WS 24/25, PVS-Master - Ablauf

=> nur noch Inf-Stud im prakt. Teil (AI->PVS->derzeit Pflichtmodul)

=> trotzdem APL-Abschluss angestrebt/angeboten

=> bewertbare Eigenleistungen:

1.1 "1.HA" (im V-Kontext 2/HW Folie 13)
... Stefan, Dorian, Tim, Leon: bereits abgegeben, praesentiert & diskutiert
... Nachreichung angeboten

1.2 "Fingeruebung" auf Basis V-Kapitel 4/Metrik (Folie 1)
1.2.1 exp. FLOPs/sek.-Bestimmung für eine selbst gewählte Plattform
1.2.2 exp. Latenz- + TR-Bestimmung für eine selbst gewählte Plattform
(TR = Transferrate in Bytes/Sekunde)

Es folgen einige Erfahrungshinweise aus dem Kurs des Vorjahres
zu 1.2:

Für 1.2 ist zunächst eine grobe Vorabbeschreibung der selbst gewählten Experiment-Plattform erforderlich (heute häufig Laptops; in WS23/24 waehlten viele Studs aber auch ihre leistungsfähigen Gaming-Desktops.

Dabei stellte sich heraus, das vielen die Begriffe: Threads vs. Hyper-Threads bezügl. ihrer Prozessoren noch gar nicht klar war.

Hierzu für WS 24/25 eine kleine Hilfestellung:
Der Begriff Threads oder auch Hyper-Treads im Zusammenhang mit Multi-Core-Prozessoren ist nicht eindeutig definiert !
Bei manchen sogenannten Multi-Threaded-CPUs ist die FPU oder sogar die Basis-ALU pro Core nur einmal vorhanden !
(s. WP-DE "Mehrkernprozessor" für erstes Einlesen ...)

(Bei der Abfrage der verwendeten Experiment-Plattform bezogen wir im WS 23/24 die Frage nach dem GPU-Co-Prozessor mit ein.
Einigen Studs, die Laptops verwenden war nicht klar, dass in diesen Geräten (mit nur geringer Grafikleistung) gar kein separater GPU-Co-Prozessor-Chip verbaut ist und das der vorhandene GPU-Co-Prozessor mit auf dem Die des CPU-Prozessors integriert wurde.
In WS 24/25 sollten wir die Frage GPU erst mal komplett draußen lassen!)

In WS 24/25 sollten wir allerdings beachten, dass inzw. in manchen CPU-Prozessoren nicht nur identische Cores ("full" Thread + Hyper-Thread) verbaut sind, sondern u.U. auch einige Cores "volle Leistung" + einige Cores "verminderte Leistung" <- zwecks laengerer Akku-Laufzeit bei mobilen Geraeten.

Erfahrungshinweis zu 1.2.1 FLOPs/sek.-Bestimmung:
Im WS 23/24 zeigte sich, dass teilweise nur geringes GTI-Wissen zu Zahlendarstellungen auf Prozessor-Ebene vs. Programmiersprachen vorhanden war !
Hier ein kleiner "Nachhilfe-Exkurs" für WS 24/25:
Unter FLOP = Floting Point Operation wird traditionell die Addition oder Multiplikation einer Gleitkommazahl verstanden.

FLOPS oder FLOP/s ist damit ein Leistungsmaß (Floting Point Operations per Second).

Es gibt unterschiedliche Darstellungsgenauigkeiten von Gleitkommazahlen auf Computerarchitekturen:
... einfache Genauigkeit (single precision) = 32 Bit (heute = 4 Byte)
... doppelte Genauigkeit (double precision) = 64 Bit (heute = 8 Byte)
... vierfache Genauigkeit (quad precision) = 128 Bit (heute = 16 Byte)
Es gibt auch sogenannte *Minifloats*:
... halbgenaue Zahlen = 16 Bit (heute = 2 Byte)
... und 8 Bit (heute = 1 Byte)
Minifloats spielen in allg. HPC-Anwendungen keine Rolle.

In der HPC-Standard-Sprache Fortran gilt:
... REAL(8), REAL(16), REAL(32), REAL(64), REAL(128)
... daneben wird auch COMPLEX(*) unterstützt

Für C (und C++) gilt (in der Regel):
... float = 32 Bit
... double = 64 Bit
... long double = ... kann variieren !
... üblich:Prog.sprache/Übersetz.sys/Laufzeitsys:
80 Bit (10 Byte) gemäß *Extended Precision Format* / IEEE 754
128 Bit (16 Byte) gemäß *Quadrupel Format* / IEEE 754
... neuere Standards für C- und C++ unterstützen
auch komplexe Gleitkommazahlen

Bei anderen Sprachen als C genau hinsehen & berichten !!!

Wir sollten unsere Studie auf REAL(64) = doppelte Genauigkeit konzentrieren !!!

