Contents

List of Figures

List of Tables

5 Results

The data collected as part of the evaluation of the introduced trigger strategies is presented and discussed in this section. The hardware and software setup are given in Section 5.1, as to allow for reproducibility of our results. Sections 5.2 and 5.3 discuss some implementation choices based on collected data. The main benchmark results including achieved speedups and default trigger parameters are presented in Section 5.4, grouped by scenario. Finally, Section 5.5 shows statistics collected through the `LiveInfo` system, as to motivate hybrid triggers.

5.1 Experimental Setup

The measurements collected for analysis were obtained on the CoolMUC4 Linux-Cluster of the Leibniz-Rechenzentrum[1]. The nodes in the `cm4` cluster consist of processors in the Sapphire Rapids family (Intel® Xeon® Platinum 8480+) with 2.1 GiB of memory per logical CPU and 488 GiB per node [1]. For benchmarking purposes, the AutoPas library and `md-flexible` were compiled with Spack GCC 13.2.0 and Intel MPI 2021.12.0 on commit `bc47d1ea7e8598afcf58bd35fc531439aa0c7dda`.

The scripts used to generate the Slurm jobs and configuration files can be found in the repository of this thesis[2].

5.2 Choice of Simulation Statistics

As referred to before in Section 3.1.2, all trigger strategies are based on the iteration runtimes excluding rebuild times. This choice can be justified by inspecting runtime data we collected: As shown in Figure 5.1, the rebuild times change little over all iterations simulated with a particular configuration. Their inclusion therefore does not provide any new information, but rather smooths out the overall measurements and thus decreases the effectivity at which a scenario change can be detected.

Additionally, rebuild iterations only happen at `rebuild-frequency`. This can lead to stability problems in trigger strategies used with a low number of samples, as the few

[1] `https://www.lrz.de/`
[2] `https://github.com/ladnik/bachelor-thesis`

rebuild iterations greatly outweigh all non-rebuild iterations.

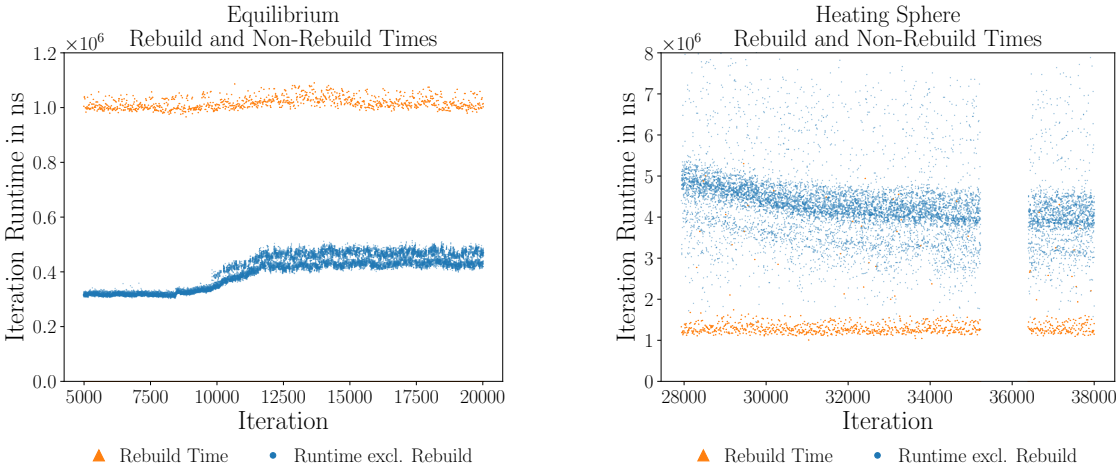


Figure 5.1: Rebuild and non-rebuild times in the equilibrium (left) and heating-sphere (right) scenario. The configurations used were `VLC-C08-N3L-AoS-CSF1` and `LC-C04-NoN3L-AoS-CSF1` respectively. The rebuild times do not contribute any new information regarding scenario change.

5.3 Computational Overhead

The implemented tuning strategies analyze data at runtime and therefore need additional computations in each iteration. The performance impact of these should be negligible in comparison to the simulation steps, as otherwise any performance gains due to fewer tuning phases are nullified. To quantify the overhead our strategies introduce, runs without any tuning iterations are compared, such that any changes in runtime that may occur due to different tuning phase initiation points are removed. This is achieved by using a single predefined configuration, such that no tuning takes place.

Figure 5.2 (left) shows the overhead obtained in that manner for the heating-sphere scenario using `LC-C04-N3L-AoS-CSF1` and $n = 500$. To better illustrate the measurements, all values are given as relative and absolute increase in average baseline runtime per iteration. At most, an overhead of $1.9\,\%$ per iteration is seen, which can be considered insignificant. Note however, that the overhead is an absolute increase; depending on the scenario, time spent computing interactions varies, therefore the relative values change. It should also be considered, that the absolute difference between the static baseline and the dynamic runs lay in the range of 1 to $7\,\mathrm{s}$ over the complete run. Hardware heterogeneity might make up a significant part of this difference — which would directly influence the measured relative and absolute overhead. Nonetheless, the results may give some indication on which strategies are more compute-intensive than others. In particular, usage of the `TimeBasedSimpleTrigger` does incur nearly no runtime penalty ($0.084\,\%$), whereas the more complex trigger strategies have a correspondingly larger impact.

To exemplify the importance of optimizing the trigger routines, Figure 5.2 (right) illustrates the runtime differences between naive and optimized triggers in the equilibrium scenario. The naive version recalculates the average over all samples each iteration, whereas the optimized version uses a ring buffer and running summation to reduce computational cost. Note that, particularly in the averaging trigger, the speedup ex-

perienced is not only due to a lowering of computational overhead, but also due to a lower number of tuning iterations. This can be explained by the feedback of the trigger strategies, as they also influence iteration runtime.

