Universität Stuttgart
Fakultät für Bau- und Umweltingenieurwissenschaften

Jiaquan Wang

Datenkompression für die Ergebnisse von Finite-Elemente-Simulationen

Motivation und Aufgabenstellung

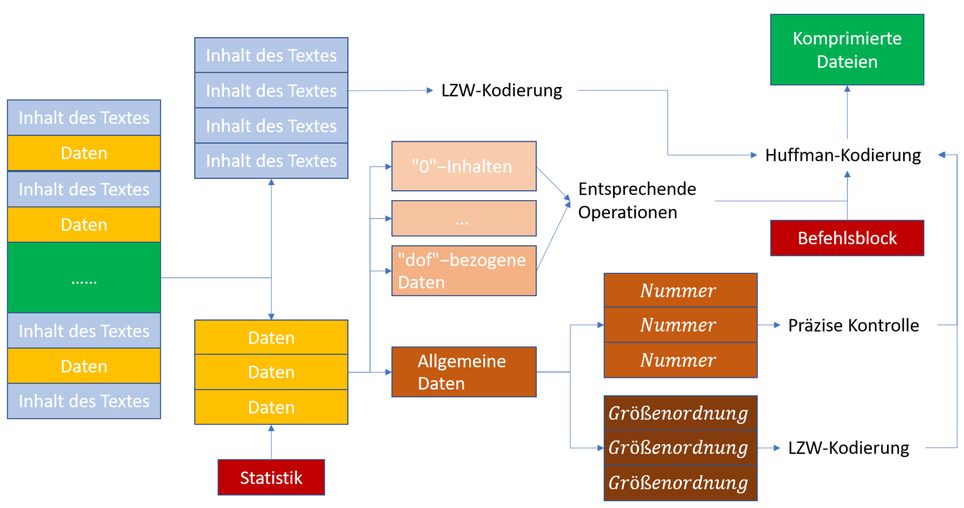
Finite-Elemente-Simulationen erzeugen oft große Datenmengen, deren Speicherung viel Speicherplatz erfordert. daher ist die Komprimierung der Ergebnisdateien von großem Interesse. In dieser Arbeit wird ein Komprimierungsalgorithmus vor allem für die Ergebnisdaten aus dem Finite-Elemente-Code NumPro implementiert, der sich auf einige der herkömmlichen Algorithmen stützt, wobei sowohl die Geschwindigkeit als auch die Wirksamkeit der Komprimierung berücksichtigt werden.

Verwandte Komprimierungsalgorithmen

- Der Varint–Algorithmus
- Der Huffman-Algorithmus
- Der Lempel-Ziv-Welch-Algorithmus

Ablauf Zusammenfassung

In dieser Arbeit besteht die Grundidee der NumPro-Dateikomprimierung darin, die Daten von Datei zunächst in verschiedene Blöcke zu klassifizieren, dann eine geeignete Komprimierungsstrategie entsprechend den Eigenschaften der verschiedenen Blöcke zu implementieren und schließlich alle Blöcke mit Hilfe des Huffmann-Algorithmus zu einer endgültigen Datei zusammenzuführen.



Im Gegensatz dazu besteht sein Dekomprimierungsverfahren darin, zunächst verschiedene Blöcke zu erhalten, indem sie mit dem Huffman-Algorithmus dekodiert werden, und dann den LZW-Algorithmus einzusetzen, um den entsprechenden Teil der Blöcke zu dekomprimieren. Dann wird die entsprechende Dekomprimierungsstrategie entsprechend den Befehlsblock abwechselnd auf die verschiedenen Blöcke angewandt, wobei zwischen der Ausgabe von Text und numerischen Zeichenfolgen abgewechselt wird, was zu einer wiederhergestellten Datei führt.

Betreuer

M. Sc. Maximilian Schilling,
M. Sc. Tobias Willmann,

https://www.ibb.uni-stuttgart.de

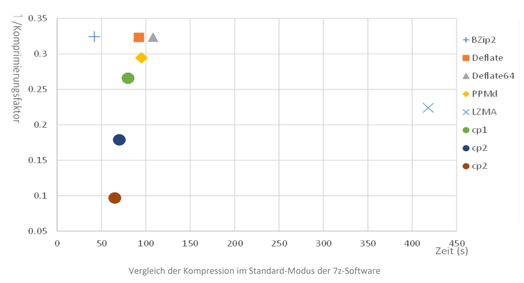
Numerisches Beispiel

Diese Arbeit konzentriert sich auf den Vergleich der Komprimierung mit den aktuellen herkömmlichen Komprimierungsalgorithmen wie BZip2, Deflate, PPMd, LZMA, die mit dem Standard- und dem Extrem-Modus der 7z-Software getestet warden, wobei hauptsächlich die Komprimierungsfaktors und die erforderliche Komprimierungszeiten verglichen werden. Die Größe der getesteten Datei “Kragarm_nln_dirich_fine.post.res” beträgt 566.394.425 Byte.

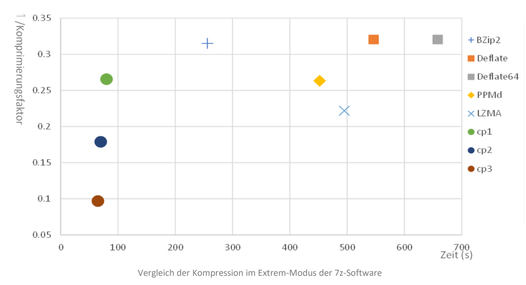
cp1 cp2 cp3

verlustfreie			6 Stellen			3 Stellen		
Größe (Byte)	Zeit (s)	Faktor	Größe (Byte)	Zeit (s)	Faktor	Größe (Byte)	Zeit (s)	Faktor
150.439.697	80	3,76	101.305.139	70	5,59	54.842.983	65	10,32

Ergebnisse für den Algorithmus aus dieser Arbeit



Vergleich der Kompression im Standard-Modus der 7z-Software



Vergleich der Kompression im Extrem-Modus der 7z-Software

Literatur

Indrasiri, K.; Kuruppu, D.: gRPC: Up and Running: Building Cloud Native Applications with Go and Java for Docker and Kubernetes. O’Reilly Media, 2020.

Levine, J.R.: Programming for Graphics Files: In C and C++. Wiley, 1994

Manz, O.: Gut gepackt – Kein Bit zu viel: Kompression digitaler Daten verständlich erklärt. Springer Fachmedien Wiesbaden, 2020 (essentials).

Stallings, W.: Computer Organization and Architecture: Designing for Performance. Pearson, 2019.

Thole, C. A.: Compression of LS-DYNA simulation results using FEMzip.

Baustatik und Baudynamik