Pondering the Problem of Programmers’ Productivity

Are we there yet?

Arun Chauhan
Indiana University
The Big Picture

Human-Computer Interface
The Big Picture

Human-Computer Interface

programming languages

- MATLAB, Python, R, functional
- Java
- C++, C, Fortran
- machine / assembly

Domain-specific Languages, Arun Chauhan, Indiana University

Systems Seminar, 2004-11-04
Widely Recognized Problem

High-end computing is experiencing a full-blown productivity crisis, whose dimensions are only now beginning to be understood. We have entered a period of decreasing productivity as measured by the programming skill required . . . to make a lasting impact on an applications area by repeatedly developing new applications for the new machine.

—Anonymous contributor to HPCwire
DoD has Noticed!
High Productivity Computing Systems (HPCS)

Provide economically viable high productivity computing systems for the national security and industrial user communities with the following design attributes in the latter part of this decade:

- Performance
- Programmability
- Portability
- Robustness

Evolution of Languages

- Machine and assembly languages
- Procedural languages (Fortran, Pascal, C)
- Object-oriented languages (Smalltalk, Eiffel, C++, Java, C#)
- Functional / declarative languages (Scheme, Lisp, ML, Prolog)
Evolution of Languages

- Machine and assembly languages
- Procedural languages (Fortran, Pascal, C)
- Object-oriented languages (Smalltalk, Eiffel, C++, Java, C#)
- Functional / declarative languages (Scheme, Lisp, ML, Prolog)
- Domain-specific languages
  - MATLAB, Perl, Python, S+, Mathematica, Maple
  - MATLAB has over 500,000 licenses worldwide
Evolution of Languages

- Machine and assembly languages
- Procedural languages (Fortran, Pascal, C)
- Object-oriented languages (Smalltalk, Eiffel, C++, Java, C#)
- Functional / declarative languages (Scheme, Lisp, ML, Prolog)
- Domain-specific languages
  - MATLAB, Perl, Python, S+, Mathematica, Maple
  - MATLAB has over 500,000 licenses worldwide
  - also called “little languages”
  - APL, SQL
Evolution of Languages

- Machine and assembly languages
- Procedural languages (Fortran, Pascal, C)
- Object-oriented languages (Smalltalk, Eiffel, C++, Java, C#)
- Functional / declarative languages (Scheme, Lisp, ML, Prolog)
- Domain-specific languages
  - MATLAB, Perl, Python, S+, Mathematica, Maple
  - MATLAB has over 500,000 licenses worldwide
  - also called “little languages”
  - APL, SQL

Scripting Languages
A 25-year Old Problem

[T]he next revolution in programming will take place only when both of the following requirements have been met: (a) a new kind of programming language, far more powerful than those of today, has been developed and (b) a technique has been found for executing its programs at not much greater cost than that of today’s programs.

–John Backus, the “Father of Fortran”
A 25-year Old Problem

[T]he next revolution in programming will take place only when both of the following requirements have been met: (a) a new kind of programming language, far more powerful than those of today, has been developed and (b) a technique has been found for executing its programs at not much greater cost than that of today’s programs.

–John Backus, the “Father of Fortran”

Still not satisfactorily solved!
The Performance Gap

- Apple PowerBook G4 (667MHz)
- Sun SPARC v9 (1200MHz)

Running time (seconds)

- MATLAB interpreter
- ParaM / gcc compiled
- ParaM / CC compiled

Domain-specific Languages, Arun Chauhan, Indiana University
The Performance Gap

Matrix Multiply

- MATLAB interpreter
- ParaM / gcc compiled
- ParaM / CC compiled

- Speedup
- Apple PowerBook G4 (667MHz)
- Sun SPARC v9 (1200MHz)

Domain-specific Languages, Arun Chauhan, Indiana University
MATLAB Example

function mcc_demo
    x = 1;
    y = x / 10;
    z = x * 20;
    r = y + z;
function mcc_demo

x = 1;
y = x / 10;
z = x * 20;
r = y + z;

static void Mmcc_demo (void) {

  . . .
  mxArray * r = NULL;
  mxArray * z = NULL;
  mxArray * y = NULL;
  mxArray * x = NULL;
  mlfAssign(&x, _mxarray0_); /* x = 1; */
  mlfAssign(&y, mclMrdive(mclVv(x, "x"), _mxarray1_)); /* y = x / 10; */
  mlfAssign(&z, mclMtimes(mclVv(x, "x"), _mxarray2_)); /* z = x * 20; */
  mlfAssign(&r, mclPlus(mclVv(y, "y"), mclVv(z, "z"))); /* r = y + z; */
  mxDestroyArray(x);
  mxDestroyArray(y);
  mxDestroyArray(z);
  mxDestroyArray(r);

