고속 병렬 컴퓨터(SPAX)를 위한 성능 감시기 설계 및 구현

김 도 형† · 김 채 규†

요 약

본 논문은 주전산기 IV로 개발된 SPAX에서 사용할 수 있는 성능 감시기의 설계와 구현에 대해 기술한다. SPAX는 지역 메모리를 갖는 노드들이 내부 네트워크에 연결되고, 노드들이 클러스터를 구성하는 계층적인 구조를 이루고 있다. 따라서, SPAX에서 효과적인 성능 감시를 하기 위해서는 SPAX의 구조적 특징을 반영한 새로운 성능 감시기가 필요하게 되었다. 구현된 성능 감시기는 SPAX의 노드, 클러스터, 그리고 전체 시스템 상태를 감시할 수 있도록 설계되었다.

Design and Implementation of Performance Monitor on Highly Parallel Computer

Do-Hyung Kim† · Chae-Kyu Kim†

ABSTRACT

This paper describes the design and implementation of performance monitor which can be used at SPAX that is developed to TICOM IV. SPAX has a hierarchical structure, at which nodes which have a local memory, are connected to interconnect network and constructed to clusters. So, to do effectivel performance monitoring at SPAX, new performance monitor is required, which is designed to consider the structure of SPAX. Implemented performance monitor is designed, which can monitor the state of node, cluster, and total system of SPAX.

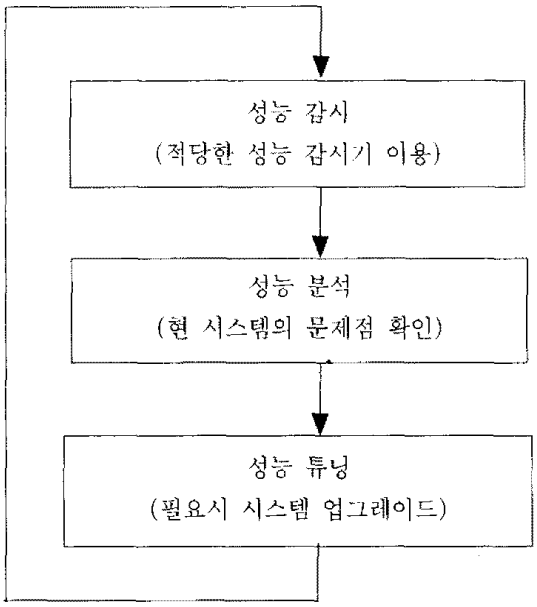
1. 서 론

유닉스 시스템의 사용이 보편화되면서 유닉스 시스템에 대한 성능 관리 또한 시스템 관리자의 중요한 임무가 되었다. 시스템의 성능을 정의하기는 어렵지만, 일반적으로 '컴퓨터 자원이 처리하고자 하는 일을 얼마나 잘 처리하는 가'로 말할 수 있다. 시스템의 성능을 효과적으로 관리하기 위해서는 그림 1과 같이 성능 감시, 성능분석, 그리고 성능 튜닝까지의 작업을 반복하여야 한다[9].

성능 감시는 시스템에 적절한 성능 감시기를 선택하여 계속적으로 시스템 상태를 감시하는 것을 말한다. 성능 분석이란 시스템 내에 어떤 문제가 존재하는지를 확인하기 위하여 성능 감시로부터 얻은 데이터를 분석하는 것을 말한다. 성능 튜닝은 시스템 성능을 향상시키기 위한 조치를 취하는 것이다. 성공적인 성능 관리는 규칙적으로 시스템을 감시하고 시스템 성능에

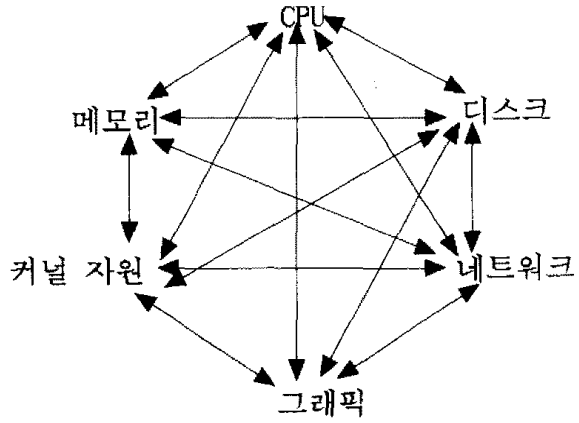
† 정 회 원 : 한국전자통신연구원 컴퓨터·소프트웨어 기술 연구소
논문접수 : 1998년 4월 30일, 심사완료 : 1998년 7월 10일

문제가 발생하기 전에 성능 튜닝 또는 시스템 용량 증가등의 의사결정을 내리는 것이다[11].



(그림 1) 성능관리 사이클
(Fig.1) Performance Management Cycle

성능 감시에서는 시스템에 적절한 감시기를 선택하여 시스템 사용 상태를 감시하게 되는데, 성능 감시기의 중요한 역할은 시스템 성능을 저하시키는 병목(Bottleneck)을 관리자가 쉽게 찾도록 도와주는 것이다. 일반적으로 어떤 자원이 병목을 발생시킨다는 것은 그 자원에 대한 요구사항의 크기가 자원의 사용가능한 크기보다 크다는 것을 의미한다. 다른 말로 하면, 병목현상이란 하드웨어 또는 소프트웨어 구성원 또는 시스템 구조의 부적합성으로 인한 시스템 성능의 한계를 말한다[11]. 그림 2는 시스템 병목을 발생시키는 구성원들의 관계를 보여준다. 전통적으로, 시스템 병목은 주로 CPU, 메모리 및 디스크에 관련되었지만, 최근에는 네트워크나 그래픽도 관련된다. 하지만, 그래픽과 네트워크 병목은 찾기 어렵고, 보통 특수한 제품을 사용하게 된다(예를들어 HP의 Openview Network Node Manager)[11]. 일반적으로 성능 감시기에는 시스템 자원들의 현 상태를 감시하기 위한 다수의 기능들을 가지고 있어서, 시스템 관리자로 하여금 시스템의 현 사용 상태를 감시할 수 있도록 한다[13]. 본 논문에서는 주전산기 IV에서 동작하는 성능 감시기 설계와 구현에 대해서 다룬다.



(그림 2) 병목의 구성원
(Fig. 2) The Member of Bottleneck

현재까지 많은 수의 성능 감시기들이 구현되었다. 구현된 성능 감시기들은 목표로 하는 컴퓨터를 감시하기 위한 다양한 성능 감시 항목들과 편리한 사용자 인터페이스를 제공한다. 기존 유닉스에서도 시스템 상태를 감시하기 위한 감시 명령어들이 존재한다. BSD 계열의 pstat, iostat, vmstat과 시스템 V계열의 sar, rtpm등이 그것이다. 이들 명령어들은 다양한 종류의 시스템 상태 정보를 관리자에게 텍스트 형태로 제공한다. 하지만, 현재 구현되고 있는 성능 감시기들은 대부분 GUI (Graphic User Interface)를 이용하여 관리자가 감시하고자 하는 항목을 감시 화면에서 선택할 수 있고, 시스템의 상태 정보를 그래프 형태로 출력한다. 일반적으로 성능 감시기는 벤더에서 하드웨어와 함께 제공되거나, 혹은 제3의 벤더에서 제공되는 경우가 있다.

CA-Unicenter[5]는 제 3의 벤더에서 제공되는 시스템 관리 툴로써, 시스템 관리자가 사용할 수 있는 다양한 관리 기능들을 포함하고 있다. 성능 감시기도 하나의 관리 기능으로 관리 툴 내에 포함되어 있다. 윈도우 95에서 제공되는 성능 감시기는 감시 항목들을 메모리, 커널, 파일 시스템등으로 구별하여 관리자가 원하는 감시 항목들을 선택할 수 있도록 하였다[8]. 하지만, 모든 감시 그래프가 같은 윈도우 프레임에 표시되어 관리자가 감시 항목을 많이 선택하게 되면, 그래프 사이즈가 줄어들게 되어 감시 그래프 간의 구별이 어렵다. 그리고, 관리자가 실수로 감시 항목을 중복되게 선택해도 별도의 그래프가 다시 생성되기 때문에 그래프 개수가 많아지는 문제점이 있다.

성능 감시기 XSam과 System Monitor는 벤더에서

하드웨어와 함께 제공된다. Cray에서 제공되는 성능 감시기 XSam은 다양한 성능 감시 항목들을 관리자에게 제공하고, 시스템 내의 메모리 맵과 디스크 구성 정보를 제공한다[4]. XSam에서는 선택 메뉴에 많은 수의 성능 감시 항목들이 제공되지만, 각 항목들이 적절하게 구분되어 있지 않아 사용하기에 복잡하다. 유닉스웨어 2.0에서는 System Monitor가 제공된다[6]. System Monitor는 비교적 간단한 사용자 인터페이스를 제공하지만, 관리자가 선택할 수 있는 감시 항목 수가 너무 적어 효과적인 성능 감시가 어렵다.

