효율적인 시스템 관리를 위한 노하우 1

시스템 관리자와 함께 하는 테마여행1

이진철 jincheol.lee@goodus.com

현재 굿어스(주) TM사업부에서 System Administrator와 DBA 업무를 담당하고 있으며, 주로 대용량 데이터베이스 시스템의 튜닝 및 컨설팅을 하고 있다. 굿어스(주)는 IT Infrastructure 전반에 대하여 OutSourcing 과 OutTasking을 제공하는 전문 IT Service 회사이다.

"시스템 엔지니어"라는 직업으로 활동하고 있는 수많은 관리자들이 오늘도 밤을 지새우며 전산실을 지키고 있을 것이다. 오늘날, 그들에게 요구되는 역할(role)은 예전처럼 단순하지 않다. 하드웨어와 몇몇 운영체제 명령어를 구사할 줄 알면, 시스템을 제대로 운용할 수 있던 시절은 이미 오래 전에 지나갔다. Enterprise Architecture를 지향하는 전산실의 업무 환경은 시스템 엔지니어에게 광범위한 기술적 지식과 또한, 자신만의 전문화를 필요로 하고 있다. 단순한 Operator가 아닌, Enterprise Architecture의 Co-ordinator로서의 시스템 관리자는 나름대로의 비즈니스 마인드과 테크니컬 비젼을 가지고 있어야 한다.

이것은 어찌보면 딜레마이기도 하다. 한 분야에서 최고의 권위자 될 정도로 전문성을 유지하면서, 전반적인 IT Infrastructure에 대하여 해박해야 한다는 것은 상반되는 개념은 아니지만, 시스템 관리자에게 끊임없는 자기 계발과 노력을 요구하는 것이다.

이 글은 System Administration의 다음 분야들을 3회에 나누어 다뤄 보고자 한다.

1. 시스템 유지보수(System Maintenance) 일반론
2. 성능관리(Performance Management) 분야
3. 가용성 관리(Availability Management) 분야
4. 용량 관리(Capacity Planing & Sizing) 분야
5. 장애 복구(Crash Recovery) 분야.
6. 서비스 엔지니어의 관점에서 바라보는 ITIL의 적용

시스템 관리라는 분야가 워낙 방대한 분야이기 때문에 어떤 분야를 다루어야 하며, 어떤 분야는 다루지 않아도 된다는 규칙같은 것은 없지만, 보편적으로 시스템 관리자들이 현재까지 해왔고, 또 앞으로 맞닥뜨리게 될 영역들을 주제로 정하게 되었다. 전반적인 글의 비중은 Administration에 관하여 Process적인 면과 Technical한 내용을 위주로 전개될 것이며, 효율적인 시스템 관리를 조언들과 경험(사례)을 포함시켜서 진행하려 한다. 부족하겠지만, 시스템 관리자이거나 시스템에 관심이 있는 사람들에게 도움이 될 만한 내용들로 정리하고자 한다.

1. 시스템 유지보수(System Maintenance) 일반론

e-비지니스 관점에서의 시스템 관리

1996년 6월 미국의 대표적인 온라인 증권회사인 찰스슈왑사는 전산 시스템이 한 시간 동안 다운되는 시련을 겪었다. 동시에 수십 만 건의 트랜잭션을 처리하던 전산 시스템은 한 순간에 고철 덩어리가 되어 버렸고, 수십 만 e-트레이딩 주자들의 항의가 빗발쳤다. 이 기업의 주식은 그 한 시간의 다운타임 동안 26%가 하락해 버렸고, 회사의 신뢰는 바닥에 떨어지고 말았다. 불안정한 IT Infrastructure의 운용이 기업의 이익 구조를 최악의 상황으로 몰고 간 것이다.

이제 IT Infrastructure는 기업 경쟁력을 좌지우지하는 핵심 기능으로 급부상하였고, 모든 업무(인사, 재무, 회계, 생산, 판매, 재고, 마케팅, 고객 서비스 ...등등)는 통합되고, 전사적자원관리(ERP), 지식관리시스템(KMS), 고객관계관리(CRM) 등의 형태로 전환되어 기업의 경영 방향을 결정짓는다. 즉, 오늘날의 기업경영에 있어서 전산 시스템 자체는 기업의 Operation 뿐만 아니라, 그 기업의 자체 경쟁력을 제고하는 중요한 전략적 도구가 되었다.

다운타임(DownTime)의 비용

다운으로 인한 비용은 값으로만 측정되지 않는다. 다운으로 인하여 잃게 되는 것은 사용자를 잃는 것과 함께 사용자가 어떠한 작업을 하다가 손해를 입었는가에 달려 있다.

만약 사용자가 개발자라면 다른 비용보다도 개발자를 고용하고 개발 기간에 든 비용이 더 클 것이다. 물론 거대한 개발 조직이라면 비용은 상당할 수 있다. 시스템 다운으로 작업할 수 없었던 부분을 보충하기 위해서 필요한 야근수당 등은 여기에서 빠져 있는 것이다.

아래의 도표는 산업 직군별 비즈니스 형태별 다운타임이 발생했을 경우 시간당 평균 비용을 나타내고 있다.(현재 시점의 환율을 적용하여 한화로 환산하여 보았다.)

표 1.1 다운타임에 따른 평균 비용(원화로 환산)

산업별	비즈니스 운영	시간당 평균 다운타임 비용
금융	증권거래	73 억 8 천 5 백만원
금융	신용카드 인증	29 억 7 천 7 백만원
미디어	유료영화 시청	1 억 7 천 1 백만원
유통	TV 홈쇼핑	1 억 2 천 9 백만원
유통	홈 카탈로그 쇼핑	1 억 3 백만원
교통	비행기 예약 시스템	1 억 2 백만원
미디어	전화티켓 예매 시스템	7 천 9 백만원
교통	택배 사업	3 천 2 백만원
금융	현금 자동 인출기 수수료	1 천 6 백만원

[자료출처 : 데이터퀘스트, 필스퀙티브, 2003년 12월 31일]

이 다운타임 비용에는 잠재고객을 다른 회사로 빼앗김으로써 장기적으로 회사에 누적될 기회비용의 손실 같은 것은 제외되어 있다. 표면적으로 회사가 입은 손실만 분석된 셈이다.

이제 시스템 관리자는 자신의 시스템이 다운되어 있는 동안 기업 입장에서 얼마만큼의 손실이 발생하는가에 대하여 정확하게 이해하고 있어야 한다. 서비스 엔지니어의 입장에서 항상 SLA(Service Level Agreement)에 관심을 가져야 하는 이유가 여기에 있다.

