Systemarchitektur eines Sensor/Aktor Knotens für dezentralisierbare Aufgabenverteilung

Bachelorarbeit

Inhaltlicher Überblick

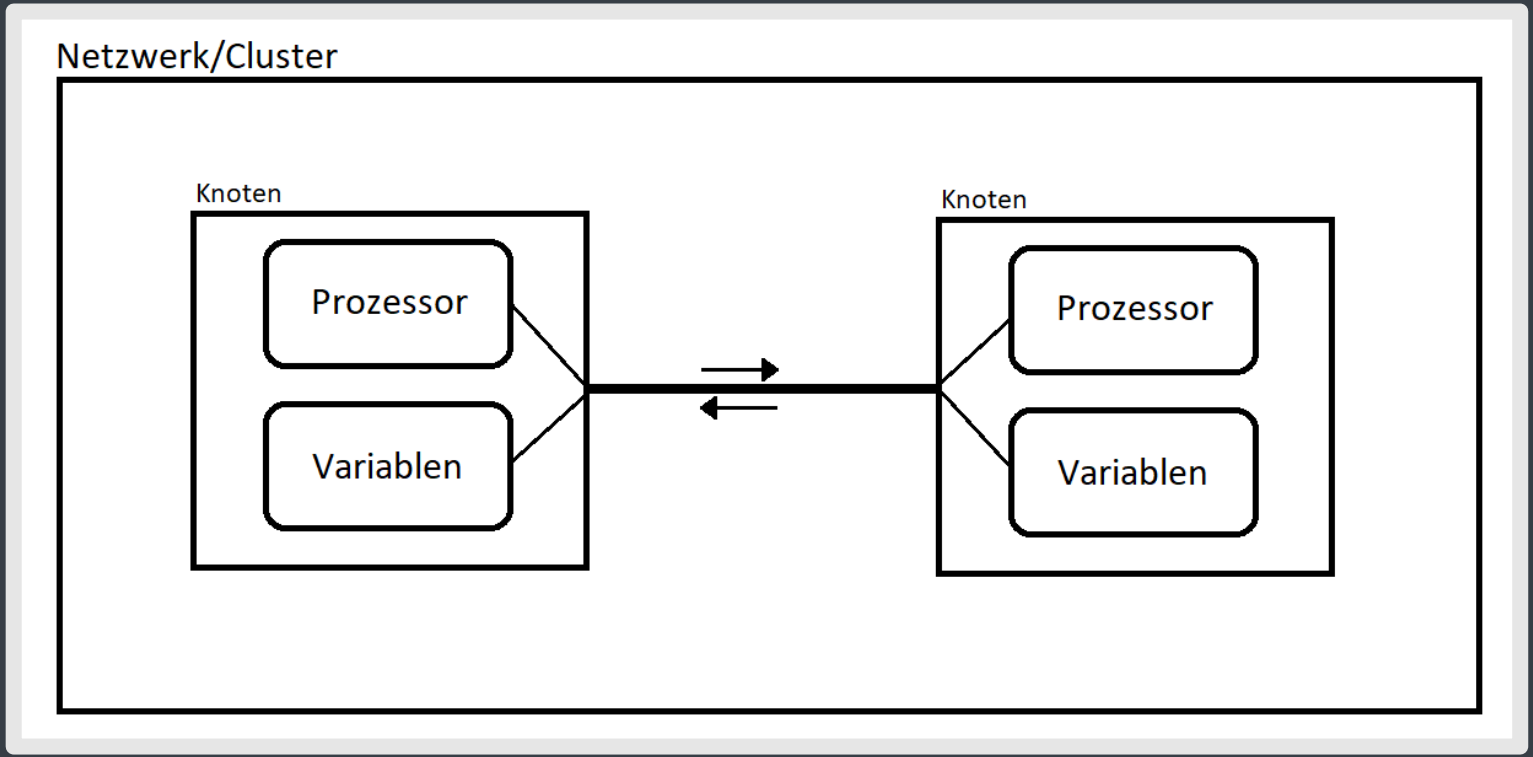
- Idee
- Modellierung
- Umsetzung
- Ergebnis
- Ausblick

