중소 제조업체의 생산정보 관리시스템을 위한 IOCP 기반 서버

임 성 락* · 송 기 석**

A IOCP-based Server for Product Information Management System of Small and Medium Size Manufacturing Companies

SeongRak Rim* · Kiseok Song**

Abstract

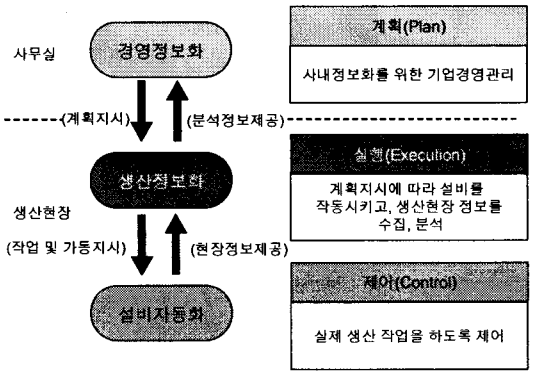
In order to keep global competitiveness, most of small and medium size manufacturing companies require to equip a product information management system which collects and analyzes the data generated at the manufacturing lines and then provides information for manager or worker to make a decision. However, these companies have a cost problem for adopting the enterprise resource planning system mostly used in large companies. To overcome this problem, we suggest an IOCP-based server for the product information management system suitable for small and medium size manufacturing companies. The basic concept of suggested server is that it is possible to process concurrently the connection requests coming from data collector and client by using the facility of asynchronous notification of OS and protect a server against overload by using the thread pool.

Keywords : Product Information, IOCP

논문접수일 : 2007년 07월 30일 논문게재확정일 : 2007년 10월 08일
※ 이 논문은 2006년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임(20060105).
* 호서대학교, 정교수, e-mail : srrim@office.hoseo.ac.kr
** 호서대학교 메카트로닉스공학과 석사과정, (336-795) 충남 아산시 배방면 세출리 호서대학교 컴퓨터공학과

1. 서 론

최근의 시장 환경은 고객 요구의 다양화와 기술의 급속한 발전으로 제품의 생명주기가 급격히 단축되고 글로벌 경쟁의 출현으로 국경을 초월한 경쟁이 가속화 되고 있다. 이러한 변화로 인해 대부분의 중소 제조업체들은 과거의 소품종 대량생산의 체제에서 다품종 소량생산 제체로 변화되고 있다. 또한 고도로 발전된 컴퓨터를 이용하여 생산라인을 탄력성 있게 운영함으로써 모든 생산 및 업무 프로세스는 갈수록 더욱 복잡해지고 있는 추세이고, 기존의 관리시스템으로는 이를 해결하지 못함에 따라 보다 혁신적이고 합리적인 관리 체계의 도입이 시급하게 되었으며, 그 중요성 역시 더욱 강조되고 있는 실정이다. 이와 같은 시장 환경의 변화로 오늘날 대부분의 중소 제조업체들은 생산정보 관리시스템을 요구하고 있다[Sang In Han, 2001; Smit. K., 1989]. 생산정보 관리시스템은 <그림 1>과 같이 기업 경영자로부터 지시를 받아 설비들을 작동시키고, 생산 현장에서 발생하는 현장정보를 수집/분석하여 기업 경영자에게 분석정보를 제공함으로써 경영정보화를 지원하는 시스템을 의미한다[중소기업정보화경영원, 2005; 차석근, 2004].



〈그림 1〉 생산정보화

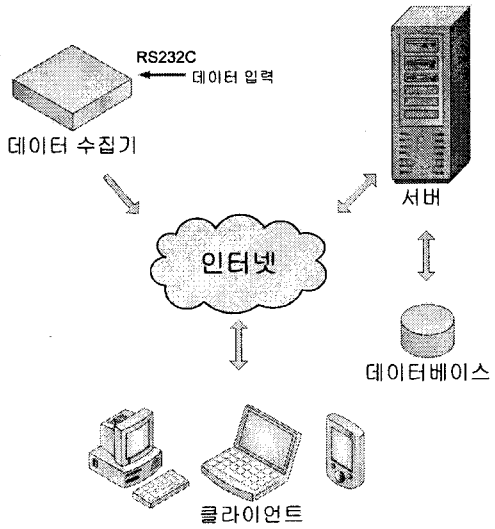
중소 제조업체들의 생산정보화를 위하여 대기업에서 사용되고 있는 기존의 상용 모델을 적용할 경우, 구조적으로 너무 복잡함은 물론 업체마다 생산제품 및 규모의 다양성 때문에 비용적인 어려움이 있다. 본 논문에서는 중소 제조업체에 적합한 생산정보 관리시스템을 위한 서버를 제시하고자 한다.

2. 시스템 설계

2.1 시스템 요구사항

기업 경영자로부터 지시를 받아 설비들을 작동시키고, 생산현장에서 발생하는 현장정보를 수집/분석하여 기업 경영자에게 분석정보를 제공하는 생산정보 관리 시스템은 <그림 2>와 같이 크게 데이터 수집기, 클라이언트, 서버의 3부분으로 구성된다.

