Enhancing Application Performance with TruStream™

TruStream control structures have been designed to address the following programming challenges adding multi-core performance to existing applications:

- Already multi-threaded programs
- Serially executing, context independent, sequences of operations
- Workloads which are composed of medium or fine-grained tasks
- Programs which show excessive "kernel" mode time
- Programs which show imbalance between elapsed and (kernel + user) time
- Programs which are Memory Bound
- Programs which are CPU Bound
- Programs that match or use any of the 55 *Berkeley Par Lab Parallel Patterns*[1]
- Programs that can utilize any of the concurrency topologies listed in CORNAMI's TruStream Programming Guide.
- High Availability / Fault Tolerant

Problems not amenable to TruStream enhancements:

- GPU Bound Problems – TruStream can access more types of parallelism and use more cores than are available in GPUs
- I/O Bound Problems

Adding performance to existing applications

Identify the performance bottlenecks in:

- Application code,
- Internal standalone libraries / classes, or
- Other external software components.

If no concurrency is being used, one must determine if a concurrency topology is applicable. Example concurrency topologies and TruStream code implementations are available in CORNAMI's TruStream Programming Guide. Please note that CORNAMI has TruStream implementations for all of the 55 Berkeley Par Lab Parallel Patterns. Once a concurrency topology has been selected, re-implement the problem code segment using TruStream.

If a concurrency topology is implemented through legacy, multi-thread programming techniques:

- Profile the concurrency topology's execution to determine where the performance bottlenecks take place.
- If contention is a problem with the existing concurrency topology implementation, recode it with TruStream using the example code provided in TruStream Programming Guide. Add additional TruStream-based concurrency elements as dictated by execution profiling runs.

[1] https://patterns.eecs.berkeley.edu/?page_id=98

Process

The following flow diagram illustrates the process that CORNAMI follows to refactor existing code into the TruStream-enhanced code:

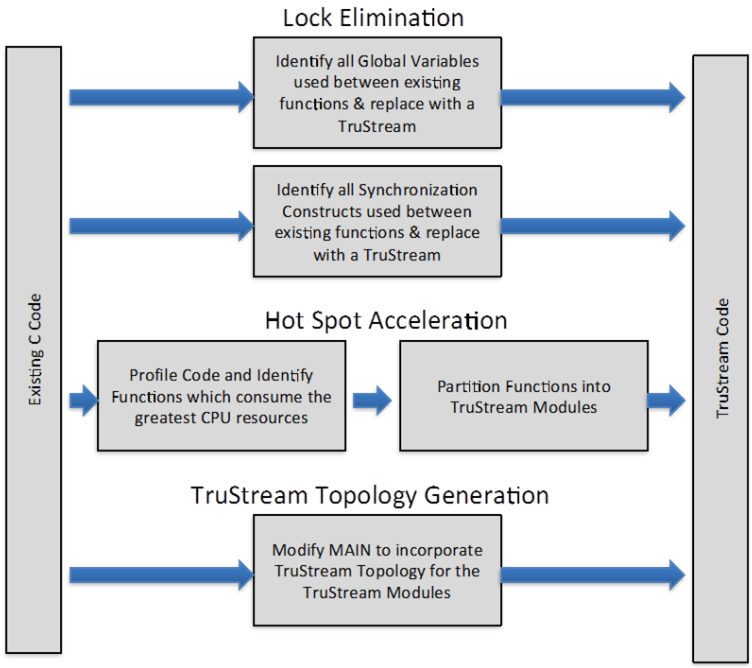


Figure 1 - TruStream Code Refactoring Process

Additional Forms of Parallelism

Pipelining

Single thread, multi-step execution

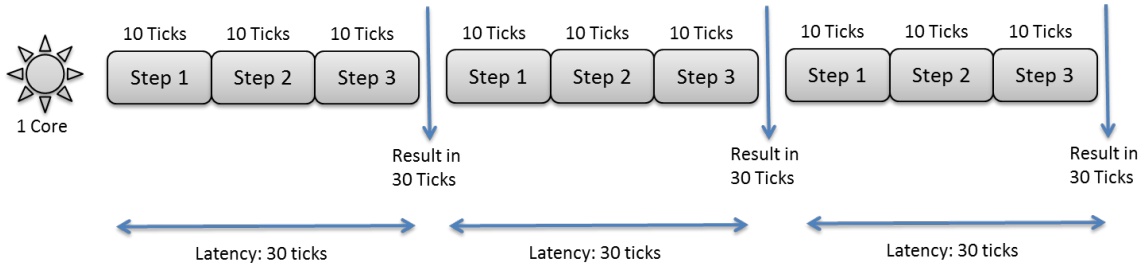


Figure 2 – Sequential Multi-Step Algorithm

But we have more processing cores now!

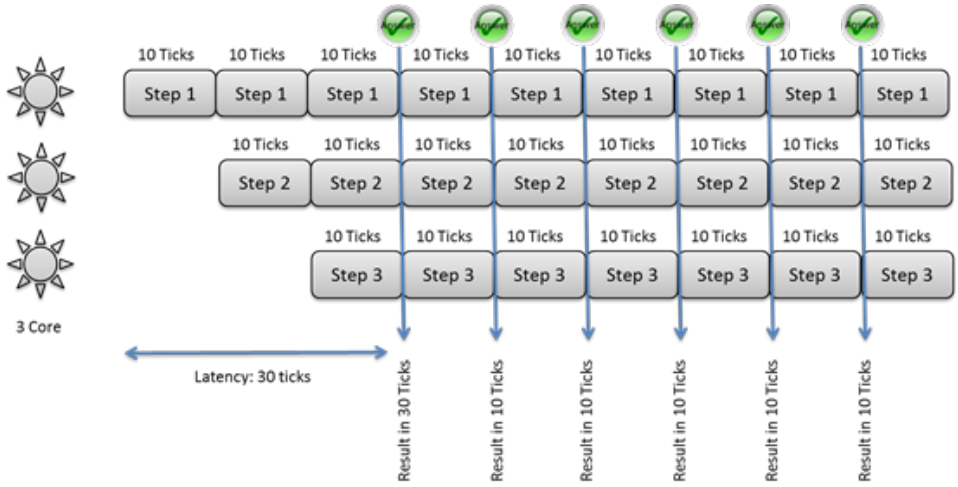


Figure 3 – Pipelined Implementation

The use of pipelining in this simple example increases the performance (as measured in throughput) by a factor of 3, after the initial latency of filling the pipeline stages has occurred. From 1 result every 30 ticks to 3 results every 30 ticks.

Systolic Arrays

Systolic Arrays are multi-dimensional pipeline structures that rhythmically compute and pass data through a system. With TruStream they become a powerful software control structure.

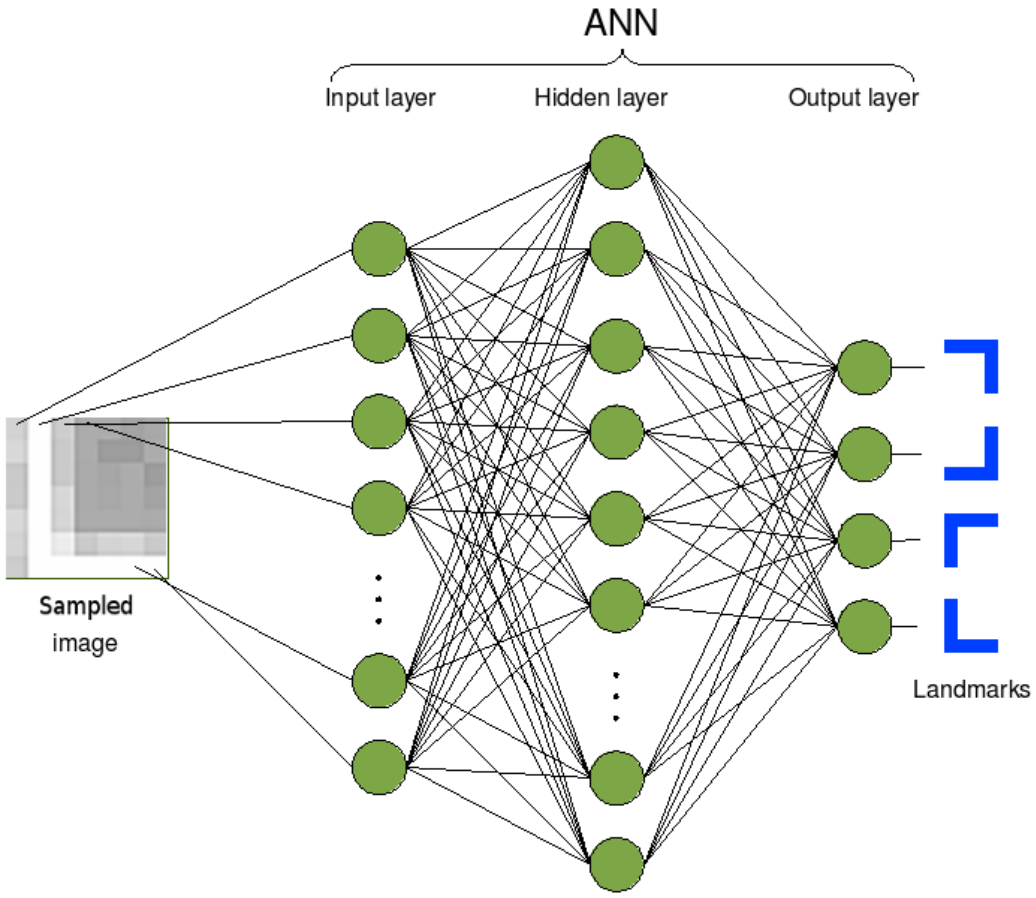
Applications:

- Linear Algebra
- Digital Signal Processing
- Image Processing
- Video Processing
- Image Recognition
- Neural Networks

Enhancing Application Performance with TruStream™

- Machine Learning
- Language Recognition
- Sorting
- Database operations
- Genetic Algorithms
- Protein Structure Prediction

Three Level Image Recognition Neural Network[2]



CORNAMI, Inc
2098 Walsh Avenue, Suite C1
Santa Clara, CA 95050
http://www.CORNAMI.net – +1 (408) 337-0070

December 13, 2016

[2] https://en.wikibooks.org/wiki/File:Artificial_neural_network_image_recognition.png