한국시뮬레이션학회 논문지
Vol. 26, No. 4, pp. 51-61 (2017. 12)
http://doi.org/10.9709/JKSS.2017.26.4.051
ISSN 1225-5904

DEVS 형식론 기반의 재겨냥성 하둡 시뮬레이션 환경 개발

김병수 † · 강봉구 · 김탁곤 · 송해상

Development of Retargetable Hadoop Simulation Environment Based on DEVS Formalism

Byeong Soo Kim† · Bong Gu Kang · Tag Gon Kim · Hae Sang Song

ABSTRACT

Hadoop platform is a representative storing and managing platform for big data. Hadoop consists of distributed computing system called MapReduce and distributed file system called HDFS. It is important to analyse the effectiveness according to the change of cluster constructions and several parameters. However, since it is hard to construct thousands of clusters and analyse the constructed system, simulation method is required to analyse the system. This paper proposes Hadoop simulator based on DEVS formalism which provides hierarchical and modular modeling. Hadoop simulator provides a retargetable experimental environment that is possible to change of various parameters, algorithms and models. It is also possible to design input models reflecting the characteristics of Hadoop applications. To maximize the user's convenience, the user interface, real-time model viewer, and input scenario editor are also provided. In this paper, we validate Hadoop Simulator through the comparison with the Hadoop execution results and perform various experiments.

Key words : Hadoop, DEVS formalism, Retargetable simulation environment

요 약

최근 빅 데이터가 증가하는 추세에 따라 이를 분석 및 처리하고 활용하는 방안에 대한 관심도 증대되고 있다. 이러한 빅 데이터를 저장, 관리하기 위한 대표적인 플랫폼으로 분산 컴퓨팅 프레임워크인 맵리듀스와 분산 파일 시스템인 HDFS로 구성된 하둡 플랫폼이 있다. 하둡은 일반적으로 수백 수천 대 이상의 클러스터로 구축되는데, 이 때 실제 클러스터 구성이나 파라미터에 따라 하둡 플랫폼이 가지는 효과도를 분석하는 것이 중요하다. 하지만 수천 대 이상의 클러스터 구축하여 이를 분석하는 것이 실질적으로 어렵기 때문에 모델링 및 시뮬레이션 기법을 통해 분석하는 것이 필요하다. 본 논문은 계층적이고 모듈러한 모델링이 가능한 DEVS 형식론을 기반으로 하둡 시뮬레이션 환경을 제안한다. 제안하는 시뮬레이션 환경은 하둡 실행 결과를 이용한 입력 모델 설계를 통해 어플리케이션의 특성을 잘 반영할 수 있으며, 파라미터/알고리즘/모델들을 다양하게 변경하여 실험할 수 있는 재겨냥성 환경을 제공한다. 또한 사용자 편의성의 극대화를 위해 사용자 인터페이스, 실시간 모델 뷰어, 입력 시나리오 편집기를 제공한다. 본 논문에서는 어플리케이션 실행 결과와의 비교를 통해 하둡 시뮬레이터를 검증하고, 다양한 파라미터에 대한 실험을 진행한다.

주요어 : 하둡, DEVS 형식론, 재겨냥성 시뮬레이션 환경

1. 서론

오늘날 디지털 경제의 확산으로 다양한 종류와 방대한 양의 정보와 데이터가 생산되는 빅 데이터(Big Data) 시대가 도래했다. 이와 함께 이러한 빅 데이터를 분석 및 처리하고 활용하는 방안에 대한 다양한 연구가 진행되고 있으며, 빅 데이터로부터 가치 있는 정보를 추출하여 혁신, 경쟁력 강화, 생산성 향상 등에 활용하고자 하는 연구에 대한 관심이 역시 증대되고 있다. 이에 따라 대규모 컴퓨팅을 가능하게 하고 폭증하는 데이터를 효율적으로 저장, 관리하기 위한 분산 컴퓨팅/스토리지 플랫폼에 대

Received: 14 December 2017, **Revised**: 18 December 2017, **Accepted**: 18 December 2017

† **Corresponding Author**: Byeong Soo Kim
E-mail: kevinzzang@kaist.ac.kr
Dept. of Electrical Engineering, KAIST, Daejeon, Korea

한 요구가 커지고 있다(Hashem, 2015). 이러한 빅 데이터를 저장, 관리하기 위한 대표적인 플랫폼 중의 하나로 하둡(Hadoop)이 있다(Apache, 2017). 하둡은 분산 컴퓨팅을 위한 맵리듀스(MapReduce) 프레임워크와 분산 파일 시스템인 HDFS(Hadoop Distributed File System)로 구성되며, 통상 수백 수천 대 이상의 실제 컴퓨터 클러스터들로 구축된다. 이러한 하둡 플랫폼의 성능 예측 및 유지 관리, 하드웨어 호환성 등을 분석하기 위해서, 실제 클러스터들의 하드웨어 사양이나 맵리듀스 및 HDFS 내부의 여러 가지 파라미터 및 알고리즘 변화에 따라 하둡 플랫폼을 실행하고 분석하는 것이 요구된다. 하지만 실제로 수천 대 이상의 클러스터를 구축한 후에 이를 분석하는 것이 경제적, 그리고 공간상의 제약 등 쉽지 않기 때문에 모델링 및 시뮬레이션 기법을 통해 이를 해결할 수 있다(Kim, 2017).

기존에도 하둡 시뮬레이션에 대한 여러 연구들이 진행되어왔다. 대표적인 하둡 시뮬레이터로 MRPerf(Wang et al., 2009), MRSim(Hammoud et al., 2010), HSim(Liu et al., 2013) 등이 있다. 이들 연구는 맵리듀스의 내부 구조 및 프로세스에 대해 구체적인 모의를 하여 정확한 시뮬레이션 결과를 보여주지만, HDFS에 대한 구체적인 모의를 하지 않고 맵리듀스의 성능 분석에만 초점을 맞추고 있다. 또한 하둡 플랫폼의 성능 모델링에 대한 연구들이 진행 되어 왔는데(Wu et al., 2015; Kim and Kim, 2017), 이들은 하둡에 대한 전반적인 성능 모델 분석을 진행하였지만, 여러 목적에 따라 파라미터, 알고리즘, 모델들을 쉽게 교체할 수 있는 재겨냥성 환경을 제공하지 않는다.

