



Energy Efficient Computing from Embedded to Server Scale

Digvijay Singh

ASCENT Lab, UCLA

Advisor: Prof. William Kaiser

Outline

- **Introduction**
- **Deep Energy Inspection**
 - Energy Endoscope
- **Per-process Energy Accounting** for single and multi-core Platforms
 - Etop
 - EnergyView
- **Energy-efficient Solutions** enabled by Deep Energy Inspection
 - SmartSync
- **Concluding Remarks**

Introduction: The Team

- Thanos Stathopoulos, ASCENT Lab
- Dustin McIntire, ASCENT Lab
- Maxim Batalin, ASCENT Lab
- Digvijay Singh, ASCENT Lab
- Sebi Ryffel, ETH Zurich

Introduction: The Problem of Energy Consumption

- **Energy Cost**
 - Second only to manpower costs in datacenters (Gartner Group)
- **Critical for mobile and embedded devices...**
 - Continuous quest to extend battery life
 - Scarcity of energy sources in outdoors & remote locations
- **...and for computing in general**
 - Data Centers: 1.2% of all electrical power usage in US (2005)
 - 4.5B USD (2007), **doubling** in 3 years
- **US Department of Energy:**
 - “Need for more metrics and data.”
 - “How do we account for computing performance?”



DOE

EPA

Deep Energy Inspection

- **Introduction**
- **Deep Energy Inspection**
 - Energy Endoscope
- **Per-process Energy Accounting** for single and multi-core Platforms
 - Etop
 - EnergyView
- **Energy-efficient Solutions** enabled by Deep Energy Inspection
 - SmartSync
- **Concluding Remarks**

Deep Energy Inspection for Embedded Platforms: Energy Endoscope

- Where is energy dissipated in computing platforms?
 - Answer needed at *runtime* to capture system dynamics
- Need for Deep Energy Inspection
 - Enables detailed energy visibility
 - Individual hardware components
 - System-wide
 - Software entities (tasks, processes, activities)
- Deep Energy Inspection Requirements
 - High Resolution
 - Low Overhead
 - Integration
- Discover, understand and mitigate energy inefficiency

Joint work by Thanos Stathopoulos and Dustin McIntire

Energy Endoscope: LEAP2 Hardware

Overview



Processing Module



EMAP2 + peripherals



EMAP2 ASIC

- **PXA270-based processing module**
 - 64 MB of SDRAM, 8MB of PSRAM
 - 32 MB NOR flash, 512MB NAND flash
 - Ethernet and 802.11 interfaces
- **EMAP2 ASIC performs energy accounting functions**
 - Less than 6mW power dissipation
 - 1ms measurement accuracy
 - Charge accumulation output
 - **Per-subsystem information:** up to 48 channels
 - **Data readback from host CPU:** less than 8usec

Power range:
25-1500mW

Conventional off-the-shelf hardware enabling low cost, good resolution and integration

Energy Endoscope: Software Architecture

- **Modular, kernel-resident software architecture**

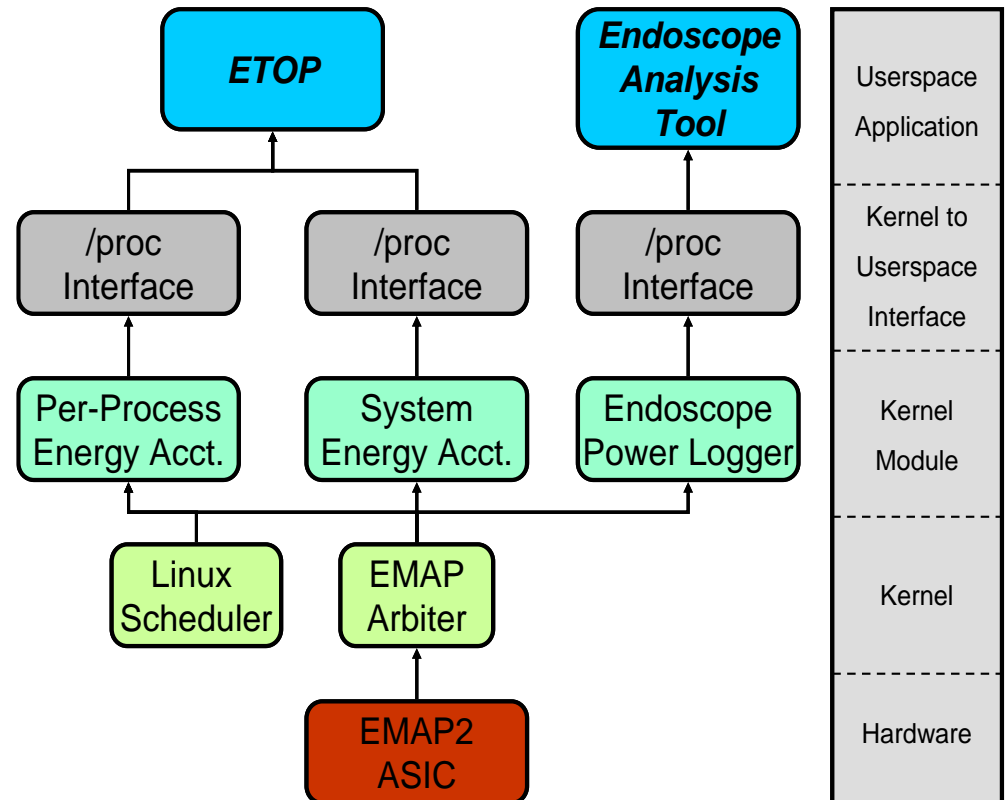
- High time resolution energy measurements enabled by ASIC
- EMAP arbiter operating system module enables resource efficient sampling
- Arbiter clients transfer data to user space

