

다분야통합해석에 기반한 설계문제의 병렬처리를 위한 부하분산알고리즘

Load Balancing Algorithm for Parallel Computing of Design Problem involving Multi-Disciplinary Analysis

조 제 석* · 주 민 식** · 송 용 호*** · 최 동 훈****
Cho, JaeSuk · Chu, Min-Sik · Song, Yong Ho · Choi, Dong-Hoon

ABSTRACT

An engineering design problem involving Multi-Disciplinary Analysis(MDA) generally requires a large amounts of CPU time for the entire design process, and therefore Multiple Processing System (MPS) are essential to reduce the completion time. However, when applying conventional parallel processing techniques, all of the CAE S/W required for the MDA should be installed on all the servers making up MPS because of characteristic of MDA and it would be a great expense in CAE S/W licenses. To solve this problem, we propose a Weight-based Multiqueue Load Balancing algorithm for a heterogeneous MPS where performance of servers and CAE S/W installed on each server are different of each other. To validate the performance, a computational experiments comparing the First Come First Serve algorithm and our proposed algorithm was accomplished.

Keywords: Multi-Disciplinary Analysis, MDA, Parallel Computing, Multiple Processor System, MPS, Load Balancing, Multiqueue

1. 서 론

최근 제조업계의 제품설계단계에서의 요구사항은 제품의 성능을 극대화하면서 설계에 소요되는 시간을 최소화하는 것이다. 그러나 만족할만한 제품성능을 얻기 위해서는 많은 수의 설계안에 대한 시스템의 반응을 구해야 하며 이로 인해 설계안의 해석을 수행하는데 많은 시간이 소요된다. 설계안에 대한 시스템의 반응을 구하기 위해 여러 분야의 공학적 원리들을 동시에 고려하는 다분야통합해석(Multi-Disciplinary Analysis: MDA)을 수행해야 하는 설계문제일 경우 설계모델의 해석에 소요되는 시간은 더욱 커지게 된다.

이러한 소요시간을 단축하기 위해 병렬처리시스템(Multiple Processor System: MPS)을 도입하고자 하는 연구가 진행되고 있다. 그러나 MDA에 기반한 설계문제의 경우, 일반적인 병렬처리방법을 적용하여 병렬처리를 효율적으로 수행하기 위해서는 MPS의 모든 서버에 MDA에 필요한 모든 CAE S/W들을 설치해야 하

* 한양대학교 최적설계신기술연구센터 연구조원 Email: danbimon@hanyang.ac.kr

** 한양대학교 최적설계신기술연구센터 연구원 Email: mdoframe@hanyang.ac.kr

*** 한양대학교 정보통신공학부 교수 Email: yhsong@hanyang.ac.kr

**** 정희원 · 한양대학교 최적설계신기술연구센터 소장 Email: dhchoi@hanyang.ac.kr

며, 설계문제의 MDA가 상용 CAE S/W 모델로 구성된 경우 이는 매우 큰 S/W 비용을 필요로 한다.

본 연구에서는 MPS의 각 서버들에 MDA에 필요한 CAE S/W들이 각각 모두 설치되어 있지는 않은 경우를 고려한 가중치 기반 멀티큐 부하분산알고리즘을 제안하였다. 제안된 알고리즘은 각 CAE S/W모델별로 별도의 큐를 관리하고, 각 모델들의 해석에 가중치를 부여하였다. 알고리즘의 성능을 검증하기 위해 각 모델의 해석과 MPS의 각 서버들을 모델링한 시뮬레이터를 구현하여 전산실험을 수행하였다.

