iPUG: Accelerating Breadth-First Graph Traversals Using Manycore Graphcore IPUs

Seminar: Supercomputer: Forschung und Innovation

Lukas Schulte

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2022-01-08

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

informatik
die zukunft

Gliederung (Agenda)

Einleitung/Motivation

- Moores Law: Exponentielles Wachstum der Transistordichte
- sehr lange zutreffend
- CPUs immer schneller
- warten auf mehr Rechenleistung
- In den letzten Jahren nicht mehr richtig zutreffend
- *new golden age for computer architecture*
- Lösung: spezialisierte Hardware

Aufbau moderner CPUs

- I/O
- off-die Memory
- Prozessorkerne
- Verbindungsnetz
- Caches

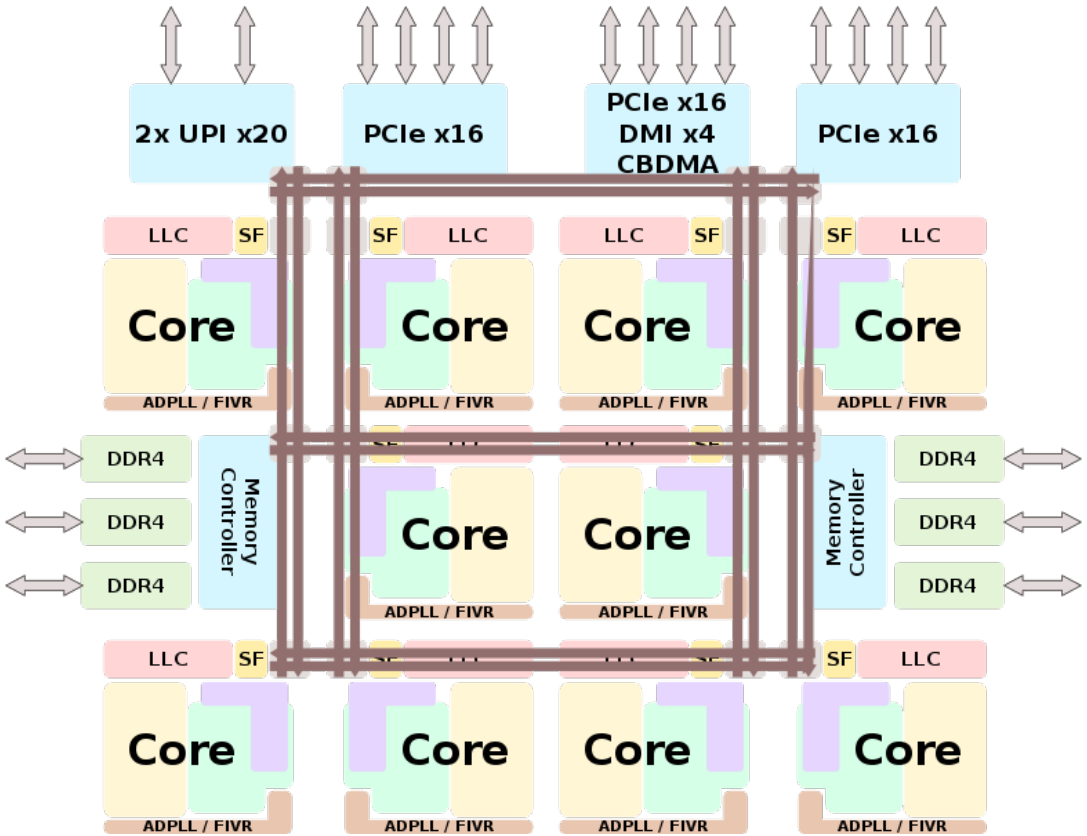


Abbildung: Quelle: [Bila]

Aufbau eines Prozessorkerns

- ALU
  - führt Berechnungen durch
- Memory/Caches
  - Implementiert Cache Struktur
  - Zugriff auf Hauptspeicher
  - L1$, L2$, L3$
- Control
  - Front End
  - synchronisation
  - Verbindungen