<그림 2>에서 데이터 수집기는 생산 현장에서 작업자 혹은 센서에 의해 입력된 생산 데이터를 수집하여 서버로 보내는 기능을 담당한다. 생산현장에서 수집되어야할 데이터 종류는 제



〈그림 2〉 시스템 구성도

조업체의 생산라인에 따라 다양하겠지만 데이터 수집기의 기능은 동일하다. 한편 데이터 수집기는 생산 데이터가 입력되는 모든 현장에 반드시 설치되어야 하기 때문에 그 수는 생산 규모에 비례하게 된다. 따라서 데이터 종류 및 생산규모에 무관하게 데이터 수집기로부터 전송된 생산 데이터를 처리할 수 있는 서버가 요구된다.

클라이언트는 경영자 및 작업자의 의사결정에 필요한 생산정보 접근을 지원하는 기능을 담당한다. 본 논문에서는 의사결정에 필요한 생산정보에 접근하는 경영자 및 작업자의 편의성 및 범용성을 고려하여 범용 웹 브라우저를 이용하도록 한다. 따라서 웹 브라우저를 통한 생산정보 요청을 처리할 수 있는 웹 서비스를 지원할 수 있는 서버가 요구된다.

마지막으로 서버는 데이터 수집기로부터 전달된 데이터를 데이터베이스에 저장하고 클라이언트의 요청에 따라 생산 데이터를 생산정보로 변환하여 응답하는 기능을 담당한다.

유한한 자원을 가진 시스템에서 이상적인 서버가 갖추어야할 요구사항은 다음과 같다[김선우, 2004].

- 모든 클라이언트 접속이 성공한다.
- 서버는 각 클라이언트의 서비스 요청에 최대한 빠르게 반응하며 고속으로 데이터를 전송한다.
- 시스템 자원 사용량을 최소화한다.

또한, 이상적인 서버를 위한 소켓 입출력 모델의 요구사항은 다음과 같다.

- 소켓 함수 호출 시 블로킹을 최소화한다.
- 입출력 작업을 다른 작업과 병행한다.
- 스레드 개수를 최소화한다.
- 유저모드와 커널모드 전환 횟수와 데이터 복사를 최소화한다.

윈도우즈에서는 Select 모델, WSAAsyncSelect 모델, WSAEventSelect 모델, Overlapped 모델, IOCP(I/O Completion Port) 모델을 지원하고 있다. 이 모델들의 성능 비교는 <표 1>과 같으며 IOCP 모델은 CPU 사용량이 적으면서도 연결시도/연결성공이 가장 높음을 알 수 있다[Anthony Jones 외 1인, 2003].

IOCP 모델의 장점은 작업 스레드의 개수를 CPU 개수에 비례하여 유지하고 입출력 완료시

〈표 1〉 윈도우즈 소켓 입출력 모델별 성능 비교

IO모델	연결시도/연결성공	메모리 사용양(KB)	Non-paged pool	CPU사용량	스레드 개수	처리량(Send/Receive Byte per Second)
Selecet	7,000/4,011	4,208	135,123	95~100%	1	0/0
	12,000/5,779	5,224	156,260	95~100%	1	0/0
WSAAsyncSelect	7,000/1,956	3,640	38,246	75~85%	3	1,610,204/1,637,189
	12,000/4,077	4,884	42,992	90~100%	3	652,902/652,297
WSAEventSelect	7,000/6,999	10,502	36,402	65~85%	133	4,921,350/5,186,297
	12,000/11,080	19,214	39,040	50~60%	192	3,217,493/3,217,493
	46,000/45,933	37,392	212,624	80~90%	791	3,851,059/3,851,059
Overlapped	7,000/5,558	21,844	34,944	65~85%	66	5,024,723/1,803,878
	12,000/12,000	60,576	48,060	35~45%	195	1,803,873/1,803,878
	49,000/48,997	241,208	155,480	85~95%	792	3,865,152/3,834,511
IOCP	7,000/7,000	36,160	31,128	40~50%	2	6,282,473/3,893,507
	12,000/12,000	59,256	38,862	40~50%	2	5027,914/5,027,095
	50,000/49,997	242,272	148,192	55~65%	2	4,326,946/4,326,496

에 LIFO(Last-In First-Out) 방식으로 가장 최근에 대기 중인 작업 스레드에게 스케줄링하기 때문에 문맥 교환이 발생하지 않는다. 블로킹을 줄이기 위해 운영체제의 소켓 버퍼를 거치지 않고 사용자 주소공간의 버퍼를 직접 이용하여 입출력을 수행한다[J. Richter 외 2명, 2003]. 또한 IOCP 모델은 스레드 풀을 사용하여 작업의 효율성을 높인다. 스레드 풀은 어떤 작업을 처리하는데 있어 매번 스레드를 생성하는 것이 아니라 몇 개의 스레드를 미리 생성한 후 사용하는 스레드 기법을 의미한다. 이런 기법의 장점은 매번 스레드를 생성하고 파괴하지 않으므로 스레드 생성과 파괴에 대하여 과부하가 없다는 점이 있다[배창우 외 3인, 2005].

