Regular paper | Journal of the KIECS. pp. 361-366, vol. 18, no. 2, Apr. 30. 2023, t. 118, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.2.361

노드의 동적 다운 스케일링을 지원하는 분산 클러스터 시스템의 설계 및 구현

류우석*

Design and Implementation of Distributed Cluster Supporting Dynamic Down-Scaling of the Cluster

Woo-Seok Ryu*

요 약

빅데이터의 분산 처리를 수행하기 위한 대표적인 프레임워크인 하둡은 클러스터 규모를 수천 개 이상의 노드까지 증가시켜서 병렬분산 처리 성능을 높일 수 있는 장점이 있다. 하지만 클러스터의 규모를 줄이는 것은 결함이 있거나 성능이 저하된 노드들을 영구적으로 퇴역시키는 수준에서 제한되어 있음에 따라 소규모 클러스터에서 여러 노드들을 유연하게 운용하기에는 한계가 있다. 본 논문에서는 하둡 클러스터에서 노드를 제거할 때 발생하는 문제점을 논의하고 분산 클러스터의 규모를 탄력적으로 관리하기 위한 동적 다운 스케일링 기법을 제안한다. 일시적 다운스케일을 목적으로 노드를 제거할 때 완전히 퇴역시키는 것이 아니라 일시적으로 해제하고 필요시 다시 연결할 수 있도록 함으로써 동적 다운 스케일링을 지원할 수 있도록 시스템과 인터페이스를 설계하고 구현하였다. 실험 결과 성능저하 없이 효과적으로 다운 스케일링을 수행하는 것을 검증하였다.

ABSTRACT

Apache Hadoop, a representative framework for distributed processing of big data, has the advantage of increasing cluster size up to thousands of nodes to improve parallel distributed processing performance. However, reducing the size of the cluster is limited to the extent of permanently decommissioning nodes with defects or degraded performance, so there are limitations to operate multiple nodes flexibly in small clusters. In this paper, we discuss the problems that occur when removing nodes from the Hadoop cluster and propose a dynamic down-scaling technique to manage the distributed cluster more flexibly. To do this, we design and implement a modified Hadoop system and interfaces to support dynamic down-scaling of the cluster which supports temporary pause of a node and reconnection of it when necessary, rather than decommissioning the node when removing a node from the Hadoop cluster. We have verified that effective downsizing can be performed without performance degradation based on experimental results.

키워드

Down-Scaling, Cluster, Hadoop, Decommission
다운 스케일링, 클러스터, 하둡, 퇴역

* 교신저자 : 부산가톨릭대학교 병원경영학과
• 접 수 일 : 2023. 02. 25
• 수정완료일 : 2023. 03. 20
• 게재확정일 : 2023. 04. 17
• Received : Feb. 25, 2022, Revised : Mar. 20, 2023, Accepted : Apr. 17, 2023
• Corresponding Author : Woo-Seok Ryu
Dept. of Health Care Management, Catholic University of Pusan,
Email : wsryu@cup.ac.kr

Ⅰ. 서 론

빅데이터 처리를 위한 대표적인 플랫폼으로서 아파치 하둡은 단일 시스템에서 수용 가능한 데이터의 한계를 넘어 다수의 노드를 클러스터로 구성하여 데이터의 병렬 처리를 수행할 수 있는 분산 플랫폼이다. 막대한 양의 데이터를 처리할 수 있음에 따라 보건의료 등 다양한 응용 분야에서 하둡을 이용한 빅데이터 분석이 활발히 수행되고 있다[1-3].

하둡이 가지는 가장 큰 장점 중의 하나는 분산 노드들로 구성된 클러스터가 실시간으로 확장 가능하다는 점이다[4]. 이를 통해 수십 대의 노드로 구성된 클러스터와 동일한 구성을 수천대의 노드로 확장하여 수행할 수 있으며, 그 또한 실시간으로 노드를 클러스터에 추가함으로써 가능한 장점이 있다. 하지만, 실시간 확장과는 달리 클러스터의 규모를 줄이는 것에는 제약이 있다. 클러스터의 규모를 줄이기 위한 방법으로는 고장이 발생하였거나 더 이상 사용하기 어려운 노드들을 클러스터에서 완전히 배제하기 위해 퇴역(decommission)시키는 방법만 제시되어 있으며 일시적으로 규모를 줄이는 방법은 제시되어 있지 않다.

