클라우드 환경에서 품질 제약형 워크플로우 스케쥴링 기법에 대한 조사

하윤기*, 윤찬현*
*한국과학기술원 전기 및 전자공학과
e-mail : {milmgas, chyoun}@kaist.ac.kr

A Survey on QoS Constrained Workflow Scheduling Scheme in Cloud

Yun-Gi Ha*, Chan-Hyun Yoon*
* Dept. of Electrical Engineering, Korea Advanced Institute of Science and Technology

요 약

클라우드 컴퓨팅은 워크플로우 응용의 처리에 있어 효율적으로 시스템을 구축할 수 있도록 한다. 그렇지만 여러 클라우드 사업자 측에서 다양한 서비스를 제공하고, 이로 인해 발생하는 복잡성으로 사용자 스스로 클라우드 인프라를 활용하여 워크플로우 처리를 수행하는 것은 어려운 일이며, 이 문제를 극복하기 위해 클라우드 브로커 시스템이 도입되었다. 이와 같은 브로커 시스템에서 워크플로우를 처리할 때 사용자가 지정한 품질 제약을 만족시키는 이슈를 해결하기 위해 많은 워크플로우 스케쥴링에 대한 연구들이 수행되었다. 본 논문에서는 클라우드 환경에서 보다 효율적인 워크플로우 처리를 위해 기존 워크플로우 스케쥴링 기법의 연구들에 대한 특징과 한계에 대한 조사를 수행하였다.

1. 서론

클라우드 컴퓨팅은 그 사용자들로 하여금 대량의 데이터 처리를 높은 성능을 보이는 컴퓨팅 자원들을 저렴한 비용에 필요한 만큼만 사용할 수 있도록 하여 워크플로우 처리에 있어 효과적이다. 이처럼 클라우드 컴퓨팅이 분산 환경에서 동작하는 워크플로우 응용에 대해 실현가능한 해결책을 제시하기 때문에, 사용자들은 스스로 위한 인프라를 직접 구축하지 않고 클라우드 인프라 사업자가 제공하는 컴퓨팅 자원을 활용하는 클라우드 브로커의 덕택으로 간편하게 워크플로우 응용의 처리를 수행할 수 있게 되었다.

하지만, 여전히 주어진 비용 내에서 보다 효율적인 워크플로우 처리를 위해 발생하는 자원 관리와 같은 문제가 존재한다. 사용자는 deadline, 비용, 그리고 신뢰성과 같은 품질 제약과 함께 워크플로우 응용에 대한 처리 요청을 보내게 되는데 이 때 일반적으로 클라우드 브로커에서는 더 높은 품질 제약에 대한 목표를 달성하기 위해 더 많은 비용을 요구하게 된다 [1]. 따라서, 클라우드 브로커는 워크플로우를 구성하는 각 작업들을 적합한 자원에 할당하여 품질 제약을 만족하도록 해야 한다. 그렇지만, 워크플로우 스케쥴링 문제는 NP-hard 문제로 이미 증명되어 이 문제를 해결하기 위한 정확한 알고리즘은 존재하지 않는 상태이다 [2].

또한, 클라우드 환경에서는 자원이 Virtual Machine(VM) 형태로 피상적인 사양 (예: CPU 코어 숫자, 스토리지 크기, 램 크기) 으로 주어지기 때문에 같은 종류의 VM 내에서도 성능의 차이를 보인다 [3]. 따라서 이러한 성능 차이를 고려한 워크플로우 스케쥴링 알고리즘의 구현 및 적용이 필요하다.

본 논문에서는 클라우드 환경에서 워크플로우 스케쥴링 문제를 다룬 기법들을 살펴보고, 해당 기법이 지닌 약점들을 극복하여 적응적으로 워크플로우 스케쥴링을 수행하는 기법을 개발하기 위한 조사를 수행하였다.

2. 본론

2.1 Satisfying the Budget Constraint within the Best Affordable Assignment [4]

해당 기법은 여러 가지 품질 제약 사항들 중 워크플로우를 처리할 때 한계치로 정한 비용, 즉 예산을 주된 품질 제약으로 두고 워크플로우 스케쥴링을 수행하는 휴리스틱이다.

