

DISPEL optimisation

perhaps a more general title,

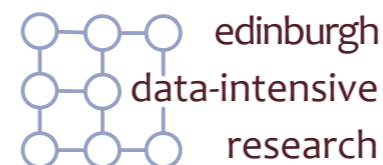
“Optimisation of the enactment of fine-grained distributed data-intensive workflows”

**Chee Sun Liew, Malcolm Atkinson,
Murray Cole, Jano van Hemert**

c.s.liew@sms.ed.ac.uk



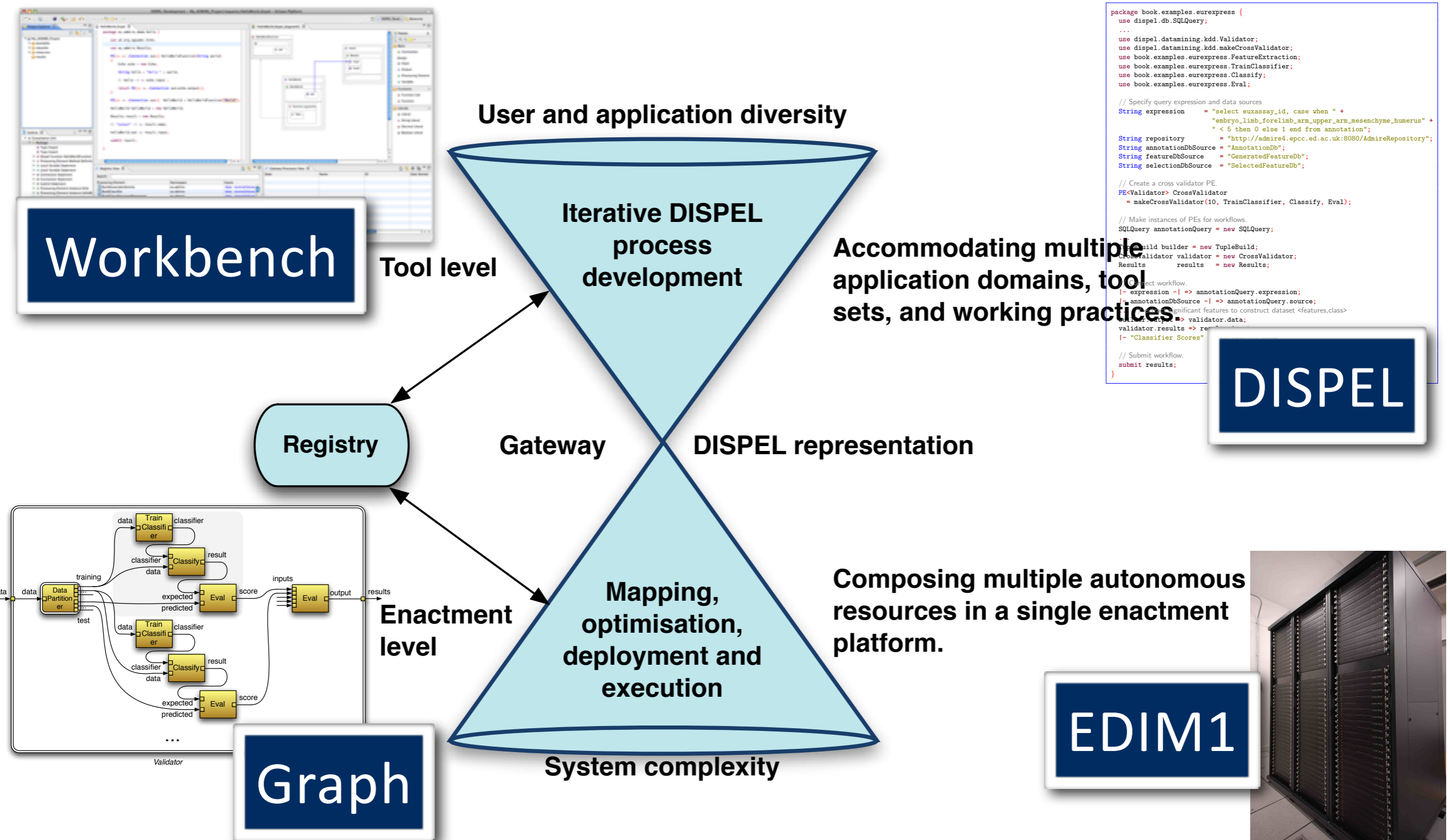
THE UNIVERSITY of EDINBURGH
informatics



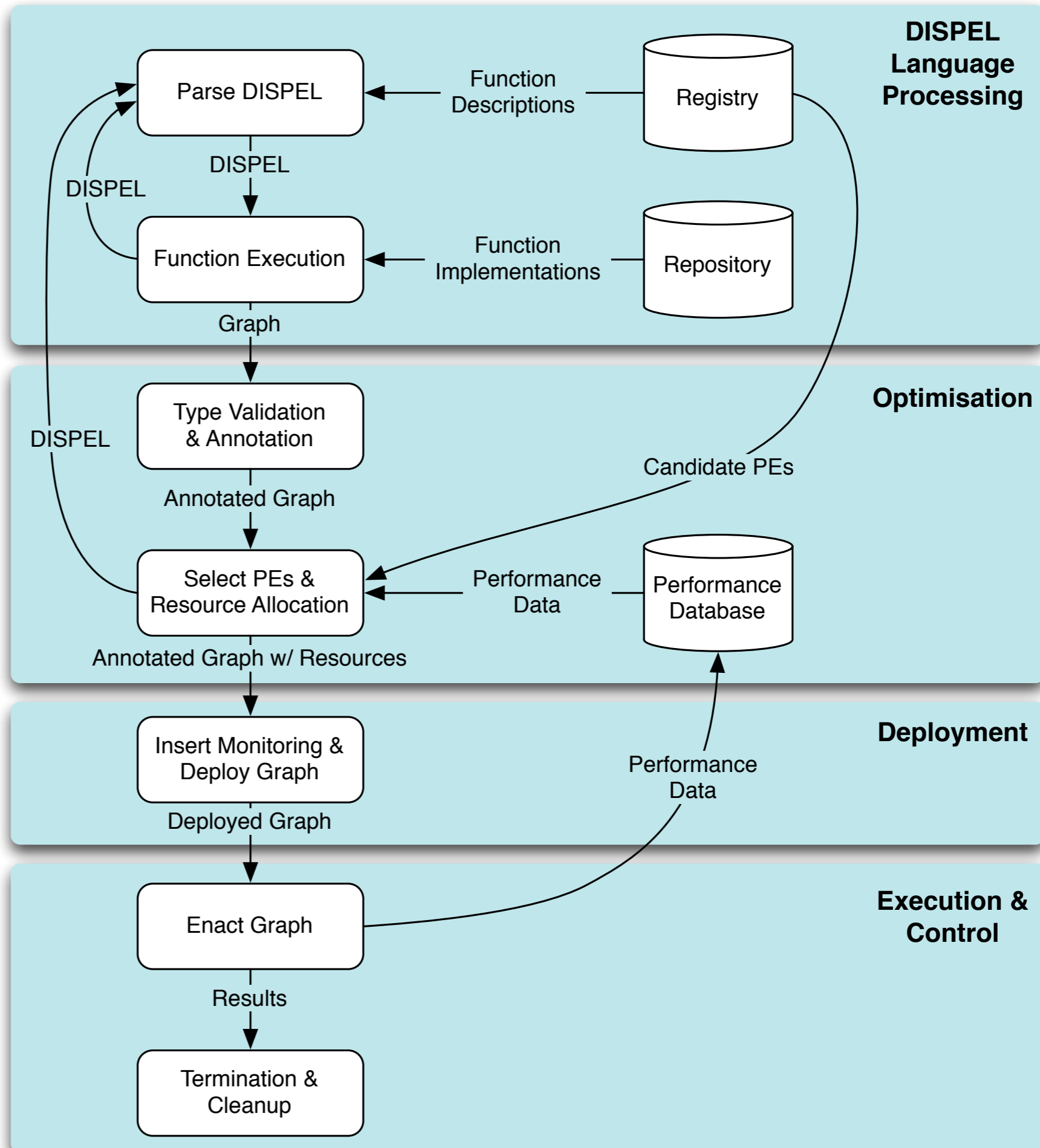
| **epcc** |



Data-intensive architecture

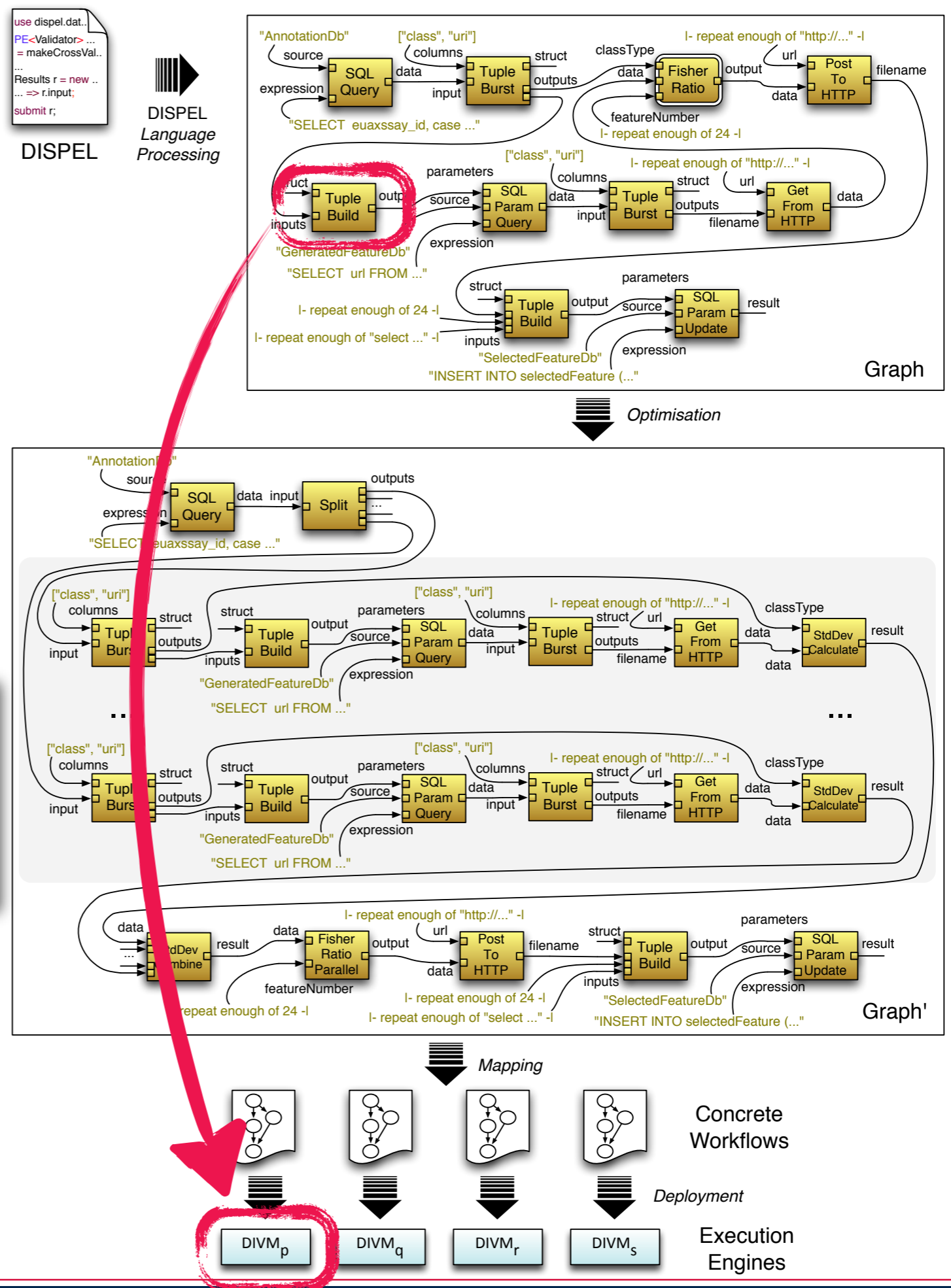


Overview of DISPEL enactment



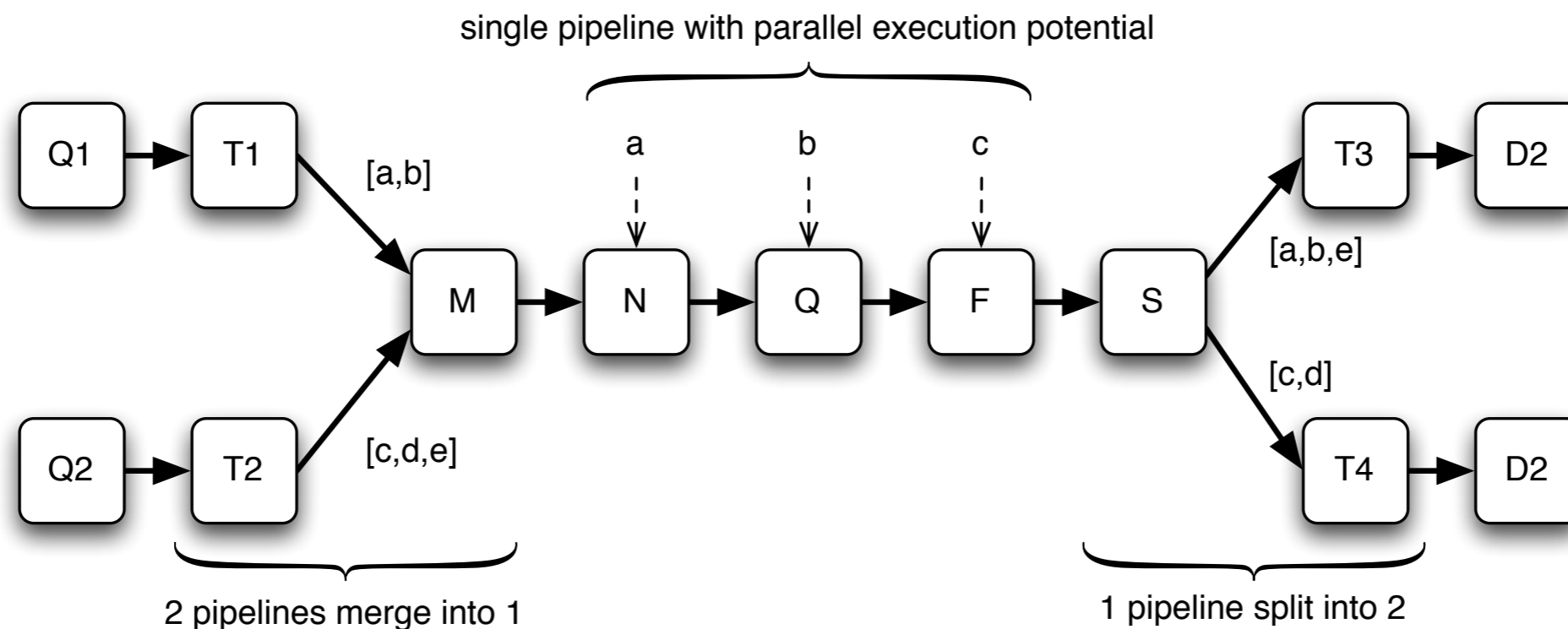
