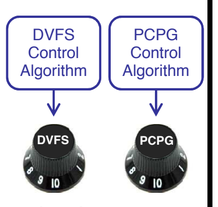
Crank It Up or Dial It Down

Coordinated Multiprocessor Frequency and Folding Control

Augusto Vega[1], **Alper Buyuktosunoglu**[1], **Heather Hanson**[2], **Pradip Bose**[1], **Srinivasan Ramani**[2]
[1]IBM T. J. Watson Research Center, [2]IBM Systems & Technology Group

Motivation

- Modern multi-core systems incorporate support for dynamic power management with multiple actuators
  - ✓ Dynamic voltage and frequency scaling (DVFS)
  - ✓ Core folding
  - ✓ Per-core power gating (PCPG)



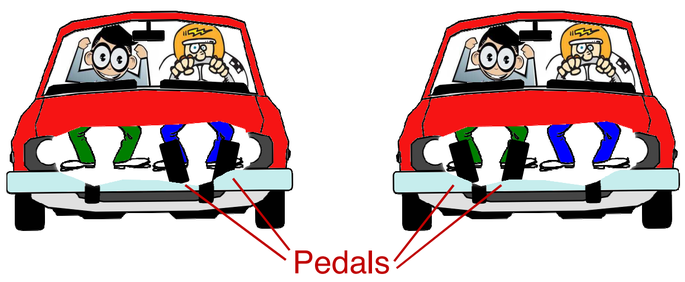
- Algorithms that control these actuators evolved independently
  - Their independent operation can result in conflicting decisions that can lead to undesirable effects on performance and power

We argue in favor of a coordinated control of these actuators to avoid such potential conflicts in dynamic power management

Sometimes Coordination Matters

What is more appropriate?

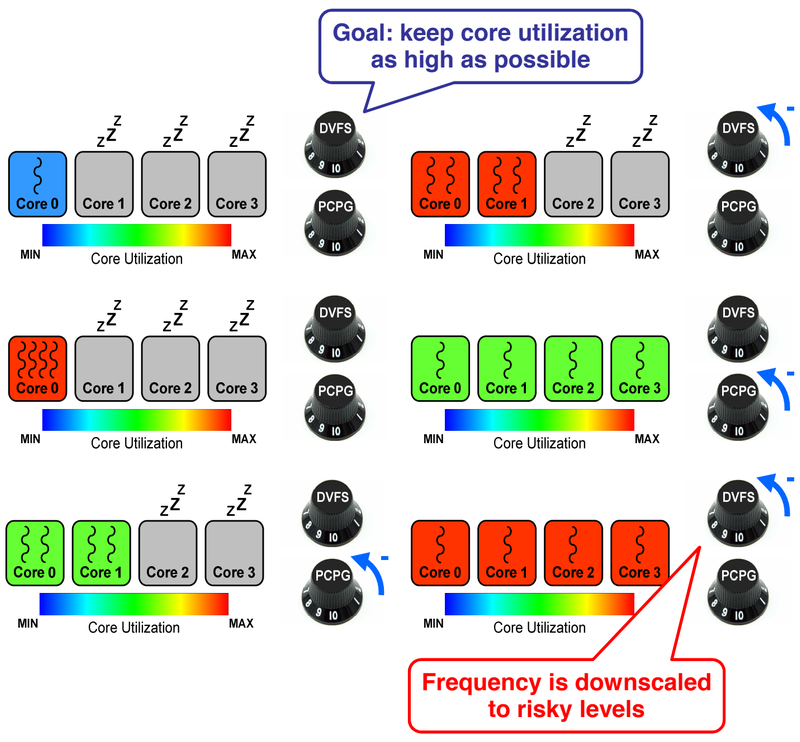
This? Or this?



