PROJEKTAUFGABE PIXCHECKTANGLE

RAPHAEL DRECHSLER

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 PROBLEMBESCHREIBUNG

Gegeben ist ein quadratisches Raster von n x n Feldern. Jedes der Felder kann schwarz oder weiß gefärbt sein. Es ist ein Algorithmus zu implementieren, welcher

- eine Einfache Eingabe eines solche Rasters erlaubt
- als Rechteck zusammenhängende Felder im Raster erkennt
- die resultierenden Rechtecke ausgibt

Dabei soll der Algorithmus für die Ausführung auf dem Cluster-System implementiert werden.
Anschließend soll mittels Laufzeitmessungen die Effizienz der Parallelisierung betrachtet werden. Dafür ist es erforderlich den Algorithmus als sequentiellen Algorithmus ausführen zu können.

2 BESCHREIBUNG DER IMPLEMENTIERUNG

Die Implementierung ist in zwei Programmen umgesetzt.

- *gridGenerator.c*: Programm zum generieren von *.txt*-Dateien, in denen das Raster gespeichert ist.
- *pixCheckTangle.c*: Programm zur Überprüfung des als *.txt*-Datei übergebenen Rasters auf Rechtecke.

In den Folgenden Abschnitten wird die Funktionalität der Programme beschrieben.

2.1 gridGenerator: Generieren der Files

Das Programm *gridGenerator.c* wird per mpirun-Befehl über die Konsole aufgerufen.

```
mpirun gridGenerator.c [gridfile.txt]
```

Wird dabei eine zuvor durch den gridGenerator erzeugte .txt-Datei als Parameter angegeben, kann diese Datei bearbeitet werden. Andernfalls wird eine neue Datei erstellt. Der Programm-Ablauf ist im Folgenden als Nassi-Shneiderman-Diagramm dargestellt.

Der Vollstädige Code ist im Anhang ab Seite (DRT oder Verweis angeben)... gelistet.

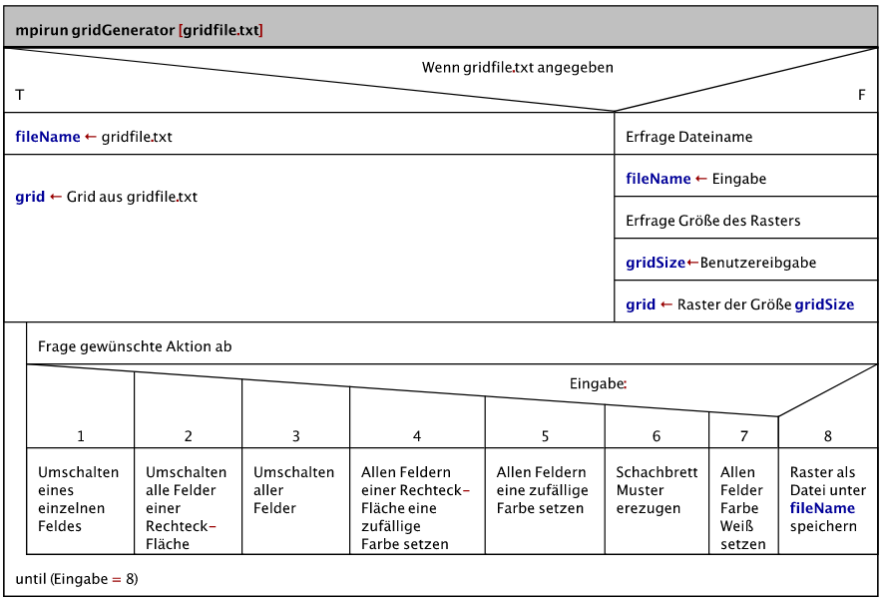


Abbildung 1: Funktionalität von *gridGenerator.c* dargestellt als Nassi-Shneiderman-Diagramm

2.2 pixCheckTangle: Beschreibung des Sequentiellen Algorithmus

Das Programm *pixCheckTangle.c* wird per mpirun-Befehl über die Konsole aufgerufen. Dabei ist als dem Aufruf eine Raster-Textdatei als Argument zu übergeben. Die übergebene Datei wird dann auf Rechtecke überprüft.

```
mpirun pixCheckTangle.c gridfile.txt
```

Für den Fall, dass nur ein Prozessor an der Ausführung des Programmes beteiligt ist, wird die sequentielle Variante des Algorithmus ausgeführt. Diese ist im Folgenden als BPMN-Daigramm dargestellt.

