韓國컴퓨터情報學會 論文誌
第14卷 第6號, 2009. 6.

2009-14-6-1-2

폐쇄된 다중 클래스 시스템에 대한 신속한 성능 시뮬레이션 모델

김 용 수*

Agile Performance Simulation Model for a Closed Multi-class System

Yong Soo Kim*

요 약

큐(queuing) 이론은 전화회선 수를 예측하기 위해 1917년 Erlang에 의해 도입된 이후 1960년대 말이 되어서야 Scherr를 필두로 컴퓨터 시스템의 성능 분석에 적용되기 시작하였다. 2000년대 들어 인터넷의 팽창 등 IT기술의 발달로 비즈니스 환경이 속도 중심으로 바뀌면서 시스템 분석가들도 사용자의 IT서비스 만족도를 충족시키기 위해 컴퓨팅 환경 분석 및 예측을 신속히 수행할 필요가 생겼다. 또한 컴퓨터 장비 가격은 급격히 하락하고 성능은 증가하는 상황에서 컴퓨터의 성능을 분석하여 개선하기 보다는 새로운 장비를 도입하여 성능을 향상시키려 함으로서 비용의 낭비는 물론 현 시스템의 용량과 성능도 파악하지 못하고 있다. 그러므로 많은 시간을 요하는 자세하고 정확한 시뮬레이션 모델 보다는 때로는 조악하지만 신속한 모델이 필요할 수도 있다. 본 논문에서는 측정된 성능 데이터를 기초로 Menasce가 제시한 분석적 모델을 검토하고 ARENA를 이용하여 시뮬레이션 모델을 구축하여 검정함으로써 개략적이지만 신속한 성능 분석 및 예측 시뮬레이션 모델을 제시하고자 한다.

Abstract

The queuing theory was adopted by Erlang to predict the availability of telephone lines in 1917 and had not been used for computer system performance analysis until late 1960s when Scherr published a performance analysis of time-shared computer system. In 2000s, the explosive Internet usage and the development of IT technology made the business environment speed-centric and analysts should react swiftly to the ever-changing situation to satisfy the user requirement. It's tempting to solve the performance problem by purchasing new devices because the price of computers and peripherals are rapidly decreasing along with their increasing performance. But this scheme not only makes it difficult to understand the overall performance of the system but also wastes money. A coarse performance model that is gotten quickly is sometimes preferred to a complex and precise one that takes longer time to get. This paper examines an analytic model

• 제1저자 : 김용수
• 투고일 : 2009. 04. 14, 심사일 : 2009. 05. 07, 게재확정일 : 2009. 06. 19.
* 경원대학교 IT대학 컴퓨터공학전공 교수
※ 이 연구는 2009년도 경원대학교 지원에 의한 결과임

suggested by Menasce based on the measured data and suggests a simulation model using ARENA that takes a short time to build.

▸ Keyword : 성능(performance), 시뮬레이션(simulation), ARENA, 다중 클래스(multi-class), 폐쇄 모델(closed model)

Ⅰ. 서 론

컴퓨터 시스템의 성능을 모니터하여 병목현상을 파악하고 조율함으로써 성능을 향상시키고 새로운 부하에 대한 예측을 할 수 있다. 전화회선 수를 예측하기 위해 1917년 Erlang[1]에 의해 도입된 큐(queuing) 이론은 1960년대 말이 되어서야 Scherr[2]등에 의해 컴퓨터 시스템의 성능 분석에 적용되기 시작하였다. 최근에는 인터넷을 중심으로 비즈니스 환경이 급격히 변화하여 신속한 IT 서비스의 제공이 성공의 가장 큰 변수 중에 하나가 되었다. 컴퓨터 장비 가격의 하락으로 신규 장비의 도입이 쉬워졌지만 신규장비의 도입만으로 시스템의 성능이 향상되지는 않는다. 사용자의 IT서비스 만족도는 새로운 IT 서비스가 신속히 제공되고 또 시스템에 빠르고 편리한 접근성이 보장될 때 높아진다. 그러므로 신속성을 고려할 때, 많은 시간을 요하는 자세한 시뮬레이션 모델 보다는 때로는 조악하지만 신속한 모델이 필요하다. 본 논문에서는 측정된 성능 데이터를 기초로 Menasce[3]가 제시한 분석적 모델을 검토하고 ARENA[4]를 이용하여 시뮬레이션 모델을 구축하여 검정함으로써 개략적이지만 신속한 성능 분석 및 예측 시뮬레이션 모델을 제시하고자 한다.

