[1] Constrained Collaborative Power Management

Bellasi Patrick*, Bosisio Stefano*, Carnevali Matteo*, Fornaciari William*, Melpignano Diego†, Siorpaes David†

*Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

†AST, STMicroelectronics, Agrate Brianza, Italy

DATE Conference 09

20-24 April, 2009, Nice-France

Introduction

The shortening of the overall design time of new generation and complex SoC's based systems is enforcing more *comprehensive design approaches* and the adoption of solutions that are easily portable among different products. General purpose operating systems like Linux, because of their predisposition to adapt easily to different application contexts, are a common choice for many new generation mobile devices. Being a key feature to improve mobility, *energy efficiency* has become a high priority design goal, and the implementation of the necessary mechanisms to optimize both power and performances can no longer disregard from the requirements of ease of development, portability and adaptability.

This work presents a *formal model* to define the problem of power vs performance control. Starting from the current Linux solution we advance the proposal for an extension that is better tailored to embedded mobile systems. The proposed solution is *competitive in adaptability* and ensures *better control on performances* and *ease of implementation*. The value and accuracy of this approach is being quantitatively and statistically proved through extensive experiments carried out on a development board designed for multimedia applications.

Main Approaches to PM

- Centralized (Fig.1) - decisions are delegated to a single entity generally based on an analytical model of the entire system. In spite of providing accurate results, in efficent and reliable way, model building and maintenance is generally more complex [2].
- Distributed (Fig.2) - drivers run a local optimization policy and transparently exchange informations, thanks to a shared communication mechanism, that constraint local policy's decision. This approach usually grants better scalability and maintanability, while overhead introduced by the communication protocol must be considered and minimized.

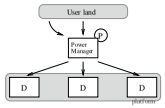

Figure 1: Centralized PM approach

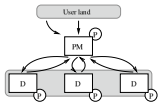

Figure 2: Distributed PM approach

QoS Framework (pm_qos_interface) [3]

Kernel infrastructure to implement coordination among *drivers* and *applications*. Devices talk in terms of constraints (*latency*, *timeout* and *troughput*) and the communication is performed through notification queues. This framework shows some limitations:

- no additive constraints concept (Fig.3)
- best effort approach: drivers express requirements and hope others fulfill them (Fig.4)

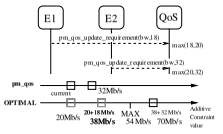

Figure 3: Additive constraints' limitation

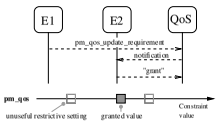

Figure 4: Best effort limitation

Problem Formalization

Mapping Abstract System-wide Metrics (ASM) to devices' *operating modes* (Fig.5) is the basic mechanism that guarantees collaboration among drivers in achieving the goal of control.

- an operating mode of a device corresponds to some *devices' local configurations* (c_{ij})
- ASMs are parameters (p_i) used to represent QoS requirements and describe the dynamic behavior of a system

The mapping of devices' operating modes to ASMs can be represented in the n-dimensions System-Wide Configuration Space (SWCS). In this space (Fig.6) the mapping of each device's operating mode identifies a Device Working Region (DWR). Feasible System Configurations (FSC) corresponds to regions of the SWCS where there is overlapping of at least a DWR for each device.

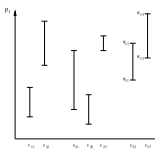

Figure 5: Operating modes to ASMs mapping

Problem Formalization (continue)

To support validation of the model and to deepen some aspects of the problem a Linear Programming (LP) Formulation is given. The *solution space* is represented by the SWCS and the QoS requirements are *constraints* (v_i) cutting that space, thus reducing the number of FSCs. The smallest polygon containing all the valid FSCs can be demonstrated to be a *convex hall*. QoS optimization goals ($\vec{g}$) define the *objective function* of the LP problem.

Features of LP formulation

- *Static Analysis* (e.g. unfeasible DWR)
- FSCs' *partial ordering* based on metrics (e.g. power consumption)

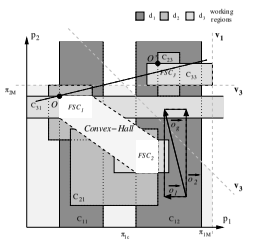

Figure 6: LP Formulation in the SWCS

Model design

The proposed model is based on few main entities:

- Execution Context - describes the user space perspective of a system, can be characterized with use-cases and provides QoS requirements
- QoS Information Manager - central component of the framework. Acting as ASMs' informations broker, it manages communication among other entities and grants correct update of ASMs
- Drivers - run local optimization policies and are completely independent each other; are influenced by ASMs or may affect them. To change the value of an ASMs a driver requests authorization

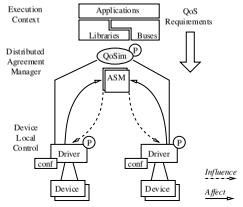

Figure 7: Model overview

System Evolution and Temporal Domains

System evolves through different states that can be identified in FSCs. A new state is entered when a driver wants to change its internal configuration. Consequently it proposes a new set of constraint's values. This proposal triggers a distributed agreement process where these new values must be negotiated and then notified with the others devices. Two temporal domains can be identified to describe the dynamics.

