JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 25(4): 293-298, Aug. 2021

병렬처리 알고리즘 적용 유도탄 점검

Inspection of guided missiles applied with parallel processing algorithm

정 의 재* · **고 상 훈** · **이 유 상** · **김 영 성**
LIG넥스원(주) PGM1연구소

Eui-Jae Jung* · **Sang-Hoon Koh** · **You-Sang Lee** · **Young-Sung Kim**
PGM Research and Development Lab, LIGNex1 Co., Gyeonggi-do, 13488, Korea

[요 약]

일반적으로 유도무기의 탐색기와 유도조종장치는 유도탄의 상태를 나타내기 위해 표적, 탐색, 인지, 포착정보를 처리하여 유도무기의 운용 및 제어를 담당하는 역할을 한다. 유도에 필요한 신호는 시선 변화율 신호, 시각 신호, 종말 단계 동체 지향 신호이며, 발사 통제에 필요한 신호는 표적, 감지 신호가 필요하다. 최근 유도탄의 복잡하고 처리하기 어려운 유도탄 신호를 실시간으로 처리하기 위해 유도탄의 데이터 처리 속도를 높여야 한다.

본 연구는 PLINQ(Parallel Language-Integrated Query)의 병렬 알고리즘 방법 중 스톱앤고와 역 열거형 알고리즘을 적용한 후 유도탄 점검 프로그램을 이용하여 실시간으로 유도탄 필요 신호 데이터 처리속도를 비교 후 처리결과를 나타내었다. 도출된 데이터 처리결과 기준으로 다중코어 처리방식과 단독코어 처리방식 CPU(Central Processing Unit) 처리속도 비교, CPU 코어 이용률을 비교하고 병렬처리 알고리즘 적용 시 유도탄 데이터 처리에 효과적 방법을 제안한다.

[Abstract]

In general, the guided weapon seeker and the guided control device process the target, search, recognition, and capture information to indicate the state of the guided missile, and play a role in controlling the operation and control of the guided weapon. The signals required for guided weapons are gaze change rate, visual signal, and end-stage fuselage orientation signal. In order to process the complex and difficult-to-process missile signals of recent missiles in real time, it is necessary to increase the data processing speed of the missiles.

This study showed the processing speed after applying the stop and go and inverse enumeration algorithm among the parallel algorithm methods of PINQ and comparing the processing speed of the signal data required for the guided missile in real time using the guided missile inspection program. Based on the derived data processing results, we propose an effective method for processing missile data when applying a parallel processing algorithm by comparing the processing speed of the multi-core processing method and the single-core processing method, and the CPU core utilization rate.

Key word : Guide Missie, PLINQ(Parallel Language Integrated Query), Pipelining, Query, Serial.

https://doi.org/10.12673/jant.2021.25.4.293

Received 23 July 2021; **Revised** 1 August 2021
Accepted (Publication) 23 August 2021 (30 August 2021)

