

TCP/IP 소켓을 이용한 원격 측정 장치의 실험 데이터 통합 관리 시스템 개발

김선영^o 조환규

부산대학교 컴퓨터공학과

s.y.kim@pusan.ac.kr hgcho@pusan.ac.kr

Management System for Experimental Data

In Remote Measurement Device Using TCP/IP Socket

SeonYeong Kim^o Hwan-Gue Cho

Dept. of Computer Science and Engineering, Pusan National University

요 약

최근의 과학 실험은 그 규모나 내용에 있어서 점차 대형화되는 동시에 복잡해지고 있다. 이로 인하여 다양한 측정 장비로부터 도출된 실험 결과를 효율적으로 분석, 관리, 종합하는 도구의 필요성이 커지고 있다. 본 논문에서는 원격 측정 장치로부터 서로 다른 포맷의 실험 데이터를 자동 수집한 후 이중 정제된 데이터들만 추출하여 웹에서 시각화하는 실험 데이터 통합 관리 시스템을 제안한다. 먼저 원격 측정 장치의 데이터를 자동으로 수집하기 위해 폴링 서버를 설계하여 장치마다 폴링 에이전트를 도입하였다. 이를 통해 관리자가 각 측정 장치에 직접 접근하지 않고도 데이터를 수집할 수 있다. 폴링으로 확보한 데이터는 파싱을 통해 정제하고, 이들 데이터로 데이터베이스를 구축한다. 정제된 데이터는 시각화하여 사용자가 웹에서 쉽게 파악할 수 있다. 데이터 폴링은 TCP/IP Socket을 통해 수행하므로 보편적으로 사용하는 FTP 방식에 비해 데이터 확보 시 신뢰성을 높일 수 있으며, 폴링 여부 판단 시에는 동기식, 실제 폴링 시에는 비동기식 통신 방법을 사용하여 폴링의 효율을 높였다. 본 시스템을 활용하여 사용자의 임의적인 데이터 접근을 최소화하였고 데이터의 전송, 저장, 관리를 자동화함으로써 편의성을 높였다. 본 시스템을 활용하여 원격 실험 장치로부터 데이터를 확보할 때의 정확성과 폴링 및 파싱 속도를 실험을 통해 측정하였고, 그 결과 폴링 시 100%의 정확도와 정상 포맷의 데이터에 대해서 100%의 파싱 결과를 보임으로써 본 시스템이 원격 장치의 실험 데이터를 통합 관리할 때 적합함을 알 수 있었다. 추후 데이터의 속성에 따라 클러스터링 할 예정이며 클러스터링에 따른 시각화 서비스를 제공할 계획이다.

1. 서 론

최근의 과학 실험의 추세는 그 규모와 내용이 점차적으로 커지고 복잡해지고 있다. 그에 따라 실험 데이터들의 크기도 방대해지고 종류도 다양해지므로, 이러한 다양한 측정 장비에서 생성된 실험 결과를 효율적으로 분석, 관리, 종합하는 도구에 대한 필요성이 증가하였다. 최근의 건물 모델링 실험의 경우 다양한 형태의 건물이 등장함으로써 인해 그 데이터가 더욱 복잡한 형태를 띠게 된 것을 예로 들 수 있다.[1]

본 보고서에서는 측정 장치로부터 서로 다른 포맷을 가진 실험 결과를 자동 수집하고 이를 분석하여 웹에서 시각화하여 보여주는 실험 데이터 통합 관리 시스템을 설계하였다. 실험 데이터는 일반적으로 해당 측정 장치에만 누적되므로 사용자가 실험결과를 얻으려면 각 장치에 직접 접근해야 하는 번거로움이 있다. TCP/IP소켓을 이용해 데이터 수집을 위한 폴링 에이전트를 각 측정 장치에 도입함으로써 사용자의 편의성을 높임과 동시에 사용자의 임의적인 접근을 배제해서 데이터의 무결성을 높일 수 있다. 또한 파일 송수신 시 보편적으로 사용하는 FTP 방법을 사용하는 것 보다 데이터를 확보할 때 신뢰성을 높일 수 있으므로, 폴링에서는 TCP/IP 소켓을 이용