- **Endoscope**

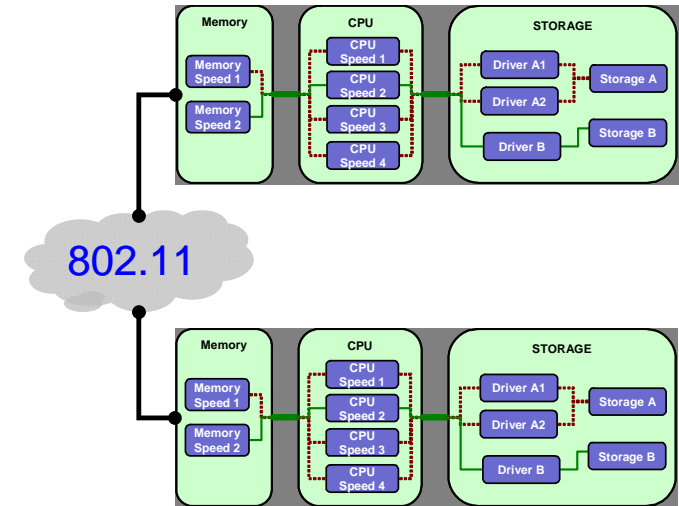
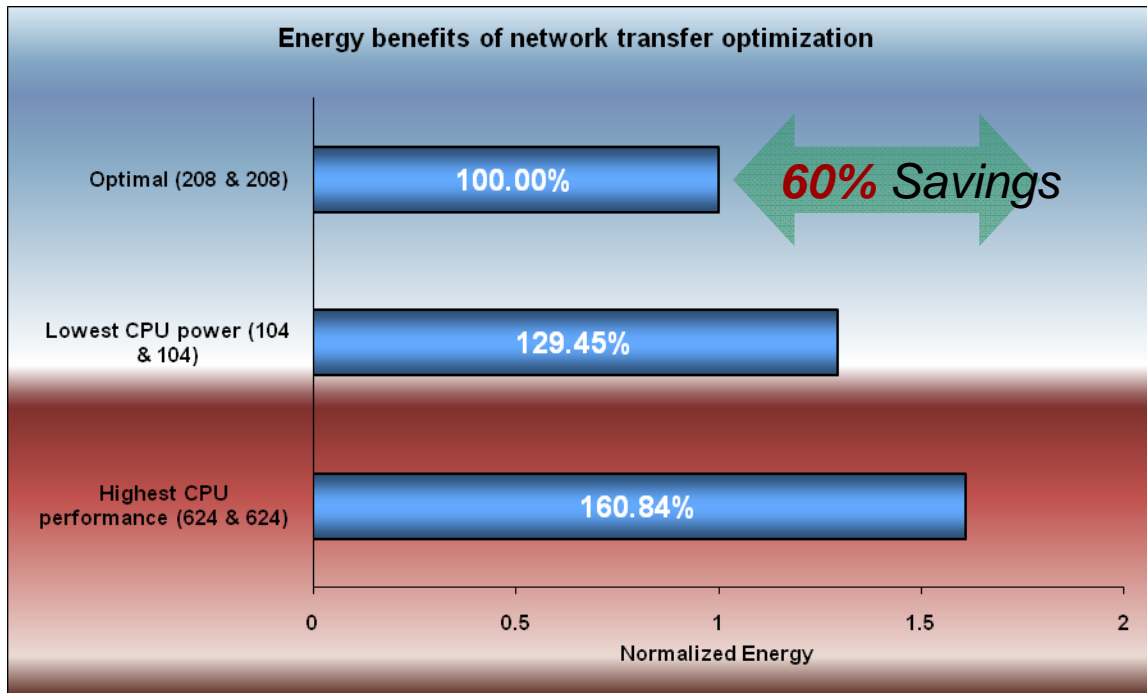
- Low-overhead design & prototype energy measurement tool

- **Etop**

- Real-time per-process energy consumption display tool
- Based on the “top” utility



Energy Endoscope: Example



- Problem of efficient network data transfer
- Use endoscope's **deep energy inspection**
 - Pick revealed optimal operating points
- Upto **60% energy savings**

Per-process Energy Accounting

- **Introduction**
- **Deep Energy Inspection**
 - Energy Endoscope
- **Per-process Energy Accounting** for single and multi-core Platforms
 - Etop
 - EnergyView
- **Energy-efficient Solutions** enabled by Deep Energy Inspection
 - SmartSync
- **Concluding Remarks**

Why Per-process Energy Accounting?

- **Accountability**
 - Energy is a system resource and should be tracked similarly to CPU, Memory and bandwidth
- **Comparison**
 - Compare energy cost of applications X and Y
 - With equal functionality, which one is more energy efficient?
- **Auditing & charging**
 - Charge applications \$ by energy usage
 - Similar concept to CPU & bandwidth utilization charging in web hosting

Per-process Energy Account for Single-core Platforms: **Etop**

Real-time display of per-subsystem current/power/energy consumption.
System Energy Acct. module

Rapid identification of significant power consumers

Real-time display of **per-process** energy consumption
Per-process Energy Acct. module

Based on well-known "top" Unix program

The screenshot shows the Etop utility interface. At the top, it displays system status: 'etop - 20:46:37 up 20:02, 4 users, load average: 0.75, 0.34, 0.14'. Below this, it lists tasks: 'Tasks: 28 total, 2 running, 26 sleeping, 0 stopped, 0 zombie'. A section titled '16 subsystems: Instantaneous & average values' lists various hardware components with their current power consumption and energy usage. At the bottom, a table shows per-process energy consumption.

