



**SEVENTH FRAMEWORK PROGRAMME
THEME ICT-2009- Call 4- 3.6
Computing Systems**

Project acronym:

2PARMA

Project full title:

**PARallel PARadigms and Run-time MAnagement
techniques for Many-core Architectures**



**Deliverable D3.3.1:
Initial prototype of the design space
exploration methodologies
for run-time support**

Version [1]

Delivery due date: M18 (June 2011)
Actual submission date: July 25th, 2011
Lead beneficiary: POLIMI

Dissemination Level of Deliverable		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	
Nature of Deliverable		
R	Report	
P	Prototype	X
D	Demonstrator	
O	Other	



Author(s):	Vittorio Zaccaria (POLIMI), Alex Bartzas (ICCS), Chantal Ykman Couvreur (IMEC)		
Reviewer(s):	Alex Bartzas (ICCS), Bart Vanthournout (Synopsys)		
WP/Task No:	3.3	Number of pages:	13
Identifier:	D3.3.1	Dissemination level:	PU
Issue Date:	July 25 th , 2011		

Keywords: Design Space Exploration, Run-time

Abstract: This document (issued at M18) represents the accompanying report of the prototype design space exploration framework to support run-time management publicly released on the 2PARMA website. The deliverable represents the results of the activity carried on from M7 to M18 in Task 3.3 (*Design space exploration of configurable parallel computing platforms for supporting run-time system management*) under the leadership of POLIMI, with contributions from POLIMI, IMEC and ICCS. The prototype showcases an approach to fast, automatic, multi-objective exploration of the software design space to support run-time management of many-core computing platforms. The methodology targets robust design space exploration of dynamic parameters (derived from analysis done in WP1) by considering robustness with respect to datasets of the Stereo Matching application as introduced in D1.2; moreover it identifies hints/guidelines for dynamic resource management (as needed by WP4).

The prototype will be extended to encompass the OpenCL toolchain (and run-time environment) as well as to run on the P2012 platform.

This report contains:

- A summary of the structure and functionality of the prototype.
- How to download and install the prototype.
- How to run the prototype.

Approved by Project Technical Manager



Approved by the Project Coordinator:



Date: July 25th, 2011

Project Technical Manager: Prof. Dr. William FORNACIARI – Politecnico di Milano
e-mail: fornacia@elet.polimi.it - **Phone:** +39-02-2399-3504- **Fax:** +39-02-2399-3411

Project Coordinator: Prof. Dr. Cristina SILVANO – Politecnico di Milano
e-mail: silvano@elet.polimi.it - **Phone:** +39-02-2399-3692- **Fax:** +39-02-2399-3411

Table of contents

I. Overview.....	4
II. Prototype Architecture.....	5
II.1 Prototype flow.....	6
II.1 Robustness with respect to workload.....	7
III. Installation Instructions	10
IV. How to run the prototype	11
V. References	13

I. Overview

Nowadays multiprocessor architectures are designed by means of a platform-based design approach. In this context, a configurable platform can be seen as a template with a set of static and dynamic parameters. Static parameters are fixed at design time (e.g. overall cache size, number of processors) while dynamic parameters are parameters that can be changed at run-time. The degree of parallelization of the application is just an example of such dynamic parameters.

Design time tuning (also called static design space exploration) should provide robustness with respect to a wide set of workload scenarios. Each scenario can impact the run-time management of the resources given the target quality of service (QoS) constraints, introducing variability that should be taken into account during static design space exploration.

Nevertheless, for a fixed configuration of the static parameters, the static design space exploration should provide a set of indications/suggestions for configuring dynamic parameters at run-time.

This prototype provides an initial approach to fast, automatic, multi-objective exploration of the software design space to support run-time management of many-core computing platforms. In particular, it prototypes a methodology for robust design space exploration of dynamic parameters (derived from the analysis done in WP1 and WP4) by considering robustness with respect to data-sets (which can be dynamically changing when considering *dynamic workloads*), while identifying hints/guidelines for dynamic resource management (as needed by WP4).

Note: the prototype contains a special release of MOST that is able to execute only ‘signed’ scripts. The prototype scripts have all been signed by POLIMI.

