Google™

Portable Native Client

David Sehr, Robert Muth, Jan Voung, David Meyer,
Betul Buyukkurt, Karl Schimpf, Jason Kim, Rafael Espindola,
Alan Donovan

Agenda

Google

Motivation

Approach

Developing Using PNaCl

Safe Translation

Status

Ongoing Work

Motivation

A NaCl-Enabled Web Application

Your favorite language

Screened for malicious instructions

System calls moderated by a virtualized OS

Performance within 5% of native code

Applications with NaCl

Google

Where Native Client Started

Google

OSX
Windows
Linux

x86-32

Where Native Client Started

Google

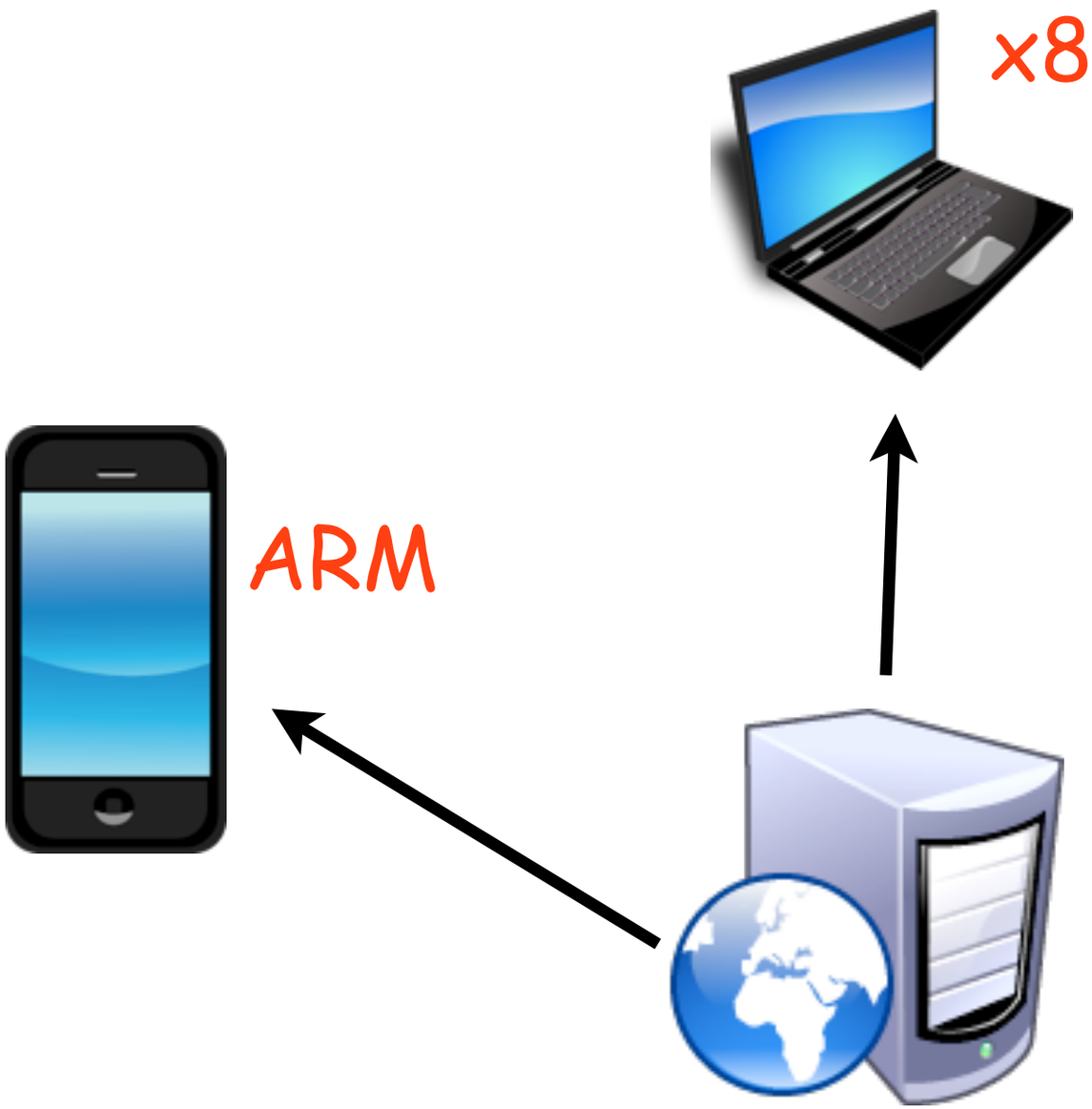
OSX
Windows
Linux

x86-32

linux-like system calls
gcc
binutils
newlib

Where We Went Next

Google



Where We Went Next

Google

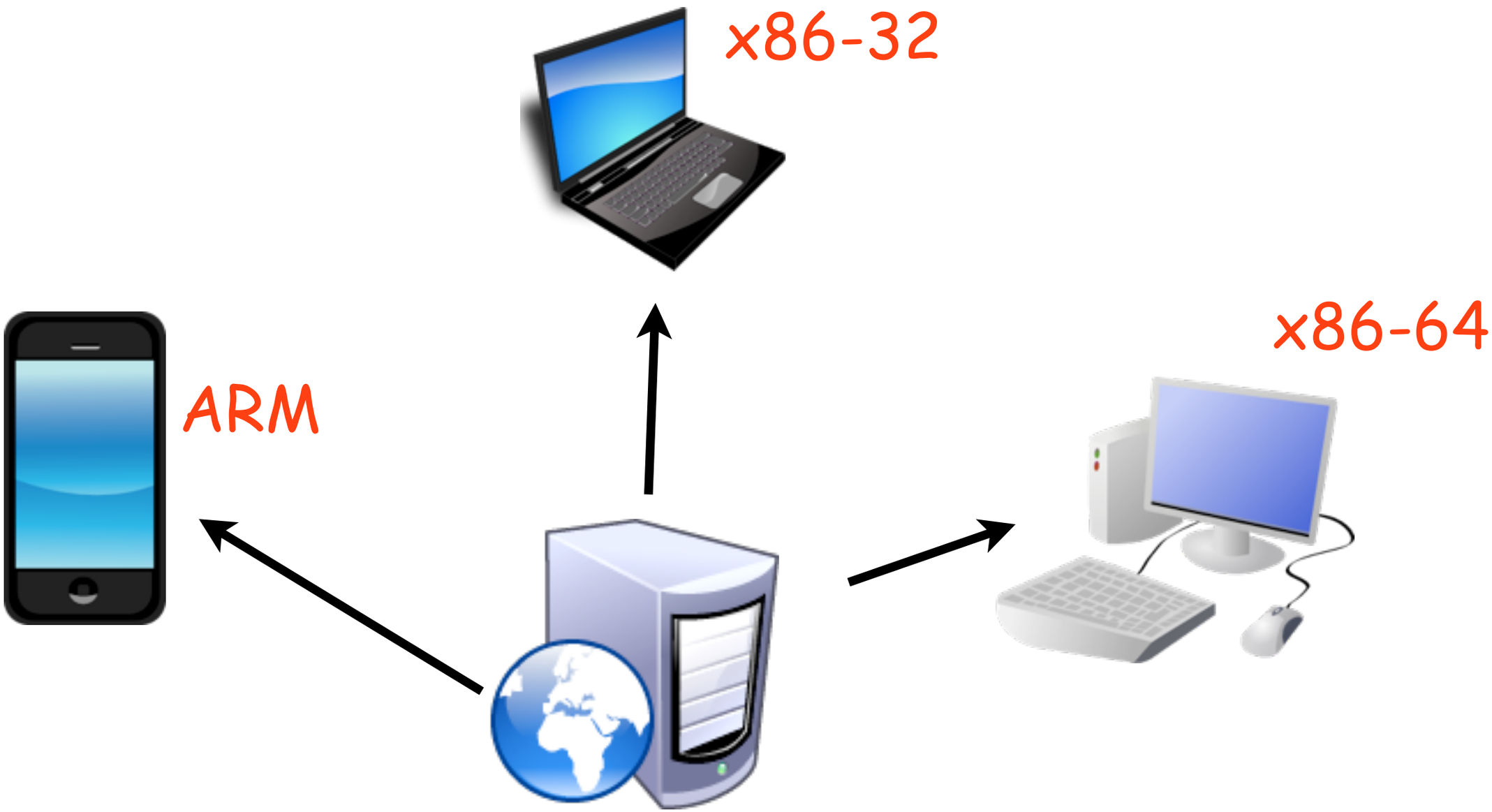
linux-like system calls
llvm
binutils
newlib