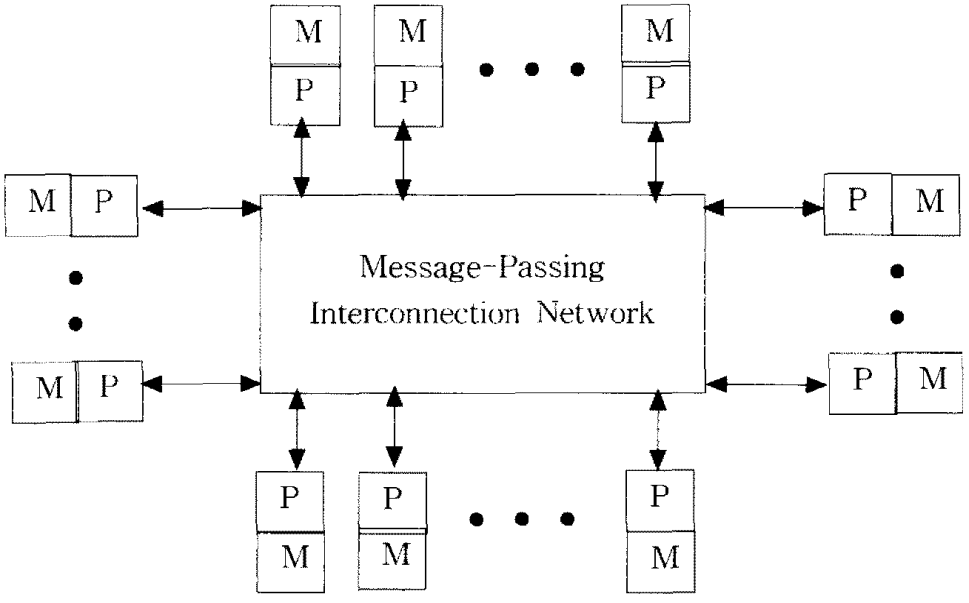
본 논문에서 다룰 성능 감시기는 주전산기 IV로 개발된 SPAX(Scalable Parallel Architecture Computer based on Crossbar Network)의 구조적 특징을 반영한 성능 감시기이다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 SPAX의 구조적 특징을 소개하고, 3장에서는 성능 감시기의 설계 및 구현에 대해 기술한다. 4장에서는 구현된 성능 감시기를 예를 들어 설명하고, 5장에서는 결론을 내리도록 한다.

2. SPAX의 구조

구현된 성능 감시기는 주전산기 IV로 개발된 SPAX에서 동작한다. SPAX는 지역 메모리를 갖는 노드들이 메시지 전달을 위한 내부 네트워크에 연결되는 이전의 병렬 컴퓨터와는 다른 구조적 특징을 가지고 있다. 그림 3은 지역 메모리를 갖는 노드들이 메시지 전송 내트워크에 연결되는 병렬 컴퓨터들의 일반적인 구조를 보여준다[14].

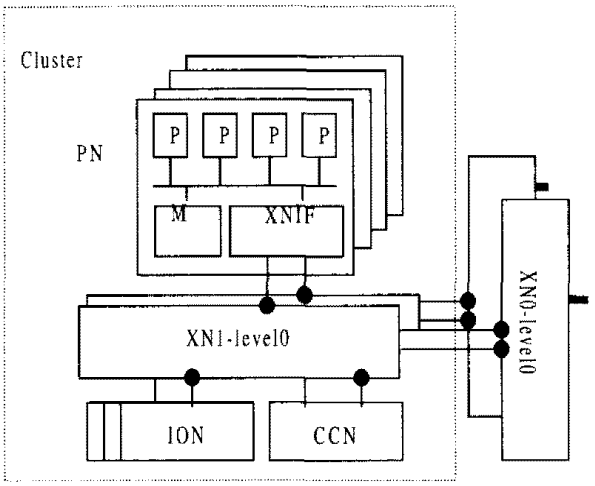
그림 3에서 보듯이 지역 메모리를 갖는 노드들이 내부 네트워크를 통해 서로 연결되지만, 노드들이 클러스터를 구성하는 계층적인 구조를 갖지않는다. 하지만, SPAX는 다중 처리기 노드들이 고속 내부 네트웍에 연결되어 클러스터를 이루는 계층적인 구조를 가지고 있다[7]. 즉, SPAX는 각 노드들이 지역 메모리를 가지고 내부 네트워크를 통해 서로 연결되고, 노드들이 클러스터를 구성하는 구조적 특징을 가지고 있다. 그림 4는 SPAX의 구조를 보여준다.

SPAX는 최대 16개의 클러스터들로 구성될 수 있고, 각 클러스터에는 프로세싱 노드(PN), 입출력 노드(ION), 통신 접속 노드(CCN)들이 최대 8개까지 다양한 구성을 가질 수 있다. 각 프로세싱 노드는 최대 4개의 P6 마이크로 프로세서, 1GB 지역 메모리, 라우팅 보드인 XNIF(Xcent-Net Interface)로 구성된다. 운영체제 MISIX(Micro-kernel based Single system Image UNIX)는 Chorus의 마이크로 커널을 기반으로 유닉스웨어(Unixware) 2.x를 다중 서버화한 것으로, 클러스터링 기반 병렬 시스템 환경 지원과 소프트웨어적 시스템 가용성 향상에 중점을 두고 설계되었다[12]. 그리



(그림 3) 메시지 전달 멀티컴퓨터의 일반적 모델
(Fig. 3) The Generic model of a message-passing multicomputer

고, 시스템의 하드웨어 의존적인 부분은 마이크로 커널이 담당하며, 서버에서 시스템 전체의 환경을 단일 시스템 이미지(Single System Image)로 관리하여 준다[15].



(그림 4) SPAX 구조
(Fig. 4) The Structure of SPAX

SPAX는 지역 메모리를 갖는 노드들이 내부 네트워크에 단순히 연결되는 이전의 병렬 컴퓨터와 달리 노드들이 클러스터를 구성하는 계층적 구조를 이루고 있다. 따라서, SPAX에서 효과적인 성능 감시를 하기 위해서는 각 노드의 상태뿐만 아니라, 클러스터, 그리고 시스템 전체의 상태를 감시할 수 있어야 한다. 하지만, 기존의 성능 감시기들은 다수의 노드들이 클러스터를 구성하는 SPAX의 구조적 특징을 반영하지 못해 SPAX에서 효과적인 성능 감시를 할 수 없다. 따라서, SPAX의 구조적 특징을 반영한 새로운 성능 감시기가 필요로 하게 되었다.

3. 성능 감시기의 설계 및 구현

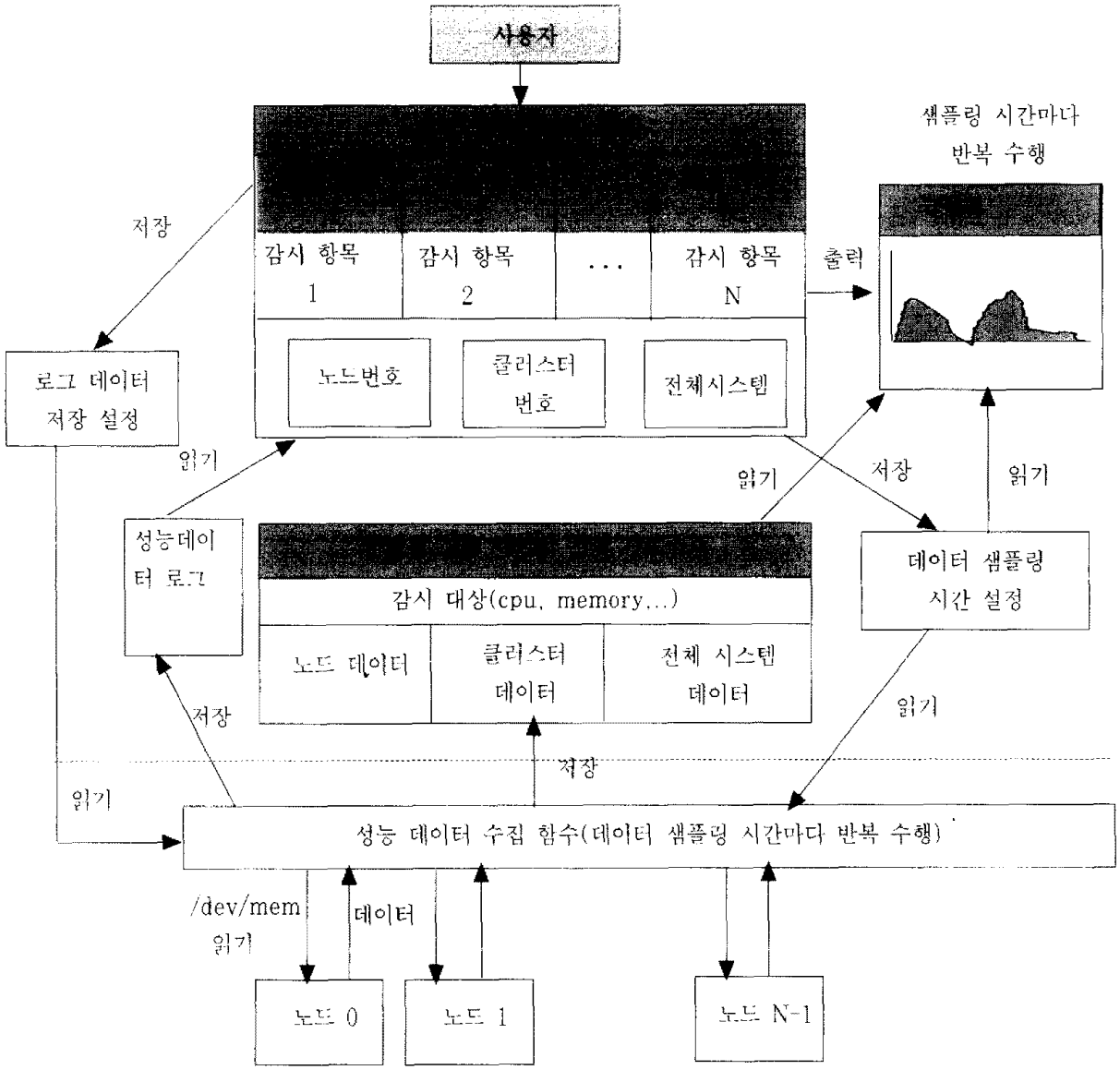
일반적으로 성능 감시기는 시스템에서 발생한 병목을 발견하기 위해 다수의 감시 항목들을 가지고 있다. 하지만, 시스템 병목은 CPU, 메모리, 디스크등의 복합적인 요인으로 발생하기 때문에, 병목 발생시에 영향을 받는 감시 항목들을 정확히 성능 감시기에 포함시키기 어렵다. 따라서, 기존의 성능 감시기들은 단순히 많은 수의 감시 항목들을 제공하기도 한다. 하지만, 너무 많은 감시 항목들은 성능 감시기 사용을 어렵고, 복잡하게 만들 수도 있다[4,8]. 반대로, 너무 적은 감시 항목들은 시스템에서 발생한 병목을 발견하기 어렵게 한다. 따라서, 시스템 병목을 발견할 수 있는 감시 항목들이 감시기내에 포함되어야 하겠지만, 이들을 적절히 구분하는 것도 중요하다. 성능 감시기는 벤더에서 하드웨어와 함께 제공되거나 제 3의 벤더에서 제공되기도 한다다. 특정 시스템에서 효과적인 성능 감시를 위해서는 성능 감시기가 그 시스템의 구조적 특징을 반영하도록 설계되어야 한다.

