PLDI'03 Tutorial: Compilers for Power and Energy Management

Ulrich (Uli) Kremer
Department of Computer Science
Rutgers University

Energy Efficiency and Low-Power Lab

This research is partially supported by NSF and HP/Compaq

Collaborators

Michael Hsiao (Rutgers), Chung-Hsing Hsu (Rutgers)
Jerry Hom (Rutgers), Yang Ni (Rutgers), Chunling Hu (Rutgers)

DarkLab

Ricardo Bianchini (Rutgers)
Taliver Heath (Rutgers), Eduardo Pinheiro (Rutgers)

SmartMessages

Liviu Iftode (Maryland/Rutgers)
Cristian Borcea (Rutgers), Deepa Iver (Rutgers)
Porlin Kang (Rutgers), Akhilesh Saxena (Rutgers)

COMPAQ

Jamey Hicks (HP/Compaq)
Jim Rehg (GeorgiaTech/Compaq)

PennState

Mahmut Kandemir (PennState), Mary Jane Irwin (Penn State),
N. Vijaykrishnan (Penn State),
H. Saputra (Penn State), J. Hu (Penn State)

Why Power/Energy Management?

- **Prolong battery life**
  - prolong operability
  - reduce weight

Why Power/Energy Management?

- **Prolong battery life**
  - prolong operability
  - reduce weight

PLDI'03 Tutorial, June 8, 2003 EEL Laboratory

Why Power/Energy Management?

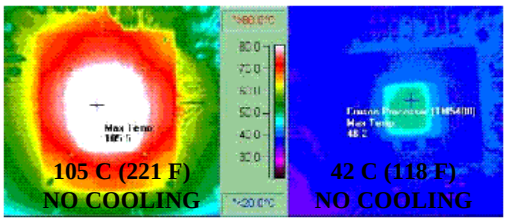
- Prolong battery life
  - prolong operability
  - reduce weight
- **Reduce heat dissipation**
  - packaging costs and cooling
  - reliability

Why Power/Energy Management?

- Prolong battery life
  - prolong operability
  - reduce weight
- **Reduce heat dissipation**
  - packaging costs and cooling
  - reliability

DVD application (Source: Transmeta)

Pentium III processor **Transmeta's Crusoe TM5400**



PLDI'03 Tutorial, June 8, 2003 EEL Laboratory

Why Power/Energy Management?

- Prolong battery life
  - prolong operability
  - reduce weight
- **Reduce heat dissipation**
  - packaging costs and cooling
  - reliability

Frying an egg on the CPU in 11 minutes

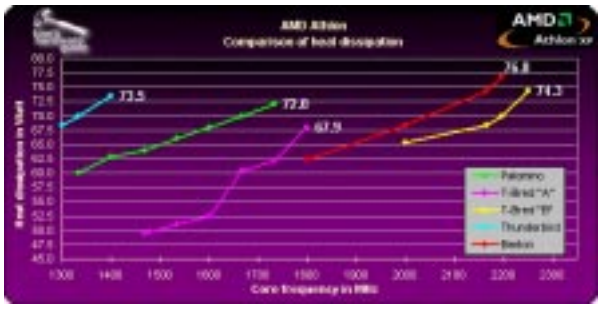
AMD XP 1500+ desktop

0.18µm
1.33GHz, 1.75V
60W, 90°C
Oct. 2001

Source: Trubador
www.hex-tech.co.uk/egg.asp

Why Power/Energy Management?

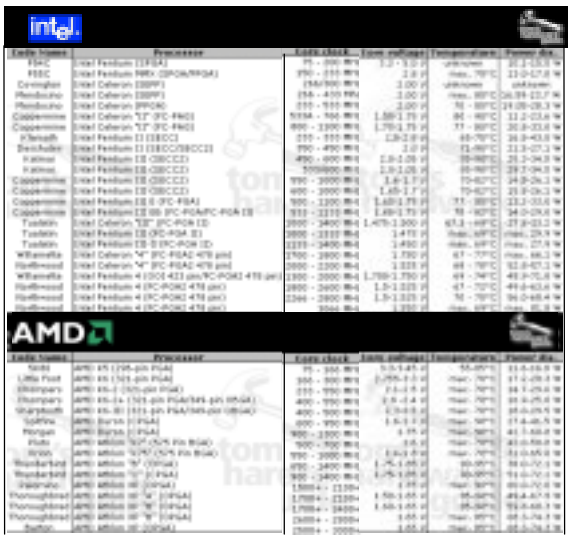
Are things getting "better"?



Source:Tom's Hardware Guide (February 2003)

Why Power/Energy Management?

Source:
Tom's Hardware Guide



http://www6.tomshardware.com/cpu/20030217/cpu_charts-32.html

Why Power/Energy Management?

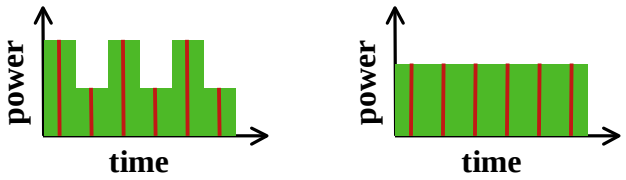
- Prolong battery life
  - prolong operability
  - reduce weight
- Reduce heat dissipation
  - packaging costs and cooling
  - reliability
- **Environmental impact**
  - power plants
  - delivery infrastructure

Why Power/Energy Management?

- Prolong battery life
  - prolong operability
  - reduce weight
- Reduce heat dissipation
  - packaging costs and cooling
  - reliability
- **Environmental impact**
  - power plants
  - delivery infrastructure

Power vs. Energy

power: activity level at a given point in time
energy: total amount of activity

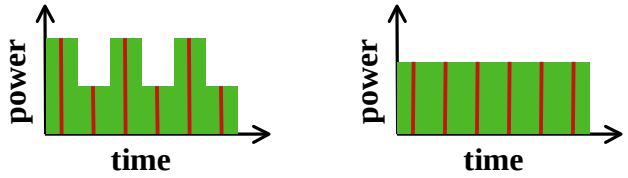


same energy, different (peak) power

optimizing for (peak) power == optimizing for energy?

Power vs. Energy

power: activity level at a given point in time
energy: total amount of activity



same energy, different (peak) power

optimizing for (peak) power == optimizing for energy?

ANSWER: Not necessarily! Example: re-schedule activities

Li et al., [1], Parikh et al. [2]

Power/Energy vs. Performance

performance: overall program execution time

optimizing for power/energy == optimizing for performance?

Power/Energy vs. Performance

performance: overall program execution time

optimizing for power/energy == optimizing for performance?

ANSWER: (a) Mostly Yes, at least for traditional optimizations that reduce overall computation and memory activity

- redundancy elimination (CSE, PRE, dead code elimination)
- strength reduction (e.g.: replace 2*a with a+a), loop invariant code motion
- memory hierarchy (locality) optimizations (register allocation, loop interchange, loop distribution, blocking for cache)

Power/Energy vs. Performance

performance: overall program execution time

optimizing for power/energy == optimizing for performance?

ANSWER: (a) Mostly Yes, at least for traditional optimizations that reduce overall computation and memory activity

- redundancy elimination (CSE, PRE, dead code elimination)
- strength reduction (e.g.: replace 2*a with a+a), loop invariant code motion
- memory hierarchy (locality) optimizations (register allocation, loop interchange, loop distribution, blocking for cache)

ANSWER: (b) Not really, in particular for optimizations that exploit tradeoff between power/energy usage and performance

- loop invariant code motion, aggressive speculation
- blocking for cache (Kandemir et al.[3])
- DFVS, resource hibernation, remote task execution, QoR

Power/Energy vs. Performance

```
for (i=0, i<10; i++) {
  a = b* 2;
  c[i] = d[i] + 2.0;
}
```

(A)

```
a = b* 2;
for (i=0, i<10; i++) {
  c[i] = d[i] + 2.0;
}
```

(B)

Which code is better in terms of power/energy and which in terms of performance?

Power/Energy vs. Performance

```
for (i=0, i<10; i++) {
  a = b* 2;
  c[i] = d[i] + 2.0;
}
```

(A)

```
a = b* 2;
for (i=0, i<10; i++) {
  c[i] = d[i] + 2.0;
}
```

(B)

Which code is better in terms of power/energy and which in terms of performance?

ANSWER: It depends

- simple RISC architecture: (B)
- VLIW or superscalar architecture with empty "slots": (A)

Tiwari et al., [22], Kandemir et al. [23], L.N. Chakrapani et al. [12], M. Valluri et al. [13]

Power/Energy vs. Performance

You can run, but you cannot hide

- pushing instructions onto the non-critical execution path; ("hiding") does not necessarily reduce energy
- higher threshold for profitability of speculation

You cannot beat hardware

- if an operation is implemented in hardware, and an applications needs it, that's the best you can do (e.g.: floating-point unit)
- need to be able to disable hardware if not in use

Keep the overall picture in mind

- performance is measured for the entire program
- power/energy should also be measured for the entire system, in addition to optimized system component(s)

Why Compiler and not OS/Hardware?

Compiler advantages:

- low or no overhead (power/energy and performance) at program execution time
- may know about "future" program behavior through aggressive, whole program analysis.
- can better identify profitability of high overhead optimizations based on large context analysis.
- can reshape program behavior through code transformations and thereby enable optimizations.

Compiler disadvantages:

- insufficient information about runtime program behavior may lead to code of poor quality

Why Compiler and not OS/Hardware?

Scenarios:

- single user environment: compiler-directed power and energy management is directly "executed" by underlying OS/HW.
- multiple user environment: power/energy application profile is used by OS to make scheduling decisions.

Compilers for Power and Energy Management

- Dynamic Frequency and Voltage Scaling (DFVS)
  - Benefit analysis (SPECfp95)
- Dynamic Resource Configuration/Hibernation
  - Benefit analysis for 802.11 card (adi, tomcatv, shal)
  - Benefit analysis for disk (mp3, mpeg, and sftp)
- Remote Task Execution
  - Benefit analysis for StrongARM/Pentium (TourGuide)
- QoR Optimizations
  - Examples
- Summary and Future Work
- Wish List for Architects

Dynamic Voltage/Frequency Scaling

Goal: Reduce the energy needed for executing an application with a **soft** execution time **deadline** constraint.

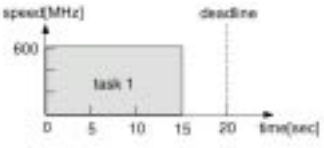
Power and Energy are proportional to $C V^2 f$

Safety:	always safe
Opportunity:	CPU idle time (CPU DFVS)
Profitability:	up to 49% (avg. 21.5%) CPU power savings with up to 4.8% (avg. 2.1%) performance penalties on SPECfp95 on 600MHz - 1.2 GHz AMD Athlon4

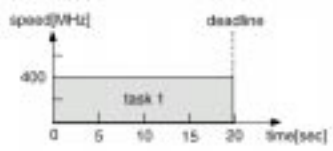
Hsu et al., [5, 6, 15, 16, 26,31], H. Saputra et al. [19], Mosse et al. [20], Azevedo et al. [18], Xie et al. [29]

Dynamic Voltage/Frequency Scaling

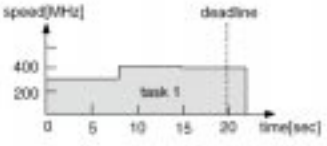
Scale voltage and frequency to save energy and still meet deadline



(a) original schedule.

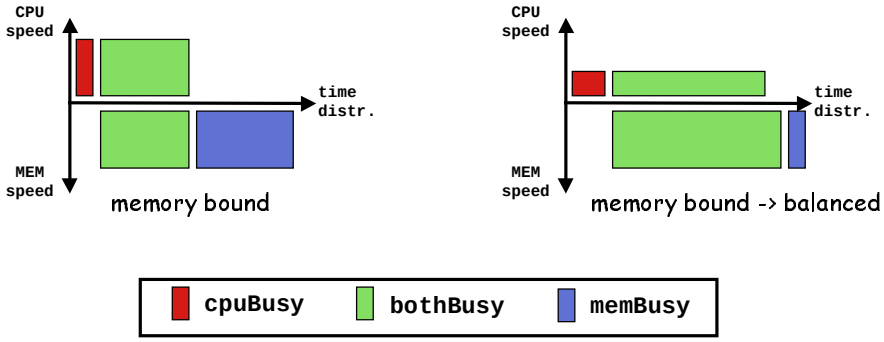


(b) voltage scaled schedule.



