韓國 컴퓨터情報學會 論文誌
第9卷 第2號, 2004. 6.

2004-9-2-2-2

정보체계 운영 아웃소싱에 있어서의 서비스 수준 측정 메트릭

김 용 수*

SLA (Service Level Agreement) Metrics in IT Operation Outsourcing

Yong-Soo Kim*

요 약

정보체계를 아웃소싱 하는데 있어서 서비스 의뢰자와 제공자 모두가 이해하고 동의할 수 있는 합리적인 서비스수준측정 메트릭(SLM, Service Level Metric)이 있어야 하며 이를 기초로 서비스수준목표(SLO, Service Level Objective)가 세워지고 서비스수준협약(SLA, Service Level Agreement)이 작성 되어야 한다. SLM에 대한 연구가 필요한 이유로는 첫째, 아웃소싱 당사자들간의 불만족에도 불구하고 컴퓨터가 도입된 초창기의 메트릭들이 아직도 사용되고 있어서 메트릭자체에 대한 검토가 필요하다. 둘째, 시스템 구성요소의 가용성이나 사용율 등을 표현하는 SLM은 있으나 사용자(end-user)의 만족도를 표현하는 SLM이 없다. 셋째, 아웃소싱 계약이 대개 서비스 제공자가 주도하므로 SLO 및 SLM이 제공자에게 유리하게 작성되는 경우가 많아서 의뢰자도 인정할 수 있는 메트릭이 필요하다. 아웃소싱이 성공하기 위해서는 서비스 의뢰자와 제공자가 자신의 역할을 잘 인식해야 하고 양자간의 관계가 정립되어 있어야 하며 서로에게 기대하는 것이 무엇인지를 명확히 해야 한다. 또한 이러한 상호간의 기대치는 정량적인 수치인 메트릭으로 표현되어야 한다.

Abstract

For the successful IT operation outsourcing, there need appropriate metrics on which both of the service clients and provider agree. The metrics are used to set up the service level objectives, which are manifested in the service level agreement with price. A study of metrics is necessary for the following reasons: First, most of the metrics used today were introduced in the early years of computers and are not satisfactory to both of the service providers and clients. Second, metrics represent the performance of system components but not end-user satisfaction. Third, because the service provider leads the outsourcing agreement, the objectives are specified more favorable to the provider. The objectives should be based on metrics that both sides fully understand and agree on.

▸ Keyword : 아웃소싱, 서비스수준협약, 서비스수준 측정 메트릭, 서비스수준목표, 성능 메트릭, 성능 측정규준, 가용성, 응답시간, 신뢰성
outsourcing, SLA (Service Level Agreement), performance metric, SLM(Service Level Metrics), SLO(Service Level Objectives), availability, response time, reliability

* 경원대학교 소프트웨어대학 소프트웨어학부

Ⅰ. 서 론

일반적으로 아웃소싱은 서비스의 범위를 정하고 서비스 수준의 목표(SLO, Service Level Objective)를 정한 다음 서비스료를 산정하여 계약(SLA, Service Level Agreement)하는 절차를 거치는데 "정보체계 운영 아웃소싱"으로 서비스 범위를 정하면 다음 단계로 운영 서비스수준 목표를 결정해야 한다. SLO의 목표 값은 서비스수준측정 메트릭(SLM, Service Level Metrics)으로 표현되어야 한다. SLM이 명확히 정의되지 않으면 모호한 SLO가 설정되어 서비스 제공자와 의뢰자 사이의 분쟁이 생기기 쉽다.

IT 아웃소싱은 1970년대 후반기에 정보체계 운영 아웃소싱으로부터 시작하여1991년 IBM의 Kodak IT 아웃소싱이 아웃소싱 시장 확장의 하나의 큰 분수령이 되었다[1]. 오늘날에는 IT 아웃소싱의 필요성을 논의하는 단계를 훨씬 지나 양질의 서비스, 비용의 절감, 시장변화에 신속한 대응 등의 이유로 널리 도입되고 있으며 아웃소싱의 여러 분야에 대한 연구도 성숙해 있다.

그러나 정보체계 운영 아웃소싱에 대한 SLM의 개발 필요성에 주목해야 하는데 그 이유는 첫째, 아웃소싱 당사자들간의 불만족에도 불구하고 컴퓨터가 도입된 초창기의 메트릭들을 아직도 사용하고 있어서 메트릭 자체에대한 연구가 필요하다. 특히 우리나라는 대기업들이 IT 방계회사를 만든 다음, 그들로부터 기업 전체의 IT를 아웃소싱 하게함으로써 당사자들이 실정에 맞는 정교한 SLM의 필요성을 느끼지 못하고 있다. 둘째, 시스템 구성요소의 가용성(availability)이나 사용율(utilization) 등을 표현하는 SLM은 있으나 사용자(end user)의 만족도를 표현하는 SLM이 없다. 셋째, 아웃소싱 계약이 대개 서비스 제공자가 주도하므로 SLO 및 SLM이 공급자가 유리하게 작성되는 경우가 많아서 양자가 공정히 인정할 수 있는 메트릭이 필요하다[2].

정보체계를 운영하는 과정에서 서비스의 질을 추정할 수 있는 많은 정보를 얻을 수 있으므로 이들 정보로부터 아웃소싱 당사자들이 모두 이해하고 동의할 수 있는 합리적이고 축약된 SLM을 추출하여 이를 기초로 SLO를 세우고 SLA를 작성 해야 한다. 또한 아웃소싱이 성공하기 위해서는 서비스 의뢰자와 제공자가 자신의 역할을 잘 인식해야 하며 양자간의 관계가 정립되어 있어야 하며[3] 서로에게 기대하는 것이무엇인지를 명확히 해야 한다[4][5][6][7]. 또한 이러한 상호간의 기대치는 정량적인 수치로 나타낼 수 있는 척도가 있어야 한다.

