JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 12, pp. 9-16, December 2022
https://doi.org/10.9708/jksci.2022.27.12.009

Design and Implementation of User-Level FileSystem in the Combat Management System

Seok-Hyun Kang*, Keun-Hee Kim*

*Engineer, Naval R&D Center, Hanwha Systems, Gumi, korea
*Engineer, Naval R&D Center, Hanwha Systems, Gumi, korea

[Abstract]

In this paper, we propose a plan to design and utilize the RDBS(Record Block Data file management System) so that data can be recovered when data files in the Combat Management System are mismatched. The CMS(Combat Management System) manages the same files in multiple IPN(Infomation Processing Node) repositories to support multiplexing. However, mismatches in data files can occur due to equipment maintenance or user immaturity. The existing CMS does not manage the history of changes in data files, and when a mismatch occurs, data file were synchronized based on the latest date. But, It is difficult to say that files with the latest date have the highest reliability, and once the file synchronization has progressed, it cannot be recovered with pre-synchronization data. To solve this problem, data was stored and synchronized in units of record blocks using RDBS proposed in this paper, and the Rsync algorithm was used to reduce the overhead of file synchronization due to units of record blocks. SW applied with RDBS was tested for performance in a simulated environment, and it was confirmed that it could be applied to CMS through normal operation confirmation.

▸ **Key words**: Combat Management File System, Record file management, Record-Level Synchronization, Backup and Restore

[요 약]

본 논문에서는 함정 전투체계의 데이터 파일의 데이터 불일치가 발생했을 때 데이터 복구를 할 수 있도록 레코드 블록 데이터 파일 시스템(RDBS)의 설계 및 활용방안을 제안한다. 함정 전투체계는 다중화를 지원하기 위해 다수의 정보처리장치 저장소에 동일한 파일을 관리하고 있다. 하지만 일부 장비의 유지보수로 인한 사용불가 상태에서 운용하거나 사용자의 운용미숙으로 정보처리장치 간의 데이터 파일의 불일치가 발생할 수 있다. 기존 함정 전투체계는 데이터파일의 변경이력을 관리하지 않으므로 데이터 불일치가 발생했을 때, 가장 최신날짜 기준으로 데이터 파일을 동기화를 진행하였다. 그러나 가장 최신 날짜의 데이터파일이 신뢰도가 가장 높다고 보기 어려우며, 한번 파일동기화가 진행된 이후에는 동기화 이전 데이터로 복구할 수 없다. 이러한 문제점을 해결하기 위해 본 논문에서 제안한 RDBS를 활용하여 레코드 블록 단위로 데이터를 저장 및 동기화 하였으며, 레코드 블록 관리로 인한 파일동기화에 발생하는 오버헤드를 줄이기 위해 Rsync알고리즘을 활용하였다. RDBS를 적용한 SW를 모의환경에서 성능시험을 하였으며, 정상 동작확인을 통해 함정 전투체계에 적용이 가능함을 확인하였다.

▸ **주제어**: 함정 전투체계, 파일동기화, 레코드 파일 관리, 레코드 동기화, 백업 및 복원

• First Author: Seok-Hyun Kang, Corresponding Author: Keun-Hee Kim
*Seok-Hyun Kang (sh14.kang@hanwha.com), Naval R&D Center, Hanwha Systems
*Keun-Hee Kim (winterkkh@hanwha.com), Naval R&D Center, Hanwha Systems
• Received: 2022. 11. 08, Revised: 2022. 11. 24, Accepted: 2022. 11. 29.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

함정 전투체계 내에서 전투관리체계(CMS: Combat Management System)는 인간의 두뇌에 해당하는 역할을 수행함으로써, 통신체계, 센서체계, 무장체계 등을 통제하고 관리하는 핵심적인 체계이다. CMS 는 센서 및 무장으로부터 메시지를 수신하여 명령 메시지를 송신하는 역할을 수행하는 연동단, 다기능콘솔(MFC: Multi Function Console), 레이더 및 TV비디오 처리장치 그리고 체계지원(System Support) 과 같은 기능을 수행하는 정보처리장치(IPN: Information Processing Node) 등으로 구성되어 있다[1][2].

정보처리장치의 경우 함정 전투체계 생존성 향상을 위해 다중화를 지원하고 있다. 다중화란 하나의 정보처리장치가 사용불가가 되더라도 다른 정보처리장치가 그 기능을 이어받아 수행할 수 있는 개념이다. 다수의 정보처리장치에 함정 전투체계 소프트웨어 및 데이터 파일이 탑재되며 데이터 파일은 정보처리장치 간 파일동기화 및 실시간 사용을 보장할 수 있어야 하므로 중앙관리 방식인 DBMS(Data Base Management System)를 사용하는 대신 기능별 데이터를 파일 단위로 관리하고 있다. 파일 단위로 데이터를 관리하기 때문에 수정된 데이터에 대한 롤백기능이 제공되지 않으며 정보처리장치가 모두 가용상태가 아닌 경우 정보처리장치 간 데이터 파일 불일치가 발생할 수 있다.

