2013/14

KLUNE Alexander

TGM

Aufgabe 04

GPGPU

Inhalt

Aufgabenstellung 2

Quellen 2

Aufgabenaufteilung 3

Zeiteinteilung 3

Geschätzte Zeit 3

Tatsächliche Zeit 3

Designüberlegung 4

Frage 1 5

Frage 2 5

Frage 3 5

Frage 4 5

Resultate 6

Niederlagen 6

Quellen 7

[1] Definition GPGPU, www.itwissen.info/definition/lexikon/GPGPU-general-purpose-graphics- processing-unit.html, zuletzt Aufgerufen 28.11.2013 7

# Aufgabenstellung

GPU Computing oder GPGPU(= General Purpose Computing on GPUs) bezeichnet die Verwendung eines Grafikprozessors (engl. Graphics Processing Unit oder GPU) für allgemeine Berechnungen im wissenschaftlich-technischen Bereich. Übersetzt bedeutet GPGPU in etwa Allgemeine Berechnung auf Grafikprozessoren.  
  
Informieren Sie sich über die Möglichkeiten der Nutzung von GPUs in normalen Anwendungen. Zeigen Sie dazu im Gegensatz den Vorteil der GPUs in rechenintensiven Implementierungen auf [1Pkt]. Gibt es Entwicklungsumgebungen und in welchen Programmiersprachen kann man diese nutzen [1Pkt]? Können bestehende Programme (C und Java) auf GPUs genutzt werden und was sind dabei die Grundvoraussetzungen dafür [1Pkt]? Gibt es transcompiler und wie kommen diese zum Einsatz [1Pkt]?  
  
Präsentieren Sie an einem praktischen Beispiel den Nutzen dieser Technologie. Wählen Sie zwei rechenintensive Algorithmen (z.B. Faktorisierung) und zeigen Sie in einem Benchmark welche Vorteile der Einsatz der vorhandenen GPU Hardware bringt [12Pkt]! Um auch einen Vergleich auf verschiedenen Platformen zu gewährleisten, bietet sich die Verwendung von OpenCL an.  
  
Diese Aufgabe ist als Gruppenarbeit (2) zu lösen. Es ist zu beachten, dass diese Aufgabe mit der Aufgabe05 gekoppelt ist, d.h. nur eine der beiden Aufgaben wird verpflichtend bewertet! Zusätzliche Abgaben erhöhen die Gesamtpunkte und können somit zur Notenverbesserung dienen.

### Quellen

<http://www.nvidia.de/page/gpu_computing.html>  
<http://developer.nvidia.com/cuda-gpus>  
<http://people.maths.ox.ac.uk/gilesm/cuda/>  
<http://www.khronos.org/opencl/>

# Aufgabenaufteilung

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| 4 Fragen beantworten | X | X |
| Algorithmus |  | X |
| Implementieren | X |  |
| Protokoll | X | X |
| Testing | X | X |
| Benchmarking | X | X |
| Gesamt | 5 | 5 |

# Zeiteinteilung

## Geschätzte Zeit

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| 4 Fragen beantworten | 1 h | 1 h |
| Algorithmus |  | 2 h 30 min |
| Implementieren | 4 h |  |
| Protokoll | 1 h | 1 h |
| Testing | 1 h | 1 h |
| Benchmarking | 2 h 30 min | 2 h 30 min |
| Gesamt | 9 h 30 min | 8 h |

## Tatsächliche Zeit

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| 4 Fragen beantworten | 1 h | 1 h |
| Algorithmus |  | 2 h 30 min |
| Implementieren | 4 h |  |
| Protokoll | 1 h | 1 h |
| Testing | 1 h | 1 h |
| Benchmarking | 2 h 30 min | 2 h 30 min |
| Gesamt | 9 h 30 min | 8 h |

# Designüberlegung

Das Design wurde aus den Beispielen von APARAPI übernommen. Diese hat gut funktioniert und die Implementierung stark erleichtert. Anfangs war eine Implementierung mit C/C++ angedacht, doch durch die vielen Schwierigkeiten die damit einhergingen, stiegen wir auf Java und APARAPI um.

