NomadGS

A Parameter based approach to Linux power management

Matthew Locke and Eugeny Mints

April, 2007

Agenda

- History
- Background
- Features/Goals
- Parameter framework
- API
- Key Internals
- Use Cases
- Issues

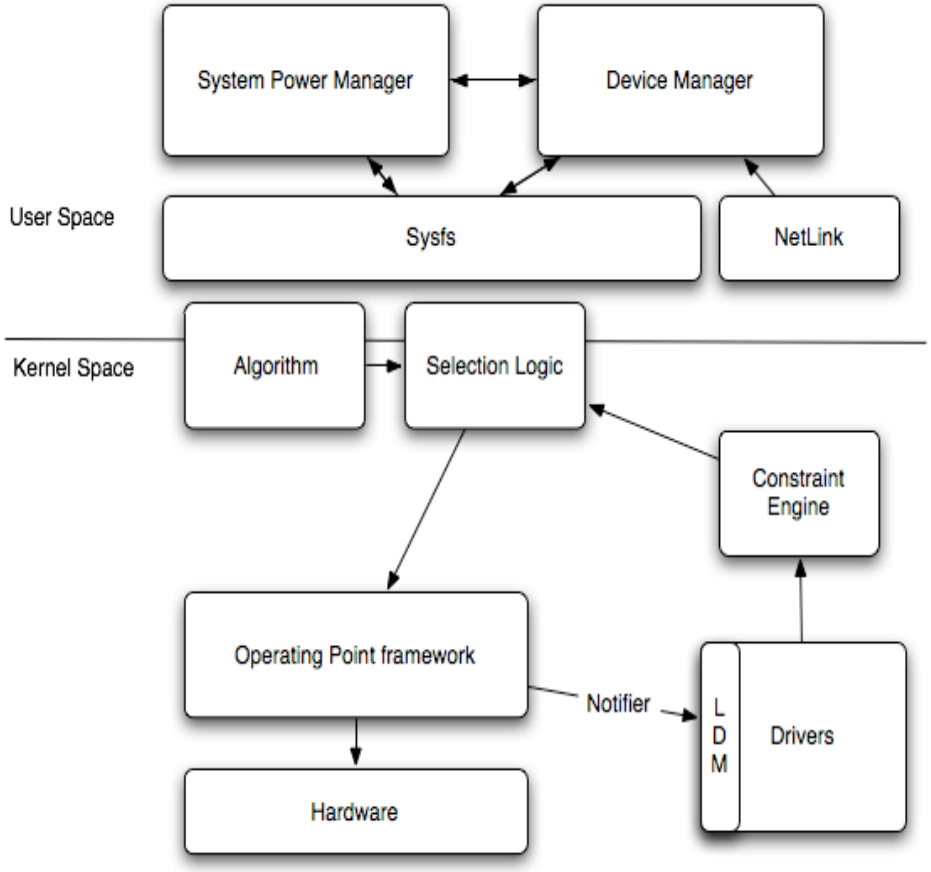
History

- All started from Dynamic Power Management (DPM) framework introduced in 2001 by Montavista and IBM
- Community rejected DPM and it wasn't pushed much further in the community
- In 2004, Todd Poynor (MV) submitted PowerOP which is the operating point layer from DPM. Not much traction.
- In 2006, Eugeny and myself (NomadGS) attempt to get PowerOP accepted by showing how it can be used on x86 as well as embedded.

- Becomes clear that the operating point concept won't work for every platform and therefore the wrong base abstraction.
- End of 2006, back to the drawing board.

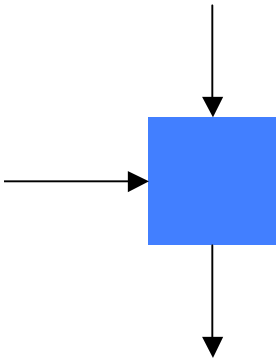
Operating Points

- Operating Points - set of system wide parameters that control power consumption.
- Parameters need to be set as a group for optimal power/performance balance or hardware dependencies
- Parameter values were platform specific - divider values not frequencies.
- Operating framework (PowerOP) maintained a list of valid operating points.
- Did not address local device driver power management.



