Praktikum Parallele Programmierung 2013

Parallele Verkehrssimulation

Marcel Ellermann, Martin Poppinga, Miriam v.Heereman

Konzept

- Eine reale Karte nehmen, z.B. Hamburg
- Autos möglichst realistisch fahren lassen
  - Geschwindigkeit
  - Verkehrsregeln
  - Routen
- Erkennen wo Staus entstehen und wie sie verhindert werden könnten

Auswahl der Idee

- Eine Welt mit Straßen (Edges) und Kreuzungen (knoten) – Graph
- Jede Straße hat Spuren auf denen Autos sind
- Autos verhalten sich im Straßenverkehr autonom
- Einzelende Autos und Straßen können parallel verarbeitet werden

Aufbau der Software

- Preprocessing
  - Einlesen und vorverarbeiten der Karte
- Die Simulation
  - Das eigentliche Programm, das es zu parallelisieren gilt
- Postprocessing
  - Die grafische Ausgabe

Preprocessing

- Einlesen der Daten von OpenStreetMap
- Aussortieren von überflüssigen Daten
- Clustern der Ampeln
- Daten in einheitliches Format bringen
- Erstellen von Spawnern
- Erstellen der Vorfahrtregeln
  - Hauptstraßen, Kreisverkehre, Rechts vor links

Simulation - Die Datenstruktur

- Kreuzungen
  - Vorfahrtsregelungen, Ampeln, Position der Straßen
- Straßen
  - Typ, Geschwindigkeit, Spuren, Länge, Richtung
- Ampelsysteme
  - Position der Ampeln zueinander
- Spawner
  - Ort, Größe, Typ
- Autos
  - Position, Geschwindigkeit, Richtung

Die Karte (Hamburg)

Simulation - Bewegung

- Autos fahren von Spawner zu Spawner
  - Abhängig von Uhrzeit
- A* Algorithmus für die Navigation
- Bezug zur Umgebung
  - Abstand zu anderen Autos
  - Brems- und Beschleunigungsphasen
  - Anhalten bei Stau, Vorfahrt, Ampeln,

Postprocessing

- Umwandeln von internen Positionen auf Koordinaten
- Erstellen einer Karte aus den OSM-Daten
- Einzeichnen der Autos
- Ganze Karte oder Ausschnitte
- Erstellen von Videos aus Daten

Videos

Optimierung

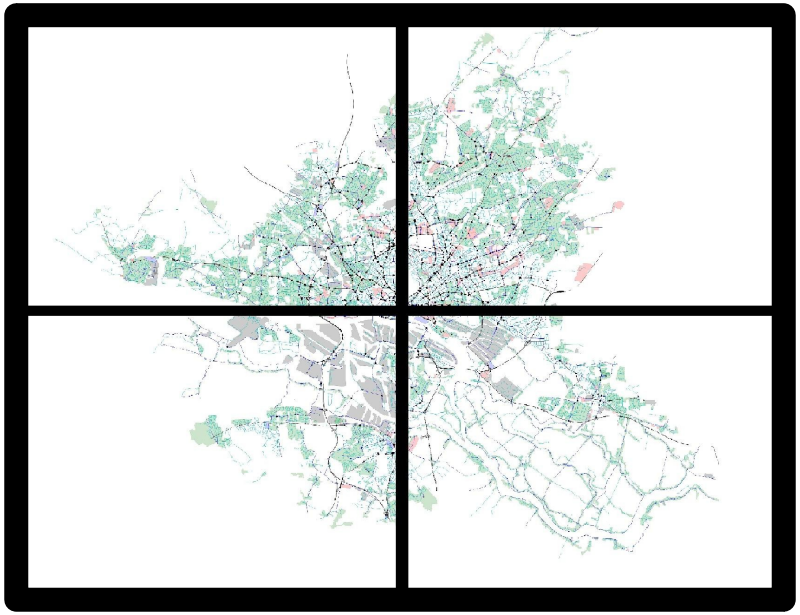
- Werkzeuge
  - Vampir, Profiler, Trace, Valgrind
- Pathfinding
  - Heap
  - Hash Map für Heap handles
    - KeyHash über NodeID
- Reduzieren von Memmory Leaks von 50MB auf 150b (3600 Zeitschritte)

Parallelisierungskonzept (OpenMP)

- OpenMP
  - Autos können an mehren Orten gleichzeitig erstellt werden
    - Suche von Punkten und Pfaden
    - Geschwindigkeitsgewinn: 2-3x
  - Straßen
    - Straßen können Parallel verarbeitet werden
    - n Threads die jeweils 1/n der Straßen bearbeitet
    - logical time step pro Auto

Parallelisierungskonzept (MPI)

- MPI
  - Die Karte wird Aufgeteilt
  - Jeder Prozess erhält einen
  - Bereich
  - Verlässt ein Auto den Bereich, wird das Objekt an anderen Prozess übergeben

