Journal of Korean Institute of Intelligent Systems, Vol. 23, No. 5, October 2013, pp. 385-391
http://dx.doi.org/10.5391/JKIIS.2013.23.5.385

빅데이터 특집 논문

빅 데이터의 MapReduce를 이용한 효율적인 병렬 유전자 알고리즘 기법

The Efficient Method of Parallel Genetic Algorithm using MapReduce of Big Data

홍성삼* · 한명묵*†

Sung-Sam Hong, and Myung-Mook Han†

*가천대학교 전자계산학과

† Department of Computer Engineering, Gachon University

요 약

빅 데이터는 일반적으로 사용되는 데이터 관리 시스템으로 데이터의 처리, 수집, 저장, 탐색, 분석을 할 수 없는 큰 규모의 데이터를 말한다. 빅 데이터 기술인 맵 리듀스(MapReduce)를 이용한 병렬 GA 연구는 Hadoop 분산처리환경을 이용하여, 맵 리듀스에서 GA를 수행함으로써 GA의 병렬처리를 쉽게 구현할 수 있다. 기존의 맵 리듀스를 이용한 GA들은 GA를 맵 리듀스에 적절히 변형하여 적용하였지만 잦은 데이터 입출력에 의한 수행시간 지연으로 우수한 성능을 보이지 못하였다. 본 논문에서는 기존의 맵 리듀스를 이용한 GA의 성능을 개선하기 위해, 맵과 리듀싱과정을 개선하여 맵 리듀스 특징을 이용한 새로운 MRPGA(MapReduce Parallel Genetic Algorithm)기법을 제안하였다. 기존의 PGA의 topology 구성과 migration 및 local search기법을 MRPGA에 적용하여 최적해를 찾을 수 있었다. 제안한 기법은 기존에 맵 리듀스 SGA에 비해 수렴속도가 1.5배 빠르며, sub-generation 반복횟수에 따라 최적해를 빠르게 찾을 수 있었다. 또한, MRPGA를 활용하여 빅 데이터 기술의 처리 및 분석 성능을 향상시킬 수 있다.

키워드 : 빅데이터, 하둡 프레임워크, 맵 리듀스, 병렬 유전자 알고리즘, 최적화

Abstract

Big Data is data of big size which is not processed, collected, stored, searched, analyzed by the existing database management system. The parallel genetic algorithm using the Hadoop for BigData technology is easily realized by implementing GA(Genetic Algorithm) using MapReduce in the Hadoop Distribution System. The previous study that the genetic algorithm using MapReduce is proposed suitable transforming for the GA by MapReduce. However, they did not show good performance because of frequently occurring data input and output. In this paper, we proposed the MRPGA(MapReduce Parallel Genetic Algorithm) using improvement Map and Reduce process and the parallel processing characteristic of MapReduce. The optimal solution can be found by using the topology, migration of parallel genetic algorithm and local search algorithm. The convergence speed of the proposal method is 1.5 times faster than that of the existing MapReduce SGA, and is the optimal solution can be found quickly by the number of sub-generation iteration. In addition, the MRPGA is able to improve the processing and analysis performance of Big Data technology.

Key Words : Big Data, Hadoop Framework, MapReduce, Parallel Genectic Algorithm, Optimization

1. 서 론

빅 데이터는 일반적으로 사용되는 소프트웨어 도구로 데이터의 처리, 수집, 저장, 탐색, 분석을 할 수 없는 큰 규모의 데이터를 말한다. 빅 데이터에 분석은 공학, 과학뿐만 아니라 기업, 정부 등 데이터를 수집하고 관리하는 모든 분야에서 관심사가 되고 있다[1]. 빅 데이터에서 주요 이슈는 데이터 규모, 데이터의 다양성, 데이터의 처리 속도를 효과적으로 대처하는 것이다. Hadoop은 이러한 빅 데이터를 저장, 관리, 처리, 분석하기 위한 도구로 잘 알려진 오픈 플랫폼이다. 빅 데이터에서는 처리속도가 아주 중요한 요소인데 Hadoop의 맵 리듀스는 빅 데이터를 처리, 검색, 분석을 효율적으로 하기 위한 병렬 분산 처리 프로그래밍으로 다양한 형태로 활용되고 있다.

유전자 알고리즘(GA : Genetic Algorithm)은 선택, 교차, 변이로 이루어진 유전자 오퍼레이터를 반복적으로 수행하면서 최적의 해를 얻는 알고리즘이다. 유전자 알고리즘은 자연계에서 진화과정의 기반인 적자생존을 원리에 기초하

접수일자: 2013년 8월 15일
심사(수정)일자: 2013년 9월 9일
게재확정일자 : 2013년 9월 15일
† Corresponding author
본 연구는 미래창조과학부가 지원한 2013년 정보통신·방송(ICT) 연구개발사업의 연구결과로 수행되었음.

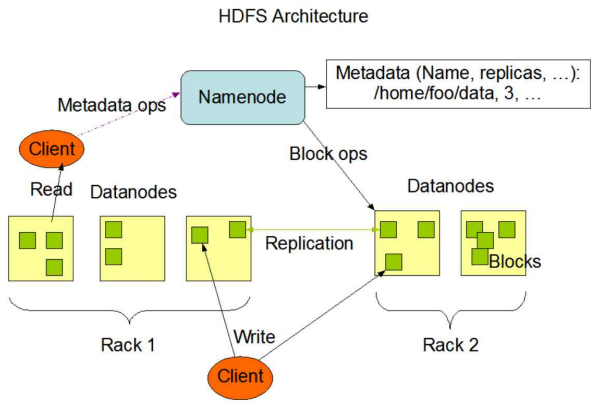

그림 1. 하둡 분산 파일 시스템
Fig. 1. Hadoop Distributed File System

여 최적의 해를 찾아가는 최적화 기법이다. 또한, 단일 점이 아닌 여러 점들의 집단에서 동시에 탐색을 하기 때문에 지역해(local optimal)가 아닌 전역 해(global optimal)를 찾을 수 있다. 이러한 장점 때문에 유전자 알고리즘은 퍼지 시스템, 신경망 등과 결합하여 다양한 분야에 적용되고 있다[2].

