결함 자가 관리를 위한 오토노믹 시스템 기술 동향

김용호 | 재료연구소

1. 서 론

복잡해진 IT기반 시스템 운영 관리 자동화 실현을 향한 오토노믹 시스템(Autonomic System)에 대한 연구가 활발히 진행되고 있다. 오토노믹 시스템의 어원은 인간의 자율신경계(Autonomous Nervous System)로부터 파생한 용어로서, 기온이 떨어졌을 경우 일정 체온을 유지하기 위해 몸을 떨어 체온을 조절하거나, 어두운 곳에 들어갔을 경우 동공을 팽창시켜 시야를 확보하려는 반응이나, 격한 운동 후 심장 박동수 또는 혈액 순환 조절을 통해 몸을 안정화시키는 인간의 자율 신경계 작용 원리를 컴퓨팅 시스템 관리에 적용하자는 아이디어에서 기원한 용어이다. 다시 말해서, 두뇌에서 의사 결정을 위한 분석을 수행하고 그에 적절한 주위 환경 변화를 인지하고 이에 대응할 수 있도록 시스템 관리 기술을 자동화하자는 것이 오토노믹 시스템의 기본 개념이다. 따라서 오토노믹 시스템에 대한 정의는 서비스 관리를 위한 일련의 과정에서 사용자나 개발자와 같은 사람의 간섭을 최소화하여 최종적으로는 시스템 스스로 자신을 관리할 수 있는 능력을 보유한 시스템이라 정의할 수 있다. 이를 통해 얻고자 하는 목표는 시스템 관리에 소요되는 비용 즉 전체 시스템 유지보수비용을 최소화하여 관리에 필요한 사람의 노력을 본질적인 서비스 제공에 충실하게 함으로써 서비스의 질을 향상시키고 투자가치를 높이자는 것이다.

오토노믹 시스템은 자가 구성(Self-configuring), 자가 치유(Self-healing), 자가 보호(Self-protecting), 자가 최적화(Self-optimizing)를 위해 설계될 수 있으므로 적용하고자 하는 시스템에 신뢰성을 제공하기 위한 최적의 솔루션이다. 다시 말해서 기존의 시스템 운영 환경 정보를 모니터링하여 예기치 못한 장치의 동작을 처리할 수 있는 자동적 적응성을 제공하도록 구성 가능하며, 사용자 및 관리자의 개입을 최소화할 수 있고, 이에 대해 기계가 스스로 학습할 수 있는 능력이 제공되도록 실현시킬 수 있는 방안이다. 최근 오토노믹 시스템에 대한 연구는 대학, 연구소, 기업에서 활발히 진행되고 있는데, 대표적인 기업의 예를 정리하면 표 1 과 같다.

하지만 오토노믹 시스템에 대한 명확한 정의나 구조가 정립되지 못한 상태이며 이를 구현하기 위한 핵심기술 또한 개발이 활발히 진행되고 있을 뿐 아직 해결하지 못한 문제가 많아 실제 상용 시스템에 적용하기가 어렵다. 오토노믹 시스템을 만들기 위해서는 결함과 같은 문제를 인지해 원인을 진단하고 적절한 처리 방안이 있어야 하는데, 기존 시스템 환경과 관리 기법이 명확하게 절차적으로 정리되어 있어야 하며, 관리가 기계적으로 처리 가능한 범위로 한정되기 때문이다.

