(a) Let n be odd, $k = n^2$, and represent the set of basic processes as the disjoint union of n sets $X_1, \ldots, X_n$ of cardinality n each. The set of processes P_i associated with job J_i will be equal to $X_i \cup X_{i+1}$, addition taken modulo n.

We claim there is no perfectly balanced assignment of processes to machines. For suppose there were, and let Δ_i denote the number of processes in X_i assigned to machine M_1 minus the number of processes in X_i assigned to machine M_2. By the perfect balance property, we have $\Delta_{i+1} = -\Delta_i$ for each i; applying these equalities transitively, we obtain $\Delta_i = -\Delta_i$, and hence $\Delta_i = 0$, for each i. But this is not possible since n is odd.

(b) Consider independently assigning each process i a *label* L_i equal to either 0 or 1, chosen uniformly at random. Thus we may view the label L_i as a 0-1 random variable. Now for any job J_i, we assign each process in P_i to machine M_1 if its label is 0, and machine M_2 if its label is 1.

Consider the event E_i, that more than $\frac{4}{3}n$ of the processes associated with J_i end up on the same machine. The assignment will be nearly balanced if none of the E_i happen. E_i is precisely the event that $\sum_{t \in J_i} L_t$ either exceeds $\frac{4}{3}$ times its mean (equal to n), or that it falls below $\frac{2}{3}$ times its mean. Thus, we may upper-bound the probability of E_i as follows.

$$\begin{aligned}
\Pr[E_i] &\leq \Pr[\sum_{t \in J_i} L_t < \frac{2}{3}n] + \Pr[\sum_{t \in J_i} L_t > \frac{4}{3}n] \\
&\leq \left(e^{-\frac{1}{2}(\frac{1}{3})^2}\right)^n + \left(\frac{e^{\frac{1}{3}}}{\left(\frac{4}{3}\right)^{\frac{4}{3}}}\right)^n \\
&\leq 2 \cdot .96^n.
\end{aligned}$$

Thus, by the union bound, the probability that any of the events E_i happens is at most $2n \cdot .96^n$, which is at most .06 for $n \geq 200$.

Thus, our randomized algorithm is as follows. We perform a random allocation of each process to a machine as above, check if the resulting assignment is perfectly balanced, and repeat this process if it isn't. Each iteration takes polynomial time, and the expected number of iterations is simply the expected waiting time for an event of probability $1 - .06 - .94$, which is $1/.94 < 2$. Thus the expected running time is polynomial.

This analysis also proves the *existence* of a nearly balanced allocation for any set of jobs.

(Note that the algorithm can run forever, with probability 0. This doesn't cause a problem for the expectation, but we can deterministically guarantee termination without hurting the running time very much as follows. We first run k iterations of the randomized algorithm; if it still hasn't halted, we now find the nearly balanced assignment that is guaranteed to exist by trying all 2^k possible allocations of processes to machines, in time $O(n^2 \cdot 2^k)$. Since this brute-force step occurs with probability at most $.06^k$, it adds at most $O(n^2 \cdot .12^k) = O(n^2 \cdot .12^n) = o(1)$ to the expected running time.)

[1] ex41.971.873