HP GlancePlus/UX User's Manual

HP Part No. B2660-90002
Printed in U.S.A.
Translated in Korea (ESSO)
E1192

Modify by SungYoon Jang
yunisj@korea.com

Contents

1st Introduction

HP GlancePlus/UX 는 시스템 activity 를 모니터하는 online 상의 diagnostic tool 이다. GlancePlus screen 은 다른 자원들이 어떻게 사용되고 있는지를 보여주는데, 그것들은 시스템 activity 를 요약하거나 특정 자원을 집중으로 보여주고, 또 개개의 프로세스가 시스템 자원을 어떻게 이용하는지를 보여준다.

A) GlancePlus/UX 가 제공하는 것을 보면:

1. 즉각적인 performance problem 을 해결하기 위한 diagnostic tool

누가 혹은 무엇이 그 시스템을 느리게 하고 있는지 쉽게 찾을 수 있게 한다. 또한 CPU 나 I/O 를 과도하게 사용하는 특정 program 이나 application 을 구분할 수 있게 하며, 최적의 application 에 맞는 입력치(input)를 제공할 수 있다.

2. 시스템을 관찰할 수 있는 모니터링 tool

시스템 activity 를 정기적으로 체크하므로써, 전형적인 performance level 을 문서화 할 수 있고 전형적인 시스템 load 를 특성화 시킬 수 있다. 이것은 performance problem 을 구분하는 것을 용이하게 한다.

3. 사용 용이 (Easy of Use)

GlancePlus/UX 는 설치(install) 및 구동(run), 그리고 사용(use)하기가 쉽다. 이것은 복잡한 set up 절차에 시간을 투자할 필요가 없고, 배우는데 많은 시간이 필요하지 않다는 것을 의미한다. 장비의 performance 에 관련된 모든 사람은 사용하는 첫날부터 HP GlancePlus/UX 를 효율적으로 사용할 수 있다.

4. 유연성 (Flexibility)

사용자와 시스템에 가장 중요한 데이터를 보여주고, 중점화 시킬 수 있는 software 를 만든다. 예를 들면, 쉽게 threshold 값을 정의하여, 그 한계치를 초과하는 프로세스들만을 activity screen 에 보여지게 할 수 있다. 나중에 그 정보를 검토하기를 원한다면, paper copy 를 얻기위한 screen 을 인쇄할 수 있다.

5. Selective levels of data

전체 시스템의 포괄적인 전시 부터, 특정 프로세스의 특정 자원에 대한 기술적인 데이터를 전시하는 것까지 다양하다. 이것은 너무 많은 데이터나 너무 적은 데이터에 의해 당황스러울 수 있는 경우를 방지한다.

6. 포괄적인 Online 상의 도움 (Extensive online help)

performance 전문가들에 의해 개발된 online help 장치들은 바로 사용자의 손가락 끝에 screens 과 mesuremnets 에 대한 정보를 보여준다.

이러한 모든 특성들은 시스템의 activity 를 모니터하고, 자원을 proactive 하게 관리하는데 도움을 줄 수 있는 정보를 제공한다.

다양한 시스템 상황을 진단하는데 사용될 수 있는 HP GlancePlus/UX 를 어떻게 screen 할 것인지에 대한 예를 보기위해서는, 4 장(Chapter 4)를 참조하면 된다.

B) How to Use This Manual

이 manual 을 여러분이 Glance 로부터 최대 가치를 얻을 수 있도록 도울 것이다.
여러분의 필요와 이전의 경험에 비추어, 여러분은 특정 정보를 얻기 위해서 아래 chapter 들을 참조하면 된다.

Chapter 2 HP GlancePlus/UX 를 install 하는 방법

Chapter 3	**그 프로그램에 대한 일반적 특성에 대한 정보**
Chapter 4	**시스템 activity 를 모니터하고, 가능성 있는 문제를 진단하기 위해 어떻게 GlancPlus 를 사용할 것인지에 대한 예들.**
Chapter 5	**각각의 GlancePlus screen 에 나타나는 통계치(statistics)에 대한 설명**
Chapter 6	**startup 절차를 만들기 위한 지시 사항들.**
Messages	**prompts 와 message 에 대한 정보를 포함**
Glossary	**특정 용어에 대한 정의를 포함**

HP-UX 시스템과 GlancePlus 에 대한 추가적인 online 정보는 *HP-UX Reference* manual 을 참조하면 된다. 이 manual 은 online man page 와 특정 HP-UX 에 관련된 topic 을 정의하는 것들로 구성되어 있다.

GlancePlus 의 **man page** 는 자동적으로 그 product 와 함께 설치 된다. GlancePlus man page 나 다른 HP-UX product 를 위한 man page 들은 아래를 타이핑 하므로써 HP-UX shell command line 으로 부터 접근 될 수 있다.

man *topic*

online 정보가 이용가능하면, 그것은 screen 에 보여질 것이다. Online manual 로부터의 정보가 이 manual 에 참조될때 이것은 *"See man page...."*로 참조될 것이다.

Online HP-UX manual 에 추가하여, Chapter 3 에서 논의될 GlancePlus 는 online help 정보를 제공한다. 그 online help 정보는 help key 인 h command 를 사용함으로써 access 될 수 있다.

2nd Installation

HP GlancePlus/UX 는 HP-UX Measurement Interface(MI)에 바탕을 둔다. 이 Interface 는 **midaemon** 이라 불리는 프로세스를 사용하면서 지속적으로 performance 정보를 모은다. **midaemon** 은 GlancePlus 프로그램에 의해 읽혀지고 GlancePlus screen 에 나타나는 통계치로 번역되는, shared-memory segment 로 이 정보를 기술한다.

이런 동일한 shared-memory segment 는 HP LaserRX/UX performance product 의 **scopeux** collector 에 의해서 사용된다. 시스템이 이미 HP LaserRX 를 가지고 있다면 **midaemon** 은 install 된 것이다. GlancePlus tape 은 또한 **midaemon** 의 version 을 install 할 것이지만, 그것은 완벽하게 HP LaserRX 와 양립할 것이다. 만약 HP LaserRX 를 가지고 있지 않다면, **midaemon** 은 GlancePlus 가 install 될 때 자동적으로 install 될것이다.

HP 9000 에 GlancePlus 를 install 하기 위해 **update** 라 불리는 HP-UX program 을 사용할 것이다. 이 program 에 대한 정보를 얻기 위해 HP-UX Reference manual 을 참조하면 된다.

A) Installing GlancePlus

HP 9000 을 위한 GlancePlus 는 GLANCE 라 불리는 하나의 file set 에 있는 installation media 에 포함되어 있다. 그 file set 의 크기는 대략 1 MB 정도이다. Measurement Interface 는 MI 라 불리는 두 번째 fileset 에 있다.

아래 설명된 절차에 따라서 GlancePlus 를 install 하고 **update** program 을 구동 시키는데 10 분 이하의 시간이 걸린다. update 를 구동시킨 후에 installation 이 성공적으로 되었는지를 확인하기 위해 update log **/tmp/update.log** 의 마지막 부분을 검토해야 한다.

GlancePlus 를 install 시키기 위한 절차는 아래와 같다:

1. login 한다: root

2. 사용하는 media 에 따라 다음중 하나를 실행
   a. 만약 1/4-inch tape cartridge 를 사용한다면:
      i. protect switch 가 SAFE 나 PROTECT 를 지시하는지를 확인
      ii. GlancePlus tape 을 넣은 후
      iii. Busy light 가 깜박이는 것을 멈춘후에 3 번째 단계로 이동
   b. 만약 1/2-inch magnetic tpae (1600 bpi)를 사용한다면:
      i. write-enable ring 이 제거 되었는지를 확인하고
      ii. GlancePlus tape 을 마운트 한후
      iii. line 에 tape device 를 놓고서 (ONLINE light 은 켜진 상태) 3 번째 단계로 이동
   c. 만약 DAT/DDS tape 을 사용한다면:
      i. GlancePlus tape 을 마운트
      ii. tape 이 load 된 후 3 단계로 이동
   d. 만약 CD-ROM 을 사용한다면
      i. CD-ROM drive 를 위한 mount directory 를 만들고, 그 directory 에 모든 사용자를 위한 read/write/execute permission 을 준다.
      예를 들면:
      ```
      mkdir  /cdrom
      chmod  777  /cdrom
      ```
      ii. file system 을 마운트
      예를 들어 300 시리즈에서는
      ```
      /etc/mount  /dev/dsk/1s0  /cdrom
      ```
      또 800 시리즈에서는
      ```
      /etc/mount  /dev/dsk/c9d0s2  /cdrom  -r  0t  cdfs
      ```

3. cd / 를 타이핑 하고, Return 키를 친다.

4. /etc/update 를 타이핑 하고, Return 키를 친다.

5. 만약 destination 이나 source 가 주 메뉴에 있는 default 와 다르면, **Change Source or Destination** 을 선택한다. Installation 에 대한 더 많은 정보를 원하면, document *Installing and Updation HP-UX* 를 참조하면 된다.

update 의 default destination 은 / (root)이다. Default source 는 update media 와 시스템의 타입에 따라 다르다.

a. 300 과 700 시리즈의 default source device 는 /dev/update.src 이다.
b. 800 시리즈는 /dev/rmt/0m 이다.

6. **Select All Filesets on the Source Media →** 를 선택한 후 Return 키를 친다.

7. 만약 모든 것이 정확하다면, **Start Loading Now….**를 선택한다.

8. prompt 에서 Return 키를 친다.

9. 다음 installation 을 중지할 것인지의 메시지가 나올 것이다. 메시지 후 prompt 에 응답한다.

Continue? (y or n)

y 를 선택하면, GlancePlus file set 이 load 될 것이고, n 를 선택하면 update 프로그램이 중지 될 것이고, installation 이 멈출 것이다.

10. 성공적으로 installation 이 되면, 다음의 메시지를 볼 것이다.

Review the log file, /tmp/update.log, for possible errors, warnings, and notes.

11. 이것은 **update** 가 완결 되었다느 것을 의미한다.

Update 프로세스는 /tmp/update.log 라는 log file 의 끝에 정보를 덧 붙인다. 이 file 은 GlancePlus installation 에 대한 시스템 메시지를 포함하고, 시스템에 나타나는 문제들을 기록한다. Install 시 쌓인 error message 를 보고 적당한 조치를 취하고 싶다면 이 file 을 참조하면 된다.

이 file 이 마지막 30 줄을 보고 싶다면 아래의 명령을 치면 된다:

tail -30 /tmp/update.log

3rd Getting Started

이 Chapter 는 HP GlancePlus/UX 프로그램을 어떻게 쉽게 이용할 수 있는 지를 보여준다. 프로그램을 시작하고, 스크린을 보고, 다른 스크린으로 이동하고, 또 프로그램을 끝낼 수 있는 방법을 알것이다. 또한 다양한 기능을 쉽게 수행 할 수 있는 function key(softkey)와 single-letter command 에 대한 설명도 있다.

A) Using HP GlancePlus/UX

Glance 를 시작하기 위해서, **glance** 를 타이핑 한다. GlancePlus 의 directory 구조는 **/usr/perf/bin/glance** 이다.
GlancePlus 가 **/usr/perf/bin** directory 에 있는 **glance** binary file 을 초기화하는데는 약 15 초 정도가 소요된다. 더 많이 load 된 시스템에서는 시간이 더 걸린다.
Glance 에 더 친숙하게 되면, 여러분은 GlancePlus startup 절차를 만들기를(customizing) 원할 것이다. Chapter 6 는 GlancePlus startup 절차를 만들고, **midadmon** startup 프로세스를 자동화 시키는 정보를 포함한다.

언제든지 그 프로그램을 빠져나오기 위해서는, 아래사항중 하나를 실행하면 된다.

- Online help screen 을 제외한 어떤 스크린에서든지 [e] 나 [q] key 를 친다.
- 만약 softkey 가 스크린에서 이용가능 한다면, Exit Glance key 를 친다.

Global screen (figure 3-1)은 Glance 를 시작했을 때 나타나는 첫번째 스크린이다. 이 스크린은 자원들이 active 한 프로세스에 의해서 어떻게 사용되고 있는지의 전체적이고 시스템상 포괄적인 요점사항을 제공한다.

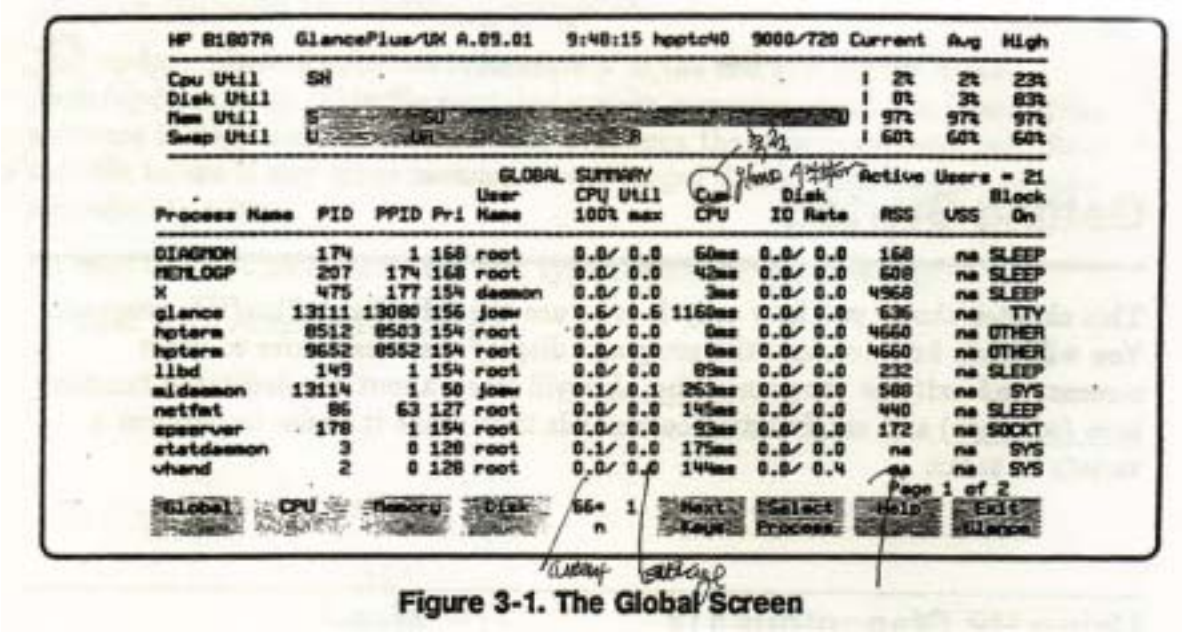


Figure 3-1. The Global Screen

다른 GlancePlus screen 에 접속하기 위해서 Global screen 의 바닥에 있는 function key 나 single-letter command 를 사용할 수 있다. 이러한 function key 나 single-letter command 는 나중에 소개된다.

B) Reading the Screen Display

모든 GlancePlus screen 은 비록 시스템 자원들에 대한 다른 정보를 보여주더라도 비슷하게 나타난다. 각 스크린은 4 개의 섹션, 즉 Banner Line, Global Bars, Elements Display, 그리고 Function Keys (Softkeys)로 나뉘어 진다. Banner Line 과 Global Bars 는 모든 스크린에서 같은 정보를 나타내고, Elements Display 섹션은 스크린마다 다른 정보를 보여준다. Function key 의 4 가지 set 은 각 스크린에서 이용가능하다.

1. Banner Line

스크린의 위에 있는 Banner Line 은 product 의 version, product 이름, 현재 시간, 시스템 명칭, 그리고 시스템 타입을 포함한다. (Figure 3-2) 그 줄은 세가지 colume heading 으로 끝나는데 – Current, Avg (for average), High –Banner Line 아래에 있는 Global Bar 섹션에서 나타나는 통계치를

지칭한다.

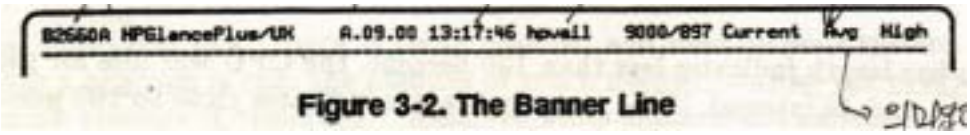


Figure 3-2. The Banner Line

2. Global Bars

4 개의 Global Bar 는 Banner Line 바로 밑에 나타난다. (Figure 3-3) 이런 bar 들과 오른쪽에 있는 percentage 는 네가지 자원 – CPU, Disk, Memory, Swap Space – 의 사용율을 나타낸다. 각 스크린에 있는 이러한 bar 들은 "global" 상태를 계속적으로 기억하는데 도움이 된다.

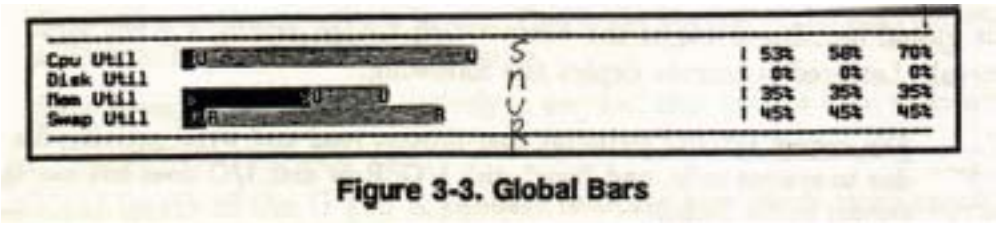


Figure 3-3. Global Bars

각각의 bar 에 포개진 문자들은 자원의 그 부분이 *어떻게* 사용되고 있는지를 보여준다.

① CPU Utilization Bar

이 global bar 는 마지막 update interval 동안 다양한 activity 에 얼마나 많은 CPU time 이 쓰였는지를 보여준다. Bar 에 있는 문자들은 아래사항을 의미한다.

S 시스템 call 의 수행, interrupt 의 handling, 그리고 context switching 과 같은 system activity 에 소비된 시간

R "real-time" priority 와 함께 구동되는 사용자 프로세스에 소요되는 시간. Real-time priority 는 일반적인 time-sharing 프로세스보다도 더 높은 우선순위에서 구동되는 프로세스를 지칭한다. (see man-page *riprio(1)*)

U user code 를 수행하는 것과 같은 user activity 에 소요되는 시간. 이것은 nice priority 와 함께 구동되는 user code 를 포함하지 않는다.

N nice 와 negative nice priority 와 함께 구동되는 user code 에 소요되는 시간. Nice priority 는 다른 프로세스보다도 더 낮은 우선순위에서 프로세스가 수행된다는 것을 의미하고, negative nice priority 는 더 높은 우선순위에서 수행하도록 하는 것이다.

만약 bar 의 길이가 100%보다 짧으면, CPU 가 한가하다는 것을 의미하고; 만약 계속해서 100%의 CPU 사용율에 근접하면 performance 상 bottleneck 에 걸릴 수도 있다는 것을 의미한다. bottleneck 에 대해서는 Chapter 4 를 참조하면 된다.

하나이상의 CPU 를 가진 시스템에서 CPU 사용율은 100% 기준으로 되어 있다. 예를들어, 시스템이 4 개의 CPU 를 가지고 있고, 그것들 중 2 개가 100% 사용된다면, global utilization 은 200%가 아니라 50%이다.

② Disk Utilization Bar

이 global bar 는 주어진 time interval 동안에 가장 바쁜(busiest) disk device 의 사용율을 보여준다. 아래에 각 문자화된 segment 가 있다:

F user-process read 와 write activity, 시스템 call 에 의한 file system I/O, 그리고 "raw" disk I/O 를 포함한 file system activity. Raw disk I/O 는 시스템 buffer cache 를 사용하지 않는다.

V paging data 에 의한 vitual memory 로의, 그리고 로부터의 disk I/O.

만약 bar 길이가 100%에 가까우면, 시스템의 가장 바쁜(busiest) disk device 가 그 queue 에 I/O pending 을 가지고 있다는 것을 의미한다. 이것은 disk I/O bottleneck 상황일 수도 있다.

③ Memory Utilization Bar

이 global bar 는 physical memory 의 사용을 보여준다. 아래 설명을 참조:

S 시스템 code 과 데이터용으로 사용된 physical memory

U 사용자 code 과 데이터용으로 사용된 physical memory

만약 bar 길이가 100%에 근접한다면, 대부분의 physical memory 가 code 와 data object 를 저장하기 위해서 이용되고 있다는 것이다. 이것은 memory bottleneck 을 유발할 수 있는 memory pressure 을 의미할 수 있다.

④ Swap Utilization Bar

이 bar 는 시스템에서 어떻게 swap space 가 reserved 되고 사용되는 지를 보여준다. 프로그램이 구동될 때 마다 주 memory 밖으로 프로그램을 바꿔주기(swap) 위해 필요되어지는 space 가 "reserved" 된다. Reserved swap space 는 어떤 disk location 에 할당되지 않고, 단순히 필요되어 질 수 있는 swap space 의 양을 지칭한다. 프로그램이 실제로 swap out 될때 disk space 가 할당되는데; 이것은 reserved space 의 일부분이 된다.

이 값은 overhead 를 줄이기 위해 매 30 초마다 update 된다. 아래의 설명을 참조:

U 실제적으로 사용되고 있는 reserved swap space. 이것은 실제적으로 쓰여진(written) reserved swap space 의 일부분이다.

R reserved 되어 있지만 실제 active 하게 사용되고 있지 않은 swap space; 즉 이것은 쓰지지 않았다. (not written to)

U 와 R bar 의 통합된 길이는 얼마나 많은 swap space 가 reserved 되어 있는지를 나타낸다. 만약 bar 길이가 100%에 다달으면, 시스템의 free swap space 가 바닥나고 프로세스가 수행을 멈출수도 있다는 것을 의미한다.

Swap space 상태에 대한 더 많은 정보를 위해서 "Swap Detail Screen"을 참조하면 된다.

⑤ Percentage Columns

Global bar(Figure 3-3)의 오른쪽에 있는 percentage column 은 각 시스템 자원의 현재(current), 평균(average), 최고(highest)의 사용율을 보여준다. Current value 는 current interval 의 사용율이고, average 는 현 모니터링 세션이 시작한 후 모아진 usuage data 을 값을 의미한다. High value 는 모니터링 동안 자원의 최고 사용율을 의미한다.

만약 Zero command (z)를 이용하여 통계치를 reset 시켰다면 current value 는 현 사용율을 나타내지만, average 와 high value 는 resetting 이후의 값을 표시한다.

Note 시스템을 모니터하기 위해 GlancePlus command 를 입력할 때, Global screen 상의 시스템 자원의 주요 사용자로서 GlancePlus 을 관찰 할 것이다. 이것은 잘못된 방향으로 갈 수도 있다. 스크린을 refresh 하고 update 하기 위해 여러분이 입력한 keystroke 은 Glance 에 의해서 CPU usuage 를 생성한다. 스크린을 더 자주 바꾸면 바꿀수록 CPU usuage 는 점점 더 커진다.

GlancePlus 의 평균적 usuage 는 어떤 문제도 일으키지 않는다. overhead 에 대한 더많은 정보는 Chapter 4 에 있다.

3. Detail Display

스크린 타입에 따라서 다른 정보들이 global bar 아래에 나타난다.(Figure 3-4) Global 스크린은 프로세스들에 대한 요약을, CPU Detail 스크린과 같은 다른 스크린들은 특정 자원에 대한 통계치를 보여준다. Individual Process 스크린은 하나의 선택된 프로세스에 대한 많은 통계치를 보여주는데, 두개의 스크린, 즉 Commands Menu 와 Interesting Process Threshold Options 스크린은, 기능을 선택하고, GlancePlus program 의 parameter 를 세팅하는데 도움을 준다

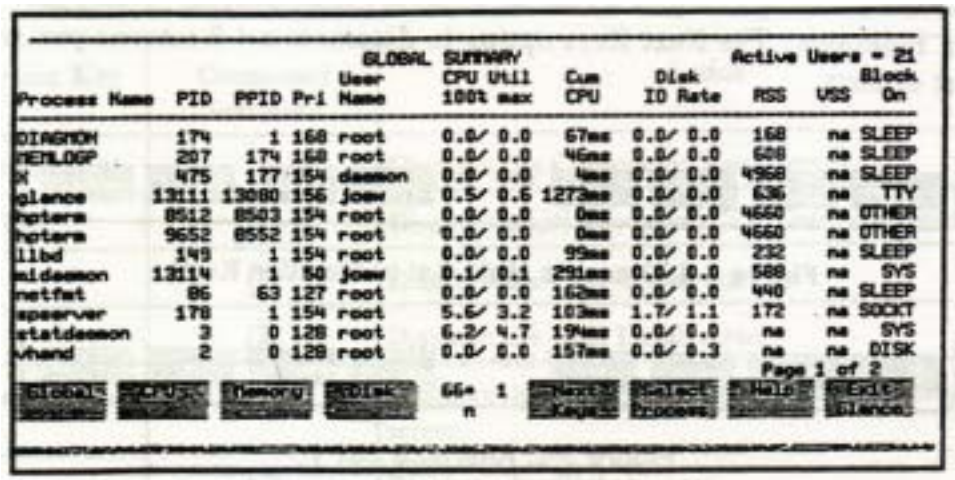


Figure 3-4. Global Summary Display

많은 시스템에서, 현재 구동중인 프로세스를 보여주는 것은 꽉찬 데이터를 가진 스크린을 취한다는 것을 의미한다. 많은 정보를 평가한다는 것은 복잡할 수도 있다. 이것을 피하기 위해서 GlancePlus 는 **interesting** 프로세스 – 미리 결정하는 threshold usage value 를 surpass 한 것 -- 로 보여지는 프로세스들에 한계치를 준다. 이러한 interesting 프로세스들은 시스템 자원들의 많은 부분을 사용하고 가능성있는 문제점을 암시한다. 프로세스 threshold 값을 세팅하는 절차는 이 장의 마지막 부분에 있다.

Global 스크린은 interesting 프로세스를 보여주는 유일한 스크린이다. 다른 GlancePlus 스크린들은 선택된 자원과 프로세스들만을 보여준다. 이러한 스크린들에 대한 설명은 5 장에서 소개된다.

4. Function Keys and Commands

스크린이나 기능(function)을 쉽게 선택하기 위해서, GlancePlus 는 각 스크린의 바닥에 한줄의 function-key label 을 제공한다.(Figure 3-5 to 3-8) **softkeys** 라고도 불리는 이러한 label 들은 F1 부터 F8 까지의 key 와 상응된다.

GlancePlus 는 네가지 셑의 **softkey** 를 가진다. GlancePlus 가 초기화 될 때 나타나는 Main set 외에도, 스크린을 활성화시키는 기능들을 나타내는 세개의 Alternate set 이 있다. 다른 set 을 보기 위해서는 Next Keys 를 치면 된다.

이렇게 활성화된 softkey 들은 다른 스크린으로 쉽게 이동할 수 있게 하고 중요한 프로그램 기능들을 수행할 수 있게 한다. Alternate set 3 에 있는 Next Key 는 Main set 으로 돌아오게 한다.

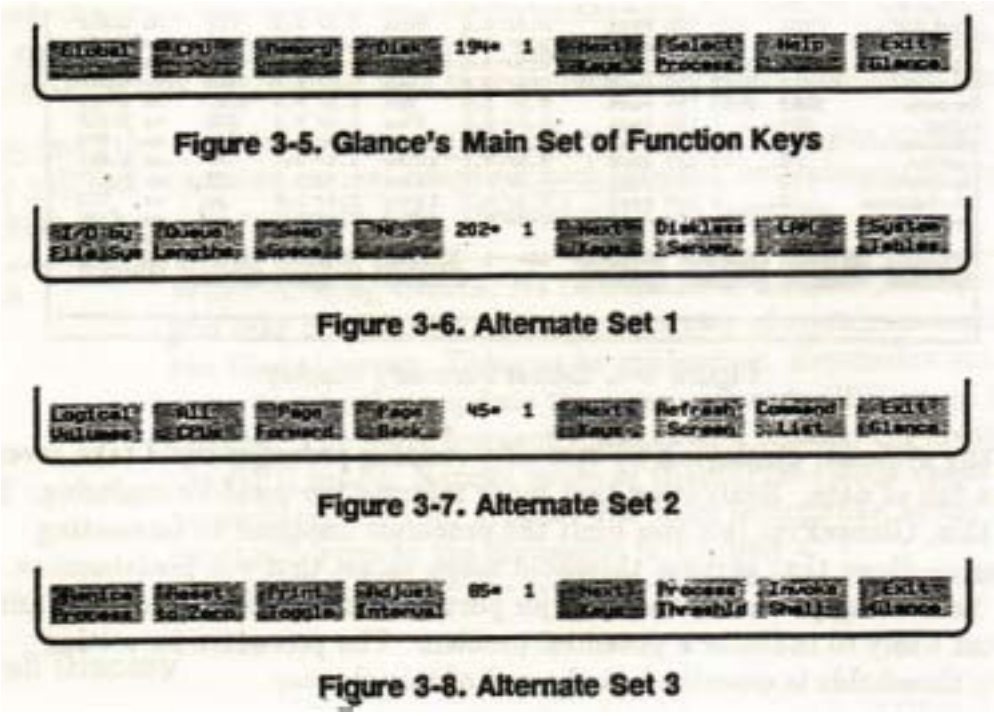


Figure 3-5. Glance's Main Set of Function Keys

Figure 3-6. Alternate Set 1

Figure 3-7. Alternate Set 2

Figure 3-8. Alternate Set 3

또한 single-letter command 를 이용하여 Glance 의 기능들을 수행할 수 있다. 각 fuction key 는 한개 이상의 상응하는 single-keystroke 을 가진다. 각 function key 와 command 는 table 3-1 과 3-2 에 나와 있다.

Table 3-1. Function Keys and Commands to Access Screens

Function Key	Command Letter(s)	Action
Command List	?	Command Menu screen 을 보여준다.
All CPUs	a	Multi-Processor CPU Details screen 을보여준다.
CPU	c	CPU Detail screen 을 보여준다.
Disk	d	Disk Detail screen 을 보여준다.
Diskless Server	k	Diskless Server Resource Utilization 을 보여준다.
Global	g	Global screen 을 보여준다.
I/O by File Sys	i	File System screen 에 의한 Disk I/O 를 보여준다.
LAN	l	LAN Detail screen 을 보여준다.
Logical Volumes	v	Logical Volumes screen 을 보여준다.
Memory	m	Memory Detail screen 을 보여준다.
NFS	n	NFS Detail screen 을 보여준다.
Process Threshold	o	Interesting Process Threshold Options screen 을 보여준다.
Queue Lengths	u	Disk Queue Lengths screen 을 보여준다.
Renice Process	y	프로세스의 "nice" value 를 바꿔준다.
Select Process	s	Individual Process screen 의 detail 에 보여지기 위해 특정 process 를 선택
Swap Space	w	Swap Space Utilization screen 을 보여준다.
System Tables	t	System Table Utilization screen 을 보여준다.

Table 3-2 Function Keys and Commands to Access Program Activities

Function Key	Command Letter(s)	Action
Next Screen	>	Next "logical" screen 을 보여준다. [Global 이나 CPU screen 으로 부터: 가장 높은 CPU consumer 에 대한 Individual Process screen; Disk Detail screen 으로 부터: Filesystem screen 에 의한 Disk I/O; Filesystem 에 의한 Disk I/O 나 Disk Queue Lengths screen 으로 부터: 가장 높은 disk user 에 대한 Individual Process screen; 그리고 Memory screen 으로 부터: System Tables screen.]
Previous Screen	<	이전 화면으로 돌아간다.
Adjust Interval	J	2 ~ 32,000 초 사이의 기간에 data update 를 하는 사이 time interval 을 resetting 하는 것을 허락한다. Default 는 5 초이다.
Exit Glance	e 또는 q	HP GlancePlus/UX 를 빠져 나간다.
Help	H	Online help 에 대해 하나 이상의 page 를 보여준다.
Invoke Shell	!	사용자의 default shell 를 invoke 한다.
Next Keys		Function key (softkey)의 Main, Alternate1, 2, 3 셑을 활성화 시킨다.
Page Back	b 또는 -	그 스크린의 이전 page 를 보여준다. 만약 한 page 만 있다면, 이 key 는 스크린을 refresh 시킨다.
Page Forward	f 또는 +또는 Space Bar	그 스크린의 다음 page 를 보여준다. . 만약 한 page 만 있다면, 이 key 는 스크린을 refresh 시킨다
Print Toggle	P	Update interval 후에 보여진 스크린이나 현재의 스크린을 인쇄하기 위해 GlancePlus 를 지시하거나 프린트를 활성화 시키기 위한 Toggle. 이 key 를 toggling off 하므로써 지속되는 printing 을 멈출 수 있다. Note: Continuous printing 은 next update 와 함께 시작될 것이다; 현재의 스크린은 single-screen option 이 선택되지 않는한 프린트 되지 않을 것이다.

Table 3-3. Function Keys and Commands to Access Program Activities

Function Key	Command Letter(s)	Action
Refresh Screen	r 또는 Ctrl – L	보여지는 정보를 update 하지 않고, 스크린 보기를 새롭게 한다.
Reset to Zero	Z	모든 cumulative 한 통계치의 데이터를 0.0 으로 바꾸고 새로운 데이터를 accumulate 한다.
Not Applicable	Return	스크린에 보여지는 정보를 update 한다.

Table 3-4. Commands from the Single Process Detail Screen

Function Key	Command Letter(s)	Action
Resource Summary	S	Resource Summary 정보로 되돌린다.
Wait States	W	Wait States 정보를 보여준다.
Memory Regions	M	Memory Regions 정보를 보여준다.
Open Files	F	Open File 정보를 보여준다.

C) Commands Menu Screen

이 스크린은 스크린에 접속하거나 program function 을 시작하기 위해 사용된 command 의 목록을 보여준다.(Figure 3-9) 이 Commands Menu screen 은 어떤 스크린에서든 ? key 를 사용하여 시작할 수 있다.

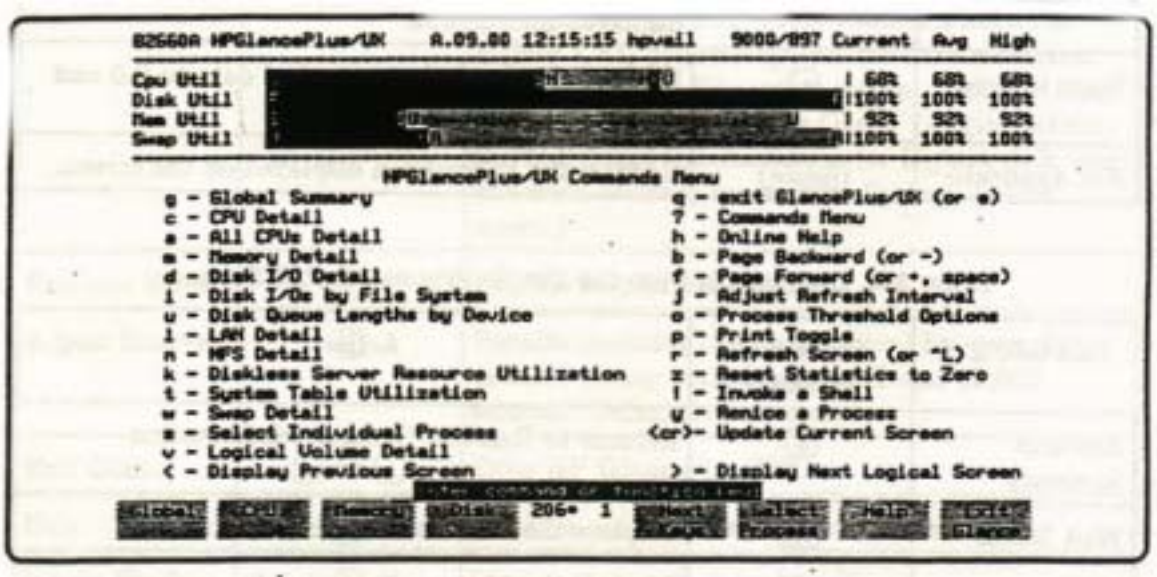


Figure 3-9. The GlancePlus Commands Menu Screen

Detail screen 에 접속하기 위해 사용된 commands 는 Commands Menu screen 의 왼편에 목록화 되어 있고, 특정 activity 를 시작하기 위해 사용되는 commands 는 오른편에 있다.

