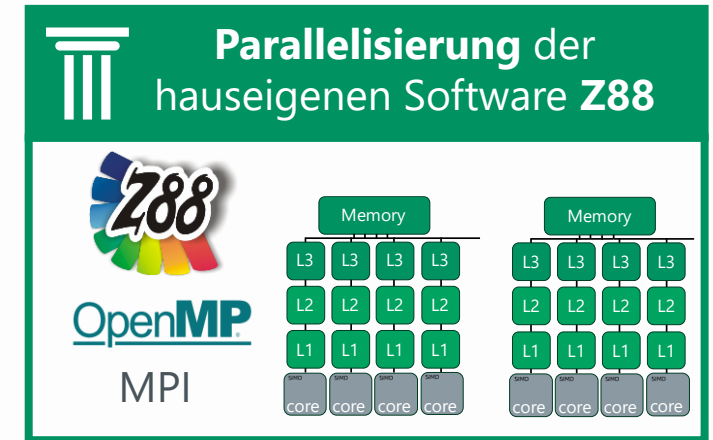


Gate2HPC

Parallel Computing für KMU:

Die Erweiterung von Z88 im Gate2HPC-Projekt



Peter Grohmann

25. Bayreuther 3D-Konstrukteurstag

11.09.2024

<https://creativecommons.org/licenses/by/4.0/>



Diese Präsentation steht, soweit nicht anders gekennzeichnet, unter der Lizenz Creative Commons Namensnennung 4.0 International (CC BY 4.0). Für gesondert gekennzeichnete Abbildungen und sonstige Inhalte Dritter gelten die jeweils angegebenen oder geltenden Rechte- und Lizenzbedingungen



Lehrstuhl für
Konstruktionslehre und CAD
Prof. Dr.-Ing. Stephan Tremmel

Bayerisches Staatsministerium für
Wissenschaft und Kunst



Kofinanziert von der
Europäischen Union

Gate2HPC

- **Gefördert durch:** Europäischer Fonds für regionale Entwicklung (EFRE)
- **Schwerpunkt:** Digitalisierung
- **Förderbereich:** 1 – Innovation und Wettbewerbsfähigkeit
- **Fördermaßnahme:** 2 – Technologietransfer von Hochschulen in KMU
- **Projektlaufzeit:** Juni 23 – Mai 27



Nutzen von Methoden der Künstlichen Intelligenz



Bereitstellung von vorgefertigten **Notebooks**



Bild
(z. B. Anomalie-erkennung)



Sprache
(z. B. Large-Language-Models)



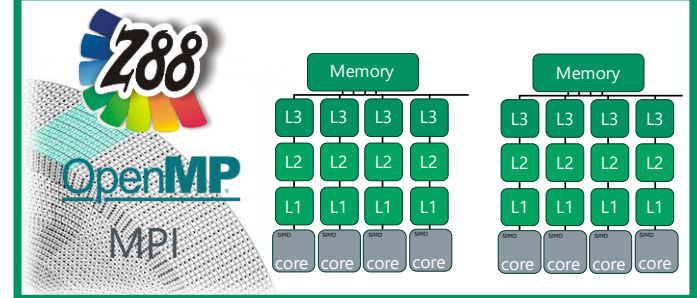
Logische Schlussfolgerungen
(z. B. Datenanalyse)



Ansteuerung von FE-Programmen



Parallelisierung der hauseigenen Software Z88



Konverter



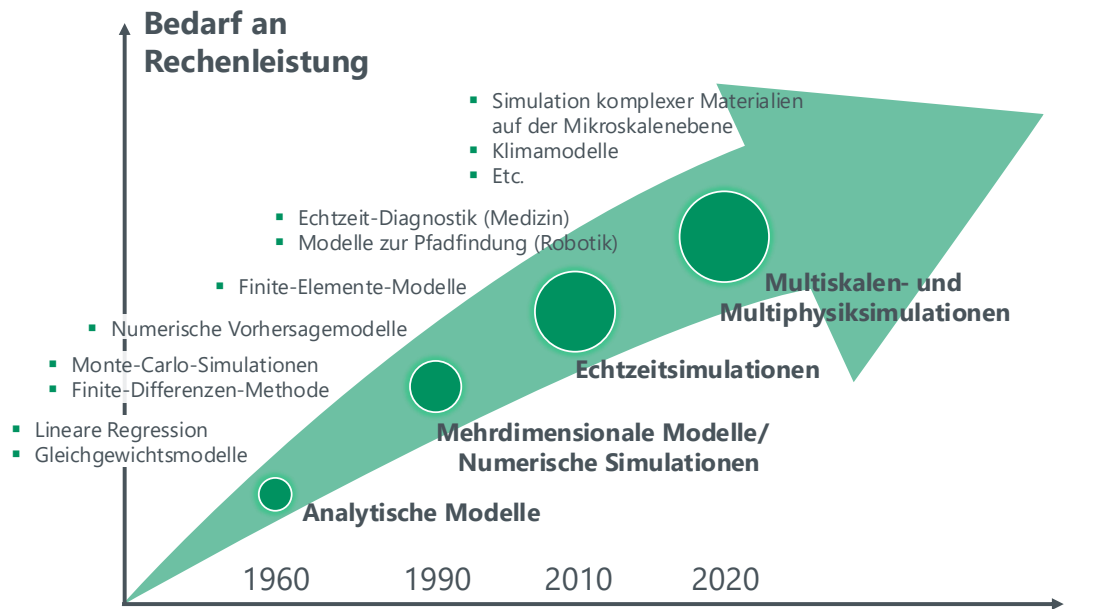
Lehrstuhl für
Konstruktionslehre und CAD
Prof. Dr.-Ing. Stephan Tremmel

Bayerisches Staatsministerium für
Wissenschaft und Kunst



Kofinanziert von der
Europäischen Union

Problemstellung



Steigender Bedarf an Rechenleistung für Simulationen und ML/KI-Anwendungen

Lösung durch...

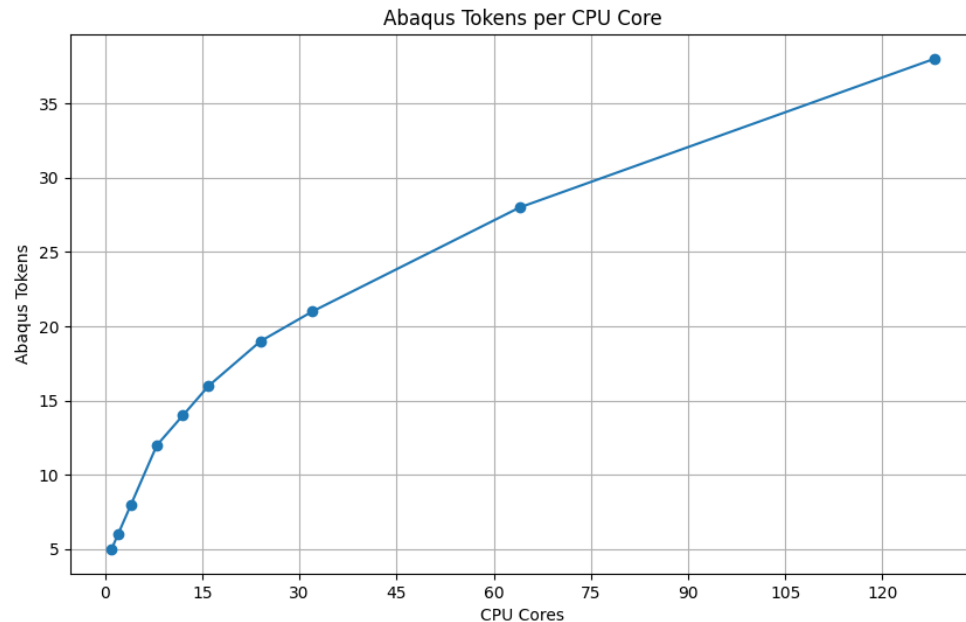
...den Einsatz von HPC

HPC-Hürden

- Software
 - Lizenzgebühren
 - Kostenpunkt Tokens
- Hardwarekosten
 - Anschaffungskosten
 - Wartung
- Niedrige Usability
- Datenschutz und Sicherheit
- Schwankender Anwendungsbedarf
- Fehlendes Know-How
- Etc.

