/***
** Notes on SPEC CPU2017
**
** Intended audience: Those who would like to learn more about
** measuring performance of modern computer systems using standardized benchmarks.
**
** Used in: CPE 631 Advanced Computer Systems and Architectures
** CPE 619 Modeling and Analysis of Computer and Communication Systems
**
** v0.1 (Spring 2018)
**
** @Aleksandar Milenkovic, milenkovic@computer.org
***/

Using SPEC CPU2017 in Blackhawk

1 About SPEC CPU2017

SPEC CPU2017 is a benchmark suite for evaluating and comparing performance of compute-intensive workloads on different computer systems. It captures performance of (a) Processor (CPU processor chips), (b) Memory hierarchy (including caches and main memory), and (c) Compilers (C, C++ and Fortran compilers, including optimizers). It does not stress other aspects of computer performance, including networking, I/O, graphics, Java libraries. SPEC CPU2017 is the latest incarnation of the SPEC CPU benchmark suites that include now obsolete versions CPU92, CPU95, CPU2000, and CPU2006.

CPU2017 contains 43 benchmarks organized into four suites: (a) SPECspeed 2017 Integer, (b) SPECspeed 2017 Floating Point, (c) SPECrate 2017 Integer, and (d) SPECrate 2017 Floating Point. These benchmarks are drawn from actual end-user applications. They are highly portable, and performance runs have to comply with set of rules to ensure comparability and repeatability of measurements. CPU 2017 include the following:

- Source code od benchmarks
- Data sets
- A tool set for compiling, running, validating, and reporting
- Pre-compiled tools for a variety of operating systems
- Source code for the SPEC CPU2017 tools
- Documentation: https://www.spec.org/cpu2017/Docs/index.html
- Run and reporting rules.

A *suite* is a set of benchmarks that are run as a group to produce one of the overall metrics. SPEC CPU2017 includes four suites that focus on different types of compute intensive performance:

Short Tag	Suite	Contents	Metrics	How many copies? What do Higher Scores Mean?
intspeed	SPECspeed 2017 Integer	10 integer benchmarks	SPECspeed2017_int_base SPECspeed2017_int_peak	SPECspeed suites always run one copy of each benchmark. Higher scores indicate that less time is needed.
fpspeed	SPECspeed 2017 Floating Point	10 floating point benchmarks	SPECspeed2017_fp_base SPECspeed2017_fp_peak	
intrate	SPECrate 2017 Integer	10 integer benchmarks	SPECrate2017_int_base SPECrate2017_int_peak	SPECrate suites run multiple concurrent copies of each benchmark. The tester selects how many. Higher scores indicate more *throughput* (work per unit of time).
fprate	SPECrate 2017 Floating Point	13 floating point benchmarks	SPECrate2017_fp_base SPECrate2017_fp_peak	

2 SPEC CPU2017 on Blackhawk

SPEC installation directory is located at /apps/arch/cpu2017.
It includes subdirectories with documentation (Doc), Spec Utilities (bin), and benchmarks (benchspec).

Please note that only system admin can modify this directory.
However, all users can run spec utilities, namely runspec, that allows for building and running benchmarks (providing configuration file specifies the SPEC output directory be the one where user can write).

To do so, a user should go to /apps/arch/cpu2017 and perform the following command:

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ pwd
/apps/arch/cpu2017
-bash-4.1$ source shrc
-bash-4.1$ ulimit -s unlimited
-bash-4.1$ export OMP_STACKSIZE=192M

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

Benchmarks are in the directory /apps/arch/cpu2017/benchspec/CPU/.