16 subsystems: Instantaneous & average values			
Channel	Amps	Power	Energy
00(Ethernet):	116.77 mA	385.34 mW	23.37 J
01(Imager):	nan mA	nan mW	0.00 J
02(Radio):	nan mA	nan mW	0.00 J
03(GPS):	nan mA	nan mW	0.00 J
04(USB):	0.91 mA	3.00 mW	0.18 J
05(CF):	59.01 mA	194.75 mW	11.85 J
06(HPM):	nan mA	nan mW	0.00 J
07(Vin):	5.08 V	0.00 --	0.00 -
08(SDRAM):	25.98 mA	46.76 mW	1.97 J
09(NOR):	0.15 mA	0.28 mW	0.02 J
10(SRAM):	0.90 mA	1.62 mW	0.10 J
11(PXA):	399.10 mA	577.75 mW	21.56 J
12(NAND):	2.65 mA	8.75 mW	0.20 J
13(Vmem):	1.81 V	0.00 --	0.00 -
14(Vcore):	1.44 V	0.00 --	0.00 -
15(Vio):	3.00 V	0.00 --	0.00 -

PID	USER	PR	NI	S	%CPU	%MEM	TIME+	ENERGY	COMMAND
1160	root	25	0	R	95.6	0.9	0:09.36	6.13(0.80/ 5.33)	md5sum
1156	root	15	0	R	3.8	1.6	0:05.10	4.04(2.45/ 5.9)	etop
178	root	25	10	S	0.0	0.0	0:01.44	1.71(0.00/ 3.71)	jffs2_gcd_mtd3
1104	root	15	0	S	0.9	3.1	0:01.40	1.11(0.29/ 0.82)	sshd
1109	root	15	0	S	0.0	2.1			
1128	root	19	0	S	0.0	2.1			
1085	root	21	0	S	0.0	3.1	0:02.40	1.08(0.29/ 0.79)	sshd
1123	root	15	0	S	0.0	3.1	0:00.69	1.07(0.27/ 0.80)	sshd

ETH: 385 mW

CF: 195 mW

SDRAM: 47 mW

CPU: 578 mW

md5sum: 6.13 Joules total

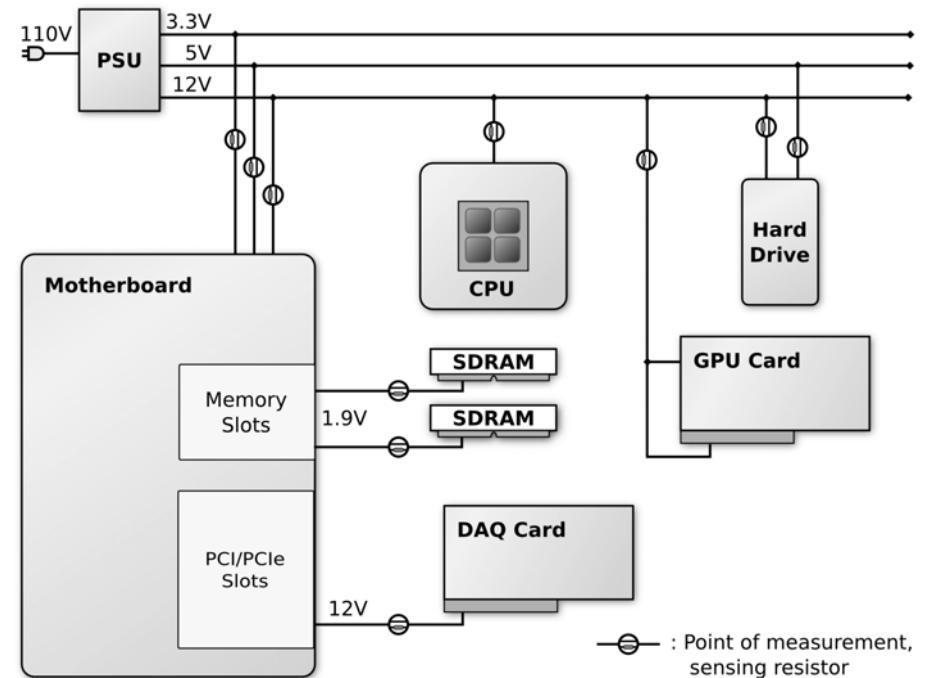
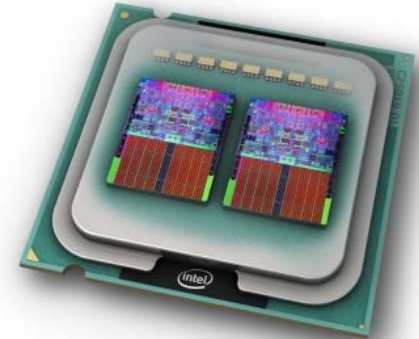
Per-process Energy Accounting in Multi-core Platforms: EnergyView

- Problems
 - **One** energy measurement per hardware resource
 - **Multiple** processes using same hardware resource
- Solution
 - More complicated than single-core solution (Etop)
 - Requires **modeling of process behavior**

Joint work by Thanos Stathopoulos and Sebi Ryffel

EnergyView: Deep Energy Inspection System

- **Modern desktop machine**
 - Intel® Core™2 Quad 2.4GHz
 - 2x4MB L2 cache
 - 4GB DDR2 SDRAM
- **Accurate energy measurements**
 - Sensing resistors
 - Up to 8kHz sampling frequency
 - 16bit resolution
 - Low overhead
- **Per-component resolution**
 - CPU
 - SDRAM DIMMs
 - GPU
 - Planar (12V, 5V, 3.3V)
 - Hard drives (12V, 5V)

