한국시뮬레이션학회 논문지
Vol. 18, No. 2, pp. 29-38 (2009. 6)

# 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델 및 시뮬레이션

노창현[1†] · 장성호[1] · 김태영[1] · 이종식[1]

# Semantic Computing-based Dynamic Job Scheduling Model and Simulation

Chang Hyeon Noh · Sung Ho Jang · Tae Young Kim · Jong Sik Lee

**ABSTRACT**

In the computing environment with heterogeneous resources, a job scheduling model is necessary for effective resource utilization and high-speed data processing. And, the job scheduling model has to cope with a dynamic change in the condition of resources. There have been lots of researches on resource estimation methods and heuristic algorithms about how to distribute and allocate jobs to heterogeneous resources. But, existing researches have a weakness for system compatibility and scalability because they do not support the standard language. Also, they are impossible to process jobs effectively and deal with a variety of computing situations in which the condition of resources is dynamically changed in real-time. In order to solve the problems of existing researches, this paper proposes a semantic computing-based dynamic job scheduling model that defines various knowledge-based rules for job scheduling methods adaptable to changes in resource condition and allocate a job to the best suited resource through inference. This paper also constructs a resource ontology to manage information about heterogeneous resources without difficulty as using the OWL, the standard ontology language established by W3C. Experimental results shows that the proposed scheduling model outperforms existing scheduling models, in terms of throughput, job loss, and turn around time.

**Key words** : Semantic computing, Dynamic job scheduling, Resource ontology, DEVS simulation

**요 약**

이기종의 자원들로 이루어진 컴퓨팅 환경에서 효율적인 자원 활용과 대용량의 데이터를 고속으로 처리하기 위해서는 실시간으로 변화하는 자원의 상태에 따라 대처 할 수 있는 동적인 작업 스케줄링 모델이 필요하다. 현재 이기종의 자원들에게 작업을 어떻게 분배 및 할당 할 것인지에 대하여 많은 자원 평가 방법 및 휴리스틱 기법들이 연구되었으나 이러한 방법들은 표준 언어를 사용하지 않기 때문에 시스템 호환 및 확장에 어려움이 많다. 또한 다양한 자원들의 상태가 실시간으로 동적으로 변화하기 때문에 기존 연구에서 제안한 방법으로는 효율적인 처리가 불가능하거나 자원의 상태 변화에 동적으로 대처할 수 없다. 본 논문은 이러한 기존 연구들의 문제에 대한 해결책으로 W3C에서 제정한 온톨로지 표준 언어인 OWL을 이용하여 자원 온톨로지를 구축함으로써 이기종의 자원 관리를 손쉽게 할 수 있으며, 자원의 동적인 변화에 따라 작업 스케줄링하는 방법을 지식 기반의 다양한 규칙들로 정의하여 추론을 통해서 최적의 자원에게 작업을 할당하는 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델을 제안한다. 시뮬레이션 실험 결과는 본 논문에서 제안한 작업 스케줄링 모델이 기존 모델에 비하여 낮은 작업 손실과 높은 작업 처리율 및 짧은 응답시간을 제공함으로써 이기종의 자원들로 구성된 시스템 전반에 걸쳐 안정적이고 고속의 데이터 처리를 제공할 수 있다는 사실을 증명한다.

**주요어** : 시멘틱 컴퓨팅, 동적 작업 스케줄링, 자원 온톨로지, DEVS 시뮬레이션

2009년 2월 9일 접수, 2009년 6월 6일 채택
[1)] 인하대학교 정보공학과
주 저 자 : 노창현
교신저자 : 노창현
E-mail; cromirak@hanmail.net

## 1. 서 론

고도의 정보화 시대가 실현되어 감에 따라 인간이 이용 할 수 있는 컴퓨팅 자원의 양은 점진적으로 증가하고 있다. 일기예보나 유전자 분석과 같이 자연과학과 공학 분야에서 대규모 데이터를 빠르게 처리하기 위해 과거에는 슈퍼컴퓨터와 같이 고성능 고가의 자원을 이용하였다[1]. 하지만 이러한 자원을 구성하기 위해서는 많은 예산이 소모되며 물리적인 공간 확보도 어려운 것이 사실이다. 그리드 컴퓨팅(Grid Computing)[2,3]은 이러한 단점을 극복하고자 지리적으로 분산되어있는 이기종의 자원들을 가상조직(Virtual Organization)으로 구성함으로써 자원을 보다 더 쉽게 공유하고 상대적으로 저가로 이용할 수 있는 방안으로 제시하였으며 현재 이와 관련된 많은 프로젝트와 연구가 활발히 진행되고 있다.

이기종의 자원들로 이루어진 그리드와 같은 컴퓨팅 환경에서 효율적인 자원 활용과 대용량의 데이터를 고속으로 처리하기 위해서는 자원의 상태 변화를 고려하지 않은 정적 작업 스케줄링 모델로는 한계가 있다. 자원의 동적인 변화에 따른 신속한 대처가 불가능하기 때문이다. 이기종 자원을 어떻게 배치하고 작업 스케줄링 할 것인지는 NP-Complete 문제[4]로 알려져 있으며 동적인 환경에서 작업 스케줄링하기 위한 방법으로 자원의 상태를 평가하거나 휴리스틱(heuristic)에 의한 방법에 대한 많은 연구가 진행되어졌다[5,6,7,8,9]. 하지만 이러한 방법들은 새로운 종류의 자원을 정의하여 추가 및 갱신하거나 사용자의 요구사항과 처리 할 데이터의 특성에 따른 최적의 작업 스케줄링 기법을 적용하기에는 호환성과 확장성에 문제가 있으며 실시간으로 변화하는 자원의 동적인 상태를 반영하여 효율적으로 작업 스케줄링하기 어렵다.

