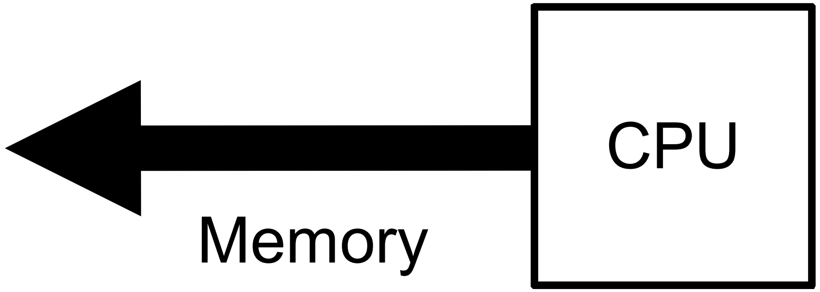
Resource-Conscious Scheduling for Energy Efficiency on Multicore Processors

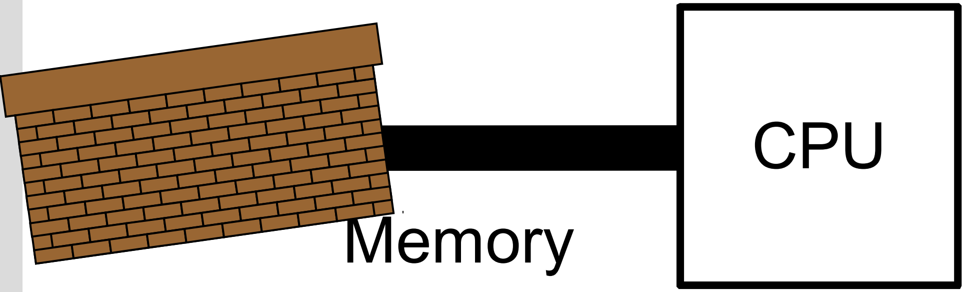
Andreas Merkel, Jan Stoess, Frank Bellosa

System Architecture Group

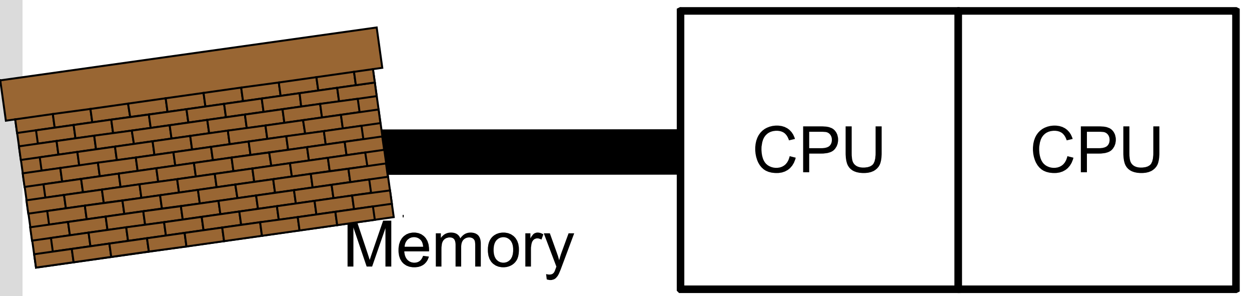
Memory Contention – a Problem on Multicores



Memory Contention – a Problem on Multicores

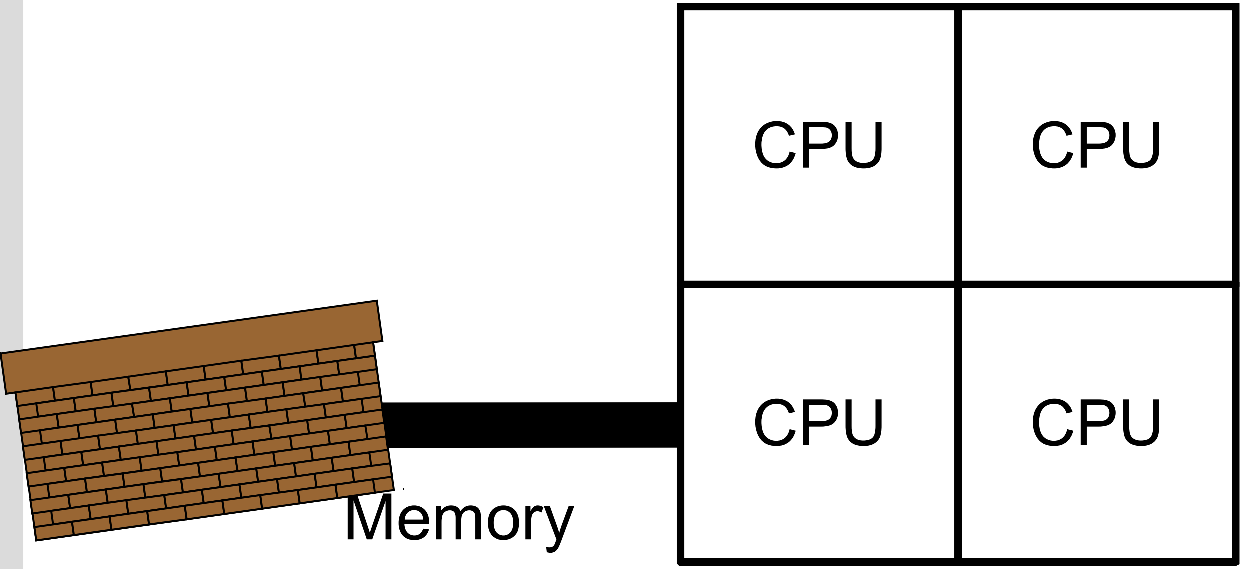


Memory Contention – a Problem on Multicores

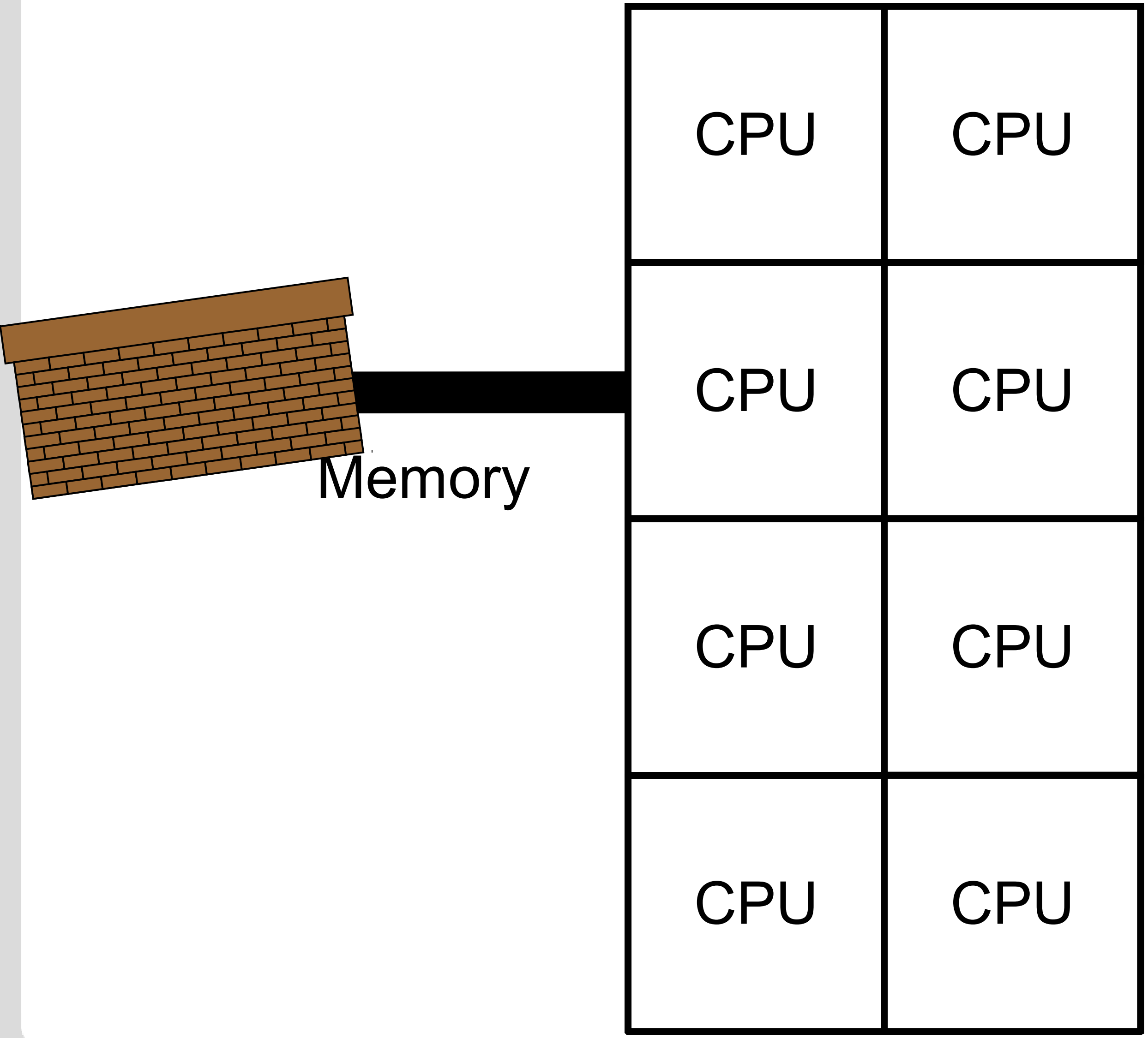


Memory Contention – a Problem on Multicores

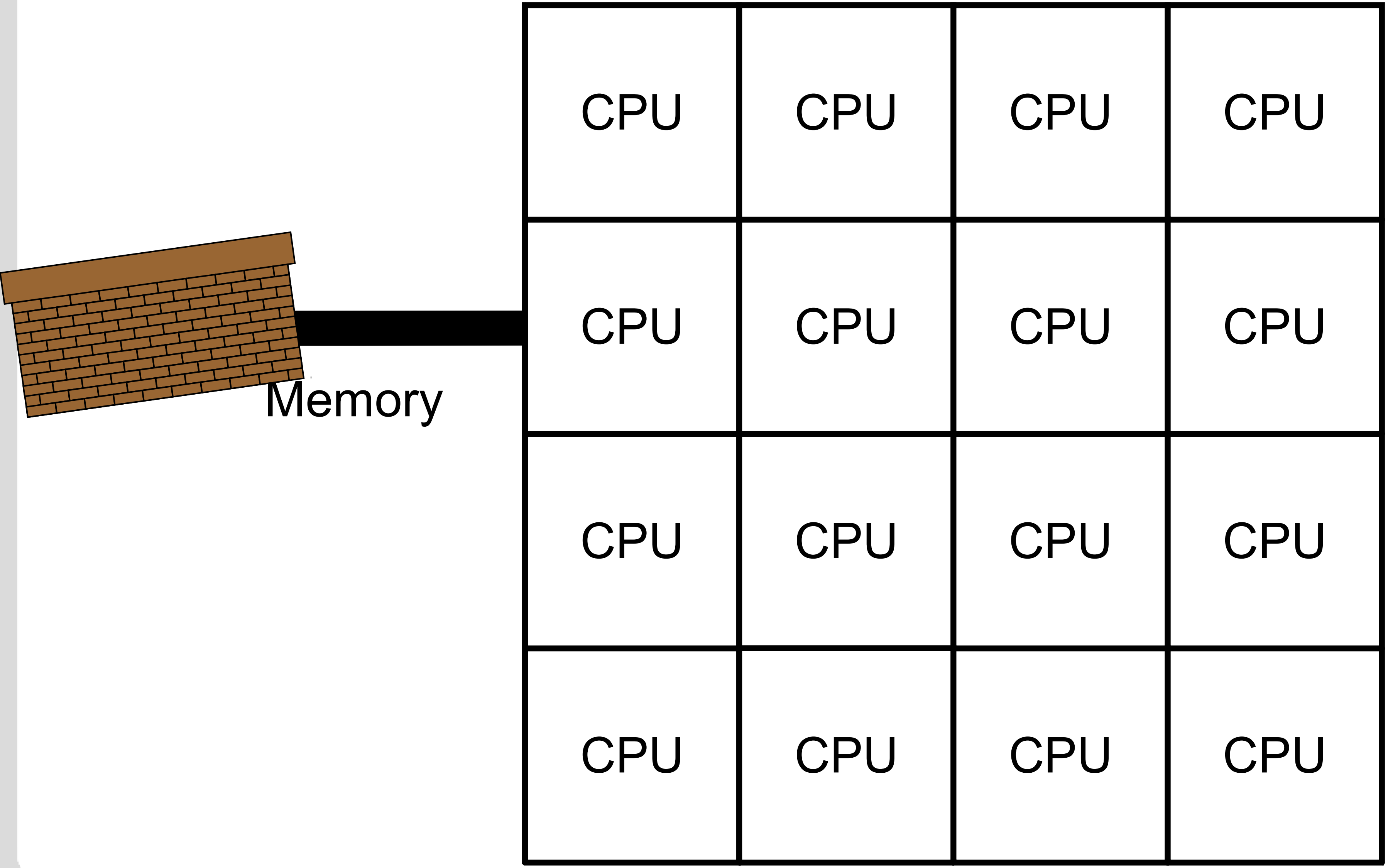
Karlsruhe Institute of Technology



Memory Contention – a Problem on Multicores

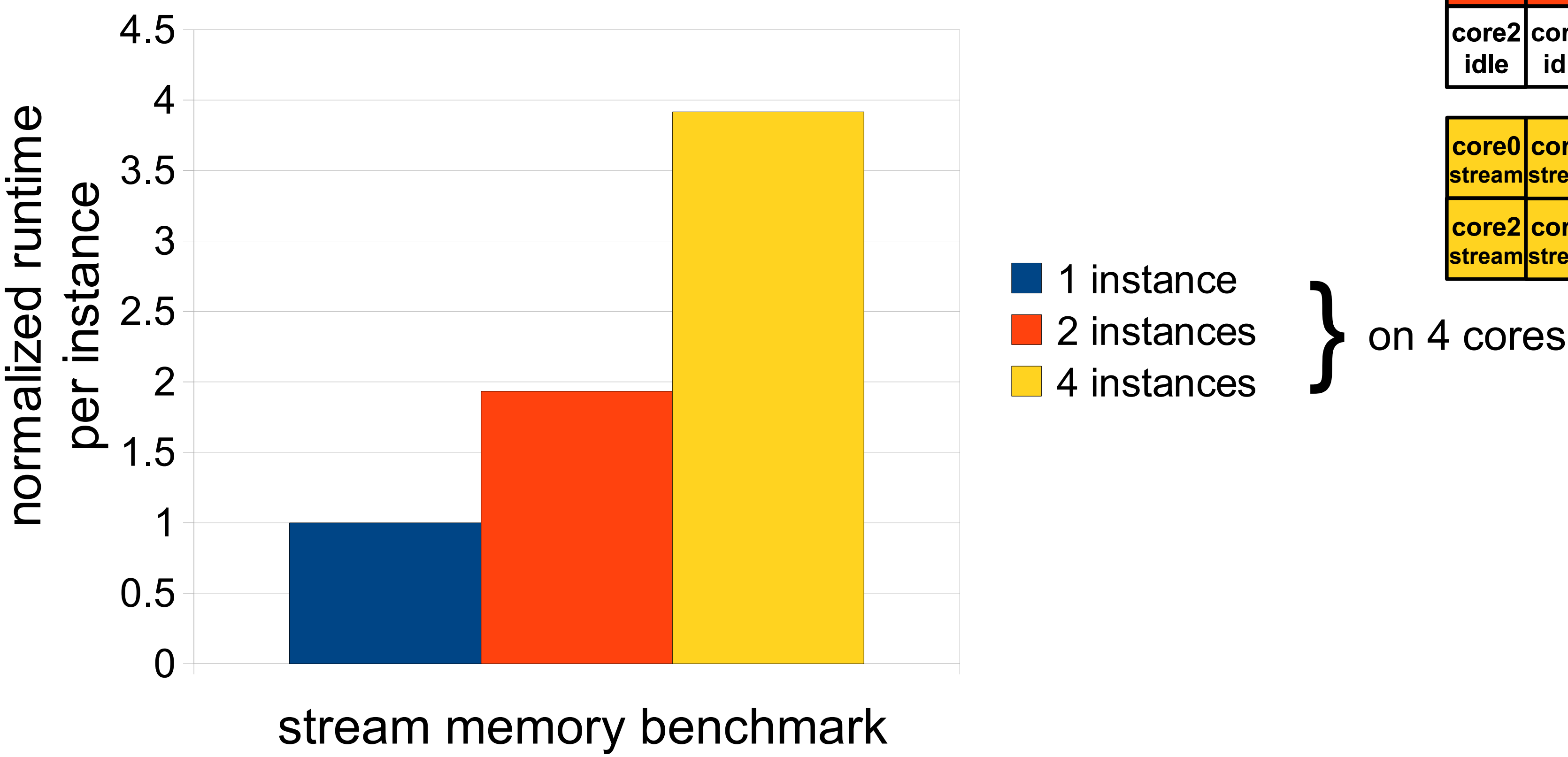


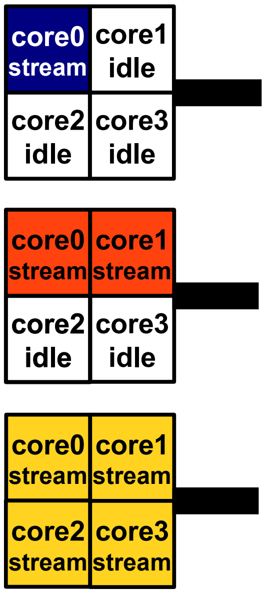
Memory Contention – a Problem on Multicores



