Unladen Swallow:
Fewer coconuts, faster Python

Collin Winter, Jeffrey Yasskin
{collinwinter,jyasskin}@google.com

unladen-swallow@googlegroups.com
#unladenswallow on OFTC

Why did Google start this?

- Lots of Python at Google.
  - Python one of Google’s three primary languages.
  - Python enables fast development, rapid prototyping.
  - Engineers should use the language they want...
  - ...and it should be fast.

- Biggest user: YouTube
  - YouTube is pure-Python.
  - #2 search site on the Internet, behind google.com.

Goals

- Make Python 5x faster.
- Source-compatible with existing Python code.
- Source-compatible with existing C extension modules.
- Focus on ease of migration.
- Open-source everything, merge back into CPython.

- Baseline requirements:
  - Embeddable in C++ applications.
  - Support existing C extension modules and SWIG.
  - Compatible with all our existing applications.
  - Compatible with our existing infrastructure.
  - As fast, or faster, than CPython.
- Baseline: branch CPython.

Why is Python slow?

```
def add(a, b):
    return a + b
```

- Everything is an object.
- Everything is a method call, eventually.
- Ducktyping means we can’t statically predict receiver types.

```
def add(a, b):
    return a + b
```

```
define %struct._object* @"#u#add"(%struct._frame* %frame) {
entry:
        %fastlocals19 = alloca [2 x %struct._object*], align 4                                              ; <[2 x %struct._object*]*> [#uses=4]
        %fastlocals19.sub = getelementptr [2 x %struct._object*]* %fastlocals19, i32 0, i32 0               ; <%struct._object**> [#uses=1]
        %0 = load %struct._ts** @_PyThreadState_Current                                                    ; <%struct._ts*> [#uses=6]
        %f_valuestack = getelementptr %struct._frame* %frame, i32 0, i32 8                                 ; <%struct._object***> [#uses=1]
        %stack_bottom = load %struct._object*** %f_valuestack                                              ; <%struct._object**> [#uses=7]
        %f_stacktop = getelementptr %struct._frame* %frame, i32 0, i32 9                                   ; <%struct._object***> [#uses=2]
        store %struct._object** null, %struct._object*** %f_stacktop
        %use_tracing = getelementptr %struct._ts* %0, i32 0, i32 5                                         ; <i32*> [#uses=2]
        %use_tracing1 = load i32* %use_tracing                                                             ; <i32> [#uses=1]
        %1 = icmp eq i32 %use_tracing1, 0                                                                  ; <i1> [#uses=1]
        br i1 %1, label %continue_entry, label %trace_enter_function

bail_to_interpreter:                                                ; preds = %call_trace, %trace_enter_function
        %f_use_llvm = getelementptr %struct._frame* %frame, i32 0, i32 16                                  ; <i32*> [#uses=1]
        store i32 0, i32* %f_use_llvm
        store %struct._object** %stack_bottom, %struct._object*** %f_stacktop
        %f_iblock = getelementptr %struct._frame* %frame, i32 0, i32 19                                    ; <i8*> [#uses=1]
        store i8 0, i8* %f_iblock
        %f_localsplus4 = getelementptr %struct._frame* %frame, i32 0, i32 23                               ; <[1 x %struct._object*]*> [#uses=1]
        %2 = bitcast [2 x %struct._object*]* %fastlocals19 to i8*                                          ; <i8*> [#uses=1]
        %3 = bitcast [1 x %struct._object*]* %f_localsplus4 to i8*                                         ; <i8*> [#uses=1]
        call void @llvm.memcpy.i64(i8* nocapture %3, i8* nocapture %2, i64 ptrtoint (%struct._object** getelementptr (%struct._object** null, i32 2) to i64), i32 1) nounwind
        %4 = tail call %struct._object* @PyEval_EvalFrame(%struct._frame* %frame)                          ; <%struct._object*> [#uses=1]
        ret %struct._object* %4

trace_enter_function:                                               ; preds = %entry
        %f_bailed_from_llvm = getelementptr %struct._frame* %frame, i32 0, i32 20                          ; <i8*> [#uses=1]
        store i8 1, i8* %f_bailed_from_llvm
        br label %bail_to_interpreter

continue_entry:                                                     ; preds = %entry
        %f_localsplus = getelementptr %struct._frame* %frame, i32 0, i32 23                                ; <[1 x %struct._object*]*> [#uses=1]
        %5 = bitcast [1 x %struct._object*]* %f_localsplus to i8*                                          ; <i8*> [#uses=1]
        %6 = bitcast [2 x %struct._object*]* %fastlocals19 to i8*                                          ; <i8*> [#uses=1]
        call void @llvm.memcpy.i64(i8* nocapture %6, i8* nocapture %5, i64 ptrtoint (%struct._object** getelementptr (%struct._object** null, i32 2) to i64), i32 1) nounwind
        %f_lineno = getelementptr %struct._frame* %frame, i32 0, i32 17                                    ; <i32*> [#uses=1]
        store i32 2, i32* %f_lineno