Idee

Idee

Motivation und konzeptionelle Grundlage

- Cluster Computing
  - Parallelisierung von Aufgaben
  - Algorithmuslaufzeit verkürzen
  - CPU Last gleichmäßig verteilen
- Load Balancing auf einem Sensor/Aktor Knoten
- Gleichberechtigte Architektur

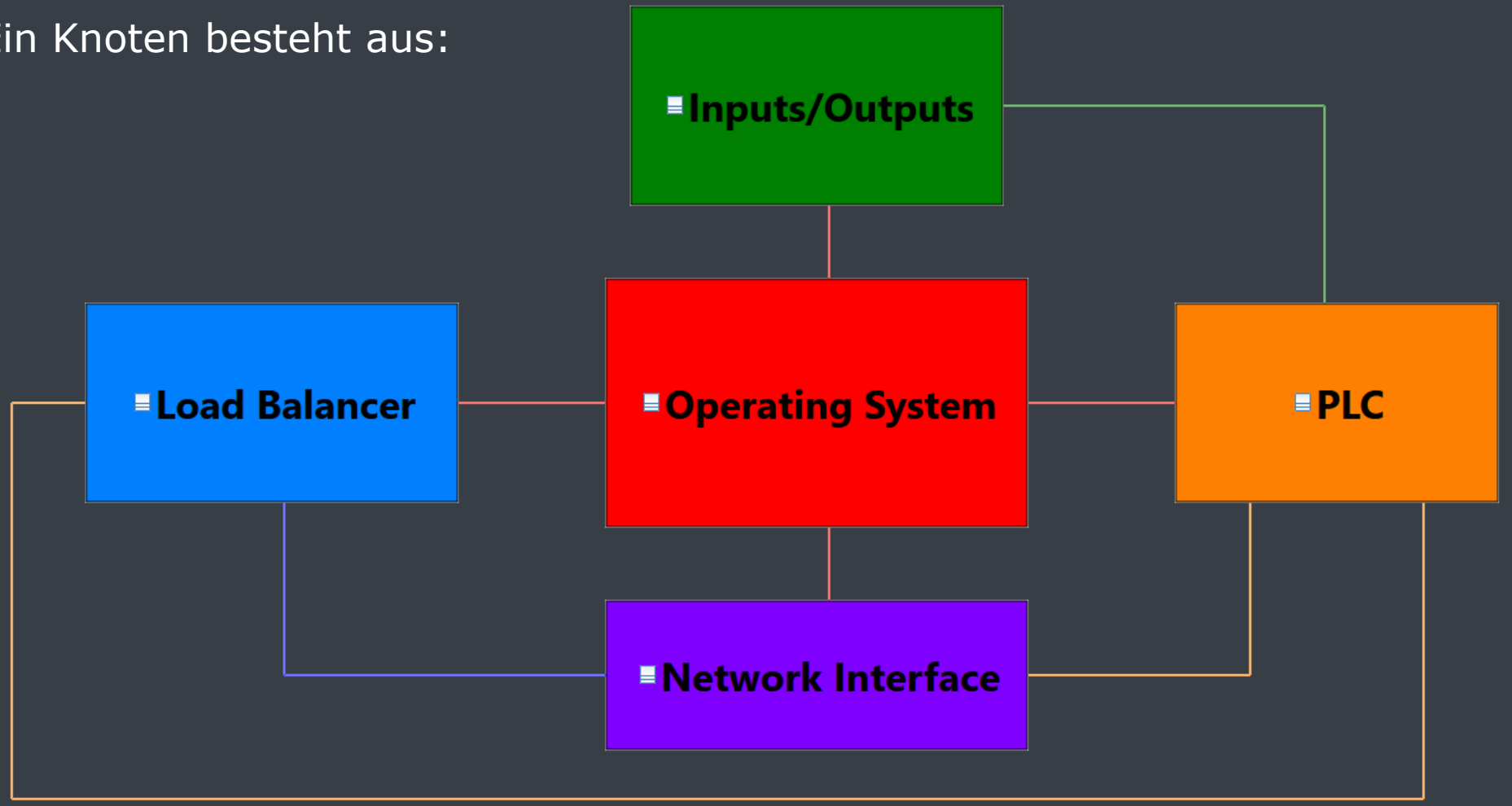


Modellierung

Modellierung

Class Diagram

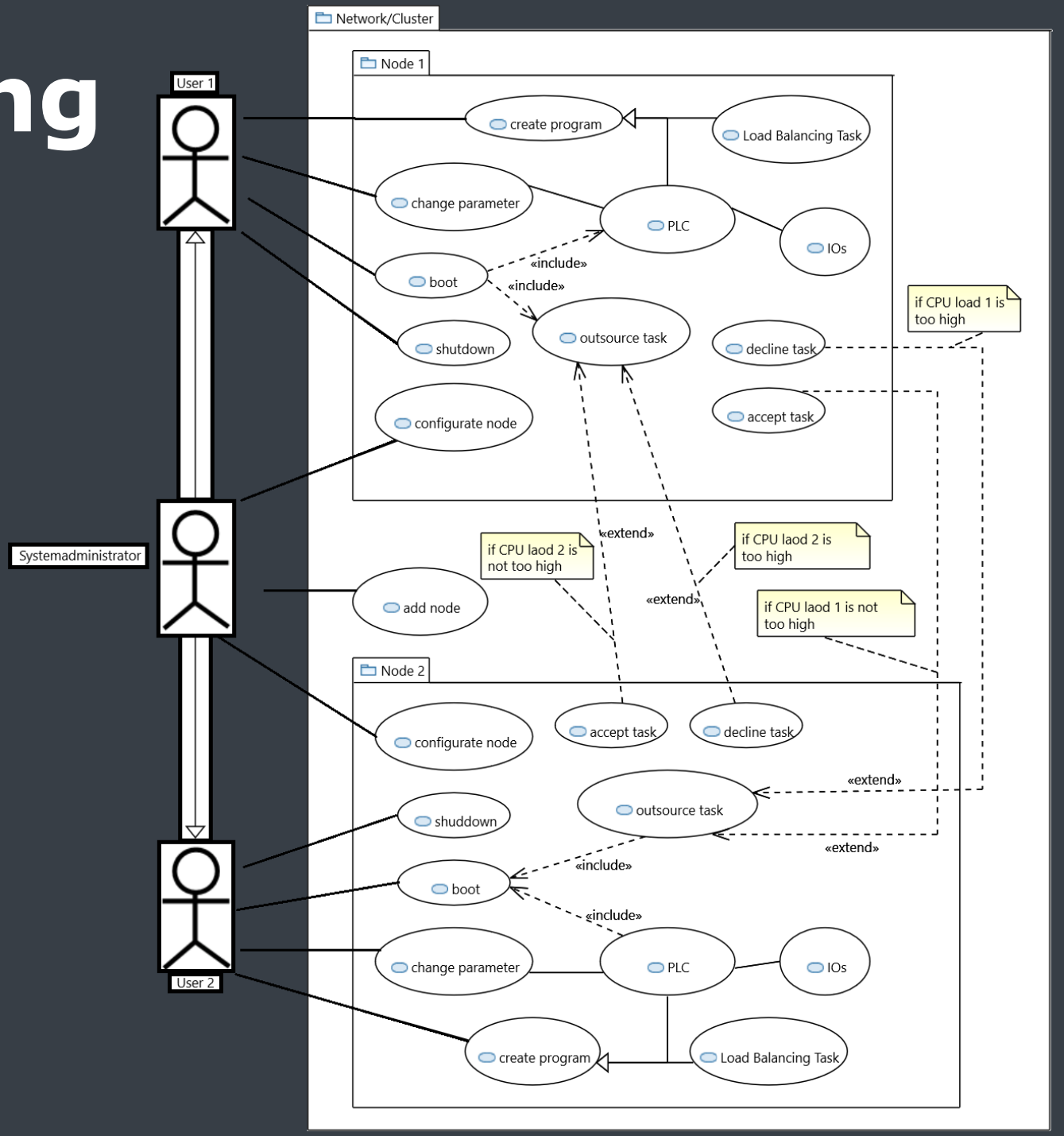
- Ein Knoten besteht aus:



Modellierung

Use Case Diagram

- Netzwerk aus 2 Knoten:

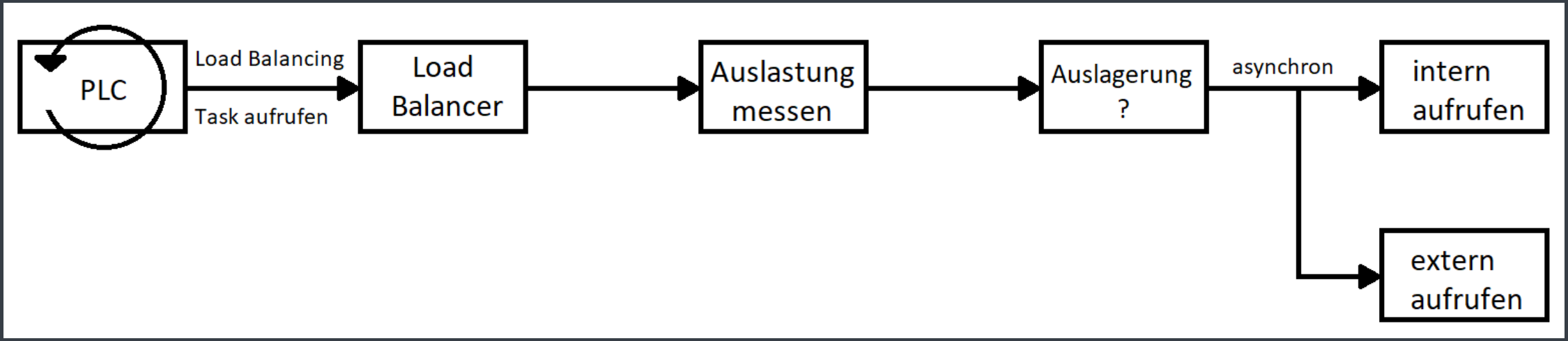


Umsetzung

Umsetzung

Ablauf

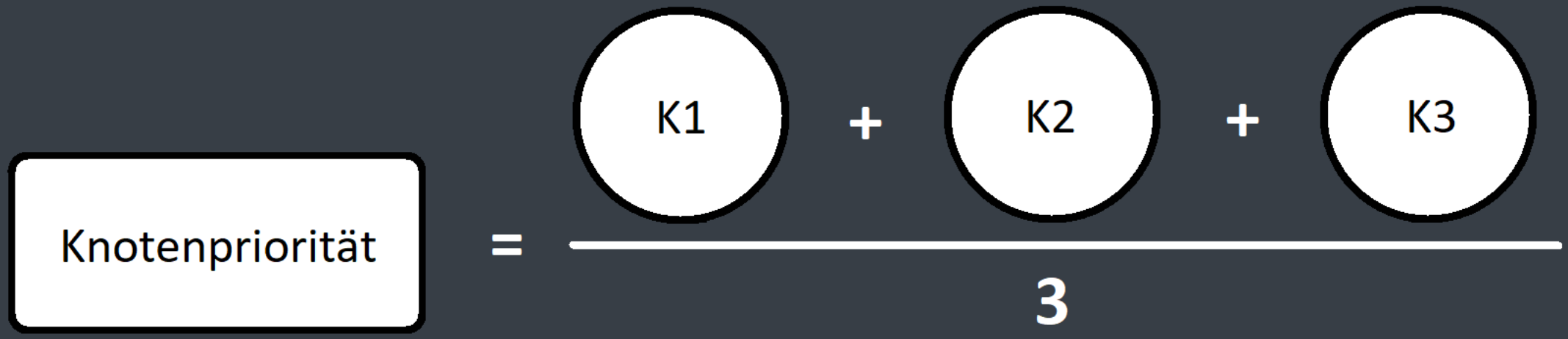
- PLC ruft bestimmte Load Balancing Tasks auf
- Load Balancing Tasks laufen asynchron zum Zyklus
- Load Balancing Tasks auf anderen Knoten auslagern, Ergebnis kommt zurück



