처리시간을 고려한 분산시스템의 서비스 품질분석

김정호* · 박종훈**†

* 국방기술품질원
** 대구가톨릭대학교 경영학과

QoS Analysis of a Distributed System Considering the Processing Time

Jung Ho Kim* · Jong Hun Park**†

* Defense Agency for Technology and Quality
** Department of Business Adminstration, Catholic University of Daegu

Key Words : Distributed System, Quality of Service, Processing Time

Abstract

In this paper, we introduce Quality of Service(QoS) analytic model of a distributed system that decentralizes the process nodes performing each task and communicates through a network for cooperation. The model advances a service reliability model of Dai et al.(2003) by means of considering the processing time. The service is assumed to be provided by a centralized heterogeneous distributed system which is composed of some subsystems managed by a control center. The QoS is defined as the probability that a service is provided successfully in an allowed time, we consider the hardware/software reliability and the processing time which include program execution time, data transfer time. We derive the processing time distribution for a required service through convolution of corresponding probability density function. An application example is used to explain the procedure of computing quality of service.

1. 서 론

분산시스템(distributed system)은 과업을 처리하기 위한 프로그램과 파일들이 하나의 대형 시스템에 집중된 것이 아니라, 공간적으로 분산된 여러 개의 하위 시스템(sub-system)에 나뉘어 배치되고, 네트워크를 통하여 상호간에 교신 처리하도록 구성되어 있는 시스템을 의미한다[Kshemkalyani & Singhal, 2008]. 이러한 분산시스템은 다수의 데이터베이스(database)와 프로세스노드(process node)를 다수의 하위 시스템에 나누어 배치함으로써 시스템의 일부가 고장이 나더라도 그 외의 하위 시스템에 존재하는 데이터와 프로그램의 공유가 가능하기 때문에 은행, 군수, 원자력 시스템 등과 같이 한 번의 고장이 큰 손해를 야기 시킬 가능성이 존재하는 분야의 정보시스템 구성에 널리 사용되고 있다[Dai et al., 2003].

이와 같이 분산시스템은 시스템의 안정적 사용에 주요 목적이 있기 때문에, 분산시스템의 성능을 평가하기 위한 핵심 척도로 "분산 환경 하에서, 주어진 서비스 또는 과업을 성공적으로 수행할 확률"로 정의되는[Dai et al., 2003. Hsieh, C. and Hsieh, 2003] 분산시스템의 신뢰성(reliability of distributed system)이 자주 고려되어왔다[2-4, 6, 7, 9-15].

분산시스템의 신뢰성을 측정/평가하기 위한 연구는 Goel & Soenjoto(1981)이 소프트웨어와 하드웨어를 결합하여 성과(performance)라는 개념으로 처음 소개하면서부터 시작하여, 초기에는 주로 하위 시스템 내의 물리적 연결성을 나타내는 Distributed Program Reli-

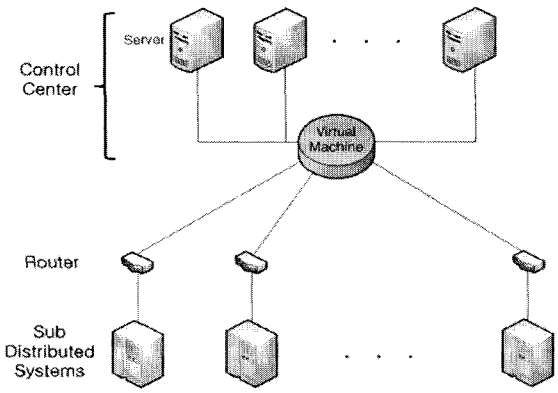
† 교신저자 icelatte@cu.ac.kr

ability(DPR)계산에 대한 연구가 주를 이루었다. 이와 관련하여 Huang et al.(1990)이 분산시스템의 신뢰성 분석을 위한 Fast Spanning Tree(FST) 생성 알고리듬을 사용하여 신뢰성 분석을 시도하였으며, Chen & Huang (1992)은 Fast Reliability algorithm을, Kumar & Agrawal(1993)은 Minimal Fast Spanning Tree(MFST) 방법론을 사용하여 DPR 계산을 시도하였다. 이후 DPR 계산의 효율성을 높이기 위해 다양한 위상의 하위시스템에 적용 가능한 알고리듬(algorithm)들이 Kumar & Agrawal(1993), Chang et al.(2000), Lin et al.(2001) 등에 의해 개발되었다.

이후의 연구들은 DPR 계산을 위한 알고리듬의 연구 성과를 기반으로 하여, <그림 1>에 보이는 바와 같이 각기 다른 기능을 가진 하위시스템들과 이를 관리하는 컨트롤센터(control center)로 구성된 Centralized Heterogeneous Distributed System(CHDS)으로 분산시스템을 모형화하고, 컨트롤센터에서 서비스를 수행하기 위하여 필요로 하는 파일과 프로그램의 가용여부를 하위 시스템의 물리적 신뢰성과 연동하여 고려하는 방향으로 확장되어 왔다. Srinivasan & Jha(1999)는 안전성의 개념을 도입하여 CHDS의 가용성(availability)을 최대화하기 위한 서비스 분배 결정 방법론을 제안하였으며, Lai et al.(2002)와 Wang(2004)은 마코프체인(Markov Chain)을 기반으로 하여 분산시스템의 가용성과 신뢰성을 계산하는 연구를 수행하였다. 특히 Dai et al.(2003)는 CHDS를 대상으로 프로그램 실행시간을 고려하도록 확장하여 서비스가 성공적으로 수행될 확률을 서비스 신뢰성으로 정의하는 분산시스템의 서비스 신뢰성 분석모형을 제시하였다.