        %7 = load i32* @_Py_TracingPossible                                                                ; <i32> [#uses=1]
        %8 = icmp eq i32 %7, 0                                                                             ; <i1> [#uses=1]
        br i1 %8, label %line_start, label %call_trace

call_exc_trace:                                                     ; preds = %PropagateExceptionOnNull_propagate
        call void @_PyEval_CallExcTrace(%struct._ts* %0, %struct._frame* %frame)
        br label %unwind_loop_header.preheader

unwind_loop_header.preheader:                                       ; preds = %PropagateExceptionOnNull_propagate, %call_exc_trace, %PropagateExceptionOnNull_pass
        %unwind_reason.ph = phi i8 [ 8, %PropagateExceptionOnNull_pass ], [ 2, %call_exc_trace ], [ 2, %PropagateExceptionOnNull_propagate ]                        ; <i8> [#uses=2]
        %retval_addr.0.ph = phi %struct._object* [ %binop_result, %PropagateExceptionOnNull_pass ], [ null, %call_exc_trace ], [ null, %PropagateExceptionOnNull_propagate ]   ; <%struct._object*> [#uses=1]
        %unwind_reason_addr.0.ph = phi i8 [ 8, %PropagateExceptionOnNull_pass ], [ 2, %call_exc_trace ], [ 2, %PropagateExceptionOnNull_propagate ]                  ; <i8> [#uses=2]
        %currently_unwinding_exception = icmp eq i8 %unwind_reason.ph, 2                                   ; <i1> [#uses=0]
        br label %pop_loop5

pop_loop5:                                                          ; preds = %13, %18, %pop_stack6, %unwind_loop_header.preheader
        %stack_pointer_addr.2 = phi %struct._object** [ %stack_bottom, %unwind_loop_header.preheader ], [ %10, %pop_stack6 ], [ %10, %18 ], [ %10, %13 ]             ; <%struct._object**> [#uses=2]
        %9 = icmp ugt %struct._object** %stack_pointer_addr.2, %stack_bottom                               ; <i1> [#uses=1]
        br i1 %9, label %pop_stack6, label %pop_done7

pop_stack6:                                                         ; preds = %pop_loop5
        %10 = getelementptr %struct._object** %stack_pointer_addr.2, i32 -1                                ; <%struct._object**> [#uses=4]
        %11 = load %struct._object** %10                                                                   ; <%struct._object*> [#uses=4]
        %12 = icmp eq %struct._object* %11, null                                                           ; <i1> [#uses=1]
        br i1 %12, label %pop_loop5, label %13

; <label>:13                                                        ; preds = %pop_stack6
        %14 = getelementptr %struct._object* %11, i32 0, i32 0                                             ; <i32*> [#uses=2]
        %15 = load i32* %14                                                                                ; <i32> [#uses=1]
        %16 = add i32 %15, -1                                                                              ; <i32> [#uses=2]
        store i32 %16, i32* %14
        %17 = icmp eq i32 %16, 0                                                                           ; <i1> [#uses=1]
        br i1 %17, label %18, label %pop_loop5

; <label>:18                                                        ; preds = %13
        %19 = getelementptr %struct._object* %11, i32 0, i32 1                                             ; <%struct._typeobject**> [#uses=1]
        %20 = load %struct._typeobject** %19                                                               ; <%struct._typeobject*> [#uses=1]
        %21 = getelementptr %struct._typeobject* %20, i32 0, i32 6                                         ; <void (%struct._object*)**> [#uses=1]
        %22 = load void (%struct._object*)** %21                                                           ; <void (%struct._object*)*> [#uses=1]
        call void %22(%struct._object* %11) nounwind
        br label %pop_loop5

pop_done7:                                                          ; preds = %pop_loop5
        %23 = icmp eq i8 %unwind_reason.ph, 8                                                              ; <i1> [#uses=1]
        %retval_addr.1 = select i1 %23, %struct._object* %retval_addr.0.ph, %struct._object* null                                                                  ; <%struct._object*> [#uses=7]
        %24 = load i32* %use_tracing                                                                       ; <i32> [#uses=1]
        %25 = icmp eq i32 %24, 0                                                                           ; <i1> [#uses=1]
        br i1 %25, label %check_frame_exception, label %trace_leave_function

check_frame_exception:                                              ; preds = %32, %tracer_raised, %trace_leave_function, %pop_done7, %37
        %retval_addr.2 = phi %struct._object* [ null, %37 ], [ %retval_addr.1, %pop_done7 ], [ %retval_addr.1, %trace_leave_function ], [ null, %tracer_raised ], [ null, %32 ]   ; <%struct._object*> [#uses=2]
        %frame9 = getelementptr %struct._ts* %0, i32 0, i32 2                                              ; <%struct._frame**> [#uses=1]