한 방식이 더 적합하다고 판단할 수 있다. 본 시스템에서는 데이터 폴링을 하기 전 폴링 여부를 판단하는 과정에서 동기식 통신 방법을 사용하고 실제 폴링이 일어날 때는 비동기식 통신 방법을 사용하였다. 또한 스레드(thread)로 처리하여 다수의 클라이언트 요청에 대해 병렬·고속 처리가 가능하도록 하였다[2]. 본 시스템에서는 데이터를 확보해서 데이터 웨어하우스(warehouse)를 구축한 후, 정제된 데이터를 추출하고 이를 시각화하여 사용자가 웹에서 데이터를 쉽게 파악할 수 있는 환경을 제공하였다.

2. 관련 연구

일반적으로 원격의 측정 장치에 있는 데이터를 서버로 전송하기 위해서는 사용자가 직접 클라이언트에 접근해서 전송할 데이터를 지정해주어야 한다. 이 과정에서 사용자에게 의한 데이터 손실, 오염, 누락이 발생할 수 있다. 파일 전송 공정을 자동화할 경우 이러한 피해를 최소화할 수 있는데, 폴링을 도입함으로써 사용자의 임의적인 접근을 배제할 수 있다. 파일 전송 시스템(FTP) Stream 전송 모드를 이용하여 폴링을 시도할 경우 자동화하기가 까다롭고 치명적인 오류가 발생할 경우 다시 전송해야

한다[3]. 반면 TCP/IP 소켓 통신은 폴링을 자동화하기 쉽고 데이터 손실을 막을 수 있으므로 FTP 방법에 비해 신뢰성이 높다. 소켓 사용 시에는 동기화 소켓을 사용하면서 스레드를 사용하는 방법이 구현도 쉽고 안정성이 높으나 자원 점유도가 높다. 비동기화 소켓을 사용할 경우 자료 재조합과정과 프로토콜 파싱 과정이 필요하고, 대용량 파일에서는 실행 흐름 제어를 해 주어야 하므로 동기화방식에 비해 속도도 빠르지 않고 구현이 매우 까다롭다. 그러나 비동기 방식으로 데이터를 전송하는 경우 데이터의 전송 속도와 무관하게 다른 작업을 할 수 있어 효율성이 높아지므로, 성능의 측면에 있어서는 비동기 방식이 동기 방식보다 우월하다[4]. 본 논문에서는 데이터의 안정적인 폴링을 가장 우선시하므로 TCP/IP 통신 방법을 사용하며, 폴링 여부를 판단하는 과정에서 동기화 폴링 기법을 사용하고, 실제 폴링 시에는 비동기화 폴링 기법으로 측정 장치로부터 데이터를 수집하는 방식을 사용하여 효율을 높이고자 한다.

3. 시스템의 설계 및 구현

3.1 데이터 폴링 모듈

데이터를 폴링하기 위해서 폴링 서버는 다중 스레드 (Multi Thread)를 사용하여 클라이언트와의 연결을 생성한다. 폴링 모듈의 개념도는 그림 1 과 같다.

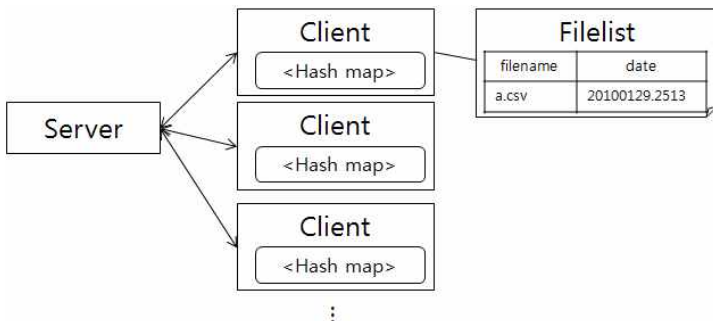


그림 1. 폴링 서버와 클라이언트 간 관계 개념도. 각 클라이언트는 모니터링 하고 있는 폴더에 누적되는 파일목록을 가지고 있다. 파일 목록에는 파일명과 갱신 시각이 표기되는데, 이 파일목록과 이전의 파일목록을 저장한 Hash map과의 비교를 통해 폴링 되지 않은 파일을 알아낼 수 있고, 폴링 되지 않은 파일들만 서버로 전송한다.

