**Design Document**

**(Draft version 3)**

**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.
* **Extendable module design**
* These modules should have the standard interface.
* The modules can be extended in 2 ways: subclass and new method.
* **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.

**2. STRUCTURE**

**2.1 Function Map**

The program contains 5 parts:

|  |  |  |
| --- | --- | --- |
| **User Interface** | |  |
|  | 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. |
|  | 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 and parameters into the simulator. |
|  | 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 design and store them into a list. * Store the data into a temp file according to the parameters. * 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. * 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, running list and the nodes, return the index of the next job which can be backfilled. * Different back fill algorithms can be chosen by invoke different subclass of the **Backfill** module. Different backfill mode can be set by the input parameters. |
|  | **Adapt** | * Receive parameters when it is initialized. * Provide adapt operation when look for the next job to start: receive the job index which need to be scanned in waiting list and related system information, change the order of the waiting jobs according to the adapt function. Then return the new order. * This module will reorder the waiting list before any job starts in this iteration. Hence, the advantage scheduling method can be implied. * Different start adapt function can be chosen by invoke different subclass of the **Adapt** module. Different modes can be set by the input parameters. |
|  | **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. * Provide a port for the **system information collect** module, receive the new system information and store the selected data into its own buffer. * Analysis the selected data and store the analysis result into the buffer. These data are used by the adapt function. * Provide adapt operation when monitor event happen: check the selected system information, get the new basic algorithm according to the adapt function. Then return the new algorithm in a string type. * This module will change the basic algorithm of the schedule depending of the system state. Hence, the algorithm can be modified in running time. * Different algorithm adapt function can be chosen by invoke different subclass of the **Adapt\_alg** module. Different modes can be set by the input parameters. * Print adapt record when the algorithm is changed. |
|  | **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. * Changing style of log can be done by design a different subclass of it. |
|  | **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.   * 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. |
|  |  |  |
| Simulator | |  |
|  | * Receive parameters and module handles. * Contain an inside event sequence, every event information includes virtual time, event code 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. * 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 - algorithm adapt function * Extend event - user designed function * Call the run-time interface to show the running time state * Print system information log when job submit, start and finish. | |
|  |  | |
| Run-time Interface | |  |
|  |  | |
|  |  | |
| Result Analysis | |  |
|  |  | |

**2.2 Command Line Design**

**2.3 Flow Diagram**

**3. Module**

**3.1 Overall**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 temp job trace file which the formatted job trace data will be stored in.  If it is None, the formatted job data will not be stored in any 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. |
| debug | (handle) | None | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | trace | (string) | - | - |
| save | (string) | None | - |
| sdate | (date) | None | - |
| start | (float) | -1 | - |
| density | (float) | 1.0 | - |
| anchor | (int) | 0 | - |
| rnum | (int) | 0 | - |
| debug | (handle) | None | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | read\_job\_trace | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Open the job trace file with path string **[job\_trace]** * Read the job trace file and store **[read\_num]** jobs starting at **[anchor]** position. * Modify the start date of the selected job trace to **[start\_date]** 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. | | | |

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. | | | |

**3.3 Filter\_node**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | node\_struc | (string) | - | - |
| node\_save | (string) | None | - |
| debug | (handle) | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | output\_node\_data | | | |
| **Input** | node\_save | (string) | None | Path and name of the temp node structure file which the formatted node structure data will be stored in. |
| **Output** | None | - |  | - |
| **Process** | * Open the temp formatted node structure file with path **[node\_save]** * Store the list and other information in the designed format. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 | | | |

**3.4 Job\_trace**

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

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

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

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

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_next\_submit | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | Return the index of the next submit job. |
| **Process** | * Return the index of the next submit job. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | get\_job\_info | | | |
| **Input** | job\_index | (int) | - | 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. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | job\_submit | | | |
| **Input** | job\_index | (int) | - | The index 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. |
| **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. |
| **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. |
| **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. | | | |

**3.5 Node\_struc**

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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\_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\_num | (int) | - | The amount of request resource |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Check whether the request resource is available. * Return 1 for available, 0 for not available. | | | |