        %"tstate->frame" = load %struct._frame** %frame9                                                   ; <%struct._frame*> [#uses=1]
        %f_exc_type = getelementptr %struct._frame* %"tstate->frame", i32 0, i32 11                        ; <%struct._object**> [#uses=1]
        %"tstate->frame->f_exc_type" = load %struct._object** %f_exc_type                                  ; <%struct._object*> [#uses=1]
        %26 = icmp eq %struct._object* %"tstate->frame->f_exc_type", null                                  ; <i1> [#uses=1]
        br i1 %26, label %finish_return, label %have_frame_exception

trace_leave_function:                                               ; preds = %pop_done7
        %is_return = icmp eq i8 %unwind_reason_addr.0.ph, 8                                                ; <i1> [#uses=1]
        %is_exception = icmp eq i8 %unwind_reason_addr.0.ph, 2                                             ; <i1> [#uses=1]
        %27 = zext i1 %is_return to i8                                                                     ; <i8> [#uses=1]
        %28 = zext i1 %is_exception to i8                                                                  ; <i8> [#uses=1]
        %29 = call i32 @_PyEval_TraceLeaveFunction(%struct._ts* %0, %struct._frame* %frame, %struct._object* %retval_addr.1, i8 %27, i8 %28)                    ; <i32> [#uses=1]
        %30 = icmp eq i32 %29, 0                                                                           ; <i1> [#uses=1]
        br i1 %30, label %check_frame_exception, label %tracer_raised

tracer_raised:                                                      ; preds = %trace_leave_function
        %31 = icmp eq %struct._object* %retval_addr.1, null                                                ; <i1> [#uses=1]
        br i1 %31, label %check_frame_exception, label %32

; <label>:32                                                        ; preds = %tracer_raised
        %33 = getelementptr %struct._object* %retval_addr.1, i32 0, i32 0                                  ; <i32*> [#uses=2]
        %34 = load i32* %33                                                                                ; <i32> [#uses=1]
        %35 = add i32 %34, -1                                                                              ; <i32> [#uses=2]
        store i32 %35, i32* %33
        %36 = icmp eq i32 %35, 0                                                                           ; <i1> [#uses=1]
        br i1 %36, label %37, label %check_frame_exception

; <label>:37                                                        ; preds = %32
        %38 = getelementptr %struct._object* %retval_addr.1, i32 0, i32 1                                  ; <%struct._typeobject**> [#uses=1]
        %39 = load %struct._typeobject** %38                                                               ; <%struct._typeobject*> [#uses=1]
        %40 = getelementptr %struct._typeobject* %39, i32 0, i32 6                                         ; <void (%struct._object*)**> [#uses=1]
        %41 = load void (%struct._object*)** %40                                                           ; <void (%struct._object*)*> [#uses=1]
        call void %41(%struct._object* %retval_addr.1) nounwind
        br label %check_frame_exception

have_frame_exception:                                               ; preds = %check_frame_exception
        call void @_PyEval_ResetExcInfo(%struct._ts* %0)
        ret %struct._object* %retval_addr.2

finish_return:                                                      ; preds = %check_frame_exception
        ret %struct._object* %retval_addr.2

line_start:                                                         ; preds = %continue_entry
        %FAST_loaded = load %struct._object** %fastlocals19.sub                                            ; <%struct._object*> [#uses=2]
        %42 = getelementptr %struct._object* %FAST_loaded, i32 0, i32 0                                    ; <i32*> [#uses=2]
        %43 = load i32* %42                                                                                ; <i32> [#uses=1]
        %44 = add i32 %43, 1                                                                               ; <i32> [#uses=1]
        store i32 %44, i32* %42
        store %struct._object* %FAST_loaded, %struct._object** %stack_bottom
        %45 = getelementptr %struct._object** %stack_bottom, i32 1                                         ; <%struct._object**> [#uses=1]
        %46 = getelementptr [2 x %struct._object*]* %fastlocals19, i32 0, i32 1                            ; <%struct._object**> [#uses=1]
        %FAST_loaded12 = load %struct._object** %46                                                        ; <%struct._object*> [#uses=5]
        %47 = getelementptr %struct._object* %FAST_loaded12, i32 0, i32 0                                  ; <i32*> [#uses=4]
        %48 = load i32* %47                                                                                ; <i32> [#uses=1]
        %49 = add i32 %48, 1                                                                               ; <i32> [#uses=1]
