CELERY SCALING
RESULTS FOR BATCH 1 TESTS

Purpose of this Document:

- To present result analysis of batch 1 tests conducted for Celery scaling.

Conditions of Tests:

- Digital Ocean VMs were used with the following configuration:
  - Codebase Server:
    - IP : 128.199.97.200
    - RAM : 1.96GB
    - Cores : 2
    - Operating System : CentOS 6.5
  - RabbitMQ Server :
    - IP : 128.199.116.114
    - RAM : 996.5 MB
    - Cores : 2
    - Operating System : CentOS 6.5
  - MySQL Server:
    - IP : 128.199.87.9
    - RAM : 996.5 MB
    - Cores : 2
    - Operating System : CentOS 6.5
- Celery version : 3.1.17
- RabbitMQ version : 3.4.4
- MySQL version : 5.6.14

Celery Configuration:

Directory Structure:

```
-- hod_app
    -- hod_app
        -- celery.py
        -- celeryconfig.py
        -- settings.py
```

- celery.py file created in inner hod_app folder.
  - Contains configuration for celery app instance.
  - Broker URL to be specified here.
  - Celeryconfig file linking also done here.

- Celeryconfig.py file created in inner hod_app folder.
  - All celery configuration and parameters to be defined here.
  - **Custom Exchanges : 7**

    - periodic_tasks : Type --> Direct, routing_key --> 'periodic_tasks'
    - sms_reminders : Type --> Direct, routing_key --> 'sms_reminders'
    - whatsapp_reminders : Type --> Direct, routing_key --> 'whatsapp_reminders'
    - phonecall_reminders : Type --> Direct, routing_key --> 'phonecall_reminders'
    - email_reminders : Type --> Direct, routing_key --> 'email_reminders'
    - generic : Type --> direct, routing_key --> 'generic'
    - billing : Type --> direct, routing_key --> 'billing'

- **Custom Queues : 7**
    - periodic_tasks : Exchange -->periodic_tasks
    - sms_reminders : Exchange --> sms_reminders
    - whatsapp_reminders : Exchanges --> whatsapp_reminders
    - email_reminders: Exchange --> email_reminders
    - phonecall_reminders: Exchange --> email_reminders
    - generic : Exchange --> generic
    - billing : Exchange --> billing

- **Tasks Registered with celery : 12**
    - billing.tasks.add_credits
    - billing.tasks.deactivate_expired_customer_purchases_and_realted_recipient_reminders
    - campaign.tasks.deactivate_expired_campaigns
    - customers.tasks.send_miss_call_verification_message
    - customers.tasks.send_sms_set_reminder_for_me
    - customers.tasks.sendmail_task
    - reminders.tasks.clean_voice_messages
    - reminders.tasks.make_phone_call_to_recepient
    - reminders.tasks.send_mail
    - reminders.tasks.send_reminders
    - reminders.tasks.send_sms_to_recepient
    - reminders.tasks.send_whatsapp_message_to_recepient

- Custom Routes were defined to route tasks to their respective queues.

- Tests were conducted using 'reminders.tasks.send_sms_to_recepient'.
    - This calls a utility function 'send_sms'.
    - URL of Project Sleuth used.
    - Request:
      r = requests.post('http://128.199.87.42/receiver/api/post/request/', data={'mode':'sms','token':'8tu30t284t0mc0m0bwl2'})

MySQL Seperation:

- MySQL was seperated from codebase server after initial test revealed primary memory bottleneck.
- Number of tasks fired : 1,00,000
- Remarks: Test could not be completed since MySQL crashed and celery was killed.
- Conclusion: Decision for MySQL seperation taken.

Units, Parameters and Definitions:

- Consumer Utilisation : ratio of publish rate and task completion rate.

- Publish rate, deliver rate, task completion rate measured per second.

- CPU Load measured as load per 1 minute average, 5 minute average, 15 minute average.

- Concurrency: The number of concurrent worker processes/threads/green threads executing tasks. If you're doing mostly I/O you can have more processes, but if mostly CPU-bound, try to keep it close to the number of CPUs on your machine. If not set, the number of CPUs/cores on the host will be used. Defaults to the number of available CPUs.

- CELERY_TASK_RESULT_EXPIRES : Time (in seconds, or a timedelta object) for when after stored task tombstones will be deleted. A built-in periodic task will delete the results after this time (celery.task.backend_cleanup). A value of none or 0 means results will never expire (depending on backend specifications). Default is to expire after 1 day.

