Kintecus OpenMP Parallel Scaling Analysis

19 Chemical Mechanisms | Serial to 8 Threads | 2018 Alienware Laptop

February/March 2026

1. Executive Summary

This report presents the results of OpenMP parallel scaling tests conducted on 19 chemical reaction mechanisms using Kintecus. Thread counts of 1 (serial), 2, 3, 4, and 8 were evaluated. The results reveal a clear bifurcation between large and small mechanisms in terms of parallel efficiency.

Key findings:

- Large mechanisms (2,000+ species) show consistent, monotonic speedup with increasing thread count.
- Small mechanisms (<500 species) show no benefit from parallelization, and some exhibit significant slowdown.
- NOX is the most problematic case: serial runtime of 3 minutes grows to 11 minutes at 3 threads — a 3.7x slowdown.
- Peak observed speedup at 8 threads is approximately 1.39x (biodiesel2), far below the ideal 8x linear scaling.
- Parallel efficiency across all scalable mechanisms averages roughly 15-20%, indicating heavy serial overhead inherent to the ODE integration workload.

2. Methodology

All tests were conducted on a 2018 Alienware laptop running Windows 11, using the Intel oneAPI toolkit with the Intel Fortran compiler and OpenMP runtime. This is a single-socket, single-chip consumer laptop CPU with a finite number of physical and logical cores. Unlike server-class hardware, laptop CPUs are subject to thermal throttling, background OS processes, and power management policies that can introduce variability into timing results.

Each mechanism was executed sequentially in a batch job controlled by a Windows batch script. START and END timestamps were captured in a log file and parsed using a custom PowerShell script (Parse-JobTimes.ps1) to calculate elapsed wall-clock time in minutes. Tests were run at the following thread configurations:

- Serial (1 thread)
- 2 threads

- 3 threads
- 4 threads
- 8 threads

Due to the one-minute resolution of the timestamp format, mechanisms completing in under two minutes are subject to timing noise and their scaling results should be interpreted with caution.

3. Full Timing Results

All runtimes are in minutes. Values of 0 indicate sub-minute completion. Red-highlighted cells indicate parallel slowdown relative to serial.

Mechanism	Serial	2T	3T	4T	8T	Species	Reactions
tnt_rdx	1	0	1	0	1	333	3,191
AramcoMech20	1	1	1	1	1	496	5,135
dimethlyether	1	1	1	1	1	79	679
dimethyloxy	1	1	0	0	0	76	979
gasoline1	10	7	9	6	6	1,375	10,733
grimech_heat_and_loss	0	1	0	1	1	58	634
heptane_pitz	1	0	0	0	0	162	1,426
iso-octane	5	3	4	2	2	873	6,934
kilpinen97	0	0	1	0	1	60	706
methycarbonate	0	0	0	1	0	101	846
nbutane	0	1	0	0	0	157	1,376
NOX	3	8	11	8	9	130	1,276
organophosphates	0	0	0	0	0	152	1,516
propane_pitz	0	1	0	0	0	118	1,252
reduced_diesel	1	0	0	1	0	165	1,674
methydecanoate	72	63	61	59	57	2,880	16,831
c8_to_c16	31	26	25	24	23	2,113	13,868
complete_diesel	129	114	107	106	101	2,848	20,141
biodiesel2	158	144	142	138	135	3,789	19,289

* Red cells indicate parallel slowdown (runtime exceeds serial).

4. Scaling Analysis

4.1 Large Mechanisms (5+ minutes serial)

The following mechanisms have sufficient computational workload to benefit from parallelization:

Mechanism	Serial (m)	8T (m)	Speedup	Efficiency	Species	Reactions
gasoline1	10	6	**1.67x**	21%	1,375	10,733
iso-octane	5	2	**2.50x**	31%	873	6,934
methydecanoate	72	57	**1.26x**	16%	2,880	16,831
c8_to_c16	31	23	**1.35x**	17%	2,113	13,868
complete_diesel	129	101	**1.28x**	16%	2,848	20,141
biodiesel2	158	135	**1.17x**	15%	3,789	19,289

4.2 Parallel Slowdown Cases

The following mechanisms ran slower with multiple threads than in serial. This is a strong indicator that parallelization overhead (thread synchronization, cache invalidation, or load imbalance) exceeds any computational benefit for these mechanism sizes.