2. 연구배경 및 연구목적

실제 산업현장의 설계분야에서 일반적으로 사용되는 설계기법인 반응표면법(Response Surface Method), 다구찌기법(Taguchi Method), 매개변수분석기법(Parametric Study) 등을 적용하기 위해서는 설계기법이 요구하는 다수의 실험점(Sampling Point)에 대한 시스템의 반응을 구해야 한다. 산업현장에서는 일반적으로 상용 CAE S/W로 구현한 정밀한 설계모델을 사용함을 고려할 때 이는 매우 긴 시간을 필요로 한다. 이러한 소요 시간을 단축하기 위해 MPS를 이용하고자 하는 연구(Kato, T., 2005; Kim, H. et al., 2005; Koch, P.N. et al., 2000)가 진행되고 있다. 설계문제가 단일분야의 해석(e. g. 구조해석, 유동해석, NVH해석)에 기반한 일반적인 문제일 경우 MPS의 각 서버에 해당 CAE S/W 하나만 설치되어 있으면 선입선출(First Come First Serve; FCFS) 알고리즘과 같은 일반적인 병렬처리기법을 적용하여도 MPS의 최대성능을 이용할 수 있다.

그러나 MDA에 기반을 둔 설계문제일 경우 단일분야의 해석에 기반한 설계문제에 적용된 병렬처리 개념을 확장하는 형태로 선입선출알고리즘을 적용하기 위해서는 MDA에 필요한 모든 CAE S/W가 MPS의 각 서버에 모두 설치되어야 한다. 일반적으로 산업현장에서는 고가의 상용 CAE S/W를 사용함을 고려할 때 이는 매우 큰 S/W비용을 필요로 한다. 그러나 MDA는 그림 1과 같이 해석들이 순차적으로 수행되어야 하는 부분이 많으며 모든 CAE S/W를 동시에 실행하는 것은 아니다. 본 논문에서는 MPS의 각 서버에 모든 CAE S/W가 설치되지 않는다고 서버들의 성능도 상이한 Heterogeneous MPS를 사용하여 병렬처리를 수행하기 위한 부하분산알고리즘을 제안하였다.

부하분산알고리즘(Load Balancing)은 병렬처리의 목적, 작업들의 특성과 MPS의 구성방식에 따라 매우 다양한 개발되었으며 문제의 조건이 다를 경우 기존 알고리즘을 적용하면 효율성을 기대하기 어렵다. 일관된 기준으로 알고리즘을 분류하는 것은 매우 어려우며 (Kameda, H. et al., 1997; BARAK, A. et al., 1993) 다양한 분류기준이 제안되었다.

일반적으로 공학이나 과학 분야에서 다루는 문제들은 각 프로세서에서 독립적으로 수행할 수 있는 부 문제(Sub Problem)로 분할할 수 있는 개수에 따라 유동해석이나 충돌해석등과 같이 무한히 많은 수의 부 문제들로 분할할 수 있어 Fine-grained Parallelism을 적용하기 적합한 형태의 문제와 단백질 구조분석이나 신약 후보물질 탐색등과 같은 'What if' 형태의 문제같이 비교적 개수가 적으며 상호독립적인 부 문제들로 분할할 수 있어 Coarse-grained Parallelism을 적용하기 적합한 문제로 분류할 수 있다.

MDA에 기반을 둔 설계문제에서 수행해야 하는 각 CAE S/W모델들의 해석은 독립적으로 수행되며 더 이상 분할할 수 없는 부 문제로 볼 수 있으므로 본 논문에서 제안하는 알고리즘은 Coarse-grained Parallelism을 적용하였다

