항공우주 구조물 가상 설계/개발 환경구축을 위한 스트립 분할 탐색 기반 병렬가시화 알고리듬 개발

A Parallel Visualization Algorithm based on Strip-wise Decomposition and Detection for Virtual Design and Development of Aerospace Structures

초 록

본 논문에서는 초대형 과학적 해석결과의 효율적인 병렬 가시화 기법을 제안하였다. 이러한 병렬 가시화 기법은 항공우주분야에서 가까운 미래에 각광받게 될 가상설계환경을 위해 반드시 선행되어야 할 연구이다. 효율적인 해석결과의 후처리를 위해서는 병렬 가시화 과정이 추가적인 비용 없이 해석과정과 유기적으로 연동이 가능하여야 하며, 이를 위해 제안된 알고리듬은 요소영역별 계산을 기반으로 하는 유한요소해석 혹은 전산유체해석의 특성에 적합하도록 고안되었다. 이러한 특성에 부합하고자 후 분류 접근 방식을 사용하였으며, 이미지 조합과정에서 효율적인 네트워크 통신을 위해여 이진 트리구조 통신 방식을 적용하여 구성하였다. 몇 가지 수치예제를 통해, 제안된 알고리듬이 기존의 병렬 가시화 방법에 비해 효율적임을 입증하였다.

Ⅰ. 서 론

모델링과 시뮬레이션 그리고 해석기술은 항공우주분야를 비롯한 여러 분야에서 날로 발전하고 있다. 해석 알고리듬의 비약적인 발전 및 빠르게 변화하는 하드웨어의 발전 속도를 고려할 때, 가상실험 및 가상설계/개발이 조만간 산업현장에서 현실화될 것으로 전망되고 있으며, 이와 더불어 대규모 슈퍼컴퓨터 및 클러스터 시스템을 활용하여 정밀해를 구하기 위한 초정밀 해석에 대한 요구도 지속적으로 증대되고 있다. 특히 최근에는 유지/보수의 용이함으로 인하여 클러스터 시스템이 슈퍼컴퓨터 분야에서 급성장함을 알 수 있는데[1], 이러한 슈퍼컴퓨터를 활용하기 위한 대규모 해석이 세계 각국에서 경쟁적으로 이루어지고 있다. 미국의 경우 ASCI[2] 프로젝트를 수행하고 있으며, 일본의 경우도 Earth Simulator[3,4] 프로젝트를 통해 초대형 해석을 수행하고 있다. 그러므로 급속히 발전하는 슈퍼컴퓨터와 클러스터를 이용한 대규모 초정밀 해석 기술이 향후 기술 경쟁력의 핵심 부분으로서 자리매김할 것으로 예상된다. 이러한 컴퓨팅 기술을 바탕으로, 가상 설계/개발은 초대형 자유도를 갖는 고 충실도 모델의 초정밀 해석 및 실시간 해석과 가시화를 통해 이루어지게 된다. 특히, 위성체나 항공기와 같은 초대형 구조물의 가상 설계/개발환경을 위해서는 이에 적합한 대규모 구조 및 유체 해석을 수행할 수 있는 병렬처리 해석과정과 이에 대한 모델링, 그리고 이러한 대규모 데이터를 실시간으로 가시화 할 수 있는 과학적 가시화 기술의 확보가 우선되어야 한다. 이러한 대규모 해석결과의 후처리를 위해 국외에서는 활발한 연구가 진행되고 있다.

미국의 경우, 스탠포드 대학에서 전산유체역학(CFD)의 해석결과를 고해상도로 구현하기 위해 병렬 가시화 기법을 이용한 wireGL[5]을 개발하여 대규모 CFD 해석결과를 고해상도의 가시화로 구현하였으며, NASA의 Ames연구소 역시 대규모 CFD데이터를 가시화하기 위한 툴킷인 Gel을 개발하였다. 또한, 미 에너지성(Dept. of Energy)은 전략적 컴퓨팅 주도 계획(ASCI)[2]하에 대규모 해석결과를 사용자에게 실시간으로 제공하기 위한 병렬 가시화 기법 연구를 수행하고 있다. 일본의 경우에도 지구전체에 대한 해석을 수행하는 GeoFEM[3,4]의 개발에 있어 대규모 데이터를 가시화하기 위한 병렬 가시화 연구를 수행하였으며, SGI(Silicon Graphics Institute)는 CATIA V5 를 가상설계와 연계시키기 위한 모델링 기법 및 도구의 개발을 현재 진행 중에 있다. 국내에서도 슈퍼컴퓨터 및 대형 클러스터 상에서 초대형 구조해석 문제를 수행하는 등 슈퍼컴퓨팅 기반 병렬 초대형 구조해석[6,7]에 대한 연구가 활발히 진행되고 있다.

위와 같이 가상설계 모델링 및 해석기와 통합 가능한 해법의 성능은 병렬 처리 하드웨

어의 발전과 더불어 그 규모가 나날이 커지고 있으나, 이에 비해 과학적 가시화 기법에 대한 연구는 미진한 실정이다. 효율적 후처리를 통한 과학적 가시화를 위해서는 무엇보다 병렬 해석과정을 통해 얻은 대용량 데이터를 별도의 처리 없이도 가시화할 수 있는 기술의 확보가 필수적이라 할 수 있다. 그러나 현재까지 국외에서 진행된 병렬 가시화 기법은 해석기와 연동성을 고려하지 않은 측면을 가지고 있어, 모델링과 해석 그리고 가시화를 유기적으로 결합하기 어려운 단점을 내포하고 있다. 그러므로 본 논문에서는 항공우주분야의 가상 설계/개발 환경을 위해 해석기와 유기적인 연동이 가능한 병렬 가시화 알고리듬을 개발하였다. 특히 해석기와 유기적인 연동이 가능한 기 개발된 선 탐색 부분 후 분류기법에 비해, 본 논문에서 제안한 알고리듬이 데이터 전송량을 최소화시키고 데이터 통신과 이미지 조합을 동시적으로 수행함으로써, 기존의 알고리듬보다 우수한 성능을 보임을 각종 벤치마킹 수치실험을 통해 입증하였으며, 제안된 기법의 병렬 성능을 다음 3장에서 고찰하였다.

Ⅱ. 병렬 가시화 알고리듬

2.1. 병렬 가시화 기법에 관한 분류

그래픽 병렬 랜더링 기법[8]은 그래픽 파이프라인을 통해 어떠한 데이터를 병렬화 시켰느냐에 따라 전 분류 방식(Sort-First)[8]과 후 분류 방식(Sort- Last)[8]으로 대별된다. 전 분류 방식은 아래의 그림 1과 같이 그래픽 파이프라인 처리 전에 데이터를 화면별로 분할하고 이를 병렬 처리하는 기법이다.

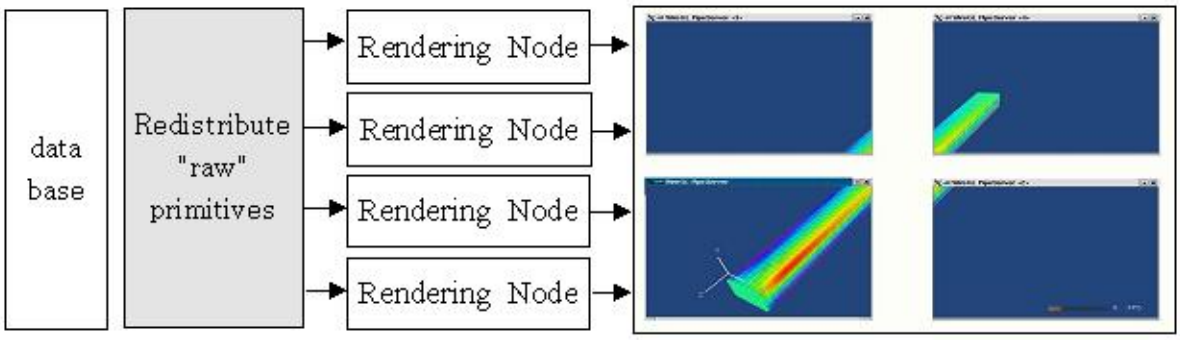

Fig. 1 전 분류 기법 개념도 [wireGL]

전 분류기법은 화면 분할 방식의 렌더링을 통한 병렬처리 방식으로 사용되기 때문에

데이터를 화면 렌더링을 담당하는 병렬 클러스터에 맞게 분산하고, 각 클러스터는 자신의 화면 렌더링만을 담당하여 대규모 데이터에 대한 가시화를 가능하게 하고자 하는 방식이다. 그러므로 화면의 개수에 대한 노드 조정이 자유로워 고해상도의 화면제공에 적합한 방식이다.

그러나 위의 그림 1에서 보듯이 화면에 분산된 데이터의 할당량이 크게 차이를 나타내게 되면 각 노드당 처리하는 그래픽 렌더링 연산 량에 차이를 보여 작업량할당(Load balancing)에 문제가 발생하고, 이는 병렬 효율을 떨어뜨리게 되는 결과를 가져오게 된다. 이러한 문제점을 감소시키고 대규모 유한 요소 모델의 특성을 이용할 수 있는 병렬 가시화 방법이 후 분류방식이다. 후 분류 방식은 그래픽 렌더링 처리 단계의 마지막 결과인 픽셀 데이터를 이용하여 병렬처리를 수행하는 기법이기 때문에 요소분할의 유한 요소 가시화에 적합하며, 병렬 유한요소 해석 시 분할된 영역을 그대로 사용할 수 있는 장점을 가지고 있다.