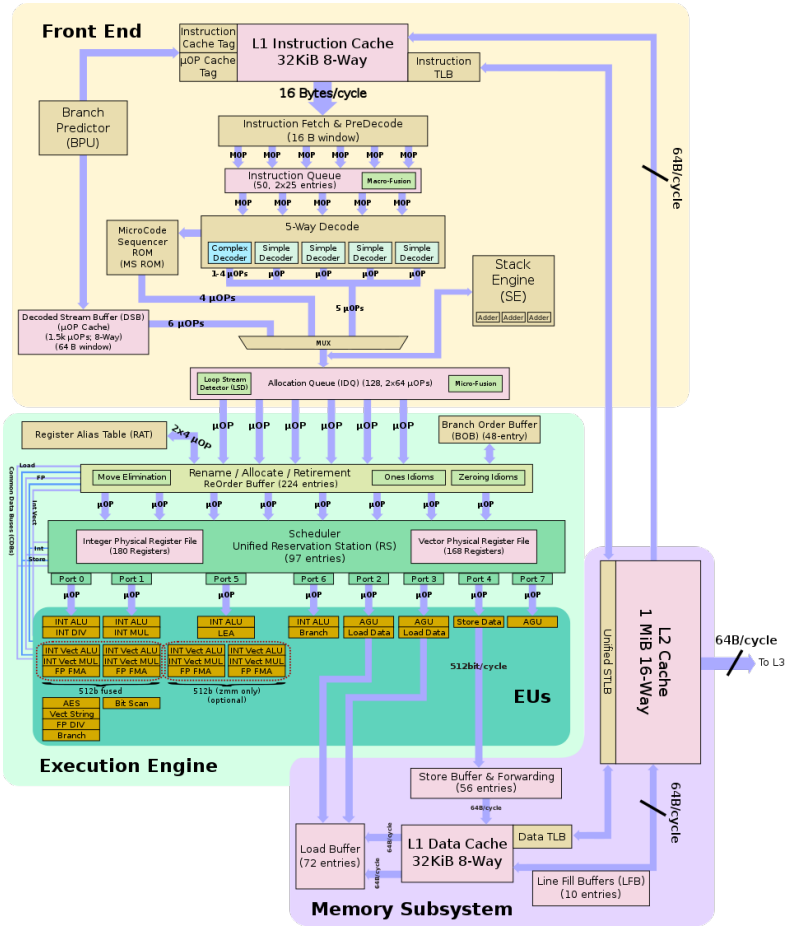


Abbildung: Quelle: [Bila]

GPUs

- Entworfen für Grafikanwendungen
- Seit Anfang 2000: Shader Modell statt fixed function Pipeline
- Ein Shader für alle Vertices, Pixel, etc.
- SIMD Ansatz, heute eher SPMT
- Sehr viele, lineare Daten
- Ziel: maximaler Durchsatz und lineare Abarbeitung
- Hardware: Maximieren von ALUs auf Kosten von Kontrollstrukturen

Graphcore Colossos IPU

- IPU = Intelligence Processing Unit
- Anwendung: *Machine Intelligence*
- => Alles im Bereich ML/KI/Graphalgorithmen
- Co Prozessor
- PCIe Steckkarte und IPU Pod

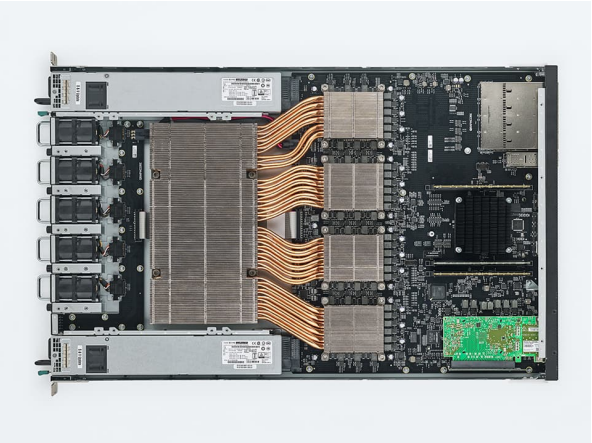


Abbildung: Links: IPU Pod, Rechts: PCIe Karte, Quelle: [Bilb], [Bilc]