본 고에서는 오토노믹 시스템의 실현을 위해 문제가 되는 결함의 특성과 결함 자가 관리 메커니즘에 대해 소개

표 1. 오토노믹 시스템 개념과 유사한 기업별 용어 및 정의

기 업	용 어	정 의	핵심구조
IBM[1]	Autonomic Computing	비즈니스 정책과 추구하는 목표를 달성하기 위해 스스로 관리하고 동적으로 적응할 수 있는 시스템	Monitor, Analyze, Plan, Execute
MS[2]	Dynamic System	시스템 관리에 필요한 오퍼레이션을 자동화함으로써 보다 관리가 용이하도록 설계된 시스템	Detect, Classify, Diagnose, Recovery, Verify
HP[3]	Adaptive Enterprise	관리 기술을 간편화, 표준화, 통합화하여 개발된 시스템	Assess, Advise, Act
Intel[4]	Cross-platform Manageability	이질적인 플랫폼을 통합관리할 수 있는 인터페이스 및 프로토콜을 개발하여 이를 활용한 시스템	Discover, Heal, Protect

하고자 한다. 오토노믹 시스템은 서비스에 대한 자율성도 중요하지만 시스템의 고가용성과 고신뢰성을 위해 결함 발생 시에 스스로 이를 인지하고 처리하는 것이 매우 중요하다. 소개하는 내용은 고성능 연구 장비나 로봇과 같은 디바이스가 운영 중에 스스로 문제를 찾고 해결하여 지속적이고 신뢰성 있는 서비스를 제공할 수 있도록 하는 등 오토노믹 시스템을 구성하기 위한 기본이 된다.

2. 결함

결함은 시스템이나 운영환경에서 나타나는 이상 증상으로써, 여러 가지 유형이 있으며 시스템이 자율적으로 처리 가능한 것과 가능하지 못한 것이 있다. 결함에 대해서는 시스템마다 해결 방법을 모색하고 있는데, 오토노믹 시스템 기술이 연구되면서 결함 관리 측면에서 결함에 대한 연구가 체계화되고 있다.

그림 1은 UKSMA에서 결함의 생명 이력을 오토마타 기법을 이용하여 일반적으로 정리한 것이다[5]. 그림 1에서 보는 바와 같이 결함은 다양한 형태로 존재하며, 사용자나 사람이 직접 개입해야 처리 가능한 경우와 자체적으로 처리 가능한 수준으로 구분할 수 있다.

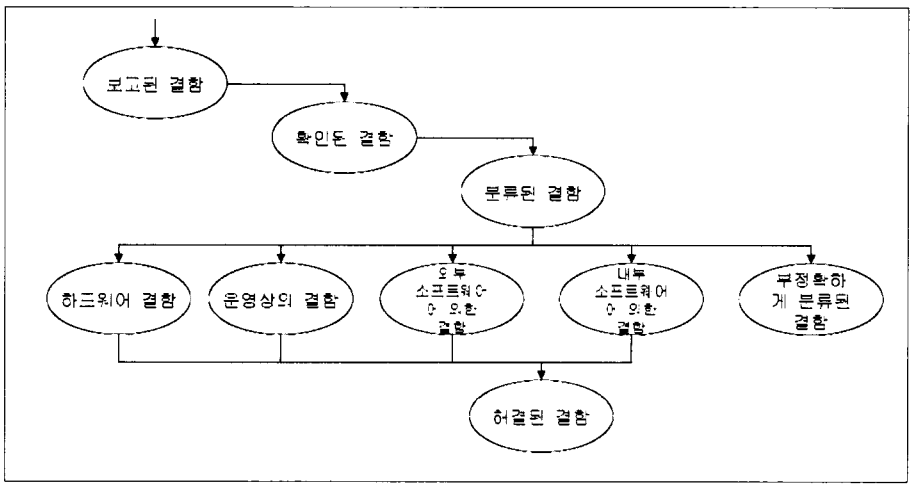

그림 1. 결함의 생명 이력 모델

그림 1의 결함 해결 절차는 결함이 보고되고, 결함의 유형을 파악함으로써 결함을 확인하고, 결함의 원인을 분류하여 해결 방법을 모색하여 해결한다. 결함 해결 절차는 사람이 개입되거나 시스템 스스로 처리하든 동일하게 적용된다. 시스템 스스로 해결하기 위해서는 발생 가능한 결함의 유형이 정립되어 데이터베이스화되어 있어야 하고 기계적으로 제어 처리 가능한 범위 내에 있어야 한다.