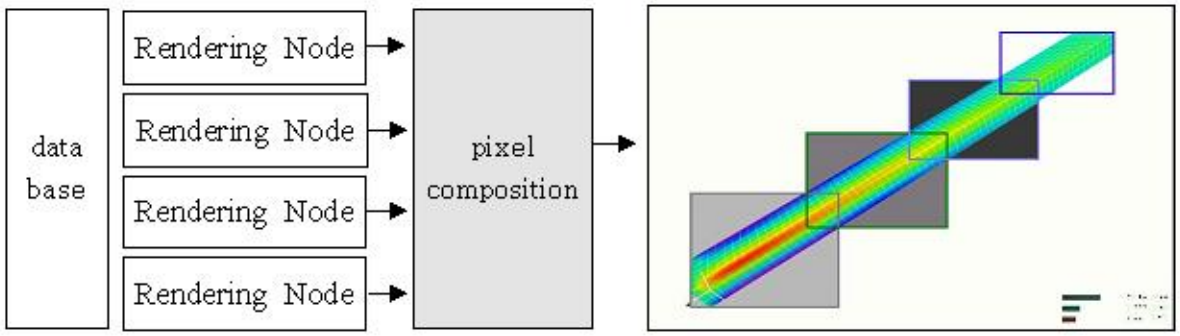

Fig. 2 후 분류기법 적용 개념도

이와 같은 후 분류기법을 적용한 요소분할 렌더링을 사용하면 병렬 유한요소 해석시 분할된 영역을 그대로 사용할 수 있다. 그러므로 병렬 가시화를 위해 추가적으로 영역을 분할하는 부담을 줄일 수 있을 뿐만 아니라, 각 노드마다 자기 해석영역의 해석 결과만 저장하고, 해석데이터를 마스터 노드 등으로 가져올 필요가 없게 되므로, 데이터양을 분산네트워크 환경에서 그대로 접근하여 가시화 할 수 있는 큰 장점을 가질 수 있다. 이러한 후 분류기법은 영역기반 해석을 수행하는 유한요소 해석 및 전산유체 해석에 있어 해석기와 직접적인 연동이 가능한 장점을 가진다. 아래 그림 3은 이러한 연동개념을 도시한 것으로, 추가적인 장비의 도입 및 데이터 분산 알고리듬의 필요없이 직접적인 연동이 가능함을 알 수 있다. 그러나 이러한 후 분류기법이 해석기와의 연동에 있어 장점은 가진다고 하나, 아래 그림에서 보듯이 각기 해석된 데이터의 가시화 결과를 사용자에게 하나의

결과로 보여주기 위한 데이터 조합과정이 들어감을 확인할 수 있다.

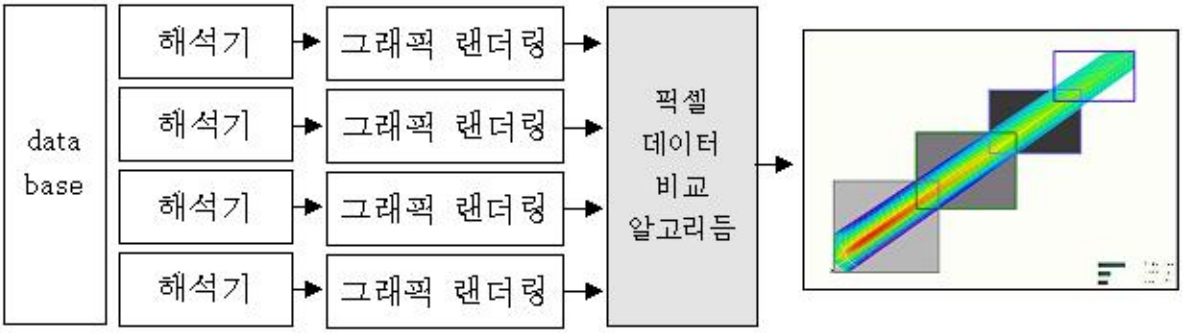

Fig. 3 후 분류기법의 병렬 해석기와의 연동개념

아래의 그림 4는 FA-18전투기의 JDAM 분리에 따른 CFD 해석결과를 후 분류 기법으로 병렬 가시화한 예제이다.

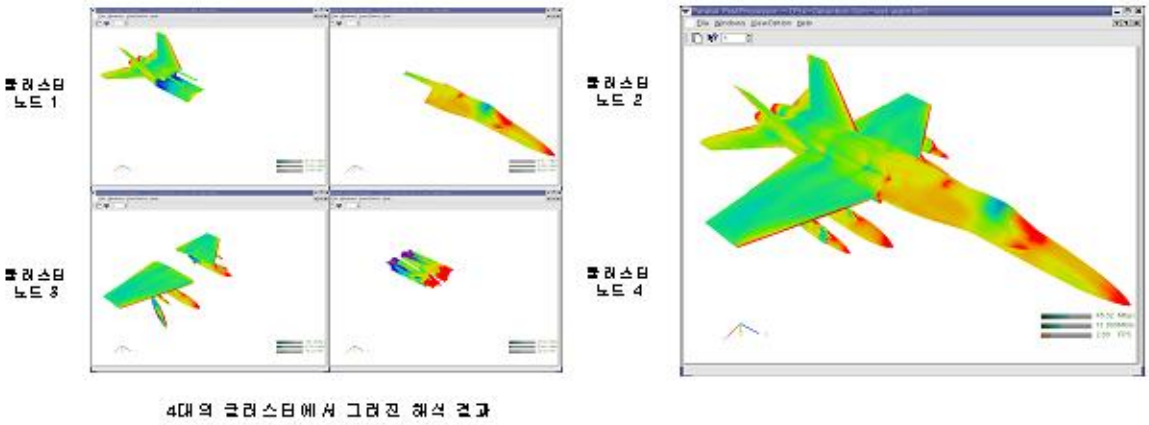

Fig. 4 해석 결과를 하나의 노드로 조합한 결과

위의 예제는 앞에서 설명한 후 분류기법을 사용하여 각 렌더링 노드에서의 가시화 결과를 하나의 단일 화면으로 재조합하여 보여주고 있다. 각각의 클러스터 노드에서 해석된 해석 데이터를 그대로 가시화하고 이를 픽셀 데이터 조합을 통해 전체적인 해석결과를 보여줌을 확인할 수 있다.

2. 기존 가시화 알고리즘의 문제점

병렬 유한요소 해석 결과 가시화에 적합한 후 분류 방식은 전체 후 분류 기법(Sort last full)과 부분 후 분류 기법(Sort last sparse)[8], 선 탐색 부분 후 분류 기법(Pre-Detection Sort last sparse, PDSL)[9] 등으로 구분된다.

아래 그림은 전체 후 부분기법을 보여 주고 있다. 이 기법은 각각의 클러스터 노드에서

생성된 전체 이미지를 저장하고, 저장된 전체 이미지를 마스터 노드에 보내어 단일 이미지로 구성하게 된다. 전체 화면에 대한 픽셀정보를 전송하게 되어, 많은 데이터 전송에 따른 통신량 병목현상을 유발할 수 있다. 특히, 유한요소 가시화와 같이 각각의 클러스터 노드가 전체 화면에 대한 가시화가 아닌, 요소단위의 부분화면에 대한 가시화만을 필요로 하는 경우, Z-buffer[10] 깊이비교시간이 증가되어, 성능 저하가 발생할 수 있다.

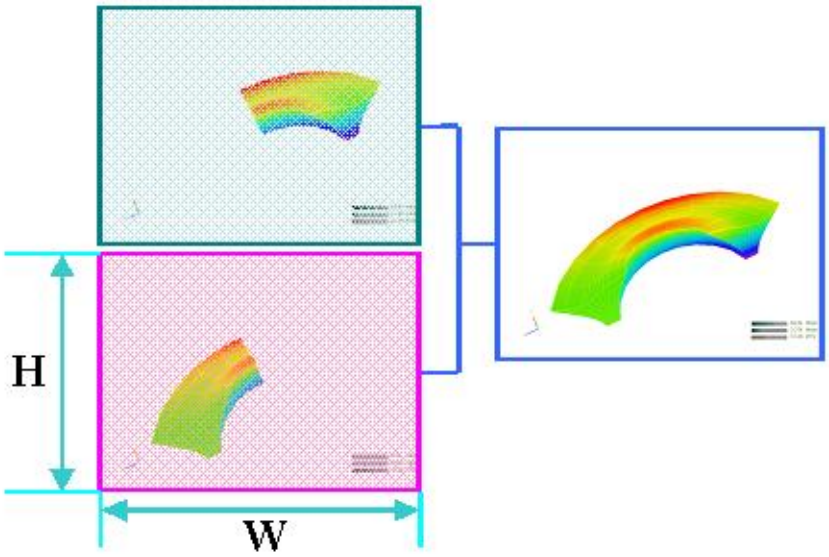

Fig. 5 전체 후 분류 기법

전체 후 분류기법(Sort-last full method)은 다음과 같이 N대의 클러스터 노드(CPU 개수 N) 사용 시 데이터 통신량 및 깊이 검사 비교 시간을 분석할 수 있다.

■ 전체 후 분류기법(Sort-last full method) 효율 분석

- 데이터 통신량 : $\sum_{i}^{N}(\Gamma \times W \times H)$
- i 번째 깊이 검사 영역 : $W \times H$

▸ N : 클러스터 수 [CPU 개수]
▸ Γ : 픽셀 데이터의 크기 [RGBA, Z-buffer depth]
▸ W : 전체 이미지의 폭
▸ H : 전체 이미지의 높이

이러한 경우 아래 그림 6과 같은 부분 후 분류기법(Sort-last sparse)을 사용하면, 부분화면에 대한 깊이비교 소요시간과 통신량 병목현상을 일부 감소시킬 수 있다. 클러스터 노드1은 실제 유효한 부분 이미지만을 마스터 노드에 보내게 되고, 마스터 노드는 클러스터 노드 1,2에서 보내온 이미지에 대하여 깊이 검사를 수행하게 된다.

