Course No. **: CSCI-522**

MPI-Vina: MPI based parallel implementation of Autodock Vina

Submitted
to
Dr. Laurence T. Yang

Submitted
by
Md Mokarrom Hossain
Student ID # 201304460
x2013idf@stfx.ca

Winter 2014

Problem Description

In Bio-informatics, virtual screening of small molecules using molecular docking has become an important tool in drug discovery. Typically, large scale virtual screening is time demanding task. Moreover, number of small molecules (ligand database) has been increasing day by day. Therefore, virtual screening has emerged as one of the most computational and data-intensive scientific applications in structure-based drug discovery. In order to perform virtual screening with such large scale data faster, using MPI based parallel tools on cluster computer system is a good solution.

AutoDock Vina [1] is an open-source program for doing molecular docking. Though Autodock Vina has multi-core capability, high performance and enhanced accuracy, it is a serial application. Therefore, we have parallelized Autodock Vina, resulting MPI-Vina which massively reduces the time of virtual screening by using state-of-the-art parallel computing.

MPI-Vina is an MPI based parallel tool for doing large scale virtual screening on Beowulf cluster. It is a simple straight-forward program that automates and parallelizes virtual screening using AutoDock Vina. Here molecular docking is distributed into different nodes of a computational cluster. By distributing docking procedure allows performing multiple ligands docking into multiple nodes concurrently. MPI-Vina source code is written in ANSI C using MPI standard and tested with Open MPI [10] (an open source MPI implementation) library at ACEnet [9] HPC Cluster.

In this report we will describe design, implementation and results of MPI-Vina which is an open source parallel version of Autodock Vina. We also depict the details performance analysis of MPI-Vina to demonstrate its scalability.

Virtual Screening with AutoDock Vina

In in-silico protein-ligand docking, Autodock Vina is a new program for molecular docking and virtual screening with multi-core capability, high performance and enhanced accuracy for a single computer. AutoDock Vina is the primary docking program which is used in MPI-Vina for virtual screening. It is used due to its accuracy and its speed, which is approximately two orders of magnitude faster than its predecessor, AutoDock4 [2].

In order to perform molecular docking using Autodock Vina, we need to provide a configuration *text* file, a target receptor (protein) *pdbqt* file and the desired ligand *pdbqt* file to the *vina* program. After docking, vina returns two files - one is *ligand_name.pdbqt.pdbqt* and another one is *ligand_name.pdbqt.txt* which contains the docking result. AutoDock Tools is utilized for preparing the *pdbqt* file of target protein and setting the size and the center of the grid box. The predicted binding affinity (kcal/mol), which indicates how strongly a ligand binds to the receptor, is calculated based on the scoring function used in AutoDock Vina. A more negative binding affinity indicates stronger binding.

Parallelization of Autodock Vina

In a single PC, execution time of Autodock Vina increases linearly along with the number of lgands because it processes ligand one after another. As millions of ligands are docked in virtual screening, parallelization of Autodock Vina is needed in order to reduce the overall screening time significantly. Therefore, we parallelized Autodock Vina using the Open MPI implementation of the MPI standard, resulting MPI-Vina. MPI-Vina is an open-source parallelization of AutoDock Vina which massively reduces the time of virtual screening.

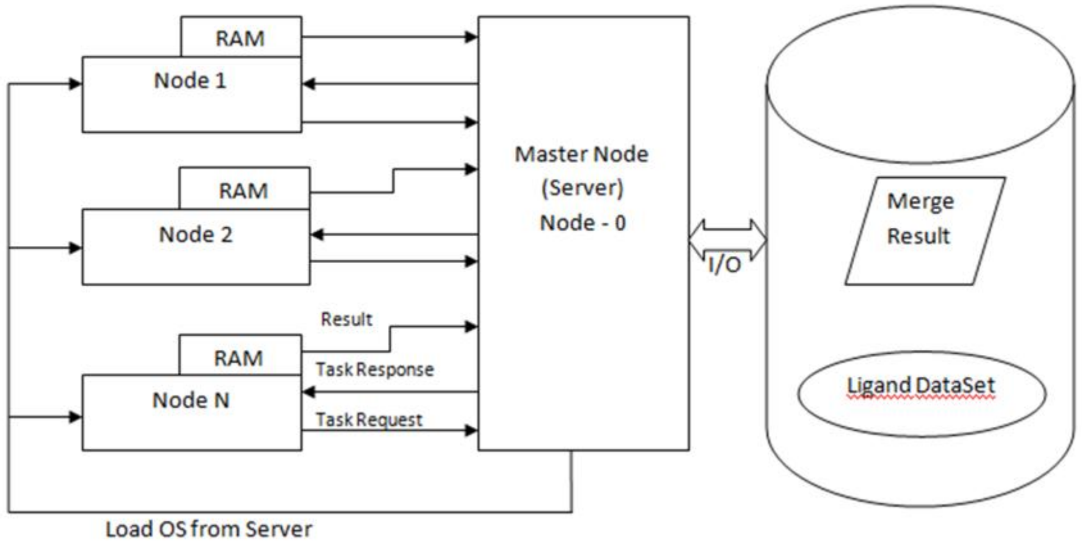


Figure-1: Architecture of MPI-Vina.