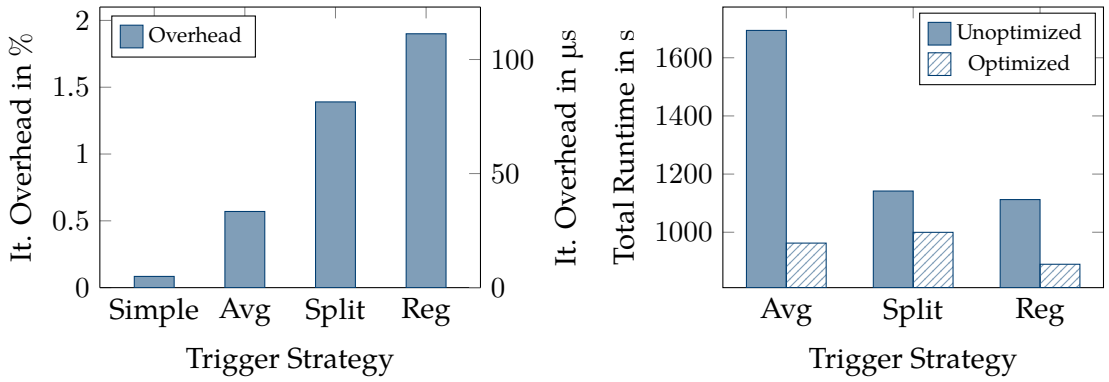


Figure 5.2: Performance comparisons between the various trigger strategies: Relative and absolute iteration overhead in the heating-sphere scenario (left) and average runtime decrease obtained through optimizations in the equilibrium scenario (right).

5.4 Benchmarking Results

The relative speedups presented in the following line charts were computed by the formula given in (5.1), where t_{baseline} represents the runtime with tuning phases at fixed intervals and t_{dynamic} the runtime of our implementation.

$$S = \frac{t_{\text{baseline}}}{t_{\text{dynamic}}} - 1 \tag{5.1}$$

For all plots showing the selected configurations for a given run, the blue scatter dots represent the runtime of that particular iteration. The colored background identifies the used configuration: same configurations map to the same color in a given plot. Gaps along the x-axis occur where tuning iterations have been logged — as their runtime is not relevant for our purposes and would distort the actual runtime plot, they are not reported here. The gray dashed lines indicate the start of a new tuning phase.

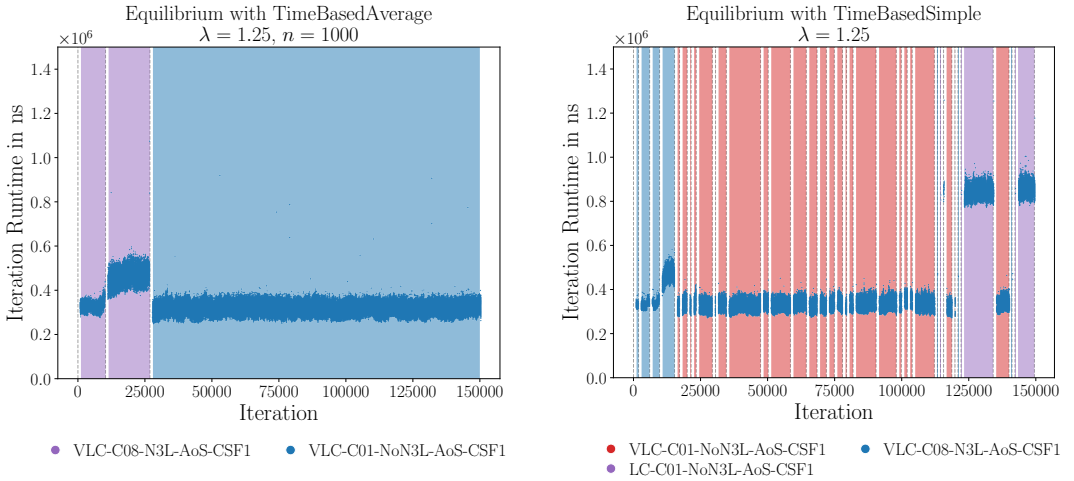
5.4.1 Equilibrium

Selected Runs

Figure 5.3 shows two of the experimental runs in detail. On the left hand side, a `TimeBasedAverageTrigger` with $\lambda = 1.25$, $n = 1000$ reliably detects scenario change. Two tuning phases are started in the initial phase, where overall iteration runtime increases. After the second tuning phase, a better configuration is found. As there is no further indication that the simulation state changes, the remaining iterations are performed using the same configuration. As was explained in Section 4.1.1, it is indeed the case that no further configuration change is needed. In this run therefore, the presented trigger was beneficial.

On the right hand side, a worst-case outcome is shown. The `TimeBasedSimpleTrigger` used in that run triggered too many new tuning phases, which lead to an increase in total simulation runtime compared to the baseline run. The main reason for the

overreaction lies in the implementation of the trigger: as long as the runtime of one iteration is greater than the one of its predecessor by a factor of λ or more, a tuning phase is initiated. The few outliers in the equilibrium scenario that were averaged out in the previous example, are detrimental to this trigger strategy.



(a) Good scenario change detection. (b) Too many unnecessary tuning phases.

Figure 5.3: Examples of trigger behavior in the equilibrium scenario.

Speedup and Default Parameters

As can be seen in Figure 5.4, a trigger factor of $\lambda = 1.5$ leads to increased speedup compared to $\lambda = 1.25$ in the majority of trigger strategies. This is however mainly due to the nature of the equilibrium scenario: After the initial relaxation, the optimal configuration is not expected to change. Therefore, not initiating any new tuning phases will lead to a decrease in total simulation runtime, which is why larger values for λ perform better, as they reduce trigger sensitivity. A factor chosen too large however, diminishes this effect again, as the changes in the initial phase are not accounted for. That the speedup is indeed a result of the decreased number of tuning iterations can be verified in the right-hand side plots; for the baseline run with static tuning intervals, $22.6\,\%$ of iterations were spent in tuning phases. Conversely, for strategies initiating too many new tuning intervals, i.e., with more tuning iterations than the baseline run, the simulation runtime increases.