따라서 본 논문은 DEVS(Discrete EVent system Specification) 형식론 기반의 재겨냥성 하둡 시뮬레이션 환경을 개발한다. 제안하는 환경은 맵리듀스와 HDFS를 포함한 하둡 플랫폼의 구성 요소들이 계층적이고 모듈러하게 구현되어 있으며 다양한 파라미터, 알고리즘, 모델들을 입력으로 사용 할 수 있다. 또한 실제 하둡 플랫폼을 이용한 실험 결과를 통해 어플리케이션의 특성을 반영하는 모델을 설계하여 하둡 시뮬레이터의 입력 모델로 사용함으로써 어플리케이션의 특징을 더욱 잘 반영할 수 있다. 사용자 인터페이스와 입력 시나리오 편집기, 실시간 모델 뷰어 등을 제공함으로써 사용자 편의성 또한 높일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 하둡에 대해 알아보며, 3장에서는 제안하는 재겨냥성 하둡 시뮬레이션 환경에 대해 설명한다. 4장에서는 제안하는 환경을 검증하고 실제 다양한 실험 디자인에 적용한 결과를 보여주며, 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해 정리한다.

2. 배경 지식

2.1 하둡

하둡은 분산 컴퓨팅/스토리지 환경으로 구글의 것을 바탕으로 Apache에서 개발한 것이다(Apache, 2017). 오픈 소스로 공개되어 사용과 접근이 용이하고 확장성과 성능이 뛰어나, 현재 수많은 기업들뿐만 아니라 학교와 연구기관 등에서 범용적으로 사용되고 있다. 일반적으로 수 백, 수 천대 규모의 클러스터로 구성되며 각각의 노드들은 상용 하드웨어를 사용함으로써 빈번하게 발생하는 노드들의 고장 상황에서도 데이터 손실이 발생하지 않도록 내고장성을 지닌다. 이러한 하둡은 기본적으로 분산 컴퓨팅 시스템인 맵리듀스(MapReduce)와 분산 파일 시스템인 HDFS(Hadoop Distributed File System)로 구성된다.

2.1.1 맵리듀스

하둡의 분산 컴퓨팅 프레임워크인 맵리듀스는 대용량의 데이터를 (키, 값) 쌍을 통하여 처리하는 프로그래밍 모델이다(Dean and Ghemawat, 2008). 이러한 맵리듀스 프레임워크는 마스터인 하나의 잡트래커(JobTracker)와 슬레이브인 다수의 태스크트래커(TaskTracker)로 구성된다. 맵리듀스는 응용 프로그램을 잡(Job)이라는 작업 단위로 관리 및 처리하는데, 잡트래커는 잡을 다수의 맵 태스크(Map Task)와 리듀스 태스크(Reduce Task)로 분할하여 개별 태스크트래커에 분배하고, 전체 작업을 스케줄링 한다. 그리고 태스크트래커는 잡트래커가 할당한 태스크들을 처리함으로써 응용 프로그램을 수행한다.

이러한 구조를 바탕으로 맵리듀스 응용 프로그램은 다음과 같이 크게 input splitting, mapping, sorting, shuffling, merging, reducing의 과정으로 진행된다.

* Input splitting: 응용 프로그램의 입력 데이터를 다수의 독립적인 블록(스플릿)으로 분할하여 각각의 블록에 대하여 맵 태스크를 생성하고, 태스크트래커에 각각의 맵 태스크를 할당
* Mapping: 정의된 맵 함수에 따라 입력 데이터를 처리
* Sorting: 리듀스 태스크의 개수에 따라 맵의 출력을 파티션을 생성하여 정렬

* Shuffling: 정렬된 맵의 출력을 파티션에 따라 그에 맞는 리듀스로 전송
* Merging: 각각의 맵 태스크로부터 전송된 맵의 출력을 통합
* Reducing: 통합된 맵 태스크의 출력을 최종적으로 처리하여 애플리케이션의 최종 결과를 출력

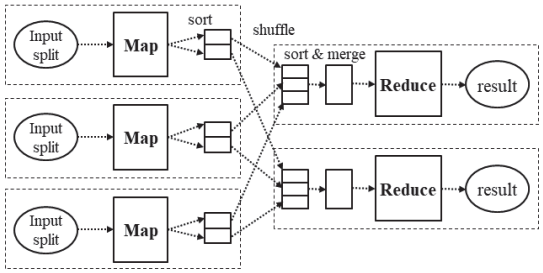


Fig. 1. Process of MapReduce

맵리듀스는 Figure 1과 같이 잡트래커가 응용 프로그램의 입력 데이터를 다수의 스플릿으로 분할하고, 이를 맵 태스크에 분배함으로 진행된다. 맵 태스크는 정의된 맵 함수에 따라 입력 데이터를 처리한다. 개별 맵 태스크에서 처리된 결과는 리듀스 태스크의 개수에 따라 다수의 파티션으로 구분되어 정렬되는데, 각각의 파티션은 셔플링(Shuffling) 과정을 통하여 리듀스 태스크의 입력 데이터로 전송된다. 전송된 개별 맵 태스크의 출력들은 리듀스 태스크에서 처리할 수 있도록 리듀스 태스크의 입력 데이터로 통합되는 과정을 거친다. 이렇게 하나로 통합된 맵 태스크의 출력을 입력으로 하여 리듀스 태스크는 정의된 리듀스 함수에 따라 입력 데이터를 처리하고, 맵리듀스 응용 프로그램의 최종 결과를 출력한다.

2.1.2 HDFS

하둡의 분산 파일 시스템인 HDFS는 Figure 2와 같이 마스터 노드인 네임 노드와 슬레이브 노드인 다수의 데이터 노드들로 구성된다(Shvachko et al., 2010). 데이터는 기본적으로 64MB 크기의 블록 단위(Chunk)로 데이터 노드들에 분산되어 저장되며, 개별 데이터 노드에 저장된 모든 블록들의 메타 데이터와 디렉터리 정보가 네임 노드의 네임 스페이스에 저장 및 관리된다. HDFS는 데이터 노드의 고장 상황에 대비하여 시스템의 내고장성을 제공하기 위하여 각각의 블록들을 기본적으로 추가적인 2개의 복제 블록을 생성하여 다른 데이터 노드들에게 저장하는 방식을 취한다. 이러한 방식은 데이터 저장 시에는 하나의 블록에 대하여 3개의 쓰기 작업을 처리해야 함으로써, 쓰기 효율이 떨어지는 단점이 존재하지만, 읽기 작업 시에는 여러 복제 블록으로 인하여 읽기 효율이 증대되는 장점을 지닌다.

