분산 환경에서 클러스터 노드 할당 시스템을 위한 유전자 기반 최적화 모델

박 경 모†

요 약

본 논문에서는 분산 컴퓨팅 환경에서 클러스터 노드 할당 시스템에 대한 최적화 모델을 제시한다. 분산 파일 시스템 구조를 지닌 제시 모델에서는 시간에 따른 시스템의 역동적인 움직임을 면밀하게 고려하여 클러스터 노드 할당 세트가 타당한지를 조사하는 클러스터 모니터 노드의 기능이 주어진다. 노드 할당 시스템의 클러스터 모니터 노드는 병렬 모듈들을 클러스터 노드들에 분산시키면서 유전 알고리즘을 이용하여 좋은 할당 솔루션을 제공한다. 실험적 연구의 일환으로 코딩 기법, 교배, 돌연변이, 개체집단 크기 같은 다양한 유전 인자 파라미터와 노드 모듈 개수에 따른 솔루션 품질 및 계산 시간에 관한 비교 실험 결과를 발표한다.

A Genetic-Based Optimization Model for Clustered Node Allocation System in a Distributed Environment

Kyeongmo Park†

ABSTRACT

In this paper, an optimization model for the clustered node allocation systems in the distributed computing environment is presented. In the presented model with a distributed file system framework, the dynamics of system behavior over times is carefully thought over the nodes and hence the functionality of the cluster monitor node to check the feasibility of the current set of clustered node allocation is given. The cluster monitor node of the node allocation system capable of distributing the parallel modules to clustered nodes provides a good allocation solution using Genetic Algorithms (GA). As a part of the experimental studies, the solution quality and computation time effects of varying GA experimental parameters, such as the encoding scheme, the genetic operators (crossover, mutations), the population size, and the number of node modules, and the comparative findings are presented.

키워드 : 분산 컴퓨팅 시스템, 클러스터(Cluster), 유전 알고리즘(Genetic Algorithms)

1. 서 론

최근 주목받고 있는 클러스터(cluster)[9] 기술을 이용한 분산 컴퓨팅은 여러 특징과 장점이 있다. 클러스터란 고성능 네트워크나 LAN으로 연 결된 컴퓨터 노드들의 집합을 의미한다. 컴퓨터 노드는 SMP(Symmetric Multi-Processor) 서버, 워크스테이션, 또는 개인용 컴퓨터(PC)가 될 수 있다. 각각의 노드는 기존 사용했던 것처럼 개별적인 사용이 가능하고 클러스터 노드들은 하나의 통합된 컴퓨팅 자원으로 공동 작업을 수행할 수 있다. 클러스터에서는 표준화된 통신망 프로토콜로써 상호 연결된 컴퓨터 노드들을 묶는 단일(single) 시스템으로 집단적 작업을 하면서 사용 중 끊기지 않고 효율적인 성능을 제공한다. 대형 병렬 컴퓨터에서의 계산은 고비용으로 사용하기가 어려운 반면 워크스테이션 클러스터로 구성된 분산 컴퓨팅 환경에서는 저비용으로 손쉽게 사용할 수 있다. 분산 클러스터 시스템에서 유휴 상태의 부하가 적은 컴퓨터 노드에게 프로그램 작업을 유효적절하게 할당시켜 프로그램의 병렬 수행이 가능하다. 즉 대기 상태로 있는 워크스테이션을 계산 노드로 이용할 수 있고 워크스테이션 중의 하나를 파일서버(file server)로 사용할 수 있어 대규모 컴퓨팅 시스템인 슈퍼컴퓨터에 버금가는 처리율(throughput)을 얻을 수 있다[9, 10].

분산 시스템은 여러 독립 분리된 컴퓨터 노드들의 집합체이지만 사용자에게는 하나의 컴퓨터를 사용하는 것처럼 환영을 만들어 사용자는 하나의 시분할 시스템으로 생각한다. 이런 분산 환경을 위해 프로세스 관리는 어떤 노드에서나 같은 방법으로 이루어져야 한다. 프로세스가 생성되고 파괴되는 방법과 시작되고 멈추는 방법이 각각의 노드마다

※ 본 연구는 2001년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.
† 정 회 원 : 가톨릭대학교 컴퓨터정보공학부 교수
논문접수 : 2001년 12월 31일, 심사완료 : 2002년 12월 2일

달라서는 안 된다. 하나의 시스템 호출이 모든 컴퓨터에 가능해야 하며 분산 환경에서 공통적으로 사용되도록 되어있다. 어디에서나 같은 시스템 호출 인터페이스를 갖는 분산 시스템은 개별적인 커널들이 시스템의 모든 CPU들을 운영할 수 있게 되고 전역적으로 실행되어야 하는 명령들을 제어하기가 쉬어진다. 예를 들어 프로세스가 시작될 때 모든 커널들은 그것을 실행하기에 가장 적합한 장소를 발견하기 위해 서로 협조해야 하고 전체적인 분산 파일 시스템이 필요하다.

분산 파일 시스템은 물리적으로 여러 노드들 간에 산재하고 있는 파일들에 대하여 통일된 파일 공유 방식을 지원하는 시스템이다. 파일서버와 클라이언트로 구성된 분산 파일 시스템[2]에서는 다수의 사용자들이 파일과 기억장치 자원들을 공유하게 된다. 이런 시스템에서 파일서버는 공유자원을 관리하고 사용자 이동성을 제공하여 백업 및 복구를 관리하는 역할을 맡게 된다. 분산 파일 시스템은 분산 시스템 상의 서버와 클라이언트, 기억장치들이 여러 지역에 흩어져 있는 형태를 갖는다. 따라서 서비스는 분산 시스템간의 네트워크를 통하여 수행되며 각각의 데이터는 독립된 기억장치들에 나누어 저장된다. 사용자들은 서로 다른 시스템으로 로그인하여 특정한 프로그램 실행을 호출할 수 있다. 분산 시스템의 신뢰성 관련하여 작업의 부하를 여러 노드들에 분산시키기 때문에 한 노드의 고장으로 나머지 시스템들에 영향이 거의 미치지 않는다. 분산 시스템 내에 서버와 클라이언트가 복수로 존재하며 각자가 자치적인 시스템 운영이 가능하다. 시스템 수행에 있어 파일 복사본이나 중복 데이터 파일을 사용한다.

본 논문에서 기술하는 클러스터 할당 시스템은 상기 언급한 분산 파일 시스템에 기반을 두고 있다. 사용 가능한 시스템 자원을 최대로 잘 이용하면서 분산처리에 따른 반환 시간 단축으로 향상된 성능을 얻을 수 있는 전역적 할당 기법 연구에 초점을 둔다. 제시하는 할당 방법은 다음과 같이 두 단계로 나누어 진행된다.

① 프로그램 모듈(작업)을 클러스터간의 상호 전송되는 메시지의 통신 트래픽(traffic)을 최소화시킬 수 있는 클러스터에 할당한다.
② 사용 가능한 클러스터내의 노드들에 태스크를 할당하되 부하균형을 고려하여 프로그램 작업부하(workload)를 동등하게 배분시킨다.