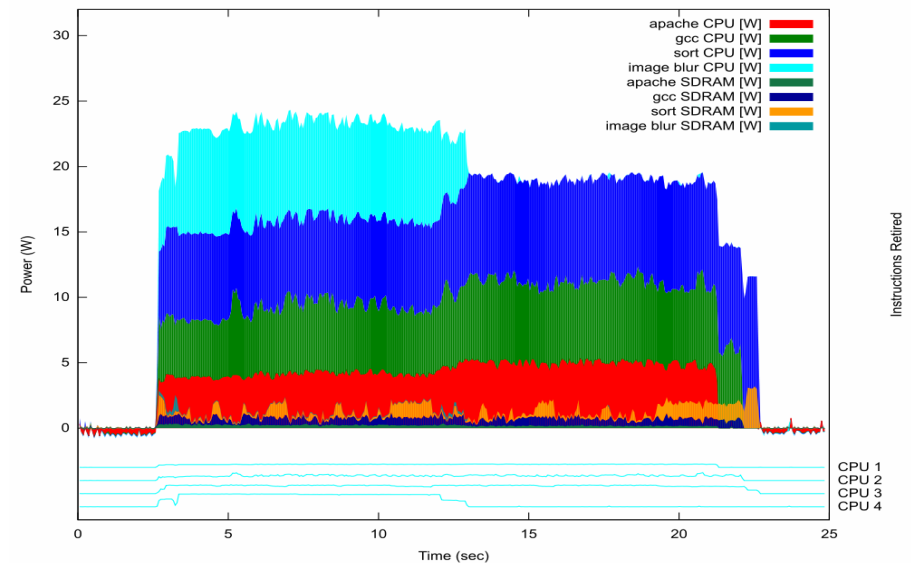
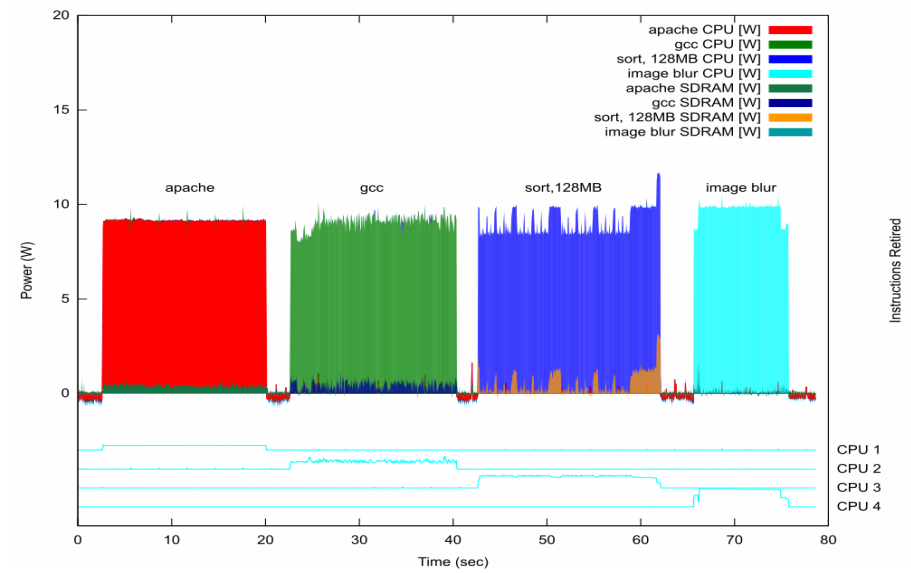


EnergyView

1. Apache
2. gcc
3. Sort
4. Image blur

Sequentially	CPU [J]	SDRAM [J]	Runtime [s]
apache	158.3	1.9	17.5
gcc	159.4	4.0	17.9
sort	174.0	5.1	19.2
blur	96.8	0.5	10.1

Concurrently	CPU [J]	SDRAM [J]	Runtime [s]
apache	83.9	2.0	18.7
gcc	109.7	5.2	19.6
sort	145.9	5.8	20.2
blur	72.7	0.4	10.3

