논문 2007-44CI-3-12

MPI 기반 PC 클러스터에서 GHT의 병렬 분산 구현

(Parallel Distributed Implementation of GHT on MPI-based PC Cluster)

김 영 수*, 김 정 삼*, 최 흥 문**

(Yeong-Soo Kim, Jeong-Sahm Kim, and Heung-Moon Choi)

요 약

MPI(message passing interface) 기반 PC 클러스터 상에서 병렬분산 GHT(generalized Hough transform)를 모델화하고 시간 분석하여 고속화 구현하였다. 파이프라인 방송(pipelined broadcast) 통신방식과 누산기 배열(accumulator array) 분할 처리 정책을 사용함으로써 통신부담을 최대한 줄였고, 전체 처리 과정에 걸쳐 통신과 계산처리를 시간 중첩시켜 구현함으로써 최대한의 속도제고를 하였다. 100 Mbps Ethernet 스위치를 이용하여 MPI 기반 PC 클러스터를 구현하고 제안한 병렬분산 GHT를 실험하여 선형에 가까운 속도 제고율 (speedup)을 확인하였다.

Abstract

This paper presents a parallel distributed implementation of the GHT (generalized Hough transform) for the fast processing on the MPI-based PC cluster. We tried to achieve the higher speedup mainly by alleviating the communication overhead through the pipelined broadcast and accumulator array partition strategy and by time overlapping of the communication and the computation over entire process. Experimental results show that nearly linear speedup is reachable by the proposed method on the MPI-based PC clusters connected through 100Mbps Ethernet switch.

Keywords: GHT, MPI, 클러스터, 병렬처리, 속도 제고율

I. 서 론

투영기반 (projection-based) 변환으로 잘 알려진 GHT는 잡음에 강하며 다른 물체에 겹쳐지거나 일부가 가려진 물체까지도 검출할 수 있는 유용한 알고리즘이긴 하나, 다양한 크기 변화나 회전된 물체까지 검출하기 위한 복잡한 과정 때문에 매우 계산 집중적 (computation intensive)인 알고리즘이다. 따라서 단일 처리기 시스템(uniprocessor system)에서는 너무 긴 처리시간이 소요되므로 병렬 고속처리 기술이 요구되고 있다.

그 동안 HT(Hough transform) 및 GHT의 고속화를 위한 병렬 구현에 대해 많은 연구들[1-12]이 진행되어 왔지만, 대부분이 HT에 관한 연구들[1-7]이며, GHT의 병렬분산처리 (parallel distributed processing)에 관한 연구들[8-12]로서는 단일 컴퓨터 내부의 병렬 하드웨어를 위한 전용 알고리즘[8,9], SIMD 배열 프로세서(array processor)[10], 공유 메모리 다중 프로세서(shared memory multiprocessor)[11], 또는 분산 메모리 다중 컴퓨터(distributed memory multicomputer)[12]를 기반으로 병렬 구현한 예들이 발표되고 있다.

그러나 이들 연구 방법들은 특정 병렬처리 컴퓨터를 위한 전용 알고리즘이므로, 이를 이용하기 위해서 현실적으로 우리 주변에 손쉽게 접근할 수 있는 고성능 병렬처리 컴퓨터를 구하려고 하여도 흔하지 않을 뿐만 아니라, 단일 처리기 시스템 상에 GHT 전용 병렬하드웨어를 구비하는 것도 무척 어려운 일이다. 따라서 최근 여러 연구소나 교육기관에 널리 보급되어 있는 많은

* 정회원, 영남이공대학 컴퓨터정보계열
(Div. of Computer Information, Yeungnam College of Science & Technology)
** 정회원, 경북대학교 전자전기컴퓨터학부
(School of Electrical Engineering & Computer Science, Kyungpook National University)
※ 이 논문은 2005학년도 영남이공대학 연구조성비 지원에 의하여 연구되었음.
접수일자: 2007년1월 24일, 수정완료일:2007년4월30일

PC들과 LAN을 이용하여 MPI 기반 PC 클러스터를 구성하고, 이를 위한 병렬분산 GHT를 개발 구현하면 큰 추가 비용 없이 고속 GHT를 활용할 수 있을 뿐만 아니라 Grid Computing[13]의 기초연구로도 활용 가능할 것으로 본다. 현재까지는 PC 클러스터에서의 구체적인 GHT 병렬 분산 구현을 찾아보기 어렵다.

본 논문에서는 MPI 기반 PC 클러스터 상에 GHT를 병렬분산 구현하여 고속화하고, 그 처리 시간을 분석하였다. 먼저 병렬분산 GHT에서 메시지 지연 및 처리시간을 모델링하여 분석한 후, 목표영상의 전송에 파이프라인 방송(pipelined broadcast)을 이용하고, 영상분할 대신 누산기 배열 분할 처리 정책을 사용함으로써 통신 부담을 최대한 줄였으며, 전체 처리 과정에 걸쳐 통신과 계산처리를 시간 중첩시켜 구현함으로써 최대한 고속화하였다.

100Mbps Ethernet 스위치로 연결된 LAN상에서 MPI 기반 PC 클러스터를 구현하고 제안한 병렬분산 GHT를 구현 실험한 결과, 선형에 가까운 속도제고율을 확인할 수 있었다.

