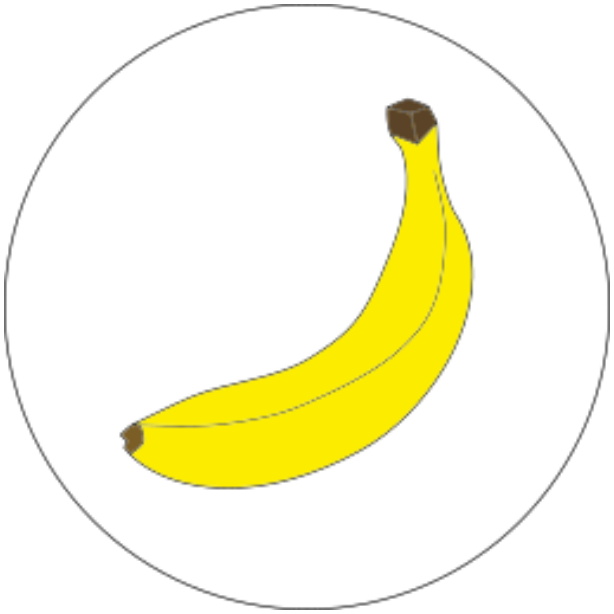


Balanced Banana

A Distributed Task Scheduling System

Qualitätssicherung

Niklas Lorenz, Thomas Häuselmann, Rakan Zeid Al Masri, Christopher Lukas Homberger und Jonas Seiler

29. März 2020

Inhaltsverzeichnis

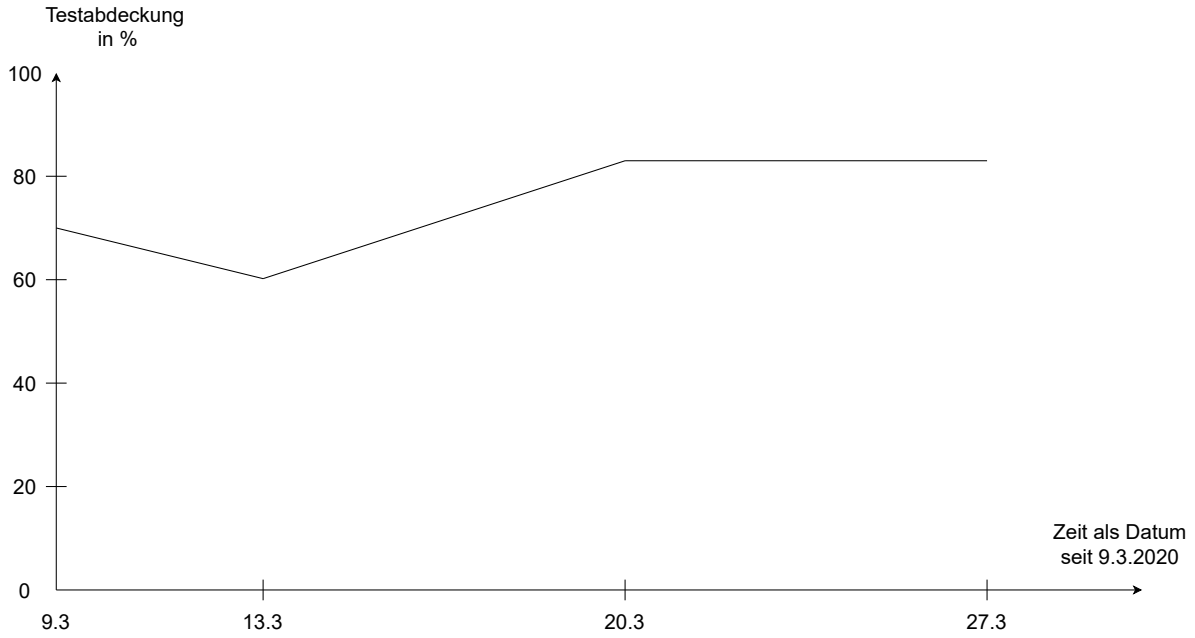
1 Einleitung

Dieses Dokument dient als Übersicht der Qualitätssicherung. Erläutert werden sowohl unsere Methoden zur Qualitätssicherung als auch das Ergebnis dieser. Ebenfalls beinhaltet dieses Dokument eine Anleitung zum Aufsetzen und Benutzen des endgültigen Programms.