그림 2는 결함의 심각성에 따라 결함을 분류한 것이다. 결함이 발생한 장치나 소프트웨어가 속한 시스템의 운영이나 성능에 끼치는 수준과 정도의 범위를 분류한 것이다. 그림 3은 결함이 해결되어야 할 긴급성을 나타낸다. 그림 2와 3은 결함 자체에 대해 일반적으로 분류하는 기준으로써 업무에 적용하거나 시스템 실현 시에 활용할 수 있다.

#	수준	설명
A	Critical	시스템의 운영을 불가능하게 하는 매우 심각한 결함이다. 일부 스함이 불가능해지며, 수작업에 의한 업무를 수행하여야 한다.
B	Major	부분적으로 시스템의 운영 성능을 심각하게 저하시키는 결함으로, 일부 시스템의 고장으로 인해 일부 업무를 처리할 수 없게 만든다.
C	Minor	사용자를 불편하게 만들지만, 업무를 수행할 수 있는 수준으로 시스템을 운영할 수 있도록 하는 결함이다.
D	Cosmetic	데이터의 형식 또는 표현 방식에 관련된 결함으로 데이터의 입력 또는 참조된 데이터를 발생시키지는 않는다.

그림 2. 결함의 심각성

#	수준	설명
1	Very Urgent	즉시 해결되어야 하는 결함이다.
2	Urgent	통상적으로 다음 운영 주기까지 신속히 해결되어야 하는 결함이다. 예를 들면, 야간 또는 주말까지 해결되어야 하는 경우이다.
3	Routine	지정된 릴리즈 일자까지 해결되도록 일정이 조정될 수 있는 결함이다.
4	Not Urgent	필요 시 해결되어도 되는 결함이다.

그림 3. 결함의 긴급성

시스템의 신뢰성을 높이기 위한 결함 처리에 대한 연구는 두 가지로 구분할 수 있다[6]. 첫째, 결함 발생원인 분석, 결함이 시스템에 미치는 영향 분석, 결함 관련 데이터 수집 메커니즘의 개발 등과 같이 시스템에서 발생 가능한 결함에 대한 모델을 정립하기 위한 연구가 있다(표2 참조). 둘째, 결함 감지 기법, 결함에 대한 대응책, 결함이 발생한 경우 복구를 위한 기술 개발 등과 같이 정의한 결함 모델을 기반으로 시스템의 대응 방안을 모색하기 위한 연구가 있다(표 3 참조). 표 2와 3은 모델 기반 결함 허용 시스템에 대한 대표적인 연구 결과[7,8,9]를 앞서 설명한 분

류 기준에 준하여 정리한 것으로, 오토노믹 시스템을 설계함에 있어 고려되어야 할 기준 항목으로 활용할 수 있다.

표 2. 결함 모델 정립을 위해 고려해야 할 특성

Property	[7]	[8]	[9]
Fault Detection	Permanent	Permanent+Intermittent	Permanent+Intermittent
Fault Manifestation	Fail fast+silent components Potentially correlated	Unexpected data feed values; Recovery only if uncorrelated	Resource exhaustion Potentially correlated
Fault Source	All non-malicious source	Representable by templates; Non-malicious	Peak resource demand; Non-malicious
Granularity	Component failure in distributed embedded system	Failure of Internet data feed	Depletion of memory, CPU, etc. in distributed system
Fault Profile Expectations	Random; arbitrary; unforeseen	Anomalies compared to prior experience	Random; resource consumption only

