한국시뮬레이션학회 논문지
Vol. 23, No. 4, pp. 219-228 (2014. 12)
http://dx.doi.org/10.9709/JKSS.2014.23.4.219
ISSN 1225-5904

분산 환경에서 계산 자원의 효율 증대를 위한 데이터 특성 기반의 작업 분류방법

문성환 · 김재권 · 김태영 · 최정석 · 조규철 · 이종식*

Job Classifying method based on Data Traits for Increased Efficiency of Computational Resources in Distributed Environment

Sung-hwan Moon · Jae-kwon Kim · Tae-young Kim · Jeong-seok Choi · Kyu-cheol Cho · Jong-sik Lee*

ABSTRACT

Various computational resources in distributed environment are to build a high-performance computing environments through virtualization technology. Recently, there is a growing need for a complicated process due to the improvement of the user-level application, which has led to demand for high-performance computing. The requested job from users is composed of data. And because of each data has own characteristics, the classifier may consider the features of data. In this paper, we propose Job Classifying method based on Data Traits for Increased Efficiency of Computational Resources in Distributed Environment (JCDT). JCDT classifies the job by data traits of the users' request, is expected to improve the job processing time and increase the processing speed of the calculation resources.

Key words : Distributed Environment, High-Performance Computing, Job Classifying, Data Traits, JCDT

요 약

분산 환경에 존재하는 다양한 이기종의 계산 자원은 가상화 기술을 통해 통합된 고성능 컴퓨팅 환경을 구축한다. 최근, 사용자 수준의 향상으로 인해 복잡한 응용 작업의 처리에 대한 요구가 증가하고 있으며, 이는 고성능 컴퓨팅에 대한 수요로 이어지고 있다. 사용자가 요구하는 각각의 작업에는 데이터가 포함되어 있고, 각각의 데이터는 고유의 특성을 가지고 있으므로, 작업의 분류와 처리는 데이터의 특성이 고려되어야 한다. 본 논문에서는 분산 환경에서 계산 자원의 효율 증대를 위한 데이터 특성 기반의 작업 분류방법(JCDT : Job Classifying method based on Data Traits for Increased Efficiency of Computational Resources in Distributed Environment) 을 제안한다. 제안하는 JCDT 는 사용자가 요구하는 작업이 지닌 데이터의 특성을 기반으로 작업을 분류하여, 계산 자원의 효율 증대와 작업 처리속도를 향상시킬 수 있을 것으로 기대한다.

주요어 : 분산 환경, 고성능 컴퓨팅, 작업 분류, 데이터 특성, JCDT

1. 서 론

전세계적으로, 사용자 수준의 향상과 네트워크 기술의 발달로 인하여 대용량 데이터 작업 및 복잡도가 높은 응용계산 작업의 처리에 대한 필요성이 증가하고 있다. 서비스 제공자들은 다양한 계산 자원으로 이루어진 분산 환경을 구축하여 이러한 고성능 컴퓨팅에 대한 수요를 충족하고 있다(G, Kim 외 2010).

분산 환경을 통한 고성능 컴퓨팅 서비스에서 사용자와

*이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 (2012R1A1A2002751) 및 방위사업청과 국방과학연구소의 지원으로 수행되었습니다 (UD140022PD).

Received: 22 November 2014, **Revised:** 22 December 2014, **Accepted:** 24 December 2014

***Corresponding Author:** Jong-sik Lee
E-mail: jslee@inha.ac.kr
Department of Computer and Information Engineering, Inha University

제공자 간에는 서비스 수준 협약(SLA : Service Level Agreement)을 정한다. SLA 체결 수준에 따라 사용자는 일정한 비용을 지불하고, 제공자는 비용에 상응하는 수준의 서비스를 보장해야 한다(H, Kang 외 2013). 사용자는 작업 처리비용이 저렴할수록 유리하며, 제공자는 투입되는 자원 대비 서비스 보장성이 높을수록 유리하다.

분산 환경은 여러 계산 자원이 가상화 기술로 통합된 고성능 컴퓨팅 환경으로 구성되어 있으며, 다수의 사용자가 복잡도가 높은 응용 작업에 대해 지속적인 처리를 요구할 경우, 작업을 효율적으로 분산된 계산 자원들에 할당해야 서비스 처리율을 보장할 수 있다.

사용자가 요구하는 작업의 대부분은 데이터 처리이며, 이것은 데이터의 단순 저장 및 정리 작업부터, 영상 처리, 이미지 처리, 음성 처리 등 정형 및 비정형 데이터의 연산 작업까지 다양하다. 각 작업은 다양한 데이터를 포함하고 있으므로 계산 자원에 대한 의존도가 다르다. 데이터의 특성을 고려하지 않고 자원을 할당할 경우 분산 환경에서 작업이 효율적으로 처리될 수 없으며, 작업 처리속도가 느려질 수 있다(B.S., Kim 외 1993).

