Network Benchmarking Methodologies and Tools

Ying-Dar Lin

Outline

- External benchmarks
  - Freeware: WebBench, Email test tool
  - Tester: Smartbits, Avalanche
- Internal benchmarks
  - gprof, kprof
- Mini-project
  - Squid-Apache
- Term-Project
  - Postfix, AMaViS, ClamAV

External benchmark

Freeware: WebBench/SPECmail

Tester: Smartbits/Avalanche

Tools for external benchmark

Application	Tool	Platform	Cost	Feature
TCP/UDP	Ttcp	Unix/Windows	Free	command-line
Web	WebBench	Windows	Free	GUI-based
Web	httperf	Unix	Free	command-line
Mail	SPECmail	Java	$900	cross-platform
Mail	Postal	Unix	Free	command-line
Mail	Email test tool	Windows	Free	GUI-based
Generic	Smartbits and its applications	Applications on Windows	Expensive Over $1000	Hardware traffic generator
FTP, Mail, Web…	Avalanche, Reflector	Applications on Windows	Expensive Over $1000	Hardware traffic generator

What is WebBench?

- Purpose: to measure the performance of Web servers
- Emulate a huge number of clients to generate HTTP requests
- One controller + a number of clients on Windows OS
- Support HTTP/1.1 persistent connections and pipelining
- Provide static and dynamic test suites
- Tunable workload
- Output in Excel formats

Install WebBench

1. WebBench available at *http://speed.cis.nctu.edu.tw/~ydlin/course/cn/exp/index.html* for controller, client and workload
2. Install controller and clients
3. Modify the HOSTS file (under system installation directory) on each client

   xxx.xxx.xxx.xxx controller

   yyy.yyy.yyy.yyy server
4. Add an entry for each client's IP address and ID number in *C:\WEBBENCH\CLIENTIDS\CLIENT.CDB* on controller

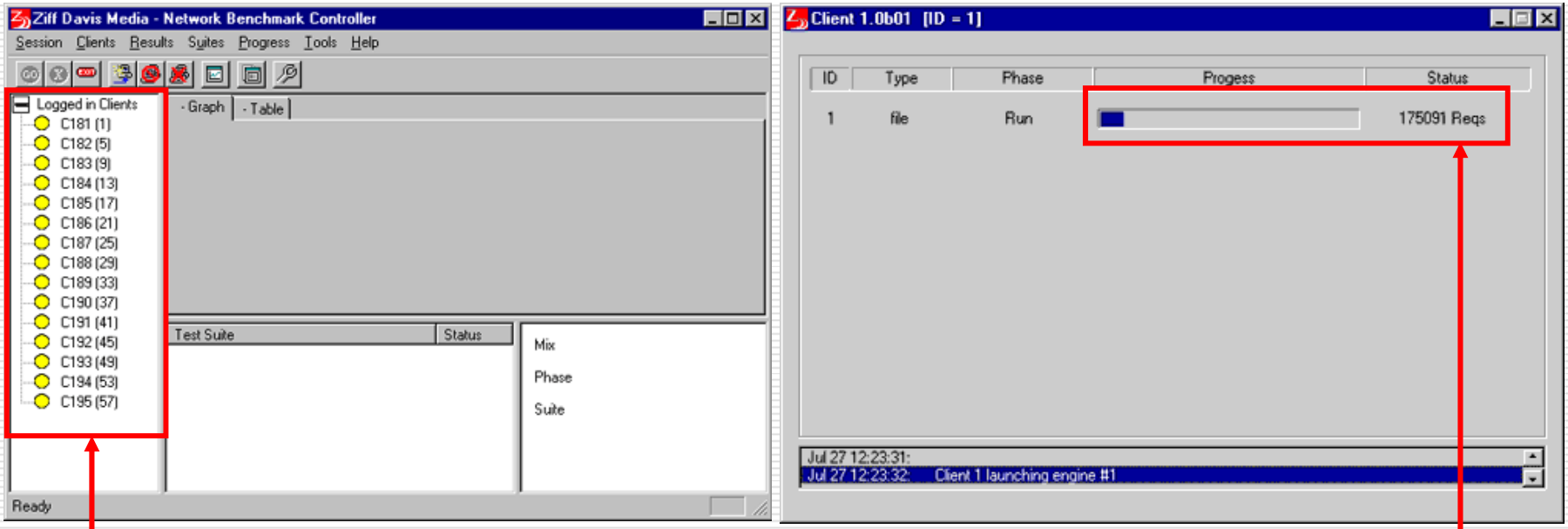
   e.g. 192.168.1.1 1

   192.168.1.2 2
5. Install workload on server

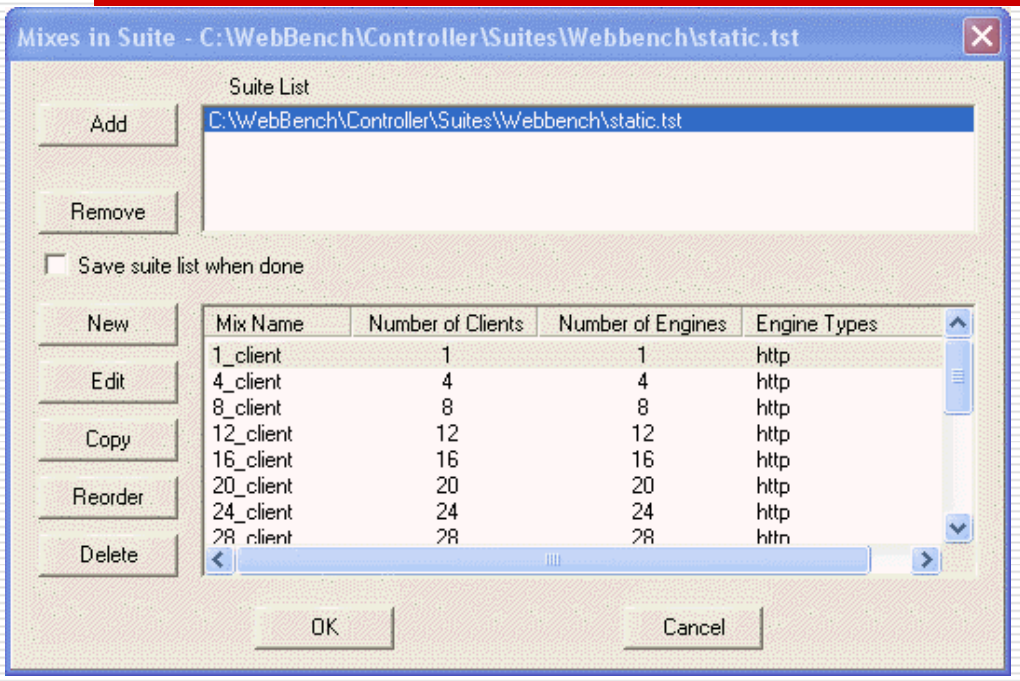
Run WebBench

1. Run controller
2. Run clients on each host. Make sure connection with controller established
3. Choose the test suite and run the test
4. Read the results after the test finishes