구현된 성능 감시기는 SPAX의 구조적 특징을 반영하여 노드나 클러스터 단위의 성능 감시가 가능하고, 관리자가 쉽게 사용할 수 있는 인터페이스를 제공한다. 그리고, 관리자에게 다양한 감시 항목들을 제공하면서도, 이들을 적절히 구분하여 사용하기에 복잡하지 않도록 설계되었다. 이 장에서는 구현된 성능 감시기의 구조와 성능 감시 항목에 대해 소개한다.

3.1 성능 감시기의 구조

그림 5는 성능 감시기의 기본 구조이다. 성능 감시기는 필요한 감시 데이터를 시스템 내의 동작 중인 노드들로 부터 수집하는 성능 감시기 하부 구조와 사용자 인터페이스를 제공하는 성능 감시기 상위 구조로 구성되어 있다.

성능 감시기 하부 구조의 성능 데이터 수집 함수는 성능 감시기를 수행하는 순간부터 종료될 때까지 설정된 데이터 샘플링 시간마다 반복 수행된다. 성능 데이터 수집 함수는 크게 동작중인 노드들로부터 데이터를 수집하는 부분과 수집된 노드 데이터를 이용하여 클러스터와 전체 시스템 데이터를 계산하는 부분으로 구성된다. 데이터 수집 부분은 현재 시스템 내에서 동작중인 노드들의 정보를 이용하여, 각 노드의 /dev/mem/으로부터 필요한 데이터를 수집, 가공하여 정의된 노드 자료구조에 저장한다. 모든 노드들로부터 데이터를 수집한 후에, 데이터 계산 부분이 수행된다. 데이터 계산 부분은 저장된 노드 데이터와 클러스터 구성 정보를 이용하여 클러스터와 전체 시스템 데이터를 계산하게 되는데, 클러스터 구성 정보는 어떤 노드가 어떤 클러스터에 속해있는지에 대한 정보를 제공한다. 즉, 데이터 계산 부분은 저장된 노드 데이터들을 차례로 읽어, 그 노드가 속한 클러스터의 데이터와 전체 시스템 데이터를 계산하여, 정의된 클러스터와 전체 시스템 자료구조에 저장한다. 따라서, 성능 데이터 수집함수가 수행되게 되면, 성능 감시기에서 필요한 모든 데



(그림 5) 성능 감시기 기본 구조
(Fig. 5) The Basic Structure of Performance Monitor

이터들이 노드별, 클러스터별, 전체 시스템 별로 해당 자료구조에 저장되게 된다. 다음은 성능 감시기에서 정의된 자료 구조를 간략히 보여준다.

시작하도록 설정할 수 있는데, 성능 데이터 수집함수는 매 샘플링 시간마다 로그 저장 기능이 설정되어 있는지를 조사하게 된다. 만약 설정되었다면, perfdata 자료구조에 저장된 데이터를 지정된 로그 파일에 저장한다. 그리고, 저장된 로그 데이터는 로그 표시 기능에 의해 출력될 수 있다.

성능 감시기 상위 구조인 사용자 인터페이스는 모든 감시 기능들을 하나의 화면에서 선택할 수 있도록 한 성능 감시기 초기 화면과 각 감시 기능을 위한 감시 화면, 그리고 출력 그래프로 구성된다. 모든 감시 기능들은 동일한 방식으로 설계되어 관리자가 쉽게 수행할 수 있고, 마우스의 클릭만으로 모든 입력이 이루어진다.

관리자가 특정 감시 기능을 수행하기 위해서는 두 단계의 수행 절차가 필요하다. 먼저, 성능 감시기 초기 화면에서 원하는 감시 기능을 선택해야 한다. 관리자가 특정 감시 기능을 선택하면, 선택된 감시 기능을 위한 감시 화면이 생성된다. 두 번째는 생성된 감시 화면에서 필요한 항목들을 선택한 다음, OK 버튼을

```
#define  MAXCPU      4                  /*  최대 4 CPUs / node       */
#define  MAXNODE     128                /*  최대 128 nodes / system  */
#define  MAXCLUSTER  16                 /*  최대 16 cluster / system */

/* CPU data structure */
typedef struct _CpuInfo {
     float     node[MAXNODE][MAXCPU][5];
                    /* 0: usage, 1: usr, 2: sys, 3: wio, 4: idle */
     float     nodeusg[MAXNODE];        /* usage */
     float     cluster[MAXCLUSTER];     /* usage */
     float     total;                   /* usage */
}T_CpuInfo;
. . .

/* perfdata structure */
struct Perf_Data {
     T_CpuInfo          cpu;           /*   CPU usage    */
     T_MemInfo          memory;        /*   memory usage*/
     T_QueueInfo        queue;         /*   queue length */
     T_PagingInfo       paging;        /*   paging info  */
     T_SwapingInfo      swaping;       /*   swaping info */
     T_CacheInfo        cache;         /*   cache info   */
     T_CallInfo         syscall;       /*   system call  */
     T_SystemTable      systbl;        /*   system table */
     T_FileSystem       file;          /*   file system  */
     T_Disk             disk;          /*   disk info    */
} perfdata;
```

눌러 감시 그래프를 생성하는 것이다. 감시 그래프는 관리자가 설정한 데이터 샘플링 시간마다 필요한 데이터를 perfdata 자료구조에서 읽어 화면에 표시하게 된다.

3.2 성능 감시 기능과 감시 항목

시스템 내의 특정 자원에서 발생한 병목을 발견하기 위해서는 발생한 병목이 영향을 미치는 감시 항목들이 성능 감시기 내에 포함되어야 한다. 하지만, 시스템 병목은 CPU, 메모리, 디스크등의 복합적인 요인으로 발생하므로, 발견하기가 쉽지 않고, 병목을 발견하기 위한 감시 항목들을 정확히 정의하기도 어렵다. 다음은 일반적으로 CPU, 메모리, 디스크 병목들이 어떤 감시 항목을 통해 발견될 수 있는 지를 보여준다.

CPU 병목을 발견하기 위한 항목들에는 전체 CPU 사용율, 시스템과 사용자 CPU 사용율이 있을 수 있다. 전체 CPU 사용율이 100% 이하인 경우에 CPU 병목은 없지만, CPU 사용율이 100%를 초과할 경우에는 전체 메모리 사용율(메모리 병목)과 전체 디스크 사용율(디스크 병목)을 조사해야 한다. 메모리 병목 현상이 발생할 경우에는 높은 스와핑 활동, 높은 페이징 활동, 아주 적은 사용가능한 메모리, 스왑 영역에 대한 높은 디스크 I/O, 시스템 모드에서 높은 CPU 활동, 그리고 높은 프로세스 실행 큐 현상이 나타난다. 디스크 병목이 발생할 때에는 높은 디스크 I/O와 디스크 I/O를 기다리는 CPU 유휴 상태, 높은 디스크 큐 길이 현상이

발생한다. 기타 I/O 병목현상은 높은 I/O 활동, 처리율의 저하 현상이 나타난다. 커널 자원 중 시스템 테이블 병목 시에는 높은 시스템 CPU 사용율, 높은 시스템 호출 비율, 높은 시스템 호출 에러 현상들이 발생하게 된다[6,11].