PGA기법[3]는 반복 수행이 많은 유전자 알고리즘의 한계를 극복하고, 지역해로 조기 수렴하는 것을 개선하기 위해 제안되었다. 하지만 기존의 PGA 방법들은 분산환경의 구성과 구현하는데 있어서 GA알고리즘을 분산환경[4, 5, 6, 7]에 맞게 재구현해야하기 때문에 비용과 노력을 많이 필요로 했다.

Hadoop기반의 맵 리듀스(MapReduce)를 이용한 병렬 GA 연구는 이러한 부분을 개선하여 기존의 Hadoop 분산환경을 활용하여, 맵 리듀스에서 GA를 수행함으로써 GA의 병렬처리를 쉽게 구현할 수 있었다. 이미 분산 환경을 구성하기 용이하게 개발된 Hadoop Framework와 병렬처리 성능이 우수한 맵 리듀스를 이용하여 GA 구현이 용이하였고, 기존의 다양한 GA 방법들을 큰 변형없이 맵과 리듀스에 적절히 구성할 수 있었다.

기존의 맵 리듀스를 이용한 GA들은 GA를 맵 리듀스에 적절히 변형하여 적용하였다[1]. 하지만 잦은 데이터 입출력에 의한 수행시간 지연과 기존의 PGA의 장점들을 살리지 못한 병렬 구조로 우수한 성능을 보이지 못하였다.

따라서 본 논문에서는 기존의 맵 리듀스를 이용한 GA의 성능을 개선하기 위해, Hadoop환경에서 맵 리듀스를 이용한 효율적인 PGA 기법을 제안한다. 제안한 MRPGA(MapReduce Parallel Genetic Algorithm)은 기존의 맵 리듀스 SGA(Scaling Simple Genetic Algorithm)방법에서 맵과 리듀싱과정을 개선하여 새로운 맵 리듀스 PGA 기법을 제안하였다. 또한 기존의 PGA의 topology 구성과 migration 기법을 MRPGA에 적용하여, 우수한 전역해를 찾을 수 있었다. 제안한 기법은 기존에 맵 리듀스 SGA에 비해 수렴속도가 1.5배 빠르며, sub-generation 반복횟수에 따라 우수한 최적해를 빠르게 찾을 수 있었다. 이를 통해 제안한 기법이 맵 리듀스 환경에서 효율적으로 병렬처리를 수행하여 우수한 성능을 나타낸다는 것을 입증할 수 있었다. 제안한 MRPGA는 최적화를 통해 빅 데이터의 처리에 성능을 향상시킬 수 있다.

본 논문의 구성은 2장에서 관련 연구에 대해 서술하며, 3장에서는 기존의 제안된 기법인 맵 리듀스 SGA기법에 대한 개요와 본 논문에서 제안하는 기법에 대해 자세하게 기술하였다. 4장에서는 제안하는 기법을 평가하기 위해 실험 및 결과 분석을 하였고, 5장에서 결론을 맺는다.

2. 관련연구

2.1 하둡 프레임워크

Hadoop은 맵 리듀스 기반 오픈 소스 소프트웨어 미들웨어로서 Yahoo, Facebook, Amazon, IBM, NexR 등 많은 기업들에서 클라우드 컴퓨팅 플랫폼으로 활용되고 있다[8]. Hadoop은 크게 분산 파일 시스템인 HDFS(Hadoop Distributed File System)와 분산 프로그래밍 모델인 맵 리듀스의 두 가지 구성요소로 이루어진다.

Hadoop의 맵 리듀스 응용은 클라이언트가 수행하는 작업단위인 잡(Job)으로 구성된다. 잡은 입력 데이터, 맵 리듀스 프로그램과 설정 정보로 구성된다. 또한 잡은 Map 태스크와 Reduce 태스크로 나누어 실행한다. 잡 실행 과정의 제어를 위해 하나의 잡 트래커(job tracker)와 다수의 태스크 트래커(task tracker)가 사용된다[9].

잡 트래커는 태스크 트래커들이 수행할 태스크를 스케줄링함으로써 시스템 전체에서 모든 잡이 수행되도록 조정한다. 태스크 트래커는 태스크를 수행하고 각 잡의 전체 결과를 잡트래커에 보낸다. 이때 태스크가 실패하면, 잡 트래커는 그것을 다른 태스크 트래커에 다시 스케줄링한다. 그림 1은 HDFS의 작동 흐름을 보인 것으로 클라이언트는 데이터 블록을 네임노드(Namenode)에 요청한다. 네임노드는 파일을 구성하는 데이터 블록의 메타데이터를 유지하며 클라이언트에 블록의 위치정보를 제공한다[10]. 이후 클라이언트는 파일에 대한 실제 연산을 수행하기 위해 특정 데이터노드(Datanode)에 접근하여 처리한다. Hadoop 클러스터 구성은 네임노드와 잡 트래커가 통합된 마스터 노드와 태스크 트래커와 데이터노드의 슬레이브 노드로 나눠진다. Hadoop에서 마스터와 슬레이브 노드 사이의 제어신호에는 RPC(Remote Procedure Call) 프로토콜이 사용되며, 마스터와 클라이언트 사이의 통신 역시 RPC가 사용된다. 그리고 데이터노드와 클라이언트는 TCP 소켓을 통해 데이터를 전달한다. 마지막으로 맵 리듀스 과정에서 태스크 트래커는 Map 태스크의 결과를 Reduce 태스크로 전달하게 되는데 Hadoop은 이들 간의 통신을 위해 HTTP를 사용한다.