Back to the Drawing Board
- Features and Goals

- Run time control of individual hardware resources that affect power consumption
  - Scale voltage and clocks; control power domains
- Track use count of hardware resources
  - Trigger action when use count is zero.
- Notify resource consumers when output value changes.
  - Subscribe for notification only when required.

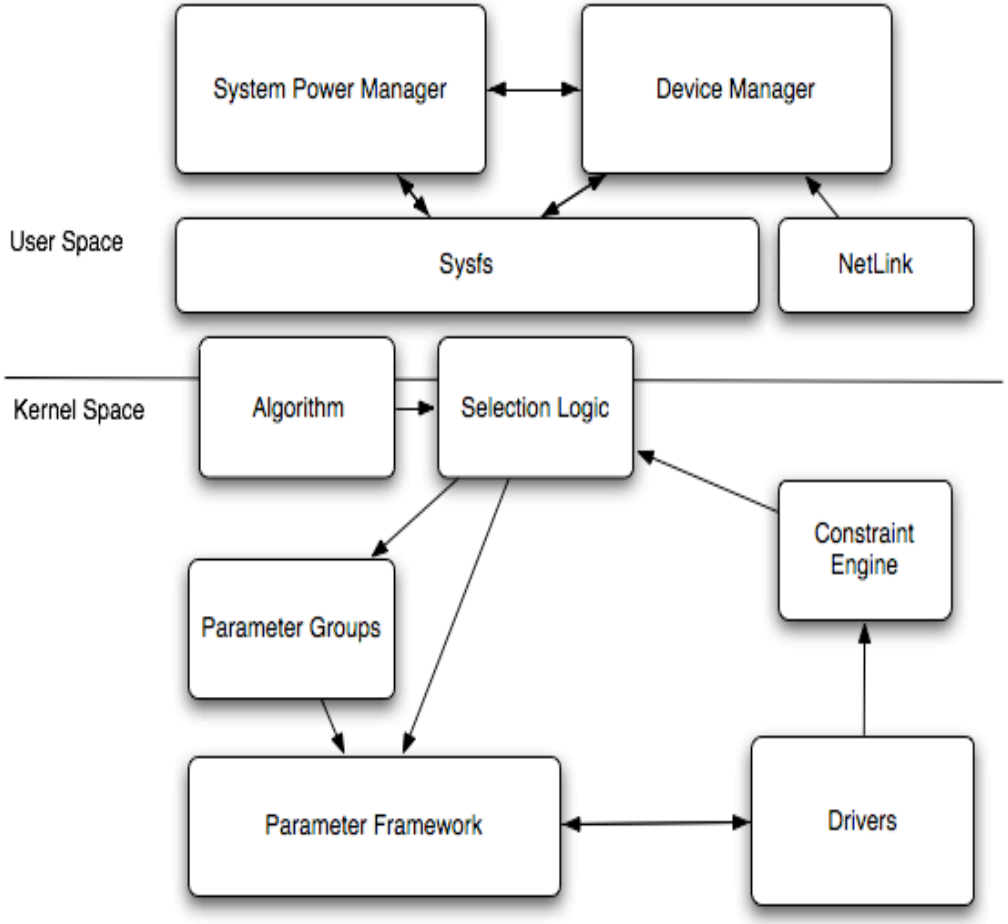


- Follow existing clock framework behavior and API as much as possible
- Modular - allow separate board and SoC definition of parameters. Runtime registration of parameter.
- Keep system operational

Parameter Framework

- Parameter framework provides individual control over power parameters.
- Tracks use count
- Captures generic relationships between h/w resources
- Provides notifications.

