Arraybasierter CFK-Baumaufbau: Vergleich auf zwei Rechnern

Pascal Bauer

Versuchsplanung und Auswertung mit ChatGPT

Messungen vom 18. September 2026

1 Fragestellung und Vollstaendigkeit

Koennen vorlaeufig gleiche Blattcodes in Teilchen-Arrays die Arbeit auch bei starker Verklumpung besser verteilen? Der Vorschlag stammt von Pascal Bauer. Die rundenweise Implementierung, Instrumentierung und konkrete Versuchsmatrix wurden von ChatGPT ausgearbeitet. Dieser Bericht wertet die abgeschlossene Serie aus; er ist nicht der fruehere Versuchsplan und nicht der spaetere Hydro-Richtungsversuch.

Rechner	Prozesse	Messzustaende	Minuten	Threads
dreihemden	786	2682	44.2	1–14
schmalzberg	462	1710	57.5	1–8

Alle 1248 geplanten Prozesse und 4392 Messzustaende liegen vor. Die Matrix wurde auf fehlende und doppelte Kombinationen geprueft; alle Threadzahlen sind fuer jeden Skalierungsfall vorhanden. Die einzelnen JSONL-Dateien stimmen mit der gesammelten Ergebnisdatei ueberein. Alle nativen exakten Referenzkontrollen bestanden. Zusaetzlich stimmen die Ergebnis-Digests fuer gleiche Eingaben ueber Modi, Threadzahlen und Rechner hinweg ueberein; der Digestvergleich ersetzt nicht die nativen vollstaendigen Vektorvergleiche.

Je Rechner sind acht Quellen-/Programmpruefsummen verifiziert, einschliesslich der im Laufstatus genannten Programmdatei. Quellen, Binaries, Compiler, Flags, Testprotokolle und Dienstjournale sind lokal archiviert. Kein neuer Rechenlauf war fuer diese Auswertung erforderlich. Die Arbeit Version 0.3 blieb unveraendert.

2 Aufbau der Versuche

Je 100000 und 300000 Teilchen, drei Seeds (807–809), uniforme Kontrolle, vier verschieden dichte Cluster und 90-Prozent-Hotspot. Ein vollstaendiger Aufwaermzustand, dann drei gemessene Aufbauten je Prozess; Varianten innerhalb desselben Datenblocks gemischt. Zusaetzlich 30 bewegte Hotspot-Zustaende je Seed bei physischer Kernzahl und maximaler Threadzahl, sowie gesonderte Profilierung bei 300000 Teilchen.

Es sind vorgegebene Koordinatenfolgen, keine physikalischen Gravitations- oder Hydrolauefe. Cluster haben Seitenlaengen 0.25, 0.10, 0.035 und 0.012. Der Hotspot enthaelt 90% der Teilchen in einem Quadrat der Seitenlaenge 0.015 um $(0.6 + 0.18 \sin t, 0.6 + 0.18 \cos t)$; der Rest bildet einen

Hintergrund, $t = 0.09\,j$. Uniforme Frames wiederholen dieselbe Geometrie. Initialisierung und Datei-I/O liegen ausserhalb der nativen Aufbauzeit; notwendiges Packen der Koordinaten ist enthalten.

3 Verfahren und Messdefinitionen

Referenz: rekursive Partitionierung und serieller Konformitaetsabschluss. Nur die abschliessende Besitzersuche verwendet die angeforderte Threadzahl. Referenz mit mehreren Threads ist daher nicht vollstaendig seriell.

Bisheriger Parallelpfad: eine grobe Front bis Tiefe sechs erzeugt Aufgaben. Threads bearbeiten Praefixgebiete und schliessen Teilungsmengen privat unter Vorfahren und Hypotenusenpartnern ab; anschliessend werden diese Mengen vereinigt. Auch mit einem Thread bleibt es dieser Algorithmus, nicht automatisch die Referenz.

Array-Runden: aktive Teilchen nach vorlaeufigem Blattcode gruppieren, eindeutig getrennte entfernen, alle verbleibenden Teilchen parallel um ein Codebit weiterfuehren. Das Gruppieren verwendet hier serielles std::sort. Ein dichter Konfliktbereich kann somit mehrere Threads beschaeftigen. Der nachfolgende Konformitaetsabschluss bleibt aber seriell wie bei der Referenz. Es waere falsch, den gesamten Unterschied zum alten Parallelpfad allein der Bitberechnung zuzuschreiben.

Auswertung: zuerst Median der drei Frames eines Prozesses, danach gepaarte Quotienten pro Seed und Median sowie volle Spannweite der drei Seeds. Bewegte Frames sind keine unabhaengigen Replikate. Keine scheinbar engen Konfidenzintervalle aus wiederholten Zeitmessungen.

Fuer Modus m und Threadzahl p zeigen wir sowohl $S_m(p) = T_m(1)/T_m(p)$ als auch $Q_m(p) = T_m(p)/T_{\mathrm{ref}}(p)$. Ein hoher Speedup gegen einen langsamen eigenen Ein-Thread-Lauf bedeutet noch keinen Vorteil gegen die Referenz. Beim Quotienten Q ist kleiner als eins guenstig.