# Frage 1

Informieren Sie sich über die Möglichkeiten der Nutzung von GPUs in normalen Anwendungen. Zeigen Sie dazu im Gegensatz den Vorteil der GPUs in rechenintensiven Implementierungen auf.

Die Graphic Process Unit ist ein wesentlicher Bestandteil der Grafikkarte der in fast jedem Computer eingebaut ist. Die Firmen Nvidia und ATI verwenden GPU um 3D Szenen als 2D Projektion zu berechnen, damit sie am Bildschirm bzw. Monitor dargestellt werden können. GPU dient im ursprünglichen Sinn der Video Rendering oder 3D Rendering bzw. der Berechnung von Visuellen Effekte zu berechnen und darzustellen. GPU werden in Mobilgeräten, Laptops, Desktops und in weiteren Geräten verwendet. Da die GPU die komplexen 3D und 2D Berechnungen durchführt wird die CPU entlastet. [1]

# Frage 2

Gibt es Entwicklungsumgebungen und in welchen Programmiersprachen kann man diese nutzen?

Entwicklungsumgebungen sind dieselben wie für das Programmieren auf der CPU, es wird lediglich ein Teil des Programms, welches parallelisierbar ist, auf die GPU ausgelagert. Es gibt Libaries, welche diese Ansteuerung ermöglichen.

Die gängige Sprache für die Programmierung von GPUs ist C oder ein Dialekt davon. OpenCL ist offiziell eine eigene Sprache, doch ähnelt sie sehr stark C und C++. Eine weitere Sprache ist „High-Level- Shading- Language.

# Frage 3

Können bestehende Programme (C und Java) auf GPUs genutzt werden und was sind dabei die Grundvoraussetzungen dafür?

Die Grundvoraussetzung für das effiziente Nutzen von Programmteilen auf der GPU, ist dass der Code parallelisierbar ist, das bedeutet er muss gleiche Operationen atomar auf alle Objekt anwenden, ohne dass diese Operationen von einem Ergebnis aus der vorangegangenen Operation aufbaut. Nur wenn das gegeben ist, kann die GPU durch ihre vielen Kerne schneller berechnen als die CPU mit ihren geringen, dafür stärkeren Kernen.

Sollte der Code parallelisierbar sein, kann dieser ausgelagert werden(Code dann in C oder andere geeignete Sprache).

# Frage 4

Gibt es Transcompiler und wie kommen diese zum Einsatz?

Ja es gibt Transcompiler bei GPGPU.

Es handelt sich dabei um Compiler die Quelltext einer bestimmten Programmiersprache in den Quelltext einer anderen Programmiersprache übersetzt. Diese werden bei GPGPU eingesetzt um den Quellcode GPU kompatibel zu machen und zur Performancesteigerung.

# Resultate

Die Aufgabenstellung wurde aufgrund der Niederlagen in C und C++ in Java gelöst. Das funktionierte ausgesprochen gut mit APARAPI. Es gab dabei kaum Probleme.

Die Algorithmen zum Benchmarken haben in kurzer Zeit funktioniert.

# Niederlagen

Wir haben zuerst versucht die Aufgabenstellung in C mittels VisualStudio zu lösen. Dadurch kam es zu massiven Zeitverlusten da wir es in C nicht lösen konnten.

Das Problem resultierte unter anderem daraus, dass C und C++ nicht gut mit OpenCL funktioniert.

Ein weiteres Problem war, dass Benchmarking. Dieses hat erst nach langem probieren funktioniert.