따라서 본 논문에서는 W3C[10](World Wide Web Consortium)에서 제정한 온톨로지 표준 언어인 OWL(Web Ontology Language)을 이용하여 지리적으로 분산되어 있는 자원들의 정보와 관계를 온톨로지로 구축함으로써 자원 정보의 추가 및 삭제, 갱신 등을 용이하게 하고 확장이 유연한 구조로 관리한다. 구축된 자원 온톨로지로부터 개체(individual)들을 생성한 후, 효율적이고 안정적인 작업 처리를 위하여 사용자의 요구사항이나 자원의 특성에 맞게 자원을 평가하고 작업 스케줄링 기법들을 지식 기반의 규칙(rule)으로 정의하여 추론 엔진을 통해서 효율성을 높일 수 있는 정제된 자원에게 작업을 할당함으로써 이기종으로 구성된 컴퓨팅 시스템의 전반적인 성능을 향상 시킬 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서 본 논문의 기초 연구가 되는 이기종 자원의 작업 스케줄링 기법과 시멘틱 컴퓨팅에 대하여 알아보고, 3장에서는 본 논문에서 제안하는 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델에 대하여 기술한다. 그리고 4장에서 시뮬레이션을 통하여 비교 모델과 성능을 평가한 후 5장에서 결론을 맺는다.

## 2. 관련 연구

### 2.1 작업 스케줄링

작업 스케줄링 모델은 크게 로컬 작업 스케줄링 모델[11]과 글로벌 작업 스케줄링 모델로 나눌 수 있다. 지리적으로 분산되어 있고 이기종의 자원으로 이루어져 있는 컴퓨팅 환경은 글로벌 작업 스케줄링 모델을 적용하며, 이는 다시 정적 작업 스케줄링 모델과 동적 작업 스케줄링 모델로 구분 할 수 있다.

정적 작업 스케줄링 모델(Static Job Scheduling Model)[12]은 작업 및 자원의 정보, 구성과 개수에 대한 변동이 없을 때 적용하는 방식이다. 자원의 변화에 대하여 적응하지 못하기 때문에 실행 중 우선순위를 바꾸지 못하지만 구현이 간단하고, 오버헤드가 적다는 장점이 있다. 최초 자원에 대한 성능을 평가한대로 작업을 할당하기 때문에 작업의 특성과 사용자의 예산과 마감기한에 따라 자원의 구성과 정보가 변화하는 동적인 환경에서는 적합하지 않다. 동적 작업 스케줄링 모델(Dynamic Job Scheduling Model)[12]은 자원의 상태가 유동적으로 변화 될 때 적용 할 수 있는 방식으로서 네트워크와 자원의 상태가 빈번하게 실시간으로 변화하는 인터넷 서비스, 그리드 컴퓨팅, 클러스터, 모바일 네트워크, 센서 네트워크 등의 분야에서 사용된다. 변화하는 자원의 환경에 따라 우선순위를 변경하여 작업 스케줄링하기 때문에 시스템의 응답 속도를 증가시켜 효율적이지만 구현이 복잡하고 오버헤드가 많다는 단점이 있다.

작업 스케줄링 기법 중 자원의 성능을 평가하는 방식으로 MET(Minimum Execution Time), MCT(Minimum Completion Time), RM[13](Reliability Measurement) 등이 있다. MET는 가장 작은 실행 시간을 갖는 자원에게 작업을 할당하는 방식으로서 자원 이용의 불균형을 초래할 수 있고, MCT는 작업 완료 시간이 가장 작은 자원에게 할당하는 방식으로서 이기종의 다양한 자원의 정보를 활용하거나 효율적으로 데이터를 처리기에는 한계가 있다. 즉, MET는 작업의 정보와 자원의 성능에 따른 기대되는 실행시간을 의미하며, MCT는 대기 큐에 쌓인 작업들을 고려하여 할당 될 작업이 처리되기 위해 필요로 하

는 준비시간과 실행시간을 더한 작업 완료시간 중 가장 작은 완료시간을 가지는 자원에게 작업을 할당하는 방식이다. RM은 자원의 신뢰성 측정 계산식을 통하여 가장 높은 신뢰성을 가지는 자원에게 작업을 할당하는 방법으로서 자원의 신뢰성이 과거의 자원 상태 데이터에 의존함으로써 자원 이용 불균형 및 효율적이지 못한 결과를 초래 할 수 있다. 또한 이러한 방법들은 표준 언어를 사용하지 않기 때문에 시스템 간의 호환에 문제가 있으며 확장하기에도 어려움이 많다. 그리고 자원을 평가하는 새로운 방법, 혹은 작업 스케줄링 방법을 수정 및 추가하는 데에도 한계가 있다.

본 논문에서는 이러한 문제를 해결하고자 온톨로지 표준 언어인 OWL을 이용하여 자원을 표현 및 관리하고, 다양한 동적인 자원의 변화에 신속히 대처 할 수 있도록 여러 가지 상황에 맞는 규칙을 지식으로 구축하여 해결하고자 한다.

### 2.2 시멘틱 컴퓨팅

시멘틱 컴퓨팅은 원래 W3C[10]에서 웹 데이터에 의미적인 정보와 관계를 부여하여 지능적이고 효율적인 검색을 도모하고자 개발 된 시멘틱 웹(semantic web)에서 확장 된 개념이다. 해당 분야의 도메인 정의를 위한 용어 및 정보와 관계 등을 온톨로지(ontology)로 구축하고 표현한 후 지식 기반의 의미를 부여함으로써 지능적이고 고급 정보 처리를 가능하게 한다. 본 논문에서 이기종의 자원으로 이루어진 환경에서 효율적인 작업 스케줄링을 위한 자원 온톨로지를 구축하기 위해서 프로티지[14](protégé)를 사용하였다.

구축 된 자원 온톨로지로부터 이기종 자원의 생성 및 수정사항이 발생하게 되면 해당 자원 관리자나 소유자에 의해서 표준 언어인 OWL의 형태로 자원 정보를 갱신 할 수 있다. 생성 및 수정이 용이한 자원 온톨로지를 기반으로 본 논문에서 제안한 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델에 지식 기반의 규칙을 정의한다. 정의 한 규칙으로부터 추론 엔진을 거쳐 정제 된 자원에게 작업을 스케줄링 하면 기존의 자원 성능을 평가하는 방식 보다 더 효율적인 작업 스케줄링이 가능하고, 제한 된 상황에서의 휴리스틱에 의한 작업 스케줄링 기법을 적용하는 것보다 더 빠르고 안정적인 작업 처리 및 확장이 용이한 장점을 가질 수 있다.

본 논문에서 구축한 자원 온톨로지를 통해 이기종 자원 정보의 생성, 삭제, 갱신이 용이해지며 국내 기술로 개발 된 RETE 기반의 보쌈[15](Bossam) 추론 엔진을 통하여 지식 기반의 규칙을 정의하고 추론에 의한 작업 스케줄링 기법을 적용함으로써 이기종 자원으로 이루어진 컴퓨팅 시스템의 효율적인 데이터 처리와 안정성을 제공한다.