이상에서 소개된 바와 같이, 기존 연구들은 분산시스템의 신뢰성이라는 개념에 충실하여 분산시스템의 성능을 평가하고 향상시키기 위한 방향으로 수행되어 왔다. 그러나 정보시스템이나 통신시스템과 같이 특정한 서비스를 제공하는 것을 목적으로 분산시스템이 구성된 경우, 해당 서비스가 제공되는 시간을 무시한 채 서비스를 성공적으로 제공할 수 있는지의 여부만을 고려하는 신뢰성의 개념만으로 분산시스템의 성능을 평가하기에는 부족함이 있다.

쉬운 이해를 위하여, 기차표를 예매하기 위한 예약/발권 서비스를 제공하는 시스템을 가정하자. 해당 시스템은 기차표 예매를 위해 필요한 열차의 일정 및 경로, 열차 좌석, 고객정보 등 기차표 예매를 위해 요구되는 다양한 정보와 좌석 조회 및 선택, 결재, 발권을 위한 여러 프로그램들을 하나의 물리적 대형 시스템에 집중시켜 놓지 않고, 열차정보시스템, 고객정보시스템, 결재시스템과 같은 다양한 개별 시스템에 관련 정보와 프로그램을 분산 배치하였으며, 사용자가 포탈(portal)을 통해 원하는 서비스를 시도할 경우 각각의 개별 시스템에 분산되어 있는 프로그램과 파일 및 데이터베이스가 네트워크를 통하여 상호간에 교신 처리됨으로써 해당 서비스를 제공하도록 구성되어 있다.



<그림 1> CHDS의 구조

예를 들어 임의의 사용자가 포탈을 통해 기차표 예매/발권서비스를 제공받으려는 시점에 각각의 하위시스템이나 통신 네트워크의 물리적인 문제는 존재하지 않지만 접속자가 많아서 프로그램을 호출하고 처리하는 시간이 급격히 늘어나 좌석조회 및 선택, 결재 및 발권을 하기 위해 대기하는 시간이 매우 길어지는 경우, 혹은 일부 하위시스템에 과부하가 걸려 해당 하위시스템에서의 병목현상이 발생하여 특정 프로그램의 실행이 늦어지고 사용자화면이 멈춘 듯이 느껴지는 경우에 다수의 사용자는 이를 서비스 실패로 간주하여 진행 중인 서비스를 종료하고 다시 시도할 것이며, 해당 서비스의 성능에 대하여 불만을 가지게 될 것이다.

위의 경우는 하위시스템이나 네트워크의 물리적인 문제가 존재하지 않으므로 서비스가 늦기는 하지만 언젠가는 제공될 것이기 때문에 서비스 제공의 성공 여부만을 고려하는 신뢰성의 입장에서는 성공으로 간주될 수 있지만, 시간을 고려한 소비자의 입장에서는 실패로 간주될 수 있으며, 이러한 문제는 여러 서비스의 연속적 실행을 통해 비즈니스 프로세스를 수행하는 대다수의 정보시스템에서 존재할 수 있다.

따라서 서비스를 제공하는 것을 목적으로 하는 분산시스템의 성능을 평가하는 경우, 서비스 제공의 성공

여부만을 고려하는 신뢰성 척도보다는 요청된 서비스가 일정시간 이내에 성공적으로 제공되는가의 여부를 고려한 서비스 품질(Quality of Service; QoS)의 개념이 더욱 실용적이라 할 수 있다. 그러나 분산시스템의 성능평가 척도로써 서비스 품질에 대한 개념은 Campbell & Keshav(1998)에 의해 그 개념 정도만이 소개되었을 뿐, 관련된 후속 연구가 이루어지지 않고 있다. 그러나 Campbell & Keshav(1998)은 분산시스템에서의 서비스 품질은 시스템의 양끝단간(end to end) 또는 응용프로그램간(application to application) 문제이며, 끝단 시스템(end system)에서의 디스크(disk)나 쓰레드(thread)의 운용계획 및 네트워크에서의 패킷(packet)과 호(cell)의 정체(congestion)등에 대한 조절이 수반되는 문제라고 소개함으로써 단순히 서비스의 성공적 제공 외에 적절한 시간의 조절 및 운용에 의한 요소가 고려되어야 함을 언급하였다.

따라서 본 연구에서는 분산시스템의 서비스 품질 척도를 끝단의 시스템에 접해 있는 고객의 입장에서 고려하여 "요청된 과업이나 서비스가 일정시간 이내에 성공적으로 제공될 확률"이라고 정의하였으며, Dai et al.(2003)의 CHDS 신뢰성 분석모형을 확장하여 프로그램 실행시간과 데이터 전송시간을 반영하여 허용시간 이내에 사용자가 요청한 서비스가 성공적으로 수행될 확률을 고려하는 서비스 품질 분석모형을 제시하였다. 이를 위해 처리시간의 총시간 분포 함수를 정의하고, 함수의 콘볼루션(convolution)을 사용하여 처리시간의 분포를 계산하고, 확률을 도출하였으며, 그 결과를 바탕으로 특정 조건과 분산시스템의 구조가 서비스 품질에 미치는 영향을 분석하였다.

2장에서는 기본 가정과 관련기호, 버츄얼머신 (Virtual Machine ; VM)의 가용성과 하위시스템의 신뢰성, 그리고 서비스가 허용시간 안에 제공될 확률을 반영한 CHDS의 서비스 품질 분석모형을, 3장에서는 그에 대한 수치예제를 제공하여 모형의 이해를 도왔다. 마지막으로 4장에서는 결론 및 추후 연구 과제를 기술하였다.

2. CHDS의 서비스 품질분석

Dai et al.(2003)는 서로 다른 형태를 가진 하위시스템들이 네트워크로 연결되어 작동하는 컴퓨터들로 구성되어 있으며, 버츄얼머신을 통하여 프로그램과 파일을 공유하고 라우터로 연결되어 있어 버츄얼머신의 통제 하에 서비스를 제공하는 형태의 CHDS를 대상으로 신뢰성분석 모형을 제시하였다. 그의 연구에서는 분산시스템의 신뢰성을 하위시스템의 신뢰성과 버츄얼머신의 가용성의 곱으로 표현하였다. 하위시스템의 신뢰성은 다수의 프로세스노드에서 실행되고 있는 프로그램의 성공적 실행과 특정 노드에서 요구한 파일의 접근 성공여부로 정의하였으며, 버츄얼머신의 가용성은 버츄얼머신 내부의 파일과 프로그램의 사용 가능 확률로 정의하였다.

