MICROSOFT .NET: EINE NEUE ENTWICKLUNGSPLATTFORM – AUCH FÜR NUMERISCHE PROBLEME?

Jan Linxweiler, Sören Freudiger

TU Braunschweig

mail@bau-ings.de
www.bau-ings.de

Microsoft .net development

Übersicht

- Microsoft .NET – Was ist das?
- .NET Framework
- Performance unter .NET
- FlowSim 2004
- Benchmark

.NET

Was ist das?

.NET is surrounded by too many buzzwords and generalities to be understandable. I'm not sure the company knows what .NET is, or whether anybody does.

John Dvorak

.NET

Was ist es <u>nicht</u>…

- *Betriebssystem*
  Weiterhin Windows 2003, XP, 2000, Me, 98
- *Programmiersprache*
  Weiterhin C++, VB, Java, Delphi, Fortran,….
  Neu: C# („C sharp“)

Microsoft .NET

- *.NET Web Services*
  - *Ständig verfügbare Internetdienste*
  - *Messenger, Code-Updates, Suchmaschienen*
- *.NET Enterprise Server*
  - *SQL Server 2000, Exchange Server 2000, …*
  - *Integration in .NET Anwendungen*
- ***.NET Framework***

….

.NET Framework

Komponenten und Richtlinien

um Anwendungen zu

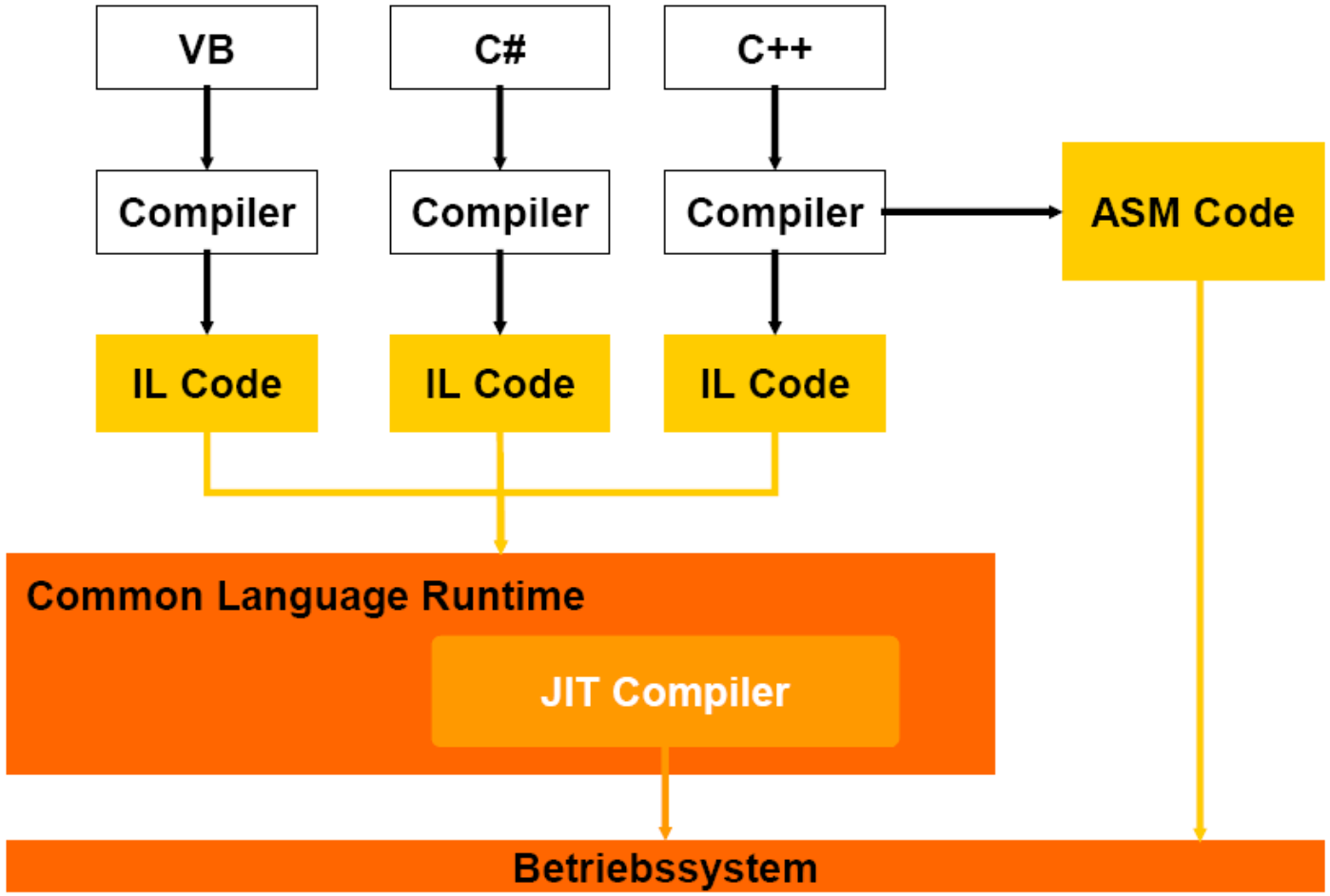
erstellen**, **kompilieren** und **auszuführen

