

# CHOAMP: Cost Based Hardware Optimization for Asymmetric Multicore Processors

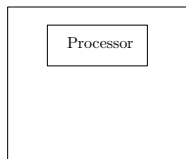
**Jyothi Krishna V S**, Shankar Balachandran and Rupesh Nasre

[jkrishna@cse.iitm.ac.in](mailto:jkrishna@cse.iitm.ac.in)

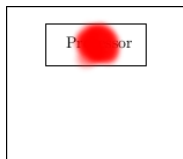
IIT Madras

Feb 13, 2017

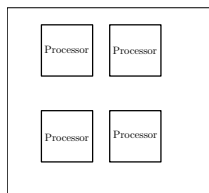
# CPU: Power Wall



CPU



CPU

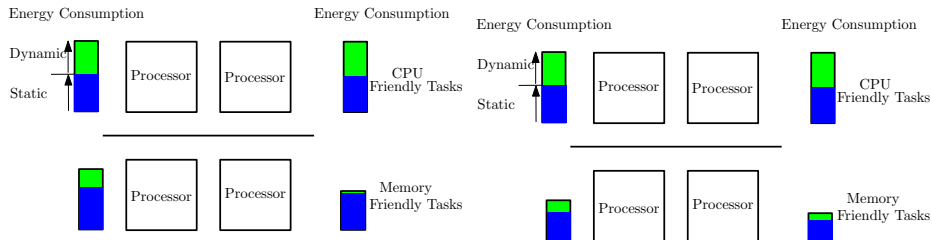


CPU

- Moore's Law hit the Power Wall
- Not practical to keep on increasing the Frequency
- Multicore environment
- Parallel Programming to keep on reaping benefit

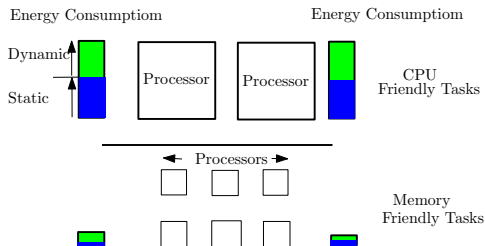
# High Static Cost

## DVFS



- CPU Friendly Tasks: Mostly ALU operation (active CPU cycles)
- Memory Friendly Tasks: Mostly memory/ Branch operations
- DVFS: reduced the negative effect
- Still had high static cost

# Asymmetric Multicore Processors (AMPs)

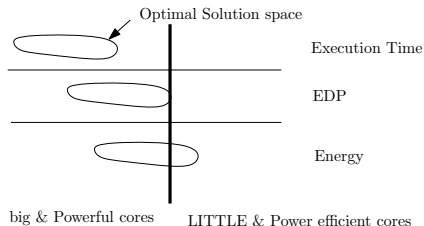


- CPU friendly tasks to big & powerful cores
- Memory friendly tasks to small & power efficient cores

# AMP: Issues with Multithreaded Program

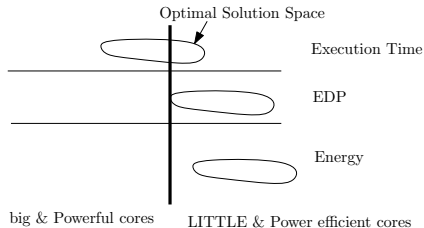
- Which type of core to run each thread?
- Optimal task to thread mapping of the cores?
- Does an AMP environment help? (Use one type of core at a time)
- Number of resources of each type?
- Optimal Frequency configuration for each core type?
- Can answers to all above questions change with change in Cost Function?
  - Execution time
  - Energy Consumption
  - EDP etc.