본 연구에서는 중소기업, 중소병원 등 소규모의 비즈니스 환경에서 하둡 플랫폼을 도입하여 분산 클러스터를 구축하기 위한 방안을 논의한다. 이 경우 별도의 분산 클러스터를 구축하여 분석에 활용하기 보다는 기존의 유휴 시스템들을 최대한 활용하여 분산 클러스터를 구축하는 것이 보다 비용 효율적이다. 이를 위해서는 클러스터의 규모를 실시간에 추가하거나 제거하고 필요시 유휴 노드들을 간단하게 다시 클러스터에 편입시킬 수 있도록 노드의 동적 스케일링을 지원하는 것이 중요하다. 이를 통해 일상적 업무에 사용하는 시스템들이 유휴 상태일 때 하둡 클러스터로 편입하여 분석에 활용하고 이를 다시 일상 업무로 활용할 수 있도록 빠르게 전환할 수 있는 장점이 있다.

하둡 클러스터의 스케일링과 관련한 관련 연구로서 연구 [5]는 워크로드에 따라 클러스터의 규모를 동적으로 제어하기 위한 DSHYARN을 제안하였다. 하지만 이 방법은 클러스터 환경에서 적용이 가능하며 클러스터의 규모를 사용자가 직접 제어하지 못함에 따라 본 연구와 차이가 있다. 얀(YARN) 프레임워크에서의 작업 수행의 에너지 효율성을 높이기 위한 맵리듀스 잡 스케쥴링 기법을 제안한 연구에서도 클러스터의 규모를 동적으로 변경하는 방법은 제시하고 있지 않다[6-7].

본 연구에서는 하둡 분산 클러스터에서 서버의 종료 없이 사용자의 요구에 따라 클러스터에 포함된 노드를 일시적으로 제거하고 필요시 다시 추가함으로써 노드의 동적 스케일링을 지원하기 위한 메커니즘의 설계 및 구현을 제시하고자 한다. 기존의 선행연구를 확장하여 본 연구에서는 구현 알고리즘을 보다 명확하고 상세하게 기술하고 실험을 통한 성능 비교를 제시하고자 한다[8-9].

본 논문의 목차는 다음과 같다. 먼저 2장에서는 하둡 클러스터의 구조를 논의하고 노드를 중지할 때 발생할 수 있는 문제들을 논의한다. 3장에서는 하둡 클러스터에서 노드의 동적 스케일링 지원을 위한 시스템 설계 및 구현을 기술한다. 4장에서는 노드를 제거하는 다양한 시나리오에 따른 성능 비교를 통해 제안한 시스템의 유효성을 평가하고 5장에서는 결론 및 향후 연구를 기술한다.

Ⅱ. 하둡 프레임워크

하둡 프레임워크는 빅데이터를 다수의 노드들에 분산 저장하여 관리하는 하둡 분산 파일시스템(HDFS, Hadoop Distributed File System)[10]과 맵리듀스 등 사용자 작업에 대한 분산 처리를 수행할 수 있는 작업 스케쥴링 및 자원관리 프레임워크인 얀(YARN)[11]으로 구성된다.

그림 1에 도시된 바와 같이 이 프레임워크는 다수의 슬레이브 노드와 하나의 마스터 노드로 구성된 클러스터에서 구동된다. HDFS의 경우 슬레이브 노드에는 데이터를 저장 관리하기 위한 데이터노드(Data Node) 프로세스가 구동되고 마스터 노드에서는 데이터노드를 관리하기 위해 네임노드(Name Node) 프로세스가 구동된다. Yarn의 경우 사용자 작업을 의미하는 잡(Job)을 관리하고 스케쥴링하기 위한 리소스 매니저(Resource Manager)가 마스터 노드에서 동작하고 개별 노드에서 수행할 수 있도록 잡을 분할한 태스크들은 슬레이브 노드에 있는 노드 매니저(Node Manager)를 통해 관리된다.

