Hadoop 상에서 MapReduce 응용프로그램 평가

(Performance Evaluation of MapReduce Application running on Hadoop)

김 준 수[‡] 강 윤 희[§] 박 용 범[¶]
(Junsu Kim) (Yunhee Kang) (Youngbom Park)

요 약 다양한 분야에서 빠르게 대용량의 자료가 생성됨에 따라 이를 처리하기 위해 분산 프로그래밍 모델인 MapReduce의 활용이 도입되고 있다. 본 논문에서는 SUN Blade150에 Solaris와 Linux 환경의 클러스터 시스템을 구축한 뒤 해당 환경에서의 MapReduce 미들웨어인 Hadoop 에서 응용수행에 대한 평균 시간 및 표준 편차를 평가하여 Hadoop 기반 MapReduce 구현이 어떠한 클러스터 시스템에 의해 성능이 영향을 미치는지를 보인다.

키워드 MapReduce, 성능 평가, 클러스터 시스템, Hadoop

Abstract According to the growth of data being generated in man fields, a distributed programming model MapReduce has been introduced to handle it. In this paper, we build two cluster system with Solaris and Linux environment on SUN Blade150 respectively and then to evaluate the performance of a MapReduce application running on MapReduce middleware Hadoop in terms of its average elapse time and standard deviation. As a result of this experiment, we show that the overall performance of the MapReduce application based on Hadoop is affected by the configuration of the cluster system.

Key words MapReduce, Performance evaluation, Cluster system, Hadoop

1. 서 론

다양한 분야에서 빠르게 대용량의 자료가 생성되고 대용량에 대한 데이터 처리 요구가 증가하고 있으나 현재 디스크에 대한 접근은 대용량의 경우 데이터 전송 시 병목 문제를 겪는다[4,6]. 이러한 문제점을 개선하기 위해 분산 컴퓨팅을 지원하기 위한 목적으로 개발된 소프트웨어 프레임워크인 MapReduce의 활용이 증가하고 있다[2].

MapReduce는 다양한 분산 컴퓨팅 환경에서 동작하기 때문에 노드들 사이의 네트워크 환경, 운영체제의 차이 등 환경적인 요소에 의해 실행 속도 차이가 발생한다. 본 논문에서는 플랫폼 및 운영체제에 따른 분산 컴퓨팅 성능 분석을 위해 Solaris와 Linux 환경에서 성능 측정 및 분석을 수행하며, 실행 환경으로는 Solaris와 Linux 환경에서 MapReduce 응용인 WordCount의 성능을 비교하고자 한다. Solaris와 Linux는 같은 Unix 시스템을 기반으로 만들어진 운영체제이나 MapReduce 응용의 성능 차이가 존재 할 수 있으므로, MapReduce 응용 어플리케이션 중 하나인 WordCount를 사용하여 실험을 진행하며, Combiner의 사용 유무로 인한 두 운영체제의 속도와 안정성에 대해 비교 및 분석을 한다. 실험 환경으로는 Sparc 프로세서를 사용하며, Solaris 버전 10, Linux 배포판으로 Debian 버전 6.0.4를 사용하여 진행한다. 실험에 사용하는 MapReduce는 Apache의 오픈소스 프로젝트인 Apache Hadoop[7]을 이용하여 구성 하고자 한다.

‡ 비 회 원 : 단국대학교 컴퓨터과학과, tomy2174@gmail.com
§ 비 회 원 : 백석대학교 정보통신학부 조교수, yunh.kang@gmail.com
¶ 회 원 : 단국대학교 컴퓨터과학과 교수, ybpark@dankook.ac.kr

논문의 구성은 2장에서 본 논문의 기초가 되는 MapReduce와 Combiner에 대한 관련 연구에 대해 기술하고, 3장에서는 Debian과 Solaris 클러스터 간의 MapReduce 응용에 대한 수행 시간 및 표준 편차에 대한 결과를 분석한다. 마지막 4장에서는 운영체제 환경에 따른 성능 분석 결과를 논하고 Hadoop 시스템 구축에 필요한 환경적 요소를 정의한 후 향후 연구 방향에 대하여 기술한다.

2. 기본 개념

2.1. MapReduce

MapReduce는 Google에서 제안한 분산 컴퓨팅 지원 목적의 프레임워크로서 주어진 문제를 추상화해서 키와 값에 대한 계산으로 변환한 프로그래밍 모델이다[1,5]. MapReduce는 Map과 Reduce 단계로 구성된다. Map에서는 입력 데이터에서 키와 값을 추출하여 이 추출된 데이터를 Reduce에서 키 기준으로 값을 연산하여 분산 파일 시스템에 결과를 출력한다. Map 함수와 Reduce 함수 수행을 위한 Mapper 태스크와 Reducer 태스크는 병렬로 데이터를 처리하며, 별도로 시작하여 상호간의 의존성은 없다. MapReduce는 추상화 되어 있기에 프로그래머는 키와 값에 대한 고려만 하여 이를 작성할 수 있으며, 데이터 흐름은 내부 플랫폼에서 담당한다. 그림 1은 MapReduce의 내부 흐름도를 보인 것이다.

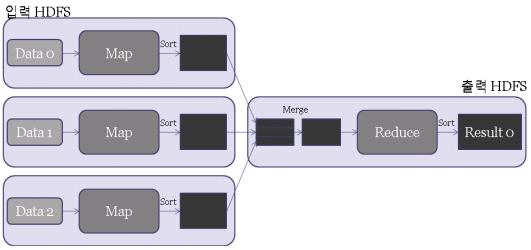

그림 1 MapReduce 내부 흐름도

2.2 Combiner

Combiner는 미니 리듀서라고도 불리며 MapReduce 응용을 수행 하는 과정에서 Map 작업에 필요한 데이터들이 어느 한 작업노드에 편중이 되어 있으면 Mapper 태스크에서 Reducer 태스크로 키와 값들을 전달할 경우 트래픽 폭주로 인하여 MapReduce 응용의 수행 속도가 느려지거나 네트워크가 stall 되는 상황이 일어날 수 있다. 이러한 문제를 해결하기 위하여 Map 작업과 Reduce 작업 사이에 Combiner라는 단계 있으며 Reduce 작업을 도와준다[3]. 그림 2는 Hadoop 미들웨어에서 Combiner의 위치를 기술한 것이다.

