The research leading to this tool is supported by the European FP7 project "Harnessing Performance Variability" (FP7-ICT-2013-10-612069, HARPA).

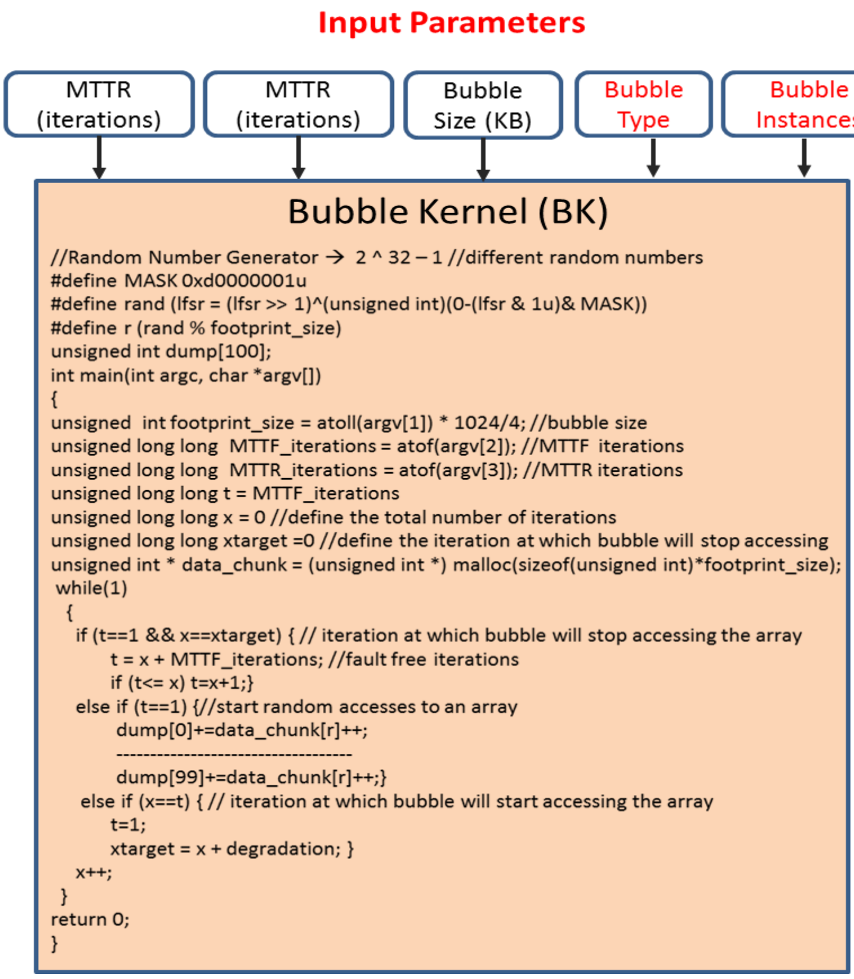
Degradation Bubble

The aim of this tool is to cause specific performance degradation to the host application by the collocation with “Bubbles”, which are stress test programs that contend on shared resources, such as last level cache, memory bandwidth etc. Using “Bubbles” and collocation with an application of interest, the main goal of this tool is to emulate the performance degradation due to faults on real HW. The performance degradation due to periodic appearance of faults with a given mean time to repair (MTTR) and mean time to failure is emulated by co-running the host application with “Bubbles” which are tuned to periodically stress the shared resources depending on the MTTR & MTTF values.

Basic Operation

The basic operation of this tool is based on “Bubbles” which are test programs used to stress shared resources, such as last level cache, by allocating and performing random memory accesses to an array, as defined in the Bubble Kernel module (BK), shown in Figure 1. The random array addresses to access, are produced by a fast random number generator using linear shift registers (LFSR), in such a way that it prevents accessing the same address twice before accessing all the allocated array entries once. The duration when “Bubble” is performing memory accesses is defined in iterations, as shown in the main loop of the Bubble Kernel module. The number of iterations “Bubble” will perform memory accesses, is given as input and is defined as MTTR iterations. Another input is the number of iterations bubble will not perform memory accesses, defined as MTTF iterations. The size of memory that bubble will access is defined as bubble size as shown in Figure 1.