Coordinated Control **Decoupled Control**

Decoupled Power Management

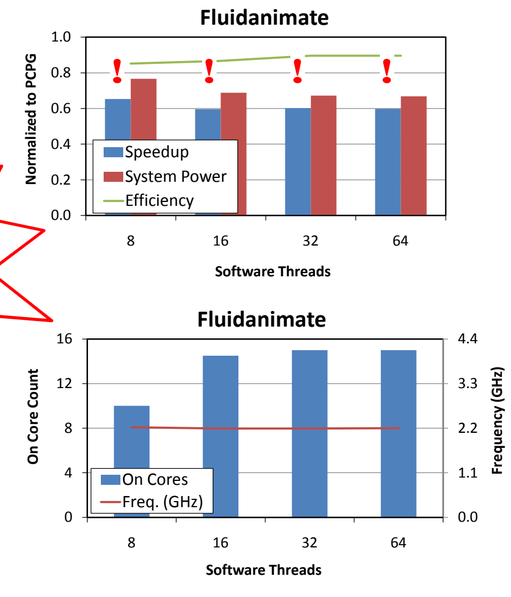
- Power control techniques have evolved in a decoupled manner
  - ✓ Less complexity
  - ✓ Different timescale granularities
- Their independent actuation can lead to conflicting decisions that jeopardize system power-performance efficiency



A robust coordination protocol is necessary in orchestrating the power management functions

When the Decoupled Approach Does Not Work

- **Fluidanimate benchmark**
  - ✓ Multi-threaded application from the PARSEC suite
  - ✓ Fluid motion physics simulation for *real-time* animation purposes
- We evaluate two scenarios
  1. Only PCPG ← baseline
  2. Decoupled PCPG+DVFS



Performance is hurt at unacceptable levels!

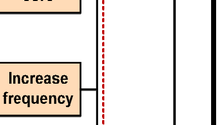
Performance *and* Throughput Awareness

OUR IDEA

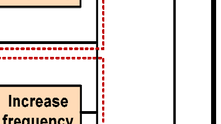
Operate power management *knobs* depending on if an application's current execution phase is single-thread performance or throughput bound

We define three scenarios:

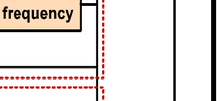
A. **Symptom:** All turned-on cores are highly utilized
Diagnosis: Application likely bound by throughput
Action: Turn cores on



B. **Symptom:** Some turned-on cores are highly utilized
Diagnosis: Application likely bound by single-thread performance
Action: Increase frequency



C. **Symptom:** All turned-on cores are low utilized or idle
Diagnosis: Enough "slack" to **safely** decrease frequency or turn cores off
Action: Decrease frequency or turn cores off



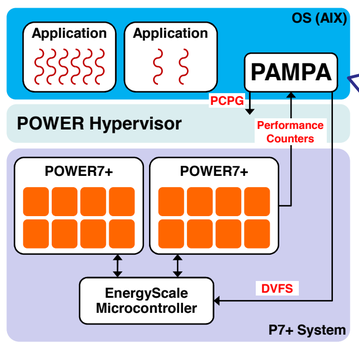
The PAMPA Algorithm

POWER AWARE MANAGEMENT OF PROCESSOR ACTUATORS ALGORITHM

Only enabled during stable execution phases

Methodology

- IBM POWER7+ system with AIX O/S
  - Two 8-core processors (16 cores total)



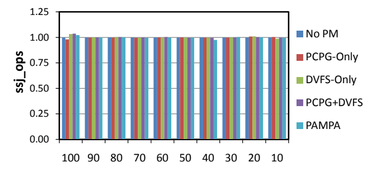
PAMPA operates at O/S level
- ✓ Estimates core-level utilization at run-time
- ✓ Actuates core folding and PCPG at O/S level
- ✓ Connects to the *EnergyScale* µcontroller to adjust processor-level voltage and frequency

- Single-thread performance- and throughput-oriented benchmarks:
  - **SPECpower_ssj2008**
  - **Four PARSEC applications**
  - **Two HPC applications**
- Five scenarios evaluated:
  1. **No power management**
  2. **PCPG Only**
  3. **DVFS Only**
  4. **PCPG + DVFS** ← baseline
  5. **PAMPA** ← our approach

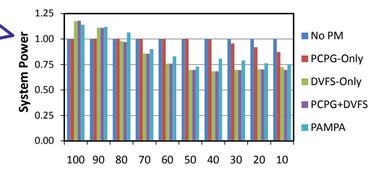
Evaluation: SPECpower Benchmark

- **SPECpower benchmark**
  - ✓ JVMs represent "warehouses" that receive requests from clients
  - ✓ Load levels: *100% .. 10%*
  - ✓ Throughput metric: SPECpower operations per second (**ssj_ops**)