Token-basiertes Lizenzierungssystem:

- Anzahl der benötigten Abaqus-Tokens richtet sich nach Anzahl der verwendeten CPU-Kerne. Je mehr für eine Berechnung genutzt werden, desto mehr Tokens werden benötigt.



$$\text{Benötigte Tokens} = 5 \times N^{0,422}$$

N : Anzahl der genutzten CPU-Kerne

HPC-Hürden

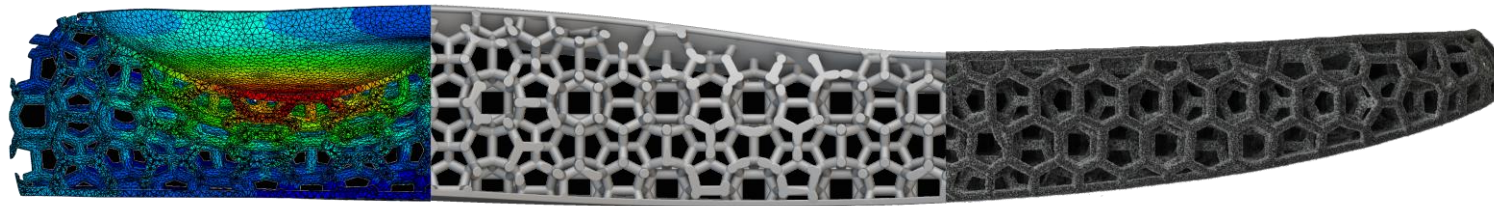
- Software
 - Lizenzgebühren
 - Kostenpunkt Tokens
- Hardwarekosten
 - Anschaffungskosten
 - Wartung
- Niedrige Usability
- Datenschutz und Sicherheit
- Schwankender Anwendungsbedarf
- Fehlendes Know-How
- Etc.

Quelle: <https://deviceanalytics.com/abaqus-token-calculator/>



MonoMat

Kaskadierter Monomaterialansatz
zur automatisierten ressourceneffizienten Fertigung digitalisierter,
personalisierter Anwendungen in Medizin, Sport und Lifestyle



Bundesministerium
für Wirtschaft
und Klimaschutz



WEBER additive



AM Polymers



UNIVERSITÄT
BAYREUTH



Neue Materialien
Bayreuth

HPC-Hürden

- Software
 - Lizenzgebühren
 - Kostenpunkt Tokens
- Hardwarekosten
 - Anschaffungskosten
 - Wartung
- Niedrige Usability
- Datenschutz und Sicherheit
- Schwankender Anwendungsbedarf
- Fehlendes Know-How
- Etc.



Lehrstuhl für
Konstruktionslehre und CAD
Prof. Dr.-Ing. Stephan Tremmel

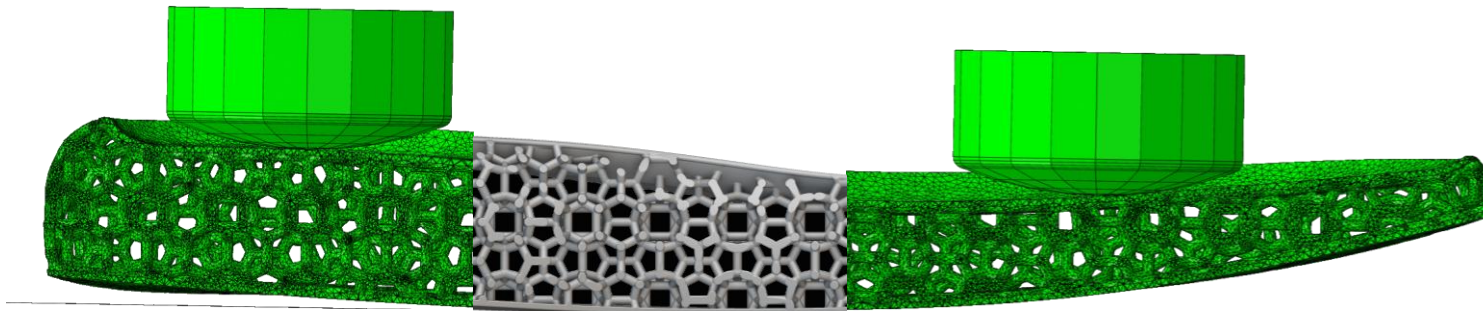
Bayerisches Staatsministerium für
Wissenschaft und Kunst



Kofinanziert von der
Europäischen Union

MonoMat | Kaskadierter Monomaterialansatz

zur automatisierten ressourceneffizienten Fertigung digitalisierter, personalisierter Anwendungen in Medizin, Sport und Lifestyle



Rechenzeit einer Rechnung: ≈ 72 Stunden

Verwendete Hardware: Intel Core i9-10980XE CPU, 128 GB DDR4

Erforderliche Abaqus-Tokens je Rechnung: ≈ 16 Tokens

Quelle: Thomas Peter Schütt (Modellaufbau und Abaqus-Rechnungen)

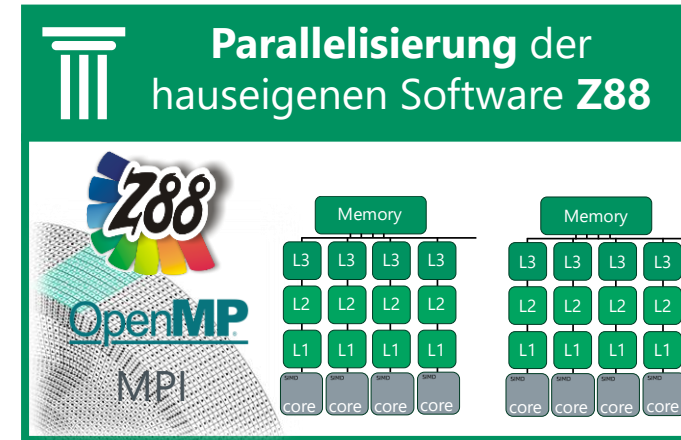
HPC-Hürden

- Software
 - Lizenzgebühren
 - Kostenpunkt Tokens
- Hardwarekosten
 - Anschaffungskosten
 - Wartung
- Niedrige Usability
- Datenschutz und Sicherheit
- Schwankender Anwendungsbedarf
- Fehlendes Know-How
- Etc.

Problemstellung

- Lehrstuhleigenes lizenzkostenfreies FE-Programmsystem Z88 nicht HPC-fähig.
 - Hindernis für die effiziente Berechnung komplexer FE-Simulationen.
 - Hindernis für die Weiterentwicklung der Forschungssoftware.

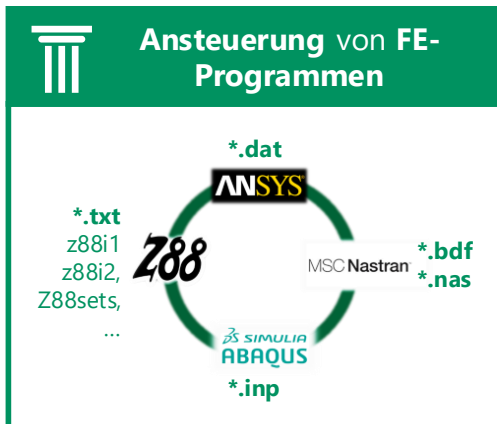
Lösung durch...