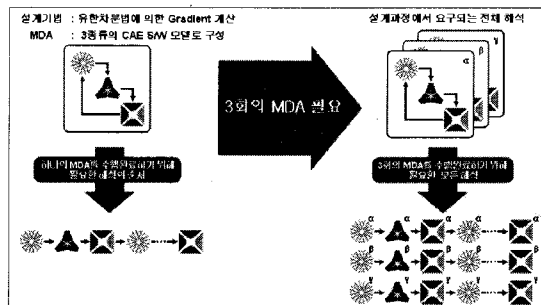


그림 1 설계기법에 의해 생성된 MDA들을 수행하기 위한 각 CAE S/W 모델들의 해석순서

3. 가중치 기반 멀티큐 부하분산알고리즘

3.1. 다분야통합해석에 기반한 설계문제와 병렬처리시스템의 모델링 방법

본 논문에서 제안된 알고리즘은 설계문제의 MDA가 I개의 CAE S/W 모델들로 구성되며 모델의 해석순서는 설계구조행렬(Design Structure Matrix: DSM)에 정의되고, 설계기법에 정의된 N개의 실험점(Sampling Point)에 의한 시스템의 반응값을 구하기 위해 N개의 MDA를 수행해야 하는 경우를 고려한다. 또한 병렬처리를 수행할 MPS는 각 서버들의 성능이 각기 다르며, 설치된 CAE S/W들의 종류도 각기 다른 M개의 서버로 구성된 Heterogeneous MPS인 경우를 고려하였다. n번째 MDA의 DSM에 속한 i번째 모델의 k번째 해석은 작업 ${}_k J_i^n$ 으로 모델링되며, ${}_k J_i^n$ 의 수행완료에 필요한 연산의 예상치인 요구계산량은 ${}_k J_i^n : C$ (unit : GHz·sec)로, DSM의 j번째 모델이 J_j^n 의 해석결과를 입력변수로 사용하는지 여부는 $J_j^n : \mu_j$ 로, MDA가 수렴하기까지 수행해야 하는 J_j^n 의 반복회수의 예상치를 $J_j^n : \delta$ 로 정의한다. 또한 MPS의 m번째 서버는 R_m 으로 모델링되며, R_m 의 성능은 $R_m : P$ (unit : GHz)로, R_m 에 DSM의 i번째 모델의 해석을 수행할 수 있는 CAE S/W의 설치여부는 $R_m : B_i$ 로 정의한다. MDA와 MPS를 모델링 예는 그림 2와 같다.

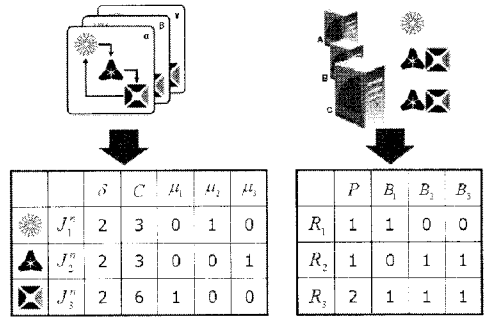


그림 2 MDA와 MPS의 모델링 예

해석소요시간 ${}_k T_i^n$ 은 ${}_k J_i^n$ 이 R_m 에서 수행되었을 때 완료에 소요되는 시간이며 식 (1)과 같이 정의된다.

$$(1) T_i^n = \frac{J_i^n : C}{R_m : P} = c/p \text{ (sec)} \quad [\text{where } J_i^n : C = c \text{ GHz} \cdot \text{sec}, R_m : P = p \text{ GHz}]$$

실제 설계문제에서는 실험점에 의해 정해지는 입력변수의 초기값에 따라 ${}_k J_i^n : C$ 와 $J_j^n : \delta$ 의 크기가 모두 다르므로 ${}_k T_i^n$ 도 모두 다르며 각 MDA가 언제 수렴할 지도 정확히 예측할 수 없다. 따라서 설계과정의 초기에는 $J_j^n : \delta$ 와 ${}_k J_i^n : C$ 를 추정하여 부여하고 설계과정 중 수렴한 MDA들의 $J_j^n : \delta$ 의 평균값 $J_j^{avg} : \delta$ 을 아직 수렴하지 않은 MDA들의 $J_j^n : \delta$ 의 예상치로, 마찬가지로 각 모델의 수행 완료된 해석들에 사용된 연산횟수의 평균값 $J_i^{avg} : C$ 을 아직 수행되지 않은 해석의 예상치로 이용해야 한다.