Combiner는 같은 노드의 Mapper 태스크에서 키와 값들을 받아 현재 작업 노드에서 자체적으로 Reduce 작업을 실행을 한 뒤 Reducer 태스크에게 Combine 작업을 한 결과인 키와 값을 전달한다. Combiner를 사용하게 될 경우 네트워크 트래픽이 크게 감소될 수도 있다. 그러나 값들의 연산이 교환법칙 혹은 결합법칙이 성립되지 않을 경우에는 값이 변경이 될 수 있으므로 사용하지 않는 것이 적절하다.

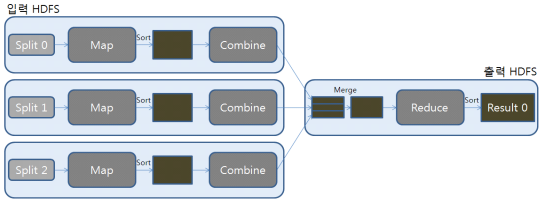

그림 2 Combiner를 사용한 MapReduce 내부 흐름도

3. 성능평가

3.1. 실험환경 구축

작성된 Hadoop 응용은 다양한 범용 컴퓨팅 자원에서 작업을 수행한다. 다음에 나오는 표 1은 실험 환경의 하드웨어 플랫폼 및 운영체제 구성을 기술한 것이다.

표 1. 실험을 위한 컴퓨팅 구성

	Type-1	Type-2
CPU 사양	Sparc Processor 650Mhz * 1 ~ 12	
코어 수	1 ~ 12	
메모리크기	512Mbyte * 1 ~ 12	
운영체제	Solaris 버전 10	Debian 버전 6.0.4
MapReduce 미들웨어	Hadoop 1.0.2	

이 실험에서는 작업 노드의 구성에 따라 Type-1과 Type-2로 구분하여 사용하였다. Type-1은 Solaris 10 버전을 사용하며, Type-2에는 Debian 6.0.4 를 사용하며, 하드웨어는 같은 기종의 서버들을 사용한다. 또한 Type-1과 Type-2의 마스터 노드는 별도의 서버에서 수행하도록 분산한다.

3.2 결과 및 성능평가

성능평가를 위하여 507MB 크기의 임의의 텍스트 파일을 이용하여 작업 노드가 1대인 경우부터 12대인 경우까지 각각 10회씩 WordCount 수행하여 수행 시간의 평균값 및 표준편차에 대하여 평가한다. <그림 3>은 노드가 추가됨에 따른 MapReduce 의 응용 수행의 응답 시간을 보인 것이다.

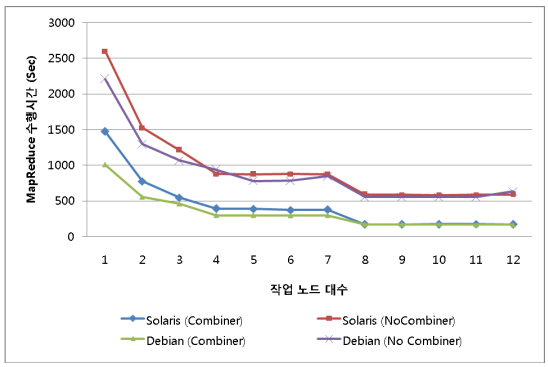

그림 3 Debian과 Solaris 환경에서의 MapReduce 응용 수행 시간

표 2는 MapReduce 의 응용 시간을 보인 것으로 실험을 통해 Combiner를 사용할 경우 Combiner를 사용 하지 않은 경우에 비하여 MapReduce 응용 수행 시간이 Solaris는 44.11%, Debian은 37.71% 더 적어진 것을 볼 수 있다. 그러나 Combiner를 사용하지 않은 경우에서는 작업 노드로 4대를 사용한 경우와 12대를 사용한 경우에는 MapReduce 응용의 평균 수행 시간이 Debian 보다 Solaris가 더 적어진 것으로 나왔으며, 7대를 사용한 경우에도 작업 노드를 6대 사용한 경우보다 상대적으로 MapReduce 응용 수행 시간이 상대적으로 증가 하였다. 이러한 이유는 각각의 작업 노드를 4대, 7대, 12대로 증가하여 실험을 하는 경우 MapReduce 작업이 실패하여 시간이 지연이 된 것이다.

표 2. Debian과 Solaris 환경에서의 MapReduce 응용 수행 시간

(단위: 초)

작업노드 대수	Solaris (Combiner)	Solaris (No Combiner)	Debian (Combiner)	Debian (No Combiner)
1	1474.5	2599.9	1007.6	2207.3
2	777.1	1524.0	554.5	1297.4
3	547.9	1213.6	458.8	1067.4
4	393.0	879.5	297.4	936.9
5	387.7	874.1	298.0	779.5
6	374.5	879.2	293.9	783.5
7	374.7	875.8	296.5	847.5
8	171.1	588.3	171.0	552.9
9	172.8	586.3	171.9	554.0
10	173.6	582.5	172.8	553.1
11	174.8	584.8	168.3	553.0
12	172.8	587.8	169.0	634.3
합계	5194.5	11775.8	4059.7	10766.8
평균	432.9	981.3	338.3	897.2

그림 4로부터 Debian은 Combiner를 사용 하지 않은 경우는 Combiner를 사용하였을 경우보다 표준 편차의 크기가 959.28%로 증가한 것을 볼

수 있었다. 그러나 Solaris에서 Combiner를 사용하지 않은 경우와 Combiner를 사용한 경우의 표준편차를 비교하면 83.94%로 오히려 감소한 것을 볼 수 있다.

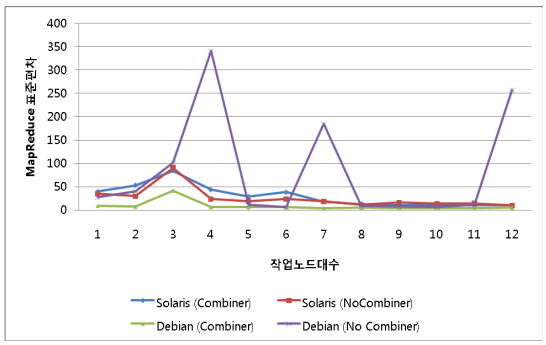

그림 4 MapReduce 응용 수행 시간에 대한 표준편차

표 3은 MapReduce 응용 시간의 표준편차를 보인 것으로 Debian 환경에서 Combiner를 사용하는 경우가 응용수행 시간의 편차가 적은 것을 보이고 있으며 Combiner를 사용하였을 때 가장 높은 편차 값을 보인다.

