The BABEL Language Interoperability Tool: software integration, and evolution of large scale simulation codes

Gary Kumfert,

Tamara Dahlgren, and Thomas Epperly

Lawrence Livermore National Laboratory

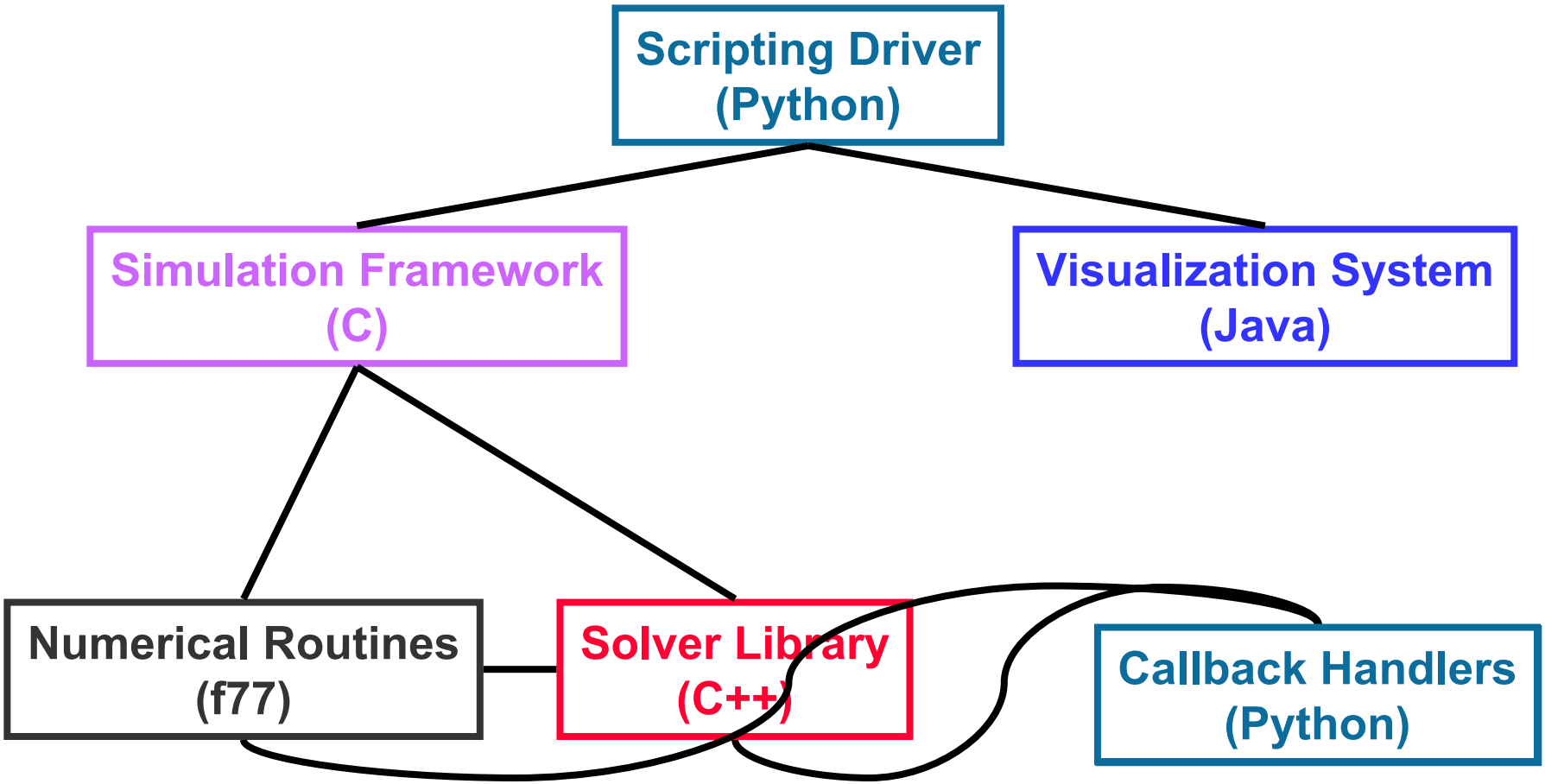
UCRL-PRES-203091

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