그러나 과거이력을 사용할 경우 알고리즘의 성능을 정확히 비교평가하기 어려워지므로 동일한 CAE S/W 모델이면 모델의 입력변수값이 다르더라도 $J_j^n : \delta$ 와 $J_i^n : C$ 의 값이 동일하며 초기에 추정하여 부여한 예상치에서 변하지 않는다고 가정한다.

3.2. 제안된 알고리즘의 개념

MDA의 DSM이 그림 1과 같고 설계기법에 의해 3회의 MDA를 수행해야 하며 $J_1^n : \delta$, $J_2^n : \delta$, $J_3^n : \delta$ 가 모두 2회인 설계문제를 살펴보자.

만약 $R_A : P, R_B : P, R_C : P$ 가 2 GHz로 모두 동일하고 $J_1^n : C, J_2^n : C, J_3^n : C$ 가 10 GHz·sec로 모두 동일하다면 각 서버가 작업들을 수행한 이력은 그림 3과 같다. 모든 작업이 완료된 시점을 비교해보면 그림 3의 b)와 같이 R_A 가 J_1^n , R_B 가 J_2^n , R_C 가 J_3^n 을 각각 수행할 수 있는 경우 정해진 해석순서로 인해

R_B, R_C 가 유휴상태(Idler)로 대기하는 시간이 약간 존재한다 하더라도 그림 3의 a)와 같이 각 서버가 모든 작업을 수행할 수 있는 경우에 근접하는 병렬처리 성능을 보일 수 있게 된다.

그러나 $R_A : P, R_B : P, R_C : P$ 는 2 GHz로 모두 동일하며 $J_1^n : C, J_2^n : C$ 도 10 GHz·sec로 동일하지만 $J_3^n : C$ 가 20 GHz·sec일 때 R_A 가 J_1^n , R_B 가 J_2^n , R_C 가 J_3^n 을 수행할 수 있을 경우 그림 4의 b)와 같이 T_1^n, T_2^n 에 비해 T_3^n 가 두 배이기 때문에 병렬처리 과정 중간에 R_A, R_B 가 유휴상태로 대기하는 현상이 발생하며 이로 인해 성능이 그림 4의 a)에 비해 현저히 떨어지게 된다.

$R_A : P, R_B : P$ 가 2 GHz로 동일하나 $R_C : P$ 는 4 GHz이고 $J_1^n : C, J_2^n : C$ 도 10 GHz·sec로 동일하나 $J_3^n : C$ 가 20 GHz·sec이며 R_A 가 J_1^n 을, R_B, R_C 가 각각 J_2^n, J_3^n 을 수행할 수 있는 경우를 고려해보자. 그림 5에서 각 서버가 작업들의 병렬처리를 수행한 이력을 살펴보면 J_2^n 와 J_3^n 을 R_B, R_C 중 어떤 서버에서 수행하느냐에 따라 전체 병렬처리 성능이 매우 달라진다. 그림 5의 b)와 같이 R_C 가 J_2^n , R_B 가 J_3^n 을 수행할 경우 T_3^n 가 T_1^n 의 두 배, T_2^n 의 네 배가 되어 그림 5의 a)와 같이 R_B 가 J_2^n , R_C 가 J_3^n 을 수행하여 T_1^n, T_2^n, T_3^n 가 모두 동일한 경우보다 병렬처리 성능이 현저히 떨어지게 된다.

따라서 병렬처리성능을 최대화하기 위해서는 각 모델들의 전체해석소요시간 중 최대값인 $Max(\sum T_i^n)$ 을 최소화해야하며 각 모델들의 $J_i^n : C$ 와 각 서버들의 $R_m : P$ 을 고려하여 가능한 한 $J_i^n : C$ 이 큰 모델부터 J_i^n 을 수행할 수 있는 서버들 중 $R_m : P$ 이 높은 서버에 할당함으로써 서버들의 부하(Workload)를 조절해야 한다. 이를 위해 제안된 알고리즘은 각 모델의 서버할당의 우선순위로 사용되는 가중치를 부여한다.