Additionally, triggers with a larger sample size will typically trigger less frequently, as more of the variability in iteration runtime is smoothed out. However, for a too large number of samples such as $n > 1000$, the speedup may decrease, as the computational overhead is directly proportional to the number of samples. This is particularly noteworthy for scenarios with fewer particles, where the computation of interactions takes shorter time. Therefore, the relative overhead of our strategies is larger, which would explain the lower speedup seen in the regression triggers: Compared to the `TimeBasedAverageTrigger`, at roughly the same number of tuning iterations, the speedup achieved is significantly lower.

Interestingly, the `TimeBasedSplitTrigger` with $n = 250$ triggered significantly more tuning iterations for $\lambda = 1.75$ than for $\lambda = 1.5$. The rather small sample size increases the sensitivity towards noise in the input data — which is why outside influences by the underlying hardware could explain the difference between the two runs. This holds

particularly if one considers that our equilibrium scenario consists of few particles with which interactions must be computed, resulting in low iteration runtimes.

The collected data suggests default parameters as presented in Table 5.1.

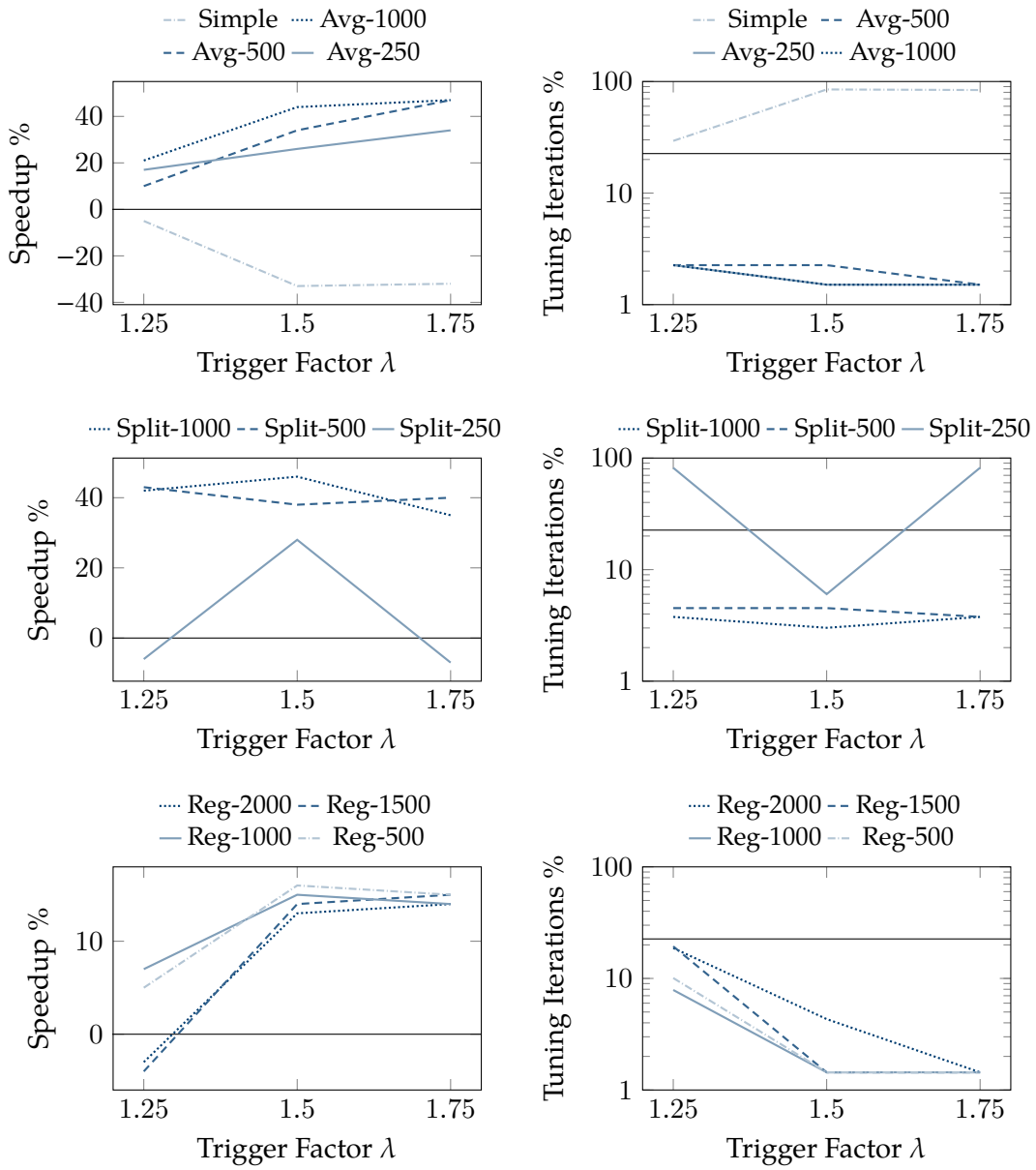


Figure 5.4: Trigger behavior in the equilibrium scenario, the numbers in the legends refer to the number of samples n considered. The line in the background represents the baseline run. Note the logarithmic scale in the plots on the right hand side.

Optimality

Our second evaluation metric, as stated in Section 4.2, concerns the quality of the chosen configurations. For efficient computation, we expect the configuration at any non-tuning iteration to be one of the best choices. As can be seen in Figure 5.5, this is achieved across all strategies except the simple trigger, with all configurations being one of the top three choices for that specific iteration. It should be noted, that the ranking of optimal configurations is only an approximation, as it is based on the baseline

Trigger	Trigger factor λ	Number of samples n
`TimeBasedSimple`	not recommended	–
`TimeBasedAverage`	1.75	500
`TimeBasedSplit`	1.5	1000
`TimeBasedRegression`	1.5	500

Table 5.1: Suggested default parameters for the equilibrium scenario.

run and therefore restricted to the resolution of that run's `tuning-interval`. The best performing strategy regarding optimality appears to be the `TimeBasedAverageTrigger`, for which 92 % of all non-tuning iterations were computed using the same configuration as in the baseline run. Except for the `TimeBasedSimpleTrigger`, all strategies computed the full simulation length with configurations among the top 3 choices of the static run.