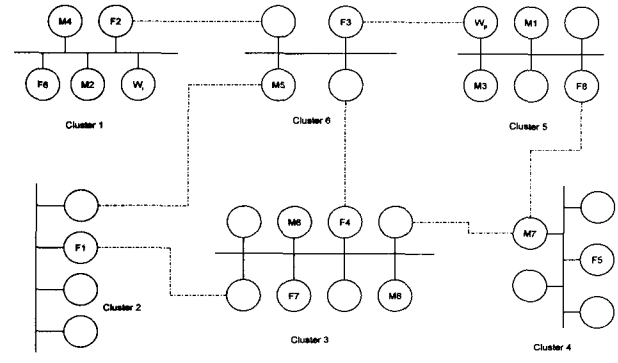
전역 할당 기법을 이용하면 클러스터간의 트래픽이 적을 뿐 아니라 CPU 유휴 사이클을 더 많이 얻을 수 있어 반환 시간을 단축시키는 효과를 얻을 수 있다. 이로 인하여 보다 나은 자원 사용도 가능해진다. 이러한 긍정적인 결과를 얻기 위해서 본 논문에서는 분산 컴퓨팅 환경에서 병렬 프로그램 실행이 가능한 클러스터 작업 할당 최적화 모델에 대하여 연구한다.

본 논문의 구성은 다음과 같다. 2장에서는 분산 컴퓨팅 환경에서 클러스터 작업 할당 문제를 설명하고 클러스터 노드 할당 시스템의 모델과 표기법 정의를 제시한다. 3장은 제안된 모델에 기초하여 시스템 설계 구현 시 유의 사항과 최적화 기법으로 사용된 유전 알고리즘을 설명하고 목적함수 모델을 기술한다. 실험적 연구의 일환으로 코딩 기법, 교배, 돌연변이, 개체집단 크기 같은 다양한 유전 인자 파라미터와 노드 모듈 개수에 따른 솔루션 품질 및 계산 시간에 관한 비교 실험 결과를 4장에서 설명한다. 끝으로 5장에서 본 연구의 결론과 추가 연구사항을 요약한다.

2. 모델 및 정의

제2장에서는 분산 시스템 환경에서 클러스터 노드 할당 모델 및 관련 표기법과 연산 정의에 대하여 설명한다. 사용자 프로그램 태스크(모듈)들은 분산 시스템내의 필요한 여러 파일들을 액세스하면서 실행된다. 클러스터 노드할당 문제는 병렬 태스크들이 실행되는 동안 클러스터간의 트래픽을 최소화하기 위해 병렬 프로그램 모듈을 어떻게 할당시켜야 하는가에 초점을 둔다. 즉, 병렬 모듈 상호간의 이동과 모듈간의 통신 트래픽을 줄일 수 있는 효율적인 할당 방법을 찾는 것이다.

(그림 2-1)은 1장에서 기술한 2단계 할당 기법을 이용하여 병렬 작업들을 최종적으로 할당한 모습이다. 주어진 병렬 프로그램은 8개의 모듈로 구성되어 클러스터 5의 워크스테이션 W_p에 있다. 프로그램의 시동은 클러스터 1의 워크스테이션 W_i에서 개시된다. 8개의 파일들은 전 시스템에 분산된다. 병렬 모듈들은 트래픽 최소화 기준에 따라서 첫 번째로 사용 가능한 6개의 클러스터들에 할당되고 나서 자기 클러스터 내의 하나의 노드에 할당된다.



(그림 2-1) 분산 컴퓨팅 환경에서 클러스터 노드 할당

할당에 있어 주요 목표는 클러스터 사이의 통신 시간을 최소화시키고 프로그램에 내재하는 병렬 성을 최대화하는 것이다. 병렬 프로그램 할당에 있어 어떤 클러스터에 주재

하는 프로그램은 m개의 병렬 모듈들로 분해되는데 각 모듈들은 사용 가능한 클러스터에서 수행될 수 있다. 여러 병렬 모듈들은 수행되는 동안 모듈 상호간에 통신하면서 여러 파일들로 이루어진 데이터 블록을 액세스한다.

제시하는 할당 시스템 모델에서는 각 클러스터 내에 모니터 노드(monitor node)를 두어 프로그램 프로필과 파일 분산에 대한 데이터를 유지 관리하고 있다. 클러스터 모니터 노드는 최적화 기법을 이용하여 병렬 프로그램을 구성하고 있는 병렬 모듈들을 클러스터에 효과적으로 분산시키는 일을 맡고 있다. 본 연구의 최적화 기법으로 유전 알고리즘(genetic algorithms : GA)을 이용하였다.

일반적으로 성취되어야 할 어떤 추상적 과제는 해가 될 수 있는 개체들의 공간에서의 탐색 문제로 생각할 수 있다. 최적해(optimal solution)를 추구하는 이러한 과제를 최적화 문제로 볼 수 있다. 문제에 대한 솔루션으로 작은 탐색공간에서는 고전적인 완전검색(exhaustive search) 방법으로 충분하지만 큰 탐색공간에서는 특수한 인공지능 기법이 사용되는데 유전 알고리즘은 그러한 기법 중 하나로 최근 많은 연구가 이루어지고 있다[5, 8, 11]. 사용되는 유전 알고리즘에 대한 설계 사항은 3장에서 기술한다.

데이터를 다루는 연산에 대한 정의를 설명하기 전에 시스템을 모델링하는데 사용된 가정들은 다음과 같다.

- 하나의 병렬 프로그램은 m개의 이주 가능한 병렬 모듈들로 분해된다.
- 각 모듈은 잠정적이거나 부분적인 결과를 출력 데이터 블록으로 각자의 모듈이 주재하고 있던 노드에 생성한다. 출력 데이터는 계산이 끝나면 프로그램이 주재하는 클러스터로 옮겨진다.
- 병렬 모듈들은 시스템 내에 분산되어 계산하는데 필요한 k개 파일들을 액세스한다.
- 시스템에서 사용 가능한 n개 클러스터가 제공된다.

제안된 모델에서 클러스터 노드 할당을 위해 사용되는 유전 알고리즘은 유전적 계승과 다윈(Darwin)의 적자생존의 자연 법칙을 모형화한 확률적인 탐색방법이다. 유전 알고리즘은 해가 될 가능성이 있는 개체집단(population)을 유지함으로써 여러 방향의 탐색을 실행하면서 이들 방향간의 정보 교환과 형성을 도모한다. 그러한 탐색은 최선의 해를 이용하는 것(exploitation)과 탐색공간을 조사하는 것(exploration) 사이의 적절한 균형을 필요로 한다. 개체집단은 자연 진화 과정을 흉내내는데 각 세대에서 비교적 좋은 해들은 재생산되고 나쁜 해들은 소멸된다. 여러 다른 해들 간의 차이를 구별하기 위해 환경의 역할을 수행하는, 즉 적합도(fitness)에 의해 해를 평가하는 목적 함수(objective function)를 사용한다. 유전자 연산 관련 파라미터에 대한 분석과 실험적 고찰 사항을 3, 4장에서 기술하고 여기에서는 목적 함수 모델을 세우는데 연구의 초점을 맞춘다.

병렬 프로그램을 모델링하는데 사용되는 표기법과 프로그램에서 파일 데이터를 다루는 연산에 대한 정의를 <표 2-1>로 요약한다.

