A bright idea for Embedded Systems
System-Wide Dynamic Power Management

Review on the IBM and MontaVista Software proposal for Embedded Systems Power Management.

Patrick Bellasi

bellasi@elet.polimi.it

Research Assistant
Departement of Electronics and Information Technology
Politecnico di Milano

September, 22 - 2006

Outline

1. Power Management
   - The Problem Space
   - The Solution Space

2. Proposed Architecture
   - Policy Architecture
   - Implementation
   - Usage

The DynamicPower Project

- a SourceForge project
  (registered 2003-09-18, last update 2006-04-14)
- sponsored by IBM Research Labs and MontaVista Software
- is a patchset against linux kernel 2.6.16
  (status: alpha-stable)
- platforms currently supported: Intel Centrino Enhanced Speedstep, TI OMAP, PowerPC 405LP Arctic III and Intel PXA27x
  - some configuration example are also provided
  - userspace tools for DPM configuration and policy management

Introduction

- traditionally the focus was on regulating the power consumption in static modes (sleep and suspend)
- DPM refers to power management schemes implemented *while programs are running*
- should exploit recent processor support for very dynamic power management strategies
  - based on dynamic voltage and frequency scaling
- power states control must be implemented in the operating system
  - highly integrated System-On-a-Chip (SOC) processors typically do not include a traditional BIOS

DynamicPower Proposals

- attempts to standardize a dynamic power management and policy framework
- support different power management strategies
  - under control of operating system components
  - or user-level policy managers
- provide a flexible framework, mostly architectural independant
  - exploiting last 2.6 linux kernel features
    (e.g. LDM, Hotplug, . . .)
- addressed to system-wide power management

Functional Requirements

- reduce *system-wide* energy consumption
    - voltage and frequency scaling the processor core may be of limited use
    - scaling bus frequencies
    - manage devices power consumption
- look for *highly dynamic* power management strategies that encompass the entire system
- *fine grained* power and performance characteristics definitions

    task-specific dynamic power management will become a hard requirement in highly energy-constrained systems

Implementation Requirements

- simplicity and flexibility

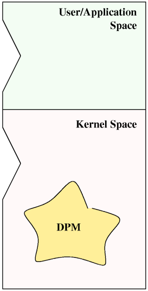
  leaving the workings of the dynamic power management system completely transparent to most tasks, and even to the core of the operating system itself
- safety and portability
  - don't actually manage device state
  - relay on low-level device drivers
- support "pluggable" power management policies

  most effective way to manage energy consumption are highly "application" dependent
- exploit the capabilities of state-of-the art systems and techniques for PM provided by modernn SoC devices

Architectural Overview

The DPM Core

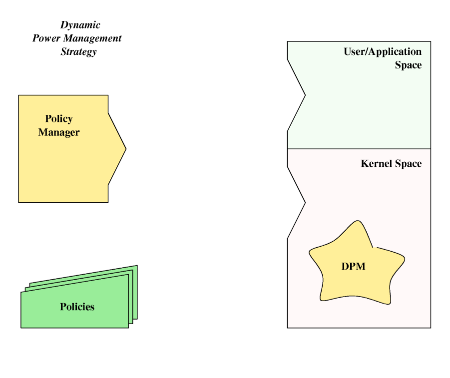
- a low-level DP component: implemented in the kernel space
- not a self-contained device driver: requires enhancements at a few key places



ARCHITECTURAL OVERVIEW

DPM strategy components

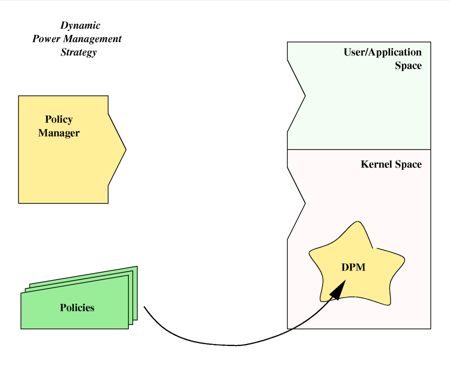
- predefined set of policies
- policy manager that manage policies activation



Architectural Overview

The Policies

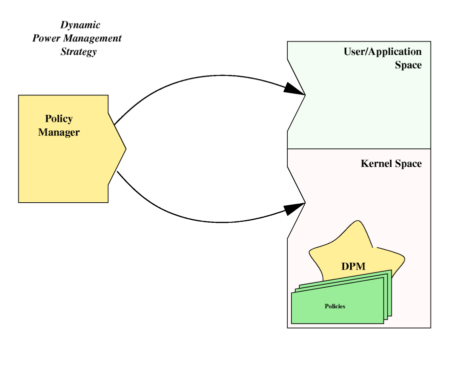
- are named data structures
- installed into the operating system kernel for efficiency
- specify the component and device-state transitions that ensure reliable operation in line with the power management strategy