Memory Contention Intel Core2 Quad

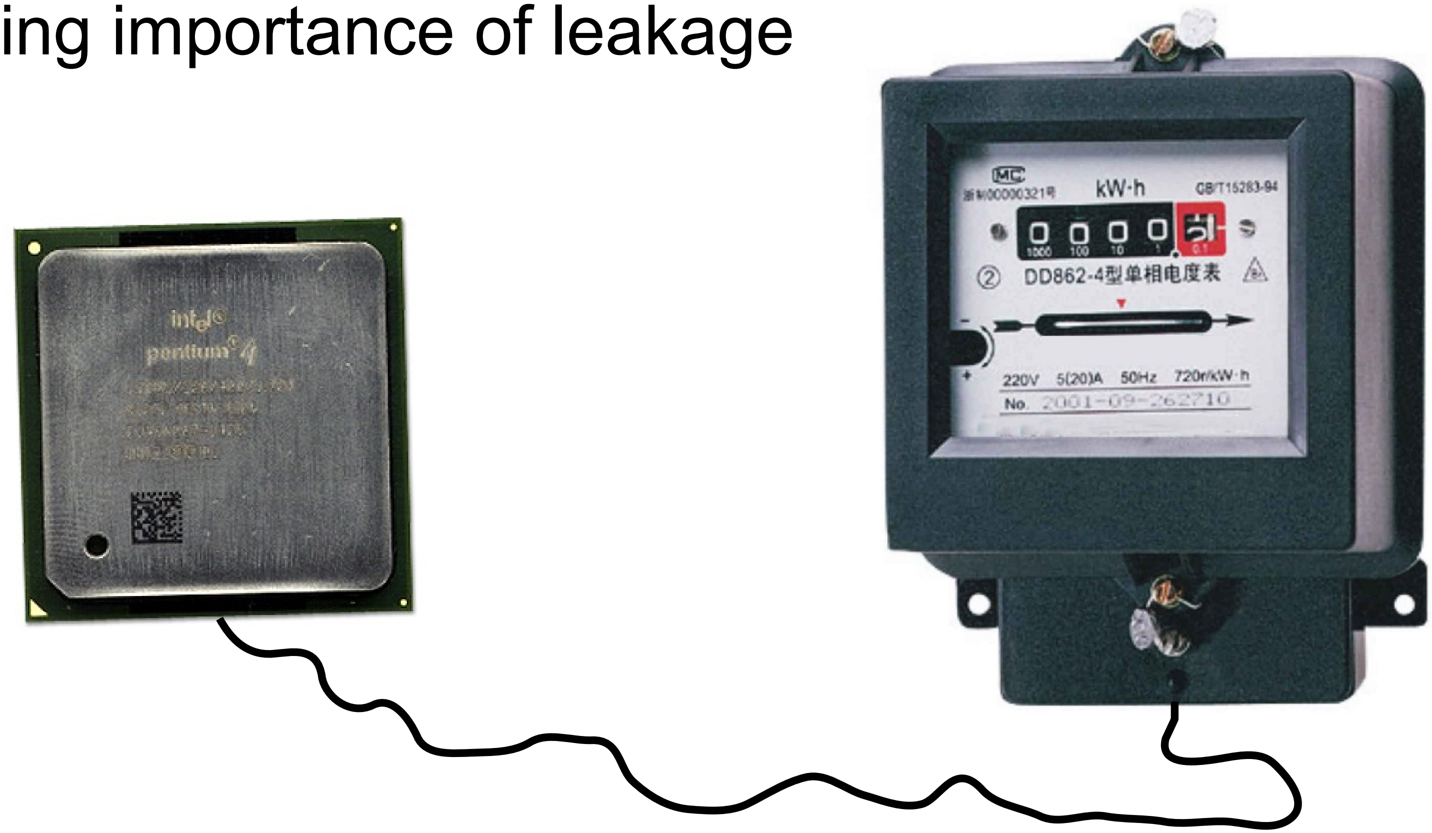
- Bottleneck: memory bus
- Stall cycles, increased runtime





Impact of Resource Contention on Energy Efficiency

- Longer time to halt
- More static power
- Increasing importance of leakage



Achieving Energy Efficiency by Scheduling

- Scheduler decides
  - When
  - Where
  - In which combination
  - At which frequency setting

  to execute tasks.

- **What ist the most energy-efficient schedule?**

Achieving Energy Efficiency via Co-Scheduling

- Combination of tasks running together determines performance and energy efficiency
- Memory-bound + memory-bound: low energy efficiency
- Avoid memory bottleneck by combining memory-bound with compute bound tasks

→ Co-schedule tasks with different characteristics

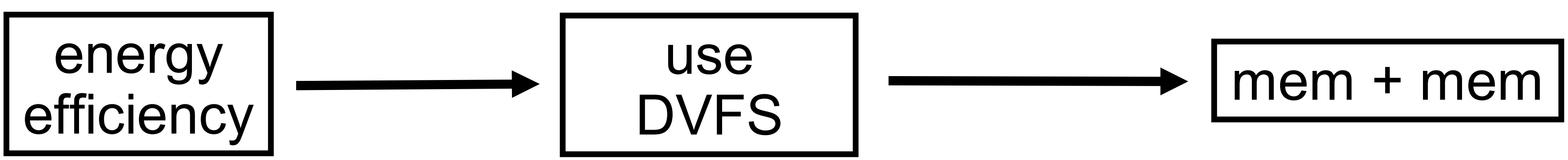
energy efficiency → avoid contention → mem + comp

Achieving Energy Efficiency via DVFS

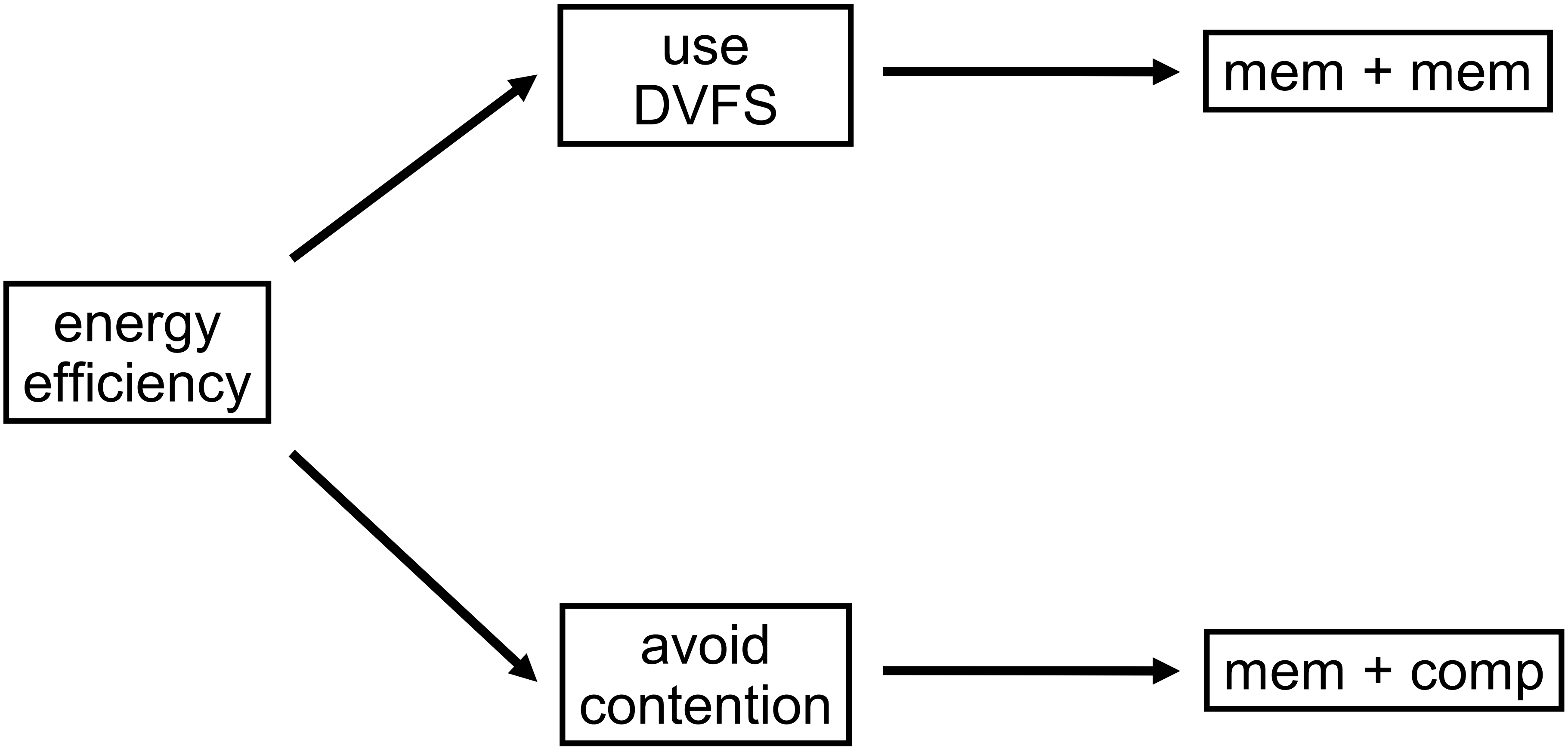
Karlsruhe Institute of Technology

- DVFS: Dynamic Voltage and Frequency Scaling
- Adapt processor frequency and voltage to task characteristics
  - Memory-bound tasks: low frequency/voltage
  - Compute-bound tasks: high frequency/voltage
- Multicore hardware limits options for frequency/voltage selection
  - Often shared frequency/voltage domains

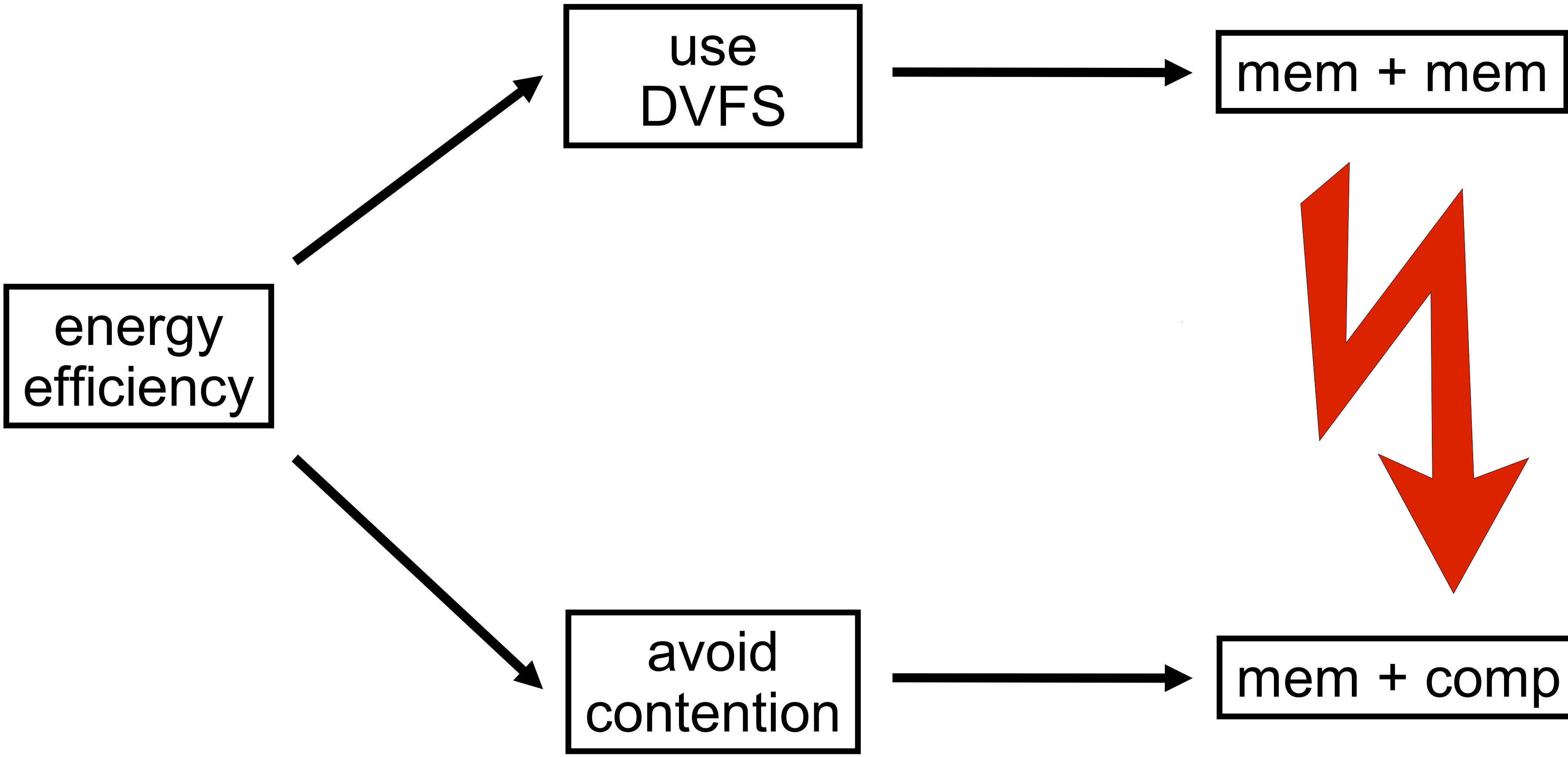
  ➔ Co-schedule similar tasks to select common best frequency and voltage



