Implementing Replica Exchange Molecular Dynamics Using Work Queue

Maritza Smith-Romero
Department of Computer Science,
Southern University A&M College, Baton Rouge Louisiana

Abstract
Replica exchange molecular dynamics (REMD) has been used by researchers for improving sampling rates. In traditional t-REMD, individual replicas run in parallel at different temperatures. Temperatures of the neighboring replicas are exchanged based on a criterion. In this research, we implemented REMD on Work Queue framework and evaluated the performance of the Work Queue based REMD. The master-worker framework was applied into the implementation. All replicas were assigned to workers; outputs from all replicas were gathered in master and processing finished. NAMD was the MD software package used in the implementation for the MD simulation. The programming language used in this project was C.

Materials and Design
All of the work was done on a Dell, Model: PowerEdge T410 server, with an Intel Xeon X5650 CPU and a processing speed of 2.67G Hz.

REMD was implemented on WorkQueue and embedded in a shell script that, allows the user to determine the number of replicas to be evaluated and the number of "workers" to be used for said evaluation. This script also allows the computer to be the only "entity" involved in determining the speed with which REMD runs, and tracks the "runtime" of each implementation of REMD.

The number of "replicas" evaluated ranged from 2 to 32, and the number of "workers" used to perform the evaluations ranged from 1 to 32, with a maximum number of workers determined by the maximum number of replicas. The increase was achieved by doubling each replica test set until the number 32 was reached.

Results
Each evaluation was done five times using the same replicas each time, and an average calculated. The evaluations were subdivided into test sets on the basis of the number of replicas being evaluated. Each average runtime within each test set was compared to the average runtime that preceded it and the percentage difference was calculated. This is depicted in Figure 1.

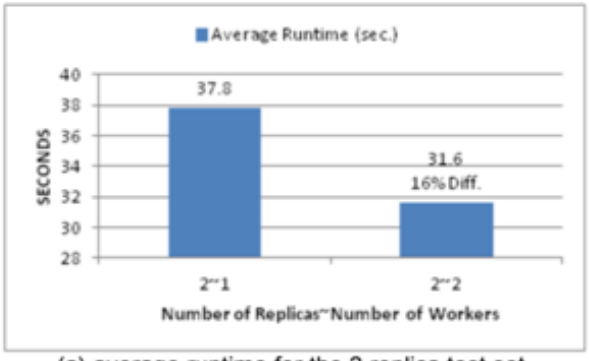
Figure 1(a) depicts the average runtime for the 2 replica test set and the percent difference between its members.

Figure 1(b) depicts the average runtime for the 4 replica test set and the percent difference between its members.

Figure 1(c) depicts the average runtime for the 8 replica test set and the percent differences between its members.

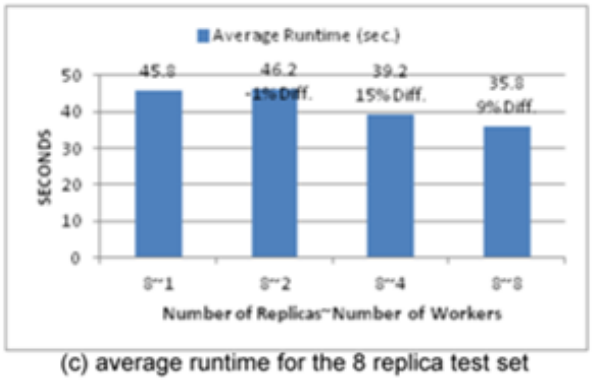
Figure 1(d) depicts the average runtime for the 16 replica test set and the percent differences between its members.

Figure 1(e) depicts the average runtime for the 32 replica test set and the percent differences between its members.

