Consideración: ¿Nodos slave como nodos libres o asociados a nodos master?

Decidimos que lo mejor sería considerar a los nodos slave como nodos libres, manteniendo un estado global de todos sus recursos en todos los masters. Básicamente, elegimos esto porque después de considerar las dos alternativas, vimos que mantener los nodos slaves asociados a los masters traía los siguientes inconvenientes:

- Habría que definir alguna forma de mappear los nodos slaves a los masters (para que todos los masters sepan qué nodo va con quién, y todos los slaves sepan a qué master acudir).
- También habría que definir una forma de que los nodos slaves se redistribuyeran en caso de que un master muriera o se agregara al cluster.
- Si se considerarán más recursos que sólo memoria, el scheduling de jobs se volvería demasiado complicado. Por ejemplo, si un nodo master tiene dos slaves asociados de 1GB de RAM y 2 CPUs, y 4 GB de RAM y 1 CPU respectivamente, no puede comunicarle al líder que sus recursos son 4 GB de RAM y 2 CPUs porque esos recursos están en slaves distintos. De esa forma, si el líder quisiera schedulear un job, tendría que ir preguntando master por master si pueden correr ese job (y el master tendría que iterar en su tabla de recursos de slaves asociados).
- Si se caen un slave que estaba corriendo un job y el master que estaba asociado a ese slave al mismo tiempo, se pierde la información de dicho job para siempre.

En principio, la solución a estos problemas implicaría el uso de algún tipo de hashing consistente con redundancia (similar al anillo de German). Por eso, creemos que la otra alternativa es mejor: al mantener a los slaves como recursos libres y el estado de sus recursos como global, se evita la reorganización de los slaves cada vez que un nodo se quita o agrega en el sistema. El precio a pagar, por supuesto, es la necesidad de la sincronización de esos datos (no nos parece que tener muchos datos repetidos sea problema, ya que justamente se supone que la cantidad de masters debería ser mucho menor a la de los slaves, por lo que no habría tanta repetición). Para mantener este sincronismo en los datos, buscaríamos hacer algo como lo siguiente:

- Cada vez que se levanta un nodo nuevo, se anuncia por el broadcast UDP como ya lo venía haciendo.
- Los masters eligen un líder siempre que haya una mínima cantidad de masters vivos para dar quorum (tal como se está haciendo ahora), y los slaves no hacen nada hasta que ese líder sea elegido.
- Cuando los slaves detectan que hay un líder, envían a éste su estado (los jobs que están corriendo y sus recursos). El líder luego se encarga de hacer llegar este estado a los demás masters.
- Cuando se tiene que schedulear un job, el líder elige un slave acorde (pero no le ordena inmediatamente correr el job). Luego, le envía un mensaje a todos los masters diciendo que ese slave va a correr ese job. Sólo cuando los masters contestan, el líder envía a ese slave el job a correr.

De esta manera, si se cae un slave (e incluso varios masters) igual la información del job sigue a disposición del líder, que puede lanzar el job nuevamente en otro slave. Agregar o quitar un nodo nunca afecta la organización de los slaves y de los jobs que están corriendo (mientras haya quorum de masters, los jobs están en los slaves corriendo y los masters lo saben). Incluso si el líder cae mientras estaba scheduleando el job la información no se pierde, porque los demás masters habían recibido antes un mensaje del líder diciendo que cierto slave iba a correr determinado job. Los datos en los masters serán siempre eventualmente consistentes (porque todos se darán cuenta si un slave muere, y todos recibirán el mensaje del líder cuando se agregue o quite algún job al cluster), y los jobs podrán schedulearse fácilment y sin tener que consultar los recursos que cada master almacena.