본 논문에서는 분산 환경에서 계산 자원의 효율 증대를 위한 데이터 특성 기반의 작업 분류방법(JCDT : Job Classifying method based on Data Traits for Increased Efficiency of Computational Resources in Distributed Environment)을 제안한다. JCDT 는 분산 서비스에서 사용자가 요구하는 데이터 처리 작업에 포함된 데이터의 특성을 기반으로 작업을 분류하고 의존도가 높은 물리 계산 자원에 작업을 할당하는 방법으로, 계산 자원의 효율 증대를 통해 작업 처리속도를 향상시키고 서비스 처리율을 보장할 수 있을 것으로 기대한다.

2. 관련 연구

2.1 클라우드 프로비저닝 서비스를 위한 퍼지 로직 기반의 자원 평가방법

퍼지 로직은 애매모호한 데이터에 대해 추론하고, 불명확한 데이터에 대해 의사결정을 하기 위한 기법이다. 클라우드 환경에서 작업 처리에 대한 스케줄링이 진행될 때, 퍼지 로직을 통해 평가된 물리 자원의 가용성 정보는 작업 할당의 기준이 되어, 효율적인 프로비저닝 스케줄링을 가능하게 한다. 퍼지 로직 기반의 자원 평가 및 작업 할당은 높은 작업 처리율과 자원 활용률을 기대할 수 있다(J.K., Kim 외 2013).

해당 관련 연구는 정확한 수치로 나타낼 수 없는 자원의 가용성 평가를 퍼지 로직을 통한 의사결정으로 해결하였으나, 본 연구는 분산 환경에 존재할 수 있는 이기종의 계산 자원 (물리 자원)의 성능을 구간별로 점수화하고 그것을 토대로 사용자의 요청 작업을 할당한 것으로, 해당 관련 연구와는 자원 평가에 대한 접근법에서 차이가 있다.

2.2 멀티미디어 저장시스템을 위한 비정형 데이터 관리기의 설계 및 구현

멀티미디어 데이터베이스 관리시스템의 비정형 데이터 관리기는 데이터의 복합객체적 성격, 용량에 따른 특성, 그리고 데이터의 접근 방식을 고려하여 설계되었다. 해당 시스템의 비정형 관리기는 영상, 음성, 이미지 및 그래픽과 같은 비정형 데이터를 각각의 속성에 따라 분류하며, 각 속성마다 개별적 관리기를 정의하여 데이터를 관리한다. 각 관리기인 미디어 데이터 모듈은 비정형 데이터의 효율적인 저장과 접근을 위해 객체 내 인덱싱 방법을 사용하였다(B.S., Kim 외 1993).

해당 관련 연구는 멀티미디어 콘텐츠에 포함된 영상, 음성, 이미지 및 그래픽과 같은 비정형 데이터를 각 속성에 따라 분류함으로써 데이터의 저장과 접근을 체계적으로 가능하게 한 것으로, 본 연구에서는 해당 관련 연구의 데이터 속성별 구분 및 관리 기법을 응용하여 계산 자원에 대한 스케줄링 기법에 적용하였다.

3. 데이터 특성 기반의 작업 분류방법

3.1 아키텍처 구성

데이터 특성 기반의 작업 분류방법은 사용자가 요구하는 작업에 포함된 데이터의 특성에 따라 작업을 분류하고, 연관성이 높은 물리 자원을 할당하여 작업을 처리한다. 제안하는 JCDT 는 Fig. 1 과 같이 아키텍처를 정의한다.

3.2 알고리즘

데이터 특성 기반의 작업 분류를 위해서는 JCDT 의 아키텍처에 따라 사용자 요청 작업을 순차적으로 처리해야 한다. 본 논문에서는 IaaS(Infrastructure as a Service) 환경(J.K., Kim 외 2013)에서의 작업 분류 및 자원 할당 방법을 제안한다. JCDT는 3개의 모듈로 작업을 처리하며, 다음은 각 모듈의 알고리즘에 대한 설명이다.

