MCR10149

Enhancing `instr_speed` with a **Composite Geometric Mean MIPS** Metric

Jeffrey H. Johnson

March 4th 2025

Problem Description

The `instr_speed` utility runs many (*currently 15*) benchmarks, printing the results for each individually. The utility does ***not*** calculate a *single overall performance metric*, which is very useful for quick comparisons. See Trac ticket #350[1] for details.

Proposed Fix

This MCR proposes modifying `instr_speed.pl1` to add 'running product' and 'test count' variables. These variables will be used to calculate a composite *geometric mean*[2] MIPS metric, which will be displayed after all tests have been run.

Rationale

The choice of the *composite geometric mean* MIPS metric to represent `instr_speed`'s *overall performance* was selected because *a*) the standard *arithmetic mean* (averaging) method is sensitive to outliers, *b*) geometric mean calculations are generally preferred when the inputs involve rates or ratios, and *c*) it is in line with industry standard practices. SPEC, for example, uses geometric means (albeit weighted) to calculate their single number composite scores.

Examples

Consider the following set, which is *extreme* for the purposes of illustration, to demonstrate the outlier effect:

{ '9.5', '10.5', '12.1', '9.9', '10.9', '11.5', '12.0', '13.5', '19.0', '11.2', '800.0' }

The standard arithmetic mean for this set is **83.6 MIPS**, while the geometric mean is **17.3 MIPS**.

This can be contrasted with these more reasonable values, as might be encountered on a real system:

{ '19.0', '25.2', '12.0', '10.5', '13.5', '10.9', '12.1', '9.5', '9.9', '11.2', '11.5' }

The standard arithmetic mean for this set is **13.2 MIPS** and the geometric mean is **12.6 MIPS**.

Using the geometric mean allows us to avoid misleadingly skewing the composite MIPS result in the case of aggressively optimizing for a single test (possibly at the expense of other tests).

[1] *Trac*: "*Add composite geometric mean MIPS result to instr_speed utility*", `https://multics-trac.swenson.org/ticket/350`.
[2] *Wikipedia:* "*Geometric mean*", `https://en.wikipedia.org/w/index.php?title=Geometric_mean&oldid=1278147482`.

Weighting?

It could be argued that it might be "more accurate" to use a *weighted* calculation (as SPEC does), where we would apply an adjustment to each of the 15 tests based on how representative each benchmark code sequence matches some (theoretical) "typical Multics workload".

For example, we would likely apply less weight to `TEST #14` (*nop*), because how quickly the CPU can do nothing is certainly not characteristic of actual programs, so it is less useful to the composite than `TEST #12` (*random mix*), to which we would likely end up applying a much greater weight.

The challenge, of course, lies in creating a profile of a "typical Multics workload" that is truly representative.

At this time, I do **not** believe expending the effort to determine appropriate weightings would be worthwhile, although if they were developed, updating the code to support such weightings would be a trivial task.

This decision may be re-examined at a later time.

Code modifications

`cpa instr_speed.pl1 instr_speed_new.pl1`:

```
A45       dcl (successes, pf_aborts, nanos, ls_aborts, type, histi, bucketmin, bucketmax, pf, ls, maxs, bucket,
                                                                                    count, nargs) fixed bin;
Changed by B to:
B45       dcl (successes, pf_aborts, nanos, ls_aborts, type, histi, bucketmin, bucketmax, pf, ls, maxs, bucket,
                                                                          count, nargs, ntests) fixed bin;


Inserted in B:
B53       dcl  mips_prod float bin(63);
Preceding:
A53       dcl  hist (0:300) fixed bin;


Inserted in B:
B124               ntests = 0;
B125               mips_prod = 0e0;
Preceding:
A123
A124      /* Now run the test for the 15 possible types of sequences */


A158                    mip_rate = mips_total/float (successes);     /* calculate mip_rate */
Changed by B to:
B161                    ntests = ntests + 1;
B162                    mip_rate = mips_total/float (successes);     /* calculate mip_rate */
B163                    if mips_prod = 0e0 then mips_prod = mip_rate; /* initialize mips_prod */
B164                    else mips_prod = mips_prod * mip_rate;       /* calculate mips_prod */


Inserted in B:
B170
B171               call ioa_ ("^/* * * * * * * * * * * * * * * * * * * * * *^/");
B172               call ioa_ ("COMPOSITE GEOMETRIC MEAN MIPS = ^6.3f^/", mips_prod ** (1 / float (ntests)));
Preceding:
A164
```