시뮬레이션 모델은 분석적 모델이 밝혀내지 못하는 시스템의 동적 요소를 이해하는데 많은 도움을 준다. 신속한 해결책을 요구하는 오늘날 시간이 소요되는 상세하고 정확한 시뮬레이션 모델보다는 간단하지만 빠른 정보를 주는 시뮬레이션 모델이 바람직하며 상세한 모델은 때로는 복잡한 시스템의 구성요소를 잘 못 반영할 수도 있다. 본 논문이 제시하는 ARENA를 이용한 대략적 모델은 측정된 데이터와 접근적으로 일치하며 구성요소의 정보를 변경시켜서 성능을 예측할 수 있다.

ARENA는 범용 시뮬레이션 도구이며 최소 지원 시간단위가 초로서 정교한 컴퓨터 시뮬레이션에는 적합하지 않다고 하지만 시간단위는 결과물에 조정할 수 있으며 실제로 초 단위 이하의 시뮬레이션에 사용된 전례가 있다 [5,6,7].

본 논문의 2장에서는 측정된 데이터에 대한 분석적 모델을 검토하고, 이를 기초로 3장에서는 시뮬레이션 모델을 제시하며, 4장은 시뮬레이션의 결과를 분석하고 5장에서는 결론을 맺는다.

Ⅱ. 분석적 모델

Menasce가 측정한 시스템은 그림 1 [3]과 같이 하나의 프로세서와 두 개의 디스크를 가지고 있으며 세 개의 query와 한 개의 update 트랜잭션이 폐쇄된 경로를 통해 순환하고 있다.

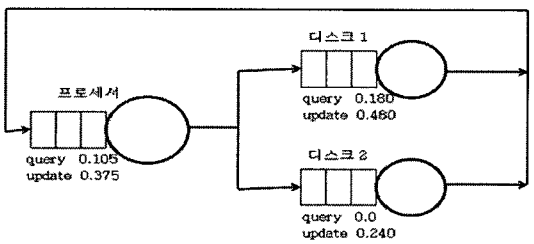

그림 1. 두 개의 클래스를 가진 큐잉 모델
Fig.1 Two-class Queuing Model

측정 기간 1,800초 동안 표 1 [3]과 같은 서비스요구(Service Demand)를 추출할 수 있었고 수행된 트랜잭션 수는 query 7,368개, update 736개 이었다.

표 1. 측정된 값
Table 1. Measurement Data

클래스	서비스요구 (초)			시스템내 트랜잭션 수	수행된 트랜잭션 수
	프로세서	디스크 1	디스크 2		
query	0.105	0.180	0.000	3	7368
update	0.375	0.480	0.240	1	736

시스템 내의 트랜잭션 수는 입력 변수이며 수행된 트랜잭션 수는 측정된 값이다. 서비스요구는 식1과 같이 방문 횟수와 방문당 서비스 시간을 곱한 값이다.

서비스요구 $D = S \times V$ ..... (식1)

여기서 S는 방문당 서비스시간이며 V는 방문횟수이다.

Menasce는 서비스요구가 어떻게 측정되었는가는 밝히지 않고 있지만 대개 성능 모니터 도구를 사용하여 사용률(utilization)을 측정한 후 다음 식2와 같이 사용률 법칙(Utilization Law)을 사용하여 계산한다.

사용률 법칙 $U = X \times D$, 그러므로 $D = U / X$ ··· (식2)

여기서 U는 사용률, X는 처리율, D는 서비스요구이다.

분석적 모델을 구축하기 위해 Menasce는 다음과 같은 몇 가지를 관찰하고 가정 한다.