II. 제안한 GHT의 병렬 분산 구현

1. PC 클러스터의 네트워크 토폴로지

제안한 GHT의 병렬분산 구현을 위해 그림 1과 같은 네트워크 토폴로지를 갖는 MPI기반 PC 클러스터를 구현하였다.

Ethernet 스위치를 중심으로 구성된 PC 클러스터에서 한 대의 PC는 마스터(master), 나머지 PC들은 워커(worker) 노드들이다. 모든 PC들이 동등한 계산능력을 가지는 동질의 네트워크(homogeneous network)이다. 워커들은 자신의 워커 프로세스만을 수행하지만, 마스터는 마스터 프로세스와 워커 프로세스를 모두 수행한다. 마스터 프로세스는 작업(task)을 초기화하고, 워커 프로세스들에게 일감(workload)을 분배하고, 일감 처리 결과를 받아 최종적인 결과를 산출하는 등 전체 작업을 관리하는 일을 담당한다. 워커는 마스터로부터 각자의 일감을 분배받아, 상호 독립적으로 일감을 처리하고 그 결과를 마스터에게 전달하는 일을 담당한다.

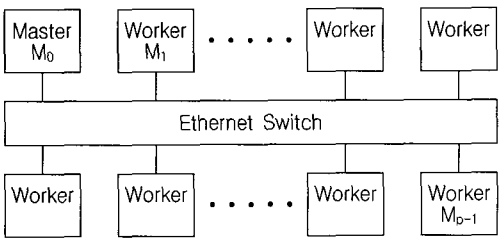

그림 1. GHT의 병렬분산 구현을 위한 PC Cluster의 네트워크 토폴로지

Fig. 1. Network topology of the PC Cluster for parallel distributed GHT implementation.

2. GHT의 병렬분산 구현 모델

기존의 순차처리 GHT와 제안한 병렬분산 처리 GHT에 대한 처리절차를 그림 2(a)와 (b)에 각각 나타내보였다. 그림 2(a)의 순차적 GHT 과정을 본 논문에서는 그림 2(b)에서와 같이 목표영상/R-table 방송, 병렬분산 GHT과정으로 바꾸었으며, 그림 2(a)의 순차적 전역 최고치 계산은 그림 2(b)의 지역 최고치 계산, 지역 최고치 수집, 전역 최고치 계산 과정으로 바꾸었다. 그림 2(b)에서 목표영상/R-table 방송에서부터 전역 최고치 계산까지의 전체 과정은 파이프라인 병렬방식으로 동작시켜 가속화 하였다.

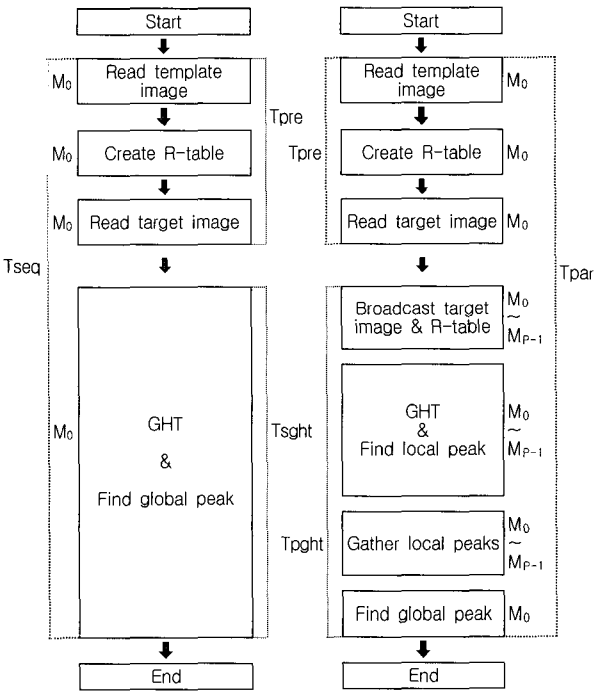

그림 2. 순차처리 GHT와 제안한 병렬분산처리 GHT의 처리절차

Fig. 2. Procedures for the sequential GHT and the proposed parallel distributed GHT.

가. 작업 분할 정책

그림 2(b)와 같은 병렬분산 구현 모델 위에서 GHT

를 고성능으로 처리하기 위하여 구체적이고 효과적인 작업 분할 정책을 찾아내야 한다. 이를 위하여 제일 먼저 기존 순차 GHT의 반복 작업에 대한 분석을 통하여 가능한 작업 분할 정책들을 도출한 다음, 이들 작업 분할 정책들이 속도제고에 미치는 영향에 대한 면밀한 분석 작업이 선행되어야 한다.

먼저 순차 GHT의 반복 작업에 대하여 분석해보면, 허프공간 H(X, Y, S, R)의 누산기 배열에 누적치를 계산할 때의 계산 복잡도는 $O(SREK)$이며, 이 계산 복잡도를 줄이기 위하여 S, R, 또는 E를 분할하여 클러스터의 PC들이 균등 분할 처리하도록 부작업(subtask)을 할당할 수 있다. 여기서 X, Y는 물체 후보 위치의 x, y 좌표, S는 크기, R은 회전각, E는 목표 영상의 에지 점들의 수, K는 템플릿 물체로부터 생성된 R-table의 정보 항목들의 수이다.

