집단화를 위한 병렬 알고리즘의 구현

배 용 근†

요 약

많은 양의 패턴들을 분석할 때, 이 패턴들을 어떤 평가함수에 의하여 여러 군으로 집단화할 필요가 있다. 이 과정은 입력 패턴의 수가 많을 경우 상당한 량의 계산을 필요로 하며, 이를 위한 병렬화 알고리즘이 요구된다. 이 문제를 해결하기 위하여 본 논문은 K-means 알고리즘을 병렬화한 병렬 집단화 알고리즘을 제안하고, 메세지 전송을 근간으로 하는 MIMD 병렬 컴퓨터하에서 이를 수행하였다. 실험 및 성능 분석을 통하여 입력 패턴이 많을 경우, 본 병렬 알고리즘이 적절함을 알 수 있었다.

Parallel Algorithm For Level Clustering

Yong-Keun Bae†

ABSTRACT

When we analize many amount of patterns, it is necessary for these patterns are to be clustering into several groups according to a certain evaluation function. This process, in case that there are lots of input patterns, needs a considerable amount of computations and is reqired parallel algorithm for these. To solve this problem, this paper propose parallel clustering algorithm which parallelized k-means algorithm and implemented it under the MIMD parallel computer based message passing. The result is through the experiment and performance analysis, that this parallel algorithm is appropriate in case these are many input patterns.

1. 서 론

집단화(clustering)는 입력 패턴들을 여러 군(group)으로 분류하기 위하여 사용되는 비감독 패턴 분류의 일종이다. 이러한 집단화를 위한 가장 보편적인 방법은 먼저 평가함수(CF)를 정의하고 이 평가함수가 최대가 되도록 입력 패턴들을 집단화하는 것이다.

많은 집단화 알고리즘들은 입력 패턴들의 수와 군의 수가 방대한 계산량 때문에 알고리즘의 병렬화를 필요로 하며, 이를 위하여 K.Hwang과 D.S Kim[5]은 영상처리 및 패턴인식 분야의 응용을 위한 OMD(Orthogonal Multiprocessor)구조를 가지고 있는 병렬 컴퓨터에서 단층 집단화(level clustering)를 위한 병렬화 알고리즘을 제안하였으며, Li[6]는 SIMD 병렬 컴퓨터에서 계층적 집단화 문제를 위한 병렬 알고리즘을 제안하였다.

본 논문에서는 메세지 전송을 근간으로 하는 MIMD 병렬 컴퓨터에서 단층 집단화를 위한 병렬화 알고리즘을 제안하고 구현하였다. 단층 집단화 알고리즘으로 많이 사용하는 K-민즈(K-means) 알고리즘을 이용하였다. 일반적으로 이 알고리즘은 많은 반복 과정을 필요로 하며, 각 과정에서 각 패턴들과 모든 군과의 유사도를 계산하고, 이를 근거로 모든 군들의 중심을 다시 계산해야 한다. 즉 이 과정은, 입력 패턴의 수가 많을 경우 대단히 많은 양의 계산을 필요로 하며, 이에 따라 병렬화 알고리즘이 요구된다.

본 논문에서는 평가함수로 집단화 알고리즘에서 많이 사용하는 자승오차의 합을 이용하였다. 여기에서 두 패턴 사이의 유사도(similarity)는 뉴클리디안 거리 방법을 이용하였으며, 이를 이용하여 같은 군에 속한 패턴들은 높은 유사도를 가지며 다른 군에 속한 패턴들 사이에는 낮은 유사도를 가지게 하였다[1, 2].

† 정 회 원 : 조선대학교 전자계산학과 조교수
논문접수 : 1994년 11월 16일; 심사완료 : 1995년 3월 25일

2. K-means 집단화 알고리즘

본 장에서는 집단화 알고리즘의 일종인 K-means 알고리즘을 소개한다. 각 패턴이 식 (2-1)과 같이 m개의 특성들로 구성된 N개의 패턴 벡터 $X_1, X_2, \cdots, X_N$들을 분류하는 것을 생각해 보자.

$$X=(x_1, x_2, x_m)^t \cdots\cdots (2\text{-}1)$$

임의의 패턴 X와 군 C_j의 중심 Z_j 사이의 뉴클리디안 거리는 유사도의 척도로 이용될 수 있으며, 식(2-2)와 같이 정의된[1].
사이의 뉴클리디안 거리는 유사도의 척도로 이용될 수 있으며, 식 (2-2)와 같이 정의된다[1].

$$D^2 = | X-Z_j |^2 = \sum_{i=1}^{m} (x_i - zi)^2 \cdots\cdots (2\text{-}2)$$

또한 이를 이용한 집단화 알고리즘의 평가함수는 식 (2-3)과 같이 정의될 수 있다.

$$CF = \sum_{i=1}^{k} \sum_{x \in C_j} | X-m_j |^2 \cdots\cdots (2\text{-}3)$$

여기에서 C_j는 K 개의 군들 중 j번째 군을 나타내며, j번째 군의 평균 특성 m_j는 식(2-4)와 같이 정의된다.

$$m_j = \{ \sum_{x \in C_j} X \} / \#(C_j) \cdots\cdots (2\text{-}4)$$

여기에서 $\#(C_j)$는 군 C_j 내의 패턴의 수이다.

군의 수 K가 정해 졌을때, 집단화 알고리즘은 평가함수를 최소로 하는 군들을 찾는 것이다. 또는 이 알고리즘은 군의 수가 정해지지 않을 경우도 반복적으로 적용함에 의해 이용할 수 있다. 이러한 집단화 알고리즘은 많은 종류[1, 2]가 있으며, 본 논문에서는 K-means 알고리즘을 이용하였다. K-means 알고리즘에 대하여는 다음과 같으며, 〈알고리즘 1〉 에 자세히 수록하였다.

K-means 알고리즘의 입력은 n개의 입력 패턴이며, 입력 패턴을 K개의 군으로 집단화한 것이 출력이 된다. 알고리즘은 먼저 K개의 초기 군 중심들 $Z_1^{(1)}, Z_2^{(1)}, \cdots Z_K^{(1)}$을 선택한다. 이것은 입력 패턴들 중에서 처음의 K개의 패턴 $X_1, X_2, \cdots X_K$로 한다. 두 번째로 선택된 패턴 $X_1, X_2, \cdots X_N$에 각각 군들 $C_1, C_2, \cdots C_K$를 다음의 방법으로 할당한다. 입력 패턴 X가 만일 $i \neq j$인 모든 $i=1,2,\cdots K$에 대하여 $| X-Z_j^{(1)} | \leq | X-Zi^{(1)} |$ 이면 입력 패턴 X에 군 C_j를 할당한다. 세 번째로, 각 군의 새로운 중심점 $Z_j^{(i+1)} = \{ \sum_{x \in C_j} /X \} \#(C_j)$를 모든 군에 대하여 구한다. 여기에서 $\#(C_j)$는 C_j 내의 패턴의 수이다. 네 번째로, 만일 모든j= 1, 2, K,에 대하여 $Z_j^{(l+1)} = X_j^{(l)}$이면, 알고리즘은 수렴하고 여기에서 집단화 과정을 끝낸다.

만일 위의 조건이 만족하지 않을 경우 두 번째 단계부터 반복하여 다시 수행한다.

〈알고리즘 1〉 K-means 집단화 알고리즘
〈Algorithm 1〉 K-means clustering algorithm

```
입력 : N개의 입력 패턴 (X1, X2, ... ,XN)
출력 : K개의 군들(C1, C2, ... CK)

[단계 1:] · for I = 1 to N
            XI의 최소거리 = ∞
          end for
          · L = 0
          · for I = 1 to K
              ZI(L) = XI
          end for
[단계 2:] · L = L + 1
          · for I = 1 to N
              for j = 1 to K
                  if |XI - Zj(L)| < XI의 최소거리
                    then XI의 최소거리 = |XI - Zj(L)|
                         XI의 소속군 = Cj
              end for
          end for
[단계 3:] · for I = 1 to K
              Zj(L+1) = { Σ X }/ #(Cj)
          end for
                    x∈Cj
[단계 4:] · for I = 1 to K
              if ZI(L+1) ≠ ZI(L) then goto [단계 2:]
          end for
          · K개의 군으로 부터 CF를 구한다.
```

3. 병렬 집단화 알고리즘

빠른 계산을 위하여 위의 집단화 알고리즘을 병렬 알고리즘으로 변환한 다음 고성능 마이크로프로세서인 트랜스퓨터를 이용한 Message passing 방식의 병렬 컴퓨터[8, 9]에서 수행하였다. 병렬처리 알고리즘은 주처리소자 내의 알고리즘과 다수의 부처리소자 알고리즘으로 나누어지게 된다. 주처리소자내의 알고리즘은 N개의 입력 패턴들을 입력으로 하여 이를 분할하여 각 부처리소자에게 전송한다. 만일 부처리소자의 수가 n이라면 각 부처리소자는 N/n개의 패턴들을 입력으로 하여 각각의 알고리즘을 수행한다.