x86-32

ARM

What Developers Want

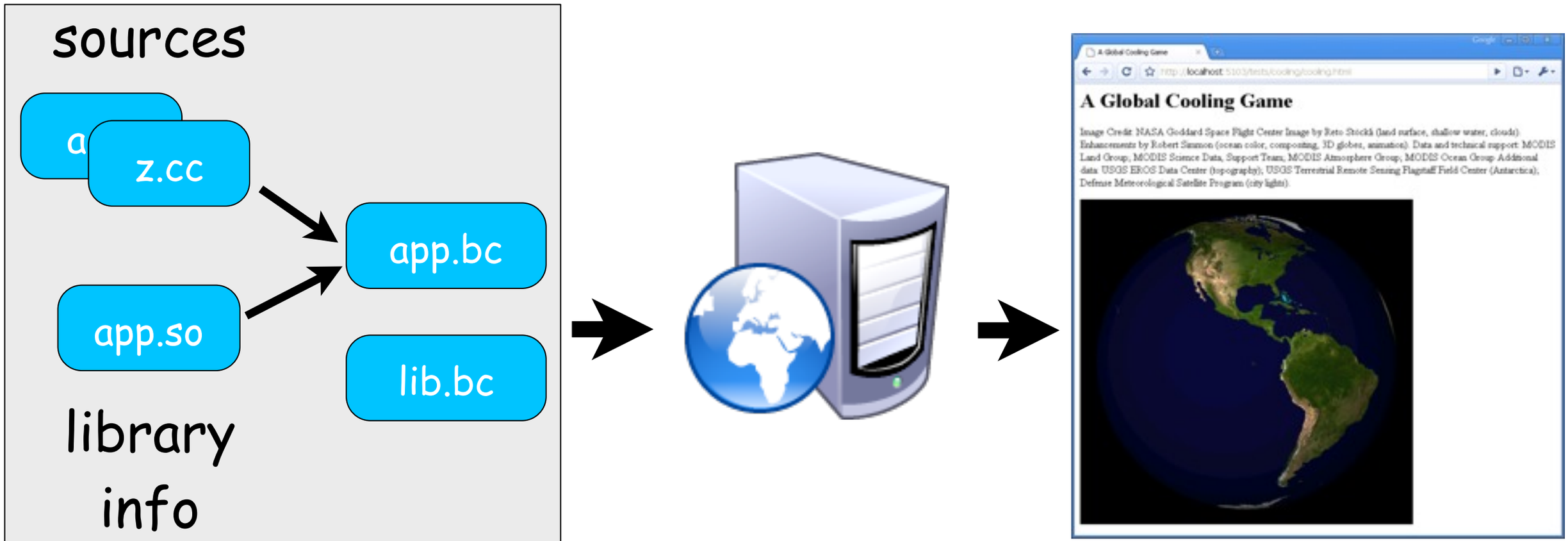
Google



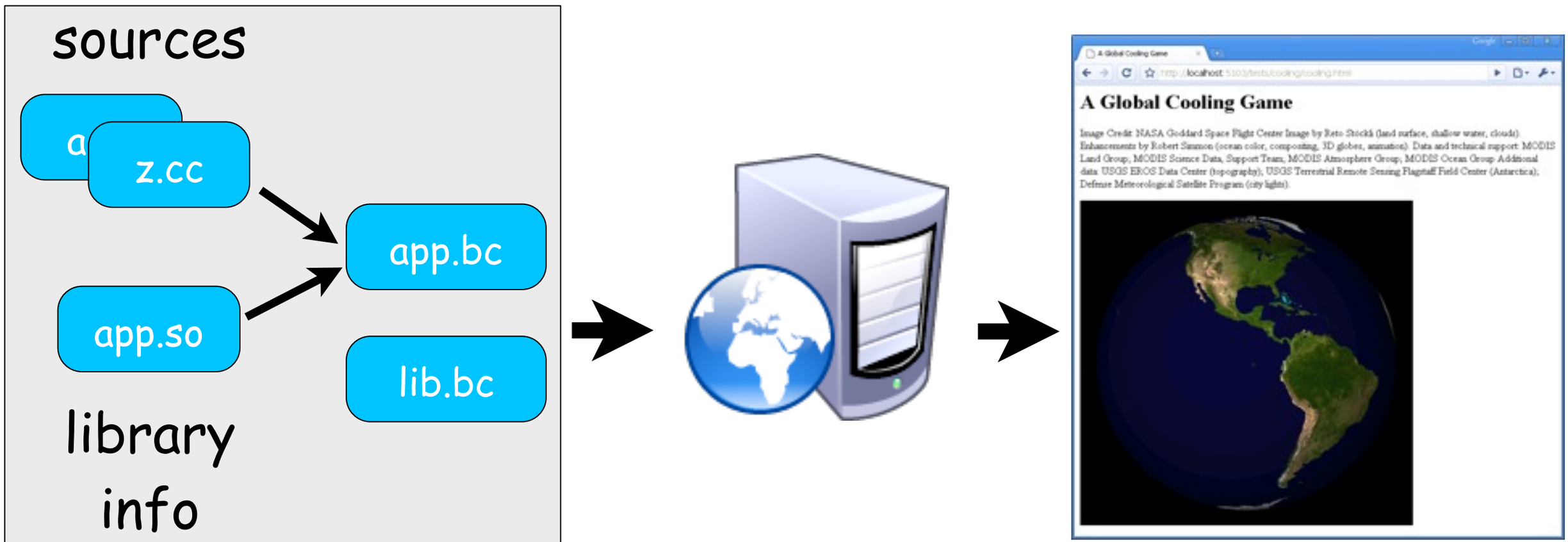
Only one porting effort

Approach

Application Life Cycle



Application Life Cycle



LLVM bitcode is PNaCl's distribution format

Client side

Native Client Helper

http://myurl/myapp.bc

translation engine

myapp.so

ELF
x86, x64, or ARM

NaCl sandbox

Developing Using PNaCl

Target Model

Google

- Address space / data model
  - ILP32 (sizeof(int) == sizeof(long) == sizeof(void*))
  - sizeof(va_list) == 24
  - 1GB maximum total address space
  - Stack pointer starts at the top of the address space
- Data types
  - IEEE fp