MPI-Vina Algorithm

Our MPI-Vina algorithm is designed based on one of the most universal *manager/worker* or *task parallelism* approach, in which node-0(master node) is the manager and all other nodes(slave nodes) are worker nodes. In *manager/worker*, the manager divides the whole work into several pieces and each piece is assigned to an individual "worker" processor. Here MPI-Vina manager algorithm executes a different algorithm from that of the MPI-Vina workers, but all of the MPI-Vina workers execute the same algorithm. Though MPI-Vina manager and MPI-Vina worker executes different algorithm, we combine manager and worker code into a single program which is more convenient, efficient and also supported by all implementation of MPI.

In case of *manager/worker* implementation the whole work can be evenly divided into exactly as many pieces as there are workers, but we have implemented *self-scheduling* approach which is more flexible and fault tolerance. In *self-scheduling* approach, manager keeps a pool of units of work which is larger than the number of workers and assign new work item dynamically to each worker as they complete their tasks and send their results back to the manager. So it works well despite varying the size of tasks and varying the speed of individual nodes. In MPI-Vina algorithm, we considered one ligand is sent at a time to a worker for processing. It is also possible to send more than one ligand at a time so that worker can process multiple ligands in parallel. But it does not speed up the overall screening rather slow down, because AutoDock Vina can uses almost all available resources of single CPU.

MPI-Vina Manager

Master node (typically node with rank 0) plays the role of MPI-Vina manager. In our MPI-Vina manager-worker algorithm manager and worker play two different roles. First of all, MPI-Vina manager read the ligand list file and create a list. As MPI-Vina follows *self-scheduling* approach, it creates a pool of work unit by all ligands where docking each ligand against a protein is considered as a work unit or work item. Initially MPI-Vina manager waits for a worker to make a request. When MPI-Vina manager receives a request with WORK_REQ_TAG from any worker, it assigns a new work item dynamically from its work pool to that worker. After finishing the computation, MPI-Vina manager sends a message with TERMINATE_TAG to all workers which indicate work done. The pseudo code of master is given below:

```
for  ( i = 0;  i < total_ligand;   i++ )  {
          recv (P_any,  work_req_tag,  P_source);     /* wait for a slave to make request */
          send (ligand [i],  P_source,  compute_tag);    /* send an work item to the requested
          slave */
}

/* computation has done, terminates all slaves */
for ( i = 0;  i < num_slaves;   i++)  {
          recv (P_any,  work_req_tag,  P_source);
          send (P_source,  terminate_tag);    /* send the termination tag */
}
```

MPI-Vina Worker

On the other hand, all slave nodes (that with rank > 0) play the role of MPI-Vina worker. At the beginning, MPI-Vina worker requests to the MPI-Vina manager for a work item with WORK_REQ_TAG. If it gets a new work item (ligand) for processing with COMPUTE_TAG, performs molecular docking operation through Autodock Vina program. Autodock Vina performs molecular docking on provided ligand against the target receptor using configuration file. When docking is done worker sends the result to MPI-Vina manager and request for another work item. When it gets another new work item with COMPUTE_TAG from manager, it repeats the same procedure and going on until manager send it an empty message with TERMINATE_TAG. Whenever any worker gets TERMINATE_TAG, it terminates right away. All sends/receives are performed via MPI_Send () / MPI_Recv () routine.

```
send (P_master,  work_req_tag);    /* request for an work item */
recv (&ligand_name,  P_master,  source_tag);
/* performs docking and request for another work item.
Repeat until manager send terminate_tag  */
while (  source_tag  ==  compute_tag )  {
          Autodock_Vina  ( &ligand_name );   /* ask Autodock Vina to perform docking */
          send  (P_master,  work_req_tag);    /* request for another work item */
          recv  (&ligand_name,  P_master,  source_tag);
}
```

