

Embedded Software Testing to Determine BCM5354 Processor Performance

Sanket Dessai*, Varuna Eswer**

* Department of Computer Engineering, M.S.Ramaiah School of Advanced Studies, Bengaluru, India

** Founder & CEO, Eudaemonic Systems

Article Info

Article history:

Received Jun 12th, 2016

Revised Aug 20th, 2016

Accepted Aug 26th, 2016

Keyword:

BCM5364

MIPS32

OS

TLB

Embedded Software Testing

ABSTRACT

Efficiency of a processor is a critical factor for an embedded system. One of the deciding factors for efficiency is the functioning of the L1 cache and Translation Lookaside Buffer (TLB). Certain processors have the L1 cache and TLB managed by the operating system, MIPS32 is one such processor. The performance of the L1 cache and TLB necessitates a detailed study to understand its management during varied load on the processor. This paper presents an implementation of embedded testing procedure to analyse the performance of the MIPS32 processor L1 cache and TLB management by the operating system (OS). The implementation proposed for embedded testing in the paper considers the counting of the respective cache and TLB management instruction execution, which is an event that is measurable with the use of dedicated counters. The lack of hardware counters in the MIPS32 processor results in the usage of software based event counters that are defined in the kernel. This paper implements embedding testbed with a subset of MIPS32 processor performance measurement metrics using software based counters. Techniques were developed to overcome the challenges posed by the kernel source code. To facilitate better understanding of the testbed implementation procedure of the software based processor performance counters; use-case analysis diagram, flow charts, screen shots, and knowledge nuggets are supplemented along with histograms of the cache and TLB events data generated by the proposed implementation. In this testbed twenty-seven metrics have been identified and implemented to provide data related to the events of the L1 cache and TLB on the MIPS32 processor. The generated data can be used in tuning of compiler, OS memory management design, system benchmarking, scalability, analysing architectural issues, address space analysis, understanding bus communication, kernel profiling, and workload characterisation.

*Copyright © 2016 Institute of Advanced Engineering and Science.
All rights reserved.*

Corresponding Author:

Sanket Dessai,

Department of Computer Engineering,

M.S.Ramaiah School of Advanced Studies,

Bangalore-560058.

Email: sanketdessai0808@gmail.com

1. INTRODUCTION

As Performance measurement is to arrive at the count of the desired event occurring in the system during execution. Processor performance measurement is about measuring the states and events related to the quantum of work done over a period by the processor. The workload on the processor involves computation and movement of data in a defined sequence, and the sequence involves the usage of subsystems such as cache, pipeline, memory, peripherals and so on. The performance measurement can be achieved with the use of either the hardware or the software counters or both. The hardware counters utilises the physical counters

provided by the processor designer, and is loaded with the counter values for measurement from the executing process (operating system or an application) [1]. The software counter on the other hand necessitates code modifications in the executing process that counts the occurrence of an event associated with the performance measurement. The advantage of hardware counters is that it is generally non-intrusive on the instruction execution cycles, while the software counters does require additional instruction cycle for counting the event. The disadvantage with the hardware counters is that the physical counters are limited in number, while the software counters are not. The focus of the hardware counters is fine-tuning either the operating system or the executing process in the identified bottlenecks, while the software counters can be utilised either for general-purpose performance measurement or for specifically measure a bottleneck. To measure the appropriate performance criteria's it is necessary to understand and analyse the requirement analysis properly. It has to be very clear what and how exactly to be measured. In this paper it had been attempted to analyse the requirement analysis for the processor performance measurement.

2. REQUIREMENT SPECIFICATIONS

The functional requirements are:

1. Data collected to be available in the ASCII format accessible for data correlation applications
2. Counter values should be available under the */proc* file system in the file *cpuinfo*
3. Categories of performance measurement is preferred in the area of the hit-miss-refresh cycle related to the:
 - a. Dcache
 - b. Icache
 - c. Scache
 - d. TLB
4. All operations defined for the cache, and the TLB are to be covered
5. Counters values can overflow; hence, the use of a separate variable to count the overflows for each metric will be essential
6. Data structure to be placed in the architecture specific *.../srclinux/linux/include/asm-mips* directory
7. Metric update routines to be placed in the architecture *.../src/linux/linux/arch/mips/kernel* directory
8. Modification to the source code should ensure minimal change in the firmware footprint size
9. The ability to compile the kernel without the metric collection has to be provided
10. Efficiency of the introduced code is *not* the goal as the focus is to get as much as data from the cache and TLB management routines

Study of a concept would necessitate the understanding of the context associated with the MIPS32 processor architecture, and this chapter develops the context of the processor pipeline and cache, performance measurement, data IO methods from the processor implementation board and the setup of the development system.

3. SYSTEM ANALYSIS

The major operations on the cache are either *write-back* or *invalidate*. The write-back operation is used when cache has been updated by the CPU; hence, necessitates the corresponding memory update. The invalidate operation is chosen to access a fresh set of data from the memory. The write-back and invalidate operations is applied on the Icache, Dcache and Scache lines. The flow of cache initialisation is first done on the Icache, followed by the Dcache. Analysing the source code, the cache operations for the BCM5354 processor are as defined by [20] are:

```
#define Index_Invalidate_I 0x00
#define Index_Writeback_Inv_D 0x01
#define Index_Writeback_Inv_SD 0x03
#define Hit_Invalidate_I 0x10
#define Hit_Invalidate_D 0x11
#define Hit_Invalidate_SD 0x13
#define Hit_Writeback_Inv_D 0x15
#define Hit_Writeback_Inv_SD 0x17
#define Hit_Writeback_I 0x18
#define Hit_Writeback_D 0x19
#define Hit_Writeback_SD 0x1b
```

The Figure 1 indicates the use case developed is in reference to the source code to determine the processor performance [20]. The actors that are hardware based are the *Dcache*, *Scache*, *Icache*, and the *TLB*. The actor *Kernel* is software based managing the on-processor caches [4], [5]. The association between the actor *Kernel* and the actors *Dcache*, *Icache*, *Scache*, and the *TLB* are unidirectional as the *Kernel* is waiting on the new set of instruction or data or the virtual address mapping is available in the respective segment and / or line for execution of the scheduled task.

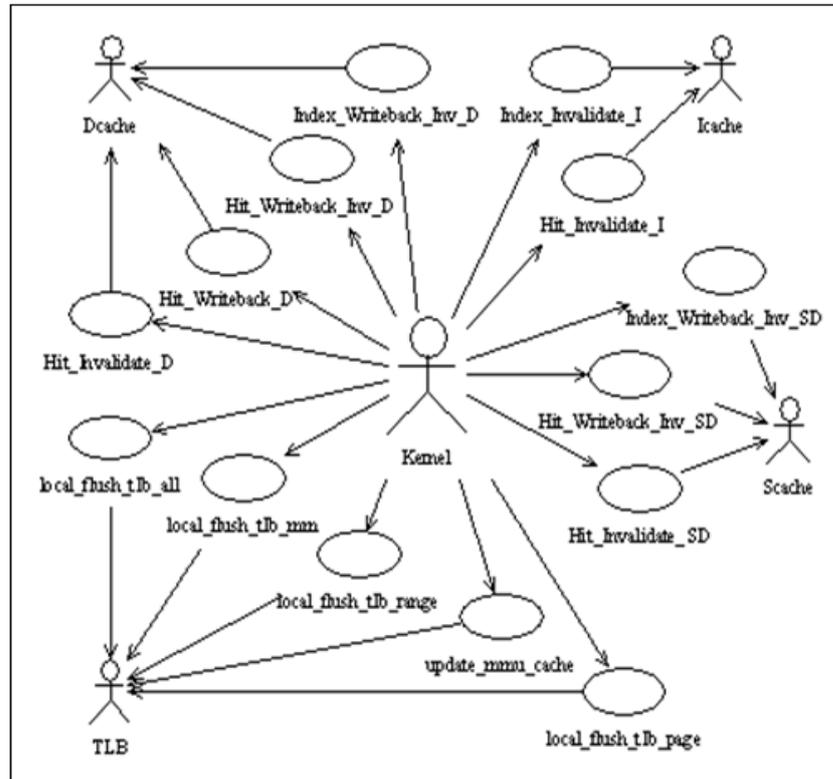


Figure 1. Use cases developed to determine the Processor Performance

4. SOFTWARE DESIGN CONSIDERATION

The listed cache operations [20] are utilised by multiple routines on the caches to flush either the lines or the KSeg0; hence, it is essential to trace the function that used the defined cache operations using unique metrics. The arrived list of metrics is from the perspective of generating the complete picture of the cache operations for the MIPS32 processor. Analysing the source code; [16], [20], [5] for the defined function calls for the caches; I, D and S; along with the TLB operations for the MIPS32 implementation is categorized in the Table 1. The cache operations as visualised from Table 1, the operations can be repeated on the lines or the ways or on KSeg0. The base metric requires a roll over counter as it helps to track the metric data over an extended period. A high rate of activity of the processor will cause a rollover of the metric. The choice of the data type for the metrics; base and the roll over; are *unsigned int*. The data type of the base and the rollover metrics will necessitate a change to an *unsigned long* if need be, with the decision based on the activity of the processor. The designed base metrics data structure is available in Appendix-A.

Table 1. Mapping of Function Calls Operation and Metrics

Listed function name	Cache operation	Metrics to be updated
flush_icache_line_indexed	Index_Invalidate_I on ways	i_way
blast_icache	Index_Invalidate_I and cache unroll of kseg0	i_unroll_kseg0
blast_icache_page_indexed	Index_Invalidate_I and cache unroll on ways	i_unroll_way
flush_icache_line	Hit_Invalidate_I of line	i_line_flush
protected_flush_icache_line	Hit_Invalidate_I of line	i_pline_flush
blast_icache_page	Hit_Invalidate_I and cache unroll on page	i_unroll_page
flush_dcache_line_indexed	Index_Writeback_Inv_D on ways	d_way
blast_dcache_page_indexed	Index_Writeback_Inv_D and cache unroll of ways	d_unroll_way
blast_dcache	Index_Writeback_Inv_D and cache unroll of kseg0	d_unroll_kseg0
flush_dcache_line	Hit_Writeback_Inv_D on line	d_line_flush
blast_dcache_page	Hit_Writeback_Inv_D and cache unroll on page	d_unroll_page
invalidate_dcache_line	Hit_Invalidate_D on line	d_invl_dln
protected_writeback_dcache_line	Hit_Writeback_D on line	d_writeback
flush_scache_line_indexed	Index_Writeback_Inv_SD on ways	s_way
blast_scache_page_indexed	Index_Writeback_Inv_SD and cache unroll on page and ways	s_unroll_pg_ways
blast_scache	Index_Writeback_Inv_SD and cache unroll on kseg0	s_unroll_kseg0
invalidate_scache_line	Hit_Invalidate_SD	s_invl_dln
flush_scache_line	Hit_Writeback_Inv_SD on line	s_line_flush
blast_scache_page	Hit_Writeback_Inv_SD and cache unroll on page	s_unroll_page
fill_icache_line	Fill_Icache_line	i_fill
local_flush_tlb_all		tlb_lflush_all
local_flush_tlb_mm		tlb_lflush_mm
local_flush_tlb_range		tlb_lflush_rng
update_mmu_cache		tlb_updt_mmu
local_flush_tlb_page		tlb_lflush_pg

The metric update is organised on the defined cache operations [20]; hence, the metric computation is switch statement based on the operations. The routines to update the metrics are defined as:

```
void update_cache_metric (int type, int cm_ops);
```

```
void update_tlb_metric (int tlb_fn);
```

The parameter *cm_ops* is the operation defined for the cache, while *type* indicates the operation of the cache on either the line, or the way, or the Kseg0 in the function *update_cache_metrics*. The parameter *tlb_fn* indicates the operation on the TLB. The parameters *type* and *tlb_fn* are defined with a unique value in the file `.../src/linux/linux/include/asm/cache_perf_mips32.h`, and is indicated as follows:

```
#define unroll_c 0xe0 /* cache unroll */
#define line_c 0xe1 /* cache line */
#define kseg0_c 0xe2 /* kseg0 address */
#define ways_c 0xe3 /* mip cache ways */
#define page_c 0xe4 /* cache page */
#define pline_c 0xe5 /* protected cache line */
#define Fill_Icache_line 0xe6 /* fill icache line */
#define lf_tlb_all 0xf0 /* local tlb flush all */
#define lf_tlb_mm 0xf1 /* local flush tlb mm struct */
#define lf_tlb_rng 0xf2 /* local flush tlb range */
#define lf_tlb_pg 0xf3 /* local flush tlb page */
```

```
#define up_tlb_mmu 0xf4 /* update tlb mmu */
```