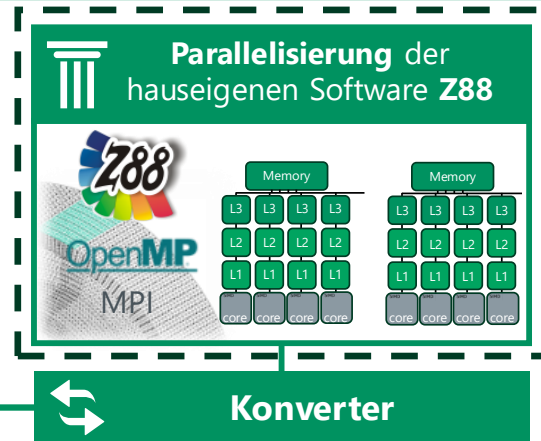
Bereitstellung eines lizenzkostenfreien HPC-fähigen FE-Programmiersystems

Gate2HPC

- **Gefördert durch:** Europäischer Fonds für regionale Entwicklung (EFRE)
- **Schwerpunkt:** Digitalisierung
- **Förderbereich:** 1 – Innovation und Wettbewerbsfähigkeit
- **Fördermaßnahme:** 2 – Technologietransfer von Hochschulen in KMU
- **Projektlaufzeit:** Juni 23 – Mai 27



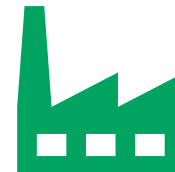
**Schaffung von
Interoperabilität**



**Leistungsfähiges
Lizenzkostenfreies
FE-Programmsystem**



**Förderung der Zusammenarbeit und des
Wissensaustausches**



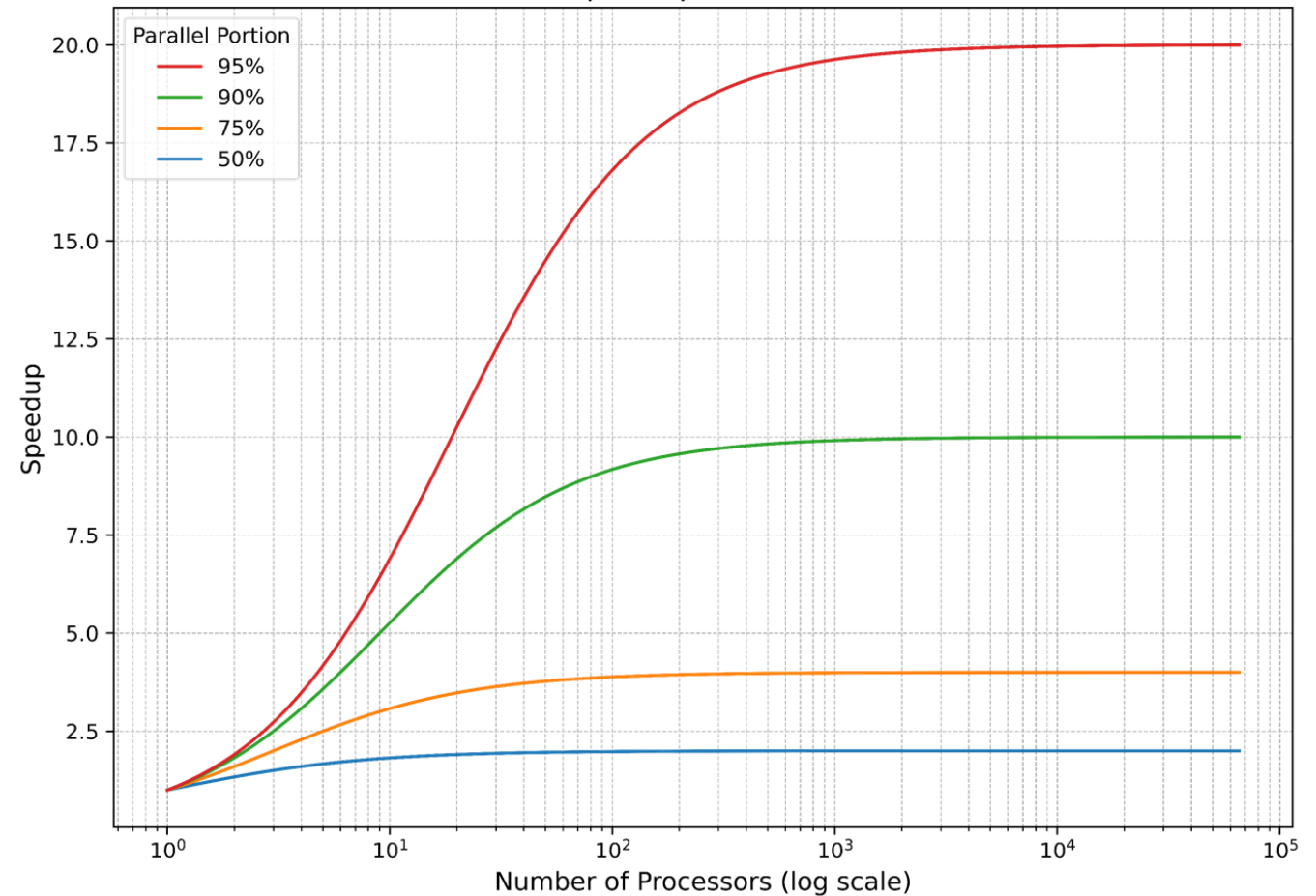
**Stärkung der Wettbewerbsfähigkeit der
teilnehmenden Unternehmen
insbesondere von KMU.**



