**CQSIM**

**Low-Level Design Document**

Ren Dongxu

**1. INTRODUCTION**

**1.1 Goals**

* **An event driven job schedule**
* Simulator scans the event sequence and do the operation related to every event in time order.
* Event can be job submit/job finish, monitor event or other event added by the user.
* An overall method invokes and initializes all the modules and the handles of the modules will be transported into the simulator.
* The simulator should be able to support other modules and their subclasses.
* **A user command line interface**
* User can pass all the parameters by command line
* Advantage user interface can be used to call the command line entry automatically.
* A system parameter config file can be used to initialize the command line parameter
* A file name config file can be used to initialize all the temp, debug and output path, name and extension name.
* The data read from the config file are on low level, so the parameter given in the command line will replace the same data read in config file.
* **Extendable module design**
* These modules should have the standard interface.
* All modules are supposed to know all the data formats. Hence, they can get correct data from the dictionary type of parameter. And any modification in data format should be specified clearly in the design document.
* The modules can be extended in 2 ways: subclass and new method.
* Also, new function can be added to the existed method. But this kind of modification should be static, which is used in all extension.
* **Running time interface**
* Keep receiving running time information and show them in the user friendly way.
* **Result analysis and show**
* Read job trace result file and do the statistics as request.
* Show the analysis result in graph.
* New graph method can be added to it easily.
* **Input and output files**
* Input raw files: Job trace and Node structure files
* Formatted files: Job trace, Node structure, Job and Node config files
* Output Result files: Job simulator result, Event log and Debug log.

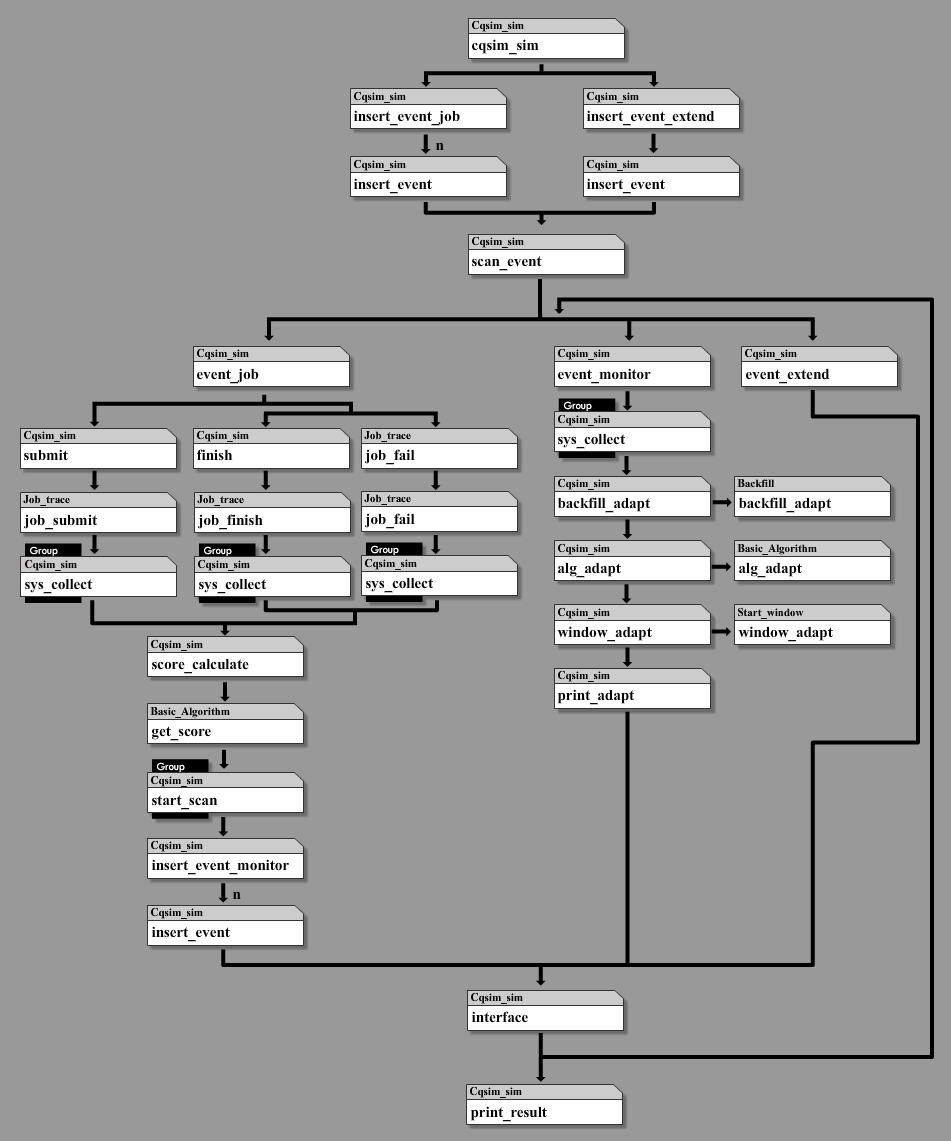
**2. STRUCTURE**

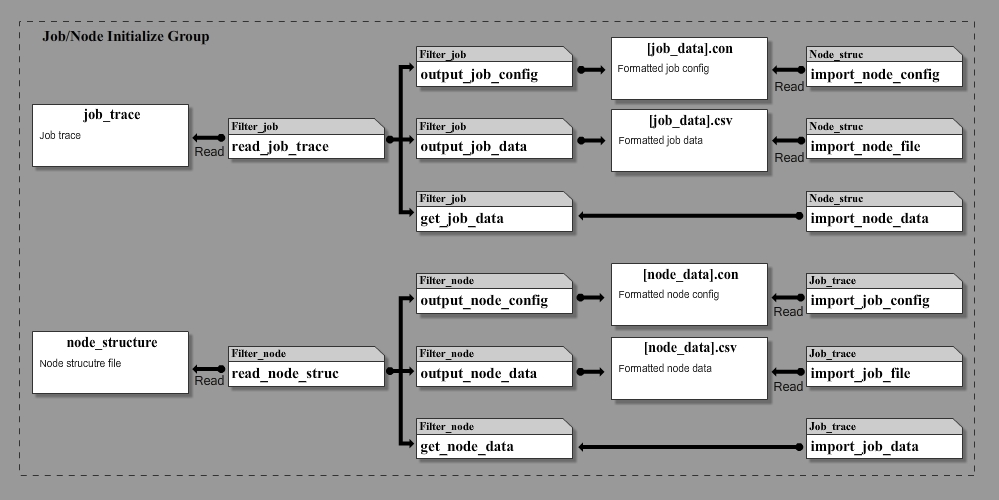
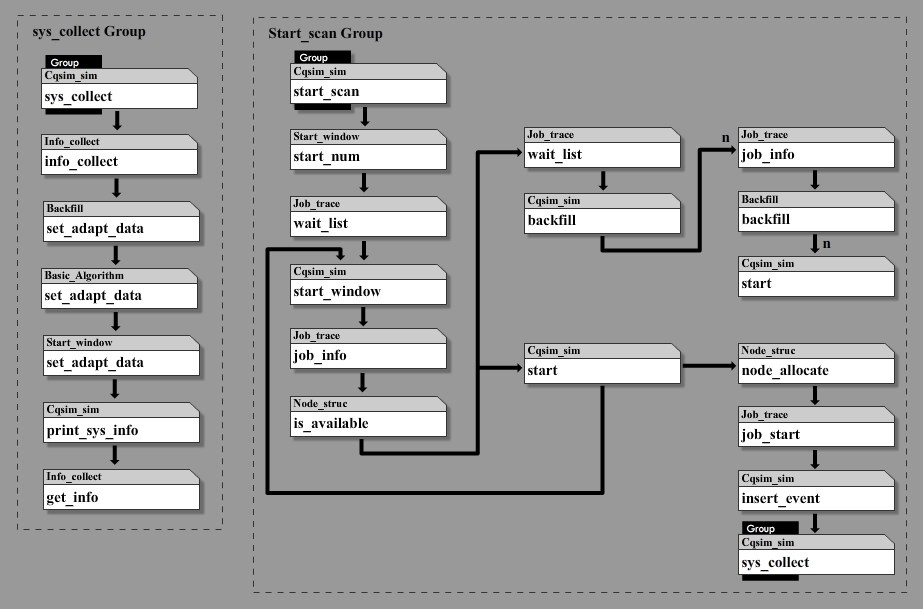
**2.1 Function Map**

