Component Integration and Optimization

LACSI Priorities and Strategies Workshop 2005

Ken Kennedy

Rice University

http://lacsi.rice.edu/meetings/internal/slides_feb05/components.pdf

LACSI

Participants

- LANL
  - Staff: Craig Rasmussen
  - Student: Christopher D. Rickett
- Rice
  - Faculty/Staff: Ken Kennedy, Bradley Broom*, Zoran Budimlic, Keith Cooper, Arun Chauhan*, Rob Fowler, Guohua Jin, Tim Harvey, Chuck Koelbel, John Mellor-Crummey, Steve Reeves, Linda Torczon
  - Students: Raj Bandyopadhyay, Alex Grosul, Mack Joyner, Cheryl McCosh, Apan Qasem, Todd Waterman, Rui Zhang, Yuan Zhao
- Tennessee
  - Faculty/Staff: Jack Dongarra, Keith Seymour
  - Students: Haihang You, Jelena Pjesivac-Grbovic,and Jeffery Chen
- Houston
  - Faculty: Lennart Johnsson
  - Students: Ayaz Ali, Purvi Shah, Haiyan Teng

Outline

- Component Integration Systems
  - Support for the maintenance and optimization of component libraries
  - High-productivity languages
- Retargetable High Performance Components
  - Automatic tuning of components for specific computing platforms
  - Design of adaptive components
- Application Drivers from LANL Weapons Program
  - Marmot, Telluride, Project A
- Previous Project, Phased Down
  - High-Level Java Optimization
    - Applicable to C++

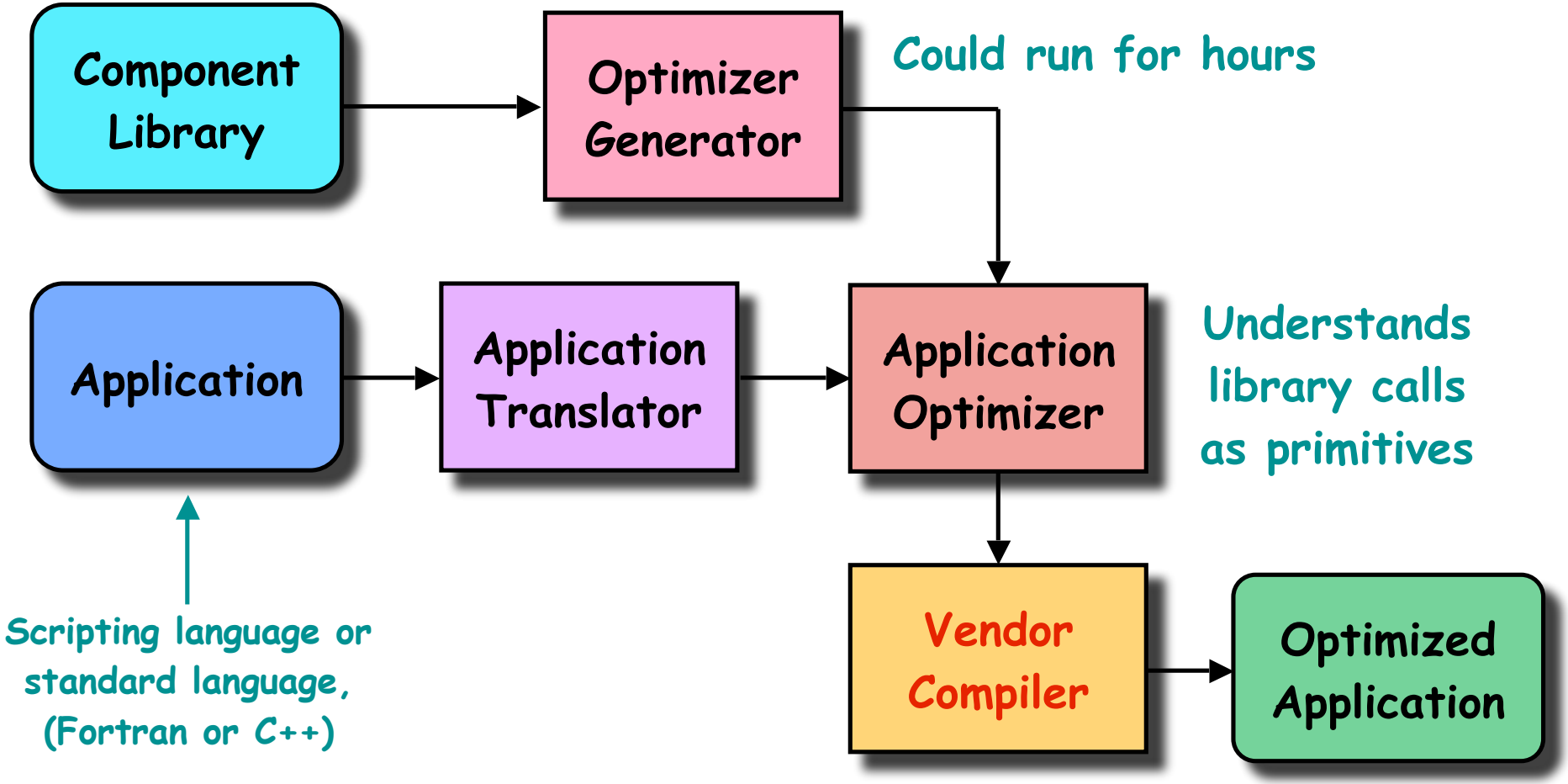
Component Integration System

- Component integration systems are important productivity tools
- Programs constructed from them can be slow
  - No context-based code improvements can be applied
- Claim: Telescoping languages can address this problem
  - Can be applied to construct component integration systems that yield high-performance applications
  - Can make components usable in contexts that have been previously considered impractical
- ASC Relevance
  - Component-based software is critical for productivity and reliability
  - Performance must be high for software to be usable
  - Useful to prototype in high-productivity language (Python, Matlab)

