Event-Driven Scalability Predictors: Improving Energy-Efficiency under Hard Performance Constraints on Multiprocessors

Dimitris Nikolopoulos
Department of Computer Science
Virginia Tech

Joint work with:
Matt Curtis-Maury
James Dzierwa
Christos Antonopoulos

How to make multi-context architectures more energy-friendly without compromising HPC

Multi-core/multi-threaded processors

Energy awareness

Concurrency control

How to make multi-context architectures more energy-friendly without compromising HPC

Scalability predictors
Continuous hardware monitors

Multi-Core and Multi-Threaded Architectures

Diminishing returns from ILP (ideal pipeline!)

$T_{\text{ILP}} \leq T_{\text{RIM}}$

$T_{\text{RIM}} \leq \frac{C}{R}$

$1 - \frac{1}{R} \leq 1 - \frac{C}{R}$
Latency and parallelism considerations

<table>
<thead>
<tr>
<th>System Component</th>
<th>Annual Increase</th>
<th>2005</th>
<th>2010</th>
<th>2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP performance</td>
<td>59%</td>
<td>4 GFLOPS</td>
<td>33 GFLOPS</td>
<td>3.3 TFLOPS</td>
</tr>
<tr>
<td>bus bandwidth</td>
<td>23%</td>
<td>.25 words/flop</td>
<td>.11 words/flop</td>
<td>.086 words/flop</td>
</tr>
<tr>
<td>memory latency</td>
<td>-6%</td>
<td>280 flops</td>
<td>1600 flops</td>
<td>94,000 flops</td>
</tr>
</tbody>
</table>

Need thousands of threads to overlap latencies, at a large cost for power (and performance?)

Representative power consumption

SMT and multi-core power over 100W. Power density comparable to nuclear reactors.

Concurrency control

Concurrency as a knob

Energy-aware

Performance

Concurrency -power trade-off revisited (HPC)

Power increases linearly with concurrency.

Performance gains diminish with higher concurrency.

Is there an "optimal" concurrency (sweet-spot)?
Exploiting the trade-off: program phases

Code regions with uniform characteristics

Characteristics of interest: IPC, memory access intensity

Our case: scalability

Exploiting the trade-off: polymorphic threading

Code regions react differently to SMT

Characteristics of interest: ILP, memory access intensity

Our case: educated choice between precomputation, TLP, and sequential execution

Optimality

Maximum performance at minimum power

Exponential space for searching $O(n_{phases}^{threads})$

Search at runtime needs heuristics

Related work

DVFS (stretching) [Penn State, Cornell, DAC'04, ISPASS'05, IPDPS'06]

DVFS during idle time at synchronization points [NCSU, VT SC'05, PPoPP'05]

Direct search methods [Cornell HPCA'06]
Scalability predictors

Predict optimal concurrency and system configuration using real-time feedback

Predict scaling of phases with varying operating points (#threads, #cores, #CPUs)

Use hardware event counters to pin-point scalability bottlenecks

IPC scalability predictors

\[ \text{IPC}_t(ES_{\text{obs}}) \]

- \( t \) represent target configuration
- \( f \) non-linear approximation function
- \( ES \) are samples of event rates with effect on scalability
- IPC denotes useful computation, excluding spinning.

IPC scalability predictors - Steps

**Static training**
- Need \( f() \)
- Find event sets
- Fit models to samples
- Find \( f \) co-efficients

**Runtime**
- Identify program phases
- Sample event rates
- Predict and adapt

IPC scalability predictors - Steps

**Static training**
- Need \( f() \)
- Find event sets
- Sample event rates
- Fit models to samples
- Find \( f \) co-efficients

**Runtime**
- Identify program phases
- Sample event rates
- Predict and adapt
IPC scalability model – event sets

\[ \text{CPI} = \sum_{i=1}^{n} \text{f}_{\text{CPI event}} \]

\[ \text{CPI event} = \text{counter instruction cycles event} \]

Available at runtime through counters

Available through training and linear regression

IPC scalability model – event set training summary

Standard benchmark set (NAS, SPEC-OMP)

Collect event counts for 22 major processor components

Run linear regression to estimate cycles/event.