.NET Framework

- *Laufzeitumgebung*
  - *Common Language Runtime (CLR)*
- *Klassenbibliothek*
  - *Framework Class Library (FCL)*
- *Typsystem*
  - *Common Type System (CTS)*
  - *Common Language Specification (CLS)*

Microsoft .NET Framework

Anwendungen erstellen, kompilieren** und **ausführen…



Common Language Runtime (CLR)

(Laufzeitumgebung – Kern von .NET)

- *JITer* - Just in Time Compiler
- *Garbage Collector* - automatische Speicherverwaltung
- *Type Checker* - Überprüfung Typkonvertierungen

…

- *Optimierungen*
  - *CPU-spezifisch*
  - *method inlining*
  - *array bounds check elimination*
  - *…*

Managed Code

Sämtlicher Code wird unter Aufsicht der Common Language Runtime ausgeführt.

- Runtime führt Sicherheitsüberprüfungen aus
- Runtime übernimmt Speicherverwaltung und Fehlerbehandlung
- Runtime führt Versionsüberprüfungen aus

Dieser Code wird als **Managed Code** bezeichnet

Common Type System (CTS)

(.NET Typsystem)

- Basis für Sprachunabhängigkeit
- Gemeinsames Typsystem für alle Sprachen
- Enthält alle verfügbaren Datentypen
- Zwei Arten: Werttypen und Referenztypen
- Definiert Vorschriften für eigene Typen

- Muß nicht von allen Sprachen unterstützt werden → CLS

Framework Class Library (FCL)

(.NET-Klassenbibliothek)

- Durchgehend objektorientiert - Basistyp *Object*
- Mehr als 2400 Klassen
- In Namespaces organisiert
- GUI - Windows Forms
- ASP.NET - Web Forms
- ADO.NET - Datenbanken

Longhorn: Win32-API → WinFX

Microsoft .NET Framework

System.Web

Services	UI
Description	HtmlControls
Discovery	WebControls
Protocols	
Caching	Security
Configuration	SessionState

System.Windows.Forms

Design	ComponentModel

System.Drawing

Drawing2D	Printing
Imaging	Text

System.Data

OleDb	SqlClient
Common	SqlTypes

System.Xml

XSLT	Serialization
XPath	Schema

System

Collections	IO	Security	Runtime
Configuration	Net	ServiceProcess	InteropServices
Diagnostics	Reflection	Text	Remoting
Globalization	Resources	Threading	Serialization

Allgemeine Vorteile

- Automatische Speicherverwaltung (keine Zeiger)
- Einheitliche (komfortable) Klassenbibliothek
- Plattformunabhängigkeit
- Sprachunabhängigkeit
- Ausführungssicherheit (Typüberprüfung, Range Checking, …)
- XCOPY-Deployment
- Hoher Entwicklungskomfort (Visual Studio .NET 2003, Intellisense)

Spezielle Vorteile

- *Automatische Speicherverwaltung*
- *CPU-spezifischer Code (JIT-Compiler)*
- *Typsicherheit*
- *Range Checking*
- *CLR - interne Optimierungsfunktionen*
- *Vorhandener Code kann integriert werden*

Nachteile

Zusätzlicher Verwaltungsaufwand durch die CLR

Ausführungsgeschwindigkeit!?