이와 같이 IOCP 모델은 클라이언트의 수가 증가하더라도 CPU 부하에 영향을 덜 받으므로 확장성 면에서 가장 뛰어나며 서버의 처리 성능을 극대화 시켜 하드웨어 비용을 감소시키는데 크게 기여하고 있다[Anthony Jones 외 1인, 2003; Gyu-bake Kim, 2003].

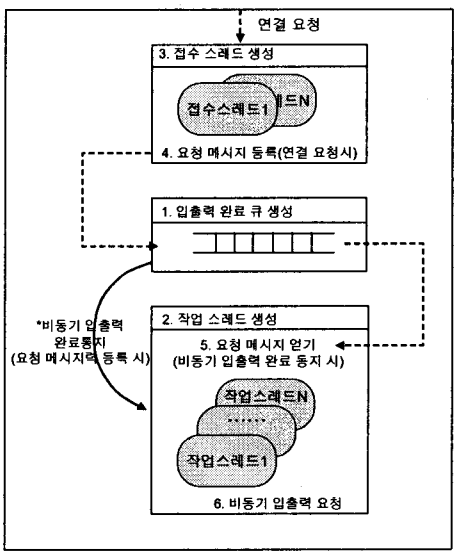
이러한 장점을 가지고 있는 IOCP 모델은 이상적인 서버의 요구사항을 가장 만족하기 때문에 본 논문에서는 생산 관리시스템을 위하여 IOCP를 기반으로 한 서버를 설계한다.

2.2 IOCP 기반의 서버

생산정보 관리시스템을 위한 서버는 기본적으로 데이터 수집기와 클라이언트를 통하여 접수된 생산정보 요청을 동시에 처리할 수 있어야 한다.

이 두 가지 서비스를 효과적으로 지원하기 위하여 <그림 3>과 같은 입출력 완료 큐를 기반으로 한 비동기 입출력 방식의 다중 스레드 모델을 제시한다.

<그림 3>에서 서버는 하나의 입출력 완료 큐



〈그림 3〉 서버의 구조

를 중심으로 다수의 접수 스레드와 작업 스레드로 구성된다. 접수 스레드는 데이터 수집기 혹은 클라이언트로부터의 연결 요청을 접수하여 연결된 소켓 정보를 입출력 완료 큐에 등록시킨다. 생산정보 관리를 위한 서버는 생산 현장에서 발생하는 데이터를 정확히 수집하기 위하여 데이터 수집기의 연결요청만을 전담하는 접수 스레드를 유지한다. 한편 작업 스레드는 입출력 완료 큐에 등록된 소켓 정보에 해당되는 요청을 처리한다. 여기에서 작업 스레드들은 비동기 입출력 함수를 호출한 후 입출력 작업의 완료 여부와 무관하게 다른 작업을 진행함으로써 처리율을 향상시킬 수 있다. 이를 위해서는 입출력 작업의 완료를 작업 스레드에게 알려주는 운영체제의 비동기 통지 기능이 요구된다.

결국, 본 논문에서 제시한 서버 모델의 기본 개념은 운영체제의 비동기 통지 기능을 이용하여 연결 요청만을 전담하는 다수의 접수 스레드와 요청 서비스 처리만을 전담하는 다수의 작업 스레드들이 비동기적으로 진행됨으로써 데이터 수집기를 통하여 접수된 생산 데이터와 클라이

언트를 통하여 접수된 생산정보 요청을 보다 효과적으로 처리할 수 있도록 하는 것이다. 또한 일정한 개수의 접수 및 작업 스레드 풀을 사용함으로써 중소 제조업체들의 규모에 따라 비례적으로 증감하게 되는 데이터 수집기의 연결 요청을 효과적으로 처리할 수 있도록 하는 것이다.

(1) 접수 스레드

데이터 수집기 및 클라이언트로부터의 연결 요청을 접수하여 연결된 소켓 정보를 입출력 완료 큐에 등록시키기 위한 접수 스레드의 흐름은 <그림 4>와 같다.

<그림 4>에서 접수 스레드는 socket(), bind(), listen()을 호출하여 소켓을 초기화 하고, accept()를 호출함으로써 대기상태에서 데이터 수집기 및 클라이언트로부터 연결 요청을 기다린다. 결국, 접수 스레드는 연결 요청이 성공할 때마다 대기상태에서 깨어나 접속된 소켓과 입출력 완료 큐를 연결한 후, 접속된 소켓 정보를 포함한 요청 메시지를 생성하여 이를 입출력 완료 큐에 등록하는 작업을 반복한다.

