­Metrics Descriptions and Formulas

1. spf
   1. spf stands for the metric, seconds per federate count.
   2. For a given benchmark, benchmark type, run, and core type, spf takes the real time values and divides them by the corresponding federate count values (if available).
   3. For example, consider echoBenchmark, full, bScQ6, and ipc. Then the spf for a real time of 1 second and federate count of 2 would be ½ = 0.5.
   4. Formula:
2. spf\_ratio
   1. spf\_ratio stands for the metric, seconds per federate count ratio.
   2. This metric takes the results of the spf calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
   3. Consider the example from (1) and let core type ‘inproc’ be the baseline. Suppose spf(inproc) = 0.2. Then spf\_ratio(ipc) = spf(ipc)/spf(inproc) = 0.5/0.2 = 2.5. Note that when performing this calculation for all the results, we will get 1 when calculating spf\_ratio(inproc) = spf(inproc)/spf(inproc).
   4. Formula:
3. real\_time\_ratio
   1. real\_time\_ratio divides the real\_time values for a given benchmark, benchmark type, run, and core type by the real\_time value of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
   2. Consider the example from (1) and let core type ‘inproc’ be the baseline. Suppose real\_time(inproc) = 0.5 seconds and real\_time(ipc) = 1.2 seconds. Then real\_time\_ratio(ipc) = real\_time(ipc)/real\_time(inproc) = 1.2/0.5 = 2.4. Note that when performing this calculation for all the results, we will get 1 when calculating real\_time\_ratio(inproc) = real\_time(inproc)/real\_time(inproc).
   3. Formula:
4. cpf
   1. cpf stands for the metric, cycles per federate count.
   2. For a given benchmark, benchmark type, run, and core type, cpf takes the real time values and multiplies them by the corresponding MHz per CPU value and the previously calculated spf values. Recall, MHz refers to clock cycles per second. So, cpf calculates how many cycles are done per federate count.
   3. For example, consider echoBenchmark, full, bScQ6, 3400 MHz, and ipc. Then the cpf for a real time of 1.2 second, spf of 0.2, MHz of 3400 would be 1.2\*3400\*0.2 = 816.
   4. Formula:
5. cpf\_ratio
   1. cpf\_ratio stands for the metric, cycles per federate count ratio.
   2. This metric takes the results of the cpf calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
   3. Consider the example from (4) and let core type ‘inproc’ be the baseline. Suppose cpf(inproc) = 240. Then cpf\_ratio(ipc) = cpf(ipc)/cpf(inproc) = 816/240 = 3.4. Note that when performing this calculation for all the results, we will get 1 when calculating cpf\_ratio(inproc) = cpf(inproc)/cpf(inproc).
   4. Formula:
6. spms
   1. spms stands for the metric, seconds per message size.
   2. For a given benchmark, benchmark type, run, and core type, spms takes the real time values and divides them by the corresponding message size values (if available).
   3. For example, consider messageSendBenchmark, full, bScQ6, and ipc. Then the spms for a real time of 1 second and message size of 8 would be 1/8 = 0.125.
   4. Formula:
7. spms\_ratio
   1. spms\_ratio stands for the metric, seconds per message size ratio.
   2. This metric takes the results of the spms calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
   3. Consider the example from (6) and let core type ‘inproc’ be the baseline. Suppose spms(inproc) = 0.4. Then spms\_ratio(ipc) = spms(ipc)/spms(inproc) = 0.125/0.4 = 0.3125. Note that when performing this calculation for all the results, we will get 1 when calculating spms\_ratio(inproc) = spms(inproc)/spms(inproc).
   4. Formula:
8. spmc
   1. spmc stands for the metric, seconds per message count.
   2. For a given benchmark, benchmark type, run, and core type, spmc takes the real time values and divides them by the corresponding message count values (if available).
   3. For example, consider messageSendBenchmark, full, bScQ6, and ipc. Then the spmc for a real time of 1 second and message count of 4 would be ¼ = 0.25.