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ pwd
/apps/arch/cpu2017/benchspec/CPU

500.perlbench_r  525.x264_r       602.gcc_s        638.imagick_s
502.gcc_r        526.blender_r    603.bwaves_s     641.leela_s
503.bwaves_r     527.cam4_r       605.mcf_s        644.nab_s
505.mcf_r        531.deepsjeng_r  607.cactuBSSN_s  648.exchange2_s
507.cactuBSSN_r  538.imagick_r    619.lbm_s        649.fotonik3d_s
508.namd_r       541.leela_r      620.omnetpp_s    654.roms_s
510.parest_r     544.nab_r        621.wrf_s        657.xz_s
511.povray_r     548.exchange2_r  623.xalancbmk_s  996.specrand_fs
519.lbm_r        549.fotonik3d_r  625.x264_s       997.specrand_fr
520.omnetpp_r    554.roms_r       627.cam4_s       998.specrand_is
521.wrf_r        557.xz_r         628.pop2_s       999.specrand_ir
523.xalancbmk_r  600.perlbench_s  631.deepsjeng_s  CPU.bset
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

3 Running SPEC benchmarks using runcpu

For reporting performance benchmarks should be compiled and run using SPEC runcpu utility. More details about runcpu can be found at https://www.spec.org/cpu2017/Docs/runcpu.html#define.
If you want to have reportable results of CPU2017, you need to understand the runcpu utility, configuration files that control compilation and execution, and redirection of output files generated in runs.

Find a Config File: To use runcpu, you need a config file - a file that defines how to build, run, and report on the SPEC CPU benchmarks in a particular environment, including any needed PORTABILITY flags for your compilers.

1. There are examples on your installed copy of SPEC CPU2017. Look for an Example that matches your compiler, operating system, and hardware, in directory:
   $SPEC/config/Example* (Unix) or
   %SPEC%\config\Example* (Windows)
2. Or, look for results for a system similar to yours at www.spec.org/cpu2017/results. Click the 'config' link.
3. Or, if you are using binaries supplied by another user of CPU2017, that person should also supply the config file.
4. Or, write your own.

Name it: Copy your selection to a new file in the config directory. Do not use blanks in the name.
Hint: make the name something useful to *you*.

%SPEC%\config\Rahuls_first_test.cfg
$SPEC/config/JeffWantsYetAnotherTest.Compiler.v11.beta4.cfg

Edit the label: Look for a line like one of these:

%define label something
or
label = something

The label is an arbitrary tag added to your binaries and directories, which comes in handy when you need to hunt them down. As with the config file name, make it something meaningful to *you*. No blanks are allowed.

Other Edits:

1. Look for any locations marked EDIT and make changes as needed.
2. Look for paths and adjust if needed (example: your compiler is in /opt/bin but the config file uses /usr/bin).
3. Look for any commands in the config file, and verify that they will not cause surprises.

Ready to run: You're ready to give it a try. Enter runcpu --config=name followed by a list of benchmarks or suites (see table above). Examples:

```
runcpu --config=LaCasa_icc-speed.cfg --define build_ncpus=12 --threads=4 --
output_root=/home/user/cpu2017/Speed/LaCasa_4Threads   --reportable fpspeed intspeed
```

The above command would build fpspeed and intspeed benchmark suits using 12 CPUS and execute using 4 threads. The executable and the results are redirected to the location defined by output root.

```
runcpu --config=LaCasa_icc-rate.cfg --define build_ncpus=12 --copies=4  --
output_root=/home/user/cpu2017/Rate/LaCasa_4Copies  --reportable fprate intrate
```

The above command would build fprate and intrate benchmark suits using 12 CPUS and execute 4 copies. The executable and the results are redirected to the location defined by output root.

runcpu --config=eniac.cfg --action=build 519.lbm_r
runcpu --config=colossus.cfg --threads=16 628.pop2_s
runcpu --config=z3.cfg --copies=64 fprate

The first example compiles the benchmark named 519.lbm_r.
The second runs the SPECspeed benchmark 628.pop2_s using 16 OpenMP threads.
The third runs 64 copies of all the SPECrate 2017 Floating Point benchmarks. The Install Guide chapter on "Testing Your Installation" has suggestions that start small and build up (Unix, Windows).