이제 시스템 다운으로 인한 직접 혹은 간접적인 손해 비용을 이해하고, 그에 따른 시스템 관리 포인트를 어디에서 기초해야 하는지는 자명해 진다. 경영진은 전산 시스템에 요구하는 가용성에 대하여 SLA에 반드시 규정하고 있으며, 심지어는 구체적인 전산 업무 처리량(Transaction 단위로)까지 명시하고 있다. 하루 백만 건의 트랜잭션을 처리해야만 정상적인 기업의 이익구조가 유지된다면 이 트랜잭션 량, 또는 한 건의 트랜잭션을 처리하는 데 소요되는 시간이 SLA에 명기되는 것이다.

따라서, 시스템 관리자가 관심을 가져야 하는 분야는 시스템을 구성하고 있는 하드웨어나 운영체제 자체가 아니라, 어떻게 관리하면, “이 시스템으로 기업이 요구하는 가용성과 성능, 업무처리량을 유지할 수 있느냐” 하는 것에 초점을 맞추어야 한다.

시스템 유지보수(System Maintenance)를 제언

만약, 내게 주어진 시스템이 세상에서 가장 완벽한 시스템이라면 어떨까? 이 시스템의 하드웨어, 운영체제(OS), 데이터베이스, 어플리케이션에 대하여 전혀 신경을 쓰지 않아도 시스템은 DownTime없이 운용되며, 고객(Customer & User)은 항상 최상의 만족도를 나타낸다고 상상해 보자. 정말 행복한 상상이 아닐 수 없다. 하지만 아직까지 필자가 거쳐본 수 백 가지 유형의 IT Infra 중에서 그러한 시스템은 없었다. 하다 못해 로그 파일이라도 정리하지 않으면 시스템은 새벽 3시의 단잠을 방해하는 말썽꾸러기로 변해 버리기 때문이다.

그래서, 시스템 관리의 제 1법칙은 "유지보수가 항상 필요하다"는 것이다. 그 유지보수는 시스템의 Availability와 Business affection을 최소화 시키는 범위 내에서 이루어지는 것이 바람직하다. 24시간 365일 운용되는 시스템의 유지보수를 위하여 Downtime을 요구하는 것은 Business Profit을 깍아먹는 것이므로 이러한 시스템에는 항상 최적의 대안으로 Resilience를 확보하여야만 한다. 시스템 유지보수(Maintenance)를 위한 전반적인 고려사항들을 정리해 보자.

[성공적인 유지보수(Maintenance)를 위한 계획과 고려사항]

- 유지보수 작업 전에 이 작업이 다른 서버, 클라이언트, 네트워크, 데이터베이스, 어플리케이션 등에 미치는 영향 을 항상 고려하여야 한다.

- 예상할 수 있는 최악의 상황에 대비하여 항상 계획적으로 작업을 진행하여야 한다. 데이터베이스를 업그레이드 하는데, 업그레이드 후 테이블의 모든 내용이 사라졌다든가, 파트 교체를 진행하는 과정에 정전기로 메인보드가 손상을 입게 된다든가 하는 상황을 예측해 볼 수 있다.

● 시스템에서 진행되는 작업이 완벽하게 진행되지 못했을 경우, 원상태로 되돌려 놓을 방안이 강구되어야 한다. 이러한 원상복구 계획없이 중요한 시스템을 변경하는 작업은 금물이다. 즉, 작업 전에 전반적인 환경에 대한 백업이 요구되며, 하드웨어 차원에서의 Redundancy를 확보해 놓은 상태에서 작업이 진행되어야 한다.

● Maintenance 작업에 영향을 미칠 수 있는 전반적인 상황을 분석하고, 설립된 계획에 대해서는 동료 또는 상관에게 자문을 구하는 것이 바람직하다. 실제로 수립된 계획이 완벽하다고 하더라도 동료의 계획에 대한 검토는 더 좋은 개선과 대안을 이끌어 낼 수 있다.

● 되풀이되는 업무에 대해서는 자동화된 스크립트나, 툴을 사용하는 것이 좋다. 숙련된 시스템 관리자라면 일반적으로 수행되는 일련의 작업들을 자동화함으로써 많은 시간과 노력을 줄일 수 있을 것이다. 자동화는 오타와 같은 단순한 오류를 방지해 주고, 작업을 성공적으로 완료하는데, 도움이 된다. 일일이 하는 것보다 수행속도가 빠르며, 사람이 할 일을 그 만큼 줄여주는 효과도 있다.

● SPOF(Single Point Of Failure)를 철저히 분석하여 제거하라.

SPOF(Single Poing Of Failure)는 시스템의 여러 요소들을 연결고리처럼 나열해 놓았을 때, 가장 끊어지기 쉬운 고리라고 생각하면 된다. 시스템의 어플리케이션이나, 또는 FirmWare처럼 고리 한 개만 끊어지면 전체적인 시스템상의 기능이 마비되는 취약점을 분석하여 이에 대한 대처방안을 마련해 놓아야 한다. 서버, 디스크, 네트워크 장치, 케이블처럼 SPOF 가능성이 있는 부분들은 대체 시스템을 구성해 전체 시스템이 다운되는 것을 막아야 한다.

● 시스템의 용도와 수준에 맞는 서비스 계약을 체결하라.

다음 항목들을 철저히 따져보고 이 조건을 충족시키는 서비스 계약을 체결해 둘 필요가 있다.

- 요구되는 Availability 수준 : 시스템 운영 시간을 얼마나 향상시킬 것인가?
- 시스템의 Service 시간 : 시스템에 문제가 발생하는 시간은?, 문제가 발생하는 빈도는?
- 우선순위(Priority) : 동시에 한 개 이상의 시스템에 문제가 발생할 경우 그 우선순위는?
- 서비스 대상 : 연계된 시스템이 여러 지역에 걸쳐 어떤 형태로 분포되어 있는가? 각각 제대로 서비스받고 있는가?

● 시스템의 이력을 확인하라

시스템의 최근 변동사항을 정확히 파악하지 못한 상태에서 최상의 시스템 관리는 이루어 질 수 없다. 시스템의 이력을 주의깊게 검토해 보면, 어떤 부분에서 취약한 관리 포인트가 발생하는지도 알 수 있고, 기존에 발생한 다운타임 요소와 원인, 평균 복구시간 등 다양한 현황 정보를 알 수 있게 된다. 이력사항의 파악에서 우리는 시스템 관리의 80대 20 법칙을 적용할 수 있다. 시스템 운용에 적은 영향을 미치는 단순한 에러를 극복하기 위하여 대부분의 시간을 보내는 것보다. 근본적인 문제점(20%)을 시스템 이력정보를 통하여 분석하고 해결한다면 시스템 문제의 발생량을 기대 이상으로 줄일 수 있을 것이다.

