Laboration 5 - TDDC78

Jonathan Karlsson - jonka293 - 890201-1991
Niclas Olofsson - nicol271 - 900904-5338

May 22, 2013

1 TotalView

The debugger TotalView was pretty complicated to get started with, it required some special setup and such, but when it did start to work then we got a nice looking window with our code. It was pretty intuitive how to use it with clear buttons for single stepping and running the code. We have previously worked with the built in debugger in eclipse and found the interface a bit more crude in TotalView, but it had similar functionality. GDB is a debugger for C that is terminal based so TotalView is better since it has a graphical interface. We didn't get TotalView to work correctly when we had problems with the laborations but we imagine that it would have been a really helpful tool to have when searching for bugs so if we would start over with the lab series we would have spent some time on using TotalView from the beginning, just to see how the program interacts on the different nodes.

2 Tracing with ITAC

We compiled the program using ITAC, ran it with mpirun on four nodes, and analyzed the result with the Intel Trace Analyzer program. Overall, this was easier to do than to get TotalView up and running. On the other hand, it took some time to understand the user interface and how to view information for our own groups, not just for the default ones. The user interface does not work very well, at least not over X11, but it is usable for shorter usage sessions.

From the global timeline in Figure 1, we can see that it spends quite much time outside our defined groups in the beginning. This is due to creation of particles. Communication and synchronization is only done at the end of each timestep, and when using MPI_Reduce() at the end.

Figure 2 show total time spent per group and node. Collission check is the group that takes most time, which is understandable since it is the most work-heavy part of the program. Communication and synchronization takes little time compared to the rest. However, it is quite interesting that we spend almost as much time waiting for synchronization as we spend on sending actual data.

Figure 3 shows a counter of the total momentum for each node, representing the pressure. We see that the pressure is much higher on two of our four nodes. This is understandable, since the nodes in the end of the box has walls on three sides, and the other nodes just has two, shorter walls.

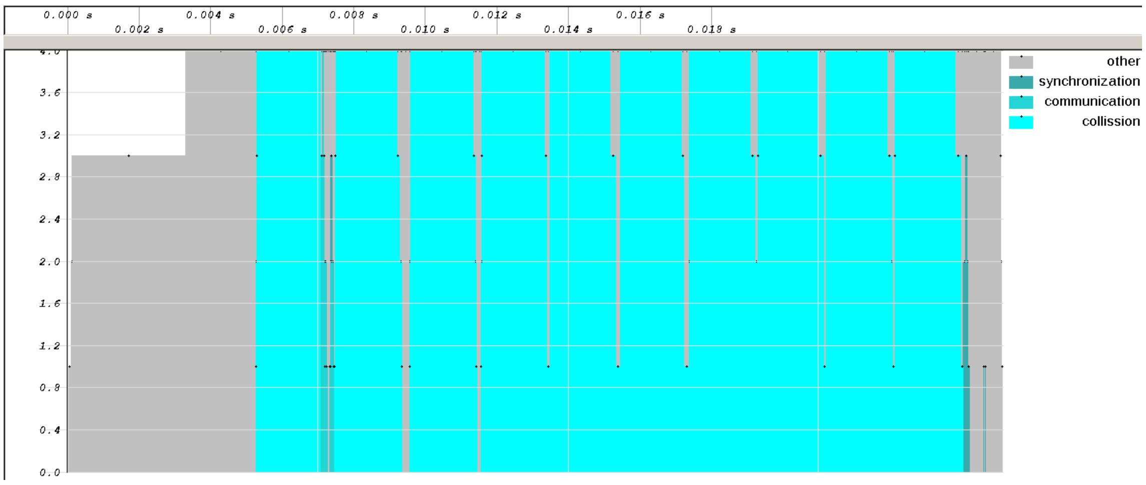
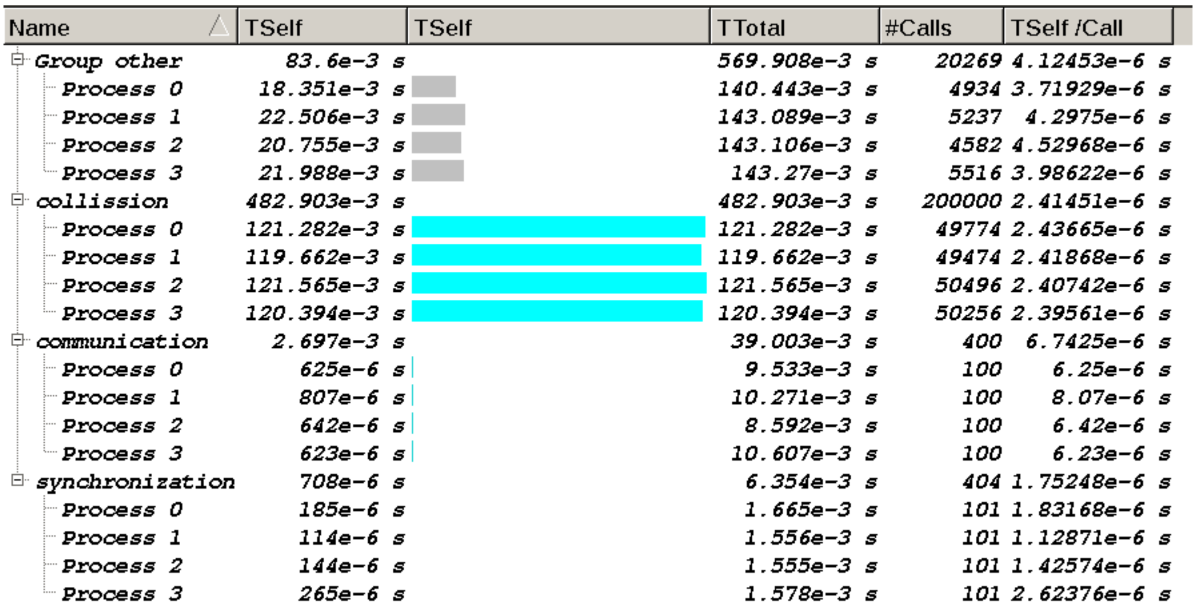