3.2.1 작업 분류 모듈

분산 서비스를 통해 사용자가 요청하는 작업은 사용자 개개인의 컴퓨팅 자원으로 처리할 수 없는 복잡한 응용

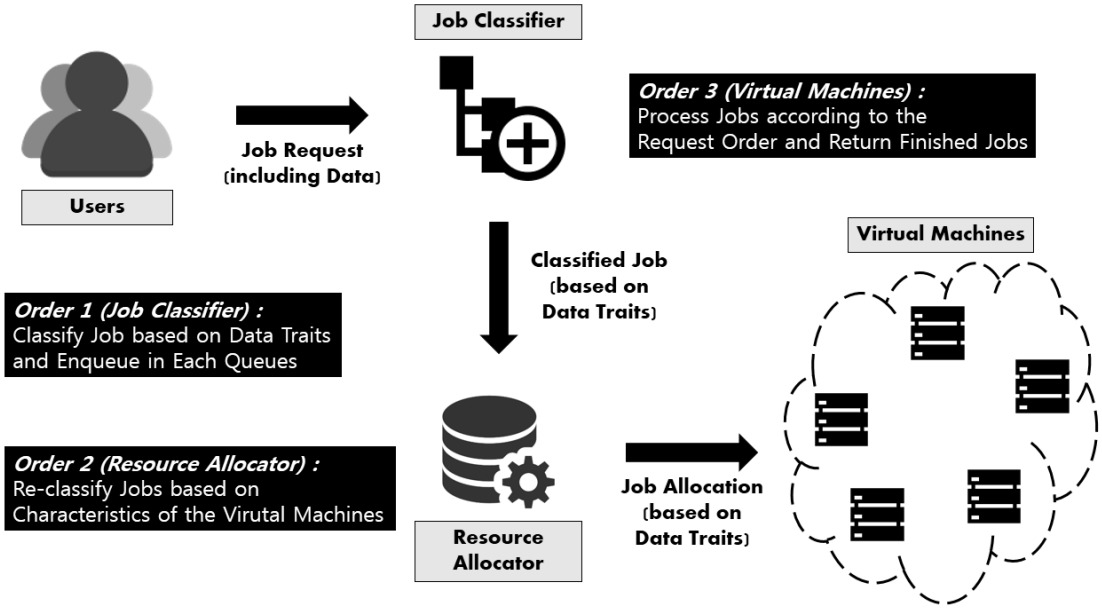

Fig. 1. JCDT Architecture

```
Algorithm 1 – Job Classifying (Class : Job Classifier)

WORKFLOW
1)      IF (Job from User) THEN
2)              Classify Job considering Data Traits
3)              SWITCH (Job related x)
4)                      CASE 'Video-Processing' :
5)                              Enqueue Job into Q_VideoProc
6)                      CASE 'Image-Processing' :
7)                              Enqueue Job into Q_ImageProc
8)                      CASE 'Audio-Processing' :
9)                              Enqueue Job into Q_AudioProc
10)                     DEFAULT                 :
11)                             Enqueue Job into Q_etcProc
12)             END
13)     END IF
14)     IF (Request from Resource-Allocator) THEN
15)             SWITCH (Request is x)
16)                     CASE 'Job related Video' :
17)                             Dequeue first Job from Q_VideoProc
18)                     CASE 'Job related Image' :
19)                             Dequeue first Job from Q_ImageProc
20)                     CASE 'Job related Audio' :
21)                             Dequeue first Job from Q_AudioProc
22)             END
23)             IF (Queue is not empty)
24)                     Send Job to Resource-Allocator
25)             END IF
26)     END IF
27)     IF (Job from Virtual-Machine) THEN
28)             Return Finished Job to User #
29)     END IF
```

Fig. 2. Algorithm of Job Classifier

작업부터, 단지 웹 기반 서비스를 사용할 목적으로 요청하는 작업까지 다양하다. 작업 분류 모듈은 여러 목적을 지닌 작업을 데이터 특성에 따라 분류하여 저장한다.

작업 분류 모듈은 JCDT 아키텍처의 Job Classifier 에 해당하는 알고리즘으로 Fig. 2와 같으며, 다음과 같이 동작한다.

1) 작업 분류 모듈은 사용자로부터 요청받은 최초의 작업이 포함하고 있는 데이터의 특성에 따라 작업을 분류하며, 각 특성에 맞게 미리 정의된 Queue 에 저장한다. Queue 는 데이터의 특성에 따라 4가지로 분류되며, 분류 기준은 Table 1과 같다.

작업은 데이터 특성에 따라 분류되어 Queue 에 저장됨

Table 1. Queue in Job Classifier Module

Queue Name	Description	Data in Job
Q_VideoProc	Enqueue the User-Request Job inclduing Video-Data	.avi, .mkv, .mp4, .wmv, etc.
Q_ImageProc	Enqueue the User-Request Job including Image-Data	.jpg, .png, .gif, .tif, etc.
Q_AudioProc	Enqueue the User-Request Job inclduing Audio-Data	.wav, .ogg, .mp3, .wma, etc.
Q_etcProc	Enqueue the User-Request Job including Other Data	.txt, .log, etc.

```
Algorithm 2 - Resource Allocating (Class : Resource Allocator)
WORKFLOW
1)      Load Information of All Virtual-Machines
        (Information : H/W Performance - CPU, RAM, NetResponse)
2)      IF (Job from Job-Classifier) THEN
3)              SWITCH (Job has Data Traits about x)
4)                      CASE 'Video-Processing' :
5)                              Enqueue Job into Q_CPU
6)                      CASE 'Image-Processing' :
7)                              Enqueue Job into Q_RAM
8)                      CASE 'Audio-Processing' :
9)                              Enqueue Job into Q_NetResp
10)                     DEFAULT                 :
11)                             Enqueue Job into Q_All
12)             END
13)     END IF
14)     IF (Request from Virtual-Machine) THEN
15)             SWITCH (Virtual-Machine has better x than others)
16)                     CASE 'CPU' :
17)                             Dequeue first Job from Queue_CPU
18)                     CASE 'RAM' :
19)                             Dequeue first Job from Queue_RAM
20)                     CASE 'NetResponse' :
21)                             Dequeue first Job from Queue_NetResponse
22)             END
23)             IF (Queue is not empty)
24)                     Send Job to Virtual-Machine
25)             END IF
26)     END IF
```

Fig. 3. Algorithm of Resource Allocator

과 동시에, 작업의 내부 속성정보에 어떤 Queue 로 분류되었는지에 대한 기록이 추가된다. 즉, 작업이 다른 모듈로 전송되더라도 해당 모듈에서 작업이 지닌 데이터 특성을 바로 파악하게하기 위한 의도이다.

2) 작업 분류 모듈은 자원 할당 모듈의 요청에 따라 각각의 Queue 에 저장된 작업들을 전송한다. 자원 할당 모듈은 작업 분류 모듈의 다음 과정으로, 작업이 실제 처리를 위해 작업 처리 모듈에 할당되기 전에 대기하는 모듈이다. 자원 할당 모듈에 존재하는 Queue 에 저장된 작업이 일정량 이하로 내려가면 작업 분류 모듈에게 신규 작업을 요청하게 된다.

3) 작업 분류 모듈은 작업 처리 모듈로부터 처리가 완료된 작업을 전송받고, 사용자가 최초에 요청한 작업이 맞는지 확인한 후에 완료된 작업을 반환한다.

3.2.2 자원 할당 모듈

작업 분류 모듈에는 데이터의 특성에 따라 분류된 작업이 존재한다. 자원 할당 모듈은 계산 자원의 처리 능력과 수용 가능 작업량을 고려하여 일정량의 신규 작업을 수시로 작업 분류 모듈에게 요청하여 전송받는다.

자원 할당 모듈은 JCDT 아키텍처의 Resource Allocator 에 해당하는 알고리즘으로, Fig. 3과 같으며, 다음과 같이 동작한다.

Table 2. Queue in Job Classifier Module

Queue Name	Description	Job Property
Q_CPU	Enqueue the Job to be assigned Node which has High-Level CPU	Job including Video-Data
Q_RAM	Enqueue the Job to be assigned Node which has High-Level RAM	Job including Image-Data
Q_NetResp	Enqueue the Job to be assigned Node which has High-NetResponse	Job including Audio-Data
Q_All	Enqueue the Job to be assigned Node on Low-Load	Job including Other Data

1) 자원 할당 모듈은 작업 분류 모듈로부터 작업을 할당 받는다. 자원 할당 모듈은 신규 작업을 제일 효율적으로 처리할 수 있는 작업 처리 모듈에 할당하기 위해, 작업 처리 모듈의 성능 정보에 따라 정의된 각각의 Queue 에 작업 분류 모듈로부터 전송된 신규 작업을 재분류하여 저장한다. Queue 는 작업 처리 모듈의 성능 정보에 따라 4가지로 분류되며, 분류 기준은 Table 2와 같다.

작업 처리 모듈의 성능 정보에 따라 작업을 재분류하는 이유는, 사용자의 작업에 포함된 데이터의 특성에 따라 작업 처리 모듈 즉, 물리 계산 자원에 대한 의존도가 다르기 때문이며, 각 데이터 특성을 가진 작업에 대한 계산 자원의 사용률은 다음과 같다.

- 영상 및 동적 이미지에 대한 작업 :

실시간 인코딩 및 디코딩, 대용량 영상 저장 등이 주요 작업이며, 작업 처리 시에 CPU 에 대한 사용률이 제일 높다(S.J., Lee 외 2010).

- 그래픽 및 정적 이미지에 대한 작업 :
  미리보기, 저장 등이 주요 작업이며, 작업 처리 시에 RAM 에 대한 사용률이 제일 높다(H.S., Oh 외 2013).
- 오디오 및 음성에 대한 작업 :
  스트리밍 서비스, 실시간 전송 등이 주요 작업이며, 작업 처리 시에 네트워크 지연을 최소화해야 한다(B.J., Kim 외 2011).

각 작업을 물리 계산 자원에 할당하기 위한 우선순위는 Eq. (1)과 같다.

```
Algorithm 3 - Job Processing (Class : Virtual-Machine)
WORKFLOW
1)      IF (Job from Resource-Allocator) THEN
2)              Start processing Job
3)      END IF
4)      IF (processing Job finished) THEN
5)              SEND finished Job to Job-Classifier
6)      END IF
```

Fig. 4. Algorithm of Virtual Machine

Job including Video-Data :

$$Q_{CPU} > Q_{NetResp} > Q_{RAM}$$

Job including Image-Data :

$$Q_{RAM} > Q_{CPU} > Q_{NetResp}$$

Job including Audio-Data :

$$Q_{NetResp} > Q_{RAM} \geq Q_{CPU} \quad (1)$$

2) 자원 할당 모듈은 작업 처리 모듈의 요청에 따라 각각의 Queue 에 저장된 작업들을 전송한다. 작업 처리 모듈은 작업을 실제로 처리하는 모듈이며, 자원 할당 모듈의 Queue가 비어있는 경우에 작업 분류 모듈에게 신규 작업을 요청하거나 대기한다.

3.2.3 작업 처리 모듈

작업 처리 모듈은 분산 환경의 물리 계산 자원이다. 사용자가 요청한 작업을 실제로 처리하는 모듈이며, 작업 분류 모듈과 작업 할당 모듈에서 작업에 대한 저장을 모두 담당하고 있기 때문에 별도의 Queue 는 갖지 않는다.

작업 처리 모듈은 JCDT 아키텍처의 Virtual Machine 에 해당하는 알고리즘으로, Fig. 4와 같으며, 다음과 같이 동작한다.

