논문 2007-44CI-6-2

정보보호를 위한 고성능 백업 및 복구 시스템의 설계 및 구현

(Design and Implementation of High Efficiency Backup and Recovery Systems for Information Protection)

이 문 구*, 성 해 경**

(Moon-Goo Lee and Hae-Kyung Seong)

요 약

데이터의 증가 추세와 중요성을 고려할 때, 효율적인 대용량의 백업시스템이 필요하다. 그러나 기존의 백업시스템들은 저장매체의 속도와 기술적인 문제 등을 갖고 있다. 이러한 문제들을 해결하기 위하여 제안하는 고성능, 대용량 백업 및 복구시스템은 고속화, 압축 기술 및 백업 가속기 등을 이용하여 설계 및 구현하였다. 구현된 백업 및 복구시스템은 고속화를 위하여 멀티 쓰레딩 기술과 멀티 프로세싱 기술 그리고 멀티 스트리밍 기술을 적용하였다. 그리고 기존의 백업미디어가 테이프 기반이지만 제안하는 백업운영모델은 디스크기반으로 설계하였다. 때문에 구현된 백업시스템은 백업미디어의 한계를 극복하면서, 고속화 및 고용량이 가능하다.

Abstract

In consideration of increment and importance for data, an efficient and large storage backup system requires. Existing backup system solutions show some limitations in speed and technical. In order to solve these deficiencies, backup and recovery system of high efficiency and large storage capacity was designed and implemented by using high speed, compression technique and backup accelerator etc. Backup and recovery system applies to multi-threading, multi-processing and multi-streaming technology. And already established systems based on tape, but proposed backup operating model designed on disk. Therefore, the implemented of system leads to these backup media problems as well as solutions to aforementioned issues with existing backup system.

Keywords : Availability, Integrity, Multi-thread, Vulnerability, Compression, Backup accelerator

I. 서 론

광대역 통신기술의 보급 및 확산을 배경으로 각양각색의 분산시스템이 구성되며, 데이터와 프로세싱 모두 분산되어 사용될 수 있다. 이러한 분산 데이터 처리방식은 데이터에 대한 로컬 제어(local control)능력을 향상시키고, 데이터가 사용자에게 신속하게 제공되며, 통신기술력 향상은 물론, 통신비용 절감과 한 사이트의 컴퓨터가 다운(down)되었을 경우 대체 프로세싱을 가능하게 한다. 그러나 이러한 분산 데이터 프로세싱은 해커와 같은 외부 침입자에 의한 시스템 침투 가능성과 바이러스와 같은 종류의 침입 가능성 그리고 여러 개의 네트워크 노드들이 존재함에 따른 보안 취약점 발생의 가능성과 보안 관리의 어려움을 내포하고 있으며, 데이터의 무결성(integrity)에 대한 문제 등 보안에 대하여 매우 취약하다. 뿐만 아니라, 네트워크 장애가 발생할 경우 큰 어려움을 겪게 된다.

일반적으로 광대역 통신기술의 보급 및 확산을 배경으로 정보의 대용량화, 다양화가 급진전되면서 스토리지 소프트웨어도 기술 성장과 함께 높은 시장 성장세를 보이고 있다. 스토리지 소프트웨어는 물리적 저장매체

* 평생회원, 김포대학 IT 학부 인터넷정보과
(Dept. of Internet Information, Division of IT, Kimpo College)
** 평생회원, 한양여자대학 컴퓨터정보과
(Dept. of Computer Science & Information, Hanyang Women's College)
접수일자: 2007년10월4일, 수정완료일: 2007년11월4일

에 저장된 정보의 접근성(accessibility), 가용성(availability), 성능(performance)을 관리하고 보증하는 소프트웨어로 정의되어지며, 운영체제, 하부 시스템, 파일 시스템은 스토리지 소프트웨어에 포함되지 않는다. 일반적으로 스토리지 소프트웨어를 기능에 따라 4개로 분류하면 백업 및 저장 소프트웨어(backup and archive software) 스토리지, 자원관리 소프트웨어(storage resource management) 스토리지, 복제 소프트웨어(storage replication software) 스토리지, 유틸리티 및 시스템관련 스토리지 소프트웨어로 나눠지며, 본 연구에서 개발한 대용량 시스템 기반 스토리지 가속기는 백업 및 저장 소프트웨어에 포함된다. 이는 파일 및 디스크 백업, 복구, 파일 저장과 관련된 기능을 수행하는 소프트웨어를 의미한다. 갈수록 데이터의 양이 급증함에 따라 백업 솔루션들의 속도 향상 문제가 중요시되고 있는 현 시점에서, 기존의 스토리지 소프트웨어인 백업 솔루션들은 웹 가속기, 데이터베이스 가속기 등은 갖추고 있지만, 백업가속기는 갖추지를 못하고 있다. 때문에 본 연구에서 제안하는 고성능, 대용량 백업 및 복구시스템은 백업매체를 기존의 테이프 기반에서 디스크기반으로 구현하고, 백업 가속기를 갖춤으로써, 시스템과 네트워크[7] 리소스를 획기적으로 감소시킬 수 있으며, 고성능 스트리밍 기술과 병렬처리기술로 효율적으로 대용량 데이터를 백업받을 수 있도록 하였다. 그리고 저장장치의 용량문제와 백업 시스템의 효율적인 제어 및 관리를 위하여 실시간 그래픽 사용자 인터페이스 기술을 기반으로 한 중앙 집중 관리로 관리의 용이성은 물론 기존 백업 시스템과 연동할 수 있도록 설계 및 구현하였으며[8], 분산시스템 환경에서 구현하고자 할 때 하드웨어와 데이터는 분산처리 하고, 메타 데이터와 제어(control)는 중앙 집중 관리방법으로 설계하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 백업 시스템의 필요성과 기존 백업 시스템의 문제점에 대하여 기술하였고, Ⅲ장에서는 본 논문에서 제안하는 백업시스템의 구성요소와 모듈 그리고 핵심기술과 알고리즘 등을 기술하고, Ⅳ장에서는 제안한 시스템의 구현결과와 성능평가를 기술하였고, 마지막 Ⅴ장에서는 본 연구에 대한 결론을 기술하였다.