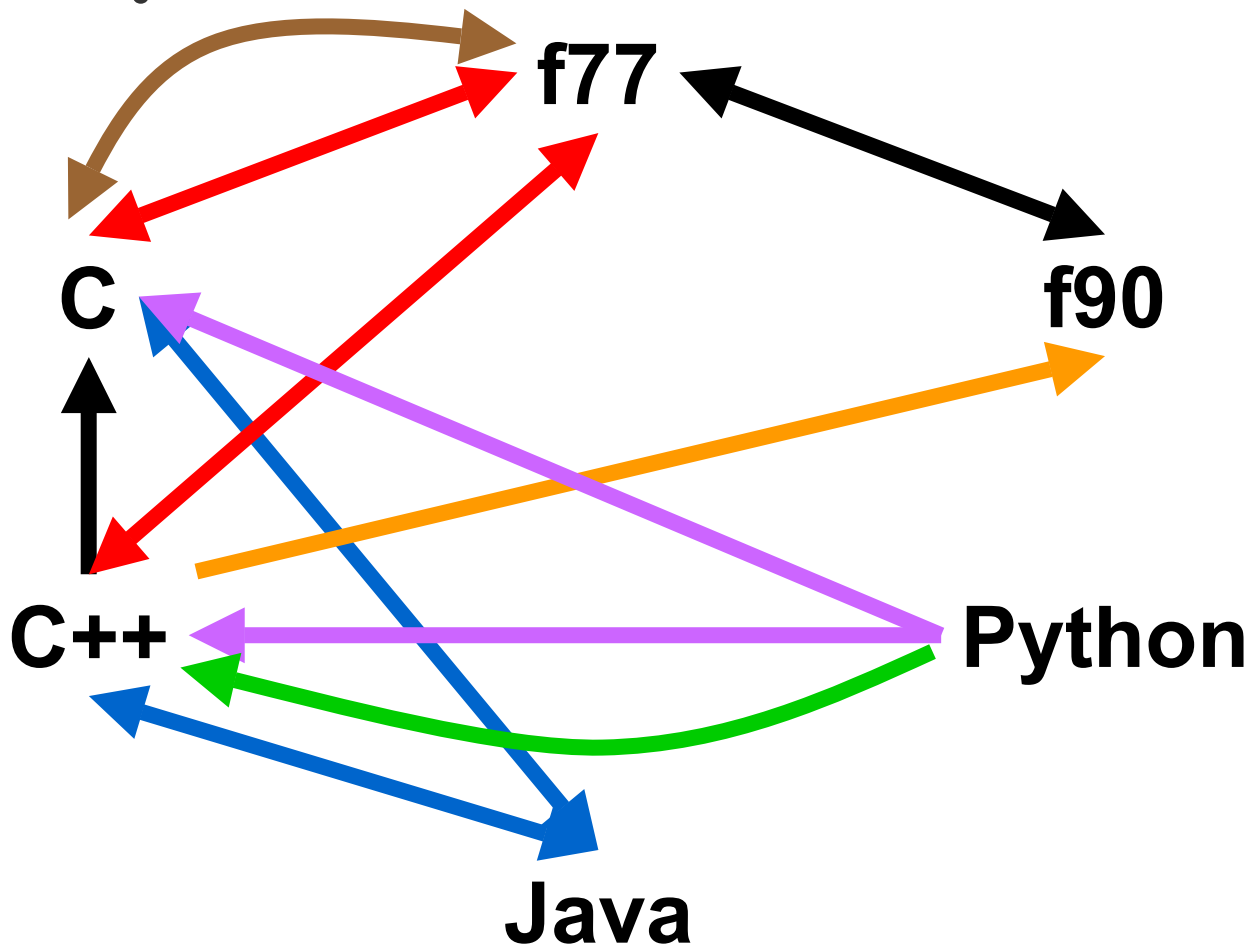
Outline

- Babel
  - Problem: Mixing Languages
  - Features
  - Performance/Overhead
- Related Work
- Large Scale Simulation Codes
  - Maintaining Correctness in face of Change
- Components, Babel, & Large Scale Simulation Software

What I mean by "Language Interoperability"