***Corresponding Author : Euijae Jung**

Tel: +82-32-8026-7447
E-mail: euijae.jung@lignex1.com

Ⅰ. 서 론

최근 정보통신 기술의 발전 및 무기체계 발전으로 데이터와 환경변수 데이터의 실시간 활용에 대한 요구가 급격하고 대용량의 자료수집 분석에 대한 효율적인 처리를 위해 처리 알고리즘이 개발이 절실히 요구된다. 또한, 무기체계 중 유도탄의 성능의 효율적인 분석과 처리를 위한 알고리즘 개발이 필요하다. 따라서 본 연구에서는 유도탄 상태를 확인하기 위해 신호의 탐색, 인지, 포착, 추적을 수행하는 신호를 빠르게 처리하여야 한다[1]. 유도탄 필요 신호를 빠르게 처리하기 위해 본 연구에서는 멀티 코어 컴퓨터에서 다수의 CPU를 이용하여 효율적인 병렬처리(Parallelism Processing)를 연구하고 성능향상을 확인하고자 하였다. 기존 방법인 단독코어 CPU를 이용하여 성능을 향상하는 방법이 아닌, 다중코어로 설계된 CPU에 병렬 알고리즘을 적용하여 유도탄 상태 데이터를 수신 후 처리속도의 효율성을 확인하였다. 고안한 설계 방법은 CPU의 코어의 2개 또는 4개와 같이 다중으로 존재하고 있어 다중 스레드가 동시에 실행된다. 병렬처리 알고리즘 적용 유도탄 점검 장비를 효율적으로 활용하기 위해 프로그래밍 코드의 병렬화 작업을 통하여 다중 프로세스에 분산할 수 있다. 이를 위해 .Net Framework 4.0 이상에서 적용된 병렬화 방식인 PLINQ 패턴 병렬처리 알고리즘을 유도탄 점검 프로그램에 적용하였다[2],[3].

본 연구는 PLINQ을 이용한 병렬처리 알고리즘을 유도탄 점검 프로그램에 적용하여 CPU의 다중코어를 사용 처리하는 방법과 실제 유도탄 점검 시 유도탄 관련 데이터를 병렬처리 알고리즘 적용 후 CPU 코어의 이용률과 처리속도, 데이터 용량 별 처리속도 결과를 단독코어를 사용한 데이터와 비교하여 병렬처리 알고리즘 적용 시 효과를 나타내었다.

Ⅱ. 관련연구 및 요소기술

PLINQ 알고리즘은 LINQ(Language-Integrated Query) 패턴의 병렬 구현이다. PLINQ는 LINQ 표준 쿼리 연산자의 전체집합을 System.Linq 네임스페이스 확장 메서드로 구현하고, 병렬 작업을 위한 추가 연산자를 포함한다. PLINQ는 LINQ의 간편성과 가독성을 병렬 프로그래밍의 기능과 결합 할 수 있다.

2-1 PLINQ 병렬처리 알고리즘

PLINQ의 병렬쿼리는 비 병렬 LINQ to Objects를 사용한다. PLNQ 쿼리는 모든 메모리 내 IEnumerable 또는 IEnumerable<T> 클래스에 작동하며 지연된 실행을 한다. IEnumerable에서 알 수 있듯 열거 가능한 것을 호출한다. 병렬 실행을 통해 PLINQ는 AsParallel 쿼리 작업을 추가하여 특정한 과거 코드에 대한 큰 성능 개선을 얻을 수 있다. 그러나 병렬 처리는 자체 복잡성을 도입할 수 있으며 모든 쿼리 작업은 PLNQ에서 더 빠르게 실행될 수 없다. 하지만 이것이 PLNQ의 실질적인 한계라고 단정할 수 없다. LINQ to SQL(Structured Query Language)이나 독립 프레임워크 조차도 쿼리를 병렬로 실행하기 위해서 병렬화된 데이터베이스 엔진을 사용하기 때문이다. 표 1은 PLINQ 병렬처리의 클래스와 주요 연산자에 대한 설명이다[4],[5].

표 1. PLINQ 알고리즘 클래스 및 주요 함수
Table 1. PLINQ algorithm class and Operator.

Using	
System.Linq	
Class	
ParallelEnumerable	
Operator	
AsParallel	The entry point for PLINQ. Specifies that the rest of the query should be parallelized, if it is possible.
AsSequential	Specifies that the rest of the query should be run sequentially, as a non-parallel LINQ query.
AsOrdered	Specifies that PLINQ should preserve the ordering of the source sequence for the rest of the query, or until the ordering is changed, for example by the use of an orderby (Order By in Visual Basic) clause.
AsUnordered	Specifies that PLINQ for the rest of the query is not required to preserve the ordering of the source sequence.
ForAll	A multithreaded enumeration method that, unlike iterating over the results of the query, enables results to be processed in parallel without first merging back to the consumer thread.
Aggregate	An overload that is unique to PLINQ and enables intermediate aggregation over thread-local partitions, plus a final aggregation function to combine the results of all partitions.