        store i32 %49, i32* %47
        store %struct._object* %FAST_loaded12, %struct._object** %45
        %50 = load %struct._object** %stack_bottom                                                         ; <%struct._object*> [#uses=4]
        %binop_result = call %struct._object* @PyNumber_Add(%struct._object* %50, %struct._object* %FAST_loaded12)                                             ; <%struct._object*> [#uses=3]
        %51 = getelementptr %struct._object* %50, i32 0, i32 0                                             ; <i32*> [#uses=2]
        %52 = load i32* %51                                                                                ; <i32> [#uses=1]
        %53 = add i32 %52, -1                                                                              ; <i32> [#uses=2]
        store i32 %53, i32* %51
        %54 = icmp eq i32 %53, 0                                                                           ; <i1> [#uses=1]
        br i1 %54, label %55, label %_PyLlvm_WrapDecref.exit17

; <label>:55                                                        ; preds = %line_start
        %56 = getelementptr %struct._object* %50, i32 0, i32 1                                             ; <%struct._typeobject**> [#uses=1]
        %57 = load %struct._typeobject** %56                                                               ; <%struct._typeobject*> [#uses=1]
        %58 = getelementptr %struct._typeobject* %57, i32 0, i32 6                                         ; <void (%struct._object*)**> [#uses=1]
        %59 = load void (%struct._object*)** %58                                                           ; <void (%struct._object*)*> [#uses=1]
        call void %59(%struct._object* %50) nounwind
        br label %_PyLlvm_WrapDecref.exit17

_PyLlvm_WrapDecref.exit17:                                          ; preds = %line_start, %55
        %60 = load i32* %47                                                                                ; <i32> [#uses=1]
        %61 = add i32 %60, -1                                                                              ; <i32> [#uses=2]
        store i32 %61, i32* %47
        %62 = icmp eq i32 %61, 0                                                                           ; <i1> [#uses=1]
        br i1 %62, label %63, label %_PyLlvm_WrapDecref.exit18

; <label>:63                                                        ; preds = %_PyLlvm_WrapDecref.exit17
        %64 = getelementptr %struct._object* %FAST_loaded12, i32 0, i32 1                                  ; <%struct._typeobject**> [#uses=1]
        %65 = load %struct._typeobject** %64                                                               ; <%struct._typeobject*> [#uses=1]
        %66 = getelementptr %struct._typeobject* %65, i32 0, i32 6                                         ; <void (%struct._object*)**> [#uses=1]
        %67 = load void (%struct._object*)** %66                                                           ; <void (%struct._object*)*> [#uses=1]
        call void %67(%struct._object* %FAST_loaded12) nounwind
        br label %_PyLlvm_WrapDecref.exit18

_PyLlvm_WrapDecref.exit18:                                          ; preds = %_PyLlvm_WrapDecref.exit17, %63
        %68 = icmp eq %struct._object* %binop_result, null                                                 ; <i1> [#uses=1]
        br i1 %68, label %PropagateExceptionOnNull_propagate, label %PropagateExceptionOnNull_pass

call_trace:                                                         ; preds = %continue_entry
        %f_lasti = getelementptr %struct._frame* %frame, i32 0, i32 15                                     ; <i32*> [#uses=1]
        store i32 -1, i32* %f_lasti
        %f_bailed_from_llvm11 = getelementptr %struct._frame* %frame, i32 0, i32 20                        ; <i8*> [#uses=1]
        store i8 2, i8* %f_bailed_from_llvm11
        br label %bail_to_interpreter

PropagateExceptionOnNull_propagate:                                 ; preds = %_PyLlvm_WrapDecref.exit18
        %69 = call i32 @PyTraceBack_Here(%struct._frame* %frame)                                           ; <i32> [#uses=0]
        %c_tracefunc = getelementptr %struct._ts* %0, i32 0, i32 7                                         ; <i32 (%struct._object*, %struct._frame*, i32, %struct._object*)**> [#uses=1]
        %70 = load i32 (%struct._object*, %struct._frame*, i32, %struct._object*)** %c_tracefunc                                          ; <i32 (%struct._object*, %struct._frame*, i32, %struct._object*)*> [#uses=1]
        %71 = icmp eq i32 (%struct._object*, %struct._frame*, i32, %struct._object*)* %70, null                                           ; <i1> [#uses=1]
        br i1 %71, label %unwind_loop_header.preheader, label %call_exc_trace

PropagateExceptionOnNull_pass:                                      ; preds = %_PyLlvm_WrapDecref.exit18
        store %struct._object* %binop_result, %struct._object** %stack_bottom
        br label %unwind_loop_header.preheader

}
```