An example of the method of function call for metric update from the functions indicated under the first column of Table 1 is:

```
update_cache_metric (line_c, Hit_Invalidate_I);
update_tlb_metric (lf_tlb_mm);
```

The metric data for further analysis will be available in the OS provided file */proc/cpuinfo*, individually categorised for the Dcache, Icache, Scache, and TLB. The metric data can be read from the */proc/cpuinfo* file as required, each read provides the current data of the metrics. The format of the data in the */proc/cpuinfo* is indicated below, with the sequence of the metric display is indicated in Table 2:

dcache metrics: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

icache metrics: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0

tlb metrics: 0,0,0,0,0,0,0,0,0,0

Table 2. Metric Display Sequence in *cpu/proc/info*

Cache Type	Metric Sequence
dcache metrics	d_way, d_way_rol, d_invl_n, d_invl_n_rol, d_writeback, d_wb_rol, d_blast, d_blast_rol, d_unroll_kseg0, d_ukseg_rol, d_unroll_way, d_uw_rol, d_line_flush, d_lf_rol, d_unroll_page, d_up_rol
icache metrics	i_way, i_way_rol, i_blast, i_blast_rol, i_unroll_way, i_uw_rol, i_unroll_kseg0, i_ukseg_rol, i_line_flush, i_lf_rol, i_pline_flush, i_plf_rol, i_unroll_page, i_up_rol, i_fill, i_fill_rol
scache metrics	s_invl_n, s_invl_n_rol, s_way, s_way_rol, s_unroll_kseg0, s_ukseg_rol, s_unroll_pg_ways, s_upw_rol, s_line_flush, s_lf_rol, s_unroll_page, s_up_rol
tlb metrics	tlb_lflush_all, tlb_lfa_rol, tlb_lflush_mm, tlb_lfmm_rol, tlb_lflush_rng, tlb_lflrng_rol, tlb_lflush_pg, tlb_lfpg_rol, tlb_updt_mmu, tlb_upmmu_rol

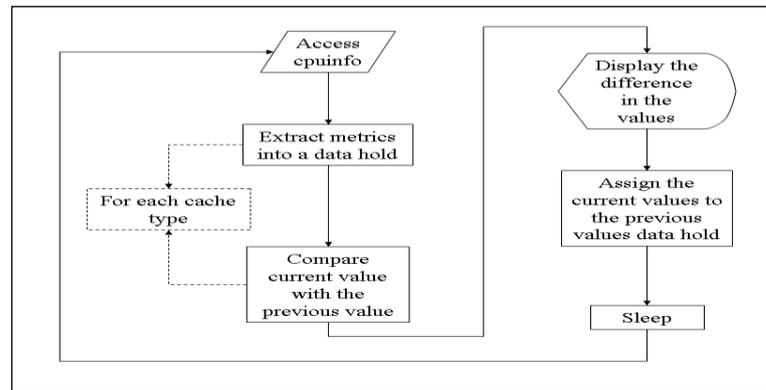


Figure 2. Flow Chart for Metric Analysis Applications

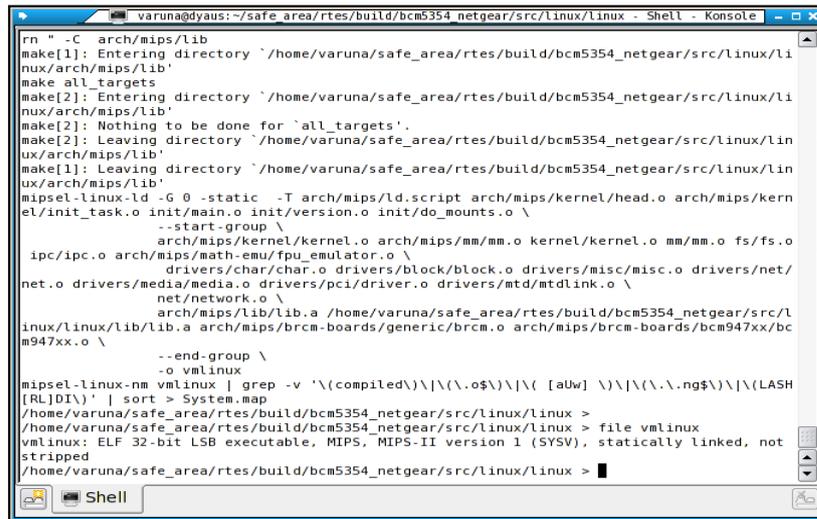
The Appendix-B has the ASCII based screen capture of the file */proc/cpuinfo* from the router indicating the collected metric values.

The Figure 2 indicates a method of utilising the generated performance metric data to for analysis purpose. The metric data read from the */proc/cpuinfo* file is extracted into a structure and is placed in the respective metric variables as indicated in Table 2. The difference between the current and the previous interval metric data is provided to the data analysis application. The duration between the current and the previous interval is controlled through the *Sleep* routine.

Likewise, the data analysis application can be developed to provide a combination of the base and derived metrics; as indicated in Equation 1; to arrive at a set of data suitable for a deep dive analysis of the processor performance. Figures 9 through 12 provides a set of histogram for the Dcache, Icache, Scache, and TLB generated using base metrics data over a period of one-hundred and ten seconds at an interval of five seconds.

5. BUILD ENVIRONMENT AND HARDWARE SETUP

The Linux source code is split into two major sections: architecture independent and architecture dependent. The NETGEAR bundled source code has a third section that is router board specific.



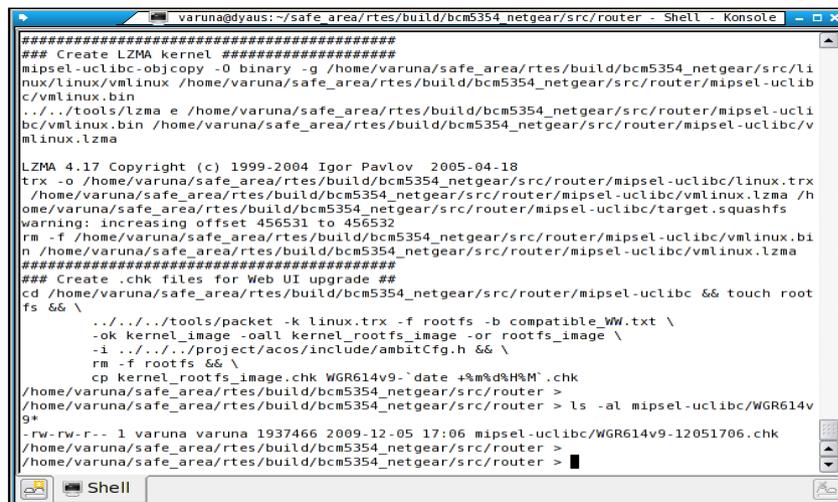
```

varuna@dyaus:~/safe_area/rtes/build/bcm5354_netgear/src/linux/linux - Shell - Konsole
rn " -C arch/mips/lib
make[1]: Entering directory `/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/li
linux/arch/mips/lib'
make all_targets
make[2]: Entering directory `/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/li
linux/arch/mips/lib'
make[2]: Nothing to be done for `all_targets'.
make[2]: Leaving directory `/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/li
linux/arch/mips/lib'
make[1]: Leaving directory `/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/li
linux/arch/mips/lib'
mipsel-linux-ld -G 0 -static -T arch/mips/ld.script arch/mips/kernel/head.o arch/mips/kern
el/init_task.o init/main.o init/version.o init/do_mounts.o \
--start-group \
arch/mips/kernel/kernel.o arch/mips/mm/mm.o kernel/kernel.o mm/mm.o fs/fs.o
ipc/ipc.o arch/mips/math-emu/fpu_emulator.o \
drivers/char/char.o drivers/block/block.o drivers/misc/misc.o drivers/net/
net.o drivers/media/media.o drivers/pci/driver.o drivers/mtd/mtdlink.o \
net/network.o \
arch/mips/lib/lib.a /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/l
linux/linux/lib/lib.a arch/mips/brcm-boards/generic/brcm.o arch/mips/brcm-boards/bcm947xx/bc
m947xx.o \
--end-group \
-o vmlinux
mipsel-linux-nm vmlinux | grep -v '\(compiled\)\|\(\.o$\)\|\([aUw]\)\|\(\.\.ng$\)\|\(LASH
[RLDI]\)' | sort > System.map
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/linux >
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/linux > file vmlinux
vmlinux: ELF 32-bit LSB executtable, MIPS, MIPS-II version 1 (SYSV), statically linked, not
stripped
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/linux/linux >
Shell

```

Figure 3. Screenshot of MIPS32 kernel build

The build process involves compiling the architecture specific code, followed by Linux specific; architecture independent; code compile that results in the kernel image, then followed by compiling the router board specific that results in the firmware image for the NETGEAR WGR614v9 router. The procedure to setup and build the firmware image for the NETGEAR WGR614v9 router is:



```

varuna@dyaus:~/safe_area/rtes/build/bcm5354_netgear/src/router - Shell - Konsole
#####
### Create LZMA kernel #####
mipsel-uclibc-objcopy -O binary -g /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/li
linux/linux/vmlinux /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclib
c/vmlinux.bin
./../tools/lzma e /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclib
c/vmlinux.bin /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/v
mlinux.lzma
LZMA 4.17 Copyright (c) 1999-2004 Igor Pavlov 2005-04-18
trx -o /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/linux.trx
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/vmlinux.lzma /h
ome/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/target.squashfs
warning: increasing offset 456531 to 456532
rm -f /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/vmlinux.bi
n /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc/vmlinux.lzma
#####
### Create .chk files for Web UI upgrade ##
cd /home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router/mipsel-uclibc && touch root
fs.chk && \
./../tools/packet -k linux.trx -f rootfs -b compatible_WW.txt \
-ok kernel_image -oall kernel_rootfs_image -or rootfs_image \
-i ./../././project/acos/include/ambitCfg.h && \
rm -f rootfs.chk \
cp kernel_rootfs_image.chk WGR614v9-`date +%m%d%H%M`.chk
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router >
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router > ls -al mipsel-uclibc/WGR614v
9*
-rw-rw-r-- 1 varuna varuna 1937466 2009-12-05 17:06 mipsel-uclibc/WGR614v9-12051706.chk
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router >
/home/varuna/safe_area/rtes/build/bcm5354_netgear/src/router >
Shell

```

Figure 4. Screenshot of NETGEAR router image build

1. Download and install the *libstdc++.so.5* OS development library.
2. Download the cross compiler tool-chain *TOOLSOURCE_2004_03_31.tgz* from *ftp://ftp.gpl-devices.org/pub/vendors/Belkin* [3] and install the tool-chain in the home directory of the user.
3. Create a symbolic link */opt/brcm* to the tool-chain directory *.../org/tools/brcm* that is available in the user home directory.
4. Download from the internet and install the utility *trx* in the directories */opt/brcm/hndtools-mipsel-linux-3.2.3/bin*, and */opt/brcm/hndtools-mipsel-uclibc-3.2.3/bin*. Ensure the utility *trx* has the execute permission for the owner, group, and the user.
5. Ensure the path to */opt/brcm/hndtools-mipsel-linux-3.2.3/bin* and */opt/brcm/hndtools-mipsel-uclibc-3.2.3/bin* directories are available in the shell environment variable *PATH*.
6. Download the NETGEAR WGR614v9 source code; example *WGR614v9-V1.2.6_18.0.17WW_src.tar.bz2.zip*; available at the website *ftp://downloads.netgear.com/files/GPL* [28], and install the code in a suitable directory under the home directory of the user.
7. Clean the existing object files, and the kernel image *vmlinux* under the Linux and the router sections of the source code tree using the followings indicated steps:

```
cd .../src/router
make clean
make router-clean
cd ../linux/linux
make clean
```

8. Build the Linux kernel image from the source code directory *.../src/linux/linux* using the following steps to generate the MIPS32 kernel image *vmlinux* as indicated in Figure 3


```
make dep
make
```
9. Build the router code in the directory *.../src/router* using the following steps:


```
make
make install
```
10. The WGR614v9 router firmware upgrade image file will be created in the directory *.../src/router/mipsel-uclibc* in the file beginning with the name *WGR614v9*, and ending with the extension *chk* as indicated in Figure 4. An example of the firmware file name is *WGR614v9-12051706.chk* as seen in Figure 4.

