# JavaGrande

www.javagrande.org

Bernhard Haumacher, Michael Philippsen
Universität Karlsruhe
Institut für Programmstrukturen

## Überblick

- Grande-Anwendungen
- Das JavaGrande Forum
- Diskutierte Probleme
- Lösungsansätze in den Bereichen
  - Numerik
  - verteiltes paralleles Rechnen
- Grande-Anwendung, was nun?

2

## Grande Anwendungen

... angesiedelt im wissenschaftlichen, ingeneurmäßigen oder kommerziellen Rechnen

... bei Modellbildung, Simulation und Datenauswertung

... sind komplex, rechen-, daten- und/oder Ein-/Ausgabe-intensiv

3

## Grande Anwendungen

- Benötigen ...
  - höchste Leistung
  - Parallelität
  - verteiltes Rechnen
- In der Java-Gemeinde bisher nur geringer Nutzeranteil

4

## Java Grande Forum (JGF)

- Gegründet: März 1998
- Offenes Forum: Industrie, US-Regierung, universitäre Beteiligung
  - Academic Coordinator:
    Geoffrey C. Fox (Syracuse)
  - Secretary:
    George Thiruvathukal (Chicago)
  - Industry Coordinator:
    Siamak Hassenzadeh (Sun)

5

## Java Grande Forum (JGF)

- Workshops & Konferenzen
  - Syracuse Dezember 96
  - PLDI Las Vegas Juni 97
  - Palo Alto Februar 98
  - EuroPar Southampton September 98
  - HPCN Amsterdam April 99
  - IPPS San Juan April 99
  - ACM JavaGrande Konferenz 12.-13. Juni
  - JavaOne 15.-18. Juni 99
  - ICS'99 Rhodos 19.-20. Juni 99

6

## Java Grande Forum (JGF)

- Besteht aus zwei Arbeitsgruppen
  - Numerics Working Group
    - Ron Boisvert
    - Roldan Pozo (NIST)
  - Distributed Computing Working Group
    - Dennis Gannon (Indiana)
    - Denis Caromel (INRIA)

7

## Ziele von Java Grande

- Schwächen von Sprache und Laufzeitumgebung für Grande-Anwendungen aufzeigen
- Bedarf gegenüber Sun bündeln
- Java für Grande-Anwendungen verwendbar machen (Außenwirkung)
- Standardisierung von Schnittstellen und Bibliotheken für Grande-Anwendungen (JGF intern)

8

## Warum Grande mit Java?

- Infrastruktur für verteiltes Rechnen
- Sprachdesign (oo, GC, multi-threaded,...)
- Portabilität: Java Virtual Machine
- Reicher Fundus an Bibliotheken
- Integrierte Entwicklungsumgebungen
- Breite Akzeptanz
- Java wird gelehrt und gelernt

9

## Überall Java in Grande?

- Benutzerschnittstelle (Applet)
  - graphische Datenaufbereitung
- Bindeglied zwischen Berechnung, Ablaufsteuerung, Datenbank und Ausgabe (Mittlere Schicht)
- Sprache für die parallele & verteilte Programmierung im Cluster (Hochleistungs-Backend)

10

## Probleme mit Java & Grande

- Java nicht für Grande-Anwendungen entworfen
  - verlangt ist ***Geschwindigkeit*** & Portabilität
  - keine Sprachunterstützung für numerisches Rechnen
- Keine numerischen Bibliotheken
  - Henne & Ei Problem

11

## Numerics Working Group

- Effiziente komplexwertige Arithmetik
- Effiziente mehrdimensionale Felder
- Leichtgewichtige Klassen
- Operator - Überladung
- Java-Fließkomma-Arithmetik
- Mathematische Bibliotheken
  - FFT, lineare Algebra, ...