Operating Point Performance History (OPPH - Fig.8)

- *"long term"* temporal domain
- sequence of *selected FSCs*
- ensures framework *efficiency* on a wide time span

Distributed Decision Process (DDP - Fig.9)

- *agreement process* triggered by a ASM's update request
- drivers' state is *"frozen"* when DDP begins
- supports conditional agreement
- if agreement is reached a new FSC is generated

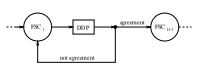

Figure 8: Digraph representing the OPPH

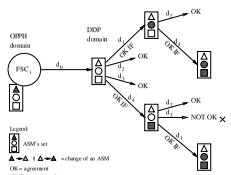

Figure 9: Tree structure representing the DDP

Implementation of DDP

Several approaches can be used to implement the DDP, the most promising is a breadth first-like serialization to visit the tree (Fig.10):

- the root of the tree contains the ASMs set with the proposed updated value(s)
- *update requests* are communicated to interested drivers
- drivers may agree, not agree or express the conditional agreement
- in case of *not agreement* the process stops, else a new FSC is generated

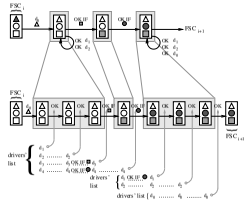

Figure 10: Breadth first-like serialization

Model Performance and Criticalities

Waste vs Overhead inside DDP

In the proposed framework DDP is a key process that impacts on performances. It requires computation, memory and time resources introducing either overhead, if it comes out with a new FSC, or waste when a not agreement is reached. A *smart implementation* is thus necessary to achieve good performances.

Main criticalities, related to the agreement process among several entities, consist in the possibility to fall in with loops or cycles in different temporal domains.

- OPPH loops - the system normally "evolves" across a serie of FSCs. This could lead to the generation of periodical system behaviors that are generally reasonable according to the running use-cases. Nevertheless, when these cycles are too fast, they *would prevent global optimization* if changes' overhead is higher than the obtained benefits.
- DDP cycles - arise when the *same ASM* is continuously updated as a result of several conditional agreements. Observing the *trend* (Fig. 11) of subsequent updates a cycle can be detected and so, the final ASM's value can be foreseen, hence reducing *overhead*
- DDP Loops - represent a deadlock situation that must absolutely be avoided, they happen during a DDP when a previously seen state is met again: in this case, due to the *determinism* that characterizes the process, all the previous decision would be infinitely *taken again* (Fig. 12).

Trend analysis A *smart implementation* for the DDP has been designed in order to reduce the possibility of critical DDP loops. Constraint's value changes are monitored and a trend analysis (Fig. 13) allows to detect unstable situations that prevent the convergence to a distributed agreement (Fig. 14).

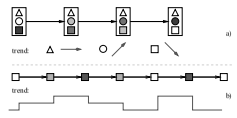

Figure 11: Cycles in DDP with different trends

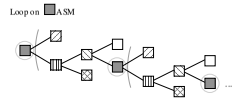

Figure 12: Loops in the DDP

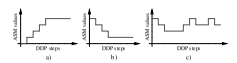

Figure 13: Trend taxonomy a) increasing b) decreasing c) floating

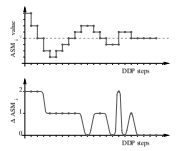

Figure 14: Floating trend and related Δ

Logical Design

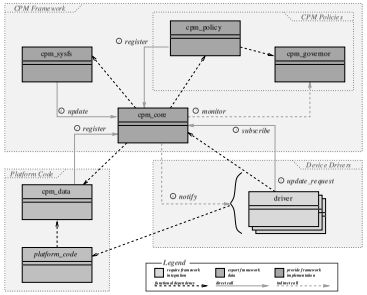

Figure 15: Class Diagram

The proposed model has been implemented as a Linux Kernel framework named cpm whom architecture is shown in Fig. 15.

To use this framework on a mobile device the only *platform specific code must be provided*, declaring the set of supported ASMs.

Drivers use the framework transparently with respect of a specific platform. The binding to platform's ASMs will be declared by the platform code itself.

Policy and governors modules allow to *experiment different optimization strategies*, the trend analysis implementation is provided as default governor.

References

[1] P. Bellasi, S. Bosisio, and M. Carnevali, *"Constrained collaborative power management"*, Politecnico di Milano, Internal report, March 2009.

[2] P. Bellasi, W. Fornaciari, and D. Siorpaes, *"Predictive Models for Multimedia Applications Power Consumption Based on Use-Case and OS Level Analysis"*, Proceedings of the DATE Conference, April 2009.

[3] M. Gross, *"Power QoS"*, web, September 2007. [Online]. Available: http://www.lesswatts.org/projects/power-qos/

{bellasi, fornacia}@elet.polimi.it {stefano.bosisio, matteo.carnevali}@mail.polimi.it {diego.melpignano, david.siorpaes}@st.com