The hardware setup is a router board with an implementation of MIPS32 core by BROADCOM processor BCM5354, and is indicated in Figure 5.



Figure 5. NETGEAR WRR614 Router Board

6. PSEUDO-CODE AND IMPLEMENTATION PROCEDURE

- a. The performance metric data collection is implemented under the architecture specific memory management routines available in the locations *.../src/linux/linux/arch/mips/mm*, and the *.../src/linux/linux/include/asm-mips* directories, and the functions listed in Table 1 is available in

- the indicated directories. The pseudo-code for collecting the performance metrics is indicated in the following steps 1 through 5:
- b. Call the function for metric update from the function calls associated with the cache and TLB management, along with the parameters of cache / TLB operation and the type of the operation; either on the ways, or the line, or the KSeg0.
 - c. Update the associated metric for the functions listed in Table 1.
 - d. If the metric counter overflows; wraps to the value zero; then increment the corresponding rollover metric counter.
 - e. Print the values of all the metrics in the `/proc/cpuinfo` file.
 - f. Repeat the steps 1 through 4 for each of the function call listed in Table 1.
 - g. The method of implementation of the pseudo-code is indicated below:
 - h. Locate the section in the source code handling cache and TLB function calls from Linux kernel and memory management routines available in the directory `.../src/linux/linux/kernel` and `.../src/linux/linux/mm` directories.
 - i. Locate the section under the architecture specific source code handling the kernel and the memory management, and identify the section handling the cache and TLB management located under the directories `.../src/linux/linux/arch/mips/mm` and `.../src/linux/linux/arch/mips/kernel`. The associated architecture specific header file is located under the directory `.../src/linux/linux/include/asm-mips`. Specific files that will be used are:
 - j. `.../src/linux/linux/include/asm-mips/mips32_cache.h`
 - k. `.../src/linux/linux/arch/mips/kernel/Makefile`
 - l. `.../src/linux/linux/arch/mips/kernel/proc.c`
 - m. `.../src/linux/linux/arch/mips/mm/tlb-r4k.c`
 - n. Define the header file listing the data structure; as seen in Appendix-A; in a file under the `.../src/linux/linux/include/asm-mips` directory. Example: `cache_perf_mips32.h`
 - o. Define the routines to update and display the metric counters; as listed in Section 5; in a file under the `.../src/linux/linux/arch/mips/kernel` directory, example: `cache_perf_proc.c`. Ensure to initialise the metrics data structure to zero.
 - p. Modify the `.../src/linux/linux/arch/mips/kernel/Makefile` to include the resulting object file generated in step 4].
 - q. Modify the architecture specific listed cache operations; listed in Table 1; in the source code file `.../src/linux/linux/include/asm-mips/mips32_cache.h` and `.../src/linux/linux/arch/mips/mm/tlb-r4k.c` to call the metric update functions; defined in step 4]; along with the necessary parameters. The parameters are listed in Table 1.
 - r. Call the metric display function; created in step 4]; from the file `.../src/linux/linux/arch/mips/kernel/proc.c` to display the data in the `/proc/cpuinfo` file.
 - s. Validate the changes on the hardware by building the firmware image for the NETGEAR WGR614v9 router based MIPS32 processor implementation along with the changes in the source code.

7. TEST SETUP

The prerequisites for executing the test cases involve the following:

- a. Linux Kernel based development system having at least Fedora 8 as the OS is available, and is patched with `libstdc++.so.5` library
- b. BROADCOM provided cross-compilation tool-chain for Linux is installed
- c. NETGEAR WGR614v9 router is available as indicated in Figure 5
- d. Network connectivity between the router and the development system is established
- e. Facility to transfer files the using FTP utility is available on the development system, refer F.2 in Appendix-F
- f. Facility to login into the router from the development system using the telnet utility is available, refer F.3 in Appendix-F
- g. NETGEAR provided Linux based source code for the WGR614v9 is installed on the development system
- h. `telnetenable-0.3` tools has been installed on the development system for enabling the telnet daemon on the router
- i. Source code having the performance metric collection routines is available in the files

`.../src/linux/linux/include/asm-mips/mips32_cache.h`

`.../src/linux/linux/arch/mips/kernel/Makefile`

```
.../src/linux/linux/arch/mips/kernel/proc.c
.../src/linux/linux/arch/mips/mm/tlb-r4k.c
.../src/linux/linux/arch/mips/kernel/cache_perf_proc.c
.../src/linux/linux/include/asm-mips/cache_perf_mips32.h
```

7.1. Test Cases

The classification; [22], [10]; of the test cases are: build validation, data display, kernel message, and processor load with the test case tag prefix: TB, TD, TK, and TL respectively.

Test case tag: TB-01

Description: Validate the cross-compilation tool-chain for the BROADCOM BCM5354 MIPS32 processor for error free compilation.

System state: The tool-chain is installed on the development system as per the steps *a* through *e* indicated in Section 5

Execution steps: A simple C programming code for printing the string 'hello world' having the file name *c1.c* is created. In the shell, execute the command:

```
mipsel-uclibc-gcc c1.c.
```

Expected result: The cross-compilation should result in the generation of the file *a.out*.

Test case tag: TB-02

Description: Validate the cross-compilation on the development system.

System state: The test case TB-01 has been executed

Execution steps: Execute the command:

```
file a.out
```

Expected result: The command should result in the display:

```
a.out: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), dynamically linked (uses shared libs), not stripped
```

Test case tag: TB-03

Description: Validate the generated binary file *a.out* on the router.

System state: The test case TB-02 has been executed. The router is available with the ability to telnet into the router enabled, and FTP data into the router is setup. Refer to the Sections F.2 and F.3 in Appendix-F to setup the FTP and enabling telnet on the router.

Execution steps: Telnet into the router. FTP the cross-compiled binary *a.out* into the */tmp* directory of the router from the development system. Execute the following command on the router:

```
./a.out
```

Expected result: The command should print the string: *hello world*

Test case tag: TB-04

Description: Generate the firmware image file using the NETGEAR provided source code.

System state: The test case TB-02 has been executed. NETGEAR provided source code for WGR614v9 has been installed as indicated in Section 5 step *f*.

Execution steps: Execute the steps *g* through *i* indicated in Section 5.

Expected result: Linux kernel image *vmlinux* will be created in the directory *.../src/linux/linux*. NETGEAR WGR614v9 firmware image will be created in the *.../src/router/mipsel-uclibc* directory in a file beginning with the name *WGR614v9* and ending with *.chk*. Refer to Figures 3 and 4 under Section 5.

Test case tag: TB-05

Description: Reprogram the router using the built firmware image.

System state: The test case TB-04 has been executed.

Execution steps: Start the NETGEAR web based administration GUI. Select the menu 'Router Upgrade'. Locate and select the firmware image file on the development system. Reprogram the router with the new firmware image.

Expected result: Post reprogramming, the router comes back online to the web based administration GUI. Refer to Figures 6 and 7 under the Section 5.

Test case tag: TB-06

Description: Modify the html section of the router code to validate if firmware bundles the introduced changes in the source code.

System state: The test case TB-05 should have been executed. NETGEAR provided source code for WGR614v9 has been installed as indicated in Section 5 step *f*.

Execution steps: Execute the instructions indicated in step 1) under Section 5. Re-execute the test cases TB-04 and TB-05. Execute the instruction in step 7) under Section 5.

Expected result: Identical to the results in test cases TB-04 and TB-05. Additionally, the web-based administration GUI should display the information as seen in Figure .3 under Section 5 later on.

Test case tag: TB-07

Description: Build the router firmware image using the source code having the performance metric collection routines.

System state: Test case TB-06 should have been executed. Source code containing the metric collection routines are installed on the development system.

Execution steps: Execute the steps *g* through *i* indicated in Section 5.

Expected result: Linux kernel image *vmlinux* will be created in the directory *.../src/linux/linux*. NETGEAR WGR614v9 firmware image will be created in the *.../src/router/mipsel-uclibc* directory in a file beginning with the name *WGR614v9* and ending with *.chk*. Refer to Figures 3 and 4 under Section 5.

Test case tag: TB-08

Description: Re-execute the test cases TB-04 and TB-05 with the firmware having metric collection routines.

System state: Test case TB-07 should have been executed. The source code for the WGR614v9 containing the performance metric collection routines.

Execution steps: Re-execute the test cases TB-04 and TB-05.

Expected result: Identical to the results in test cases TB-04 and TB-05.

Test case tag: TB-09

Description: Enable processor performance metric collection in the source code for WGR614v9.

System state: The test case TB-08 should have been executed.

Execution steps: Enable the *PERF_ENABLE* definition by modifying the *.../src/linux/linux/include/asm-mips/cache_perf_mips32.h* file. Re-execute the test cases TB-04 and TB-05.

Expected result: Identical to the results in the test cases TB-04 and TB-05.

Test case tag: TB-10

Description: Enable kernel message from the performance metric collection implementation.

System state: The test case TB-08 should have been executed.

Execution steps: Enable the *PERF_DEBUG* definition by modifying the *.../src/linux/linux/include/asm-mips/cache_perf_mips32.h* file. Re-execute the test cases TB-04 and TB-05.

Expected result: Identical to the results in test cases TB-04 and TB-05.

Test case tag: TB-11

Description: Disable the processor performance metric collection in the source code for WGR614v9.

System state: The test case TB-09 should have been executed.

Execution steps: Disable the *PERF_ENABLE* definition by modifying the file *.../src/linux/linux/include/asm-mips/cache_perf_mips32.h* file. Re-execute the test cases TB-04 and TB-05.

Expected result: Identical to the results in the test cases TB-04 and TB-05.

Test case tag: TB-12

Description: Disable the kernel print messages from the performance metric collection routines.

System state: Test case TB-10 should have been executed.

Execution steps: Disable the *PERF_DEBUG* definition by modifying the file *.../src/linux/linux/include/asm-mips/cache_perf_mips32.h* file. Re-execute the test cases TB-04 and TB-05.

Expected result: Identical to the results in the test cases TB-04 and TB-05.

Test case tag: TD-01

Description: Display the cache and TLB performance metrics available in the file */proc/cpuinfo* with *PERF_ENABLE* enabled.

System state: Test case TB-09 should have been executed.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/cpuinfo`

Expected result: The format of the command output should be of the format as seen in Appendix-B for the same command.

Test case tag: TD-02

Description: Try to display the cache and TLB performance metrics from `/proc/cpuinfo` file without `PERF_ENABLE` set.

System state: The test cases TB-11 should have been executed.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/cpuinfo`

Expected result: The output of the command should *not* display the metrics data.

Test case tag: TK-01

Description: Display the cache and TLB performance metrics available in the file `/proc/kmsg` with `PERF_DEBUG` enabled.

System state: Test cases TB-09 and TB-10 should have been executed.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/kmsg`

Expected result: The output of the command should resemble the listing as seen for the identical command in Appendix-C.

Test case tag: TK-02

Description: Try to display the cache and TLB performance metrics from `/proc/cpuinfo` file with `PERF_DEBUG` disabled.

System state: The test cases TB-09 and TB-12 should have been executed.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/kmsg`

Expected result: The output of the command should not resemble the listing as seen for the identical command in Appendix-C.

Test case tag: TL-01

Description: Check the load average on the router with activity.

System state: The test case TK-01 should have been executed.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/loadavg`

Expected result: The output of the command indicates a value *greater than* zero, and should resemble an identical command output listing as seen in Appendix-B.

Test case tag: TL-02

Description: Check the load average on the router with no activity.

System state: The test cases TL-01 should have been executed and no additional activity from the user for about five minutes.

Execution steps: Login into the router; refer to F.3 in Appendix-F. Execute the command:

`cat /proc/loadavg`

Expected result: The output of the command indicates a value *equal to* zero, and should resemble an identical command output listing as seen in Appendix-B.

