CELERY SCALING
Documentation 2

Purpose of this Document :

- To present comprehensive summary of all tests done.
- To present inferences drawn from each test case.
- To document inferences and settings that resulted in high performance.

Tests Conditions :

- Condition 1 : 8 GB RAM, 4 Cores
- Condition 2 : 16 GB RAM, 8 Cores
- Multiple nodes scenarios

Benchmarking of single node scenario done comprehensively and conclusively. Analysis and settings to be used have been presented below.

Multiple node testing results were inconclusive.

Parameters used to judge test results:

- CPU Load (1 min average)
- RAM Utilisation
- Task completion rate
- Publishing rate
- Consumer Utilisation
- Effeciency : Ratio of total processes used and task completion rate
- Resource Utilisation Ratio (CPU utilisation/Memory Utilisation) : To judge usage of resources.

Summary of Tests :

Condition 1. a :

- 8 GB RAM, 4 cores
- Workers = 12
- Concurrency = 30
- CELERYD_PREFETCH_MULTIPLIER = 4

Server	CPU Load (1 min avg)	Memory	Task Rate	Tasks	Publish/s	Consumer Utilisation	Additional Information
Codebase	3.61	7.62 GB	256/s	50000	500-700/s	38%	Publishing rate bottlenecked.
RabbitMQ	1.06	658 MB	-	-	1000/s	-	-

- A maximum task completion rate of 250-260/s was achieved.
- Memory Utilisation: 97.56%
- CPU Utilisation (based on 1 min average): 90.25%
- CPU Utilisation vs Memory Utilisation : 92.50%

Condition 1 . b :

- 8 GB RAM, 4 cores
- Workers = 16
- Concurrency = 25
- CELERYD_PREFETCH_MULTIPLIER = 4
- 'Ofiar' option used (With this option enabled the worker will only write to processes that are available for work, disabling the prefetch behavior.)

Server	CPU Load (1 min avg)	Memory	Task Rate	Tasks	Consumer Utilisation	Additional Information
Codebase	3.76	7.71 GB	312/s	20000	10-20%	Ofair significantly increased performance.
RabbitMQ	<1	650 MB	-	-	-	-

Conclusion :

- Performance significantly increases when using Ofair option.
- Effeciency (Total processes/ task rate) : 78%
- Memory Utilisation : 98.71%
- CPU Utilisation : 94%
- CPU Utilisation vs Memory Utilisation : 95.22%
- Further tests were performed to test optimal usage of 8GB/4C machine. Turns out that 14 workers, 25 concurrency consistently gave a task rate of ~290/s with 7.23 GB RAM consumption(92.5%) & CPU Load of ~2.
- Optimal usage condition gave effeciency of ~79 – 80%.
- Hence for optimal usage of 8GB/4C machine, settings should be:
  - 14-16 Workers/ 25 Concurrency
  - Use Ofair option
  - Higher RAM would improve CPU utilisation

Condition 2 :

Several tests were performed to benchmark 16GB/8C machine. Following are the test reults of highest task execution rate achieved:

- 16 GB RAM, 8 cores
- Workers = 19
- Concurrency = 40
- CELERYD_PREFETCH_MULTIPLIER = 4
- MAX_TASKS_PER_CHILD = 100
- Ofiar option used.
- Celery updated to v3.1.17 (Cipater)
- RabbitMQ updated to v3.5.0

Server	CPU Load (1 min avg)	Memory	Task Rate	Tasks	Publish/s	Consumer Utilisation	Additional Information
Codebase	3	14.5 GB	645 /s	20000	700 – 800 /s	15 – 20 %	Around 31-34 seconds were taken to complete 20000 tasks
RabbitMQ	<1	650 MB	-	-	1000/s	-	-

Conclusion :

- Performance significantly increased after updating Celery and RabbitMQ.
- Effeciency (Total processes/ task rate) : 85.00%
- Memory Utilisation : 88.05%
- CPU Utilisation : 37.5%
- Resource utilisation ratio : 42.61%
- 5-10 tests were run with identical settings. Results were consistent as 20000 tasks were completed within 31-33 seconds.