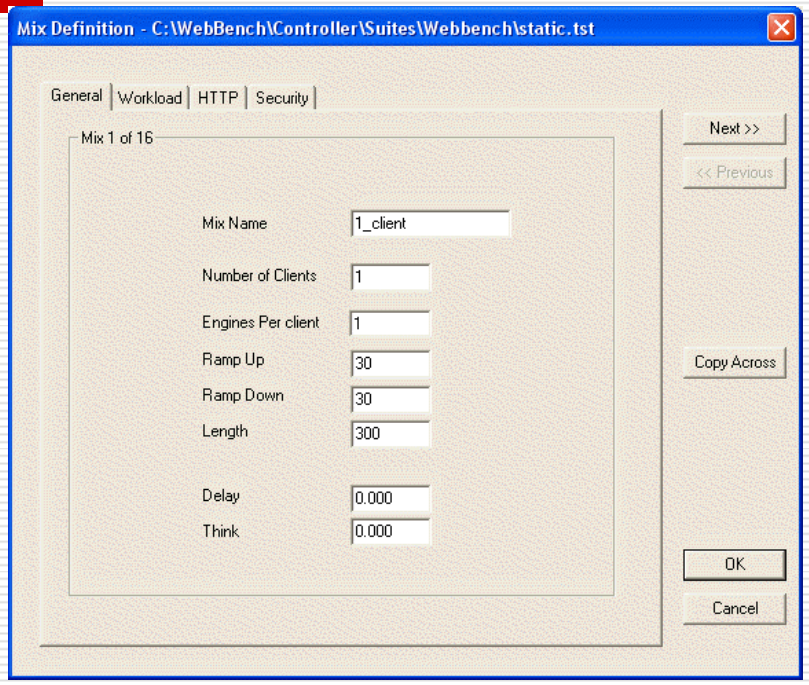
Some GUIs (controller & client)



GUIs (editing test suite)



Mixes in test suite



Editing mixes

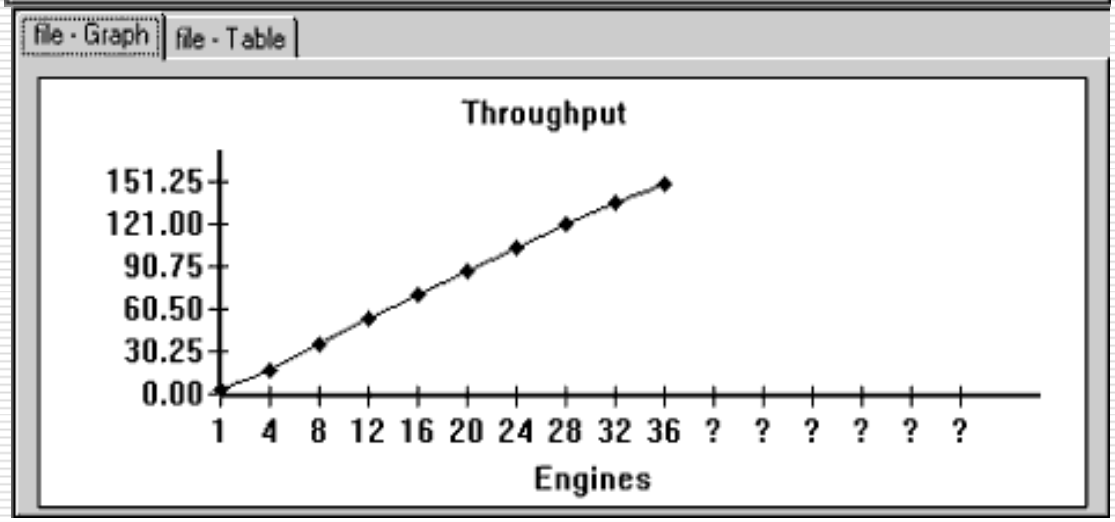
Mix: a group of requests and the parameters for the requests.
Engine: the process that executes the test workload on the client.
Test suite: collection of one or more test mixes that WebBench executes in sequence

GUIs (benchmark results)

file - Graph | file - Table

Engines	Throughput	Errors
1	4.710647447	0
4	18.62259924	0
8	36.96891266	0
12	54.86984267	0
16	72.08924348	0
20	89.06379648	0
24	105.6280931	0
28	122.1050999	0
32	137.4701442	0
36	151.986784	0

Table of results



Graph of results

GUIs (final report)

Table 1: WebBench Summary

C:\WebBench\Controller\Suites\Webbench\static_mt.tst

Mix Name	Requests Per Second	Throughput (Bytes/Sec)	Test Information
1_client	63.583	395836.875	Engine Types: http
4_client	79.388	477667.563	WebBench 5.0
8_client	79.154	498054.000	Start Suite: Sat Aug 23 19:56:04 2003
12_client	81.162	506301.813	Finish Suite: Sat Aug 23 21:19:41 2003
16_client	81.200	501224.156	Elapsed Time: 01:23:37
20_client	79.208	485184.156	Status: Suite completed successfully
24_client	80.646	486480.375	Comments:
28_client	82.100	480891.719	
32_client	65.258	388099.969	
36_client	72.304	435581.219	
40_client	73.471	442779.063	
44_client	76.754	468235.031	
48_client	75.313	448606.500	
52_client	75.658	461913.156	
56_client	80.108	500430.969	
60_client	77.617	461723.406	

What is Email test tool?

- Purpose: to measure the performance of mail servers
- Emulate a huge number of clients to generate SMTP, POP3 and IMAP4 traffic
- One controller + a number of clients on Windows OS
- WebBench-like testing framework
- Results measured in *messages per second*
- More information and tools available at *http://speed.cis.nctu.edu.tw/~ydlin/course/cn/exp/index.html*

Install Email test tool

1. Install controller and clients
2. Modify the HOSTS file (under system installation directory) on each client

   xxx.xxx.xxx.xxx controller

   yyy.yyy.yyy.yyy server
3. Add an entry for each client's IP address and ID number in *C:\WEBBENCH\CLIENTIDS\CLIENT.CDB* on controller