(c) power-performance tradeoffs.

Opportunity: Unbalanced Program Regions



Basic Compilation Strategy

Goal: Assign minimal voltages and frequencies to different program regions such that overall performance is only slightly decreased (~ 1%).

Opportunity:

Program regions with unbalanced computation and memory requirements.

Architectures that allow overlap of computations and data accesses.

Basic Compilation Strategy

Current Algorithm Outline

(1) Identify program regions as scheduling candidates
(sequences of loop nests, procedure calls, if-statements)

(2) Performance modeling
- determine cpuBusy, memBusy, bothBusy of scheduling candidates
- determine relative execution times of scheduling candidates
- use results to compute slowdown factor δ
(CPU slow-down) under a soft deadline constraint
(≤ 1% performance penalty), and select **single** best candidate

(3) Generate voltage/frequency scheduling instructions
and adjust performance optimizations.

Basic Performance Model

$$T = cpuBusy + memBusy + bothBusy$$

$$T_{new}(\delta) = \delta * cpuBusy + \max \begin{pmatrix} bothBusy + memBusy \\ \delta * bothBusy \end{pmatrix}$$

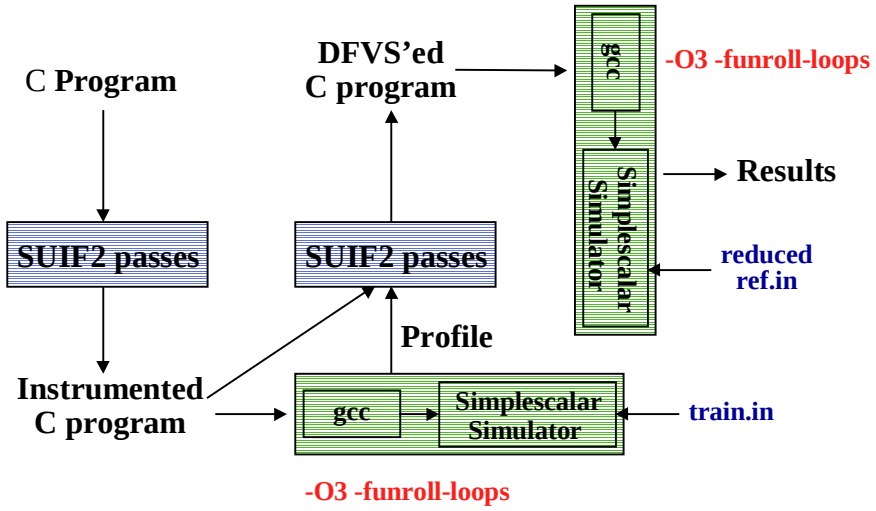
Constraints on choosing δ :

(1) $(\delta - 1) * cpuBusy \leq 1\%$

(2) $1 \leq \delta \leq 1 + \frac{memBusy}{bothBusy}$

(3) memory latency is divisible by δ

EEL_{dfvs} Prototype Compiler



SUIF [24], Burger et al. [25]

Benefit Analysis

- SimpleScalar with memory hierarchy extensions
  - cycle accurate simulation
  - out-of-order superscalar processors
  - branch prediction and speculative execution
- Simulated out-of-order target architecture:
  - 1 cycle L1 cache, non-blocking
  - 10 cycles L2 cache, non-blocking
  - 100 cycles memory, blocking
  - instruction window size = 64
  - instruction issue width = 4 per cycle
- Switching overhead of 10,000 cycles

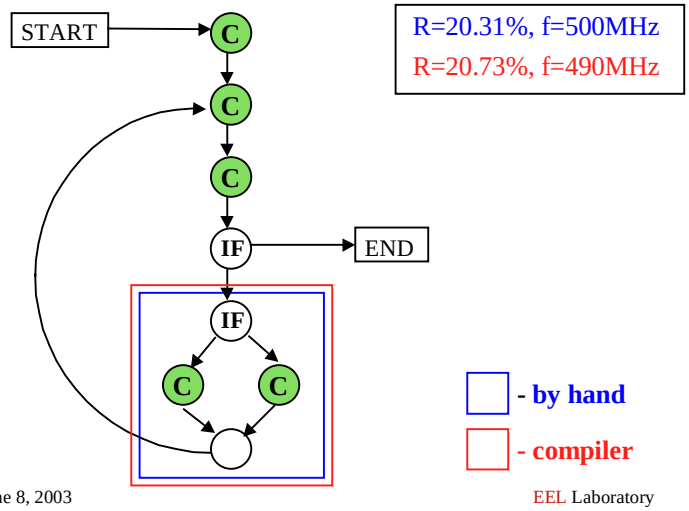
Benefit Analysis (ref.in)

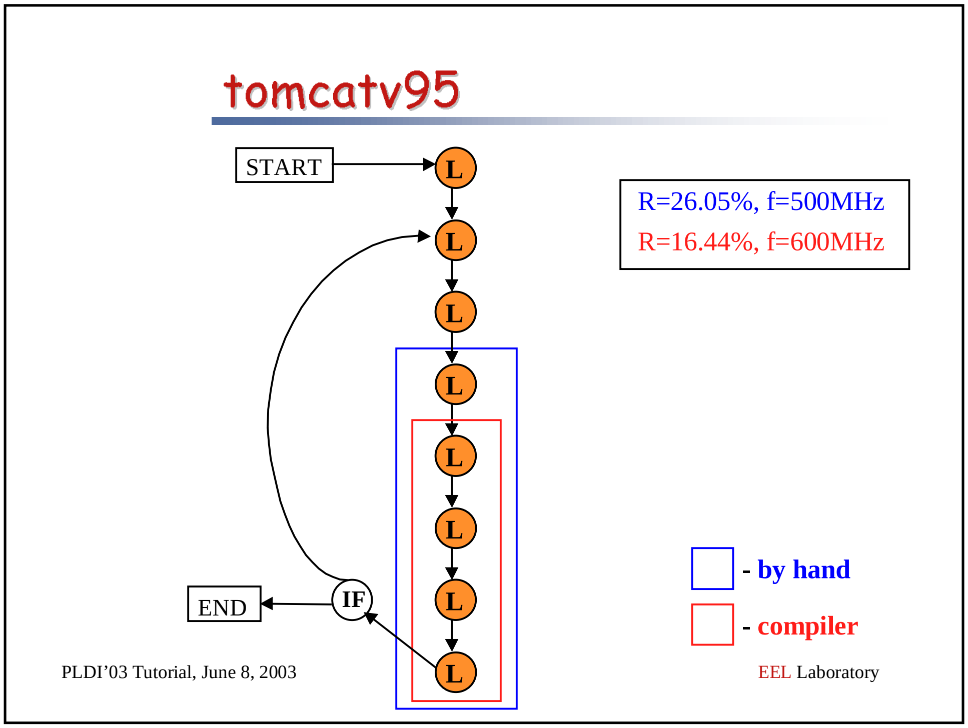
Benchmark	by hand + ref.in			compiler		
	Slow-down	Exec. Time	CPU Energy	Slow-down	Exec. Time	CPU Energy
swim95	2.02	101.68%	76.79%	2.07	102.67%	75.70%
tomcatv95	2.44	101.99%	76.25%	1.69	100.47%	83.49%
applu	1.58	101.82%	90.43%	1.24	101.22%	93.94%
hydro2d	1.33	101.47%	84.61%	1.11	101.69%	83.42%

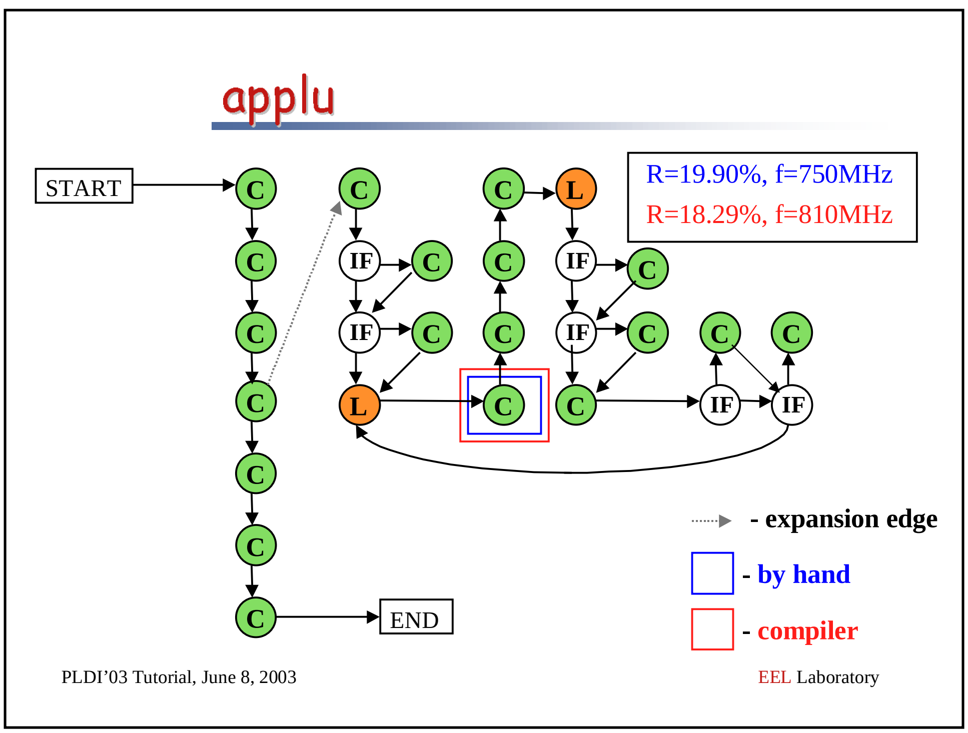
compiler
- **considers DFVS overheads**
- **enumerates all possible regions**
- **automates the process**
- **uses different input for training**

