JLBS의 효율적인 운용을 위한 시뮬레이션

남동진* · 한영신** · 이칠기***

Simulation for Efficient Employment of JLBS

Dong-Jin Nam, Young-Shin Han and Chil-Gee Lee

요 약

JLBS는 multi-server multi-queue 방식을 채택하고 있으며, 각 tool별로 할당된 license 수에 비례하여 system을 할당한다. 그러나 이러한 방식의 문제점은 특정 tool에만 job이 집중될 경우 비효율적이다. 즉 특정 system은 모두 사용되고 있고, queue에는 많은 job이 대기하고, 여타 tool에 해당하는 queue에는 대기하는 job이 없어서 system이 그냥 놀고 있는 현상이 발생한다. 본 논문에서는 이러한 단점을 극복하기 위해 각 tool에 license 수에 비례하여 할당되었던 system을 모든 tool들이 system을 공유할 수 있도록 하는 방법을 제안했다. 대신 system의 숫자는 줄이고, license의 숫자는 더 할당하는 방법으로, 기존의 방법보다 더 효율적으로 나타났다.

1. 서론

전자, 반도체, 자동차, 철강 등 여러 분야에 있어서 현대의 제조 시스템의 공통적인 특성이란 설비, 제어, 및 운영이 결합된 상태로 살아서 움직이고 변화해 가는 시스템이라는 것이다. 시간이 흐를수록 이들요소의 결합은 더욱더 유기적이고 빠른 템포로 진행되고 있다. 중 저가 제품의 대량생산에서 고부가 제품을 더 빨리 더 많이 더 탄력적으로 생산할 수 있는 생산 시스템의 구축과 운영을 위해서는 필연적으로 표준화, 자동화를 거쳐 시스템화를 추구하게 되며 이 과정에서 있어 특히 자동화에서 시스템화로 전개되는 단계에서는 갈수록 복잡하고 거대해지는 제조 시스템의 동적 역학을 분석할 수 있는 기법의 도입이 생산시스템 경쟁력 및 생산 기술력 향상을 위한 필연적 요구 사항이 된다.

현재 우리나라의 대부분의 생산시스템 기술 또는 시스템 설계 기술은 경험과 느낌을 바탕으로 해 왔으나 최근 수년 동안은 Simulation과 같은 과학적이고 합리적인 분석 기법의 도입 필요성을 절감하는 단계에 있었다. 시뮬레이션이란 금세기 최고의 시스템 분석 기법으로서 작게는 Engineer자신의 문제해결을 위한 Engineering Tool이며 나아가 Engineer와 Engineer사이, Engineer와 의사 결정권자 사이의 Communication Tool이다.

본 연구에서는 2개의 simulation model의 performance를 queue에서 평균적으로 대기하고 있는 job의 수, 평균적 대기시간, 각 server의 utilization 3가지 관점에서 비교분석 하였다.

2. JLBS(Job Loading Busy System)

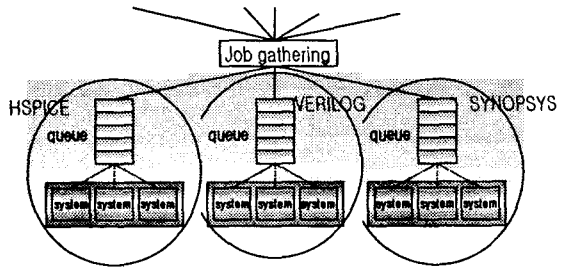
JLBS는 local system에서 다수의 job을 동시에 수행하는 경우, system의 performance 저하와 설계기간을 증가시키는 원인을 해결하

* 성균관대학교 정보통신공학부

고자 제안된 system이다. JLBS는 별도로 전용 system을 할당하고, 하나의 system에서는 하나의 job만을 수행하도록 하여, job 수행의 편리성을 사용자에게 제공함으로써, 설계 환경의 최적화를 목적으로 하는 system이다.

