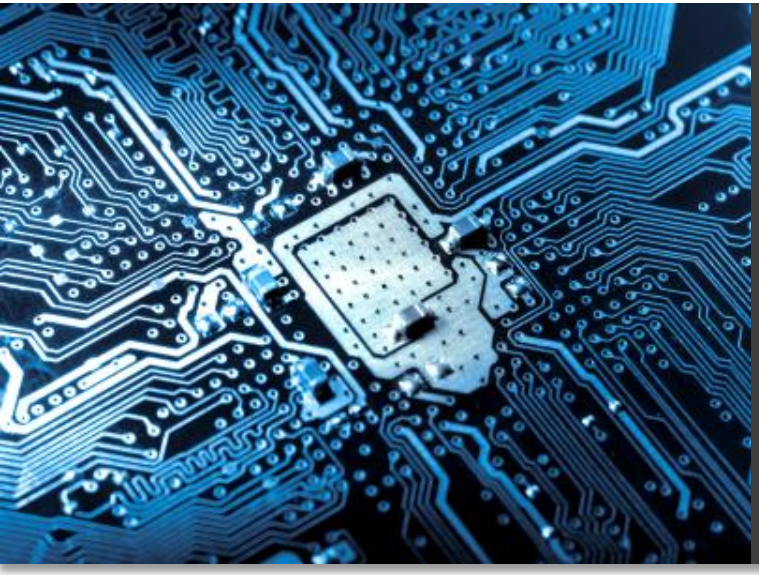
EM Photonics
Accelerated Computing Solutions



Power-Aware Software on ARM

Paul Fox

Outline

Motivation

Linux Power Management Interfaces

A Unified Power Management System

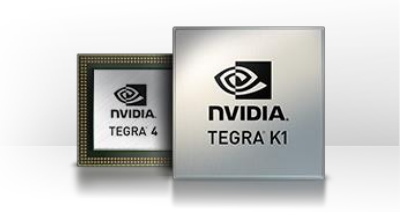
Experimental Results and Future Work

EM Photonics
Accelerated Computing Solutions

MOTIVATION

MOTIVATION

- ARM SoCs designed for phones and tablets are increasingly being used for embedded applications in other areas.

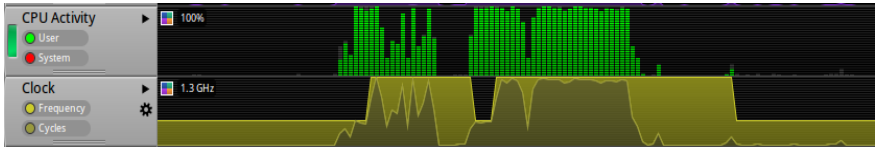


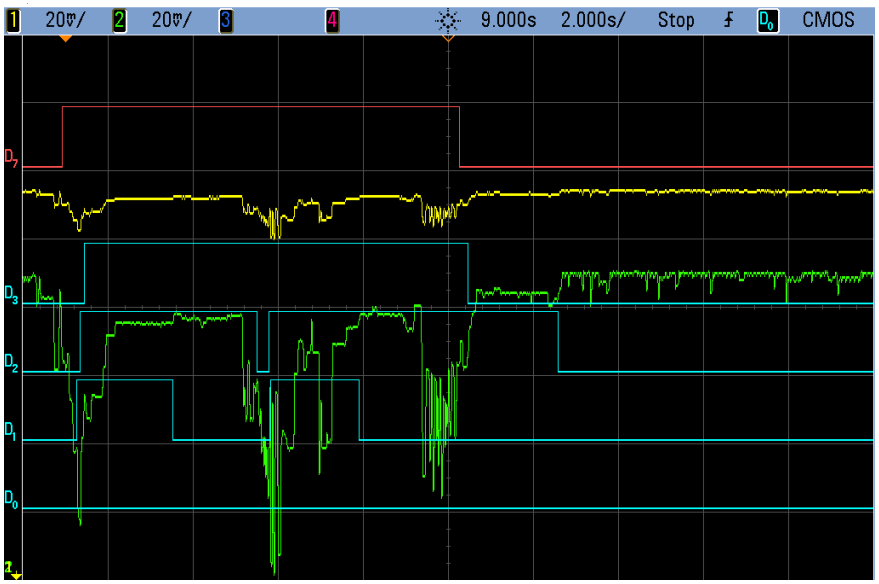
- These devices are typically designed to conserve power by sleeping between bursts of activity.