$$S_{Amdahl} = \frac{1}{(1 - P) + \frac{P}{N}}$$

- S : Speedup
- P : Proportion of parallel code
- N : Number of processors

Amdahl's Law - Speedup vs Number of Processors



Overhead

Nachrichtengekoppelte Parallelität

Gemeinsames Arbeiten an Daten,
Datenaustausch, Kommunikation

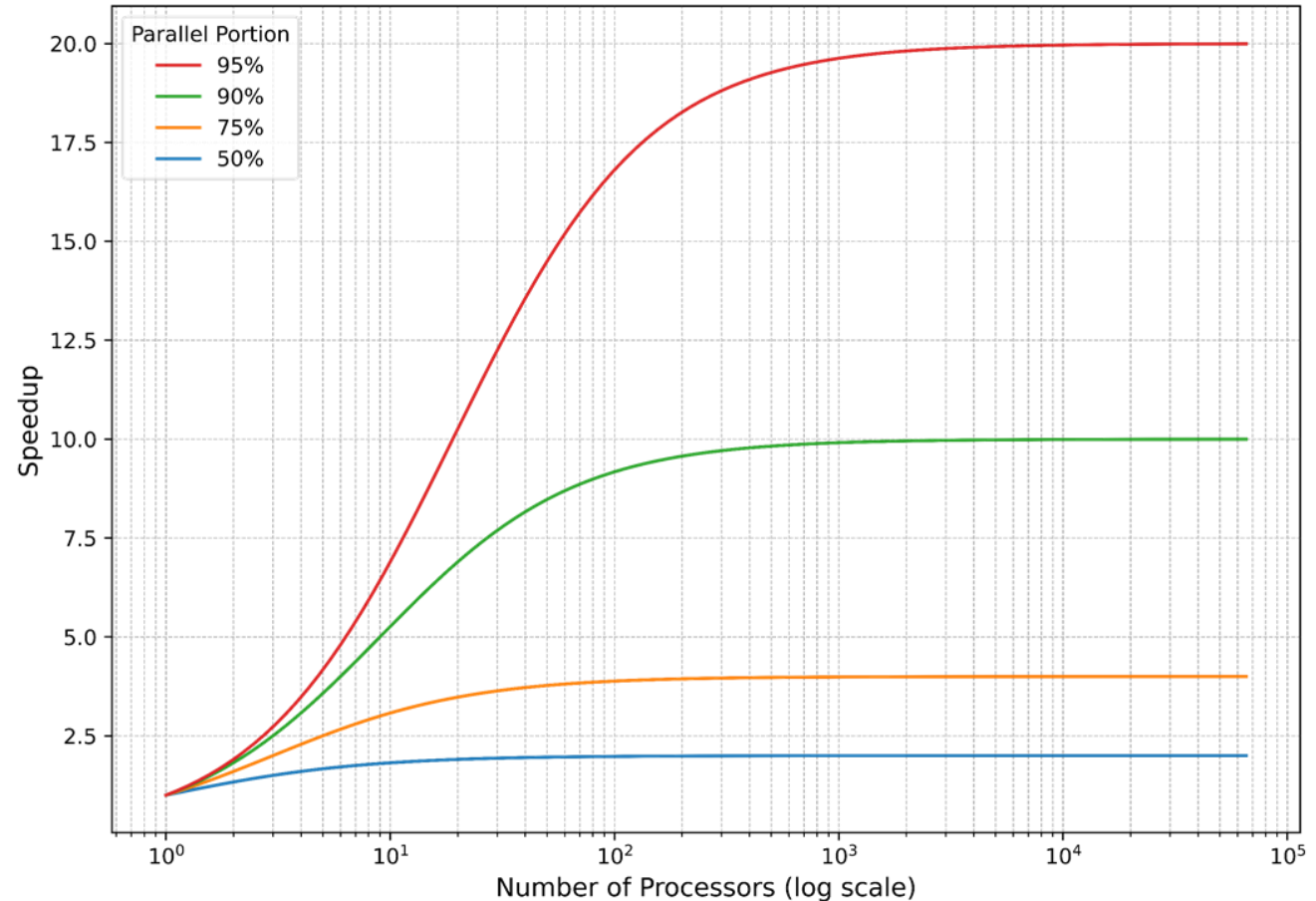
Geteilter Speicherparallelismus

Schreiben gemeinsamer Daten
in den gemeinsamen Speicher

Problem der Synchronisation

Prozesse, die auf langsamere Prozesse
warten müssen, um z.B. zu vermeiden dass
der langsamere Prozess falsche Daten erhält

Amdahl's Law - Speedup vs Number of Processors



Overhead

Nachrichtengekoppelte Parallelität

Gemeinsames Arbeiten an Daten,
Datenaustausch, Kommunikation

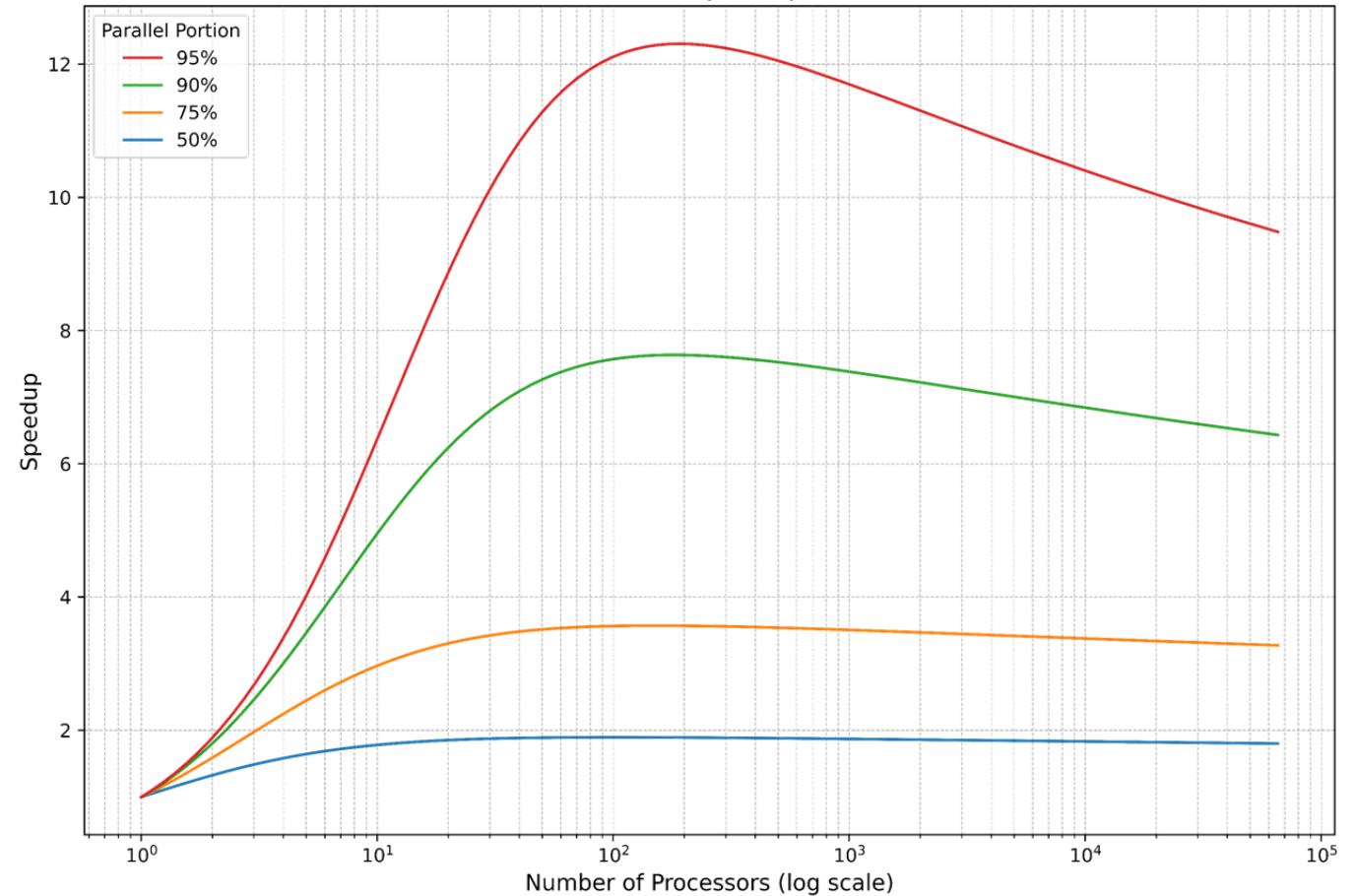
Geteilter Speicherparallelismus

Schreiben gemeinsamer Daten
in den gemeinsamen Speicher

Problem der Synchronisation

Prozesse, die auf langsamere Prozesse
warten müssen, um z.B. zu vermeiden dass
der langsamere Prozess falsche Daten erhält

