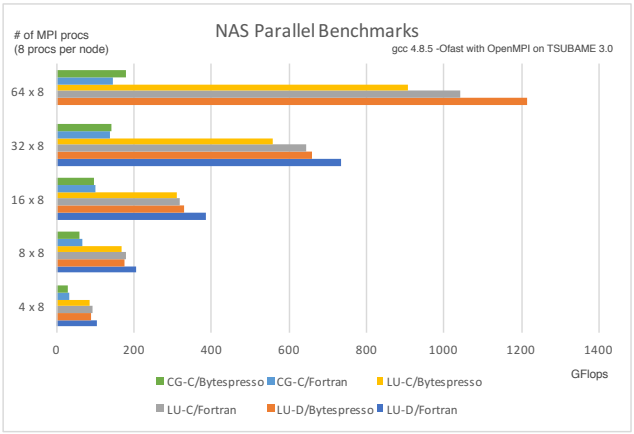
Modularity for Supercomputing - Applying modern software engineering technology to HPC

Overview

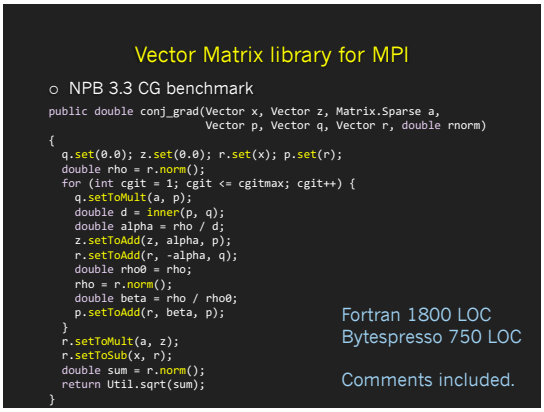
A single general programming language or framework that covers all subjects will not be feasible for post petascale supercomputing. The goal of this project is to apply modern techniques for software engineering and theoretical foundations of programming languages, such as software modularization, to supercomputing. The progress in software modularity techniques, for example, for web applications is significant in this decade. By applying these techniques, such as domain-specific languages, test driven development, object/aspect orientation, and program verification, backed by fundamental theory, the project enables domain experts to develop frameworks optimized for a specific computing platform and/or algorithms. The project thereby improves the efficiency of software development in super computing. The project also collaborates with the ExaStencil project of SPPEXA and shows that our technologies are effective in the domain of stencil computing.



Platform for Domain-Specific Languages

Bytespresso [Chiba, et al., PPPJ'16] :

It improves the execution performance of a program written in a domain-specific language (DSL) embedded in Java on HPC machines. It supports MPI and CUDA as the execution platform. A key idea developed for Bytespresso is deep reification. It enables the implementation of an embedded DSL to extract code fragments embedded in a host Java program and translate on-the-fly to a program written in a language that efficiently runs on the target platform, such as MPI-C and CUDA. An embedded DSL implemented with deep reification is categorized into implicit deep embedding, which we named, a good mixture of shallow embedding and deep embedding. Such a DSL provides intuitive algorithm expression like shallow embedding and achieves good execution performance like deep embedding.



Ikra [Springer and Masuhara, ARRAY'16]

It is a DSL that enables GPU-based array processing in Ruby. It provides an extended Array class, which is executed by the Ruby interpreter with native compiled code for array processing. Ikra's challenges are generalization of application-specific optimizations: supporting a wide range of parallel architectures and combining various optimization techniques such as communication/computation overlapping and temporal blocking. Ikra addresses these challenges by parameterized templates, which is unique against related systems based on brute-force implementations or post-processing. The methods currently supported by Ikra are map, reduce, and stencil kernels. The optimizations such as loop peeling, communication/computation overlapping, and shuffling warps are supported. The performance of Ikra programs is close to that of hand-written code and it scales up to three GPUs.

Our software systems that are publicly available:

Bytespresso

A development framework for Java-based embedded DSLs. `https://github.com/csg-tokyo/bytespresso`

Ikra

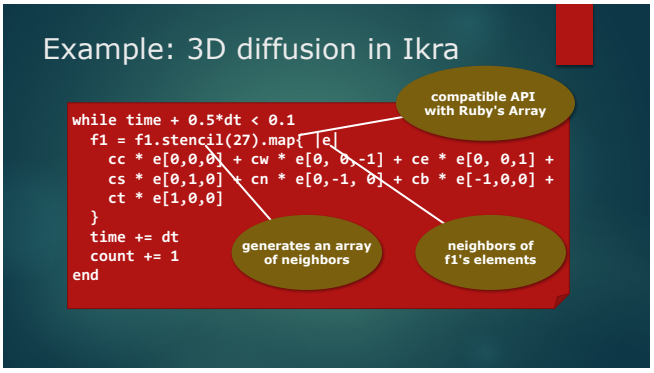
An embedded DSL (or a library) for array processing in Ruby by using GPUs.

`https://rubygems.org/gems/ikra`

VeriCUDA

A static verification system for CUDA programs on the basis of Hoare logic.

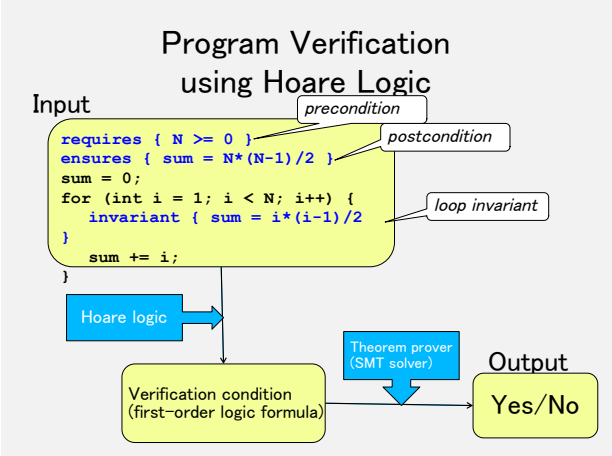
`https://github.com/SoftwareFoundationGroupAtKyotoU/Vericuda`



Verification of HPC Code

Vericuda [Kojima, et al., APLAS'13, VSTTE'16]

It can automatically verify, for example, an optimized matrix multiplication program written in CUDA. This verifier exploits Hoare logic. For automatic verification, the programmers have only to annotate loop invariants. Then Vericuda verifies that, if the given precondition is satisfied, then the post condition is also satisfied after the program execution. Vericuda is useful to discover a bug that have been sneaked while the programmers are modifying their programs for performance optimization.



HPCUnit [Sato et al., ISSTA'15]

It supports runtime verification, or a unit test for scientific-computing. It tests calculation coverage to detect duplication, leak, and out-of-order calculation bugs produced during performance optimization. A Java program using MPI can be tested by this tool. We found a duplication bug in Java Grande Benchmark suites.

Exploring new application domains

We applied HPC to software repositories mining. Nowadays, a large number of programs are publicly available on the internet. Those programs are actively investigated by researchers for discovering new software-engineering insights. This investigation is called mining software repositories. However, since the total size of the programs is huge, the computation time for the investigation has been a major obstacle to discover an interesting insight. We used a supercomputer to perform the mining. We could successfully execute the Type-3 code clone detection among the Apache 131 projects except a few extremely large source files.

Project information:

Project leader: Shigeru Chiba (Univ. of Tokyo)

Project member: Hidehiko Masuhara (Tokyo Tech), Atsushi Igrashi (Kyoto Univ.), and Naoyasu Ubayashi (Kyusyu Univ.)

Contact:
Shigeru Chiba, `chiba@acm.org`