그러나 Dai et al.(2003)의 연구에서는 하위시스템의 작동여부와 프로세스노드에서 실행되는 프로그램의 동작 및 파일의 접근 등의 과정에서 지연될 수 있는 서비스 처리시간을 고려하지 않고 있다. 따라서 본 연구에서는 고객의 입장에서는 중요한 품질 평가 요소가 될 수 있는 서비스의 처리시간을 고려하기 위하여 Dai et al.(2003)의 분석모형을 기반으로 하여 프로그램 실행시간과 데이터 전송시간을 고려하도록 확장하여 분산시스템의 서비스 품질 분석 모형을 제안하였으며, 제안된 모형에서 사용된 기본 가정과 관련 기호는 다음과 같다.

<가정>

(1) 버츄얼머신 내부에 프로그램의 선행단계와 필요한 파일에 대한 기록이 코딩되어 있다.
(2) 프로세스노드의 프로그램과 데이터들은 반드시 버츄얼머신을 거쳐 다른 프로세스노드로 이동한다. 단, 노드 내의 프로그램이 필요한 파일이 같은 노드에 있는 경우 파일을 바로 사용할 수 있다.
(3) 분산시스템 내의 모든 프로세스노드는 고장이 나지 않는다.
(4) 버츄얼머신과 프로세스노드 내의 파일과 프로그램이 독립적으로 작동하며, 이를 통해 계산한 DSR_l 값들은 모두 독립이다.
(5) 프로그램은 각각 다른 모수 값을 가지는 실행시간 분포를 따른다.
(6) 분산시스템 내의 링크들은 동일한 모수 값을 가지는 이동시간 분포를 따른다.

<기호>

$A(t)$: 시간 t에서 버츄얼머신의 가용성
DSR_l : 하위 시스템 l의 물리적 신뢰성
$P_0(t)$: 시간 t에서 버츄얼머신이 작동할 확률
$P_1(t)$: 시간 t에서 버츄얼머신이 작동하지 않을 확률
t_i : 서비스의 시작시간(initial time)

T_{bf}^{j} : j번째 프로그램이 버츄얼머신 내의 파일을 필요로 하는 시간

T_{bp}^{k} : 버츄얼머신 내부의 k번째 프로그램이 실행을 시작하는 시간

T_{ex}^{k} : 버츄얼머신 내부의 프로그램 k의 실행시간

T_{g}^{m} : 프로그램 m의 수행에 대한 허용시간

e_a : 두 개의 프로세스노드를 연결하는 링크 a

f_b : 프로그램 실행을 위한 파일 b

p_c : 프로그램 c

X_{e_a} : 링크 a의 데이터 전송시간

X_{p_c} : 프로그램 c의 실행시간

$P(p_m)$: 프로그램 m 실행을 위한 m을 포함한 선행 프로그램들의 집합

$F(p_m)$: $P(p_m)$중 첫 번째 선행프로그램이 필요로 하는 파일들의 집합

$L(f_b)$: 버츄얼머신에서 f_b가 있는 최단거리 노드까지의 주경로(critical path)에 속한 링크들의 집합

$L(p_c)$: 버츄얼머신에서 p_c가 있는 최단거리 노드까지의 주경로(critical path)에 속한 링크들의 집합

$Q_s^m(T,t_i)$: 서비스 요청시간이 t_i일 때, 프로그램 m이 T시간 내에 처리될 확률, 즉 서비스 품질함수

2.1 CHDS의 서비스 품질 분석모형

분산시스템에 요청된 서비스가 허용시간 내에 성공적으로 제공되기 위해서는

1) 버츄얼머신내에 있는 파일과 프로그램을 요청하는 서비스가 발생할 때 사용가능,

2) 요청된 서비스와 관련되어 하위시스템에서 실행되는 프로그램들이 성공적으로 수행,

3) 허용시간 내에 서비스 제공을 완료 해야 한다. 버츄얼머신 내에 있는 파일과 프로그램이 서비스 요청 시 사용가능하기 위해서는 파일이 필요한 시점과 프로그램이 실행되는 동안 버츄얼머신이 작동하고 있어야 한다. 따라서 버츄얼머신 내에 있는 파일과 프로그램의 요청 시 사용 가능할 확률은 파일이 요구되는 시점에서의 버츄얼머신의 순간가용성과 프로그램이 실행되는 동안의 버츄얼머신의 구간 평균가용성으로 표현될 수 있다. 따라서 프로그램 k가 버츄얼머신 내의 파일을 필요로 하는 시간이 T_{pb}^{k}이고, 그 시작시간과 실행시간이 각각 T_{pb}^{k}와 T_{ex}^{k} 일 때 버츄얼머신 내의 파일이 서비스 요청 시 사용가능할 확률을 $P_f(k)$, 프로그램이 서비스 요청 시 사용가능할 확률을 $P_{pr}(k)$로 정의하면, 각각은 다음과 같이 표현할 수 있다[Dai et al., 2003].

$$P_f(k)=A\left(T_{bf}^{k}\right),\ k=1,2,... \tag{1}$$

$$P_{pr}(k)=\int_{T_{bp}^{k}}^{T_{bp}^{k}+T_{ex}^{k}} A(t)dt/T_{ex}^{k} \quad k=1,2,... \tag{2}$$

만약 성공적인 서비스가 제공되기 위해 필요한 버츄얼머신 내에 있는 파일을 사용하는 프로그램이 J개이고, 버츄얼머신 내에 있는 프로그램이 K개라면 해당 서비스를 위한 버츄얼머신의 가용성은 식 (3)과 같이 표현된다.

$$\prod_{j=1}^{J}P_f(j)\prod_{k=1}^{K}P_{pr}(k)=\prod_{j=1}^{J}A(T_{bf}^{j})\cdot\prod_{k=1}^{K}\left(\int_{T_{bp}^{k}}^{T_{bp}^{k}+T_{ex}^{k}}A(t)dt/T_{ex}^{k}\right) \tag{3}$$

가정 (1)에 의해 특정 서비스가 제공되기 위해 필요한 프로그램의 선후관계와 필요한 파일에 대한 기록은 이미 버츄얼머신 내에 존재하기 때문에, 주경로 분석법(Critical Path Method)을 통해 데이터 전송시간과 프로그램 실행시간을 더하여 파일과 프로그램의 요청시간, 시작시간을 계산할 수 있다.

