H2020 FETHPC-1-2014

An Exascale Programming, Multi-objective Optimisation and Resilience Management Environment Based on Nested Recursive Parallelism
Project Number 671603

D4.2 – Early AllScale runtime system prototype

WP4: Unified runtime system for extreme scales

Version: 0.1
Author(s): Thomas Heller (FAU)
Date: 08/11/16

Due date:	PM14
Submission date:	day/month/year
Project start date:	01/10/2015
Project duration:	36 months
Deliverable lead organization	FAU
Version:	0.1
Status	Draft
Author(s):	Thomas Heller (FAU)
Reviewer(s)	Stephane Monte (NUM), Peter Zangerl (UIBK)

Dissemination level	
PU	*Public*

Acknowledgements

The work presented in this document has been conducted in the context of the EU Horizon 2020. AllScale is a 36-month project that started on October 1st, 2015 and is funded by the European Commission.

The partners in the project are UNIVERSITÄT INNSBRUCK (UBIK), FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN NÜRNBERG (FAU), THE QUEEN'S UNIVERSITY OF BELFAST (QUB), KUNGLIGA TEKNISKA HÖGSKOLAN (KTH), NUMERICAL MECHANICS APPLICATIONS INTERNATIONAL SA (NUMECA), IBM IRELAND LIMITED (IBM).

The content of this document is the result of extensive discussions within the AllScale Consortium as a whole.

More information

Public AllScale reports and other information pertaining to the project are available through the AllScale public Web site under http://www.allscale.eu.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	08/11/16	Frist draft	Thomas Heller
1.0	28/11/16	Addressing review comments	Thomas Heller
1.0	DD/MM/YY	Final version	

Table of Contents

Executive Summary

This document describes the status and structure of the early AllScale runtime system prototype deliverable (D4.2). Instructions on how to obtain, build and run the prototype will be provided as well as which parts of the interfaces as defined in D4.1 have been implemented and are part of this deliverable.

1 Introduction

Deliverable D4.2 is a source code deliverable providing the first early prototype of the AllScale runtime system. This prototype is an implementation of the API specification of the deliverable D4.1 for shared memory systems.
As such, the prototype implemented the necessary interfaces to schedule work, interface with the monitoring and resiliency component as well as providing a first prototype for a multi objective scheduler.
The remainder of this document will provide instructions on how to build the prototype as well as an overview of the implemented interfaces. So far, there were no deviations from the specification as defined in D4.1.

2 Build instructions

The first prototype is the basis of development for the AllScale runtime systems and provides a starting point of work for the other collaborators in the Project to start their experiments.

Since the AllScale runtime system is based on HPX, we have an installation of HPX as a prerequisite. Information on how to install HPX can be found here: http://stellar-group.github.io/hpx/docs/html/hpx/manual/build_system.html
Once HPX is installed, the runtime system prototype is expected to have all needed dependencies.

The source code of the prototype can be currently found in our internal git repository at this URL: http://10.10.5.114/allscale/allscale-runtime.

Once the sources have been obtained either via the Git SCM or a downloadable snapshot, the build process is configured using CMake, which interfaces with the native build tools of the developers platform (make, Visual Studio, Eclipse, etc.) and the runtime system can be built together with the unit tests as well as an accompanying example.

The code is organized as follows:

- <project root>
    - allscale – all headers needed to interface with the runtime system
    - src – The source files that contain the implementation of the runtime system
    - example – Examples showing the use of the runtime system
    - tests – Unit tests to ensure a functional runtime system
    - CMakeLists.txt – cmake build script
    - README.md – short summary and usage hints

3 Shared Memory Implementation

This deliverable is providing the first prototype of the AllScale runtime system. This first prototype has implemented the basic functionality needed to support shared memory parallelism within the AllScale programming model. The interfaces, as defined in D4.2, and this prototype implements are centered on the concept of work items. Those are the basic blocks of the AllScale runtime system to execute tasks. A scheduler prototype has been included in this deliverable demonstrating a basic, naïve scheduling policy for different work item variants. The runtime has been additionally supplemented with hooks allowing the performance monitoring and resiliency manager to gather information on the execution of work items. As Data Items as described in D4.1 are only needed for distributed memory applications, they are not part of this deliverable.

3.1 Work Item

The work item implementation was modeled after the interface definition from D4.1.. The work_item_description class has been implemented exactly as defined in D4.1., This class is used to initialize an object of class work_item. The definition of this class in this protoype is as follows:

```
struct work_item
{
    work_item();

    template <typename WorkItemDescription, typename Treeture, typename
...Ts>
    work_item(WorkItemDescription, Treeture tre, Ts&&... vs);

    work_item(work_item const& other);
    work_item(work_item && other);

    work_item &operator=(work_item const& other);
    work_item &operator=(work_item && other);

    bool valid() const

    this_work_item::id& id()

    const char* name() const

    void process();
    void split();

    template <typename Archive>
    void load(Archive & ar, unsigned);

    template <typename Archive>
    void save(Archive & ar, unsigned) const;
};
```

The main deviation from the interface defined in D4.1 is the fact that work_item itself is not derived from a common base class describing the interface of the work item concept, but rather it is implemented as a specialized type erasing value type. This allows the runtime implementation to easily store work items in containers as well as pass them around as arguments to function calls without the necessity of using pointers to a base class. For this reason, the work_item is initialized with a work_item_description which defines the specific variants of the work item, the treeture which is used to transfer the result of the work item as well as the closure of the work item, denoted as a variadic template pack.

In this prototype, the interface consists of either calling the process variant or the split variant. Which function to call is upon the scheduler to decide.

3.2 Scheduler prototype

The scheduler prototype has been implemented conforming to the interface definition in D4.1. We implemented a basic, naïve scheduling policy which directly dispatches the passed work items onto the underlying HPX thread manager. This is sufficient to demonstrate basic functionality.

3.3 Monitoring hooks

Monitoring hooks have been implemented to allow the performance analysis as well as the resiliency manager to get notified whenever a work_item gets enqueued, dequeued, started or finished execution.

4 Future Work

After the first prototype of the runtime system has been completed, the next deliverable in WP4 is to supply a first, shared memory version of the multi-objective optimizing scheduler. In regard to reach MS2, we will refine the implementation and interfaces to support WP2 and WP5 to interface with the runtime in order to reach a first fully integrated AllScale Programming Environment. We are planning on releasing a first public release of the AllScale runtime system for MS2. In addition, we are preparing the already existing implementation to be used in a distributed memory application.