TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik - Institut für Technische Informatik - VLSI-Entwurfssysteme, Diagnostik und Architektur

Leistungsanalyse des Java-Bytecode-Prozessors SHAP

Belegverteidigung

Christian Greth
s7277485@mail.inf.tu-dresden.de

Dresden, 6. Januar 2010

Aufgabenstellung

1. Literaturstudium zu Metriken für Leistungsbewertung und Benchmarks
2. Literaturstudium zu Java-Bytecode Prozessoren
3. Vergleich einiger Java-Plattformen mittels ausgewählter Benchmarks
4. Einordnung des Java-Bytecode Prozessors SHAP

Motivation

- Leistungsanalysen haben wichtige Rolle im Bereich CPU-, Speicher- und System-Entwicklung eingenommen
- Produkt auf dem aktuellen Markt einordnen
- Probleme aufdecken und Verbesserungen finden

Kenngrößen und Benchmarks

- Kenngrößen/Maßzahlen liegen keiner Messung zu Grunde
- Berechnungen einer theoretischen maximalen Leistung

- Benchmarks sind Sammlungen von Programmen oder Programmkernen
- Vergleichszahlen ergeben sich durch Messungen

Allgemeine Methoden

- Millions of Instructions Per Second - MIPS
  - eine der bedeutendsten Maßzahlen
  - theoretische Maßzahl
  - Anzahl der pro Sekunde ausführbaren Befehle
  - berücksichtigt keine angeschlossene Peripherie

- Millions of Floating-Point Operations Per Second - MFLOPS
  - theoretische Maßzahl wie MIPS
  - Anzahl der pro Sekunde ausführbaren Gleitkommaoperationen
  - vernachlässigt ebenfalls Wortbreiten und Ein-/Ausgabevorgänge

- schnelle und einfache Berechnung
- weit verbreitet als Maßzahlen zum Vergleich

Allgemeine Methoden

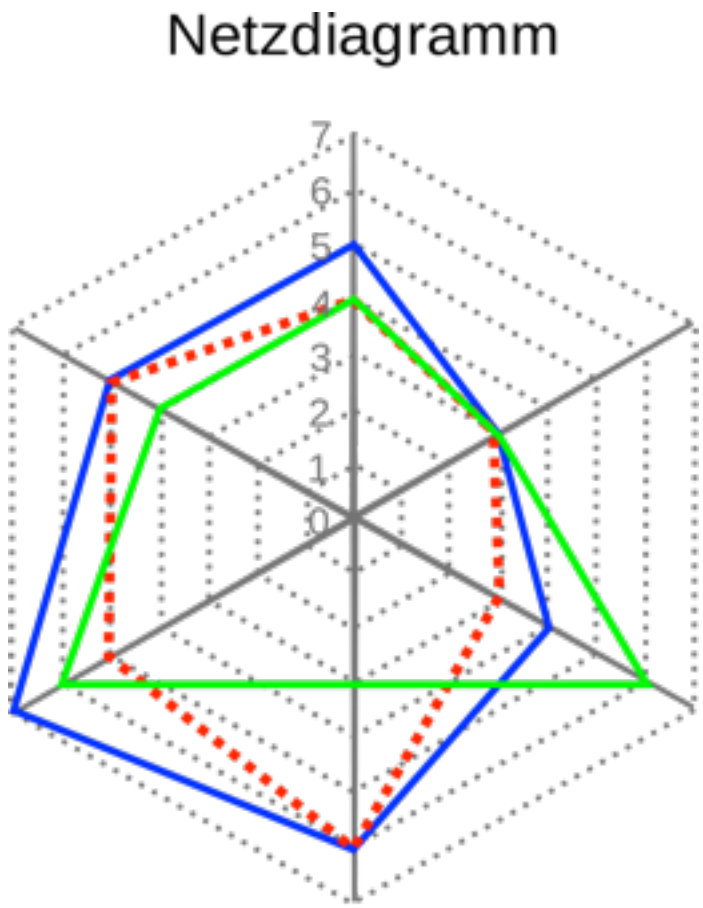
- Befehlsmixe und Kernprogramme
  - gibt die durchschnittliche Ausführungszeit eines Befehls an
  - Ausführungszeiten können gewichtet bewertet werden
  - Ausführungszeiten der Befehle sind bekannt
  - meist typische Folgen von Maschinenbefehlen (spezielle Anwendung)

