Applikation Profiling

Seminar Effiziente Programmierung

Universität Hamburg

07.01.2021

Chams Alassil Khoury

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Was ist Appliaktion Profiling?

- Methode für das Messen der Performanz einer Applikation
- Gibt Informationen über eine Applikation, wie z.B.
  - wie oft eine Funktion aufgerufen wird
  - wie viel die Ausführung einer Funktion dauert
- hilft, die Applikation oder das Programm zu optimieren

Warum benutzt man Applikation Profiling?

- Die Performanz der Applikation zu bewerten
- Die Anzahl der Probleme, die später in der Implementierung auftreten könnten, zu reduzieren
- Die Applikation zu verbessern und zu optimieren
- Zu bestimmen, in welcher Funktion es Effizienzprobleme gibt

Welche Komponente kann man profilen?

- Die Laufzeit bzw. die Geschwindigkeit der Applikation
- Die Speichernutzung
- Die Nebenläufigkeit
- Die Ein- und Ausgabe (I/O)
- Den Energieverbrauch
- Das Netzwerk

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Profiling-Methoden

- Zwei unterschiedliche Methoden
- Jede Methode verwendet unterschiedliche Strategien

Instrumentierung

- Wird auch Tracing genannt
- Für bestimmte Programmabschnitte wird ein Code hinzugefügt
- Der Code gibt, wann der Abschnitt ausgeführt wird, und sammelt Informationen
- Nachteil: der Code erhöht den Arbeitsaufwand der Applikation und das führt zur Steigerung der Laufzeit

- Die zusätzliche Zeit wird als Profiling-Overhead bezeichnet

- Der Profiling-Overhead darf nicht zu groß werden

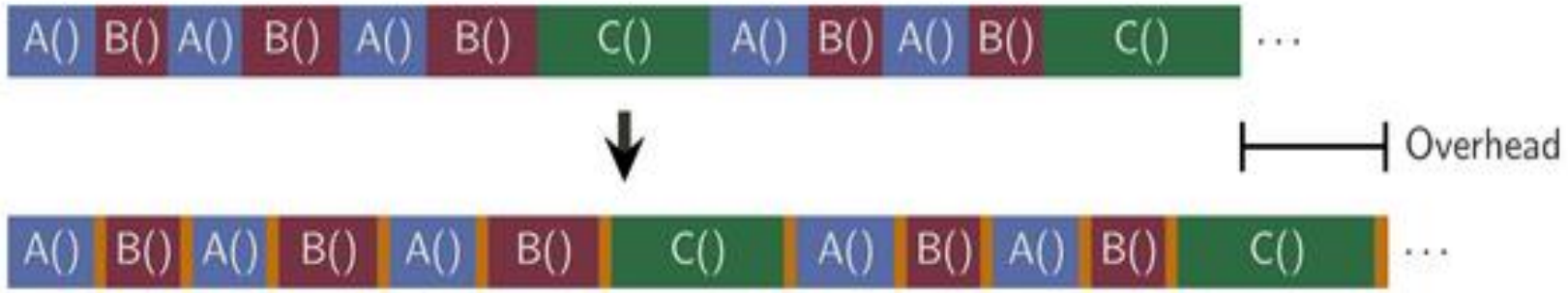

Abbildung 1: ein Beispiel für den Profiling-Overhead in der Instrumentierung

Phil Tooley, Application Performance Profiling: Part 2 – Choosing tools for Effective Profiling
https://www.nag.com/blog/application-performance-profiling-part-2-choosing-tools-effective-profiling
[19.12.2020/10:27]

Sampling

- Der Code wird nicht geändert
- Methoden, die vom Betriebssystem bereitgestellt, werden verwendet
- Betriebssystemsunterstützung ist erforderlich
- Der Profiler fordert den Betriebssystemkernel, die Applikation zu überprüfen und Informationen zu sammeln, z.B. welche Funktion jetzt ausgeführt wird und welche Hardwareressourcen verwendet wurde
- Der Profiling-Overhead ist vernachlässig

