ALTIBASE 에서의 윈도 이벤트 로그 기법 구현

전호원*, 최재남**, 이상원*
*성균관대학교 정보통신공학부
**㈜알티베이스 연구개발본부
e-mail : rozehime@naver.com, unclee@altibase.com, spun@skku.edu, swlee@skku.edu

Windows Event Logging in ALTIBASE

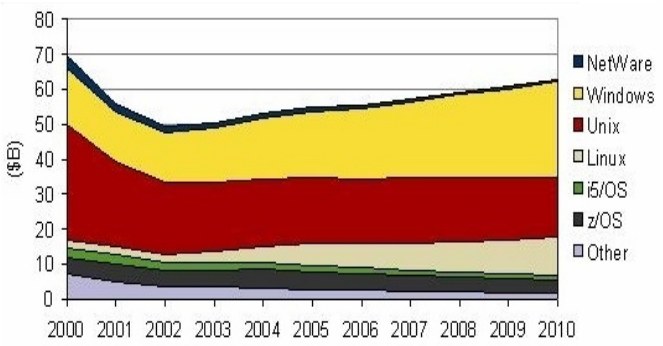
Ho-Won Jeon*, Jae-Nam Choi**, Sang-Won Lee*
*School of Information and Communication Engineering, Sungkyunkwan University
**R&D Division, ALTIBASE Corporation

요 약

㈜알티베이스에서 개발한 메모리상주형 관계형 DBMS 인 ALTIBASE 는 로그 정보를 자체 로그 파일에 기록한다. 하지만 로그 발생 모듈과 중요도로 각각 분류되어 별개의 로그 파일에 기록되기 때문에 사용자가 참조하는데 불편함이 따른다. 본 논문에서는 윈도 운영체제의 이벤트 로그 기능을 데이터베이스 시스템의 로그 기록 방법으로 구현하여 사용자 편의성을 제공하는 방법을 제시한다.

1. 서론

전세계 서버 운영체제의 성장률을 보여주는 그림 1(IDC 자료)에서 확인할 수 있듯이 윈도 서버의 성장률은 인상적이다[1]. 국내 시장의 경우는 유닉스와 리눅스가 전체 서버 운영체제 시장의 약 70%를 차지하고 있으며 윈도는 약 30%를 차지하고 있다. 하지만 국내 시장에서도 유닉스의 점유율은 점점 떨어지고 있으며, 윈도와 리눅스의 점유율이 늘어나고 있는 추세이다 [2].



(그림 1) 전세계 서버 운영체제 성장률

이러한 상황에서 다양한 DBMS 개발 업체들이 자사 제품의 윈도 버전을 통하여 시장 공략에 나서고 있다. 국산 관계형 DBMS 개발 업체인 ㈜알티베이스의 역시 윈도 버전을 지원하고 있지만 전체 매출 규모의 극히 일부만을 차지할 뿐이다. 하지만 앞으로 윈도 서버 시장의 확대와 국외 사장을 감안할 때 윈도 버전의 기능 개선 및 추가가 요구되고 있다.

이와 같은 ALTIBASE 윈도 버전의 기능 개선 및 추가 요구사항 중 하나가 윈도 이벤트 로그의 지원이다. 기존의 ALTIBASE 는 모든 트레이스 로그 정보를 그 로그의 발생 모듈과 중요도에 따라 자체 로그 파일에 각각 별개의 파일로 분류해서 기록하였다. 따라서 사용자가 필요로 하는 특정 로그를 참조하기 위해서는 모든 모듈의 로그 파일을 검색해야 하는 불편함이 있었다.

본 논문에서는 윈도 운영체제의 이벤트 로그 기능을 데이터베이스 시스템의 로그 기록 방법으로 구현하여 사용자 편의성을 제공하는 방법을 제시한다.