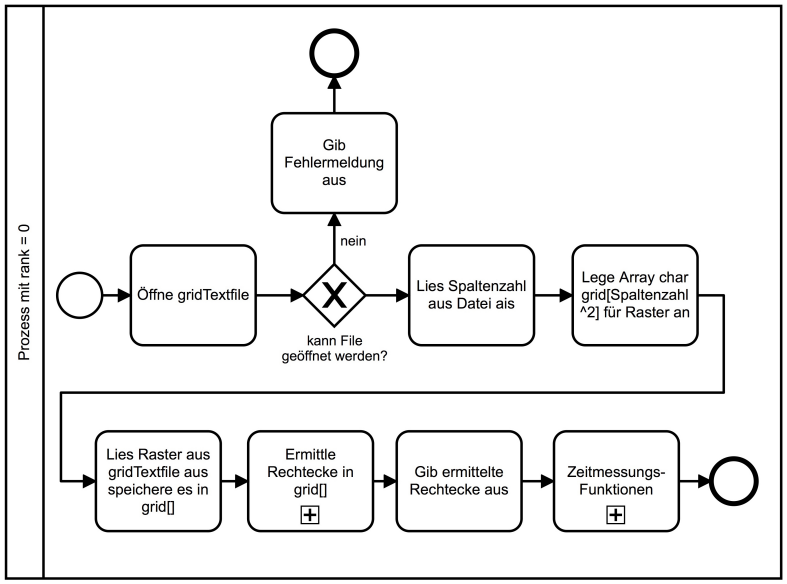


Abbildung 2: Sequentieller Ablauf von *pixCheckTangle.c* dargestellt als BPMN-Diagramm

Der vollständige Code zum Algorithmus ist im Anhang enthalten. Die Hauptfunktionalität findet sich dabei in dem Codeabschnitt, welcher ausgeführt wird, wenn der ausführende Prozess den Rang 0 hat und es nur einen Prozess gibt.

Die Funktionalität zum Ermitteln der Rechtecke, wird im Kapitel 2.4 näher beschrieben. Nach welchem Prinzip die Zeitmessung erfolgen, wird in Kapitel 3.1 beschrieben.

2.3 pixCheckTangle: Beschreibung des Cluster-Algorithmus

Für den Fall, dass mehrere Prozessoren an der Abarbeitung des Programmes beteiligt sind, wird die Cluster-Variante des Algorithmus ausgeführt.
Dabei übernimmt der Prozess mir dem Rang 0 die Rolle des Master-Prozesses. Alle übrigen Prozesse übernehmen die Rolle von Worker-Prozessen.
Das Raster wird dann in vom Master-Prozess in mehrere Teile zerlegt. Jeder Worker-Prozess übernimmt dann die Abarbeitung eines Teil-Rasters.

Beispiel für Verarbeitung durch Cluster-Prozess

Beispielsweise wird das folgende Raster betrachtet. Das Zeichen # repräsentiert dabei ein schwarzes Feld, weiße Felder werden durch Leerzeichen dargestellt.

```
   | 1 2 3 4 5 6 7
---|---------------
1  | # # #   #   #
2  |       #   #
3  | # #     #   #
4  | # #   #   #
5  | #   #   #   #
6  |   #   #   #
7  | #   #   # # #
```