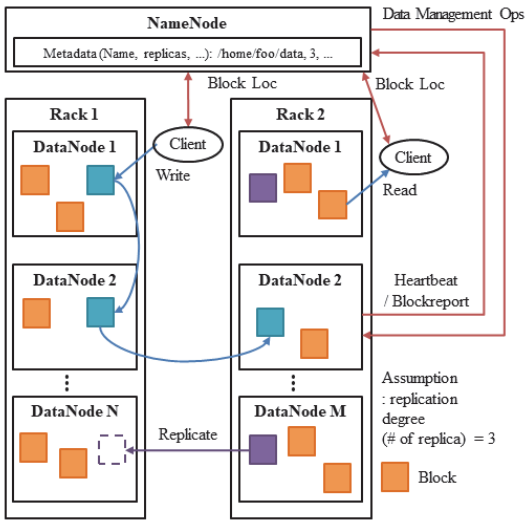


Fig. 2. Structure of HDFS

2.2 DEVS 형식론

DEVS 형식론은 집합론에 근거한 형식론으로 원자 모델(Atomic Model)과 결합 모델(Coupled Model)을 통하여 전체 시스템을 모델링할 수 있는 모델링 도구이다(Zeigler et al., 2001). 원자 모델은 더 이상 분해 할 수 없는 단위로 3개의 집합과 4개의 함수를 통하여 시스템의 구성 요소의 기본적인 동작을 표현하며, 결합 모델은 부 모델들의 결합체로 3개의 집합과 3개의 관계, 하나의 함수를 통해 다수의 원자 모델 혹은 또 다른 결합 모델을 합쳐 새로운 모델을 구성한다. DEVS 형식론은 시스템을 계층적으로 모듈화 하여 구조적 모델링이 가능하기 때문에 계층적으로 복잡한 구조로 이루어져 있는 복잡한 하둡 시스템을 모델링하기에 가장 적합한 모델링 방법론으로, 이에 본 논문에서는 DEVS 형식론을 이용하여 하둡 플랫폼 모델을 개발한다.

3. 재겨냥성 하둡 시뮬레이션 환경 개발

3.1 하둡 모델 설계

3.1.1 OAI 매트릭스를 통한 입출력 관계 식별

Table 1. Example of OAI matrix

Object		Attribute Variable	Analysis Index		
			Throughput	Latency	Storage Util.
S/W	MapReduce	# of mappers	O		
		# of reducers	O		
		Job scheduling algorithm	O	O	O
		Heartbeat period			
	HDFS	Data management policy	O		O
		Chunk size	O		
		Heartbeat period			
		Blockreport period	O	O	
H/W	Platform	# of nodes			O
		# of task slots		O	
		CPU clock speed		O	
		# of CPU cores		O	
	Storage	Storage type		O	
		Storage access speed		O	
	Network	Bandwidth	O	O	
		Topology	O	O	

하둡 플랫폼 분석을 위해서는 먼저 분석 목적에 따라 성능 지수와 그 성능 지수에 영향을 주는 모델의 파라미터 및 변수들을 식별할 필요가 있다. 본 논문에서는 입출력 사이의 관계를 식별하기 위해 OAI(Object-Analysis Index) 매트릭스를 사용한다(Kim and Sung, 2007). OAI 매트릭스의 열은 측정하고자하는 성능지수에, 행은 성능지수 측정에 필요한 객체 및 파라미터를 의미한다. 성능지수에는 처리량, 데이터 트래픽, 처리 시간, 스토리지 사용량 등이 있으며 객체는 크게 소프트웨어와 하드웨어로 구분된다. 소프트웨어 객체는 맵리듀스 모델과 HDFS 모델로 구성되며 하드웨어 객체는 플랫폼 모델, 스토리지 모델, 네트워크 모델이 구성된다. 하둡 플랫폼 시뮬레이션을 위한 OAI 매트릭스의 예는 Table 1과 같다. 본 논문에서는 식별한 파라미터들을 고려하여 하둡 시스템을 분석하고 모델링한다.

3.1.2 구체적인 모델 설계

본 절에서는 앞 절에서 식별한 객체와 파라미터, 분석지수를 토대로 하둡 플랫폼의 구체적인 모델을 설계한다. 하둡 플랫폼을 모델링하기 위해서는 Figure 3과 같이 플랫폼 전체를 각각의 프로세스와 기능들로 나누어 분석하고 식별하며, 이때 분석 목적에 따라서 각 프로세스나 기능들의 추상화 수준을 결정한다. 모델링에 반영할 프로세스(Process, P)는 다음과 같은 순서를 따른다.

P1. LFS(Local File System)에서 HDFS에 필요한 입력 파일들을 복사한다.