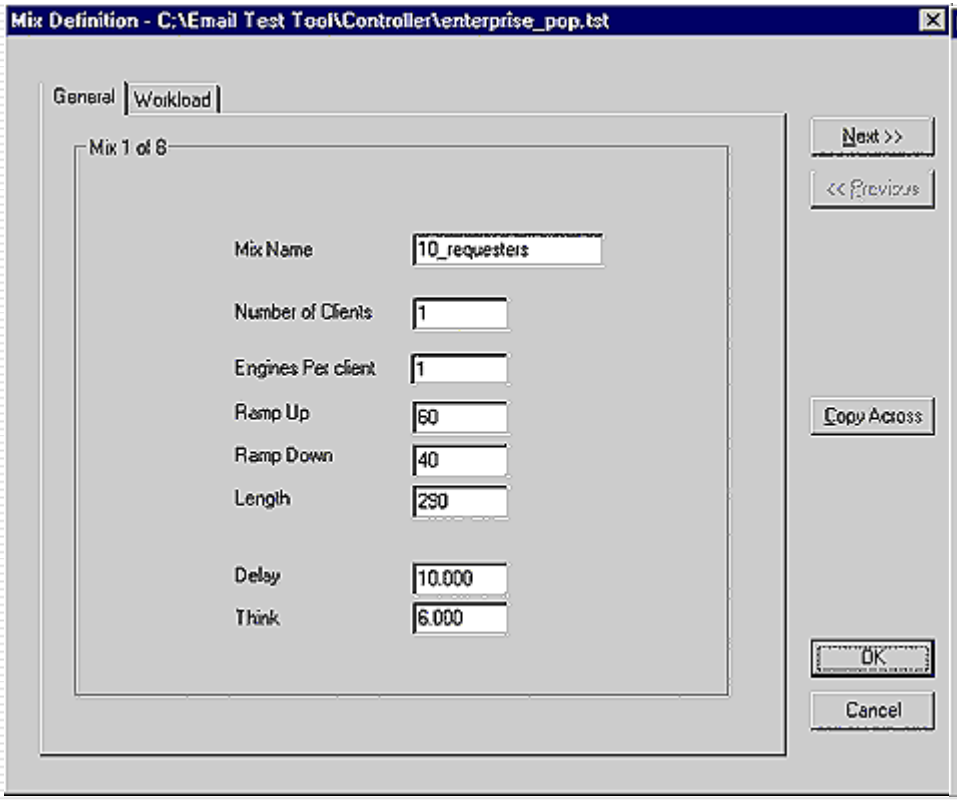
   e.g. 192.168.1.1 1

   192.168.1.2 2

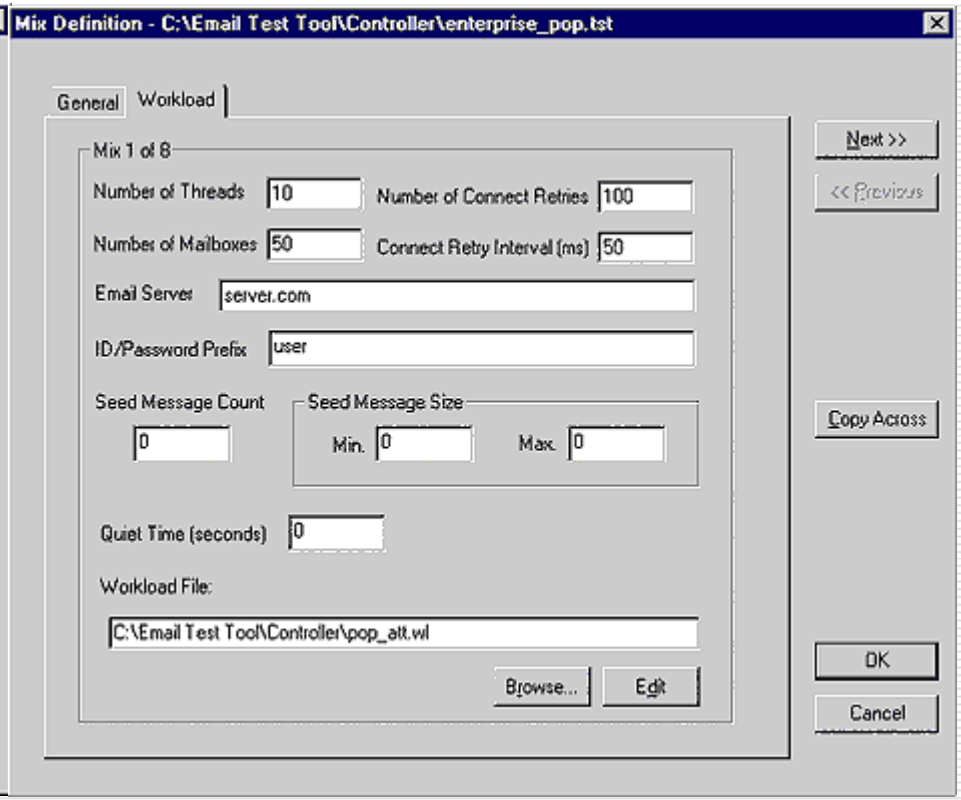
Edit the workload file

```
DEFINE_PROFILE
        POP_TRAN: 70
        SMTP_1K_TRAN: 12
        SMTP_4K_TRAN: 6
        SMTP_8K_TRAN: 9
DEFINE_TRANSACTIONS
 POP_TRAN:
        # Login to a mailbox using POP3, download all the mail,
        # and then delete the mail from the server.
       LOGIN POP3
        GET ALL
        DELETE ALL
 SMTP_1K_TRAN:
        # Send a 1K message to 3 mailboxes.
 SEND 3 1024
 SMTP_4K_TRAN:
        # Send a 4K message to 3 mailboxes.
 SEND 3 4096
 SMTP_8K_TRAN:
        # Send a 128-byte message with a 8K attachment to 3 mailboxes.
 SEND 3 128 ATT=a8k.txt
```

GUIs (editing mix definition)



Mix definition



Workload

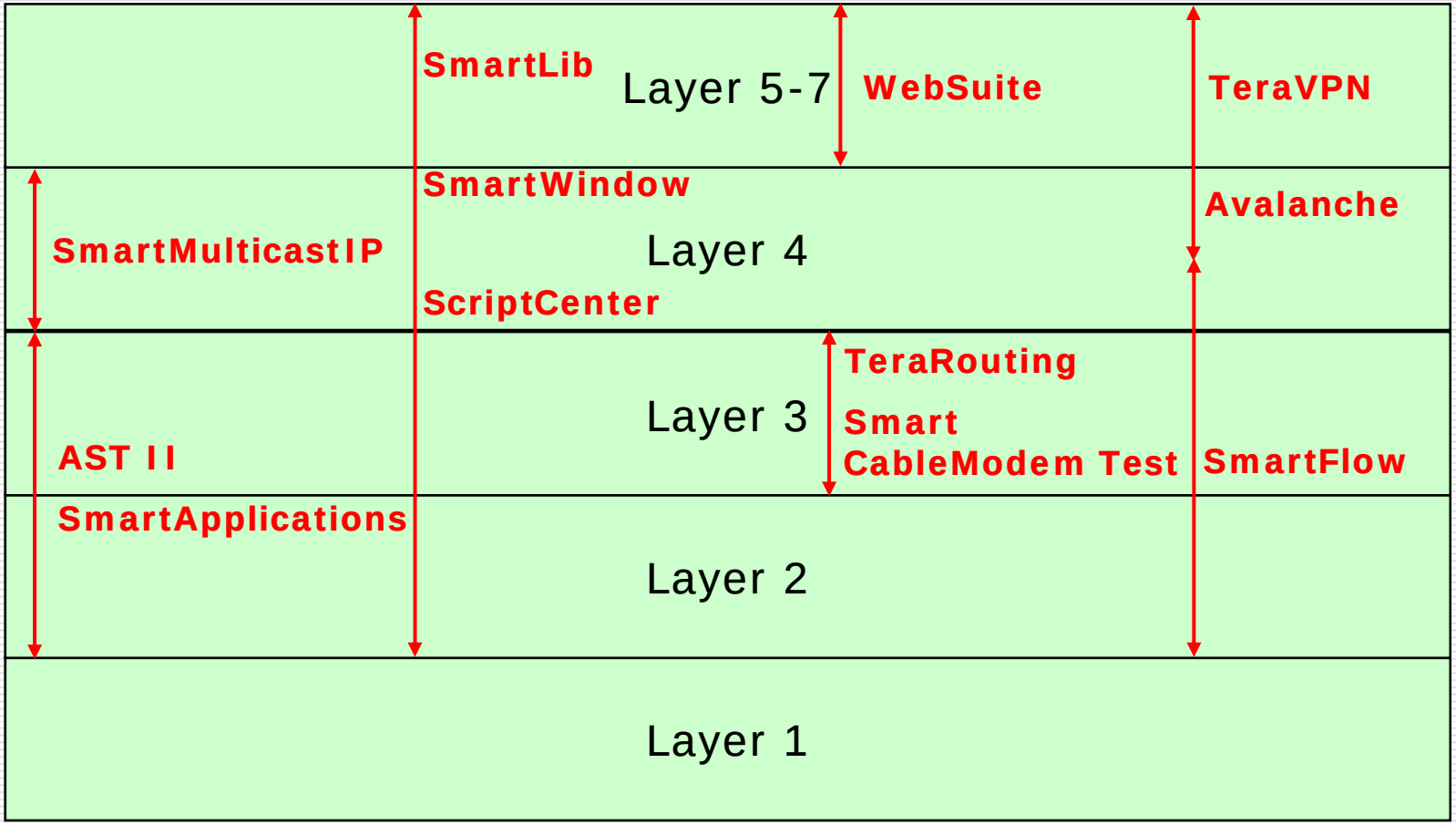
GUIs (final report)