Bubble Kernel Characterization (BKC) module can be used on top of Bubble Kernel to convert the MTTR and MTTF from milliseconds to the number of iterations where “Bubble” of a given size is stressing or not stressing the memory subsystem while co-running a given Host Application, as presented in Figure 2. In more detail bubble characterization is explained in section Structure (Bubble_Kernel_Characterization).



```
//Random Number Generator → 2 ^ 32 – 1 //different random numbers
#define MASK 0xd0000001u
#define rand (lfsr = (lfsr >> 1)^(unsigned int)(0-(lfsr & 1u)& MASK))
#define r (rand % footprint_size)
unsigned int dump[100];
int main(int argc, char *argv[])
{
unsigned  int footprint_size = atoll(argv[1]) * 1024/4; //bubble size
unsigned long long  MTTF_iterations = atof(argv[2]); //MTTF  iterations
unsigned long long  MTTR_iterations = atof(argv[3]); //MTTR iterations
unsigned long long t = MTTF_iterations
unsigned long long x = 0 //define the total number of iterations
unsigned long long xtarget =0 //define the iteration at which bubble will stop accessing
unsigned int * data_chunk = (unsigned int *) malloc(sizeof(unsigned int)*footprint_size);
 while(1)
  {
   if (t==1 && x==xtarget) { // iteration at which bubble will stop accessing the array
        t = x + MTTF_iterations; //fault free iterations
        if (t<= x) t=x+1;}
   else if (t==1) {//start random accesses to an array
         dump[0]+=data_chunk[r]++;
         -----------------------------------
         dump[99]+=data_chunk[r]++;}
    else if (x==t) { // iteration at which bubble will start accessing the array
        t=1;
        xtarget = x + degradation; }
   x++;
  }
return 0;
}
```

