Analyzing CBT Benchmarks in Jupyter

Nov 16, 2016

Ben Lynch

Minnesota Supercomputing Institute
Background

- MSI has 1 production Ceph cluster used for Tier 2 storage.
- The Tier 2 storage is available exclusively through the S3 API.
- It has 3.2 PB of raw disk that we use with 4+2 erasure coding.
- Deployed running Firefly, now it's running Jewel

- MSI is deploying a new Ceph cluster as part of a new OpenStack cloud resource.
- We need to run some tests on our new hardware
Goals

Test the hardware and configuration for a new Ceph cluster

- Identify faulty hardware or configuration

Tune Ceph and system parameters to improve performance

- Number of placement groups, recovery threads, FileStore settings ...
- System network parameters
- TCMalloc memory usage

Test recovery time for various failures or system changes

- Single HDD failure
- Single SSD failure
- 12 HDD OSD failure
- Placement group split from 4096 to 8192 pgs
More Goals
Test less-supported features

- BlueStore

Collect base storage performance for reference when we move to application benchmarks on OpenStack VMs

- Bandwidth for SATA and SSD data pools
- IOPs and latency distribution for SATA and SSD pools
What is CBT?

Ceph Benchmark Tool

CBT is a python tool for configuring Ceph clusters and running tests.

- A passphraseless ssh key is setup for cbt
- cbt user is given sudo privileges
- CBT uses pdsh to
  - configure the monitor
  - configure the OSDs
  - format disks
  - run tests
  - copy test results back to server where cbt is running

What CBT does not do

Parsing

- CBT does not parse the output from the tests
What is Jupyter?

- Web-based environment for data analysis and more
- Evolved from iPython notebooks
- Run R, Python, or Julia

```python
import cbtworkspace

In [2]:
We input the location for our cbt output.
In [3]:
cw = cbtworkspace.cbtWorkspace('/Users/blynch/Documents/Code/benchmarks/stratus')
In [4]:
cw.ls()
Out[4]:
['test1',
 'test10',
 'test100',
 'test101',
 'test102',
 'test103',
 'test104',
 'test105',
 'test11',
 'test12',
 'test13',
 'test14',
 'test15',
 'test16',
 'test17',
 'test18',
 'test19',
 'test20',
 'test21',
 'test22',
 'test23',
 'test24',
 'test25',
 'test98',
 'test99']
```

`ls()` looks at all the subdirectories and determines which ones look like CBT output.
We can get a general description of a test

cw.describe('test13')

