Betriebssysteme Praktikum 5 - Linux und Docker

Vorbereitung: Visual Studio Code für Docker Funktionalitäten installieren

```
sebastian@sebastian-VirtualBox:~$ sudo snap install --classic code
code 899d46d8 from Visual Studio Code (vscode✓) installed
```

Vorbereitung: Docker installieren

Siehe Anleitung https://docs.docker.com/engine/install/ubuntu/

1. Alte Docker Instanzen deinstallieren, falls vorhanden

```
sebastian@sebastian-VirtualBox:~$ sudo apt-get remove docker docker-engine docker.io containerd runc
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package docker-engine
```

```
sebastian@sebastian-VirtualBox:~$ sudo apt-get update
Hit:1 http://de.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://de.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease
Get:4 http://de.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Fetched 222 kB in 1s (229 kB/s)
Reading package lists... Done
```

2. Docker Repository erstellen:

```
sebastian@sebastian-VirtualBox:~$ sudo apt-get install \
>     ca-certificates \
>     curl \
>     gnupg \
>     lsb-release
Reading package lists... Done
Building dependency tree
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu2).
lsb-release set to manually installed.
ca-certificates is already the newest version (20210119~20.04.2).
ca-certificates set to manually installed.
gnupg is already the newest version (2.2.19-3ubuntu2.1).
gnupg set to manually installed.
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 161 kB of archives.
After this operation, 412 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://de.archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7.68.0-1ubuntu2.7 [161 kB]
Fetched 161 kB in 0s (763 kB/s)
Selecting previously unselected package curl.
(Reading database ... 184201 files and directories currently installed.)
Preparing to unpack .../curl_7.68.0-1ubuntu2.7_amd64.deb ...
Unpacking curl (7.68.0-1ubuntu2.7) ...
Setting up curl (7.68.0-1ubuntu2.7) ...
Processing triggers for man-db (2.9.1-1) ...
```

```
sebastian@sebastian-VirtualBox:~$  curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
 sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
sebastian@sebastian-VirtualBox:~$      echo \
>       "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive
-keyring.gpg] https://download.docker.com/linux/ubuntu \
>       $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/nul
l
```

3. Docker Engine installieren

```
sebastian@sebastian-VirtualBox:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 http://de.archive.ubuntu.com/ubuntu focal InRelease
Get:3 https://download.docker.com/linux/ubuntu focal InRelease [57,7 kB]
Get:4 http://de.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://de.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [13,5 kB]
Fetched 293 kB in 1s (214 kB/s)
Reading package lists... Done
sebastian@sebastian-VirtualBox:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

4. Docker standard sudo Rechte geben (für ausführen und stoppen über VScode):

```
sebastian@sebastian-VirtualBox:~$ sudo usermod -aG docker $USER
sebastian@sebastian-VirtualBox:~$ newgrp docker
sebastian@sebastian-VirtualBox:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:2498fce14358aa50ead0cc6c19990fc6ff866ce72aeb5546e1d59caac3d0d60f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Dockercontainer mit CPP-Programm aufsetzen

Siehe Anleitung:
https://devblogs.microsoft.com/cppblog/c-development-with-docker-containers-in-visual-studio-code/

1. Programm erstellen (CPUUsage nicht verwendet: Zu viel Zeitaufwand - stattdessen über „docker stats“ auslesen):

main.cpp

```cpp
#include <iostream>
#include "Storage.h"
#include "Manager.h"

int main()
{
    /* Init manager */
    Manager *manager = new Manager();

    Storage *storage = new Storage();
    /* Fill Vector */
    storage->fillVector();
    /* Read Vector */
    storage->readVector();
    /* Close programm */
    return 0;
}
```

Manager.cpp

```cpp
#include "Manager.h"        Jörg Quick, 2 hours ago • Added Manager class.

uint32_t Manager::amount;

Manager::Manager()
{
    amount = 0;
}

uint32_t Manager::getAmount()
{
    return amount;
}

void Manager::increaseAmount()
{
    amount++;
}
```

Manager.h

```cpp
#include <iostream>
#include <vector>

Jörg Quick, 2 hours ago | 1 author (Jörg Quick)
class Manager
{
public:
    Manager();

    static uint32_t getAmount();
    static void increaseAmount();

private:
    static uint32_t amount;

};
```