Achieving Energy Efficiency



Achieving Energy Efficiency

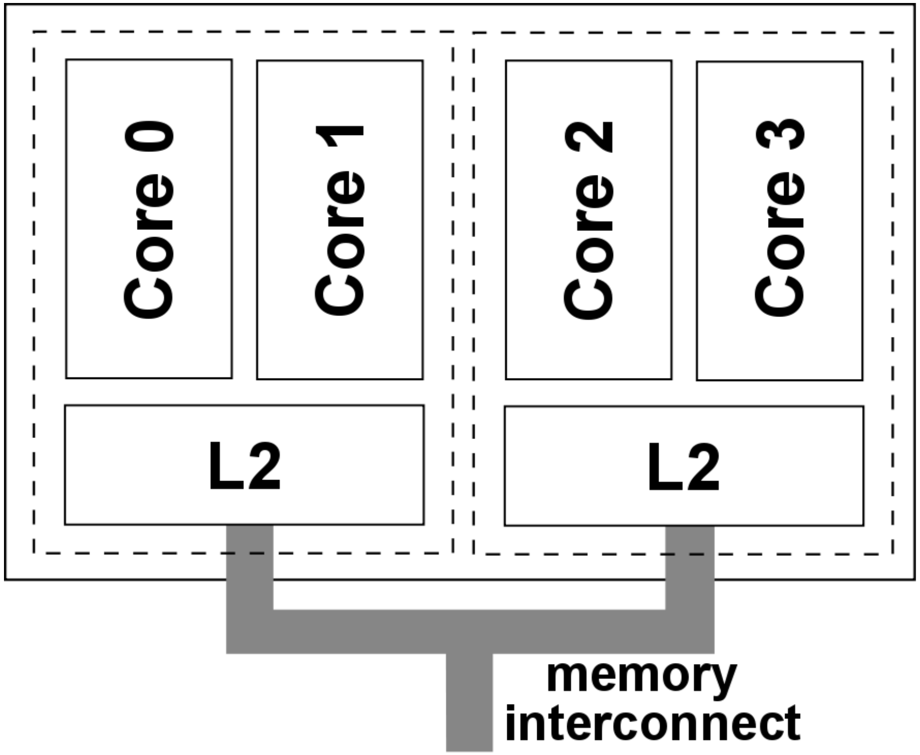


Outline

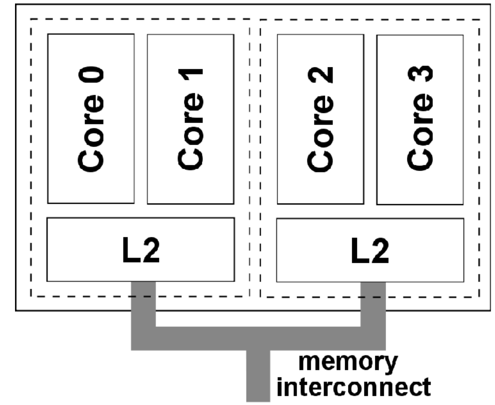
- Analysis
  - Resource contention
  - Shared frequency/voltage domains
- Resource-conscious scheduling for energy efficiency
  - OS task scheduling
  - VM scheduling
  - Frequency selection
- Evaluation
  - Reduction of resource contention
  - Increase in energy efficiency by 10 to 20%

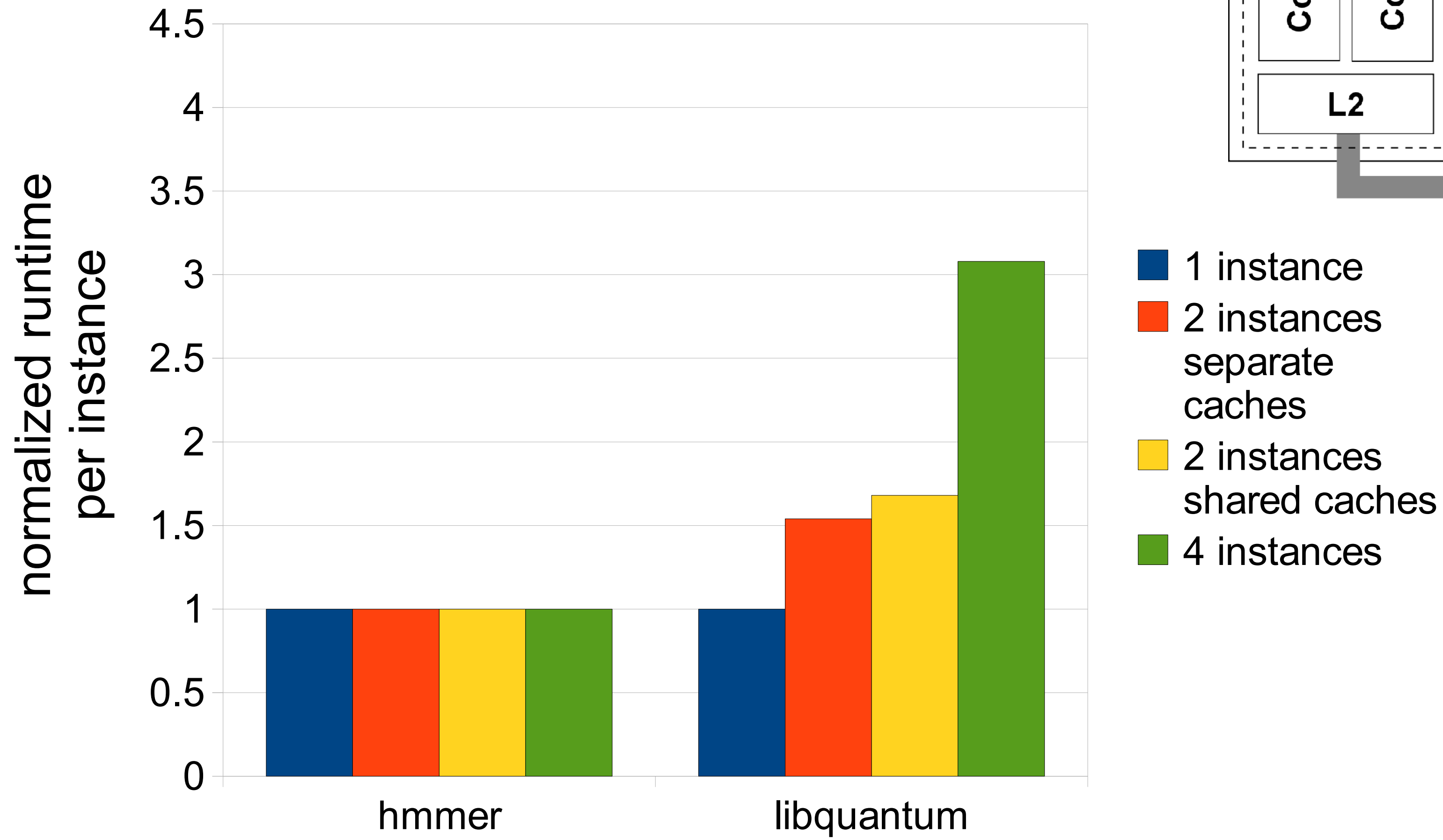
Analysis of Resource Contention on the Intel Core2 Quad Q6600

- Contention for shared resources reduces energy efficiency
  - Shared L2 caches (two cores)
  - Shared memory interconnect (four cores)

