1. Today I am going to talk about my work related to statistical methods for optimal job scheduling and resource utilization in computing systems.
2. Our work is motivated by 2 main trends in computer science.

Energy is increasingly important. Crucial for datacenters, personal computers, laptops and mobile phones alike. Every system is constrained by power dissipation. Reducing energy consumption reduces operating costs in datacenters. Similarly, it increases battery life in laptops and mobile devices.

These days most of the devices are often underutilized. Here is an image from 5000 servers during 6 month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels. On mobile phones such as Samsung Exynos octa, the device cannot run at full utilization for more than a second before it melts.

1. Therefore to make the system more configurable over different utilization and energy demands, Hardware designers expose these resource to software to optimize the resources. Configuration could be a form of resource allocation. For instance, in this picture we have 32 cores, 16 clock speed and 2 memory controllers. Hence the configuration space could just be the Cartesian product of each of the cores, clockspeed and memory controller.
2. In my thesis we have looked at some principled statistical approaches to solve this problem. We broke it down into various sub problems. First, how do we build statistical model for a single application on the system. Second how to make that application work under dynamic scenarios and finally how do we extend it to multiple applications.
3. First is done in masters thesis and the last one was presented at my candidacy exam, and I will go over them briefly and I am going to spend most time in the middle one.
4. We want to automate this process of resource allocation to minimize its energy consumption. We would like the system to adaptively figure out the best optimal configuration at runtime based on its utilization levels. But this requires the knowledge of power and performance at each of those configurations. So, this is the problem that we want to solve. We want to find this profile for each application during its run-time.
5. Let us find out why is this be a difficult problem? It may not be feasible to run the application across the entire configuration space since it could be quite large and the application might finish before that. Moreover, we can not get it from other machines since its also machine dependent. Also it is input dependent, it could change with different inputs for example the video streaming application x264 would perform differently for different video inputs.
6. Lets us look at an example of kmeans algorithm. Here is a contour plot of performance rate for different cores and clockspeed. The lighter color describes better performance here. We would like to find the configuration with the highest performance. But the surface is very complex. The performance around the 8 cores is mostly better than its neighborhood. This is due the phenomena where due to the intersocket communication between the cores, addition of the cores does not improve but rather decreases the performance. Also, the surface has many local solutions, hence simple solutions may not work in this case.
7. Now, we present our solution to the problem. We call our solution LEO (learning for energy optimization). We use the profiling data for the other previously seen applications combined with very small partially observed data of the new application, in order to predict the performance profile of this new application.
8. Here are some results of our method. We use our method to predict the performance for the kmeans clustering algorithm. On left hand side we have the plot of performance rate for different configurations. And we observe that we are able to predict the performance pretty well with very high accuracy.
9. Now we have examples of some other applications, SWISH which is a websever and its performance suffers from hyperthreading, therefore after 16th core the performance diminishes. At the bottom we have X264 whose performance stays the same after 16th core because it does not utilize hyperthreading.
10. Performance is measured as application iterations (or heartbeats) per second.

Explain what the charts represent. What is x axis, performance. Saw tooth nature. Picture of tensor.

1. So far we know how to estimate the performance and power curve for an application. Now we would like to use these estimates to make the resource allocation in dynamic scenarios with varying utilization requirements.
2. So we now move on to some mobile applications. This is a contour plot for an application called bodytrack which is a computer vision application. This is a typical benchmarks. So similar to previous contour plots, this plot has cores on x-axis and DVFS setting on y axis. There are 2 types of core, big and little. Little cores are supposed to be energy efficient and good for lower utilization levels and big cores are good for high performance tasks. And if the utilization levels change in a dynamic scenario, how do vary the resource allocation. On the plot in right hand side we have those 2 scenarios. While the Bodytrack is running, we start another application on the big core and a dynamic model can detect that change and choose another configuration so that the application can maintain its performance but if we do not have this control mechanism a simple learner cannot adjust its resources for this change.
3. Now we have another benchmark which has a very atypical behavior, its called STREAM. Now this is a memory intensive benchmark. And a general hueristic would say thatwe should run memory intensive benchmarks on little core because they don’t need higher clockspeeds which completely false in this particular case. We should spend all the time on big cores and none on the little cores. Therefore a learning mechanism can detect but a controller without the explicit knowledge of STREAM would switch between big and little cores and miss a lot of deadlines.
4. Therefore we present CALOREE which can provide an abstraction for the learning mechanism with an adaptive controller. A adaptive lightweight controller runs on the mobile device and it send some application samples to the server and gets back a model from the server which the controller can use in the future to do better resource optimization.
5. This is a simple multiple input multiple output controller. Where x is the controller’s hidden state, u is the signal generated by the system and y is the output that is the goal. In order to convert it into an adaptive controller we would have to estimate these big matrices A,B and C periodically which is inefficient. So Connor’s system called POET came up with this idea of summarizing the control problem by adding a layer of indirection. Instead of maintaining performance they maintain a normalized performance speedup. And the output speedup is later used to determine the resource configuration which is most energy efficient.
6. 26. Pole is like the inertia of the system as in how fast should system react to dynamic changes. So if the pole is too small the system may not converge and if its too large it might converge too slowly. Since, we provide noisy estimator for power and performance, we expect the pole to increase by certain amount to account for the extra noise in the system. So we prove that if the estimator for performance have sigma amount of variance and S hat denotes the estimator. The pole should atleast be as large as this quantity for the system to converge. Empirically we observe that a max over this pole and this quantity works the best.

19. CALOREE converts Eqn. 1 into a single-input (performance), single-output (speedup) controlling using A = 0, B = *b*(*t*), C = 1,u = *speedup*, and *y* = *perf*; where *b*(*t*) is a time- varying parameter representing the application’s *base speed*— the speed when all resources are available—and *per f* is the measured performance.