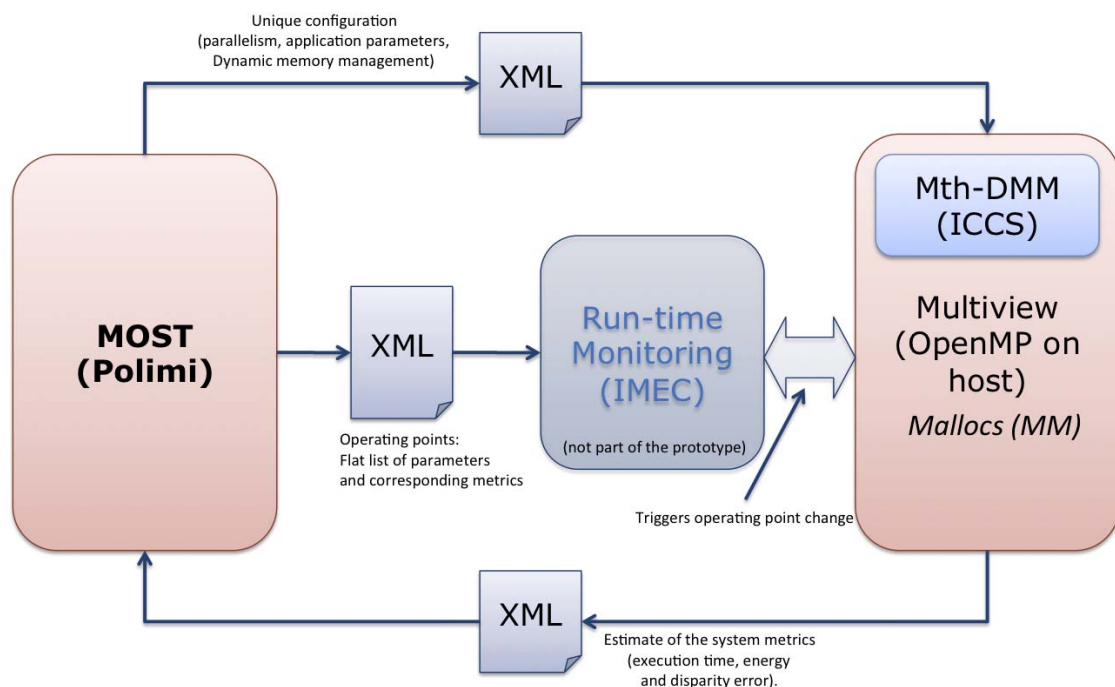
II. Prototype Architecture

The architecture of the prototype is shown below in the figure. The most important components are:

- The design space exploration tool (MOST, provided by Polimi)
- The Multiview application (OpenMP port of the CUDA application provided by IMEC)
- The run-time dynamic memory manager (Mth-DMM, provided by ICCS)

The goal of the prototype is to create a list of *operating points*, to be used by the *run-time monitoring* routines, by probing the configuration space of the Multiview application while identifying trade-offs on the system metrics.

The Multiview application and the DSE tool have been integrated by using the Multicube XML R1.4 interface. The XML output format used by the DSE tool to provide the operating points to the run-time monitoring is also derived from Multicube integration efforts.



These application parameters correspond to the application level features that can be changed at run-time. The tuple [parameter-values, system-metrics] corresponds to an *operating point*. Among the parameters we can find:

- Application level parameters:
 - ArmLengthMax, DprStep, Threshold (Multiview Heuristic Parameters, see D1.2)
- Run-time parallelism parameters: Number of threads (OpenMP)
- Dynamic memory management parameters:

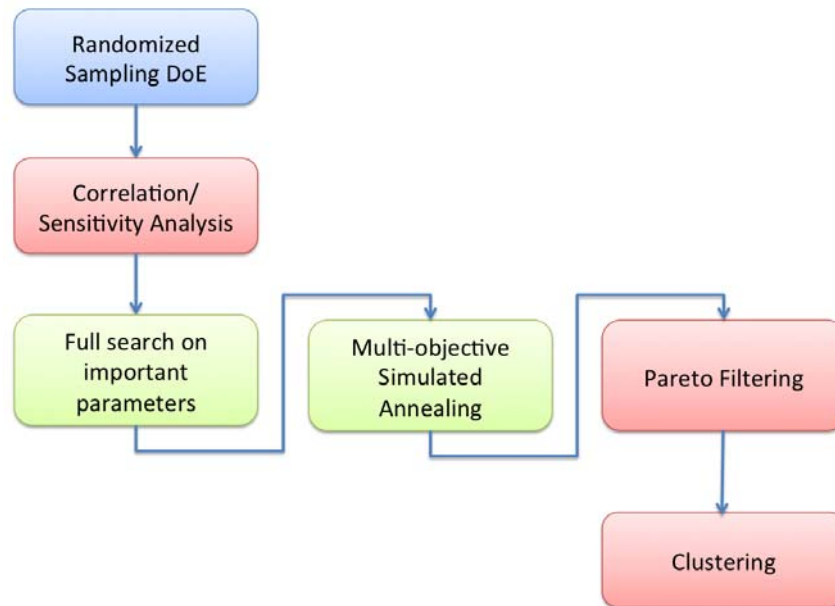
Among the system metrics we can find:

- **Wall time**
- A proxy of **Energy consumption**
- **Disparity Error**
- **Memory fragmentation**
- **Memory footprint**

II.1 Prototype flow

The prototype flow is shown in the following figure. This strategy is called Revised Sensitivity Analysis (RSAn) and it is an extension of some previous research in the field of Design Space Exploration (see [1]). It is composed of 7 steps whose outcome is the generation of a clustered set of operating points:

1. **Randomized Sampling - Design of Experiments.** In this phase, the original design space (several millions of combinations) is sampled to gather initial, coarse grained statistics. Each *sample* consists of the invocation of the Multiview application on a set of 7 reference images (see next Section).
2. **Correlation/Sensitivity Analysis:** Interaction effects (by means of HTML reports) and correlation between parameters and system metrics are derived to identify the most important parameters of the design space (i.e., those impacting majorly the objective system metrics). This analysis is done by using some tools available within the MOST framework
3. **Full search on important parameters:** once identified significant parameters, a full search is done over this sub-design space. In the Multiview use case, both application parameters and parallelism parameters have been identified as ‘important’.
4. **Multi-objective Simulated Annealing.** The full search results are refined by means of a randomized search to escape from local minima. We have chosen Simulated Annealing for this phase.



5. **Pareto filtering and clustering.** The results of the MOSA algorithm are then Pareto-filtered and clustered around few clusters by using k-means clustering.

II.1 Robustness with respect to workload









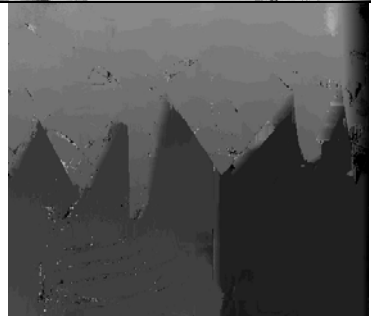






A workload can be defined as a *relationship between [...] an operator and task demands*¹. In this deliverable we consider the task demand of the application as a sequence of variable size image pairs for which the disparity map should be computed (See D1.2).

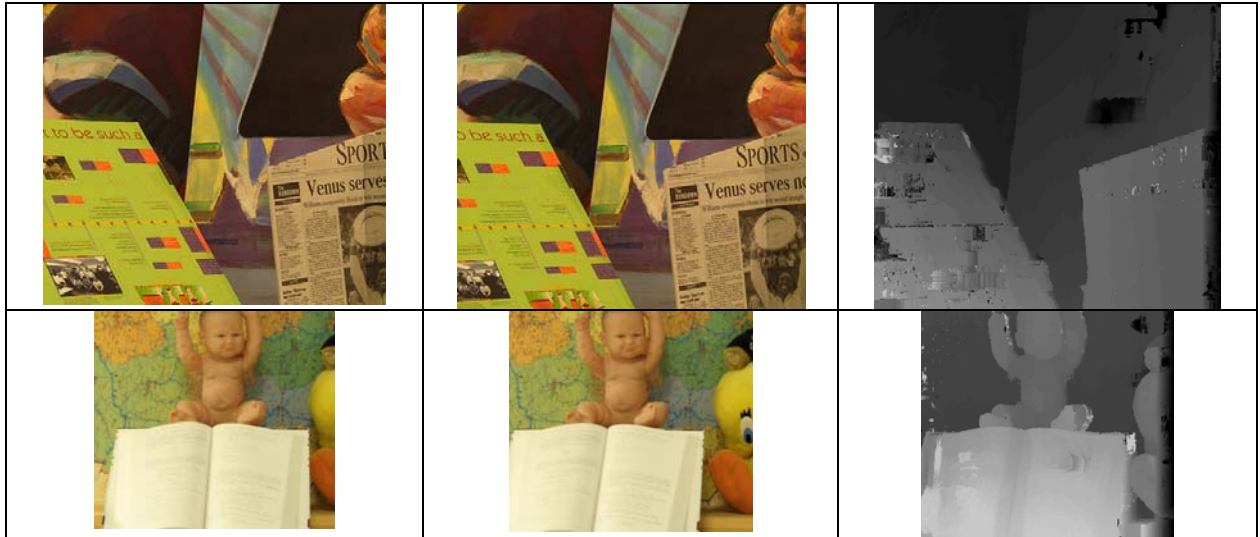
As the content of the images can vary, we can safely assume to deal with a *variable* workload. Our goal is to guarantee that the system metrics associated with the operating points derived in step 5 of the methodology vary as little as possible when dealing with the different input images. To this end, each parameter configuration is evaluated on 7 different image pairs. The resulting system metrics are then averaged with a mix of arithmetic and geometric mean².