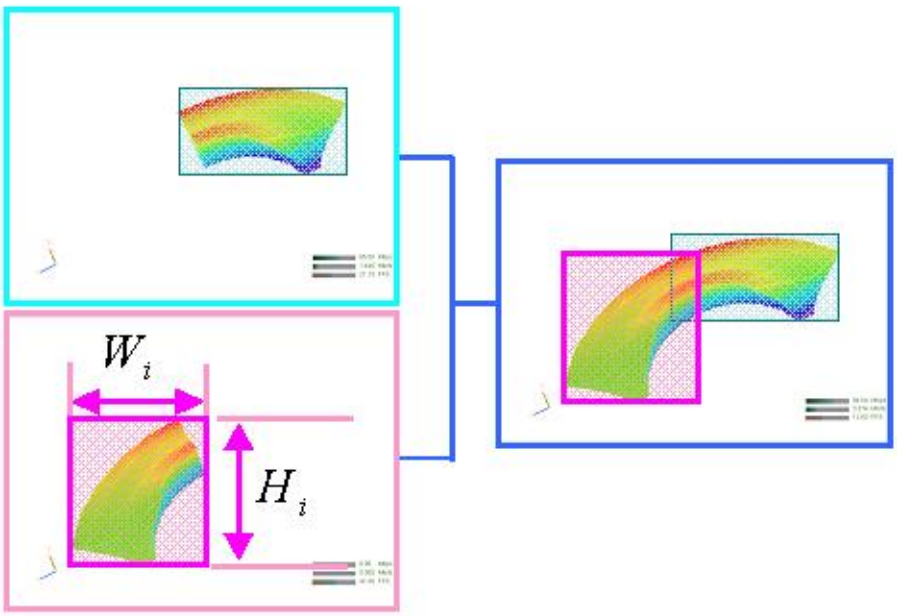

Fig. 6 부분 후 분류 기법

그러나 위의 그림 6의 부분 후 분류기법을 이용하면 마스터 노드는 유효 픽셀 영역 전체에 대한 깊이비교를 수행하기 때문에, 실제 겹치는 부분보다 더 많은 부분에 대해 깊이비교를 수행하게 되고, 그만큼 소요시간이 증가하게 된다. 부분 후 분류기법은 다음과 같이 N대의 클러스터 노드(CPU 개수 N) 사용 시 데이터 통신량 및 깊이 검사 비교 시간을 분석할 수 있다.

아래의 결과에서 알 수 있듯이, 전체 후 분류기법 보다 부분 후 분류기법이 데이터 통신량이나 깊이 검사 영역에서 작은 값을 나타내어 연산 량이 적음을 알 수 있다. 선 탐색 과정을 통한 부분 후 분류기법은 아래 그림 7과 같이 각각의 렌더링 노드에서 그려진 그림의 픽셀위치 정보를 깊이 비교 이전에 파악하여 겹치는 부분에 대한 정보를 미리 얻는 것으로 비교영역이 줄어들어 깊이 비교 소요시간을 감소시키고, 이를 통해 병렬성능을 향상시키는 방법이다.

■ 부분 후 분류기법(Sort-last sparse method) 효율 분석

- 데이터 통신량 : $\sum_{i}^{N}(\Gamma \times W_i \times H_i)$
- i 번째 깊이 검사 영역 : $W_i \times H_i$

- N : 클러스터 수 [CPU 개수]
- Γ : 픽셀 데이터의 크기 [RGBA, Z-buffer depth]
- W : 이미지의 폭
- H : 이미지의 높이
- X^i_{max} : i 번째 노드 이미지 x 좌표 값의 최대값
- X^i_{min} : i 번째 노드 이미지 x 좌표 값의 최소값
- Y^i_{max} : i 번째 노드 이미지 y 좌표 값의 최대값
- Y^i_{min} : i 번째 노드 이미지 y 좌표 값의 최소값
- W_i : i 번째 노드 이미지 폭 $X^i_{max} - X^i_{min} = W_i \le W$
- H_i : i 번째 노드 이미지 높이 $Y^i_{max} - Y^i_{min} = H_i \le H$

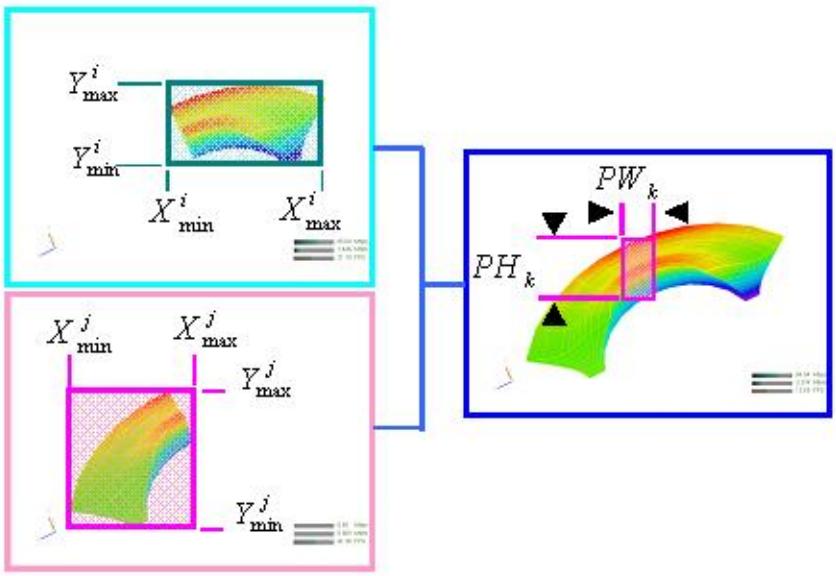

Fig. 7 선 탐색 부분 후 분류 기법

위의 그림을 통해 전송 데이터양은 부분 후 분류기법과 같더라도 깊이 비교 영역에서 적은 연산량을 보이는 것을 알 수 있다. 다음은 선 탐색 부분 후 분류기법의 성능을 분석한 것이다.

■ 선 탐색 부분 후 분류기법(Pre-Detection Sort-Last sparse algorithm) 효율 분석

- 데이터 통신량 : $\sum_{i}^{N}(\Gamma \times W_i \times H_i)$
- i 번째 깊이 검사 영역 : $PW_i \times PH_i$

- N : 클러스터 수
- Γ : 픽셀 데이터의 크기 [RGBA, Z-buffer depth]
- W : 이미지의 폭
- H : 이미지의 높이
- W_i : i 번째 노드 이미지 폭 $X^i_{\max} - X^i_{\min} = W_i \le W$
- H_i : i 번째 노드 이미지 높이 $Y^i_{\max} - Y^i_{\min} = H_i \le H$
- PW_k : $\min[X^i_{\max}, X^j_{\max}] - \max[X^i_{\min}, X^j_{\min}]$ $PW_k \le W_k \le W$
- PH_k : $\min[Y^i_{\max}, Y^j_{\max}] - \max[Y^i_{\min}, Y^j_{\min}]$ $PH_k \le H_k \le H$

병렬 유한 요소 가시화에서는 각 노드마다 해석결과에 대한 부분 데이터만을 가시화하기 때문에 실제 겹치는 부분의 깊이 검사영역이 전체 유효 픽셀영역에 비해 작기 때문에, 실제 겹치는 부분에 대한 깊이 비교만을 수행하게 되면, 병렬 효율을 높일 수 있다. 그러나 위의 가시화 기법들에서 보듯이 그려진 해석결과를 단일 사각형만으로 전송하기 때문에 복잡한 형상의 경우, 실제 형상에는 관계없는 불필요한 데이터가 전송되게 되며 이는 해석 결과의 형상이 복잡할수록, 또한 데이터가 많을수록 병렬 효율을 떨어뜨리는 원인으로 작용하게 된다. 이에 본 연구에서는 불필요한 데이터 전송을 최소화하기 위해 이미지 데이터를 스트림 형태로 분할시켜 전송하고, 전송과 동시에 이미지 조합을 수행할 수 있는 스트림 분할 탐색 알고리듬을 제안하였다.

3. 스트립 분할 탐색 병렬 가시화 알고리듬

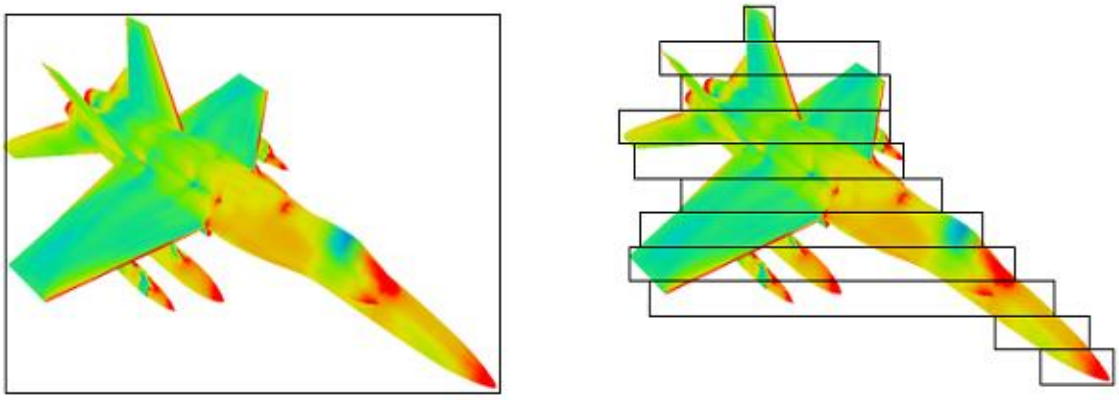

Fig. 8 스트립 분할 탐색 알고리듬 개요도

위의 그림 8은 단일 사각형으로 탐색된 영역과 스트립 형태로 탐색된 영역을 비교 도시한 것으로서 단일 사각형에 비해 스트립 형태로 탐색을 수행할 경우 불필요한 데이터를 최소화 시킬 수 있음을 알 수 있다. 따라서 복잡한 모델일수록 스트립 형태의 탐색을 수행하는 것이 보다 병렬 가시화에 있어 효율적임을 알 수 있다. 다음 도시된 그림 9를 통해 앞에서 설명했던 기존 가시화 알고리즘과의 차이점을 볼 수 있다.