Rechner: Dreihemden: AMD Ryzen 5 7600, sechs Kerne/zwoelf Hardwarethreads, GCC 13.3; 13 und 14 Threads sind Ueberbelegung. Schmalzberg: Intel i7-8565U, vier Kerne/acht Hardwarethreads, GCC 11.4. Beide: C++20, -O3, keine schnelle unsichere Gleitkommaarithmetik, FP-Kontraktion aus. OpenMP: DYNAMIC=FALSE, PROC_BIND=spread, PLACES=cores, WAIT_POLICY=PASSIVE. Keine kontinuierliche Temperatur- oder Taktkontrolle; Rechnerunterschiede sind wegen verschiedener Compiler keine reine Hardwaremessung.

Tatsaechliche Teams: Besitzersuche entspricht in allen Messungen der Anforderung. Der alte parallele Abschluss verwendet ebenfalls die angeforderte Teamgroesse; Referenz und Arrayabschluss bleiben bei eins. Array-Praefixteams entsprechen der Anforderung. Beim alten Parallelpfad wird der Praefixzaehler nur bei Profilierung gesetzt: dort wurden sechs bzw. vier Threads bestaetigt. Der Wert eins in unprofilierten Laeufen ist dort kein Nachweis serieller Ausfuehrung.

Warum oft nur ein Thread sichtbar war: serielle Aufbauphasen und vollstaendige serielle Referenzkontrollen zwischen den Messungen. Prozessdauer und Peak-RSS enthalten diese Kontrollen, die native Aufbauzeit nicht. Aus einer Taskmanager-Momentaufnahme laesst sich kein gemessener Speedup ableiten.

4 Gesamtzeiten: dreihemden

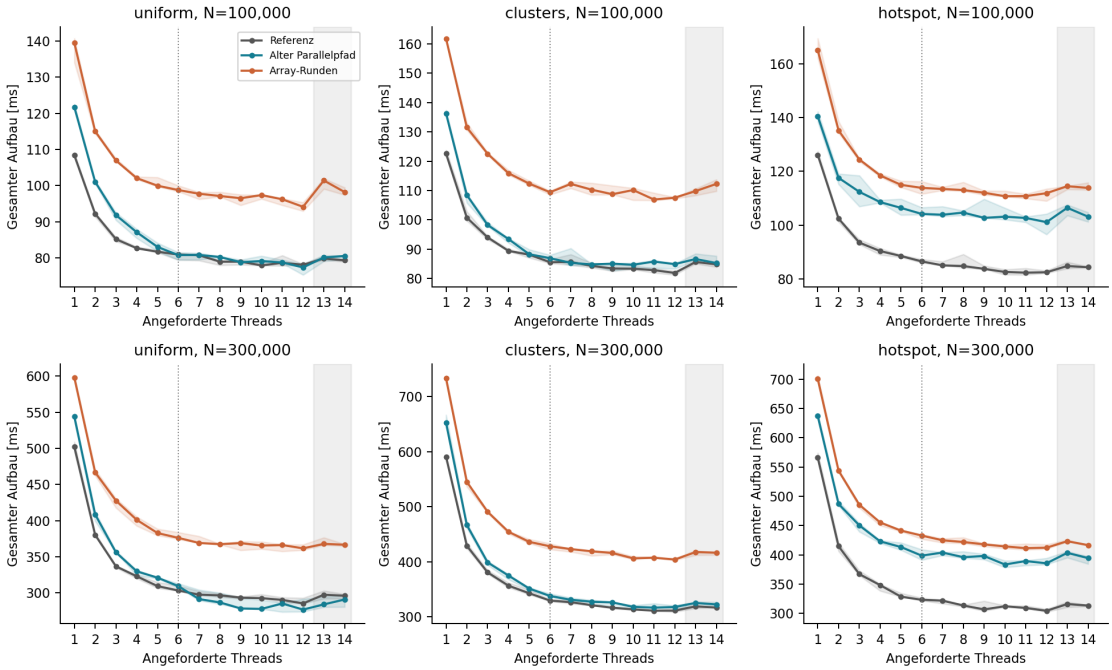

Abbildung 1: Alle Threadzahlen, drei Verfahren und beide Teilchenzahlen. Linien: Median der drei Seeds; Baender: volle Spannweite. Senkrechte Linie: physische Kerne; grauer Bereich auf Dreihemden: Ueberbelegung.

Die Gesamtzeit ist das primaere Entscheidungskriterium. Die Referenz kann durch parallele Besitzersuche ebenfalls schneller werden. Unterschiede zwischen Formen entstehen nicht nur durch Teilchenzahlen, sondern auch durch Praefixtiefen, Konformitaetsaufwand und Speicherzugriffe.