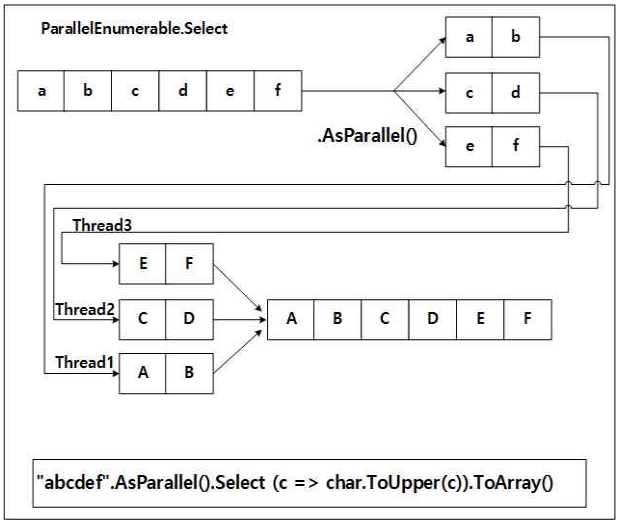


그림 1. PLINQ 알고리즘 처리 구조
Fig. 1. PLINQ Algorithm processing structure.

그림 1의 연산자 AsParallel은 System.Linq.Parallel Enumerable의 메서드 구현 예제 코드이다. ParallelQuery<T>를 기반으로 절차를 입력한다. 추후 호출하는 LINQ 쿼리 연산자가 메서드 ParallelEnumerable에 정의 된 대체 확장 메서드 집합에 배정된다. 이들은 각 표준 쿼리 연산자의 병렬 구현을 제공한다. 기본적으로 입력 절차를 다른 스레드에서 실행되는 코드로 나누어 결과를 다시 단일 출력 절차로 조합하여 처리한다.

2-2 최적의 병렬처리 설계 제시

PLINQ는 스레드를 병렬화를 위해 3개의 알고리즘을 사용하는데 파이프라이닝(Pipelining), 스톱앤고 (Stop and go), 역 열거형(Inverted enumeration)이 있다. 파일프라이닝에서는 하나의 스레드가 순회하는 과정을 처리한다. 파이프라이닝 모드에서는 PLINQ가 사용하는 스레드 수는 일반적으로 코어 수와 같다. 코어가 많은 장비라면 더 많은 병렬로 처리할 수 있다. 또한, 순회하는 스레드가 존재한다. 즉 파이프 라이닝은 코어 순회 스레드를 생성한다. 데이터가 복잡하고 많은 경우에 적합하지만, 별도의 순회 스레드를 관리해주어야 하는 단점이 있다. 두 번째로 스톱앤고 알고리즘은 순회를 시작한 스레드가 쿼리 표현 식을 수행하는 나머지 스레드와 조인(Join)한다. 이 방법은 연산자 ToList 나 ToArray 를 사용해서 쿼리 결과를 즉각 반환해야 할 때 사용된다. 그 외에도 PLINQ가 정렬 연산처럼 작업을 수행 전 전체 결과가 필요할 때 사용된다. 스톱앤고 방식은 메모리를 많이 사용하게 되지만 속도는 조금밖에 개선되지 않는 경우가 있다. 이 알고리즘 방식은 전체 쿼리를 먼저 구성하기 때문에 오버헤드로 인해 성능상의 이점마저 잠식할 수 있다. 마지막으로 최적의 알고리즘으로 사용되는 알고리즘은 역 열거형 알고리즘이다. 이 알고리즘은 결과를 생성하지는 않고, 모든 쿼리식의 결과에 다른 행동을 취하는 경우 사용된다[6]. 그림 2는 역 열거형 계승계산을 수행하는 코드이다.

```
var nums = form n in data.AsParallel()
           where n < 150
           select factorial(n);
Foreach (var item in nums)
      Console.WriteLine(item)
```