Ⅱ. 백업 시스템의 필요성과 문제점

1. 백업시스템의 필요성

일반적으로 백업은 절도, 손상, 그리고 환경적 문제로부터 안전하게 보호되어 저장되어야한다. 특히, 시스템을 실패나 오동작으로 몰고 갈 수 있는 소프트웨어의 실패는 시스템을 취약하게 만들거나 또는 시스템을 불안정하게 만들 수 있고 소프트웨어는 하드웨어보다 자주 변하므로 백업(back-up)절차가 지속적으로 이루어져야한다. 온라인 파일은 처리 로그(transaction log) 파일에 변화를 기록하며, 이는 원래의 파일과 분리한다. 특히 마스터 파일(master files)은 적당한 간격을 두고 갱신되어야 하며, 처리 파일(transaction files)은 이러한 마스터 파일과 일치되도록 보존되어야 한다.

보안 관리와 재난관리를 위해서 시스템 백업을 수행하는데, 시스템 백업이란 운영 가능한 응용프로그램, 데이터파일, 데이터베이스, 시스템 소프트웨어 제품, 시스템 개발 프로그램 그리고 유틸리티 프로그램 등을 포함한다. 이러한 시스템 백업은 하드웨어 고장, 디스크 손상, 정전, 소프트웨어 손상이나 재난으로부터 복구하고 재시작 할 수 있는 능력을 제공하거나 정보의 파괴를 막을 수 있다. 시스템 백업을 자주하는 것은 손상된 파일을 재구성(recovery)하는 것과 컴퓨터 프로그램의 재가동(restart)에 도움을 준다. 그러나 이러한 백업과 복구를 위하여 데이터베이스는 너무 자주 재구성 되어질 수 있으므로 성능저하 문제를 일으킬 수도 있고, 과도한 물리적 입출력은 처리지연을 야기 시킬 수도 있다.

뿐만 아니라, 정보화 사회에 접하면서 E-biz환경, ERP (Enterprise Resources Planning : 기업 내 통합정보시스템구축) /CRM (Customer Relationship Management : 고객관계관리), SCM (Supply Chain Management : 공급 망 관리)/KM(Knowledge Management : 지식관리), DW(Data Warehouse : 의사결정 지원을 위한 데이터베이스)/DM(Data Mining : 대량의 데이터로 부터 유용한 정보들을 추출하는 과정), e-Hub, EAI(Enterprise Application Integration : 기업 애플리케이션 통합) 등과 같은 비즈니스 환경에서 대량의 데이터에 대한 안정성의 문제는 하드웨어 및 운영체제의 장애, 애플리케이션의 장애, 전원불안정, 환경적인 장애 그리고 홍수 및 지진과 같은 자연재해 장애요인으로 인하여 중요한 사회의 이슈가 되고 있다. 그러나 치명적인 데이터 손실로부터 고통 받는 기업 중 6%만이 생존하는 반면, 43%는 업무를 재개하지 못하고, 51%는 재개가 불가능한 경우가 현실이다. 그러므로 대량의 데이터를 안전하게 백업하고 재난에 대하여 복구하는 것은 정보화 사회의 필수 요건이라고 할 수 있다.

그러나 기존의 백업 시스템들은 분산 처리 및 시스

템 환경에서 외부 침입자의 침투, 바이러스, 악성 코드 등의 침입 등 보안 관리의 어려움과 데이터의 기밀성, 무결성 그리고 가용성에 대하여 대단히 취약한 면을 갖고 있다. 뿐만 아니라 무엇보다도 대용량의 데이터를 어떠한 가공도 하지 않고 백업 시스템에 보관하는 것은 데이터 백업에 사용되어지는 미디어의 소모량 등에 많은 부담을 주게 되며, 다른 시스템으로 백업 데이터를 전송하고자 할 때에도 네트워크 대역폭에 부담을 주게 된다.

그러므로 본 논문에서는 기존의 백업시스템의 성능 저하 문제와 과도한 미디어 소모량 등의 부담을 줄이기 위하여 초고속 압축[1] 및 복원 알고리즘과 데이터 스트리밍기술로 병렬처리를 함으로써 효율적인 백업 및 복구를 위한 고성능 보장과 동시에 신속성과 관리의 편이성을 충족시키도록 구현하였다.

Ⅲ. 제안하는 고성능 백업 및 복구시스템

1. 백업시스템의 구성요소

본 논문에서 제안하는 백업 및 복구시스템은 [그림 1]과 같이 마스터계층, 서버(엔진)계층, 클라이언트(에이전트)계층 등 3계층(Tier)으로 구성되며, 이러한 3계층 구조는 분산처리 환경에 구축된 시스템들이 갖는 보안 문제를 효율적으로 제어할 수 있도록 하였으며, 기존의 백업 시스템과 연동 하고자 할 때 증설과 확장이 용이하도록 설계하였다.