요청된 서비스와 관련된 하위시스템에서 실행되는 프로그램들이 성공적으로 수행되기 위해서는 하위시스템들이 물리적으로 동작을 하고 있어야 하며, 그 확률은 가정 (3), (4)에 의해 하위시스템의 신뢰성인 DSR_i을 의미한다. 따라서 N개의 하위시스템으로 구성되어 있는 분산시스템에 요청된 서비스가 성공적으로 수행될 확률은 $\prod_{i=1}^{N}DSR_i$이다.

이상의 Dai et al.(2003)의 연구결과를 처리시간까지 고려하도록 확장함으로써 사용자 관점에서 분산시스템의 성능을 평가할 수 있는 서비스 품질 분석모형을 제시하였으며, 이를 위하여 허용시간 내에 서비스 제공을 완료할 확률을 도출하는 과정이 필요하다.

허용시간 내에 서비스 제공을 완료하기 위해서는 임의의 프로그램 m이 성공적으로 수행될 때까지의 총시간과 그 분포의 계산이 필요하다. 링크의 데이터 전송시간(X_{e_a})과 각각의 프로그램 실행시간(X_{p_c})들이 가정

(5), (6)을 만족하는 일반 분포를 따른다면, 프로그램 m이 성공적으로 수행될 때까지의 총시간 Y_m은 식 (4)와 같이 표현할 수 있다.

$$Y_m = 2\underset{f_b \in F(p_m)}{Max}\left\{\sum_{e_a \in L(f_b)} X_{e_a}\right\} + \sum_{p_c \in P(p_m)} T_{p_c} + 2\sum_{p_c \in P(p_m)}\sum_{e_a \in L(p_c)} X_{e_a} \quad (4)$$

식 (4)에서 $2\underset{f_b \in F(p_m)}{Max}\left\{\sum_{e_a \in L(f_b)} X_{e_a}\right\}$은 프로그램 m의 실행을 위한 첫 번째 선행프로그램이 필요로 하는 파일들의 집합인 $F(p_m)$중에서 해당 파일까지의 링크거리가 가장 긴 파일을 호출하여 첫 번째 선행프로그램이 있는 노드로 전송하는데 소요되는 시간을 의미한다. 일반적으로 데이터 전송시간보다 프로그램 실행시간이 훨씬 길기 때문에 이후 선행 프로그램의 필요한 파일의 전송시간은 프로그램의 실행시간 안에 포함되므로 고려하지 않아도 된다. 식 (4)의 두 번째 합부분인 $\sum_{p_c \in P(p_m)} T_{p_c}$에서 $P(p_m)$은 프로그램 m을 포함한 프로그램 m이 실행되기 전에 필요한 선행 프로그램들의 집합이며, 프로그램 m까지의 주경로에 속한 프로그램들을 의미한다. 즉, 식 (4)의 두 번째 합 부분은 프로그램 m이 성공적으로 수행되기 위해 필요한 모든 프로그램들의 실행시간의 합을 표현한 것이다. 또한, 식 (4)의 세 번째 합부분인 $2\sum_{p_c \in P(p_m)}\sum_{e_a \in L(p_c)} X_{e_a}$은 $P(p_m)$의 원소인 프로그램 p_c가 필요로 하는 데이터 전송시간과 실행 후 얻게 된 결과 값 전송시간의 합을 의미한다. p_c의 실행에 필요한 결과 값을 가져오는 시간과 프로그램 p_c의 실행결과 값을 다시 버츄얼머신에 돌려주는 시간을 동시에 고려해야 하므로 2를 곱해준다. 식 (4)를 사용하여 프로그램 m의 성공적인 실행결과를 제공받는데 걸리는 총 처리시간을 구할 수 있으며, 이를 통해 프로그램 m이 허용시간 내에 성공적으로 수행될 확률을 다음과 같이 계산할 수 있다.

$$P(Y_m < T_g^m) = \int_0^{T_g} f_{Y_m}(t)dt = \int_0^{T_g} f_{S_n}(s) \bullet f_{T_1+T_2+\cdots+T_k}(t-s)ds \quad (5)$$

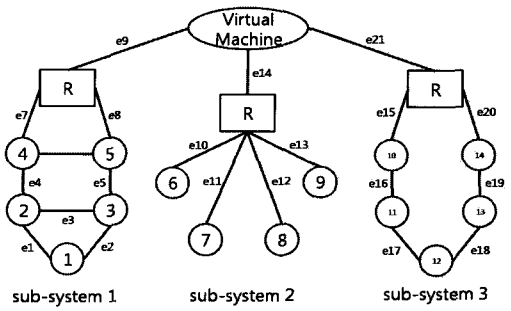
식 (5)를 통해, 데이터 전송시간과 프로그램 실행시간이 일반분포를 따를 때, 특정 프로그램이 허용시간 안에 성공적으로 제공될 확률은 콘볼루션(convolution)을 통해 계산할 수 있다. 이를 종합하면, 본 연구에서 제안하는 시작시간 t_i에 대한 분산시스템 서비스 품질 모형은 식 (6)과 같이 프로그램 완료까지의 허용시간과 해당 프로그램의 수행을 요청한 시간과의 함수로 표현된다.

