Algorithm Engineering Exam Assignments

# Exam Assignment Woche 1

## Unterschied zwischen Concurrency und Parallelität

Der Begriff Concurrency umfasst die Fähigkeit eines Systems parallel zu arbeiten was bedeutet, dass das System an mehreren Prozessen arbeiten kann dies aber nicht gleichzeitig passieren muss. Arbeitet ein System parallel dann führt es gleichzeitig mehreren Aufgaben durch.

## Fork-Join Parallelität

Fork-Join Parallelität funktioniert ausgehend von einem Master Thread, welcher für parallel zu bearbeitende Regionen theoretisch beliebig viele weitere Threads erschaffen kann welche dann beschränkt durch die Hardware tatsächlich parallel arbeiten und am Ende einer parallelen Region wieder vereinigt werden. Dabei besteht die Möglichkeit, dass jeder Thread eines parallelen Abschnitts wiederum weitere parallele Threads erzeugt und vor seiner eigenen Zusammenführung zusammenführt.

## Diskussion: There’s planty of room at the Top: What will drive computer performance after Moore’s law?

A picture containing diagram

Description automatically generated

Die Abbildung gibt eine Übersicht über die Ansatzpunkte von denen man in der Zukunft weitere Performanzsteigerrungen erwartet. Im unteren Teil der Abbildung sieht man als „The Bottom“ bezeichnet die den frühreren Haupttreiber von Performanzsteigerrungen, die Semiconductor Technologie. Dort konnte mit der Miniaturisierung der Transistoren Moor’s law begründet werden, nachdem sich alle 2 Jahre die Transistoranzahl eines Prozessors verdoppelte. Bei der Größe der Transistoren ist man Mittlerweile bei wenigen Atomen angelangt. Eine weitere Verkleinerung der Transistoren ist dort aus physikalischen Gründen nicht möglich, was nun die Aufmerksamkeit der Entwicklung auf die höheren Abstaktionsstufen eines Computersystems lenkt, wie dem Oberen Teil der Abbildung zu entnehmen ist. Mit „The Top“ meint man also Softwareverbesserungen, das Finden effizienterer Algorithmen und das Entwickeln besserer Hardwarearchitekturen. Softwareverbesserrungen ergeben sich dabei z.B. aus dem Vermeiden von Software Bloat bzw. dem Entwickeln von für die entsprechende Hardware maßgeschneiderten Software. Das Finden effizienterer Algorithmen durch neue domänenspezifischer Modelle sowie das entwickeln domänenspezifischer Hardwarearchitekturen trägt zu weiteren Performanzsteigerrungen auf den höheren Abstraktionsebenen eines Computersystems bei. Zusammenfassend ist festzustellen, dass künftige Performanzsteigerrungen domänenspezifischischer sind und weniger vorhersehbar sind und sich vor allem aus der besseren Anordnung Koordination und Konnektivität der Entsprechenden Komponenten des Computersystems ergeben.

**Chapter 1 Computer Systems A Programmers Perspective**

## Diskussion: Concurrency und Parallelität

Die Bedeutung von Concurrency und Parallelität in der Informatik liegt in der gesteigerten Effizienz begründet. Dabei versucht man durch den Einsatz von Concurrency und Parallelität die den Zeitlichen Aufwand zu reduzieren und die räumlichen Gegebenheiten durch gleichzeitige Verarbeitung besser zu nutzen.

Parallelität unterscheidet sich von Concurrency insofern, dass Parallelität die tatsächlich gleichzeitige Bearbeitung mehrerer Aktionen meint und Concurrency lediglich die sequenzielle Bearbeitung mehrerer kontextuell zusammengehöriger Aktionen. Es folgt also, dass jede parallele Verarbeitung auch concurrent ist aber nicht jede concurrente Verarbeitung auch parallel ist. Somit ist die Parallelität eine Teilmenge von Concurrency.

Daraus folgt aber auch, dass im Vergleich zu Concurrency nur Parallelität eine echte Zeitersparnis bei der Verarbeitung bringt.

In Computer Systemen lassen sich Parallelität und Concurrency auf verschiedenen Abstraktionsebenen ausnutzten.

Demnach unterscheidet man z.B zwischen Thread-Level Concurrency, Instruction level Parallelität, und SIMD Parallelität.

Unter Thread-Level Concurrency versteht man dabei die Fähigkeit des Computer Systems meherere Threads innerhalb mehrerer Prozesse gleichzeitig zu bearbeiten. Diese Art von Concurrency ermöglicht es dem Nutzer mehrere Programme gleichzeitig zu benutzten oder einem Server mehrere Clients zu bedienen. Des Weiteren erlaubt die Technik des simultaneous multi-threading einer CPU mehrere Kontrollflüsse auszuführen. Hyperthreading erlaubt es einer CPU zudem für jeden Rechenzyklus zu entscheiden welcher Thread bearbeitet werden soll.

Wenn ein Prozessor in der Lage ist mehrere Instruktionen gleichzeitig zu verarbeiten spricht man von Instruction level parallelism. Dies lässt sich in Form von Pipeling realisieren, wobei die auszuführende Instruction in mehrere Bearbeitungsschritt zerlegt wird. Die Einheiten, welche jeweils einen Bearbeitungsschritt verantwortlich sind, bearbeiten diesen parallel. Prozessoren, die schneller mehr als eine Instruktion pro Taktzyklus verarbeiten können werden supercalare Prozessoren genannt.