1) 표 1에서 보는 바와 같이 update 트랜잭션은 디스크 1과 디스크 2를 접근하지만 query 트랜잭션은 디스크 1만 접근한다.
2) 디스크의 서비스 시간은 두 클래스에 대해 모두 동일하다. 그러므로 두 클래스가 디스크 1에 대한 서비스 요구의 차이는 순전히 방문 횟수 때문이다.
3) 디스크의 Read/write 우선순위, 특정 디스크 스케줄링 등은 고려하지 않고 FCFS(First Come First Served) 를 가정하였다.

다중 클래스 모델은 두 클래스를 단일화한 복합 클래스 모델로 변환하여 풀 수 있다. 즉 표 1의 데이터를 사용하여 프로세서와 디스크의 복합 클래스 모델에 대한 서비스 요구를 다음 식3과 같이 구할 수 있다.

프로세서의 복합 서비스 요구 $Dp = (Dpq \times Nq + Dpu \times Nu) / (Nq + Nu)$ ····· (식3)

여기서 Dp는 프로세서의 복합 서비스 요구, Dpq는 프로세서의 query 서비스 요구, Dpu는 프로세서의 update 서비스 요구이며, Nq는 query 트랜잭션의 수, Nu는 update 트랜잭션의 수이다. 표 1의 데이터를 식3에 대입하여 Dp = (0.105 x 7368 + 0.375 x 736) / (7368 + 736) = 0.1295초를 얻는다. 같은 방법으로 디스크 1에 대한 서비스 요구 $Dd1$ = (0.18 x 7368 + 0.48 x 736) / (7368 + 736) = 0.2072초, 디스크 2에 대한 서비스요구 $Dd2$ = (0.24 x 736) / (7368 + 736) = 0.0218초를 얻을 수 있다.

다중 클래스 모델을 복합 클래스 모델로 변형하여 얻은 값은 다중 클래스 모델의 최악의 성능 한계를 나타내므로 다중 클래스 모델로부터 얻은 값이 오차 범위 내에서 맞는지를 판단하는데 사용된다[8].

다중 클래스 모델을 분석적으로 풀기 위해서 도착이론(arrival theorem)이 도입 되어야 한다[9,10]. 도착이론은 "N개의 트랜잭션이 수행되는 폐쇄 모델에서 서비스 센터 i에 도착하는 클래스 r 트랜잭션에게는 마치 $N - 1$개의 트랜잭션이 정지상태의 분포(steady-state distribution)을 이루고 있는 것이 같이 보인다."는 것으로 직관적으로도 동의할 수 있는 이론이다. MVA(Mean Value Analysis)는 트랜잭션이 2개 있을 경우의 답을 1개 있을 때로부터 얻고, 3개 있을 경우를 2개 있을 때의 값으로 얻는 등 도착이론을 반복적으로 적용하여 원하는 N개의 트랜잭션의 경우의 답을 얻는 방법이다[11].

이 과정을 적용하여 Menasce가 다중 클래스 모델로부터 얻은 결과는 표 2 [3]와 같다.

표 2. MVA를 적용하여 얻은 값
Table 2. Results of MVA

클래스	응답시간 (초)			총 응답시간	처리율
	프로세서	디스크 1	디스크 2		
query	0.204	0.529	0.000	0.733	4.093
update	0.704	1.500	0.240	2.444	0.409

Menasce는 update 클래스를 모두 다른 시간대로 옮겨서 4개의 query 트랜잭션만 수행되는 시스템에 대해 처리율 5.275을 얻어서 28.87%의 처리율 향상을 예측하였다.

MVA는 클래스의 수가 증가함에 따라 계산 복잡도가 기하급수적으로 증가하는 단점이 있다. 트랜잭션의 수 N이 대단히 크면 $N - 1$을 N과 같다고 대략적으로 가정할 수 있으므로 반복적으로 수행해야 할 계산을 줄여 대략적 답을 구할 수 있으나, 이 답을 정확한 MVA 결과로 검정해 주어야 하는 모순에 봉착한다. 본 논문에서 제시한 시뮬레이션 모델은 이러한 단점들 없이 신속히 성능을 모델링 할 수 있다.