  - “natural” alignment
    - (e.g., double is aligned 0mod8)
- Byte order
  - Little Endian

Target Model

Google

C++ Exception Handling

- x86-32 Linux model

varargs

- sizeof(va_list) == 24
- Front end emits va_arg instruction

setjmp

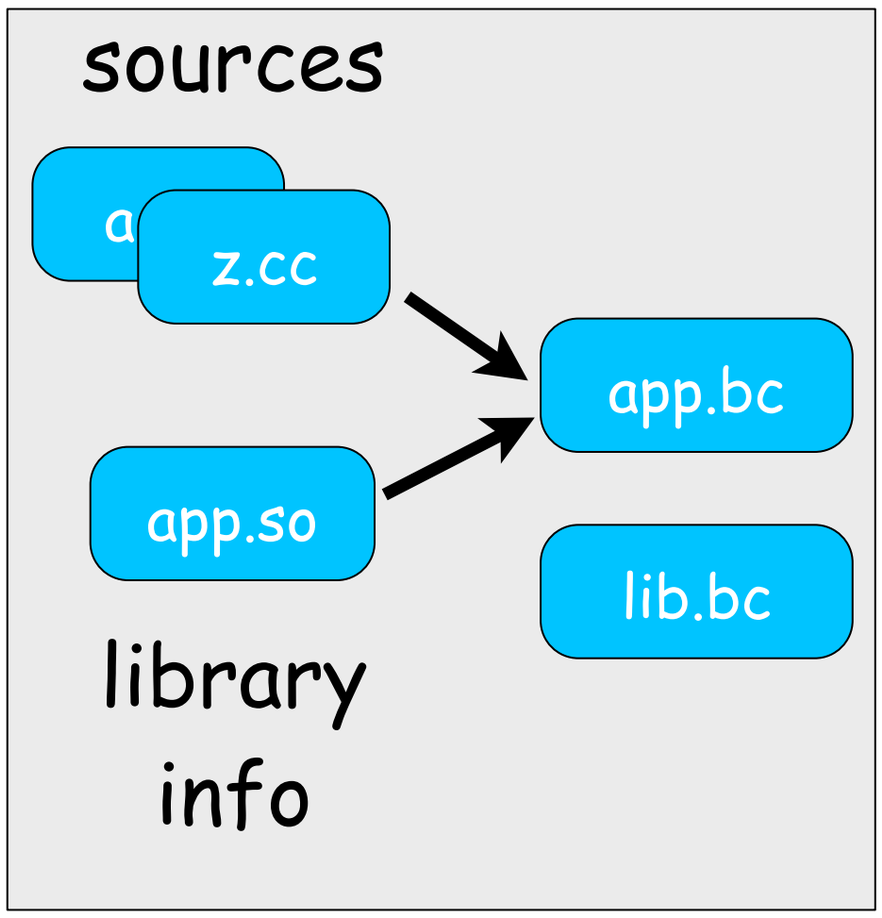
- Consistent jmp_buf size

Target Model

Google

- Calling conventions
  - Bitcode file is calling convention neutral
  - Actual target convention determined by translator
- Concurrency and memory model
  - Assume a least common denominator
    - Store ordering within a thread
    - Explicit synchronization across threads
  - We expect people to use llvm atomic/barrier intrinsics where needed