표 3. 결함에 대한 시스템 반응 방안 분석

Property	[7]	[8]	[9]
Fault Detection	State variable staleness	Anomaly detection	Resource monitoring alarm
Degradation	Fail-operational; Maximize system utility	Not addressed	Preserve predetermined baseline functions; eject nonessential tasks
Fault Response	Reconfigure SW based on data and control flow graphs	Substitute redundant data feed	Admission control policy; admit baseline tasks and reject some enhanced tasks
Recovery	Reconfigure SW & reboot system	On-the-fly feed switch	Terminate enhanced tasks as necessary
Time constants	Long time between failures; Can handle multiple failures	Valid data samples occur much more often than anomalies	Can handle multiple failures; Tasks can be terminated instantly
Assurance	Future work; reliability-driven	"Good enough" data quality	Static analysis of baseline load

오토노믹 시스템을 구성하기 위해서는 결함 원인 및 영향에 대한 분석이 먼저 수행되어야 하는데, 이와 관련된 기존 연구 결과를 정리하면 크게 3가지로 요약할 수 있다(그림 4)[6].

- IT 관련 서비스 제공을 위한 지출의 80% 가량이 시스템 운영 관리, 유지 보수 및 사소한 기능 향상에 소요되고 있음
- 현재 기업에서 시스템 결함이 발생하여 이를 처리하고 복구하는데 드는 비용이 총 소유비용(TCO, Total Cost of Ownership)의 33%에서 50%까지 차지함

- 시스템 결함의 주요 원인이 하드웨어에서 소프트웨어로, 또한 설치 시 발생하는 원인에서 운영상 발생하는 원인으로 전환되었음

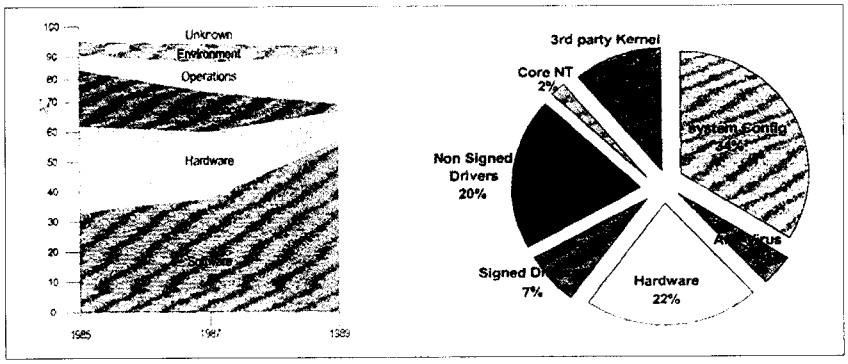

그림 4. (좌) Tandem 시스템의 주요결함 발생 원인, (우) 윈도우2002 및 NT 시스템의 주요 결함 원인

따라서 결함 발생 원인을 감지하고 이를 처리하는 일련의 과정을 자동화하여 시스템 스스로 결함에 대응할 수 있는 능력을 제공하는 것은 서비스 제공자 입장에서는 시스템 운영 관리 및 유지보수 비용을 절감할 수 있으며, 사용자 입장에서는 중단없는 안정적인 서비스를 제공받을 수 있는 효과를 가져 올 수 있다. 또한 결함 유형을 살펴보면 악의적인 형태의 공격도 신뢰성을 저해하는 요소이므로 결함허용 시스템에서 전통적으로 다루던 결함 유형 이외에 바이러스나 공격도 시스템 주요 결함으로 간주하여 통합적인 신뢰성 향상을 위한 오토노믹 시스템을 구성하여야 한다.

3. 자가 관리 메커니즘

자가 관리 메커니즘은 오토노믹 시스템을 구성하기 위한 기본 구조이다. 자가 관리 메커니즘은 표 1의 기업 별 핵심구조에서 보는 바와 같이 기본적인 내용이 유사한 특성이 있다. 자가 관리 메커니즘에서는 주로 자가 구성(Self-configuring), 자가 치유(Self-healing), 자가 보호(Self-protecting) 및 자가 최적화(Self-optimizing) 기능을 갖춘 시스템을 창출해 내는 것이 주목적이다. 자가 관리 메커니즘은 기본적인 구조는 유사하나 실제 적용에서는 여러 가지 형태로 다르게 적용된다. 실제 적용에서 문제의 인식이 중요한 경우가 있고, 어떤 경우에는 문제의 해결 자체가 중요한 경우가 있다.

