High-Performance De Novo RNA-Transcript Assembly Leveraging Distributed Memory and Massive Parallelization

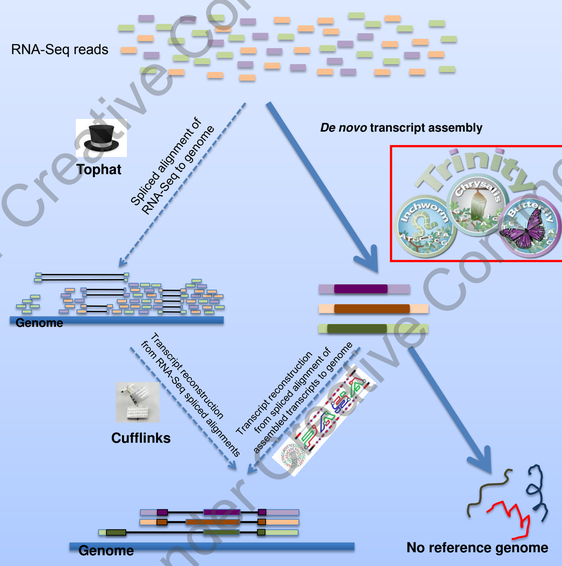
Pierre Carrier, Bill Long, *Cray, Inc.* **Carlos P. Sosa**, *Cray Inc. and BICB, University of Minnesota Rochester ;* **Thomas William** *TU-Dresden;* **Brian Haas, Timothy Tickle** *Broad Institute*

Abstract

Exemplifying collaborative software development between industry and academia to tackle computational challenges in manipulating large volumes of next-gen sequence data, leveraging advances in algorithm development and compute hardware, we describe our efforts to optimize the performance of the Trinity RNA-Seq de novo assembly software. Three versions of Trinity's Inchworm computationally intensive part (one that is based on the original OpenMP version, and two new versions that are based on MPI and on Fortran2008). New results of Inchworm's parallel performance for various real-life problems (e.g., mouse, schizosaccharomyces pombe) are presented, as well as a detailed discussion of the MPI and PGAS computations scheme for Inchworm.

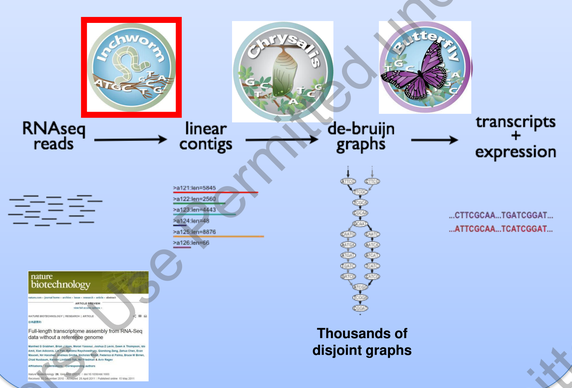
RNA-Seq

Contemporary strategies for transcript reconstruction from RNA-Seq



Trinity de novo RNA-Seq Assembly Pipeline

Three main sections define the TrinityRNASeq software.
We focus our work on the Inchworm part:



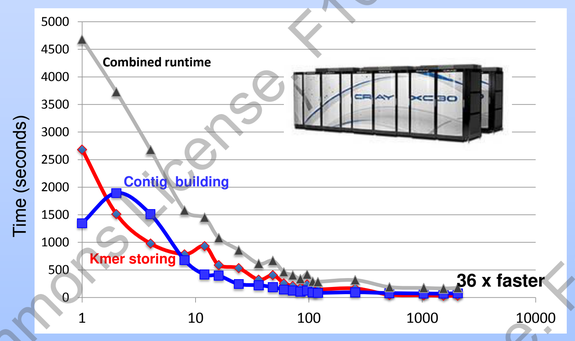
MPI inchworm

Inchworm is the first part of the TrinityRNASeq software. We describe here the new inchworm program based on **Message Passing Interface (MPI)**.

Inchworm is made of three main parts, as shown in the graph below showing scaling. The **kmer** building and the **contigs** building are now fully parallel, and indeed show good scaling. **Harvest** contigs is still sequential and is in the process of being parallelized.