4 Running SPEC benchmarks (outside SPEC utilities)

We often want to run individual benchmarks from a command line, or to run a spec benchmark on top of an architectural simulator.
To do this, you may locate executable and input files and copy them in your working directory.
You also need to know how to launch a particular benchmark.

We will use 500.perlbench_r benchmark as an example.
Note: all benchmarks in SPEC CPU2017 share identical structure.

Go to a directory for 500.perlbench_r.

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ cd 500.perlbench_r/
-bash-4.1$ pwd
/apps/arch/cpu2017/benchspec/CPU/500.perlbench_r
-bash-4.1$ ls
Docs  Spec  data  src  version.txt
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

You see doc, data, Spec, and src directories.
For example, src contains source files.

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ cd src/
-bash-4.1$ ls
DynaLoader.c          cop.h            ext            inline.h             time64.h
EXTERN.h              cpan             fakesdio.h     intrpvar.h           mg_vtable.h
INTERN.h              cv.h             feature.h      invlist_inline.h     mro_core.c
Makefile              deb.c            form.h         iperlsys.h           mydtrace.h
Opcode.c              dist             git_version.h  keywords.c           nostdio.h
XSUB.h                doio.c           globals.c      keywords.h           numeric.c
av.c                  doop.c           gv.c           l1_char_class_tab.h  op.c
av.h                  dquote.c         gv.h           locale.c             op.h
caretx.c              ebcdic_tables.h  hv.c           mg.c                 opcode.h
charclass_invlists.h  embed.h          hv.h           mg.h                 opnames.h
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

4.1 Executables

Executables can be located in the exe directory.
They are prepared by running runspec utility with a configuration file that specifies conditions of compilation (compiler, optimization level, etc).

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ cd exe/
-bash-4.1$ ls
perlbench_r_base.icc_rate
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

Copy the executable into your working directory.

4.2 Inputs

The directory data contains input files.
SPEC supports three classes of inputs test (smallest), train (a bit larger), and reference input sets that are used in reporting performance.

```
<<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-bash-4.1$ ls data
all  ref  test  train

-bash-4.1$ ls data/refrate/input/
checkspam.in  checkspam.pl  diffmail.in  splitmail.in
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>>
```

Copy selected input files to your working directory.

The next section describes how to run individual benchmarks from the command line (some benchmarks have multiple input data sets).

5 SPEC CPU2017 command lines

The following command lines for each of the SPEC CPU2017 benchmarks were obtained using 'specinvoke -nn' in the needed run directories.

NOTE: the command lines below should be modified to match executables.
The following groupings of benchmarks are based on the suits.

int speed

600.perlbench_s (3 inputs)
---

reference inputs:

```
perlbench_s_base.icc_speed -I./lib checkspam.pl 2500 5 25 11 150 1 1 1 1 0<&- >
checkspam.2500.5.25.11.150.1.1.1.1.out 2>> checkspam.2500.5.25.11.150.1.1.1.1.err

perlbench_s_base.icc_speed -I./lib diffmail.pl 4 800 10 17 19 300 0<&- >
diffmail.4.800.10.17.19.300.out 2>> diffmail.4.800.10.17.19.300.err

perlbench_s_base.icc_speed -I./lib splitmail.pl 6400 12 26 16 100 0 0<&- >
splitmail.6400.12.26.16.100.0.out 2>> splitmail.6400.12.26.16.100.0.err
```