7.2. Build Environment Validation

The procedure to validate the tool-chain and the build environment is:

- Modify the file `.../project/acos/www/html/LGO_logout.htm` file by introducing a web page printable string that can be utilised for identification of the file modification. As an example, the string introduced in the `LGO_logout.htm` will be "varuna testing on 24nov09 --"
- Repackage the router image file using the command under the `.../src/router` directory in the source code tree:
- make install
- Ensure that the router image file beginning with `WGR614v9` and ending with `.chk` has been created in the `.../src/router/mipsel-uclibc` directory.
- Launch the NETGEAR web based administration tool, and select the menu for router upgrade as indicated in Figure 6

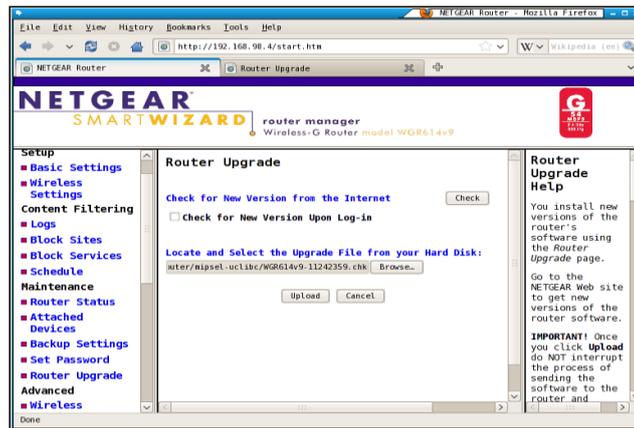


Figure 6. Router upgrade menu using the web admin GUI

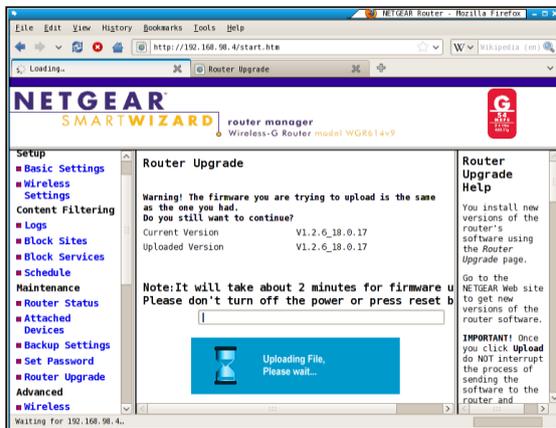


Figure 7. Router firmware WRR614 Router Board

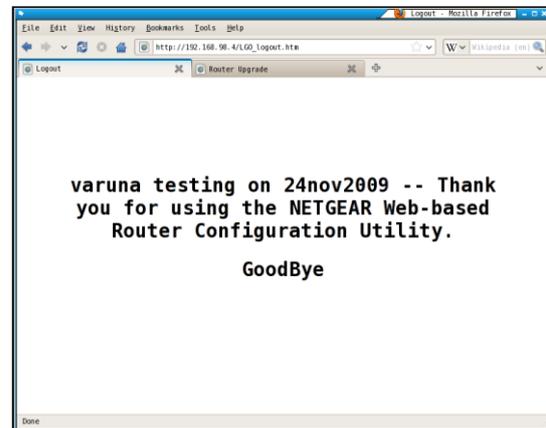


Figure 8. Router firmware upgrade in progress with NETGEAR Administration

- Browse the location that has the binary file in the `.../src/router/mipsel-ucLibc` directory of the source code tree and upload the `.chk` file into the router as seen in the Section 5 Figure 4.
- The upgrade information will be displayed as seen in Figure 7. The router checks for the current firmware version versus the version of the firmware being uploaded.
- Post upgrade, choose the menu to logout from the NETGEAR web based administration tool, the string introduced in the step 1) should be displayed as indicated in Figure 8

8. RESULTS

The result of processor performance measurement involves in understanding the metrics and makes its interpretations and concludes the results associated with these interpretations. Based on these interpretations wherever required modify the code and analyse the performance to understand the performance measurement.

8.1. Metric Interpretation

Consider an example of the cache operation `Index_Invalidate_I`, and as seen in Table 1 under Section 4, with the operations performed on the Icache line, Icache ways and on the KSeg0. In order to get an exact figure of the number of times the operation `Index_Invalidate_I` was performed, the method is:

$$\text{Number of Index_Invalidate_I} = i_way + i_unroll_kseg0 + i_unroll_way \quad (1)$$

Likewise, for each defined operation of the cache and TLB, the corresponding metrics indicated in Table 1 are to be added as indicated in Equation 1. The combination of metrics yields a derived metric, while the

value of the individual metrics is termed as the base metric. The Column 3 of the Table 1 indicates the base metrics. The data provided by the metrics; base and derived; can be utilised to draw a histogram tracing the processor cache activity over a period.

The data generated by the metric analysis tool *zmet* has been transformed into histograms; Figures 9, 10, 11 and 12; for the chosen set of metrics for the Dcache, Icache, Scache, and the TLB. The y-axis for the histograms indicates the number of events, while the x-axis is the time in seconds. The data collected for the histogram has been every five seconds, over a period of one-hundred and ten seconds. The commands that were executed for data collection are standard OS commands that for displaying the contents of the directory and files under the */proc* file system, etc. In order to get a better perspective of the effect of the commands, consider an example of the command: *cat /proc/cpuinfo*. The execution of the command has the following indicative steps:

1. The shell spawns a new process by creating a process table entry, and copies the file descriptors. The process will be associated with the *cat* application.
2. Scheduler places the created process for execution.
3. The code for *cat* is accessed, brought into the memory from the file system.
4. Executing the code, the file name is parsed. In this case, the file is */proc/cpuinfo*.
5. The file table entry is accessed to check if the file is a directory, if yes, exit.
6. Open the file for reading.
7. Read the line until end of line mark and store in the buffer.
8. Access the character device driver for console terminal.
9. Open the device for writing.
10. Get back to the file read operation, and now call the print routine to output data to the character device file.
11. Repeat the Steps 7 through 10 until end of file.
12. Release the system resources occupied for reading the file */proc/cpuinfo*.
13. Release the system resources occupied by the *cat* application.
14. Release the system resources associated with process table entry.

The Steps 1 through 14 has multiple instructions each that either can be in the memory or on the flash file system of the router. The possibility of the instruction being in the memory either can be due to an earlier execution of the same instruction or has been pre-fetched. The indicated steps when viewed for the perspective of the cache and TLB usage, non-availability of the instruction in the memory will cause a flush of the Icache. This operation will cause a Dcache flush, and if need be, a flush of the TLB. Referring to the histograms in Figure 9, 10, 11 and 12, the first five-seconds of data collection has seen a high activity of the Dcache and Icache when compared to the TLB and Scache. This corresponds to the pre-fetch of the instruction into the memory, then to the Icache as seen in Figure 1.

The execution of the instruction will cause a fetch of the data from the memory into the Dcache as seen in Figure 3. Prior to loading of the Icache or Dcache, the entire KSeg0, or the ways or the lines are flushed, with the simultaneous update of the TLB. The each activity on the Icache, Dcache, Scache and TLB is a measurable event; hence, the corresponding metrics are updated. The flush activity either can be a writeback or invalidate of the respective caches. The four histograms; Figures 9,10,11 and 12 have to be visualised simultaneously in order to arrive at the cache and TLB activity on the processor.

8.2. Analysis of the Code Modifications

The lines of code that were added into the stock source code [5] have been listed in the Table 3. The files *mips32_cache.h*, *Makefile*, *proc.c*, *tlb-r4k.c*, *Config.h*, *applets.h* and the *usage.h* were part of the source code tree, while the files *cache_perf_proc.c*, *cache_perf_mips32.h* and *zmet.c* were added into the source code tree as part of the implementation of the cache and TLB performance measurement metrics for MIPS32 architecture. The corresponding location of modification and introduction of the new files has been indicated in Table 3. The total number lines that were added into the source code tree has been 1009 (one-thousand and nine), counted without the comments or introduced blank lines for formatting of the code.

8.3. Analysis of the Code Modifications

The *\$linux* and *\$busybox* used in Table 3 indicates the source code location *.../src/linux/linux*, and *.../src/router/busybox* respectively. The Table 4 lists the comparison of the file size; in bytes; listed under Rows 2 through 5, 7 through 10, 12 through 15, 17 through 20, and 22 through 25; for combinations of router firmware image built on the development system indicated in Rows 1, 6, 11, 16, and 21 respectively. The column 3 lists the file sizes for the corresponding file type indicated in Column 2. The focus of the modifications has been to ensure the size of the vmlinux, busybox, target.squashfs and the firmware image do not exceed the respective size indicated in for Row 1 Vanilla. The target.squashfs is the file system that will be programmed into the router, while the firmware image is the result of the Step j of Section 5.

Table 3. Lines of Code added without comments and blank lines.

File name	Line of code added
<i>\$linux/include/asm/mips32_cache.h</i>	69
<i>\$linux/arch/mips/kernel/Makefile</i>	1
<i>\$linux/arch/mips/kernel/proc.c</i>	,2
<i>\$linux/arch/mips/mm/tlb-r4k.c</i>	20
<i>\$busybox/{Config.h, applets.h, usage.h}</i>	1, 3, 4
<i>\$linux/arch/mips/kernel/cache_perf_proc.c</i>	499
<i>\$linux/include/asm/cache_perf_mips32.h</i>	81
<i>\$busybox/zmet.c</i>	329

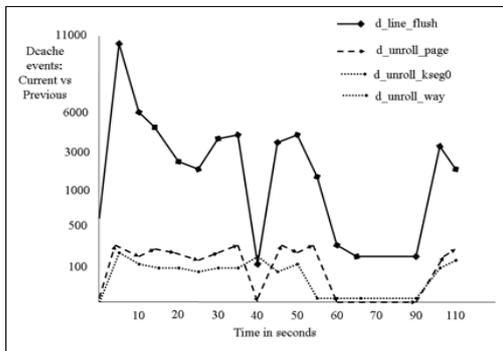


Figure 9. Histogram of Dcache events

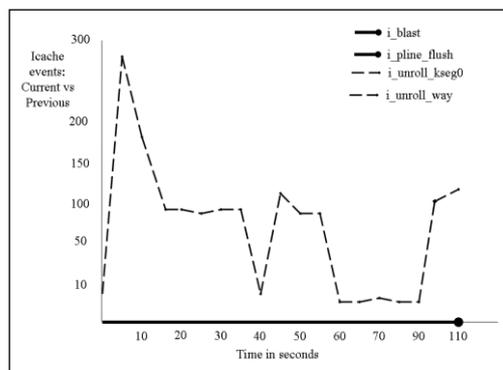


Figure 10. Histogram of Icache events

Table 4. Listing of files sizes.

1	Vanilla (water mark)	
2	vmlinux	1,864,769
3	busybox	242,644
4	target.squashfs	1,409,024
5	firmware image	1,867,834
6	With printk & metrics	
7	vmlinux	1,869,878
8	busybox	242,644
9	target.squashfs	1,404,928
10	firmware image	1,863,738
11	With printk & metrics, trimmed html pages & busybox	
12	vmlinux	1,869,878
13	busybox	238,180
14	target.squashfs	1,388,544
15	firmware image	1,847,354
16	With printk & metrics, trimmed html & busybox, zmet application	
17	vmlinux	1,869,878
18	busybox	242,788
19	target.squashfs	1,392,640
20	firmware image	1,851,450
21	Without printk, with metrics, trimmed html & busybox, zmet application	
22	vmlinux	1,865,782
23	busybox	242,788
24	target.squashfs	1,392,640
25	firmware image	1,851,450

Referring to Table 3, the additional code added to the kernel will increase the kernel size, and this increase is seen in the Rows 7, 12, 17 and 22 of Table 4, for vmlinux. The Rows 11 through 25 of Table 4 indicates the usage of trimmed html pages, and was achieved for additional space on the router flash for the files under the `.../project/acos/www/html` directory. Likewise, the default utilities; cp, ping, etc., bundled in busybox were trimmed. The corresponding reduction in the size of busybox is seen in Row 13 Column 3 of Table 4. The metric analysis tool zmet has been integrated into busybox; linked into the directory `/usr/bin`; has increased the size of busybox as seen in Rows 18 and 23 Column 3 of Table 4. Although the size of the kernel vmlinux and busybox has increased with the integration of the performance measurement metrics; Column 3 Rows 7, 12, 17, and 22 of Table 4, the overall size of the target. squashfs is less than the corresponding size indicated under Row 4 Column 3 under Table 4.