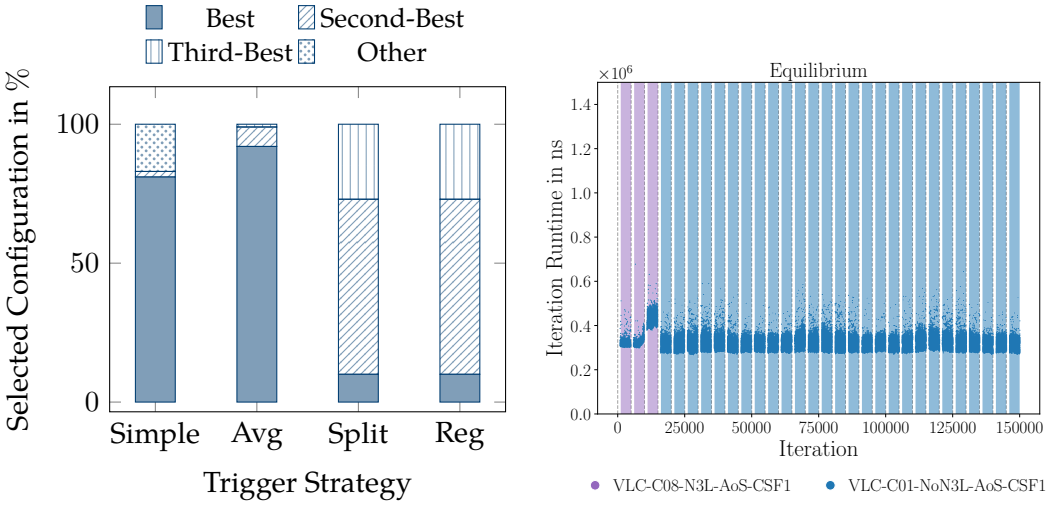


Figure 5.5: Ranking of configurations selected by the best run in the equilibrium scenario for each trigger strategy (left) and selected configurations in the baseline run (right).

5.4.2 Exploding Liquid

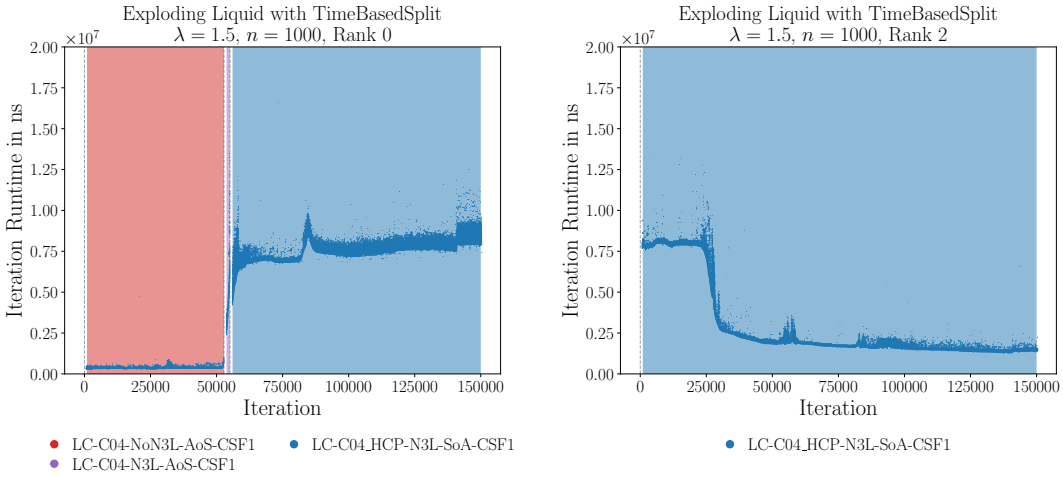
Selected Runs

The exploding-liquid scenario was executed on 6 MPI processes, with the simulation domain subdivided along the y-axis. As a result, different ranks encounter the "particle wave" at different points in time, as it spreads outward from the center (cf. Section 4.1.2). This effect can be seen in Figure 5.6, where the plot showing rank 2 (in the center of the domain, plot on the right hand side) experiences high iteration runtimes at the beginning of the simulation. Rank 0 (edge of the domain, plot on the left hand side), however, shows this influx of particles not until iteration 50 000.

The figure on the left shows an optimal response to the peak in iteration runtime by the `TimeBasedSplitTrigger`. The initial configuration remains optimal until the particles enter this part of the domain, after which two tuning phases are triggered. The first trigger at iteration 52 814 is not needed, as due to the rapid increase in iteration runtime, a second one at iteration 55 088 is tripped. This additional tuning phase could be prevented by using a bigger factor λ, the effect on total simulation runtime would how-

ever be limited. The second half of the simulation again has one optimal configuration, which is well reflected in the trigger's behavior.

The second plot on the right does not necessarily illustrate bad trigger behavior, but rather a limitation of our approach. After the initial expansion, the subdomain of rank 2 significantly transforms at iteration 25 000, with a different configuration possibly being better suited. However, our triggers only consider increases in input data. A more appropriate method in such cases would be to consider changes in magnitude (cf. Section 3.1.3), i.e., any deviation from the current average.



(a) Good scenario change detection. (b) No reconfiguration after peak.

Figure 5.6: Examples of trigger behavior in the exploding-liquid scenario.