구현된 성능 감시기는 CPU, 메모리, 디스크 병목들을 발견할 수 있는 감시 항목들을 가능한 포함하도록 설계되었고, 포함된 감시 항목들을 감시되는 자원에 따라 구분하였다. 특정 자원을 감시하기 위한 감시 기능에는 CPU, 메모리, 디스크, 파일 시스템, 페이징, 스와핑, 시스템 호출, 시스템 버퍼, 시스템 테이블들이 포함된다. 그리고, CPU 감시 기능은 CPU 사용율 감시와 CPU 실행 큐 감시로 나누어지고, 시스템 버퍼, 시스템 호출, 시스템 테이블 감시 기능들은 구현상 하나의 감시 화면에 통합하였다.

CPU 사용율 감시기능은 각 CPU마다 전체 시간을 100으로 놓았을 때 단위 샘플링 시간 동안, 전체 CPU 사용율(%usage), 시스템 CPU 사용율(%sys), 사용자 CPU 사용율(%usr), 디스크 I/O를 기다리는 시간의 비율(%wio), 어떤 작업도 하지 않고 쉬는 시간의 비율(%idle)을 감시할 수 있는 항목들을 포함하고, CPU 실행 큐 감시 기능은 단위 샘플링 시간 동안, 평균 각 CPU 실행 큐의 길이(prunq/s)를 감시할 수 있는 항목을 가지고 있다.

메모리 사용율 감시 기능은 샘플링 시점에서의 전체 메모리 크기에 대한 사용 가능한 메모리 크기의 백분율(%usage) 정보를 감시할 수 있는 항목을 포함한다. 페이징 활동 감시 기능은 단위 샘플링 시간 동안, 초당 주소 변경 페이지 폴트 수(vflat/s), 초당 커널 리사이클링 정책에 의해 free 리스트에 놓여지는 페이지 수(pfree/s), 초당 커널 리사이클링 정책에 의해 스캔되는 가상 메모리 페이지 수(vscan/s), 초당 페이징 아웃되는 페이지 수(ppgout/s), 초당 페이징 안되는 페이지 수(ppgin/s)를 감시할 수 있는 항목을 가지고 있다. 그리고, 스와핑 활동 감시 기능은 단위 샘플링 시간 동안, 초당 스왑 인되는 페이지 수(pswin/s), 초당 스왑 아웃되는 페이지 수(pswot/s), 스왑 큐에 있는 프로세스들의 수(swapq/s)를 감시할 수 있는 항목들을 포함한다.

파일 시스템 감시 기능은 샘플링 시점에서 시스템 V 계열 파일 시스템(s5), 유닉스 파일 시스템(ufs)/보안 파일 시스템(sfs), 베리터스 파일 시스템(vxfs) 들의 전체 블록 수에 대한 사용 가능한 블록 수의 백분율(%block usage), 전체 i-node 수에 대한 사용 가능한 i-node 수의 백분율(%inode usage) 정보를 감시하기 위한 항목과 단위 샘플링 시간 동안 시스템 V 계열 파일 시스템(s5), 유닉스 파일 시스템(ufs)/보안 파일 시스템(sfs), 베리터스 파일 시스템(vxfs) 들의 초당 i-node 엔트리에 의해서 지정된 전체 파일의 수(iget/s)를 감시할 수 있는 항목을 가지고 있다.

디스크 감시 기능은 단위 샘플링 시간 동안 특정 디스크의 초당 읽기와 쓰기 연산의 수(r+w/s), 초당 전송된 디스크 블록들의 수(transfer blocks/s), 디스크가 서비스를 제공하는 데 사용한 시간의 백분율(%busy), 디스크 연산 요청들이 디스크 큐에서 기다리는 평균 대기 시간(average wait time/s), 하나의 디스크 연산 요구가 디스크에 의해서 종료될 때까지의 평균 시간(average service time/s)을 감시하기 위한 항목과 샘플링 시점에서 전체 스왑 영역 크기에 대한 사용 가능한 스왑 영역 크기의 백분율(%swap space usage) 정보를 감시할 수 있는 항목을 가지고 있다.

시스템 호출 감시 기능은 단위 샘플링 시간 동안 초당 문맥교환 횟수(process switch/s), 초당 전체 시스템 호출 횟수(total system call/s), 초당 fork 시스템 호출 횟수(fork/s), 초당 exec 시스템 호출 횟수(exec/s)를 감시할 수 있고, 시스템 버퍼 감시 기능은 단위 샘플링 시간 동안 시스템 버퍼에서 논리적인 읽기가 일어날 비율(%rcache), 시스템 버퍼에 논리적인 쓰기가 이루어질 비율(%wcache), 디렉토리 네임 룩업 캐쉬 힛트 비율(%dnlc) 정보를 감시할 수 있는 항목을 가지고 있다. 그리고, 시스템 테이블 감시 기능은 샘플링 시점에서 커널에 의해 할당되고 사용되는 프로세스 테이블 엔트리 수(process), 커널에 의해 할당되고 사용되는 i-node 테이블 엔트리 수(inode), 커널에 의해 할당되고 사용되는 LWP(Light Weight Process) 테이블 엔트리 수 (lwp), 커널에 의해 할당되고 사용되는 파일 테이블 엔트리 수(file), 커널에 의해 할당되고 사용되는 공유 메모리 레코드 테이블 엔트리 수(shared memory), 그리고 각 시스템 테이블의 오버플로우 횟수(process fail, lwp fail, inode fail, file fail, shared memory fail) 정보를 감시할 수 있는 항목들을 포함한다.

마지막으로, 로그 감시 기능은 로그 파일에 저장된 각 감시 기능들의 감시 항목들을 출력할 수 있다.

관리자는 시스템 병목 발생시에 어떠한 감시 항목 값들이 변화하는지를 조사함으로써 병목을 발생시키는 자원을 찾을 수 있다. 구현된 성능 감시기는 성능 감시에 필요한 감시 항목들을 필요 시에 쉽게 추가할 수 있도록 설계되었다.

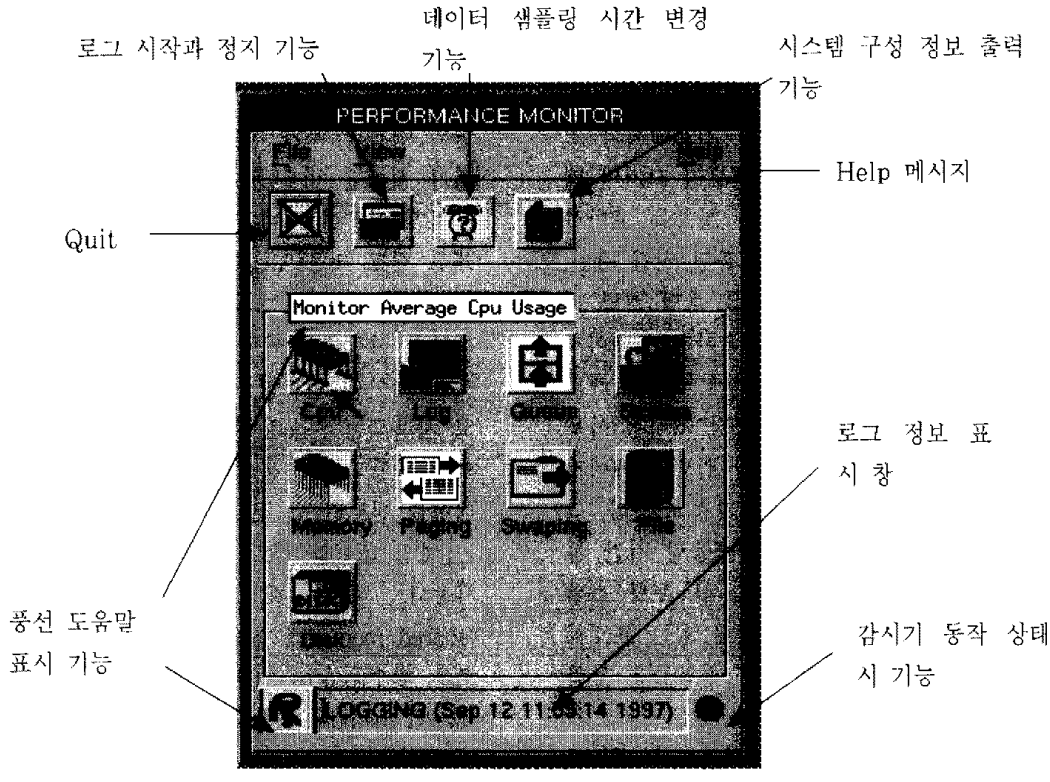
4. 사용자 인터페이스

구현된 성능 감시기의 사용자 인터페이스 특징은 SPAX의 노드나 클러스터 단위의 성능 감시가 가능하고, 관리자가 사용하기 쉽도록 설계되었다는 것이다. 성능 감시기 내의 대부분의 감시 기능 화면들은 특정 노드, 클러스터등을 선택할 수 있고, 일정한 규칙에 따라 설계되어, 동일한 방식으로 수행된다. 이 장에서는 구현된 성능 감시기의 초기 화면과 CPU 감시 기능, 페이징 감시 기능, 그리고 파일 시스템 감시 기능을 예로 들어 사용자 인터페이스의 특징들을 보여준다.