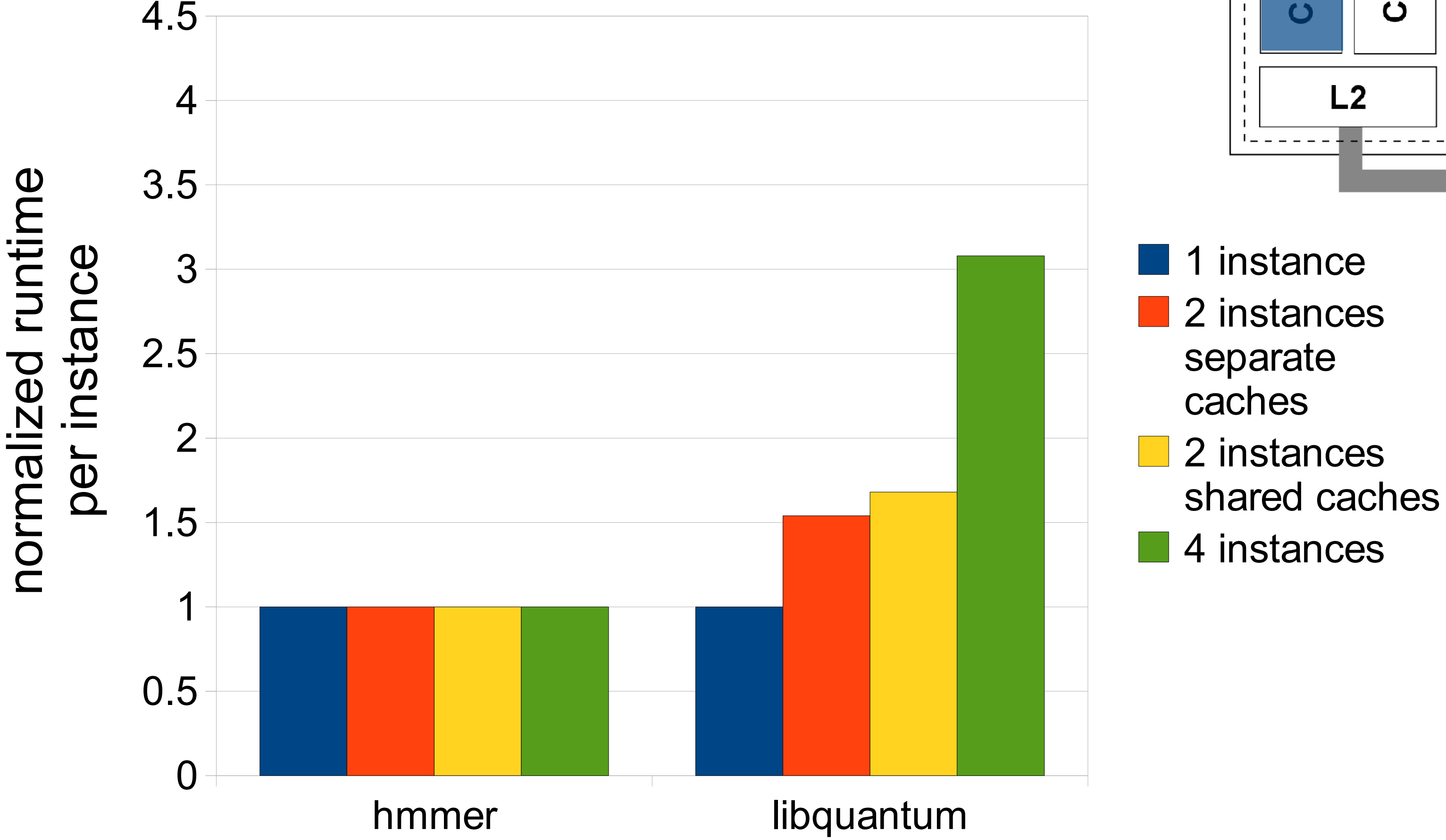


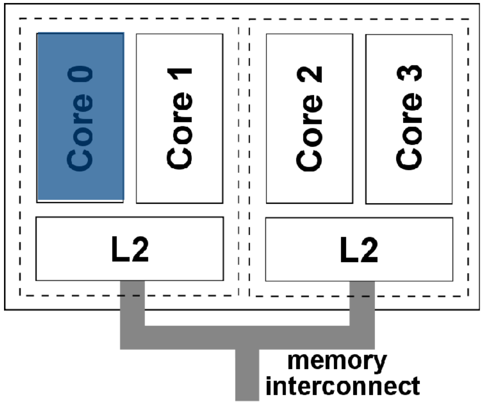
Resource Contention SPEC CPU 2006



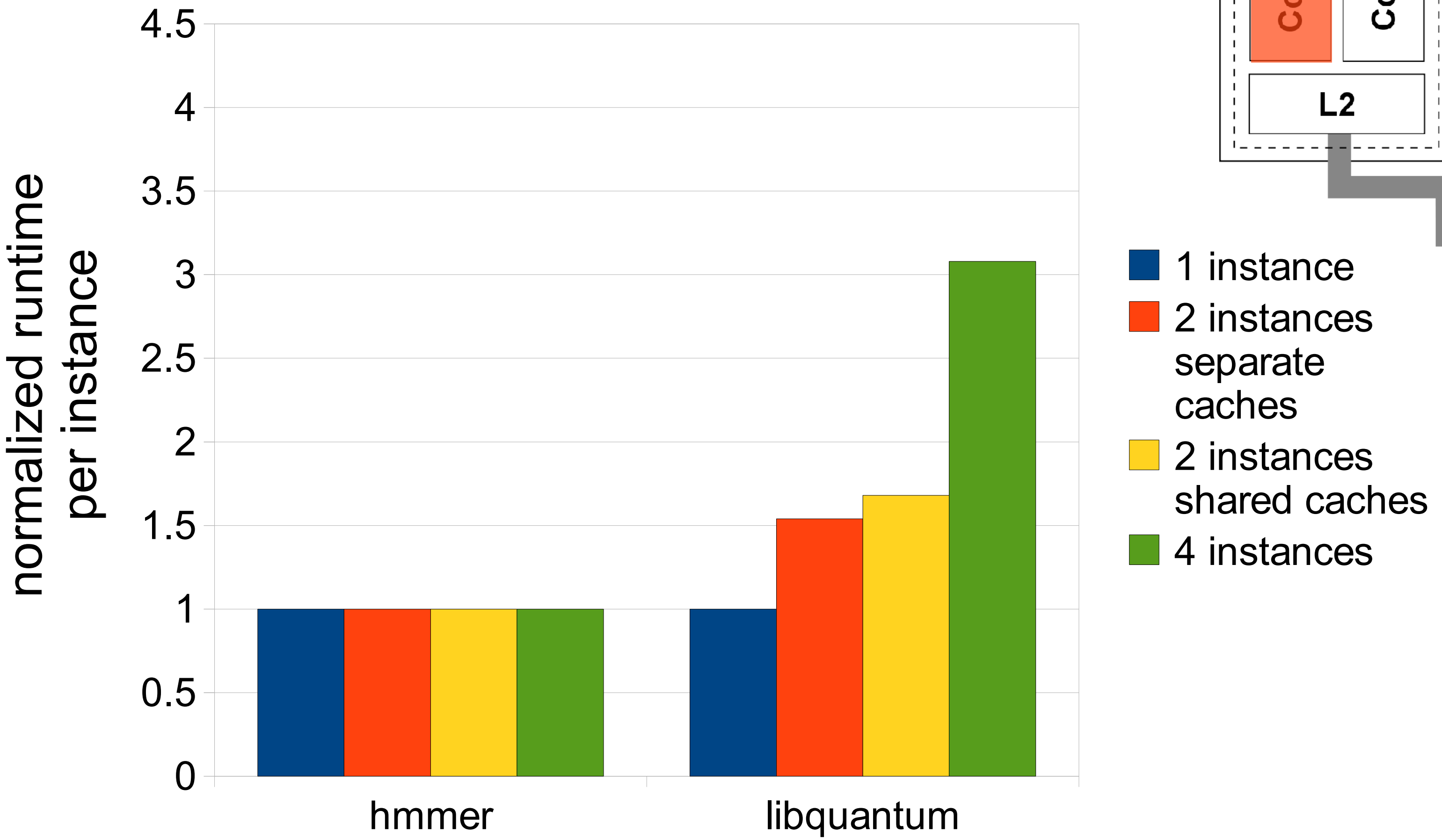


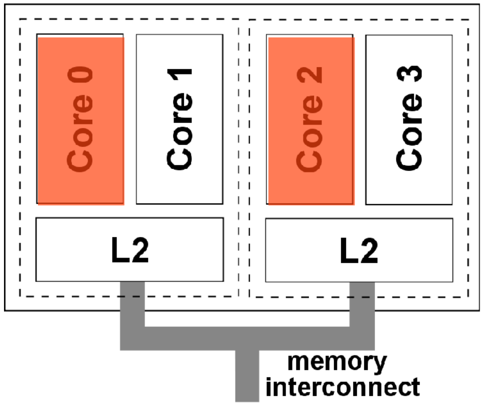
Resource Contention SPEC CPU 2006



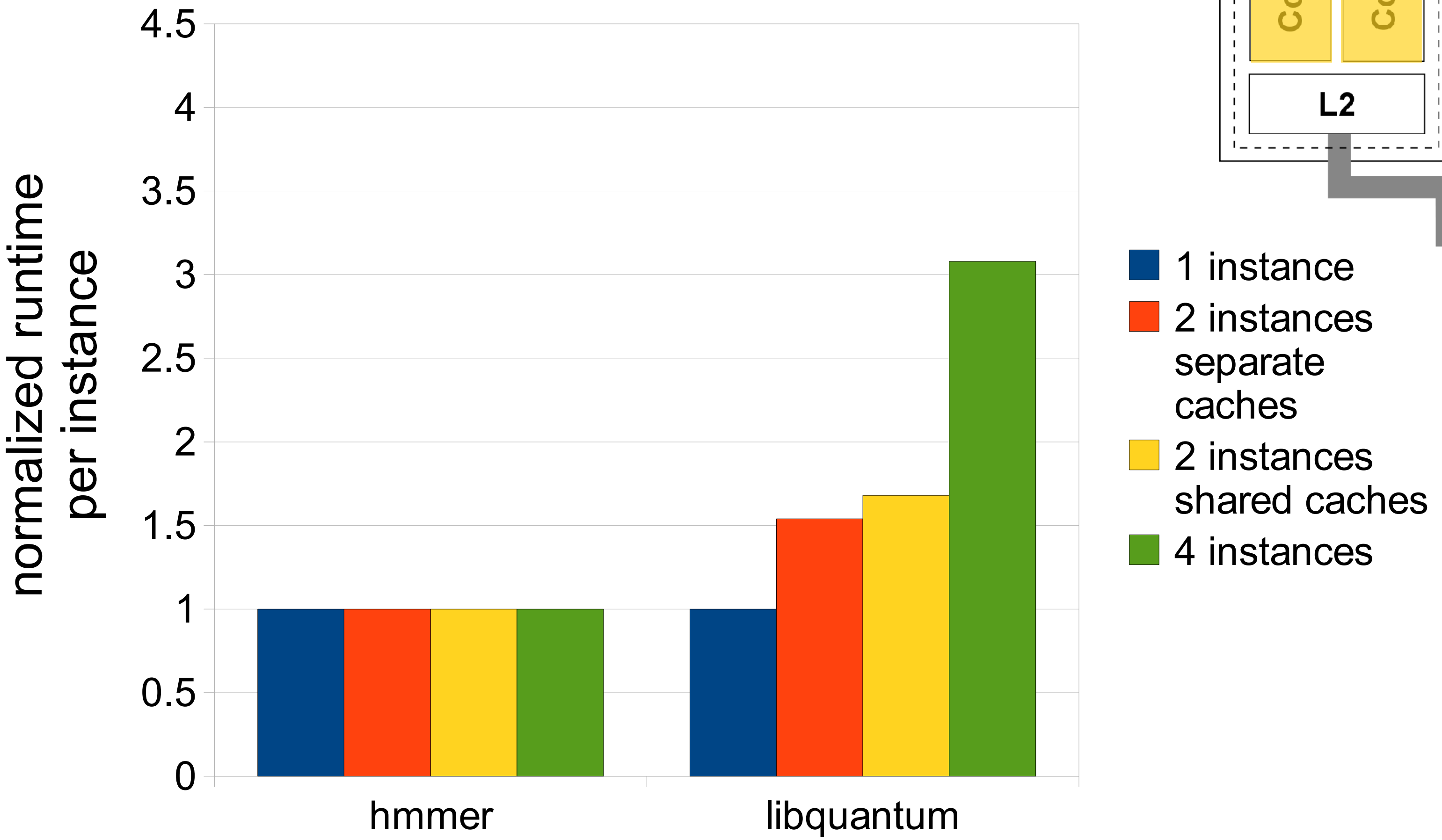


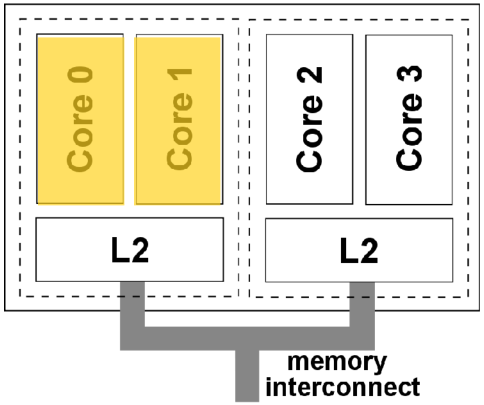
Resource Contention SPEC CPU 2006



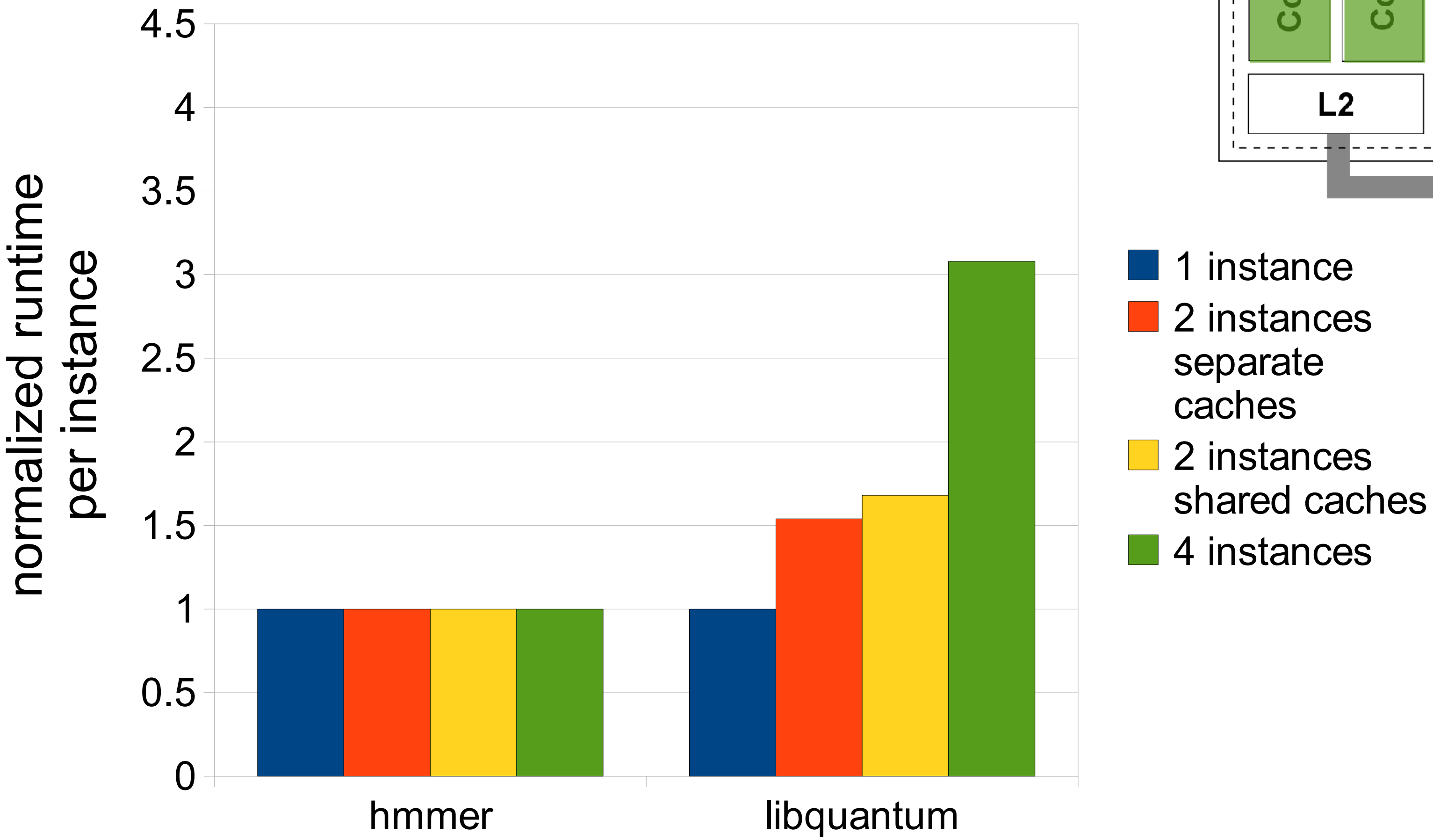


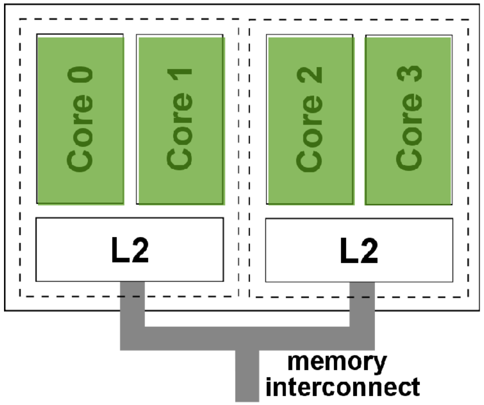
Resource Contention SPEC CPU 2006



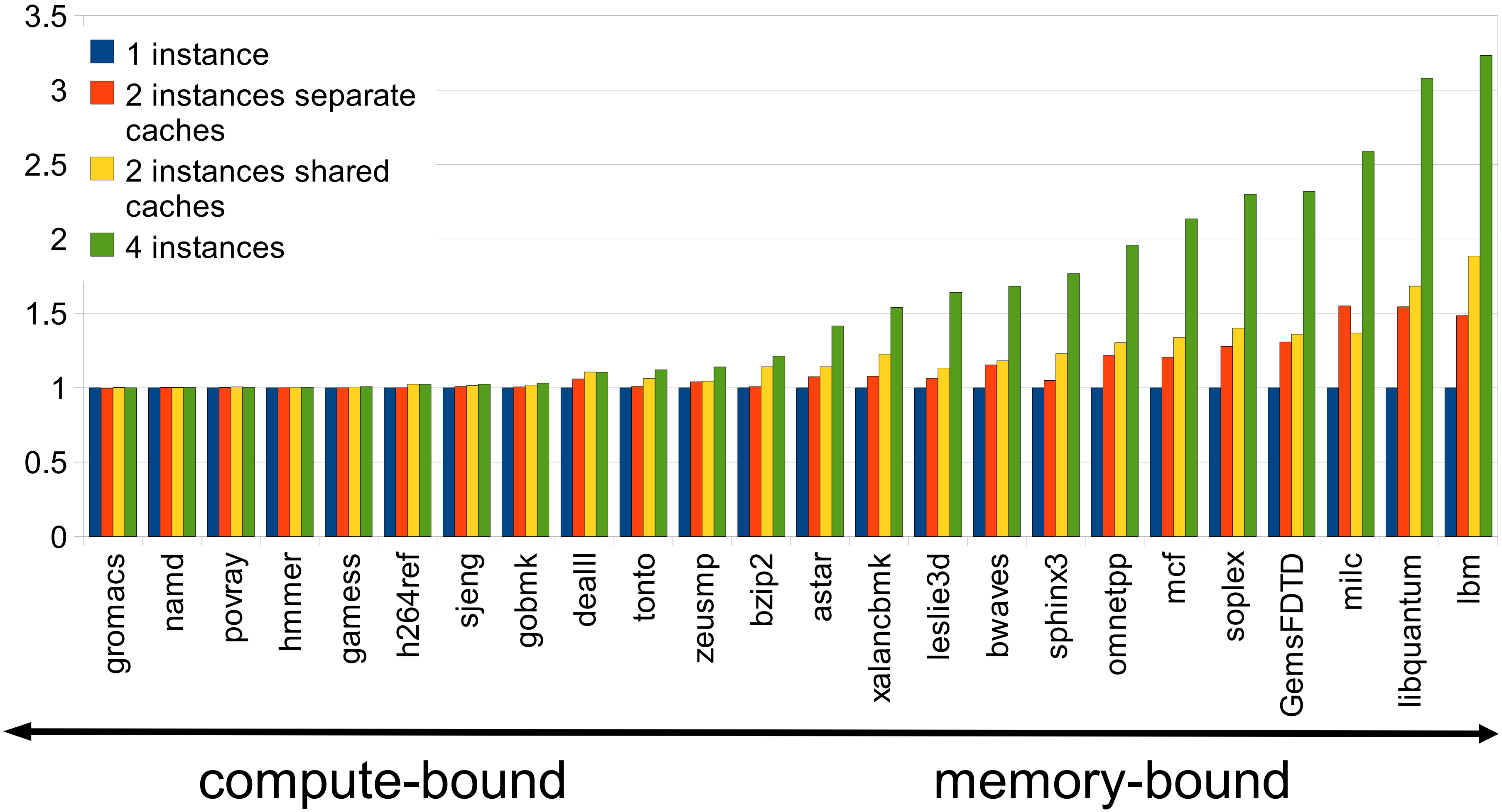


Resource Contention SPEC CPU 2006





Resource Contention SPEC CPU 2006

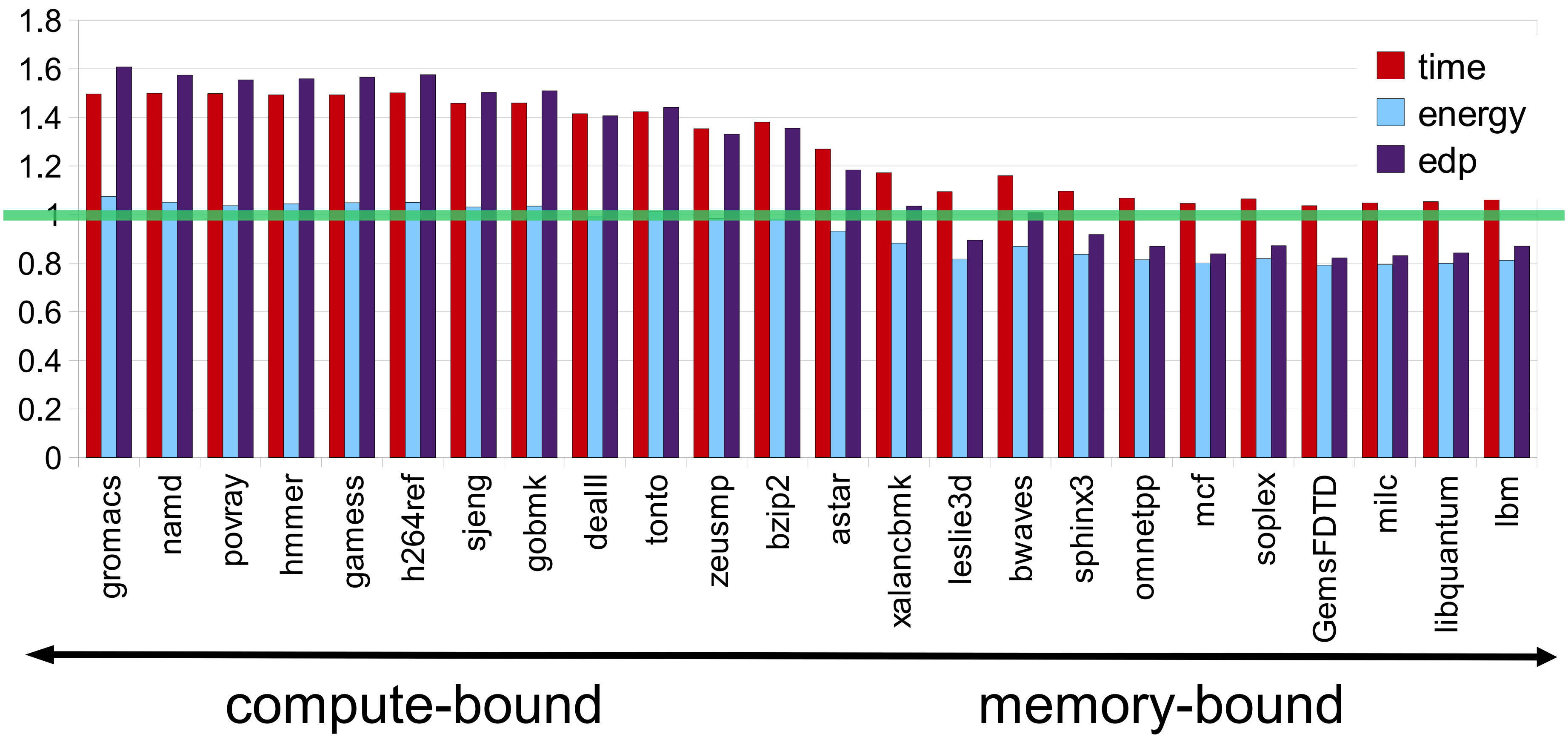


Resource Contention SPEC CPU 2006

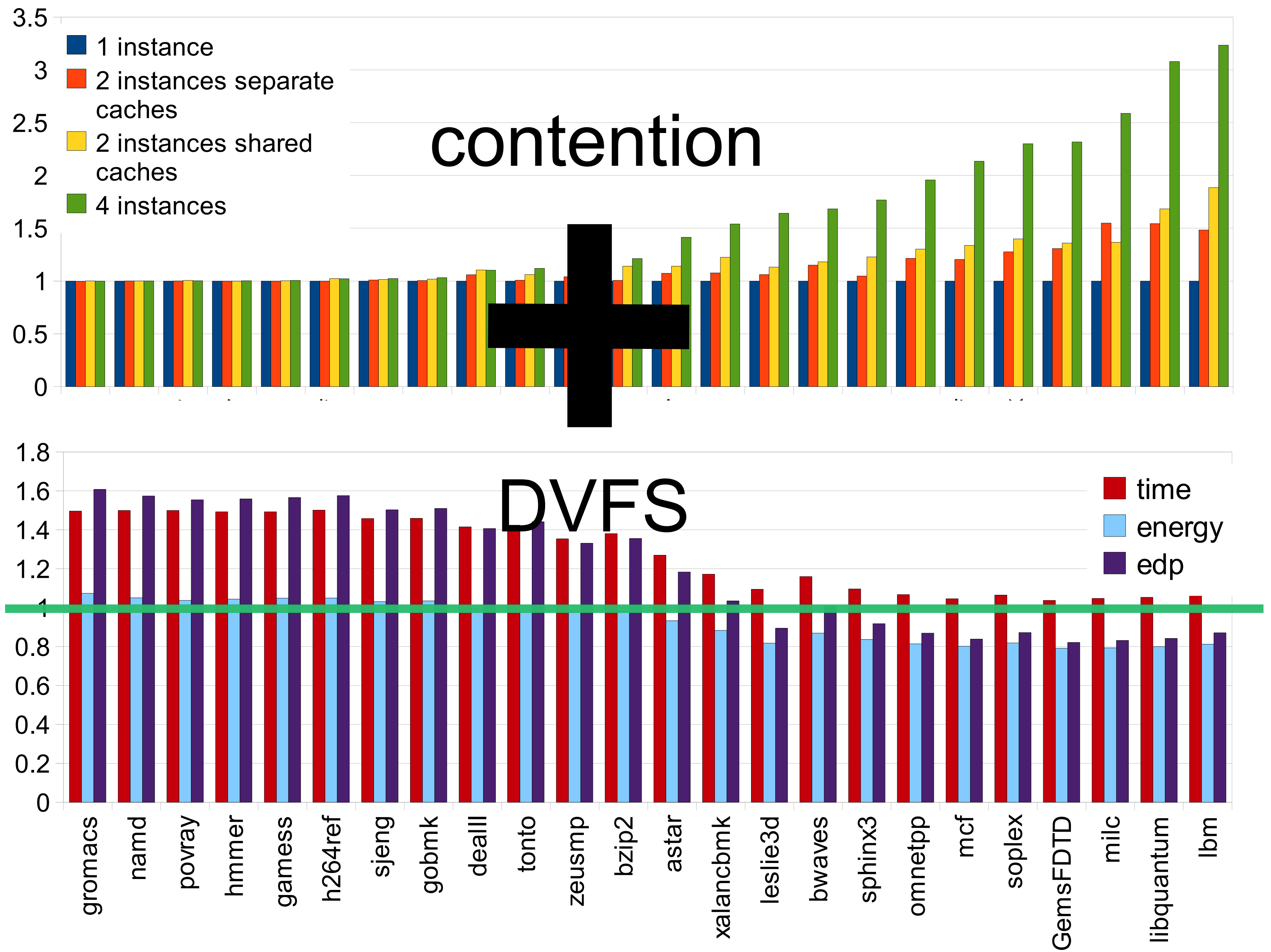
- Compute-bound benchmarks
  - Little resource contention
- Memory-bound benchmarks
  - Severe slowdown caused by memory contention
  - Huge increase in memory demands since SPEC 2000
  - Cache contention of comparatively little importance

Energy Efficiency under DVFS

- Comparison of 1.6GHz to 2.4GHz
- 4 instances of benchmark
- Reducing the frequency pays off for memory intensive tasks

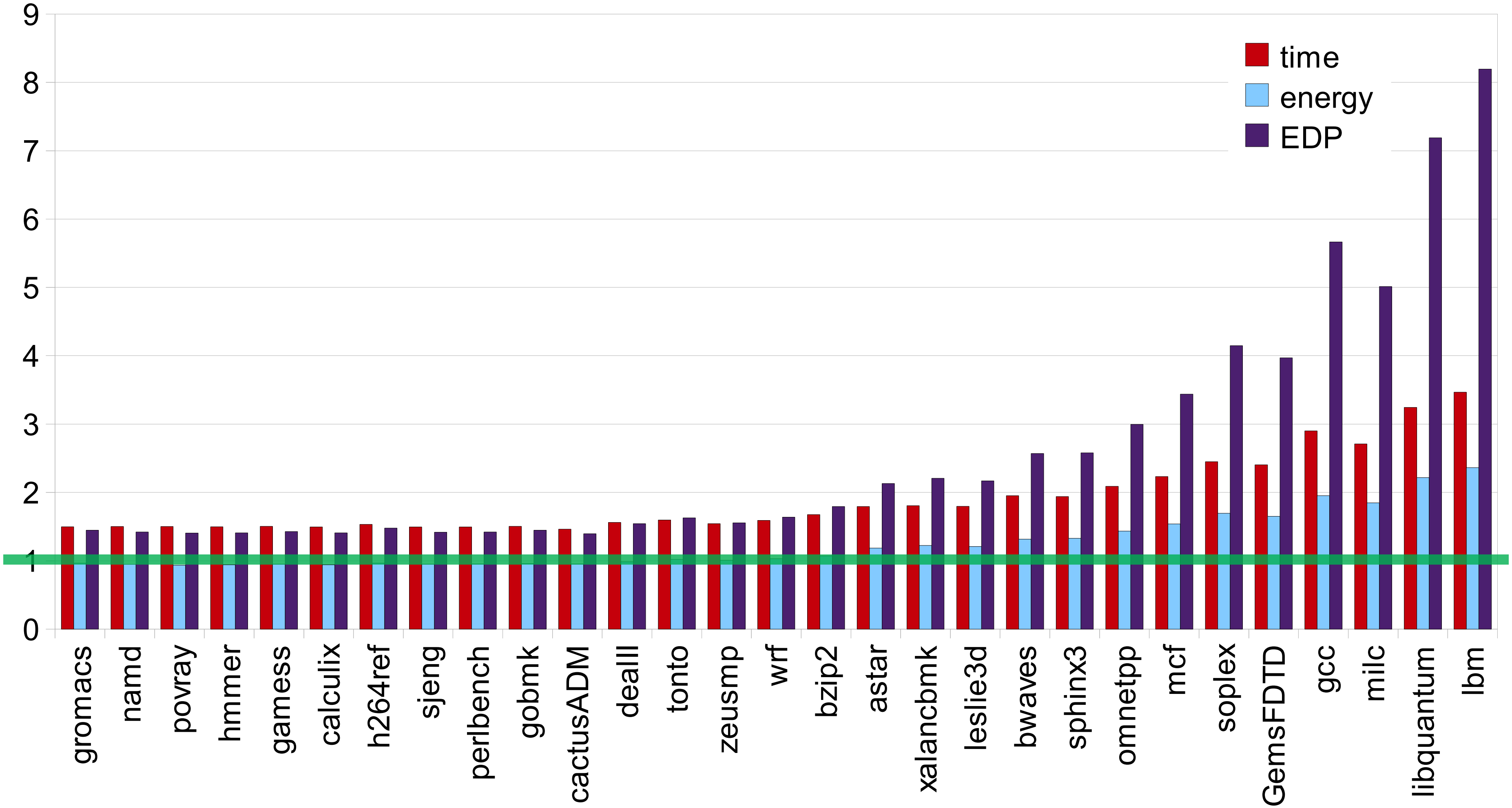


