Benchmarking CLI for High-Performance Computing

Werner Vogels
Cornell University
Shortened version of talk

- Why we are doing this
- What are we doing
- When do we expect results

- Time to talk about sscli.net
Why?

- Performance is important, always.
- Different levels
  - Micro: primitive performance
  - Macro: compound performance
  - Application: interaction between compounds

- In the end *only* Application performance is important
- Performance directly relates to scalability
Why? - II

- Driving the understanding of how systems work.
- Comparison with other virtual machines.
- Feedback to the developers about what is important.
What?

- Application area: High-Performance Scientific and Financial Computing
- Port and developed a collection of benchmarks that investigate whether the CLI is usable in HPC
- Comparison:
  - Different CLI implementations
  - Other virtual machines
Interlude: The Java Problems

- No true multi-dimensional arrays
- New types, such as complex numbers, are not as efficient as the primitive types
- New types with just value semantics are not possible
- Exploit FP Hardware
- Operator overloading
More on Java

- IBM solved some of these issues with clever hacks
- But no generally accepted
- Java is becoming increasingly popular in the context of grid computing
What? - II

- Collection of known benchmarks
  - Linpack
  - Scimark
  - ByteMark
  - Java Grande Forum
- Some new benchmarks
  - Micro benchmarks (concurrency)
  - Financial computation benchmarks
The Benchmarks - Micro

- Arithmetic
- Assignments
- Object creation
- Array creation
- Method calls
- Loop overheads
- Exceptions
- Basic Math
Micro - II

- Matrix operations
  - Single dimension arrays
  - True multi-dimensional arrays
  - Jagged arrays
  - Homogeneous/heterogeneous types

- Concurrency
  - Thread operations
  - Synchronization costs
    - Methods vs blocks
    - Single/multiprocessors
We will not do:

- Serialization
- Garbage Collection
- Graphics
Arithmetic performance compared to .Net 1.1
Assignments

Assignments (absolute values)

- Assign: Same: Scalar: Local
- Assign: Same: Scalar: Instance
- Assign: Same: Array: Local
- Assign: Same: Array: Instance
- Assign: Other: Scalar: Class
- Assign: Other: Scalar: Instance
- Assign: Other: Array: Class
- Assign: Other: Array: Instance

- .Net 1.1
- rotor
- mono
- Java
Exceptions

Exception handling base

- :Exception:Throw
- :Exception:New
- :Exception:Method
Method invocations

Methods Invocations compared to .NET 1.1
Math Library

Math Library Performance compared to .Net 1.1
Math Library

Math performance

Math:AbsI
Math:AbsLong
Math:AbsFloat
Math:AbsDouble
Math:MaxI
Math:MaxLong
Math:MaxFloat
Math:MaxDouble
Math:MinI
Math:MinLong
Math:MinFloat
Math:MinDouble
Math:SinDouble
Math:CosDouble
Math:TanDouble
Math:AsinDouble
Math:AcosDouble
Math:AtanDouble
Math:Atan2Double
Math:FloorDouble
Math:CeilDouble
Math:SquareDouble
Math:ExpDouble
Math:LogDouble
Math:PowerDouble
Math:ToIntDouble
Math:ToIntFloat
Math:ToIntLong
Math:ToFloatDouble
Math:ToFloatLong
Math:ToFloatFloat
Math:ToFloatI
Math:RoundFloat
Math:RoundDouble
Math:RoundLong
Math:RoundToIntDouble
Math:RoundToIntFloat
Math:RoundToIntLong
Math:RoundToFloatDouble
Math:RoundToFloatLong
Math:RoundToFloatFloat
Math:RoundToFloatI
Math:IEEEReminderDouble
Math:IEEEReminderFloat
Math:IEEEReminderLong
Math:IEEEReminderI
Matrices

Matrix copies per second

- prim rect direct
- obj rec direct
- prim jagged direct
- obj jagged direct
- prim rect assec
- obj rec assec
- prim jagged assec
- obj jagged assec
The Benchmarks - Macro

- Lufact (Linpack)
  - Solves a NxN Linear system using LU factorization followed by a triangular solve. Basically measures dense array and basic FP performance

- Sparse Matrix Multiply
  - SMM in Compressed row
Macro – II

- **Series**
  - Computes the first $N$ Fourier coefficients of $f(x) = (x+1)^x$

- **FFT**
  - One dimensional forward transform of $N$ complex numbers
Macro - III

- Crypt
  - IDEA on $N$ Bytes
- SOR
  - Successive over-relaxation on a $NxN$ grid
- Heapsort
  - Sorts an array of $N$ integers
Applications

- Euler
  - Solves the time-dependent Euler equations for flow in a channel with a ‘bump’ on one of the walls
- MonteCarlo
  - Financial simulation using MC techniques to price products derived from the price of an asset
- MolDyn
  - N-body modeling of the behavior of N argon atoms under Lennard-Jones potential
- Search
  - Solves a game of connect-4 on a 6x7 board using a alpha-beta pruned search technique
- Raytracer
  - Rendering of 64 spheres in a $N \times N$ pixel space
When?

- Most of the benchmarks are converted to C#
- Currently validating the correctness
- SciMark for C# has been out since January 2003
- CLI-Grande out by this summer
- Concurrency benchmark by the fall
SciMark

- Small & Large memory Model
- Kernels:
  - MonteCarlo
  - LUfact
  - SOR
  - SparseMatMul
  - FFT
- Produce a single result: MFlops
SciMark - Small

SciMark results (April 2003)

- C++
- IBM 1.3.1
- .Net 1.1
- BEA Rockit
- Sun 1.4 Hotspot
- j#
- mono 0.23
- rotor
Individual Kernels
Compared to C

![Bar chart showing performance comparisons between different technologies and applications](chart.png)
CLI-Grande

- Project at sscli.net
- Contains SciMark for C#, J# & C