본 절에서는 자가 관리 메커니즘을 3가지를 소개하고자 한다. 첫 번째, IBM의 MAPE(Monitor, Analyze, Plan, Execute)는 자가 관리 컴퓨팅 시스템에 적용할 수 있는 제어 루프와 직접 실험해볼 수 있는 툴킷을 제공한다. 두 번째는 무어대학의 OSAD (On-demand Services Assembly and Delivery) 모델이다. 이것은 서비스가 분산되어 있는 시스템 환경에서 애플리케이션의 서비스 단위로 자가 치유 기능을 제안한 것이다. 끝으로 오라클의 자가 관리에 대한 내용이다. 이는 DBMS 시스템에 대한 전반적인 자가 관리 기능을 구현하고 있다. 자세한 사항은 세부 내용에서 기술한다.

3.1 IBM MAPE[1]

자가 관리 컴퓨팅 시스템에서 이벤트 및 상태를 발견, 제어할 때 주로 제어 루프를 사용한다. 이 루프는 취급되는 이벤트를 찾는 시스템을 계속 관리한다. 그림 5의 IBM의 자가 관리 컴퓨팅 기준 메커니즘인 MAPE가 나와 있으며 이 메커니즘에 의해 제어 루프가 정의된다.

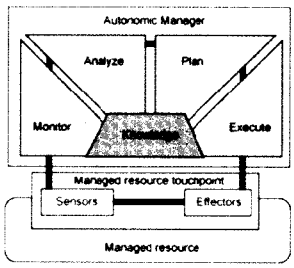

그림 5. 제어 루프

제어 루프는 이벤트들을 탐지하고 다루는 시스템이다. 이벤트를 탐지하고 다루는 과정은 다음과 같은 4단계가 있다.

1) 모니터(Monitor): 먼저 시스템은 로그 파일이건 인-메모리 과정이건 상관없이 센서에 의해 탐지되는 이벤트를 찾는다. 시스템은 지식베이스를 활용해 이벤트의 내용을 인식한다.
2) 분석(Analyze): 이벤트가 발생한 경우 지식베이스는 이벤트 해결책을 진단하는 데 도움을 주는 정보를 포함한다.
3) 계획(Plan): 이벤트를 탐지하고 분석한 뒤 시스템은 지식베이스를 이용해 이벤트 해결책을 진단한다. 증상 데이터베이스는 이벤트 정보를 포함하고 중심정책 서버는 이벤트 실행에 관한 진단을 내린다.
4) 실행(Execute): 계획을 체계화하면 기존 지식베이스에 나와 있는 대로 계획을 통해 실제 이벤트를 실행한다.

제어 루프가 단일 개념 프로세스이긴 하지만 단일 제품 상에서 수행될 필요는 없다. 예를 들어 IBM@Director사는 제어루프 과정 중 모니터 및 분석 과정을 수행하고 Toshiba Cluster Perfect사는 계획 및 실행 단계를 수행함으로써 두 회사 제품은 제어루프를 공유하고 있다.

IBM에서는 자가 컴퓨팅 툴킷을 제공하고 있다. 이 툴킷은 자가관리엔진(AME)을 포함하고 있다. 이 엔진은 제어 루프 4단계를 모두 수행한다. 물론 AME는 복잡하지 않은 제품과 교신해야 한다. 또한 AME에 적절한 리소스 모델이 있는 한 AME는 어떤 제품과도 교신한다. 로그 엔트리 또는 특수 과정 상태 가운데 리소스 모델에서 선택한 것을 AME에 나타낸다.

