Paraver Efficiencies Guide

1 Introduction

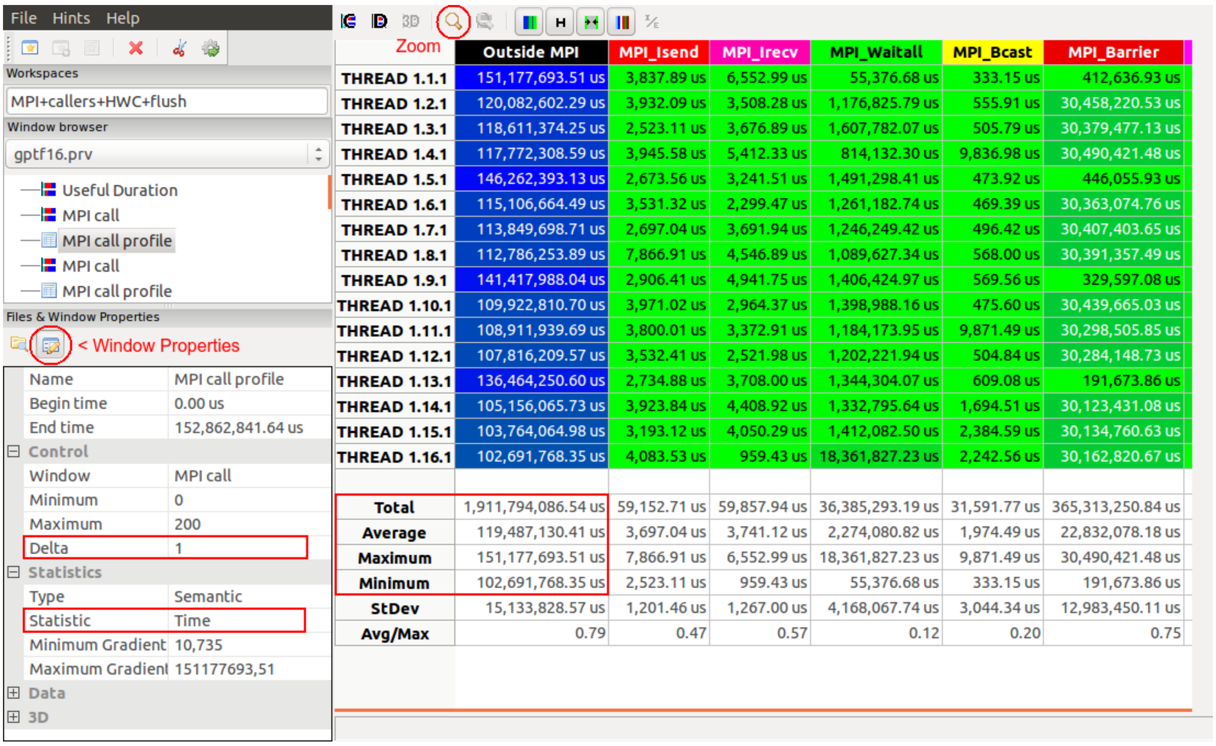
The Paraver Efficiencies Guide gives an introduction on how to obtain the required values for efficiencies and scaling for the POP Audits. Prerequisites are a basic knowledge of Paraver (and Dimemas) and the correctly recorded Paraver traces cut to the focus of analysis

2 Gather Necessary Values

You need the following values to compute all the efficiencies:

- Average computation time
- Maximum computation time
- Total computation time
- Runtime
- Total number of useful instructions
- Total number of useful cycles

Average, maximum, total computation time and runtime: use an MPI histogram and in the Paraver main window set *Window Properties → Statistics → Statistic* to *Time* and set the zoom level in the histogram window so you can see a table. Alternatively use *POP_MPI_Histogram.cfg* and you're good to go. The very left column is called *Outside MPI* aka computation without MPI. Scroll to the bottom of the table and you will find the values for *average, maximum*, and *total* computation time. After that, set *Window Properties → Control → Delta* to a value bigger or equal to *Maximum* (in the row above). This way, the table in the histogram should collapse to one single column. *Minimum, Maximum* and *Average* in that column should be equal and each of the values provides the *runtime*.