첫째, E를 분할하는 것은 목표 영상 분할을 수반하므로 영상 분할 정책이 요구되며, 둘째, S 또는 R을 분할하는 것은 허프공간 H(X, Y, S, R)을 분할하는 것이므로 누산기 배열 분할 정책이 요구된다. 여기서 K는 R-table의 정보로서 분할할 수 없으므로 분할 처리 대상에서 배제된다.

다음으로, 속도제고에 큰 영향을 미치면서 작업 분할 정책을 결정할 때도 반영되는 방송 방식의 선택을 고려하여야 한다. Ethernet 스위치에서는 그 포트들 간에 거점간(point-to-point link) 동시 통신이 이루어지므로, 방송트리(broadcast tree) 구성에 따라 다양한 방송 방식들이 실현 가능하다. 표 1[14]에서와 같이 방송메시지가 충분히 클 경우에는 선형트리(linear tree) 구성 위에서 파이프라인 방송(pipelined broadcast) 방식이 가장 효율적이므로, 초기일감 분배와 작업분할 정책의 결정에도 이를 반영하였다. 여기서 초기일감은 목표 영상과 R-table 정보를 합한 것을 의미하며, 그 크기는 처리 영상의 크기에 좌우된다.

본 연구에서는 효과적인 작업 분할 정책을 결정하기 위하여 영상 분할 정책과 누산기 분할 정책 각각에 대하여 Ethernet스위치 기반 통신시간과 계산 복잡도를 모델링하여 표 2에서와 같이 비교하였다. GHT 계산과 최고치(peak value) 계산을 총괄하는 계산복잡도 측면에서는 두 분할 정책이 큰 차이가 없으나, 초기 일감 분배와 마지막 결과들을 취합하는 통신시간 측면에서는 영상 분할 정책의 $T(N^2SR)$가 누산기 분할 정책의 $T(N^2)$에 비하여 현저히 큰 것을 알 수 있다. 이는 영상 분할 정책이 분할된 영상들로부터의 허프공간의 누적치들을 모든 PC들에 대하여 이웃 PC들 간에 이진트리(binary tree) 형태로 통신하고 합산해야하기 때문이다. 그러므로 본 연구에서는 일감 분배와 결과 취합에서의 통신부담을 최대한 줄이기 위하여, 영상 분할 처리 정책 대신에 누산기 배열 분할 정책을 사용하였다.

표 1. Ethernet 스위칭 클러스터 상에서 충분히 큰 메시지에 대한 각종 방송 알고리즘들의 성능비교[14]

Table 1. Comparison of the broadcast algorithms for sufficiently large messages on the Ethernet switched cluster[14].

Algorithm	Performance
Flat tree	$(p-1)\times T(msize)$
Binomial tree	$\log_2(p)\times T(msize)$
Linear tree (pipelined)	$T(msize)$
Binary tree (pipelined)	$2\times T(msize)$
k-ary tree (pipelined)	$k\times T(msize)$
Scatter/allgather	$2\times T(msize)$

p : Number of processors,
$T(msize)$: Sending time for an $msize$-byte message

표 2. Ethernet 스위치 PC 클러스터 상에서 분할 정책의 모델링과 성능비교

Table 2. Modeling and performance comparison of the partition strategy on the Ethernet switched PC cluster.

Strategy		Image partition	AA partition
Comm.	Distri-bution	$T(\frac{N^2}{p})$	$T(N^2)$
	Gath-thering	$\log_2(p)\times T(N^2SR)$	$(p-1)\times T(C)$
	Subtotal	$T(N^2SR)$	$T(N^2)$
Comp.	GHT	$O(\frac{E}{p}KSR)$	$O(EKS\frac{R}{p})$
	Peak	$\log_2(p)\times O(N^2SR)$	$(p-1)\times O(1)$
	Subtotal	$O(\frac{EKSR}{p})+\alpha$	$O(\frac{EKSR}{p})$

p : Number of processors,
$T(nsize)$: Sending time for an $nsize$-byte message
C : Size of a return message in bytes.
N : Width and depth of an image,
S : Number of scale factors,
R : Number of rotation factors,
E : Number of edge points in the target image,
K : Number of R-table entries,

나. Master/worker 프로세스의 기능

그림 2에서 보는 바와 같이 순차처리와 병렬분산처

리는 공히 템플릿 영상을 읽어 들이고, 이로부터 R-table을 생성하고, 목표 영상을 읽어 들이는 등의 전처리 과정들은 동일하게 진행하지만, GHT 및 최고치 계산과정은 서로 다르다. 순차처리에서는 단일 처리기 시스템이 GHT를 수행하고 이로부터 전역 최고치를 곧바로 찾아내지만, 병렬분산처리에서는 p개의 PC들에게 일감을 배분하여 누산기 배열 분할 정책에 의거하여 GHT 및 지역 최고치 계산을 병렬분산 처리하게 하여 이들을 수합한 다음, 이들로부터 전역 최고치를 찾아낸다.