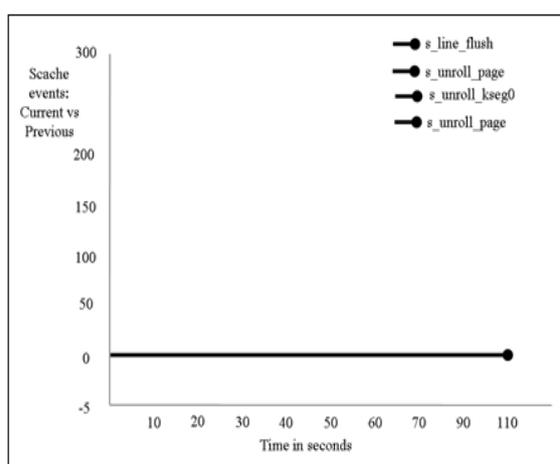


Figure 11. Histogram of Scache events

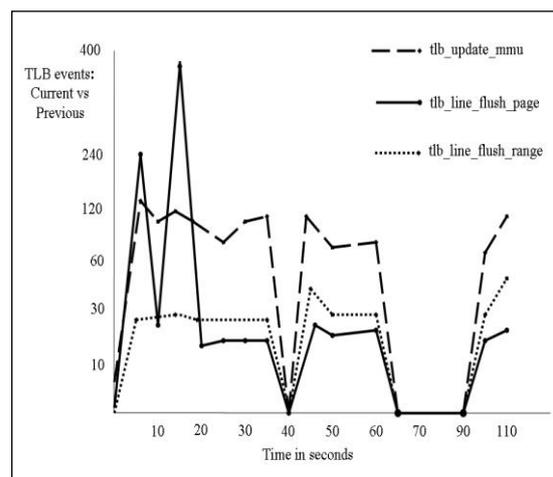


Figure 12. Histogram of TLB events

9. CONCLUSION

Hardware counters are not available on the MIPS32 architecture; hence, performance measurement necessitates the usage of software counters. Software counters are defined in the kernel to track the events being measured; hence, there is a possibility of the counters overflowing. Usage of overflow counters has been incorporated. L1 Cache and TLB management is external to the hardware; hence, the OS manages the cache and TLB. Twenty-seven base metrics has been defined for measuring the L1 cache and TLB events on the MIPS32 architecture implementation, with an equal number of overflow counters. Linux Kernel 2.4.20 has been instrumented for metric data collection on NETGEAR WGR614v9 router having an implementation of MIPS32 processor from BROADCOM. The generated data aids to monitor, optimise, tune, model and benchmark the system that comprise of the architecture, its subsystems, and the executing processes; be it the operating system or the applications using the operating system. A metric data extraction application *zmet* has been developed to aid analysis of the activities of the L1 cache and TLB through validation. The challenges that were offered by the source code and the NETGEAR WGR614v9 router during the implementation phase were:

1. Space on router for additional code 9717 bytes of binary code (metrics + zmet)
2. Source code compatibility with loadable modules
3. Documentation support with the source code
4. Command line parameters processing
5. Proc file writing

REFERENCES

- [1] Broadcom Corporation, "BCM5354 Product Brief – Optimised 802.11G Router with Broadrange," *Broadcom Corporation*, 5354-PB01-R, 11 Sept 2007
- [2] BELKIN Inc., "Vendor toolchains," <ftp://ftp.gpl-devices.org/pub/vendors/Belkin>, Sept 2009
- [3] Brinkley Sprunt, "The Basics of Performance Monitoring Hardware," *0272-1732/02/2002 IEEE*, 2002

- [4] Don Anderson, "Universal Serial Bus System Architecture," *Second Edition, Addison-Wesley Developer's Press*, ISBN 0-201-46137-4, 2001
- [5] David S. Miller, "Cache and TLB Flushing Under Linux," *Linux Documentation*, .../Documentation/cachetlb.txt, 2001
- [6] David S. Miller and Ralf Baechle, "arch/mips/mm/tlb-r4k.c," *Linux Kernel 2.4.20 source code*, 1997
- [7] Dominic Sweetman, "See MIPS Run," *Second Edition, Morgan Kaufmann Publishers*, ISBN 13: 978-0-12-088421-6, 2007
- [8] Guy G. F. Lemieux, "Hardware Performance Monitoring in Multiprocessors," *Masters degree thesis - University of Toronto*, 1996
- [9] Gregory S. Freeland, Joel L. Gross, and Jose A. Laboy, "Tuneable Processor Performance Benchmarking," *USA Patent 20070136726 A1*, <http://www.freepatentsonline.com/20070136726.html>, June 14 2007
- [10] IEEE, "Software Engineering Body of Knowledge," *IEEE, 2004*, pp 2-1 to 2-10, <http://www.swebok.org>
- [11] Jack Dongarra, Kevin London, Shirley Moore, Phil Mucci, and Dan Terpstra, "Using PAPI for hardware performance monitoring on Linux systems," *Innovative Computing Laboratory, University of Tennessee*, <http://icl.cs.utk.edu/publications/pub-papers/2001/papi-linuxrev1.pdf>, Sept, 2009
- [12] Lance M. Berc, Sanjay Ghemawat, Moniika H. Henzinger, Richard L. Sites, Carl A. Waldspurger, and William E. Weihl, "High Frequency Sampling of Processor Performance Counters," *USA Patent 5796939*, <http://freepatentsonline.com/5796939.html>, Aug 18 '98, Oct, 2009
- [13] M. Bolado, H. Posadas I, J. Castillo, P. Huerta I, P. Sánchez, C. Sánchez, H. Fouren, and F. Blasco, "Platform based on Open-Source Cores for Industrial Applications," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition 1530-1591/04 IEEE*, 2004
- [14] Marco Zaghera, Brond Larson, Steve Turner, and Marty Itzkowitz, "Performance Analysis using the MIPS R10000 Performance Counters," *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing (SC'96)*, 0-89791-854-1/96, 1996
- [15] Michael Huy Phan, "Performance Measurement for Embedded Systems," *USA Patent 6643609 B2, Nov 4 2003*, <http://www.freepatentsonline.com/6643609.html>
- [16] MIPS Technologies, "MIPS32 ® 4Kc™ Processor Core Datasheet – Revision 01.03," *MIPS Technologies Inc., MD00247*, June 2000
- [17] MIPS Technologies, ".../include/asm-mips/mips32_cache.h," *Linux Kernel 2.4.20 source code*, 1996
- [18] M. Warner Losh, "An overview of FreeBSD/mips," *AsiaBSDCon 2009, February 2009*, <http://2009.asiabsdcon.org/papers/abc2009-P4B-paper.pdf>, Aug 2009
- [19] NETGEAR Inc., "Wireless-G Router WGR614v9 Reference Manual," *NETGEAR Inc., 202-10308-01*, May 2008
- [20] Paul J. Drongowski, "Basic Performance Measurements for AMD Athlon 64, AMD Opteron and AMD Phenom Processors," *Advanced Micro Devices, Inc.*, http://developer.amd.com/Assets/Basic_Performance_Measurements.pdf, September 25 2008, (Nov 2009)
- [21] Ralf Baechle, "Cache operations for the cache instruction," *Linux kernel 2.4.20, .../include/asm-mips/cacheops.h*, 2002
- [22] Roger S. Pressman., "Software engineering: a practitioner's approach," - *5th ed, McGraw-Hill*, ISBN 0-07-365578-3
- [23] S. Browne, J Dongarra, N. Garner, G. Ho, P. Mucci, "A Portable Programming Interface for Performance Evaluation on Modern Processors," *Computer Science Department, University of Tennessee, and Oak Ridge National Laboratory*, <http://icl.cs.utk.edu/publications/pub-papers/2000/papi-journal-final.pdf>, Sept 2009
- [24] Shirley Moore, Patricia Teller, and Michael Maxwellll, "Efficiency and Accuracy Issues for Sampling vs. Counting Modes of Performance Monitoring Hardware," *University of Tennessee-Knoxville and University of Texas-El Paso*
- [25] Tsuyoshi Nagao and Hitoshi Suzuki, "Processor System and Performance Measurement Method for Processor System," *USA Patent 20070277178A1, Nov 29 2007*, <http://www.freepatentsonline.com/20070277178.html>
- [26] Warner Losh, "A Brief history of FreeBSD/MIPS," *BSDCan 2008 Canada*, <http://www.freebsd.org/~imp/bsdcan2008.pdf>, Aug 2009
- [27] Wendy Korn, Patricia J. Teller, and Gilbert Castillo, "Just how accurate are performance counters," *University of Texas as El Paso*, <http://www.cs.utep.edu/pcat/papers/IPCCC2001paper.pdf>, Aug 2009
- [28] Wiplove Mathur, and Jeanine Cook, "Improved Estimation for Software Multiplexing of Performance Counters," http://www.ece.nmsu.edu/~jecook/pubs/jcook_multiplexing.pdf, Sept 2009
- [29] WGR614v9 router source code, <ftp://downloads.netgear.com/files/GPL>, Sept 2009
- [30] Xiao Zhang, Sandhya Dwarkadas, Girts Folkmanis, and Kai Shen, "Processor Hardware Counter Statistics As A First-Class System Resource," *Department of Computer Science, University of Rochester*, 2007, <http://www.cs.rochester.edu/u/sandhya/papers/hotos07.pdf>, Aug 2009
- [31] FreeBSD/MIPS Project, <http://www.freebsd.org/platforms/mips.html>, Aug 2009
- [32] Toolchains, <http://www.linux-mips.org/wiki/Toolchains>, Aug 2009
- [33] Vendor toolchains, <ftp://ftp.gpl-devices.org/pub/vendors/Belkin>, Sept 2009

BIOGRAPHIES OF AUTHORS



Sanket Dessai received his BSc, MSc degree in Physics and MSc [Engg] in Real-Time Embedded Systems from Goa University, India and Coventry University, UK. He is in Academic position as Assistant Professor. He is also hardcore Consultant, Researcher, and Trainer for many MNC. His research interests include the field of System on Chip Design, Embedded Systems, MEMS/NEMS Engineering, Nanophysics and Nanotechnology, Solid State Physics and Engineering and Photonics.



Varuna Eswer, received his B.Tech in Computer Science and Engineering and MSc [Engg] in Real-Time Embedded Systems from Mysore University, India and Coventry University, UK. His research interests include the field of Open BSD, Operating Systems, Embedded Systems and System on Chip Design.

