Leistungsmessung

- Ziele und Vorgehensweise
- Vor- und Nachteile der verschiedenen Verfahren
- Primitive Hilfsmittel zur Messung
- Offline-Werkzeuge
- Online-Werkzeuge
- Optimierung des Programm-Codes
- Messmethodik

Leistungsmessung
Die zehn wichtigsten Fragen

- Wie ist die Vorgehensweise bei der Programmoptimierung?
- Was sind die Vor- und Nachteile der Offline- und Online-Werkzeuge?
- Welches Verfahren eignet sich wozu?
- Welche primitiven Hilfsmittel gibt es?
- Was bietet Vampir?
- Was bietet Paradyn?
- Wie optimiert man ein Programm?
- Wieso ist die korrekte Messmethodik wichtig?
- Welche Fehlertypen gibt es?
- Was kann man gegen diese unternehmen?

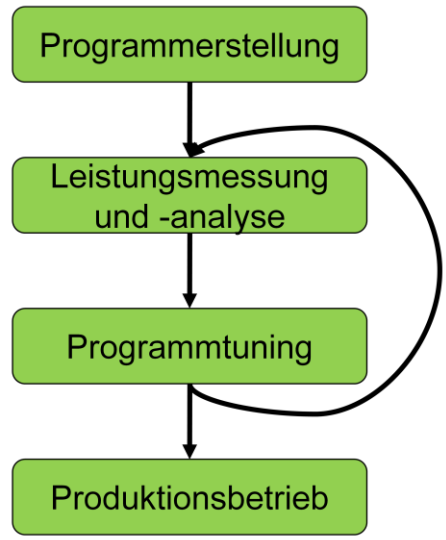
Ziele der Leistungsmessung

- Ausführungszeit minimieren
  - Programmergebnis so schnell wie möglich
- Durchsatz maximieren
  - So viele zufriedene Benutzer wie möglich
- Auslastung optimieren
  - Optimale Nutzung der investierten Gelder

Ziele der Leistungsmessung…

- Wie gut werden die Ziele erreicht?
- Zu diesem Zweck Leistungsmessung
  - Zunächst Messung der Daten
  - Bewertung der Leistungsdaten
  - Code-Änderungen oder Änderung der Nutzungsweise des Rechners

Vorgehensweise



Vorgehensweise…

- Hypothese über Grund des Leistungsmangels aufstellen
- Messungen durchführen
- Wenn Hypothese richtig
  - Leistungsengpass korrigieren
  - Evtl. vorher Hypothese verfeinern
- Wenn Hypothese falsch
  - Andere Hypothese wählen
- Konzeptionell wie bei der Fehlersuche
  - „Debugging for performance“

Messverfahren

- Wir kennen schon zwei Klassen
  - Offline-Verfahren (Spurbasiert)
    - Erfasst Daten zur Laufzeit
    - Auswertung nach Programmende
  - Online-Verfahren
    - Erfasst Daten zur Laufzeit
    - Auswertung zur Laufzeit
    - Änderung der Art der erfassten Daten

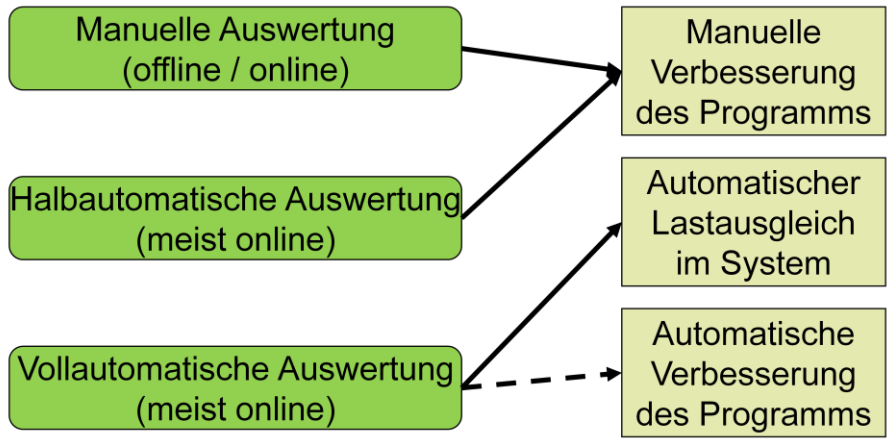
Offline-Messverfahren

- Vorteile
  - Viele Informationen dauerhaft erfasst
  - Später beliebig auswählbar und analysierbar
  - Vergleich verschiedener Programmläufe möglich
- Nachteile
  - Konstante Zusatzlast durch Spurgenerierung
  - Nicht situationsbedingt verfeinerbar
- Probleme
  - Hohe Datenmenge (Mega-/Gigabyte-Bereich)
  - Genau synchronisierte Uhren notwendig