# Benchmarks

## Avg. / min / max

### Vector Multi

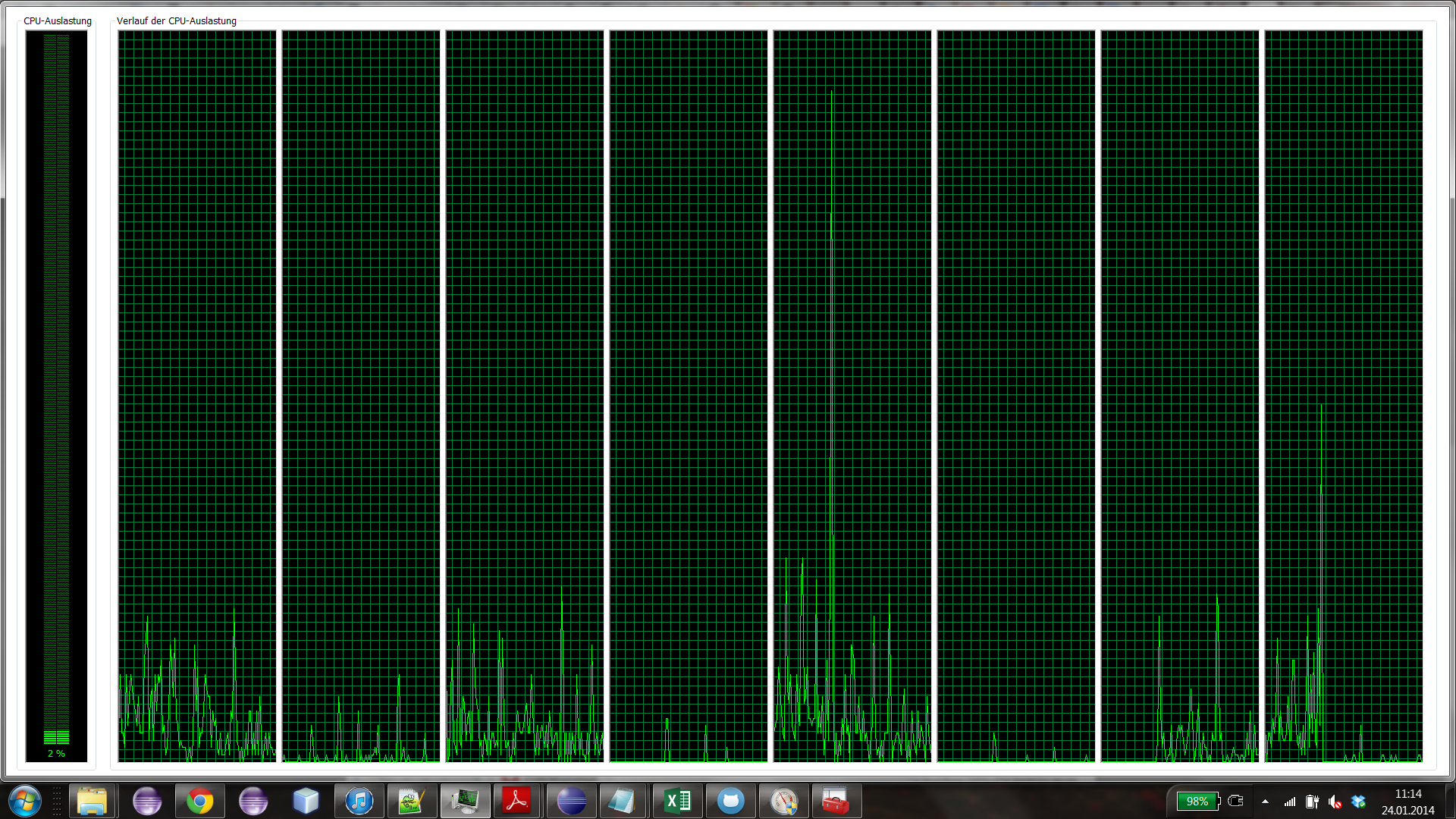
Durchschnittliche Rechendauer: 390 ms

Rechendauer Minimum: 360 ms

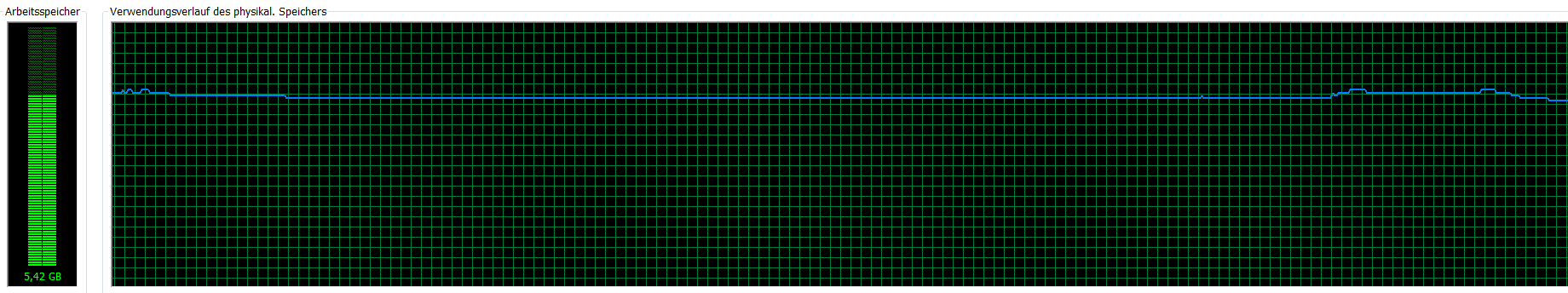
Rechendauer Maximum: 836 ms

