한국시뮬레이션학회 논문지
Vol. 26, No. 3, pp. 13-22 (2017. 9)
http://doi.org/10.9709/JKSS.2017.26.3.013
ISSN 1225-5904

다중 스레드 파이프라인 병렬처리를 통한 실시간 시뮬레이션 시각화의 성능 향상 해석 및 적용

이준희† · 송희강 · 김탁곤

Analysis and Application of Performance Improvement of a Real-time Simulation Visualization based on Multi-thread Pipelining Parallel Processing

Jun Hee Lee† · Hee Kang Song · Tag Gon Kim

ABSTRACT

This research proposes and applies a pipelining parallel processing technique to enhance the speed of visualizing the results of real-time simulations. Generally, a simulation with real-time visualization consists of three processes: executing a simulation model, transmitting simulation result, and visualizing simulation result. If we have these processes in serial, the latency from simulation to visualization will be very long, which degrades the speed of visualization of data from real-time simulation. Thus, the main purpose of this research is maximizing performance by adapting pipelining parallel processing technique to the real-time simulation visualization. Also we show that performance is improved by adding multi-threading technique to each process. This paper proposes a theoretical performance model and simulation results of the techniques and then we applied this to an air combat simulation model as a case study. As the result, it shows that the performance is greatly enhanced than the original model's execution time.

Key words : Simulation with Runtime Visualization, Pipelining Parallel Processing, Multi-threading, Performance Model and Analysis

요 약

본 연구는 시뮬레이션을 진행하면서 그 결과를 실시간으로 시각화하는 경우에 파이프라이닝 병렬처리 기법을 적용하여 성능을 개선할 수 있음을 보인다. 일반적으로 실시간 시각화를 포함한 시뮬레이션에서는 모델을 실행하는 프로세스와, 시뮬레이션 결과를 시각화 도구로 전송하는 프로세스, 결과를 받아서 시각화 하는 3개의 프로세스가 있다. 만약 이 프로세스들을 직렬화해서 실행하면 전체 실행시간이 매우 길어져서 시각화의 성능이 저하될 수밖에 없다. 본 연구에서는 기존의 직렬 방식 대신에 파이프라이닝 병렬처리 기법을 적용하여 성능을 개선하고자 한다. 추가적으로 각 프로세스에 다중 스레드 기능을 더하여 더 큰 성능의 개선이 있음을 보인다. 이를 위해 본 논문은 제안된 기법에 대한 이론적 성능모델을 세우고 최대, 최소 성능 향상 조건을 이론적으로 해석하였으며 모의실험하였다. 이 이론을 바탕으로 실시간으로 시각화하는 실시간 공중전 시뮬레이션에 적용한 결과 기존의 직렬화된 실행 성능보다 제안된 이론을 적용한 후의 실행 성능이 크게 향상되었음을 보였다.

주요어 : 실시간 시뮬레이션 시각화, 파이프라이닝 병렬처리, 다중 스레드, 성능모델 및 해석

1. 서론

모델링 및 시뮬레이션(M&S)은 실제 세상에 있는 시스템을 상세하게 묘사한 모델을 만들고, 그 모델에 다양한 입력을 주어가며 시뮬레이션 할 수 있는 환경을 제공

Received: 7 June 2017, **Revised**: 23 June 2017,
Accepted: 23 June 2017

† **Corresponding Author**: Jun Hee Lee
E-mail: the78910@kaist.ac.kr
Electrical Engineering Dept., KAIST, Daejeon, Korea

한다. 모델이 잘 설계되어있다면 그 모델만 시뮬레이션 해도 실제와 유사한 결과를 얻을 수 있고, 실제로 시스템을 테스트해야 할 때 소요되는 자원과 시간을 크게 절약할 수 있다는 점에서 큰 의의가 있다. 이것은 어뢰를 모델링해서 선박이 어떻게 하면 어뢰의 공격을 잘 회피할 수 있는지 분석하는 것 등의 국방 M&S 분야에서 특히 효율적으로 사용될 수 있다(Seo et al., 2011).

단순히 시뮬레이션을 실행하는 것 이외에 산출된 결과를 바탕으로 각화를 하면 데이터의 추세를 육안으로 직접 확인하기가 쉽기 때문에 모델을 더 효율적으로 검증하고 분석할 수 있다. 이러한 시뮬레이션 시각화는 미사일 시스템을 분석하기 위한 국방 M&S 분야(Hwang, 2000) 뿐만 아니라 다른 민간 분야(Lee et al., 2007; Yoon et al., 2014) 에서도 많이 사용되고 있고, 시각화 도구로는 미 해군에서 개발한 SIMDIS(U.S. Naval Research Lab, 2015) 가 자주 활용된다. 이 도구를 활용하면 시뮬레이션이 진행되는 도중에 외부 파라미터를 받아서 시뮬레이션 결과나 내용에 변경이 있을 때 실시간으로 시각화를 진행하는 것으로 실시간으로 변경된 동작에 대한 검증이 더 쉽다는 장점이 있다.

일반적으로 국방 M&S에서는 시뮬레이션에 참가하는 개체의 수가 많다는 특징이 있다. 또한, 전투기등과 같이 상세하게 움직임을 묘사해야하는 필요성이 있을 때에는 단위 시간당 생성되는 데이터가 매우 많아지게 된다. 많은 수의 개체가 있을 때 각 개체가 생성하는 데이터의 수도 많으면 짧은 시간에 매우 큰 수의 데이터가 생성되기 때문에 시뮬레이션을 실행하면서 동시에 시각화를 하기에는 어려움이 생긴다. 따라서 복잡한 국방 M&S에서의 실시간 시각화를 위해서 추가적인 성능 개선 방법이 필요하다.