2. 관련 연구

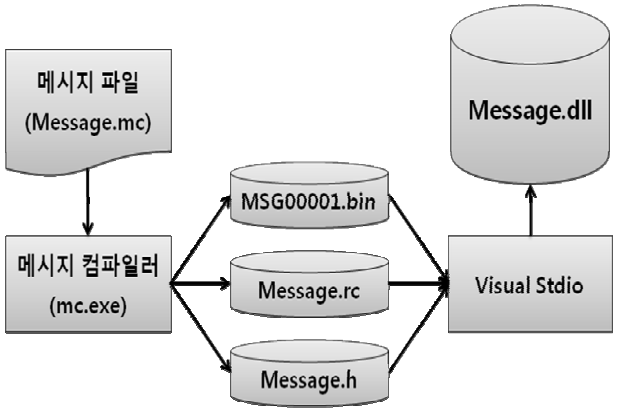
2.1 윈도 이벤트 로그 서비스

윈도 이벤트 로그 서비스는 윈도가 제공하는 표준 이벤트 보고 메커니즘(Standard Event Reporting Mechanism)으로 이는 하나의 공통된 표준 로그 포맷을 제공하여 사용자가 시스템이나 여러 어플리케이션에서 발생한 이벤트를 하나의 이벤트 뷰어로 확인 할 수 있게 한다[3].

이러한 이벤트 로그는 표준 이벤트 뷰어에 의해 크게 3 가지(시스템, 보안, 응용 프로그램)로 나눠지며 각각의 이벤트 로그는 그 로그의 정보가 기록되는 10 개의 필드(종류, 발생시간, 기록시간, 원본, 범주, 이벤트 ID, 사용자, 시스템, 설명, 바이너리 데이터)를 가지고 있다. 이들 중 종류, 원본, 범주, 이벤트 ID 는 소

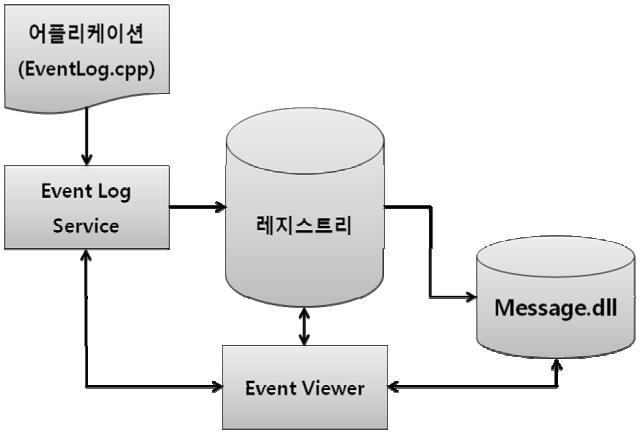
† 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0046))

스 어플리케이션이나 서비스에서 정의해 주어야 하며, 설명의 경우는 이벤트 ID 에 의해 message dynamic link library(message DLL) 파일에서 참조된다[3]. 따라서 이벤트 로그 서비스를 사용하기 위해서는 그림 2 와 같은 방법으로 message DLL 파일의 생성이 필요하며, 이 때 사용되는 메시지 파일에는 이벤트 ID 와 그에 대응되는 설명이 정의 되어있어야 한다.



(그림 2) Message DLL 파일의 생성 구조

이렇게 생성된 message DLL 파일을 이벤트 로그 서비스와 연결시키기 위해서는 레지스트리에 원본(소스 어플리케이션 혹은 서비스)과 같은 이름의 서브 키를 추가하고 레지스트리 값에 해당 message DLL 파일의 경로를 추가하면 된다. 이러한 과정은 그림 3 과 같이 간략하게 나타낼 수 있다.



(그림 3) Message DLL 파일의 참조 구조

2.2 ALTIBASE 트레이스 로그 시스템

ALTIBASE 의 트레이스 로그 시스템은 발생하는 모든 트레이스 로그를 발생시킨 모듈에 따라 6 개로 나누고 다시 이를 중요도에 따라 33 개의 레벨로 분류하고 있다. 이러한 트레이스 로그들은 모듈의 종류에 따라 각각 다른 파일에 기록되게 되며, 사용자는 레벨에 따라서 로그의 기록을 제한할 수 있다.

파일에 출력되는 로그는 헤더와 텍스트로 이루어져 있는다. 여기서 헤더에는 로그가 발생된 시간, 로그의 레벨, 스레드 번호가 출력된다.