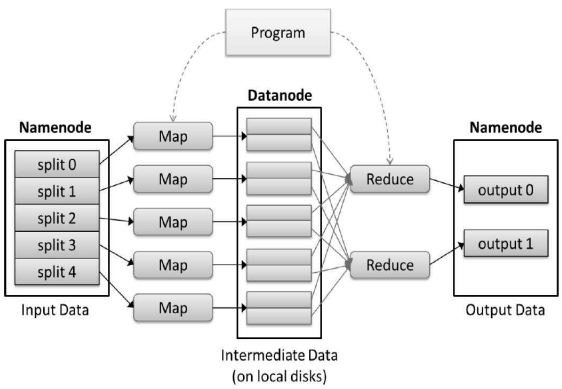

그림 2. MapReduce 구조
Fig. 2. The Structure of MapReduce

2.2 맵 리듀스

대규모 분산처리는 광범위하게 구성된 클러스터 환경에서 대용량 데이터를 분산 처리하는 기술이다. 대용량데이터 처리에 적합한 시스템을 위해 분산 파일 시스템, 분산 데이터 관리 시스템 등이 개발되고 있다. 광범위한 분산 처리는 대규모 데이터 처리에 적합한 데이터 웨어하우스, 데이터 마이닝, 정보 검색, 그리고 바이오 인포메틱스와 같은 분야에 유용하다[11].

분산 파일 시스템은 클라이언트 측에서 서버에 저장된 데이터에 접근하여 마치 자신에게 저장되어 있는 데이터인 것처럼 처리할 수 있는 클라이언트/서버 기반의 파일 시스템이다. 분산 파일 시스템은 기존의 분산 파일 시스템과 다르게 거대하고, 고성능이어야 한다. 이를 통해 지속적인 데이터 증가에 효율적으로 대비해야 하고, 빈번하게 발생되는 고장에 대해서도 대처가 가능한 매커니즘이 필요하다.

대표적인 분산 파일 시스템은 구글에서 개발한 GFS(Google File System)[12]과 GFS와 동일한 구조와 기능으로 개발된 HDFS[13]가 있다. HDFS는 오픈 소스 소프트웨어 개발 프로젝트인 Hadoop[14]에서 분산 컴퓨팅 프레임워크를 지원하기 위해 개발된 분산 파일 시스템이다. 현재 Amazon, IBM, Yahoo 등과 같은 기업들이 클라우드 컴퓨팅의 기반 플랫폼으로 널리 활용하고 있다. 대규모 데이터의 분산 병렬 처리를 위한 기술로 맵 리듀스가 있다. 구글에서 2004년도에 맵 리듀스 병렬 처리 시스템을 논문[8]로 공개하고, 구글의 다양한 서비스에 적용하였으며, 이를 기반으로 많은 업체에서 개발되었다. 기존의 MPI(Message Passing Interface)와 같은 병렬 처리 모델은 고성능의 컴퓨팅을 요구하는 분야에 적합한 반면 대규모의 데이터 처리에 적용할 경우에는 문제가 많았다. 맵 리듀스는 데이터양에 따른 확장성, 노드간 데이터 이동시 네트워크 트래픽 최소화 등의 요구사항을 고려하여 만들어졌으며, 현재 대규모 데이터 처리분야에서 맵 리듀스 병렬 처리 모델은 거의 표준이 되고 있다. 맵 리듀스는 그림 2와 같이 네임노드의 입력데이터는 맵 함수를 통해 각 컴퓨터의 로컬 디스크인 데이터 노드에 분산되고, 각 컴퓨터에 분산된 중간 데이터(intermediate data)는 정렬된 후, 리듀스 함수를 통해 수집되어 출력 데이터에 방출된다. 맵 리듀스에서는 Hash 자료구조와 유사한 (key, value)의 쌍으로 데이터를 처리 및 저장한다. 그리하여 쉽게 분산, 수집이 가능하고, 같은 키 그룹으로 정렬이 이루어진다.

3. MRPGA(MapReduce Parallel Genetic Algorithm)

3.1 Simple Scaling Genetic Algorithm using MapReduce

맵 리듀스를 이용한 SGA[15]에서는 맵과 리듀스에 맞게 변형된 GA 초기모델을 제시하였다. SGA 기법 중 Selecto-recombinative genetic algorithms[16], [17]의 단계를 맵과 리듀스 과정에 적절하게 모델링하여 구현하였다. 이 논문에서 맵 리듀스 데이터 흐름은 그림 3과 같다.

