

How to eHive

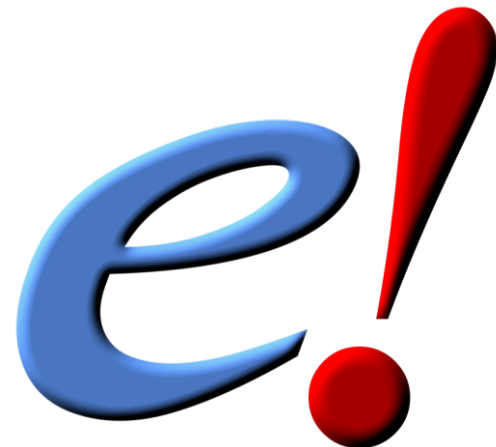
Leo Gordon

EnsEMBL/Compara

EMBL-EBI

Wellcome Trust Genome Campus

Hinxton CB10 1SD, UK



eHive: definitions

- **eHive**: a database-centric system for setting up and running software pipelines on a multithreaded compute resource
- **software pipeline**: a system of inter-linked *data processing steps* where the input of one step is the output of another
- **Runnable/code**: a PERL module to be run
- **job**: an individual data processing step (*completely defined by Runnable+parameters*)
- **input & output**: job parameters in the form of PERL hash

Flat parameter space vs. analysis abstraction

Flat parameter space:

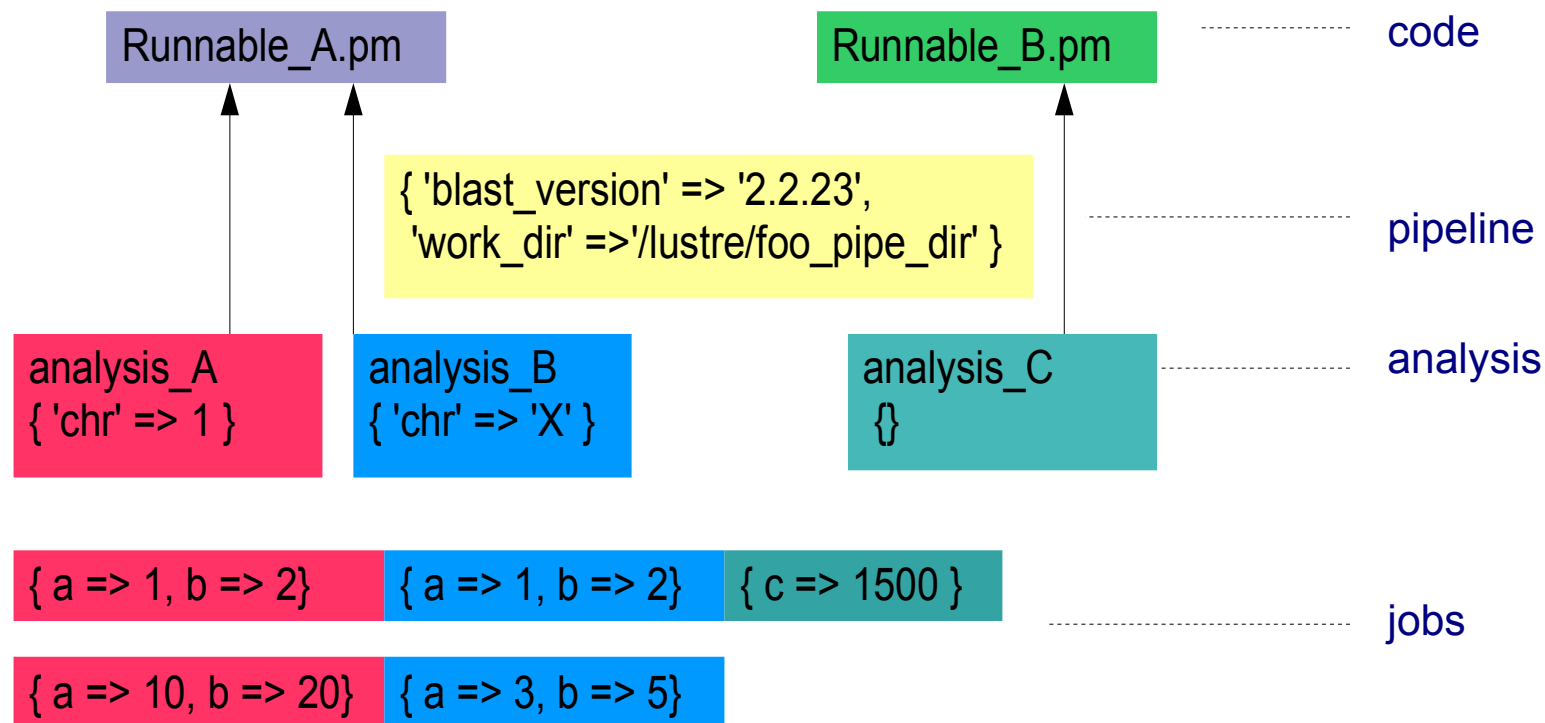
```
{ 'species' => 'Human', 'chr' => 1}  
{ 'species' => 'Human', 'chr' => 2}  
...  
{ 'species' => 'Human', 'chr' => 'X'}  
  
{ 'species' => 'Mouse', 'chr' => 1}  
{ 'species' => 'Mouse', 'chr' => 2}  
...  
{ 'species' => 'Mouse', 'chr' => X}
```