Viele Prozessoren sind des Weiteren in der Lage mit einer Instruktion mehrere Eingabedaten zu verarbeiten. Dieses Prinzip wird als single-instruction, multiple-data (SIMD) bezeichnet und eignet sich besonders um Mehrdimensionale Daten wie z.B Bild-, Video- oder Tondaten zu verarbeiten.

Abschließend lässt sich sagen, dass Concurrency und besonders Paralelität für die Performace eines Computersystems eine große Rolle spielt. Des Weiteren sind sie bereits auf allen Abstraktionsebenen des Computersystems implementiert, allerdings bietet zumeist die Software als höchste Abstraktionsebene das meiste potenzial für weitere Optimierung durch Concurrency und Parallelität.

# Exam Assignments 2

## Was ist false sharing?

False sharing tritt auf wenn eine Variablen in Cache lines verschiedener Prozessoren von jeweils einem unterschiedlichen Thread modifiziert werden. Diese gleichzeitige Veränderung führt zur Invalidierung der Cachelines und erzwingt ein Update des Speichers, um den Cache kohärent zu halten.

## Wie verhindern Mutual exclusion Konstrukte race conditions?

Mutual exclusion Konstrukte dienen zur Synchronisation von Threads, um zu garantieren, dass nur jeweils ein Thread einen bestimmten Codeabschnitt ausführt. Dies verhindert, dass mehrere Threads gleichzeitig den Wert einer Variablen ändern.

## Erkläre den Unterschied zwischen static und dynamic schedules bei OpenMP.

Bei OpenMP bedeutet static scheduling, dass das scheduling zur Compilezeit stattfindet, im Vergleich zu dynamic scheduling, was die scheduling Entschiedungen zur Laufzeit des Programms trifft.

## Was kann man machen, wenn die Lösung einer Berechnung in einer parallelen OpenMP Schleife lange vor dem Verlassen der schleife gefunden wurde?

Es ist nicht möglich einfach aus der Schleife auszubrechen, aber mit einem einfachen Code Konstrukt kann nach dem Finden der Lösung einen Flag setzen, welcher allen Threads signalisiert, dass sie bis zum Verlassen der Schleife nur noch „CONTINUE“ ausführen sollen.

## Wie funktioniert std::atomic::compare\_exchange\_weak?

Die Funktion vergleicht atomar das aufgerufene Objekt mit einem erwarteten Objekt. Wenn der bitweise Vergleich erfolgreich ist, wird es durch das gewünschte Objekt ersetzt. Das bedeutet, dass es bitweise gleich dem erwarteten Objekt ist. Es ist möglich, dass die Funktion ein falsches Ergebnis liefert, bei dem der Vergleich nicht erfolgreich ist, selbst wenn die Objekte gleich sind. Als Kompromiss ist diese Version der Funktion in Schleifen möglicherweise leistungsfähiger.

# Exam Assignments 3

## Wie funktionierten ordered clauses in OpenMP in Verbindung mit parallelen For-Schleifen?

Ordered clauses in OpenMP sorgen dafür das eine ansonsten potenziell parallel ausgeführte Region sequenziell ausgeführt wird. Somit lassen sich in parallelen For-Schleifen Regionen markieren, in denen der Programmcode wie in einer sequenziellen Schleife ausgeführt wird.

## Für was benutzt man die Kollapsklausel in OpenMP?

Kollapsklausel werden in OpenMP verwendet, um ineinander Verschachtelte For-Schleifen zu parallelisieren. Die Kollapsklausel ermöglicht die Parallelisierung von verschachtelten Schleifen. Innere Schleifen würden nicht parallel ausgeführt werden. Deshalb ermöglicht die Reduzierung einer Schleife mit m Schritten innerhalb einer Schleife mit n Schritten in eine Schleife mit n∗m Schritten, eine effizientere Parallelisierung und eine bessere Lesbarkeit des Codes.

## Erklären Sie, wie reductions in OpenMP (intern) fuktionieren.

Reductions werden verwendet um einen Werte aus vielen Werten, für gewöhnlich eine Liste von Werten, mit derselben mathematischen Operation berechnen. Dies wird intern erreicht indem lokale Kopien der Listen Variablen, in Abhängigkeit von der Operation gemacht werden. Die einzelnen Werte werden lokal in jedem Thread verändert bis sie schließlich, mit der entsprechenden Operation, in einen einzelnen Wert reduziert und mit dem globalen Wert kombiniert werden.

## Was ist der Nutzen von barriers beim parallelen Rechnen?

Barriers werden zu Synchronisation benutzt. Eine barrier fungiert dabei als Haltepunkt für alle Threads, welche dann erst mit der Programmausführung fortfahren können, wenn der letzte Thread den Haltepunkt erreicht hat.

## Erklären Sie, den UNterschied zwischen den Bibliotheksroutinen: *omp\_get\_num\_threads(), omp\_get\_num\_procs() und*

## *omp\_get\_max\_threads().*

omp\_get\_num\_threads(): Gibt die Anzahl an Threads an.

omp\_get\_num\_procs(): Gibt die Anzahl an logischen cores an.

omp\_get\_max\_threads(): Gibt die maximale Anzahl von Threads in einer parallelen Region an.