Mix Name	Engines Participating	Server Response Time (seconds)	Total Valid Messages Sent	Total Valid Messages Received
10 requesters	1	0.048	270	195
100 requesters	10	0.144	2550	1901
140 requesters	14	0.139	3501	2569
180 requesters	18	0.133	5484	4317
220 requesters	22	0.169	9132	6997
260 requesters	26	0.158	18459	9526
300 requesters	30	0.211	25587	9403
400 requesters	40	1.164	25707	7935
500 requesters	50	2.933	33072	9551
600 requesters	60	4.681	30618	6781

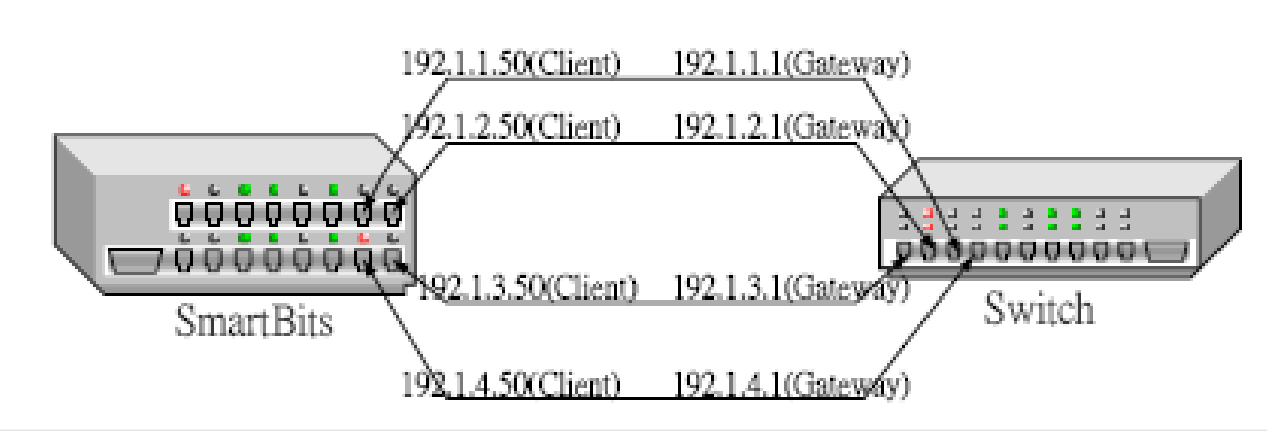
What is Smartbits?

- Smartbits is a chassis
  - SMB 200/2000, SMB 600B/6000C
  - Accommodate multiple slots for testing modules
- Smartbits generates traffic at wire-speed for testing
- Support testing on various technologies
  - Ethernet, ATM, Fiber channel, Frame relay, xDSL, cable modem…
- Feature various test applications
  - IP QoS, VoIP, MPLS, Multicast, TCP/IP, IPv6, routing, SAN, VPN…