MPS가 하나의 큐만으로 모든 작업들을 관리할 경우 유휴상태로 대기중인 서버가 발생했을 때 해당서버가 큐의 최상단에 대기중인 J_1^n 은 수행할 수 없으나 큐의 중간부분에 대기중인 J_2^n 은 수행할 수 있다 하더라도 J_1^n 을 수행할 수 있는 다른 서버가 유휴상태가 되어 J_1^n 이 큐에서 제거될 때 까지 유휴상태로 대기해야 한다. 이로 인한 병렬처리의 효율성의 저하를 방지하기 위해서는 작업별로 독립적인 큐를 생성해야 한다.

3.3. 가중치 부여방법

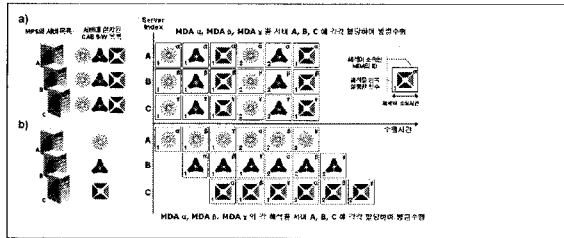


그림 3 MPS의 서버들의 성능이 동일하고 MDA의 해석들의 요구계산량도 동일한 설계문제의 병렬처리

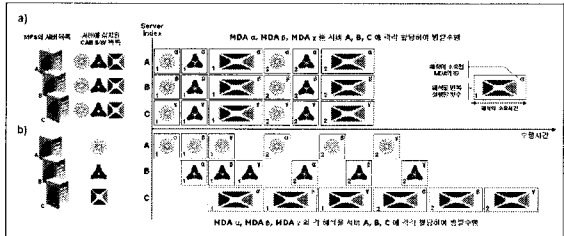


그림 4 MPS의 서버들의 성능은 동일하고 MDA의 해석들의 요구계산량은 상이한 설계문제의 병렬처리

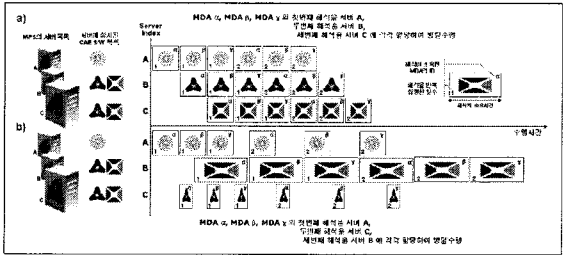


그림 5 MPS의 서버들의 성능이 상이하고 MDA의 해석들의 요구계산량도 상이한 설계문제의 병렬처리

제안된 알고리즘은 식 (2)를 통해 가중치를 각 CAE S/W 모델에 부여하며 각 모델의 가중치를 구하기 위해서는 각 서버들이 어떤 작업들을 담당할지 결정해야한다. 각 서버들이 담당할 작업들은 $Max(\sum T_i^n)$ 을 최소화할 수 있는 조합을 찾음으로서 결정된다. 가중치 ${}_k J_i^n$ 은 ${}_k J_i^n$ 의 해석결과를 입력변수로 사용하는 다른 모델들 중 ${}_k J_i^n$ 보다 DSM에서 상단에 위치하는 모델이 존재하는지 여부에 따라 다른 식이 적용된다.