- weitere Methoden
  - Cycles Per Instruction - CPI
  - niedriger CPI-Wert steht für hohe Effizienz der CPU

  - Instructions Per Cycle - IPC
  - hoher IPC-Wert steht für hohe Effizienz der CPU

Möglichkeiten zur Visualisierung

- Sterndiagramme/Netzdiagramme
  - Möglichkeit Maßzahlen eines Gesamtsystems darzustellen und in Verbindung miteinander zu setzen
  - jede Achse entspricht einer Maßzahl
  - direkter Vergleich mehrere Systeme möglich
  - überdeckende Werte werden als pareto-optimal bezeichnet (besser im Sinne der Maßzahl)



Benchmark-Programme

- Messungen im praktischen Umgang mit dem zu bewertenden System
- direkter Zugriff auf ein System oder Simulation nötig
- besteht aus einem oder mehreren Programmen
- im Normalfall als Quellcode, um für verschiedene Zielsysteme übersetzen zu können
- bewerten immer ein Gesamtsystem

- Problem: durch Übersetzung/Compilieren mit unterschiedlichen Compilern für unterschiedliche Zielsysteme kann unterschiedlich optimierter Code entstehen

Benchmark-Programme

- SPEC-Benchmarks
  - Benchmark-Suiten für zahlreiche Zielsysteme und Bereiche
  - alle Maßzahlen immer im Bezug auf eine Referenzmaschine
  - besteht im Kern aus Integer- und Gleitkomma-Programmen
  - zusätzlich aber auch Programme um andere Bereiche zu testen
  - die Programme sind oft aus Anwenderprogrammen abgeleitet (komplexe Algorithmen, etc.)

  - die aktuelle Suite SPEC-CPU2006 besteht derzeit aus 12 Integer-intensiven und 17 Gleitkomma-intensiven Benchmarks
  - Benchmark auch für Java-Umgebungen verfügbar

Benchmark-Programme

- LINPACK-Benchmarks
  - nutzt BLAS (Basic Linear Algebra Subprogramms)
  - Lösungen von Systemen linearer Gleichungssysteme
  - numerische Lösungsverfahren
  - Anwendung für Parallelrechner, TOP-500 Liste nutzt LINPACK

- Whetstone- und Drystone-Benchmark
  - jeweils ein einzelnes Programm
  - Whetstone führt Standardtests aus (Ganzzahl-/Gleitkommaoperationen, bedingte Sprünge, Array-index-Operationen)
  - Drystone ist ein synthetischer Benchmark (Anweisungen aus Analysen von tatsächlich verwendeten Sprachkonstrukten)
  - im Powerstone-Benchmark zusammengefasst

Java-Benchmarks

- zahlreiche Benchmarks für Java-Umgebungen verfügbar
- Problem des Vergleichs zwischen VM und Java-Bytecode-Prozessor
- Leistungsfähigkeit der VM durch darunter liegende Hardware bestimmt
- Vergleich von VMs nur auf gleicher Hardware möglich

- Benchmarks: SPECjvm98/2008 und SPEC-JBB2000, LINPACK for Java, Java Bench Embedded, CaffeineMark, UCSD, Java Grande Forum Benchmark Suite, EEMBC

Java-Benchmarks

- SPECjvm98/2008- und SPEC-JBB2000/2005-Benchmark
  - Methoden für XML-Parsing, Serialisierung, Echtzeitanwendung, Crypto-Rechnungen, Kompressions-Tests, MPEG-Audio-Dekodierung
  - viel Speicherbedarf und hohe API-Anforderungen
    - ➡ Einsatz auf eingebetteten Systemen schlecht realisierbar
  - Anwendung im Bereich Desktop-Computer und Server