3.2 OSAD[10]

OSAD는 On-demand Services Assembly and Delivery 모델을 약칭하는 것이다. OASD는 오토노믹 시스템의 특징 중 하나인 자가 치유 부분에 치중한 모델로서, 다음의 네 액티비티(Activity)를 단계적으로 수행한다.

1) 애플리케이션 모니터링(Monitoring the application)
2) 결함 감지 또는 변경 계획
3) 대체 컴포넌트 디스커버리
4) 결함 컴포넌트를 대체 컴포넌트로 변경 또는 대체

OSAD 모델은 JINI 미들웨어를 사용하여 JAVA 언어로 구현되었다. JINI는 분산 환경에서 service-oriented 시스템을 지원하는 현대 IT 시스템의 전형적인 형태를 지원한다. OSAD의 자가 치유 라이프사이클은 위의 네 액티비티를 따른 모델로 그림 6과 같다.

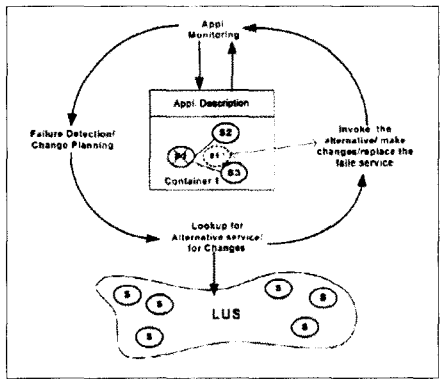

그림 6. OSAD 모델 자가 치유 라이프사이클

그림 6에서 보는 바와 같이, 가상 컨테이너와 같은 특정 장소에 서비스의 그룹들이 있으며, 서비스들을 모니터링한다. 즉, 가상 컨테이너에 있는 새로운 애플리케이션이 동작하면 변경이나 결함을 감지하기 위해 모니터링이 요구된다. 모니터링 시스템은 각 컨테이너에서 정보를 얻는다. 이 정보는 애플리케이션 라이프사이클이 개별 컨테이너에서 시작된 후에 생성되는 것이다. 이것은 애플리케이션의 위치, 애플리케이션을 구성하는 서비스들의 수, 서비스들 간의 의존도에 대한 정보를 포함하고 있다. 이러한 형태의 애플리케이션 설명서(Appl. Description)에 따라 모니터링 시스템은 컨테이너를 계속 모니터링 한다. 애플리케이션의 컴포넌트 중 하나가 결함이 발생하면 즉시 복구(Recovery) 동작을 수행한다. 결함 컴포넌트의 이름(Name)을 알고 있다면, 시스템은 미리 약속한 대로 대체를 위한 대상 컴포넌터를 찾아서 결함 컴포넌트의 위치에 대체한다. 이러한 액티비티들은 시스템 작동 중에 처리된다. 다시 말하면 시스템의 자가 치유를 수행한 것이다.

3.3 Orcle 10g 자가 관리 DB 시스템[11]

Oracle Database 10g는 성능 모니터링을 단순화하고, 성능상의 문제점을 자동 감지, 치유할 수 있는 기능을 제공한다. Oracle은 시스템 관리 영역에서 자가 관리 및 튜닝이 가능한 데이터베이스를 상용하였다. 자동적인 Undo 관리, 자동적인 SQL 실행, 메모리 관리, 그리고 자동화된 세그먼트 공간 관리 등과 같은 기능들이 데이터베이스 관리자가 일상적으로 하는 일을 대체하고 있으며, 동시에 전체 데이터베이스의 가용성 및 성능을 높이고 있다. 즉,

DBA의 반복적인 작업을 분석하여 시스템이 기계적으로 처리 가능한 부분을 시스템화한 것이다. 아래의 그림은 Oracle의 자가 관리 시스템의 메커니즘이 반영된 서비스 시스템 구조이다.