Umsetzung

Knotenpriorität

- K1: Interprozesskommunikation
- K2: Netzwerkkommunikation
- K3: CPU Intensität



Umsetzung

Load Balancing Tasks

Task Nummer	Name	K1	K2	K3	NP
0	databaseAccess	2	5	2	3
1	websiteCall	1	2	3	2
2	imageProcessing	5	9	1	5
3	averageCalc	6	7	5	6
4	sendEmail	4	5	2	3,666
5	userInput	3	4	6	4,333
6	blinky	8	9	4	7

Umsetzung

Auslastungsmessung

- Ausgabe von ps (process status)
- In Kombination mit Linux Priorität
- Prioritätsscheduling
- Einmaliger Aufruf führt zu Spitzen
- Daher Mittelwert über mehrere Aufrufe

```
[root@host-1-201 ~]# ps -eo pid,uid,priority,ni,pcpu,comm --sort -pcpu
  PID   UID PRI  NI %CPU COMMAND
 1432    42  20   0  3.6 gnome-shell
 1650     0  20   0  3.4 sshd
    1     0  20   0  2.0 systemd
 1658     0  20   0  1.0 bash
  591   999  20   0  0.8 polkitd
  602    81  20   0  0.5 dbus-daemon
 1406    42  20   0  0.5 gnome-settings-
    9     0  20   0  0.4 rcu_sched
  289     0   0 -20  0.3 kworker/0:1H
  660     0  20   0  0.3 NetworkManager
  989     0  20   0  0.3 tuned
 1038     0  20   0  0.3 Xorg
 1676     0  20   0  0.3 abrt-dbus
    4     0  20   0  0.2 kworker/0:0
  455     0  20   0  0.2 systemd-journal
  999     0  20   0  0.2 libvirtd
  481     0  20   0  0.1 systemd-udevd
  988     0  20   0  0.1 rsyslogd
 1499    42  20   0  0.1 goa-daemon
 1515     0  20   0  0.1 udisksd
    2     0  20   0  0.0 kthreadd
    3     0  20   0  0.0 ksoftirqd/0
    5     0   0 -20  0.0 kworker/0:0H
    6     0  20   0  0.0 kworker/u2:0
    7     0 -100   -  0.0 migration/0
    8     0  20   0  0.0 rcu_bh
   10     0 -100   -  0.0 watchdog/0
   12     0  20   0  0.0 kdevtmpfs
```

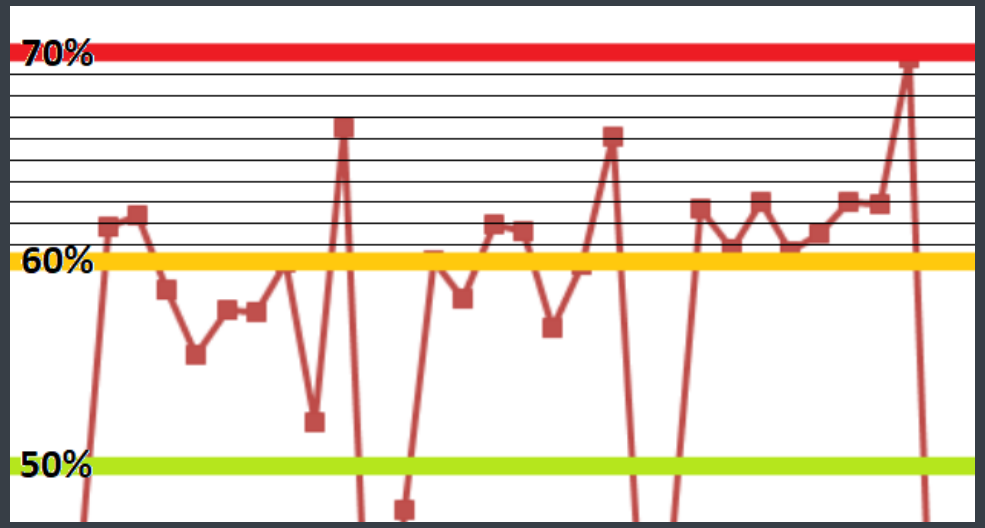
Umsetzung

Auslagerungsverfahren

- Auslagerungsverfahren mit konfigurierbaren Grenzen

- Schwellwerte:
  - 70% theoretisches Maximum
  - 60% Beginn des Auslagerungsverfahrens
  - 50% Hysterese Grenze

