OpenMP compiler for Heterogeneous Multicore Processors

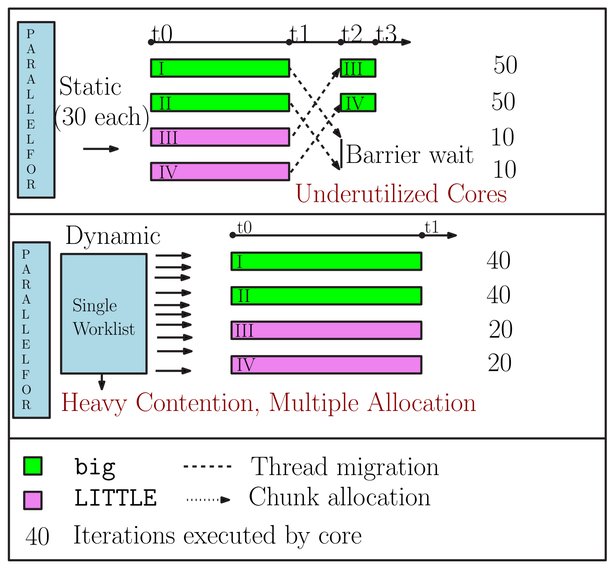
Jyothi Krishna V S and Shankar Balachandran
jkrishna @ cse.iitm.ac.in

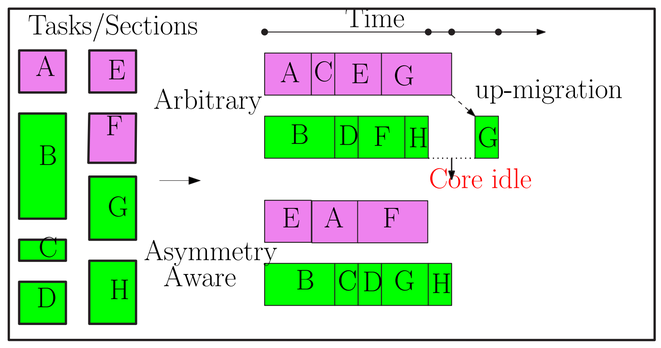
Introduction

- OpenMP compiler called CES that is aware of asymmetry in `big.LITTLE`.
- CES provides schedules to the run-time system.
- Handles:
  - Homogeneous Workloads: (i) OpenMP `for` construct. (ii) OpenMP parallel regions.
  - Heterogeneous workloads: (i) OpenMP `sections`.

	`LITTLE` core	**`big` core**
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 [1]	3.5-4.01 DMIPS

Motivations

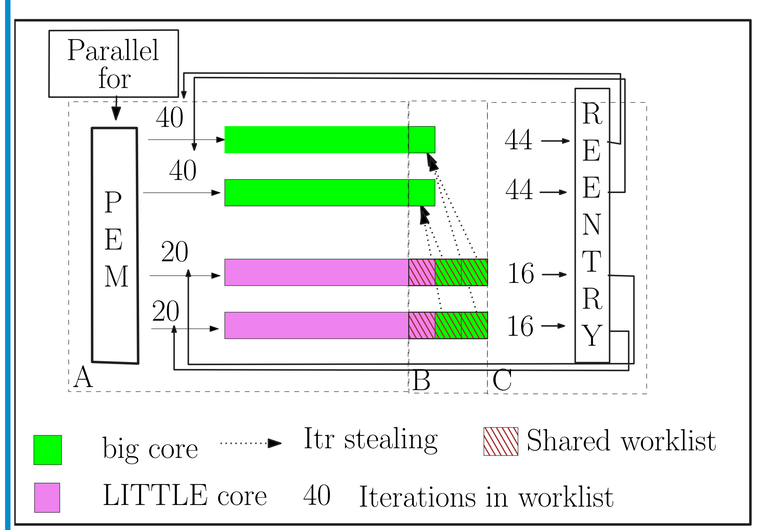




- HMP [2] scheduling solely based on history.
- Does not optimize for symmetric workload.