- LINPACK for Java
  - Teile aus LINPACK für Java portiert
  - berechnet Gleichungssystem Ax = b mit Matrix-Dimension 500x500
  - Ergebnis sind Gleitkomma-Operationen pro Sekunde (Mflop/s) und Gesamtausführungszeit

Java-Benchmarks

- Java Bench Embedded
  - von den Entwicklern des Java Optimized Processor (JOP) erstellt
  - wenig Speicherbedarf und geringe API-Anforderungen
    ➡ sinnvoll für eingebettete Systeme
  - gibt die Anzahl der pro Sekunde ausgeführten Bytecodes an

- CaffeineMark
  - bekannte Benchmark-Suite für Java-Umgebungen
    ➡ auch in einer Version für eingebettete Systeme
  - 9 Tests, die sowohl numerische Berechnungen als auch Grafik- und Zeichenverarbeitungen ausführen
  - Mittelwerte der Zeitmessung bilden auf Referenzmaschine bezogenes Ergebnis

Java-Benchmarks

- UCSD Benchmark
  - Sammlung einfacher Benchmarks: Testmethoden für Arrays, Felder, Schleifen und Exceptions
  - Auswertung der Laufzeit der einzelnen Tests

- EEMBC (GrinderMark-Score)
  - speziell für mobile Geräte mit Java-Unterstützung
  - Cryptography, PNG-Dekodierung, parallele Matrix-Multiplikation, XML-Parsing, Schach-Simulation

- Java Grande Forum Benchmark Suite
  - umfangreiche Benchmark-Suite mit drei Bereichen
  - Low-Level-Operationen, Fourier-Transformationen und Matrix-Multiplikation, 3D-RayTracing und Monte-Carlo Simulation
  - zusätzlich Version für Multiprozessor

Java-Bytecode-Ausführung

- Java-Bytecode vor der Ausführung in Microcode übersetzen
  - ➡ Java Optimized Prozessor
- Mischform aus direkter Bytecode-Ausführung und Ausführung in Microcode
  - ➡ Moon, Komodo, aJile, Jazelle von ARM, AVR32
- Bytecode löst Interrupt aus und die Interruptroutine führt Microcodefolge aus (auch als Traps oder Tapps bezeichnet)
- Speicher nötig um Interruptroutinen abzuspeichern
  - ➡ aJile, Jazelle, AVR32

Java-Plattformen und Konfigurationen

	Technologie	System	Taktung (MHz)	Speicher	Anmerkung
SHAP1	FPGA Xilinx Spartan-3 XC3S1000	-	50 MHz	1 MByte externer SDRAM, 8 kByte Stack, 2 kByte Methoden-Cache	bis zu 32 Threads
SHAP2	FPGA Xilinx Spartan-3E XC3S500E	-	50 MHz	64 MByte externer DDR-RAM (DDR-200), 8 kByte Stack, 2 kByte Methoden-Cache	bis zu 32 Threads
SHAP3	FPGA Xilinx Virtex5 XC5VLX50T	-	66 MHz	1 MByte externer SDRAM, 8 kByte Stack, 2 kByte Methoden-Cache	bis zu 32 Threads

- drei unterschiedliche SHAP-Konfigurationen
- durch XILINX-Entwicklerboards charakterisiert

