

1 PG-Thema:

XAVER: Algorithm Engineering XXL

2 **Zeitraum:** WiSe 2006/07 & SoSe 07

3 **Umfang:** 8 SWS pro Semester

4 **Veranstalter:** Markus Chimani, Carsten Gutwenger, Karsten Klein,
Informatik LS 11, Tel. 755-7707, `vorname.nachname@cs.uni-dortmund.de`

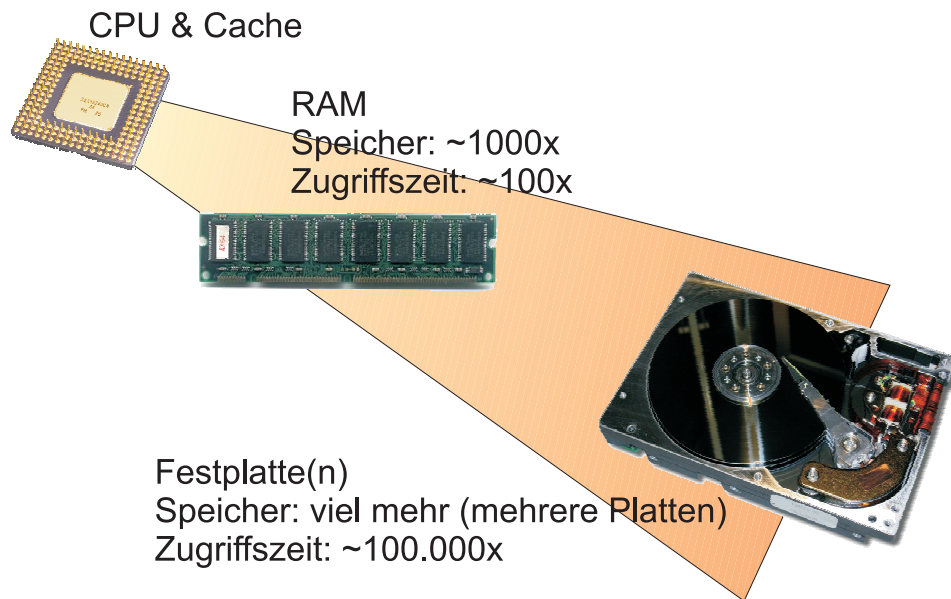
5 Aufgabe

Das noch sehr junge Forschungsgebiet des *Algorithm Engineering* befasst sich mit der Entwicklung und Evaluierung von Algorithmen und berücksichtigt insbesondere die Aspekte realer Rechnerarchitekturen. Während die klassische Algorithmik das sehr einfache von-Neumann Maschinenmodell verwendet und sich häufig mit der asymptotischen Worst-Case Analyse von Algorithmen begnügt, orientiert sich das Algorithm Engineering stärker an der Praxis:

- Moderne Prozessorarchitekturen sind komplex: Sie beinhalten Parallelismus (pipelining, branch prediction) und Multi-Level Caches.
- Der interne Speicher eines Computers ist begrenzt; wollen wir größere Datenmengen verarbeiten, dann müssen wir auf den sehr viel langsameren externen Speicher (Festplatten) ausweichen.
- Die in der Theorie „besten“ Algorithmen sind in vielen Fällen sehr komplex und daher schwierig und fehleranfällig zu implementieren; in der Praxis bevorzugt man daher einfache Algorithmen.
- In der Praxis gibt es häufig „typische“ Eingaben für einen Algorithmus und es ist sehr wichtig, dass ein Algorithmus für diese Eingaben schnell ist. Daher gibt es für viele Probleme Sammlungen von Eingabeinstanzen, so genannte Benchmark-Daten, mit denen Implementierungen von Algorithmen auf realen Rechnern evaluiert werden können. Auch sind konstante Faktoren, die bei der klassischen Laufzeitanalyse in der O -Notation verschwinden, in der Praxis sehr wohl von Bedeutung.

Im Rahmen dieser PG wollen wir uns mit Algorithmen für große Datenmengen beschäftigen. Gerade in den letzten Jahren ist die Flut der zu verarbeitenden Daten stark gewachsen. So misst die NASA das Datenvolumen von Satellitenbildern in Petabytes (10^{15} Bytes)! Auch in der Bioinformatik müssen große Mengen an Daten verarbeitet werden, die durch immer neue Methoden wie z.B. Genexpressionsanalyse produziert werden. Neue Screeningtechnologien erzeugen riesige Datenmengen mit mehr als 50 Millionen Werten pro Monat!