알고리즘의 전체적인 개요는 다음과 같다. 주처리소자는 k개의 군 평균들을 정한 후 이를 각각의 부처리소자에게 전송한다. 각 부처리소자는 이 K개의 부분합과 부분군 수를 주처리소자에 전달한 뒤 앞의 상태로 가서 기다린다. 주처리소자는 모든 부처리소자들로 부터 부분합과 부분군 수를 입력받아, 이로부터 K개의 군 평균들을 계산하여 이를 각각의 부처리소자에 전송하고 부처리소자는 이들 입력으로 하여 각 처리소자내의 패턴들을 분류하고 K개의 부분합과 부분군 수들을 주처리소자에 전달한 후 앞의 상태로 가서 기다린다. 이 상태는 주처리소자에서 집단화의 종결조건이 만족될 때까지 반복된다.

즉, 본 병렬처리 알고리즘은 K-means 집단화 알고리즘 중에서 가장 계산량이 많은 〈알고리즘 1〉의 [단계2:]를 병렬화하여, 빠른 결과를 얻도록 하였다. 3.1에 주처리소자내의 알고리즘을 소개하였으며 〈알고리즘 2〉 에 해당 과정을 자세히 수록하였다. 또 3.2에 부처리소자내의 알고리즘을 소개하였으며 〈알고리즘 3〉에 해당 과정을 자세히 수록하였다.

3.1 주처리소자내의 알고리즘

주처리소자내의 알고리즘은 입력 패턴들을 부처리소자로 분배해 주고 부처리소자들이 이를 이용하여 중간 결과를 전송하면 이를 이용하여 집단화 결과를 계산하고 조건이 만족되지 못하면 계산된 중간 결과를 다시 부처리소자들에게 전송한 후 처리소자들의 계산 결과를 기다리게 된다. 이런 과정이 조건이 만족할 때까지 반복되며, 이를 간략하게 소개하면 다음과 같다.

먼저 N개의 입력 패턴들을 모든 처리소자에 분배한다. 만일 처리소자가 n개 이면 각 처리소자는 N/n의 입력 패턴들을 가지게 된다. 두 번째로, K개의 초기 군 중심점 $Z_1^{(1)}$, $Z_2^{(1)}$,…,$Z_K^{(1)}$들을 입력 패턴들 중 처음의 K개 X_1, X_2,…X_K로 선택한다. 세 번째로, 이 군 중심점 들을 모든 처리소자에 전송한다. 네 번째로, 모든 처리소자들로 부터 K개의 부분합과 부분군 수들을 받는다. 이를 근거로 개의 전체 합과 군 수들을 계산한다. 다섯번째로, 이들을 이용하여 군들의 중심점을 다시 계산한다. 마지막으로, 알고리즘의 수렴 조건을 검사하고 조건이 만족한 경우, 집단화 과정을 끝내고, 위의 조건이 만족하지 않을 경우 세 번째 단계부터 다시 반복하여 수행한다.

〈알고리즘 2〉 K-means 집단화 병렬 알고리즘 (주처리소자 부분)
〈Algorithm 2〉 K-means clustering parallel algorithm (main processing element part)

```
입력 : N개의 입력 패턴 (X1, X2, ... ,XN)
출력 : K개의 군들(C1, C2, ... CK)

[단계 1:] - for i = 1 to N
              Xi의 최소거리 = ∞
            end for
          - for I = 1 to n
              N/n개씩 입력 패턴을 I 번째 처리소자 Pi에 전송
            end for
[단계 2:] - L = 0
          - for i = 1 to K
              Zi(L) = Xi
            end for
[단계 3:] - L = L + 1
          - for i = 1 to n
              K개의 중심점 Z1(L),Z2(L), ... ,ZK(L)를
              처리소자 Pi에 전송
            end for
[단계 4:] - for i = 1 to n
              처리소자 Pi로 부터
              K개의 부분합 Sum1(1),Sum1(2), ... ,Sum1(K)
              및 K개의 부분군수 N1(1),N1(2), ... ,N1(K)를 받음
            end for
          - for i = 1 to K
              Sum(i)=Sum1(i)+Sum2(i)+, ... ,+SumN(i)
              N(i)=N1(1)+N2(i)+, ... ,+NN(i)
            end for
[단계 5:] - for i = 1 to K
              Zi(L+1) = Sum(i) / N(i)
            end for
[단계 6:] - for i = 1 to K
              if Zi(L+1) ≠ Zi(L) then go to [단계 3:]
            end for
          - K개의 군으로 부터 CF를 구한다.
```

3.2 부처리소자에서의 알고리즘

본 절에서는 다수의 부처리소자에서 수행하는 프로그램의 알고리즘을 소개할 것이며, 그중 P 번째 부처리소자의 알고리즘을 소개하면 다음과 같다.