- Jedes ganzzahlige Prozent über 60%
  lagert einen Task mit einer Knotenpriorität + 1 aus



Ergebnis

Ergebnis

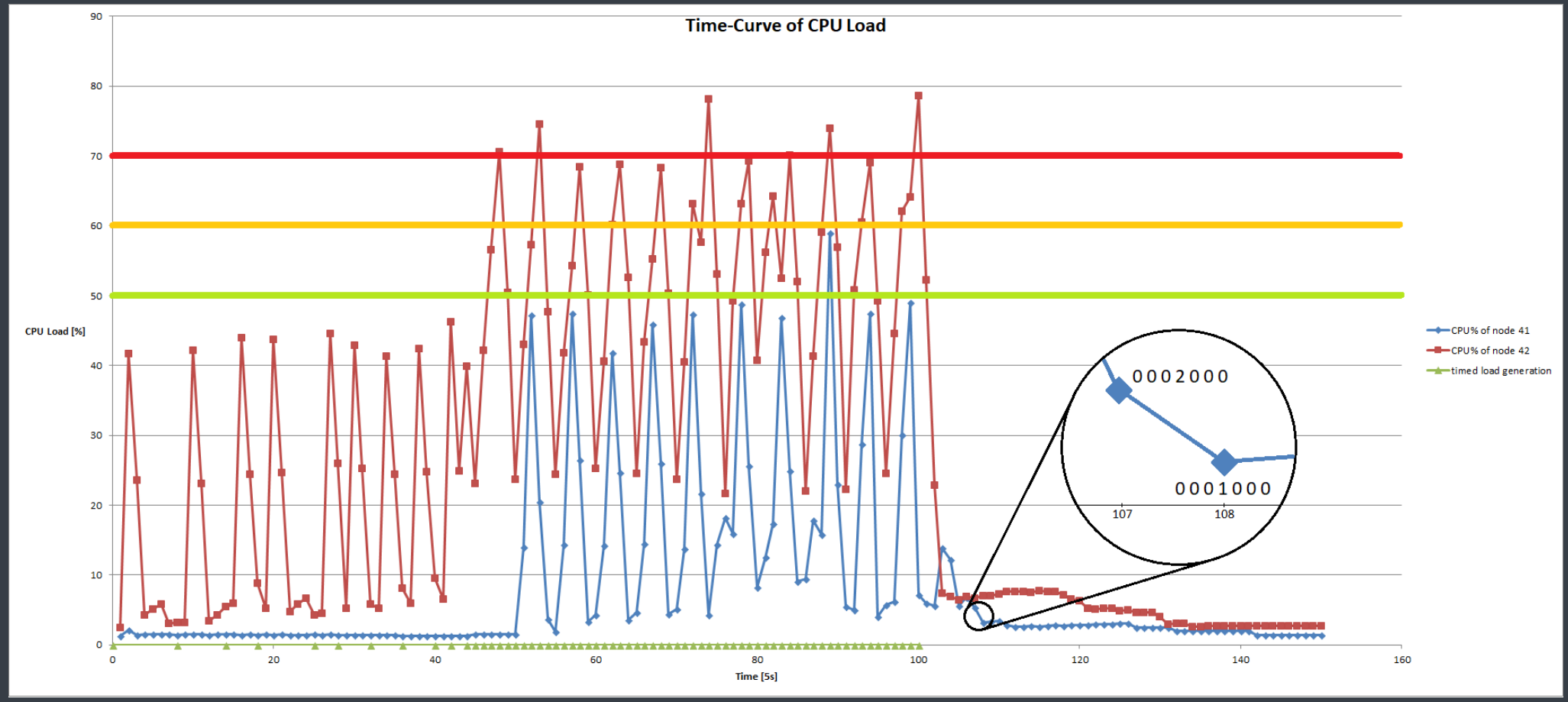
Resultat des „Proof Of Concept“

- Feste Laufzeit des zyklischen Programms der Load Balancer
- Zyklisches Programm erhöht kontinuierlich die Last
- Aufzeichnung der CPU Last
- Auslastungsmessung: Mittelwert über 5 und 10 Aufrufe