Forschungszentrum Karlsruhe in der Helmholtz-Gemeinschaft
Universität Karlsruhe (TH) Research University · founded 1825

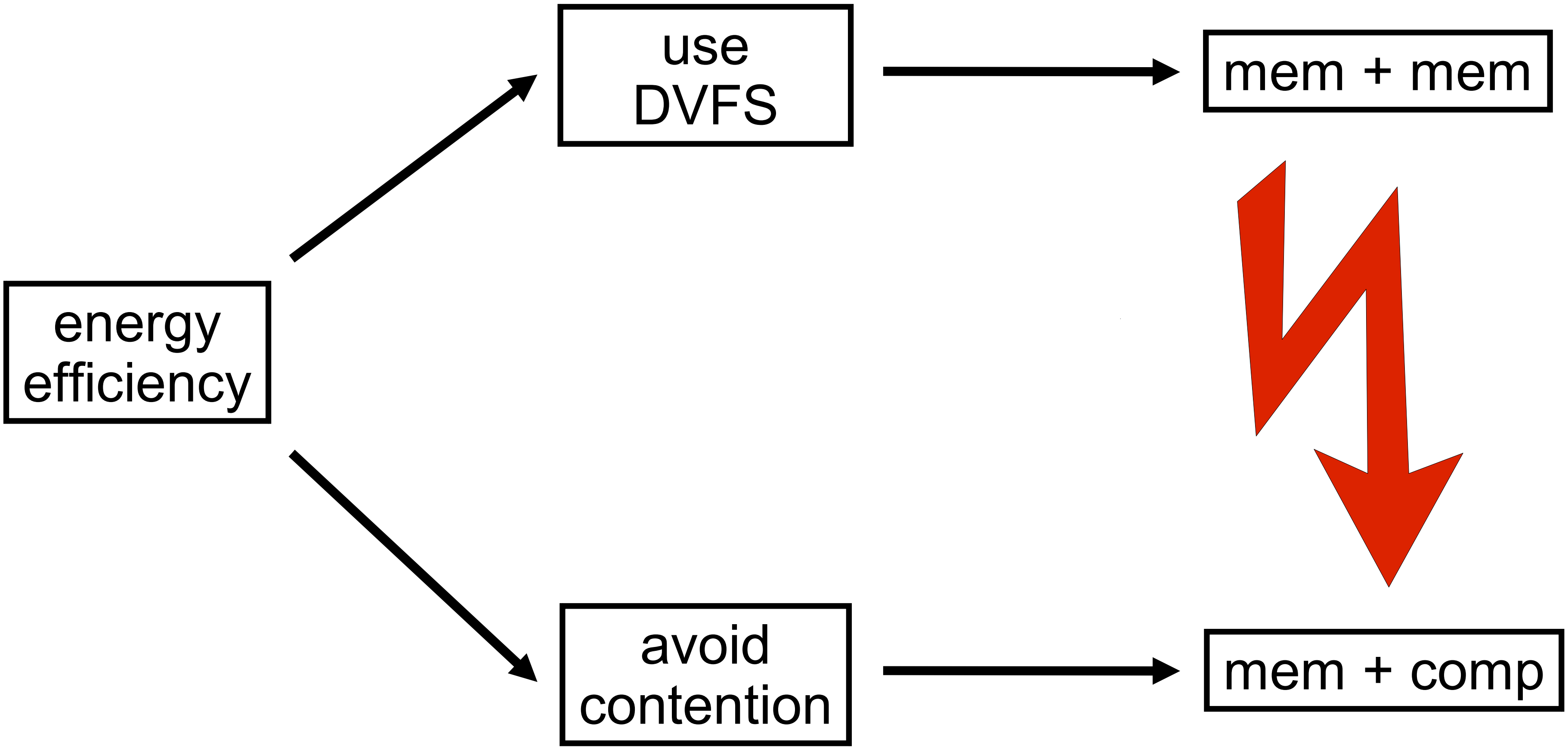


contention
DVFS
1 instance
2 instances separate caches
2 instances shared caches
4 instances
time
energy
edp
gromacs
namd
povray
hmmer
gamess
h264ref
sjeng
gobmk
dealII
tonto
zeusmp
bzip2
astar
xalancbmk
leslie3d
bwaves
sphinx3
omnetpp
mcf
soplex
GemsFDTD
milc
libquantum
lbm

Energy Efficiency under DVFS and Resource Contention

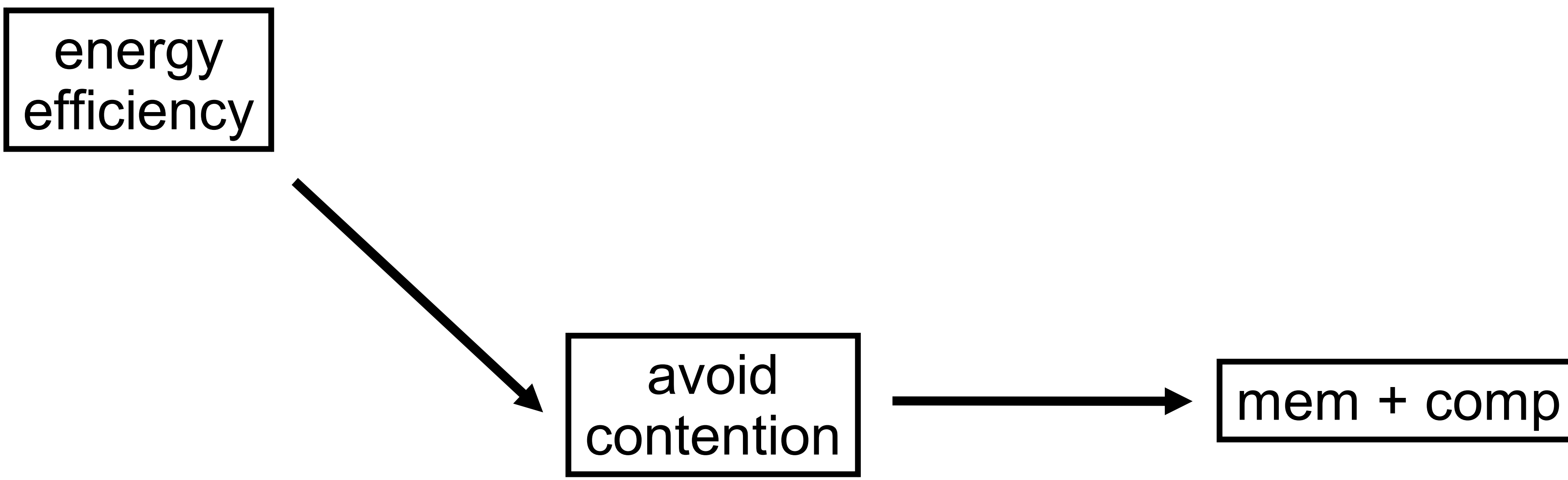


Energy-Efficient Co-Scheduling



Energy-Efficient Co-Scheduling

KIT
Karlsruhe Institute of Technology



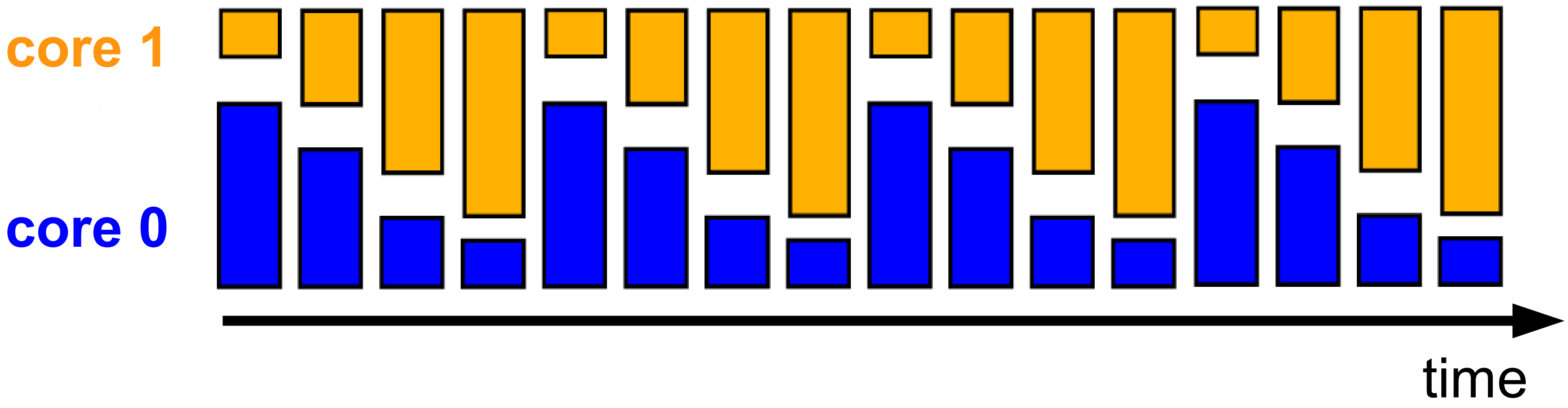
Energy-Efficient Co-Scheduling

Karlsruhe Institute of Technology

- Avoiding resource contention
  - Requires knowledge of task characteristics
  - Requires coordination of task selection across cores
- Merkel and Bellosa, EuroSys 2008
  - Task characterization
  - Execution of tasks in a defined order (runqueue sorting)
  - Used for mitigating thermal effects
- Take advantage of runqueue sorting to provide coordination with low overhead