602.gcc (3 inputs)
---

reference inputs:

```
 sgcc_base.icc_speed gcc-pp.c -O5 -fipa-pta -o gcc-pp.opts-O5_-fipa-pta.s 0<&- >
gcc-pp.opts-O5_-fipa-pta.out 2>> gcc-pp.opts-O5_-fipa-pta.err

sgcc_base.icc_speed gcc-pp.c -O5 -finline-limit=1000 -fselective-scheduling -
fselective-scheduling2 -o gcc-pp.opts-O5_-finline-limit_1000_-fselective-scheduling_-
fselective-scheduling2.s 0<&- > gcc-pp.opts-O5_-finline-limit_1000_-fselective-
scheduling_-fselective-scheduling2.out 2>> gcc-pp.opts-O5_-finline-limit_1000_-
fselective-scheduling_-fselective-scheduling2.err

sgcc_base.icc_speed gcc-pp.c -O5 -finline-limit=24000 -fgcse -fgcse-las -fgcse-lm
-fgcse-sm -o gcc-pp.opts-O5_-finline-limit_24000_-fgcse_-fgcse-las_-fgcse-lm_-fgcse-
sm.s 0<&- > gcc-pp.opts-O5_-finline-limit_24000_-fgcse_-fgcse-las_-fgcse-lm_-fgcse-
sm.out 2>> gcc-pp.opts-O5_-finline-limit_24000_-fgcse_-fgcse-las_-fgcse-lm_-fgcse-
sm.err
```

605.mcf_s (1 input)
---

reference inputs:

```
mcf_s_base.icc_speed inp.in 0<&- > inp.out 2>> inp.err
```

620.omnetpp_s (1 input)
---

reference inputs:

```
    omnetpp_s_base.icc_speed -c General -r 0 0<&- > omnetpp.General-0.out 2>>
omnetpp.General-0.err

623.xalancbmk_s (1 input)
-----------------

    reference inputs:
    xalancbmk_s_base.icc_speed -v t5.xml xalanc.xsl 0<&- > ref-t5.out 2>> ref-t5.err

625.x264_s (3 inputs)
------------------

    reference inputs:

    x264_s_base.icc_speed --pass 1 --stats x264_stats.log --bitrate 1000 --frames 1000
-o BuckBunny_New.264 BuckBunny.yuv 1280x720 0<&- > run_000-
1000_x264_s_base.icc_speed_no_affinity_x264_pass1.out 2>> run_000-
1000_x264_s_base.icc_speed_no_affinity_x264_pass1.err

    x264_s_base.icc_speed --pass 2 --stats x264_stats.log --bitrate 1000 --dumpyuv 200
--frames 1000 -o BuckBunny_New.264 BuckBunny.yuv 1280x720 0<&- > run_000-
1000_x264_s_base.icc_speed_no_affinity_x264_pass2.out 2>> run_000-
1000_x264_s_base.icc_speed_no_affinity_x264_pass2.err

    x264_s_base.icc_speed --seek 500 --dumpyuv 200 --frames 1250 -o BuckBunny_New.264
BuckBunny.yuv 1280x720 0<&- > run_0500-1250_x264_s_base.icc_speed_no_affinity_x264.out
2>> run_0500-1250_x264_s_base.icc_speed_no_affinity_x264.err



631.deepsjeng_s (1 input)
--------------------
    reference inputs:
    deepsjeng_s_base.icc_speed ref.txt 0<&- > ref.out 2>> ref.err

641.leela_s (1 input)
---------------------
    reference inputs:
    leela_s_base.icc_speed ref.sgf 0<&- > ref.out 2>> ref.err


648.exchange2_s (1 input)
-----------------------
    reference inputs:
    exchange2_s_base.icc_speed 6 0<&- > exchange2.txt 2>> exchange2.err

657.xz_s (2 input)
----------------------

    reference inputs:

    xz_s_base.icc_speed cpu2006docs.tar.xz 6643
055ce243071129412e9dd0b3b69a21654033a9b723d874b2015c774fac1553d9713be561ca86f74e4f16f2
2e664fc17a79f30caa5ad2c04fbc447549c2810fae 1036078272 1111795472 4 0<&- >
cpu2006docs.tar-6643-4.out 2>> cpu2006docs.tar-6643-4.err

    xz_s_base.icc_speed cld.tar.xz 1400
19cf30ae51eddcbefda78dd06014b4b96281456e078ca7c13e1c0c9e6aaea8dff3efb4ad6b0456697718ce
de6bd5454852652806a657bb56e07d61128434b474 536995164 539938872 8 0<&- > cld.tar-1400-
8.out 2>> cld.tar-1400-8.err
```