[그림 2]는 백업 시스템의 내부구성요소를 도식화 한 것이다.

서버(엔진) 계층은 백업정책에 따라 서버 백업데이터와 백업로그를 관리하며, 마스터로부터 백업정책 등록과 수동백업 및 복구명령을 받아서 수행한다. 그리고 등록된 스캐쥴 정보에 맞춰 자동백업을 수행하며, 만기

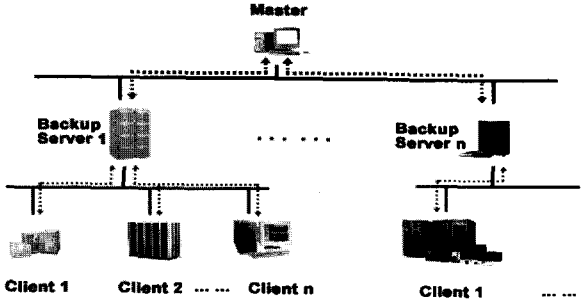

그림 1. 세 계층 구조
Fig. 1. 3 Tier Structure.

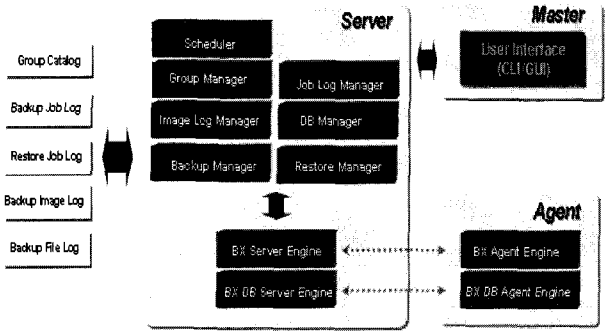

그림 2. 백업 시스템의 구성
Fig. 2. Configuration of Backup system.

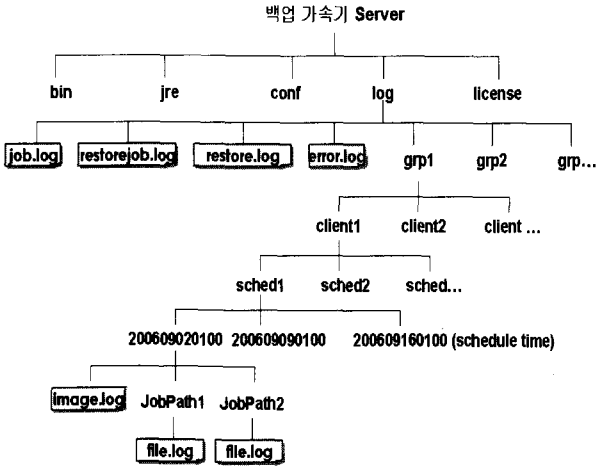

그림 3. 백업가속기 서버의 로그파일
Fig. 3. Log file of Backup accelerator server.

일이 지난 백업 이미지(데이터)는 자동 삭제한다. 그리고 에이전트에 백업 및 복구시작 명령을 전달하고 에이전트로부터 데이터를 수신하여 압축[6], 저장하며 작업대상 증가에 따른 유연한 확장성을 제공한다. 이처럼 서버계층은 백업서버에 구성정보 및 수행명령어를 전송하면, 백업 및 복구 상태를 모니터링 하여 사용자에게 전달 할 뿐만 아니라 로그파일[그림 3]에 의해 백업로그(이미지로그, 파일로그, 에러정보 등)를 검색한다. 그리고 수동백업 및 복구요청을 수행하여, 단일 마스터로 또는 중앙 집중 관리 체계로 관리할 수 있도록 한다. 로그파일은 그룹 카탈로그 로그(Group Catal Log), 백업로그(Backup Log), 리스토어 작업 로그(Restore Job Log), 백업 이미지 로그(Backup Image Log), 백업 파일 로그(Back File Log)등이다.

마스터 계층의 주 기능은 백업서버들의 정보를 원격지에서 관리한다. 그렇기 때문에 사용자와 그래픽(GUI) 혹은 명령어(CLI) 등의 인터페이스로 백업서버와 사용자가 실시간 서비스를 제공할 수 있도록 한다. 그리고