이러한 로그의 기록은 다음과 같은 구조를 가진 함수의 호출에 의해 이루어진다.

```
Logging_Function()
{
      Log_File_Open_Function()
      Log_Header_Printing_Function()
      Log_ Description_Printing_Function()
      Log_File_Close_Function()
}
```

2.3 윈도 이벤트 로그와 ALTIBASE 로그의 비교

먼저 ALTIBASE 에서의 모듈에 의한 로그 분류 방법은 윈도 이벤트 로그의 범주 필드에 해당된다고 볼 수 있다. 중요도에 따른 33 레벨의 경우는 윈도 이벤트 로그에는 없는 개념이다. 하지만 사용자가 레벨에 따라 로그의 기록을 제한 할 수 있으므로 이벤트 로그에서도 적용 가능하다.

로그의 기록 방법에 있어서는 ALTIBASE 로그의 경우 어떠한 작업을 수행하는 도중 각각의 과정을 실시간으로 로그 파일에 기록하는 방법을 사용한다. 이러한 방법은 하나의 작업에 대한 결과 로그를 각각의 과정으로 분할하여 하나의 헤더 뒤에 각 과정에 해당하는 텍스트만 파일에 출력하기 때문에 사용자는 이들을 하나의 로그로 인식한다. 하지만 이러한 방식을 하나의 이벤트가 발생하는 즉시 하나의 독립적인 로그를 발생시키는 윈도 이벤트 로그 서비스를 그대로 적용시킬 경우 하나로 등록되어야 할 이벤트 로그가 여러 개로 나눠져 등록되는 등의 불필요한 이벤트 로그의 등록이 예상되므로 3.4 절에서 이를 방지할 수 있는 방법을 제안한다.

또한 ALTIBASE 는 로그를 기록하는 함수를 호출할 때 로그 텍스트로 출력될 내용을 문자열과 가변인자로 전달하기 때문에 로그 텍스트를 유연하게 표현할 수 있다. 반면 윈도 이벤트 로그는 로그 텍스트가 message DLL 파일에 미리 정의되어 있고 각 로그 텍스트는 고유의 이벤트 ID 로 참조되어 출력된다. 따라서 3.3 절에서 ALTIBASE 의 로그 시스템에 대한 수정을 최소화하여 윈도 이벤트 로그 서비스를 적용할 수 있는 방법을 모색한다.

3. 윈도 이벤트 로그 서비스 적용 방법 설계

3.1 이벤트 소스의 레지스트리 등록

이벤트 로그를 적용하기 위해서는 이벤트 소스 서브키(ALTIBASE)를 다음의 레지스트리에 등록하여야 한다 [3].

```
HKEY_LOCAL_MACHINE
    SYSTEM
        CurrentControlSet
            Services
                EventLog
                    Application
                        Event Source
```

레지스트리에 이벤트 소스 서브 키를 등록한 후에는 해당 레지스트리 값으로 차후 생성할 message DLL 파일의 경로를 지정한다.

이러한 과정은 최초 ALTIBASE 의 설치과정에서 이루어져야 할 것이다.

3.2 Message DLL 파일의 생성

본 제안에서 message DLL 파일은 이벤트 ID 와 로그 텍스트의 연결과 범주에 대한 정의를 모두 포함한다.

앞서 언급했듯이 범주는 ALTIBASE 의 모듈 이름으로 정의될 수 있으므로 메시지 파일에서 모듈의 종류에 따라 SERVER, SM, RP, QP, DL, LK 의 6 가지로 정의한다.

이벤트 ID 와 로그 텍스트의 연결에서는 ALTIBASE 의 트레이스 로그 시스템에 대한 수정을 최소화하여 적용시키기 위해 이벤트 ID 의 유일성을 제한하였다. 즉, 각각의 로그 텍스트들을 고유하게 식별하던 이벤트 ID 를 모듈명과 레벨로 분류된 로그 그룹들을 식별 하도록 하였다. 로그 텍스트를 문자열과 가변인자로 받는 ALTIBASE 로그 시스템의 특성과 ALTIBASE 의 분할되어 출력되는 로그 텍스트들을 하나의 이벤트 로그로 통합하여 등록하기 위해서이다.