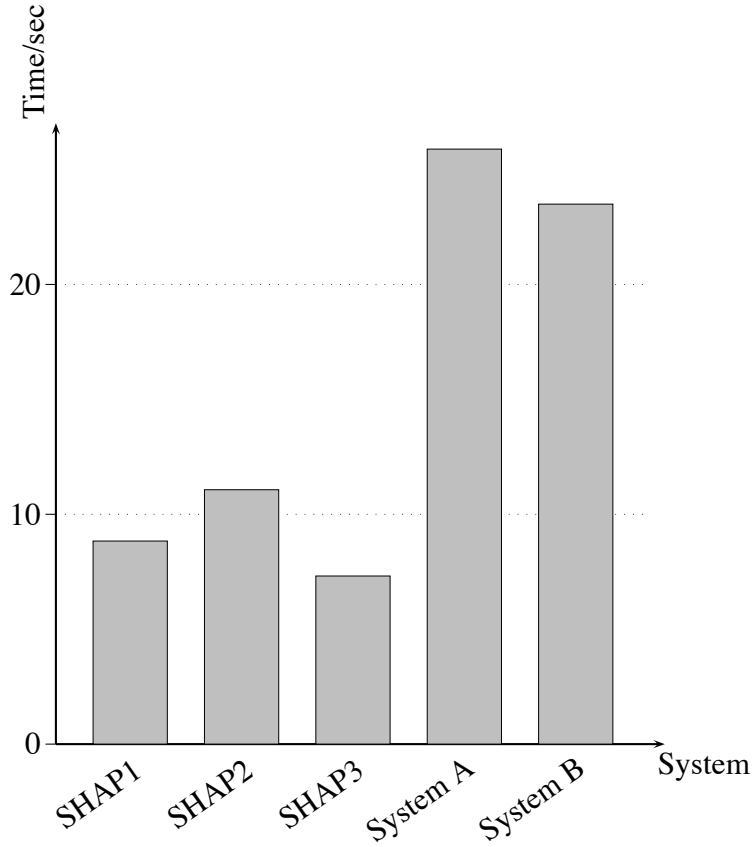
Java-Plattformen und Konfigurationen

	Technologie	System	Taktung (MHz)	Speicher	Anmerkung
JOP	FPGA Altera, Xilinx	-	100 MHz	1830 LCs, 4 kByte Methoden-Cache	
Komodo	FPGA Xilinx	-	33 MHz	2600 LCs, 4 kByte Methoden-Cache	multithreadfähig
aj100	ASIC 0.25μ	-	103 MHz	48 kByte RAM	
System A	RISC Power PC 603e	MAC OS 8	200 MHz	80 MByte RAM 512 kByte Cache	
System B	CISC Pentium	Linux	200 MHz	32 MByte RAM	

- zwei SHAP-ähnliche Systeme zum Vergleich (JOP und Komodo)
- eine ASIC-Entwicklung zum Vergleich (aj100)
- System A/B nutzen eine VM

UCSD Benchmark

- UCSD-Tests:
  - Durchlauf einer leeren Schleife
  - Multiplikation und Addition von Integerzahlen
  - Array-Zuweisungen
  - Objektattribut-Zugriff
  - Methodenaufruf eines Objektes
  - auslösen einer Exception
  - Threadwechsel
- jeder Test 1.000.000 ausgeführt



UCSD Laufzeit-Vergleich

CaffeineMark

Benchmark	SHAP1		SHAP2		SHAP3	
Highscore	443		347		536	
	TestRes	LastRes	TestRes	LastRes	TestRes	LastRes
Sieve	155	98	124	98	189	98
Loop	193	2017	136	2017	221	2017
Logic	336	0	328	0	445	0
String	443	708	347	708	536	708
Method	262	166650	211	166650	322	166650

- Sieve: Berechnungen von Primzahlen nach der Methode von Eratosthenes
- Loop: Sortierungen und Sequenz-Generationen
- Logic: Lösung Boolscher Terme
- String: konkateniert und trennt Char-Sequenzen
- Method: rekursive Methodenaufrufe