1) 작업 처리 모듈은 자원 할당 모듈로부터 전달된 작업을 처리한다. 각각의 작업 처리 모듈은 각기 다른 성능을 가지고 있는데, 자원 할당 모듈이 분산 환경에 존재하는 작업 처리 모듈 즉, 계산 자원에 대한 성능을 고려하여 작업을 할당하기 때문에 단순히 작업을 처리하는 것으로 작업 처리 모듈의 역할이다.

2) 작업 처리 모듈은 처리가 완료된 작업을 작업 분류 모듈로 전송하여, 사용자의 요청에 응답한다.

4. 실 험

본 논문에서 제안하는 JCDT의 성능을 입증하기 위하여 DEVS(Discrete Event System Specification) 형식론(Zeigler 외, 2000)을 적용하여 가상의 분산 환경을 구성하고, 성능에 대한 지표로 평균 작업 처리시간 (Average of Job Processing Time) 과 서비스 처리율 (Throughput) (J.K., Kim 외 2013)을 측정한다.

Table 3. Node (Job Processing Module) Performance

Nodes	CPU	RAM	NetResponse
Node #0	9	8	8
Node #1	8	9	8
Node #2	8	8	9
Node #3	9	8	6
Node #4	8	9	7
Node #5	7	8	9

4.1 실험환경 구성

JCDT의 성능 테스트를 위하여 DEVS 형식론 기반으로 분산 환경을 구성하였으며, 모델은 Fig. 5와 같다.

본 논문에서 제안하는 JCDT 와 유사한 분산 환경을 구성하였고, 테스트 베드에 대한 설명은 다음과 같다.

첫째, User 는 작업을 요청 (생성) 하는 '사용자'를 모델링 한 것이며, 작업을 생성하거나 반환받는 Generator 에 상응한다.

둘째, Job Classifying Module은 '작업 분류 모듈'을 모델링 한 것이며, 작업을 데이터 특성에 기반하여 분류하고 저장하는 Queue 에 상응한다.

셋째, Resource Allocating Module은 '자원 할당 모듈'을 모델링 한 것이며, 각 작업을 계산 자원에 대한 의존도에 따라 재분류하고 할당하는 Queue-Processor 에 상응한다.

넷째, Node 는 작업을 처리하는 계산 자원 즉, '작업 처리 모듈'을 모델링 한 것이며, 각 작업을 단순히 처리만 하는 Processor 에 상응한다.

다섯째, Performance Evaluating Module 은 데이터의 특성을 기반으로 한 작업 분류방법을 적용했을 경우, 작업 처리에 대한 성능이 향상되는지의 여부에 대해 평가하는 모듈을 모델링 한 것으로, 실험에 따라 선택 사항이 될 수 있으며, Transducer 에 상응한다.

위와 같이 가상의 분산 환경을 구성하였으며, 각 노드 즉, 작업 처리 모듈의 성능 분포는 Table 3과 같고, 이에 대한 설명은 다음과 같다.

각 노드는 실제의 분산 환경에 존재하는 다양한 이기종 자원들을 의미한다. 이기종의 자원들 간의 성능 차이

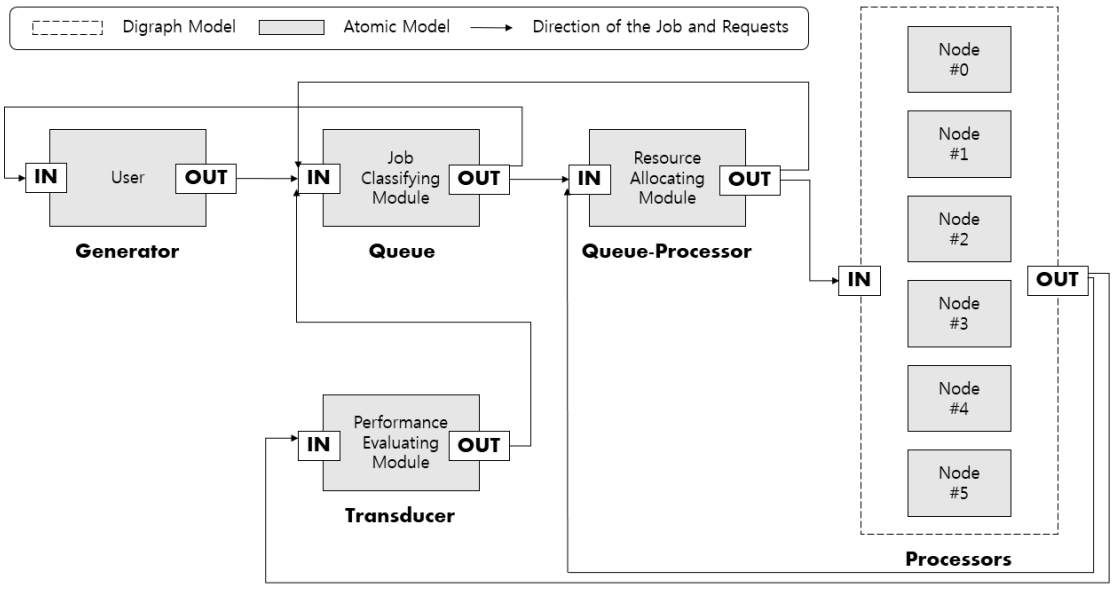

Fig. 5. Test Bed for JCDT based on DEVS Formalsim

는 본 실험에서 노드 즉, 작업 처리 모듈 간의 성능 차이로 대응되며, 각 노드의 성능 요소인 CPU, RAM, NetResponse의 수준은 0~9로 구분하였다.