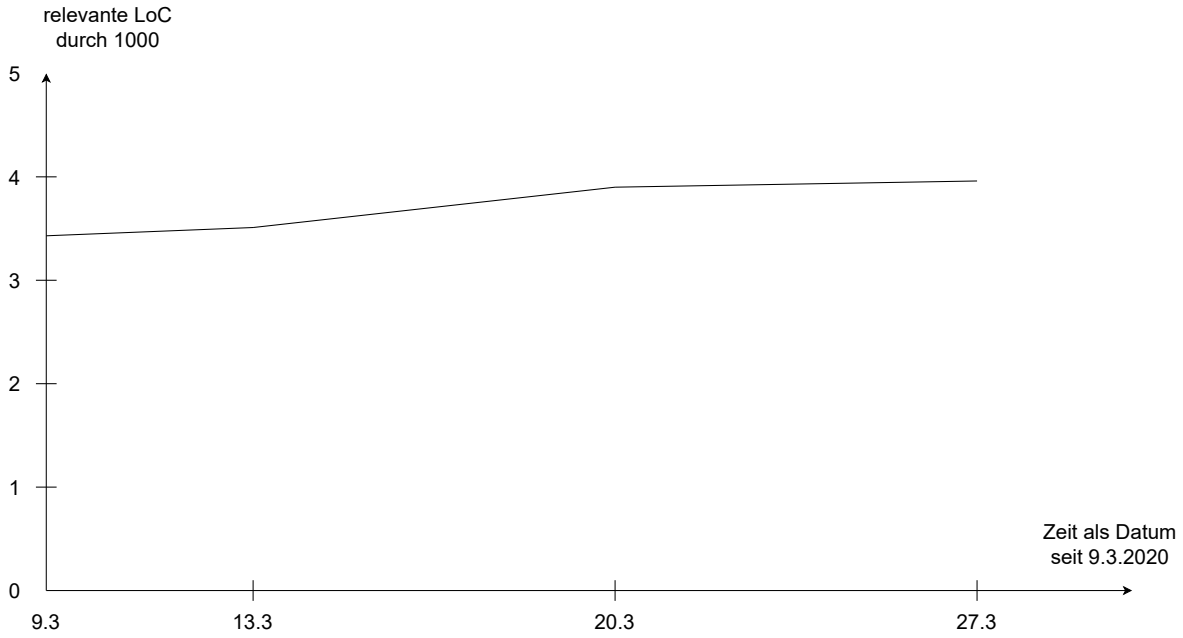
2 Testabdeckung

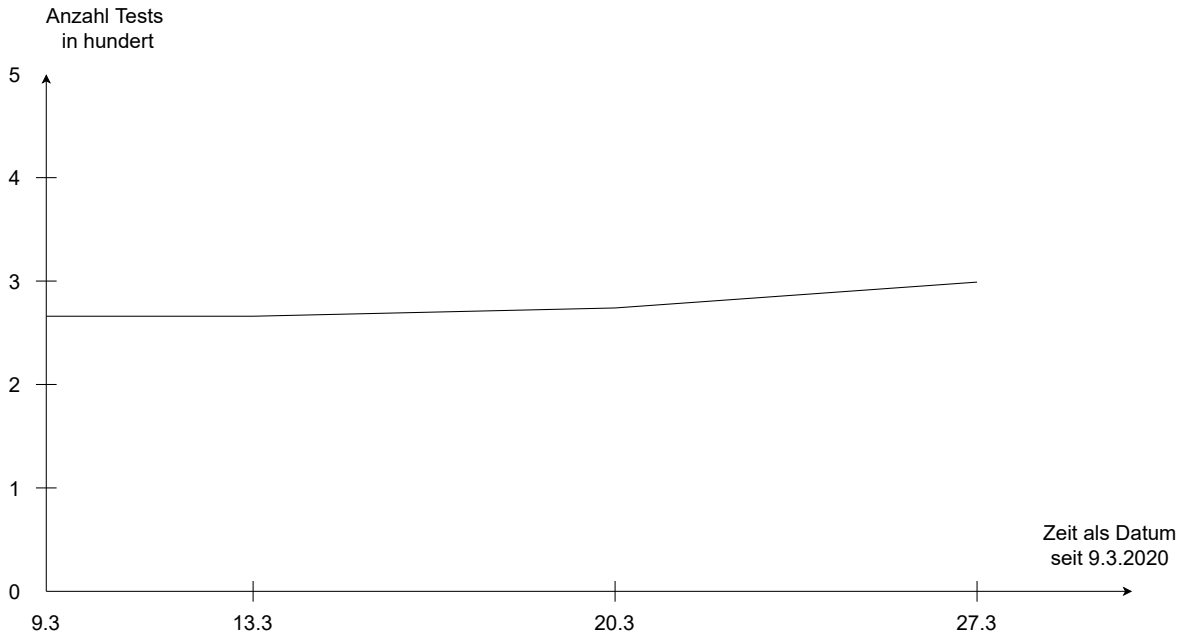
2.1 Klassenüberdeckung

Die Testüberdeckung der einzelnen Klassen ist in der ersten Woche der Qualitätssicherung gefallen, dies liegt daran, dass einige Klassen zwar intern funktioneren, sie jedoch im Zusammenspiel mit anderen nicht. In der ersten Woche wurden also hauptsächlich fehlerhafte Schnittstellen oder andere Fehler beseitigt und entsprechende Unit Tests erst in den folgenden Wochen eingeführt.



Entsprechend sehen wir einen Anstieg in Lines of Code (LOC) sowie der Anzahl Tests.





Da der Großteil des Codes bereits implementiert ist, sowie Integrationstests eher länger dafür aber weniger als Unittests sind, steigt auch die Anzahl der Tests nicht stark.

2.2 Integrationstests

Für das Automatisierte Testen des ganzen Systems haben wir ein Skript in Python benutzt, dieses beinhaltete diverse Szenarien und prüfte mit Positiv- als auch mit Negativtests. Neben diesem haben wir auch einiges "von Hand"getestet. Die, im Pflichtenheft definierten, Testfälle werden wie folgt abgedeckt:

T	Kurzbeschreibung	Wie getestet?
T1	Verbinden des Clients mit Server	Automatisch
T2	Festlegen von Prioritäten	Manuell
T3	Festlegen des Betriebssystems	Manuell
T4	Abfragen eines Aufgabenstatus	Automatisch
T5	Benachrichtigung bei Abschluss einer Aufgaben	Manuell
T6	Erstellen von Sicherungen	Manuell
T7	Anforderung von Ausgabe	Manuell
T8	Abbrechen von zu langen Aufgaben	-
T9	Manuelles Stoppen von Aufgaben	Manuell