일반적으로, 시뮬레이션을 실시간으로 시각화하는 경우에는 모델을 실행하는 프로세스(Simulation Process)와, 실행된 결과를 시각화 프로그램으로 전송해주는 프로세스(Communication Process), 그리고 받은 정보를 바탕으로 시각화 하는 프로세스(Visualization Process)의 총 3가지 프로세스가 직렬로 연결되어 동작한다. 각 프로세스의 1회 반복 수행시간이 각각 T_{sim}, T_{com}, T_{vis}라고 정의할 때 직렬 프로세스구조에서의 총 실행시간 $T_{sequence}$ 은 식(1)과 같이 나타낼 수 있다. 이 때 N은 시뮬레이션 loop의 횟수이다.

$$T_{sequence} = N \times (T_{sim} + T_{com} + T_{vis}) \quad (1)$$

본 논문에서는 위의 직렬화된 프로세스들을 병렬화해서 성능을 개선하고(Randolph et al., 1988), 다중 스레드 기법을 추가하여 추가적으로 성능이 개선될 수 있음을 이론적으로 해석한 후에 그 이론을 검증하는 실험 결과를 제시한다. 또한 실제의 실시간시뮬레이션 시각화 사례 연구로서 다수의 전투기로 구성된 공중전 시뮬레이션에 제시된 이론을 적용하여 얼마나 성능이 향상되었는지를 보인다.

2장에서 프로세스 파이프라인 병렬처리 기법과 성능 해석, 모의실험 결과를 보이고, 3장에서는 이 이론을 적용하여 실시간 시각화를 하는 시뮬레이션의 사례연구를 소개한다. 마지막으로 4장에서는 결론을 맺고 본 논문을 마무리한다.

2. 제안하는 파이프라인형 병렬 처리

2.1 파이프라인형 병렬처리

파이프라이닝이란 프로그램의 성능을 개선시키는 기법 중 하나로 명령어들을 병렬적으로 수행하여 실행시간을 단축시킨다. 실제로 이러한 파이프라이닝 기법을 적용하면 프로그램의 성능이 향상된다는 연구 결과가 있다(Cha, Jeon, 1995; Sin, Lee, 1993). 파이프라이닝의 종류로는 CPU level에서 명령어를 병렬적으로 실행하는 명령어 파이프라인, 소프트웨어 상에서 루프를 재구성하는 소프트웨어 파이프라인, 프로세스들이 병렬적으로 실행되면서 데이터를 처리하는 프로세스 파이프라인 등이 있다. 본 연구에서는 프로세스 파이프라이닝 기법을 사용한다.

프로세스들 간에 병렬처리를 할 때, 데이터의 종속성을 주의해야 한다. 서론에서 언급한 실시간 시각화를 포함한 시뮬레이션 상황을 예로 들어서 설명하자면, 모델이 먼저 실행이 되고 그 결과를 바탕으로 시각화를 해야 하기 때문에 프로세스간의 데이터 종속성이 생기게 된다. 따라서 어느 정도 시간차를 두고 실행을 해야 하는데, 이것의 개념적인 구조는 Fig. 1과 같이 도식화될 수 있다. 즉, Simulation 프로세스의 job의 결과를 Communication 프로세스가 전송하고, Visualization 프로세스가 받아 시각화를 담당한다.

Fig.1에서 확인할 수 있듯이, 실제 경우에서 각 job의 수행시간이 모두 다를 수 있기 때문에 무조건 구현하여 성능을 측정하는 것이 아니라 어느 조건에서 최대/최소 성능을 내는지 모델을 세워 해석하고 분석한 후에 실제 시스템에 해석된 결과를 적용하고자 한다.

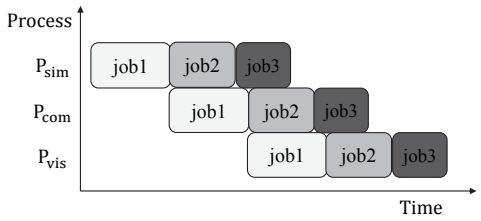

Fig. 1. Process Pipelining

2.2 단일 스레드 기반 파이프라인형 병렬처리

2.2.1 단일 스레드 기반 성능 해석

2.1에서 언급했듯이 프로세스의 종류가 Simulation, Communication, Visualization의 세 가지가 있는 경우에서, 어떤 프로세스의 실행시간이 제일 오래 걸리는지에 따라서 다음과 같이 세 가지 경우로 분류할 수 있다.

$$case1 : \max(T_{\text{sim}}, T_{com}, T_{vis}) = T_{\text{sim}}$$
$$case2 : \max(T_{\text{sim}}, T_{com}, T_{vis}) = T_{com}$$
$$case3 : \max(T_{\text{sim}}, T_{com}, T_{vis}) = T_{vis}$$

이에 각 경우에 시간에 따라서 프로세스들이 task를 처리할 때 소요되는 시간에 대한 그래프는 각각 Fig. 2, Fig. 3, Fig. 4와 같이 나타낼 수 있다.

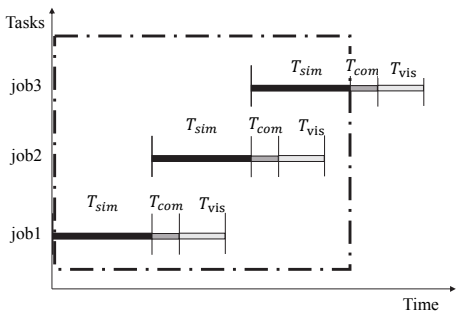

Fig. 2. Task Handling Process According to Time of Case1

Fig. 2, Fig. 3, Fig. 4에서 전체 실행시간은 각각의 경우마다 가장 시간이 오래 걸리는 프로세스에 따라서 영향을 받는 것을 확인할 수 있다. 즉, 실행시간이 가장 긴 프로세스 (네모 상자로 표시되어 있는)의 N번의 job 실행시간에 나머지 두 프로세스의 자투리 실행시간을 더한 것이 총 실행시간이 된다는 것을 확인할 수 있다. 이것을 수식으로 표현하면 다음과 같이 나타낼 수 있다. 여기서 N은 Task(job)의 개수 또는 시뮬레이션 루프의 횟수이다. 즉,

$$T_{case1} = N \times T_{\text{sim}} + T_{com} + T_{vis}$$
$$T_{case2} = T_{\text{sim}} + N \times T_{com} + T_{vis}$$
$$T_{case3} = T_{\text{sim}} + T_{com} + N \times T_{vis}$$

