CoherentPaaS

Coherent and Rich PaaS with a Common Programming Model

ICT FP7-611068

Hypervisor Resource Usage Monitoring and Profiling (initial version)

D8.3.1

September, 2014

Document Information

Scheduled delivery	30.09.2014
Actual delivery	24.10.2014
Version	1.0
Responsible Partner	FORTH

Dissemination Level: RE

PU Public
PP Restricted to other programme participants (including the Commission)
RE Restricted to a group specified by the consortium (including the Commission)
CO Confidential, only for members of the consortium (including the Commission)

Revision History

Date	Editor	Status	Version	Changes
06.10.2014	A. Bilas	Draft	0.1	Table of contents and initial version
06.10.2014	S. Papageorgiou	Draft	0.2	Content
07.10.2014	A. Bilas	Draft	0.3	Review and comments
08.10.2014	S. Papageorgiou	Draft	0.4	Edits
09.10.2014	A. Bilas	Draft	0.5	Review
10.10.2014	S. Papageorgiou	Draft	0.6	Release for project review
12.10.2014	D. Dominguez-Sal	Draft	0.6	Comments
13.10.2014	V. Spitadakis	Draft	0.6	Comments
15.10.2014	A. Bilas	Prefinal	0.7	Edits
24.10.2014	A. Bilas	Final	1.0	Edits

Contributors

S. Papageorgiou (FORTH) and Angelos Bilas (FORTH)

Internal Reviewers

David Domínguez-Sal (Sparsity), Vasilios Spitadakis (Neurocom)

Acknowledgements

Research partially funded by EC 7th Framework Programme FP7/2007-2013 under grant agreement n° 611068.

More information

Additional information and public deliverables of CoherentPaaS can be found at: http://coherentpaas.eu

1. Executive Summary

Performance analysis and evaluation are integral parts of the development cycle and a cornerstone in deploying new technologies in real systems. Kernel-space modules pose a different set of challenges compared to ordinary user-space applications when it comes to profiling: limited visibility inside the kernel, high concurrency, interactions with multiple layers above and below the deployed module and multiple entry points are some of the challenges.

In this work we present a methodology that combines fine-grain, kernel-level profiling with metrics provided by the operating system in order to create a visual representation of the execution. We present results in the form of stacked bar graphs that visualize system performance and we quantify modules where the CPU spends its time. In addition, by performing additional analysis over profiling results and by cross-correlating with kernel source code, we highlight underlying kernel subsystems and succeed in providing a more meaningful image of these metrics, while at the same time limiting information loss.

We implement this methodology in the form of the SSWAT framework. We provide a detailed overview of SSWAT's structure, features and capabilities. We also examine interesting cases to show how this framework can be used to analyze system performance under various scenarios. In the next Period SSWAT will be integrated with the x-ray infrastructure by logging its performance events in the common framework and allowing x-ray to use these events for overall performance analysis.