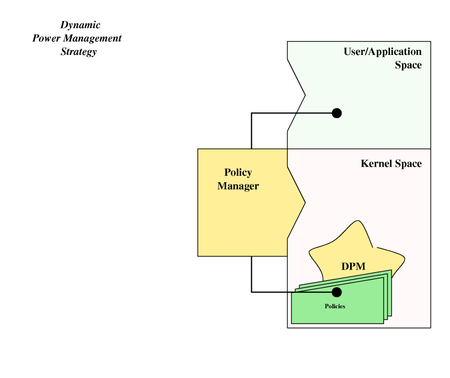
Architectural Overview

The Policy Manager

- can execute either as part of the kernel or in user space (or both) as required by the strategy
- provide to activate a suitable policy based on system state or user requests



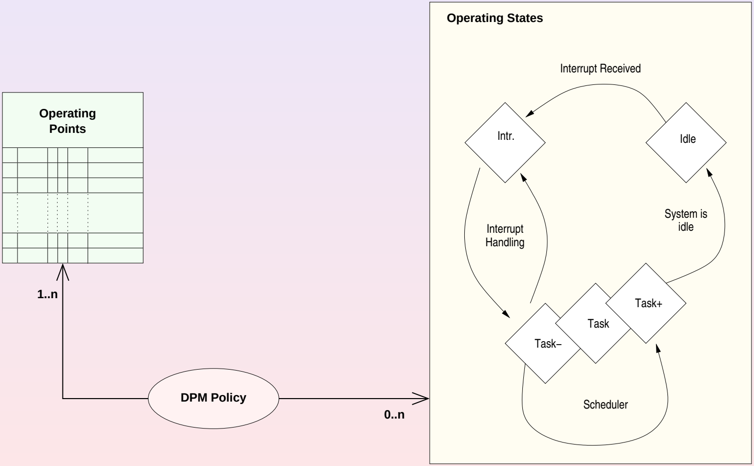
Architectural Overview



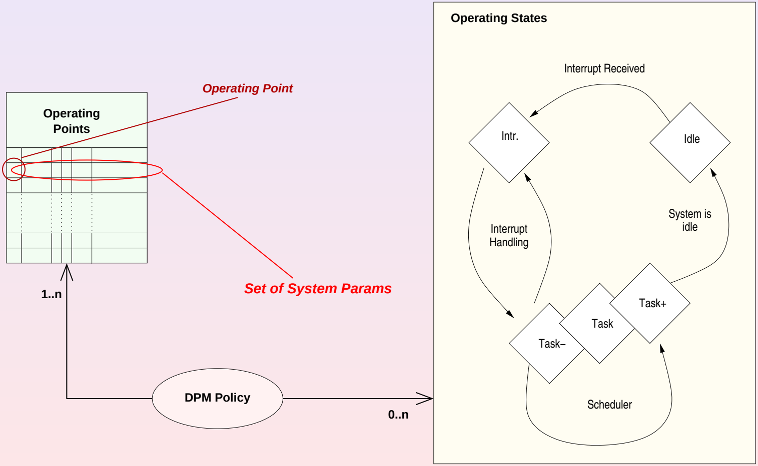
Policy Management

- activate policies by name
- may be very active or more passive
- effective strategies may even consist of a single policy installed at system initialization

