Notes.7-21-17

July 21, 2017

1 A Healthy ParSplice Cluster

Danny said that a higher temperature would create more activity. He mentioned the `PARPSLICE_TASK_SEGMENT`, `Temperature`, and `InitTemperature` variables in the `ps-config.xml` configuration file for the LAMMPS engine that ParSplice uses. He suggested starting with a temperature of 2000 and an initial temperature of 400; then increasing from there. In this notebook, we try parameters sweeps on our old cluster (9 nodes each with 4 cores and 8GB of RAM) and on a CloudLab cluster (7 nodes each with 36 cores and 64GB of RAM). First, we define some helper functions for plotting `times.out` and the performance counters collected by our customized LevelDB version.

```
In [1]: from helpers import *
        %matplotlib inline
```

1.1 Unhealthy ParSplice

1.1.1 Experiment 1: Scale Temperatures Up

Setup: Old Cluster, 30 Minute Runtime

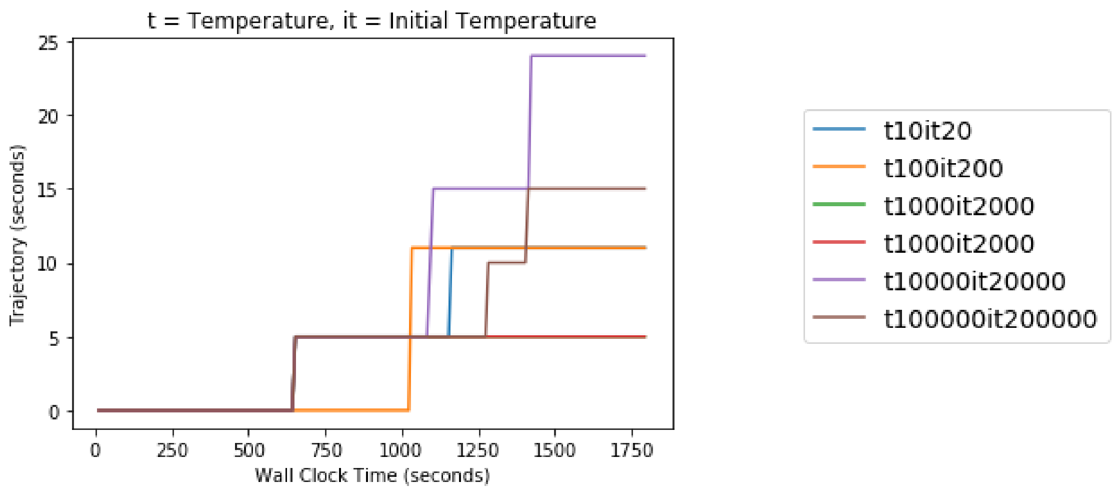
First we scale both temperatures up by a factor of 10, starting at a temperature of 10 and and initial temperature of 20. In the nomenclature below, `t` is temperature and `it` is initial temperature.

```
In [2]: results = ["results-old/results-t10it20.xml",
                   "results-old/results-t100it200.xml",
                   "results-old/results-t1000it2000.xml",
                   "results-old/results-t1000it2000.xml",
                   "results-old/results-t10000it20000.xml",
                   "results-old/results-t100000it200000.xml"]
```

1.1.2 Overall Health (`times.out`)

These lines should be linear. One tool to make sure the system is healthy is the `times.out` file produced by ParSplice. The relationship between wall clock time and trajectory duration should be linear because it means the system is keeping up with the simulation load and is mirroring real time. According to Danny "In an ideal case, the trajectory length would proportional to the wall-clock time. Of course, seeing a linear trend alone does not tell you that you are running at the optimal simulation rate (given by the slope), but usually a nice straight line means you are."

```
In [3]: plot_times(results)
```