이들 처리 과정들은 마스터와 워커들의 프로세스들이 상호 협력하여 처리하며, 각 처리 과정마다 그 과정 수행을 담당하는 PC 노드를 M_i, i = 0, 1, 2, ···, (p-1)로 표기하여 그림 2에 표시하였다. 여기서 p는 PC의 대수이다. 마스터 M_0에서의 마스터 프로세스의 기능은 다음과 같다.

Master Process:
1. Reads a template file and stores 2-dimensional array.
2. Creates a R-table list from the edge points of template, and stores it.
3. Reads a target file and stores 2-dimensional array.
4. Broadcasts each worker a target image and a R-table list as the workload.
5. Gathers the local peaks from all the workers.
6. Finds the global peak, returns $X, Y, S,$ and R values.

한편 마스터와 워커들에서의 워커 프로세스들은 균등하게 분할된 GHT 과정을 상호 독립적으로 수행하며 그 기능은 다음과 같다.

Worker Process:
1. Gets a target image and a R-table list from master.
2. Processes the GHT for a ($1/p$) equally-sized accumulator array, and finds a local peak.
3. Sends their own local peak to the master.

다. 병렬분산 GHT의 구현

일반적으로 병렬분산처리 시스템에서는 병렬처리에서와는 달리 속도제고율을 높이기 위해서는 가급적 통신부담을 줄여 단위일감크기(grain size)를 키워야 한다. 즉, 계산 대 통신 비(ratio of computation to communication)를 최대한 증가시켜야 하므로, 본 연구에서는 그림 3의 병렬분산 GHT의 처리시간 분석을 위

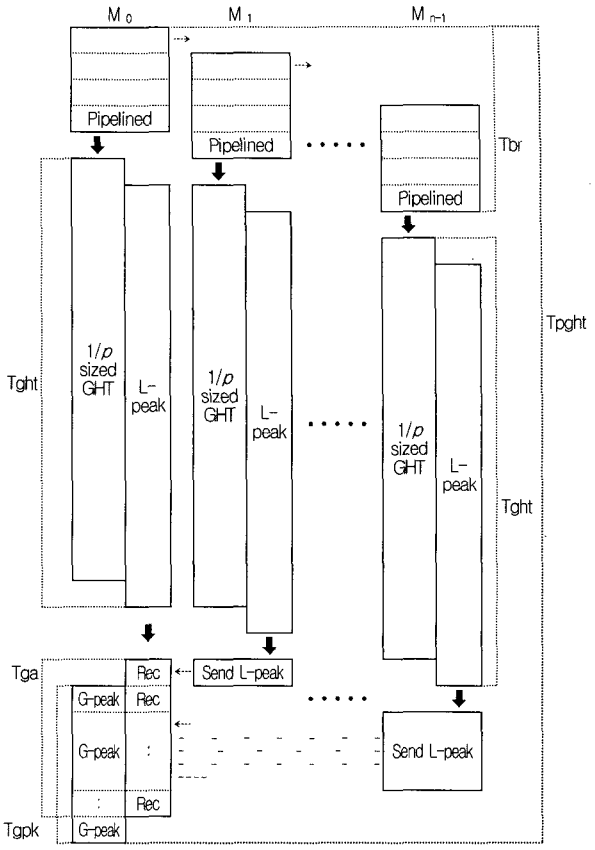

그림 3. 제안한 GHT의 병렬분산 구현을 위한 시간 분석 모델

Fig. 3. Time analysis model for the proposed parallel distributed implementation of the GHT.

한 모델에서 보는 바와 같이, 통신의 병렬 구현, 작업의 병렬분산 구현, 계산과 통신의 오버랩 등 여러 가지 병렬처리 방안들을 활용하였다. 이들은 오버랩 환경을 제공하는 MPI 기능을 이용함으로써 메모리의 추가비용 없이 구현되었다.

(1) 통신의 병렬 구현

PC 클러스터에서 프로세스 간 통신(interprocess communication : IPC)은 PC 외부의 네트워크를 경유하기 때문에 지연시간이 매우 크므로, 이를 줄이는 것이 속도제고율 향상에 매우 중요하다. 이에 본 연구에서는 Ethernet 기반의 PC 클러스터 환경에서 마스터가 워커들에게 초기 일감을 분배할 때 그림 3에서 보는 바와 같이 파이프라인 방송 방식을 도입함으로써 통신지연의 최소화를 시도하였다. Ethernet 스위치를 이용한 파이프라인 방송의 통신시간 T_{br}은

$$T_{br} = (X + p - 1) \times T\left(\frac{msize}{X}\right) \approx T(msize), \quad \text{if } X \gg (p-1) \tag{1}$$

로 계산된다. 이 식에서 p는 클러스터의 PC 대수이며, *msize*는 방송할 전체 메시지의 바이트 수이고, X는 *msize* 메시지를 1024B 고정크기의 세그먼트로 나누어 전송할 때의 전송 회수이므로, $T(\frac{msize}{X})$는 한 세그먼트에 대한 송신 시간이다. 방송 메시지 *msize*가 매우 클 경우 전송회수 X도 큰 값이 된다.