Figure 1: Bubble Kernel

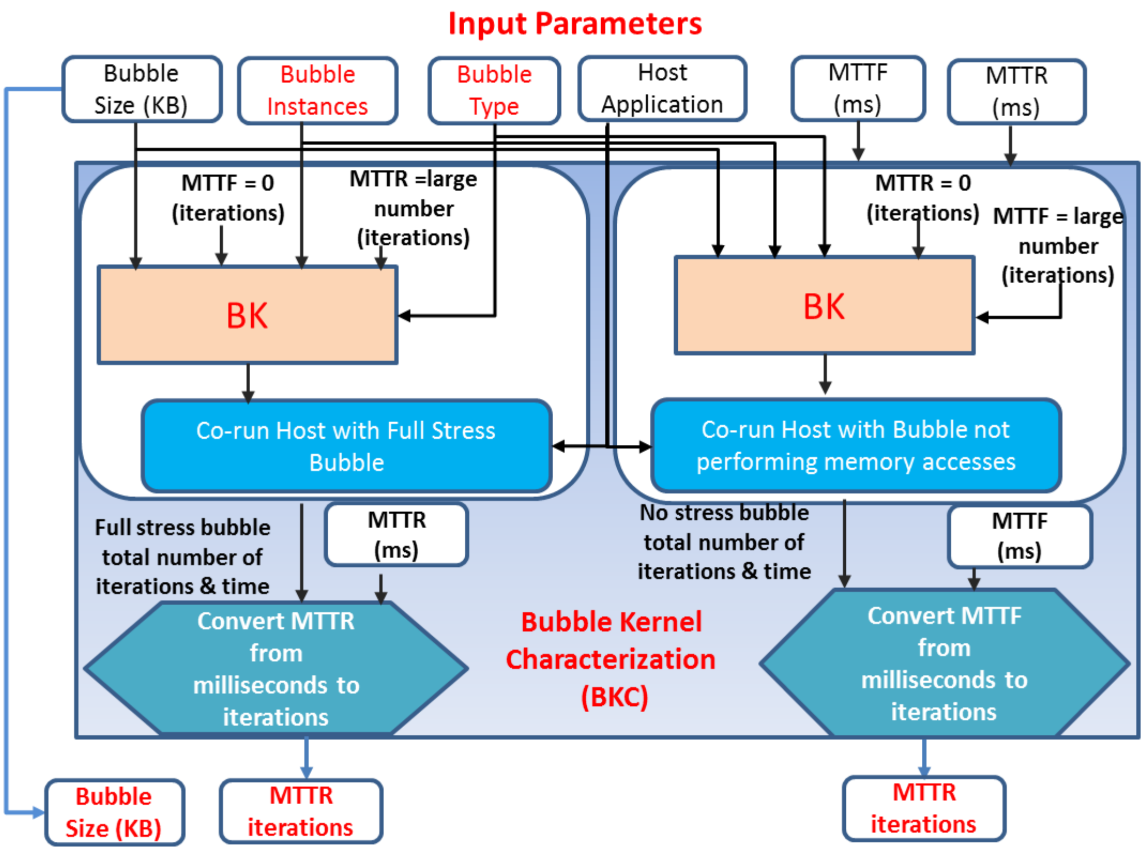

Input Parameters
Bubble Size (KB)
Bubble Instances
Bubble Type
Host Application
MTTF (ms)
MTTR (ms)
MTTF = 0 (iterations)
MTTR =large number (iterations)
MTTR = 0 (iterations)
MTTF = large number (iterations)
BK
BK
Co-run Host with Full Stress Bubble
Co-run Host with Bubble not performing memory accesses
Full stress bubble total number of iterations & time
MTTR (ms)
No stress bubble total number of iterations & time
MTTF (ms)
Convert MTTR from milliseconds to iterations
Bubble Kernel Characterization (BKC)
Convert MTTF from milliseconds to iterations
Bubble Size (KB)
MTTR iterations
MTTR iterations

Figure 2: Bubble Kernel Characterization

Download & Installation

Requirements:
-bash version 4.1.2(1)-release, awk, gnuplot, perf version 2.6.32-358.6.1.el6.x86_64.debug, linux taskset to pin jobs to a specific core

Note: gnuplot is used only in the validation phase (Section Wrappers)

Installation

The following steps describe how to install bubble tool and use the different scripts from any directory.

a. Create a folder bin to your home directory in case that it doesn't exist
   1. cd $HOME
   2. mkdir bin
b. Download bubble_tool_V1.tar.gz to $HOME/bin and decompress it using
   1. tar -zxvf bubble_tool_V1.tar.gz
c. To install the tool first identify the start-up script on your shell doing ls -lart. It should be either .bash_profile or .bashrc
d. In case that the start-up script doesn't exists in your local $HOME directory create a .bash_profile or .bashrc file
e. Once you have identified or created the start-up script, add the following line in your script
   1. export PATH=$PATH:$HOME/bin/bubble_tool_V1/bubble_kernel
   2. export PATH=$PATH:$HOME/bin/bubble_tool_V1/bubble_kernel_characterization
   3. export PATH=$PATH:$HOME/bin/bubble_tool_V1/wrappers
   4. export wrapper_config_file_dir="$HOME/bin/bubble_tool_V1/wrappers/wrapper_input_configuration_file"
f. After adding the source files directories in your start-up script then log out and log back in to use the script from any directory.

A sample of .bash_profile or .bashrc can be found in:

> $HOME/bin/bubble_tool_V1/sample_bash_profile

Use

To use the bubble tool first you need to compile the bubbleKernel.c. In the directory bubble_tool_V1/bubble_kernel you can find the Makefile. Execute "Make all" to create the bubbleKernel executable.

To use the rest of the bash scripts, you need once to change the permission to executable of different scripts available in the tool. In the directory bubble_tool_V1 you can find the script change_scripts_mode.sh and by executing it you are making all the available scripts in the tool executables.

```
> cd $HOME/bin/bubble_tool_V1

$HOME/bin/bubble_tool_V1 > chmod +x change_scripts_mode.sh

$HOME/bin/bubble_tool_V1 > bash change_scripts_mode.sh
```

After performing the previous steps the user can execute bubbleKernel, bubble_kernel_characterization.sh and wrapper_periodic_stress from any directory. More details on the input parameters needed to call each executable are presented in section Structure.

An important constraint to use the bubble_tool is the isolation. You should run the tool in isolation without running other processes. This is important to reduce the variations induced by the other processes. Also variations can be induced by OS and can affect the accuracy of the results the tool will produce. This can be reduced by performing multiple confidence runs. Background processes or other process such as top running at the same time the user is running the tool, can add some deviations on the results.

Structure

The tool is written in bash and C and is organized in the following subdirectories

+ bubble_kernel
+ bubble_kernel_characterizations
+ wrappers

A. Bubble Kernel

+ bubble_kernel/bubbleKernel.c

The bubble_kernel directory consists of the bubbleKernel.c program written in C. bubbleKernel is a stress program which allocates an array and consist of a main loop where it is performing random array accesses, as shown in Figure 1 and described in Basic Operation Section.