폴링 서버는 단일 스레드로 구성하고 여러 대의 클라이언트는 멀티 스레드로 구성한다. 서버가 폴링을 하기 전에 폴링 대상 파일에 대해 두 번의 검사 과정을 거친다. 우선 클라이언트는 각 측정 장치의 특정 폴더에 실험 데이터를 누적하고, 해시 맵(Hash map)에 파일의 이름과 갱신 시각을 저장한다. 그 후 주기적으로 폴더 내의 파일 목록과 해시 맵을 비교하여 갱신된 파일이 존재할 경우 서버와 연결을 함으로써 폴링 할 파일을 한번 검토한다. 연결 후에 서버가 클라이언트로부터 폴링 대상 파일 목록을 수신하면 이와 실제로 서버에 존재하는

파일과의 비교 검토를 통해 정확히 받아야 할 파일만을 수신함으로써 폴링 해야 할 파일에 대해 총 두 번의 확인을 거쳐서 폴링을 수행한다.

서버와 클라이언트간의 연결은 동기식으로 이루어진다. 클라이언트가 서버에게 연결 요청을 보내면서 파일의 정보를 넘겨주면, 서버는 파일 정보와 실제 파일의 비교를 통해 클라이언트에게 폴링 여부를 알려주는 ack를 보내면서 연결이 이루어진다. 파일 전송 과정은 비동기식으로 클라이언트와 서버 간의 폴링을 수행하고 나면 클라이언트는 그에 대한 서버의 ack를 기다리지 않는다. 이것은 측정 장치에 파일이 누적되어 있으므로 매 폴링 시도마다 전송하지 않은 파일을 재검사를 할 수 있기 때문이며, 웨어하우스 구축에 시간이 많은 영향을 끼치지 않기 때문에 효율적이다.

그림 2는 폴링 서버의 구동 모듈이다. 소켓 생성 후 연결이 되면 파일 이름과 각 파일의 갱신 시각을 읽는다. 파일 길이를 측정해서 서버에 존재하는 파일이면 새로 송신하고 동일한 파일이 존재할 경우 파일 수신을 거부한다. 파일이 존재하지 않을 경우 파일 갱신 시각을 동기화하고 폴링한 후 이어서 포맷에 맞는 파싱을 시작한다.

```

while(true) {
    Socket Csok = mServerSocket.accept();
    DataOutputStream out = new
    DataOutputStream(Csok.getOutputStream());
    DataInputStream in = new
    DataInputStream(Csok.getInputStream());

    String FName = in.readUTF();
    long FDate = in.readLong();
    FilenameFilter mCompareFileName = FName;
    File[] Founded =
    DestDir.listFiles(FnameFilter);
    File newF =new
    File(DestDir.getPath()+"\\"+Fname);
    newF.createNewFile();
    out.writeBoolean(true);
    long newFLen = in.readLong();
    if (Founded.length == 0)
        BufferedReadingNParsing();
    else if (Founded.length > 0)
        if(Founded[0].lastMod()<FDate){
            out.writeBoolean(true);
            long FLen = in.readLong();
            BufferedReadingNParsing();
        }
    else
        out.writeBoolean(false);
}
    
```

그림 2. Polling Server의 구동 방법이다. 소켓 생성 후 연결이 되면 파일이름과 갱신 시각을 읽고, 파일 길이를 측정해서 폴링을 시작한다. 폴링이 완료되면 즉시 포맷에 따른 파싱을 시작한다.