D) Getting Help Online

스크린에 나타나는 통계치에 대한 정보를 얻기 위해 h 를 타이핑하거나 function key Help 를 이용하여 online help 에 접근할 수 있다. 스크린에 관련된 정보는 스크린 안의 창에 나타날 것이다. (Figure 3-10)

다음 page 에 이동하기 위해서는 f , +, Space Bar 를 타이핑하거나 function key Page Forward 를 이용할 수 있고, 이전 page 로 이동하기 위해서는 b, -, 혹은 function key Page Back 을 이용하면 된다.

help 를 빠져나오려면 e 나 F8 을 이용한다.

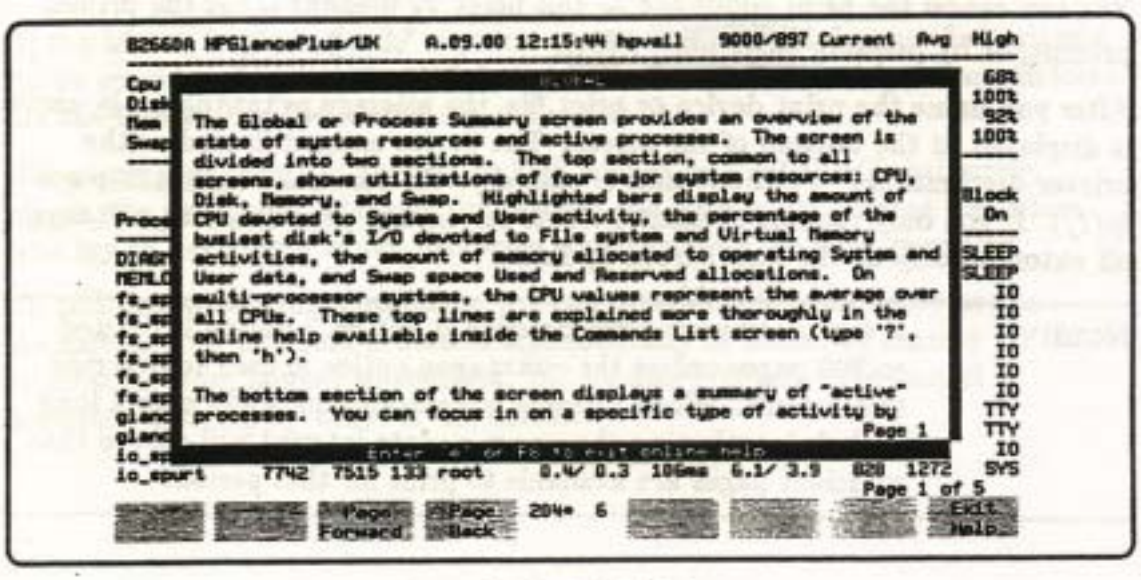


Figure 3-10. A Help Screen

E) Printing a Screen

p 나 Print Toggle key 를 이용하여 쉽게 GlancePlus screen 을 인쇄할 수 있다. toggle 과 같은 key operate 중 하나가 printing on 이나 off 로 바뀐다. Toggled on 될때, Print command prompt 는: **Single scrren or continuous printing (s/c)?:** s 를 치면 현재 스크린만을 인쇄할 것이고, c 를 치면 GlancePlus 는 next update 를 시작하면서 printer 에 screen image 를 보내고, printing 이 다른 Print command 에 의해서 toggled off 될때까지 각 update 를 지속할 것이다.

printing 을 toggle on 시킬때, 어떤 print device 나 print file 이 printing 을 할것인지 규정지을 것이다. System printer, 전형적으로 lp, 가 default print device 이다. 만약 적당치 않은 device 를 입력하면, 출력이 되지 않는다.

Printer prompt 에서 q 나 Cancel (F8)을 이용하여 print command 를 취소 시킬 수 있다. Print device 나 print file 을 규정지은 (name) 후 lp name 으로의 message printing 이 스크린의 밑부분에 나타날 것이다. 입력된 device 는 printer destination (**-d** option)으로 사용된다.(see man-page

lp(1)) 만약 single screen 인쇄를 요청했다면, print function 은 자동적으로 next screen update 때 꺼질 것이다.

Note 만약 지속적으로 print function 이 남아 있다면, **-maxpages** option 이 이 default 를 덮어씌우지 않는한 printing 은 200 page 까지 한정되어 있다. printing screen 이 자동적으로 오랜 시간동안 걸쳐 있을 때, screen update interval 을 늘리는 것은 그 시간동안 충분한 page 가 인쇄되도록 하기 위한 것이다.

F) Setting Process Thresholds

Glance 에 익숙하게 되면, 시스템 환경을 맞추기 위해 threshold 값을 세팅하기를 원할 것이다. (Figure 3-11) 예를 들면, 단지 초과적인(excessive) disk activity 를 보여주는 프로세스를 보기를 원한다면, CPU Utilization 과 Resident Set Siz 에 대해서 높은 threshold 값을 세팅하고, Disk I/O Rate threshold 에 대해서는 낮은 값을 세팅하면 된다.

Match Logic setting 은 이러한 threshold 들이 어떻게 다루어 지는지를 결정한다. Match Logic 가 "or"면, 특정 범주(criteria)의 어떤 것 *(any)*을 맞추는 프로세스가 interesting 으로 고려되어지고 Global screen 에 보여진다. 이것은 만약 User Name, Program Name, 혹은 TTY Path Name 이 "all"로 세팅되어 있다면 미미하게 다양할 것이다. 그러한 경우에서는 "or" match logic 가 첫 세번째 범주를 지나는 프로세스들을 고려할 것이다.

만약 Match Logic 가 "and"면, interesting 으로 고려되기 위해서 프로세스는 특정 범주의 모두(*all)* 를 맞추어야(match) 한다. Global screen 이 Interesting Process Threshold options 에 의해서 영향받는 유일한 스크린이라는 것을 참고하자.

한 page 에 Global screen display 를 제한할 만큼 충분히 높게 .threshold 값을 세팅하도록 권유된다. 한 page 를 초과하는 display 는 어떤 프로세스가 가장 많은 자원을 썼는지를 결정하는데 혼동을 주기 쉽다.

Threshold 를 세팅하는 것 외에, 어떻게 프로세스들이 정렬(sort) 될 것인지를 결정할 수 있다. 이름의 알파벳순으로, 현재 CPU utilization 사용순으로, 그리고 disk I/O rate 순으로 프로세스들을 정렬할 수 있다. 이름순으로 정렬하는 것이 default 이다.

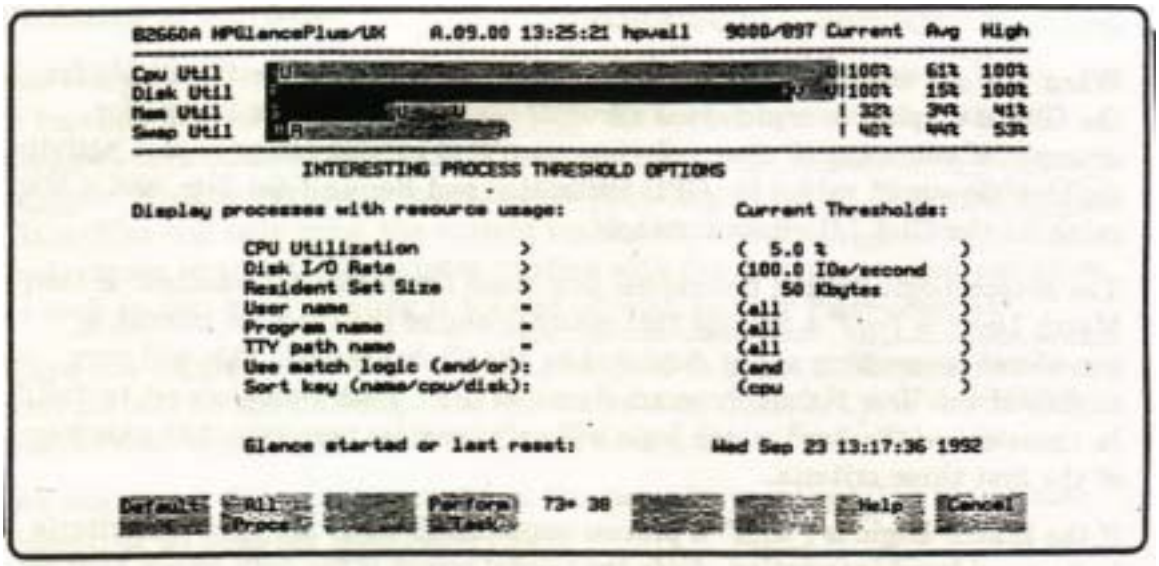


Figure 3-11. The Interesting Process Threshold Options Screen

Interesting Process Threshold Options screen 은 6 가지 프로세스 threshold 값을 세팅할 수 있게 한다.(Figure 3-11) 첫번째 3 개 – CPU utilization, Disk I/O Rate, Resident Set Size –는 수치상의 값을 필요로 하고, 마지막 3 개의 threshold 는 특정 user, program, 혹은 TTY path 를 규정하는 필터들이다.

이 스크린에 접근하기 위해서 o 나 Process Threshold 를 이용하면 된다.
현재 threshold 값들은 스크린의 오른쪽에 보여진다. 여러분은 새로운 threshold 값들을 타이핑하거나 threshold 값을 세팅하기 위해서 function key 를 사용할 수 있다.

default value 로 프로세스를 세팅하기 위해서: **default** 를 타이핑하거나 Default 로 labeled 된 F1 key 를 누른다. 숫치상의 threshold default 값은 각각 0.0%, 0.0 IOs/second, 그리고 1000.0 Kbytes 이다. 필터의 default 값은 all 이고, Match Logic default 는 or 이다.

midaemon 이 기록을 시작한 이후로 모든 프로세스들이 어떤 activity 를 했는지를 보기 위한 option 을 세팅하기 위해서는, 첫번째 줄에서 **allprocs** 를 치거나 All Procs 라 labeled 된 F2 를 누른다. (See chapter 6 for information about the midaemon)

이전에 세팅한 threshold 값을 restore 하기 위해서는 첫 줄의 **cancel** 를 치거나 Cancel 이라 labeled 된 F3 을 이용한다. 모든 줄에 Return 을 누르는 것은 같은 일을 하는 것이다.
현존하는 값을 계속해서 유지하기 위한 key 는 Return 이다.
만약 reset 을 하면, 새로운 값은 user 의 home directory 에 위치한 **.glanceerc** file 에 저장되고, GlancePlus 가 다시 구동 될 때 restore 된다.

1. Examples of Process Threshold Settings

프로세스 threshold 들은 매우 유연하여 여러분의 특정 필요를 맞춰줄 수 있지만 또한 매우 복잡하다. 아래의 예들은 어떻게 그것들이 적용되는지 이해하는데 도움을 줄것이다.

어떤(*any*) disk activity 에 연관된 프로세스들에 중점을 두기를 원한다면, 아래의 세팅을 따르면 된다. 이런 thresholds 값으로 Global screen 은 physical disk I/O 를 수행하는 모든 프로세스들을 보여줄 것이다.

CPU Utilization	>	all
Disk I/O Rate	>	0
Resident Set Size	>	all
User name	=	all
Program name	=	all
TTY path name	=	all
Use match logic (and/or):		and
Sort key (Name/CPU/Disk)	=	name

만약 "vi"에 관련된 모든 경우를 보고 싶다면, 아래의 세팅을 따르면 된다:

CPU Utilization	>	all
Disk I/O Rate	>	all
Resident Set Size	>	all
User name	=	all
Program name	=	vi
TTY path name	=	all
Use match logic (and/or):		and
Sort key (Name/CPU/Disk)	=	name

Disk I/O 를 행하고(and), 1MB 이상의 resident set size 를 가진 모든 프로세스를 보고 싶다면, 아래처럼:

CPU Utilization	>	all
Disk I/O Rate	>	0
Resident Set Size	>	1000
User name	=	all
Program name	=	all
TTY path name	=	all
Use match logic (and/or):		and
Sort key (Name/CPU/Disk)	=	name

"root"라는 user name 을 가지고 있고(and), 그것들이 pseudoterminal "pty/ttys6"에 attach 되어 있는 모든 프로세스에 관심이 있다면, 아래처럼:

CPU Utilization	>	100
Disk I/O Rate	>	100
Resident Set Size	>	10,000
User name	=	root
Program name	=	all
TTY path name	=	/dev/pty/ttys6
Use match logic (and/or):		or
Sort key (Name/CPU/Disk)	=	name

마지막 예에서 자원의 고 소비자인 다른 프로그램들이 "interesting" 되는 것을 막기위해 CPU, Disk, 그리고 Resident Set Size 를 세팅했다.

만약 threshold 값을 올린 후에, "interesting"으로서 충분한 프로세스 quality 를 가지지 못한다면, threshold 값을 낮추면 된다.

각각의 상황에 따라 다른 threshold 값을 요구하고, 시스템은 각자 독자적이다. Process ThresholdsDefinition screen 에 친밀해져서 어떤 threshold 값이 여러분의 시스템에 가장 적합한 정보를 주는 지를 판단할 필요가 있다.

4th Improving Performance Using HP GlancePlus/UX

이 Chapter 는 HP GlancePlus/UX 가 어떻게 다른 수준의 성능 분석을 제공할 수 있는지를 보여준다. 첫번째 섹션에서는 일반적인 시스템 성능 문제들을 묘사하고, 시스템 성능을 분석하기 위해서 스크린의 통계적 데이터를 어떻게 사용하는지에 대한 일련의 예들을 제공한다. 두번째 섹션에서는 GlancePlus 가 어떻게 성능 문제를 인지하도록 도와주는지를 상세하게 보여주는 경우들을 제공한다.

A) Understanding System Performance

모든 시스템은 다르다: 어떤 시스템은 많은 CPU-intensive program 을 구동할 수도 있고, 어떤 시스템은 자원에 대한 더 높은 요구을 필요로 할 수도 있다. 여러분 시스템의 유형(type)이나 크기(size)에 관계없이 특정 행위들이 특정 문제에 연관되어 있다는 것을 알게 될 것이다.

GlancePlus 는 다양한 정보를 제공하지만, 이 정보들에 대한 최적의 사용은 얼마나 잘 시스템의 operation 을 이해하고 있는 지와 시스템에 무엇이 적절한 조치인지를 알고 있는지에 달려있다. 여러분 시스템의 성능을 관찰하기 위해서 GlancePlus 를 사용하기 때문에, 정상적이지 않은 패턴 – 문제를 일으킬 수도 있는 패턴 -- 을 인식하는 것을 배울 것이다.

1. Bottlenecks

bottleneck 은 가장 일반적인 형태의 문제이다. 이것은 hardware 나 software 가 맞춰져야 하는 요구사항을 만족시키지 못할 때 발생하고, 프로세스들은 자원이 이용 가능하게 될 때 까지 기다려야만 한다. 이것은 block 과 long queue 로 결과되어 진다.

여러분의 시스템은 마치 freeway 시스템이 교통을 다루는 것처럼 프로세스들을 다룬다. 일반적인 시간동안에는 freeway 가 적당하게 교통 양을 다루고 차들은 최적의 속도로 달릴 수 있다.

그러나 rush hour 동안 너무 많은 차들이 freeway 를 이용하려 들 때, 길은 막히게 되고 교통은 느려지거나 멈추게 된다. 그 freeway 는 bottleneck 에 걸린 것이다.

비슷하게, 프로세스가 이용 가능한 것보다 더 많은 CPU time 을 필요로 하거나 시스템에 구성된 것 보다 더 많은 memory 를 요구할 때 bottleneck 이 일어날 수 있다. 또한 데이터를 이동시키기 위한 충분한 disk I/O bandwidth 가 없거나, swap space 가 적절하게 구성되어 있지 않을 때에도 일어 날 수 있다.

bottleneck 은 쉽게 고칠 수 있는 일시적인 문제일 수도 있다. 늦은 밤에 구동하는 batch program 을 다시 스케쥴링하는 것과 같이 workload 를 재배치 하므로써 해결할 수도 있다. disk bottleneck 의 해결책으로 모든 이용가능한 disk 들 사이에서 disk load 를 확장하는 것만을 요구하기도 한다.

하지만, 반복하는(recurring) bottleneck 은 상황이 더 악화 되는 long-term situation 이 될 수 있다. 아마도 그 시스템은 현재 그 시스템을 사용하기 보다는 사용자가 거의 없는 경우이거나, workload 가 시스템 용량 밖으로 점차 증가한 경우이다. 유일한 해결책은 hardware upgrade 일 수도 있는데…. 그렇지만 어떻게 알 수 있겠는가?

만약 정확하게 bottleneck 을 구별할 수 있다면, (문제를 더 악화시킬 수도 있는) 시스템 튜닝(tuning)을 피할 수 있고, 성능에 전혀 도움을 주지 못하는 extra hardware 를 추가하는 것을 피할 수 있을 것 이다. 또한 필연적인(corollary) bottleneck – primary bottleneck 에 의해 발생하는 – 을 해결하면서 자원을 확장 시키는 것을 피할 수 있다.

2. Characteristics of Bottlenecks

일반적인 시스템 bottleneck 은 몇 가지 일반적인 특징이나 증상을 가진다. GlancePlus screen 의 통계치와 이러한 증상들을 비교함으로써, 시스템 성능을 분석하고 가능성이 있거나 이미 존재하는 bottleneck 을 탐지 할 수 있다. 비록 하나의 증상이 문제를 지시하지는 않을 것이지만; 통합된 몇 가지 증상들로 bottleneck 상태를 탐지할 수 있다.

① Symptoms of a CPU Bottleneck

available idle time 없는 long run queue

user mode 에서의 high activity
system mode 에서의 reasonable activity (high activity 는 또한 다른 bottleneck 문제를 야기할 수있다.
많은 프로세스들이 종종 priority(PRI)에 blocked on 되어 있다.

② Symptoms of a Memory Bottleneck

High swapping activity
High paging activity
매우 적은 free memory available
swap device 에서의 high disk activity
system mode 에서의 high CPU usage

③ Symptoms of a Disk Bottleneck

High disk activity
I/O requests 를 충족하기 위한 CPU idle waiting
Physical reads/writes 의 high rate
Long disk queues

④ Symptoms of Other I/O Bottlenecks

High LAN activity
Low I/O throughputs

여러분들은 하나의 bottleneck 을 해결하는 것이 다른 또 하나를 찾아내거나 생성할 수도 있다는 것을 알게 될 것이다. 하나의 시스템에서 하나 이상의 bottleneck 을 가지는 것이 가능하다. 사실, workload 를 변화시키는 것은 시스템 성능을 변화 시키는 데에 지속적으로 반영된다. 목표는 최종 해결책을 찾는 것이 아니라, 주어진 시간 내에 최적의 성능을 추구하는 것이다.

3. An Approach to Monitoring System Behavior

시스템 성능을 모니터링 하는 최적의 접근방법은 일반적으로(*usually*)시스템이 어떻게 행동

(behave) 하는지에 익숙해 지는 것이다. 이것은 **activity** 상의 갑작스런 변화가 정상적인지 혹은 문제 발생 여지가 있는지를 인지하는데 도움을 준다.

Global screen 은 처음 GlancePlus 를 시작할 때 나타나는 첫 스크린이다. 이 스크린은 넓게 시스템 activity 를 요약하고, usage threshold 를 초과하는 모든 프로세스들의 리스트를 보여준다. 이 스크린의 정보는 자원이 초과적으로 사용되고 있는지, 혹은 한 프로세스가 이용가능한 자원들을 독점하고 있는지를 보여준다. Global screen 은 시스템 activity 와 성능을 관찰하기 위한 일반적인 starting point 이다.

전체적인 시스템 activity 를 모니터하기 위해서 Global screen 상의 통계치를 사용할 필요가 있고, 혹은 특정 분야의 시스템 사용에 중점을 두기 위해 세부적인 데이터 스크린을 참조할 필요도 있을 것이다. 이 장의 예들은 모든 GlancePlus screen 의 사용을 설명한다. 각 스크린은 Chapter 5 에서 상세하게 묘사되어 있다.

B) Examples of GlancePlus/UX in Use

1. Evaluating System Activity

어느날 오후 Dong 은 시스템 response 가 느려졌다는 것을 알아차렸다. 그는 전체적인 시스템 activity 를 보기 위해 GlancePlus 를 구동시켰고 Global screen 을 관찰 했다. 그는 CPU usage 가 100%에 다다른 것을 보았고, 그는 이것이 점검하기로 결정했다.

Dong 은 usage threshold 를 초과하는 모든 프로세스가 목록화 되어 있는, Global screen 의 process summary section 을 보고서 하나의 프로세스가 100% CPU usage 의 많은 부분을 차지 한다는 것을 알아냈다.

그 특정 프로세스에 대한 더 많은 정보를 얻기 위해 그는 **Individual Process** screen – 어떤 특정 프로세스에 대한 상세한 정보를 제공하는 – 을 체크했다. 그 스크린을 검토하면서, Dong 은 그 프로세스가 어떤 I/O 도 수행하지 않고 user code 에서 모든 시간을 소비한다는 것을 알아냈다. 이것은 그 프로세스가 CPU loop 에 걸렸다는(trapped) 것을 의미한다.

user 의 이름을 알아낸 후, 그는 그 프로세스를 죽여도 되는지 알아보기 위해 전화를 했다.
이런 상황에서, 다른 프로세스들이 그 늘어짐(slack)을 차지하기 때문에 그 looping process 를 끝내더라도, CPU use 는 떨어지지(drop) 않는다. 하지만, 다른 프로세스들이 기다릴 필요가 없기 때문에 response time 은 향상된다.

2. Evaluating CPU Usage

Dean 은 어느날 오후 시스템 response time 이 현저히 느려진 것을 알고서, 시스템을 체킹하고 있었다. 그는 GlancePlus 의 Global screen 을 이용하여 CPU usage 가 거의 100%에 달하고, Memory Disk, 그리고 Swap 과 같은 나머지 자원들의 사용율은 매우 낮다는 것을 알았다.

좀더 관찰하여, CPU (PRI)를 사용하는 다른 프로세스들 때문에 몇몇 프로세스들이 막혀 있다는 것을 알아냈다 – 이것은 그것들이 더 높은 우선 순위를 가진 프로세스의 수행이 끝날 때 까지 기다리고 있다는 것을 의미한다.
Dean 은 CPU time 이 다른 상태 (state or activity)에 어떻게 할당되었는지를 알아보기 위해서 **CPU Detail** screen 에 접속했다. 그는 Real-Time activity 가 다른 activity 들보다도 훨씬 더 많은 percentage 로 사용하고 있다는 것을 찾아냈다.

Dean 은 우선순위(priority)를 체크하기 위해서 Global screen 으로 돌아왔다. 한명의 user 가 real-time processing priority 인 127 의 priority 를 가지고 구동하고 있었다. Dean 은 이 특정 프로그램이 CPU-intensive 이고, 그런 높은 우선순위에서 그것을 구동시키는 것은 다른 프로세스들의 수행을 막을 것이라는 것을 알고있었다. 이미 그것은 시스템 성능 저하를 야기하고 있었다. 그는 GlacnePlus renice command 를 이용하여 더 낮은 timeshare priority 로 그 프로세스의 우선순위를 reset 시켰다. 이것은 다른 프로세스들이 더 지속적인 CPU 로의 접속을 할 수 있고, 시스템 response time 은 향상된다.

3. Evaluating Wait States

Jose 의 시스템은 다른 application 을 install 하기 전까지 잘 구동되고 있었다. 이제 그 application 을 구동할 때마다 response time 은 저하된다. 그 application 이 그 시스템의 유일한 변화였기 때문에, Jose 는 그것이 시스템을 어떻게 사용하고 있는지를 검토하기 시작했다. Glance Individual Process screen 을 보면서, 그는 CPU utilization 이 약 7%이고 이것은 문제가 아니라는 것을 확인했다. 다음으로 시스템에서의 전체적인(overall) CPU utilization 을 검토하였는데, 평균 약 48%였다.

– 이것은 새로운 application 의 추가하에 충분한 CPU 자원이 있다는 것을 의미한다. Jose 는 disk I/O 를 검토하고서, 그 application 이 초당 약 5 I/O 를 –그것 중 대부분은 virtual memory I/O -- 프로세싱하고 있다는 것을 알았다. Jose 에게 그것은 느리다고 느껴졌고, 그래서 그는 그 프로세스가 무엇을 기다리고(waiting on) 있는지를 알기 위해서 Wait States screen 을 보았다.

Jose 는 그 프로세스가 약 7%의 time utilizing the CPU, 27%의 time waiting for terminal input, 그리고 66%의 time waiting on virtual memory 을 쓰고 있다는 것을 알아냈다. 그것은 time 의 너무 많은 양이다. Jose 는 시스템의 다른 프로세스들을 체크하고 그것들도 비슷한 waits for virtual memory 를 가진다는 것을 보고서, 새로운 application 이 그 시스템의 memory 를 overload 하고 있다는 것을 눈치챘다. 그는 관계된 screen 들을 복사해서 이 상황을 manager 에게 설명할 수 있었다.

4. Evaluating Memory Usage

Terry 의 시스템은 response time 이 매우 늦다. 그녀는 전체적인 윤곽을 잡기 위해서 Global screen 을 점검했다. 모든 4 개의 시스템 자원들이 (CPU, Disk, Memory 그리고 Swap) 거의 100%에 가깝다. Disk bar activity 의 대부분은 virtual memory activity 를 보여준다. 더불어 **swapper** 시스템 데몬은 지속적으로 구동하기 위해서 나타나 있다.

Terry 는 이것이 memory bottleneck 일 수도 있다고 생각했다. 그녀는 **Memory Detail** screen 을 점검했는데, 이것은 page fault, paged-in 되고 paged-out 되는 page 의 수, 그리고 virtual memory 로의 접속 수와 같은 memory management event 에 대한 통계치를 제공한다. 그 스크린은 Free Memory 가 0.0mb 로 나왔고 -- 이것은 이용할 수 있는 메모리의 부족현상을 지칭한다 – Swap In/Outs 은 초당 1 보다 약간 높은 비율로 나왔다.

문제가 memory bottleneck 이라고 규정짓고서, Terry 는 active process 들을 관찰하기 위해서 Global screen 으로 돌아왔다. 그녀는 더 많은 memory 를 추가하거나 active process 들의 memory 요구를 줄임으로써 memory bottleneck 을 해결할 수도 있다는 것을 알고 있다. 이런 경우에, 대부분의 active process 들에 대한 큰 Resident Set Size 에 의해서 높은 swap rate 이 발생할 수도 있다는 것을 의심했다.

하나의 test 프로그램은 같은 비율로 증가하게 보이는 큰 resident set size 을 보여 주었다. 이런 상황을 좀더 자세히 조사하면서, Terry 는 그 프로그램이 "memory leak"을 가지고 있다는 것을 알게 되었다. 그것은 **malloc** 을 사용하여 memory 를 할당하였지만, **free** 를 사용하여 memory 의

free 를 높이지는 않았다. 그 프로세스의 메모리 할당(memory allocation)은 시스템에 memory pressure 를 유발하면서 점차 증가한다.

그녀는 program code 를 연구하고 memory leak 을 찾아낸 개발자에게 말했다. 그 test program 은 변화되어 훨씬 적은 memory 를 사용하도록 다시 컴파일 되었고, 따라서 memory bottleneck 은 완화되었고 시스템의 response time 은 향상되었다고.

5. Evaluating Disk Usage

Vivian 의 회사는 종종 available 한 memory 에 부담을 지우는 프로세스들을 구동시킨다. 그녀는 **Disk Detail** screen 을 점검하면서 – 이 스크린은 모든 mount 된 disk device 의 logical 하고 physical 한 I/O request 를 보여준다. – 상황을 계속해서 살핀다. 그것은 또한 User, Virtual Memory, System, 그리고 Raw request 로 physical 한 request 를 구분 짓는다. 이 스크린은 언제 많은 수의 physical read 와 write request 가 일어났는지, 이 상황이 프로세스에 의한 과도한 page fault 의 결과인지를 보여준다. Vivian 은 또한 virtual memory request rate 을 검검하는데 또한 그것은 시스템의 요구가 (demand) physical memory 역량에 부담을 지울 때 높아진다.

Virtual memory activity 가 높을 때 어떤 프로세스들이 active 한지를 주의 깊게 봄으로써, 시스템 load 의 balance 를 맞추기 위해 activity 들을 재분배하는 것에 대한 현명한 결정을 내릴 수 있다. 이것은 시스템의 전체적인 이해를 넓히는데 도움이 된다.

6. Evaluating I/O by File System

Ingrid 는 시스템이 swapping 을 행할 때 시스템 성능이 갑작스럽게 저하된다는 것을 알게 되었다. Global screen 을 보고서, swapper 프로세스가 구동중인 것을 알고 virtual memory 사용이 Disk Utilization 의 높은 percentage 를 차지한다는 것을 알게 되었다.

그녀는 어떤 disk 가 가장 바쁜지(busiest) 확인하기 위해서 **Disk I/O by File System** screen 을 검토했다. **Disk I/O File System** screen 은 각 file system 혹은 mounted-disk partition 에 대한 I/O rate 의 세부사항을 제공한다. 이 정보는 disk load 를 balancing 하는데 매우 유용하다. 그녀가 Disk I/O by File System screen 에서 Disk I/O 를 보았을 때, 하나의 disk 가 모든 다른 disk 보다도 더 많이 사용되고 있음을 발견할 수 있었다. 가장 많이 사용되고 있는 그 disk 는 swap disk 였다.

Ingrid 는 그 하나의 disk 에 대한 load 를 완화하기 위해서 추가적인 swap disk area 를 더하기로 결정했다. 그녀는 덜 사용되는 file system 에 dynamic swap area 를 할당하는 것을 고려했을 수도 있다.

7. Evaluating Disk Queue Lengths

Ray 는 이미 그의 시스템이 disk I/O bottleneck 에 걸렸다고 결정했다; Global screen 을 보고서 Disk Utilization 이 거의 항상 100%에 달했다는 것을 보았기 때문이다. 그는 몇 가지 disk 들이 너무 많이 사용되고 있는지를 보여주는 Disk I/O by File System screen 을 검토했다.

Ray 가 찾고자 하는 것은 그 상황을 좀더 쉽게 하는 방법을 찾는 것이다. 그는 disk 들이 얼마나 잘 I/O request 를 process 할 수 있는지를 보여주는 **Disk Queue Lengths** screen 을 참조한다. 그는 바쁜 disk 중에 어떤 것이 서비스에 대한 가장 긴 지연을 갖게 하는지를 알고 싶어한다.

그는 "busy" 한 disk 가 꼭 long queue length 를 가지지는 않는다는 것을 알고 있다. 높은 disk utilization 은 프로세스들이 disk 를 사용하기 위해서 기다리지 않는 한 문제가 되지 않는다. 예를 들어, 전화 시스템에서 높은 비율의 회선은(line) call 연결되기만 한다면 문제가 되지 않는다는 것이다.

Ray 는 또한 long queue length 가 그 drive 가, 다른 request 들을 서비스하는 동안 몇 가지 disk request 가 기다려야만 한다는 것을 의미한다는 것을 알고있다. 예를 들면, 전화 회선이 busy 하다면, 걸려오는 전화는 기다려야만 한다는 것이다.

그가 명확하게 그 상황을 알게 되었을 때, Ray 는 workload 를 좀더 균등하게 분배하기 위하여 몇 가지 file 들을 다른 file system 으로 옮김으로써 large queue length 를 줄였다.

8. Evaluating NFS Activity

Paul 은 network file system 으로 사용되는 시스템을 관리한다. 하나의 local disk 가 몇 가지 다른 node 로부터 NFS-mounted 되어있다. 어느날 오후, Paul 은 시스템의 response time 이 부실하다는 것을 알아냈다. Remote 시스템에 의해서 mounted 된 file system 은 매우 active 하다.

Paul 은 local 시스템에 대한 inbound 와 outbound network file system(NFS) activity 에 대한 현재 통

계치를 제공하는 **NFS Detail** screen 을 검토했다. 그는 어떤 remote 시스템이 disk 를 가장 많이 사용하고 있는지를 알고 싶어 한다. 그는 하나의 시스템으로부터 큰 **Inbound Reads** rate 을 찾아냈다. 이것은 그에게 왜 NFS-mounted disk 를 과도하게 사용하는지를 알아내기 위해 remote 시스템을 검토하게 한다.

그의 검토는 remote 시스템에서 single user 로의 상황을 정확하게 지적했다. 그 user 는 NFS-mounted disk 에 있는 file 들에게 반복되고 불필요한 **greps** 을 하고 있었다. Paul 은 user 에게 그 문제를 설명했고 많은 disk 사용을 줄이기 위해 그녀와 함께 조치를 취했다. 이것은 NFS server 에서의 load 를 줄였고, 전체적인 response time 을 향상 시켰다.

9. Evaluating LAN activity

Lee 는 LAN 을 통해 data 를 접속하기 위한 datacomm 서비스를 사용하는 application 에 대한 response time 이 느리다는 것을 알게 되었다. 그는 무엇이 문제를 일으키고 있는지를 알아내기 위해서 **LAN Detail** screen 을 검토했다. LAN Detail screen 은 그 시스템에 구성된 각 LAN card 에 대한 4 가지 function 들을 묘사한다. Network 시스템에서 이 정보는 과도한 LAN activity 에 의해서 야기될 수 있는 bottleneck 을 보여준다.

Lee 는 **Collision** 과 **Errors** rate 가 보통보다 더 높다는 것을 찾아냈다. 이 정보는 그로 하여금 프로세스들이 LAN resource 를 위해 경쟁하는지 혹은 LAN software 나 hardware 를 과부화 시키고 있는지의 여부를 조사하게 했다.

이 경우, **netipc** 에 의해 부적절하게 쓰여진 application 이 bottleneck 을 일으키고 있다. 일단 이 프로그램이 멈추면, LAN 을 이용한 다른 프로그램들은 그들의 response time 을 향상시킬 수 있을 것이다.

10. Evaluating Diskless Server Resource Utilization

Roberto 의 시스템은 swap space 를 낮게 구동시키는 서버를 가지고 있었다. 그는 어떤 client 가 그 자원의 대분분을 사용하고 있는지를 알고 싶어 했다. 그는 많은 memory 요구를 가진 프로세스들이 더 많은 swap space 를 예약하고 있다는 것을 알게 되었다. 그는 또한 서버의 disk 가 너무 busy 하면, 어떤 client 가 서버에 대한 I/O request 를 가장 많이 유발하는지를 추적하기 위해서 **Diskless Server Resource Utilization** screen 을 사용할 수 있다. 이 스크린은 cluster 서버의 자원에

대한 diskless cluster client 요구에 관한 정보를 제공한다.