Speedup and Default Parameters

Although the exploding-liquid scenario is more complex than the equilibrium scenario, there is still a performance gain in all trigger strategies: The highest speedup is reached by the `TimeBasedAverageTrigger`, with a reduction in total simulation time of 20 %. This is less than was reached in the equilibrium scenario, but still significant. In general, the results look very similar to those of the previous scenario, with some notable exceptions. For example, the `TimeBasedSimpleTrigger` reaches positive speedup for larger values of λ. Considering that in most ranks, the configuration change happens at the initial inflow of particles into the subdomain, as simple trigger is sufficient. With $\lambda = 1.25$ however, the trigger appears to be too sensitive, which again leads to an excessive number of tuning phases.

The `TimeBasedSplitTrigger` displays opposite behavior; for larger values of λ, the speedup is greater, as triggering too few tuning phases leads to a large number of iterations being computed using a suboptimal configuration. Similarly, triggers with a larger sample size tend to smooth out the changes in iteration runtime and thus have a delayed reaction to the rapid transformation of the domain. The `TimeBasedRegressionTrigger` again leads to lower speedups than averaging or split triggers at optimal configuration. However, the difference is minor and can thus be attributed to the higher overhead of this strategy.

The collected data suggests default parameters as presented in Table 5.2.

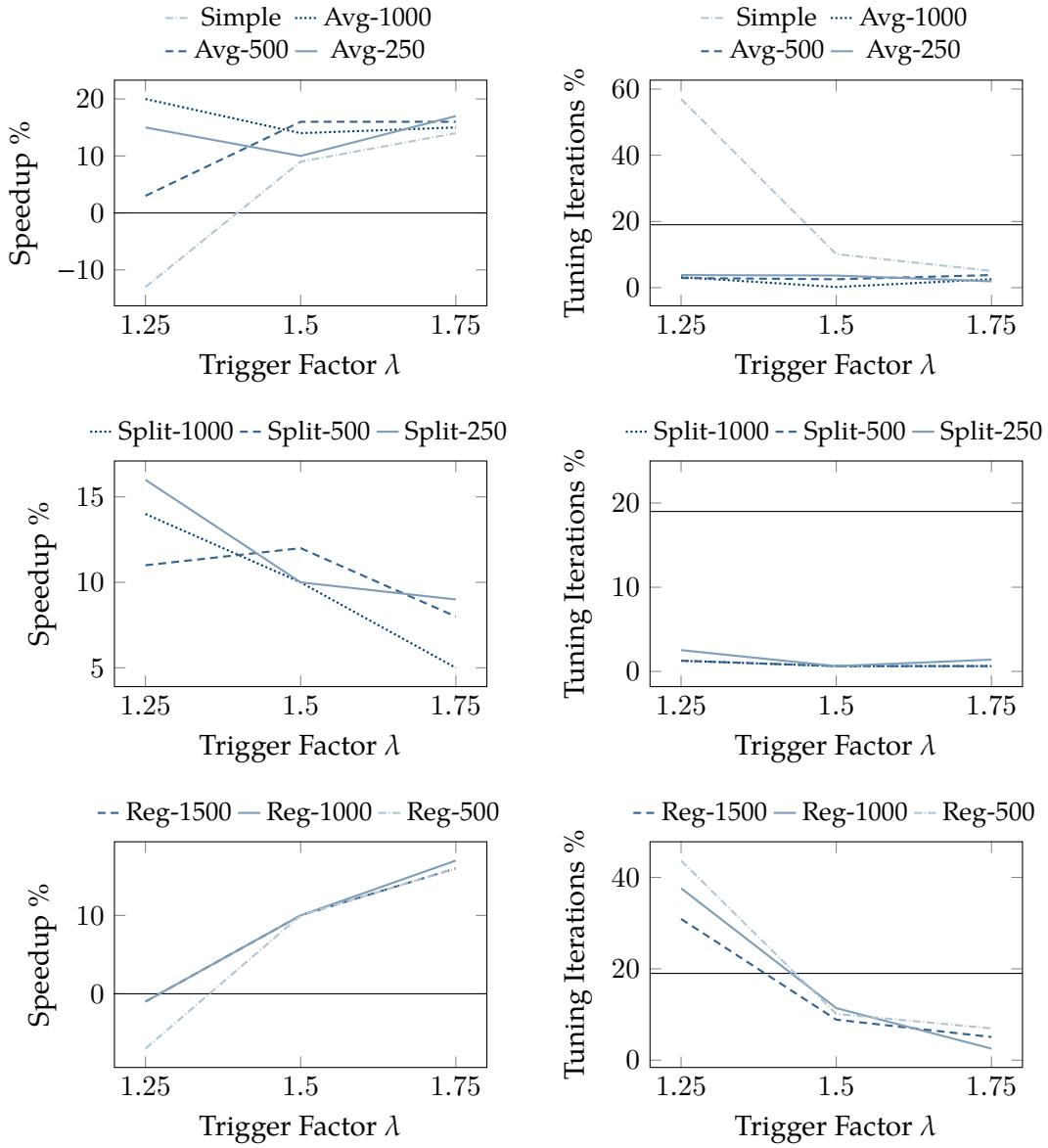


Figure 5.7: Trigger behavior in the exploding-liquid scenario, the numbers in the legends refer to the number of samples n considered. The line in the background represents the baseline run.

Trigger	Trigger factor λ	Number of samples n
`TimeBasedSimple`	1.75	–
`TimeBasedAverage`	1.25	1000
`TimeBasedSplit`	1.25	250
`TimeBasedRegression`	1.75	1000

Table 5.2: Suggested default parameters for the exploding-liquid scenario.

Optimality

The configuration fit is best for the `TimeBasedSimpleTrigger`, `TimeBasedAverageTrigger` and `TimeBasedRegressionTrigger`: As shown in Figure 5.8, these strategies select, on average, the best configuration for 95 % of all non-tuning iterations. The strategy that appears to have the worst fit is the `TimeBasedSplitTrigger` — note, however, that it used the fourth best configuration in 81 % of iterations.

It should also be mentioned, that these statistics are based on rank 0 across all runs. Therefore, the problem of missing reconfiguration after a drop in iteration runtime, as discussed previously, is not reflected in the results.