Application Life Cycle



gcc-like driver

llvm-gcc front end

LLVM's link-time optimization

Produce smallest bitcode

Expensive opts. off client

Safe Translation

Translating in a Sandbox

Google

The translator must run in the browser

Malicious bitcode files are a potential attack vector

Translating in a Sandbox

Google

The translator must run in the browser

Malicious bitcode files are a potential attack vector

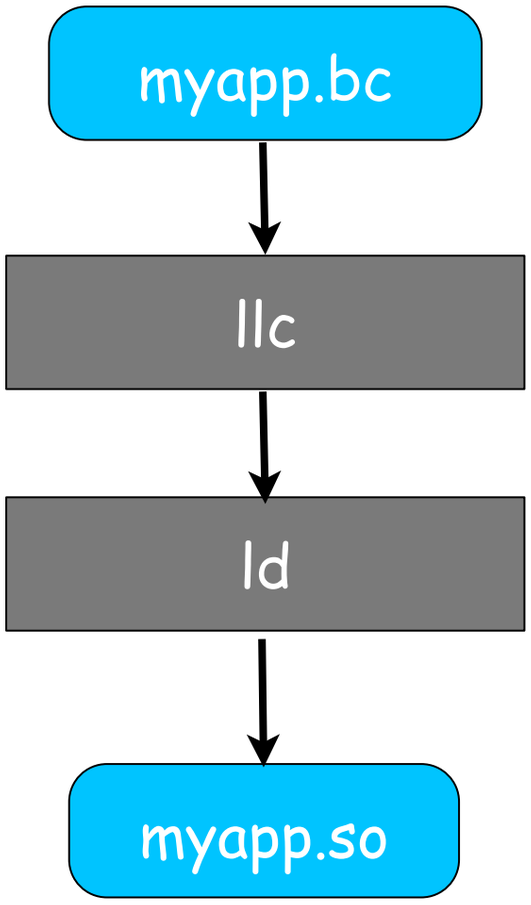
Translator phases are run as NaCl modules

Translator

Google

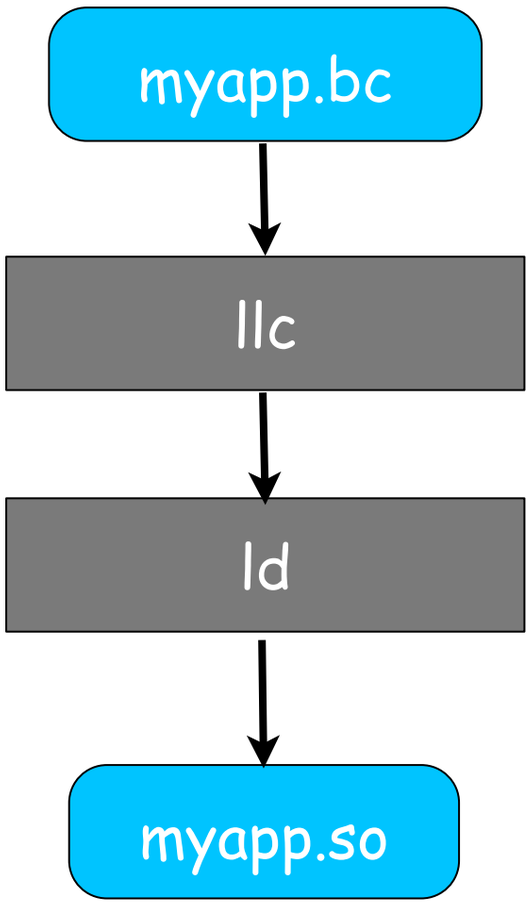
today

Rapid code generation



Translator

Google



goal

Know the platform (uarch)

Can collect/use profiling data

- Webpage-specific specialization

Can translate at

- invocation time
- install time
- asynchronously

Status

What's running?

Google

- One bitcode file translates, validates, and runs on three architectures
  - All of SPEC2000 int and the four C fp tests pass
- The translator is self-built and sandboxed
  - llc and ld run as NaCl modules on x86-32 and 64
- ARM getting close
  - ARM ELF direct object generation passing most tests
  - Still a bit of debugging on ARM self-build

What's running?

Google

Examples are running in Chromium

(With a command-line flag for now)

Ongoing Work

Developer Tools

Google

LTO producing the smallest possible .bc files

- Improve gold plugin for marking symbols internal

Gold/LTO as full linker without .bc to .o

Collecting soname list for .so generation

Translator

Google

Direct .so generation

- Emission work for DT_NEEDED

Translation time

Want to Learn More?

Google

http://www.chromium.org/nativeclient

(Follow Portable Native Client link)

http://code.google.com/p/nativeclient