P2. HDFS로부터 해당 맵퍼(Mapper)에 할당된 블록

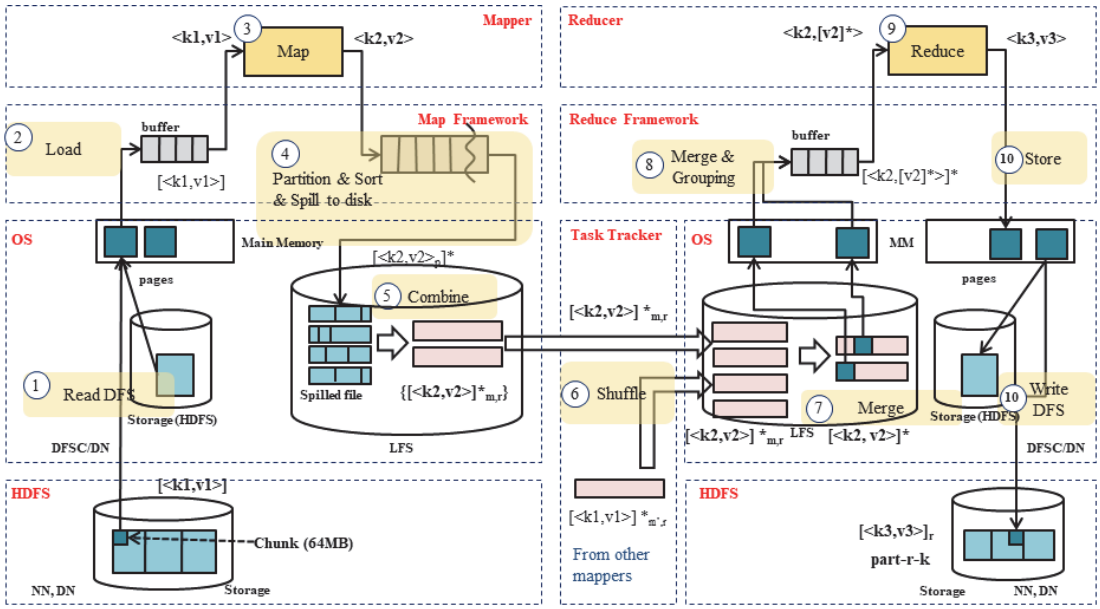


Fig. 3. Analysis of Hadoop process for modeling

을 읽어드린다.
P3. 읽어드린 입력 레코드를 맵퍼에서 처리한다.
P4. 맵퍼의 출력 레코드를 파티셔닝(Partitioning)을 하고 파티션 및 키에 대해 정렬하여 버퍼에 저장한다.
P5. 레코드들을 파티션 번호에 따라 병합한다.
P6. 키에 따라서 맵퍼에서 각각의 리듀서(Reducer)로 파티션된 결과를 전송해주는 셔플링이 이루어진다.
P7. 맵퍼로부터 가져온 여러 개의 파티션들을 병합정렬 하여 저장한다.
P8. 레코드를 리듀서로 읽어드린다.
P9. 읽어드린 레코드를 리듀서에서 처리한다.
P10. 출력 레코드를 버퍼링하고 HDFS에 저장한다.

또한 식별된 내용에 따라 다음과 같은 기능(Function, F)들을 반영하여 모델링한다.

F1. 노드를 자유롭게 추가 및 변경할 수 있다.
F2. 맵퍼 또는 리듀서가 노드에 분산되어 수행된다.
F3. 노드는 하나 이상의 맵퍼 또는 리듀서를 가진다.
F4. 스토리지가 각 노드에 분산되어 있다.
F5. 네임 노드와 네임 스페이스를 통해 실제와 동일한 프로세스로 입력 청크(이름, 크기, 위치)를 관리한다.
F6. 어플리케이션 모델을 반영하여 맵리듀스 컴퓨팅을 진행한다.
P7. 재겨냥성을 고려하여 파라미터, 알고리즘, 모델 등을 쉽게 변경할 수 있도록 모듈러하게 설계한다. 이에 대한 자세한 내용은 다음 절에서 설명한다.

마지막으로 하둡 플랫폼을 모델링할 때의 가정 사항(Assumption, A)은 다음과 같다.

A1. 단위 랙의 네트워크 스위치 속도는 충분히 빨라 전체 성능에 영향을 거의 미치지 않는다고 가정한다.
A2. CPU의 속도는 충분히 빠르다고 가정한다. 멀티코어로 인한 주 메모리 동기화 문제는 발생하지 않으며 캐쉬 메모리가 충분히 커서 각 코어가 독립적으로 동작한다고 가정한다.
A3. 빅 데이터를 직접 저장하거나 실제 맵리듀스에서 (키, 값) 쌍을 처리하지 않으며 어플리케이션 모델을 바탕으로 데이터 위치 및 용량, 처리 속도를 모델에 반영한다. 추후 실제 처리 가능성을 위해 협력객체를 통한 확장성을 고려하여 설계한다.

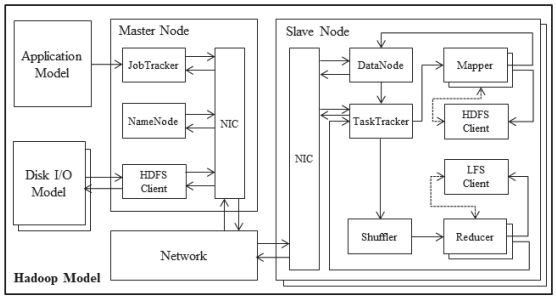


Fig. 4. Overall structure of Hadoop model

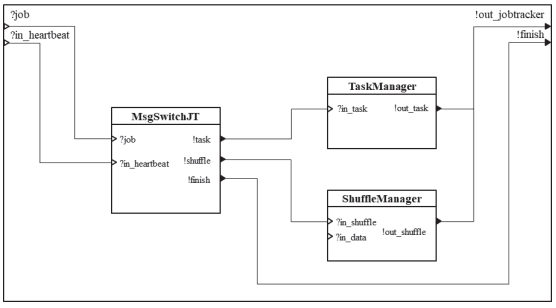


Fig. 5. Example of coupled model: jobtracker model

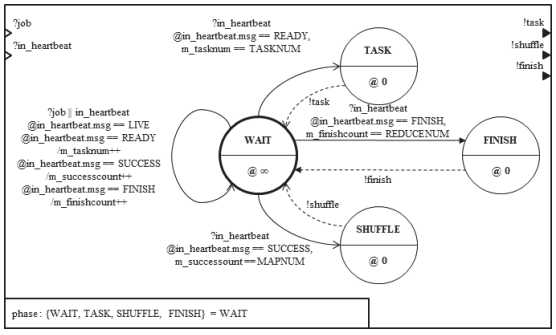


Fig. 6. Example of atomic model: heartbeat model

앞서 언급한 프로세스와 기능, 가정 사항을 바탕으로 하둡을 DEVS 형식론에 따라 모델링을 진행하며, 그 전체 구조는 Figure 4와 같다. 또한 Figure 5와 Figure 6은 DEVS 형식론에 따라 모델링된 구체적인 결합 모델과 원자 모델의 예를 보여준다.

3.2 실행 기반의 어플리케이션 모델 설계

하둡은 플랫폼에 어플리케이션을 구현하여 실행하는 구조이기 때문에, 하둡 플랫폼을 모델뿐만 아니라, 하둡에서 동작하는 어플리케이션에 대한 모델링이 필요하다. 시뮬레이션을 할 때 실제 어플리케이션을 구동시키는 것

이 아닌 그 어플리케이션의 특징을 반영하는 어플리케이션 모델을 설계하고 그 모델을 시뮬레이터에 입력으로 사용하여 시뮬레이션을 하게 된다. 어플리케이션 모델은 실제 하둡 플랫폼의 실행을 통해 나온 처리 시간, 처리량 등의 로그 값 분석을 통해 필요한 파라미터를 추출하여 설계한다. 즉, 본 논문에서는 WordCount와 TeraSort 어플리케이션을 16대의 노드로 구축된 실제 하둡 플랫폼을 가지고 실행하여 그 결과를 통해 하둡 시뮬레이션에 필요한 파라미터들을 추출하고 실행 기반의 어플리케이션 모델을 설계한다. WordCount는 주어진 입력 데이터 셋에서 각 단어의 빈도수를 카운트 하는 가장 기본적인 하둡 어플리케이션이며, TeraSort는 입력 값들을 정렬하는 어플리케이션이다.