본 실험에서 6개의 노드에 대한 성능 분포는 Node #0,3 은 CPU 성능이 높은 자원, Node #1,4 는 RAM 성능이 높은 자원, Node #2,5 는 NetResponse 의 성능이 높은 자원이며, 데이터 특성에 따라 각 자원의 성능에 대한 의존도가 다른 것을 보이기 위하여 Table 3과 같이 1개의 성능 요소에 대해 2개의 Node 가 최고 수준인 9 점을 갖는 가상의 분산 환경을 구성하였다.

JCDT 의 효율을 입증하기 위해 분산 환경에서 자원 할당에 사용되는 알고리즘 2가지와 그 알고리즘에 각각 JCDT를 적용한 알고리즘 2가지를 더하여, 총 4가지를 실험 모델로 선택하였다. 첫 번째 실험 모델은 라운드 로빈 알고리즘(Round Robin Scheduling Algorithm : RR) (Pooja 외, 2013), 두 번째 실험 모델은 JCDT가 적용된 라운드 로빈 알고리즘(Advanced Round Robin Scheduling Algorithm using JCDT : RR-JCDT), 세 번째 실험 모델은 최소 부하 알고리즘(Minimum Load First Scheduling Algorithm : MLFS)(Janaszka, T. 외 2012), 네 번째 실험 모델은 JCDT가 적용된 최소 부하 알고리즘(Advanced Minimum Load First Scheduling Algorithm using JCDT : MLFS-JCDT) 이다. 본 실험은 기존의 두 실험 모델에 JCDT를 적용하였을 경우에 성능이 향상되었는지의 여부를 각각 비교분석 한다.

4.2 실험 수행 및 결과

RR, RR-JCDT, MLFS, MLFS-JCDT 의 성능 측정을 위해서 평균 작업 처리시간과 서비스 처리율을 측정한다.

첫 번째 실험으로, 평균 작업 처리시간을 측정한다. 평균 작업 처리시간은 각 실험 모델이 동일한 개수의 작업을 처리하는데 소요되는 시뮬레이션 시간으로, 성능을 나타내는 지표이다. 사용자가 요청한 모든 작업이 처리되어 반환되기까지의 시간을 측정한 후에 완료된 작업으로 나누어서 계산하며, Eq. (2)와 같다.

$$Average\,of\,Job\,Processing\,Time = \frac{\sum_{i=1}^{n} Job\,Processing\,Time}{The\,number\,of\,Finished\,Jobs} \tag{2}$$

작업의 개수를 100에서 1000까지 증가시키며 평균 작업 처리시간을 측정한 결과, RR 과 RR-JCDT 에 대한 실험 결과는 Fig. 6과 같다. 또한, 동일한 조건으로 수행한 MLFS와 MLFS-JCDT 에 대한 실험 결과는 Fig. 7과 같다.

작업의 개수가 적을 때에는 기존의 알고리즘과 JCDT를 적용한 알고리즘의 성능에 큰 차이가 없었지만, 작업의 개수가 증가하면서 JCDT를 적용한 알고리즘이 기존