본 논문에서는 함정 전투체계의 데이터 파일의 실시간 동기화를 보장하며 시점 별로 데이터 저장 기능을 통해 데이터 불일치로 인한 데이터 소실을 복구 할 수 있도록 레코드 블록 데이터 파일 관리시스템(RBDS: Record Block Datafile System)을 제안한다. 레코드 블록 관리 시스템은 파일단위 관리의 문제점을 해결하기 위해 DBMS[3]의 레코드 구조를 파일내부에서 관리하며 실시간 파일 사용 및 파일동기화를 보장함으로써 분산된 파일을 데이터베이스 레코드처럼 활용할 수 있는 기능을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 통해 함정 전투체계의 데이터 파일 사용 현황에 대해 설명하고 레코드 블록 단위로 인해 발생하는 오버헤드를 최소화하기 위한 Rsync 알고리즘을 설명한다. 3장에서는 레코드 블록 단위의 파일시스템 설계 및 파일처리 알고리즘에 대해 설명한다. 4장에서는 기존 파일시스템 대비 제안한 파일시스템의 성능 분석 및 레코드 블록 단위의 데이터 복구를 확인하며 마지막 5장에서 결론으로 논문을 마무리한다.

II. Preliminaries

1. Data File in Combat Management System

함정 전투체계에서 사용하는 파일은 전술데이터 파일과 영상 파일, 레이더비디오 파일로 분류 할 수 있다. Table 1은 함정 전투체계에서 사용하는 전술 데이터, 비디오, 영상 파일의 평균 파일사이즈를 확인했으며, 전술데이터 파일은 복수의 정보처리장치에 저장된다. 전술데이터 파일은 평균 10KB수준의 파일사이즈로 가장 최신의 파일을 기준으로 파일동기화를 보장하고 있다. 영상 파일과 레이더 비디오 파일은 분산된 저장소에 저장하지만 파일 사이즈가 큼에 따라 파일동기화를 하지 않는다.

Table 1. Tactical Data File in CMS

Category	File size(average)	File Sync
Tactical	10 KB	O
Video	500 MB	X
Radar	152 MB	X

2. Rsync

저자 Ho Min Jung[4]의 연구에서는 중복제거를 통해 중복되지 않은 데이터를 전송하는 경우 파일동기화 시간을 줄일 수 있음을 확인하였다. 중복을 처리하면서 파일동기화하는 대표적인 연구로 Rsync가 있으며 롤링 체크썸(Rolling Checksum)이라는 중복 데이터를 검색하는 알고리즘을 사용해 새로운 데이터의 복사만 일어난다.

Rsync는 네트워크상의 원격지와 로컬의 파일을 동기화하는 알고리즘이자 툴이다. 파일을 효율적으로 동기화하기 위한 방법으로 중복제거 기법을 이용할 수 있다[5]. Rsync는 파일 동기화를 위한 데이터 전송 시 네트워크 대역폭을 최소화 하기 위해 파일동기화를 진행하는 두 파일의 차이점만 전송하는 델타 인코딩 알고리즘(Delta Encoding Algorithm)을 사용하였다[6][7].

Rsync의 델타 인코딩은 파일을 일정 크기의 블록을 나누어 각 블록의 체크썸을 계산 및 비교함으로써 두 파일의 사이의 내용이 같은 블록을 검색하는 방법이다. 파일의 내용 대신 각 블록의 체크썸을 비교하기 때문에 파일 전체 데이터 비교의 오버헤드를 줄일 수 있다. 체크썸을 분석하는 블록의 사이즈는 Rsync 수행 시 가변적으로 변경하여 처리가 가능하다.

III. The Proposed Scheme

함정 전투체계의 생존성 향상을 위해 특정 파일서버를 사용하지 않는 저장환경에서 전술데이터 파일의 실시간 동기화, 이력정보 관리를 통한 오류 발생 시 시점 복원기능을 구현하기 위해 아래와 같이 레코드 블록 데이터 파일 관리시스템을 설계하였다.

1. Design Specifications

레코드 블록 데이터 파일 관리시스템에 필요한 요소는 다음과 같다.

- 레코드 블록 데이터 쓰기, 읽기 방식 설계
- 레코드 블록 단위의 실시간 파일동기화 설계
- 저장된 레코드 블록 단위의 시점 복구 기능 설계

설계된 파일시스템을 사용자에게 제공하기 위해 Table 2의 API를 제공한다.