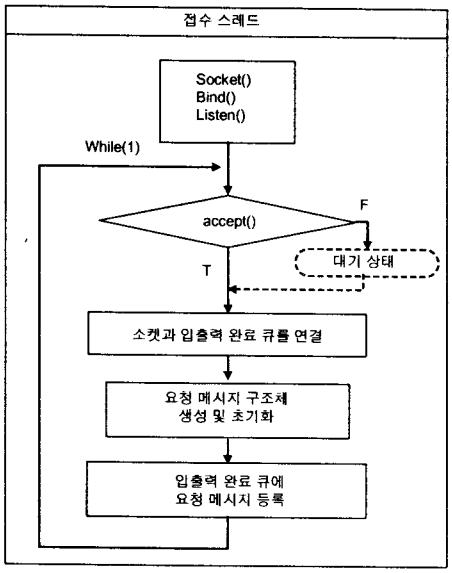
(2) 작업 스레드

입출력 완료 큐에 등록된 요청 메시지를 처리하기 위한 작업 스레드의 흐름은 <그림 5>와 같다.

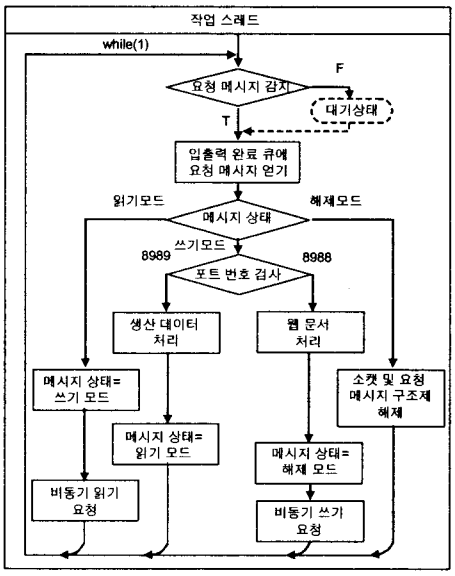
<그림 5>에서 작업 스레드는 입출력 완료 큐의 상태를 검사한 후, 큐가 비어 있을 경우 대기 상태에서 기다린다. 새로운 요청 메시지가 입출력 완료 큐에 도착 할 때마다 대기상태에서 깨어나 입출력 완료 큐에 등록된 요청 메시지를 처리하는 작업을 반복한다.

본 논문에서는 요청 메시지를 보다 효과적으로 처리하기 위하여 입출력 완료 큐에 존재하는 메시지의 상태를 다음과 같은 세 가지 모드(읽기, 쓰기, 해제)로 구분한다.

- 읽기모드 : 소켓 수신버퍼에서 데이터 읽기.
- 쓰기모드 : 소켓 송신버퍼에 데이터 쓰기.
- 해제모드 : 소켓 및 메시지 구조체 해제.



〈그림 4〉 접수 스레드 흐름도



〈그림 5〉 작업 스레드 흐름도

접수 스레드에 의해 생성된 메시지의 초기상태는 읽기모드로 입출력 완료 큐에 등록된 후 작업 스레드에 의해 처리된다. 읽기모드일 경우, 쓰기모드로 변환한 후 소켓 수신버퍼에서 데이터를 읽기 위한 비동기 읽기 요청을 한다. 비동기 읽기작업이 완료될 때 입출력 완료 큐에 쓰기모드로 등록된다. 쓰기모드일 경우 데이터 수집기와 클라이언트의 요청을 구별하여 처리한다. 데이터 수집기의 경우, 접수된 생산 데이터를 데이터베이스에 저장하고, 메시지 모드를 읽기모드로 변경한다. 반면, 클라이언트의 경우, 응답 메시지를 생성하고 메시지 모드를 해제모드로 변경한 후 비동기 쓰기 요청을 한다. 비동기 쓰기작업이 완료될 때 입출력 완료 큐에 해제모드로 등록된다. 해제모드일 경우 서버에서 더 이상 사용하지 않는 소켓 및 메시지 구조체를 해제한다.

3. 구현 및 실험

본 논문에서 제시한 생산정보 관리시스템을 위한 IOCP 기반의 서버의 타당성을 평가하기 위하여 <표 2>와 같은 환경에서 기본적인 모듈들을 구현하여 실험하였다.

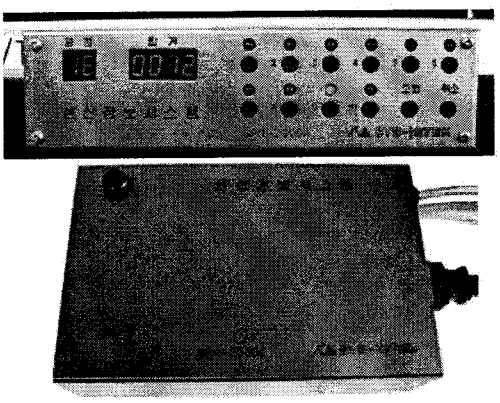
〈표 2〉 개발 및 사용 환경

		운영체제	사용 도구	하드웨어 사양
데이터수집기		Linux	ARM-GCC	임베디드 보드
클라이언트		윈도우즈	Explorer	CPU 2.8GHz RAM 1GB
서버	PC-1,	윈도우즈 NT계열	Visual Stdio.Net	CPU 3.0 GHz RAM 2GB
	PC-2	윈도우즈 NT계열	Visual Stdio.net	CPU 2.4GHz RAM 1GB
데이터베이스		윈도우즈 2000	MySQL	CPU 2.0GHz RAM 1GB

3.1 구현

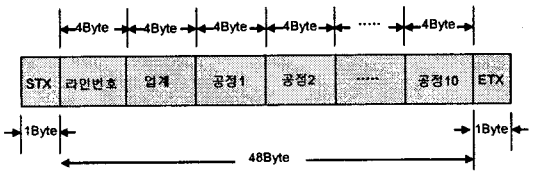
(1) 데이터 수집기

<그림 6>과 같은 생산 데이터 입력 장치로부터 생산 데이터를 받아 서버로 전송하기 위하여 데이터 수집기를 임베디드 리눅스 시스템에서 구현하였다.