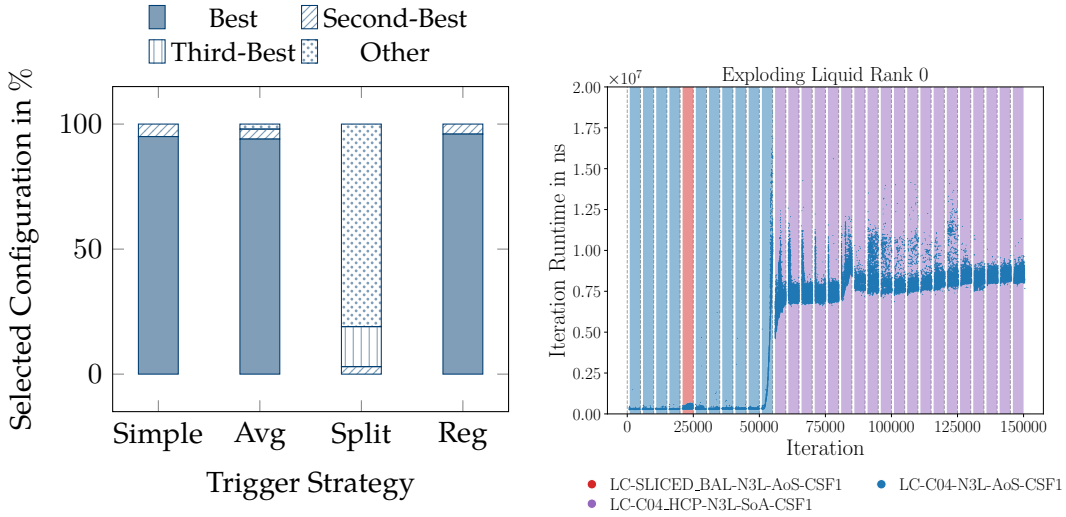


Figure 5.8: Ranking of configurations selected by the best run in the exploding-liquid scenario for each trigger strategy (left) and selected configurations in the baseline run (right).

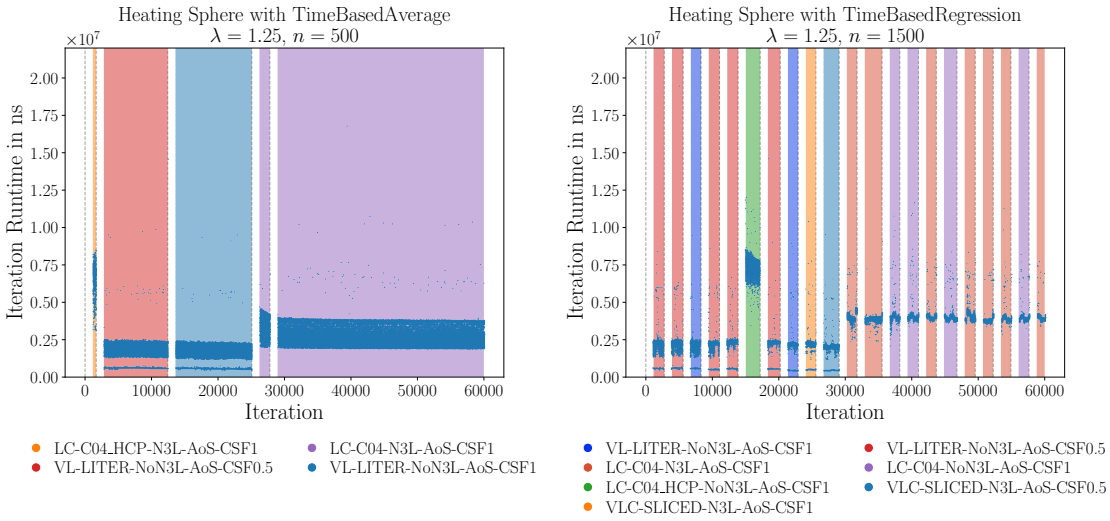
5.4.3 Heating Sphere

Selected Runs

Figure 5.9 again displays two sample runs to illustrate optimal and unsatisfactory results. The left figure shows a reduction in the number of tuning phases initiated by the `TimeBasedAverageTrigger`. After the last tuning phase, the configuration does not change from the previous one — that tuning phase was therefore unnecessary. Compared to the equilibrium scenario, the iteration runtimes form a broad band, which indicates a high variance. The absolute ranges of these variations lie in the range of 1×10^6 to 5×10^6 ns, in contrast to a spread of 1×10^5 ns in the equilibrium scenario. Additionally, more outliers are seen, which might worsen the performance of the strategies susceptible to oscillations; one of them is the `TimeBasedAverageTrigger` depicted. Without any clear indication of an increase in runtime, new tuning phases are initiated, since a single outlier larger than the last n samples by a factor of λ can trigger the strategy. Interestingly, most outliers do not have a significant impact due to the averaging approach. As shown in the plot, the use of the averaging trigger results in fewer tuning phases than in the baseline run, which in turn explains the speedup measured.

Worse results can be seen in the `TimeBasedRegressionTrigger`, pictured on the right hand side. After multiple tuning phases, the same configuration is selected, which again indicates unnecessary triggering. The regression approach is suboptimal in the

heating-sphere scenario, due to the high variance in iteration runtimes described before: Outliers can skew the slope derived by the least-squares method, which leads to a incorrect prediction of the runtime in the next interval, triggering our strategy. Hence, a slowdown compared to the baseline run is observed. This suggests that the regression based trigger may not be used in scenarios that behave similarly. To address this issue, the linear least squares estimation in the regression approach could be replaced with a more robust method like the Theil-Sen estimator [3].



(a) Reduced number of tuning phases. (b) Too many unnecessary tuning phases.

Figure 5.9: Examples of trigger behavior in the heating-sphere scenario.