T10	Manuelle Sicherung von Aufgaben	Manuell

In den Integrationstests haben wir hauptsächlich Verhalten getestet die so im Pflichtenheft noch nicht voraussagbar waren, daher haben wir tatsächlich mehr Tests als nur durch die vordefinierten ersichtlich.

3 Umgesetzte Funktionale Anforderungen

FA	Kurzbeschreibung	Implementiert?	Überdeckt von?
FA1	Client verbindet sich mit Server	Ja	T1
FA2	Benutzer authentifiziert sich	Ja	Integrationstest
FA3	Benutzer kann Aufgaben einreihen	Ja	T2 & T3
FA4	Benutzer kann Parameter übergeben	Ja	T2 & T3
FA41	Client speichert Parameter in Config	Ja	Manuelltest
FA42	Benutzer kann Config übergeben	Ja	Manuelltest
FA43	Benutzer kann Priorität festlegen	Ja	T2
FA44	Benutzer kann min und max Cores festlegen	Ja	Manuelltest
FA45	Benutzer kann min und max RAM festlegen	Ja	Manuelltest
FA46	Benutzer kann Betriebssystem festlegen	Ja	T3
FA47	Benutzer kann angeben ob Client blockiert	Ja	Manuelltest
FA48	Benutzer übergibt Pfad zu Aufgabe	Ja	Integrationstest
FA49	Es können Standardwerte benutzt werden	Ja	Integrationstest
FA5	Benutzer kann Status von Aufgabe einsehen	Ja	T4
FA6	Benutzer bekommt Email bei Abschluss	Ja	T5
FA7	Server erstellt Backups	(Nein)	T6
FA8	Benutzer kann Ausgabe anfordern	Ja	T7
FA9	Benutzer kann Statistiken abfragen	Ja	Manuell
OFA1	Benutzer kann Restzeit einsehen	Nein	-
OFA2	Server stoppt lange Aufgaben	Nein	T8
OFA3	Benutzer kann manuell Aufgaben stoppen	Nein	T9
OFA4	Benutzer kann manuell Aufgaben sichern	(Nein)	T10
OFA5	Benutzer kann Pausierbarkeit angeben	(Nein)	-