Abbildung 3: Ein Raster-Beispiel

Für den Fall, das an der Abarbeitung des Programmes 4 Prozesse beteiligt sind, wird das Raster vom Master-Prozess in drei Teil-Raster geteilt. Die jeweiligen Teile werden dann den Worker-Prozessen zugewiesen und von diesen abgearbeitet.

```
   | 1 2 3 4 5 6 7
---|-------------------------
1  | # # #   #   #   Teil-Raster 1
2  |       #   #     -> Bearbeitet WorkerProzess 1
---|-------------------------
3  | # #     #   #   Teil-Raster 2
4  | # #   #   #     -> Bearbeitet WorkerProzess 2
---|-------------------------
5  | #   #   #   #   Teil-Raster 3
6  |   #   #   #     -> Bearbeitet WorkerProzess 3
7  | #   #   # # #
```

Abbildung 4: Aufteilung des Beispiel-Rasters bei 4 Prozessen

Die Worker-Prozesse können nun feststellen, ob es sich bei einzelnen/zusammenhängenden Pixeln (im Folgenden als Figur bezeichnet) um Rechtecke handelt oder nicht. In der Folgenden Grafik, werden Figuren, die kein Rechteck darstellen mit einem *X* dargestellt. Rechtecke werden weiterhin durch das Zeichen # repräsentiert.

```
  WorkerProzess 1          WorkerProzess 2          WorkerProzess 3

   | 1 2 3 4 5 6 7          | 1 2 3 4 5 6 7          | 1 2 3 4 5 6 7
---|---------------      ---|---------------      ---|---------------
1  | # # #   #   #       3  | # #     #   #       5  | #   #   #   #
2  |       #   #         4  | # #   #   #         6  |   #   #   X
                            |                        7  | #   #   X X X
```

Abbildung 5: Worker-Prozesse identifizieren Nicht-Rechteck-Figuren

Wenn ein Rechteck in einem Teil-Raster zu einem Rand der Teil-Rasters reicht und an diesem Rand im gesamten Raster ein weiteres Raster angrenzt, so könnte die Figur im angrenzenden Teil-Raster fortgesetzt werden. In diesem Fall ist also durch das Verarbeiten des Teil-Rasters keine Aussage darüber möglich, ob es sich um ein tatsächlich um ein Rechteck handelt. Entsprechende Rechtecke (Im Weiteren als potentielle Rechtecke) werden in der folgenden Grafik als *?* gekennzeichnet.
Das Ergebnis der Abarbeitung der Teil-Raster in den Worker-Prozessen sieht also wie folgt aus:

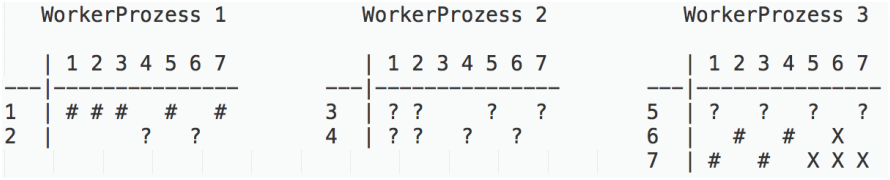


Abbildung 6: Worker-Prozesse identifizieren Rechtecke und potentielle Rechtecke

Nach dem Verarbeiten der Teil-Raster durch die Worker-Prozesse, muss die finale Überprüfung der potentiellen Rechtecke vom Master-Prozess übernommen werden.

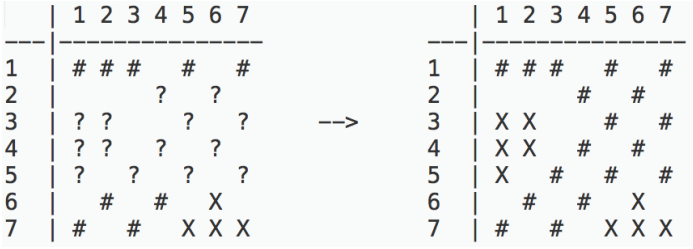


Abbildung 7: Ein Raster-Beispiel

Damit liegt dem Master-Prozess nun das vollständig überprüfte Raster vor. Die Einzelnen Rechtecke können nun dem Benutzer ausgegeben werden.

Kommunikation und Ablauf im Cluster-Prozess
Wie dabei Arbeits-Aufteilung und die Kommunikation zwischen den Prozessen verläuft, sei im Folgenden als BPMN-Diagramm dargestellt.

