Pipelines in Perl With eHive

YAPC::EU 2016

24th – 26th August 2016

Andy Yates: Team Leader, EMBL-EBI

www.ebi.ac.uk

www.ensembl.org

https://github.com/Ensembl/ensembl-hive

@enscore

EMBL-EBI

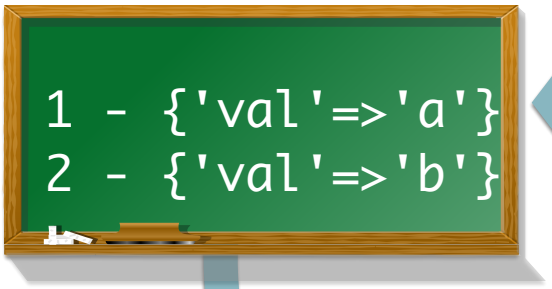
Processing Workflows with Perl and eHive

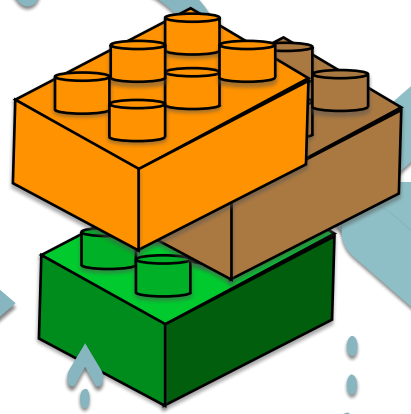
- https://github.com/Ensembl/ensembl-hive
- A workflow management system in Perl
- Uses traditional schedulers (LSF/OpenLava) to submit work
- Workflows are DAGs using semaphores to replicate programming constructs

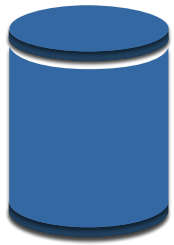
eHive: How it Works

The central blackboard holds a list of jobs to process

An analysis can create more jobs on the blackboard to preconfigured endpoints

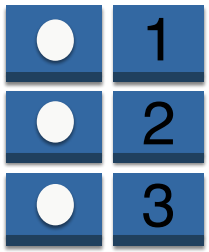






Output can flow into a database table ...

… or perform a custom action e.g. writing a collection of files to disk ...



… or accumulate data for a funnelled downstream analysis

Jobs are given to an analysis as a collection of key/value pairs defining the unit of work. Analyses responding to the incoming parameters by executing a set of instructions similar to a subroutine call

Failed jobs are automatically retried

```perl
package CountATGC;

use strict;
use warnings;
use Bio::SeqIO;
use base qw/Bio::EnsEMBL::Hive::Process/;

sub fetch_input {
  my ($self) = @_;
  my $chunkfile = $self->param_required('chunk_name');
  $self->param('chunk_in', Bio::SeqIO->new(-file => $chunkfile));
  return;
}

sub run {
  my ($self) = @_;
  my ($at, $gc) = (0,0);
  foreach my $chunkseq ($self->param('chunk_in')->next_seq()) {
    my $seqstring = $chunkseq->seq();
    $at += @{[$seqstring =~ /([AaTt])/g]};
    $gc += @{[$seqstring =~ /([GgCc])/g]};
  }
  $self->param('at', $at);
  $self->param('gc', $gc);
  return;
}

sub write_output {
  my ($self) = @_;
  $self->dataflow_output_id({
    at => $self->param('at'),
    gc => $self->param('gc'),
  });
  return;
}

1;
```