이 기법에서 맵에서는 fitness 계산을 수행하고, best fitness를 가진 individual을 저장한다. 리듀스에서는 GA의 연산인 선택과 교차를 수행하여, 새로운 offspring을 얻어내어 HDFS에 저장하여 다음 세대로 넘겨준다. 이렇게 한번의

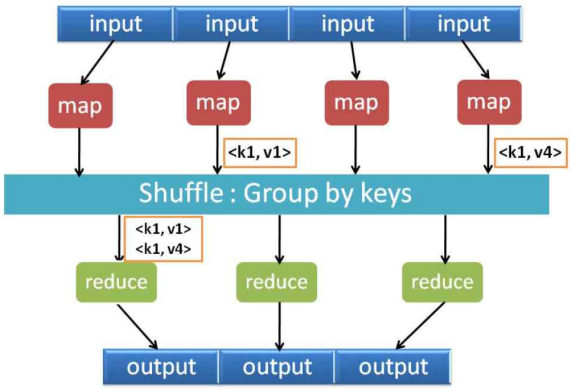

그림 3. MapReduce SGA 데이터 흐름
Fig. 3. Data Flow of MapReduce SGA

맵 리듀스과정을 거치면 한 세대가 끝나고, 정해진 세대수(반복횟수)에 따라 반복을 수행한다. 세대가 넘어갈 때 맵과 리듀스에서는 HDFS와의 데이터 입출력이 발생한다.

이 논문에서는 기존의 SGA의 구조를 쉽게 맵 리듀스에 적용할 수 있도록 알고리즘을 제안하였고, 이로 인해 SGA를 맵 리듀스에 맞게 변형하고, 대규모 분산처리를 통해 큰 크기의 문제를 해결할 수 있었다.

3.2 제안하는 MRPGA(MapReduce Parallel Genetic Algorithm) 기법

기존의 맵 리듀스를 이용한 SGA[15]는 맵에서는 데이터를 입력을 받아 Fitness 값 계산과 가장 우수한 전역해을 저장만 하고, 각 Reducer에서 GA의 연산을 분산처리하는 구조를 사용하고 있다. 맵에서 리듀스를 거치면 1번의 generation(세대)을 수행하게 되고, 정해진 세대수만큼 반복 수행하여 점점 최적해로 수렴한다. 여기서 Criteria는 반복횟수에 있다. 여기서 문제는 각 세대마다 HDFS에 의한 파일 입출력이 발생한다는 점이다. 맵에서는 HDFS에서 population을 입력을 받아 처리하여 리듀스로 전달하면, 리듀스에서도 작업을 수행하여 얻어진 데이터(여기서 데이터는 population의 각 individual 값들)를 다시 HDFS에 입력한다. 다음 세대으로 넘어가면 다시 맵에서는 저장된 데이터(population)을 HDFS로부터 입력받아서 처리한다. 이렇게 맵 리듀스마다 파일입출력이 발생하면, 반복이 많은 GA의 특성상 시간이 지연이 많이 되기 때문에 효율성이 떨어지고 오버헤드가 발생할 수 있다. 그리고 기존 SGA 방법의 경우 리듀스에서 GA를 수행하고 있는데, 맵 리듀스의 구조적 특성상 리듀스의 역할은 주로 데이터의 통계만 수행하고, 맵에서 데이터를 만들어서 보내준다. 따라서 맵에서 GA의 주요 연산을 수행하여 데이터를 만들어서 보내주면, 리듀스에서는 계산 및 통계적인 작업만 수행하는 것이 효율적이다.

또한 기존의 PGA 알고리즘들의 장점인[3] subpopulation 별 세대 반복과 migration을 수행할 수 없으며 subpopulation의 topology 구성간의 특성을 얻기가 어렵다. migration 방법들은 이미 기존의 많은 PGA 연구 논문으로부터 그 GA의 diversity를 보장할 수 있는 장점들이 입증되었다.

따라서 본 논문에서 이 점을 개선하기 위해 맵 리듀스를 이용한 새로운 PGA 기법인 MRPGA기법을 제안하고자 한다. 제안하고자 하는 MRPGA 구조는 다음 그림 4와 같다.

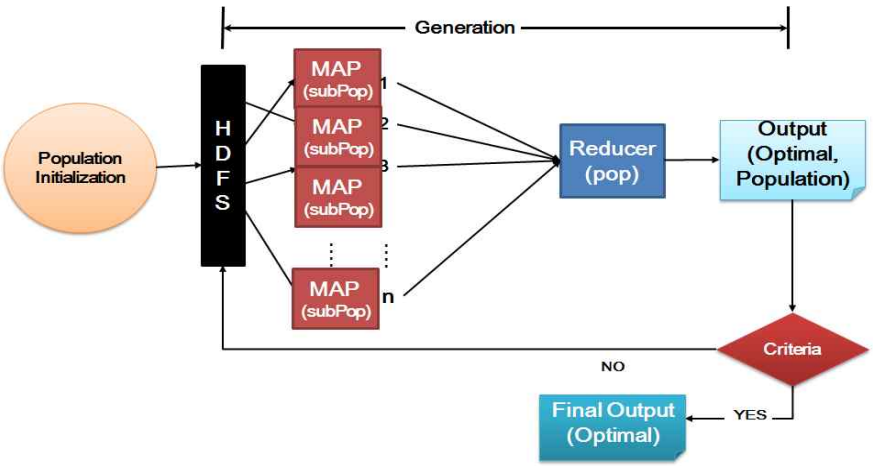

그림 4. 제안하는 MRPGA의 구조
Fig. 4. The Structure of proposal MRPGA

앞서 SGA논문에서 사용한 Selecto-recombinative genetic algorithms을 응용 및 개선하여 selection과 crossover를 연계한 local search 형태의 맵 구조를 제안하고, 우수한 individual을 reduce에 전달하면서 맵 리듀스 구조를 이용한 migration을 적용하였다. 과정을 요약하면 아래와 같다.