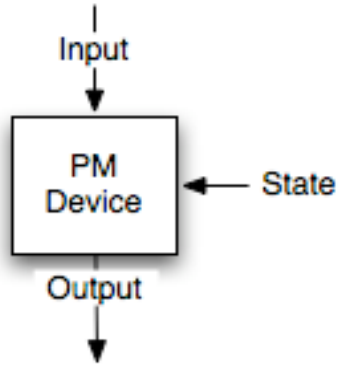
- Parameter Group allows s/w to set parameters as a group for optimal power/performance balance.
- Also enables capturing platform specific h/w dependencies



Hardware resources

- Hardware resources are abstracted as a PM device
- PM device has input, output, and state.
- Export control over output and state

Control output not configuration of the resource.



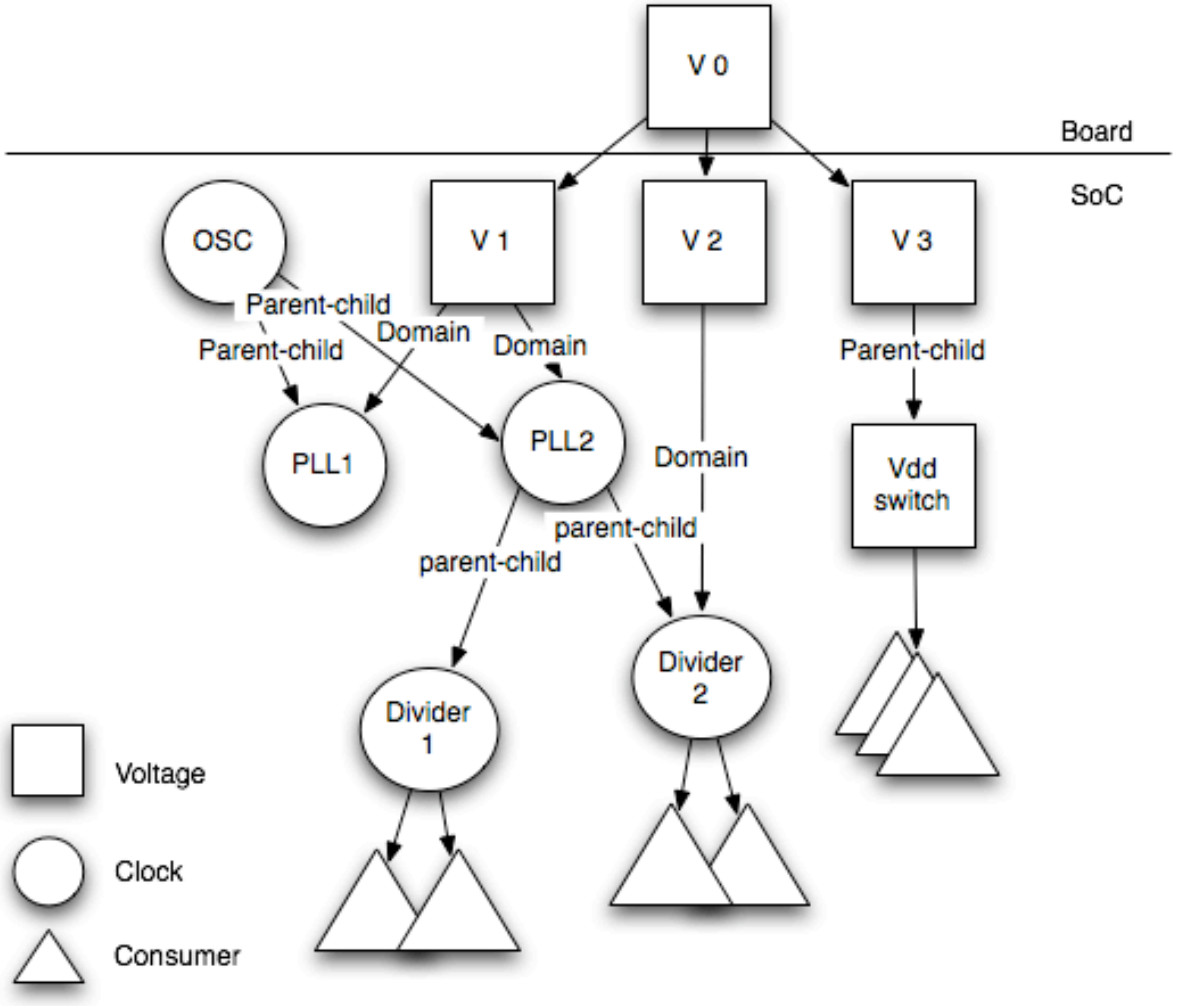
- State allows generic control over pm device when use count is zero. We don't have to special case output values.
- State is platform and resource specific