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

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

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | node\_release | | | |
| **Input** | node\_num | (int) | None | Resource number will be released.  If it is None, then all resource which marked as **[job\_index]** will be released. |
| job\_index | (int) | None | The index of the job which release the resource.  If it is None, then the first **[node\_num]** node will be released. |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Release the **[node\_num ]** resource which marked as **[job\_index]**. * This method need at least 1 input parameter and the parameter should be identically named. * Mark the released resource with "idle" * Modify other related information * Return 1 if every thing is OK, otherwise return 0. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_avail | | | |
| **Input** | node\_num | (int) | - | The amount of request resource |
| start | (float) | - | Current virtual time |
| end | (float) | **[start]** | 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. * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reserve | | | |
| **Input** | node\_num | (int) | - | The amount of request resource |
| start | (float) | - | Current virtual time |
| end | (float) | - | Job expect end time. |
| job\_index | (int) | - | The index of the job which requests the resource. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Reserve the job can from **[start]** to **[end]** in the prediction data. * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_delete | | | |
| **Input** | node\_num | (int) | - | The amount of request resource |
| job\_index | (int) | - | The index of the job which requests the resource. |
| **Output** | (int) | (int) |  | 1: Yes 0: No |
| **Process** | * Delete **[node\_num]** number of resources fromthe reserved job whose index is **[job\_index]** * Return 1 for available, 0 for not available. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | pre\_modify | | | |
| **Input** | node\_num | (int) | - | The amount of request resource |
| start | (float) | - | Current virtual time |
| end | (float) | - | Job expect end time. |
| job\_index | (int) | - | The index of the job which requests the resource. |
| **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\_clean | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Clean the prediction data. | | | |

**3.6 Backfill**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | mode | (int) | 0 | Backfill mode, no difference will be made if only one mode designed. |
| para\_list | (dictionary) | None | Additional parameter. |
| debug | (handle) | - | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | mode | (int) | 0 | - |
| para\_list | (dictionary) | None | - |
| debug | (handle) | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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** | (int) | (int) |  | Index of the first backfill job. -1 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 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** | (int) | (int) |  | Index of the first backfill job. -1 for no job can be backfill. |
| **Process** | * Provide the backfill function. * Return the first backfill job index. * It can be extended as an entry and select different backfill mode by the input parameter **[mode]**, and invoke corresponding backfill method. | | | |

**3.7 Adapt**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | mode | (int) | 0 | Adapt mode, no difference will be made if only one mode designed. |
| para\_list | (dictionary) | None | Additional parameter. |
| debug | (handle) | - | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | mode | (int) | 0 | - |
| para\_list | (dictionary) | None | - |
| debug | (handle) | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt | | | |
| **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 can be extended as an entry and select different adapt mode by the input parameter **[mode]**, and invoke corresponding adapt method. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | adapt\_len | | | |
| **Input** | None | - | - | All the parameters should be stored in the local buffer. |
| **Output** | (int) | (int) |  | The maximum length of the adapt job list. |
| **Process** | * As the adapt 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 maximum length of the input job list. If waiting job list is longer than that, the adapt module do not care about the rest part. | | | |

**3.8 Basic\_Algorithm**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | mode | (int) | 0 | Adapt mode, no difference will be made if only one mode designed. |
| element | (list) | - | Element list of the algorithm. |
| para\_list | (dictionary) | None | Additional parameter. |
| printer | (handle) | None | The log print module.  If it is None, print function will be turn off. |
| debug | (handle) | - | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. * Assemble the element list into the algorithm string | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | mode | (int) | 0 | - |
| element | (list) | - | - |
| para\_list | (dictionary) | None | - |
| printer | (handle) | None | - |
| debug | (handle) | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. * Assemble the element list into 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. |
| job\_info | ( dictionary ) | - | The detail of the selected job |
| para\_list | (dictionary) | - | Related system current information. |
| **Output** | (int) | (int) |  | The score of the 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. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | collect\_sys\_info | | | |
| **Input** | sys\_info | ( dictionary ) | - | Selected system information need to be stored in the algorithm module. |
| **Output** | None | - |  | - |
| **Process** | * Receive new selected system information and store them in the local list. * This method is called by **info\_collect** module. As the algorithm adapt function may need to analyze the whole system log, we keep a copy of the related system information list in it to avoid passing the whole system information list every time. * Call the **log\_analysis** method to fill the record. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | log\_analysis | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Look in the local system information log and fill the analysis result of the last record. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | alg\_adapt | | | |
| **Input** | para\_in | (dictionary) | None | Running time parameters in the dictionary type. |
| **Output** | (string) | (string) |  | New algorithm string |
| **Process** | * Receive the current additional parameter. * Check the analysis result and assemble the new algorithm string. | | | |