Mixing Languages: hard, not portable, and unscalable



Native

cfortran.h

SWIG

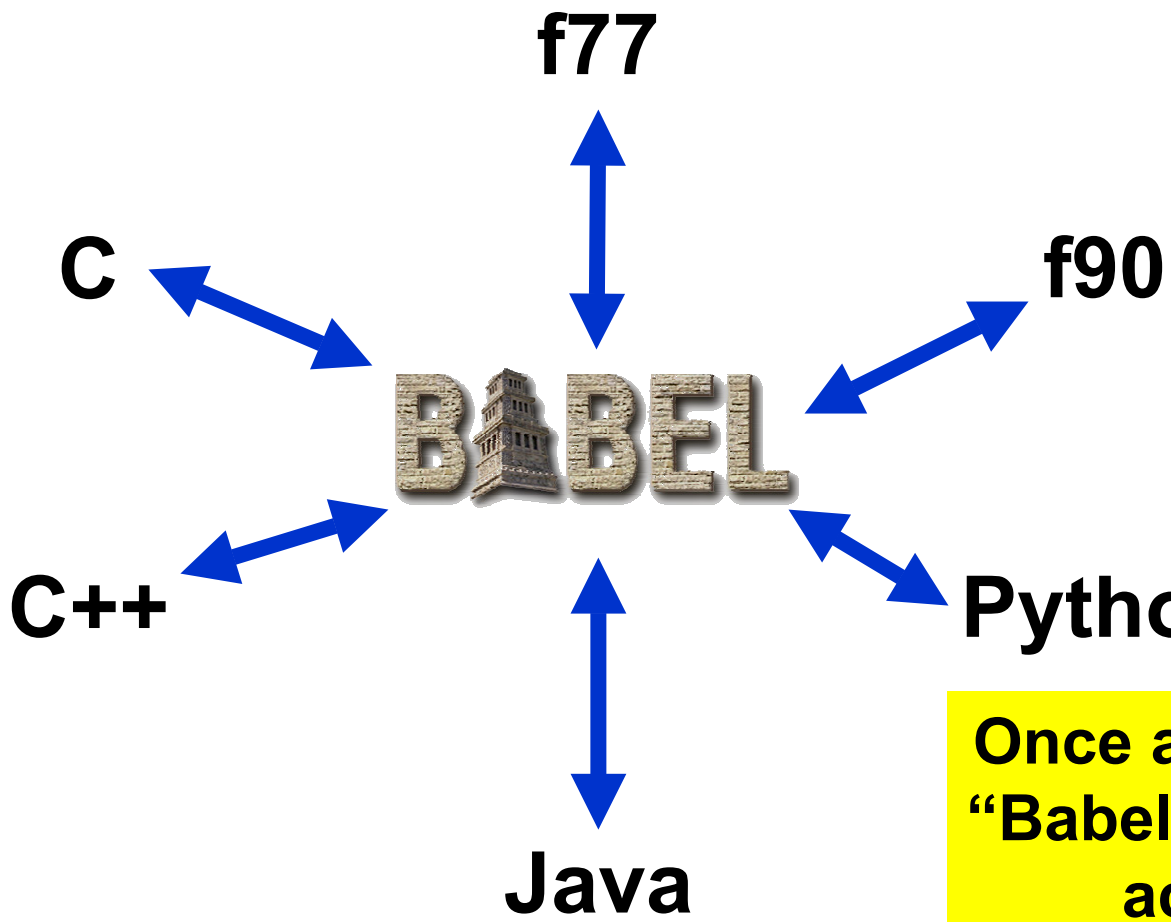
JNI

Siloon

Chasm

Platform Dependent

Babel makes all supported languages peers



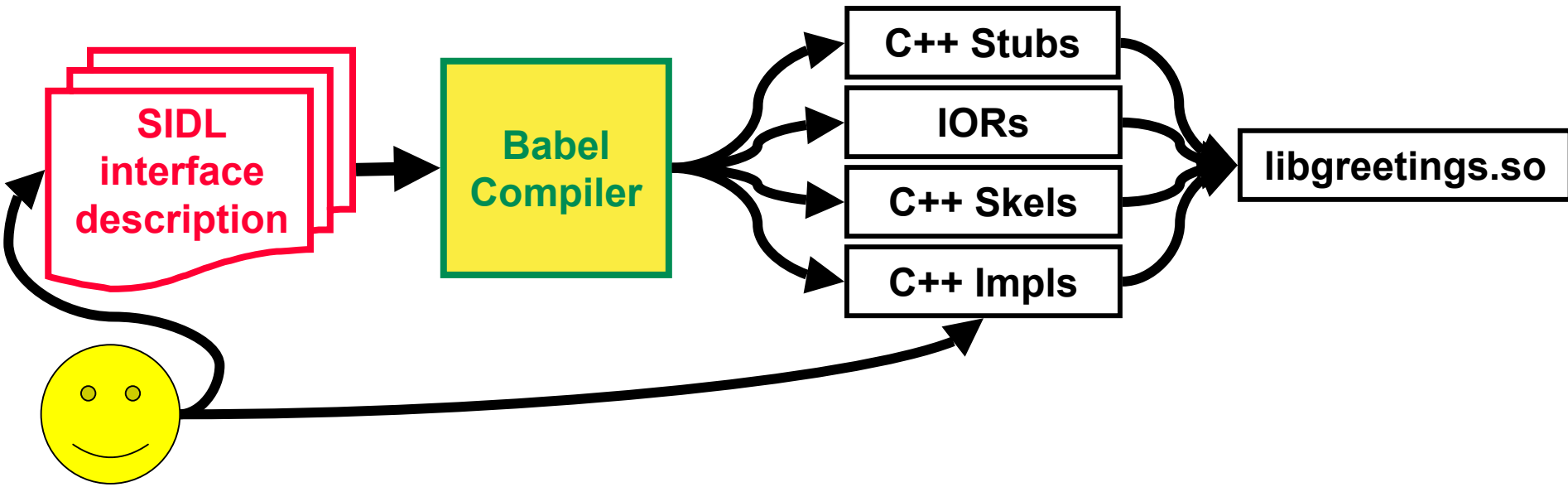
This is not a Lowest Common Denominator Solution!