Why is Python slow?

```
def foo(x):
    yield len(x)
    yield len(x)

>>> g = foo(range(5))
>>> g.next()
5
>>> len = lambda y: 8
>>> g.next()
8
```

How do you make Python faster?

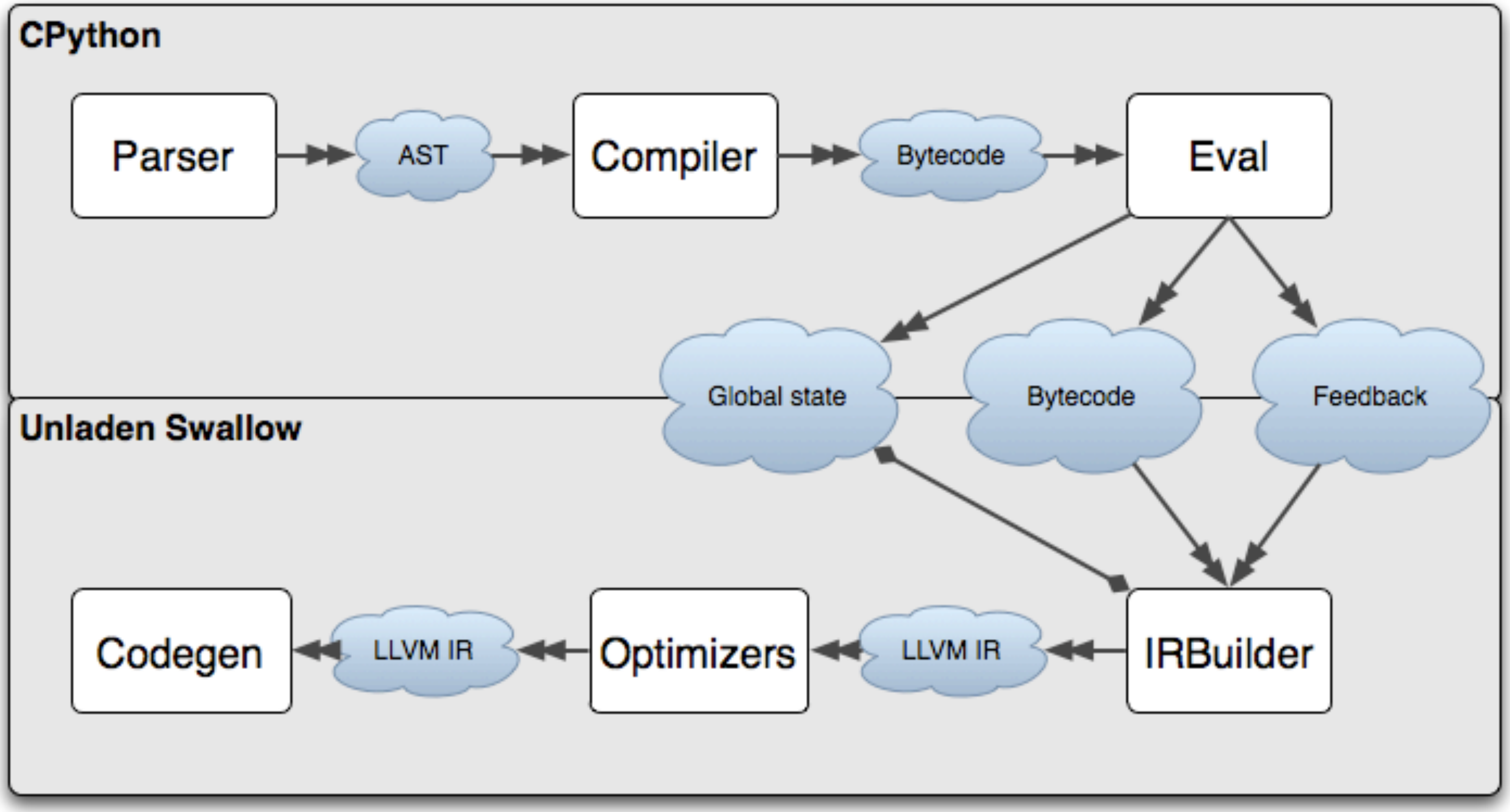
- CPython today:
  - Stack-based bytecode interpreter.
  - Missed the last 30 years of research.

