Paper summary

- this summary is for idea sketch, result and conclusion not included

- result/conclusion : ... anyway, Asymo is great. Period. (not needed to be summerized)

**AsyMO: Scalable and Efficient Deep-Learning Inference on Asymmetric Mobile CPUs**

**0. ABSTRACT**

a. What is Asymo?

Asymo : Thread pool implementation of DL inference framework on mobile CPUs

b. Current problems and corresponding solutions of Asymo

problem 1

Poor scalability -> cost-model-directed partioning and asymmectric aware scheduling

problem 2

Energy saving -> least-energy cost frequency based on data reuse rate of a model

**1. Introduction**

a. why not cloud (device -> cloud inference)

- network latency

- privacy problem when inference

b. GPU, cpu have similar performance on mobile GPU -> ONLY focus on CPUs for inference

c. Issues

1. currently -> do not fully utilize big-little CPU structure -> poor scalability and performance

2. energy inefficiency because of improper CPU frequency setting

- os unawareness of DL model

- not reponsive enough for small DL models

d. Root causes

1. unbalanced task distribution on AMP cores

- task distribution is not proportional to their capacity

- unbalance due to interference-prone characteristic of mobile environment

2. inferior task-partitioning

e. Challenges

1. hardware asymmetry -> unified block size partitioning and # task-based scheduling : bad

2. seperated cache btw big/little processors : accessing remote cache is time-consuming

3. high competition for small cache btw multiple processors

4. Interference-prone environment -> consd to avoid lagging thread

f. Approach of Asymo

use the fact that DL model is deterministic -> jointly consider DL structure and AMP CPUs

1. for performance scalability

- cost-model-directed partioning and asymmectric aware scheduling

- processor level -> core level partitioning for task size optimization (mnz MM latency)

- cost model : consider lots of aspect to estimate task-size impact on latency

2. least-energy frequency setting

- based on computation/memory intensive workloads on DL

- offline profile of E – curve and set frequency

**2. Background**

a. Parallelism in DL inference



- Two levels fo parallelism

1. inter-op parallelism -> time cost is sig-lower than intra-op ... ignore

2. intra-op parallelism

- # of intro-op threads(hardware) == # of cpu cores, scheduled by scheduler(OS etc)

b. MM partitioning

- MM is the major cost of DL inference -> proper partitioning for MM is critical for inference peformance



- selection of mc and nc(task size) : ATLAS -> to eq-div computation for the threads & cache size consd : not optimal for mobile AMP CPUs

c. Mobile AMP and OS DVFS

- big/little core has diff hardware chract, isolated power domain -> diff freq setting possible

- OS DVFS : big/little core freq setting base on timely chaning workload, togather

- short-run DL inference : not responsive enough (as mentioned above)

d. On-device DNN accelerators (why CPU are dominant hardware for on-divice inference)

- CPUs are always availiable

- AI accelerators ecosystem is closed and immature

- lack of flexibility

- technique generality of CPU

**3. Performance evaluation and motivation**

**3.1 Poor performance scalability on AMP**

- OS EAS : lack of workload understanding -> not capacity-proportional workload distribution

- unbalance between big and little processors

- unbalance within a processor(btw cores)

**3.2 DVFS mismatch**

- big mismatch between OS CPU frequency scaling and DL inference

**4. AsyMo system design**

**4.1 System overview**

a. Asymo design

- Asymo designed by both consd DL inference / AMP CPUs charcteristics

- DL inference charateristics

- deterministic execution

- embarrassingly parallel tensor operation (x consd critical path, balance with core capa)

- typical hardware behaviour characteristics (comp / mem intensive)

- mobile AMP CPUs characteristics

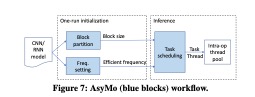
- asymmetric core capability

- seperated caches

- small caches

- interference-prone environment

b. Asymo workflow



**4.2 Cost-model-directed block partitioning**

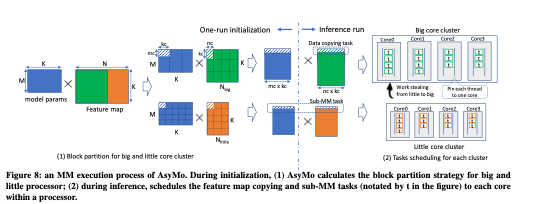
a. Design guidelines for cost model & actual model

- for better task balance, task size should be minimized

- to minize memory access, task size should be maximized (to avoid cache thrasing)



b. Partitioning for big and little processors



- processor level partitioning -> core level partitioning, why?

- proper block size is different at each processor

- to avoid costly data transfer between to processor : seperated cache

**4.3 Asymmetry-aware scheduling**

- schedules each task to the shortest thread queue in the corresponding processor

- work stealing : only from small -> big processor due to different task/cache size

- two types of task : data-copying and sub MM tasks

- data-copying : block-major contiguous space copying for cache locality

- schedule two types of task in same processor to avoid data movement btw processors

**4.4 Frequency setting for energy efficiency + 5. Energy profiling**

- Energy cost of workload : static + dynamic energy

- static energy : f ↑, E ↓

- dynamic energy: f ↓, E ↑

- Computation intensive/memory intensive workload comparision

- computation intensive : decrease when freq go up

- memory intensive : decrease rapidly after specific threshold

- offline profiling by enegy curve : find optimal frequency for energy efficiency

- little processor less sensitive to freq : fixed to highest at AsyMo