PULSAR

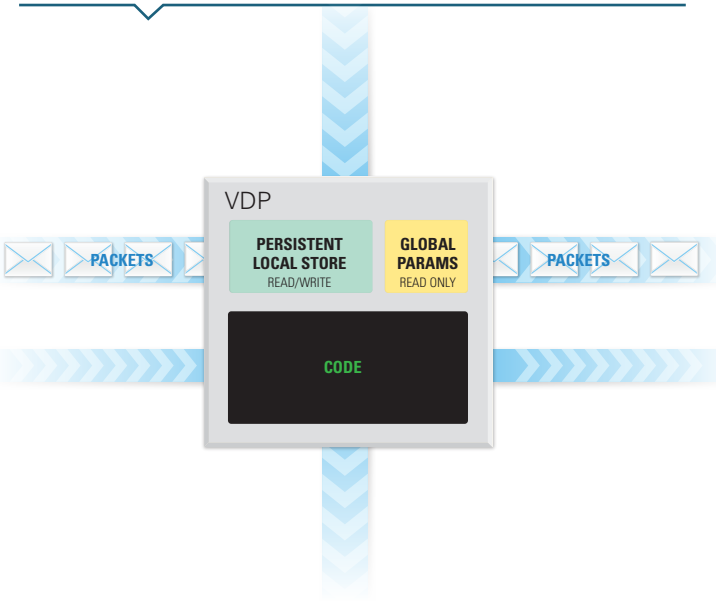
INNOVATIVE COMPUTING LABORATORY

PULSAR (Parallel Ultra Light Systolic Array Runtime) offers a simple programming model for large-scale distributed-memory machines with multicore processors and hardware accelerators. PULSAR automates multithreading, message-passing, and multi-stream multi-GPU programming.

BENEFITS OF PULSAR

- Simple Programming Model
- Lightweight Runtime System
- Multithreading & Message-passing
- Multi-stream Multi-GPU execution
- Minimum Overhead / Maximum Scalability

PROGRAMMING MODEL

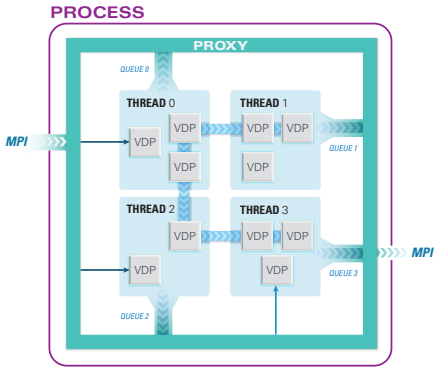


PULSAR offers a simple programming model, where the user defines the computation in the form of a Virtual Systolic Array (VSA), which is a set of Virtual Data Processors (VDPs), connected with data channels. The VDP is assigned a function, which defines its operation. Within that function, the VDP has access to a set of global parameters, its private, persistent local storage, and its channels. The runtime invokes that function when there are packets in all of the VDP's input channels. This is called firing. When the VDP fires, it can fetch packets from its input channels, call computational kernels, and push packets to its output channels.

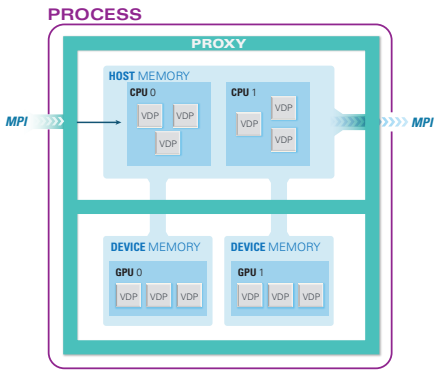
RUNTIME SYSTEM

This programming model is accessible to the user through a very small and simple Application Programming Interface (API). All the complexity of executing the workload on a large-scale system is hidden in the runtime implementation. While the user invokes simple push and pull channel operations, the runtime takes appropriate actions, depending on the boundaries crossed by the channel, i.e., uses shared memory for VDPs residing in the same node, uses message-passing for VDPs residing in different nodes, uses Direct Memory Access (DMA) transfers between CPU VDPs and GPU VDPs.