- die Tests Graphics, Image und Dialog entfallen in der eingebetteten Variante

Java Bench Embedded

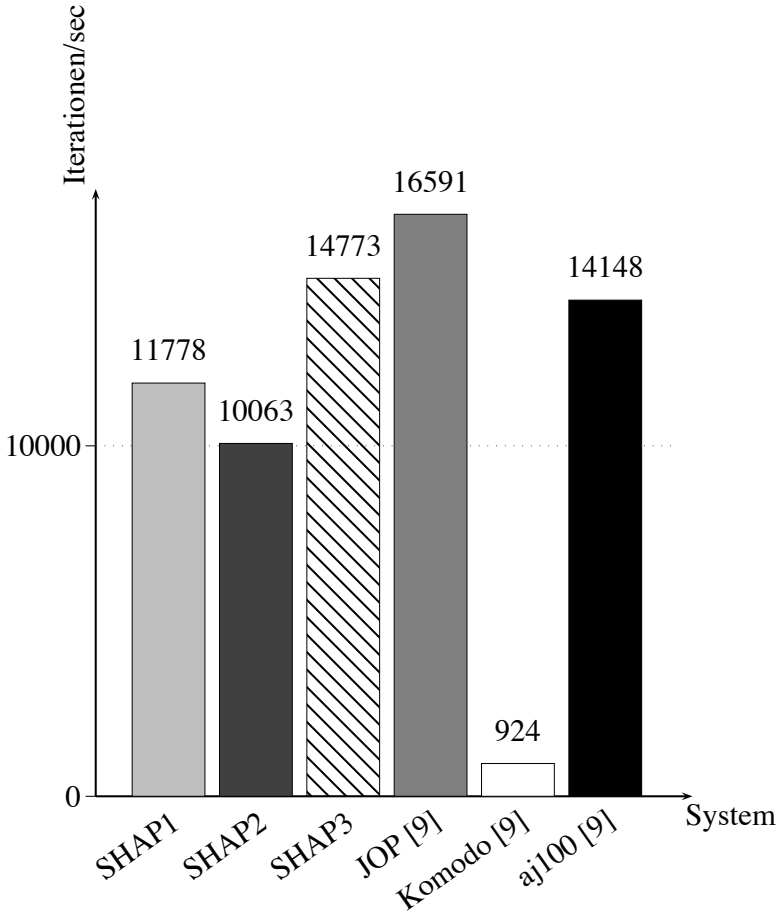
Micro Benchmarks	SHAP1	SHAP2	SHAP3	JOP [9]	Komodo [9]	aj100 [9]
iload 3 iadd	2	2	2	2	8	8
iinc	5	4	5	8	4	11
ldc	9	7	8	9	40	9
if icmplt taken	7	7	6	6	24	18
if icmplt not taken	7	7	6	6	24	14
getfield	12	18	14	22	48	23
getstatic	12	18	14	15	80	15
iaload	16	27	20	37	28	13
invoke	50	79	59	133	384	115
invokestatic	36	41	37	100	680	95
invokeinterface	51	79	60	149	1617	153

Angabe erfolgt in Anzahl der benötigten Taktzyklen

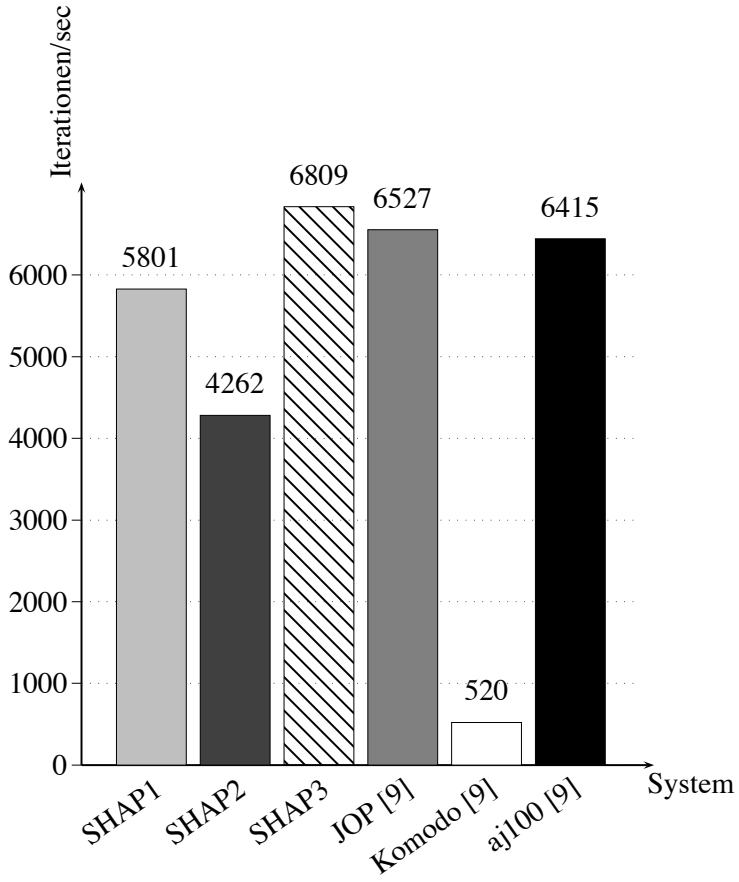
- Abarbeitung interessanter Bytecodes und Bytecode-Sequenzen
- Ergebnis: Anzahl der benötigten Taktzyklen