	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Bcast	MPI_Barrier
THREAD 1.1.1	151,177,693.51 us	3,837.89 us	6,552.99 us	55,376.68 us	333.15 us	412,636.93 us
THREAD 1.2.1	120,082,602.29 us	3,932.09 us	3,508.28 us	1,176,825.79 us	555.91 us	30,458,220.53 us
THREAD 1.3.1	118,611,374.25 us	2,523.11 us	3,676.89 us	1,607,782.07 us	505.79 us	30,379,477.13 us
THREAD 1.4.1	117,772,308.59 us	3,945.58 us	5,412.33 us	814,132.30 us	9,836.98 us	30,490,421.48 us
THREAD 1.5.1	146,262,393.13 us	2,673.56 us	3,241.51 us	1,491,298.41 us	473.92 us	446,055.93 us
THREAD 1.6.1	115,106,664.49 us	3,531.32 us	2,299.47 us	1,261,182.74 us	469.39 us	30,363,074.76 us
THREAD 1.7.1	113,849,698.71 us	2,697.04 us	3,691.94 us	1,246,249.42 us	496.42 us	30,407,403.65 us
THREAD 1.8.1	112,786,253.89 us	7,866.91 us	4,546.89 us	1,089,627.34 us	568.00 us	30,391,357.49 us
THREAD 1.9.1	141,417,988.04 us	2,906.41 us	4,941.75 us	1,406,424.97 us	569.56 us	329,597.08 us
THREAD 1.10.1	109,922,810.70 us	3,971.02 us	2,964.37 us	1,398,988.16 us	475.60 us	30,439,665.03 us
THREAD 1.11.1	108,911,939.69 us	3,800.01 us	3,372.91 us	1,184,173.95 us	9,871.49 us	30,298,505.85 us
THREAD 1.12.1	107,816,209.57 us	3,532.41 us	2,521.98 us	1,202,221.94 us	504.84 us	30,284,148.73 us
THREAD 1.13.1	136,464,250.60 us	2,734.88 us	3,708.00 us	1,344,304.07 us	609.08 us	191,673.86 us
THREAD 1.14.1	105,156,065.73 us	3,923.84 us	4,408.92 us	1,332,795.64 us	1,694.51 us	30,123,431.08 us
THREAD 1.15.1	103,764,064.98 us	3,193.12 us	4,050.29 us	1,412,082.50 us	2,384.59 us	30,134,760.63 us
THREAD 1.16.1	102,691,768.35 us	4,083.53 us	959.43 us	18,361,827.23 us	2,242.56 us	30,162,820.67 us
Total	1,911,794,086.54 us	59,152.71 us	59,857.94 us	36,385,293.19 us	31,591.77 us	365,313,250.84 us
Average	119,487,130.41 us	3,697.04 us	3,741.12 us	2,274,080.82 us	1,974.49 us	22,832,078.18 us
Maximum	151,177,693.51 us	7,866.91 us	6,552.99 us	18,361,827.23 us	9,871.49 us	30,490,421.48 us
Minimum	102,691,768.35 us	2,523.11 us	959.43 us	55,376.68 us	333.15 us	191,673.86 us
StDev	15,133,828.57 us	1,201.46 us	1,267.00 us	4,168,067.74 us	3,044.34 us	12,983,450.11 us
Avg/Max	0.79	0.47	0.57	0.12	0.20	0.75

Total number of useful instructions: use a histogram for useful instruction (e.g. *POP_Useful_Instructions.cfg*) and set *Window Properties* → *Statistics* → *Statistic* to *Sum bursts*. In addition, set *Window Properties* → *Control* → *Delta* to a value bigger or equal to maximum to reduce the table to a single column. At the bottom of the table, *Total* contains the total number of instructions.
Total number of useful cycles: use a histogram for useful cycles (e.g. *POP_Useful_Cycles.cfg*) and set *Window Properties* → *Statistics* → *Statistic* to *Sum bursts*. In addition, set *Window Properties* → *Control* → *Delta* to a value bigger or equal to maximum to reduce the table to a single column. At the bottom of the table, *Total* contains the total number of cycles.

3 Compute Efficiencies and Scalabilities

Based on the values above the efficiencies and scalabilities can be computed as follows:

1) Load Balance = Average / maximum computation time
2) Communication Efficiency = Maximum computation time / runtime
3) Parallel Efficiency = Load Balance * Communication Efficiency
4) Computation Scalability = …
   a) Strong scaling: Total computation time of reference[1] / total computation time
   b) Weak scaling: Total computation time of reference[1] / (total computation time / LIF[2])
5) Global Efficiency = Parallel Efficiency * Computation Scalability
6) IPC Scalability =
   (Total instructions / total cycles) / (total instructions of ref.[1] / total cycles of reference[1])
7) Instructions Scalability = …
   a) Strong scaling: Number of instruction of reference[1] / number of instructions
   b) Weak scaling: Number of instruction of reference[1] / (number of instructions / LIF[2])

[1)] reference = the smallest run recorded, i.e., 4, 5, 6, and 7 only make sense for multiple measurements with an increasing number of processes/threads.
[2)] LIF = load increase factor, i.e., by which factor was the total load increased. Usually this should be equal to the increase in processes/threads.

Please note: If everything is computed correctly, all values should be between 0% and 100%; except for measuring inaccuracy. In addition, all scalability values should stay constant (ideal) or decrease but not increase; except for measuring inaccuracy.

4 Detailed Communication Efficiency with Dimemas

Use Dimemas with an ideal network configuration. From the resulting trace get maximum computation time and runtime as described above and compute the remaining efficiencies:

8) Synchronization = Maximum computation time / Runtime of Dimemas trace
9) Transfer = Communication Efficiency / Synchronization