Sheet für Präsentation/ Punkte der schriftlichen Ausarbeitung

Einleitung:

* Celeste
  + said Viral Shah, CEO of Julia Computing. “Researchers can now focus on problem solving rather than programming.”
  + “With Celeste, we are closer to bringing Julia into the conversation because we’ve demonstrated excellent efficiency using hybrid parallelism – not just processes, but threads as well – something that’s still impossible to do with Python or R.”
* (Auch möglich: Partner aufzählen wie Intel,… und Frage stellen, was an der Sprache ist, das sie so besonders macht und diese Unternehmen auf sie setzen)

Julia Tutorial:

* Def Julia
* Mit Konzept LLVM, multiple dispatch, und Typenbaum starten oder lieber später einschieben?
  + Für multiple dispatch könnte man auch Machine Code kurz zeigen
    - Multiple dispatch = Funktionen sind prinzipiell generisch und haben teilweise über 100 implementierungen (@which +)
    - Julia Compiler optimiert für bestmöglichen Typ hin und kann so einen enormen Speedup bekommen -> Typenangabe beim Programmieren sehr wichtig
    - Speedup vorführbar bzw zahlen nennbar siehe JuliaBox.com
  + LLVM und wie Julia compiliert zu Tief?
    - Typenbestimmung und Ersetzung dann LLVM via JIT und zum Schluss Compilieren für Performance zu Maschinencode \*\*
    - Julia Compiler arbeitet „intelligent“ und spart sich unnötige checks wie zb bei logischen Bedingungen (false && b ) wird b niemals aufrufen (nicht wichtig)
    - Auch wenn dynamische Sprache, Typenangabe ist „feste Definition“ für Variablen und Funktionen und sorgt für große Performance boosts, da der Compiler nicht mehr dagegen checken muss
    - Julia Code wird nur einmal compiliert, wenn man Code danach erneut aufruft, ist er deutlich schneller -> erklären warum?
  + Typenbaum nur im Ausschnitt, aber wichtig, da notwendig zu verstehen wie Typen funktionieren in Julia
    - Grundstruktur + Typ Any und None kurz ansprechen und dann zu Int, String, Float, Char, Byte,… und dazu auch kurz etwas zu BigInt und BigFloat sagen
* Soll ich nun einfache Codebeispiele vorbringen oder in der Form weitervorgehen um bestimmte Sachen zu Funktionen, Schleifen und Pakten zu sagen?

Nur wo nötig

* + Zusätzlich mögliche Themen wären auch Collections, Metaprogramming, (I/O) und anschnitt zu parallele Programmierung wobei ich dies eher günstig für die Überleitung halte
  + External Libraries/Programs wie Python oder C Code
  + Macros (@ Notations)
* Zu Macros, Metaprogrammierung und I/O kann ich bisher nicht viel sagen (kommt im einsteiger Tutorials nie vor und im Buch erst weiter hinten
* Zu external Programs/Libraries wäre zu sagen das es echt einfach zu gehen scheint, siehe JuliaBox und sogar vorführbar wäre, großer Pluspunkt für alle Umsteiger, da bestehender Code einfach genutzt werden kann
  + Als nebensatz max ein Stichpunkt, nur das es geht und einfach ist
* Collections ähnlich zu Funktionen und Schleifen sicher auch ganz interessant, genauso wie Einstieg in lineare Algebra um Ähnlichkeit zu MatLab sehen zu können nur ein kurzer Punkt (alles Punkte für Umstieg
  + Prinzipiell aber eher unspektakulär, da so erwartet
* Auch möglich wäre eine komplexere Aufgabe in Julia gelöst zu zeigen um auch vorzuführen wie einfach es ist, schnellen JuliaCode zu schreiben, wofür meiner Meinung nach Julia vorallem steht -> guten HandsOn Code
  + zb implementierung von bestimmten Algorithmen wie Levenshtein oder Viterbi oder Dijkstra wäre möglich

zum 2. Teil bisher noch nicht viel gemacht, ich habe aber gemerkt, das Fallstricke(3. Punkt) besser jeweils an den Stellen passt und nicht nochmal als eigener Punkt, da es da so losgelöst wirkt und ich eher direkt an der Stelle begründen können möchte, wieso es so ist

als Überleitung würde ich den Einstieg in die parallele Programmierung nehmen, d.h. ich gehe kurz auf grundsätzliche Möglichkeiten ein, wie man Julia parallel schreibt am Beispiel von Code

message parsing über knoten hinweg oder shared memory programming

am Ende von Vortrag eine Art Resultat ziehen zusammen auch sagen? So eine Art Gegenüberstellung Pro-Cons und auf Anfang nochmal wieder eingehen und darauf Antworten- Schaut euch das an, besser als python

Allgemeine Fragen: stand jetzt bin ich wahrscheinlich etwas hinter der Zeit oder?

* GitHub Project derzeit öffentlich? Soll ich das ändern, dann zu Bitbucket oder Gitlab umziehen
* Anteil 1:3 für Grundlagen:Parallel?
* Lesen sie immer wieder korrektur und wie sollte ich heran gehen, wenn der Vortrag zuerst fertig sein muss ja mit hinweis
* Gibt es neuigkeiten für mich zum Arbeiten am Rechenzentrum
* Nächstes Treffen in 2-3 Wochen
* Quellen/Lizenzen bei code aufpassen

Fallstricke sequenziell nicht mehr weiter, muss optimieren was kann falsch laufen

Speicheranalyse

Nur grobe Metaprogrammierung kaum macros, aber funktional und alles ausdrücke

Auf was muss man achten hardware, software wie gut kann man parallel programmieren

Anfang schon eher zu viel eher einkürzen 5 Folien, je nachdem wie interessant der rest ist weniger, 1 folie 1 minute

20 min

Grafiken gut erklären x achse das y achse das je höher je mehr… speedup diagramme

Research weiter betreiben 2-3-5 Codebeispiele aber nicht alles selbstgeschrieben

Maxtrix\*Matrix oder sowas

Mittwoch 2.5. um 9:00 Uhr Proseminar treffen

Geeignete Benchmarkprogramme

Problemvergrößerung wie verhält es sich mit mehr ressourcen (Skallierung) starke und schwache unterschieden—problemgröße gleich immer mehr recheneinheiten -> time to solution sinkt

Problemgröße pro parallele einheit konstant, mehr einheiten aber auch größeres problem -> gleich bleibt idealfall