# Grafikkarten in der Gittereichtheorie

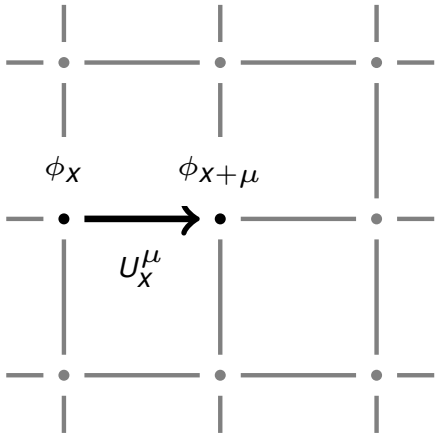
Bjoern Walk
bwalk@kph.uni-mainz.de

Mittwoch, 25. November 2009

# Kurze Übersicht über die Gittereichtheorie

1. Definiere eine klassische, euklidische Feldtheorie im Kontinuum
2. Diskretisiere die entsprechende Lagrange-Dichte
3. Quantisiere die Theorie durch Definition des funktionalen Pfadintegrals
4. Berechne Teilchenspektrum mittels euklidischer Korrelationsfunktionen

- Spinor-Felder $\phi$ auf den Gitterpunkten
- Eichfelder $U$ auf den Verbindungslinien



# Übersicht

- Schnelle Implementierung des **Wilson-Dirac-Operators** $D_{\mathrm{W}} = \gamma_5 M$ auf GPUs

$$M_{x,x'} = -\frac{1}{2}\sum_{\mu=1}^{4}\left(\mathbb{P}_{-}^{\mu} U_{x}^{\mu}\delta_{x+\hat{\mu},x'} + \mathbb{P}_{+}^{\mu} U_{x-\hat{\mu}}^{\mu\dagger}\delta_{x-\hat{\mu},x'}\right) + (4+m)\delta_{x,x'}$$

- Implementierung des **Neuberger-Operators inklusive Projektion** auf tief-liegende Eigenmoden

$$D_{\mathrm{N}} = \frac{1}{a}\left(1 - \frac{A}{\sqrt{A^{\dagger}A}}\right), \quad A = 1 - aD_{\mathrm{W}}(m=0)$$

# Inversion der Fermion-Matrix $D$

Schnelle Invertierung nötig

- als Schritt im HMC-Algorithmus
- zur Bestimmung der Fermion-Massen

# Inversion der Fermion-Matrix $D$

Schnelle Invertierung nötig

- als Schritt im HMC-Algorithmus
- zur Bestimmung der Fermion-Massen

Problematik

- $D$ ist hochdimensional und dünn besetzt, Beispiel für $16^4$-Gitter

$$\dim D = 786432, \quad 0.012\% \text{ besetzt}$$

- man benötigt iterative Algorithmen wie CG, GMRES, BiCG, ...
- es werden **sehr viele** Matrix-Vektor-Multiplikationen des Typs $D\phi = \eta$ benötigt

$\longrightarrow$ effiziente Implementierung der Fermion-Matrix $D$ ist erforderlich

# Wilson-Dirac Kernel

- jedem Gitterpunkt wird ein GPU-Thread zugewiesen
- Spinor $\phi$ besitzt **24**, Eichfeld $U$ besitzt **18** reelle Einträge
- Kenngrössen: 1320 Flop und 1440 Byte
- **aber** für GPUs in der Regel 10:1-Verhältnis

# Wilson-Dirac Kernel

- jedem Gitterpunkt wird ein GPU-Thread zugewiesen
- Spinor $\phi$ besitzt **24**, Eichfeld $U$ besitzt **18** reelle Einträge
- Kenngrössen: 1320 Flop und 1440 Byte
- **aber** für GPUs in der Regel 10:1-Verhältnis