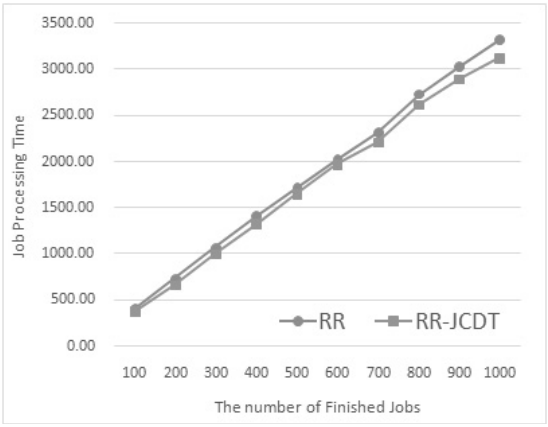

Fig. 6. Result of Average of Job Processing Time for RR and RR-JCDT

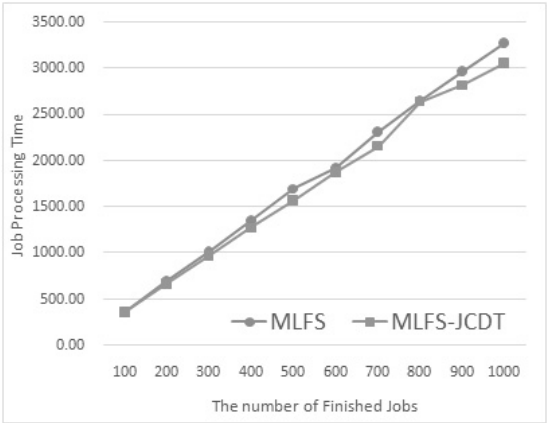

Fig. 7. Result of Average of Job Processing Time for MLFS and MLFS-JCDT

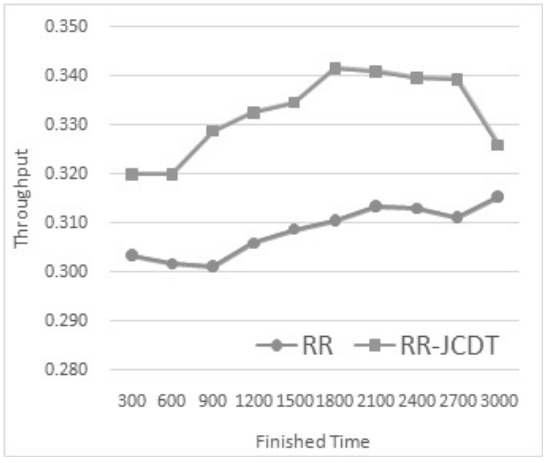

Fig. 8. Result of Throughput for RR and RR-JCDT

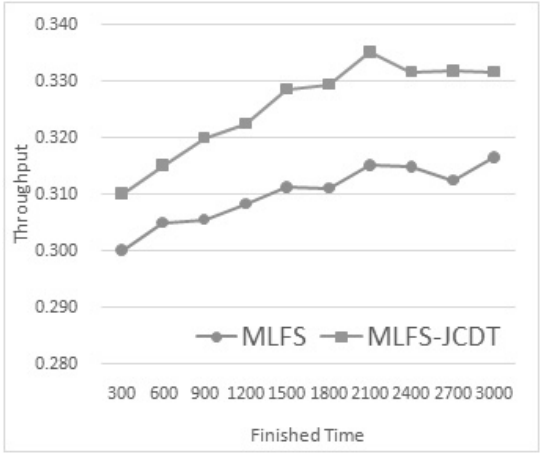

Fig. 9. Result of Throughput for MLFS and MLFS-JCDT

의 그것보다 약간 우세한 것을 알 수 있다. 즉, 작업의 개수가 1000 이내인 구간에서 RR-JCDT는 RR에 비해 평균적으로 약 5.49%의 작업 처리시간을 단축하였으며, MLFS-JCDT는 MLFS 에 비해 평균적으로 약 5.04%의 작업 처리시간을 단축하였다.

작업 처리시간을 단축할 수 있었던 이유는 사용자가 요청하는 작업에 포함된 데이터의 특성을 기반으로 작업을 분류하고, 해당 작업이 효율적으로 처리될 수 있는 계산 자원에 작업을 우선적으로 할당하였기 때문이다. 가령, 영상 및 동적 이미지에 대한 작업은 높은 성능의 CPU를 보유한 Node #0 혹은 #4에 우선적으로 할당되므로 나머지 Node 에 할당될 때보다 작업 처리시간이 단축되었다. 이와 같이 동일한 작업에 대해 작업 처리시간을 단축할 경우 더욱 원활한 분산 서비스의 제공이 가능하며, SLA 또한 보장할 수 있다.

두 번째 실험으로는, 서비스 처리율을 측정한다. 서비스 처리율은 각 실험 모델이 동일한 시간동안 처리할 수 있는 서비스의 총량을 비교하며, 성능을 나타내는 지표이다. 사용자가 요청한 서비스는 단위 작업으로 분할될 수 있으므로, 본 실험에서는 작업의 총량을 서비스의 총량으로 간주하였다. 서비스 처리율은 성공적으로 반환된 서비스의 개수를 완료된 시간으로 나누어서 계산하며, Eq. (3)과 같다.

$$Throughput =$$

$$\frac{The\ number\ of\ Service\ Response}{Finished\ Time} \quad (3)$$

완료된 시뮬레이션 시간을 300에서 3000까지 증가시키며 서비스 처리율을 측정한 결과, RR 과 RR-JCDT 에 대한 실험 결과는 Fig. 8과 같다. 또한, 동일한 조건으로 수행한 MLFS 와 MLFS-JCDT 에 대한 실험 결과는 Fig. 9와 같다.

시뮬레이션 시간이 3000 이내일 때, 각 실험 모델의 평균 서비스 처리율은 RR이 0.308, RR-JCDT 는 0.332, MLFS는 0.310, MLFS-JCDT는 0.326으로 나타났으며 JCDT를 적용한 알고리즘은 기존의 그것보다 약간 우세한 것을 알 수 있다.

서비스 (작업) 처리율을 높일 수 있었던 이유는 사용자가 요청하는 작업에 포함된 데이터의 특성을 기반으로 서비스를 처리하여, 각 노드의 단위 시간당 작업 효율이 증가하였기 때문이다. 결과적으로 동일한 시간 대비 서비스 처리율이 증가할 경우, 제공자 입장에서 서비스 품질을 보장하기 더 쉬울 것으로 간주된다.

5. 결 론

분산 환경 기반의 서비스는 대용량 데이터 및 복잡한 응용계산 작업을 처리할 수 있는 고성능 컴퓨팅 자원을 제공한다. 제공자 입장에서 한정된 자원으로 SLA를 보장하기 위해서는 계산 자원의 효율 증대가 필수적이다.