The program contains 5 parts:

|  |  |  |
| --- | --- | --- |
| **User Interface & Overall Method** | | |
|  | cqsim | * Basic user command line interface. * All parameters should be transferred by command line. * Additional profile is allowed, but corresponding explain program should be designed. |
|  | filter | * Job and node filter command line interface. * Call the filter process to read raw files and output the data into the formatted file. * Also provide a port to output the formatted data list. |
|  | cqsim\_ad | * Advanced user interface, to simplify the user input. * Parameters are stored in a profile. * Can be designed as a command line interface that user need to only provide the profile file name, or a graphic user interface. * Call the basic command line interface **cqsim** with the data. |
|  | cqsim\_main | * Define all modules and transfer these modules to the simulator . * Different modules can be chosen here. * Call the simulator **Cqsim\_sim**, transfer the modules(in a dictionary data) and parameters into the simulator. * Start the simulation process. * Import the path file Cqsim\_path.py. |
|  | cqsim\_path | * Contain all path value * Be invoked if the file need to access other file in some other place |
|  | factory\_import | * Import all versions of modules * Build a module group dictionary data. This data will be invoked by the factory object to select module group. |
|  | factory | * Factory class which “produce” modules * Read the module group data, receive the module name and select the corresponding modules * Pass the income parameter to selected module and return the module to caller |
|  | Result\_analysis | * Call the result analysis program to deal with the result. |
|  |  |  |
| **Modules** | |  |
|  | * All the modules should contain: \_\_init\_\_(), reset() method to initialize and reset the basic setting. * At least one interface for other module to call it with the input running time parameters. | |
|  | **Filter\_job** | * Receive job trace file name and other parameters. * Read the file and extract the necessary information. * Format the data according to the parameters and store them into a list. * Store the data into a temp file according to the parameters. * Store the overall job trace information into a config file. * Provide output port to transfer the formatted data. |
|  | **Filter\_node** | * Receive node structure file name and other parameters. * Read the file and extract the necessary information. * Format the data according to the parameters and design and store them into a list. * Store the data into a temp file according to the parameters. * Store the overall node tructure information into a config file. * Provide output port to transfer the formatted data. |
|  | **Job\_trace** | * Receive formatted job trace file name or the formatted job trace data. * Read the temp file and store the data into a list. * Provide all the job trace operations, and keep tracing the information of every job. |
|  | **Node\_struc** | * Receive formatted node structure file name or the formatted node structure data. * Read the temp file and store the data into a list. * Provide all the node structure operations, and keep tracing the information of every node. * Provide the prediction of the state of the node structure. * Provide the function to check the prediction data. |
|  | **Backfill** | * Receive parameters when it is initialized. * Provide backfill operation: receive the current state of the waiting list, make some prediction by calling the **node structure object**, return the index list of the jobs which can be backfilled now. * Different backfill mode can be added by designing a new backfill method and build the relationship between mode number and backfill function in **main()** method * Adapt function can be called by the simulator, to modify the parameters in running time depend on the changing of system state. * Adapt config file name is transmitted into the module in adapt parameter list. * All the adapt parameters and the requested average utilization interval list are get from the config file. * Provide a method to analysis and set the adapt value in the **info\_collect** module * Check the most new system information in **info\_collect** module to see whether it reach the adapt request. Call the adapt method if so. * Extend: User can also design a subclass of it if the current backfill structure can not reach the request.   If you do so, import the right subclass in **cqsim\_main()** method, and modify the input running-time parameters in **backfill()** method in **Cqsim\_sim** class.  Also, you may want to modify the initial parameters in **cqsim\_main()** method and re-design the command line in both **Cqsim()** method and **Cqsim\_ad()** method. So does the corresponding config files. |
|  | **Start\_window** | * Receive parameters when it is initialized. * Provide window operation when look for the next job to start:   Receive ***x*** job indexes with related system information which need to be scanned in waiting list,  Change the order of the waiting jobs according to the window function. Then return the new order.  The simulator will call the window operation again when ***y*** job has started after the last window operation in one event iteration.  Provide port to output***x*** and ***y***   * This module will reorder the waiting list before any job starts in this iteration. * Different window mode: Similar to **Backfill** module * Adapt function: Similar to **Backfill** module * Extend: Similar to **Backfill** module |
|  | **Bacis\_Algorithm** | * Receive parameters when it is initialized. * Receive algorithm list and assemble the elements into an algorithm string. * Receive the information of a job and return the job score. Also can receive a list of job information and then return the corresponding list of scores in the same order. * Adapt function: Similar to **Backfill** module * Extend: Similar to **Backfill** module |
|  | **Info\_collect** | * Collect all the system information for record and analysis. * Provide collect and read operations. Hence other methods can check and store the information. |
|  | **Log\_print** | * Provide all the output file operation for the simulator. * Result, running time information and debug log can be done by invoking this module. * Provide the basic operation on files: open, write and close. * Changing style of log can be done by design a different subclass of it. * Every **Log\_print** object can only manage a file in one time. |
|  | **Debug\_log** | * Receive the debug level:   0: No debug  1-3: Three debug level, 3 is the highest.  4: Print the debug information on the screen.  5,6: Print the method and module name.   * User should provide the debug log content with the level number. * The debug module will print the given content depending on the input level number. |
|  | **Output\_log** | * Provide 3 output log print method:   System information log  Job result log  Adapt information log  .   * System information log and Adapt information log method are invoked in every iteration. * Job result log is printed when all jobs are done. |
|  |  |  |
| Simulator | |  |
|  | * Receive parameters and module handles. * Contain an inside event sequence, every event information includes virtual time, event type, event priority and event parameter list. * The simulator can add, delete or modify the event sequence in running time. * There are 3 kinds of event: job event(Job submit/finish), monitor event and extend event which is specially designed for new requirement. * Job submit events added to the sequence before all the process.   Monitor events (from time A to B) added to the sequence when a job starts at A and finish at B. If there exist same monitor event at one time point, no new monitor event will be added.  Job finish event added when the job start.  User designed event added depending on the design.   * In running time, simulator move its virtual time from one event to the next, and stop when all events are done and no more new event comes. * Simple flow of the 3 kinds of event: * Job event - job start scan - system information collect * Monitor event - adapt function * Extend event - user designed function * Call the run-time interface to show the running time state after every event * Print system information log at every event.   Output job result file when all jobs are done. | |
|  |  | |
| Run-time Interface | |  |
|  |  | |
|  |  | |
| Result Analysis | |  |
|  |  | |

**2.2 Flow Diagram**

****

****

**3. Module**

**3.1 Overall**

This is a sample.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | *Method name* | | | |
| **Input** | *Parameter Name* | *(type)* | *Initial value* | *Comment*  *The parameter is necessary if it has no initial value* |
| **Output** | *Return value type* | *(type)* |  | *Comment* |
| **Process** | * *Detail of the duty of the method* | | | |

**3.2 Filter\_job**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | trace | (string) | - | Path and name of the job trace file. |
| save | (string) | None | Path and name of the format job trace file which the formatted job trace data will be stored in. |
| config | (string) | None | Path and name of the format job trace config file |
| sdate | (date) | None | The date and time of the first selected job.  If it is None, no modification will be made. |
| start | (float) | -1 | Virtual submit time of the first selected job.\_j |
| density | (float) | 1.0 | The scale of the submit time of the job trace. The virtual submit time will be:  **[**(Original submit time - first job submit time + **start**) \* **density]** |
| anchor | (int) | 0 | The index of the first job will be read in the job trace file. |
| rnum | (int) | 0 | The number of jobs will be read. |
| max\_node | (int) | dictionary | max number of node structure, this is used to check whether the node request is more than max |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | trace | (string) | None | - |
| save | (string) | None | - |
| config | (string) | None | - |
| sdate | (date) | None | - |
| start | (float) | None | - |
| density | (float) | None | - |
| anchor | (int) | None | - |
| rnum | (int) | None | - |
| max\_node | (int) | None | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_config\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset config data | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | read\_job\_trace | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the job trace file with path string **[trace]** * Read the job trace file and store **[rnum]** jobs starting at **[anchor]** position. * Modify the start date of the selected job trace to **[start]** if it is not None. * Modify the submit time of the jobs:   **[**(Original submit time - first job submit time + **start**) \* **density]**   * Formatted all the selected job data and store them into a local list. * Also get some config data from the original file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | input\_check | | | |
| **Input** | jobInfo | (dictionary) | - | Input job data |
| **Output** | (int) | (int) |  | 1 for correct, <0 for error |
| **Process** | * Check the input job data. * Correct some error if the it can be corrected simply. * Return negative number if any error found. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | config\_set | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * This method provide the addition change on config file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_job\_data | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the formatted job trace data |
| **Process** | * Return the formatted job trace data without other additional information | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_job\_num | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | ( int ) |  | Return the length of the formatted job list |
| **Process** | * Return the length of the formatted job list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | output\_job\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the formatted job data file with path **[save]** * Store the list and other information in the designed format. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | output\_job\_config | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the formatted job config file with path **[config]** * Store the overall job config data | | | |

**3.3 Filter\_node**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | struc | (string) | - | Path and name of the node structure file |
| save | (string) | None | Path and name of the temp node structure file which the formatted node structure data will be stored in. |
| config | (string) | None | Path and name of the format node structure config file |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | struc | (string) | None | - |
| save | (string) | None | - |
| config | (string) | None | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_config\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset config data | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | read\_node\_struc | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the node structure file with path string **[struc]** * Formatted the node structure and store them into a local list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | input\_check | | | |
| **Input** | nodeInfo | (dictionary) | - | Input node data |
| **Output** | (int) | (int) |  | 1 for correct, <0 for error |
| **Process** | * Check the input node data. * Return negative number if any error found. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_node\_num | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | ( int ) |  | Return the length of the formatted node list |
| **Process** | * Return the length of the formatted node list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_node\_data | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the formatted node structure data. |
| **Process** | * Return the formatted node structure data without other additional information | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | output\_node\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the formatted node structure file with path **[save]** * Store the list and other information in the designed format. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | output\_node\_config | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the formatted node config file with path **[config]** * Store the overall node config data | | | |

