THE JOURNAL OF KOREAN INSTITUTE OF ELECTROMAGNETIC ENGINEERING AND SCIENCE. 2025 August; 36(8), 814～817.

http://dx.doi.org/10.5515/KJKIEES.2025.36.8.814
ISSN 1226-3133 (Print) · ISSN 2288-226X (Online)

다기능레이다 데이터 분석 소프트웨어의 속도 향상 방안

Enhancing the Computational Efficiency of Multi-Function Radar Data Analysis Software

권 민 정[1]

Min-Jeong Kwon[1]

요 약

본 논문에서는 다기능레이다 데이터 분석 소프트웨어(SW)의 분석 절차에 대해 설명하고 각 분석 단계별로 분석 속도를 개선하기 위한 방안을 제안한다. 제안하는 방법 적용 시 레이다 시험 데이터의 크기가 커질수록 속도 향상에 효과적임을 확인하였으며, 기존 데이터 분석 SW의 분석 속도 대비 평균 약 4.14배 분석 속도가 향상되었음을 확인하였다.

Abstract

Herein, we outline analysis procedure for multifunctional radar data analysis software and propose a method to enhance the processing speed at each analysis step. The proposed method was effective as the size of the radar test data increased. Specifically, the average analysis speed improved by approximately 4.14 times compared with that of the existing data analysis SW.

Key words: Multi-Function Radar, Data Acquisition & Analysis System (DAAS), Memory Mapped File, Key-Value Storage, Ordered-Map

Ⅰ. 서 론

최근 다기능 레이다의 기능이 점점 더 다양해짐에 따라, 각 기능별 성능을 검증하기 위한 자체 분석 도구를 개발하는 선행 연구들이 진행되고 있다[1]. 그러나 이러한 연구들은 정성적 성능 평가에 치중되어 있으며, 데이터 증가에 따른 처리 문제는 충분히 다루지 않았다.

본 논문은 대용량 데이터 환경에서 분산 처리와 인메모리 기반 처리 기술을 적용하여 데이터 분석 속도를 향상시킨 기존 연구[2],[3]를 바탕으로 실제 다기능 레이다 분석 SW의 성능을 정량적으로 최적화할 수 있는 구체적인 방안을 제시한다는 점에서 기존 연구와 차별성을 갖는다.

당사에서 개발한 다기능레이다 분석 SW의 운용 하드웨어 사양 및 개발 환경은 표 1과 같다.

다기능레이다 분석 SW는 그림 1과 같은 순서로 데이터를 분석 및 전시한다.

참조 파일은 binary 형태인 다기능레이다 시험 데이터를 해석하기 위해 패킷별 데이터 구조를 JSON 파일 형태로 나타낸 파일이며, daas reference file이라는 의미로 그림 1에 .drf 확장자를 가진 형태로 나타냈다. 참조 파일 분석을 위해 C++ 11 이상에서 사용 가능하고 읽기/쓰기를 지원하는 경량 JSON 파서인 rapidjson을 사용하였다. 참

한화시스템(Hanwha Systems)
1: 선임연구원(https://orcid.org/0009-0000-2432-0568)
· Manuscript received May 2, 2025 ; Revised May 9, 2025 ; Accepted June 8, 2025. (ID No. 20250502-046)
· Corresponding Author: Min-Jeong Kwon (e-mail: minjeong.kwon93@hanwha.com)

표 1. 운용 하드웨어 사양 및 개발 환경
Table 1. Operating hardware specifications and development environment.

CPU	12th Gen Intel(R) Core(TM) i7-12700H
Memory	DDR4 24 GB
Disk	512 GB SSD
OS	Windows 11
IDE	Visual Studio 2022
Language	C++ 17
Compiler	19.40.33811
Framework	MFC, Chartdir, Codejock toolkit

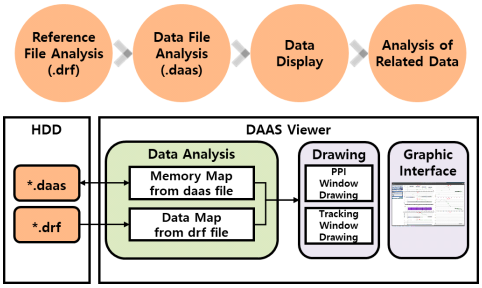

그림 1. 데이터 분석 & 전시 방법
Fig. 1. Data analysis & display method.

조 파일에 대한 분석은 다기능레이다 데이터 분석 SW 실행 시 1회 수행되며, 데이터 파일을 변경하여도 패킷별 데이터 구조가 변경되지 않는 이상 참조 파일을 따로 재분석하지 않아도 된다.