float_speed

```
603.bwaves_s (1 input)
------------------
    reference inputs:
    speed_bwaves_base.icc_speed bwaves_1 < bwaves_1.in > bwaves_1.out 2>> bwaves_1.err

    speed_bwaves_base.icc_speed bwaves_2 < bwaves_2.in > bwaves_2.out 2>> bwaves_2.err

607.cactuBSSN_s (1 inputs)
--------------------
    reference inputs:

    cactuBSSN_s_base.icc_speed spec_ref.par 0<&- > spec_ref.out 2>> spec_ref.err


619.lbm_s (1 input)
--------------------
    reference inputs:
    lbm_s_base.icc_speed 2000 reference.dat 0 0 200_200_260_ldc.of 0<&- > lbm.out 2>>
lbm.err

621.wrf_s (1 inputs)
---------------------
    reference inputs:

    wrf_s_base.icc_speed 0<&- > rsl.out.0000 2>> wrf.err

627.cam4_s (1 input)
--------------------
    reference inputs:
    cam4_s_base.icc_speed 0<&- > cam4_s_base.icc_speed_no_affinity.txt 2>>
cam4_s_base.icc_speed_no_affinity.err

628.pop2_s (1 input)
--------------------
    reference inputs:
    speed_pop2_base.icc_speed 0<&- > pop2_s.out 2>> pop2_s.err

638.imagick_s (1 input)
-----------------------
    reference inputs:
    imagick_s_base.icc_speed -limit disk 0 refspeed_input.tga -resize 817% -rotate -
2.76 -shave 540x375 -alpha remove -auto-level -contrast-stretch 1x1% -colorspace Lab -
channel R -equalize +channel -colorspace sRGB -define histogram:unique-colors=false -
adaptive-blur 0x5 -despeckle -auto-gamma -adaptive-sharpen 55 -enhance -brightness-
contrast 10x10 -resize 30% refspeed_output.tga 0<&- > refspeed_convert.out 2>>
refspeed_convert.err


644.nab_s (1 inputs)
--------------------
    reference inputs:

    nab_s_base.icc_speed 3j1n 20140317 220 0<&- > 3j1n.out 2>> 3j1n.err

649.fotonik3d_s (1 input)
-------------------
    reference inputs:
    fotonik3d_s_base.icc_speed 0<&- > fotonik3d_s.log 2>> fotonik3d_s.err
```

```
654.roms_s (1 input)
----------------------
    reference inputs:
    sroms_base.icc_speed < ocean_benchmark3.in > ocean_benchmark3.log 2>>
ocean_benchmark3.err
```