〈표 2-1〉 표기법과 연산 정의

표 시	데이터 연산 정의
CB	Code Blocks - 프로그램이 있는 클러스터로부터 실행될 클러스터로 옮겨질 각 병렬 모듈에 대한 코드 블록 길이 (모듈 벡터).
FD	File Distribution - 전 클러스터를 통한 파일 분산($k \times n$ 행렬) ; $FD_{i,j}=1$: 파일 i는 클러스터 j의 어떤 노드에 저장 ; $FD_{i,j}=0$: 파일 i가 클러스터 j의 노드에 저장 안됨.
GPOD	Global Program Output Data - 전역 결과의 출력 데이터 블록 수(스칼라) ; 프로그램 클러스터로부터 시동 클러스터로 옮겨짐.
IMC	Inter-Module Communication - 모듈간의 통신($m \times m$ 행렬) ; $IMC_{i,j}$: 수행 모듈 i에서 모듈 j로 옮겨진 데이터 블록의 수.
LMOD	Local Module Output Data - 병렬 모듈에 의한 출력 데이터 블록 수 (모듈 벡터) ; $LMOD_i$: 계산되는 동안에 모듈 M_i에 의한 지역 결과의 출력 데이터 블록 수
MTS	Maximum Transfer Speed - 클러스터 사이의 최대 속도 ($m \times m$ 행렬) ; MTS_{C_i,C_j} : 네트워크 연결 클러스터 i와클러스터 j사이의 최대 전송속도.
NRC	Number of Remote Commands - 시동시 원격 명령 실행 요청에 따른 데이터 블록의 수
RWD	Read/Write Data - 병렬 모듈에 의해 액세스된 데이터 블록 수($k \times m$ 행렬) ; $RWD_{i,j}$는 파일 i에 대한 모듈 j의 읽기/쓰기 액세스 수.

2.1 목적 함수

워크스테이션 네트워크 환경에서 최적의 클러스터 배분을 하기 위해 병렬 모듈 프로필, 프로그램 파일과 관련 데이터 파일들에 대한 현재의 할당 정보에 따라 클러스터간 통신 트래픽을 최소화하여 병렬 모듈들을 할당시켜야 한다. 이러한 효율적인 할당을 위해 사용되는 목적 함수는 워크스테이션 클러스터간의 소요되는 총 트래픽 비용을 다음과 같이 표현한다.

$$OF = IO + RWA + IMC$$

IO : Input/Output, RWA : Read/Write Accesses,
IMC : Inter-Module Communication.

주어진 수식에서의 입출력(Input/Output) 비용에는 사용자 요청에 의한 프로그램 실행 및 사용할 수 있는 클러스터에 병렬 모듈을 보내는 드는 시동 비용과 실행 클러스터로부터 시동 클러스터로 결과를 전송하는데 드는 출력 비용을 포함한다.

시동 비용에 대하여 먼저 생각해 보기로 한다. 사용자는 어떤 클러스터에서 프로그램 p의 실행을 시작할 수 있다.

사용자 노드가 있는 클러스터를 C_u, 호출될 프로그램 코드가 있는 클러스터를 C_p라 하자. 시동 프로세스는 두 단계로 나뉜다. 첫 번째 단계에서는 전송 관련하여 C_u에서 C_p로 원격 명령 실행을 요청하는 것이고 두 번째 단계에서는 병렬 모듈 이주와 관련하여 할당 배분 벡터 $C_D = \{E_1, \cdots, E_m\}$에 따라서 C_p에서 선택된 각 실행 클러스터로 이주시키는 것이다. 여기서 첨자 m은 프로그램을 구성하는 모듈의 수를 뜻하고 C_D는 모듈 i가 클러스터 E_i에 할당되어야 한다는 것을 지정하는 모듈 벡터이다. 사용 가능한 클러스터 세트가 $C_S = \{C_s^1, \cdots, C_s^n\}$이라면 실행 클러스터 할당 배분은 $C_D = \{E_1, \cdots, E_m\}$, $E_i \in C_S$, $1 \le i \le m$의 관계를 갖는다. 그러면 시동 비용($Init_c$)은 아래와 같은 수식으로 표현된다. 여기서 사용된 표기는 앞서 기술된 표기법의 정의를 따른다.

$$Init_c = Init_1 + Init_2$$

$$Init_1 = NRC / MTS_{C_u,C_p} \quad \text{if} \quad C_u \neq C_p$$

$$Init_2 = \sum_{i=0}^{m} \frac{CB_i}{MTS_{C_p,E_i}}$$

위 수식에서 만일 클러스터 C_u와 C_p가 동일하다면 $Init_1 = 0$가 된다. 클러스터 C_p에서 클러스터 모니터 노드는 유전 알고리즘 기법을 이용하여 병렬 모듈들을 클러스터에 분산시키는 일을 맡고 있다.

출력 비용에 대한 사항으로 개별적인 병렬 모듈로부터 얻은 부분적 결과나 최종적인 출력은 먼저 병렬 프로그램이 있는 클러스터 C_p로 이송되어 진다. 그리고 C_p에서 필요한 지역적인 계산을 완료한 후에 전역 계산 결과를 C_p 프로그램 클러스터에서 C_u시동 클러스터로 전송한다. 이러한 출력(*Output*) 비용은 다음과 같이 표현된다.

$$Output = \frac{GPOD_p}{MTS_{C_u,C_p}} + \sum_{i=1}^{m} \frac{LMOD_i}{MTS_{E_i,C_p}}$$

파일을 액세스하는데 드는 비용에는 여러 모듈들로부터 데이터 파일들을 읽기 및 쓰기를 위한 액세스(Read Write Accesses)를 포함한다. 제시하는 시스템 모델은 분산 시스템에서 병렬 계산을 하는 동안에 액세스된 파일들의 복사본(replica)이 있다고 가정한다. 이 가정은 다음과 같은 이유에서 타당하다. 서로 다른 프로세서 노드에 같은 파일이 중복되어 있다면 해당 파일에 대한 유용성이 증가되고 성능 면에서도 이점이 있다. 이는 액세스 요청에 대한 가장 짧은 서비스 시간을 갖는 복사본을 선택할 수 있기 때문이다. 이런 장점들 때문에 대부분의 분산 시스템에서는 파일 중복(file replication) 기법이 채택되고 있다.

일반적으로 파일들을 액세스하는 데는 입력 데이터 요청 및 데이터 파일 수정 작업이 부수된다. *RWA* 비용에는 클러스터로부터 해당 파일을 읽거나 클러스터에서 쓰는데 드는 데이터 블록 이송하는 작업을 포함한다. 사용되는 클러스터에는 할당받을 실행 모듈이 있다. *RWA* 비용은 다음과 같다.

$$RWA = \sum_{i=1}^{k} \sum_{j=1}^{m} \left(\frac{RWD_{i,j}}{Max\{MTS_{E_{i,C_p}} : (p=1, \cdots, N \wedge FD_{i,p}=1)\}} \right)$$

본 연구에서 제시하는 모델에서는 앞서 언급한 파일 중복 기능을 고려하여 관련 목적 함수와 데이터 구조를 부분적으로 수정한다. 즉 *RWD* 블록을 *RD*(*ReadData*) 및 *WD*(*WriteData*) 블록으로 분리해 나누고 파일들 중에 가장 빠른 MTS 액세스 파일을 선택하여 읽는 경우도 생각하여 다음 수식으로 나타낸다.