데이터 파일은 다기능레이다 시험 데이터를 의미하며 binary 형태의 파일로 저장되며 data acquisition & analysis system에서 사용하는 파일이라는 의미로 그림 1에 .daas확장자를 가진 형태로 나타냈다.

다기능레이다 시험 데이터의 분석 및 전시 시 전체 속도에 영향을 미치는 요인을 측정하였을 때, 분석 및 전시 완료까지 소모되는 시간은 데이터 파일의 크기에 따라 크게 달라진다. 500 MB를 기준으로 참조 파일 분석에서부터 데이터 전시까지 95 %의 분석 시간을 차지하는 요인이 저장된 참조 및 데이터 파일을 읽어와 차트에 전시할 요소의 맵을 생성하는데 소모된다. 이외 5 % 정도는 차트 구성요소를 시각적으로 표현하기 위해 화면에 그리는 렌더링 작업을 수행하는 데 소모되었다. 그러므로 본 논문에서는 많은 시간이 소모되는 참조 및 데이터 파일을 분석하여 차트에 전시할 자료구조를 생성하는 과정에서의 속도를 개선하는 방법에 대해 논의한다.

Ⅱ. 본 론

2-1 참조 파일 분석 방법 변경

JSON 파일 형태의 참조 파일을 분석하기 위해 기존 rapidjson에 비해 SIMD(single instruction multiple data) 명령어를 활용하여 병렬로 데이터를 처리하는데 특화되어 있고 효율적인 메모리 접근과 분기에 특화된 개선 등으로 속도를 높인 최신 JSON 파서인 simdjson을 활용하였다. 이 파서는 많은 양의 데이터를 반복적으로 파싱 해야 하는 경우 큰 효율을 보인다[4]. 그러므로 다기능레이다 시험 데이터의 크기가 커짐에 따라 대용량 파싱에 더 적합한 simdjson을 활용하는 것으로 변경하였다.

해당 파서를 이용하여 500~4,000 MB 용량의 시험 데이터 파일 8개를 캐시드 I/O(cached I/O) 효과를 안정화하기 위해 각각 13회씩 파싱하였다. 초기 3회는 워밍업(run-in) 구간으로 간주하여 제외하고, 이후 10회 실행 시간을 평균 내었을 때, 기존 rapidjson 대비 평균 약 64.64배 속도가 향상되었음을 확인하였다.

또한 참조 파일을 데이터 파일 내 포함시킴으로써 기존 참조 파일과 데이터 파일이 분리되어 있음으로 인한 버전 관리 불편을 해소하였다.

다기능레이다 시험 데이터 기반 메모리 맵을 통해 데이터의 주솟값을 값 형태로 변환하는 것을 효율적으로 처리하기 위해 참조 파일 기반 요소 맵을 형성하였다.

참조 파일 기반 요소 맵이란 저장된 데이터의 구조를 각 요소 별 크기와 오프셋의 크기를 가진 맵 형태로 정의하여 이를 이용하여 특정 데이터 요소의 값을 얻기 위해 주솟값을 계산하는 반복 연산을 줄였다.

2-2 시험 데이터 파일 분석 방법 변경

기존 레이다 시험 데이터 파일 분석 시 메모리 맵 파일 형태로 가상 메모리 공간에 파일을 매핑하고 그 주솟값을 맵 형태로 가지는 자료구조를 활용하였다. 이러한 자료구조는 실제 데이터에 접근 시 물리 메모리에 올리는

형태로 메모리를 효율적으로 사용하고 페이지 폴트가 발생하는 순간만 디스크에서 데이터를 읽어와 불필요한 디스크 I/O를 줄일 수 있다는 장점이 있다.

그러나 대부분의 시험 데이터는 시간순 분석이 필요하므로 모든 시간에 대한 데이터 검색이 요구된다. 이러한 과정은 대용량 데이터의 경우, 빈번한 페이지 폴트를 발생하게 하여 불필요한 디스크 I/O로 성능 저하의 요인이 된다. 또한 기존 메모리 맵을 이용하여 특정 주솟값의 데이터를 값 형태로 얻기 위해 소모되는 시간을 측정한 결과, 송신 빔 데이터 중 방위각 108,397개를 값 형태로 읽어오는 작업을 10회 수행 시 평균 약 2 ms가 소모되었다는 점을 감안하면 전시 항목 또는 데이터 용량이 늘면 분석 시간이 증가할 것을 예상할 수 있다.