- CELERY_IGNORE_RESULT : Whether to store the task return values or not (tombstones). If you still want to store errors, just not successful return values, you can set CELERY_STORE_ERRORS_EVEN_IF_IGNORED.

- CELERYD_MAX_TASKS_PER_CHILD : Maximum number of tasks a pool worker process can execute before it's replaced with a new one. Default is no limit.

- CELERY_DISABLE_RATE_LIMITS : Disable all rate limits, even if tasks has explicit rate limits set.

- CELERY_ACKS_LATE : Late ack means the task messages will be acknowledged after the task has been executed, not just before, which is the default behavior.

Total Test Cases : 7

- ### Test Case 1:

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.02, 0.005, 0
    - Memory Consumption : 667 MB, 780 MB (with flower and workers)
  - RabbitMQ server:
    - CPU Load: 0.08, 0.04, 0.01
    - Memory Consumption: 417 MB
  - Celery Configuration:
    - Workers : 1
    - Concurrency : 20
    - CELERY_TASK_RESULT_EXPIRES = 3600
  - Total tasks fired : 1,00,000

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	0.86,0.35,0.17	1.76 GB	17.58/s	33768	1000/s	-	-	-
RabbitMQ	0.95,0.90,0.14	658 MB	-	33768	1000/s	17.5/s	1%	-

- ### Test Case 2:

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.01, 0.02, 0.0075
    - Memory Consumption : 368.55MB MB, 523 MB (with flower and workers)
  - RabbitMQ server:
    - CPU Load: 0.06, 0.04, 0.01
    - Memory Consumption: 404 MB
  - Celery Configuration:
    - Workers : 1
    - Concurrency : 30
    - CELERY_TASK_RESULT_EXPIRES = 1
    - CELERY_IGNORE_RESULT : True
    - CELERY_DISABLE_RATE_LIMITS = True
    - CELERY_ACKS_LATE = True
  - Total Tasks fired : 1,00,000

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	1.04,0.46,0.23	1.75 BG	26/s	37022	998/s	-	-	Spike in memory consumption (550MB) observerd. Could have been caused by Flower memory leak.
RabbitMQ	0.95,0.90,0.14	658 MB	-	33768	998/s	28/s	2%	

- **Test Case 3:**

  Initial Configuration:
  - Codebase Server : Similar to test casse 2
  - RabbitMQ server : Similar to test case 2
  - Celery Configuration: Similar to test case 2
  - Total Tasks fired : 1,00,000
  - Total tasks completed : 1,00,000
  - This test was conducted to investigate memory spike.

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	1.04,0.46,0.23	1.75 BG	26/s	1,00,000	998/s	-	-	Memory spike not observed. Flower was not running.
RabbitMQ	0.95,0.90,0.14	658 MB	-	1,00,000	998/s	28/s	2%	

- **Test Case 4:**

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.02, 0.03, 0.005
    - Memory Consumption: 212 MB, 255MB (flower, htop, top, 3 workers)
  - RabbitMQ server :
    - CPU Load : 0.13, 0.07, 0.02
    - Memory Consumption: 404 MB
  - Celery Configuration:
    - Workers : 3
    - Concurrency (per worker) : 10
    - CELERY_TASK_RESULT_EXPIRES = 1
    - CELERY_IGNORE_RESULT : True
    - CELERY_DISABLE_RATE_LIMITS = True
    - CELERY_ACKS_LATE = True

  - Total Tasks fired : 50,000
  - Total tasks completed : 50,000
  - This test was conducted to investigate multiple worker scenario.

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	1.13, 0.68, 0.51	1.45 GB	26/s	50,000	1000/s	-	-	Memory spike not observed. Flower was not running.
RabbitMQ	0.77,0.27,0.09	542 MB	-	1,00,000	998/s	28/s	3%	

REMARKS :

- Multiple worker nodes (3 workers, 10 concurrency each) scenario performed equivalent (marginally better) as compared to simgle worker scenario (1 worker, 30 concurrency).

- Memory spike was not observerd.

- Tasks were being equally distributed among the 3 worker nodes.
  W1 : 16670 tasks
  W2 : 16666 tasks
  W3 : 16667 tasks

- Memory was not being released after completion of all tasks.

- Another test with similar configuration conducted without stopping workers. This was done to check any change in performance. No noticable change in performance was observed.