Ergebnis

Resultat des „Proof Of Concept“

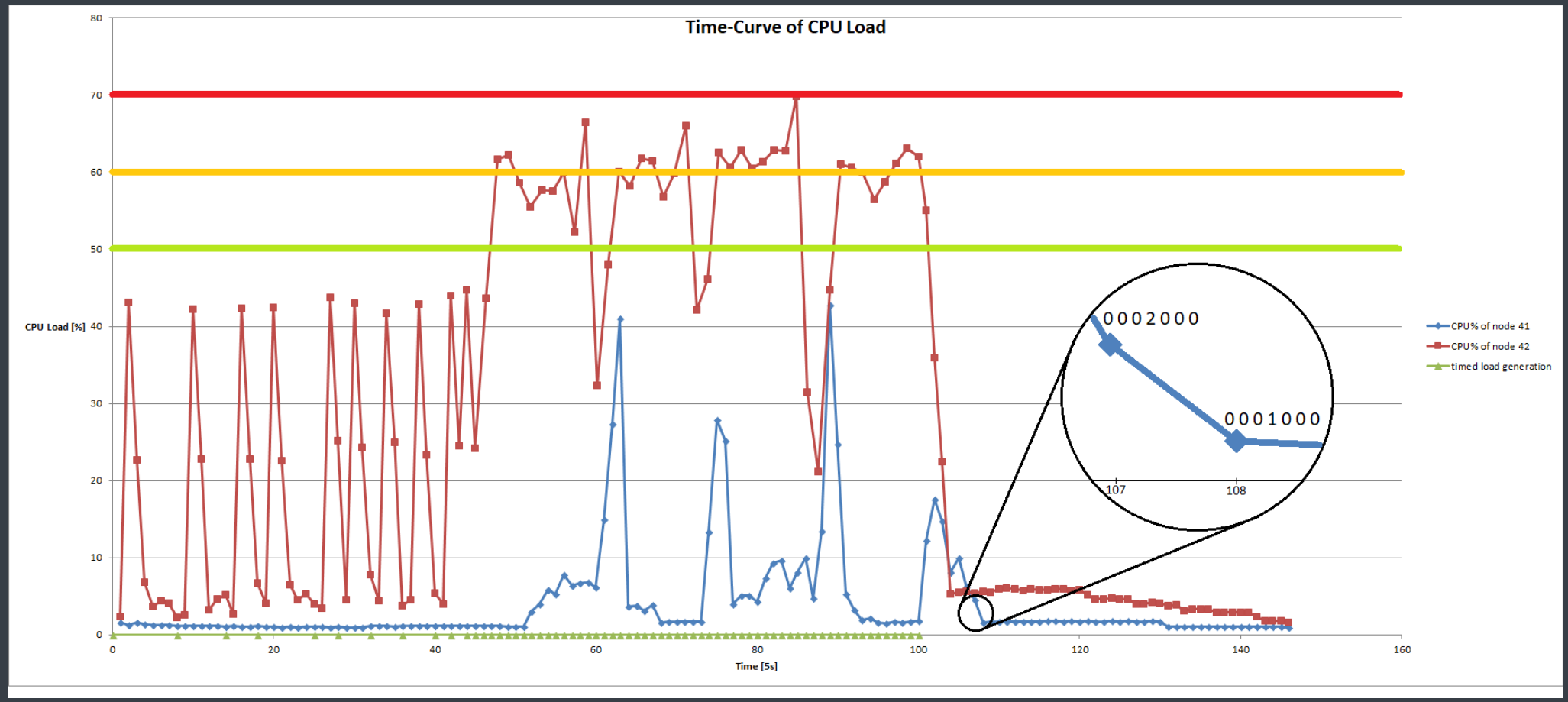
- Auslastungsmessung Mittelwert über 5 Aufrufe



Ergebnis

Resultat des „Proof Of Concept“

- Auslastungsmessung Mittelwert über 10 Aufrufe



Ausblick

Ausblick

Erweiterungsmöglichkeiten für zukünftige Projekte

- Tasks während der Laufzeit unterbrechen
- Dort fortsetzen wo die erste CPU beendet hat
- Knotenpriorität ermitteln – statisch, dynamisch, kombiniert