Roberto 는 그 스크린을 보고서, client node 를 대신하여 reserved 된 서버의 local swap space 의 양을 체크했다. 그리고서 가장 많은 swap space 를 사용하고 있는 client 에 대한 GlancePlus 를 구동시켰다. 그 Global screen 은 어떤 프로세스들이 가장 많은 memory requirement 를 가지고 있는지를 보여주었다. 그 client 의 관리자에게 높은 memory 사용의 프로그램들을 동시에 사용하지 말도록 제안함으로써, Roberto 는 그 cluster server 의 swap space 부족 현상을 완화 시킬 수 있었다.

장기간의 해결을 위해서, Roberto 는 그 client 에게 그 자신의 local swap disk 를 주는 것을 고려할 수도 있다.

11. Evaluating System Table Utilization

Debbie 가 시스템의 한 프로그램을 구동 시켰을 때, 그 프로그램은 다음과 같은 message 를 주면서 오류가 났다.: **fork failed – too many processes.**

kernel 에 있는 nproc 의 값을 재 구성할 것인지 아닌지를 결정하기 위해서, 그녀는 그 Process Table 에 얼마나 많은 공간이 있는지를 알 필요가 있었다. 그녀는 몇가지 중요한 internal system table 의 구성된 size 와 그 사용에 대한 정보를 제공하는 **System Table Utilization Detail** screen 을 참조했다. 그 정보는 kernel configuration 이 얼마나 잘 그 시스템의 평균 workload 로 변화되는지에 대한 feedback 를 제공한다.

Debbie 는 그녀가 **Proc Table** 에 공간이 바닥 났다는 것을 확인하고, 그 시스템의 buffer cache 가 꽉 차게 이용되고 있다는 것을 알았다. 한계치에 달하기 전에 kernel variable 을 적정하게 재 구성하기 위하여 다른 table 들을 그녀는 proactive 하게 모니터링 할 수 있었다.

12. Evaluating Swap Usage

Philippe 는 그의 시스템의 유용한 swap space 의 양을 걱정해서 **Swap Detail** screen 을 이용하여 그것을 체크하기로 맘 먹었다. 그 스크린은 모든 swap area 에 대한 space 할당 값(allocation figure)을 제공한다. Device swap space 에 대한 이용 가능한 area 를 체크해서, 그는 **swapon** command 에 의해 enabled 된 것보다 더 많은 이용가능한 swap area 를 가지고 있음을 알게 되었다. 그 command 를 이용하여 그는 추가적인 swap space 를 enable 시켰고, 그 상태를 다시 체크하기 위해

Swap Detail screen 을 이용했다.

Philippe 의 시스템은 또한 filesystem swap 이 있는데 그는 그 상태를 체크해야 했다. 그가 구성했던 그 값보다도 더 적은 이용가능한 양이 있었다. 이것은 구성된 file system space 의 최대 양보다도 swapping 을 위한 부분에(partition) 더 적은 공간을 남겨 놓은 채, 그 file system 이 일반적인 file 들로 가득 차 있다는 것을 의미했다. Philippe 는 원래의 file system 에 있는 swap 에 대한 더 많은 공간을 공급하기 위해서 다른 partition 으로 일반적인 file 들을 옮기기로 결정했다.

13. Evaluating an Individual Process

Cliff 는 한 프로세스가 너무 느리게 구동하고 있다는 것을 알아냈다. 그는 GlancePlus 를 구동시켜 그 프로세스에 대한 상세 정보를 확인했다. 그리고서 특정 프로세스에 대한 상세 정보를 제공하는 **Single Process Detail** screen 에 대한 통계치를 검토했다.

Cliff 는 만약 그 프로세스가 메모리 부족 때문에 느려졌다면, context switches 와 fault counts 가 증가 된 것을 확인할 수 있다고 생각했다.

그는 I/O read 와 write conunt 가 크고 그 프로그램이 많은 I/O 를 수행하고 있다는 것을 알아차렸다. 그는 그 프로세스가 무엇을 block on 하고 있는지를 체크했고 **Disk I/O** block 에 대한 높은 percentage 를 발견했다. 그는 disk throughput 용량(capacity)에 대한 경쟁 때문에 느려졌다고 의견을 비췄다.

그 프로세스는 Priority 에 blocked 되어있는 높은 percentage 를 보였고, 그 CPU 는 높은 dispatching priority 들을 가진 프로세스들에 의해 사용되기 때문에, 그 프로세스는 구동할 준비는 되어 있으나 그렇게 수행하지 못했던 것이다.

14. Evaluation Open Files

Kathryn 은 remote 시스템과 통신하기 위한 application 을 실행시키고 있었다. request 를 받을 때, 그 application 은 socket 을 열고(open) 그 특정 data 를 보낸다. 하지만, Kathryn 이 그 application 을 test 할 때 어떤 data 도 remote 시스템에서 받지 못했다. 무엇이 문제인가 알기 위해 그녀는 Glance OpenFiles screen 을 체크해서, opened socket 을 보았을 때 그것이 전혀 open 되지 않았다는 것을 알게 된다. 그녀는 application 으로 돌아가 coding error 가 있는지 찾아 본다.

15. Evaluating Memory Regions

어느날 Glance 의 global summary screen 을 검토하면서, Nancy 는 몇몇 프로세스들이 매우 큰 resident set size 들을 가지고 있다는 것을 알게 되었다. 이것은 과연 application 의 memory usage 상의 문제를 일으킬 수 있을까? 그녀는 나중에 추가적인 사용자들을 위해서 physical memory size 를 증가시킬 것인지에 대해 의문이 들었다. 그녀는 현 시스템의 성능이 좋고 memory size 가 적절하지만, 미래에 성능 저하를 막고자 하는 것이었다.

어떠한 결정을 내리기 전에, 그녀는 그 상황을 좀더 자세히 분석하기 위하여 Glance 의 Memory Regions screen 을 관찰했다. 그녀는 모든 affected process 들이 약 200KB 의 shared memory region 을 가지고 있다는 것을 알게 되었다. 개인적인 DATA 와 TEXT region 을 덧붙였을 때, 이것은 large resident set size 를 일으킨다. SHMEM region 의 virtual address 를 체크함으로서, 그 프로세스들에 의해 사용되고 있는 shared memory region 이 같다는 것을 알게 되었다. 그것은 shared region 에 있기 때문에 단지 한번 physical 하게 메모리에 있는 것이지만, Glance 는 각각의 프로세스가 shared memory region 에 붙어있는 것으로 보여준다. 그녀는 이것이 문제가 없다는 것을 알고서 미소를 지었다. Shared memory region 을 사용함으로써 그 프로세스는 실제 나타나는 것보다 훨씬 적은 memory 를 사용하고 있는 것이다.

16. Evaluating Activity on Logical Volumes

최근 Yuki 는 GlancePlus/UX 를 이용하여 시스템을 체크하는데, 그 global disk utilization bar – Glance screen 의 상위에 나와있는 부분 – 가 종종 100%에 이르는 것을 발견했다. Yuki 의 시스템은 multiple disk drive 들을 가지고 있지만, 그 global disk utilization figure 는 가장 바쁜(disk) disk 의 activity 를 지시하는 것이다.

Yuki 는 일어날 수 있는 I/O bottleneck 을 피하기 위해, drive 들 사이에 좀더 균등하게 disk I/O 를 퍼트리고 싶어한다. 그와 같은 목표를 가지고, 처음 Disk Detail screen 을 체크했는데, logical disk activity 가 높게 나왔다. 좀더 상세하게 보기 위해서, 그는 Logical Volumes screen 을 보았는데 logical volume /dev/vg00/lvol2 에서 높은 write activity 를 찾아 냈다. Glance 를 나와 Unix shell 로 들어가 그 volume 과 관련된 physical disk 이름을 확인하기위해 그는 "vgdisplay –v /dev/vg00"을 타이핑했다.

Glance 에서 다시 그 volume 에서 가장 busy 한 disk 를 결정하기 위해서 Disk Queue Lengths screen 을 보았다. 그리고 시스템이나 user activity 에 의해 발생된 disk activity 인지 아닌지를 알기 위해 Disk Detail screen 을 체크했다. Yuki 는 시스템 activity 보다는 application 을 지칭하는, Virtual Memory physical access 들이 낮다는 것을 알아냈다. 그는 어떤 application 이 disk 로의 그렇게 많은 writes 을 유발하는지를 알아보기 위해서 Open Files screen 을 체크했다. 야호! Fred 가 다시 야구 게임을 하고 있었던 것이다.

Fred 의 I/O need 에 대해 의견을 교환한 후, Yuki 는 logical volume 을 재 배치하기 위해 LVM commnad 를 사용하면서, I/O load 의 balance 를 맞추기 위해 console 을 찾아 간다.

17. Evaluating All CPUs Statistics

Rosalie 는 multi-processor 시스템에서 일하는데, 어느날 All CPUs screen 을 체크하면서 하나의 CPU 가 다른 것들보다 지속적으로 더 busy 하다는 것을 알게 되었다. 그 프로세스들 사이에 load balance 가 잘 되어 있다면 overall system throughput 은 향상될 것이다. All CPUs screen 을 보고서 하나의 PID 가 항상 그녀의 가장 바쁜 CPU 인 CPU 1 에서 실행하는 마지막 PID 같았다. 그녀가 **mpctl** 을 체크 했을 때, 그 프로세스가 CPU 1 에 할당된 것을 보았다. mpctl –f 를 이용하여, 그녀는 그 프로세스를 floater 가 되게 재 할당하게 하여, 그 시스템이 어떤 프로세서가 그 프로세스를 구동시켜야 하는지를 결정할 수 있게 했다. Rosalie 은 그리고서 CPU 1 에 할당된 다른 프로세스들을 체크하여 그것들을 floating process 들로 재할당 했다. 그렇게 한 후, 그녀는 All CPUs screen 을 다시 체크하고, 모든 그 프로세서들 사이에서 load 가 훨씬 더 균등하게 나타나는 것을 관찰하였다. 따라서 어떤 single CPU 에 대해서도 발생 가능한 bottleneck 의 여지를 완화시킨 것이다.

C) Case Study of a Diskless Cluster System

이 섹션은 Glance 가 어떻게 성능문제를 인지하게 하고 그 중요성을 평가하게 하는지를 보여주는 예가 되는 경우의 연구를(sample case study) 제공한다.
모든 시스템은 독특하고, 이 study 에 있는 상태와 응답(condition and response)들은 여러분의 경험

과 다를 수도 있다. 이 study 는 모니터링과 성능문제 해결에 관련하여 Glance 의 유용성을 표현하는 특징적인 접근방식을 보여준다.

1. The Situation

Dave 는 새로운 software product 를 개발하는 engineering 회사의 시스템 관리자이다. 매일 아침, Dave 는 GlancePlus 를 돌려서 5 개의 diskless cluster 들에 대한 activity 들을 관찰 한다. 그는 전체적인 activity 를 체크하고 어떤 cluster 도 즉각적인 조치가 필요한 문제를 일으키지 않는 다는 것을 확인한다.

오늘 아침, 그는 Server 1 을 사용하는 diskless cluster 를 특별히 주의 집중한다. 어제 그 cluster 를 사용하는 user 가 최근의 성능이 평소보다 느려졌다고 그에게 말했기 때문이다.

① Checking Porcess Thresholes

분석을 하기 전에, Dave 는 Interesting Porcess Threshold Option screen 을 관찰한다. 그는 interesting process 들중 오직 한 page 만이 Global screen 에 보여지도록 확실히 하기 위해, 그가 세팅한 threshold 값을 체크하기를 원한다. 그는 threshold option 들을 증가시킴으로서 Global screen 에 나타나는 프로세스들의 수를 줄인다는 것을 알고 있다. 그의 시스템의 threshold option 들은 figure 4-1 에 보여진다.

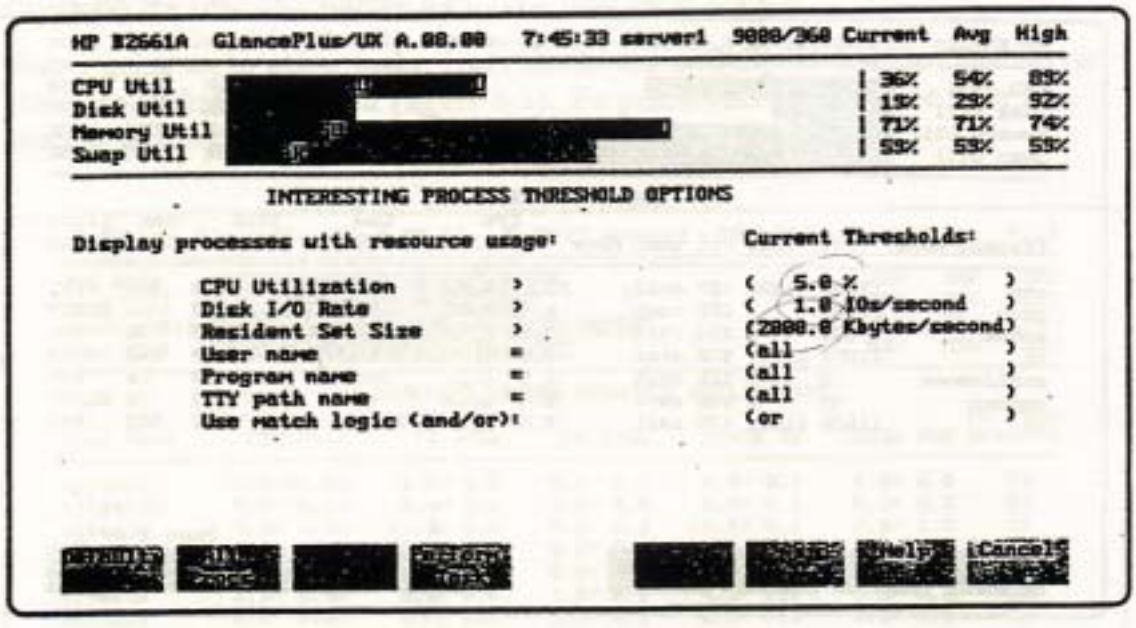


Figure 4-1. Process Thresholds Screen

Dave 는 현재 선택된 threshold 값들에 만족해서, Global screen 으로 돌아가기 위해 Cancel 을 누름으로써 존재하는 그 값을 그대로 남겨둔다.

② Checking the Global Screen

Dave 는 프로세스들의 데이터가 다른 threshold 들을 초과 했을 때 그 스크린으로부터 나왔다 사라졌다 하는 것을 보면서 잠시동안 그 스크린을 검토하고, 낮은 activity level 에서 끝내거나 지속시킨다. 그는 그가 보고 있는 그 화면의 행위들이 하루 중 그 시간대에 정상적이라는 것을 매우 잘 알고 있다. (figure 4-2)

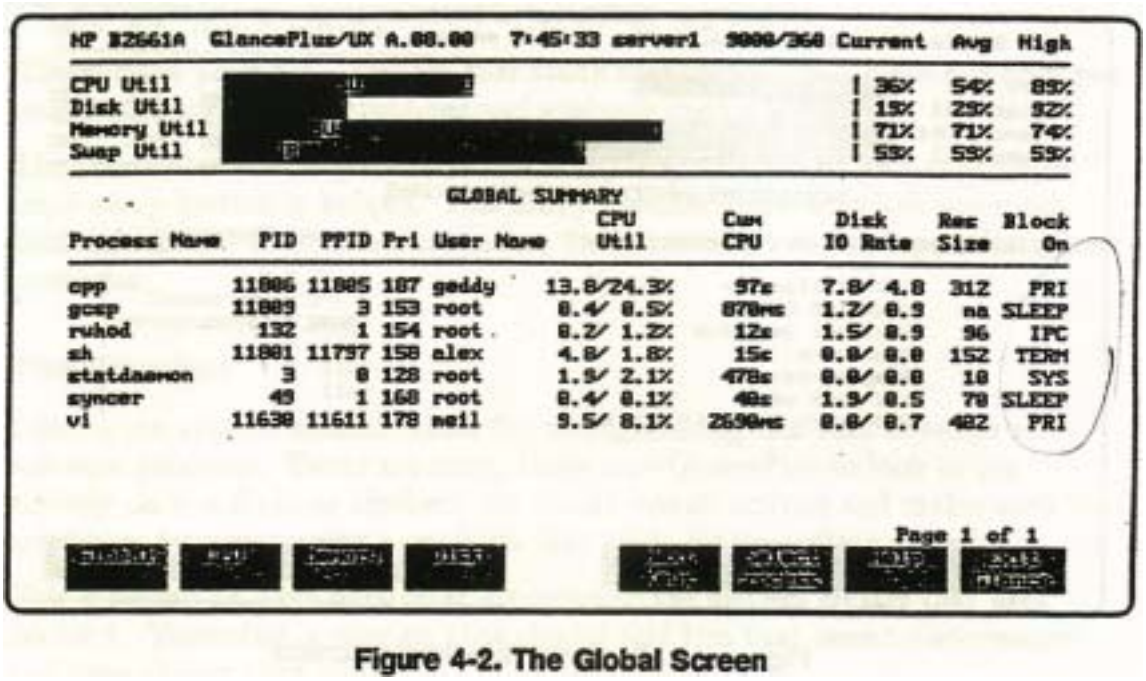
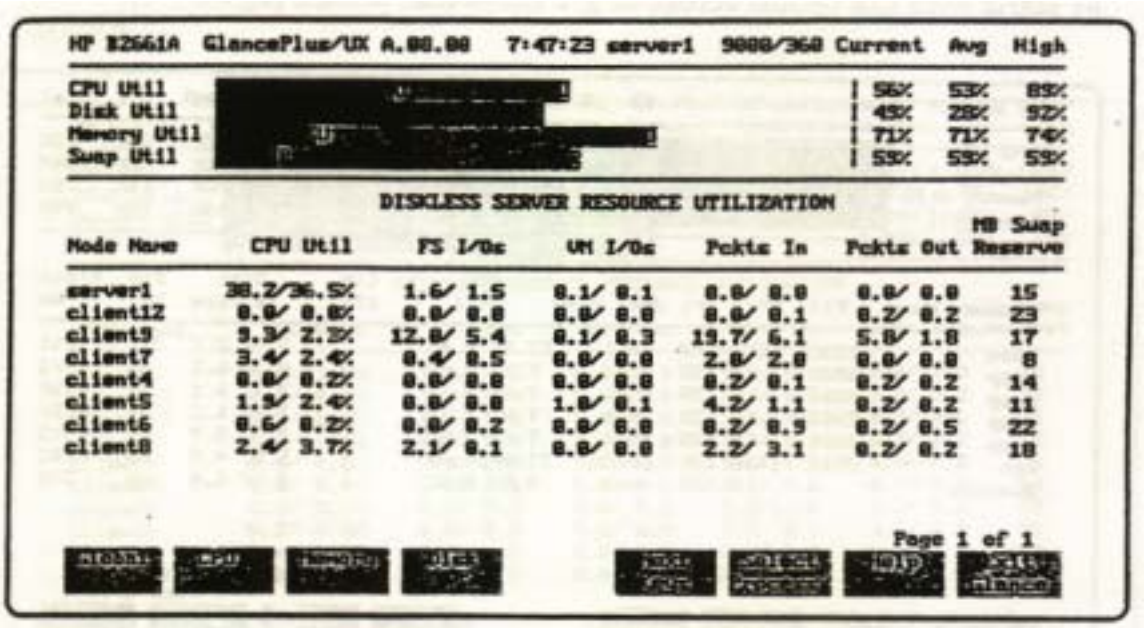


Figure 4-2. The Global Screen

③ Checking the Diskless Server Resource Utilization Screen

다른 task 로 가기 전에, Dave 는 Diskless Server Resource Utilization screen (figure 4-3)에 있는 activity 를 체크 한다. 나중의 activity 를 상대로 그것을 체크하기 위해 그는 이 스크린을 인쇄한다.



Node Name	CPU Util	FS I/Os	VM I/Os	Pckts In	Pckts Out	MB Swap Reserve
server1	38.2/36.5%	1.6/ 1.5	0.1/ 0.1	0.0/ 0.0	0.0/ 0.0	15
client12	0.0/ 0.0%	0.0/ 0.0	0.0/ 0.0	0.0/ 0.1	0.2/ 0.2	23
client9	3.3/ 2.2%	12.0/ 5.4	0.1/ 0.3	19.7/ 6.1	5.0/ 1.0	17
client7	3.4/ 2.4%	0.4/ 0.5	0.0/ 0.0	2.0/ 2.0	0.0/ 0.0	8
client4	0.0/ 0.2%	0.0/ 0.0	0.0/ 0.0	0.2/ 0.1	0.2/ 0.2	14
client5	1.5/ 2.4%	0.0/ 0.0	1.0/ 0.1	4.2/ 1.1	0.2/ 0.2	11
client6	0.6/ 0.2%	0.0/ 0.2	0.0/ 0.0	0.2/ 0.9	0.2/ 0.5	22
client8	2.4/ 3.7%	2.1/ 0.1	0.0/ 0.0	2.2/ 3.1	0.2/ 0.2	18

Figure 4-3. Diskless Server Screen

④ Later in the Morning

10:30 am 에, server1 을 사용하는 diskless cluster 의 한 user 가 "things aren't running fast enough"라고 불평 전화를 한다. Dave 는 그 서버의 GlancePlus 를 구동시켜 상황을 점검한다. 일단 다시 전체 상황 파악 (overall picture)를 얻기 위해 Global screen 을 시작한다.

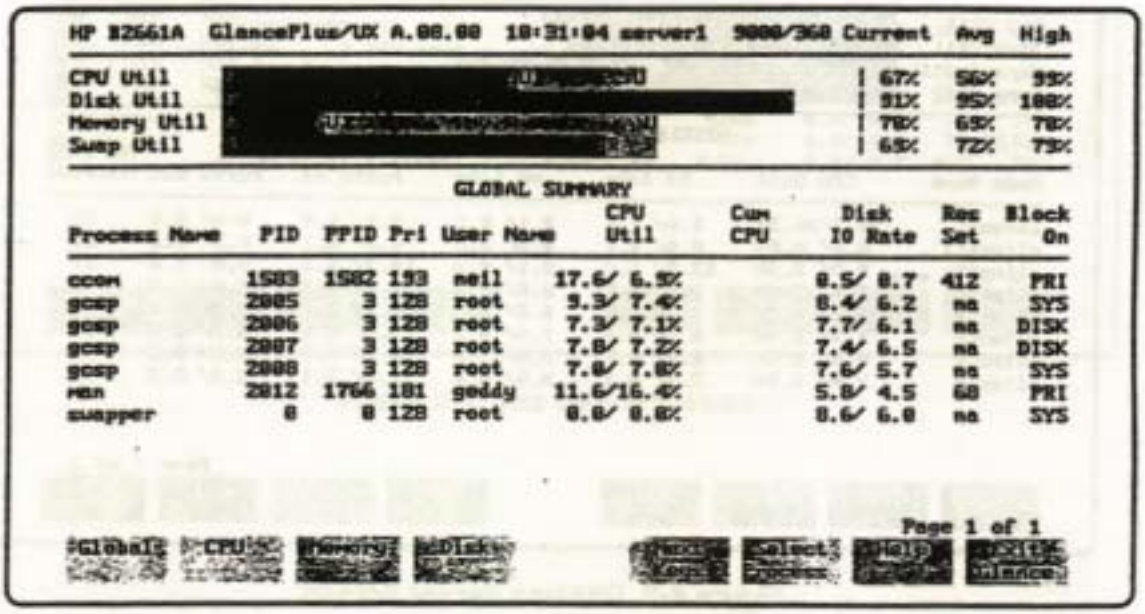


Figure 4-4. The Global Screen at 10:31 a.m.

⑤ Rechecking the Diskless Server Screen

Global Cluster Server Process (**gcsp**)들이 file 들이나 swap space 와 같은 자원들에 대한 client requests 를 만족시키기 위해서 서버에서 구동한다. 그 자체에 있는 것은 정상적이지만, Dave 는 그 요구(demand)가 지속적으로 높은 것 같다고 생각한다. 좀더의 데이터를 얻기 위해서, 그는 또다시 Diskless Server Resource Utilization screen 을 본다.

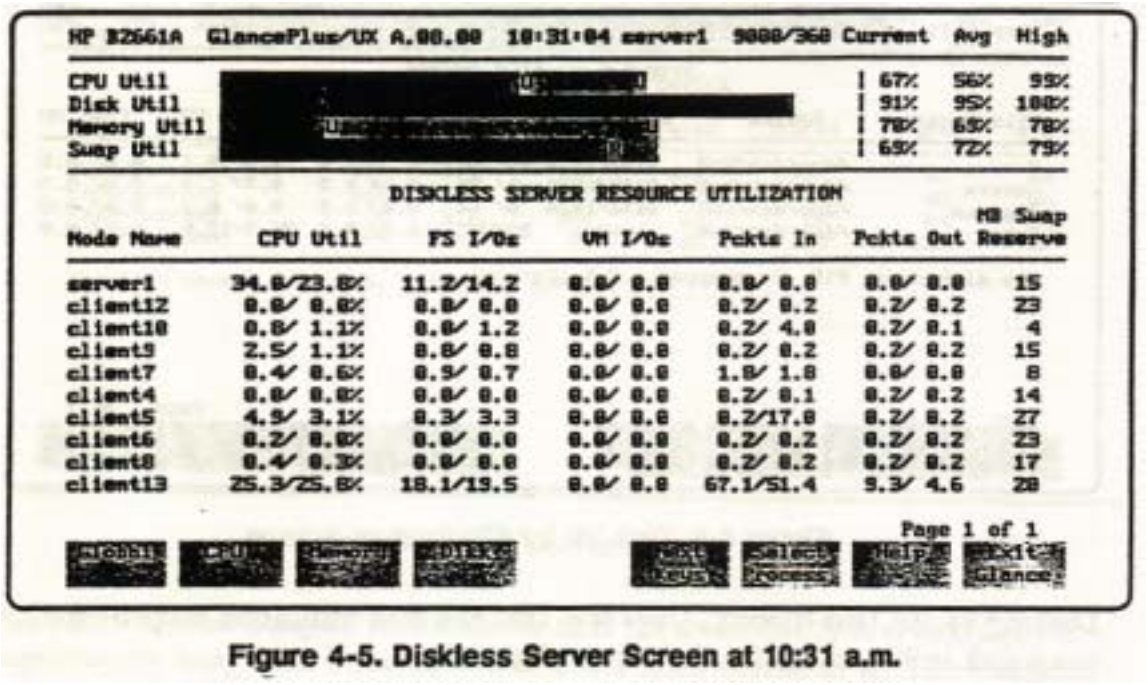


Figure 4-5. Diskless Server Screen at 10:31 a.m.

그가 이 스크린을 관찰할 때, Dave 는 client 13 으로 관심을 옮아간다. Client 13 의 usage 가 매우 매우 높다 – 다른 client 들의 usage 들보다 훨씬 높다. Dave 는 그 서버에 대한 client 13 의 demand 가 떨어지는지를 보기 위해 몇 가지 update 들을 통해 그 스크린을 관찰한다. 떨어지지는 않았고, usage 는 비정상적으로 높게 남아있다.

그 서버의 **Disk Util** bar 는 또한 일반적이지 않게 너무 높은 것 같다.

⑥ Checking the Disk I/O Screen

Dave 는 어떤 disk 가 active 한지를 보기 위해서 Disk I/O by File System screen 을 체크한다. (figure 4-6)

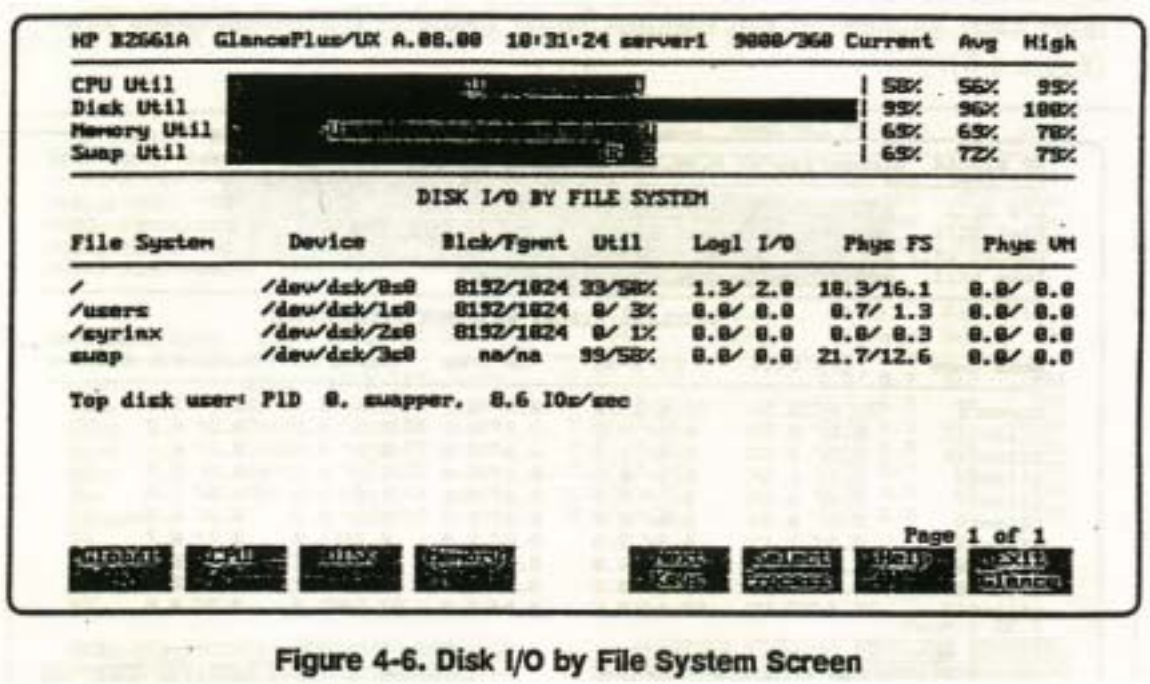


Figure 4-6. Disk I/O by File System Screen

Util column 을 보고서, Dave 는 그의 swap disk 의 disk utilization 통계치가 current interval 에서 99% 에 달한다는 것을 확인한다.

⑦ Reviewing the Disk Queue Lengths

Dave 는 그 swap disk 의 queue lengths 를 체크하기 위해서 Disk Queue Lengths screen 으로 옮겼다 (figure 4-7)

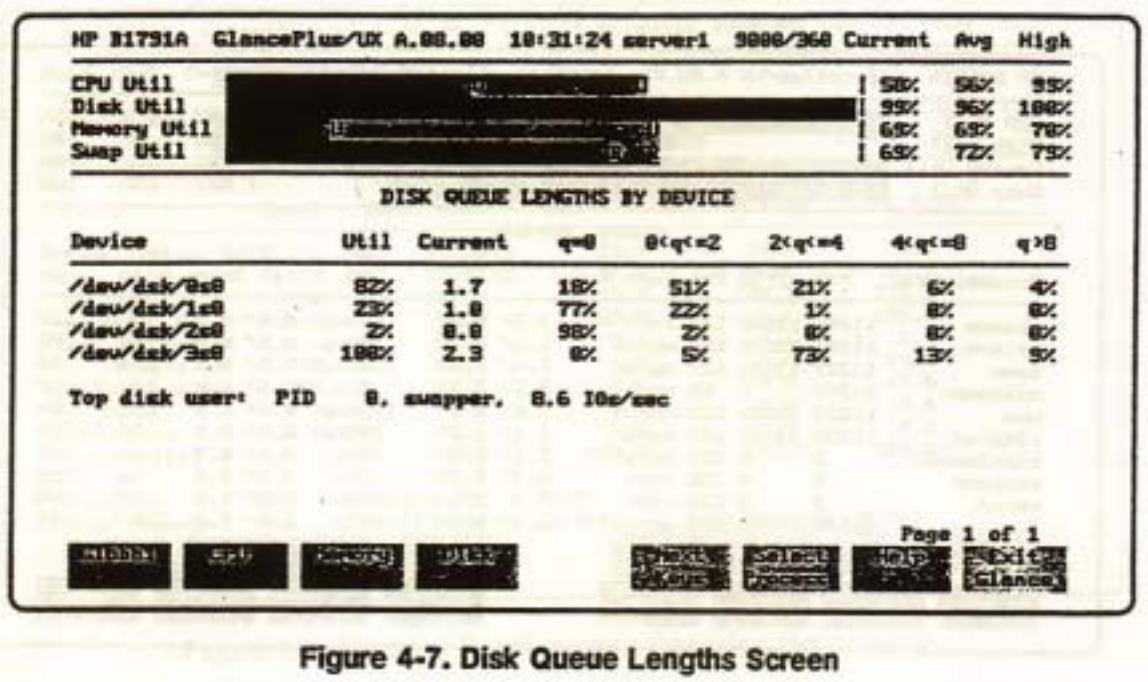


Figure 4-7. Disk Queue Lengths Screen

그 Disk Queue Lengths screen 은 그 swap disk (/dev/dsk/3s0)가 2 와 4 사이의 I/Os queue 사이에 (2<q<=4) 지속적으로 busy 하다는 것을 보여준다. 이것은 Dave 에게 그 swap disk 에 I/O bottleneck 이 있을 수도 있다는 의심을 준다.

⑧ Checking the Client Node

Dave 는 그 diskless node 즉 Client 13 에 대한 GlancePlus 를 구동 시킴으로써 그 bottleneck 의 원인을 찾으려 한다. 그는 그 client 시스템으로 logon 하기 위해 **rlogin** 을 사용한다.

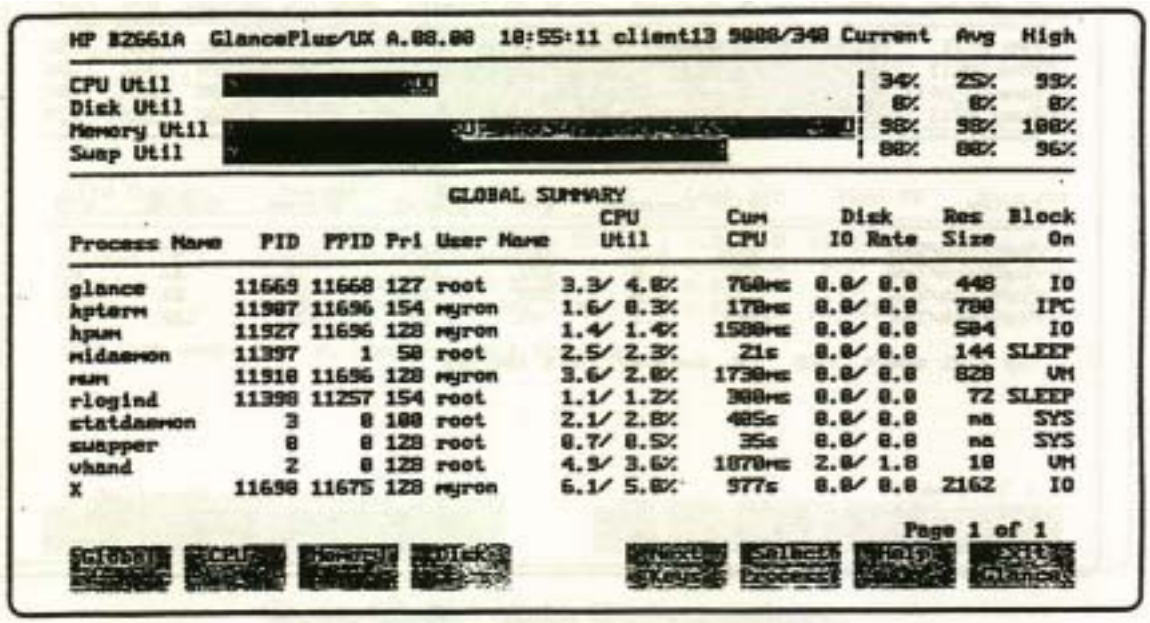


Figure 4-8. Client13 Global Screen

Client 13 은 새로 추가된 diskless workstation node 로, 이 workstation 의 주인인 Myron 이 이 cluster 의 새로운 user 가 된다. Client 13 의 Global screen 을 보고서, Dave 는 그 스크린이 CPU bottleneck 을 지시하진 않지만, memory utilization 은 매우 높다는 것을 알게 된다. Dave 는 이것이 diskless 시스템이므로 disk activity 를 체크하지는 않는다.