MPI parallel

mouse.fasta file → **Building** Kmer and partitioning according to **central** kmer value and thread number → **Building** Contigs → **Working on parallelizing Sequential I/O** → **Harvesting** contigs → **iworm.fasta**



We are also working on building the RNA-seq contigs of the mexican axolotl,

which has a sequence that is many times larger than the mouse.
Credit: Jessica Whited
at *Brigham* and Women's Hospital (BWH)
Regenerative Medicine Center

Fortran2008 inchworm

"Cray BioLib" was a library of fortran90 routines that was developed by Bill Long a few years ago. We are now adapting and using this library in order to build a **Fortran2008** version of inchworm. We focused first on reading, and on sorting the kmers. We are now working on defining an efficient algorithm for the kmer store and contig builds. Performance of **Fortran2008** reads, bit conversion, and sorting is excellent. The library is also helping define the above MPI version.
Note that **co-arrays will be part of gnu fortran (gfortran) starting with version 4.10**.

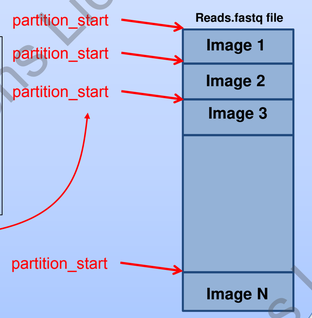
This Cray bioinformatics library includes the following list of Fortran library routines:

- **Parallel search and sort routines**
- **Parallel Smith-Waterman alignment routines**
- **Parallel sequence manipulation routines**
- **Parallel file handling routines**
- **Bit manipulation routines**

Example of use for partitioning kmers, or reads, file among images
("images" in Fortran2008 are equivalent to the concept of MPI ranks):

Actual Fortran2008 code, not a pseudocode

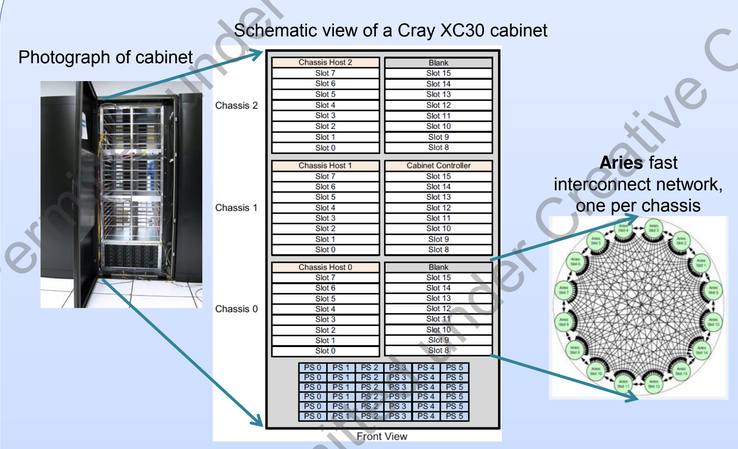
```
partition_size  = input_lengh/total_number_of_PEs
partition_start = max(1, (my_image-1)*partition_size - read_overlap)
partition_stop  = min(input_length,  my_image*partition_size)
if (my_image == total_number_of_images) partition_stop = input_length
partition_length= partition_stop - partition_start + 1
allocate(Buffer_kmers(partition_length), stat=ios) ! Same as malloc in C
read(unit,pos=partition_start,iostat=ios) Buffer_kmers
```



Example using bit manipulation (ASCII to binary) in Cray BioLib:

```
A = 0100 0001
C = 0100 0011
T = 0101 0100
G = 0100 0111
```

Computations at Cray, Inc.



A single cabinet of the **Cray XC30** is composed of 3 chassis. One chassis contains 16 slots, that each hosts a blade. One blade contains 4 nodes. In a given cabinet populated with, for example, IVB-10 (as used for the computations shown here), each node has a total of 20 MPI ranks (or images). With this configuration the fast Aries network (shown above, right) interconnects 16 X 4 x 20 processes, or 1280 MPI ranks (or images). One cabinet of IVB-10 can thus host a total of 3840 MPI ranks. The XC30 figure below (as part of the *GalaxyProject* at Cray) shows an example with 2 rows of 2 pairs of cabinets.
Note: Some of the blades can be replaced with **solid-states drives (SSD).** Those SSD's then become part of the Aries interconnect, **for fast I/O**. It can be used in conjunction to an external file system (***Sonexion***). This configuration is called the ***Local Storage Hierarchy (LSH)*** architecture.

All computations are performed on the XC30. We also developed a Galaxy interface on one of our Cray XC30, and show that all features of the GalaxyProject are available on a Cray computer. The figure above shows the **GalaxyProject at work on the Cray XC30.**

We thank for their support:

grant #**1-U24-CA180922-01**