단계 1 : population을 초기화 한다.
단계 2 : population의 모든 individual에 대하여 fitness 값을 계산한다.
단계 3 : 각 sub-population마다 우수한 해를 선택하여, 교차를 수행한다. offspring은 다시 sub-population의 individual로 replace되고, 해당 individual들의 fitness만 다시 계산한다.
단계 4 : 3의 과정을 정해진 각 sub-generation주기만큼 반복한다. 마지막 sub-generation에서 선택된 우수한 해들을 sub-population 간에 random으로 migration한다.
단계 5 : 모든 individual에 대하여 fitness를 계상한다.
단계 6 : 정해진 generation(반복)만큼 3-5과정을 반복수행한다.

제안한 방법은 맵 리듀싱과정에서의 sub-generation을 수행하여 파일 입출력 시간을 최소화함으로써 속도를 개선하였다. 또한 맵과정에서 local search를 수행함으로써 우수한 최적해를 빠르게 찾도록 하였다.

기존의 PGA 방법들의 장점을 활용하여 싱글 리듀싱을 통해 각 서브 Population의 값을 통합하고 근사해를 탐색하며, 맵으로의 재입력으로 값들을 섞음으로써 다양성을 유지, 수렴속도를 향상시켰다. 맵에서 데이터를 만들고, 리듀스에서는 계산과 통계과정만 거침으로써 맵 리듀스의 구조의 활용도를 극대화할 수 있었다.다만, mapper의 개수가 적으면 기존의 맵 리듀스 PGA 방법들에 비해 병렬처리효율이 떨어져서 전반적인 효율성이 급격히 떨어질 수 있다.

3.2.1 맵 과정

먼저 Population을 초기화하고 Fitness를 계산하여 데이터를 구성하고 이를 HDFS에 저장한다. 다수의 맵에서 이 데이터(population)를 입력을 받아 GA의 연산자인 선택, 교차를 수행한다. 이 과정에서 정해진 mapper의 개수에 따라 subpopulation이 구성되며, 각 subpopulation별로 모든 individual에 대해서 GA의 연산자를 수행한다. 여기서 선택 방법은 tourment selection 방법을 사용하고, 교차는 uniform crossover[18]을 사용한다. 맵의 자세한 과정은 표 1에 수도코드에 나타나 있다.각 맵에서는 정해진 주기만큼의 sub-generation을 수행한다. 선택, 교차를 통해 발생한 offspring을 다시 sub-population으로 replacement하여 local search를 통한 우성의 최적해를 빠르게 얻을 수 있도록 한다. migration 과정은 맵 리듀스의 구조를 활용하여 구성한다. sub-generation의 마지막 반복세대에서는 선택된 우수한 individual들을 emit하여, partitioner를 통해 reduce에 전달되면서 shuffling이 된다. reduce가 HDFS로 모든 individual을 출력하고, 다음 generation에서 HDFS는 맵으로 이 데이터들을 입력함으로써 각 sub-population으로는 우수한 individual들이 migration된다. 이를 통해 좀 더 빠르게 최적해를 탐색할 수 있으며, 각 sub-population의 local convergence를 줄이고 문제공간의 다양성을 유지할 수 있다. 총 population의 수는 (맵의 개수 * sus-population크기)에 따라 결정된다.

3.2.2 리듀스 과정

Reducer는 단일 Reducer로 구성하여, 맵에서 전달되는 값 가장 우수한 값(해)만을 찾는데 사용한다. Reducer의 결과 값이 정해진 기준에 도달하지 않거나 정해진 generation 만큼 반복하지 않으면, 다시 맵 리듀스를 반복 수행함으로써 기존의 PGA의 Migration 형태의 방법으로 다양성을 유지한다. 리듀스에서는 계산 및 통계만 수행하기 때문에 간단하게 구성되어 있으며, 슈도코드는 표 2와 같다.

4. 실험 및 평가

본 논문에서는 제안하는 기법을 평가하기 위해 Hadoop Framework 환경에서 맵 리듀스를 이용한 Simple Scaling Genetic Algorithm과 제안하는 기법을 실험한다.

표 1. 맵 과정
Table 1. Map procedure

Algorithm 1 mapreducePGA Map procedure
seleted [tournamentSize]
MAP(key, value):

```
sub_pop <- Individual(key)
fitness <- value
tSize <- tournamentSize

for i  to sub-generation do
  for j to sub-population_size do
    selected[j % tSize] = Selection(fitness,tSize);
    if j % tSize == 0 then
      for k to tSize
        tmp_ind[j % tSize] = sub_pop[selected[j %
        tSize]]);
        tmp_ind[j % tSize+1] = sub_pop [selected[j %
        tSize+1]
      end for
      crossover(tmp_ind);
    end if

   if last iteration of sub-generation then
     emit(tmp_ind, fitness)
   else
     sub_pop <-tmp_ind
     fitness <- fitness(tmp_ind)
   end if

  end for
end for
```

4.1 실험환경 및 최적화문제

실험에 사용된 Hadoop 버전은 1.2.0 버전이며, OS는 Ubuntu 12.0.4 버전이다. CPU는 Intel Core(TM) i5 650(3.20GHz)이며, RAM은 3GB이다. 실험에 사용된 Hadoop 모드는 단일 PC에서 가상의 분산환경을 구성할 수 있는 pseudo-distribution mode를 활용하여 병렬처리를 수행한다.