  . . .
}

Domain-specific Languages, Arun Chauhan, Indiana University
MATLAB Example

```matlab
function mcc_demo
    x = 1;
y = x / 10;
z = x * 20;
r = y + z;

static void Mmcc_demo (void) {
    . . .
    double r;
double z;
double y;
double z;
mLfAssign(&x, _mxarray0_); /* x = 1; */
mLfAssign(&y, mclMrdivide(mclVv(x, "x"), _mxarray1_)); /* y = x / 10; */
mLfAssign(&z, mclMtimes(mclVv(x, "x"), _mxarray2_)); /* z = x * 20; */
mLfAssign(&r, mclPlus(mclVv(y, "y"), mclVv(z, "z"))); /* r = y + z; */
mxDestroyArray(x);
mxDestroyArray(y);
mxDestroyArray(z);
mxDestroyArray(r);
. . .
}
```

Domain-specific Languages, Arun Chauhan, Indiana University

Systems Seminar, 2004-11-04
MATLAB Example

static void Mmcc_demo (void) {
    ...
    double r;
    double z;
    double y;
    double z;
    scalarAssign(&x, 1); /* x = 1; */
    scalarAssign(&y, scalarDivide(x, 10)); /* y = x / 10; */
    scalarAssign(&z, scalarTimes(x, 20)); /* z = x * 20; */
    scalarAssign(&r, scalarPlus(y, z)); /* r = y + z; */
    mxDestroyArray(x);
    mxDestroyArray(y);
    mxDestroyArray(z);
    mxDestroyArray(r);
    ...
}

function mcc_demo
    x = 1;
    y = x / 10;
    z = x * 20;
    r = y + z;
function mcc_demo

x = 1;
y = x / 10;
z = x * 20;
r = y + z;

static void Mmcc_demo (void) {
    ...  
double r;
double z;
double y;
double z;
x = 1; /* x = 1; */
y = x / 10; /* y = x / 10; */
z = x * 20; /* z = x * 20; */
r = y + z; /* r = y + z; */
    /* mexDestroyArray(x); */
    /* mexDestroyArray(y); */
    /* mexDestroyArray(z); */
    /* mexDestroyArray(r); */
    ...
}

Domain-specific Languages, Arun Chauhan, Indiana University
Inferring Types

- **type** ≡ <τ, δ, σ, ψ>
  - τ = intrinsic type, e.g., int, real, complex, etc.
  - δ = array dimensionality (or rank), 0 for scalars
  - σ = size (or shape), δ-tuple of positive integers
  - ψ = “structure” (or pattern) of an array

- **Examples**
  - x is scalar, integer
    ⇒ type of x = <int, 0, ⊥, ⊥>
  - y is 3-D 10 × 5 × 20 dense array of reals
    ⇒ type of y = <real, 3, <10,5,20>, dense>
Type-based Specialization

jakes: Type-specialized FORTRAN vs MATLAB

- Sun SPARC 336MHz
- SGI Origin
- Apple PowerBook G4 667MHz

- MATLAB 6.x
- FORTRAN
Fundamental Observation

• Libraries are the key in optimizing high-level scripting languages

\[ a = x \times y \Rightarrow a = \text{mclMtimes}(x, y) \]
Fundamental Observation

• Libraries are the key in optimizing high-level scripting languages

\[ a = x \times y \Rightarrow a = \text{mclMtimes}(x, y) \]

• Libraries practically define high-level scripting languages
  - high-level operations are often “syntactic sugar”
    * runtime libraries implement operations
  - a large effort in HPC is toward writing libraries
  - domain-specific libraries make scripting languages useful and popular

Domain-specific Languages, Arun Chauhan, Indiana University
Hierarchy of Libraries

MATLAB
- user-level libraries

C, Fortran
- run-time library

...
Domain Algebra

\sin^2(\theta) + \cos^2(\theta) \equiv 1

\tan^2(\theta) + 1 \equiv \sec^2(\theta)

\tan(\theta) \equiv \frac{\sin(\theta)}{\cos(\theta)}

\sin(2\theta) \equiv 2\sin(\theta)\cos(\theta)

\cos(2\theta) \equiv \cos^2(\theta) - \sin^2(\theta)