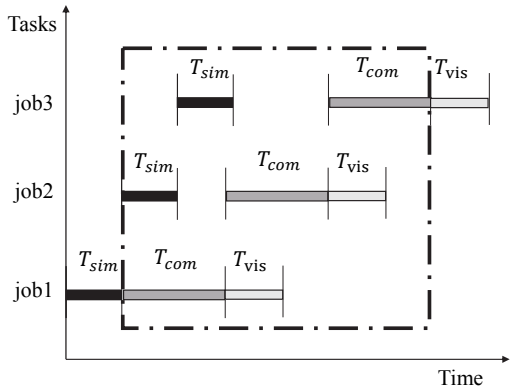

Fig. 3. Task Handling Process According to Time of Case2

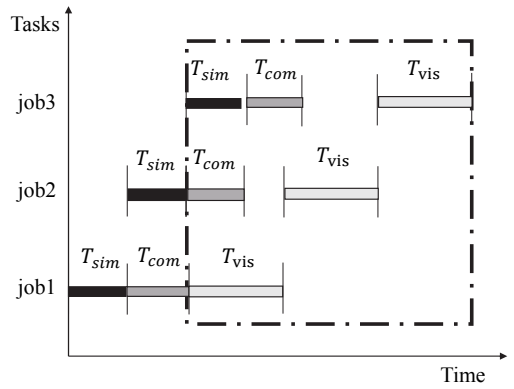

Fig. 4. Task Handling Process According to Time of Case3

위 수식을 일반화하여 병렬처리를 했을 경우의 전체 소요시간을 다음과 같은 식(2)로 나타낼 수 있다.

$$T_{\text{parallel}} = (T_{\text{sim}} + T_{com} + T_{vis}) + (N-1) \times \max(T_{\text{sim}}, T_{com}, T_{vis}) \quad (2)$$

이것을 바탕으로, 각 프로세스를 순차적으로 실행했을 때보다 병렬적으로 실행했을 때 얼마나 성능이 개선되었는지를 비율로 나타낸 가속비율(Acceleration Ratio)을 T_{sequence}를 T_{parallel}로 나눈 것으로 정의하고, 정리하면 식(3)과 같다.

$$Accerelation\ Ratio(AR) = T_{\text{sequence}} / T_{\text{parallel}}$$

$$= \frac{N\times(T_{sim}+T_{com}+T_{vis})}{(T_{sim}+T_{com}+T_{vis})+(N-1)\times\max(T_{sim}+T_{com}+T_{vis})} \quad (3)$$

식 (3) 에서 N의 값이 커지면 N-1은 N으로 근사가 가능하다. 따라서 시뮬레이션 횟수가 커진다면 다음과 같은 식으로 유도가 가능하다.

$$AR \approx \frac{N\times(T_{sim}+T_{com}+T_{vis})}{N\times\max(T_{sim},T_{com},T_{vis})} \quad (4)$$

$$= \frac{(T_{sim}+T_{com}+T_{vis})}{\max(T_{sim},T_{com},T_{vis})}$$

$$= \begin{cases} 1+\dfrac{(T_{com}+T_{vis})}{T_{sim}} & \text{if } case1 \\ 1+\dfrac{(T_{sim}+T_{vis})}{T_{com}} & \text{if } case2 \\ 1+\dfrac{(T_{sim}+T_{com})}{T_{vis}} & \text{if } case3 \end{cases}$$

즉, N-1을 N으로 근사하지 않은 가속비를 나타내는 식 (3)에서, 프로세스의 순차 실행시간 $T_{sequence}$인 분모를 상수라고 했을 때, 가속비율의 최대값과 최소값을 다음과 같이 근사값을 구할 수 있다.

$$AR_{\max} = \frac{3N}{N+2}(where,\ T_{sim}=T_{com}=T_{vis}) \approx 3,\ (N\to\infty)$$
$$AR_{\min} \approx 1,\ (\ T_i\ is\ dominant, i\in\{sim, com, vis\}) \quad (5)$$

식(5)의 AR_{max}에서 모든 수행시간이 같을 때, 즉 T_{sim} = T_{vis}= T_{com}인 경우에 가속비율이 최대 3의 값을 가진다는 것을 알 수 있다. 또한 식(4)의 AR_{min}에서 어떤 한 값이 다른 두 값에 비해 현저하게 클 때는 가속비율이 1로 수렴함을 관찰할 수 있다. 즉, T_{sim}의 값이 T_{com} + T_{vis}보다 현저하게 크거나(case1: $T_{sim} \gg T_{com} + T_{vis}$), T_{com}의 값이 T_{sim} + T_{vis}보다 현저하게 크거나 (case2: $T_{com} \gg T_{sim} + T_{vis}$), 아니면 T_{vis}의 값이 T_{sim} + T_{com}보다 현저하게 크면(case 3: $T_{vis} \gg T_{sim} + T_{com}$) 가속비율이 1로 수렴한다는 것 알 수 있다.

2.2.2 단일 스레드 기반 성능 모의실험

위 해석결과를 검증하기 위하여 세 개 프로세스에 대해 3단계 파이프라인 형태의 병렬처리 실행 환경을 UNIX 환경 상의 모의 모델로 구현하여 각각의 수행시간을 달리 하면서 모의실험을 진행하였다. 프로세스간 통신은 TCP/IP로 하고, 각 수행시간은 주어진 시간만큼 지연하는 알고리즘을 적용하였다.