Track use count and keep system operational

- Must keep track of relationships between parameters
- Define 3 types of relationships:
  - **Domain** is between different types - clk, voltage
  - **Parent-child** is between the same type - pll, clk dividers
  - **Functional** requires "set" method to be coordinated in some way

Example relationship tree

- V1, V2, V3 are voltage domains on a SoC
- V0 is the voltage regulator on the board. It may supply the same voltage to all the domains or supply separate sources.



PM structures

- struct pm_device_ops - a pm provider driver methods
  - **init**: initialize pm device
  - **set**: set new output value
  - **round**: round a given value to hardware supported value
  - **set_state**: state that is used when ref count is zero
  - **recalc**: determine new output value given parent value
- struct pm_device
  - **ops:** pm provider driver methods
  - **parent/child**: track parent and children
  - **master/slave**: track domains
  - **consumers**: subscribed to the pm provider.
  - **target_value**: output value set when node is enabled
  - **state**: power state set when use count is zero
  - **usecount**: tracks if devices is in use or not

API

- **pm_dev_get** - get handle to a pm device
- **pm_dev_put** - release handle

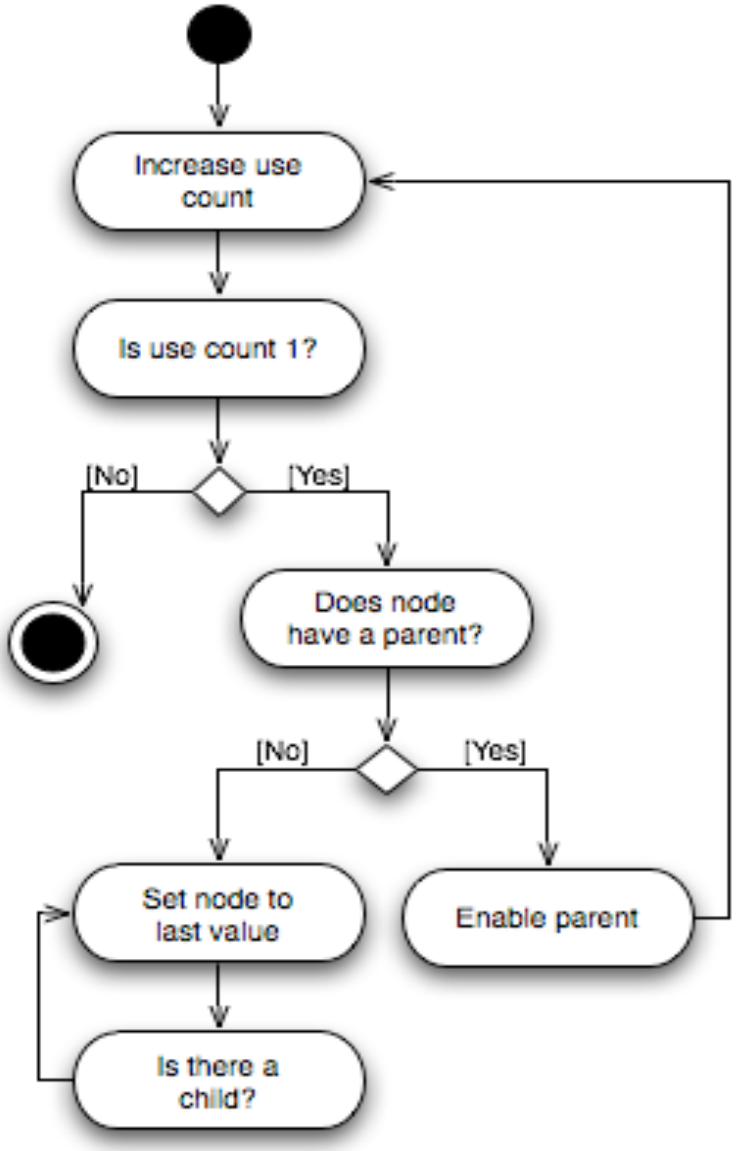
- **pm_dev_enable** - tell pm device to become active and increase use count.
- **pm_dev_disable** - decrease use count and set state