Storage.cpp

```cpp
Storage::Storage()
{
    storage = new std::vector<int>;
    Manager::increaseAmount();
}

void Storage::fillVector()
{
    for (int i = 0; i < VECTOR_SIZE; i++)
    {
        storage->push_back(i);
        std::cout <<  std::endl << "Storing data(" << i << ") in storage(" << Manager::getAmount << ")...";
        printUsages();
    }
}

void Storage::readVector()
{
    for (int j = 0; j < VECTOR_SIZE; j++)
    {
        std::cout <<  std::endl << "Reading data(" << storage->at(j) << ") in storage(" << Manager::getAmount << ")...";
        printUsages();
    }
}
```

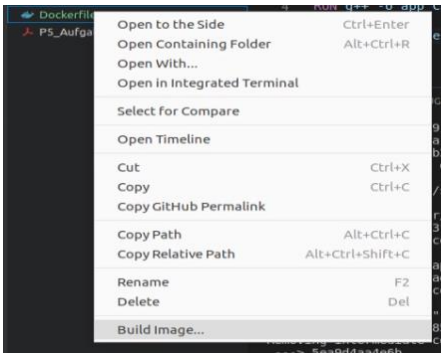
```cpp
uint64_t Storage::getMemUsage()
{
    struct rusage r_usage;
    uint32_t ret = getrusage(RUSAGE_SELF, &r_usage);
    if (ret == 0) return r_usage.ru_maxrss;
    return ERROR_CODE;
}

uint32_t Storage::getCpuUsage()
{
    CPUSnapshot previousSnap;
    std::this_thread::sleep_for(std::chrono::milliseconds(1000)); //Need 1000ms delay to read CPU usage. (Htop not!)
    CPUSnapshot curSnap;

    const float ACTIVE_TIME = curSnap.GetActiveTimeTotal() - previousSnap.GetActiveTimeTotal();
    const float IDLE_TIME   = curSnap.GetIdleTimeTotal() - previousSnap.GetIdleTimeTotal();
    const float TOTAL_TIME  = ACTIVE_TIME + IDLE_TIME;
    uint32_t usage = 100.f * ACTIVE_TIME / TOTAL_TIME;
    return usage;
}

void Storage::printUsages()
{
    const uint64_t memUsage = getMemUsage();
    //const uint32_t cpuUsage = getCpuUsage();

    if (memUsage != ERROR_CODE) std::cout << "Memory usage: " << memUsage << " KB.";
    //if (cpuUsage != ERROR_CODE) std::cout << "CPU usage: " << cpuUsage << "%.";
}
```