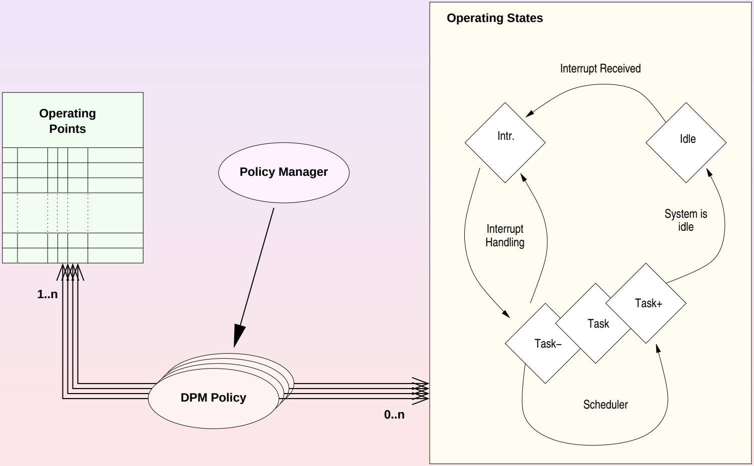
DPM Policy Overview



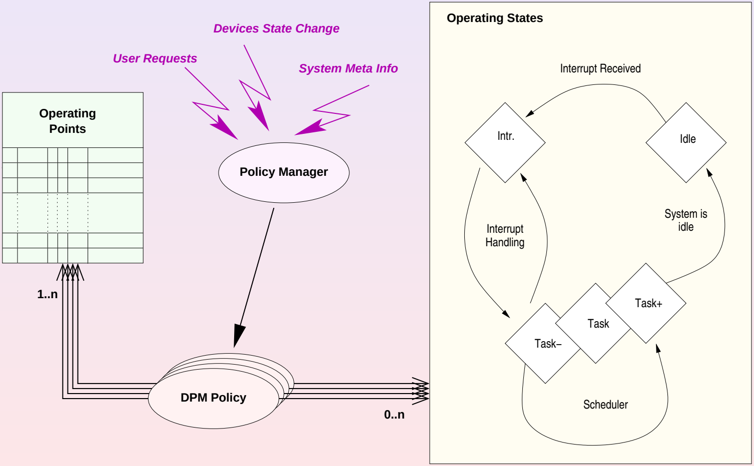
DPM Policy Overview



DPM Policy Overview



DPM Policy Overview



Operating Points

Operating Point

An Operating Point (OP) encapsulates the minimal set of inter-dependent, physical and discrete parameters that define a specific system performance level and energy cost.

- are processor and system dependent
- a system:
  - has many OP
  - at any given point in time is executing at a particular OP

Operating States

Operating State

An operating system can be thought of as a state machine moving through different states in response to events. Each one of these system states is an Operating State (OS).

- each operating state may be associated with an operating point
  - specific to the requirements of that state
- support task-specific operating points for power-aware tasks
  - tasks with special requirements may specify, or be specified to run in different *task states*, each of which may be associated with a different operating point

Congruence Class

Congruence Class

A Congruence Class (CC) is a subset of operating points that the system designer considers equivalent for specific operating states modulo a power management strategy

- given an OS each OP in the corresponding CC is acceptable
- *device constraints* might render some members of the class invalid
- *power considerations* might cause one OP to be preferred over other valid OP in the class

Device Constraint

Device Constraint

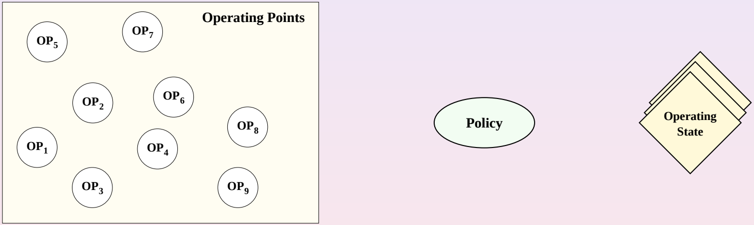
A Device Constraint (DC) is a device requirement associated with particular device state

LCD Example:

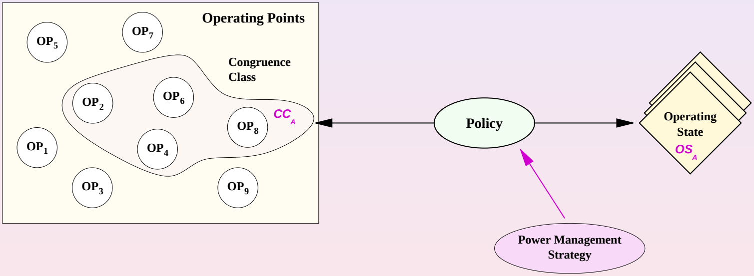
Active State pixel clock range $16 \div 25 MHz$

Idle State pixel clock range *undefined*

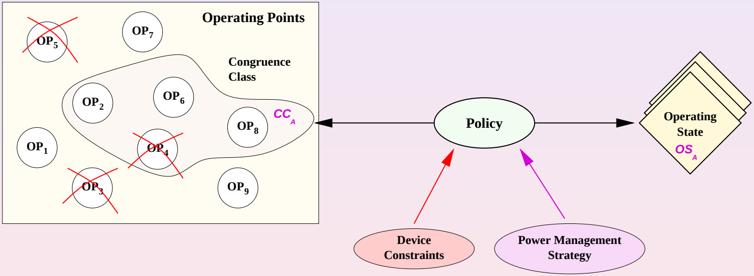
DPM Policy Reviewd



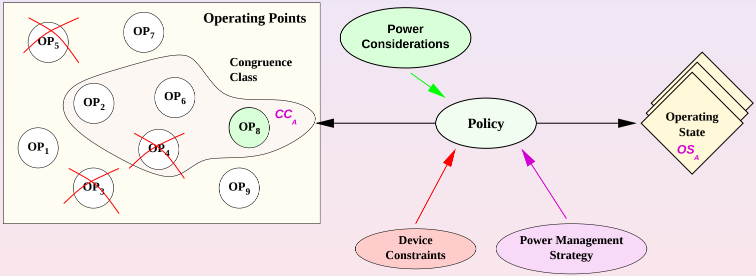
DPM Policy Reviewd



DPM Policy Reviewd



DPM Policy Reviewd



Strategy

Power Management Strategy

A Power Management Strategy (PMS) is a collection Policy P

- a DPM system has at least one Policy
- each policy is adressed to different running context (e.g. run on Battery, run on AC, . . .)