Soft deadline: 1%
Single region
Up to 1 GHz
Scaling cost=10µs

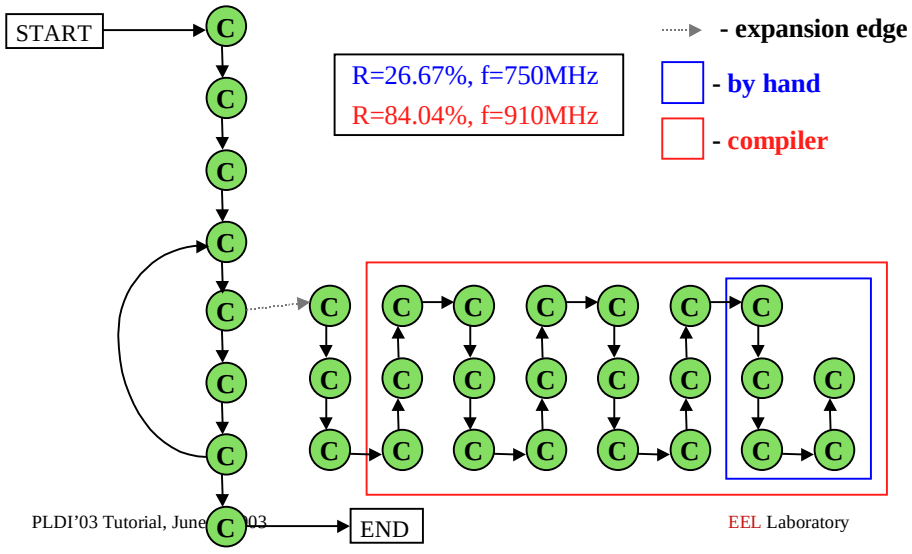
swim95




tomcatv95
START
R=26.05%, f=500MHz
R=16.44%, f=600MHz
L
L
L
L
L
L
L
L
IF
END
- by hand
- compiler
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory


applu
START
R=19.90%, f=750MHz
R=18.29%, f=810MHz
C
IF
L
END
- expansion edge
- by hand
- compiler
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory

hydro2d



DVFS Opportunity Study

Highly optimizing compiler: DEC's f90 compiler -O5

Alpha 21264-like processor, ref.in input data set

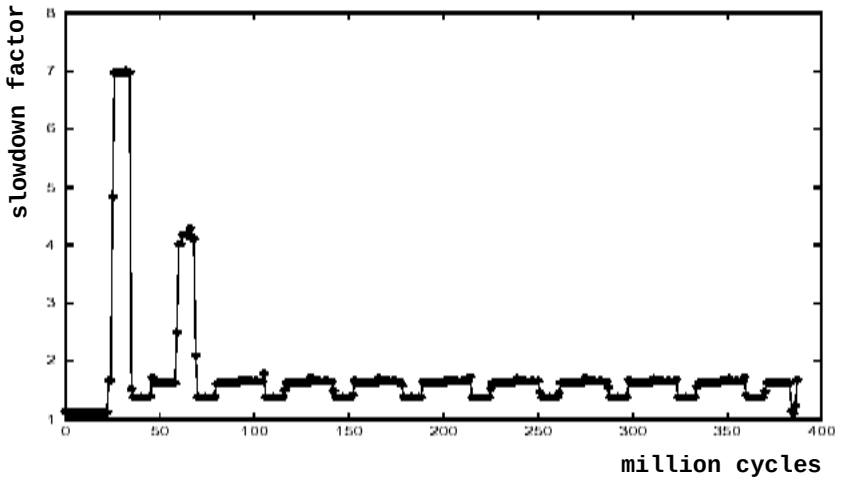
modified Wattch (SimpleScalar based) simulator, 10^6 cycles summaries

10% soft deadline performance penalty, no switching costs

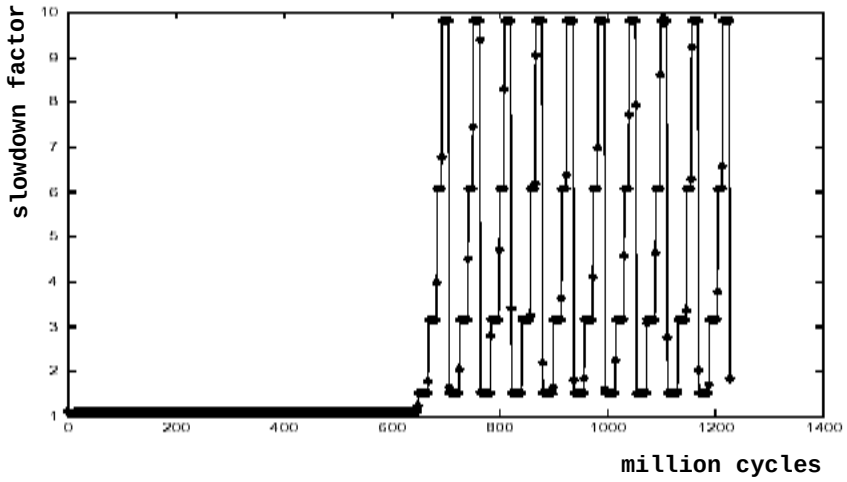
Benchmark	CPU Energy	Performance Penalty
swim95	38.86%	9.50%
tomcatv95	28.30%	8.42%
applu	51.35%	7.27%
hydro2d	48.73%	7.32%

Hsu et al. [5], Brooks et al. [21]

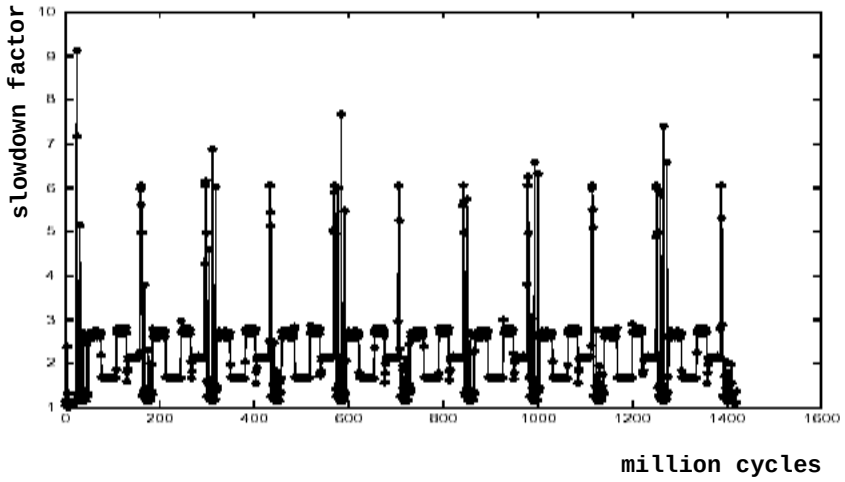
swim95



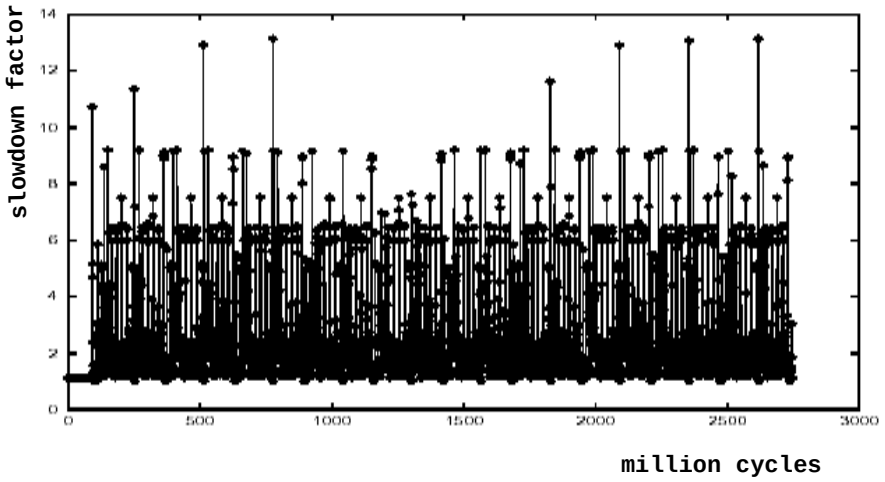
tomcatv95



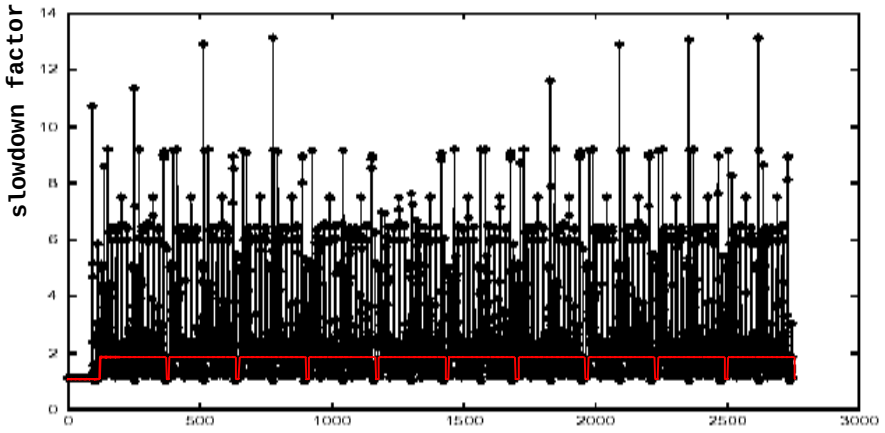
applu



hydro2d



hydro2d



compiler δ= 1.8

Two Commercial DFVS Processors

Performance level	Compaq Presario 715US **Mobile Athlon 4**		Fujitsu LifeBook P2040 **Crusoe TM5800**	
	f (MHz)	**V (volts)**	**f (MHz)**	**V (volts)**
1	600	1.15	300	1.00
2	700	1.20	533	1.10
3	800	1.25	667	1.20
4	900	1.30	733	1.25
5	1000	1.35	800	1.30
6	1100	1.40		
7	1200	1.45		

Compaq Presario 715US (ref.in)