(Bindung an Windows)

Performance unter .NET

- *Verschiedene Arraytypen in .NET*
  (eindimensionale, sequenzielle und verschachtelte Arrays)
- *Zeiger in .NET*
  (direkter Zugriff auf den Speicher unter C# möglich)
- *Interoperabilität mit unverwaltetem Code*
  (z.B. vorhandenen Code einbinden)

Arrays in .NET

- *Eindimensionale Arrays*
  - *int[ ] arr = new int[3];*
  - *Optimierung durch JITer möglich*
- *Sequenzielle Arrays (rectangled arrays)*
  - *int[,] arr = new int[3,3];*
  - *Zusammenhängend in Speicher abgelegt → hohe Datenlokalität*
  - *Zugriff kann vom JITer nicht so hoch optimiert werden*
  - *Verwenden bei diagonalem (unzusammenhängendem) Arrayzugriff*
- *Verschachtelte Arrays (jagged arrays)*
  - *int[ ][ ] arr = new int[3][3];*
  - *Array aus eindimensinalen Arrays → Vorteile eindimesionaler Arrays*
  - *Unzusammenhängend in Speicher abgelegt → geringe Datenlokalität*
  - *Verwenden bei sequenziellem Arrayzugriff*

Zeiger in .NET

- *Nur in C# (und Mangaged C++)*
- ***Direkter Zugriff auf den Speicher***
  → *Keine Sicherheitsüberprüfungen durch CLR*
  - *Typsicherheit*
  - *Range Checking*
  - *Schlüsselwort: „unsafe“*
- *Schlüsselwort: „fixed“*
  → *Einschränkung der Optimierungsfunktion der CLR*
- *In C# rudimentäre Zeigerarithmetik*
  → *Schlechte Lesbarkeit des Quelltextes*
  - *rho=pf[m_ny*9*i+9*j+0]+pf[m_ny*9*i+9*j+1]+...*
  - *(rho = m_f[i,j,0]+m_f[i,j,1]+m_f[i,j,2]+…)*

Managed C++

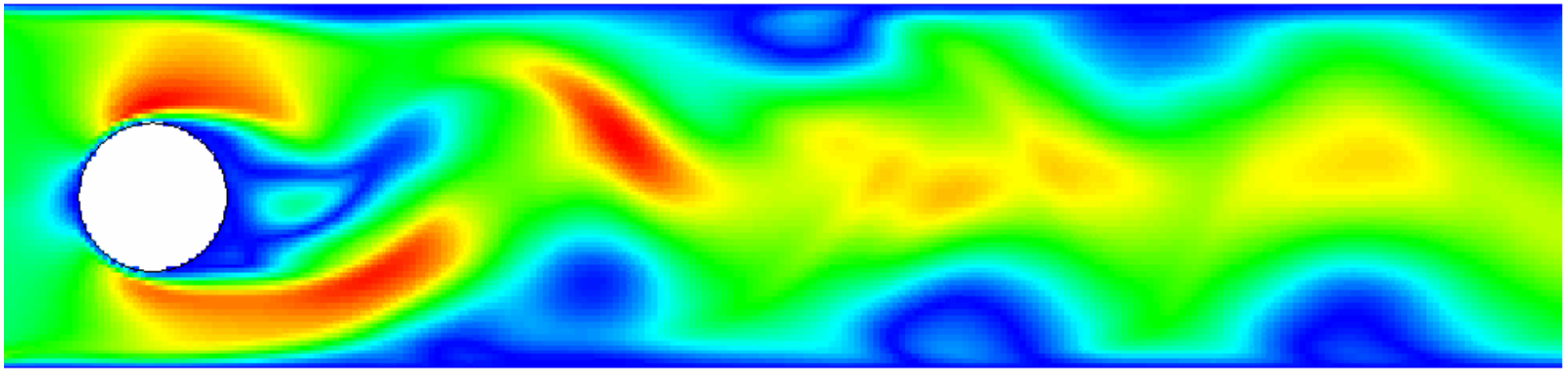
- *Erweiterung der Sprache C++*
  (Problem: C++ ANSI/ISO-Standard → schlechte Lesbarkeit)
- *Sprache der Wahl für Performance in .NET*
  - Compiler erzeugt schnellsten IL-Code
  - Alles ist möglich…!!!
- *Mischen von managed und umanaged Code*
- *Integration von vorhandenem Code (Wrapper-Klassen)*
  (Problem: Kontextwechsel geht zu Lasten der Performance)
- *C++/CLI wird ISO-Standard werden*

FlowSim 2004 .NET

- *Interaktiver Strömungssimulator*
- *Lattice-Boltzmann-Methode*
- *Schwach kompressible, laminare Strömungen*
- *Uniforme 2D-Gitter*