12

## Numerics Working Group

- John Brophy, Visual Numerics
- Sid Chatterjee, University of North Carolina
- David S. Dixon, Mantos Consulting
- Steve Hague, NAG
- Lennart Johnsson, University of Houston
- William Kahan, University of California, Berkeley
- Cleve Moler, The MathWorks
- Jose Moreira, IBM
- Kees van Reevwijk, Technical University, Delft
- Keith Seymour, University of Tennessee
- Nik Shaylor, Sun
- Marc Snir, IBM

13

## Komplexwertige Arithmetik

- Forderungen …
  - so effizient und bequem wie float und double
- Komplex als normale Klasse liefert
  - inakzeptablen Objekt-Mehraufwand
  - falsche Semantik von = und ==
  - undurchschaubare Methodenaufrufe der Form a.assign(b.times(c).plus(d))

14

## Komplexwertige Arithmetik

- IBM-Ansatz
  - Klasse Complex
  - Nativer Übersetzer
  - Spezialfall im Übersetzer fest verdrahtet
    - Einbetten
    - keine Heap-Allokation
    - kein Objekt-Mehraufwand
  - http://alphaworks.ibm.com/tech/ninja

15

## Komplexwertige Arithmetik

- *CJ:* Complex Numbers in Java
  - Uni Karlsruhe
  - Neuer Basistyp complex
  - mit allen Operatoren + - * /...
  - Übersetzung durch Präprozessor
    - Auflösen von *complex* in zwei *double*
    - Transformation der Operatoren nach Java
  - reine Java-Lösung
  - http://wwwipd.ira.uka.de/JavaParty/

16

## Mehrdimensionale Felder

- Forderungen …
  - Übersetzer-optimierbarer Feldzugriff
  - Keine Bereichstests zur Laufzeit
  - Verläßliche Feldanordnung ➔ Auswahl des besten Algorithmus
- Probleme bei mehrdim. Java-Feldern
  - zerstückelt mit Zeilen-Aliasen
  - Klasse zur Kapselung verursacht unlesbaren und ineffizienten Code

17

## Mehrdimensionale Felder

- IBM
  - Klassen für
    0D-, 1D-, 2D- und 3D- Matrizen
  - native Übersetzung
  - http://alphaworks.ibm.com/tech/ninja/
  - http://www.research.ibm.com/ninja/

18

## Eine allgemeinere Lösung

- Leichgewichtige Klassen
  - eingeschränkte Klassen mit Wertsemantik
  - wenig Mehraufwand
  - können eingebettet werden
- Operator-Überladung
  - wenigstens für existierende Operatoren: Arithmetik, Vergleich, Zuweisung, Indizierung

19

## Fließkomma-Arithmetik

- Forderungen ...
  - *immer* akzeptable Geschwindigkeit
  - nur *bei Bedarf*:
    - Hochgeschwindigkeit
    - plattformübergreifend reproduzierbare Ergebnisse
  - Zugriff auf IEEE-Merkmale

20

## Fließkomma-Arithmetik

- Ist-Zustand / Probleme
  - *run anywhere, get the same answers everywhere*
  - keine Ausnutzung proprietäter Fließkomma-Hardware
    - 80-bit Register auf x86 ⇒ 2-10x langsamer
  - *fused multiply-add* verboten
  - gängige Übersetzer-Optimierungen verboten (z.B. Assoziativität ausnutzen)

21

## Sun: Fließkomma-Vorschlag

Lange diskutiert, erste Reaktion:

- **widefp** (standard)
  - gleichberechtigte Verwendung von float- und double-extended
  - *kein* FMA
  - *eingeschränkte* HW-Unterstützung
- **strictfp**
  - bisherige Konventionen

22

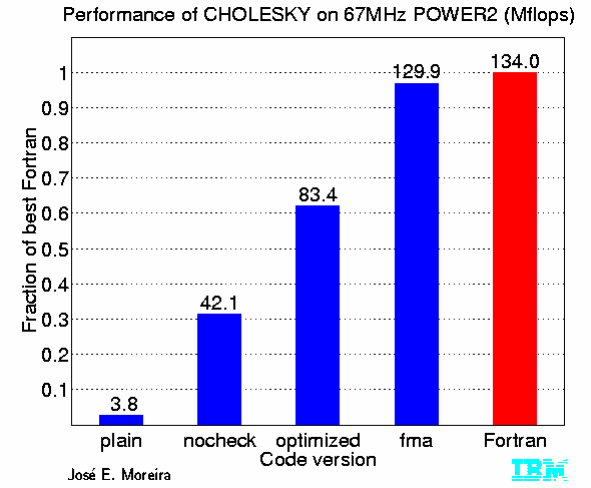
# JGF: Fließkomma-Vorschlag

Einführung von 3 Modi

- **standard**
  - 15-bit Exponent,
  - fused multiply-add
- **strictfp**
  - bisherige Konventionen
- **associativefp**
  - Assoziativität ausnutzen, ... mehr ?