Designgrundlagen

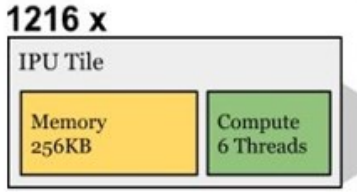
Eigenschaften von MI:

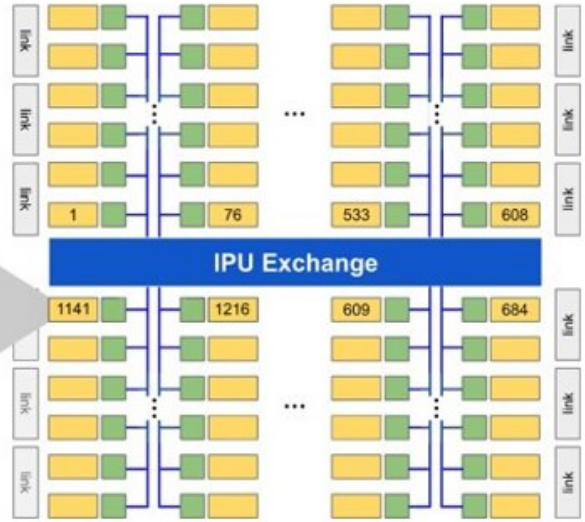
- Effizient zu parallelisieren
  - MIMD Ansatz mit sehr vielen Kernen
- Sehr geringe Präzision
  - schmale ALUs
  - Integer statt Float
- Statische Struktur von Graphen
  - Partitionierung und Scheduling statisch
  - Beides wird durch den Compiler gelöst
- dünn besetzte Matrizen

Hardware: Speicher und Verbindungsnetzwerk auf Kosten von ALUs (Präzision) und Kontrollstrukturen

Topologie

- Jedes Tile:
  - 256KiB RAM
  - 6 Taktzyklen Zugriffszeit
  - 6 Threads
- 4 Tiles pro Island
- 19 Islands eine Spalte
- 16 Spalten
- =>1216 Tiles/7296 Threads





Topologie

- 45 TB/s Speicherbandbreite
- IPU-Exchange: 8TB/s all to all
- IPU-Link: 320GB/s chip to chip
- IPU-Host: 64GB/s bidirectional

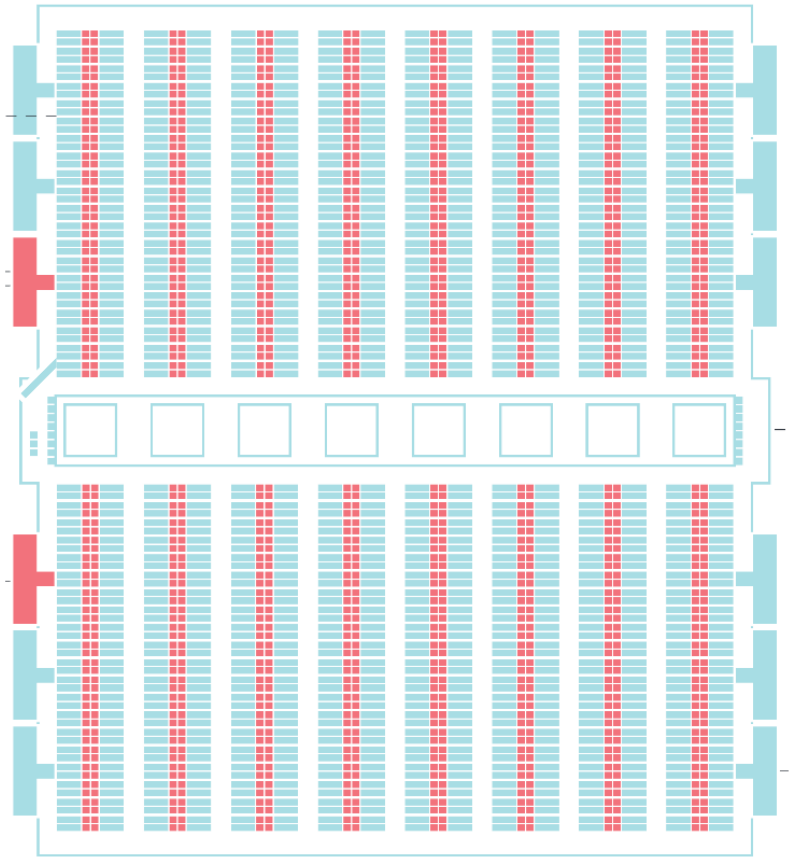


Abbildung: Quelle: [Bilb]

