Faster than C: Static type inference with Starkiller

Michael Salib
msalib@alum.mit.edu

Dynamic Languages Group
Computer Science & Artificial Intelligance Lab
Massachusetts Institute of Technology

March 24, 2004

This talk in 60 seconds

- There is no time. Ask me later!
- Python is slow
- Starkiller is a Python to C++ compiler
- It handles the entire language except one tiny, insignificant, morally deranged part
- Starkiller has one goal: lighting fast native code
- Of course, if we want speed, maybe C++ wasn't such a good idea...fortunately, it is very simple and easy to understand

This talk in 120 seconds

- Starkiller's heart is its type inferencer
- We will cut open its heart
- Starkiller has been implemented, but not finished
- Remember this number: 60
- Moral of the story: the past is dead

The Past

- Python is slow
- No it's not!
- Yes it is!
- This is why it is slow
- Everything else sucks

Python is slow

- I've done everything with Python
  - High speed network servers
  - Databases
  - Statistical natural language processing
  - Scientific computing
  - Signal and Image processing
  - AI type job schedulers
- And its been slow

Python is not slow!

- You're a heretic!
- Most apps spend all their time waiting
  - on a socket (network servers)
  - on a slow human (GUIs)
  - on Oracle (databases)
  - on disk IO (most things)
- Fast libraries written in C/C++
- Numeric!
- Die infidel, die!

Yes, Python is slow

- I've used all those lines myself
- I even believe them
- They're relevant most of the time
- But they don't change the fact that Python is slow
- Sometimes, straightforward Python code is much clearer and easier to write than fighting with Numeric
- For the 15% of apps where speed matters, pure Python can't do the job alone
- I don't want to use crappy C/C++

Why Python is slow

- It is not the VM: p2c showed that
- Layers of indirection
- Dynamic binding
- Dynamic dispatch
- No structure/size information
- Run time choice points foil the last 30 years of optimization research
- Speed comes from eliminating run time choice points

Other languages suck

- Java sucks beyond all measure and comprehension
- C++ and Java suffer the same performace problems as Python when it comes to dynamic dispatch
- Dynamic dispatch prevents the compiler from using all the cool optimizations like inlining
- Inlining is the canary in the coal mine: if you can't inline, you probably can't do loop hoisting, strength reduction, etc.

The Present

- Starkiller type inference
  - nodes and constraints
  - functions and templates
  - objects and classes
  - external code
- Status
- Results

Starkiller

- Compiling to C++ is not enough (cf p2c)
- Need static type inference to eliminate dynamic binding and dispatch
- Starkiller compliments rather than replaces CPython
- Covers the entire language except eval, exec, and dynamic module loading
- Not all run time choice points can be eliminted, but many can

Starkiller type inference

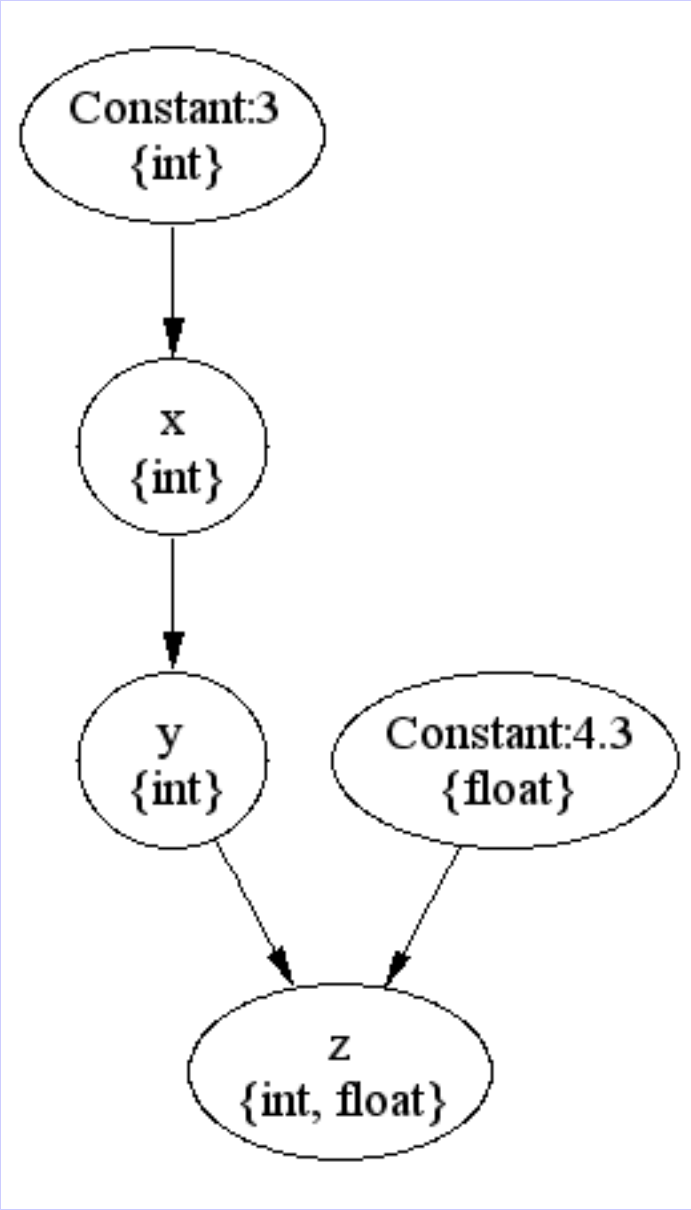
- Based on Ole Agesen's Cartesian Product Algorithm (see his Stanford thesis)
- Represent Python programs as dataflow networks
- Nodes correspond to expressions and have a set of concrete types those expressions can achieve at runtime
- Constraints connect nodes together and enforce a subset relation between them
- Types flow along constraints