5 Skalierung und Referenzvergleich: dreihemden

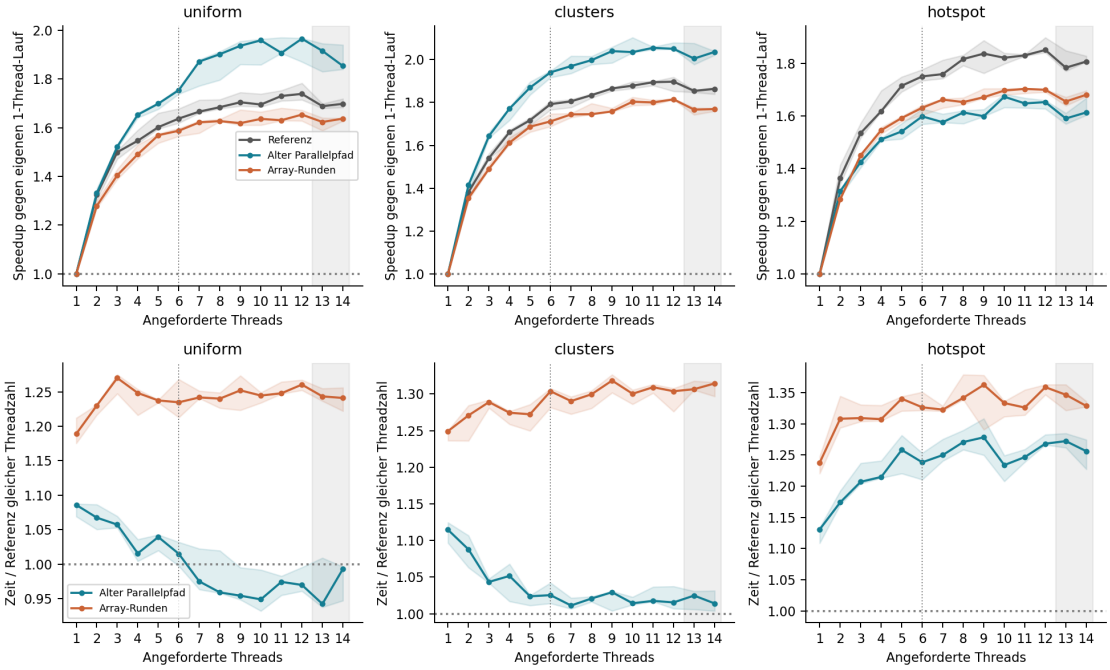

Abbildung 2: 300000 Teilchen: oben Speedup gegen den eigenen Ein-Thread-Lauf (groesser ist besser), unten Zeit relativ zur Referenz gleicher Threadzahl (kleiner ist besser). Drei gepaarte Seeds, keine idealisierte lineare Skalierung unterstellt.

Die Daten erlauben eine Auswahl innerhalb dieser getesteten Konfiguration, nicht die Behauptung einer universell optimalen Threadzahl. Das nachtraeglich beobachtete Minimum ist insbesondere keine unabhaengig bestaetigte Einstellung fuer andere Verteilungen.

6 Phasen: dreihemden

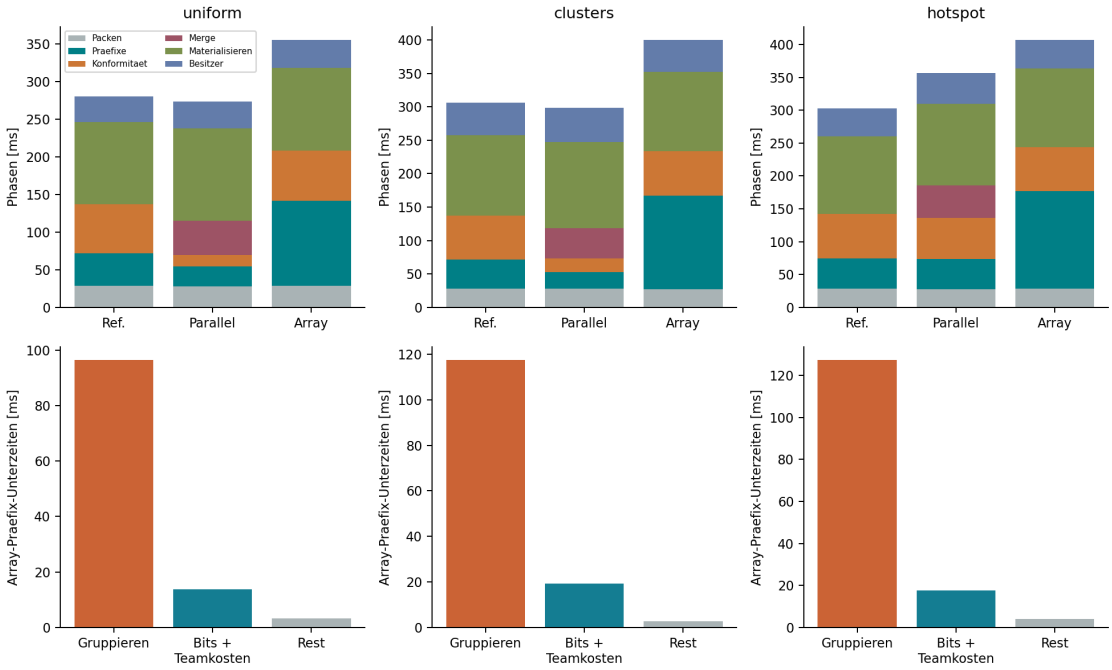

Abbildung 3: 300000 Teilchen bei physischer Kernzahl. Oben getrennte Aufbauphasen; unten nur Unterzeiten der Array-Praefixphase. code_bits und grouping werden nicht nochmals zur Gesamtzeit addiert. Gestapelte Phasenmediane muessen wegen der Medianbildung und unzugeordneter Restkosten nicht exakt den Gesamtmedian ergeben.