Besonderheiten

- TMT statt SMT
  - Sequenziell statt Parallel
  - jeder Thread ein Takt
  - Bestfall: Keine Speicherlatenzen
- BSP: Bulk Synchronous Parallel
  - entweder Kommunikation oder Berechnung
  - => Alle Tiles Kommunizieren oder keine Kommunikation
  - Verbessert Performance
- Speichermodell ähnlich zu *Scratchpad memory*

Scratchpad memory und Caching

Klassisches Caching

- vollständig transparent aus Programmsicht
- auf keiner Ebene aktiv beeinflussbar
- Kopien von Daten

Scratchpad memory

- bestimmer Adressbereich für *Cache*
- Manuelles verschieben von Daten
- Erweiterung des Speichers

Vergleich CPU, GPU, IPU

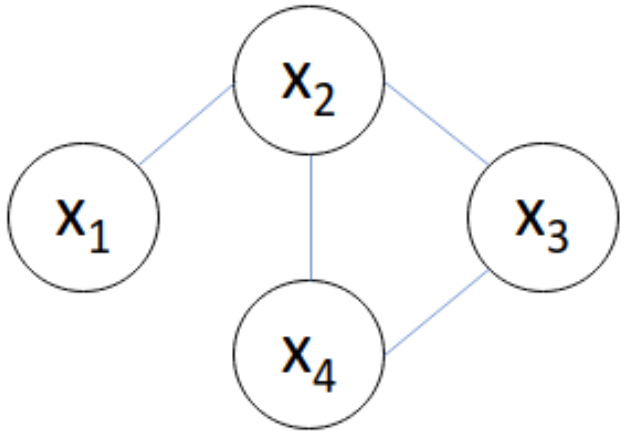
	CPU (7601)	GPU (V100)	IPU (Colossos)
Kerne	32	*80*	1216
SIMD Breite	256b	*1024b*	64b
On-Die RAM	83MiB	16MiB	304MiB
Off-Die RAM	< 2TB	32GiB	Host
TDP	180W	250W	150W
Die Größe	$852mm^2$	$815mm^2$	$800mm^2$

Einordnen der Performance

- BFS gemäß Graph500 Spezifikation
- Graph500 Liste
  - Einführung 2010 als Alternative zu Top500
  - BFS Algorithmen
  - Kroneker Graphen
  - Einheit: TEPS (Transversed Edges Per Second)

BFS in der linearen Algebra

- BFS:
  - Ebenenweise
  - Queue
- Implementierung nach linearer Algebra
- BFS Schritt: Adjazenzmatrix x Frontiervektor

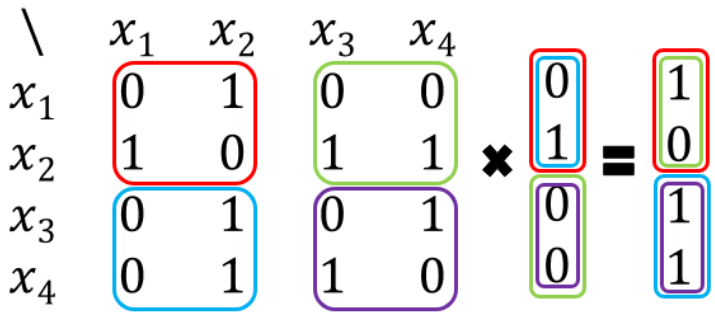


$$\begin{array}{c|cccc} \backslash & x_1 & x_2 & x_3 & x_4 \\ x_1 & 0 & 1 & 0 & 0 \\ x_2 & 1 & 0 & 1 & 1 \\ x_3 & 0 & 1 & 0 & 1 \\ x_4 & 0 & 1 & 1 & 0 \end{array} \times \begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \end{array} = \begin{array}{c} 1 \\ 0 \\ 1 \\ 1 \end{array}$$