(2) 작업의 병렬분산 구현

허프공간 H(X, Y, S, R)의 크기 S와 회전각 R 중에서 어느 것을 분할 처리해도 speedup 측정에는 영향을 주지 않으므로, S 대신에 R을 균등 분할하여 누산기 분할 정책에 적용하였다. 또한 S와 R의 범위 및 증가량은 GHT의 응용 분야에 따라 적절하게 임의의 값으로 설정될 수 있으나[15], 본 연구에서는 speedup 측정을 위하여 R의 증분값을 1°씩 변화시킴으로써 R의 범위를 360으로 설정하고 그림 3에서 보는 바와 같이 모든 워커 프로세스는 $1/p$로 균등 분할된 R 범위에 걸쳐서 상호독립적으로 GHT를 처리함으로써 최대한 대단위(coarse grained) 작업 분할이 이루어지도록 병렬분산 처리를 구현하였다.

(3) 계산과 통신의 시간중첩

그림 3과 같이 워커 프로세스들은 파이프라인 방송에 의해 차례로 일감을 받고, 받는 즉시 GHT 계산을 진행시켜 통신과 계산을 시간 중첩시켰다. 또한 각 워커 프로세스 내에서 GHT 계산과 지역 최고치 계산을 순차적으로 처리하지 않고, 병행하여 처리하도록 수정하여 불필요한 코드실행을 줄였다. 결과 수집에서는 워커 프로세스마다 특별한 사정에 의해 일감을 받은 순서대로 결과를 출력하지 못할 경우를 대비하여 GHT 계산이 우선적으로 완료되는 워커 프로세스가 먼저 결과를 보내고 이를 마스터가 수집(gathering)하게 함으로써, 일감을 분배했던 순서대로 결과를 수집함으로 인해 GHT 계산을 완료했음에도 불필요하게 결과 전달 차례를 대기하는 프로세스가 없도록 함으로써 통신지연을 최대한 줄였다. 이로써 때 이른 워커 프로세스의 결과 통보와 늦은 워커 프로세스의 GHT 계산이 중첩되도록 처리하여 불필요한 대기시간을 최대한 줄였다. 또한 마스터 프로세스의 결과 수집과 전역 최고치 계산을 병행하여 처리하도록 수정하여 불필요한 코드실행을 줄였다.

3. 병렬분산 GHT의 시간분석

가. 작업처리 시간분석

제안한 병렬분산 GHT의 각 처리과정과 그 작업처리 시간을 도해적으로 분석하기 위하여 그림 2와 그림 3에서 각 작업처리 시간을 표시하였다. 그림 2(a)에 표시된 T_{pre}는 마스터 프로세스의 전처리 시간을, T_{pght}는 병렬분산 처리에 의한 GHT 수행시간을 의미한다. 그림 3은 T_{pght}를 더 세부적으로 분석한 그림이며, 그림 3에 표시된 T_{br}은 마스터 프로세스의 일감의 방송 시간을, T_{ght}는 각 워커 프로세스의 GHT 시작에서 지역 최고치 계산까지의 시간을, T_{ga}는 마스터 프로세스의 결과 수집 시간을 의미하며, T_{gpk}는 마스터 프로세스의 전역 최고치의 계산 시간을 의미한다. 그리고 이들은 다음과 같이 수식으로 표현할 수 있다.

$$T_{pre} = 2 \times T_{gr} + T_{rt} \tag{2}$$

$$T_{br} = (X + p - 1) \times T(\frac{msize}{X}) \tag{3}$$

$$T_{ght} \approx (\frac{1}{p}) \times T_{sght} \tag{4}$$

$$T_{ga} = (p-1) \times T(C) \tag{5}$$

$$T_{gpk} \approx T_{ga} \tag{6}$$

여기서, 식(2)의 전처리 시간 T_{pre}는 그림 2에서 보는 바와 같이 템플릿과 목표 영상을 읽어 들이는데 걸리는 시간 ($2\times T_{gr}$) 및 템플릿 영상으로부터 R-table을 생성하는데 걸리는 시간 T_{rt}를 합한 시간이며, T_{gr}은 영상의 크기에 비례하고 T_{rt}는 영상 크기 및 템플릿 물체에지 점들의 수에 비례한다. 식(3)의 파이프라인 방송시간 T_{br}은 일감 크기가 정수배 증가하는 경우에는 거의 비례하여 증가하지만, PC 대수가 정수배 증가하는 경우에는 비례하여 증가하지 않고 거의 일정하다. 이는 식(3)에서 일감 크기에 비례하는 X가 PC 대수 p보다 큰 값이므로 X가 수식의 값을 지배하기 때문이다. 식(4)의 병렬분산처리 GHT 시간 T_{ght}는 순차처리 GHT 시간 T_{sght}의 ($1/p$) 시간에 근사한다. 식(5)의 결과 수집 시간 T_{ga}는, 하나의 워커가 40바이트 고정크기의 지역 최고치 관련 정보 [x, y, S, R, *peak value*]를 unicast로 반환하는 시간이 $T(C)$이므로 (p-1)에 비례하며, $T(C)$는 그림 3에서 보는 바와 같이 거의 $(1/p)\times T_{ga}$가 된다.

식(6)의 마스터 프로세스에서의 전역 최고치에 대한 계산 시간 T_{gpk}는 이 과정이 결과 수집 과정과 병행하므로 그림 3에서 T_{ga}와 거의 같은 값으로 표시되었지만, 실제로 $(1/p)\times T_{gpk}$ 시간은 거의 0에 근접한다.