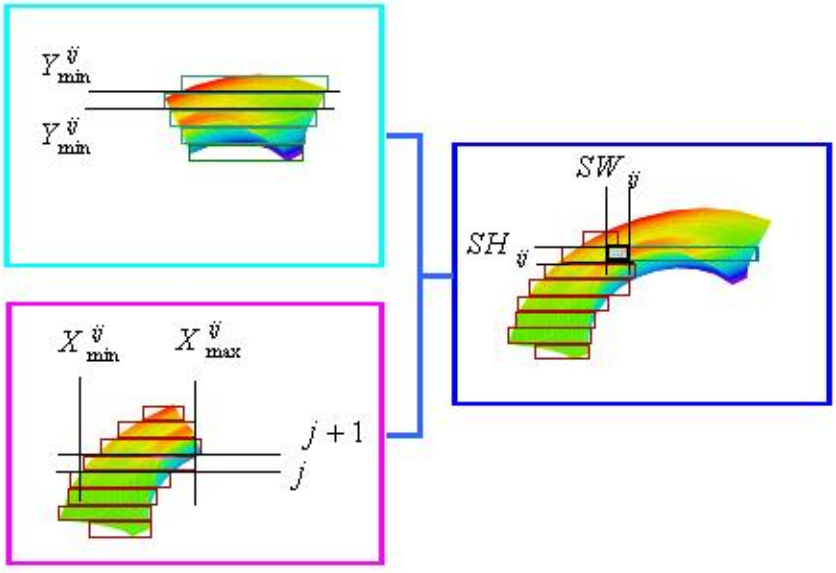

Fig. 9 스트립 형 분할 기법

따라서 위와 같은 스트립 분할 형태의 알고리즘을 이용하면 각 노드에서 해석한 가시화 결과를 마스터 노드로 전송하는데 있어 데이터양이 줄어들게 되며, 이로 인해 깊이 검사 영역과 이미지 조합 과정의 시간을 단축할 수 있다. 다음은 스트립 분할 후 분류기법의 효율을 분석한 것이다.

■ 스트립 분할 후 분류기법(Strip-wise decomposition Sort-Last sparse algorithm) 효율 분석

- 데이터 통신량 : $\sum_{i}^{N}\sum_{j}^{S} \{ (\Gamma \times W_{ij} \times H_{ij}) \times f \}$
- i 번째 노드 j 번째 스트립의 깊이 검사 영역 : $SW_{ij} \times SH_{ij}$

- N : 클러스터 수
- S : 스트립의 수
- Γ : 픽셀 데이터의 크기 [RGBA, Z-buffer depth]
- f : 최적화 성능을 위한 픽셀 라인수의 비율
- W : 이미지의 폭
- H : 이미지의 높이
- SW_{ij} : 겹쳐지는 부분의 스트립 폭
- SH_{ij} : 겹쳐지는 부분의 스트립 높이

위의 분석결과로부터 알 수 있듯이, 기존의 개발된 알고리듬에 비해 최소의 데이터 전송량을 보임을 확인할 수 있다. 위의 분석결과 중 최적화 성능을 위한 픽셀 라인수의 비율은 스트립 분할 탐색 알고리듬의 특성으로 분할시킬 스트립의 수를 조정하여 성능을 향상시킬 수 있는 요인으로 작용한다. 이러한 최적화를 위한 라인수 조절에 대한 결과는 다음 4장 성능평가에서 픽셀 라인수를 조정하여 성능을 향상시킨 것으로부터 확인할 수 있다.

본 논문에서 제안된 스트립 분할 후 분류기법은 기존의 알고리듬을 모두 포함할 수 있다. 예를 들어 스트립의 수를 1로 가정하면 그에 대한 성능 비율은 1이 되고, 이는 기존의 선 탐색 부분 후 분류기법과 같은 결과가 유도됨을 확인할 수 있다. 또한 이미지의 깊이 검사영역을 부분 이미지에 대해 수행하게 되면 기존의 부분 후 분류기법과 같은 성능을 나타내게 되고 검사영역을 전체 이미지로 확대하게 되면, 전체 후 분류기법과 같은 성

능을 나타낼 수 있음을 알 수 있다. 그러므로 본 논문에서 분석하고 제안된 알고리듬은 기존의 알고리듬을 대체할 수 있고, 또한 성능을 크게 향상시킬 수 있는 알고리듬이다.

위의 수식을 기존 알고리듬과 비교한 결과로부터 다음과 같은 사항을 알 수 있다.

■ 데이터 전송량

스트립 분할 탐색 부분후 분류 기법 < 선 탐색 부분 후 분류기법
= 부분 후 분류기법 ≤ 전체 후 분류기법

SDSL < PDSL = SL-sparse ≤ SL-full

■ 깊이 비교 영역에 따른 연산량

스트립 분할 탐색 부분후 분류 기법 < 선 탐색 부분 후 분류기법
≤ 부분 후 분류기법 ≤ 전체 후 분류기법

SDSL < PDSL ≤ SL-sparse ≤ SL-full

그러므로 스트립 분할 탐색 부분 후 분류기법(SDSL)은 기존의 후 분류기법에 비해 데이터 전송량 및 깊이 비교에 따른 연산량이 적어 병렬 효율을 높일 수 있다. 아래의 그림 10은 로켓의 해석 결과를 8개의 노드에 분산하여 병렬 가시화 한 예제이다.

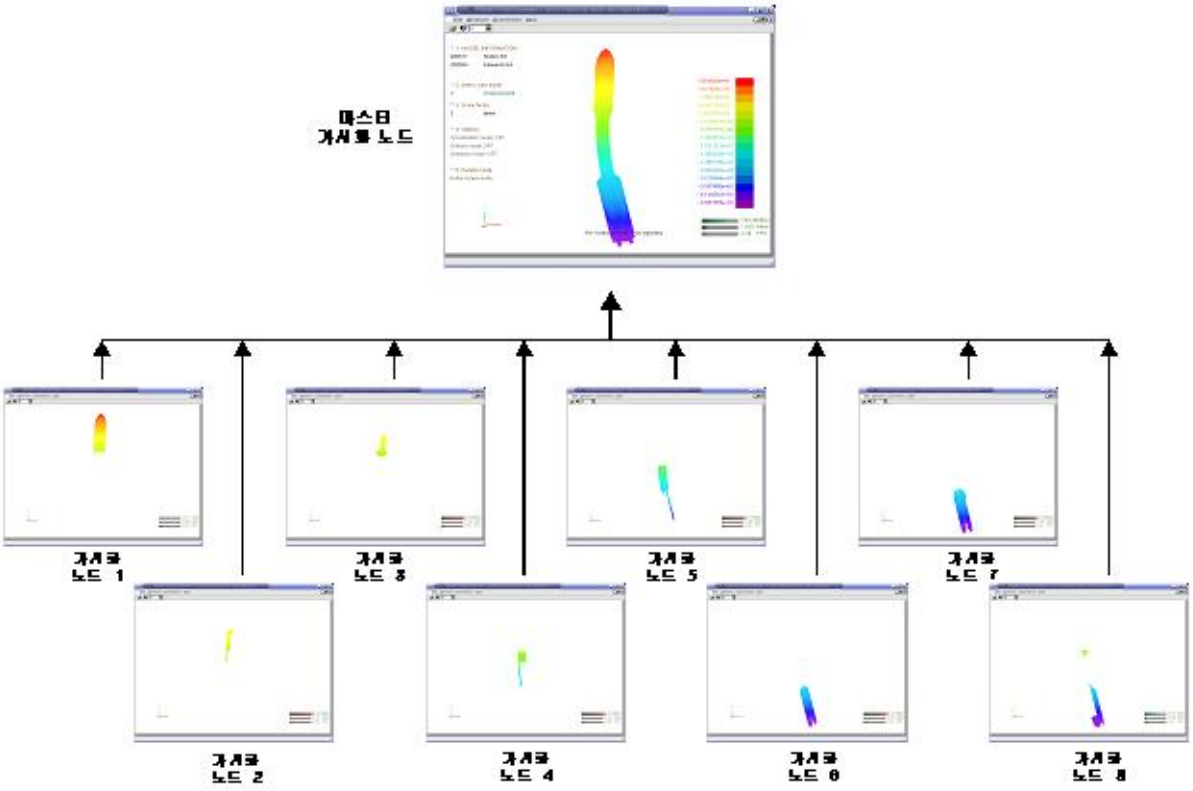

Fig. 10 로켓 해석 결과 가시화

8개의 각 노드에서 해석된 결과는 스트림 형태의 탐색 법으로 하나의 마스터 노드로 전송되어 해석 결과 전체를 보여주게 된다. 다음의 그림11의 결과는 위의 그림10과 같은 로켓 모델의 가시화를 통해 얻은 각각의 가시화 알고리듬의 병렬 효율을 나타낸 것이다.