그림 3은 폴링 에이전트의 구동 모듈이다. 클라이언트

가 측정 장치 내의 폴더를 주기적으로 모니터링하면서 갱신된 파일의 정보를 감지하면 서버에게 해당 파일의 내용을 전송한다. 서버로부터 전송 요청을 받으면 비로소 폴링을 수행한다.

```
while (true) {
    File[] Files=SDir.listFiles(new FilenameFilter() {
        public boolean accept(File dir, String name) {
            return name.endsWith(".csv");
        }
    });
    for (File F: Files) {
        Date ModDate = HMap.get(F.getName());
        if (ModDate != null) {
            if (F.lastMod() > ModDate.getTime()) {
                HMap.put(F.getName(), new Date(F.lastMod()));
                SendFile(F);
            }
        } else {
            HMap.put(F.getName(), new Date(F.lastMod()));
            SendFile(F);
        }
    }
    Thread.sleep(1000);
}
```

그림 3 . Polling Agent의 구동 방법이다. 측정 장치의 .csv 파일을 주기적으로 모니터링하고, 파일의 갱신 내용을 감지하면 서버에게 해당 파일들의 정보를 전송한다. 서버로부터 전송 요청을 받으면 폴링을 시작한다.

3.3 데이터 파싱 및 정제 모듈

데이터를 수집한 후 정제한 데이터만으로 웨어하우스를 구축하기 위해서는 데이터 파싱이 이루어져야한다.

파싱은 서로 다른 세 종류의 데이터에 대해 수행한다. 한 종류의 데이터에는 xls, csv, txt 등 데이터 저장 시 파일 포맷 세 가지가 도출된다. 이들 포맷에 대해 파싱을 수행하기 위해서 jxl, opencsv 2.1의 라이브러리를 사용하여 적절히 데이터를 파싱할 수 있다. 그림 3은 파싱해야할 실험 데이터의 한 종류이다. 실험 데이터의 속성과 값이 가로, 세로 형식으로 일정하지 않고 그래프를 포함하고 있다. 그래프는 실험 결과로 도출된 값을 시각화한 것이다.

xls, csv, txt 등의 파일 내용을 제대로 추출하면 이를 데이터 객체에 저장하여 쿼리(Query)를 생성할 수 있도록 한다. 포맷이 다른 데이터와 전체적, 부분적으로 유사되었거나 변형이 가해진 데이터 또는 의도하지 않은 위치에 발생하는 가비지(garbage) 데이터 등에 대한 예외 처리를 수행하여, 이러한 예외가 발생할 경우 시스템이 바로 종료되지 않고 다음 이벤트로 넘어가되 로그를 남겨서 오류내역을 살펴 볼 수 있도록 하였다. 위와 같은 과정을 거치면 정제한 데이터만으로 웨어하우스를 구축할 수 있다.

3.3 데이터 시각화 모듈의 설계

정제한 데이터를 사용자가 한 눈에 파악하기 위해서 웹에서 데이터를 시각화하여 보여준다. 그림 5는 그림 4의 데이터를 Flex로 시각화하여 웹에서 나타낸 것이다.

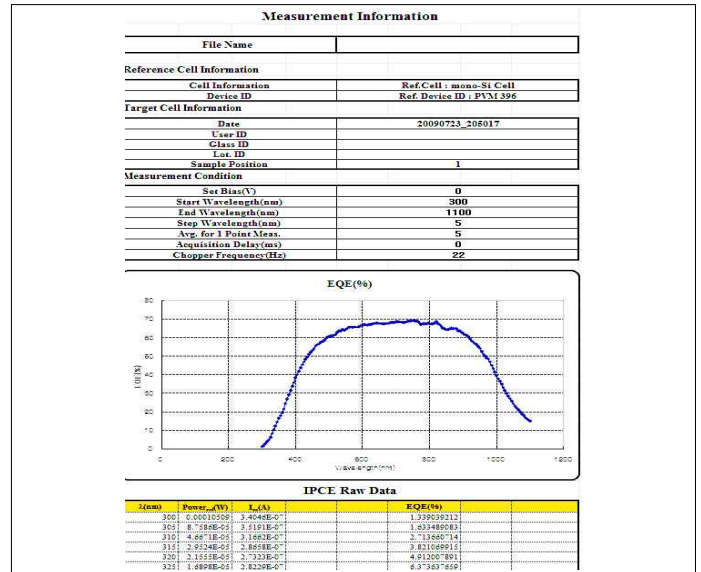


그림 4. Parsing해야 할 데이터 K3100의 파일 포맷. 데이터의 속성과 값이 일정한 형태가 아니며 그래프도 포함하고 있다. 그래프는 실험 결과 데이터를 시각화하여 나타낸 것이다.