나. 속도제고율

그림 2에서 보는 바와 같이 단일 처리기 시스템에서의 순차처리 GHT의 수행시간 T_{seq}는

$$T_{seq} = T_{pre} + T_{sght} \tag{7}$$

로 표현할 수 있다. 한편 p개의 PC들로 구성된 클러스터에서의 병렬분산처리 GHT의 수행시간 T_{par}은

$$T_{par} = T_{pre} + T_{pght} \tag{8}$$

로 주어진다. 여기서 T_{pght}는 그림 3에 표시한 바와 같이 전처리 과정을 제외한 순수한 병렬 GHT 수행과정이며,

$$T_{pght} = T_{br} + T_{ght} + T_{ga} + T_{gpk} - T_{ov1} - T_{ov2} \tag{9}$$

로 주어진다. 여기서 T_{ov1}은 T_{ght}와 T_{ga}의 중첩 시간이며, T_{ov2}는 T_{ga}와 T_{gpk}의 중첩 시간이다. 그러므로 T_{pght}를 근사화하면

$$T_{pght} \approx T_{br} + (1/p)\,T_{sght} + (1/p)\,T_{ga} + (1/p)\,T_{gpk} \tag{10}$$

이 된다.

따라서 제안한 병렬분산 GHT의 속도제고율 Sp는 식(7), (8) 및 (10)으로부터

$$Sp = \frac{Tsght + Tpre}{(\frac{1}{p})Tsght + Tpre + Tbr + (\frac{1}{p})Tga + (\frac{1}{p})Tgpk} \tag{11}$$

로 정의된다. 충분한 일감이 주어진 상태에서는 T_{sght}에 비해 T_{pre}, T_{br}, T_{ga} 및 T_{gpk} 등은 아주 작은 값이므로 속도제고율 Sp는 점근해석 (asynmptotic analysis)하여

$$Sp = O(p)$$

로 근사화될 수 있다. 따라서 제안한 병렬분산 GHT 알고리즘은 어느 한계의 p까지는 거의 선형적으로 p에 비례하는 속도제고율을 기대할 수 있다.

III. 실험 및 고찰

실험에 사용된 PC들은 3.0GHz P4 프로세서와 512MB 메모리를 가진 HP DX6120MT이다. 모든 PC들은 Windows XP Professional을 실행한다. 각 PC의 Ethernet 카드는 100Mbps 속도와 full duplex 동작 모드를 가진 Broadcom 5751이다. 클러스터의 스위치는 CISCO Catalyst 2960 (48port 100Mbps Ethernet switch)이며, full duplex 모드로 동작한다. 미들웨어(middleware)는 MPICH 1.2.5이며, 이와 관련된 관리도구들을 사용하였다. 병렬분산 GHT 알고리즘을 위한 응용프로그램은 Microsoft Visual C++로 구현하였다.

실험에 사용된 영상들은 그림 4에 예시한 바와 같이 템플릿과 목표 영상 각각에 대하여 128×128, 256×256,

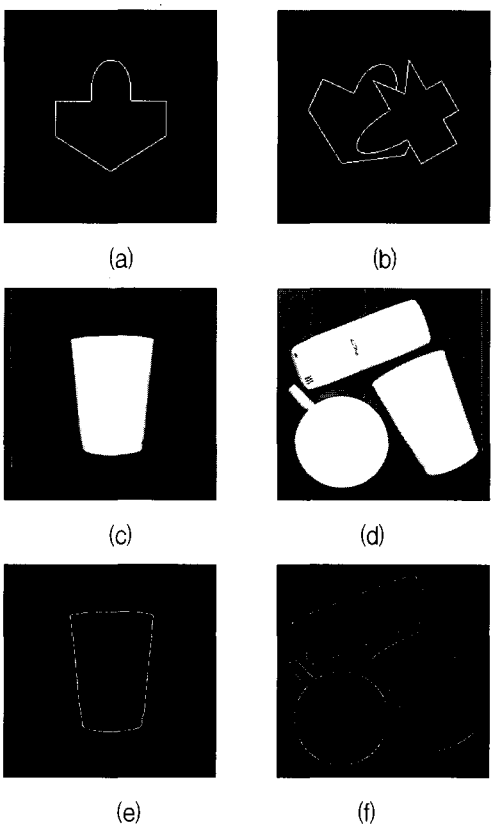

그림 4. 실험에 사용된 인공영상과 실영상의 예:
(a) 템플릿 (b) 목표 인공영상 (이진, 256×256)
(c) 템플릿 (d) 목표 실영상 (회색, 512×512)
(e) 템플릿 (f) 목표 실영상(이진, 512×512)

Fig. 4. Example of the artificial and real images used in the experiment.
(a) Template (b) Target for artificial image (binary, 256×256) (c) Template (d) Target for real image (gray, 512×512) (e) Template (f) Target for real image (binary, 512×512).

표 3. 실험에 사용된 일감에 대한 방송 메시지 크기
Table 3. Broadcast message size for workload used in experiment.