- **Test Case 5:**

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.035, 0.04, 0.16
    - Memory Consumption: 1.11GB
  - RabbitMQ server :
    - CPU Load : 0.29, 0.09, 0.03
    - Memory Consumption: 410 MB
  - Celery Configuration:
    - Workers : 3
    - Concurrency (per worker) : 17
    - CELERY_TASK_RESULT_EXPIRES = 1
    - CELERY_IGNORE_RESULT : True
    - CELERY_DISABLE_RATE_LIMITS = True
    - CELERY_ACKS_LATE = True
  - Total Tasks fired : 50,000
  - Total tasks completed : 50,000
  - This test was conducted to investigate multiple worker scenario with higher concurrency.

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	2.54, 1.78, 1.33	1.56 GB	41.22/s	50,000	1000/s	-	-	Memory spike not observed. Flower was not running.
RabbitMQ	1.11, 0.51, 0.18	602 MB	-	50,000	1000/s	41/s	3%	

REMARKS :

- Total concurrency was 51, although task completion rate was 41.22/s.
- This shows bottleneck in task completion.
- CPU Load showed a maximum of 2.54 (1 min average) and was above 2 consistently throughout observation time.
- Hence CPU bottleneck for Dual core was reached.
- Memory spiker was not observed.

- **Test Case 6:**

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.0, 0.01, 0.025
    - Memory Consumption: 203 MB, 724 (4 workers, shell_plus)
  - RabbitMQ server :
    - CPU Load : 0.23, 0.08, 0.02
    - Memory Consumption: 392 MB
  - Celery Configuration:
    - Workers : 4
    - Concurrency (per worker) : 12
    - CELERY_TASK_RESULT_EXPIRES = 1
    - CELERY_IGNORE_RESULT : True
    - CELERY_DISABLE_RATE_LIMITS = True
    - CELERY_ACKS_LATE = True
  - Total Tasks fired : 50,000
  - Total tasks completed : 50,000
  - This test was conducted to investigate worker/concurrency limits of a 2GB, dual core DO VM.

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	3.26, 2.1, 1.57	1.73 GB	41.6/s	50,000	1000/s	-	-	-
RabbitMQ	0.94, 0.36, 0.17	610 MB	-	50,000	1000/s	41/s	4%	-

REMARKS :

- Total concurrency was 48, although task completion rate was 41.6/s.
- More tests with similar configuration were conducted and task completion rates were either equivalent or marginally better.
- A marginal increase in the task completion rate indicates that decrease in overall concurrency and moderate increase in number of workers, increases performance.
- Test shows that balance between number of worker pool processes and number of worker nodes must be reached through experiments and analysis.
- Memory spike was not observed.
- Next test was conducted without stopping workers, hence memory was not released.

- **Test Case 7:**

  Initial Configuration:
  - Codebase Server :
    - CPU Load : 0.57, 0.03, 0.53
    - Memory Consumption: 1.73 GB
  - RabbitMQ server :
    - CPU Load : 0.23, 0.08, 0.02
    - Memory Consumption: 412 MB
  - Celery Configuration:
    - Workers : 4
    - Concurrency (per worker) : 12
    - CELERY_TASK_RESULT_EXPIRES = 1
    - CELERY_IGNORE_RESULT : True
    - CELERY_DISABLE_RATE_LIMITS = True
    - CELERY_ACKS_LATE = True
    - CELERYD_MAX_TASKS_PER_CHILD = 200

  - Total Tasks fired : 50,000
  - Total tasks completed : 50,000
  - This test was conducted to investigate effect of CELERYD_MAX_TASKS_PER_CHILD parameter.

Server	CPU Load	Memory	Task Rate	Tasks	Publish/s	Deliver/s	Consumer Utilisation	Additional Information
Codebase	4.02 2.57, 1.81	1.83 GB	40.6/s	50,000	1000/s	-	-	-
RabbitMQ	0.94, 0.36, 0.17	610 MB	-	50,000	1000/s	41/s	4%	-

REMARKS :

- Memory spike was not observed.
- Memory utilisation dipped as (almost) constant time intervals.
- CELERYD_MAX_TASKS_PER_CHILD could help prevent memory leak.

Conclusion:

- Target task completion rate to be achieved : 950 tasks per second.
- Limits of 2BG Dual core machine were observed.
- CPU processing limits proved to be a bottleneck.
- Maximum worker processes limit for a 2GB Dual core machine can be estimated at around 40 -45.
- Maximum task completion observed : 42/s
- Target Consumer utilisation must be 100%.
- MySQL seperation had a positive effect on performance.
- Broker sepration has a positive effect on performance.
- Further tests are being conducted on 8GB, quad core machines.
- Vertical limits of a single 8GB, 4 core machine will be estimated.
- Benchmarking of a 8GB machine will help in creating a generalised formula for task completion rate vs number of servers vs consumer utilisation.