4.1 성능 감시기 초기 화면

관리자가 처음 성능 감시기를 수행하게 되면, 그림 6과 같은 성능 감시기 초기 화면이 생성된다. 성능 감시기 초기 화면에서 각 성능 감시 기능은 하나의 버튼으로 표현되어 있다. 만약 관리자가 감시기 초기 화면에서 특정 감시 버튼을 선택하게 되면, 선택된 기능을 위한 감시 화면이 생성된다. 성능 감시기 초기 화면은 관리자에게 모든 성능 감시 기능들을 하나의 화면에서 선택할 수 있도록 하는 기능외에, 성능 감시를 도와주는 다수의 부가적인 기능들이 포함되어 있다. 이 기능들에는 데이터 샘플링 시간 변경 기능, 로그 시작과 정지 기능, 시스템 구성 정보 표시 기능, 성능 감시기 동작 상태 표시 기능이 포함되어 있다.

데이터 샘플링 시간 변경 기능에서는 관리자가 데이터 샘플링 시간을 5초에서 3600초 사이에서 임으로 변경시킬 수 있다. 성능 감시기 하부 구조인 성능 데이터 수집 함수와 상위 구조인 데이터 출력 그래프는 설정된 데이터 샘플링 시간 간격으로 노드들로부터 데이터를 수집하여 계산하거나 출력하기 때문에, 관리자는 데이터 샘플링 시간을 변경함으로써, 성능 감시를 빠르게 혹은 느리게 할 수 있다. 즉, 작은 데이터 샘플링 시간은 관리자로 하여금 짧은 시간 간격으로 성능



(그림 6) 성능 감시기 초기화면
(Fig. 6) The Initial Screen of Performance Monitor

간격을 수행할 수 있게 하고, 큰 샘플링 시간은 큰 시간 간격으로 성능 감시를 할 수 있게 한다. 하지만, 너무 작은 데이터 샘플링 시간은 시스템에 성능 감시기 수행으로 인한 부하를 증가시키므로, 관리자가 디폴트 데이터 샘플링 시간이하로는 변경할 수 없다. 현재 디폴트 데이터 샘플링 시간은 5초로 설정되어 있다.

로그 시작과 정지 기능에서 관리자는 로그 파일을 지정하여, 데이터 로깅을 시작할 수 있다. 관리자가 로그 시작과 정지 기능 버튼을 선택하게 되면, 로그 설정 화면이 생성된다. 로그 설정 화면에서 관리자가 로그 파일을 입력하고 데이터 로그를 시작하면, 성능 감시기 초기 화면 하단의 로그 정보 표시 창에 로그 시작 정보가 표시된다. 그리고, 성능 데이터 수집 함수는 perfdata 자료구조에 저장된 데이터들을 지정된 로그파일에 저장하게 된다.

시스템 구성 정보 출력 기능은 현재 시스템에 구성된 클러스터, 노드들과 각 노드의 CPU, 메모리, 디스크등의 하드웨어 정보를 트리 형태로 출력하여 관리자에게 시스템 구성 정보를 대략적으로 보여준다.

감시기 동작 상태 표시 기능은 그림 5의 우측 하단에 램프로 표시되어 있다. 표시 램프는 5초 간격으로 크기가 변화하므로, 관리자가 데이터 샘플링 값을 크게하여 시스템 성능을 감시하고자 할 때, 성능 감시 기능들이 제대로 동작 중인지를 판단하는데 도움을 준다.

풍선 도움말 표시 기능은 관리자가 성능 감시기 초기 화면의 각 감시 버튼들이 어떤 감시 기능을 수행하는 지를 쉽게 알 수 있도록 한다. 관리자가 풍선 도움말 표시 기능 버튼(그림 6의 왼쪽 하단에 위치)을 선택한 후에, 마우스를 특정 감시 버튼에 위치하게 되면 풍선 도움말이 화면에 나타난다.

성능 감시기 초기 화면에는 관리자가 원하는 감시 화면과 감시 그래프를 찾아주는 기능도 포함되어 있다. 관리자가 많은 수의 성능 감시 기능들을 동시에 수행하게 되면, 감시 화면과 그래프들이 화면 내에서 서로 겹쳐 관리자가 특정 감시 그래프를 쉽게 찾을 수 없게 된다. 이때, 관리자가 초기 화면의 해당 감시 버튼을 선택하게 되면, 선택된 감시 기능 화면과 데이터 출력 그래프가 서로 인접하여 화면 위로 나타나게 된다. 이 기능은 그래프 개수가 많을 때, 매우 유용하다.

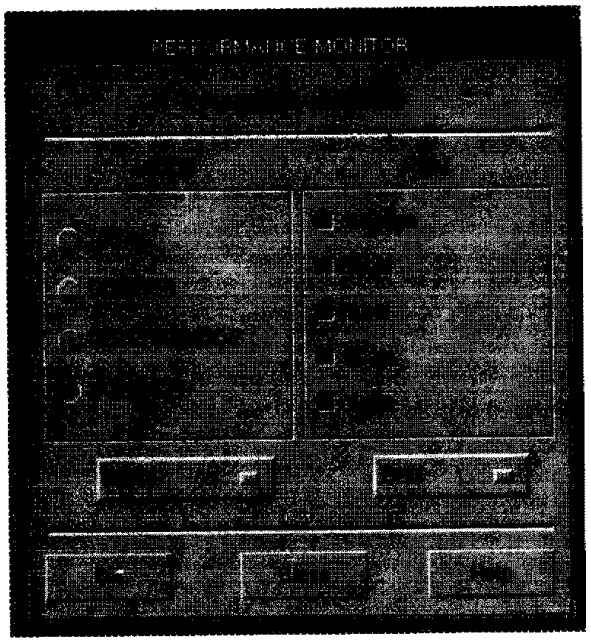
그리고, 모든 성능 감시 화면에는 help 메시지 출력 버튼이 있다. 관리자가 Help 버튼을 선택하게 되면, help 메시지 윈도우가 생성된다. Help 메시지는 윈도우 내에 텍스트 형태로 표시되는데, 관리자에게 각 감시 기능과 감시 항목에 대한 자세한 도움말을 제공한다.

4.2 CPU 사용율 감시 기능

성능 감시기 초기 화면에서 Cpu 감시 버튼을 선택하게 되면 그림 7과 같은 CPU 사용율 감시 화면이 생성된다. CPU 사용율 감시 화면은 크게 감시 모드(MODE)와 감시 항목(ITEM), 그리고 노드나 클러스터 번호를 선택하는 부분들로 구성된다. 감시 모드는 SPAX의 구조적 특징을 반영하여 전체시스템(TOTAL), 클러스터(CLUSTER), 클러스터/노드(CLUSTER/NODE), 노드/CPU(NODE/CPU)모드들로 나누어진다. 감시 항목에는 전체 CPU 사용율(%usage), 시스템 CPU 사용율(%sys), 사용자 CPU 사용율(%usr), 디스크 I/O를 기다리는 시간의 비율(%wio), 어떤 작업도 하지 않고 쉬는 시간의 비율(%idle)들이 있다. 그리고, 노드나 클러스터 번호 선택 부분은 관리자가 특정 감시 모드를 선택할 때 마다 입력 가능한 노드나 클러스터 번호를 보여준다. 예를 들어, 관리자가 클러스터/노드 감시 모드를 선택하면, 그림 7의 왼쪽 옵션 버튼에는 현재 시스템에서 동작중인 클러스터들이 나타나고, 오른쪽 옵션 버튼에는 현재 왼쪽 옵션 버튼에서 선택된 클러스터에 속한 노드들을 보여주게 된다. 또, 관리자가 전체시스템 모드를 선택하게 되면, 노드나 클러스터 번호를 입력할 필요가 없으므로, 왼쪽과 오른쪽 옵션 버튼들은 화면에 표시되지 않는다.