Amdahl's Law with Overhead* - Speedup vs Number of Processors

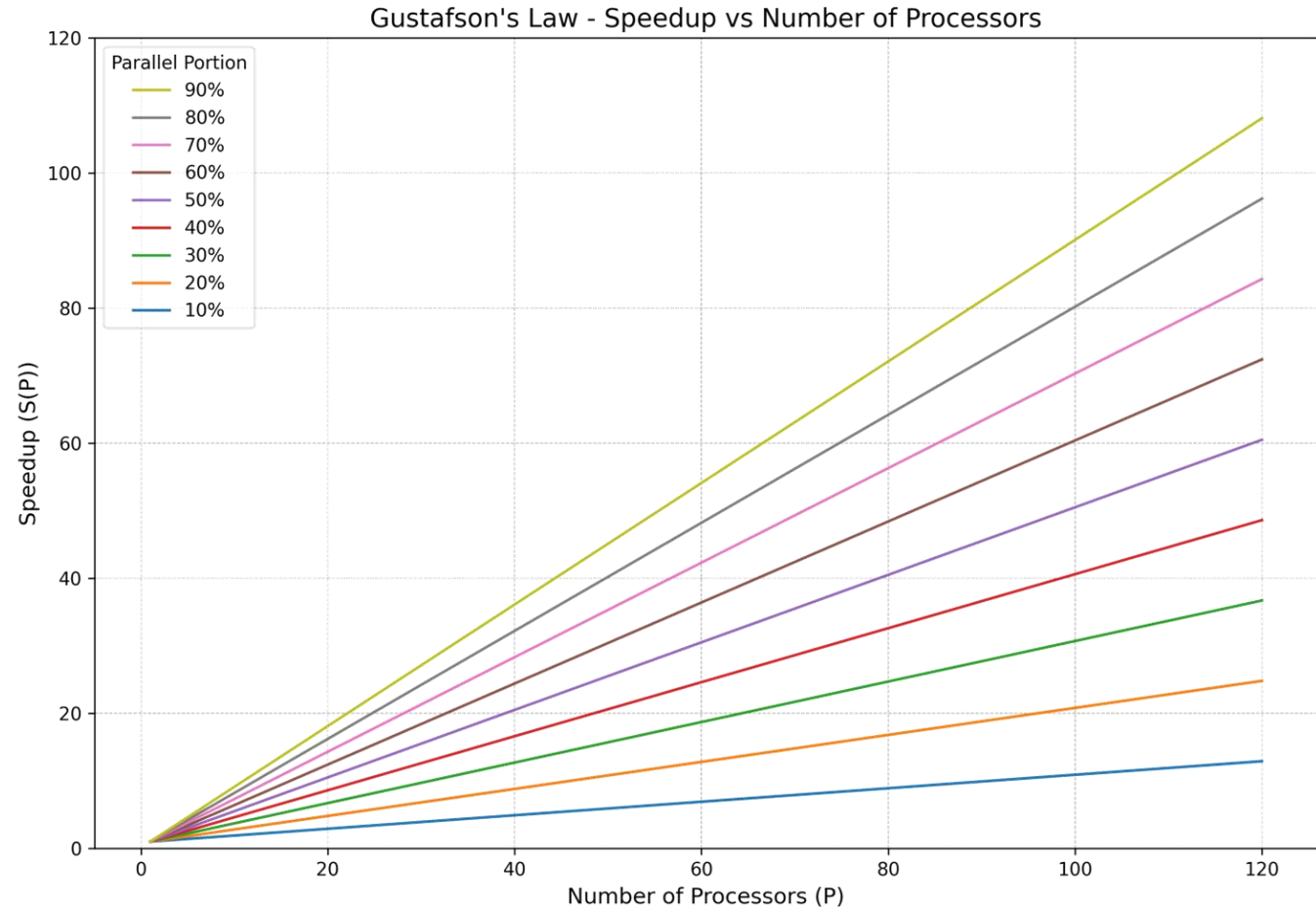


*Annahme eines logarithmisch ansteigenden Overheads



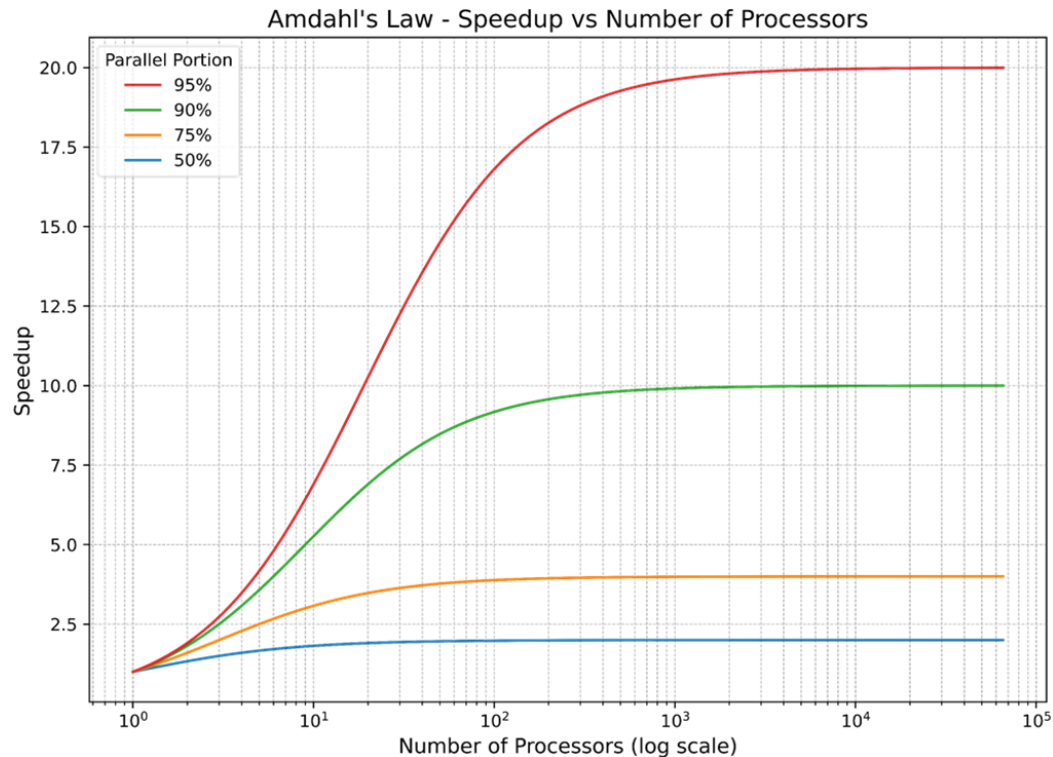
$$S_{Gustafson} = (1 - p) + p \times N$$

- S : Speedup
- p : Proportion of parallel code
- N : Number of processors

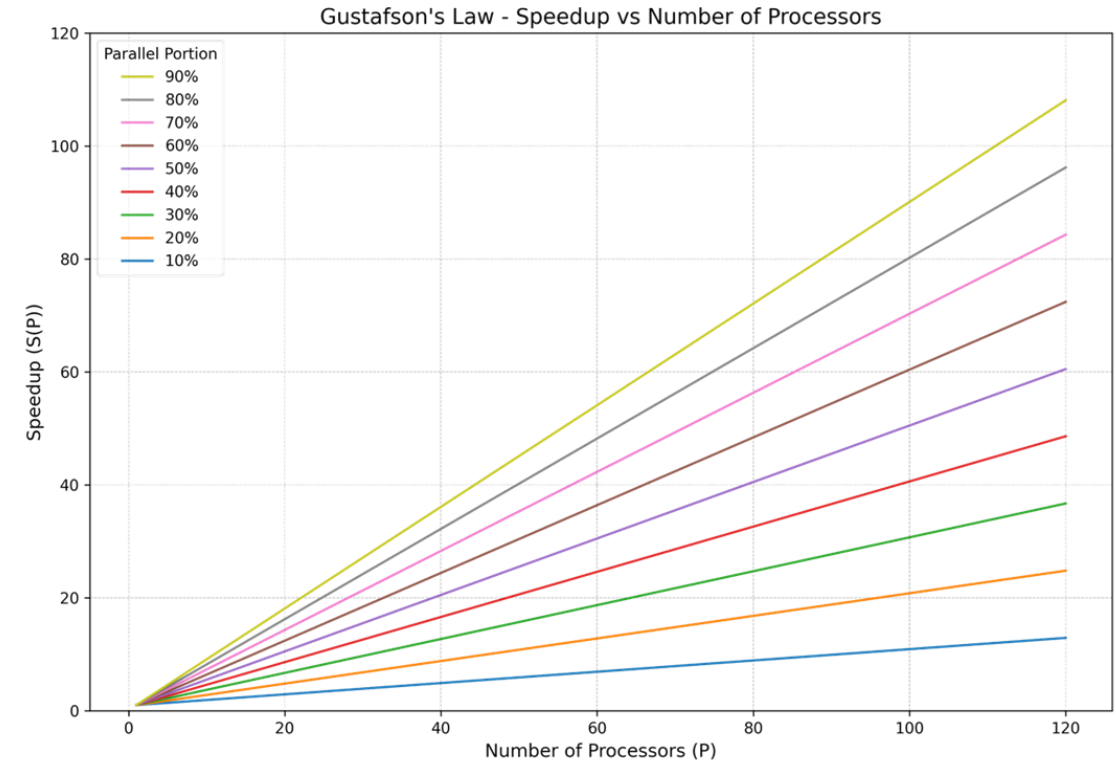


Grundlagen der Parallelisierung

Grenzen der Parallelisierung



Der Geschwindigkeitsgewinn durch Parallelisierung wird durch den festen sequentiellen Anteil begrenzt, der nicht parallelisierbar ist, wodurch der Speedup bei einer höheren Anzahl von Prozessoren abnimmt.