Für die Effizienz eines Algorithmus in der Praxis ist es wichtig, dass die Speicher-Hierarchie moderner Rechnerarchitekturen möglichst gut ausgenutzt wird. So ist der Zugriff auf den Cache-Speicher des Prozessors etwa 100-mal schneller als der Zugriff auf den Hauptspeicher, d.h. wir wollen möglichst wenige *Cache Misses* (Zugriff auf Daten die noch nicht im Cache sind) verursachen. Sind andererseits die zu verarbeitenden Datenmengen so gross, dass auch der Hauptspeicher nicht ausreicht, dann müssen die Daten teilweise auf externen Speicher wie Festplatten ausgelagert werden. Zugriffe auf den externen Speicher sind aber etwa 1000-mal langsamer als auf den Hauptspeicher; allerdings kann bei jedem Zugriff ein Block von konsekutiven Daten auf



einmal übertragen werden. Für die Effizienz des Algorithmus ist es wichtig, die Anzahl dieser Block-Transfers (*I/O-Zugriffe*) möglichst klein zu halten.

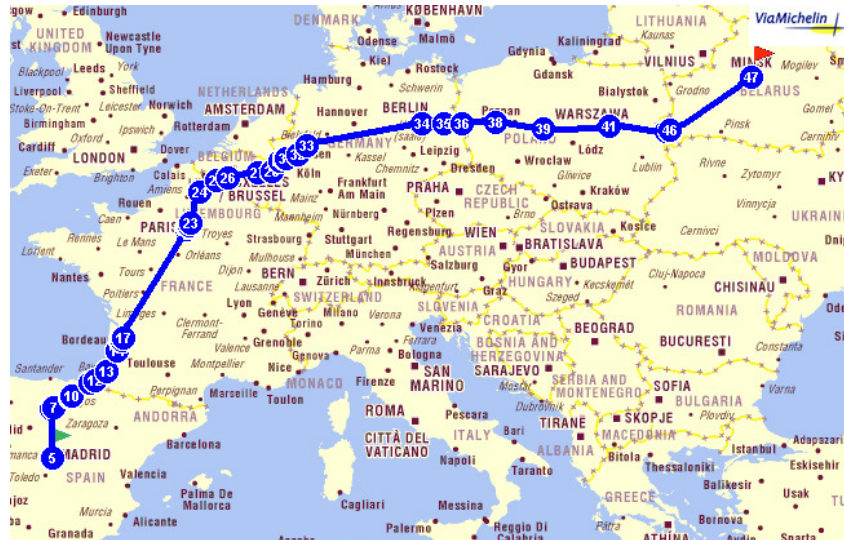
Wir konzentrieren uns im Rahmen dieser PG auf zwei algorithmische Probleme:

- *Sequenz-Suche in Protein-Datenbanken*. Dies ist ein typisches Problem der Bioinformatik, bei dem eine Vielzahl von als Zeichenketten codierten Proteinen durchsucht werden müssen. Die Datenmenge übersteigt die Größe des Hauptspeichers um ein Vielfaches; man löst dieses Problem durch externe Suffix-Arrays.
- *Kürzeste Wege in großen Graphen*. Für das Kürzeste Wege Problem findet derzeit eine DIMACS Challenge [Dem06] statt, die auch Benchmark-Daten von z.B. großen Straßennetzwerken zur Verfügung stellt (z.B. Europa mit 19 Millionen Knoten und 23 Millionen Kanten und die USA mit 24 Millionen Knoten und 29 Millionen Kanten).

Das Ziel der PG ist die Entwicklung einer in C++ geschriebenen Plattform zur Implementierung und Evaluierung von Externspeicher-Algorithmen (*EM-Algorithmen, external memory*), sowie die Implementierung verschiedener Varianten zum Aufbau von Suffix-Arrays und Algorithmen zur Berechnung von kürzesten Wegen in großen Graphen. Bei der Behandlung der algorithmischen Probleme soll die PG nicht nur bereits vorhandene Veröffentlichungen implementieren; ein besonderer Schwerpunkt soll auch die Erarbeitung und Umsetzung eigener Ideen zur Verbesserung der Algorithmen sein! Die einzelnen Aufgabenblöcke sind wie folgt:

1. *Externspeicher-Plattform XAVER (eXternal-memory Algorithms in a Versatile EnviRonment)*. Diese Plattform soll eine in C++ geschriebene Bibliothek sein, welche die Implementierung und Evaluierung von EM-Algorithmen unterstützt. Insbesondere regelt sie die Verwaltung des internen Speichers und den Zugriff auf den externen Speicher. EM-Algorithmen verwenden im Wesentlichen zwei Parameter: Die Größe M des internen Speichers und die Blockgröße B eines I/O-Zugriffs; letzteres ist die Anzahl konsekutiver Bytes, die bei einem I/O-Zugriff auf einmal in den internen Speicher übertragen werden. Entsprechend soll die Plattform auch die Simulation verschiedener Werte für M und B unterstützen. Zwei Simulationsmodi sollen dabei vorhanden sein:

- (a) Der interne *und* der externe Speicher werden im RAM des Computers gespeichert, d.h. $M + B \leq \text{RAM-Speicher}$. Beim Zugriff auf den „externen“ Speicher wird ei-



ne künstliche Verzögerung durchgeführt, um den langsameren Zugriff zu simulieren. Diese Variante erlaubt es, die Zugriffsgeschwindigkeit des externen Speichers bei der Evaluierung von Algorithmen zu variieren.

