Scalable Scheduling of Updates

Grzegorz Mrukwa

20.06.2016

# Implemented parts

During the implementation process, following features were included:

* Simple abstraction layer of a warehouse
* Simple testing interface
* Tables staleness checking by external observers
* Global scheduling algorithms:
  + EDF Partitioned
  + Proportional Partitioning
* Local scheduling algorithms:
  + Prioritized EDF
  + Max Benefit
  + Max Benefit with Lookahead
* Transient overload resolved by update chopping

# Parts not implemented

* Comprehensive load tests
* System maintaining derived tables updates (which is in fact absolutely disjoint system):
  + Ordering of derived table updates, along with its priorities assignment
  + Consistency preservation mechanism (with updates limited to safe trailing edge only)
* Punctuations (which are in fact part of consistency preservation mechanism)

# Known issues

* Max Benefit with Lookahead has quite naive implementation as no decrease-key operation is present for Java priority queue. Therefore the number of items considered by the scheduler each time grows exponentially and the removal of redundant data is necessary.

# Important packages

## warehouse.model

Provides simple models of Job, Table, Track and IO.

## scheduler.global

Contains global scheduling algorithms: EDF Partitioned and Proportional Partitioning. They share common interface, only the group assignment method for every job was changed and method of job redirection to a track.

## scheduler.local

Here are all three local scheduling algorithms: Prioritized EDF, Max Benefit and Max Benefit with Lookahead. They exhibit common interface that allows to exchange them easily in a solution. Differences are defined in particular implementations (comparators or polling method).

# Summary

* Max Benefit with Lookahead algorithm is significantly slower than the rest of local scheduling algorithms in contrary to what was written in the publication. It may be caused by the naive implementation - smarter one may possible resolve this issue.
* The idea of scheduling presented in the publication was not so obvoius at the few first times, but is necessary to take into the account heterogeneity of enqueued jobs. Therefore crucial observation is that scheduling of a job is performed in such a way:
  + Assign all jobs their groups
  + For every jobs as it is released:
    - Find group of that job (previously assigned) and place it into the corresponding local scheduler
  + Independently, for every job at the output of local scheduler, for every local scheduler:
    - Find resources necessary to run the job
* There are two independent actions that may be taken in parallel: assignments of resources and group finding. That is in contrary to initially assumed model:
  + global scheduler
  + local scheduler
  + execution