$$RWA = \sum_{i=1}^{k} \sum_{j=1}^{m} \left(\frac{RD_{i,j}}{Max\{MTS_{E_j,C_p} : (p=1, \cdots, N \wedge FD_{i,p}=1)\}} \right) + \left(\frac{WD_{i,j}}{Min\{MTS_{E_{i,C_p}} : (p=1, \cdots, N \wedge FD_{i,p}=1)\}} \right)$$

모듈 상호간 통신 (Inter-Module Communication) 비용에는 모듈간의 메시지나 데이터를 전송하는데 드는 비용을 말하고 시스템 모델을 간소화시키기 위해 다음과 같은 가정을 둔다. 여기서는 두 클러스터 사이의 통신 링크는 양쪽 어느 방향에서든 가능하고 모듈 i와 모듈 j 사이의 통신비용 측정이 있어 한 방향에 있는 모듈에서 다른 방향으로 전송된 블록 수로써 계산한다. 여기서 m은 프로그램을 구성하는 모듈의 수이다.

$$IMC = \sum_{i=1}^{m-1} \sum_{j=i+1=0}^{m} \frac{IMC_{i,j}}{MTS_{E_i,E_j}}$$

모델 구현과 관련하여 병렬 프로그램에 대한 평균 프로필(profile)을 정하기 위한 모니터링 프로세스와 클러스터 할당 기법으로 이용되는 유전 알고리즘 프로세스가 필요하다. 이러한 프로세스들을 통하여 얻은 출력결과는 액세스할 수 있는 시스템 데이터베이스를 형성하고 있으며 각 클러스터의 CMN 노드에 위치해 있다. 즉 프로그램 프로필에 대한 정보는 클러스터 모니터 노드로부터 알 수 있다. 프로필 정보를 얻기 위해 병렬 프로그램 및 모듈들에 대한 움직임을 모니터링하고 수집된 관련 정보와 이에 따른 *CB, LMOD, RWD, IMC* 등을 초기화하거나 또는 최신 정보로 갱신한다.

네트워크상의 어떤 클러스터도 시동 클러스터 또는 실행 클러스터가 될 수 있으므로 각 클러스터의 CMN 노드는 프로그램 파일과 데이터 파일의 위치 정보를 제공하는 데이터베이스를 유지하고 있다. 클러스터에 속한 모니터 노드에는 클러스터내의 어떤 노드에 있는 프로그램 파일에 해당되는 프로필과 유전 알고리즘을 실행시켜 얻은 출력 결

과를 가지고 있다. 이밖에도 클러스터 모니터 노드는 클러스터들의 사용 상태 정보를 최신 정보로 경신한다.

프로그램 작성 후 컴파일 하여 초기 실행을 하고 나면 프로그램 프로필이 만들어진다. 사용자 요구조건에 따라 프로그램들이 이따금씩 자주 수정될 수 있다. 신뢰성 있는 프로필을 유지하기 위해 시기적절한 업데이트가 있어야 한다. 또한 새로운 프로그램 버전이 만들어질 때마다 자주 업데이트를 해야 한다. 새로운 프로필을 만들고 나서 업데이트 할 때 유전 알고리즘을 이용한 클러스터 할당 기법을 적용시킨다.

유전 알고리즘을 실행하면 현재 사용 가능한 클러스터 세트에 대한 출력 결과를 제공받는다. 각 클러스터에 대한 현재 사용 여부 상태를 클러스터 모니터 노드로 하여금 점검하도록 한다. 프로그램을 실행할 때 프로그램이 있는 클러스터의 모니터 노드는 다음 두 단계의 할당 절차를 따른다.

① 현재 사용 가능한 클러스터 세트에 대한 대체 방안을 차례로 점검한다. 즉 어떤 C_I 벡터에 대해 이 벡터의 모든 구성 성분들이 현재 사용 가능한 클러스터의 세트 C_K에 속하는지를 검사한다.

② 주어진 할당 방안으로 현재 시스템 상태를 만족시킬 수 있으면 매핑 후 바로 할당 프로세스를 중지한다. 만일 사용할 수 있는 클러스터가 없어 주어진 방안을 만족시킬 수 없다면 새로운 클러스터 할당을 위해 다시 유전 알고리즘을 이행시킨다.

클러스터 시스템은 프로세서, 기억장치, 디스크, 입출력장치, 네트워크 등에 있어 많은 중복을 가지면서 비용 효과 면에 있어 높은 가용성을 제공할 수 있다. 가용성(availability)은 사용자에게 이용 가능한 시스템 가동시간의 비율을 의미한다. 클러스터 가용성 면에 있어 거의 변동 없이 유지되는 인터넷 분산 시스템 환경에서 유전자 기반의 할당 프로세스 수행이 꼭 있을 필요가 없다고 생각할 수 있지만 분산 시스템에서 병렬 프로그램을 수행할 수 있으며 전체 시스템 성능을 향상시킬 수 있는 장점이 있다. 할당 최적화를 위한 추가적인 처리시간이 소요되겠지만 클러스터의 가용성 지원 관련 하드웨어 및 소프트웨어 유지보수, 고장수리, 업그레이드 등 안정적인 시스템 서비스를 위해 이러한 할당 프로세스 유지가 필요하며 통신 시스템 자원의 효율적 이용을 위해서도 중요하다.

주어진 병렬 작업들을 클러스터에 할당하고 나서 각 클러스터 내의 실행 노드를 정하기 위해 부하 균형[16] 기법을 사용할 수 있다. 즉 어떤 특정 작업을 동시에 처리될 수 있는 여러 모듈로 분할하여 여러 노드에 분산시켜 수행한다면 전체 실행 속도를 향상시킬 수 있고 과부하 상태의 노드의 작업 일부를 부하가 적은 노드로 이동시켜 수행하도록 하는 것도 가능하다. 이에 대한 선택 판단은 CPU 가용성을 어떻게 정의하는가에 따라 달라질 수 있다. 만일 유휴 워크스테이션이 많아 이중에 하나를 이용한다면 특별히 부하 균형 기법을 사용할 필요가 없다. 부하가 작은 워크스테이션을 간단히 찾으려면 인증된 처리능력에 대한 실행 프로세스의 개수를 계산하여 최소 비율을 갖는 워크스테이션을 선택한다.

3. 유전자 기반 할당 알고리즘

본 3장에서는 클러스터 노드 할당에 이용되는 유전 알고리즘의 설계에 관해 논의한다. 노드 할당 문제에서 m개 노드 모듈과 사용 가능한 n개 클러스터가 주어지고 가능한 모든 할당을 탐색하여 이에 따른 비용을 계산한다. 이러한 할당은 문제탐색공간의 크기는 $| n^m |$ 로 다루기 어려운 문제(intractable problem)이다. 이 문제에 대하여 최적 결과는 아니지만 최적해(optimal solution)에 근사한 할당을 얻기 위해 유전 알고리즘을 이용한 휴리스틱(heuristic) 솔루션을 제시한다. 실제로 크고 복잡한 문제에 대한 최적 결과를 얻기란 쉽지 않고 오랜 계산 시간을 필요로 하기 때문에 일반적으로 목적함수로부터 유도되는 평가함수를 통해 합리적인 솔루션 품질(solution quality)을 평가한다.