## Speicher Attribute: private vs first-private

Private: variables are uninitialized, and each thread gets a local, unshared copy

Firstprivate: variables are initialized, each thread gets a local, unshared copy

## Coding warmup und pseudocode

Die Das C++ Programm für die Pi-Implementierung ist im Ordner beigefügt.

Algorithmus 1 Pi-Parallelisierung

1: procedure PI\_THREAD\_HELPER(Schritt, Breite, Iterationen, Summe)

2: x ←0.0

3: lokale Summe ←0.0

4: for i : Iterationen do

5: x ←(Schritte ∗Iterationen + Iteration + 0,5) ∗Breite

6: lokale\_Summe ←lokale\_Summe

7: end for

8: CRITICAL do

9: Summe ←Summe + locale\_Summe

10: end CRITICAL

11: end procedure

12: procedure PI\_ITERATIVE(num steps, num threads)

13: Summe ←0.0 Iterationen ←num\_steps / num\_threadswidth ←1/num\_steps

14: for i : num threads do

15: create\_thread(pi step, i, width, iterations, sum)

16: end for

17: alle\_Subthreads\_verbinden()

18: pi ←sum ∗width ∗4

19: return pi

20: end procedure

# Exam Assignments 4

## Erklären Sie wie Divide-and-Conquer Algorithmen mit openMP tasks parallelisiert werden können.

OpenMP tasks sind unabhängige Arbeitseinheiten, welch dazu genutzt werden können, um Divide-and-Conquer Algorithmen zu parallelisieren, indem jeder rekursive Aufruf (bis zu einer bestimmten Problemgroße) in einem neuen Task bearbeitet wird. Dies geschieht, indem eine dynamische Queue Threads mit der Bearbeitung von Threads beauftragt, bis alle Tasks bearbeitet wurden.

## Beschreiben Sie einige Möglichkeiten, wie man merge Sort beschleunigen kann.

Zum einen lässt sich der Divide-and-Conquer Algorithmus Merge-Sort mit wie eben beschrieben mit Tasks parallelisieren, was zu einer Beschleunigeung der Ausführung führt, zum anderen kann das ständige Kopieren des zu sortierenden Arrays verhindert werden, indem ein Buffer verwendet wird, um das Ergebnis zwischenzuspeichern.

## Was ist die Idee hinter multithreaded merging?

Die Idee hinter multithreaded mearging ist das Finden des mittleren Elements des vereinigenten Arrays, welches im Median Element in den Beiden sortierten Teilarrays bildet. Nach dem Kopieren des Median Elements an den richtigen Index des vereinigenten Arrays werden die sortierten Teilarrays ohne das Median-Element rekursiv vor bzw. hinter das Median Element einsortiert.

## 2interessante Aspekte des Papers: What every systems programmer should know about concurrency.

## Atomicity:

Atomicity bezieht sich hier wie auch im Allgemeinen auf Unteilbarkeit. Folglich müssen die read und wirte opertionen von Threads bei geteilten Daten atomic sein, torn reads und writes zu verhindern. Diese tron-reads-and-writes passieren, wenn ein thread mehrere Load-Operationen für einen Write benötigt und nach dem ersten Load ein anderer Thread eine Read-Operation durchführt. Damit Atomicity sichergestellt wird, dürfen Variablen, die zur Thread Synchronisation verwendet werden, nicht länger als die Wordlänge der CPU sein.

## Atomic Operations as building blocks

Atomic Loads, Stores und RMW-Operationen sind die Grundbausteine jedes Concurency tools. Es ist nützlich diese in die zwei Lager *blocking* und *lockless* Synchronisation aufzuteilen. Obwohl die Blockierende synchronisations Methoden meist einfacher zu handhaben sind können sie in deadlocks oder livelocks geraten, was unbedingt verhindert werden muss. Selbst wenn man mutexe betrachtet, welche den alleinigen Zugriff eines Threads auf eine geteilte Variable garantieren, blockiert dieser Zugriff bis auf Weiteres alle anderen Zugriffe.

Im Gegensatz dazu garantiert lockless Synchronisation, dass das Programm fortfahren kann. Keine der beiden Herangehensweisen ist grundsätzlich überlegen. Je nach Anwendungsfall ist die eine oder andere Methode vorteilhaft. Des Weiteren garantiert auch die alleinige Verwendung von atomaren Operationen nicht das Blockieren des Programms. Letztlich liegt bei der Betrachtung von Concurrncy-Problemen immer ein Kompromiss zwischen Komplexität und Performance vor.

# Exam Assignments 5

## Was ist CMake?

CMake ist ein Cross-Plattform Build-File Generator, wobei Cmake selbst nicht für den Build des Programms verantwortlich ist. Dafür wird ein Compiler benötigt. Des Weiteren ist CMake in der Lage Build-Files für verschiedene Plattformen zu generieren, wie z.B. MS Visual Studio, Xcode, make, etc. Dabei muss das CMake File den Namen CMakeLists.txt haben. CMake kann neben dem Software-Build (cmake) auch testen (ctest) und package (cpackage). Die CMake Language ist eine kommando-basierte Sprache, wobei jede Zeile ein Kommando haben darf. Generell gilt, dass CMake Kommandos keine Rückgabewerte liefern, was zusätzlich bedeutet, dass Verschachtelungen nicht möglich sind. Variablen werden in CMake mit dem Kommando set() festgelegt und mit dem Kommando unset() entfernt.

