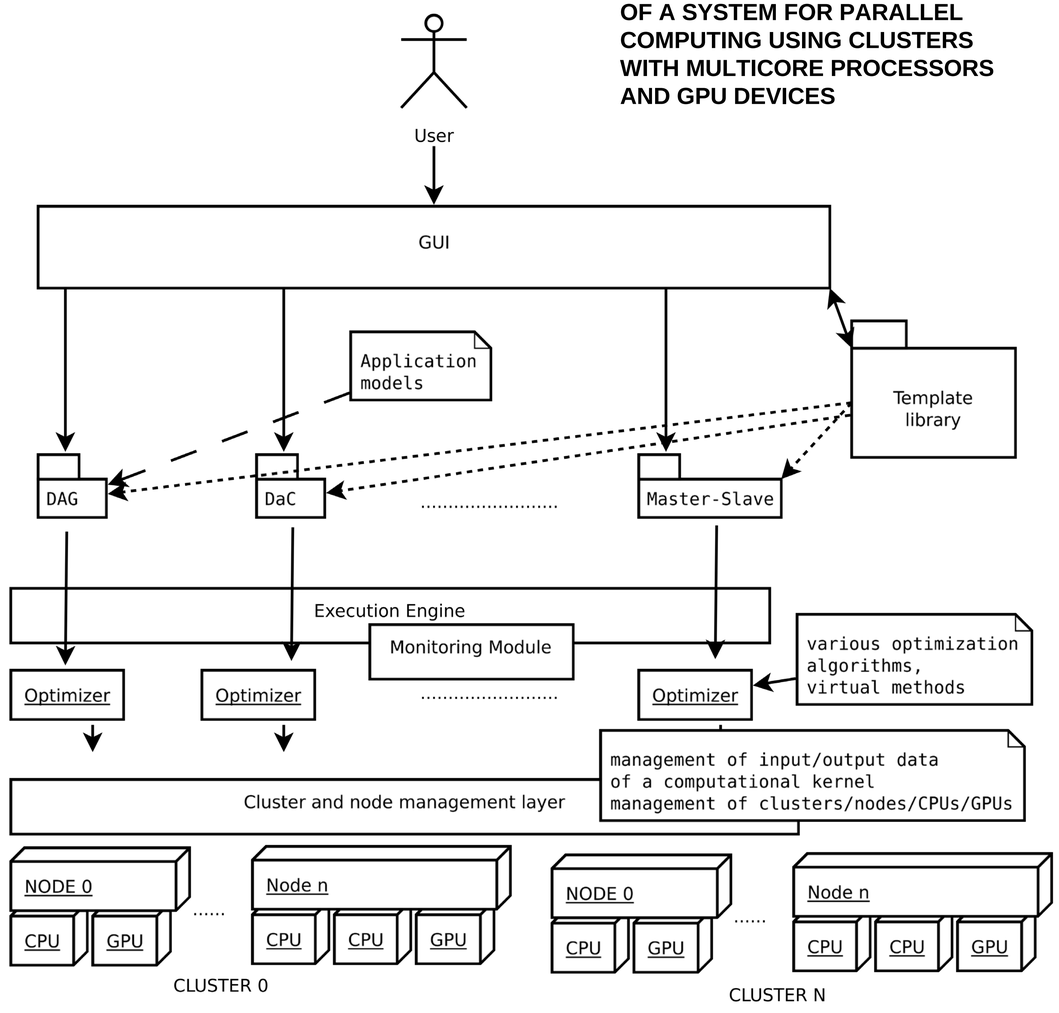
CONTACT NAME
Pawel Czarnul: pczarnul@eti.pg.gda.pl

Multi-level Parallelization of Computations using Clusters with GPUs

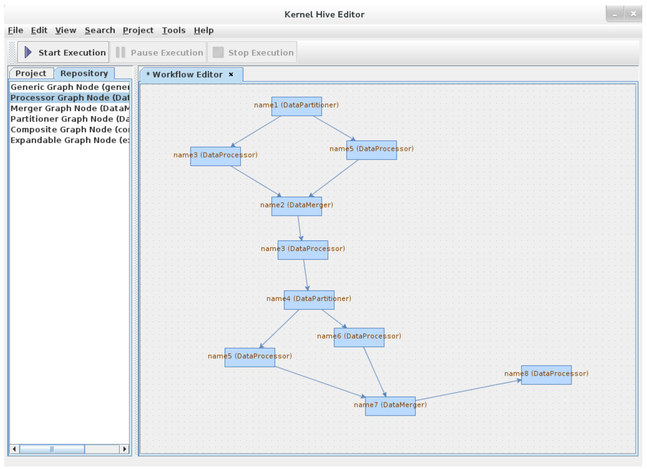
Paweł Czarnul, Rafał Lewandowski, Paweł Rościszewski, Marcel Schally-Kacprzak
Gdańsk University of Technology, Poland, http://pczarnul.eti.pg.gda.pl pczarnul@eti.pg.gda.pl

ABSTRACT:
The poster presents an approach for multi-level parallelization of computations among clusters that are equipped with both multicore CPUs and modern GPUs. A multi-level and modular scheme is presented that allows the programmer to define an application as a workflow using ready-to-use elements and constructs. The programmer just needs to code partitioners, mergers, computational kernels and provide input data. The application is then parallelized automatically among available CPUs and GPUs, possibly on various clusters. The results for a compute intensive application show promising scalability, depicted on the poster.

A MULTI-LEVEL ARCHITECTURE OF A SYSTEM FOR PARALLEL COMPUTING USING CLUSTERS WITH MULTICORE PROCESSORS AND GPU DEVICES



EDITOR FOR AN APPLICATION SPANNING CLUSTERS WITH GPUS



The application is defined as a workflow modeled as a directed acyclic graph (DAG).