먼저, 주처리소자로 부터 N/n개의 입력 패턴들을 받는다. 그다음 두 번째로, 주처리소자로 부터 K개의 군 중심점들을 받는다. 세 번째로, 부처리 소자 P에 있는 모든 패턴들에 대하여 다음의 방법으로 군 C_1, C_2,…C_K중 하나를 할당한다. 부처리소자 P 내에 있는 모든 입력 패턴 X에 대하여, 만일 $i \neq j$인 모든 i=1,2,…K에 대하여 $|X-Z_j^{(1)}| \leq |X-Z_i^{(1)}|$ X에 군 C_j를 할당한다. 네 번째로, K개의 개의 부분합 및 부분 군의

수들을 모든 군에 대하여 식 (3-1) 및 (3-2)와 같이 계산한다.

$$sum^P(i) = \sum_{X \in C_i} X \quad \cdots\cdots(3\text{-}1)$$

다섯 번째로,

$$N^P(i) = \#(C_i) \quad \cdots\cdots(3\text{-}2)$$

다섯 번째로 K개의 부분합 $sum^P(1)$, $sum^P(2)$,…,$sum^P(K)$ 및 부분군의 수 $N^P(1)$, $N^P(2)$, …$N^P(K)$를 주처리소자로 전송한 뒤, 두 번째 단계로 간다.

3.3 처리소자간 메세지 전송 형태

처리소자간 메세지 전송 형태를 보면 처음 시작할 때 주처리소자는 부처리소자에게 입력 패턴들을 일정량씩 나누어 주며 또한 군들의 초기 중심점들을 나누어준다. 그뒤 다음의 메세지 전송들이 주처리소자의 종결 조건이 만족될 때까지 반복된다. 각 부처리소자는 내부 계산이 끝나면 부분합과 부분군 수를 주처리소자에게 전송해 주며 주처리소자는 이를 받아 각 군들의 중심점들을 다시 계산해서 이를 부처리소자들에게 나누어 준다.

〈알고리즘 3〉 K-means 집단화 병렬화 알고리즘
(P 번째 부처리소자 부분)
〈Algorithm 3〉 K-means clustering parallel algorithm
(P th sub processing element part)

```
입력 : N/n개의 입력 패턴 및 K개의 중심점들
출력 : K개의 부분합 및 부분군 수

[단계 1:] • 주처리 소자로 부터 N/n개의 입력 패턴들을 받음
[단계 2:] • for i = 1 to K
              Sum^P(i) = 0
              N^P(i)   = 0
            end for
          • 주처리소자로 부터 K개의 중심점들
            Z_1^(1), Z_2^(1), ... ,Z_K^(1)을 받음
[단계 3:] • for N/n개의 각 입력 패턴 X_i
              for j=1 to K
                  If ||X_i - Z_j^(1)|| < X_i의 최소거리
                    then X_i의 최소거리 = ||X_i - Z_j^(1)||
                         X_i의 소속군 = C_j
              end for
            end for
[단계 4:] • for i = 1 to K
              for j군 C_i내의 각각의 X_j
                  Sum^P(i)=Sum^P(i)+X_j
                  N^P(i)  =N^P(i)+1
              end for
            end for
[단계 5:] • K개의 부분합 Sum^P(1),Sum^P(2), ... ,Sum^P(K)
            및 부분군 수 N^P(1),N^P(2), ... ,N^P(K)를 주처리소자로
            전송
          • goto [단계 2:]
```

본 병렬 알고리즘에서 사용한 처리소자간 통신 형태는 다음과 같다.

시작정보	송신처리소자	수신처리소자	전송정보	끝정보

여기에서 시작정보와 끝정보는 전송정보의 시작과 끝을 알리기 위함이며, 송신처리소자는 정보를 전송하는 처리소자의 고유 숫자이며, 수신처리소자는 정보를 수신하는 처리소자의 고유 숫자이다. 또, 전송정보는 보내고자 하는 전송정보의 수와 전송될 정보를 포함하고 있으며, 본 알고리즘에서 사용한 전송정보 형식은 다음과 같다. 첫 번째로, 주처리소자 알고리즘 [단계 1:]에서 각각의 부처리소자에게 입력 패턴을 전송하는 경우 전송정보의 형식은 아래와 같이 분배될 입력 패턴의 수와 입력 패턴들로 구성되어 있다. 처리소자의 수가 n개이고 입력 패턴의 수가 N개이고 각 입력 패턴내 특성의 수가 m이라면 각 부처리소자로 전송되는 통신량은 대략 Nm/n 이 된다.

분배될 입력패턴의 수	입력패턴들

두번째로, 주처리소자 알고리즘 [단계 3:]에서 각각의 부처리소자에게 군의 중심점들을 전송하는 경우 전송정보의 형식은 아래와 같이 군의 중심점 수와 군의 중심점들로 구성되어 있다. 만일 군의 수가 K라면 각 부처리소자로 전송되는 통신량은 대략 Km이 됨을 알 수 있다.