23

# IBM Fließkomma-Ergebnisse




24

## Weitere FP-Vorhaben

- "reproduzierbare" mathematische Funktionen
  - Java-Version von fdlibm für strictfp
- Standard zur Beeinflussung von
  - IEEE Fließkomma-Flags
  - IEEE Rundungsmodi
- Implementation von IEEE-Funktionen

25

## Klassen-Bibliotheken

- Komplexe Zahlen (IBM, VNI)
- Mehrdimensionale Felder (IBM)
- Lineare Algebra (MathWorks & NIST)
- In Arbeit
  - Intervalle
  - FFT
  - multiprecission

26

## SciMark Benchmark

- Java Benchmark-Applet für Numerik
- kombinierte Ergebnisse aus
  - FFT (komplex)
  - Gauss-Seidel Relaxation
  - Monte Carlo Integration von $e^{-x^2}$
  - Multiplikation dünnbesetzter Matrizen
  - LU-Faktorisierung dichter Matrizen mit Pivotisierung
- normalisiert: SPARC 10, Netscape 4.04

27

## Distributed Computing W.G.

- schnelles RMI
- JavaParty - Transparente entfernte Objekte in Java
- MPI für Java
- Benchmark Initiative
  - http://www.epcc.ed.ac.uk/research/javagrande/benchmarking.html
- Seamless Computing Standards

28

## Distributed Computing W.G.

- George Crawford, MPI Software Technology
- Vladimir Getov, University of Westminster, UK
- Piyush Mehrotra, ICASE
- Michael Philippsen, University of Karlsruhe, Germany
- Omer Rana, University of Wales, Cardiff, UK
- Tony Skjellum. MPI Software Technology
- Henry Sowizral, Sun
- Martin Westhead, EPCC, University of Edinburgh, UK

29

## Remote Method Invocation

RMI

- entfernter Methodenaufruf
- hat Forschung im Bereich verteiltes Rechnen beflügelt
- Geschwindigkeit für Cluster Computing nicht ausreichend
- Programmiermodell erfüllt nicht alle Wünsche

30

## Remote Method Invocation

- Nicht entworfen für eng gekoppelte Netzwerke
- Geschwindigkeitsprobleme
  - aufwendige, langsame Serialisierung
  - Verluste bei interner Verwaltung
  - Keine flexible Transportschicht
    - unzureichende Unterstützung von Hochgeschwindigkeitsnetzwerken (ParaStation, Myrinet, ATM, ...)

31

## Schnelle Serialisierung

- Entwickelt an der Uni Karlsruhe
- Ersetzt JDK-Serialisierung unter der Annahme eng gekoppelter Netzwerke
- Zusammen mit RMI verwendbar
- ca. um Faktor 10 schneller als der Standard im JDK
- Verfügbar

