

Priyadarshini Malusare
Last Updated, March 10th 2009

malusare@mcs.anl.gov
(213) 590-6718

- Education**
- M.S. Computer Science, May 2006
 - University of Southern California
 - Area of Study, Computer Architecture and Compiler
 - B.E. Computer Engineering, May 2004
 - University of Pune, India
 - with distinction
 - Thesis, Compile Time Type Inferencing of Flow Sensitive Polymorphic Types.
 - Area of Study, Programming Languages
 - Polytechnic, Computer Engineering, May 2001
 - Maharashtra State Board of Technical Education, India
 - State Merit List Rank 1
- Technical Skills**
- **Compilers** GCC, IBM/XL, SUIF, Open64, ROSE/SAGE, lex, yacc
 - **Languages** C/C++, Fortran, SQL, Assembly, perl
 - **Web Technologies** Java, JSP, XML, HTML, Apache, Tomcat
 - **Scientific Computing** MPI, OpenMP, OpenAD, BlueGene/P, SPEC2006
 - **Tools** CVS, Subversion, emacs, gdb, Latex
 - **Environments** Unix, Windows, PowerPC and Solaris
- Work Experience**
- Performance Analysis of BlueGene/P dual floating point unit**
Argonne National Laboratory July 2008-current
- Tuning SPEC2006 floating point benchmarks (C/C++/Fortran) for IBM Blue Gene/P dual floating point unit in order to improve SIMD parallelization.
 - For each benchmark investigated performance bottlenecks by instrumenting and profiling code.
 - Trying to evaluate the performance of the code generated by IBM/XL compiler in terms of SIMD parallelization.
 - Applications that show poor speedup and under-utilizes the BlueGene/P's dual floating point unit are modified for better speedup.
- OpenAnalysis**
Argonne National Laboratory July 2006-July 2008
- Funded by U.S. Department of Energy
 - Developed an infrastructure to support compiler intermediate representation (IR) independent program analysis for imperative programming languages.
 - Refined general and domain-specific program analyses from 20 hours to 30 minutes for MITgcm, a numerical benchmark for studying ocean and climate
 - Fixed more than 50 bugs in the program analysis by extending the regression test framework.
 - Refined functionalities of analysis specific interface to improve programmability
 - Reimplemented OpenAnalysis abstractions in order to get precise information from IRs.
 - Implemented prototype dataflow analysis framework for analysis over control flow graphs, interprocedural control flow graphs and call graphs.

- Extending OpenAnalysis to support new domain specific analyses, adding multiple algorithms for these analysis, and providing interface for various compilers.
- Developing analysis specific interface for Open64 (Rice University) and ROSE (LLNL) compiler
- Contributing OpenAnalysis tutorial

Compiler for Transaction Coherence and Consistency Parallel Programming Model

Information Sciences Institute, University of Southern California Jan 2005- May 2006

- Funded by National Science Foundation
- Extended SUIF1 compiler intermediate representation to support transactional memory (TM) programming with a set of compiler directives.
- Implemented software TM specific optimizations on the top of SUIF
- Performed experiments on NAS and SPEC benchmarks using TCC compiler (USC) and TASSEL simulator (Stanford)

Compiler directed Array replication in Configurable Architectures

Information Sciences Institute, University of Southern California Jan 2005- Dec 2005

- Funded by National Science Foundation
- Primary research contribution was, application of array replication transformation to take advantage of flexibility of configurable architectures.
- Extended existing array data-flow analysis to identify opportunities for concurrent execution of entire loops when array sections are replicated.
- Presented experimental results for compiler analysis and array data replication algorithm for mapping set of kernel computations to a contemporary configurable device - Xilinx Virtex FPGA.

Compile Time Inferencing of Flow-Sensitive Polymorphic Types

Indian Institute of Technology, Bombay

Sep 2003-May 2004

- Extended bi-directional data flow analysis for polymorphic type inferencing
- Demonstrated that the framework for polymorphic type inferencing works for monomorphic examples without any significant modifications
- Showed static analysis of dynamic polymorphism provide an opportunity for program optimizations as well as ensures type safety

Activities

- Member of Grace Hopper Women in Computing
- Oregon Summer School, July 07
- Co-ordinated Math-Puzzles, Argonne National Lab Open House 06.