EEL_{dfvs} **: 7 discrete performance levels, actual measurements**

Linux 2.4.18, compiler: g77 -O2,

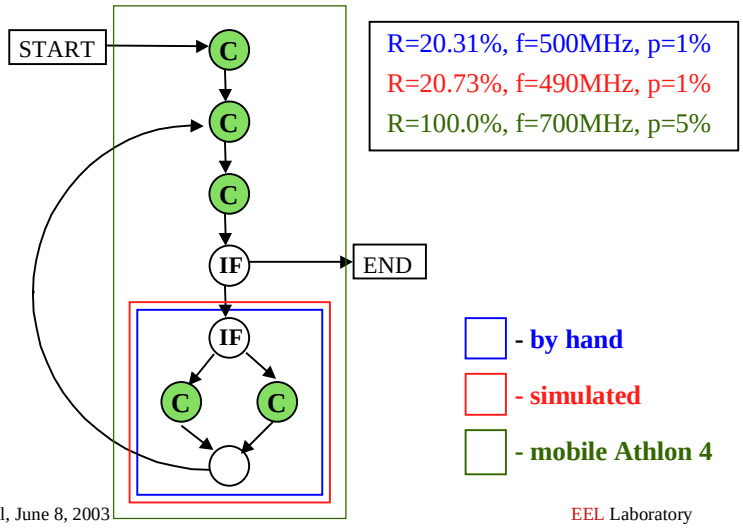
Mobile Athlon 4 processor

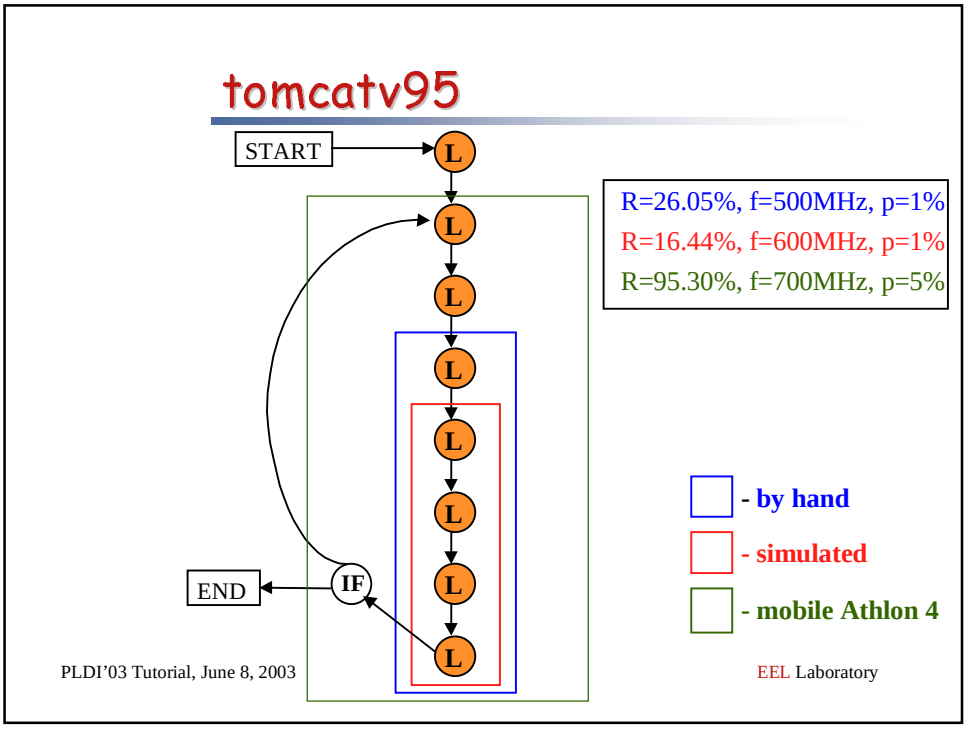
Physical measurement using power meter

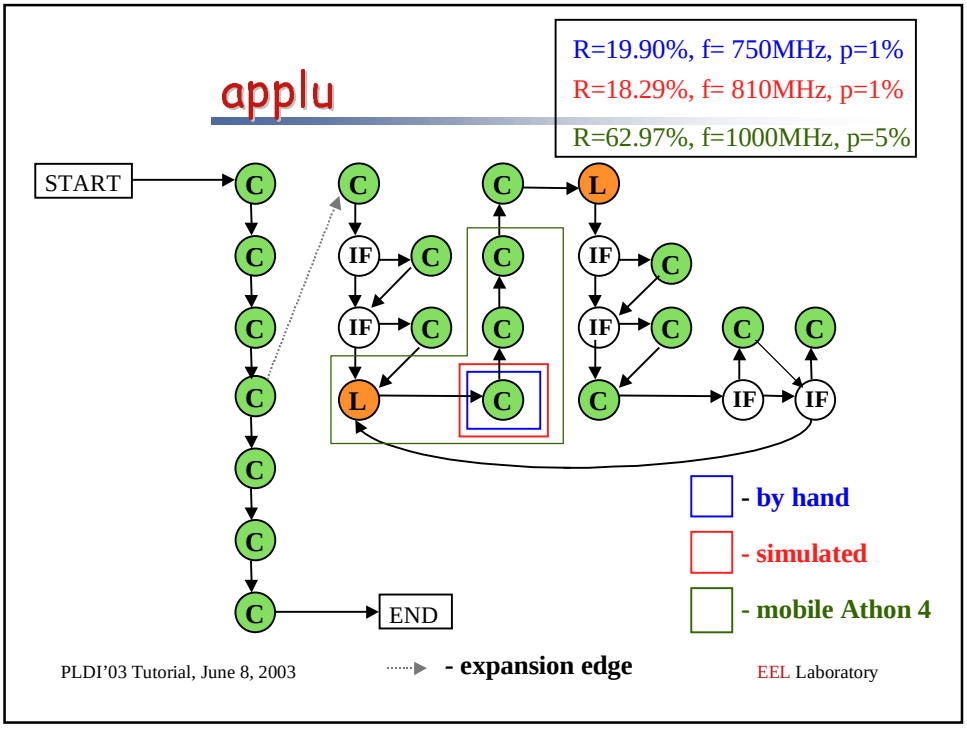
5% soft performance deadline

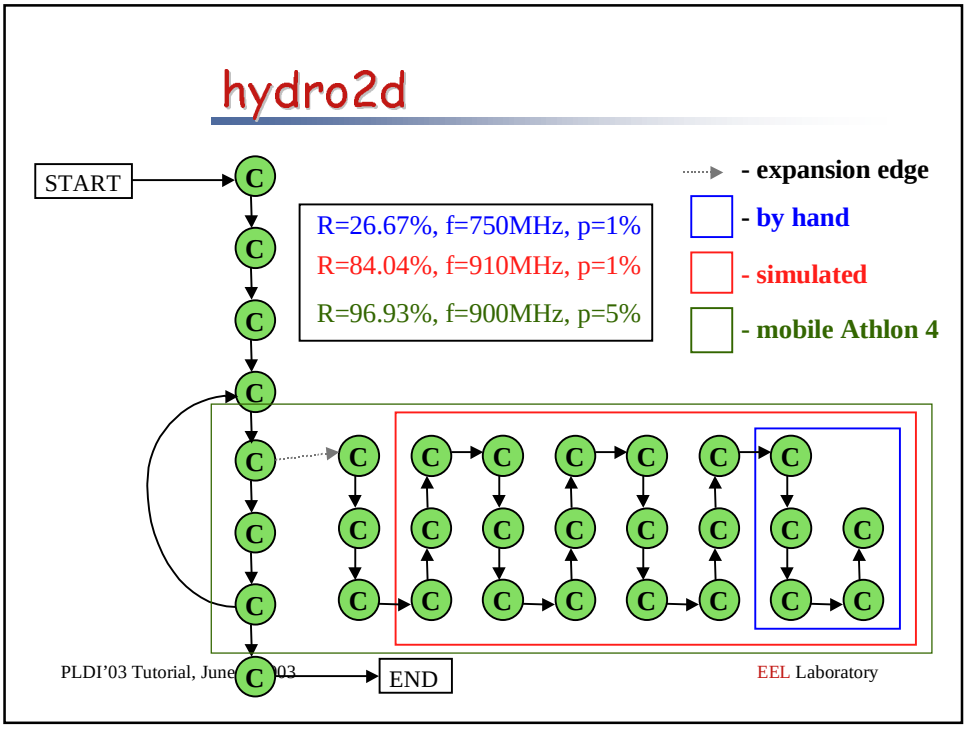
Benchmark	CPU Power	Performance Penalty
swim95	57.13%	2.93%
tomcatv95	50.56%	1.18%
applu	73.16%	4.72%
hydro2d	61.03%	2.21%

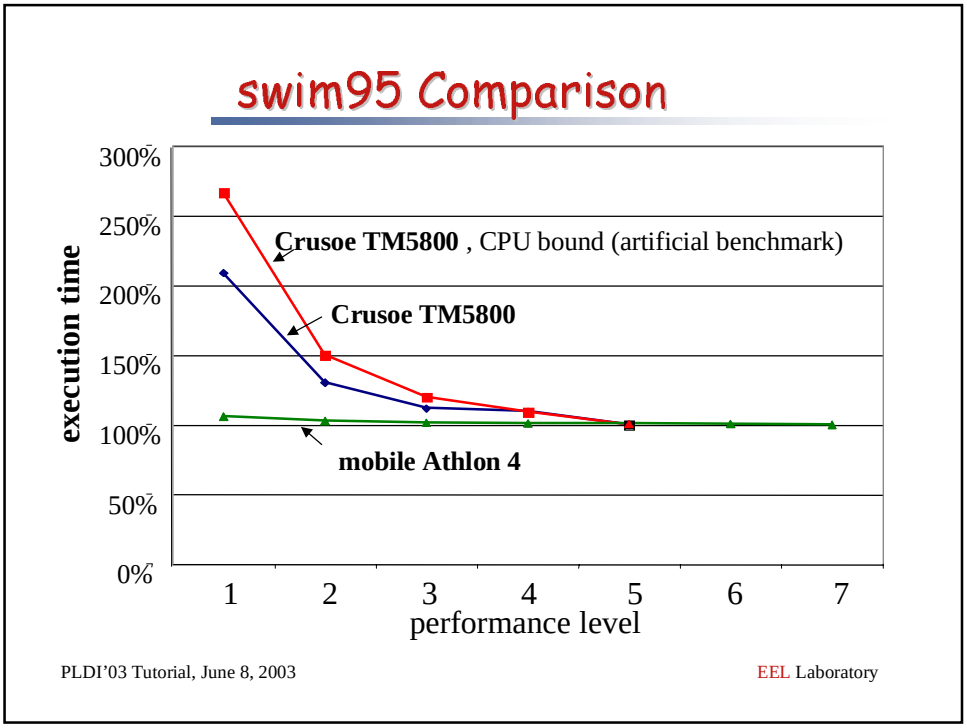
swim95




tomcatv95
START
L
L
L
L
L
L
L
L
IF
END
R=26.05%, f=500MHz, p=1%
R=16.44%, f=600MHz, p=1%
R=95.30%, f=700MHz, p=5%
- by hand
- simulated
- mobile Athlon 4
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory


applu
R=19.90%, f= 750MHz, p=1%
R=18.29%, f= 810MHz, p=1%
R=62.97%, f=1000MHz, p=5%
START
C
L
IF
END
- by hand
- simulated
- mobile Athon 4
- expansion edge
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory


hydro2d
START
END
C
R=26.67%, f=750MHz, p=1%
R=84.04%, f=910MHz, p=1%
R=96.93%, f=900MHz, p=5%
- expansion edge
- by hand
- simulated
- mobile Athlon 4
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory


swim95 Comparison
execution time
300%
250%
200%
150%
100%
50%
0%
Crusoe TM5800 , CPU bound (artificial benchmark)
Crusoe TM5800
mobile Athlon 4
1
2
3
4
5
6
7
performance level
PLDI'03 Tutorial, June 8, 2003
EEL Laboratory

Energy Delay Product Comparison

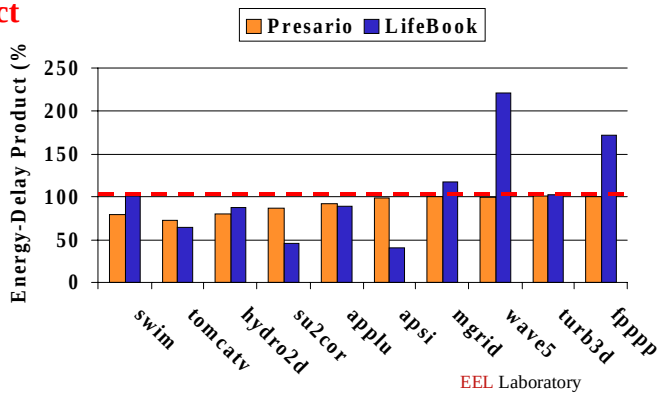
Overall system physical measurements
Comparison relative to Compaq Presario at peak (1.2GHz)
LifeBook uses Transmeta's LongRun technology
Linux 2.4.18, g77 -O2

Energy Delay Product

$E * T = P * T^2$

Presario 715US:
33.6W - 57.3W

LifeBook P2040:
13.0W - 15.9W



Compilers for Power and Energy Management

- ☑ Dynamic Frequency and Voltage Scaling (DFVS)
  - Benefit analysis (SPECfp95)
- ❑ **Dynamic Resource Configuration/Hibernation**
  - Benefit analysis for 802.11 card (adi, tomcatv, shal)
  - Benefit analysis for disk (mp3, mpeg, and sftp)
- ❑ Remote Task Execution
  - Benefit analysis for StrongARM/Pentium (TourGuide)
- ❑ QoR Optimizations
  - Examples
- ❑ Summary and Future Work
- ❑ Wish List for Architects

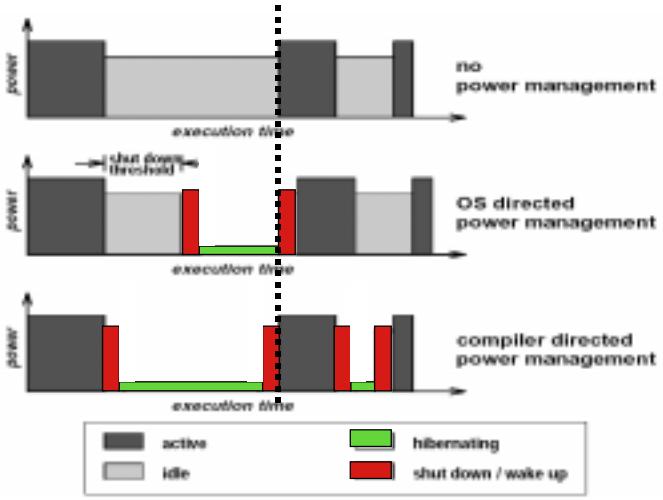
Dynamic Resource Configuration/ Hibernation

Goal: Reduce power/energy by dynamically hibernating resources not required by the application (dynamic power management)

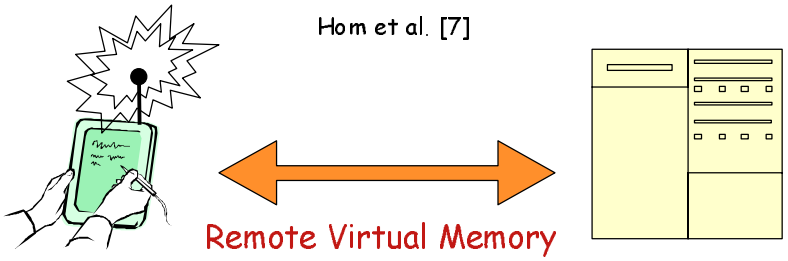
Safety: mostly safe
Opportunity: communication card, disk, cache lines, memory blocks, …
Profitability: energy reductions up to 20% (802.11b/iPAQ, over OS approach), and up to 89% (disk)