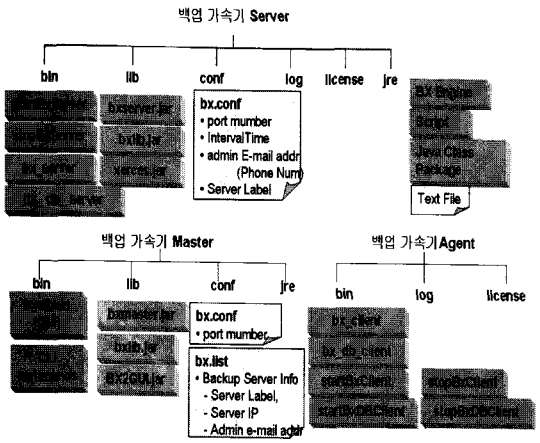

그림 4. 백업 시스템 모듈

Fig. 4. Backup system module.

```
<group>
  <groupname> nt_test </groupname>
  <activestate> false </activestate>
  <compression> true </compression>
  <savedirectory> /export/test1 </savedirectory>
  <clientlist>
   <client>
    <clientname> 61.250.236.168 </clientname>
    <hardware> intel </hardware>
    <os> win2000 </os>
    <maximumjob> 3 </maximumjob>          ← 동시에 백업 되는 Job 수
    <multistream> 2 </multistream>        ← Thread number per a Job
    <clientfilelist>
     <file> e:\bxdata </file>
    </clientfilelist>                     ← Backup Job Path in a Client
    <backuptimelist>
     <backuptime label="f_sched"> 2002 09 11 09 05 ← 마지막 백업 성공 시간
    </backuptime>
    </backuptimelist>
   </client>
  </clientlist>
  <schedulelist>                          ← Backup Job Path in All Client
   <schedule>
    <schedulelabel> f_sched </schedulelabel>
    <scheduletype> FULL </scheduletype>
    <starttime> 2006 03 03 03 02 </starttime>  ← 다음 백업 진행 시간
    <nexttime> 2006 09 03 03 02 </nexttime>
    <period> 3 </period>                  ← 백업 주기
    <duration> 3 </duration>              ← 백업 이미지 보유 기간
   </schedule>
  </schedulelist>
  <groupfilelist></groupfilelist>
</group>
```

그림 5. 그룹 카탈로그 소스 코드

Fig. 5. Group catalog source code.

백업서버 등록, 서버별 그룹관리, 대상별 백업 담당자의 정보(E-mail, phone등) 등록, 스케줄, 파일로그, 에러정보 등의 백업정책들을 등록, 수정, 삭제 및 관리하며 수동백업 및 복구 명령을 실행한다. [그림 4]는 백업 시스템의 모듈을 도식화 한 것이다.

마지막으로 클라이언트(에이전트) 계층은 백업데이터를 보유한 n개의 운영서버로 구성되며, 대상 데이터를 고속으로 읽고 서버로 전송한다. 클라이언트의 에이전트를 그룹으로 관리하여 백업정책의 적용 및 관리를 용이하게 하며, 원거리에 있는 서버의 백업도 가능하다. 서버로부터의 백업 요청 시 조건에 맞는 데이터를 검색하여 백업 서버로 전송하고, 서버로부터의 복구 요청 시 데이터를 받아서 복구한다. 백업 및 복구시스템의 백업가속기는 서버와 에이전트를 연동하여 백업 및 복구 정책을 구현한다. 백업가속기 서버 엔진은 백업가속기 스캐줄러 알고리즘과 백업 정책에 따라 클라이언트로부터 데이터를 압축하고 백업가속기 백업 데이터가 존재하는 에이전트와 연동되며, 등록된 스케줄 정보에 따라 자동 백업을 수행하며, 만기일이 지난 백업 이미지(데이터)는 자동 삭제하도록 한다. 백업 가속기 데이터베이스[4] 엔진은 백업가속기 에이전트 엔진과 연동한다.

백업가속기의 그룹(Group)이란 개념은 동일한 백업 정책을 가진 클라이언트와 스케줄의 집합으로, [그림 5]는 그룹 카탈로그 소스 코드이며, 백업 정책은 이러한 그룹화를 통한 용이한 관리기능을 제공하고, 스케줄 등록에 의한 자동백업과 백업정책에 등록되지 않은 특정 스케줄의 수동 백업, 그리고 백업된 데이터의 유지 기간 지정(자동 삭제)을 수행하도록 한다.

2. 시스템의 핵심 구성

가. 다양한 백업 기능

제안하는 고성능 백업 및 복구 시스템은 다양한 데이터베이스 백업[3] 기능을 제공하기 위하여 풀(full)백업과

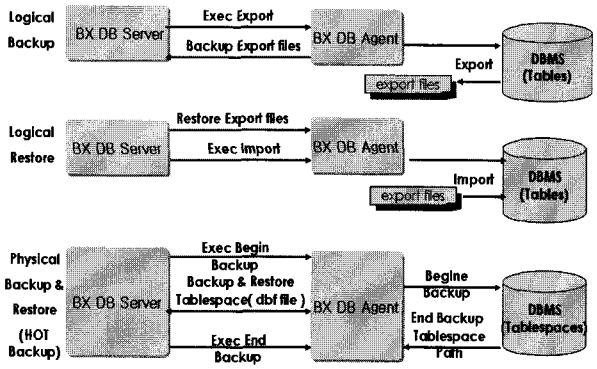

그림 6. 데이터베이스 백업

Fig. 6. DataBase Backup.

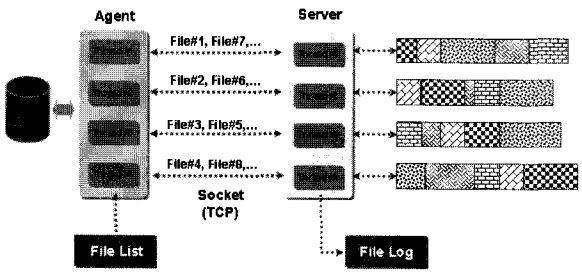

그림 7. 파일 백업 멀티 스트리밍

Fig. 7. File Backup Multi-Streaming.