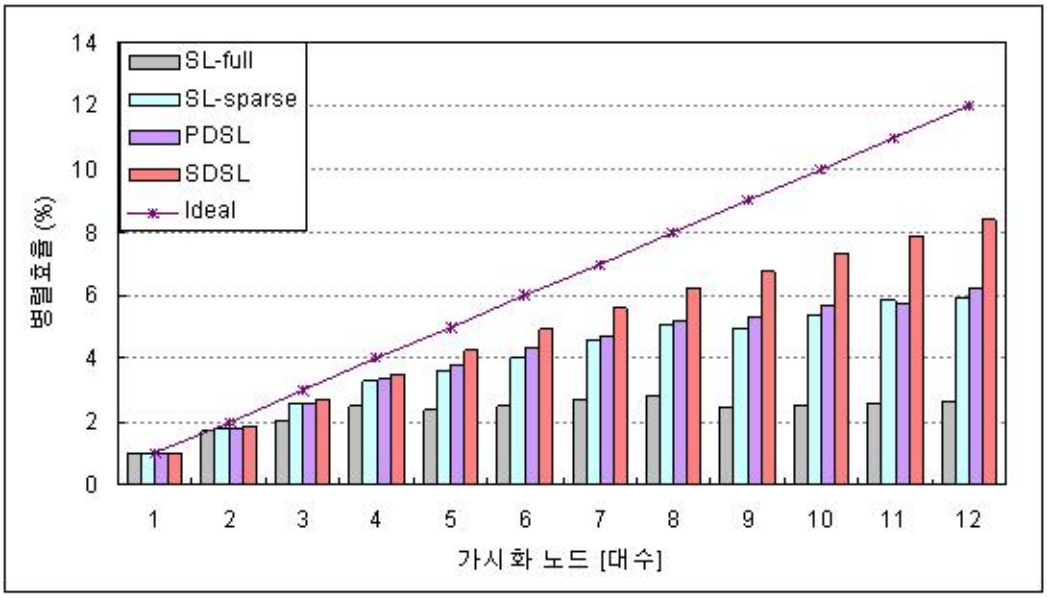

Fig. 11 병렬 가시화 알고리듬에 따른 병렬 효율 비교

위의 결과를 살펴보면 전체 후 분류 기법(SL-full) 이 가장 낮은 병렬 효율을 나타내고 스트림 분할 탐색 기법(SDSL)이 가장 높은 효율을 나타내고 있다. 또한 부분 후 분류 기법(SL-sparse)과 선 탐색 부분 후 분류 기법(PDSL)은 거의 유사한 병렬 효율을 나타내고 있다. 이러한 결과로부터 전체 후 분류 기법의 경우 화면 전체에 해당하는 이미지 전송과 깊이 비교로 병렬 성능이 떨어져 가시화 노드의 증가에 따른 성능향상을 얻을 수 없음을 알 수 있고, 부분 후 분류기법과 선 탐색 부분 후 분류기법의 경우, 전체 후 분류기법에 비해 네트워크 전송량 및 깊이검사 소요시간이 감소하여 성능 향상은 발휘되나, 이 역시 노드 증가에 따른 성능 감소가 급격히 나타남을 알 수 있다. 그러나 본 논문에서 제안한 스트림형 부분 후 분류기법의 경우 기존의 알고리듬에 비해 성능증가가 지속적으로 나타남으로써 기존의 알고리듬에 비해 성능이 우수함을 알 수 있다. 위와 같은 병렬 효율 차이의 주된 원인은 앞서 설명한 바와 같이 실제 유효한 해석결과의 이미지 전송에 따른 네트워크 전송량에 차이를 보여 성능 차이가 나타난 것인데, 이는 다음 그림 12의 네트워크 통신량 비교를 통해 확인할 수 있다.

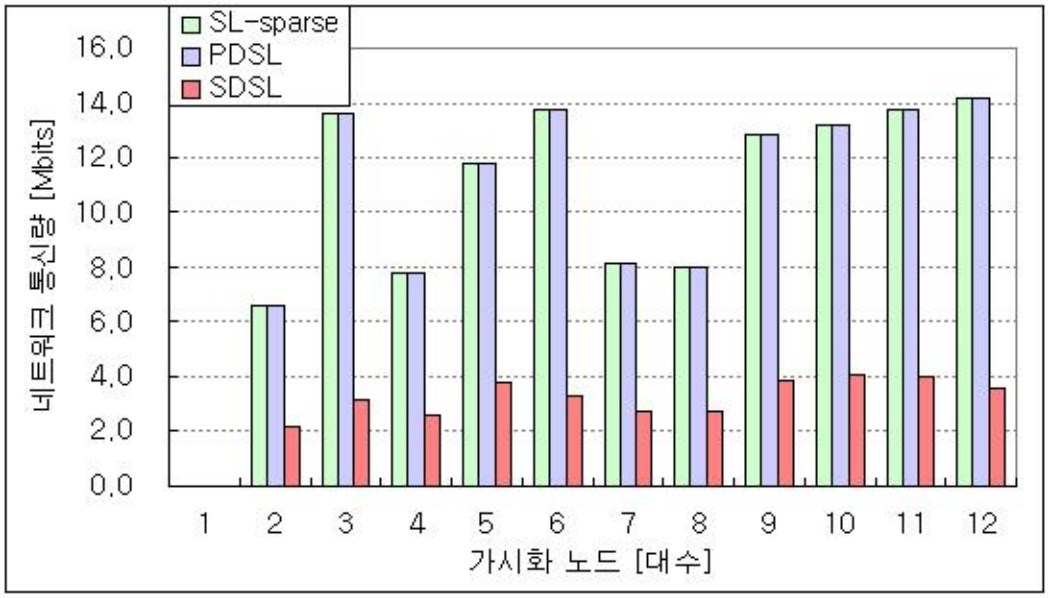

Fig. 12 네트워크 최대 통신량 비교

위의 그림12는 다음 4장에서 설명할 이진트리구조 통신알고리듬을 사용하여 측정한 결과이고 네트워크 최대 통신량을 표시하였다. 위의 결과를 통해, 부분 후 분류기법과 선탐색 부분 후 분류기법의 통신량은 동일하고 스트림 분할 부분 후 분류기법의 경우 네트워크 통신량이 다른 알고리듬에 비해 현격히 적음을 확인할 수 있다. 전체 후 분류기법의 경우 통신량이 다른 알고리듬에 비해 월등히 크게 나타나 위의 그림 12의 결과에서는 표시하지는 않았으나, 평균 27Mbits의 전송량을 나타내었다. 이러한 결과는 앞에서 설명한 바와 같이 스트림형 부분 후 분류기법의 경우 해석결과의 부분 이미지 전송이 아닌 스트림으로 분할한 유효 픽셀의 이미지만을 전송하기 때문에 나타난 결과이다. 스트림형 부분 후 분류기법은 기존의 가시화 기법에 비해 데이터 전송량 및 깊이 검사영역을 감소시키고 이를 통해 병렬 효율을 높일 수 있는 알고리듬이다.

Ⅲ. 스트립 분할 부분 후 분류 알고리듬 성능평가

3.1. 알고리듬 성능평가 방법

성능평가는 랜더링 시간, 네트워크 전송 데이터 량에 대해 수행되었다. 각 측정시간은 모델을 360도 자동으로 회전시켜가면서 총 24장의 그림을 가시화할 때 걸린 시간으로 정하였다.

이렇게 측정한 이유는 같은 모델이라 할지라도 모델을 보는 각도에 따라 픽셀 데이터 전송량과 깊이비교 시간이 변화할 수 있고 이에 따라 성능지수가 변화될 수 있기 때문이다. 그러므로 본 연구에서 측정된 랜더링 시간 및 네트워크 전송 데이터 측정량은 특정 화면각도에서의 성능이 아니라 평균성능을 의미한다고 할 수 있다.

가시화용 컴퓨터 노드 4대를 사용하여 그림 13의 모델 A, B를 정지화면의 다른 각도에서 묘사할 경우에 대한 시간을 측정하면 그림 14와 같은 결과를 얻을 수 있으며, 이로부터 전술한 바와 같이 모델을 묘사하는 각도에 따라 성능이 달리 나타날 수 있음을 재확인할 수 있다.

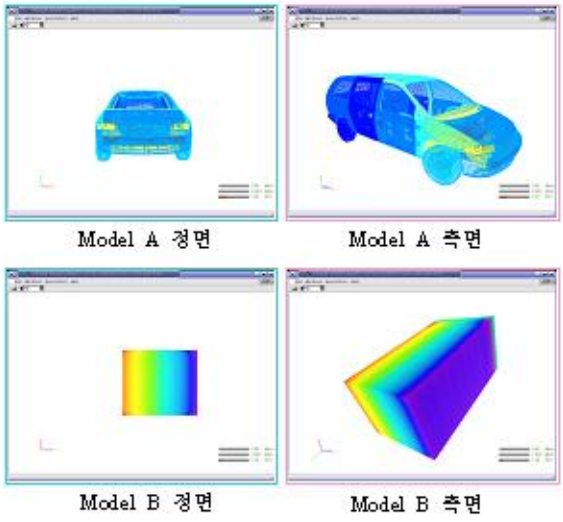

Fig. 13. 성능 측정을 위한 유한요소 모델

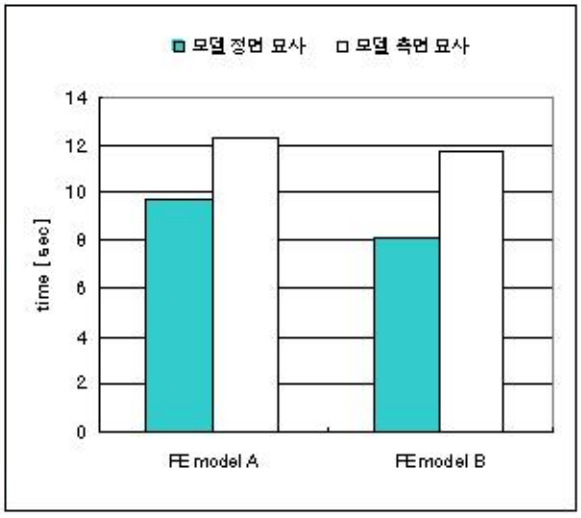

Fig. 14. 모델 묘사방향에 따른 렌더링 시간측정결과

그러므로 본 논문에서 채택된 평균적 성능평가 방법이 특정 정지화면에 대한 가시화 성능을 측정하는 것보다 더욱 합리적임을 확인할 수 있다.

3.2. 알고리듬 성능 평가를 위한 네트워크 통신 구조 설정 및 시스템 환경

알고리듬의 성능 평가는 부분 후 분류방식과 이진 트리통신 구조를 적용하여 구성된 병렬 가시화 알고리듬을 다음 테이블 1에 나타낸 시스템 환경 하에서 구현하였다. 각 가시화 노드에서 생성된 부분 이미지들의 조합을 위해서는 데이터 통신이 필수적으로 수반되며, 이러한 통신량은 노드 수에 비례하여 증가하게 된다. 그러므로 병렬가시화 성능을 최대한 제고하기 위해서는 효율적인 통신패턴 구조를 사용하여야 한다. 본 논문에서는 이러한 병렬효율에 보다 효과적으로 통신시간을 감소시키기 위해서 이진 트리구조 통신패턴을 적용하였다. 순차적 통신 패턴을 사용하면 통신시간이 노드 개수 N에 선형적으로 비례하여 증가되지만 이진 트리구조 통신패턴을 사용하게 되면 동시적 통신을 가능하게 하여 이론적으로는 통신시간을 $\log_2(N)$에 비례하는 수준으로 감소시킬 수 있으며 병목현상을 최소화 할 수 있다.