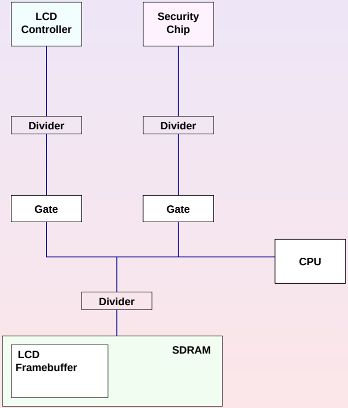
Policy Manager

Policy Manager

A Policy Manager (PM) is the component that activate a policy in reason of some events

- the decision of what policy to activate come from some collected information (e.g. Operatin System state, user preferences, running programs, phisical devices state, . . .)
- location, type of information collected and actions taken are implementation dependent

Operating Point View



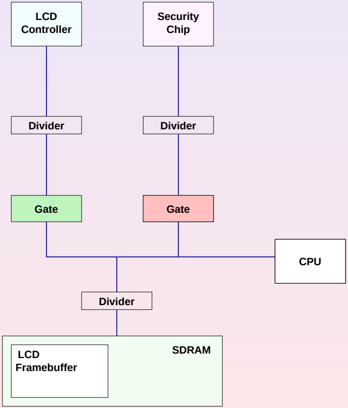
Running State

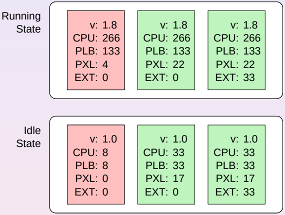
v: 1.8	v: 1.8	v: 1.8
CPU: 266	CPU: 266	CPU: 266
PLB: 133	PLB: 133	PLB: 133
PXL: 4	PXL: 22	PXL: 22
EXT: 0	EXT: 0	EXT: 33

Idle State

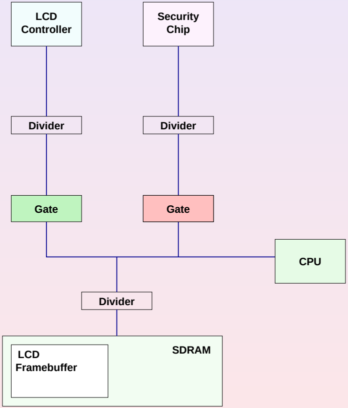
v: 1.0	v: 1.0	v: 1.0
CPU: 8	CPU: 33	CPU: 33
PLB: 8	PLB: 33	PLB: 33
PXL: 0	PXL: 17	PXL: 17
EXT: 0	EXT: 0	EXT: 33