Once a library has been "Babelized" it is equally accessible from all supported languages

Babel Goals and Boundaries

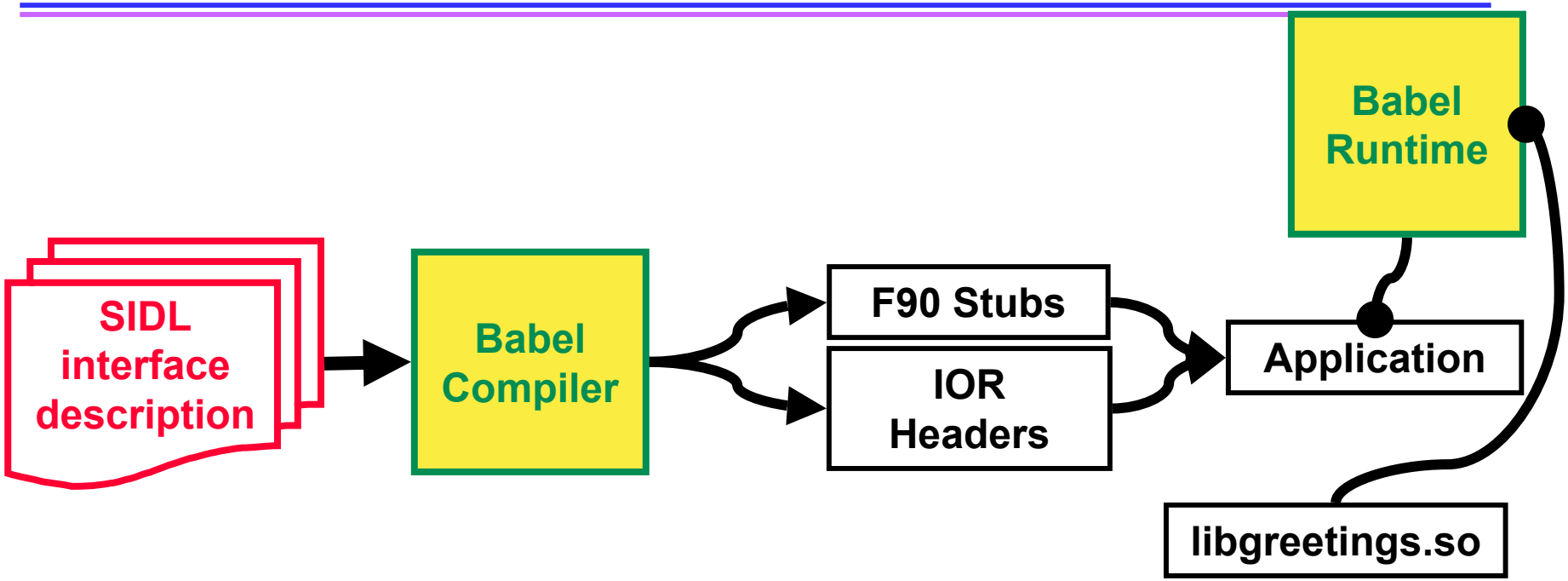
- Complete Language Transparency
- High Performance / Binary Interoperability
- Acceptable
  - Generate lots of code
  - Dictate compiler flags, etc.
- Not Acceptable
  - Require custom compilers, linkers, etc.
  - Generate code beyond language standards.

Library Developer Does This...



1. Write SIDL File
2. `babel --server=C++ greetings.sidl`
3. Add implementation details
4. Compile & Link into Library/DLL

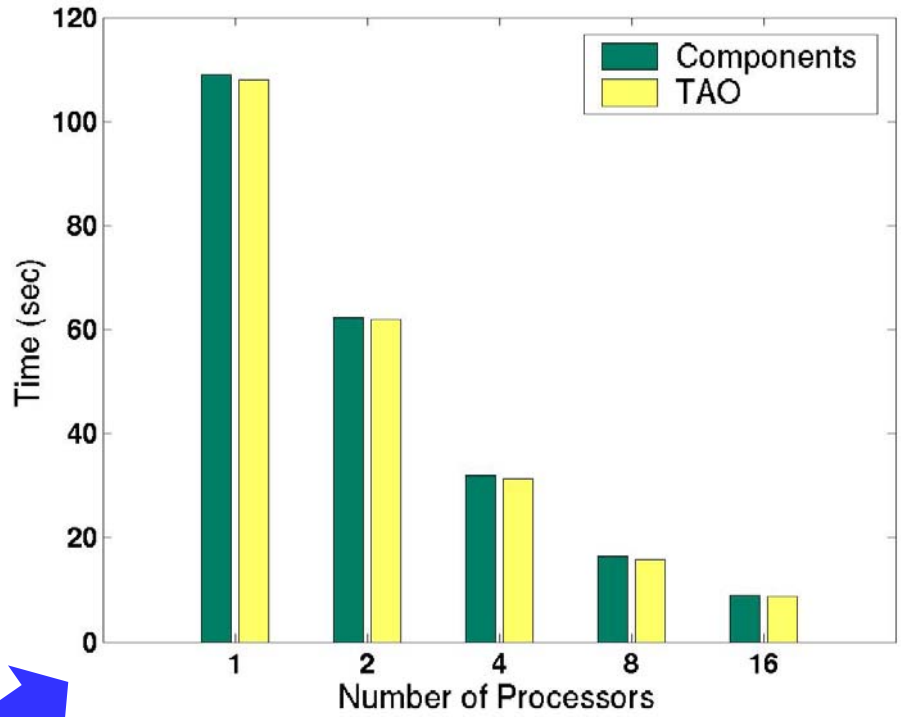
Library User Does This...