Table 2. File Management API

API	Description
RBDS_Init()	Initialize record blocks
RBDS_Write()	Update data in a set record block
RBDS_Read()	Load data from the most recent record block
RBDS_FileSync()	Synchronize files per record block
RBDS_Restore()	Create a new record block based on the selected record

레코드 블록 데이터 파일관리 시스템의 구조도는 아래 Fig. 1과 같다. 활성화(Active) 상태의 정보처리장치에 동기화 대상 저장소인 비활성(Standby) 상태의 정보처리장치의 저장소가 네트워크 마운트 상태로 연결된 구조이다. 활성화 상태의 정보처리장치의 어플리케이션에서는 RBDS를 통해 파일을 쓰고/읽는 경우 레코드 블록 단위 파일 생성 및 갱신이 발생하며 파일이 변경되는 경우 파일동기화가 함께 진행된다. 정보처리장치 외부의 백업 관리 장치에서 RBDS를 통해 레코드 블록 단위로 저장된 파일을 특정 레코드 블록 데이터로 복구를 할 수 있다.

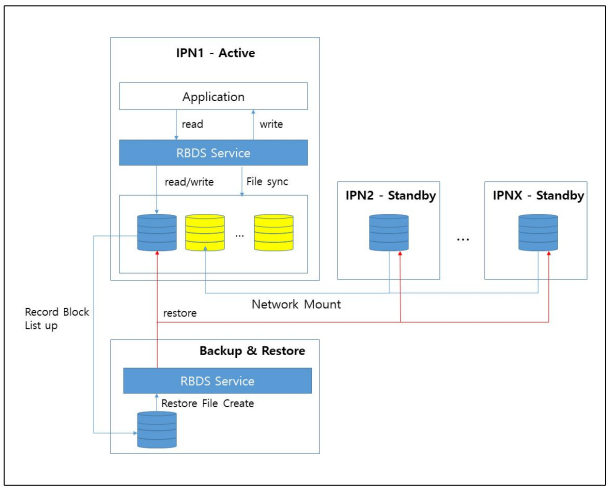

Fig. 1. Structure Chart of RBDS

2. Write/Read Record Block Files Process

레코드 블록 단위의 데이터 관리를 위해 파일시스템은 아래와 같은 블록 단위로 저장하게 된다.

<데이터 헤더, 전술 데이터>

데이터 헤더는 데이터 생성 시점 초기화를 구분할 수 있는 데이터 블록 초기화 시간정보와 데이터 사이즈로 구성되며 설정한 데이터 사이즈만큼의 영역에 전술데이터를 저장하게 된다.

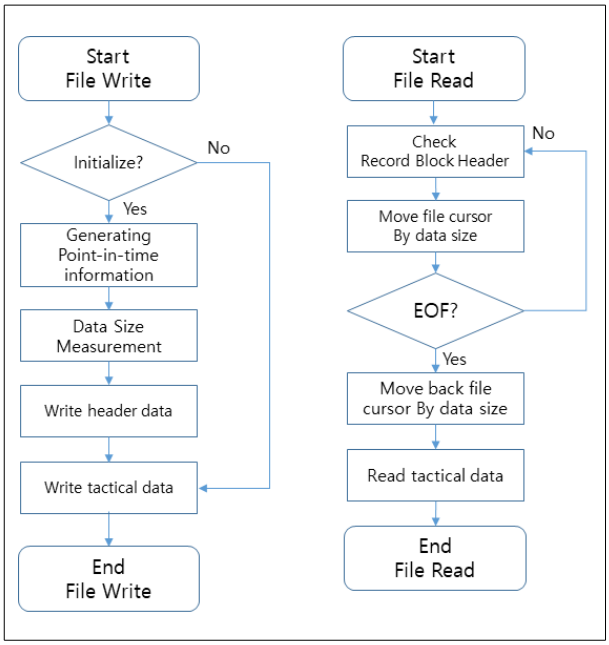

Fig. 2. Record Block Data Write/Read Process

파일의 레코드 블록 생성 및 업데이트는 Fig. 2와 같이 진행된다. 레코드 블록 시작 초기화를 호출하게 되면 레코드 블록의 시점 데이터 및 블록 사이즈를 지정하게 된다. 생성한 데이터 헤더와 함께 전술데이터를 함께 파일에 바이너리 타입으로 저장하게 된다. 레코드 블록 초기화 사용 이후에 전술데이터가 업데이트 되는 경우 레코드 데이터 헤더는 변경하지 않고 전술데이터만 업데이트하게 된다.

여러 시점의 레코드 블록이 생성되면 데이터 파일 내부는 데이터 헤더와 전술데이터로 결합된 레코드 블록 단위로 전술데이터 이력이 존재하게 된다.