III. 시뮬레이션 모델

시스템의 구성요소가 큐로 구현되어 있다고 가정할 때, 트

랜잭션의 도착율과 구성요소에서의 처리시간 및 분포만 주어진다면 Little의 법칙을 비롯한 여러 가지의 운용법칙(Operational Law)[12]을 사용하여 분석적 방법으로 컴퓨터 시스템의 성능을 분석하고 예측할 수 있다. 그러나 2장에서 소개한 분석적 접근은 평균값에 의존하고 있으므로 동적인 도착율과 처리시간 분포의 실제 값을 모델링하기 어렵다. PDQ(Pretty Damn Quick)은 도착율과 구성요소에서의 서비스 시간을 가장 보편적이고 타당하다고 여겨지는 지수분포로 가정하여 큐잉 시스템의 성능을 분석하고 예측하는 도구이다[11]. 그러나 ARENA와 같이 그래픽 인터페이스가 있는 시뮬레이션 도구를 사용하면 각종 분포를 사용할 수 있을뿐더러 간단하고 신속하게 시뮬레이션 모델을 구축할 수 있다.

그림 2는 그림 1을 ARENA로 구현한 것이다.

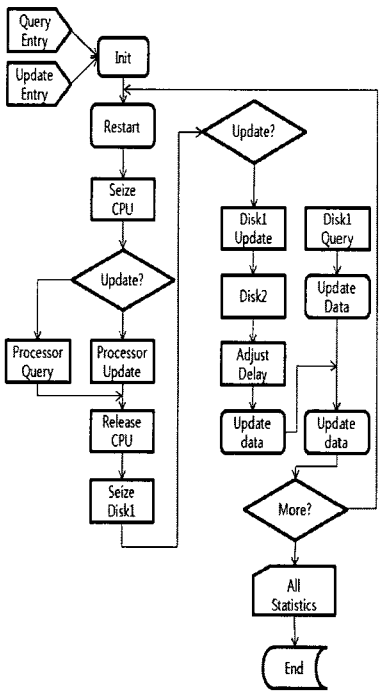

그림 2. 폐쇄 다중 클래스 시뮬레이션 모델

Fig. 2 Simulation Model for a Closed Multi-class System

3개의 query 트랜잭션과 1개의 update 트랜잭션이 "Query Entry" 및 "Update Entry" 프로세스에 의해 생성되고 "Init" 프로세스에 의해 초기화된 후 각 트랜잭션은 그림 1에서 표시한 프로세서와 디스크에 대한 서비스요구를 가지고 1,800초 동안 그림 2의 폐쇄모델을 수행하게 된다. "Update data" 프로세스는 통계에 사용될 데이터를 수집하며, "All Statistics" 프로세스는 최종 결과를 도출한다.

식2의 사용률의 법칙으로부터 100%의 사용률을 가정하면, 초당 처리할 수 있는 최대 트랜잭션 수는 X=1/D이 되므로 폐쇄된 모델에서 3개의 query 트랜잭션이 1,800초 동안 이론적으로 최대 수행할 수 있는 트랜잭션 수는 식4에서와 같이 18,947개이다. 같은 방법으로 1개의 update 트랜잭션은 식5에서와 같이 1,644개이다

이론적 최대 query 트랜잭션 수 Nq-max
= (1800 x 3) / (Dpq + Dd1q + Dd2q) ····· (식4)
= (1800 x 3) / (0.105 + 0.180 + 0.0) = 18,947

이론적 최대 update 트랜잭션 수 Nu-max
= (1800 x 1) / (Dpu + Dd1u + Dd2u) .....(식5)
= (1800 x 1) / (0.375 + 0.48 + 0.24) = 1,644

식4와 식5에서 Dpq는 query 트랜잭션의 프로세서 서비스요구이고, Dd1q는 query 트랜잭션의 디스크 1에 대한 서비스요구이며, Dd2q는 query 트랜잭션의 디스크 2에 대한 서비스요구이다. 또 Dpu는 update 트랜잭션의 프로세서 서비스요구이고, Dd1u는 update 트랜잭션의 디스크 1에 대한 서비스요구이며, Dd2u는 update 트랜잭션의 디스크 2에 대한 서비스요구이다.