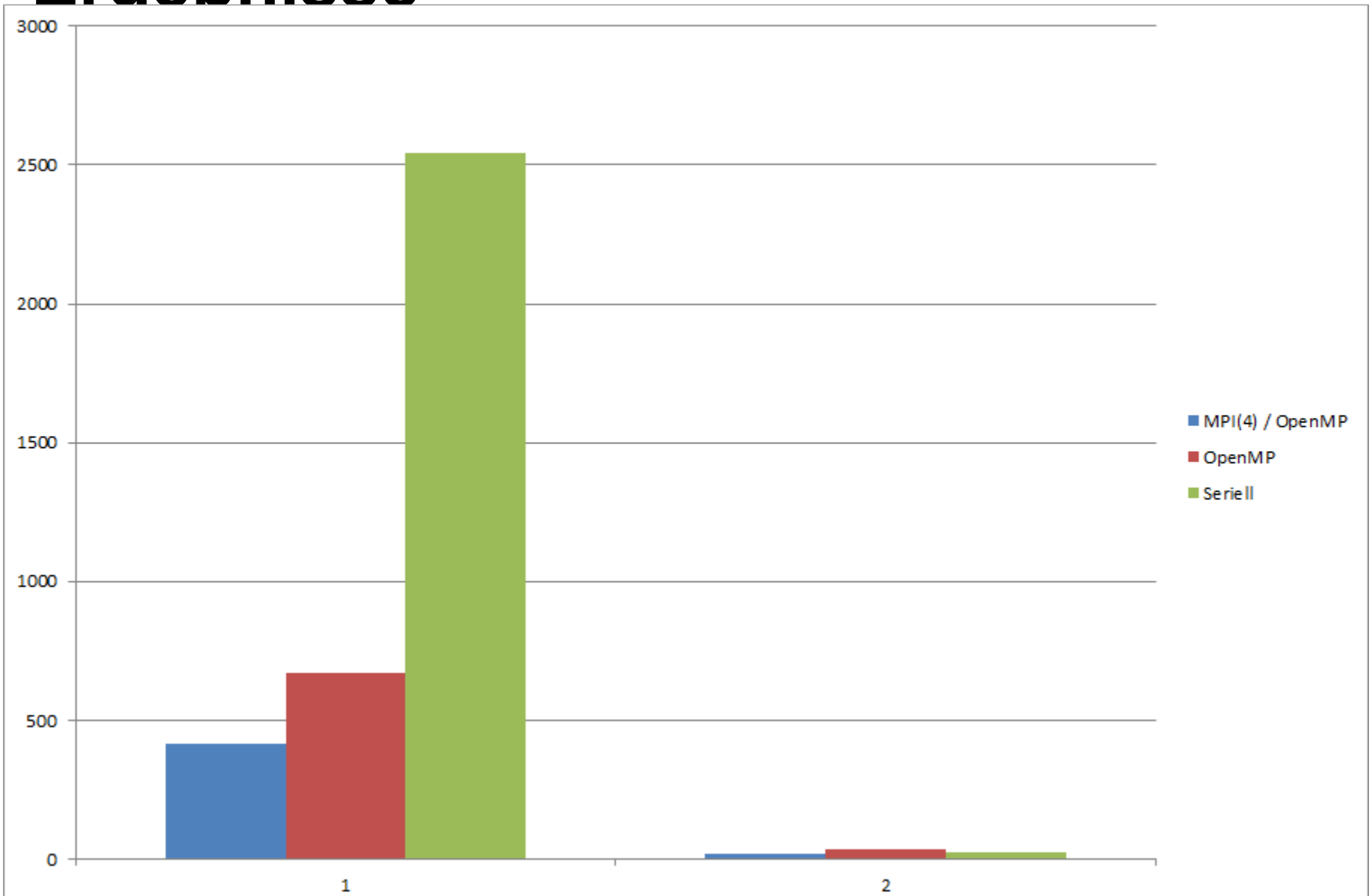


Weitere Optimierungen

- Präzise locks für einzelne Vector elemente
  - Nicht der ganze Vector wird gesperrt
  - Wichtig für Node locking
  - Wichtig für Straßen Zugriffe

- MPI, alle Nachrichten werden als ISend geschickt und am ende gewartet bis alle verarbeitet sind.

Ergebnisse



Fazit

- Lies sich sinnvoll Parallelisieren
- Sehr Komplex und Arbeitsaufwändig
  - Kann noch realistischer gemacht werden
    - (Vorfahrtsregelungen, Intelligentere Ampelsysteme, Stauumfahrung, etc.)
- Grafische Ausgabe muss anders gelöst werden

Fragen?

Vielen Dank für die Aufmerksamkeit!