**3.4 Job\_trace**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | start | (float) | -1 | Virtual submit time of the first selected job.\_j |
| num | (int) | 0 | The number of jobs will be read. |
| anchor | (int) | 0 | The index of the first job will be read in the job trace file. |
| density | (float) | 1.0 | The scale of the submit time of the job trace. The virtual submit time will be:  **[**(Original submit time - first job submit time + **start**) \* **density]** |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | start | (float) | None | - |
| num | (int) | None | - |
| anchor | (int) | None | - |
| density | (float) | None | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_job\_file | | | |
| **Input** | job\_file | (string) | - | Path and name of the formatted temp job data file. |
| **Output** | None | - |  | - |
| **Process** | * Open the temp job data file with path string **[job\_file]** * Store the information into the local buffers. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_job\_config | | | |
| **Input** | config\_file | (string) | - | Path and name of the formatted job config file. |
| **Output** | None | - |  | - |
| **Process** | * Open the job config file with path string **[config\_file]** * Store the config information into the local buffers. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_job\_data | | | |
| **Input** | job\_data | (list) | - | Formatted job trace data list. |
| **Output** | None | - |  | - |
| **Process** | * Store the income job data into the local list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | submit\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the job list which have not been submitted. |
| **Process** | * Return the job list which have not been submitted. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | wait\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the current waiting list. |
| **Process** | * Return the current waiting list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | run\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the current running list. |
| **Process** | * Return the current running list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | done\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list) |  | Return the job list which are done. |
| **Process** | * Return the job list which are done. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | wait\_size | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | ( int ) |  | Return the total size of the waiting job |
| **Process** | * Return the total size of the waiting job. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_start\_date | | | |
| **Input** | None | - | - | - |
| **Output** | (date) | ( date ) |  | Return the start date |
| **Process** | * Return the start date . | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_virtual\_start\_time | | | |
| **Input** | None | - | - | - |
| **Output** | (float) | (float ) |  | Return the virtual start time |
| **Process** | * Return the virtual start time | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | refresh\_score | | | |
| **Input** | score | (float) | - | The new score or score list (if **[job\_index]** is None) |
| job\_index | (int) | None | The index of the selected job. |
| **Output** | None | None |  | - |
| **Process** | * Refresh the score of the selected job if index is given * Refresh the scores of all jobs in the old order if no index is given. * Reorder the wait list in the order of score (from high to low-) | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | scoreCmp | | | |
| **Input** | jobIndex\_c1 | (int) | - | - |
| jobIndex\_c2 | (int) | - | - |
| **Output** | <cmp> | <cmp> |  | - |
| **Process** | * Method used to order. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_info | | | |
| **Input** | job\_index | (int) | -1 | The index of the selected job. |
| **Output** | (dictionary) | ( dictionary ) |  | Return the detail of the job indicated by the input index #. |
| **Process** | * Return the detail of the job. * If job\_index is -1, return the whole job trace information | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_submit | | | |
| **Input** | job\_index | (int) | - | The index of the selected job. |
| job\_score | (int) | 0 | The score of the selected job. |
| job\_est\_start | (int) | -1 | The estimated tart time of the selected job. |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Submit the selected job * Move the submit pointer to the next job and add the index of the job to waiting list. * Modify the state of the job form "not-submit" to "waiting". * Fill other information of the job. (e.g. scores of the job) * Return 0 if any error ocurr. Otherwise return 1. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_start | | | |
| **Input** | job\_index | (int) | - | The index of the selected job. |
| time | (float) | - | Start time |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Start the selected job * Delete the index of the job from waiting list and add the index of the job to running list. * Modify the state of the job form " waiting " to "running". * Fill other information of the job. (e.g. start time) * Return 0 if any error ocurr. Otherwise return 1. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_finish | | | |
| **Input** | job\_index | (int) | - | The index of the selected job. |
| time | (float) | None | Finish time |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Finish the selected job * Delete the index of the job from running list and add the index of the job to done list. * Modify the state of the job form "running " to "done". * Fill other information of the job. * Return 0 if any error ocurr. Otherwise return 1. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_fail | | | |
| **Input** | job\_index | (int) | - | The index of the selected job. |
| time | (float) | None | Finish time |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Mark the selected job failed * Delete the index of the job from running list and add the index of the job to fail list. * Modify the state of the job form "running " to "fail". * Fill other information of the job. * Return 0 if any error ocurr. Otherwise return 1. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_set\_score | | | |
| **Input** | job\_index | (int) | - | The index of the selected job. |
| score | (float) | - | The score of the selected job |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Modify the score of the job * Fill other information of the job. * Return 0 if any error ocurr. Otherwise return 1. | | | |

**3.5 Node\_struc**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | read\_list | | | |
| **Input** | source\_str | (string) | None | The string need to be analysis into a list |
| **Output** | (list) | (list) |  | The list get from the string |
| **Process** | * Translate a string into a list of int * The string must be like [a,b,…,z] | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_node\_file | | | |
| **Input** | node\_file | (string) | - | Path and name of the formatted temp node data file. |
| **Output** | None | - |  | - |
| **Process** | * Open the temp node data file with path string **[node\_file]** * Store the information into the local buffers. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_node\_config | | | |
| **Input** | config\_file | (string) | - | Path and name of the formatted node config file. |
| **Output** | None | - |  | - |
| **Process** | * Open the node config file with path string **[config\_file]** * Store the config information into the local buffers. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | import\_node\_data | | | |
| **Input** | node\_data | (list) | - | Formatted node structure data list. |
| **Output** | None | - |  | - |
| **Process** | * Store the income node data into the local list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | is\_available | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Check whether the request processe is available. * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_tot | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return total processe number. |
| **Process** | * Return total processe number. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_idle | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return current idle processe number. |
| **Process** | * Return current idle processe number. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_avail | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return current max available idle processe number. |
| **Process** | * Return current max available idle processe number. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | node\_allocate | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| start | (float) | - | Current virtual time |
| end | (float) | - | Job expect end time. |
| job\_index | (int) | - | The index of the job which requests the processe. |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Find the available processe and mark them with the **[job\_index]**. * Modify other information. * Return 1 if every thing is OK, otherwise return 0. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | node\_release | | | |
| **Input** | job\_index | (int) | - | The index of the job which release the processe. |
| end | (float) | - | Job end time. |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Release all theprocesse which marked as **[job\_index]**. * This method need at least 1 input parameter and the parameter should be identically named. * Mark the released processe with "idle" * Modify other related information * Return 1 if every thing is OK, otherwise return 0. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_avail | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| start | (float) | - | Current virtual time |
| end | (float) | None | Job expect end time. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Check whether the job can run from **[start]** to **[end]** with all the prediction. * If **[end]** is None, then set it to **[start]** * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reserve | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| job\_index | (int) | - | The index of the job which requests the processe. |
| time | (float) | - | Job expect run time. |
| start | (float) | None | Current virtual time |
| index | (int) | -1 | The index of the prediction list start to scan |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Reserve the job can from **[start]** to **[end]** in the prediction data. * If **[start]** is None, just find a space to reserve it * If **[index]** is -1, scan the prediction list from 0, otherwise scan from **[index]** * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_delete | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| job\_index | (int) | - | The index of the job which requests the processe. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Delete **[node\_num]** number of processes fromthe reserved job whose index is **[job\_index]** * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_modify | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| start | (float) | - | Current virtual time |
| end | (float) | - | Job expect end time. |
| job\_index | (int) | - | The index of the job which requests the processe. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Modify the reserve data of the selected job. * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_get\_last | | | |
| **Input** | None | - | - | - |
| **Output** | (dictionary) | (dictionary) |  | The dictionary contain the last value of all kind of information |
| **Process** | * Scan the prediction job list and return the last value of start and end time | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_reset | | | |
| **Input** | time | (int) | - | Current virtual time |
| **Output** | (int) | (int) |  | 1: Success 0: No |
| **Process** | * Reset the prediction list * Clean the prediction list, then scan the node state and build the initial prediction list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | find\_res\_place | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| index | (int) | - | The index of prediction list start to scan |
| time | (int) | - | Current virtual time |
| **Output** | (int) | (int) |  | -1: Can reserve the job starting at **[index]**  >=0: The index not available for the reservation |
| **Process** | * Scan the prediction list from **[index]**, return the index of the position in prediction list where the is not available for the reservation. Otherwise, return -1 | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | find\_ place | | | |
| **Input** | node\_req | (dictionary) | - | Request node/core/process.. |
| **Output** | (list) | (list ) |  | List of the allocated job index |
| **Process** | * Find the request node, return the list of node index | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | recover\_place | | | |
| **Input** | node\_list | (list) | - | The node index lit need to release |
| **Output** | None | - |  | - |
| **Process** | * Release the node whose index are in the input list. | | | |

**3.6 Backfill**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | mode | (int) | 0 | Backfill mode, no difference will be made if only one mode designed. |
| ad\_mode | (int) | 0 | Adapt backfill mode |
| node\_module | (handle) | None | Node structure module handle |
| info\_module | (handle) | None | System information module handle |
| debug | (handle) | None | Debug module handle |
| para\_list | (list) | None | Additional parameter. |
| ad\_para\_list | (list) | None | Adapt parameter. |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Initialize the adapt parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | mode | (int) | None | - |
| ad\_mode | (int) | None | - |
| node\_module | (handle) | None | - |
| info\_module | (handle) | None | - |
| debug | (handle) | None | - |
| para\_list | (list) | None | - |
| ad\_para\_list | (list) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. * Reset the adapt parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill | | | |
| **Input** | wait\_job | (list) | - | The list of the related waiting job with the details. Each job information is a dictionary. |
| para\_in | (dictionary) | None | Running time parameters in the dictionary type. |
| **Output** | (list) | ( list ) |  | List of the backfill jobs. None for no job can be backfill. |
| **Process** | * This is the entry of the backfill module. * Receive the running time information and store them into the local buffers, then invoke **main** method to deal with the request. * Get the first backfill job index(in wait list) from the **main** method and return it to the invoker. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | main | | | |
| **Input** | None | - | - | All the parameters should be stored in the local buffer. |
| **Output** | (list) | ( list ) |  | List of the backfill jobs. None for no job can be backfill. |
| **Process** | * Provide the backfill function. * Return the List of index of the backfill jobs . * It select different backfill mode by the input parameter **[mode]**, and invoke corresponding backfill method. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill\_EASY | | | |
| **Input** | None | - | - | All the parameters should be stored in the local buffer. |
| **Output** | (list) | ( list ) |  | List of the backfill jobs. None for no job can be backfill. |
| **Process** | * EASY backfill * Return the List of index of the backfill jobs . | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill\_cons | | | |
| **Input** | None | - | - | All the parameters should be stored in the local buffer. |
| **Output** | (list) | ( list ) |  | List of the backfill jobs. None for no job can be backfill. |
| **Process** | * Conservative backfill * Return the List of index of the backfill jobs . | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_reset | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Read the adapt config file and reset the adapt parameter * Add average utilization interval time into **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | set\_adapt\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Analysis the information in the **Info\_collect** module and add the new adapt data in the most new item in  **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_info\_name | | | |
| **Input** | None | - | - | - |
| **Output** | (string) | (string) |  | The name of the adapt data name in **Info\_collect** module |
| **Process** | * Return the name of the adapt data name in **Info\_collect** module | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_read\_config | | | |
| **Input** | fileName | (string) | - | Config file name |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Read the adapt config file * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill\_adapt | | | |
| **Input** | para\_in | (list) | - | Current running time parameters |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Call the selected adapt method depending on the adapt mode * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_1 | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Adapt method * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_list | | | |
| **Input** | inputstring | (string) | - | Input string which need to be analysis into a list |
| regex | (string) | r”([^,]+)” | Regular expression string |
| **Output** | (list) | (list ) |  | The result list |
| **Process** | * Analysis the income string and use the income regular expression sample to analysis it. * Return the result list of string. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list ) |  | The list of parameters which may be modified when adapt |
| **Process** | * Return the list of parameters which may be modified when adapt | | | |

**3.7** **Start\_window**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | mode | (int) | 0 | Window mode, no difference will be made if only one mode designed. |
| ad\_mode | (int) | 0 | Adapt window mode |
| node\_module | (handle) | None | Node structure module handle |
| info\_module | (handle) | None | System information module handle |
| debug | (handle) | None | Debug module handle |
| para\_list | (list) | [5,0,0] | Additional parameter list. |
| para\_list\_ad | ( list ) | None | Additional parameter list for adapt function. |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * **[win\_size]** = **[para\_list[0]]** * **[check\_size\_in]** =**[para\_list[1]]**, **[check\_size\_in]** = **[win\_size]** if **[para\_list[1]]** is -1 * **[start\_max\_size]** =**[para\_list[2]]**, **[start\_max\_size]** = **[win\_size]** if **[para\_list[1]]** is -1 | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | mode | (int) | None | - |
| ad\_mode | (int) | None | - |
| node\_module | (handle) | None | - |
| info\_module | (handle) | None | - |
| debug | (handle) | None | - |
| para\_list | (list) | None | - |
|  | para\_list\_ad | ( list ) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_window | | | |
| **Input** | wait\_job | (list) | - | The list of the related waiting job with the details. Each job information is a dictionary. |
| para\_in | (dictionary) | None | Running time parameters in the dictionary type. |
| **Output** | (list) | ( list ) |  | The reordered sequence of the input job list. |
| **Process** | * This is the entry of the adapt module. * Receive the running time information and store them into the local buffers, then invoke **main** method to deal with the request. * Get the reordered job sequence from the **main** method and return it to the invoker. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | main | | | |
| **Input** | None | - | - | All the parameters should be stored in the local buffer. |
| **Output** | (list) | ( list ) |  | The reordered sequence of the input job list. |
| **Process** | * Provide the adapt function. * Return the reordered job sequence . * It select different window mode by the input parameter **[mode]**, and invoke corresponding window method. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | window\_size | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the window size |
| **Process** | * As the window function only change the order of the first x waiting jobs, it is not necessary for the simulator to pass the whole waiting list into the adapt module. * Return the window size. If waiting job list is longer than that, the window module do not care about the rest part. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | check\_size | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the check size |
| **Process** | * Return the check size. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_num | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | The number of the jobs which are started before next window. |
| **Process** | * Return the number of the jobs which are started before next window | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_list | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the buffers and rebuild the sequence list by calling the recursion method **build\_seq\_list()**. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | build\_seq\_list | | | |
| **Input** | seq\_len | (int) | - | Sequence list length |
| ele\_pool | ( list ) | - | Element pool in order |
| temp\_index | ( int ) | - | The position of the number set in this iteration. |
| **Output** | None | - |  | - |
| **Process** | * This is a recursion method. It keep calling itself until no more element in **[ele]**. * In every iteration, the method takes an element out from the element pool. * When no more element in the pool, it stop recursion and record all the elements in order, hence a new sequence is produced and be added to the sequence list. * This design is to make the sequence list be able to fit different check size in running time:   if check size is 3, take first 1\*2\*3=6 sequences in the list  if check size is 4, take first 1\*2\*3\*4=24 sequences in the list, and so on. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | window\_check | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | ( list ) |  | The reordered sequence of the input job list. |
| **Process** | * Do the window check and return the reordered sequence of the input job list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_reset | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Read the adapt config file and reset the adapt parameter * Add average utilization interval time into **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | set\_adapt\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Analysis the information in the **Info\_collect** module and add the new adapt data in the most new item in  **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_info\_name | | | |
| **Input** | None | - | - | - |
| **Output** | (string) | (string) |  | The name of the adapt data name in **Info\_collect** module |
| **Process** | * Return the name of the adapt data name in **Info\_collect** module | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_read\_config | | | |
| **Input** | fileName | (string) | - | Config file name |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Read the adapt config file * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | window\_adapt | | | |
| **Input** | para\_in | (list) | - | Current running time parameters |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Call the selected adapt method depending on the adapt mode * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_1 | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Adapt method * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_list | | | |
| **Input** | inputstring | (string) | - | Input string which need to be analysis into a list |
| regex | (string) | r”([^,]+)” | Regular expression string |
| **Output** | (list) | (list ) |  | The result list |
| **Process** | * Analysis the income string and use the income regular expression sample to analysis it. * Return the result list of string. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list ) |  | The list of parameters which may be modified when adapt |
| **Process** | * Return the list of parameters which may be modified when adapt | | | |

**3.8 Basic\_Algorithm**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | ad\_mode | (int) | 0 | Adapt mode, no difference will be made if only one mode designed. |
| element | (list) | None | Element list of the algorithm. |
| info\_module | (handle) | None | System information module handle |
| debug | (handle) | None | Debug module handle |
| ad\_para\_list | (dictionary) | None | Adapt parameter. |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Assemble the element list into the algorithm string | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | ad\_mode | (int) | None | - |
| element | (list) | None | - |
| info\_module | (handle) | None | - |
| debug | (handle) | None | - |
| ad\_para\_list | (dictionary) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. * Assemble the element list into the algorithm string | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | build\_alg\_str | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Group the algorithm elements and form the algorithm string. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_score | | | |
| **Input** | wait\_job | (list) | - | The list of all waiting job with the details. Each job information is a dictionary. |
| currentTime | ( float ) | - | Current virtual time |
| para\_list | (dictionary) | None | Related system current information. |
| **Output** | (list) | ( list ) |  | The score list of the wait job. Return None if any error ocurr. |
| **Process** | * Receive the job information and system information. * Calculate the job score depending on the input information. * Return the score list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_reset | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Read the adapt config file and reset the adapt parameter * Add average utilization interval time into **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | set\_adapt\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Analysis the information in the **Info\_collect** module and add the new adapt data in the most new item in  **Info\_collect** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_info\_name | | | |
| **Input** | None | - | - | - |
| **Output** | (string) | (string) |  | The name of the adapt data name in **Info\_collect** module |
| **Process** | * Return the name of the adapt data name in **Info\_collect** module | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_read\_config | | | |
| **Input** | fileName | (string) | - | Config file name |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Read the adapt config file * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | alg\_adapt | | | |
| **Input** | para\_in | (list) | - | Current running time parameters |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Call the selected adapt method depending on the adapt mode * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_1 | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int ) |  | 1. success 0. not |
| **Process** | * Adapt method * Return 1 if success. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_list | | | |
| **Input** | inputstring | (string) | - | Input string which need to be analysis into a list |
| regex | (string) | r”([^,]+)” | Regular expression string |
| **Output** | (list) | (list ) |  | The result list |
| **Process** | * Analysis the income string and use the income regular expression sample to analysis it. * Return the result list of string. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_adapt\_list | | | |
| **Input** | None | - | - | - |
| **Output** | (list) | (list ) |  | The list of parameters which may be modified when adapt |
| **Process** | * Return the list of parameters which may be modified when adapt | | | |

**3.9 Info\_collect**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | ave\_uti | (list) | None | Average utilization interval list. |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Reset the output data, set the data obtain methods of job/monitor events. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | ave\_uti | (list) | None | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | <User Defined Method> | | | |
| **Input** | None | - | - | Input and output are always None |
| **Output** | None | - |  | - |
| **Process** | * This stands for the methods user defined to get the data. * After implementing these methods, you need to connect it to the data name you want to set. * You can use the all the known data for calculating. And you are supposed to know them when you build this method. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_info\_data | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the output data. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | add\_info\_data | | | |
| **Input** | data\_name | (string) | - | Dictionary name of the data |
| j\_func | (method) | None | The method running at job event |
| m\_func | (method) | None | The method running at monitor event |
| **Output** | None | - |  | - |
| **Process** | * Add the item to **[info\_data]** which concluding dictionary name, job event method and monitor event method. * Create the corresponding overall buffer. * Reset the data number. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_state\_date | | | |
| **Input** | date | (date) | - |  |
| **Output** | None | - |  | - |
| **Process** | * Reset the start date. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | info\_collect | | | |
| **Input** | time | (float) | - | Virtual time of this information |
| event | (int) | - | 1:Job, 2:Monitor, 3:Extend, -1:Initial |
| uti | (float) | - | The utilization at this time |
| extend | (list) | None | Other new characters may be added. |
| current\_para | (list) | None | Current parameter |
| **Output** | None | - |  | - |
| **Process** | * Receive formatted system information and store them as a new item in the list. * Call **calculate\_ave\_uti()** to get the required average utilization. * Call **info\_analysis ()** to get the required data. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | info\_analysis | | | |
| **Input** | event | (dictionary) | - | Event information |
| **Output** | None | - |  | - |
| **Process** | * Call every data calculate method to get data | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_info | | | |
| **Input** | index | (int) | - | The index of the request information.  If it is None, return the whole list. |
| **Output** | (dictionary) | ( dictionary ) |  | Return the request system information |
| **Process** | * Return the request system information list. * Return None if index is exceeded | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_len | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the length of the system information list. |
| **Process** | * Return the length of the system information list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_current\_index | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the current data index. |
| **Process** | * Return the current data index. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | calculate\_ave\_uti | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Calculate the average utilization for the most new system information. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_avg\_interval | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the average interval list | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reorder\_uti\_interval | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reorder the utilization interval list by reset the order list. * The order list is for the module to calculate average utilization quicker. | | | |

**3.10 Log\_print**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | filePath | (string) | - | file name |
| mode | (int) | 0 | 0: renew file 1: add log |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Call **file\_open** method to open the specified | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | filePath | (string) | None | - |
| mode | (int) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | file\_open | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: success 0: Fail |
| **Process** | * Open the specified file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | file\_close | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: success 0: Fail |
| **Process** | * Close the opened file if any file is opened. * Return 1 if success, otherwise return 0 | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | log\_print | | | |
| **Input** | context | (string) | - | Context to print. |
| isEnter | (int) | 1 | 1: print Enter after context otherwise: not print enter |
| **Output** | (int) | (int) |  | 1: success 0: Fail |
| **Process** | * Print the log to the file specified before. * Return 1 if success, otherwise return 0 | | | |

**3.11 Debug\_log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | lvl | (int) | 2 | 0 to 5, 0 is no debug log printed |
| show | (int) | 2 | The lowest level which will be print on the screen. |
| path | (string) | None | Debug log path and name. |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | lvl | (int) | None | - |
| show | (int) | None |  |
| path | (string) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_log | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Clean the specified debug log. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | set\_lvl | | | |
| **Input** | lvl | (int) | 0 | 0 to 4, 0 is no debug log printed, 3 is printing all debug log. |
| **Output** | None | - |  | - |
| **Process** | * Reset the debug level. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | debug | | | |
| **Input** | context | (all type) | - | Debug content, will be changed into string. |
| lvl | (int) | 3 | 1 to 4, 0 is no debug log printed, 3 is printing all debug log. |
| isEnter | (int) | 1 | 1 for enter after the context, 0 for not enter. |
| **Output** | None | - |  | - |
| **Process** | * Call the log print module to add the content to the debug log if **[lvl]** is smaller than the print log level. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | line | | | |
| **Input** | lvl | (int) | 1 | 1 to 4, 0 is no debug log printed, 3 is printing all debug log. |
| signal | (string) | “-” | Signal to form the line |
| num | (int) | 15 | Duplication number of the signal |
| **Output** | None | - |  | - |
| **Process** | * Call the log print module to add the content to the debug log if **[lvl]** is smaller than the print log level. | | | |

**3.12 Output\_log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | output | (dictionary) | None | Output file name dictionary |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Initialize all the output file name | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | output | (dictionary) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_output | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the output file path and name. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_sys\_info | | | |
| **Input** | sys\_info | (dictionary) | - | System information needed to be printed. |
| **Output** | None | - |  | - |
| **Process** | * Print the current system information to the system information file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_adapt | | | |
| **Input** | adapt\_info | ( dictionary ) | - | Adapt information needed to be printed. |
| **Output** | None | - |  | - |
| **Process** | * Print the adapt information to the adapt information file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_result | | | |
| **Input** | job\_module | (handle) | - | Job trace module |
| **Output** | None | - |  | - |
| **Process** | * Print all the job result. | | | |