APPENDIX A. DATA STRUCTURE FOR PERFORMANCE METRICS

Data structure for performance metrics

```

/*
 * Structure containing the cache performance metrics for mips32 processor.
 * Can consider to change the roll over counter from __u32 to
 * __u64, and depends on the size of the kernel & the activity
 * noticed with respect to the roll over counters.
 */
struct perf_m
{
    /* dcache related metrics */
    __u32 d_way;          /* ways updates */
    __u32 d_way_roll;    /* ways roll */
    __u32 d_invl_d_ln;   /* invalidate line */
    __u32 d_invl_n_roll; /* invalidate line roll over */
    __u32 d_writeback;   /* writeback counter */
    __u32 d_wb_roll;     /* writeback counter roll over */
    __u32 d_blast;       /* blast counter */
    __u32 d_blast_roll;  /* blast counter roll over */
    __u32 d_unroll_kseg0; /* cache unroll for kseg0 */
    __u32 d_ukseg_roll;  /* cache unroll for kseg0 roll over */
    __u32 d_unroll_way;  /* cache unroll on ways */
    __u32 d_uw_roll;     /* cache unroll on ways roll over */
    __u32 d_line_flush;  /* cache line flush */
    __u32 d_lf_roll;     /* cache line flush roll over */
    __u32 d_unroll_page; /* cache unroll */
    __u32 d_up_roll;     /* cache unroll roll over counter */

    /* icache related metrics */
    __u32 i_way;          /* ways updates */
    __u32 i_way_roll;    /* ways update roll */
    __u32 i_blast;       /* blast counter */
    __u32 i_blast_roll;  /* blast counter roll over */

```

```

__u32 i_unroll_way; /* cache unroll on ways */
__u32 i_uw_roll; /* cache unroll on ways roll over */
__u32 i_unroll_kseg0; /* cache unroll for kseg0 */
__u32 i_ukseg_roll; /* cache unroll for kseg0 roll over */
__u32 i_line_flush; /* cache line flush */
__u32 i_lf_roll; /* cache line flush roll over */
__u32 i_pline_flush; /* cache line flush */
__u32 i_plf_roll; /* cache line flush roll over */
__u32 i_unroll_page; /* cache unroll */
__u32 i_up_roll; /* cache unroll roll over counter */
__u32 i_fill; /* cache line fill */
__u32 i_fill_roll; /* cache line fill roll over counter */

/* scache related metrics */
__u32 s_invl_d_ln; /* invalidate line */
__u32 s_invl_n_roll; /* invalidate line roll over */
__u32 s_way; /* ways update */
__u32 s_way_roll; /* ways update roll over */
__u32 s_unroll_kseg0; /* cache unroll for kseg0 roll over */
__u32 s_ukseg_roll; /* cache unroll for kseg0 roll over */
__u32 s_unroll_pg_ways; /* cache unroll, page and ways */
__u32 s_upw_roll; /* cache unroll, page and ways roll over */
__u32 s_line_flush; /* cache line flush */
__u32 s_lf_roll; /* cache line flush roll over */
__u32 s_unroll_page; /* cache unroll */
__u32 s_up_roll; /* cache unroll roll over counter */

/* tlb related metrics */
__u32 tlb_lflush_all; /* tlb local flush all */
__u32 tlb_lfa_roll; /* tlb local flush all roll */
__u32 tlb_lflush_mm; /* tlb local flush of mm */
__u32 tlb_lfmm_roll; /* tlb local flush of mm roll over */
__u32 tlb_lflush_rng; /* tlb local flush of a range */
__u32 tlb_lflrng_roll; /* tlb local flush of a range roll over */
__u32 tlb_lflush_pg; /* tlb local flush of page */
__u32 tlb_lfpg_roll; /* tlb local flush of page roll over */
__u32 tlb_updt_mmu; /* tlb update mmu */
__u32 tlb_upmmu_roll; /* tlb update mmu roll over */
};

```

APPENDIX B. METRIC DATA DISPLAY

Metric data display in /proc/cpuinfo

Trying 192.168.98.4...

Connected to 192.168.98.4.

Escape character is '^['.

BusyBox v0.60.0 (2010.03.04-08:34+0000) Built-in shell (msh)

Enter 'help' for a list of built-in commands.

```
# cat /proc/loadavg
```

```
0.04 0.01 0.00 2/17 116
```

```
### cat /proc/cpuinfo
```

```
system type      : Broadcom BCM5354 chip rev 3
```

```

processor      : 0
cpu model     : BCM3302 V2.9
BogoMIPS      : 237.56
wait instruction : no
microsecond timers : yes
tlb_entries   : 32
extra interrupt vector : no
hardware watchpoint : no
VCED exceptions : not available
VCEI exceptions : not available
unaligned_instructions : 0
dcache metrics: 0,0,0,0,41,0,0,0,10002,0,10002,0,115440,0,13953,0
icache metrics: 0,0,0,0,10002,0,10002,0,0,0,41,0,0,0,0,0
scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0
tlb metrics: 6,0,296,0,3915,0,1493,0,8074,0
### cat /proc/cpuinfo
system type   : Broadcom BCM5354 chip rev 3
processor     : 0
cpu model     : BCM3302 V2.9
BogoMIPS      : 237.56
wait instruction : no
microsecond timers : yes
tlb_entries   : 32
extra interrupt vector : no
hardware watchpoint : no
VCED exceptions : not available
VCEI exceptions : not available
unaligned_instructions : 0
dcache metrics: 0,0,0,0,51,0,0,0,10280,0,10280,0,131393,0,14396,0
icache metrics: 0,0,0,0,10280,0,10280,0,0,0,51,0,0,0,0,0
scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0
tlb metrics: 6,0,304,0,3991,0,1550,0,8327,0
### cat /proc/loadavg
0.02 0.01 0.00 2/18 121
#
# cat /proc/cpuinfo
system type   : Broadcom BCM5354 chip rev 3
processor     : 0
cpu model     : BCM3302 V2.9
BogoMIPS      : 237.56
wait instruction : no
microsecond timers : yes
tlb_entries   : 32
extra interrupt vector : no
hardware watchpoint : no
VCED exceptions : not available
VCEI exceptions : not available
unaligned_instructions : 0
dcache metrics: 0,0,0,0,64,0,0,0,10760,0,10760,0,155797,0,15133,0
icache metrics: 0,0,0,0,10746,0,10746,0,0,0,64,0,0,0,0,0
scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0
tlb metrics: 6,0,320,0,4147,0,1653,0,8739,0
###

```

/* end of cache performance metrics structure */

APPENDIX C. KERNEL MESSAGE DISPLAY

Kernel message listing

```
varuna@dyaus:~/safe_area > home/varuna/safe_area > telnet 192.168.98.4
```

```
Trying 192.168.98.4...
```

```
Connected to 192.168.98.4.
```

```
Escape character is '^]'.

```

```
BusyBox v0.60.0 (2010.03.04-08:34+0000) Built-in shell (msh)

```

```
Enter 'help' for a list of built-in commands.

```

```
#

```

```
#

```

```
# cat /proc/loadavg

```

```
0.05 0.01 0.00 2/17 114

```

```
#

```

```
#

```

```
# cat /proc/kmsg

```

```
<7>d_line_flush 62171, <7>
```

```
<7>d_line_flush 62172, <7>
```

```
<7>d_line_flush 62173, <7>
```

```
<7>d_line_flush 62174, <7>
```

```
<7>d_line_flush 62175, <7>
```

```
<7>d_line_flush 62176, <7>
```

```
<7>d_line_flush 62177, <7>
```

```
<7>d_line_flush 62178, <7>
```

```
<7>d_unroll_page 13523, <7>
```

```
<7>tlb_lflush_pg 1440, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7830, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_page 13524, <7>
```

```
<7>tlb_lflush_pg 1441, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7831, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_page 13525, <7>
```

```
<7>tlb_lflush_pg 1442, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7832, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_kseg0 9743, <7>d_unroll_way 9743, <7>
```

```
<7>i_unroll_kseg0 9743, <7>i_unroll_way 9743, <7>
```

```
<7>tlb_lflush_mm 289, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_page 13526, <7>
```

```
<7>d_unroll_page 13527, <7>
```

```
<7>tlb_lflush_pg 1443, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7833, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_kseg0 9744, <7>d_unroll_way 9744, <7>
```

```
<7>i_unroll_kseg0 9744, <7>i_unroll_way 9744, <7>
```

```
<7>d_unroll_page 13528, <7>
```

```
<7>d_unroll_page 13529, <7>
```

```
<7>tlb_lflush_pg 1444, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7834, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_kseg0 9745, <7>d_unroll_way 9745, <7>
```

```
<7>i_unroll_kseg0 9745, <7>i_unroll_way 9745, <7>
```

```
<7>d_unroll_page 13530, <7>
```

```
<7>d_unroll_page 13531, <7>
```

```
<7>tlb_lflush_pg 1445, <7> --->> update_TLB_metrics
```

```
<7>tlb_updt_mmu 7835, <7> --->> update_TLB_metrics
```

```
<7>d_unroll_kseg0 9746, <7>d_unroll_way 9746, <7>
```

```
<7>i_unroll_kseg0 9746, <7>i_unroll_way 9746, <7>
```

```
<7>d_unroll_page 13532, <7>
```

```
<7>d_unroll_page 13533, <7>
```

<7>tlb_lflush_pg 1446, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7836, <7> --->> update_TLB_metrics
<7>tlb_lflush_all 6, <7> --->> update_TLB_metrics
<7>d_unroll_page 13534, <7>
<7>tlb_lflush_pg 1447, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7837, <7> --->> update_TLB_metrics
<7>d_unroll_page 13535, <7>
<7>tlb_lflush_pg 1448, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7838, <7> --->> update_TLB_metrics
<7>d_unroll_page 13536, <7>
<7>d_unroll_page 13537, <7>
<7>tlb_lflush_pg 1449, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7839, <7> --->> update_TLB_metrics
<7>d_unroll_page 13538, <7>
<7>tlb_lflush_pg 1450, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7840, <7> --->> update_TLB_metrics
<7>d_unroll_page 13539, <7>
<7>d_unroll_page 13540, <7>
<7>tlb_lflush_pg 1451, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7841, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9747, <7>d_unroll_way 9747, <7>
<7>i_unroll_kseg0 9747, <7>i_unroll_way 9747, <7>
<7>d_unroll_page 13541, <7>
<7>d_unroll_page 13542, <7>
<7>tlb_lflush_pg 1452, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7842, <7> --->> update_TLB_metrics
<7>d_unroll_page 13543, <7>
<7>tlb_lflush_pg 1453, <7> --->> update_TLB_metrics
<7>tlb_updt_mmu 7843, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9748, <7>d_unroll_way 9748, <7>
<7>i_unroll_kseg0 9748, <7>i_unroll_way 9748, <7>
<7>d_unroll_kseg0 9749, <7>d_unroll_way 9749, <7>
<7>i_unroll_kseg0 9749, <7>i_unroll_way 9749, <7>
<7>tlb_lflush_rng 3839, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9750, <7>d_unroll_way 9750, <7>
<7>i_unroll_kseg0 9750, <7>i_unroll_way 9750, <7>
<7>tlb_lflush_rng 3840, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9751, <7>d_unroll_way 9751, <7>
<7>i_unroll_kseg0 9751, <7>i_unroll_way 9751, <7>
<7>tlb_lflush_rng 3841, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9752, <7>d_unroll_way 9752, <7>
<7>i_unroll_kseg0 9752, <7>i_unroll_way 9752, <7>
<7>tlb_lflush_rng 3842, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9753, <7>d_unroll_way 9753, <7>
<7>i_unroll_kseg0 9753, <7>i_unroll_way 9753, <7>
<7>tlb_lflush_rng 3843, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9754, <7>d_unroll_way 9754, <7>
<7>i_unroll_kseg0 9754, <7>i_unroll_way 9754, <7>
<7>tlb_lflush_rng 3844, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9755, <7>d_unroll_way 9755, <7>
<7>i_unroll_kseg0 9755, <7>i_unroll_way 9755, <7>
<7>tlb_lflush_rng 3845, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9756, <7>d_unroll_way 9756, <7>
<7>i_unroll_kseg0 9756, <7>i_unroll_way 9756, <7>
<7>tlb_lflush_rng 3846, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9757, <7>d_unroll_way 9757, <7>
<7>i_unroll_kseg0 9757, <7>i_unroll_way 9757, <7>
<7>tlb_lflush_rng 3847, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9758, <7>d_unroll_way 9758, <7>

