Scaling behavior of short read sequence aligners

This blog explores the scaling behaviors of the most popular three short read sequence aligners; BWA-mem, Bowtie and Bowtie2. Aligning process is the beginning of all Next Generation Sequencing (NGS) data analysis and typically the most time-consuming step in any NGS data analysis pipeline.

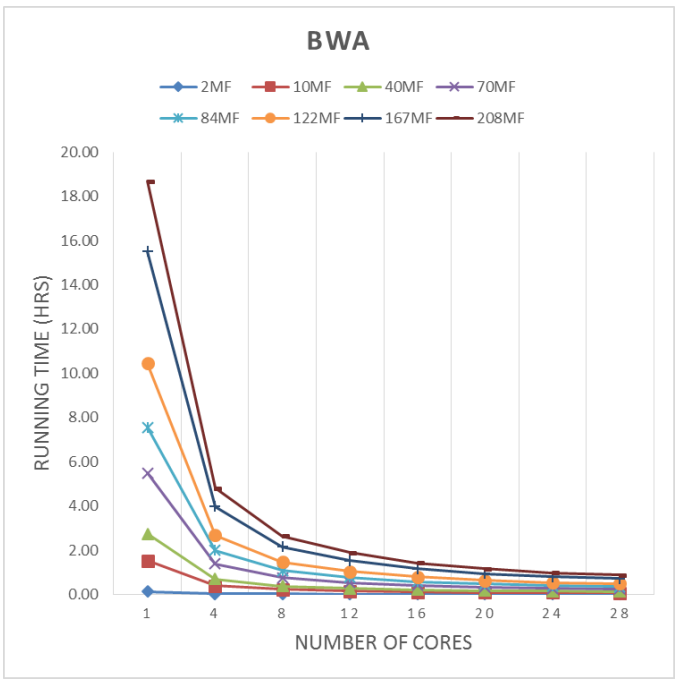
It is clear that using more cores for the aligning process will help to speed up the process, but as you already know parallelization comes with cost. Higher complexity, perhaps order of magnitude, due to multiple instruction streams and data flowing between the instruction streams can easily overshadow the speed gain by parallelizing the process. Hence, it is not wise to use the entire cores in a compute node to speed up blindly especially when the overall throughput is more important. Identifying a sweet spot for the optimal number of cores for each aligning process and maximizing the overall throughput at the same time is not an easy task in a complex workflow.

Table 1 Server configuration and software

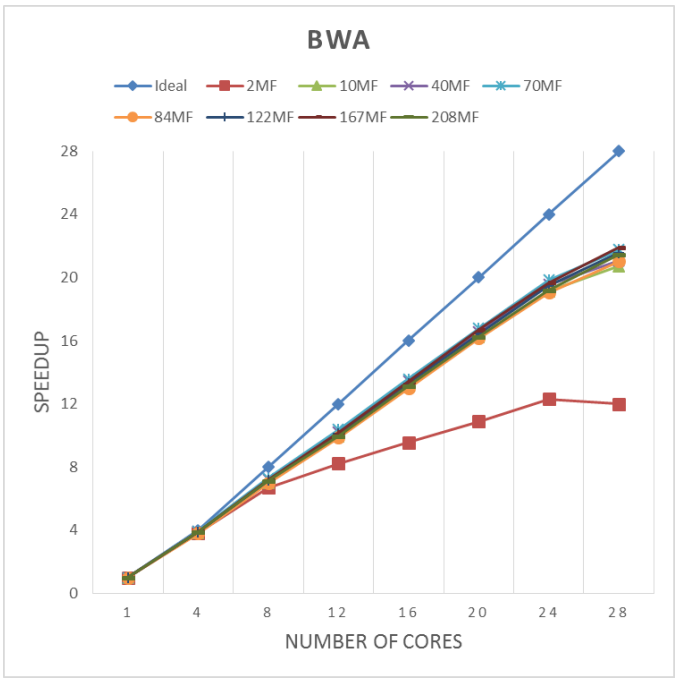
Component	Detail
Server	PowerEdge FC430 in FX2 chassis - 2 Sockets
Processor	Intel® Xeon® Dual E5-2695 v3 - 14 cores, total 28 physical cores
Memory	128GB - 8x 16GB RDIMM, 2133 MT/s, Dual Rank, x4 Data Width
Storage	480TB IEEL (Lustre)
Interconnect	InfiniBand FDR
OS	Red Hat Enterprise 6.6
Cluster Management tool	Bright Cluster Manager 7.1
Short Sequence Aligner	BWA 0.7.2-r1039, Bowtie 1.1.2, Bowtie 2.2.6

Table 1 shows the summary of the system for the test here. Hyperthreading was not enabled for the test although it helps to improve the overall performance.