# AMP: CPU Friendly Threads



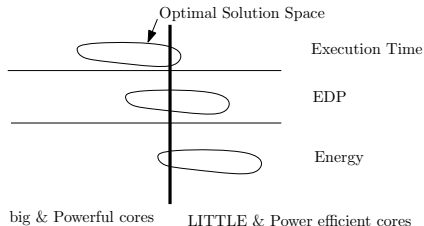
- Execution Time: Powerful Cores
- EDP: Mostly Powerful Cores
- Energy: Uncertain

# Memory Friendly Threads



- Execution Time: Uncertain
- EDP: Mostly Weaker Cores
- Energy: Weaker Cores

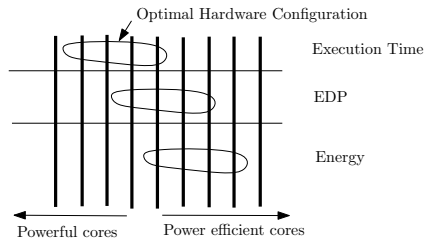
# Intermediate Threads



- Execution Time: Uncertain
- EDP: Uncertain
- Energy: Uncertain

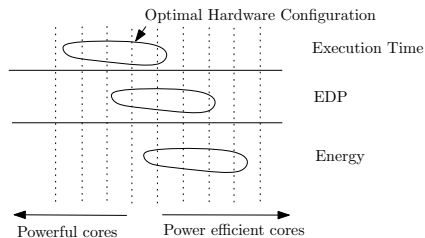


# Multiple CPU Types



- Execution Time: Uncertain
- EDP: Uncertain
- Energy: Uncertain

# CPU Types + DVFS + NUCA ...



- Relative Power: Unknown
- Execution Time: Uncertain
- EDP: Uncertain
- Energy: Uncertain

# Optimal Hardware

Given an input program, a user selected cost function and the possible hardware configurations, can we find an optimal hardware configuration which can run the program with minimal cost function.

- Static/Compile time: CHOAMP
- Dynamic: Execution time decisions : HMP Scheduling<sup>1</sup>
- Hybrid: Compile Time Instrumentation and Runtime Decision: PIE<sup>2</sup>, CES

---

<sup>1</sup>[http://www.arm.com/files/pdf/Heterogeneous\\_Multi\\_Processing\\_Solution\\_of\\_Exynos\\_5\\_Octa\\_with\\_ARM\\_bigLITTLE\\_Technology.pdf](http://www.arm.com/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf)

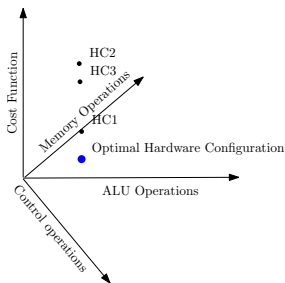
<sup>2</sup>Van Craynest et al, Scheduling Heterogeneous Multi-cores Through Performance Impact Estimation (PIE)

# Static v/s Dynamic Scheduling

- Static
  - Unknown variables
  - Unknown Task-Thread Mapping
  - Hardware Resource Reservation
- Dynamic
  - Runtime overhead
  - Scalability of the engine
  - Context switching with other processes
  - Accurate knowledge of workload

# Static Scheduling

- Unknown Loop bounds: Represent workload as a function of unknown variables
- Wasted (wait?) CPU cycles for synchronization constraints
- Handle large solution space (Hardware configurations)

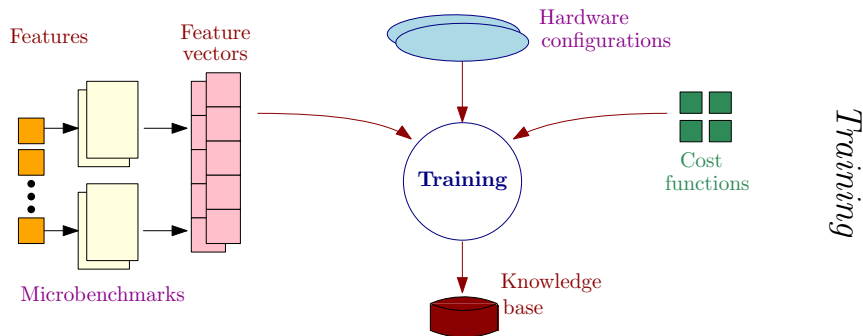


- A Trained Regression engine to identify the optimal solution from the solution space
- Feature set selection: relevant features