Bitzeit enthaelt Teamstart und Synchronisation; sie ist keine reine Rechenzeit der Bitoperation. Gruppierung enthaelt Sortierung, Konflikterkennung und Kompaktierung. Beim Parallelpfad sind Konformitaet und Merge separat; beim Arraypfad ist Konformitaet seriell, Merge null.

7 Gesamtzeiten: schmalzberg

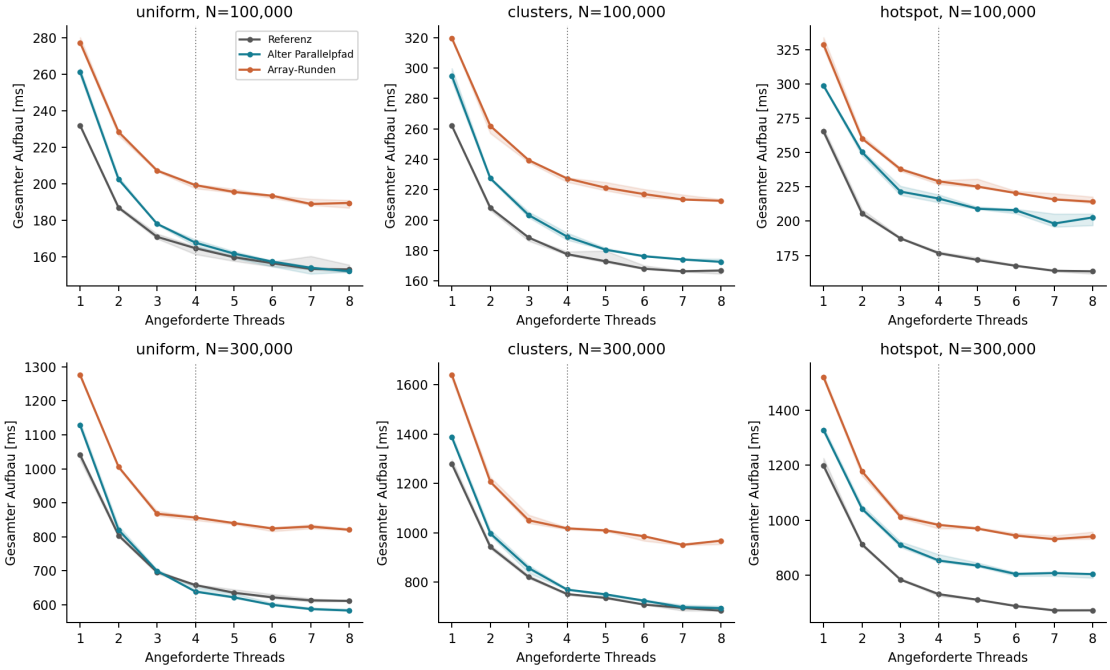

Abbildung 4: Alle Threadzahlen, drei Verfahren und beide Teilchenzahlen. Linien: Median der drei Seeds; Baender: volle Spannweite. Senkrechte Linie: physische Kerne; grauer Bereich auf Dreihemden: Ueberbelegung.

Die Gesamtzeit ist das primaere Entscheidungskriterium. Die Referenz kann durch parallele Besitzersuche ebenfalls schneller werden. Unterschiede zwischen Formen entstehen nicht nur durch Teilchenzahlen, sondern auch durch Praefixtiefen, Konformitaetsaufwand und Speicherzugriffe.

