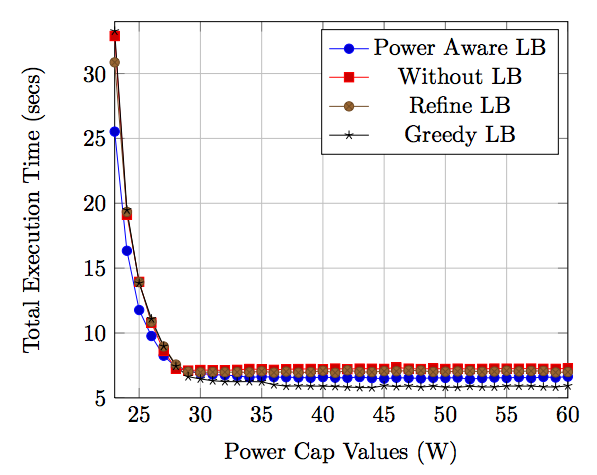
**- How our LB performed w.r.t other load balancers w.r.t time and the reasons**

To analyze the performance of our power-aware load balancer, the metrics of heterogeneity are compared. This comparison is done with the performance of Jacobi2d execution in the absence and presence of existing load balancers like RefineLB and GreedyLB. Power aware LB has proved to be more efficient than other LBs. The main reason is the frequency awareness our load balancer takes into account while trying to minimize the load among the nodes. RefineLB and GreedyLB are not aware of the differences in frequency that exists at lower power caps. This makes them power unaware. They only consider the workload in terms of object wall time while balancing the load. When migrating Chare objects from one node to other, our load balancer works takes in to account the existing workload and the time taken to complete the execution for that amount of workload, at both sending and receiving ends. This makes it perform well even in the presence of heterogeneity at lower power caps.

**- Detailed explanation of the results and how we compared with other LBs and possible reasons why others could have performed better in some cases and some times not.**

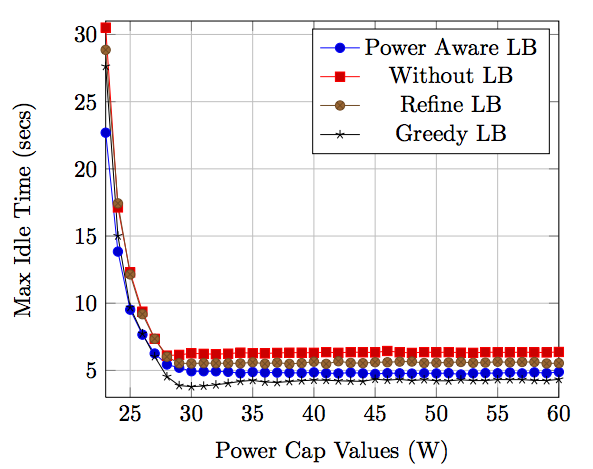
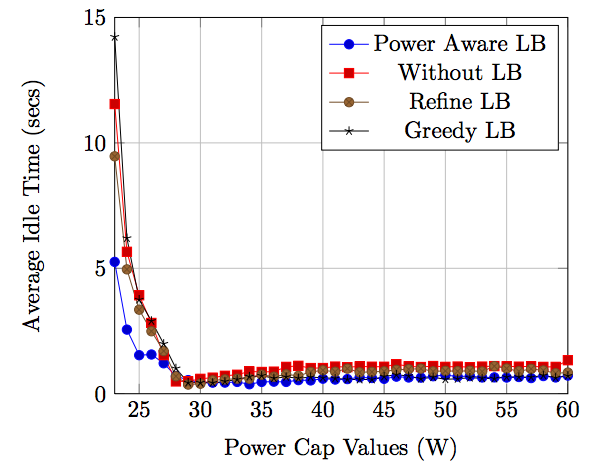
****

A maximum speedup of 1.3x was seen with the use of power aware load balancer instead of RefineLB and GreedyLB. The total execution time when run without any load balancer at 23W power cap is ~34sec. This is somewhat reduced in case of RefineLB and GreedyLB. But the power aware LB is able to decrease is to ~25sec as shown in figure [?].

There is little or no speed up in cases of higher power caps when using the power aware load balancer. The reason behind this is the absence of heterogeneity at higher power caps. The product of object wall time ‘w’ and the workload ‘x’ comes to be almost the same, and so it leaves power aware LB no scope of further balancing the workload based on frequency. Load imbalance is observed only at lower power caps. Hence, there is very little migration that power aware load balancer does at higher power caps. This makes GreedyLB perform better at higher power caps, as it takes the greedy approach of freshly loading the heaviest object with lightest loaded processor.

GreedyLB and RefineLB are unaware of the differences in frequencies of different nodes that exist at lower power caps. These do not take into account the frequency of each node and assume that all the nodes in the cluster have identical performance capability. There is equal weightage given to each node in GreedyLB and RefineLB. Since the only parameter these power unaware LBs take into account is the object wall time, they do not perform better than our load balancer in the presence of heterogeneity among the nodes.