Mechanism	Serial	2T	3T	4T	8T	Species	Reactions
NOX	3	**8**	**11**	**8**	**9**	130	1,276

5. Discussion

5.1 Why Large Mechanisms Scale Better

Chemical mechanisms with thousands of species and reactions provide substantially more computational work per ODE timestep. This larger workload amortizes the fixed costs of OpenMP thread creation, synchronization barriers, and memory access coordination. Mechanisms such as biodiesel2 (3,789 species, 19,289 reactions) and complete_diesel (2,848 species, 20,141 reactions) show consistent, monotonic improvement from serial through 8 threads.

5.2 NOX Anomaly

NOX exhibits the most severe parallel slowdown of all mechanisms tested. With only 130 species and 1,276 reactions, the computational kernel is too small to occupy 2-8 threads efficiently. Runtime increases from 3 minutes (serial) to a peak of 11 minutes at 3 threads — a

3.7x degradation. On a laptop CPU with a limited number of physical cores, spawning 3-8 threads for a 130-species mechanism means threads spend more time coordinating than computing. Windows background processes and the OS scheduler also compete for cores, exacerbating the overhead. NOX should be excluded from parallel runs and executed serially.

5.3 Amdahl's Law and Serial Fractions

Even for the largest mechanisms, speedup at 8 threads reaches only approximately 1.27-1.39x rather than the theoretical maximum of 8x. This implies that roughly 85-90% of the wall-clock time is spent in inherently serial sections of the code — most likely the ODE integrator itself, Jacobian matrix factorization, or file I/O. Amdahl's Law predicts that no amount of additional parallelism can overcome this serial fraction.

5.4 Diminishing Returns Beyond 4 Threads

For most large mechanisms, the incremental speedup from 4 to 8 threads is marginal. For example, methydecanoate drops from 59 minutes at 4 threads to 57 minutes at 8 threads — only a 3% improvement for doubling the thread count. This suggests the parallel scalability limit has been effectively reached at 4 threads for the current code structure. On a laptop with a limited core count, requesting 8 threads may also exceed the number of physical cores available, forcing the OS to time-share threads on the same cores — further reducing efficiency.

6. Recommendations

- Run NOX and all small mechanisms (<200 species) in serial mode only. Parallel overhead causes net degradation.
- Use 4 threads as the practical optimum for large mechanisms. The 4-to-8 thread improvement is marginal and may not justify the resource cost on shared compute nodes.
- Profile the ODE integrator (likely VODE or LSODE) to identify the dominant serial bottleneck. Reducing this fraction is the highest-leverage optimization available.
- Investigate OpenMP schedule(dynamic) for reaction rate loops, which may reduce load imbalance for mechanisms with variable-cost reactions.
- Set OMP_PROC_BIND=close and OMP_PLACES=cores to keep OpenMP threads pinned to physical cores and prevent the Windows scheduler from migrating them during a run.
- Consider a mechanism-size threshold in the batch controller: automatically select serial vs. parallel mode based on species count, using ~500 species as a suggested cutoff.

7. Summary

Parallel execution provides meaningful benefit for 6 of the 19 mechanisms tested. The remaining 13 mechanisms are either too small to benefit or actively harmed by multi-threading.

A selective parallelization strategy, rather than blanket application of OpenMP threads, is recommended.

Category	Count	Recommended Threads	Max Speedup Observed
Large mechanisms (2000+ species)	4	4	**1.39x**
Medium mechanisms (500-2000 species)	2	2-4	**1.27x**
Small mechanisms (<500 species)	13	**Serial only**	N/A

Report generated automatically from Kintecus batch log files using Parse-JobTimes.ps1.
Hardware: 2018 Alienware laptop, Windows 10, Intel oneAPI / Fortran compiler.