<7>i_unroll_kseg0 9758, <7>i_unroll_way 9758, <7>
 <7>tlb_lflush_rng 3848, <7> --->> update_TLB_metrics
 <7>d_unroll_kseg0 9759, <7>d_unroll_way 9759, <7>
 <7>i_unroll_kseg0 9759, <7>i_unroll_way 9759, <7>
 <7>tlb_lflush_rng 3849, <7> --->> update_TLB_metrics
 <7>tlb_lflush_mm 290, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13544, <7>
 <7>d_unroll_page 13545, <7>
 <7>d_unroll_page 13546, <7>
 <7>tlb_updt_mmu 7844, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13547, <7>
 <7>d_unroll_page 13548, <7>
 <7>tlb_updt_mmu 7845, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13549, <7>
 <7>d_unroll_page 13550, <7>
 <7>d_unroll_kseg0 9760, <7>d_unroll_way 9760, <7>
 <7>i_unroll_kseg0 9760, <7>i_unroll_way 9760, <7>
 <7>tlb_updt_mmu 7846, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13551, <7>
 <7>d_unroll_page 13552, <7>
 <7>d_unroll_kseg0 9761, <7>d_unroll_way 9761, <7>
 <7>i_unroll_kseg0 9761, <7>i_unroll_way 9761, <7>
 <7>tlb_updt_mmu 7847, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13553, <7>
 <7>tlb_updt_mmu 7848, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13554, <7>
 <7>d_unroll_page 13555, <7>
 <7>d_unroll_kseg0 9762, <7>d_unroll_way 9762, <7>
 <7>i_unroll_kseg0 9762, <7>i_unroll_way 9762, <7>
 <7>tlb_updt_mmu 7849, <7> --->> update_TLB_metrics
 <7>d_unroll_kseg0 9763, <7>d_unroll_way 9763, <7>
 <7>i_unroll_kseg0 9763, <7>i_unroll_way 9763, <7>
 <7>tlb_lflush_rng 3850, <7> --->> update_TLB_metrics
 <7>d_unroll_kseg0 9764, <7>d_unroll_way 9764, <7>
 <7>i_unroll_kseg0 9764, <7>i_unroll_way 9764, <7>
 <7>tlb_lflush_rng 3851, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13556, <7>
 <7>d_unroll_page 13557, <7>
 <7>d_unroll_kseg0 9765, <7>d_unroll_way 9765, <7>
 <7>i_unroll_kseg0 9765, <7>i_unroll_way 9765, <7>
 <7>tlb_updt_mmu 7850, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13558, <7>
 <7>d_unroll_page 13559, <7>
 <7>d_unroll_kseg0 9766, <7>d_unroll_way 9766, <7>
 <7>i_unroll_kseg0 9766, <7>i_unroll_way 9766, <7>
 <7>tlb_updt_mmu 7851, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13560, <7>
 <7>d_unroll_page 13561, <7>
 <7>d_unroll_kseg0 9767, <7>d_unroll_way 9767, <7>
 <7>i_unroll_kseg0 9767, <7>i_unroll_way 9767, <7>
 <7>tlb_updt_mmu 7852, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13562, <7>
 <7>d_unroll_page 13563, <7>
 <7>tlb_updt_mmu 7853, <7> --->> update_TLB_metrics
 <7>d_unroll_page 13564, <7>
 <7>d_unroll_page 13565, <7>
 <7>d_unroll_kseg0 9768, <7>d_unroll_way 9768, <7>
 <7>i_unroll_kseg0 9768, <7>i_unroll_way 9768, <7>
 <7>tlb_updt_mmu 7854, <7> --->> update_TLB_metrics

<7>d_unroll_page 13566, <7>
<7>d_unroll_page 13567, <7>
<7>d_unroll_kseg0 9769, <7>d_unroll_way 9769, <7>
<7>i_unroll_kseg0 9769, <7>i_unroll_way 9769, <7>
<7>tlb_updt_mmu 7855, <7> --->> update_TLB_metrics
<7>d_unroll_page 13568, <7>
<7>d_unroll_page 13569, <7>
<7>d_unroll_kseg0 9770, <7>d_unroll_way 9770, <7>
<7>i_unroll_kseg0 9770, <7>i_unroll_way 9770, <7>
<7>tlb_updt_mmu 7856, <7> --->> update_TLB_metrics
<7>d_unroll_page 13570, <7>
<7>tlb_updt_mmu 7857, <7> --->> update_TLB_metrics
<7>d_unroll_page 13571, <7>
<7>tlb_updt_mmu 7858, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9771, <7>d_unroll_way 9771, <7>
<7>i_unroll_kseg0 9771, <7>i_unroll_way 9771, <7>
<7>tlb_lflush_rng 3852, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9772, <7>d_unroll_way 9772, <7>
<7>i_unroll_kseg0 9772, <7>i_unroll_way 9772, <7>
<7>tlb_lflush_rng 3853, <7> --->> update_TLB_metrics
<7>d_unroll_page 13572, <7>
<7>d_unroll_page 13573, <7>
<7>tlb_updt_mmu 7859, <7> --->> update_TLB_metrics
<7>d_unroll_kseg0 9773, <7>d_unroll_way 9773, <7>
<7>i_unroll_kseg0 9773, <7>i_unroll_way 9773, <7>
<7>tlb_lflush_rng 3854, <7> --->> update_TLB_metrics
<7>d_unroll_page 13574, <7>
<7>d_unroll_page 13575, <7>
<7>d_unroll_kseg0 9774, <7>d_unroll_way 9774, <7>
<7>i_unrollline_flush 62947, <7>
<7>d_line_flush 62948, <7>
<7>d_line_flush 62949, <7>
<7>d_line_flush 62950, <7>
<7>d_line_flush 62951, <7>
<7>d_line_flush 62952, <7>
<7>d_line_flush 62953, <7>
<7>d_line_flush 62954, <7>
<7>d_line_flush 62955, <7>
<7>d_line_flush 62956, <7>
<7>d_line_flush 62957, <7>
<7>d_line_flush 62958, <7>
<7>d_line_flush 62959, <7>
<7>d_line_flush 62960, <7>
<7>d_line_flush 62961, <7>
<7>d_line_flush 62962, <7>
<7>d_line_flush 62963, <7>
<7>d_line_flush 62964, <7>
<7>d_line_flush 62965, <7>
<7>d_line_flush 62966, <7>
<7>d_line_flush 62967, <7>
<7>d_line_flush 62968, <7>
<7>d_line_flush 62969, <7>
<7>d_line_flush 62970, <7>
<7>d_line_flush 62971, <7>
<7>d_line_flush 62972, <7>
<7>d_line_flush 62973, <7>
<7>d_line_flush 62974, <7>
<7>d_line_flush 62975, <7>
<7>d_line_flush 62976, <7>

```
#
# cat /proc/loadavg
0.01 0.01 0.00 2/17 122
#
```

APPENDIX D. LOAD FOR zmet APPLICATION

Load for zmet application

```
varuna@dyaus: /home/varuna/safe_area > telnet 192.168.98.4
Trying 192.168.98.4...
Connected to 192.168.98.4.
Escape character is '^'.
BusyBox v0.60.0 (2010.03.08-04:20+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.
```

```
#
# cat /proc/cpuinfo
system type      : Broadcom BCM5354 chip rev 3
processor        : 0
cpu model        : BCM3302 V2.9
BogoMIPS         : 237.56
wait instruction : no
microsecond timers : yes
tlb_entries      : 32
extra interrupt vector : no
hardware watchpoint : no
VCED exceptions  : not available
VCEI exceptions  : not available
unaligned_instructions : 0
dcache metrics: 0,0,0,0,41,0,0,0,9682,0,9682,0,47538,0,13457,0
icache metrics: 0,0,0,0,9682,0,9682,0,0,0,41,0,0,0,0,0
scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0
tlb metrics: 5,0,287,0,3827,0,1351,0,7780,0
```

```
#
# ps
PID Uid  Stat Command
  1 0    S    init noinitrd
  2 0    S    [keventd]
  3 0    S    [ksoftirqd_CPU0]
  4 0    S    [kswapd]
  5 0    S    [bdflush]
  6 0    S    [kupdated]
  7 0    S    [mtdblockd]
 77 0    S    httpd
 83 0    S    dnsmasq -h -n -c 0 -N -i br0 -r /tmp/resolv.conf -u root
 86 0    S    udhcpd /tmp/udhcpd.conf
 89 0    S    ddnsd &
 92 0    S    heartbeat
105 0    S    udhcpc -i eth0 -p /var/run/udhcpc0.pid -s /tmp/udhcpc
111 0    R    telnetd
112 0    S    /bin/sh
113 0    S    /bin/sh
115 0    S    zmet
116 0    R    ps
```

```
#
#
# cat /proc/cpuinfo
system type      : Broadcom BCM5354 chip rev 3
processor        : 0
cpu model        : BCM3302 V2.9
BogoMIPS         : 237.56
```

```

wait instruction      : no
microsecond timers   : yes
tlb_entries          : 32
extra interrupt vector : no
hardware watchpoint  : no
VCED exceptions      : not available
VCEI exceptions      : not available
unaligned_instructions : 0
dcache metrics: 0,0,0,0,41,0,0,0,10211,0,10211,0,66085,0,13974,0
icache metrics: 0,0,0,0,10197,0,10197,0,0,0,41,0,0,0,0,0
scache metrics: 0,0,0,0,0,0,0,0,0,0,0,0
tlb metrics: 6,0,295,0,3903,0,1618,0,8067,0
#
#
#
# cd /tp
/tp: bad directory
# ls -al
drwxr-xr-x  1 500  500   2969 Mar  8 2010 www
lrwxrwxrwx  1 500  500     7 Mar  8 2010 var -> tmp/var
drwxr-xr-x  1 500  500   29 Mar  8 2010 usr
drwxr-xr-x  1 0    0     0 Jan  1 2000 tmp
drwxr-xr-x  1 500  500   241 Mar  8 2010 sbin
dr-xr-xr-x 28 0    0     0 Jan  1 2000 proc
drwxrwxr-x  1 500  500     0 Mar  8 2010 mnt
drwxr-xr-x  1 500  500   87 Mar  8 2010 lib
drwxr-xr-x  1 500  500   51 Mar  8 2010 etc
drwxr-xr-x  1 0    0     0 Jan  1 1970 dev
drwxrwxr-x  1 500  500   105 Mar  8 2010 bin
# cd /tmp
# ls -al
drwxr-xr-x  1 0    0     0 Jan  1 2000 var
lrwxrwxrwx  1 0    0     8 Jan  1 1970 ldhclnt -> /sbin/rc
-rw-r--r--  1 0    0     2 Jan  1 1970 wlan_time
-rw-r--r--  1 0    0    40 Jan  1 1970 nas.lan.conf
-rw-r--r--  1 0    0     0 Jan  1 00:00 udhcpd.leases
-rw-r--r--  1 0    0   245 Jan  1 00:00 udhcpd.conf
-rw-r--r--  1 0    0   62 Jan  1 00:00 udhcpd_resrv.conf
-rw-r--r--  1 0    0     3 Jan  1 00:00 udhcpd.pid
drwxr-xr-x  1 0    0     0 Jan  1 00:00 ppp
lrwxrwxrwx  1 0    0    18 Jan  1 00:00 udhcpc -> /sbin/acos_ser
vice
-r--r--r--  1 0    0   938 Jan  1 00:00 lan_dev
-rw-r--r--  1 0    0     3 Jan  1 00:00 lan_time
-rw-r--r--  1 0    0     3 Jan  1 00:00 lan4_time
-rw-r--r--  1 0    0     2 Jan  1 00:00 udhcpc0.expires
-rw-r--r--  1 0    0     0 Jan  1 00:00 resolv.conf
drwxr-xr-x  1 500  500    72 Mar  8 2010 ..
drwxr-xr-x  1 0    0     0 Jan  1 2000 .
# # cd ..
# ls -al
drwxr-xr-x  1 500  500   2969 Mar  8 2010 www
lrwxrwxrwx  1 500  500     7 Mar  8 2010 var -> tmp/var
drwxr-xr-x  1 500  500   29 Mar  8 2010 usr
drwxr-xr-x  1 0    0     0 Jan  1 2000 tmp
drwxr-xr-x  1 500  500   241 Mar  8 2010 sbin
dr-xr-xr-x 28 0    0     0 Jan  1 2000 proc
drwxrwxr-x  1 500  500     0 Mar  8 2010 mnt
drwxr-xr-x  1 500  500   87 Mar  8 2010 lib