Parallele BFS

- 1D Partitionierung
  - Einfacher zu implementieren
  - weniger Kommunikation
  - weniger Rechenaufwand
  - Platzbedarf: $O(n)$
- 2D Partitionierung
  - Platzbedarf: $O(n/\sqrt{p})$

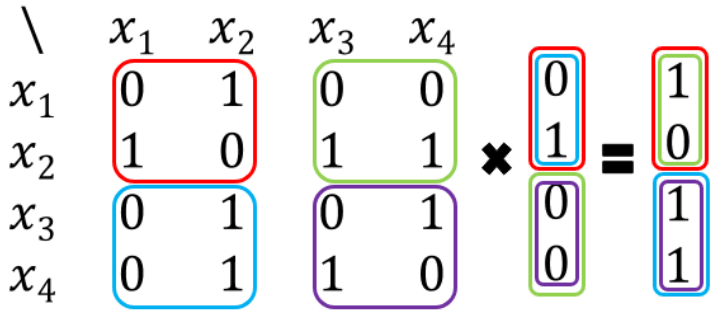
\	x_1	x_2	x_3	x_4
x_1	0	1	0	0
x_2	1	0	1	1
x_3	0	1	0	1
x_4	0	1	1	0



Parallel top-down 2D BSP

Algorithmus:

1. Expansionsphase
   - *Spaltenweise Kommunikation* der neuen Array Offsets
   - Berechnen des BFS Schritt
2. Reduktionsphase
   - *Zeilenweise Kommunikation* zur Reduktion
   - Reduktion der temporären Arrays



Optimierungsmöglichkeiten

- Entfernen isolierter Ecken
  - reduziert den Speicherbedarf um bis zu 36% bzw 74%
- Nutzen von Threads
  - Durch CSC Format schwierig

Testsysteme - Hardware

- CPU Systeme:
  - AMD EPYC 7601 - 64 Kerne (2 x 32, 360W)
  - Intel Xeon Gold 6130 - 32 Kerne (2 x 16, 250W)
- GPU System:
  - NVIDIA Tesla V100 (32GiB RAM, 250W)
- IPU
  - Graphcore Colossos IPU (150W)
  - PCIe Version

Testsysteme - Software

- CPU Systeme:
  - Ref (Graph500)
  - GAP
  - TiTech
- GPU System:
  - Enterprise
  - Gunrock
- IPU
  - iPUG (parallel top-down 2D BSP)

Testsysteme - Startparameter

- Graph500
  - kron_(n)_(e)
- SuiteSparse
  - kron_g500-logn(n)
  - G43
  - coAuthorsDBLP und coPapersDBLP
  - Journals
  - uvm.