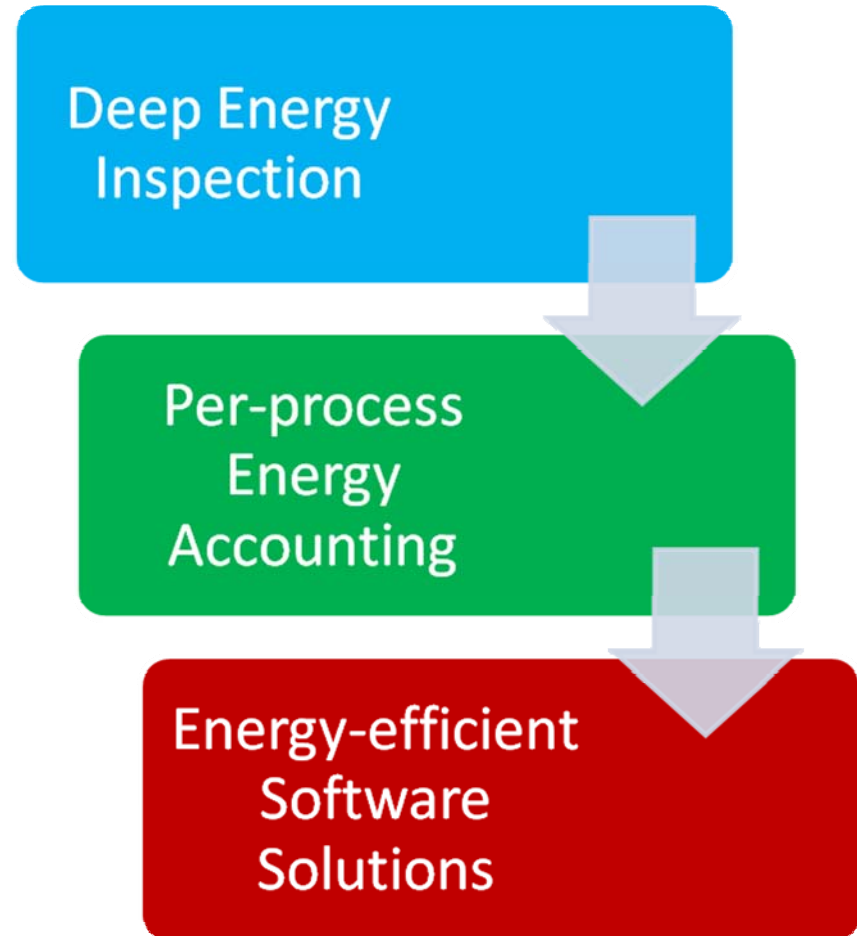


Energy-efficient Solutions

- **Introduction**
- **Deep Energy Inspection**
 - Energy Endoscope
- **Per-process Energy Accounting** for single and multi-core Platforms
 - Etop
 - EnergyView
- **Energy-efficient Solutions** enabled by Deep Energy Inspection
 - SmartSync
- **Concluding Remarks**

Energy-efficient Solutions Enabled by Deep Energy Inspection: SmartSync

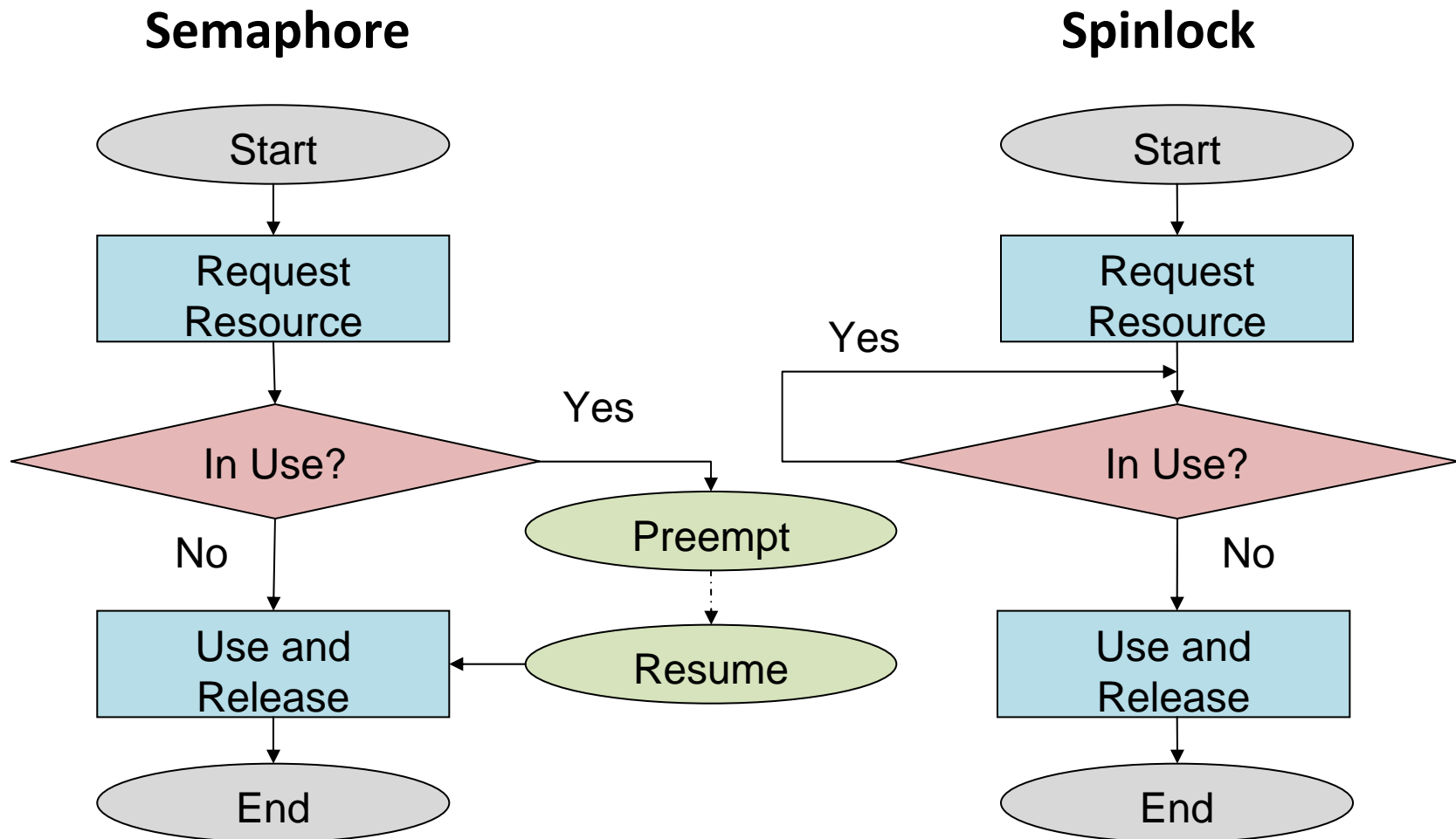
- Deep Energy Inspection and Per-process Energy Accounting
 - Set of tools and applications
 - Measure energy consumption
 - Discover energy inefficiency
- SmartSync
 - Software solution that evolved from the information
 - Target energy bottlenecks
 - Specifically target resource synchronization



SmartSync: Energy-efficient Synchronization in Multi-core Platforms

- Why isn't performance scaling in proportion with number of cores?
 - But energy is!
 - **Resource contention**
- Current resource sharing primitives have considerable overhead
 - Locking is a very common approach
 - ***Runtime information needed*** to “intelligently” lock a resource