Online-Messverfahren

- Vorteile
  - Sofortige Reaktion möglich
  - Last der Instrumentierung regelbar
  - Bedingte Messung (einzelne Abschnitte)
- Nachteile
  - (Meist) keine nachträgliche Auswertung
- Probleme
  - Instrumentierung des laufenden Programms
  - Datentransport zum zentralen Programmteil
  - Online-Verfahren vs. Batch-Betrieb im Cluster

Benutzungsmethoden



Messwerkzeuge

- Primitive Hilfsmittel
  - C-Funktionalität
- Offline-Werkzeuge
  - Vampir
  - Sunshot
- Online-Werkzeuge
  - Paradyn

Primitive Hilfsmittel

- Manchmal sind Werkzeuge zu aufwendig und/oder zu teuer
- Rückgriff auf das, was die Programmierbibliotheken bereits anbieten
- (Erinnerung: Lieblingsdebugger: `printf()` ☺)

Primitive Hilfsmittel…

- `clock_gettime()`-Funktion
  - Liefert CPU-Zeit mit Nanosekundenauflösung

```
#include <time.h>

struct timespec starttime, endtime;
time_t elapsed;

clock_gettime(CLOCK_REALTIME, &starttime);
/* Code */
clock_gettime(CLOCK_REALTIME, &endtime);

elapsed = endtime.tv_sec – starttime.tv_sec;
```

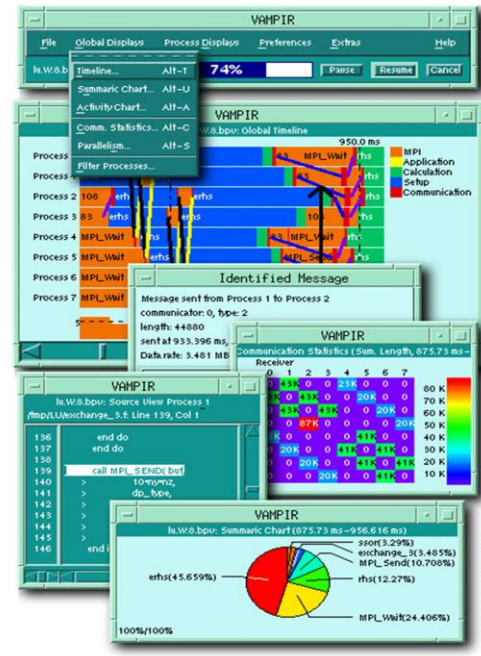

CLOCK_REALTIME gibt die Zeit seit 1970-01-01T00:00:00Z zurück. tv_sec enthält die Sekunden, tv_nsec die Nanosekunden.

Offline-Werkzeuge: Allgemein

- Betrachtung aus Sicht eines Informatikers
  - Erstellung der Spuren schwierig
    - Zusätzlicher Code an verschiedenen Stellen im System
    - Zusammenführung
    - Zeitsynchronisation
  - Auswertung der Spuren macht Spaß
    - Verschiedenste bunte Displays, die die enthaltenen Informationen unterschiedlich darstellen
  - Folge: Es gibt viele Offline-Werkzeuge ☺

Offline-Werkzeug Vampir

- Visualization and Analysis of MPI Programs
- Ursprünglich vom Forschungszentrum Jülich
- Jetzt hauptsächlich an der Technischen Universität Dresden
- Vampir bietet eine sehr leistungsfähige Offline-Inspektion des Programms
- Weitverbreitetes Werkzeug auf Cluster-Architekturen und manchen Parallelrechnern



Vampir: Allgemein

- Offline-Spuranalyse für Programme mit Nachrichtenaustausch
- Anpassbare Benutzungsschnittstelle
- Skalierbar bzgl. Zeit und Prozess-/Prozessor-Anzahl
- Aufwendige Filterung und Zoom
- Darstellung und Analyse von Ereignissen aus MPI und der Anwendung
  - Routinen des Anwenderprogramms
  - Punkt-zu-Punkt-Kommunikation
  - Kollektive Kommunikation
  - MPI-I/O-Operationen
- Alle Diagramme vielfach anpassbar
- Viele verschiedene Displays für alle Informationen