표 3. MapReduce 응용 수행 시간에 대한 표준편차

(단위: 초)

작업노드 대수	Solaris (Combiner)	Solaris (No Combiner)	Debian (Combiner)	Debian (No Combiner)
1	39.4863	34.7321	9.2280	26.6252
2	53.1736	29.4165	7.4722	39.9811
3	83.9093	90.1359	41.8484	100.8444
4	43.7061	23.2869	6.6366	340.4365
5	29.1892	18.9059	6.1464	10.8858
6	38.2281	23.5504	5.8013	5.8926
7	18.0373	18.2257	4.1700	184.0551
8	13.0678	11.1360	4.8762	8.1302
9	10.3688	16.3371	4.2804	6.6667
10	11.3744	13.1255	4.3665	6.5056
11	10.2610	14.0539	3.8601	10.6249
12	10.0532	9.9867	5.2915	256.7857
합계	360.9	302.9	104.0	997.4
평균	30.1	25.2	8.7	83.1

4. 결 론

우리는 앞서 Solaris 와 Linux 간의 Combiner를 사용 하였을 경우와 사용하지 않은 경우에 대하여 MapReduce 응용에 대한 수행 속도와 안정성의 차이를 살펴보았다. 이 실험에서 Combiner를 사용하고 Solaris와 Linux의 MapReduce응용 수행 시간을 측정한 결과 Linux가 Solaris 보다 상대적으로 안정적이고 전체적으로 적은 시간이 소요된 것을 볼 수 있었다.

Combiner를 적용 시키지 않고 실험한 결과 또한 전반적으로 Linux가 Solaris에 비해 시간이 적게 소요가 되었으나 안정적인 측면에서 Solaris는 MapReduce 응용 수행에 실패한 경우가 없으나 Linux는 작업 노드의 수가 4대, 7대, 12대인 경우 MapReduce 응용 수행이 실패한 결과가 전체 작업 시간에 영향을 미쳐 Linux는 Solaris 보다 수행 시간이 큰 것을 관찰 하였다.

본 논문에서는 MapReduce를 구동할 경우 Linux를 사용하면 속도가 Solaris보다 높은 속도를 낼 수 있으나 Combiner를 사용하지 않아 네트워크 리소스를 많이 사용하게 될 경우에는 Linux보다 Solaris가 더 안정적인 것을 볼 수 있었다.

본 논문에서는 MapReduce 측정 실험으로 적은 용량의 파일로 WordCount 응용을 사용하여 작업을 수행 하였으나 더 대용량 파일 및 다른 MapReduce 응용에서의 성능평가를 수행할 예정이다.

참 고 문 헌

[1] J. Dean and S. Ghemawat, “Mapreduce: Simplified Data Processing on Large Clusters,“ Communications of the ACM, Vol. 51(1), pp. 107-113, Jan., 2008.

[2] 강윤희, “FutureGrid 클라우드 컴퓨팅 환경에서의 Twister 수행 MapReduce 응용 구성“, 한국 정보 기술 학회 논문지 제9권 제4호, 147-154, 2011.4

[3] Chuck Lam, “Hadoop in Action“, Manning, 2010

[4] Tom White, “Hadoop: The Definitive Guide, 2nd Edition“, O’Reilly Media / Yahoo Press, 2010

[5] J. Dean and S. Ghemawat, MapReduce: A Flexible Data Processing Tool. CACM, vol. 53, pp. 72-77, 2010

[6] K. Morton, A. Friesen, M. Balazinska and D. Grossman, Estimating the Progress of MapReduce Pipelines, IEEE 26th International Conference on Data Engineering (ICDE), 2010, pp. 681 - 684, Long Beach, CA, 2010

[7] Hadoop, http://hadoop.apache.org

저 자 소 개

김 준 수

2011년~현재 단국대학교 컴퓨터과학과 재학

<관심분야> 클라우드 컴퓨팅, 분산 처리 시스템

강 윤 희

1989년 동국대학교 컴퓨터공학과(공학사)
1991년 동국대학교 컴퓨터공학과(공학석사)
2002년 고려대학교 컴퓨터과학과(이학박사)
2000년~현재 백석대학교 정보통신학부 조교수

<관심분야> 클라우드 컴퓨팅, 그리드 컴퓨팅, 결함포용

박 용 범

1991년 N.Y. Polytechnic University Couputer Sci. & Eng. Ph.D.
1993년~현재 단국대학교 컴퓨터과학 교수

<관심분야> 지능형 소프드웨어 공학, 보안 소프트웨어 개발

커뮤니티 검출기법을 이용한 소프트웨어 아키텍쳐 모듈 뷰 복원†

(Recovering Module View of Software Architecture using Community Detection Algorithm)

김정민‡ 이찬근§
(Jungmin Kim) (Changun Lee)

요 약 본 논문은 소프트웨어 클러스터링 기법과 커뮤니티 검출 기법의 비교를 통하여 아키텍쳐 모듈 복원 프로세스에 커뮤니티 검출 알고리즘의 적용가능성을 제시한다. 또한, 대표적인 클러스터링 알고리즘과 커뮤니티 검출 알고리즘의 값과 나눠진 모듈간의 상관관계와 차이점을 분석한다. 이를 통하여 커뮤니티 검출 알고리즘이 소프트웨어 아키텍쳐 모듈 뷰 복원에 활용되어질 수 있다는 몇 가지 근거를 제시하였고, 기존의 클러스터링 결과와 커뮤니티 알고리즘의 결과치를 비교함으로써, 서로의 결과 데이터가 어떠한 연관성을 가지는지 제시하였다.

키워드 소프트웨어 모듈 뷰, 커뮤니티 스트럭쳐, 소프트웨어 클러스터링

Abstract This article suggests applicability to community detection algorithm from module recovering process of software architecture through compare to software clustering metric and community dectection metric. in addition to, analyze mutual relation and difference between separated module and measurement value of typical clustering algorithms and community detection algorithms. and then only sugeested several kinds basis that community detection algorithm can use to recovering module view of software architecture and, by so comparing measurement value of existing clustering metric and community algorithms, this article suggested correlation of two result data.