Image category		Artificial			Real
Target image	Image size	128×128	256×256	512×512	512×512
	Broadcast message size	16KB	64KB	256KB	256KB
	Number of edge points	375	752	1501	2192
R-table	Number of edge points	210	414	828	879
	1 / (total number of pixels)	1.28%	0.63%	0.32%	0.34%
	Broadcast message size	<4KB	<8KB	<16KB	<16KB

표 4. 프로세서 수 변화에 따른 실행 시간 [초]
Table 4. Execution time for the variable number of processors. [sec]

Category	Artificial			Real
Size / p	128×128	256×256	512×512	512×512
1	23.100	91.360	361.635	556.454
2	11.560	45.703	180.907	278.306
4	5.790	22.870	90.486	139.141
8	2.914	11.467	45.426	69.655
16	1.494	5.884	23.370	35.827
32	0.790	3.080	12.303	18.760
48	0.541	2.073	8.169	12.524

512×512 인공 영상(artificial image)들과 512×512 실영상(real image)이다. 영상크기 변화가 속도제고율에 미치는 영향을 알아보기 위하여 세 가지 템플릿 인공 영상들과 세 가지 목표 인공 영상들 각각은 공히 같은 모양의 1 픽셀 두께의 윤곽선을 가지며, 영상의 폭과 높이가 2배씩 커짐에 따라 그 에지 점들의 수는 거의 2배씩 커지도록 만들었다.

표 3은 각 영상에 대하여 목표 영상과 R-table의 에지 점들의 수 및 방송 메시지 크기를 나타내었다. 일감에 대한 방송 메시지의 전체 크기는 목표 영상과 R-table의 방송 메시지 크기를 합한 값이다.

표 4는 표 3에 나타난 각 영상 크기 및 프로세스 수의 변화에 따른 실행 시간을 나타내었다. 표 4의 각 실행시간 값은 두 번 실험하여 평균을 구한 값이며, 각 PC에 별도의 작업부하가 없는 실험실 환경에서 실험하였으므로 실험 편차는 거의 없었다.

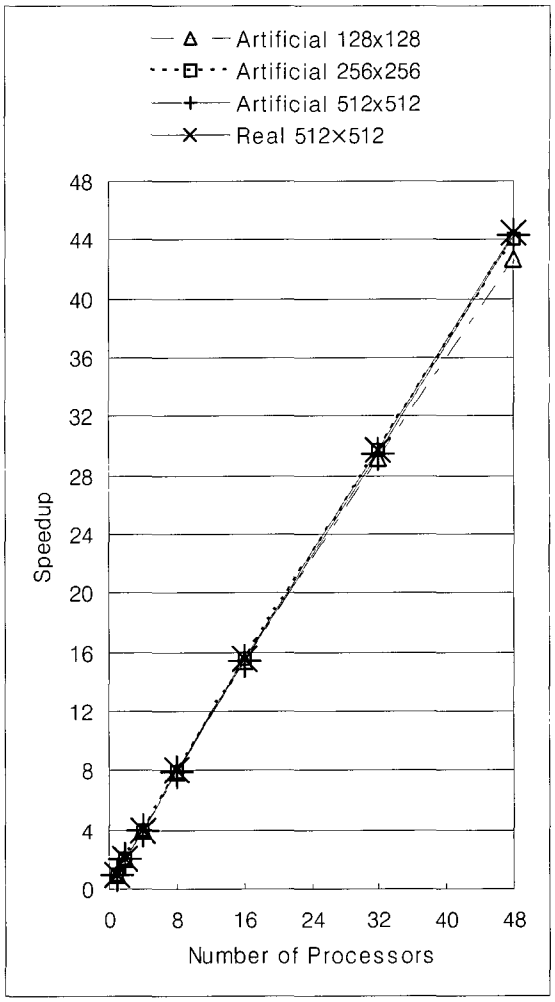

그림 5. 제안한 GHT 알고리즘에 대한 속도제고율
Fig. 5. Speedup for the proposed GHT algorithm.

그림 5는 인공영상과 실영상을 사용하여 제안한 병렬분산 GHT에 대한 속도제고율을 그래프로 나타내었다. 어느 영상에서나 노드 수가 증가함에 따라 통신시간이 증가하므로 속도제고율은 감소된다. 영상 크기가 클수록 단위일감 크기가 증가하여 속도제고율이 더 우수한 것으로 나타났다. 이는 Ethernet 스위치 기반 클러스터에서 보다 큰 메시지 전송에 우수한 파이프라인 방송 전송방식이 통신시간을 줄인 것으로 분석된다. 128

512×512 실영상은 동일한 크기의 인공 영상보다 템플릿과 목표 영상의 에지 점들의 수가 각각 6.2%, 46.0% 정도 더 많으므로 단일 프로세서의 순차적 GHT 계산시간이 53.9% 정도로 더 큰 값을 나타내었지만, 병렬분산 GHT에 대한 속도제고율은 측정 결과 동일크기의 인공영상과 거의 같은 성능을 나타내었다.