$$Q_s^m(T_g^m, t_i) = \prod_{j=1}^{J} P_f(j) \prod_{k=1}^{K} P_{pr}(k) \prod_{i=1}^{N} DSR_i \bullet P(Y_m < T_g^m) \quad (6)$$

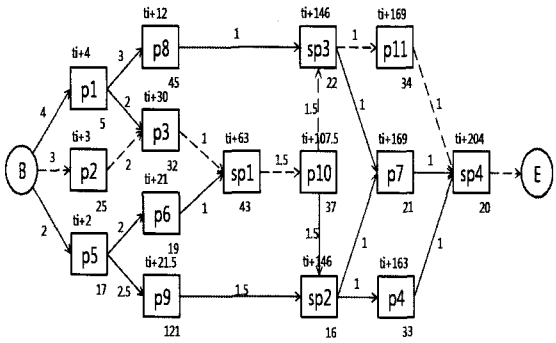
즉, 서비스 품질은 CHDS의 버츄얼머신의 파일과 프로그램의 가용성, $\prod_{i=1}^{N} DSR_i$, 그리고 프로그램 m이 허용시간 안에 성공적으로 수행될 확률의 곱으로 계산할 수 있으며, 버츄얼머신의 하드웨어 신뢰성과 프로그램, 파일의 소프트웨어 신뢰성 및 특정 프로그램에 대한 총 처리시간을 고려한 임의의 시작시간(t_i)에서의 서비스 품질을 계산할 수 있다.

3. 수치예제

제안한 수식을 적용한 CHDS의 서비스 품질 분석을 위해 Dai et al.(2003)의 연구에서 예시로 사용된 CHDS 구조를 참조하였다. 예제시스템은 <그림 2>에서와 같이 버츄얼머신을 통해 통제되는 3개의 하위시스템으로 이루어진 구조이며, 하위시스템은 각각 네트워크, 별, 원형 위상을 따르며 R은 라우터를 의미한다. 또한, <표 1>에는 각 노드별로 설치된 프로그램과 파일에 대한 데이터가 주어졌으며, <표 2>에는 특정 프로그램이 필요로 하는 파일, 선행프로그램, 프로그램별 실행시간이 주어져 있다.



<그림 2> 수치예제 시스템의 분산구조



<그림 3> 예제의 프로세스 다이어그램과 주경로

<표 1> 각 노드별로 설치된 프로그램과 파일

Node	Program	Files
1	p1	f1, f5
2	p4	f1, f2
3	p2, p3	f2, f5
4	p2, p3	f2, f5
5	p4	f3, fF6
6	p5, p7	f6
7	p6	f7, f8, f9
8	p7	f7, f8, f9
9	p5, p6	f6
10	p8, p11	f10, f11, f12
11	p9	f11
12	p10	f10
13	p9, p10	f12
14	p8,p11	f10, f11, f12
VM	sp1, sp2, sp3, sp4	f4, f13, f14

<그림 3>은 <표 2>를 참고하여 나타낸 선행관계 흐름 그래프(flow graph)이며, box 밑의 숫자는 실행시간, box 위의 숫자는 프로그램 시작시간, 화살표 위의 숫자는 데이터 전송시간이며, 점선으로 주경로를 표시한다.

3.1 버츄얼머신의 가용성 함수와 총 처리시간 분포

본 장에서는 버츄얼머신의 고장시간의 분포가 형상모수(shape parameter)가 β, 척도모수(scale parameter)가 α인 와이블(Weibull)분포를 따른다고 가정하였으며, 이때 사용된 고장률 함수는 다음과 같다.

$$\lambda(t)=\alpha\beta \cdot e^{-\beta t} \quad (7)$$

<표 2> 프로그램의 실행을 위해 요구되는 파일과 선행프로그램 그리고 그 실행시간

Program	Required files	Precedent programs	Average Execution time
p1	f1, f2, f3	-	5
p2	f2, f4, f6	-	25
p3	f1, f3, f5	p1, p2	32
p4	f1, f2, f4, f6	sp1, sp2	33
sp1	f6	p3,p6	43
p5	-	-	17
p6	f6, f13, f9	p5	19
p7	f6, f8	sp2, sp3	21
sp2	f2, f11	p9, p10	16
p8	-	p1	45
p9	f11, f12	p5	121
p10	f11, f14	sp1	37
sp3	f3, f8	p8, p10	22
p11	f14, f10, f12	sp3	34
sp4	f5, f12	p4, p7, p11	20

수치예제는 버츄얼머신의 고장률 함수의 모수가 $\alpha=10$, $\beta=0.01$임을 가정하여 수행되었으며, 버츄얼머신에 고장이 발생하는 경우, 즉시 수리가 시작되고 수리시간은 $\mu=0.5$인 지수분포를 따른다고 가정하였다.

버츄얼머신의 가용성을 계산하기 위하여 Welke et al.(1995)의 연구에서와 같이 버츄얼머신이 작동하는 상태를 0, 작동하지 않는 상태를 1이라고 정의하고, 마르코프 프로세스 구성하여 모델링하면, t시점에 버츄얼머신이 작동할 확률인 $P_0(t)$와 작동하지 않을 확률인 $P_1(t)$을 구할 수 있으며, 이를 Kolomogorov's differential equation을 통해 계산하면 $P_0(t)$인 식 (8)을 구할 수 있다.

$$P_o(t)=\left[\int_0^t \mu e^{(\mu x-ae^{-bx})}dx+1/e^a\right]e^{-\mu t+ae^{-bt}} \quad (8)$$

이때, $P_0(t)$를 버츄얼머신의 가용성 함수 $A(t)$라고 표현할 수 있으며, 식 (3)에 대입하면 버츄얼머신의 가용성을 계산할 수 있다.

하위시스템의 물리적 신뢰성의 경우 <그림 2>의 CHDS 구조에 각각의 링크가 사용가능할 확률이 0.9라고 가정하면, Kumar & Agrawal(1993)의 GEAR algorithm에 의해 하위시스템 1의 *DSR*은 0.9496, 하위시스템 2의 *DSR*은 0.8817, 하위시스템 3의 *DSR*은 0.9068로 계산된다.