이벤트 ID 의 유일성을 제한하기 위해서 message DLL 파일에서 로그 텍스트 정의 시 그 수가 미리 정의된 문자열 변수를 사용할 수 있다는 점을 이용한다. 즉, 메시지 파일 작성시 다음과 같이 하나의 문자열 변수로만 정의된 일종의 템플릿과 같은 로그 텍스트를 정의한다.

```
MessageId = 1000
SymbolicName = MC_SERVER_0
Language = English
%1
.
```

그리고 ALTIBASE 의 로그 시스템이 로그 파일에 출력하는 텍스트를 버퍼에 저장하여 이를 이벤트 로그 등록 시 인자로 전달한다. 이렇게 하면 로그 파일에 출력되는 텍스트와 이벤트 뷰어를 통해 확인할 수 있는 이벤트 로그 텍스트가 일치하는 결과를 얻을 수 있다.

3.3 이벤트 로그 등록 함수 설계

ALTIBASE 의 로그 기록은 앞서 소개한 Logging_Function()에 의해 이루어진다. 이 함수는 출력하고자 하는 로그를 발생시킨 모듈, 로그의 레벨, 로그의 텍스트를 인자로 받아 해당 모듈에 해당되는 파일에 로그를 출력하게 된다. 하지만 실질적으로 로그 텍스트를 파일에 출력하는 것은 이 함수가 호출하는 Log_Description_Printing_Funcrion()이다.

본 논문에서는 Log_Description_Printing_Funcrion() 의 내부에 이벤트 로그를 등록하는 함수를 추가하는 방법을 제안한다. 새롭게 추가할 함수는 win32 API 에서 Winbase.h 에 정의된 ReportEvent 라는 함수를 사용하여 이벤트 로그를 등록하게 되며[4], 이벤트 ID, 로그 종류, 로그 범주, 로그 텍스트를 인자로 받는다. 이벤트 ID 는 본 제안에서는 유일성을 제한하여 모듈과 레벨로 분류된 로그 그룹을 식별하는 용도로 사용되었기 때문에 Logging_Function()이 호출될 때 인자로 받은 모듈과 레벨에 의해서 정의 될 수 있다. 로그 종류의 경우 ALTIBASE 의 로그 시스템에서는 분류를 하고 있지 않다. 따라서 윈도 이벤트 로그 서비스의 적용을 위해서는 모든 로그를 종류에 따라서 정보, 경고, 오류로 분류할 필요가 있다. 또한 이러한 로그 종류는 Logging_Function()이 호출될 때 인자로 전달되어야 한다. 로그 범주와 텍스트는 Logging_Function() 이 호출될 때 인자로 받은 모듈과 텍스트에 의해서 정의 될 수 있다.

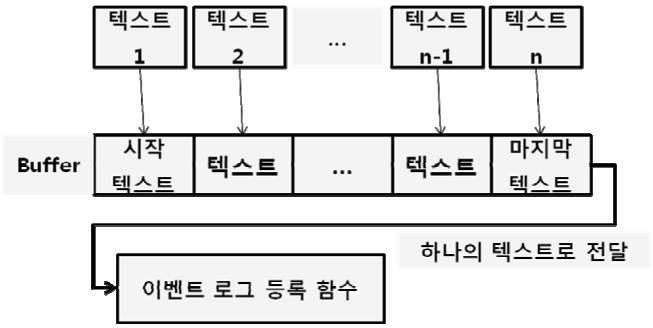
이러한 구조로 새로운 함수를 추가하게 되면 ALTIBASE 의 어떤 모듈에서 로그가 발생되어 해당 파일에 기록하게 될 때 파일에 기록되는 내용과 동일한 텍스트가 이벤트 로그로도 출력되게 된다.

3.4 분할된 로그의 통합 기능 추가

이벤트 로그를 등록하는 함수를 Logging_Function() 의 내부에서 호출하지 않고 Log_Description_Printing_Function() 함수의 내부에서 호출한 이유가 바로 분할된 로그의 통합을 위해서 이다. 앞서 설계한 방법은 로그가 분할되지 않았을 때의 경우이다. 하지만 실제로 분할된 로그의 출력을 위해서는 Logging_Function() 이 호출되지 않고 Logging_Function()의 내부에서 호출된 함수들이 개별적으로 호출된다. 즉, 먼저 Log_File_Open_Function()과 Log_Header_Printing_Function()이 호출되고 이후 각 과정별 결과를 출력하기 위해 Description_Printing_Function()이 반복되어 호출된다. 작업이 끝나게 되면 최종적으로 Log_File_Close_Function()이 호출되어 로그를 완성한다.