2.1 기존의 JLBS 방식

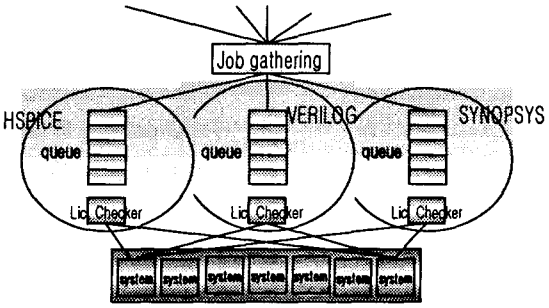
<그림1>은 현재 JLBS 방식을 보여주는데, multi-server multi-queue 방식을 채택하고 있으며, 각 tool별로 할당된 license 수에 비례하여 system을 할당한다. 그러나 이러한 방식의 문제점은 특정 tool에만 job이 집중될 경우 비효율적일 수 있다는 것이다. 즉 특정 system은 모두 사용되고 있고, queue에는 많은 job이 대기하고, 여타 tool에 해당하는 queue에는 대기하는 job이 없어서 system이 그냥 놀고 있는 현상이 발생할 가능성이 있다는 것이다.



<그림 1> 기존 JLBS 방식

2.2 제안된 JLBS 방식

<그림 2>는 기존의 JLBS 방식을 개선하고자 제안하는 방식이다. 각 tool에 license 수에 비례하여 할당되었던 system을 모든 tool들이 system을 공유할 수 있도록 하는 방법을 생각해 보았다. 대신 system의 숫자는 줄이고, license의 숫자는 더 할당하는 방법으로, 물론 완벽한 해결책은 아니지만, 기존의 방법보다는 더 효율적일 것이다. 즉 전용system의 숫자보다 license의 숫자를 더 많이 할당함으로써, 유휴 system을 줄이자는 것이 목표이다. 별도로 license control system을 두고 이를 통해 할당된 license의 수만큼만 수행될 수 있도록 하는 방식이다.

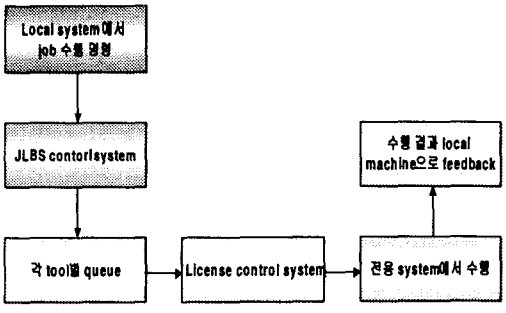


<그림2> JLBS 개선안

3. JLBS 수행 procedure

<그림 3>은 JLBS를 통해 job을 수행할 때의 procedure를 나타낸 것이다.

① 사용자가 local machine에서 job 수행명령을 내린다.
② job이 JLBS control system으로 모인다.
③ 수행하고자 하는 tool의 queue로 들어간다.
④ queue에 대기하고 있는 job이 있으면, License control system에서 가용 license와 가용 system이 있는지 확인하고, 있으면 전용 system으로 job을 수행하라는 명령을 내린다.
⑤ job수행이 완료되면 수행결과를 local machine으로 feedback 하여 준다.



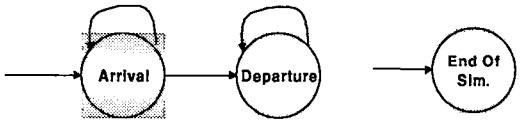
<그림3> JLBS 수행 procedure

기존의 방식에서는 license 수와 system의 수가 동일하게 할당되어 있기 때문에, 가용 system이 존재하면 job을 수행할 수 있었는데, 개선된 방식에서는 engine system이 모든 tool이 공유하고, system보다 license를 더 많이 할당함으로써, license control system을 통해 대기중인 job을 수행할 것인지를 판단하는 것이다.

4. 기존 JLBS Modeling

4.1 event

기존 JLBS의 event는7개가 있다. 각tool에 해당하는 job이 queue에 도달하는 arrival event, job이 수행을 하는 departure event, simulation을 종료하는 event로 구분할 수 있으며, <그림 4>에 event graph를 표현하였다.