Der vollständige Code findet sich im Anhang. Die wesentlichen Funktionalitäten für den Master-Prozess finden sich dabei in dem Codeabschnitt, welcher ausgeführt wird, wenn der ausführende Prozess den Rang 0 hat und es mehr als einen Prozess gibt. Der Code für die Worker-Prozesse findet sich im Abschnitt, der ausgeführt wird, wenn der Rang des Prozesses nicht 0 ist.

Die Funktionalität zum Ermitteln der Rechtecke, wird im Kapitel 2.4 näher beschrieben. Nach welchem Prinzip die Zeitmessung erfolgen, wird in Kapitel 3.1 beschrieben.

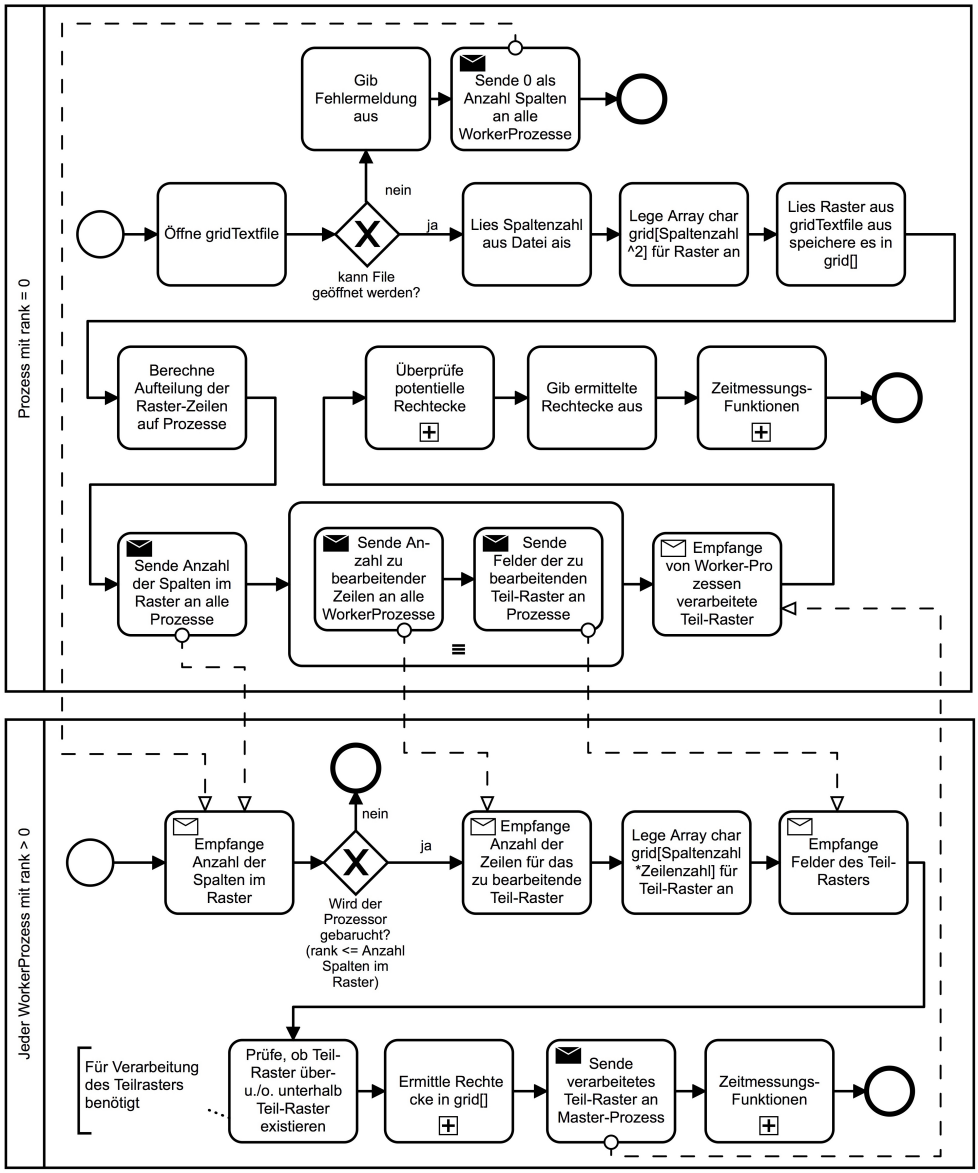


Abbildung 8: Ablauf des Cluster-Prozesses von *pixCheckTangle.c* dargestellt als BPMN-Diagramm