- Conservative ideas:
  - Computed goto-based interpreter loop.
  - Add new, specialized opcodes.
  - Superinstructions (WPython)

Unladen Swallow

- Unladen Swallow: just-in-time compilation.
  - Preserve C extension compatibility.
  - Use LLVM for code generation, optimization.
  - Use Clang for inlining C functions.
  - Runtime feedback for specialization.
  - Inspired by Self-93, HotSpot, V8, Psyco.

Unladen Swallow



Top-down Inlining Opportunties

```
>>> x = dict()
>>> len(x)
```

...len() calls PyObject_Size()

...which looks up x->ob_type->tp_as_sequence->sq_length

...which is NULL, so call PyMapping_Size()

...which looks up x->ob_type->tp_as_mapping->mp_length

...which is a function pointer to dict_length()

...which returns x->ma_used

Using Clang and llc for Inlining

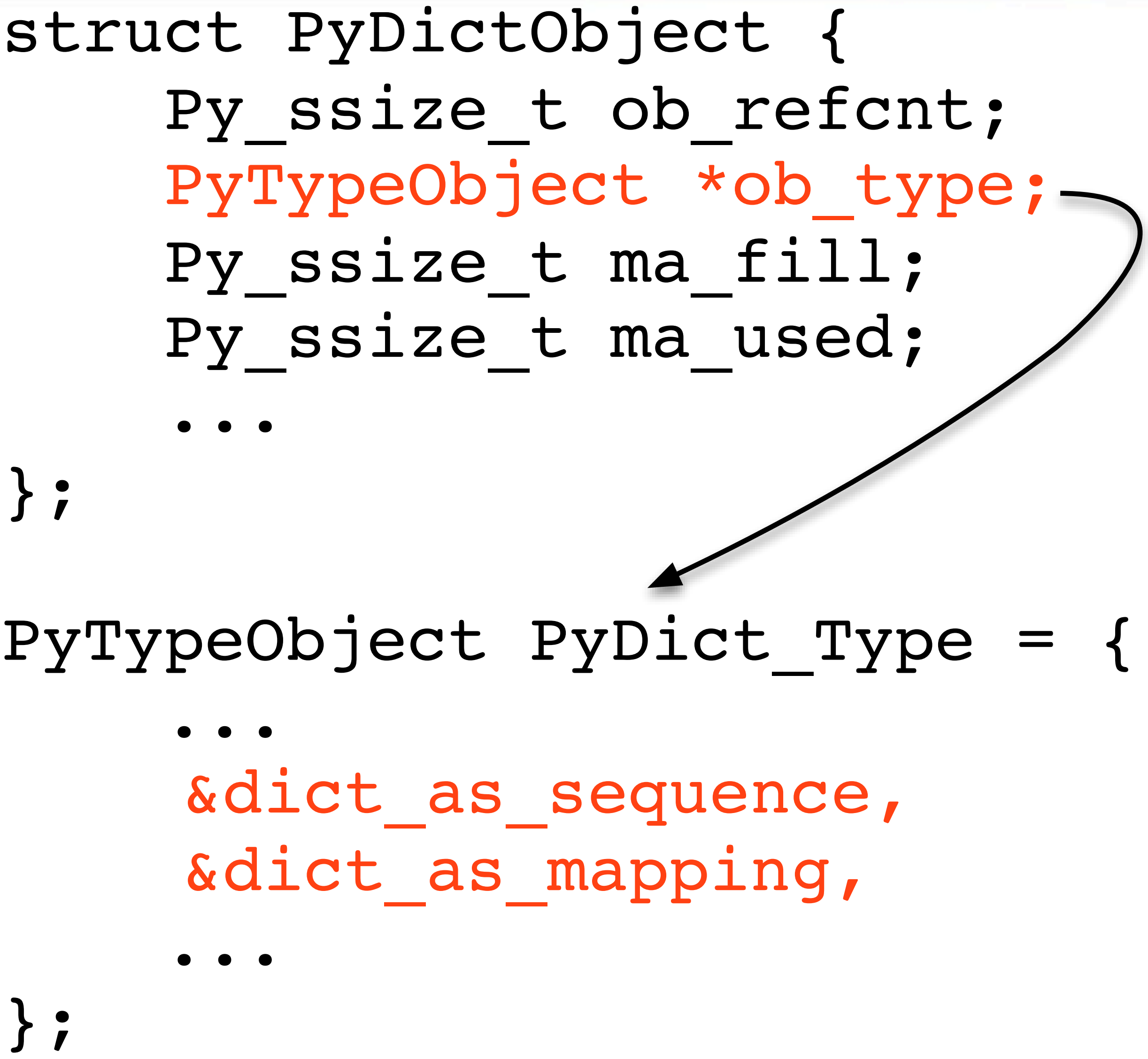
- Pipeline: C --> LLVM .bc --> C++ API calls
- Inline useful macros:

```
/*** Python/llvm_inline_functions.c ***/

void __attribute__((always_inline))
_PyLlvm_WrapDecref(PyObject *obj)
{
    Py_DECREF(obj);
}
```

