적으로 병합하기 때문에 자원 낭비를 최소화 할 수 있다. 이에 의하여 GAIN 으로 자원 초기화를 하는 가상 자원 패킹 및 MRSR 병합 기법은 기존의 MDP 기반 및 원 GAIN 기법에 비하여 각각 10%, 20% 그리고 24%, 31% 정도의 비용 절감 성능의 증가를 보이며 LOSS 로 초기화를 하는 경우에는 기존 MDP 기반 및 원 LOSS 기법에 비하여 각각 24%, 31% 그리고 34%, 34% 의 비용 절감 성능 증가를 보인다.

그림 5 는 기존의 기법과 제안 기법을 비교하여 워크플로우의 총 프로세싱 시간 결과를 그래프로 보이고 있다. 제안 기법의 경우 태스크를 가상 자원 인스턴스에 병합하기 때문에 일반적으로는 기존 워크플로우 스케쥴링 기법보다 프로세싱 시간이 길어질 수 있다. 그러나 본 그래프 결과에서 GAIN 기반의 MRSR 병합 기법은 기존의 MDP 및 원 GAIN 기법에 비하여 오히려 프로세싱 시간이 감소됨을 볼 수 있는데 이는 MRSR 병합 기법에 의해 태스크에 원래 할당된 가상 자원 인스턴스 flavor 타입 보다 더 성능이 높은 flavor 타입으로 병합이 이루어진 것에 의해 도출된 것으로 분석될 수 있다.

LOSS 기반의 가상 자원 패킹과 MRSR 병합 기법의 경우 원 LOSS 기법과 동일한 프로세싱 시간 성능을 보이고 있다. 즉 본 논문의 제안 기법은 워크플로우 데드라인을 초과하지 않는 범위내에서만 태스크 병합을 수행하므로 사용자의 요구 서비스 질은 만족하면서도 자원 사용 비용은 효과적으로 줄일 수 있다.

4. 결론

본 논문에서는 클라우드 환경에서의 가상 자원 사용 비용을 절감하면서도 사용자의 요구하는 서비스 질은 만족시킬 수 있는 새로운 워크플로우 스케쥴링인 가상 자원 패킹 및 MRSR 병합 기법을 제안하였다. 기존의 그리드 워크플로우 스케쥴링은 고정된 자원 할당 시간에 기반을 둔 클라우드 서비스의 가격 정책 특성을 고려하지 못하여 불필요한 비용 낭비가 발생하였으나 본 논문의 기법은 이를 고려하여 가상 자원 비용을 효율적으로 절감시킬 수 있다. Montage 프로젝트를 예제로 채택하여 오픈스택 기반의 클라우드 플랫폼 위에서 실험을 수행한 결과 기존 기법에 비해 평균적으로 26% 의 자원 사용 비용 절감을 보였다. 또한 비용 절감에도 불구하고 워크플로우의 요구 데드라인을 위반하지 않음을 확인할 수 있었다. 추후 연구에서는 가상 자원 인스턴스의 컴퓨팅 능력 뿐 아니라, 인스턴스간의 통신 비용까지 고려한 태스크 병합 알고리즘을 제안할 것이다.

Acknowledgment

이 논문은 2013 년도 정부(교육과학기술부)의 재원으로 한국연구재단-클라우드 Collaboration 기술 사업(No. 2012-0020522) 및 지식경제부의 재원으로 '유전체 분석용 슈퍼컴퓨팅 시스템 개발' (No. 10038768) 사업의 지원을 받아 수행된 연구임

참고문헌

[1] GoGrid (2013), http://www.gogrid.com/

[2] J. Yu, R. Buyya, and C. K. Tham, "Qos-based Scheduling of Workflow Applications on Service Grids," Proc. Int'l Conf. e-Science and Grid Computing, pp. 140-147, July 2005.

[3] R. Sakellariou, and H. Zhao, "Scheduling workflows with budget constraints," Integrated Research in GRID Computing, CoreGRID Series, S. Gorlatch, and M. Danelutto, eds., pp. 189-202, Springer, 2007.

[4] Openstack (2013) http://www.openstack.org/

[5] http://montage.ipac.caltech.edu/

[6] D. K. Kang, S. H. Kim, Y. Ren, B. S. Kim, W. J. Kim, Y. S. Kim, C. H. Youn, and C. S. Jeong, "Enhancing a Strategy of Virtualized Resource Assignment in Adaptive Resource Cloud Framework," Proc. Int'l Conf. ACM ICUIMC, 2013