유전 알고리즘에서는 순환 t동안에 솔루션으로 가능성이 있는 개체(염색체 벡터), $P(t) = \{x_1^t, \cdots x_n^t\}$의 개체집단을 유지한다. 각 솔루션 x_i^t는 평가되어 적합도 척도로 사용된다. 더욱 적합한 개체를 선택함으로써 순환 $t+1$에서 새로운 개체집단이 구성된다. 이 개체집단의 멤버들은 유전 연산자에 의해 변천과정을 겪어 새로운 솔루션을 구성한다. 작은 변화에의해 한 개체로부터 새로운 개체를 생성하는 단일변환 돌연변이(mutation)와 둘 또는 그 이상의 여러 개의 개체로부터 부분들을 결합하여 새로운 개체를 생성하는 고차변환인 교배(crossover)가 함께 존재한다.

사용되는 구체적인 연산과정의 내용을 각 절에서 설명한다. 적당한 수의 세대 이후에 알고리즘은 수렴하고 이때 가장 좋은 개체가 근사 적합한 최선해(best solution)를 나타내는 것으로 기대된다. 아래 (그림 3-1)는 염색체로 표현되는 개체집단 중에서 적응도가 높은 개체의 유전자가 다음

Step [1] 무작위로 초기 개체집단 $P(t)$를 생성한다. t(세대) $\leftarrow 0$
Step [2] 개체집단의 각 개체 염색체에 대한 적합도를 계산한다.
Step [3] 종료조건을 만족할 때까지 다음 단계를 반복한다.
  [3.0] $t \leftarrow t+1$
  [3.1] $P(t-1)$로부터 $P(t)$를 선택한다.
  [3.2] 확률 P_c교배시킨다.(교배율)
  [3.3] 확률 P_m돌연변이 시킨다.(돌연변이율)
Step [4] 현재 개체집단을 새로 경신된 개체집단으로 교체시킨다.
Step [5] ***Step*** **[2]**로 간다.

(그림 3-1) 유전 알고리즘의 구조

세대로 전달되는 진화과정을 통하여 우수한 개체를 만들어 가는 유전 알고리즘의 전체적인 구조를 나타낸다.

3.1 염색체 표현방법

본 연구에서의 염색체 표현방법에 있어 개체의 i-번째 구성요소가 j라는 것은 i-번째 노드는 j라는 클러스터에 할당되는 것을 의미한다. 예를 들어, 스트링 11100011 는 다음과 같이 1, 2, 3, 7, 8 노드들은 클러스터 1에 할당하고 다른 4, 5, 6 노드들은 클러스터 0에 할당하는 매핑을 나타낸다.

전통적으로 사용되어 온 이진(binary) 표현은 정보의 비트당 최대수의 스키마타를 제공하고 해의 비트 스트링 코딩으로 이론적 분석을 편리하게 해주는 등 큰 비중을 차지해 왔다. 그러나 이진 코딩을 다차원 고정밀 수치문제에 적용하였을 때 전체 개체집단이 비전역적 최적값으로 조기 수렴하거나 미세한 국소 조정을 하지 못하거나 복잡한 구속조건이 존재할 때 동작하지 않는 등의 단점들을 가지고 있다. 암시적 병렬성 효과는 비트 스트링의 사용과 무관하고 실수 코딩이 많은 실제 문제에서 잘 동작한다는 실험적인 보고가 있다[6, 7, 11].

실수로 코딩된 부동점 유전인자의 사용은 최근 점차로 더 증가하고 있는 추세이다. 실수 공간의 특수한 성질을 이용함으로써 유전 연산자들로 하여금 문제에 잘 맞도록 한다. 즉 실수 표현은 표현 공간에서 상호 가까운 두 개의 점들이 문제공간에서도 역시 가까워져야 하고 그 반대도 동일하다는 성질이 있다. 이것은 표현 거리가 보통 다른 비트 위치의 수에 의해 정의되는 이진 방식에서는 성립하지 않는다. 본 논문에서 제시하는 클러스터 유전자 할당 모델에 서로 다른 코딩 방법을 적용하여 비교 실험을 볼 충분한 가치가 있다고 생각한다.

그레이 코딩(gray coding) 표현은 문제공간에서 서로 인접한 모든 두 점들이 한 비트만 다르다는 성질을 가지고 있다. 즉 매개변수의 값의 한 단계 증가는 코드에서 한 비트의 변화에 해당한다. 이진수 $b=\langle b_1,\cdots b_m\rangle$를 그레이 코드 $g=\langle g_1,\cdots g_m\rangle$로 변환하는 과정과 그 반대로 그레이를 이진 코드로 바꾸는 과정을 (그림 3-2)에 나타냈다. 여기에서 m은 이들 표현에서의 비트 수를 나타낸다.

```
procedure Binary-to-Gray
  begin
      g1 = b1 ;
      for k = 2 to m do
          gk = bk-1 XOR bk ;
  end.
procedure Gray-to-Binary
  begin
      value = g1 ;
      b1 = value ;
      for k = 2 to m do begin
      if g1 = 1 then value = ¬value;
        bk = value;
      end
  end.
```

(그림 3-2) 이진-그레이 변환 알고리즘

3.2 개체집단 크기 및 초기화

초기화에 관련하여 개체집단의 크기와 개체집단을 초기화하는 방법에 대하여 결정해야 한다. 개체집단의 크기를 코딩할 스트링 크기에 따라 정할 수 있다. 예를 들어, 32비트를 갖는 염색체는 개체집단의 크기를 32로 잡을 수 있다. 초기 연구자들은 좋은 솔루션을 만들기 위해 염색체 스트링 길이를 고려하여 개체집단의 크기를 지수적으로 증가시킬 필요가 있는 것으로 생각했다. 초기 개체집단의 구성 개체들은 실현 가능한 솔루션 공간에 균등하게 흩어져 있고 이러한 공간이 크다면 더 나은 수렴을 위해 더 많은 지점을 샘플링 하는 것이 바람직하다고 보는 것이다. 그러나 최근 연구보고[3, 4, 14]에 따르면 아주 적은 개체집단의 크기로도 만족스러운 결과를 얻을 수 있다는 것을 보여주고 있다. 4장에서는 두 세트의 실험을 통하여 이러한 결과를 확인하여 볼 것이다. 또한 초기 개체집단을 만드는 방법에는 무작위 초기화와 휴리스틱 초기화 2가지가 있는데 본 실험 연구에서는 무작위 초기화 방법을 사용한다.

3.3 평가함수와 선택

2장에서 기술한 목적함수(OF) 모델로부터 유도되는 평가함수(evaluation function)를 통해 합리적인 솔루션 품질(solution quality)을 평가한다. 목적/평가함수를 통한 적응력이 좋은 개체를 다음 세대에 구성하기 위한 선택(selection) 단계를 수행한다.