Fig. 5는 가속비율의 최대값을 산출하기 위한 실험에 대한 결과를 나타낸 것이다. 시뮬레이션 루프의 반복 횟수(N)는 100회이고, T_{sim}과 T_{com}이 모두 10ms가 걸린다고 가정하고 T_{vis}의 시간을 5ms에서 15ms까지 변화시켜가며 실험을 진행하였다. T_{vis}가 10ms에 가까워질수록 가속비율이 최대 값인 3에 근접하고, 시간이 10ms보다 작거나 커지게 되면 가속비율이 감소하는 추세를 확인할 수 있다. 따라서 $T_{vis} = T_{sim} = T_{com}$일 때 가속비율이 최대값을 갖는다는 것을 확인할 수 있다.

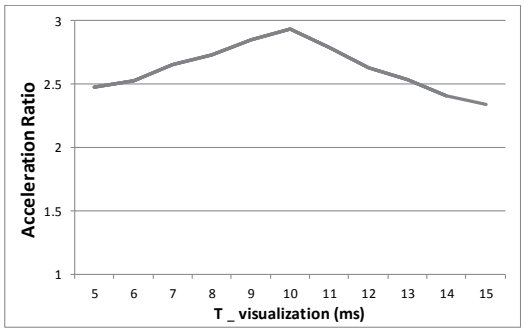

Fig. 5. Acceleration Ratio According to Relative Execution Time, $T_{sim} = T_{com} = 10ms$

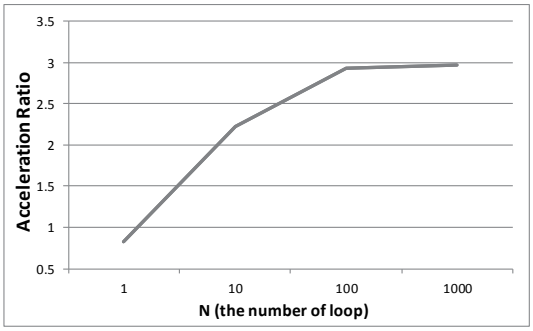

Fig. 6. Acceleration Ratio According to N(the number of loop), $T_{sim} = T_{com} = T_{vis}$

다음으로, 모든 프로세스의 실행시간이 10ms로 동일하다고 가정했을 때 N의 크기에 따라서 가속비율이 어떻게 변화하는지를 실험해 보았다. Fig. 6은 N의 크기가 커질수록 가속비율이 이론적 최대값인 3에 근접하는 것을

보여준다.

2.3 다중 스레드 기반 파이프라인형 병렬처리

2.3.1 다중 스레드 기반 성능 해석

본 절에서는 2.2 절에서 제안한 파이프라이닝 기법에 추가적으로 각 프로세스 내에 다중 스레드 기술을 추가하는 방식을 사용하여 시뮬레이션이 완료된 데이터를 전송하는 통신프로세스와 그것을 시각화 하는 시각화프로세스에 걸리는 부하를 줄이는 방법을 고안하였다. 여기서, 시뮬레이션 프로세스의 다중 스레드는 고려하지 않기로 하였는데 이는 현실적으로 시뮬레이션 모델 실행은 시뮬레이터 알고리즘에 영향을 받기 때문에 아무런 고려 없이 스레드를 추가하면 모델이 정상적으로 실행되지 않을 위험이 있기 때문이다.

통신(communication) 프로세스의 스레드 개수를 k1, 시각화(visualization) 프로세스의 스레드 개수를 k2개라고 할 때, 이상적으로 스레드 개수만큼 수행시간이 짧아진다고 가정하면 $T_{com,k1} = T_{com}/k1$으로 나타낼 수 있고, $T_{vis,k2} = T_{vis}/k2$로 표현할 수 있다. 이를 2.2 절의 식 (2)에 적용시켜 보면 다음과 같이 다중 스레드 기법을 추가한 병렬수행시간 $T_{parallel}$은 식(6)과 같이 유도될 수 있다.

$$T_{\text{parallel}} = (T_{\text{sim}} + \frac{T_{com}}{k1} + \frac{T_{ani}}{k2}) + (N-1) + \max(T_{\text{sim}}, \frac{T_{com}}{k1}, \frac{T_{vis}}{k2}) \quad (6)$$

또한 식(3)에 이를 반영한 새로운 가속비율은 식 (7)과 같이 얻을 수 있다.

$$AR = \frac{N \times (T_{\text{sim}} + T_{com} + T_{vis})}{(T_{\text{sim}} + \frac{T_{com}}{k1} + \frac{T_{vis}}{k2}) + (N-1) \times \max(T_{\text{sim}}, \frac{T_{com}}{k1}, \frac{T_{vis}}{k2})} \quad (7)$$

위 식 (7)을 기반으로 2.2절에서와 같이 다중 스레드 기법을 추가한 가속비율의 최대값과 최소값도 단일 스레드에 대해서 식(5)에서와 구한 것과 같은 논리로 식(8)과 같이 구할 수 있다.

$$AR_{\max} = \frac{(1+k1+k2)N}{N+2} \ (where,\ T_{sim} = \frac{T_{com}}{k1} = \frac{T_{vis}}{k2}) \approx 1+k1+k2,\ (N \to \infty)$$

$$AR_{\min} = \min(1, k1, k2)\ (where,\ T_{sequence} = T_{parallel}) = 1. \quad (8)$$