그림 5에서 보듯이 기존의 그래프에서 파악하기 어려웠던 그래프 위의 점도 마우스를 가져가면 수치를 쉽게 확인할 수 있다.

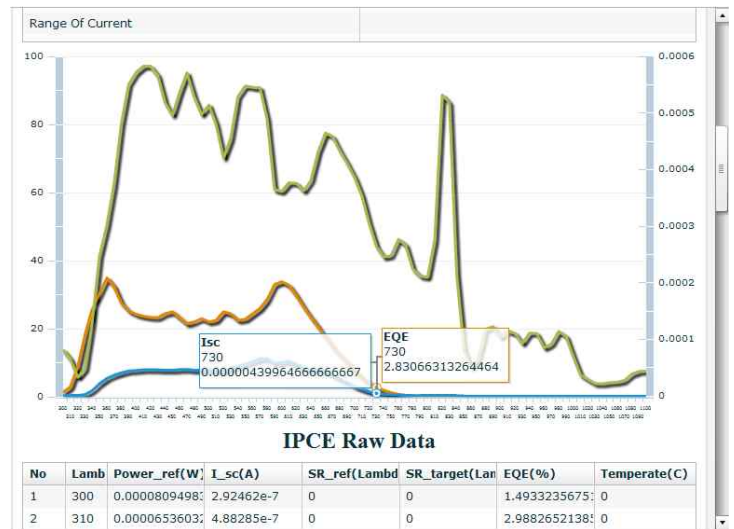


그림 5. 그림 4의 데이터를 웹 상에서 시각화 한 모습. 기존의 그래프에서는 그래프 위 점의 수치를 파악하기가 매우 어려웠으나 시각화 모듈을 사용하여 사용자가 마우스오버 만으로 해당 점의 정확한 수치를 파악하기가 쉬워졌다.

4. 실험

본 시스템의 성능을 측정하기 위해 각 측정 장치에서 도출되는 서로 다른 서식을 가진 3가지 종류의 데이터 K3000, K3100, K3600에 대해 실험하였다. 종류와 무관하게 실험에 사용한 csv 파일의 크기는 평균 3KB이고, txt 파일의 크기는 평균 3KB, xls 파일의 크기는 평균

59KB이다. 정상포맷의 데이터 수/전체 데이터 수를 표 1에 나타내었다.

표 1. 본 시스템의 성능 측정을 위한 실험에 사용된 실험 데이터의 분포. 각 데이터마다 확장자가 txt, csv, xls 인 세 가지 포맷이 존재한다. 표의 숫자는 데이터의 개수를 나타낸다. 표의 내용은 정상포맷 데이터/전체 데이터를 나타낸다. 장치 A, B, C는 모두 서로 다른 실험 측정 장비로 다른 측정 장치라면 포맷이 같아도 데이터 서식은 다르다. txt, csv 파일의 평균은 3KB, xls 파일의 평균은 59KB이다.

종류 \ 포맷	txt	csv	xls
장치A(K3000)	96/100	135/148	95/100
장치B(K3100)	138/144	126/134	10/11
장치C(K3600)	-	91/95	-

실험은 서버와 클라이언트가 서로 내부 네트워크와 외부 네트워크에 위치할 때의 폴링 속도, 포맷에 따른 파싱 속도를 측정하였다.