Experiments and Results

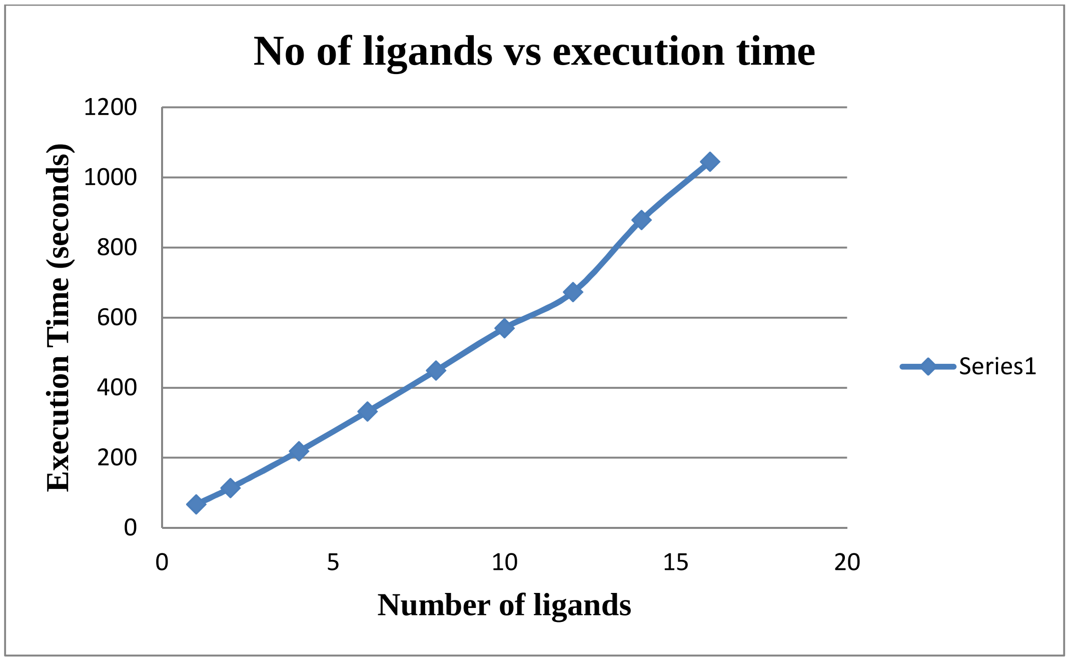
Autodock Vina Performance

Before assessing the MPI-Vina performance, let's analyze the performance of Autodock Vina, because it is the base program of MPI-Vina. The *table-1* shows the Autodock Vina execution time for different number of ligands.

No. of Ligand	Execution Time (seconds)	No. of Ligand	Execution Time (seconds)
1	67	10	570
2	114	12	673
4	219	14	879
6	332	16	1045
8	449	18	1404

Table-1: Autodock Vina execution time.

In order to ease the observation, we can draw a line graph using data from *table-1*. From the graph below, it is observed that execution time increases almost linearly with the number of ligands.

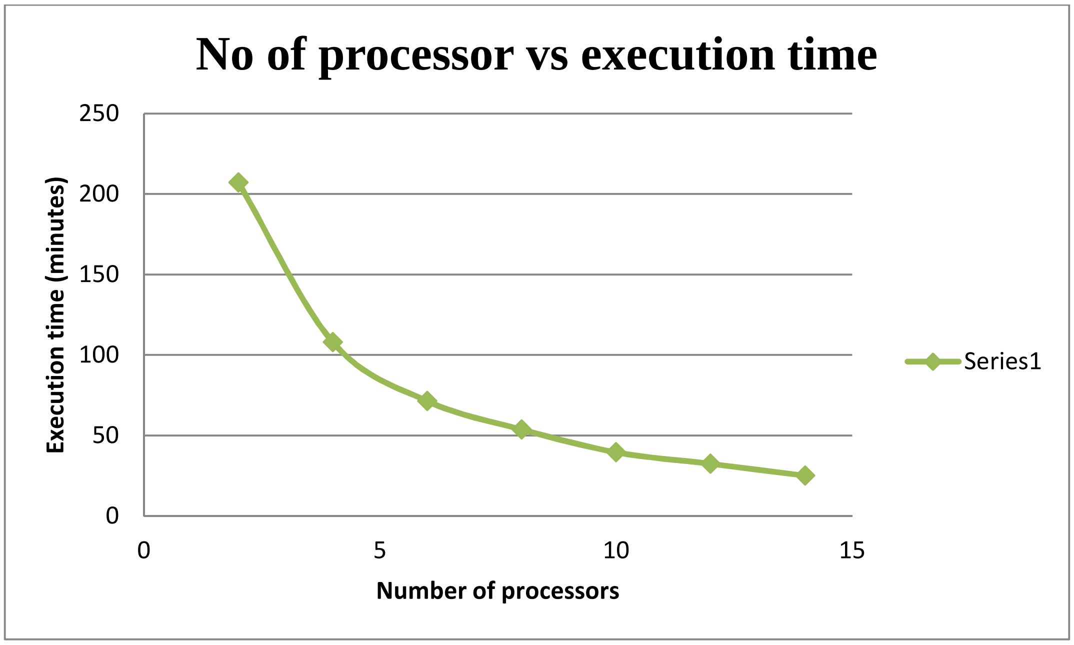


MPI-Vina Performance

In order to ease the experiment we have built couple of test scripts in *bash*. We have executed our test script several times and noted the execution time. Here is the one observed data.

