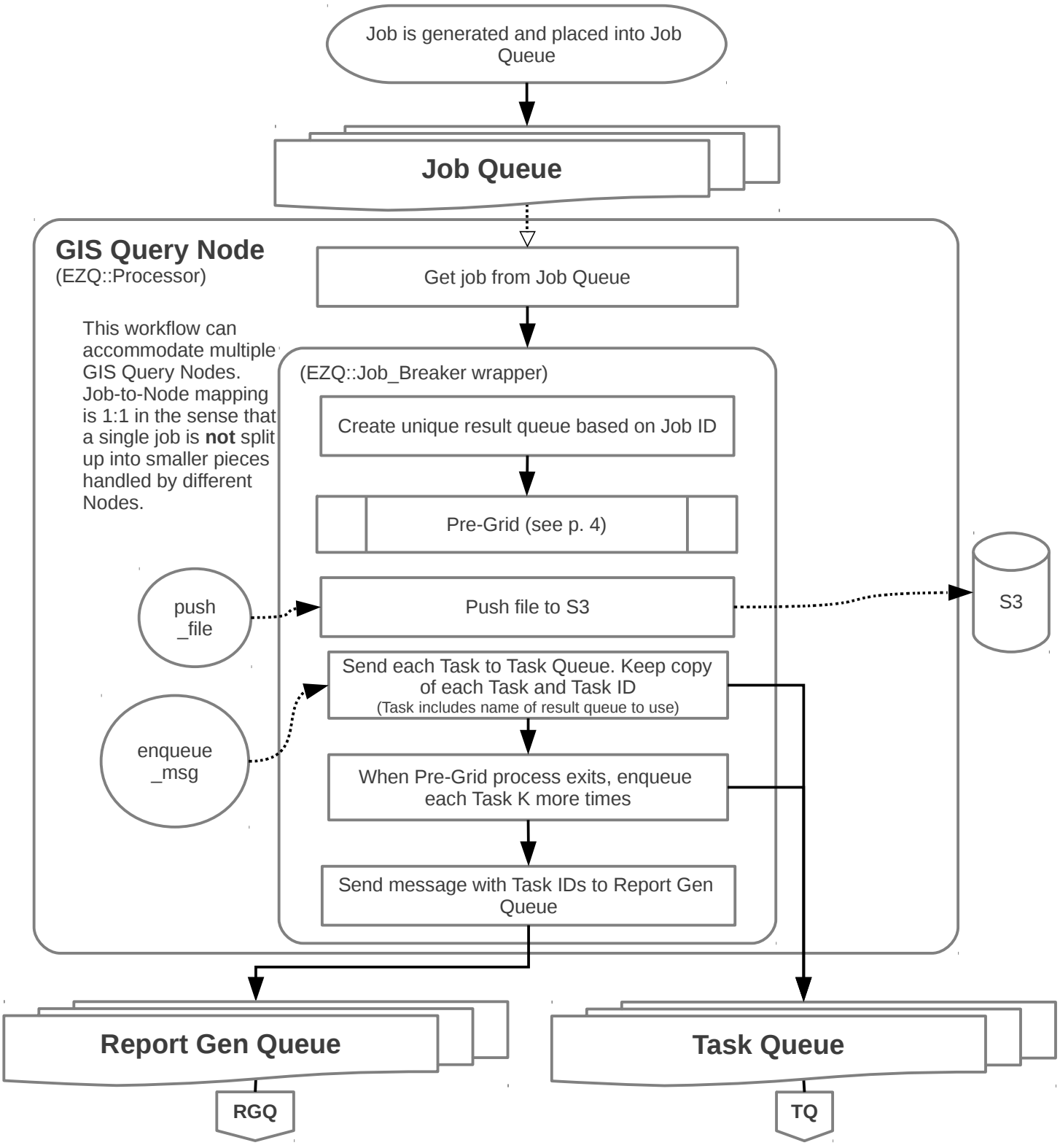
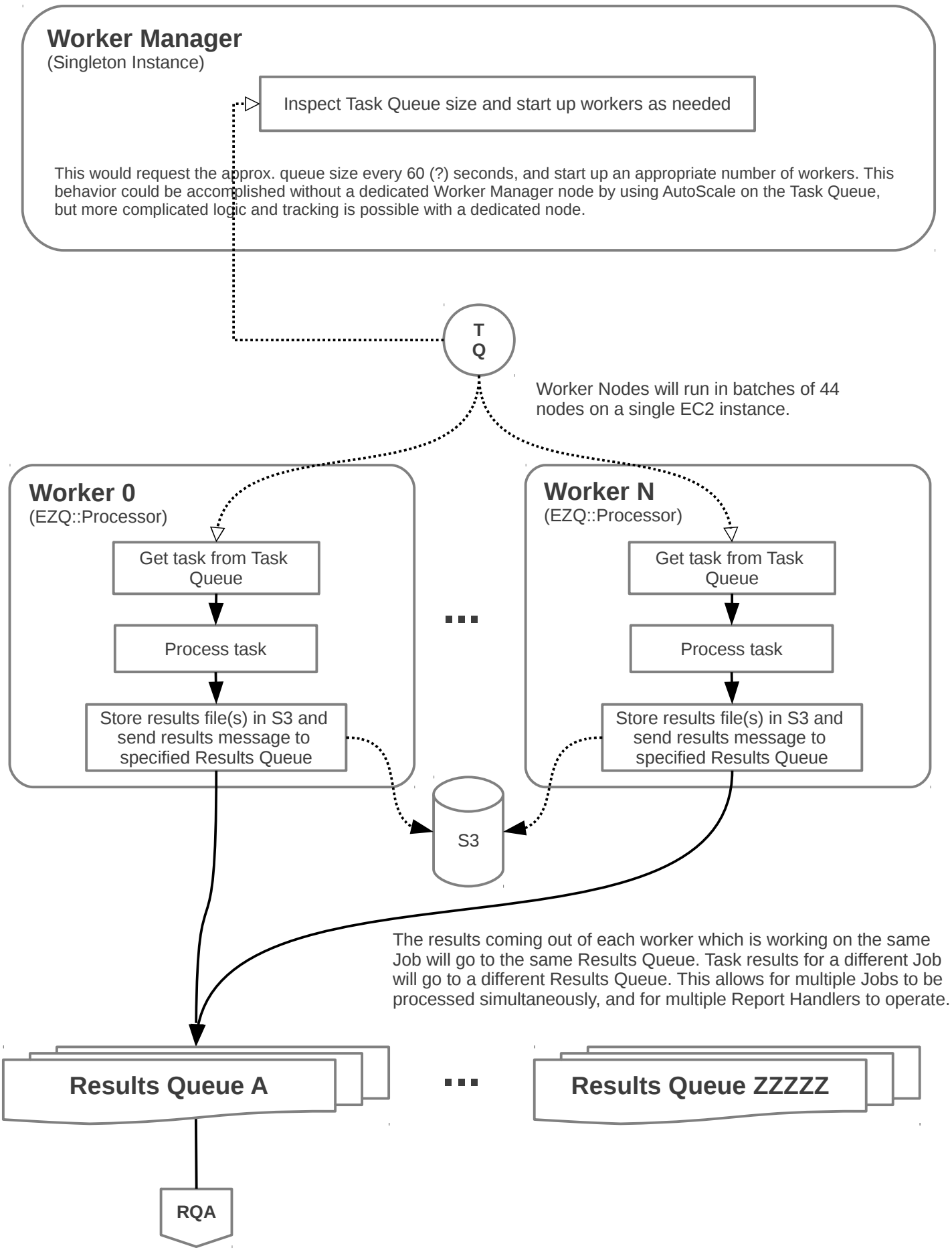
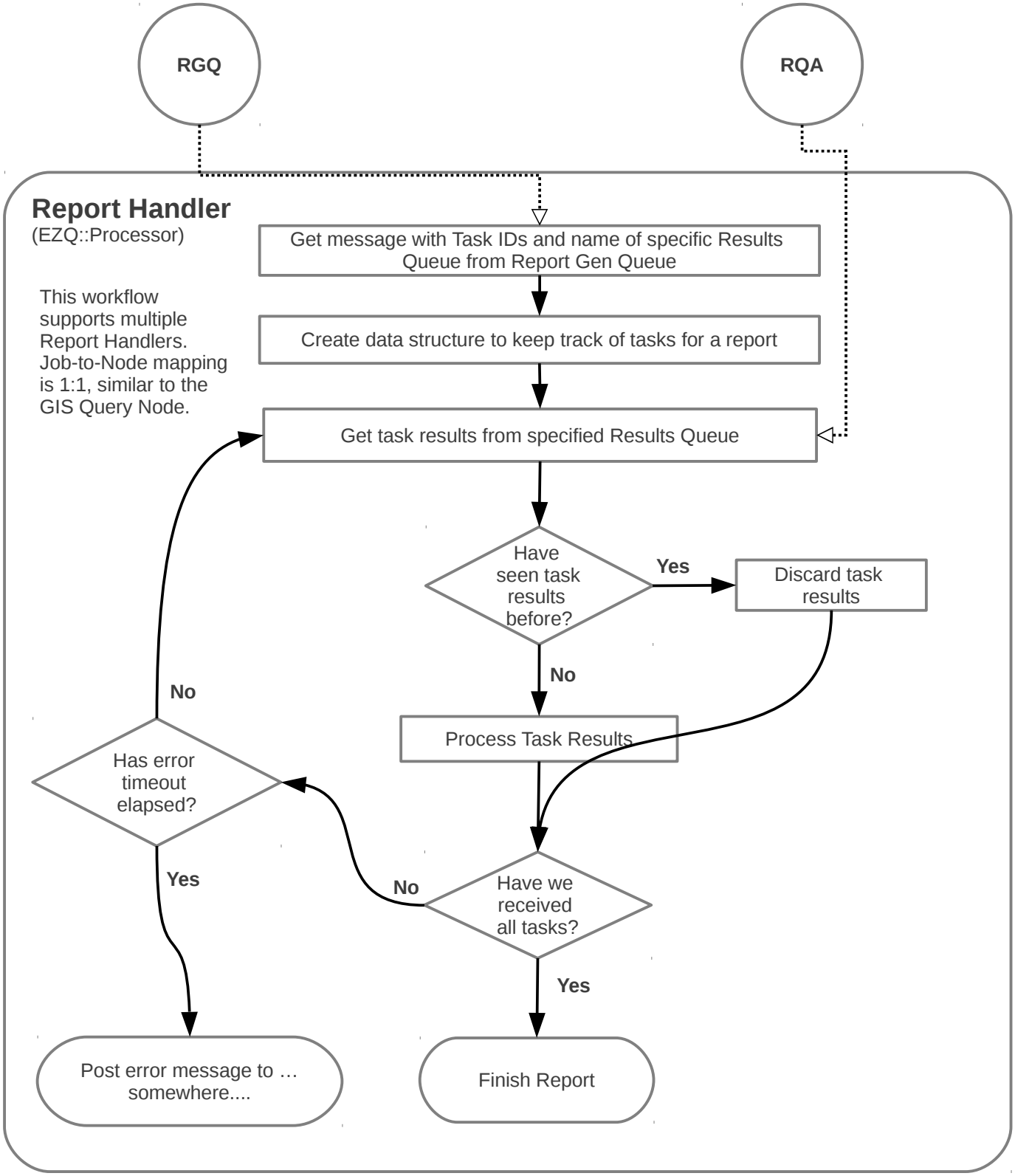
Modified workflow #1 for 6k worker case

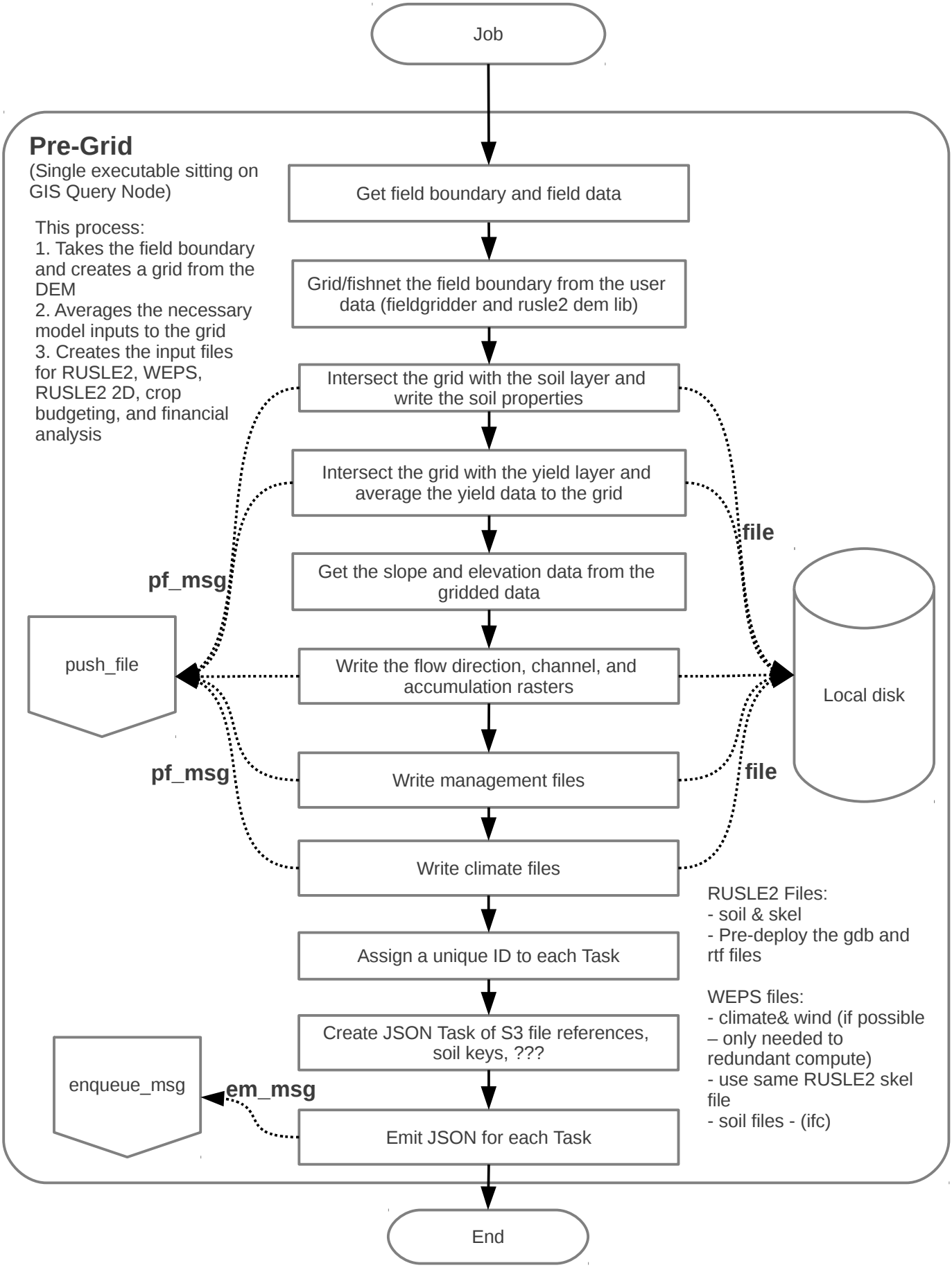
This workflow replaces direct network connections with queues, and uses processes wrapped inside an EZQ::Processor to handle queued messages. The queuing provides task/job/etc. persistence in the case of node failures, as well as allowing easy scaling via running multiple instances of GIS Query Node and Report Handlers.

A ⇢▷ **B** indicates B pulls data from A.
A ⇢▶ **B** indicates A pushes data to B.
A → **B** indicates flow of execution sequence from A to B.




Worker Manager
(Singleton Instance)
Inspect Task Queue size and start up workers as needed