In Figure 1, BWA running times were measured for each different sizes of paired end read sequencing data. The size of sequence read data is represented as million fragments (MF) here. For example, 2MF means that the sequence data consist of two fastq files containing two million sequence reads in each file. One read is in a fastq file, and the



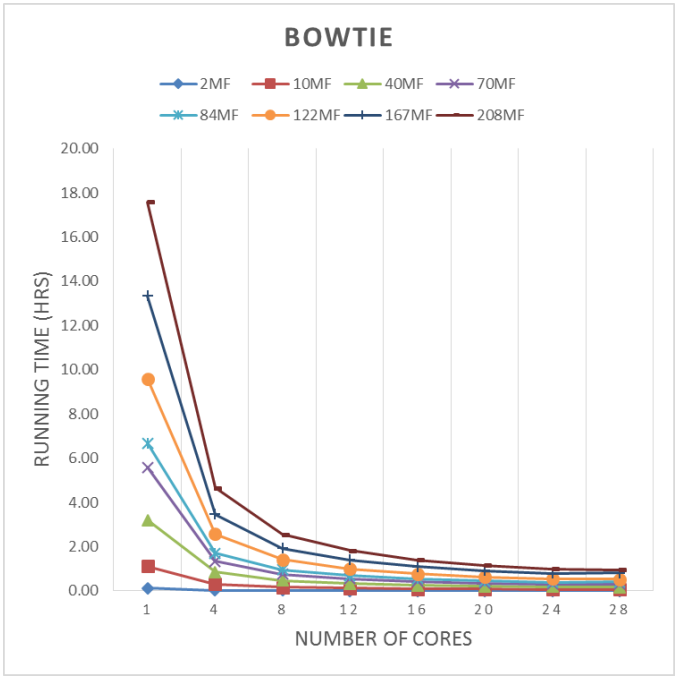
(a)



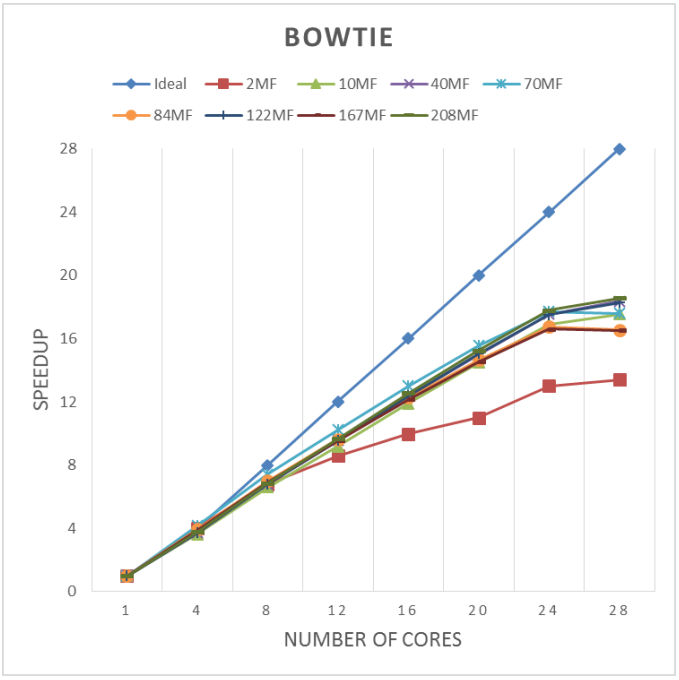
(b)

Figure 1 BWA running time and speed up over various number of cores: (a) BWA run time over the various sizes of inputs (b) Calculated speedup gained by the different number of cores

corresponding paired read is in the other fastq file. A sweet spot for BWA-mem is from 4 to 16 cores roughly depending on the sequence read file size. Typically, the sequence read size is larger than 10 million and hence, 2MF and 10MF results are not realistic. However, larger sequence read data follow the behavior of the smaller input sizes as well.