본 논문에서는 이러한 메모리 맵 구조의 단점을 최소화하기 위하여 전시를 위해 사용하는 데이터의 종류를 미리 식별하여 구조체 형태로 선언하였다. 그 외 전시하지 않는 데이터는 참조 파일 및 데이터 파일 기반 메모리맵을 이용하여 기존과 같은 방식으로 값을 얻는다.

전시를 위한 데이터는 레이다 시험 데이터 파일에 최초 접근 시 구조체로 값을 복사한 후, 시간 정보를 키로 하는 STL의 정렬된 맵 컨테이너에 벡터 형태로 저장하였다. 인메모리 방식의 자료구조는 기존 메모리 맵 형태를 활용하는 것보다 빠른 속도로 데이터를 처리하는데 유리하다[3].

STL 컨테이너 중 끝에 데이터를 삽입하는 속도는 데이터의 정렬로 인해 상대적으로 느리지만, 레이다 시험 데이터 분석 SW처럼 시계열 데이터를 처리하고 분석하는 시스템에서는 정렬된 키-값 구조를 활용하여 시간 기반으로 데이터를 관리하고 조회하는 것이 효율적이다[3].

그림 2는 참조 파일이 포함된 구조의 레이다 시험 데이터 파일의 변경된 분석 방법에 대해 설명하는 그림으로 앞서 설명한 전시를 위한 데이터의 정렬된 맵과 전시하지 않는 데이터의 메모리 맵으로 나눠 데이터를 관리함을 확인할 수 있다.

제안하는 방법으로는 전시 데이터의 경우 불필요한 주소 연산 없이 값 형태의 데이터를 얻을 수 있으며 인메모리 구조에서 빠르게 값 형태의 데이터 접근이 가능하여 데이터 용량이 클수록 이러한 방법은 큰 이점이 있음을 확인할 수 있다.

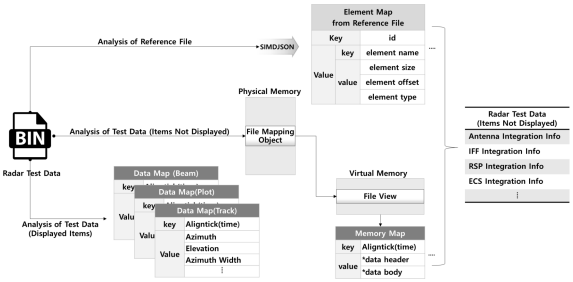

그림 2. 레이다 시험 데이터 파일 분석 방법 변경
Fig. 2. Changes to radar test data file analysis method.

2-3 연관된 데이터 분석 방법 변경

전시된 데이터를 선택하여 분석을 요하는 시점의 연관된 데이터를 통합하여 분석하는 것 또한 레이다 시험 데이터 분석 시 필요한 기능이다[1].

연관된 데이터를 찾기 위해선 aligntick이라는 시간 개념이 필요하다. aligntick이란 기존 데이터가 저장된 시간 정보와는 달리 연관된 데이터를 찾기 위해서 동시에 봐야 하는 시간 정보로서 aligntick과 데이터 저장 시간은 다를 수 있다. 연관 데이터를 찾는데 용이하기 위해 데이터 패킷 헤더에 aligntick을 추가하여 변경하는 자료구조의 키로 활용하였다.

더불어 송신 빔, 탐색 정보, 추적 정보, 모의 표적 정보 등을 전시할 때 그림 3처럼 각각 다른 전시 레이어(layer)를 이용하여 데이터를 보여줄 수 있도록 변경하였다. 이때 레이어 이름은 전시 항목명으로 설정하였다. 이렇게 전시 레이어 별로 항목별 데이터 세트를 넣고 전시함으로써 특정 심볼을 선택했을 때 어떤 항목의 데이터를 선택했는지 빠른 식별이 가능하다.

이러한 구조는 기존 단일 레이어를 이용하여 전시하는 방법 대비 사용자 이벤트 처리(마우스 클릭 시 어떤 레이

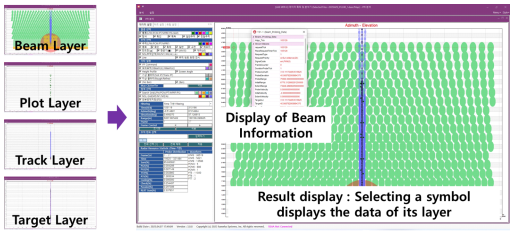

그림 3. 데이터 종류별로 layer 구분
Fig. 3. Layer classification by data type.

