APEX APL Compiler Status

Robert Bernecky
Snake Island Research Inc
Toronto, Ontario
Canada

Overview

- Potential Applications
- Introduction to APEX
- Supported Dialect & Features
- Performance
- Problem Areas
- Licensing
- Future Work
- Credits and Support
- Summary

Potential Applications

- High-performance APL-based apps
- Application certification
- Packaged binaries (no APL run-time required)
- Cross-platform code (Mac, Linux, Solaris, Unix)
- Code obfuscation
- Application development for Digital Signal Processors (DSP) & embedded systems

Introduction to APEX

- A research compiler, designed to facilitate exploration of array language optimizations
  - Easy to extend/change
  - Not yet production quality (like Windows...)
  - Compile time not a major concern
- Parallel, multi-thread run-time support
- Runtime: Generates SAC->C for Linux/Solaris/Mac (Old version was SISAL->C)
- APEX runs under APL+Unix

SISAL vs. SAC

- Both are high-performance (faster than Fortran or C!), parallel array languages
- SISAL is vector-of-vectors:
  - Transpose transpose 5 0 reshape 0
- SAC is true array language
- SISAL: moribund: LLNL is building bombs again
- SAC: Recovering from major SSA surgery – many optimizations still disabled

Supported Dialect

- Flat ISO APL (no nested arrays)
- No execute
- System function support via #pragmas, libraries (E.g., GetFile)
- No goto (but does have :for)
- Some primitives not coded yet
- Some syntax rules not coded yet
- A few declarations required

Language Extensions

- Familiar to SHARP APL & J users
- Extended take/drop, e.g.,
  - 3 take matrix
- Cut conjunction
  - Windowed and partitioned operations
- Rank conjunction – consistent and powerful
- Replicate, e.g.,
  - 2 0 3 4 /'abcd' -> aacccddddd

Almost Unique Features

- Compositions, e.g.,
  - Monadic matrix product:
    - mdot <-filter & +.*
- Monadic window-reduce, e.g.,
  - Convolution:
    - (rho filter) mdot / seismictrace
  - String search
    - (rho string) string&^.=/ text

Performance and Correctness

- 85 benchmarks in test suite
  - Today, 17 compile and execute correctly
  - Some primitives missing (membership, scan, base value...)
  - At least one dataflow analysis bug, one SSA bug
  - Function cloning not implemented yet
  - As of last week, nothing compiled correctly...
- Performance OK, but not earthshaking
- Earthshaking to follow

Earthshaking

- Performance vs interpreted APL
  - Today: 0.5X->150X faster
  - Tomorrow: One missing SAC optimization gives 12X speedup on a slow benchmark
  - Next month: up to 1000X faster
- Improved SAC compiler optimizations
- Improved APEX data-flow analysis
- Improved APEX run-time algorithms
- Automatic parallelism (SMT, MP, multi-core)
- One-bit Boolean support in SAC needed

Application Certification via APEX

- Similar to materials testing in mechanical engineering
- Static analysis of application code
  - NO execution!
- All type domain errors detected ('X'+4)
- All rank errors detected
- Many length errors detected
- Some index errors detected

Problem Areas

- Integer overflow
  - 64-bit ints will help in practice
  - Otherwise, use declarations
- Definition of “floor double”
  - Is result double or int?
  - If int, possible wrong answer
- Still need coercions
  - How do I get “floor double” to coerce to integer?
- Efficient nested array support: NOT easy!

Problem Areas

- Integer overflow
  - 64-bit ints will help in practice
  - Otherwise, use declarations
- Definition of “floor double”
  - Is result double or int?
  - If int, possible wrong answer
- Still need coercions
  - How do I get “floor double” to coerce to integer?
- Nested array support: NOT easy!

Licensing

- APEX available under GPL by year-end
- Or perhaps under “Lesser GPL” or GPL 2.0?
- Check www.snakeisland.com at month-end for announcement
- FREE!
- Open source

Possible Development Model

- APEX/application consulting at $X/hour
- APEX consulting at 2/3 $X/hour if results are GPL
- Volunteer or funded development
  - Join in!
  - Send money!

What you can do

- Write primitives or speed them up
- Prototype language extensions
- Port to Windoze, PlayStation
- Write .net (sigh...) code generator
- Write DSP code generator
- Write 1-bit Boolean SAC support
- Create new optimizations (APEX, SAC)
- Send Money

Future Work

- J dialect compiler, based on APEX principles
- System functions via pragmas & modules
- Nested arrays, structures (ala C)
- Support for Windows, .net
- Prototype advanced array optimizations
- Perform shape analysis research

Future Work II

- Better application certification
  - Function coloring (display of potential run-time index & length errors)
  - Improved index error detection
  - Improved length error detection
  - Potential application to other languages
- Improved SAC compiler optimizations

Credits & Support

- Walter Fil, Future Perfect
- Eric Baelen, John Walker, APL2000
- Sven-Bodo Scholz, U. Hartfordshire
- Other SAC developers (U. Lubeck, U. Hartfordshire)

Summary

- APEX – A good flat APL compiler
- Acceptable performance now
- Fantastic performance is on the way
- Automatic parallelization
- Application certification
- The end of the beginning (Thanks, Winston!)