## Was für eine Rolle spielen targets in CMake?

Targets in CMake sind die Resultate des Builds, also in der Regel Executables und Software-Bibliotheken. Targets kann man sich als Objekte vorstellen. Des Weiteren haben Targets Eigenschaften, welche man sich als Member-Variablen vorstellen kann. Targets werden mit Konstruktoren wie add\_executable() und add\_library() erstellt. Zusätzlich gibt es Member-Funktionen, welche die Member-Variablen des targets verändern, wie z.B. target\_source(), target\_compile\_options(), target\_include\_directories und target\_link\_libraries.

## Wie geht man bei der Code-Optimierung vor?

Die Code-Optimierung folgt einem einfachen, fünf-schrittigen Algorithmus:

1. Finde eine Lösung für das Problem vorerst ohne Optimierung.
2. Stelle sicher (mithilfe von Unittests), dass das Programm korrekt funktioniert.
3. Falls die Performance des Programms für den Anwendungsfall ausreicht, ist man an dieser Stelle fertig.
4. Andernfalls optimiert man die Implementierung des Programms, indem man nach Bottlenecks sucht, den Algorithmus verfeinert und das Programm parallelisiert und vektorisiert.
5. Gehe zu Schritt 2.

# Exam Assignments 6

## Zählen Sie einige der Chrakteristiken der (Vektor) Instruction-Sets: SSE, AVX(2), und AVX-512 auf.

SSE: Vector länge: 128-bit, Zeitraum: 1999-2009, Register: xmm0 – xmm15 Prozessoren: Pentium III, 3DNow!

AVX, AVX2: länge: 256-bit, Zeitraum: 2011-2013, Register: ymm0 – ymm15

Prozessoren: Sandy bridge, Bulldozer

AVX-512: länge: 512-bit, Zeitraum 2017- heute, Register: zmm0 – zmm15

Prozessoren: Xeon, Skylake, Core-X, Ryzen

## Wie beeinflusst memory aliasing die performance?

Memory aliasing bedeutet, dass mehrere Pointer auf dieselbe Stelle im Speicher zeigen.

Für sichere Optimierungen muss der Compiler davon ausgehen, dass es Aliase von verschiedenen Pointer gibt. Weiß der Compiler, dass die referenzierte Stelle im Speicher einzigartig ist, kann er von mehreren Optimierungen, wie z.B. Vektorisierung Gebrauch machen, da er nicht Gefahr läuft Überschneidungen mit eventuellen anderen Referenzen zu verursachen.

## Was ist der Vorteil von Unit-Stride (Stride-1) Speicherzugriff im Vergleich zu Speicherzugriffen mit größeren Stride (e.g. Stride 8)

Unit-Stride Speicherzugriffe nutzen die Bandbreite des Speichers besser aus und reduzieren die Latenz, außerdem erleichtert es dem Compiler den Code zu automatisch zu vektorisieren.

## Wann würde man eine Anordnung der Elemente als Struktur von Arrays bevorzugen?

Möchte man Operationen (z.B. Reduzierende Operationen wie: min, max, avg., sum) auf alle Elemente eines Arrays der Struktur anwenden, vergleichbar mit der Operation auf einer Spalte einer Datenbank, so ermöglicht die Lokalität der Arrays der SoA eine bessere Performanz.

# Exam Assignments 7

## Erklären Sie drei Vektorisierungsklauseln welche mit „#pragma omp simd“ benutzt werden können.

aligned: Bestimmtm dass Datenstrukturen wie Listen byteweise ausgerichtet werden.

safelen: Bestimmt die maximale Größe, die ein Vector nicht überschreiten wird, aber unterschreiten kann.

reduction: Bestimmt eine mathematische Reduktionsoperation (z.B „+“, „\*“,…), die auf eine Datenstruktur angewendet wird und diese auf einen Wert reduzieren soll.

## Geben Sie Gründe an, welch für bzw. gegen die Verwendung von guided-vectorisation und OpenMP sprechen.

|  |  |  |
| --- | --- | --- |
|  | „pro“ | „contra“ |
| Guided-vectorisation | -Hoher Perfromanzgewinn bei richtiger Verwendung  -Hohe Kontrolle durch Programmierer\*in | -Schwieriger zu benutzten  -Längerer Entwicklungsaufwand  - Intrinsics von Prozessor abhängig |
| OpenMP | -Einfache Benutzbarkeit  -Plattformübergreifend | -nur grobe Optimierungen  möglich |

## Was sind die Vorteile von Vector-Intrinsics gegenüber assembly code?

Intrinsische Funktionen haben den Vorteil, dass man nicht explizit Register angeben muss.

Des Weiteren sind sie portabler zwischen verschieden verschiedenen Betriebssystemen und Compilern einsetzbar. Vector-Intrinsics sind außerdem einfacher zu lernen, einfacher zu benutzten und lesbarer als Assembly-Code. Einfach gesprochen bauen Vector-Intrinsics direkt auf den korrespondierenden Assembly-Befehlen auf.

## Was sind die entsprechenden Vectoren für die drei intrinsic Datentypen:

\_\_m256: ein Vktor im 8 Floatingpoint-Zahlen

\_\_m256d: ein Vektor mit 4 Double-Precistion-Zahlen

\_\_m256i: Ein Vektor mit Integerzahlen, mit bzw. ohne Vorzeichen

# Exam Assignments 8

## Erklären Sie die Namenskonventionen von intrinsic Funktionen

Intrinsic-Funktionen haben die From

\_<vetrosize>\_<operation>\_<suffix>

Wobei die <vectorsize> für die Große des Vektors steht.

mm ist dabei ein 128-bit Vektor (SSE).

mm256 ist der 256-bit Vektor (AVX, AVX2)

mm512 ist der 512-bit Vektor (AVX-512)

<operation> beschreibt welche Funktionalität die Intrinsic-Funktion umsetzt, wie z.B.