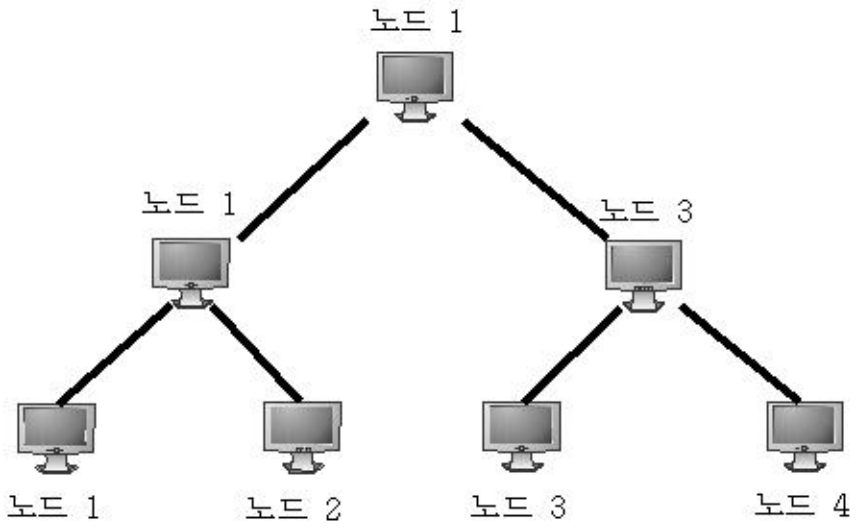

Fig. 15 이진 트리구조 통신 알고리듬의 적용

■ **이진 트리구조 통신 알고리듬의 적용**
- **노드간 단일 통신시간은 T로 일정하게 유지된다고 가정**
- **전체 노드개수 : N**
- **$M=\min\{x \mid x=2^a \geq N,\ a=1,2,3,\cdots\}$**
- **통신시간 : $T\ \log_2(M)$**

아래의 그림 16을 통해 본 논문에서 적용한 이진 트리구조 통신 패턴을 확인할 수 있다.

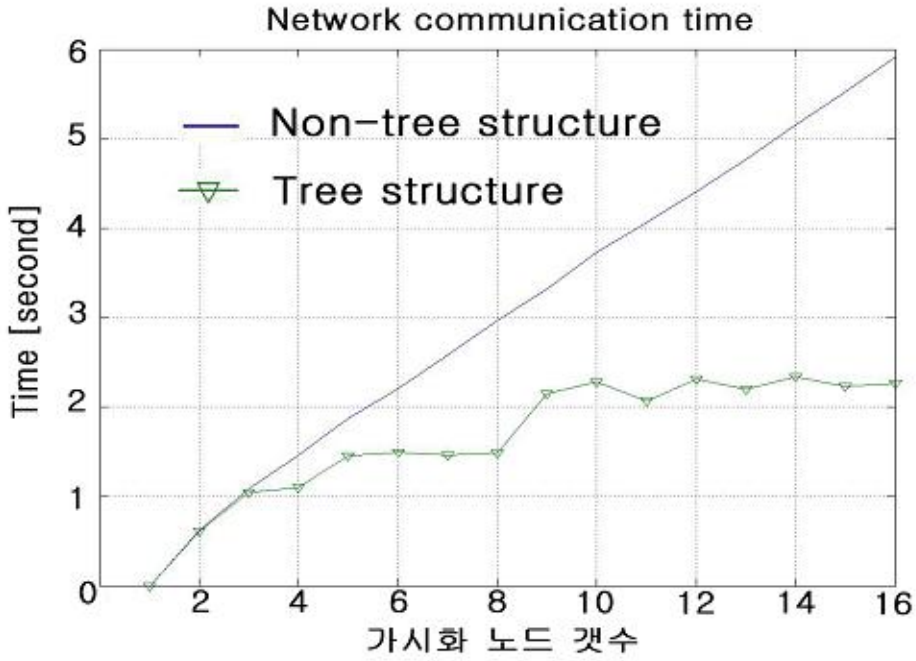

Fig. 16. 이진 트리 통신패턴과 순차적 통신패턴 사용시 통신 시간 측정

아래의 표1,2와 같은 시스템 환경은 병렬 해석을 수행할 수 있는 일반적인 저가의 분산 메모리 병렬 계산 환경이다. 본 연구에서는 이러한 환경에 적합한 병렬 가시화 소프트웨어를 개발하여 고가의 그래픽 서버나 추가의 가시화용 컴퓨터 없이 기존의 병렬해석용으로 사용되고 있는 저가의 분산 메모리 병렬계산 시스템만으로도 대용량 해석결과의 병렬 가시화가 가능할 수 있도록 하였으며, 이를 통해 해석과정과 해석결과의 가시화 과정을 동일한 시스템 상에서 원활하게 연동시킬 수 있도록 하였다.

Table. I 운용 하드웨어 시스템 환경

분 류	내 용
Cluster 사용목적	일반 목적 분산메모리 병렬처리용
CPU 사용 갯수	운용에 따라 가변
C P U	P4 2.4GHz
Graphic card	Geforce FX 5700
Memory	1 GByte/node
Network card	1 GBit / node
Switching HUB	1GBit HUB

Table.II 운용 소프트웨어 시스템 환경

분 류	내 용
Operating System	Linux
Kernel version	2.4.18-15hl
compiler version	gcc-3.2
QT	3.0.5
lam mpi	6.5.6
openGL	1.3 (glut 포함)

성능 평가를 위해 위의 그림 11,12에서 확인하였듯이, 기존의 알고리듬 중 랜더링 시간과 네트워크 전송량이 최소인 PDSL과 SDSL의 렌더링 시간과 네트워크 전송 데이터량 그리고 병렬 효율을 측정하여 비교하였다.

3.3. 스트립 분할 부분 후 분류 알고리듬 최적 픽셀 라인수 비교

위의 2절 3장에서 살펴보았듯이, 스트립 분할 부분 후 분류 알고리듬은 스트립 라인을 조절함으로써, 병렬 가시화 성능을 조절할 수 있는 장점을 가진다. 이는 픽셀 라인수를 조절함에 따라 유효영역의 탐색에 차이가 생기고, 이에 따라 네트워크 전송량 및 깊이 비교영역에 차이가 남으로써 나타나는 특성인데, 위에서 언급하였듯이 이러한 특성으로 기존의 병렬 가시화 알고리듬을 대체할 수 있고 최적의 스트립 라인을 찾음으로써 최적의 성능을 발휘할 수 있다.

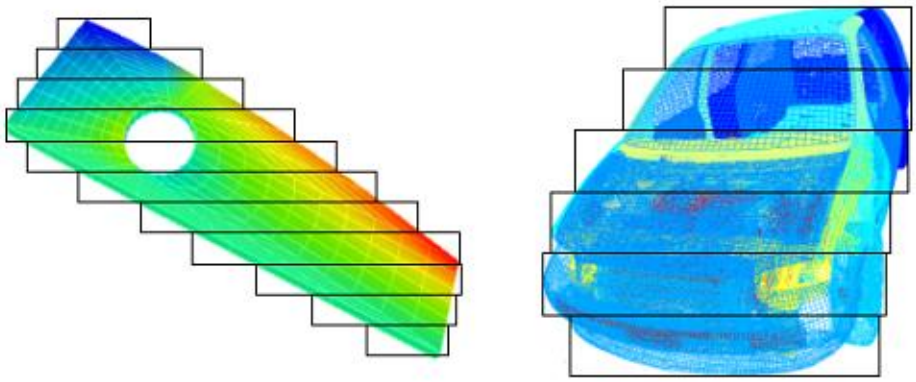

Fig. 17 픽셀 라인 수에 따른 스트립 형태

위의 그림 17은 각기 다른 모델에 따른 스트림 형태를 적용한 예이다. 형상에 따라 한 개의 스트림을 이루는 픽셀 라인 수가 다르게 적용될 수 있음을 보여주고 있다.

아래의 그림 18의 모델은 유한요소 100만개의 절점으로 이루어진 HEXA 모델이다. 해상도 800×600 과 1024×768 두 경우에 대해 픽셀 라인 수에 따른 최적화된 성능을 테스트 하였다.