표 2. 세 가지 종류의 장치로부터 각각 txt, csv, xls 포맷의 파일을 획득하여 폴링과 파싱을 실험한 결과. 폴링 결과는 '폴링에 성공한 데이터의 개수/전체 파일 개수' 이고, 파싱 또한 그렇다. 폴링 시간은 전체 데이터를 폴링 하는데 소요한 시간이며 파싱 시간은 정상 포맷의 데이터를 파싱하고 cracked data를 에러 처리 하는데 소요한 시간의 합이다. 실험 결과 폴링은 데이터나 파일 포맷의 종류와 무관하게 폴링에 모두 성공했고, 파싱은 cracked data를 제외한 정상 포맷의 데이터를 모두 파싱하는데 성공했음을 알 수 있다.

장치 종류	데이터 포맷	폴링		파싱	
		결과	시간 [sec]	결과	시간 [sec]
A	txt	100/100	25.00	96/100	10.01
	csv	148/148	33.00	135/148	14.82
	xls	100/100	41.00	95/100	18.66
B	txt	144/144	37.00	138/144	14.7
	csv	134/134	30.00	126/134	13.8
	xls	11/11	4.53	10/11	1.15
C	csv	95/95	25.65	91/95	9.70

실험 결과 모든 데이터에 대해서 폴링은 100%의 성공률을 보였고, 파싱에서는 cracked data를 제외한 정상 포맷을 따른 파일은 100%파싱에 성공하였다. cracked data의 경우에는 모두 데이터 포맷이 지켜지지 않은 경우로, 시스템이 정상적으로 예외 처리를 수행함을 알 수 있었다. 원격 실험 장치에서 얻는 데이터를 관리하는 것은 시간에는 민감하지 않으나 정확도가 매우 높아야 하므로 본 시스템은 원격 데이터 관리에 적합한 성능을 보임을 알 수 있다.

5. 결론 및 추후 과제

본 논문에서는 TCP/IP Socket을 이용해서 원격 실험 장치에 있는 데이터를 통합 관리하는 시스템을 개발하였다. 데이터 폴링에서 파싱까지의 전 공정을 자동화하여 사용자의 임의적인 접근을 최소화하였고, 편의성 또한 높였다. 폴링으로 구축한 데이터를 정제하여 데이터베이스를 구축하면, 사용자는 이 데이터들이 웹 상에서 시각화된 모습을 확인할 수 있다. 본 시스템을 활용하여 원격 실험 장치로부터 데이터를 확보할 때의 정확성은 100%로 데이터 손실 없이 파일을 전송함을 확인할 수 있었다. 또한 정상 포맷의 데이터에 대해서도 100%의 파싱 결과를 보임으로써 원격 측정 장치에서 도출되는 데이터를 신뢰도 높게 확보할 수 있었다. 이러한 데이터들을 관리함에 있어 폴링이나 파싱 시간보다는 그 정확도가 매우 중요한 척도이므로 본 시스템은 원격 측정 장치의 데이터로 데이터 웨어하우스를 무결하게 구축하기에 매우 적합함을 알 수 있었다. 또한 사용자는 데이터에 일절 직접 접근하지 않으면서도 웹을 통해 실험한 데이터의 결과를 파악할 수 있고, 실제 데이터만으로는 부족했던 그래프 부분까지 각 점마다 정확한 수치를 표현함으로써 데이터의 시각화도 제대로 수행되었음을 확인할 수 있었다. 추후 데이터의 속성 별 클러스터링과 클러스터링에 따른 시각화 서비스를 제공할 계획이다.

참고 문헌

- [1] 김정현, 다양하고 복잡한 건물의 자동 모델링을 위한 LiDAR 데이터 처리 기법, 2008.
- [2] Elliotte Rusty Harold, Java Network Programming, 3rd edition, 105-107쪽, 2004.
- [3] 박성인, 최돈은, 김동한, 서영상, 서정수, 이재영, FTP에서 Stream 전송 모드의 개선, 한국정보과학회, 제 25권, 제 2호, 600~602쪽, 1998
- [4] 윤성우, TCP/IP 소켓 프로그래밍, 475쪽, 2003.