Key words : Software module view, Community Structure, Software Clustering

1. 서 론

네트워크의 현대적인 연구는 복잡계 네트워크의 융합 학문분야에 접목되어지는 과학에 집중되어 있다.[3] 많은 복잡계 네트워크 시스템들은 네트워크의 모양으로 나타낼 수 있고, 시스템의 최소 단위를 이루는 부분들로 이루어지는데, 그러한 네트워크는 노드와 링크를 통해 서로의 상호 연관성을 만들어 낸다. 이와 관련하여 복잡계 네트워크 안의 노드들은 한 모듈 안의 내부 링크끼리의 강한 밀집도를 통하여 노드들의 집합의 모양으로 나타내어 지는데, 반대로 다른 모듈간의 이어진 링크의 밀집도를 최소화 시켜서 각 모듈간의 독립성을 최대화 시키는 것이 연구의 핵심이다. 여기에서의 네트워크를 그래프로 표현할 때, 그래프 안의 내부 링크간의 밀집도가 높게 구성되어진 서브그래프를 모듈(클러스터링 기법) 또는 커뮤니티(커뮤니티 검출 기법)라 칭하게 된다.

복잡계 네트워크의 노드들을 구분하는 방법을 찾는 것은 그 시스템의 특성과 그들의 규칙에 의존

† 본 연구는 한국연구재단 기초연구사업(과제번호 2010-0015636, 2011-0013924)의 지원을 받았습니다.

‡ 학생회원 : 중앙대학교 컴퓨터공학부, jungmink26@lycos.co.kr

§ 종신회원 : 중앙대학교 컴퓨터공학부 교수, cglee@cau.ac.kr

하게 되는데, 그러므로 네트워크상의 커뮤니티들을 찾는 것은 네트워크 과학의 가장 기본적인 문제이다.

많은 방법들이 이미 제시되었고, 측정 도구는 물리학, 생물학, 그래프 이론에서의 수학적인 접근 그리고 컴퓨터와 소셜과학의 분야를 통틀어 활용되어지고 있다. 현재 네트워크(그래프)의 모듈화의 방법은 위에서 말했듯이 두 가지 접근 방법으로 설명이 가능한데, 그 두 기법은 약간의 차이가 있다.[1]

첫 번째로 소프트웨어 클러스터링의 주된 목적은 이미 개발자에 의해 구축되어진 아키텍쳐를 비교군으로 잡아, 그 소프트웨어 구조를 알고리즘으로 하여금 분류(군집)하게 하여 기존 아키텍쳐와 알고리즘을 통한 모듈화 결과의 일치성을 비교한다. 이는, 좋은 알고리즘이 소프트웨어 설계구조를 정의할 수 있다는 것으로 해석될 수 있으며, 이를 위해 몇 가지 솔루션을 연구 중에 있다.

두 번째로 커뮤니티 검출 기법은 “척도없는 네트워크”라는 소셜네트워크를 기반으로 한 네트워크 구조상에서의 적절한 커뮤니티를 나누고 그 나눠진 커뮤니티의 모듈성 측정을 목적으로 한다. 커뮤니티 검출의 경우 기본 컨셉이 매우 많은데, 각 노드의 연결관계를 통해 특정 노드의 영향력을 나타낸다거나, 네트워크의 특성에 따른 노드와 간선간의 의미를 어떻게 정의하느냐에 따라서 커뮤니티를 검출하는데에 큰 영향력을 끼친다고 할 수 있다. 하지만 기본적인 컨셉(네트워크의 특성)은 그래프 생성 단계에서 그래프의 구조적 방법으로 정의되기 때문에 알려진 몇몇 대표적인 알고리즘들이 존재한다.

다음으로는 이와 관련한 본 논문의 실험의 접근 방법 및 실험내용과 결과를 알아보도록 하겠다.

2. 접근 방법

커뮤니티 검출 기법의 본래 목적은 복잡계 네트워크상에서의 각 정점들끼리의 강한 연결 요소를 찾아내어 서로 다른 커뮤니티끼리의 독립성을 보장하는 데에 있다.

본 연구의 접근 방법은 이러한 커뮤니티 검출 기법이 실제 소프트웨어 아키텍쳐 모듈 복원에 활용되어 질 수 있는지를 알아본다. 이 실험을 통해 기존 클러스터링 알고리즘의 접근 방법(closeness, centrality)등에 비해 커뮤니티 검출 알고리즘(betweenness, modularity) 최대화 기법이 어느 정도의 효율을 보일 수 있고, 그 효과가 기존 알고리즘 등과 비교했을 때 적용가치가 있는지 등을 알아보겠다. 기본 동기는 현재 클러스터링 연구 분야가 기존 네트워크의 구조적 특성이나 노드의 각각의 특정 수치에 의존하여 모듈을 나눔으로써 의미를 파악하는데에 집중하고 있는 점을 비추어 볼때, 순수하게 노드의 영향력적인 가치를 반영하는 커뮤니티 검출방법이 이러한 모듈 뷰 복원 일치성을 높이는 노력에 도움이 될 수 있다 생각했기 때문이다.

3. 연구 내용

검출 데이터는 기본적으로 클래스 의존 관계와 우수한 package 모듈을 가지고 있는 자바 라이브러리를 사용하였다. 모든 데이터는 기본적으로 50개 이상의 클래스로 구성된 중/대규모 패키지이며, 이는 복잡계 네트워크를 표현할 수 있는 규모를 가지고 있다고 판단하였다. 자바 패키지가 척도없는 네트워크의 특성인 멱함수 분포를 보인다는 것은 이미 증명되어 있다. 이러한 실험을 진행하기 위한 추출되어질 라이브러리는 다음과 같다.

표 1 검출 데이터

검출 데이터	설 명
javax.swing	javax swing package
java org	java org library
com imageio	java com library
java.util	java util package

3.1. 추출 알고리즘

기존 클러스터링 기법과 커뮤니티 검출 알고리즘 2가지를 사용하여 같은 데이터에 대한 아키텍쳐 복원율을 비교한다. 대표적인 클러스터링 알고리즘으로서는 WeakComponentClusterer(WC) 기법을 활용할 것이다.[6]

커뮤니티 검출 알고리즘으로써 사용한 기법은 Girven-Newman의 EdgeBetweenness(EB), Modularity maximization(MM)이다. 이 두 알고리즘은 서브그래프를 추출하는 점에서는 클러스터링 알고리즘과 같으나, 커뮤니티를 검출한다는 점이 차이점이다.

