HPC BLAST: Best Practices

The Application Acceleration Center of Excellence (AACE) and
Intel Parallel Computing Center at the
Joint Institute for Computational Sciences
University of Tennessee

May 7, 2019

1 Introduction

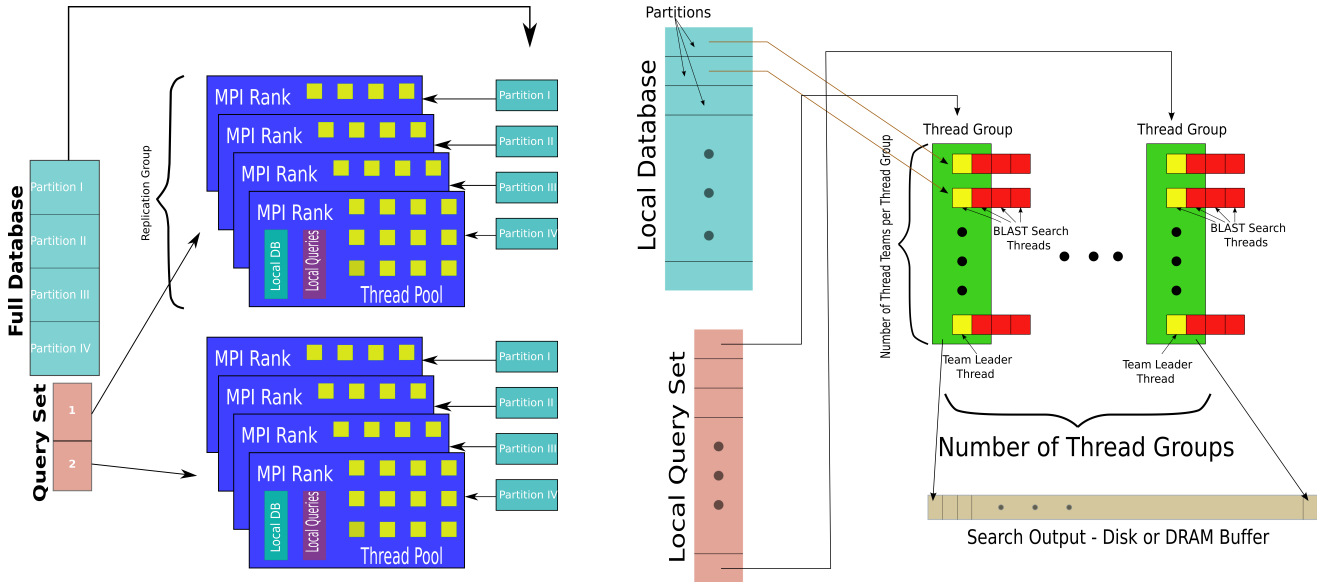
This document is intended to outline a series of best practices, or rules of thumb, for using HPC-BLAST to perform large scale sequence alignment searches on current and emerging architectures. The collected best practices are derived from scaling studies and other data gathering experiments and should provide a reasonable level of performance for many cases a user might encounter. However, please note that the given practices are based on results from execution on particular architectures and specific data sets and may not reflect the best parameters or guidelines for other compute resources or data sets.

2 Best Practices

As explained in the HPC-BLAST User Manual, there are a number of different parameters that can be adjusted to specify the manner in which a parallel HPC-BLAST job can execute: the number of MPI ranks, number of replication groups, number of ranks that constitute a replication group, number of thread groups, number of database partitions (team leaders) per thread group, and number of search threads used inside the BLAST engine.