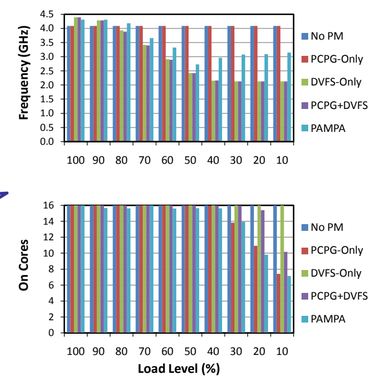
PAMPA preserves throughput



PAMPA slightly increases system power



- For this benchmark:
  - ✓ PCPG+DVFS operates properly
  - ✓ PAMPA exhibits efficiency levels close to PCPG+DVFS

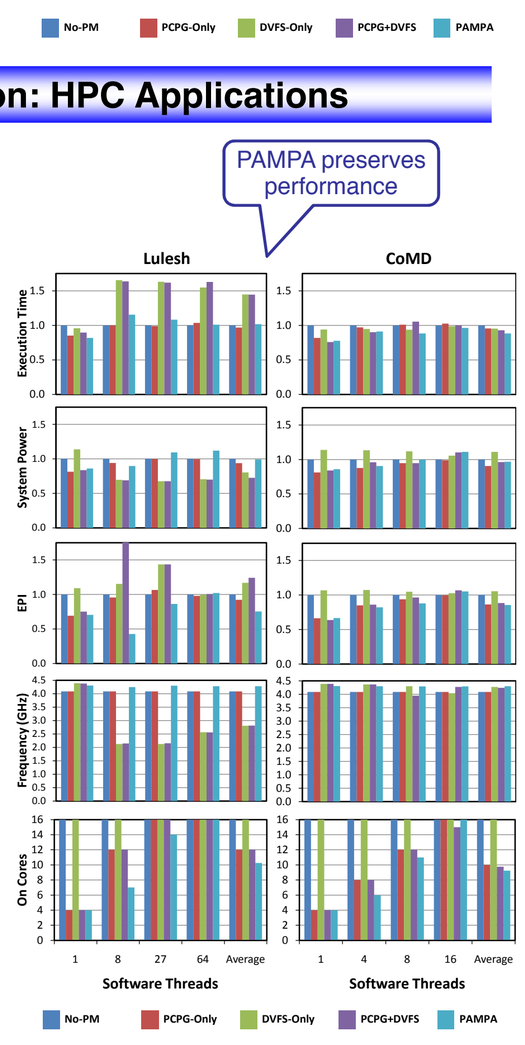


PAMPA aggressively actuates DVFS and PCPG

Evaluation: PARSEC Benchmarks

- **PARSEC benchmarks**
  - ✓ Multi-threaded
  - ✓ CPU-intensive
  - ✓ Sensitive to frequency scaling
- For these benchmarks:
  - ✓ PCPG+DVFS degrades performance excessively
  - ✓ PAMPA preserves performance
    - ✓ Prevents frequency downscaling

PAMPA preserves performance



PAMPA properly actuates DVFS and PCPG

Evaluation: HPC Applications

- **Lulesh and CoMD**
  - ✓ **Lulesh:** Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics
  - ✓ **CoMD:** classical molecular dynamics algorithm
  - ✓ Message Passing Interface (MPI)
  - ✓ CPU-intensive
  - ✓ Sensitive to frequency scaling
- For Lulesh:
  - ✓ PCPG+DVFS degrades performance excessively
  - ✓ PAMPA preserves performance
    - ✓ Prevents frequency downscaling

PAMPA preserves performance

PAMPA properly actuates DVFS and PCPG

Summary

- Modern systems incorporate multiple actuators for dynamic power management
- Algorithms that control these actuators have evolved independently
  - Their independent operation can result in conflicting decisions that can lead to undesirable effects on performance and power
  - We propose a coordinated, in-order actuation of the power knobs
- **PAMPA: P**ower-**A**ware **M**anagement of **P**rocessor **A**ctuators
  - Dynamically "detects" if an application is single-thread performance or throughput bound and actuates the knobs accordingly

Why should I use PAMPA?

- ✓ Exhibits power-performance efficiencies comparable to the most aggressive, decoupled PCPG+DVFS approach
- ✓ Avoids excessive performance degradation in cases where PCPG+DVFS results in conflicting decisions