위 식 (8)에서 $AR_{\max}$는 식 (5)와 같이 다중 스레드를 적용한 각 프로세스의 실행시간 $T_{com,k1} = T_{vis,k2} = T_{sim}$ 이 모두 같았을 때에 최대 스피드가 나오게 된다. 즉, $T_{com,k1} = T_{com}/k1 = T_{vis,k2} = T_{vis}/k2 = T_{sim}$ 로 가정할 때 최대의 가속비를 얻게 된다. 역으로 T_{sim}의 상대속도를 1로 했을 때, $T_{com} = k1 \cdot T_{sim}$, $T_{vis} = k2 \cdot T_{sim}$ 즉, 통신프로세스의 수행시간이 시뮬레이션 프로세스의 $k1$ 배, 시각화 프로세스가 $k2$ 배 걸릴 때, 각각 스레드 수를 $k1$, $k2$개를 두어 세 프로세스 속도를 맞추어주었을 때 최대임을 나타낸다. 또한 식 (8)에서 $AR_{\min}$은 $k1$, $k2$가 1이상일 경우이므로 최소 가속비는 1에 그침을 알 수 있다. 또한, T_{com}이나 T_{vis} 가 T_{sim} 보다 작게 걸릴 경우, 즉 $k1$, $k2$가 1보다 작을 경우에는 최대 가속비는 식 (5)와 같이 3 미만이 됨을 알 수 있고, 1보다 클 경우 경우에는 3이상이 될 수도 있음을 알 수 있다.

여기서 유의할 것은 식 (6)에 대한 가정, 예를 들어 $T_{com,k1} = T_{com}/k1$은 다중 스레드를 적용했을 때 스레드 수만큼 속도가 증가한다는 가정인데, 이는 스레드간 알고리즘에 전혀 간섭이 없고, CPU 및 메모리 자원이 충분하다는 것을 전제한 것이다. 하지만, 현실적으로는 그렇지 않기 때문에 식 (8)에서 유도된 가속비의 최소, 최대값은 이상적인 경우에만 적용된다는 것이다. 따라서 실제로는 등식이 성립할 때의 스레드의 개수 k1, k2는 가속비율의 최댓값을 산출하기 위한 최소의 스레드 개수가 되며, 가속비율의 최대값은 그 k1, k2값에 바운드 된다.

2.3.2 다중 스레드 기반 성능 모의실험

실제 상황에서는 각 프로세스의 실행시간이 동일한 값을 가지기는 어렵다. 따라서 1회 시뮬레이션 반복시간 T_{sim}은 10ms, T_{com}은 30ms, T_{vis}은 15ms로 가정하고 2.2.2절과 같은 모의 모델에 다중 스레드를 추가한 시뮬레이션 모델을 구현하고 모의 실험을 진행하였다. 각 프로세스 및 스레드를 구현하였고, 프로세스간 통신은 TCP/IP를 사용하였으며, 각 스레드는 수행시간만큼 시간을 지연하는 알고리즘을 사용하였다.

먼저 통신프로세스의 스레드의 개수만 변화시키고 나머지 프로세스의 스레드 개수는 1개로 고정해서 실험을 진행했다. 앞선 가속비율의 최대값 공식에 따르면 $T_{sim} = T_{com,k1} = T_{com}$/k1이 될 때 가속비율이 최대값을 갖는

다. 이 예에서는 T_{com} = 3 • T_{sim}의 비율을 갖기 때문에 (T_{com} = 30ms, T_{sim} = 10ms) k1 = 3 이상이 될 때 최대가 되어야 한다. 모의실험 결과를 Fig. 7에 나타냈으며 이론치에 근접하게 k1 ≥ 3일 때 가속비율이 최대가 되는 것을 확인할 수 있다. 스레드의 개수가 그것보다 커지면 가속비율이 더 증가하거나 소폭 감소할 수도 있는데, 이것은 실험에서 생기는 오차범위 안에 속한다.

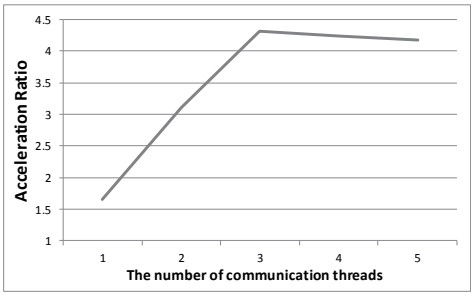

Fig. 7. Acceleration Ratio According to the number of communication thread k1

다음으로는 visualization 프로세스의 스레드의 개수만 변화시키고 나머지 프로세스의 스레드는 1개로 고정해서 실험을 진행했다. T_{com} = 1.5 • T_{sim}의 비율을 갖기 때문에 (T_{sim}= 10ms, T_{vis} = 15ms) k2 = 1.5가 될 때 가속비율이 최대값이 되어야 하지만, 스레드는 정수로만 나타낼 수 있기 때문에 k2가 2개 이상일 때 최대값이 나오는 것을 Fig. 8에서 확인할 수 있다. 이 예에서도 가속비율이 소폭 감소하는 것은 실험 내 오차범위 안에 속한다.

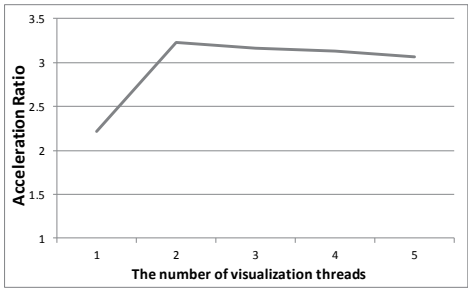

Fig. 8. Acceleration Ratio According to the number of visualization thread k1

Fig. 9는 이 두 가지 경우를 합쳐서 실험을 진행한 결과를 나타낸다. 스레드 k1이 3개, 스레드 k2가 2개 일 때 가속비율이 최대가 되는 것을 확인할 수 있고 이것은 앞서 세운 가설인 가속비율의 최대값은 T_{sim} = T_{com}/k1 = T_{vis}/k2 일 때로 결정된다는 것을 입증한다.