From DISPEL request to concrete workflow

data-streaming
model



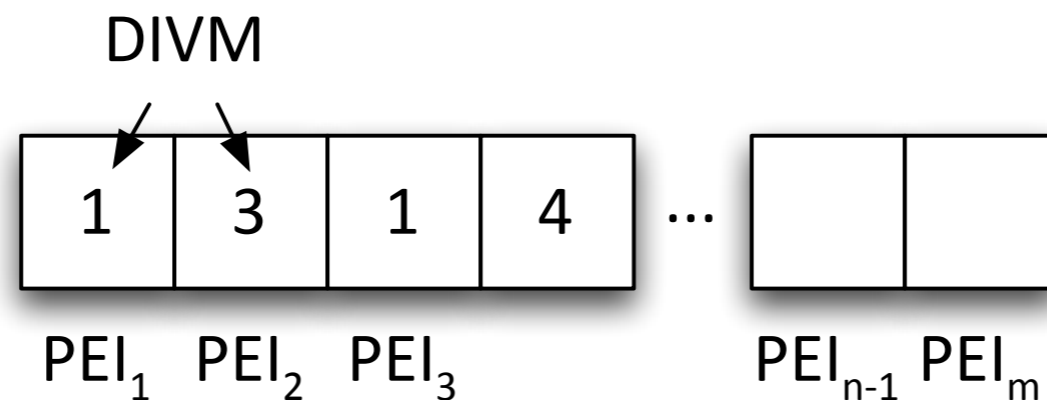
Distributed Data Streaming Graph

- We model workflow request as a graph
 - **Processing Elements (PEs)** - vertices
 - **Data Streams** - edges