int rate

```
500.perlbench_s (3 inputs)
---------------------------

    reference inputs:

    perlbench_r_base.icc_rate -I./lib checkspam.pl 2500 5 25 11 150 1 1 1 1 0<&- >
checkspam.2500.5.25.11.150.1.1.1.1.out 2>> checkspam.2500.5.25.11.150.1.1.1.1.err

    perlbench_r_base.icc_rate -I./lib diffmail.pl 4 800 10 17 19 300 0<&- >
diffmail.4.800.10.17.19.300.out 2>> diffmail.4.800.10.17.19.300.err

    perlbench_r_base.icc_rate -I./lib splitmail.pl 6400 12 26 16 100 0 0<&- >
splitmail.6400.12.26.16.100.0.out 2>> splitmail.6400.12.26.16.100.0.err


502.gcc (5 inputs)
--------------------
    reference inputs:

   cpugcc_r_base.icc_rate gcc-pp.c -O3 -finline-limit=0 -fif-conversion -fif-
conversion2 -o gcc-pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2.s 0<&-
> gcc-pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2.out 2>> gcc-
pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2.err

    cpugcc_r_base.icc_rate gcc-pp.c -O2 -finline-limit=36000 -fpic -o gcc-pp.opts-O2_-
finline-limit_36000_-fpic.s 0<&- > gcc-pp.opts-O2_-finline-limit_36000_-fpic.out 2>>
gcc-pp.opts-O2_-finline-limit_36000_-fpic.err

    cpugcc_r_base.icc_rate gcc-smaller.c -O3 -fipa-pta -o gcc-smaller.opts-O3_-fipa-
pta.s 0<&- > gcc-smaller.opts-O3_-fipa-pta.out 2>> gcc-smaller.opts-O3_-fipa-pta.err

    cpugcc_r_base.icc_rate ref32.c -O5 -o ref32.opts-O5.s 0<&- > ref32.opts-O5.out 2>>
ref32.opts-O5.err

   cpugcc_r_base.icc_rate ref32.c -O3 -fselective-scheduling -fselective-scheduling2 -
o ref32.opts-O3_-fselective-scheduling_-fselective-scheduling2.s 0<&- > ref32.opts-
O3_-fselective-scheduling_-fselective-scheduling2.out 2>> ref32.opts-O3_-fselective-
scheduling_-fselective-scheduling2.err


505.mcf_s (1 input)
--------------------
    reference inputs:

    mcf_r_base.icc_rate inp.in 0<&- > inp.out 2>> inp.err

520.omnetpp_s (1 input)
---------------------

    reference inputs:

    omnetpp_r_base.icc_rate_no_affinity -c General -r 0 0<&- > omnetpp.General-0.out
2>> omnetpp.General-0.err
```

```
523.xalancbmk_s (1 input)
-----------------

    reference inputs:
    cpuxalan_r_base.icc_rate -v t5.xml xalanc.xsl 0<&- > ref-t5.out 2>> ref-t5.err

525.x264_s (3 inputs)
------------------

    reference inputs:

    x264_r_base.icc_rate --pass 1 --stats x264_stats.log --bitrate 1000 --frames 1000
-o BuckBunny_New.264 BuckBunny.yuv 1280x720 0<&- > run_000-
1000_x264_r_base.icc_rate_no_affinity_x264_pass1.out 2>> run_000-
1000_x264_r_base.icc_rate_no_affinity_x264_pass1.err

    x264_r_base.icc_rate --pass 2 --stats x264_stats.log --bitrate 1000 --dumpyuv 200
--frames 1000 -o BuckBunny_New.264 BuckBunny.yuv 1280x720 0<&- > run_000-
1000_x264_r_base.icc_rate_no_affinity_x264_pass2.out 2>> run_000-
1000_x264_r_base.icc_rate_no_affinity_x264_pass2.err

    x264_r_base.icc_rate --seek 500 --dumpyuv 200 --frames 1250 -o BuckBunny_New.264
BuckBunny.yuv 1280x720 0<&- > run_0500-1250_x264_r_base.icc_rate_no_affinity_x264.out
2>> run_0500-1250_x264_r_base.icc_rate_no_affinity_x264.err


531.deepsjeng_s (1 input)
--------------------
    reference inputs:
    deepsjeng_r_base.icc_rate_no_affinity ref.txt 0<&- > ref.out 2>> ref.err


541.leela_s (1 input)
---------------------
    reference inputs:
    leela_r_base.icc_rate ref.sgf 0<&- > ref.out 2>> ref.err


548.exchange2_s (1 input)
-----------------------
    reference inputs:
    exchange2_r_base.icc_rate6 0<&- > exchange2.txt 2>> exchange2.err

557.xz_s (2 input)
----------------------

    reference inputs:

    xz_r_base.icc_rate cld.tar.xz 160
19cf30ae51eddcbefda78dd06014b4b96281456e078ca7c13e1c0c9e6aaea8dff3efb4ad6b0456697718ce
de6bd5454852652806a657bb56e07d61128434b474 59796407 61004416 6 0<&- > cld.tar-160-
6.out 2>> cld.tar-160-6.err

    xz_r_base.icc_rate cpu2006docs.tar.xz 250
055ce243071129412e9dd0b3b69a21654033a9b723d874b2015c774fac1553d9713be561ca86f74e4f16f2
2e664fc17a79f30caa5ad2c04fbc447549c2810fae 23047774 23513385 6e 0<&- >
cpu2006docs.tar-250-6e.out 2>> cpu2006docs.tar-250-6e.err

   xz_r_base.icc_rate input.combined.xz 250
a841f68f38572a49d86226b7ff5baeb31bd19dc637a922a972b2e6d1257a890f6a544ecab967c313e37047
8c74f760eb229d4eef8a8d2836d233d3e9dd1430bf 40401484 41217675 7 0<&- > input.combined-
250-7.out 2>> input.combined-250-7.err
```

