# APUVS, Blatt 1

Jan Fajerski and Kai Warncke and Magnus Müller

31. Oktober 2010

## Aufgabe 1.3

Das Programm lässt sich nicht eindeutig einem Dwarf zuordnen. Wir identifizieren folgende Dwarfs in der Problemstellung:

Dwarf 1 Die Daten sind als dich besetzter Vektor abgespeichert und die Arbeit wird durch das Aufteilen von kontinuierlichen Speicherstücke verteilt.

Dwarf 8 Summierung implementiert in diesem Fall eine Hashfunktion/Checksumme

Dwarf 10 Das Aufsummieren der Arrayelemente wird durch das Zerlegen in kleinere Teilprobleme parallelisiert.

**Dwarf 1** *Ich bin mir nicht sicher, wie man den begründen kann. Vgl. `http://fluid.stanford.edu/~barad/teaching/cme212_winter2009/Lecture03.pdf`: Diese charakterisieren *Dense Linear Algebra* als Komputation auf Matrizen (Matrixoperationen). Wir wenden hier ja eigentlich keine Matrixoperationen an, oder?*

**Dwarf 8** Vgl. hierzu [Wik10]:

> A hash function is any well-defined procedure or mathematical function that converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index to an array (cf. associative array). The values returned by a hash function are called hash values, hash codes, hash sums, checksums or simply hashes.

In diesem Sinne kann man das Aufsummieren der Feldelemente als Bildung einer Checksumme verstehen, was unter die Klasse der Kombinatorischen Algorithmen (Dwarf 8) fällt.

**Dwarf 10 - dynamic programming** Dwarf 10 wird in [ABC+06, S. 16] folgendermaßen charakterisiert:

> Computes a solution by solving simpler overlapping subproblems. Particularly useful in optimization problems with a large set of feasible solutions.

Das Aufteilen des Arrays in kleinere Stücke, auf denen dann jeweils das Problem gelöst wird, stellt also solch eine dynamische Programmierung dar.

# Aufgabe 1.4

**a)** SIMD — Die gleichen Instruktionen werden parallel auf verschiedenen Daten ausgeführt. — *data level parallelism.*

**b)** Wie im Protokoll des Skripts `call.sh` in Aufgabe 1.2 zu sehen, bringt in dieser Problemstellung die Parallelisierung keinen nennenswerten Vorteil (zumindest nicht, solange nicht die Problemgröße in Form der Größe des Arrays eine gewisse Grenze überschreitet). Daher könnte man sinnvollerweise das Problem durch eine *SISD* Architektur umsetzen. Die Ausführung der Summation erfolgt also auf einer einzelnen Recheneinheit schneller als bei der Verteilung auf verschiedene Einheiten.

# Literatur


[ABC+06] Asanovic, Krste ; Bodik, Ras ; Catanzaro, Bryan C. ; Gebis, Joseph J. ; Husbands, Parry ; Keutzer, Kurt ; Patterson, David A. ; Plishker, William L. ; Shalf, John ; Williams, Samuel W. ; Yelick, Katherine A.: The Landscape of Parallel Computing Research: A View from Berkeley / University of California at Berkeley. 2006. – Forschungsbericht

[Wik10] Wikipedia: *Hash function — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Hash_function&oldid=392346095`, 2010. – [Online; accessed 31-October-2010]