International
JavaScript
Conference

Node.js Add-ons for High Performance Numeric Computing

Athan Reines | @kgryte | @stdlibjs

Survey

Overview

1. Intro
2. Toolchain
3. Numeric Computing
4. Basic Example
5. BLAS
6. Performance
7. Challenges
8. N-API
9. Conclusions

Native Add-ons

Interface between JS running in Node.js and C/C++ libraries

APIs

- V8
- libuv
- Internal Libraries
- Dependencies

Examples

- leveldown
- node-nanomsg
- node-canvas
- node-sqlite3
- stdlib

Why?

- Leverage existing codebases
- Access lower-level APIs
- Non-JavaScript features
- Performance

Toolchain

node-gyp

GYP

Challenges

- V8
- NAN
- GYP
- Engine Bias

Numeric Computing

Basic Example

```c
/* hypot.h */
#ifndef C_HYPOT_H
#define C_HYPOT_H

#ifdef __cplusplus
extern "C" {
#endif

double c_hypot( const double x, const double y );

#ifdef __cplusplus
}
#endif

#endif
```

```c
/* hypot.c */
#include <math.h>
#include "hypot.h"

double c_hypot( const double x, const double y ) {
    double tmp;
    double a;
    double b;
    if ( isnan( x ) || isnan( y ) ) {
        return NAN;
    }
    if ( isinf( x ) || isinf( y ) ) {
        return INFINITY;
    }
    a = x;
```

```cpp
/* addon.cpp */
#include <nan.h>
#include "hypot.h"

namespace addon_hypot {

    using Nan::FunctionCallbackInfo;
    using Nan::ThrowTypeError;
    using Nan::ThrowError;
    using v8::Number;
    using v8::Local;
    using v8::Value;

    void node_hypot( const FunctionCallbackInfo<Value>& info ) {
        if ( info.Length() != 2 ) {
```

```
# binding.gyp
{
  'targets': [
    {
      'target_name': 'addon',
      'sources': [
        'addon.cpp',
        'hypot.c'
      ],
      'include_dirs': [
                '<!(node -e "require(\'nan\')")',
                './'
      ]
    }
  ]
```

```
# Navigate to add-on directory:
$ cd path/to/hypot/binding.gyp

# Generate build files:
$ node-gyp configure

# On Windows:
# node-gyp configure --msvs_version=2015

# Build add-on:
$ node-gyp build
```

```
/* hypot.js */
var hypot = require( './path/to/build/Release/addon.node' ).hypot;

var h = hypot( 5.0, 12.0 );
// returns 13.0
```

	ops/sec	perf
Builtin	3,954,799	1x
Native	4,732,108	1.2x
JavaScript	7,337,790	1.85x