Add , Sub , Mul, etc.

Das <suffix> spezifiziert den Datentyp der primären Argumente der Funktion, also die Datentypen, welche im Vektor genutzt werden. E.g., epi32 is for signed 32-bit integer or epu16 for unsigned 16-bit integer.

## Worüber geben die Metriken Latenz (latency) und (throughput) Aufschluss, in Hinsicht auf die Performance von Intrinsic-Funktionen

Die Latenz ist die Anzahl an Zyklen, die eine Intrinsic-Funktion zur Berechnung ihres Ergebnises benötigt. Der Durchsatz beschreibt, wie viele Zyklen es dauert, bis die nächste Intrinsic-Funktion dieser Art berechnet werden kann.

## Wie wird in modernen Prozessoren Instruction-Level Parallelismus umgesetzt?

Instruction-Level Parallelismus wird durch das Verwenden von mehreren funktionalen Einheiten umgesetzt, welche die Instrctions parallel ausführen. Dies erfordert verschiedene funktionale Einheiten in der Ausführungseinheit eines Kerns, was durch Hinzufügen mehrerer Ports zum Scheduler realisiert wird (einen pro funktionaler Einheit). Jeder dieser Ports kann für eine Reihe von Mikrooperationen verwendet werden, z. B. für Verzweigungen, Gleitkommaaddition oder das Laden von Daten.

## Wie beeinflusst Loop-Unrolling die Ausführungszeit des compilierten Programmcodes?

Loop-unrolling ist eine Methode, um schleifen derartig zu transformieren, dass sie in der Lage sind Instruction-Level Parallelismus zu nutzen.

Dabei wird er Programmcode innerhalb der Schleife ersetzt und die Programmlogik, welche die Anzahl an Iterationen kontrolliert, angepasst. Die beschleunigte Programmausführung hat allerdings den Preis einer höheren Compelierzeit, größeren Programmgröße, benötigt mehr Platz im Instruction Cache und mehr Registerplatz.

## Was bedeutet ein hoher IPC Wert (instruction per cycle) in Bezug auf die Performance eines Algorithmus?

Der IPC Wert wird berechnet indem man die Anzahl an ausgeführten Instruktionen durch die Anzahl der Zyklen teilt. Ein hoher IPC Wert impliziert eine hohe CPU-Auslastung, was allerdings nicht unbedingt bedeutet, dass der Algorithmus performant ist, sondern dass er die CPU gut auslasten kann.

# Exam Assignments 9

## Wie unterscheiden sich bandbreitengebundene Berechnungen von berechnungsgebundenen Berechnungen?

Bandbreitengebundene Berechnungen beziehen sich auf Programme oder Teile von Programmen, bei denen

die Ausführungsgeschwindigkeit durch den Speicherzugriff begrenzt wird. Rechengebundene Berechnungen hingegen haben den Engpass bei der CPU. Ersteres ist derzeit viel häufiger der Fall, was es sehr wichtig, den Speicher so effizient wie möglich zu nutzen, indem Ladevorgänge reduziert werden.

## Erklären Sie, warum zeitliche und räumliche Lokalität die Programmleistung verbessern können?

Da bandbreitenbeschränkte Berechnungen an der Tagesordnung sind, ist es wichtig, dass

so wenig Lade- und Speicheroperationen wie möglich verwendet werden. Hält sich ein Programm an die Prinzipien der zeitlichen und räumlichen Lokalität, können bereits geladene Daten am effizientesten genutzt werden, wodurch die Engstelle abgebaut wird. Zeitliche Lokalität bezieht sich auf die mehrfache Wiederverwendung eines Speicherplatzes innerhalb eines kurzen Zeitraums - also die Wiederverwendung von

Daten, die bereits in den Cache geladen wurden. Wenn eine Cache-Zeile nicht mehr benötigt wird, kann sie dann ohne Leistungseinbußen verworfen werden. Räumliche Lokalität bezieht sich auf die umgebenden Speicherplätze. Im Idealfall werden Speicherplätze in der Nähe der

referenzierten Stelle in naher Zukunft vorliegen. Ein gutes Beispiel für die richtige

räumliche Lokalität ist der zuvor besprochene Speicherzugriff auf die Einheit Stride.

## Was sind die Unterschiede zwischen datenorientiertem Design und objektorientiertem Design?

Bei DOD wird das Endformat berücksichtigt und verwendet, um eine optimale Umwandlung

der Eingabe in die gewünschte Ausgabe zu transformieren. Ziel ist es, die Daten so zu strukturieren, dass der Transformationsaufwand in Funktionen reduziert wird. Dies optimiert die CPU-Cache-Nutzung und macht das Programm leichter parallelisierbar. Häufig

verwendeten Daten werden daher oft in einem SoA gespeichert.

Im Gegensatz dazu geht OOD beim Programmentwurf von der Modellierung von Objekten aus, die den Konzepten der realen Welt entsprechen. Im Gegensatz zu DOD kann OOP im Allgemeinen einfacher schreiben und verstehen, da sie in er Regel ausführlicher ist. Einzelne Transformationen, Funktionen und Darstellungen werden jedoch wahrscheinlich nicht effizient sein, was zu einem Leistungsverlust bei der Ausführung führt.