8 Skalierung und Referenzvergleich: schmalzberg

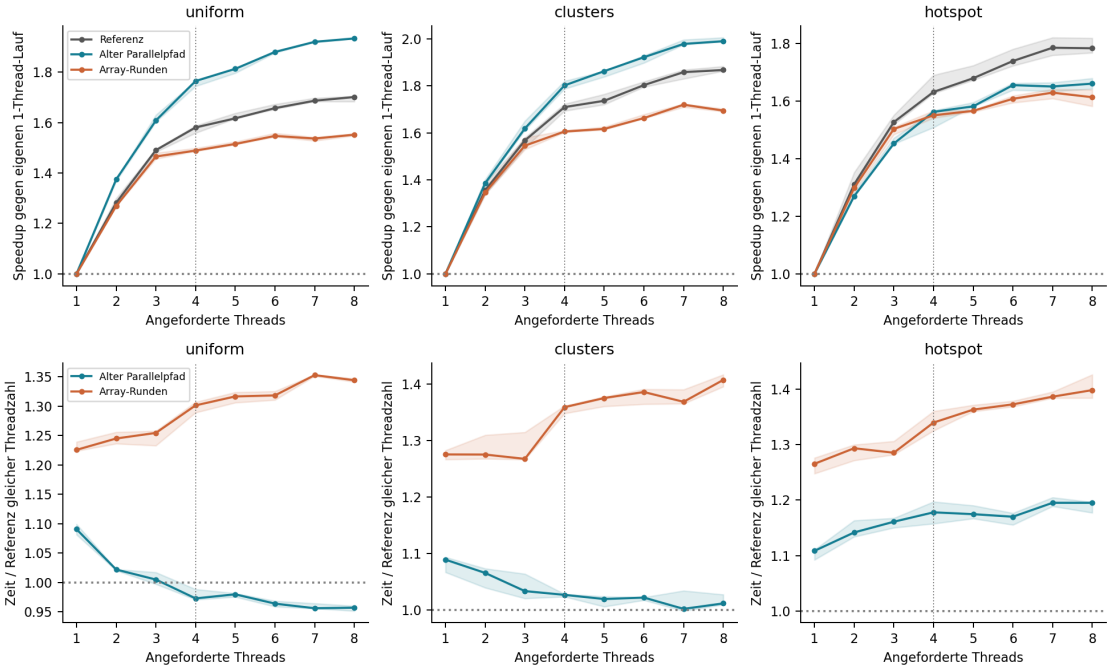

Abbildung 5: 300000 Teilchen: oben Speedup gegen den eigenen Ein-Thread-Lauf (groesser ist besser), unten Zeit relativ zur Referenz gleicher Threadzahl (kleiner ist besser). Drei gepaarte Seeds, keine idealisierte lineare Skalierung unterstellt.

Die Daten erlauben eine Auswahl innerhalb dieser getesteten Konfiguration, nicht die Behauptung einer universell optimalen Threadzahl. Das nachtraeglich beobachtete Minimum ist insbesondere keine unabhaengig bestaetigte Einstellung fuer andere Verteilungen.

9 Phasen: schmalzberg

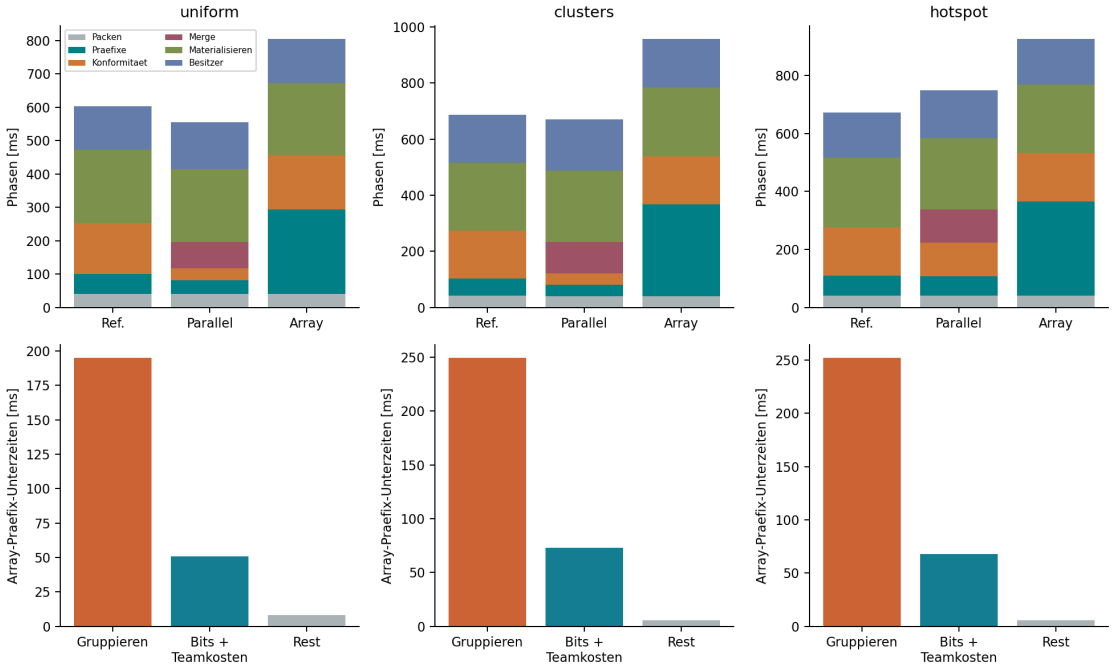

Abbildung 6: 300000 Teilchen bei physischer Kernzahl. Oben getrennte Aufbauphasen; unten nur Unterzeiten der Array-Praefixphase. code_bits und grouping werden nicht nochmals zur Gesamtzeit addiert. Gestapelte Phasenmediane muessen wegen der Medianbildung und unzugeordneter Restkosten nicht exakt den Gesamtmedian ergeben.

Bitzeit enthaelt Teamstart und Synchronisation; sie ist keine reine Rechenzeit der Bitoperation. Gruppierung enthaelt Sortierung, Konflikterkennung und Kompaktierung. Beim Parallelpfad sind Konformitaet und Merge separat; beim Arraypfad ist Konformitaet seriell, Merge null.

10 Load-Balancing

Fuer unabhaengige Aufgaben sei B_j die Summe der Task-Wandzeiten des Threads j: $L = \sum_j B_j / (p \max_j B_j)$. Fuer Array-Runden ist entscheidend, die jeweils langsamsten Threads vor jeder Synchronisation zu beruecksichtigen: $L_{\mathrm{Runden}} = \sum_{r,j} b_{rj} / (p \sum_r \max_j b_{rj})$. Blosses Aufsummieren pro Thread ueber alle Runden koennte wechselnde Nachzuegler verdecken. Leere Threads sind enthalten.