Der Geschwindigkeitsgewinn durch Parallelisierung kann nahezu linear mit der Anzahl der Prozessoren wachsen, da bei wachsender Problemgröße der parallele Anteil dominiert und der sequentielle Anteil relativ klein bleibt.

High Performance Computing

Ziel: Leistungsfähigkeit einer einzelnen Anwendung oder Simulation zu maximieren. Hierbei treten häufig starke Abhängigkeiten zwischen den Berechnungen, die durchgeführt werden müssen auf.

Probleme die mehr oder weniger viele Abhängigkeiten beinhalten.

Beispiel: Lösen von Differenzialgleichungen (Fluidmechanik, Finite-Elemente-Methode, Wettermodelle,...).



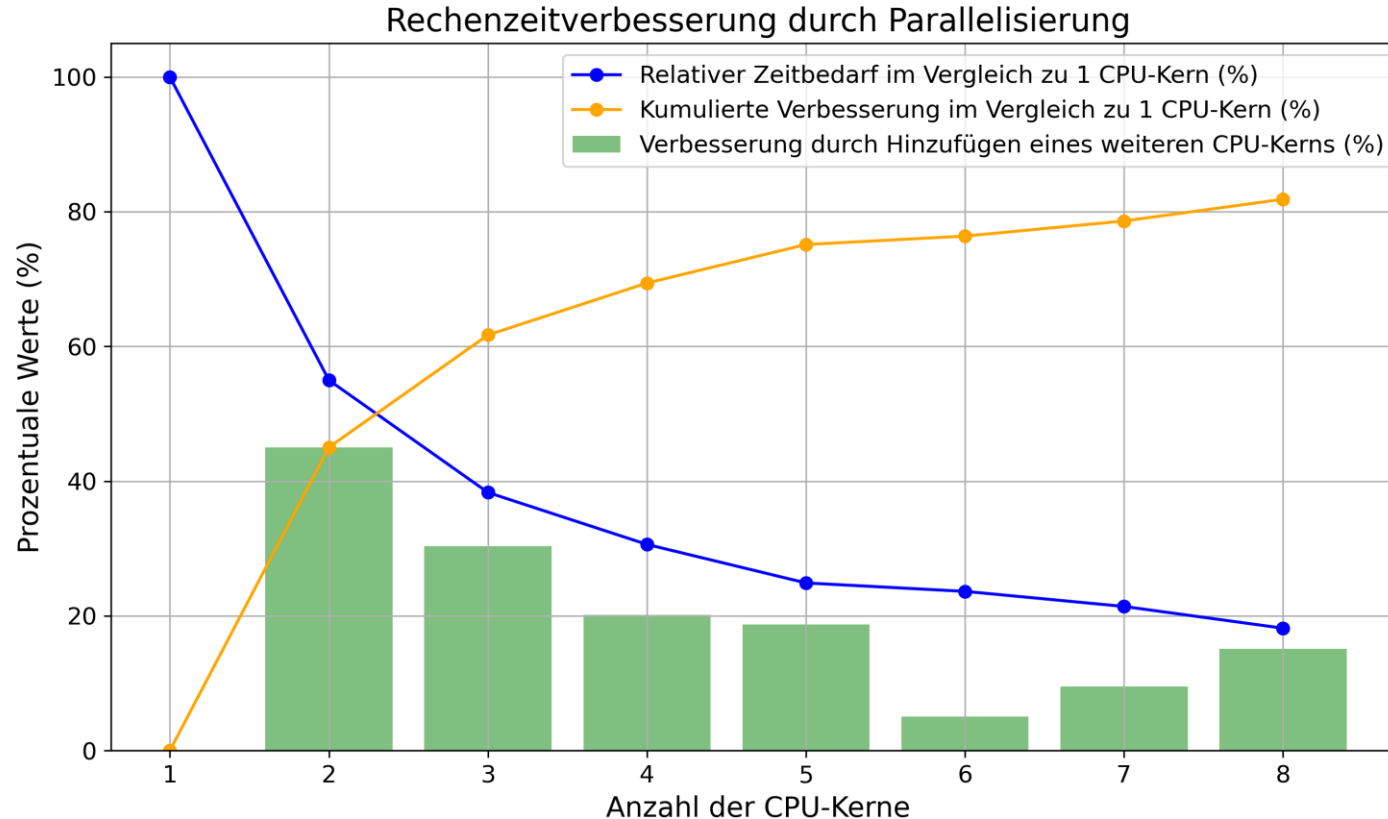
High-Throughput Computing

Ziel: Maximierung des Durchsatzes von mehreren unabhängig laufenden Aufgaben oder Prozessen innerhalb eines bestimmten Zeitrahmens.

Probleme die „peinlich bzw. inhärent“ parallel sind

Beispiel: Datenanalyse, Textverarbeitung, Bildanalyse, Brute-Force Password knacken





Hintergrundinformationen zum FE-Modell:

- Lineare FEA (3D)
- Quadratische Tetraederelemente
- Anzahl Elemente: 47.736
- Freiheitsgrade 227.232

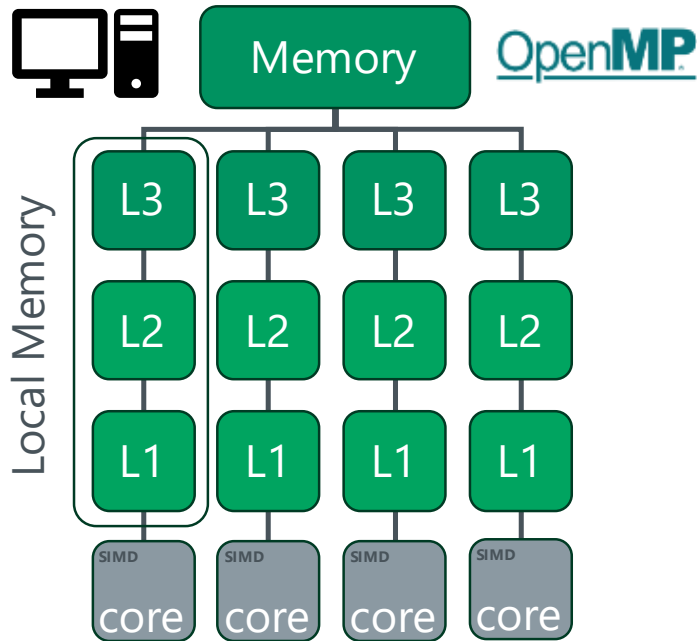
Lösen des Gleichungssystems durch
PARDISO-Gleichungslöser

Zusätzliche CPU-Kerne
beschleunigen die Berechnung, aber
es gibt auch eine vom Modell
abhängige **Effektivitätsgrenze**