Analysis abstraction level:

```
Analysis_A: { 'species' => 'Human'}  
  { 'chr' => 1}  
  { 'chr' => 2}  
  ...  
  { 'chr' => 'X'}
```

```
Analysis_B: { 'species' => 'Mouse'}  
  { 'chr' => 1}  
  { 'chr' => 2}  
  ...  
  { 'chr' => 'X'}
```

parameters (code, pipeline, analysis, job)



Static vs. dynamic job creation

Database: blackboard with jobs to do. Why not write on it?

Static:

(EnsEMBL pipeline RuleManager.pl)

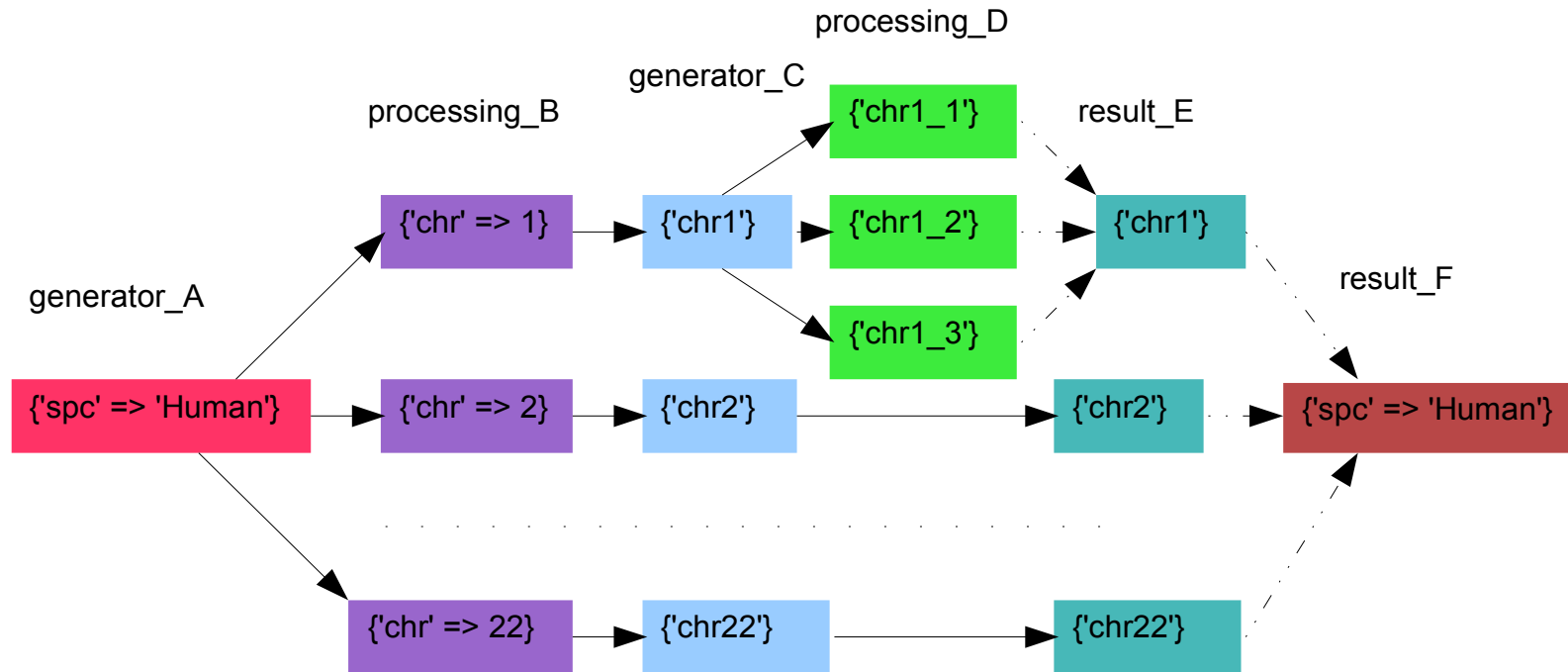
1. Create all the jobs first
2. Create “blocking” rules that define the order of execution
3. Run jobs that happen to be unblocked
4. Refresh blocked status by reapplying rules
5. Repeat from 3.