Optimisation Problem Definition

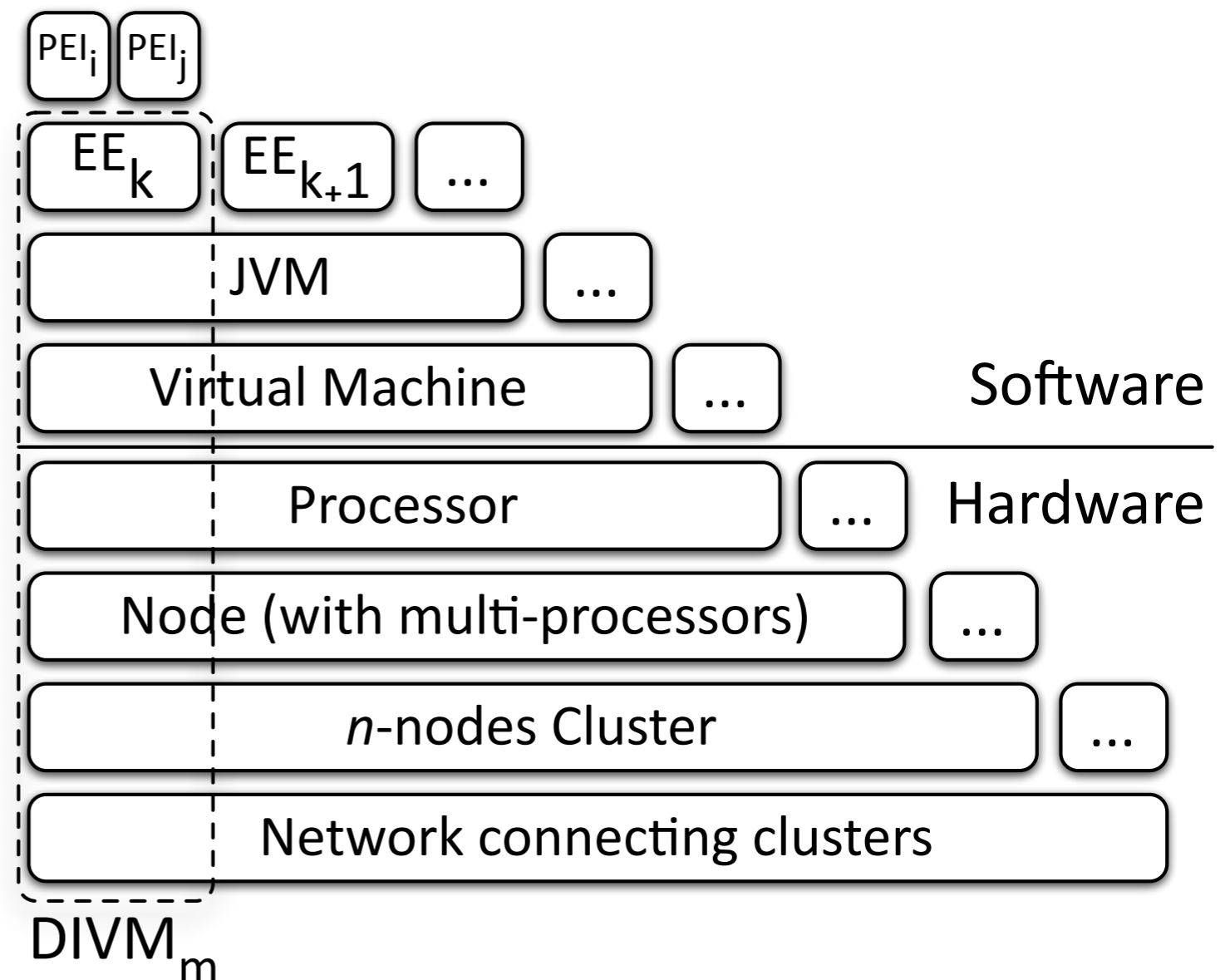
- given a fixed set $P = \{PEI_i\}$ of **PE Instances** that need to be allocated to a set of $M = \{DIVM_j\}$ of **available DIVM** for the enactment of R_k






- Let $|P| = m$ and $|D| = n$, then the number of **possible assignments**, $A(P, V)$, is n^m . There is a **cost function** $C(A)$ that we need to minimise to find the optimum mapping, with performance collected from previous enactment.

Data-intensive Virtual Machine

- Abstraction for the computational environment in which a PE instance runs during enactment.



Information required

- The **descriptions of components** provided by the
 Registry ✓ analysis experts (who designed and implemented
- The **performance of the component instances**
 PDB ✓ on observing previous enactments, e.g. processing time
data (unit cost), memory footprint, etc...
- The **descriptions of the data-intensive platforms,**
 Gateway ✓ properties (e.g. number of computing nodes,
in their connections, etc) and software properties
(e.g. database engines, execution engines, etc).

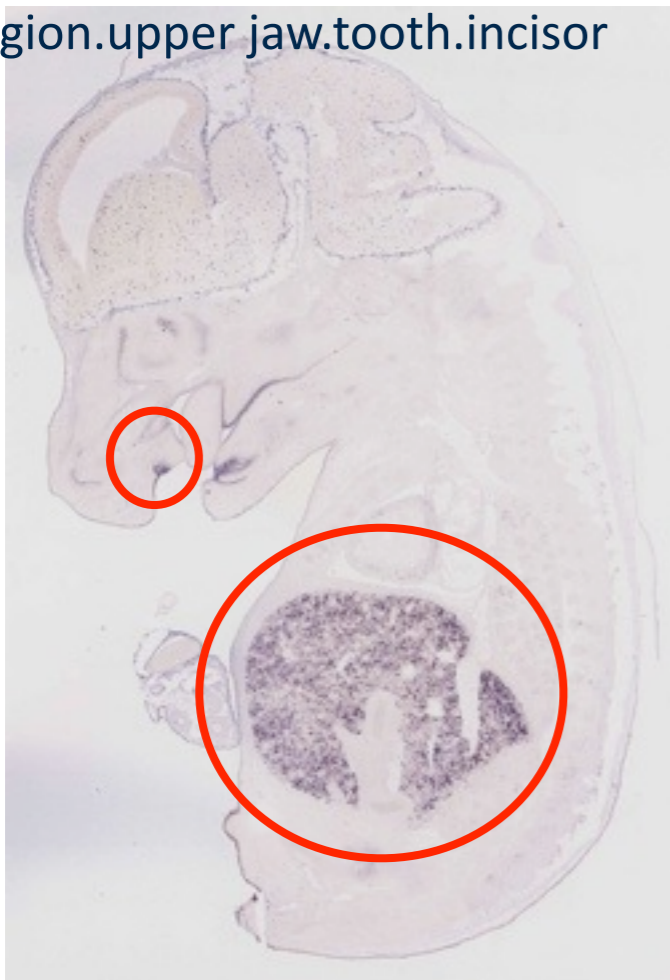
Phases in Optimisation work

- **Phase 0:**
 - investigating the streaming behaviour of PEs
- **Phase 1:**
 - building up measurement framework and PDB
- **Phase 2:**
 - implementing three-stage mapping algorithm
- **Phase 3:**
 - exploring graph transformation options