A.1 Input parameters

bubbleKernel can be executed alone giving three arguments. It will be extended for more parameters as discussed earlier.

1. bubble size in KB
2. MTTR iterations
3. MTTF iterations

An example on how to execute the Kernel alone is shown below

```
> bubbleKernel 12288 3000 50
```

Where:
bubble size = 12288KB
MTTF iterations = 3000
MTTR iterations = 50
Be careful the bubbleKernel does not terminate. To terminate the kernel, if you decide to run it alone, you need to send termination signal such CTRL-C etc.

B. Bubble Kernel Characterization

+ bubble_kernel_characterization/bubble_kernel_characterization.sh

The bubble_kernel_characterization directory consists of bubble_kernel_characterization.sh written in bash.

B.1. Functionality: Convert the MTTF & MTTR from milliseconds to MTTR & MTTF iterations

The main goal of this module is to convert the MTTF and MTTR from milliseconds to the corresponding number of iterations where bubble co-running with the host application is stressing and not stressing the shared resources. The input parameters of BKC (bubble kernel characterization) are shown in Table 1. The user should pass to the script the MTTF, MTTR in milliseconds and the Host Application. The rest of the inputs are optional and in case that they are not defined they are initialized with default values. To convert the MTTR from milliseconds to iterations, the host application is co-running first with a bubble of a given size which continuously is stressing memory, calling the bubble kernel passing as parameter MTTF iterations = 0 and MTTR iteration a large integer number. From this step we have the total time and the total number of iterations that bubble kernel passed stressing memory, while co-running the host application, and then we calculate the time in milliseconds per iteration. As previously mentioned the MTTR corresponds the time that bubble should perform memory accesses and having the time of one iteration while bubble is performing memory accesses, we can find the corresponding number of MTTR iterations (MTTR iterations = MTTF (ms)/time per one iteration while stressing(ms)). To convert the MTTF from milliseconds to the number of iterations that bubble should not perform memory accesses , first we co-run the host application with a bubble (BK) that is not accessing the array, passing as parameters the defined bubble size , MTTR iterations = 0 and MTTF iterations a large integer number. In this step we conduct the total time and the total number of iterations that bubble is not stressing memory, during the host application runtime, and the time in milliseconds needed per iteration. To convert the MTTF from milliseconds to iterations we divide the MTTF ms by the time needed for one iteration (MTTF iterations = MTTF (ms)/time per iteration no stress bubble).
Another important step in the bubble kernel characterization module is to check if the Host Application required performance degradation, specified by the MTTR (ms) and MTTF (ms), can be reached by the co-location with the bubble of a given size. This can be done by profiling the host application while running alone and co-running it with full stress bubble, using performance counters monitoring tool (perf), to keep track on the IPC on both cases. The slowdown on the host application IPC caused by bubble gives the maximum

performance degradation that can be reached to the host application from the co-location with bubble.

B.2. Input Parameters

The user can run the bubble_kernel_characterization.sh from any directory by using the input defined in Table 1. The first three parameter values MTTF (ms), MTTR (ms) and Host Application should be defined to be able to use the bubble_kernel_characterization script. The rest of the inputs are optional, in case that they are not defined default values are given to them as shown in Table 1. The module will be extended to take as input the number of bubble instances the user want to co-run with the host application and the type of bubble (streaming vs random). In this version of the tool the bubble_kernel_characterization takes only the first seven inputs defined in Table 1.

An example on how to run the bubble kernel characterization module giving as parameter MTTF = 3000 (ms), MTTR = 50(ms) and host_application = “test_program input_file” is shown below

```
> bubble_kernel_characterization.sh 3000 50 “test_program input”
```

Bubble Kernel Characterization Input Parameters

1. MTTF	Mean time to failure in milliseconds. Should be defined
2. MTTR	Mean time to repair in milliseconds. Should be defined
3. Host Application	Full command to run the host application. Should be defined
4. Bubble Size	Bubble size in KB. If not defined the default value is the LLC cache of the underlying machine
5. Host Core	Define the core to run the host application. If not defined the default value is 2
6. Bubble Core	Define the core to run the bubble kernel. If not defined the default value is 0
7. Confidence runs	Define the number of confidence runs. If not defined the default value is 0
8. Bubble instances	Define the number of bubbles to co-run with the host application *
9. Bubble type	Define the type bubble to co-run with the host application (random, streaming, both) *