그림 2. 역 열거형 계승계산 코드
Fig. 2. Inverted enumeration factorial calculation.

```
var query = from num in nums.AsParallel()
            Where n < 150
            select num;

query.ForAll(e =>
concurrentBag.Add(Compute(e)));
```

그림 3. ForAll 역 열거형 연산자 계승계산 코드
Fig. 3. ForAll Operator inverted enumeration factorial calculation.

역 열거형은 스톱앤고 보다 메모리를 적게 사용하며 결과에 대해 병렬 작업을 수행할 수 있다. 연산자 ForAll은 스톱앤고 보다 메모리를 적게 사용하는 연산자이다. 그림 3 은 ForAll 역 열거형 연산자 계승 코드이다. 그림 2와 그림 3은 역 열거형의 연산자 Foreach와 ForAll는 코드 내용을 같지만, 내부 메모리에서 사용하는 방식이 다르다. 그림 4 는 연산자 Foreach와 ForAll의 차이를 보여준다. 순차적 LINQ 쿼리에서 실행하는 쿼리가 루프에서 또는 연산자 ToList, ToArray 또는 ToDictionary 를 호출하여 열거될 때 지연된다. PLINQ에서 연산자 Foreach 사용하는 경우 병렬로 실행되지 않으므로 모든 병렬 작업의 출력은 반복이 실행되는 스레드로 다시 병합된다. PLINQ에서 쿼리 결과의 최종 순서 지정을 유지해야 하는 경우 및 각 요소에 대해 출력하는 경우 직렬 방식으로 결과를 나타내게 된다. 빠른 쿼리 실행을 위해 유지가 필요하지 않았을 때 와 결과 처리 자체가 병렬로 처리될 수 있는 때 연산자 ForAll를 사용하여 PLINQ 쿼리를 실행한다. ForAll는 최종 병합 단계를 수행하지 않는다.

병렬 알고리즘은 역 열거형 알고리즘의 암달의 법칙(Amdahl's law)을 따르며 다중코어를 사용하게 되며 프로그램의 속도 향상은 데이터 순차적 정렬 기준에 따라 달라진다. ParralleEnumerable의 확장 메서드 또한 예외 없이 이 규칙을 따른다. 메서드의 대부분은 병렬할 수 있지만, 프로그램 전체적인 속도는 병렬 화 할 수 없는 부분의 비중에 의해 결정된다. 표 1의 연산자 AsOrdered 와 AsUnordered 를 사용하면 PLINQ가 결과 절차에서 순서를 지키거나 무시할 수 있다.

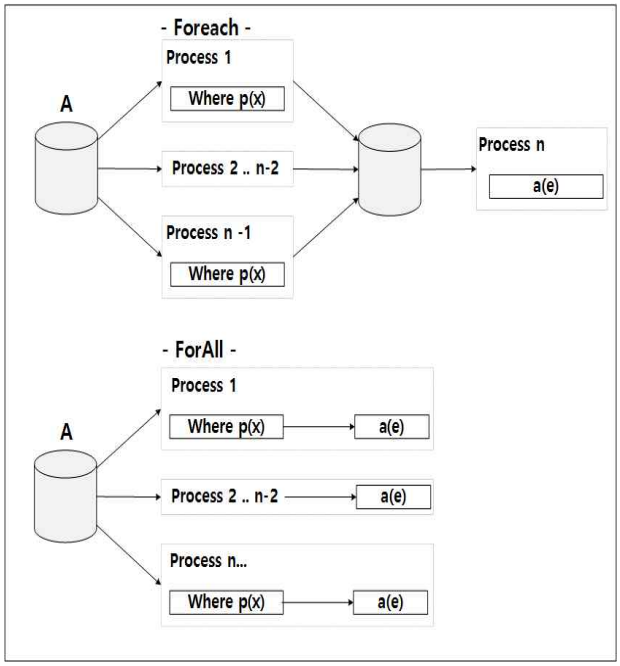


그림 4. Foreach와 ForAll 연산자의 연산 처리방식
Fig. 4. Operation processing method of Foreach and Forall Operator.