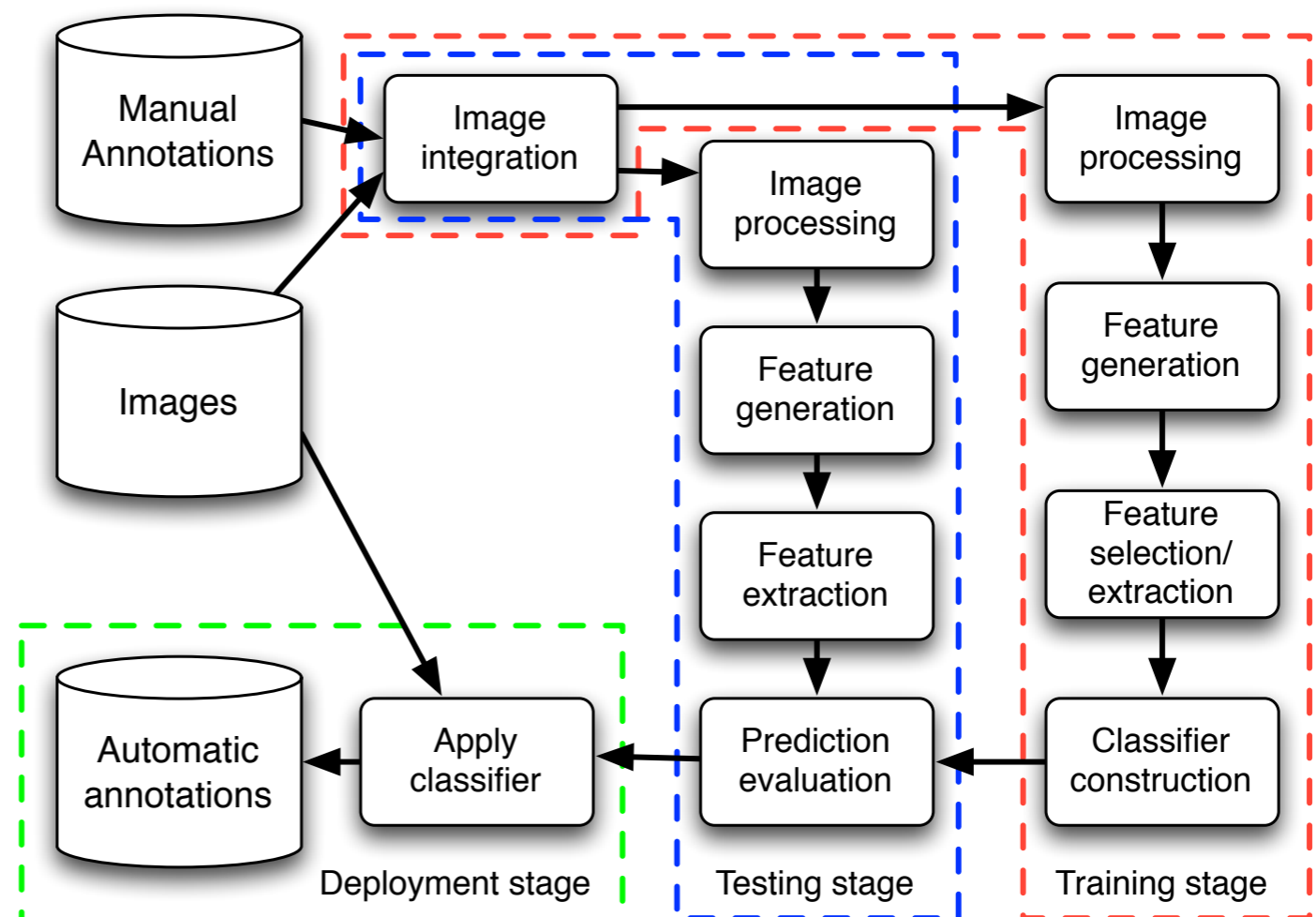
Phase 0: Preliminary experiments

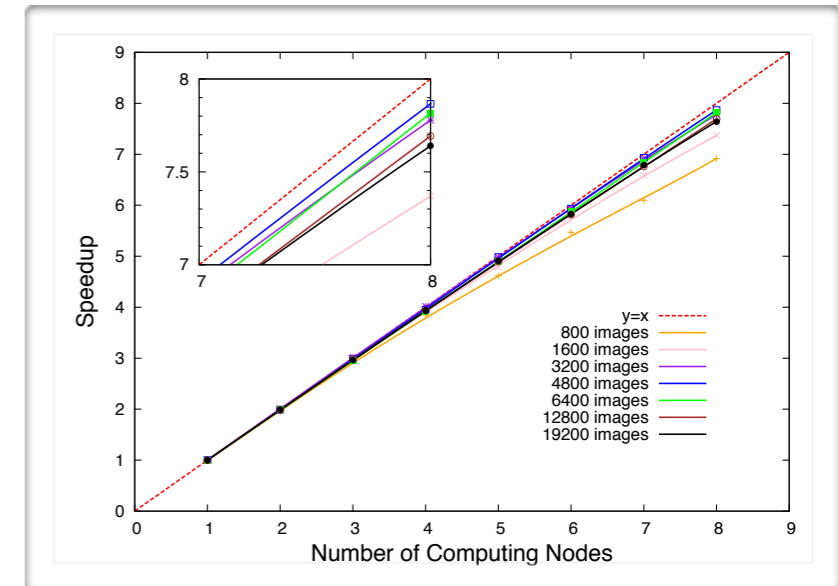
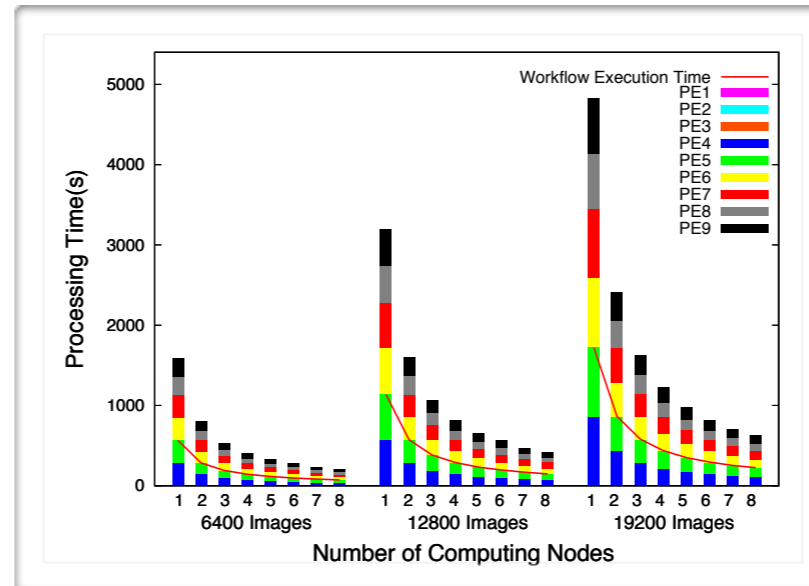