Sorted Co-Scheduling

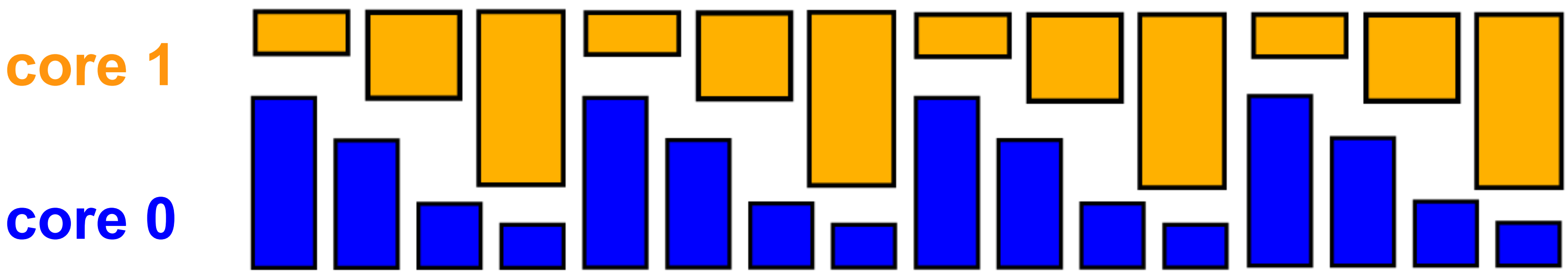
- Group cores in pairs
- Sort runqueues by critical resource (memory bandwidth)
- Coordinate processing of runqueues
- Co-schedule tasks with complementary resource demands



Sorted Co-Scheduling

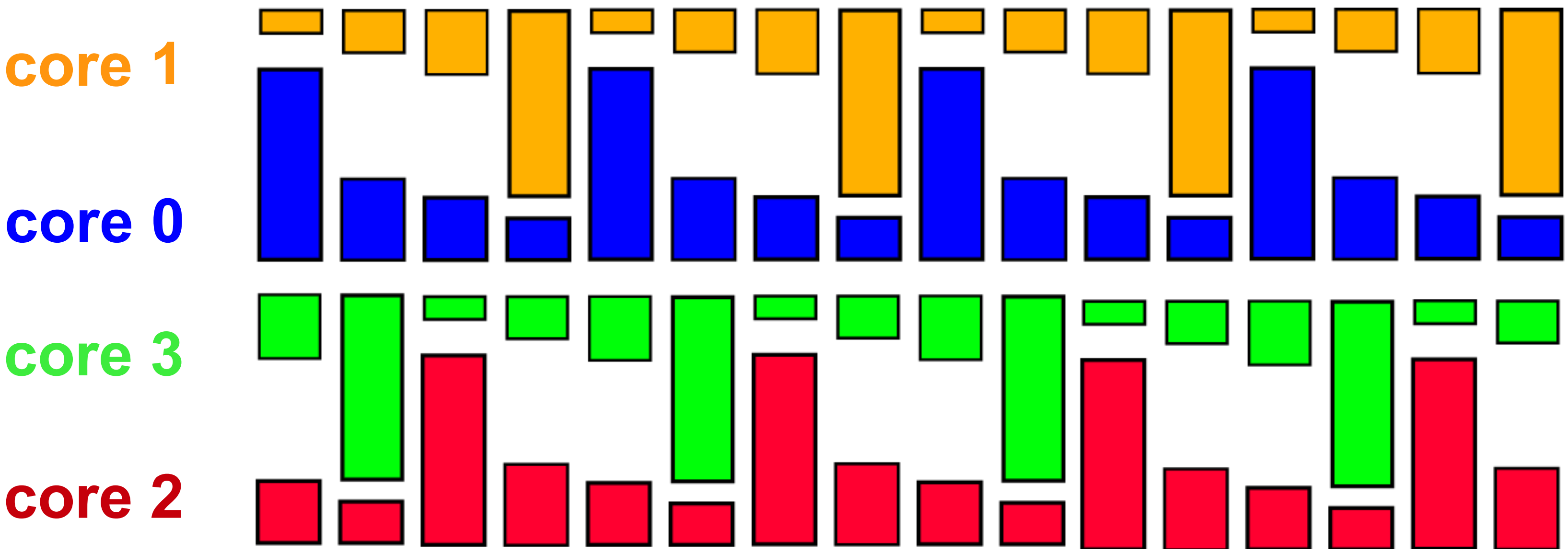
- Dealing with unequeal runqueue lengths

- Example: core 0 executes one task more than core 1
  - Time needed to process runqueues does not even out
    → increase length of timeslices on core 1



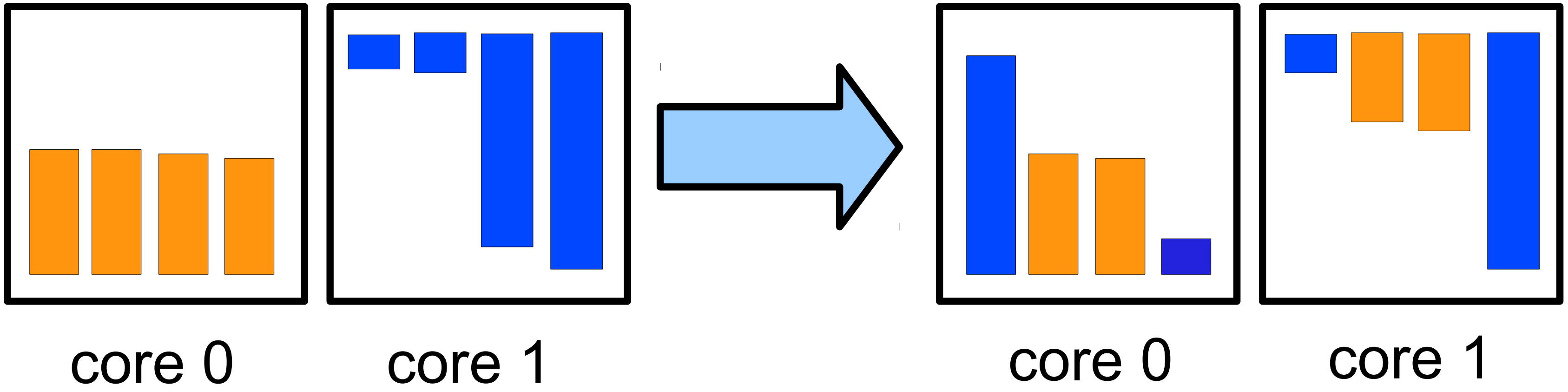
Sorted Co-Scheduling

- Shift runqueues of additional cores
- Avoid running most memory intensive tasks together



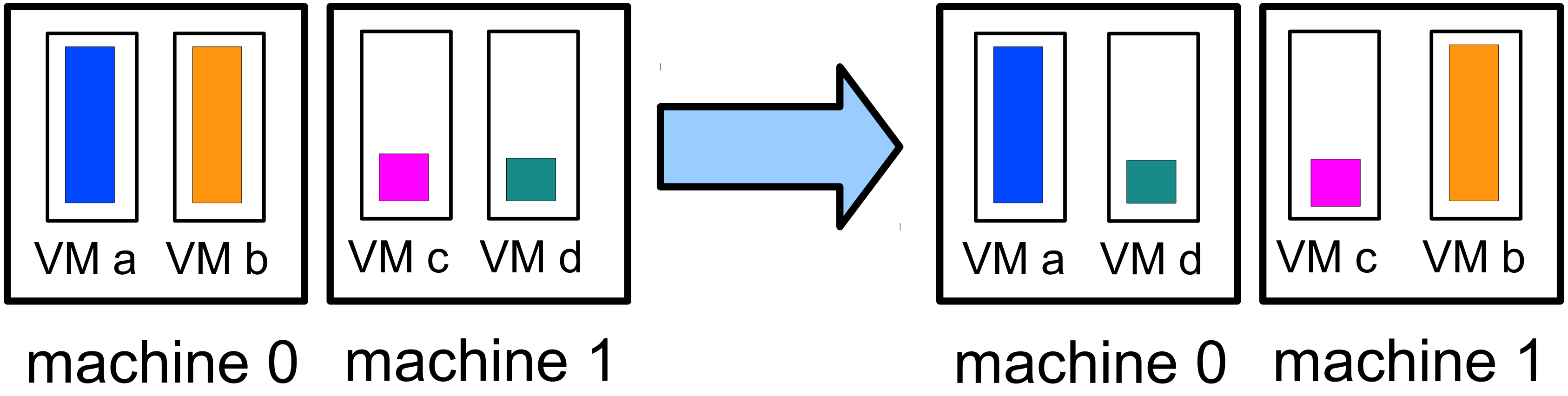
Resource-Conscious Load Balancing

- Sorting requires tasks with different characteristics on each core
- Migrate task if variance among tasks in runqueue is increased



Virtual Machine Scheduling

- Leverage workload diversity of several physical machines
- Extend balancing strategy using the concept of virtualization
- Migrate entire virtual machines
- Co-scheduling of virtual machine instances



Frequency Heuristic

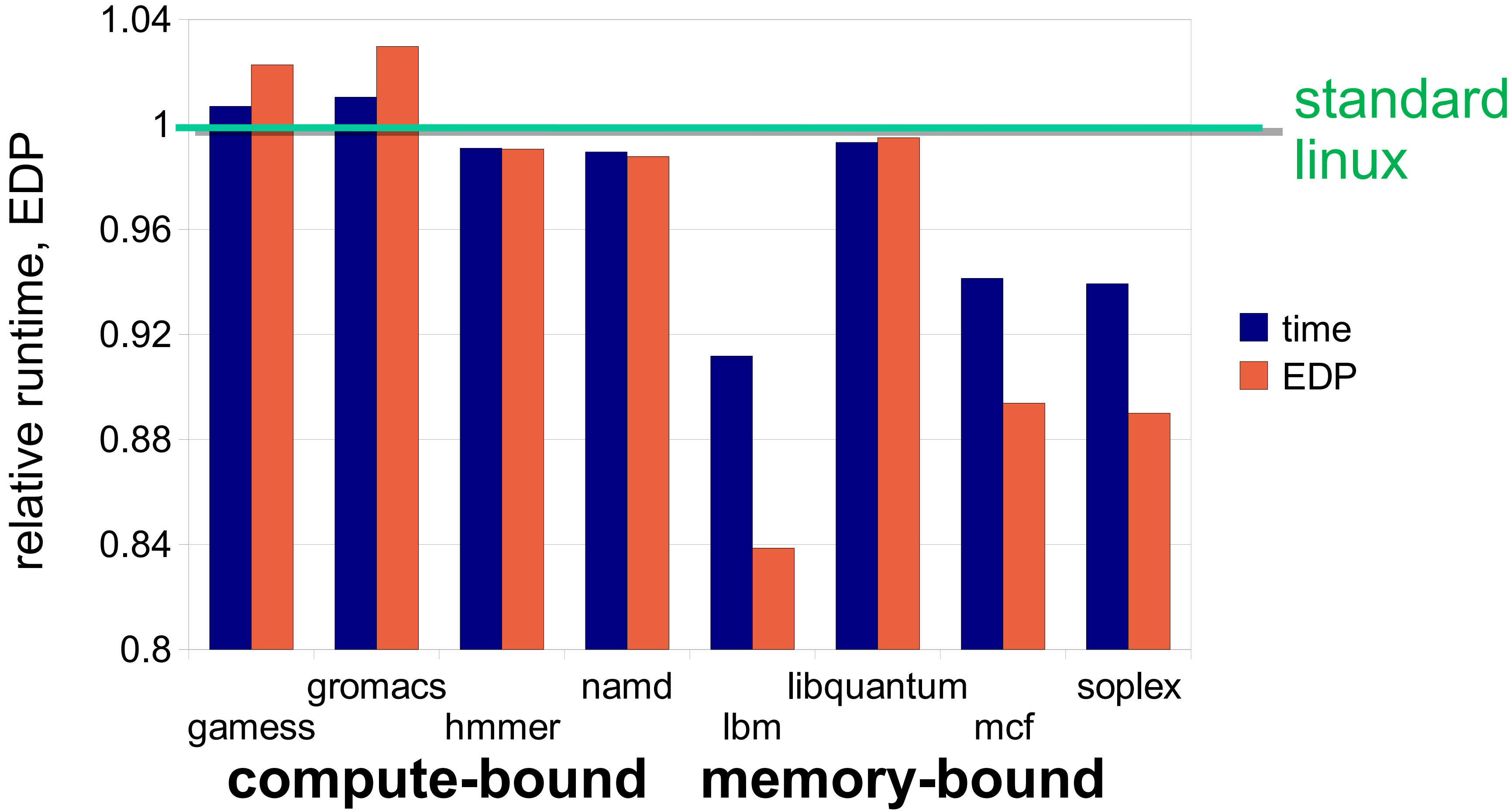
- Fall back to frequency scaling if workload does not allow avoiding contention
- Frequency heuristic takes effect when:
  - Too many memory-bound tasks/VMs are present
  - Sorted scheduling has to co-schedule memory-bound tasks
- Estimate if lower frequency would reduce EDP

Evaluation

- Prototype
- Modified Linux 2.6.22 kernel
  - Runqueue sorting
  - Resource-conscious load balancing
- KVM for virtualization
  - Schedule KVM instances within a physical machine like normal OS tasks
  - Use KVM migration features to move VMs between physical machines