Ⅲ. PLINQ 알고리즘 적용 유도탄 점검 장비 시스템

유도탄 점검 장비는 유도탄의 성능을 보장하는 장비로, 유도탄 성능에 관한 결과가 데이터로 저장된다. 유도탄 점검 장비 병렬처리 시스템 구축 시 점검 장비의 모의 명령 송신 데이터, 대상 유도탄의 전자장치 송신에 대한 모의 명령을 응답 데이터 유도탄 점검 장비는 유도탄의 성능을 보장하는 장비로, 유도탄 성능에 관한 결과가 데이터로 저장된다. 유도탄 점검 장비 병렬처리 시스템 구축 시 점검 장비의 모의 명령 송신 데이터, 대상 유도탄의 전자장치 송신에 대한 모의 명령 응답 데이터와 유도탄 내부 상태를 나타내는 실시간 상태 데이터를 대상으로 수행하여야 한다.

본 연구에서는 유도탄 점검 장비와 DAQ(Data Acquisition)를 이용하여 유도탄 상태 신호 데이터를 병렬처리 알고리즘을 적용하여 병렬처리 알고리즘 적용 시와 단독처리 적용 시 CPU의 처리속도, 처리 사용량, 사용 코어 수 등의 효율성을 나타내었다.

3-1 유도탄 점검 장비 병렬처리 시스템 구축

그림 5와 같이 DAQ를 사용하여 유도탄 제어 신호와 유도탄 상태 신호를 송 / 수신할 수 있다. DAQ 내부에는 아날로그 신호와 디지털 신호, 카운터가 외부 입출력 단자를 통해 신호를 전압 신호로 전달되어 DAQ의 클럭 신호에 따라 프로그램으로 데이터를 송 / 수신한다[7]. 병렬처리 알고리즘 내부설계는 그림 6 병렬처리 의사 코드와 같이 스톱앤고와 역 열거형 알고리즘을 혼합 사용한다. 스톱앤고 알고리즘은 DAQ 데이터를 각 점검별 필요 변수에 배열을 전달하고, 역 열거형은 수신된 데이터의 검색 쿼리를 이용하여 유도탄 상태를 전시하였다. 스톱앤고 방식은 메모리를 많이 사용하고 속도 개선 부분이 필요하지만 각 필요 점검 모듈로 병렬처리 알고리즘 내부의 연산자 ToArray 또는 ToList를 사용하여 전체 데이터를 전달하도록 설계되었다. 분석을 위한 시스템 규격은 표 2에 나타내었다. DAQ는 5 KHz 단위로 10개 데이터를 연속 수신으로 병렬처리 알고리즘 내부 변수에 저장한다. 할당된 DAQ 채널에 데이터 수신이 완료된 시점에서 병렬처리 알고리즘은 각 유도탄 점검 모듈로 데이터를 병렬처리하여 전달한다. 그림 7과 같이 데이터 전달 시 병렬처리 알고리즘은 스톱앤고 와 역 열거형 알고리즘을 이용하여 필요 데이터를 동시에 전달하도록 하였다. 동시 전달의 상태는 CPU의 코어 상태로 확인할 수 있다. 그림 6에서 DAQ에서 수신된 채널별 데이터를 병렬처리 알고리즘 내부에서 저장 후, 10msec 간격으로 필요 점검 모듈에 동시에 전달한다.