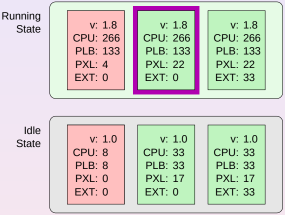
Operating Point View



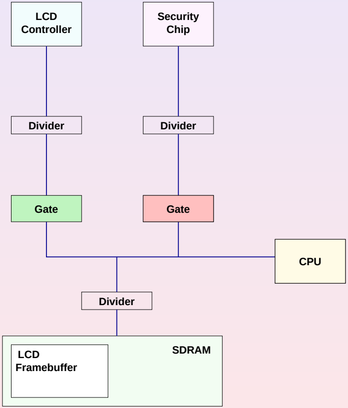


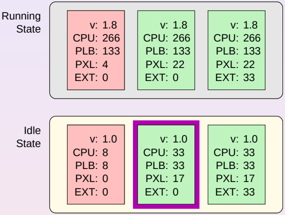
Operating Point View



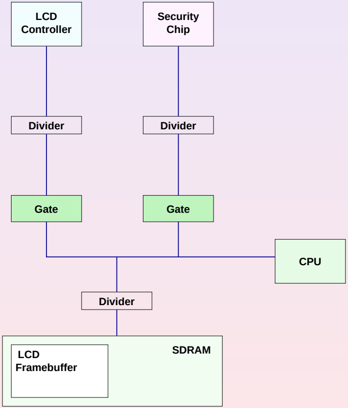


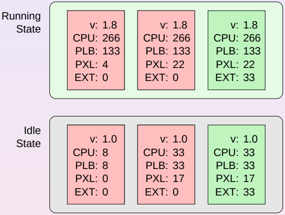
Operating Point View



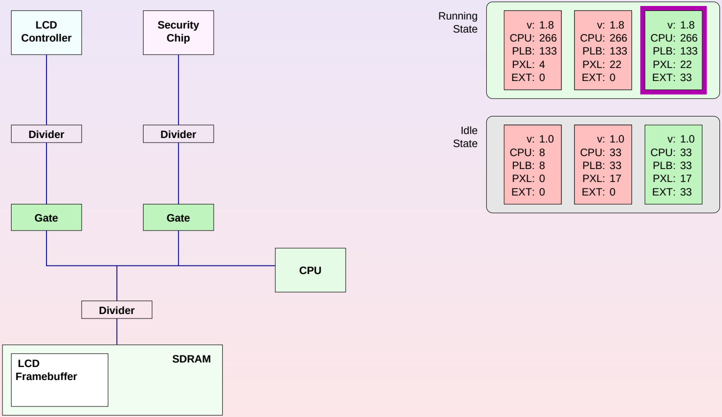


Operating Point View

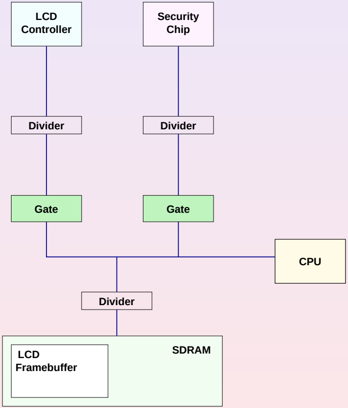


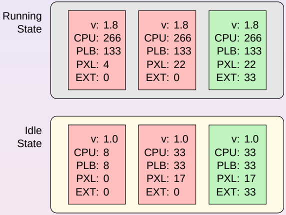


Operating Point View

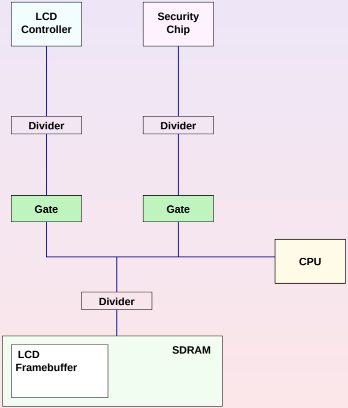


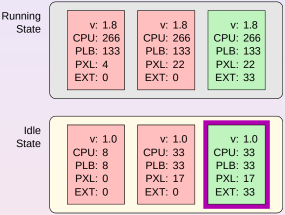
Operating Point View



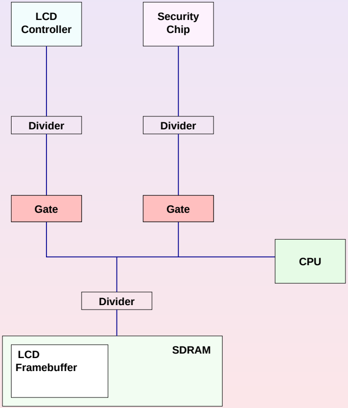


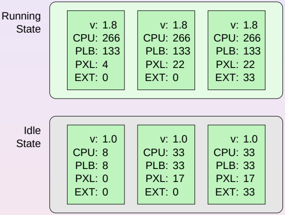
Operating Point View



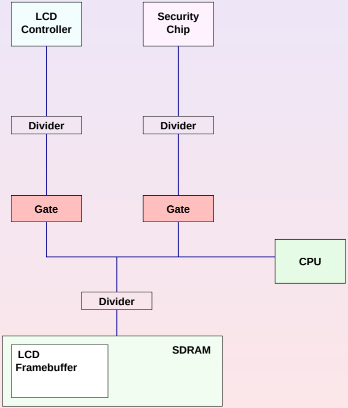


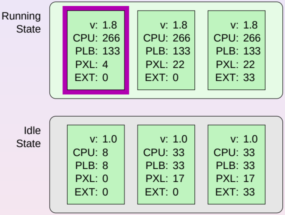
Operating Point View





Operating Point View





NOTES

- OP could be range-defined (insted of enumerated)

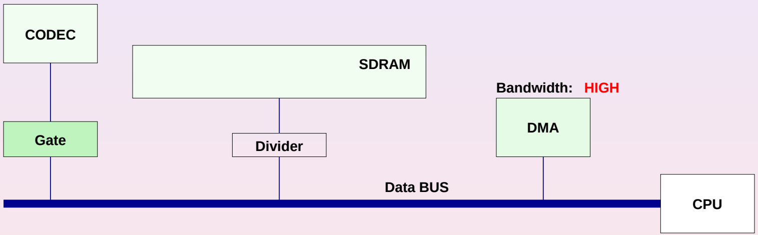
  OP ranges, device contraints and strategy rules provide a system whose solution is the optimal operating point to be activated