군의 중심점 수	K개의 군의 중심점들

세 번째로, 부처리소자 알고리즘 [단계 5:]에서 k개의 부분합 및 부분군들을 주처리소자에 전송하는 경우 전송정보의 형식은 아래와 같으며, 만일 군의 수가 K라면 통신량은 각각의 경우 Km이 됨을 알 수 있다.

부분합의 수	K개의 부분합들

부분군의 수	K개의 부분군들

이러한 메세지들은 처리소자내의 4개의 채널을 통하여 다른 처리소자에게 전달되며, 이 채널을 이용하는 방법은 병렬처리 C언어의 라이브러리 함수를 이용한다. 즉 정보를 전송할 때는, word 단위로 정보를 전달하는 chan_out_word(

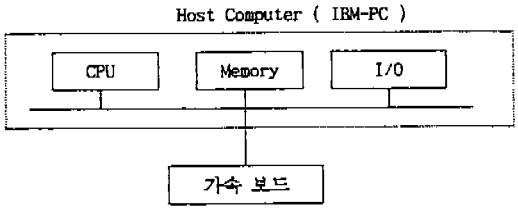
word, channel_number) 함수와, message단위로 정보를 전달하는 chan_out_message (message_size, message, channel_number) 함수를 이용한다. 또, 정보를 수신할 때는, word로 정보를 수신하는 chan_in_word(word, channel_number) 함수와, message단위로 정보를 수신하는 chan_in_message (message_size, message, channel_number) 함수를 이용한다.

4. 성능 분석 및 실험 결과

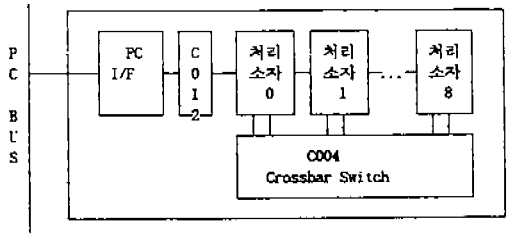
4.1 실험 환경

본 논문에서 사용한 병렬처리 컴퓨터는 병렬화 데이타의 병렬성 정도에 따른 Flynn의 분류법에 의하여 MIMD 컴퓨터 구조에 속하며, 통신 방식은 처리 소자들 간에 직접적인 통신 링크를 통하여 정보를 교환하는 메세지 전송(message passing) 방식이며, (그림 1)과 같이 IBM PC를 호스트 컴퓨터로 하고 PC 버스에 장착되는 가속기의 형태로 이루어져 있다. 이 가속기는 (그림 2)와 같이 9개의 처리소자를 가질 수 있으며, 본 논문에 이용된 가속기는 3개의 처리소자로 이루어져 있다. 이 병렬 가속기의 특징은 각 처리소자의 CPU를 병렬처리용 전용칩을 사용하고, 또한 CPU내에 다른 CPU와의 통신을 위하여 4개의 직렬 통신 포트를 가지고 있어 처리소자간 연결이 간단하고 이로 인하여 통신 매카니즘이 간단하다는점과, 처리소자들의 연결구조를 소프트웨어에 의하여 제어할 수 있도록 하기 위하여 재구성을 갖고 있는 크로스바 스위치를 사용한 점들이다. 9개의 처리소자중 1개는 호스트 컴퓨터와 통신하기 위하여 연결되어 있으며, 각 처리소자들은 처리소자의 CPU내에 있는 4개의 직렬 통신 채널들을 통하여 다른 처리소자들과 동기화된 통신을 할 수 있다. 각 처리소자는 CPU로 INMOS 사의 트랜스퓨터를 이용하였으며 이는 Hoare의 CSP(Communication Sequential Process)에 기초를 두는 OCCAM 프로그래밍 언어의 모델을 기반으로 하여 single-chip 하드웨어로 병렬처리를 지원하는 마이크로프로세서이다.

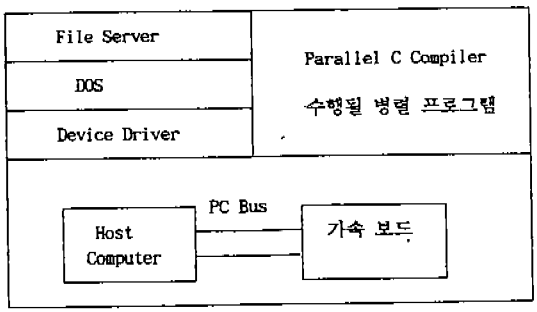
병렬 컴퓨터의 소프트웨어를 보면 (그림 3)에 보는것 처럼 호스트 컴퓨터를 위한 것과 가속기 보드를 위한 것이있다. 호스트 컴퓨터를 위한 것은 전체 시스템을 운영하는 데 필요한 운영체제와 이와 관련된 device driver, 그리고 그 위에서 실행 환경을 구성해 주는 file server등이 있고, 가속기 보드를 위한 것은 프로그램 개발을 위한 컴파일러 및 응용 프로그램등이 있다. 모든 수행 프로그램의 컴파일 및 수행 환경은 호스트 컴퓨터를 통하여 이루어지며, 실제 수행되는 병렬 프로그램은 가속기 보드위에서 수행된다.