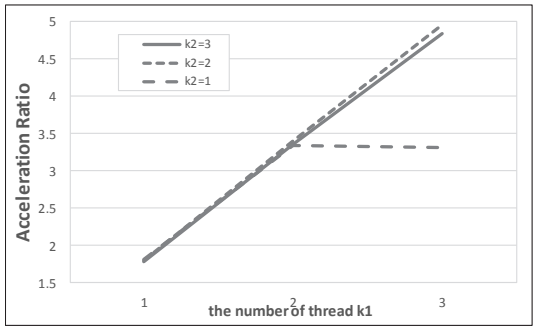

Fig. 9. Acceleration Ratio According to the number of thread k1 and k2

3. 사례 연구: 공중전 시뮬레이션 시각화

3.1 공중전 모델

2장에서 제안한 이론의 검증을 위하여 모델 실행, 통신, 시각화의 세 가지 프로세스로 구성된 공중전 모델을 사용하였다. 본 모델은 6자유도 기동 방정식으로 묘사된 전투기(Choi et al., 2016) 들이 입력받은 일정 시나리오에 따라서 공대공 교전 혹은 공대지 교전을 수행하고, 그 임무 수행 과정을 시각화 툴인 SIMDIS를 활용해서 실시간으로 시각화한다(Lee et al., 2017).

공중전 모델은 Fig. 10에서 보이는 것과 같이 아군 진영과 적군 진영이 대칭적으로 전투기 모델, 미사일 모델, 기지 모델, 지상 레이더 모델, 그리고 통신기기 모델로 구성되어 있다. 그 중 전투기 모델은 내부적으로 기동모델, 조종사모델, 그리고 탐지모델을 가지고 있는 하나의 독립적인 전투 객체 모델이다. 전투기의 탐지모델은 전투기를 기준으로 적기를 탐지하며, 조종사 모델이 상황 판단을 하고 결심을 내리면 그 결심을 바탕으로 기동모델이 전투기의 움직임을 제어하게 된다. 일반적으로 전투기 2대가 하나의 편대로 구성되는데, 편대의 리더가 특정 기동결심을 하고 편대원에게 기동결심 정보를 전송해주면 편대원이 그 정보를 받아서 같이 임무를 수행하는 구조이다. 모든 객체들 간의 의사소통 과정은 통신 장비를 통해서 이루어지게 된다. 지상 기지는 전투기에게 이륙하라는 명령을 내리는 역할을 한다. 지상 레이더 기지는 적 전투기 위치를 탐지해서 아군 전투기들에게 적 전투기의 위치를 알려주는 역할을 한다.

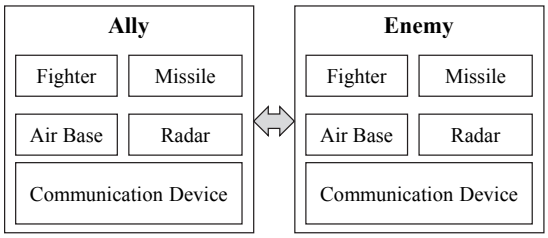

Fig. 10. Air Combat Model Structure (Lee et al., 2017)

공중전 모델의 시스템 구조는 Fig. 11과 같다. 공중전 모델을 시뮬레이션 엔진에서 실행하고(Simulation 프로세스) 실행된 결과를 SIMDIS로 전송한다(Commun ication 프로세스). 그리고 SIMDIS는 수신한 결과를 바탕으로 시각화 작업을 수행한다(Visualization 프로세스).

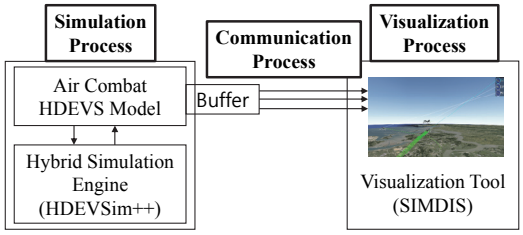

Fig. 11. Air Combat Model System Structure (Lee et al., 2017)

공중전 모델에서 SIMDIS로 전송하는 정보는 화면에 디스플레이 될 객체들의 좌표정보이다. 전투기는 10ms마다 기동방정식을 실행하며 위치를 갱신하고 그 정보를 SIMDIS로 전송하는데, 갱신 주기가 짧기 때문에 전투기 개체수가 많아질수록 데이터 전송에 더 큰 부하가 걸리게 된다. 따라서 본 사례연구에서는 전송할 객체정보를 buffer에 저장하고 통신 프로세스의 스레드를 여러 개 두어서 buffer에 있는 정보를 빠르게 전송하는 방식을 사용했다.

SIMDIS로 정보를 전송하는 통신 프로세스의 소스코드 부분은 Fig. 12에 보여준다. 모델이 실행될 때마다 위에서 언급한 buffer(Fig. 12에서의 GLOBAL_POS_BUFFER)에 시각화에 필요한 정보를 삽입한다. 이를 통신 프로세스가 buffer에서 정보를 추출해내어 SIMDIS로 전송한다. 다중 스레드로 동작하기 때문에 데이터의 일관성을 위해서 여러 스레드가 동시에 buffer에 접근하지 못하도록 critical section lock을 두었다. lock을 얻은 스레드는 buffer에서 하나의 entity를 추출한 후, SIMDIS의 SendData()라고 하는 API를 이용하여 시각화 될 정보를 SIMDIS에 전송한다.

```
while(true)
{
    EnterCriticalSection(&g_cs);
    if(!GLOBAL_POS_BUFFER.empty()){
        pos = GLOBAL_POS_BUFFER.front();
        GLOBAL_POS_BUFFER.pop();
        LeaveCriticalSection(&g_cs);
        SIMDIS_IF->SendData(pos._id,pos._n,pos._e,
        pos._d, pos._yaw, pos._pitch, pos._roll);
    }
    else
        LeaveCriticalSection(&g_cs);

    if(isSimEnd && GLOBAL_POS_BUFFER.empty())
        break;
}
```

Fig. 12. Communication Thread Source Code

본 공중전 모델을 실행해서 SIMDIS를 통해 시각화를 한 모습이 Fig. 13에 나타내었다.