The different components of HPC-BLAST are summarized here so that the terminology used below is clear. HPC-BLAST employs a hierarchical approach for assigning search tasks. At the highest level, MPI ranks are utilized to distribute the pieces of a database partition among compute resources. A replication group is a collection of MPI ranks that aggregately posses the entire database. Each MPI rank in a replication group has the same query input since the database is distributed. Multiple replication groups can be used so that the query input can be distributed between different replication groups. It is also possible to use replication groups with only a single rank; i.e. the database is not distributed, though it may still be partitioned. Within each MPI rank, HPC-BLAST employs a novel threading model to distribute tasks in a similar manner as the MPI layer. A thread group plays the role of the replication group. Here, a thread group is a collection of threads that share the same query subset but have different pieces of the local database. Threads in a thread group are called team leaders since they can instantiate searcher threads. The searcher threads are the threads that the NCBI BLAST engine provides. The MPI and thread levels can be seen visually in Figure 1.

It is natural to ask how many MPI ranks should be launched and how many threads should each rank use. First, let us address the number of threads to use on a single compute node. Scaling studies have demonstrated that it is best to use the number of threads equal to the number of logical cores. For dual socket systems, this can be around 100 threads, depending on the particular processor. For now, suppose we are using only a single MPI rank with HPC-BLAST per target; we will address multiple ranks per target below. Given this constraint there are some rules of thumb regarding the allocation of the threads to the different categories: thread groups, team leaders, and search threads. One caveat of using database partitioning inside the BLAST process (the team leader threads) is that it does not guarantee 100% NCBI compliant results. Since moving to the NCBI BLAST toolkit version 2.7.1,



(a) The MPI layer.

(b) The thread layer.

Figure 1: The HPC-BLAST hierarchy.

scaling studies have shown that allocating threads to the NCBI search engine (using the `-num_threads` command line option) provides the best performance.

If it is decides to use thread groups, make sure that there is enough input to justify the number of thread groups. At a minimum, there should be enough residues (or letters) so that each thread group has one batch's worth of queries. Here, a batch refers to the concatenated query that NCBI-BLAST uses; i.e. multiple queries from the input are combined and searched through the database as one entity. For *blastp* searches, the batch is 10,000 residues. For *blastn*, the NCBI-BLAST engine uses an adaptive batch size, but 100,000 residues represents a good minimum number to assign each thread group. If there is not enough work to give all thread groups at least one batch worth of queries, then it is probably best to consider a database partition at the MPI level and distribute the database subsets to the MPI ranks inside a single replication group.

Scaling experiments have also demonstrated that *blastp* searches can see increased performance by using multiple MPI ranks as opposed to a single rank with as many threads as there are logical cores. A good rule of thumb is to use 4, 6, or 8 MPI ranks per compute node. Further, the best performance was observed when 2 replication groups were used. As an example, consider the following scenario. We decide to use 6 MPI ranks and 2 replication groups. Thus, we would partition the database into 3 subsets (3 ranks per replication group). Each HPC-BLAST MPI rank would then have at most (#logical cores)/(#MPI ranks per node) to use towards search threads, or thread groups.

The number of MPI ranks that should be launched during a parallel search is influenced by both the size of the query input and the size of the database. The size of the database will contribute to the number of ranks in a replication group. Good performance has been observed with large databases, such as **nr**, when the size of the formatted sequence files (the *.psq* or *.nsq* files) are around 2 GB. The number of MPI ranks per replication group is then determined by how many subsets are generated in the database partition by the *makeblastdb* tool (see the HPC-BLAST User Manual for instruction). As for the number of replication groups to employ, the above advice should be followed in that each replication group should have enough queries to give each thread group at least one full batch worth of residues; though, this is a lower bound and a more reasonable middle ground might be to aim for

ten batches worth per replication group.

2.1 Summary

- Use all available logical cores on the desired architecture before adding more compute resources.
- Avoid using team leaders unless there is a good reason for using them.
- Use between 4 and 8 ranks per compute note with two replication groups.
- With HPC-BLAST version 1.1.0, it is generally better to allocate threads to searcher threads rather than to have multiple thread groups.
- The database should be partitioned such that sequence files are all around 2 GB in size; this also determines how many MPI ranks will form a replication group.
- For large runs with multiple replication groups, use a number of replication groups so that each group has around 10 batches worth of queries, if resources allow.