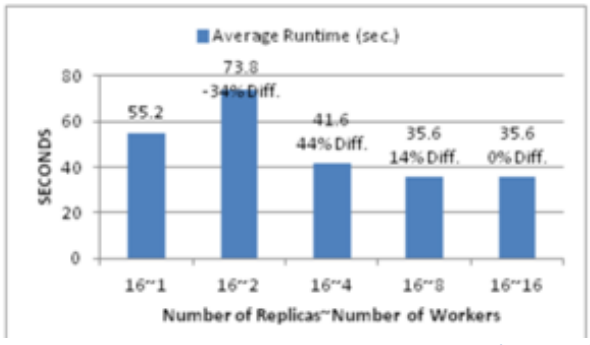


(a) average runtime for the 2 replica test set

(b) average runtime for the 4 replica test set



(c) average runtime for the 8 replica test set



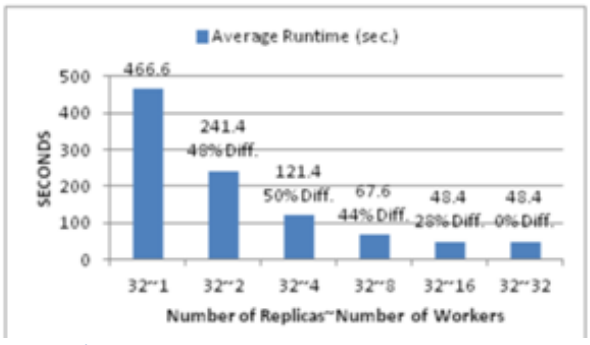


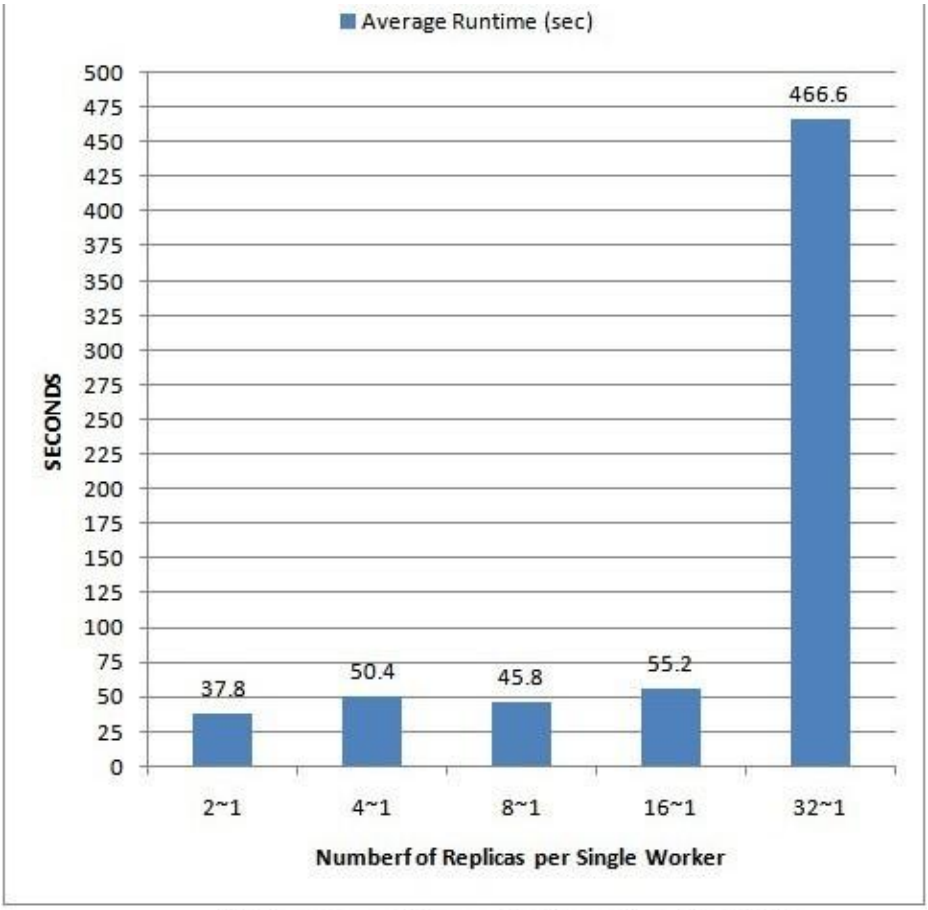
Figure 1: Average Runtimes

The data in Figures 1(a) and 1(b) suggests that as the number of workers increases relative to the number of replicas, the average runtime decreased. A similar outcome was expected of the data in Figures 1(c) and 1(d) but such was not the case. In both instances the average runtime achieved with two workers

