CFK-Baumaufbau mit Teilchen-Arrays

Vorab festgelegter Versuchsplan: alle Threadzahlen

Pascal Bauer

Versuchsplanung und Implementierung mit ChatGPT

18. September 2026

1 Fragestellung und Hypothese

Pascal Bauer schlug vor, Teilchen in getrennten Array-Einträgen zu bearbeiten, zunächst gleiche Blattcodes zuzulassen und Kollisionen dieser Codes in weiteren Durchgängen aufzulösen. Dadurch kann die Arbeit auch innerhalb einer einzigen stark besetzten Baumregion auf mehrere Threads verteilt werden.

Die Hypothese lautet: Eine gleichmässigere Verteilung dieser Arbeit kann den Aufbau verklumpter Verteilungen beschleunigen. Das ist noch kein Ergebnis. Zusätzliches Gruppieren, Speicherverkehr und Synchronisation können den Gewinn aufheben. Gemessen werden deshalb sowohl die einzelnen Phasen als auch der gesamte Baumaufbau. Es wird keine physikalische Simulation integriert.

2 Drei kontrollierte Varianten

Referenz: Rekursive Mengenpartitionierung und serieller Konformitätsabschluss; bestehende parallele Besitzerzuordnung.

Bisher parallel: Grobe Aufgabenfront bei Tiefe 6, private parallele Konformitätsabschlüsse und anschliessende Vereinigung.

Array-Runden: Parallele Kindentscheidungen pro Teilchen, serielles Gruppieren gleicher Codes und Entfernen bereits eindeutiger Gruppen. Konformitätsabschluss wie in der Referenz. Produktionsstandard unverändert.

Schritt	Aktive Codes (schematisch)	Aktion
Start	$1, 1, 1, 1$	Wurzel mehrfach belegt
Erstes Kindbit	$10, 10, 11, 11$	beide Gruppen weiter teilen
Zweites Kindbit	$100, 101, 110, 110$	erste zwei Teilchen fertig
Drittes Kindbit	$1100, 1101$	keine Konflikte mehr

Die Tabelle zeigt Binärcodes, keine konkreten Messkoordinaten. Eindeutige Teilchen behalten ihre Identität; nur die aktive Indexliste wird sortiert. Alle noch aktiven Codes liegen in derselben Tiefe. Konformität kann danach zusätzliche Teilungen erzwingen; die endgültigen Besitzer werden neu bestimmt.

3 Matrix und Umfang

Rechner	physische Kerne	Hardware-Threads	getestete Threads
Schmalzberg	4	8	jeder Wert $1, \dots, 8$
Dreihemden	6	12	jeder Wert $1, \dots, 14$

13 und 14 Threads auf Dreihemden sind ausdrückliche *Überbelegung*. Jede Threadzahl wird mit allen drei Varianten, 100 000 und 300 000 Teilchen, drei Verteilungen und drei Seeds geprüft. Die Verteilungen sind eine uniforme Kontrolle, vier verschieden dichte Gruppen und ein bewegter 90-Prozent-Hotspot.

Je Prozess gibt es einen vollständigen Aufwärmaufbau und drei Messzustände. Die Reihenfolge innerhalb eines Datenblocks wird reproduzierbar gemischt. Ergänzend laufen 30 aufeinanderfolgende Hotspot-Zustände bei physischer Kernzahl und maximaler Threadzahl, jeweils mit drei Seeds und beiden Parallelvarianten. Separate instrumentierte Kontrollen erfassen die Lastverteilung bei physischer Kernzahl.

Geplant sind 786 Prozesse auf Dreihemden und 462 auf Schmalzberg, zusammen 1248 Prozesse mit 4392 Messzuständen zuzüglich Aufwärmzuständen. Das Budget beträgt höchstens zwei Stunden Messzeit pro Rechner einschliesslich Referenzkontrollen, maximal vier Minuten pro Prozess. Unvollständige Serien werden als solche ausgewiesen und nicht automatisch neu gestartet.

4 Messung und Gültigkeitskontrollen

Jeder Zustand wird ausserhalb der Messzeit vollständig mit einem separat berechneten Referenzgitter verglichen: Knoten, notwendige Teilungen, Teilchenbesitzer und Tiefenverteilung. Zusätzlich wird die Konformität geprüft.

Getrennt gemessen werden Packung, Teilchenpräfixe, Konformitätsabschluss, Vereinigung, Materialisierung und Besitzerzuordnung. Beim Arraypfad werden Bitberechnung und Gruppierung als *Unterzeiten* der Präfixphase erfasst; sie dürfen nicht nochmals zur Gesamtzeit addiert werden. Rundenanzahl und Teilchenbesuche dokumentieren die tatsächliche Arbeit. Prozess-RSS umfasst auch die Referenzkontrolle und ist kein isolierter Speicherverbrauch des Solvers.

Für Thread-Wandzeiten B_{rj} in Runde r wird neben den Einzelzeiten die rundenweise Balance betrachtet:

$$L_{\mathrm{Runden}} = \frac{\sum_r \sum_j B_{rj}}{p \sum_r \max_j B_{rj}}.$$

So können sich wechselnde Nachzügler nicht durch blosses Aufsummieren pro Thread verbergen. Diese Grösse ist weder CPU-Auslastung noch direkte Barrierewartezeit. Die uninstrumentierten Läufe bestimmen die Skalierung.

5 Betrieb und Auswertung

Pro Rechner läuft nur ein Messprozess gleichzeitig. Teamgrösse ist fest, Threadbindung erfolgt verteilt auf Kern-Plätze; für alle Varianten gilt passives OpenMP-Warten. Die Betriebssystemlast wird vor und nach jedem Prozess erfasst. Verschiedene Compiler und fehlende kontinuierliche Temperaturkontrolle begrenzen Aussagen über reine Hardwareunterschiede.

Die drei Seeds bilden die Replikate; benachbarte Frames sind keine unabhängigen Versuche. Berichtet werden Median und volle Spannweite, Skalierung gegen den eigenen Ein-Thread-Lauf sowie Vergleich mit der Referenz bei gleicher Threadzahl. Die Diagramme markieren physische Kerne, SMT und Überbelegung. Ein schnellerer Teilabschnitt allein genügt nicht für eine Freigabe des Gesamtverfahrens.

Quellstand, Binärdateien, Compilerflags, Rohmessungen und Prüfsummen werden separat archiviert. Die Idee stammt von Pascal Bauer; die konkrete Rundenvariante und diese Kontrollmatrix wurden von ChatGPT ausgearbeitet. Die Arbeit Version 0.3 sowie frühere Messarchive bleiben unverändert.