   4. Formula:
9. spmc\_ratio
   1. spmc\_ratio stands for the metric, seconds per message count ratio.
   2. This metric takes the results of the spmc calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
   3. Consider the example from (8) and let core type ‘inproc’ be the baseline. Suppose spmc(inproc) = 0.34. Then spmc\_ratio(ipc) = spmc(ipc)/spmc(inproc) = 0.25/0.34 = ~0.735. Note that when performing this calculation for all the results, we will get 1 when calculating spmc\_ratio(inproc) = spmc(inproc)/spmc(inproc).
   4. Formula:
10. cpms
    1. cpms stands for the metric, cycles per message size.
    2. For a given benchmark, benchmark type, run, and core type, cpms takes the real time values and multiplies them by the corresponding MHz per CPU value and the previously calculated spms values. Recall, MHz refers to clock cycles per second. So, cpms calculates how many cycles are done per message size.
    3. For example, consider messageSendBenchmark, full, bScQ6, 3400 MHz, and ipc. Then the cpms for a real time of 2.2 second, spms of 1.3, MHz of 3400 would be 2.2\*3400\*1.3 = 9724.
    4. Formula:
11. cpms\_ratio
    1. cpms\_ratio stands for the metric, cycles per message size ratio.
    2. This metric takes the results of the cpms calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (10) and let core type ‘inproc’ be the baseline. Suppose cpms(inproc) = 1040. Then cpms\_ratio(ipc) = cpms(ipc)/cpf(inproc) = 9724/1040 = 9.35. Note that when performing this calculation for all the results, we will get 1 when calculating cpms\_ratio(inproc) = cpms(inproc)/cpms(inproc).
    4. Formula:
12. cpmc
    1. cpms stands for the metric, cycles per message count.
    2. For a given benchmark, benchmark type, run, and core type, cpmc takes the real time values and multiplies them by the corresponding MHz per CPU value and the previously calculated spmc values. Recall, MHz refers to clock cycles per second. So, cpmc calculates how many cycles are done per message count.
    3. For example, consider messageSendBenchmark, full, bScQ6, 3400 MHz, and ipc. Then the cpmc for a real time of 0.2 second, spmc of 2.4, MHz of 3400 would be 0.2\*3400\*2.4 = 1632.
    4. Formula:
13. cpmc\_ratio
    1. cpmc\_ratio stands for the metric, cycles per message count ratio.
    2. This metric takes the results of the cpmc calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (12) and let core type ‘inproc’ be the baseline. Suppose cpmc(inproc) = 142. Then cpmc\_ratio(ipc) = cpmc(ipc)/cpmc(inproc) = 1632/142 = ~11.493. Note that when performing this calculation for all the results, we will get 1 when calculating cpmc\_ratio(inproc) = cpmc(inproc)/cpmc(inproc).
    4. Formula:
14. spi
    1. spi stands for the metric, seconds per interface count.
    2. For a given benchmark, benchmark type, run, and core type, spi takes the real time values and divides them by the corresponding interface count values (if available).
    3. For example, consider messageLookupBenchmark, full, bScQ6, and ipc. Then the spi for a real time of 1 second and interface count of 200 would be 1/200.
    4. Formula:
15. spi\_ratio
    1. spi\_ratio stands for the metric, seconds per interface count ratio.
    2. This metric takes the results of the spi calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (14) and let core type ‘inproc’ be the baseline. Suppose spi(inproc) = 0.2. Then spi\_ratio(ipc) = spi(ipc)/spi(inproc) = 0.005/0.2 = 0.025. Note that when performing this calculation for all the results, we will get 1 when calculating spi\_ratio(inproc) = spi(inproc)/spi(inproc).
    4. Formula:
16. cpi
    1. cpi stands for the metric, cycles per interface count.
    2. For a given benchmark, benchmark type, run, and core type, cpi takes the real time values and multiplies them by the corresponding MHz per CPU value and the previously calculated spi values. Recall, MHz refers to clock cycles per second. So, cpi calculates how many cycles are done per interface count.