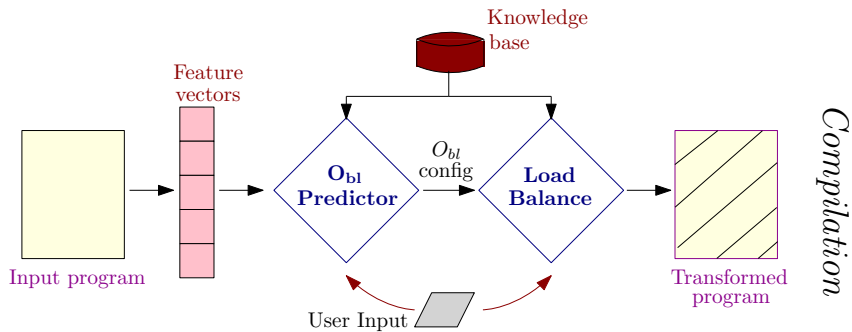
# Feature Selection

- Low level : based on Hardware dissimilarities of different types of cores
  - ALU operations
  - Memory operations
  - Branch operations
  - False sharing
- High Level: synchronization constructs provided by the language (OpenMP)
  - Barriers
  - Atomic
  - Critical sections
  - Flush operations
  - Reduction operations.

# CHOAMP: Training Phase

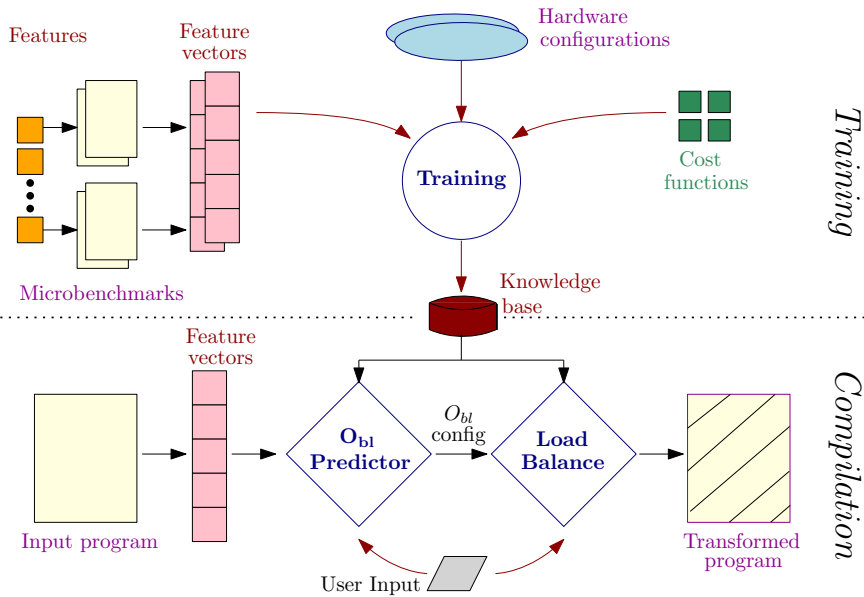


# CHOAMP: Compilation Phase

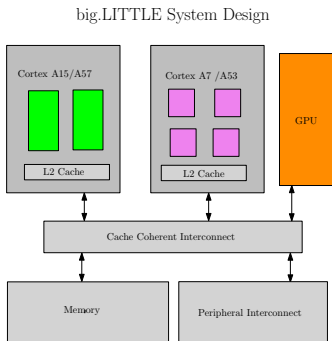




# CHOAMP: The Big Picture



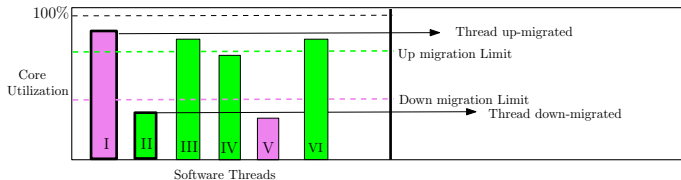
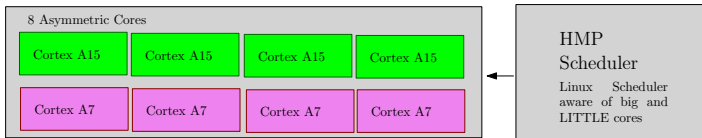
# Test Environment : big.LITTLE



- Asymmetric Multicore Architecture from ARM
- Targets mobile platforms which has strict power constraints

Core Types	Cortex-A7	Cortex-A15
Pipeline	simple 8 stage in-order	Out of Order, multi-issue
Frequency	600 - 1300 MHz	800 - 1900 MHz
Speed	1.9 DMIPS	3.5-4.01 DMIPS
Instruction Set	Thumb-2	