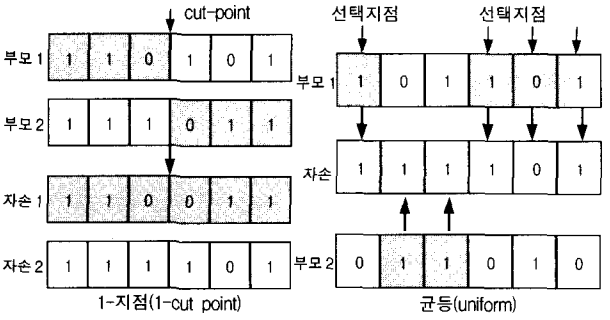
선택 연산자는 유전 알고리즘의 추진력을 제공한다. 너무 많은 힘을 받으면 유전 탐색이 일찍 성숙하여 종료될 수도 있다. 너무 힘이 적으면 진화 속도가 너무 느려지게 된다. 전형적으로 유전자 탐색의 초기에는 탐색공간의 새로운 영역에 대한 폭넓은 전역탐색으로 인한 낮은 선택 압력을 받도록 하지만 마지막에서는 탐색공간을 좁히기 위해 높은 선택압력을 주도록 권장하고 있다. 선택 연산자는 탐색 공간 중에서 기대할 수 있는 지역으로 유전자 탐색을 하도록 지시한다.

개체를 선택할 수 있는 여러 방법들이 있으나 본 실험 연구에서 사용한 방법은 Holland의 고전적인 룰렛바퀴(Roulette wheel) 선택 기법이 아니라 Back이 제안한 $(\mu+\lambda)$기법[1]이다. 이 방법에서는 부모와 자손으로부터 가장 좋은 염색체를 선택하는 결정적 절차로써 부모(μ)와 자손(λ)의 생존 경쟁으로부터 다음 세대의 최선해(μ)를 선택한다. 이

선택방법은 개체집단에서 중복 개체의 염색체 선택을 금지시키기 때문에 효율적인 해를 얻을 수 있으며 최근에 많은 연구자들이 선호하고 있다.

3.4 교 배

교배(crossover) 연산자는 부모의 염색체의 일부분을 서로 바꿈으로써 부모의 특징을 결합하여 두 개의 유사한 자손(offspring)을 구성하고 해가 될 가능성이 있는 것들 사이의 정보 교환을 위해 적용된다. 많이 사용되는 교배 연산자에는 1-지점, 2-지점, 균등 교배 연산자 등 세 개가 있는데 본 연구에서는 세 개의 교배 연산자들을 다른 유전 연산자들과 조합하여 성능 실험 결과를 비교한다.

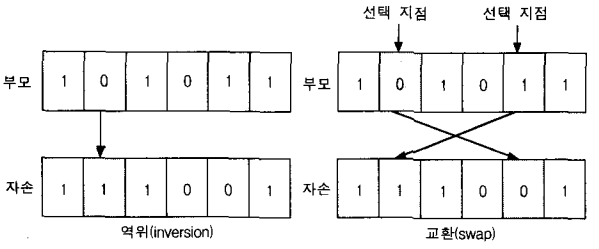


(그림 3-3) 교배 연산의 구조

1-지점 교배 연산은 (그림 3-3)에서와 같이 한 쌍의 염색체에 대해 한 지점(cut-point)을 무작위로 선택한 다음 이 지점부터 끝까지 첫째 염색체의 유전인자 특징 값들을 같은 범위를 갖는 둘째 염색체와 교환한다. 둘째 염색체의 특징 값들은 첫째 염색체의 특징 값들로 각각 옮겨진다. 2-지점 교배에서는 두 지점(1차, 2차)을 무작위로 선택한 다음 1차 지점에서 2차 지점까지 첫째 염색체의 특징 값들을 같은 범위의 둘째 염색체와 교환한다. 균등 교배[15] 연산에서는 동적인 방법으로 마스크(mask)라는 대치될 지점들의 세트를 각 염색체에 대해 무작위로 선택한 다음 무작위로 대치될 염색체들의 특징 값들을 생성된 지점들에 따라 상호간 교환한다. 즉 선택 연산에 의해 두 부모의 염색체를 임의로 골라낸다. 각 부모로부터 유전자를 교환할 요소를 임의로 선택하여 자식에 해당되는 유전자를 구성한다.

3.5 돌연변이

돌연변이(mutation) 연산자는 개체집단에 추가로 변화를 도입하기 위해 적용되는 것으로 돌연변이율(mutation rate)과 동일한 확률을 가지고 선택된 염색체에서 하나 또는 그 이상의 유전인자를 임의로 변경시키는 것이다. 이는 국부적인 최적에 빠지지 않게 만들어주기 위해 시행된다. 본 논문을 위해 사용된 돌연변이 연산자는 역위(inversion) 및 교환(swap) 연산자이다. 역위 돌연변이 연산에서는 비트 단위로 조금씩 수행되는데 돌연변이 시킬 지점이 지수 2의 한 성분이라면 (그림 3-4)과 같은 자손이 만들어진다. 교환 돌연변이 연산은 간단하게 두 지점을 무작위로 선택하여 그 내용을 서로 교환한다. 본 연구에서는 두 개의 돌연변이 연산자가 다른 유전 연산자들과 함께 사용되면서 나타나는 성능을 비교한다.



(그림 3-4) 돌연변이 연산의 구조

3.6 종료 조건

끝으로 하나의 중요한 관리 매개변수로써 종료 조건(termination condition)은 세대 수, 계산 경과 시간, 적합도 수렴 등을 포함하여 여러 가지 설정 기준이 있으며 구현 방법에 따라 차이가 있을 수 있다. 여기서 적합도 수렴은 개체집단의 모든 염색체가 같은 적합도 값을 가질 때에 일어나며 본 연구에서 실험 종료 조건으로써 사용된다.

4. 시뮬레이션 결과

4장에서는 앞서 설명한 유전 알고리즘에 기초하여 관련 파라미터 사용에 따른 성능 실험 결과를 분석 논의한다. 시뮬레이션 실험은 Sun Sparc/Solaris 워크스테이션에서 C 언어로 프로그램 되었다. 실험결과에 대한 평가를 위해 사용되는 두 가지 성능 측정치는 목적 함수의 값과 계산 시간이다. 실제로 크고 복잡한 문제에 대한 최적 결과를 얻기란 쉽지 않고 오랜 계산시간을 필요로 하기 때문에 일반적으로 목적함수로부터 유도되는 평가함수를 통해 합리적인 솔루션 품질(solution quality)과 계산속도(시간)를 고려하여 근사 최적해로 채택된다. 여기서 솔루션 품질로 사용되는 목적 함수의 값은 하나의 솔루션을 얻는데 필요한 비용을 의미한다. 유전자 실험 설계 관련하여 사용된 5가지 실험 변수는 개체집단 크기, 코딩 방법, 교배, 돌연변이, 노드 모듈 개수 등으로 그 내역은 다음과 같다.

- 2개의 개체집단 크기 : 20
- 2개의 코딩 방법 : 이진 및 실수 코딩
- 3개의 교배 방법 : 1-지점, 2-지점, 균등 교배
- 2개의 돌연변이 방법 : 역위 및 교환
- 다양한 크기의 노드 모듈 : 20, 40, 80

유전 연산자 값으로 보통 높은 교배율(예 : 0.80~0.95)과

낮은 돌연변이율(0.005~0.01) 사용을 권장하나 본 실험에서 사용된 교배율과 돌연변이율은 각기 1.0이 사용됐으나 기본적으로 교배 및 돌연변이 확률은 코딩방법의 선택과 주어진 문제에 따라 휴리스틱에 의해 경험적으로 결정한다.