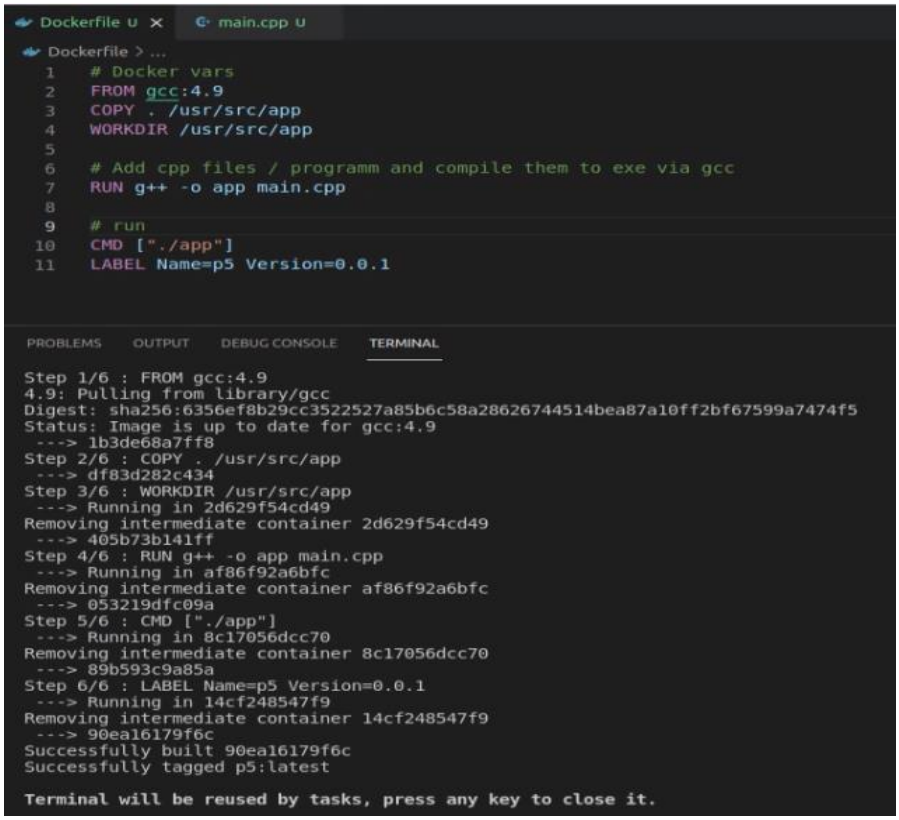
Storage.h

```cpp
#include <iostream>        Jörg Quick, 15 hours ago • Working on objects now.
#include <sys/resource.h>
#include <vector>
#include <chrono>
#include <thread>
#include "CPUSnapshot.h"

//==================================================
#define VECTOR_SIZE 10000000
#define ERROR_CODE -1
//==================================================

Jörg Quick, 2 hours ago | 1 author (Jörg Quick)
class Storage
{
public:
    Storage();

    void fillVector();
    void readVector();

private:
    std::vector<int>* storage;

    uint64_t getMemUsage();
    uint32_t getCpuUsage();

    void printUsages();
};
```

2. Docker Erweiterung für VScode installieren

3. Docker Buildfiles erstellen & Container bauen





Programm kann nun für Tests in Docker Container ausgeführt werden.

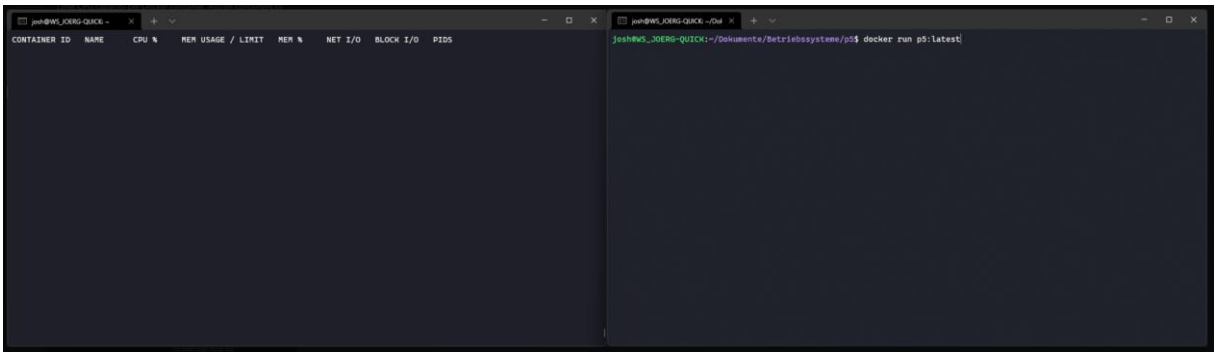
Aufgabenausführung: CPP-Programmausführung mit Docker Containern

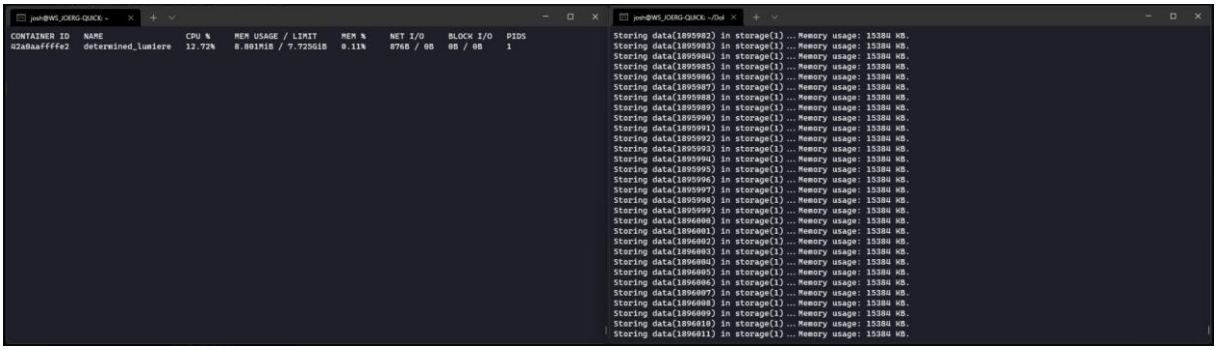
Umgebungsdaten: VM mit 4 CPU Kernen (außer für CPU-gewichtung - 1 Kern)