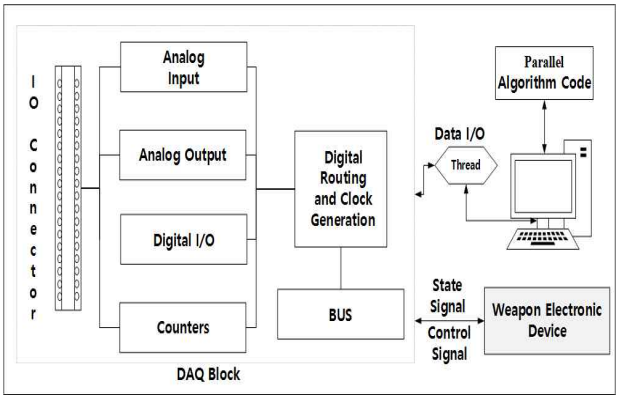


그림 5. 병렬처리 알고리즘 적용 유도탄 점검 장비
Fig. 5. Missile assembly test set using parallel algorithm.

```
Device Chassis = DaqSystem.Local.LoadDevide(Devied)
List<T>ListDevied = new List<T>();
IntPtr pTH = MarshI.AllochHGlobal(10);
Byte[]Data = new Byte[4];
AnalogWaveform ReveiceData;
Using(Task DoWork = new Task())
{
        ListDevied[T].AOChannels.CrateChannel(Chassis[0].DeiviedID,"PORT0",
                                              ChannelLineGrouping.OneChannel
                                               ForAllLine);

        ReveiceData = TaskReader.EndReadWaveform(IAsyncResult)
        foreach(AnalogWaveform<T> waveform in ReveiceData )
        {
            var nums = form In waveform.AsParallel();
                       where ...
                       select ...
            nums.ForAll(item => ...)
        }
        GlovalVariable = nums.ToList()

}
```

그림 6. 병렬처리 알고리즘을 이용한 데이터 처리 의사코드
Fig. 6. Data processing pseudo code using parallel processing algorithm.

표 2. PLINQ 클래스와 연산자
Table 2. PLINQ class and operator.

	CPU	Memory	Operating System
Computer	Intel Core I7 4.20GHz	32 GByte	Windows 10
	Channel Count	Speed	Data Count
DAQ	32 Channel	5 khz	10

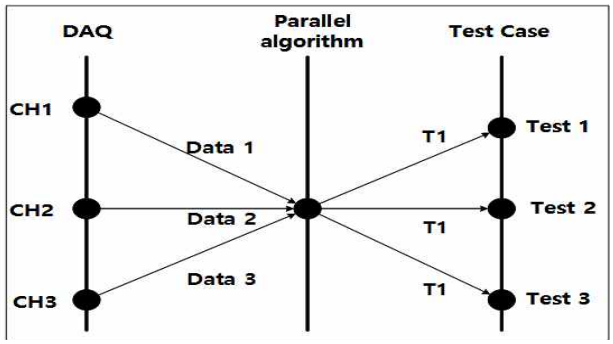


그림 7. 병렬처리 알고리즘을 적용 점검 장비 데이터 처리 구조
Fig. 7. Parallel processing algorithm test equipment data.

processing structure

3-2 유도탄 점검 장비 병렬처리 시스템 검증 및 결과

병렬처리 알고리즘을 적용한 유도탄 점검 장비의 타당성 및 효과성에 대한 검증은 CPU의 처리속도, 처리 사용량, CPU 코어 활용, 병렬처리 수신 시 용량별 데이터 처리 시간을 측정하여 개선 여부를 확인하고자 하였다.

