HDF File-based interoperability for multiscale modelling

Fabio Sacconi, Stefano Bellocchio

Tiberlab Srl, Rome, Italy
OUTLINE

- Deepen Project Overview

- Deepen Multiscale platform: Unicore implementation

- HDF choice as Deepen Common Data Format
  - HDF features
  - Implementation
  - Examples of use
From atom-to-Device Explicit simulation Environment for Photonics and Electronics Nanostructures

**Objective:**
Developing of an integrated open source multiscale simulation platform for predictive design of novel materials and nanostructures for CMOS and LED applications

*FP7-NMP Project DEEPEN, No. FP7-604416.*
New device simulators require

- An **atomic-scale description** of selected **critical regions** of advanced channel devices or nitride-based LEDs to capture details otherwise inaccessible.
- To resolve the considerable **uncertainty in many critical parameters** required for device optimisation.

**DEEPEN Challenges and Goals**

- DEEPEN addresses these challenges, coherently **linking** and **coupling** several state-of-the-art existing and newly developed tools for simulation of **electronic properties** and **quantum transport**, integrated within a multiscale **framework** spanning from **first-principles to macroscopic models**.
Deepen Platform

- Based on the **UNICORE** standard (http://www.unicore.eu) for access to modern heterogeneous computer networks
- **Control of Simulation Flow** to be implemented using UNICORE clients for workflow generation
- **HDF**-based Open source Common Data Format (CDF) implemented to allow interoperability between different simulation tools (DFT, Tight-binding, NEGF, Drift-Diffusion)
- **CDF Translators** and **UNICORE APIs** allow to exchange data while keeping application tools separated from the platform.
- **Metadata**: e-CUDS, e-CUBA in HDF5 format
Deepen Platform

GRIDBEAN API

UNICORE CLIENT

TIBERCAD
OMEN

WORKFLOW MANAGER

WANNIER90

UNICORE SERVER

QUANTUM ESPRESSO

CDF Translators

OMEN CDF
TIBERCAD CDF
WANNIER90 CDF
QE CDF

CDF API

TSI

SGE

CPU

LIBRARY
UNICORE – Schema Components

- **Client-Server Grid Middleware**
- Provides a seamless, secure, and intuitive access to different distributed computing and data resources
- *Open source* under BSD license.
- *Standards-based*, conforming to the latest standards from the Open Grid Forum (OGF), W3C, OASIS, and IETF
- *Open and extensible* realized with a modern Service-Oriented Architecture (SOA)
UNICORE – Schema Components

Generation of Workflows of the jobs to be submitted
Through different clients:
- Command-line (UCC)
- Graphic interface Eclipse-based (URC)
- APIs (HiLA)

Client side
UNICORE – Schema Components

Server side

The **UNICORE/X server** is the heart of a UNICORE site. Via web services it provides access to *storage resources, file transfer services, job submission and management services.*

- The **Gateway** component acts as the entry point to a UNICORE site and performs the authentication of all incoming requests. May serve several resources/target systems behind it.
- The **workflow engine** deals with high-level workflow execution.
- The **service orchestrator** layer is responsible for executing the individual tasks in a workflow, handling job execution and monitoring on the Grid.
- In the **TSI** component the abstracted commands from the Grid are translated to system-specific commands. In a typical grid environment, a resource management software (**RMS**) is implemented.
- The **USpace** is UNICORE's job directory. A separate directory exists for every job, where the XNJS and TSI stores all input data and where stdout and stderr are written to.
- **External Storage** is a space accessible from Client and from TSI.
Unicore APIs (GridBeans)

- **UNICORE APIs** allow to exchange data while keeping application tools separated from the platform.

- A UNICORE GridBean is an object responsible for:
  - generating job description for grid applications
  - providing graphical user interface for input data
  - providing graphical user interface for output data

- UNICORE GridBeans are divided into several modules:
  - a job description generation module
  - one or more user interface modules
GridBean - Overview

App Plugin

- Input Panels
- Output Panels

App Model

- Input Parameters
- Resource Requirements
- Output Parameters

Generic Panels

- Variable Panel
- Data Import Panel
- Resource Panel
- Data Export Panel

Application specific

Job description

Generic
A Common Data Format (CDF) based on the Hierarchical Data Format (HDF5) has been implemented in Deepen to allow interoperability between different simulation tools.

HDF software first developed in 1987 at the National Center for Supercomputing Applications (NCSA) at the University of Illinois.
HDF Features

- A versatile **data model** that can represent very complex data objects and a wide variety of metadata.

- A software library that runs on a range of computational platforms, from laptops to massively parallel systems, and implements a high-level API with C, C++, Fortran 90, and Java interfaces.

- A **free and open source** (BSD license) general purpose platform for storing, managing, archiving, and exchanging data: thanks to **Open Format** HDF5 is widely supported in many softwares, including open source programming languages like R and Python, commercial programming tools like Matlab and open GIS tools like QGIS.
**HDF Features**

- **Self-Describing:** Similar to XML, the datasets with an HDF5 file are self-describing allowing an application to interpret the structure and contents of a file with no outside information.

