Batch job load balancing

You did not specify target server for job, however, you notice "Uneven distribution of background jobs" symptom. you want to know why.

The purpose of this page is to clarify the understanding of the background system logic and explanation&suggestions in relation to batch job load balancing.

Since batch jobs can have different start conditions:
- time based start
- event based start
- immediate start

The "Uneven distribution of background jobs" symptom must be analysed case by case.

1) start condition 'immediate':

If the start condition is 'immediate start', the batch scheduler is not involved. The jobs are started right away, and for this case we have very good distribution algorithm for choosing the execution server. e.g. If a process starts N batch jobs, they are assigned to the servers in a round trip.

For more info about "immediate start job mechanism", please refer to SCN wiki: Immediate start jobs - SAP architecture and mechanism

2) start condition 'date/time'(time based job):

Usually, The uneven distributed background jobs(time based jobs, e.g. time based periodic jobs) have a common feature -- these jobs have a full minute as start time, i.e. HH:MM:00.

Since time based jobs are picked up by instance time scheduler running, So these jobs are picked up by the first time scheduler which runs at or after the full minute.

You can check "last run" time of time scheduler for each instance by SM61 -> instance -> double click "Time-Driven Scheduler" -> "Last run"

The period of the batch scheduler is determined by the server specific profile parameter rdisp/btctime.

If rdisp/btctime = 60(default) on all servers, this will always be the time scheduler on the same server, and this will lead to bad load balancing.

rdisp/btctime should be set as recommended in note 1508504. Then we get a good load balancing for 'date/time' jobs at least over a longer time frame.

1508504 - Load balancing in background processing

For detail info about "Time based jobs start mechanism", please refer to SCN wiki: Time based jobs - SAP architecture and mechanism

3) start condition 'after event'
Here we actually have to consider 3 sub cases:

**case 1**

Start condition 'after job' (which is technically: 'after event 'SAP_END_OF_JOB').

This event is triggered implicitly by the batch system at job end. If a job A finishes on a server X, the even scheduler starts on this server in order to check, if some jobs are waiting for the end of job A. If yes, these jobs are started also on server X if there are free batch work processes on this server.

For example, there are 'job trees', i.e. the first job has n successors, and each of the successors has again m successors and so on.

In such a chain (or job tree) each successor runs on the same server as the predecessor, if there are free batch work processes on this server.

**case 2**

The event is triggered explicitly by an application. This can only be done with the function BP_EVENT_RAISE. Note 1511784 describes an improvement of this function regarding load balancing.

**case 3**

The event is triggered from OS command line (or script or similar) via the kernel program sapevt. This program is normally only used by the SAP transport system to trigger transport jobs. I think we can neglect it here since it should not causes Batch job load balancing issue. For more info about SAPEVT, please refer to SCN WIKI: SAPEVT

Some other points:

- If a server has more background work processes than another server, it is very likely that it executes more jobs over time.
- The more background work processes reserved for class A jobs on a server, the fewer jobs this server executes.
- If many jobs have "Server 1" as the target server, but only a few jobs have one of the other servers (Server 2 to N) as the target server. It is probable that more jobs without an execution target will run on "Server 1" because the time scheduler is triggered each time when a batch job finishes. (SAP Notes: 923228)

Please keep in mind:

For the SAP batch processing system, it is more important to minimize job delays than to guarantee perfect load balancing.

Even if you have a perfect load balancing for all jobs, that would only mean that the number of jobs is perfectly distributed. We don't know the run time of a job in advance, so the servers will have a different batch load nevertheless.