Ⅳ. 결 론

MPI-기반 PC 클러스터에서 병렬분산 GHT를 모델링하고 시간 분석한 후 고속화하여 구현하였다. 통신부담을 가급적 줄이고 통신과 계산을 최대한 시간 중첩시켜 병렬분산 처리하였다. 일감 배분과 결과 취합에서의 통신부담을 최대한 줄이기 위하여 영상 분할 정책 대신에 누산기 배열 분할 정책을 사용하였고, 또한 일감 분배에 있어서 전송 메시지가 클수록 효율적인 파이프라인 방송을 사용하였다. 100Mbps Ethernet 스위치에 연결된 PC 클러스터에서 실험한 결과, 작은 영상보다 큰 영상에서 더 높은 속도제고율을 얻을 수 있었으며, 제안한 병렬분산 GHT 알고리즘은, 512×512 영상에 대하여 48포트 스위치에서 이용 가능한 p=48까지의 실험에서 거의 선형 속도 제고율을 확인하였다. 향후 여러 대의 스위치들을 캐스케이드(cascade)로 연결한 대규모 Ethernet 스위치 환경 및 인터넷 기반 그리드 환경에서의 병렬분산처리에 의한 speedup 분석이 연구과제로 남아있다.

참 고 문 헌

[1] T. M. Silberg, "The Hough transform on the geometric arithmetic parallel processor," *Proc. IEEE Comput. Soc. Workshop Comput. Arch. Pattern Anal. Image Database Manag*, pp. 387-393, Nov. 1985.

[2] C. Guerra and S. Hambrusch, "Parallel algorithms for line detection on a mesh," *Journal of Parallel and Distributed Computing Archive*, vol. 6, no. 1, pp.1-19, Feb 1989.

[3] Y. Pan and Y. H. Chuang, "Parallel Hough transform algorithms on SIMD hypercube arrays," *Proc. of ICPP*, vol. 3, pp. 83-86, Aug 1990.

[4] M. Atiquzzaman, "Pipelined implementation of the multiresolution Hough transform in a pyramid multiprocessor," *Pattern Recognition Letters*, vol. 15, no. 9, pp. 841-851, Sep 1994.

[5] A. N. Choudhary and R. Ponnusamy, "Implementation and evaluation of Hough algorithms on a shared-memory multiprocessor," *Journal of Parallel and Distributed Computing*, vol. 12, no. 2, pp. 178-188, June 1991.

[6] A. Underhill, M. Atiquzzaman, and J. Ophel, " Performance of the Hough transform on a distributed memory multiprocessor," *Microprocessors and Microsystems*, vol. 22, no. 7, pp. 355-362, Jan 1999.

[7] N. Guil and E. L. Zapata, "A parallel pipelined hough transform," *Euro-Par*, vol. II, pp. 131-138, Aug 1996.

[8] Meribout, M., Nakanishi, M., and Ogura, T., "A parallel algorithm for real-time object recognition," *Pattern Recognition*, vol. 35, no. 9, pp. 1917-1931, Sep 2002.

[9] R. Strzodka, I. Ihrke, and M. Magnor. "A Graphics Hardware Implementation of the Generalized Hough Transform for fast Object Recognition, Scale, and 3D Pose Detection," *Proc. of IEEE International Conference on Image Analysis and Processing (ICIAP'03)*, pp. 188-193, 2003.

[10] Z. Li, B. Yao, and F. Tong. "A linear generalized hough transform and its parallel implementation," *CVPR*, pp. 672.-673, June 1991.

[11] T. Achalakul and S. Madarasmi, "A concurrent modified algorithm for Generalized Hough Transform," *Proc. of IEEE International Conference on Industrial Technology (ICIT'02)*, vol. 2, pp. 965-969, Dec 2002.

[12] D. Baumann and S. Ranka. "The Generalized Hough Transform on an MIMD Machine," *Journal of Undergraduate Research in High-Performance Computing*, 2, 1992.

[13] N. Sanguandikul and N. Nupairoj, "Implicit Information Load Sharing Policy for Grid Computing Environment," *The 8th International Conference on Advanced Communication Technology (ICACT 2006)*, vol. 3, Feb 2006.

[14] P. Patarasu, A. Faraj, and X. Yuan. "Pipelined Broadcast on Ethernet Switched. Clusters." *The 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, Apr 25-29, 2006.

[15] K. C. Wong, H. C Sim, and J. Kittler, "Recognition of two dimensional objects based on a novel generalized Hough transform method.", *Proc. of International Conference on Image Processing*, vol. 3, pp.376-379, 23-26 Oct. 1995.

저 자 소 개

김 영 수(정회원)
1980년 경북대학교 전자공학과 학사 졸업
1982년 경북대학교 전자공학과 석사 졸업
1999년 경북대학교 전자공학과 박사 수료
1986년~현재 영남이공대학 컴퓨터정보계열 교수
<주관심분야 : Parallel Distributed Processing, Computer Networks, Ubiquitous & Embedded System>

최 홍 문(정회원)
제39권 CI편 제1호 참조
현재 경북대학교 전자전기컴퓨터학부 교수

김 정 삼(평생회원)
1987년 경북대학교 전자공학과 학사 졸업
1990년 경북대학교 컴퓨터공학과 석사 졸업
1998년 경북대학교 컴퓨터공학과 박사 수료
2002년~현재 영남이공대학 컴퓨터정보계열 조교수
<주관심분야 : Wireless Networks, Ad-hoc Networks, Wireless MAC Protocols, Wireless PAN, USN/RFID,>