저장된 레코드 블록 파일을 이후에 로드하는 경우는 Fig. 2와 같이 진행된다. 블록의 개수가 가변적이므로 가장 최신의 레코드 블록을 탐색하는 과정을 거치게된다. 각 블록의 헤더정보를 확인하고 헤더에 저장된 데이터 크기만큼 Seek() 메소드를 활용하여 파일 커서를 이동한다. 이동한 파일의 커서가 파일의 종료지점인 경우 마지막 읽은 레코드 블록이 최신 블록으로 판단하게 된다. 파일의 끝에 있는 커서를 다시 데이터 사이즈만큼 뒤로 이동하여 데이터를 읽어 사용자에게 데이터를 반환한다.

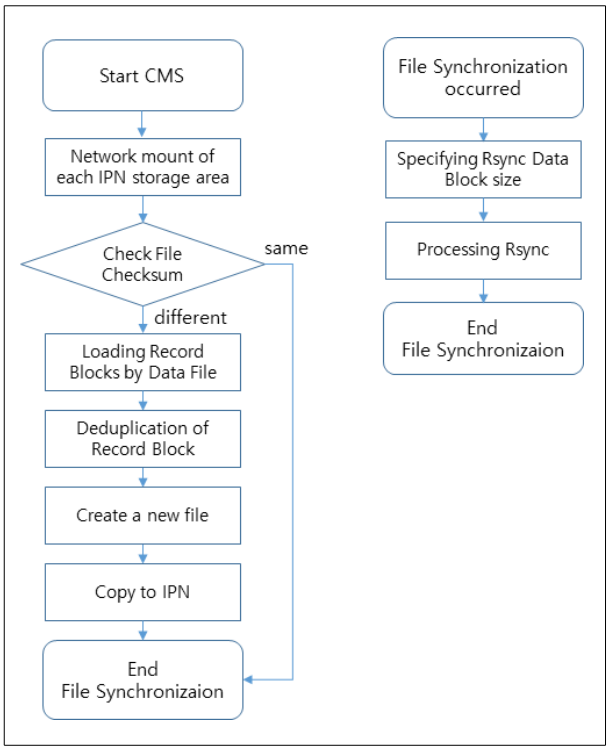

Fig. 3. File Synchronization Process

3. File Synchronization

여러 정보처리장치에 저장되는 데이터 파일의 실시간 동기화를 보장하기 위해 파일동기화 절차를 설계하였다.

파일동기화 절차는 파일동기화 초기화, 파일 업데이트 실시간 동기화 두 가지 케이스로 분류되어 진행한다. 함정 전투체계의 저장영역은 다중화로 관리되고 있으며 장비의 가용여부에 따라 최초 실행 시 파일의 저장 된 레코드 블록이 다른 경우가 발생할 수 있다.

Fig. 3은 데이터 파일 동기화에 대한 절차이다. 데이터 파일 동기화는 함정 전투체계 시작과 함정 전투체계 운용 중 파일갱신이 발생하는 케이스 두 가지로 구분할 수 있다.

유지보수 등으로 전체 장비가 가용되지 못해 저장된 데이터 파일 간 차이가 발생할 수 있으므로 함정 전투체계 시작 시 가용된 정보처리장치의 레코드 블록을 모두 로드하여 중복 제거[7-9]하는 절차를 수행한다. 레코드 블록 중복 제거는 헤더에 저장된 레코드블록 저장 시점 기준으로 분석하여 제거한다. 중복이 제거된 레코드 블록들을 새로운 파일에 저장한 후 다시 전체 정보처리장치에 복사하여 저장하는 방식으로 최초 부팅 시 파일동기화 절차를 종료한다.

레코드 블록 수집, 중복제거, 파일복사 등 오버헤드가 발생하지만 초기 파일 동기화를 수행하며 레코드 블록 단위로 파일 일치를 확인하므로 초기화가 완료된 이후에는 업데이트 된 데이터 블록만 동기화 할 수 있다.

전체 레코드 블록의 동기화가 이루어 진 이후 함정 전투체계를 운용하며 데이터 변경이 발생하는 경우 가장 최신의 데이터 블록이 추가되거나 수정된 경우이므로 변경된 레코드 데이터 블록만큼의 데이터 동기화만 진행되면 된다.

Rsync()는 데이터 블록 사이즈를 지정 후 블록 사이즈 만큼의 체크썸을 비교하여 실제로 변경된 데이터 블록 사이즈만큼 데이터 전송이 가능하므로 동기화 시간 및 동기화로 인한 네트워크 사용량을 최소화 할 수 있다.

함정 전투체계 초기 파일동기화 이후 데이터 생성 시 데이터 블록이 생성되며 생성된 데이터 블록의 데이터 파트가 변경되며 업데이트가 진행된다. 변경이 발생할 때마다 대기상태의 정보처리장치 저장소의 파일이 업데이트 되어야 하며 Fig. 3의 파일 갱신 발생 처리와 같이 지정된 데이터 블록의 사이즈만큼 수정된 데이터 파일과 나머지 데이터 파일을 Rsync()를 통해 데이터 동기화를 진행한다.

