Thesis Plan
Streaming Parallelism On A Multicore DSP
Week 27

Risto Vuorio (84525R)
risto.vuorio@aalto.fi

June 29, 2015

Contents

1 Summary

This thesis investigates stream processing on a multicore DSP featuring hardware accelerated scheduling. The research platform is the Texas Instruments implementation of Open Event Machine which utilizes hardware accelerated scheduling through Multicore Navigator.

In this thesis: 1. Two versions of a realistic stream processing application are implemented using PREESM and OpenEM. 2. The performance of the applications is measured under dynamic load. 3. A resource network simulation model of the OpenEM application is constructed.

2 Background

Potentially confusing concepts are explained here. The background section in the thesis will be written to explain the context to a reader who's not familiar with the problem space. The background chapter will explain 1. How the thesis fits in to context of Industrial Internet. 2. Why stream processing is an important problem. 3. What kinds of programming models are used. 4. Why DSPs are a promising candidate for stream processing. 5. ...

Stream Processing is a programming paradigm that aims to simplify the design of parallel programs. In stream processing computations happen in kernel functions. The kernel functions are applied upon each element of the input stream. Stream Processing is a high level programming model and can be implemented in a multitude of ways. Stream Processing is called uniform if all data is processed by all kernels. (NVIDIA Streaming Multiprocessor references this type of streaming.)
http://en.wikipedia.org/wiki/Stream_processing

Stream Computing is often considered in IoT or Industrial Internet context. In Stream Computing data is not in the memory as in V. Neumann architecture but in a "stream" that passes through the process. A problem or a task involving lots of continuous I/O without necessarily knowing the end of the input is a stream computing problem.

Flow, a high-level concept where computations happen when dependencies are updates. For example excel implements kind of flow computation.

Dataflow Programming models a program as a directed graph of data flowing between operations. Operations have explicitly defined inputs and outputs. Operations are carried out as soon as the inputs to that particular operation become valid. The explicit dependencies between the operations make Dataflow Programming well suited for parallel processing because when the operation becomes ready it can be executed any time on any hardware resource without further need for considering its dependencies.
http://en.wikipedia.org/wiki/Dataflow_programming

Dataflow Architecture is a hardware architecture in which the execution of

instructions is based on the availability of input arguments. The execution order of the instructions is thus indeterministic.
http://en.wikipedia.org/wiki/Dataflow_architecture

Dataflow (computing) is a software architecture based on the Flow concept. Writing software in dataflow pattern helps reduce coupling related code in programs, meaning programmers need to write less code to explicitly update values based on their dependencies.
http://en.wikipedia.org/wiki/Dataflow
http://en.wikipedia.org/wiki/Kahn_process_networks

Actor model Actors can be thought of as similar to OOP objects but with special emphasis on parallelism. Anything actors do can be done in parallel with other actors. In actor model all computations are done in actors. Actor model is based on the idea of actors sending messages to each other. As a response to receiving a message an actor can create more actors, send messages, change how to respond to next message etc. Actors can send messages at any time, but the receiver can respond whenever it is ready.
http://en.wikipedia.org/wiki/Actor_model

3 Objectives

The objective of this thesis is to understand the Open Event Machine programming model and the OpenEM runtime system in the context of stream processing. A stream processing application is implemented using OpenEM to understand the dynamic scheduling. A resource network simulation model of the OpenEM application is constructed to further understand the OpenEM runtime system and to evaluate modeling and simulation as analysis method for stream processing on load balanced platforms.

For example a DSP faces a situation where it is processing N video streams at time A. At time B there are M video streams. The problem this thesis solves is how to allocate the computation resources on the DSP dynamically so that the variable amount of streams can be processed efficiently.

4 Method

Workload construction, measurements and modeling are discussed in detail in the experiment plan.

The OpenEM programming model and runtime system are studied using **comparison** and **modeling** as the methods. A workload application is constructed with a simple runtime system (PREESM) and then converted to OpenEM runtime. The two applications are compared under different load conditions.

The OpenEM application is modeled for resource network simulation. The construction of simulation model needs detailed information about the OpenEM runtime system and the workload application. The data from the measurements is used for the timings needed in the simulation model.

To keep the implementation simple all I/O is omitted from the experiment. The omission of I/O is justified by keeping the experiment strictly focused on the computation in stream processing. The effects of an I/O layer to the studied workload are investigated through literature.

5 Material / Resources

5.1 Where's all the material?

The key material to the experiment is introduced in detail in the experiment plan. The key material can be found in the following listing.

- Workload
- PREESM
- TMS320C6678
- Analysis
- Code Composer Studio v5
- OpenEM

5.2 I/O

The thesis will include a section about I/O in context of stream processing, virtualized I/O and the omission of I/O from the thesis experiment. The importance of I/O to the stream processing and I/O virtualization was explored in ESG study circle.