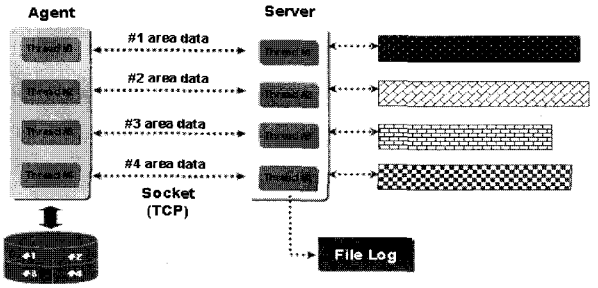

그림 8. 볼륨 백업 멀티 스트리밍
Fig. 8. Volume Backup Multi-Streaming.

증분(incremental)백업을 파일 및 볼륨단위로 [그림 6] 백업하며, 압축 및 복원을 위한 알고리즘으로 하위 디렉토리나 특정 파일에 대한 부분 복구가 가능하도록 구현하였다.

고속백업을 위하여 다수의 작업들을 백업 디렉토리를 이용하여 동시 백업이 가능하도록(multiple job backup)설계하였다. [그림 7]은 에이전트가 파일을 멀티 스트리밍 하기 위하여 4 쓰레드[2] 파일 백업을 도식화 한 것이다.

[그림 8]은 에이전트가 볼륨 백업 멀티스트리밍 하기 위하여 4 쓰레드 볼륨 백업 과정을 도식화 한 것이다. 이처럼 멀티 스트리밍을 위하여 한 개의 작업을 멀티 쓰레드 병렬 기법으로 구현하여 여러 개의 디스크를 동시에 접근해서 데이터를 고속으로 백업 서버로 전달하는 다중화 백업을 수행하도록 구현하였다.

나. 고속 백업 동작

기존의 스토리지 소프트웨어인 백업 솔루션들은 웹 가속기, 데이터베이스[5] 가속기 등은 갖추고 있지만, 백업가속기를 갖추지 못하고 있기 때문에 [그림 9]처럼 서버에서 제어하여 스토리지의 데이터 52GB를 테이프 라이브러리에 저장하는데 소요되는 시간은 평균 약 3시간51분 정도였다.

반면에 제안하는 백업시스템은 [그림 10]처럼 스토리지의 데이터를 1차 압축한다. 이때 압축을 위해 소요되

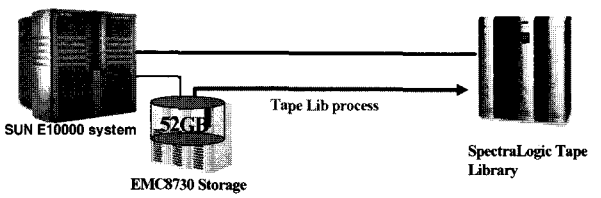

그림 9. 일반적인 백업 시스템
Fig. 9. General Backup System.

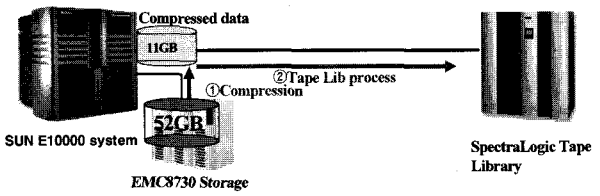

그림 10. 제안하는 백업 시스템
Fig. 10. Proposed Backup System.

는 시간은 25분29초 이며, 두 번째 단계에서는 52GB의 데이터를 11GB로 압축하여 테이프 라이브러리에 저장하는데 15분51초이고, 총 소요되는 시간은 평균 66분정도였다. 때문에 압축 및 백업에 소요되는 시간과 압축률은 기존의 백업시스템보다 약 3~5배정도 높아졌고, 시간도 약 1/3~1/5정도로 단축이 되었다.

3. 제안하는 백업 시스템 알고리즘

[그림 11]은 백업가속기 스케줄러 알고리즘으로, 백업가속기의 예외 처리 상황을 실행하고, 그렇지 않은 경우는 call에 대한 "add" 이 수행 되도록 한다.

```
//Parse
public void process(String[] pArgs)
{   //parses arguments
//sets the server
//sets attributes
    //if error exist print "bxException"
    //else call correspondent method
        //if "-add" call addSchedule ()
        //if "-modify" call modifySchedule ()
        //if "-delete" call deleteSchedule ()
        //if "-deleteall" call deleteAllSchedule ()
        //if "-show" call getSchedule()
    //print aOutStr
        //if(aOutStr != null) print aOutStr   }
//Add Schedule to Group
public void addSchedule(String pAttribute)
{   String sMessage        // this is combined command with
values
     //constructs a sMessage :bxAddSchedule+ pAttribute
     //calls sendMessage(sMessage)
        //if  Group  doesn't  exist  append  the
result(bxException&INVALID_GROUP) to aOutStr
//if  same  Schedule  exist  append  the
result(bxException&FOUND) to aOutStr
        //else append the result(bxAck) to aOutStr
}
```

그림 11. 백업가속기 스케줄러 알고리즘
Fig. 11. Backup accelerator scheduler algorithm.