Delaluz et al. [5], Hom et al. [7], Heath et al. [14]

Threshold based OS vs. Compiler Directed Hibernation



Pervasive Computing Environment



Hand-held PC

- "Workstation" class machine
- Depends on battery power
- Wireless communication
- No disk (limited resources)

Desktop / Server

- One order of magnitude more resources than handheld

Basic Compilation Strategy

Phase 1: Region analysis
- granularity of program regions for card hibernation
- for each region: card must be on, or card may be off

Phase 2: Reshape analysis
- page fault clustering ⇒ move page faults to region entry
- loop index set splitting ⇒ adjustment of granularity

Phase 3: Hibernate/activate instruction generation
- use performance prediction to
  (a) avoid hibernation if closely followed by activation
  (b) activate just-in-time to avoid performance penalty

EEL_{RM} Prototype Compiler

Based on SUIF2 compiler infrastructure

Phase 1: Region analysis

- Regions: inner loop nests (phases) or system calls (printf)
- Build region control flow graph (RCFG)
- For each region compute REF sets (entire data objects; future: DAD representation)
- Determine for each regions what data objects / code will be in memory; simulate LRU page replacement policy by "walking" over RCFG.
- Mark regions as "card on" or " card may be off"

EEL_{RM} Prototype Compiler

Phase 2: Reshape analysis

Page fault clustering (hand simulated)

Phase 3: Hibernate/activate instruction generation

No performance prediction ⇒

- no just-in-time card activation, i.e., either on demand or when reaching "card on" region
- card hibernation forced if region is marked as "card may be off"

Experimental Benefit Study

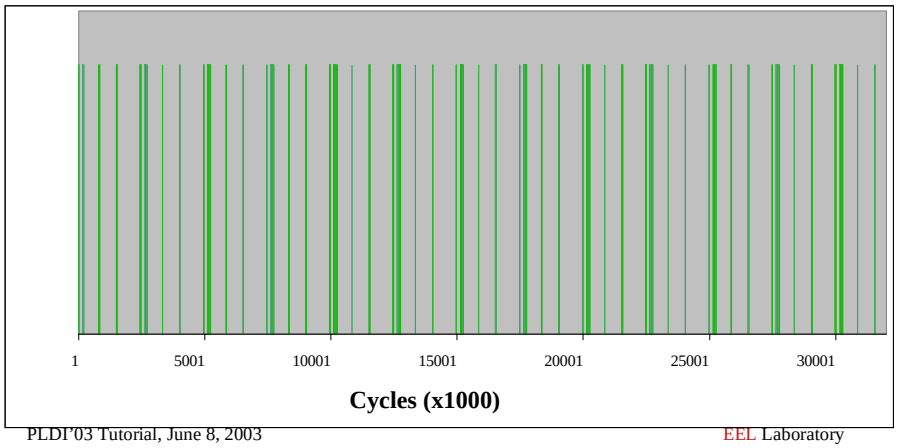
***N*: array dimension length (float)**
***M*: # of 4KB memory pages**

Parameters	*shal*	*adi*	*tomcatv*
N	32	16	32
M	32	16	16

- Prediction accuracy of page faults at region granularity; based on modified SimpleScalar simulator
- Comparison with threshold based OS techniques
  - relative energy savings
  - performance penalties (SimpleScalar)

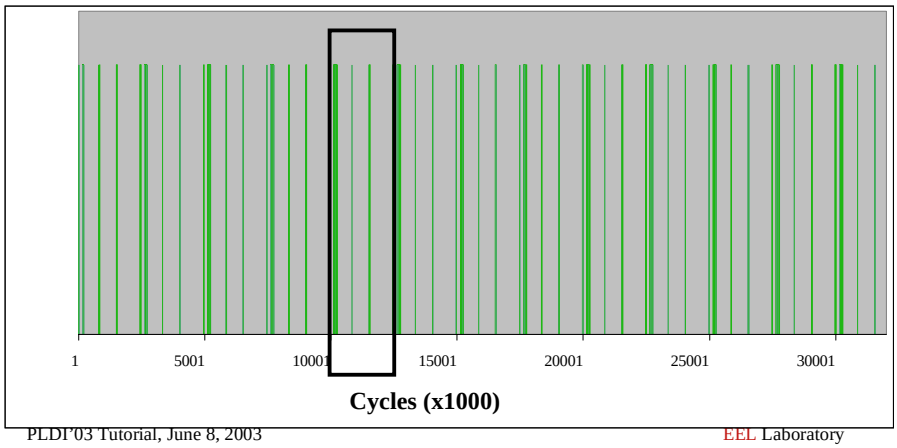
EEL_{RM} : Experimental Results

dynamic page faults for tomcatv (region summary)



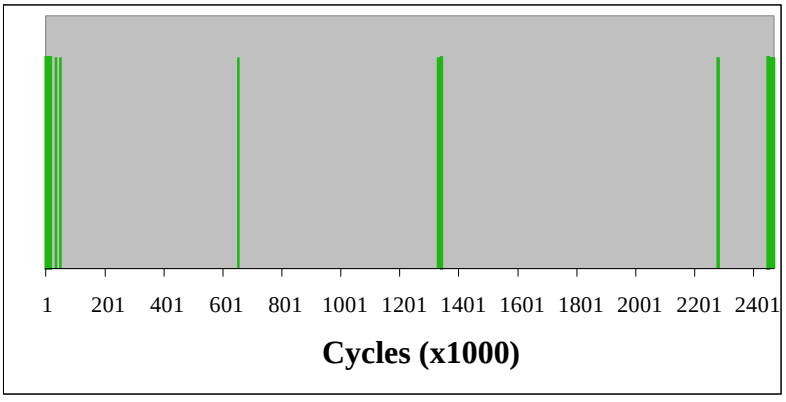
EEL_{RM} : Experimental Results

dynamic page faults for tomcatv (region summary)



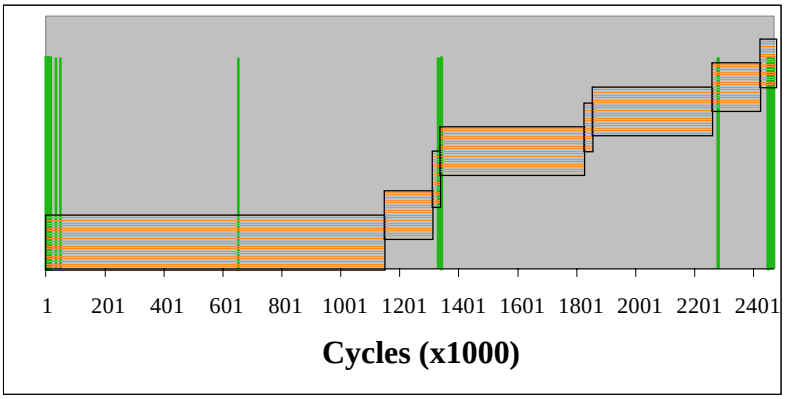
EEL_{RM} : Experimental Results

dynamic page faults for tomcatv



EEL_{RM} : Experimental Results

dynamic page faults for tomcatv with loop structure



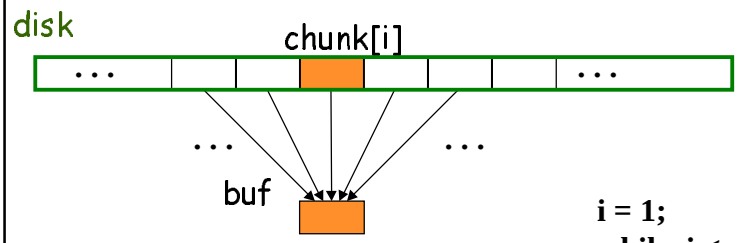
Relative Energy Savings vs. threshold based OS techniques

1 x threshold = 25,000 cycles, 40% energy cost of card

OS threshold	EEL_{RM} Energy Results			
	shal	*adi*	*tomcatv*	*tomcatv (PFC)*
1×	101.0	99.3	126.5	95.3
10×	100.1	92.6	116.3	87.6
20×	99.7	86.2	104.2	78.5
24×	**99.4**	—	—	—
30×	99.7	80.6	98.6	74.3
35×	99.7	78.1	**96.7**	**72.9**
54×	99.7	**69.1**	96.7	72.8
∞	99.7	71.3	96.7	72.8

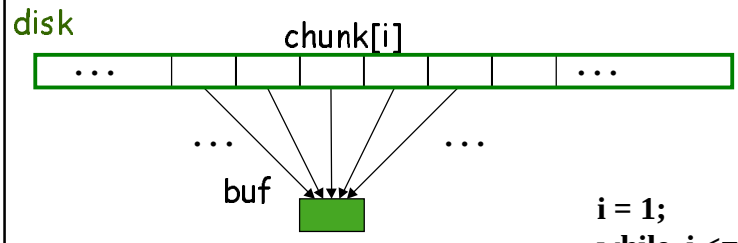
Hand-held / Lap-top with Local Disk

Heath et al., [14]



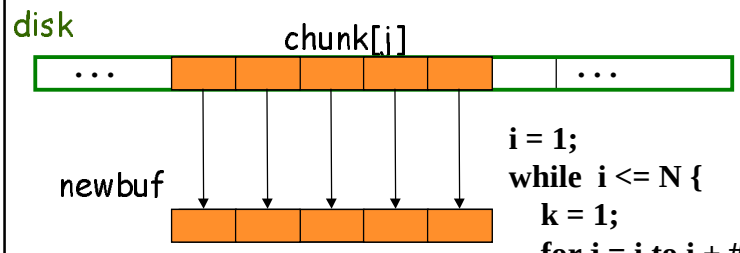

```
i = 1;
while  i <= N {
  read chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```


Hand-held / Lap-top with Local Disk



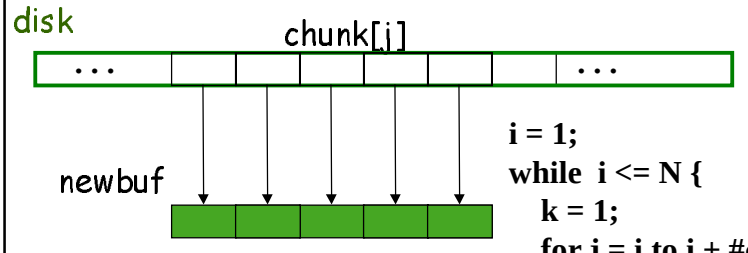

```
i = 1;
while  i <= N {
  read chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```

Hand-held / Lap-top with Local Disk



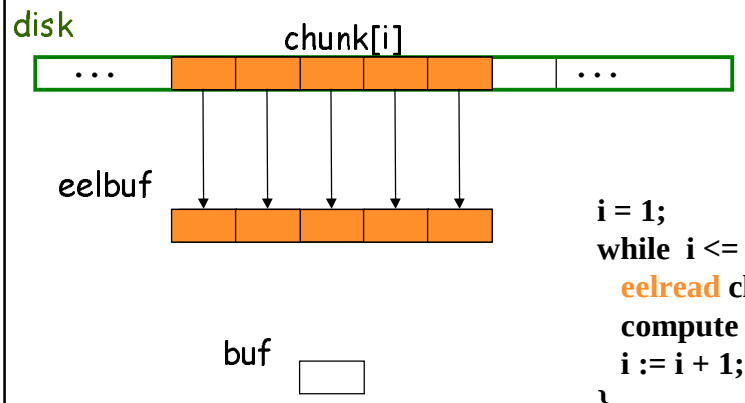

```
i = 1;
while  i <= N {
   k = 1;
   for j = i to i + #chunks
     read chunk[j] into newbuf[k]
   for k = 1 to i + #chunks
     compute newbuf[k]
  i := j + 1;
}
```


