Component-oriented Approaches for Software Development and Execution in the Extreme-scale Computing Era

Vladimir Getov
University of Westminster, London
V.S.Getov@westminster.ac.uk

HPC-10 Workshop, Cetraro, Italy
22 June 2010
Overview

- Background – software crisis
- New challenges in the extreme-scale computing era
- Example component framework overview - GCM
- Problem-to-solution development pipeline
- Component-oriented integrated environment
- Case studies: Developing component-based codes, Componentising existing applications, Wrapping legacy codes
- Future research topics and conclusions
Background – Software Crisis

- Software vs Hardware – short-term and long-term issues
- Software crisis – legacy code and software development cycle problems
- Particularly difficult in fast developing and changing complex computer systems
- Need much shorter development cycle in order to be able to catch up with the pace of development of underlying hardware
- Need methodology for quickly adapting/porting legacy code
New Challenges – Extreme-scale Computing

Where ‘just more of the same’ does not work

We need to improve (in no particular order):
- degree of parallelism,
- performance,
- cost,
- footprint,
- power,
- reliability,
- programmability,
- productivity, etc.

The complexity is qualitatively harder and multidimensional – addressing this unprecedented conundrum of challenges is called ‘Extreme-scale Computing’.
Component-based Platform

- The new challenges require major breakthroughs in hardware and software – e.g. the introduction of multi/many-core architectures.
- The higher level of complexity involves a wider range of requirements and resources → Dynamic intelligent properties and flexibility → Component-based design methodology
- To develop the design methodology of a generic component-based platform for both applications and system frameworks to have a single, seamless, “invisible” system image.
Component-based Platform Architecture

Component Model

Integrated Component-based Framework

System Software

OS Kernel

Multi/Many Cores (homogeneous or heterogeneous)

Applications

Tuning Interface

Non-functional properties

Platform
Example Component Framework Overview

Starting point: Fractal component model. The main technical features of the component framework are:

- Support for primitive and composite distributed components and hierarchical composition.
- XML based architecture description language (ADL).
- Collective interfaces to comply with specific multi-way communication requirements.
- A comprehensive runtime API.
- Support for non-functional aspects such as component control, skeletons, and autonomy.
- Advanced component scheduling/deployment via the notion of virtual nodes and deployment descriptors.
GCM

- A new advanced Grid Component Model (GCM) providing high level of abstraction and specifically designed for large scale dynamic Grid infrastructures.
- Specified within the CoreGRID European project.

GridCOMP

- EU project: Grid programming with COMPonents.
- INRIA, ERCIM, Univ Westminster, Univ Pisa, CNR, IBM ZRL, Atos Origin, Grid Systems, Tsingua Univ, Univ Chile, Univ Melbourne
- Design and implementation of a Grid component framework based on GCM.
- Includes the development of a Grid IDE and several use case applications.
- Middleware reference platform implementation.
GridCOMP Component Framework Overview

Hierarchical composition: all three components can be distributed
GridCOMP Component Framework Overview

Collective interfaces: The framework takes care of parallel invocations, data distribution, and synchronization.
Component-Centric Problem-to-Solution Pipeline

- Main issues: composition and dynamic properties – deployment, monitoring and steering
- Component-based Grid platform design methodology

Grid Integrated Development Environment

Applications (Algorithms) → Programming Model - GCM → Composition → Scheduling & Deployment → Monitor & Steer → Obtaining the Solution

Metadata Description incl. ADL, etc.
Strategy: Eclipse Framework for GIDE

- Simplify complexity through graphical composition/tools
- But, allowing ONLY graphical composition can be inflexible and inefficient
- Support for 3 levels of Development
  - Graphical Composition
  - Based on GCM and using ADL
  - Source code level
- Seamless integration with Eclipse
  - Widely supported with many potential plug-ins
Grid IDE Architecture - Core Block Diagram

**Applications (Use Cases)**

1. Development IDE
2. Data Centre IDE

**Eclipse Framework**

- Composition
- Scheduling and Deployment
- Monitoring
- Steering

**Grid IDE Toolset**

1. 1.2.1-ADL Parser/Verifier
2. 1.2.2-ADL Renderer
3. 1.2.3-Code/ADL Generator
4. 1.3.1-Test Tool
5. 1.3.2-Debug Tool
6. 1.3.3-Finalization Deployment Tool
7. 2.1.1-Component Monitor
8. 2.1.2-Node Resource Monitor
9. 2.2.1-Start/Stop
10. 2.2.2-Install/Remove

**Uses**

API

**Requirements**

Uses
Case Study: Biometric Identification System (BIS)

- Identify people solely on their biometric information (1:N match)
- Use fingerprint biometrics
- Consider multiple fingers per person to work reliably on large user population
- Use distributed matching to achieve real-time performance
- Based on business process (workflow) engine for adaptability
Case Study: Biometric Identification System (BIS)

Grid component architecture, bindings, and deployment

1. Biometric matching component `ComplIDMatcher` is deployed on each node
2. DB of known identities is distributed across the nodes
3. Identification requests are broadcasted via multicast interface I2
4. Each node searches its part of the DB
Development – IBM ZRL BIS Use Case

Grid programming with components: an advanced COMPonent platform for an effective invisible grid
Componentizing existing applications

• Methodology: – manual with on-going activity on identifying parts for automatic support and tools

• Sample Code: Jem3D – 3-dimensional Maxwel’s equations solver for aircraft wing design

• Experimental Results: Componentising a Scientific Application – Jem3D
Componentising a Scientific Application – Jem3D

- numerical solver for the 3D Maxwell’s equations modelling the time domain propagation of electromagnetic waves
- follows typical “geometric decomposition” parallelisation
Jem3D Architecture
Wrapping Legacy Software

- Methodology: (semi-)automatic or manual

- Sample Code: GENIE Application (Environmental Modelling)

- Motivation: Enable legacy applications to evolve as a part of the scalable problem solving environments within modern Grid systems.

- Framework: Componentising existing applications along with domain-specific metadata so that issues arising thereof can be addressed using this metadata.

- Experimental Results: Domain-Specific Metadata for Model Validation and Performance Optimisation
Domain-Specific Metadata for Model Validation–Legacy Applications

GENIE is an interactive, legacy code for Earth system modelling. Our hypothesis is that componentising the application and using domain-specific metadata will help transforming it into a scalable yet efficient software system.
Summary

- Productivity based on higher level of abstraction
  - Enables the use of new modern technologies such as graphical composition
  - Source code generation
  - Repositories for components re-use

- The use of behavioural skeletons reduces further the development effort

- Optional features
  - Dynamic composition validation using OCL
  - Static composition validation while generating final ADL file(s)
  - Domain-specific validation
  - Dynamic verification

- GIDE prototype - an Eclipse plug-in using GMF.
Conclusions and Future Research

- Created the core framework using Eclipse

- Robust and friendly

- The full prototype of the GIDE toolset has been completed

- The hierarchical component composition results are promising – higher development productivity and easier software components re-use

- Develop the design and development methodology for building modern component-based software
Acknowledgements

- Alessandro Basso
- Alexander Bolotov
- Artie Basukoski
- Stavros Isaiadis
- Matthieu Morel
- Nikos Parlavantzas
- JeyanThiyagalingam
- Thomas Weigold
- And other colleagues from the CoreGRID and GridCOMP projects