**Optimierung**

Reduktion der benötigten Speicherzugriffe

# Wilson-Dirac Kernel

- Einträge des Spinor-Felder sind unabhängig
- aber $U_x^\mu \in \mathrm{SU}(3)$, damit $UU^\dagger = \mathbb{1}$, $\det U = +1$
- ein Minimum von **8 Parametern** ist für die Beschreibung ausreichend

# Wilson-Dirac Kernel

- Einträge des Spinor-Felder sind unabhängig
- aber $U_x^\mu \in \mathrm{SU}(3)$, damit $UU^\dagger = \mathbb{1}$, $\det U = +1$
- ein Minimum von **8 Parametern** ist für die Beschreibung ausreichend

**SU(3)-Rekonstruktion**

Speichere eine gepackte Darstellung und rekonstruiere $U_x^\mu$ on-the-fly

# SU(3)-Rekonstruktion

**Spalten-Rekonstrukion (12 Parameter)**

Sei $U_x^\mu = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix}$ und $\mathbf{a}, \mathbf{b}$ vorgeben. Dann ist $\mathbf{c} = (\mathbf{a} \times \mathbf{b})^*$.

Vorteil: schnell, praktikabel

Nachteil: nicht der minimale Satz

# SU(3)-Rekonstruktion

## Winkel-Rekonstruktion (8 Parameter)

Analog zu Euler-Winkeln für SO(3) können die Matrixelemente als Funktionen von 8 Winkeln ausgedrückt werden (Bronzan 1988).

Vorteil: minimale Parameterzahl

Nachteil: ineffizient durch viele trigonometrische Funktionen

# Weitere Optimierung (WIP)

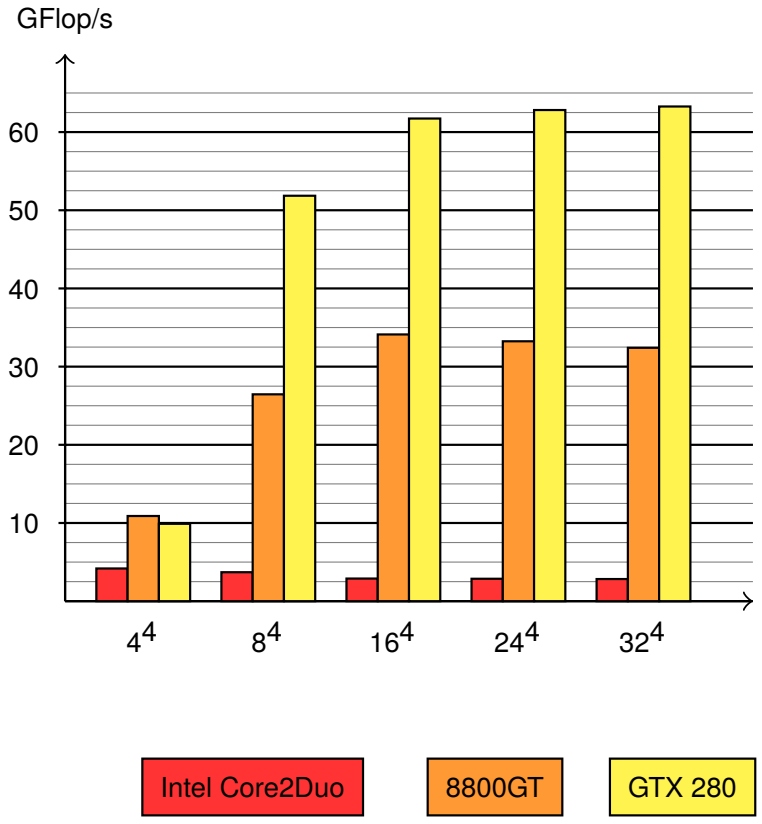
- effizientere **Rekonstruktions-Schemata** (SU(2)-Rotation)
- **mixed-precision**-Algorithmen erlauben geringere Präzession
  $\longrightarrow$ FP16-Kernel besitzt nur noch 50% Speicherbedarf
- **Eichfixierung** erlaubt triviales $U_x^4 \equiv \mathbb{1}\ \forall\, x$ (bis auf Randterme)
- **Basiswechsel** der $\gamma$-Matrizen erlaubt eine diagonale Matrix, z.B.