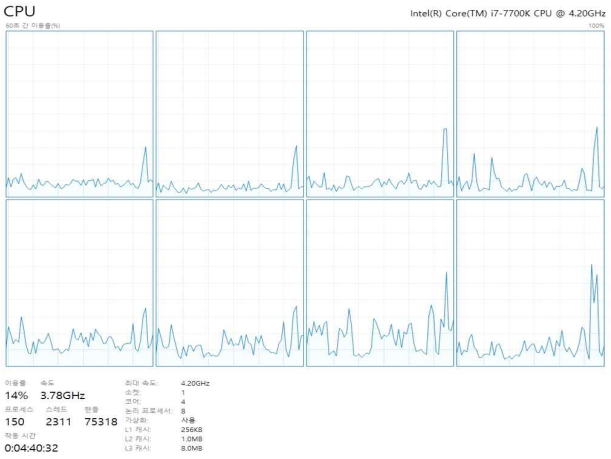


그림 8. 독립처리를 적용한 CPU 처리 결과
Fig. 8. CPU processing result appling serial processing.

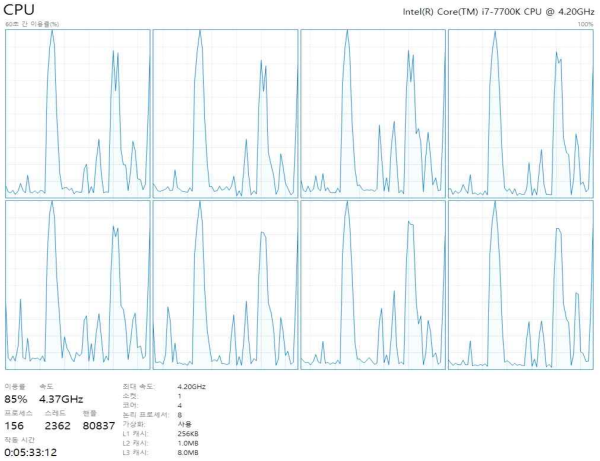


그림 9. 병렬처리 알고리즘 적용한 CPU 처리 결과
Fig. 9. CPU processing result applying parallel processing algorithm.

표 3. CPU 처리방식별 이용률 비교
Table 3. Comparison of CPU utilization by processing method.

	CPU Utilization	Speed	Using Core Count
Serial	31 %	4.33 GHz	1
Parallel	85 %	4.37 GHz	4

표 4. 데이터 용량 별 처리 속도
Table 4. Processing speed by data capacity.

Data Size	Serial Processing	Parallel Processing	Data Format
0.5 MBytes	249 msec	325 msec	format : Binary unit : unsigned short
1 MBytes	439 msec	381 msec	
5 MBytes	202 msec	496 msec	
10 MBytes	4109 msec	809 msec	

그림 8 과 그림 9 는 데이터를 단독처리와 병렬처리 시 CPU 코어의 상태를 확인하였다. 전달되는 데이터의 양에 따라 변화는 있지만, 병렬처리 알고리즘을 적용한 그림 9와 같이 CPU의 코어를 모두 효율적으로 활용한 부분에서 효과적으로 처리되었음을 알 수 있다. 또한, 표 3 와 같이 CPU의 이용률이 약 54 % 속도는 0.04 GHz 차이가 나는 것을 확인하였다. 또한, CPU 내부 코어 수 이용률은 코어가 4개인 CPU를 사용했을 경우 전체를 사용하여 처리한다는 것을 알 수 있다. 표 4는 용량별 처리 데이터를 각 점검데이터 모듈로 전달되는 시간을 측정하였다. 병렬처리 알고리즘 적용 시 단독처리 시보다 데이터가 적었을 때 시간의 효율이 낮았지만, 점검데이터의 용량이 증가 될수록 데이터 처리 속도가 병렬처리와 단독처리 차이를 표 4에서 확인 할 수 있었으며, 10 MBytes 에서는 약 3.3 초 처리속도 차이가 나타나는 것을 확인하였다.