- Uses a custom single-function inlining pass.

Python Objects: constant(ish)



```
struct PyDictObject {
    Py_ssize_t ob_refcnt;
    PyTypeObject *ob_type;
    Py_ssize_t ma_fill;
    Py_ssize_t ma_used;
    ...
};

PyTypeObject PyDict_Type = {
    ...
    &dict_as_sequence,
    &dict_as_mapping,
    ...
};
```

Experience with LLVM

- Generally very positive!
- Obstacles overcome:
  - JIT bugs.
  - gdb/oprofile support.
  - Some quadratic behaviour.
- Obstacles remaining:
  - LLVM does not cure cancer.
  - Some optimizations missing.
  - More lurking quadratic behaviour.
  - Python semantics.
- Stop; collaborate; listen.

Measurement & Testing

- Benchmarks representing real-world applications:
  - YouTube hotspots: Spitfire templates.
  - Libraries: pickling, regular expressions.
  - Apps: 2to3, Django.
  - Microbenchmarks: GC, IO, string operations.

- Correctness:
  - SWIGed code, extensions used by YouTube.
  - Google's large Python codebase.
  - Large Python projects: Mercurial, NumPy, Twisted, etc.
  - Randomized testing.

Status Report

- Two releases so far:
  - 2009Q1: 15-20% faster than CPython.
    - cPickle 2x faster.
  - 2009Q2: 10% faster than Q1; full JIT on top of LLVM.

- 2009Q3: stabilizing, close to release.
  - Optimized LOAD_GLOBAL opcode.
  - Optimized calls to C functions.
  - Less unnecessary error checking.
  - Better constant propagation.
  - More data exposed to LLVM
  - gdb, oprofile support.
  - 20-75% faster than Q2.

Looking Forward: Q4

- 2009Q4:
    - More typefeed back!
    - More inlining!
    - More Clang compilation!
    - Fewer frame allocations!
    - Fewer GIL checks!
    - Begin merger with CPython!

GIL?

Fin

Questions?

http://code.google.com/p/unladen-swallow/

collinwinter@google.com
unladen-swallow@googlegroups.com
#unladenswallow on OFTC

Fin

Backup Slides

Make it faster: specialize dynamically

```
def foo(x):
    ...
    y = len(x)
    ...
```

→

```
...
LOAD_GLOBAL   7 (len)
LOAD_FAST     0 (x)
CALL_FUNCTION 1
STORE_FAST    4 (y)
...
```

Make it faster: LOAD_GLOBAL

```
...
LOAD_GLOBAL   7 (len)
...
```

↓

```
x = PyDict_GetItem(globals, “len”)
if (x == NULL) {
    x = PyDict_GetItem(builtins, “len”)
    if (x == NULL) {
        PyErr_NoGlobals()
        return -1;
    }
}
```