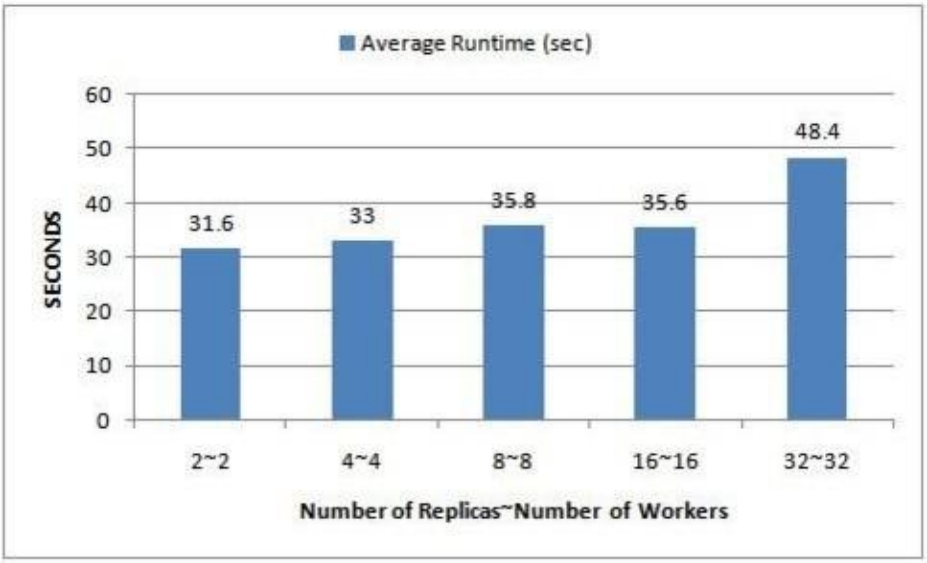
exceeded the runtime one worker. point, as the workers relative to the replicas, the runtime did expected.

The data in 1(b), and 1(c) the average reached its the one-to-worker ratio. Figures 1(d) suggests that average achieved one-to-one worker ratio

A side-by-comparison between the averages of test with a and a separate comparison between each the one-to-the number of This is Figure 2



(a) Comparison of Average Runtimes with a Single Worker



(b) Comparison of Average Runtimes with Replicas and Workers in a One-To-One Ratio

Figure 2: Comparison of Average Runtimes With a Single Worker & Comparison of Average Runtimes of Replicas in a One-To-One Ratio with Workers

average achieved with After that number of increased number of average decrease as

Figures 1(a), suggests that runtime minimum at one replica to The data in and 1(e) the minimal runtime was before the replica to is reached.

side was made runtime each replica single worker side-by-side was made replica test in one ratio with workers. depicted in

The data in Figure 2(a) suggests that with a single worker, as the number of replicas increased the average runtime also increased, there being an exception between the 4 to 1 replica test set in the 8 to 1 replica test set. It is at this point that the 8 to 1 replica test had an average runtime that was faster than that of the 4 to 1 replica test.

The data in Figure 2(b) is inconclusive. While it is a comparison of the number of replicas in a one-to-one ratio with the number of workers, it does seem to support the idea that that as the number of replicas increased the average runtime increased until replica to worker number reaches 16. Here the average runtime was actually faster than that found for the replica to worker number 8.

Conclusion

The results of this project suggests that REMD implemented on WorkQueue is scalable (i.e. the number of replicas evaluated and the number of workers can be changed up or down as needed). The data also suggests that increasing the number of workers relative to the number of replicas implemented does reduce the average runtime, although there are points in the data that counter this suggestion.

The data also suggests that for a certain number of replicas beyond 8, there is a "replica to worker" ratio that allows for the average runtime to reach its minimum prior to the one-to-one relationship.

What cannot be determined by these results is whether WorkQueue implemented REMD is better/faster than REMD in the standard parallel environment.

Further work

Given that the average runtime for the replica test set of 8~2 was faster than that for 8~1, and that the average runtime for the replica test set of 16~2 was faster than that for 16~1, further exploration of REMD on the WorkQueue framework is desirable.

Considering that both the 16 and 32 replica test sets reached their minimal average runtime prior to reaching the one-to-one replica to worker ratio, further exploration of this aspect of REMD is also desirable.

Acknowledgments

Due to the assistance of Graduate Student Zangwei Li, Graduate Student Jin Niu, Professor Alvin Allen and my mentor, Dr. Shuju Bai, I was able to complete this challenging project.

I learned far more than anticipated and I am grateful.

[1] Anthony Canino, Jesus A Izaguirre, and Douglas Thain Dinesh Rajan, "Converting A High Performance Application to an Elastic Cloud Application," in *2011 Third IEEE International Conference on Cloud Computing Technology and Science*, Notre Dame, 2011, pp. 383-390.

[2] Zachary W Ulissi, "Replica-Exchange Molecular Dynamics on Hadoop," MIT, Cambridge, Rep 18.337 Final Report, 2011.

[3] Muan Hong Ng, Steven Johnston, Stuart E Murdock, Bing Wu, Kaihsu Tai, Hans Fangohr, Paul Jeffreys, Simon Cox, Jeremy G Frey, Mark S.P Sansom and Jonathan W Essex Christopher J Woods. (2012, June) rsta.royalsocietypublishing.org. [Online]. http://rsta.royalsocietypublishing.org/content/363/1833/2017.ful

[4] Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, Douglas Thain Peter Bui, Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications, 2011, Workshop on Python for High Performance and Scientific Computing at SC11.