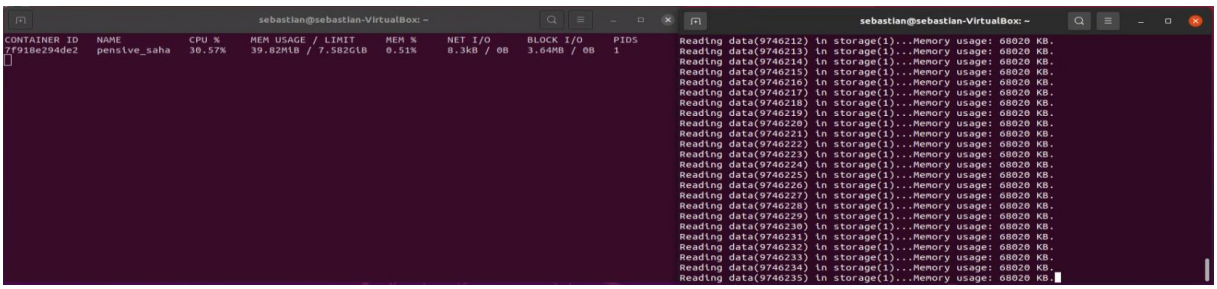
Umgebungsdaten auslesen: **`docker stats`**

Einschränkungen für Containerumgebung setzen siehe:
https://docs.docker.com/config/containers/resource_constraints/