The reference image pairs are shown in the next page.

¹ <http://en.wikipedia.org/wiki/Workload>

² Geometric mean is used when the variation range of the samples is comparable with the absolute value of the samples. It tends to dampen the effect of very high and very low samples that would bias the arithmetic mean.

<i>Left image</i>	<i>Right image</i>	<i>Ref. Disparity Map</i>
		
		
		
		
		



III. Installation Instructions

The prototype can be downloaded from the public area of the 2PARMA website:

<http://www.2parma.eu/download-area/deliverables.html>

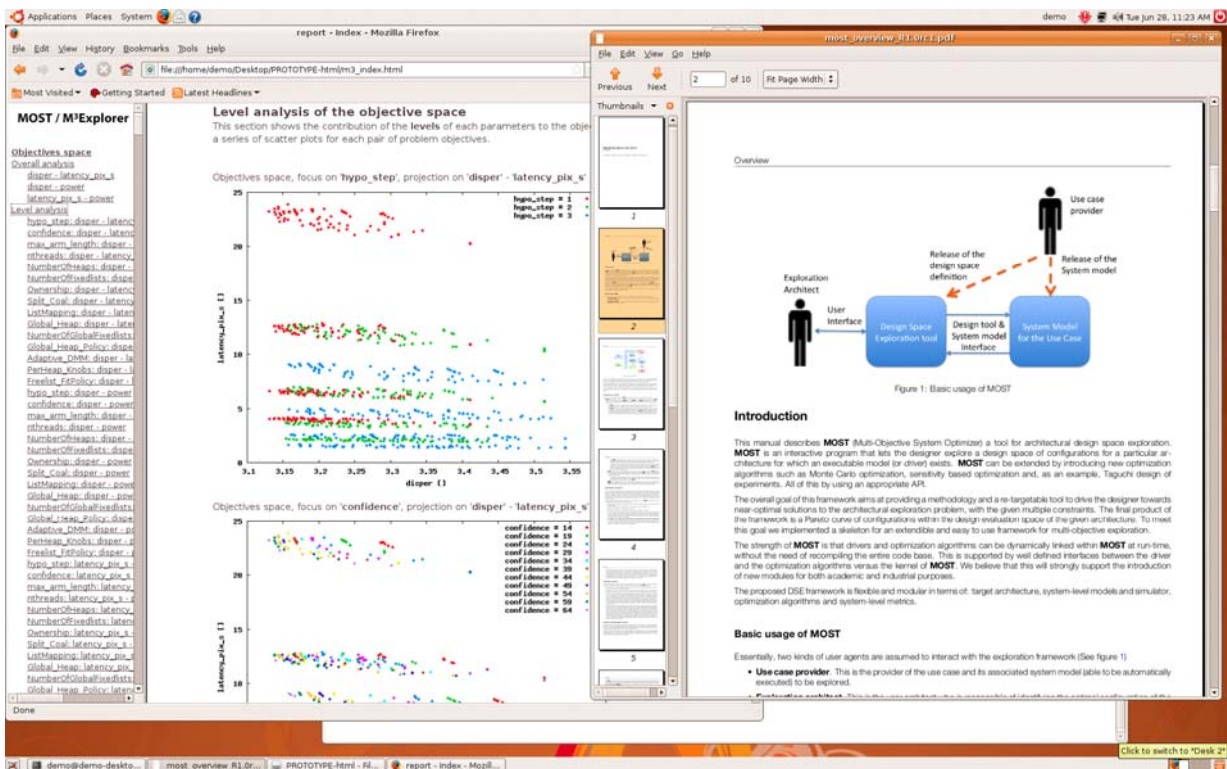
The deliverable is in the form of a VM-Ware Ubuntu 8.04 Virtual Machine. Once it has been booted, login into the virtual machine with the following data:

Login: *demo*

Password: *demo*

The prototype can be run by using a terminal; you will find a terminal icon and a link to the documentation on the desktop appearing just after login. In order to run the prototype, follow the instructions in the next section or, alternatively, those provided in the README.proto.html file on the Desktop of the virtual machine.