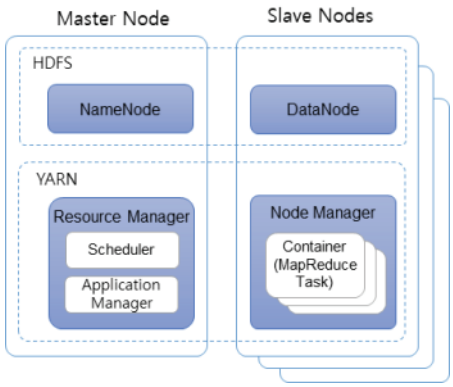

그림 1. 하둡 프레임워크의 구조
Fig. 1 Structure of the Hadoop Framework

특정 노드의 중지가 필요한 경우 HDFS와 Yarn은 각각 해당 노드에 대한 퇴역(decommission) 작업을 수행한다. Yarn은 해당 노드에서 수행되고 있는 태스크들을 종료시키고 다른 노드에서 수행하도록 다시 스케쥴링하는 방식으로 퇴역작업을 진행하며, HDFS의 경우 데이터 블록의 복제 계수(Duplication Factor)를 유지시키기 위해 해당 노드가 저장하고 있는 데이터 블록들을 다른 노드들로 복제하는 방식으로 퇴역작업이 진행된다.

퇴역이 완료된 슬레이브 노드는 더 이상 유효한 데이터 블록을 갖고 있지 않게 된다. 이로 인해 기 퇴역한 노드를 다시 클러스터에 편입해야 하는 경우 데이터 복제를 다시 수행해야 하는 문제가 발생함에 따라 즉각적인 재 편입이 어려운 문제가 있다. 여러 슬레이브 노드들을 실시간에 클러스터에서 제거하고 이를 다시 편입하게 되면 데이터 블록의 복제로 인한 시스템 부하가 더욱 커질 수 밖에 없는 문제가 발생한다.

III. 시스템 설계 및 구현

이 장에서는 클러스터의 동적 스케일링을 위해 슬레이브 노드를 퇴역시키고 다시 편입시키는 대신 일시적으로 중지하고 필요시 다시 재개할 수 있도록 하둡 분산 클러스터를 수정 구현한 메커니즘을 기술하고자 한다. 이는 아파치 하둡 버전 2.7.4 버전을 기준으로 소스 코드를 일부 수정하고 쉘 스크립트를 구현하는 형태로 개발되었다. 먼저 수정된 부분에 대한 전체 흐름을 기술하고 쉘 인터페이스, HDFS, Yarn 순서대로 기술한다.

3.1 전체 시스템 구조

본 연구에서 구현한 전체 시스템 구성도는 그림 2와 같다. 본 연구에서 구현한 NodeManage 쉘 스크립트를 통해 사용자가 노드의 일시정지를 요청하면(단계1) 이 스크립트는 마스터 노드의 네임노드 프로세스와 리소스 매니저 프로세스에게 각각 노드의 리프레시를 요청한다.(단계 2~3), 네임노드에서는 해당 노드를 퇴역이 아닌 일시 정지된 노드로 지정하고 이후 데이터노드의 신호가 올 때 프로세스를 종료한다.(단계 4~5). 리소스 매니저에서는 해당 노드를 퇴역처리하고 다른 노드에서 해당 태스크를 다시 수행할 수 있도록 처리한다(단계 6~7).