3.2. 소프트웨어 아키텍쳐 복원율 측정

실험은 크게 두 가지로 나뉜다. 첫째로 클러스터링 계수측정을 통하여 커뮤니티 검출 알고리즘의 효용성을 판단하고 두 번째로, 실제 커뮤니티 검출 알고리즘이 소프트웨어 모듈 뷰 복원율 향상에 어느 정도 효과가 있는지를 알아볼 것이다. 실험 순서는 다음과 같다.

(1) 주어진 DataSet의 클래스 의존 그래프를 추출하고 커뮤니티 검출 알고리즘을 통해 DataSet의 모듈성을 측정한다.
(2) 각 알고리즘들에 관하여 클러스터링 계수를 측정해 본다.
(3) 클러스터링 알고리즘과 커뮤니티 검출 알고리즘을 이용하여 그래프를 모듈화 시킨 후, 그 결과를 MoJo알고리즘을 이용하여 기존 소프트웨어의 패키지구조와 비교한다.[5]
(4) 그 결과를 토대로 (1)번에서의 모듈성 측정값과, 복원율 측정값의 비례여부를 조사한다.

3.4. 실 험

3.4.1. 각 데이터의 클래스를 그래프화하여 정점과 간선의 구조로 나타내었다.

표 2 클래스 의존 그래프의 각 패키지 정보

데이터	정점수	간선수	방향성
javax.swing	2021	5874	없음
java org	767	373	없음
com imageio	211	726	없음
java.util	772	1239	없음

각 데이터를 살펴보면, swing 패키지는 매우 많은 수의 정점과 간선의 수가 존재한다. 그러므로 알고리즘 수행 시간이 오래걸리는 반면 가장 신뢰성 있는 결과를 기대할 수 있다. util 패키지는 그보다 약간 소규모 패키지로써, 정점수가 간선수보다 적다. org패키지와 com패키지는 서로 대조적인 정점수와 간선수를 보여주고 있다.

위의 클래스 의존 그래프를 커뮤니티 검출 알고리즘을 통해 모듈화 하였다. modularity는 0부터 1까지의 값으로써, 각각의 커뮤니티가 얼마나 서로 독립적인지를 수치로 나타내어 준다. 만약 나눠진 모듈안의 모든 정점들이 서로 다른 모듈의 정점들과 연결되는 간선이 없다면 그 측정치는 1이 되며, 완전 그래프의 경우에는 측정치가 0이 된다. 아래의 표는 검출 알고리즘 시행 후, 그 결과이다.[7]

org, util 패키지는 상대적으로 높은 모듈성을 가진것과 달리 swing, imageio 패키지는 낮은 모듈

성을 보이고 있다. 이는 패키지의 노드 대비 간선수와 비례한다고 볼 수 있는데, 간선이 많다는 것은 그만큼 전체 클래스 의존성이 서로 얽혀 있어서 독립적인 모듈이 나오기 힘들다는 것을 의미한다. 유사실험 논문에서 모듈성 측정값을 0.5이상을 기준으로 좋은 결과라 생각하는 것을 참고했을때, 노드 대비 간선수가 비교적 적을수록 직관적으로 커뮤니티 독립성이 강하다는 점을 알 수 있다.[8] 또한, EB의 경우 알고리즘의 시간 복잡도가 높은 대신, MM에 비해 우수한 커뮤니티 검출 능력을 보여주는 것으로 실험 결과 나왔다.

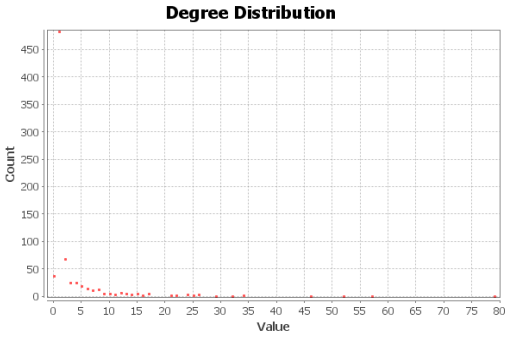

그림 1 java.util의 차수 분포

Value값이 차수이고, Count는 해당 차수를 가진 노드의 수를 의미한다.

표 3 데이터 모듈성 측정

데이터	MM	EB
javax.swing	0.494	0.598
java org	0.782	0.836
com imageio	0.407	0.505
java.util	0.692	0.842

3.4.2 클러스터링 계수는 그래프에서의 각 노드가 얼마나 강하게 밀집되어져 있는지를 나타내는 척도이다. 외부 모듈끼리의 독립성을 커뮤니티 모듈이 보장하였을때, 클러스터링

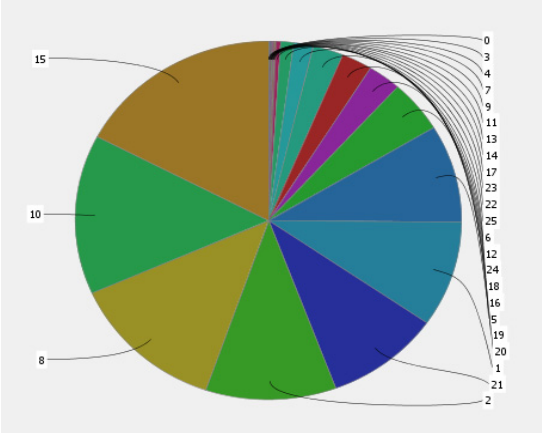

그림 2 MM-검출커뮤니티(swing)

계수가 높게 나온다면 직관적으로 커뮤니티 알고리즘 검출 결과가 아키텍쳐 모듈 뷰의 가치로써도 높을 거라 예상 할 수 있다. 반면에, 허브 노드의 존재로 인해 클러스터링 계수가 높아진다면 그로 인하여 커뮤니티 검출에 좋지 않은 영향을 끼친다고 볼 수 있다. 그런 이유에서 각 데이터의 클러스터링 계수를 구한다.