Nach Z88-Workshopunterlagen



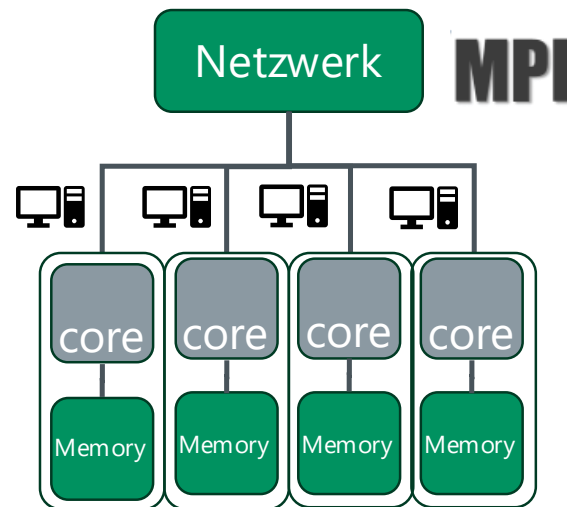
Shared-Memory- Programmierung



Mehrere Prozesse nutzen
denselben Speicherbereich

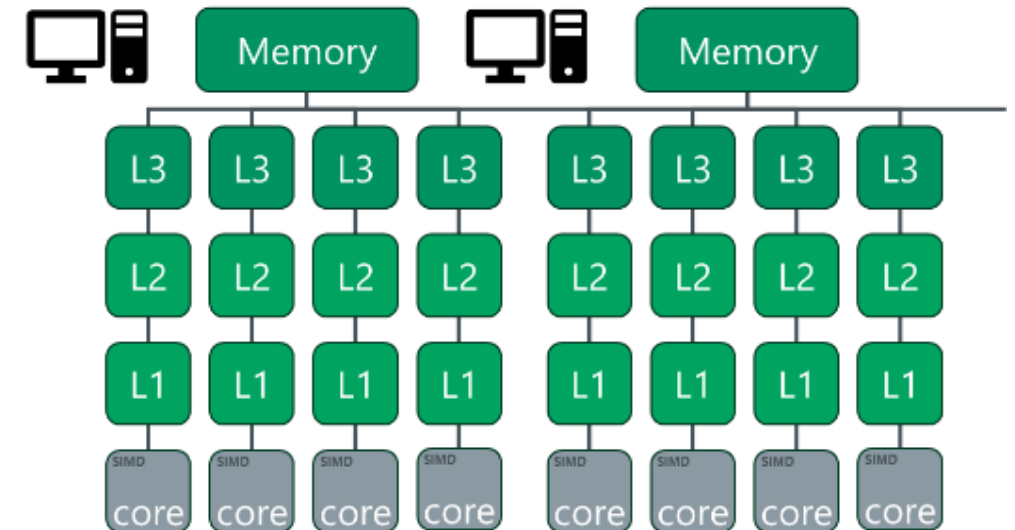
Nach [1]

Distributed-Memory- Programmierung



Jeder Prozess hat seinen
eigenen Speicherbereich

Hybrid-Ansatz (OpenMP + **MPI**)



Threads teilen sich Speicher innerhalb
eines Knotens, während Prozesse eigenen
Speicher zwischen Knoten nutzen

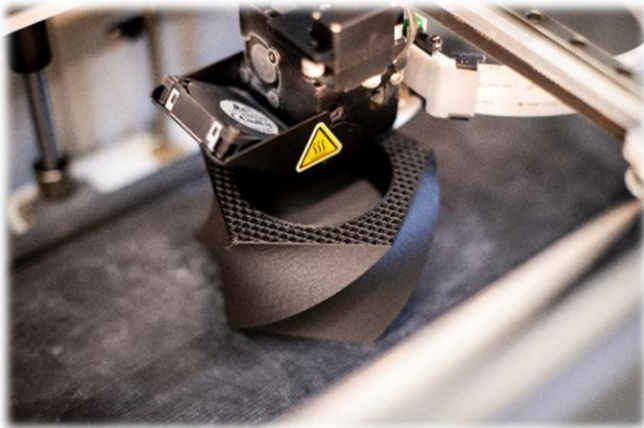




Weiterführende Informationen etwa zu
Lehrveranstaltungen, aktuellen
**Projekten, Studien- und
Abschlussarbeiten** oder
Industrieraufträgen:

www.lscad.de

konstruktionslehre.cad@uni-bayreuth.de



Wir freuen uns über Ihr Interesse!



Vielen Dank für ihre Aufmerksamkeit!



- [1] Bader, R., Hager, G., Weinberg, V., & Wittmann, M. (2023). Advanced OpenMP Programming. LRZ; RRZE; NHR@FAU.
- [2] Lago, R. (2022). Intel® oneAPI Base and HPC Toolkits: Diagnostics & Profiling Tools.
- [3] Shadrina, A. (2023, June). Parallelizing heterogenous applications with Intel® OpenMP and OpenMP offloading: Advanced Topics. Workshop presented at LRZ. Email: alina.shadrina@intel.com
- [4] Coles, J. (2021, November 3). The Roofline Model. Presented at the HPC Code Optimisation Workshop.
- [5] Nguyen, Duc Thai. *Parallel-Vector Equation Solvers for Finite Element Engineering Applications*. Boston, MA: Springer US, 2002. <https://doi.org/10.1007/978-1-4615-1337-7>.
- [6] Rauber, Thomas, und Gudula Rünger. *Parallele Programmierung*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. <https://doi.org/10.1007/978-3-642-13604-7>.