Hand-held / Lap-top with Local Disk



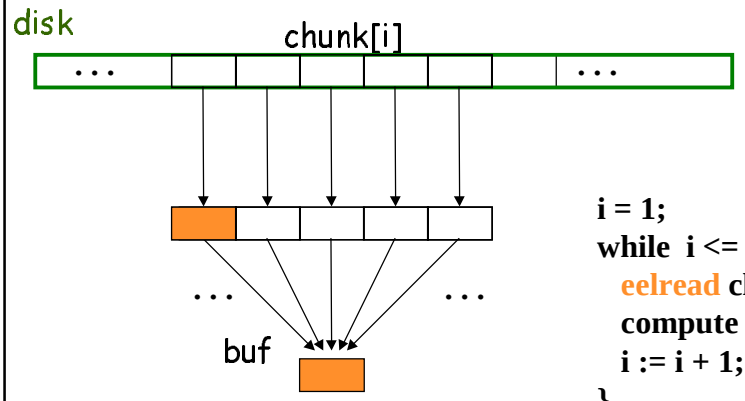

```
i = 1;
while  i <= N {
   k = 1;
   for j = i to i + #chunks
     read chunk[j] into newbuf[k]
   for k = 1 to i + #chunks
     compute newbuf[k]
  i := j + 1;
}
```

Hand-held / Lap-top with Local Disk



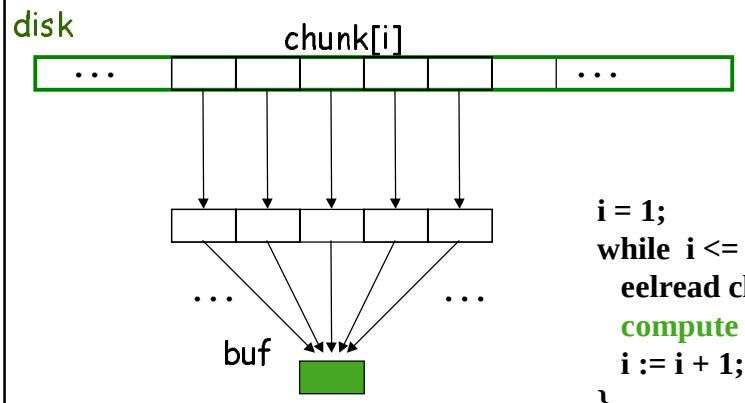

```
i = 1;
while  i <= N {
  eelread chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```


Hand-held / Lap-top with Local Disk



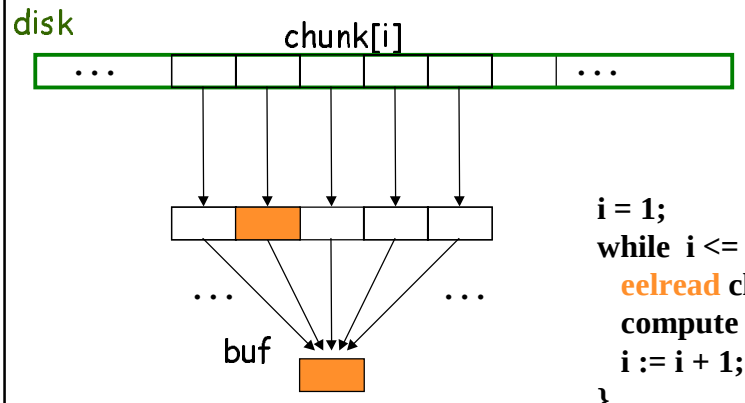

```
i = 1;
while  i <= N {
  eelread chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```

Hand-held / Lap-top with Local Disk



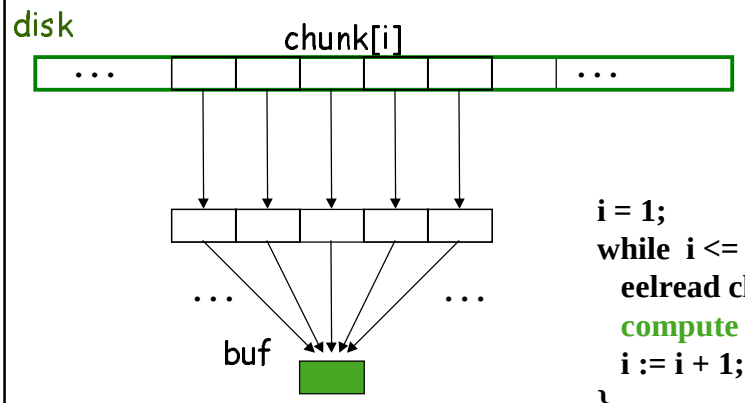

```
i = 1;
while  i <= N {
  eelread chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```


Hand-held / Lap-top with Local Disk




```
i = 1;
while  i <= N {
  eelread chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```

Hand-held / Lap-top with Local Disk




```
i = 1;
while  i <= N {
  eelread chunk[i] into buf
  compute on buf;
  i := i + 1;
}
```


Basic Compilation Strategy

Current Algorithm Outline

Compiler (1) User annotates file descriptors where buffering should be performed
(2) Compiler propagates file attributes through program and replaces calls to I/O operations by calls to EEL library I/O operations (currently read, lseek)

eelread (1) preserves the semantics of original read
(2) measures the performance characteristics of the disk through user-transparent runtime profiling
(3) allocates and manages buffer of appropriate size
(4) notifies OS about expected idle times of disk

Benefit Analysis

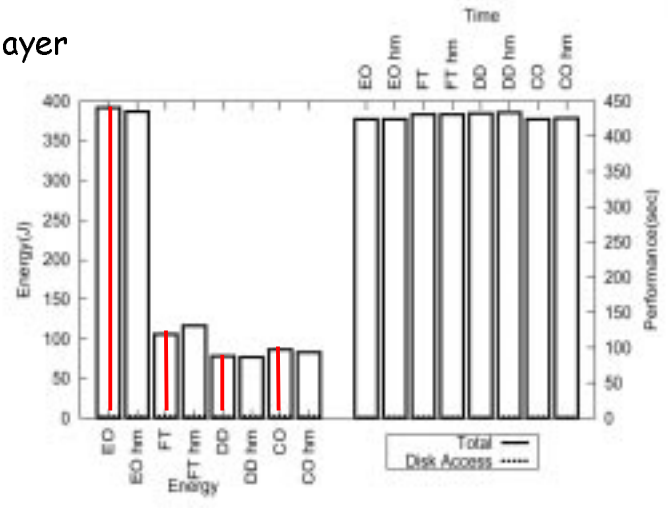
- Fujitsu Disk, 6 Gbyte, 4200 rpm, ATA-5 interface, has four power states (active, idle, standby, sleep)
- OS supports different device management policies:
  - Energy-Oblivious (EO)
  - Fixed-Thresholds (FT)
  - Direct Deactivation (DD)
  - Pre-Activation (PA)
  - Combined DD + PA (CO)
- physical measurements for disk energy consumption for three applications: MP3 player, MPEG player, and sftp

Results: - disk energy savings in the range of 55% - 89%
- hand-modified and compiler nearly identical quality

Hand-held / Lap-top with Local Disk

MP3 player

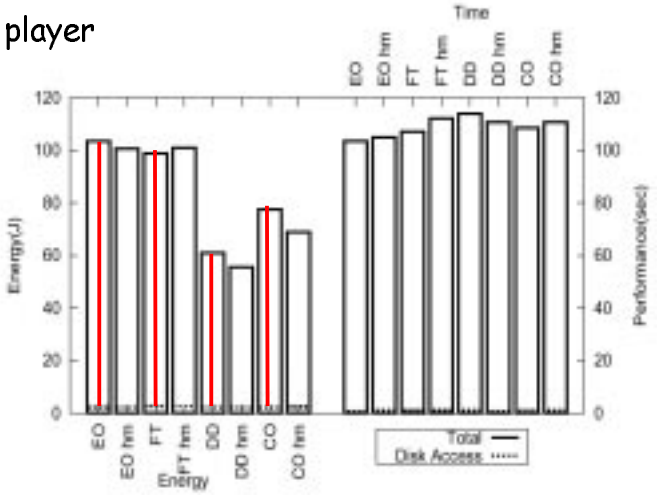
compiler



Hand-held / Lap-top with Local Disk

MPEG player

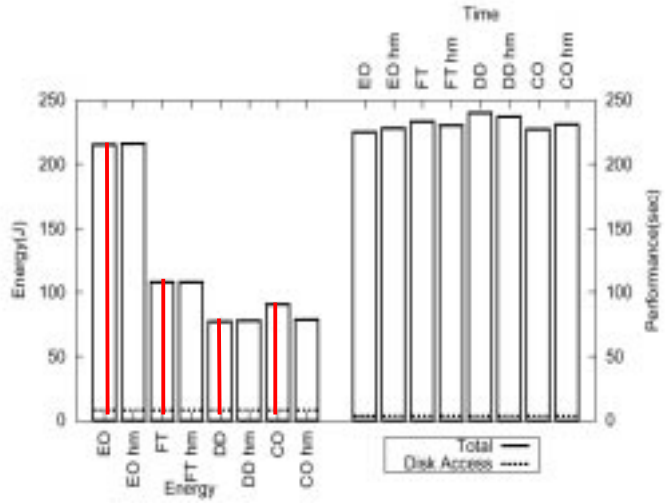
compiler



Hand-held / Lap-top with Local Disk

sftp

compiler



Compilers for Power and Energy Management

- ☑ Dynamic Frequency and Voltage Scaling (DFVS)
  - Benefit analysis (SPECfp95)
- ☑ Dynamic Resource Configuration/Hibernation
  - Benefit analysis for 802.11 card (adi, tomcatv, shal)
  - Benefit analysis for disk (mp3, mpeg, and sftp)
- ❑ **Remote Task Execution**
  - Benefit analysis for StrongARM/Pentium (TourGuide)
- ❑ QoR Optimizations
  - Examples
- ❑ Summary and Future Work
- ❑ Wish List for Architects

Remote Task Execution

Goal: Save energy on mobile device by off-loading computation to remote host

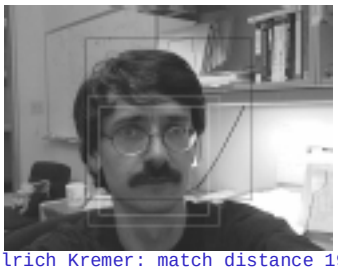
Safety:	Sophisticated compile-time analyses
Opportunity:	Mobile applications that contain tasks with small to moderate data exchange between them
Profitability:	Up to one order of magnitude (10x) on image understanding application (TourGuide) running on Skiff and iPAQ (StrongARM)

Kremer et al. [8], Flinn et al. [17], Li et al. [11], Palm et al. [27]

TourGuide: A Face Detection and Recognition System

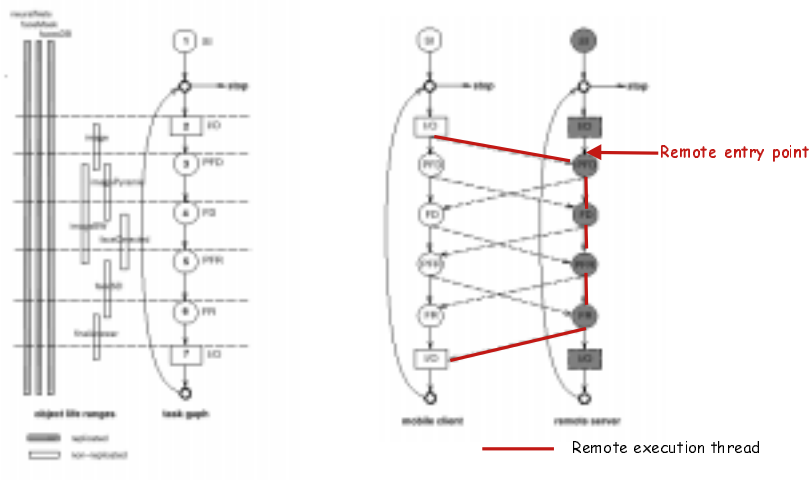
- Developed at Compaq's Cambridge Research Laboratory (CRL)
- Input: uncompressed B/W image; output: answer string
- Approx. 6000 lines of C code
- Runs under Linux on x86 and StrongARM
- Contains fixed-point "package" for efficient floating-point emulation on StrongARM
- Face data base contains 21 individuals (3x7). Each individual is represented by 480 16x16pixel images (10 originals + shifts)