Dynamic:

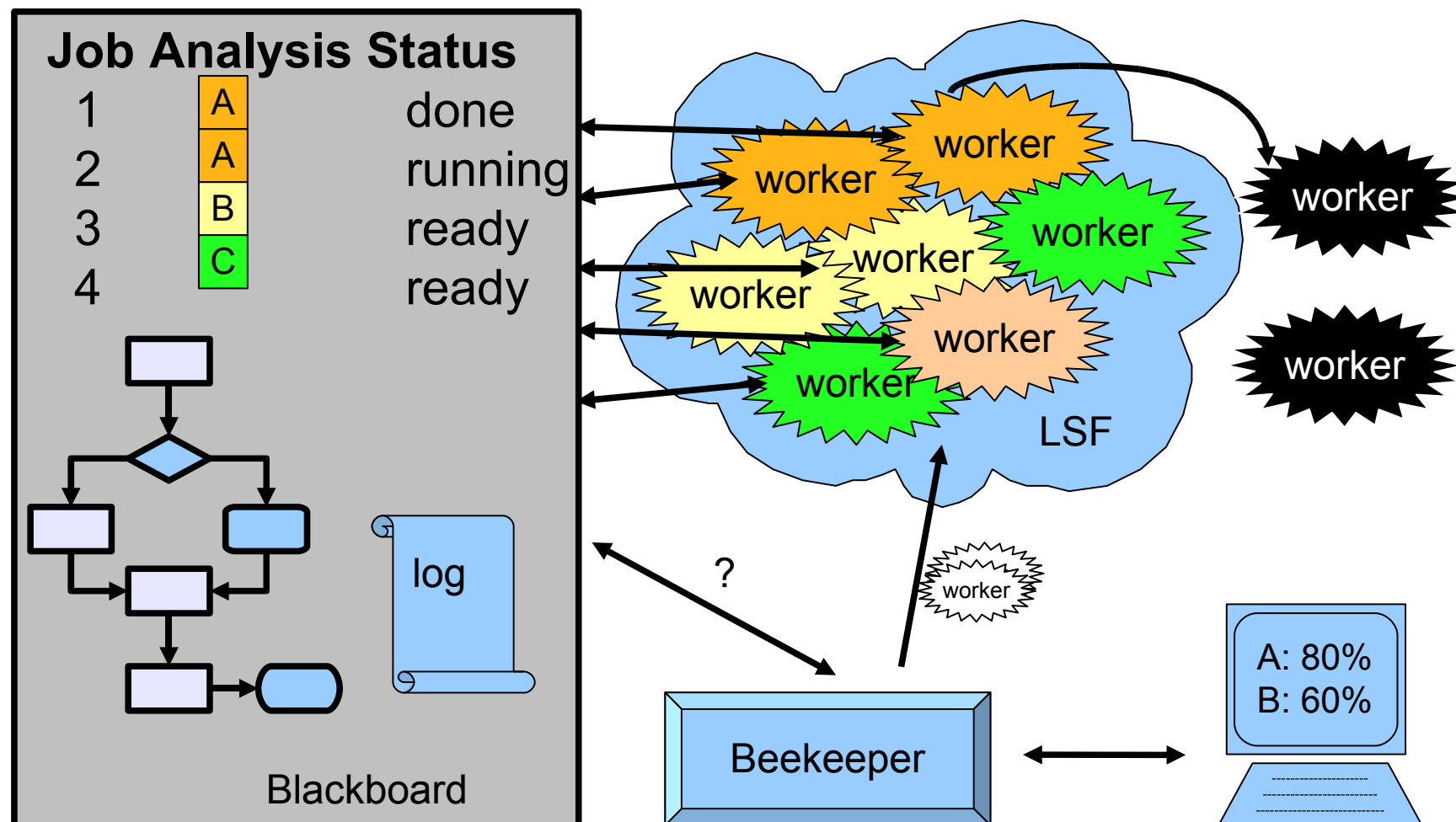
(eHive's beekeeper + Workers)