Speedup and Default Parameters

The high variance in iteration runtimes directly influences the behavior of our trigger strategies, as shown in Figure 5.10. The strategies which should be unstable due to these variations are the `TimeBasedAverageTrigger`, `TimeBasedSimpleTrigger` and `TimeBasedRegressionTrigger`. However, results only show poor performance of the latter two. The simple trigger under-performs the static approach by up to $-41\,\%$ with a highly unfavorable $99\,\%$ of all iterations being tuning iterations. Similarly, the regression approach leads to runtime increases across almost all tested combinations of (λ, n).

The `TimeBasedAverageTrigger`, however, performs exceptionally well, with speedups of up to $40\,\%$, proportional to the decreased number of tuning iterations. Rather counterintuitively, this outcome could be accounted for precisely by the prevalence of outliers: If there occur enough outliers within the trigger's sample interval, they raise the average runtime, which in turn reduces the impact of any current outlier.

In summary, it can be said that our approach is not suitable for the heating-sphere scenario. Considering that there is no clear indication of scenario change in runtime (cf. Section 5.5), this was expected.

The collected data suggests default parameters as presented in Table 5.3.

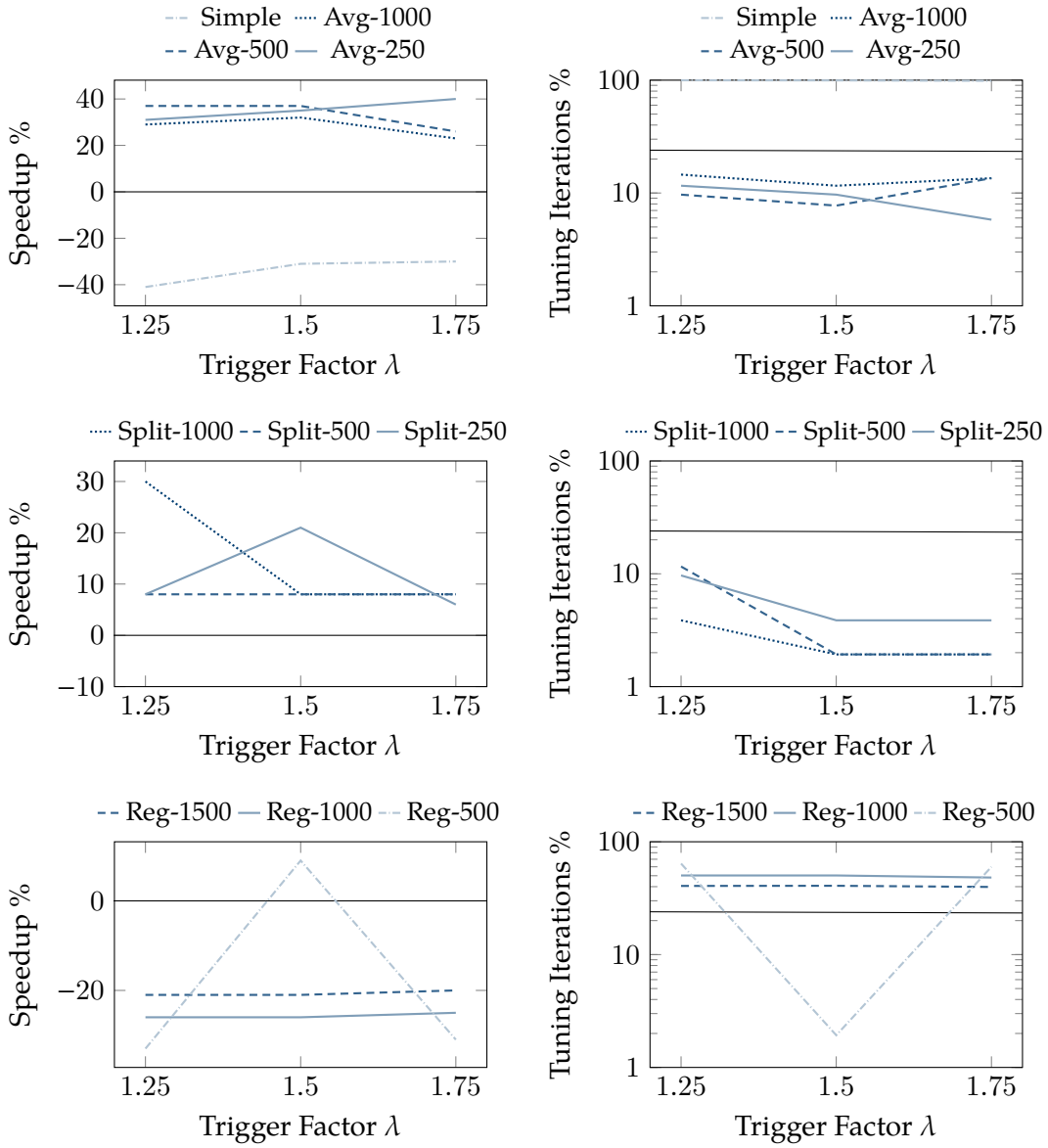


Figure 5.10: Trigger behavior in the heating-sphere scenario, the numbers in the legends refer to the number of samples n considered. The line in the background represents the baseline run. Note the logarithmic scale in the plots on the right hand side.

Trigger	**Trigger factor λ**	**Number of samples n**
`TimeBasedSimple`	not recommended	–
`TimeBasedAverage`	1.5	500
`TimeBasedSplit`	1.5	250
`TimeBasedRegression`	not recommended	–

Table 5.3: Suggested default parameters for the heating-sphere scenario.

Optimality

Figure 5.11 shows the configuration selected by the best performing run for each trigger. The values again are given for all non-tuning iterations, compared to the configurations selected in the baseline run. Note that, e.g., the simple trigger displays better configuration fit than the averaging trigger; this is primarily due to tuning iterations being ignored. Additionally, not triggering a new tuning phase has a higher runtime impact than computing iterations with suboptimal configuration fit.