SLA는 크게 성능(performance), 사후처리(reactive), 및 사전예방(proactive) SLA의 세가지 범주로 나눌 수 있다[8]. 성능 SLA는 사용자가 일상적으로 IT 서비스를 만족하게 받는 수준을 나타내며, 시스템의 가용성(availability), 응답시간(response time) 등이 여기에 속한다. 사후처리 SLA는 문제가 발생했을 때 서비스 제공자의 반응 속도와 서비스 질에 관한 것이며, 문제의 중요도(긴급, 중요, 보통 등)에 따라 차별을 두어 협약할 수 있다. 가용성과 응답시간은 성능 SLA의 범주에 들지만 시스템이 중지되거나 응답시간이 과도하게 느릴 경우 이를 바로 잡는 행위는 사후처리 SLA의 범주에 들어 간다. 사전예방 SLA는 문제를 미리 예방하는 조치를 표현하며, 시스템의 모니터링(monitoring), 데이터의 백업(backup), 보안감시, 패취(patch)와 업그레이드(upgrade)의 설치, 성능 조율(tuning), 용량계획(capacity planning) 등이 여기에 속한다. 본 논문에서는 Hawkins[9]가 제시한 SLA 구성요소 가운데 서비스 명세(service specification)를 중점으로 성능 SLA에 필요한 중요한 메트릭에 대해서 논의 할 것이며 기존 메트릭의 개선, 새로운 메트릭의 정의, 사용자 중심 메트릭의 중요성 등이 포함될 것이다. 메트릭 정의를 위해 필요한 기초 데이터는 시스템의 운영 중에 생산된 로그 파일(log file)이나 유틸리티(utility), 성능 모니터 등을 통해서 얻는데 본 논문에서는 기초 데이터의 추출 방법에 대해서는 취급하지 않았다.

Ⅱ. 기존 메트릭의 고찰

성능 메트릭은 속도(speed), 신뢰성(reliability), 가용성의 범주로 나눌 수 있는데 응답시간, 처리율(throughput), 사용율 등이 속도의 범주에 속한다. 오류발생률 또는 오류발생빈도는 신뢰성의 범주에, 평균가동시간(MTTF, Mean Time to Failure)은 가용성의 범주에 들어 간다. 성능 메트릭들을 정리하면 다음과 같다[10].

1. 응답시간

사용자가 가장 민감하게 반응하는 메트릭의 하나로서 시스템의 구성에 따라 세 가지 측정 방법(호스트, 중간노드, 클라이언트 노드)을 취할 수 있다.

2. 처리율

단위시간 당 일(task, transaction)을 처리하는 수를 말하며, 전체 시스템의 성능을 측정하는 메트릭으로 사용되지만 사용자가 피부로 느낄 수 있는 메트릭은 아니다. 흔히 처리율이 높아지면 시스템이 분주해져서 응답시간이 늘어나는 경향이 있다.

- 처리율 = (측정시간에 처리된 총 트랜잭션 수) / 측정시간 ····· (1)

프로세서의 처리율에는 MIPS (Million Instructions per Second) 또는MFLOPS(Millions of Floating-Point Operations per Second) 등이 사용되며 네트워크의 대역폭을 나타내는 데는 bps(bits per second) 등이 사용 된다.

3. 사용율

시스템 자원의 사용율을 말하며 자원사용시간 대비 전체 측정시간의 비율로 나타낸다. 역시 사용자의 관점에서 흥미있는 메트릭은 아니다.

- 사용율 = (측정시간 중 자원 사용시간) / 측정시간 ····· (2)

4. 오류발생율

프로그램의 오작동, 통신기기의 비트 에러 등이 여기에 속한다.

예를 들어, 통신기기에 대한 경우 해 일정기간을 모니터한 후 오류발생율을 다음과 같이 표현할 수 있다

- 오류발생율 = (비트 에러 발생 건수) / (송수신 패킷 수) ····· (3)

5. 가용율

MTTF는 고장이 날 때까지 시스템이 연속적으로 가동되는 평균시간을 나타낸다. MTTR(Mean Time to Repair)은 평균고장수리시간을 의미하며, MTTF와 MTTR을 이용하여 MTBF(Mean Time Between Failure)를 정의 하기도 한다.

- MTBF = MTTF + MTTR ····· (4)

그리고 (식4)를 이용하여 가용율(availability)을 다음과 같이 계산할 수 있다.

- 가용율(%) = (MTTF / (MTTF + MTTR)) × 100
  = (MTTF / MTBF) × 100 ····· (5)

상기의 메트릭 중 응답시간, 오류발생율 및 가용율은 사용자의 만족도에 직접 영향을 주는 메트릭이며 처리율, 사용율은 사용자의 만족도를 간접적으로 추측해 볼 수 있는 성능 메트릭이다.

III. 성능 메트릭

1. 가용성

1-1 최근가용율 및 최근가용시간

시스템은 정상작동과 고장을 반복함으로 가용율을 (식5)와 같이 표현하면 현재 작동 중인 시스템의 가장 최근의 가용율을 표현하는데 애매한 면이 있다. 시스템은 언제나 최초에 정상작동으로부터 시작함으로 (식 6)과 같이 최근 가용율을 표현할 수 있다.

- 최근가용율 = (last_Runtime + current_Runtime)/last_RepairTime ····· (6)

여기서 current_Runtime은 현재 작동하고 있는 시스템의 작동시간이며 last_RepairTime은 가장 최근에 고장이 났을 때의 복구 시간이며, last_RunTime은 가장 최근 고장 이전의 가장 최근의 작동시간 이다.