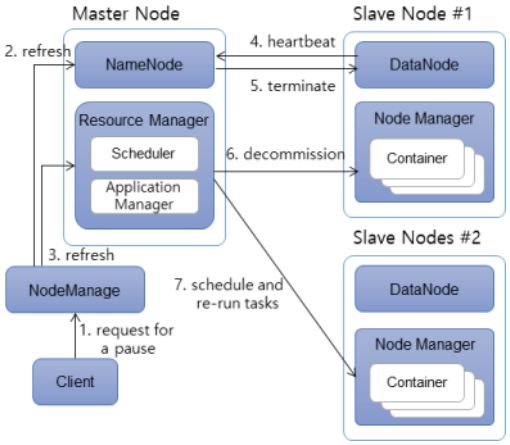

그림 2. 슬레이브 노드 일시 정지의 흐름도
Fig. 2 Flow Diagram for Pausing a Slave Node

3.2 시스템 인터페이스 구현

슬레이브 노드의 일시 정지 및 재시작을 처리하기 위해 본 연구에서 구현한 호출 인터페이스의 알고리즘은 그림 3과 같다. NodeManage.sh는 쉘 스크립트로서 일시정지(pause) 또는 재편입(resume) 요청에 따라 HDFS와 Yarn에게 명령을 전송한다. 일시정지 요청의 경우 해당 노드의 이름(nodename)을 pause 파일에 저장 후 HDFS와 YARN에게 refreshNode 명령을 각각 호출한다.

```
NodeManage.sh pause|resume nodename

pause/resume : a command of the procedure
nodename : An ID of the node to be processed

 1  if the command is pause then
 2      add nodename to the pause file
 3
 4      call hdfs dfsadmin -refreshNodes
 5      call yarn rmadmin -refreshNodes
 6  else if the command is resume then
 7      remove nodename from the pause file
 8
 9      call hdfs dfsadmin -refreshNodes
10      call yarn rmadmin -refreshNodes
11
12      call hdfs datanode
13      call yarn nodemanager
14  end if
```

그림 3. NodeManager.sh의 알고리즘
Fig. 3 An Algorithm of the NodeManage.sh

재편입 요청의 경우 pause 파일에서 해당 노드의 이름을 제거한 후 앞과 동일하게 refreshNode 명령을 각각 호출한다. 그 이후 HDFS와 YARN을 다시 시작함으로써 해당 노드 프로세스들을 다시 시작한다.

3.3 HDFS의 구현

슬레이브 노드의 데이터노드 프로세스는 정상적으로 동작하는 동안 주기적으로 하트비트(heartbeat) 신호를 마스터 노드에 위치한 네임노드 프로세스에게 전송한다. 이 신호가 일정 시간(timeout)동안 전송되지 않는 경우 네임노드 프로세스는 해당 프로세스가 정지된 것으로 판단하고 퇴역 작업을 시작하여 해당 슬레이브 노드가 가지고 있던 데이터 블록들을 다른 슬레이브 노드들에게 복제한다.

데이터노드를 퇴역시키지 않고 일시적으로 정지하는 상태를 표현하기 위해 본 연구에서는 데이터노드의 상태에 PAUSED를 추가한다. 이를 위해 수정된 구현 내용은 다음과 같다.

1. pause 파일에 일시 정지가 필요한 노드의 이름을 지정한다.

2. HostFileManager 모듈이 노드의 리프레시 명령을 받으면 기존의 include, exclude 외에 pause파일을 함께 읽어서 일시 정지할 노드를 확인하고 이를 일시 정지 노드로 지정하고 DatanodeManager 모듈의 리프레시 명령을 호출한다.

3. DatanodeManager는 해당 노드가 일시 정지 요청을 받은 경우 퇴역 작업을 시작하지 않고 일시 정지 상태를 유지한다.

4. DatanodeManager 모듈은 슬레이브 노드에 있는 데이터노드 프로세스로부터 하트비트 신호 수신시 해당 노드가 일시정지 상태임을 확인하면 해당 노드의 프로세스를 중지한다.

5. DatanodeManager는 일시정지 상태인 노드에 대해서는 하트비트 신호의 미수신으로 인한 타임아웃이 발생하더라도 퇴역작업을 진행하지 않고 일시 정지 상태를 유지한다.

3.4 Yarn의 구현

Yarn에서는 슬레이브 노드의 일시 정지가 필요한 경우 해당 노드에서 동작하고 있는 노드매니저 프로세스를 중지시키고 더 이상 잡 태스크들이 할당되지 않도록 한다. 이는 Yarn의 리소스매니저 프로세스에서 처리하는 노드 제외(node exclude)와 동일하게 처리가 가능하다. 즉, Yarn이 속성 중 제외 노드의 이름을 지정하는 exclude-path 속성이 지정하는 파일에 일시정지할 노드의 이름을 등록하는 방법으로 구현이 가능하다.

exclude-path 속성이 지정한 파일에 일시 정지할 노드를 추가하고 노드의 Refresh를 호출하면 해당 노드는 자동으로 종료가 되고 수행 중인 잡 태스크들은 다른 노드에서 자동으로 재시작된다. 슬레이드 노드의 재시작이 필요한 경우에는 exclude-path에 지정된 노드 리스트에서 해당 노드를 삭제하고 노드의 리프레시를 Yarn에 요청한다.

Ⅳ. 시스템 구현 결과

제안된 시스템의 타당성을 검증하기 위한 분산 클러스터 시스템으로 하나의 마스터 노드와 4개의 슬레이브 노드로 구성된 하둡 클러스터를 구축하였다. 각 노드는 2-코어 펜티엄 프로세서와 4GB RAM를 장착하고 우분투 14 운영체제로 구동되며 1기가비트 이더넷 스위치로 노드들을 연결하였다.