**3.13 Cqsim\_sim**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | module | (dictionary) | - | The dictionary of the input module handle. |
| monitor | (float) | None | Monitor event time interval. |
| mon\_para | (list) | None | Monitor parameter list |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | module | (dictionary) | None | - |
| monitor | (float) | None | - |
| mon\_para | (list) | None | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | show\_module\_info | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Show module information in debug file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | cqsim\_sim | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * The main process of the simulator. * Initialize the event sequence with the job submit event, monitor event and extend event. * Scan the event sequence and deal with all the event in the sequence. * Output the job result. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | insert\_event\_job | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Read the job trace and insert the job submit event in the event sequence in time order. * event information:   type : 1  time: submit time  priority: 2  para: [1,job index], means this is a submit event. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | insert\_event\_monitor | | | |
| **Input** | start | (float) | - | Start time of the start job |
| end | (float) | - | End time of the start job |
| **Output** | None | - |  | - |
| **Process** | * Insert the monitor event in the event sequence from **[start]** to **[end]**.(Contain **[start]**, not **[end]**) * event information:   type : 2  time: monitor time  priority: 5  para: **[mon\_para]** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | insert\_event\_extend | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Insert the extend event in the event sequence in time order. * event information:   type : 3  time: user designed  priority: user designed  para: user designed | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | insert\_event | | | |
| **Input** | type | (int) | - | 1: job 2: monitor 3: extend |
| time | (float) | - | Virtual time of the event |
| priority | (int) | - | Priority of the job |
| para | (list) | None | Parameter list of the event. |
| quick | (index) | -1 | Quick insert signal,1 for just add it to the end of the queue |
| **Output** | None | - |  | - |
| **Process** | * Insert the event in the sequence, automatically find the place by parameters **[time]** and **[priority]**. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | delete\_event | | | |
| **Input** | type | (int) | - | 1: job 2: monitor 3: extend |
| time | (float) | - | Virtual time of the event |
| index | (int) | - | The index of the deleting event |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Delete the selected event which is indicated by **[index]** or **[time** & **type]** * If invoker provides **[index]** and **[time** & **type]**, the **[index]** parameter has higher priority. * Return 1 if success, otherwise 0. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_index\_monitor | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the current monitor pointer. |
| **Process** | * Return the current monitor pointer. * This helps inserting the monitor event | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | scan\_event | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Scan the event sequence recursively. * Call the corresponding method to deal with the current event in the sequence, then move the pointer to the event. * Stop when no event left in the sequence. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | event\_job | | | |
| **Input** | para\_in | (list) | None | Parameter list of the event. |
| **Output** | None | - |  | - |
| **Process** | * Deal with the job event (submit/finish). * Calculate the scores of the waiting job after the event is done. * Call the start scan method group: window - start new job - backfill * Store the system information. * Insert monitor event from current time to time of the next event. * Call the user interface module to show the current system state. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | event\_monitor | | | |
| **Input** | para\_in | (list) | None | Parameter list of the event. |
| **Output** | None | - |  | - |
| **Process** | * Deal with the monitor event. * Call the adapt functions. * Call the **print\_adapt()** method if needed. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | event\_extend | | | |
| **Input** | para\_in | (list) | None | Parameter list of the event. |
| **Output** | None | - |  | - |
| **Process** | * Deal with the extend event. * Call the extend process. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | submit | | | |
| **Input** | job\_index | (int) | - | Index of the submitting job. |
| **Output** | None | - |  | - |
| **Process** | * Submit the job by calling the corresponding method in **job\_trace** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | finish | | | |
| **Input** | job\_index | (int) | - | Index of the finish job. |
| **Output** | None | - |  | - |
| **Process** | * Finish the job by calling the corresponding method in **job\_trace** module. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start | | | |
| **Input** | job\_index | (int) | - | Index of the finish job. |
| **Output** | None | - |  | - |
| **Process** | * Start the job by calling the corresponding method in **job\_trace** module. * Insert job finish event | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | score\_calculate | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Calculate the score for all jobs in waiting list. * Reorder the waiting list depending on the score list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_scan | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Scan the jobs in waiting list till no job can be start or backfill. * Window function will be used before job start. * Backfill function will be used when no job can be started. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_window | | | |
| **Input** | temp\_wait\_B | (list) | - | Job wait list |
| **Output** | (list) | ( list ) |  | New list after window check |
| **Process** | * Call the window function to modify the order of the waiting job. * Return the new reorder list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill | | | |
| **Input** | temp\_wait | (list) | - | Job wait list |
| **Output** | (int) | (int) |  | 1: Success 0: no |
| **Process** | * Call the backfill function and get the backfill job list. * Start them if any job in the list. * Return 1 for some jobs are backfill, 0 for no. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | sys\_collect | | | |
| **Input** | sys\_info\_list | (dictionary) | - | Current system information list |
| **Output** | None | - |  | - |
| **Process** | * Collect the current system information and call the **Info\_collect** module to store them. * Print the current system information. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | interface | | | |
| **Input** | sys\_info | (dictionary) | None | Current system information need to be shown |
| **Output** | None | - |  | - |
| **Process** | * Call the running time user interface module to show the inforamtion. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill\_adapt | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: modify 0: not modify |
| **Process** | * Call the **adapt** method in **backfill** module to modify the parameter of backfill in the monitor event process. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | alg\_adapt | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: modify 0: not modify |
| **Process** | * Call the **adapt** method in **Basic\_algorithm** module to modify the algorithms in the monitor event process. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | window\_adapt | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: modify 0: not modify |
| **Process** | * Call the **adapt** method in **Start\_window** module to modify the parameter of window in the monitor event process. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_sys\_info | | | |
| **Input** | sys\_info | (dictionary) | - | System information needed to be printed. |
| **Output** | None | - |  | - |
| **Process** | * Print the current system information to the system information file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_adapt | | | |
| **Input** | adapt\_info | ( dictionary ) | - | Adapt information needed to be printed. |
| **Output** | None | - |  | - |
| **Process** | * Print the adapt information to the adapt information file. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_result | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Print all the job result. | | | |

**4. Data**

**4.1 Overall**

All modules are suppose to know the format of all public data although they do not really know them. Hence, they can get the right data from the incoming dictionary.

So, any change on data format should be record clearly. This section list all public data in every module, the corresponding format are discussed in the next section.

**4.2 Public Data**

White data are list or dictionary. Gray data are variable.

Some data have not commentary, because they are not important or have been explained.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Filter\_job** | jobList | | job data list | | | | |
| config\_data | | list of data information which will be stored into the config file. | | | | |
| start | | start virtual time | | | | |
| sdate | | start date | | | | |
| density | | job submit density modification rate | | | | |
| anchor | | position of the first read job in the original job trace file | | | | |
| rnum | | read job number | | | | |
| trace | | original job trace file name | | | | |
| save | | formatted job trace file name | | | | |
| config | | config file name | | | | |
| jobNum | | read job number | | | | |
| debug | |  | | | | |
|  | | | | | | | | |
|  | **Filter\_node** | nodeList | | | node data list | | | |
| config\_data | | | list of data information which will be stored into the config file. | | | |
| struc | | | original node structure file name | | | |
| save | | | formatted node structure file name | | | |
| nodeNum | | | total node number | | | |
| config | | | config file name | | | |
| debug | | |  | | | |
|  | | | | | | | | |
|  | **Job\_trace** | jobTrace | | | formatted job data list | | | |
| job\_submit\_list | | |  | | | |
| job\_wait\_list | | |  | | | |
| job\_run\_list | | |  | | | |
| job\_done\_list | | |  | | | |
| job\_wait\_size | | |  | | | |
| start | | | virtual start time | | | |
| start\_date | | | start date | | | |
| anchor | | |  | | | |
| read\_num | | |  | | | |
| density | | |  | | | |
| start\_offset\_A | | | This is the offset time made by user input virtual start time in job filter. It is get from the config file. | | | |
| start\_offset\_B | | | This is the offset time made by user input virtual start time in job trace. | | | |
| debug | | |  | | | |
|  | | | | | | | | |
|  | **Node\_struc** | nodeStruc | | | formatted node data list | | | |
| nodePool | | | idle node index pool. | | | |
| temp\_nodePool | | |  | | | |
| job\_list | | | running job index list | | | |
| predict\_node | | | predict node index | | | |
| predict\_job | | | predict job index | | | |
| tot | | | total node number | | | |
| idle | | | idle node number | | | |
| avail | | | max available node number | | | |
| debug | | |  | | | |
|  | | | | | | | | |
|  | **Backfill** | para\_list\_in | | |  | | | |
| ad\_para\_list\_in | | |  | | | |
| current\_para | | |  | | | |
| ad\_ current\_para | | |  | | | |
| wait\_job | | |  | | | |
| para | | | mode | |  | |
| ad\_mode | |  | |
| size | |  | |
| ad\_config | |  | |
| node\_module | | |  | | | |
| debug | | |  | | | |
| adapt\_data\_name | | |  | | | |
| adapt\_data\_para | | |  | | | |
| check\_data\_name | | |  | | | |
| check\_data\_para | | |  | | | |
| ave\_uti\_interval | | |  | | | |
| ave\_uti\_index | | |  | | | |
| adapt\_item | | |  | | | |
| bound\_item | | |  | | | |
| adapt\_info\_name | | |  | | | |
|  | | | | | | | | |
|  | **Start\_window** | para\_list |  | | | | | |
| ad\_para\_list |  | | | | | |
| current\_para |  | | | | | |
| seq\_list |  | | | | | |
| temp\_list |  | | | | | |
| wait\_job |  | | | | | |
| para | mode | | |  | | |
| ad\_mode | | |  | | |
| win\_size | | | window size | | |
| check\_size | | | the first x job will be reorder to find the quickest sequence | | |
| max\_start\_size | | | The max number of job can be start between 2 window function | | |
| ad\_config | | |  | | |
| node\_module |  | | | | | |
| info\_module |  | | | | | |
| temp\_check\_len |  | | | | | |
| debug |  | | | | | |
| adapt\_data\_name |  | | | | | |
| adapt\_data\_para |  | | | | | |
| check\_data\_name |  | | | | | |
| check\_data\_para |  | | | | | |
| ave\_uti\_interval |  | | | | | |
| ave\_uti\_index |  | | | | | |
| adapt\_item |  | | | | | |
| bound\_item |  | | | | | |
| adapt\_info\_name |  | | | | | |
|  | | | | | | | | |
|  | **Bacis\_Algorithm** | ad\_para\_list |  | | | | | |
| scoreList |  | | | | | |
| para | mode | | |  | | |
| ad\_mode | | |  | | |
| element | | | algorithm element | | |
| sign | | | algorithm sign | | |
| ad\_config | | |  | | |
| algStr | algorithm string | | | | | |
| debug |  | | | | | |
| adapt\_data\_name |  | | | | | |
| adapt\_data\_para |  | | | | | |
| check\_data\_name |  | | | | | |
| check\_data\_para |  | | | | | |
| ave\_uti\_interval |  | | | | | |
| ave\_uti\_index |  | | | | | |
| adapt\_item |  | | | | | |
| bound\_item |  | | | | | |
| adapt\_info\_name |  | | | | | |
|  | | | | | | | | |
|  | **Info\_collect** | sys\_info | | | system information list | | | |
| avg\_inter\_in | | | Average interval list | | | |
| order\_seq | | | The index of average utilization interval in order. | | | |
| eventType | | | monitor | | | ‘C’ |
| submit | | | ‘Q’ |
| start | | | ‘S’ |
| end | | | ‘E’ |
| info\_data | | | Data information list  Contain all data item which need to be store. | | | |
| overall\_info | | | This provide the buffer of each item in **[info\_data]** | | | |
| current\_index | | | current information index | | | |
| start\_date | | | start date | | | |
| alg\_module | | |  | | | |
| total\_uti | | |  | | | |
| data\_num | | | item number of **[info\_data]** | | | |
| debug | | |  | | | |
|  | | | | | | | | |
|  | **Log\_print** | modelist | | | mode list (write or add) | | | |
| filePath | | |  | | | |
| mode | | | mode | | | |
| logFile | | | log file object | | | |
|  | | | | | | | | |
|  | **Debug\_log** | debugFile | | |  | | | |
| path | | |  | | | |
| lvl | | |  | | | |
| show | | |  | | | |
|  | | | | | | | | |
|  | **Output\_log** | event\_seq | | |  | | | |
| sys\_info | | |  | | | |
| adapt\_info | | |  | | | |
| job\_result | | |  | | | |
|  | | | | | | | | |
|  | **Cqsim\_sim** | module | | | contain all input module handles | | | |
| mon\_para | | | monitor parameter. | | | |
| event\_seq | | | event sequence list | | | |
| event\_pointer | | | current event index | | | |
| monitor\_start | | | next monitor event index | | | |
| current\_event | | | current event | | | |
| job\_num | | | total job number | | | |
| currentTime | | | current virtual time | | | |
| startTime | | | virtual start time | | | |
| monitor | | | monitor interval time | | | |
| debug | | |  | | | |
|  |  |  | | |  | | | |