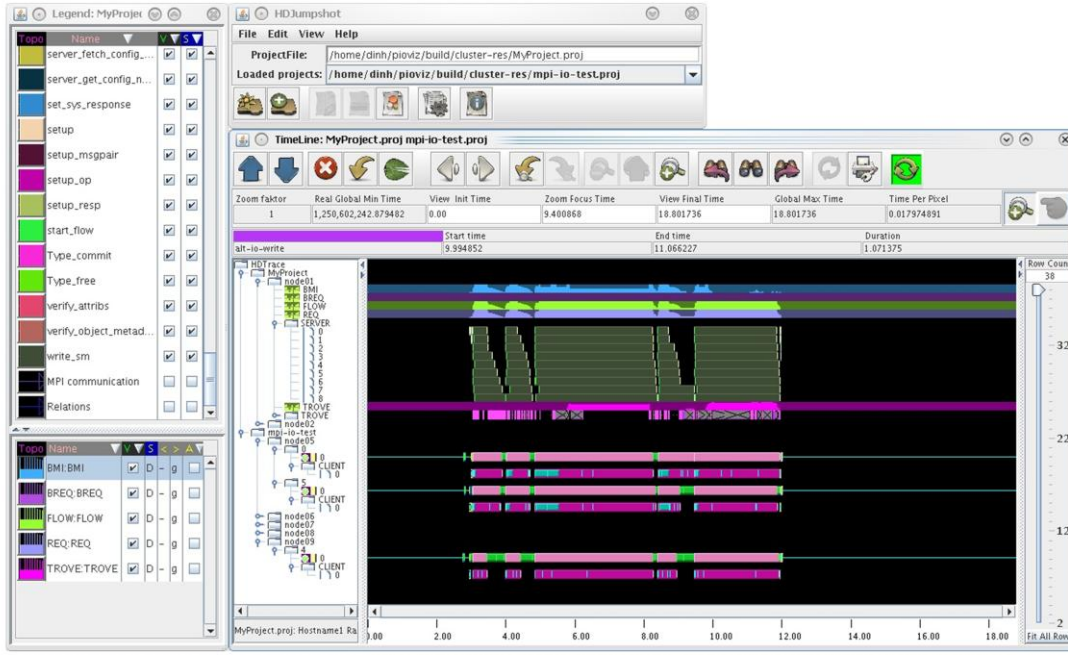
Offline-Werkzeug Sunshot

- Weiterentwicklung von Jumpshot
- GNU General Public License
  - Quellcode ist verfügbar
- Experimentelle Features
  - Benutzerdefinierte Datentypen anzeigen
  - Unterstützung für HDTrace
  - Anzeige der Topologie
- Anzeige von ...
  - Geschachtelten Ereignissen
  - Beziehungen zwischen Ereignissen
  - Profilen

Jumpshot ist Teil von MPE2 (MPI Parallel Environment), welches wiederum Teil von MPICH2 (http://www.mcs.anl.gov/research/projects/mpich2/) ist.