This would request the approx. queue size every 60 (?) seconds, and start up an appropriate number of workers. This behavior could be accomplished without a dedicated Worker Manager node by using AutoScale on the Task Queue, but more complicated logic and tracking is possible with a dedicated node.
T
Q
Worker Nodes will run in batches of 44 nodes on a single EC2 instance.
Worker 0
(EZQ::Processor)
Get task from Task Queue
Process task
Store results file(s) in S3 and send results message to specified Results Queue
Worker N
(EZQ::Processor)
Get task from Task Queue
Process task
Store results file(s) in S3 and send results message to specified Results Queue
S3
The results coming out of each worker which is working on the same Job will go to the same Results Queue. Task results for a different Job will go to a different Results Queue. This allows for multiple Jobs to be processed simultaneously, and for multiple Report Handlers to operate.
Results Queue A
Results Queue ZZZZZ
RQA


RGQ
RQA
Report Handler
(EZQ::Processor)
This workflow supports multiple Report Handlers. Job-to-Node mapping is 1:1, similar to the GIS Query Node.
Get message with Task IDs and name of specific Results Queue from Report Gen Queue
Create data structure to keep track of tasks for a report
Get task results from specified Results Queue
Have seen task results before?
Yes
Discard task results
No
Process Task Results
Has error timeout elapsed?
No
Yes
Have we received all tasks?
No
Yes
Post error message to … somewhere....
Finish Report


Job
Pre-Grid
(Single executable sitting on GIS Query Node)
This process:
1. Takes the field boundary and creates a grid from the DEM
2. Averages the necessary model inputs to the grid
3. Creates the input files for RUSLE2, WEPS, RUSLE2 2D, crop budgeting, and financial analysis
Get field boundary and field data
Grid/fishnet the field boundary from the user data (fieldgridder and rusle2 dem lib)
Intersect the grid with the soil layer and write the soil properties
Intersect the grid with the yield layer and average the yield data to the grid
Get the slope and elevation data from the gridded data
Write the flow direction, channel, and accumulation rasters
Write management files
Write climate files
Assign a unique ID to each Task
Create JSON Task of S3 file references, soil keys, ???
Emit JSON for each Task
pf_msg
push_file
pf_msg
file
file
Local disk
enqueue_msg
em_msg
RUSLE2 Files:
- soil & skel
- Pre-deploy the gdb and rtf files
WEPS files:
- climate& wind (if possible – only needed to redundant compute)
- use same RUSLE2 skel file
- soil files - (ifc)
End