- zusätzlich zwei „Real-World“-Applikationen
  - kfl simuliert ein Motorsteuergerät
  - UdpIp simuliert UDP-Kommunikation zwischen Server und Client

Java Bench Embedded



JBE: kfl-Application Laufzeitergebnisse



JBE: UdpIp-Application Laufzeitergebnisse

Java Grande Forum Benchmark Suite

Benchmark	SHAP1		SHAP2		SHAP3	
	Zeit	Ergebnis	Zeit	Ergebnis	Zeit	Ergebnis
Section1:Arith	ms	add/s	ms	add/s	ms	adds/s
Add:Int		6447855		6447855		8617715
Add:Long		2180463		2177912		2914265
Section1:Arith	ms	divides/s	ms	divides/s	ms	divides/s
Div:Int		1187246		1185871		1590062
Div:Long		4571		4557		6124
Section1:Assign	ms	assign/s	ms	assign/s	ms	assign/s
Same:Scalar:Class		2259238		1769636		2725397
Same:Array:Class		872008		588911		979998
Other:Scalar:Class		2256003		1769636		2722861
Other:Array:Class		858916		581322		969513
Section1:Loop	ms	iterations/s	ms	iterations/s	ms	iterations/s
For		4536996		4535237		6051562

- die Suite besteht aus 3 Sektionen
  - Section1 prüft arithmetische Operationen, Zuweisungen und Schleifen
  - Section2 behandelt typische Anwendungsgebiete
  - Section3 führt spezielle komplexe Funktionen aus

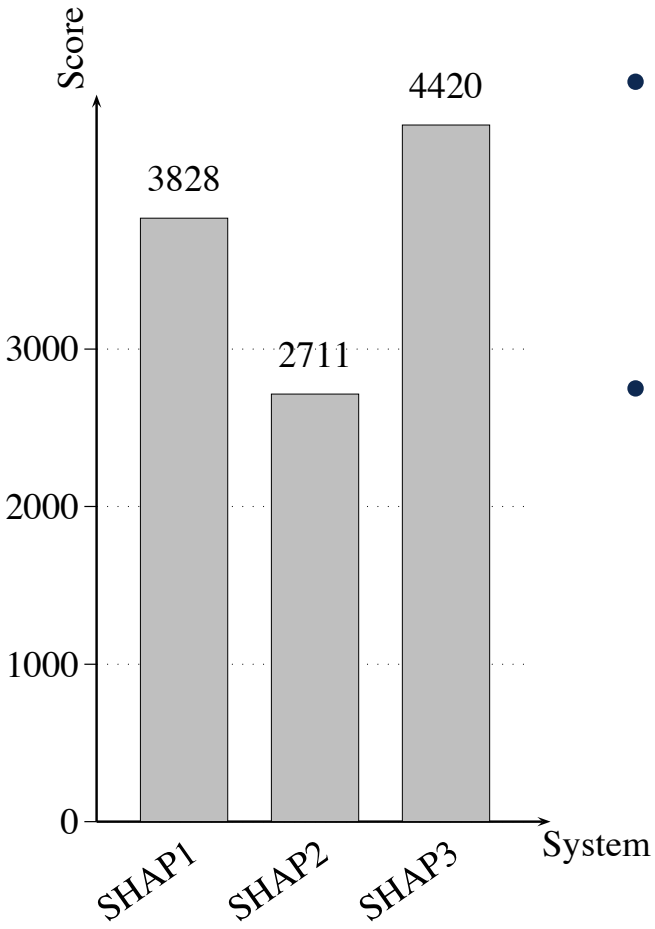
Java Grande Forum Benchmark Suite

Benchmark	SHAP1		SHAP2		SHAP3	
Section2:Crypt	ms	byte/s	ms	byte/s	ms	byte/s
Kernel:SizeA	28457	1054	28773	1042	21394	1402
Section2:HeapSort	ms	items/s	ms	items/s	ms	items/s
Kernel:SizeA	1045	14354	1508	9946	921	16286
Section3:RayTracer	ms	Objects/s	ms	Objects/s	ms	Objects/s
Init:SizeA	68	941	71	901	52	1230