2.4 pixCheckTangle: Funktion für das Abarbeiten eines (Teil-)Rasters

Im Wesentlichen folgt das Vorgehen für die Unterscheidung zwischen Rechtecken und Nicht-Rechteck-Figuren dem folgenden Prinzip:

1. Erkenne Spalten-Reichweite der betrachteten Figur anhand der erster Zeile
2. Ist In Zeile über der Figur innerhalb der Spalten-Reichweite ein Feld schwarz, ist die Figur kein Rechteck
3. Enthält eine Zeile unterhalb der ersten innerhalb der Spalten-Reichweite der Figur schwarze und weiße Felder, ist die Figur kein Rechteck
4. Grenzt an eine schwarze Zeile der Figur (von Spalten-Reichweite eingegrenzt) ein schwarzes Feld, ist die Figur kein Rechteck

Zusätzlich ist dabei zu berücksichtigen, ob es sich bei einem erkannten Rechteck, wie weiter oben beschrieben, nur um ein potentielles Rechteck ist.
Die Implementierung der Funktion folgt danach dem folgenden, als Pseudo-Code

dargestellten Algorithmus:

```
   Daten : Raster, InfoZuAngenzedeTeilraster
   Ergebnis : RasterVerarbeitet
 1 für jedes Feld in Raster zeilenweise gelesen tue
 2     wenn Feld ist Schwarz und nicht markiert dann
 3         FigurIstRechteck ← wahr;
 4         startZeile ← aktuelleZeile;
 5         startSpalte ← aktuelleSpalte;
 6         endeZeile ← aktuelleZeile;
 7         endeSpalte ← Spalte von letzem Feld in startZeile, das mit aktuellem Feld
             zusammenhängt;
 8         wenn In Zeile über startZeile im Bereich von startSpalte bis endeSpalte ist
             ein schwarzes Feld dann
 9             FigurIstRechteck ← falsch;
10         sonst
11             RechteckGeprueft ← falsch;
12             solange FigurIstRechteck und nicht RechteckGeprueft tue
13                 wenn Zeile unter endeZeile enthält schwarze und weiße Felder dann
14                     FigurIstRechteck ← falsch;
15                 sonst
16                     wenn Zeile unter endeZeile enthält nur schwarze Felder dann
17                         wenn In Zeile unter endeZeile ist feld links von startSpalte
                             oder/und rechts endeSpalte von schwarz dann
18                             FigurIstRechteck ← falsch;
19                         sonst
20                             endeZeile ← endeZeile + 1;
21                         Ende
22                     sonst
23                         RechteckGeprueft ← wahr;
24                     Ende
25                 Ende
26             Ende
27         Ende
28         wenn FigurIstRechteck dann
29             wenn Figur liegt an Raster-Rand, der an weiterem Teil-Raster angrenzt dann
30                 Markiere alle schwarzen Felder innerhalb der Reichweite, die von
                     startZeile,endeZeile,startSpalte und endeSpalte aufgespannt
                     wird als potentielles Rechteck-Feld;
31             sonst
32                 Markiere alle schwarzen Felder innerhalb der Reichweite, die von
                     startZeile,endeZeile,startSpalte und endeSpalte aufgespannt
                     wird als Rechteck-Feld;
33             Ende
34         sonst
35             Markiere alle schwarzen Felder innerhalb der Reichweite, die von
                 startZeile,endeZeile,startSpalte und endeSpalte aufgespannt wird
                 als Nicht-Rechteck-Feld;
36         Ende
37     Ende
38 Ende
```

Der Code für die Funktionalität findet sich im Anhang im Programm *pixCheckTangle.c* in der Funktion *processSubgrid*.