[시스템 변경시의 제안 사항]

● 시스템에 발생시키는 변경 작업은 한 번에 한 가지씩만 적용하라

필자가 초보 DBA로 데이터베이스 시스템을 운용할 때, 인스탄스 튜닝을 하면서 동시에 다섯 가지의 파라미터 값(Value)을 변경시킨 적이 있었다. 데이터베이스를 재기동 시켰을 때, 문제가 발생했는데 어떤 파라미터의 수정이 원인이 되었는지 알아 내기 위해서는 많은 시간이 소모되었다. 시스템의 하드웨어 교체, 운영체제의 업그레이드, patch, 어플리케이션의 환경 설정 등의 모든 작업에도 이 규칙은 적용되어져야 한다.

● 시스템에서 발생하는 모든 변경작업은 문서화되어야 한다.

사실 엔지니어들의 취약점이 바로 문서화인데, 자신이 작업한 내용을 반드시 틀에 얽매인 완벽한 문서로 남길 필요는 없다. 작업 중에 자신이 참조한 기술문서에 주석처리(comment) 형태로 기술해도 좋고, 작업 화면을 spooling 해서 남겨 놓아도 좋다. 요즘에는 화면에서 작업하는 모든 과정을 동영상으로 Capture 해주는 utility 프로그램도 좋은 것들이 많이 사용되어지고 있다.

● 시스템의 변경을 자동으로 관리해 주는 툴을 사용하는 것도 좋은 방법이다.

국내에서는 별로 이러한 툴들이 보편화되지 않았는데, 변경관리를 자동으로 모니터링하고, logging하면, alarm 기능을 제공하는 프로그램들은 매우 유용하다. 변경 이력사항을 정리해 놓고 있으면 이전 환경과의 비교가 용이해 복구가 쉬워지고, 관리자간의 인수인계나 심지어는 해킹에 의한 시스템 파일의 변경/파괴에도 쉽게 대처할 수 있다.

● 시스템에 적용될 모든 것을 테스트 하라

계획만 테스트할 필요가 있는 것이 아니라, 새로운 프로그램, 변경된 시스템 소프트웨어, 하드웨어 등 모든 것이 적용되기 전에 테스트 되어야 한다. 사용중인 환경과 유사하고, 가능한 실제와 동일한 시스템 환경을 구축하여 테스트하여 발생할 수 있는 모든 문제점들을 검토한 후 이상이 없는 것이 확인되면 운영기 시스템에 적용해야 한다. 이 테스트 단계에는 사용자의 시뮬레이션 참가와 성능 분석 프로그램의 도입이 반드시 병행되어야 한다. 실제로 시스템을 이용할 사용자(User)의 판단과 실제로 적용되었을 단계에서 나타날 수 있는 성능상의 문제점을 사전에 분석하고 조율할 수 있어야 한다.

● 하나의 솔류션은 그 문제에만 적용하라.

두 개의 다른 상황의 문제점에 같은 하나의 솔류션을 적용했을 경우, 이 솔류션은 대부분 다른 한 쪽에서는 문제를 야기하기 마련이다. 복잡한 문제는 다양한 영향(부수적인 문제들)을 서로 주고 받으며 난해하게 얽혀있는 것이 일반적이다. 비슷한 유형의 문제라도 사실은 내부적인 요인들에 의해 같은 솔류션으로 해결되지 않을 수 있다. 모든 문제를 하나의 솔류션이 해결해 주지는 못한다는 것이다. 오히려 다른 시스템과의 호환성 문제 등 생각지 못한 또 다른 문제가 발생할 수 있다.

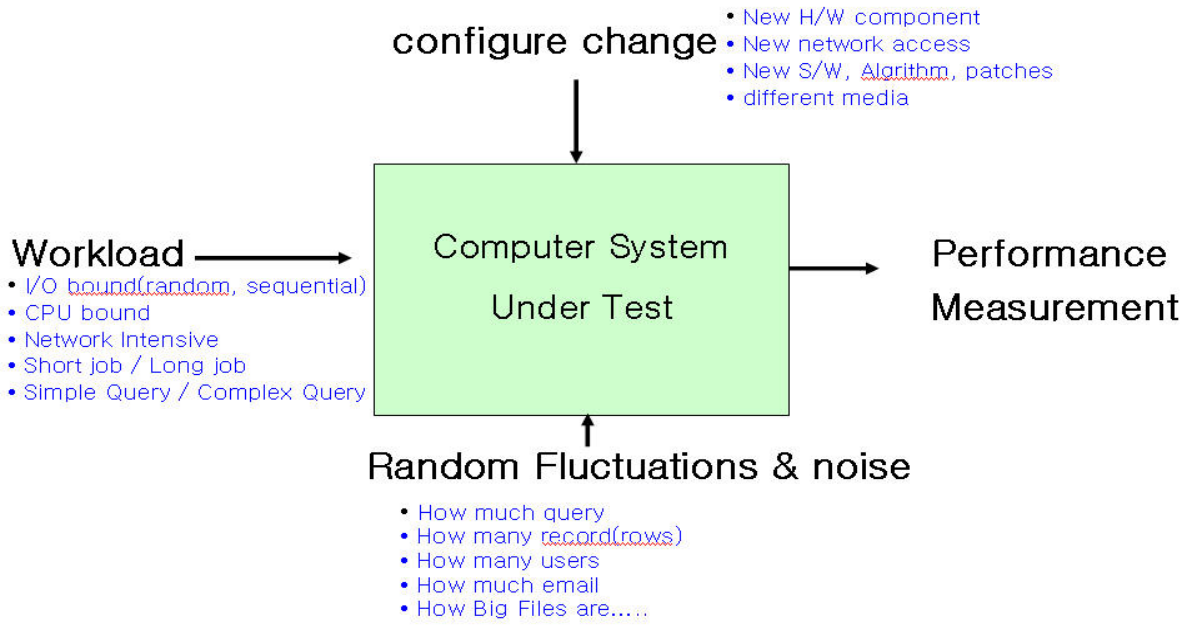
2. 성능관리(Performance Management) 분야

시스템의 성능관리는 매우 핵심적인 주제이며, 시스템 엔지니어에게 필수적인 분야이다. 이 분야에 대해서는 조금 심도있는 내용을 들여다 보아야 한다. 성능관리(Performance management)는 튜닝과 직결되는데, 시스템이나 데이터베이스, 어플리케이션에서 발생하는 Tuning Point를 이해하고, 적절한 방법을 적용하는 것은 그리 쉽고 간단하지 않다. 하드웨어의 특성이나, 운영체제의 커널(Kernel)에 대한 이해에서부터 시스템 자원(Resource)을 이용하는 전반적인 프로그램(프로세스)에 대해서도 자세히 알고 있어야 한다. 시스템 관리자는 잘못된 프로세스의 자원사용 패턴, 플랫폼별 병목현상의 특징, 심지어는 하드웨어의 성능을 무시하고 원천적으로 잘못 설치(install)된 경우의 상황분석 능력까지 가지고 있어야 한다.