Ex-girlfriends say I'm insensitive

- Starkiller's type inference algorithm is flow-insensitive
- It has no notion of time
- Code like x = 3; doSomething(x); x = 4.3; doSomething(x) will suffer loss of precision
- I don't care. I'm insensitive, remember?

Type inference in action

- A simple example
  - x = 3
  - y = x
  - z = y
  - z = 4.3



Functions and Templates

- Parametric polymorphism (same function with different argument types) reduces precision
- We regain precision by taking cartesian product of argument type list and analyzing one template for each monomorphic argument list
- Given polymorhic calls max(1, 2) and max (3.3, 4.9), we analyze templates for (int, int), (float, int), (int, float), and (float, float)

Functions and Definitions

- A Python function defintion creates a first class object at runtime
- Function objects can capture variables defined in their lexical parent(s)
- Starkiller models function definition using a function definition node that has constraints from all default args and expressions the function closes over
- The definition node takes the cartesian product and generates monomorphic function types

Objects and Classes

- Class definition works just like function definition!
- Instances work in the same way as classes!
- Calling a class triggers the creation of an instance definition node
- ID nodes are the repository for the polymorphic state of an instance
- They generate monomorphic instance state types and send them into the world

Foreign Code

- Type inference cannot see into an extension module
- There lies doom
- Starkiller gives extension writers a minilanguage for declaring the type inference properties of their extensions
- Most extensions are real simple: int(x) always returns an integer
- Of course, when Starkiller is done, there will be no reason for foreign code (just kidding)

Foreigner code, among us, plotting against us!

- Some extensions are unspeakably compli-cated
  - they might call arbitrary functions
  - they might mutate their arguments or some ob-ject that is part of global state
- The external type description language is really Python
  - External type descriptions run as extensions of the Starkiller type inferencer
  - You can use them to raise the dead

Where are we now?

- Starkiller type inferencer is mostly implemented
  - almost all of the hard parts are done
  - most of the unfinished work is boring detail
- The compiler is in the very early stages
  - a prototype works on simple code that doesn't push it too hard
  - no runtime system, no builtin types except int and float

“I wrote emacs, will you sleep with me?” -RMS

- Where is the code?
  - The compiler will be released under the GPL
  - The runtime library will be under the LGPL
  - We're still waiting for MIT to do the paperwork
  - So no code for you today! Sorry!
  - If you kill me now, you'll never get it
  - Ask me if you want to look at the code here
- Expect a release in early May
- I know, you hate the GPL
  - Starkiller is a research toy that will never be useful in a production environment

Suckling on the government teat

- Who owns Starkiller? MIT!
- Who paid for Starkiller's development? You did! Pat yourselves on the back!
- Thank you taxpayers!
- “So, that means that you are a whore, MIT is your pimp, and DARPA is the john who likes to play rough. . .Hey Mike, is there anything you won't do for money?”
- A secret: don't tell DARPA I'm not building the sun destroying weapon they think I am.

Justify your existance

- Very preliminary benchmark with the proto-type compiler and type inferencer
- All benchmarks are lies
- This one is pathological
- Call the factorial and fibonacci functions
- In a loop. Over and Over.
- CPython completion time: 03:16
- Starkiller completion time: 00:03.25
- Speedup: 60

The Future

- Development plan
- Future directions
- Doorway to a new world
- Questions

Development Plan

1. Finish thesis and graduate
2. Find job and avoid sleeping under bridge
3. Find new girlfriend
4. In copious spare time, make Starkiller take over world
   - finish type inferencer
     - get eval/exec working
   - finish compiler
   - extra optimization passes
     - static and stack allocation

Future Directions

- The same techniques that Starkiller uses for type inference can be used to solve other problems
  - Free threading
  - Static error detection
  - Object lifecycle tracking
  - Vectorizing and loop fusion

Doorway to a new world

- The past is dead
- The old limits don't apply anymore
- We will feast on the flesh of the fortran pro-grammers!
- In two years, Python will be faster than C/C++ for scientific computing
- Most people still won't accept it, but that is because technical reality takes a backseat to culture

Questions

- Indictment of the sun
  - We hatessss it! It burns, it burns!
  - The pale yellow face mocks us, keeps us from hearing the machine
  - Causes global warming
  - Sunburns
  - DARPA says sun bad. It warms our enemies.
  - Weakens our dependence on foreign oil
- There is only one logical conclusion: we must destroy the sun