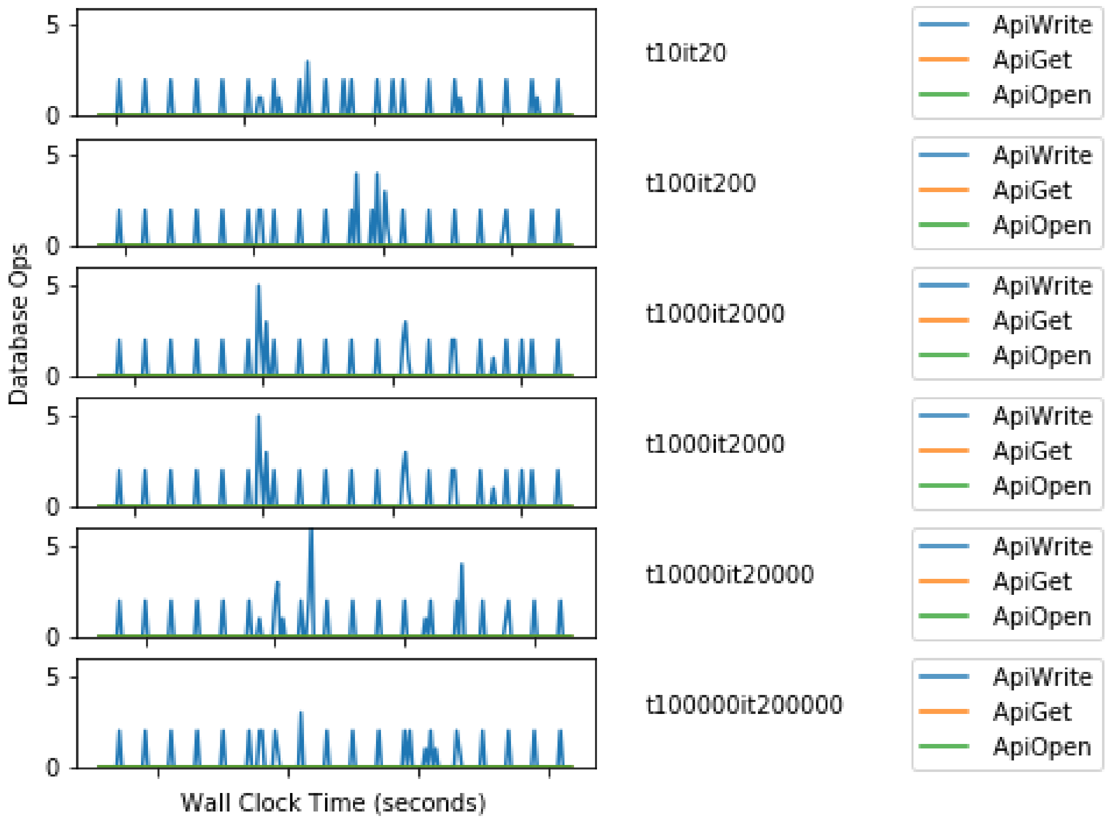


It *looks* like the higher the temperature, the more activity. But the database does not see much action.

1.1.3 LevelDB Performance Counters

In the graphs below, we plot the activity on the persistent DB:

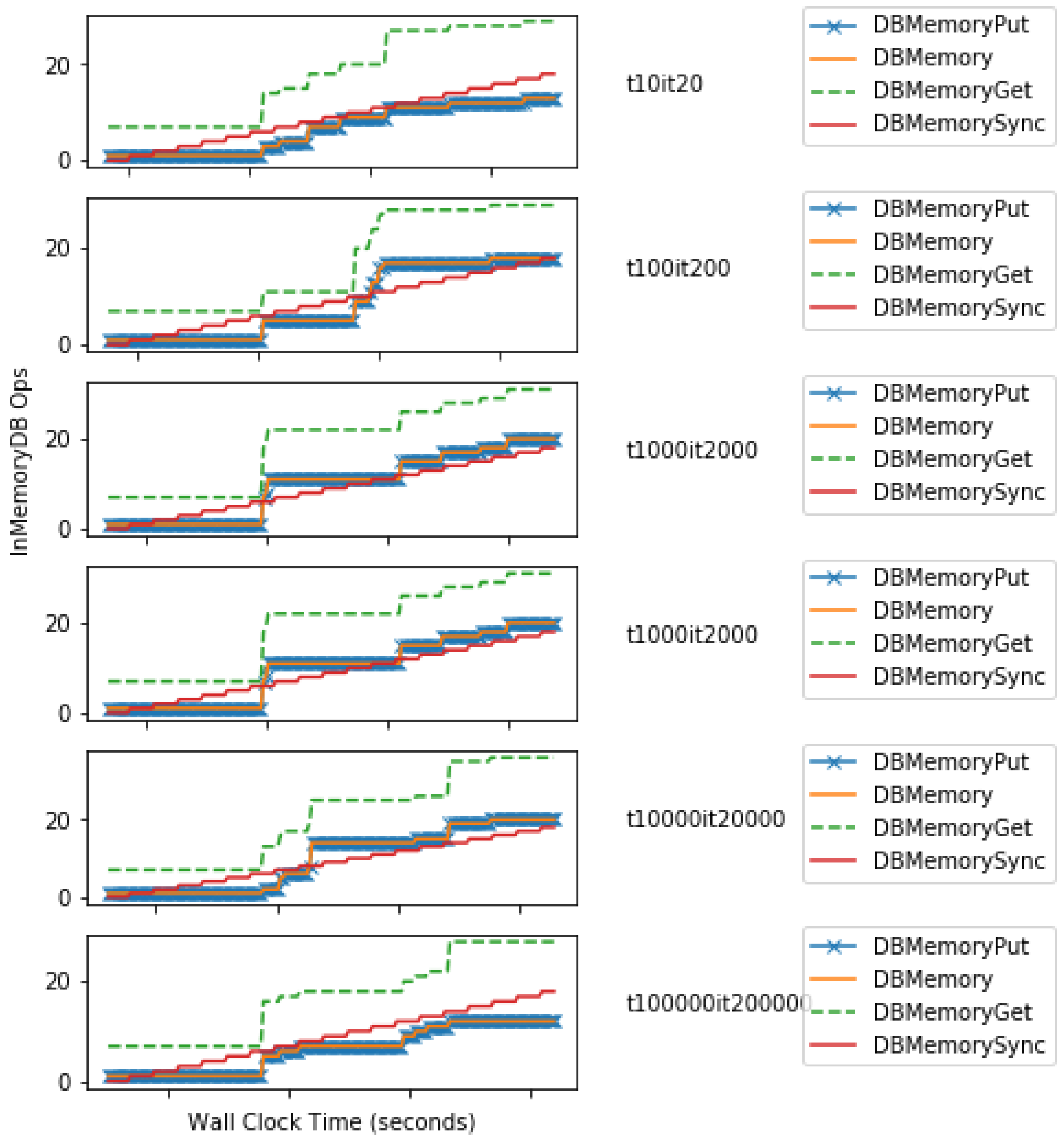
```
In [4]: plot_dbactivity(results)
```



This does not look right. We only see writes and they are not that frequent. There is definitely not a correlation between the temperature and the number of database operations.

ParSplice Performance Counters Finally, we can take a look at the performance counters we added to ParSplice. We look at the in-memory database operations. Recall that this database is just an STL C++ unordered map that sits in front of the persistent database.