0. Programm ohne Einschränkungen:
   `docker run p5:latest`



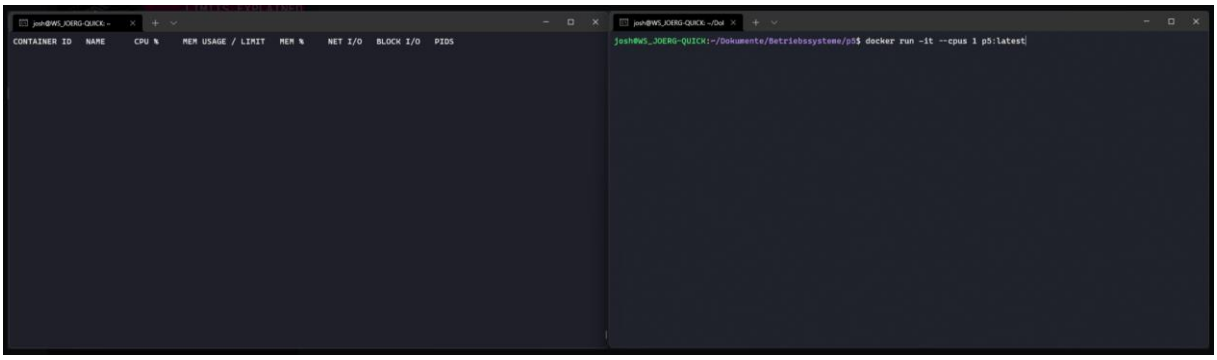


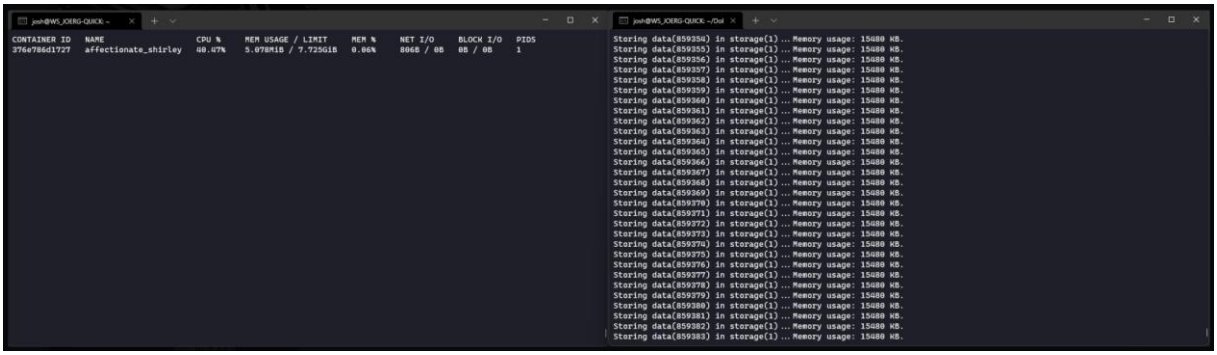


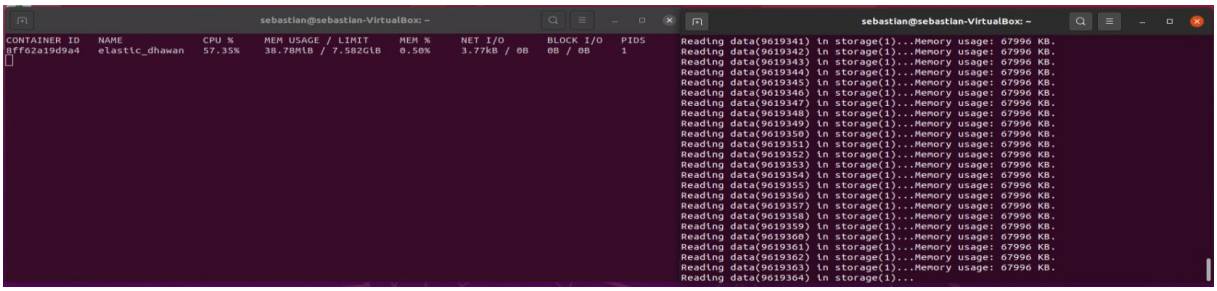
1. Programm mit Einschränkungen: Nur eine CPU per cpus="1" flag

a. Anteil des Containers an der CPU limitieren

`docker run -it --cpus="1" p5:latest`







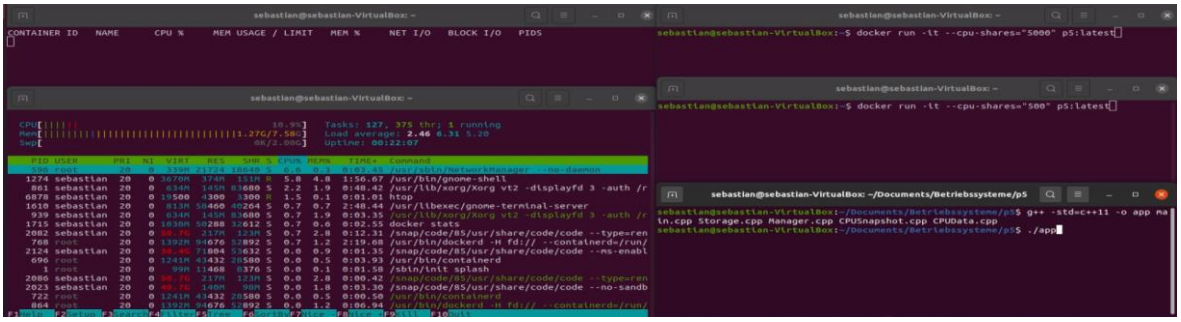
b. Gewicht (weight - default: 1024) im Vergleich zu anderen ändern
Notiz: Auf VM mit nur einem zugewiesenem Kern getestet, da laut Docker Docs bei mehreren Kernen das Programm auf diese verteilt wird.