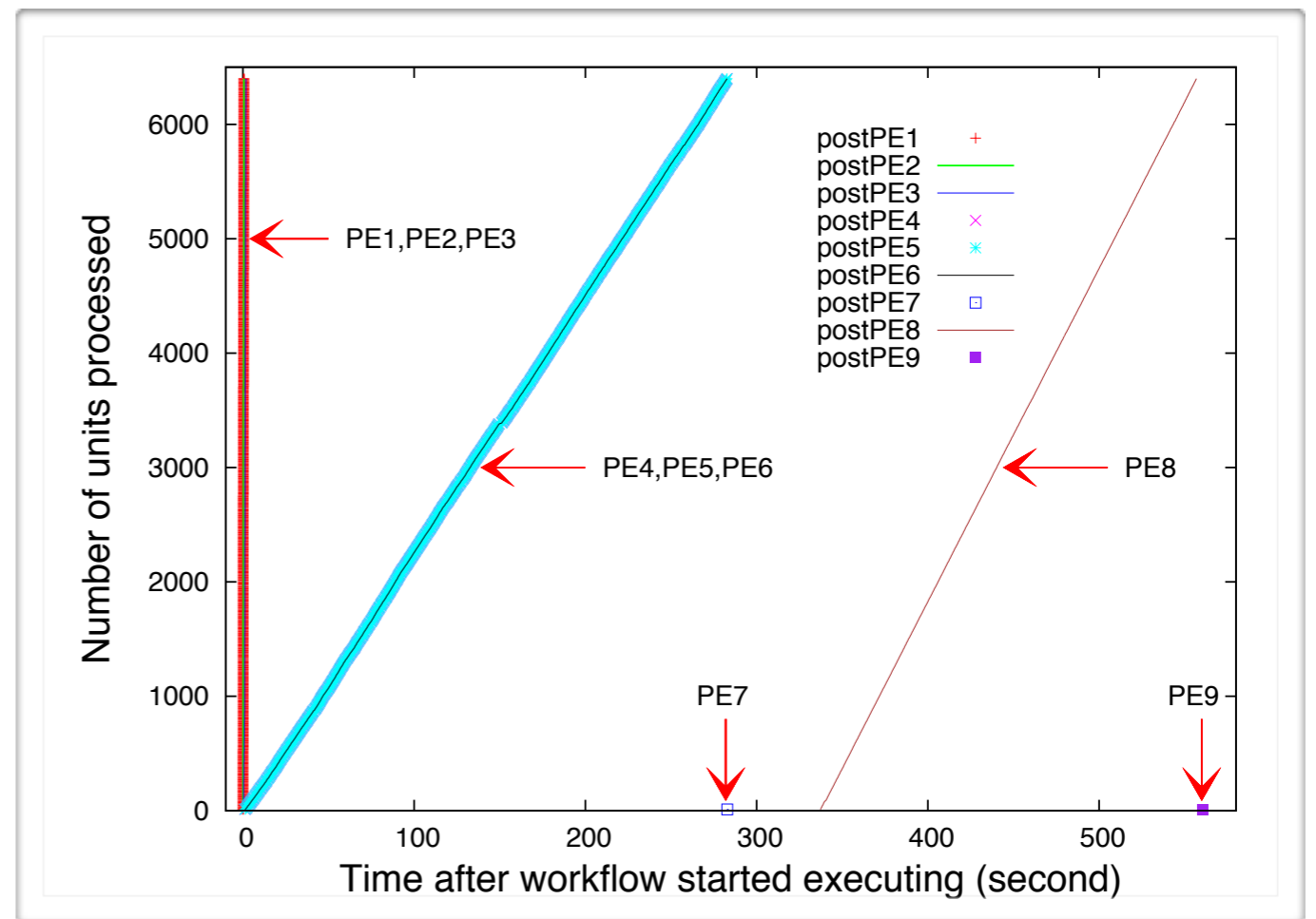
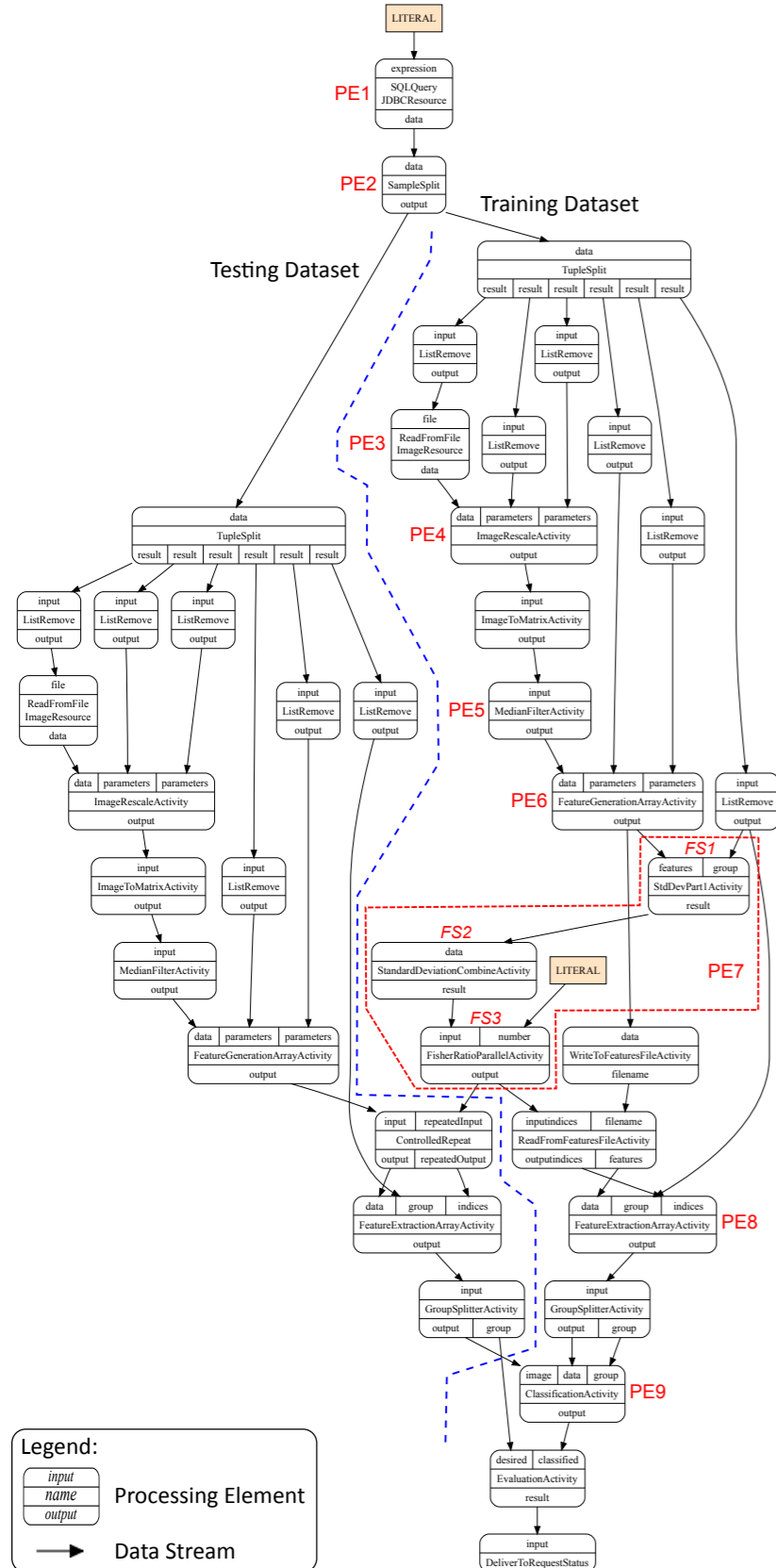
Aim: Automate annotation of mouse embryos images
by **building classifiers** to recognise gene expression