- (b) Der interne Speicher wird im RAM und der externe Speicher auf der Festplatte gespeichert. Diese Variante erlaubt also die Verarbeitung von Datenmengen, die nicht in den RAM des Computer passen; dadurch können die EM-Algorithmen dann auch für praktische Anwendungen genutzt werden.
2. *Grundlegende Datenstrukturen und Algorithmen.* Einige Datenstrukturen wie Stacks, Queues und Listen, sowie Algorithmen wie das externe Sortieren werden immer wieder in EM-Algorithmen verwendet und sollen daher von der XAVER-Plattform zur Verfügung gestellt werden.
 3. *Suffix-Arrays.* Zum Aufbau eines Suffix-Arrays existieren bereits verschiedene EM-Algorithmen, z.B. [DKMS05, CF02]. Neben der Implementierung dieser Algorithmen sollen auch Verbesserungsmöglichkeiten, insbesondere im Kontext gegebener Benchmark-Daten aus der Bioinformatik, untersucht werden.
 4. *Kürzeste Wege in großen Graphen.* Die meisten Algorithmen zur Lösung dieses Problems basieren auf der Idee der Breitensuche (BFS). Für dieses Konzept existieren bereits auf Externspeicher zugeschnittene Vorgehensweisen. Neben klassischen Varianten zur Suche von kürzesten Wege sollen hier auch neue Ideen für EM-Algorithmen entwickelt und miteinander experimentell verglichen werden.
 5. *Evaluierung der Algorithmen.* Die XAVER-Plattform erlaubt uns, die Performance von EM-Algorithmen experimentell zu evaluieren. Daher sollen verschiedene EM-Algorithmen zum Aufbau eines Suffix-Arrays und zur Berechnung von kürzesten Wegen in großen Graphen an Hand von praxisrelevanten Benchmark-Daten verglichen werden.

6 Teilnahmevoraussetzungen

Programmiererfahrung in C++ oder Java	(V)	Kenntnisse in C++	(W)
Algorithm Engineering oder	(M)	Übersetzerbau	(W)
Effiziente Algorithmen			

7 Minimalziele

1. Lauffähige XAVER-Plattform inklusive beider Simulationsmodi (externer Speicher als Datei auf Platte bzw. im RAM simuliert)
2. Korrekte Implementierung der Basisdatenstrukturen und -algorithmen wie externe Listen und externes Sortieren
3. Mindestens zwei implementierte Verfahren zum Aufbau eines Suffix-Arrays
4. Mindestens zwei implementierte Verfahren zur Suche von kürzesten Wegen
5. Vergleichsstudie zwischen verschiedenen Verfahren, und der Einflüsse der Parameter (Speicherzugriffszeit, Blockgröße,...)

8 Literatur

Algorithm Engineering

- [Mut05] P. Mutzel. Vorlesungsfolien zu *Algorithm Engineering*. <http://ls11-www.cs.uni-dortmund.de/lehre/WiSe0506/ae.jsp>, 2006.
- [Dem06] C. Demetrescu. 9th DIMACS Implementation Challenge: *Shortest Paths*. <http://www.dis.uniroma1.it/~challenge9/>, 2006.

Bioinformatik

- [JP04] N. C. Jones and P. A. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, 2004.

EM-Algorithmen

- [MSS035] U. Meyer, P. Sanders, and J. Sibeyn (Eds.). *Algorithms for Memory Hierarchies*. LNCS 2625, Springer, 2003.
- [DKMS05] R. Dementiev, J. Kärkäinen, J. Mehnert, and P. Sanders. Better external memory suffix array construction. *Proceedings of ALENEX '05*, 2005.
- [CF02] A. Crauser and P. Ferragina. A theoretical and experimental study on the construction of suffix arrays in external memory. *Algorithmica*, 32:1–35, 2002.
- [CGR96] B. V. Cherkassky, A. V. Goldberg, T. Radzik. Shortest Paths Algorithms: Theory And Experimental Evaluation. *Mathematical Programming Series A*, 73(2):129–174, 1996. Siehe auch: <http://citeseer.ist.psu.edu/cherkassky93shortest.html>

C++ Programmierung

- [DD05] H. M. Deitel and P. J. Deitel. *C++ How to program*. Prentice Hall, 2005.
- [DD03] B. Stroustrup. *The C++ Programming Language. Special Edition*. Addison-Wesley Professional, 2003.