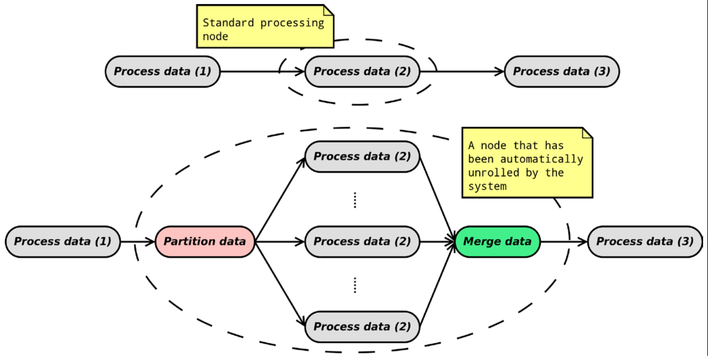
Two types of source codes are assigned to workflow nodes:
1. predefined constructs such as:
- data partitioners
- data mergers
2. computational kernels defined in OpenCL for portability – executed on GPUs and multicore CPUs

The kernels and kernel templates can be stored in a repository and filled in when designing a new application.
The user supports an URL where input data is stored.
The application model is completely independent from the underlying infrastructure.

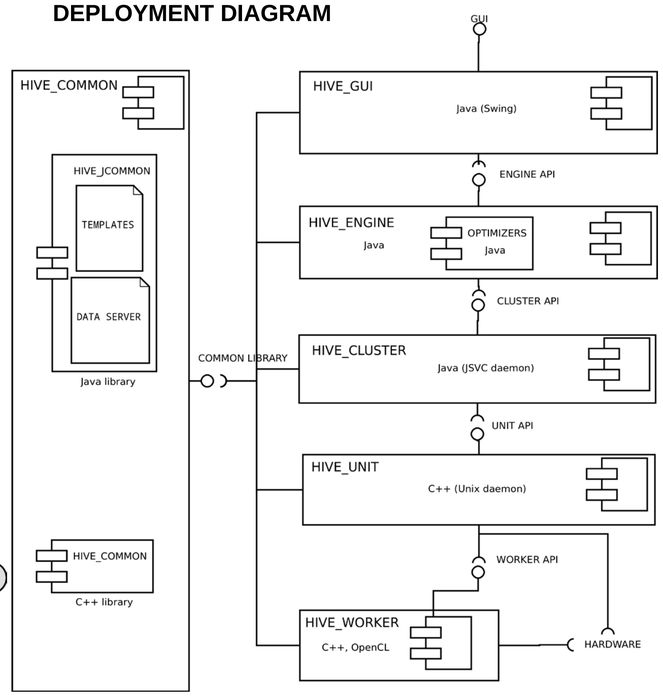
SAMPLE WORKFLOW APPLICATION

The sample parallel application finds a password that gives a predefined MD5 hash using brute force. The data is fetched from the designated data server, partitioned into a certain number of data chunks (dependent on the number of clusters, CPUs and GPUs), processed and returned.

The kernel function has the following form:
__kernel void processData(__global unsigned char *input, unsigned int dataSize, __global unsigned char *output, unsigned int outputSize) { … }



DEPLOYMENT DIAGRAM



ARCHITECTURE

The architecture consists of the following layers:

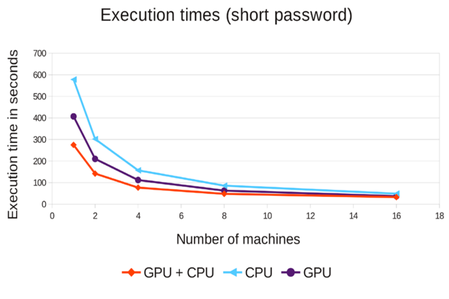
1. application editor and template repository
2. execution engine (spanning clusters)
3. cluster and node management layer including management of executables on GPUs and CPUs

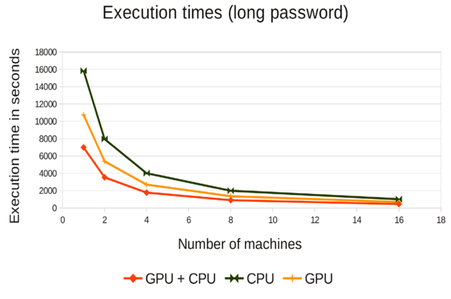
TESTBED ENVIRONMENT

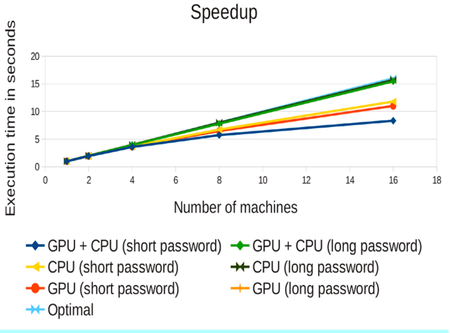
The testbed environment consists of:

16 workstations, each with:
- Intel i7-2600K CPUs each with 8 logical cores for a total of 128 cores
- NVIDIA GeForce 450 GPU each with 192 CUDA cores for a total of 3072 cores
- SUSE Linux version 3.1.10-1.16
- 8 GBs RAM for a total of 128 GBs RAM
- Gigabit Ethernet

EXECUTION TIMES AND SPEED-UPS IN A REAL ENVIRONMENT







For information about the project go to: http://kask.eti.pg.gda.pl/projekty/kernelhive/
Contact information: Paweł Czarnul
Email: pczarnul@eti.pg.gda.pl, WWW: http://pczarnul.eti.pg.gda.pl, phone: +48 58 3471288
Department of Computer Architecture, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland