Optimierungen der Lattice Boltzmann Methode auf x86-64 basierten Architekturen

J. Treibig, S. Hausmann, U. Ruede

15.09.05 / ASIM 2005 - Erlangen

Gliederung

Motivation

- Compiler generieren ineffizienten Code:
  - Aktuelle Hardwareentwicklungen werden nicht berücksichtigt
  - Die Hochsprache bietet zu wenige Informationen für eine effiziente Optimierung
- Für beste Ergebnisse ist es notwendig aktuelle Fähigkeiten der Prozessoren zu nutzen Techniken wie SIMD und Prefetching erweitern

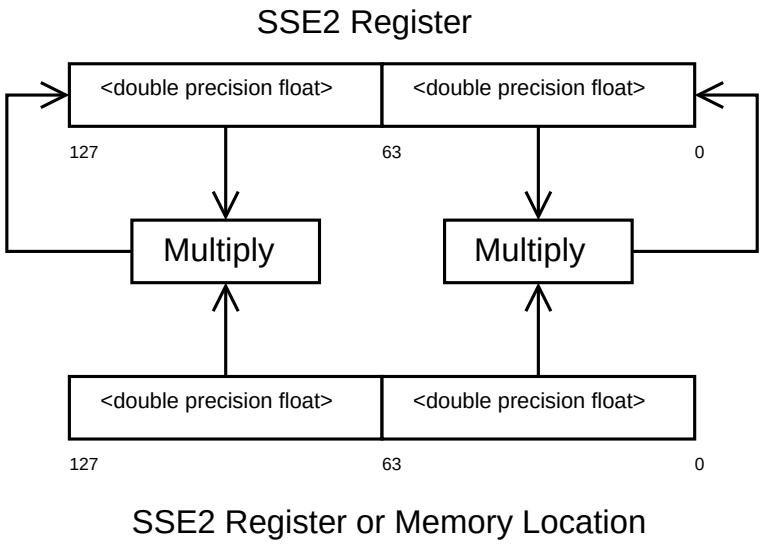
Ziel: Was können moderne Architekturen leisten.

Grundlagen

- Lattice Boltzmann in 3D
  - D3Q19 Modell
  - Reguläres Gitter
  - 19 Werte pro Zelle
  - Rechnung mit Fließkommazahlen doppelter Genauigkeit
- Untersuchung auf x86-64 basierten Architekturen
  - Intel Pentium 4 / Xeon
  - AMD Athlon-64 / Opteron
- Es wurden die effizientesten Algorithmen in Assembler programmiert mit dem Schwerpunkt aktuelle Intruktionssatzerweiterungen besser zu nutzen und mögliche Engpässe zu beseitigen.

Pseudo-Vektorisierung

SSE2 Befehle ermöglichen die Kodierung von zwei/vier Operationen mit einer Instruktion

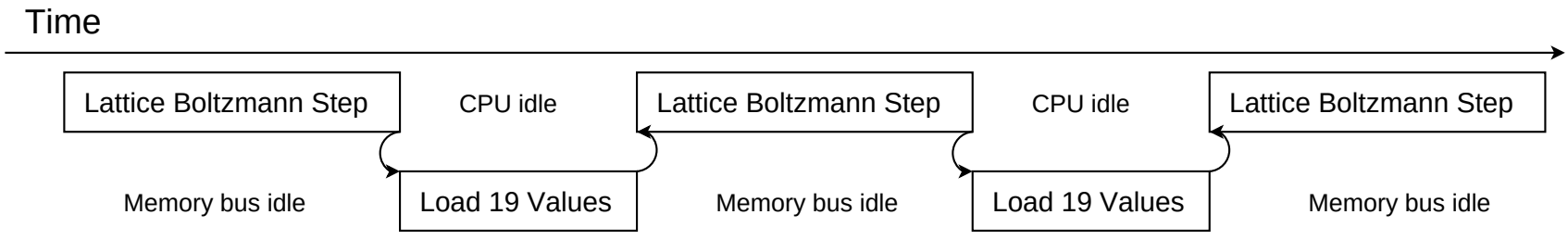


Momentan noch keine *echt* parallele Ausführung, aber:

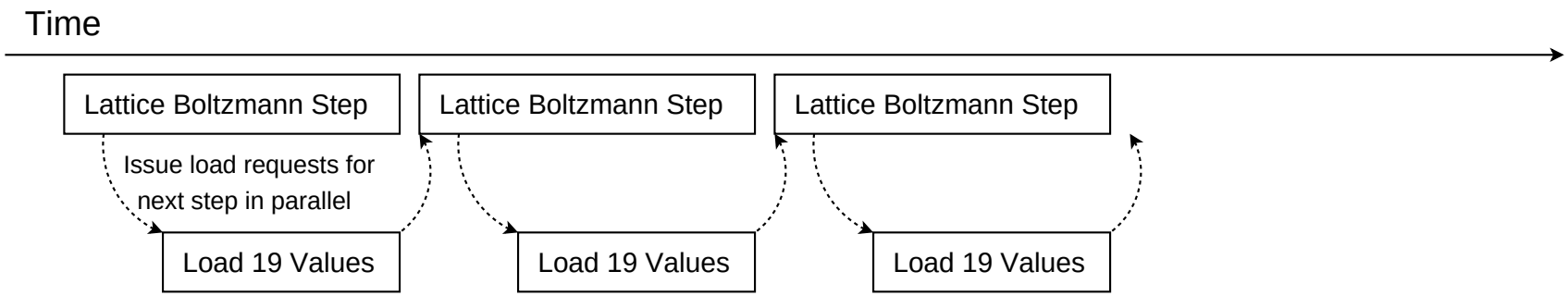
- Mehr Befehle passen in Instruktionsfenster des Prozessors
- Erleichtert Erkennung von Abhängigkeiten

Software Prefetching

Ohne Prefetching:



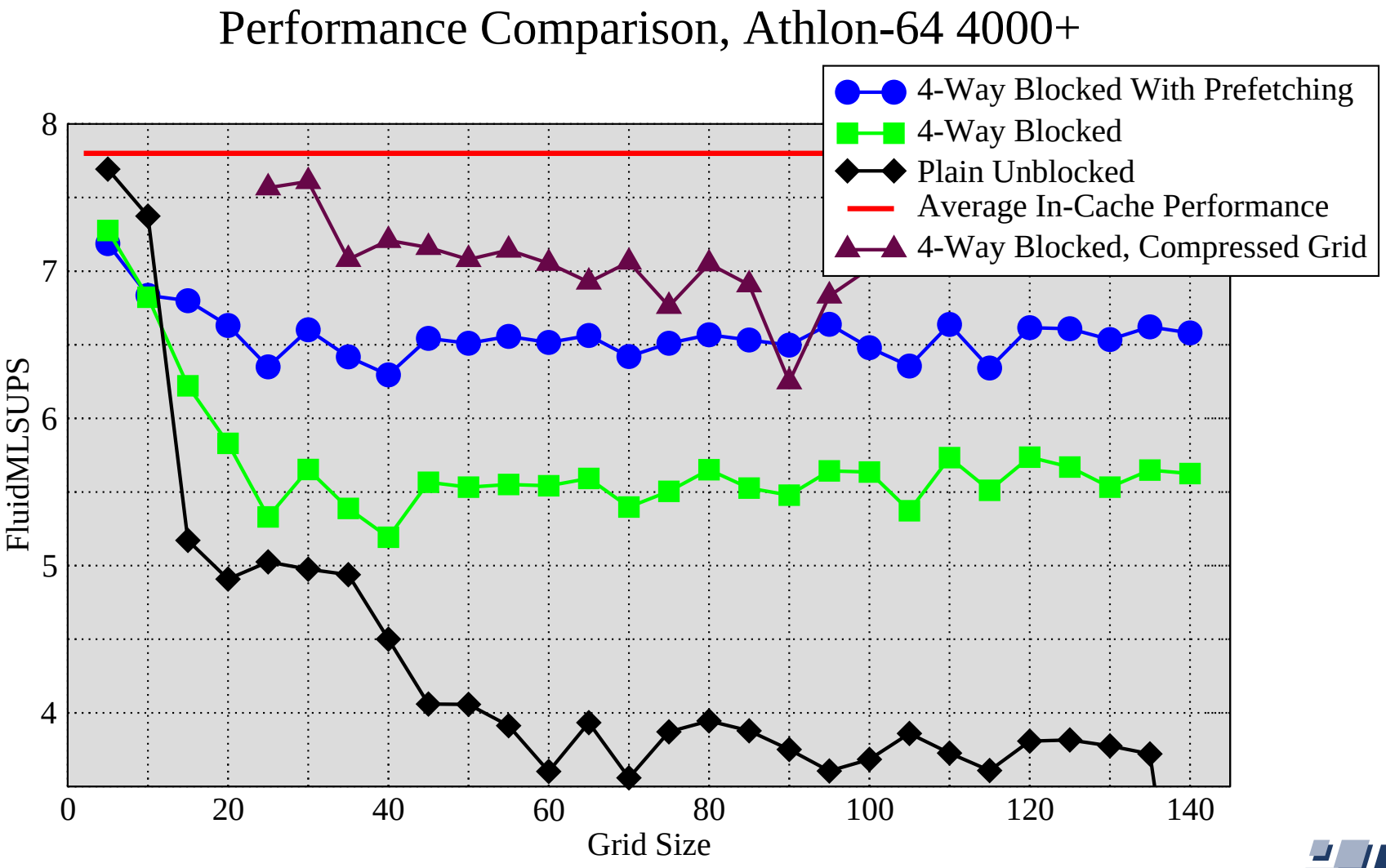
Mit Prefetching:



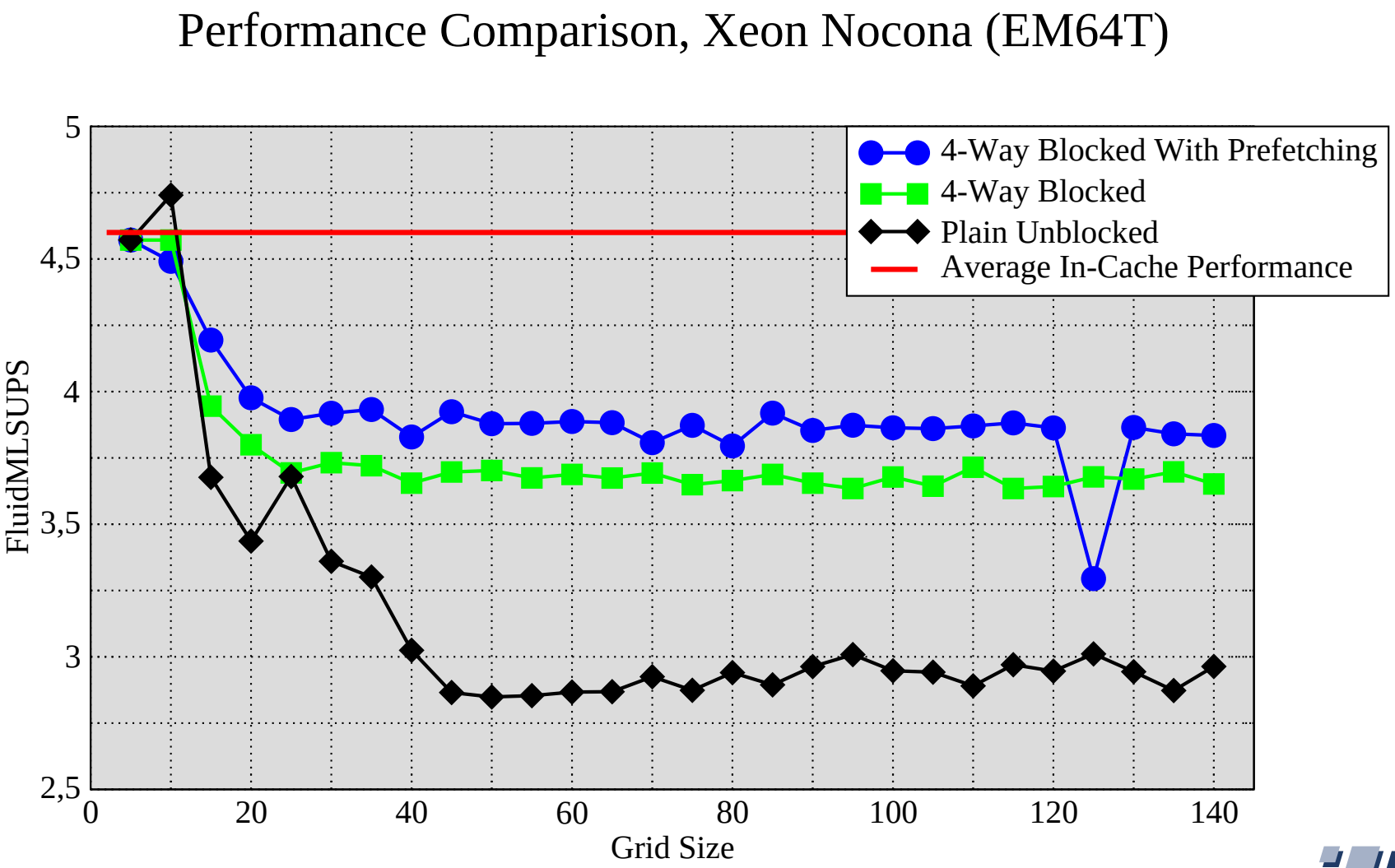
Verwendete Maschinen

	Athlon-64 4000+	**Xeon Nocona**
Taktfrequenz	2.4 GHz	3.4 GHz
L2 Cache	1 MByte	1 MByte
L1 Cache	64 kByte	16 kByte
L2 Zugriffszeiten	11 Takte	27 Takte
L1 Zugriffszeiten	3 Takte	3 Takte
Cacheline Größe	64 Byte	64(128) Byte