<그림4> 기존 JLBS modeling을 위한 event graph

4.2 기존 JLBS modeling을 위한 Arrival function

Arrival function은 job의 next arrival event를 scheduling함에 의해서 시작된다. 다음으로 여러개의 server 중에서 idle인 server가 존재하면 해당 job의 delay를 0으로 하고, idle인 server에서 job 수행을 진행하고, server는 busy 상태로 변환시킨다. 그리고 해당 job의 departure event에 대한 scheduling을 한다. 만약 idle인 server가 존재하지 않으면, queue에 맨 마지막으로 들어가서 대기하게 된다.

5. New JLBS modeling

Event, Arrival function은 기존 JLBS의 방법과 동일하다.

5.1 Departure function

Departure function은 기존 JLBS의 departure function과는 다르다. 우선 기존에는 각 tool별로 할당되어 있는 server가 정해져 있었지만, 지금은 모든 server를 모든 tool의 queue와 connection이 되어 있고, departure function 초기에 license control system을 추가하여, license의 state가 idle인지, busy인지의 여부를 판단하는 routine이 들어가게 된다. Job이 server에서 service를 종료하게 되면 할당된 license 중에서 idle한 license가 존재하는지 판단을 한다. 만약 idle한 license가 존재하지 않고, 모두 busy하다면 server의 state는 idle로 변환하여 주고, return을 하게 된다. idle한 license가 존재하면, 다시 각 queue에 대기하고 있는 job이 있는지를 판단하게 된다. 만약 각 queue에 대기하고 있는 job이 존재하지 않는다면, 여기서 server의 state는 idle로 변환하여 주고, return을 하게 된다. 반대로 대기하고 있는 job이 존재한다면, 해당 queue에서 대기하고 있는 첫번째 job을 queue에서 remove시키고, queue에서의 delay time을 계산하고, 해당 server로 들어가서 service를 수행하게 된다. service를 수행하게 되면 해당 job의 departure event를 scheduling하고 return하게 된다.

6. 시뮬레이션 및 결과분석

6.1 Environment

2개의 simulation model을 구현하는데, 기본적으로 C/C++을 사용하여 modeling하였다. 내부적으로 arrival function, departure function, report function 등은 simlib library를 이용하였다. 기본적으로는 Windows 98환경에서 구현하고, 실행할 수 있으며, 별도로 GUI가 없기

때문에 Windows 환경외에도 UNIX-OS 환경에서도 수행할 수 있도록 되어 있다. Simulation model을 구현하고 test한 환경은 다음과 같다.

- CPU: Celleron 466
- RAM : 96M
- Simluation tool : Simlib library
- Compiler : Microsoft Visual C++ 6.0

6.2 Simulation model의 performance 비교 조건

2개의 simulation model은 약간의 차이가 있다. 기존의 방법은 tool별로 별도로 할당된 server가 있고, 할당된 server만큼의 license가 있다. 그러나 새로운 방법은 모든 tool이 server를 공유하게 되고, 대신에 departure function에서 license control routine을 두는 방법이다. 2개의 simulation model의 performance를 비교하기 위해 입력변수도 일부 다르게 지정되어 있다. 그러나 performance를 비교하는 기본적인 조건은 비용이다. 즉 license는 1copy당 5만$, server는 1대당 2만$이라고 가정하자. 그러면 기존의 방법에서 각 tool당 server가 7대씩 할당되어 있다고 하면, 기존의 JLBS system을 구축하는데 (5+2)*7*3 = 147만$이 소요된다. 그러면 새로운 simulation model에 적용할 때는 system을 구축하는데 147만$을 가지고 license와 server를 융통성있게 조절하여 입력으로 적용하는 것이다. 즉 기존 model에서는 server가 총 21대였는데, 여기서 server 3대를 줄이고, 대신 license 1copy를 추가할당하면 146만$이 되고, 이 비용으로 system을 구축한다는 전제하에서, 2개의 simulation model의 performance를 비교하는 것이다.