SmartSync: Spinlock and Semaphore



SmartSync: Timeout Lock

START

Timer T;

Threshold TH;

Resource_lock L;

while(T < TH)

 Try_to_lock(L);

 Update_timer(T);

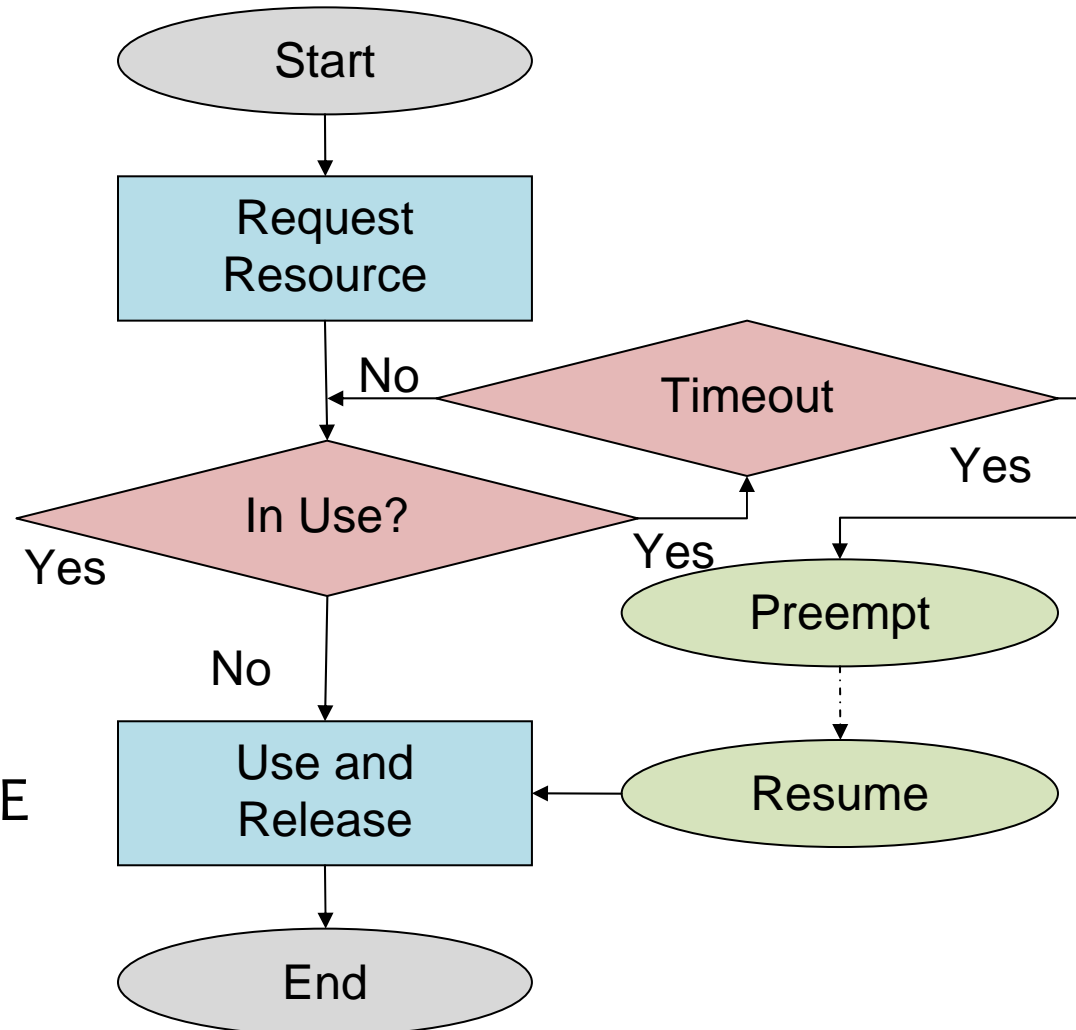
 if(Not_gotten(L))

 Semaphore(L);

 USE LOCKED RESOURCE

 Release_lock(L);

END



SmartSync: Predictive Lock

START

Prev_Times T_1, \dots, T_k ;

Threshold TH;

Predicted_time T;

Resource_lock L;

$T = F(T_1, \dots, T_k)$;

if($T > TH$)

 Semaphore(L);