[그림 12]는 백업가속기 스케줄러 알고리즘으로 "modify", "delete", "deleteall", "show"등을 call하도록 한다.

```
//Modify Schedule
public void modifySchedule(String pAttribute)
{
    String sMessage
   // this is combined command with values
     //constructs a sMessage: bxModifySchedule+ pAttribute
    //calls sendMessage(sMessage)
        //if Group doesn't exist
append the result(bxException&INVALID_GROUP) to aOutStr
       //if Schedule doesn't exist
append the result(bxException&NOT_FOUND) to aOutStr
//else append the result(bxAck) to aOutStr
}
//Delete Schedule from Group
public void deleteSchedule(String pAttribute)
{
      String sMessage
      // this is combined command with values
//constructs a sMessage: bxDeleteSchedule+ pAttribute
  //calls sendMessage(sMessage)
      //if Group doesn't exist
append the result(bxException&INVALID_GROUP) to aOutStr
   //if Schedule to delete doesn't exist
append the result(bxException&NOT_FOUND) to aOutStr
//else append the result(bxAck) to aOutStr
}
//Delete All Schedules from Group
public void deleteAllSchedule(String pA·ttribute)
{  String sMessage
      // this is combined command with values
    //constructs a sMessage: bxDeleteAllSchedule+ pAttribute
    //calls sendMessage(sMessage)
        //if Group doesn't exist
append the result(bxException&INVALID_GROUP) to aOutStr
        //if Schedule to delete doesn't exist
append the result(bxException&NOT_FOUND) to aOutStr
//else append the result(bxAck) to aOutStr
}
//Get Shchedule
public void getSchedule(String pAttribute)
{  String sMessage
 // this is combined command with values
        //constructs a sMessage: bxGetSchedule+ pAttribute
        //calls sendMessage(sMessage)
            //if Group doesn't exist
  append the result(bxException&INVALID_GROUP)to aOutStr
            //if Schedule doesn't exist
  append the result(bxException&NOT_FOUND)to aOutStr
            //else append the result(bxInfoSchedule&... ...)
            to aOutStr   }
```

그림 12. 백업가속기 스케줄러 알고리즘(계속)

Fig. 12. Backup accelerator scheduler algorithm(con).

다. 백업가속기의 복구 알고리즘

[그림 13]은 백업 가속기의 목록(list) 명령어를 실행하는 알고리즘이다.

[그림 14]는 백업 가속기의 복구(recovery) 알고리즘이다.

```
public void process ()
{
    // parses arguments
    // sets a backup server hostname( selected or default
backup server )
    // sets attributes
    // calls related method
    // prints the result ( aOutStr.toString() )
}
public void searchImages ()
    String sMsg = "";        // message to Backup Server
    String sResult = "";     // result from Backup Server
    // constructs a message for sending to Backup Server
"sMsg"
    sResult = sendMessage( sMsg );
    // construct a result for user format
    // append result to aOutStr    }
public void searchFiles ()
{   String sMsg = "";        // message to Backup Server
    String sResult = "";     // result from Backup Server
    // constructs a message for sending to Backup Server
"sMsg"
    sResult = sendMessage( sMsg );
    // construct a result for user format
    // append result to aOutStr   }
public void deleteImages ()
 }
{   String sMsg = "";        // message to Backup Server
    String sResult = "";     // result from Backup Server
    // constructs a message for sending to Backup Server
"sMsg"
    sResult = sendMessage( sMsg );
    // construct a result for user format
    // append result to aOutStr   }
```

그림 13. 백업가속기 목록 명령 알고리즘

Fig. 13. Backup accelerator list command algorithm.

```
   // main process
   public void process ()
   {  // parses arguments
     // sets a backup server hostname( selected or default
backup server )
     // sets attributes
     // calls related method
     // prints the result ( aOutSt1r.toString() )  }
   public void restore()
   {  String sMsg = ""  // message to Backup Server
     String sResult = ""  // result from Backup Server
     // constructs a message for sending to Backup Server
"sMsg"
     sResult = sendMessage( sMsg );
     // append result to aOutStr   }
public void runClientFilesBackup ( )
{
    String sMsg = ""        // message to Backup Server
    String sResult = ""     // result from Backup Server

    // constructs a message for sending to Backup Server
"sMsg"
    sResult = sendMessage( sMsg );
    // append result to aOutStr
}
```

그림 14. 백업가속기 복구 알고리즘

Fig. 14. Backup accelerator recovery algorithm.

IV. 구현결과 및 성능 평가

가. 구현 결과

[그림 15]는 제안하는 백업 및 복구 시스템의 대화창으로 백업환경의 설정, 갱신, 삭제를 위한 자동백업 스케줄러를 설정하여 백업시간, 주기 등을 지정하고 서버별 그룹관리 기능과 대상별 백업 담당자정보

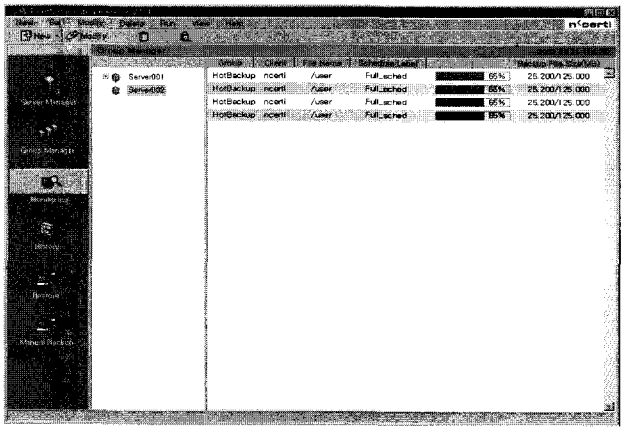

그림 15. 백업 시스템의 스케줄러
Fig. 15. Scheduler of Backup system.