## 3. 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델

이 장에서는 이기종의 자원으로 이루어진 컴퓨팅 환경에서 다양한 상태 변화에 대처하며 효율적으로 작업을 처리 할 수 있는 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델(SCDJSM : Semantic Computing-based Dynamic Job Scheduling Model)을 제안한다. 또한, 효율적으로 자원을 관리하고 시스템간의 호환성과 확장성을 위해 구축 된

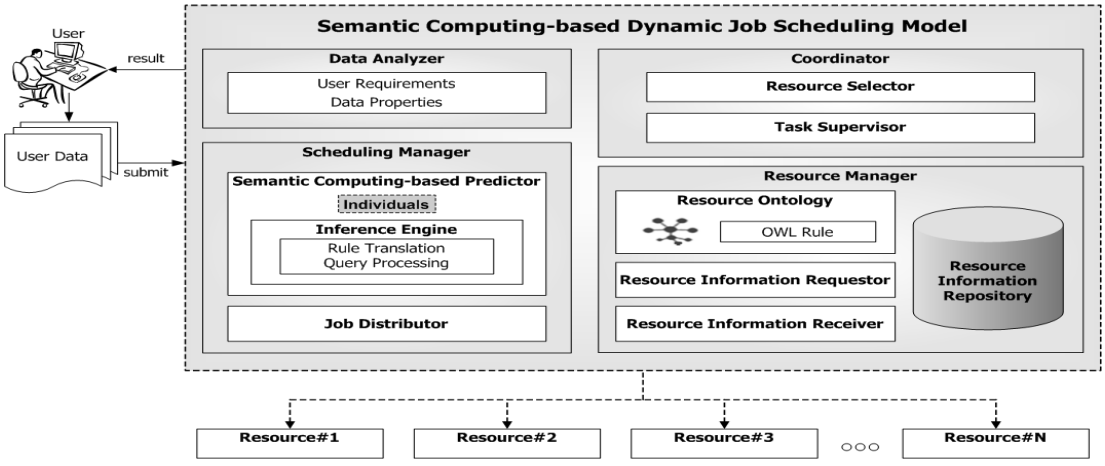


**그림 1.** 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델

자원 온톨로지에 대해 소개하고 작업 처리 시 발생 가능한 여러 가지 다양한 상황에 적합한 규칙 기반의 지식을 생성하여 이를 바탕으로 작업을 스케줄링하는 기법에 대해 설명한다.

### 3.1 시스템 모델링

본 논문에서 제안한 SCDJSM은 그림 1과 같이 자원 관리 컴포넌트(Resource Manager), 데이터 분석 컴포넌트(Data Analyzer), 중재 컴포넌트(Coordinator), 작업 스케줄링 관리 컴포넌트(Scheduling Manager)로 구성되어 있다.

그림 1에서 자원 관리 컴포넌트가 자원 정보의 수집, 저장, 갱신 및 삭제와 자원 온톨로지(Resource Ontology)를 자동으로 생성하는 역할을 수행 한다. 자원 정보 요청 모듈(Resource Information Requester)이 지리적으로 분산되어 있는 자원들에게 주기적으로 정보를 요청하면 자원들은 이에 대한 응답 메시지를 보내게 되고, 자원 정보 수신 모듈(Resource Information Receiver)이 수신하여 저장소(Resource Information Repository)에 해당 자원 정보를 저장 및 갱신, 삭제하게 된다. 또한 자원들은 요청 메시지가 오지 않았더라도 자신의 정보가 변경되는 즉시 이벤트를 발생 시켜서 갱신 메시지를 보내게 된다.

데이터 분석 컴포넌트는 작업 처리를 요청한 사용자의 데이터를 수신하고, 요구사항(User Requirements)과 특성(Data properties)을 추출하여 중재 컴포넌트로 전송한다. 사용자는 자신의 예산 및 처리 마감기한에 대한 정보와 작업의 특성 등을 자원 선택을 위해 미리 정의 된 XML 형태의 메시지로 전송한다. 데이터 분석 컴포넌트는 XML 메시지를 구문분석(Parsing)하여 자원 선택에 필요한 정보를 중재 컴포넌트로 전달하며 작업 분배 모듈(Job Distributor)에 의해 데이터가 분배 및 할당되기 전 처리해야 할 데이터를 저장하고 있는 역할을 한다.

중재 컴포넌트는 데이터 분석 컴포넌트로부터 자원 선택에 필요한 정보를 전달 받아 부합되는 자원을 선택하고, 전체 작업 수행에 대한 관리 감독의 역할을 수행한다. 자원 선택 모듈(Resource Selector)이 사용자의 요구사항과 데이터 특성에 부합되는 자원을 선택 할 때, 자원 저장소에 유지되어 있는 자원의 정보를 바탕으로 그림 2의 자원 온톨로지 형태로 개체들을 생성하여 자원 선택에 관한 규칙 및 스케줄링 알고리즘에 의하여 후보군 중 가장 적합한 자원들을 선택한다. 작업 관리 모듈(Task Supervisor)은 전체 작업 수행에 대한 감독 역할을 수행하는 모듈로써, 선택한 자원이 작업 처리 중 가용한 상태가 아니거나

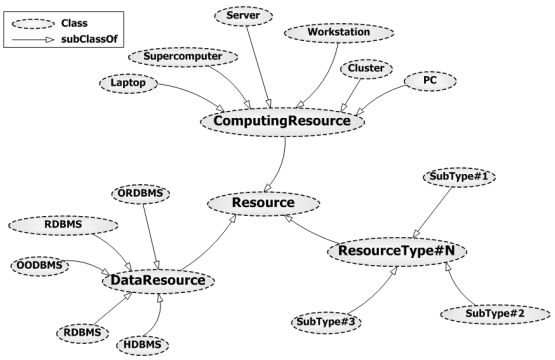


**그림 2.** 자원 온톨로지

**표 1.** Resource 클래스의 정적 속성

| Property Name | Description | Value Type | Min. Value | Unit |
|---|---|---|---|---|
| ID | 자원 식별자 | Integer | 0 | --- |
| OSType | OS의 종류 | String | --- | --- |
| CPUArchType | CPU의 구조 | String | --- | --- |
| CPUSpeed | CPU 최대 클럭 속도 | Float | 700 | Mega hertz |
| CPUNumber | CPU 개수 | Integer | 1 | |
| CacheSize | cache 메모리의 크기 | Float | 256 | Kilo Bytes |
| RAMSize | 주 기억 장치의 크기 | Float | 32 | Mega Bytes |
| StorageSize | 보조 기억장치의 크기 | Float | 10 | Giga Bytes |
| bandwidth | 네트워크의 대역폭 | Float | --- | Mega BPS |
| cost | 자원의 사용 금액 | Integer | --- | G$ |
| . . . | . . . | . . . | . . . | . . . |

작업을 실패 할 경우 이에 따른 작업 재전송 및 재처리를 명령하는 역할을 수행하며 최종 결과를 사용자에게 전송하는 역할을 담당한다.

작업 스케줄링 관리 컴포넌트는 지식 기반의 작업 스케줄링을 위한 추론 엔진(Inference Engine)을 포함한 시멘틱 컴퓨팅 기반 예측 모듈(Semantic Computing-based Predictor)과 사용자의 작업을 자원에게 할당하는 작업 분배 모듈로 구성되어 있다. 시멘틱 컴퓨팅 기반 예측 모듈은 그림 2의 자원 온톨로지로 생성한 OWL 파일을 JAVA API로 불러온 후, 리소스의 종류에 따라 표 1의 정적인 속성으로 개체를 생성한다. 이후 작업 스케줄링이 진행되면서 동적 속성을 가진 자원 개체를 생성 및 유지하고, 정의한 규칙에 의하여 추론엔진을 통해 규칙 해석과 질의를

**표 2.** Resource 클래스의 동적 속성

| Property Name | Description | Value Type | Value Range |
|---|---|---|---|
| CPUStatus | CPU 사용여부 | Boolean | true or false |
| CPUUtilization | CPU 사용률 | Float | 0~100 |
| RAMUtilization | 주기억 장치 사용률 | Float | 0~100 |
| RAMAvailable Size | 주기억 장치 가용 공간 | Float | 0~Available memory size |
| StorUtilization | 보조 기억 장치 사용률 | Float | 0~100 |
| StorAvalableSize | 보조 기억 장치 가용 공간 | Float | 0~Available storage size |
| NetStatus | 네트워크의 상태 | boolean | true or false |
| JobCompletion Time | Buffer에 쌓여있는 Job이 처리완료 되는 예상시간 | Float | 0~Estimated Time |
| Rank | Resource의 순위 | Integer | 1~10 |
| . . . | . . . | . . . | . . . |

처리한 후, 정제된 자원 중 작업 처리 효율을 높이기 위한 최적의 자원을 선별하여 작업 분배 모듈에게 알려준다. 작업 분배 모듈은 시멘틱 컴퓨팅 기반 예측 모듈로부터 전달받은 자원에게 작업을 분배하고, 처리가 완료된 개개의 작업을 자원으로부터 수신하여 결과 파일을 만드는 역할을 수행한다.

### 3.2 온톨로지 구축

이기종의 자원을 실시간으로 변화하는 동적인 상황에서 지능적이며 효율적으로 동작하는 규칙 기반의 작업 스케줄링 방법을 제공하기 위해 본 논문에서는 자원 정보를 바탕으로 그림 2와 같이 자원 온톨로지를 구축하였다. 우리는 SCDJSM에서 자원을 최상위 클래스 Resource로 만들었으며, 자원의 종류에 따라 컴퓨팅 자원(Computing-Resource)과 데이터 자원(DataResource)으로 구분하였다. 또한 이전에는 없었던 새로운 종류의 자원이 추가될 경우 새로운 타입의 자원을 ResourceType#N 클래스로 정의해서 온톨로지에 포함 시켜 시스템 확장이 용이하고 유연한 구조를 제공한다.

다양한 동적인 변화에 적합한 작업 스케줄링 기법을 제공하기 위해서는 여러 상황에 적합한 추론 방법을 지식 기반의 규칙으로 정의해야 한다. 추론을 하기 위해서는 개체들이 필요한데, 본 논문에서는 개체가 생성 될 때 하드웨어 사양과 같은 정적인 속성 값과 네트워크 상태나 CPU 사용률, 메모리 가용률에 따른 동적인 속성 값을 가진다. 자원 온톨로지에서 새로운 종류의 자원이 발견 되면 추가 될 수 있듯이, 속성 값 역시 자원을 설명하는데 필요에 의해 추가 및 삭제, 변경 등이 자유롭다. 표 1은 Resource 클래스의 정적인 속성에 대한 설명이고, 표 2는 동적인 속성을 나타낸다.

**표 3.** 동적 작업 스케줄링을 위한 규칙

| Rule | Domain rule |
|---|---|
| 1 | Resource(?x) ∧ StorAvalableSize(?x, ?y) ∧ [?y > jobSize] → AvailableResource(?x) |
| | 남아있는 디스크 공간이 처리 할 job의 크기보다 크면 가용자원으로 정의한다. |
| 2 | Resource(?x) ∧ CPUSpeed(?x, ?y) ∧ [?y > 3.0] ∧ CacheSize(?x, ?z) ∧ (?z > 4096) ∧ CPUUtilizationr(?x, ?k) ∧ [?k < 30] → HighSpeedLowWorkloadResource(?x) |
| | CPU의 최대 클럭 속도가 3.0GHz 이상이고, cache의 크기가 4096 kilobytes 이상이고, CPU 사용률이 30% 미만이면 낮은부하의고속처리CPU자원으로 정의한다. |
| 3 | Resource(?x) ∧ CPUUtilization(?x, ?y) ∧ [?y > 90] ∧ StorUtilization(?x, ?z) ∧ [?z > 90] ∧ RAMUtilization(?x, ?k) ∧ [?k > 90] ∧ NetStatus(?x, ?n) ∧ [?n > threshold] → hasRank(?x, 1) |
| | CPU 사용률, 디스크 사용률, 메모리 사용률이 모두 90%를 초과하고 네트워크가 혼잡상태이면 자원의 Rank는 1이다. |
| 4 | Resource(?x) ∧ CPUUtilization(?x, ?y) ∧ [?y > 40] ∧ [?y < 50] ∧ StorUtilization(?x, ?z) ∧ [?z > 40] ∧ [?z < 50] ∧RAMUtilization(?x, ?k) ∧ [?k > 40] ∧ [?k < 50] ∧ NetStatus(?x, ?n) ∧ [?n < threshold] → hasRank(?x, 5) |
| | CPU 사용률, 디스크 사용률, 메모리 사용률이 40~50% 이고, 네트워크가 정상 상태이면 자원의 Rank는 5이다. |
| 5 | Resource(?x) ∧ CPUUtilization(?x, ?y) ∧ [?y < 10] ∧ StorUtilization(?x, ?z) ∧ [?z < 10] ∧ RAMUtilization(?x, ?k) ∧ [?k < 10] ∧ NetStatus(?x, ?n) ∧ [?n < threshold] → hasRank(?x, 10) |
| | CPU 사용률, 디스크 사용률, 메모리 사용률이 모두 10% 미만이고 네트워크가 정상 상태이면 자원의 Rank는 10이다. |
| 6 | Resource(?x) ∧ NetStatus(?x, ?y) ∧ [?y < threshold] ∧ bandwidth(?x, ?z) ∧ [?z >= 100] → HighSpeedNetworkResource(?x) |
| | 네트워크 환경이 100Mbps 이상을 지원하고, 임계값 보다 작은 RTT를 가지면 고속네트워크자원으로 정의한다. |
| . . . | . . . |