측정한 값은 당연히 이론적 최대치 보다 적은 query 7,368회와 update 736회이다. 측정치를 그대로 시뮬레이션 모델에 대입하면 결과가 측정치와 일치하지 않고 query 5,307회, update 1,243회가 된다. 그 이유는 실제 시스템에는 모델링 할 수 없는 숨어있는 요소들이 있기 때문인데, 이 요소들은 각 구성요소에 골고루 분배하든지, dummy 프로세스 등을 가정하여 보정해 주어야 한다[11]. 이dummy프로세스는 큐 등을 사용함으로써 다른 구성요소들의 일을 방해해서는 안 되므로 "delay" 프로세스로만 정의했고, 표 2의 시뮬레이션 모델에서는 "Adjust Delay" 프로세스가 이 역할 하고 있다. 최초의 모델에서 구한 트랜잭션 수가 측정치 보다 적으므로 이를 증가시켜야 하는데 이는 query가 update보다 서비스요구가 작으므로 update 시간을 더 연장시킴으로써 query가 더 많이 활동하게 하여 전체 트랜잭션 수를 증가시킬 수 있다.

"Adjust Delay" 프로세스의 delay값을 1.158초로 선택하면 query와 update 트랜잭션 수 비율을 측정치 비율

1:10과 같은 비율로 만들 수 있다. 이렇게 수정한 모델을 수행하면 query 트랜잭션 수 6,977개, update 트랜잭션 수 697로 비율은 1:10이며 각기 측정치와 5.3%의 오차가 있다.

모든 프로세스의 서비스요구는 지수분포를 갖는다고 가정하였다. 디스크 2는 서비스시간(S)과 방문횟수(V)를 개별적으로 고려하지 않고 서비스요구(D=SV)로 일괄적으로 반영하여 모델을 간단하고 신속하게 구축하였다.

IV. 결 과

표 3은 시뮬레이션의 결과와 MVA로 얻은 값을 비교한 것이다. 3장에서 모델을 수정했을 때 트랜잭션의 수가 측정치와 5.3% 차이가 났고 이 차이가 MVA와 시뮬레이션 결과의 오차에 반영되었다. 표 3은 시뮬레이션 결과가 분석적 모델과 유사한 결과를 얻었음을 보여주고 있다.

표 3. MVA와 시뮬레이션 결과의 비교
Table 3. Comparison of MVA with Simulation

구분		MVA	시뮬레이션	오차
응답시간	query	0.733	0.774	5.6%
	update	2.444	2.582	5.7%
처리율	query	4.093	3.874	5.4%
	update	0.409	0.387	5.4%

측정된 트랜잭션 수와 시뮬레이션 모델을 위해 수정된 트랜잭션 수를 식6의 사용률 법칙(Utilization Law)[3]에 대입하여 얻은 사용률을 시뮬레이션 모델의 결과와 비교하면 〈표 4〉와 같다.

$$Utilization = Throughput \times Service\ Demand \quad \text{(식6)}$$

표 4. 측정치로 계산된 사용률과 시뮬레이션 결과의 비교
Table 4. Comparison of measured data with simulation

구분		서비스 요구	처리율		계산된 사용률 (U = XD)		시뮬레이션 결과 사용률
			측정	수정 모델	측정	수정 모델	
프로세스	query	0.105	4.093	3.876	43.0%	40.7%	
	update	0.375	0.409	0.387	15.3%	14.5%	
계					58.3%	55.2%	56%
디스크 1	query	0.18	4.093	3.876	73.7%	69.8%	
	update	0.48	0.409	0.387	19.6%	18.6%	
계					93.3%	88.4%	88%
디스크 2	update	0.24	0.409	0.387	9.8%	9.3%	9%

표 4에서 보는 바와 같이 측정치를 기초로 한 프로세서의 사용률이 58.3%, 트랜잭션 수를 수정하여 계산한 사용률이 55.2%, 그리고 시뮬레이션의 결과는 56%이다. 디스크 1은 각각 93.3%, 88.4% 및 88%이며 디스크 2는 9.8%, 9.3% 및 9%이다. 시뮬레이션은 수정 모델을 기초로 했으므로 계산된 값과 시뮬레이션 결과가 거의 일치하는 것을 알 수 있다.