It can be seen that, particularly for the strategies triggering fewer tuning phases, i.e., the `TimeBasedAverageTrigger` and `TimeBasedSplitTrigger`, the configuration fit is suboptimal. For both, on average 25 % of all non-tuning iterations were computed using a configuration that was not in the top 3 choices of the baseline run.

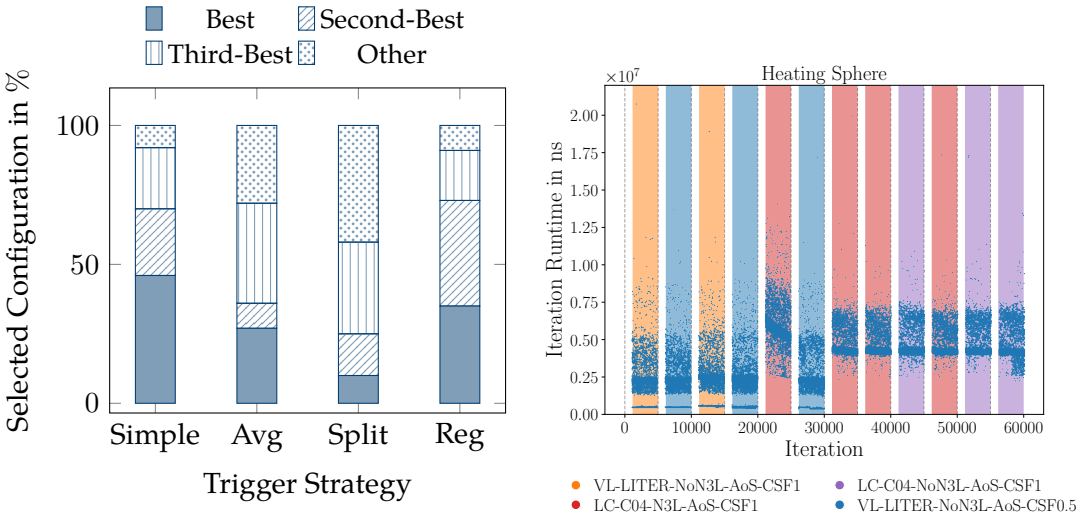


Figure 5.11: Ranking of configurations selected by the best run in the heating-sphere scenario for each trigger strategy (left) and selected configurations in the baseline run (right).

5.5 Hybrid Triggers

Time-based approaches may not be suitable for scenarios in which iteration runtime alone is not good enough indicator for scenario change. This was seen, e.g., in the heating-sphere scenario. Since AutoPas provides additional live simulation statistics through its `LiveInfo` interface, these could be used in combination with iteration runtimes to find better strategies in detecting scenario change. As a motivation of this approach, Figure 5.12 shows an exemplary run for the heating-sphere scenario. Using static containers, the initial optimal configuration is `VL-List_Iter-NoN3L-AoS`, and changes to `LC-C04-N3L-AoS-CSF1` later on [2]. Although a better configuration is available, the iteration runtimes do not change. However, the `maxDensity` statistic indicates the shift towards a different simulation state; considering that the particles are packed closely in the initial phase and expand outwards, this decrease in particle density is consistent with expectations. The `maxDensity` statistic would therefore be a reasonable trigger input.

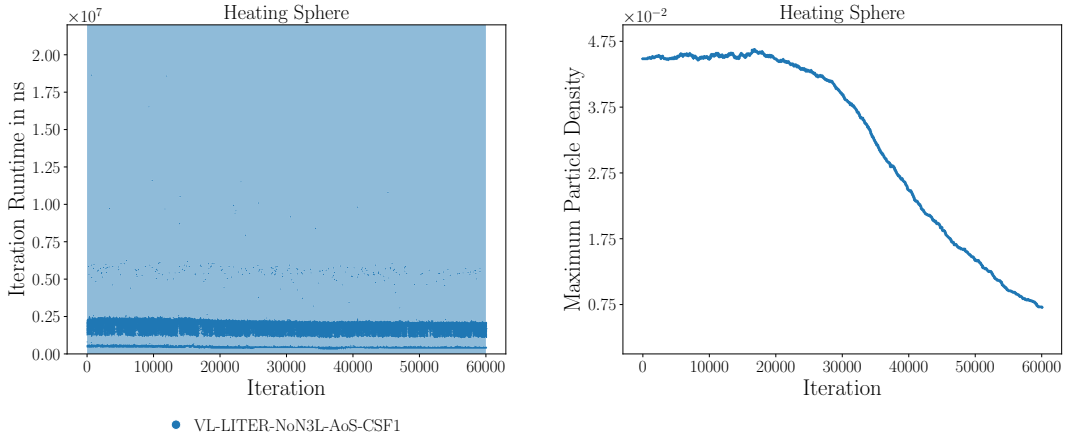


Figure 5.12: Iteration runtime (left) and the maximum particle density (right) for the heating-sphere scenario with single configuration `VL-List_Iter-NoN3L-AoS`. The iteration runtime does not indicate scenario change, but the `maxDensity` statistic shows the transformation of the simulation state.

Bibliography

[1] Leibniz Supercomputing Centre. *Job Processing on the Linux-Cluster*. Accessed: 2025-09-01. 2025. URL: https://doku.lrz.de/job-processing-on-the-linux-cluster-10745970.html.

[2] Samuel James Newcome et al. "Algorithm Selection in Short-Range Molecular Dynamics Simulations". In: (May 2025). DOI: 10.48550/ARXIV.2505.03438. arXiv: 2505.03438 [cs.CE].

[3] Rand Wilcox. "Chapter 10 - Robust Regression". In: *Introduction to Robust Estimation and Hypothesis Testing (Third Edition)*. Ed. by Rand Wilcox. Third Edition. Statistical Modeling and Decision Science. Boston: Academic Press, 2012, pp. 471–532. ISBN: 978-0-12-386983-8. DOI: https://doi.org/10.1016/B978-0-12-386983-8.00010-X.