- Nachteile:

  kein deterministisches Verhalten, sondern statisches

  die Anzahl der Methodenaufrufe kann nicht angegeben werden

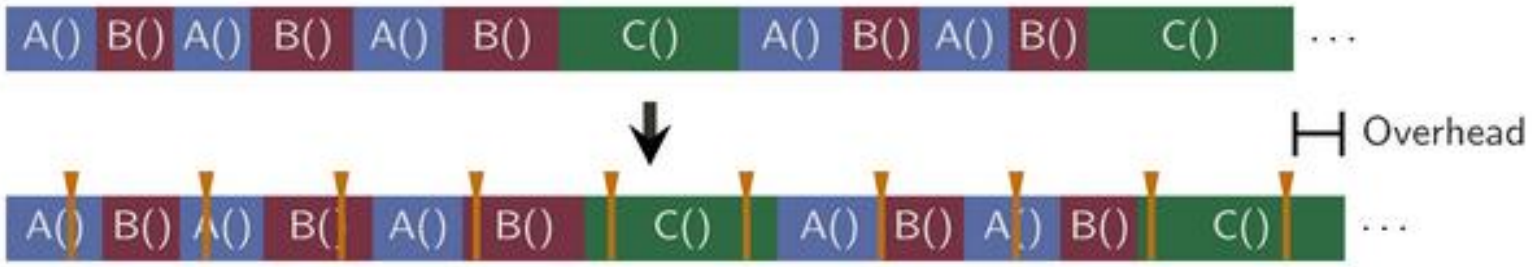

Abbildung 2: ein Beispiel für den Profiling-Overhead im Sampling

Phil Tooley, Application Performance Profiling: Part 2 – Choosing tools for Effective Profiling
https://www.nag.com/blog/application-performance-profiling-part-2-choosing-tools-effective-profiling
[19.12.2020/10:27]

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Laufzeit des Programms Profiling

- Die Aufrufe und Durchläufe der Funktionen zu zählen und zu messen
- Den Code zu optimieren
- Funktionen, die häufig verwendet und aufgerufen werden, werden untersucht, weil sie die Laufzeit beeinflussen können
- Viele Profiler stehen zur Verfügung

Gprof

- Instrumentierung, um die Funktionsaufrufinformationen zu sammeln
- Sampling, um Informationen zum Laufzeit-Profiling zu sammeln
- Für alle Sprachen, die von der GNU Compiler Collection (GCC) unterstützt werden
- zuerst das Programm mit der –pg Option kompilieren, danach das Programm ausführen
- Das Programm mit dem Befehl gprof analysieren und der Befehl gprof erzeugt ein Flat-Profile und ein Aufrufdiagramm

- Das Flat-Profile zeigt, wie lange die Ausführung jeder Funktion gedauert hat

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 84.04      0.85     0.85        1   848.84   848.84  yet_another_test
  6.00      0.91     0.06        1    60.63   909.47  test
  1.00      0.92     0.01        1    10.11    10.11  some_other_test
  0.00      0.92     0.00        1     0.00   848.84  another_test
```

Abbildung 3: ein Beispiel für das Flat-Profile
Himanschu Arora, How to Use Gprof Profiling Tool on Linux (Tutorial)
https://linoxide.com/tools/gprof-performance-analysis-programs/ [19.12.2020/11:05]

- Das Aufrufdiagramm stellt dar, welche Funktionen von anderen aufgerufen wurden und wie viel Zeit jede Funktion gebraucht hat