(2) a) [if all of $J_i: \mu_D=0$ ($1 \leq D < i$)]

$${}_k W_i^n = {}_k T_i^n + \sum_{d=i+1}^I ({}_k J_i^n: \mu_d \times {}_k T_d^n)$$

b) [if any of $J_i: \mu_D=1$ ($1 \leq D < i$)]

$${}_k W_i^n = {}_k T_i^n + \sum_{d=i+1}^I ({}_k J_i^n: \mu_d \times {}_k T_d^n) + ({}_k J_i^n: \delta - k) \times \sum_{d=D}^i {}_k T_d^n$$

[where, $T_i^n = \frac{J_i^n: C}{\sum_{m=1}^M (R_m: B_i \times R_m: P)}$]

각 모델의 가중치가 정해지면 DSM에 정의된 순서에 따라 각 작업들을 해당 작업의 큐에 등록한다. 서버들 중 유휴상태로 대기중인 서버가 발생하면 각 큐들의 최상단에 대기중인 작업들을 비교하여 해당서버에서 수행할 수 있고 가중치가 가장 높은 작업을 할당하며 또한 각 작업의 가중치들을 주기적으로 갱신한다.

4. 전산실험 결과

4.1. 실험환경 및 비교평가방법

제안된 알고리즘의 성능을 평가하기 위해 C++ 로 구현한 시뮬레이터를 작성하였다. 3.1절에 설명한 모델링 방법으로 작업과 자원, 큐를 구현하고, 3.3절에 설명한 동작방식으로 각 CAE S/W모델의 해석이 서버에서 수행되는 과정을 모사하였으며 일반적인 병렬처리기법인 싱글큐 선입선출 알고리즘의 성능과 비교분석하였다. 싱글큐 선입선출알고리즘은 큐의 최상단에 대기중인 작업을 수행할 수 있는 서버가 발생하면 작업을 서버에 할당하게 되며 작업을 수행할 수 있는 서버가 동시에 2개 이상 발생할 경우 서버의 성능에 대한 고려 없이 서버의 ID가 앞쪽에 있는 서버부터 할당하도록 구현하였다.

4.2. 실험결과 및 고찰

MDA의 DSM의 구조와 MPS의 형태는 그림 6과 같다. DSM은 CAE S/W모델의 해석순서에 두 개의 루프가 존재하고 두 개의 루프사이에 루프에 속하지 않는 모델이 끼어 있는 비교적 복잡한 형태이며, MPS는 각 서버들이 수행할 수 있는 작업들의 종류가 비교적 독립적인 형태의 문제이다.

MDA가 15개로 비교적 많으므로 유휴상태에서 대기하는 서버가 발생하였을 때 해당 서버가 수행할 수 있는 작업이 큐에 존재할 가능성이 매우 높다. 여러 종류의 작업들이 하나의 큐에 대기하고 있고 큐의 뒤쪽에 있는 작업을 수행할 수 없는 싱글큐보다는, 작업의 종류별로 별도의 큐를 유지하는 멀티큐가 서버들이 유휴상태에서 대기할 가능성이 적다. 따라서 제안된 알고리즘이 싱글큐 선입선출알고리즘보다 우수한 성능을 보일 수 있는 문제이다.

전산실험결과 그림 7과 같이 주어진 15개의 MDA의 수행을 완료하는데 Heterogeneous MPS를 이용한 실험

Example 2) 상용기법에 대해 상용기 15개 문제

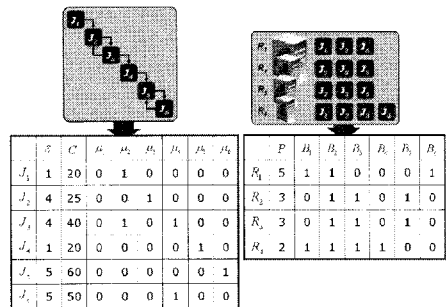
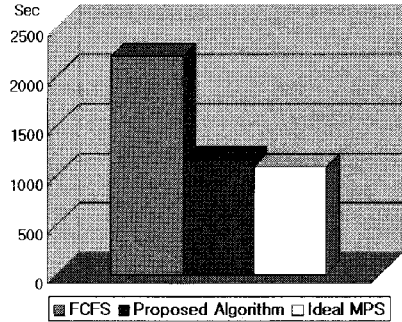


그림 6 전산실험 예제문제의 모델링 결과

그림 6 전산실험 예제문제의 모델링 결과. DSM은 CAE S/W모델의 해석순서에 두 개의 루프가 존재하고 두 개의 루프사이에 루프에 속하지 않는 모델이 끼어 있는 비교적 복잡한 형태이며, MPS는 각 서버들이 수행할 수 있는 작업들의 종류가 비교적 독립적인 형태의 문제이다.