어의 데이터를 선택했는지 판별하는 로직)가 추가됨에 따라 코드 라인 수가 기존 대비 약 2 % 증가했지만, 이러한 로직으로 인해 기존 전시 항목별로 사용자가 선택한 데이터와 같은지 비교하는 구문이 제거되며 실제 실행 코드 라인 수는 약 23 % 감소시키는 효과가 있다.

2-4 분석 시간 비교

이처럼 속도 개선 방안을 적용한 데이터 분석 SW로 항공기 5대를 반복 모의하여 저장한 추적 시험 데이터를 병합하여 500~4,000 MB 크기를 갖는 파일 8개를 만들었다. 각 파일 별로 13회씩 분석을 수행하며 초기 3회는 파일 시스템 캐시 안정화 단계로 간주해 제외하고 이후 10회의 각 항목별 평균 분석 시간을 비교하여 그림 4와 같이 나타내었다.

참조 파일 분석의 경우 저장된 데이터의 구조가 변경되지 않아 데이터 파일 크기에 크게 영향을 받지 않았으며, simdjson 파서 적용 시 약 64.64배 분석 속도가 향상되었다. 데이터 파일 분석 시 약 1.58배 속도가 향상되었으며, 데이터 파일의 크기가 증가할수록 데이터 파일 분석 시간이 증가함을 보였다. 전시된 데이터를 선택하여 동일한 시간의 연관 데이터를 분석하는 경우 약 5.51배의 속도가 향상되었으며, 제안하는 방법은 정렬된 맵 컨테이너로 구현되어 있어 탐색시간은 데이터의 크기에 따라 로그 형태로 증가하여 비교적 효율적임을 확인하였다.

결론적으로 전체적인 데이터 분석 속도는 표 2의 각 항목 별 평균 분석 시간을 합하여 비교하였을 때, 평균 약 4.14배 향상되었다. 그러나 해당 기법은 인메모리 자료구조 형태의 정렬된 맵을 사용하기 때문에 메모리 사용량

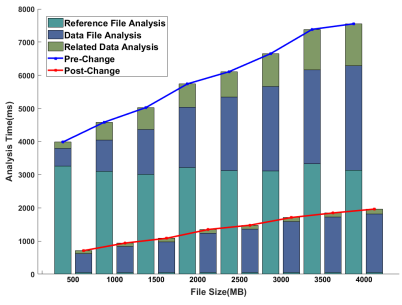

그림 4. 파일 크기 별 데이터 분석 시간 비교
Fig. 4. Comparison of data analysis time by file size.

표 2. 각 항목 별 평균 분석 시간 및 메모리 사용량 비교
Table 2. Comparison of average analysis time and memory usage for each item.

	Reference file analysis (ms)	Data file analysis (ms)	Related data analysis (ms)	Memory usage (GB)
Pre-change	3159	1929	647	1.94
Post-change	48.87	1218.52	117.25	2.77
Enhancement	64.64	1.58	5.51	0.70

이 평균 약 1.40배 증가함을 확인하였다.

Ⅲ. 결 론

본 논문에서는 다기능레이다 시험 데이터 분석 SW의 속도 개선을 위해 인메모리 기반의 정렬된 맵 구조 및 고속 JSON 파서를 도입하였다. 제안한 기법은 분석 속도를 평균 약 4.14배 향상시켰으며, 메모리 사용량은 약 1.40배 증가하였다. 성능 개선과 자원 소비의 trade-off가 존재하나, 대용량 데이터 환경에서 유의미한 개선 효과를 보였다.

References

[1] J. H. Song, "The development of the data acquisition & analysis system for multi-function radar," *Journal of the Korea Institute of Military Science and Technology*, vol. 14, no. 1, pp. 106-113, Feb. 2011.

[2] H. Lee, Y. W. Kim, and K. Y. Kim, "Study of in-memory based hybrid big data processing scheme for improve the big data processing rate," *The Journal of Korea Institute of Information, Electronics, and Communication Technology*, vol. 12, no. 2, pp. 127-134, Apr. 2019.

[3] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan, "Gorilla: A fast, scalable, in-memory time series database," in *Proceedings of the 41st International Conference on Very Large Data Bases*, Kohala Coast, Hawaii, Aug. 2015, pp. 1816-1827.

[4] G. Langdale, D. Lemire, "Parsing gigabytes of JSON per second," *The VLDB Journal*, vol. 28, no. 6, pp. 941-960, Oct. 2019.