Für Anforderungen mit einem (Nein) existiert zwar in manchen Teilen des Programms die Funktionalität dafür, die Funktion selber ist jedoch nicht umgesetzt oder nutzbar.

4 Benutzeranleitung

4.1 Aufsetzen des Schedulers

Zuerst müssen wir das System aufsetzen, das Programm wird in einem Paket geliefert, die entsprechende Funktionen finden sich dann in den einzelnen ausführbaren Dateien.

- Herunterladen des Programms unter
  `https://github.com/balancedbanana/balancedbanana/releases`
- Einrichten der MySql Datenbank mit

  ```
  cat balancedbanana.sql | sudo mysql
  ```

  Die SQL Datei ist im Archiv unter
  `share/balancedbanana/balancedbanana.sql`

zu finden.
Nun ist die Datenbank unter dem Schema balancedbanana installiert. In der MySQL-Konsole einen Datenbank Nutzer für balancedbanana anlegen mit folgenden MySQL-Befehlen:

```
CREATE USER 'balancedbanana'@'localhost' IDENTIFIED BY
'balancedbanana';

GRANT ALL PRIVILEGES ON balancedbanana.* TO 'balancedbanana'@'localhost';

FLUSH PRIVILEGES;

exit
```

- Configfiles pro Benutzer anlegen unter
  `$HOME/.bbc` bzw `.bbs` oder `.bbd`.
  Dazu einfach die Configfiles von
  `share/balancedbanana/.bbc` bzw `.bbs` oder `.bbd`
  kopieren.
- Das Passwort `balancedbanana` nach `IDENTIFIED BY` bei Bedarf anpassen und in
  `$HOME/.bbs/appconfig.ini`
  in Zeile `databasepassword` anpassen.

Nun ist der Scheduler aufgesetzt.

4.2 Benutzen des Clients

Der Client kann mit dem Programm `bbc` gestartet werden. Der Client hat verschiedene Subcommands sowie Flags für diese, nachführend sind diese erläutert. Für nicht angegebene Parameter wird zuerst in der Benutzerconfig (`$HOME/.bbc`) dann in der Applikationsconfig (`share/balancedbanana/.bbc`) und danach wird entweder auf Standardwerte oder auf gesetzte Standardparameter im Scheduler gesetzt. Bei Unklarheiten der Kommandos oder deren Parameter kann immer mit dem Argument -h ein Hilfemenü aufgerufen werden.

4.2.1 Run

Der Run-Command reiht eine Aufgabe ein, folgende Parameter sind verfübar:

- –server. Gibt die IP Adresse des Schedulers an. Angaben als String.
- –port. Gibt den Port am Scheduler an. Angaben als 16 Bit Integer.
- –block / -b. Gibt an ob der Client blockiert, bis die Aufgabe beendet ist. Angaben als Boolean.
- –email / -e. Gibt die Email an, die benachrichtigt wird, wenn die Aufgabe beendet ist. Angaben als String.