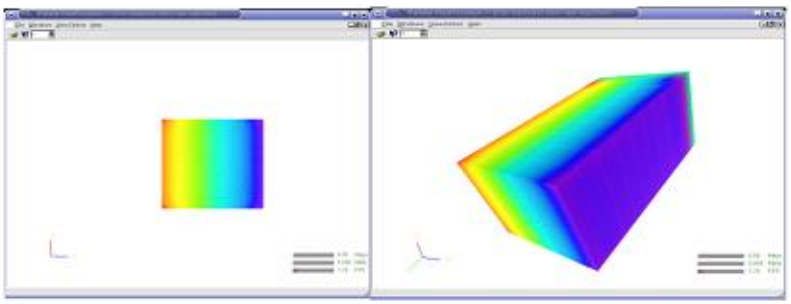

Fig. 18 100 만 HEXA 모델

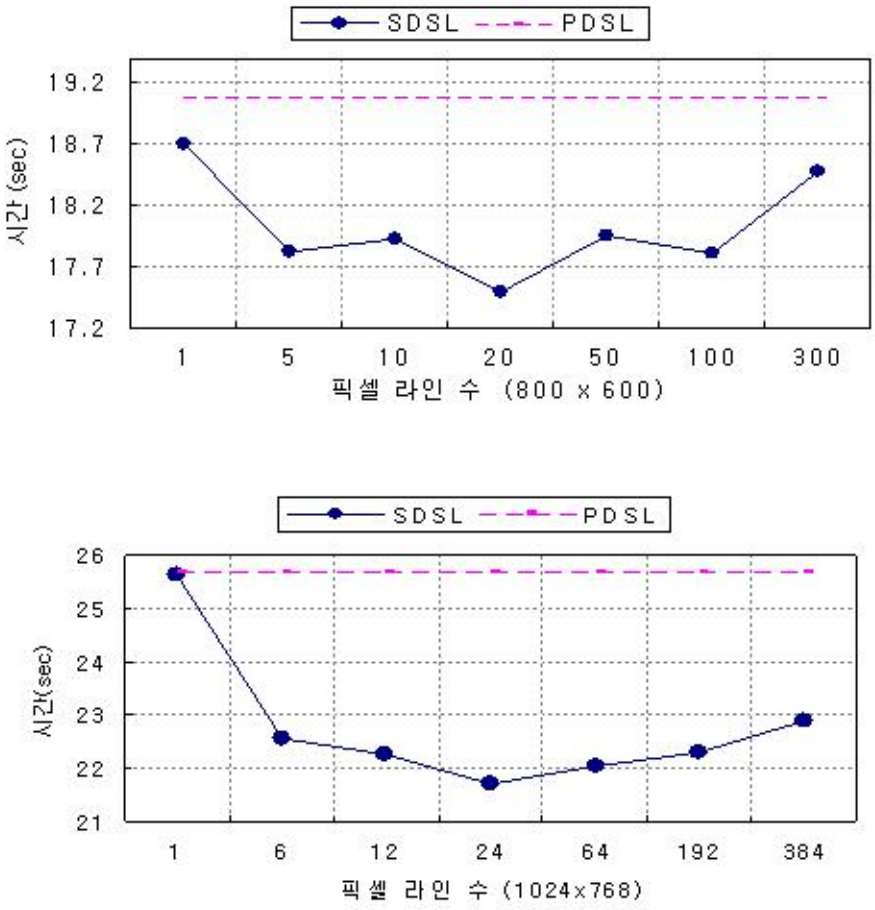

Fig. 19 해상도와 픽셀 라인 수에 따른 시간 비교

위의 그림 19와 같이 특정 픽셀의 라인에서 가장 빠른 속도를 보였다. 이러한 결과는 네트워크 전송량을 조절 가능하게 함으로써 최적화된 네트워크 전송량을 알 수 있으며 이를 통해 최적화된 병렬 가시화 성능을 낼 수 있음을 의미한다. 이와 같은 결과는 전송되는 데이터의 량을 줄이기 위해 라인수를 적게 할시 스트림 탐색에 걸리는 연산시간이 많게 되고, 스트림 탐색을 위한 연산시간을 줄이기 위해 라인수를 많게 할시 전송되는 데이터의 량이 증가하게 되어 나타난 결과이다. 즉, CPU에 의한 연산 시간과 네트워크의 대역폭에 따른 데이터 전송시간과의 최적한 접점을 찾을 수 있음을 의미한다.

위와 같은 수치실험을 통해 최적의 라인수를 예측할 수 있는데, 이러한 최적의 가시화 속도를 얻을 수 있는 스트림의 픽셀 라인 수는 해상도에 관계없이 전체 해상도 세로 픽셀의 약 3%에 해당된다. 이는 최적화된 성능을 얻기 위한 특정 픽셀 라인 수를 사전에 예측할 수 있음을 의미한다. 위 결과로부터 예측된 최적 픽셀 수는 아래와 같다.

최적화 성능을 위한 픽셀 라인수 = 해상도 세로 픽셀 수 X f

(f = 최적 라인 수의 비율　예; 600 X 3% = 18 , 768 X 3% = 23)

3.4. 알고리듬에 따른 성능 비교 및 평가

PDSL 과 SDSL 알고리듬의 성능에 대한 비교 및 이에 대한 평가를 수행하였다. 병렬 가시화 성능 시험은 속도 증가율(Speed-up)과 병렬 효율측정을 통해 수행되었으며, 다음 수식 (1), (2)를 통해 계산되었다. 성능 평가를 위한 해석결과는 유한요소 모델 및 해석결과를 사용하였다.

□ 속도 증가율 (Speed-up) = $s(n)$

$$s(n) = \frac{\text{순차프로그램의 실행시간}}{\text{병렬프로그램의 실행시간}(n\text{개 노드})} \quad ①$$

□ 병렬 효율 (Efficiency) = $E(n)$

$$E(n) = \frac{S(n)}{n}[\times 100(\%)] \quad ②$$

3.4.1. 유한요소 해석결과를 이용한 성능 평가 모델 ①

아래의 모델은 인공위성의 유한요소 해석결과로, 4대의 가시화 노드를 사용할 시 다음 그림 20과 같이 분산된 데이터 형태 및 가시화 결과를 확인할 수 있다.

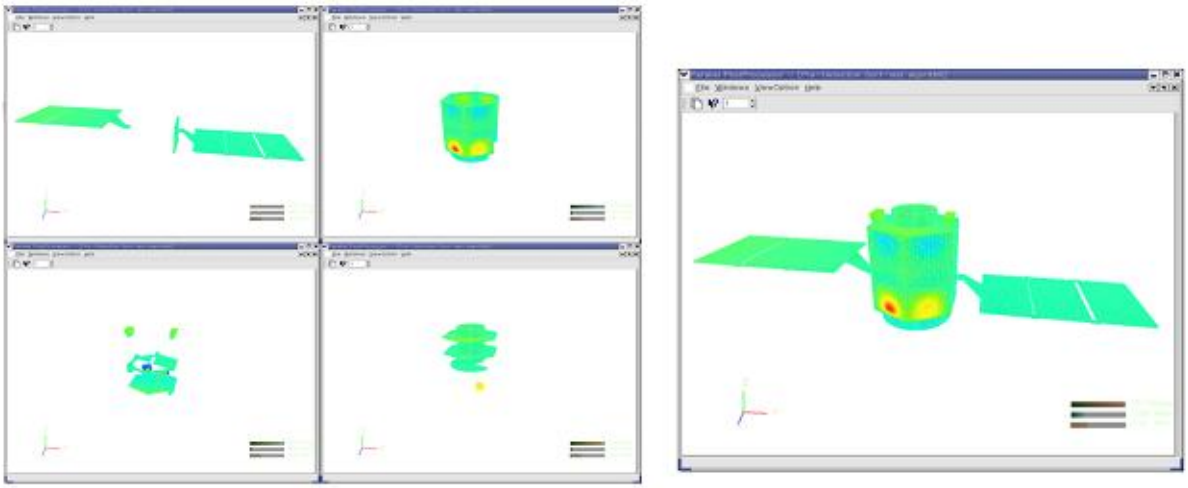

Fig. 20 4대의 노드에 분산된 인공위성 해석결과의 병렬 가시화 예제 ①

제안된 스트림 분할 탐색 알고리듬을 이용하여 인공위성 모델을 1Gbps 네트워크 환경에서 해상도 800×600으로 가시화 하였으며, 이 경우 제안된 기법의 데이터 전송량과 기존 PDSL기법의 데이터 전송량을 그림 21에서 확인할 수 있다. 아래의 그림 21 에서 알 수 있듯이 스트림 기법을 이용한 경우, PDSL기법 보다 네트워크 전송량이 월등히 감소하는 것을 확인할 수 있다. 가시화노드 12대를 사용하였을 시 SDSL의 성능이 PDSL보다 12% 증가함을 알 수 있다.

Fig. 21 유한요소 모델 ①에 대한 네트워크 통신량 비교

3.4.2. 유한요소 해석결과를 이용한 성능 평가 모델 ②

아래의 그림22의 결과는 위의 그림 18의 100만 HEXA 요소 모델의 병렬 효율 결과이다.

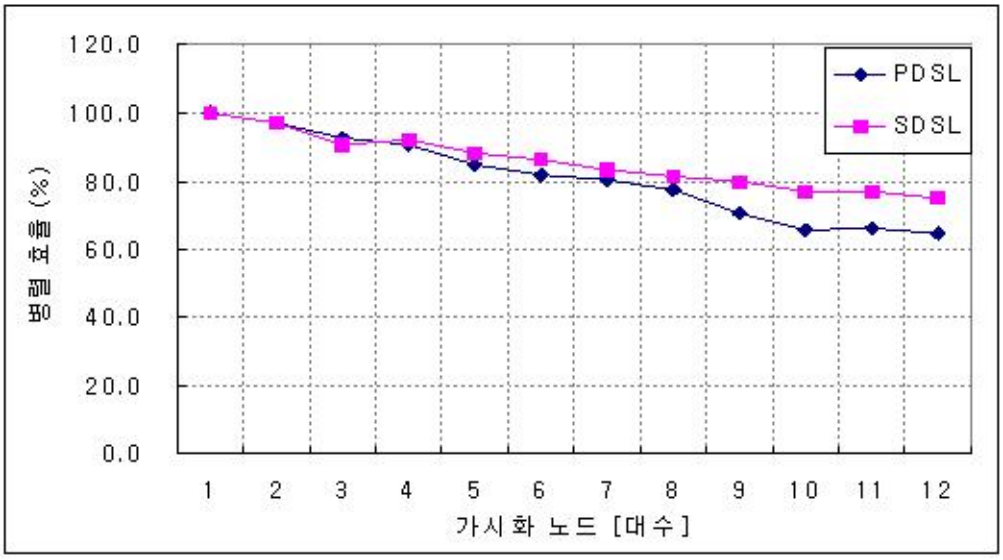

Fig. 22 100만 HEXA 모델 해석결과의 PDSL과 SDSL의 병렬 효율 비교

가시화 노드 12대의 결과에서는 SDSL 은 74.96%, PDSL 은 64.84% 로 SDSL 이 10% 이상 성능이 우수함을 확인할 수 있다. 위의 100만 HEXA 모델의 Speed-up을 나타내면 다음 그림 23과 같다. 아래의 결과로부터 12대의 가시화 노드를 사용할 경우 SDSL은 8.99, PDSL은 7.78의 속도 증가를 얻음을 확인할 수 있으며, 이로부터 제안된 SDSL 기법을 이용할 경우 기존 PDSL 기법보다 13% 높은 병렬 가시화 성능을 얻을 수 있음을 확인할 수 있다.