- This strategy causes problems for applications that need to run continuously for long periods of time on battery power (e.g. computer vision)

Problems

- Multiple power-management systems in the Linux kernel do not have close integration.
- These systems use principally *reactive* techniques, which rely on estimates of future demand.
- Bad estimates and interference between systems can reduce efficiency.
- They cannot be easily controlled by the application programmer.
  - Difficult to control power consumption
  - Difficult to utilize full potential of thermally-constrained systems.





EM Photonics

Goals for Low Power Development

Straightforward Program Development

Streamline the creation and maintenance of software targeted at low power devices

Platform Independent

A single program should work on multiple devices without the need to rewrite using platform-specific tools

Options for Manual and Automated Optimization

Allow developers the flexibility to make power optimization decisions

EM Photonics
Accelerated Computing Solutions

Linux Power Management Interfaces

The Operating System

- The Linux Kernel provides interfaces through sysfs to several power-saving systems.
  - `cpufreq`
  - `cpu hotplug`
- Several other power-management systems lack userspace control
  - `cpuidle`
  - `cpu cluster switching`

CPUFREQ

- Controls CPU frequency and voltage dynamically.
- Frequency can be set by the programmer through the `sysfs` kernel interface.
- Generally mature and works well, but offers limited power savings on modern ARM SoCs.

CPU HOTPLUG

- This kernel subsystem allows CPUs to be taken offline, gating power to the core and moving all threads to other cores.
- This can save a lot of power by preventing regular wakeups.
- However, it currently has not been optimized for performance, and therefore has a high overhead.
- Work is being done to improve the performance.[1]

[1] Cleaning Up Linux's CPU Hotplug For Real Time and Energy Management

CPUIDLE

- This kernel subsystem allows the kernel to place the CPU in a sleep state while it is idle.
- Deeper sleep states are selected if the governor *estimates* that the CPU will be idle for a long time.
- Currently no userspace control, but it should be possible to add a userspace governor similar to cpufreq.

CPU Cluster Switching

- Many ARM SoCs, including recent Tegra parts, have a low-power cluster, with one or more CPUs that are designed for low power consumption instead of performance.
- The kernel can decide when to switch to running on this low-power cluster, dramatically reducing power consumption.
- Support for this in the kernel is still immature and there is no user-space control.

EM Photonics
Accelerated Computing Solutions

A Unified Power Management System

TYING IT TOGETHER WITH POWERMP™

- Lots of different interfaces for power management with little integration.
- Want an easy way to permit application power management.
- Precedent: OpenMP
  - Steers parallelism in code
- **PowerMP**
  - Steer power usage in code