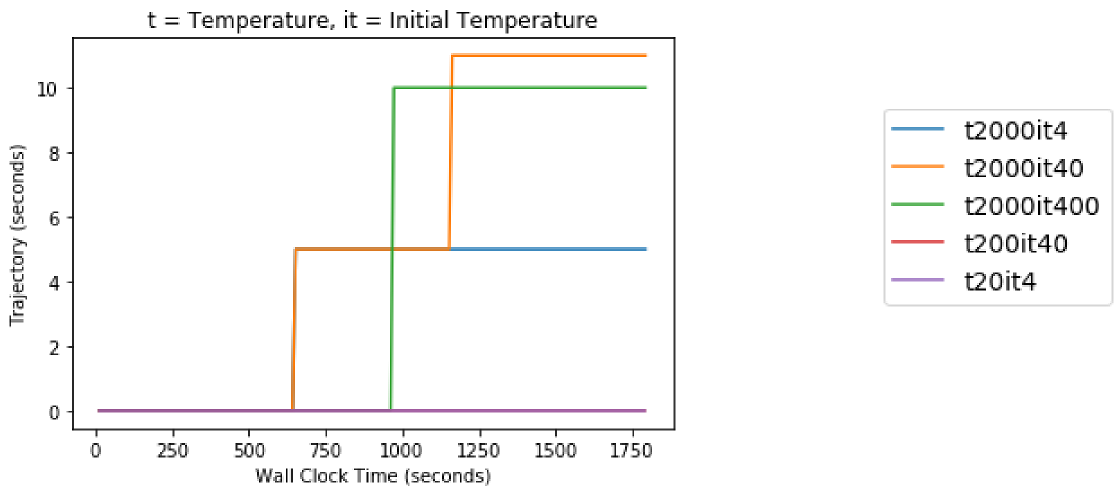
```
In [5]: plot_psactivity(results, dbmemnode="issdm-12")
```