성능관리(Performance Management, Tuning)는 "시스템이 가지고 있는 자원을 보다 적절히 배치해서 사용자에게 빠른 응답시간을 제공하기 위한 작업"이라고 정의할 수 있다. 구체적으로 '불필요한 작업의 Unload','자원 사용 방법의 개선', 'System capacity 증설', '유저의 시스템에 대한 이해 향상','시스템을 정확하게 업무목적에 적용'등의 Action이 수반된다.

사실, Performance Tuning은 끝이 없다. 성능을 유지하기 위한 Baseline Measurement를 설정(SLA기반)해 놓고, Bottleneck을 분석하고, 이것을 제거하는 절차는 시스템을 이용하는 Customer가 만족할 때까지 지속적으로 반복된다. 시스템의 Performance를 좌우하는 요소들을 카테고리별로 분석하면 아래 그림과 같다.

그림.1 Performance 영향요소 분석



위 그림에서 나타나는 세가지 카테고리(시스템의 환경 변화, 워크로드의 특성, 시스템의 처리량 변화)가 전반적인 시스템의 성능을 좌우하며, 따라서 이러한 각 요소들에 대해서 정확한 파악이 있어야 성능관리의 토대가 준비되었다고 볼 수 있다. 어떤 하드웨어가 설치되었으며, 이 시스템에서 실행되고 있는 운영체제 및 S/W에 대한 경향분석(예를 들어, 프로그램이 CPU 지향적인가, Disk I/O 지향적인가)은 시스템의 성능관리 및 튜닝을 위한 기초자료로서 활용될 것이다.

성능관리를 시작하기 전에 분명히 규정하고 넘어가야 할 것들이 있는데, 바로 Performance Tuning을 위한 용어 정의이다. 용어에 대한 이해가 선행되지 않으면 개념적으로 난해한 요소들에 부딪힐 수 있기 때문이다. 그 중에서 몇 가지 핵심적인 용어의 의미를 명확히 정의하여 보자. 이 용어들은 용량 관리(Capacity Planing & Sizing) 분야의 이해를 위해서라도 꼭 이해하고 있어야 한다.

<박스 1> 성능관리 분야의 용어 정리

♦ Bandwidth :
- * 초과될 수 없는 최상의 capacity
- * Overhead를 무시한 측정치
- * The best(perfect) case number
- * 실행시 도달할 수 없는 어떤 값(실행시 inefficiency(무능, 무효과, 비능률), Overhead가 있기 때문)

♦ Throughput :
- * 특정시간 동안 실제 할 수 있는 작업의 량(what you really get)
- * Bandwidth 중 실제로 사용되는 capacity
- * 실제 Throughput의 측정값은 다양한 요소들에 의해 영향 받는다.(H/W, S/W, human, Random access)
- * 정확한 Throughput 값의 측정은 불가능하며, 단지 근사치(Approximated value)만을 구할 수 있다.
- * Maximum Throughput을 측정한다면 100% 사용율(utilization)에서 얼마나 많은 작업량이 가능한가를 평가 하는 것이 된다.

♦ Response Time :
- * 총 경과시간 (+ wait time 포함, + Run Queue 대기시간)
- * user 가 응답을 받기까지 걸린 총 시간
- * Elapsed Time for an operation to complete
- * Usually the same thing as latency

♦ Service Time :
- * 실제 request를 처리하는데만 사용된 시간(Queue 대기시간, Wait Time 제외) = actual processing time
- * The best(perfect) case number
- * Response Time에서 Wait Time(Queue Delay)을 빼면 Service Time이 된다.

♦ Utilization :
- * 측정대상 작업을 수행하기 위해 사용된 자원의 사용량(독자적이건, 종합적이건)
- * 자원량 : Throughput을 수행하기 위한
- * 백분율(%)로 표시되면 일반적으로 busy%(률)로 나타낸다.

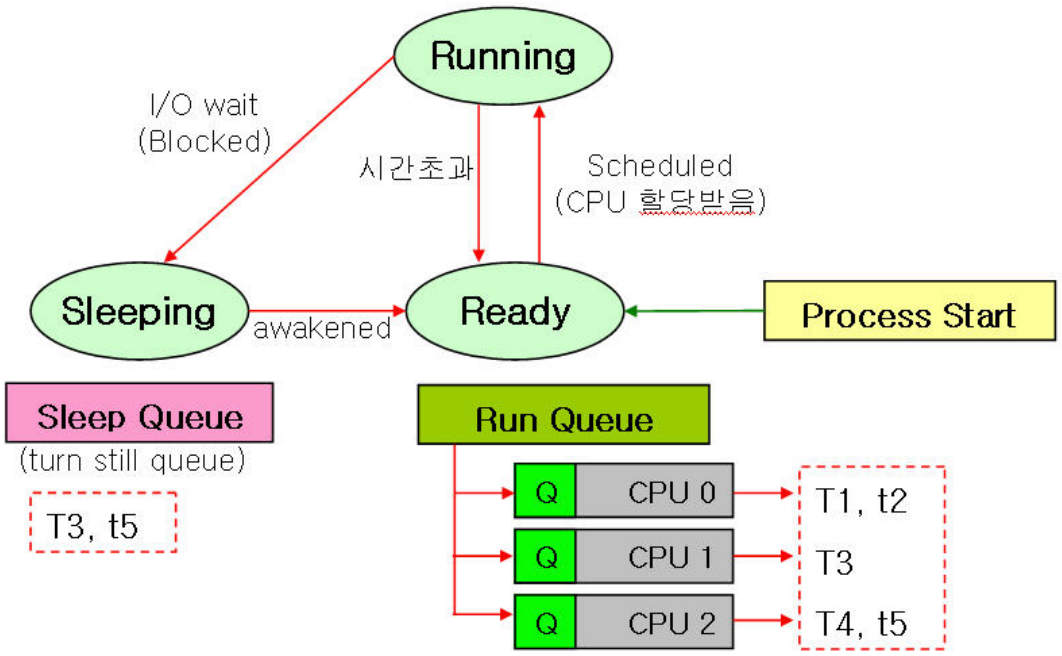
이 용어들은 서로 상관관계를 가지고 있는데, Utilization이 100%에 도달하는 순간이 되면 Response Time은 빠르게 되지만(자원의 이용방법에 따라 다르다), Troughput은 포화상태에 도달하여 떨어지게 되어 있다. 이 시점의 Throughput을 “Drop-Off"라고 한다. 결국, Performance Tuning은 100% Busy 상태에 도달한 자원(Drop-Off된 자원)의 소재를 찾아서 그 가용성을 높여주는 것이 목표이다.