- **pm_dev_set** - set output of pm device
- **pm_dev_get_value**

- **pm_dev_set_state** - set the state that pm device should enter at zero use count.
- **pm_dev_get_state**

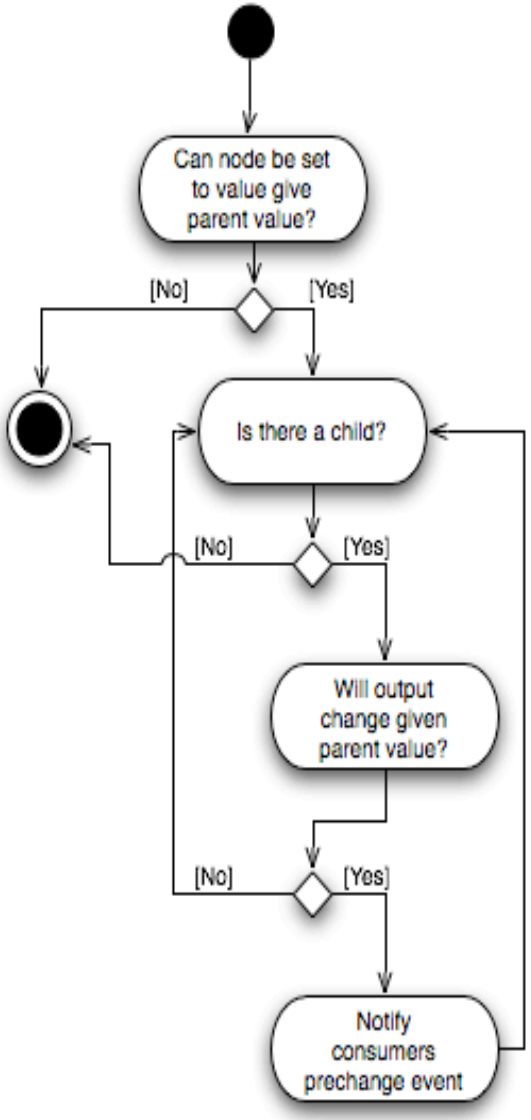
Enable node activity

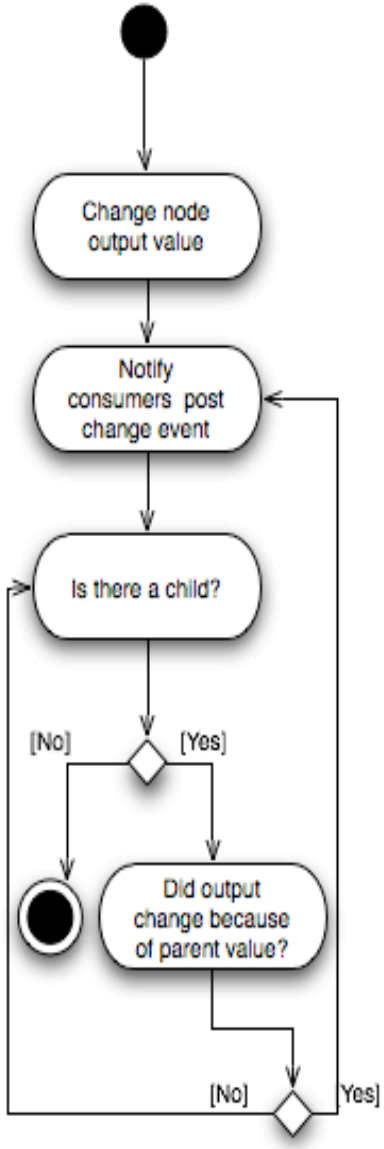
- Enable on a node triggers framework to walk up the tree and enables parents/masters.
- Starting from top set enabled node to last value passed into set method.
- Stop when reach top or an enabled node.