### 3.3 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 알고리즘

표 3은 본 논문에서 작업 스케줄링을 위해 정의한 규칙과 설명을 나타낸 것으로, 시스템 관리자나 전문가에 의해서 규칙을 추가로 정의 하거나 우선순위를 동적으로 변화하는 자원 및 주변의 상황에 따라 변경 할 수 있다.

표 3의 규칙은 Resource 클래스의 정적인 속성과 동적

인 속성을 동시에 조합하여 만들 수 있다. 이렇게 여러 가지 상황에 효율적으로 작업을 처리 할 수 있는 규칙을 시멘틱 컴퓨팅 기반 예측 모듈에 정의 한다. 정의한 규칙으로부터 보쌈 추론 엔진을 통해서 처리할 자원을 선별한 후 선별 된 자원에게 작업을 할당 한다.

본 논문에서 사용한 스케줄링 기법을 그림 3에 의사코드(pseudo code)로 나타내었으며 방법은 다음과 같다. 데이터 분석 컴포넌트로부터 작업의 크기를 입력 받으면 시멘틱 컴퓨팅 기반 예측 모듈은 자원 온톨로지 OWL 파일을 불러와서 정적인 속성과 동적인 속성으로 개체들을 생성해서 리스트 형태로 자원 개체 집합인 *RST*를 만든다. *RST* 중 가용 기억 장치의 크기(*SAS*)가 작업의 크기보다 큰 자원들을 후보군 리스트 *CRL*에 삽입한다. 시멘틱 컴퓨팅 기반 예측 모듈은 자원이 송신하는 상태 메시지를 수신하여 동적인 상태 정보에 대하여 알고 있는데, 그 정보

```
Input : job size → JS
Output : x (resource index)
1. Import the OWL document from the resource ontology to the bossam inference
   engine of SCP
2. Creates the set RST of individuals representing resources ontology
3. Execute some inference rules to find candidates from RST
   CRL set null //Candidate Resource List
   for i = 0 to RST.length
     if JS < RST[i].SAS
       ADD RST[i] to CRL
     end if
   end for
4. Calculate the job completion time JCT, CPU utilization CU, RAM utilization RU,
   RAM available size RAS, storage utilization SU, storage available size SAS each
   individual in CRL
5. Set the rank of resources using CU, RU, RAS, SU, SAS to RR
   RR is the Resource Rank
6. Sort CRL by RR in an descending order
   for i = 0 to CRL.length
     for int j = 1 to CRL.length
       if CRL[i].RR < CRL[j].RR then SWAP(CRL[i].RR, CRL[j].RR)
       if CRL[i].RR == CRL[j].RR then sort CRL by JCT in an ascending order
       if (CRL[i].RR == CRL[j].RR && CRL[i].JCT == CRL[i].JCT) then sort CRL
         by NetStatus in an ascending order
     end for
   end for
7. Return the resource index of CRL[0]
```

**그림 3.** 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 알고리즘

를 바탕으로 작업 처리 완료 예상시간(*JCT*), CPU 이용률(*CU*), 기억장치 이용률(*RAS*), 저장 장치 이용률(*SU*)을 *CRL*의 개체마다 계산한다. 그리하여 자원에게 랭킹(ranking)을 부여하는데, 그 방법은 *CU*, *RAS*, *SU*를 이용하여 표 3의 규칙 3, 4, 5번과 같이 정의 할 수 있다. 계산 된 랭킹이 높은 순으로 *CRL*의 개체들을 정렬하고, 만약 같은 랭킹의 개체들이 존재하면 *JCT*가 작은 순으로 정렬한다. 랭킹과 *JCT* 마저도 같으면 응답속도(NetStatus)가 좋은 순으로 정렬한다. 이상의 절차를 거친 *CRL* 리스트의 첫 번째 인덱스에 있는 자원이 현재 처리 할 작업을 할당 받게 될 자원이 되고, 이 자원의 인덱스 정보를 시멘틱 컴퓨팅 기반 예측 모듈이 작업 분배 모듈로 전송하여 스케줄링 하게 된다.

표 3의 규칙과 그림 3에서 제시한 스케줄링 기법 이외에도 자원의 동적인 랭킹을 부여하는 방법으로 여러 가지 정책을 규칙에 포함 시킬 수 있음으로써 동적인 상황에 적절하게 대처 할 수 있다.

## 4. 시뮬레이션 및 결과 분석

본 논문에서 제안한 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델이 효율적이며 지능적으로 작업을 스케줄링 한다는 것을 증명하기 위해 이기종의 자원으로 구성된 컴퓨팅 환경에서 작업을 분배하여 처리 할 때, 측정 가능한 성능 지표인 작업 손실량(Job Loss), 평균 작업 처리율(Throughput), 평균 작업 처리 시간(Average of Turn Around Time)을 통해 비교 모델과 성능을 평가한다.