Menasce는 update 트랜잭션을 모두 off-peak 시간으로 옮겼을 때, 즉 query 트랜잭션만 4개 수행한다고 가정했을 때의 성능을 표 5의 "계산/분석모델" 행과 같이 예측하고 있다. 이 경우는 그림 2의 시뮬레이션 모델에서 update 경로를 없애거나 "Update?" 프로세스를 모두 query로 가게 하여 간단히 시뮬레이션할 수 있다. 그 결과도 표 5의 "시뮬레이션" 행에 표시하였다.

표 5. query 트랜잭션만 가정했을 때의 처리율 비교
Table 5. Comparison of throughput for query transactions

구분	처리율		오차
	계산/분석모델	시뮬레이션	
측정치 기준	4.093	3.874	5.4%
query만 가정	5.275	5.195	1.5%
증가	28.87%	34.12%	

표 5에서 보는 바와 같이 분석적 모델로 예측된 처리율과 시뮬레이션에서 예측한 처리율의 차이가 1.5%로서 거의 일치함을 알 수 있다.

V. 결 론

속도 중심의 비즈니스 환경에 부응하기 위해 시스템의 성능 분석 및 예측도 신속한 결과가 도출되어야 한다. 본 논문에서는 ARENA를 사용하여 측정된 데이터를 기초로 짧은 시간 내에 시스템의 시뮬레이션 모델을 구축할 수 있으며 그 결과가 MVA 등 분석적 모델로 계산한 값과 큰 차이가 없음을 보였다. 나아가 간단히 모델을 수정하여 새로운 가정에 대한 성능 예측을 할 수 있다. 여기서는 부하 독립(load-independent)인 분리 가능한 형태(product-form)의 폐쇄형 다중 클래스 모델을 시뮬레이션하였다. 본 논문을 응용하여 개방형 모델, 부하 종속(load-dependent) 그리고 분리가 용이하지 않은 형태(non-product-form)의 시뮬레이션에 대해서도 빠르고 간단한 시뮬레이션 모델이 제시될 수 있을 것이다.

참고문헌

[1] A. K. Erlang, "On the Rational Determination of the Number of Circuits," in the Life and Works of A. K. Erlang, 1917

[2] A. A. Scherr, "*An Analysis of Time-Shared Computer Systems*," Cambridge, Mass: MIT Press, 1967

[3] Daniel A. Menasce et al, "*Performance by Design*," Prentice Hall PTR, 2004

[4] W. Dave Kelton et al., "*Simulation with ARENA 3e*," McGraw-Hill, 2003

[5] Yong Soo Kim, "Response Time Simulation for 2tier and 3 tier C/S System," The Korea Society of Computer and Information, vol.9, no.3, pp.45~53, September 2004.

[6] Yong Soo Kim, "Simulation of Storage Capacity Analysis with Queuing Network Models," The Korea Society of Computer and Information, vol.10, no.4, pp. 221~228, September 2005.

[7] Yong Soo Kim, "Coarse-grained Simulation Model for Web Application Performance Analysis," The Korea Society of Computer and Information, vol.13, no.6, pp. 25~31, November 2008.

[8] L. W. Dowdy et al., "Single-class bounds of multiclass queuing networks," ACM, vol.39, no.1, January 1992

[9] M. Reiser and S. Lavenberg, "Mean-value analysis of closed multi-chain queuing networks," ACM, vol.27, no.2, 1980

[10] K. C. Sevick and I. Mitrani, "The distribution of queuing network states at input and output instants," ACM, vol.28, no.2, 1982

[11] Neil J. Gunther, "*Analyzing Computer System Performance with Perl::PDQ*," Springer, 2005

[12] Neil J. Gunther, "*The Practical Performance Analyst*," McGraw Hill, 2000

저 자 소 개

김 용 수

경원대학교 IT대학

컴퓨터공학전공

〈관심분야〉운영체제, 성능분석/관리