32

## KaRMI: Schnelles RMI

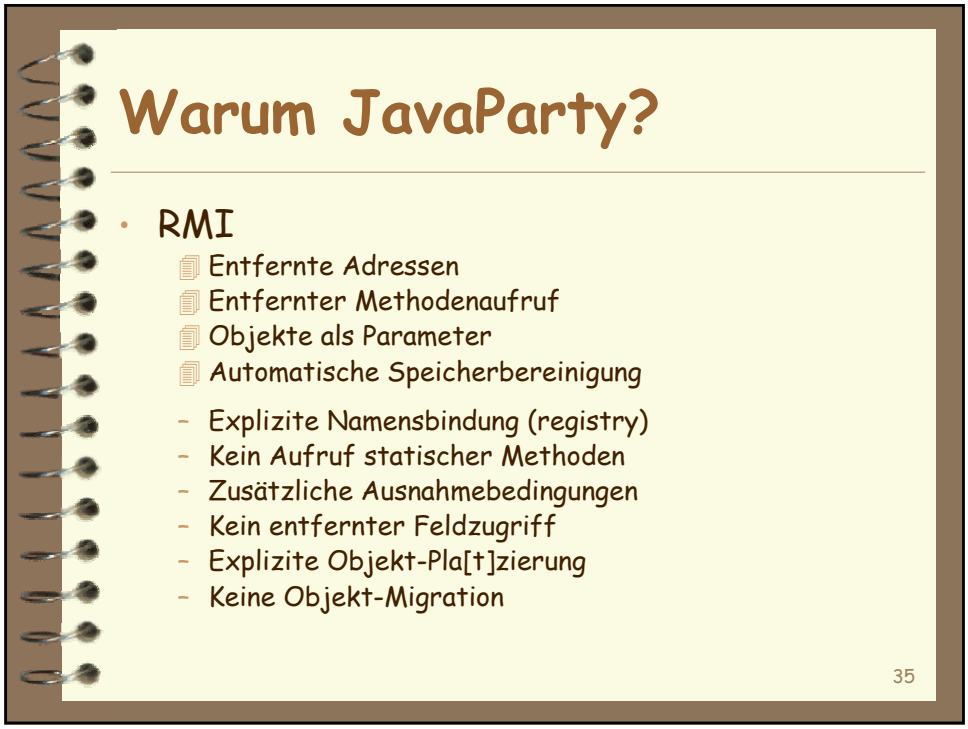
- Entwickelt an der Uni Karlsruhe
- Anpaßbar an Hochgeschwindigkeits-Netze durch Komponenten-Austausch
- Schlanke Implementierung
- Verwendet schnelle Serialisierung
- Transportschicht austauschbar, für ParaStation und Ethernet vorhanden
- 80µs für entfernten Methodenaufruf

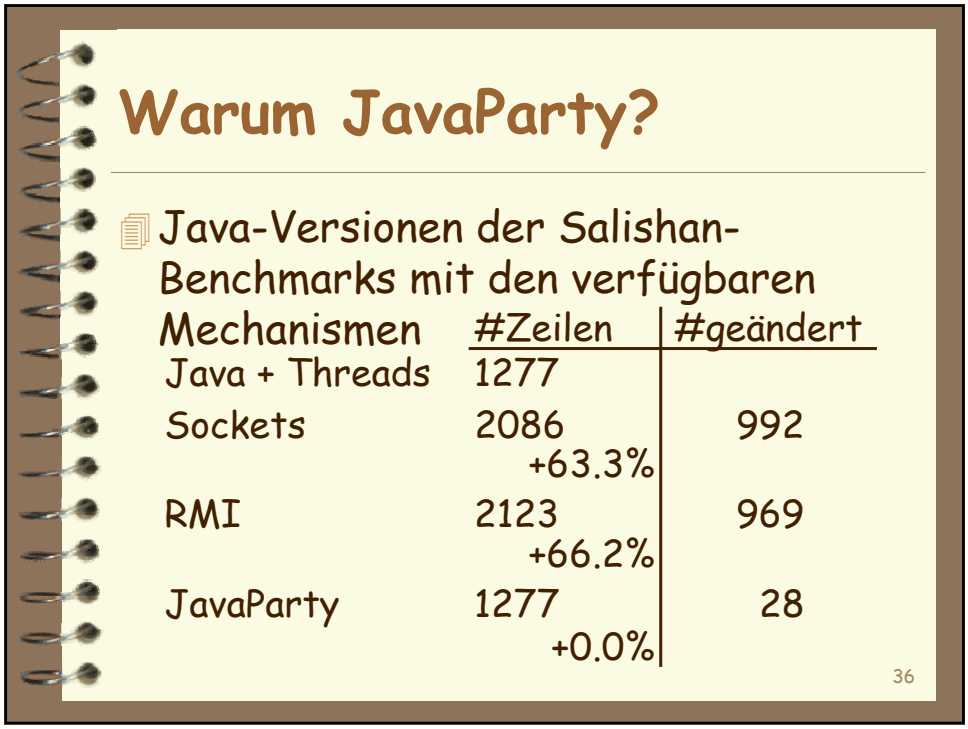
33

## JavaParty

- Java-Mechanismen unzureichend für das Programmieren von Rechnerbündeln
  - parallel -> Threads
  - verteilt -> ???
- Sockets
  - Keine entfernten Adressen
  - Kein entfernter Methodenaufruf
  - Keine automatische Serialisierung
  - Keine automatische Speicherbereinigung