(그림 1) 본 논문에서 이용한 병렬컴퓨터의 하드웨어 구성도.
(Fig. 1) Hardware block diagram used in this paper.



(그림 2) 병렬 컴퓨터내의 가속기의 구성도.
(Fig. 2) The block diagram of accelerator in the parallel computer.



(그림 3) 본 논문에서 이용한 병렬컴퓨터의 소프트웨어 구성도.
(Fig. 3) Software configuration diagram used in this paper.

4.2 성능 분석

병렬처리를 통한 성능 향상 효과를 평가하기 위한 척도로서 다음의 요소들을 이용하였으며 이를 근거로 실험한 후 결과를 뒤에 도시하였다. [3, 4, 10, 11, 13, 14].

(1) 수행시간(running time)

병렬처리 효과를 평가하는 가장 중요한 척도로서 실행을 시작해서 끝날 때 까지의 경과된 시간으로 정의된다. 수행시간은 경로 이동 구간과 계산 구간으로 구성되며, 경로 이동 시간은 자료가 네트워크를 통해 처리기 사이를 이동하는데 걸리는 시간이며, 계산 시간은 한 처리기 내에서 자료에 가해진 연산 작업에 소요된 시간이다.

(2) 가속율(speed up)

직렬 시간 대 병렬 시간의 비를 가속성이라 하며 아래와 같이 정의된다.

$$\text{가속율} = \frac{\text{직력 수행시간}}{\text{병렬 수행시간}}$$

(3) 효율성(efficiency)

계산에 소요되는 비용 효율의 척도로서 직렬비용 대 병렬비용의 비로 정의된다.

$$\text{효율성} = \frac{\text{가 속 율}}{\text{처리소자의 수}}$$

이중 가속율을 계산해 보면 다음과 같다. N은 입력 패턴의 수이고, n은 처리소자의 수이고, K는 클래스(class)의 수이고, m은 각 입력 패턴내의 특성의 수이고, L 은 k-memans 알고리즘에서 수렴때 까지의 반복 횟수라고 하였다. 이를 이용하여 단일 소자를 이용했을 경우의 처리 스텝과 n개의 처리소자를 이용했을 경우의 처리 스텝을 계산하여 보면 다음과 같다. 먼저 〈알고리즘 1〉에서 보는 바와 같이 하나의 단일 처리소자를 이용했을 경우 위의 집단화 알고리즘의 처리 스텝을 계산하기 위하여 각 단계의 처리 스텝을 계산하면 아래와 같다.

[단계 1] N + Km
[단계 2] LNKm
[단계 3] LKm
[단계 4] LKm

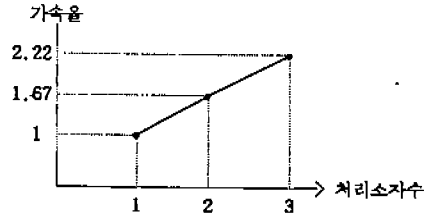
이로부터 전체 단계의 처리 스텝은 N+Km+LNKm+2LKm 이며 만일 입력 패턴 N 의 수가 상당히 크다면 위의 처리 스텝은 LNKm이 됨을 알 수 있다. 다음으로, n개의 처리소자를 이용했을 경우 전체 처리 스텝을 계산하기 위여 먼저 〈알고리즘 2〉에 있는 주처리 알고리즘의 처리 스텝을 아래와 같이 계산하여보았다.

[단계 1] N + C1*n = N + mN (C1 = mN/n)
[단계 2] Km
[단계 3] LnC2=LnKm (C2 = Km)
[단계 4] LnC3+LnC4=2 LnKm (C3=C4 =Km)
[단계 5] Lkm
[단계 6] Lkm

또한 〈알고리즘 3〉에 있는 부처리 알고리즘의 처리 스텝을 계산하기 위하여 각 단계의 처리 스텝을 계산하면 아래와 같다.