<표 3> Performance 비교를 위한 입력 조건

기존 model	각 tool당 7개의 server와 license를 할당 -> 147만$	Verilog server : 7, 6, 5 Verilog license : 7, 6, 5 Synopsys server : 7, 6, 5 Synopsys license : 7, 6, 5 Hspice server : 7, 6, 5 Hspice license : 7, 6, 5
New model	기존 model의 input 조건 중에서 server의 개수를 3개 줄이고, license를 1copy 추가 할당 -> (18*2)+(5*22) = 146만$	Total server : 18, 15, 12 Verilog license : 8, 7, 6 Synopsys license : 7, 6, 5 Hspice license : 7, 6, 5

6.3 결과분석

simulation은 총비용이 84만$, 105만$, 126만$, 147만$의 4가지 경우를 적용하였다. 그 결과는 표 5, 6, 7에 나타내었다. 표 5, 6, 7의 결과를 분석하면 다음과 같은 결론을 유추할 수 있다.

- Average utilization은 전반적으로 server를 공유하는 새로운 방식이 더 좋음을 확인하였다.
- Queue에 대기하고 있는 average job number와 job의 average delay time은 총비용이 84만$, 105만$일때는 기존방식이, 126만$, 147만$일때는 새로운 방식이 더 좋음을 확인하였다.
- 결국 server의 개수가 많을수록 server를 공유하는 새로운 방식이 더욱 효율적이고, server의 개수가 적을수록 효율성은 떨어짐을 확인하였다.
- 기적용하려는 JLBS system은 통상적으로 server의 갯수가 이번에 test한 server의 개수보다 는 더 많으므로 server를 공유하는 방식으로 교체하였을 경우에 효율성에서 이득을 얻을 수 있을 것이라고 판단된다.

6.3.1 Average number in queue

<표 5> Average number in queue

	기존	New
84만$	227.380	461.400
105만$	98.300	183.792
126만$	8.632	7.731
147만$	3,951	3.532

6.3.2 Delay in queue

<표 6> Delay times in queue

	기존	New
84만$	193.496	303.789
105만$	75.550	79.326
126만$	10.777	5.798
147만$	10.383	8.534

6.3.3 Server utilization

<표 7> Average server utilization

	기존	New
84만$	0.816	0.887
105만$	0.796	0.900
126만$	0.686	0.815
147만$	0.612	0.712

7. Conclusion

JLBS는 multi-server multi-queue 방식을 채택하고 있으며, 각 tool별로 할당된 license 수에 비례하여 system을 할당한다. 그러나 이러한 방식의 문제점은 특정 tool에만 job이 집중될 경우 비효율적이다. 즉 특정 system은 모두 사용되고 있고, queue에는 많은 job이 대기하고, 여타 tool에 해당하는 queue에는 대기하는 job이 없어서 system이 그냥 놀고 있는 현상이 발생한다. 본 논문에서는 이러한 단점을 극복하기 위해 각 tool에 license 수에 비례하여 할당되었던 system을 모든 tool들이 system을 공유할 수 있도록 하는 방법을 제안했다. 대신 system의 숫자는 줄이고, license의 숫자는 더 할당하는 방법으로, 기존의 방법보다 더 효율적으로 나타났다.

감사의 글

본 연구는 한국과학재단 목적기초연구 (R01-2000-00250) 지원으로 수행되었음.

참고문헌

1) Averill M. Law, W.David Kelton : Simulation Modeling and Analysis, Third Edition, Chap4. 235 ~ 260 (2000)
2) Brian W. Kernighan, Dennis M. Ritchie : The C Programming Language, Second Edition, (1988)
3) Christopher J. Van Wyk : Data Structures and C Programs, (1989)
4) Law, Averill M. Kelton, W. David, Simulation Modeling & Analysis, McGraw-Hill, (1991)
5) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Fivest, Introduction to Algorithms, The MIT Press, Cambridge, england, (1989)