총 처리시간 분포 함수를 계산하기 위하여 버츄얼머신 내의 모든 링크가 동일한 $\lambda=0.5$인 지수분포를 따르며, 모든 프로그램들이 각각 다른 λ_i (<표 2> 참조)를 가지는 지수분포를 따른다고 가정하여 식 (4)의 분포를 따르는 확률밀도함수를 계산하면 식 (9)과 같다. 이때

$Y_m=2Max_{f_b\in F(p_m)}\left\{\sum_{e_a\in L(f_b)}X_{e_a}\right\}+2\sum_{p_c\in F(p_m)}\sum_{e_a\in L(p_c)}X_{e_a}$은 hyper-exponential 분포를, $\sum_{p_c\in F(p_m)}T_{p_c}$는 hypo-exponential 분포를 따르는 변수이다.

$$\begin{aligned}f_{Y_m}&=\int_0^t f_{S_a}(s)\bullet f_{T_1+T_2+\cdots+T_k}(t-s)ds\\&=\int_0^t\sum_{i=1}^k C_{i,k}\lambda_i e^{-\lambda_i(t-s)}\bullet\lambda e^{-\lambda s}\frac{(\lambda s)^{n-1}}{(n-1)!}ds\\&=\sum_{i=1}^k B_{i,k,n}\left[\sum_{j=0}^{n-1}\left\{-t^j\bullet\frac{(n-1)!e^{-(\lambda-\lambda_i)t}}{j!(\lambda-\lambda_i)^{n-j}}\right\}+\frac{(n-1)!}{(\lambda-\lambda_i)^n}\right]\\&,where\quad C_{i,k}=\prod_{j\neq i}\frac{\lambda_j}{\lambda_j-\lambda_i},B_{i,k,n}=\frac{\lambda^n C_{i,k}\lambda_i e^{-\lambda t}}{(n-1)!}\end{aligned} \quad (9)$$

프로그램 m이 성공적으로 실행될 때, 소요되는 총 처리시간을 구하는 알고리듬을 간략히 기술하면 다음과 같다.

단계 0 : 주경로 분석의 순서에 의해 $P(p_m)$의 프로그램에 번호($d=1,2,...,q$)를 부여

단계 1 : 총 처리시간을 초기화 sum_time=0

단계 2 : $F(p_m)$의 f_b중 $L(f_b)$의 합이 가장 큰 파일의 호출시간과 전송시간을 더함

sum_time=sum_time+ $2Max_{f_b\in F(p_m)}\left\{\sum_{e_a\in L(f_b)}X_{e_a}\right\}$

단계 3 : d번째 프로그램이 있는 노드에 필요한 결과값 혹은 파일을 전송한 후 프로그램을 실행시키고, 그 결과 값을 버츄얼머신으로 보낸 후 이를 q번 반복

for d=1:q find the nearest node to VM which includes d^{th} program if the program is in VM

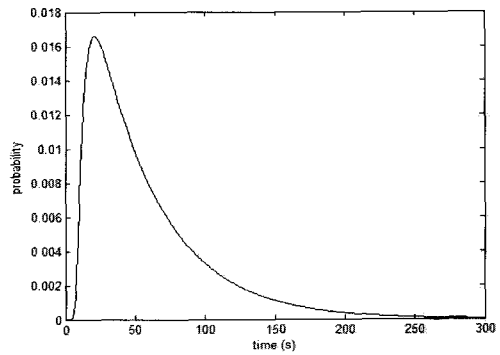
sum_time=sum_time+ T_{p_d} else

sum_time=sum_time+ $2\sum_{p_c\in F(p_m)}\sum_{e_a\in L(p_c)}X_{e_a}$

end

단계 4 : 총 처리시간을 계산한다.

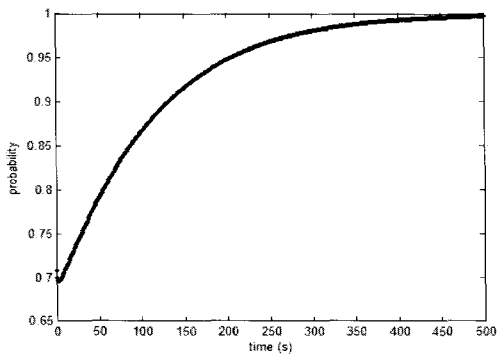
위의 알고리듬과 <그림 3>을 참고하여 p8의 총 처리시간을 계산하면 총 처리시간은 $\{X_{e9}+X_{e7}+X_{e7}+X_{e9}+X_{e9}+X_{e7}+X_{e4}+X_{e1}+T_{ex}^1+X_{e1}+X_{e4}+X_{e7}+X_{e9}+X_{e21}+X_{e15}+T_{ex}^8+X_{e15}+X_{e21}\}$이며, p8의 총 처리시간 분포의 확률밀도함수는 <그림 4>와 같은 형태를 보인다.



<그림 4> 처리시간 확률밀도함수

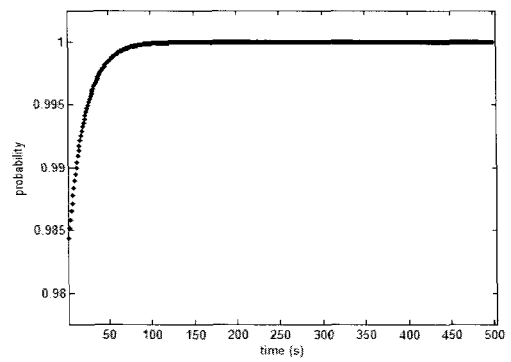
3.2 서비스 품질 함수분석

버츄얼머신 내에 존재하는 파일은 f4, f13, f14이며, 이 파일을 사용하는 프로그램은 {p2, p4, p6, p10, p11}이다. <그림 3>을 참고하여 T_{bf}^j를 구하면 $\{t_i+3, t_i+21, t_i+107.5, t_i+163, t_i+169\}$이며, 이 값을 가지고 버츄얼머신의 파일 가용성을 계산하면 <그림 5>의 형태를 따름을 확인할 수 있다. <그림 5>에서 분산시스템 가동 이후 초반에 가용성이 약간 떨어진 후 다시 상승함을 확인할 수 있는데, 이는 아이덴티파이드 버그(identified bug) 때문이라고 알려져 있다[Dai et al., 2003].