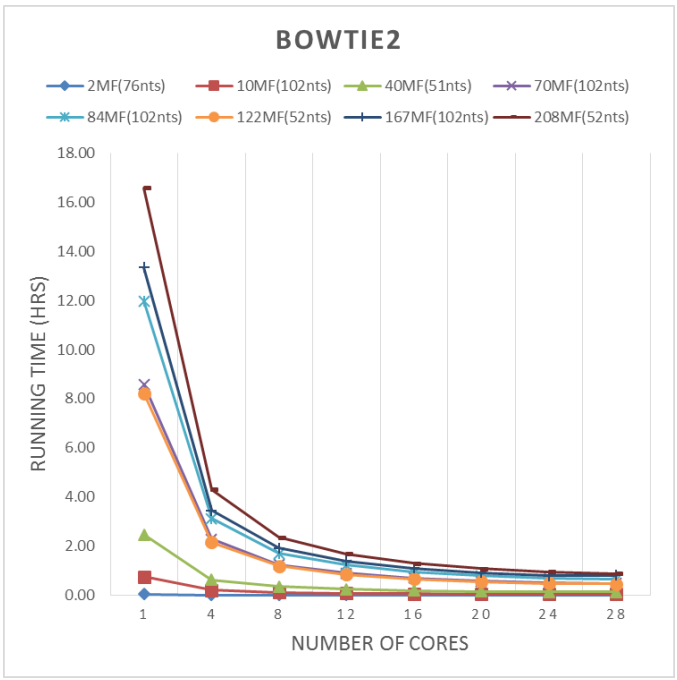
(a)



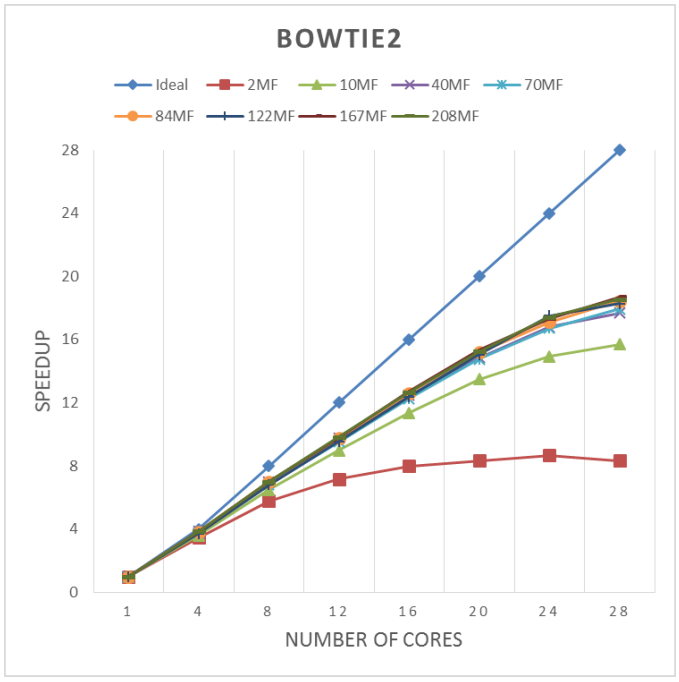
(b)

Figure 2 Bowtie running time and speed up over various number of cores: (a) Bowtie run time over the various sizes of inputs (b) Calculated speedup gained by the different number of cores

Bowtie results in Figure 2 shows that the similar sweet spot comparing to BWA-mem results. However, the running time is slightly faster than BWA-mem's. It is notable that Bowtie and Bowtie2 are sensitive to the read lengths while BWA-mem shows more consistent scaling behavior regardless of the sequence read lengths.



(a)



(b)

Figure 3 Bowtie2 running time and speed up over various number of cores: (a) Bowtie2 run time over the various sizes of inputs (b) Calculated speedup gained by the different number of cores

Although Bowtie2 is even faster than Bowtie, it is more sensitive to the length of the sequence reads as shown in Figure 3. Actually, the total number of nucleotides could be a better matrix to estimate the running time for a given sequence read data.

In practice, there are more factors to consider to maximize the overall throughput. One of critical factors is the bandwidth of existing storages in order to utilize the sweet spot instead of using the entire cores in a compute node. For example, if we decided to use 14 cores for each aligning process instead of using 28 cores, this will double up the number of samples processed simultaneously. However, the twice number of processes will fill up the limited storage bandwidth, and overly used storages will slow down significantly the entire processes running simultaneously.

Also, these aligning processes are not used alone typically in a pipeline. It is frequently tied up with a file conversion and a sorting process since subsequent analysis tools requiring these alignment results sorted in either chromosome coordinates or the name of the reads. The most popular approach is to use ‘pipe and redirection’ to save time to write multiple output files. However, this practice makes the optimization harder since it generally requires more computational resources. More detailed optimization for NGS pipelines in this aspect will be discussed in the next blogs.