640x480 (320x320)

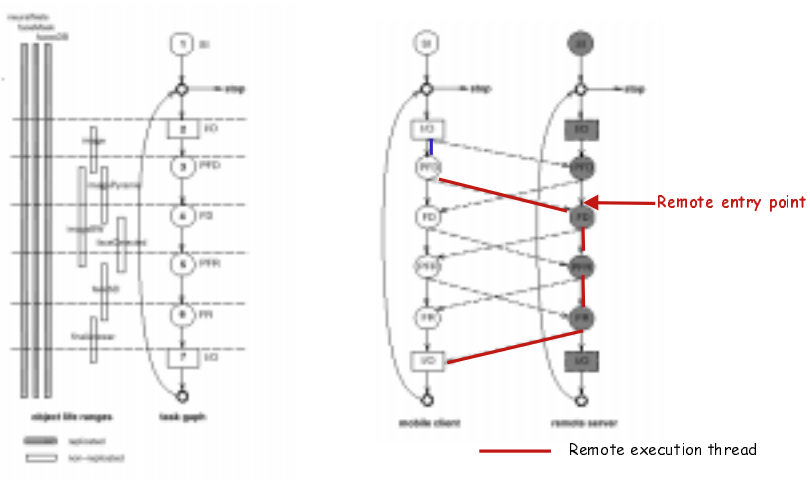


Ulrich Kremer: match distance 192

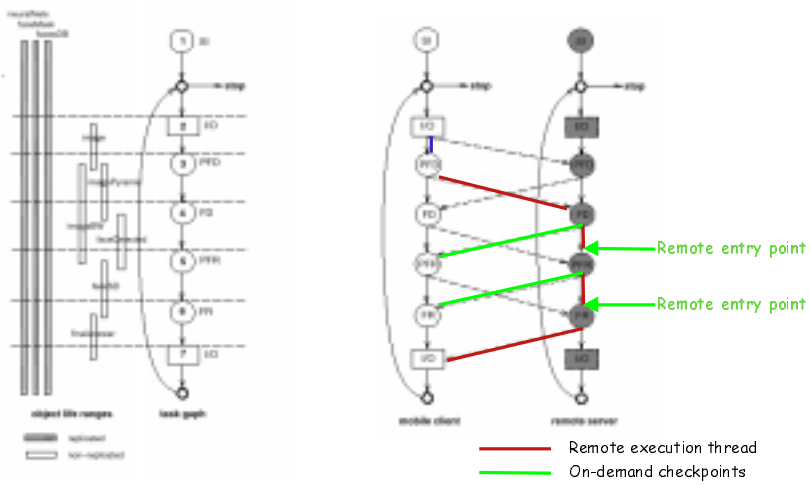
Basic Compilation Strategy



Basic Compilation Stragegy (Cont.)



Face Detection and Recognition System (Cont.)



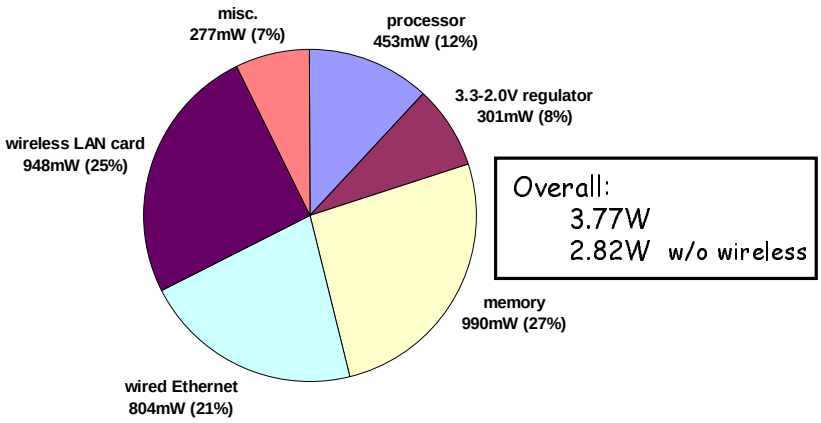
Benefit Analysis: Skiff V2

- 233MHz SA110 + 21285 corelogic, 16KB I-cache, 16KB D-cache (L1)
- 48MHz 32MB SDRAM, 8MB Flash,
- 10Mbps Ethernet, USB, serial port
- Controllers: USB, PC-Card, Ethernet
- Voltage Regulators: 2V, 3.3V, 5V

- ❑ Here: "mobile" Linux box, single user environment
- ❑ Separate power planes: 2V, 3.3V, 5V
- ❑ simple RISC, blocking loads, non-blocking cache-line fills
- ❑ No dynamic voltage or frequency scaling
- ❑ No floating-point unit

Initial Benefit Analysis

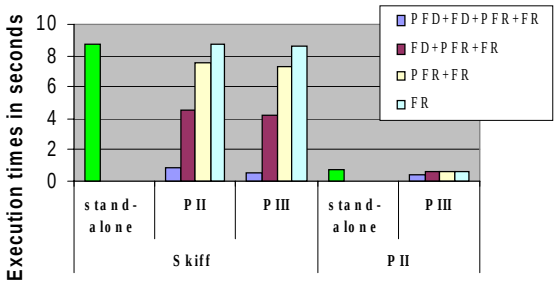
Average power consumption and power distribution for the four main tasks of TourGuide.



Benefit Analysis: Skiff V2

Experiment: **Execution times** using different remote threads indicate potential energy savings.

- communication times based on read/write to files (NFS)
- communication through Compaq's Wireless 11Mbps LAN card
- does not consider hibernation of mobile client machine



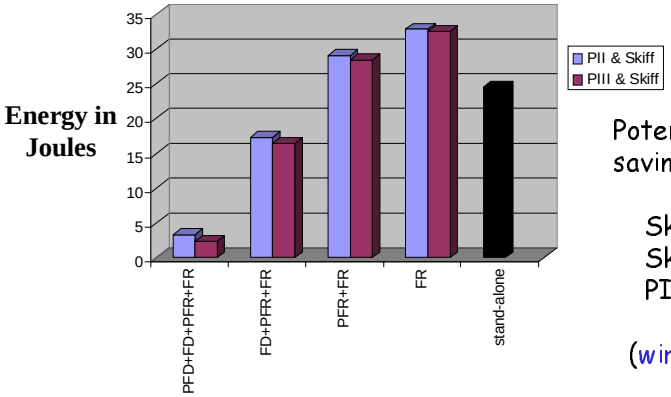
Potential energy savings due to execution time reductions (speed-up):

Skiff & PII : **9.9x**
Skiff & PIII: **13.9x**
PII & PIII: **1.5x**

(no hibernation!)

Benefit Analysis: Skiff V2

Energy consumption of the four remote execution threads without hibernation; stand-alone version without wireless LAN card.



Potential energy savings:

Skiff & PII : **7.4x**
Skiff & PIII: **10.0x**
PII & PIII: **30%**

(wireless always on/off)

Benefit Analysis: iPAQ H3650

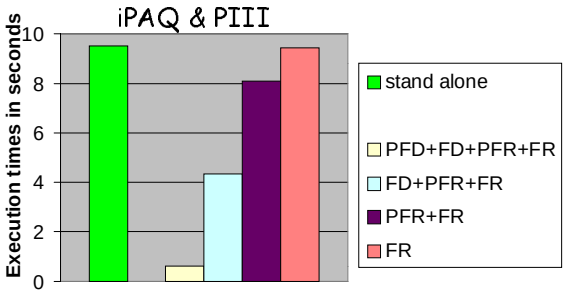
- 206MHz SA1110, dynamic frequency scaling 59MHz - 206MHz 16KB I-cache, 8KB D-cache (L1)
- 66MHz 32MB SDRAM, 16MB Flash
- USB and serial port through cradles; PCMCIA and Compact flash through sleeves; light sensor, microphone, thin film transistor (TFT) color display
- 940mAh lithium polymer battery (12 hours)

- ❑ Here: "mobile" Linux box, single user environment
- ❑ 5V supply voltage, CPU 2V
- ❑ simple RISC, blocking loads, non-blocking cache-line fills
- ❑ Dynamic frequency scaling, but no dynamic voltage scaling
- ❑ No floating-point unit

Benefit Analysis: iPAQ H3650

Experiment: **Execution times** using different remote threads indicate potential energy savings.

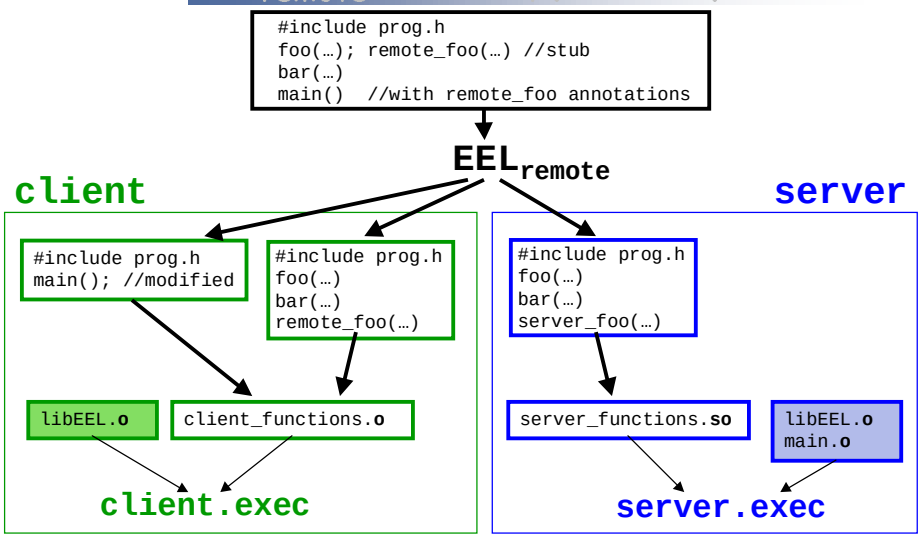
- communication times based on read/write to files (NFS)
- communication through Lucent's Wireless 11Mbps LAN card
- does not consider hibernation of mobile client machine
- **power distribution**: system 1.25W, PC card 0.95W, display off



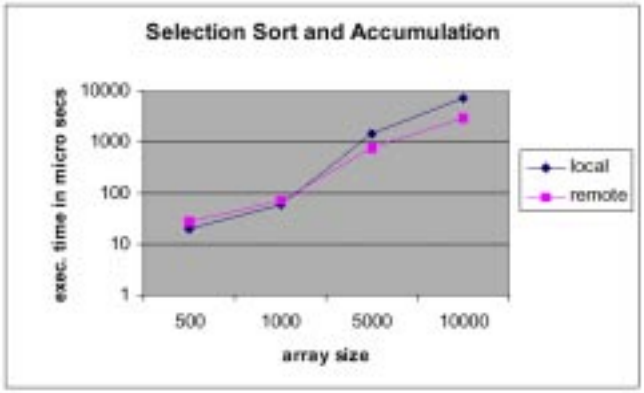
Potential energy savings due to execution time reductions (speed-up):
15.3x
(no hibernation!)

Potential energy savings with wireless card on/off:
8.6x

EEL$_{remote}$ Prototype Compiler



EEL$_{remote}$ Prototype Compiler



Compilers for Power and Energy Management

- ☑ Dynamic Frequency and Voltage Scaling (DFVS)
  - Benefit analysis (SPECfp95)
- ☑ Dynamic Resource Configuration/Hibernation
  - Benefit analysis for 802.11 card (adi, tomcatv, shal)
  - Benefit analysis for disk (mp3, mpeg, and sftp)
- ☑ Remote Task Execution
  - Benefit analysis for StrongARM/Pentium (TourGuide)
- ❑ **QoR Optimizations**
  - Examples
- ❑ Summary and Future Work
- ❑ Wish List for Architects

QoR Optimizations

Goal: Provide power/energy vs. precision tradeoff in a **best effort semantics** programming environment

Safety: User specifies range of semantically acceptable answers

Opportunity: Applications that allow a best effort semantics

Profitability: Probably substantial (10x or more), but not yet verified.