1. `babel --client=F90 greetings.sidl`
2. Compile & Link generated Code & Runtime
3. Place DLL in suitable location

Performance Impact on Whole Apps: Negligible

- ***hypre*:**
  "Lost in the noise"
  - **Kohn et. al. *Divorcing Language Dependencies from a Scientific Software Library*. SIAM PP01. Portsmouth, VA, March 12-14, 2001**
- **TAO/PETSc: "overhead of using components is negligible and it does not affect the scalability of the algorithm"**

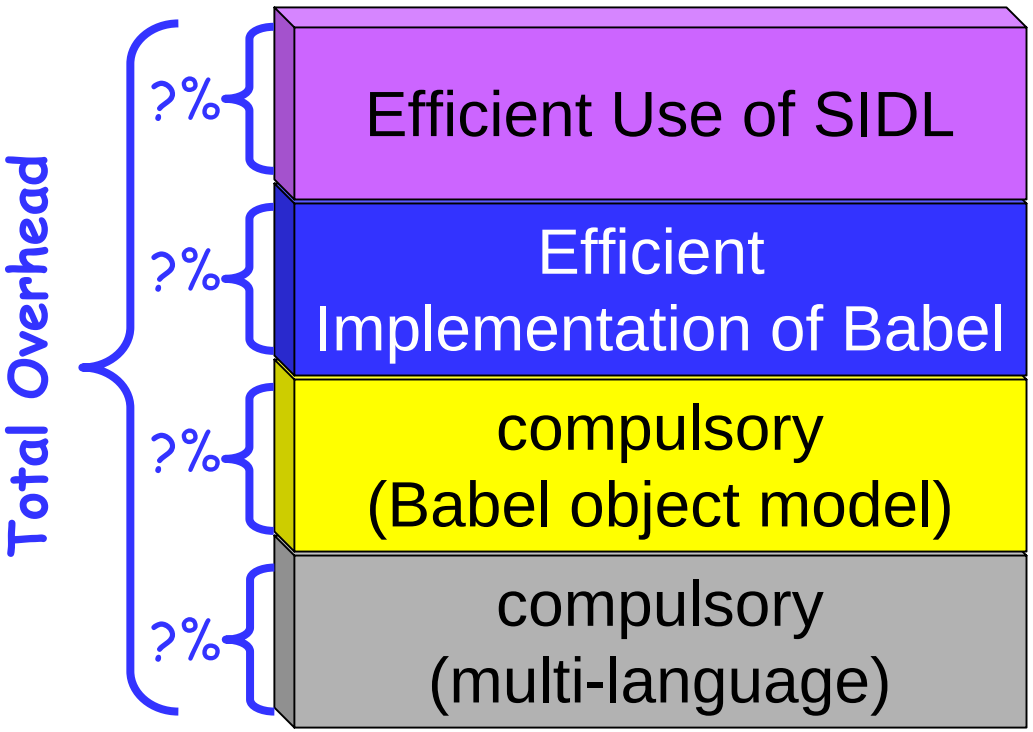


Total execution time for a surface minimization problem using a fixed-sized 250x250 mesh. Dual 550 MHz Pentium III nodes with 1-Gb of RAM each, connected with Myrinet

Overhead on Single Function Call: Small & Variable

- **Bernholdt, et. al. *A Component Architecture for High-Performance Computing*, POHLL-02 New York, NY. 22 June 2002**
  - **"avg" Babel overhead ≈ 3.8 * F77**
    - **Depends on argument modes, argument types and languages involved**
    - **All Babel calls are virtual (C++ virtual ≈ 2.2 *F77)**
  - **CORBA ≈ 25 * Babel**

Babel Performance Models: Joint work /w PERC & TSTT



also, how hard for customer to use SIDL efficiently?

not performance tuned yet

e.g. no IOR shortcut if caller & callee in same language

e.g. No C++-style inline

Bottom Line on Performance:

- Minimal overhead (nsecs/call) on a per-process basis.
  - Can construct pathological worst cases
  - Yet to see real-world example where Babel was “too heavy weight”
- No effect on parallel scalability
  - Communication latencies dominate
  - Hypre cannot measure Babel overheads on a modest parallel run.