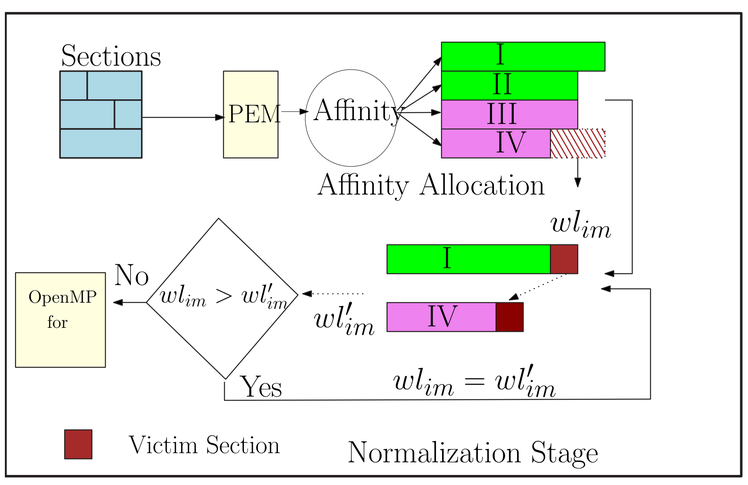
CES Transformations

OpenMP `for` construct



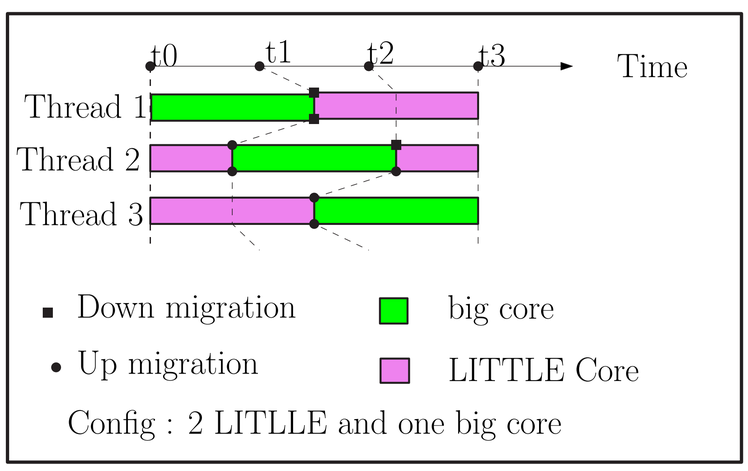
- Shared worklist protected using OpenMP locks.
- An atomic variable used to denote the state of worklist.
- Re-entry module added only when the parallel region is re-visited.

OpenMP `section` construct



- Allocation done in two stages (i) Affinity Allocation (ii) Normalization Allocation
- Converted to OpenMP for and later to worklist to follow the allocation.

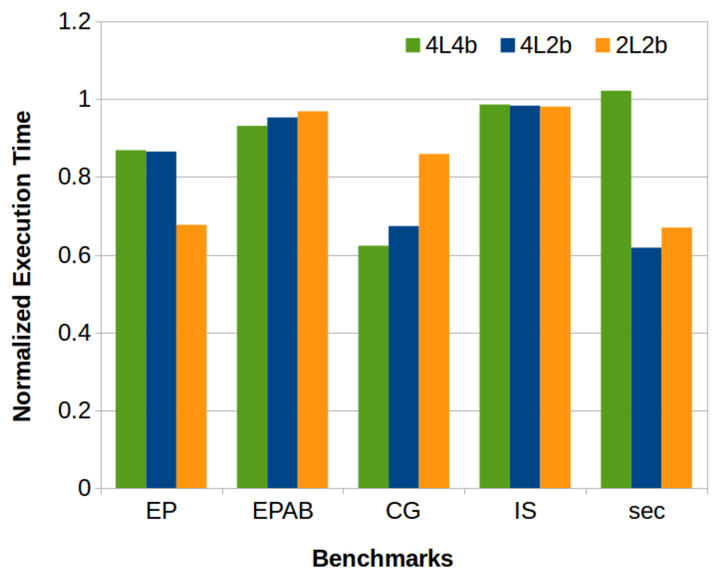
OpenMP `parallel` construct

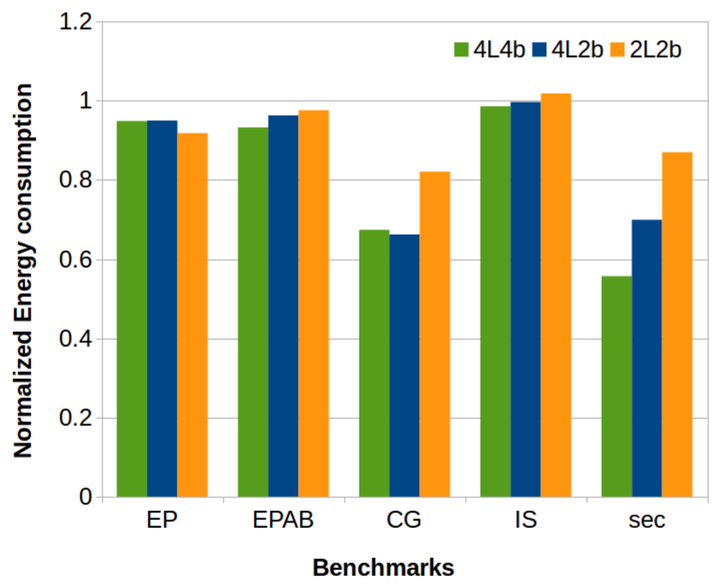


- Exposed two system calls based on the `big` and `LITTLE` migrations.
- Minimum Guarantee Point: Point at which the thread is ready to be down-migrated.
- Migration Point: Point t which the thread can be up-migrated.