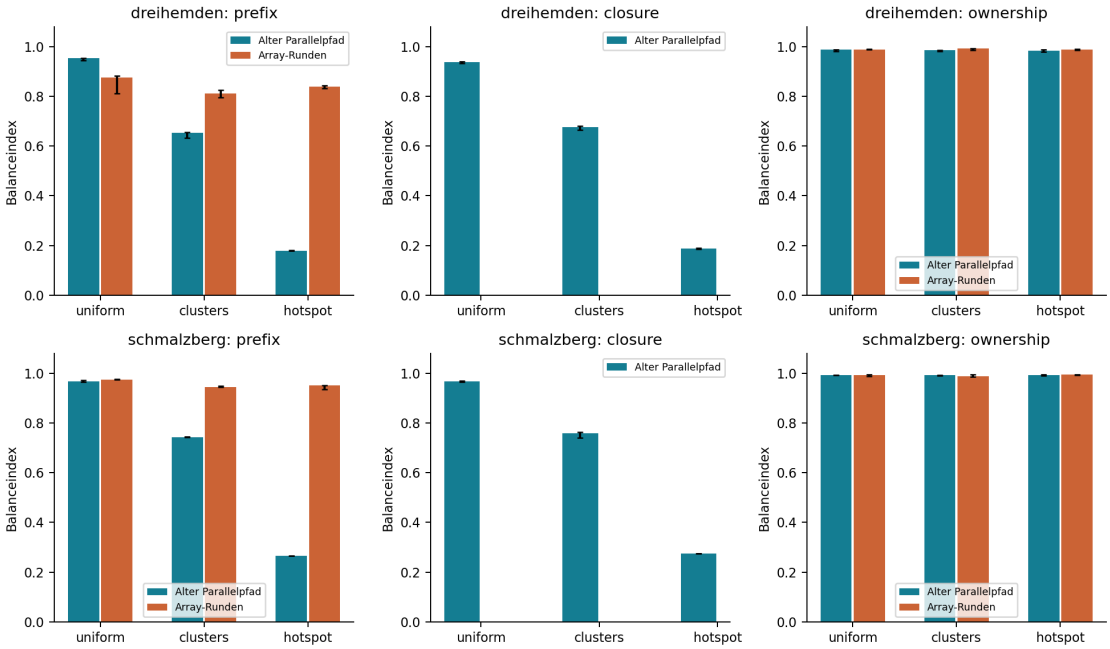

Abbildung 7: Profilierte 300000-Teilchen-Laeufe bei physischer Kernzahl, Median/Spannweite dreier Seeds nach Zusammenfassung der Frames. Praefixindex des Arraypfads rundenweise; alter Pfad aufgabenweise. Beim Arrayabschluss wird bewusst kein paralleler Balancebalken gezeigt: er ist seriell.

Ein Index nahe eins bedeutet aehnliche gemessene Taskzeiten, nicht automatisch hohe CPU-Auslastung oder kurze Gesamtzeit. Allokation, Speicherwartezeit und Praeemption beeinflussen die Wandzeiten. Barrierewartezeit und Hardwareauslastung wurden nicht direkt gemessen. Die Indizes beider Algorithmen haben unterschiedliche Aufgabenzuschnitte und sind deshalb diagnostische, keine alleinigen Siegerkriterien.

11 Laengere bewegte Folgen

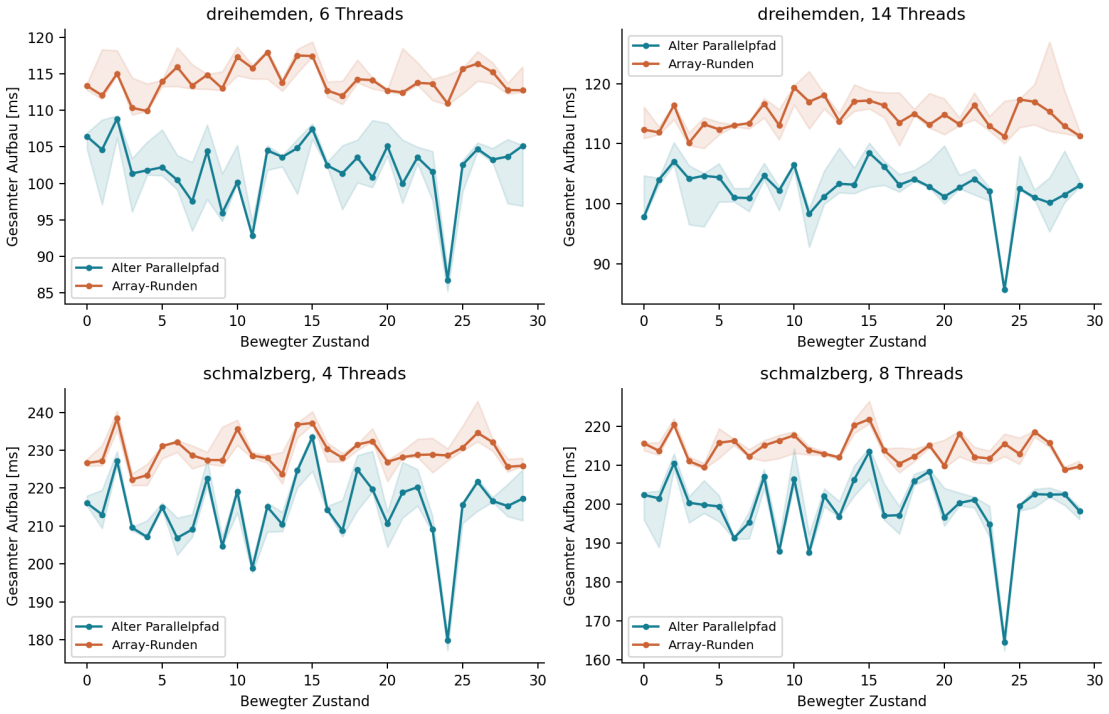

Abbildung 8: Je 30 Hotspot-Zustaende mit 100000 Teilchen, drei Seeds. Linke Spalte: physische Kerne; rechts: maximale getestete Threadzahl (14 auf Dreihemden sind Ueberbelegung). Keine neue physikalische Simulation.