이러한 분할된 로그를 하나로 통합하여 이벤트 로그로 출력하기 위해서 전체 텍스트가 완성 될 때까지 출력된 텍스트들을 저장할 버퍼가 필요하며, 각 텍스

트들은 전체 텍스트의 어느 부분(시작, 중간, 끝)인지 정의되어야 한다. 이와 같은 조건이 만족된다면 그림 4 와 같은 구조로 출력되는 분할된 텍스트들을 버퍼에 저장하였다가 마지막 텍스트가 들어오면 이 버퍼를 하나의 텍스트로 취급하여 복사한 후 새롭게 추가된 이벤트 로그 등록 함수의 인자로 전달하는 방법을 사용할 수 있다.



(그림 4) 분할 텍스트의 통합 구조

여기서 사용되는 버퍼는 시작 텍스트의 입력 시 메모리가 할당되며 마지막 텍스트가 입력되면 메모리를 해제하게 된다.

이러한 설계를 위해서 Log_Description_Printing_Function()이 직접 호출될 때에는 해당 텍스트가 전체의 어느 부분인지를 정의하는 인자를 추가하여야 한다. 반면 Logging_Function()이 호출될 경우에는 분할된 로그가 아니므로 본래 설계를 따르도록 한다.

4. 결론

4.1 이벤트 로그 적용 방법 제안의 결론 및 의의

본 논문에서는 ALTIBASE 의 트레이서 로그 시스템과 윈도 이벤트 로그 서비스를 분석하여 ALTIBASE 에서의 윈도 이벤트 로그 적용 방법을 제안하였다. 제안된 방법은 이벤트 ID 의 유일성을 제한하여 message DLL 파일에 고정된 로그 텍스트를 정의하는 대신에 일종의 텍스트 틀을 정의하여 ALTIBASE 의 트레이스 로그 시스템에 대한 수정을 최소화하여 윈도 이벤트 로그 서비스를 적용시킬 수 있는 방법이다. 또한 하나의 로그 텍스트가 여러 텍스트로 분할되어 출력되는 ALTIBASE 로그의 특성에 대해서는 해당 텍스트가 전체의 어떤 부분인지를 정의하여 첫 번째 부분이 입력되면 버퍼에 메모리를 할당하고 저장하여, 이후 마지막 부분이 입력될 때까지 입력되는 텍스트들을 버퍼에 차례대로 저장한 후 마지막 부분이 입력되면 버퍼 전체를 하나의 텍스트로 취급하여 이를 복사한 후 버퍼에 메모리를 해제하고 복사 값을 인자로 전달하는 방법을 제안하였다.

본 제안은 ALTIBASE 의 트레이스 로그 시스템에 대한 수정을 최소화 하면서 윈도 이벤트 로그 서비스를 적용하도록 했다는 점에 그 의의가 있다.

4.2 추후 연구 과제

본 논문에서는 ALTIBASE 에서의 윈도 이벤트 로그 적용 방법을 제안하였다. 이 과정에서 ALTIBASE 의 트레이스 로그 시스템에 대한 수정을 최소화하기 위한 방법 중 하나로 각각의 로그 텍스트들을 유일하게 식별하는 이벤트 ID 의 유일성을 제한하는 방법을 사용하게 되었다. 하지만 이러한 방법은 사용자로 하여금 이벤트 로그를 식별하는데 불편함을 줄 수도 있다. 따라서 각각의 로그 텍스트들을 유일하게 식별하는 이벤트 ID 의 본래 성질을 유지하며 적용시킬 수 있는 방안에 대한 연구가 필요하다.

참고문헌

[1] 도안구 “윈도 서버용 DB 강자는 누구?”, http://eyeball.bloter.net/903

[2] John Enck, Philip Dawson, George J. Weiss, Rakesh Kumar, Jeffrey Hewitt, Uko Tian, Thomas J. Bittman. “Predicts 2008: Servers and Operating Systems”, 2007

[3] Jeffrey Richter, Jason D. Clack. “Programming Server-Side Applications for Microsoft Windows 2000”, Microsoft Press, 2000, Chapter 6

[4] Charles Petzold. “Programming Windows, 5/E : The Definitive Guide to the Win32 API” Microsoft Press, 1999