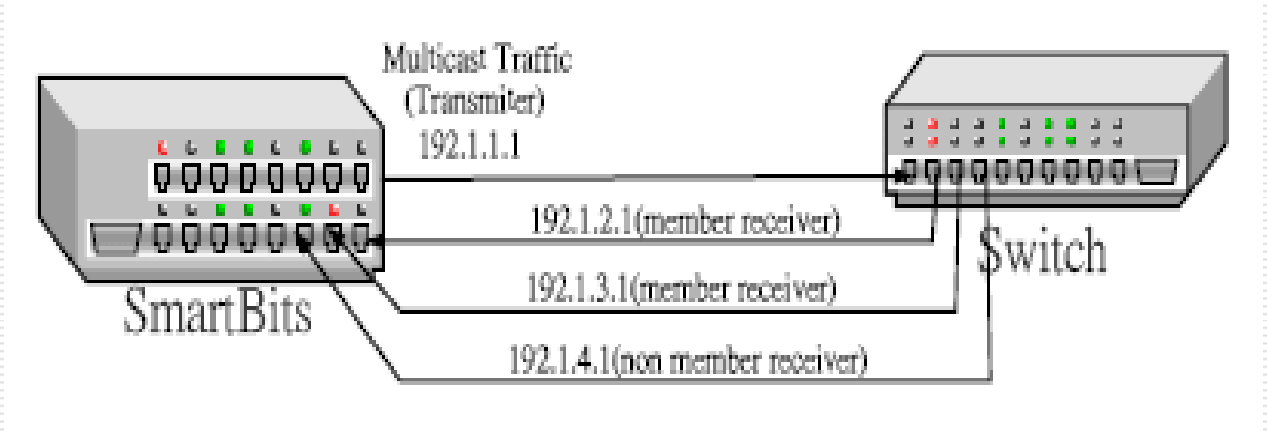
Smartbits applications



Smartbits test environment

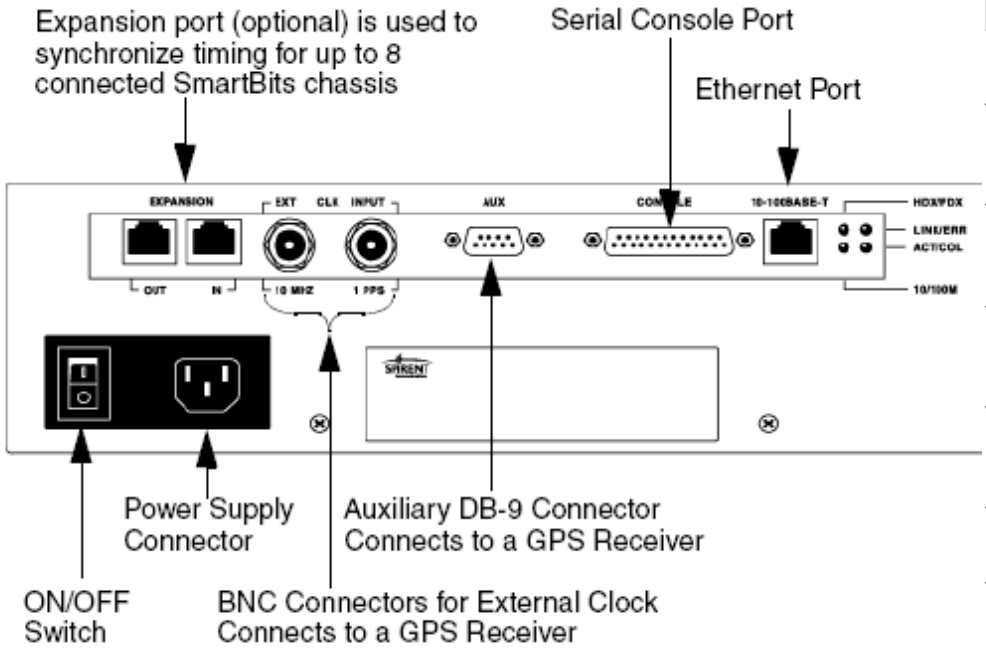


SmartFlow



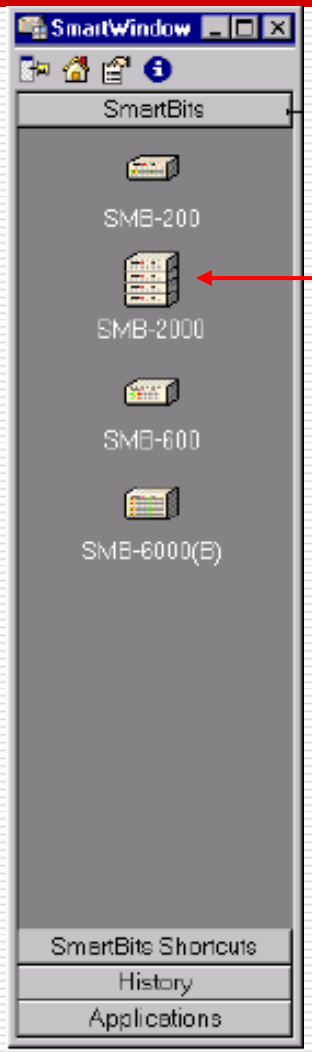
SmartMulticast

Smartbits installation -- connection



Connect the SmartBits chassis…	to …	using this cable…
Power supply connector located on the back panel	Power outlet	Power cord
CONSOLE port (DB-25) – for IP address assignment ONLY	PC Serial port (RS-232)	DB-25-to-DB-25 cable (plus a DB-25-to-DB-9 adapter if needed)
10/100BASE-T (RJ-45) port – to connect a SmartBits chassis to applications and to the network	Your hub or LAN	Blue 10 ft. (3.048 m.) straight-through LAN cable
10/100BASE-T (RJ-45) port – To connect a SmartBits chassis directly to a PC using the Default IP Address	PC	White 10 ft. (3.048 m.) Ethernet crossover cable
EXPANSION OUT or IN RJ-45 port (optional)	Another SmartBits chassis and its RJ-45 EXPANSION IN port	Purple 3 ft. (.9144 m.) straight-through LAN cable
EXT CLK INPUT port	An external timing device	RG58 coaxial cable (BNC)
AUX	A GPS receiver	DB-25 to DB-25 male-to-male with DB-9 adapter

Smartbits installation -- SmartWindow



Show chassis icons,
click to connect for configuration
default IP address: 192.168.0.100

Check connection if it fails

Smartbits installation – Set up IP address via HyperTerminal

1. Connect chassis with console port
2. Launch HyperTerminal
3. Set the following parameters
   - Bits per second: 38400
   - Data bits: 8
   - Parity: none
   - Stop bits: 1
4. After connection, type

   ipaddr xxx.xxx.xxx.xxx to change the IP address
5. ping the new address to confirm connection

What is Avalanche/Reflector?

- Avalanche
  - Generate large quantities of realistic user traffic for stress test
- Reflector
  - simulate the behavior of Web sites, application environments, and database server clusters.

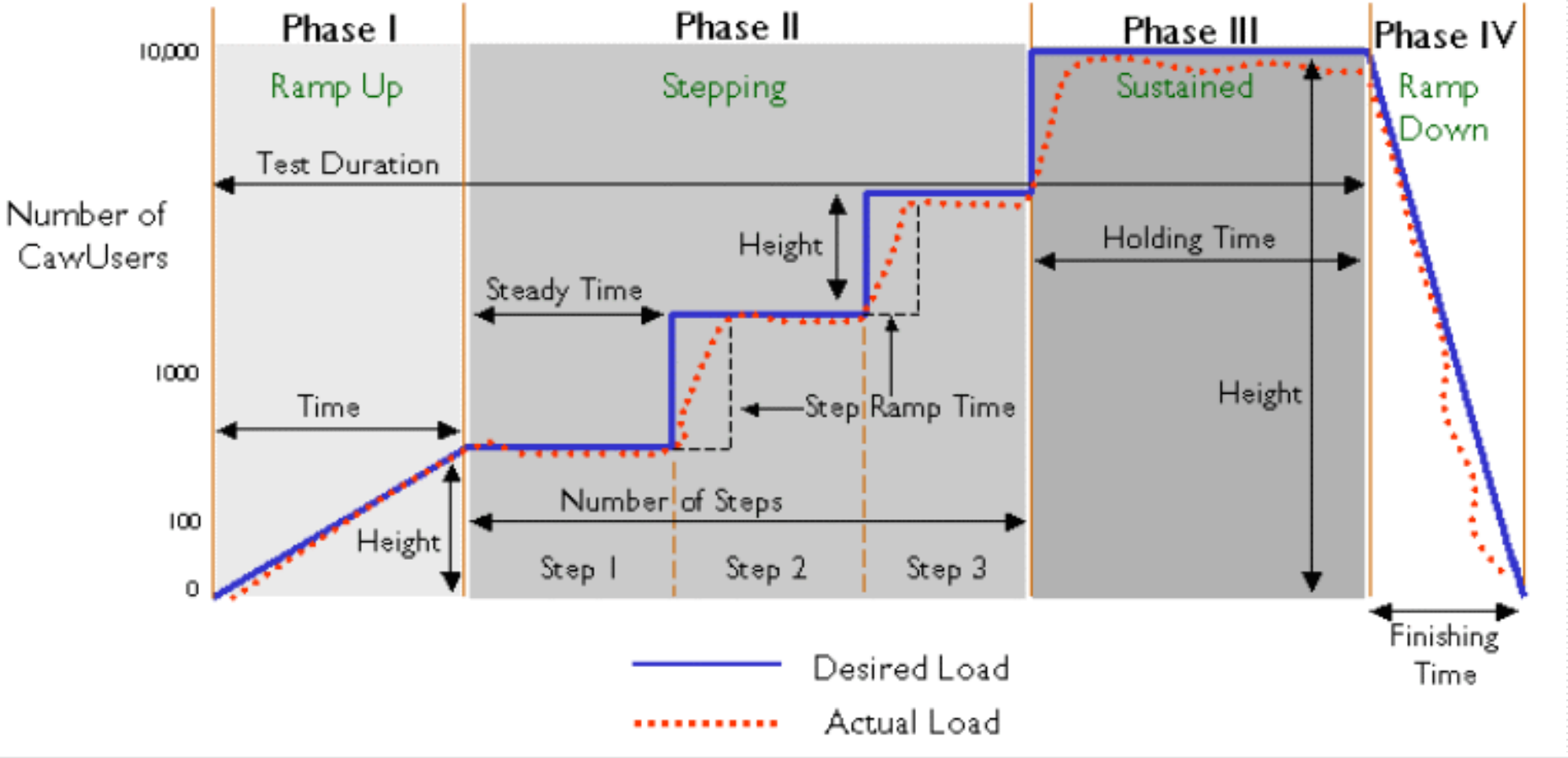
Features of Avalanche

- Simulate over two million concurrently-connected users with unique IP addresses
- Interoperate with Reflector to facilitate equipment evaluation
- Java-based GUI for configuration
- Can generate traffic of the following protocols
  - HTTP, HTTPS, RTSP, FTP, POP3, SMTP

Features of Reflector

- manage up to 2 million open connections and handles more than 45,000 HTTP requests per second (60,000 HTTP 1.1 requests with persistence) 30,000+ concurrent streaming media requests
- Java-based GUI for configuration
- Can emulate the following servers
  - HTTP, SSL, RTSP, FTP

Test phase (1)


Phase I
Ramp Up
Phase II
Stepping
Phase III
Sustained
Phase IV
Ramp Down
10,000
1000
100
0
Number of CawUsers
Test Duration
Height
Steady Time
Holding Time
Time
Step Ramp Time
Height
Number of Steps
Height
Step 1
Step 2
Step 3
Finishing Time
Desired Load
Actual Load