MPI MULTICORE



MPI MULTICORE ACCELERATORS



ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

SPONSORED BY

National Science Foundation

BEAST

INNOVATIVE COMPUTING LABORATORY

The goal of BEAST (Bench-testing Environment for Automated Software Tuning) is to create a framework for exploring and optimizing the performance of computational kernels on hybrid processors that:

1. **applies to a diverse range of computational kernels,**
2. **(semi)automatically generates better performing implementations on various hybrid processor architectures, and**
3. **increases developer insight into why given kernel/processor combinations have the performance profiles they do.**

We call this form of optimization "bench-tuning" because it builds on the model used for traditional benchmarking by combining an abstract kernel specification and corresponding verification test with automated testing and data analysis tools to achieve this threefold goal.

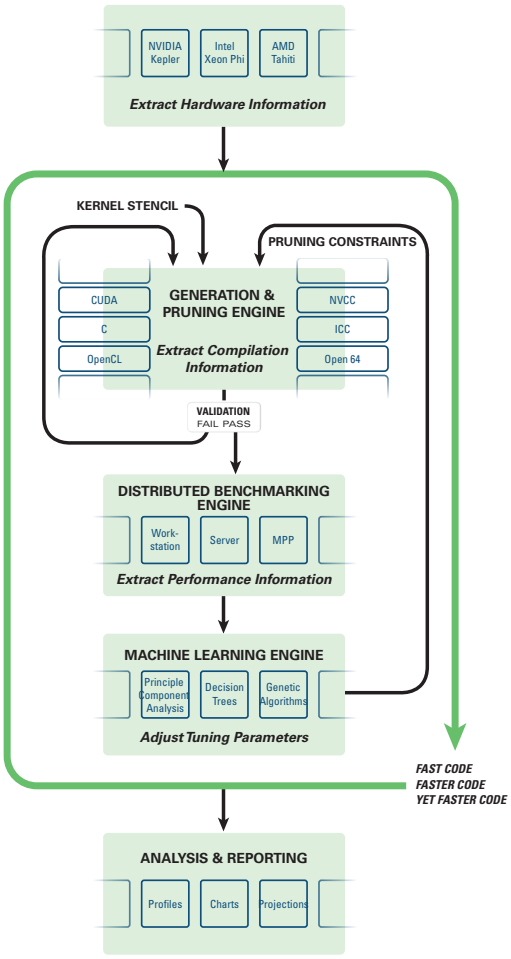
INCLUDED IN THE CURRENT RELEASE:

- A complete workflow for a tuning sweep: defining the search space, compiling and benchmarking a set of kernels
- A declarative, Python-based notation for setting up the search space using a set of iterators and pruning filters
- A build system for multithreaded compilation on a multicore machine
- A fully generalized and highly optimized implementation of matrix multiplication for NVIDIA GPUs

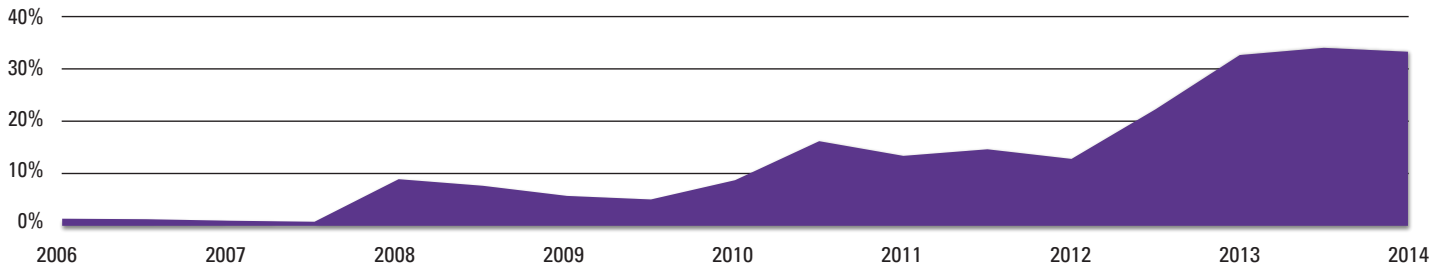
COMING SOON:

- An infrastructure for parallel benchmarking in a distributed memory environment
- An infrastructure for collection and analysis of information from hardware performance counters

BEAST ARCHITECTURE



TOP500 PERFORMANCE SHARE OF ACCELERATORS



ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

SPONSORED BY
National Science Foundation