```

```

drwxr-xr-x 1 500 500 51 Mar 8 2010 etc
drwxr-xr-x 1 0 0 0 Jan 1 1970 dev
drwxrwxr-x 1 500 500 105 Mar 8 2010 bin
#
# cd /sbin
# ls -al
lrwxrwxrwx 1 500 500 2 Mar 8 2010 write -> rc
lrwxrwxrwx 1 500 500 12 Mar 8 2010 version -> acos_service
lrwxrwxrwx 1 500 500 12 Mar 8 2010 uptime -> acos_service
lrwxrwxrwx 1 500 500 2 Mar 8 2010 stats -> rc
lrwxrwxrwx 1 500 500 2 Mar 8 2010 reset_no_reboot -> bd
lrwxrwxrwx 1 500 500 14 Mar 8 2010 reboot -> ../bin/busybox
lrwxrwxrwx 1 500 500 12 Mar 8 2010 read_bd -> acos_service
-rwxr-xr-x 1 500 500 76628 Mar 8 2010 rc
-r-xr-xr-x 1 500 500 341436 Mar 8 2010 pppd
-rwxr-xr-x 1 500 500 13608 Mar 8 2010 ntpclient
lrwxrwxrwx 1 500 500 14 Mar 8 2010 insmod -> ../bin/busybox
lrwxrwxrwx 1 500 500 2 Mar 8 2010 init -> rc
lrwxrwxrwx 1 500 500 14 Mar 8 2010 ifconfig -> ../bin/busybox
lrwxrwxrwx 1 500 500 2 Mar 8 2010 hotplug -> rc
-rwxr-xr-x 1 500 500 5736 Mar 8 2010 gpio
lrwxrwxrwx 1 500 500 2 Mar 8 2010 erase -> rc
lrwxrwxrwx 1 500 500 2 Mar 8 2010 burnsn -> bd
lrwxrwxrwx 1 500 500 2 Mar 8 2010 burnrf -> bd
lrwxrwxrwx 1 500 500 2 Mar 8 2010 burnpin -> bd
lrwxrwxrwx 1 500 500 2 Mar 8 2010 burnthermac -> bd
lrwxrwxrwx 1 500 500 2 Mar 8 2010 burnboardid -> bd
-rwxr-xr-x 1 500 500 10056 Mar 8 2010 bd
-rwxr-xr-x 1 500 500 93844 Mar 8 2010 acos_service
lrwxrwxrwx 1 500 500 12 Mar 8 2010 acos_init -> acos_service
# version
Release version : Netgear Wireless Router WGR614v9
U12H09400/V1.2.6/18.0.17
Time : Jan 8 2009 12:04:20
CFE version : 2.6
#
#
# cat /proc/loadavg
0.10 0.04 0.01 1/18 124
#

```

APPENDIX E. OUTPUT OF MATRIC VALIDATION APPLICATION

```

Output of metric validation application zmet
varuna@dyaus: /home/varuna/safe_area > telnet 192.168.98.4
Trying 192.168.98.4...
Connected to 192.168.98.4.
Escape character is '^]'.
BusyBox v0.60.0 (2010.03.08-04:20+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.
#
# zmet
duration between metric value display = 5

d_line_flush 49691, d_unroll_page 13609, d_unroll_kseg0 9776, d_unroll_way 9776
i_blast 0, i_pline_flush 41, i_unroll_kseg0 9776, i_unroll_way 9776
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 1366, tlb_updt_mmu 7865, tlb_lflush_rng 3856

d_line_flush 555, d_unroll_page 4, d_unroll_kseg0 7, d_unroll_way 7

```

i_blast 0, i_pline_flush 0, i_unroll_kseg0 7, i_unroll_way 7
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 1, tlb_updt_mmu 3, tlb_lflush_rng 0

d_line_flush 10703, d_unroll_page 235, d_unroll_kseg0 272, d_unroll_way 272
i_blast 0, i_pline_flush 0, i_unroll_kseg0 258, i_unroll_way 258
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 236, tlb_updt_mmu 128, tlb_lflush_rng 29

d_line_flush 6021, d_unroll_page 143, d_unroll_kseg0 177, d_unroll_way 177
i_blast 0, i_pline_flush 0, i_unroll_kseg0 177, i_unroll_way 177
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 22, tlb_updt_mmu 83, tlb_lflush_rng 30

d_line_flush 5120, d_unroll_page 161, d_unroll_kseg0 94, d_unroll_way 94
i_blast 0, i_pline_flush 0, i_unroll_kseg0 94, i_unroll_way 94
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 389, tlb_updt_mmu 95, tlb_lflush_rng 29

d_line_flush 2834, d_unroll_page 155, d_unroll_kseg0 94, d_unroll_way 94
i_blast 0, i_pline_flush 0, i_unroll_kseg0 94, i_unroll_way 94
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 15, tlb_updt_mmu 89, tlb_lflush_rng 29

d_line_flush 2079, d_unroll_page 133, d_unroll_kseg0 85, d_unroll_way 85
i_blast 0, i_pline_flush 0, i_unroll_kseg0 85, i_unroll_way 85
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 18, tlb_updt_mmu 76, tlb_lflush_rng 29

d_line_flush 3433, d_unroll_page 157, d_unroll_kseg0 94, d_unroll_way 94
i_blast 0, i_pline_flush 0, i_unroll_kseg0 94, i_unroll_way 94
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 19, tlb_updt_mmu 91, tlb_lflush_rng 29

d_line_flush 3915, d_unroll_page 160, d_unroll_kseg0 94, d_unroll_way 94
i_blast 0, i_pline_flush 0, i_unroll_kseg0 94, i_unroll_way 94
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 19, tlb_updt_mmu 94, tlb_lflush_rng 29

d_line_flush 124, d_unroll_page 0, d_unroll_kseg0 6, d_unroll_way 6
i_blast 0, i_pline_flush 0, i_unroll_kseg0 6, i_unroll_way 6
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 3298, d_unroll_page 165, d_unroll_kseg0 114, d_unroll_way 114
i_blast 0, i_pline_flush 0, i_unroll_kseg0 114, i_unroll_way 114
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 22, tlb_updt_mmu 92, tlb_lflush_rng 48

d_line_flush 3491, d_unroll_page 134, d_unroll_kseg0 85, d_unroll_way 85
i_blast 0, i_pline_flush 0, i_unroll_kseg0 85, i_unroll_way 85
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 19, tlb_updt_mmu 77, tlb_lflush_rng 29

d_line_flush 1405, d_unroll_page 168, d_unroll_kseg0 102, d_unroll_way 102
i_blast 0, i_pline_flush 0, i_unroll_kseg0 88, i_unroll_way 88
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 19, tlb_updt_mmu 82, tlb_lflush_rng 29

d_line_flush 371, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 6, d_unroll_way 6
i_blast 0, i_pline_flush 0, i_unroll_kseg0 6, i_unroll_way 6
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 4, d_unroll_way 4
i_blast 0, i_pline_flush 0, i_unroll_kseg0 4, i_unroll_way 4
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 123, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 3273, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 197, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 568, d_unroll_page 4, d_unroll_kseg0 8, d_unroll_way 8
i_blast 0, i_pline_flush 0, i_unroll_kseg0 8, i_unroll_way 8
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 1, tlb_updt_mmu 3, tlb_lflush_rng 0

d_line_flush 225, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

d_line_flush 1905, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0

```
d_line_flush 2115, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0
```

```
d_line_flush 656, d_unroll_page 134, d_unroll_kseg0 85, d_unroll_way 85
i_blast 0, i_pline_flush 0, i_unroll_kseg0 85, i_unroll_way 85
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 19, tlb_updt_mmu 77, tlb_lflush_rng 29
```

```
d_line_flush 3586, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0
```

```
d_line_flush 652, d_unroll_page 163, d_unroll_kseg0 111, d_unroll_way 111
i_blast 0, i_pline_flush 0, i_unroll_kseg0 111, i_unroll_way 111
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 21, tlb_updt_mmu 91, tlb_lflush_rng 46
```

```
d_line_flush 226, d_unroll_page 0, d_unroll_kseg0 5, d_unroll_way 5
i_blast 0, i_pline_flush 0, i_unroll_kseg0 5, i_unroll_way 5
s_line_flush 0, s_unroll_page 0, s_unroll_kseg0 0, s_unroll_page 0
tlb_lflush_pg 0, tlb_updt_mmu 0, tlb_lflush_rng 0
```

```
#
```

APPENDIX F.1. HOW TO INSERT MODULE in WGR614v9

How to insert your own module for WGR614 using the Netgear source code:

Get into the $\$(TOPDIR)/arch$ directory, and further into the kernel directory of the associated architecture of the Embedded System that you are developing the driver for. For example, I am working on the MIPS architecture; hence, I will be getting into $\$(TOPDIR)/arch/mips/kernel$ directory.

Create the source code file that will contain the driver functions.

Modify the $\$(TOPDIR)/arch/mips/kernel/Makefile$ and append the object file name of the device driver that into the listed variables:

```
export-objs
```

```
obj-y
```

Locate the *.config* file in the $\$(TOPDIR)$ of the code base.

Identify the configuration variable, check if it has been set to *y*. As an example to add a */proc* file for the driver, search for *CONFIG_PROC_FS* in the *.config*.

Search for the occurrence of the pattern *CONFIG_PROC_FS* in the entire code base using *cscope* tool.

A listing of code under the $\$(TOPDIR)/kernel$ directory will be displayed, get deep into the code, and figure out where to plug in the function call that will create the required */proc* entries, and when it should be called.

For example, a */proc fs* entry cannot be created prior to */proc* being mounted by the *init* process.

Call the device driver functions from the identified file in the directory $\$(TOPDIR)/kernel$ from the section that has the *#ifdef* for *CONFIG_PROC_FS*.

Build from $\$(TOPDIR)$ after cleaning the code base of the earlier built object files.

Check for the availability of the *printk* messages listed in the kernel image file $\$(TOPDIR)/vmlinux$ for the "pattern" used in the LKM:

Check for the availability of the driver functions, and the introduced data structures listed in the file $\$(TOPDIR)/System.map$ using the *grep* command. This file also gives you the address of the functions.

APPENDIX F.2. HOW-TO-FTP DATA INTO WGR614v9

1. Edit */etc/vsftpd/vsftpd.conf* file and ensure the following is visible:

```
local_enable=YES
```

```
write_enable=YES
```

```
local_umask=022
```

```
dirmessage_enable=YES
```

```
xferlog_enable=YES
```

```

connect_from_port_20=YES
xferlog_file=/var/log/vsftpd.log
xferlog_std_format=YES
idle_session_timeout=600
data_connection_timeout=120
ftpd_banner=Welcome to shastry FTP service.
ls_recurse_enable=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES

```

2. Enable *vsftpd* on the development system using the command:

```
service vsftpd restart
```
3. Login into the WGR614v9 using *telnet* and change into directory to */tmp*.
4. On the router, execute the following command:

```
/usr/sbin/ftpc -d dev_ip -u username -p userpasswd -f localfile -s sourcefile
```

Note: *localfile* is on the router, *sourcefile* is on the development system, *username* & *userpasswd* is on the development system used for getting the ftp data, *dev_ip* is the IP address of the development system. The assumption in here is that the *sourcefile* is available in the home directory of the *username*.

APPENDIX F.3. HOW-TO-TELNET INTO THE ROUTER

- 1: Install the tool *telnetenable-0.3* available in the internet for the NETGEAR routers.
- 2: In the shell prompt, execute the command to get the MAC address of the router:

```
arp -a
```

- 3: Execute the following command to enable the telnet permission on the router:

```
telnetenable r_ip r_mac Gearguy Geardog
```

Note: *r_ip* is the IP address, while *r_mac* is the MAC address of the router.

Example:

```
telnetenable 192.168.98.4 0024B29280F0 Gearguy Geardog
```

- 4: Telnet into the router using the command:

```
telnet r_ip
```

Note: *r_ip* is the router IP address.

APPENDIX F.4. HOW-TO-GET ADDITIONAL SPACE ON THE ROUTER FLASH

Edit *../src/router/busybox/Config.h* and comment the following:

```
#define BB_CP
```

```
#define BB_PING
```

```
#define BB_FEATURE_FANCY_PING
```

```
#define BB_RMDIR
```

Note: Prior to modifying the indicated files, please check if you require the services provided by the file is required in your setup.

Remove the code between the html tags `<script>` and `</script>` from the following files in the directory *../project/acos/www/html*:

```
BKS_keyword.htm
```

```
BKS_service.htm
```

```
DNS_ddns.htm
```

```
FW_forward.htm
```

```
FW_forward_service.htm
```

```
FW_pt.htm
```

```
FW_pt_service.htm
```

```
FW_schedule.htm
```

```
FW_remote.htm
```

```
UPNP_upnp.htm
```

```
WAN_wan.htm
```

```
WIZ_sel.htm
```

```
WLG_wds.htm
```

```
WLG_wireless1.htm
```

```
WLG_wireless2.htm
```

WLG_wireless3.htm

Note: Prior to modifying the indicated files, please check if you require the services provided by the file is required in your setup.

Trimming of the above web pages and the *Config.h* file yields about 80KB of additional flash space on the router.

Build the router section of the code under *.../src/router* using the following commands:

make clean

make

make install