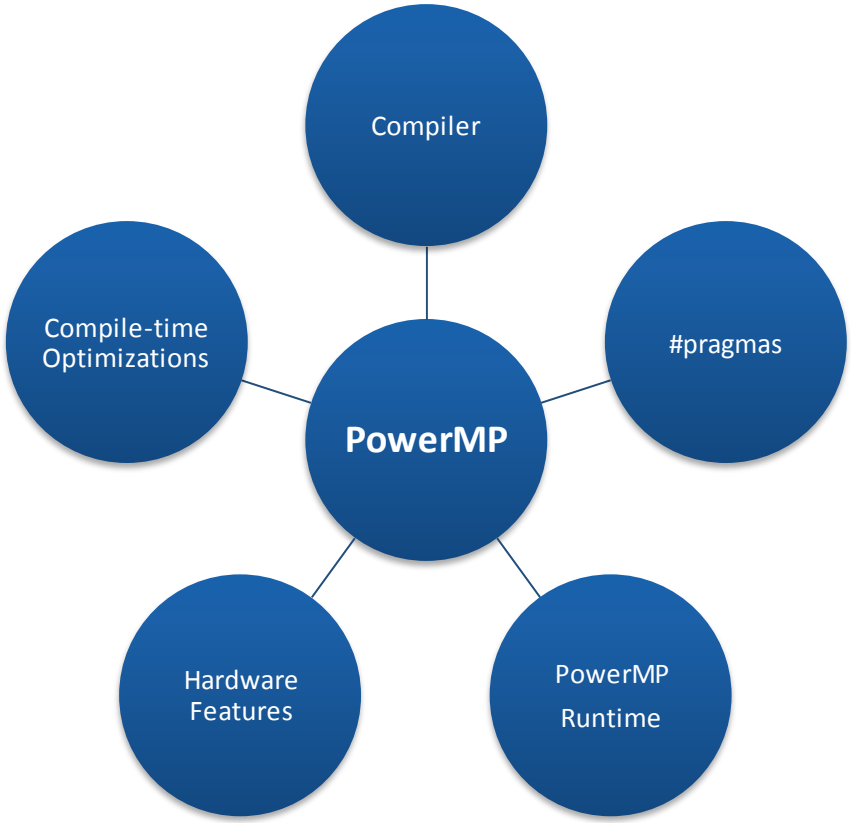
OpenMP

```
#pragma omp parallel for
for (int i = 0; i < 16; ++i)
    f(i);
```

PowerMP

```
event_t e;
#pragma pmp lowpower
while(wait_for_events(e)) {}

#pragma pmp performance
handle_event(e);
```

PowerMP Architecture



Implementation similar to OpenMP

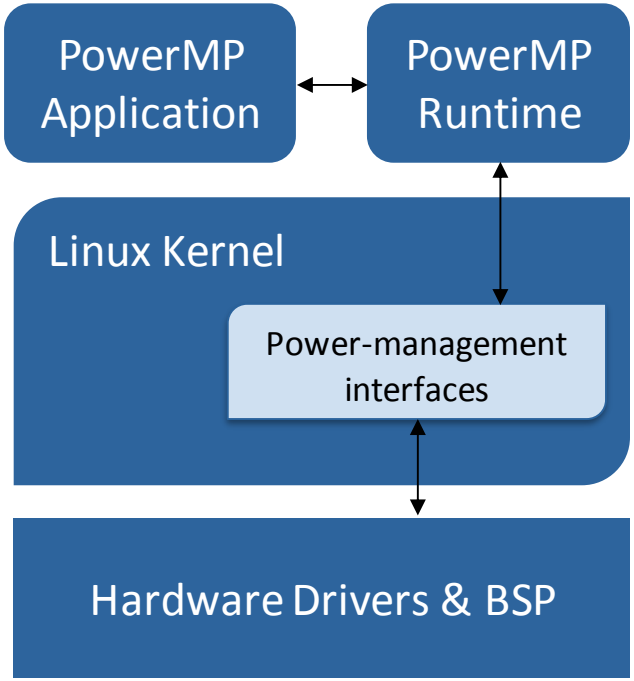
- » Touches parts of the existing tool-chain
- » **#pragmas** are used to provide design intent
- » **Compiler** auto-instruments the code with **runtime library** calls and local/global **compile-time optimizations**
- » **Hardware features** are directly exercised by the **runtime library** and indirectly by **optimizations**
- » Extensible to new and existing codes and platforms

EM Photonics

INTERACTING WITH POWERMP

Compiler-level tool that allows developers to easily write low-power applications

- Developer uses pragma-based directives to create PowerMP application
  - Directives are ignored on unsupported systems.
- Platform: runtime + a supported kernel
  - An open specification & reference implementation
  - Implementation heavily leverages existing kernel features.
  - Kernel support for additional sysfs (userspace) interfaces
- Benefit: Supporting a new hardware device does not require any application changes.