Results

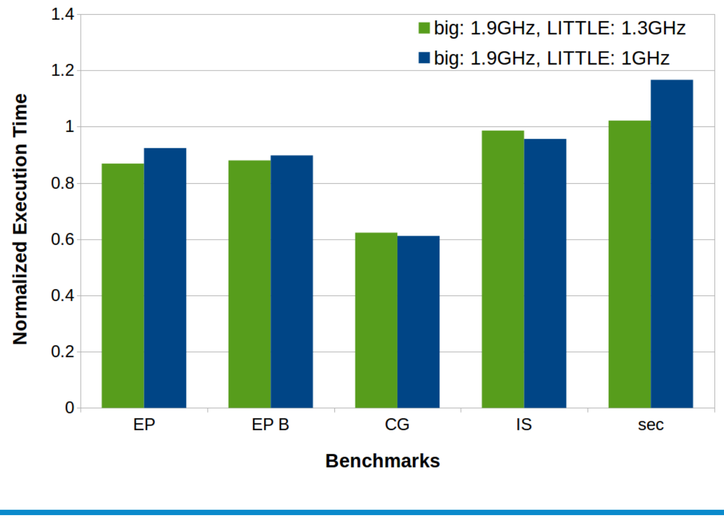
Different `big.LITTLE` combinations

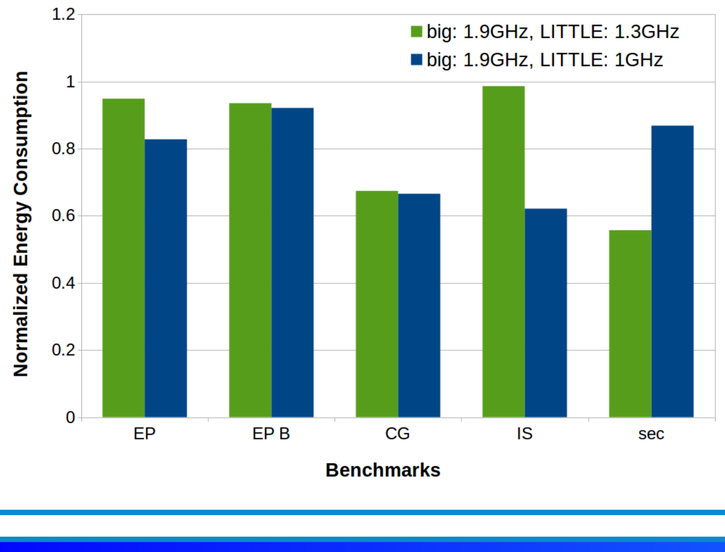




- The transformations realized using IMOP [3] a source to source OpenMP compiler.
- Standard NPB OpenMP benchmarks [5] along with two modified benchmarks (`EPAB`, `sec`) used ran in Odroid-XU3 [4] board.
- Three combinations of `big` and `LITTLE` configurations used.
  - 4l4b : 4 `LITTLE` cores and 4 `big` cores
  - 4l2b : 4 `LITTLE` cores and 2 `big` cores
  - 2l2b : 2 `LITTLE` cores and 2 `big` cores

Different frequency combinations for `big` and `LITTLE`





Results

- Average 18% reduction in execution time and 14% reduction in energy consumption.
- `sec` shows higher execution time (upto 16%), but with high reduction in energy consumption (upto 42%) due to affinity allocation.
- Without stealing module `IS` sometimes shows higher power consumption (upto 2%) than HMP + OpenMP static scheduling.

I am looking for summer internship opportunities.

References

[1] R P Weicker,*Dhrystone: A Synthetic Systems Programming Benchmark*
[2] Chung et al.,`http://www.arm.com/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf`
[3] Nougrahiya et al., *IMOP, a source to source compiler,* `http://www.cse.iitm.ac.in/~amannoug/imop/`
[4] Odroid-XU3,`http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127`
[5] Bailey et al.,*The NAS Parallel Benchmarks:Summary and Preliminary Results*

Conclusions

- Compiler enhanced scheduling shows advantage for both homogeneous and heterogeneous workloads.
- Future work involves handling more OpenMP constructs.
- Compilation for optimizing mobile applications.
- User handles for programmer to direct the optimization.