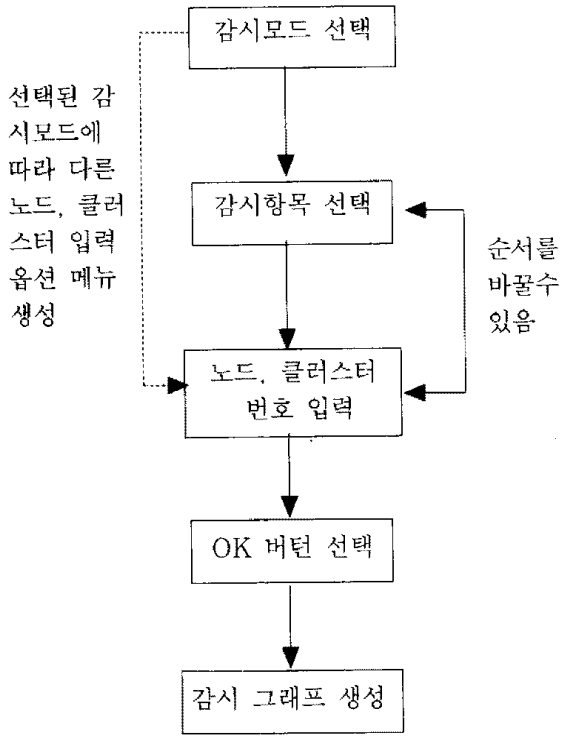
CPU 사용율 감시 기능에서는 시스템 내의 모든 클러스터, 특정 클러스터 내의 모든 노드, 특정 노드 내의 모든 CPU들을 동시에 감시할 수 있는 기능도 제공한다. 예를 들어, 그림 7에서 관리자가 클러스터 감시 모드를 선택하고, 왼쪽 클러스터 옵션 버튼에서 ALL CLUSTERS를 선택하게 되면, 현재 시스템에서 동작중인 클러스터들을 동시에 감시할 수 있다. 이때 오른쪽 옵션 버튼은 화면에 표시되지 않는다.

성능 감시기 내의 대부분의 감시 화면들은 CPU 사용율 감시 화면 처럼 감시 모드, 감시 항목, 노드나 클러스터 번호 입력 부분으로 구성되도록 설계되었고, 그림 8과 같은 순서로 수행된다.



(그림 7) CPU 사용율 감시 화면

(Fig. 7) The Screen of a CPU Usage Monitor



(그림 8) 성능 감시 기능 실행 순서

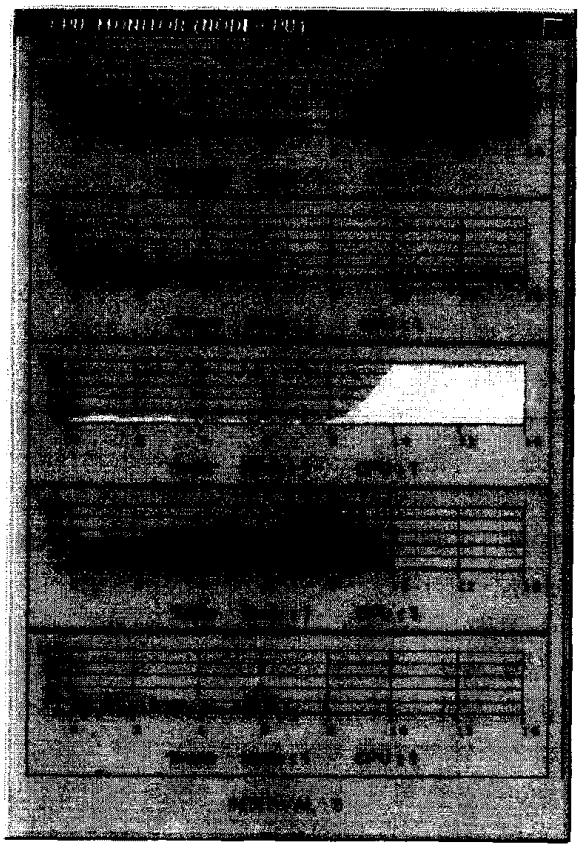
(Fig. 8) The Execution Sequence of Performance Moniroting Function

그림 8에서 처음 감시 화면이 생성되면 관리자는 감시 모드만 선택할 수 있다. 만약 관리자가 특정 감시 모드를 선택하게 되면, 감시 항목을 선택할 수 있고, 현재의 감시 모드에서 선택 가능한 노드나 클러스터 번호 옵션 버턴이 화면에 표시된다.

SPAX는 최대 16개의 클러스터들로 구성될 수 있고, 한 클러스터에는 최대 8개의 노드들로 구성될 수 있다. 만약 그림 7의 클러스터 감시 모드에서 모든 클러스터들을 동시에 감시하고자 할 경우, 모든 감시 항목(5개)을 선택할 수 있게 되면, 최대 80(16x5)개의 그래프가 생성되어, 관리자가 감시하기엔 너무 많게 된다. 구현된 성능 감시기에서는 감시의 복잡성을 줄이기 위해 최대 16개 그래프 이상은 동시에 생성하지 않는다. 따라서, CPU 사용율 감시 기능에서 전체시스템, 클러스터, 클러스터/노드 감시모드와 노드/CPU 감시모드에서 특정 노드 내의 모든 CPU들을 동시에 감시하는 경우에 관리자는 감시항목 중에서 평균 CPU 사용율(%usage)만 선택할 수 있다. 만약 관리자가 노드/CPU 감시모드에서 특정 노드 내의 특정 CPU를 감시하고자 하는 경우에 모든 감시 항목(%usage, %sys, %usr, %wio, %idle)들을 선택할 수 있다. 관리자가 필요한 정보를 모두 입력한 다음, 감시 화면 내의 OK 버턴을 누르게되면 그림 9와 같은 감시 그래프가 생성된다. 만약 필요한 정보가 모두 입력되지 않고, OK 버턴을 누르게 되면 에러메시지가 출력된다.

감시 그래프는 매 샘플링 시간 마다 perfdata 자료구조에서 필요한 데이터를 읽어 그래프로 표시하는 기능외에, 부가적인 기능들이 포함되어 있다. 성능 감시기 내의 모든 감시 기능에서 관리자는 특정 감시 항목들을 감시하는 중간에, 그래프를 새롭게 생성하지 않고도 필요한 항목을 생성된 그래프에 추가하거나 불필요한 항목을 제거할 수 있다. 예를 들어, 관리자가 그림 9와 같은 그래프를 생성한 후에, 그림 7의 감시 화면에서 %idle, %wio 버턴을 언토글(untoggle)하게 되면, 그림 9의 그래프에서 %idle, %wio 항목 그래프들은 사라지게 된다. 만약 다시 %idle, %wio을 선택하게 되면 그래프 내에 %idle, %wio 항목 그래들이 나타나게 된다. 이 기능은 관리자가 특정 자원을 감시하는 중간에 필요한 감시 항목을 추가하거나 불필요한 항목을 제거할 수 있는 장점이 있다. 그리고, 선택된 감시 항목 개수에 따라 그래프 생성 시에 모양이 달라진다. 감시 항목이 6개 이하인 경우에는 하나의 칼럼으로 그

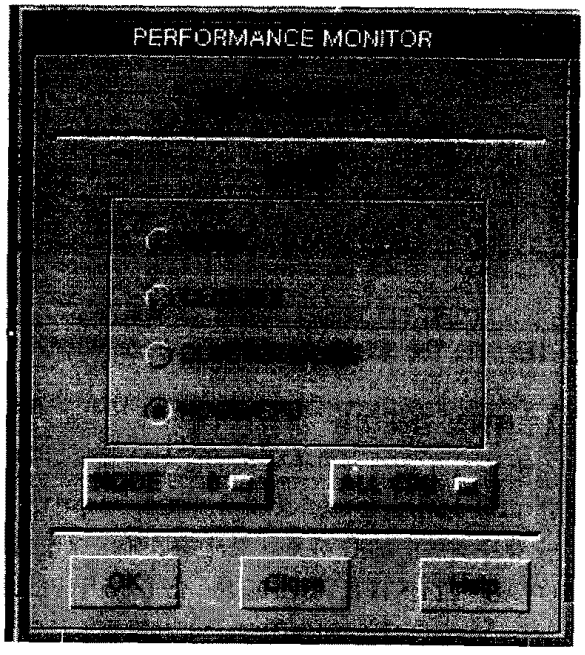
래프가 생성되고(그림 9), 12개 이하이면 두 칼럼으로 생성된다. 구현된 감시기에서는 최대 18개의 감시 항목 그래프들이 생성될 수 있으므로, 최대 3개의 칼럼으로 그래프가 생성될 수 있다. 그리고, 감시 그래프는 그래프 생성 시에 감시 항목 개수에 따라 칼럼당 할당되는 항목 수를 조절하여 밸런스를 맞추어 준다. 예를 들어, 감시 항목이 7개인 경우에 첫 번째 칼럼에 6개, 두 번째 칼럼에 한 개의 항목 그래프가 생성되는 것이 아니라, 첫 번째에 4개, 두 번째는 3개의 항목 그래프가 생성된다. 마지막으로, 감시 그래프는 그래프 속도를 향상시키기 위해서 더블 버퍼링(Double Buffering) 기법을 사용하여 구현하였다.



(그림 9) CPU 사용율 감시 그래프
(Fig. 9) The Graph of a CPU Usage Monitor

4.3 CPU 실행 큐 감시 기능

성능 감시기 초기 화면에서 Queue 감시 버튼을 선택하게 되면 그림 10과 같은 CPU 실행 큐 감시 화면이 생성된다. CPU 실행 큐 감시 화면은 크게 감시 모드(MODE)와 노드나 클러스터 번호를 선택하는 부분으로 구성된다. 감시 모드는 CPU 사용율 감시 화면처럼 전체 시스템(TOTAL), 클러스터(CLUSTER), 클러스터/노드(CLUSTER/NODE), 노드/CPU(NODE/CPU)모드들로 나누어진다. CPU 실행 큐 감시 기능에서의 디폴트 감시 항목은 초당 각 CPU 실행 큐의 길이이다. 즉, 사용자가 특정 감시 모드를 선택하게 되면, 디폴트 감시 항목이 함께 선택되는 것이다. CPU 실행 큐 감시 기능에서는 CPU 사용율 감시 기능처럼 전체 시스템, 시스템 내의 전체 클러스터 혹은 특정 클러스터, 특정 클러스터 내의 전체 노드 혹은 특정 노드, 그리고 특정 노드내의 전체 CPU 혹은 특정 CPU의 초당 CPU 실행 큐 길이를 감시할 수 있다. 관리자가 특정 감시 모드와 노드나 클러스터 번호를 선택한 다음, OK 버튼을 선택하게 되면 그림 9와 같은 감시 그래프가 생성된다.

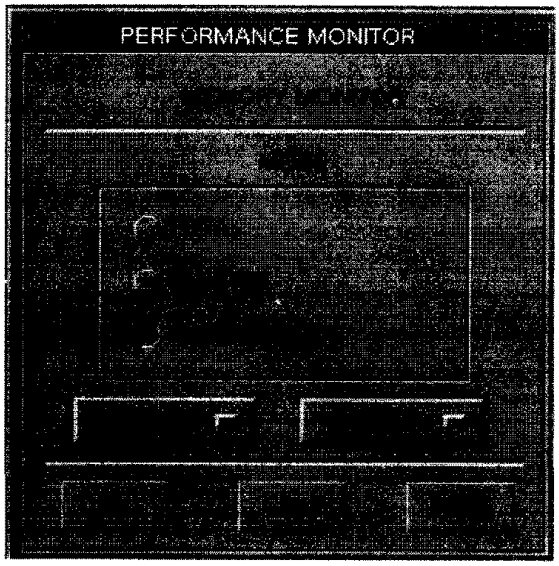


(그림 10) CPU 실행 큐 감시 화면
(Fig. 10) The Screen of a CPU Run Queue Monitor

4.4 메모리 사용율 감시 기능

성능 감시 초기 화면에서 Memory 감시 버튼을 선택하면, 그림 11과 같은 메모리 사용율 감시 화면이 생성된다. 메모리 사용율 감시 화면은 감시 모드(TOTAL, CLUSTER, CLUSTER/NODE)와 노드와 클러스터 번호를 선택하는 부분으로 구성되어 있다. 주전산기 IV에서는 각 노드마다 지역 메모리가 설치되어 있으므로, 메모리 사용율 감시 기능은 전체 시스템, 시

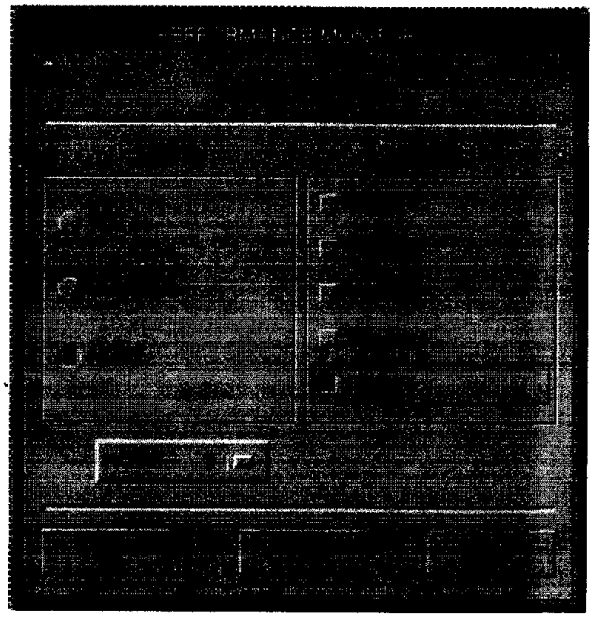
스템 내의 모든 클러스터 혹은 특정 클러스터, 특정 클러스터 내의 모든 노드 혹은 특정 노드의 메모리 사용율을 감시할 수 있도록 설계되었다. 메모리 사용율 감시 기능에서의 디폴트 감시 항목은 샘플링 시점에서의 메모리 사용율이다. 관리자가 특정 감시 모드와 노드나 클러스터 번호를 선택한 다음, OK버턴을 선택하게 되면 그림 8과 같은 감시 그래프가 생성된다.



(그림 11) 메모리 사용율 감시 화면
(Fig. 11) The Screen of a Memory Usage Monitor

4.5 페이징 감시 기능

성능 감시기 초기 화면에서 Paging 감시 버턴을 선택하면, 그림 12와 같은 페이징 감시 화면이 생성된다. 페이징 감시 화면도 CPU 사용율 감시 화면처럼 감시 모드(TOTAL, CLUSTER, NODE)와 감시 항목(Vflat/s, Pfree/s, Vscan/s, Ppgout/s, Ppgin/s), 그리고 노드와 클러스터 번호를 입력하는 부분으로 구성되어 있고, 실행 순서도 CPU 사용율 감시 기능과 동일하다. 다만, 페이징 감시 기능에서는 사용자가 전체시스템, 특정 클러스터, 특정 노드의 초당 주소 변경 페이지 폴트 수(Vflat/s), 초당 커널 리사이클링 정책에 의해 free 리스트에 놓여지는 페이지 수(Pfree/s), 초당 커널 리사이클링 정책에 의해 스캔되는 가상 메모리 페이지 수(Vscan /s), 초당 페이징 아웃되는 페이지 수(Ppgout/s), 그리고 초당 페이징 인되는 페이지 수(Ppgin/s)를 동시에 감시할 수 있다. 관리자가 특정 감시 모드, 감시 항목, 그리고 노드나 클러스터 번호를 선택한 다음, OK버턴을 선택하게 되면 그림 8과 같은 감시 그래프가 생성된다.

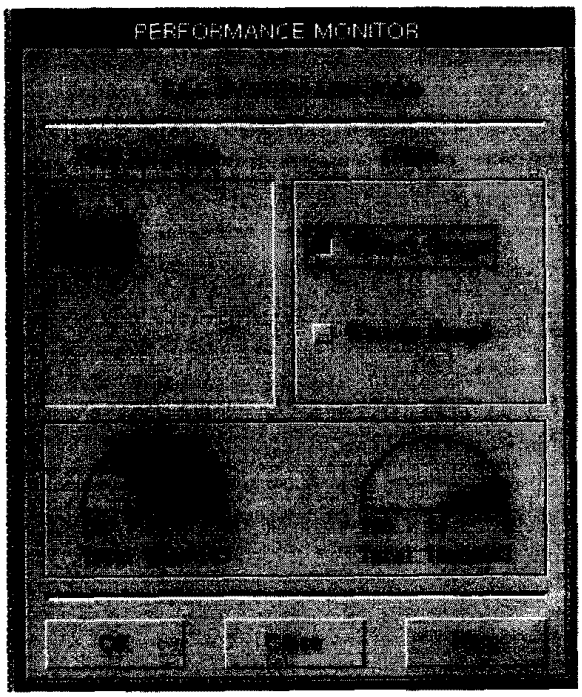


(그림 12) 페이징 감시 화면
(Fig. 12) The Screen of Paging Monitor

4.5 파일 시스템 감시 기능

성능 감시기 초기 화면에서 File 감시 버턴을 선택하게 되면, 그림 13과 같은 파일 시스템 감시 화면이 생성된다. 파일 시스템 감시 화면은 노드나 클러스터 번호를 선택할 필요가 없으므로, 사용자가 선택할 수 있는 파일 시스템들을 표시하는 부분(s5, sfs/ufs, vxfs)과 감시항목(%Block Usage, %Inode Usage, IGet/s)을 나타내는 부분으로 구성된다. 파일 시스템 표시 부분이 그림 8의 감시 모드에 해당되고, 관리자가 특정 파일 시스템을 선택해야만 감시 항목들을 선택할 수 있다. 관리자가 선택할 수 있는 파일 시스템에는 시스템 V 계열 파일 시스템(s5), 유닉스 파일 시스템(ufs)/보안 파일 시스템(sfs), 베리터스 파일 시스템(vxfs) 들이 있다. 관리자가 선택할 수 있는 감시 항목들에는 현재 선택된 파일 시스템의 전체 블록 수에 대한 사용가능한 블록 수의 백분율(%Block Usage), 전체 i-node 수에 대한 사용가능한 i-node 수의 백분율(%Inode Usage), 그리고 초당 i-node 엔트리에 의해서 지정된 전체 파일의 수(IGet/s)가 있다. 파일 시스템 감시 기능도 감시되는 세가지 파일시스템들의 전체 블록과 i-node 사용율 정보를 감시 화면 내에 표시하고 있다. 모든 감시 기능 화면에는 해당 정보의 전체 시스템 데이터들이 표시되도록 설계되었다. 관리자가

특성 파일 시스템을 선택하고, 감시 항목을 선택한 다음, OK 버튼을 누르게 되면, 감시 그래프가 생성된다.



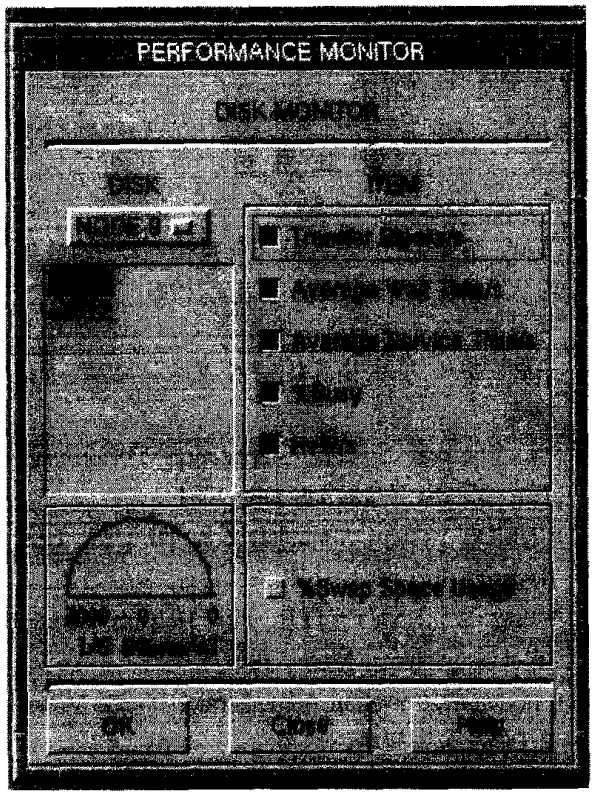
(그림 13) 파일 시스템 감시 화면
(Fig. 13) The Screen of File System Monitor

4.6 디스크 감시 기능

성능 감시기 초기 화면에서 Disk 감시 버튼을 선택하게 되면, 그림 14와 같은 디스크 감시 화면이 생성된다. 디스크 감시 화면은 크게 디스크 트래픽 감시 부분과 디스크 스왑 영역 감시 부분으로 구성되어 있다. 디스크 트래픽 감시 부분은 크게 노드를 선택하는 부분, 특정 노드에 설치된 특정 디스크를 표시하는 부분, 감시 항목(Transfer Block/s, Average Wait Time/s, Average Service Time/s, %Busy, R+W/s) 그리고 특정 노드에 설치된 모든 디스크들의 전송 데이터 크기를 표시하는 부분으로 나누어 진다. 디스크 스왑 영역 감시 부분은 감시 항목(%Swap Space Usage)으로 구성된다.

디스크 트래픽 감시 부분의 수행은 관리자가 특정 노드를 선택하는 것으로 시작된다. 주전산기 IV에서는 디스크가 각 노드에 설치될 수 있기 때문에, 디스크 트래픽 감시 화면에 특정 노드를 선택하는 부분이 포함되어야 한다. 특정 노드가 선택되면, 선택된 노드에 설치된 디스크들이 디스크 표시 부분에 나타나게 된다. 관리자는 특정 디스크를 선택한 후에 디스크 트래픽 감시 항목들을 선택할 수 있다. 디스크 감시 기능에서 관리자는 특정 노드에 설치된 특정 디스크의 초당 읽기와 쓰기 연산의 수(r+w/s), 초당 전송된 디스크 블록들의 수(transfer blocks/s), 서비스를 제공하는 데 사용한 시간의 백분율(%busy), 디스크 연산 요청들이 디스크 큐에서 기다리는 평균 대기 시간(average wait time/s), 그리고, 하나의 디스크 연산 요구가 디스크에 의해서 종료될 때까지의 평균 시간(average service time/s)을 동시에 감시할 수 있다. 관리자가 필요한 정보를 선택한 다음 OK 버튼을 누르게 되면 그림 8과 같은 감시 그래프가 생성된다.

디스크 스왑 영역 감시 부분의 수행은 간단하다. 관리자가 단순히 감시 항목을 선택(또는)하면, 디스크 스왑 영역 감시 그래프가 생성되고, 감시 항목을 언토글하면 감시 그래프는 제거된다. 디스크 트래픽 감시 그래프와 디스크 스왑 영역 감시 그래프는 독자적으로 생성된다.



(그림 14) 디스크 감시 화면
(Fig. 14) The Screen of Disk Monitor

5. 결 론

유닉스 시스템의 사용이 보편화되면서 유닉스 시스템에 대한 성능 관리 또한 시스템 관리자의 중요한 임무가 되었다. 성능 관리는 성능 감시, 성능 분석, 그리고 성능 튜닝의 작업을 반복함으로써 수행되는데, 성능 감시에서는 시스템에 적절한 성능 감시기를 선택하여 시스템 상태를 감시하게 된다. 성능 감시기는 시스템 자원들의 현 상태를 감시하기 위한 다수의 기능들을 가지고 있어서, 시스템 관리자가 시스템의 병목을 쉽고 편리하게 찾아내어 성능을 향상시킬 수 있는 방법을 결정하는데 도움을 준다.

주전산기 IV로 개발된 SPAX는 이전의 병렬 컴퓨터와는 다른 구조적 특징을 가지고 있기 때문에 SPAX의 구조적 특징을 반영한 새로운 성능 감시기가 요구되었다. 현재 구현된 성능 감시기는 SPAX의 노드나 클러스터 단위의 성능 감시가 가능하고, 관리자가 쉽게 사용할 수 있도록 모든 감시 기능 화면은 동일한 방식으로 설계되었다.

성능 감시기도 시스템에 부하를 줄 수 있으므로 시스템에 최소한의 부하를 주면서 성능 감시가 가능한 최소 데이터 샘플링 시간이 중요하다. 앞으로, 구현된 성능 감시기에 맞는 최소 데이터 샘플링 시간을 SPAX에서 구해야 할 것이다.

참 고 문 헌

[1] "고속 병렬컴퓨터(SPAX) 하드웨어 서브시스템 설계서", 한국전자통신연구소, Apr. 1995

[2] 박성훈, "고속 병렬컴퓨터 성능 감시기의 설계 및 구현", 한국 정보 과학회 춘계 학술 발표회, June 1996.

[3] "X Window System Application Performance Tuning", the X journal, july, 1994.

[4] UNICOS System Administration TR-USA, Cray Research Inc.

[5] Using CA-UNICENTER, For Hewlett-Packard HP-UX Systems, March 1993.

[6] Unixware 2.0 System Administrators Guide Manual Page, Novell Co.

[7] 정성인, "고속병렬컴퓨터의 메시지 전송 구조 및 성능 평가", 한국 정보 과학회 추계 학술 발표회, Oct. 1996.

[8] 윈도 95 사용자 매뉴얼.

[9] 유웅준, "유닉스시스템의 성능과 튜닝(Tuning) (1)", 컴퓨터와 커뮤니케이션, pp.198-205, Dec., 1994.

[10] 유웅준, "유닉스시스템의 성능과 튜닝(Tuning) (2)", 컴퓨터와 커뮤니케이션, pp.202-207, Jan. 1995.

[11] 유웅준, "유닉스시스템의 성능과 튜닝(Tuning) (3) 성능 저하를 초래하는 병목현상 진단방법", 컴퓨터와 커뮤니케이션, pp.207-215, Feb. 1995.

[12] "운영체제 서브시스템 설계서" 한국전자통신연구소, May, 1995.

[13] "고속 병렬 컴퓨터(주전산기 IV)의 시스템 관리 소프트웨어 개발에 관한 연구(III)", 시스템공학 연구소, Jan., 1997.

[14] Kai Hwang, 'Advanced Computer Architecture', McGraw-Hill, 1993.

[15] 이 경석, 우영춘, 김진미, 지동해, "ParaC 언어의 설계 및 구현", 정보처리논문지, 제4권, 제11호, pp.2903-2913, 1997.

김 도 형

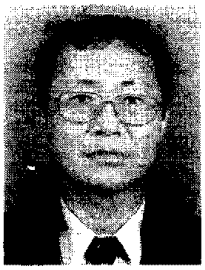
1993년 경북대학교 컴퓨터공학과 (공학사)

1995년 포항공과대학 전자계산학과(이학석사)

1995년~1997년 시스템공학연구소 연구원

1998년~현재 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 연구원

관심분야 : 결함포용, 병렬처리, 성능평가



김 채 규

1978년 고려대학교 수학과(이학사)

1993년 호주 Univ. of Technology Sydney 전산학(석사)

1997년 호주 Univ. of Wollongong 전산학(박사)

1977년~1990년 KIST 전산개발센터 연구원, 시스템공학연구소 책임연구원

1998년~현재 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소 책임연구원

관심분야 : 데이터베이스, 병렬처리