Table 1: bubble_kernel_characterization input values

* Not Implemented in this version

B.3. Output

The output of the bubble_kernel_characterization.sh is saved in the file BKC.HostApplicationName.bubbleSize.out. The file is saved in the directory bubble_kernel_characterization_outdir created in the current directory from where you run the script. An example of the output file is shown below:

```
Host_alone_avgIPC:          1.12
Host_alone_avgIPC_stdv:     0
MTTR_iterations:            193663.2212990128207618928 8
MTTF_iterations:            1213122155.462993946266525516 29
BubbleSize:                  12288
Host_Application:
"/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2006/403.gcc/exe/gcc_base.i3
86-m32-gcc42-nn
/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2006/403.gcc/data/ref/input/1
66.i  -o
/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2006/403.gcc/data/ref/output/
166.s "
```

C. Wrappers

+ wrappers/ wrapper_periodic_stress.sh
+ wrappers/ wrapper_input_configuration_file/default_config_file

The wrappers directory +wrappers consists of the wrapper_periodic_stress.sh script and +wrapper_input_configuration_file directory where the default input parameters are passed to the wrapper through the default_config_file. Be careful, do NOT REMOVE the default_config_files, the wrapper is loading first the default values from it and the user can change them by different options as presented in Table 2

C.1. Functionality: Cause a Specific Performance Degradation to the Host Application defined by MTTF and MTTR

The main goal of this use case is to emulate the performance degradation that an application can experience due to periodic appearance of faults. In presence of faults, with a given MTTF and MTTR, during the host application run time, it is expected to suffer a specific performance degradation, defined by MTTR/(MTTF + MTTR). During host application run time, there are two phases, the faulty free phase, where the host application is not accessing any faulty unit, defined by the mean time to failure (MTTF), and it is running without experiencing any performance degradation. The next phase, referred as the faulty phase, where the host application is accessing faulty units, defined by the mean time to repair (MTTR), where extra overhead is added to repair from faults. This behaviour is emulated by co-running the Host Application with a bubble (BK) which is stressing the memory subsystem, contending periodically on the shared last level cache, based on the MTTF and MTTR values. The phase where bubble is performing memory accesses emulates the faulty phase and depends on the MTTR value. The phase where bubble is not performing memory accesses emulates the faulty free phase and is defined by the MTTF value. Both values are converted into iterations by the Bubble Kernel Characterization (BKC) module as it is shown in Figure 3. Than the host application is co-running with the bubble (BK) which takes as input the MTTR & MTTF number of iterations produced by the BKC module and the bubble size. As have been mentioned before, the bubble kernel will be extended to take as input the bubble type and number of instances, shown also in Figure 3. The required performance degradation of the host application is reached by the co-location with bubble by the sharing of the last level cache and the periodic stress that bubble performs on this resource. The performance degradation that can be caused to the host application is specified as the slowdown on the IPC of the host application while co-running bubble compared to that of running alone (output from BKC module). At the end of this procedure the required performance degradation is reached to the host application by co-location with bubble, for more details refer to the subsection C.3. Output.