FlowSim 2004 .NET

Interaktiver Strömungssimulator auf Basis der Lattice-Boltzmann-Methode



Lattice-Botzmann-Methode

Lattice-Boltzmann-Gleichung (LBGK)

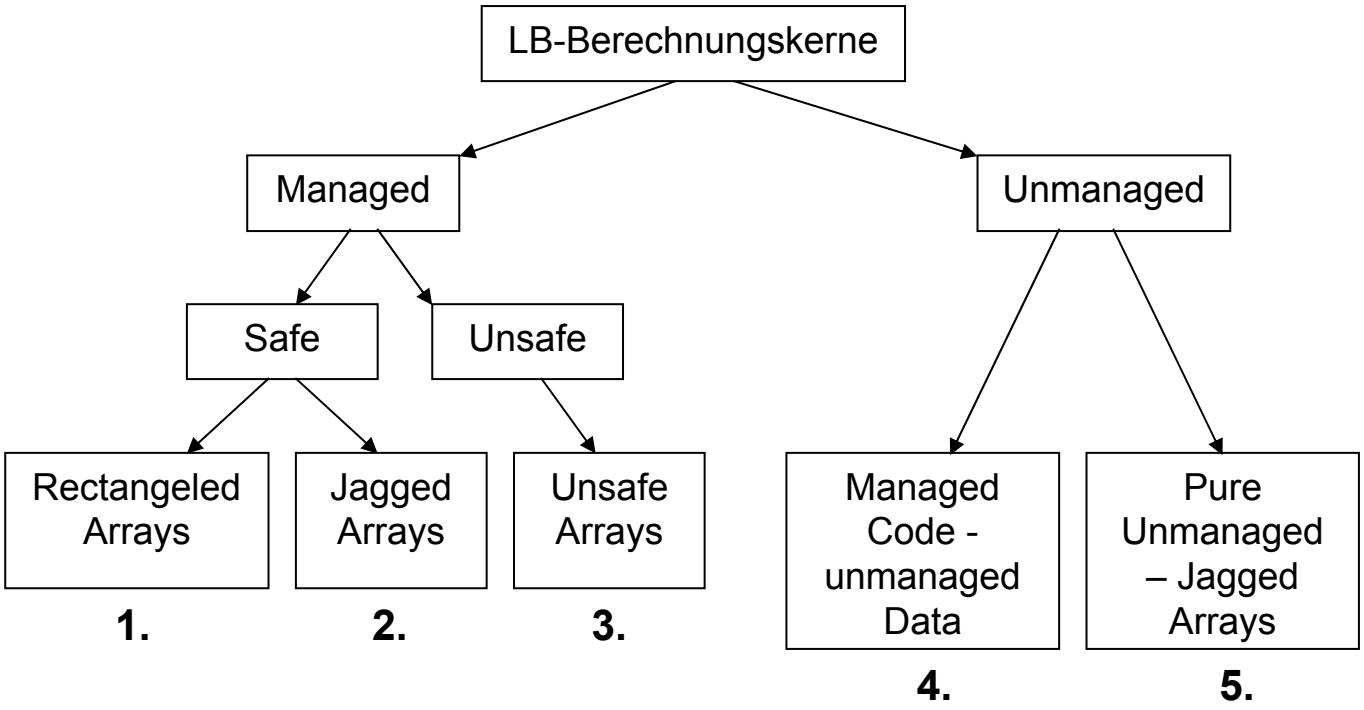
$$f_i(\vec{x}+\vec{e}_i\cdot\Delta t, t+\Delta t) - f_i(\vec{x},t) = \frac{\Delta t}{\tau}[f_i(\vec{x},t) - f_i^{eq})]$$

Kollision: $f_i^{new}(\vec{x},t) - f_i(\vec{x},t) = \Omega_i$

Propagation: $f_i(\vec{x}+\vec{e}_i\cdot\Delta t, t+\Delta t) = f_i^{new}(\vec{x},t)$

```
Init Matrices geo, ux, uy, rho;
Init Distributions f;
Calc tau;
Loop                              //Berechnung
{
   collide;                       //Kollision
   propagate;                     //Propagation
   if (TimeStepsCalculated)       //Wenn X Berechnungsschritte
       OnTimeStepsCalculated;     //Benachrichtigung der GUI
}
```