Übersicht der Ergebnisse

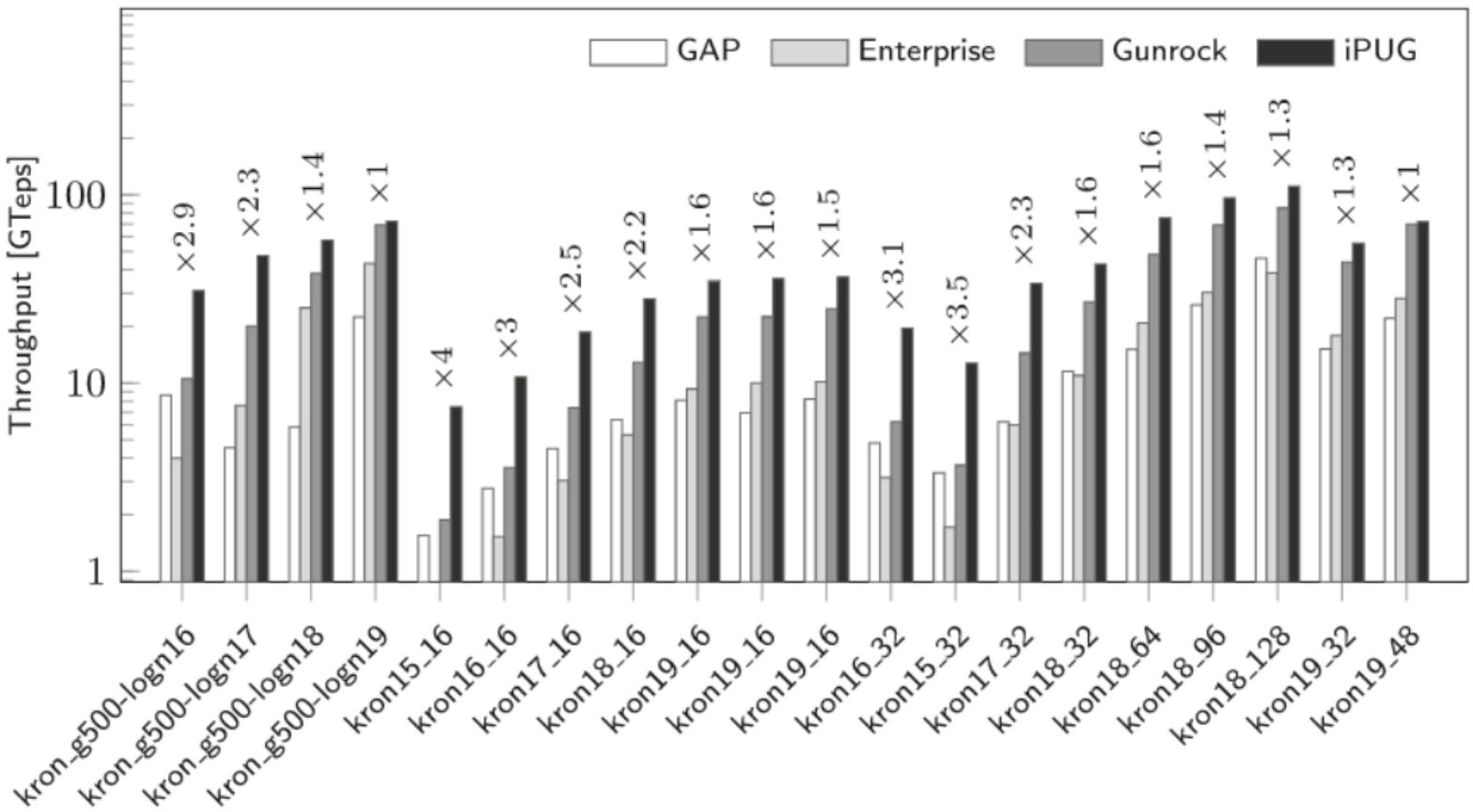


Abbildung: Quelle: [LB21]