- RAM utilisation kept increasing on each sucessive test by about 200 MB. To mitigate max utilisation, Max tasks per child option was used. This setting helped release memory timely.
  Calculation of Max tasks per child also needs to be done precisely, a value of 300 would take around 2,00,000 tasks for all processes to restart.

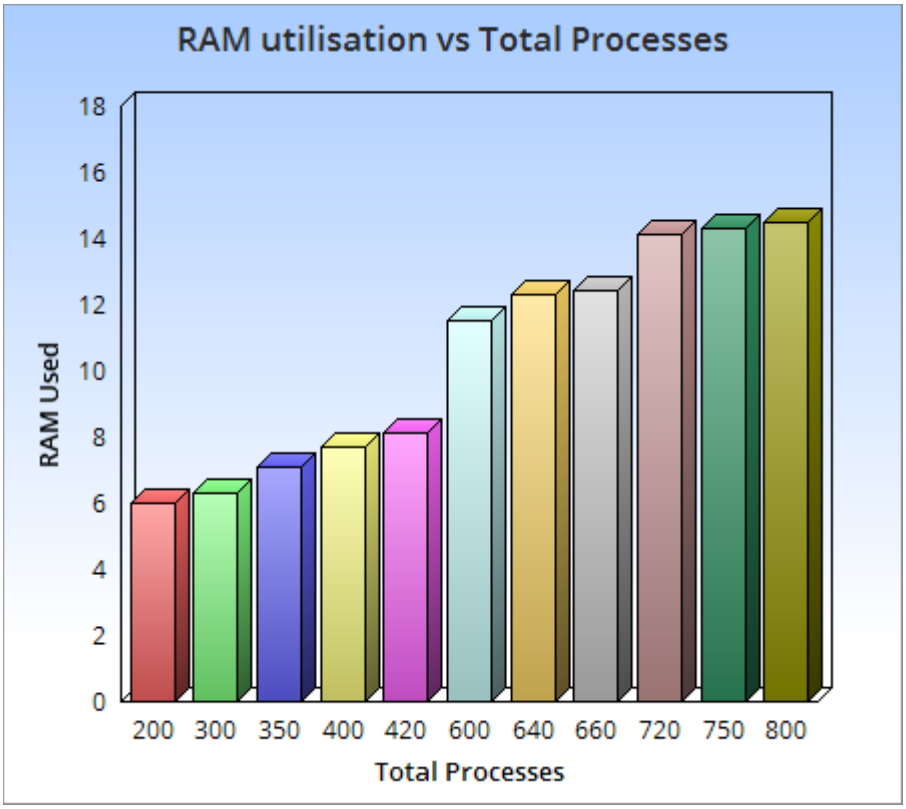
- Another test with identical settings was run for 1,00,000 tasks and results were consistent : 606/s task rate (165 seconds in total, 33sec * 5 = 165 seconds), effeciency : 84%, CPU load : 3