    3. For example, consider messageLookupBenchmark, full, bScQ6, 3400 MHz, and ipc. Then the cpi for a real time of 0.82 second, spi of 0.25, MHz of 3400 would be 0.82\*3400\*0.25 = 697.
    4. Formula:
17. cpi\_ratio
    1. cpi\_ratio stands for the metric, cycles per interface count ratio.
    2. This metric takes the results of the cpi calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (16) and let core type ‘inproc’ be the baseline. Suppose cpi(inproc) = 385. Then cpi\_ratio(ipc) = cpi(ipc)/cpi(inproc) = 697/385 = ~1.81. Note that when performing this calculation for all the results, we will get 1 when calculating cpi\_ratio(inproc) = cpi(inproc)/cpi(inproc).
    4. Formula:
18. spe
    1. spe stands for the metric, seconds per event count.
    2. For a given benchmark, run, and core type, spe takes the real time values and divides them by the corresponding event count values (if available).
    3. For example, consider EchoLeafFederate, 4799007, and tpcss. Then the spe for a real time of 1 second and interface count of 1000 would be 1/1000.
    4. Formula:
19. spe\_ratio
    1. spe\_ratio stands for the metric, seconds per event count ratio.
    2. This metric takes the results of the spe calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (18) and let core type ‘tcp’ be the baseline. Suppose spe(tcp) = 0.35. Then spe\_ratio(tpcss) = spe(tcpss)/spe(tcp) = 0.001/0.35 = ~0.00286. Note that when performing this calculation for all the results, we will get 1 when calculating spe\_ratio(tcp) = spe(tcp)/spe(tcp).
    4. Formula:
20. cpe
    1. cpe stands for the metric, cycles per event count.
    2. For a given benchmark, run, and core type, cpe takes the real time values and multiplies them by the corresponding MHz per CPU value and the previously calculated spe values. Recall, MHz refers to clock cycles per second. So, cpe calculates how many cycles are done per event count.
    3. For example, consider EchoLeafFederatel, 4799007, 2100 MHz, and tpcss. Then the cpe for a real time of 0.65 second, spe of 0.05, MHz of 2100 would be 0.65\*2100\*0.05 = 68.25.
    4. Formula:
21. cpe\_ratio
    1. cpe\_ratio stands for the metric, cycles per event count ratio.
    2. This metric takes the results of the cpe calculations and divides them by the result of a specific core type’s value. This specific core type is considered the baseline for the runs. The most common core types are ‘inproc’ for single node benchmark runs and ‘tcp’ for multi-node benchmark runs.
    3. Consider the example from (20) and let core type ‘tcp’ be the baseline. Suppose cpe(tcp) = 6.5. Then cpe\_ratio(tcpss) = cpe(tcpss)/cpe(tcp) = 68.25/6.5 = 10.5. Note that when performing this calculation for all the results, we will get 1 when calculating cpe\_ratio(tcp) = cpe(tcp)/cpe(tcp).
    4. Formula:
22. cpu\_score
    1. cpu\_score stands for the CPU Benchmark Score for the HELICS benchmarks.
    2. This metric is used to measure the overall performance of HELICS across time. It uses the average of the available metrics for a specific HELICS version run (in other words, a version of a HELICS benchmark run on a specific date and time). These averages are scaled by the median of them. Then, we compute the average of these scaled values to get the CPU Benchmark Score.
    3. Suppose we have HELICS version 2.3.1 data from 2020-05-01. Suppose further the average available metrics’ values for echoBenchmark, ringBenchmark, and messageLookupBenchmark are 0.5, 1.1, 2.4, 1.3, 0.7, 0.9, and 3.3, respectively. The median of these values is 1.1. We scale these values by dividing them by the median to get 0.45, 1, 2.18, 1.18, 0.63, 0.81, and 3, respectively. Then the average of these values is 1.32, which is considered the CPU Benchmark Score.
    4. Formula:

where *b* = specific Benchmark.