1. Create some jobs
2. Create rules of job creation (s.-c. dataflow)
3. Create blocking/waiting rules
4. Run jobs that happen to be unblocked
5. Refresh blocked status by reapplying rules
6. Repeat from 3.

typical eHive dataflow diagram



eHive: Workers as autonomous agents



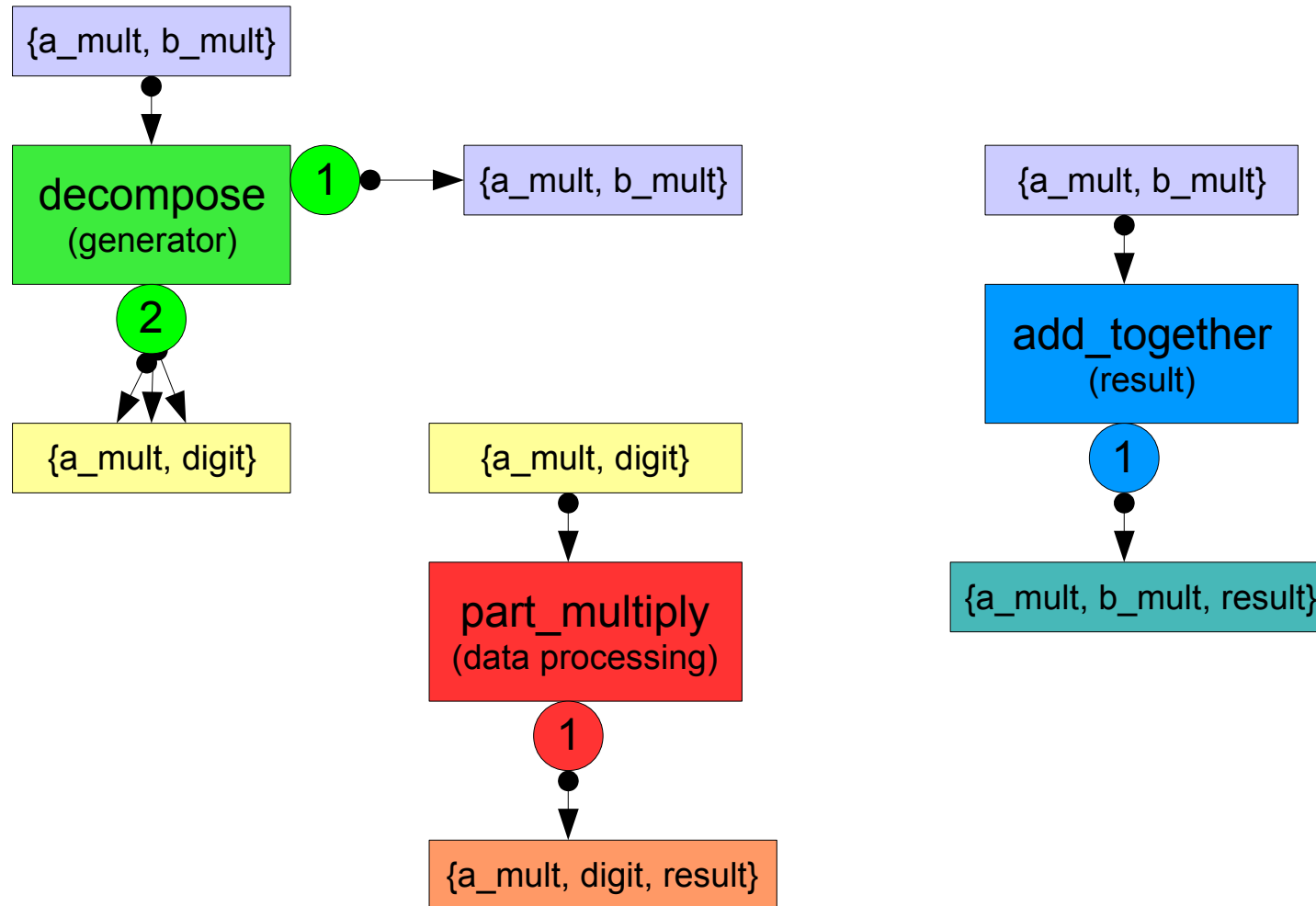