식(5)나 (6)을 사용하여 가용율을 구할 때 어느 정도까지 정확하게 계산할 것인지를 정해야 한다. 만약 소수점 셋째 자리까지 구하여 99.999%의 가용율을 얻었다면 이는 년 5.2분 정도 고장 상태에 있었던 시스템의 가용율이 될 것이다. 또, 소수점 두자리 까지 구하여 99.99%를 얻었다면 년 52분 정도 고장 상태에 있었던 시스템의 가용율이 될 것이다. 가용율이 여러 개의 9를 포함하는 숫자로 표현되면 가동율과 고장시간을 시간으로 인식하고 또 직관력적으로

비교하는데 불편하다. 고장시간은 분 단위로 표현하는 것이 이해하기 좋다. 가동시간은 고장시간에 비해 일반적으로 굉장히 크므로 이 두수를 같은 단위로 동시에 표현하는 경우에는 로가리즘(logarithm)을 이용하면 편리하다. 사용자는 최근의 시스템가용율에 관심이 크므로 다음과 같은 메트릭을 정의할 수 있다.

- 최근가용시간
  log(last_Runtime):log(last_RepairTime):log(current_Runtime) ····· (식 7)

예를 들어 3개월 동안 작동하다가 30분 고장을 일으킨 후 현재까지 10일간 문제없이 작동하고 있는 시스템의 최근가용시간은 "5.1:1.5:4.2"로 표현된다.

log(3×30×24×60):log(30):log(10×24×60)를 계산해 보면 대략 "5.1:1.5:4.2"를 얻을 수 있다. 정수 자리를 10 제곱해서 원래의 값을 추측할 수 있고 소수점 이하도 고려해서 이해할 수 있다. 이러한 표현은 가동시간과 고장시간을 사용하여 최근의 시스템 가용 현황을 간단히 표현할수 있을 뿐만 아니라 다른 최근가용시간과 비교하기에도 편리하다.

1-2 트랜잭션 가용율

현대의 정보체계 운영환경은 여러 종류의 서비스를 담당하는 다수의 서버, 각종 통신 장비를 포함하는 네트워크 환경, 사용자 환경 등 복잡한 양상을 띄우고 있어서 (식 5)-(식 7)에서 정의한 가용율이나 가용시간을 시스템의 각 구성요소에 적용하여 서비스수준목표를 정하면 복잡해져서 효용성이 떨어진다. 예를 들어 세 개의 DB 서버, 한 개의 OLTP(Online Transaction Processing) 서버 및 두개의 라우터(router)로 구성된 시스템에서는 여섯 개 시스템의 가용율을 산정해야 하며 이러한 가용율이 어떤 유기적인 관계를 가져서 사용자의 만족도에 영향을 주는지는 파악할 수 없다. 만약 모든 시스템이 선형적인 인과관계가 있다고 할 때만 가장 낮은 가용율을 전체 가용율로 대변할 수 있을 것이다. 이러한 복잡한 환경에서 전체 시스템의 가용율을 나타낼 수 있는 메트릭이 필요하며 본 논문에서는 온라인 트랜잭션을 기초로 전체 시스템의 가용성을 나타낼 수 있는 메트릭을 메인프레임(mainframe)과 C/S(client/server) 및 웹 환경에서 정의 한다.

(1) 메인프레임

IBM 메인프레임 환경에서 통합 시스템의 가용율을 트랜잭션에 기초하여 도출한 시스템이 있다[11]. 이 환경에서는 트랜잭션 처리를 위한 시스템의 구성요소는 (그림 1)과 같이 정의되어 있다.

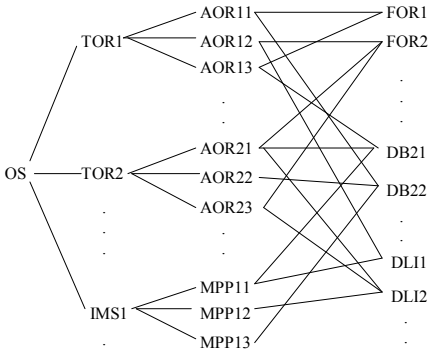

그림 1. CICS/IMS region과 DBMS와의 관계
Fig. 1 The Relationship between CICS/IMS Region and DBMS

IBM의 CICS(Customer Information Control System)는 OLTP의 일종으로서 TOR(Terminal Owning Region), AOR(Application Owning Region) 및 FOR(File Owning Region)로 구성되어 있으며 이들 세 개의 region은 하나의 주소공간(address space)에서 수행될 수도 있고, 한 서버 내의 서로 다른 주소공간에서 수행될 수 있으며, 각기 다른 서버에서 수행될 수도 있다. (그림 1)은 한 서버 내의 서로 다른 주소공간에서 수행되는 경우를 표현하고 있다. IMS(Information Management System)도 OLTP의 일종으로서IMS/CTL(IMS control region), DLI(Data Lange/1) 및 MPP(Message Processing Region)으로 이루어져 있다. 여기에서 DB2는 관계형 DBMS이다.

CICS 환경의 경우, TOR은 사용자로부터 트랜잭션을 받아서 AOR로 보내며, AOR은 그 트랜잭션에 해당하는 프로그램을 수행한다. 이 수행과정에서 DBMS나 FOR에 접근한다. 응답은 AOR로부터 TOR을 거쳐서 사용자에게 전송된다. IMS 환경의 경우, IMS/CTL이 전 시스템을 통괄하면서 사용자로부터 트랜잭션을 받아서 MPP로 보내고 MPP는 트랜잭션에 해당하는 프로그램을 수행한다. 그리고 수행과정에서 DBMS나 DLI을 접근하고 응답은 IMS/CTL.을 통해 사용자에게 전달된다. TOR(또는 IMS/CTL), AOR(또는 MPP), FOR(또는 DBMS)의 세 계층으로 나누어진 환경은 각 웹 환경에서의 웹서버, WAS(Web

Application Server) 그리고 DB 서버에 해당된다. IBM 메인프레임은 대단히 용량이 큰 서버임으로 (그림 1)과 같이 한 운영체제 내에 여러 구성요소가 동시에 운영되는 경우가 많다. (그림 1)과 같이 AOR11이 다운(down)되어도 AOR12와 AOR13를 통하는 트랜잭션은 수행된다. 그러나 TOR1이 다운되면 TOR1을 거쳐 처리되는 모든 트랜잭션(즉 AOR11, AOR12, AOR13 등)을 처리할 수 없게 된다. 또 DB21이 다운되면 AOR13 및 AOR21을 통해서 DB21을 접근할 수 없다. 정상적으로 운영되는 시스템에서 각 AOR을 통해 수행되는 트랜잭션의 수를 시간대별로 수집할 수 있다. 만약 어떤 AOR이 어느 시간대에 얼마동안 다운된다면 다운되어 있는 동안 앞서 수집해 놓은 트랜잭션 수만큼 수행하지 못하는 것으로 추정할 수 있다.

물론 내부사용자의 경우 꼭 처리해야 할 일이라면 AOR이 정상 작동된 후 다시 처리할 수 있을 것이나 온라인 쇼핑몰의 고객인 경우는고객을 잃을 수 있다. 여기서 다운된시간동안 수행치 못한 추정 트랜잭션의 수를 LST(Lost Services of Transactions)로 정의하고 AOR을 기본시스템으로 하여 AOR과 TOR, FOR, DBMS 등과의(그림 1)과 같은 관계를 고려하여 (식 8) -(식 11)을 도출할 수 있다. LST는 소실된 트랜잭션 또는 지연된 트랜잭션의 수를 나타내고, 시스템의 어떤 구성요소가 다운되더라도 하나의 메트릭으로 표현됨으로 시스템 전체를 대표하는 메트릭이라고 할 수 있다. 또한 평일, 휴일, 가장 바쁜 시간(peak time), 밤, 낮, 여타 시간 등에 따라 별도의 시스템 가동율의 목표 값을 정할 필요가 없다. 왜냐하면 LST는 이러한 시간대별 고려가 메트릭 안에 포함되어 있기 때문이다.

- $TOR_i = \sum_k AOR_k \_\_ TOR_i$ ····· (8)
- $FOR_i = \sum_k AOR_k \_\_ FOR_i$ ····· (9)
- $DB2_i = \sum_k AOR_k \_\_ DB2$ ····· (10)
- $DLI_i = \sum_k AOR_k \_\_ DLI_I$ ····· (11)

여기서 ($\sum AOR_k \_\_ TOR_i$)는 TOR_i에 연결된 모든 AOR의 집합이며 표기의 편의상 수식의 좌우변은 트랜잭션의 수를 나타낸다고 하자.

마찬가지로 ($\sum AOR_k \_\_ FOR_i$)는 FOR_i에 연결된 모든 AOR의 집합이며, ($\sum AOR_k \_\_ DB2_i$)는 $DB2_i$에 연결된 모든 AOR의 집합이며, ($\sum AOR_k \_\_ DLI_i$)는 DLI_i에 연결된 모든 AOR의 집합이다.

IMS의 경우도 MPP를 기본시스템으로 하여 MPP를 AOR과 같이, IMS/CTL을TOR과 같이 취급하면 (식 12) - (식 14)를 얻을 수 있다.

- $IMS_i = \sum_k MPP_k \_\_ IMS_i$ ····· (12)
- $DB2_i = \sum_k MPP_k \_\_ DB2_i$ ····· (13)
- $DLI_i = \sum_k MPP_k \_\_ DLI_i$ ····· (14)

Master/Slave 형태의 SNA(System Network Architecture) 네트워크 환경도 (그림 2)와 같이 표현할 수 있다.

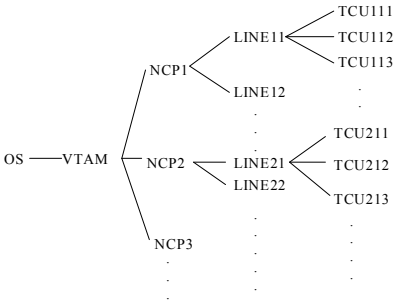

그림 2. SNA 네트워크의 구조
Fig. 2 SNA Network Structure

VTAM(Virtual Telecommunications Access Method)은 서버에서 네트워크 프로토콜을 관장하며 실제 하부 통신을 관장하는 CCU(Communication Control Unit)와 연결되어 있다. NCP(Network Control Program)는 CCU의 운영체제이며, CCU는 하부 라인과, 라인에는 사용자 터미널을 관장하는 TCU(Terminal Control Unit)가 연결되어 있다. 네트워크환경에서도 TCU를 기본시스템으로 하여 LST를 (식 15)-(식 18)과 같이 정리할 수 있다.

- $LINE_i = \sum_m TCU_m \_\_ LINE_i$ ····· (15)
- $NCP_k = \sum_i LINE_i \_\_ NCP_k$ ····· (16)
- $CCU_k = NCP_k$ ····· (17)
- $VTAM = \sum_k CCU_k + \sum_i local \_\_ LINE_i$ ···· (18)

여기서 ($\sum$ TCU_m — $LINE_i$)는 $LINE_i$에 연결된 모든 TCU의 집합이며, ($\sum$ $LINE_I$ — NCP_k)는 NCP_k에 연결된 모든 LINE의 집합이다. (식 17)은 CCU의 다운은 NCP의다운과 같음을 나타내고 (식 18)은 VTAM의 다운은 모든 CCU의 다운과 CCU를 통하지 않고 연결된 모든 local LINE이 다운된 것과 같음을 나타낸다.

온라인 시스템의 경우 운영체제, 호스트(H/W), VTAM의 다운이 전체 시스템의 다운과 같고 (식 19)와 같이 나타낼 수 있다.

- $OS = Host = VTAM = \sum TOR_i + \sum IMS_j$ ..... (19)

LST는 시스템의 구성요소가 다운되어 처리하지 못하거나 지연 처리되는 트랜잭션의 수이다. 시스템이 정상작동 할 때의 일정한 기간에 처리되는 전체 트랜잭션의 수(이를 TVOL이라 한다)를 알고 그 기간 중 장애가 난 시간대의 LST를 알면 트랜잭션 가용율(TA, Transaction Availability)을 (식 20)과 같이 구할 수 있다.

- $TA = (TVOL - LST)/TVOL)$ ..... (20)

또 TVOL을 LST에 나누어서 (식 21)과 같이 LSTI(LST Interval)을 구할 수 있는데, 이는 LSTI 분량 만큼의 트랜잭션을 수행하다가 1개의 트랜잭션의 지연 요인이 생겼음을 나타낸다.

- $LSTI = TVOL/LST$, if $LST > 0$
- $LST = TVOL$, If $LST = 0$

..... (21)

TVOL과 LST의 크기에 따라 LSTI 크기의 차이가 심해서 LSTI의 값으로 가용율을 비교하기가 어렵다. LSTI에 로가리즘을 취하면 (식 22)와 같은 LSTIA (LSTI Adjusted)를 얻는다.

- $LSTIA = \log(LSTI)$ ..... (22)

LSTIA는 전체 시스템의 가용성을 하나의 수로 나타낸다. 만약 지난 한달 동안 1억2천만 건의 트랜잭션을 처리하는 시스템에서 구성요소의 일부분에 장애가 있어서 5천6백건의 LST가 발생했다고 하면, 트랜잭션 가용율은 (1억2천 5천6백)/1억2천 = 0.999953(즉 99.9953%)이고, LSTI는 1억2천만/5천6백 = 21,429로서 21,429건 처리할 때마다 한 건의 지연이나 손실이 있었다고 해석할 수 있다. 그러나 여러 개의 TA나 LSTI를 비교할 경우 시스템의 안정도에 따라 값의 차이가 커서 비교하기 어렵다. 그러나 이를 LSTIA로 조정했을 경우 log(21429) = 4.3의 간단한 메트릭으로 나타낼 수 있고 정수 부분을 보고 1만(10^4)의 숫자대임을 알 수 있다.

(2) C/S와 웹환경

C/S는 2 계층으로 구성될 경우 대개 클라이언트와 DB 서버로 구성되어 있고 3 계층으로 구성될 경우 클라이언트, OLTP 및 DB 서버로 구성된다. 웹 환경도 클라이언트, 웹 서버 및 DB 서버로 구성될 경우 3 계층 C/S와 유사한 구성을 가지나 WAS 서버가 웹 서버와 DB 서버의 중간에 위치할 경우 4 계층이 된다. 시스템 환경이 CORBA(Common Object Request Broker Architecture)가 지향하는 컴포넌트(component)에 기초한 객체지향 모델로 전환되면 다차원의 계층을 이루는 시스템이 가능해 진다. 본 논문에서는 2-3 계층 C/S 및 웹 환경을 중심으로 LSTIA를 추출하는 방법을 논의 하겠지만다 계층에서도 응용가능하다.

① 2 계층 C/S 환경

2 계층 C/S는 (그림 3)과 같이 클라이언트가 여러 개의 DB 서버를 접근할 수 있게 되어 있으나 수정 락(lock) 등이 수행될 경우 심각한 성능 저하를 초래할 수 있으므로 대개 하나의 DB 서버만 있는 작은 시스템 환경을 이루고 대부분 LAN(Local Area Network)에서 운영되고 있다. 또한 성능과 유지보수의 편의성을 위해 DB 접근 시 stored procedure를 이용해서 DB 서버를 한 번 접근해서 하나의 트랜잭션을 이루는 모양을 취한다. DBMS로부터 stored procedure 및 개별 SQL을 수행한 통계를 얻어서 트랜잭션 수를 추정할 수 있고 기본시스템은 DBMS가 된다.

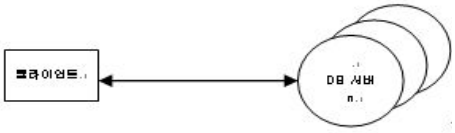

그림 3. 2 계층 C/S 구성도
Fig. 3 Two Tier C/S Structure

② 3 계층 C/S 환경

3 계층 C/S 환경은 (그림 1)과 유사하며 (그림 4)와 같이 나타낼 수 있다. (그림 4)에서 OLTP는 대부분 (그림 1)에서의 TOR과 AOR의 역할을 겸하고 있다. OLTP는 트랜잭션 처리의 중심에 있으므로 수행한 트랜잭션의 자료는 OLTP로부터 얻어야 하며 각 DB 서버별로 분류되어야 한다. OLTP내에 응용프로그램을 집단적으로 모아 놓은 서버의 개념이 있지만간단히 하기위해 기본시스템을 DBMS로 한다

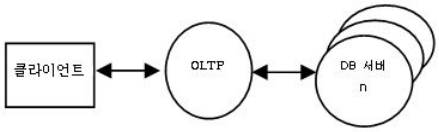

그림 4. 3 계층 C/S 구성도
Fig. 4 Three Tier C/S Structure

OLTP와 DB 서버와 관계는 (식23)과 같이 나타낼 수 있다.

- $OLTP = \sum_{i} DB_i$ ..... (23)

OLTP가 다운되면 사용자의 입장에서는 모든 DB)가 다운된 것과 같다. C/S 환경에서 널리 사용되는 OLTP로는 Tuxedo, Top End, CICS/6000 등이 있다.