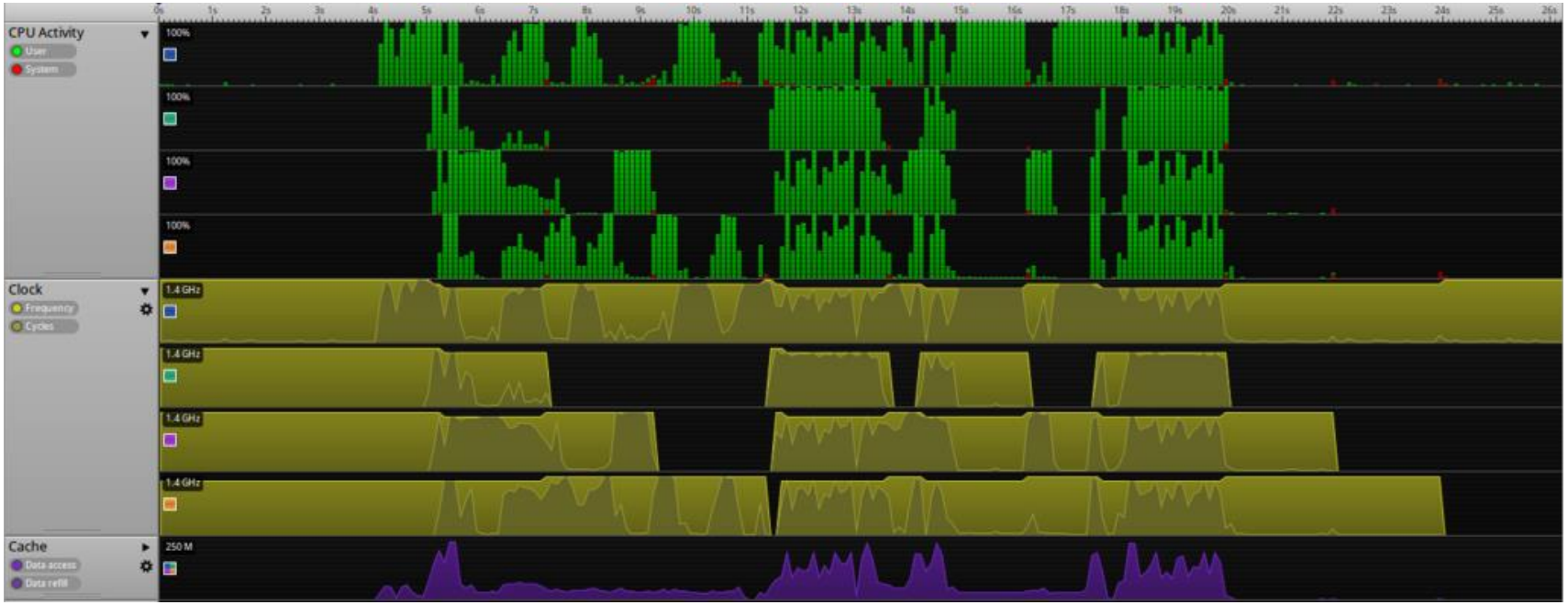
EM Photonics

EXAMPLE CODE

```
»    nPxlsRead = fread(image, nBytesPerPxl, nPxls, fp);

»    {
»        #pragma omp parallel for
»            for (i = 0; i < nPxlsRead; i++)
»            { … }
»    }
```