TS23.embryo.organ system.visceral
organ.alimentary system.oral
region.upper jaw.tooth.incisor



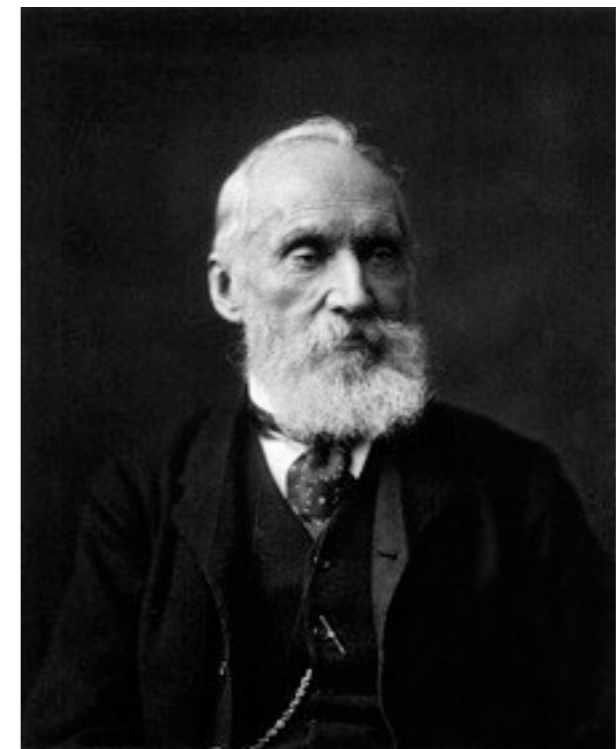
TS23.embryo.organ system.visceral
organ.liver and biliary system.liver.lobe