### 4.1 모델 구현 및 시뮬레이션 환경

우리는 본 논문에서 제안한 SCDJSM을 DEVS[16] 형식론을 적용하여 그림 4와 같이 컴포넌트를 구현한 후 시

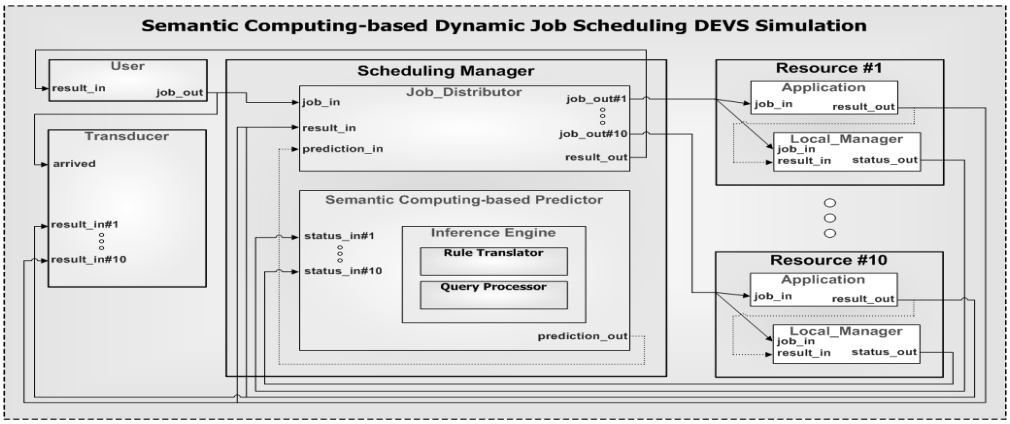


**그림 4.** DEVS로 시뮬레이션한 SCDJSM

뮬레이션 하였다. 그림 1과 그림 4에서 동일한 이름으로 표기 된 컴포넌트는 앞서 3장에서 설명한 것과 같이 동작하며, 그림 4에서 자원 부분에 Local_Manager 컴포넌트를 추가하였다. 이 컴포넌트는 자원의 상태나 작업의 처리 상태를 시멘틱 컴퓨팅 기반의 예측 컴포넌트에게 알리는 역할을 수행한다. 성능을 비교하기 위하여 Round-Robin Job Scheduling Model[17](RRJSM), Random Job Scheduling Model(RAJSM), Reliability Measurement[13] Job Scheduling Model(RMJSM)들을 추가로 구현하였고, 4개의 작업 스케줄링 모델 모두 동일한 조건으로 실험하여 결과를 비교하였다. RRJSM는 처리해야 할 작업을 순차적으로 분배하는 방식이고, RAJSM는 임의의 자원에게 분배한다. RMJSM는 자원의 신뢰성을 측정한 History data를 바탕으로 가장 높은 신뢰성을 가진 자원에게 할당하는 방식이다.

그림 4에서 User는 자원이 처리 할 작업을 일정 간격으로 발생시키며 Job_Distributor가 표 3의 규칙으로 Semantic Computing-based Predictor의 추론 엔진을 통하여 작업을 처리 할 자원을 선정한 후 작업을 할당한다. 본 논문에서 제안한 스케줄링 모델과 비교 모델들의 동일한 실험 환경을 위하여 자원의 개수는 10개로 고정하였다. 자원들은 각각의 모델들의 스케줄링 기법에 의해 할당 받은 작업을 처리하고 완료 된 작업들을 Transducer로 전달한다. 자원의 처리시간 및 저장 할 수 있는 매체의 용량 등은 표 1의 형태로 이기종의 자원으로 구성한 후, Transducer 컴포넌트가 작업 손실량, 평균 작업 처리율, 평균 작업 처리 시간을 측정한다. 실험결과에 표기 된 DEVS Time(sec)은 시뮬레이션 수행 시간으로서 일반 시간과는 다른 개념의 time unit이다.

## 4.2 실험 결과

### 4.2.1 실험의 목적

이기종의 자원을 이용 할 때 자원의 이질성으로 인하여 나타날 수 있는 문제점 중 하나인 작업 손실이 비교 모델의 성능과 견주어 어느 정도 감소 할 수 있는지에 대한 실험 및 검증이 필요하다. 또한 이기종 자원을 사용하는 목적중 하나인 처리속도 향상이 어느 정도 이루어졌는지에 대하여 평균 작업 처리율, 평균 작업 처리 시간으로 측정해야 한다. 따라서 자원의 이질성을 극복하고 효율적인 작업 처리를 위한 스케줄링 기법을 찾기 위하여 제안한 본 논문의 스케줄링 기법에 포함 된 규칙과 알고리즘이 동적인 자원 상태의 변화에 따라 적절히 대처하는지 검증하는 것이 실험 목적이다.

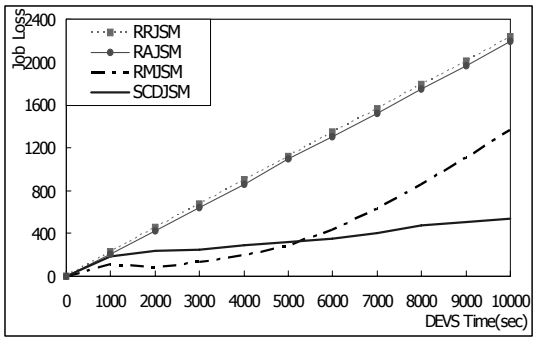


**그림 5.** 작업 손실 비교

### 4.2.2 실험 결과 1 : 작업 손실량

첫 번째 실험은 4개의 작업 스케줄링 기법을 적용하였을 때, 주어진 작업을 얼마나 안정적으로 처리할 수 있는지에 대하여 알 수 있는 손실된 작업의 개수를 측정하였다. 그림 5의 결과에서 보듯이 10000 DEVS Time 동안 RRJSM, RAJSM, RMJSM, SCDJSM의 순서대로 2230, 2193, 1367, 538개의 작업 손실을 기록하였다. RRJSM과 RAJSM의 작업 손실 개수가 비슷한 수준이었으며, 0~5000 DEVS Time에서 RMJSM이 SCDJSM보다 더 낮은 작업 손실량을 기록하였다. 이는 RMJSM이 자원의 신뢰성을 누적된 값으로 측정하여 가장 높은 신뢰성을 보이는 자원에게 작업 스케줄링을 하는데, 0~5000 DEVS Time 동안에는 최대 성능이나 혹은 그와 비슷한 성능을 가지는 자원에게 계속 할당하여 좋은 결과를 보였으나, 5000 DEVS Time 이상부터는 그 자원의 처리 능력이나 저장 능력이 한계에 이르러 SCDJSM보다 더 많은 작업 손실을 발생 시켰다. 즉, 자원에게 작업을 분배 할 때 단순히 신뢰성에 의존함으로써 작업 부하를 적절히 분산시키지 못한다는 것을 알 수 있다. 이와는 대조적으로 SCDJSM의 작업 손실량은 시뮬레이션 시간이 지남에 따라 크게 증가하지 않으므로 보다 더 안정적으로 작업을 처리 할 수 있으며 자원의 동적인 변화에 따라 부하량을 적절히 분산시킨다는 것을 증명한다.