Change node output activity

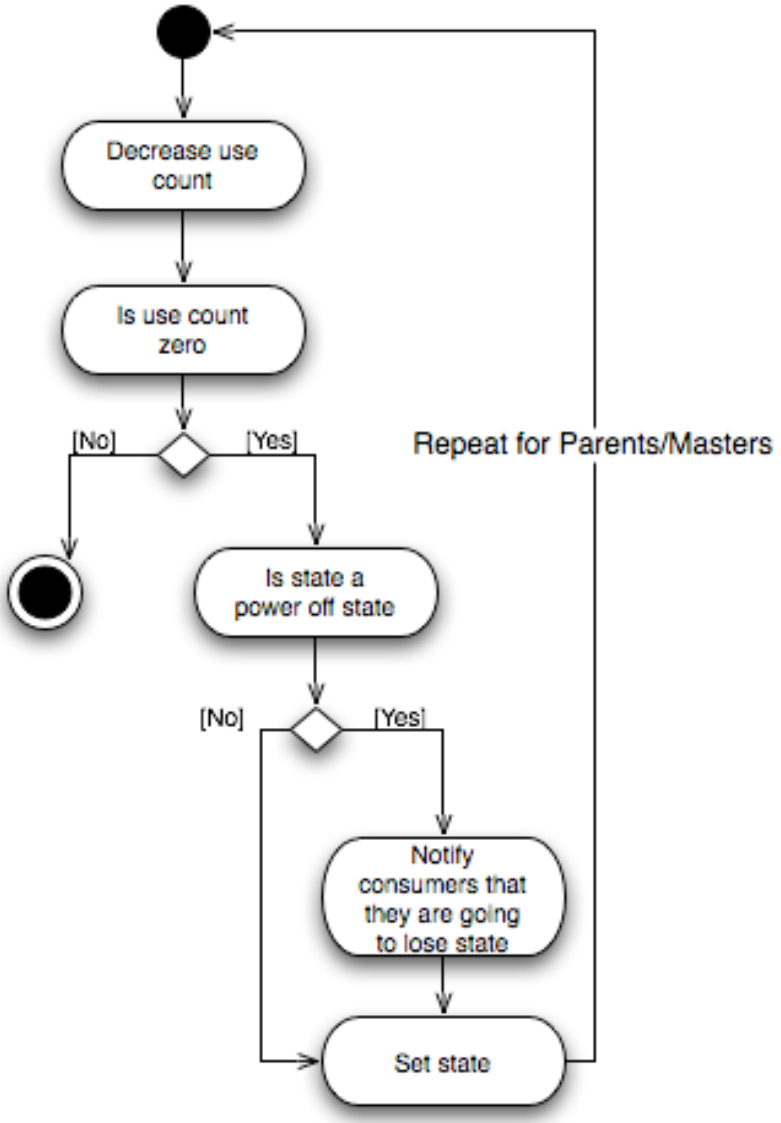
- Changing a node output triggers framework to tell children to recalc.
- Children either change configuration to stay at same output value or configuration stays the same and output value changes.
- If a change occurs notification is sent out to consumers of the pm_dev.