⑨ Returning to the Memory Detail Screen

Memory usage 의 비율을 좀더 알기 위해서, Dave 는 Memory Detail screen 을 검토한다. (figure 4-9)

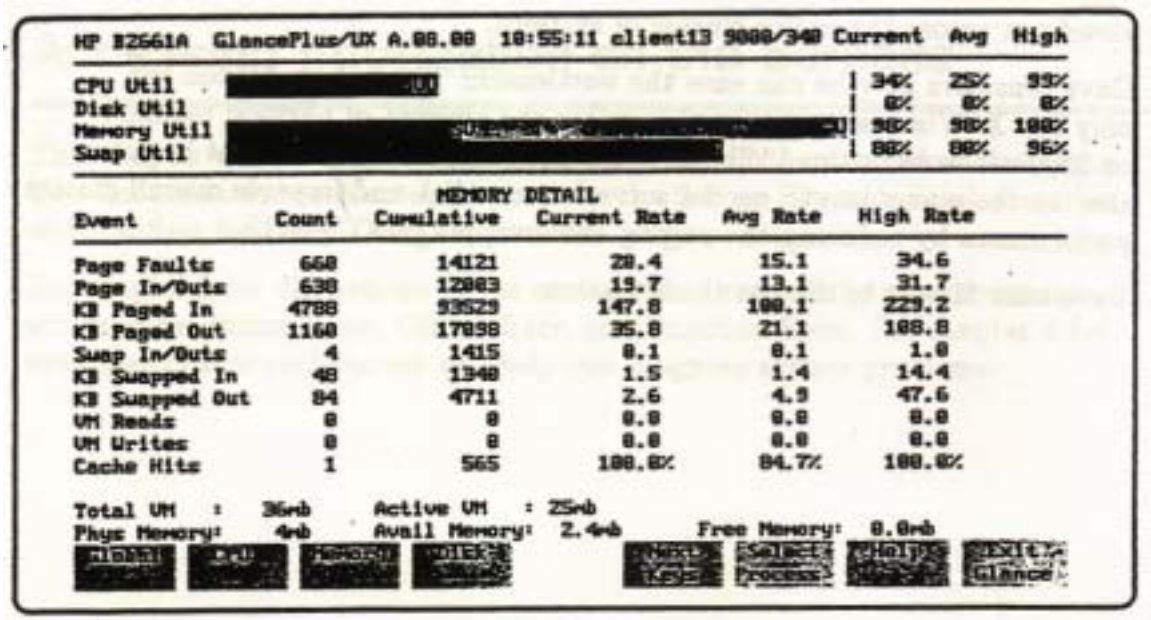


Figure 4-9. Memory Detail Screen Provides the Answer

스크린의 바닥에, Dave 가 찾고자 하는 답 – 즉 Myron 이 그 workstation 에서 단지 4 MB 의 memory 만으로 구동하고 있다는 것을 찾는다.

Global screen 을 다시 빨리 검토하여, 그가 4MB 의 memory 를 가진 **two X Window** managers 를 구동하려 시도하고 있다는 것을 보여준다. (**mwm** on the primary display and **hpwm** in the 340CH's overlay planes)

Note diskless client 시스템에서, client screen 들은 virtual memory reads 와 writes 필드에 단지 0 을 보여줄 것이다. (figure 4-9) 이러한 시스템들의 virtual memory I/O 는 서버에서 일어난다.

---

⑩ Resolving the Problem

paging 과 swapping 을 위한 Memory Detail screen 통계치는 매우 크다 – 이것은 그 client 에 대한 심각한 memory bottleneck 을 지시한다. 이것은 **real** bottleneck 인데 – 이 bottleneck 은 시스템들의 전 cluster 에 대해 높은 swap rate 과 slowdown 으로 이어진다.

Dave 는 어떻게 그 bottleneck 을 완화 할 수 있는지를 고려한다.: Myron 에게 단지 하나의 **X Window manager** 를 돌리게 하던지 Myron 의 workstation 의 physical memory 의 양을 증가하던지 무엇을 결정하던지, Dave 는 그가 Client 13 에 대한 paging 과 swapping 을 줄임으로써 전체적인 cluster 성능을 향상 시킬 수 있고, 그 서버의 swap disk 에 대한 queue length 를 줄일 수 있다는 것을 알고 있다.

Dave 는 Myron 에게 전화를 걸어 이 상황을 논의한다.

5th Accessing Information on the Screens

이 Chapter 는 detailed data screen 들에 대한 metrics 와 statistics 를 묘사한다. 각 스크린에 대한 논의는 Screen Elements display section 에 초점을 맞춘다.

모든 스크린에 같은 것으로 남아 있는 – Banner Line, Global Bars, Function Keys –섹션들에 대한 논의는 3 장에 나와 있다. 각 스크린이 시스템 문제들을 치유하는 도움을 어떻게 줄 것인지에 대한 예들을 보려면 4 장을 참조하면 된다.

A) Global Screen

이 스크린은 넓게 시스템 activity 를 요약하고, usage threshold 값을 초과하는 모든 프로세스들을 목록화 한다. (figure 5-1) Global screen 은 다른 초기 스크린을 규정짓는 startup option 을 선택하지 않는 한 (see chapter 6) Glance 를 시작할 때 나타나는 첫 스크린이다.
이 스크린에 나타난 정보는 자원이 초과적으로 사용되고 있는지, 혹은 프로세스가 이용가능한 자원들을 모니터링 하고 있는 지이다. 이 스크린은 시스템 activity 와 성능을 검토하기 위한 일반적인 starting point 이다.

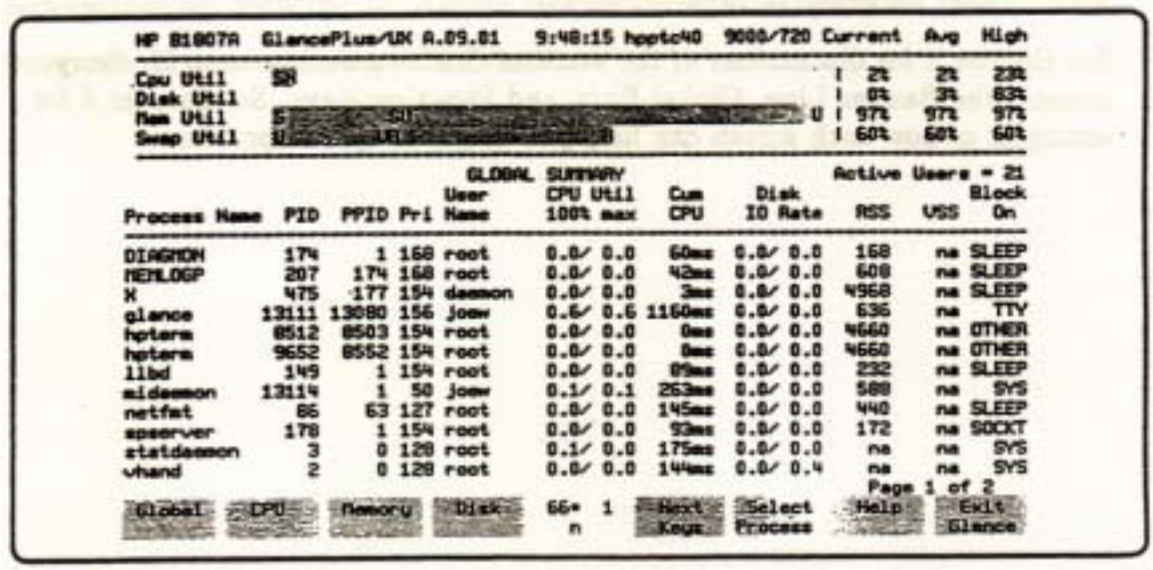


Figure 5-1. The Global Screen

여러분들은 g 나 Global 을 누름으로써 다른 스크린으로부터 Global screen 으로 접속할 수 있다.

1. Process Summary Section

한 줄의 data 는 현(current) threshold 한계를 초과하는 각 프로세스를 나타낸다. (figure 5-2) 이 목록화된 프로세스들은 process threshold 세팅에 따라 정렬되어 있다. 한 page 에 목록들을 한계지울 수 있을 만큼 충분히 높게 threshold 값을 세팅하도록 권유된다. 이것은 대부분의 자원들을 사용하는 프로세스들에 초점을 맞추는 것을 쉽게 한다.

만약 threshold 한계치 (limit)를 초과하는 프로세스들의 수가 한 스크린 이상이면, message reading page 1 of …. 이 그 스크린의 오른쪽 구석에 나타날 것이다. f 그리고 b , 또는 만약 이용가능하면 Page Forward 와 Page Back 을 이용하여 page 들 사이 앞 뒤로 이동할 수 있다.

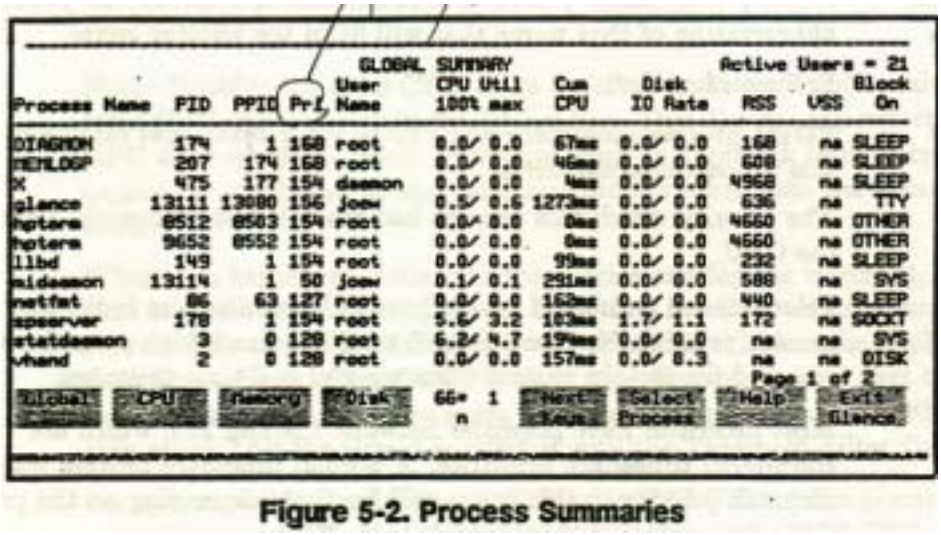


Figure 5-2. Process Summaries

① Column Definitions

각 프로세스의 Statistics(통계치)에 대한 9 개의 column 이 스크린의 왼쪽에 목록화 되어 있다.

Column	Definition
Process Name	이 프로세스의 수행가능한 code 를 load 하기 위해 사용되는 이름 (“exec name"), 혹은 allotted space 의 적합한 그 이름의 약자(abbreviation)
PID	Process identifier
PPID	Parent process identifier – 이 프로세스를 처음으로 fork 시킨 프로세스의 PID
Pri	CPU 에 대한 control 을 마지막으로 그만 두었을 때 이 프로세스가 가지는 priority

Priority 는 더 높은 우선순위를 지칭하는 낮은 숫자를 가지고 숫자화 된다.
0 부터 127 까지는 특정 시스템 daemon 들이나 real-time process 들을 위해 예약된 high priority 들이다.
대부분의 프로세스들은 **timeshare** priority 라 고려되어지는 128 에서 255 사이의 priority 를 가진다. 이러한 범위의 dispatch priority 를 가진 정상적인 timeshare 프로세스들은 프로세스의 CPU demand 와 시스템의 load 에 따라 달라진다. **nice** priority 에서 수행하는 프로세스들은 이 column 에서 전형적으로 더 높은 숫자를 가진다.

User Name 이 프로세스를 시작 시킨 user 의 로그인 이름

Column	Definition
CPU Util (100% max)	**current/average** format 으로 보여지는 **CPU usage** 예를 들어, figure 5-2 에서 매 5 초의 default update interval 이 사용되었다. 이 column 의 **midaemon** 을 보면 current interval 의 .1%을 보여준다. 두번째 숫자 .1%는 **midaemon** 프로세스에 의한 average CPU usage 를 반영한다. **Note:** 일반적으로 모든 프로세스들에 대한 전체 CPU usage 는 Global Bar 에서 보여지는 전체 CPU usage 와 동일하지 않다. 이런 어긋남은 kernel 그 자체 보다는 다른 특정한 프로세스들에 attributed 된 (hardclock interrupt 과 같은) 시스템 interrupt 에 소비되는 CPU time 을 반영한 것이다. interrupt 를 유발하는 그 프로세스를 결정할 수 있는(can) kernel 이 있는 곳에서, 시간(time)이 그 프로세스이 탓이 된다. 그 프로세스를 목록화한 CPU time 은 그 프로세스에 귀착된 time 만을 반영하고, 반면에 Global CPU Bar 는 그 시간이 귀결될 수 (attributed) 있는지에 관계 없이 모든(all) CPU time 을 반영한다. 따라서 "All Processes"가 Threshold Filter Options 에 의해 선택됐을 때라도 서로 동일하지 않을 것이다. 하나 이상의 CPU 를 가진 시스템에서, CPU Util column 은 100%로 일반화되지

않는다. 그것은 전체 CPU 역량(capacity)의 percentage 를 보여줄 것이다. 예들 들어 두개의 CPU 를 가진 시스템에서, CPU Util column 의 최대치는 200%가 될 것이다. 만약 CPU 의 숫자가 4 개면 이 column 의 최대치는 400%가 된다.

Cum CPU 그 프로세스가 fork 되거나 statistics 가 reset 된 이래, 이 프로세스에 의해 누적된 (cumulative) CPU time 이 사용된다.

Column	Definition
Disk IO Rate	current/average format 으로 보여지는 physical disk I/Os 의 비율 (rate) 예를 들어, 프로세스 **spserver** 의 last interval 동안 초당 1.7 의 physical disk I/O 와, midaemon 이 시작되거나 statistics 가 reset 된 이후 매초 당 1.1 의 physical access 들의 평균치를 기록했다. (figure 5-2)
RSS	Resident Set Size, 즉 이 프로세스가 현재 사용하고 있는 physical RAM 의 kilobytes 이다. 이것은 library segments 에 있는 shared memory 뿐만 아니라 그 프로세스의 data, stack, 그리고 text segments 로 할당된 memory 를 포함한다. 비록 shared regions 의 전체 size 가 그것을 참조하는 각 프로세스로 목록화 되어 있을지라도, 그 region 은 단지 한번 physical 하게 memory 에 있다. **statdaemon** process (figure 5-2)와 같은 몇몇의 HP-UX 시스템 daemon 들은 kernel memory 를 사용하고 메모리의 많은 부분을 할당하지 않는다. 이러한 daemon 들은 이 필드에서 **na** (not applicable)를 전시한다.
VSS	이 프로세스가 현재 사용하고 있는 disk 공간의 kilobytes. 이것은 그 프로세스의 data, stack, shard, 그리고 text segments 를 위해 할당된 disk space 를 포함한다.

Block On	마지막으로 기록된 process block 의 이유(reason). 예를들어, process **vhand** 는 DISK 를 기다리고 있다 – 그것은 완성하기 위하여 Disk I/O operation 을 기다리고 있다. (figure 5-2) **Note:** 마지막 update interval 동안 새롭게 형성된 프로세스들은 이 column 에 **new** 를 가지고 있다. 마지막 interval 동안 끝난(terminated) 프로세스들은 **died** 를 가진다.

② Block On Reasons

여기에 Block On 의 reason 들이 목록화 되어 있다.

Block Indicator	Reason for Block
CACHE	프로세스가 **getblk** 나 **getnewbuf** 와 같은 memory buffer cache operation 을 기다리면서 blocked 된다.
DISK	(read, write, 혹은 bufffer access, control 과 같은) disk request 를 기다리면서 blocked 된다.
DUX	(read, write, 혹은 bufffer access, control 과 같은) diskless operation 을 기다리면서 blocked 된다.
INODE	system inode request 를 기다리면서 blocked 된다.
IO	(HIL, SRM, VME, 혹은 GPIO 와 같은) non-disk I/O requests 를 기다리거나, disk (physio)쪽으로나 부터의 raw I/O 를 기다리면서 blocked 된다.
IPC	shmat 과 같은 shared-memory control operations 을 기다리면서 blocked 된다.
LAN	LAN hardware card request 를 기다리면서 blocked 된다.
MBUF	memory buffer request (이경우 buffer 는 inbound 와 outbound cluster traffic 을 위해 사용되는) 를 기다리면서 blocked 된다.
MESG	**msgrcv, msgsnd** 와 같은 message operation 을 기다리면서 blocked 된다.
NFS	(read, write, 혹은 control 과 같은) network file system 을 기다리면서 blocked 된다.
PIPE	pipe operation 을 기다리면서 blocked 된다.
PRI	CPU 를 사용하는 다른 프로세스 때문에 프로세스가 blocked 되는 것을 말한다. 이것은 다른 프로세스가 더 높은 우선순위를 가지거나 현재 프로세스의 이용가능한 time slice quantum 이 고갈될 때 일어난다.

RFA	read 와 같은 remote file access request 를 기다리면서 일어난다.

Continuus……

Block Indicator	Reason for Block
SEM	프로세스가 **semop** 와 같은 semaphore operation 을 기다리면서 blocked 된다.
SLEEP	sleep 이나 wait call 을 기다리면서 blocked 된다.
SOCKT	(connect 나 send 와 같은) socket operation 을 기다리면서 blocked 된다.
SYS	이용가능하게 되기 위해서 (audit, security, 혹 page control 과 같은) 일반적인 kernel resources 를 기다리면서 blocked 된다.
TERM	(read, write, 혹은 control 과 같은) terminal (tty or pty) request 를 기다리면서 blocked 된다.
VM	(page in, page out, 그리고 memory lock 과 같은) virtual memory operation 을 기다리면서 blocked 된다.
OTHER	위에서 언급된 것과 관련이 없는 자원들을 기다리면서 blocked 된다.

B) CPU Detail Screen

이 스크린은 CPU time 이 어떻게 다른 상태 (states or activities)에 할당되는지를 보여준다. 이 분석은 시스템 time 이 초과되고 있는지, Context switches 의 비율(rate)이 일어나고 있는지, 그리고 CPU time 을 기다리고 있는 프로세스의 평균 수를 보여준다.

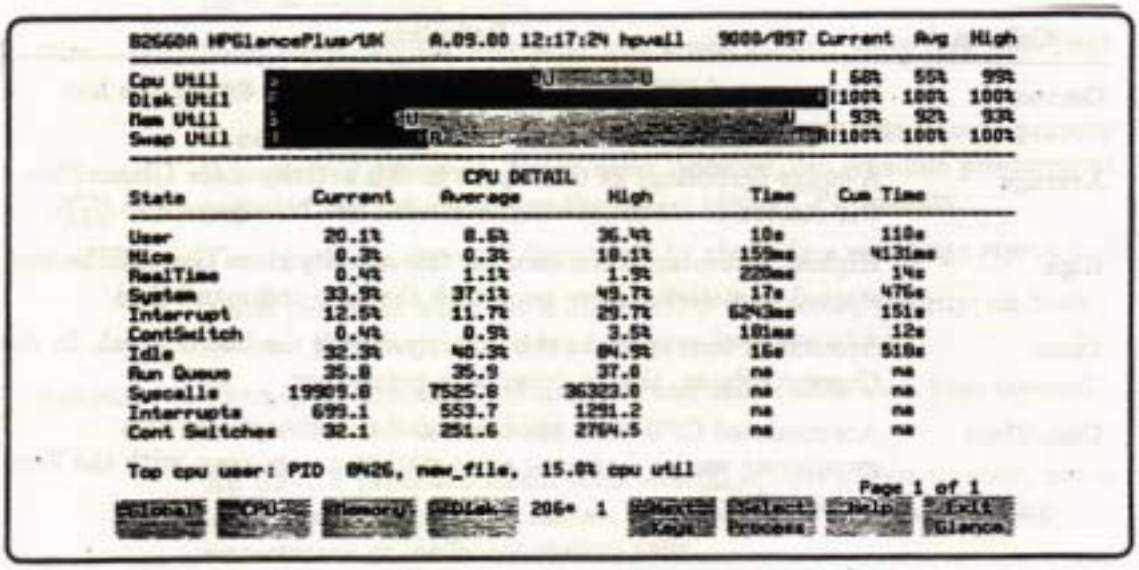


Figure 5-3. The CPU Detail Screen

어느 화면에서나 c 나 CPU key 를 이용하여 이 스크린에 접속할 수 있다.

1. Screen Elements

① Time Statistics for Each Activity

Detail Display section 의 위에 있는 column headings 은 왼쪽에 목록화 되어 있는 5 가지의 CPU states 나 activities 에 소비되는 시간의 percentage 를 지시한다.

Column	Definition
Current	last interval 동안 이 activity 에 전념된 CPU time 의 percentage.
Average	GlancePlus 가 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 이 activity 에 소비된 평균 percentage.
High	GlancePlus 가 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 이 activity 에 사용된 최고치의 percentage time
Time	last interval 동안 이 activity 에 소비된 시간의 양. Current column 에서는 percentage 로 나타난다.
Cum Time	GlancePlus 가 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후

이 activity 에 할당된 축적된 CPU time

② CPU States or Activities

CPU time 은 아래의 목록에서 정의 된 것처럼 7 가지 목적으로 사용된다.

State	Definition
User	정상적인 우선순위(priority)가 적용될 때 user program code 를 수행하는데 소비되는 시간. User time 의 나머지는 Nice CPU 나 Real-Time CPU 아래로 계산된다.
Nice	**nice** priority 값으로 user code 를 구동하는데 소비되는 시간 (*see* man-page **nice (1)**) 일반적으로 nice value 는 양수이고, 그것은 일반적인 프로세스들보다도 더 낮은 priority 를 준다. 하지만 super-user 들은 그 code 에 더 높은 CPU priority 를 주기위해 음수의 nice value 로 프로세스들을 스케줄링 할 수 있다.
Real Time	**rtprio** 에 의해서 dispatch 된 프로세스들에 주어진 시간 (*see* man-page ***riprio (1)***) 이러한 프로세스들은 더 높은 CPU dispatching priority 를 가진다.:busy 시스템에서 그것들은 다른 프로세스들이 수행하는 것을 막는다.
System	(ContSwitch 와 Idle 아래로 cover 된 시간 이라기 보다) 시스템 call 과 code 를 수행하는데 소비되는 시간 모든 HP-UX 프로세스들은 system code 에 몇몇의 수행시간 (execution time)을 소비하지만, 큰 System value 는 불필요하거나 비효율적인 시스템 call 을 만드는 프로그램들을 가진 문제의 지칭일 수도 있다.
Interrupt	code 를 다루는 시스템 interrupt 를 수행하는데 소비된 축적된 시간. 높은 Interrupt rate 은 높은 I/O rate 을 가진 시스템에서 일어난다. 비정상적이게 높은 Interrupt rate 은 hardware 문제를 지칭하는 것일 수도 있다.
ContSwitch	프로세스들 사이에 context switching 을 하는데 소비되는 시간 이것은 time allotment 가 만기된 후에 구동하는 프로그램이 CPU 를 포기해야만 할 것인지 어떤지를 고려하는데 소비되는 시간을 포함한다. 이것은 idle loop 에서 소비되는 시간이나 다음에 구동될 프로세스가 어느 것인지를 결정하기 위해서 시스템에 의해서 사용된 시간을 포함하지는 않는다.
Idle	시스템이 프로세스 없이 idle 한 상태로 있던 시간과 다음에 구동될 프로세스를 선택하는데 (the function of the "idle loop") 소비되는 시간의 합. 초과적인

idle time 은 비효율적인 CPU power 의 사용을, zero 의 idle time 은 CPU bottleneck 상황을 지시한다.

③ Run Queue Length

Screen Detail screen 의 다음 줄은 시스템의 run queue length, 혹은 load average 에 대한 정보를 포함한다. 아래에 applicable column 들이 정의 되어 있다.

Column	Definition
Current	last interval 동안 CPU time (average run queue)를 기다리는 수행가능한 프로세스들의 평균 수
Average	모니터링을 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 수행가능한 프로세스들의 평균 수 (average run queue length)
High	모니터링을 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 수행가능한 프로세스들의 가장 높은 수 (highest run queue length)

④ Activity Rates

다음, 중요한 CPU activity 들에 대한 rate 들이 스크린에 나와 있다.: System Calls, Interrupts, 그리고 Context Switches.

Activity	Definition
SysCalls	모든 프로세스들에 의해 만들어진 system call 들의 rate. Example: 만약 5 초의 update interval 에 걸쳐 SysCall rate 이 2.0 이라면, last screen update 이후 일어난 system call 은 10 이다. (10 / 5 = 2)
Interrupts	전 시스템에 대한 device interrupt 들의 rate
Cont Switches	CPU 가 한 프로세스를 구동시키는 것으로부터 다른 것으로 변화시킬 때의 context switch 들의 rate

⑤ Top CPU User

Current interval 에 가장 많은 CPU time 을 소비한 프로세스는 그 스크린의 바닥에 list 되어 있다.

C) Memory Detail Screen

이 스크린은 page faults, paged-in 되고 paged-out 된 page 들의 수,그리고 virtual memory 로의 접속 (access) 수와 같은 memory management event 들에 대한 통계치를 제공한다. (figure 5-4)

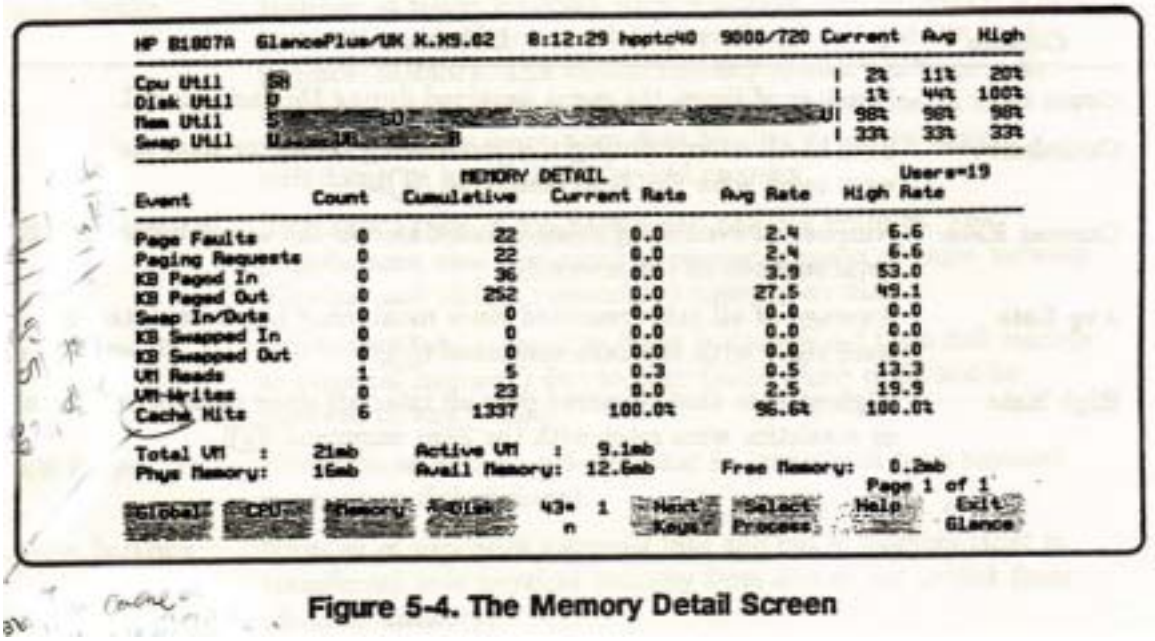


Figure 5-4. The Memory Detail Screen

여러분들은 m 이나 Memory key 를 이용하여 어느 스크린에서나 이 스크린으로 접속할 수 있다.

1. Screen Elements

5 개의 다른 metric headings 들이 그 섹션의 위에 Memory management 에 관련된 각 타입의 event 로 보여진다.

① Memory Statistics

Column	Definition
Count	last interval 동안 event 가 발생한 횟수의 수
Cumulative	모니터링이 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후의

	모든 count 들의 합
Current Rate	초당 event 들의 수 (interval 내에서 전체 초 (시간)에 의해 나누어진 count 로 축적된)
Avg Rate	모니터링이 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 기록된 모든 rate 의 평균
High Rate	모니터링이 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 모든 interval 들에 걸쳐 발생된 가장 높은 rate.

② Memory Events Measured

주어진 counts 와 rates 은 스크린의 왼쪽에 목록화 되어있는 그 타입의 memory events 위한 것이다. 이러한 event 들은 아래 정의되어 있다.

Event	Definition
Page Faults	프로세스가 code 지시나 참조를 수행하려 할 때 기록된 fault 의 수. Data page 는 physical memory 에 있지 않다. 그 virtual memory 시스템은 수행을 지속하기 위해서 missing code 나 data 를 page-in 할 것이다. 큰 (large) paging rate 은 빈약한 data locality, 지나친 context switching, 혹은 충분치 못한 physical memory 를 말한다.
Paging Requests	일상적인 pagein 과 pageout call 들의 수. **pagedaemon** 은 physical 과 virtual (secondary) memory 사이에 page 들의 전송을 요구하기 위해서 이 count 를 이용한다.
KB Paged In	어떤 cache 로도 해결될수 없는 page fault 때문에 paged in (즉 disk storage 로부터 physical memory 로 전송된) 된 data 의 kilobytes
KB Paged Out	paged-out 된 (즉 physical memory 로 부터 disk storage 로) data 의 kilobytes
Swap In/Outs	swapped in 되고 out 된 (즉 disk 로부터 physical memory 로, 또 memory 로 부터 disk 로 이동된) 프로세스들의 수 이것은 발생할 수 있는 memory bottleneck 의 훌륭한 지침자이다. 만약 시스템이 workload 를 효율적으로 수행하도록 필요한 memory 를 부족해 한다면, 많은 프로세스들이 swapped in 과 swap out 할 것이다. (be swapped into and out of memory) 이것은 **thrashing** 에 이를 수도 있는데, 시스템이 유용한 작업을 하는데 시간을 보내지 않고 swapping 하는데 너무 많은 시간을 소비한다는 것이다.

KB Swapped In	Swapped in 된 data 의 kilobytes. 이것은 memory pressure 때문에 swapped out 된 프로세스들이 수행해야만 할 때 발생하고, 따라서 memory 로 되돌아와져야 한다.
KB Swapped Out	memory pressure 때문에 프로세스들이 disk 로 옮겨질때 swapped out 된 data 의 kilobytes.

Event	Definition
VM Reads	virtual memory 서비스를 위해서 모든 disk drive 들로부터의 physical reads 의 총 count. 이것은 page fault 가 nonmemory-resident file 에 있는 data 를 참조할때 일어난다. Diskless client 시스템에서 이 필드는 항상 0 이다. 왜냐하면 VM read 혹은 write 에 의해 발생하는 physical I/O 는 서버에서만 일어나기 때문이다.
VM Writes	모든 disk drive 로의 virtual memory 에 있는 file page 들의 physical writes 의 총 count 이것은 다른 demand 를 위한 공간을 만들기 위해 file page 가 physical memory 로부터 disk 로 옮겨질때 발생한다.
Cache Hits	buffer cache 에 이미 있는 data 에 의해 만족된 read-ahead requests 와 buffterd read 의 rate. 이 event 의 통계치는 cache 효율을 반영한다. 효율적인 cache hit rate (높은 percentage) 는 file system 을 대신하는 disk access 가 거의 없고, CPU resource 의 더 좋은 사용을 의미한다. 하지만 high hit ratio 는 또한 file system buffer cache 용으로 구성된 memory 양의 너무 많은 초과의 결과 일 수 도 있다. 이것은 이용가능한 physical memory 를 줄일 수 있고, VM system 에 의해 초과의 paging 과 swapping 을 초래할 수도 있다. 따라서, 만약 file system buffer cache 에서 high hit ratio 를 발견하면, VM system 에서 넘치는 (excessive) activity 인지 체크하면 된다.

③ Status of Available Memory

스크린의 바닥에 보이는 5 가지의 값은 memory allocation 의 현 상태를 묘사한다. 아래를 참조하면

Memory Type	Definition
Total VM	시스템에서 구동하는 모든 프로세스들의 개인 데이터 (private data)로 할당된

virtual memory 의 양. 이것들 모두가 current physical memory 는 아니다. 이러한 metrics 는 shared memory, text, 그리고 library segments 와 같은 shard data 를 포함하지 않는다.

Active VM active 하게 프로세스들을 수행하므로써 사용된 virtual memory 의 양. shared memory, text, 그리고 library segments 와 같은 shard data 는 이 active VM metrics 에 포함되지 않는다.

Phys Memory 모든 작업에 이용가능한 physical memory 의 양. 이것은 모든 hardware memory board 의 전체 용량과 동일하다.

Avail Memory kernel table 과 resident buffer 용으로 시스템에 의해 reserved 되지 않은 physical memory 의 양. 이것은 모든 user 프로세스들에게 할당용으로 이용가능한 전체 memory 이다.

Free Memory 어떤 프로세스나 구조에 현재 할당되지 않은 physical memory 의 양. 이 memory 는 새로운 프로세스들에 의해서 이용가능하다. 이 값이 낮아질 때, swapping 이 발생하기 쉽다.

D) Disk Detail Screen

이 스크린은 모든 마운트된 disk device 의 logical 하고 physical 한 I/O request 를 보여주고, User, Virtual Memory, System, 그리고 Raw request 로서의 physical request 들을 분류한다. (figure 5-5) 이 스크린은 또한 inbound 와 outbound 된 network file system (NFS) activity 를 보여준다.

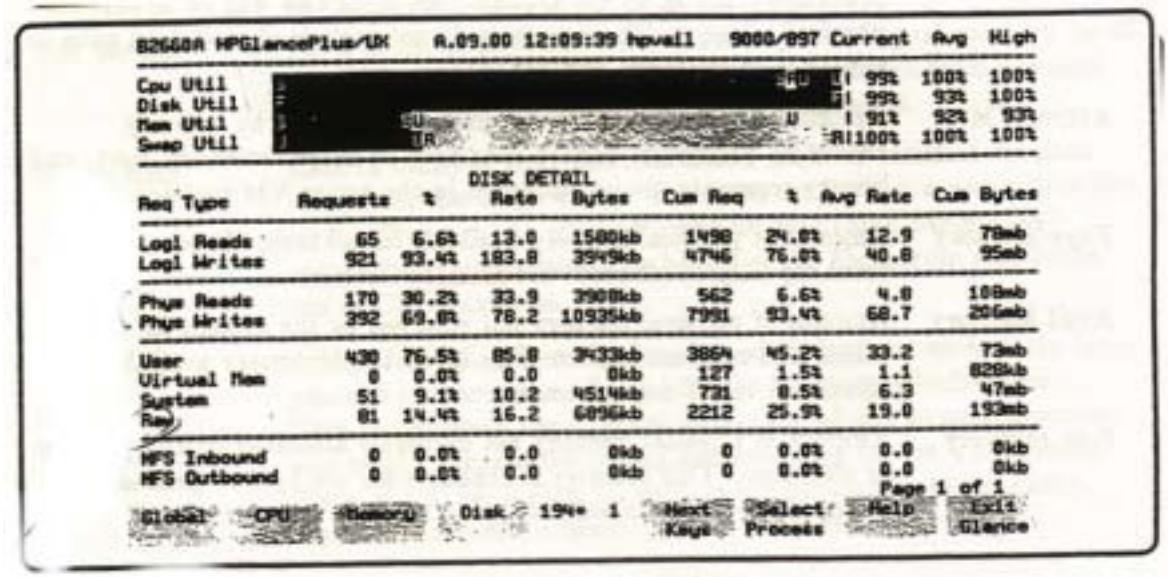


Figure 5-5. The Disk Detail Screen

여러분들은 d 나 **Disk** key 를 이용하여 어느 스크린으로부터도 이 스크린으로 이동할 수 있다.