Fig. 13. Screenshot of Air Combat Model Visualization using SIMDIS

3.2 공중전 시뮬레이션 실험

공중전 시뮬레이션 실험을 진행한 컴퓨터 환경은 OS: Windows10, CPU: i7-6700(3.4GHz), RAM: 16GB, GPU: GTX960과 같다.

첫 번째로 전투기가 10대일 때 기준으로 통신 프로세스의 스레드 개수를 증가시키며 실험을 진행했다. Simulation 실행시간만 따로 산출한 결과는 28초, Communication 프로세스의 실행시간은 스레드의 개수를 1개부터 5개까지 하나씩 변화시켜가며 20번씩 테스트를 진행한 결과, 평균적으로 각각 119.17초, 50.54초,

51.44초, 47.60초, 47.13초가 나왔다. 평균 실행시간과 그에 따른 표준편차가 Table 1에 나와 있다.

Table. 1. Average execution time and standard deviation according to the number of thread

Number of repetition	The number of thread	Average execution time (sec)	Standard deviation
20	1	119.17	5.16
	2	50.54	1.07
	3	51.44	0.63
	4	47.60	0.63
	5	47.13	0.81

시뮬레이션 프로세스와 통신 프로세스의 실행시간 비율을 구하기 위해서 통신 스레드가 1개일 때의 시간과 시뮬레이션 프로세스의 실행시간을 비교한다. 스레드가 한 개일 때 실행시간이 100초로서 약 T_{com} = 3.6 * T_{sim}의 비율을 갖게 된다. 따라서 스레드의 개수가 4개 이상일 때 가속비율이 최대가 되는 것으로 기대될 수 있고, Fig. 14에서 그것을 확인할 수 있다. 이상적인 이론의 가속비율 최대값은 1 + k1으로 결정되어야 하지만 실제 실험결과는 다른 것을 확인할 수 있다. 실험을 할 때 T_{sim}의 실행시간이 T_{vis}의 실행시간보다 더 길다는 가정 하에서 수행되었지만, T_{vis}의 실행시간이 T_{sim}의 실행시간보다 더 길어서 그 시간에 최대 가속비율이 bound 되었을 것이라는 추론을 할 수 있다. SIMDIS는 기존에 있는 API를 사용해서 활용할 수밖에 없고, 통신 프로세스를 제외한 T_{vis}의 시간만 따로 측정하기 어렵다는 한계점이 있다.

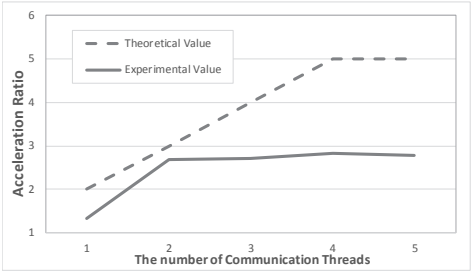

Fig. 14. Acceleration Ratio According to the number of communication thread

다음으로 아무런 병렬처리 과정을 거치지 않은 기존의 공중전 시뮬레이션을 실행한 것과, 단일 스레드 기반 병렬처리 기법을 적용시킨 것, 다중 스레드 기반 병렬처리 기법을 적용시켜 실행한 것의 실행 시간을 동일한 모델과 시나리오 상에서 비행기 개수만 변화시켜가며 비교함으로써 제안한 이론을 실제 프로그램에 적용했을 때 성능향상이 얼마나 일어나는지 확인했다. 다중 스레드 기반에서는 위의 실험 결과에 따라 스레드가 4개일 때를 고려했다. 실험에 사용된 시나리오는 전투기들이 이륙 지점에서 이륙하고 편대를 구성해 공격목표 지점을 경유해서 다시 이륙 지점 기지로 귀환하는 경로 비행을 하는 것이다. 실시간 시뮬레이션 시각화에 걸리는 시간을 기존의 방법과 제안된 방법을 적용하여 비교 측정했다.

Fig. 15에서 확인할 수 있듯이, 별다른 기법을 적용하지 않았을 때의 실행시간과 다중 스레드 기반 파이프라이닝 병렬처리 기법을 적용시켰을 때의 실행시간은 약 3배 정도 차이 나는 것을 알 수 있고, 이것은 Fig. 14에서 나온 가속비율과 유사한 정도로 시간이 단축되었음을 입증한다.

본 사례연구는 SIMDIS라는 상용화된 툴을 사용했기 때문에 모델 개발자가 시각화 프로그램의 내부적인 동작까지 자세히 알기가 어렵고 수정이 불가능하다. 그런 이유로 visualization 프로세스의 스레드 개수를 바꿀 수 없기 때문에 Fig. 11에서 볼 수 있듯이 communication 프로세스의 스레드만 추가해서 실험을 진행했다는 한계점이 있다. 그럼에도 불구하고 괄목할 만한 성능의 개선이 있었다는 점에서 큰 의의가 있다.

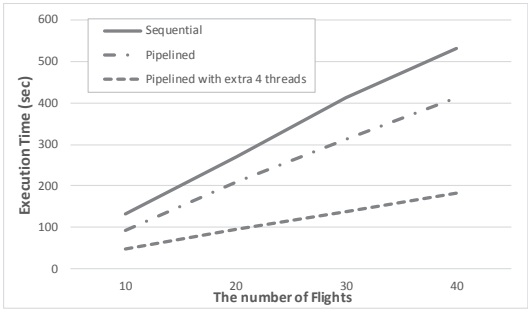

Fig. 15. Comparison of Execution time According to the number of flights

4. 결론

실시간 시각화가 포함된 실시간 시뮬레이션의 성능을 개선하기 위하여 본 논문에서는 기존의 직렬 프로세스 방식 대신에 파이프라인형 병렬화 기법 및 스레드 기법을 적용하기 위하여 성능해석 모델을 세우고 분석하여

최대/최소 성능을 내기 위한 조건을 해석하고 이를 사례 연구에 적용하였다.