`Comparison finished: 5 differences, 13 lines.`

Documentation

Any composite metric, when used alone, can serve as a filter that obscures rather than clarifies benchmark results. There is a danger that users may rely *only* on the new metric in isolation and draw inappropriate conclusions regarding system performance. To prevent this, the documentation (provided via the *info* segment `instr_speed.info`) will be updated so users will be informed that the composite is an *additional* piece of information and a *rough estimation* that, while still useful, should not be solely relied upon.

The following text is suggested for inclusion in the "**Notes**" section of the `instr_speed.info` segment:

```
Synthetic benchmarks evaluate only narrow aspects of system performance.  Consider
all instr_speed results, especially the COMPOSITE GEOMETRIC MEAN MIPS, as one
data point and not a sole indicator to avoid drawing misleading conclusions
about overall system performance.
```

Test

The changes described will be tested by running the updated `instr_speed` program on several Multics systems with varying performance characteristics.

The results of preliminary testing have been positive. The new geometric mean MIPS metric closely aligns with the other MIPS-reporting benchmarks (*i.e.* '`perftest`', '`nqueensx`'[3]) currently in regular use by the **DPS8M Development Team** for performance estimation:

System Name	**Geometric Mean** MIPS	'`perftest`' MIPS	'`nqueensx`' MIPS
BAN-AI	`14.1`	`14.2`	`14.1`
CASINO	`13.8`	`13.9`	`14.0`
DPDEV	`11.6`	`11.1`	`11.2`
ENGINE	`1.81`	`1.95`	`1.84`
RP-3	`1.56`	`1.54`	`1.55`
RP-2	`0.91`	`0.94`	`0.95`

Future considerations

While reviewing the `instr_speed` program, several potential improvements beyond the previously discussed *weighting* emerged. We will evaluate the following potential improvements for future implementation:

- Eliminating references to "page fault" retries, which should be consistently zero in the current version of the `instr_speed` program.
- Enhancing the histogram display (enabled with the `-long` control argument) to accurately scale to systems with performance levels exceeding `1.5 MIPS`.
- Providing users with additional statistical insights, such as alternative calculation methods, improved detection of statistical outliers, and metrics such as standard deviations.
- Enabling privileged users to test *all* online processors of a multiprocessor system with a single `instr_speed` invocation, rather than requiring multiple invocations with `set_proc_required`.
- Allowing the user to increase the number of test iterations.
- Showing the results relative to a baseline.

[3] '`perftest`' and '`nqueensx`' are standalone benchmarks for use with the *DPS8M Simulator*, based on solving the *N-queens puzzle*, where, by default, $N = 10$ for '`perftest`' and $N = 11$ for '`nqueensx`'. The '`perftest`' benchmark runs in a minimal environment simulating *only* the main processor (with all I/O disabled), while '`nqueensx`' runs in the standard simulator environment. See the code at GitLab (`https://gitlab.com/dps8m/dps8m/-/tree/master/src/perf_test`) for additional details.

Version History

2025-03-02	1.0	Initial version of the MCR.
2025-03-03	1.1	Add preliminary testing results, proposed info text, and changes from early reviewers.
2025-03-03	1.2	Link to Trac ticket #350.
2025-03-04	1.3	Assigned MCR10149.
2025-03-04	1.4	Desribe 'perftest' and 'nqueensx' benchmarks, adjust "Future considerations" points.
2025-03-04	1.5	Add external hyperlinks to source code.