# HMP 3



- Integrated in Linux kernel Complete Fair Scheduling (CFS)
- Scheduling based on history of utilization
- Threads with higher utilization scheduled in big

<sup>3</sup><http://www.linuxplumbersconf.org/2012/wp-content/uploads/2012/09/2012-lpc-scheduler-task-placement-rasmussen.pdf>

# Implementation

- Hardware Configurations: Odroid XU3 <sup>4</sup>
  - Core Configurations: 4L4b, 0L4b, 4L2b, 2L2b, 4L0b
  - big Frequencies : 2GHz, 1.8GHz, 1.6GHz, 1.4 GHz, 1.2GHz
- Micro-benchmark generation: Python 2.7
- Dynamic Scheduling: HMP and CES <sup>5</sup> for dynamic load balancing
- Analysis and Transformation: IMOP <sup>6</sup>
- Regression Tool: Java Scientific Library <sup>7</sup>
  - Regression Engines: Linear, Quadratic and Gaussian
- Benchmark : NPB <sup>8</sup>

---

<sup>4</sup>[http://www.hardkernel.com/main/products/prdt\\_info.php?g\\_code=g140448267127](http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127)

<sup>5</sup>CES: Compiler Enhanced Scheduling: V S and Balachandran

<sup>6</sup>Nougrahiya and Nandivada, IMOP: <http://www.cse.iitm.ac.in/~amannoug/imop/>

<sup>7</sup>Flanagan M, <http://www.ee.ucl.ac.uk/~mflanaga/java/index.html>

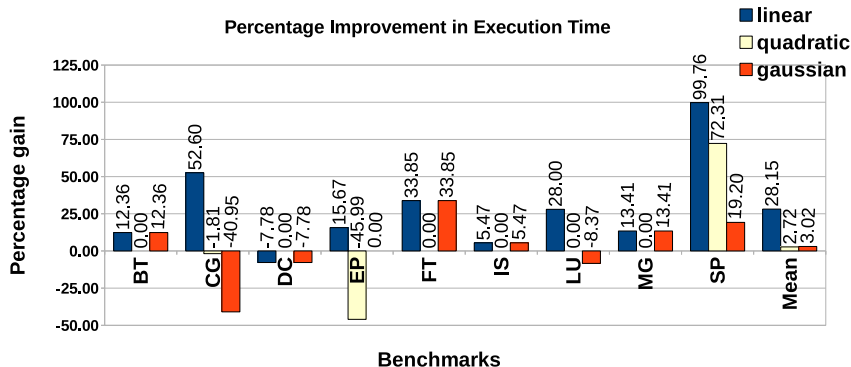
<sup>8</sup>Bailey et. al. The NAS parallel benchmarks-summary and preliminary results

## Predicted Configurations

**Table:** Optimal hardware configuration selected for NPB benchmarks computed by Linear regression. All LITTLE cores run at 1.4 GHz. The value in brackets shows the GHz frequency of big.

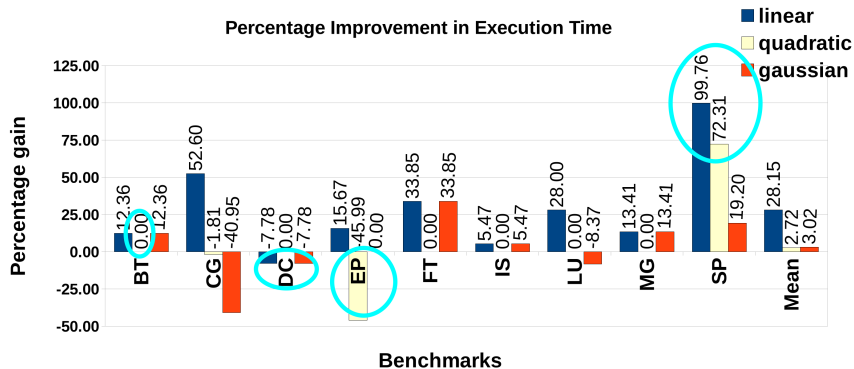
Benchmark	Configuration chosen for a cost function		
	Execution Time	Energy	EDP
BT	4L4b(1.8)	4L0b	4L4b(1.4)
CG	0L4b(1.8)	4L0b	0L4b(1.4)
DC	4L4b(1.8)	4L0b	4L4b(1.4)
EP	4L4b(1.8)	0L4b(1.2)	4L4b(1.6)
FT	4L4b(1.8)	4L0b	4L4b(1.6)
IS	4L4b(1.8)	4L0b	4L4b(1.4)
LU	4L2b(2.0)	4L0b	0L4b(1.4)
MG	4L4b(1.8)	4L0b	4L4b(1.6)
SP	0L4b(1.8)	4L0b	0L4b(1.4)

# CHOAMP: EXECUTION TIME



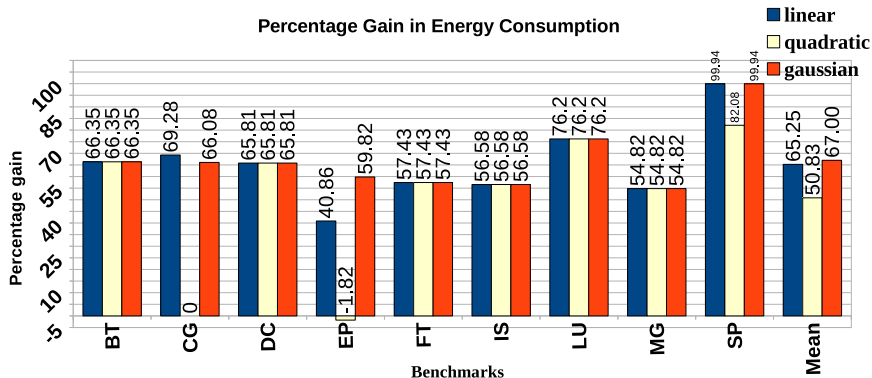
- On an average 28% improvement with Linear Regression Oracle

# CHOAMP: EXECUTION TIME



- BT: Selecting the same configuration as base case
- DC: Unknown Parallel Operations.
- EP: Highly parallelizable,  $\uparrow$  Threads  $\uparrow$  Gain
- SP: High Sync costs,  $\downarrow$  Threads  $\uparrow$  Gain

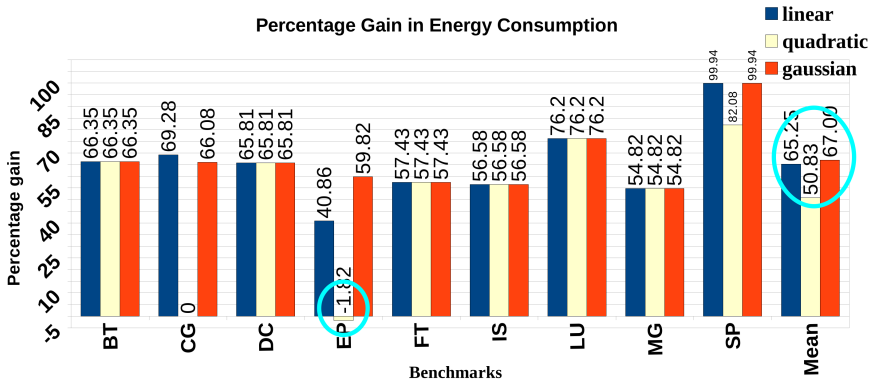
# CHOAMP: ENERGY



- On an average 65% improvement with Linear Regression Oracle
- Mostly Lower Configuration than base yield Higher energy gains

