Ad-hoc Architectures for modern DBMS: a HW/SW Co-Design Approach

L. Pomante, P. Di Felice

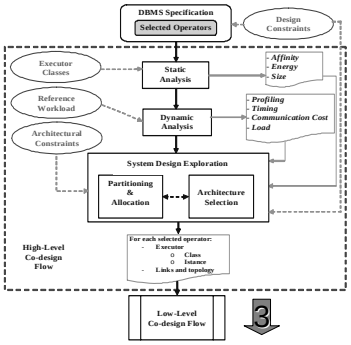
DEWS/DIEI
University of L'Aquila, Italy

Overview

- DB technology is fundamental in several application domains
  - *GIS*, *Bioinformatic Information Systems*, etc…
- Such domains involve data-types very different from traditional ones
  - They upset traditional guidelines focused on the I/O cost
    - The consequence is the need of support for DBMS to manage such data
      - Several attempts have been made to provide some "accelerators"
    - However, existing works lack of generality
      - They select specific DB operators according only to the designer experience
- To fill such a gap, this work follows an *hw/sw co-design approach* to
  - Support the designer in the selection of those DB operators that could benefit from an ad-hoc executor (GPP, DSP, GPU, FPGA, etc…)
  - Identify also the system architecture (number and classes of executors, and interconnection topology) appropriate to reach the desired goals

Follow the navigation tips!

The Proposed Methodology



The Proposed Methodology

- The entry point is the *all-sw implementation* of a DBMS
  - It is considered as a system-level executable specification
    - The source code of the implementation could be not completely available
    - The implementation will be not fully synthesizable/mappable
- The **Static Analysis** provides metrics to define the architecture
  - *Affinity: quantification of the matching between operators and executors*
  - *Energy: estimations of the energy required for the execution of a given operator*
  - *Size: estimations of the "space" required to implement a given operator*
- Once defined a **Reference Workload…**
  - The **Dynamic Analysis** provides other information about
    - ***Profiling:*** *number of executions of the statements composing an operator*
    - ***Timing:*** *estimations of the time required by HW/SW for the execution of an operator*
    - ***Communication Cost:*** *amount of data exchanged between different components*
    - ***Load Estimation:*** the load that a given operator imposes to a GPP
- ***System Design Exploration***
  - The output of this step is the allocation of the selected DBMS components on the proposed architectural elements

2

Entry point

- To give the flavour of how the methodology could work, it is presented a meta-example on ad-hoc acceleration of DBMS operators
  - Let us considering an ORDBMS with *geometric data types* and some related operators
    - `Op1:IsEmpty():Integer`
    - `Op2:Intersects(AG:Geometry):Integer`
    - `Op3:Buffer(dist:Double):Geometry`

A Meta-Example

4

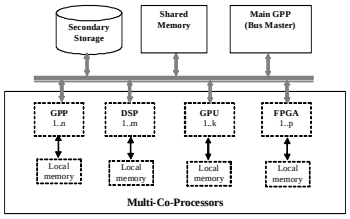
6

5

8

Reference platform

- System Design Exploration
  - Target Architecture: main PC with possible accelerators



7

Meta-Example: Step 1

- Static Analysis
  - Considering only two executors classes (GPP and FPGA), the *Static Analysis* applied to the three methods provides the following information

	Affinity					Size
	GPP	*DSP*	*FPGA*	*GPU*	*SW (KB)*	*HW (FPGA Cells)*
$Op1_m$	0.61	0.33	0.24	0.23	258	128
$Op2_m$	0.43	0.21	0.48	0.42	365	125
$Op3_m$	0.45	0.15	0.87	0.21	912	233

- Reference Workload
  - Such a task could be performed with two different goals
    - to optimize the implementation with respect to a target DB schema
    - to optimize the implementation generally
      - In this example, we consider a single schema designed for a specific application and a set (e.g. 100) of different queries representative of the DB utilization

10

Specification & Requirements

- Let each operator be implemented in SW by means of a C++ method
  - *Op1m*, *Op2m*, *Op3m*
    - The source code is supposed to be available
- Let the design constraint be oriented to accelerate the selected operators in order to obtain an execution time of 30% less than the all-sw single-GPP architecture

9

11

13

Meta-Example: Step 2

- Dynamic Analysis

	Profiling *(#executions/query)*	**Timing** *(sec/#executions)*	**Load Estimation** *(30% exe time saving)*
$Op1_m$	2.4	0.9	0.20
$Op2_m$	0.8	1.1	0.29
$Op3_m$	3.5	1.8	0.38

(#bytes/query)	***DBMS-Core***	$Op1_m$	$Op2_m$	$Op3_m$
DBMS-Core	0	22	81	76
$Op1_m$	22	0	0	0
$Op2_m$	81	0	0	0
$Op3_m$	76	0	0	0

Communication

12

Meta-Example: Step 3

- System Design Exploration
  - The final step of the proposed flow is the *System Design Exploration*
    - Such a step takes as input all the information provided by the previous steps plus some *Architectural Constraints* by the designer
  - Defining a cost function with equal weights for size, affinity, communication cost and load, such a step, based on a genetic algorithm provides the following result

14

Meta-Example: Result

- Ad-hoc Architecture
  - Definition, Partitioning & Mapping