③ 웹 환경

웹 환경은 구성상 OLTP가 웹서버로 대체 되든지 또는 웹서버와WAS 서버로 대체되는 (그림 5)와 같은 모양을 하고 있다. 또 메인프레임과 비교하면,웹서버는 TOR, WAS는 AOR과 유사한 역할을 한다. 트랜잭션은 웹서버, WAS 서버 등에서 수집되어야 하고 WAS의 트랜잭션이 각 DB 서버별로 분류되어야 한다. 기본시스템은 WAS이며 WAS가 없는 환경에서는 DBMS가 된다.

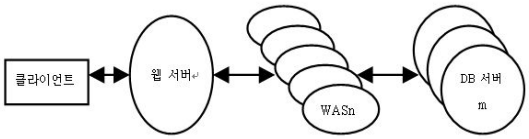

그림 5. 웹 환경
Fig. 5 Web Environment

한 웹서버로 가정하면 기본시스템을 중심으로 구성요소간의 관계는 (식 24), (식 25)와 같이 나타낼 수 있다.

- $WEB = \sum_{i} WAS_i$ ..... (24)
- $DB_I = \sum_{k} WAS_k \_\_ DB_i$ ..... (25)

④ TCP/IP 네트워크

LAN과 WAN(Wide Area Network) 또는 인터넷 등 모든 네트워크 환경은 클라이언트의 요청을 최초로 받는 웹서버나OLTP를 중심으로 라우터를 통해 연결된 (그림 6)과 같은 구조를 가지고 있다. 네트워크가 여러 경로를 통해 연결되어 있는 경우 한 라우터가 작동하지 않더라고 다른 라우터를 통해 전송경로가 형성되므로 모든 경로가 차단된 경우를 제외하고는 정확한 LST를 산출하기 어렵다. 그러나 어떤 라우터가 작동하지 않은 경우와 작동하는 경우의 전체 트랜잭션 처리수를 비교하여 특정라우터의 LST를 추출할 수 있다. 웹서버나 OLTP에 TCP/IP 스니퍼(sniffer)나 HTTP 스니퍼 등을 설치하여 서브네트워크(sub-network)별로 트랜잭션 수를 구할 수 있다. 사용자별로 트랜잭션 수도 구할 수 있겠지만 DHCP(Dynamic Host Configuration Protocol)을 사용할 경우 클라이언트를 부팅(booting)할 때마다 주소가 바뀌고, 사설 IP 주소를 사용할 경우 이 주소는 서버에 전달되지 못하므로 실효성이 없다. 인터넷의 경우 ISP(Internet Service Provider)와 연결된 라우터 바깥의 환경에 대해서는 LST를 산정할 수 없다. TCP/IP 환경에서의 기본시스템은 서브네트워크이다.

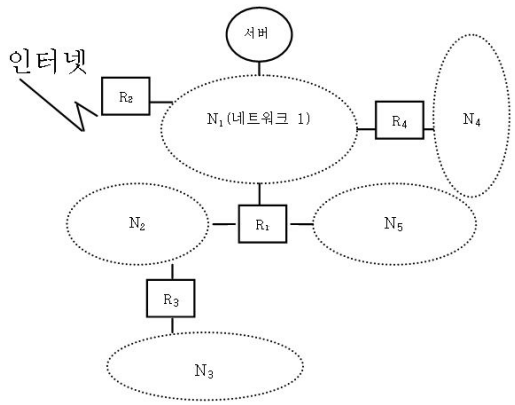

그림 6. 네트워크 환경의 예
Fig. 6 Example of a Network Environment

(그림 6)과 같은 네 개의 라우터(R_1 - R_4)와 다섯 개의 네트워크(N_1 - N_5) 및 인터넷을 접근할 수 있는 환경을 가정하고 모든 서버가 N_1에 존재한다고 하면, N_3에 있는 사용자는 R_3에 문제가 생기면 서버를 접근할 수 없고 N_4에 있는 사용자는 R_4에 문제가 생기면 서버를 접근할 수 없다. R_2가 고장이 나면 인터넷을 통해 서버를 접근할 수 없다. 만약 R_1이 고장나면 N_2, N_3 및 N_5에 있는 모든 사용자가 서버를 사용할 수 없게 된다. (그림 6)에서 라우터 R1이 다운될 때의 LST를 R_1이라고 하면 그 LST는 서브네트워크 N_2, N_3, N_5의 LST 모두 더한 것이 된다. 일반적으로 라우터 R_i가 다운되면 R_i를 통해 서버를 접근하는 모든 서브네트워크의 LST를 모두 더해 R_i의 LST를 계산한다.

- $R_i = \sum_k N_k$ ····· (26)

2. 응답시간(Response Time)

응답시간은 네트워크 상의중간 노드 또는 호스트에서 측정할 수 있으나 사용자의 관점에서 볼 때 가장 정확한 시간은 클라이언트에서 측정하는 것이다. DBMS도 로그(log) 또는 온라인 성능 모니터를통해 여러 가지 성능 데이터를 보여주지만 더욱 정확한 것은 요청자의 관점에서 모니터한 값이다[12]. 응답시간은 LAN 환경에서는 1초 이하(sub-second), WAN 환경에서는 3초 이내가 적절하다고 여겨지고 있다. 웹환경에서 사용자가 참을수 있는 최대 응답시간은 11초라는 연구 결과도 나와 있다[13].

2-1 호스트에서 측정

메인프레임 환경에서는 전통적으로 클라이언트는 더미 터미널(dummy terminal)이었으므로 응답시간을 모니터할 수 있는 로직을 가지고 있지 않았다. PC가 널리 보급되어 PC 운영체제 하에서 터미널 이뮬레이터(emulator)를 구동할 때에도 응답시간을 모니터 하는 기능들을 넣지는 않았다. 대신 (그림 7)의 방법으로 호스트에서 사용자 응답시간(end-to-end response time)을 추정하였다.

그림 7. 호스트에서의 사용자 응답시간 측정
Fig. 7 End-to-End Response Time Measurement at Host

응답시간 추정 순서는 다음과 같다.