1. Screen Elements

각 request type (Req Type)의 통계치 (statistics)는 두 셑 (set)의 네 column 으로 나타나는데 첫번째 set 은 아래에 나와 있는 데로 이다.

① Statistics Displayed

Column	Definition
Requests	last interval 동안의 request 의 수

%	같은 그룹에서 다른 타입들과 비교한 이러한 request 타입의 percentage Example: 그림 5-5 에서, last interval 동안 170 의 physical read request 가 일어났는데, 이것은 전체 physical read/write request 의 30.2 %를 나타낸다.; 또 physical write request 는 392 번인데 전체 physical rade/wirte 의 69.8 %를 나타낸다.
Rate	초당 평균 request 의 수 Example: 그림 5-5 에서, last interval 동안 Phys Reads 의 rate 은 33.9 이고, Phys Writes 에 대해서는 78.2 이다.
Bytes	current update interval 동안 모든 requests 를 통해서 보내진 data 의 양

그 스크린의 오른쪽에, GlancePlus 가 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 모든 것(*all*)에 걸쳐 축적된 data 를 보여준 두번째 4 개의 column (the second four column)이 있다.

② Disk Requests Measured

왼쪽에 목록화된 request type (Req Type)들은 reads 와 write, disk activity, 그리고 network file system (NFS) activity 를 참조한다.

Request Type	Definition
Logl Reads	그 시스템의 call level 에서 측정된, NFS-mounted disks 를 포함한 모든 disk 들로(from) 부터의 logical reads 의 수. 모든 logical reads 는 disk 로의 physical access 로 결론지어지지 않는다.:읽혀지는(read) data 는 memory buffer cache 에 있게 된다. 거꾸로, 모든 physical reads 가 logical reads 로 부합되지는 않는다. 왜냐하면, swapping 과 같은 몇 가지 I/O 들은 그 file system 을 우회하기 때문이다.
Logl Writes	그 시스템의 call level 에서 측정된, NFS-mounted disks 를 포함한 모든 disk 들로의(to) logical writes 의 수. Logical writes 은 그 data 가 즉시 disk 에 쓰여져야만 하지 않는 한 Physical I/O 로 결론지어지지 않는다.
Phys Reads	driver level 에서 측정된 모든 disk 들로부터 (from) physical reads 의 수. Physical I/O 는 file system 때문에 logical read activity, virtual memory management, 혹은 다른 system activity 가 될 수 있다.
Phys Writes	driver level 에서 측정된 모든 disk 들로의(to) physical writes 의 수. Physical reads 처럼, writes 도 다른 종류의 activities 로부터 결론 지어진다.

③ Disk Usage by Source

아래 data Reads 와 Writes 의 네 가지는 그것의 근원(source)에 바탕을 둔 disk activity 에 통계치를 기록한다.

Source	Type of Usage
User	user file I/O operation 들 때문에 생긴 physical disk I/O 들의 수
Virtual Mem	시스템의 virtual space management function (paging 과 같은) 들에 의해 생긴 physical disk I/O 들의 수
System	시스템 activity 들에 의해 생긴 physical disk I/O 들의 수 – 예를 들면, inode 나 superblock 들로의 I/O 들
Raw	direct- 나 raw-mode physical disk I/O 들의 수 (file system 을 우회하는 것들)

④ NFS Activity Measured

스크린의 밑쪽에 있는 두 줄은 request 의 수, 다른 타입의 request 들의 percentage, 각 초(second or rate)당 request 들의 평균 수, 그리고 전송된(tranceferred) data 의 양 (byte 단위로) 등을 포함한 network file system (NFS)에 관한 통계치를 보여준다.

Activity	Source
NFS Inbound	NFS-mounted 된 local system disk 들을 가진 remote 장비들에, SysCall reads 혹은 writes 때문에 생긴 network file system activity. 이것은 physical 한 측정이고 직접적으로 logical read 와 write 으로 부합되지 않는다.
NFS Outbound	NFS-mounted 된 remote 시스템에 디스크들을 가진 local 시스템에, SysCall read 혹은 writes 때문에 생긴 network file system 의 physical 한 activity.

E) Disk I/O by File System Screen

이 스크린은 각 file system 혹은 mounted-disk partition 의 I/O rates 에 대한 상세 정보를 제공한다. (figure 5-6) 이런 정보는 disk load balancing 에 유용하다.

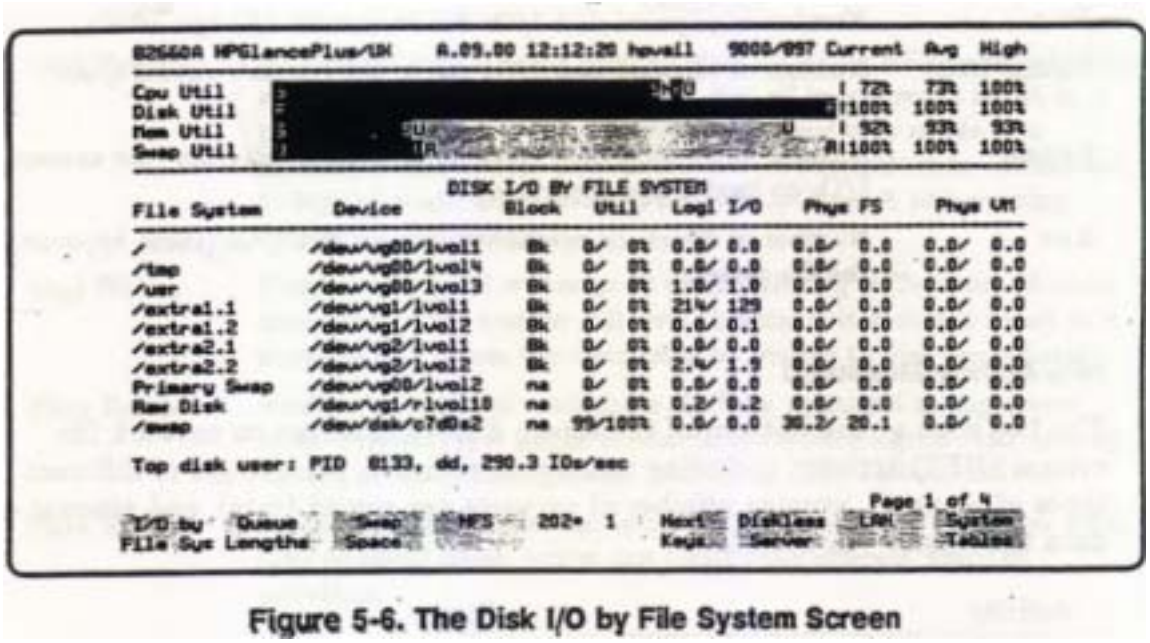


Figure 5-6. The Disk I/O by File System Screen

여러분들은 i 나 I/O by File Sys key 를 이용하여 어느 스크린에서나 이 스크린으로 접속할 수 있다.

1. Screen Elements

첫번째 두 column 은, File System 과 Device, 각 줄에 있는 disk partition 설명을 분류한다. 나머지 column 은 그 partition 들에 특정 정보를 제공한다.

이런 분류된 정보는 kernel File System 구조에서 모아지고, **/etc/mnttab** 과 **/etc/checklist** file 로부터 만들어진다.

800 시리즈 시스템에서, **/etc/checklist** 에 entry 를 가지지 않은 primary swap 은 **File System** column 에서 primary swap 으로 구분되고, **Device** column 에서는 major/minor device code 를 보여 줄것이다.

/etc/checklist 에 목록화 되지 않거나 마운트 되지 않은 다른 active disk device 들은 major/minor device code 에 의해 구분 될 것이다.

① Statistics Measured

각 partition 을 위해 제공된 통계치는 아래에 정의 되어있다.

Column	Definition
File System	이 disk partition 의 file system 위치
Device	이 disk partition 의 physical disk 위치 NFS mounted disk 에서, 시스템 이름이 처음에 나오고, 다음에 device 이름이 나온다.: **system1:/dev/dsk3s0**
Block	이 partition 의 block 크기. Block size 는 file system 이 생성될 때 file system size set 용으로 사용된 정보의 첫번째 unit 이다.
Util	current/average 포맷으로 보여진 이 partition 의 이용율(utilization) 이것은 last interval 동안 I/O 들이 이 file system 에 대하여 쌓였던 (queued) 시간의 percentage 를 반영한다.
Logl I/O	current/average 포맷으로 보여진 이 partition 의 logical I/O rate. 모든 file 의 reads 와 writes 는 비록 physical disk I/O 들이 일어나지 않았을 지라도 logical I/O 들을 생성한다. 예를들면, 만약 그 data 가 buffer cache 에 있다면 logical I/O 가 일어나지만, 어떤 data 도 disk 로 부터 읽혀(read)지거나 쓰여(write)지지 않는다. 반대로 swapping 과 같은 logical I/O 에 대응 없이 physical I/O 가 일어날 수 있다.
Phys FS	Physical I/O 들이 user programs, system I/Os, 그리고 raw I/Os (virtual memory I/Os 을 제외한 모든것)을 대신하여 일어난다. data 는 current/average 포맷으로 보여진다.
Phys VM	current/average 포맷으로 보여지는 virtual memory physical I/Os (paging)의 rates.

② Top Disk User

Current interval 에 걸쳐 가장 높은 disk I/O rate 을 가졌던 프로세스는 그 스크린의 바닥에 나타나 있다.

F) Disk Queue Lengths Screen

이 스크린은 disk 들이 얼마나 잘 I/O request 들을 처리하는지를 보여준다.(figure 5-7) disk queue 의 길이(length)는 처리되기(processed) 위해 기다리는 I/O request 의 수로 측정된다. 그 queue length 는 device 가 그 요구를 맞출 수 있을 만큼 빨리 I/O request 를 처리할 수 없을 때 증가한다.

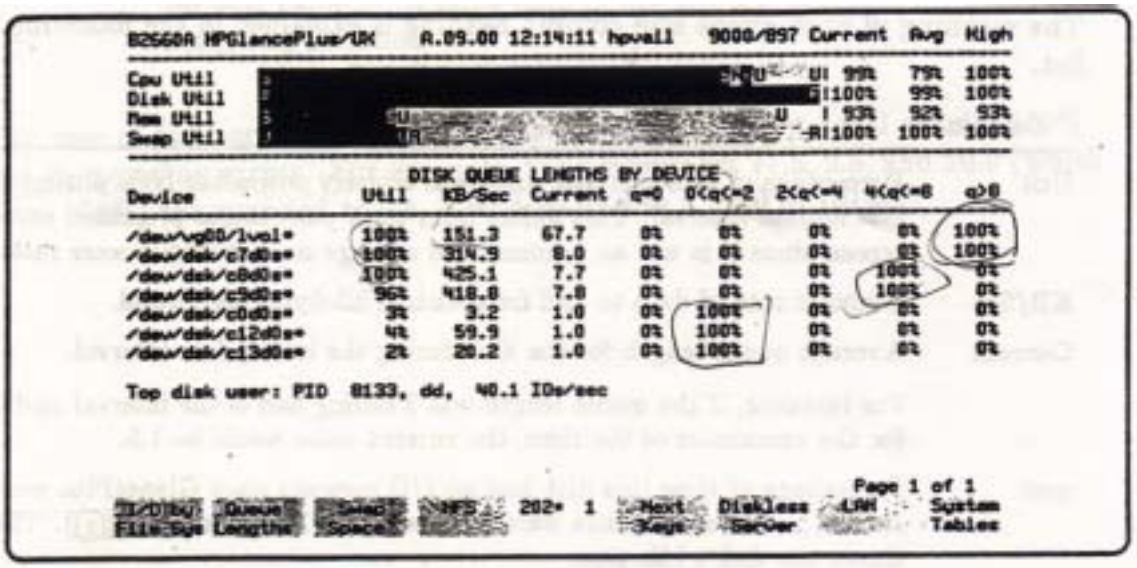


Figure 5-7. The Disk Queue Lengths Screen

여러분들은 어느 스크린에서나 u 나 Queue Lengths key 를 이용하여 이 스크린에 접속할 수 있다.

1. Screen Elements

이 스크린은 현재 마운트된 모든 disk device 를 보여준다. 각 disk device 의 줄은 현 사용율(utilization), queue length, 그리고 queue length 를 각 size queue 에 소비되는 시간과 비교하는 percentage 들을 보여준다.

① Queue Lengths Measured

각 queue size column heading 의 의미는 아래에 설명되어 있다.

Column	Definition

Util	last update interval 동안에 disk drive 가 I/O 들을 active 하게 처리하는 시간의 percentage. 이것은 통계치가 reset 된 이후 축적된 average 가 아니기(*not*) 때문에 스크린 상에 나타난 다른 percentage 들과 다르다.
KB/Sec	각 초당 kilobyte 단위로 나오는 disk 로의 혹은 disk 로부터의 data 전송율 (transfer rate)
Current	last update interval 동안 그 disk 의 평균 queue length 예를 들어, 만약 그 interval 의 반 동안에 queue length 는 3 이고 나머지의 시간 동안에 0 이면, current value 는 1.5 가 된다.
q=0	GlancePlus 가 시작되거나 Zero command (z)에 의해 statistics 가 reset 된 이후 r 그 disk 가 어떤 I/O request 도 가지지 않은 시간의 percentage. 이것은 그 disk 의 idle time 을 보여준다.
0<q<=2	disk queue 가 0 보다는 크고 2 보다는 작거나 같은 시간의 percentage. 만약 그 drive 가 queue 된 하나의 I/O 만을 항상 가진다면, 이 value 는 항상 100%이다. Example: 그림 5-7 에서 disk device **/dev/dsk/c8d0s*** 가 statistics 가 reset 된 후 100%의 시간의 queue 에 5 에서 8 의 request 를 가진다. 동일한 interval 동안에 disk **/dev/dsk/c0d0s*** 는 100% 시간의 queue 는 하나 혹은 둘의 request 를 가졌다.

Column	Definition
2<q<=4	device 가 2 보다는 크고 4 보다는 작거나 같은 I/O request 를 가지고 busy 한 시간의 percentage.
4<q<=8	device 가 4 보다는 크고 8 보다는 작거나 같은 I/O request 를 가지고 busy 한 시간의 percentage.
q>8	device 가 8 보다 큰 requests pending 을 가지고 busy 한 시간의 percentage. 이 마지막 필드에서의 큰 값은 disk I/O bottleneck 상황을 가르킬 수 도 있다.

queue lengths 의 percentage 들의 합은 rounding error 들 때문에 정확히 100%가 되지 않을 수도 있다. 예를들어, percentage 77.5, 2.5, 그리고 20.0 (전체 100%)는 101%의 전체를 주기위해서 78, 3, 그리고 20 으로 rounded 되었다.

G) Swap Detail Screen

이 스크린은 모든 swap area 에 대해서 space allocation 수치를 제공한다. (figure 5-8)

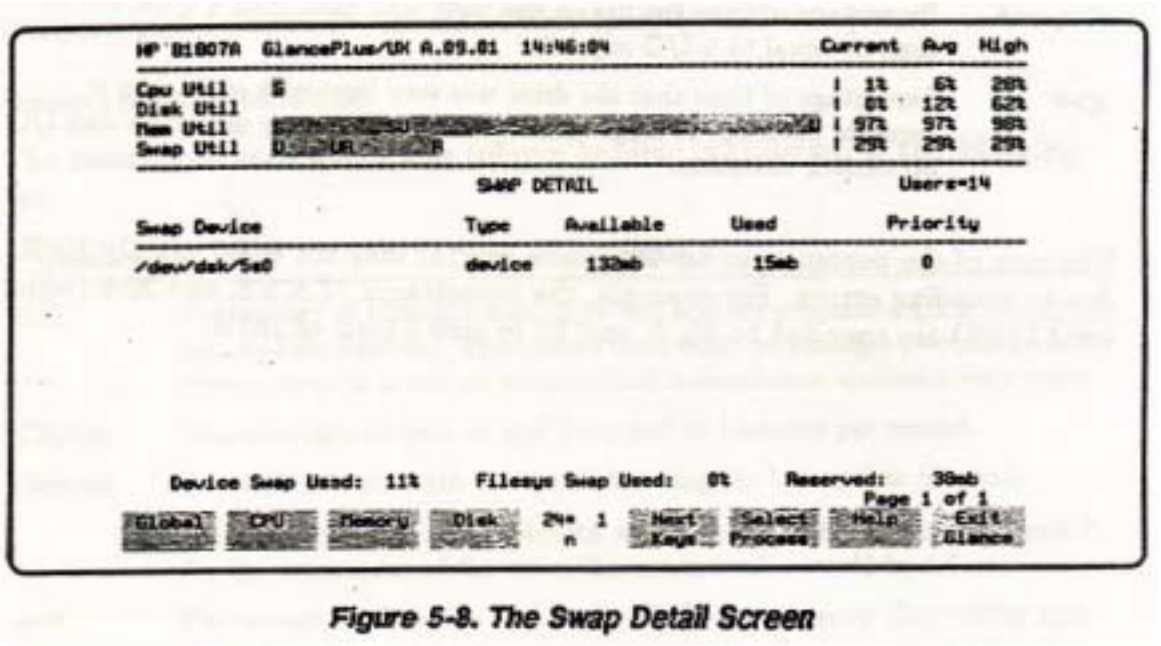


Figure 5-8. The Swap Detail Screen

여러분들은 w 나 Swap Space key 를 이용하여 어떤 스크린에서나 이 스크린으로 접속할 수 있다.

1. How Swap Space is Used

문제가 있는지를 판단하기 위해 swap space 가 어떻게 사용되는지 이해하는 것이 중요하다. 이용가능한(available) swap space 는 filesystem 내에 있는 지정된 swap device 에나 혹은 remote computer 에 위치할 것이다. Filesystem swap 에서는 예외적으로 available swap space 가 statis quantity 이다. 하지만 filesystem swap 은 역동적으로 "swchunk" increment 에서 증가한다. (*How HP-UX Works: Concepts for the System Administrator* 를 참조하라). GlancePlus 는 각 swap device 에서 현재 available 한 swap space 를 보고해 준다.

Process 가 생성될때, swap 을 위해 필요한 space 가 "reserved" (not "used") 된다. space 를 reserving

하는 것은 counter 를 incrementing 하는 것과, 이용가능한 reserved 되지 않은 swap space 가 있는지를 체크하는 것을 포함한다. 만약 그렇지 않으면, 그 프로세스 생성은 실패하게 된다. 이것은 프로세스가 swapped out 될때 swap space 를 할당 (allocating or using) 하는 것과 비교되어지는 비교적 빠른 operation 이다. 구 버전의 Unix 는 프로세스가 생성될 때 space 를 할당하고 그 프로세스를 swap area 로 copy 했을 것이다. 이런 overhead 는 file attributes 에 "sticky" bit 을 생성하는 주요한 이유가 된다.

Swap Util Bar 를 이용하여 GlancePlus 는 현재 얼마나 많은 available swap space 가 reserved 되어 있고, 그것들 중 얼마만큼이 physical 하게 사용되고 있는지를 보여준다. Filesystem swap area 에서 available swap space 는 dynamic 하게 증가하고, 그 증가치는 "swchunk" 단위라는 것을 기억하라.

Swap Util Bar 가 100%에 다다를 때, 그것은 swap space 가 고갈 되고 있음을 지시한다. (비록 그 space 가 아직 사용되지 않았을 지라도). 그 실제적으로 사용된 부분의 막대의 증가는 (Actively Used portion of bar increasing) 실제적으로 프로세스들이 swapped out 되고 있음을 말하고, 주 memory 의 요구가 증가하고 있음을 말할 수도 있다.

2. Screen Elements

각 swap device 는 스크린의 왼쪽에 있는 **Swap Device** column 에 목록화 되어있다. 통계치는 swap area 의 유형(type), 이용가능하고(available) 사용된(used) 양, 그리고 우선순위(priority)를 보여준다. 전체적인 swap space 사용율(utilization) 또한 보여진다.

① Swap Detail Measurements

4 column 의 통계치가 목록화된 swap device 에 대하여 제공된다.

Column	Definition
Swap Device	device, file system, 혹은 remote swap areas 와 관련된 swap device Device 와 filesystem swap device 들은 그들의 path name 들에 따라 구분된다. Remote swap area 는 그들이 붙어있는(attached) host name 에 따라 구분된다.
Type	swap 의 유형 -- entry 가 device, file system, 혹은 remote area 용인지를 지시한다.
Available	swapping 을 위해 이용가능한(available) swap space 의 양. 모든 "available"한 swap area 의 합은 Swap Util bar 에서 percentage 를 계산하는데 사용되는

denominator 이다.

Device swap area 에서, 이것은 swapon command 을 통해 enabled 된 swap space 의 양이다.

File system swap area 에서, 이것은 swapping 을 위해 이용가능한 space 의 양이다. 이것은 file system swap area 가 일반적 file 들과 함께 공유되기 때문에 enabled 된 양보다 적을 수도 있다.

Remote swap area 에서, 이것은 서버로부터 현재 요청된(requested) space 의 양이다.

Used 현재 쓰여진(written) available area 의 양. 이것은 remote swap area 를 위해서는 available 하지 않다. 모든 "used" area 의 합은 Swap Util bar 의 "U" portion 을 나타낸다.

Priority swap area 에 attached 된 최고치의 비교 우선순위로 0 (가장 높다)부터 10 (가장 낮다) 까지의 범위를 가진다. 가장 높은 우선순위 swap area 가 처음으로 사용된다.

② Swap Utilization Statistics

Device 와 file system 의 전체 사용율(utilization)과 swap space 의 양이 밑쪽에 주어진다. 이러한 통계치는 아래에 묘사되어 있다.

Swap Utilization	Definition
Device Swap Used	"device" 유형 swap area 들의 모든 "available"한 값의 합으로 나누어진(devided) 그러한 device 들의 모든 "used" 값의 percentage.
Filesys Swap Used	file system swap area 의 모든 available 한 값의 합으로 나누어진 그러한 area 들의 모든 used 값의 percentage
Reserved	프로세스들을 위해 reserved 된 전체 swap area. 이 수치는 Swap Util bar ("U"와 "R" 둘다)를 계산하기 위한 numberator 로 사용된다. Swap Util bar 계산(calculation)에서 denominator 는 위의 "Available" column 의 합이다.

Note 보고된 available file system swap space 의 양은 **newfs** command 의 **fs_minfree** parameter 에 의해 특성화된 reserved file space 를 포함하지 않는다.

Reserved block 은 단지 super-user 에 의해서 접속될 수 있다. Super-user 가 reserved blocks 을 사용할 때, file system swap space utilization 과 disk space

utilization 은 100%를 초과할 수도 있다. (뒤의 것은 **bdf** command 에 의해서 보여진다.)

fs_minfree parameter 는 free 가 될 수 있는 file system block 의 최소 수용가능한 percentage 를 정의한다. ; default 는 10%이다. (**fs_minfree** 에 대해서 더 많은 정보를 얻기 위해서는 man-page *fs* 를 참조하라)

H) NFS Detail Screen

이 스크린은 local system 의 inbound 와 outbound 된 network file system (NFS) activity 에 대한 현재의 통계치를 보여준다.(figure 5-9) Network file system 에서, local system 은 remote system 이 NFS-mount 를 통해서 사용하고 있는 disk 들을 가질 것이다. 바꾸어서, local system 은 local system 에 NFS-mounted 된 remote disk 를 가질 것이다. Inbound 통계치는 remote system 에 의해 야기된 local disk 에서의 activity 를 반영하고, outbound 통계치는 NFS-mounted 된 remote disk 에서의 local system 에 의한 activity 를 반영한다.

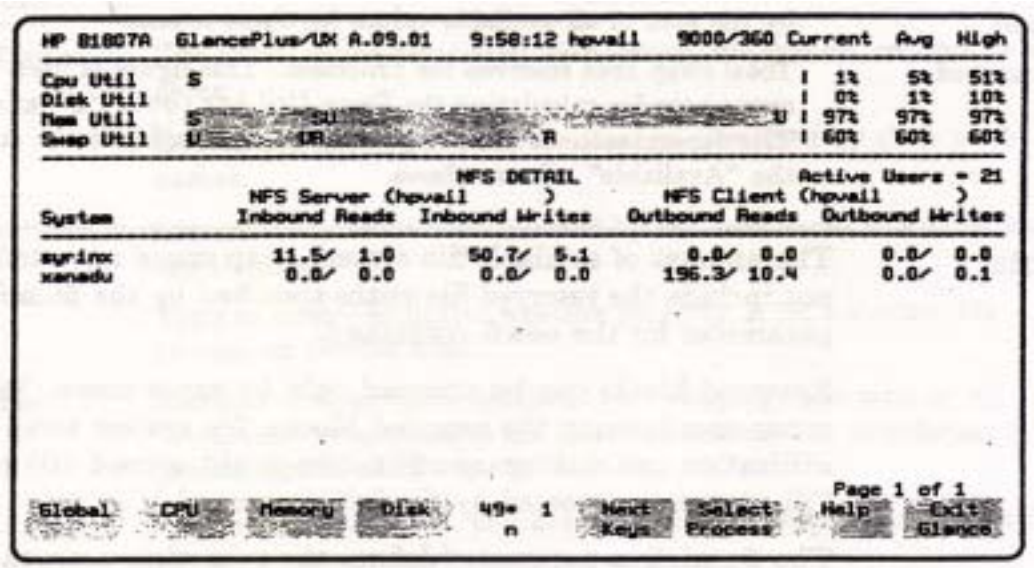


Figure 5-9. The NFS Detail Screen

여러분들은 n 이나 NFS key 를 이용하여 어느 스크린에서나 이 스크린으로 접속할 수 있다.

1. Screen Elements

수집된 NFS 통계치는 접속되어 있는 시스템에 의존한다.

예를 들어, Inbound 나 Outbound column 에서의 통계치는 주어진 상황을 단지 0 으로 보여줄 수도

있다. 이것은 어떠한 activity 도 관계(relation)에서 발생하지 않거나, 어떠한 관계도 존재하지 않는다는 것을 의미한다.

Figure 5-9 에서, 예를 들어, system hpvail 에서 GlancePlus 가 구동하고 있는데, system syrinx 와 xanadu 와의 관계 (relation) 가 보여진다. 이 예에서, local hpvail system 은 remote system syrinx 의 server 이며, syrinx 가 NFS-mounted 되어있는 local disk 로부터의 inbound NFS activity 를 보여준다.

어떤 outbound activity 가 보여지지 않기 때문에, hpvail,은 syrinx 로 NFS-mounted 된 disk 를 가지지 않거나, disk 상 어떠한 activity 도 일어나지 않는다는 것을 의미한다.

System hpvail 의 xanadu 와의 관계는 NFS client 로서이다. 이것은 hpvail 이 xanadu 로 연결된 하나 이상의 NFS-mounted disk 들을 가진다는 것이다. 따라서, hpvail 과 xanadu 사이의 관계를 위한 activity 는 Outbound column 아래에 보여진다. Inbound column 아래에는 어떤 activity 도 보여지지 않는데, 이것은 xanadu 가 hpvail 에 mount 된 remote disk 로의 어떠한 access 도 없거나, 그 remote disk 가 active 하지 않다는 것을 의미한다.

만약 두개 이상의 disk 들이 같은 remote system 으로부터 NFS-mounted 되어 있다면, 그러한 disk 들의 모든 I/O 들은 같은 row 에 보여질 것이다.

Measurement Interface 가 시작된 후 disk 가 추가 되었다면, 그것은 "system" column 에 "other"로 나타난다.

① NFS Statistics by System

스크린의 왼쪽에 목록화된 각 시스템에 대하여, read/write activity 의 4 column 들이 current/average 포맷으로 보여진다. (figure 5-9)

Column	Definition
System	어떤 통계치가 수집되었는지와 관련하여 시스템을 구분 한다.
NFS Inbound Reads	NFS-mounted 된 local disk partition 을 가진 remote system 으로부터 기초한 Reads. remote system 에서 구동하는 프로세스가 local 하게 mount 되어있는 disk 로부터 physical read 로 결과가 나타나는 request 를 행할 때, 이것은 Inbound Read

request 로 세어진다. 이 수치는 cashing 과 buffering 때문에 logical read request 에 직접적으로 상응하지는 않는다.

Example: 이 column 에서 11.5/1.0 의 값은 last update interval 동안, 지정된 시스템으로부터의 remote process 들이 local 하게 mount 된 disk 에 대하여 초당 평균 11.5 의 read 를 생성했다는 것을 의미하고, remote system 에서의 프로세스들이 통계치 수집이 시작되거나 혹은 reset 된 후 초당 1.0 의 read request 를 생성했다는 것을 의미한다.

Inbound Writes remote 프로세스들로부터 remote system 이 NFS-mount 시킨 local disk partition 으로의 physical write request 들.

Outbound Reads local 프로세스로부터 NFS-mounted 된 remote disk partition 으로의 physical read request 들

Example: 만약 여러분의 local system 이 **foobar** 라 불리는 시스템으로부터 **/extradata** 라는 이름의 disk partition 을 mount 했다면**, /extradata** 아래의 file 로부터 읽는 local 프로그램은 **foobar** 로 lable 된 row 에 outbound physical NFS read 들을 생성할 것이다.

Outbound Writes local 프로세스가 remote NFS-mounted 된 disk 에 쓸 때 (write) 일어나는 physcal write request 들

I) The LAN Detail Screen

이 스크린은 각 시스템에 구성된 각 local area network card 에 대한 4 가지 기능들을 제공한다. (figure 5-10). Network 시스템에서, 이 정보는 과도한 LAN activity 에 의해 유발될 수 있는 bottleneck 을 보여줄 수 있다.

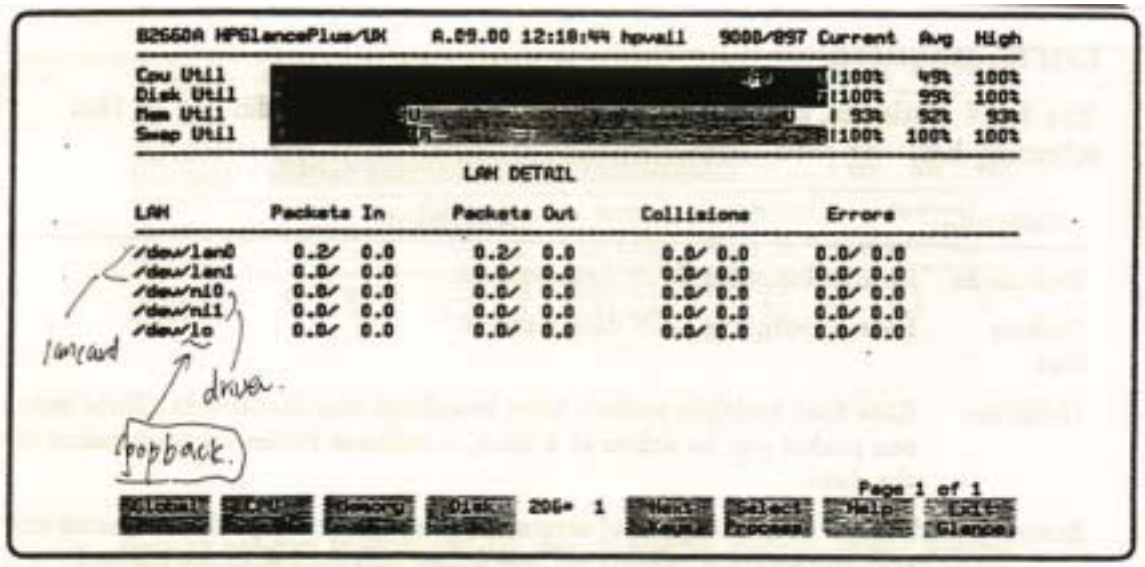


Figure 5-10. The LAN Detail Screen

여러분들은 l (소문자 L)이나 LAN key 를 이용하여 어느 스크린에서나 이 스크린으로 이동할 수 있다.

1. Screen Elements

첫번째 LAN column 은 LAN interface card 을 나타내는 각 device file 을 나타낸다. 추가적인 entry 들이 internet protocol frame 들을 송수신하는데 관여하는 interface device driver 들 (/dev/ni*)과 loopback function (/dev/lo)을 위해서 보여진다.

① LAN Measurements

Current/average 포맷으로 보여지는 LAN 통계치가 아래의 목록에 나와 있다.

Column	Defiintion
Packets In	들어오는(incoming) LAN data packet 의 비율(rate)
Packets Out	나가는(outgoing) LAN data packet 의 비율
Collisions	다중의(multiple) packet 들이 동시에 broadcast 되는 비율. 한번에 단지 하나의 packet 만이 active 되기 때문에, collision 은 data 의 재전송을 강요한다.
Errors	packet 에 일어날 수 있는 다양한 유형의 error 들의 합(sum): frame 들이 전송 안된 경우, frame 을 받았지만 전달 되지 않은 경우, bad Cyclical Redundancy Check (CRC) error 를 가진 packet 의 경우, 잘못 할당된 packet 의 경우, 자원의 부족에 의해 놓친 packet 의 경우, 유효하지 않은 control field 값을 가진 packet 의 경우, 그리고 알 수 없는 protocol field 때문에 떨어진(dropped) packet 의 경우가 있다. GlancePlus 는 이러한 유형의 error 들 사이의 차이점을 구별 못한다. LAN 상태에 대한 더 많은 정보를 얻기 위해서는 LANDIAG diagnostic tool 을 사용할 수 있다.