C.2. Input Parameters

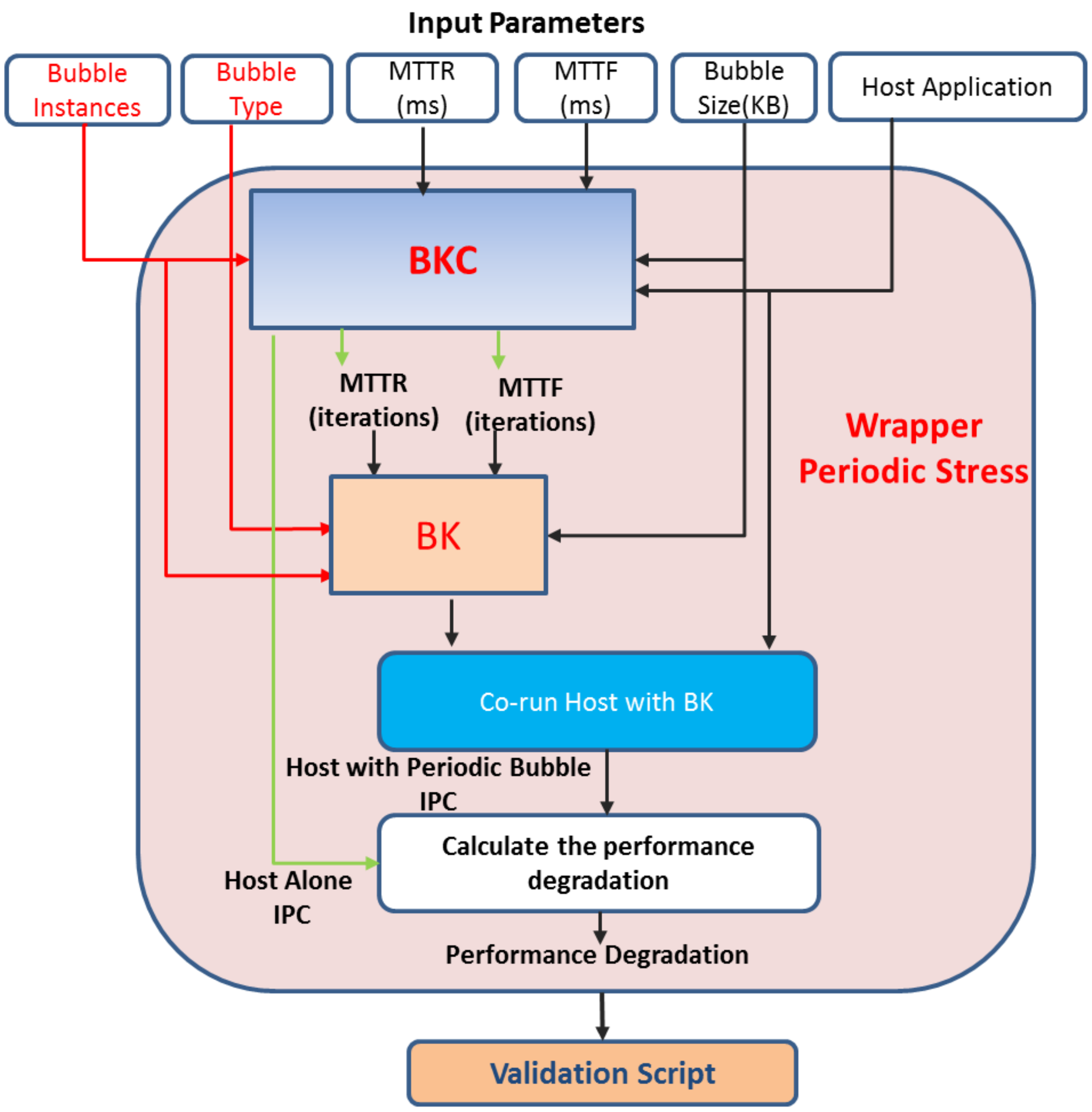


Figure 3: Basic Functionality of wrapper_periodic_stress

The user can run the wrapper_periodic_stress.sh by giving different parameters through different options. By default the input parameter values are the ones defined in the default_config_file. The format of the file and different input options are presented below with # presenting comments:

Wrapper Periodic Stress: default_config_file

The wrapper can be executed by giving different options through command line arguments as presented in the help message below. To show the help message run

```
>wrapper_periodic_stress.sh
>wrapper_periodic_stress.sh –h or –help
```

To execute the wrapper you need to pass at least one option otherwise the help message will be show. An example on how to run

```
>wrapper_periodic_stress.sh –MTTF 3000 –MTTR 50
```