- –image / -i. Gibt den Pfad des Images an, dass für die Aufgabe benutzt werden soll. Angaben als String.
- –priority / -p. Gibt die Priorität der Aufgabe an. Mögliche Werte sind "Low","Normal", "High".
- –max-cpu-count / -C. Gibt die maximale Anzahl, der benutzten Kerne, der Aufgabe, an. Werte als 32 Bit Integer.
- –min-cpu-count / -c. Gibt die minimale Anzahl, der benutzen Kerne, der Aufgabe, an. Werte als 32 Bit Integer.
- –max-ram / -R. Gibt den maximal nutzbaren Arbeitsspeicher in miB an. Werte als 64 Bit Integer.
- –min-ram / -r. Gibt den minimal nutzbaren Arbeitsspeicher in miB an. Werte als 64 Bit Integer.
- –job / -j. Gibt die Eingabe an, die den Job startet. Alles hinter dieser Eingabe wird als Kommando aufgefasst, dieses Argument sollte also das letzte des Run Kommandos sein. Angaben als String.

Beispiel:

```
./bbc run --server 192.168.178.82 --port 25565 -b true -e example@hello.com
-i catsimulation -p High -C 4 -c 1 -R 4096 -r 1024 -j sh catsimulationfile.sh
```

4.2.2 addImage

Der addImage-Command registriert ein Image, sodass es mit –image eingereiht werden kann. Das einzige Argument ist der Pfad zum Dockerfile. Beispiel:

```
./bbc addImage catsimulation ../../Simulations/Catsimulation
```

4.2.3 removeImage

Entfernt ein registriertes Image unter dem gegebenen Namen. Das einzige Argument ist der Name des Images. Beispiel:

```
./bbc removeImage parrotsimulation
```

4.2.4 status

Frägt den Scheduler nach dem Status eines Jobs. Das einzige Argument ist die ID des Jobs als unsigned Integer. Beispiel:

```
./bbc status 42
```

4.2.5 tail

Frägt den Scheduler nach den letzten ausgegebenen Zeilen der Aufgabe. Das einzige Argument ist die ID des Jobs als unsigned Integer. Beispiel:

```
./bbc tail 42
```

4.2.6 stop

Stopt den angegebenen Job. Das einzige Argument ist die ID des Jobs als unsigned Integer. Beispiel:

```
./bbc stop 43
```

4.2.7 pause

Pausiert den angegebenen Job. Das einzige Argument ist die ID des Jobs als unsigned Integer. Beispiel:

```
./bbc pause 43
```

4.2.8 continue

Setzt den angegebenen Job fort. Das einzige Argument ist die ID des Jobs als unsigned Integer. Beispiel:

```
./bbc continue 43
```

4.2.9 backup

Sichert den angegebenen Job. Das einzige Argument ist die ID des Jobs. Beispiel:

```
./bbc backup 43
```

4.2.10 restore

Stellt den angegeben Job aus einem Backup wieder her. Das einzige Argument ist die Backup ID. Beispiel:

```
./bbc restore 240
```

4.3 Benutzen des Schedulers

Der Scheduler hat im Gegensatz zum Client keine unterschiedlichen Kommandos und muss nur gestartet werden. Bei Unklarheiten kann wieder mit –help ein Hilfemenü aufgerufen werden. Folgende Parameter stehen zur Verfügung:

- –server / -s. Legt fest auf welcher IP-Adresse der Scheduler gestartet wird. Angaben als String.
- –webapi / -w. Legt fest auf welcher IP-Adresse die WebAPI des Schedulers gestartet wird. Angaben als String.
- –serverport / -S. Legt fest auf welchem Port der Scheduler gestartet wird. Angaben als 32 Bit Integer.
- –webapi-port / -W. Legt fest auf welchem Port die WebAPI gestartet werden soll. Angaben als 32 Bit Integer

Hier ein Beispielaufruf:

```
./bbs -s 192.168.178.81 -w 192.168.178.82 -S 25565 -w 25566
```

4.4 Benutzen des Workers

Der Worker hat ebenfalls nur ein Kommando womit er gestartet wird. Bei Unklarheiten kann ebenfalls wieder mit –help ein Hilfemenü aufgerufen werden. Folgende Parameter stehen zur Verfügung:

- –server / -s. Legt fest welche IP-Adresse der Scheduler hat. Angaben als String.
- –serverport / -S. Legt fest auf welchem Port der Scheduler gestartet wurde. Angaben als 32 Bit Integer.

Hier ein Beispielaufruf:

```
./bbd -s 192.168.178.82 -S 25565
```

5 Probleme und Bugs

6 Konklusion

Wie lief es, Leistung, etc.