Rechner	Threads	Array / alter Parallelpfad
dreihemden	6	1.108
dreihemden	14	1.112
schmalzberg	4	1.068
schmalzberg	8	1.066

Quotienten aus dem Median der 30 Aufbauzeiten je Seed. So werden die bewegten Frames nicht als 90 unabhaengige Replikate behandelt. Einzelne Spitzen sind sichtbar; die Reihe deckt nur den vorgeschriebenen Bewegungsabschnitt ab.

12 Speicherbedarf und Profilierung

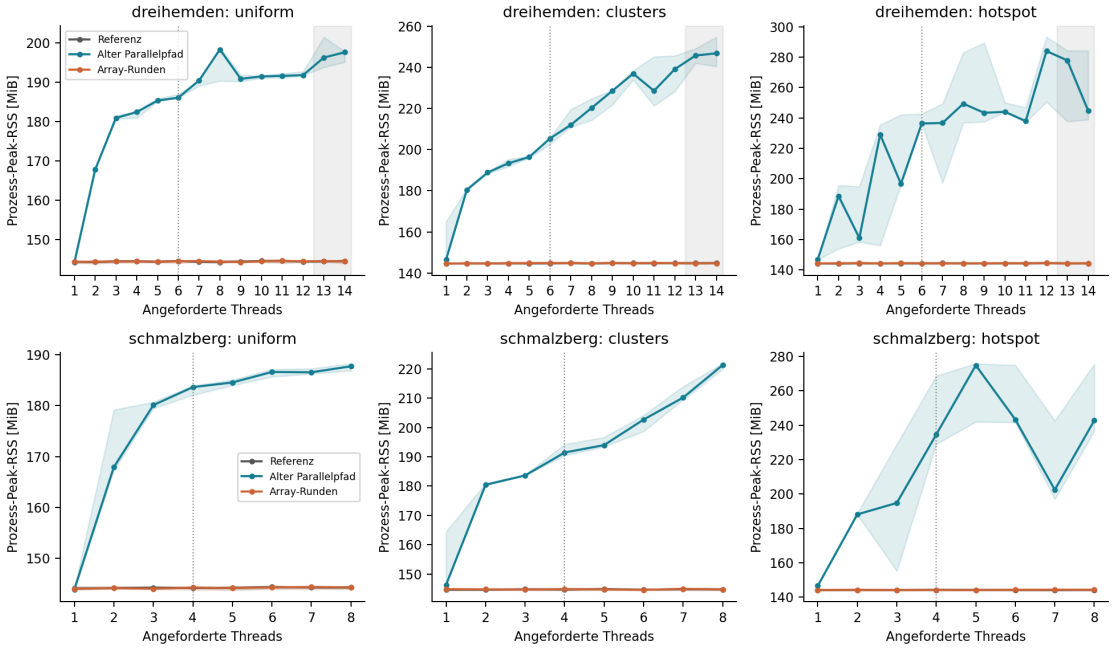

Abbildung 9: Peak-RSS pro Prozess fuer 300000 Teilchen, alle Threadzahlen. Enthalten sind Eingaben, gemessener Baum, Referenzbaum und Kontrollen. Kein isolierter Speicherbedarf eines einzelnen Baumes und keine daraus ableitbare maximale Simulationsgroesse.

Der private Konformitaetsabschluss des bisherigen Parallelpfads benoetigt mehrere Mengen plus Vereinigung; der Arraypfad aktive Index-/Codearrays. Die Prozessmessung zeigt den praktischen Spitzenbedarf dieses Versuches, trennt diese Datenstrukturen aber nicht einzeln.

13 Instrumentierung und quantitative Zusammenfassung

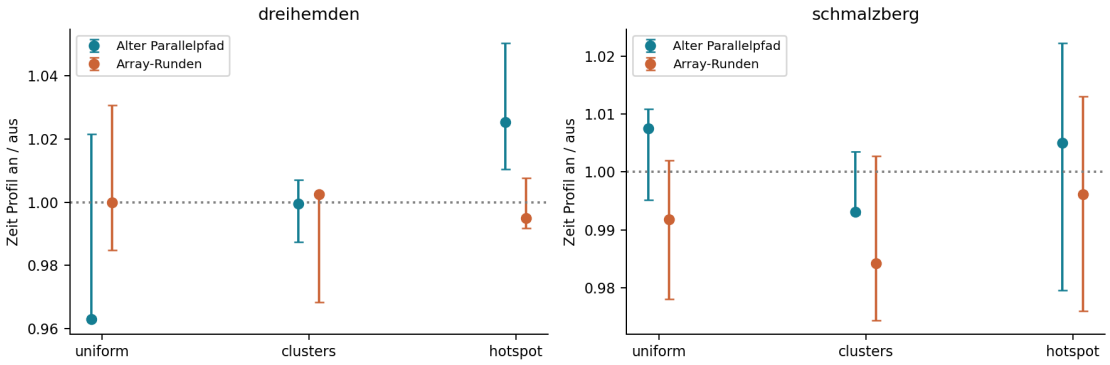

Abbildung 10: Profilierte gegen passende unprofilierte Laeufe: gleiche Form, Seed, 300000 Teilchen und physische Kernzahl. Die Paare wurden zeitlich getrennt ausgefuehrt und sind driftanfaellig; Werte unter eins beweisen keine Beschleunigung durch Profilierung.