Meine Empfehlung für die "FLOPs/sek.-Fingeruebung" in WS 24/25:
Benutzt für Euer Experiment lieber erst mal C double ... da könnt Ihr derzeit mit hoher Wahrscheinlichkeit davon ausgehen, dass Ihr tatsächlich das übliche FLOPs/sek.-Maß bestimmt (64-Bit-Gleitkomma-Zahlen).

BTW: Ein Stud in WS 23/24 benutzte die Sprache Rust + teilweise Inline-Assembler (Richard/IET-Stud) und konnte damit auch die enorme Leistungsfähigkeit der SIMD-Fähigkeit ganz normaler CPU-Cores aufzeigen (Stichworte: MMX, SSE, etc.)
Eine solche Fähigkeit ist bei unseren Studs heute nat. selten anzutreffen, weil Ihr Basiswissen über Compterarchitektueren in Eurem Curr. nicht mehr hoert :(

Weiterer Erfahrungshinweis zu 1.2.1 FLOPs/sek.-Bestimmung:
Falls Ihr gar keine Idee zur Ermittlung der "Vorab-Spec." Eurer Testplattform habt, hier ein interessanter Link aus WS 23/24:
(ebenfalls von Richard beigetragen)
https://setiathome.berkeley.edu/cpu_list.php
(Vielleicht findet Ihr Euren Prozessor in dieser DB.)

An dieser Stelle sei nochmals erwähnt, dass Ihr das FLOPs/sek.-Maß Eurer gewählten Testplattform natürlich auch über einen standardisierten Benchmark ermitteln könnt.
Leider sind diese Benchmarks über die Jahre recht komplex geworden.
Hier ein kleiner "Exkurs" für WS 24/25 zum ersten Einstieg in dieses Thema:

(nur bei Interesse relevant)

Whetstone ... "nasser Schleifstein" ... einer der Ur-Vaeter zur Bestimmung der Rechenleistung von Computer-HW ... Entstehung ca. 1972 zunächst in ALGOL 60 ... bis heute weitergepflegt ... inzw. ausführbar in allen relevanten Prog.sprachen.
Der "nasse Schleifstein" versucht gleichzeitig die Integer- und die FLOP-Leistung von Computer-HW zu messen.

Dhrystone ... "trockener Schleifstein" ... aehnlich Whetstone ... beschraenkt sich aber auf die reine Integer-Leistung von Computer-HW.
Entstehung ca. 1984 ... Erstimpl.sprache habe ich nicht mehr auf dem Schirm (vermutl. VAX 11/780 Assembler) ... WP-EN meint, dass es zunaechst eine Mischung aus Fortran-, PL/1-, SAL-, ALGOL68- und Pascal-Programmen war, was kein Widerspruch ist, weil alle genannten Sprachen -> Compiler !
Ich habe die Weiterentwicklung von Dhrystone *nicht* bis heute verfolgt ... derzeit unklar, ob akt. Dhrystone-Vers. verfügbar sind ???

LINPACK-Benchmarks ... erste Version 1979 ... von **Dongarra, Bunch, Moler, und Stewart** (diese Namen sollten inzw. jedem Inf und Ing, die sich mit HPC befassen, bestens bekannt sein !)
Wie schon aus dem Namen des Benchmarks erkennbar, ist die primäre Impl.sprache nat. Fortran (hochopt. Compiler -> dann Assembler der jeweiligen HW).
Untersuchungsgegenstand ist auschließlich FLOP-Leistung.
Std-Benchmark für TOP-500 ... durchgehend auf **netlib.org** dokumentiert.
Leider sind die LINPACK-Benchs aus meiner Sicht selbst für Master/ 2.Sem. inzw. zu komplex :(

Das Einlesen zu den o.g. Std-Benchs ist (Stand 2023) via WP möglich.
Man sollte aber sowohl DE als auch EN lesen !
(Zu LINPACK-Benchmarks habe ich Stand 2023 nur in der WP-EN einen Eintrag gefunden.)

Erfahrungshinweise zu 1.2.2
exp. Bestimmung von Latenz + TR für eine selbst gewählte Plattform (TR = Transferrate in Bytes/Sekunde):
Hierzu empfehle ich anstelle von Benchmarks eher auf das erworbene Wissen im Studium zu setzen (ENP, 6. Sem. Socket-Programmierung und Anwendung auf einen RTT-Test; RTT = Round TripTime, s. 4/Metrik, Folie 1).
Um den Aufwand möglichst gering zu halten,
bietet sich hier die Ausmessung des sogenannten "Loopback-Device" an,
was prakt. auf allen OS zur Verfügung steht,
via C/Sockets/TCP und/oder UDP (oder auch IP=Raw-Sockets) an.
(Hinter dem "Loopback-Device" steht auf den heute relevanten OS in der Regel physischer RAM via Unix-Domain-Protokoll.)