Table 2. Execution of Hadoop for application modeling

Parameter	Value
Application	WordCount, TeraSort, TestDFSIO
Chunk size (MB)	64
Input data size (GB)	0.05, 0.1, 0.5, 1, 2, 4, 8
# of nodes	1, 2, 4, 8, 15

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	SLOTS_MILLIS_MAPS	0	0	41,344
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	2
	Data-local map tasks	0	0	2
	SLOTS_MILLIS_REDUCES	0	0	18,447
File Input Format Counters	Bytes Read	100,000	0	100,000
File Output Format Counters	Bytes Written	0	100,000	100,000
FileSystemCounters	FILE_BYTES_READ	258	102,006	102,264
	HDFS_BYTES_READ	100,184	0	100,184
	FILE_BYTES_WRITTEN	146,402	124,170	270,572
	HDFS_BYTES_WRITTEN	0	100,000	100,000
Map-Reduce Framework	Map output materialized bytes	102,012	0	102,012
	Map input records	1,000	0	1,000
	Reduce shuffle bytes	0	102,012	102,012
	Spilled Records	1,000	1,000	2,000
	Map output bytes	100,000	0	100,000
	Total committed heap usage (bytes)	230,957,056	15,859,712	246,816,768
	CPU time spent (ms)	1,870	3,140	5,010
	Map input bytes	100,000	0	100,000
	SPLIT_RAW_BYTES	184	0	184
	Combine input records	0	0	0
	Reduce input records	0	1,000	1,000
	Reduce input groups	0	1,000	1,000
	Combine output records	0	0	0
	Physical memory (bytes) snapshot	319,561,728	65,662,976	385,224,704
	Reduce output records	0	1,000	1,000
	Virtual memory (bytes) snapshot	2,127,224,832	1,070,981,120	3,198,205,952
	Map output records	1,000	0	1,000

Fig. 7. Execution result of Hadoop application

Table 2는 어플리케이션 모델링을 위해 실제 하둡 플랫폼 실행에 대한 실험 디자인이다. WordCount와 TeraSort 각각에 대해 청크 크기와 입력 크기, 그리고 노드 수를 변화시켜가며 하둡 플랫폼을 실행하여 어플리케이션 각각의 특징을 반영하고 있는 결과 데이터를 수집할 수 있다. 또한 파일시스템의 I/O를 측정하는 벤치마크인 TestDFSIO를 이용하여 동일 실험 디자인에 대해 하둡 플랫폼을 실행함으로 파일 I/O에 대한 결과를 획득할 수 있다.

각 어플리케이션을 이용한 실제 실행 결과는 Figure 7과 같이 획득되며, 결과 데이터들 중에서 모델링 목적에 맞게 필요한 파라미터를 추출하여 통계적으로 모델링 할 수 있다. Table 3은 실행 결과를 통해 하드웨어의 특성을 반영하는 내부 파라미터에 대한 결과이며, Table 4는 각 어플리케이션의 특징을 반영하는 입력 모델에 필요한 파라미터를 나타낸다. 본 논문에서는 각 실험 디자인을 한 번씩 실행하여 확정적인 값을 가지지만, 어플리케이션 특징들의 더욱 정확한 반영을 위해 각 실험 디자인 마다 반복 실험을 통해 통계적인 값을 구함으로 신뢰도를 높일 수 있다.

Table 3. Value of internal parameters

Type	Parameter	Value
MapReduce	Shuffle time per node (sec/MB)	0.064
HDFS	Write (MB/sec)	24
	Read (MB/sec)	67

Table 4. Value of application parameters

Type	Parameter	Value	
		WordCount	TeraSort
Mapper	MapperSizeRatio	1.58	2
	MapperProcTime	0.24	0.051
Reducer	SortSizeRatio	0.0000534	0.49
	SortProcTime	0	0.0015
	ReducerSizeRatio	0.0089	1
	ReducerProcTime	5.52	0.026

3.3 하둡 시뮬레이션 환경 구현

본 절에서는 앞서 DEVS 형식론으로 모델링 한 하둡 모델을 구현하고 이에 필요한 추가적인 환경을 구현한다. 하둡 시뮬레이션 환경은 Figure 8과 같이 크게 하둡 모델, 하둡 사용자 인터페이스, 입력 시나리오 편집기와 실시간 모델 뷰어로 구성된다. 하둡 모델은 앞 절에서 하둡의 추상화를 통해 개발된 모델들로, DEVS 형식론을 사용하여 이산사건 시스템을 객체 지향적으로 모델링하고 시뮬레이션 할 수 있는 환경인 DEVSim++를 이용하여 구현 및 실행 가능하다(Kim et al., 2011). 또한 가시화를 위한 도구인 하둡 사용자 인터페이스는 시뮬레이션 제어 및 시나리오 선택, 그리고 실시간으로 시뮬레이션 결과 출력 등이 가능하다.