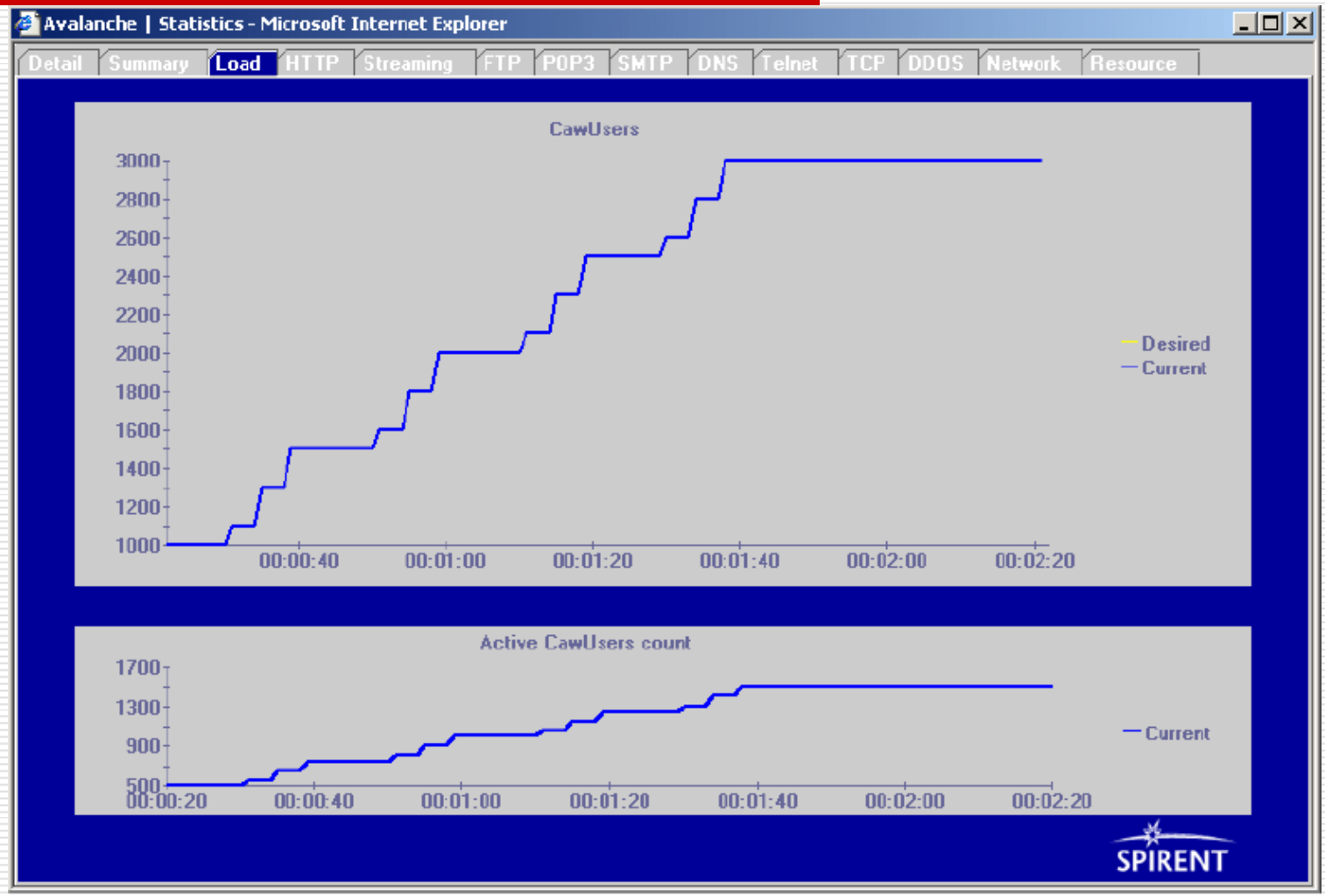
Test phase (2)

- Ramp up
  - Slowly increase the load to a given threshold to avoid a DoS alarm
- Stepping
  - Continue increase the load through a series of steps
- Sustained
  - Keep a constant load for a duration of time
- Ramp down
  - Slowly close the testing

User profiles

- A specification to emulate a user population's behavior
- Two sections
  - User behavior: control a user's action, such as *think time*, *abort time*, etc.
  - Browser emulation: control the protocols, headers, SSL configuration and user authorization

Sample results



Avalanche DDoS module

- Simulate the occurrence of DDoS attacks
- Support the following attacks
  - ARP flood, Ping sweep, TCP reset flood, Smurf, TCP SYN flood, TCP port scan, UDP data flood, UDP port scan, Xmas tree
- Support custom attack template by specifying
  - Packet rate, inter-packet timing
  - Header field values
  - Payload data
  - Non-conforming or experimental protocol
  - Corrupted or malicious data

Internal profiling

- Why internal profiling
  - To know where your program spends its time
  - To know which function calls other functions
  - To refine the segment of code that could be a bottleneck
- Steps in profiling using gprof
  - Compile and link the program with profiling enabled
  - Execute program to generate a profile data file
  - Run gprof to analyze the profile data

gprof commands

1. specify the `-pg' option when running the compiler and the linker
   cc -g -c myprog.c utils.c -pg
   cc -o myprog myprog.o utils.o -pg
   or cc -o myprog myprog.c utils.c -g -pg
2. If using ld for linking, add /lib/gcrt0.o and -lc_p
   ld -o myprog /lib/gcrt0.o myprog.o utils.o -lc_p
3. Run the executable normally, but it could be slower
4. The profile data gmon.out is left in the current directory **at the time it exits normally**
5. Run gprof to generate the profiling report
   gprof *options* [*executable-file* [*profile-data-files*...]] [> *outfile*], eg., gprof *executable*

Profiling result – flat profile

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
33.34	0.02	0.02	7208	0.00	0.00	open
16.67	0.03	0.01	244	0.04	0.12	offtime
16.67	0.04	0.01	8	1.25	1.25	memccpy
16.67	0.05	0.01	7	1.43	1.43	write
16.67	0.06	0.01				mcount
0.00	0.06	0.00	236	0.00	0.00	tzset
0.00	0.06	0.00	192	0.00	0.00	tolower
0.00	0.06	0.00	47	0.00	0.00	strlen
0.00	0.06	0.00	45	0.00	0.00	strchr
0.00	0.06	0.00	1	0.00	50.00	main
0.00	0.06	0.00	1	0.00	0.00	memcpy
0.00	0.06	0.00	1	0.00	10.11	print
0.00	0.06	0.00	1	0.00	0.00	profil
0.00	0.06	0.00	1	0.00	50.00	report

Profiling result – call graph

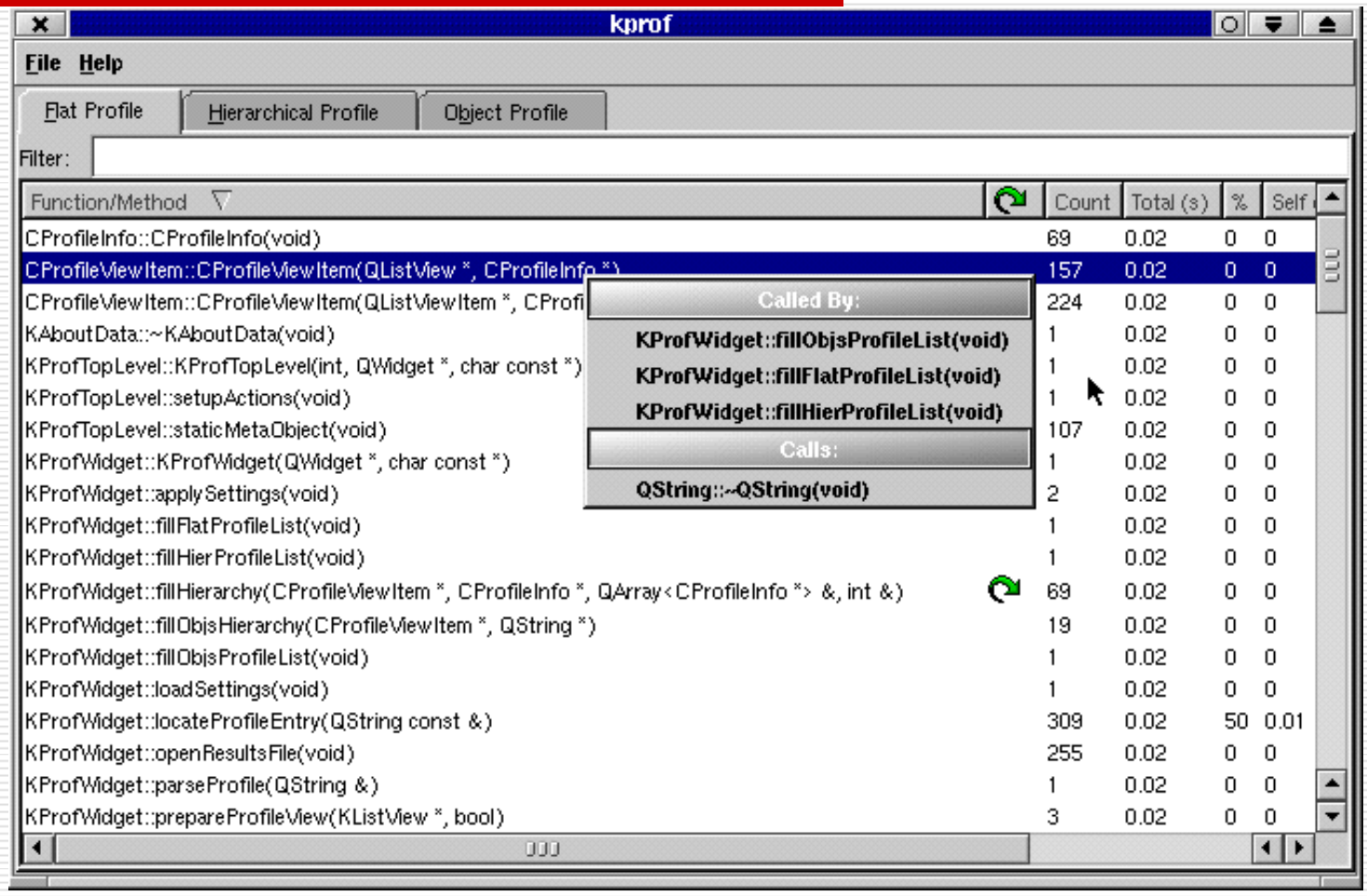
```
granularity: each sample hit covers 2 byte(s) for 20.00% of 0.05 seconds
index %  time    self    children    called     name
                                               <spontaneous>
[1]      100.0  0.00       0.05                start [1]
                0.00       0.05       1/1      main [2]
                0.00       0.00       1/2      on_exit [28]
                0.00       0.00       1/1      exit [59]
-----------------------------------------------
                0.00       0.05       1/1      start [1]
[2]      100.0  0.00       0.05       1        main [2]
                0.00       0.05       1/1      report [3]
-----------------------------------------------
                0.00       0.05       1/1      main [2]
[3]      100.0  0.00       0.05       1        report [3]
                0.00       0.03       8/8      timelocal [6]
                0.00       0.01       1/1      print [9]
                0.00       0.01       9/9      fgets [12]
                0.00       0.00      12/34     strncmp <cycle 1> [40]
                0.00       0.00       8/8      lookup [20]
                0.00       0.00       1/1      fopen [21]
                0.00       0.00       8/8      chewtime [24]
                0.00       0.00       8/16     skipspace [44]
-----------------------------------------------
[4]       59.8  0.01       0.02     8+472      <cycle 2 as a whole> [4]
                0.01       0.02     244+260    offtime <cycle 2> [7]
                0.00       0.00     236+1      tzset <cycle 2> [26]
-----------------------------------------------
```

What is kprof?

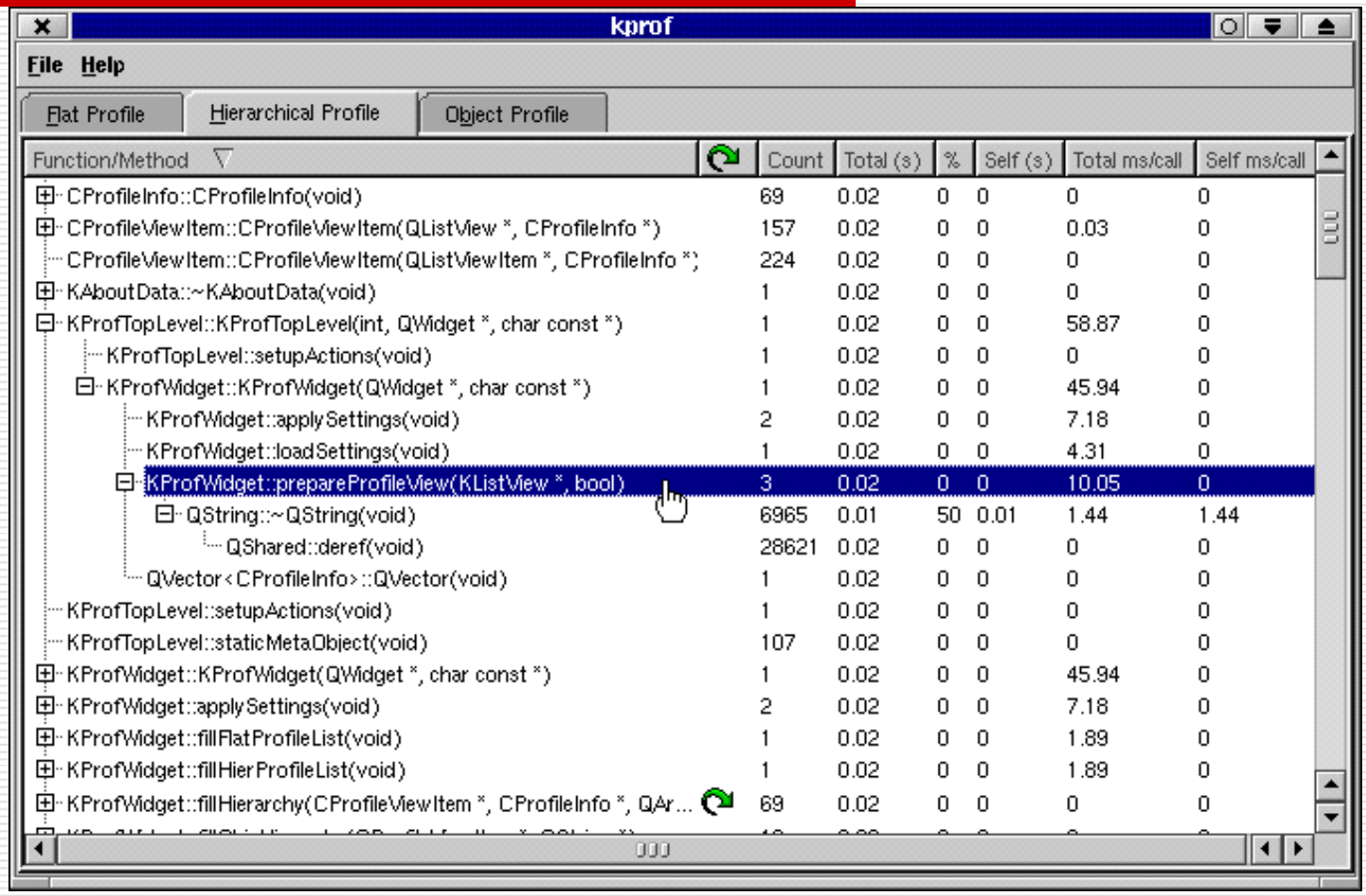
- Purpose: a visual tool displaying the execution profiling output generated by code profilers
- Provide list- or tree-views that make information very easy to understand.
- Usage
  - Generate report from gprof