[단계 1] C1=mN/n
[단계 2] LK+LC2=LK+LKm
[단계 3] LNKm/n
[단계 4] LNm/n
[단계 5] LC3+LC4=2LKm

이로 부터 n개의 처리소자가 이용되는 경우 전체 과정의 처리 스텝은 주처리 알고리즘과 부처리 알고리즘의 처리 스텝을 더한 것이 되며, 이는 부처리 알고리즘에서 처리하는 계산 시간과 주처리 알고리즘과 부처리 알고리즘들 사이의 통신 시간이 주요한 부분이며, 일반적으로 집단화 문제의 경우 대부분 입력 패턴 N 의 수가 상당히 크고 상대적으로 군의 갯수 K가 적을 경우이며, 이 경우 각 부처리 알고리즘에서의 집단화를 위한 처리 시간이 주처리 알고리즘과 부처리 알고리즘들 간의 통신 시간보다 훨씬 많아 지며 이

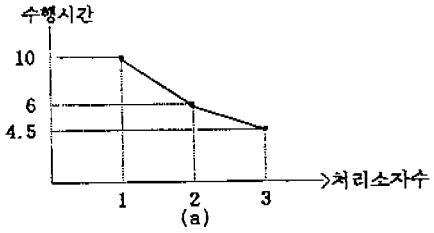


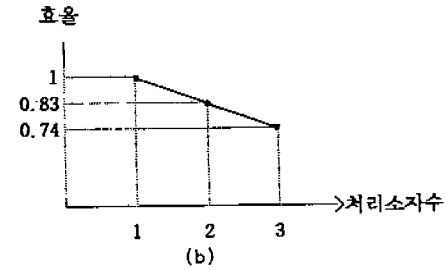
(그림 4) 처리소자 수에 따른 가속율
(Fig. 4) Sn for processing element n

경우 전체 처리 스텝은 LNKm/n이 됨을 알 수 있다. 이로부터 가속율을 계산하면 거의 처리소자의 수와 같음을 알 수있고, 입력 패턴의 수가 많고 군의 수가 적은 일반적 집단화 문제의 경우 메세지 전송형 MIMD 병렬 컴퓨터를 위한 본 알고리즘은 상당히 효율적임을 알 수 있다.

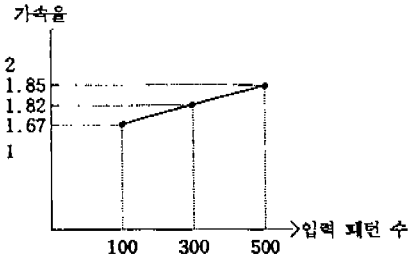
4.3 실험 결과

본 알고리즘은 3개의 트랜스퓨터를 가지고 있는 가속기를 장착한 IBM-PC[8,9]에서 구현되었으며, 알고리즘을 구현한 언어는 병렬 C(parallel ANSI C) 언어[12]를 이용하였다. 1개의 처리소자내에 주처리소자 알고리즘및 부처리소자 알고리즘을 두었으며, 나머지 2개의 처리소자내에 각 1개씩의 부처리소자 알고리즘을 두었다. (그림 4)은 처리소자의 수가 늘어남에 따른 가속율을 도시한 그래프이며, 처리소자에 대한 알고리즘의 수행시간 및 효율을 (그림 5)에 도시하였다. (그림 6)은 두개의 프로세서가 사용되었을 때, 입력 데이타의 수에 대한 가속율을 도시한 것이다. 실험 결과를 통 하여, 입력 패턴의 수가 적을 때는 전체 계산 과정이 통신량에 비해 크지 않으므로 속도 향상이 적고, 입력 패턴의 수와 군의 수가 많을 경우 통신량에 비해 전체 계산 과정이 매우 크기 때문에 사용한 처리소자의 수만큼 속도 향상을 얻을 수 있음을 실험 결과를 통하여 알 수 있었다.





(그림 5(a)) 처리소자 수에 따른 수행시간
(그림 5(b)) 처리소자 수에 따른 효율
(Fig. 5(a)) running time for processing element n
(Fig. 5(b)) efficiency for processing element n



(그림 6) 입력 패턴 수에 따른 가속율
(Fig. 6) Sn for input patterns N

실험 결과에서 보듯이 처리소자가 많아질수록 효율이 낮아지는 이유는 여러가지가 있으나 그중 처리소자간 통신 시간과 처리를 빨리 끝낸 처리소자가 늦게 처리를 끝낸 처리소자를 기다려야 하는 시간이 주된 이유이다. 또한 군의 수가 일정할 경우 입력 패턴의 수가 많아짐에 따라 가속율이 개선되는 이유는 처리소자간 통신 시간은 일정한 반면 상대적으로 각 처리소자내의 처리시간은 많아지기 때문이다. 따라서, 본 알고리즘은 입력 패턴의 수가 많고 군의 수가 적을 경우 유리함을 알 수 있다.