다음으로 입력 시나리오 편집기는 입력 시나리오 작성 및 편집 기능을 제공한다. 하둡 플랫폼 실험 결과를 통해

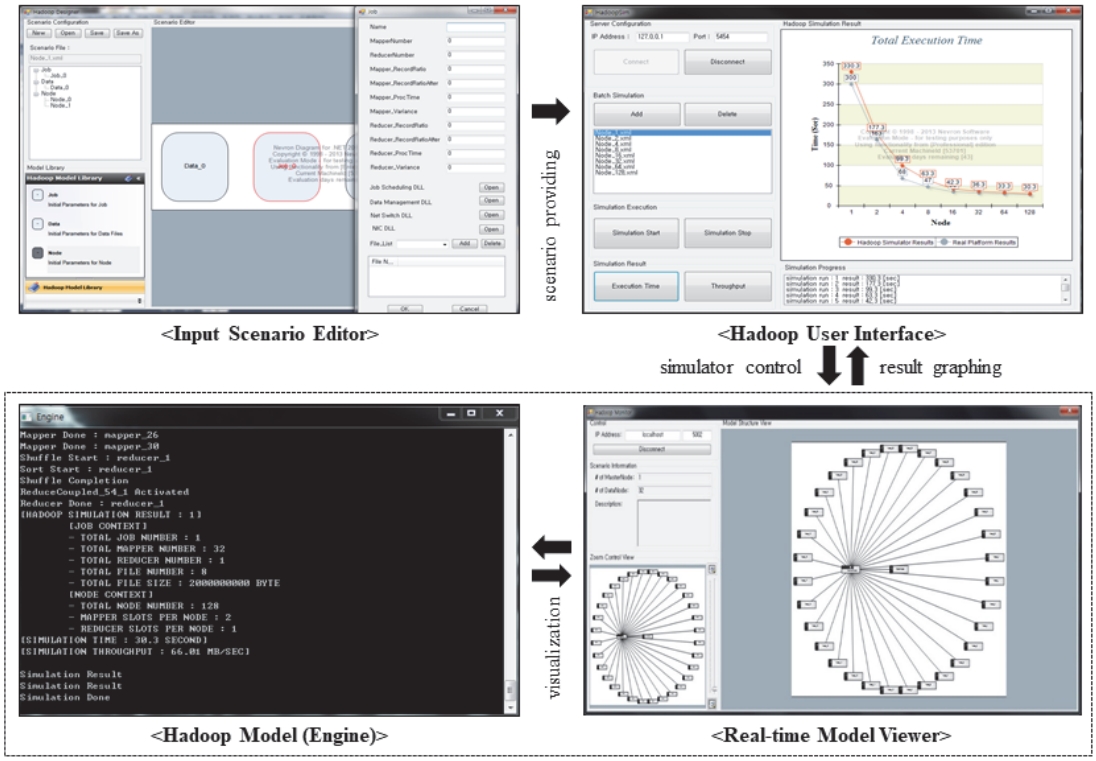


Fig. 8. Hadoop simulation Environment

만들어진 어플리케이션 모델과 노드 정보 및 입력 데이터 셋에 대한 정보 등을 입력하여 사용자가 손쉽게 시나리오를 작성하고 이를 통해 시뮬레이션이 가능하다. 즉, 실험 목적에 따라 시뮬레이터 요구 사항이 바뀔 때 마다 파라미터, 알고리즘, 모델 교체가 용이한 재겨냥성 하둡 시뮬레이션 환경을 제공해준다.

마지막으로 실시간 모델 뷰어는 하둡 시뮬레이터가 실행될 때 모델 구조를 실시간으로 도시해 준다. 전체 노드 구성에 대한 도시와 각 노드 내부의 모델과 내부 모델 간의 연결 관계, 그리고 모델의 입출력 포트 등을 도시해주는 기능을 한다. 이와 같은 각각의 시뮬레이션 환경을 이용하여 사용자는 시나리오를 만들고, 하둡을 시뮬레이션 하며, 그 과정과 결과를 분석할 수 있는 사용자 친화적인 환경을 제공받게 된다.

4. 실험

본 장에서는 앞에서 만들어진 하둡 모델을 실증하고, 완성된 재겨냥성 하둡 시뮬레이션 환경을 이용하여 확장성에 대한 실험, 청크 크기 변화에 따른 실험, 알고리즘 교체에 따른 실험 등을 수행하여 본 환경이 가지는 이점에 대해 보여준다.

4.1 하둡 모델 실증

제안하는 환경을 사용하기에 앞서 먼저 모델 실증의 과정이 필요하다. 모델 실증은 설계 된 모델이 실제 시스템을 제대로 모의하고 있는지 확인하는 과정으로 실제 하둡 플랫폼의 실행 결과와 시뮬레이션 결과 비교를 통해 이루어진다. 본 논문에서는 WordCount, TeraSort 두 가지의 기본 어플리케이션에 대하여 앞 장의 Table 2에서 설계된 시나리오를 이용하여 하였다. 이 때 하둡 시뮬레이션에 필요한 어플리케이션 모델은 3.2절에서 만들어진 모델이 사용 된다. 또한 실행 결과와 시뮬레이션 결과의 비교를 위해 사용되는 결과 값으로는 실험 목적에 따라 다양한 파라미터들이 있을 수 있으며, 본 논문에서는 잡 수행 시간과 처리량에 대한 결과를 비교한다.

Figure 9와 Figure 10은 각각 WordCount 어플리케이션에 대해서 노드수를 증가 시켰을 때와 TeraSort 어플리

케이션에 대해 파일 사이즈를 변화시켰을 때의 실제 플랫폼의 잡 수행 시간과 시뮬레이션의 잡 수행 시간에 대한 결과 비교 그래프이다. 그래프에서 볼 수 있듯이 두 어플리케이션에 대해서 매우 정확한 예측 결과를 볼 수 있으며, 예측 값과 실제 값의 차이를 보여주는데 널리 사용되는 지표인 평균 제곱근 편차(Root Mean Square Error, RMSE)도 각각 24.3과 49.9로 수행 시간 대비 낮은 값을 가짐을 볼 수 있다.