Ⅲ. 결 론

본 논문에서는 병렬처리 알고리즘을 유도탄 점검 장비 적용 후 각 DAQ에서 수신된 데이터를 병렬처리 알고리즘의 스톱앤고와 역 열거형을 이용하여 데이터 처리속도와 CPU 코어 이용률을 확인하여 단독처리 방식보다 병렬처리 시 효율적인 속도 처리의 차이를 확인하였다. 속도를 높이기 위해 병렬처리 알고리즘 쿼리에는 오버헤드를 처리할 만큼 병렬처리 작업이 필요하다. 작업은 각 대리자의 컬렉션의 요소 수를 곱한 값으로 표시될 수 있다. 병렬처리 알고리즘을 유도탄 점검 장비에서 적용 시 통신라인 또는 연구에 사용한 DAQ를 통해 대용량의 데이터를 정해진 시간 안에 처리되어야 한다. 즉 실시간성이 보장되어야 한다. 연산속도 또는 처리속도를 개선을 위한 적합한 점검 장비 구성과 병렬처리 알고리즘으로 병렬처리 시 효과적인 처리를 나타내었다. 그러나 병렬처리가 단독처리보다 더 효율적이다고 결론 내릴 수 없다. 이유는 대용량 데이터를 처리하고자 할 때 쿼리에 병렬처리 포함하지 않는 쿼리가 포함돼 병렬 실행을 느리게 하는 일부 연산자가 포함할 수 있다. 병렬처리 알고리즘 적용 시 적용 장비의 작업의 오버헤드를 포함할 만큼의 병렬처리가 되어야 하며, 시스템의 논리 코어 수를 확인하여 전체 속도를 향상할 수 있다. 또한, PLINQ의 쿼리문 연산자 중 GroupBy, Join 작업에서 오버헤드가 발생하는 쿼리문은 순차처리로 수정하여 적용하면 처리속도를 높일 수 있다.

References

[1] T. L. Song, “Target adaptive guidance for passive homing missiles”, *IEEE Trans. On AES*, Vol33, No1, pp.312~316, Jan. 1997.

[2] D. Geer “Chip makers turn to multicore processors”, *IEEE Computer*, Vol. 38, No 5, pp.11~13, May. 2005.

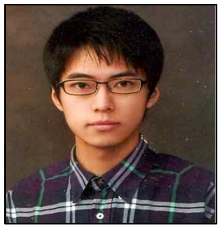
[3] J. Cieslewicz, K. Ross, “Database optimizations for modern hardware”, *Proceedings of IEEE*, Vol 96, No 5, pp.863~879, May. 2008.

[4] MicroSoft. Documentation. Introduction ti PLINQ[Internet]. Available:https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq.

[5] Threading C#, PARALLEL PROGRAMMING [Internet]. Available: http://www.albahari.com/threading/part5.aspx.

[6] Bill Wagner, More effective c#, *Hanbit Media*, pp. 199-251, 2017.

[7] George M. Siouris, “An Implementation for Test set of Missile Fin Control Using DAQ”, *Conference on Information and Control Systems(CICS’11)*, Jejudo Island, pp.261-262, 2011.



정 의 재 (Eui-Jae Jung)
2008년 2월 : 컴퓨터공학과 공학사
2008년2월 ~ 2016년 2월 : ACE Technology S.W연구소 선임연구원
2016년2월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※관심분야 : 계측제어, 정보통신, 유도무기

고 상 훈 (Sang-Hoon Koh)
2011년 2월 : 전기전자컴퓨터공학과 공학석사
2011년 1월 ~ 현재 : LIG넥스원 유도무기연구소 수석연구원
※관심분야 : 유도무기, 정보통신

이 유 상 (You-Sang Lee)
2006년 2월 : 컴퓨터공학과 공학사
2006년 3월 ~ 현재 : LIG넥스원 유도무기연구소 수석연구원
※관심분야 : 유도무기, 정보통신

김 영 성 (Young-sung Kim)
2017년 2월 : 컴퓨터공학과 공학석사
2017년 2월 ~ 현재 : LIG넥스원 유도무기연구소 선임연구원
※관심분야 : 유도무기, 무인항공기 지상체, 자율주행, PUF