그렇다면, 이제 시스템의 어떤 자원(Resource)이 100% Busy 상태에 도달하여 있으며, 어떻게 이것을 분석해 낼 것인가 하는 문제가 대두된다. 시스템의 구성 요소를 Performance Tuning의 시각으로는 CPU, Memory, Disk I/O, Network, Users(Session or Process(thread))의 다섯 가지 요소로 분리하여 분석한다.(결국, 이 다섯 가지가 자원(Resource)이다.)

일반적인 Unix system에서는 Kernel에서 발생하는 모든 system activity를 Kstat 이라고 하는 스트럭쳐에 저장해 놓고, sar, vmstat, iostat, mpstat, netstat 또는 OS dependent한 명령어나, 툴(IBM AIX의 topas, nmon, HP-UX의 glance, Sun Solaris의 SMC, memtool, 그리고 top과 같은 freeware도 많이 사용된다)을 사용하여 현재의 성능 상태 정보를 분석할 수 있다. 운영체제마다 제공하는 툴이나 명령어는 조금씩 다르지만, 기본적인 Kernel Mechanism을 이해하면, 대부분의 운영체제가 같은 방법으로 성능관리 정보를 나타내고 있음을 알 수 있다.

먼저, 아래 그림.2를 통하여 CPU 및 Disk I/O의 Bottleneck을 분석하는 원리를 이해하여 보자.

그림.2 Process Life Cycle



하나의 프로세스(thread)가 시작되면 이 프로세스를 구성하고 있는 실행 최소단위로서의 thread는 CPU 자원을 바로 획득할 수 없는 경우 CPU 에 있는 Run Queue에서 대기하게 된다. 만약 CPU 자원이 충분하게 여유가 있다면 이 thread들은 Ready 상태에서 바로 CPU 상으로 올라가 Running 상태에 도달하게 될 것이다. 결국 Run Queue 상에 Wait하고 있는 thread의 수가 일정 기준치 이상으로 나타난다면 CPU는 병목상태에 있다고 판단할 수 있다.

위 그림에서는 추가적으로 Disk I/O 병목현상에 대한 상황도 포함되어 있다. CPU상의 Running 중인 thread가 100% CPU Intensive한 Job이 아닌 한, 이 thread는 메모리나 디스크로부터 처리해야 할 데이터를 요구하게 될 것이다. 디스크로부터 대량의 데이터를 요구했을 경우 만약, Disk I/O 속도가 느리다면 Running Thread는 디스크로부터 데이터가 모두 올라올 때까지 Sleep Queue(또는 Turn Stil Queue라고도 한다)에서 대기하다가, 처리해야할 데이터가 모두 퍼올려지면 다시 Ready 상태로 가서(awakened됨), schedule deamon으로부터 CPU Time을 할당 받아, 올라온 데이터를 처리하게 된다. 이 때 Sleep Queue에 대기하고 있는 thread를 Blocked thread라고 하는데, 이 수치가 발생하는 것은 Disk I/O 병목현상이 발생하는 것을 의미한다.

일반적으로 sar 나 vmstat, iostat 명령어를 통하여 다음과 같은 정보를 얻어 낼 수 있다.

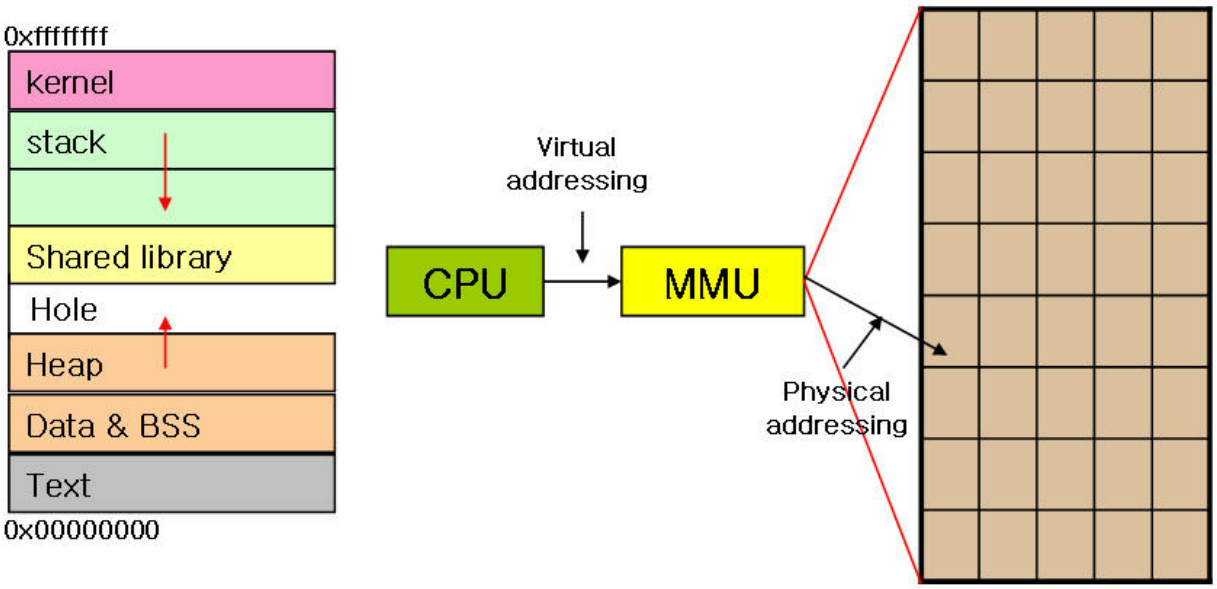
```
# sar -q 3 10

SunOS ekist3 5.8 Generic_108528-21 sun4u      10/19/04

17:31:12 runq-sz %runocc swpq-sz %swpocc
17:31:15       1.0       33
17:31:18
.....

# vmstat 3
 procs      memory            page                  disk          faults      cpu
 r b w    swap   free   re   mf pi po fr de sr m0 m1 m2 m1    in    sy    cs us sy id
 0 0 0 12882976 5217864 752 1114 2274 12 12 0 0 2 1 1 0 2091 2504 1437 198 90 548
 0 0 0 12492912 4991664 776 2 5077 0 0 0 0   0   0   0   0   943 2639 2274   4   1 95
 0 0 0 12492224 4992376 556 344 5128 0 0 0 0 0 0 0   0   0   770 3364 1981   9   3 88
 0 0 0 12492840 4995552 726 315 6114 0 0 0 0 0 0 0   0   0 1280 4235 2869   4   2 94
 0 0 0 12492832 4996656 579 0 5370 0 0 0 0   0   0   0   0 1119 2709 2266   1   2 97
 0 0 0 12492832 4996616 632 0 5130 0 0 0 0   0   0   0   0 3063 18771 6342 16 4 80
 0 0 0 12492832 4996248 643 48 5173 0 0 0 0 0   0   0   0 2003 9592 4110 24   2 74

# iostat -xnP 3
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t   %w   %b device
    0.2    1.2    9.8   10.4  0.0  0.0    0.1   22.4    0    2 c1t1d0s0
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0    0    0 c1t1d0s1
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0    0    0 c1t1d0s2
    0.0    0.0    0.0    0.0  0.0  0.0    0.0   21.7    0    0 c1t1d0s3
    0.0    0.0    0.0    0.0  0.0  0.0    0.0   25.2    0    0 c1t1d0s4
    0.1    1.3    7.1   14.3  0.0  0.0    0.0   12.9    0    2 c1t1d0s5
    0.0    1.0    0.0    0.5  0.0  0.0    0.0   24.6    0    2 c1t1d0s7
    0.2    1.2    9.8   10.4  0.0  0.0    0.1   23.7    0    2 c1t0d0s0
    0.0    0.0    0.0    0.0  0.0  0.0    0.0   15.9    0    0 c1t0d0s1
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0    0    0 c1t0d0s2
    0.0    0.0    0.0    0.0  0.0  0.0    0.0   24.3    0    0 c1t0d0s3
    0.0    0.0    0.0    0.0  0.0  0.0    0.0   26.6    0    0 c1t0d0s4
```

```
    0.1    1.3    7.1   14.3   0.0   0.0    0.0   13.0    0    2 c1t0d0s5
    0.0    1.0    0.0    0.5   0.0   0.0    0.0   25.4    0    2 c1t0d0s7
   35.6   21.4 3408.0  280.9   0.0   0.2    0.0    2.8    0   10 c4t40d0s0
    0.0    0.0    0.0    0.0   0.0   0.0    0.0    0.0    0    0 c4t40d0s2
    0.0   44.9    0.0 1437.2   0.0   0.0    0.0    1.0    0    4 rmt/0
    0.1   45.0    1.9 1440.0   0.0   0.0    0.0    1.0    0    4 rmt/1
    0.0    0.0    0.0    0.0   0.0   0.0    0.0    1.7    0    0 ekist3:vold(pid715)
```