제안한 이론을 공중전 시뮬레이션 시각화 사례연구에 적용한 실험을 통해서 기존의 순차실행 방식에 비해 다중스레드 기반의 병렬처리 방식을 적용했을 때 최대 3배에 달하는 성능의 개선이 있음을 확인했다. 이 연구 성과는 앞으로의 실시간 시뮬레이션 시각화를 수행하는 시뮬레이션 연구 분야에 있어서 현실적으로 적용할 수 있는 이론적, 실제적 사례가 될 것으로 본다.

References

Cha, H. K., and Jeon. J. S. (1995) "A Pipelining Scheduling for Algorithm Loops", *Journal of KISS(A)*, 22(10), 1426-1436.
(차혜경, 전주식. (1995) "루프의 파이프라이닝 스케쥴을 위한 알고리즘", 정보과학회논문지(A), 22(10), 1426-1436.)

Hyo-kwang Lee, Pilwon Hur, JunKyu Park, Soonhung Han. (2007) "Real-time 3D Visualization of Underwater Vehicle Simulation", *Proceedings of the Society of Computational Design and Engineering Conference 2007*, 401-407.
(이효광, 허필원, 박준규, 한순흥. (2007) "수중운동체 시뮬레이션의 3 차원 실시간 가시화", 한국CDE 학회 학술발표회 논문집, 401-407.)

Hwang Heung-Suk (2000) "Short-Range Missile System Performance Evaluation Model Using GUI-Type Simulation", *Proceedings of the Korea Society for Simulation Conference 2000*, 148-152.
(황흥석 (2000) "시각화-시뮬레이션방법을 이용한 단거리미사일시스템의 성능산정모델", 한국시뮬레이션학회 학술대회 논문집, 148-152.)

Jun Hee Lee, Seon Han Choi, Ho Dong, Yoo, Jung Koo, and Tag Gon Kim (2017) "Development of Air Combat HDEVS Model Implemented in HDevsim++ Environment", *Proceedings of the 2017 Summer Simulation Multiconference*, Bellevue, Washington, USA.

Kyung-Min Seo, Hae Sang Song, Se Jung Kwon and Tag Gon Kim (2011) "Measurement of Effectiveness for an Anti-torpedo Combat System Using a Discrete Event Systems Specification-based Underwater Warfare Simulator", *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol. 8, No. 3, pp. 157-171.

Nelson, Randolph, Don Towsley, and Asser N. Tantawi (1988) "Performance analysis of parallel processing systems", *IEEE Transactions on software engineering*, 14.4: 532-540.

Seon Han Choi, Jun Hee Lee, Sang Hyun Lee, Ho Dong, Yoo, Jung Koo, and Tag Gon Kim (2016), "6 DoF Aircraft Simulation Model Capable of Handling Maneuver Events (WIP)", *Proceedings of the 2016 Summer Simulation Multiconference*, Montreal, Canada.

Sin, H. J., Lee, K. H. (1993) "A Study on Efficient Software Pipelining Algorithm", *Proceedings of the Communications of the Korean Institute of Information Scientists and Engineers Conference 1993*, 20(2), 449-452.
(신화정, 이기호. (1993). "효율적인 소프트웨어 파이프라이닝 알고리즘에 관한 연구", 한국정보과학회 학술발표논문집, 20(2), 449-452.)

U.S. Naval Research Laboratory (2015), SIMDIS User's Manual, Available from: https://simdis.nrl.navy.mil

WonBae Yoon, Namil Lee, CheolHo Hwang, JungHyun Han. (2014). "Real-time Simulation and Visualization of Iron Filing in Magnetic Field", *Proceedings of the HCI Society of Korea Conference 2014*, 547-550.
(윤원배, 이남일, 황철호, 한정현. (2014) "자기장 내 철가루 움직임의 실시간 시뮬레이션 및 시각화", 한국HCI학회 학술대회, 547-550.)

이 준 희 (the78910@kaist.ac.kr)

2016 한동대학교 컴퓨터공학 학사
2016~ 현재 KAIST 전기및전자공학부 석사과정

관심분야 : DEVS Modeling & Simulation, Hybrid Modeling & Simulation

송 희 강 (hghsong95@naver.com)

2014~ 현재 한동대학교 전산전자공학부 학사과정

관심분야 : Signals & Systems, Communication Systems, Modeling & Simulation

김 탁 곤 (tkim@kaist.ac.kr)

1988 Univ. of. Arizona, 전기및컴퓨터공학과 박사
1980~1983 부경대학교, 통신공학과, 전임강사
1987~1989 (미)아리조나 환경연구소, 연구엔지니어
1989~1991 Univ. of Kansas, 전기및컴퓨터공학과, 조교수
1991~현재 KAIST 전기전자공학부 교수
- 한국시뮬레이션 학회 초대 편집위원장 및 학회장 역임
- SIMULATION(국제시뮬레이션학회(SCS) 논문지)지 Editor-In-Chief 역임
- SCS Fellow, 미국 M&S 기술사
- Who's Who in the World (Marguis 16thEdition, 1999) 등재
- 연합사, 국방부/합참, 기품원, ADD, 방사청 자문위원(역임)
- KIDA Fellow 역임

관심분야 : 모델링/시뮬레이션 이론, 방법론 및 환경개발, 시뮬레이터 연동