Rechner	Form	Ref. [ms]	Parallel [ms]	Array [ms]	Array/Ref.
dreihemden	uniform	303.2	309.3	376.0	1.235
dreihemden	clusters	329.5	337.9	428.1	1.304
dreihemden	hotspot	323.2	398.6	432.9	1.326
schmalzberg	uniform	658.1	639.0	856.6	1.301
schmalzberg	clusters	751.5	769.3	1017.7	1.359
schmalzberg	hotspot	731.4	853.6	983.1	1.340

Tabelle: physische Kernzahl. Die Zeiten sind Mediane ueber drei Seeds; der Quotient ist der Median der gepaarten Quotienten und muss daher nicht exakt dem Quotienten der angezeigten Mediane entsprechen.

14 Ergebnisse, Grenzen und naechste Schritte

14.1 dreihemden

uniform: Array/Referenz 1.23, Array/Parallel 1.22; Praefix-Balance alt 0.95, Array rundenweise 0.88. Gruppierung beansprucht 85% der Array-Praefixzeit.

clusters: Array/Referenz 1.30, Array/Parallel 1.25; Praefix-Balance alt 0.65, Array rundenweise 0.81. Gruppierung beansprucht 84% der Array-Praefixzeit.

hotspot: Array/Referenz 1.33, Array/Parallel 1.09; Praefix-Balance alt 0.18, Array rundenweise 0.84. Gruppierung beansprucht 86% der Array-Praefixzeit.

Im Hotspot enthaelt die groesste alte Praefixaufgabe im Median 90.3% aller Teilchen. Dynamische Vergabe fertiger Aufgaben kann eine bereits laufende grosse Aufgabe nicht automatisch zerlegen.

14.2 schmalzberg

uniform: Array/Referenz 1.30, Array/Parallel 1.33; Praefix-Balance alt 0.97, Array rundenweise 0.97. Gruppierung beansprucht 77% der Array-Praefixzeit.

clusters: Array/Referenz 1.36, Array/Parallel 1.32; Praefix-Balance alt 0.74, Array rundenweise 0.95. Gruppierung beansprucht 76% der Array-Praefixzeit.

hotspot: Array/Referenz 1.34, Array/Parallel 1.15; Praefix-Balance alt 0.27, Array rundenweise 0.95. Gruppierung beansprucht 78% der Array-Praefixzeit.

Im Hotspot enthaelt die groesste alte Praefixaufgabe im Median 90.3% aller Teilchen. Dynamische Vergabe fertiger Aufgaben kann eine bereits laufende grosse Aufgabe nicht automatisch zerlegen.

14.3 Technische Folgerung

Entscheidung: Array-Runden nicht zum Standard machen. Bei 300000 Teilchen und physischer Kernzahl sind sie gegenueber der Referenz je nach Rechner und Verteilung etwa 23–36% langsamer. Gegenueber dem bisherigen Parallelpfad betraegt der Nachteil etwa 9–33%. Zugleich verbessert sich der Hotspot-Praefix-Balanceindex auf Dreihemden von 0.18 auf 0.84 und auf Schmalzberg von 0.27 auf 0.95. Die Idee loest also ein reales Verteilungsproblem, doch die Kosten dieser konkreten Umsetzung ueberwiegen.

Die Auswertung muss zwei Fragen trennen: Laesst sich ein dichter Teilchenbereich auf mehrere Threads aufteilen? Und wird dadurch der gesamte Baum schneller fertig? Verbesserte Lastverteilung kann durch Sortieren, Barrieren, seriellen Abschluss oder Materialisierung aufgebraucht werden. Der getestete Array-Prototyp ist keine allgemeine Widerlegung oder Bestaetigung aller arraybasierten Baumverfahren.

Prioritaet haben nun gezielte technische Varianten: (1) adaptive Aufgabengroessen statt fester Fronttiefe im bisherigen Parallelpfad; (2) Gruppierungskosten im Arraypfad senken, etwa durch Erhalt der Gruppenstruktur oder Radix-Verfahren; (3) erst danach Kopplung an einen gepruefteten parallelen Konformitaetsabschluss. Mehrere Bits pro Runde sind ein weiterer Kandidat, aber kein bereits gemessener Erfolg. Jede Aenderung muss wieder dieselben exakten Kontrollgitter erzeugen.

Kein automatischer Produktionswechsel und keine neuen Simulationen wurden aus dieser Auswertung gestartet. Die vollstaendige Ergebnismatrix, per-Seed-Quotienten, rundenweisen Kontrollen, Quellen und Compilerangaben bleiben nachvollziehbar archiviert. Offene Optimierungen sind Vorschlaege, keine Zusagen eines bestimmten Speedups.