**- Detailed explanation of the results w.r.t average idle time reduction and maximum idle time reduction and possible reasons when we didn't perform well. Please be as detailed as possible**

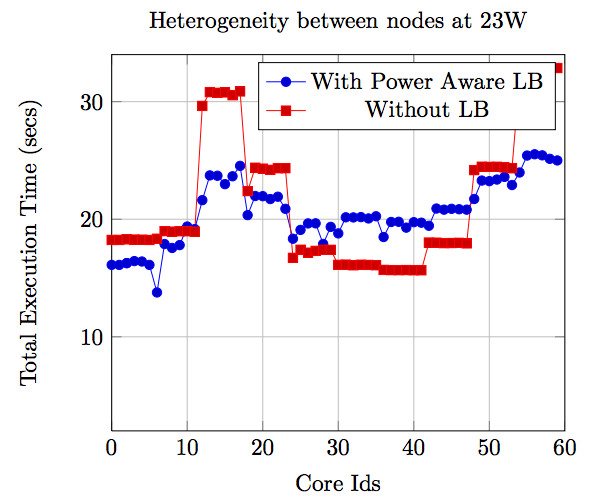
****

Performance of power aware LB is further analyzed with respect to other LBs in terms of heterogeneity factors – average idle time and max idle time. Power aware LB has out performed the other existing power unaware LBs in terms of average and max idle times. The average idle time is reduced using the power aware LB by as high as 60% when compared to the average idle time of the run without any LB. If compared with RefineLB and GreedyLB, the reduction is as high as 54% and 63% respectively. The average idle time at 23W comes to be ~12,10,14 and 5sec for runs without LB, with Refine, Greedy and power aware LBs respectively as shown in figure [?].

The reduction in maximum idle time is as high as 25% in the presence of power aware LB. It is ~30sec in the absence of any load balancer and it comes down to ~23sec when run with power aware LB at a low power cap of 23W as shown in figure [?]. We could have further decreased the maximum idle time. But due to an off node in the cluster with an unexpected behavior led to an increased max idle time. This has to be further investigated.

Power aware LB performs equally well in cases of higher power caps when there is little or no heterogeneity among the nodes. One positive aspect here is that power aware load balancer does not try to do unnecessary balancing at higher power caps that could have led to an increase in the average or max idle leading to an increase in the total execution time.

**- Finally explanation about our execution time distribution in comparison with without LB (and also RefineLB. Please reach out to Sandeep for this plot). And how we could have improved further.**

****

One another observation of the total execution time is done on a per core basis at the low power cap of 23W. The power cap is taken to be low so that there is some visible scope of decreasing the amount of heterogeneity. This depicts the amount of heterogeneity among the nodes. There is a lot of difference to be seen in the amount of total times taken by each node, with each node containing 6 cores, in the absence of any load balancer. The total execution time ranges from 15-35 sec for the run without any load balancer. This shows that there is a lot of scope for redistributing the load among the nodes in the cluster.

The power aware load balancer proves to redistribute the workload as intended. As depicted in figure [?], the range of total execution time has been brought down 15-25 sec with most of the nodes close to the average total. This could further be optimized among the nodes by making use of node specific information or hard coding based on behavior of how well a particular node is performing. But these options are not incorporated as they make the load balancer usable only for specific applications running on known clusters.

**- Summarize the whole work and give the conclusions and the future recommendations in detail.**

We observed that the execution time decreases with decrease in CPU power, but the rate at which it decreases was seen to be different with some nodes performing better than others, leading to performance heterogeneity. This leads to some nodes exhibiting higher performance than others, and thus such nodes tend to have higher idle-times waiting for other slower ones to complete their iteration, before the next iteration could be started. Heterogeneity is measured in terms of average idle times of the nodes in the cluster. This heterogeneity becomes significant when the power is pulled down to the allowed minimum. This load imbalance leads to more wait times among the nodes and thus there is scope to minimize the imbalance by having a power aware load balancer that gathers information of the initial few iterations and then based on the collected information about the frequency and workload of different nodes, tries to bring down the idle times.

To help mitigate this performance imbalance, we developed a power-aware load balancer that helps in minimizing the heterogeneity. Our load balancer performed better than the existing power-unaware load balancers in the Charm++ framework. It helped reduced the existing amount of heterogeneity and achieve a maximum speed up of 1.3x with respect to other load balancers when used with Jacobi2d application.

The current limitation with this load balancer is that it does not take into account the size of the workload on a particular node. This limits its usage in the applications where the object size varies with time. One such applications that we studied for heterogeneity at lower power regions was LeanMD, where there is particle migration happening as the application executes leading to different object sizes at different times.

Our future work aims at making our power-aware load balancer aware of the changing size of the workload, and thus incorporating this parameter while balancing the workload among the various nodes. This work also involves periodic invocation of the load balancer in order to keep the idle times minimized. This has to be done carefully, keeping the overall load balancing workload to the minimum.