5.3 People

Vesa Hirvisalo - instructor
Jussi Hanhirova - PHD student researching similar subjects
Kristian Hartikainen - research on similar subjects

6 Challenges

Risto is currently an IEM (TuTa) student. Switching to CS is only a matter of time now. All the necessary paperwork has been signed and submitted. The thesis topic can be officially approved only after Risto has started at CSE which will happen in the beginning of September.

7 Work Plan

7.1 First iteration

The PREESM workload has been implemented. The current focus is on instrumentation of the applications. After instrumentation is understood well enough and some usable performance data can be captured, the focus shifts to OpenEM.

Hopefully most of the implementation of the applications is completed by week 30. After both applications are implemented and instrumented, the simulation model will be constructed. The construction of simulation model will be interleaved with some of the measurements because the precise measurements needed are dictated by the simulation model.

The goal of the first iteration is to create a working experiment setup which yields some measurement results.

7.2 Second iteration

The experiment will be refined in the second iteration. Unnecessary parts of the experiment are abandoned and the rest are improved to produce publishable results. The construction of the simulation model in the end of first iteration will help shape the experiment to a better form. After the second iteration we should have a simulation model based on measurement data.

8 Schedule

Week 25	Planning the experiment, studying virtual I/O **done**
Week 26	Implement the workload with PREESM **done**
Week 27	Studying the instrumentation for the performance measurements. Instrumenting the PREESM application.
Week 28	Begin the workload implementation with OpenEM
Week 29	OpenEM workload implementation completed
Week 30	Vacation
Week 31	Instrumentation of OpenEM application and Measurements
Week 32	Modeling
Week 33	Measurements
Week 34	Second iteration and Writing
Week 35	Second iteration and Writing
Week 36	Second iteration and Writing
Week 37	The contribution and most of the writing should be done by week 37. Risto has a lot of school in the fall so hopefully only simple additions and editions after week 37.

9 Dissemination

ParallaX2 / ParallaX3 OpenEM streaming vs. PSE

Appendices

A Thesis structure

- Abstract
- Preface
- Contents
- Abbreviations

1. Introduction
   Research Problem
   Contributions
   Structure
2. Background (subject to change until implementation is completed)
   Industrial Internet
   Streams
   DSPs
3. Analysis of Parallelism
   Static Analysis
   Dynamic Analysis
   Tools
   Modeling and Simulation
4. Texas Instruments TMS320C6678
   Overview
   Multicore Navigator in TMS320C6678
5. OpenEM
   Runtime System
   OpenEM programming model
   Texas Instruments OpenEM implementation
6. PREESM
   Runtime System
   Programming model
7. Performance Simulation Environment
   Resource Network Simulation

8. Construction
    - PREESM
    - OpenEM
    - Instrumentation
    - Simulation Model
    - Omission of IO
9. Experiments
    - Parameters and Factors
    - Measurement setups
10. Results
    - PREESM
    - OpenEM
    - Simulation Model
11. Discussion
    - Challenges
    - Discoveries
    - Related Work
    - Future Work
12. Conclusions

B Key Literature

General Background

E.A. Lee. The Problem with Threads. doi:10.1109/MC.2006.180

Bonomi & al. Fog computing and its role in the internet of things. doi:10.1145/2342509.2342513

D. Perino & al. a content router for high-speed forwarding on content names

Zhou & al. Scheduling of Parallelized Synchronous Dataflow Actors

Bachelor's theses

Teemaa, Taavi Rinnakkaisuuden mallit ohjelmointikielissä

Kiljunen, Olli Tehtävärinnakkaisuus ohjelmoinnissa

Master's theses

Hanhirova, Jussi Performance analysis of hardware accelerated scheduling

Saarinen, Joonas Parallel Processing of Vehicle Telemetric Data

Rasa, Marko Instrumentation of OpenMP task scheduling

Texas Instruments Documentation

Texas Instruments SPRUGR9H KeyStone Architecture Multicore Navigator User's Guide

Texas Instruments SPRUGW0C TMS320C66x DSP CorePac User Guide

Texas Instruments SPRUGW7A KeyStone Architecture Multicore Shared Memory Controller (MSMC)

Texas Instruments SPRUGZ2A KeyStone I Architecture Debug and Trace User Guide

Texas Instruments TMS320C6678 datasheet

Dataflow & Actors

Lee, Seshia Introduction to Embedded Systems

Preesm

Pelcat M. et. al. Preesm: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming 10.1109/EDERC.2014.6924354

OpenEM

Texas Instruments Open Event Machine Library API Spec

Texas Instruments Open Event Machine library User Guide

Texas Instruments OpenEM White Paper

F. Moerman. Open event machine: A multi-core run-time designed for performance

Stotzer & al. OpenMP on the Low-Power TI Keystone II ARM/DSP System-on-Chip

Static and Dynamic Analysis

Wilhelm et al. The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools

Banks et al. Introduction to Discrete-Event Simulation

Gustavsson et al. Timing Analysis of Parallel Software Using Abstract Execution

Fujimoto et al. Parallel Discrete-Event Simulation