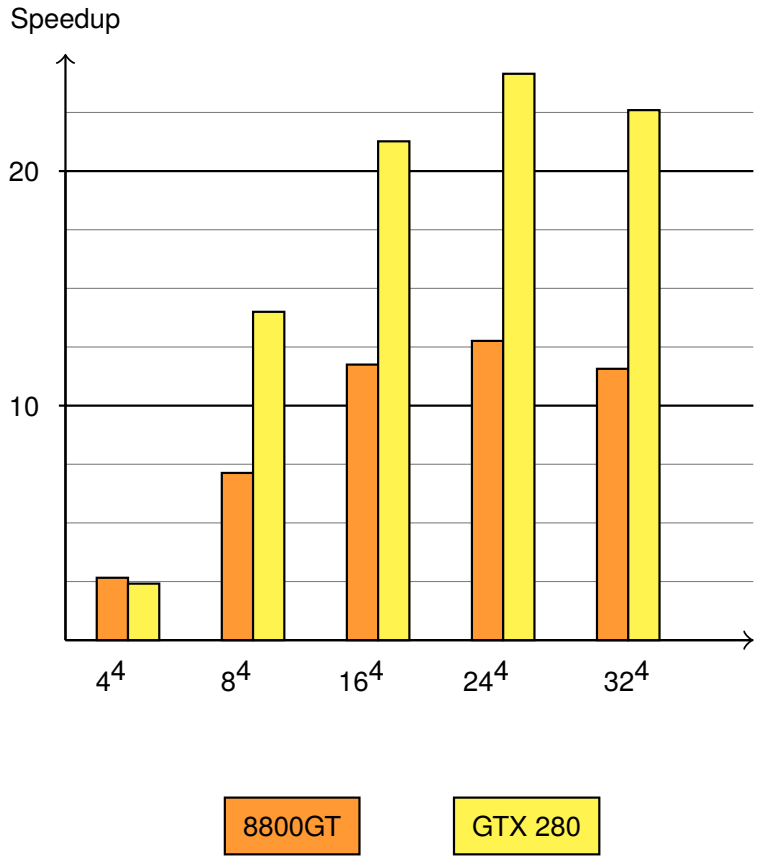
$$P_+^4 = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad P_-^4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

# Neuberger-Operator

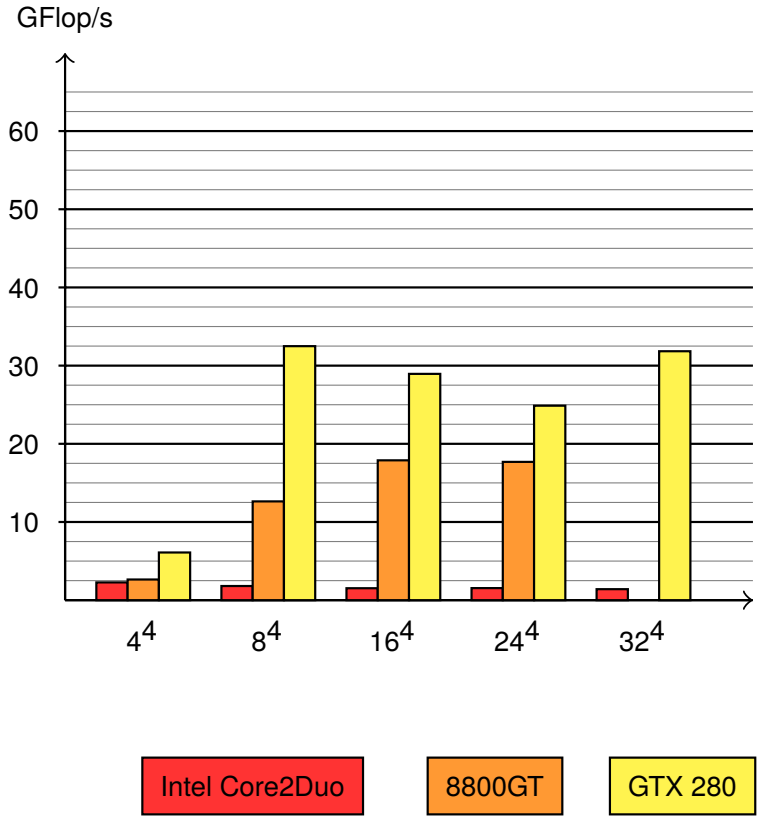
- sign($A$) als polynomielle Approximation, Chebychev-Polynome für numerische Stabilität
- tief-liegende Eigenmoden werden exakt behandelt und herausprojeziert
- Projektion auf Eigenmoden durch Minimierung des Ritz-Funktionals
- viele BLAS-artige Funktionen wie xAXPY, xGEMV und Reduktionen für Normquadrate werden benötigt