Figure 1: Global timeline showing usage of the different groups over time



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group other	83.6e-3 s		569.908e-3 s	20269	4.12453e-6 s
Process 0	18.351e-3 s		140.443e-3 s	4934	3.71929e-6 s
Process 1	22.506e-3 s		143.089e-3 s	5237	4.2975e-6 s
Process 2	20.755e-3 s		143.106e-3 s	4582	4.52968e-6 s
Process 3	21.988e-3 s		143.27e-3 s	5516	3.98622e-6 s
collission	482.903e-3 s		482.903e-3 s	200000	2.41451e-6 s
Process 0	121.282e-3 s		121.282e-3 s	49774	2.43665e-6 s
Process 1	119.662e-3 s		119.662e-3 s	49474	2.41868e-6 s
Process 2	121.565e-3 s		121.565e-3 s	50496	2.40742e-6 s
Process 3	120.394e-3 s		120.394e-3 s	50256	2.39561e-6 s
communication	2.697e-3 s		39.003e-3 s	400	6.7425e-6 s
Process 0	625e-6 s		9.533e-3 s	100	6.25e-6 s
Process 1	807e-6 s		10.271e-3 s	100	8.07e-6 s
Process 2	642e-6 s		8.592e-3 s	100	6.42e-6 s
Process 3	623e-6 s		10.607e-3 s	100	6.23e-6 s
synchronization	708e-6 s		6.354e-3 s	404	1.75248e-6 s
Process 0	185e-6 s		1.665e-3 s	101	1.83168e-6 s
Process 1	114e-6 s		1.556e-3 s	101	1.12871e-6 s
Process 2	144e-6 s		1.555e-3 s	101	1.42574e-6 s
Process 3	265e-6 s		1.578e-3 s	101	2.62376e-6 s

Figure 2: Time spent in the different groups, for each node

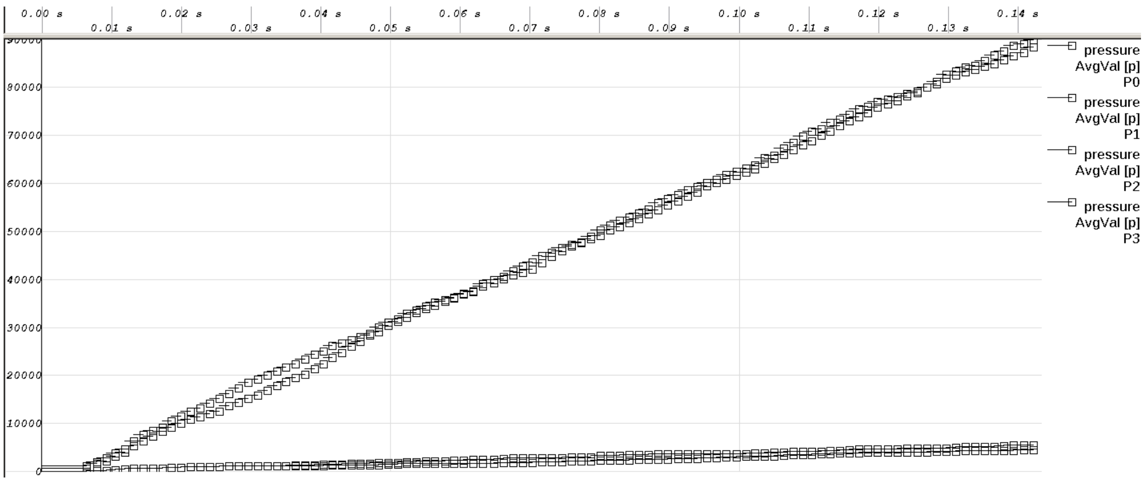


Figure 3: Count graph of the total momentum for each node