- Laufzeitauswertung und Bearbeitungsgeschwindigkeit

- SHAP3 schließt bei allen Tests als beste Konfiguration ab
- die Begründung ist in der höheren Systemtaktung zu suchen

EEMBC



- 5 verschiedene Tests (chess, parallel, crypto, png, xml)
- Quellcode musste angepasst werden an:
  - Methodengröße
  - Arraygrößen
  - Objektreferenzen
- PNG-Test kann nur unkomprimierte Bilder mit einer Größe von 10x10 Pixel verarbeiten

	chess	parallel	crypto	PNG	kxml	GrinderMark Score
SHAP1	879	1377	1364	222988	2229	3828
SHAP2	638	894	897	177930	1608	2711
SHAP3	1025	1512	1511	271255	2657	4420

Zusammenfassung

- einige Java-Benchmarks verfügbar
- viele Benchmarks schwer realisierbar für eingebettete Systeme
- Quellen teilweise nicht offen oder nicht bestätigt
- wenige Benchmarkergebnisse zum Vergleich

- mehrere Möglichkeiten zur Java-Bytecode Ausführung
  - unterschiedliche Optimierungsmöglichkeiten

- wenige Informationen der Hersteller/Entwickler über die Hardware

UCSD Benchmark - Ergebnisse

Benchmark	SHAP1	SHAP2	SHAP3	Sys. A [18]	Sys. B [18]
Empty loop iterated 1000000 times	0.349	0.340	0.255	0.058	0.640
Added 1000000 ints in loop	0.769	0.862	0.610	0.278	1.397
Multipled 1000000 ints in loop	0.769	0.863	0.610	0.135	1.418
Assigned to 1000000 array ints	0.750	1.000	0.500	0.271	1.298
1000000 object int field accesses	0.749	0.944	0.616	0.111	1.169
1000000 method calls in same object	1.270	1.489	0.980	0.231	1.681
1000000 method calls in other object	1.467	2.147	1.294	0.219	1.832
Threw and caught 1000000 exceptions	2.164	2.970	1.863	6.295	2.456
3 threads, switched 10000 times each	0.121	0.147	0.098	0.290	0.407
Cumulative runtime	8.855	11.087	7.328	(25.918)	(23.530)

Java Grande Forum Benchmark Suite

Benchmark	SHAP1		SHAP2		SHAP3	
	Zeit	Ergebnis	Zeit	Ergebnis	Zeit	Ergebnis
Section1:Arith	ms	add/s	ms	add/s	ms	adds/s
Add:Int		6447855		6447855		8617715
Add:Long		2180463		2177912		2914265
Section1:Arith	ms	divides/s	ms	divides/s	ms	divides/s
Div:Int		1187246		1185871		1590062
Div:Long		4571		4557		6124
Section1:Assign	ms	assign/s	ms	assign/s	ms	assign/s
Same:Scalar:Class		2259238		1769636		2725397
Same:Array:Class		872008		588911		979998
Other:Scalar:Class		2256003		1769636		2722861
Other:Array:Class		858916		581322		969513
Section1:Loop	ms	iterations/s	ms	iterations/s	ms	iterations/s
For		4536996		4535237		6051562
Section2:Crypt	ms	byte/s	ms	byte/s	ms	byte/s
Kernel:SizeA	28457	1054	28773	1042	21394	1402
Section2:HeapSort	ms	items/s	ms	items/s	ms	items/s
Kernel:SizeA	1045	14354	1508	9946	921	16286
Section3:RayTracer	ms	Objects/s	ms	Objects/s	ms	Objects/s
Init:SizeA	68	941	71	901	52	1230