4. Restoring Point-in-time Files

데이터 파일의 시점 복원을 위한 데이터 파일 생성은 Fig. 4와 같이 진행한다. 먼저 전투체계에서 사용하는 백업프로그램을 활용하여 목록을 백업하고 복원 시 파일에 저장되어 있는 레코드 블록 리스트를 운용자가 선택한다. 복구를 위해 선택한 레코드 블록의 데이터를 로드하여 가장 최신 데이터 블록으로 관리할 수 있도록 새로운 헤더를 생성하고 기존파일의 마지막에 새로운 데이터 레코드 블록을 저장한다[10].

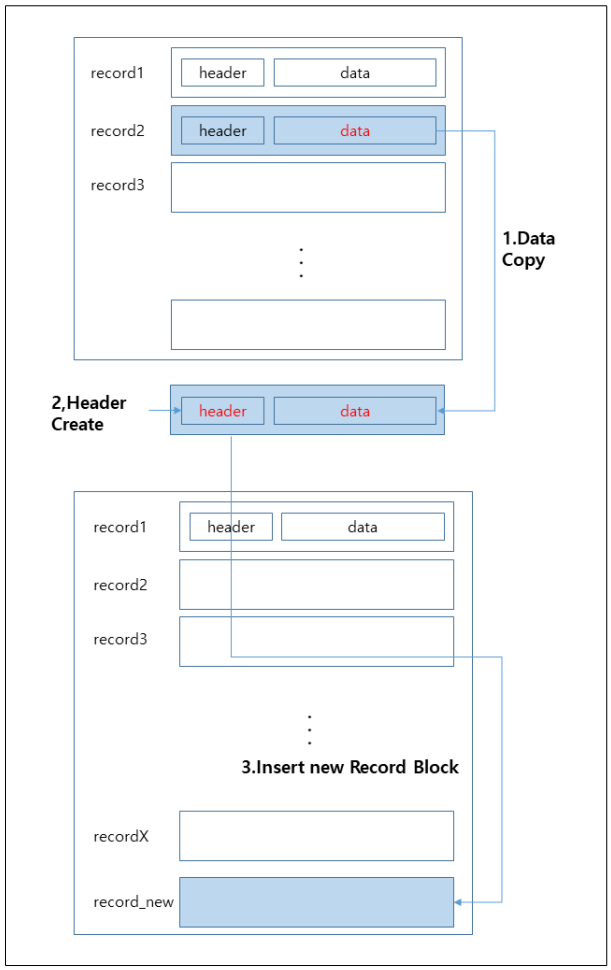

Fig. 4. Create restore Files

새로 생성된 레코드 블록이 저장된 데이터 파일을 백업 프로그램을 통해 전체 정보처리장치에 복사하여 시점 복원[11]을 완료한다.

5. Expectation Effectiveness

레코드 블록 단위로 데이터를 관리하여 시점복원이 가능하도록 RDBS를 설계하였고, 기존 파일시스템 대비 파일사이즈 및 동기화 단위, 시점복원 여부에 대한 비교는 아래 Table 3과 같다.

Table 3. Compare to existing file system

	Exist File System	Proposed File System
File size(Avg.)	10KB	110KB
File Synchronization unit	File	Record Block
Restoring point-in-time data	X	O

레코드블록 단위로 데이터파일을 관리하기 때문에 레코드블록을 10개 제한하여 관리하는 경우 헤더 사이즈와 합쳐 10배 정도의 파일사이즈가 필요하며 파일동기화 단위는 기존 파일시스템의 파일전체 복사를 통한 파일단위 동기화에서 레코드 블록 단위로 비교하여 파일동기화를 진행하게 된다. 또한, 레코드블록 단위로 관리 된 데이터파일을 사용하여 시점 복원이 가능하다.

IV. Test and Evaluation

본 논문에서 설계한 레코드 블록 데이터 파일 관리시스템(RBDS)의 읽기/쓰기, 파일동기화 성능을 분석하여 함정 전투체계에 실시간성을 보장하는 수준인지 평가하였으며 레코드 블록 단위로 시점 복원 기능을 함정 전투체계 백업 프로그램을 통해 확인하였다.

읽기/쓰기, 파일동기화 성능 비교를 위해 함정 전투체계의 구역경고관리 SW에 파일시스템을 적용하여 시뮬레이터로 활용하였다. 구역경고관리 SW에 파일 쓰기 기능을 RDBS로 대체하여 사용하여 레코들블록 단위로 데이터를 저장하도록 수정하였다. 또한, 레코드 블록 단위로 시점 복원 기능을 확인하기 위해 기존 함정 전투체계 백업 프로그램을 시뮬레이터로 활용했으며, 백업 프로그램에 RDBS를 적용하여 선택한 시점으로 최신 레코드블록을 생성할 수 있도록 수정하여 시점복원 시험을 진행하였다.