J) Diskless Server Resource Utilization Screen

이 스크린은 cluster server 의 자원에 대한 diskless cluster client demand 에 대한 정보를 제공한다. (figure 5-11). 이 스크린은 단지 diskless cluster server 에만 관계가 있다. 만약 시스템이 diskless server 가 아니면서 이 스크린에 접속하기를 원한다면, 다음과 같은 문구가 뜰 것이다.: This system is not a diskless server

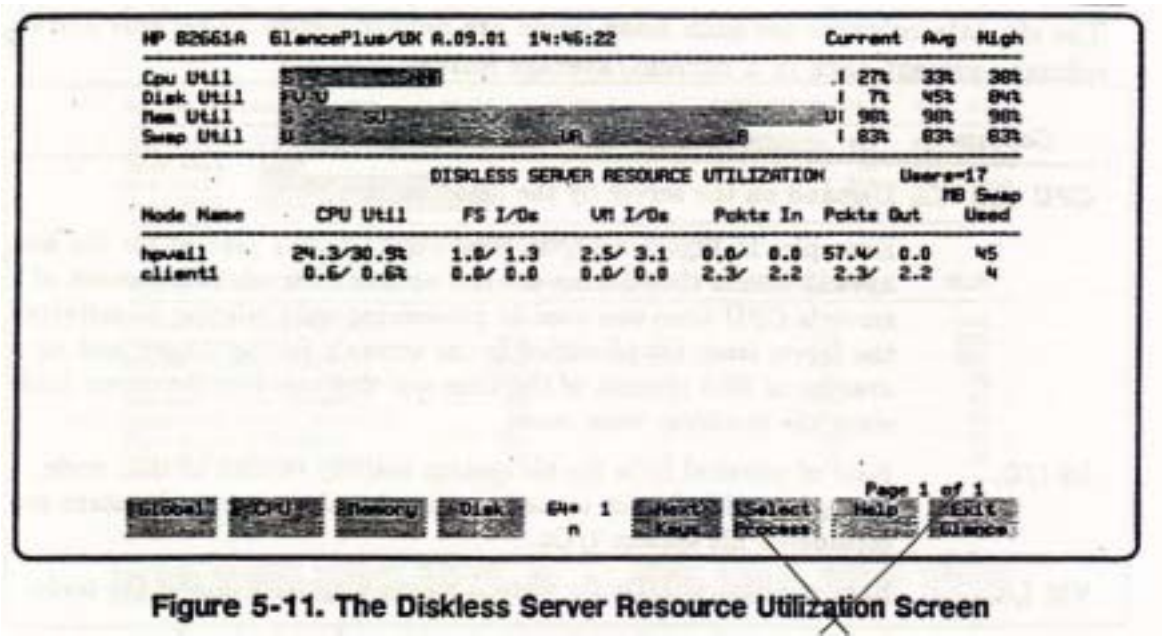


Figure 5-11. The Diskless Server Resource Utilization Screen

여러분은 k 나 Diskless Server key 를 이용하여 어느 스크린에서나 이 스크린을 접속할 수 있다.

Note 만약 GlancePlus 가 구동되는 동안에 cluster 가 변화되면 (예를 들어, 새로운 cluster 를 추가한 경우) 그 스크린은 GlancePlus 를 빠져나가거나 midaemon 이 재시작될 때 까지 변화를 반영하지 않을 것이다.

1. Screen Elements

Node Name 이라고 labled 된 첫번째 column 은 서버로서 사용되는 local node 를 포함하여 그

cluster 에 대한 모든 active node 들을 목록화 한다. 다른 column 들은 client node 들에 의한 시스템 자원 사용에 대한 percentage 를 포함한다.

① Measurements for Each Node

Node name 에 대한 6 개의 data column 이 아래에 정의 되어 있다. 처음 5 개의 data column 들은 current/average 포맷으로 data 를 나타낸다.

Column	Definition
CPU Util	목록화된 node 에 의한 그 서버에 대한 요구(demand) Example: Figure 5-11 에서, node hpvail 에 대한 수치 24.3/30.9 %는, last update interval 동안에 그 서버의 CPU time 의 24.3 %가 그 서버 자체의 activity 와 관련된 request 를 처리하기 위해서 사용된 것이고 (스크린의 Banner Line 에 정의된 것 처럼), 그 time 의 평균 30.9 %가 통계치가 reset 된 이래 그 server node 에 지정된 것이다.
FS I/Os	그 node 에 관련된 file system activity 에 대한 physical I/Os 의 비율(rate) Virtual memory subsystem 에 의해서 유발되지 않은 physical I/O 들이 file system I/O 로 고려되어 진다.
VM I/Os	그 node 때문에의 virtual memory activity 에 대한 physical I/Os 의 비율. Example: 여기에서의 값 1.4/0.5 는, current interval 동안 초당 1.4 I/O 가 일어났고, 통계치가 시작되거나 reset 된 후 평균 0.5 의 VM I/O 가 일어났다는 것을 의미.
Pckts In	diskless network 에서 packet 은 서버와 client node 사이에서 교환되기 때문에, "in packets"은 client 가 서버로 보내는 requests 이다. 하나의 logical request 는 몇몇의 각개 (individual) packet 들로 보내질 수 있다.
Pckts Out	diskless network 에서 packet 은 서버와 client node 사이에서 교환되기 때문에, "out packets"은 서버가 client 로 보내는 requests 이다. 하나의 logical request 는 몇몇의 각개 (individual) packet 들로 보내질 수 있다.
MB Swap Reserve	각 client node 를 대신하여 reserved 된 그 서버의 local swap space 의 양

K) System Table Utilization Screen

이 스크린은 각 중요한 internal system table 들의 구성 size 와 사용에 대한 정보를 제공한다. (figure 5-12). 이 정보는 평균 workload 가 얼마나 잘 kernel configuration 을 의존하는지를 보여준다. 여러분들은 이 정보를 해석하기 전에, 이러한 table 들을 이해해야만 한다. 그것들은 *HP-UX System Administration Tasks* manual 에 잘 나와 있다.

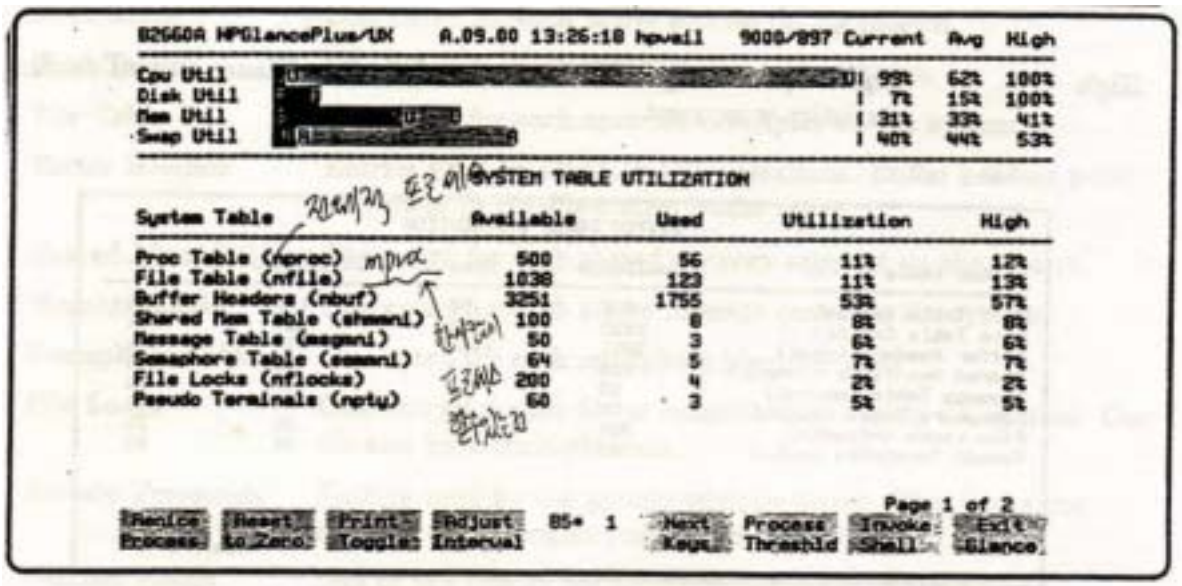


Figure 5-12. The System Table Utilization Screen

여러분들은 t 나 System Tables key 를 이용하여 어느 스크린에서나 이 스크린으로 이동할 수 있다.

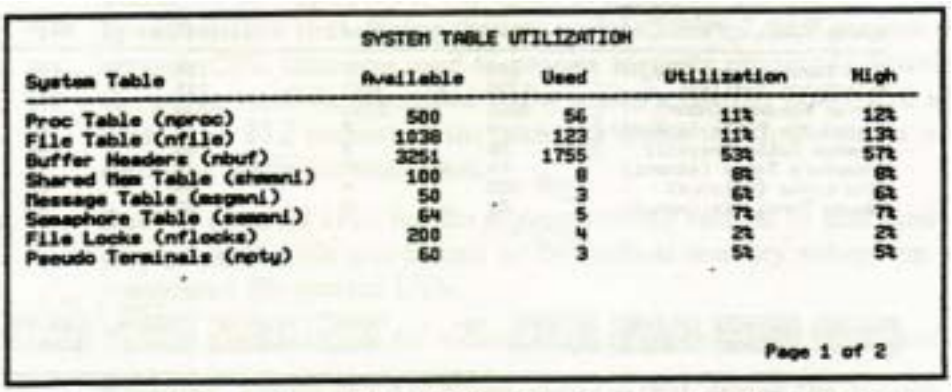
1. Screen Elements

System Table Utilization 스크린은 약간 다른 column header 를 가진 두개의 page (screen)을 가지고 있다. Page-1 스크린은 다양한 시스템 table 에 대한 통계치를 보여주고 (figure 5-12), page-2 는 선택된 시스템 buffer pool 에 대한 정보를 포함한다.

① Page 1 – System Tables

스크린의 왼쪽에 목록화된 **System Table** 에 대한 4 가지 유형의 정보가 제공된다.

Column	Definition
Available	구성된(configured) entry 들의 수
Used	현재 사용중인 entry 들의 수
Utilization	첫 번째 2 개의 column 들로부터 계산된 사용율의 percentage.
High	Glance 가 시작 되거나 통계치가 reset 된 이후 기록된 최고치의 percentage utilization



SYSTEM TABLE UTILIZATION

System Table	Available	Used	Utilization	High
Proc Table (nproc)	500	56	11%	12%
File Table (nfile)	1038	123	11%	13%
Buffer Headers (nbuf)	3251	1755	53%	57%
Shared Mem Table (shmmni)	100	8	8%	8%
Message Table (msgmni)	50	3	6%	6%
Semaphore Table (semmni)	64	5	7%	7%
File Locks (nflocks)	200	4	2%	2%
Pseudo Terminals (npty)	60	3	5%	5%

Page 1 of 2

Figure 5-13. System Table Utilization Screen Details—Page 1

② System Tables Measured

열 개의 system table 들이 page-1 스크린에 목록화 되어 있다. 각각의 의미는 아래의 page 에 설명되어 있다. 좀더 상세화된 정보는 *HP-UX System Administration* manual 에서 찾을 수 있다. 그 table 의 최대 size 를 결정하는 kernel configuration parameter 는 각 table 이름을 뒤따르는 괄호안에 지시되어 있다.

Table	Type of Statistic Displayed
Proc Table	시스템의 각 active 프로세스에 대한 하나의 entry
Text Table	각각의 active 한 공유된 text segment 에 대한 하나의 entry
File Table	각 open file descriptor 에 대한 하나의 entry
Buffer Headers	모든 block I/O operations 에 의해 사용된 entry 들. Buffer headers 는 file-system buffer cache 내의 buffer 를 지칭한다.
Shared Mem Table	각 shared memory segment 에 대한 하나의 entry
Message Table	각 active message queue 에 대한 하나의 entry
Semaphore Table	각 semaphore identifier (set)에 대한 하나의 entry

File Locks 시스템에서 active 한 각 file 이나 record lock 에 대한 하나의 entry. 하나의 file 은 다중의 lock 을 가질 수 있다.

Pseudo Terminals pseudo-teletype driver 에 의해 사용된 entry 들. 이것은 시스템에 pty login 가능한 수를 제한한다.

Diskless fsbufs cluster file system buffer pool 의 size. 이 pool 은 inbound cluster traffic 을 위해 사용되는 file system buffer 의 모음(collection)이다.

③ Page 2 – Selected Buffer Pools

System Table Utilization 스크린의 두번째 page 는 선택된 (selected) system buffer pool 들에 대한 정보를 보여준다. (figure 5-14)

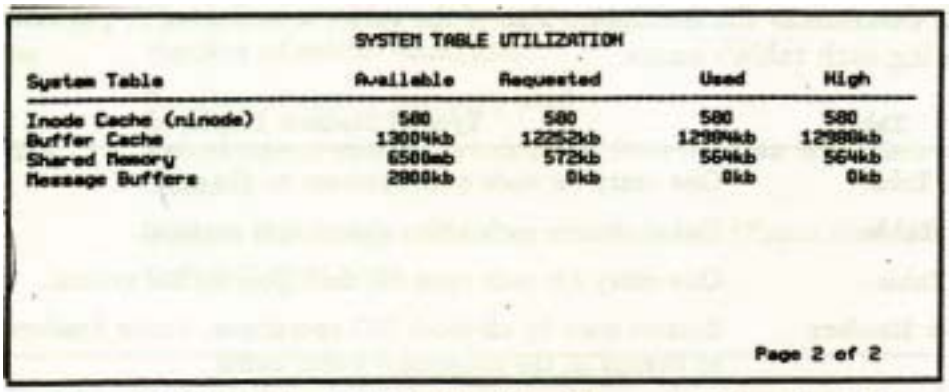


Figure 5-14. System Table Utilization Screen Details—Page 2

④ Statistics Displayed

아래의 column 들은 Detail Display section 에서 보여진다.

Column	Definition
Available	memory pool 의 configured size
Required	Inode Cache 로 사용되는 entry 들의 수 이거나(or), pool 로부터 요구되어지는 memory 의 양
Used	Inode Cache 로 사용되는 entry 들의 수 이거나(or), pool 에 의해서 사용되는 physical memory 의 양 Buffer cache 에 대해서, 이 숫자는 fragmentation 때문에 요구되는 것보다 더

	커질 수 있다. Shared memory 에 대해서는, 만약 몇 shared memory 가 swapped out 되었다면 이 수치가 사용된 양 보다도 더 작을 수 있다. 요구되어지고(required) 사용된(used) memory 는 message buffer 와 항상 같을 것이고, 현재 요구되어지는 양 이라기 보다는 지금까지 요구된 최대치의 양을 지시한다.
High	GlancePlus 가 시작되거나 혹은 reset 된 후 최고치의 utilization.

목록화된 buffer pool 들은 아래에 정의 되어 있다.

Table Buffer Pool	Definition
Inode Cache	open in-core inode 를 포함하는 cache. 이것은 memory cache 이기 때문에 일반적으로 full 이다.
Buffer Cache	I/O drivers 를 위한 buffer 를 포함하는 모든 block I/O buffer 들이 저장되어 있는 pool. 이것은 cache 이기 때문에, 일반적으로 채워질 것이다(filled).
Shared Memory	모든 할당된(allocated) shared-memory segment 가 저장되어 있는 pool. HP-UX Measurement Interface 는 하나의 shared-memory segment 를 사용한다는 것을 알아두자.
Message Buffer	모든 사용된(used) message-queue buffer 가 저장되어 있는 pool.

L) Logical Volumes Screen

이 스크린은 시스템에서 생성된 logical volume 에 대한 정보를 제공한다.(figure 5-15). 2 가지 형태의 entry 가 보여진다. 한 줄은 Vol Group/Log Volume column 에서 VGnn 으로 지정된 logical volume group 을 보여준다. Open LVs 와 mirror write consistency cache Size and Qlen metrics 는 각 volume group 별로 특성화 되어있다. 각 volume group 아래의 줄들은 그 volume 에서 opened 된 logical volume 들을 지시한다. Volume name, read, write, hit and miss metrics 는 각 logical volume 별로 보여진다.

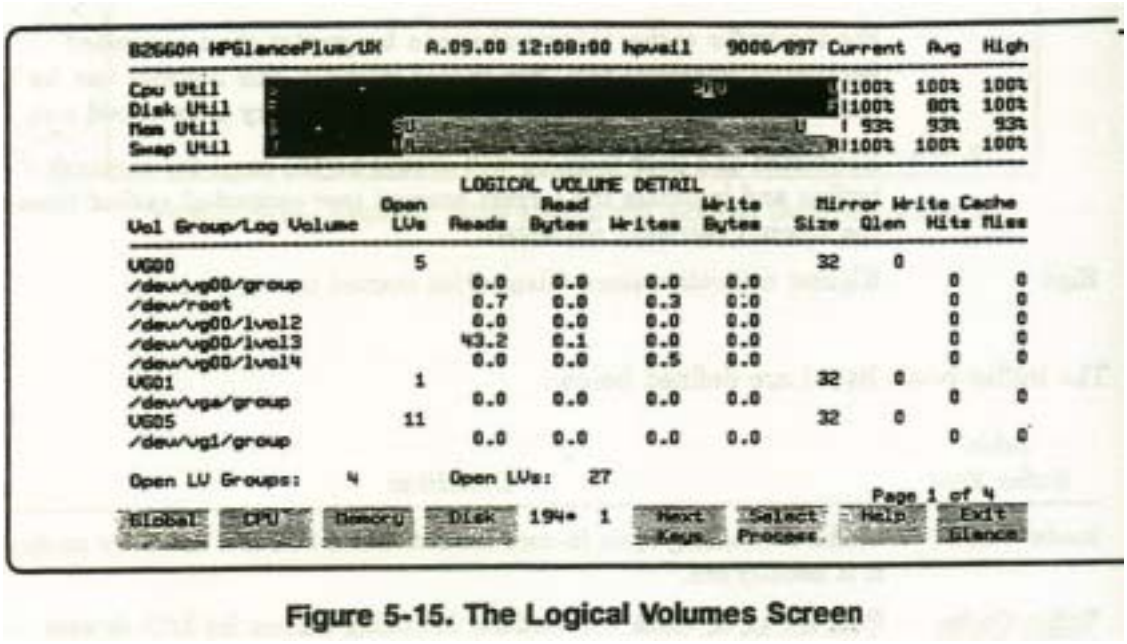


Figure 5-15. The Logical Volumes Screen

여러분들은 v 나 Logical Volumes key 를 이용하여 어떤 스크린에서나 이 스크린으로 접속할 수 있다.

1. Screen Elements

Metrics	Definition
Open LVs	이 volume group 을 위해 open 된 logical volume 의 수
Reads	그 volume 에 대한 physical read count read (초당 reads)
Read Bytes	이 volume 에 read 하는 전송율 (transfer rate) (초당 kilobytes 단위)
Writes	그 volume 에 대한 physical write count rate.

Writes Bytes 이 volume 으로 write 하는 전송율.

다음 4 개의 필드는 그 Logical Volume Manager 의 Mirror Write Cache facility 와 특별히 관계된다.

Metrics	Definition
Mirror Size	그 volume group 에 대한 Mirror Write Cache 의 size (kilobyte 단위로)
Write Qlen	Mirror Write Cache 로 queue 로 쌓인 request 들의 현 수(number)
Cache Hits	Mirror Write Cache hits 의 현 수.
Cache Miss	Mirror Write Cache misses 의 현 수

아래의 전체 (global) metrics 는 스크린의 밑부분에 보여진다.

Metrics	Definition
Open LV Groups	logical volume group open 의 전체 수.
Open LVs	전체 volume group 에서 logical volumes open 의 전체 수

M)All CPUs Detail Screen

Multi-CPU 시스템에서, 이 스크린은 각 CPU 들에 대한 상세화된 정보를 제공한다.(figure 5-16)

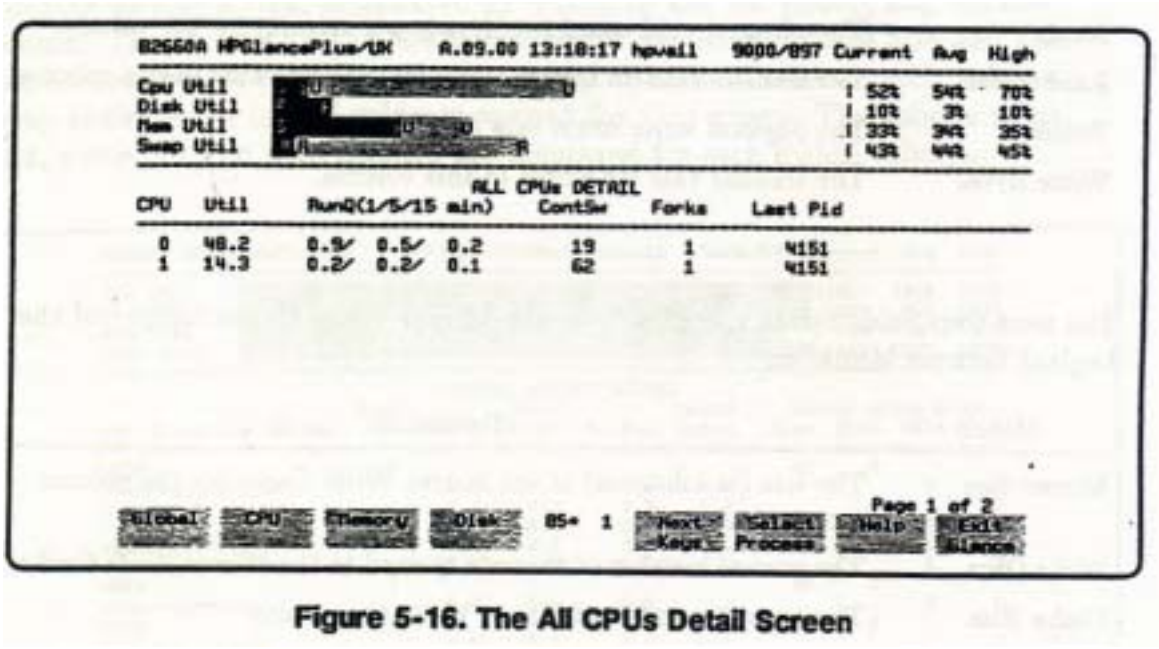


Figure 5-16. The All CPUs Detail Screen

여러분들은 a 나 All CPUs key 를 이용하여 어느 스크린에서나 이 스크린으로 이동할 수 있다.

1. Screen Elements – Page 1

All CPUs Detail 스크린의 Page 1 은 아래의 정보를 제공한다.

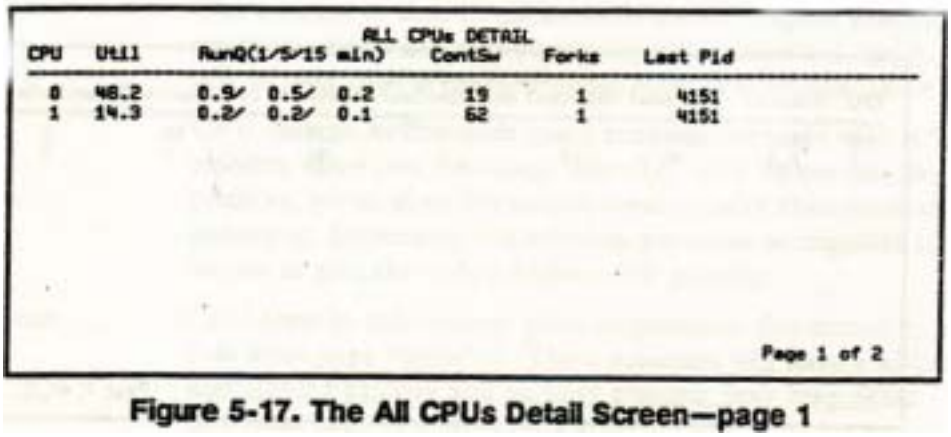


Figure 5-17. The All CPUs Detail Screen—page 1

Column	Definition
CPU	CPU identifier
Util	last update interval 에 걸쳐 이 CPU 가 busy 했던 시간의 percentage. 이러한 percentage 의 평균값은 스크린 위쪽에 있는 CPU Util bar 에서 보여진다. 비록 짧은 update interval 에서는 어떤 통계치가 샘플로 쓰였는지의 각 시간의 차이가 다양함을 유발할 수도 있지만.
RunQ(1/5/15 min)	run queue length 혹은 이 CPU 에 대한 load average. Last 1-분, 5-분, 15-분의 시간에 걸쳐 CPU time 을 기다리고 있는 수행가능한 프로세스들의 수가 보여진다.
ContSw	last interval 동안에 이 CPU 에 있는 context switch (CPU 가 구동하는 하나의 프로세스에서 다른 것으로 switch 될때) 의 수.
Forks	last interval 동안에 이 CPU 에서 생성된 process forks 의 수.
Last Pid	이 CPU 에서 구동하고 있는 last 프로세스의 PID

2. Screen Elements – Page 2

Page 2 는 6 개의 다른 activity 들에 대한 processor time allocation 에 대한 정보를 보여준다.(figure – 18). 이 allocation 정보는 각 CPU 에 대해서 전개된다.

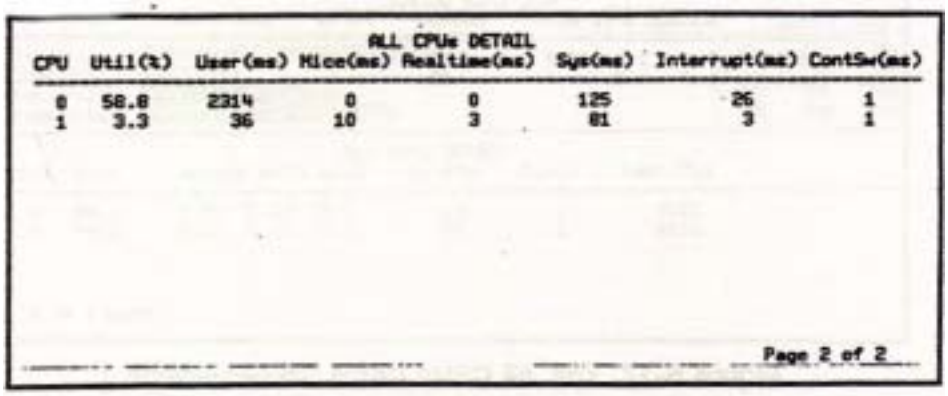


Figure 5-18. The All CPUs Detail Screen—page 2

Column	Definition
CPU	CPU identifier
Util	last update interval 에 걸쳐서 CPU 가 busy 했던 시간의 percentage.

User 정상적인 우선순위 조절이 적용될 때 CPU 가 user program code 를 수행하는데 소비한 (millisecond 단위의) 시간의 양. User time 의 나머지는 Nice CPU 나 Realtime CPU 하에서 count 된다.

Nice "nice" 우선순위 값을 가지고 (*see* man-page *nice (1)*) user code 를 구동하는데 소비된 (millisecond 단위의) 시간의 양. Nice 값은 양수 값을 가질 수 있는데, 그것은 그 code 에 일반 프로세스들보다도 더 낮은 우선 순위를 준다. Superuser 는 code 에 더 높은 CPU 우선순위를 주기 위해서 음수의 nice 값에서 프로세스들을 스케줄할 수 있다.

Realtime rtprio (*see* man-page *rtprio(1)*)에 의해 파견된 프로세스들에 주어진 (millisecond 단위의) CPU time. 이러한 프로세스들은 더 높은 CPU dispatching priority 를 가질 것이고, busy 시스템에서는 다른 프로세스들이 주어진 CPU 에서 수행하는 것을 막을 것이다.

Sys 시스템 call 과 code 를 수행하는데 소비된 CPU time (Interrupt 와 ContSw 아래로 수행된 시간이라기 보다는).

Interrupt code 를 조절하는 시스템 interrupt 을 수행하는데 소비된 CPU time. 높은 interrupt rate 은 높은 I/O rate 을 가진 시스템에서 일어난다. 비정상적으로 높은 interrupt rate 은 hardware 문제를 지칭할 수도 있다.

ContSw 프로세스들 사이에서 context switching 에 소비되는 CPU time. 이것은 time slice 가 만기된후 구동하고있는 프로그램이 CPU 를 포기해야만 하는지를 고려하는데 소비된 시간을 포함한다. 이것은 idle loop 나 다음에 어떤 프로세스가 구동하는지를 결정하기 위해 시스템에 의해서 취해진 시간은 포함하지 않는다.

N) Individual Process Screen

이 스크린은 특정 프로세스에 대한 상세화된 정보를 제공한다. (figure 5-19). 여러분들은 s 나 Select Process key 를 이용하여 어느 스크린에서나 이 스크린으로 접속할 수 있다.

어느 경우에든, 보고자 하는 프로세스의 PID 를 위한 prompt 가 뜰 것이다. 만약 그 PID 를 모르면, 모든 interesting 프로세스들의 PID 들이 목록되어 있는 Global screen 을 참조하면 된다. Default PID 는 마지막으로 선택된 프로세스이고, 만약 아무 것도 선택되지 않았다면 현재 top CPU user 에 있는 프로세스가 된다. Default 를 선택하기 위해서는 Return 을 치면 된다.

Select PID 를 취소하고 원래 스크린으로 돌아가고 싶으면, q (Quit) 나 Cancel (F8)을 치면 된다.

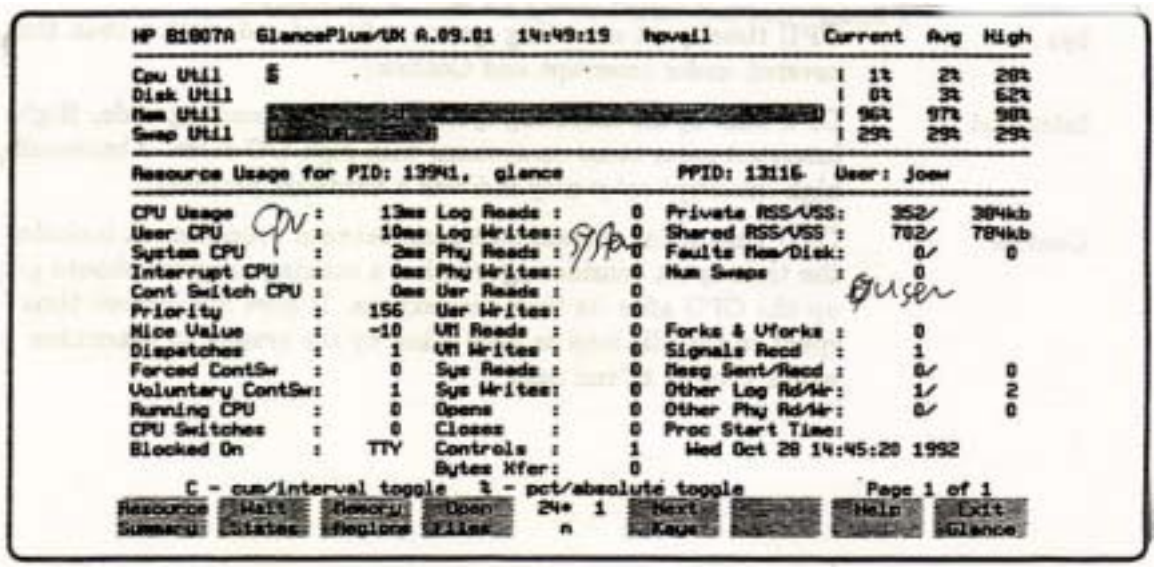


Figure 5-19. The Individual Process Screen

1. Screen Elements

스크린의 process display portion 의 첫 줄은 모니터 되고 있는 프로세스의 PID, 그것의 이름, parent PID, 그리고 그 프로세스와 관련된 user name 을 보여준다.

Note: 만약 원래의 parent PID 가 죽거나(died), 그것이 init process 에 의해서 채택되었다면(adopted), 그 프로세스는 그것의 PID 1 대신에 그것의 PPID 로서 원래 parent PID 를 보여줄 것이다.

그 줄 아래로, 통계치들이 3 개의 column 으로 구성되어 있다. 비록 labled 되지 않았지만, 첫번째 column 은 CPU-related 통계치, 두번째는 I/O 통계치, 그리고 세번째는 memory management 값에 대한 것이 나온다.

이 스크린의 통계치는 last update interval 동안 취해진 count 나, 그 프로세스가 살아있는 동안이나 midaemon 이 시작된 이래로 측정된 축적된 count 로서 보여질 수 있다. 축적된 것과(cumulative) interval display 들 사이에 toggle key 로 "C"를 사용할 수 있다.

① CPU-Related Statistics (Column 1)

첫번째 column 은 프로세스의 CPU 사용과 어떤 우선순위 아래에서 프로세스가 구동하고 있는

지에 대한 상세 정보를 제공한다. 전체 CPU activity 의 cumulative (absolute) time 과 percentage (pct) 사이의 CPU measurement display 를 붙잡기 위해 "%" key 를 사용하면 된다.

Metrics	Definition
% CPU Usage	last update interval 동안에 시스템 CPU time 의 percentage 혹은 이 프로세스에 devote 된 absolute CPU time
User CPU	last update interval 동안에 이 프로세스의 user code 에서 구동하는데 소비되는 time 의 percentage 혹은 absolute time. Example: 만약 update interval 이 5 초로 세팅되어 있고, 그 프로세스가 그 시간동안 CPU time 의 10%를 사용했다면, 이 user code 에서 수행된 모든, 이 값은 500ms 이 된다.
System CPU	kernel code 내에서 이 프로세스를 구동하는데 소비된 time 의 percentage 나 absolute time.(system call 에 의해서). 모든 프로세스들은 system code 를 수행하는데 약간의 시간을 소비하지만, System CPU 에 대한 큰 값은 불필요하거나 불충분한 system routine 을 call 하는 프로그램과 같은 문제를 가르킬 수 도 있다.

Metrics	Definition
Interrupt CPU	프로세스가 구동을 시도하고 있는 동안에 system interrupt handling code 를 수행하는데 소비되는 time 이나 absolute time 의 percentage. 높은 interrupt rate 은 높은 I/O rate 을 가진 시스템에서 일어난다. 그 프로그램은 그것이 얼마나 자주 interrupt 되는지에 대한 조절(control)을 거의 가지지 않는다.
Cont Switch CPU	프로세스를 blocked state 로부터 running state 로 switching (context switching)하는데 소비되는 time 이나 absolute time 의 percentage. User 는 이러한 값에 대한 조절을 거의 가지지 않는다 (일반적으로 꽤 작은 값을 가지는).
Priority	마지막으로 파견된 (dispatched) 프로세스가 가지는 CPU 우선순위. 정상적인 timesharing process 에서, 이 수치는 그 프로세스와 시스템의 run-time 특성에 따라 변할 것이다.
Nice value	"nice" command 나 system call 을 사용하는 이 프로세스를 시작 시킴으로써 유발되는 정상적인 time-sharing 우선순위로의 조절(adjustment). Nice value 는 정상적인 time-sharing 우선순위보다 높거나 혹은 낮을 수도 있다. 예를 들면, default 로 GlancePlus 는 그 자체에 –10 이라는 nice value 를 줌으로

	써 정상적인 것보다 더 높은 dispatching priority 에서 구동한다.
Dispatches	last interval 동안에 CPU 에서 수행하는 (executing) 프로세스 – was dispatched – 가 만들어 지는 횟수. 이것은 어떤 다른 프로세스가 CPU 를 조절하기 때문에 context switch 와 관련된 dispatch 들을 포함하고, 또 그러는 동안에 다른 프로세스가 구동하지 않으면서 프로세스가 멈추고, 다시 시작하고 하는 횟수를 포함한다.
Forced ContSw	프로세스가 CPU 를 항복 시키도록 강요되어지고, 다른 프로세스가 구동 되기 때문에 즉시 시작되지 못한 횟수.

Metrics	Definition
Voluntary ContSw	일반적으로 system call 때문에, 프로세스가 suspend 된 횟수.
Running CPU	multi-CPU 환경에서, 프로세스가 현재 구동시키고 있는 CPU 의 수.
CPU Switches	multi-CPU 환경에서, 프로세스가 몇 번이나 CPU 를 switch 시켰는가에 대한 정보
Blocked On	프로세스의 수행이 blocked 된 것에 대해 마지막으로 기록된 이유 (last recorded reason)

② Input/Output Statistics (Column 2)

Individual Process 스크린의 두번째 column 은 선택된 프로세스에 대한 input/output 정보를 제공한다. 이전처럼, 대부분의 last update interval 동안에 count 가 축적된 것이다.