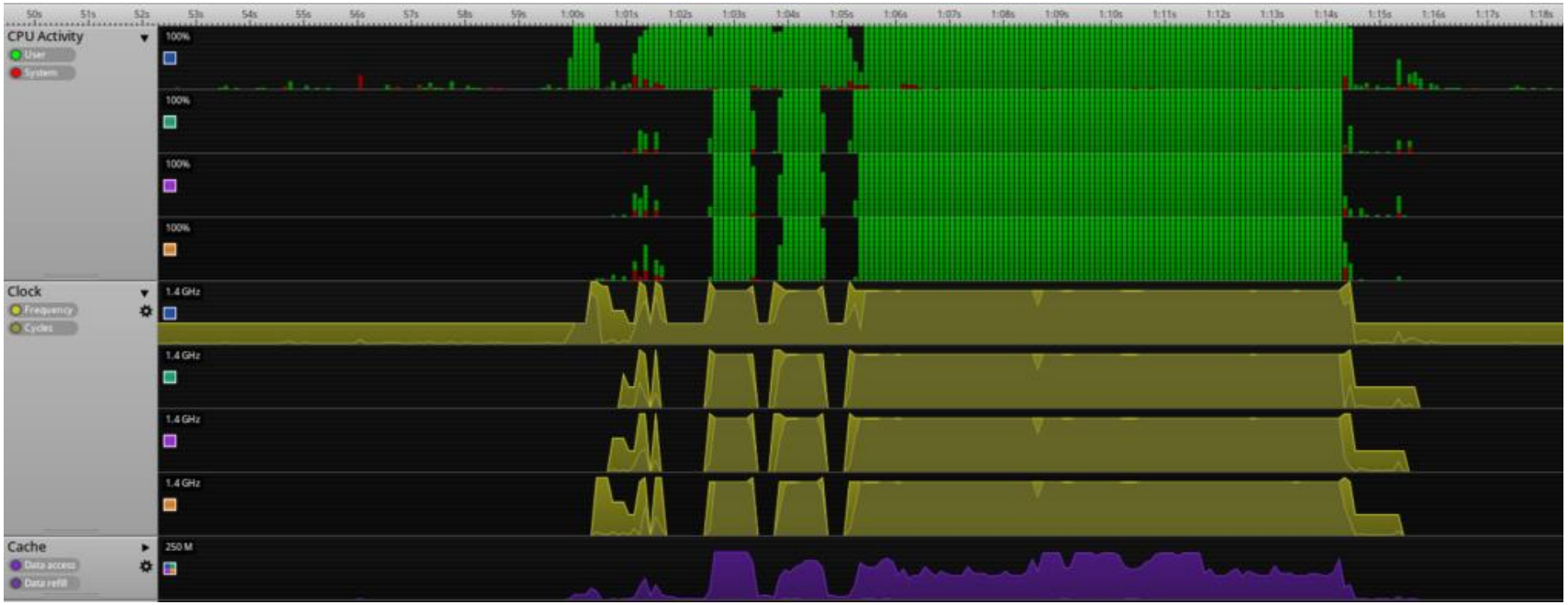
```
»    #pragma pmp lowpower
»    nPxlsRead = fread(image, nBytesPerPxl, nPxls, fp);

»    #pragma pmp performance
»    {
»        #pragma omp parallel for
»            for (i = 0; i < nPxlsRead; i++)
»            { … }
»    }
```