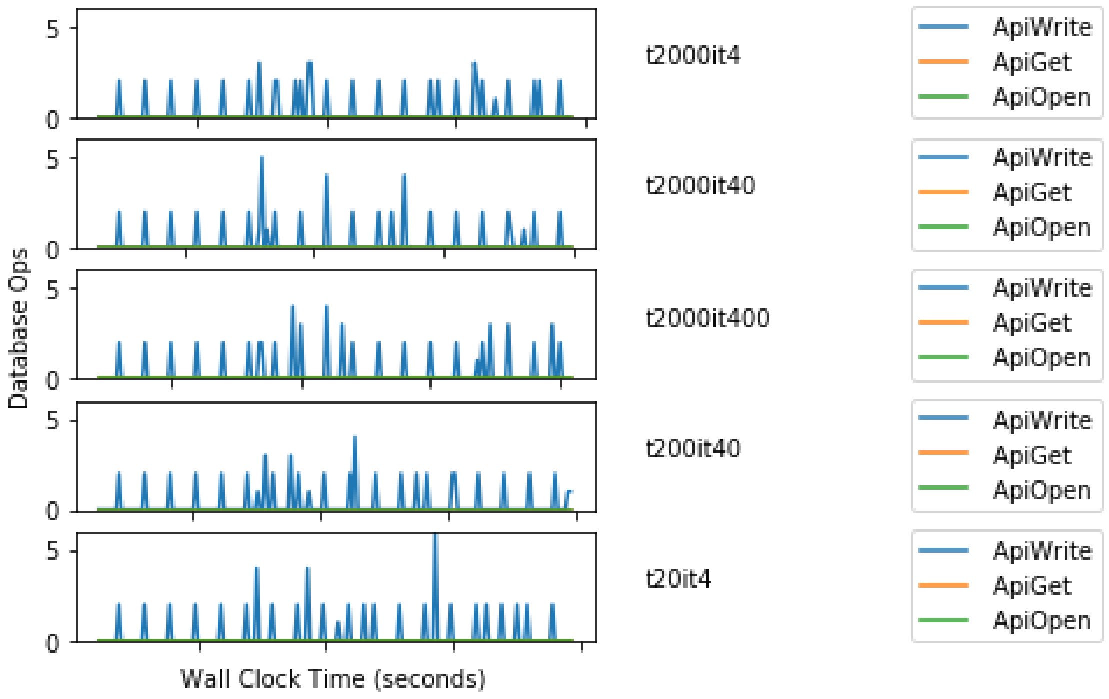


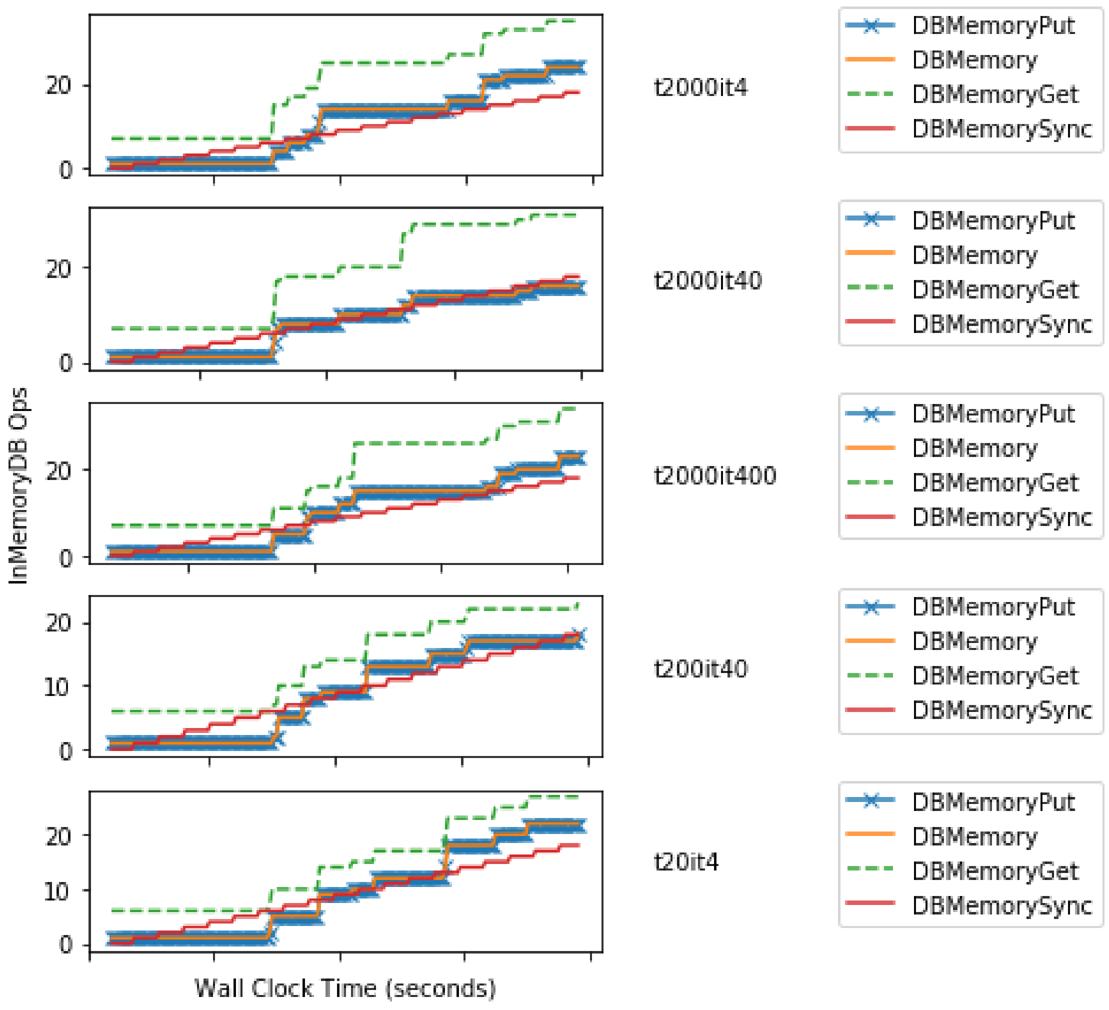
1.1.4 Experiment 2: Fix Temperature, Scale Initial Temperature

Perhaps I was playing with the wrong temperatures. Let's scale along another axis.

```
In [6]: results = ["results-old/results-t2000it4.xml",
                   "results-old/results-t2000it40.xml",
                   "results-old/results-t2000it400.xml",
                   "results-old/results-t200it40.xml",
                   "results-old/results-t20it4.xml"
                  ]
        plot_times(results)
        plot_dbactivity(results)
        plot_psactivity(results)
```




Database Ops
Wall Clock Time (seconds)
5
0
t2000it4
t2000it40
t2000it400
t200it40
t20it4
ApiWrite
ApiGet
ApiOpen



Again, we do not see a correlation and there are long periods of inactivity.