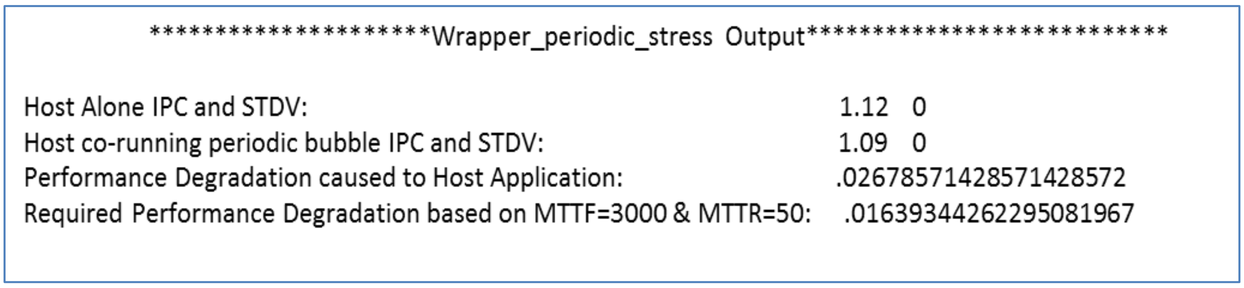
-h | --help	List the available command line arguments of the wrapper_periodic_stress
-host_application	Follow the full command used to execute the Host Application Example -host_application "[full path to executable] [options] [inputs] [etc]" Host Application full command should be given in " "
-v	Selelct -v option in case that user want to see a detailed report which includes different statistics
-host_core	Define the Processor Number to run the Host Application Example -host_core [number] The number should be an existing processor.
-bubble_core	Define the Processor Number to run the Bubble Example -bubble_core [number] The number should be an existing processor.
-confidence_runs	Define the total number of confidence runs Example -confidence_runs [number]. Select an integer number.

-sampling	Define the sampling interval in seconds to keep perf statistics while co-running Host Application with periodic bubble This is used only for validation purpose in the script produced at the end. (THIS PART IS MISSING FROM THE TOOL AT THIS VERSION)
-bubbleSize	Define the size of memory that Bubble will allocate and pressure. Example -bubbleSize [size]. The [size] should be an integer number in KB.
-MTTF	Select the MTTF of the periodic bubble in ms. MTTF corresponds to the time in ms which is converted to iterations that bubble will not stress shared resources
-MTTR	Select the MTTR of the periodic bubble in ms. MTTR corresponds to the time in ms which is converted to iterations where bubble will stress shared resources
-config_file	Define the full path of the configuration file you want to use Example -config_file [/home/lndreu/configuration_file] The configuration file should have the format of the default_config_file as shown in Bubble Tool wrapper_input_configuration_file directory The default_config_file are the default values of different parameters used in Bubble Tool Don't remove it. Change the default values through command line options or by selecting another configuration file.

Table 2: wrapper_periodic_stress input options

C.3. Output

The output of the wrapper_periodic_stress.sh is presented in the screen. If the user selected to run the wrapper with the option –v (verbose) than a report is produced and is saved in the file HostApplicationName.bubbleSize.report. The file is saved in the directory wrapper_periodic_stress_outdir created in the current directory from where you run the script. Examples of the results presented in the screen and in the report file are presented below:



```
                    ********************Wrapper_periodic_stress  Output**************************

Host Alone IPC and STDV:                                                1.12   0
Host co-running periodic bubble IPC and STDV:                           1.09   0
Performance Degradation caused to Host Application:                    .02678571428571428572
Required  Performance Degradation based on MTTF=3000 & MTTR=50:         .01639344262295081967
```

Wrapper_periodic_stress output in the screen

```
****************Input Parameters selected to run wrapper_periodic_stress*********
Host Application:
/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2
006/403.gcc/exe/gcc_base.i386-m32-gcc42-nn
/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2
006/403.gcc/data/ref/input/166.i  -o
/home/lndreu01/MemoryHierarchy/Threads_2/lfsr_bench/SPEC2006INST/benchspec/CPU2
006/403.gcc/data/ref/output/166.s
MTTF(ms):                                               3000
MTTR(ms):                                                50
Bubble Size selected to co-run host application:         12288(KB)
Core number to pin Host Application                       6
Core number to pin Bubble                                 0
Total number of confidence runs                           1
Verbose Mode Enabled

************Wrapper called Bubble Kernel Characterization to convert MTTF & MTTR
from millisecons to iterations******

BKC Input parameters:              MTTR = 50(ms)   MTTF = 3000
BKC Output:          MTTR iterations=223997.62456532145207460225(iterations) and MTTF
iterations=3562299484.88552369645531959950(iterations)
Host Application running alone average IPC and STDV:                        1.12    0
Host Application co-running periodic stress bubbble average IPC and STDV:    1.09    0

********************Wrapper_periodic_stress Output**************************
Host Application Required performance degradation:MTTR/(MTTF+MTTR)      .016393
Host Application Performance Degradation caused by periodic Bubble:          .026785
```

Wrapper periodic stress output: gcc_base.i386-m32-gcc42-nn.12288.report