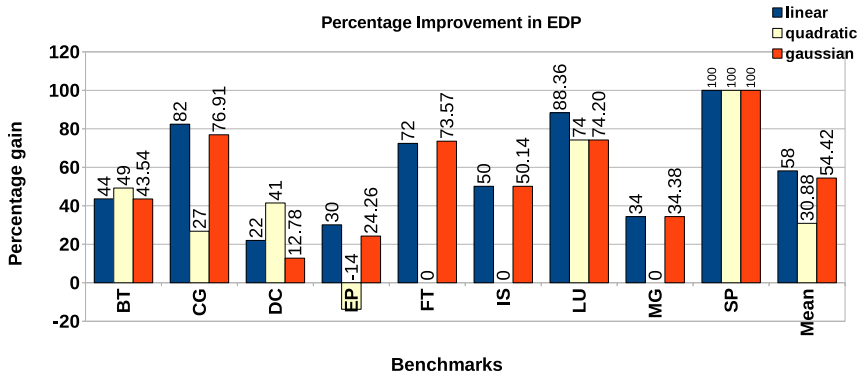


# CHOAMP: ENERGY



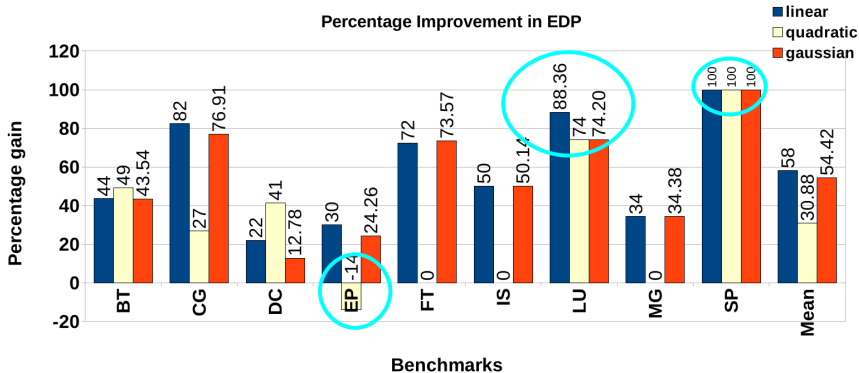
- EP: Highly parallel

# CHOAMP: EDP

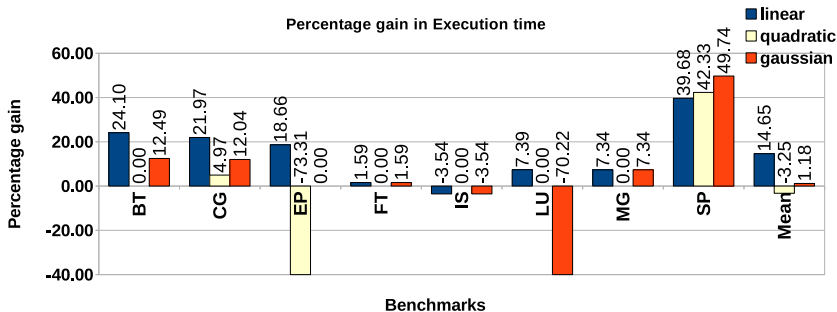


- On an average 54% improvement with Linear Regression Oracle
- Mostly a mid configuration yield good results

# CHOAMP: EDP

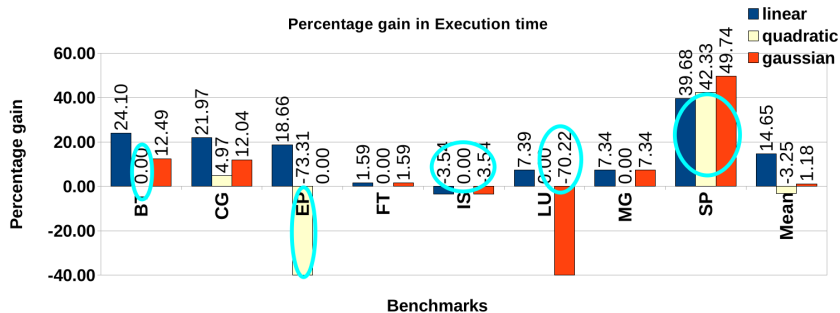


- LU: Gain higher than Energy, Execution Time negative
- SP: Rounded off



- The trained Oracle without knowledge of CES
- On an average 14% improvement in execution time

# CES + CHOAMP



- BT: Base configuration selected
- EP, IS, LU: Loss Amplified, Better Load balancing
- SP: Gain reduced, Lower Sync costs

# Conclusions

- Asymmetry aware compilation can produce a more efficient execution environment
- Predicts an optimal hardware configuration for a given input program
- On an average 28% improvement in execution time
- Average 62% improvement in energy consumption
- Average 58% improvement in EDP
- Average 14% improvement in energy consumption with CES
- Works well in tandem with dynamic scheduling algorithms (i) HMP (ii) CES.
- Future Work: Handle Improper parallelism, Recursive parallelism
- Future Work: Compiler/User directed Migration
- Future Work: Distributed AMP