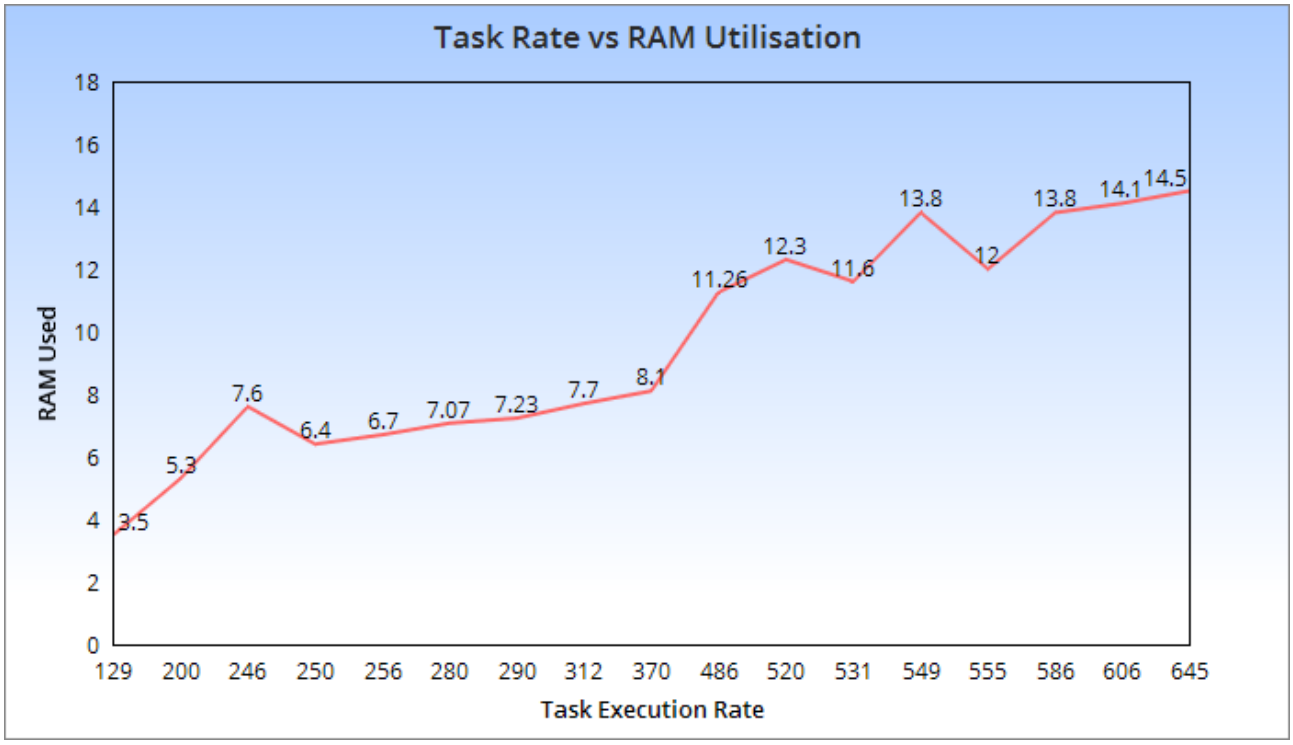
- Big drop in resource utilisation ratio. CPU being significantly under utilised.

- High effeciency indicates optimal worker vs concurrency settings and hardware.

- An 8 core machine with higher RAM (30GB) should give higher resource utilisation.

Analysis Graphs :





Multiple Nodes Scenario :

- Two 4GB/2C machines results :

Server	CPU Load	Memory Consumption	Task Completion Rate	Tasks Completed	Publish	Time Taken	Effeciency	Workers	Concurrency	Total Processes
Codebase 1	2.07	3 GB	105/s	24967		237	87.5	8	15	120
Codebase 2	2.03	3.03 GB	105/s	25033		237	87.5	8	15	120
RabbitMQ					600-800					

Time Taken:	238	Task Execution rate:	210/s	Tasks Completed	50000	Effeciency :	87.50%	Total Processes:	240

- Tests were also run using 8GB/4C machines, but results were not better than single node 16GB/8C machine.

- Tests were also run using 16GB/8C machines, maximum task rates of 720-750/s were observed with RAM usage crossing 13GB for both machines.

  Effeciencies were low for both machines. Resource utilisation ratio was also not high.

- However, task rate crossed 850/s when 3 servers were used.
  - A 4GB/2C machine for producing tasks (celery disabled in this).
  - Tow 16GB/8C machines for consuming tasks.
  - This configuration resulted in a high publishing rate which is favourable for a high task execution rate.

- In conclusion, tests were slightly inconclusive for multiple node scenario, further testing (and more time) is required using seperated producer and cunsumers configuration to strike balance between workers and concurrency and various other settings.

Important Inferences :

- Version updation of Celery and RabbitMQ is important as it significantly increases performance.
- Ofair option significantly increased performance.
- RAM bottlenecks were reached before CPU load in almost all cases.
- Optimal concurrency observed: 25 – 40.
- With every successive run, RAM consumption increases marginally. Hence to mitigate reaching max levels, max tasks per child option must be used.
- Prefetch multiplier with a low value usually gave better performance than with a higher value.
- 8GB/4C and 16GB/8C machines give the highest effeciencies, however resource utilisation is low for 16GB/8C machine due to RAM bottleneck.
- Vertical scaling limits for present requirements should be estimated around these two (8GB/4C and 16GB/8C) machine configurations .
- For multiple nodes scenario, it seems best to seperate producer of tasks from consumers as this increased the publishing rate significantly. Further testing and more data is required to validate this theory.