\ldots
Domain Algebra

\[ \sin^2(\theta) + \cos^2(\theta) \equiv 1 \]

\[ \tan^2(\theta) + 1 \equiv \sec^2(\theta) \]

\[ \tan(\theta) \equiv \frac{\sin(\theta)}{\cos(\theta)} \]

\[ \sin(2\theta) \equiv 2\sin(\theta)\cos(\theta) \]

\[ \cos(2\theta) \equiv \cos^2(\theta) - \sin^2(\theta) \]

\[ \ldots \]

\[ \ldots \text{and beyond} \]

\[ x = \sin(\theta) \]

\[ y = \cos(\theta) \equiv [x, y] = \sin \cos(\theta) \]
Domain Algebra

\[
\sin^2(\theta) + \cos^2(\theta) \equiv 1
\]

\[
\tan^2(\theta) + 1 \equiv \sec^2(\theta)
\]

\[
\tan(\theta) \equiv \frac{\sin(\theta)}{\cos(\theta)}
\]

\[
\sin(2\theta) \equiv 2\sin(\theta)\cos(\theta)
\]

\[
\cos(2\theta) \equiv \cos^2(\theta) - \sin^2(\theta)
\]

\ldots

\ldots and beyond

\[
x = \sin(\theta)
\]

\[
y = \cos(\theta) \equiv [x, y] = \sin\cos(\theta)
\]
Procedure Strength Reduction

\[
\text{for } i = 1:\text{N} \\
\quad \ldots \\
\quad f(c_1, c_2, i, c_3); \\
\quad \ldots \\
\text{end}
\]

\[
\text{f_init}(c_1, c_2, c_3);\\n\text{for } i = 1:\text{N} \\
\quad \ldots \\
\quad f\_\text{iter}(i); \\
\quad \ldots \\
\text{end}
\]
Speedup by PSR

- Speedups for top-level procedures in ctss relative to unoptimized

- Distribution of the total execution time among top-level procedures in ctss

Domain-specific Languages, Arun Chauhan, Indiana University

Systems Seminar, 2004-11-04
Open Issues

• Language to express identities
• Developing a cost metric
• Techniques to exploit identities
• Automatic discovery of identities
• Effect on compilation time
System Building

• Research through building systems
• Systems expose several unforeseen problems
• Real-life applications stress-test research ideas
• Give a real product to users—fruits of research
MATLAB Compilation System

Parser and Front-End

Type Infer. Engine

Spl’n Engine

Code Gen.

Annot.

Lib.

Opt. Specs

C
Why there isn’t a Parallel MATLAB

It doesn’t make good business sense for us to undertake fundamental changes in MATLAB’s architecture. There are not enough potential customers with parallel machines.

–Cleve Moler, Chairman and Chief Scientist
The MathWorks
## Improving the Performance of MATLAB

<table>
<thead>
<tr>
<th>Parallelization</th>
<th>Compilation</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>MATLAB</td>
</tr>
<tr>
<td>yes</td>
<td>MATLAB(^p), multiMATLAB, pMATLAB</td>
</tr>
<tr>
<td>no</td>
<td>FALCON, MaJIC, MATCH, Telescoping Languages, CONLAB, Otter, MENHIR</td>
</tr>
</tbody>
</table>

**proposed**
ParaM
Funded by Ohio Supercomputing Center

- Collaborators (P.I.s)
  - Ashok Krishnamurthy (Ohio Supercomputing Center)
  - P. Sadayappan (Ohio-State University)

- Technical Collaborators
  - Ken Kennedy (Rice University)
  - Jarek Nieplocha (Pacific-Northwest National Lab)
Challenges

- Performance bottlenecks in using the Global Array abstraction
- Automatic analysis to extract parallelism at suitable granularity
- Data distribution analysis
- Working with parallel libraries
- Extending to grid-computing environment
Research Problems

- Library identities
  - language, rewriting systems, theorem proving, databases

- Evolving library compilation system
  - databases, machine learning, AI

- Data-distribution based library-aware automatic parallelization
  - parallel and dist’d computing, runtime systems, algorithms

- Automatic adaptation to dynamic environments
  - AI, runtime systems, grid computing, algorithms
Conclusions

- Scripting or Domain-Specific Languages have gained enormous popularity
  - enable users to program easily in their domains of interest
- Key compilation technologies needed to take these languages beyond protoyping tools
- Library-centric approach shows a lot of promise
Conclusions

• Scripting or Domain-Specific Languages have gained enormous popularity
  - enable users to program easily in their domains of interest

• Key compilation technologies needed to take these languages beyond prototyping tools

• Library-centric approach shows a lot of promise

Not yet, but we are getting close!