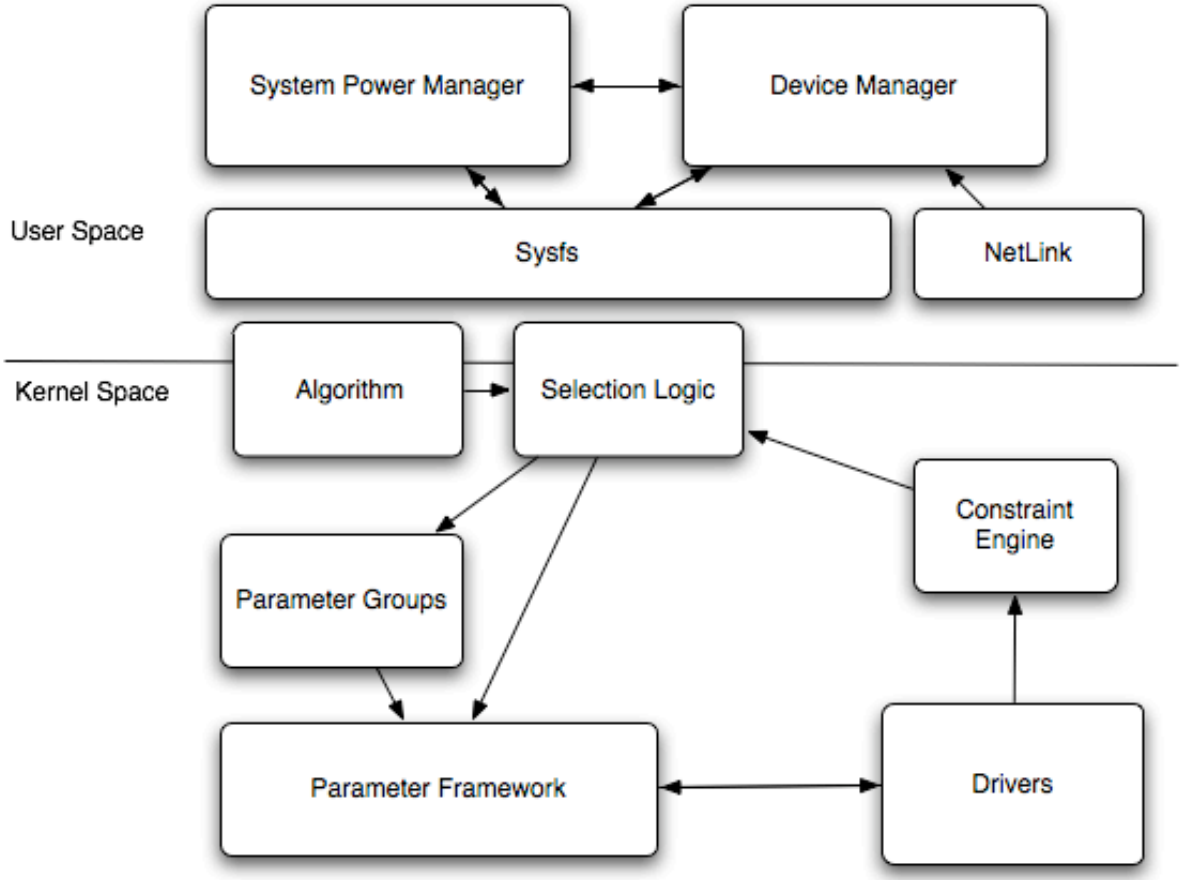


Disable node activity

- Disable checks use count. If zero call set_state.
- If state causes pm device to lose power, notify consumers.
- Repeat for parents and masters.



PM stack


System Power Manager
Device Manager
User Space
Sysfs
NetLink
Kernel Space
Algorithm
Selection Logic
Constraint Engine
Parameter Groups
Parameter Framework
Drivers

Use Cases

- Select lower power states when pm device use counts are zero
  - On PXA, voltage domains are controlled by the idle and sleeps states. If voltage domain use counts are zero, a lower idle or even sleep state can be selected in idle loop
- Selection logic (governor or equivliant) can change output of a shared pm device.
  - Framework ensures that all children of the device are adjusted and consumers are notified
- Device drivers can control local pm devices (not shared)
- Parameter group can collect arbitrary pm devices into groups and set the group using the parameter framework API. (Platform independent)

Issues

- Separate frameworks for voltage and clocks?
  - Typed interface
- Recently submitted voltage framework has different behavior
- Is there enough justification to track relationships between clocks and voltages
  - What we can do depends on the hardware.
- Current clock framework is interface only. Does it make sense to move the code common among platforms to be generic?

NomadGS

Matthew Locke – CTO

408-386-1482