BEFORE



EM Photonics

After

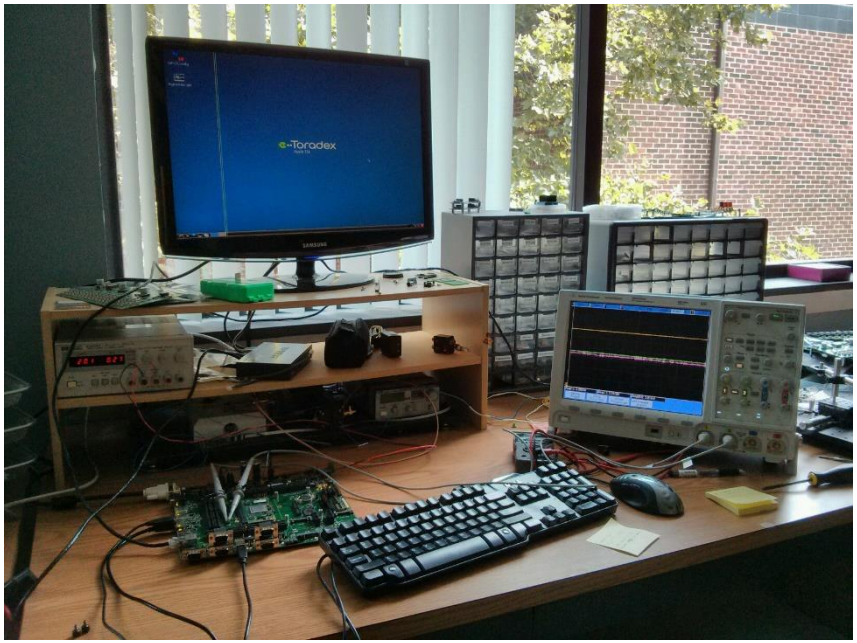


EM Photonics

EM Photonics
Accelerated Computing Solutions

Experimental Results

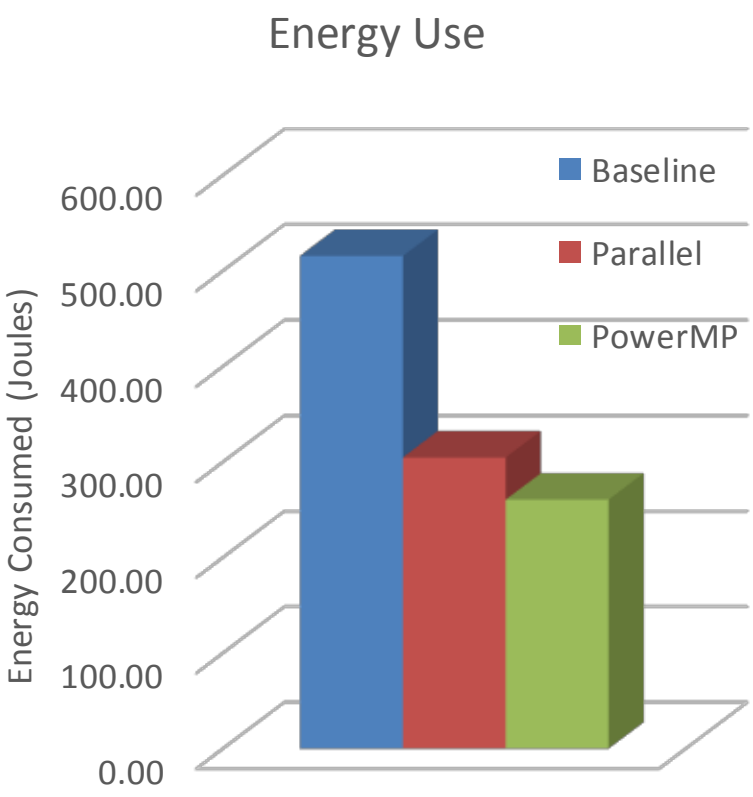
EXPERIMENTAL SETUP



- Toradex Apalis T30 (Tegra 3) board modified with resistors to measure current to SoC.
- Kernel customized to set GPIO pins indicating CPU state.
- Oscilloscope for measurement.
- Used hotplug and cpufreq interfaces in userspace.

MEASURED POWER SAVINGS

- Adding parallelism vastly reduces energy consumption for a given task.
- PowerMP instructed version improves energy consumption by 10%.
- Currently limited to coarse-grained manipulation because of CPU hotplug limitations.



EM Photonics

EM Photonics
Accelerated Computing Solutions

Future Work

IMPROVING THE TOOLCHAIN

- Power-aware PGO for automated empirical power optimization.
- Can be used to intelligently schedule power-state switching.
- Can be used to achieve a desired performance or power-consumption target while optimizing the other parameter.
- Instrumentation can give developers a better idea of how their application uses power.

Opportunities of New Hardware: GPUs

- New hardware, such as the Tegra K1, is starting to have GPUs suitable for general purpose computing.
- This offers the potential for greater efficiency gains for suitable algorithms.
- The rest of the system can be forced into a low power state while waiting on GPU tasks to finish.
- This will require kernel support to expose this functionality.

Improving Kernel Support

- Better kernel support is needed for further improvements.
  - CPUIdle userspace
  - Cluster switching control in userspace
  - NO_HZ_FULL to reduce kernel scheduling wakeups (potentially cooperative multitasking).
  - cgroups integrated with OpenMP to reserve CPUs for high-performance tasks, facilitating faster power-state switching.

PowerMP Runtime Improvements

- Enable fine-grained control of power states.
- Enable dynamic decision-making based on supplied policies.
  - Real-time applications
  - Hard or soft power consumption contraints
  - Respond to changes in system state (e.g. low battery).
  - Mediate between multiple processes (PowerMP daemon)

CONCLUSION

- Maximal power-savings require a “full-stack” awareness of power.
- Kernel support is necessary, but not sufficient.
- User-space interfaces to the drivers can be used to improve power consumption, but they aren’t easy to use.
- An OpenMP-like runtime can be used to bridge the gap.

THANK YOU

Questions?

EM Photonics