〈그림 6〉 생산 데이터 입력 장치

데이터 수집기는 RS232C 통신을 통하여 생산 데이터 입력 장치로부터 생산 데이터를 받아 <그림 7>과 같은 생산 데이터 메시지 프레임을 생성한 후 매초마다 서버로 보낸다.



〈그림 7〉 생산 데이터 메시지 프레임

S T X : 메시지 프레임의 시작. 'ㄱ' 입력.
E T X : 메시지 프레임의 끝. 'ㄴ' 입력.
라인번호 : 입력이 들어오는 공정기계 번호.
합 계 : 모든 공정의 데이터의 합(오류 체크)
공 정 : 실제 공정의 데이터.

(2) 서버

1) IOCP 및 작업 스레드 생성

CreateCompletionPort()와 CreateThread()를 사용하여 IOCP와 작업 스레드를 각각 생성한다. <그림 8>에서 작업 스레드의 개수는 "CPU의 수*2+2"로 제한한다[Jeffrey Richter, 2000].

```
IOCP() {
  CreateIoCompletionPort(): // IOCP 생성

  SYSTEM_INFO si;
  GetSystemInfo(&si);
  m_Work ThreadNumber = si.dwNumberOfProcessors*2+2;      // 작업 스레드이 개수
  CreateThread(NULL, 0, WorkThread, this, 0, &dwThreadID);  // 작업 스레드 생성
}
```

〈그림 8〉 IOCP 및 작업 스레드 생성

2) 접수 스레드

데이터 수집기로부터의 연결 요청과 클라이언트로부터의 연결 요청을 각각 전담하는 두 개의 접수 스레드를 <그림 9>와 같이 서로 다른 포트번호를 부여하여 생성한다.

```
objWebServer = new WebServer();
objIOCP = new IOCP(objWebServer);

  objCollectorAcceptServer = new AcceptServer(objIOCP,
                        "CollectorServer", "210.119.104.161", 8989);
  objWebAcceptServer = new AcceptServer(objIOCP,
                        "WebServer", "210. 119. 104. 161", 8988);
AcceptServer::AcceptServer(IOCP*iocp, const char *serverName,
                        const char *ip, unsigned short port):m_iocp(iocp)
{
  strcpy_s(m_serverName, 256, serverName);
  strcpy_s(m_bindIP, 16, ip);
  m_port=port;
  m_threadHandle = CreateThread(
      NULL,  // 생성하는 스레드의 보안 관련 설정(default 설정)
      0,     // 생성하는 스레드이 스넥 크기 설정(default 설정)
      AcceptThreadProc,  // 스레드에 의해 호출되는 함수의 포인터를 인사로 전달
      this,  // 전달될 인사를 지정
      0,     // 새로운 스레드 생성 후 바로 실행 가능과 대기 상태 설정(바로 실행 가능)
      &m_theradID);      // 생성하는 스레드이 ID를 저장하기 위한 변수 포인터
}
```

〈그림 9〉 접수 스레드 생성

각각의 접수 스레드는 <그림 10>과 같이 socket(), bind(), listen(), accept()를 호출하여 데이터 수집기와 클라이언트로부터의 연결 요청을 기다린다. 연결 요청이 들어오면 CreateIoCompletionPort()를 호출하여 소켓과 입출력 완료 큐를 연결하고 요청 메시지를 생성한 후 PostQueuedCompletionStatus()를 호출하여 요청 메시지를 입출력 완료 큐에 등록한다.

```
DWORD ACCEPTsERVER::aCCEPTtHREADpROC() {

  ListenSock = socket();                  // 소켓을 생성
  bind();                                 // IP주소와 포트 번호
  listen();                               // 연결 상태 변경

  while(TRUE) {
  clieantSocket = accept();               // 접속 대기
  CreateIoCompletionPort();               // 소켓과 입출력 완료 큐를 연결
  clientContxt = new ClientContext();     // 요청 메시지 구조체 생성 및 초기화
  PostQueuedCompletionStatus();           // 입출력 완료 큐에 요청 메시지 등록
  }
}
```

〈그림 10〉 접수 스레드

3) 작업 스레드

작업 스레드는 GetQueuedCompletionStatus()를 호출하여 입출력 완료 큐에 등록된 하나의 메시지를 가져와 요청 메시지의 상태에 따라 각각 처리한다.