**3.9 Info\_collect**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | alg\_module | (handle) | 0 | Handle of the algorithm module. It is used to invoke the information collect method in algorithm module. |
| debug | (handle) | - | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | alg\_module | (handle) | 0 | - |
| debug | (handle) | - | - |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | info\_collect | | | |
| **Input** | sys\_info | (dictionary) | - | Formatted system information. |
| **Output** | None | - |  | - |
| **Process** | * Receive formatted system information and store them as a new item in the list. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | info\_analysis | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Check the new system information and fill the analysis result. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. | | | |

**3.10 Log\_print**

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | log\_print | | | |
| **Input** | file | (string) | - | - |
| mode | (int) | 0 | 0: renew file and add log 1: only add log |
| context | (string) | - | Context to print. |
| **Output** | None | - |  | - |
| **Process** | * Print the log to the file. | | | |

**3.11 Debug\_log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | printer | (handle) | - | Log\_print module handle |
| lvl | (int) | 0 | 0 to 3, 0 is no debug log printed, 3 is printing all debug log. |
| path | (string) | - | Debug log path and name. |
| debug | (handle) | - | Debug module handle |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset\_log | | | |
| **Input** | path | (string) | - | Debug log path and name. |
| **Output** | None | - |  | - |
| **Process** | * Create the debug log or clear the exist log. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | debug | | | |
| **Input** | cont | (all type) | - | Debug content, will be changed into string. |
| lvl | (int) |  | 0 to 3, 0 is no debug log printed, 3 is printing all debug log. |
| **Output** | None | - |  | - |
| **Process** | * Call the log print module to print the content if **[lvl]** is smaller than the print log level. | | | |

**3.12 Cqsim\_sim**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | \_\_init\_\_ | | | |
| **Input** | module | (dictionary) | - | The dictionary of the input module handle. |
| **Output** | None | - |  | - |
| **Process** | * Initialize the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | reset | | | |
| **Input** | module | (dictionary) | - | The dictionary of the input module handle. |
| **Output** | None | - |  | - |
| **Process** | * Reset the parameters. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 1  para: [1], means this is a submit event. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | insert\_event\_monitor | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Insert the monitor event in the event sequence in time order. * event information:   type : 2  time: monitor time  priority: 5  para: None | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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) | - | Parameter list of the event. |
| **Output** | (int) | (int) |  | 1: Success 0: Fail |
| **Process** | * Insert the event in the sequence, automatically find the place by parameters **[time]** and **[priority]**. * Return 1 if success, otherwise 0. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | print\_job\_result | | | |
| **Input** | path | (string) | - | Path and name of job result file |
| **Output** | None | - |  | - |
| **Process** | * Print the job result to the file whose path is **[path]** after all the events are done. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | event\_job | | | |
| **Input** | para\_in | (list) | - | 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: adpat - start new job - backfill * Store the system information. * Call the user interface module to show the current system state. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | event\_extend | | | |
| **Input** | para\_in | (list) | - | 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. * Collect the new system information. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. * Collect the new system information. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_scan | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Scan the jobs in waiting list till no job can be start or backfill. * Adapt function will be used in this process. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | start\_adapt | | | |
| **Input** | None | - | - | - |
| **Output** | (int) | (int) |  | 1: modified 0: not modified |
| **Process** | * Call the adapt function to modify the order of the waiting job. * Return 1 if any modification is made, otherwise return 0. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. * Collect the new system information. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | backfill | | | |
| **Input** | job\_index | (int) | - | Index of the finish job. |
| **Output** | (int) | (int) |  | 1: yes 0: no |
| **Process** | * Check whether the selected job can be backfill. Reserve it if it can be backfill. * Return 1 for yes, 0 for no. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | sys\_collect | | | |
| **Input** | None | - | - | - |
| **Output** | None | - |  | - |
| **Process** | * Collect the current system information and call the **Info\_collect** module to store them. * Call the **Log\_print** module to print new system log. | | | |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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** | 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** | print\_sys\_info | | | |
| **Input** | path | (string) | - | Path and name of system inforamtion file |
| sys\_info | (list) | - | System information list needed to be printed. |
| **Output** | None | - |  | - |
| **Process** | * Add new system information to the file whose path is **[path]**. | | | |

**4. Data**

**4.1 Overall**

**4.2 Waiting, running and done list**

|  |  |
| --- | --- |
| Order | |
| Waiting list | Priority order (High to Low) |
| Running list | Estimated finish time order (Early finish early first) |
| Done list | Finish order |