## Was sind Streaming Speicher?

Streaming-Speicher umgehen den Cache und schreiben Daten direkt in den RAM. Dies spart Platz im knappen Cache und verbessert möglicherweise die Leistung. Dies ist in Situationen nützlich, in denen die Daten nicht vor oder kurz nach dem Schreiben gelesen werden. Andernfalls kann die Leistung sogar negativ beeinflusst werden.

## Beschreiben Sie eine typische Cache-Hierarchie in Intel CPUs.

Die Intel Core i7 Cache-Hierarchie hat zwei L1-Caches pro Kern, einen für Doubles und einen für Ganzzahlen. Pro Kern gibt es einen L2-Cache, der von Doubles und Integern gemeinsam genutzt wird. Alle Kerne teilen sich den vereinheitlichten L3 Cache, der vom Hauptspeicher abgeleitet ist. Natürlich sind die Caches der höheren Ebene größer.

## Was sind Cache Konflikte?

Cache-Konflikte treten auf, wenn auf mehr Speicher zugegriffen wird, als in Cache Zeilen gespeichert werden kann, was zu Cache-Zugriffen auf dieselben Cache-Speicherplätze führt und ständige Schreibvorgänge sowie Lesevorgänge in Caches höherer Ebenen erzwingt. Diesen ungewollten Vorgang bezeichnet man als Thrashing.

# Exam Assignments 10

## Erklären Sie einige nützliche Compiler flags auf, die während der Entwicklung nützlich sind.

Mit „-Wall“ aktiviert man die Warnhinweise des Compilers. Das flag „-g“ versorgt Debugger und Profiler mit Debuginformationen. Mit „-fsanitize=address“ erkennt man memory leaks, out-of-bounds Zugriffe und use-after-free Zugriffe.

## Wie kann die Intel oneAPI dabei helfen, bessere Programme zu schreiben?

Die Intel oneAPI eignet sich gut zum profilen während der Entwicklung, allerdings nicht um stand-alone Executabls zu generieren, da in der Regel einige Abhängigkeiten existieren. Diese abhängikeiten müsseten dann für den Intel Compiler erst noch installiert werden.

## “Premature optimization is the root of all evil” Warum?

Man muss zuerst sicherstellen, dass das Programm richtig funktioniert. Es ist des Weiteren wichtig zu wissen, welche Auswirkungen (Fremdwirkung) eine Änderung am Programm nach sich zieht.

# Exam Assignments 11

## Was ist Cython?

Bandbreitengebundene Berechnungen beziehen sich auf Programme oder Teile von Programmen, bei denen

die Ausführungsgeschwindigkeit durch den Speicherzugriff begrenzt wird. Rechengebundene Berechnungen hingegen haben den Engpass bei der CPU. Ersteres ist derzeit viel häufiger der Fall, was es sehr wichtig, den Speicher so effizient wie möglich zu h

## Was sind Möglichkeiten der Beschleunigung von Python-Programmen mit Cython.

Da Cython eine Obermenge von Python ist, ist der einfachste Weg zur Beschleunigung eines bestimmten Python-Programms zu beschleunigen, es einfach mit dem Cython-Compiler kompiliert. Dadurch, dass da es nicht interpretiert werden muss, wird die Ausführungsgeschwindigkeit bereits verbessert. Es gibt ein hohes Potential für Leistungssteigerungen durch das Hinzufügen von statischer Typisierung zu Python-Programmen, da die Typen nicht abgeleitet werden müssen, wenn der C-Code aus Python erzeugt wird. Weitere Möglichkeiten zur Leistungssteigerung sind die Verwendung von (effizienten) C/C++-Bibliotheken, die Verwendung von Cython-Compiler-Direktiven (siehe 11.4) und die Verringerung des Overheads von Funktionsaufrufen mit cdef /cpdef-Funktionen

## Was sind 2 Möglichkeiten der Kompilierung von .pyx-Modulen?

Cython-Module können im Voraus mit dem Kommandozeilenwerkzeug cythonize kompiliert werden. Eine weitere Möglichkeit ist die Verwendung der pyximport-Bibliothek, wie unten gezeigt. Diese kompiliert die Cython-Module im Arbeitsverzeichnis während des Betriebs.

## Was sind Compiler-Direktiven in Cytho?

Compiler-Direktiven in Cython werden am Anfang eines Moduls in einem Header Kommentar oder über die Kommandozeile. Sie aktivieren/deaktivieren Python-Funktionen, mit unterschiedlichen Auswirkungen auf die Leistung.

## cdivision

Diese boolesche Anweisung ermöglicht eine schnellere Division nach C-Standards, zum Beispiel zum Beispiel das Überspringen von Null-Divisionsprüfungen. Laut den Cython-Dokumenten kann dies zu einer Leistungssteigerung von bis zu 35% bei der Division führen, wenn sie auf True gesetzt ist.

## Sprachlevel

Diese Direktive kann auf 2, 3 oder 3str gesetzt werden, was sich auf die Sprachversion von Python bezieht. Da die Standardeinstellung 2 ist und neue Python-Projekte selten noch Python 2 verwenden, ist dies eine nützliche Direktive.