Keep dominant events (maximum feasible to count simultaneously with available hardware counters)

IPC scalability predictors - Steps

Static training

Need \( f() \)

Find event sets

Fit models to samples

Runtime

Identify program phases

Sample event rates

Predict and adapt

Models of event rates

Polynomial model (INSRET, UOPQWR, BRRET)

Two-phase exponential model (L2M, TCM, BUS_ACC)

\[ f(t, e_1, e_2) = w_1 t e_1^2 + w_2 t e_1 e_2 + w_3 t e_2 + w_4 t e_1^2 e_2 \]

\[ f(t, e_1, e_2) = e^{t \lambda_1} - e^{t \lambda_2} + e^{t \lambda_3} \]

INSRET

BUS_ACC

L2M

TCM

UOPQWR

BRRET-1
Scalability prediction – layered architecture

Intel SMP
- INSRET
- BUS_ACC
- L2M
- TCM
- UOPQWR
- BRRET-1

Intel SMT
- INSRET
- BRRET-2
- L2M
- TCM
- UOPQWR
- BRRET-1

Models of event rates

Polynomial model (INSRET, UOPQWR, BRRET)
\[ f(t, e_1, e_2) = w_1 \cdot e_1 \cdot e_2 + w_2 \cdot e_1 + w_3 \cdot e_2 \]

Two-phase exponential model (L2M, TCM, BUS_ACC)
\[ f(t, e_1, e_2) = e_1^\lambda \cdot e_2^\mu - e_1^\lambda \cdot e_2^\mu \]

Train with two programs (MM5, NAS UA):
- Find best fit for each dominant event

IPC scalability model – event sets

\[ CPI = \sum_{i=1}^{n} \text{CPI}_{\text{event}_i} \]

Select events with maximum statistical effect on performance and scalability

Available at runtime through counters

Available through training and linear regression

IPC scalability predictors - Steps

Static training
- Find event sets
- Fit models to samples
- Find f co-efficients

Runtime
- Identify program phases
- Sample event rates
- Predict and adapt
Dominant phases – runtime BBV analysis

Instrument //loop entry points

Set of distinct points defines phase

Maximal cardinality repeated set defines period

IPC scalability predictors - Steps

Static training
Need f()
Find event sets
Fit models to samples
Find f co-efficients

Runtime
Identify program phases
Sample event rates
Predict and adapt

PACMAN

Accurate, fine-grain performance monitoring

Dynamic partitioning of HPM register file between threads

Software identification and elimination of spinning events
IPC scalability prediction - Steps

∀ phase {
∀ layer L, nl ≥ L ≥ 1 of parallelism from the innermost to the outermost {
sample event rates at max max/2 //;
predict // to optimize;
adapt on next instance; }}

execute phases for analysis; // Max 3 outer iterations in NAS/SPEComp
∀ layer L, nl ≥ L ≥ 1 of parallelism from the innermost to the outermost {
//Max 3 layers in current architectures
config = c_0,c_1,...,c_l,p_1,...,p_l; ES
= ES_{config};
execline phase with configuration config
and record IPC, ES;
predict number of processing elements p_L
to use at layer L for maximizing IPC;}

IPC scalability prediction - Steps

∀ phase {
∀ layer L, nl ≥ L ≥ 1 of parallelism from the innermost to the outermost {
sample event rates at max max/2 //;
predict // to optimize;
adapt on next instance; }}

IPC scalability predictors - Steps

Static training
Need f()
Find event sets
Fit models to samples
Find f co-efficients

Runtime
Identify program phases
Sample event rates
Predict and adapt
SMT configuration predictors

Pruned tree at < 4 levels, confidence >= 80%
Adding fpuops/cycle increases confidence to >93%

Results – experiments with real silicon

Initial platform:
Dell PowerEdge 6650
(8 threads, 4 dual-SMT Xeon @1.4 Ghz, 8-KB L1, 256KB L2, 512KB L3, 1 GB RAM)
perfctr interface (custom kernel) plus PACMAN

New platform:
IBM OpenPower 720
(16 threads, dual-SMT-core Power5 1.65 Ghz, 32-KB L1, 1.92 MB L2, 36 MB L3, 8 GB RAM)
oprofile plus extensions.

Results – Power modelling and measurement

Power model
Adapted from Martonosi and Isci [MICRO'03].
Uses hardware counters and area ratios to estimate nominal power consumption in major components
Used for energy prediction during adaptation and fine-grain measurements (err <= 5%)

Physical infrastructure
Yokogawa WT230 multimeter
granularity: 100ms

Results – Opportunities on IBM system
Results – Energy-performance gains on IBM (NAS, SpecOMP)

Results – configuration prediction accuracy on IBM

Summary

Dynamic discovery and exploitation of power/performance adaptation opportunities

More energy-efficient multicore processors

Further research: inter-phase interference

More info: ICS’06, HP-PAC’06, QEST’06

http://www.cs.vt.edu/~dsn

Questions?

32 for $1M

32 for $2K or 13K for 1M
Results – IPC prediction accuracy

- IPC error distribution - Mean 12.6%

- Histogram of samples with good IPC error

Energy considerations

- Power increases fast in the desktop/server world, somewhat slower in the laptop world...