Long multiplication example

$$\begin{array}{r} 9650156169 \\ \times 327358788 \\ \hline 77201249352 \\ 77201249352 \\ 67551093183 \\ 77201249352 \\ + 48250780845 \\ 28950468507 \\ 67551093183 \\ 19300312338 \\ 28950468507 \\ \hline 3159063427494563172 \end{array}$$

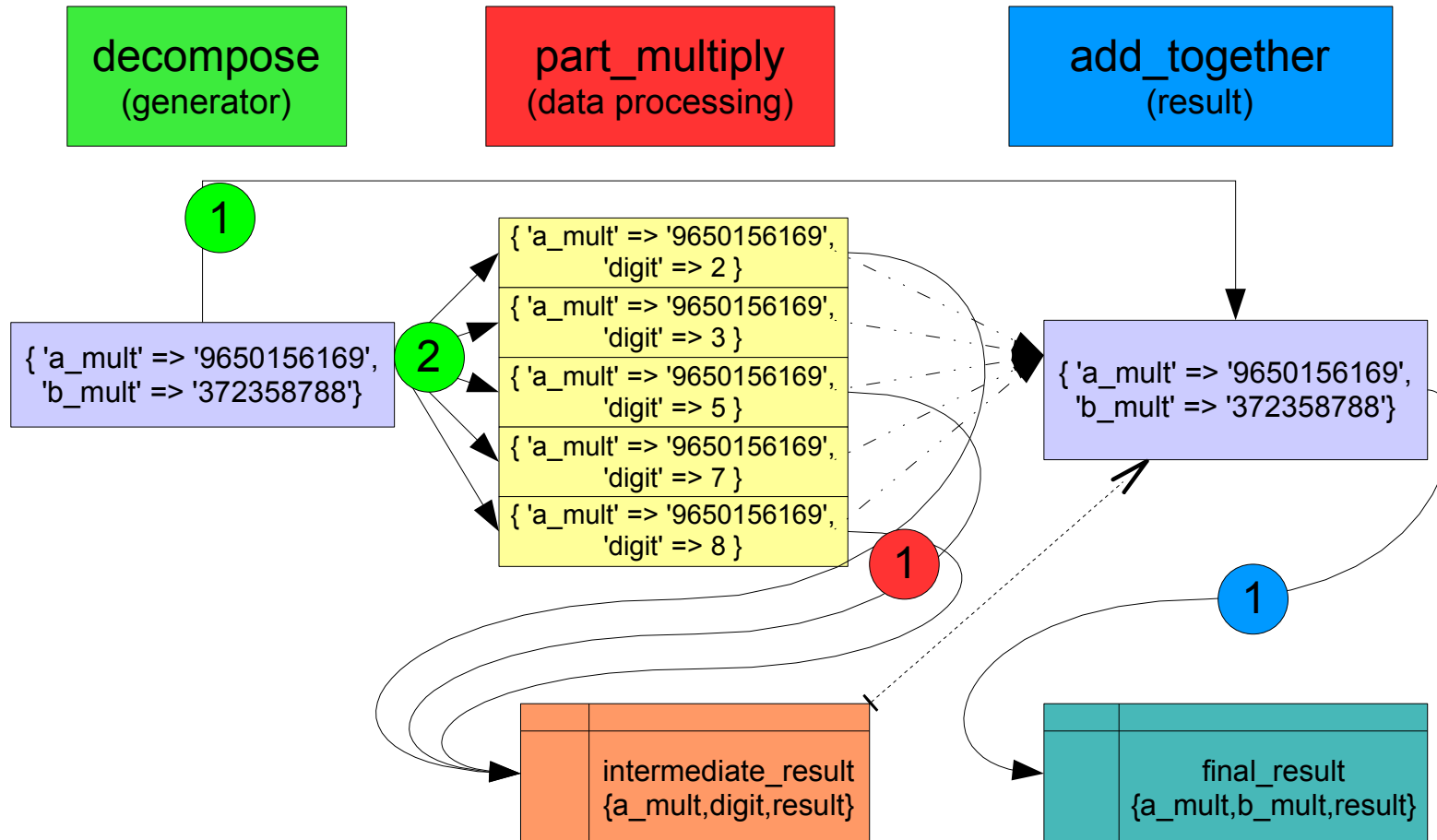
Intermediate results:

$$\begin{array}{l} 9650156169 \times 2 = 19300312338 \\ 9650156169 \times 3 = 28950468507 \\ 9650156169 \times 5 = 48250780845 \\ 9650156169 \times 7 = 67551093183 \\ 9650156169 \times 8 = 77201249352 \end{array}$$

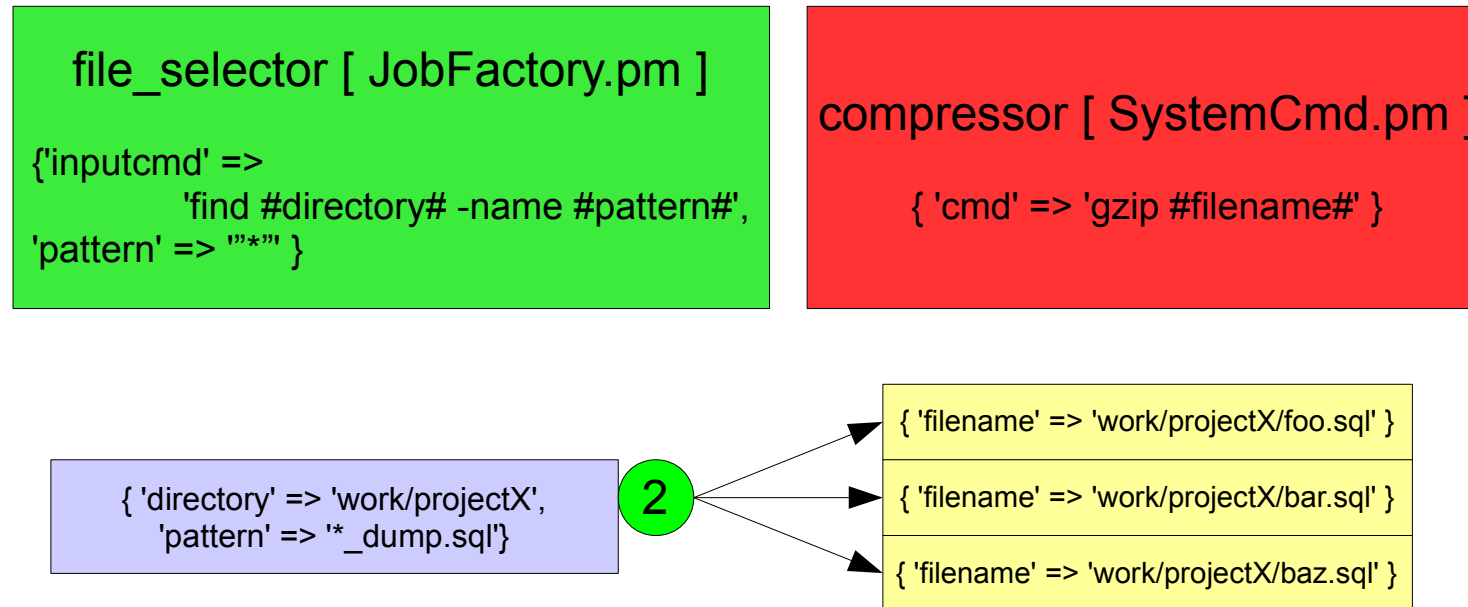
Long multiplication pipeline building blocks and interfaces