Metrics	Definition
Logl Reads	disk 로의 logical read 의 수. 이것은 terminal 과 modem 으로의 read 와 같은 non-file-system I/O 를 포함하지 않는다.
Logl Writes	disk 로의 logical write 의 수. 이러한 logical counter 들은 시스템 call 들을 read 하고 write 하는 것과 직접적으로 관계가 있다.
Phys Reads	이러한 프로세스를 대표하여 만들어진 disk device 로부터의 physical read 의 수. 이것은 file-system-, raw-, virtual memory-, 그리고 다른 시스템의 I/Os 과 같은 모든 유형의 input 을 포함한다. Physical transfer 는 cache buffering 때문에 logical I/Os 와 직접적으로 관련이 있는 것은 아니다.
Phys Writes	이 프로세스를 대표하여 만들어진 disk device 로의 모든 유형의 physical write 의 수.
User Reads	user file system 과 raw device file reads 를 위한 file system device 로부터의

	physical read 의 수. 이것을 Phys Reads 의 subset 이다.
User Writes	user file system 과 raw device file reads 를 위한 file system device 로의 physical write 의 수. 이것은 Phys Writes 의 subset 이다.
VM Reads	이 프로세스와 관련된 disk device 로부터의 virtual memory management (page ins) 에 의한 physical read 의 수 .
VM Writes	이 프로세스와 관련된 disk device 로의 virtual memory management (page outs)에 의한 physical write 의 수 .
Sys Reads	이 프로세스와 관련된 system activity 로부터 결과 지어지는 file system 으로부터의 physical read 의 수 .
Sys Writes	이 프로세스와 관련된 system activity 로부터 결과 지어지는 file system 으로의 physical write 의 수 .

Metric	Definition
Opens	이 프로세스가 살아 있거나 **midaemon** 이 시작된 이후 만들어진 file open 의 축적된 수.
Closes	이 프로세스가 살아 있거나 **midaemon** 이 시작된 이후 만들어진 file close 의 축적된 수.
Controls	file 로의 **ioctl** call 의 축적된 수.
Bytes Xfer	이 프로세스가 살아 있거나 **midaemon** 이 시작된 이후 file 로의 모든 reads 와 writes 용으로 사용된 전체 bytes.

③ Memory Management Statistics (Column 3)

Individual Process 스크린의 세번째 column 은 선택된 프로세스에 대한 memory management 정보를 포함한다. 두개의 memory size value 는 private (local to this process only)이나 shared (shared space with other processes)로 표시된 모든 memory region 의 전체 값을 지시한다. Resident size (memory 에 현재 있는 양)과 virtual size (그 프로세스에 할당된 space 의 양)을 나타내는 두 value 는 slash (/)에 의해서 분리된다. Virtual size 에서 모든 space 가 physical memory 에 있지는 않을 것이다. 남아 있는 metrics 는 memory 에 대한 프로세스들의 사용의 지시자(indicator)들이다.

Metric	Definition
Private RSS/VSS	이 프로세스에 의해 소비된 모든 private memory region 의 전체 size. 그 region 에 대한 더 많은 설명이 Memory Regions 스크린에 보여질 것이다.
Shared RSS/VSS	이 프로세스에 의해 소비된 모든 shared memory region 의 전체 size. 그 region 에 대한 더 많은 설명이 Memory Regions 스크린에 보여질 것이다.

Faults Mem/Disk 이 프로세스에 대한 memory (minflt)와 disk (majflt)로의 전체 page fault.

Num Swaps 이 프로세스에 대한 page out swap 의 전체 수.

남아 있는 (remaining) metric 들은 toggle 에 따라서 cumulative 하거나 interval 통계치 중 하나이다.

Metric	Definition
Forks/Vforks	**midaemon** 이 처음으로 시작되거나 그 프로세스 그 자체가 처음으로 fork 된 이래로 그 프로세스가 생성한 fork 나 vfork (children process)들의 수.
Signals Received	이 프로세스에 의해서 받아진 signal 들의 수. 이 counter 는 또한 그 프로세스가 살아있는 동안 받아진 모든 signal 의 축적된 전체값을 유지한다. 예를 들면, 만약 그 프로세스가 완성되기 위하여 terminal read 를 기다리고 있다면, 그것의 Block On reason 은 **TERM** 이 될 것이다.
Mesg Sent/Recd	프로세스에 의해서 보내지고 받아지는 message 들의 수.
Other Log Rd/Wr	모든 non-disk device 로의 physical read 와 write 의 수.
Proc Start Time	그 프로세스가 처음으로 dispatch 될 때의 starting time.

2. Wait States Screen

Wait States 스크린은 그 프로세스가 blocked 된 reason 을 목록화하고, "blocked on" states 사이의 분배(distribution)를 보여준다. (figure 5-20)

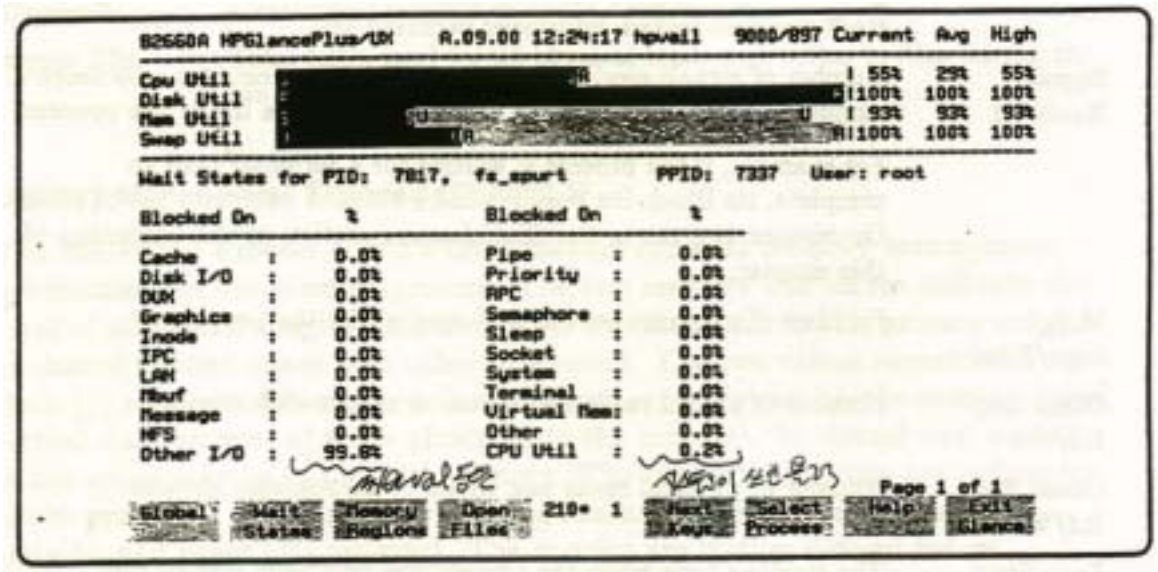


Figure 5-20. Wait States Screen

invent

여러분들은 W 나 Wait States key 를 이용하여 Individual Process 스크린으로부터 이 스크린으로 접속할 수 있다.

스크린의 data section 을 가로지르는 status line 은 모니터 되고 있는 프로세스의 PID 와 이름, Parent PID, 그리고 그 프로세스와 연관된 user name 을 보여준다.

스크린 밑부분의 CPU Utilization value 은 last update interval 동안에 그 프로세스가 unblocked 되게 구동된 시간이다.

3. Memory Regions Screen

이 스크린은 그 프로세스의 virtual address space usage 에 대한 상세화된 정보를 제공한다.(figure 5-1). 스크린 display 의 각 줄은 memory 영역을 나타낸다.

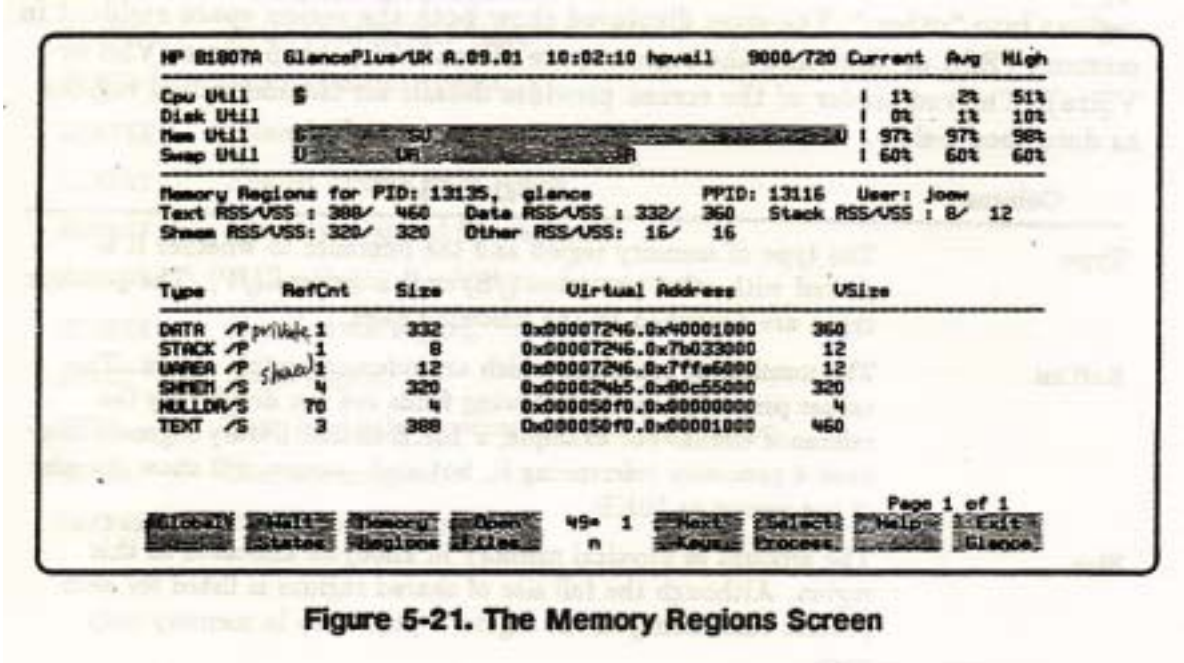


Figure 5-21. The Memory Regions Screen

여러분들은 M 이나 Memory Regions key 를 이용하여 Individual Process 스크린으로부터 이 스크린으로 접속할 수 있다.

① Screen Elements

스크린의 data section 을 가로지르는 status line 은 모니터 되고 있는 프로세스의 PID 와 이름, Parent PID, 그리고 그 프로세스와 연관된 user name 을 보여준다.

다음 5 개의 요소들은 (유형에 관계 없이) text, data, stack, shmem, 그리고 모든 remaining region 의 accumulation 에 관한 region 들을 요약한다. Memory 에 있는 region space (RSS or Size)와 이 프로세스에 할당된 region space (VSS or Vsize) 모두가 전시되는 size 에 보여진다. 스크린의 remainder 는 아래 묘사된 것처럼 각 개개인의 region 에 대한 상세한 정보를 제공한다.

Column	Definition
Type	memory region 의 유형과 그것이 다른 프로세스들과 shared 되었는지 (/S) 혹은 private 인지 (/P)의 참조. 가능한 유형들은 다음 page 에 묘사되어 있다.
RefCnt	이 region 을 참조하는 프로세스들의 수. 아래 필드에 나타난 값들은 이 reference count 에 의해 분리되는 것이 아니다. 예를 들면, 10KB 의 shared library segment 는 아마도 그것을 참조하는 4 개의 프로세스들을 가지지만, 각 프로세스들은 10KB 로 region 의 size 를 보여줄 것이다.
Size	이 region 에 할당된 kilobyte 단위로 나타나는 physical memory 의 양. 비록 shared region 의 full size 가 그것을 참조하는 각 프로세스를 위해 목록화 되었지만, 그 region 은 단지 한번 physical 하게 memory 에 있는 것이다.
Locked	physical memory 로 locked 되어진 (paged out 이나 swapped out 되지 않은) region 의 kilobyte 단위로 나타나는 양. 이 size 는 locking 을 위해 요구되어지는 memory pages 때문에 "Size" 필드를 초과할 것이다. Note: 이것은 단지 Series 800 시스템에서 이용가능하다.
Vsize	kilobyte 단위로 나타나는 그 region 의 virtual (즉, total) size. 이것은 최소 "Size" 필드 만큼 클 것이다.
Virtual Address	그 memory region 의 virtual address (*spaceid.off*)

② Memory Region Types

아래에 memory region 유형들의 완벽한 목록이 있다.

Type	Definition
DATA	Data region
GRAPH	Frame buffer lock page.
IOMAP	Input/Output region
LIBTXT	Shared library text (code) region
LIBDAT	Shared library data region

MEMMAP	Memory-mapped file region
NULLDR	Null pointer dereference shared page.
SIGSTK	Signal stack region
STACK	Stack region
TEXT	Text (code the process is executing) region.
UAREA	User area region
UNKNWN	Region of unknown type.

4. Open File Screen

Open Files 스크린은 examined 되고 있는 프로세스에 대한 open files 에 있는 정보를 목록화 한다. (figure 5-22). 이 정보는 GlancePlus 프로그램이 file 이름을 찾아내기 위해서 disk inode 을 읽어야 할 수 도 있기 때문에 extra overhead 가 요구된다.

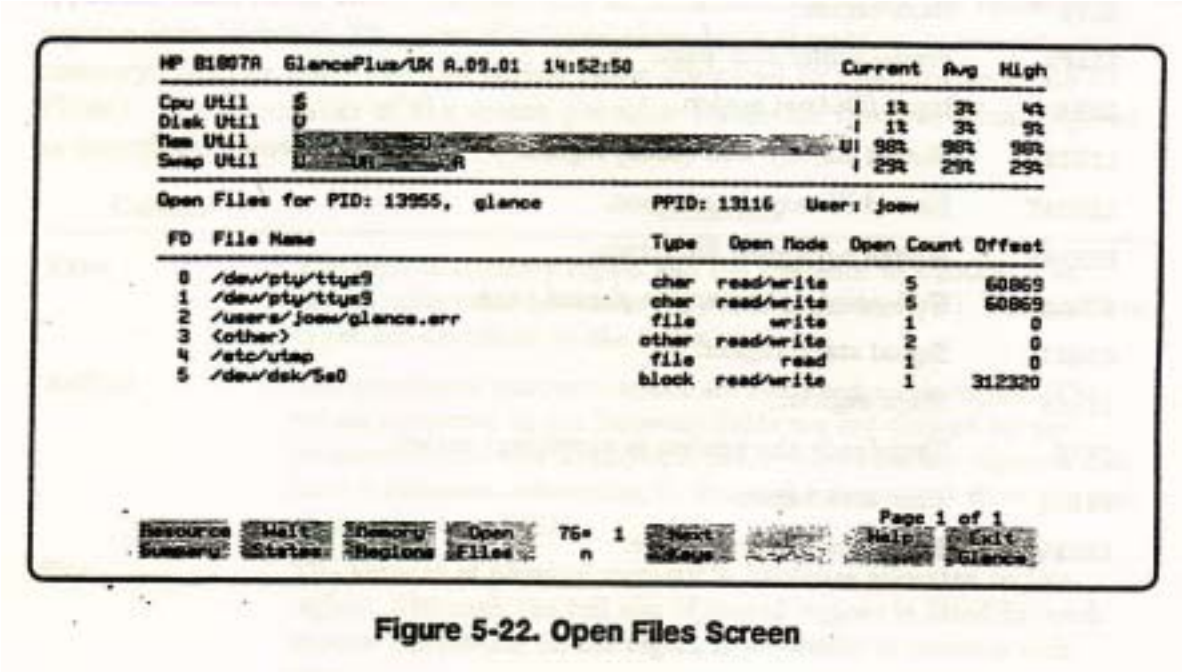


Figure 5-22. Open Files Screen

여러분들은 F 나 Open Files key 를 이용하여 Individual Process 스크린에서 이 스크린으로 접속할 수 있다.

이 스크린의 data portion 의 위에 있는 status line 은 이전 page 에서 했던 것처럼 다른 정보와 함

께 모니터 되고 있는 프로세스의 PID 를 보여준다.

Status line 아래의 data portion 에서는, 각 file 이 한 줄의 정보로 보여진다. Data 의 column 들은 아래에 묘사되어 있다.

Column	Definition
FD	open system call 에 의해 return 된 file descriptor integer. File descriptor 의 sequence 는 일반적으로 그 file 들이 open 된 순서를 따른다. 만약 한 page 에 list 될 수 있는 것보다 더 많은 file 들이 open 되었다면, 추가의 page 들이 사용될 것이다. 이러한 extra page 들은 f, Space Bar, 혹은 Next Page 를 타이핑 함으로써 접속 될 수 있다.
File Name	open file 의 이름. 어떤 file 들은 available 하지 않을 수도 있는데, 이 경우에는 이 필드에 <unknown> 이라고 나타날 것이다.
Type	open file 의 유형은: ● **file**: regular disk file ● **dir**: file system directory inode ● **block**: block device file ● **char**: character device file ● **link**: symbolic link ● **pipe**: data pipe ● **socket**: network socket connections endpoint ● **remote**: remote file (RFA access) ● **other**: unknown file type
Open mode	file 이 reading, writing 혹은 둘다 (read/write)를 위해 open 되었는지 어떤지를 보여준다.
Open count	현재 이 file 이 open 된 횟수. Terminal device 들은 종종 몇 가지 다른 프로세스들에 의해 한번 이상 open 될 것이다.
Offset	다음의 sequential read 나 write 가 그 data 를 대신하는 file 로의 현 byte offset.

6th Customizing HP GlancePlus/UX

HP GlancePlus/UX 와 그것의 다양한 option 들에 친숙하게 되었을 때, 여러분들은 시스템 환경에 맞는 GlancePlus 를 customize 시킬 것이다. 이 appendix 는 여러분의 환경속에서 GlancePlus 의 효율성을 증가시키는 세가지 방법을 묘사한다: start-up 절차의 최적화, 자동적인 **midaemon** start-up, 그리고 overhead impact 의 최소화.

A) Glance Start-Up Options

HP GlancePlus/UX 를 install 한 후, 여러분은 **/usr/perf/bin** directory 내에 있는 glance binary file 을 시작시킴으로써 terminal 에 그 프로그램을 시작 시킬 수 있다. 그 directory 를 여러분 shell 의 path 에 놓음으로써 start-up 프로세스를 더 쉽게 할 수 있다.

몇가지 start-up option 들을 (*see* man-page *glance(1)*) 가지고 여러분의 시스템을 위한 start-up 절차를 customize 할 수 있다.

/usr/perf/bin/glance **[-J interval] [-P [dest]]**
[-maxpages numpages] [-command]
[-nice nicevalue] [-nosort]
[-lock]
[-cnodes <n>]
[-disks <n>]
[-ios <n>]
[-kernel <kernel name>]
[-nfs <n>]
[-pids <n>]

혹은, 만약 여러분들이 여러분들의 path 에 그 directory 를 추가하였다면,

glance [-J interval] [-P [dest]] [-maxpages numpages] [-command]

[-nice nicevalue]

[-nosort]

[-lock]

[-cnodes <n>]

[-disks <n>]

[-ios <n>]

[-kernel <kernel name>]

[-nfs <n>]

[-pids <n>]

where:

-J *interval* 이 따르는, default 값인 5 초를 변화시키면서, 스크린 updates 들 사이의 시간을 미리 setting 할 수 있도록 한다. 예를 들어, **-J 60** 은 자동적으로 update interval 을 60 초로 preset 시킨 것이다.

-P GlancePlus 가 시작될 때 켜져야 하는 print option 을 지시한다. 이것은 오랜 interval 에 걸쳐서 printing 스크린에 유용하다. 만약 *dest* parameter 가 추가되지 않았다면 printing 은 default printer 를 지시한다. *dest* 가 따를 때, output 은 file, printer, 그리고 그 프린터의 class 가 지시된다.

-maxpages p command 를 통해서 프린트 되도록 허락받은 page 들의 최대 수를 변화 시킨다. *numpages* parameter 는 200 page 의 default 최대치를 여러분이 원하는 대로 바꿔준다.

-command 최초의 Global 스크린에서 직접적으로 대체 선택으로 갈 수 있게 한다. 예를 들어, **-w** parameter 의 추가는 GlancePlus 를 Swap Detail 스크린에서 시작 할 수 있게 한다. 다양한 스크린을 불러내기 위해 사용할 수 있는 single-letter command 는 Command Menu 스크린의 오른쪽에 나타나 있다. 이 목록에 접근하기 위해서는 ?를 사용.

-nice 어떤 nice value 에서 glance 를 구동 시킬 것인가를 지시. 음수가 가능하고, default 는 –10 이다. (*see* man-page *nice(1)*).

-nosort Global Summary 스크린에 목록화 되어 있는 interesting 프로세스들을 정렬하지 않도록 (*not* sort)요구할 수 있게 한다. 이것은 CPU overhead 를 줄인다.

-lock	GlancePlus 를 lock 시킨다.
-cnodes <n >	MI Performance Database 의 CNODE Class 에서 사용되고 있는 cnodes sub-class entry 들의 수를 변화시킨다. *(see* man-page *midaemon (1)*)
-disks <n>	MI Performance Database 의 DISK Class 에서 사용되고 있는 disk sub-class entry 들의 수를 변화시킨다. *(see* man-page *midaemon (1)*)
-ios <n>	MI Performance Database 의 IO Class 에서 사용되고 있는 io sub-class entry 들의 수를 변화시킨다. *(see* man-page *midaemon (1)*)
-kernel <kernel name>	midaemon 에 kernel name list 를 시스템에서 부팅되는 실행가능한 file 로 특성화 시킨다. *(see* man-page *midaemon (1)*)
-nfs <n>	MI Performance Database 의 NFS Class 에서 사용되고 있는 nfs sub-class entry 들의 수를 변화시킨다. *(see* man-page *midaemon (1)*)
-pids <n>	MI Performance Database 의 PID Class 에서 사용되고 있는 pid sub-class entry 들의 수를 변화시킨다. *(see* man-page *midaemon (1)*)

예를 들어, 만약 밤새도록 시스템 사용율을 모니터하기를 원한다면, 다음과 같은 string 으로 그 tool 을 시작할 수 있다.:

glance –J 3600 –P

이것은 GlancePlus 를 시작해서 다음 모니터링을 하고 그 프로그램을 빠져 나올 때까지 시간당 (3600 초) 한번 프린트를 할 것이다. 만약 좀더 특별히 하기를 원한다면, 아래의 string 을 이용할 수도 있다.

glance –J 7200 –P –maxpages 300 –w

이것은 Swap Space Utilization 스크린으로 그 tool 을 시작하고, 시스템의 특성을 파악하고, 여러분이 그 프로그램을 빠져 나오거나 혹은 최대 300 page 가 프린트 될 때까지 매 2 시간 단위로 그 스크린을 인쇄할 것이다.

B) Automatching midaemon Startup

Measurement Interface Daemon 은 지속적으로 성능 정보를 모집하는 low-overhead 프로세스이다. 그것은 HP GlancePlus/UX 가 기능을 수행하기 전에 구동되야 하는데, 그 프로그램이 **midaemon** 이 그 정보를 어디에 썼는가 (write)하는 shared-memory segment 를 읽기 때문이다. 동일한 shared-memory segment 가 HP LaserRX/UX 성능 분석 product 의 **scopeux** collector 에 의해서 사용된다.

Note 만약 HP LaserRX/UX 가 설치되어 있고, starting **scopeux** 을 위한 function call routine 이 *HP LaserRX/UX User's Manual: Collection Software* 의 Chapter 2 에서 권유한대로 **/etc/rc** 에 추가 되었다면, 이 section 의 나머지 부분을 무시하라.

여러분들은 Glance 가 그것이 구동하는 매 시간 **midaemon** 을 시작하게 할 수 있지만, 만약 midaemon 이 지속적으로 구동한다면 GlancePlus 는 좀더 빨리 시작할 것이다. 만약 프로세스가 CPU time 을 사용한다면, **midaemon** 은 그것이 시작한 이후의 초기화된 프로세스 혹은 그것이 시작하기 전의 초기화된 프로세스를 인지할 것이다. 만약 midaemon 이 다른 시스템 daemon 들 보다도 일찍 시작되었다면, 그러한 시스템 daemon 들은 비록 그것들이 CPU resource 를 이용하고 있지 않을 지라도 GlancePlus threshold 가 "All Processes"로 세팅 되었을 때 보여질 것이다.

GlancePlus 가 **midaemon** 을 시작하기 위해서 사용된다면, 그것은 GlancePlus 가 끝날 때 shutdown 될 것이다. **midaemon** start-up 을 **/etc/rc** file 에 추가하는 것은 GlancePlus 를 더 빨리 시작 시킬 뿐 아니라, **midaemon** 으로 하여금 GlancePlus 가 종료된 이후에도 지속적으로 구동하도록 할 수 있다.

시스템을 시작할 때 **midaemon** 을 invoke 시키기 위해, **/etc/rc** 에 새로운 function 을 추가하라:

```
start_mi ( )
{
# Start-up the Measurement Interface for HP GlancePlus/UX
/usr/perf/bin/midadmon
```

}

이제 여러분은 **start_mi** function call 을 **/etc/rc** script 의 executable part 에 추가해야만 한다. HP-UX 에 제공되는 **/etc/rc** file 에서 **swapper** 가 시작된 이후 (swap_start function call) 이 function call 을 위치하라.

적당한 위치선정은 **/etc/rc** 내에서 시작되는 프로세스들에서 성능 data 가 수집될 수 있도록 확실히 해 준다.

Note	**start_mi** function call 을 **/etc/rc** 내에 적절하게 위치시키는 것은 매우 중요하다. script 가 수행될 때, 그것은 **midaemon** 프로세스를 시작 시킬 것이다. 이 지점에서 **midaemon** 은 시스템에서 구동중인 프로세스들에 대한 data 를 수집하기 시작할 것이다.

C) Minimizing Performance Analysis Overhead

시스템에 대한 성능을 분석할 때, 여러분들은 HP GlancePlus/UX 에 의해 생성된 overhead 를 고려해야만 할 수도 있다. 시스템 성능을 모니터하는 어떤 프로그램은 그 시스템에 특정 양의 overhead 를 부과할 것이다. 수행 프로그램이 시스템 자원의 이용을 피할 수 없을 때, bottleneck 된 자원을 측정하는 프로세스가 더 심각한 bottleneck 을 유발 할 수 있는 가능성이 존재한다.

아래의 설명들은 GlancePlus product 가 생성하는 overhead 를 이해하고, 시스템의 요구를 만족시키기 위해 overhead 를 최소화시킬 수 있는 방법들을 이해하는데 도움을 줄 것이다. Glance 프로그램이나 midaemon 둘 다 substantial I/O 가 아니므로, 각각에 대한 overhead 은 memory 나 CPU 에 대한 요구(demand)로 나타날 것이다.

1. Memory Overhead

midaemon 프로그램은 **glance** 프로그램에 읽혀지는 shared memory 내의 memory-resident structure

를 할당하고 쓴다. 이 structure 의 size 는 시스템이 configuration 에 따라 매우 다양하다. 예를 들어, **nproc** 의 구성된 value 는 할당된 PID subclasses 의 default 수를 정의하는데, 그곳에서 각 PID subclass 는 shared-memory segment 에 있는 data 의 block 을 사용한다.*(see* man-page *glance(1)).* nproc 에 대한 매우 큰 value 는 초과될 수 있는 default memory 요구를 유발 할 수 있다. 전형적인 diskless clients 에서 이러한 segment 의 전체 size 는 100 KB 이하 일 것이다. Series 800 시스템에서는, shared memory size 가 500 KB 가 된다. **Glance** 프로그램 자체의 memory 요구 (requirement)는 이 segment 의 size 에 비례하여 증가한다.

midaemon 은 그 자체의 프로세스를 memory 로 *lock* 한다. 이러한 할당의 size 은 여러분 시스템을 의존하고 **midaemon** 이 정의하는 subclass structures 에 의해서 결정된다. 전형적인 시스템에서 1 MB 보다 약간 큰 memory 가 프로그램들에 의해 할당된다.

GlancePlus 를 시작할 때 그리고 시스템이 available lockable memory 를 부족해 할 때 **–lock** option 을 사용한다면, 여러분이 glance 프로그램을 시작할 때, 다음과 같은 message 를 볼 것이다: **could not lock or unable to allocate memory or swap space.** 이러한 일이 일어난다면, -lock option 사용을 멈추어라.

2. CPU Overhead

midaemon 은 두가지 방법으로 CPU overhead 를 생성한다. 그것은 HP-UX kernel 에서 **ki tracing** 으로 불리는 특별한 subsystem 을 지칭한다. 이 subsystem 은 특정 kernel routine 들로부터 trace buffer 로 정보를 log 한다. 이것은 어떤 시스템 call 들이 수행하는데 걸리는 시간을 약간 증가시킬 수 있다. 이 subsystem 은 **open** 과 같은 느린 시스템 call 보다는 **gettimeofday** 와 같은 빠른 시스템 call 들에 더 많은 영향을 미친다. 이런 overhead 가 모든 active process 에 어느정도 영향을 미치므로, 이것은 Glance 에 attributable 한 것으로 나타나지 않는다. 일반적인 시스템에서 이것은 전체 CPU time 의 0.1 % 이하로 측정 되었지만, 매우 높은 syscall rate 을 가진 시스템에서는 이 overhead 가 5% 정도 만큼 증가할 수 있다.

midaemon 은 또한 shared-memory segment 에서 그것이 유지하는 counter 들을 updating 하는 동안에 CPU time 을 사용한다. 사용된 CPU time 의 양은 얼마나 많은 data 가 kernel trace buffer 들에 log 되었는지에 달려 있다. 시스템 call 들은 이 tracing 의 대부분을 책임지기 때문에, **midaemon** 의 CPU overhead 은 시스템에서 일어나는 syscall 들의 수에 직접적으로 비례한다. 일반적인 시스템은 **midaemon** CPU overhead 가 CPU time 의 3 % 이하를 기록하지만, 높은 syscall rate 을 가진

시스템들은 10 %로 이 overhead 가 증가 할 수 있다.

glance 프로그램은 스크린들을 update 하고 user input 에 응답할 때 CPU time 을 이용한다. 일반적으로 overhead 는 CPU time 의 5 % 이하이지만, 만약 많은 command 가 들어왔거나 update interval 이 매 5 초보다도 자주 일 때 증가할 수 있다. Update interval 을 10 초나 그 이상으로 조절하는 것은 data 의 손실 없이 overhead 를 감소 시킬 것이다.

GlancePlus 는 subsequent strarup 의 속도를 맞추기 위해 /usr/perf/log/'hostname' directory 내에 **milib.data** 와 **midaemon.data** file 들을 생성할 수도 있다. 이것은 자동적으로 행해지며 어떤 user action 도 필요하지 않다.

3. Reducing Overhead

시스템 성능을 모니터 하는 동안에 Glance 의 overhead 는 주로 memory 와 CPU time 에 대한 요구 때문이고, 개개의 시스템에 따라 다양하다. 만약 Glance 의 CPU overhead 가 너무 높다고 생각되면, j command 를 사용하여 update interval 을 증가시킴으로써 그것을 낮출 수 있다. 추가적으로 **–nosort** start-up option 은 interesting 프로세스들의 정렬을 disable 시키는데 사용될 수 있다. 이것을 하기 위해 사용된 하나의 방법은 자동 스크린 update 사이의 시간을 매우 길게, 예를 들어 600 초, 만들기 위해 j command 을 사용하는 것이다. 그리고, update 가 필요할 때, Return key 를 이용하여 수동으로 그것을 요청하라.

만약 **glance** 와 **midaemon** resident memory demand 가 너무 높다고 생각 되면, 아래를 수행하여 그것을 낮출 수 있다.

- 이 appendix 의 **automating midaemon startup** 설명에서 언급한 것처럼 **/etc/rc** 로부터 **midaemon** 을 시작시킨다.
- **/etc/rc** script 내에 있는 default 보다도 더 낮은 value 에서 passing 함으로써 **midaemon** 에 의해 결정된 subclass data area 의 수를 줄인다. (midaemon parameter 에 대한 정보를 위해 midaemon man-page 를 참조)

7th Prompts and Messages

이 Chapter 는 Glance 를 구동하는 동안에 여러분이 부딪칠 다른 prompts 와 messages 의 의미를 묘사한다. 오류 (error)의 원인과 위치에 대한 추가적인 정보는 user 의 home directory **glance.err** file 에서 찾을 수 있다.

Initializing …

이 message 는 GlancePlus 를 시작할 때 나타난다. 만약 midaemon 이 이미 구동되고 있지 않다면, GlancPlus 는 midaemon 을 시작시켜야 하고 그것이 통계치를 생산하기 전에 두가지 샘플의 data 를 수집한다. 이것은 잠시의 시간이 걸린다.

Could not lock

이것은 단지 정보를 주는 message 일 뿐이다. GlancePlus 를 시작할 때 –lock option 을 사용하였을 때 나타나고, GlancePlus 는 그것의 data structure 를 memory 로 lock 할 수 없다. 비록 이것은 GlancePlus 가 구동하지 못하게 하나, 그 시스템에 대한 memory pressure 는 GlancePlus data 가 swapped out 되게 하는 원인이 된다. (not locked into memory). 이러한 상황에서 GlancePlus 는 poor response time 을 가질 것이다.

Glance 의 memory requirement 는 얼마나 많은 data 가 process, disk, cluster node 등을 위해서 저장되야 하는지에 따라 시스템 마다 다르다. 대부분의 시스템에서 GlancePlus 는 800KB 에서 1.4 MB 의 memory 를 필요로 한다.

dmesg command 를 사용하여 여러분의 시스템에 구성된 lockable memory 가 얼마나 되는지 체크해보라. (*see* man-page *dmesg (1M)*) . dmesg output 에서 여러분은 memory 의 양 (byte 단위로)을 지시하는 message 를 볼 것이다. 예를 들면, **lockable mem = 13455360**. 이것은 이용가능한 lockable memory 가 약 13 MB 라는 것이다 – ample space for running Glance.

예를 들어, 만약 dmesg message 가 **lockable mem = 993280** 이라면, 이용가능한 lockable memory 가 1 MB 이하이고, 아마도 GlancePlus 을 시작할 때 **could not lock** message 을 가질 것이다.

GlancePlus 가 locked 되게 구동시키기 위해서, 더 많은 lockable memory 를 제공하기 위해서 kernel 을 재구성 할 수도 있을 것이다. (*HP-UX System Administration Tasks manual*, under *unlockable_mem.*을 참조)

A) Start-Up Failure Messages

1. Sorry, you must be a superuser

GlancePlus binary file (**/usr/perf/bin/glance**)는 **setuid root** permission 을 가지고 install 된다. permission 이 **–rwsr-sr-x** 가 되었는지 확실히 하기 위해 **ls –l /usr/perf/bin/glance** 를 해보고, 그 owner 는 root 이다.

2. Sorry, I need to know a more specific terminal type than "unknown"

이것은 curses library 가 여러분의 terminal type (TERM variable)에 decipher 하지 않거나, 지원하지 않은 terminal 인 경우를 의미한다. (*see* man-page *curses(3X)*)

3. Unable to access /usr/perf/bin/midaemon

GlancePlus 는 반드시 midaemon file 에 접속할 수 있거나 수행할 수 있어야 한다. /usr/perf/bin/glance file 이 setuid root 인지 확인하기 위해서 /usr/perf/bin/midaemon 의 permission 을 체크하라.