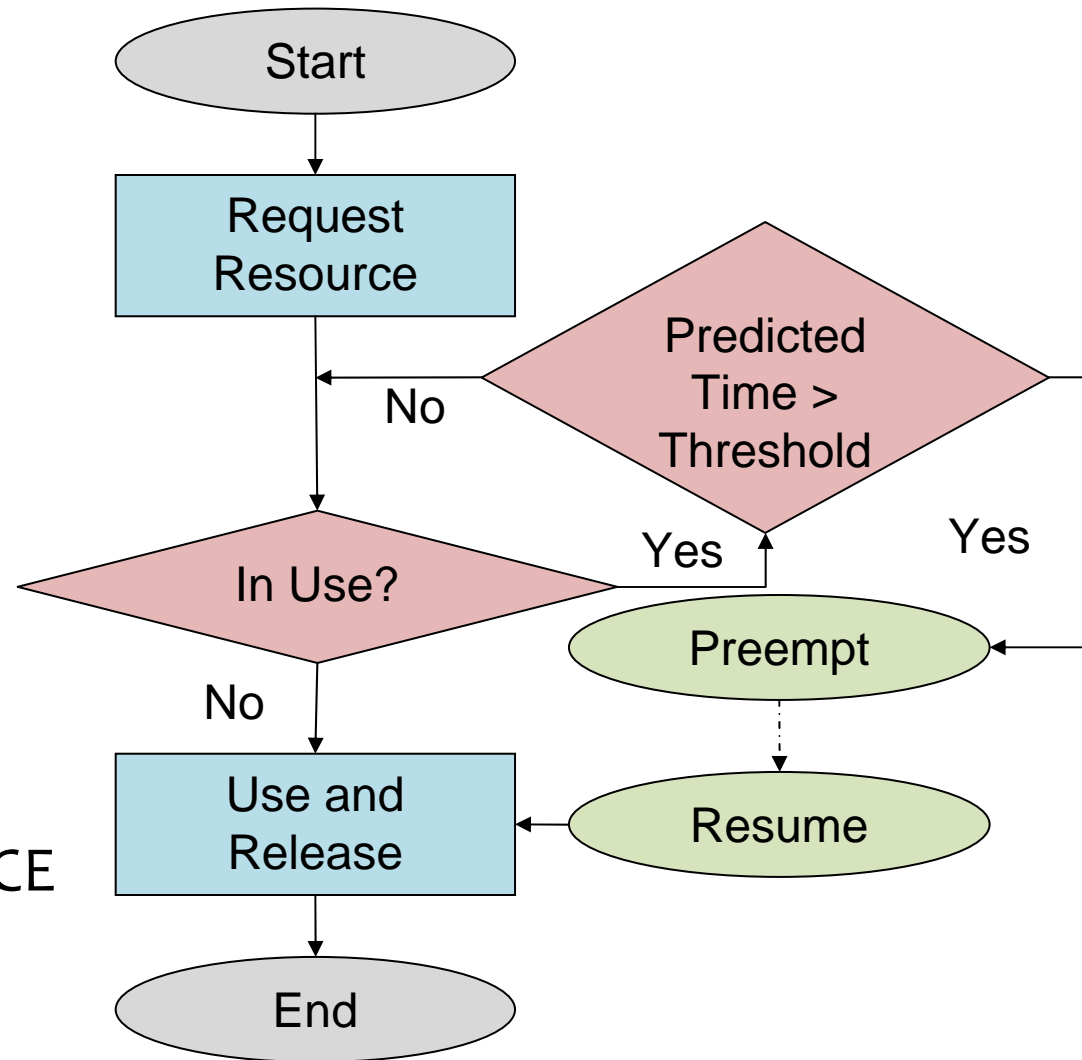
else

 Spinlock(L);

USE LOCKED RESOURCE

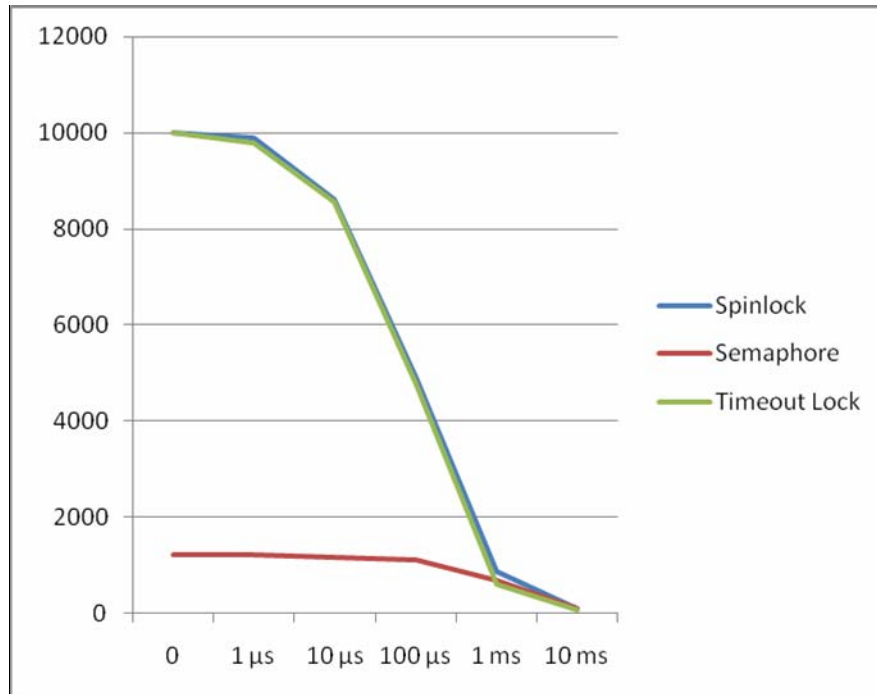
 Release_lock(L);