float_rate

```
503.bwaves_r (3 input)
-------------------------
    reference inputs:
    bwaves_r_base.icc_rate bwaves_1 < bwaves_1.in > bwaves_1.out 2>> bwaves_1.err
    bwaves_r_base.icc_rate bwaves_2 < bwaves_2.in > bwaves_2.out 2>> bwaves_2.err
    bwaves_r_base.icc_rate bwaves_3 < bwaves_3.in > bwaves_3.out 2>> bwaves_3.err
    bwaves_r_base.icc_rate bwaves_4 < bwaves_4.in > bwaves_4.out 2>> bwaves_4.err

507.cactuBSSN_r (1 inputs)
----------------------
    reference inputs:

    cactusBSSN_r_base.icc_rate spec_ref.par 0<&- > spec_ref.out 2>> spec_ref.err

508.namd_r (1 input)
-------------------
    reference inputs:
    namd_r_base.icc_rate --input apoa1.input --output apoa1.ref.output --iterations 65
0<&- > namd.out 2>> namd.err

510.parest_r (1 input)
-----------------
    reference inputs:
    parest_r_base.icc_rate ref.prm 0<&- > ref.out 2>> ref.err

511.povray_r (1 input)
---------------------
    reference inputs:
    povray_r_base.icc_rate SPEC-benchmark-ref.ini 0<&- > SPEC-benchmark-ref.stdout 2>>
SPEC-benchmark-ref.stderr

519.lbm_r (2 inputs)
--------------------
    reference inputs:

    lbm_r_base.icc_rate 3000 reference.dat 0 0 100_100_130_ldc.of 0<&- > lbm.out 2>>
lbm.err

521.wrf_r (1 input)
-----------------
    reference inputs:
    wrf_r_base.icc_rate 0<&- > rsl.out.0000 2>> wrf.err

526.blender_r (1 input)
----------------------
    reference inputs:
    blender_r_base.icc_rate sh3_no_char.blend --render-output sh3_no_char_ --threads 1
-b -F RAWTGA -s 849 -e 849 -a 0<&- > sh3_no_char.849.spec.out 2>>
sh3_no_char.849.spec.err



527_cam4 (1 input)
-----------------------
    reference inputs:
    cam4_r_base.icc_rate 0<&- > cam4_r_base.icc_rate_no_affinity.txt 2>>
cam4_r_base.icc_rate_no_affinity.err
```

```
538.imagick_r (1 input)
-----------------------
    reference inputs:
    imagick_r_base.icc_rate -limit disk 0 refrate_input.tga -edge 41 -resample 181% -
emboss 31 -colorspace YUV -mean-shift 19x19+15% -resize 30% refrate_output.tga 0<&- >
refrate_convert.out 2>> refrate_convert.err


544.nab_r (1 input)
-----------------------
    reference inputs:
    nab_r_base.icc_rate 1am0 1122214447 122 0<&- > 1am0.out 2>> 1am0.err


549.fotonik3d_r (1 input)
-----------------------
    reference inputs:

    fotonik3d_r_base.icc_rate 0<&- > fotonik3d_r.log 2>> fotonik3d_r.err

554.roms_r (1 input)
-----------------------
    reference inputs:

    roms_r_base.icc_rate < ocean_benchmark2.in.x > ocean_benchmark2.log 2>>
ocean_benchmark2.err
```