TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Vorstellung der zur Auswahl stehenden Praktikumsaufgaben

14. Januar 2009

INF 1046
Nöthnitzer Straße 46
01187 Dresden
0351 - 463 38783

Andy Georgi

ZIH
Center for Information Services & High Performance Computing

Inhalt

Assembler-Programmierung

- Implementierung der Matrixmultiplikation in Assembler
- Integration und Optimierung in BenchIT sowie Messung der Laufzeit für die verschiedenen Optimierungschritte
- Vergleich der Laufzeit mit der Implementierung in einer höheren Programmiersprache

Performance Counter

- Machen Sie sich mit den BenchIT-Kerneln `numerical.matmul.C.0.0.double` und `numerical.matmul.C.0.0.double_blocked` vertraut
- Ergänzen Sie die beiden BenchIT-Kernel um das Auslesen von Performance-Countern, wobei der konkrete `EventCode` explizit im PARAMETERS File angegeben werden kann und im Plot dargestellt wird
- Finden Sie heraus aus welchem Grund `numerical.matmul.C.0.0.double_blocked` bei gleicher Matrixgröße unterschiedliche FLOPS-Raten erzielt als `numerical.matmul.C.0.0.double`
- Weisen Sie Ihre Aussage mit Hilfe der Performance Counter nach und validieren Sie die Werte mit Hilfe von VampirTrace

Verbindungsnetzwerk I

- Messen Sie Latenzzeit und Datenrate des Netzwerks für die Interprozesskommunikation
- Bestimmen Sie den Einfluss durch den Switch
- Optimieren Sie die TCP/IP Einstellungen um die Latenzzeit zu verringern bzw. die Datenrate zu erhöhen
- Untersuchen Sie den Einfluss von Jumbo Frames auf die Performance

Verbindungsnetzwerk II

- Ersetzen Sie den TCP/IP-Stack durch "Reliable Low Latency Ethernet Sockets"
- Vergleichen Sie die Performance beider Stacks auf Protokollebene
- Integrieren Sie den bereitgestellten ETH Byte Transport Layer (BTL) und vergleichen Sie die resultierende Latenzzeit und Datenrate mit dem vorhanden TCP BTL
- Literatur: `http://www.tu-chemnitz.de/informatik/service/if-berichte/pdf/CSR-06-06.pdf`
- Software: `http://archiv.tu-chemnitz.de/pub/2006/0025/data/enet_module_btl.tar.gz`

Compiler

- Machen Sie sich mit POV-Ray (`http://www.povray.org/`) und RAxML (`http://icwww.epfl.ch/~stamatak/index-Dateien/Page443.htm`) vertraut
- Führen Sie einen Referenzlauf mit einem für die Architektur geeigneten Compiler ohne Optimierungen durch und bestimmen Sie die Laufzeiten für definierte Problemgrößen
- Bestimmen Sie anschließend die optimalen Compilerflags und zeigen Sie welche Parameter einen besonders starken Einfluss auf die Performance haben

Fortran I

- Implementieren Sie den `numerical.matmul.C.0.0.double_blocked` Kernel in Fortran
- Vergleichen Sie das Laufzeitverhalten des C und Fortran Kernels
- Geben Sie Ursachen für die ggf. auftretenden Unterschiede an und versuchen Sie diese mit Hilfe von Vampir zu belegen

Fortran II

- Implementieren Sie einen parallelen Sortieralgorithmus über abstrakten Datentypen in Fortran
- Bestimmen Sie den Speedup durch die Parallelisierung und vergleichen Sie die Laufzeit mit einer C oder Java-Implementierung
- Stellen Sie den Programmablauf aller Versionen mit Hilfe von Vampir dar

MAUI

- Analysieren Sie die in MAUI integrierten Scheduling-Algorithmen
- Finden Sie die optimalen Einstellungen für
  - maximalen Durchsatz
  - minimale Wartezeit
  - maximale Auslastung
  - faires Scheduling (identische mittlere Wartezeit)

  für unterschiedliche definierte Lastszenarien
- Stellen Sie die Belegung der Prozessoren über der Zeit mit Hilfe eines Gantt-Diagramms dar

Monitoring

- Implementieren Sie ein Plugin für das auf dem System verwendete Monitoring-System, welches in definierten Zeitintervallen mit Hilfe der verfügbaren Performance Counter die Anzahl der Floating Point und Integer Operationen auf den Compute Nodes ausliest
- Stellen Sie entsprechend die FLOPS und IOPS-Raten graphisch dar
- Werden definierte Grenzwerte über- bzw. unterschritten, soll eine Benachrichtigung über das Extensible Messaging and Presence Protocol (XMPP) erfolgen

Wissenschaftliche Anwendungen und Bibliotheken I

- Machen Sie sich mit dem Wave Propagation Program (`https://computation.llnl.gov/casc/serpentine/index.html`) vertraut
- Führen Sie Referenzläufe mit unterschiedlichen Eingangsdaten ohne zusätzliche Optimierungen durch
- Bestimmen Sie anschließend die optimalen Compilerflags für Ihr Cluster
- Vergleichen Sie die Ergebnisse mit Messungen auf `mars` und `deimos` unter Verwendung der gleichen Eingangsdaten

Wissenschaftliche Anwendungen und Bibliotheken II

- Bestimmen Sie die optimale Laufzeit für RAxML auf dem Intel Atom Cluster
- Nutzen Sie anschließend die `pthreads`-Version von RAxML mit den vorgegebenen Datensätzen um den Speedup durch Hyperthreading zu bestimmen
- Generieren Sie dazu Überlastszenarien und visualisieren Sie das Ergebnis

Wissenschaftliche Anwendungen und Bibliotheken III

- Installieren Sie HPCC und führen Sie eine Referenzmessung durch
- Optimieren Sie mit Hilfe von Compilerflags und alternativen Bibliotheken
- Bestimmen und dokumentieren Sie den resultierenden Speedup in den verschiedenen Optimierungsschritten

Vampir Trace-File-Generator

- Implementieren Sie einen Trace-File-Generator für BenchIT, welcher folgende Anforderungen erfüllt:
  - Definition einer Metasprache, mit dessen Hilfe die Art und Menge der auftretenden parallelen Konstrukte, die Anzahl der Prozesse und die Nachrichtengröße spezifiziert werden kann
  - Generierung eines ausführbaren C oder Fortran Kernels welcher der Spezifizierung entspricht und über der Nachrichtengröße iteriert
  - Automatische Instrumentierung so dass nach Ausführung des generierten Kernels in BenchIT die Laufzeiten in Abhängigkeit der Nachrichtengröße dargestellt werden und zusätzlich ein Trace-File zum Programmablauf erzeugt wird

Cell-Programmierung

- Verwenden Sie den seriellen BenchIT Gauss-Kern zum lösen linearer Gleichungssysteme als Ausgangspunkt und implementieren Sie zwei parallele Versionen für die Cell Broadband Engine
- Die erste Version soll mittels OpenMP-Direktiven und dem IBM XLC Single Source Compiler, die zweite dagegen möglichst effizient unter Nutzung der SPEs von Hand programmiert und mittels ppu-gcc erzeugt werden
- Vergleichen Sie beide Versionen mittels BenchIT und bewerten Sie Ihre Ergebnisse