Evaluation of Lulesh with SMGuard

Please refer to Section 6.1 of SMGuard paper for the detail information about experimental environment.

1 Kernels in Lulesh

Table 1: Kernels of Lulesh.(Problem size=100×100×100)

Kernel	Abbreviation	Grid size	Block size	Single duration(ms)	Overall running time(ms)	Running time ratio
AddNodeForcesFromElems_kernel	Kernel1	16099	64	1.962	1991.092	6.14%
CalcVolumeForceForElems_kernel	Kernel2	15625	64	14.808	15030.340	46.31%
CalcKinematicsAndMonotonicQGradient_kernel	Kernel3	15625	64	10.004	10154.035	31.29%
CalcAccelerationForNodes_kernel	Kernel4	8050	128	0.387	392.838	1.21%
CalcPositionAndVelocityForNodes_kernel	Kernel5	8050	128	0.839	851.445	2.62%
CalcMonotonicQRegionForElems_kernel	Kernel6	7813	128	1.256	1275.270	3.93%
ApplyMaterialPropertiesAndUpdateVolume_kernel	Kernel7	7813	128	2.309	2343.984	7.22%
CalcTimeConstraintsForElems_kernel	Kernel8	1024	128	0.402	408.127	1.26%
ApplyAccelerationBoundaryConditionsForNodes_kernel	Kernel9	80	128	0.002	2.025	0.01%
CalcMinDtOneBlock	Kernel10	2	1024	0.004	4.399	0.01%

Table 1 shows the information of kernels in our evaluated Lulesh. There are 10 CUDA kernels used in Lulesh, and the total running time ratios of each kernel are shown in Table 1. Kernels *CalcVolumeForceForElems_kernel* and *CalcKinematicsAndMonotonicQGradient_kernel* occupy most of overall kernel running time.

2 Performance Improvement

Figure 1 shows the performance improvement for Lulesh over MPS with different number of reserved CapSM when corunning lulesh with different applications. When the reserved number of CapSM is 0 it means the use of GPU resource of co-location pairs is not restricted, which is actually the default MPS mechanism. When the reserved number of CapSM is 13 it means running Lulesh standalone. From the results in Figure 1, we can see that when the GPU resource is reserved, performance of Lulesh can have a great improvement in co-location scenario. And the more GPU resource is reserved for Lulesh, the better the performance. From the results we can see that SMGuard is effective for real applications like Lulesh.

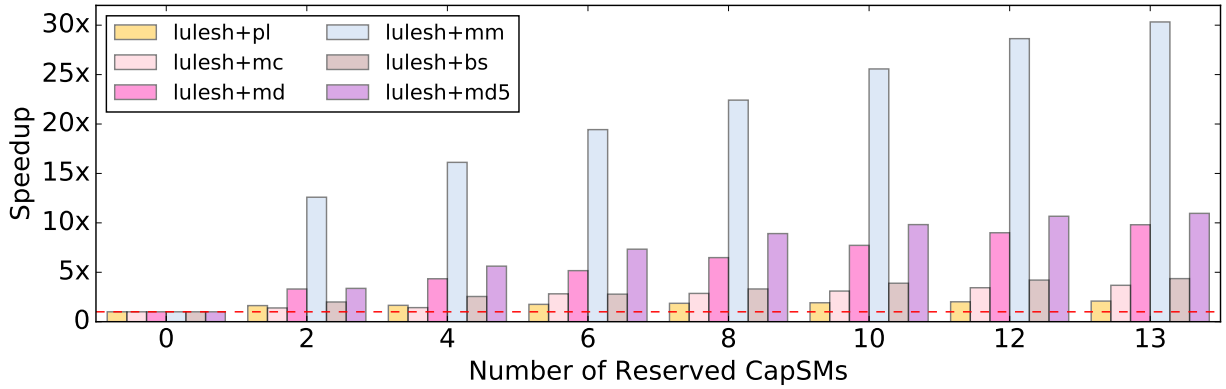

Figure 1: Performance improvement for Lulesh over MPS with different number of reserved CapSM when co-running lulesh with different applications.0 reserved CapSM means the use of GPU resource of co-location pairs is not restricted; 13 reserved CapSMs means running lulesh standalone.

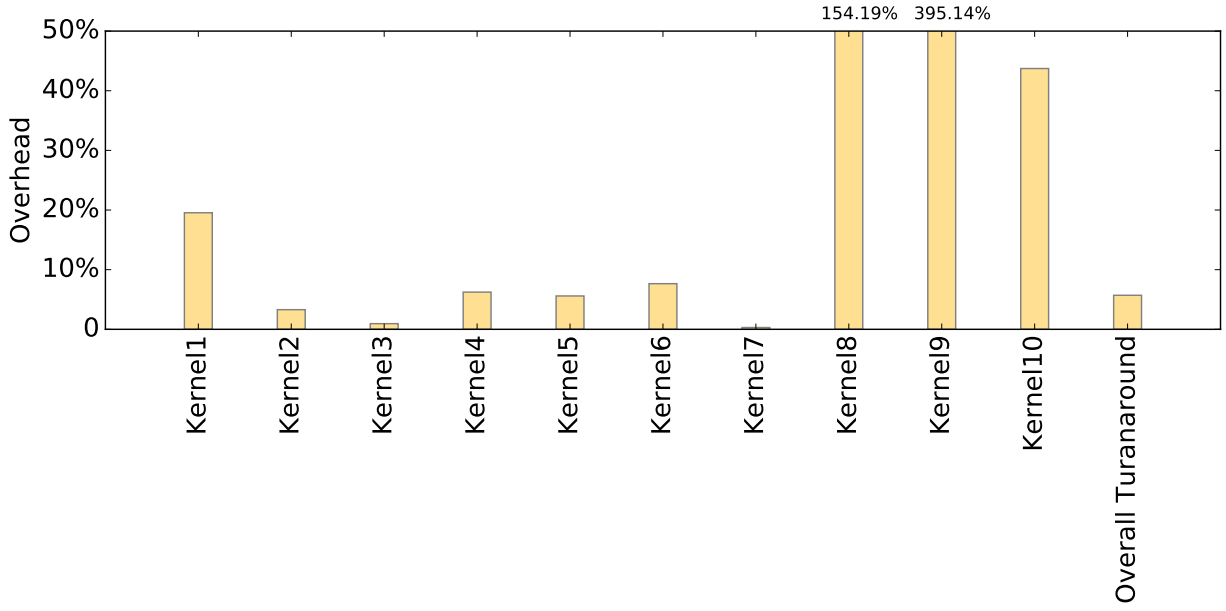

Figure 2: Introduced overhead of execution time for different kernels and overall turnaround time of lulesh over baseline.

3 Introduced Overhead

To evaluate the overhead introduced by SMGuard after Lulesh is transformed. we run original and transformed versions of Lulesh in standalone mode respectively. We collect the running time of each kernel in Lulesh and the overall turnaround time of all the launched kernels, then we calculate the ratio between increased running time and baseline as the introduced overhead . The results are shown in Figure 2. We can see that the introduced overhead of most kernels of Lulesh is less than 10%. Although the introduced overhead of kernels *CalcTimeConstraintsForElems_kernel* and *ApplyAccelerationBoundaryConditionsForNodes_kernel* is very serious, which can be as high as 395.14%, it has negligible impact on the overall turnaround time because of the very small running time ratio of kernels *CalcTimeConstraintsForElems_kernel* and *ApplyAccelerationBoundaryConditionsForNodes_kernel* as shown in Table 1. The introduced overhead of the overall turnaround time of Lulesh is only about 5.70%.