**5 Format**

**5.1 User Command Line Format**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **5.1 Cqsim Command Line** | | | | | | |
| ID | Name1 | Name2 | Type | Default | Dest | Comment |
| 1 | -j | --job | string | None | job\_trace | job trace file name |
| 2 | -n | --node | string | None | node\_struc | node structure file name |
| 3 | -J | --job\_save | string | [job trace name] | job\_save | formatted job trace data file name |
| 4 | -N | --node\_save | string | [job trace name]+”\_node” | node\_save | formatted node structure data file name |
| 5 | -f | -- frac | float | 1 | cluster\_fraction | job density adjust |
| 6 | -s | --start | float | 0 | start | first job start virtual time |
| 7 | -S | --start\_date | date | None | start\_date | first job start date |
| 8 | -r | --anchor | int | 0 | anchor | first job position in job trace |
| 9 | -R | --read | int | -1 | read\_num | number of jobs read from the job trace |
| 10 | -p | --pre | string | "CQSIM\_" | pre\_name | previous file name |
| 11 | -o | --output | string | [job trace name] | output | simulate result file name |
| 12 |  | --debug | string | “debug\_”+[job trace name] | debug | debug file name |
| 13 |  | --ext\_fmt\_j | string | ".csv" | ext\_fmt\_j | formatted job data extension type |
| 14 |  | --ext\_fmt\_n | string | ".csv" | ext\_fmt\_n | formatted job data extension type |
| 15 |  | --ext\_fmt\_j\_c | string | ".con" | ext\_fmt\_j\_c | temp job trace config extension type |
| 16 |  | --ext\_fmt\_n\_c | string | ".con" | ext\_fmt\_n\_c | temp job trace config extension type |
| 17 |  | --path\_in | string | “InputFiles/” | path\_in | input file path |
| 18 |  | --path\_out | string | “Reults/” | path\_out | output result file path |
| 19 |  | --path\_tmp | string | “Temp/” | path\_tmp | temp result file path |
| 20 |  | --path\_debug | string | “Debug/” | path\_debug | debug file path |
| 21 |  | --ext\_jr | string | ".rst" | ext\_jr | job result log extension type |
| 22 |  | --ext\_si | string | ".ult" | ext\_si | system information log extension type |
| 23 |  | --ext\_ai | string | ".adp" | ext\_ai | adapt information log extension type |
| 24 |  | --ext\_d | string | ".log" | ext \_debug | debug log extension type |
| 25 | -v | --debug\_lvl | int | 4 | -debug\_mode | debug mode |
| 26 | -a | --alg | list | None | alg | basic algorithm list |
| 27 | -A | --sign | list | None | alg\_sign | sign of the basic algorithm element |
| 28 | -b | --backfill | int | 0 | backfill | backfill mode |
| 29 | -B | --bf\_para | list | None | bf\_para | backfill parameter list |
| 30 | -w | --win | int- | 0 | win | window mode |
| 31 | -W | --win\_para | list | None | win\_para | window parameter list |
| 32 | -l | --ad\_bf | int | 0 | ad\_bf | backfill adapt mode |
| 33 | -L | --ad\_bf\_para | list | None | ad\_bf\_para | backfill adapt parameter list |
| 34 | -d | --ad\_win | int | 0 | ad\_win | window adapt mode |
| 35 | -D | --ad\_win\_para | list | None | ad\_win\_para | window adapt parameter list |
| 36 | -g | --ad\_alg | int | 0 | ad\_alg | algorithm adapt mode |
| 37 | -G | --ad\_alg\_ para | list | None | ad\_alg\_para | algorithm adapt parameter list |
| 38 | -c | --config\_n | string | "config\_n.set" | config\_n | config file - file name and path |
| 39 | -C | --config\_sys | string | "config\_sys.set" | config\_sys | system config file |
| 40 | -m | --monitor | int | None | monitor | monitor interval time |
| 41 | -M | --mon\_para | list | None | mon\_para | monitor parameter list |
| 42 | -u | --uti | list | None | ave\_uti | average utilization interval list |
| 43 | -e | --ver | string | “ORG” | ave\_uti | version name |

**5.2 Basis Algorithm Format**

The basic algorithm use some simple letters to represent the different informations of a job. The algorithm method stores the information in these buffers and then calculate the scores with them.

|  |  |
| --- | --- |
| s | Job submit time |
| t | Job estimated running time |
| n | Job required nodes # |
| w | Job waiting time |
| m | Current idle nodes # |
| l | Longest job estimated time (in waiting list) |
| z | Longest job waiting time (in waiting list) |

.

The structure of the algorithm string is stored as [elements of the algorithm string, the signal of the element] pairs in a list.

|  |  |
| --- | --- |
| element | A string contain the element. |
| signal | 1: The element will be changed in future  0:The element will not be changed in this simulator |

For example the algorithm list is:

|  |  |  |  |
| --- | --- | --- | --- |
| "0.75" | "\* w/z+" | "0.25" | "\*l/t" |
| 1 | 0 | 1 | 0 |

So, the algorithm string is "0.75\* w/z+0.25\*l/t " and the elements will be changed in future are "0.75" and "0.25".

**5.3 Job Trace Format**

The type of the job trace is list of dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| id | int | The id of the job | -1 |
| submit | float | Submit time of the job | -1.0 |
| wait | float | Actual waiting time | -1.0 |
| run | float | Actual running time | 0.0 |
| usedProc | int | Actual processes the job takes | 0 |
| usedAveCPU | float |  | 0.0 |
| usedMen | float | Actual used memory | 0 |
| reqProc | int | The processes required by user | 0 |
| reqTime | float | The running time required by user | 0.0 |
| reqMem | float | Kilobytes per processor | 0.0 |
| status | int | Status of the job | 0 |
| userID | int | User ID | -1 |
| groupID | int | Group ID | -1 |
| num\_exe | int | Executable number | 0 |
| num\_queue | int | Queue number | 0 |
| num\_part | int | Partition number | 0 |
| num\_pre | int | Preceding job number | 0 |
| thinkTime | int | Think time from preceding job | 0 |
| start | float | Job start time | -1.0 |
| end | float | Job end time | -1.0 |
| score | int | Job scores, shows the priority of the job | 0 |
| state | int | 0: Not submit, 1:In waiting list,  2:Running, 3:Done | 0 |
| happy | int | 0: Not happy, 1:Happy, -1:Not care | -1 |
| estStart | float | Estimated start time, the time predicted to run when the job is submitted considering no backfill or any other modification in job order. | -1.0 |
| extend | list | Other new characters may be added. | None |

**5.4 Node Structure Format**

The type of the node structure is list of dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| id | int | The id of the node. | -1 |
| location | list | The location of the node, kind of [x,y,z] or [x,y]. Can also be None if you do not care about the location of the node. | None |
| group | int | Group ID of the node. | 1 |
| state | int | -1: Idle, Other: The index of the job which takes the node | -1 |
| proc | int | Processes number in the node. | 1 |
| start | float | Start time of the occupy of the node. | -1 |
| end | float | Estimated end time of the occupy of the node. | -1 |
| extend | list | Other new characters may be added. | None |

The type of the predict node structure is list of dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| time | float | Time of the event take place | - |
| idle | int | Idle process number | - |
| avail | int | Available process number | - |

The type of the predict job structure is list of dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| job | int | Job index | - |
| start | float | Job estimate start time | - |
| end | float | Job estimate end time | - |