이전에 구현된 병렬 집단화 알고리즘과 비교하여 보면 Li[6]는 본 논문에서 병렬화한 단층 집단화 알고리즘과는 다른 계층적 집단화 알고리즘을 병렬화하였으며, Kim[5]은 본 논문과 같은 단층 집단화 알고리즘을 공유 메모리 구조를 가지는 특수한 병렬 컴퓨터에서 구현하였다. 병렬처리 컴퓨터는 크게 처리소자들이 메모리를 공유할 수 있는 공유 메모리 병렬처리 컴퓨터와 각각의 처리소자가 자신의 메모리를 가지고 처리소자간 통신은 공유 메모리가 아닌 통신 채널을 이용하는, 메세지 전송형 병렬처리 컴퓨터로 분류할 수 있다.

본 논문은 패턴인식용 특수 목적 메모리 공유 병렬처리 컴퓨터에서 이용한 병렬 집단화 알고리즘을 일반적인 메세지 전송 병렬처리 컴퓨터에서 수행할 수 있도록 변경하였으며, 따라서 Kim의 병렬화 알고리즘은 패턴인식을 위한 특수한 공유 메모리형 병렬 컴퓨터에서 알고리즘을 병렬화하

였으므로 공유 메모리형 병렬처리 컴퓨터에서 쉽게 수행될 수 있다는 장점이 있으며, 본 논문은 일반적인 메세지 전송형 MIMD 병렬 컴퓨터에서 구현되었으므로 임의의 메세지 전송형 병렬 컴퓨터에서도 본 병렬 알고리즘은 쉽게 수행될 수 있는 장점을 가지고 있다.

5. 결 론

집단화란 입력 패턴들을 주어진 평가함수를 이용하여 다수의 군으로 분류하는 것이다. 이를 위한 알고리즘은 많으며, K-means 알고리즘은 그 중 하나이다. 이 알고리즘은 입력 패턴의 수가 많을때, 상당한 량의 계산을 필요로 하며 이를 위하여 본 논문에서는 병렬화 알고리즘을 소개하였다. 실험 결과와 성능 분석을 통하여, 입력 패턴의 수가 적을 때는 전체 계산 과정이 통신량에 비해 크지 않으므로 속도 향상이 적고, 입력 패턴의 수와 군의 수가 많을 경우 통신량에 비해 전체 계산 과정이 매우 크기 때문에 사용한 처리 소자의 수 만큼 속도 향상을 얻을 수 있음을 실험 결과를 통하여 알 수 있었다.

참 고 문 헌

[1] Richard O. Duda and Peter E. Hart' Pattern Classification and Scene Analysis', Wiley-Interscience Publication, 1973.

[2] Robert Schalkoff, 'Pattern recognition statistical, structural and neural approaches', Wiley, 1992.

[3] A.H.Karp, "Programming for Parallelism", IEEE Computer, pp. 43-57, May 1987.

[4] R.G.Babb II, 'Programming Parallel Processors', Addision-Wesley, 1988.

[5] Kai Hwang and Dongseung Kim,"Parallel Pattern Clustering on a Multprocessor with Orthogonally Shared Memory", Parallel processing conference, pp. 913-916, 1990.

[6] Xiaobo Li,"Parallel Algorithms for Hierarchical Clustering and Cluster Validity", IEEE, PAMI, Vol. 12, No. 11, pp. 1088-1092, Nov. 1990.

[7] Inmos Limited, 'The Transputer Databook', Inmos, 1989.

[8] Inmos Limited, 'The transputer development and iq systems databook', Inmos, 1990.

[9] Inmos limited, 'IMS B008 IBM PC module motherboard product overview', Inmos, 1990.

[10] K.Hwang and F.A.Briggs, 'Computer Architecture and Parallel Processing', McGraw-Hill, 1984.

[11] K.Hwang, "Multiprocessor Super-computers for Scientific/Engineering Applications",IEEE Computer, June 1985.

[12] Inmos Limited, '3L Parallel C IMS D711 Delivery Manual', Inmos, Feb. 1989.

[13] D.I.Moldovan, 'Modern parallel processing', Univ. of Southern Califonia, Jan. 1986.

[14] D.Tabak, 'Multiprocessors', prentice Hall, 1990.

배 용 근

1984년 조선대학교 전산기 공학과 졸

1984~88년 조선대학교 전자계산소 근무

1986년 조선대학교 대학원 전자과(공학석사)

현재 원광대학교 대학원 전자과(박사과정)

현재 조선대학교 전자계산학과 조교수

관심 분야 : 마이크로 프로세서 응용, 병렬처리 객체지향 분석설계, 멀티미디어.