2.5 pixCheckTangle: Funktion für das Festlegen potentieller Rechtecke

Für die Funktion zum Überprüfen der potentiellen Rechtecke, sobald die Worker-Prozesse die Teil-Raster abgearbeitet haben, wird die im Folgenden per Pseudo-

Code beschriebene Vorgehensweise angewandt. Die Kernidee ist dabei nur noch Stichprobenartige Prüfungen zu machen, anstatt die Schritte aus dem oben aufgezeigte Prozess zu wiederholen.

```
   Daten : Raster,zuPruefendeZeile InfoZuRasterumbrueche
   Ergebnis : RasterVerarbeitet
 1 für Jedes schwarze Feld in zuPruefendeZeile mit Markierung =
     potentielles-Rechteck-Feld tue
 2 |   startZeile ← aktuelleZeile;
 3 |   startSpalte ← aktuelleSpalte;
 4 |   endeZeile ← aktuelleZeile;
 5 |   endeSpalte ← Spalte von letzem Feld in startZeile, das mit aktuellem Feld
   |     zusammenhängt;
 6 |   wenn zuPruefendeZeile ist Zeile oberhalb von Umbruch dann
 7 |   |   wenn In Zeile oberhalb startZeile ist in Reichweite von startSpalte bis
   |   |     endeSpalte mindestens ein schwarzes Feld dann
 8 |   |   |   zeilenMarker ← MIN(End-Zeile der Figur, Zeile vor folgendem
   |   |   |     Teil-Raster-Umbruch) ;
 9 |   |   |   Markiere alle schwarzen Felder innerhalb der Reichweite, die von
   |   |   |     startZeile,zeilenMarker,startSpalte und endeSpalte aufgespannt
   |   |   |     wird als Nicht-Rechteck-Feld;
10 |   |   |   Fahre bei Feld nach der endeSpalte fort.
11 |   wenn hinter zuPruefendeZeile folgt min 1 Teil-Raster-Umbruch dann
12 |   |   solange Ein Teilraster-Bruch folgt tue
13 |   |   |   wenn Figur setzt unter Teilraster-Bruch fort dann
14 |   |   |   |   wenn In erster Zeile nach Teilraster-Bruch ist die Laenge der Figur
   |   |   |   |     ungleich nicht mit der bisherigen Spalten-Reichweite dann
15 |   |   |   |   |   FigurIstRechteck ← falsch;
16 |   |   Ende
17 |   |   startZeile ← Anfang der Figur;
18 |   |   wenn FigurIstRechteck dann
19 |   |   |   zeilenMarker ← Ende der Figur;
20 |   |   |   Markiere alle schwarzen Felder innerhalb der Reichweite, die von
   |   |   |     startZeile,zeilenMarker,startSpalte und endeSpalte aufgespannt
   |   |   |     wird als Rechteck-Feld;
21 |   |   |   Fahre bei Feld nach der endeSpalte fort.
22 |   |   sonst
23 |   |   |   zeilenMarker ←Letzte Zeile in der Rechteck-Bedingug fuer Figur
   |   |   |     erfuellt war;
24 |   |   |   Markiere alle schwarzen Felder innerhalb der Reichweite, die von
   |   |   |     startZeile,zeilenMarker,startSpalte und endeSpalte aufgespannt
   |   |   |     wird als Nicht-Rechteck-Feld;
25 |   |   |   Fahre bei Feld nach der endeSpalte fort.
26 |   |   Ende
27 |   sonst
28 |   |   zeilenMarker ← Ende der Figur;
29 |   |   Markiere alle schwarzen Felder innerhalb der Reichweite, die von
   |   |     startZeile,zeilenMarker,startSpalte und endeSpalte aufgespannt wird
   |   |     als Rechteck-Feld;
30 |   |   Fahre bei Feld nach der endeSpalte fort.
31 |   Ende
32 Ende
```

Der Code für die Funktionalität findet sich im Anhang im Programm *pixCheck-Tangle.c* in der Funktion *checkPotentialRectangle*.