- Supports **Large, Complex Data:** HDF5 is a compressed format that is designed to support large, heterogeneous, and complex datasets. HDF5 format and software include features to store and access large datasets (even 1 TB in a file).
  - Efficient storage options, such as chunking, compression.
  - **Fast random access:** optimized access to very large arrays, even when datasets are compressed.
  - supports parallel I/O
HDF Features

- **Support Heterogeneous Data**: Different types of datasets can be contained within one HDF5 file: tables, images, streams of data from instruments, and structured grids all in the same file. New objects can be easily added to existing HDF5 objects.

- **Direct access** to parts of the file without first parsing the entire contents: “Data slicing”, or extracting portions of the dataset as needed for analysis, means large files don’t need to be completely read into the computer’s memory or RAM.

- **Hierarchical data ordering**: Supports complex data relationships and dependencies through the objects **Groups** (container structures) and **Datasets** (multidimensional arrays).
HDF: hierarchical structure

- **Group** - a collection of objects (including groups), analogous to a directory
- Group membership is implemented via *link* objects
- **Dataset** - a multidimensional array of data
- **Dataspace** - dimensions of a multidimensional array
- **Datatype** - a description of a specific class of data element
HDF implementation in Deepen

- HDF structured like a unix filesystem
- Hierarchical representation of data, similar to a path

Example OMEN Diagram
HDF hierarchical structure allows database implementation using a combination of groups and datasets.
HDF hierarchical structure allows database implementation using a combination of groups and datasets.

Example DB Diagram:

- **/** root
- **Material**
  - **Band structure**
    - **Alloy conc.**
      - **CB1**
        - **Valley 1**
          - **Band energy**
            - description
            - units
          - **Effective mass**
            - description
            - units
          - **Def. potentials**
            - description
            - units
Ab-initio transport: inter-changeable program flows

DFT Tool (VASP)

Ab-initio electronic structure calculation

Calculation of current vs voltage in nanostructures

VASP

Ab-initio integral transport: inter-changeable program flows

Wannier90

Conversion from plane-wave to MLWF basis

OpenMX

Welcome to OpenMX

open source package for material informatics

Ab initio electronic Structure calculation

QT Tool (OMEN)

Quantum Transport

SiO

TIMeS

Quantum Transport

Tyndall

ETH Zürich

ICME 2016

Barcelona, 12-15 April 2016
Test Workflow: DFT/Tight-Binding linking

Quantum Espresso

wannier90

OSI

Tight-Binding

Input needed:
- DFT Hamiltonian
- Relaxed atom positions

OMEN

Output provided:
- Unit cell data
- Initial atom positions

Density-Functional Theory

Plane-Wave Basis

Localized Wannier Basis

Test Workflow:
DFT/Tight-Binding linking
Test Workflow: DFT/Tight-Binding linking
Workflow DFT/Tight-Binding linking in Deepen
HDF for model linking

Example OPENMX Diagram

<table>
<thead>
<tr>
<th>Table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.4341351...</td>
</tr>
<tr>
<td>1</td>
<td>-6.6562974...</td>
</tr>
<tr>
<td>2</td>
<td>4.5695353...</td>
</tr>
<tr>
<td>3</td>
<td>4.8331513...</td>
</tr>
<tr>
<td>4</td>
<td>0.0453266...</td>
</tr>
<tr>
<td>5</td>
<td>-0.0785091...</td>
</tr>
<tr>
<td>6</td>
<td>-7.1024028...</td>
</tr>
<tr>
<td>7</td>
<td>2.7286385...</td>
</tr>
<tr>
<td>8</td>
<td>-1.6493942...</td>
</tr>
<tr>
<td>9</td>
<td>-0.1561578...</td>
</tr>
<tr>
<td>10</td>
<td>0.1920289...</td>
</tr>
<tr>
<td>11</td>
<td>-2.2529795...</td>
</tr>
<tr>
<td>12</td>
<td>1.8144068...</td>
</tr>
<tr>
<td>13</td>
<td>0.0763550...</td>
</tr>
<tr>
<td>14</td>
<td>0.1322559...</td>
</tr>
<tr>
<td>15</td>
<td>2.0213786...</td>
</tr>
<tr>
<td>16</td>
<td>-2.7641957...</td>
</tr>
<tr>
<td>17</td>
<td>-4.9302217...</td>
</tr>
<tr>
<td>18</td>
<td>-0.0029104...</td>
</tr>
<tr>
<td>19</td>
<td>0.0007222...</td>
</tr>
<tr>
<td>20</td>
<td>-3.1068171...</td>
</tr>
<tr>
<td>21</td>
<td>1.5469833...</td>
</tr>
<tr>
<td>22</td>
<td>0.0009103...</td>
</tr>
<tr>
<td>23</td>
<td>-0.0102379...</td>
</tr>
<tr>
<td>24</td>
<td>1.1030901...</td>
</tr>
<tr>
<td>25</td>
<td>-5.4772887...</td>
</tr>
<tr>
<td>26</td>
<td>8.1181275...</td>
</tr>
<tr>
<td>27</td>
<td>0.00019105...</td>
</tr>
<tr>
<td>28</td>
<td>0.0003653...</td>
</tr>
<tr>
<td>29</td>
<td>0.00021056...</td>
</tr>
<tr>
<td>30</td>
<td>0.00091298...</td>
</tr>
<tr>
<td>31</td>
<td>0.00191056...</td>
</tr>
<tr>
<td>32</td>
<td>0.00091298...</td>
</tr>
</tbody>
</table>
Thank you!