Outline

- Babel
  - Problem: Mixing Languages
  - Features
  - Performance/Overhead
- Related Work
- Large Scale Simulation Codes
  - Maintaining Correctness in face of Change
- Components, Babel, & Large Scale Simulation Software

Other IDL Projects In Scientific Computing

- **ASE: Argonne SIDL Environment**
  - http://www.mcs.anl.gov/ase
  - Knepley and Smith @ Argonne
  - Based on Babel-0.6 (Dec'01)
  - Foundation for PETSc 3.0
- **PIDL: Parallel Interface Definition Language**
  - http://www.cs.utah.edu/~damevski/thesis.pdf
  - Damevski & Parker @SCI Institute, Utah
  - C++ only
  - Parallel RMI

"Automatic" Wrapper Generators

- e.g., SWIG, Chasm, PyFort
- Parse existing code
  - Heavily rooted in a particular language
- Are not 100% automatic
  - Often need manual hints, tweeks, etc.
- Cannot wrap 100% of the existing code
  - PyFort does a subset of F90
- Great for code you don't control

SWIG v. Babel

(David Beazley @ U Chicago)

- Call from Tcl, Perl, Python, Java, Ruby, mzscheme, or Guile
- Implement in C, C++
- Reads existing code
  - Library User can do independently
  - C++ “type system”
  - Auxiliary .i files fill in details
- Better suited for fast prototyping

- Call from C, C++, F77, F90, Python, and Java
- Implement in C, C++, F77, F90, and Python
- Hand-written SIDL
  - Library Developer task (or “motivated” user?)
  - SIDL “object model”
  - SIDL is self contained, no extra hints needed
- Better suited for production use

Outline

- Babel
  - Problem: Mixing Languages
  - Features
  - Performance/Overhead
- Related Work
- Large Scale Simulation Codes
  - Maintaining Correctness in face of Change
- Components, Babel, & Large Scale Simulation Software

How Big is a "Big Code"? (lines of source)

- **Simulator for Major Systems in a Tokamak?**
- **Simulator for capsule physics in NIF?**
- **Hewlett-Packard Printer Driver?**

Thanks to Paul F. DuBois, LLNL

How Big is a "Big Code"? (lines of source)

- **Simulator for Major Systems in a Tokamak?** 300,000
- **Simulator for capsule physics in NIF?** 500,000
- **Hewlett-Packard Printer Driver?** 5,000,000

Thanks to Paul F. DuBois, LLNL

How Big is a "Big Code"? (lines of source)

- Simulator for Major Systems — 300,000

From a software engineering perspective: if the large scale simulations aren't really that big, why do they seem so difficult?

- Driver? — 500,000

5,000,000

Challenges in Scientific SW Differ from Industry

- Correctness is harder to achieve
- Domain knowledge is very specialized
- Long development times for physics
  - the rest of the code evolves rapidly.
- Users needs vary quickly
- Platforms vary quickly
- Distribution is usually source

Correctness

- Software engineering literature and commercial tools commonly assumes a static “specification”
- Unit-testing literature commonly assumes verification against textual output

Scientific Computing Software is Dominated by Change

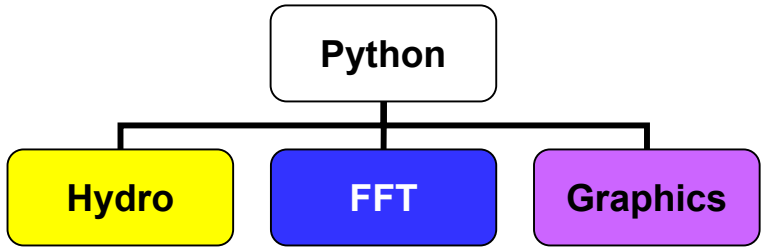
- **Scientific programs are changed much more often than programs of similar size in other fields.**
  - **A twenty-year-old LLNL program changed substantively 75 times in one year. It was not a period of major new development or a new machine.**
- **The developers are not the only ones who need to change the program – the users do too.**
- **Even the application area may change or expand.**

Thanks to Paul F. DuBois, LLNL

Change Oriented Software

- Absorb change without losing correctness
- Empower and exploit the creativity of users
- Reduce dependency entanglement among developers

Current Change-Oriented State of Art: Scripting



- Python is BIG at Livermore
- SciPy.org:
  - SWIG and PyFort shrink-wrapped codes
  - Enthought, Inc. provides the consulting services
- PyMPI
  - gives you a interactive session to parallel machine