글루 선입선출알고리즘은 2202.5 sec, 제안된 알고리즘은 1165.0 sec의 시간이 소요되었으며 MPS의 각각의 서버에 MDA의 수행에 필요한 모든 CAE S/W가 설치된 이상적인 MPS에서 병렬 처리를 수행한 경우에도 1089.5 sec가 소요되었다. 위 실험을 통해 제안된 알고리즘의 성능이 동일한 MPS를 사용할 경우 싱글 큐 선입선출알고리즘에 비해 약 47% 우수함을 확인하였으며 이상적인 MPS에 근접하는 성능을 나타냄을 확인할 수 있었다.



5. 결론

그림 7 전체 MDA의 완료에 소요된 시간

Heterogeneous MPS를 이용하여 복수의 MDA들의 병렬처리를 수행할 경우 효율성을 향상시키기 위한 가중치기반 멀티큐 부하분산알고리즘을 제안하였으며 알고리즘의 특징은 다음과 같다.

① MDA를 구성하는 각 CAE S/W모델들이 서버를 할당받을 수 있는 우선순위를 나타내는 가중치를 계산하고 MPS의 각 서버들 중 유휴상태(Idle)로 대기하는 서버가 발생하게 되면 MPS의 큐들을 검색하여 해당서버에서 해석을 수행할 수 있는 CAE S/W모델들 중 가장 가중치가 높은 모델을 할당한다.

② CAE S/W모델의 해석의 가중치는 각 CAE S/W 모델의 해석에 필요한 연산의 크기와 MPS의 각 서버들의 성능, 해당모델의 해석결과를 입력변수로 사용하는 모델들에 대한 가중치를 고려하여 부여한다.

③ CAE S/W모델의 종류에 기반한 멀티큐를 도입하여 수행대기중인 작업들을 종류별로 독립적으로 관리함으로써 유휴상태의 서버 발생 시 서버가 유휴상태로 대기하게 되는 시간을 최소한으로 줄였다.

작업과 서버, 큐를 모델링하였으며 C++를 이용하여 구현한 시뮬레이터를 구축하였으며 알고리즘의 성능을 싱글큐 선입선출알고리즘과 비교, 평가한 결과 제안된 알고리즘의 성능이 전반적으로 우수함을 확인하였다.

감사의 글

본 연구는 최적설계기술연구센터와 2007년도 2단계 두뇌한국21사업, 산업자원부의 국제공동기술개발사업, 서울시 산학연 협력 사업에 의하여 지원되었으며 지원해주신 각 당국에 감사의 뜻을 표합니다.

참고문헌

- Kato, T. (2005) Realizing Grid Computing as Engineering System for Collaborative Parameter Study, *GridWorld/GGF15*, Boston, MA, USA.
- Kim, H., et al. (2005) A Parallel Trade Study Architecture for Design Optimization of Complex Systems, *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*.
- Koch, P.N., B. Wujek, O. Golovidov. (2000) A Multi-Stage, Parallel Implementation of Probabilistic Design Optimization in an MDO Framework, *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*.
- BARAK, A., S. GUDAY, R.G. WHEELER, (1993) *The MOSIX Distributed Operating System - Load Balancing for UNIX*, Springer-Verlag.
- Kameda, H., et al., (1997) *Optimal Load Balancing in Distributed Computer Systems*, Springer.