이 명령들이 나타내는 결과는 현재 시스템에서 어떠한 Bottleneck도 발생하고 있지 않다는 것을 보여주고 있다. sar 명령의 runq-sz 필드와 vmstat 명령의 r 필드가 바로 Run Queued thread의 개수를 의미하며, b 필드가 blocked thread의 수치를 나타내고 있다. 또 한가지 iostat 명령을 통하여 나타난 asvc_t라는 필드는 average service time(실제로는 Response time)을 의미하는데, 이 필드가 20~30ms를 넘으면 Disk I/O 로드량이 많다고 볼수 있고, 50ms(밀리세컨드) 이상의 시간을 넘어서면 bottleneck이라고 규정하는 것이 일반적이다. %b 필드는 자원의 utilization을 백분율로 나타낸 것으로 이 수치가 100%에 도달하면 throughput은 drop-off 되기 시작한다.
그 외에도 해당 Disk Device에서 발생하는 초당 read/writer 량을 분석할 수 있으며 디스크 장치의 Contention으로 발생하는 wait 발생 건 수도 측정할 수 있다.

메모리 자원의 일반적인 운용원리와 성능분석 포인트를 알아보자. 일반적으로 메모리는 다음과 같은 구조로 시스템 아키텍쳐상의 역할을 수행한다.

그림.3 Virtual / Physical Memory의 운용

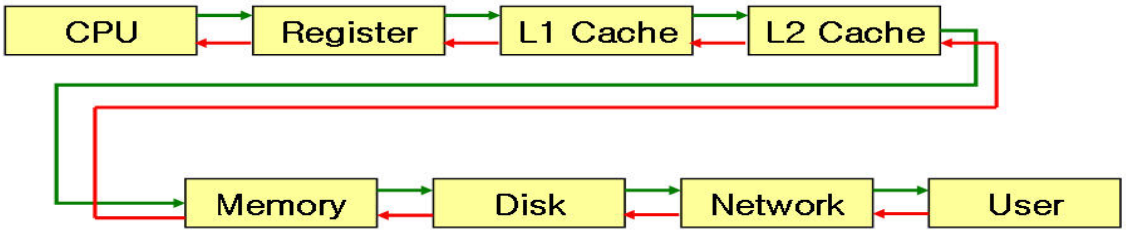


하나의 프로세스가 점유하는 메모리 영역은 Physical Memory영역과 SWAP 영역에서 동시에 구성되는데, Active 하게 실행되는 일부가 Page In & Page Out 과정을 통하여 physical memory 영역을 이용하는 것이다. Virtual Memory 기법은 제한된 Physical Memory를 프로세스가 직접 이용하는 것보다 보다 더 많은 메모리를 프로세스(Process)에게 제공하기 위하여 사용되는 기법이다. 프로세스가 이용하는 데이터는 Data & BSS 영역(상수값, Static 변수, Global 변수 등이 저장됨), Heap(data를 저장하기 위하여 malloc() 함수 등으로 동적으로 할당되는 공간), Stack(지역변수(local variable)이 저장) 영역 등에 저장되고, 이 데이터를 CPU 상의 Thread가 요구할 때, MMU(Memory Management Unit)에서 제공되는 Address Translation Table을 통하여 Physical Memory 상으로 로드하게 된다.

physical memory를 운용할 때는 일반적으로 LRU Algorithm이 사용되는데, 이것은 physical memory 상에서 최근에 자주 사용된 데이터(데이터가 있는 memory page)는 조만간 또 사용될 가능성이 있으므로 오래 남아 있도록 하고, 최근에 자주 사용되지 않은 데이터는 조만간 참조될 가능성이 적으므로 Page Out(또는 Age-Out) 하여 Free Memory를 확보하도록 하는 알고리즘을 말한다. 시스템의 Kernel 구조에서 Memory Management Module은 조만간 메모리 자원을 요구할 Process를 위하여 일정량의 Free Memory를 미리 확보하려는 구조로 설계되어 있다. 요구되는 일정량의 Free Memory가 확보되어 있는 정상적인 상태에서는 추가적인 Free Momory를 확보하려는 활동을 하지 않지만, 일단 Free Memory가 부족해지면 Page-Out Deamon은 LRU Algorithm에 의하여 자주 참조된 데이터와 참조되지 않은 데이터 영역을 검색하여 Page-In, Page-Out 활동을 시작한다. 이 활동이 극대화되어 기준치보다 많은 Page Scanning 활동이 이루어 지면 Momory Bottleneck을 규정할 수 있으며, 만일 하나의 프로세스(Process) 사용하는 Physical Memory 상의 Page를 하나씩 Page-Out 시키는 활동으로 메모리 확보가 어려워지면, 그 프로세스(Process)가 사용하는 모든 Page를 한꺼번에 Age-Out 시키는데, 이것을 Swap-Out이라고 한다. Swap-Out이 발생한 상황이라면 이것은 의심할 여지없는 Momory Bottleneck라고 볼 수 있다.