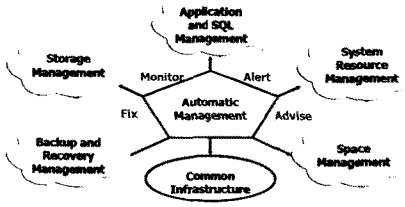

그림 7. Oracle 자가 관리 기반 구조

다음은 자가관리와 관련한 주요 모듈과 기능을 설명한다.

- Automatic Workload Repository(AWR)

AWR은 데이타베이스 운영과 관련된 성능 데이터와 통계정보를 담고 있는 영구 저장소이다. 오라클 데이타베이스는 미리 지정한 인터벌에 맞춰, 주요 통계정보와 업무부하 정보에 대해 스냅샷(snapshot)을 받아 이것을 AWR에 저장한다. 이러한 통계정보는 오라클 데이타베이스 10g의 진단 기능에 필요한 데이터이며, proactive, reactive 모니터링 모두에 필요한 정보이다.

- Automatic Database Diagnostic Monitor(ADDM)

데이터베이스 내의 자동화된 진단 감시 엔진이다. ADDM(Automatic Database Diagnostic Monitor)은 오라클 데이터베이스 시스템을 진단하고 튜닝하는데 소요되는 시간을 획기적으로 줄일 수 있는 방법을 제공한다. ADDM은 자원을 과도하게 사용하는 것으로 보이는 작업과 항목들을 집중 모니터링 함으로써 데이터베이스의 문제를 진단하고 인스턴스를 분석하는 역할을 담당하는 서버 모듈이다. 대부분의 시스템에서의 성능문제는 자원의 효율적 활용을 저해하는 병목현상으로 발생되는데, ADDM은 자동적으로 그러한 병목현상을 감지한다. 또한, 다른 오라클 관리기법과 함께 ADDM은 문제를 해결할 수 있는 권고안을 제시해 주거나 문제의 원인을 찾아내기 위한 Drill down 기법을 제공한다. 이와 반대로 튜닝이 시스템에 이점을 가져다주지 못하는 부분에 대한 정보도 제공한다. DBA, OEM에서 지정할 수 있는 것처럼 ADDM은 MMON 프로세스와 상호 통신하여 데이터베이스를 추적하고 경고정보를 생성해 낼 수 있다.

- Automatic Tuning Optimizer(ATO)와 SQL Tuning Advisor(STA)

자동 SQL 튜닝은 자동 튜닝 옵티마이저(automatic tuning optimizer)를 기반으로 한다. 자동 튜닝 모드에서 오라클 커리 옵티마이저는 튜닝 프로세스에 필요한 조사와 검증을 위해 일반적인 운영 모드에서는 사용하지 않는 동적 샘플링(dynamic sampling)이나 부분 수행(partial execution)과 같은 기술을 적용한다. 이러한 기술은 옵티마이저가 예측한 cost,selectivity, cardinality 등을 검증하는데 도움을 주며, 그 결과로 튜닝이 잘된 SQL 문장을 만

들어 낼 수 있는 가능성을 높여준다.

SQL Tuning Advisor는 SQL 문장의 실행계획을 향상시킬 수 있는 권고안을 제시한다. 단, 시스템이 SQL 문장에 의해 참조되는 객체에 대한 통계정보를 가지고 있어야 한다. Advisor는 더 나은 실행계획을 얻기 위해 SQL 문장이 어떻게 작성되었는가를 먼저 분석하고, 제약조건과 같은 요소 그리고 인덱스와 같은 다른 물리적 구조요소를 확인하여, Advisor는 현재의 실행계획 보다 더 적은 비용이 소요되는 실행계획을 작성하게 된다. Advisor는 시스템 통계정보를 갖고 실제 실행시의 동작을 검증하고 비교할 수 있는데, SQL 문장이 실행되는 동안 시스템정보를 검사하여 더 나은 권고안을 작성한다.