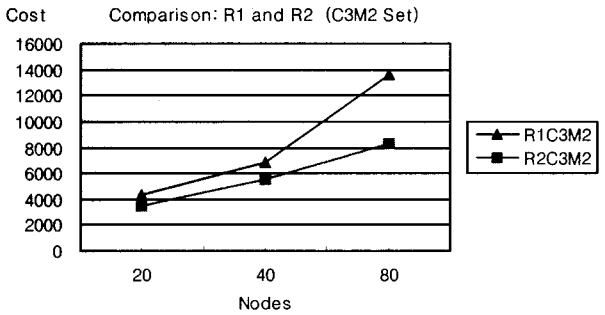
<표 4-1>는 실험결과를 보여주는데 각기 다른 개체집단에 대하여 셀(cell)의 수는 동일하다. 표에서 열은 노드 모듈 개수(20, 40, 80)를 표시하면서 뒤의 열에 있는 노드가 앞의 열보다 두 배 더 많다. 이러한 노드들은 10개의 클러스터로 주어진다. 표에서 행은 유전 인자의 다양한 조합을 나타낸다. 사용된 모든 유전 인자들의 조합으로 총 48(=2×3×2×4)개의 셀들이 만들어진다. 각각의 셀에 대한 10개의 데이터 세트를 무작위로 만들었고 여기에 표시된 모든 시험 결과는 서로 다른 난수 종자 값을 사용해 10번 실행하여 얻은 평균값으로 나타낸 것이다. 진하게 표시된 셀들은 최선해를 의미한다.

〈표 4-1〉 노드 모듈 및 유전인자 조합에 따른 성능 비교

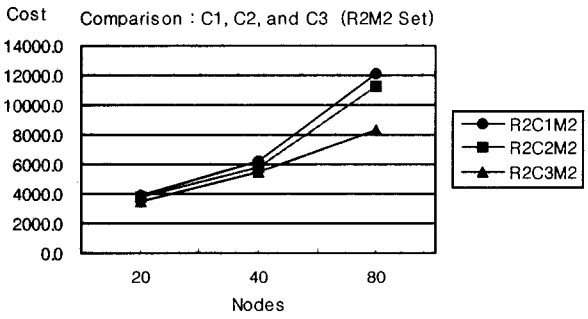
노드 모듈 : → 20			40		80	
유전인자	비용	시간(초)	비용	시간(초)	비용	시간(초)
R1C1M1	4383.1	13.3	7854.7	77.3	13252.8	801.4
R1C2M1	4282.6	14.0	7693.0	91.6	13310.4	835.7
R1C3M1	4285.1	11.6	6811.1	72.5	12866.4	761.3
R1C1M2	4503.1	15.4	7940.5	76.8	14973.6	735.8
R1C2M2	4391.6	16.3	7754.2	100.9	14701.0	939.0
R1C3M2	4287.5	23.4	6820.8	178.6	13653.6	2489.2
R2C1M1	3824.5	10.8	5309.2	79.4	7396.8	997.6
R2C2M1	3826.9	10.8	5323.8	**71.5**	7356.0	925.4
R2C3M1	**3312.4**	**10.6**	**5132.5**	73.9	**7244.6**	886.8
R2C1M2	3924.9	12.7	6197.2	75.8	12140.1	462.1
R2C2M2	3797.9	13.2	5747.7	62.9	11234.8	**385.2**
R2C3M2	3502.2	13.0	5494.1	80.9	8267.2	575.1

<표 4-1>에서는 여러 다른 조합의 유전 인자를 갖는 솔루션 품질에 대한 절대값 변화에 있어 상대적으로 적은 개수의 노드에 비하여 많은 노드를 갖는 시스템이 더 뚜렷하다. 즉 최고에서 최저 비용 치의 차이에 있어 20노드 시스템은 681에 비하여 80노드 시스템은 약 7632이다. 그러므로 노드가 많은 시스템이 비용이 최소인 해로부터 더 멀리 떨어져 있을 가능성이 더 크다. 계산 시간에 있어서도 유사한 경향을 보여준다. 20노드에서는 13.3의 차이를 보이는 반면 80노드에서는 2108의 큰 차이를 보였다. 이러한 결과는 올바른 유전인자 파라미터를 선택적 사용이 중요하다는 것을 보여준다.

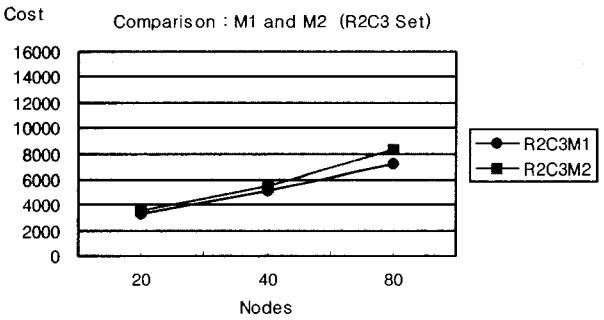
개체집단의 선정 관련하여 20개의 개체집단을 갖는 실험에서 그 두 배인 40개를 갖는 경우와 마찬가지로 솔루션 품질과 계산 시간에 있어 유사한 경향을 보여주었다. 큰 개체집단을 사용하면 직관적으로 작은 것보다 더 나은 솔루션을 얻을 것으로 기대하나 실제 실험결과에서는 약간의 비용 및 시간 증감이 있을 뿐 개체집단의 큰 차이 없는 솔루션을 얻을 수 있었다. 즉 개체집단의 크기에 따른 목적 함수 비용의 미비한 변화를 보여준다. 변화하는 개체집단 크기에 따른 목적 함수 비용과 계산 시간 결과를 보면 작지만 좋은 개체집단의 크기로 20을 선택하였다. GA 실험적 연구의 어려움은 개체집단 크기를 비롯하여 파라미터 사용에 관한 기술 이론을 간단하게 일반화할 수 없다는 것이다. 그러므로 통상적으로 적절한 파라미터의 값 결정은 특별히 고안된 문제에 대한 경험적인 휴리스틱 선택 판단에 의존한다.



(그림 4-1) 유전인자 C3M2 세트에 대한 표현방법(R1, R1) 비교



(그림 4-2) 유전인자 R2M2 세트에 대한 교배 연산자 (C1, C2, C3) 비교



(그림 4-3) 유전인자 R2C3 세트에 대한 돌연변이(M1, M2) 비교

(그림 4-1)~(그림 4-3)에서는 대표적으로 추출된 유전인자 세트에 대한 염색체 표현방법(R1, R1), 교배 연산자(C1, C2, C3), 돌연변이 연산자(M1, M2)를 비교하여 실험한 결과를 보여준다. 이러한 실험 결과는 다음과 같이 요약된다.