표 4 그래프 클러스터 일치성 측정(노드들의 평균값)

데이터	AVG.cluster coefficient	노드	간선
javax.swing	0.156(3211 triangle)	2021	5874
java org	0.091(28 triangle)	767	373
com imageio	0.182(264 triangle)	211	726
java.util	0.2(253 triangle)	772	1239

[표 4]의 결과에서 org 패키지의 클러스터링 계수의 수치가 매우 낮다. 이것은 노드 대비 간선의 수가 매우 적기 때문에 기본 클러스터링 계수측정 알고리즘의 트라이앵글을 만족하는 노드 집합생성 확률이 다른 데이터에 비해 적기 때문인 것으로 분석된다. 이 값이 낮다는 것은 반대로 커뮤니티 모듈성이 높아지는 결과를 야기한다는 것을 [표 3]과의 비교과정에서 알 수 있는데, 그래프 각 노드를

연결하는 간선이 밀집되지 않는다는 것이 커뮤니티를 나누었을 때 모듈의 크기에 따라 결과가 더 큰 독립성을 보장할 수 있기 때문이다.

3.4.3. 다음으로 클러스터링 기법과 (3.4.1.)번 실험에서 쓰여진 MM, EB의 모듈성 측정 결과를 기존 소프트웨어 패키지와 비교해 보겠다. 클러스터링 알고리즘으로는 WC를 썼다. 세 알고리즘의 모듈 복원율을 측정하는 알고리즘은 MoJo알고리즘(Move & Join 알고리즘)을 활용하였다. 이 알고리즘은 Move와 Join연산을 통하여 두 모듈간의 유사성을 0에서 100사이의 값으로 나타낸다. 아래 식은 이 알고리즘의 정의이다.[5]

$$Q(M) = (1 - \frac{MoJo(A,B)}{n}) \times 100\%$$

그림 3 Mojo FM

표 5 데이터 복원율 측정

(각 수치는 100%로 환산한 Mojo FM 측정치이다.)

데이터	MM	EB	WC
javax.swing	47.1	32.0	66.5
java org	66.2	31.5	58.6
com imageio	43.5	25.5	54.1
java.util	63.6	40.6	57.1

실험 결과 값을 종합 해 보았을 때, 같은 커뮤니티 검출 기법 중에서도 특히 modularity maximization의 경우 상당히 좋은 복원율을 보였다. 그에 반해 EB는 괄목할만한 성과가 나오지 않았으며, 그 이유는 MM이 모듈성을 높이기 위해 기존 클러스터링 알고리즘과 마찬가지로 노드들 사이의 연결 밀도를 중점적으로 높인 알고리즘이기 때문이다. EB는 알고리즘 기본 컨셉이 내부 노드의 강한 연결성 보다는 영향력이 큰 간선들을 잘라서 나누어진 커뮤니티이기 때문에 패키지 일치성 측면에서는 좋은 결과가 나오지 않았다고 생각해볼 수 있다.

(3.4.1.)번에서 실험한 커뮤니티 모듈성 측정값과 MoJo FM 값의 연관성을 살펴보겠다. MM의 경우엔 커뮤니티 모듈성 측정 방법과 일치성 측정값이 매우 유사한 패턴을 보였고, EB는 크게 일치하지는 않았으나, 모듈성 결과에 따라 근소한 차이를 보이는 것을 알 수 있었다. 이는 커뮤니티의 모듈성 수치가 패키지 복원율의 수치와 무관하지 않다라는 것을 말해준다.

5. 결 론

실험결과에 의하여 커뮤니티를 검출해 내는 특정 알고리즘에 한하여서는 상당히 흥미로운 결과가 나왔다. 커뮤니티 검출 결과에 따른 모듈성이 소프트웨어 아키텍쳐 복원 알고리즘의 복원율과 유사한 수치를 보였기 때문이다. 이 실험이 정확히 증명되기 위해서는 좀 더 정확하고 많은 숫자의 데이터를 통한 동일 실험이 선행되어야 할 것이다. 한편, 현재 각 커뮤니티 기법의 stability를 검증하는 실험 또한 진행 중에 있으며, 그 실험에 대해서는 stability 측정에 필요한 몇몇의 알고리즘과 도구의 구현이 필요할 것으로 보이기 때문에 개발 중에 있다. 커뮤니티 검출 기법이 아키텍쳐 모듈 복원 분야에서 어떠한 성능을 내는가에 관하여 특정 일치하는 부분을 찾는다면 클러스터링과 커뮤니티 검출의 융합이 가능할 것으로 보인다.

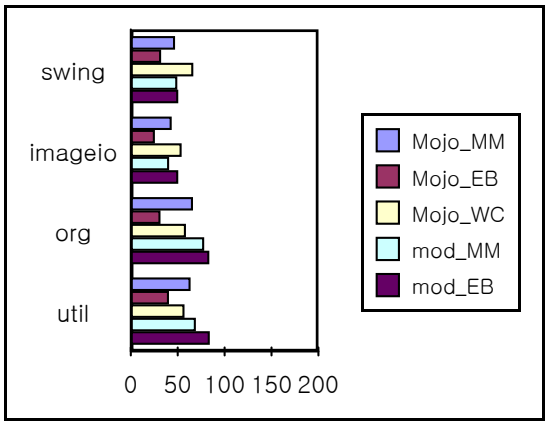

그림 4 각 실험 데이터 분포표

각 데이터의 MojoFm 측정 결과값과 모듈성 측정결과이다. 커뮤니티 모듈성과 데이터 복원율이 비례하는 것을 알 수 있다.

6. 참고문헌

[1] A.L. barabasi, "Link," 2002.

[2] S.Fortunato, "Community detection in graphs," Physics Reports, 2009.

[3] M.V. SteenGraph, Theory and Complex Networks, Maarten van Steen, 2010.

[4] L.Šubelj, M.Bajec, " Community structure of complex software systems: Analysis," Physica A: vol.390, pp.2968-2975, 2011.

[5] V.Tzerpos, R.C.Holt "MoJo, A Distance Metric for Software Clusterings," proc. WCRE, 1999.

[6] Network Work bench, "Weak Component Clustering", https://nwb.slis.indiana.edu/community/?n=AnalyzeData.WeakComponentClustering.

[7] M. E. J. Newman, "Detecting community structure in networks" Europian Physics Journal. B vol.38 pp.321-330, 2004.

[8] B. H. Good, Y. A. de Montjoye and A. Clauset, "The performance of modularity maximization in practical contexts," Physical Review. E pp.81, 2010.

저자소개

김 정 민

2013년 중앙대학교 컴퓨터공학부 학사.
2013년~현재 중앙대학교 컴퓨터공학과 석사과정.