최적해를 찾기위한 문제는 널리 사용되고 있는 간단한 최적화 문제인 OneMax Problem[19]를 사용하였다. 이 문제는 $\vec{x} = x_1, x_2, \ldots, x_N$, $x_i \in 0,1$일 때 fitness값이 최대인 값을 찾는 문제로 $F(\vec{x})$는 다음과 같다.

$$F(\vec{x}) = \sum_{i=0}^{N} x_i \qquad (1)$$

4.2 실험 및 평가

4.2.1 convergence 성능 평가

본 실험에서 제안하는 알고리즘이 최적해로 수렴하는 시간을 알아보았다. 10^2 variable의 individual 크기로 4개의 mapper을 사용하였고, 총 population size는 400개로 sub-generation은 10회, 실험결과는 아래 그림 5와 같다.

실험 결과를 살펴보면 약 53 세대부터 최적해에 근접하여 수렴하는 것을 확인할 수 있었다. 각 세대별 수행시간은 평균 20.625초가 걸리기 때문에 결과적으로 약 1000초정도 진행되면 수렴하는 것을 볼 수 있었다. 맵 리듀스 SGA 기법과 비교해보면 그림 6과 같다. SGA의 경우 유사한 조건에서 약 82세대부터 수렴을 시작하는 것을 볼 수 있다. 맵 리듀스 SGA도 수행시간을 측정한 결과 한 세대별 수행시간은 두 기법이 소요시간이 약 20초로 유사한 것을 확인할 수 있었다. 이를 통해 제안하는 MRPGA 기법이 맵 리듀스 SGA 기법에 비해 164%(약 1.5배)의 우수한 성능(속도)으로 전역 최적해를 빠르게 탐색하여 수렴한다고 판단할 수 있다. 따라서, MRPGA 기법이 기존의 방법보다 최적해를 찾는 성능이 우수하다는 것을 알 수 있다.

표 2. 리듀스 과정
Table 2. Reduce procedure

Algorithm 2 mapreducePGA Reduce procedure
REDUCE(key, values):

```
while values has a next value do
  individual = key;
  nowfit = fitness(individual);

  if nowfit > bestFitness then
    bestFitness <- nowfit
    bestIndividual <- individual
  end if

emit(key, nowfit)

end while
```

4.2.2 문제영역 확장 및 mapper 확장에 따른 성능 평가

MRPGA의 처리능력과 확장성을 분석하기 위해 먼저 각 맵의 sub-population을 100개로 고정하고, individual당 유전자 개수를 100개로 고정하였다. 그리고 mapper의 개수를 늘려 문제영역을 확장하여 소요되는 시간을 측정하였다. 문제 영역이 mapper에 개수에 따라 확장되면서 population의 다양성 보장과 우수한 글로벌 최적해를 찾을 수 있다. 하지만 그림 7과 같이 mapper개수가 증가함에 따라 선형으로 한 세대당 소요되는 시간이 증가하기 때문에 해결하고자하는 문제의 크기와 보유한 자원(node)의 개수의 따라 적절하게 mapper를 확장하여 사용해야한다.

4.2.3 sub-generation 횟수 성능 평가

본 실험에서는 제안하는 기법의 각 sub-population에서의 sub-generation 횟수에 따른 성능 분석을 하였다. sub-generation 반복에 의해 제안하는 기법에서는 각 mapper에서 local search를 수행하게 되어 좀 더 빠르게 전역 최적해를 탐색할 수 있도록 하였다. 따라서 반복회수 증

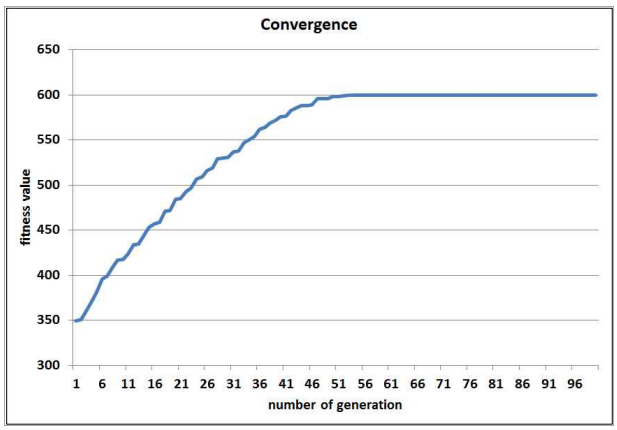

그림 5. 제안하는 MRPGA 기법의 convergence 결과
Fig. 5. The Result of proposal MRPGA convergence

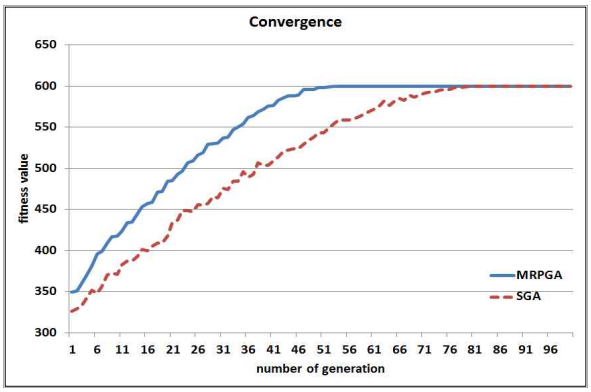

그림 6. MRPGA와 SGA의 convergence 결과
Fig. 6. The Convergence result of MRPGA and SGA

가에 따른 GA의 성능(수렴 속도)을 평가하였다. 결과는 그림 8과 같다. 실험은 총 population크기는 400개, individual 당 유전자 개수를 100개로 하였고 mapper의 개수는 4개이다. 각 1회, 10회, 50회의 sub-generation을 수행하여 fitness 값을 계산하여 수렴속도를 분석하였다.