해당 기법에서는 작업-VM 초기 할당을 바탕으로 두 가지의 워크플로우 스케쥴링에 대한 접근법을 소개한다.

첫 번째 방법은 LOSS 라고 불리는 방법으로, 이 경우 먼저 각기 다른 VM 위에서의 작업별 수행시간 테이블을 바탕으로 HEFT []나 HBMCT [] 를 통해 작업-VM 초기 할당을 수행하여 워크플로우의 completion time 을 최소화하는 스케쥴을 생성한다. 해당 스케쥴의 비용을 사용자의 예산과 비교하여 초기 스케쥴의

비용이 더 작을 경우 이 스케쥴을 최종 스케쥴으로 정한다. 만약 그렇지 않을 경우 task 의 작업에 대한 재할당과정을 수행한다. 이 과정의 목적은 completion time 을 최소한으로 증가시키면서 비용을 최대한 절감하는 작업을 찾아 VM 을 다시 매핑시키는 것이다. 재할당과정을 수행하기 위해 각 작업에 대해 *LossWeight(i,m)* 를 다음 식 (1) 과 같이 계산한다.

$$LossWeight(i,m) = \frac{T_{new} - T_{old}}{C_{old} - C_{new}} \quad (1)$$

위 식 (1)에서 i 는 초기 스케쥴에 사용된 VM, m 은 재할당과정에서 사용된 VM 을 가리키며 T 는 해당 할당 상태에서의 예상 작업 수행시간, C 는 해당 할당 상태에서의 예상 작업 소모비용을 가리키며 해당 값이 큰 작업부터 재할당과정을 거쳐서 사용자가 정한 예산과 비교하는 작업을 수행한다.

두 번째 방법은 GAIN 이라는 이름을 지닌 방법이며, 각 작업은 초기에 최저비용으로 처리가능한 상태로 VM 이 할당된다. 이 방법은 LOSS 와 비슷한 방법으로 GainWeight 를 식 (2)와 같이 정의하여 초기 스케쥴에 대한 재할당과정을 수행한다.

$$GainWeight(i,m) = \frac{T_{old} - T_{new}}{C_{new} - C_{old}} \quad (2)$$

해당 기법을 요약하면 LOSS 에 의해 completion time 이 최소화된 초기 스케쥴 혹은 GAIN 에 의해 비용이 최소화된 초기 스케쥴에 대하여 재할당을 수행하여 최적의 스케쥴링을 찾는 것이다. 이 기법의 단점은 각 재할당 과정을 위하여 해당 기법은 추가적으로 *LossWeight* 나 *GainWeight* 의 계산과 그에 따른 각 재할당 스케쥴링과 사용자가 지정한 비용 및 completion time 과의 비교를 수행해야 하므로 해당 휴리스틱은 높은 복잡도를 요구한다는 점이다. 뿐만 아니라 휴리스틱을 이용하여 획득한 스케쥴에 따라 워크플로우를 처리할 경우 서론에서 소개한 균일하지 못한 VM 성능과 같은 같은 이유로 최종적으로 결정된 스케쥴에서 지정된 각 작업의 completion time 에 대한 위반이 발생하여 전체 워크플로우의 데드라인을 보장하지 못하는 경우가 발생할 수 있다는 점을 보완해야 할 점으로 꼽을 수 있다.

2.2 Phased Workflow Scheduling Scheme Based on Petri-Nets [ ]

해당 기법은 위에서 소개한 정적인 워크플로우 스케쥴링 기법이나 복잡도가 높은 동적인 워크플로우 스케쥴링 방식이 아니라 두 가지 스케쥴링 방식의 특징을 적절히 혼합한 페이즈드 워크플로우 스케쥴링 방식이다.