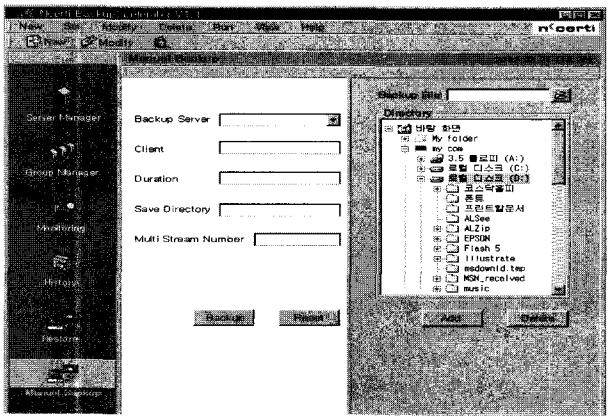

그림 16. 매뉴얼 백업 및 복구 화면
Fig. 16. Screen of Manual Backup and Recovery.

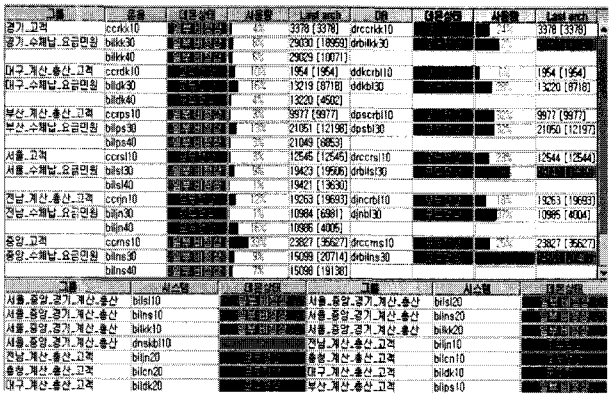

그림 17. 백업 실시간 모니터링 화면
Fig. 17. Screen of Backup real-time monitoring.

(Email/Phone등)을 등록할 수 있다[9, 12].

[그림 16]은 일회 백업 등을 위한 매뉴얼 백업기능과 백업된 데이터를 특정디스크로 복구하는 기능을 제공하도록 구현된 대화창이다[10].

[그림 17]은 백업 시스템의 실시간 모니터링 기능을 실행할 수 있는 대화 화면으로서 백업진행 상태 및 디스크 여분을 보여주는 기능과 장애 발생 시 담당자에게 알려주는 기능 그리고 백업 히스토리 관리를 위하여 과거의 모든 백업 히스토리 정보를 관리하는 기능과 특정기간에 발생한 모든 백업결과를 검색하는 기능 등을 제공한다.

나. 성능 평가

압축률 테스트를 위한 환경은 CPU가 Intel P4 1.3GHz 메모리는 256 MB 하드디스크는 40G UDMA/100 그리고 운영체제는 WindowsXP에서 데이터는 948MB Volume 776 MB이며, 73,000개 파일을 사용하였으며, [표 1]은 제안하는 백업 및 복구시스템과 다른 시스템과의 압축률 비교결과 자료이다[11].

기존의 백업 시스템들은 압축률이 59%이고, 속도는 최대 1.7MB/Sec를 초과하지 못하였으나 제안하는 백업 시스템은 압축률 68%에 2.0MB/Sec의 속도를 보여주었다.

백업시간 테스트 환경은 성능평가 기준(benchmark) 시스템 환경에서 시스템 이름은 ms1, ms2, ms3, ms4로 하였으며, 하드웨어의 머신 타입은 4 * SUN E4500, CPUs는 4 * 10 CPUs 메모리 용량은 4 * 9 GBytes, 디스크 용량은 16 * 480GBytes (total 7680GBytes, Sun), 입출력 채널의 수는 4 fiber Channel이다. 소프트웨어로는 Solaris2.6운영체제이고 성능평가 기준의 대상 파일 시스템의 파일 용량은 6.768Tbytes 이다.

표 1. 백업시스템의 압축률 비교
Table 1. Compression rate comparisons of backup system.

백업 시스템	압축률	속도
제안하는 백업시스템	68%	2.0 MB/sec
WinRAR (Window기반)	55%	0.47 MB/sec
Gzip (Linux기반)	59%	1.7 MB/sec
A사 백업 S/W	53%	0.12 MB/sec

표 2. 백업 볼륨 시간 비교
Table 2. Compression of Volume Backup time Table.

백업 시스템	Volume Backup without Backup accelerator (48GB)	Volume Backup with Backup accelerator (48GB)
백업시간	2h 30m	15m

표 3. 백업 시간 비교(압축시간 포함)
Table 3. Compression of Backup time Table. (*included compression time)

System / Name Disk	ms1	ms2	ms3	ms4
Tray 1	6h 53m	6h 54m	7h 1m	7h 2m
Tray 2	6h 53m	6h 54m	7h 1m	7h 2m
Tray 3	6h 54m	6h 54m	7h 2m	7h 2m
Tray 4	6h 54m	6h 54m	7h 1m	7h 2m

표 4. 백업 시스템의 비교
Table 4. Compression of Backup System.