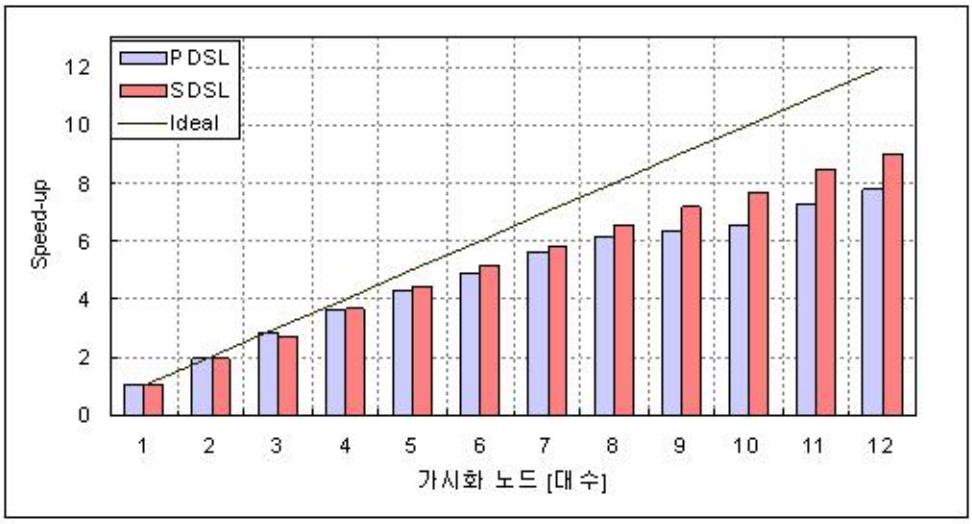

Fig. 23 100만 HEXA 모델의 성능 향상도

3.4.3. 유한요소 해석결과를 이용한 성능 평가 모델 ③

아래의 그림은 Pantheon 신전의 유한요소 해석결과를 병렬 가시화 한 예제이다.

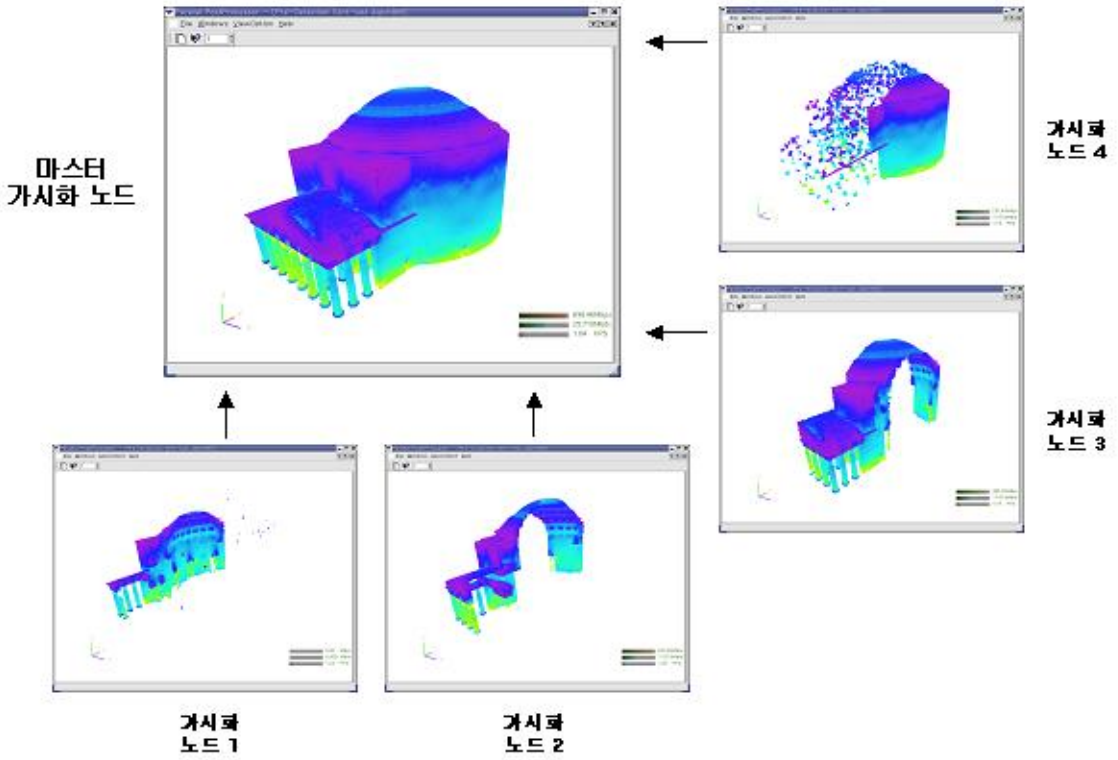

Fig. 24 Pantheon 신전 유한요소 해석결과의 병렬 가시화 예제 ③

아래 그림 25와 같은 Speed-up 결과를 통해 SDSL이 PDSL보다 성능이 향상됨을 확인할 수 있다.

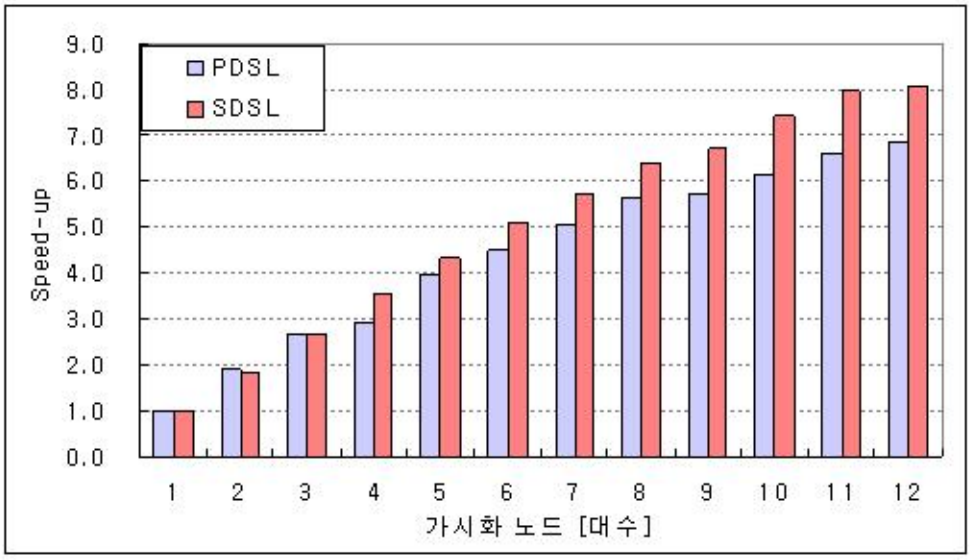

Fig. 25 Pantheon 신전 유한요소 해석결과의 병렬 가시화 성능 향상도

Ⅳ. 결　론

점차 증가 하고 있는 병렬 처리 규모에 따라 유한요소 해석의 규모도 증가하고 있다. 이로 인해, 대규모 해석을 수행할 수 있는 병렬처리 해석 과정과 과학적 가시화 기술의 확보가 우선되어야 한다. 실제적인 과학적 해석과정이 포함된 이러한 일련의 과정은 분산 메모리 환경에서의 초대형 해석결과의 유한요소 병렬 가시화에 관한 연구의 수행으로 이룰 수 있다. 병렬 처리를 통한 가상 설계/개발 시스템 구축을 연구하고, 구조 해석결과 및 과학적 가시화에 적합한 가시화 기법을 개발하여 가상설계/개발 시스템의 핵심 기술로 확보할 수 있다면 향후 항공우주구조 설계개발은 물론이고 공기역학, 기계, 토목, 건축, 원자력 설계분야 등에도 이용될 수 있을 것으로 판단된다.

본 논문에서 제안된 스트립 분할 탐색 알고리듬은 위와 같은 가상 설계/개발 기술의 기반이 되는 병렬 가시화 기법으로 , 해석 결과의 모델에 따라 한 개의 스트립을 이루는 꾁셀 라인 수를 다르게 조절 할 수 있으며 이를 통해 최적 꾁셀 라인 수를 구할 수 있는 장점을 가지고 있다. 각종 수치예제를 통하여 제안된 스트립 분할 탐색 알고리듬을 사용할 경우 기존 PDSL 알고리듬에 비해 네트워크 전송량과 깊이검사 시간을 감소시킬 수 있음을 확인하였다. 따라서 제안된 스트립 분할 탐색 알고리듬이 효율적인 병렬 가시화 성능을 발휘함을 입증하였다. 향후 병렬 유한요소해석 모듈과 제안된 스트립 분할 탐색 병렬 가시화 알고리듬 기반의 가시화 모듈을 통합하여 효율적 초대형 병렬 가상 구조시험 시스템을 구축할 수 있을 것으로 사료된다.

참고문헌

1) www.top500.org TOP500

2) Bhardwaj, M., Pierson, K., Reese, G., Walsh, T., Day, D., Alvin, K., Peery, J., Farhat, C., Lesoinne, M., "Salinas:A Scalable Software for High-Performance Structural and Solid Mechanics Simulations", *Proceedings of the IEEE/ACM SC2002 Conference*, November 2002, pp.35

3) Yagawa, G., Okuda, H., Nakamura, H., "GeoFEM : Multi-Purpose Parallel FEM system for Solid Earth", *Fourth World congress on Computational Mechanics*, Vol. 2, 1988, pp.1048.

4) Hiroshi Okuda, "Development of Solid Earth Simulation Platform," *FY2002 Report of Earth Simulator Results*, 2003.

5) G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett and P. Hanrahan. "WireGL : A scalable graphics system for clusters." Proceedings of SIGGRAPH 2001, pages 129-140, 2001

6) Kim, J.S., Lee, C.S., Kim. J.H., Joh, M.S., Lee, S.S., "IPSAP : A High-performance Parallel Finite Element Code for Large-scale Structural Analysis Based on Domain-wise Multifrontal Technique", *Proceedings of the ACM/IEEE SC2003 Conference*, 2003, p.32

7) 김승조, 이창성, 손정우, 하병언, 조진연, "인터넷 슈퍼컴퓨팅 기술의 구현", 한국 항공우주학회지, 29권 3호, 2001년, pp28-37

8) MolnarS., Cox, M., Elleworth, D., Fuchs, H., "A sorting Classification of Parallel Rendering" *IEEE Computer Graphics and Application* Vol.14, No.4, 1994, pp.23-32

9) 김창식, 송유미, 김기욱, 조진연 "분산 병렬 계산 환경에 적합한 초대형 유한요소 해석 결과의 효율적 병렬 가시화", 한국항공우주학회 2004 , 게재승인 (04-92호)

10) Foley, J., Dam, A., Feiner, S., Hughes, J., "Computer Graphics : Principles and practice", *ADDISON WESLEY*, second edition, 1997, pp.668-672