3 MESSUNGEN

3.1 Vorgehen beim Messen

Für die Ermittlung des Speedups sollen Zeitmessungen durchgeführt werden. Da ein großer Anteil der Laufzeit für die Netzwerkkommunikation benötigt wird, soll hier zunächst eine Vorüberlegung angestellt werden, wie die Netzwerk-Kommunikation aus der Ausführungszeit herauszurechnen ist.
Die Folgende schematische Darstellung zeigt die von den einzelnen Prozessen benötigte Zeit, in einem Cluster-Prozess mit 3 Worker-Prozessen.

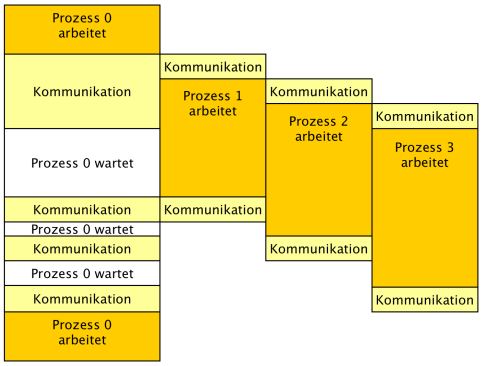


Abbildung 9: Skizze: Zeitbedarf mit Netzwerkkommunikation

Nimmt man hierbei an, dass die Netzwerkkommunikation keine Zeiteinheiten kostet, so erhält man näherungsweise die folgende schematische Darstellung.

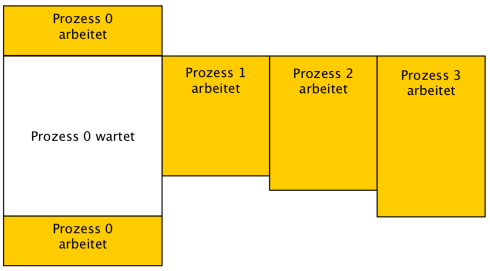


Abbildung 10: Skizze: Zeitbedarf ohne Netzwerkkommunikation

Daraus geht hervor, dass sich die zu messende Zeit unter Ausschluss der Netzwerkkommunikation näherungsweise aus der von Prozess 0 benötigten aktiven Arbeitszeit und der benötigten Arbeitszeit, des am längsten laufenden Worker-Prozesses zusammensetzt.
Die Zeit-Messungen werden im Code mittels **MPI_Wtime()**-Befehl realisiert. Zum ermitteln der maximalen Arbeitszeit aller Worker-Prozesse wird im Anschluss an die eigentliche Programm-Ausführung der **MPI_Reduce()**-Befehl genutzt.
Die Resultate der Zeitmessungen im Code werden mittels der Funktion *writeTimeToFile* in eine Datei geschrieben. Für die Ausführung von mehren Messreihen in einem Lauf werden die folgenden hilfs-Skripte/Programme verwendet. Diese sind im Anhang einzusehen.

- *hostfileGenerator.c*: Generiert eine hostfile-Datei mit n Prozessoren.
- *bsCreateHostfiles.sh*: ruft *hostfileGenerator.c* auf, um mehrere hostfiles zu erzeugen.
- *bsRunGridTest.sh*: Ruft *pixCheckTangle* per MPI für eine Raster-Datei und mehre Prozesse auf.

Gemessen werden:

- Laufzeit mit und ohne Netzwerkkommunikation
- Für Raster mit *n={100/1000/10000}* Feldern
- Pro Rastergröße: Messungen mit je *p={1,2,..50}* Prozessen.
- Pro Rastergröße und Prozess-Anzahl: drei verschiedene Probleminstanzen
  - Alle Felder Weiß
  - Alle Felder Schwarz
  - Schachbrett-Raster

Jede Messung wird drei mal durchgeführt. Als Messwert wird der daraus gebildete Mittelwert betrachtet.