- OPs, Consguence Classes and Policy must be pre-coded; device contraints instead are knowe run-time and modify the set of valid OPs
- OP and device state changes must be transparent to the Operating System, which is free to move from state to state

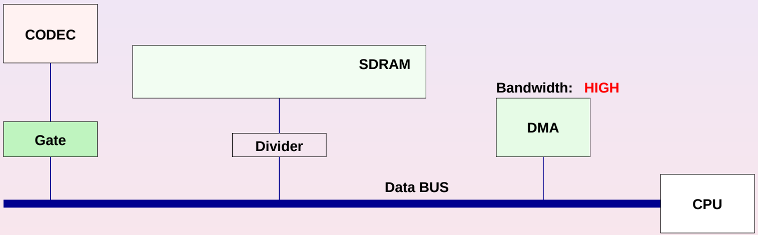
  even if device state changes shuld be performed by low-level device drivers:

  no single Device Driver has a complete view on how a new operating point will be reached

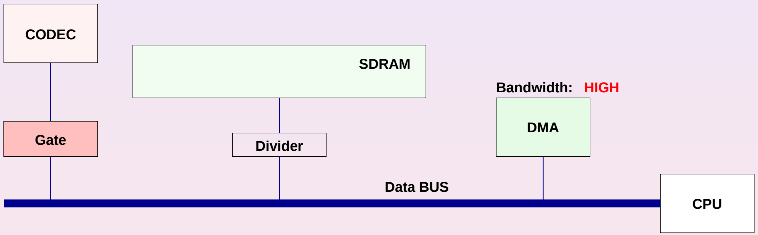
NOTES



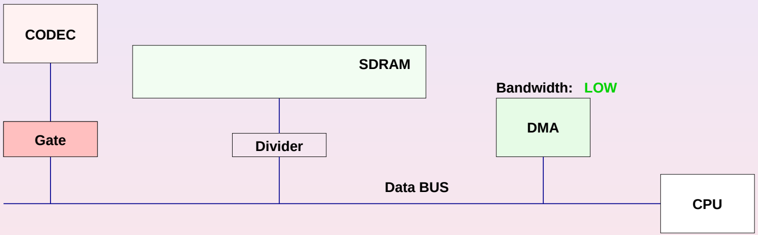
NOTES



Notes



Notes



PATCHSET COMPOSITION

- `include/linux`: DPM data structure definition, and update of some subsystems (process scheduler, device and power management)
- `include/linux/asm-(arm|i386))`: platform dependant DPM definitions (e.g. task states, board dependant callbacks, . . .)
- `drivers/dpm`: platform independent DPM core implementation, DPM callbacks and userspace interface (as a subsystem: `/sysfs/dpm`)
- `drivers/base/power`
- `arch/(arm|i386)/kernel/cpu/dpm)`: processor dependent DPM's handlers (processor voltage and frequency scaling support)

DPM Entry Points

1. `assert_constraints`
2. `remove_constraints`
3. `set_operating_state`
4. `set_policy`
5. `set_task_state`

- first tree in kernel context only, the last two also in user context (syscall)
- different callers (device drivers, scheduler, event handlers, Policy Manager or user)

Robustness

- an OP is valid $\iff$ it satisfies all device constraints
- a CC is valid $\iff$ at least one of its OP is valid
- a P is valid $\iff$ map each OS into a valid CC/P

Given that the system is initially running on a valid policy P

DPM implementation ensure that the current P will never become invalid

Task State

- support for task-specific OP

  a policie maps task-states to CC/OP
- `struct task` embeds a task-state descriptor that could be used by the scheduler (`set_operating_state`)

  RT processes and high power processes may not be correlated (e.g. MP3 player)
- syscall (`set_task_state`) to allow user-space changes
- the special task-state "`no-state`" allow a new scheduled task to run on current policy

  useful for system threads (`keventd`, `softirqd_*`), and frequent short run process, to avoid short duratono changes on OP

Example Power Srategies

Static

- one P only
- each task-state mapped to a single CC
- no needs for a userspace power manager

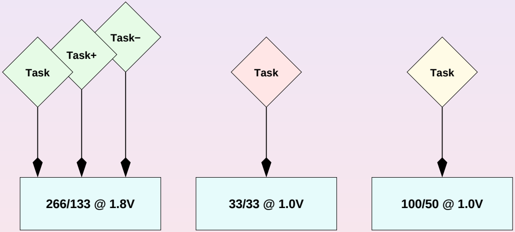
Simple Dynamic

- multiple (static) P
- use P settings to move the system throught OPs
- policy manager nedded

Task-State Dynamic

- like simple dynamic
- P associated to *“meta-info”* (e.g. battery level)
- *meta-info* changes trigger a P change