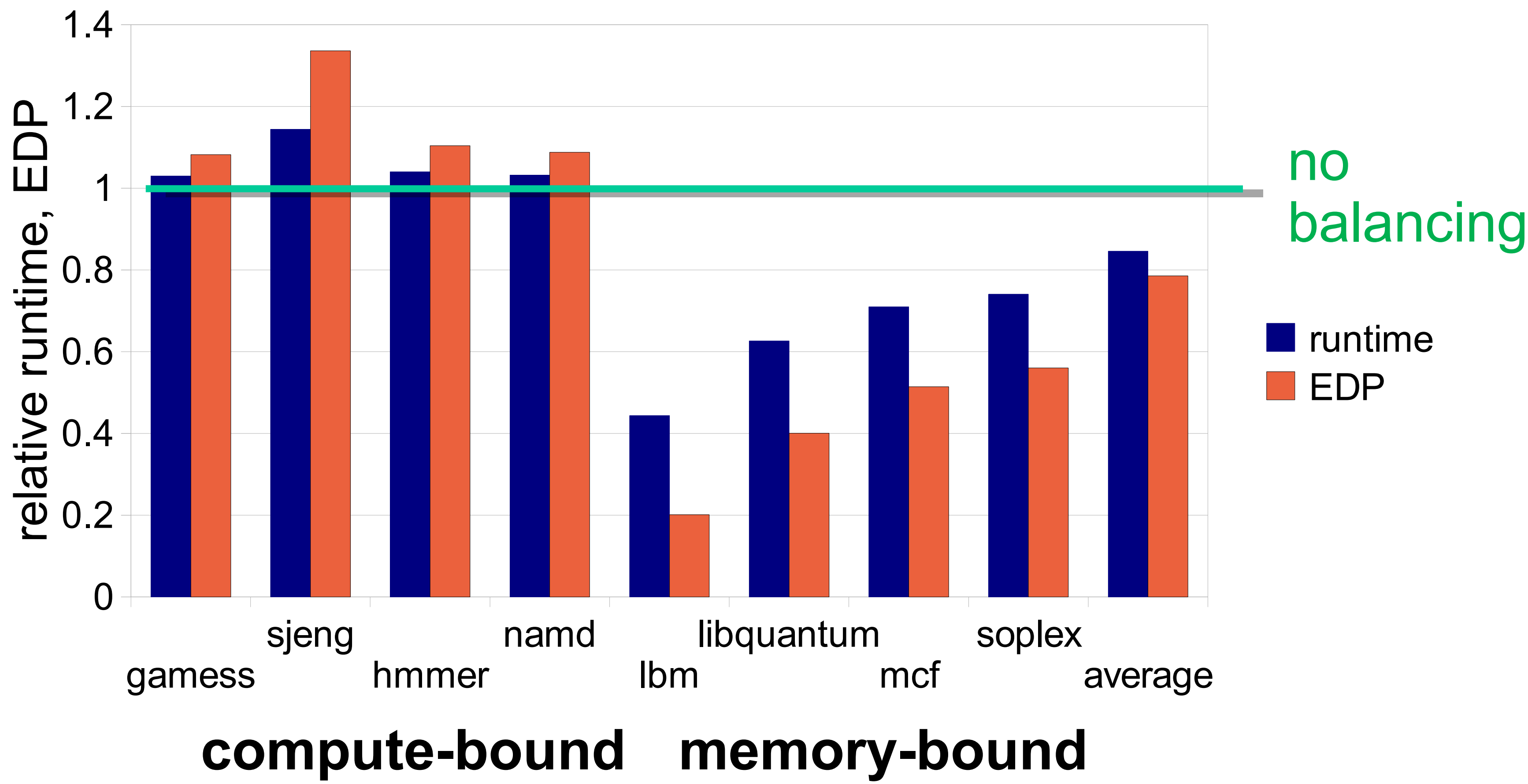
Evaluation

One Intel Core2 Quad, no virtualization
Workload: 8 SPEC benchmarks



Evaluation

Two Intel Core2 Quads
Workload: 8 SPEC benchmarks, each in a separate VM
Worst case: 4 memory-bound benchmarks on one physical machine



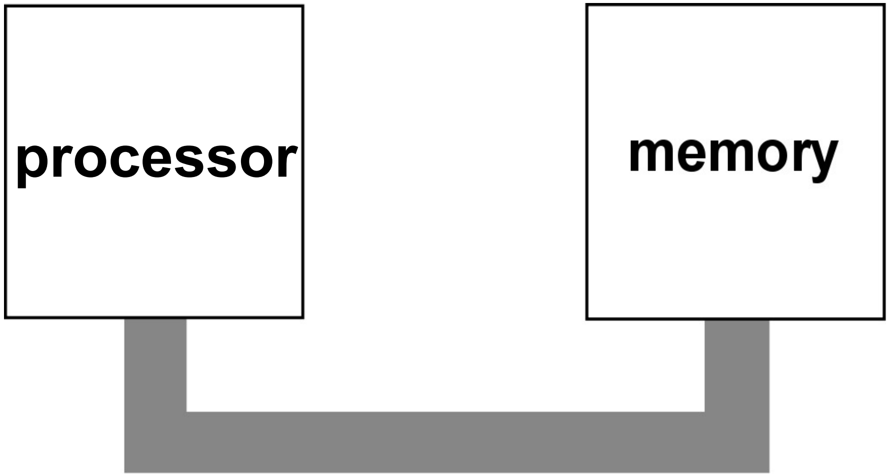
Conclusion

Karlsruhe Institute of Technology

- Cross-effects lead to low energy efficiency in multicores
  - Resource contention
  - Shared voltage domains
- Analysis: contention avoidance more important than common optimal frequency/voltage
- Approach: co-scheduling by sorting memory intensity in different directions
  - Resource-conscious load balancing
  - VM scheduling and migration
  - Frequency scaling as fallback
- Result: reduction of EDP by 10 to 20%

Energy Efficiency under DVFS

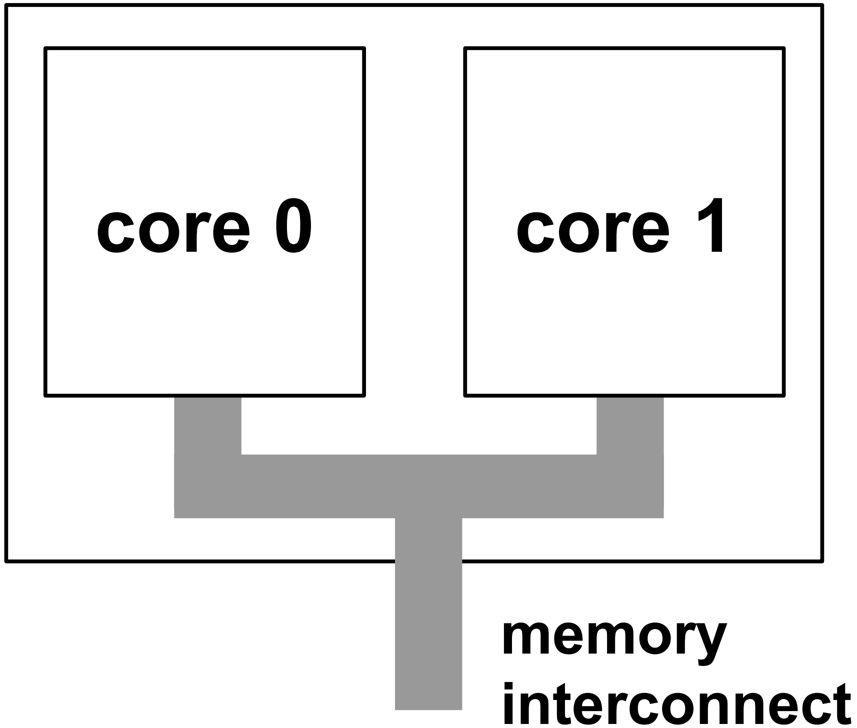
- Task specific optimal processor frequency/voltage
  - Memory-bound task ⟶ low frequency
  - Compute-bound task ⟶ high frequency



Resource Contention

- Tasks compete for shared chip resources
  - e.g., caches, memory (CMP)

- Impact on
  - Runtime
  - Energy efficiency



New Challenges for OS Scheduling

- Scheduler determines task execution
  - When
  - Where
  - What combination

- Scheduling decisions have impact on
  - Energy efficiency
  - Resource contention

→ **Information about task characteristics is crucial!**

Resource Contention vs. Frequency Selection

- Reducing contention has much greater potential for increasing energy efficiency than DVFS

➔ Schedule tasks in a way that avoids contention, even if some tasks have to run at the “wrong” frequency

New Challenges for OS Scheduling

- Task characterization in today's general purpose OS schedulers
  - User-specified priorities
  - I/O-intensive vs. CPU-intensive
  - No indicators for energy efficiency, or resource contention

Task Characterization

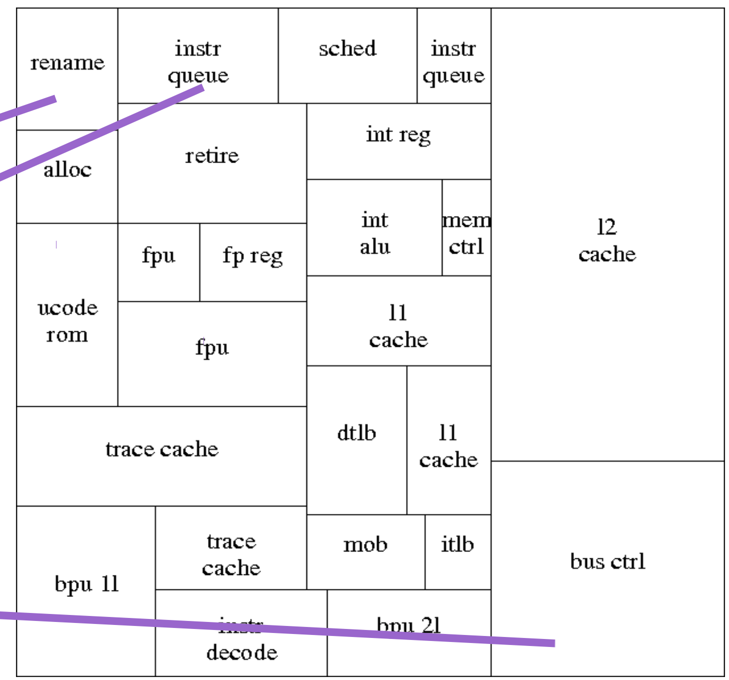
- Task activity vectors
  - Characterize tasks by their resource utilization
    - (e.g., functional unit, cache, memory interconnect, ...)
  - Provide information to smart schedulers
- Resource utilization: versatile indicator for
  - Temperature
  - Optimal frequency
  - Contention

Task Activity Vectors: A New Metric for Temperature-Aware Scheduling
Andreas Merkel and Frank Bellosa
Third ACM SIGOPS EuroSys Conference, 2008

Task Activity Vectors

- Vector with n components
  - Each component represents a resource
  - Component value: utilization of resource while task is running
    - Inferred on-line from performance monitoring counters

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \cdots \\ v_n \end{pmatrix}$$



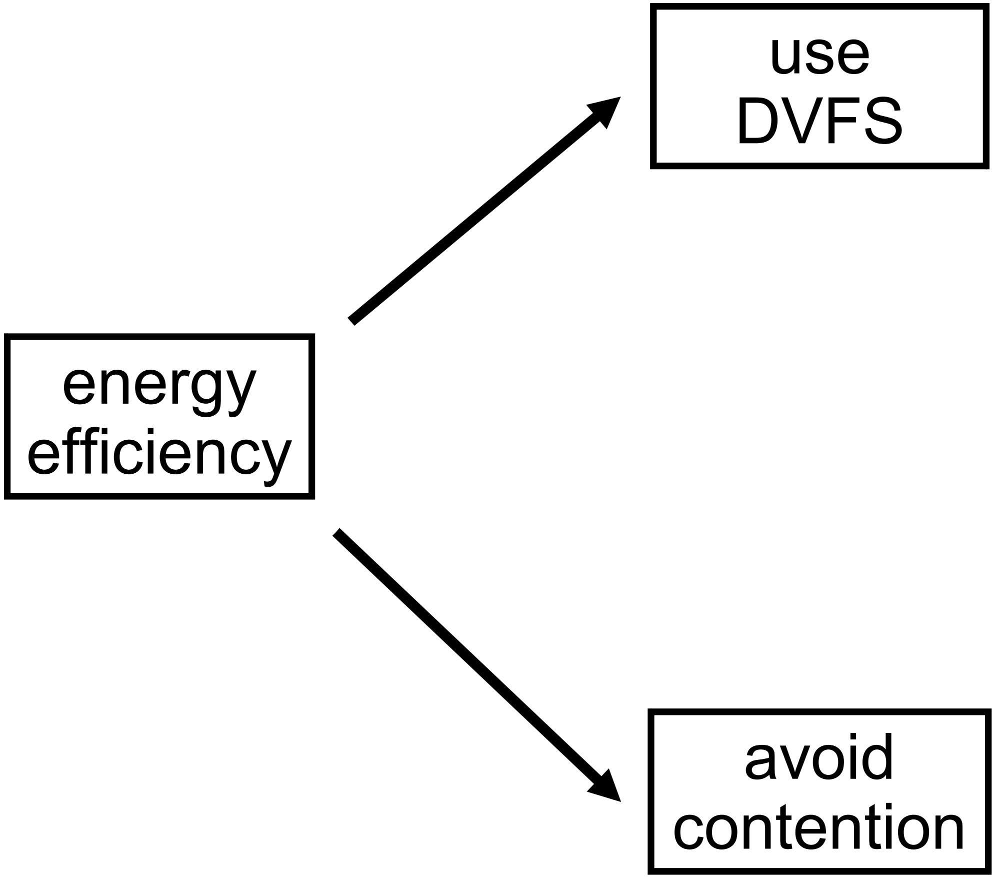
Vector-Based Scheduling for Energy Efficiency

Karlsruhe Institute of Technology

- Multiprocessor schedulers make decisions independently for each processor
  - Arbitrary combinations of tasks running together
    - Disregarding of interference
    - Disregarding of task-specific optimal frequency

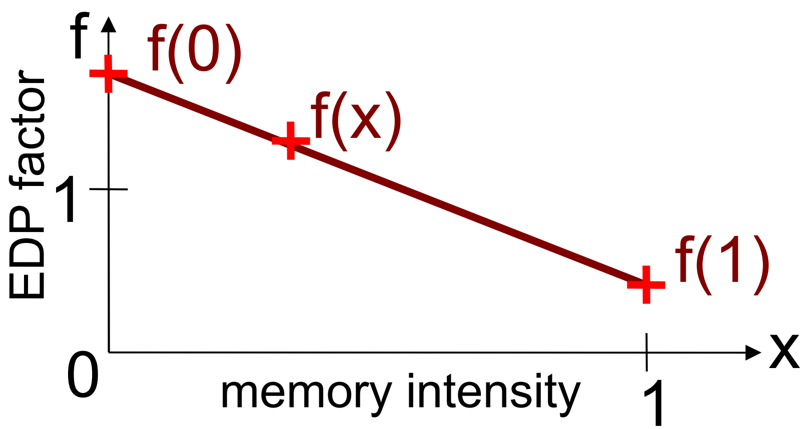
→ Resource contention
→ Prolonged task runtimes
→ Inefficient use of energy

Energy-Efficient Co-scheduling



EDP Estimation

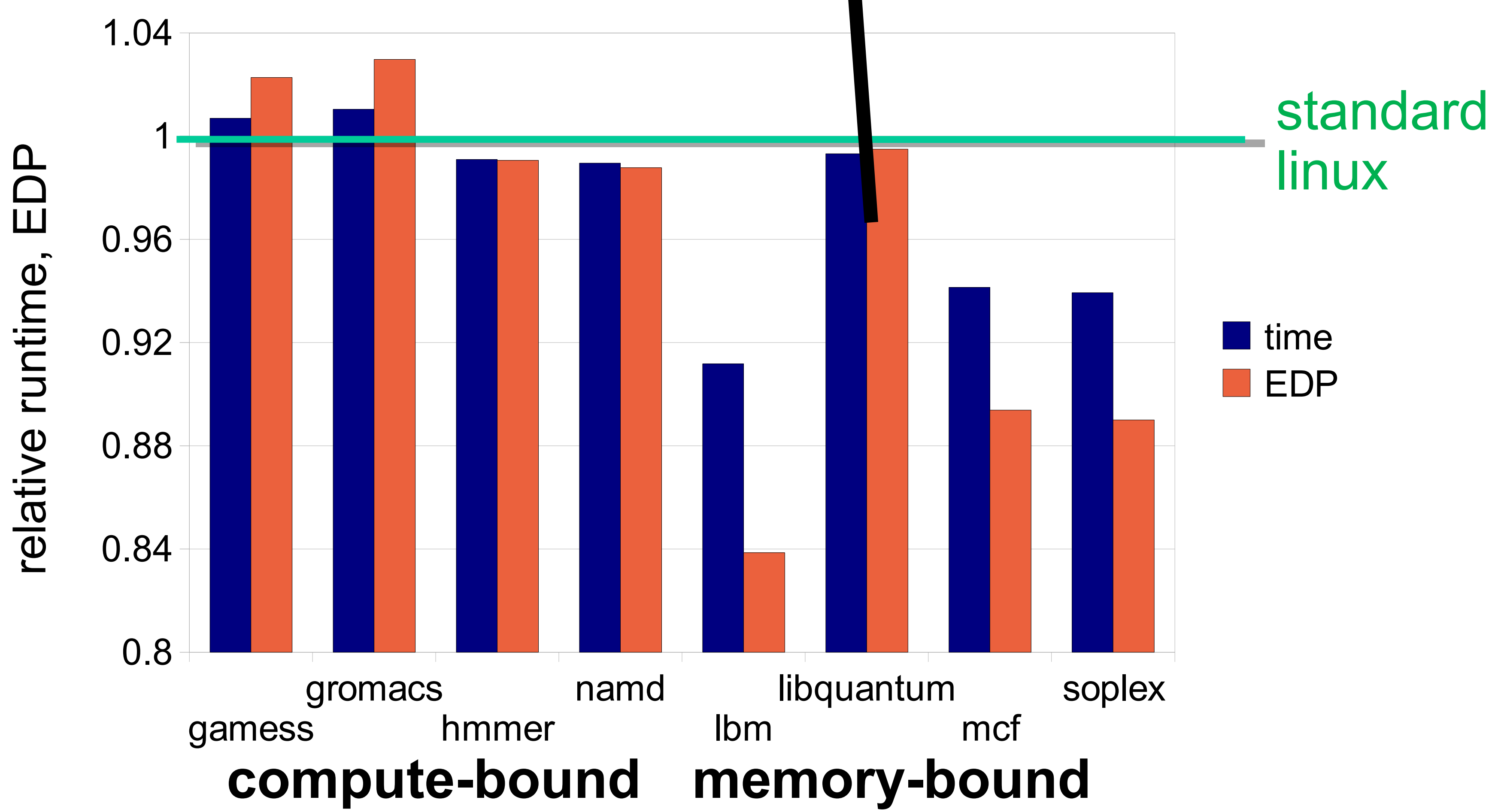
- Linear interpolation
  - f(1): EDP factor of completely memory-bound microbenchmark
  - f(0): EDP factor of completely compute-bound microbenchmark
  - Estimation for EDP factor of task with memory bus utilization x:
    $f(x) = x * f(1) + (1-x) * f(0)$



Evaluation

Moscibroda and Mutlu
Memory performance attacks: denial of memory service in multicore systems
USENIX Security Symposium 2007

Modified Linu
Workload: eig



New Processor Topologies

- On-chip thread-level parallelism
  - simultaneous multithreading (SMT)
    chip multiprocessors (CMP)

- shared resources

- shared power management

Old Scheduling Policies

Karlsruhe Institute of Technology

- Schedulers designed for traditional SMP systems

- Independent scheduling decisions for each processor
  - combination of tasks running at a time is arbitrary
  - is this optimal for SMT/CMP?
  - what about resource contention?
  - what about power management features like frequency scaling?