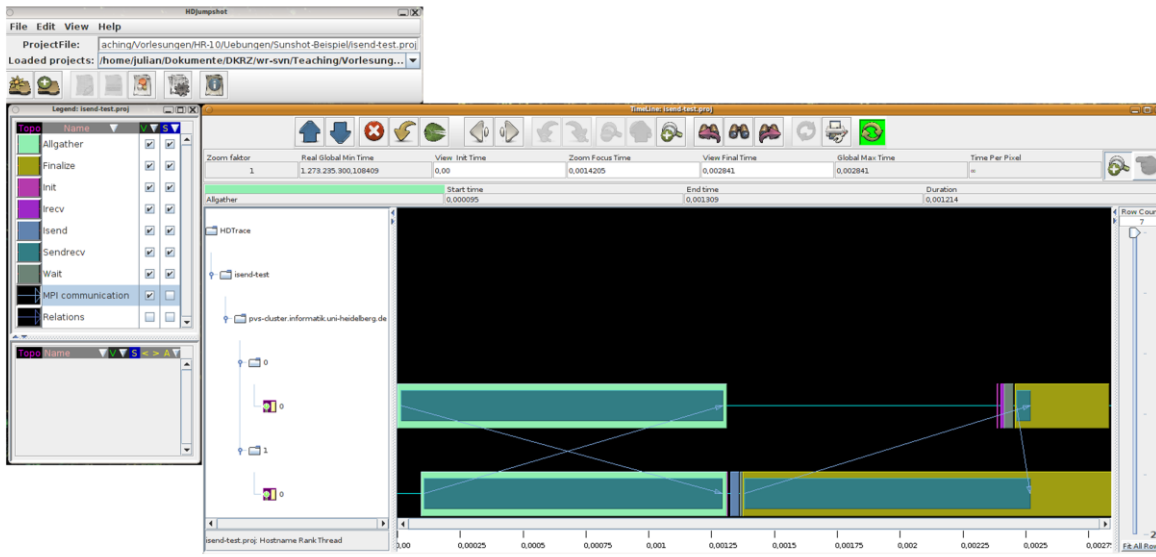
HDTrace ist ein Trace-Format wie z. B. SLOG2 und OTF, bietet aber weitergehende Funktionalität.

Sunshot: Allgemein

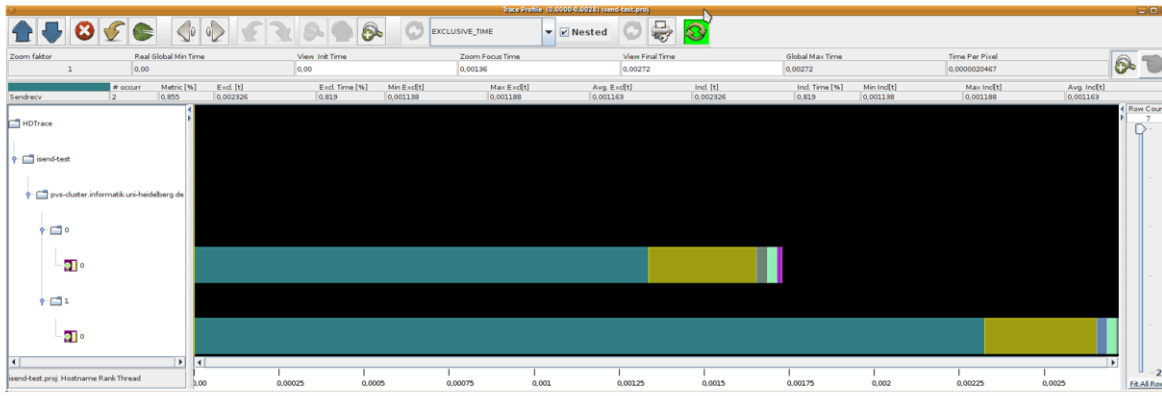


Sunshot: Detailinformationen

Sunshot: Beziehungen zwischen Ereignissen



Sunshot: Profile



Praktische Fragestellung

- Mein Programm hat 100 Funktionen, läuft 5 Tage lang auf 256 Knoten mit 512 Prozessen
- Frage: Wo ist der Leistungsengpass im Programm?

- Mit Vampir/Sunshot: Langes Betrachten bunter Bilder(?)
- Mit Paradyn: Halbautomatisches Auffinden des Engpasses

Online-Werkzeug Paradyn

- Langlaufendes Projekt an der University of Wisconsin-Madison unter Barton Miller
- Zwei Haupteigenschaften
  - Performance Consultant
    Findet halbautomatisch Leistungsengpässe
  - Dyninst
    Dynamische Instrumentierung des Binär-Codes

Paradyn: Messszenario

- Wenn man eine Idee hat, wo das Problem liegt
  - Wähle Komponenten, Leistungsmetriken und Darstellungsformen
    - Die notwendige Instrumentierung wird in das Programm eingebaut und aktiviert
    - Aufgelaufene Messwerte werden zyklisch ausgelesen
  - Analysiere die dargestellten Informationen
  - Umbau des Programms oder ähnliches

