R1:

We are sorry that the reviewer does not see the contribution of our work. We consider this work novel and ideal for the ISPASS conference, as it is a thorough evaluation of AMCs. It extensively highlights the limitations of the state-of-the-art OS scheduling approaches (e.g. GTS) and demonstrates how to tackle them by using task-based parallel runtimes. This work is novel as it demonstrates for the first time how the scheduling responsibility should be distributed across different layers of the software stack. To our knowledge, none of the previous works on AMCs provides such a thorough and reliable performance, energy and power evaluation of real highly parallel applications.

R2:

Our work focuses on the examination of the new mobile-based AMCs for their use in the next generation multi-cores. As the main challenge for future multi-core system design is energy efficiency, many researchers are pushing towards the use of the energy efficient AMCs to build such systems. However, our work is not tied to ARM. Our insights apply to AMCs in general. Regarding ARM, their HPC software ecosystem is maturing at all levels: compilers, libraries, tools; and important systems are going to be based on ARM(www.nextplatform.com/2016/06/23/inside-japans-future-exaflops-arm-supercomputer, http://www.theregister.co.uk/2017/01/16/arm\_mont\_blanc\_hpc\_european\_commission/, www.arm.com/hpc). Looking at ARM in particular, with HPC applications:  
-N. Rajovic,…: The Mont-Blanc Prototype: An Alternative Approach for HPC Systems, SC16  
-N. Rajovic,…: “Tibidabo1: Making the case for an ARM-based HPC system”, FGCS2014  
-N. Rajovic,…: “Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?”, SC13  
-N. Rajovic,…: Experiences With Mobile Processors for Energy Efficient HPC, DATE’13  
-Butko,…: “Design Exploration For Next Generation High-Performance Manycore On-chip Systems: Application To big.LITTLE Architectures”, ISVLSI’15  
-K. Chronaki,…: Criticality-Aware Dynamic Task Scheduling for Heterogeneous Architectures, ICS2015  
-E.Castillo,…: CATA: Criticality-Aware Task Acceleration for Multi-Core Processors, IPDPS2016

R2&R3-PARSEC-suitability:

We agree with both Reviewers that we could improve the motivation. PARSEC is the best candidate for this evaluation. The mention of supercomputing and HPC is to show that previous works have already used mobile SoCs outside the mobile market. We selected PARSEC because it consists of real highly-parallel applications, not tied to a particular field of study, making the study more relevant.

R3&R4-Fugures:

Figure 4 is not normalized. Figure 5 is normalized with respect to the energy with static-threading using 4 little-cores. We plan to use the same baseline for all figures.

R4:

ii.If the task/chunk sizes are the same then the difference comes from the fact that loop-based dynamic scheduling requires a barrier at the end of each loop. A task-based implementation removes that barrier by using task-dependencies across loops/parallel regions, resulting in less waiting. If the loop-based case is well-balanced, then, at same task/chunk sizes performance should be similar. The key insight is that dynamic task/loop scheduling is required in these systems. Our contribution is the detailed performance analysis of these options, considering software aspects such as task/chunk size, which seems relevant for ISPASS.

iii.The power ratio between big and little-cores was calculated during this study and ranges from 2x to 2.5x. While writing we saw that it didn’t add much insight to the reader we decided to omit it, but it is trivial to add it back on Table1. We don't have a definitive answer on why the power does not increase proportionally to the number of cores. We can speculate that, adding little-cores, the task-based implementation gets much better utilization than the static case, but probably not 100%. This is hard to measure given the complexity of the evaluated codes. Our guess is that there is a small utilization drop of the big-cores that is compensated by the addition of the little-cores leading to a similar power than the only big-core scenario. The locality effects of adding a second cluster of cores with another L2 cache introducing potential invalidations in the big-core caches could explain why big-cores lose some utilization.

iv.We use the PARSECSs suite that contains the task-based versions of PARSEC. In this suite there are 10 of the original PARSEC benchmarks. From these 10 benchmarks we exclude freqmine as it has no pthreads version to represent static-threading and GTS.

Major-ii.We tried to use multiple frequencies but we saw that the outcomes of these studies do not differ between different settings. This is because the most important reason behind the observed performance ratios is the micro-architecture (big=out-of-order, little=in-order). We stick with the specific frequency combination because when we set the frequency of big-cores to the maximum(2GHz), the machine was switched-off due to high temperatures, because of long running times.  
Minor-all:We will address all minor comments