- Assumption: a set of unrelated, single-treaded processes is running
  - no communication

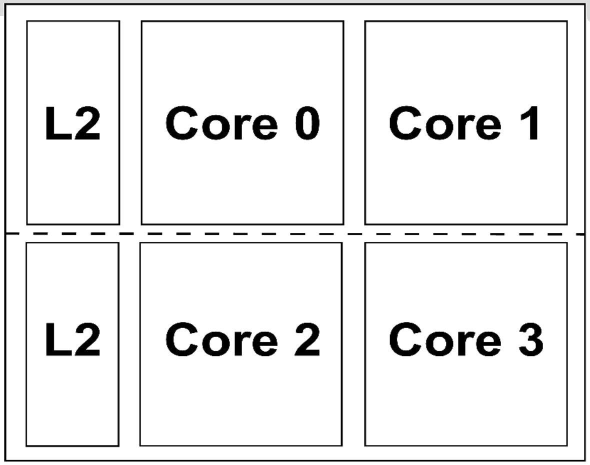
Power Management

- Frequency selection
  - SMP: independently for each processor
  - SMT: affects all logical threads of a processor
  - CMP: per-core selection possible at the price of hardware complexity, but often only per-chip
- Some tasks run more efficiently at a certain frequency than others
  - memory-bound tasks: lower frequencies
  - compute-bound tasks: higher frequencies

Multiprocessor Architectures

- Classical SMP
  - physically different chips
  - interference via memory bus (shared bus, cache coherency)
- SMT
  - multiple logical threads on one chip
  - heavy contention for almost all resources
- CMP
  - multiple processors on one chip
  - interference via memory access logic, memory bus
  - sometimes shared caches

Experiments



- Intel Core2 Quad
  - resource contention
    - L2 cache shared between 2 cores
    - memory access infrastructure shared by all 4 cores
  - frequency selection
    - frequency shared by two cores
    - voltage scaling only for entire chip
- Microbenchmarks
- SPEC CPU 2006 benchmarks

Discussion

Karlsruhe Institute of Technology

- Lower frequency is beneficial if all cores execute memory-intensive tasks
- But: Overhead in terms of time and energy if all cores execute memory intensive tasks
- Do the benefits outweigh the overhead?

No:

Contention causes runtime to increase by up to factor 2 to 4

Frequency scaling reduces energy by factor 0.7 at best

=> avoiding contention central issue for energy efficiency

Example Scenario

- 4x hmmer (compute-intensive)
- 4x soplex (memory-intensive)

	hmmer, 4 instances	soplex, 4 instances	total
energy 24	86427.92	70741.44	157169.36
time 24	1210	1230	2440
energy 16	90885.56	57048.03	147933.59
time 16	1817.5	1282.5	3100
energy 24/16			143475.96
time 24/16			2492.5

	2 hmmer (with 2 soplex)	2 soplex (with 2 hmmer)	total
energy 24	79308.38	51718.62	131026.99
time 24	1230	805	2035
energy 16	86645.61	42120.7	128766.32
time 16	1840	899	2739

Goals

- Design scheduling policy that is optimal for the new architectures
- Use the resource CPU as efficiently as possible in terms of
  - energy
  - time
- Sometimes controversial goals
  - compromise: EDP = energy * delay

Goals

- Run tasks in combinations that cause no interference
- Run each task at its optimal frequency
  - combination matters, if frequency selection affects multiple CPUs
- => we need to be able to determine what tasks run simultaneously

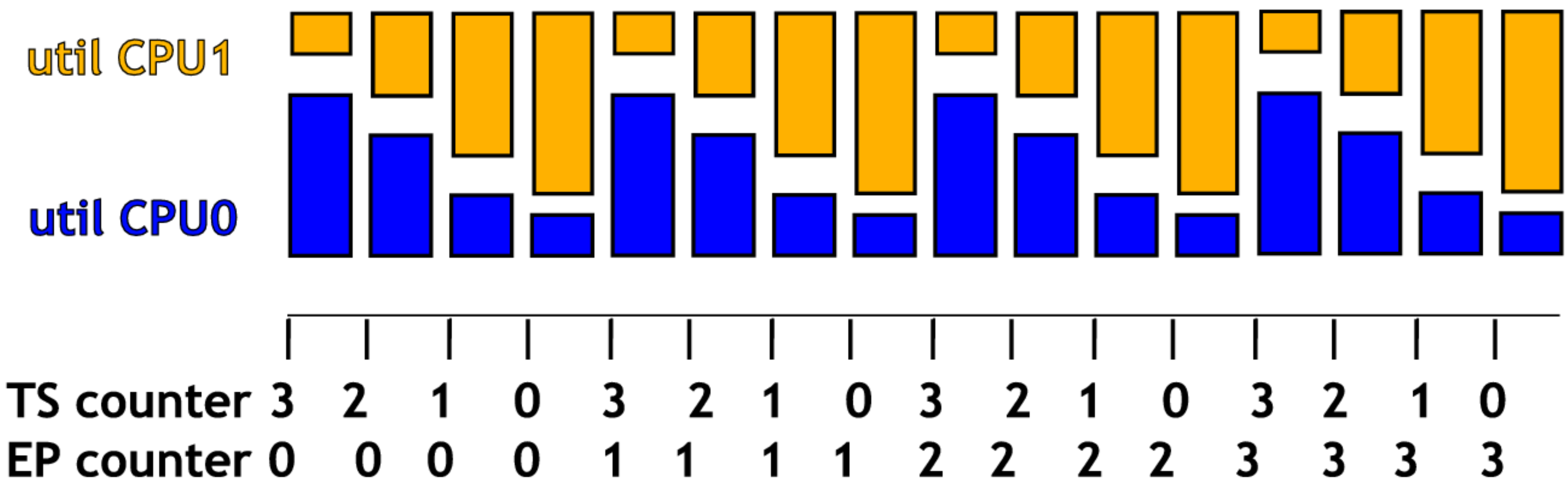
Mechanisms

- Task migrations
- Coordination of scheduling decisions (sort of gang scheduling)

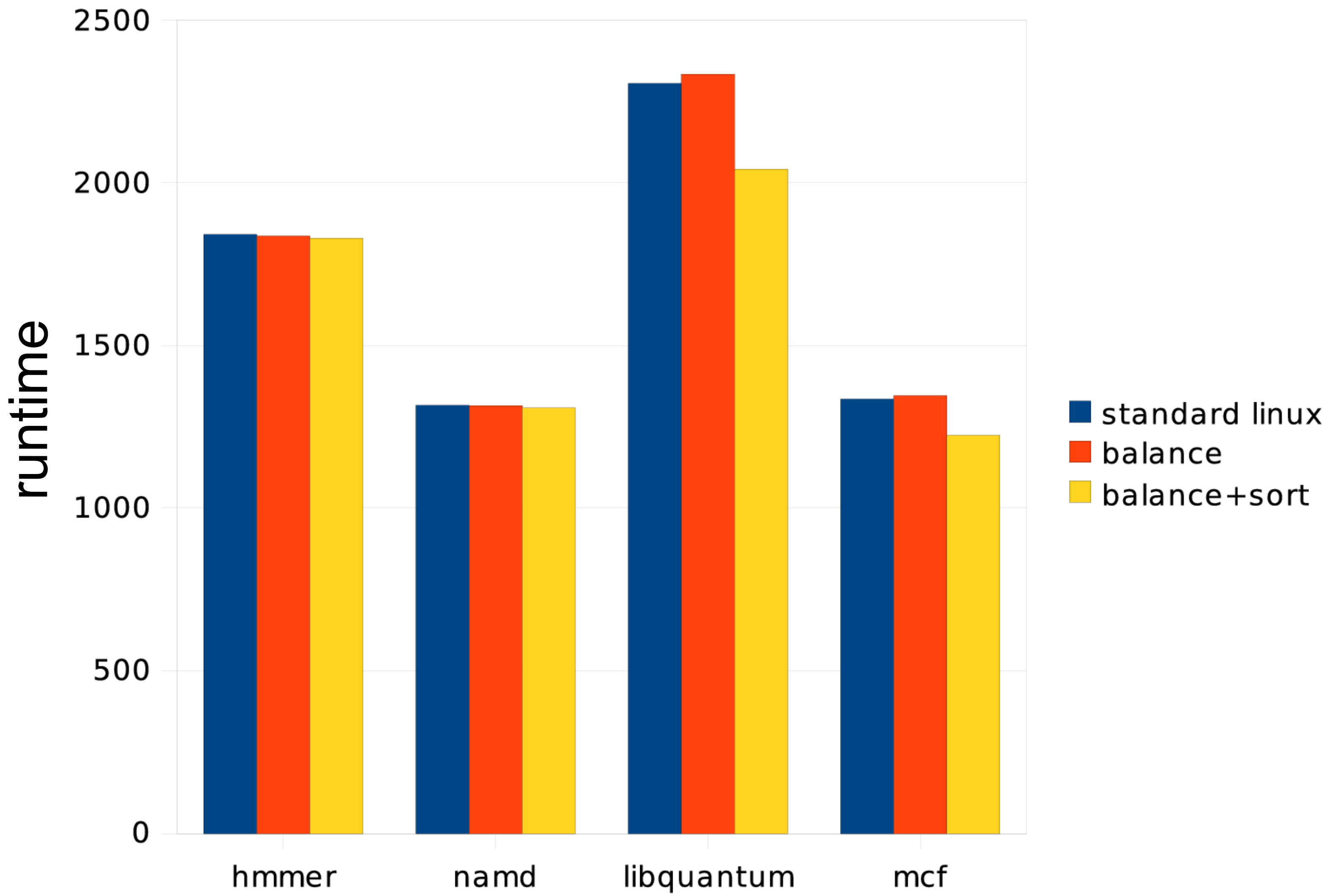
Result

- Run memory-intensive tasks parallel to compute-intensive tasks at highest frequency
- Only lower the frequency if nothing but memory-intensive tasks are available for execution

Sorted Scheduling



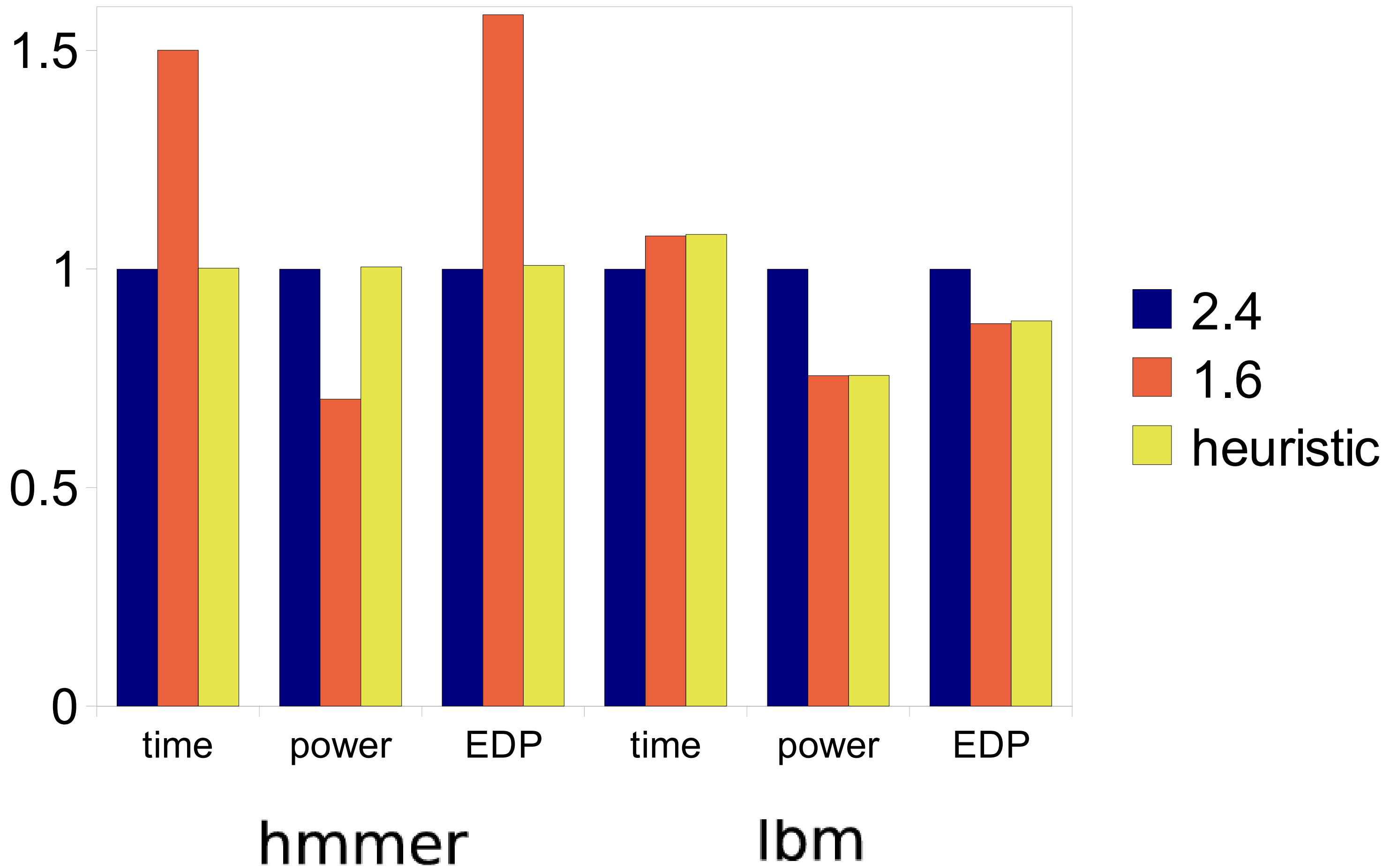
Evaluation Sorting (Dual Core)



compute-bound memory-bound

Evaluation Frequency Heuristic

- Execution of 4 x hmmer and 4 x lbm
- normalized to 2.4 GHz



Evaluation: discussion

- Improved runtime and EDP by avoiding contention
  - Reduction of EDP by reduction of runtime
- Frequency scaling only beneficial if scheduling cannot avoid contention
  - Reduction of EDP by reduction of power