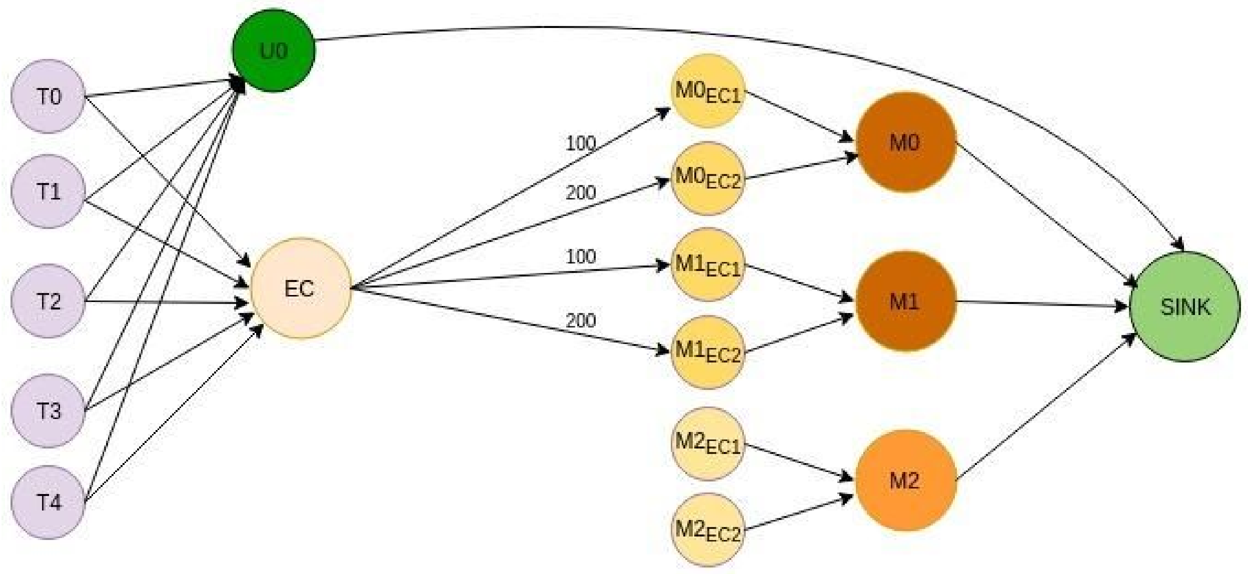
CPU Cost Model:

In CPU cost model the task equivalence class(EC) gets created based on the task's cpu and memory request. Each machine will have a set of predefined number of machine ECs($M0_{EC1}$,$M0_{EC2}$,..,$M2_{EC2}$) to favor load distribution among the machines during each scheduling iteration. The arcs between EC and machine ECs are drawn based on three categories of affinity selectors: hard, soft and complex constraints. Cost on the arcs from EC to machine ECs are assigned based on the available resource on that machine and each machine EC index.

Hard Constraints:

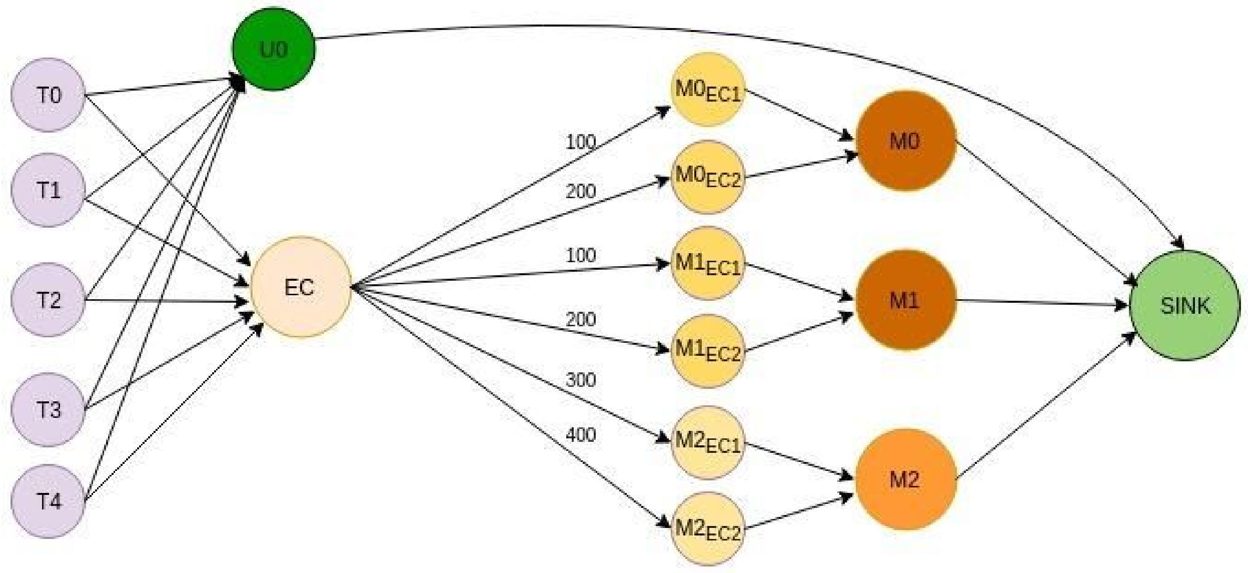
In case of hard constraints a task can only be placed on a machine if it meets requirement. Figure 1 shows a flow graph with hard constraints. Here machines M0 and M1 meet requirements and machine M2 do not meet requirements so arcs are drawn from EC to machine ECs only for machines M0 and M1. Example below shows the incremental costs on arcs drawn to each machine ECs of a machine which helps in achieving load distribution in a scheduling round. Based on the flow graph created tasks T0,T1,T2 and T3 gets scheduled on $M0_{EC1}$, $M0_{EC2}$, $M1_{EC1}$ and $M1_{EC2}$ respectively. Task T4 stays unscheduled until it gets any of the machine ECs available from machine M0 or M1.



