Performance Optimization for the Trinity RNA-Seq Assembler

9th Parallel Tools Workshop, Dresden, Germany

Michael Wagner, Ben Fulton, and Robert Henschel
michael.wagner@zih.tu-dresden.de
Outline

- Background: de novo transcriptome assembly
- Identification of optimization targets
- Performance optimizations
- Tool challenges and restrictions
- Conclusion
Introduction – Transcript Reconstruction

- RNA-seq reads

- Align RNA-seq to reference genome
  - Genome

- De novo transcript assembly
  - Genome

- Reconstruct transcripts
  - Genome
Introduction – Trinity

• Framework for de novo reconstruction of transcriptomes from RNA-seq reads

• Developed by the Broad Institute at MIT with help of Indiana University and TU Dresden

• Freely available as Open Source
  https://github.com/trinityrnaseq/trinityrnaseq/releases

• Since May 2011 about 1000 software downloads/month
Introduction – Trinity Pipeline

RNA-seq reads → Linear contigs → de-Bruijn graphs → Transcripts

Nice introductive video: http://www.broadinstitute.org/videos/trinity-how-it-works
Identification of Optimization Targets

- Trinity is a pipeline of 27 individual components (C++, Java, Python, Perl)
- "Typical" performance tools fail to provide overview
- Collectl 5 second sampling of CPU, Memory and IO utilization

http://collectl.sourceforge.net/

Schizosaccharomyces pombe (50M base pairs)
Identification of Optimization Targets

Insufficient CPU utilization
Repeated fork and join of threads

Runtime (hours)

Core Utilization

RAM usage GB

00:00 00:15 00:30 00:45 01:00 01:15 01:30 01:45 02:00 02:15 02:30

Jellyfish
Inchworm
Bowtie
Samtools
Scaffold_iworm_contigs
ReadToTranscripts
GraphFromFasta
Sort
Butterfly
Identification of Optimization Targets

- Collectl CPU utilization hides unbalanced parallel behavior (busy waiting)
- Analysis of parallel scaling with collectl timings of components

<table>
<thead>
<tr>
<th>Number of Cores</th>
<th>Parallel Speedup</th>
<th>Component Runtime Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Trinity</td>
<td>1.00</td>
<td>2.07</td>
</tr>
<tr>
<td>Jellyfish</td>
<td>1.00</td>
<td>1.84</td>
</tr>
<tr>
<td>Inchworm</td>
<td>1.00</td>
<td>1.07</td>
</tr>
<tr>
<td>Bowtie</td>
<td>1.00</td>
<td>1.63</td>
</tr>
<tr>
<td>Samtools sort</td>
<td>1.00</td>
<td>1.64</td>
</tr>
<tr>
<td>Samtools view</td>
<td>1.00</td>
<td>1.44</td>
</tr>
<tr>
<td>Scaffold_iworm_contigs</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>GraphFromFasta</td>
<td>1.00</td>
<td>1.36</td>
</tr>
<tr>
<td>ReadsToTranscripts</td>
<td>1.00</td>
<td>2.31</td>
</tr>
<tr>
<td>Sort</td>
<td>1.00</td>
<td>1.13</td>
</tr>
<tr>
<td>Butterfly</td>
<td>1.00</td>
<td>1.40</td>
</tr>
</tbody>
</table>
Optimizations – A Short Overview

Schizosaccharomyces pombe (50M base pairs)

- **Inchworm**: Reimplemented using MPI [1]
- **Samtools/Scaffold_iworm_contigs**: No binary-to-text conversion (10 min.)
- **ReadsToTranscripts**: Increased number of reads per parallel region (6 min.)
- **Sort**: Use parallel sort and enough memory (5 min.)
- **Butterfly**: Benefits from other optimizations (1 min.)

Analysis of GraphFromFasta

- 100% CPU utilization but poor scaling $\rightarrow$ speedup of 2.3 for 16 cores
- Assumption: Inefficient parallel patterns (e.g. critical section)
- Detailed performance analysis with Score-P and Vampir

Score-P + Compiler instrumentation = $\times$ Massive application slow down and large data sizes

Score-P + OpenMP only = $?$ Trace did not show OpenMP faults

Score-P + Manual instrumentation = $\checkmark$ 

Optimization of the Trinity RNA-Seq Assembler

Center for Information Services & High Performance Computing

9/2/2015
Analysis of GraphFromFasta

Test data set (30k base pairs). Note: Load imbalances only occurred in this test data set. Regular data sets did not show load imbalances.

Optimization of the Trinity RNA-Seq Assembler
Optimization of GraphFromFasta

Drosophila melanogaster (Shortened to 4M base pairs)
Optimization Results

- Runtime reduction from 2:20h to 1:48h (22% reduction)
- Similar results for Drosophila melanogaster (fruit fly) and Mus musculus (mouse)
Tool Challenges and Restrictions

- Trinity includes components written in nonstandard languages
  - “Typical” performance tools are not applicable
  - Collectl was very helpful to get general overview
- In Trinity’s Butterfly stage small applications are launched millions of times
  - “Typical” one-result-per-binary performance tools are not applicable
  - Collectl’s five second interval biases timing of components
- Busy-waiting is not detectable in collectl’s CPU chart
- Tracing can result in massive data collection and application slow down
  - Manual instrumentation limited data volumes and slow down
- Function-level recording may provide too little detail
  - Instruction-level instrumentation helped identifying the problem
Conclusions

- Performance optimization is infeasible without appropriate tools
- Tools helped to identify and analyze critical performance issues
- Tool usage is not always easy
- The further you get away from the “standard” MPI application …
  … the less tools support your application
  … the more difficult it gets to use the tools