## Was bedeuten die def, cdef und cpdef in Cython?

Die Schlüsselwörter def, cdef und cpdef werden in Cython verwendet, um Funktionen zu definieren. Cdef wird auch für die Zuweisung oder Definition von Variablentypen verwendet. def-Funktionen sind äquivalent zu Python-Funktionen und können als solche verwendet werden, auch in Python Code verwendet werden.

cdef-Funktionen hingegen sind nicht mit Python kompatibel. Diese führt zu einem geringeren Overhead bei Funktionsaufrufen, bietet aber keine Schnittstelle zu reinen Python-Programmen.

cpdef-Funktionen definieren sowohl eine Cython- als auch eine Python-Schnittstelle und bieten so die schnellere Aufrufkonvention in Cython, während die Funktionen weiterhin für die Verwendung in reinen Python-Modulen zur Verfügung stehen, wo die Leistungssteigerung nicht zum Tragen kommt.

## Was sind typisierte Memoryviews und wieverwendet man sie in Cython?

Typisierte Memoryviews sind eine Methode, um Vanilla- oder NumPy-Arrays zu verwenden und den Overhead zu vermeiden, der bei Verwendung der Python-Schnittstelle entstehen würde. Die Syntax lautet <Typ> [:, ..., :], wobei jedes : eine weitere Dimension bezeichnet.

# Exam Assignments 12

## Was sind extension types im Kontext von Python?

Python extension types sehen so aus wie Python Code und sind wie Python Objekte benutzbar, sind allerdings kompilierter Code (sie sind also ähnlich zu build-in object types).

Des Weiteren sind sie sehr nützlich, um externen C/C++ Code zu wrappen, und geben damit ein Python like Interface ab. Darüber hinaus sind Extension Types relativ schnell.

## Wie unterscheiden sich Extension-Data-Fields in Cython von Data-Fields in Python Klassen?

Die Extension-Data-Fields in Cython sind nur in Cython Code zugänglich. Der Zugriff von Python ist nur mit der Deklaration des Zugriffs, z.B. readonly oder public, möglich. Es gilt des Weiteren, dass die Data-Fields von vornherein auf der Klassenebene deklamatiert werden müssen. Man kann außerdem Data fields deklarieren, welche wiederum Extension-types sind. Für Data Fields in Cython gilt zusätzlich, dass sie schnell sind.

## Beschreiben Sie kurz. Wie man C/C++ Code in Cython wrappet.

Zuerst muss der C/C++ Code wie gewöhnlich als .cpp Datei und .h Datei vorliegen. Anschließend werden in einem .pyx Cython Datei die entsprenden C/C++ .h Header Dateien importiert. Des Weiteren muss den Cython-Compieler entsprechende Direktiven erteilt werden, um unter andrem festzulegen das ein C/C++ programm gewrappet wird. Es müssen auch all erforderlichen C/C++ Datentypen importiert werden. Die zu Wrappende C/C++ Funktion mit dem „cdef“ primitiv festgelegt und entspricht dem Funktionsnamen im entsprechenden Header. Anschließend wird die Funktion mit „def“ als Python-Funktion definiert. Diese Python-Funktion kann dann in jedem Python Programm, dass das Cython-Modul importiert, aufgerufen werden.

# Exam Assignments 13

## Grenzen Sie die folgenden SSD-Komponenten voneinander ab: Cells, Pages, Blocks

Cells: In einer SSD ist eine Zelle die kleinste Einheit, in der Daten gespeichert werden. Pro Zelle, werden je nach Anwendungsfall, zwischen 1 und 4 Bits gespeichert. Im Allgemeinen sind mehr Bits pro Zelle billiger, aber weniger haltbar.

Pages: Die kleinste Einheit, die geschrieben oder gelesen werden kann sind Pages. Im Allgemeinen enthält eine Seite 4 KB an Daten. Interessanterweise gibt es keinen Überschreibvorgang für Seiten, so dass geänderte Daten als geänderte Kopie auf einer anderen Seite gespeichert werden. Geänderte Seiten werden als "veraltet" markiert und können später gelöscht werden.

Blocks: Die kleinste Einheit des Löschens ist der Block. In der Regel enthält ein Block 128 oder 256 Seiten, was 512 KB bzw. 1 MB entspricht.

## Was ist der Zweck von Garage-Collection in SSDs?

Da Datenseiten auf SSDs nicht einzeln gelöscht werden können, kann eine hohe Anzahl von ungültigen (aktualisierten) Seiten auftreten, insbesondere wenn sie auf verschiedene Blöcke verteilt sind. Dies führt zu einer großen Menge an "totem Speicherplatz". Wenn es mehr ungültige Seiten als einen bestimmten Schwellenwert gibt, werden die gültigen Seiten in einen anderen Block verschoben, um den alten Block zum Löschen freizugeben.

## Was ist der Zweck von Wear-Leveling (Abnutzungsausgleich) in SSDs?

Bei SSDs werden die Daten durch die Spannung in einer Zelle kodiert. Mit der Zeit wird es immer schwieriger, 0s von 1s zu unterscheiden (dies erklärt auch, warum Zellen mit nur einem Level haltbarer sind - da es mehr Schwellenspannungen gibt). Um zu verhindern, dass ein Teil der SSD schneller ausfällt als andere, werden die Daten auf verteilte Seiten auf der SSD geschrieben, was vom Controller über den Flash Translation Layer (FTL) gesteuert wird. Datenblöcke, die lange Zeit bestehen, werden regelmäßig gelesen und neu geschrieben, um sicherzustellen, dass die Seiten nicht unterausgelastet sind.