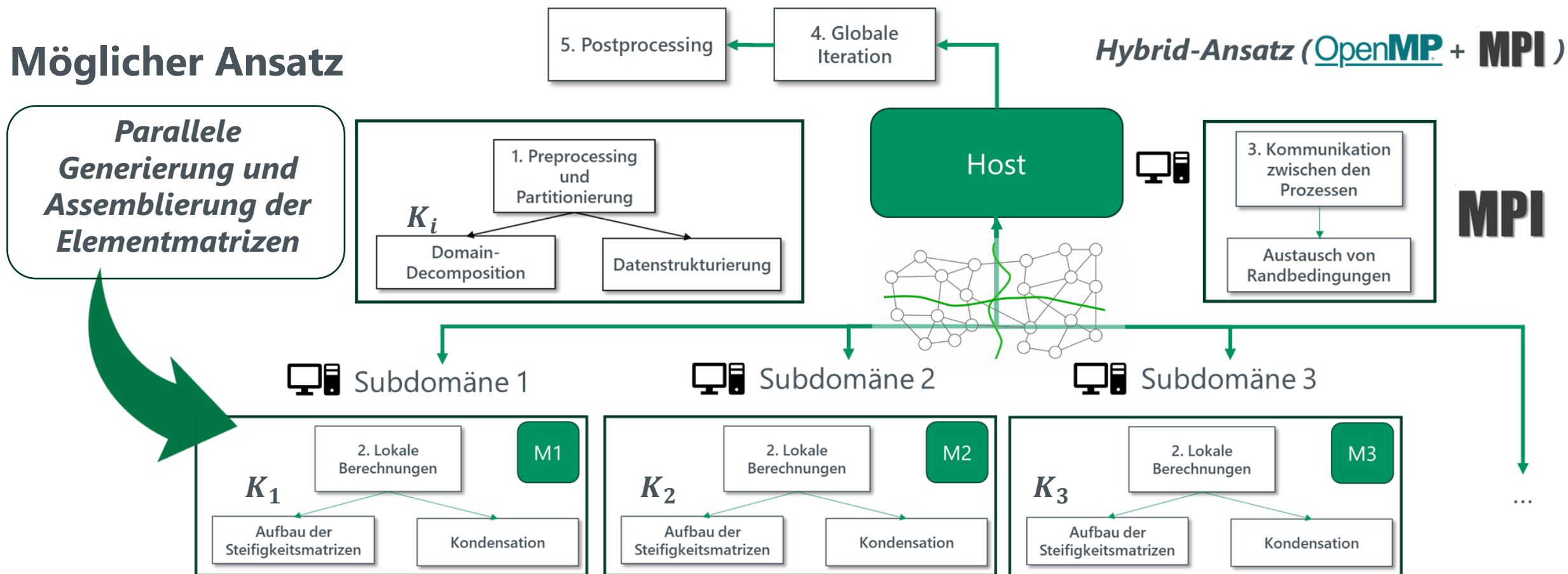
Anhang



Grundlagen der Parallelisierung

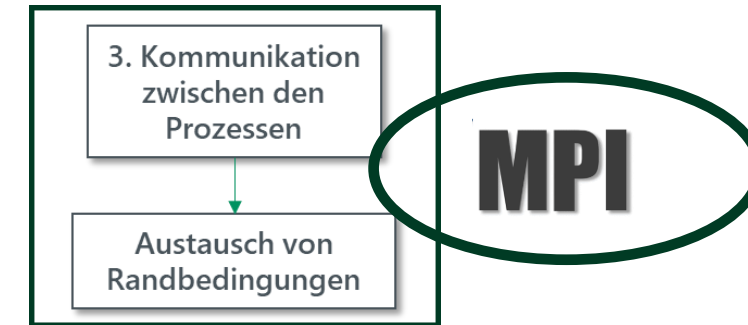
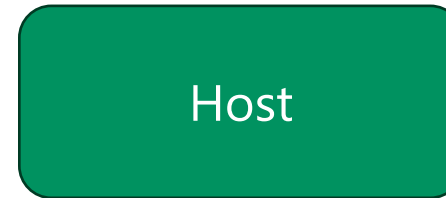
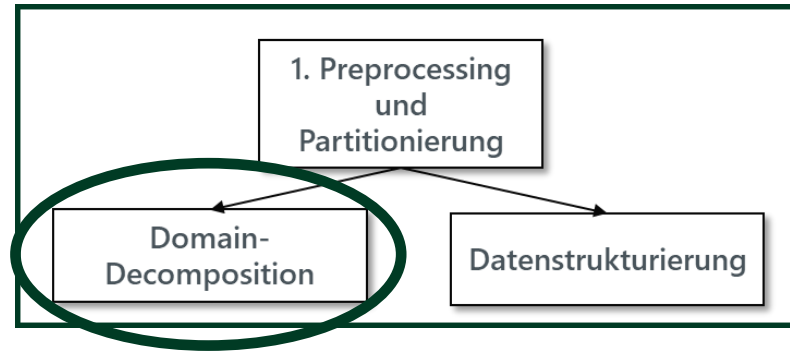
Aufteilen der Arbeit und Daten in der FEM

Möglicher Ansatz

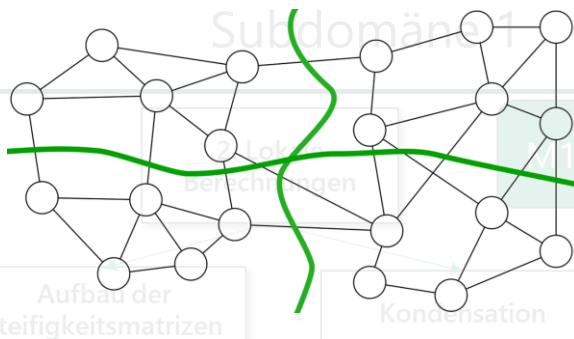


Grundsätze der Optimierung

Domain-Decomposition Methoden

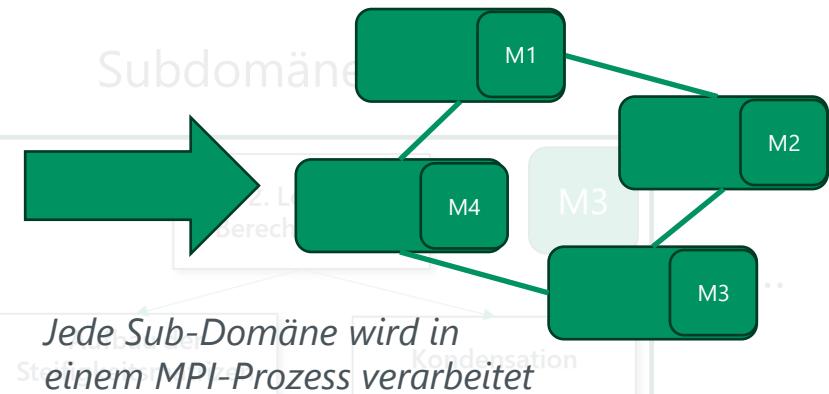


Netzpartitionierung



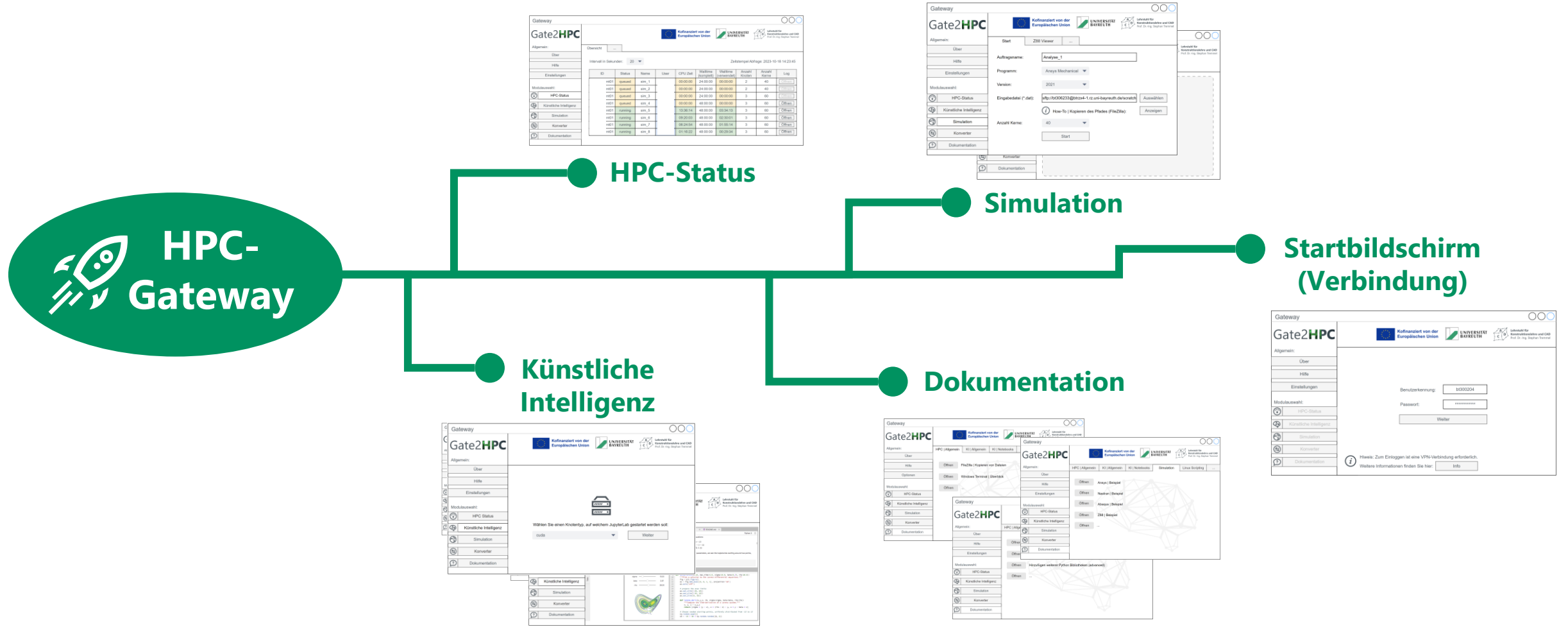
Ziele:

- Ähnliche Arbeitsbelastung in jeder Unterdomäne
- Minimierung der maximalen Anzahl von Nachbarschaftsverbindungen zwischen Teilgebieten
- Minimierung der Gesamtzahl benachbarter Sub-Domänen jeder Sub-Domäne



Das Projekt Gate2HPC

Gateway



**Lehrstuhl für
Konstruktionslehre und CAD**
Prof. Dr.-Ing. Stephan Tremmel

Bayerisches Staatsministerium für
Wissenschaft und Kunst



OBERFRANKEN
STIFTUNG



Kofinanziert von der Europäischen Union