Advances in Power Management in the 2.6 kernel

The Aims of Power Management

- Maximise Battery Life.
- Minimise Impact On Performance.
- Speedy Return To Usable State Post-Suspend.
- Flexible - Cope With Variety Of Capabilities, Policies etc.
- Unified – Include All Buses & Interactions Between Them.
- Reliable Operation In All Modes And Transitions.
- Transparency To Userspace (so far as possible).

The Challenges of Power Management

- Needs to work with an extremely wide variety of hardware.
  - Platforms: Desktop / Laptop / Embedded
  - Architectures: PPC/Intel|AMD/Sparc...
  - Buses: Firewire/USB/IDE/SCSI/PS2/PCMCIA

The Challenges of Power Management

- Needs to work with an variety of configurations:
  - Preemption
  - SMP
  - Highmem

The Challenges of Power Management

- Ambiguous Specifications
- Imperfect And Incomplete Conformity In Hardware
- Our Aims Are At Least Potentially Contradictory
- Interaction Between Subsystems (Hot[un]plug!)
- Unpredictability Of User Initiated Actions

Past Implementations

- 2.4 Kernel:
  - No Driver Model
  - Power Management Notifier Chain
  - No OS Based S1 or S3 Support
  - Out Of Mainline Suspend2 (STD) Implementation

    Limited Arch Support (x86, PPC, ARM)
  - Out Of Tree CPU Frequency Management Support

The Current State Of Play

- Many PM Related Features Introduced In 2.6 Kernel
  - ACPI Support Actively Being Improved
  - Driver Model Introduced And Being Developed (Focus Currently Primarily On System State Management)
  - Hotplug Support Added & Further Development In Works
  - CPU Frequency Support Improved
  - Mainline Support For S1, S3, S4.
  - Working Toward Merging Improved S4 Support

Current Hurdles To Be Overcome

- Driver Support For Power Management An Ongoing Issue
  - USB
  - DRM/DRI
  - Arch != x86/x86_64/PPC-32/PPC-64.
  - Video Support In General For S3 Resume
  - Runtime Power Management Needs Lots Of Work
  - Lack Of Polished User Interface & Apps For Configuring,
    And Diagnosing Issues.
  - Communication Between Developers

What Can You Do To Help?

- Implement Driver Model Support In Your Drivers
- Submit It For Mainline Inclusion Where Appropriate
- Loan/Provide Hardware And Docs For Porting & Testing (I - at least – will submit myself to NDAs)
- Get Involved In Mainline Development Discussions So We Consider Your Needs
- Help With User Interface Configuration Apps

```
#ifdef CONFIG_PM
static int e100_suspend(struct pci_dev *pdev, u32 state)
{
	struct net_device *netdev = pci_get_drvdata(pdev);
	struct nic *nic = netdev_priv(netdev);


	if(netif_running(netdev))
		e100_down(nic);
	e100_hw_reset(nic);
	netif_device_detach(netdev);


	pci_save_state(pdev);
	pci_enable_wake(pdev, state, nic->flags & (wol_magic | e100_asf(nic)));
	pci_disable_device(pdev);
	pci_set_power_state(pdev, state);


	return 0;
}
```

```
static int e100_resume(struct pci_dev *pdev)
{
	struct net_device *netdev = pci_get_drvdata(pdev);
	struct nic *nic = netdev_priv(netdev);


	pci_set_power_state(pdev, 0);
	pci_restore_state(pdev);
	e100_hw_init(nic);


	netif_device_attach(netdev);
	if(netif_running(netdev))
		e100_up(nic);


	return 0;
}

#endif  /* CONFIG_PM */
```