읽기모드일 경우, <그림 11>과 같이 메시지 상태를 쓰기모드로 변경한 후 WSARecv()를 호출하여 연결된 소켓 버퍼로부터 비동기 읽기를 요청한다.

```
DWORD IOCP::WorkThreadFunc() {

while(TRUE) {
  GetQueuedCompletionStatus()      // 입출력 완료 큐에서 요청 메시지를 가져옴
  if(clientContext ->GetMode() = = READ)  {
    m_mode = WRITE;                // 요청 메시지 모드를 쓰기모드로 바꿈
    RecvPost(clientContext);       // 소켓 버퍼 읽기 요청 및 에러 처리
  }
  if(clientContext -> GetMode() = = WRITE) {
    SendCompleteEvent();           // 웹 문서 및 생산 데이터 처리
  }
  else if(clientContext -> GetMode() = = END) {
    EndCompleteEvent();            // 소켓 및 요청 메시지 구조체 해제
  }
}
```

〈그림 11〉 읽기모드일 경우

쓰기모드일 경우, <그림 12>과 같이 요청 메시지의 포트번호를 검사한다. 웹 문서 요청일 경우(포트번호 : 8989), 웹 문서를 처리한 후 비동기 출력을 위한 WSASend()를 호출 하여 응답 메시지를 보낸다. 생산 데이터일 경우(포트번호 : 8988), 메시지의 내용을 데이터베이스에 저장한다.

```
BOOL IOCP::SendCompleteEvent()
{
  switch(CheckPort(clientContext)) {
    case 8988:          // 웹 문서 처리
      WebProcess(clientContext);
      SendPost(clientContext);
      break;
    case 8988:          // 생산 데이터 처리
      DataCollector(clientContext);
      break;
    default:            // 잘못된 접속
      CloseClient(clientContext);
      break;
      }
  return TRUE;
}
```

〈그림 12〉 쓰기모드일 경우

해제모드일 경우, 더 이상 사용하지 않는 소켓 및 요청 메시지 구조체를 해제한다.

4) PHP 엔진 사용

PHP 엔진을 사용하기 위하여 <그림 13>과 같이 IsapiHttpExtensionProc()를 호출하고 작업의 동기화를 목적으로 WaitForSingleObject()를 호출하여 실행완료를 기다리도록 하는 한다.

```
rc = IsapiHttpExtensionProc(&ECB);
            // PHP 엔진을 사용하기 위한 함수 호출

if (rc = = HSE_STATUS_PENDING) {
  WaitForSingleObject(context.waitEvent, INFINITE);
            // 동기화를 목적으로 실행완료 대기
}
```

〈그림 13〉 PHP 엔진 사용

데이터베이스의 생산정보를 요청하는 메시지는 <그림 14>와 같이 MySQL의 정보와 쿼리문을 포함한다. PHP 엔진은 MySQL과 연동하여 쿼리문을 처리한다[Laura Thomson외 1명, 2004].

```
mysql_connect("localhost", "서버 ID", "서버의 암호");
mysql_select_db("서버의 이름");
echo mysql_result(mysql_query(쿼리문 내용)
```

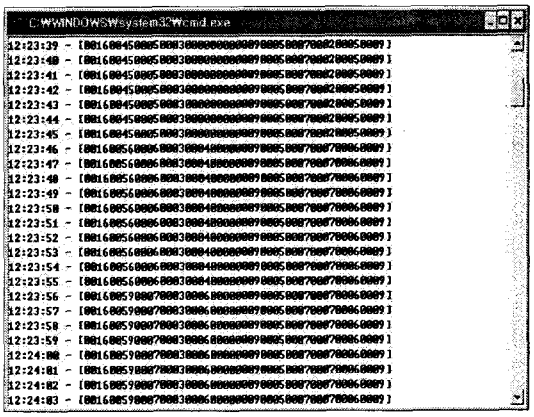
〈그림 14〉 MySQL에 작업 요청

3.2 실험

(1) 생산 데이터 처리

[단계-1] 데이터 수집기로부터 생산 데이터 받기

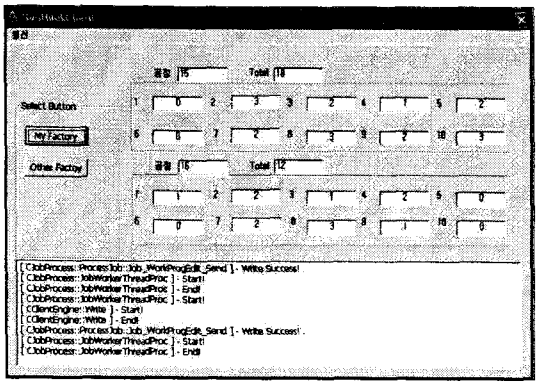
데이터 수집기에서 매초마다 생산 데이터 메시지 프레임을 전송하고, 서버에서 접수된 데이터를 <그림 15>와 같이 출력한다.



〈그림 15〉 생산 데이터 메시지 프레임 출력

[단계-2] 생산 데이터를 MySQL에 저장

수신된 생산 데이터를 공정마다 분리 하여 MySQL에 저장한다. MySQL에 저장된 생산정보를 <그림 16>과 같이 서버에서 직접 확인한다.