본 논문에서는 분산 환경에서 계산 자원의 효율 증대를 위한 데이터 특성 기반의 작업 분류방법 (JCDT) 를 제안한다. JCDT 는 사용자가 요구하는 작업에 포함된 데이터의 특성을 기반으로 작업을 분류하고, 계산 자원에 대한 의존도에 따라 작업을 할당하였으며, JCDT를 기존의 자원 할당 알고리즘에 적용함으로써 성능이 향상된 것을 확인하였다.

향후 연구로는 데이터의 세부 특성을 이용한 지능형 자원 관리기법을 연구하여, 이미 완료된 작업의 처리 과정에 대한 재학습을 통해 새로운 작업에 대한 서비스 처리속도를 개선할 예정이다.

References

1. Bernard P. Zeigler, Herbert Praehofer, Tag Gon Kim (2000), “Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems”, Academic Press, pp. 76-96, 2000.
2. B.J., Kim, “Service Quality Criteria for Voice Services over a WiBro Network”, The Journal of the Korea Institute of Electronic Communication Sciences, Vol. 6, No. 6, pp. 823-829, 2011.
3. B.S., Kim, S.D., Lee, T.G., Kwon and S.H., Lee, “Design and Implementation of the Unformatted Data Manager for Multimedia Storage System”, Journal of KIISE, Vol. 20, No. 2, pp. 191-194, 1993.
4. G, Kim, W, Lee and C, Jeon, “Virtualization Technology for Cloud Computing”, Journal of the Korea Society of Computer and Information, Vol. 18, No. 1, pp. 25-33, 2010.
5. H, Kang, J, Koh and Y, Kim, “A SLA-based VM Auto-Scaling Method in Hybrid Cloud Computing for Scientific Computational Applications”, Journal of KIISE, System and Theory, Vol. 40, No. 6, pp. 266-273, 2013.
6. H.S., Oh, “Tiled Image Compression Method to Reduce the Amount of Memory Needed for Image Processing in Mobile Devices”, Journal of Korea Game Society (2013), Vol. 13, No. 6, pp. 35-42, 2013.
7. Janaszka, T., Bursztynowski, D. and Dzida M., “On popularity-based load balancing in content networks”, Teletraffic Congress (ITC 24), 24th International. IEEE, pp. 1-8, 2012.
8. J.K., Kim and J.S., Lee, “Fuzzy Logic-driven Virtual Machine Resource Evaluation Method for Cloud Provisioning Service”, Journal of the Korea Society for Simulation, Vol. 22, No. 1, pp. 77-86, 2013.
9. Samal, Pooja and Pranati Mishra, “Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing”, International Journal of Computer Science and Information Technologies, Vol. 4, No. 3, pp. 416-419, 2013.
10. S.J., Lee, E.J., Lee, S.W., Hong, H.N., Choi and Y.W., Chung, “Secure and Energy-Efficient MPEG Encoding using Multicore Platforms”, Journal of the Korea Institute of Information Security and Cryptology, Vol. 20, No. 3, pp. 113-120, 2010.

문 성 환 (shmoon@inhaian.net)

2014 인하대학교 컴퓨터정보공학과 학사
2014～인하대학교 컴퓨터정보공학과 석사과정

관심분야 : 클라우드 컴퓨팅, 모델링 & 시뮬레이션

김 재 권 (jaekwonkorea@naver.com)

2011 가천의과학대학교 정보처리과 학사
2011 인하대학교 컴퓨터정보공학과 석사
2013～인하대학교 컴퓨터정보공학과 박사과정

관심분야 : 클라우드 컴퓨팅, 인공지능, 모델링 & 시뮬레이션

김 태 영 (taeyoung.kim@selab.inha.ac.kr)

2007 인하대학교 컴퓨터공학과 학사
2009 인하대학교 컴퓨터정보공학과 석사
2009～인하대학교 컴퓨터정보공학과 박사과정

관심분야 : 병렬 분산 컴퓨팅, 모델링 & 시뮬레이션

최 정 석 (jeongseokchoi.korea@gmail.com)

2011～인하대학교 컴퓨터정보공학과 학사과정

관심분야 : 클라우드 컴퓨팅, 모델링 & 시뮬레이션

조 규 철 (kccho@etri.re.kr)

2005 인하대학교 컴퓨터공학과 학사
2007 인하대학교 컴퓨터정보공학과 석사
2013 인하대학교 컴퓨터정보공학과 박사
2013～한국전자통신연구원 선임연구원

관심분야 : 패턴 인식, 그리드 컴퓨팅, 소프트웨어공학

이 종 식 (jslee@inha.ac.kr)

1993 인하대학교 전자공학과 학사
1995 인하대학교 전자공학과 석사
2001 미국 애리조나대 전기·컴퓨터공학과 박사
2001～2002 캘리포니아 주립대학교 전기·컴퓨터공학과 전임강사
2002～2003 클리블랜드 주립대학교 전기·컴퓨터공학과 조교수
2003～ 인하대학교 컴퓨터정보공학과 교수

관심분야 : 소프트웨어공학, 모델링 & 시뮬레이션