구 분	제안한 백업 시스템	국외			국내	
		V사	L사	C사	B사	R사
백업 데이터 압축률	60 ~80%	50 ~60%	50 ~60%	50 ~60%	50%	50%
압축백업 속도	빠름	보통	보통	보통	보통	보통
다중 스트리밍	○	○	○	○	×	×
타 솔루션 연동	○	×	×	×	×	×

"ms1"시스템의 경우 총 데이터는 1.692TB 이고, 압축된 데이터는 399.2GB 그리고 압축률은 23.6% (1.692TB -> 399.2GB)이었으며 평균 압축 속도는 68.2MB/sec (Tray 1)이었다. [표 2]는 제안하는 백업시스템은 백업 가속기를 가동함으로써 기존의 2h30m이 소요되던 백업시간을 15m으로 단축함으로써 백업은 10배의 속도향상을 보여 주었다. [표 3]은 성능평가를 실시한 시스템 이름을 ms1, ms2, ms3, ms4라고 임의로 정의하고, 각 시스템에 대한 압축 및 백업 시간의 산출 결과 이며, m1, m2, m3, m4의 각 디스크 트레이의 압축률을 포함한 백업시간은 평균 6h54m에서 7h2m으로 기존의 시스템보다 약 1/3에서 1/5배 정도 단축되었다.

[표 4]는 국내 및 국외의 백업시스템과 제안한 백업시스템과의 백업 데이터 압축률, 압축 백업 속도 그 그리고 다중 스트리밍의 시행 여부와 타 솔루션과의 연동 여부를 비교한 자료를 정리한 자료로서 데이터 백업 압축률은 10%이상이 향상되었으며, 타 시스템과의 연동이 가능함을 알 수 있다.

V. 결 론

분산 환경 시스템에서 백업 및 복구 시스템들은 보안에 대하여 많은 취약점을 갖는다. 또한, 기존의 백업 및 복구 시스템들은 데이터양이 증가하면 속도가 매우 느려져서 시스템 백업에서 실시간 활용이 거의 불가능하다. 그러나 제안하는 고성능 백업 및 복구 시스템은 다중 데이터 스트림, 입·출력 다중 클라이언트, 다중 프로세스, 다중 쓰레드(thread) 등의 단위 지원이 가능하도록 구현하였다. 그리고 파일 개수가 많은 파일시스템 백업의 성능향상을 위해 다수의 쓰레드로 동일 파일시스템의 파일을 자동 분할하여 다중 백업하는 기능을 제공한다. 이는 여러 개의 디스크를 동시에 병렬로 접근하고 이를 고속으로 백업 서버로 전달하는 기술을 사용하여 백업속도의 최적화를 구현하였다. 이러한 고성능 백업은 백업가속기를 두어 그룹화 작업과 압축 기술로 가능하다. 또한 백업 시스템이 서버, 마스터, 에이전트의 3 계층으로 구성되어 분산 처리 및 중앙 집중 관리 등 구성의 융통성과 확장성을 갖으며, 기존 백업시스템과의 연동이 용이하여 비용절감의 효과를 갖도록 구현하였다. 그 결과 압축 및 백업에 소요되는 시간과 압축률은 기존의 백업시스템보다 약 3에서 5배정도 높아졌고, 시간도 약 1/3에서 1/5정도로 단축이 되었으며, 압축률도 2.0 MB/sec 로서 기존 Linux 기반 백업시스템의 최대 59% 보다도 향상된 68%의 결과를 도출하였다.

참 고 문 헌

[1] B. R. Iyer and D. Wilhite : "Data Compression Support in Databases", VLDS, 1994.
[2] Bil Lewis, Daniel J. Berg, : "Threads Primer - A Guide to multithreaded Programming-" SunSoft Press, A Prentice Hall Title, 1996.
[3] M. A. Roth and S. J. Van Horn : "Database Compression", SIGMOD Record 22(3), 1993.
[4] Silvana Castano, Mariagrazia Fugini, Giancarlo Martella, Pierangela Samarati, : "DATABASE

SECURITY", ADDISION-WESELY, 1994.
[5] T. Westmann, D. Kossmann, S. Helmer, and G. Moerkotte : "The Implementation of Compressed Database", SIGMOD Record 29(3), 2000.
[6] Steve Kleiman, Devang Shah, Bart Smaalders, : "Programming with Threads", SunSoft Press, A Prentice Hall Title, 1998.
[7] Charlie Kaufman, Radia Perlman, Mikes Speciner, "Network Security", PTR Prentice Hall, 1995.
[8] Carles Arehart, Nirmal Chidambaram, etc. "Professional WAP" Wrox. 2000.
[9] http://www.lsbu.ac.uk/oracle/oracle7/serve r/doc/EBADM/chap6.htm#996993
[10] http://www.openssl.org/
[11] http://www.unisoninfo.com/datasheets/Synthetic-Full-Backup-Data-Sheet.pdf
[12] http://www.ndmp.org/ 의 NDMP Specification

저 자 소 개

이 문 구(평생회원)
1984년 숭실대학교 전자계산학과 학사 졸업.
1993년 이화여자대학교 전산교육학 석사졸업.
2000년 숭실대학교 컴퓨터 시스템 공학박사 졸업.
2000년~현재 김포대학 IT학부 인터넷정보과 부교수
<주관심분야 : 네트워크 프로그래밍, 인터넷 보안, 시스템 보안, 암호화 알고리즘, 전자상거래 보안, 침입 탐지 시스템, 침입차단시스템>

성 해 경(평생회원)
1976년 홍익대학교 전자계산학과 학사졸업.
1978년 홍익대학교 대학원 전자계산학과 석사졸업
2000년 대전대학교 대학원 컴퓨터 공학과 박사졸업
1987년~현재 한양여자대학 컴퓨터정보과 교수
<주관심분야 : 소프트웨어공학, 멀티미디어 시스템, 웹 프로그래밍, 정보보호 시스템>