〈그림 16〉 서버에서 출력한 생산정보

[단계-3] 생산 데이터와 생산정보 비교

<그림 15>의 생산 데이터 메시지 프레임과 <그림 16>의 생산정보를 서로 비교 확인한다. 이 실험을 통하여 생산 데이터 입력 장치, 데이터 수집기, 서버 그리고 MySQL 간의 원활한 통신을 확인한다.

(2) 웹 서비스 처리

[단계-1] 클라이언트를 통한 서버 접속

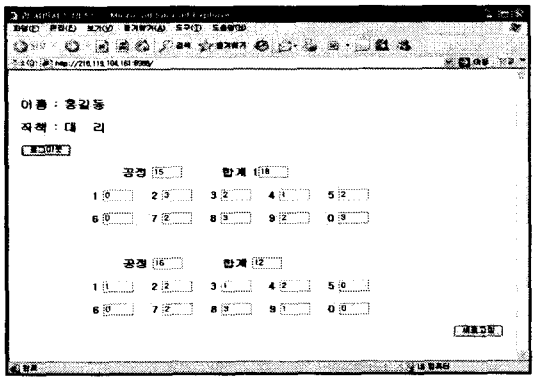
서버의 IP 주소와 포트번호를 웹 브라우저의 주소창에 입력하여 서버에 연결 요청을 시도한다. 연결 요청이 성공하면 서버는 로그인을 할 수 있는 웹 페이지를 전송하여 사용자에 아이디와 암호를 입력하도록 한다.

[단계-2] 웹 서비스를 통한 MySQL의 생산정보 전송

입력된 사용자 아이디와 암호를 수신한 서버는 이를 검사하여 인증된 사용자에게는 <그림 17>과 같은 사용자의 정보와 생산정보를 보여주는 웹 페이지를 전송한다.

[단계-3] MySQL과 클라이언트의 생산정보 비교

<그림 16>의 생산정보와 <그림 17>의 생산정보를 서로 비교해 봄으로써 데이터 수집기로부터 접수된 생산정보가 경영자 및 작업자에게 정확하게 전달되고 있음을 확인한다.

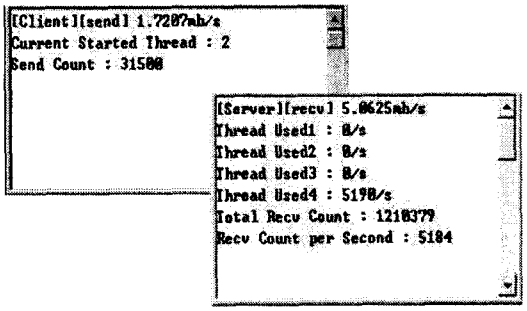


〈그림 17〉 클라이언트에서 출력한 생산정보

(3) 클라이언트의 접속 성공률 측정

제시한 서버의 성능을 검토하기 위하여 데이터 수집기를 포함하여 모든 클라이언트들의 접속 성공률을 다음과 같은 방법으로 측정한다.

4개의 스레드로 구성된 테스트 프로그램을 작성한다. 각각의 스레드는 서버에 연결을 시도한 후, 성공할 경우 1KB 크기의 메시지를 전송한다. 서버는 수신된 메시지를 연결된 클라이언트로 되돌려 주는 것을 1회의 접속 성공으로 간주한다. 이와 같은 방법으로 클라이언트의 연결 요청 횟수와 서버의 수신 횟수를 <그림 18>와 같이 측정한다.

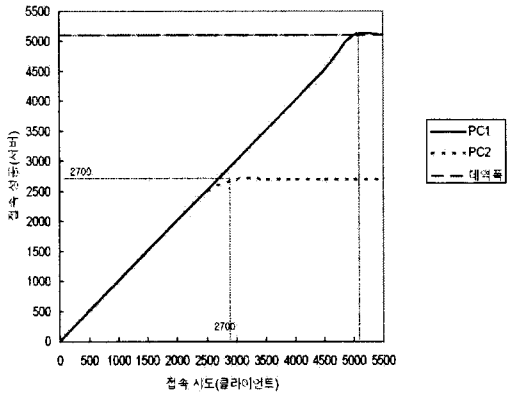


〈그림 18〉 클라이언트 송신과 서버 수신

한편, 1초 동안에 서버에서 최대 처리할 수 있는 클라이언트의 수(n)는 다음과 같이 계산할 수 있다.

$$n = \frac{bandwidth[dyte/\sec]}{massagesize[byte]}$$

실험에 사용된 대역폭은 5MB/s이고 메시지의 크기는 1KB이므로 1초당 5,120번의 접속 성공이 예상된다. 따라서 초당 5,120번 이상의 접속을 시도가 필요하다. 이를 위하여 3대의 PC에서 테스트 프로그램을 실행하여 각각 1초당 2,400, 1,800, 1,000씩 총 5,200회의 접속을 시도하도록 한다.