테스트 환경은 Fig. 5와 같이 환경을 구성하여 시험을 진해하였다. 전시처리장치 PC, 정보처리장치PC, 시뮬레이터 PC를 랜 스위치를 통해 연결하였으며 정보처리장치는 2대로 구성하여 정보처리장치 PC1, 2에 저장되는 데이터 파일 동기화를 확인하였다.

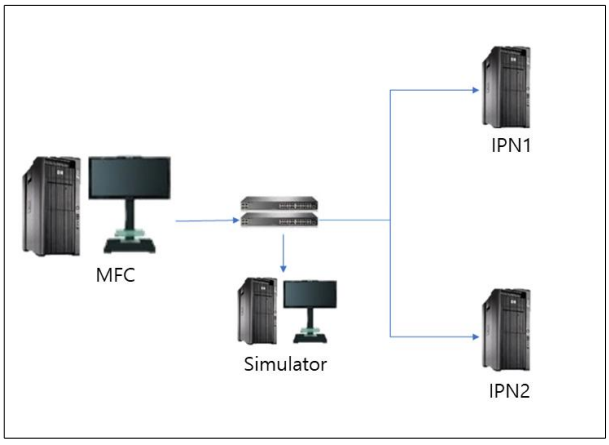

Fig. 5. Test Environment

MFC(전시처리장치PC)에는 운용자화면을 위한 전투체계 SW를 실행하며 IPN(정보처리장치PC)에서는 운용자입력에 대한 처리를 위한 전투체계 SW를 실행하였다. 시점복원을 위한 백업 및 복원 프로그램은 Simulator(시뮬레이터PC)를 통해 실행하였다. 환경 구성의 세부정보는 Table 4와 같다.

Table 4. Performance of Test Environment

	CPU	Mem	Disk	OS
MFC	Intel Core i7-4770 @3.40GHz	16GB	512GB	Windows7
IPN	Intel Core i5-6400 @2.70GHz	16GB	512GB	RTST Linux
Simulator	Intel Core i5-6400 @2.70GHz	16GB	512GB	Windows7

1. Compare Performance

1.1 Compare Time of File Read/Write

함정 전투체계에서 사용하는 전술 데이터 파일 중 구역경고 관리의 데이터 업데이트 사이즈 10 KB를 기준으로 데이터 읽기/쓰기 성능을 평가하였다. 평가 방식은 기존 파일쓰기 방식과 제안한 파일시스템에서 10 KB의 데이터를 읽기/쓰기를 10회 반복 후 평균 시간으로 성능비교를 진행하였다.

Table 5. Compare Time of File Read/Write

	Exist File System	Proposed File System
File Read Time	1.5 ms	1.7 ms
File write Time	8.5 ms	11.5 ms

Table 5의 결과를 통해 기존 파일시스템 대비하여 읽기시간은 13%, 쓰기시간은 35% 수준으로 증가함을 확인하였다. 읽기 속도의 증가는 가장 최신 레코드 블록을 찾기 위해 발생하지만 0.2ms 수행시간 증가는 미미한 수준이며 전투체계 초기화 시에 파일 읽기가 수행되기 때문에 초기화 시간 일부 증가는 운용상 문제가 없을 것으로 판단된다. 쓰기 속도의 시간은 헤더 생성 및 파일 내 데이터 업데이트 위치 파악을 위한 오버헤드가 발생하여 3ms 정도 추가되었다. 데이터파일의 쓰기는 운용자가 정보를 생성, 수정, 삭제하는 경우 발생하게 되며 테스트에서 확인한 파일쓰기의 3ms의 추가시간은 한 번의 운용자가 의도한 데이터 수정이 처리되는 시간이 추가되는 시간이며 이는 운용자가 수정 후 기능사용에 문제가 되지 않는 오버헤드로 판단된다.

1.2. Compare Time of File Synchronization

파일동기화는 두 가지 케이스로 분류하여 분석을 진행하였다. 최초 실행 시 전체 블록에 대해 분석 후 동기화 진행과 실시간으로 데이터 수정이 발생하는 경우로 구분하여 측정한 데이터를 평가했다.

최초 실행 시 불일치 블록 증가에 따른 동기화 시간을 측정했다. 두 개의 파일은 각 10개의 레코드 블록을 가지고 있으며 레코드 블록의 불일치를 1개에서 10개까지 증가시키며 동기화 시간을 분석했다.