<그림 5> 버츄얼머신 내 파일의 가용성

버츄얼머신 내에 있는 프로그램인 {sp1, sp2, sp3, sp4}의 T_{bp}^{k}는 $\{t_i+63,\ t_i+146,\ t_i+146,\ t_i+204\}$이다. T_{ex}^{k}인 {43, 16, 22, 20}과 함께 식 (4)에 대입하여 버츄얼머신의 프로그램 가용성을 계산하면, <그림 6>의 형태를 갖는다.



<그림 6> 버츄얼머신 내 프로그램의 가용성

프로그램의 실행시간이 파일 전송시간에 비하여 대체로 길기 때문에 $\prod_{k=1}^{K} P_{pr}(k)$의 값이 $\prod_{j=1}^{J} P_f(j)$보다 전송속도에 영향을 적게 받아 안정적임을 확인할 수 있다.

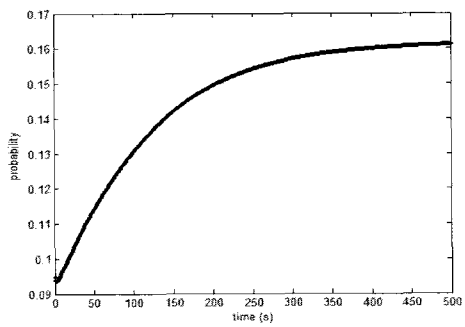
CHDS의 DSR값은 하위시스템의 DSR값들을 곱하여 계산할 수 있으며, 각각의 링크가 사용가능할 확률이 0.9라고 가정하면, CHDS의 DSR값은 $\prod_{i=1}^{3} DSR_i = 0.9496 \times 0.8817 \times 0.9068 = 0.7592$이며, 이는 버츄얼머신의 가용성이 1이 될 때, CHDS의 서비스 신뢰성은 0.7592로 수렴하게 됨을 의미한다.

식 (9)를 이용하여 주경로의 가장 나중에 있는 버츄얼머신 내의 sp4가 $T_g^{sp_4} = 220$안에 성공적으로 서비스를 제공할 확률을 계산하면 $P(Y_{sp4} < 220) = \int_0^{220} f_{Y_{sp4}}(t)dt = 0.2132$이다. 즉 CHDS의 모든 하위 시스템과 버츄얼머신이 작동가능해도 sp4가 220초 안에 처리될 확률은 0.2132라는 것을 의미하며, 만약 고객이 sp4에 해당되는 서비스가 220초 내에 처리되기를 원한다면, 현재의 시스템이 물리적으로 완전히 작동되고 있더라도 고객의 요구사항을 만족시켜줄 확률은 0.2132에 불과함을 의미한다. 물론 sp4에 대한 허용시간을 크게 설정할 경우 확률 값, 즉 고객이 제시간에 서비스 받을 확률은 증가하게 된다.

이상의 과정을 통해 주어진 조건하에서 서비스의 시작시간 t_i에 따른 서비스 품질을 계산할 수 있음을 확인하였다. <그림 7>은 버츄얼머신이 구동된 후 임의의 시작시간 t_i에 서비스가 요청되고, 프로그램의 수행 순서에 따라 프로그램 sp4가 허용시간(220초) 안에 성공적으로 제공될 확률(식 (10))을 도시한 것이다. <그림 7>에서 가로축은 서비스 요청시간 t_i를 세로축은 서비스가 220시간 내에 제공될 확률을 표시하고 있다.

$$Q_s^{sp4}\left(T_g^{sp4}=220, t_i\right) = \prod_{i=1}^{3} DSR_i \prod_{j=1}^{5} P_f(j) \prod_{k=1}^{4} P_{pr}(k) \cdot P(Y_{sp4} < 220) \quad (10)$$

<그림 7>을 사용하여 서비스의 허용시간이 주어진 상황에서 시작시간의 변화에 따른 서비스 성공여부의 변화를 확인할 수 있으며, 가동 초기에는 버츄얼머신의 파일의 identified bug 때문에 CHDS의 서비스 품질이 잠시 떨어졌다가 다시 상승함을 확인할 수 있다.

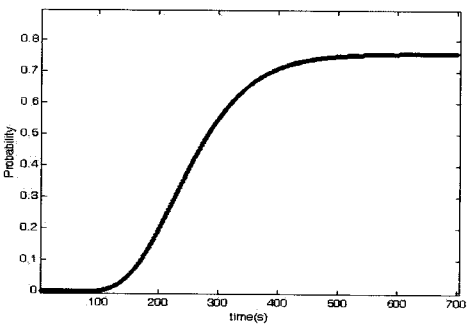


<그림 7> $T_g^{sp4} = 220$일 때 서비스 시작시간에 따른 서비스 품질함수

또한, 서비스의 요청 시작시간 t_i가 주어졌을 때, 시간에 따른 서비스 성공확률의 변화를 확인하기 위하여 $t_1 = 100$인 경우, 허용시간의 변화에 따른 서비스 품질, 즉, 식 (11)을 계산하였다.

$$Q_s^{sp4}\left(T_g^{sp4}, t_i = 100\right) = \prod_{i=1}^{I} DSR_i \prod_{j=1}^{J} P_f(j) \prod_{k=1}^{K} P_{pr}(k) \cdot P\left(Y_{sp4} < T_g^{sp4}\right) \quad (11)$$

<그림 8>은 sp4를 사용해야 하는 서비스가 버츄얼머신이 구동되고 100초가 지난 뒤에 요청되었을 때, 시간에 따른 서비스가 성공할 확률(식 (11))을 도시한 것이다. 이 값들은 sp4의 총 처리시간의 누적분포함수 형태를 따르고 있으며, 일정시간이 경과한 후 $\prod_{i=1}^{3} DSR_i$값인 0.7592로 수렴함을 확인할 수 있다. 서비스의 총 처리시간(Y_m)은 사용자의 대기시간과 동일하기 때문에 식 (11)을 통해 시스템의 관리자는 임의의 시작시간이 주어진 특정 서비스에 대해 분산시스템의 물리적 신뢰성과 버츄얼머신의 가용성을 고려한 일정 수준 이상의 서비스 품질을 제공하기 위한 대기시간을 설정하여 사용자에게 제공, 확인시킬 수 있다.