3.2 Messergebnisse

3.3 Interpretation der Messergebnisse

4 FAZIT

4.1 Einschätzung des Nutzens einer Parallelisierung

4.2 Mögliche Optimierungen

5 BLA

A statement requiring citation [1].

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Some mathematics in the text: $\cos \pi = -1$ and α.

6 METHODS

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

1. First item in a list
2. Second item in a list
3. Third item in a list

6.1 Paragraphs

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

PARAGRAPH DESCRIPTION Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

DIFFERENT PARAGRAPH DESCRIPTION Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

6.2 Math

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

$$\cos^3\theta = \frac{1}{4}\cos\theta + \frac{3}{4}\cos 3\theta \tag{1}$$

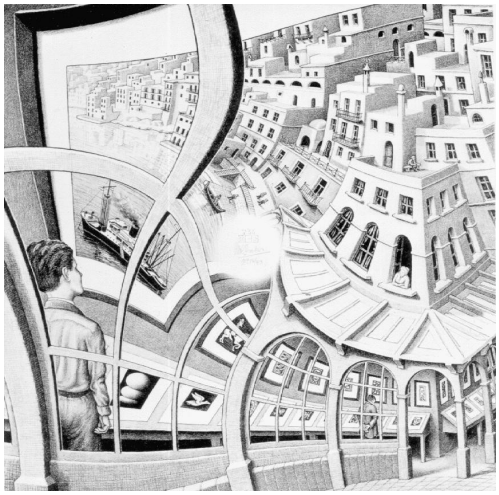


Abbildung 11: An example of a floating figure (a reproduction from the *Gallery of prints*, M. Escher, from http://www.mcescher.com/).

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Definition 1 (Gauss)**.** To a mathematician it is obvious that $\int_{-\infty}^{+\infty} e^{-x^2}\,dx = \sqrt{\pi}$.

Theorem 1 (Pythagoras)**.** *The square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides.*

Proof. We have that $\log(1)^2 = 2\log(1)$. But we also have that $\log(-1)^2 = \log(1) = 0$. Then $2\log(-1) = 0$, from which the proof. □

7 RESULTS AND DISCUSSION

Reference to Figure 11.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

7.1 Subsection

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

7.1.1 *Subsubsection*

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

WORD Definition

CONCEPT Explanation

IDEA Text

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

- First item in a list
- Second item in a list
- Third item in a list

7.1.2 *Table*

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Tabelle 1: Table of Grades

Name		
First name	Last Name	Grade
John	Doe	7.5
Richard	Miles	2

Reference to Table 1.

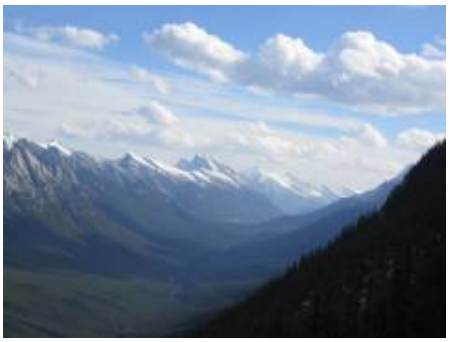
7.2 Figure Composed of Subfigures

Reference the figure composed of multiple subfigures as Figure 12 on the following page. Reference one of the subfigures as Figure 12b on the next page.

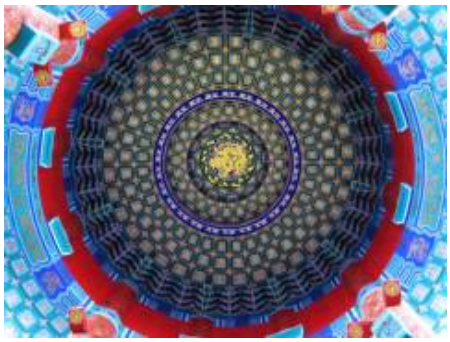
Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue,

(a) A city market.

(b) Forest landscape.



(c) Mountain landscape.



(d) A tile decoration.

Abbildung 12: A number of pictures with no common theme.

nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

REFERENCES

[1] A. J. Figueredo and P. S. A. Wolf. Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330, 2009.