Users Like Scripting

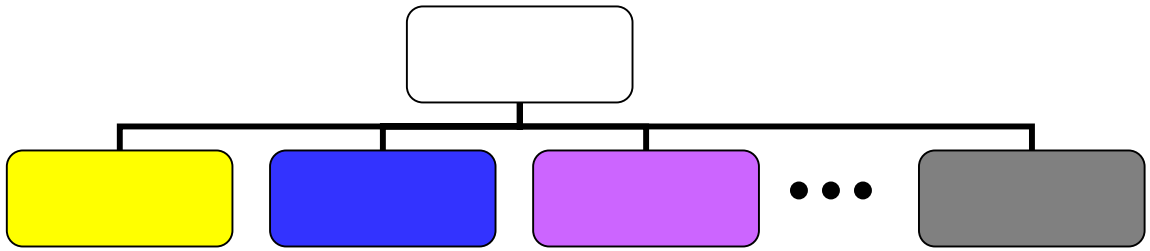
- Developers aren't a bottleneck
- Users share domain-specific expertise with each other.
- Users are much more productive
- Users enjoy coding (scripting is fun)
- If you expose the "main loop", they can add physics or modify quantities (e.g., adding noise to boundary conditions or energy deposition).

Developers Like Scripting

- Developers get built-in graphical debugger
- Prototype algorithms in interpreter.
- Many facilities get out of compiled code for good
  - If 90% of runtime is spent in 10% of code, why not script the other 90%?
- Can try new uses/configurations for existing pieces without a lot of investment

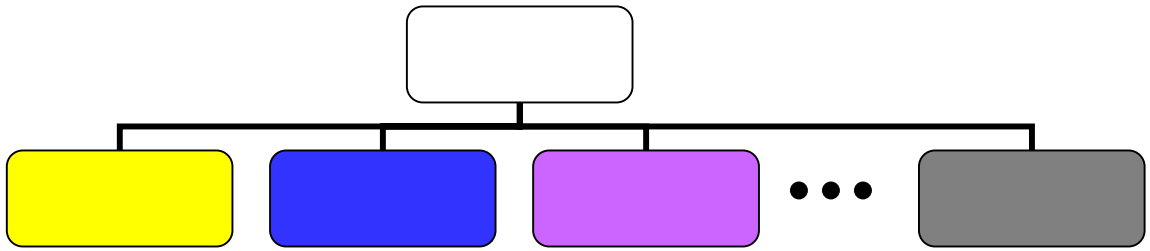
Downside of Scripting

- **Does your code look like this?**

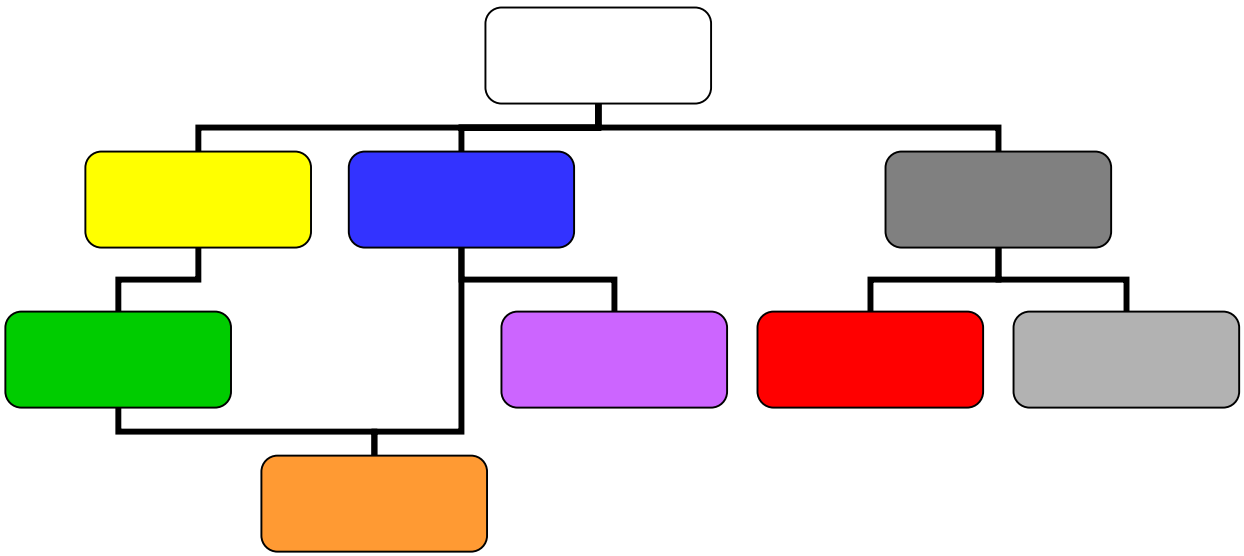


Downside of Scripting

- Does your code look like this?



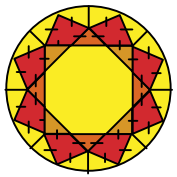
- Or like this?