Static Power Srategy Example



Static Policy : no Policy Manager required

Dynamic Power Srategy Example

POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 OP_1	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

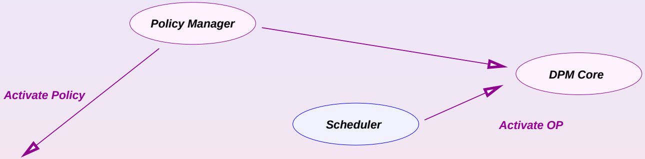
Dynamic Power Srategy Example

Policy Manager

Activate Policy

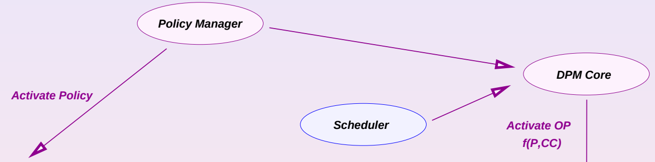
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 OP_1	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

DYNAMIC POWER SRATEGY EXAMPLE



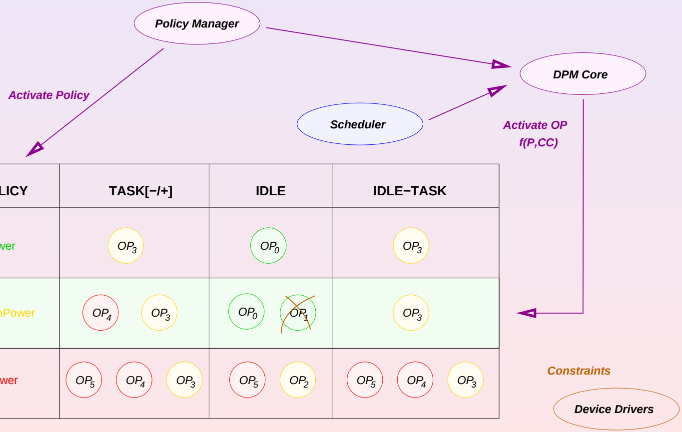
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 OP_1	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

DYNAMIC POWER SRATEGY EXAMPLE



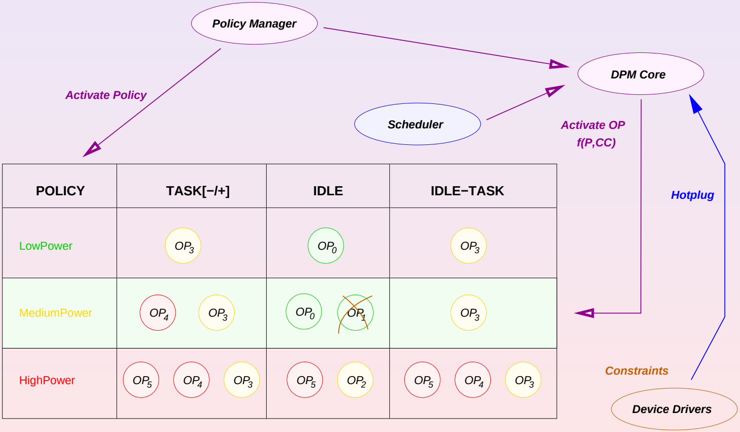
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 OP_1	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

DYNAMIC POWER SRATEGY EXAMPLE



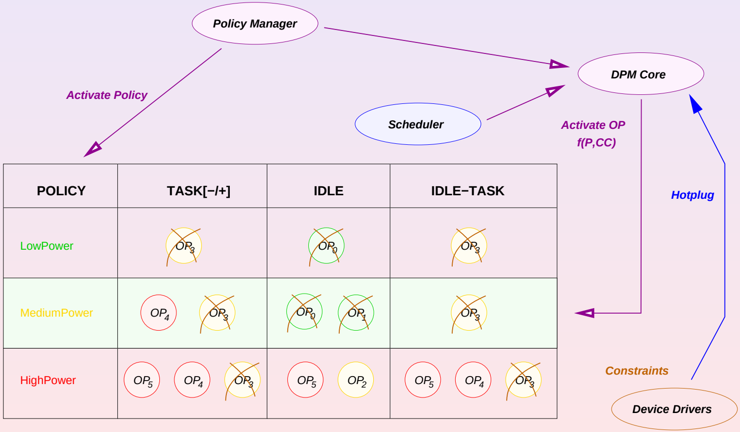
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 $\cancel{OP_1}$	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

