Documentation

Contact : david.maas@laposte.net

I. User guide

Running the server

SIMULATION_FOLDER environment variable can point to a valid folder to save the result. Otherwise it will be stored in current user working directory.

Launching the server with 16Gb of memory (you can use less memory to start)

```
> java -Xmx16384m -jar app-0.0.1.jar --server.connection-timeout=5000
```

Perform a request

Minimal API call request example :

```
> curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "42"}'  http://localhost:8080/fileboard/api/simulation/step
```

See appendix for more examples.

Case sensitive Json option	Comment	Default value
step	Number of step to run	42
cellResolution	Number of pixel for cell width and heigth	5
drawingLibrary	- **java.awt** until 2^31-1 pixels (by default support cellResolution and Grid OR - **pngj** no pixels limits (cellResolution is 1 and grid system is disabled)	java.awt
matrixResolution	Resolution of the 2 dimensional array. A big value use more memory. Default is 256 Mb x 4	16384
withGrid	Only with java.awt, draw a grid with grey lines.	false

Until 50.000.000 points you could use java.awt drawing system. It provide a clearner output by created 5 pixels size cell with a grid.
Above this limit you should consider reducing the number of the cellResolution because you reach the limit of java.awt image max size : width * height cannot be bigger than Integer.MAX_VALUE.

Above this limit the library pngj can be used to draw the image. Note the grid syste, is not implemented with this drawing, cellResolution is 1 pixel by default. This library does not load the whole image into memory , it draw the image line by line instead.
At this point the limit is the memory that your machine can allocate to the matrix and the JavaEngine (see table below ignoring headers size)

Matrix size (height, width) x boolean size (4)	Random Access Memory Size
16384 x 16384 x 4	256 Mb x 4
65536 x 65536 x 4	4 Gb x 4
131072 x 131072	16 Gb x 4
262144 x 262144	64 Gb x 4
524228 x 262144	256 Gb x 4

II. Technical guide

Dependencies

Thirt-party used for the application :

- Springboot for the http api
- java.awt* to draw image (image load entirely in memory)
- pngj to draw image (line by line drawing)

Performance and architecture note

The implementation choosed privileged calculation speed over memory usage.
Using a boolean[][] as storing structure benefit an O(1) complexity and a clean structure.
In this application the number of black cells is very low compared to number of white cells. Using a HashMap as storing structure is also powerfull, but remain less adapted for larger usage in which we could store more information : HashMap keys number is limited to 2^31-1, the 2 dimensional qrray is limited by 2^31-1 x 2^31-1 which is far more powerfull considering we have enough memory to run it.

Possible improvement

- As the memory usage is the biggest limit, we can create a system of smaller matrix that will be loaded and save on a harddrive. To do so, introducing a storing class loaded value an fix sized defined aera based on the the indice return by x / areaSize, and y / areaSize. This system can be also implemented to split the generated image into multiple part.
- The drawing system is implemented in the Grid class. It should be separated in a separated drawing class.
- For big simulation, the http api should consider implementing a follow request system to avoid timeout. Request status can be « running » , « completed » or in « error ».

Performance measures

Note : Resolution time does not include image generation time

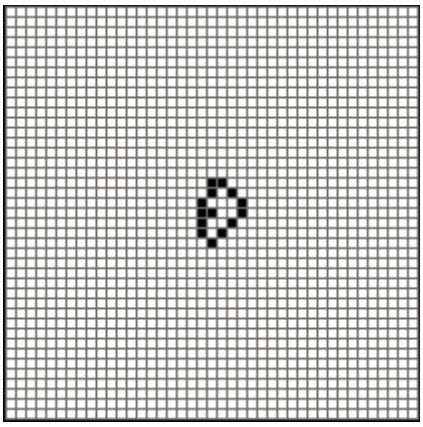
Number of step	Library	Cell Resolution	Resolution time (milliseconds)	Comment	Width x Height

10.000.000	java.awt.*	5	202 ms	No errror	8036 x 15861
20.000.000	java.awt.*	5	320 ms	No error	11311 x 22411
50.000.000	java.awt.*	5	1123 ms	Out of Memory	NA
500.000.000	java.awt.*	1	11 459 ms	No error	11207 x 22372
1.000.000.000	java.awt.*	1	32824 ms	No error	15838 x 31634
2.000.000.000	pnjg	1	58399 ms	No error	44730 x 22385
3.000.000.000	pnjg	1	85899 ms	No error	54781 x 27411

III. Appendix

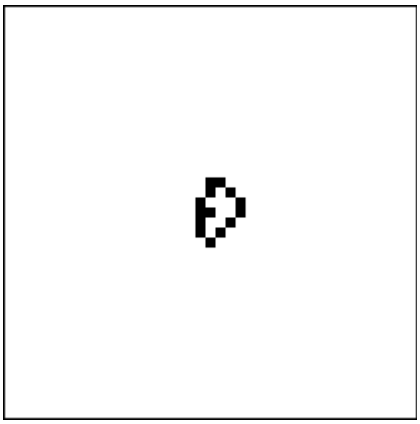
Request 42 steps with grid :

curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "42", **"withGrid" : true** }' http://localhost:8080/fileboard/api/simulation/step
{"details":"Simulation performed in 0 milliseconds is stored in C:\\Users\\Soularkh\\Documents\\DEV\\fileboard\\app\\target\\42.png","msg":"Simulation OK","simulationRequest":
{"step":42,"cellResolution":5,"drawingLibrary":"java.awt","matrixSize":16384,"withGrid":true}}



Same example without the grid :

curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "42", **"withGrid" : false }**' http://localhost:8080/fileboard/api/simulation/step
{"details":"Simulation performed in 0 milliseconds is stored in C:\42.png","msg":"Simulation OK","simulationRequest":
{"step":42,"cellResolution":5,"drawingLibrary":"java.awt","matrixSize":16384,"withGrid":false}}



Example of out of bound errors with default matrixSize of 16384 (the error is visible on server side (java.lang.ArrayIndexOutOfBoundsException: 16384) :

```
curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "500000000", "withGrid" : false }'  http://localhost:8080/fileboard/api/simulation/step
{"timestamp":"2020-10-25T15:21:14.810+00:00","status":500,"error":"Internal Server Error","message":"","path":"/fileboard/api/simulation/step"}
```

Running the same query on a bigger matrix, drawing with pngj is successfull :

```
 curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "500000000", "withGrid" : false, "drawingLibrary" : "pngj", "matrixSize" : 65536 }'
http://localhost:8080/fileboard/api/simulation/step
{"details":"Simulation performed in 8817 milliseconds is stored in C:\\Users\\Documents\\DEV\\fileboard\\app\\target\\500000000.png","msg":"Simulation OK","simulationRequest":
{"step":500000000,"cellResolution":5,"drawingLibrary":"pngj","matrixSize":65536,"withGrid":false}}
```

Exampple with many 4000000000 steps :

```
curl -s -X PUT -H "Accept: application/json" -H "Content-type: application/json" -d '{"step" : "4000000000", "drawingLibrary" : "pngj", "matrixSize" : 65536 }'
http://localhost:8080/fileboard/api/simulation/step
{"details":"Simulation performed in 113572 milliseconds is stored in C:\\Users\\Documents\\DEV\\fileboard\\app\\target\\4000000000.png","msg":"Simulation OK","simulationRequest":
{"step":4000000000,"cellResolution":5,"drawingLibrary":"pngj","matrixSize":65536,"withGrid":false}}
```