### 4.2.3 실험 결과 2 : 평균 작업 처리율

두 번째 실험은 작업 스케줄링 모델들의 단위 시간당 처리 된 작업의 개수를 나타내는 평균 작업 처리율을 측정하였다. 그림 6의 결과에서 보듯이 네 개의 작업 스케줄링 기법을 적용하였을 때의 평균 작업 처리율은 RRJSM, RAJSM, RMJSM, SCDJSM의 순서로 0.277, 0.281, 0.363, 0.446을 기록하였다. 작업 손실의 결과와 비슷한 패턴으

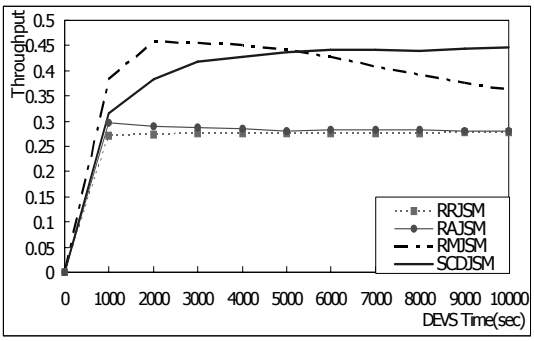


**그림 6.** 평균 작업 처리율 비교

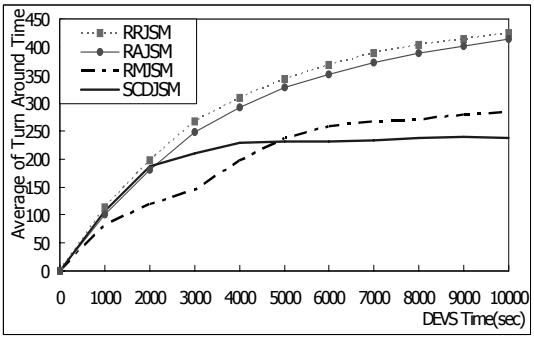


**그림 7.** 평균 작업 처리시간 비교

로 RRJSM과 RAJSM은 비슷한 성능을 보였으며 5000 DEVS Time 이후부터 SCDJSM이 다른 모델 보다 더 나은 성능을 기록하였다. RMJSM이 SCDJSM보다 더 낮은 평균 작업 처리율을 기록하는 이유는 자원의 신뢰성을 측정 할 때, 자원의 상태 측정 시 과거부터 누적된 데이터를 사용함으로써 현재의 자원 상황을 정확히 반영하지 못하여 효율적인 작업 스케줄링을 하기에는 한계가 있다는 것을 의미한다. 즉, SCDJSM이 가장 높은 평균 작업 처리율을 기록한 그림 6의 결과는 이기종의 자원으로 이루어진 컴퓨팅 환경에서 같은 시간에 더 많은 작업을 효율적으로 처리 할 수 있다는 것을 의미한다.

#### 4.2.4 실험 결과 3 : 평균 작업 처리 시간

세 번째 실험은 평균 작업 처리시간을 측정하였다. 평균 작업 처리시간은 요청된 하나의 작업 당 처리가 완료되어 반환되는 시간의 평균값이다. 그림 7은 네 개의 작업 스케줄링 모델의 평균 작업 처리시간을 나타내는 것으로써 RRJSM, RAJSM, RMJSM, SCDJSM의 순서로 425.23, 414.38, 284.79, 237.28을 기록하였다. 하나의 작업을 처리 할 때마다 짧은 응답시간을 제공한다는 것은 SCDJSM이 다른 모델에 비하여 효율적인 작업 스케줄링 방법을 제공한다는 것을 의미한다.

### 4.3 실험의 의미

그림 5와 그림 6, 그림 7을 통하여 우리는 SCDJSM 모델이 작업을 처리 할 때, 낮은 작업 손실량과 높은 작업 처리율 및 짧은 응답시간을 제공한다는 것을 알 수 있다. 작업 손실량의 경우, RMJSM이 단순히 자원의 과거 누적 상태 값에 의존한 높은 신뢰성을 가진 자원에게 스케줄링 함으로써 작업 부하를 분산시키지 못한 반면, 규칙에 의하여 같은 랭킹에 의한 자원이라 하더라도 동적으로 변화하는 CPU 사용률이나 디스크 사용률과 같은 여러 가지 사항을 고려하여 스케줄링한 SCDJSM이 작업 부하를 효율적으로 분산시키며 보다 더 안정적으로 작업을 처리 할 수 있음을 의미한다. 평균 작업 처리율의 결과는 SCDJSM이 가장 높은 값을 기록함으로써, 같은 작업과 이기종으로 이루어진 동일한 자원 집합이라 할지라도 자원의 동적인 상황에 적절하게 대처 할 수 있도록 다양한 규칙을 정의하여 스케줄링 기법에 적용하면 다른 모델들에 비하여 효율적으로 작업을 처리 할 수 있음을 의미한다. 또한 평균 작업 처리 시간 결과를 통해서 짧은 응답시간을 제공하는 SCDJSM이 개개의 작업에 대한 처리 및 응답 속도가 빨라지므로 결국 시스템 전반에 걸쳐 성능이 향상 된다는 것을 의미한다.