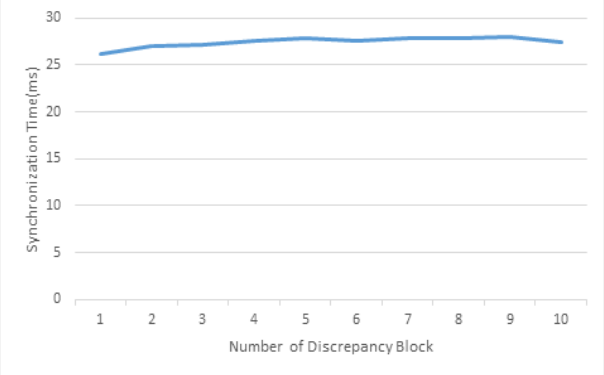

Fig. 6. File Synchronization Time

Fig. 6은 레코드 블록 불일치 수에 따른 파일동기화 시간을 측정한 결과이다. 해당 시험 결과를 통해 레코드 블록의 개수가 차이가 나는 것에 있어서 파일동기화 처리 속도의 차이는 크지 않음을 알 수 있었다. 전체 블록을 수집하고 수집된 항목의 중복제거를 위해 전체 레코드 블록을 확인하기 때문에 처리 속도는 블록의 차이가 나더라도 비슷한 것으로 확인할 수 있었고, 결과물의 레코드 블록차이로 데이터를 새로 쓰는 양이 레코드블록 개수의 차이정도인 것은 최종 처리결과에 큰 영향이 없는 것으로 확인하였다.

Table 6. Compare of File Synchronization

	Exist File System copy()	Proposed File System Rsync()
Time of File Synchronization	31 ms	55 ms
Transfer Data	100 KB	100 KB

두 번째 케이스인 실시간 동기화의 경우 레코드 블록이 변경되었을 때 두 개의 파일이 동기화되는 시간을 측정한 결과는 Table 6과 같다.

기존 파일시스템은 전체 파일 복사가 발생하므로 100KB의 데이터 전송이 발생하며 시간은 33ms로 측정되었다. RBDS의 파일동기화 방식을 사용한 경우 Rsync의 델타인코딩 방식을 통해 변경된 100KB의 전송이 동일하게 발생함을 확인할 수 있었으며 55ms의 처리시간이 측정되었다. 기존 33ms대비 55ms시간 소모는 블록사이즈 별 체크썸 확인을 통해 변경항목을 검색하는데 소요되는 시간으로 분석된다. 파일 읽기/쓰기와 마찬가지로 전투체계 정상운용이 가능한 범위 내의 동기화시간이 발생하는 것으로 확인된다.

1.3 Test SW Applied RBDS

파일 일기/쓰기 성능 및 파일동기화 시간 분석을 통해 레코드 블록 관리를 위한 오버헤드가 발생함을 확인하였다. 해당 오버헤드로 인한 함정 전투체계 SW의 실시간성을 보장하며 정상 동작하는지 확인하기 위해 전술 데이터 파일을 사용하는 구역경고관리 SW에 적용하여 기능시험을 진행하였다. 기능시험을 개발 문서의 시험평가 절차서에 따라 절차를 진행했으며 구역경고관리 SW가 정상 동작함을 확인하였다.

2. Test restoring point-in-time data files

레코드 블록을 저장한 데이터를 시점에 따라 복원하는 기능을 확인하기 위해 전투체계 전술데이터 파일 사용 기능 중 하나인 구역경고관리의 전술데이터 파일을 사용하여 시점 복원 기능을 확인하였다.

기존 전투체계의 백업 프로그램에 RBDS를 적용하여 시뮬레이터 PC에서 시점 복원 기능을 확인하였다. 기능 확인을 위해 전투체계의 구역경고관리의 데이터 파일을 사용했다. 3가지 시점의 레코드를 파일에 저장한 후 백업프로그램에서 시점을 선택 각 시점으로 복구하여 정상적으로 데이터 시점 복원이 되는지 확인하였다.

Fig. 7. List of record block in file

Fig. 7은 백업 프로그램에서 백업 후 시점 복원을 위한 레코드 블록 리스트 전시화면이다. Record1, 2, 3 항목을 하나씩 선택 후 복원 기능을 활용하여 데이터파일을 정보처리장치에 저장하였다. 원하는 시점으로 정상적으로 복구되었는지 확인하기 위해 각 시점 복원 후 전투체계를 재실행하여 정상적으로 데이터가 복원되었는지 확인하였다.

Fig. 8. Result of restoring data file

운용자가 선택한 각 레코드 블록의 데이터가 정상적으로 복원되었음을 구역경고관리 전시화면을 통해 확인하였다. 각 레코드 블록에 저장된 데이터가 Fig. 8의 record1, 2, 3과 같이 정상 복원되었으며 재부팅으로 데이터파일이 정상적으로 사용할 수 있음을 함께 확인하였다.