END

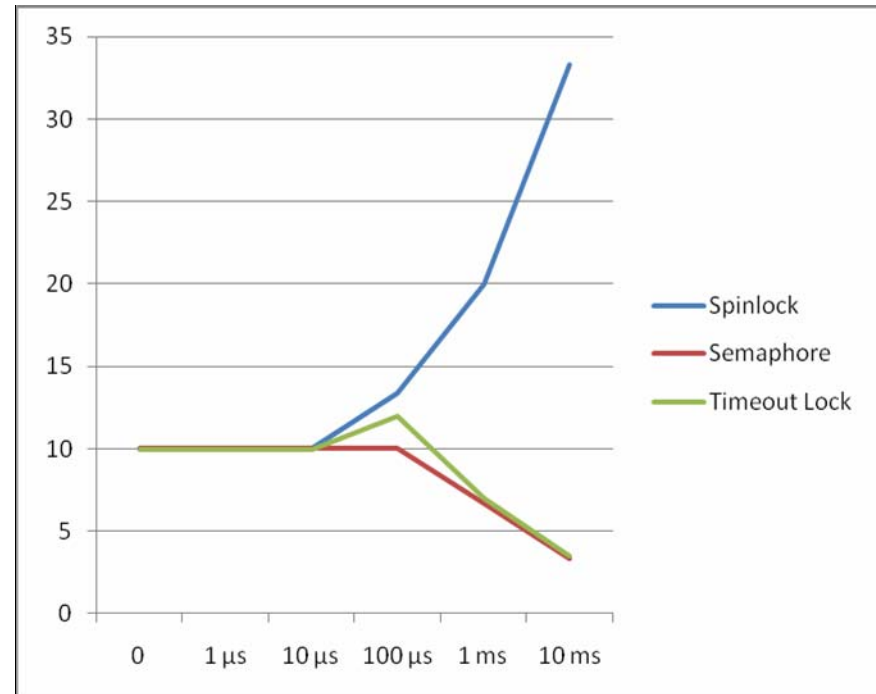


SmartSync: A 100 μ s Timeout Lock

**Average Throughput
(Locks/sec Vs Hold time)**

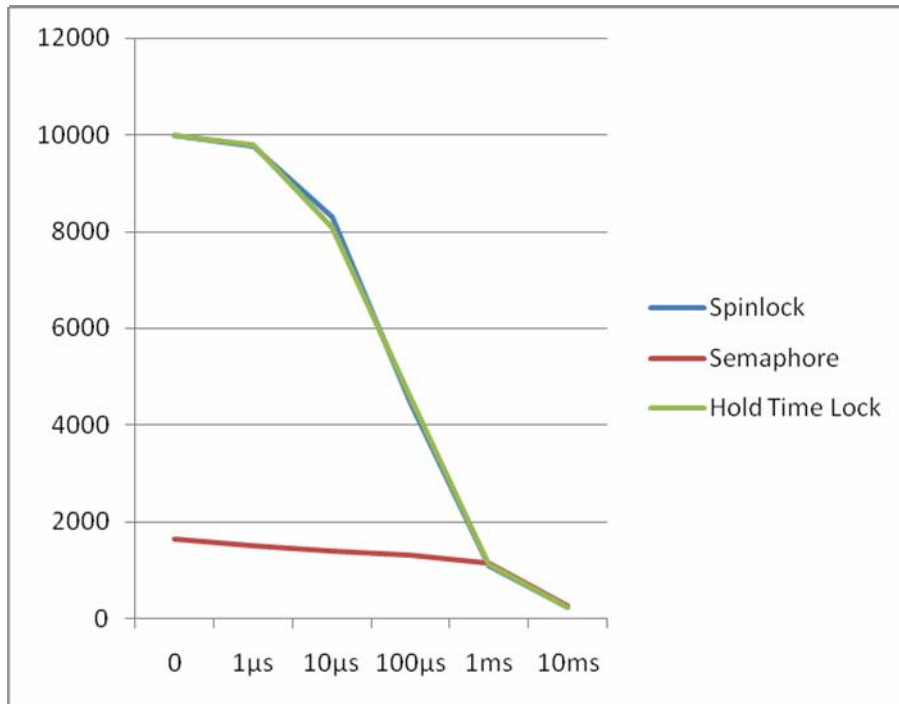


**CPU Utilization
(% Vs Hold Time)**

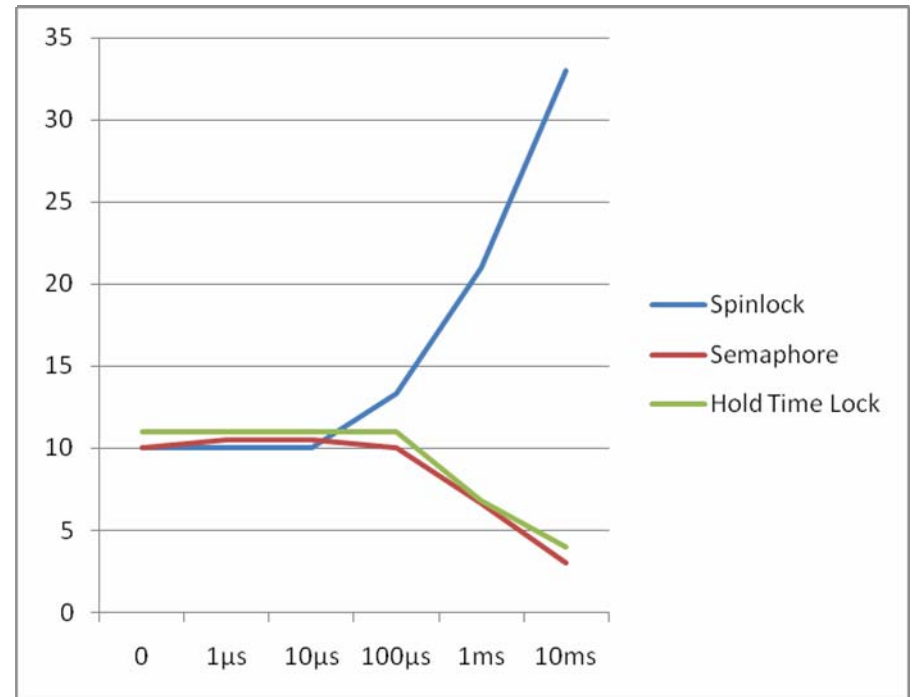


SmartSync: A 100 μ s Predictive Lock

**Average Throughput
(Locks/sec Vs Hold time)**



**CPU Utilization
(% Vs Hold Time)**



Concluding Remarks

- **Deep Energy Inspection**
 - Energy Endoscope
 - First deep energy inspection system
 - System and component level energy usage
- **Per-process Energy Accounting**
 - Etop
 - EnergyView
 - First system to give process-level energy accounting for computing platforms
- **Energy-efficient Software Solutions**
 - SmartSync
 - “Intelligent” resource locking based on run-time data
 - Better energy efficiency than standard locking primitives