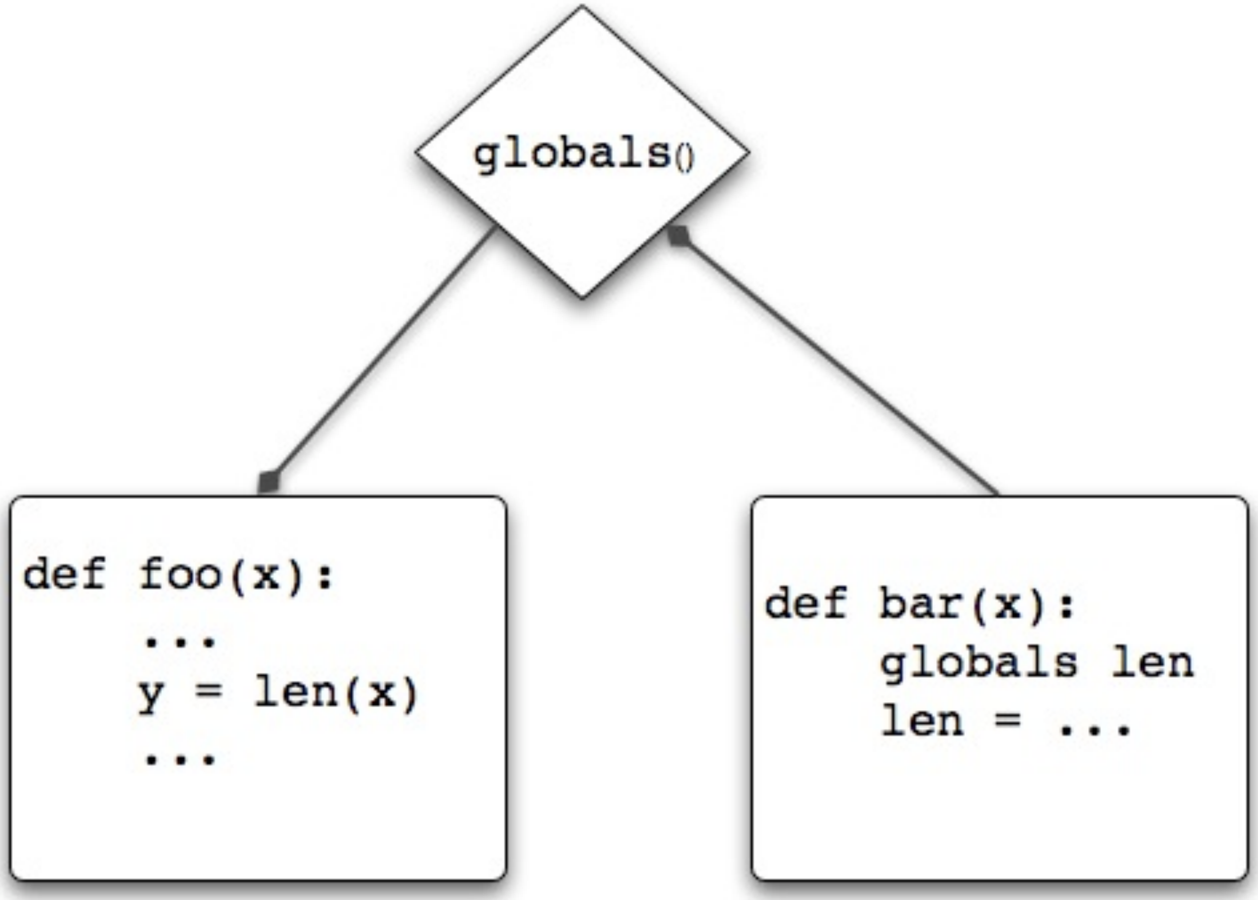
Make it faster: LOAD_GLOBAL

```
...
LOAD_GLOBAL   7 (len)
...
```

↓

```
if (world_has_not_changed) {
    x = (PyObject *)11712432
}
else {
    goto bail_to_interpreter
}
```

Make it faster: `if (world_has_not_changed) {`



globals()

```
def foo(x):
    ...
    y = len(x)
    ...
```

```
def bar(x):
    globals len
    len = ...
```

Make it faster: specialize dynamically

```
def template(data, rows):
    for row in rows:
        data.append(“<tr>”)
        for col in row:
            data.append(“<td>%s</td>” % col)
        data.append(“</tr>”)




>>> our_data = []
>>> template(our_data, our_rows)
>>> print “”.join(our_data)
```

Make it faster: call sites

```
data.append(“<td>%s</td>” % col)
```

↓

```
LOAD_FAST                   0 (data)
LOAD_ATTR                   0 (append)
LOAD_CONST                  1 ('<td>%s</td>')
LOAD_FAST                   1 (col)
BINARY_MODULO
CALL_FUNCTION               1
```

Make it faster: call sites

```
    LOAD_FAST              0 (data)
    LOAD_ATTR              0 (append)
    ...
    CALL_FUNCTION          1
```

↓

```
data = locals[0];
if (Py_TYPE(data) != EXPECTED_TYPE)
    goto bail_to_interpreter;
if (Py_TYPE_VERSION(data) != EXPECTED_VERSION)
    goto_bail_to_interpreter;
// LOAD_CONST
// LOAD_FAST
// BINARY_MODULO
retval = list_append(data, modulo_result);
```