**5 Format**

**5.1 User Command Line Format**

**5.1.1 Cqsim Command Line**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **5.1.1 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\_data.tmp" | job\_save | formatted job trace data file name |
| 4 | -N | --node\_save | string | "node\_data.tmp" | 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 |  | --ext\_tmp\_j | string | ".csv" | ext\_tmp\_job | temp formatted job data extension type |
| 13 |  | --ext\_tmp\_n | string | ".csv" | ext\_tmp\_node | temp formatted job data extension type |
| 14 |  | --path\_in | string | “Input Files/” | path\_in | input file path |
| 15 |  | --path\_out | string | “Reults/” | path\_out | output result file path |
| 16 |  | --path\_tmp | string | “Temp/” | path\_tmp | temp result file path |
| 17 |  | --ext\_jr | string | ".rst" | ext\_jr | job result log extension type |
| 18 |  | --ext\_si | string | ".ult" | ext\_si | system information log extension type |
| 19 |  | --ext\_ai | string | ".adp" | ext\_ai | adapt information log extension type |
| 20 | -a | --alg | list | None | alg | basic algorithm list |
| 21 | -A | --sign | list | None | alg\_sign | sign of the basic algorithm element |
| 22 | -b | --backfill | int | 0 | backfill | backfill mode |
| 23 | -B | --bf\_para | list | None | bf\_para | backfill parameter list |
| 24 | -d | --ad\_sta | int | 0 | ad\_ sta | starting adapt mode |
| 25 | -D | --ad\_sta\_para | list | None | ad\_ sta\_para | starting adapt parameter list |
| 26 | -g | --ad\_alg | int | 0 | ad\_alg | algorithm adapt mode |
| 27 | -G | --ad\_alg\_ para | list | None | ad\_alg\_para | algorithm adapt parameter list |
| 28 | -c | --config\_n | string | "config\_n.set" | config\_n | config file - file name and path |
| 29 | -C | --config\_sys | string | "config\_sys.set" | config\_sys | system config file |

**5.1.2 Filter Command Line**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **5.1. 2 Filter 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\_data.tmp" | job\_save | formatted job trace data file name |
| 4 | -N | --node\_save | string | "node\_data.tmp" | 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 |  | --ext\_tmp\_j | string | ".csv" | ext\_tmp\_job | temp formatted job data extension type |
| 11 |  | --ext\_tmp\_n | string | ".csv" | ext\_tmp\_node | temp formatted job data extension type |
| 12 |  | --path\_in | string | “Input Files/” | path\_in | input file path |
| 13 |  | --path\_tmp | string | “Temp/” | path\_tmp | temp result file path |
| 14 | -c | --config\_n | string | "config\_n.set" | config\_n | config file - file name and path |

**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 |
| run | float | Actual running time | 0.0 |
| node | int | Actual nodes the job takes | 0 |
| reqTime | float | The running time required by user | 0.0 |
| reqNode | float | The node required by user | 0.0 |
| user | int | User ID | -1 |
| group | int | Group ID | -1 |
| 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 |
| weight | int | Weight of the node. Sometimes a node has more than 1 capability. | 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 |

**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. | 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** |
| 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 | int | 1:Job, 2:Monitor, 3:Extend, -1:Initial | -1 |
| extend | list | Other new characters may be added. | None |

**5.7 Config File Format**

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

**5.7.1 File Name And Path Config File**

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| pre\_name | string | previous file name |
| ext\_tmp\_j | string | temp formatted job data extension type |
| ext\_tmp\_n | string | temp formatted job data extension type |
| path\_in | string | input file path |
| path\_out | string | output result file path |
| path\_tmp | string | temp result 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 |

**5.8 Temp File Format**

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

* 1 ~ *n*  lines: *n* overall data information. Each information takes a single line starting with ";" symbol.

|  |  |  |
| --- | --- | --- |
| ; | data name | data content |

* *n+1* ~ *n+1+m*  lines: *m* items. Each item takes a single line and the first symbol can not be ";". For each line, the data are stored in the order which is described in previous section. Every single data in the extend part should be store as a single data.

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

* Space, tab are all considered as the separated signal in a line. "\n" are used to separate lines.

**5.9 Result Log Format**

**5.9.1 Job Result Log**

Same as job data temp file.

**5.9.2 System Information Log**

Store all the system information data.

**5.9.3 Algorithm Adapt Log**

Store all the change of the algorithm.

Each iterm takes a single line and the first symbol can not be ";". For each line, the elements are stored in the order which is described in previous section. Only the elements with signal 1 will be stored in the log.

**5.9.4 Debug Log**

--

**5. Extension**

**3.1 Overall**