**5.5 Event Sequence Format**

The type of the node strucutre is list of dictionary.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| type | int | 1:Job, 2:Monitor, 3:Extend, -1:Initial | -1 |
| time | float | Virtual time when the event takes place | -1.0 |
| priority | int | Priority of the event, higher priority will take place earlier if there is another event at the same time. | 5 |
| para | list | Parameter list which will be transferred into the corresponding method.  Job event:  submit: [1, job index] (Q)  start: [3, job index] (S)  finish: [2, job index] (E) | None |

**5.6 System Information Format**

The type of the system information is list of dictionary. It will make a record when an event takes place (event here includes job start).

|  |  |  |  |
| --- | --- | --- | --- |
| **Dictionary Name** | **Type** | **Comment** | **Initial** |
| date | date | Date of the job trace start time  **[date]**+**[time]** suppose to be the real time when the event happen if user did not modify any of them. | None |
| time | float | Virtual time of this information | -1.0 |
| inter | float | Interval time between this information and next one. | -1.0 |
| uti | float | The utilization at this time | -1.0 |
| waitNum | int | Waiting job number at this time | -1 |
| waitSize | int | Total size of all waiting job | -1 |
| event | string | ‘Q’: submit ‘S’: start ‘E’: end ‘C’: monitor | None |
| tot\_ave\_uti | float | Overall average utilization | 0.0 |
| ave\_uti | list | Average utilization list in order. | [] |
| extend | list | Other new characters may be added. | None |

**5.7 Config File Format**

Every line contains a data: [data name]=[data value]

No rest space should appeared in the line. If there are some spaces, the regular expression function will not take them as “useless” signal, sp some error may occur because the system can not transform the space into a number or can not find the file because of the addition space.

Or you may want to add some codes to ignore the additional space. But these codes are not there now.

**5.7.1 File Name And Path Config File**

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| pre\_name | string | previous file name |
| ext\_fmt\_j | string | formatted job data extension type |
| ext\_fmt\_n | string | formatted job data extension type |
| ext\_fmt\_j\_c | string | formatted job trace config extension type |
| ext\_fmt\_n\_c | string | formatted node structure config extension type |
| path\_in | string | input file path |
| path\_out | string | output result file path |
| path\_tmp | string | temp result file path |
| path\_debug | string | debug file path |
| ext\_jr | string | job result log extension type |
| ext\_si | string | system information log extension type |
| ext\_ai | string | adapt information log extension type |
| ext\_debug | string | debug log extension type |

**5.7.2 System Parameter Config File**

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| cluster\_fraction | float | job density adjust |
| start | float | first job start virtual time |
| start\_date | date | first job start date |
| anchor | int | first job position in job trace |
| read\_num | int | number of jobs read from the job trace |
| debug\_lvl | int | debug level |
| alg | list | basic algorithm list |
| alg\_sign | list | sign of the basic algorithm element |
| backfill | int | backfill mode |
| bf\_para | list | backfill parameter list |
| win | int | start window mode |
| win\_para | list | start window module parameter:  [window size],[check size],[max start size],[max window size] |
| ad\_win | int | start window adapt mode |
| ad\_bf | int | backfilladapt mode |
| ad\_alg | int | algorithm adapt mode |
| ad\_win\_para | list | adapt start window parameter list  It contains the config file name |
| ad\_bf\_para | list | adapt backfill parameter list  It contains the config file name |
| ad\_alg\_para | list | adapt basic algorithm parameter list  It contains the config file name |
| config\_n | string | config file - file name and path |
| monitor | float | interval time of monitor event |
| mon\_para | list | monitor parameter list |
| ave\_uti | list | The interval list of the average utilization.  This parameter will be transmitted into the info\_collect module. |
| job\_trace | string | job trace file name |
| node\_struc | string | node structure file name |
| avg\_uti | list | average utilization interval list |
| module\_ver | string | average utilization interval list |

**5.7.3 Adapt Config File(Basic Algorithm/Backfill/Start Window)**

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| adapt\_data\_name | list | Adapt data name in order:  names should not combine if they are same.  **example:** [name 1],[name 2],…,[name X] |
| adapt\_data\_para | list | Adapt data parameter in order:  In most time, the parameter is the index of the corresponding data, -1 mean this data is not a list.  This is really depend on the design of the config file reading method.  **example:** [para 1],[ para 2],…,[ para X] |
| check\_data\_name | list | The name of data need to be check when adapt.  **example:** [name 1],[name 2],…,[name Y] |
| check\_data\_para | list | The parameter of data need to be check when adapt.  **example:** [para 1],[ para 2],…,[ para Y] |
| avg\_uti | list | Interval list.  This list contain all the average data need to be check |
| adapt\_item | list | This is the main part of the adapt function.  One **adapt\_item** can be add if you want a new adapt choice.  For example, you can add an **adapt\_item** to indicate that the *ith* data in **adapt\_data\_name** will -1 if the data in **check\_data\_name** is in case A. And add another **adapt\_item** to indicate that the *ith* data in **adapt\_data\_name** will +1 if the data in **check\_data\_name** is in case B. Hence the *ith* data can be modified in running time depending on different cases of the check data.  One thing needed to be mentioned is you can add conflicted case, but the function will only choose the first one who satisfy the request.  All **adapt\_item** should be written in the right format:  All data should be written as in a list separating by”,” without no addition space.   |  |  | | --- | --- | | 0 | Index of the data in **adapt\_data\_name** | | 1 | 0: change the adapt data to the next value  1: add the next value to the adapt data | | 2 | The new value will be set to/ad to the corresponding adapt data. | | 3~(3+2\*Y+1) | Y is the number of check data – 1.  These data indicate the case request.  For jth check data, it can be considered as “in the case” if  **adapt\_item[3+j\*2]≤check\_data[j]< adapt\_item[3+j\*2+1]**  If all the check data is in the case, then this adapt item is the right one. | |
| bound\_item | list | This is similar to the **adapt\_item**. It defines the bound of all the adapt data when you add the new value in adapt item to them.   |  |  | | --- | --- | | 0 | Index of the data in **adapt\_data\_name** | | 1 | Smallest value | | 2 | Biggest value | |

**5.8 Formatted File**

2 kinds of formatted file(job/node data temp file) have the same structure:

* Each item takes a single line. For each line, the data are stored in the order which is described in previous section(white part). Every single data in the extend part should be store as a single data.

|  |  |  |  |
| --- | --- | --- | --- |
| data 1 | data 2 | data 3 | ...another data |

* “;” is used as the separated signal in a line. "\n" are used to separate lines.

Formatted config file:

* Each value takes a single line: [data name]=[data value]
* No additional space

**5.9 Parameters Format**

**5.9.1 wait\_job (Method: backfill Class: Backfill)**

|  |  |  |
| --- | --- | --- |
| wait\_job (Method: backfill Class: Backfill) | | |
| Name | Type | Comment |
| index | int | Job index |
| proc | int | Request processes number |
| node | int | Request nodes number |
| run | float | Request running time |
| score | float | Job score |

**5.10 Output Format**

The output data separated with “;”.

**5.10.1 Job result**

|  |  |
| --- | --- |
| Name | Comment |
| ID | Job ID (not index) |
| Request process | Request processes number |
| Request node | Request nodes number |
| Request time | Request time |
| Run | Run time |
| Wait | Waiting time |
| Submit | Job submit time |
| Start | Job start time |
| End | Job finish time |
| Node list | The nodes which the job take |

**5.10.2 Event Log**

|  |  |
| --- | --- |
| Name | Comment |
| Start date | Job trace start date  Format: %m/%d/%Y &H:%M:%S |
| Event type | Q: submit S: job start E: job end C: monitor |
| Virtual event time | virtual event time |
| Other parameter | [data name 1]=[data value 1] [data name 2]=[data value 2] …  Different data separate by space |

|  |  |
| --- | --- |
| Other parameter in Event Log | |
| Name | Value |
| uti | System ultilization |
| waitNum | Wait job number at that time |
| waitSize | Wait job total size |

**5.10.3 Adapt Log**

|  |  |
| --- | --- |
| Name | Comment |
| Virtual time | virtual time |
| Start window adapt data | Start window adapt data, separated with “;” |
| Basic algorithm adapt data | Basic algorithm adapt data, separated with “;” |
| Backfill adapt data | Backfill adapt data, separated with “;” |

**5. Extension**

**3.1 Overall**

The program is designed to be an extendable one. All module except **Info\_collect\*** can be modified to fit new request with keeping the port same.

\*In order to keep **Start\_window**, **Basic\_algorithm** and **Backfill** module independent and efficient, they know the inside structure of the **Info\_collect** module. So you need to modify the three module when you modify the **Info\_collect** module.

It can be extended in 3 ways:

1. Build new subclass of modules to fit special request. You should import the subclass in the **factory\_import.py**, and also add a new module group for it.

2. Add new method. All old functions remain same in this way. New function is added, but you can choose not to use it.

3. Modify the original code. You should make this kind of modification only when it fit all application. For example, you may want to modify the original code when you need to trace more running time information, the additional information is useful in most case and you can easily choose not to track them if you don’t need.

You should make sure that all related parts know that change and call the new function in the right way. This is easy to implement because the modules are all highly independent.