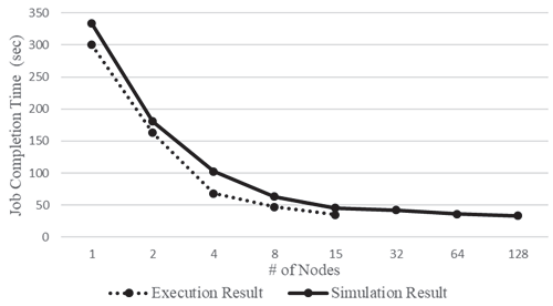


Fig. 9. Validation result: WordCount

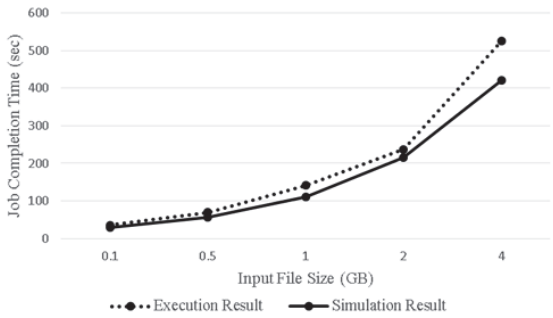


Fig. 10. Validation result: TeraSort

4.2 실험 설계

앞서 검증한 하둡 시뮬레이터의 효용성을 보이기 위하여 본 논문에서는 Table 5와 같이 실험을 설계하였다. 첫 번째는 확장성에 대한 실험으로 노드 수와 입력 데이터의 크기 증가에 따른 잡 수행 시간 및 처리량을 예측하여 하둡 시뮬레이션 환경이 많은 노드 수나 입력 데이터가 증가함에도 시뮬레이션 가능한 것을 보여준다. 두 번째는 재겨냥성에 대한 실험으로 목적에 따라 파라미터, 알고리즘, 모델 등을 바꿔가며 실험 할 수 있음을 보여준다. 본 논문에서는 파라미터와 알고리즘을 변경해가며 실험하며, 파라미터로는 HDFS 내부 파라미터인 청크 사이즈 변경에 따른 성능 비교를 실험한다. 또한 알고리즘으로는 잡 스케줄링 알고리즘을 변경해가며 알고리즘 성능 비교 실험을 진행한다. 이를 위하여 두 가지 잡 스케줄링 알고리즘을 사용하여 각 알고리즘에 대하여 노드 수를 증가시켜가며 전체 실행 시간과 처리량을 측정하는 실험을 수행하였다.

Table 5. Experimental design for Hadoop simulation

Experiment	Parameter	Value
Experiment 1 : Scalability	Input data size	2~64GB
	# of nodes	4~1024
Experiment 2 : Chunk size (Parameter)	Chunk size	32~128MB
	# of nodes	1~256
Experiment 3 : Job scheduling algorithm	Algorithm types	Fair scheduler, modified fair scheduler
	# of nodes	1~32

이때 하둡 시뮬레이션 환경에 사용된 나머지 파라미터들 값들은 Table 6과 같으며, 실험을 위한 파라미터 외에는 고정된 기본 값들을 사용한다.

Table 6. Experimental design for Hadoop simulation

Parameter	Value
Application	Experiment 1 (TeraSort), Experiment 2, 3 (WordCount)
# of mappers / reducers	32 / 1
# of files	8
# of replications	3
Input data size	2GB
# of nodes	4

4.3 실험 결과

앞 절에서 설계한 실험에 대한 결과는 다음과 같다. 먼저 Figure 11과 Figure 12는 확장성에 대한 실험 결과로 노드 수와 입력 데이터 크기를 증가시킴에 따라 잡 처리 시간 및 처리량이 각각 어떻게 변화하는지 확인 할 수 있다. 제안된 하둡 시뮬레이션 환경을 통하여 천 대 규모 이상의 하둡 클러스터를 시뮬레이션이 가능함을 확인 할 수 있으며, 추후 더 큰 규모의 클러스터나 더 많은 데이터를 이용하여 실험 가능하다.

다음으로는 파라미터 및 알고리즘 변경에 대한 실험 결과로, 먼저 Figure 13은 청크 사이즈에 따른 잡 처리 시간 변화에 대해 보여준다. 청크 사이즈가 커질수록 잡 처리 시간이 소폭 감소하는 것을 알 수 있다. Figure 14는 잡 알고리즘 변경에 따른 처리량 비교 그래프로, 잡

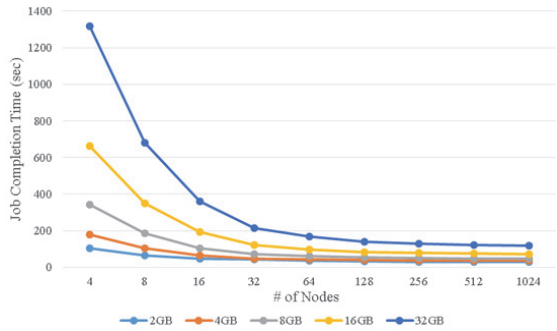


Fig. 11. Experimental result 1: job completion time

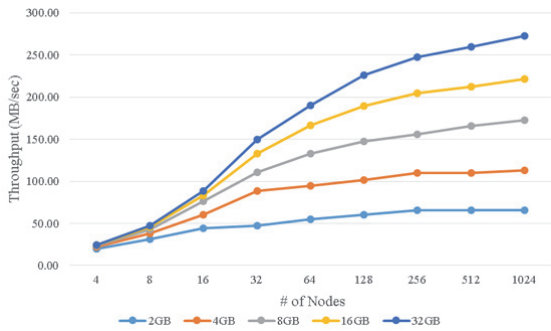


Fig. 12. Experimental result 1: throughput

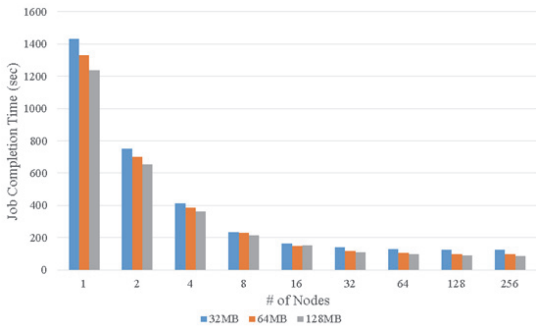


Fig. 13. Experimental result 2: job completion time

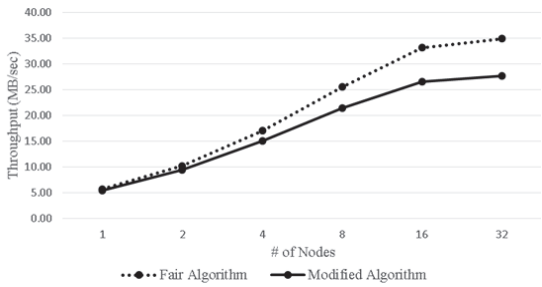


Fig. 14. Experimental result 3: throughput

알고리즘의 종류에 따른 하둡 플랫폼의 성능 변화, 즉 처리량에 미치는 영향을 알 수 있다. 노드의 수가 작을 때는 두 알고리즘의 성능차가 미미하지만, 노드의 수가 많을 때는 기존 Fair 스케줄링 알고리즘이 처리량에서 높은 성능을 가짐을 확인할 수 있다.

위와 같은 실험들을 통해 제안된 재겨냥성 하둡 시뮬레이션 환경을 사용하여 사용자의 목적에 따라 다양한 파라미터, 알고리즘, 모델 등을 변경해가며 분석 가능함을 알 수 있다. 본 논문에서는 WordCount, TeraSort의 두 어플리케이션만을 이용한 실험을 수행하였지만, 이 외에도 사용자가 구현한 다양한 어플리케이션도 제안하는 환경 및 방법을 이용하여 분석 가능하다.