첫째로 그레이 코딩을 사용한 결과가 솔루션 품질을 나타내는 목적함수 비용과 계산 시간 측면에서이진 코딩의 결과보다 우수하였다. 이는 3장에서 설명한대로 부동점 표현이 개념적으로 문제공간과 좀더 가깝고 연산자들을 효율적으로 구현할 있다는 것을 확인해준다. 둘째로 1-지점, 2-지점, 균등 교배 연산자 등 세 개의 성능을 비교한 결과 2-지점 교배 연산자가 1-지점 교배 보다 일관되게 더 좋으며 균등 교배가 세 연산자 중에서 가장 효과적이다. 계산 시간에 있어 1-지점 교배 연산이 2-지점이나 균등 교배 연산보다 빠르지만 솔루션 품질을 고려할 때 균등 교배 연산이 1-지점, 2-지점 교배 연산 보다 더 좋은 결과를 보였다. 이는 1-지점 교배 연산의 단순성으로 얻는 속도는 있지만 개체들의 진화 생성하는 과정을 거쳐 최종 비용을 봤을 때 상대적으로 높다는 것을 의미한다. 마지막으로 교환 돌연변이 연산은 솔루션 품질과 계산 시간에 있어 역위 돌연변이 연산 보다 좋은 결과를 보였다.

5. 결 론

본 논문에서는 워크스테이션 클러스터로 구성된 분산 컴퓨팅 환경에서 병렬 프로그램 실행을 위한 클러스터 노드 할당을 위한 시스템 모델을 제안했다. 병렬 처리를 할 수 있는 분산 클러스터 시스템에서 유전 알고리즘 기법을 이용하는데 여러 유전인자 파라미터의 조합에 따른 할당 솔루션을 비교하였다. 제안된 시스템 설계모델에서는 분산 파일 시스템 구조를 바탕으로 시간에 따른 시스템 역동성을 고려하여 클러스터 할당 세트가 타당한 지를 점검하는 모니터 노드의 기능을 소개하고 파일 중복 기능을 포함하여 다양한 할당을 가능하다는 것을 보여준다. 본 연구에서는 성능에 영향을 주는 유전인자 파라미터로 코딩 방법, 개체집단의 크기, 교배, 돌연변이 연산자 등을 변화시켜 노드 모듈 개수에 따른 솔루션 품질 및 계산 시간에 관한 의미 있는 비교 실험결과를 얻었다.

향후 추가적인 연구과제로는 세 가지로 요약된다. 첫째로 본 시뮬레이션 연구에서는 부하균형에 관련하여 정적인 경우를 가정했으나 동적인 경우 프로세스 이주를 고려한 추가로 연구가 있어야 한다. 둘째로 유전 알고리즘 기법 외에 다른 휴리스틱 기법들, 예를 들어, 시뮬레이드 어닐링, 신경망, 타부 탐색 등의 기법과 이론적 연구를 하고 제시된 시스템에 적용하여 성능을 측정 분석하여 비교 실험 평가하는 과제가 있다. 이에 관한 연구보고[12, 13, 17, 19]가 최근에 나와 있다. 마지막으로는 제안된 시스템 모델 및 설계 개념에 따른 시뮬레이션 결과에 기초하여 보다 실제 시스템으로 구현하는 과제가 있다. 순차 노드 할당 알고리즘을 병렬 분산 시스템에서 적용하여 얼마나 성능 향상을 가져왔는지 평가하는 것이다.

참 고 문 헌

[1] T. Back, "Selective Pressure in Evolutionary Algorithms : A Characterization of Selection Mechanisms," *The First IEEE Conference on Evolutionary Computation*, Piscataway, NJ, pp.57-62, 1994.

[2] U. M. Borghoff, "Design of Optimal Distributed File Systems : A Framework for Research," *Operating Systems Review*, ACM Press, Vol.26, No.4, October, 1992.

[3] H. Chou, et al., "Genetic Algorithms for Communication Network Design-An Empirical Study," *IEEE Transactions on Evolutionary Computation*, Vol.5, No.3, pp.236-249, June, 2001.

[4] M. Gen and R. Cheng, "Genetic Algorithms & Engineering Optimization," John Wiley & Sons, 2000.

[5] D. E. Goldberg, "Genetic Algorithms in Search Optimization, Machine Learning," Addison-Wesley, Reading, MA, 1989.

[6] D. E. Goldberg, "Real-Coded Genetic Algorithms, Virtual Alphabets and Blocking," UIUC, Technical Report No. 90001, September, 1990.

[7] F. Herrera and M. Lozano, "Gradual Distributed Real-Coded Genetic Algorithms," *IEEE Transactions on Evolutionary Computation*, Vol.4, No.1, pp.43-63, April, 2000.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Also, MIT Press, 1975 and 1992.

[9] K. Hwang and Z. Xu, "Scalable Parallel Computing : Technology, Architecture, Programming," McGraw-Hill, 1998.

[10] L. Kleinrock and W. Korfhage, "Collecting Unusing Processing Capacity : Analysis of Transit Distributed Systems," *ACM Symposium on Operating Systems Principles*, pp.482-489, 1989.

[11] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Third, Revised and Extended Edition, Springer, 1996.

[12] K. Park, "A Coarse-Grained Parallel Genetic Algorithm for Clustered Document Allocation in Multiprocessor Information Retrieval Systems," *Journal of Electrical Engineering and Information Science*, Vol.4, No.6, pp.641-649, December, 1999.

[13] K. Park, "Comparison of Genetic Algorithms and Simulated Annealing for Multiprocessor Task Allocation," *The Transactions of The Korea Information Processing Society*, Vol.6, No.9, pp.2311-2319, Sept., 1999.

[14] C. Reeves, "GAs for Flow Shop Sequencing," *Computing Operation Research*, Vol.22, No.1, pp.5-13, 1995.

[15] G. Syswerda, "Uniform Crossover in Genetic Algorithms,"

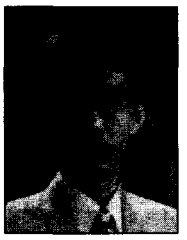
The Third International Conference on Genetic Algorithms, San Mateo, CA, pp.2-9, 1989.

[16] Y. Zhang, *et. al.*, "A Performance Comparison of Adaptive and Static Load Balancing in Distributed Systems," *The 28th Annual Simulation Symposium*, Phoenix, AZ, pp.332-340, April, 1995.

[17] A. Y. Zomaya, C. Ward, and B. Macey, "Genetic Scheduling for Parallel Processor Systems : Comparison Studies and Performance Issues," *IEEE Transactions on Parallel and Distributed Systems*, Vol.10, No.8, pp.795-812, August, 1999.

[18] S. Jang and B. Yoon, "A Comparative Study on Real-number Processing Method in Genetic Algorithms," *KIPS Transactions*, Vol.5, No.2, pp.361-371, Feb., 1998.

[19] B. Hamidzad도, L. Y. Kit and D. J. Lilja, "Dynamic Task Scheduling Using Online Optimization," *IEEE Transactions on Parallel and Distributed System*, Vol.11, No.11, pp.1151-1163, November, 2002.



박 경 모

e-mail : kpark@catholic.ac.kr

1980년 중앙대학교 전산학과 졸업(학사)

1983년 서울대학교 계산통계학과 전산과학전공(이학석사)

1983년~1985년 삼성전자 컴퓨터개발 연구원

1990년 미국 New Jersey Institute of Technology

1994년 미국 George Mason University(GMU) 컴퓨터정보공학부(공학박사)

1994년~1995년 GMU IT&E Lab. 연구원

1995년~1996년 한국전자통신연구소/ETRI 연구원

2001년~2002년 조지메이슨대 컴퓨터학과 방문교수

1996년~현재 가톨릭대학교 컴퓨터정보공학부 부교수

관심분야 : 컴퓨터 시스템 성능평가, 분산·병렬처리, 시스템 최적화