34


Warum JavaParty?
RMI
Entfernte Adressen
Entfernter Methodenaufruf
Objekte als Parameter
Automatische Speicherbereinigung
Explizite Namensbindung (registry)
Kein Aufruf statischer Methoden
Zusätzliche Ausnahmebedingungen
Kein entfernter Feldzugriff
Explizite Objekt-Pla[t]zierung
Keine Objekt-Migration
35


Warum JavaParty?
Java-Versionen der Salishan-Benchmarks mit den verfügbaren Mechanismen
#Zeilen
#geändert
Java + Threads
1277
Sockets
2086
+63.3%
992
RMI
2123
+66.2%
969
JavaParty
1277
+0.0%
28
36

## JavaParty

- Java-ähnliches multi-threaded Programmiermodell für Rechnerbündel
- Illusion: verteilte virtuelle Maschine
- **entfernte Objekte** verteilt im Cluster
  - Parallelität durch **entfernte Threads**
  - **automatische** Lokalität & Lastbalance

37

## JavaParty

- Transparente entfernte Objekte mit (näherungsweise) Java-Semantik
  - entfernte Erzeugung (new)
  - entfernter Methodenaufruf (auch statische Methoden)
  - Keine zusätzl. Ausnahmebedingungen
  - entfernter Variablenzugriff
  - Synchronisation auf entf. Objekten
  - Objekt-Migration

38

## JavaParty

- Entfernte Threads
  - selbe Schnittstelle wie java.lang.Thread
  - auf entferntem Knoten startbar
- Lokalität & Lastbalance
  - Explizite Platzierung
  - Strategie-Objekt
  - Dynamisch zur Laufzeit
  - Statische Analyse beim Übersetzen

39

## MPI für Java

- RMI & Sockets: Client/Server-Modell
- MPI-1, MPI-2 Standard:
  - Nachrichtenaustausch
  - symmetrische Kommunikation
- benötigt: Java-MPI API Spezifikation
  - ermöglicht portable MPI-basierte Grande-Anwendungen in Java

40

# MPI für Java (Status)

- mpiJava
  - JNI -Wrapper für die C++-Bindung von MPI.
  - **http://www.npac.syr.edu/projects/pcrc/HPJava/**
- JavaMPI
  - automatisch erzeugte JNI-Wrapper für die C-Bindung von MPI
  - **http://perun.hscs.wmin.ac.uk/JavaMPI/**
- MPIJ
  - Reine Java-Implementierung von MPI.
  - Stark an die C++ Bindung angelehnt
  - **http://ccc.cs.byu.edu/DOGMA/**

41

# MPI für Java (Status)

- JMPI (angekündigt)
  - MPI Soft Tech Inc.
  - kommerzielles Produkt
- MPI-Spezifikation (Entwurf)
  - http://www.javagrande.org/reports

42

## Seamless Computing

- DATORR
  - Desktop Access to Remote Resources
  - http://www-fp.mcs.anl.gov/~gregor/datorr/
- Java-Rahmenwerk für „Computing Services“

43

## Seamless Computing

- Rechnen als verteilter Dienst
- Auffinden von Ressourcen
- Abfrage von Eigenschaften
- Web-Schnittstelle um jeden Job auf jedem Computer mit beliebiger Datenquelle auszuführen
- Schnittstelle zum Zusammenschalten mehrerer Computer für einen Job

44

## JavaGrande Europe

- Stärkung der Europäischen Rolle bei Standardisierungsbemühungen
- Zwei Gruppen
  - Metacomputing and Interoperability
    - Dr. Mark Baker, Prof. Denis Caromel
  - High Performance Applications
    - Prof. David Walker, Dr. Michael Philippsen
- **http://www.irisa.fr/EuroTools/Sigs/JavaGrande.top.html**

45

## Mitarbeit bei JavaGrande

- Mitarbeit an Standards, Schnittstellen & Bibliotheken
- Benchmarks beisteuern
- Interessenvertretung durch JGF wahrnehmen
- Nutzung des Forums
  - http://www.javagrande.org/

46