BLAS

```
! dasum.f
! Computes the sum of absolute values.
double precision function dasum( N, dx, stride )
  implicit none
  integer :: stride, N
  double precision :: dx(*)
  double precision :: dtemp
  integer :: nstride, mp1, m, i
  intrinsic dabs, mod
  ! ..
  dasum = 0.0d0
  dtemp = 0.0d0
  ! ..
  if ( N <= 0 .OR. stride <= 0 ) then
    return
```

```
! dasumsub.f
! Wraps dasum as a subroutine.
subroutine dasumsub( N, dx, stride, sum )
  implicit none
  ! ..
  interface
    double precision function dasum( N, dx, stride )
      integer :: stride, N
      double precision :: dx(*)
    end function dasum
  end interface
  ! ..
  integer :: stride, N
  double precision :: sum
  double precision :: dx(*)
```

```
/* dasum_fortran.h */
#ifndef DASUM_FORTRAN_H
#define DASUM_FORTRAN_H

#ifdef __cplusplus
extern "C" {
#endif

void dasumsub( const int *, const double *, const int *, double * );

#ifdef __cplusplus
}
#endif

#endif
```

```
/* dasum.h */
#ifndef DASUM_H
#define DASUM_H

#ifdef __cplusplus
extern "C" {
#endif

double c_dasum( const int N, const double *X, const int stride );

#ifdef __cplusplus
}
#endif

#endif
```

```c
/* dasum_f.c */
#include "dasum.h"
#include "dasum_fortran.h"

double c_dasum( const int N, const double *X, const int stride ) {
    double sum;
    dasumsub( &N, X, &stride, &sum );
    return sum;
}
```

```cpp
/* addon.cpp */
#include <nan.h>
#include "dasum.h"

namespace addon_dasum {

    using Nan::FunctionCallbackInfo;
    using Nan::TypedArrayContents;
    using Nan::ThrowTypeError;
    using Nan::ThrowError;
    using v8::Number;
    using v8::Local;
    using v8::Value;

    void node_dasum( const FunctionCallbackInfo<Value>& info ) {
```

```
$ gfortran \
    -std=f95 \
    -ffree-form \
    -O3 \
    -Wall \
    -Wextra \
    -Wimplicit-interface \
    -fno-underscoring \
    -pedantic \
    -fPIC \
    -c \
    -o dasum.o \
    dasum.f
$ gfortran \
    -std=f95 \
```

```
# binding.gyp
{
  'variables': {
    'addon_target_name%': 'addon',
    'addon_output_dir': './src',
    'fortran_compiler%': 'gfortran',
    'fflags': [
      '-std=f95',
      '-ffree-form',
      '-O3',
      '-Wall',
      '-Wextra',
      '-Wimplicit-interface',
      '-fno-underscoring',
      '-pedantic',
```

```
# binding.gyp (cont.)
  'targets': [
    {
      'target_name': '<(addon_target_name)',
      'dependencies': [],
      'include_dirs': [
        '<!(node -e "require(\'nan\')")',
        '../include',
      ],
      'sources': [
        'dasum.f',
        'dasumsub.f',
        'dasum_f.c',
        'addon.cpp'
      ],
```

```
# binding.gyp (cont.)
      'rules': [
        {
          'extension': 'f',
          'inputs': [
            '<(RULE_INPUT_PATH)'
          ],
          'outputs': [
            '<(INTERMEDIATE_DIR)/<(RULE_INPUT_ROOT).<(obj)'
          ],
          'conditions': [
            [
              'OS=="win"',
              {
                'rule_name': 'compile_fortran_windows',
```

```
# binding.gyp (cont.)
    {
      'target_name': 'copy_addon',
      'type': 'none',
      'dependencies': [
        '<(addon_target_name)',
      ],
      'actions': [
        {
          'action_name': 'copy_addon',
          'inputs': [],
          'outputs': [
            '<(addon_output_dir)/<(addon_target_name).node',
          ],
          'action': [
```

```
$ cd path/to/dasum/binding.gyp
$ node-gyp configure
# node-gyp configure --msvs_version=2015
$ node-gyp build
```

```
/* dasum.js */
var dasum = require( './path/to/src/addon.node' ).dasum;

var x = new Float64Array( [ 1.0, -2.0, 3.0, -4.0, 5.0 ] );
var s = dasum( x.length, x, 1 );
// returns 15.0
```

Length	JavaScript	Native	Perf
10	22,438,020	7,435,590	0.33x
100	4,350,384	4,594,292	1.05x
1,000	481,417	827,513	1.71x
10,000	28,186	97,695	3.46x
100,000	1,617	9,471	5.85x
1,000,000	153	873	5.7x

```
/* dasum_cblas.h */
#ifndef DASUM_CBLAS_H
#define DASUM_CBLAS_H

#ifdef __cplusplus
extern "C" {
#endif

double cblas_dasum( const int N, const double *X, const int stride );

#ifdef __cplusplus
}
#endif

#endif
```

```c
/* dasum_cblas.c */
#include "dasum.h"
#include "dasum_cblas.h"

double c_dasum( const int N, const double *X, const int stride ) {
    return cblas_dasum( N, X, stride );
}
```

```
# binding.gyp
{
  'variables': {
    'addon_target_name%': 'addon',
    'addon_output_dir': './src',
  },
  'targets': [
    {
      'target_name': '<(addon_target_name)',
      'dependencies': [],
      'include_dirs': [
        '<!(node -e "require(\'nan\')")',
        './../include',
      ],
      'sources': [
```

Length	JavaScript	wasm	Native	Perf
10	22,438,020	18,226,375	7,084,870	0.31x
100	4,350,384	6,428,586	6,428,626	1.47x
1,000	481,417	997,234	3,289,090	6.83x
10,000	28,186	110,540	355,172	12.60x
100,000	1,617	11,157	30,058	18.58x
1,000,000	153	979	1,850	12.09x

Challenges

- Bugs
- Standards
- Proprietary
- Windows
- Portability
- Complexity

Modularity

```
{
    "options": {
        "os": "linux",
        "blas": "",
        "wasm": false
    },
    "fields": [
        {
            "field": "src",
            "resolve": true,
            "relative": true
        },
        {
            "field": "include",
            "resolve": true,
```

N-API

Features

- Stability
- Compatibility
- VM Neutrality

```cpp
/* addon.cpp */
#include <node_api.h>
#include <assert.h>
#include "hypot.h"

namespace addon_hypot {

    napi_value node_hypot( napi_env env, napi_callback_info info ) {
        napi_status status;

        size_t argc = 2;
        napi_value argc[ 2 ];
        status = napi_get_cb_info( env, info, &argc, args, nullptr, r
        assert( status == napi_ok );
```

Conclusions

- Parity
- Performance
- Progress

Thank you!

https://github.com/stdlib-js/stdlib

https://www.patreon.com/athan

APPENDIX

THE END