NativeClient Forth

Saturday November 19, 2011

Motivation

- NativeClient allows machine code to run in the browser.
- Forth excels at bootstrapping new environments.
- My past Javascript Forths have been slow.

NativeClient (NaCl)

- Static Verification Sandbox
- Multi-threaded
- Execution in the Web Browser (currently requires web store install)
- Resource restrictions similar to Javascript
- Performance similar to machine code ~15%
- Only option for native code (machine code) under ChromeOS/ChromiumOS
- Sandbox versions for x86, x86-64, (ARM coming soon)
- Portable NaCl (PNaCl) coming soon

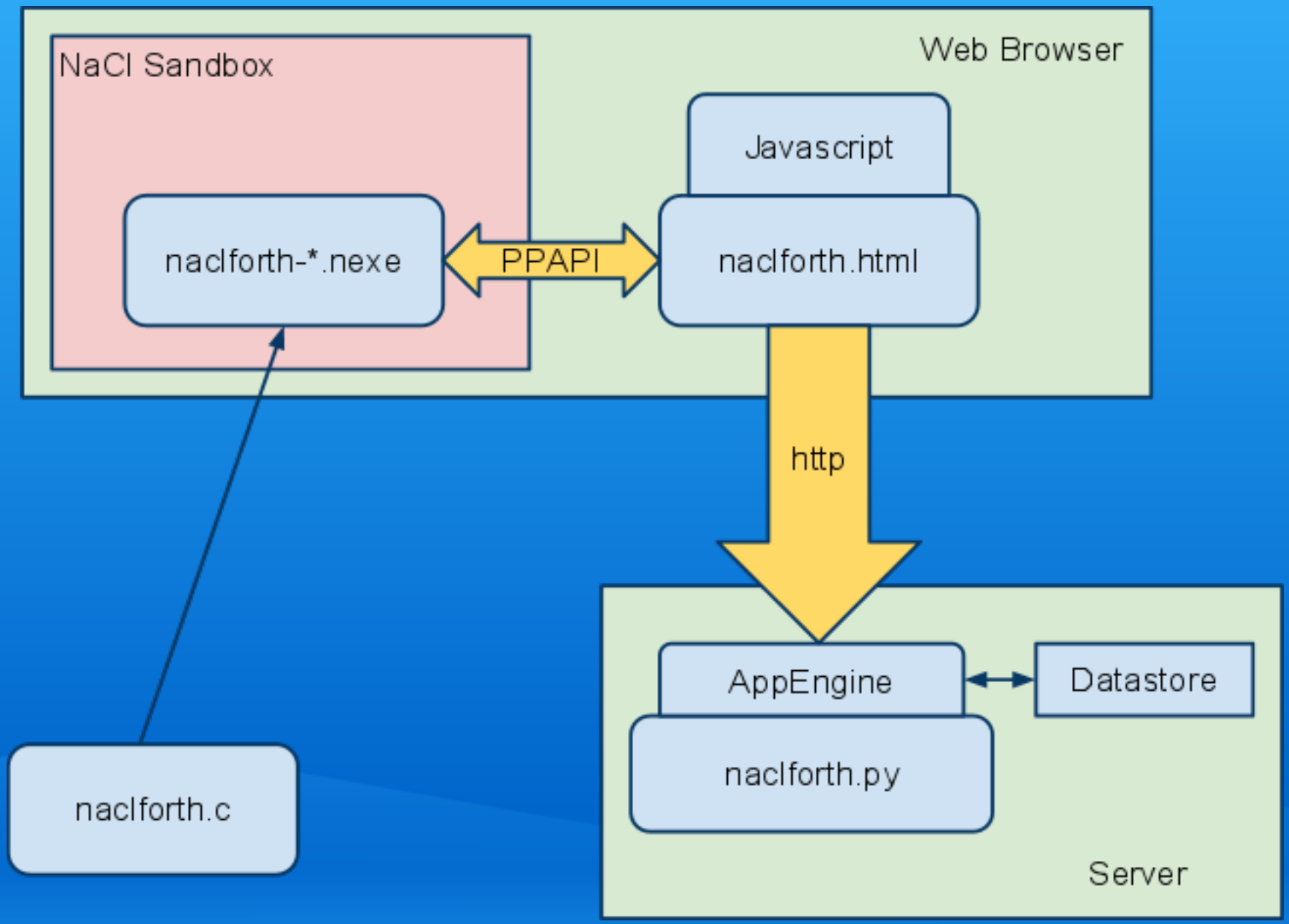
NaCl Sandboxes

- Each 32-byte block of code can be straight line disassembled
- All jumps / calls / returns must be to a 32-byte boundary
  - Enforced by requiring masking off bottom bits before control flow instructions
- Data writes restricted via page faults / segments
- I/O via system provided 'trampolines'
- Special compiler (gcc/llvm) - validation at load time
- Dynamic code (validated) via trampoline calls

Implementation

- Indirect-threaded kernel in ~750 lines of C code
  - Uses gcc computed gotos for NEXT
  - Avoids having to code x86-32 + x86-64 manually
- Kernel uses ~250 lines Javascript for simple I/O
- Google Accounts used to manage access to cloud storage
- Abandoned previous ColorForth based implementation
  - communication model to NaCl changed before release
  - it was only x86-32
  - given alignment constraints, its subroutine threading was probably slower than indirect threading

Moving Parts



Future Directions

- Implement PPAPI bindings
  - Allows 2d/3d graphics + sound
  - Allows direct HTTP
- Generate platform specific shims (create .EXEs etc for download)
- Turn into a per-page extension (allow inline Forth on webpages)
- Resurrect dynamic code generation for CODE words or subroutine-threading / optimization

Demo / Code Tour

Questions?