이제 Memory Bottleneck과 Disk I/O 와의 상관관계와 그 원리를 이해하여 보자.

그림.4 System Cache Hierarchies



cache	Size	Access Time	Nano Sec.	Second	Managed by
Register	1kbyte	3ns	3	3초	Compiler
L1	~32kbyte	10ns	10	10초	H/W
L2	0.5~8mb	60ns	60	1분	H/W
Memory	16mb~64gb	450ns	450	7.5분	S/W
Disk	1gb~11Tb	20ms	20,000,000	7.7개월	S/W, people
Network	N/A	200ms	200,000,000	6.5년	S/W, people

벤더사의 Platform에 따라 독특한 시스템 아키텍쳐를 가지고 있지만, 일반적인 시스템의 cache hierarchy는 위와 같은 형태로 구성되어 있다. 위 그림에서 CPU의 Register에서 데이터를 발견하여 처리하였을 경우를 3초로 환산하면, Physical Memory상에서 처리되는 상황을 7.5분으로, Memory에서 발견되지 않아서 Disk 로부터 Memory로 로드하여 처리하는 경우는 7.7개월로 환산할 수 있다. (이 비례값들은 SCSI 장치의 종류에 따라, 또는 Fiber Channel을 사용하는 경우에 따라 조금씩 달라질 수 있다.)

처리 단위를 유심히 보면, Register Memory, L1/L2 Cache, Physical Memory 까지는 nano second 단위로 데이터 이동 처리가 진행되지만, Disk I/O로 이어지는 순간 mili second 단위로 바뀐다. 결국은 시스템의 Memory 분야와 I/O 분야의 튜닝은 "어떻게 하면 Disk나, Network으로부터의 물리적인 I/O량을 줄일 것인가"에 따라 결정된다고 볼 수 있다. 가장 병목현상이 빈번하게 발생하며, 성능상에 가장 큰 문제를 유발하는 요소가 바로 이 Disk I/O 분야이다. 그래서 시스템의 데이터 캐슁 처리기법이나 어플리케이션의 데이터 Access 방법이 시스템의 Performance를 좌우하는 것이다. 예를 들어, 데이터베이스를 Access하는 어플리케이션의 SQL 문장이 다음과 같은 방법(Plan)으로 데이터를 쿼리한다면 어떨까? 가장 단순한 예이지만, 아래 나타난 두 개의 SQL 실행계획 및 그 결과를 비교해 보고, 시스템의 입장에서 분석해 보자.

```
SQL>   select SERIAL_NO,DOCU_NO,OPEN_DATE,ASSOCIATE_NAME
         from DATA_TAB1
        where SERIAL_NO='200042204000019';

SERIAL_NO,       DOCU_NO,          OPEN_DATE         ASSOCIATE_NAME
-----------------------------------------------------------------
200042204000019 98107653036       2004-10-20              홍길동

Elapsed: 00:00:06.80                                ----> Estimated Time
Execution Plan
----------------------------------------------------------
0       SELECT STATEMENT Optimizer=CHOOSE
1    0    TABLE ACCESS (FULL) OF 'DATA_TAB1'      ---> 테이블 전체를 메모리로 로드함.

Statistics
----------------------------------------------------------
          0   recursive calls
          0   db block gets
       2458 consistent gets                          ---> Physical Memory 사용량.
       2363 physical reads                           ---> 발생된 Disk I/O 량.
          0 redo size
        584 bytes sent via SQL*Net to client
        503 bytes received via SQL*Net from client
          2 SQL*Net roundtrips to/from client
          0 sorts (memory)
          0 sorts (disk)
          1 rows processed
```

[튜닝 후 데이터 엑세스 방법의 변경]

```
Elapsed: 00:00:00.14
Execution Plan
----------------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   TABLE ACCESS (BY INDEX ROWID) OF 'DATA_TAB1'
2    1     INDEX (RANGE SCAN) OF 'DATA_TAB1_IDX01' (NON-UNIQUE)
                                          --> 필요한 Index block 과 데이터 테이블만 메모리로 로드함.
Statistics
----------------------------------------------------------
  0 recursive calls
  0 db block gets
  5 consistent gets                       ---> 메모리 사용량과 Disk I/O 량을 현격하게 줄임.
  2 physical reads
  0 redo size
584 bytes sent via SQL*Net to client
503 bytes received via SQL*Net from client
  2 SQL*Net roundtrips to/from client
  0 sorts (memory)
  0 sorts (disk)
  1 rows processed
```

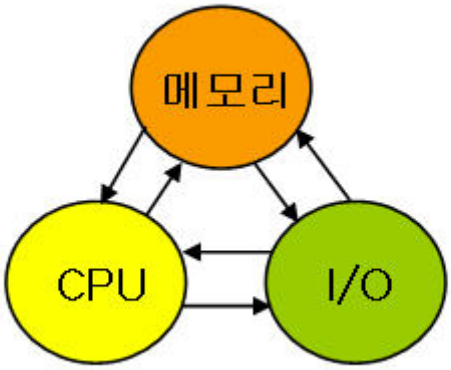
첫 번째 실행계획은 적절한 인덱스가 없었기 때문에 100만 건이 들어있는 테이블의 모든 데이터를 메모리로 로드하여 단 한 건의 데이터를 추출하고 있다. 결국, 이 SQL 문장이 한 번 실행되는 동안 2363개의 데이터 블록(8Kb block)이 Disk I/O를 통하여 메모리로 전달되었고(약 18.4Mb), 2458개의 메모리 블록을 사용(19.2Mb)한 것이다.
이 SQL 문장의 튜닝(적절한 인덱스 생성) 후, Disk I/O 량은 16Kb, 메모리 사용량은 40Kb로, 획기적으로 줄어들었으며, 처리시간도 6.8초에서 0.14초로 단축되었다. 시스템은 어플리케이션의 요구를 처리하기 위하여 Free Memory를 확보하기 위한 Page Scanning, Page In, Page Out을 수행할 것이고, 데이터의 메모리 로드를 위하여 Disk I/O 작업을 수행한다. 튜닝을 통하여 전체적인 시스템의 자원(Resource) 사용량은 1000배 이상으로, 처리시간은 48.5배로 줄어든 것이다.

지금까지 Network 자원을 제외한 시스템 상의 CPU, Disk I/O, Memory 자원의 운용과 Bottlneck을 Trace하는 원리에 대하여 알아보았다. 이 외에도 성능관리를 위하여 심도있게 관심을 가져야 하는 분야는 “Process 자체에 대한 원리”, “ System Locking Mechanism", "File System 및 대용량 Storage", ”Network" 등의 추가적인 이해가 선행되어야 한다. 앞에서 살펴 본 자원(Resource) 위주의 운용원리 이해는 시스템 성능관리를 위한, 가장 기본이 되는 요구조건들이다.