4. Unable to allocate memory/swap space

다른 프로그램들처럼, GlancePlus 도 그것을 수행할 수 있기 전에 특정 양의 memory 와 swap space 를 가져야만 한다. GlancePlus 의 memory requirement 는 그것이 active device 들과 프로세스들에 대한 통계치를 기록하기 때문에 상당히 높다. 만약 stressed 시스템에서 GlancePlus 를 구동시켜야만 한다면, stress 전에 그 프로그램을 시작하고 load 가 심각해질 때까지 그것을 구동시켜라. 만약 –lock option 을 사용했더라면, 그것을 사용하는 것을 멈춰라.

5. Unable to find /usr/perf/bin/midaemon

/usr/perf/bin/midaemon file 은 GlancePlus 가 동작하기 위해서 반드시 존재해야 한다. 이 file 은 Glance 가 install 될 때 다른 프로그램 file 들과 함께 생성된다. 이 file 이 제거 되었는지를 체크해라.

6. Unable to initialize MI

이것은 MI (measurement interface) library call 을 가진 심각한 문제가 있다는 것을 지시한다. 도움을 위해서 HP support representative 에 전화를 걸어라. 이런 message 는 file /hp-ux 보다는 HP-UX kernel 을 사용하고 그 이름을 특성화 (specify)하지 않았을 때 나타날 수도 있다.

7. Unable to initialize terminal

GlancePlus 는 이런 유형의 terminal 에서 작동할 수 없다. 그것은 또한 curses 내의 시스템 call 이 fail 날 때 fail 날 것이다.

8. Unable to start midaemon

어떤 이유에서 midaemon 이 구동하지 않는다. Root 로 login 하고 **/usr/perf/bin/midaemon &**을 타이핑하라. 만약 midaemon 이 끝나지 않고 수행되면, GlancePlus 는 오류없이 구동할 것이다.

만약 midaemon 이 즉각적으로 끝난다면, man-page *midadmon(1)*에서 주어진 정보를 가진 termination message 에 있는 return code 를 체크하라.

B) Fatal Errors

이러한 error 들은 발생해서는 안된다. 만약 발생했다면, 여러분의 terminal 은 immobilized 된 ("hung") 걸 것이다. Terminal setting 을 잃었거나 (lost), shell 이 더 이상 Return 을 이해하지 못한 것이다. 회복하고 정상적인 동작으로 shell 을 되돌리기 위해, **^Jtset^J** 를 타이핑하라. (^J 는 Ctrl-j 를 의미)

오류가 발생했을 때, internal debugging 정보가 user 의 home directory 내 **glance.err** 라는 file 에 쓰여 질 수 도 있다. 이 file 은 software 의 예상가능한 문제를 치유하기 위해 HP Support personnel 에 의해 사용될 수도 있다.

- Error initializing terminal sub-window
- Error reading from terminal
- Error refreshing terminal screen
- Error writing to terminal

이러한 message 들 중의 하나는 그것이 "Curses"를 구동하는 terminal 과 통신하는데 문제를 가진다는 것을 지시하고, Glance 의 terminal-handling 시스템은 error message 을 되돌린다.

1. Error retrieving MI data

아마도 midaemon 이 죽은 경우이다. Midaemon 이 구동되는지 체크하기 위해 **ps –ae** 를 사용해라. 만약 그것이 있다면, 더 많은 정보를 위해서 **Unable to start midaemon** message 아래 list 된 debugging procedures 를 관찰하라.

2. Unable to allocate memory/swap space

malloc 과 같은 시스템 call 은 swap space 나 memory 의 부족 때문에 프로그램 내에서 fail 됐다. swap space 를 추가하고 Glance 를 다시 구동시키기 위해 SAM 을 이용해라. 초기화 동안에 이 message 가 나타나면, 이전 page 의 설명을 참조하라. Open Files 스크린의 세번째 page 에 이 message 가 나타나면, 이 chapter 의 "Appearing on Single Process Screen"을 참조해라. Glance 를 시작하는데 그것을 특성화 했다면 –lock 을 사용하는 것을 끊어라.

C) Printing-related Messages

1. Single screen or continuous printing (s/c) ?

GlancePlus 가 현 스크린을 프린트 할 것인지 혹은 다음 스크린 update 를 시작하기 위해서 지속적인 프린팅을 toggle-on 할 것인지을 request 할 수 있다. 현 스크린을 프린트 하기 위해서는 "s"를, 지속적인 프린팅을 toggle-on 하기 위해서는 "c"를 치면 된다.

2. Print to device or file (d/f)

device 를 프린트 하기 위해서 "d"를 치면, print device 를 넣으라는 질문을 받는다:

3. Enter print device for lp -d option (system default):

프린팅이 Print command (p)와 함께 toggled on 되어질 때 이 message 가 나타난다. 만약 Return 을 쳐서 이 prompt 에 응답하면, 그 시스템은 프린트 하기 위해서 그것의 default printing device 을 사용한다.

4. Enter print file name (~/glance.print) :

File 을 프린트 하기 위해서 "f"를 입력하면, file name 을 입력하라는 질문이 떨어진다.

5. Maximum of n pages of output reached, printing disabled

종이의 낭비를 막기 위해서 만약 우연히 p 가 toggled-on position 에 남아 있다면, 한번에 프린트 될 수 있는 default 최대값은 200 page 이다. Maxpages option 을 가지고 default 를 overriding 함으로써 허용될 수 있는 page 의 수를 증가 시키거나 감소 시킬 수 있다. (6 장, " GlancePlus Start-Up Options"를 참조)

6. Printing ….

이 message 는 여러분에게 Print option (p)이 toggled-on position 에 있다는 것을 알게끔 해준다. 프린팅을 멈추기 위해서 p 를 다시 한번 친다.

7. Printing disabled

이 message 는 Print option (p)가 toggled off 되었다는 것을 알게 끔 해준다.

D) Appearing on Global Screen

1. No processes exceed current threshold settings

이 message 는 현재 어떤 프로세스도 interesting 으로 qualify 되지 않았다는 것을 의미한다. 여러분은 아마도 value 를 낮추기 위해서 threshold 값을 reset 시키기를 원할 것이다.

E) Appearing on Threshold Options Screen

1. Are the above thresholds corrects (y/n) ?

이 prompt 는 여러분이 o-screen 에서 선택한 threshold 값을 확인한다. Single character answer 로 응답한다: y or n

2. Invalid format, enter number in format x.xx or 'all' :

CPU, disk rate, 그리고 resident set size 를 위한 숫자상의 threshold prompt 를 입력하기 위해서 numbers (integers)나 key word "all" 만을 사용할 수 있다. 만약 다른 input 을 넣었다면 이런 message 들이 나타날 것이다. 이러한 integer requirement 때문에 만일 여러분들이 threshold options 스크린을 위한 Online Help 에 접속하고자 한다면, 어떤 ASCII data 를 넣기 전에 Help 를 눌러야 하는데, 그렇지 않으면 그 프로그램은 function key 를 data 로 고려할 것이기 때문이다.

3. TTY device not found, enter full path name or 'all':

Specified tty device 가 발견되지 안았다. 단지 **ttyv9** 대신에 **/dev/pty/ttyv9** 와 같은 full path name 으로 data 를 입력하도록 한다.

4. User not found, enter user name or 'all':

/etc/passwd file 에서 specified user name 이 발견되지 않았다.

F) Appearing on Online Help Screen

1. Enter 'e' or F8 to exit online help

이 prompt 는 모든 online help subwindow 의 아래부분에 나타난다. 어떤 entry 든 online help 를 빠져 나가도록 하는 것이다. 연속으로 e key 를 두 번 누르지 않도록 하라, 만약 그러면 그 프로그램을 빠져 나가게 될 것이다.

2. Help file not available

/usr/perf/help/C directory 에 있는 glance.help file 은 online help text 를 포함한다. 이 file 이 purged 되지 않고 읽을 수 있는지 확인하라.

G) Appearing on Commands Menu Screen

1. Enter command or function key:

Command Menu 스크린에 접속할 때마다 나타난다. Single-letter command 나 function key 이용.

H) Appearing When Entering PIDs or Update Intervals

1. Enter integer data only, re-enter PID:

알파벳 data 를 입력했다. 프로세스를 선택하기 위해서 반드시 프로그램 이름이 아닌, 숫자로 된 PID 을 입력해야 한다. Request 를 취소하기 위해서, Cancel 을 사용.

2. Enter integer data only, re –enter interval:

알파벳 data 를 입력했다. Update interval 을 바꾸기 위해, 반드시 숫자를 사용해야 한다. Current interval 을 남겨 놓고 싶으면 Return 을 이용.

3. Enter PID:

상세히 보고자 하는 프로세스의 process identifier number 를 입력.

4. Enter update interval in seconds (n) :

스크린을 update 하기 전에 GlancePlus 를 멈추고자 하는 시간을 입력. Parentheses 에서 숫자 (n)은 current update interval 이다. 현재 number 를 남겨 놓고 싶으면 Return 을 이용

5. Invalid PID , enter new PID:

입력된 PID 가 현재 수행중인 프로세스에 상응하지 않거나, MI 가 그 PID 를 인지하지 못한 경우. 이것은 (1) 프로세스가 midaemon 이 시작되기 전에 생성되었거나 (2) midaemon 이 시작된 이후로 그 프로세스가 어떤 CPU time 도 사용하지 않았을 경우 일어 날 수 있다.
만약 모든 시스템 daemon 들을 모니터 할 수 있기를 원한다면, chapter 6 에서 언급한 것처럼, **/etc/rc** file 과 함께 midaemon 을 시작하면 된다.

6. Too many digits, re-enter up to 5 digits:

Update interval 은 최대 5 digits (32,767 seconds)까지 한계되어 있다.

7. Update interval out of range (2 – 32767 seconds), re-enter:

Update interval 은 2 와 32,767 초 사이의 수가 되야 한다. 이 범위를 벗어난 어떠한 수도 수용 가능하지 않다.

I) Appearing on Single Process Screen

1. Process no longer executing

모니터하고 있는 프로세스가 끝났고, 통계치는 이용가능하지가 않다.

2. Unable to allocate memory

GlancePlus 는 open files 의 큰 숫자를 가지는 프로세스에 대한 open-file 정보를 제공하지 않을 수도 있다.

J) Appearing on Diskless Server Screen

1. This system is not a diskless server

Diskless cluster 의 부분이 아닌 시스템에서 GlancePlus 를 구동하고 있거나, 그 시스템이 diskless client node 이다.

K) Appearing on Disk Detail, Disk I/O, and Queue Lengths Screens

1. Diskless Client : no disk I/O data available

Diskless client 에 대한 모든 I/O 가 server 에서 일어나기 때문에 이 스크린에 이용가능한 어떠한 disk I/O data 도 없다.

L) Appearing on LAN Detail Screen

1. No LANs found on this system

시스템에 LAN card 가 구성 (configured)되어 있지 않다.

M)Appearing on NFS Detail Screen

1. No NFS Activity

여러분의 시스템은 다른 시스템으로부터 NFS-mount 된 disk 를 접속할 수 없고, 다른 어떤 시스템들도 현재 여러분의 시스템의 NFS-mount 되었을 수도 있는 disk 를 접속할 수 없다.

N) Appearing on Swap Detail Screen

1. No local swap disks found

이 message 는 여러분이 local swap disk 없는 diskless client 일때 나타난다.

O) Appearing on Logical Volumes Screen

1. No logical volumes found

이 message 는 만약 여러분이 어떤 logical volume 도 가지지 않다면 나타난다.

P) Appearing After Pressing Ctrl – c

1. Continue execution (y/n)?

Interrupt character 를 친 경우. y 는 GlancePlus 로 돌아가는 것이고 n 은 끝내는 것을 의미.

8th Glossary

- **alternate function key sets**

첫 스크린에 나타난 최초의 셑으로부터 enable 된 3 개 셑의 function keys (*softkeys*). Glance 에서 alternate function keys 는 F5 나 Next Keys 를 치면 활성화 된다. (see also **softkeys**)

- **banner line**

display 스크린의 꼭대기를 가로질러 나타나는 한 줄의 text. 이것은 product 의 name, vision number, current time, 시스템 designation, 그리고 시스템 type 을 포함한다.

- **block I/O buffer**

File system 입력과 출력을 통해 disk 로부터 block-mode device , character-mode device 나 raw-mode device 에 반대되는 , 까지 전송되고 있는 data 를 저장하기 위해서 사용되는 buffer.

- **block I/O operation**

Block-mode device 에서 행해지고 있는 operation (read, write, 혹은 mount 와 같은)

- **block mode**

Buffered input/output. Data 는 buffer cache 에 묶여 있고, 후에 fixed-size block 으로 전송된다. Block 에서 data 를 전송하고 수신하는 hardware device 가 block-mode device 이다. **Character mode** 와 비교.

- **block on**

마지막으로 저장된 프로세스 block 을 위한 이유 (reason). Blocked state 혹은 blocked-on state 라 불리기도 한다.

- **block size**

File system 을 위해 사용된 정보의 primary unit 의 size. 이것은 file system 이 생성될 때 세팅된다.

- **blocked state**

See block on

- **bottleneck**

시스템 자원이 그 용량을 초과하는 요구를 만족 시키지 못할 때 발생하는 상황. 자원이 bottleneck 에 걸렸다고 한다. "bottlenecked" . bottleneck 은 시스템의 성능 저하를 가져온다. Bottleneck 의 주요 특징은, 그것이 동시에 모든 자원들에게 일어나지 않는다는 것이다; 다른 자원들을 충분히 이용하지 못한 채.

- **buffer**

Input/output operation 용으로 사용 될 때까지 code 나 data 를 간직하기 위해 사용되는 일시적인 memory 저장 장소.

- **buffer cache**

Memory 에 일시적으로 저장된 한 개 이상의 block input 과 output buffer 들의 모음. CPU 에 의해서 요구된 data 를 cache 가 잡고 있기 때문에, disk 로의 I/O 수는 감소 한다. File system buffer cache, buffer cache pool, 혹은 단순히 buffer pool 이라고도 한다.

- **buffer header**

File system buffer cache 에서 buffer 쪽으로 모든 block I/O operation 에 의해 사용된 entry 들

- **buffer pool**

See buffer cache

- **cache**

See buffer cache

- **cache efficiency**

어떤 buffered read 와 read-ahead request 가 이미 cache 에 있는 데이터에 의해서 만족될 수 있는지의 정도.

- **cache hits**

buffer cache 에 이미 있는 data 에 의해 만족되는 read 와 read-ahead requests. See also **cache efficiency**.

- **character mode**

Block 단위보다는 byte 단위로 (byte-by-byte) 수행되는 data 전송. Printer, plotter, 그리고 terminal 이 character-mode device 의 예들이다. Raw mode 라고도 한다. **Block mode** 와 비교.

- **child process**

Fork 나 vfork 시스템 call 을 통해, 다른 활성화된 프로세스의 요구 때 생기는 새로운 프로세스. 요구를 하는 프로세스는 parent process 이다.

- **client**

Server 로부터 서비스를 요구하는 시스템. Diskless cluster 에서는, client 가 server 의 disk 를 사용하고 그 자신의 것이 없다. NFS 의 경우는 client 가 physical 하게 다른 시스템에 있는 file system 을 mount 한다. (the Network File System server)

- **cluster**

LAN 에 의해 연결된 (linked) 하나 이상의 work station 들, 그러나 단지 하나의 root file system 을 가진다.

- **cluster server processes (CSPs)**

cluster 에 있는 장치에서 구동하고, remote cnodes 로부터의 요구를 다루는 특정 kernel process.

- **cnode**

Diskless system 에서의 client. Cnode 라는 이름은 "client node"에서 나온 것이다.

- **context switching**

하나의 프로세스에서 다른 것으로 변화시키는 (dispatch) 행위

- **context-sensitive help**

Glance 에서 보여지는 스크린에 직접적으로 연결되는 online help 정보. h 나 Help 에 의해서 접속 가능.

- **cyclical redundancy check (CRC)**

A networking checksum protocol.

- **daemon**

유용한 activity 를 수행하면서 background 에서 지속적으로 구동하는 프로세스. 예를 들어, **midaemon** 은 시스템 activity 를 지속적으로 모니터하는 daemon 이다.

- **data locality**

Associated data 혹은 file 에 연관된 data 나 file 의 위치. 만약 서로 근처에 위치해 있다면 좋은 data locality 를 가진다고 한다. 왜냐하면 접속이 적은 수의 page 에 제한되어 있고, 그 data 는 memory 에 있기 쉽기 때문이다. Poor data locality 는 associated data 가 반드시 다른 data page 들로부터 취해져야 한다는 것을 의미한다.

- **data segment**

프로그램의 static, dynamic data 를 정렬하기(sort) 위해서 reserved 된 한 section 의 memory.

- **device file**

hardware device 에 직접적인 접속 (access)을 허락하는 특별 file.

- **device drives**

peripheral device 와 수행하는 프로세스 사이에 가장 낮은 level 의 input 과 output 을 다루는 kernel routines 과 data structures 의 모음.

- **device swap**

See swap space, device

- **direct-mode physical disk I/O**

Raw- 혹은 character-mode device file 로의 input 혹은 output.

- **dirty buffer**

Data 가 쓰여진 (write) memory buffer, 하지만 그 buffer 로부터 아직 data 는 쓰여지지 않았거나 disk 로 "flushed " 되지 않았다.

- **diskless cluster server**

Diskless client node 를 위한 disk activity 를 지원하는 disked 시스템.

- **diskless network packet exchange**

diskless cluster 에서 request 와 service 들을 교환하는데 사용된 LAN data packet 들. **Diskless transfer** 라 불리기도 한다.

- **dispatch priority**

이 우선 순위는 CPU 가 프로세스들을 구동시키는 순서를 – 즉, 어떤 프로세스가 실행될 것인지를 – 결정

- **exec name**

프로세스를 위해 실행가능한 code 를 load 하기 위해 사용된 마지막 file name.

- **file descriptor**

Open file 의 한 instance 를 track 하기 위해서 사용된 한 셑의 data.

- **file lock**

한명 이상의 user 에 의해 특정 file 로의 동시 접속을 막는 방법.(See man-page *fcntl(2)* and *lockf(2))*

- **file system**

hard disk 에 있는 file 과 directory 들의 organization 과 placement. 이 file system 은 file 들을 naming 하고 이러한 file 들로의 접속을 조절하기 위한 OS 시스템의 facility 를 포함한다.

- **file system buffer cache**

See buffer cache

- **file system activity**

file system 에 의해 열린 (opened) block I/O files

- **file system swap**

See swap space, file system

- **filter**

Glance 에서 Global 스크린에서 보여지는 프로세스들은 특정 user name, program name, 혹은

terminal connection 으로 제한 하는 수단.

- **fork**

한 프로세스가 그 자체를 두개의 동일한 프로세스들 – parent process 와 child process –로 복사할 수 있게 하는 시스템 call. Vfork 시스템 call 과는 달리, 생성된 child process 는 parent 프로세스의 memory 와 control 로의 access 를 가지지 않을 것이다. See also **vfork**

- **fragment**

file 에 할당 될 수 있는 정보의 가장 작은 단위. File system 은 full block 이나 하나 이상의 fragment 로 disk space 를 file 로 할당할 수 있다. 만약 full block 의 size 가 크다면, fragments 로의 할당은 손상된 space 를 줄일 수 있다. Fragment 의 size 는 file system 이 생성될 때 세팅된다.

- **global bars**

GlancePlus display 스크린의 banner line 바로 밑에 나타나는 4 개의 highlighted 된 band. 이 bar 들은 4 개의 자원들 –CPU, Disk, Memory, 그리고 Swap Space –을 그 시스템이 어떻게 사용하고 있는지를 보여준다.

- **HP-UX Measurement Interface**

Midaemon 이 성능 정보를 GlancePlus 에 제공하기 위해서 수행하는 counter 들이나 library call 들의 셑.

- **idle loop**

구동할 수 있는 (runnable) 프로세스를 찾을 수 없을 때 CPU 가 수행하는 것.

- **in-core inode cache**

자주 사용되는 file system 정보를 저장하는 memory-resident table.

- **inbound read/write**

remote 시스템에서 구동하는 프로세스가 local 하게 mount 된 disk 로 부터의 read 나 disk 로의 write 을 request 하기 위해서 NFS 를 사용할 때 사용되는 designation. See also **outbound read/write.**

- **inode**

file 을 묘사하기 위해서 file system 에 의해서 사용된 data structure. 이것은 data block 이 어디에

위치하는지, file 의 유형, data 로의 pointer 들, owner, group, 그리고 protection information 에 대한 정보들을 포함한다. Inode 라는 말은 "index node" 로부터 나온 것이다.

- **interesting process**

Glance 의 정의된 threshold 들 중 하나를 충분히 초과하는 active 한 프로세스.

- **InterProcess Communication (IPC)**

프로세스들 사이에서 사용되는 communication protocol 들 (see man-pages *msgop(2), semop(2), shmop(2)*)

- **interrupt**

CPU 가 그것이 수행하고 있는 것을 멈추게 하고, I/O complements 와 같은 다른 특정 프로세싱 작업을 수행하도록 하게 하는 event.

- **interrupt time**

시스템에 의해 interrupt 가 서비스되는 동안의 시간.

- **I/O driver**

선택된 device 로나 (to) 부터 (from) data transfer 를 지원하는, kernel 에 attach 되어 있는, code 의 특별한 section 들.

- **I/O redirection**

File 이나 device 로나 (to) 부터 (from) 프로그램의 input 과 output 의 흐름.

- **kernel**

HP-UX operating system 의 중심(core). 이것은 memory 를 할당하고 수행 프로그램을 스케줄링하는 것과 같은 function 들의 수행과 그 컴퓨터 자원들의 관리를 책임지는 code 이다. Kernel 은 또한 전반적인 시스템 성능을 위해 요구되어 지는 관리적인 function 들도 수행한다.

- **kernel configuration**

시스템의 kernel 을 생성하기 위해서 사용된 unique 한 parameter 들. 예를 들어, **nproc** 은 kernel configuration parameter 이다.

- **kernel table**

Process Table 이나 Text Table 과 같은 internal 시스템 table 들. Table 의 configured size 는 시스템 행위에 영향을 줄 수 있는데, 예를 들어, 만약 **nproc** value 가 너무 낮으면 프로세스들은 그 한계치에 도달 했을 때 재빨리 수행을 멈출 것이다.

- **load average**

CPU 스케줄링을 기다리면서 "run" state 에 있는 프로세스들의 평균 수. 또한 run queue length 라고도 불린다.

- **local node**

여러분이 Glance 를 구동 시키고 있는 그 컴퓨터.

- **local swapping**

Local node 에 부착된 swap device 에서 발견된 space 를 사용하는 swapping.

- **logical read/write**

physical disk access 로 결론지어 질 수 도 있고 아닐 수도 있는 read 나 write 시스템 call.

- **major number**

kernel 에 있는 device driver table 로의 index. 이것은 device driver 와 access method (block 이나 character) 에 바탕을 둔다. 이것은 주변장치(peripheral)로 communicating 하기 위해서 필요 되어진다. Major device code 로 불리기도 한다.

- **man-Page**

HP-UX Reference online manual 로부터 이용 가능한 정보.

- **measurement interface (MI)**

See HP-UX Measurement Interface

- **message buffer pool**

시스템에서 모든 사용된 message-queue buffer 들을 저장하기 위해서 사용된 cache.

- **message-queue buffer**

message 를 보내기 위해서 사용된 user-defined buffer (*see* man-page *msgop(2)*)

- **midaemon**

HP GlancePlus/UX 와 함께 install 되는, 시스템 성능을 모니터하고 GlancePlus 와 HP LaserRX/UX 에 의해서 read 되고 display 되는 counter 들을 지원하는 프로세스 (daemon). (see man-page *midaemon(1)*)

- **minor number**

select code 와 peripheral device 에 특정화된 다른 정보를 구성하는 16 진수 의 수. Minor number code 라고도 함.

- **mounting/unmounting**

root file system 으로나(to) 로부터(from) 추가적이고 기능적으로-독립적인 file system 들을 추가하거나 제거하는 프로세스.

- **Network File System (NFS) transfer**

NFS 서비스를 지원하기 위해서 LAN 을 통한 data 의 transfer.

- **NFS-mounted**

하나의 시스템에 software 에 의해 연결된 (connected), 그러나 physical 하게는 다른 시스템의 disk 에 있다, file system 을 위한 용어.

- **nice**

time-shared 프로세스의 우선 순위를 바꾸기 위해서 사용된 value. 양수의 nice value 는 그 우선 순위를 낮추고; 음수의 value 들은 그 우선순위를 증가 시킨다. (see man-page *nice(1)*)

- **node**

network 에 있는 컴퓨터.

- **node name**

그 시스템의 host name (see man-page *hostname(1)*)

- **non-file system I/O**

disk device 로의 raw- 혹은 character-mode input 과 output.

- **outbound read/write**

local process 가 remote 시스템 으로부터(from) read 나 으로(to) write 를 request 할때 사용되는 말 (designation). See also **inbound read/write**

- **packets in/out**

LAN 을 통해 서버와 client 들 사이에 data 를 transfer 하는 diskless network 의 방법. Client 들에 의해 server 로 보내지는 request 들은 "in" packets 이고; server 에 의해서 client 들로 보내지는 request 들은 "out" packets 이다.

- **page**

virtual memory 의 기본 unit. 프로세스는 swap space 로(to)와 로부터(from) swapped 된 virtual memory 의 page 들에 저장된다. Series 300 에서는 page 의 size 가 4 KB 이고, series 800 에서는 2 KB 이다.

- **page fault**

프로세스가 physical memory 에 있지 않은 code instruction 을 수행하려 하거나 data page 를 참조하려 할 때 기록된 event. Virtual memory 는 반드시 수행이 지속되게 하기 위하여 missing code 나 data 를 page-in 해야 한다.

- **paged-in/paged-out pages**

virtual memory (disk) 로부터 physical memory 로 (paged-in), 혹은 그 반대로 이동된 data 의 page 들.

- **Pagedaemon**

Virtual memory 시스템의 paging 역량을 지원하기 위해서 secondary storage (disk) 로의(to) 프로세스의 address space 의 부분을 writing 하는 것을 책임지는 시스템 daemon.

- **pagein routine**

프로세스의 address space 의 page 들을 physical memory 로 가져오는 kernel routine.

- **pageout routine**

scarce(falls below lotsfree)를 얻기 위해서 physical memory space 가 시작될 때, 그리고 swap space 나 file system 으로 writing 함으로써 pagedaemon 이 memory 로부터 최소로 필요 되어지는 page 들을 제거하기 위해서 activate 될 때 수행하는 kernel routine.

- **parent process**

다른 프로세스를 생성해 내는 프로세스. See also **child process**

- **physical memory**

컴퓨터 시스템내에 포함된 실질적인 hardware memory 구성요소들.

- **physical read/write**

data 가 memory 로부터 disk 로, 혹은 그 반대로 전송되는 input/output operation.

- **PID**

A process identifier – 프로세스의 unique 한 identification number 는 그것을 다른 프로세스들로부터 구별 시킨다.

- **pipe**

관계된 프로세스들 사이에 한방향 (unidirectional) 의 data 흐름을 지원하는 interprocess communication capability 로, 한 프로세스로부터의 output 이 다른 것으로의 input 으로서 서비스하도록 한다.

- **PPID**

A parent process identifier – 다른 프로세스를 fork 시키거나 vfork 시키는 프로세스의 process identifier.

- **priority**

CPU scheduler 의 중요성을 결정하는 PID 에 할당된 수 (우선 순위)

- **process**

operating system 이 한 unit 의 work 으로 관리하는 running program.

- **processor states (activities)**

CPU 에 의해 수행되는 다른 작업들 (tasks). 예를 들면: User, Nice, System, 그리고 Interrupt 등.

- **process identifier**

See PID and PPID

- **pty (pseudo-teletype driver)**

rlogin 이나 telnet 과 같은 command 에 통해 logged in 된 "software terminal".

- **queue**

Resource 가 available 할 때까지 unsatisfied 한 request 가 쌓이는 waiting line.

- **raw mode**

Device 와 data 를 요구하는 user 프로그램 사이에 직접적으로 data 를 transfer 하는 unbuffered 된 input/output. 이것은 file system 의 buffer cache 를 우회한다.(bypass). Character mode 라고도 한다. **Block mode** 와 비교.

- **real-time CPU dispatching priority**

Time-share process 를 위한 일반적인 priority 보다 더 높은 CPU dispatching priority. (see man-page *rtprio(1)*)

- **record lock**

file 에 있는 특정 record 에 동시 access 를 방지하는 방법. (see man-page *lockf(2)*)

- **refresh screen**

현존의 통계치를 update 하지 않고 Glance 의 current screen 을 다시 display 해 줌.

- **remote node**

HP GlancePlus/UX 를 구동시키고 있는 시스템이 아닌 network 상의 시스템.

- **remote swapping**

다른 시스템의 swap device 상에 위치한 pool 로부터 swap space 를 사용하는 swapping. 이러한 유형의 swapping 은 server 에 swap on 되어 있는 diskless 시스템에서 종종 사용된다.

- **resident buffer**

physical memory 에 저장된 data.

- **resident set size.**

프로세스가 사용하고 있는 physical memory 의 양. 이것은 그 프로세스의 data, stack, 그리고 text segment 를 위해 할당된 memory 를 포함한다.

- **root file system**

설치된 file system hierarchy - 다른 volume 들이 mount 될 수 있는 file system 의 portion - 에 HP-UX kernel (code)를 포함하는 file system.

- **run queue length**

See load average

- **runnable (executable) process**

operational 한 프로세스

- **semaphore**

프로그램 code 의 주어진 section 에 몇 가지 프로세스들의 access 를 조절하는 special file. (see man-page *semaphore(2)*)

- **server machine**

See diskless cluster server

- **server swap space**

diskless cluster server 에 위치한 swap space

- **shared library**

한 프로그램에 linked 된 수행가능한 code library 이며 아마도 다른 프로그램들에 의해 shared 된다.

- **shared-memory segment**

몇 가지 프로세스들을 위한 sharing data 에 dedicated 된 시스템 memory 의 portion.

- **shared memory pool**

Shared-memory segment 가 저장되어 있는 cache.

- **shared text segment**

몇몇의 프로세스들 사이에 shared 된 code.

- **shell**

User 와 operating system 사이를 연결하는(interface) 프로그램. 그것은 user 와 operating system (kernel) 사이에 command 를 해석하고 전송하는 command-line interpreter 를 포함한다.

- **sleeping process**

그 스스로 block 했거나 혹은 blocked 된 프로세스로, waiting state 로 놓여진다.

- **socket operation**

communication 을 위해 endpoint 를 생성하고 모든 subsequent socket-related 시스템 call 내에서 사용하는 descriptor 를 return 시키는 프로세스.

- **Softkeys**

특정 functional activity 에 dedicated 된 function key 로, 종종 terminal display 의 바닥에 따르는 lable 들로 대체된다.

- **standalone**

HP-UX diskless cluster 의 부분이 아닌 machine.

- **super-user**

특별한 특권들을 가지는 root user – System Administrator.

- **swap area**

swapping-in 과 swapping –out 프로세스들을 위해 reserved 된 disk 의 영역

- **swap in/out**

disk 의 swap 영역으로부터 주 memory 로 혹은 그 반대로 프로세스를 copying 하는 procedure.

- **swap spaces**

Virtual memory 에 의한 사용을 위해 reserved 된 disk drive 들과 같은 secondary storage media 에 있는 지속적인 영역.

- **swap spaces, device**

file system 과 shared 되지 않은 특정 device 에 위치하고 있는 고정된 swap spaces. (see man-page *swapon(1)*)

- **swap spaces, file system**

shared 된 file system space 로부터 취해진 swap spaces (see *HP-UX System Administration Concepts Manual*, "Swap Space Management" or the *HP-UX System Administration Task Manual*, "Managing Swap Space.")

- **swapon command**

interleaved 된 paging 과 swapping 을 위한 swap space 를 초가하기 위하여 사용된 command. (see man-page *swapon (1M)*)

- **swapping**

disk 상의 main memory 와 dedicated(reserved) 영역 사이에서 전체 프로세스들을 이동시키기 위해서 사용된 memory management technique. Swapping 은 시스템이 주어진 시간에 main memory 에 맞출 수 있는 것보다도 더 많을 프로세스들을 구동시키게 해 준다. Physical memory 가 매우 부족하게 될 때 (falls below desfree), 프로세스는 deactivated 되고 그것의 page 들은 swap device 로 moved (swapped) 된다.

- **switching context**

See context switching

- **system buffer cache**

See **buffer cache**

- **system buffer pool**

Data buffer 를 hold 하기 위해서 제외된 memory 영역.

- **system calls**

그 프로그램을 대신해서 특정 service 를 수행하도록 kernel 에게 알리기 위해 프로그램이 사용하는 command 들.

- **system code**

system call 들을 통해 수행된 kernel code.

- **system interrupt handling code**

interrupt 들을 프로세스하기 위해 사용된 kernel code.

- **system kernel code**

See system code

- **system startup**

"Booting the system." Powered-down 되고 inactive 한 상태에서 HP-UX 가 구동하고 있고 input 을 받아들일 준비가 되어 있는 interactive 하고 functional 한 상태로 컴퓨터를 취하는(take) 프로세스.

- **terminal transfer (tty or pty)**

Terminal device 로 부터의 read 혹은 terminal device 로의 write.

- **text segment**

수행 가능한 프로그램 code 를 간직하는 memory segment.

- **thrashing**

시스템이 data 를 swapping in 하고 out 하는데 너무 많은 시간을 소비하고, 유용한 작업을 하는데는 거의 시간을 투자하지 않은 상태. 이것은 너무 많은 page fault 가 생성되었거나 너무 많은 swapping 이 발생한 상태의 특징이다. Thrashing 은 시스템 성능 저하를 불러오고 interactive 한 user 들에 대한 response time 을 상당히 증가 시킨다.

- **threshold**

Glance 에서, Global 스크린에 나타나는 interesting 프로세스들의 수를 조절하기 위해서 사용될 수 있는 세가지 선택적인 delimiters (CPU, Disk, or Swap rate)들 중 하나.

- **tty**

각 terminal device 에 연관된 special file. 이것은 user 의 프로그램과 terminal 사이의 data 의 교환을 관리하는데 사용된다.

- **tty path-name filter**

Glance 에서, 단지 그 terminal 에 attached 된 process 들에 프로세스 display 를 제한하기 위해서 terminal device 을 특성화 한 것.

- **unmounting/mounting**

Root file system 으로나(to), 으로부터(from) functional 하게 독립적인 file system 들을 제거하거나 추가하는 프로세스. (see man-page *mount(1M)*)

- **update interval**

GlancePlus 가 성능 통계치를 update 하기 전에 스크린을 display 하는 (초단위로 측정된) 기간 (period). Default interval 은 5 초 이고, 그것은 i command 를 이용하여 변화시킬 수 있다.

- **user code**

시스템 kernel 과 연관되지 않은 code.

- **user name**

프로그램을 시작하는 user 의 login name.

- **user path-name filter**

Glance 에서, 그 user 에 attached 된 프로세스들에만 프로세스 display 를 제한하기 위해서 user name 을 특성화 한 것.

- **vfork**

Parent 프로세스와, code 와 data 를 공유할 수 있는 child 프로세스를 낳은 fork system call 의 version. (see man-page *vfork(2)*)

- **virtual memory**

Disk 나 다른 storage device 의 portion 에 존재하는 secondary memory. 이것은 primary physical memory 의 확장으로 사용된다.