5. 결론

본 논문에서는 하둡 플랫폼 분석을 위해서 DEVS 형식론을 이용한 재겨냥성 하둡 시뮬레이션 환경을 개발하였다. 이를 위해 먼저 하둡을 구성하는 맵리듀스 및 HDFS와 하드웨어 플랫폼을 분석하고 이를 DEVS 형식론을 이용하여 모델링한 후 구현한다. 개발된 시뮬레이션 환경은 Pre-execution을 통한 어플리케이션 모델 설계를 통해 어플리케이션의 특징을 반영할 수 있으며, 사용자의 배경 지식에 따라 파라미터, 알고리즘, 모델 교체가 가능하다. 즉, 개발된 시뮬레이션 환경을 소스 코드 변경 없이 사용자가 여러 형태의 시뮬레이터로 합성 할 수 있는 재겨냥적인 시뮬레이션 환경을 제공한다. 또한 사용자 편의성을 위해 사용자 인터페이스와 입력 시나리오 편집기, 실시간 모델 뷰어 등을 제공한다. 개발된 환경을 실제 16대 노드로 구축된 하둡 플랫폼을 이용하여 실증하고, 세 가지 실험을 통해 다양한 상황에서 시뮬레이션이 가능함을 보여준다.

추후 연구로 미반영된 하드웨어 모델들을 반영하여 사용자가 더욱 폭넓은 시뮬레이션이 가능하도록 하며, 시뮬레이션 성능 향상을 통해 대규모 노드 실험 시에 걸리는 시간 및 자원을 줄이는 연구를 진행 할 예정이다.

References

Apache Hadoop, http://hadoop.apache.org (last accessed: 12.12.17.)

Dean, J. and S. Ghemawat (2008) "MapReduce: Simplified data processing on large clusters", *Communications of the ACM*, 51(1), 107-113.

Hammoud, S., M. Li, Y. Liu, N.K. Alham and Z. Liu (2010) "MRSim: A discrete event based MapReduce simulator", *In Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on,* 2993-2997.

Hashem, I.A.T., I. Yaqoob, N.B. Anuar, et al. (2015) "The rise of Big Data on cloud computing: Review and open research issues", Information Systems, 47, 98-115.

Kim, B.S. and T.G. Kim (2017) "Cooperation between data modeling and simulation modeling for performance analysis of Hadoop", In Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2017 International Symposium on, 1-7.

Kim, T.G. (2017) http://sim.kaist.ac.kr/course/EE612/, Lecture Note on EE612: Discrete Event Systems Modeling and Simulation, School of Electrical Engineering, KAIST

Kim, T.G. and C.H. Sung (2007) "Objective-driven DEVS modeling Using OPI matrix for performance evaluation of discrete event systems", In Proceedings of the 2007 Summer Computer Simulation, San Diego, CA, USA, Jul.

Kim, T.G., C.H. Sung, S.Y. Hong, et al. (2011) "DEVSim++ toolset for defense modeling and simulation and interoperation", *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,* 8(3), 129-142.

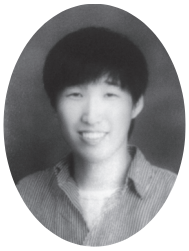
Liu, Y., M. Li, N.K. Alham and S. Hammoud (2013) "HSim: a MapReduce simulator in enabling cloud computing", *Future Generation Computer Systems,* 29(1), 300-308.

Shvachko, K., H. Kuang, S. Radia and R. Chansler (2010) "The Hadoop Distributed File System", *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies,* Incline Village, USA, May.

Wang, G., A.R. Butt, P. Pandey and K. Gupta (2009) "Using realistic simulation for performance analysis of mapreduce setups", *Proceedings of the 1st ACM workshop on Large-Scale system and application performance.*

Wu, X., Y. Liu and I. Gorton (2015) “Exploring performance models of Hadoop applications on cloud architecture”, *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures,* 93-101.

Zeigler, B.P., H. Praehofer, and T.G. Kim (2001) *Theory of modeling and simulation, 2nd ed.,* ACADEMIC PRESS.



김 병 수 (kevinzzang@kaist.ac.kr)

2010 KAIST 전기 및 전자공학과 학사
2012 KAIST 전기 및 전자공학과 석사
2012~ 현재 KAIST 전기 및 전자공학과 박사과정

관심분야 : 모델링 및 시뮬레이션, 빅 데이터, 하둡, 기계 학습

강 봉 구 (kbgmode@kaist.ac.kr)

2011 한양대학교 전자통신컴퓨터공학부 학사
2013 KAIST 전기 및 전자공학과 석사
2013~ 현재 KAIST 전기 및 전자공학과 박사과정

관심분야 : 모델링 및 시뮬레이션, 이산 사건 시스템 모델링, C3 시스템, 국방 시뮬레이션

김 탁 곤 (tkim@kaist.ac.kr)

1975 부산대학교 전자공학과 학사
1980 경북대학교 전자공학과 석사
1988 Univ. of. Arizona, 전기및컴퓨터공학과 박사
1980~1983 부경대학교 통신공학과 전임강사
1987~1989 (미)애리조나 환경연구소 연구엔지니어
1989~1991 Univ. of Kansas 전기 및 컴퓨터공학과 조교수
1991~현재 KAIST 전기 및 전자공학과 교수
- 한국시뮬레이션 학회 회장 역임
- 국제시뮬레이션학회(SCS) 논문지(Simulation) Editor-In-Chief 역임
- SCS Fellow
- 모델링 시뮬레이션 기술사(미국)
- *Who's Who in the World* (Marguis 16thEdition, 1999) 등재
- 연합사, 국방부/합참, 기품원 자문위원 역임
- KIDA Fellow 역임
- ADD 자문위원(현)

관심분야 : 모델링/시뮬레이션 이론, 방법론 및 환경개발, 시뮬레이터 연동

송 해 상 (hssong@seowon.ac.kr)

2000 KAIST 전기 및 전자공학과 박사
1999~2000 고등기술연구원
2001~2002 (주)스페이스네트
2002~현재 서원대학교 컴퓨터공학과 교수

관심분야 : 시스템 모델링 시뮬레이션 이론 및 응용, 국방 시스템 공학, 소프트웨어 공학