Übersicht der Ergebnisse

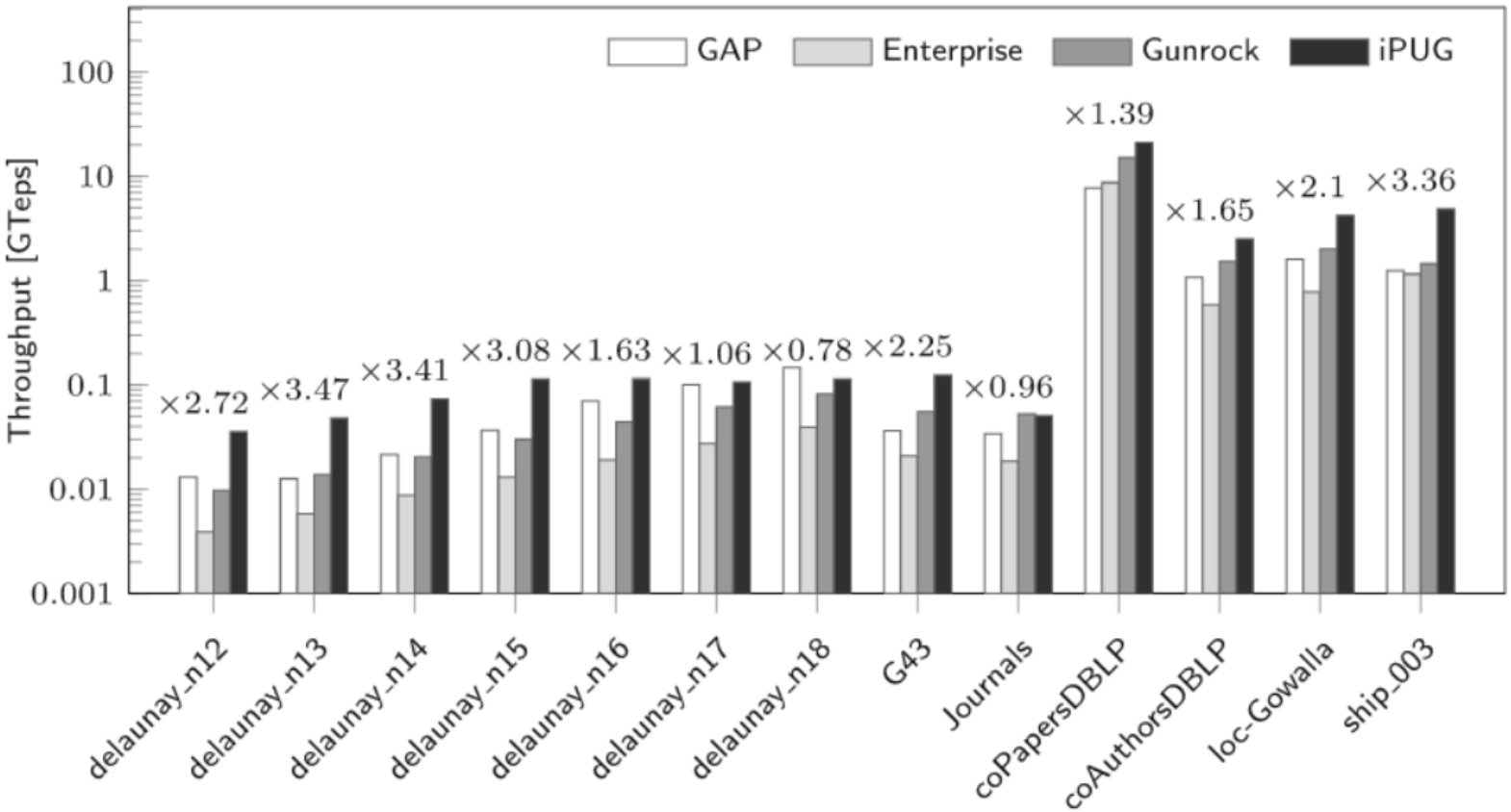


Abbildung: Quelle: [LB21]

Erkenntnisse

- Speedup von 0,78 bis 4 ggü. nächstbeste Architektur
- nur zwei Fälle langsamer
- Durchschnhittlicher Speedup: 2,1
- Kroneker Graphen im Mittel 50% schneller