1회에서는 약 70세대에서 수렴을 시작하였고 완전한 최적해로 수렴하지는 못하였다. 10회의 경우 약 53세대부터 수렴을 시작하여 최적해로 수렴하였다. 50회의 경우 약 50세대부터 수렴을 시작하여 최적해로 수렴하였다. 따라서, sub-population별 반복에 따라 local search를 수행하는 것이 GA의 성능을 개선시켜주는 것을 확인할 수 있었다.

두 번째 실험으로 sub-generation 횟수별 1세대당 소요시간을 측정하였다. 이는 반복횟수에 따른 overhead가 발생하는 것을 분석하기 위해서다. 결과는 그림 9와 같다. 결과에서 확인할 수 있듯이 1회, 10회, 50회는 각 세대별 수행시간이 20.13, 19.88, 20.21로 근소한 차이를 나타내고 있다. 이는 반복횟수에 따른 overhead가 거의 나타나지 않는다고 판단할 수 있다.

두 가지 실험을 통해 제안하는 기법은 반복을 하지 않는 것에 비해 약 140%(1.4배)정도의 빠른 속도로 수렴하였고, 100회의 generation 후 얻은 최적해도 1회는 fitness값이 598, 10회와 50회는 fitness 값이 600으로 더 우수한 전역 최적해를 얻는 것을 확인할 수 있었다. 따라서 제안하는 기법은 overhead가 거의 없이 속도와 탐색(수렴) 성능에서 우

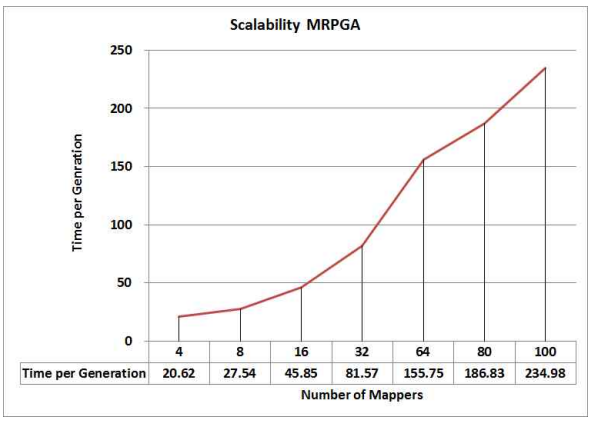

그림 7. 문제영역확장성에 따른 수행시간 결과
Fig. 7. The Result of time performance by the scalability of problem

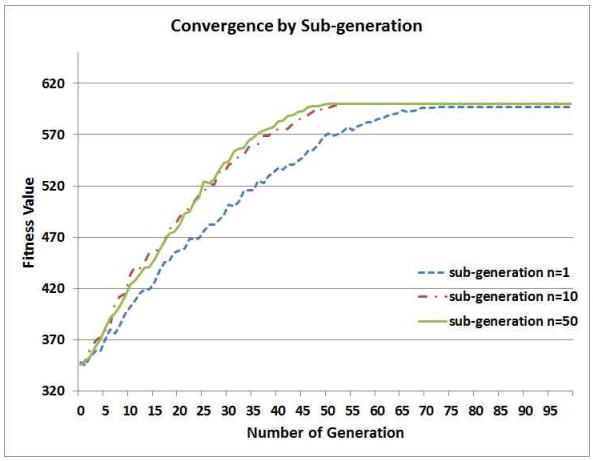

그림 8. sub-generation 수에 따른 convergence 결과
Fig. 8. The Result of convergence by sub-generation

수한 성능을 나타내는 것을 확인할 수 있었다. 다만, sub-generation의 횟수가 너무 클 경우 local search에 의해 GA가 조기 수렴할 수 있다.

5. 결론 및 향후연구

본 논문에서는 Hadoop환경으로부터 분산환경을 구축하여, 맵 리듀스를 통해 PGA를 구현하는 방법을 제안하였다. 또한 기존의 방법들을 개선하기 위해 새로운 맵 리듀스 PGA 방법을 제시하였다. I/O 횟수를 줄여 PGA의 속도를 개선하였고, 싱글 리듀싱을 통해 탐색영역의 통합과 분산 맵으로 유전자의 다양성을 유지할 수 있었다. 또한 sub-population별로 반복수행을 통한 local search로 PGA의 성능을 개선하였고, 맵 리듀스의 특성을 이용해 효율적인 migration을 구현하였다.

본 논문을 통해 효율적인 맵 리듀스 병렬 유전자 알고리즘 기법을 제안하였고, 실험을 통해 우수한 성능을 증명하였다. 추후 연구로는 population 초기화 방법을 통해 PGA의 성능을 개선하고, 병렬처리를 극대화하기 위해 효율적인

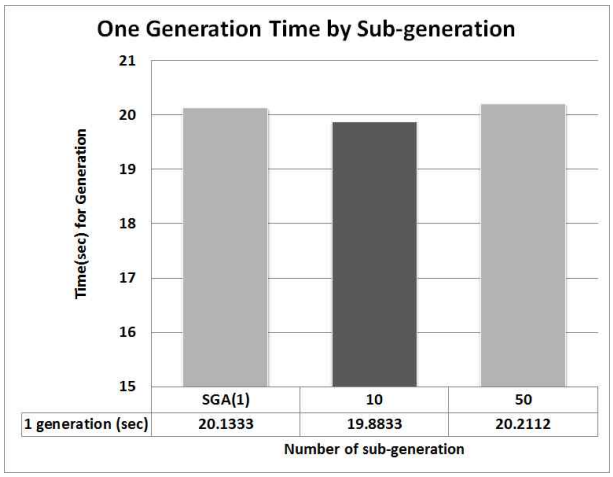

그림 9. sub-generation 수에 따른 수행시간 결과
Fig. 9. The result of time performance by sub-generation