No of Ligands	No of Processors	Execution Time (minutes)
420	2	207.296
	4	107.998
	6	71.386
	8	53.725
	10	39.578
	12	32.416
	14	25.068

In order visualize the relationship between execution time and number of processor easily we can draw a line graph.



From the above line graph we observed that in the beginning, execution time reduces significantly according to the number of processor increase. Then it reduces moderately in middle part and in the later part of line graph execution time reduces slightly according to the processor number increases. The reason of such pattern is increasing the communication overhead. As long as, we add more processor in the communication group, communication cost also increases. Based on network infrastructure, after a certain number of processors bottleneck problem may occur. Network I/O is big concern in that case.

Fault Tolerance

In case of high-performance computing (HPC) applications, fault tolerance has been identified as one of the most serious issues due to usage of large-scale computing resources for long time. Our MPI-Vina algorithm also gives waiver from the failure of a node tactfully. MPI-Vina follows self-scheduling approach. In this approach, if one or more nodes fail, computation will continue on the remaining nodes.

On the other hand, if manager node (rank with 0) fails during the processing, it also possible to resume the screening process from immediate previous state of failure. MPI-Vina always segregate finished task from unfinished task so that it can resume easily. Since MPI-Vina algorithm follows self-scheduling approach, it does not affect the performance when physical parameter of individual node varies or even individual ligand size varies.

Conclusion

In this work we presented MPI-Vina a simple and straightforward program that performs parallel virtual screening of compound databases against protein receptors, using AutoDock Vina as docking program. It is a new open source and MPI-based parallelization of Autodock Vina. The design of MPI-Vina parallel algorithm was thought-out so that the cluster remains stable even when using computers with different speeds and characteristics. We described the detailed design of MPI-Vina and presented experimental evaluation on different experimental as well as real cluster systems. Our results show an order of magnitude improvement in performance with MPI-Vina in some cases. The goal of MPI-Vina is to reduce the overall time of screening ligand data set.

Our experimental results on ACEnet cluster demonstrate that our self-scheduling approach nicely coordinates dynamic computation load-balancing and allows large-scale screening on general parallel computers. Although the result was supposed to be linear in ideal, but for the file system I/O and network I/O interrupt our system isn’t showing linear property. Though the delay caused by the physical reasons made the system not reliable all the time, we have found our desired output which was shown in the result section. Our findings indicate that MPI-Vina scales well to at least one hundred nodes.

Acknowledgement

Computational facilities are provided by ACEnet, the regional high performance computing consortium for universities in Atlantic Canada. ACEnet is funded by the Canada Foundation for Innovation (CFI), the Atlantic Canada Opportunities Agency (ACOA), and the provinces of Newfoundland and Labrador, Nova Scotia, and New Brunswick.

Attachment

Annotated source code of the program has been enclosed.

References

[1] O. Trott, A. J. Olson, “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading”, *Journal of Computational Chemistry* vol. 31, pp. 455-46, 2010.

[2] Morris GM, et al, “Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function”, *Journal of Computational Chemistry* vol. 19, pp.1639-1662, 1998.

[3] W. Feng, M. Warren, and E. Weigle, “The bladed beowulf: A cost-effective alternative to traditional beowulfs”, *In Proceedings of IEEE Cluster* 2002.

[4] M. Warren, E. Weigle, and W. Feng, “High-density computing: A 240-node Beowulf in one cubic meter”, *In Proceedings of SC* 2002.

[5] Andrew P Norgan *et all*, “Multilevel Parallelization of AutoDock 4.2”, *Journal of Cheminformatics*, vol. 3, pp. 12, 2011.

[6] Development and Applications of Virtual Screening System; Kazuto YAMAZAKI, Masaharu KANAOKA; SUMITOMO KAGAKU 2005- I.

[7] SUMITOMO KAGAKU 2005- I; Structure-based virtual screening: an overview; DDT Vol. 7, No. 20 October 2002.

[8] Jorgensen WL (March 2004), “The many roles of computation in drug discovery”, *Science* **303** (5665): 1813–8, PMID 15031495.

[9] ACEnet Cluster, http://www.ace-net.ca/wiki/ACEnet.

[10] Open MPI, http://www.open-mpi.org/.