(Figure 1: Example for hard constraints)

Soft Constraints:

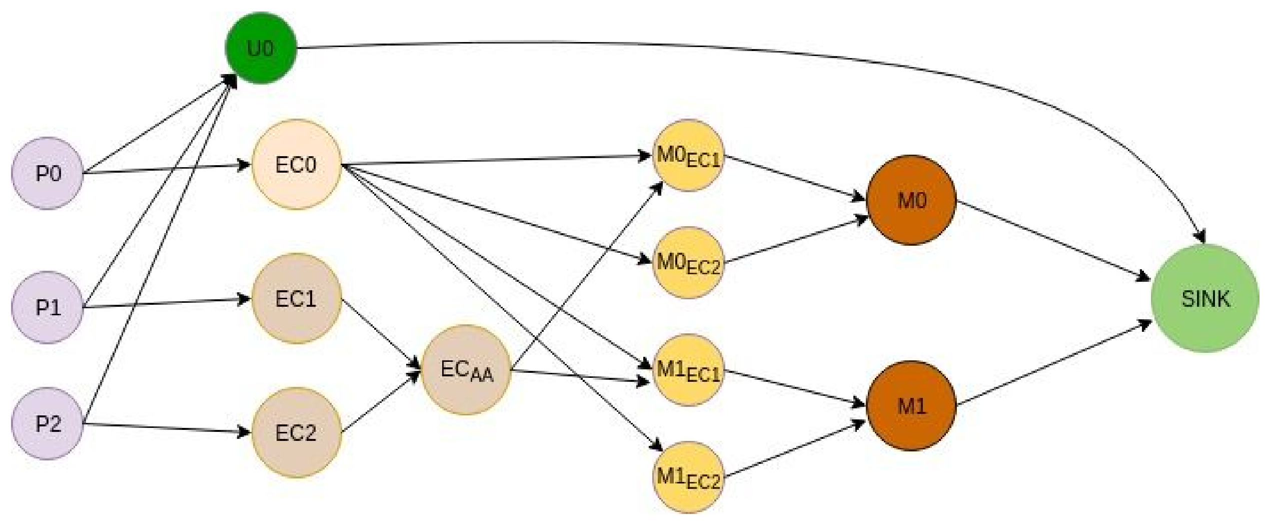
In case of soft constraints a task has a preference to meet certain requirements but can also be placed in a machine which does not meet requirements for the same cpu memory request. Figure 2 shows a flow graph with soft constraints. Here machines M0 and M1 meet requirements and machine M2 do not meet requirements. The machine ECs from M0 and M1 are the preferred machine ECs and the ones from M2 are non-preferred machine ECs. A task should always prefer the preferred machines ECs first unless all preferred machine ECs are exhausted, so the cost on arcs drawn from EC to non-preferred ECs should be always greater than the cost on any arc drawn from EC to preferred machine ECs. Based on the flow graph created tasks T0,T1,T2 and T3 will be placed on preferred machines i.e. M0 and M1. Task T4 gets placed on machine M2 as all other preferred machine ECs are already in use.



(Figure 2: Example for soft constraints)

Pod Anti-Affinity:

In case of pod anti-affinity no two pods of same anti-affinity label should be placed onto the same machine. Figure below shows an example of the design we have tried to implement pod anti-affinity. Pod P0 does not have any anti-affinity labels. Consider two different pods P1 & P2 having same pod anti-affinity labels. EC1 & EC2 are created based on cpu, memory and labels of pod P1 & P2 respectively. EC_{AA} is created based on common pod anti-affinity labels of P1 & P2. The arcs from EC1 and EC2 are drawn to EC_{AA} if both EC1 & EC2 have the same pod anti-affinity label as EC_{AA} is created on. EC1 & EC2 will have set of machine ECs which satisfy cpu, memory and labels request for P1 & P2 respectively. But EC_{AA} will only have mappings to the common set of machine ECs between EC1 and EC2. And from that set of common machine ECs again arcs from EC_{AA} are drawn to only one machine EC of each machine with max capacity one to prevent pods having same pod anti-affinity label to be scheduled on the same machine. In case where we don't have a common set of machine ECs between EC1 & EC2 there is no need to create a pod anti-affinity EC like EC_{AA}.



(Figure 3: Example for pod anti-affinity)

This design has many problems as demonstrated by few scenarios below.

1. When we filter out common machine ECs between EC1 & EC2 and draw arcs from EC$_{AA}$ to those filtered common machine ECs only then there is a possibility that a pod may not be placed in the best fit machine available. For example consider a case where EC1 can be mapped to both M0 and M1 having M1 as the best fit for pod P1. And EC2 can be mapped to only M0. In EC$_{AA}$ we filter out the common machines between EC1 and EC2 so we can map EC$_{AA}$ to only M0 as it is the only common machine between EC1 and EC2. Hence M1 which is the best fit machine for P1 is not considered for placement from EC$_{AA}$.
2. Assume a scenario having three pods P1, P2 & P3. Each one has different cpu, memory and label request. P1 has two pod anti-affinity labels for example key=security,value=S1 and key=security,value=S2. P2 has one pod anti-affinity label key=security,value=S1. P3 has one pod anti-affinity label key=security,value=S2. In this case we end up creating two separate anti-affinity ECs one satisfying key=security,value=S1 and other satisfying key=security,value=S2. There is a chance that P1 and P3 gets place in the same machine which will be against pod anti-affinity rule.

There might me many more scenarios where it will not work as expected.