```
                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 1.09% of 0.92 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    0.92                 main [1]
                0.06    0.85       1/1           test [2]
                0.01    0.00       1/1           some_other_test [5]
-----------------------------------------------
                0.06    0.85       1/1           main [1]
[2]     98.9    0.06    0.85       1         test [2]
                0.00    0.85       1/1           another_test [3]
-----------------------------------------------
                0.00    0.85       1/1           test [2]
[3]     92.3    0.00    0.85       1         another_test [3]
                0.85    0.00       1/1           yet_another_test [4]
-----------------------------------------------
                0.85    0.00       1/1           another_test [3]
[4]     92.3    0.85    0.00       1         yet_another_test [4]
-----------------------------------------------
                0.01    0.00       1/1           main [1]
[5]      1.1    0.01    0.00       1         some_other_test [5]
-----------------------------------------------
```

Abbildung 4: ein Beispiel für das Aufrufdiagramm
Himanschu Arora, How to Use Gprof Profiling Tool on Linux (Tutorial)
https://linoxide.com/tools/gprof-performance-analysis-programs/ [19.12.2020/11:05]

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Speichernutzung Profiling

- Der Code bestimmt, wann und wie erstellten Objekte zugewiesen und zerstört werden
- Wenn das richtig gemacht wird, ist die Speichernutzung effizient, im anderen Falle verwendet die Applikation mehr Speicher
- Der Speicher-Profiler identifiziert Objekte, die zur mehr Speichernutzung führt
- CPU- und GPU-Auslastung profilen

CPU

- Verschiedene Tools, um die CPU zu profilen

Perf

- Ein Profiler-Tool für Linux 2.6+
- Basiert auf dem Perf-events System, das auf Sampling basiert
- Verwendet CPU-Leistungsindikatoren, um die Applikation zu profilen
- In C geschrieben
- Der Befehl „perf record“ sammelt Samples und erzeugt eine Ausgabedatei „perf.data“

- „perf report“ und „perf annotate“ analysieren die erzeugte Datei
- Mit –F kann die Sampling-Frequenz angegeben werden

Gprof

- Kann sowohl die Laufzeit als auch die CPU-Auslastung profilen

Google Performance Tools

- Gperftools gekürzt
- Eine Sammlung von Tools, um Multithread Applikationen zu analysieren
- Nicht nur CPU-Profiler, sondern auch schnelle Speicherallokation malloc, Heap-Checker und Heap-Profiler

- Auf x86 Linux System entwickelt
- Nur auf Linux funktioniert

GPU

CodeXL

- Enthält ein GPU-Debugger, GPU- und CPU-Profiler
- Nutzt Sampling
- Funktioniert unter Linux und Windows
- Eignet sich gut für C und C++

- Der GPU-Profiler ist ein Tool zur Leistungsanalyse und sammelt während der Ausführung Daten aus der API-Laufzeit und der CPU für OpenCL Applikationen, die verwendet werden können, um die Kernelausführung zu optimieren
- Misst die Ausführungszeit eines OpenCL-Kernels
- Sammelt und visualisiert Daten Hardware-Leistungsindikatoren, Anwendungsabläufe und Kernel-Belegung
- AMDTActivityLogger misst die Ausführung von Segmenten im Programm und vergleicht mehrere Sessions denselben oder verschiedener Programme
- Es ist möglich, die Daten in einer Textdatei zu speichern

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Nebenläufigkeit Profiling

- Analysiert nebenläufige Prozesse (Threads)
- Analysiert Synchronisierungsprobleme, die später auftreten können
- Hilft dem Programmierer, das Laufzeitverhalten von nebenläufigen Prozessen besser zu verstehen und Fehler, wie Verklemmung (Dead Lock) zu entdecken

Arm MAP

- Funktioniert auf Linux
- Eignet sich gut für C, C++ und Python
- Wegen der Unterstützung der Multiprozesse kann für parallel Message Passing Interface (MPI) und OpenMP Applikationen verwendet werden

- Nicht nur für die Nebenläufigkeit sondern auch für die Speichernutzung, den Energieverbrauch, die E/A und die Kommunikation

Vtune Profiler

- Von Intel entwickelt
- Für Linux und Windows
- Das Profiling durch Sampling und Instrumentierung
- Für C, C++, C#, Java, Python, Go, die Nebenläufigkeit unterstützt, und Assembler

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Ein- und Ausgabe Profiling

- Spielt eine große Rolle für die Applikationsperformanz insbesondere für die Hochleistungsrechnen-Systeme und –Performanz

- Gründe, warum das System die E/A-Anforderungnen langsamer ausführt:
    1. nicht sequentielle Ein- und Ausgabe
    2. langsame Metadaten-Operationen, wie unlink, create und rename
    3. das Überprüfen, ob die Daten auf das Medium geschrieben sind
    4. viele Systemaufrufe

- Zwei Mögliche, um die E/A-Pattern zu bestimmen
    1. die Applikation auszuführen und den Speicher währen der Ausführung zu beobachten

2. ein E/A-Profiler zu nutzen

- Kein E/A-Profiler, das Tabellen wie Gprof erstellt

Top Linux Command oder Vmstat Command

- Linux bietet Tools, wie Top Linux Command oder Vmstat Command

- Sie geben Informationen über Betriebssystemspeicher, Interrupts und E/A Blocke
- Beispiel:

  Cpu(s): 14.4%us, 9.2%sy, 0.0%ni, 0.0%id, 75.5%wa, 0.3%hi, 0.7%si, 0.0%st

Iotop

- Braucht ein aktuelles Kernel (>= 2.6.20)

- Berechnet die Zeit, die die Prozesse auf die E/A-Operationen warten

- Zeigt, wie viele Daten gelesen und geschrieben werden
- Beispiel:

```
5359 be/4 daper    413.09 K/s    0.00 B/s  0.00 % 93.71 % find /
```

Iostat

- Für einen Administrator nützlicher als für einen Programmierer
- Für mehrere Geräte gleichzeitig nutzbar

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6,80    0,00    8,00   85,20    0,00    0,00

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0,00   125,20  103,80    5,00   415,20   520,80    17,21     1,57   14,38   8,59  93,44
```