그림 4는 클러스터에 구성된 슬레이브 노드 중 하나에 대해 일시정지를 수행한 화면이다. 그림에서 볼 때 HN3은 일시정지(Paused) 상태로 전환된 상태에서 하트비트 신호를 미수신한 경과 시간이 18초로 표시되어 있는 것으로 볼 때 일시정지 상태가 적절히 전환되는 것을 확인할 수 있다.

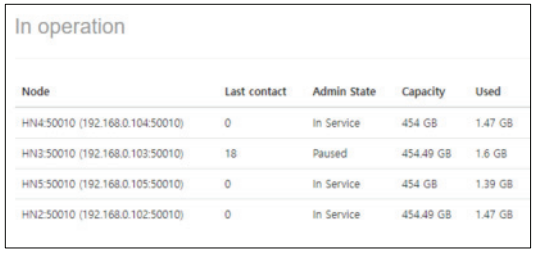

그림 4. 노드 일시 정지시의 하둡 관리자 화면
Fig. 4 Hadoop Admin Page When a Node is Paused

슬레이브 노드를 제거하는 다양한 시나리오에 따른 결과를 비교 실험하기 위해 2GByte 용량의 텍스트 데이터셋을 HDFS에 적재시키고 하둡에 내장된 워드카운트(wordcount) 프로그램을 잡으로 실행하여 그 수행시간을 측정하였다. 이때의 시나리오는 네 가지로 구성하였는데 각각 제거하지 않음(NORMAL), 하둡에서 지원하는 퇴역을 수행(DECOM), 본 연구에서 구현한 일시정지(PAUSE), 프로세스의 강제 종료(KILL)로 구분하였다.

네 가지 시나리오별로 5번씩 실험을 수행한 결과는 그림 5와 같다. 실험결과 NORMAL과 비교하여 DECOM과 PAUSE는 조금의 시간 차이는 발생하나 통계적으로 유의미한 정도의 차이는 발생하지 않았다. PAUSE의 경우 즉시 해당 노드를 종료하고 다른 노드를 통해 분산 수행하므로 전체적인 수행 시간의 차이가 크지 않다고 할 수 있다. DECOM의 경우 노드 제거시에는 PAUSE와 유사한 속도를 보이나 HDFS에 저장된 더 이상 유효하지 않은 데이터 블록들을 다른 노드들로 옮기는 작업이 백그라운드에서 발생하므로 불필요한 I/O 비용이 추가적으로 발생한다고 할 수 있다.

KILL 시나리오의 경우 노드의 하트비트 신호를 타임아웃 시간까지 기다린 다음 퇴역 작업을 이관함에 따라 수행 시간이 상대적으로 매우 길고 수행시간의 차이 또한 매우 다양함을 확인할 수 있다. 실험결과 본 연구에서 제안한 일시 정지 기법이 수행시간의 손실을 최소화하는 동시에 노드를 완전히 퇴역시키지 않음에 따라 언제든지 다시 재사용 가능하다는 이점을 동시에 확보할 수 있다.

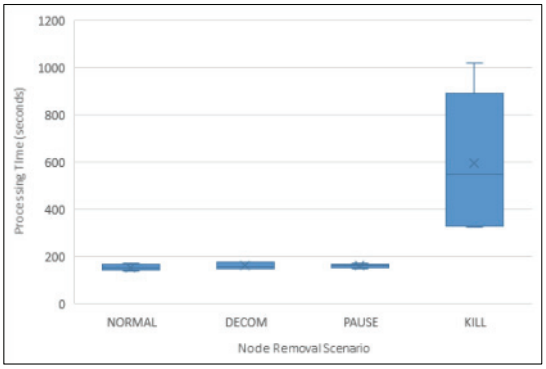

그림 5. 노드 제거 시나리오별 처리 시간을 비교한 박스플롯(x축: 시나리오, y축: 처리시간(초))
Fig. 5 A Box plot for Comparing Processing Time According to the Node Removal Scenarios