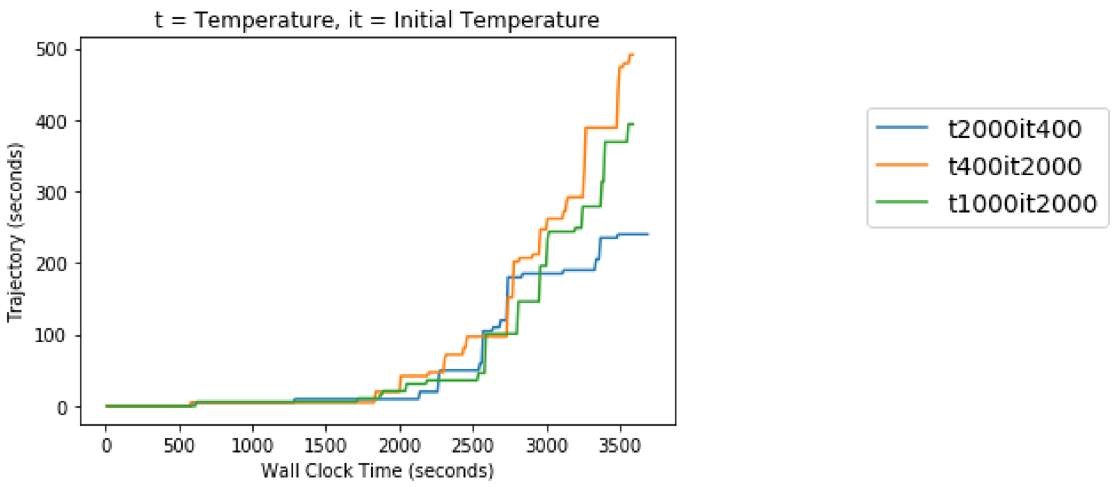
1.2 Healthy ParSplice

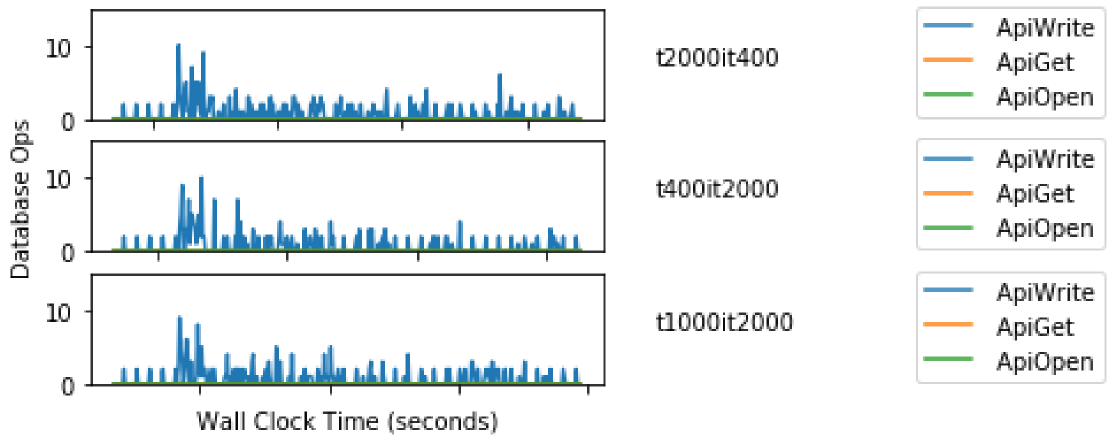
1.2.1 1.3 Experiment 3: Bigger Cluster, Increase the Number of Workers