<관심분야> 실시간 소프트웨어, 소프트웨어 클러스터링, 척도없는 네트워크

이 찬 근

1996년 중앙대학교 전자계산학과 학사
1998년 KAIST 전산학과 석사.
2005년 Univ. of Texas at Austin 전산학과 박사.
2005년~2007년 미국 인텔 소프트웨어 엔지니어.
2007~현재 중앙대학교 컴퓨터공학부 조교수.

<관심분야> 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅

회원 가입 안내 및 회비 납부 요령

한국정보과학회 소프트웨어공학소사이어티는 회원 여러분에게 유익한 정보를 제공해 드리기 위하여 보다 충실한 내용의 논문지 발간 배포, 그리고 국제·국내 학술발표회 및 초청강연회와 단기강좌 등의 여러 가지 사업들을 추진하고 있습니다.

소프트웨어공학 소사이어티의 가입을 통해 정보 및 기술 교류, 그리고 인적 네트워크의 구성에 참여하시기를 기대합니다. 회원 가입을 위하여 아래의 회비 안내를 참고하시어 회비를 납부하시고, 다음 쪽의 입회원서를 작성하시어 아래 소프트웨어공학소사이어티 주소로 보내주시거나 팩스 또는 이메일을 통해 보내어 주시기 바랍니다.

한국정보과학회 소프트웨어공학소사이어티 연락처는 아래와 같습니다.

◈ 소프트웨어공학소사이어티

주 소 : (우) 121-742 서울시 마포구 신수동 1번지, 서강대학교 신과학관 202호
한국정보과학회 소프트웨어공학소사이어티 박수용

전 화 : (02) 705-8928

팩 스 : (02) 704-8273

전자우편 : sypark@sogang.ac.kr (박수용교수), bjlee@uos.ac.kr (이병정교수)

홈페이지 : http://www.sigse-kiss.or.kr/

◈ 회비 안내

회원구분	• 학생회원 : IT 분야 학과 또는 관심 있는 학생 • 정회원, 종신회원 : IT 분야 종사자
가입비	학생회원, 정회원 : 20,000원, 종신회원 : 200,000원
년회비	학생회원, 정회원 : 20,000원

- 회비납부 방법

(1) 무통장입금 또는 계좌이체 후 입회원서 발송
: 계좌 번호 : 제일은행 150-20-358028 (이병정)

(2) 소사이어티 주관 학술행사 개최시, 행사장 당일 가입 및 납부 가능

개인회원용 입회원서

회원구분	학생회원 ( ) 정회원( ) 종신회원( )				
성명	한글		생년월일		
	영문				
연락처	직장전화		휴대전화		
	e-mail				
주소	직장명/부서		직급		
	직장주소	(우)			
학력	학사	년 월 - 년 월	대학교	과	
	석사	년 월 - 년 월	대학원	과	
	박사	년 월 - 년 월	대학원	과	
관심분야					

본인은 한국정보과학회 소프트웨어공학소사이어티의 취지에 찬성하여 회원으로 가입하고자 이에 입회원서를 제출합니다.

년 월 일

신 청 인: (인)

한국정보과학회 소프트웨어공학 소사이어티 회장 귀하

논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 4회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구논문과 산업계 논문을 게재해 오고 있습니다. 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제
- ○ 소프트웨어 설계 및 아키텍처
- ○ 소프트웨어 재사용 및 프로덕트라인
- ○ 요구공학
- ○ 소프트웨어 품질 및 테스트
- ○ 관리 및 프로세스
- ○ 소프트웨어 정형 기법
- ○ 서비스기반 소프트웨어 개발
- ○ 임베디드, 모바일, 웹기반 소프트웨어 개발
- ○ 기타 소프트웨어 응용 (국방, 자동차, 조선 등의 분야)

◈ 논문심사
- ○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
- ○ 심사료 및 게재료: 없음

◈ 논문 제출
- ○ 소프트웨어공학 소사이어티의 논문지 투고 양식(http://www.sigse-kiss.or.kr/)을 사용하며, 논문의 분량은 10장으로 제한합니다.
- ○ 논문지 투고규정에 따라 작성된 심사용 논문파일은 온라인투고시스템을 통하여 투고하시기 바랍니다.

◈ 문의처 (편집위원회)
- ○ 편집이사 : 강성원 교수 (KAIST, 042-350-3512, sungwon.kang@kaist.ac.kr)
- ○ 편집이사 : 윤회진 교수 (협성대학교, 031-299-0841, hjyoon@uhs.ac.kr)
- ○ 편집이사 : 이관우 교수 (한성대학교, 02-760-5864, kwlee@hansung.ac.kr)
- ○ 편집이사 : 채흥석 교수 (부산대학교, 051-510-3517, hschae@pusan.ac.kr)
- ○ 편집이사 : 김문주 교수 (KAIST, 042-350-3543, moonzoo@cs.kaist.ac.kr)
- ○ 편집위원 : 김정아 교수 (관동대학교, 033-649-7801, clara@kwandong.ac.kr)
- ○ 편집위원 : 김현수 교수 (충남대학교, 042-821-6657, hskim401@cnu.ac.kr)
- ○ 편집위원 : 이우진 교수 (경북대학교, 053-950-6378, woojin@mail.knu.ac.kr)
- ○ 편집위원 : 박수진 교수 (서강대학교, psjdream@sogang.ac.kr)
- ○ 편집위원 : 이지현 교수 (대전대학교, jihyun30@dju.ac.kr)
- ○ 편집위원 : 최종무 교수 (단국대학교, choijm@dankook.ac.kr)
- ○ 편집위원 : 김태호 박사 (ETRI, taehokim@etri.re.kr)

투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.
   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 A4용지에 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.
   [1]Cole, R., "Parallel Merge Sort", SIAM Journal of Computing, vol.17, No.4, pp.770-785, 1988.
   [2]김수형, 강명호, 조형제, 송주석. "안전하고 효율적인 침입자 역추적 시스템", 정보과학회논문지, 제25권, 제10호, pp.1123-1131, 1998

6. 논문은 소프트웨어공학 소사이어티(http://www.sigse-kiss.or.kr/)의 온라인 투고 시스템을 통해 제출한다.

7. 논문투고신청서를 반드시 작성하여 이메일(hjyoon@uhs.ac.kr)로 제출한다.

7. 원고 접수는 수시로 하며, 접수일은 온라인 접수일로 한다.

8. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

한국정보과학회 소프트웨어공학소사이어티 임원명단

구분		성 명	소속기관명	E-Mail
회 장		한 혁 수	상명대학교	hshan@smu.ac.kr
부회장 (기획)		권 기 현	경기대학교	khkwon@kyonggi.ac.kr
부회장 (편집)		강 성 원	KAIST	sungwon.kang@kaist.ac.kr
부회장 (학술)		홍 장 의	충북대학교	jehong@chungbuk.ac.kr
부회장 (조직)		이 병 걸	서울여자대학교	byongl@swu.ac.kr
부회장 (협력)		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
운영위원회	총무이사	이 정 원	아주대학교	jungwony@ajou.ac.kr
		유 준 범	건국대학교	jbyoo@konkuk.ac.kr
	기획이사	이 병 정	서울시립대학교	bjlee@uos.ac.kr
		서 주 영	아주대학교	jyseo@ajou.ac.kr
	조직이사	백 종 문	KAIST	jbaik@kaist.ac.kr
		김 영 철	홍익대학교	bob@hongik.ac.kr
	학술이사	인 호	고려대학교	hoh_in@korea.ac.kr
		고 인 영	KAIST	iko@kaist.ac.kr
	편집이사	윤 회 진	협성대학교	hjyoon@uhs.ac.kr
		이 관 우	한성대학교	kwlee@hansung.ac.kr
		채 흥 석	부산대학교	hschae@pusan.ac.kr
		김 문 주	KAIST	moonzoo@cs.kaist.ac.kr
	홍보이사	조 은 숙	서일대학교	escho@seoil.ac.kr
		이 찬 근	중앙대학교	cglee@cau.ac.kr
		박 용 범	단국대학교	ybpark@dankook.ac.kr
	협력이사	이 세 영	NIPA	sarahlee230@gmail.com
감사		차 성 덕	고려대학교	scha@korea.ac.kr
		이 상 은	NIPA	selee@nipa.kr
자문위원회		강 교 철	포항공과대학교	kck@postech.ac.kr
		권 용 래	KAIST	kwon@cs.kaist.ac.kr
		성 기 수	KISTI 고문	Kss13@truefriend.com
		배 두 환	KAIST	bae@se.kaist.ac.kr
		신 규 상	ETRI	gsshin@etri.re.kr
		양 승 민	숭실대학교	smyang@ssu.ac.kr
		우 치 수	서울대학교	wuchisu@selab.snu.ac.kr
		이 경 환	중앙대학교	kwlee@object.cau.ac.kr
		이 단 형	KAIST	danlee@cs.kaist.ac.kr
		정 기 원	숭실대학교	chong@comp.ssu.ac.kr
		박 수 용	서강대학교	sypark@sogang.ac.kr
		황 선 명	대전대학교	sunhwang@dju.kr
		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
		김 수 동	숭실대학교	sdkim777@gmail.com
		이 긍 해	항공대학교	khlee@kau.ac.kr
		최 병 주	이화여자대학교	bjchoi@ewha.ac.kr

구분	성 명	소속기관명	E-Mail
이사	김 정 아	관동대학교	clara@kwandong.ac.kr
	남 영 광	연세대학교	yknam@yonsei.ac.kr
	박 수 진	서강대학교	psjdream@sogang.ac.kr
	염 근 혁	부산대학교	yeom@pusan.ac.kr
	오 재 원	카톨릭대학교	jwoh@catholic.ac.kr
	윤 희 병	국방대학원	hbyoon37@hanmail.net
	이 우 진	경북대학교	woojin@knu.ac.kr
	이 은 석	성균관대학교	leees@skku.edu
	이 석 원	아주대학교	leesw@ajou.ac.kr
	이 은 주	경북대학교	ejlee@knu.ac.kr
	전 태 웅	고려대학교	jeon@korea.ac.kr
	정 인 상	한성대학교	insang@hansung.ac.kr
	최 승 훈	덕성여자대학교	csh@duksung.ac.kr
	최 종 무	단국대학교	choijm@dankook.ac.kr
	최 호 진	KAIST	hojinc@kaist.ac.kr
	현 창 문	탐라대학교	cmhyun@tnu.ac.kr
	계 승 교	삼성SDS	seankae@samsung.com
	권 경 룡	국방기술품질원	ka-ja17@hanmail.net
	권 원 일	STA컨설팅	wonil@softwaretesting.co.kr
	김 상 기	현대자동차	sangkikim@hyundai-motor.com
	김 진 태	SEEG	jtkim@swexpertgroup.com
	민 경 오	LG전자	davidmin@lge.com
	민 상 윤	솔루션링크	sang@sol-link.com
	박 복 남	핸디피엠지	pbnkhb@hitel.net
	박 찬 규	국방SW산학연합회	milspark@hotmail.com
	배 현 섭	슈어소프트테크	hsbae@suresofttech.com
	손 세 창	인천공항공사	scsohn@airport.kr
	손 진 규	삼성탈레스	Jinkyu.son@samsung.com
	신 석 규	SW시험인증센터	skshin@tta.or.kr
	양 상 욱	KAI	sangyang@koreaaero.com
	유 영 수	현대엠앤소프트	ysyoo@hyundai-mnsoft.com
	윤 태 권	한국SW기술진흥협회	tkyune@empal.com
	윤 형 진	케피코	Hyoungjin.yoon@kefico.co.kr
	이 근	삼성전자	gskeun.lee@samsung.com
	이 성 남	방위사업청	dapalee@korea.kr
	이 우 복	삼성전자	woobok.yi@samsung.com
	이 장 수	한국원자력연구원	jslee@kaeri.re.kr
	장 주 수	모아소프트	jsjang@moasoftware.co.kr
	정 연 대	N3SOFT	ydchung@n3soft.co.kr
	조 병 인	국방과학연구소	chobyun@dreamwiz.com
	조 상 윤	다한테크	sycho@dahan.co.kr

소프트웨어공학소사이어티 논문지 제25권 제4호 (통권 96호)

발 행 일 ॥ 2012년 12월 31일

발 행 인 ॥ 한혁수
편 집 인 ॥ 강성원

발 행 처 ॥ 사단법인 한국정보과학회 소프트웨어공학소사이어티
연 락 처 ॥ 서울특별시 종로구 홍지문 2길 20, 상명대학교 소프트웨어 대학관 G511
전 화 : 02-2287-5033, 팩 스 : 02-2287-0049
홈페이지 : http://www.sigse-kiss.or.kr/

인 쇄 처 ॥ (주)참기획 (전화 : 042-861-6380, 팩스 : 042-861-6381)