Dynamic Power Srategy Example



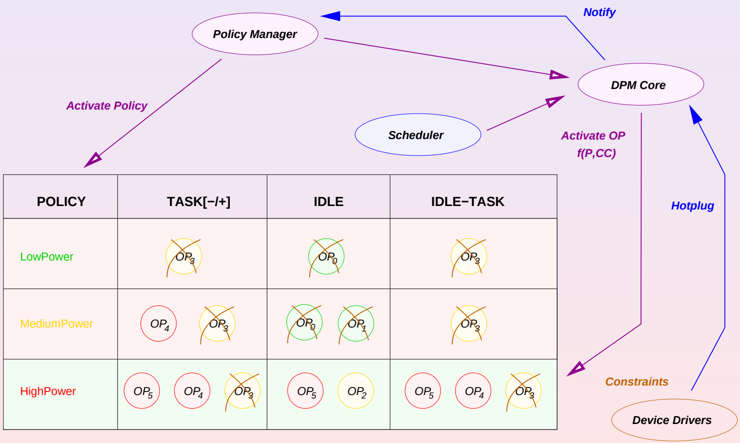
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	OP_3	OP_0	OP_3
MediumPower	OP_4 OP_3	OP_0 $\cancel{OP_1}$	OP_3
HighPower	OP_5 OP_4 OP_3	OP_5 OP_2	OP_5 OP_4 OP_3

Dynamic Power Srategy Example



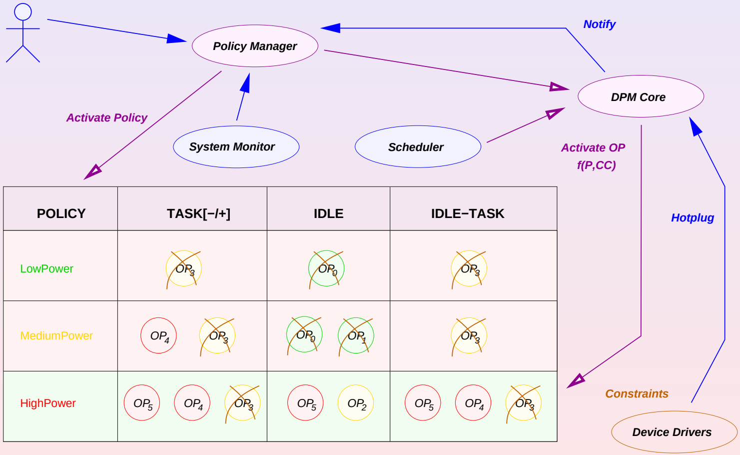
POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	$\cancel{OP_3}$	$\cancel{OP_0}$	$\cancel{OP_3}$
MediumPower	OP_4 $\cancel{OP_3}$	$\cancel{OP_0}$ $\cancel{OP_1}$	$\cancel{OP_3}$
HighPower	OP_5 OP_4 $\cancel{OP_3}$	OP_5 OP_2	OP_5 OP_4 $\cancel{OP_3}$

DYNAMIC POWER SRATEGY EXAMPLE



POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	~~OP_3~~	~~OP_0~~	~~OP_3~~
MediumPower	OP_4 ~~OP_3~~	~~OP_0~~ ~~OP_1~~	~~OP_3~~
HighPower	OP_5 OP_4 ~~OP_3~~	OP_5 OP_2	OP_5 OP_4 ~~OP_3~~

DYNAMIC POWER SRATEGY EXAMPLE



POLICY	TASK[−/+]	IDLE	IDLE−TASK
LowPower	$\cancel{OP_3}$	$\cancel{OP_0}$	$\cancel{OP_3}$
MediumPower	OP_4 $\cancel{OP_3}$	$\cancel{OP_0}$ $\cancel{OP_1}$	$\cancel{OP_3}$
HighPower	OP_5 OP_4 $\cancel{OP_3}$	OP_5 OP_2	OP_5 OP_4 $\cancel{OP_3}$

Userspace Interaction

- Sysfs support for OP, CC and Policy management
- hotplug events generation (on constraints changes, policy update, . . .)
- userspace (hotplug) power agent (could dinamically manage policy using the sysfs interface)
- userspace tools for initial configuration and runtime policy management

THE SYSFS INTERFACE

- Operating Points definition (`/sys/dpm/op/control`)
  syntax: *create <OP_name> [<OP_param>...]*
- Classes definition (`/sys/dpm/class/control`)
  syntax: *create <Class_name> [(<OP>)...]*
- Policies definition (`/sys/dpm/policy/control`)
  syntax: *create <Policy_name> [(<OP>|<CC>)...]*
- Active policies management
- DPM subsystem management (init, enable, disable)

Example of OP params for NDK10

System Clock Source (SCLK/CLK)

PLL, Low Speed Oscillator or High Speed Oscillator

PLL1 Multiplier (HCLK)

SDRAM frequency

PLL2 Gating

devices control

OP changes trigger a NDK run mode change or simply activate/deactivate some devices

The End