A Pattern-supported Parallelization Approach

Ralf Jahr, Mike Gerdes, Theo Ungerer
University of Augsburg, Germany

The 2013 International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM 2013)
Outline

- Motivation
- Parallel Design Patterns
- Pattern-supported Parallelization Approach
  - Two phases
  - Activity and Pattern Diagram
  - Pattern Catalogue
- Case study: Unmanned Aerial Vehicle
- Summary
Motivation

- Multicore and manycore CPUs in embedded systems

- Goals:
  - Faster execution of a workload
  - Concurrent execution of multiple tasks
  - Shorter reaction times
  - Energy savings because of lower clock frequency

→ Need for parallel applications

- But, especially for embedded systems:
  - Much legacy code
  - Limited development resources
  - Complicated testing and debugging
Parallel Design Patterns

- **Design Patterns**
  - Idea initially in architecture
  - Recurring problems → best practice solutions
  - Transfer to software engineering
  - Mainly object oriented design, see “Gang of four”
  - Standardized description: Pattern Catalogue

- **Parallel design patterns**
  - Extended concept: design patterns providing parallelism
  - Tradeoff: flexibility in design vs. development effort
Pattern-supported Parallelization Approach

- **Starting point:**
  - Sequential program (“legacy code”)
  - Pattern Catalogue with parallel design patterns

- **Phase 1: Targeting Maximum Parallelism**
  - Create model to reveal parallelism
  - Model consisting of sequential parts and parallel design patterns
  - Platform independent

- **Phase 2: Targeting Optimal Parallelism**
  - Agglomeration of nodes, definition of parameters
  - Creation of threads and mapping onto target architecture
  - Platform dependent
The Pattern Catalogue:
- Basis for parallelization
- Contains all allowed parallel design patterns
- Description according to meta-pattern
- Description is textual, no reference implementations
- Implementation examples are optional
- Grows over time

Example: “Our Pattern Language”
- [http://parlab.eecs.berkeley.edu/wiki/patterns/patterns](http://parlab.eecs.berkeley.edu/wiki/patterns/patterns)
- Organized in multiple layers
Extension of UML2 Activity Diagram:
- *Parallel design pattern* is new node type similar to activity
- Activities: either sequential or encapsulate APD
- Parallel design patterns: Multiple activities in parallel

Patterns are only way to introduce parallelism

Advantages over inventing a new notation:
- Well known, easy to understand, tools exist
- Support for dependencies, branches, and nesting
Activity and Pattern Diagram

Parallel Pattern

Start node

Initialize Weightmap Data Parallel

Obstacle Map

Sequential code

Calculate Sectors

Weightmap

Sectors

Propagation Weights Pipeline

Weightmap

Sectors

End node

Reading Access

Writing Access
Goal: Reveal sufficient parallelism for any platform as Activity and Pattern Diagram (APD)

Start with single activity, repeatedly apply two operations:

a) **Replacement**: apply parallel design pattern  
b) **Splitting**: decompose into multiple activities
Transition from maximum to optimal parallelism by agglomeration

Similar to optimization problem:
- Global Objective: reduce execution time, energy consumption, ...
- Execution time influenced by e.g. communication/computation ratio, cost for synchronization, etc.
- Side conditions: number of available cores/threads; dependencies (control, data, timing), etc.
Agglomeration is...

- **Replacing a parallel design pattern** by an activity, e.g., replacing pipeline by activity → Reduction of parallelism
- **Joining elements** of parallel design pattern, e.g., multiple pipeline stages to single one → Reduction of overhead
- **Defining parameters**, e.g., concurrent workers for data parallelism → Tailoring design patterns to target platform
Pattern-supported Parallelization Approach: Phase 2

- **Mapping**
  - Find optimal mapping between code (APD) and threads/tasks and cores/clusters
  - Trade-off between optimal use of resources vs. parallelism
  - Not in focus of parallelization, different research area

- **Objectives for parallelization process**
  - Speedup/rough approximation of speedup
  - Resource usage
  - Energy consumption
  - Implementation effort (e.g. number of patterns)

- **If necessary: iterative application of process!**
A Pattern-supported Parallelization Approach

- Manual process with clear methodology
- Fast modelling of parallelism with Activity and Pattern Diagram; derived from UML2
- Pattern Catalogue
  - Easier implementation of parallel program
  - Higher Documentation Quality
- Algorithmic skeletons for reduced implementation effort

Similar to Mattson

Pattern Catalogue

Sequential Program / Problem description

Model of Program with Parallel Design Patterns

Source Code

Similar to Foster

Maximum parallelism by decomposition

Optimal parallelism by agglomeration
Example & Work in Progress: Unmanned Aerial Vehicle (UAV)

http://www.oberwelzdesign.com/en/project/quadcopter01
The Software

- Autonomous flight over terrain
  - Obstacle detection
  - Automatic path planning (Laplace operator)

- Assumptions:
  - Sequential software exists

- Overview of the software:
  - Initialize system
  - Loop until goal is reached:
    - Determine position
    - Mark obstacles
    - Plan path
    - Set course
Parallelization

- **Phase 1**
  - Goal: Expose parallelism
  - Finished, see paper
  - Six instances of parallel design patterns

- **Phase 2**
  - Goal: Tailor parallelism to target platform
  - But: work in progress, no target platform yet defined
  - Approximated speedup based on profiling: 7.8
    - Enough parallelism for 2 to 6 cores
    - Further work necessary for 8+ cores
Pattern-supported parallelization approach

- Two phases:
  - Reveal parallelism: architecture independent
  - Agglomerate and map: architecture dependent
- Only parallel design patterns to introduce parallelism
- Parallel design patterns are described in Pattern Catalogue
- Supporting structure: Activity and Pattern Diagram, similar to UML2 Activity Diagram
- Limited effort for parallelization and implementation of parallel program

Future work:
- Tool support for parallelization, especially Phase 2
- Extend parallelization process for hard real-time systems
- More case studies, different platforms → gain knowledge