<그림 8> $t_i = 100$일 때 서비스를 특정시간 내에 제공할 확률

4. 결 론

본 논문에서는 CHDS의 서비스 품질을 일정 시간 안에 서비스를 성공적으로 제공할 수 있는 확률로 정의하였다. 즉 분산시스템의 서비스 품질 계산에 처리시간이라는 새로운 척도를 포함시킴으로써 시스템의 관리자와 사용자에게 보다 정확한 서비스 품질을 확인할 수 있도록 제시하였다. 실제 처리시간이라는 척도를 고려한 서비스 품질 값이 버츄얼머신의 가용성과 하위시스템의 물리적 연결성만을 고려한 서비스 신뢰성 값보다 작거나 같음을 확인하였으며, 이는 관리자가 처리시간을 고려한 서비스 품질을 계산할 때 처리시간을 고려하지 않은 서비스 신뢰성과 동일한 수준을 유지하기 위하여 DSR이나 파일, 프로그램 가용성을 높여야 한다는 것을 의미한다.

확장된 서비스 품질 함수를 통해 분산시스템의 관리자는 사용자의 서비스 요청에 대해 허용시간 안에 서비스가 성공적으로 제공될 확률을 확인 및 제시할 수 있으며, 분산시스템의 사용자는 위의 확률을 통해 허용시간과 서비스 품질을 고려하여 서비스의 진행상황을 확인하고, 서비스 취소여부를 결정할 수 있다.

본 연구는 CHDS의 노드가 고장 나지 않음을 가정한 서비스 품질 분석을 하였으나, 버츄얼머신뿐만 아니라 노드도 고장이 발생할 수 있음을 가정하여 노드의 가용성을 포함한 서비스 품질 분석이 추후 연구로 가능할 것으로 판단된다.

참고문헌

[1] Campbell, A. T. and Keshav, S. (1998), "Quality of service in distributed systems", *Computer Communications*, Vol. 21, No. 4, pp.291-293.

[2] Chang, M.S, Chen, D.J., Lin, M.S. and Ku, K.L. (2000), "The Distributed Program Reliability Analysis on Star Topologies", *Computers and Operations Research*, Vol. 27, 129-142.

[3] Chen, D .J. and Huang, T .H. (1992), "Reliability Analysis of Distributed Systems Based on a Fast Reliability Algorithm", *IEEE Transaction on Parallel and Distributed Systems*, Vol. 3, No 2, pp. 139-154.

[4] Dai, Y. S., Xie, M., Poh, K. L. and Liu, G. Q (2003), "A study of service reliability and availability for distributed systems", *Reliability Engineering and System Safety*, Vol. 79, No. 1, pp. 103-112.

[5] Goel, A. L. and Soenjoto, J. (1981), "Models for hard ware-software system operational-performance evaluation", *IEEE Transactions on Reliability*, Vol. 30, No. 3, pp. 232-239.

[6] Hsieh, C. and Hsieh, Y. (2003), "Reliability and cost optimization in distributed computing systems", *Computers and Operations Research*, Vol. 30, No. 8, pp. 1103-1119.

[7] Huang, T .H., Chen, D .J. and Shend M. C. (1990), "An Algorithm to Generate FST's for the Reliability Analysis of Distributed Systems", *IEEE Region 10 Conference on Computer and Communication Systems*, pp. 150-154.

[8] Kshemkalyani, A. D. and Singhal, M. (2008), *Distributed computing: principles, algorithms, and systems*, Cambridge University Press, New York.

[9] Kumar A, and Agrawal, D. P. (1993), "A generalized algorithm for evaluating distributed-program reliability", IEEE Tra*nsactions on Reliability*, Vol. 42, No. 3, pp. 416-24.

[10] Lai, C. D., Xie, M., Poh, K. L. and Dai, Y. S. (2002), "A model for availability analysis of distributed software/hardware systems" *Inform Software Technology*, Vol. 44, No. 6, pp. 343-350.

[11] Lin, M .S., Chang, M. S., Chen, D. J. and Ku, K. L. (2001), "The distributed program reliability analysis on ring-type topologies", *Computers & Operations research*, Vol. 28, No. 7, pp. 625-635.

[12] Raghavendra, C. S., Kumar, V .K. P. and Hariri S. (1988), "Reliability Analysis in Distributed Systems", *IEEE Transaction on Computers*, Vol. 37, No. 3, pp. 352-358.

[13] Srinivasan, S. and Jha, N. K. (1999), "Safety and reliability driven task allocation in distributed systems", *IEEE Transaction on Parallel Distribution System*, Vol. 10, No. 3, pp. 238-251.

[14] Vidyarthi, D. P. and Tripathi, A. K. (2001), "Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm", *Journal of System Architecture*, Vol. 47, No. 6, pp. 549-554.

[15] Wang, J .L. (2004), "Markov-chain based reliability analysis for distributed systems", Computers and Electrical Engineering, Vol. 20, No. 3, pp. 183-205.

[16] Welke, S. R., Johnson, B. W. and Aylor, J. H. (1995), "Reliability modeling of hardware/ software systems", *IEEE Transaction on Reliability*, Vol. 44, No. 3, pp. 413-418.

2011년 8월 28일 접수, 2011년 9월 16일 수정, 2011년 9월 17일 채택