위의 실험 결과들로부터 이기종의 자원으로 구성된 컴퓨팅 환경에서 SCDJSM이 RRJSM, RAJSM, RMJSM들에 비하여 빠르고 안정적인 작업 처리를 할 수 있으며, 자원의 동적인 변화에 빠른 상황 대처 능력과 지능적인 작업 스케줄링 방법을 제공하며 시스템 전반의 성능을 향상시킬 수 있다는 것을 의미한다.

## 5. 결 론

이기종의 자원들로 이루어진 컴퓨팅 환경에서 기존의 작업 스케줄링 기법인 자원 평가에 의한 방법이나 휴리스틱에 의한 방법은 시스템간의 호환성과 확장에 한계가 있으며 다양한 자원의 동적인 상황을 반영한 효율적인 작업 스케줄링을 보장하기 어렵다. 본 논문에서는 이러한 문제들을 해결하고자 W3C에서 제정한 표준 언어인 OWL을 이용하여 이기종의 자원들을 기술하는 자원 온톨로지를 제안하였으며 시스템 확장이 용이하고 유연한 시스템 구조를 제공하였다. 또한 자원의 동적인 변화에 따른 신속한 대처가 가능하도록 여러 가지 스케줄링 기법을 지식

기반의 규칙으로 정의하여 추론을 통해서 최적의 자원에게 작업을 할당 할 수 있는 시멘틱 컴퓨팅 기반의 동적 작업 스케줄링 모델을 제안하였다.

시뮬레이션 실험 결과를 통해서 본 논문에서 제안한 작업 스케줄링 모델이 비교 모델들에 비하여 낮은 작업 손실, 높은 작업 처리율, 짧은 응답시간으로 데이터를 처리하여 향상된 결과를 제시하는 것으로 나타났다. 또한, 이기종의 자원으로 구성 된 컴퓨팅 환경에서 작업 처리 상황에 적합하게 대처 할 수 있는 많은 지식을 다양한 규칙들로 구성하여 이를 작업 스케줄링에 적용할 경우 시스템 전반의 성능을 향상 시킬 수 있다. 이외에도 자원 정보에 의미 및 관계를 부여하여 이를 추론에 이용함으로써, 사용자의 요구사항 또는 특성에 적합한 자원을 제공할 수 있으며, 또한 자원 등록 및 탐색, 서비스 관리, 오류 극복 등 분산 컴퓨팅과 관련된 다양한 분야로 확장 가능할 것으로 기대된다.

## 참 고 문 헌


1. 김학두, 김진석, 박형우. (2004), “GRID 시스템을 위한 온라인 스케쥴링 알고리즘”, 정보과학회논문지, 제31권, 제1·2호, pp. 95-101.
2. Foster, I. C. Kesselman, S. Tuecke. (2001), “The Anatomy of the Gird: Enabling Scalable Virtual Organizations”, International Journal of High Performance Computing Applications, Vol. 15, No. 3, pp. 200-222.
3. Foster, I. and C. Kesselman. (1999), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers.
4. S. H. Bokhari. (1987), Assignment Problems in Parallel and Distributed Computing, Kluwer Academic Publisher.
5. Baghban, H., Rahmani, A.M. (2008), “A Heuristic on Job Scheduling in Grid Computing Environment”, Proceedings of the 2008 Seventh International Conference on Grid and Cooperative Computing, Shenzhen, pp. 141-146.
6. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M. (2001), “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp. 810-837.
7. Buyya, R. (2002), Economic-based Distributed Resource Management and Scheduling for Grid Computing, Ph.D Thesis, Monash University, Melbourne, Australia, pp. 47-79.
8. Ibarra, O. H., Kim, C.E. (1977), “Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors”, Journal of the ACM, Vol. 24, No. 2, pp. 280-289.
9. Munir, E.U., Li, J., Shi, S., Zou, Z., Yang, D. (2008), “MaxStd: A Task Scheduling Heuristic for Heterogeneous Computing Environment”, Information Technology, Vol. 7, pp. 679-683.
10. Deborah L. McGuinness, Frank van Harmelen. “OWL Web Ontology Language Overview”, W3C Recommendation 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210/
11. Casavant, Thomas L., Kuhl, Jon G. (1988), “A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems”, IEEE Transactions on Software Engineering, Vol. 14, No. 2, pp. 141-154.
12. Maheswaran, M., Braun, T.D., Siegel, H.J. (1999), “Heterogeneous Distributed Computing”, Encyclopedia of Electrical and Electronics Engineering, J. G. Webster, editor, John Wiley & Sons, Vol. 8, pp. 679-690.
13. 박다혜, 이종식. (2006), “자원 신뢰성 측정을 통한 효율적인 그리드 자원 작업 스케줄링 모델”, 한국시뮬레이션학회 논문지, Vol. 15, No. 2, pp. 129-136.
14. Protégé available at http://protege.stanford.edu/
15. Bossam Inference Engine available at http://bossam.wordpress.com/
16. Zeigler, B.P., et al. (1997), “The DEVS Environment for High-Performance Modeling and Simulation”, IEEE Computational Science and Engineering, Vol. 4, No. 3, pp. 61-71
17. 배정민, 송영금, 김동우. (2007), “고속 이동 통신을 위한 적응 가능한 라운드 로빈 스케줄링 방식”, 정보과학회논문지, 제34권, 제1호, pp. 27-32.

**노 창 현** (cromirak@hanmail.net)

2006 수원대학교 인터넷 정보공학과 학사
2008 인하대학교 정보공학과 석사
2008～현재 인하대학교 정보공학과 박사과정

관심분야 : 그리드 컴퓨팅, 온톨로지, 시스템 모델링 및 시뮬레이션

**장 성 호** (ho7809@hanmail.net)

2004 용인대학교 컴퓨터 정보공학과 학사
2006 인하대학교 컴퓨터 정보공학과 석사
2006～현재 인하대학교 정보공학과 박사과정

관심분야 : 그리드 컴퓨팅, RFID, 웹서비스, 소프트웨어 모델링

**김 태 영** (silverwild@gmail.com)

2007 인하대학교 컴퓨터공학부 학사
2009 인하대학교 정보공학과 석사
2009~ 현재 인하대학교 정보공학과 박사과정

관심분야 : 모델링&시뮬레이션, 분산처리

**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
2003～2006 인하대학교 컴퓨터공학부 조교수
2006～현재 인하대학교 컴퓨터공학부 부교수

관심분야 : 시스템 모델링 및 시뮬레이션, 그리드 컴퓨팅