Get your input data sets

Get a path to a data file, initalise another object for reading and stash it

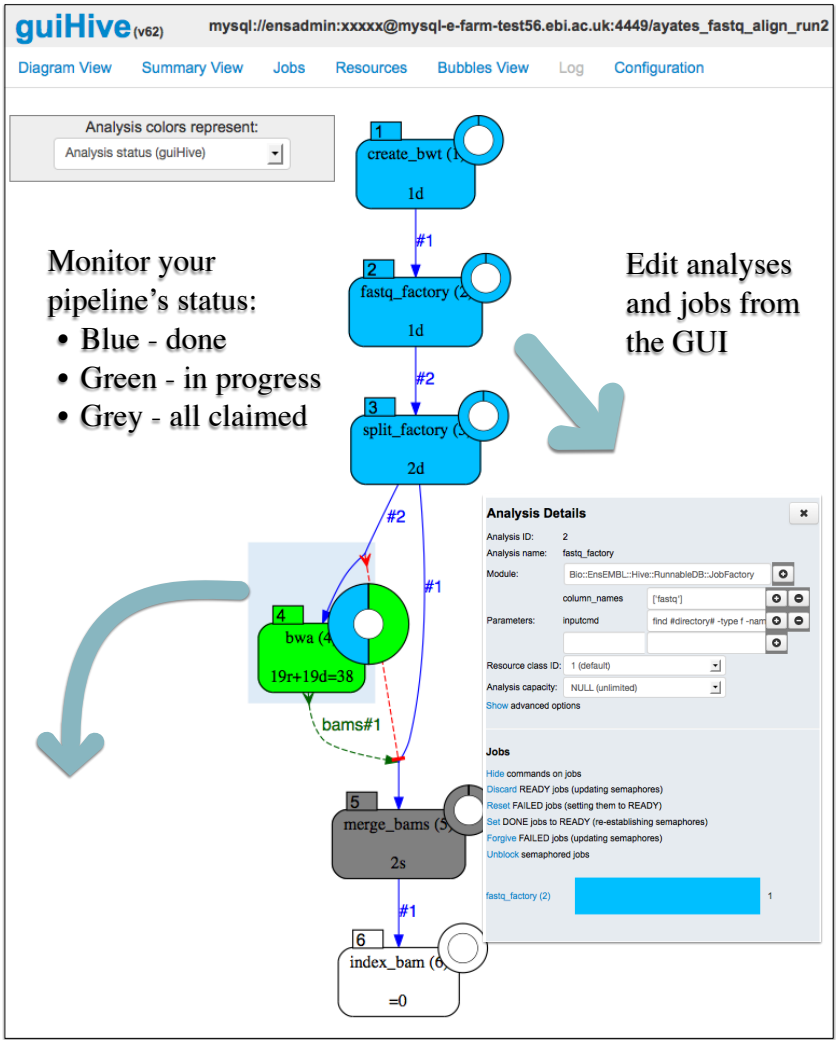
Do your processing

We are using the object to get sequence, counting the number of AT/GC by regex and stashing the result

Send a hash to the next task

Current task does not need to know what will consume the hash

guiHive for Monitoring Your Application



- Written in GO with D3.js
- Persistent web server talking to multiple hives
- Edit your pipeline as it is running
- Watch your application execute if you really want to

Tutorial from

https://github.com/andrewyatz/eHiveDemo-yapceu2016

The Entire Ensembl Team

Acknowledgements

Andrew Yates[1], Wasiu Akanni[1], M. Ridwan Amode[1], Daniel Barrell[1,2], Konstantinos Billis[1], Denise Carvalho-Silva[1], Carla Cummins[1], Peter Clapham[2], Stephen Fitzgerald[1], Laurent Gil[1], Carlos García Girón[1], Leo Gordon[1], Thibaut Hourlier[1], Sarah E. Hunt[1], Sophie H. Janacek[1], Nathan Johnson[1], Thomas Juettemann[1], Stephen Keenan[1], Ilias Lavidas[1], Fergal J. Martin[1], Thomas Maurel[1], William McLaren[1], Daniel N. Murphy[1], Rishi Nag[1], Michael Nuhn[1], Anne Parker[1], Mateus Patricio[1], Miguel Pignatelli[1], Matthew Rahtz[2], Harpreet Singh Riat[1], Daniel Sheppard[1], Kieron Taylor[1], Anja Thormann[1], Alessandro Vullo[1], Steven P. Wilder[1], Amonida Zadissa[1], Ewan Birney[1], Jennifer Harrow[2], Matthieu Muffato[1], Emily Perry[1], Magali Ruffier[1], Giulietta Spudich[1], Stephen J. Trevanion[1], Fiona Cunningham[1], Bronwen L. Aken[1], Daniel R. Zerbino[1] and Paul Flicek[1,2,*]

[1]European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, UK and [2]Wellcome Trust Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB10 1SA, UK

Funding

wellcome

EMBL

National Human Genome Research Institute

Open Targets

BLUEPRINT epigenome

BBSRC bioscience for the future

TRANSFORMING GENETIC MEDICINE INITIATIVE

MedBioinformatics

Co-funded by the European Union