    gprof -b sample3 >sample3.prof1
  - Start kprof and open the file sample3.prof1

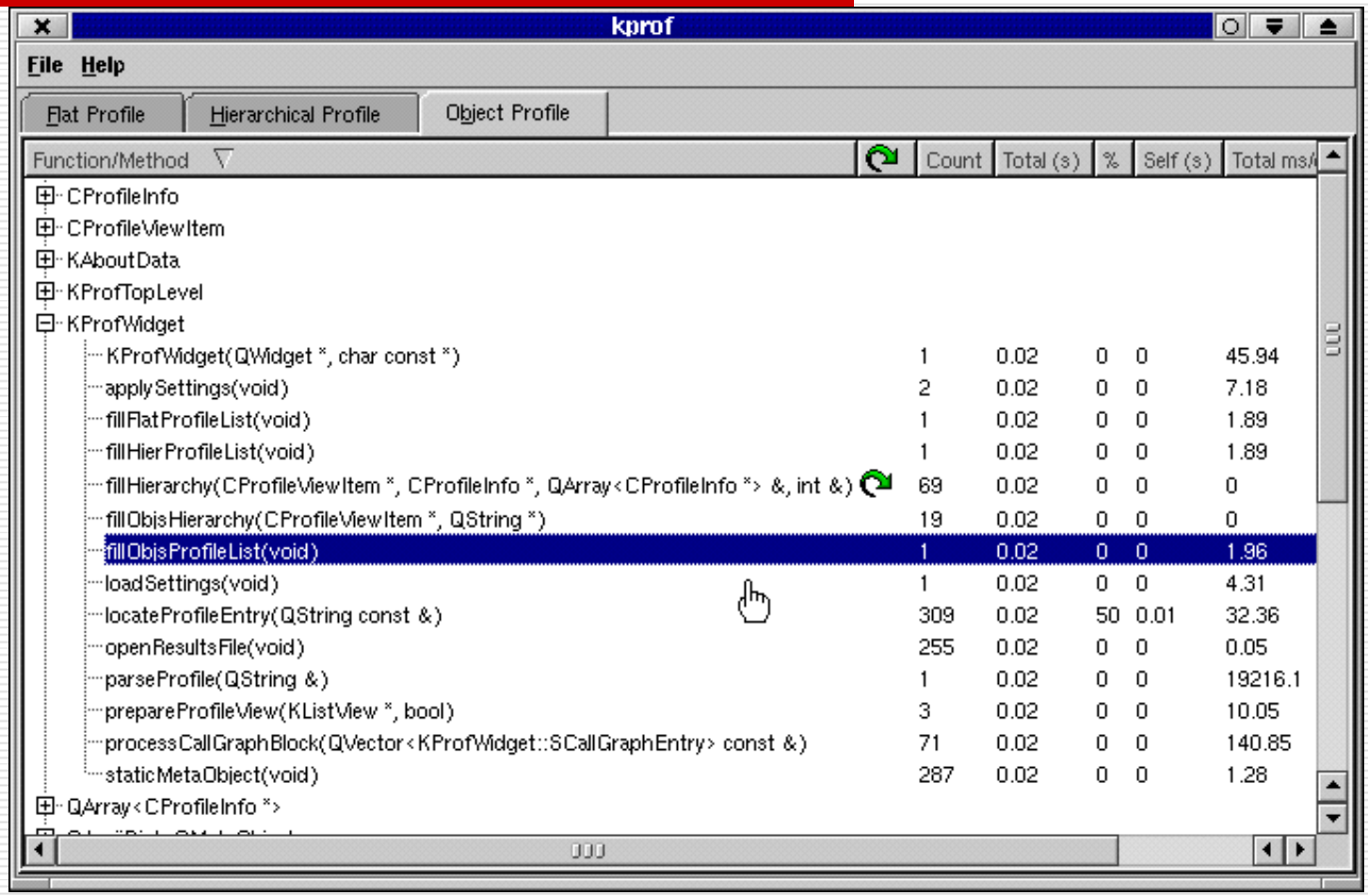
The flat profile view



The hierarchical profile view



The object profile view



Hands-on projects

Mini-project:
Benchmark the performance of a Web server using WebBench

Term-project:

Study the bottleneck of an anti-virus mail server

Mini-project: Performance benchmarking of a Web server

- Purpose
  - Be familiar with the usage of WebBench
  - Learn to install and configure a Web server
  - Study the performance of a Web server in various conditions

Mini-project: Pre-requisite of the mini-project

- A PC to serve as the Web server
  - Windows + IIS or Linux + Apache, etc.
- One or more powerful Windows PCs for WebBench
- Full set of WebBench packages (including workload)

Procedure of mini-project

1. Install a Web server to be tested.
2. Set up one controller and at least one client (may be on the same host).
3. Expand and install the standard workload in the document root of the server.
4. Start WebBench on controller and on each of clients.
5. Once all the clients are running WebBench, select the test suite STATIC.TST on the controller.
6. Choosing View Results on the controller after the test suite finished to look at the results.

Discussions of mini-project (1)

What is the maximum throughput (i.e., the server is saturated) in the following conditions.

- Tune the percentage of HTTP/1.0 traffic in the Mix definition window for 0%, 50%, 100%.
- Tune the number of engines per client for 10, 20, 30.
- Tune the receive buffer size for 1KB, 2KB, 4KB.
- Turn on pipelining option to see how pipelining accelerate the performance. (optional)
- If you have a proxy, you can test the performance connections through it. (optional)

Discussions of mini-project (2)

1. Show the screenshots of the results (in graphs) for each benchmark.
2. How does HTTP/1.1 increase the performance? Can you analyze it?
3. Is only one WebBench client enough to saturate your server? If not, how many are needed in your estimation.
4. How does the receive buffer size affect the performance?
5. Is server response time inversely proportional to the number of requests per second?

Discussions of mini-project (3)

6. How does pipelining affect the performance? Discuss it. (optional)
7. Is the proxy or the Web server the bottleneck in your benchmark? (optional)
8. How much time have you spent in this project?

Topic of term-project: Bottleneck analysis in virus-scanning mail proxy

- Purpose
  - Set up a virus-scanning mail proxy
  - Be familiar with the use of gprof
  - Be familiar with the use of email test tool
  - Study the program flow of ClamAV
  - Learn to analyze the bottleneck in a typical proxy server

Procedure of term-project

1. Install email test tool on Windows PC, and Postfix, AMaViSd and ClamAV on a Linux PC.
2. Compile ClamAV for gprof.
3. Send emulated email with various files (compressed, text, binary...) from the email test tool.
4. Try to find out which functions are most frequently called and most time-consuming in virus scanning. (You may need to track the program flow of ClamAV).

Discussions of term-project

1. List the top ten most frequently called functions and their percentage.
2. List the top ten time-consuming functions and their percentage.
3. Please examine the most time-consuming function and explain why.
4. How much is the difference in virus-scanning time for text files and binary files?
5. How much time is the decompression compared with virus scanning?
6. How much is the execution time increased after turning on the –pg option for gprof?
7. How much time have you spent in this project?

Useful resources for term-project

- Syntax of workload in email test tool: *http://speed.cis.nctu.edu.tw/~ydlin/course/cn/exp/index.html*
- ClamAV: *http://www.clamav.net/*
- AMaViSd: *http://www.ijs.si/software/amavisd/*
- Postfix: *http://www.postfix.org/*
- The data structure in ClamAV: *Y. Miretskiy, A. Das, C. P. Wright and E. Zadok, "Avfs: An on-Access anti-virus file system," USENIX Security Symp., San Diego, CA, 2004.*
- A. Aho and M. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, 18(6), pp. 333-340, 1975.
- S. Wu and U. Manber, "A fast algorithm for multi-pattern searching," *Report TR-94-17,* Department of Computer Science, University of Arizona, 1994.