V. 결 론

본 논문에서는 하둡 클러스터에서 노드를 다운 스케일링할 때 발생하는 문제점을 논의하고, 클러스터를 동적으로 스케일링할 수 있도록 노드의 완전한 퇴역 대신 일시정지를 시키고 필요시 바로 다시 클러스터에 편입하는 방식으로 노드의 동적 스케일링을 지원하는 메커니즘을 설계하고 하둡 오픈 소스를 수정 구현하였다. 실험을 통해 그 성능을 확인한 결과 노드를 제거하지 않은 경우와 거의 동일한 성능을 보였으며, 해당 노드를 강제 종료시킨 것에 비해 보다 우수한 수행 성능을 보였음에 따라 노드를 보다 유연하게 관리할 수 있음을 확인할 수 있었다.

본 연구 결과를 활용하면 하둡 클러스터를 지정된 노드로만 구성하는 대신 기존의 유휴 노드들을 선택적으로 활용할 수 있으며, 비교적 작은 규모의 환경에서도 유연하게 빅데이터 처리를 수행할 수 있다. 본 연구의 향후 연구로서 맵 리듀스에서의 작업을 일시정지하고 다른 노드로 연계하여 처리하기 위한 연구를 통해 동적 스케일링을 보다 고도화할 필요가 있다.

감사의 글

이 연구는 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구 임(No. NRF-2016R1C1B1012364).

References

[1] H. Ryoo and M. Lee, "The study of Recommendation system for food ingredients through the controlling food recipe's ingredient rate with the ICT and Big Data," *J. of The Korea Institute of Electronic Communication Sciences*, vol. 16, no. 2, 2021, pp. 339-346.

[2] H. Harb, H. Mrouse, A. Mansour, A. Nasser, and E. M. Cruz, "A hadoop-based platform for patient classification and disease diagnosis in healthcare applications." *Sensors*, vol. 20, no. 7, 2020, pp. 1931-1951.

[3] S. Yang, Y. Kim, S. Kim, and W. Kim, "Energy Big Data Pre-processing System for Energy New Industries," *J. of The Korea Institute of Electronic Communication Sciences*, vol. 16, no. 5, 2021, pp. 851-858.

[4] W. K. Lai, Y. U. Chen, and T. Y. Wu, "Towards a framework for large-scale multimedia data storage and processing on Hadoop platform," *Journal of Supercomputing*, vol. 68, no. 1, 2013, pp. 488-507.

[5] W. Nemouchi, S. Boudouda, and N. Zarour, "A Dynamic Scaling Approach in Hadoop YARN," *Int. J. of Organizational and Collective Intelligence (IJOCI)*, vol. 12, no. 2, 2022, pp. 1-17.

[6] Y. Gao and C. Huang, "Energy-efficient scheduling of mapreduce tasks based on load balancing and deadline constraint in heterogeneous hadoop yarn cluster," In *IEEE 24th Int. Conf. on Computer Supported Cooperative Work in Design (CSCWD)*, Dalan, China, May, 2021, pp. 220-225.

[7] V. Pandey and P. Saini, "A heuristic method towards deadline-aware energy-efficient mapreduce scheduling problem in Hadoop YARN," *Cluster Computing*, vol. 24, no. 2, 2021, pp. 683-699.

[8] W. Ryu, "Implementation of dynamic node management in Hadoop cluster," In *Int. Conf. on Electronics, Information, and Communication (ICEIC 2018)*, HI, USA, 2018, pp. 814-815.

[9] W. Ryu, "Design of Elastic Hadoop Supporting Dynamic Scaling of the Cluster," In *4th Int. Conf. on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2018)*, IARIA, Athens, Greece, 2018, pp. 26-27.

[10] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," In *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, NV, USA, May 2010, pp. 1-10.

[11] Y. Gao and K. Zhang, "Deadline-aware Preemptive Job Scheduling in Hadoop YARN Clusters," In *IEEE 25th Int. Conf. on Computer Supported Cooperative Work in Design (CSCWD)*, Hangzhou, China, May, 2022, pp. 1269-1274.

저자 소개

류우석(Woo-Seok Ryu)

1997년 부산대학교 컴퓨터공학과 졸업 (공학사)
1999년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사)
2012년 부산대학교 대학원 컴퓨터공학과 졸업(공학박사)
2013년 ~현재 부산가톨릭대학교 병원경영학과 부교수
※ 관심분야 : 의료정보, 의학용어, 빅데이터