Outline

- Babel
  - Problem: Mixing Languages
  - Features
  - Performance/Overhead
- Related Work
- Large Scale Simulation Codes
  - Maintaining Correctness in face of Change
- Components, Babel, & Large Scale Simulation Software

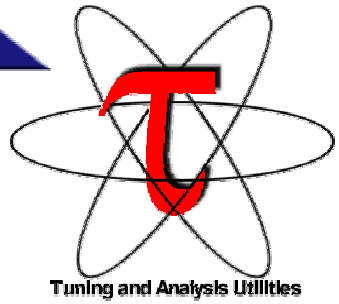
Babel is a funded part of the CCA

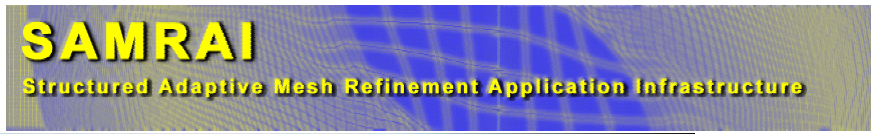


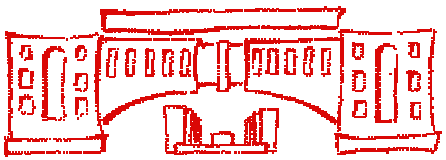
CCA
Common Component Architecture

I implemented a Babel-based interface for the hypre library of linear equation solvers. The Babel interface was straightforward to write and gave us interfaces to several languages for less effort than it would take to interface to a single language.

--Jeff Painter, LLNL.







research.cs.vt.edu/lacsa

Software Components: Commercial vs. Computational

- Industry developed component technology to
  - increase reuse
  - control costs
  - scale to large systems
- Large Scale Simulation needs it for
  - integration of small systems to large ones
  - amenability to change
  - manage correctness in the face of change

Babel's Contributions to Change-Oriented Software

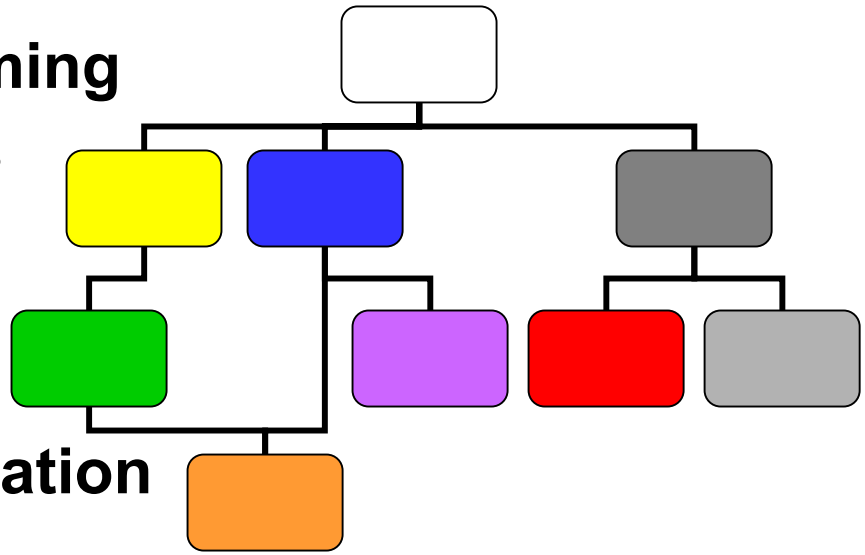
- SIDL
  - Compilable Software Contract btwn developer and user
  - Language Independent Standards
    - CCA Specification in SIDL
  - Version Management of Interfaces
  - Ongoing Research: Adding semantic specifications

Babel's Contributions to Change-Oriented Software

- **Language Transparent Software**
  - **Keeps implementation details from driving the design**
  - **Lowers integration barriers**
- **Stories:**
  - **Babel helps NWChem mix F77 w/ F77**
  - **Babel in Adaptive Algorithm Research**

CCA's Contributions to Change-Oriented Software

- **Pure Babel**
  - still imperative programming
  - assembly of call graph is embedded in code
- **CCA**
  - separates component development from application assembly
  - application assembly can be deferred to last minute (like scripting)
  - Loosely coupled systems are inherently more changeable



For More on CCA

- CCA tutorial next month at SIAM Parallel Processing in San Francisco
- CCA Quarterly meeting Thurs-Fri in downtown Knoxville

Conclusions

- From a SE perspective: the dominant feature of Scientific Software is Change
  - Assuring correctness is also especially vexing
- Scripting is current state-of-art for change-oriented software
- Component technology is cutting-edge research, but offers more than scripting

Contact Info

- **Project:** http://www.llnl.gov/CASC/components
- **Project Team Email:** components@llnl.gov
- **Mailing Lists:** majordomo@lists.llnl.gov

  subscribe babel-users *[email address]*

  subscribe babel-announce *[email address]*