## Mandel

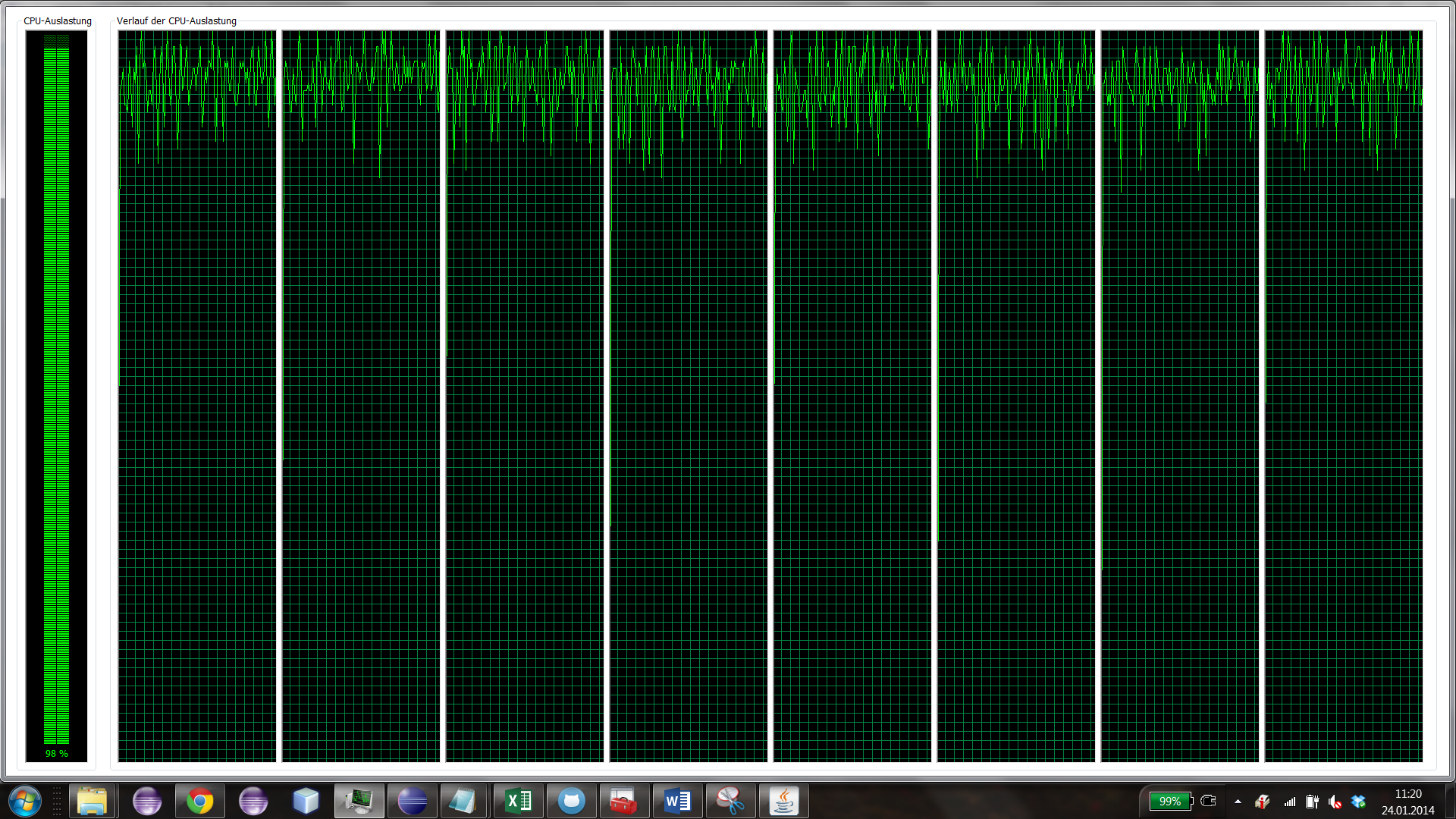
## CPU Leerlauf



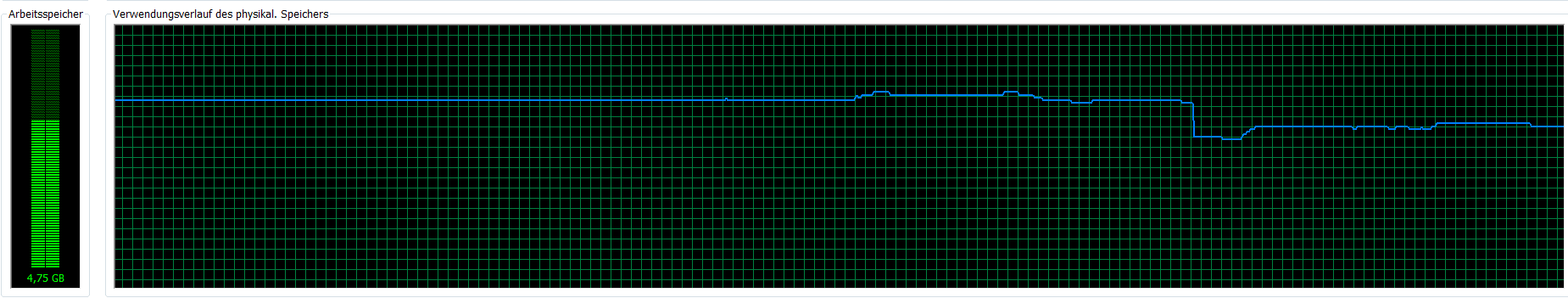
## RAM Leerlauf



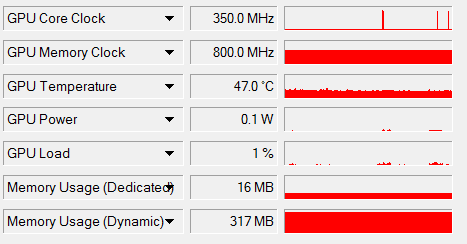
## CPU mit Mandelbrot



## RAM mit