Abbildung 5: ein Beispiel für Iostat
daper, Profiling Input/Output performance
https://www.linuxprogrammingblog.com/io-profiling [19.12.2020/12:41]

Vtune Profiler

- Analysiert die Ein- und Ausgabe
- Ermittelt die Performanz-Schwächen in E/A-intensiven Applikationen sowohl auf Hardware- als auch Softwareebene
- Ermittelt den Speicherbandbreitenverbrauch, den E/A-Bandbreitenverbrauch und den Memory Mapped E/A-Verkehr

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Energieverbrauch Profiling

- Die Entwickler stehen vor einer neuen Herausforderung
- Die Programmierung einer Applikation muss nicht nur bezüglich der Performanz und der Speichernutzung, sondern auch bezüglich des Energieverbrauchs optimiert werden
- Der Energieverbrauch-Profiler entdeckt, wo es Energieverschwendung in der Applikation gibt
- Zeigt auch, wie viel Energie die CPU verbraucht

Visual Studio

- Analysiert den Energieverbrauch von Universal Windows Plattform Apps (UWP) auf Tablets mit geringer Leistung
- Eignet sich gut für C#, C++, Visual Basic (VB) und F#
- F# ist eine Multi-Paradigmen-Sprachen, die auf funktionale Programmierung fokussiert

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Netzwerk Profiling

- Es ist wichtig, die Vorgänge im Netzwerk zu visualisieren, um zu verstehen und zu beheben, was zur Langsamkeit des Netzwerks führt

- Es gibt verschiedene Netzwerk-Profiler

ManageEngine NetFlow Analyzer

- Überwacht das Netzwerk

- Erkennt, wie die Bandbreiten des Netzwerks benutzt wird

- Analysiert den Netzwerk-Verkehr

- Für Linux und Windows

Iftop

- Ein Befehl Tool
- Überwacht das Netzwerk
- Funktioniert unter Linux
- Eignet sich für die Programmiersprache C
- Gibt einen schnellen Überblick über die Netzwerkaktivitäten

Nload

- Ein Befehl Tool für Linux
- Überwacht den Netzwerkverkehr und die Bandbreitennutzung
- Eignet sich gut für C++
- Mithilfe von Diagrammen können den ein- und ausgehenden Datenverkehr überwacht werden
- Gibt Informationen, wie z.B. die Gesamtmenge der übertragenen Daten und die minimale/maximale Netzwerknutzung

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Benchmarking vs Profiling

Benchmarking

- Benchmarking ist ein Test
- Misst , wie schnell der Code ist
- Überprüft, wie lange die Ausführung des Codes dauert und wo die Schwäche im Code sind
- Optimiert den Code
- Zuerst das Programm benchmarken, danach wird es in kleinere Programme zerlegt
- Jeder Teil wird optimiert

- Ein großes und schweres Problem wird in kleinere und einfache Teilprobleme zerlegt
- Zeigt, wie gut die Applikation oder ein Teil bei der Konfiguration funktioniert hat
- Bestimmt die Auswirkung der Änderungen im Code auf die Leistung der Applikation und vergleicht die Änderungen
- Wird verwendet, wenn die Geschwindigkeit der Applikation getestet werden muss

Profiling

- Methode, die die Leistungsschwäche in der Applikation analysiert
- Zeigt die Laufzeit, die Speichernutzung und was in der Applikation geschieht
- Analysiert Probleme in der Applikation
- Verwendet werden, wenn die Teile der Applikation, die am meisten Zeit braucht, identifiziert werden müssen

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Strss Testing

- Überprüft die Stabilität und Zuverlässigkeit einer Applikation
- Das Ziel: die Robustheit und Fehlerbehandlungsfähigkeit der Applikation unter extremen Bedingungen zu bewerten, und sicherzustellen, dass die Applikation nicht abstürzt
- Zeigt, wie die Applikation unter extremen Bedingungen reagiert
- Zeigt, wie die Applikation nach dem Auftreten eines Fehlers verhält
- Wird verwendet, um die Obergrenze, an der die Applikation abstürzt, zu bestimmen

Gliederung

- Applikation Profiling
  - was ist Applikation Profiling?
  - warum benutzt man Applikation Profiling?
  - welche Komponente kann man profilen?
- Profiling-Methoden
  - Instrumentierung
  - Sampling
- Laufzeit des Programms Profiling
- Speichernutzung Profiling
- Nebenläufigkeit Profiling
- Ein- und Ausgabe Profiling
- Energieverbrauch Profiling
- Netzwerk Profiling
- Benchmarking vs Profiling
- Stress Testing
- Zusammenfassung
- Literaturverzeichnis