Here you can find a snapshot of the running virtual machine:



IV. How to run the prototype

1. Open a terminal
2. > cd prototype/stereo_matching

Important Note: You can either directly run only the significant step of the methodology (for generating operating points, **step 5**) or run each of the steps singularly. So if you want to directly generate the operating point list, jump to step 5. All the databases have been pre-populated before the prototype release but they can be re-generated by repeating the steps of the methodology as follows:

1. Randomized Sampling - Design of Experiments.

```
> make p0
```

This step invokes the MOST tool with the following command line:

```
$(MOSTIMAGE)/bin/most -x design_space.xml -f most_scripts/0_proto_rsample.scr
```

The script performs a batch of randomized executions of the Multiview application. To gather an initial database of experiments (written into databases/RSAMPLE.db).

Warning: the exploration is very long. This step can be skipped since a pre-populated database has been provided with the prototype.

2. Correlation/Sensitivity Analysis:

```
> make p1
```

This step invokes the MOST tool with the following command line:

```
$(MOSTIMAGE)/bin/most -x design_space.xml -f most_scripts/1_proto_analysis.scr
```

The script performs the correlation analysis associated with the design space. An HTML report can be optionally used to perform main effect analysis. The HTML report can be found in ~/prototype/stereo_matching/html (file *m3_index.html*) and it is also provided on the Desktop of the virtual machine.

3. Full search on important parameters:

```
> make p2
```

This step invokes the MOST tool with the following command line:

```
$(MOSTIMAGE)/bin/most -x design_space.xml -f most_scripts/2_proto_fsearch.scr
```

The script performs a full search on the most important parameters of the application. In our experiment, they have been identified as being the Multiview application parameters and the number of threads.

4. Multi-objective Simulated Annealing.

> make p3

This step invokes the MOST tool with the following command line:

```
$(MOSTIMAGE)/bin/most -x design_space.xml -f most_scripts/3_proto_mosa.scr
```

The script refines the full search by navigating through the design space associated with the dynamic memory management parameters

5. Pareto filtering and clustering.

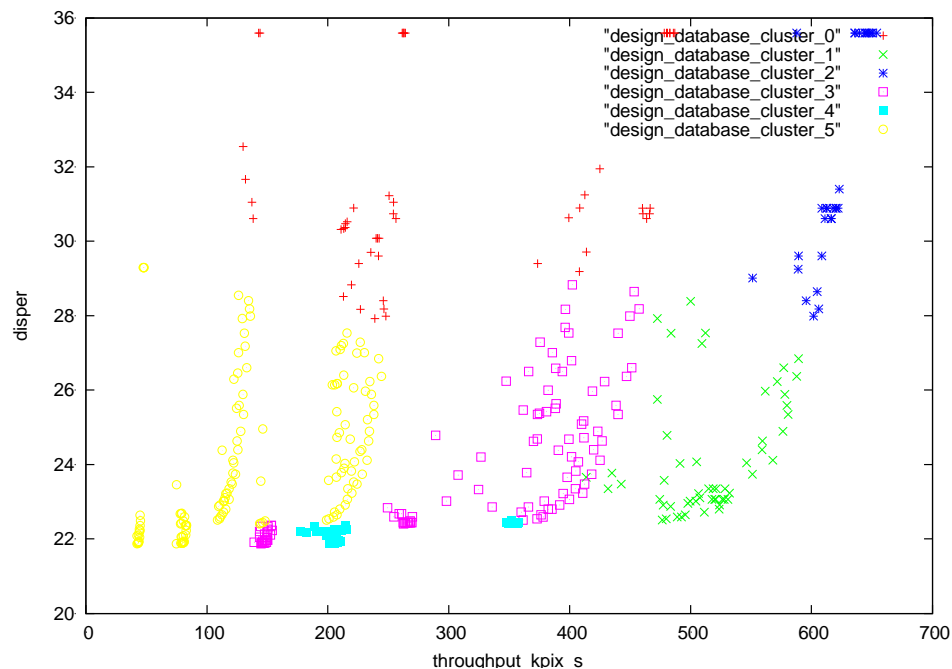
> make p4

This step invokes the MOST tool with the following command line:

```
$(MOSTIMAGE)/bin/most -x design_space.xml -f most_scripts/4_proto_gen_op.scr
```

This script reads all the databases generated (or pre-populated) in the previous steps and generate the clustered XML list of operating points in the file ‘clus.xml’.

The last step, generates a set of postscript files showing the system metrics of the clustered operating points (different colors correspond to different clusters). The following is a representative image generated with MOST for the considered Use Case (showing the throughput per pixel and the disparity error for the generated clusters (6)):



V. References

- [1] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria. A sensitivity-based design space exploration methodology for embedded systems. *Design Automation for Embedded Systems*, 7(1):7-33, September 2002