The DNA sequences are huge in size and the databases are growing at an exponential rate. For example, the human genome in raw format ranges from 2 to 30 Terabytes. The main reason for this is the invention of new species and increasing number of DNA profiles. The growth of the DNA affects the storage as well as bandwidth when these sequences need to be transferred. Applications such as DNA profiling, Real time DNA crime investigation require access to the DNA sequences in real time. The inherent property of DNA is that it contains many repeats which makes it highly compressible. However, the applications mentioned not only require good compression ratio but also needs faster compression. Multicores and GPUs can be used to perform the compression quickly. In this paper, we propose a new algorithm with a focus on the throughput along with the compression ratio. The algorithm scales well on GPUs and achieves a speedup of 11 on multi-cores and up to 23 on GPUs when run on M2070 Tesla card and up to 57 on K20 Kepler GPUs. We also extended this algorithm such that it adapts to the input sequence depending on the number of consecutive repeats and accordingly chooses the right algorithm which leads to a better compression.

**Abstract:**

The DNA sequences are huge in size and the databases are growing at an exponential rate. For example, the human genome in raw format ranges from 2 to 30 Terabytes. The main reason for this is the invention of new species and increasing number of DNA profiles. The growth of the DNA affects the storage as well as bandwidth when these sequences need to be transferred. Applications such as DNA profiling, Real time DNA crime investigation require access to the DNA sequences in real time. The inherent property of DNA is that it contains many repeats which makes it highly compressible. However, the applications mentioned not only require good compression ratio but also needs faster compression. Multicores and GPUs can be used to perform the compression quickly. In this paper, we propose a new algorithm with a focus on the throughput along with the compression ratio. The algorithm scales well on GPUs and achieves a speedup of 11 on multi-cores and up to 23 on GPUs when run on M2070 Tesla card and up to 57 on K20 Kepler GPUs. We also extended this algorithm such that it adapts to the input sequence depending on the number of consecutive repeats and accordingly chooses the right algorithm which leads to a better compression.

GPU Implementations:

The steps involved in GPU implementation are as follows:

- The resultant output array from the phase 1 is copied into the global memory of GPU from the CPU host memory.
- The kernel is launched with the number of threads and the blocks varying according to the size of the given input sequence. For a small input sequence, we use threads starting from 50 and as the size increases the no. of threads launched increases up to 5000.
- Each thread finds the repetitions and stores the result in a buffer in the global memory.
- After all the threads finish their job, this buffer is copied from global memory to host memory.

Future Work:

We are also extending the proposed algorithm to RNA sequences for compression. It also helps to calculate phylogeny. Also, the GPU implementation is used to solve the multiple-sequence alignment problem and the work is in progress in this direction. This algorithm also helps in reducing the time in searching databases especially when the sequences are really long. A half-byte can be used instead of a full code-byte in order to save the space consumed by the extra code-byte when there are no repetitions.

**Results:**

- [Graph 1: Timing And Compression Ratio Of Algorithm]
- [Graph 2: Adaptive GenCodex : A Novel Algorithm for Compressing DNA Sequences on Multi-cores and Graphical Processing Units.]
- [Graph 3: Conclusions: A new compression algorithm is proposed to compress the DNA sequences. The main focus was on the throughput along with the compression ratio. As the number of consecutive repeats increases, the algorithm achieves the best compression. If the fragments are repeating only twice or there are no repetitions then the algorithm may not perform better. This lead us to develop an adaptive DNA compression algorithm which takes samples from the given input sequence and chooses the correct algorithm depending on the number of consecutive repeats. The compression ratio remains same for both the serial and parallel versions. We noticed a very good improvement in the throughput when the algorithm was implemented on multi-cores and GPUs. We observed a speedup of 11 on multi-cores and 23 on GPUs when run on Lonestar which has NVIDIA M2070 card and there is a significant improvement when the parallel version is run on Stampede which has Kepler K20 GPUs with a speedup of up to 57 being achieved. Experiments showed us a good scalability on GPUs for the standard data-sets. The results show that our method achieves a good compression ratio along with better throughput compared to other existing methods. We are modifying the proposed algorithm in such a way that it can utilize the property of dynamic parallelism of Kepler K20 GPUs. This results in achieving high throughput as GPU threads themselves spawn new threads without communicating with the CPU. Also, we are working further such that different CPU cores simultaneously utilize the CUDA cores on a single Kepler GPU which increases GPU utilization and cuts down the CPU idle times. This really makes our algorithm robust achieving a very high throughput.

**Contact Information:**

Ajith Padyana: ajithpadyana@sssihl.edu.in