해당 기법이 워크플로우 스케쥴링을 위해 작동하는 순서는 다음과 같다. 먼저 스케쥴링 페이즈가 실행된다. 해당 페이즈에서는 다른 VM 에서의 historical execution time data 를 이용하여 각 작업의 load 를 조사한다. 이후 각 작업의 load 에 대한 조사를 바탕으로 워크플로우 critical path[]를 결정하여 unexecuted critical path load 에 대한 각 작업의 비율을 알아내어 load distribution rate 를 결정한다. 이후 익스큐션 페이즈에서 사용자가 워크플로우 처리 요청과 함께 제출한 completion time 에 각 작업의 load distribution rate 를 곱한 값에 가장 가까운 실행 시간을 갖는 VM 을 해당 작업에 스케쥴링하도록 한다.

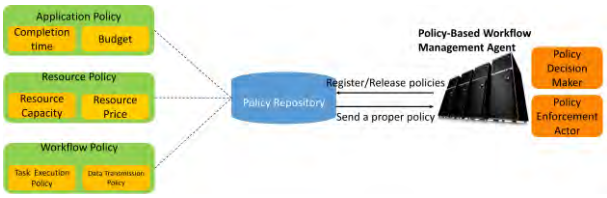
해당 기법은 1:1 작업-VM 매핑이 아닌 load distribution rate 에 따른 VM 할당으로 클라우드 환경에서 발생하는 VM 성능의 변동에도 어느 정도 대처할 수 있는 적응력을 지니고 있다.

그렇지만 워크플로우 스케쥴링에 활용되는 N 개의 VM resource type 중 가장 처리속도가 빠른 VM resource type 에서 모든 task 들을 스케쥴하더라도 realtime 으로 워크플로우를 처리하게 되면 요청된 completion time 을 만족하지 못하는 경우가 발생하는 약점이 있다.

2.3 정책기반 워크플로우 스케쥴링 시스템 [ ]

정책기반 워크플로우 스케쥴링이란 워크플로우 토폴로지 및 개별 작업의 특성, 그리고 품질 제약의 수준에 따라 각기 다른 정책을 반영하여 앞에서 소개한 2.1 과 2.2 와 같은 워크플로우 스케쥴링 기법이 갖는 근본적인 한계를 최소한의 비용을 희생하여 해소하기 위해 고안된 방법이다. 여기서 정책이란 브로커 사용자가 지정한 워크플로우 처리에 있어서 품질 제약을 만족시키기 위한 실행 전략 [] 으로, 정책에 근거하여 자원을 관리하는 개념은 네트워크 매니지먼트에서 오랫동안 사용되어 왔다.

정책 기반 워크플로우 스케쥴링 시스템에는 그림 1 과 같이 Policy Repository Management, Policy Repository, Policy Decision Maker 그리고 Policy Enforcement Actor 의 4 가지 요소가 존재하며, 이들은 각기 다른 응용, 워크플로우, 그리고 자원에 대해 정의된 정책을 관리하여 이들 정책을 각 워크플로우 처리 요청에 적용한다.



(그림 1) 정책기반 워크플로우 관리 시스템 개념도

3. 결론

본 논문에서 소개한 기존의 워크플로우 스케쥴링 기법들은 본론에서 살펴본 것과 같이 클라우드 환경을 효율적으로 활용하는 데 한계가 존재한다. 이와 같은 문제를 극복하기 위해, 우리는 워크플로우 토폴로지 및 개별 작업의 특성에 따라 다른 워크플로우 처리 정책을 적용하여 적은 비용을 사용하여 효과적으로 워크플로우 스케쥴링을 수행하는 방법을 고안하

고자 한다.

Acknowledgement

본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음 (NIPA-2014(H0301-14-1020))

참고문헌

[1] Yu, Jia, Rajkumar Buyya, and Chen Khong Tham. "Cost-based scheduling of scientific workflow applications on utility grids." e-Science and Grid Computing, 2005. First International Conference on. IEEE, 2005

[2] Pinedo, Michael L. Scheduling: theory, algorithms, and systems. Springer, 2012.

[3] GoGrid. Available: http://www.gogrid.com/

[4] Sakellariou, Rizos, et al. "Scheduling workflows with budget constraints." Integrated Research in GRID Computing. Springer US, 2007. 189-202.

[5]