QoR Optimizations

Examples

- use float instead of double precision data types
- allow lower resolution quality in image processing application, or lower sound quality in audio applications,

  Nobel et al. [32]
- network of embedded systems (NES)
  - range of acceptable respond times
  - range of acceptable energy costs
  - limited monetary budget

  Kremer et al. [9], Iftode et al. [10], Gay et al. [28], Zeng et al. [30]

Characteristics of NES

Location-sensitivity

a node is interesting because of its location in addition to the service it provides

Volatility

nodes join and leave at any time

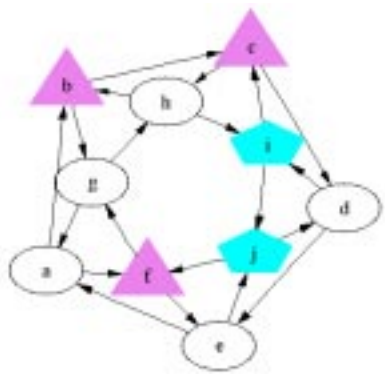
Resource-limitation

execution time, battery life, and monetary budget are limited

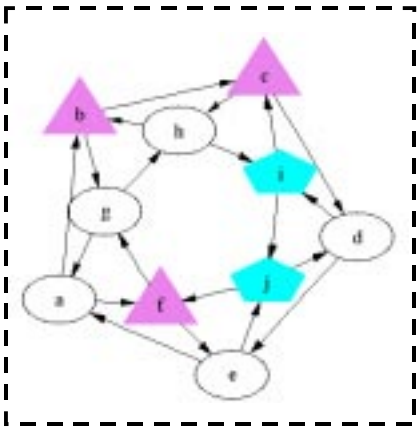
Programming NES : Spatial Views

- nodes are abstracted as
  virtual nodes = (service, location)
  - ⇒ a physical node providing multiple services represents multiple virtual nodes
  - ⇒ a moving physical node represents multiple virtual nodes
- spatial view: a dynamic collection of virtual nodes that share a *service* and a *space*
- operations on spatial views: iteration and selection
  - ⇒ executed under constraints: spatial, time, energy

Programming NES : Spatial Views



Programming NES : Spatial Views



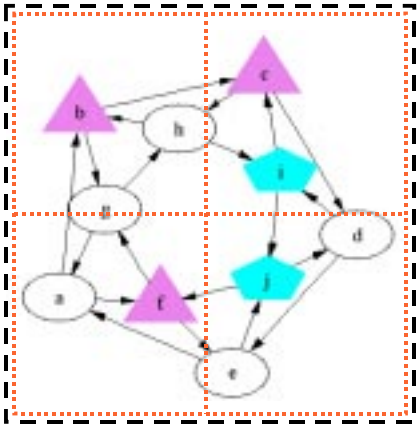
face detection

"smart" camera

other

Spatial View: Cameras on third floor of a building

Programming NES : Spatial Views



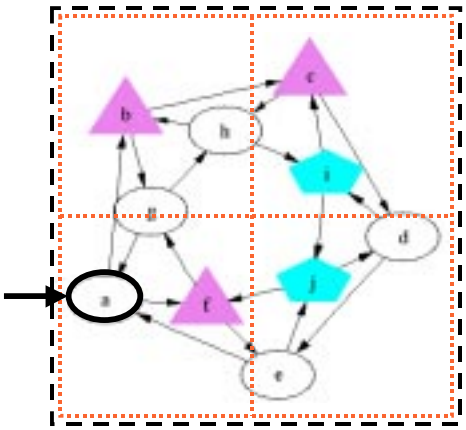
face detection

"smart" camera

other

Spatial View: Cameras on third floor of a building; iterate under density constraint

Programming NES : Spatial Views



- KVM implementation
- execution time
  + with face detection: 23.1 secs
  + without face detection: 10 secs
- routing: gossiping with backtracking (DFS)

Summary and Future Work

- Compiler support for power and energy management is still in its infancy. Exciting "new" area for compiler research.
- Remote Task Execution: one order of magnitude potential energy savings (with limited hibernation).
- Dynamic Frequency and Voltage Scaling: Significant opportunities even in highly optimized codes with no or minimal impact on program performance.
- Resource Management: Initial application: virtual memory on diskless devices and disk management. Both show significant energy savings.
- Initial prototype systems based on SUIF2 infrastructure: EEL_{remote}, EEL_{dfvs}, and EEL_{RM}.

Summary and Future Work

- Develop effective compile-time performance models for power dissipation and execution time.
- Study interactions / trade-offs between different low power and low energy optimizations.
- Explore Quality of Result (QoR) optimizations.
- More experiments, more implementation work.

Wish list for Architects

- Better performance / power / energy prediction models
- Hardware designs that are easier to predict, i.e., make predictability a major design goal
- Hardware designs that allow physical power measurements
- "Direct" control of systems resources by applications (e.g., by compiler generated instructions) through standardized interfaces

More Information

Energy Efficiency and Low-Power Lab

http://www.cs.rutgers.edu/~uli/eel

Bibliography

[1] T. Li and C. Ding, Instruction Balance and Its Relation to Program Energy Consumption. International Workshop on Languages and Compilers for Parallel Computing (LCPC'01), Cumberland Falls, Kentucky, August 2001

[2] A. Parikh, M. Kandemir, N. Vijaykrishnan, and M.J. Irwin, Instruction Scheduling Based on Energy and Performance Constraints, IEEE CS Annual Workshop on VLSI, Orlando, FL, 2000.

[3] M. Kandemir, N. Vijaykrishnan, M.J. Irwin, and H.Y. Kim, Experimental Evaluation of Energy Behavior of Iteration Space Tiling, International Workshop on Languages and Compilers for Parallel Computing (LCPC'01), Cumberland Falls, Kentucky, August 2001.

[4] V. Delaluz, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, Energy-Oriented Compiler Optimizations for Partitioned Memory Architectures, International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01), Atlanta, GA, November 2001.

[5] C-H. Hsu and U. Kremer, Dynamic Voltage and Frequency Scaling for Scientific Applications, International Workshop on Languages and Compilers for Parallel Computing (LCPC'01), Cumberland Falls, Kentucky, August 2001.

Bibliography (Cont.)

[6] C-H. Hsu, U. Kremer, and M. Hsiao, Compiler-Directed Dynamic Frequency/Voltage Scheduling for Energy Reduction in Microprocessors, International Symposium on Low Power Electronics and Design (ISLPED'01), Huntington Beach, CA, August 2001.

[7] J. Hom and U. Kremer, Energy Management of Virtual Memory on Diskless Devices, Workshop on Compilers and Operating Systems for Low Power (COLP'01), Barcelona, Spain, September 2001. To appear as a chapter in Compilers and Operating Systems forLow Power, Kluwer Publishers.

[8] U. Kremer, J. Hicks, and J. Rehg, Compiler-Directed Remote Task Execution for Power Management, Workshop on Compilers and Operating Systems for Low Power (COLP'00), Philadelphia, PA, October 2000.

[9] U. Kremer, L. Iftode, J. Hom, and Y. Ni, Spatial Views: Iterative Spatial Programming for Networks of Embedded Systems, Technical Report, Department of Computer Science, Rutgers University, June 2002.

[10] L. Iftode, A. Kochut, C. Borcea, C. Intanagonwiwat, and U. Kremer, Programming Computers in the Physical World, Proceedings of the 9th Workshop on Future Trends in Distributed Computing Systems (FTDCS 2003), May 2003.

Bibliography (Cont.)

[11] Z. Li, C. Wang, and R. Xu, Computation Offloading to Save Energy on Handheld Devices: A Partitioning Scheme, International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01), Atlanta, GA, November 2001.

[12] L. N. Chakranpani, P. Korkmaz, V.J. Mooney III, K. V. Palem, K. Puttaswamy, W.F. Wong, The Emerging Power Crisis in Embedded Processors: What Can a Poor Compiler Do?, International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01), Atlanta, GA, November 2001.

[13] M. Valluri and L. John, Is Compiling for Performance == Compiling for Power?, Workshop on Interaction between Compilers and Computer Architectures (INTERACT-5), 2001.

[14] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, Application Transformations for Energy and Performance-Aware Device Management, International Conference on Parallel Architectures and Compilation Techniques (PACT'02), Charlottesville, VA, September 2002.

[15] C-H. Hsu and U. Kremer, Compiler-Directed Dynamic Voltage Scaling Based on Program Regions, Technical Report, Department of Computer Science, Rutgers University, November 2001.

Bibliography (Cont.)

[16] C-H. Hsu and U. Kremer, Compiler-Directed Dynamic Voltage Scaling for Memory-Bound Applications, Technical report, Department of Computer Science, Rutgers University, August 2002.

[17] J. Flinn, D. Narayanan, and M. Satyanarayanan, Self-Tuned Remote Execution for Pervasive Computing, Eight Workshop on Hot Topics in Operating Systems (HotOS), Elmau, Germany, May 2001.

[18] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau, Profile-Based Dynamic Voltage Scheduling Using Program Checkpoints, Design Automation and Testing in Europe (DATE), March 2002.

[19] H. Saputra, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, J. Hu, C-H. Hsu, and U. Kremer, Energy-Conscious Compilation Based on Voltage Scaling, ACM SIGPLAN Joint Conference on Languages, Compilers, and Tools for Embedded Systems, and Software and Compilers for Embedded Systems (LCTES/SCOPES'02), Berlin, Germany, June 2002.

[20] D. Mosse, H. Aydin, B. Childers, and R. Melhem, Compiler-Assisted Dynamic Power-Aware Scheduling for Real-Time Applications, Workshop on Compilers and Operating Systems for Low Power (COLP'00), Philadelphia, PA, October 2000.

Bibliography (Cont.)

[21] D. Brooks, V. Tiwari, and M. Martonosi, Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, 27th International Symposium on Computer Architecture (ISCA'2000), June 2000.

[22] V. Tiwari, S. Malik, and A. Wolfe, Instruction Level Power Analysis and Optimization of Software, Journal of VLSI Signal Processing, p.1-18, 1996.

[23] M. Kandemir, N. Vijaykrishnan, M.J. Irwin, and W. Ye, Influence of compiler optimizations on system power, Design Automation Conference (DAC), June 2000.

[24] National Compiler Infrastructure (NCI) Project, overview available online at http://www-suif..stanford.edu/suif/nci/index.html

[25] D. Burger and T. Austin, The SimpleScalar tool set version 2.0, Technical Report, Computer Science Department, University of Wisconsin, June 1997.

[26] C-H. Hsu and U. Kremer, The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction, SIGPLAN Conference on Programming Languages, Design, and Implementation (PLDI'03), June 2003.

Bibliography (Cont.)

[27] J. Palm, H. Lee, A. Diwan, and J. Moss, When to Use a Compilation Service, LCTES'02-SCOPES'02, June 2002.

[28] D Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, The nesC Language: A Holistic Approach to Networked Embedded Systems, PLDI'03, June 2003

[29] R. Xie, M. Martonosi, and S. Malik, Compile-Tie Dynamic Voltage Scaling Settings: Opportunities and Limits, PLDI'03, June 2003.

[30] H. Zeng, C. Ellis, A. Lebeck, and A. Vahdat, ECOSystem: Managing Energy as a First Class Operating System Resouce, ASPLOS X, October 2002.

[31] C-H. Hsu and U. Kremer, The Design, Implementation, and Evaluation of a CompilerAlgorithm for CPU Energy Reduction, PLDI'03, June 2003.

[32] B. Noble, M. Satyanarayanan, D. Narayanan, J. Tilton, J. Flinn, and K. Walker, Agile Application-Aware Adaptation for Mobility, SOSP'97, October 1997