Theoretisch ohne die Optimierung auch:

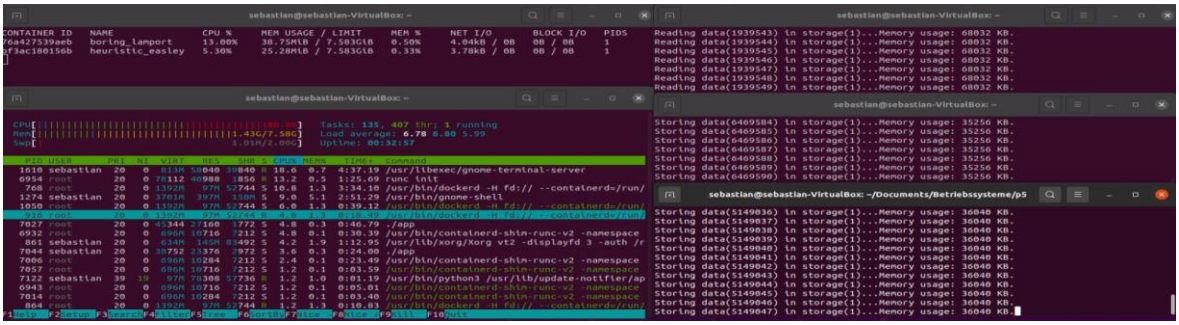
```
docker run -it --cpuset-cpus=”1” --cpu-shares 5000 p5:latest
docker run -it --cpuset-cpus=”1” --cpu-shares 500 p5:latest
```

Auf VM:

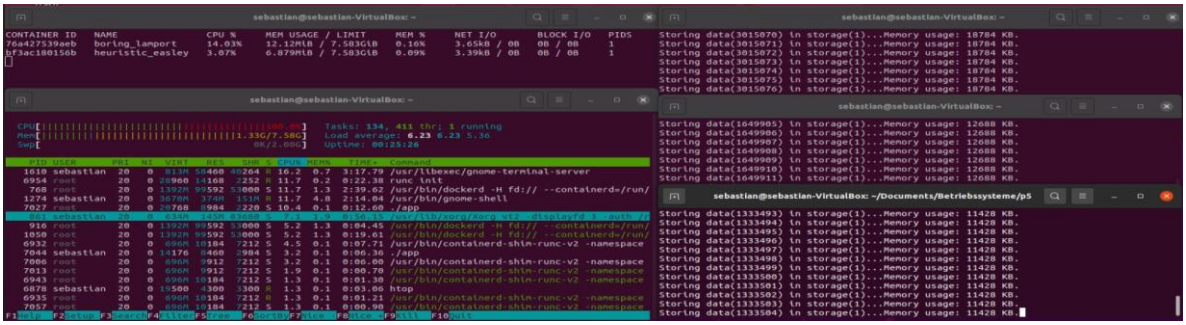
```
docker run -it --cpu-shares 5000 p5:latest
docker run -it  --cpu-shares 500 p5:latest
```



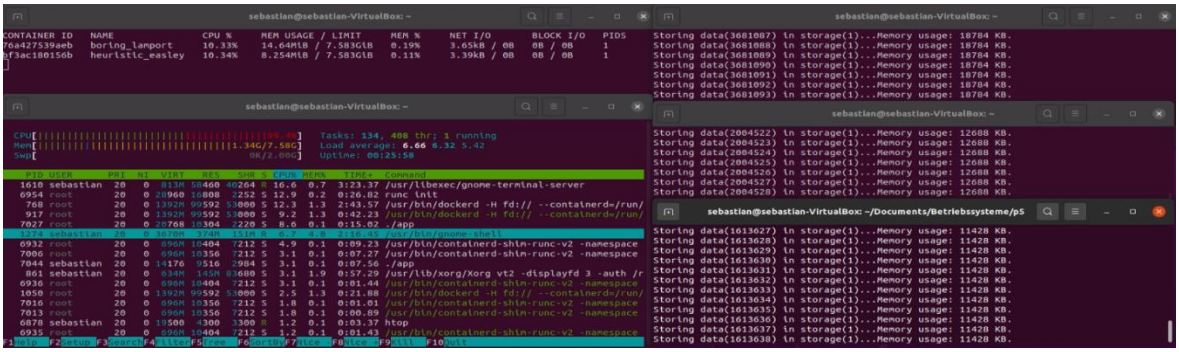
Großteil:



Extrem:



Teilw. auch kurz ausgeglichen:



2. Hauptspeicherverbrauch des Programmes limitieren

`docker run -it -m 8m p5:latest`