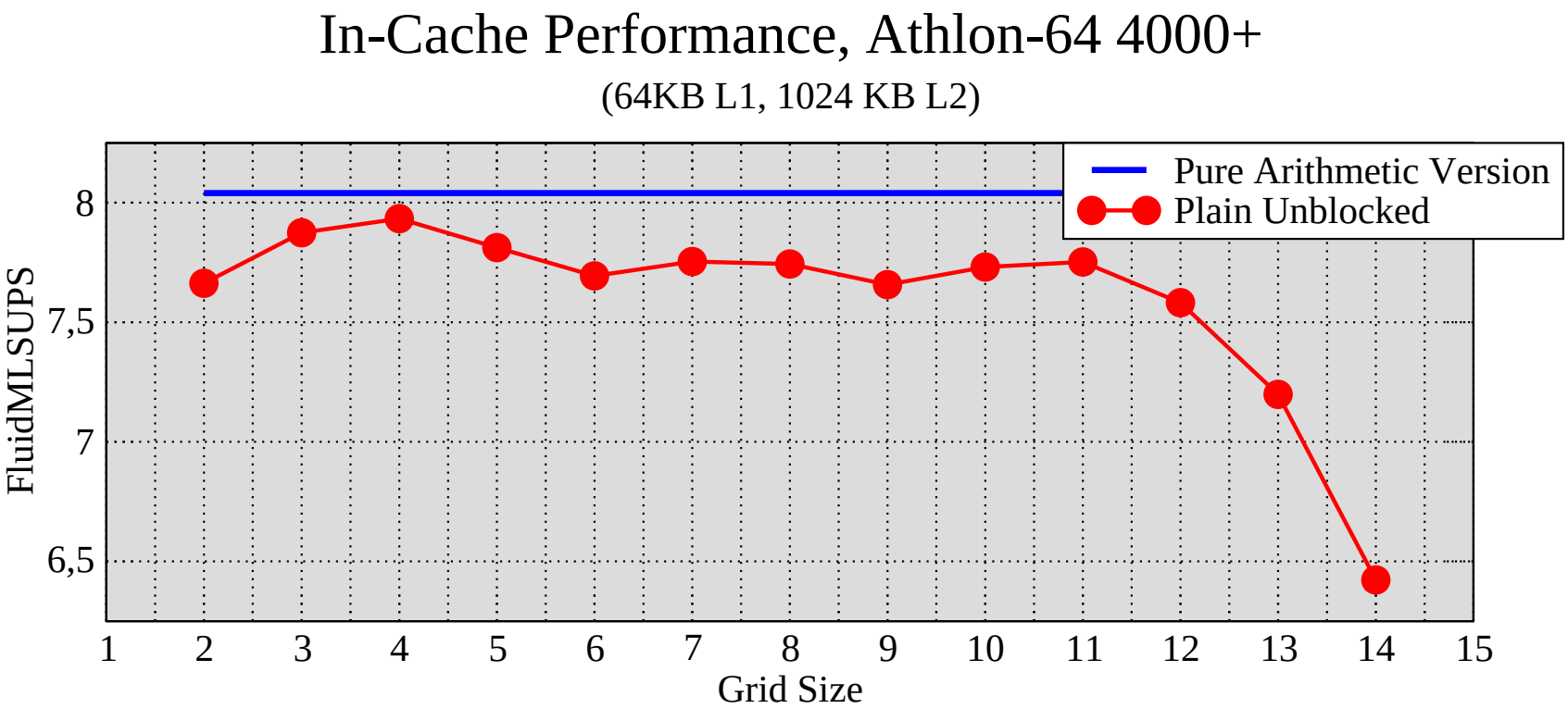
Athlon-64 4000+



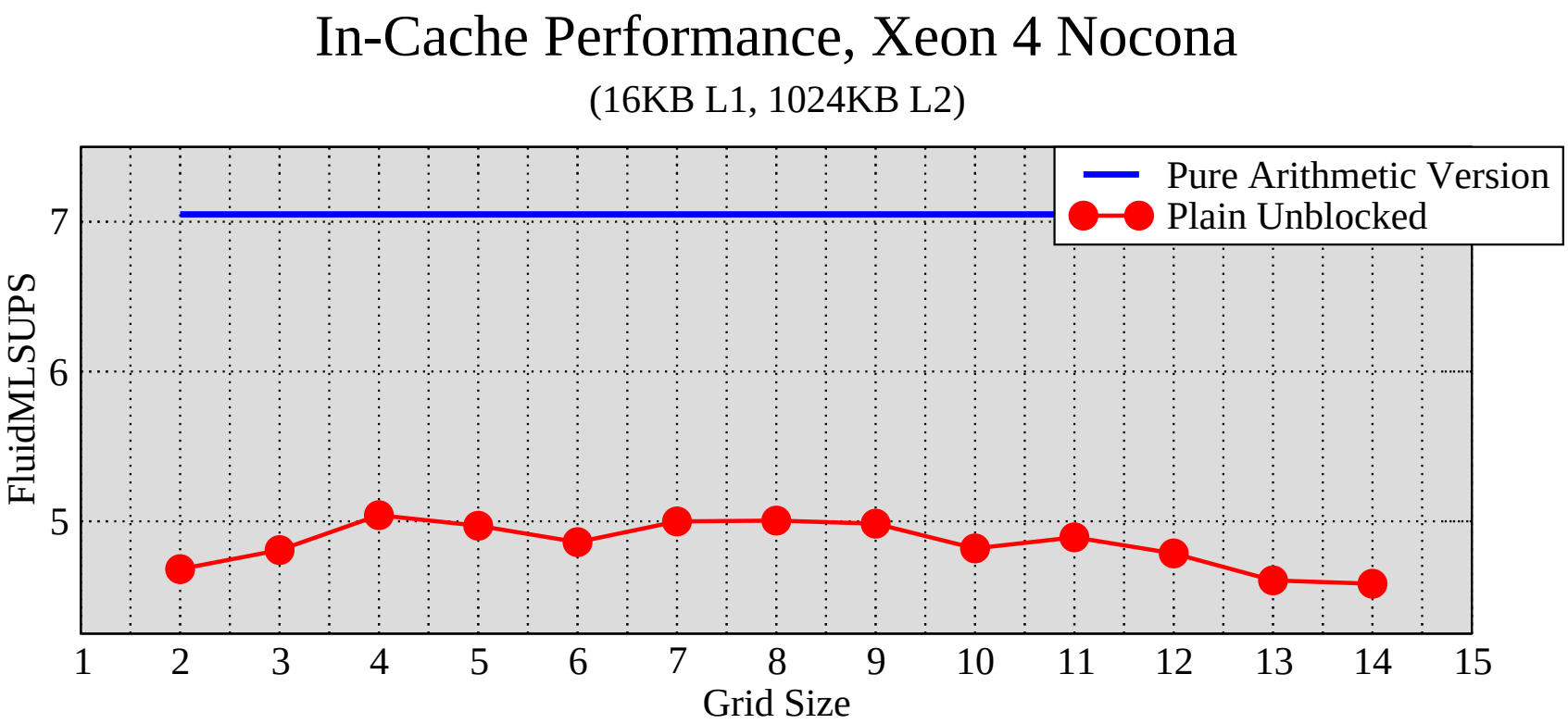
Xeon Nocona


Performance Comparison, Xeon Nocona (EM64T)
FluidMLSUPS
Grid Size
5
4,5
4
3,5
3
2,5
0
20
40
60
80
100
120
140
4-Way Blocked With Prefetching
4-Way Blocked
Plain Unblocked
Average In-Cache Performance

Athlon-64 4000+



Xeon Nocona


In-Cache Performance, Xeon 4 Nocona
(16KB L1, 1024KB L2)
Pure Arithmetic Version
Plain Unblocked
FluidMLSUPS
7
6
5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Grid Size

Arithmetische Abschätzung

Verhältnis aus Multiplikationen zu Additionen verringert erreichbare Performance:

$$P_{\mathrm{LBM}} = \frac{n_{\mathrm{a}} + n_{\mathrm{m}}}{2 \cdot \max(n_{\mathrm{a}}; n_{\mathrm{m}})} \cdot \mathrm{PeakFLOP/s}$$

Ergibt $\approx$ 87 %

- Athlon-64: 26 MLSUPS
- Xeon Nocona: 17 MLSUPS

Zusammenfassung

- Die Speicherlimitierung verschiebt sich nach Optimierungen zu einer arithmetischen Limitierung
- Mit Cache Optimierungen 80-90 % der In-Cache Performance gemessen
- In-Cache Performance zumindest bei Athlon-64 sehr nah an speicherloser Performance
- Große Lücke zu arithmetischem Limit
- Ein besseres *software scheduling* könnte bessere Ergebnisse bringen