(1) 호스트(VTAM)는 대상 터미널의 모드를 DR (definite response mode)로 한다. DR 모드에서는 호스트로부터의 패킷 마다 각각 대응하는 ACK(DR)를 보낸다.

(2) 사용자의 요청이 사용자 터미널로부터 호스트에 도착한다①. 패킷의 크기 P를 보관한 후 해당 응용프로그램으로 전달하여 수행케 한다.

(3) 응용프로그램으로부터의 결과를 사용자에게 보낸다②. 이때 호스트에서 걸린 시간 A 및 현재 시간 T1를 보관한다.

(4) 사용자 터미널은 응답에 대한 ACK를 보낸다. 호스트는 ACK를 받은 시간 T2를 구하고 ACK의 길이 L과 요청 패킷의 길이 P의 크기 차이를 비교하여 ①의 시간을 ③의 시간으로 대처함으로 생기는 오차 α를 보정한다. 이때 송수신 라인의 대역폭이나 통신 환경이 같다는 것을 전제하고 있다.

- 사용자 응답시간 = $(T_2 - T_1) + A + \alpha$ ····· (27)

요청 패킷마다 DR을 요구하면 트래픽(traffic)과 DR 처리부하를 증가 시킨다. 사용자 터미널에서 $(t_2 - t_1)$ 시간을 측정하는 것이 사용자 관점에서 가장 정확한 응답시간이 될

것이다.

2-2 중간 네트워크 노드에서 측정

TCP/IP 환경에서 중간 네트워크 노드에서 응답시간을 측정하는 방법이 시도되고 있다. Packeteer Inc.[14]는 네트워크 QOS(Quality of Service) 장비를 판매하는 회사로서 자사의 장비로 사용자 응답시간을 구하고 이를 네트워크에서 걸린 시간과 서버에서 걸린 시간으로분리할 수 있는 알고리즘을 개발하였다. SLA에 사용할 메트릭은 사용자 관점의 응답시간이므로 이를 간략히 (그림 8)과 같이 나타낼 수 있다. t_1과 t_2 사이에 여러 개의 중간 요청 패킷이 있을 수 있고 역시 T_3과 T_4 사이 다수의 중간 응답 패킷이 있을 수 있으나 그림을 간략히 하기 위해 생략하였다. PSH는 마지막 패킷을 나타내는 TCP의 push 플래그이다.

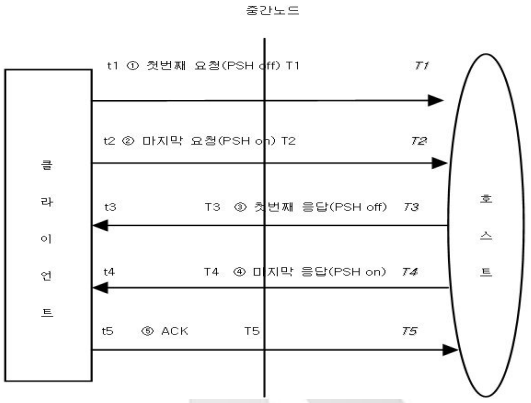

그림 8. 중간노드에서의 응답시간 측정
Fig. 8 Response Time Measurement at an Intermediate Node

클라이언트의 관점에서 가장정확한 응답시간은 $(t_4 - t_1)$ 이다. 중간노드에서 이에 가장 유사한 시간은 $(T_5 - T_1)$ 이다. 중간노드에서는 $(T_1 - t_1)$ 시간을 $(T_5 - t_5)$ 시간으로 대체 했다. 중간노드는 첫번째 요청 패킷의 길이와 ACK의 길이 알고 있으므로 이를 보정(β)하여 (식 28)을 얻는다.

- 사용자 응답시간 = $(T_5 - T_1) + \beta$ ····· (28)

2-3 사용자(클라이언트)에서 측정

응답시간을 호스트나 중간노드에서 측정하면 전체 시스템의 현황을 쉽게 파악할 수 있는 장점이 있는 반면 측정으로 인하 부하가 전체 시스템에 미치며 가끔 사용자가 동의할 수 없는 결과를 보여주기도 한다. 클라이언트에서 응답시간을 수집하면 수집된 데이터를 통합하는 번거러운 작업을 거쳐야 하지만측정 부하를 클라이언트가 흡수하여 호스트나 중간노드에 부하를 주지 않고 언제나 사용자가 동의하는 결과를 얻는다는 장점이 있다. 2 계층 C/S 시스템에서는 클라이언트 프로그램에 응답시간을 측정하는 기능을 넣을 수 있고(instrumentation) 특정 클라이언트를 특정 시간대에만 측정하게 할 수도 있을 것이다. 웹을 사용할 경우 브라우져가 응답시간에 대한 정보를 추출할 수 있는 가장 좋은 곳이나 아직 이를 지원하는 것은 없다. 메인프레임의 경우 이뮬에이터 프로그램에 응답시간 측정 기능을 넣을 수 있다. TCP/IP를 사용하는 C/S나 웹 환경에서는 TCP/IP 스니퍼(sniffer) 또는 HTML 스니퍼 들이 많이 나와 있다. 이러한 스니퍼를 클라이언트에서 이용하면 사용자 관점에서 정확한 사용자 응답시간을 구할 수 있다. 전통적으로 응답시간은 사용자가 Enter 키 또는 전송 버튼을 통해 요청을 한 시간부터 응답의 첫 부분이 도달할 때 까지로 정의하고 있다. 그러나 웹환경에서는 하나의 사용자 요청에 대해 여러 개의 응답이 전달되는 경우가 많고, 광고나 공지사항 배너(banner) 등을 고려하면 첫 부분의 도달을 기준으로 삼을 때 사용자가 동의하지 않을 수 있다. 그러므로 트랜잭션의 성격을 파악해서 핵심 정보를 담고 있는 응답이 도달하는 시점을 기준점으로 잡아 SLO에 포함시켜야 한다.

