CS267 Assignment 0  
Genome Assembly on Campus Grids  

Angie Zhu  
January 31, 2014  

1 Bio  

I am a fourth year Ph.D. student in Statistics. My research interests are statistical genomics and bioinformatics. I am proficient in R and familiar with Python, C, and Java. I would like to learn the basics and implementation of parallel computing, which can be utilized in my research projects.

2 Genome Assembly on Campus Grids  

The ultimate physical map of a genome is its sequence. The recently developed next-generation sequencing technology has enabled scientists to obtain more genomic data faster and cheaper than before. Whole-genome shotgun sequencing has rapidly expanded our knowledge of genomes. But current sequencing technologies only allow accurate reading of no more than 500 to 800 base pairs of contiguous DNA sequence [1], which is very small even comparing to the length of the genes. For instance, human genes have an average length of 10,000 to 15,000 base pairs with enormous variation [4]. Then the sequence of an entire genome must be assembled from the collection of short reads. This process is called DNA sequence assembly, which is computationally expensive.

The general assembly pipeline as summarized by Deonier et al. [1] is implemented by various sequence assemblers, even though their levels of complexity vary:

1. Screening out vector sequences or chimeric reads
2. Trimming off unreliable base calls from each read
3. Computing overlaps between pairs of reads, using the highest-quality portions of each read
4. Screening out doubtful overlaps
5. Constructing contigs
6. Producing a consensus sequence by multiple sequence alignment with reliability scores at each position
Figure 1: A modular assembler

Moretti et al. [2] divided the assembly process into three steps: candidate selection, alignment, and consensus. Their proposed modular assembler paralleled the steps of candidate selection and alignment as shown in Figure 1. *Anopheles gambiae* and *Sorghum bicolor* were assembled using both conventional sequential assembler, Celera [3], and the proposed modular assembler, where the latter showed major improvement in term of computing time as shown in Table 2. Each data set was run multiple times. Their largest run achieved a 927x speedup with 71.3 percent efficiency.

This assembler was implemented on campus grid using Work Queue [6], a general purpose master-worker system. Their campus grid was managed by Condor [5]. The assembler could be used on high performance clusters as well. The system used distributed memory. For the memory intensive candidate selection step, the parallel algorithm was designed to minimize data transfer.

Moretti et al. [2] tested the modular assembler with varying number of workers, from 16 to 512. Most jobs were run on modes in the University of Notre Dame Condor pool and machines at Purdue University and the University of Wisconsin were used when needed. This application is certainly scalable. But the waiting time for network transfers may be long.
### References


---

**Table 1: Summary of Results**

* indicates time estimated from campus grid run

<table>
<thead>
<tr>
<th>Assembly Framework</th>
<th>Alignment Algorithm</th>
<th>System Size</th>
<th>Candidate Selection</th>
<th>Alignment</th>
<th>Consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeron</td>
<td>Celeron</td>
<td>4 cores</td>
<td>4 hrs, 20 min</td>
<td>3 hrs, 11 min</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Complete</td>
<td>1 core</td>
<td>* 1 hr, 35 min</td>
<td>* 12 days, 4 hrs</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Banded</td>
<td>1 core</td>
<td>* 1 hr, 35 min</td>
<td>* 7 hrs, 9 min</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Complete</td>
<td>campus grid</td>
<td>5 min</td>
<td>45 min</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Banded</td>
<td>campus grid</td>
<td>5 min</td>
<td>11 min</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 2: Summary of Results**

<table>
<thead>
<tr>
<th>Assembly Framework</th>
<th>Alignment Algorithm</th>
<th>System Size</th>
<th>Candidate Selection</th>
<th>Alignment</th>
<th>Consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeron</td>
<td>Celeron</td>
<td>4 cores</td>
<td>crashed after 9 days</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Complete</td>
<td>1 core</td>
<td>* 14 hr, 39 min</td>
<td>* 50 days, 15 hrs</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Banded</td>
<td>1 core</td>
<td>* 14 hr, 39 min</td>
<td>* 46 hrs, 17 min</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Complete</td>
<td>campus grid</td>
<td>34 min</td>
<td>2 hrs, 40 min</td>
<td></td>
</tr>
<tr>
<td>Modular</td>
<td>Banded</td>
<td>campus grid</td>
<td>34 min</td>
<td>43 min</td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 3: Summary of Results**

* indicates time estimated from campus grid run