Infrastructure - Task #3038

smart queueing plugin for jenkins that takes buildtime/slow builders into account

2019-11-26 14:38 - Christian Lohmaier

Status:	New	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Gerrit		
Target version:	Qlater		
Tags:		**URL:**	

Description

Overall picture:
LibreOffice uses multijobs to verify gerrit changesets, consisting of Windows, Mac and 2 linux builds
Of those the Windows are the ones taking the longest and don't differ between build machines. They take around 1h to complete.
The macs mostly are uniformly fast, with around 20-30min for a build, but there are outliers with slow bots where a build can take 90 to 120 minutes (those bots typically do tinderbox builds where that doesn't matter). For linux, the always-on-hosts are also uniformly fast, with around 20-30 minutes per build. In recent weeks the number of workers couldn't keep up with peak loads though, so Amazon EC2 and Hetzner cloud workers were examined as a way to cope with many builds submitted at once.

The Problem:
Jenkins scheduling is "dumb" - when it runs out of fast workers, it will assign jobs to the slow workers in a FIFO fashion, that results in unnecessary delays for a multijob's completion.
For example: A job has Windows and Mac and gcc builds already done (so it has been in active queue for at least 60min already, the minimum achievable with the Windows jobs) but all clang workers are busy, Jenkins will then happily use a slow worker that can take way over an hour instead of assigning the worker to a job from the queue where windows build wasn't started/didn't complete yet. If it waited for one of the current busy workers to complete their current job, the job would only have to wait 40min for completion in worst case (if builds on busy workers did just start, they take 20min to complete that, and then are free to accept the job that would take another 20min to complete)

The Solution:
A queue-extension needs to be written that is aware of the connection between the builds and the speed of the individual builders (determined by e.g. a label on the node). For sake of simplicitly assume slow builders are more than twice as slow as the regular builders:

- it would be faster to wait for one the fast workers to complete their current job and build the waiting job than to start the build on the slow node.
  thus never use the slow worker for the first <number of fast workers> jobs in the queue
- also for the next jobs waiting in queue: prefer one where the minimum-time-worker has not been started yet/is building for the shortest amount of time
  in the time it takes for the slow worker, the fast workers can clear a queue of 2x<number-of-fast-workers> and the situation where it would have been faster overall to not use the slow worker at all is avoided/the penalty of additional build time due to the slow worker is minimized.

History

#1 - 2019-11-26 15:11 - Florian Effenberger

Who will take care of this task, something you want to look into?