<table>
<thead>
<tr>
<th>Archive</th>
<th>Benchmark</th>
<th>Size</th>
<th>Test</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>test13</td>
<td>4096</td>
<td>seq</td>
<td>9.07837</td>
<td>0.00000</td>
</tr>
<tr>
<td>1</td>
<td>test13</td>
<td>4096</td>
<td>seq</td>
<td>9.20957</td>
<td>0.00000</td>
</tr>
<tr>
<td>2</td>
<td>test13</td>
<td>4096</td>
<td>seq</td>
<td>9.01341</td>
<td>0.00000</td>
</tr>
<tr>
<td>3</td>
<td>test13</td>
<td>4096</td>
<td>seq</td>
<td>8.78927</td>
<td>0.00000</td>
</tr>
<tr>
<td>4</td>
<td>test13</td>
<td>4096</td>
<td>write</td>
<td>0.00000</td>
<td>8.68023</td>
</tr>
<tr>
<td>5</td>
<td>test13</td>
<td>4096</td>
<td>write</td>
<td>0.00000</td>
<td>8.79946</td>
</tr>
<tr>
<td>6</td>
<td>test13</td>
<td>4096</td>
<td>write</td>
<td>0.00000</td>
<td>8.86850</td>
</tr>
<tr>
<td>7</td>
<td>test13</td>
<td>4096</td>
<td>write</td>
<td>0.00000</td>
<td>8.40140</td>
</tr>
<tr>
<td>8</td>
<td>test13</td>
<td>4194304</td>
<td>seq</td>
<td>365.91600</td>
<td>0.00000</td>
</tr>
<tr>
<td>9</td>
<td>test13</td>
<td>4194304</td>
<td>seq</td>
<td>370.25200</td>
<td>0.00000</td>
</tr>
<tr>
<td>10</td>
<td>test13</td>
<td>4194304</td>
<td>seq</td>
<td>367.47700</td>
<td>0.00000</td>
</tr>
<tr>
<td>11</td>
<td>test13</td>
<td>4194304</td>
<td>seq</td>
<td>369.61100</td>
<td>0.00000</td>
</tr>
<tr>
<td>12</td>
<td>test13</td>
<td>4194304</td>
<td>write</td>
<td>0.00000</td>
<td>489.35300</td>
</tr>
<tr>
<td>13</td>
<td>test13</td>
<td>4194304</td>
<td>write</td>
<td>0.00000</td>
<td>507.49200</td>
</tr>
<tr>
<td>14</td>
<td>test13</td>
<td>4194304</td>
<td>write</td>
<td>0.00000</td>
<td>497.59000</td>
</tr>
<tr>
<td>15</td>
<td>test13</td>
<td>4194304</td>
<td>write</td>
<td>0.00000</td>
<td>513.46000</td>
</tr>
</tbody>
</table>
Bandwidth for RADOS Write Benchmark on a single node (4k - 4M block sizes)

In [7]:
cw.bar_graph(['test4', 'rados_write', title='Sequential Write RADOS Benchmark',
              log=False, bar_label=False)

<matplotlib.figure.Figure at 0x113f23320>
Comparison of a single storage node before and after proper partition alignment on the NVMe

```
cw.bar_graph(['test4', 'test5'], 'rados_write',
              title='Sequential Write RADOS Benchmark', log=False)
```

![Bar graph showing performance comparison](attachment:bar_graph.png)
Hardware

- 32 x 40GigE
- OSD 01
- OSD 02
- OSD 03
- 10 GigE
- 40 GigE
- Compute 001
- Compute 002
- Compute 003
- Compute 004
- Compute 005
Endurance and Recovery Tests

Step 1. Fill cluster to ~50% full and test if it maintains reasonable write speeds

Step 2. Remove a disk and measure time to recover

```
In [27]:
plt.figure(num=None, figsize=(15, 8), dpi=80, facecolor='w', edgecolor='k')
plt.ylim((0,1200))
plt.xlim((0,29000))
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1177ce048>
```
Filling the cluster to 50.9% in 20:17:30

In [31]:
plt.figure(num=None, figsize=(15, 8), dpi=80, facecolor='w', edgecolor='k')
plt.ylabel('MB/s')
plt.xlabel('Time (s)')
plt.ylim((0,1500))
plt.xlim((0,75000))
plt.scatter(x, y, s=area, c=colors, alpha=0.5)

Out[31]:<matplotlib.collections.PathCollection at 0x116f62a58>
SSD and HDD 3X Replicated Read Bandwidth

In [41]:

cw.bar_graph(['test95', 'test70'], 'rados_read_seq', title='SSD and HDD 3X Replicated Read Bandwidth', log=True, bar_label=True)

<matplotlib.figure.Figure at 0x1183d7b70>
BlueStore

In [43]:
cw.bar_graph(["test101","test70"],'rados_write',title='BlueStore vs FileStore Write Bandwidth',
               log=True, bar_label=True, legend_labels=['BlueStore', 'FileStore'])

<matplotlib.figure.Figure at 0x117fd3630>

In [46]:
cw.geomean('test101','rados_write')/cw.geomean('test70','rados_write')

Out[46]:
1.2428060674586985

1.2428060674586985
Acknowledgements

Graham Allan

Matt Mix