Ⅳ. 응용

SLA 메트릭들은 대부분 수준에 따라 등급을 매겨서 보상과 벌금을 부과할 수 있다. 예를 들어 응답시간의 95%가 1.5초 이내 이면 B 등급을 부여하고 기본약정금의 5%를 더 지급하고 2초 이내 이면 C 등급을 부여하고 기본 약정금만 지급하는 협약을 만들 수 있다. 가용성의 경우, 각 시스템의 구성요소별로 가용율을 정해야 하므로 너무 복잡해져서 실용성이 없고엉뚱한 결과를 낳기 쉽다.

그러나 LST는 시스템의 구성요소 중 어떤 부분에 장애가 발생하드라도 이를 지연 또는 소실 트랜잭션의 수라는 단일한 메트릭으로 표현하고 또 이 가용성을 이용하여 기회손실비용을 계산할 수 있다. 특히 인터넷 쇼핑몰의 경우 총 매출액(R)을 총 트랜잭션 수(T)로 나누어 트랜잭션 당 매출액을 산정해 놓고 이 값을 LST에 곱하여 손실을 계산할 수 있다.

• 손실 = LST × (R / T) ····· (29)

응답시간은 사용자의 만족도와 관련된 가장 중요한 메트릭이지만 이를 비용으로 환산할 경우 경험적이고 모호한 부분이많으므로 (식 29)와 같이 메트릭으로부터 직접 비용을 산정할 수 있는 메트릭들이 더 많이 개발되어야 한다. 또 현재 사용중인 시스템 구성요소에 대한 SLM들을 시스템 전체를 평가할 수 있는 (식 30)과 같은 메트릭으로 변환하는 연구도 기대된다.

• $SLM_T = (SLM_i \times C_i)$ ····· (30)

여기서 SLM_i는 구성요소들의 각종 SLM들이고 C_i는 해당 조정계수 이며 SLMT는 전체 시스템의서비스수준측정 메트릭이다.

V. 결론

사용자의 관점에서 서비스수준목표를 정하는데 가장 중요한 두 가지 메트릭인,응답시간과 가용성에 대해 논의하였다. 가용성은 MTTF와 MTTR을 사용해서 계산하여 왔으나 최근가용율과 최근가용시간이라는 새로운 메트릭을 제시하여 현재 가동중인 시스템의 가용율에 대한 정보를 쉽게 파악하고 비교할 수 있게 했다.

특히 온라인 서비스의 소실이나 지연을 나타내는 LST 메트릭을 개발하여 전체 시스템을 나타내는 메트릭이 될 수 있음을 보였다.

실제 LST는 어떤 아웃소싱회사의 데이터센터에서 온라인 서비스의 수준을 나타내는 지표로 사용되고 있다.

응답시간은 사용자 측에서 관측하여야 사용자가 수긍할 수 있는 메트릭이 될 수 있고 웹환경에서는 응답시간의 정의가 수정되어야 함을 보였다. 시스템 환경이 점점 복잡해지고 있으므로 고려해야 할 메트릭과 서비스수준목표가 많아지겠지만 향후 메트릭을 통합하여 간단히 해 가는 연구와 더불어 메트릭을 이용해서 직접적으로 비용을 산정하는 방법들도 연구되어야 할 것이다.

참고문헌

[1] Applegate, L.M., and Montealegre, R. "Eastman Kodak Co.: Managing Information Systems Through Strategic Alliances," 192-030, Harvard Business School, Boston, MA.

[2] Jin Martin, Arne Nisson "On Service Level Agreements for IP Networks", IEEE, 2002

[3] Levina, N., and Ross, J.W. "From the Vendor's Perspective: Exploring the Value Proposition in IT Outsourcing" MIS Quarterly (27:3), pp 331-364, September 2003

[4] Sabherwal, R. "The Role of Trust in Outsourced IS Development Projects", Communications of the ACM (42:2), pp 80-86, February 1999

[5] Willcocks, L., and Lacity, M.C. "Relationships in IT Outsourcing: A takholder Perspective", in: Framing the Domains of IT Management, R. Zmud (ed.), Pinnaflex Inc., pp 355-384, Ohio, 2000

[6] Goles, T. "The Impact of Client-Vendor Relationship on Outsourcing Success," University of Houston, Houston, TX, 2001

[7] Elitzur, R., and Wensley, A. "Game theory as a tool for understanding information services outsourcing," Journal of Information Technology (12), pp 45-60, 1997

[8] Piotr SZYMCZYK "Developing Service Level Agreement for Outsourced Processes and Systems", International Carpathian Control Conference ICCC' 2002, Malenovice, Czech Republic, May 27-30, 2002

[9] Hawkins S. "Service-Level Agreement for Outsourcing, Strategic Analysis Report", Gartner Group 1997

[10] Raj Jain "The Art of Computer Systems Performance Analysis", pp 33-40, John Wiley & Sons, April 1991
[11] Yong-Soo Kim "AVAILABILITY METRICS OF AN ONLINE SYSTEM", Computer Measurement Group, Rino 1999
[12] 김용수, 이금석 "사용자 주소공간에서 DBMS 응답시간의 온라인 모니터", 한국정보과학회 '93 가을 학술발표논문집, pp 218-221, 1993
[13] N. Bhatti, et. Al. "Integrating User-Perceived Quality into Web Server Design", Ninth International WWW Conference, May 2000
[14] Packeteer Inc. "How to Measure the Impact of Application Performance On Your Business", PACKETEER TECHNICAL FOCUS PAPER

저 자 소 개

김 용 수
경원대학교 소프트웨어대학
소프트웨어학부
<관심분야> 운영체제, 성능분석/관리