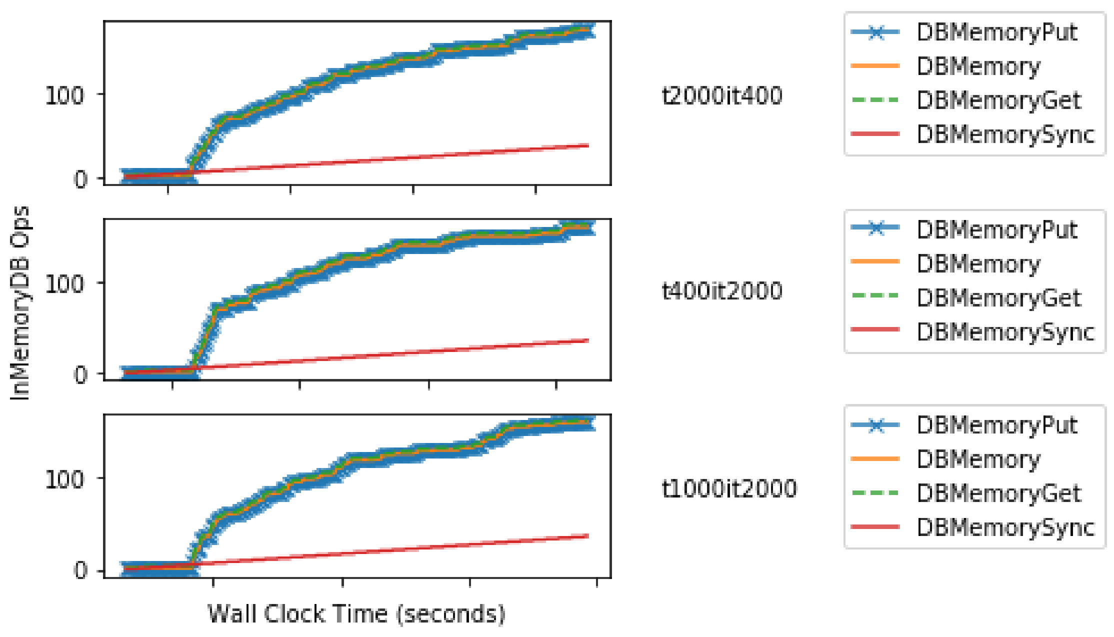
Setup: CloudLab Cluster, 1 Hour Runtime

We run the job with more workers to see if there is more database activity in the latter stages.

```
In [7]: results = ["results-t2000it400.xml", "results-t400it2000.xml", "results-t1000it2000.xml"
        plot_times(results)
        plot_dbactivity(results, dbnode="node-1", ylim=15)
        plot_psactivity(results, dbmemnode="node-3")
```


t = Temperature, it = Initial Temperature
Trajectory (seconds)
Wall Clock Time (seconds)
t2000it400
t400it2000
t1000it2000


Database Ops
Wall Clock Time (seconds)
t2000it400
t400it2000
t1000it2000
ApiWrite
ApiGet
ApiOpen



I showed this to Danny and he said this looks like a healthy setup. Some observations he made about my experiments:

- this U02 input does not stress the system enough, that's why the databases (both persistent and in-memory) can adequately service all reqeusts. We should see more transitions with the nanoparticle simulation he added yesterday. We want to make sure that the temperatures are set at around 700K for these runs. If these still does not stress our system, we can try the KMC engine which simulates a bigger cluster by issuing and executing the database puts/gets with `no-op` calculations.
- the `times.out` results look good. The `t400it2000` experiment flat-lining makes sense because the temperature should be `2x` greater than the initial temperature for the run to make sense.
- the 1-to-1 relationship of the in-memory database puts/gets looks normal. For this run, the cache on the work manager (implemented as an STL unordered map) services all producer requests. On a run with less locality, we should see this relationship lessen a bit as producers may need data from a cache on a different work manager.
- the persistent database does not see any `get()`s because all reads are satisfied from the in-memory database.

```
In [ ]:
```