그런데, 성능관리(Performance Management)와 Tuning 작업을 하면서 고려하여야 할 매우 중요한 원칙이 하나 있다. 바로, Performance Tuning 과정에서 Bottleneck 전이(轉移)를 잘 관찰하고, 이를 고려하여 튜닝하여야 한다는 것이다. 필자는 어느 고객사의 대용량 데이터베이스 시스템을 튜닝하면서, Disk I/O 병목현상을 해결하기 위하여 엄청난 고생을 한 적이 있다.

운영체제(OS), Database, Application 등 모든 분야를 분석한 결과 어플리케이션의 데이터 Access 방식을 개선하여 Disk I/O 병목현상을 해결할 수 있음을 알게 되었다. 이 튜닝 방법이 적용되자 마자 전체 고객사의 모든 User들은 자신의 Client Application 이 Hang-Up 상태가 되어 전혀 실행되지 않는다고 아우성치기 시작했다.

원인은 Disk I/O 병목현상이 지속되던 동안 CPU는 느린 I/O 속도를 타고 천천히 올라오는 데이터를 대상으로 여유있는 작업을 하고 있었으나, Disk I/O 병목현상이 제거되자 엄청난 데이터들이 빠른 속도로 CPU 상의 Process(Thread)에게 제공되기 시작한 것이다. 결국 Disk I/O Bottleneck은 CPU Bottleneck으로 전이되어 원래의 Disk Bottleneck이 지속되던 시스템 상태보다 훨씬 심각한 장애로 나타났던 것이다.



- 튜닝은 서로 영향을 준다. → 따라서, 관련부분을 같이 튜닝하여야 한다.
- CPU문제를 해결 후 보통 I/O문제가 발생할 수 있다. (I/O 문제가 해결되면, CPU문제가 발생)
- Bottleneck은 전이(傳移) 가능하다.
- 작업특성을 고려해서 반드시 Trade-off를 만들어 놓고 튜닝 포인트를 잡아서 작업을 진행한다.

균형적인 튜닝을 위해서 권장되는 것이 반드시 Trade-Off 작업 계획서를 기반으로 튜닝을 진행하라는 것이다. 한 가지 시스템 자원(Resource)을 튜닝하여 개선할 경우 다른 자원에 어떤 영향을 미칠 것인가를 반드시 분석하여 놓고 Tuning을 실행하여야 한다. 보편적으로 Trade-Off는 성능 대 비용의 상관관계를 분석할 때 많이 사용하지만, Performance Management & Tuning 과정에서도 반드시 적용되어야 한다. 결국, 필자는 개선된 Disk I/O 속도를 적절한 수준(?)으로 낮추어 튜닝함으로써 CPU Hang-Up 문제를 해결하고, 동시에 고객이 원하는 적절한 수준의 Performance를 제공할 수 있었다.

성능관리(Performance Tuning) 및 Tuning 업무 영역을 보다 효과적으로 진행하기 위해서 요구되는 분야의 지식이 바로 Application 에 관련된 것들이다. 실제로 System Administrator가 관리하고 있는 하드웨어 자원(Resource)을 이용하는 것은 Application 이다. System에서 Performance Issue가 발생했을 때, top과 같은 명령으로 시스템을 관찰하면 전체 리소스의 대부분을 장악하고 있는 일부 프로그램(Process)을 쉽게 찾을 수 있다. 이 프로그램이 어떤 Architecture를 기반으로 만들어져 있으며, 어떤 형태로 자원을 점유하고 있는 지를 찾아내는

것이 Performance Tuning의 핵심이라고 볼 수도 있다.

실제로 Process를 실행상태를 Tracing하는 것은 프로그래머의 협조를 얻어야 하는 경우가 많다. 그러나, 시스템 레벨에서 truss 명령, adb,sdb,dbx, debug 등등의 디버깅 툴을 이용하여 시스템 레벨에서의 문제점을 찾을 수 있다. 구체적인 디버깅 방법이나 성과는 Platform(또는 운영체제)별로 매우 다양하며, 각 운영체제는 이를 위한 강력한 툴들을 지원하고 있다. 시스템 관리자라면 손에 익은 디버깅 툴 하나 정도는 능숙하게 구사할 수 있어야 한다.

결 언

몇 해전 미국 IBM의 선임 컨설턴트로 계시는 분의 초빙을 받아 시카고에서 열린 시스템의 전반적인 분야와 데이터베이스에 대한 세미나에 참석한 적이 있다. 교육 받은 내용은 둘째 치고, 미국의 IT 풍토에서 50세 가까이 되신 분이 아직도 전문 IT Engineer로서 컨설팅과 전산 인프라 관리를 담당하고 계신 모습이 매우 인상적이었다. 상대적으로 우리 나라에서는 아직도 40세를 넘기면 Sales(Marketing) 분야로의 전환을 고려해야만 하는 현실이 너무도 아쉽고 서글펐다. 썬의 “수석 엔지니어” 건, IBM의 “탑건” 이건 나이가 들어서도 가방 들고 고객사를 관리해야 하는 System Administrator의 활동에 우리의 IT 환경은 고운 눈길을 보내지 않고 있다. 신기술에 대한 적응능력이나 속도가 늦다고 해서 IT Engineer의 길을 포기하는 사례가 일반적일 것이다. 하지만, IT Engineer로서의 비젼을 고민하다, 40이 넘어 주위를 둘러 보니 같이 입사한 동료가 어느새 자신의 인사권을 좌지우지 하는 것을 보고 실의에 빠지는 현실은, 기술을 중시한다는 국가적 모토와는 무엇인가 맞지 않는 것 같다.

결국, 이러한 현실은 막대한 국가적인 손실이며, 오래된 Engineer의 경험과 노하우가 허무하게 사장되는 풍토속에서 우리의 IT Infra.가 운영되고 있는 것이다. 이제는 숙련된 프로페셔널 시스템 관리자가 인정받고 정착할 수 있는 환경이 필요하다. 이들이 IT 강국의 실질적인 반석이 되어야 할 것이다.

참고자료

❶ 시스템 가용성 100%를 향하여/Marcus Stern 저/최홍근 역

❷ 성능향상을 위한 데이터베이스 설계 비법/Kyte 저/박민호 역

❸ Sun Performance Guide / Sun Press / 2004.

전 세계에는 현재 200여명의 OCM(Oracle Certified Master)들이 활동하고 있습니다.
대한민국에는 현재 44명의 OCM들이 Professional Oracle Service를 펼치고 있습니다.
굿어스㈜에는 현재 5명의 OCM들이 여러분들의 데이터베이스를 위하여 노력하고 있습니다.