A note on dependency graph

Guoli Ding and Rod Tohid

August 24, 2020

Suppose we are given a user program $\mathbb{P}$, which consists of a sequence of instructions (on how variable arrays are processed). Our goal is to schedule these instructions on p (a fixed number) processors. Our idea is to define a dependency graph and then apply known scheduling algorithms to this graph.

To define the dependency graph, we first need a clear definition on jobs. A **job** is a set of instructions of $\mathbb{P}$ that can be grouped together and can be viewed as a single task. We insist that every instruction of $\mathbb{P}$ belongs to precisely one of these jobs [i.e., no instruction is left out and there is no overlap between jobs]. Note that we will schedule jobs, not instructions [unless we happen to consider every instruction a job].

After knowing what the jobs are, we need to know how these jobs are related. Let $J_1, ..., J_n$ be the jobs. We consider a typical job J_k. This job may involve in several arrays, say $A_1, ..., A_t$. For example, J_k could be the instruction: $A = X * Y$. Then the involved arrays are A, X, Y.

For each A_r ($r = 1, ..., t$), we need to know when was the last time the array gets updated. The instruction that updates A_r is part of a job, say J_i. We see that J_k cannot start before J_i is done (since J_k needs A_r, which is produced by J_i). In this sense we say that J_k **depends** on J_i.

Suppose $A_1, ..., A_t$ lead to t jobs $J_{i_1}, ..., J_{i_t}$. Then we see that J_k depends on only these t jobs, and not any other jobs (since J_k involves in only t arrays $A_1, ..., A_t$, and these arrays are updated by $J_{i_1}, ..., J_{i_t}$).

Now we are ready to define our **dependency graph** $G_{\mathbb{P}}$, which is a directed (acyclic) graph:

- the vertices of $G_{\mathbb{P}}$ are precisely the jobs $J_1, ..., J_n$, and
- the arcs are determined as follows:

  for each job J_k, we determine jobs $J_{i_1}, ..., J_{i_t}$ as described above;
  then $J_{i_1}J_k$, ..., $J_{i_t}J_k$ are the arcs directed to J_k.

If we run this for $k = 1, ..., n$ then we obtain all arcs of $G_{\mathbb{P}}$.