Component Integration Challenge

- Integration of different component libraries that
  - Implement data structures (e.g., sparse matrices)
  - Implement functions on data structures (e.g., linear algebra)
- Problem: Performance
  - High function overhead for data structure access (frequently invoked)
  - Need optimization for special contexts
    - e.g., invocation in loops
- Telescoping languages well-suited to this challenge
  - Advance generation of specialized entries
  - Transformation pass to perform substitution

Telescoping Languages



What We Have Done

- Developed base-language compiler technology
  - Type inference: Key to generation of C or Fortran from Matlab, S, or Python
    - Useful even if C++ or Fortran is your scripting language
- Conducted preliminary studies
  - Matlab SP (Signal Processing), LibGen (library generation)
    - Six papers, one Ph.D., two Master's
  - R compilation (funded separately by DOD)
- Demonstrated benefits of telescoping languages as component integration system (via LibGen)
- Developed strategy for generalized data structures
  - Including addition of parallelism to scripting languages (funded by ST-HEC program from NSF/DARPA)
- Met with Marmot Project to explore collaboration opportunities

LACSI

LACSI Interactions

- Priorities and Strategies Meetings
  - Inputs from Steven Lee and Ken Koch were pivotal in direction change
- Attended Common Component Architecture (CCA) Workshop
  - LACSI Symposium 2002
- Initial Components Workshop (April 16-17, 2003)
  - Organized by Craig Rasmussen
- Discussions with Marmot Project
  - Monterrey Methods Workshop (March 16-18, 2004)
  - Components Workshop at LANL (June 24, 2004)
    - Developed an outline plan for collaboration

What We Plan to Do

- Seek (and solve) component integration challenge problem
  - Based on work from ASC applications
  - Emphasis on efficiency of frequent component-crossing
    - Integration of data structure and function
- Continue interactions with Marmot Project
  - Goal: build tools to help them on their second or third iteration
    - Build on work on component integration and optimization of object-oriented languages
- Explore opportunities in other ASC codes
- Relevance to ASC
  - Success will make it easier to use modern component-based software development strategies in ASC codes
    - Without sacrificing performance

Automatic Component Tuning

- Participants: Four Groups within LACSI
  - —Tennessee: Jack Dongarra
    - Collaboration with LLNL ROSE Group (Dan Quinlan, Qing Yi)
  - —Rice: Ken Kennedy and John Mellor Crummey
    - Students Apan Qasem and Yuan Zhao
  - —Rice: Keith Cooper, Devika Subramanian, and Linda Torczon
    - Students Todd Waterman and Alex Grosul
  - —Univ of Houston: Lennart Johnsson
    - Students Ayaz Ali, Purvi Shah, Haiyan Teng

Automatic Component Tuning

- Goal: Pretune components for high performance on different computing platforms (in advance)
  - Models: ATLAS, UHFFT
  - Generate tuned versions automatically
- Strategy: View as giant optimization problem with code running time as objective function
  - For each critical loop nest:
    - Parameterize the search space
    - Prune using static analysis
    - Employ heuristic search to find optimal point and generate optimal code version
  - Typical optimizations:
    - Loop blocking, unroll, unroll-and-jam, loop fusion, storage reduction, optimization of target compiler settings, inlining, optimization of function decomposition

Automatic Tuning

- Successes
  - Experimental infrastructure
    - LoopTool, MSCP, ATLAS2, CODELAB
  - Large-scale experiments
  - Principles demonstrated
    - Effectiveness of heuristic search
  - Papers published
    - Seven refereed publications and one technical report (see web site)
- Relevance
  - Dramatically increases productivity of scientific programming
- Connections to ASC
  - Sweep3D, Marmot, Truchas, Project A

A Previous Effort

- JaMake Java Framework
  - Collaboration with CartaBlanca Project
  - Performs object inlining on arrays of objects
    - Overcomes the cost of using full OO polymorphism
    - Achieved 80% improvement on the LANL Parsek code
  - Results apply to C++ and Python
  - Attracted NSF funding, published 6 refereed papers
  - Applicable to other object-oriented languages (e.g., C++)

Plan for FY 05

- Refocus on Marmot as Component Challenge Problem
  - Interactions at Monterrey Workshop and a follow-up meeting at LANL (June 2004)
  - Abstract Mesh data structure to increase flexibility
  - Develop plan for activity by Q4 FY04
- Supporting Technologies for Component Integration
  - Transformation systems to eliminate overheads due to abstraction
  - Component integration systems to automate specialization
    - Key problem: integration of data structure components with functional components
- Retargetable High Performance Components
  - Pretuning arbitrary apps to new architectures