Mandelbrot



## GPU Leerlauf



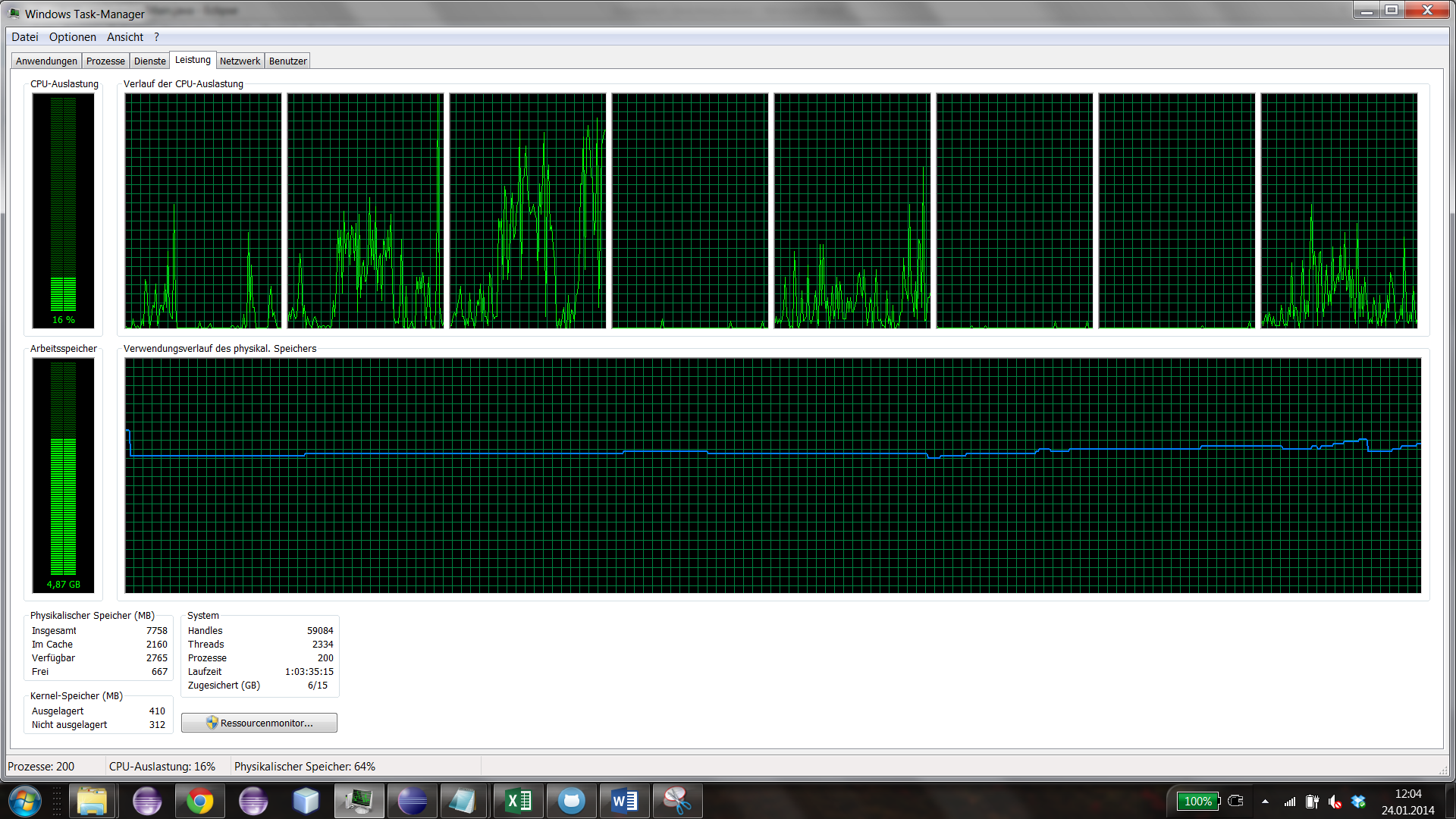
## GPU Mandelbrot incl. Speicher

Werte in „GPU\_Mandel.txt“

## GPU Vector Multi incl Speicher

Werte in „GPU\_Vector\_Multi.txt“

## CPU Vector Multi



# Quellen

[1] Definition GPGPU, [www.itwissen.info/definition/lexikon/GPGPU-general-purpose-](http://www.itwissen.info/definition/lexikon/GPGPU-general-purpose-) graphics- processing-unit.html, zuletzt Aufgerufen 28.11.2013