V. Conclusions

함정 전투체계의 데이터 파일의 유효성은 전투체계를 정상운용하기 위해 가장 중요한 요소이다. 데이터 파일의 수정이 발생하는 경우 이전 데이터로 복구가 불가능한 기존 전투체계 데이터 관리 시스템을 개선하기 위해 레코드 블록 단위의 파일관리 시스템을 제안하였다. 제안된 파일관리 시스템을 구현 및 테스트를 통해 저장된 데이터로 함정 전투체계의 기능이 정상 동작을 확인하였으며 데이터 파일 동기화, 특정 시점 복구시험을 통해 데이터 파일 관리의 유효성 향상을 확인하였다.

파일 읽기 처리시간은 기존 대비 13%, 쓰기 처리시간은 35% 증가를 나타냈으며 함정 전투체계 정상동작에 영향을 주는 정도의 추가 시간이 발생하지는 않는 것으로 확인하였다.

본 논문에서 제안하는 레코드 블록 데이터 저장 방식은 파일에 저장된 데이터를 DBMS의 이력 관리방식처럼 사용하고자 하는데 목적이 있었다. 이력관리로 발생하는 파일 사이즈 증가로 실시간 파일동기화 오버헤드 증가를 고려하여 Rsync를 적용한 알고리즘을 제안했으며, 기존 파일

동기화 대비 네트워크 사용량 및 동기화시간이 동일한 수준으로 분석되었다. 마지막으로 기존 백업프로그램에서 저장한 레코드 블록을 활용하여 시점 복원기능을 제공하여 파일 불일치로 발생하는 데이터 변경을 복구 할 수 있음을 확인하였다.

추후 연구과제로 파일의 오류로 파일전체를 사용하지 못하는 상황을 대비하여 레코드 블록을 파일단위로 분리하고 블록을 관리하는 메타파일[12]을 이용하는 방식으로 전투체계 파일관리 시스템 연구를 진행할 예정이다.

REFERENCES

[1] Shin Hun Yong, Kim Joo Yong, "Research of OSD Standardization in Naval Combat System" The Korean Institute of Electrical Engineers, pp. 354-355, Oct 2012.

[2] Juwon Lee, “Development of Message Define & Management System based on Distributed Processing Environment for Naval Combat Systems”, KIISE Transactions on Computing Practices, Vol.23, No.12, pp.670-676, Dec 2017

[3] C. J. Date, “An Introduction to Database Systems”, 7th addition-Wesley, 2000

[4] Ho Min Jung, Young Woong Ko, Jae Min park, Jin Sang Kong, “A File Synchronization System using Similarity-based Deduplication”, Deduplication. Journal of KIISE : Computing Practices and Letters, pp 548-552, Jul 2012

[5] Young-Jun Yoo, Young Woong Ko, “Design and Implementation of File Synchronization System using Data Deduplication Meta Data File”, Korea Computer Congress 2014, pp 1551-1553, Jun 2014

[6] Mingyu Lim. (2022). “A File Synchronization Framework Based on RSync Protocal for Cloud Storage Services”, The transactions of The korean Institute of Electrical Engineers, pp 1164-1175, Jul 2022

[7] Younng-Jun Yoo, Sun-Jeong Kim, Young Woong Ko, “Cloud File Synchronization Scheme using Bidirectional Data Deduplication”, The Journal of Korean Institute of Infomatioin Techhnology, pp. 103-110, Jan 2014

[8] In-Cheol Hwang, Oh-Young Kwon, “Design of Adaptive Deduplication Algorithm Based on File Type and Size“ Journal of the Korea Institute of Infomation and Communication Engineering, pp 149-157, Feb 2020

[9] Sung-ouk Jung, Hoon Choi. “Performance Analyssis of Open Source Based Distributed Deduplication File System”, KIISE Transactions on Computing practices, pp 623-631, Dec 2014

[10] Dong-Kweon Hong, “Implementations of Record_Level Sysnchronized Safe personal Cloud”, Journal of Korean Institute of Intelligent Systems, pp 239-244, Dec 2012

[11] Kuk-Heon Lee, Doo-Won Jeong, Cheul-Hoon Kang, Sang-Jin lee. (2014). “The Method of Recovery for Delete Record in the DB2 Databse”, Journal of Digital Forensics, pp 1-14, May 2014

[12] Jin-Hyung Kim, Hae-Woong park, Han-Sol Ahn, Young-Wook Jo, Jon-Woo Jung, “A Study on Priority Back-Up System with File Metadata”, Rrocessing of KIIT Conference, pp 739-742, Nov 2021

Authors

Seok-Hyun Kang received the B.S. degrees in Infomation and Communication Engineering from Yeungnam University, Korea, in 2014. He is currently working in Hanwha systems Co. from 2014. He is interested in Naval Combat System, File System and Data Backup.

Keun-Hee Kim received the B.S. degrees in Computer Engineering from Yeungnam University, Korea, in 2014. She is currently working in Hanwha systems Co. from 2016. She is interested in Combat Management System, File System, Naval Support Software and Design Pattern.