Pacbiokit4b – an integrated toolset for PacBio bioinformatics

Release 0.4.6

CAUTION – this document has not been updated to include new functionality added since original cloning of the BioKanga 4.2.0 documentation.

Overview

pacbiokit4b is an integrated single process image integrating a number of subprocesses, each providing bioinformatics task functionality. Having to a single process image instead of multiple independent process images provides assured integration and release compatibility between the various subprocesses; if ‘pacbiokit4b’ is loaded then all dependent subprocesses are guaranteed available and these subprocesses will all be at the same release level as they are part of the same process binary image.

pacbiokit4b can be natively compiled for execution on either Windows (Visual Studio 2017 or 2019) or Linux (GNU tool chain) x64 hosting platforms – build instructions are provided in the source release INSTALL text document.

Subprocesses within ‘pacbiokit4b’ are invoked by specifying the subprocess name on the command line immediately following the ‘pacbiokit4b’ startup request . Entering ‘kit4bpacbio’ with no subprocess specified will return a list of available subprocesses. You can get help on any subprocess by following the subprocess name with the ‘-h’ command parameter –e.g “pacbiokit4b ecreads –h”.

Example ‘pacbiokit4b’ command invocation

```
> pacbiokit4b

 Help for pacbiokit4b Version 0.4.6

  pacbiokit4b PacBio Processing Toolset

  Please specify one of the following subprocesses:

  filter          Filter PacBio reads for retained hairpins

  ecreads         Error correct PacBio reads

  contigs         Assemble error corrected PacBio reads into contigs

  eccontigs       Error correct assembled PacBio contigs

  swservice       Distributed computing SW service provider
```

kmerdist	Generate exact matching K-mer distributions from MAF

To obtain parameter help on any subprocess then enter that subprocess name e.g:

```
pacbiokit4b filter -h
```

Subprocess Overview

Use the 'pacbiokit4b filter' subprocess request to invoke filtering of raw reads for identification of putative retained SMRTbells. Reads containing putative SMRTbells will be split at the SMRTbell into shorter subreads.

The 'pacbiokit4b ecreads' subprocess will identify putative overlaps between a given sequence and all other reads which at least partially overlap the given sequence and then use Smith-Waterman scoring on these overlaps to confirm the overlap. Accepted overlaps are then stacked and consensus bases used to error correct the given sequence. All reads meeting user specified length thresholds will be, in turn, error corrected. This error correcting phase is usually repeated as on the initial error correction processing it is difficult to reliably determine the consensus where there has been a deletion from the read currently being error corrected. A third iteration of 'pacbiokit4b ecreads' is used in a different processing mode whereby the overlaps between all pairs of reads are reported – these allow the reads to assembled into contigs by the next processing phase.

In the next processing phase by 'pacbiokit4b contigs' a DAG graph is built from the overlaps reported by the last iteration of 'pacbiokit4b ecreads', split into subgraphs, and paths through each subgraph scored with the maximal scoring path chosen as the contig to be assembled. Resultant contigs are then assembled and reported.

The final processing phase by 'pacbiokit4b eccontigs' is to take the assembled contigs as generated by 'pacbiokit4b contigs', overlap these with the error corrected reads from 'pacbiokit4b ecreads' and use the stacked overlaps to further correct (polish) the assembled contigs.

The most expensive, in terms of CPU resource requirements, is the Smith-Waterman processing required during the error correction phases. Accordingly, the error correcting workflow can take advantage of the distributed computing capability offered by Smith-Waterman service providers. A service provider contacts a pacbiokit4b instance acting as a controller over IP4/6 and offers to provide a number of Smith-Waterman instances.

Installation

To install, simply copy the compiled pacbiokit4b binary image into a directory which is on your executable path. There are a number of user specified parameters specific to each subprocess of the pacbiokit4b toolkit. Most of these are optional, and the optional parameters would generally only be required to be specified by the user when targeting some specific analytics issue. In general, the only mandatory parameters are those specifying input datasets and where to write output result sets.

Resource Requirements

The pacbiokit4b architecture is optimised for execution on modestly resourced hardware. It is multithreaded so as to take advantage of modern multicore CPUs, and processes memory resident datasets which are loaded from disk at process initiation.

To avoid memory paging with consequent disk threshing it is recommended that there should be at least the following amount of physical memory installed: 6x the total size (bp) of all PacBio reads used when error correcting or 6x the total of all contigs plus the total size of all error corrected reads used for error correction when polishing assembled contigs. Additionally, each CPU core used in Smith-Waterman overlap detection and stacking for error correction may require an additional 4GB of memory. Thus, assuming the raw PacBio dataset to be error corrected totals 20Gbp and 32 cores are available for Smith-Waterman processing the total memory required will be around 248GB. If the distributed Smith-Waterman service provider capability is utilised then a service provider will need up to 4GB of memory per service instance only, irrespective of the total size of the PacBio reads dataset. Thus, if a service provider is providing 32 service instances then that provider will need 128GB of memory. The 4GB per Smith-Waterman instance is an upper limit (banded Smith-Waterman is utilised) and will be dependent on a number of factors including read lengths, error rates, and expected overlap lengths.

Disk requirements are minimal, in addition to the space required to hold to original raw readsets then space will be needed for the error corrected reads plus assembled contigs and the final polished contig assembly.