Long multiplication pipeline dataflow diagram



Parameter substitution & Universal building blocks (1)



- Expressive power of Unix shell + parallel execution of the farm
- Job tracking, retry, throttling, tricky dependencies, etc

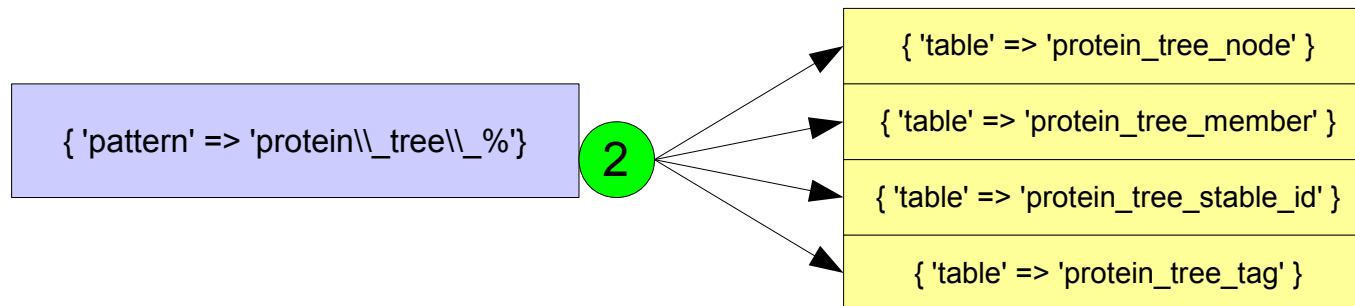
Parameter substitution & Universal building blocks (2)

table_selector [JobFactory.pm]

```
{'inputquery' =>  
  'SHOW TABLES LIKE #pattern#',  
  'pattern' => ""%"} }
```

myisamiser [SqlCmd.pm]

```
{ 'sql' =>  
  'ALTER TABLE #table# engine=MyISAM' }
```



- JobFactory.pm – turns anything into a list of jobs (files, shell, sql, lists). Parser.
- SystemCmd.pm – executes any shell command
- SqlCmd.pm – executes an SQL session, returns insert_ids

References & acknowledgements

- BioMed Central Bioinformatics (11 May 2010):
- ***"eHive: An Artificial Intelligence workflow system for genomic analysis"***
- **Jessica Severin^{1,2} , Kathryn Beal¹ , Albert J Vilella¹ , Stephen Fitzgerald¹ , Michael Schuster¹ , Leo Gordon¹ , Abel Ureta-Vidal^{1,3} , Paul Flicek¹ and Javier Herrero¹**
- **Compara:**
 - Kathryn Beal
 - Stephen Fitzgerald
 - Albert Vilella
 - Javier Herrero
- **Everyone in EnsEMBL, esp**
 - Bronwen Aken
 - Paul Flicek
 - Pablo Marin-Garcia
- **original developers:**
 - Jessica Severin
 - Abel Ureta-Vidal
 - Michael Schuster
- **ehive-users mailing list members**
- **ISG & farm support**