〈그림 19〉 접속시도/접속성공

<그림 19>에서 PC-1의 경우 약 5,100회 정도의 성공으로써 대역폭에 제한을 받고, PC-2는 2,700회 정도의 성공으로써 대역폭을 넘지 않는 선임에도 불구하고 PC-2의 사양에 제한을 받고 있음을 알 수 있다. 따라서 제시한 서버 모델을 일반적인 사양인 PC-2에 적용할 경우에도 1초당 2,700회 정도의 클라이언트들의 접속 성공이 가능함으로 중소기업의 규모에 적용할 수 있음을 확인한다.

4. 결 론

생산정보 관리시스템의 서버는 기본적으로 생산 현장에서 발생하는 데이터의 정확한 수집과 경영자 및 작업자들의 정보 요청에 대한 신속한 응답이 요구된다. 본 논문에서는 이러한 요구사항을 고려하여 IOCP 큐를 중심으로 다수의 접수 스레드와 작업 스레드로 구성된 생산정보 관리시스템에 적합한 서버를 제시하였다.

제시한 서버의 특징은 연결 요청만을 전담하는 접수 스레드를 대기시킴으로써 데이터 수집기와 클라이언트를 통한 경영자 및 작업자의 연결 요청을 병행적으로 처리하고, IOCP 큐를 기반으로 한 작업 스레드는 운영체제의 비동기 소켓 입출력 기능을 사용함으로써 데이터 수집기와 클라이언트의 서비스 요청에 대한 처리율을 향상시키는 것이다. 또한 스레드 풀을 사용함으로써 생산 규모에 따라 비례적으로 증감하게 되는 데이터 수집기 수와 무관하게 스레드 개수를 유지할 수 있다.

반면 제시한 서버는 운영체제의 비동기 통지 기능을 이용하고 있기 때문에 운영체제에 의존하게 되는 단점과 성능향상을 위하여 비동기 입출력과 입출력 완료 포트 그리고 스레드 풀을 사용하므로 다른 모델에 비하여 코딩이 복잡하다는 단점이 있다. 그러나 단점의 하나인 운영체제의 의존 부분은 대부분의 중소 제조업체에서 윈도우즈 NT계열 운영체제를 사용하고 있기 때문에 제시한 서버 모델의 적용이 쉽다.

참 고 문 헌

[1] 김선우, *윈도우 네트워크 프로그래밍*, 제 3판, 한빛미디어, 2004.

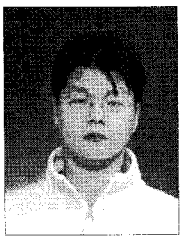
[2] 배창우, 최영호, 조경민, 유경훈, 윤상배, *윈도우&리눅스 TCP/IP 소켓 프로그래밍*,

초판, 한빛미디어, 2005.
[3] 중소기업정보화경영원, *생산성 향상은 생산공정 디지털화로*, 중소기업청, 2005.
[4] 차석근, *생산정보화 (e-Manufacturing) 솔루션 기능과 구현 사례*, (주) 에이시 에스, 기술연구소, 2004.
[5] Anthony Jones, and Jim Ohlund, *NETWORK PROGRAMMING FOR MICROSOFT WINDOWS*, 정보문화사, 2003.
[6] Gyu-bake Kim, "An Effective Processing Server for Various Database Operations of Large-scale On-line Games", IASTED International Conference on Information and Knowledge Sharing(IKS) 2003, Arizona, USA, November 2003.
[7] J. Richter, S. Stoller, and S. Ur., "Benchmark and Framework for Encouraging Research on Multi-Threaded Testing Tools", PADTAD'03, Nice, France, April 2003, pp. 22-26.
[8] Kalen Delaney, *Inside Microsoft SQL Server 2000*, Microsoft Press, 2000.
[9] Laura Thomson, and Luke Welling, *PHP and MySQL Web Development*, SAMS, 2004.
[10] Mike Gunderloy, and Joseph L. Jordan, *MASTERING SQL SERVER 2000*, Sybex, 2000.
[11] Jeffrey Richter, and Jason D. Clark, *Programming ServerSide Applications for microsoft windows 2000*, Microsoft Press, 2000.
[12] Sang In Han, "Development of Internet Based Monitoring and Control System for Manufacturing Facilities", *Journal of the Korean Institute of Plant Engineering*, Vol. 6, No. 3, SEP 2001, pp. 73-83.
[13] Smit. K., "interactive computer systems for maintenance management" Maintenance Management International, Vol. 7, 1983, pp. 7-15.

저자소개

임 성 락

서강대학교 전자공학(학사), 서울대학교 컴퓨터공학(석사) 서울대학교 컴퓨터공학(박사), 금성반도체 선임연구원, 현재 호서대학교 컴퓨터공학과 교수로 재직중이며 주요 관심분야는 운영체제, 임베디드 시스템



송 기 석

호서대학교에서 컴퓨터공학 학사를 취득했고, 2006년부터 호서대학교 석사과정, 주요 관심분야는 임베디드 시스템이다.

◈ 이 논문은 2007년 07월 30일 접수하여 1차 수정을 거쳐 2007년 10월 08일 게재확정되었습니다.