다중 리듀싱 방법 연구와 싱글 리듀서에서 Local Search 및 휴리스틱 알고리즘을 환경에 맞게 적용하여 하이브리드 형태의 맵 리듀스 PGA를 연구하고자 한다. 또한, 제안하는 기법을 이용하여 빅 데이터 기술의 성능을 향상시키는 연구를 하고자 한다.

References

[1] Young Jun Kim1, Kyung Soon Hwang and Keon Myung Lee, "A MapReduce-based Algorithm for Semantic Hashing with Extended Boundaries," *Proceedings of KIIS Fall Conference*, Vol. 22, No. 2, 2012

[2] Chong-Ho Yi and Dong W. Kim, "Comparisons of Robot-Moving Strategies with Evolutionary Algorithm and Neuro-Fuzzy Method", *Journal of KIIT*, Vol. 10, No. 2, pp. 227-232, 2012..

[3] E. Cantú-Paz, "A survey of parallel genetic algorithms," Calculateurs Parallèles, *Réseaux et Systèmes Répartis*, Vol. 10, No. 2, pp. 141 - 171, 1998.

[4] Jae Hoon Cho , Dae-Jong Lee , Jin-Il Park , Myung-Geun Chun, "Hybrid Feature Selection Using Genetic Algorithm and Information Theory," *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS*, Vol.13 No.1, pp 69-77, 2013

[5] Myung-Mook Han, "Parallel Genetic Algorithm based on a Multiprocessor System FIN and Its Application to a Classifier Machine," *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS*, Vol.8, No5, pp 61-71, 1998

[6] Dongho Song, Yougil Lee, Tae-Hyoung Kim, "A Study on Distributed Particle Swarm Optimization Algorithm with Quantum-infusion Mechanism," *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT*, Vol.22, No.4, pp 527-531, 2012

[7] Seung-Hyung Jung, Jeoung-Nae Chi, Sung-Kwun Oh, Hyun-ki Kim, "Design of Optimized Fuzzy PD Cascade Controller Based on Parallel Genetic Algorithms," *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT*, Vol.19, No.3, pp 329-336, 2009

[8] Dean, J. and Ghemawat, S., "MapReduce: Simplified Data Processing on Large Clusters," *Communication of the ACM*, Vol. 51, No. 1, pp.107-113, 2008.

[9] Yun-Hee Kang amd Myoung-Woo Hong, "Sensory Data Processing by Using Hadoop Framework," *Journal of KIIT*, Vol. 11, No.2, pp 169-174 , 2013

[10] Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li, and Ying Li, "Hadoop high availabilitythrough metadata replication", *Proceeding of the first International Workshop on Cloud datamanagement*, pp. 37-44, 2009.

[11] Suan Lee and Jinho Kim, "Sort-Based Distributed Parallel Data Cube Computation Algorithm using MapReduce," *Journal of The Institute of Electronics Engineers of Korea*, Vol. 49, NO. 9, 2012.

[12] Ghemawat, S., Gobioff, H., and Leung, S. T.,"The Google File System," *In Proc. 19th on Operating Systems Principles*, pp. 29-43, 2003.

[13] HDFS, http://hadoop.apache.org/hdfs/.

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

[15] Abhishek Verma, Xavier Llor`a, David E. Goldberg, and Roy H. Campbell., "Scaling Genetic Algorithms Using MapReduce," *In Proceedings of the Ninth International Conference on Intelligent Systems Design andApplications (ISDA)*, 2009.

[16] D. E. Goldberg. *Genetic algorithms in search, optimization,and machine learning*. Addison-Wesley, Reading, MA, 1989.

[17] D. E. Goldberg. T*he Design of Innovation: Lessons fromand for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.

[18] G. Sywerda., "Uniform crossover in genetic algorithms," *Proceedings of the third international conference on Genetic algorithms*, pages 2 - 9, San 1989.

[19] J. Schaffer and L. Eshelman., "On Crossover as an Evolutionary Viable Strategy," *Proceedings of the 4th International Conference on Genetic Algorithms*, pp 61 - 68, 1991.

저 자 소 개

홍성삼(Sung-Sam Hong)

2009년 : 경원대학교 전자거래학과 졸업
2011년 : 경원대학교 전자계산학과 석사과정 졸업
2011년 2월~현재 : 가천대학교 전자계산학과 박사과정

관심분야 : Security, Network, Optimization, Algorithm
Phone : +82-31-750-5818
E-mail : sungsamhong@gmail.com

한명묵(Myung-Mook Han)

1980년 : 연세대학교 공과대학 졸업 (공학사)
1987년 : 뉴욕공과대학교 컴퓨터공학과 석사 졸업 (공학석사)
1997년 : 오사카시립대학교 정보공학부 졸업(공학박사)
1998년~현재 : 가천대학교 IT대학 교수
1998년~ 현재 : 한국지능시스템학회 이사

관심분야 : Security, Algorithm, Data Mining
Phone : +82-31-750-5522
E-mail : mmhan@gachon.ac.kr