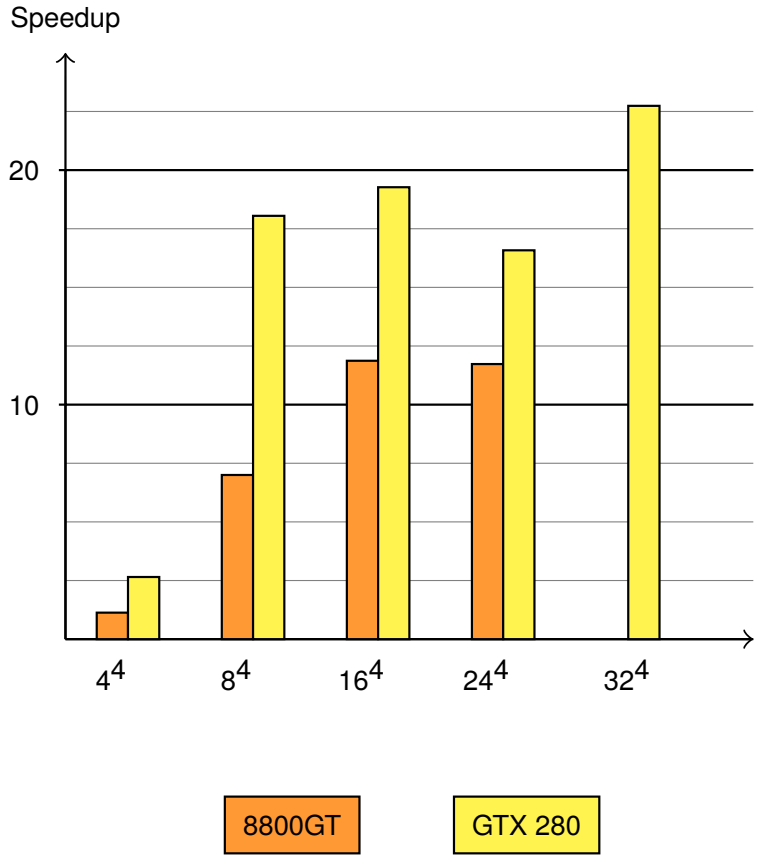
# Benchmark Ergebnisse (Dirac-Operator)



# Benchmark Ergebnisse (Neuberger-Operator)



# Zusammenfassung und Ausblick

| | |
|---|---|
| Dirac-Operator | **65 GFlop/s**, Speedup $\times$**22** |
| Neuberger-Operator | **30 GFlop/s**, Speedup $\times$**20** |

- Algorithmen der Gittereichtheorie sehr speicherintensiv
- Effiziente Implementierung auf Grafikkarten dennoch möglich
- Weitere Performance-Verbesserungen in Aussicht (ref. *arXiv:0911.3191*)
- Richtige Physik auf Grafikkarten