Zusammenfassung

- Applikation Profiling misst die Performanz einer Applikation
- Mithilfe des Profilings kann eine Applikation verbessert und optimiert werden
- Es gibt zwei Profiling-Methoden, Instrumentierung (Tracing) und Sampling
- Bei Instrumentierung wird ein Code hinzugefügt
- Der Overhead kann bei Instrumentierung zu groß werden
- Bei Sampling wird kein Code hinzugefügt
- Der Overhead ist vernachlässig

- Statisches Verhalten
- Die Anzahl der Methodenaufrufe kann nicht angegeben werden
- Man kann die Laufzeit, die Speichernutzung, die Nebenläufigkeit, die Ein- und Ausgabe, den Energieverbrauch und das Netzwerk profilen
- Benchmarking ist ein Test und wird verwendet, um die Geschwindigkeit der Applikation zu bestimmen
- Stress Testing bewertet die Robustheit und Zuverlässigkeit der Applikation und zeigt wie die Applikation unter extremen Bedingungen reagiert
- Stress Testing zeigt, wie die Applikation verhält, nachdem ein Fehler auftritt

Literaturverzeichnis

- Chris Farrell (2010), The Fast Guide to Application Profiling, https://www.red-gate.com/simple-talk/dotnet/net-performance/the-fast-guide-to-application-profiling/#:~:text=Using%20profiling%20tools%20to%20look,without%20adding%20too%20much%20overhead [19.12.2020/15:51]
- Phil Tooley (2020), Application Performance Profiling: Part 2 – Choosing tools for Effective Profiling https://www.nag.com/blog/application-performance-profiling-part-2-choosing-tools-effective-profiling [19.12.2020/15:55]
- Euccas (2017), CPU Profiling Tools on Linux http://euccas.github.io/blog/20170827/cpu-profiling-tools-on-linux.html [19.12.2020/15:57]
- Gernot Klingler (2015), gprof, Valgrind and gperftools – an evaluation of some tools for application level CPU Profiling on Linux https://gernotklingler.com/blog/gprof-valgrind-gperftools-evaluation-tools-application-level-cpu-profiling-linux/ [19.12.2020/16:00]
- Manuel Meyer (2015), Analyse von .NET-Anwendungen mit Visual-Studio-Bordmitteln https://entwickler.de/online/was-laeuft-hier-eigentlich-160240.html [19.12.2020/16:09]
- Aaron Kili (2020), 16 Useful Bandwidth Monitoring Tools to Analyze Network Usage in Linux https://www.tecmint.com/linux-network-bandwidth-monitoring-tools/ [19.12.2020/16:15]
- CodeXL Documentation, GPU Profiler https://documentation.help/CodeXL/gpu-profiler.htm [19.12.2020/16:20]
- Daper (2011), Profiling Input/Output performance https://www.linuxprogrammingblog.com/io-profiling [19.12.2020/16:23]

- Linux New Media USA (2020), HPC Storage – Getting Started with I/O Profiling https://www.admin-magazine.com/HPC/Articles/HPC-Storage-I-O-Profiling [19.12.2020/16:30]
- Guru99, what is STRESS Testing in Software Testing? Tools, Types, Examples https://www.guru99.com/stress-testing-tutorial.html#1 [19.12.2020/16:45]
- CodePlanet (2010), C/C++ Profiler http://www.codeplanet.eu/tutorials/cpp/68-c-cpp-profiler.html [19.12.2020/16:57]
- Wikipedia, Arm MAP https://en.wikipedia.org/wiki/Arm_MAP [19.12.2020/17:15]
- Intel, Intel VTune Profiler https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/vtune-profiler.html [19.12.2020/17:33]
- Michael Kruckenberg, Jay Pipes (2005), Pro MySQL https://link.springer.com/chapter/10.1007/978-1-4302-0048-2_6 [19.12.2020/17:55]
- Tutorialspoint, Benchmarking and Profiling https://www.tutorialspoint.com/concurrency_in_python/concurrency_in_python_benchmarking_and_profiling.htm [19.12.2020/18:07]
- GitHub, Difference between Benchmarking and Profiling https://gist.github.com/vinhnglx/16283871d5872da491927ac9bab8bc39 [19.12.2020/18:30]
- Rbspy docs, Benchmarking your code https://rbspy.github.io/benchmarking-your-code/ [19.12.2020/ 18:44]