- Automatic Storage Management(ASM)

스토리지 관리를 자동화해 주는 서비스이다. 디스크 추가/삭제 작업시 자동적으로 데이터를 재분배해 주며, striping 효과를 데이터파일 단위로 지정할 수 있는 고성능 클러스터 파일 시스템이다.

4. 결 론

오토노믹 시스템에 대한 정의는 서비스 관리를 위한 일련의 과정에서 사용자나 개발자와 같은 사람의 간섭을 최소화하여 최종적으로는 시스템 스스로 자신을 관리할 수 있는 능력을 보유한 시스템이라 정의할 수 있다. 이를 통해 얻고자 하는 목표는 시스템 관리에 소요되는 비용 즉 전체 시스템 유지보수비용을 최소화하여 관리에 필요한 사람의 노력을 본질적인 서비스 제공에 충실하게 함으로써 서비스의 질을 향상시키고 투자가치를 높이자는 것이다.

하지만 오토노믹 시스템에 대한 명확한 정의나 구조가 정립되지 못한 상태이며 이를 구현하기 위한 핵심기술 또한 개발이 활발히 진행되고 있을 뿐 아직 해결하지 못한 문제가 많아 실제 상용 시스템에 적용하기가 어렵다. 오토노믹 시스템을 만들기 위해서는 결함과 같은 문제를 인지해 원인을 진단하고, 시스템 관리 기법이 명확하게 절차적으로 정리되어 있어야 하며, 기계적으로 처리 가능한 범위로 한정되기 때문이다.

그러나 오토노믹 시스템의 개념은 IT 기술이 발전함에 따라 그 실현 가능성이 매우 높다고 평가할 수 있다. 현재 각종 디바이스들의 성능은 좋아지고 있으며, 디바이스의 운영 환경과 관리 방법을 분석하고 정의함으로써 디바이스 스스로 문제를 찾고 해결해 지속적이고 신뢰성 있는 서비스를 제공할 수 있도록 하는 연구가 지속적으로 이루어질 것으로 예상된다.

⚜ 참고문헌

[1] IBM White Paper, "An Architecture Blueprint for Automatic Computing," 3rd Edition, June 2005.

[2] Microsoft Corporation White Paper, "Dynamic Systems Initiative Overview," Oct. 2005.

[3] HP White Paper, "Management for the Adaptive Enterprise," Feb. 2005.

[4] Intel White Paper, "Reducing Costs with Intel Active Management Technology," Aug. 2005.

[5] UKSMA, "Quality Standards Defect Measurement Manual," Oct. 2000.

[6] 최창열, "고신뢰성을 위한 자가관리 오토노믹 시스템", 아주대학교정보통신전문대학원, 2007년 1월.

[7] C. Shelton, et al., "A Framework for Scalable Analysis and Design of Sytem-wide Graceful Degradation in disributed Embedded Systems," Proceedings of the 8th IEEE International Workshop on Object-oriented Real-time Dependable Systems, pp. 156-163, Jan. 2003.

[8] O.Raz, et al., "Enabling Automatic Adaptation in Systems with Under-Specified Elements," Proceedings of the 1st Workshop on Self-Healing Systems(WOSS'02), pp. 55-60, Nov. 2002.

[9] C. Hoover, et al., "The Amaranth Framework: Policy-based Quality of Service Management for High-assurance Computing," International Journal of Reliability, Quality, and Safety Engineering, Vol. 8, No 4, pp. 1-28, 2001.

[10] E. Grishikashvili, N.B., et al., "Autonomic computing: A service-oriented framework to support the development and management of distributed applications," in: 3rd Annual Postgraduate Symposium, The Convergence of Telecommunications, Networking and Broadcasting, PGNet, 2002, School of CMS, Liverpool John Moores University, UK, 2002.

[11] Oracle, "http://www.oracle.com"

김 용 호

· 재료연구소 재료정보그룹 선임연구원
· 관심분야 : 인터넷 응용, 지능형 시스템
· E-mail : yhkim@kims.re.kr