Paradyn: Performance Consultant

- Wenn man nicht weiß, wo das Problem liegt
  - Starte automatische Suche durch Performance Consultant
    - Eine vordefinierte Menge von Hypothesen wird getestet (**welches** Problem liegt vor?)
    - Instrumentierung wird eingebaut und die Messungen aktiviert
    - Schrittweise Verfeinerung, um das Problem zu finden (**wo** liegt das Problem?)
    - Unterteile das Programm in Phasen und setze die Untersuchung fort (**wann** tritt das Problem auf?)
  - W^3-Modell des Performance Consultant

Paradyn: Performance Consultant…

- Beantwortet drei Fragen
  - Warum ist das Programm so langsam?
  - Welche Code-Teile sind problematisch?
  - Wann tritt das Problem auf?
- Performance Consultant hat eine Menge an Hypothesen eingebaut und kennt den Code-Aufbau

Optimierung des Programms

- Was kann man denn nun konkret verändern?
  - Code-Aufbau und Datenstrukturen
  - Verteilung der Daten zu den Prozessen
  - Kommunikation
  - Synchronisation
  - Abbildung der Prozesse auf die Prozessoren

Optimierung des Programms…

- Code-Aufbau und Datenstrukturen
  - Bessere Nutzung des Caches durch geänderte Datenstrukturen und Code-Strukturen
    - Anordnung der Feld-Elemente
    - Aufbau der Schleifen
  - Datenstrukturen so aufbauen, dass man einfach senden und empfangen kann

Optimierung des Programms…

- Verteilung der Daten zu den Prozessen
  - Gleichverteilung der Daten nur, wenn das auch Gleichverteilung der Lasten bedeutet
  - Bei Gitterstrukturen
    - Kanten, entlang derer kommuniziert wird, minimieren
  - Evtl. dynamische Umverteilung der Daten ermöglichen

Optimierung des Programms…

- Kommunikation
  - So **selten** wie möglich wegen Zeitverlust
  - Möglichst große Pakete wegen Aufsetzzeit
  - Möglichst **frühzeitig senden**, damit Empfänger die Daten schnell bekommen
  - Möglichst **spät empfangen**, damit Sender Zeit gewinnen
  - Möglichst **asynchrones Empfangen** und Überlagerung mit anderen Berechnungen

Optimierung des Programms…

- Synchronisation
  - Möglichst selten globale Synchronisation
  - Möglichst selten globale Kommunikation, da diese meist auch synchronisieren

  - Zuviel Synchronisation: Zu langsam
  - Zuwenig Synchronisation: Komplizierte Fehler

Optimierung des Programms…

- Abbildung der Prozesse auf die Prozessoren
  - Gleichbelastung der Prozessoren erzielen
  - Minimierung der Kommunikation zwischen den Prozessoren erzielen
  - Scheduling des Betriebssystems auf dem einzelnen Knoten bedenken

Messmethodik

- Ziel: Physikalische Größen quantitativ erfassen
  - Z. B. Durchsatz in MB/s, Latenz, Speedup
- Veröffentlichte Messwerte sollten wissenschaftlich fundiert erhoben werden
  - Messwerte sind fehlerbehaftet
  - Es ist üblicherweise nicht ausreichend nur eine einzelne Messung durchzuführen

Messfehler

- Zufällige Fehler
  - Heben sich bei unendlicher Messung im Mittel auf
  - Ergebnisse variieren immer etwas
    - Natürliche Varianz (Netzwerkübertragungen, Speicherzugriffe)
    - Betriebssystemaktivitäten im Hintergrund
  - Größere Abweichungen sind auch möglich
    - Benutzung der Ressourcen durch andere Benutzer
    - Hardwarefehler (RAID-Rebuild, Paketverluste, …)
    - Lastbalancierung
- Systematische Fehler
  - Heben sich bei unendlicher Messung im Mittel nicht auf
  - Falsche Messmethodik/-realisierung

Messfehler...