Skalierungsverhalten

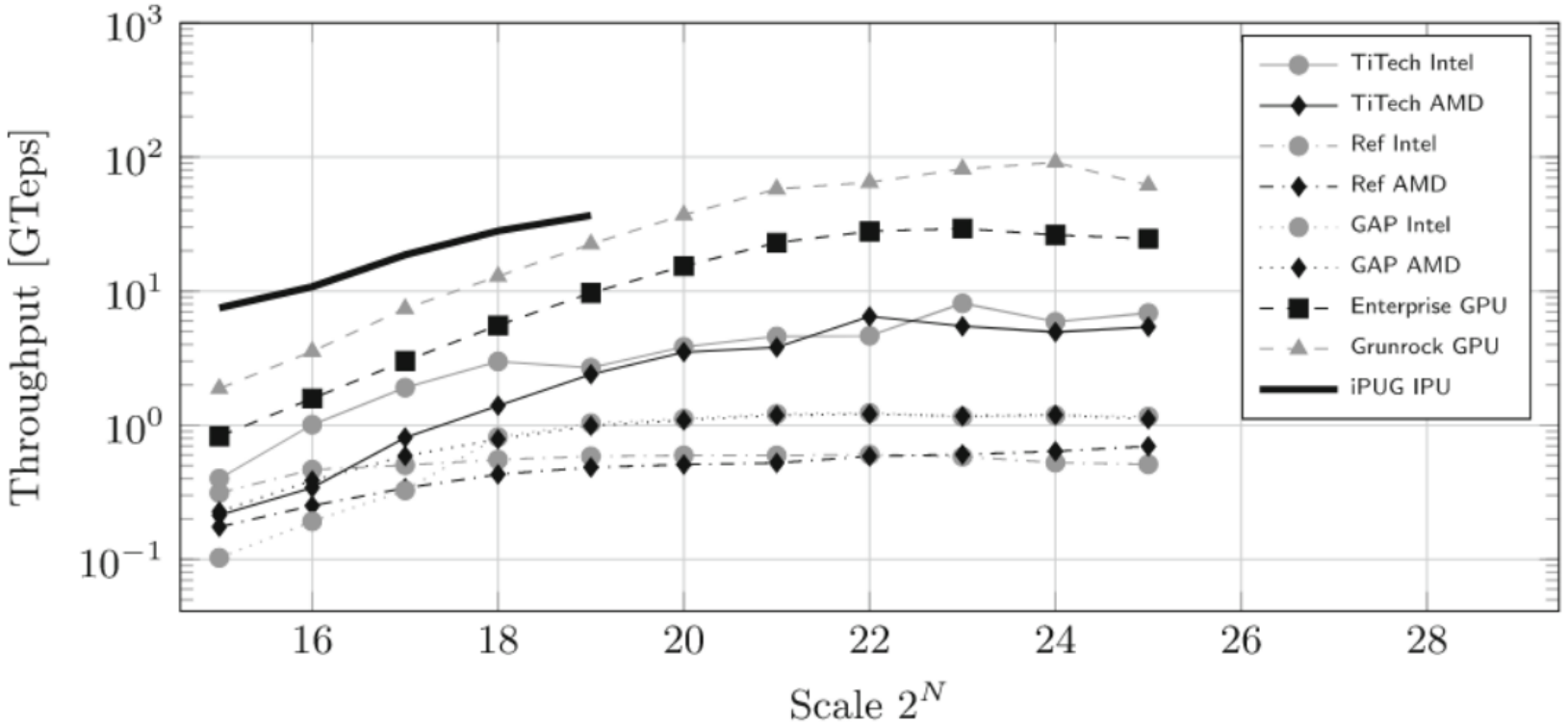


Abbildung: Quelle: [LB21]

Diskussion und Erkenntnisse

- NVIDIA A100 möglicherweise schneller
- unfaire Vergleiche
- Optimierungspotential:
  - Zweite IPU nutzen
  - Multithreading Implementierung verbessern
  - Host für mehr RAM nutzen
- Dennoch: sehr viel Potential
- Caching bei CPUs bringt sehr wenig
- Scratchpad memory wäre besser geeignet
- Wenn Speicherproblem gelöst: Sehr großer Performancegewinn

Literatur I

[Bila] [Online; accessed 13-Januar-2022 URL: https://en.wikichip.org/wiki/intel/microarchitectures/cascade_lake].

[Bilb] [Online; accessed 13-Januar-2022 URL: https://www.graphcore.ai/products/ipu].

[Bilc] [Online; accessed 13-Januar-2022 URL: https://www.servethehome.com/hands-on-with-a-graphcore-c2-ipu-pcie-card-at-dell-tech-world].

[LB21] Daniel Thilo Schroeder Konstantin Pogorelov Johannes Langguth Luk Burchard, Johannes Moe. iPUG: Accelerating Breadth-First Graph Transversals Using Manycore Graphcore IPUs. *Lecture Notes in Computer Science*, 06 2021.

Literatur II

[Lita] CUDA C++ Programming Guide. [Online; accessed 13-Januar-2022 URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html].

[Litb] Designing the Colossos MK2 IPU: Simon Knowles at HOT CHIPS 2021. [Online; accessed 13-Januar-2022 URL: https://www.graphcore.ai/posts/designing-the-colossus-mk2-ipu-simon-knowles-at-hot-chips-2021].

[Litc] How to build a processor for machine learning. [Online; accessed 13-Januar-2022 URL: https://www.graphcore.ai/posts/how-to-build-a-processor-for-machine-learning].

Literatur III

[Litd] How to build a processor for machine learning (Part 2). [Online; accessed 13-Januar-2022 URL: https://www.graphcore.ai/posts/how-to-build-a-processor-for-machine-intelligence-part-2].