## Erklären Sie einige interessante Dinge über SSDs mit M.2-Formfaktor Faktor.

Für M.2 SSDs kann NVMe als Kommunikationsprotokoll über die Host Schnittstelle PCIe verwendet werden, die im Vergleich zu SATA-Schnittstellen viel leistungsfähiger ist, die AHCI als Kommunikationsprotokoll verwenden. NVMe bietet höhere Bandbreiten, sowie größere Übertragungsraten. Es gibt aber auch M.2-SSDs, die über SATA verwendet werden über SATA verwendet werden, die nicht die bessere Leistung bieten, so dass der Formfaktor allein nicht unbedingt auf den verwendbaren Standard schließen lässt. Für die Konnektivität sind auch die Anschlusssteckplätze relevant. Es gibt B-Key-, M-Key- und B&M-Key-Anschlüsse mit unterschiedlichen Pin-Anordnungen, die ihrerseits PCIe oder SATA unterstützen können oder auch nicht.

## Welchen Einfluss haben Garbage Collection und Wear Leveling auf die Schreibverstärkung einer SSD?

Die Schreibverstärkung beschreibt den Faktor der Daten, die logisch geschrieben werden müssen gegenüber den Daten, die physisch auf die SSD geschrieben werden. Garbage Collection und Wear Leveling beeinflussen beide die write amplification negativ. Durch Wear Leveling werden Schreibvorgänge über verschiedene Blöcke und Seiten verteilt. Wenn einige dieser Seiten aktualisiert werden, d. h. kopiert, geändert und auf eine andere Seite geschrieben werden, kann die Garbage Collection eingesetzt werden, um die Blöcke neu zu schreiben. Da der physische Speicher in Betriebssystemen im Allgemeinen durch logische Dateien abstrahiert wird, könnte dies bei logisch verbundenen Daten, die physisch auf verschiedene Blöcke verteilt sind, möglicherweise negative Synergieeffekte haben. Das heißt die Aktualisierung kann zu mehr Garbage Collection führen. Dies könnte jedoch vom Controller berücksichtigt werden. Ein weiteres Szenario, bei dem das Wear Leveling zu einer höheren Schreibverstärkung führt, ist die Neuzuweisung von Dateien, die schon lange bestehen.

## Besprechen Sie drei verschiedene Vorschläge zum Schreiben von Code für SSDs.

**Verwendung kompakter Datenstrukturen:** Es ist besser, Dateien zu verwenden, die mindestens 4 KB oder Vielfache von 4 KB groß sind. Dies liegt daran, dass keine Lese- oder Schreiboperationen für Einheiten, die kleiner als Seiten sind, durchgeführt werden können. Die Verwendung einer großen Datei, anstelle mehrerer kleinen Dateien ist ebenfalls leistungsfähiger, da die Daten besser von mehreren Threads gelesen werden können und die Bandbreite optimal genutzt wird.

**Multithreading für kleine IO:** Wenn Daten nicht in großen Dateien gespeichert sind, ist es vorzuziehen, 16-64 Threads für die Aktionen zu verwenden. Wie im ersten Vorschlag erwähnt, führt dies zu einer angemessenen Nutzung der Warteschlange der SSDs und der verfügbaren Bandbreite. Die konkrete Anzahl der Threads hängt von der SSD und ihrem Kommunikationsprotokoll ab.

**Multithreading für große IO (10 MB):** Da SSDs über einen internen Parallelisierungsmechanismus verfügen, sind nicht so viele Threads (2-4) erforderlich, um IO auf größeren Dateien durchzuführen. Eine zu hohe Anzahl von Threads führt tatsächlich zu einem verringerten

Durchsatz.

## Wie kann die CPU-Last für IO reduziert werden?

Synchrone IO-Vorgänge auf dem SSD blockieren Threads auf dieselbe Weise wie ein Spinning Lock sonst zu einer hohen CPU-Belastung ohne jeglichen Nutzen führen würde. In diesem Fall kann die Verwendung asynchroner Aufrufe an das System die CPU-Last erheblich reduzieren. Streaming-Stores und -Reads können die CPU-Belastung auch dadurch verringern, dass auf Zwischenspeicherung verzichtet wird und dies der Anwendung überlassen. Der letzte Punkt gilt hauptsächlich für DBMS.

# Wie gestaltet man einen Algorithmus um, der mehr Platz benötigt als durch den DRAM zur Verfügung gestellt wird, ohne den Programmcode grundsätzlich zu verändern?

Es können NVMe-SSDs genutzt werden, um nahezu speicherinterne Geschwindigkeiten für Berechnungen bereitzustellen. Subramanya et al. erreichten dies durch Pipeline-IO (Laden der nächsten Daten während der Berechnung) mit den Berechnungen einer linearen Algebra-Bibliothek. Das Pipelining kann die Latenz von SSDs im Vergleich zum RAM-Zugriff teilweise verbergen. Ein ähnlicher, einfacher Ansatz zur effizienten Nutzung großer Mengen von Flash-Disk-Daten sind Memory-Mapping-Dateien. In diesem Fall können die Dateien im Code als gewöhnliche Arrays interpretiert werden, wobei das Betriebssystem alle speicherbezogenen Fragen behandelt.