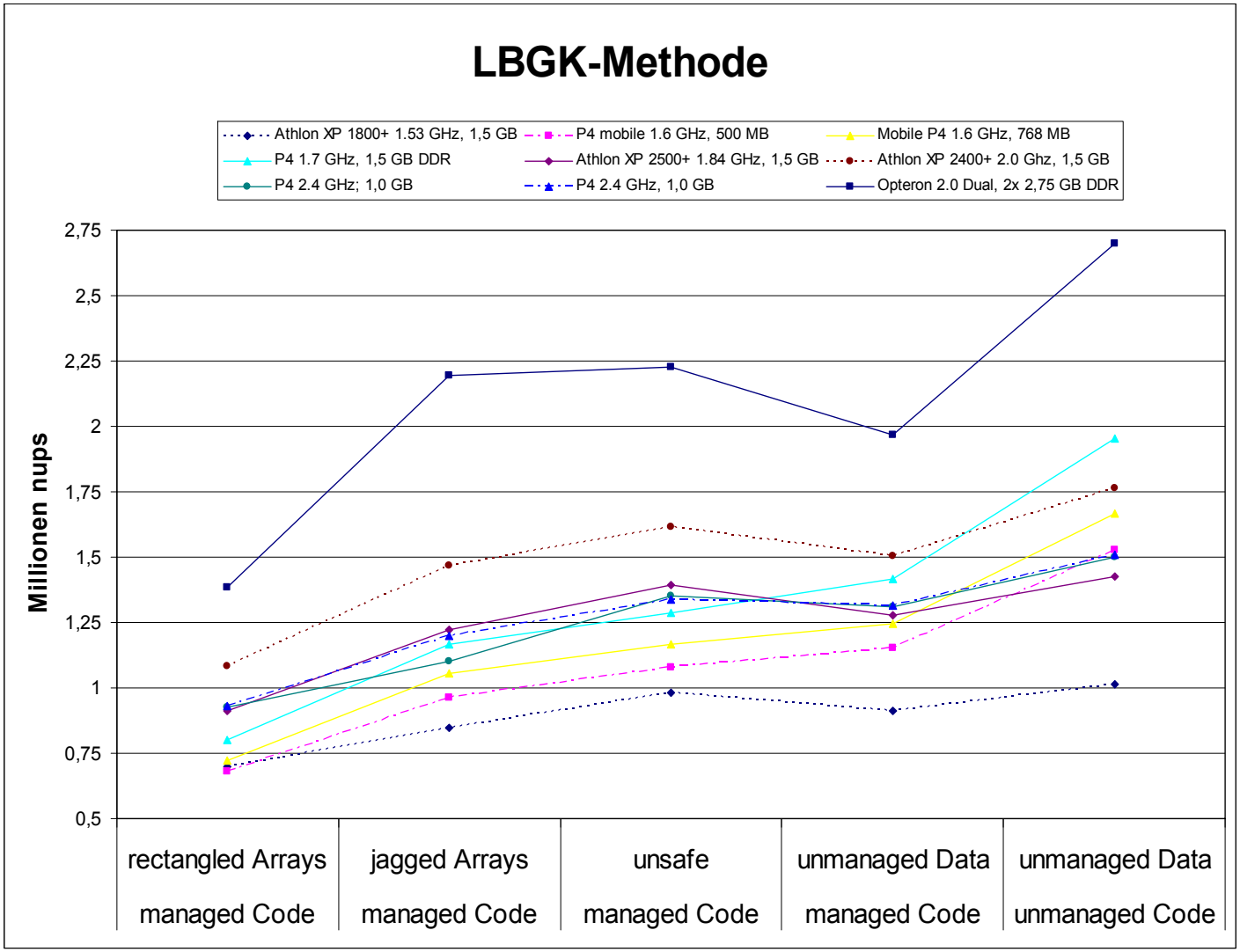
Struktur der Berechnungskerne



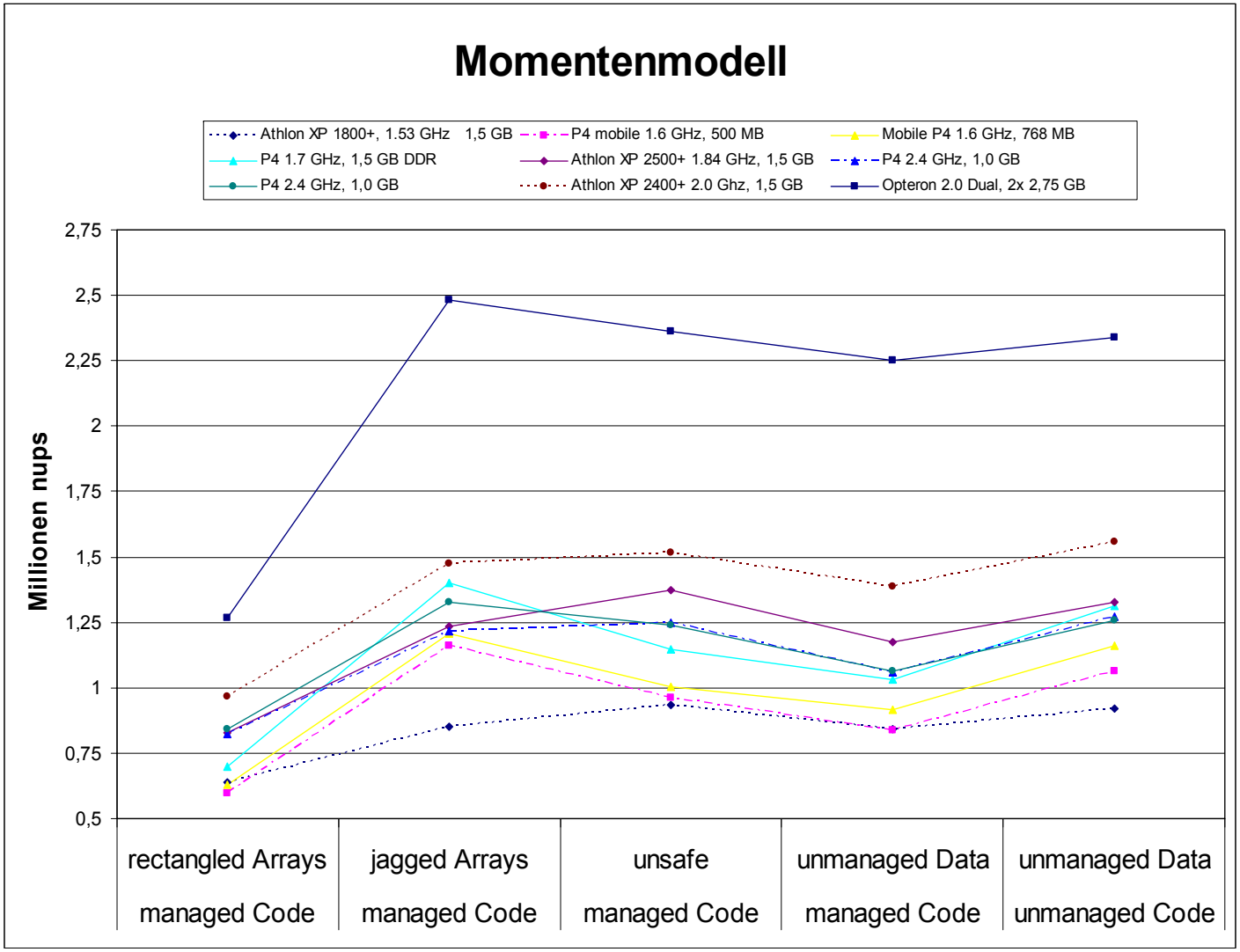
Benchmark

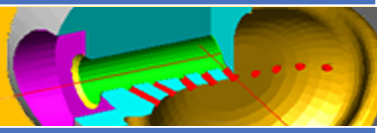
- *30.000 Knoten*
- *5 Berechnungskerne*
- *9 Testrechner*
- *LBGK & Momentenmodell*

FlowSim 2004 .NET



FlowSim 2004 .NET

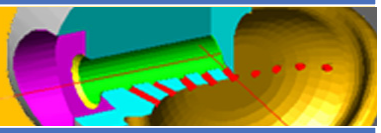




Ergebnisse

- ***Nativer Code*** *i.A. beste Performance*
- *Bestehenden Code integrieren mit Managed C++*

- ***Managed Code*** *u.U. sehr gute Ergebnisse + hohen Komfort*
- *Wahl der Arraytypen beachten !!!*
- *Evtl. Verwendung von Zeigern (in C# unverhältnismäßig viel Aufwand)*
- *Managed C++ Sprache der Wahl für Performance unter .NET*



Ausblick

- *JIT-Compiler werden in Zukunft statische Compiler übertreffen*
- *Ausführungsgeschwindigkeit in .NET 2.0 verbessert*
- *C++/CLI wird ISO-Standard*

Ende