Master's Thesis Experiment Plan

Kristian Hartikainen (222956)
kristian.hartikainen@aalto.fi

June 16, 2015

The goal of this experiment is to gather data to answer the questions raised in the actual thesis. We will investigate a model of a stream computing system combining both inter-node (shared memory) and intra-node (distributed memory) level parallelism. The experiment hardware setup consists of 8 Cavium Octeon II blades, each consisting of 32 MIPS cores. The workload for the model will be chosen so that it brings out the real-life characteristics of the load-balancing. The software is modeled using OpenEM type, event based, processing where the inter-blade communication and load-balancing is implemented using future-promise type synchronization constructs.

Due to the hard nature of the setup, the focus of the experiment will be in the simulation of the system. The parameters needed for the simulation model will be gathered by several smaller experiments done with the actual hardware setup. The most important parameters considering the model are related to the communication delays between the blades and the memory latencies of the blades.

We might need more than one measurement setup to gain all the needed information for the model. Some of the simpler measurements can be done with already existing code while some of the more complex measurements require implementation of things like simple OpenEM queue mechanism and MPI and future-promise type message passing.

We need to measure how the transfer times of the raw data (packets) between the blades correspond to the varying packet sizes and packet counts. First, we will create a local area network between the Blades (running Linux) and an external packet generator. The packet generator will send packets, using a networking tool (e.g. netcat or mausezahn), to one of the blades, which then duplicates the packet and sends them to the other blades. We will measure at least the communication latencies (trasferred packets per time) as a function of packet size and packet count. The program for sending data packets between the experiment blades is almost ready, however it has to be improved such that the packet size and count is parameterizable by the user. The passthrough.c example from the Cavium SDK examples has been used as a base.

For the more complex measurements we will have to implement a program that also includes actual processing and termination of the packets. This requires

understanding of the actual memory-to-memory communication chain between the blades. For example, with a simple OpenEM queue implementations, we can measure the load-balancing overhead when the amount of data transferred exceeds the computation capacity of the blades.

One of the main questions is that, can future-promise constructs be used to extend OpenEM to work on a multi-blade system. We will need to figure out what kind of context information needs to be (can be) transferred between the blades, and also where the context is actually stored. The context most likely has a large effect to the communication delays. We can measure the throughput and latency effects future-promise communication by for example running an MPI program with bare bone future-promise implementation that simulates the real-life communication.

We will also create a simulation model of the system, using Performance Simulation Environment (PSE). The measurement results are used to amplify the simulation model, and the simulation results are compared to the results from the experiments.

The Cavium measurements will be run on top of Linux, which makes the measurements and workload handling easier. Cavium SDK offers ready-built interface for the use of performance counters. MPICH2 library and OpenEM reference implementation of OpenEM are used to guide the implementation.