- Was kann man dagegen unternehmen?
  - Wohldefinierte Hard-/Softwareumgebung
  - Dokumentation der Versuchsanordnung
  - Äußere Einflüsse minimieren
    - Z. B. Einzelbenutzerbetrieb während der Messung
  - Messdauer erhöhen
  - Messungen wiederholen
    - Eliminiert zufällige Fehler
  - Vergleichen der Messwerte mit erwarteter Leistung
    - Z. B. durch Modellierung

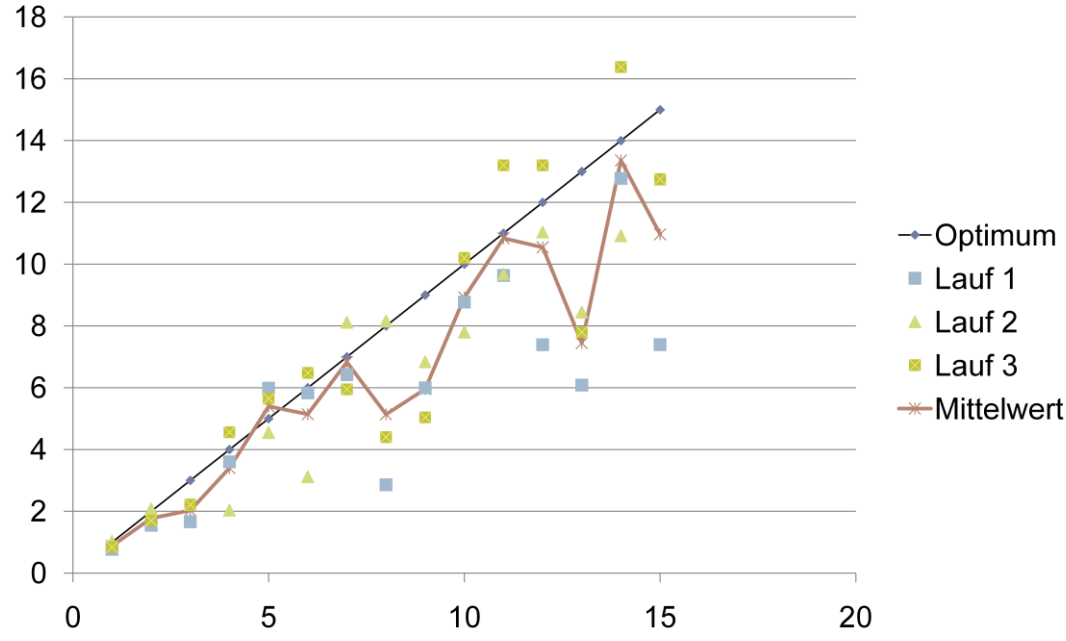
Die Dokumentation der Versuchsanordnung ist wichtig um eine Reproduzierbarkeit zu gewährleisten.

Eine Minimierung der äußeren Einflüsse ist nicht immer möglich, insbesondere in Mehrbenutzersystemen.

Wiederholungen sollten zu verschiedenen Zeiten gemacht werden um die Einflüsse von temporären Störungen zu reduzieren.

Kurzzeitige Einflüsse können durch länger andauernde Testläufe reduziert werden bzw. im Mittel bestimmt werden.
Aber eigentlich wollen wir so kurz wie möglich testen um das System für sinnvolle Aufgaben zu nutzen.

Beispiel: Speedup



Jeder Punkt ist ein gemessener Wert.
Der optimal zu erreichende Speedup ist ebenfalls dargestellt.

Ist der Mittelwert der reale Wert?

Statistische Methoden

- Mittelwert (Erwartungswert)
- Varianz
- Standardabweichung (σ = Wurzel der Varianz)
- Normalverteilung
  - 68,3% im Intervall σ um den Mittelwert
  - 95,4% im Intervall 2σ um den Mittelwert
  - 99,7% im Intervall 3σ um den Mittelwert
  - Ausreißer > 2σ bis 3σ vom Mittelwert entfernt
- Viele Zufallsgrößen sind annähernd normalverteilt
  - Obige Aufstellung gilt als Faustregel

Hier nur kurz angeschnitten. Es gibt auch Tests um zu prüfen ob die gemessenen Messwerte eine (Normal-)Verteilung sein könnten und wie wahrscheinlich das ist.

Leistungsmessung
Zusammenfassung

- Leistungsmessung, -analyse und Programmoptimierung ist ein zyklischer Prozess
- Verschiedene Werkzeugtypen stehen zur Auswahl: Offline- und Online-Werkzeuge
- Ziel ist die automatische Erkennung und Beseitigung von Leistungsengpässen
- Vampir ist eines der leistungsfähigsten Offline-Werkzeuge
- Paradyn ist das leistungsfähigste Online-Werkzeug
- Insbesondere Optimierung der Kommunikation ist wichtig für die Programmeffizient
- Die Messmethodik ist wichtig, um sinnvolle Ergebnisse zu erhalten