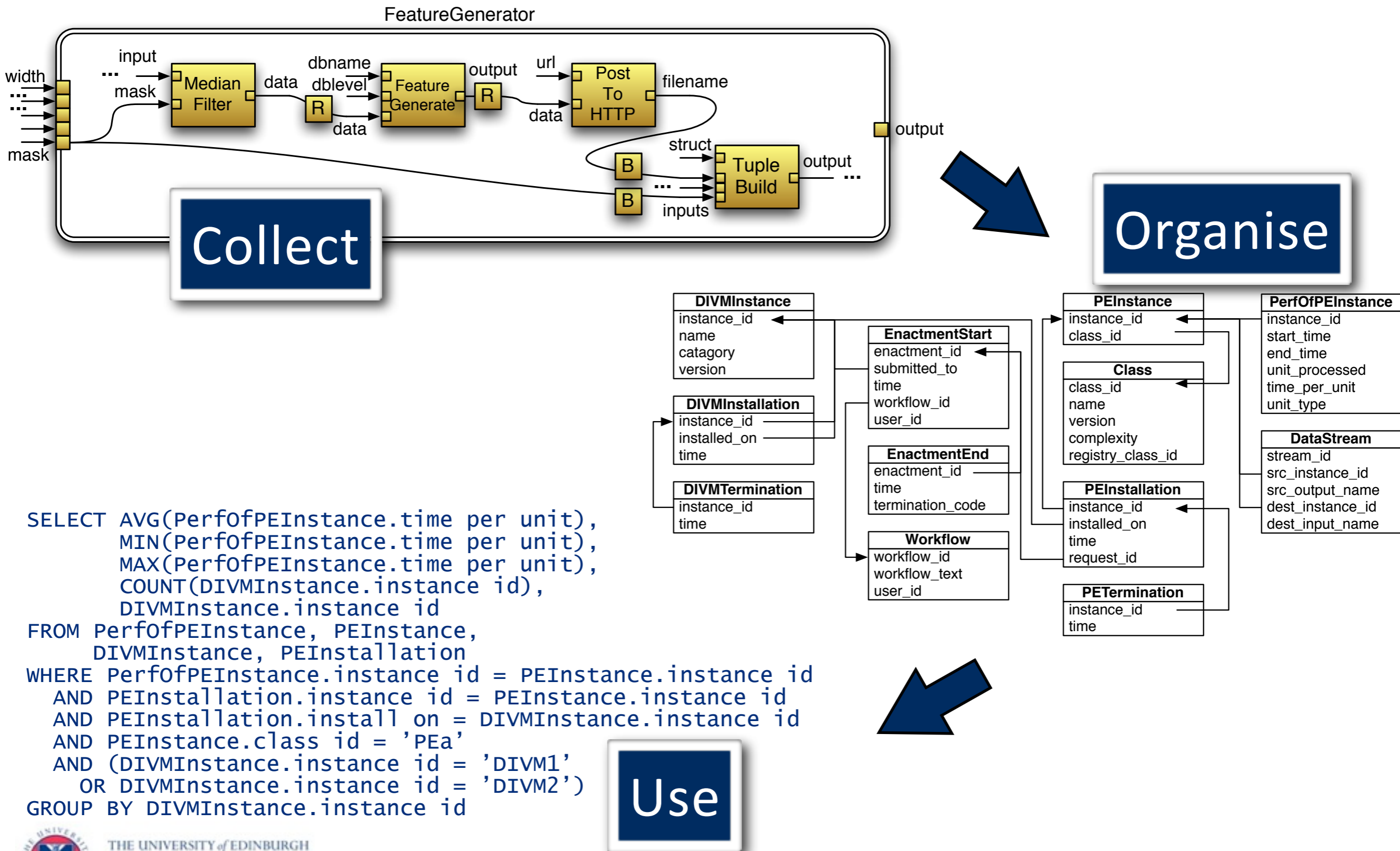
Chee Sun Liew, Malcolm P. Atkinson, Jano I. van Hemert, and Liangxiu Han. Towards optimising distributed data streaming graphs using parallel streams. In DIDC 10: Proceedings of the third international workshop on Data-intensive distributed computing, pages 725-736. ACM, 2010.

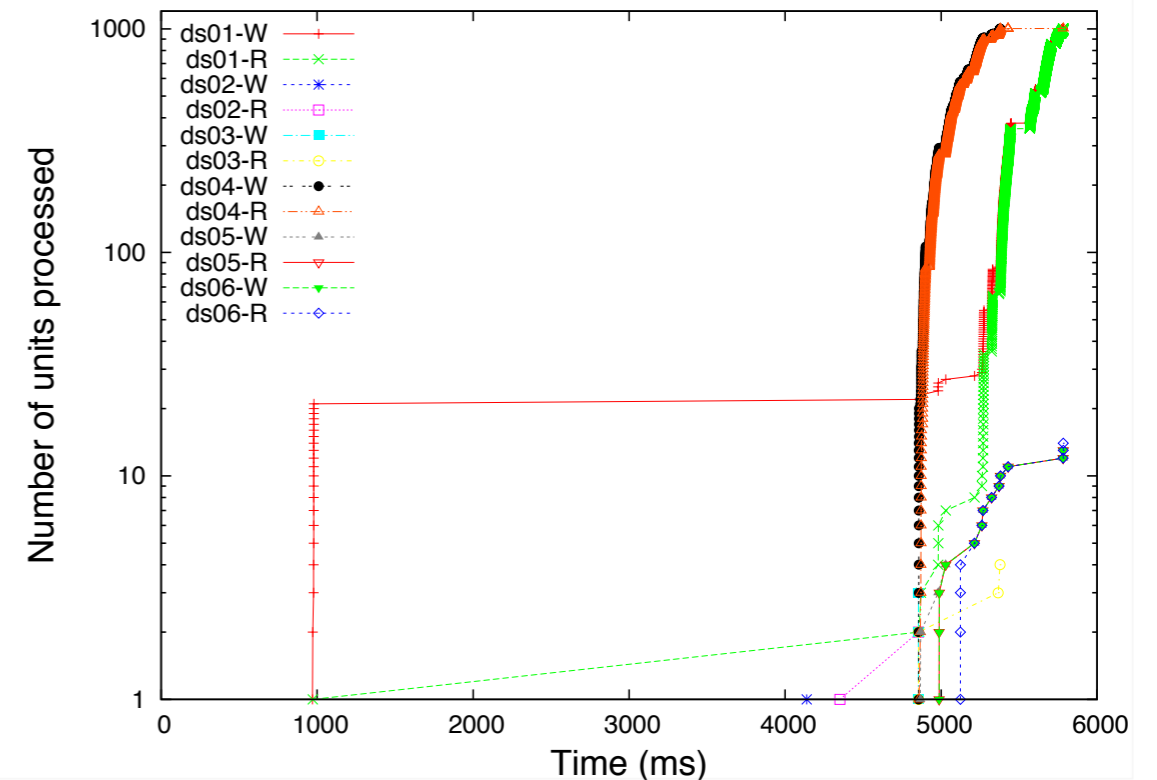
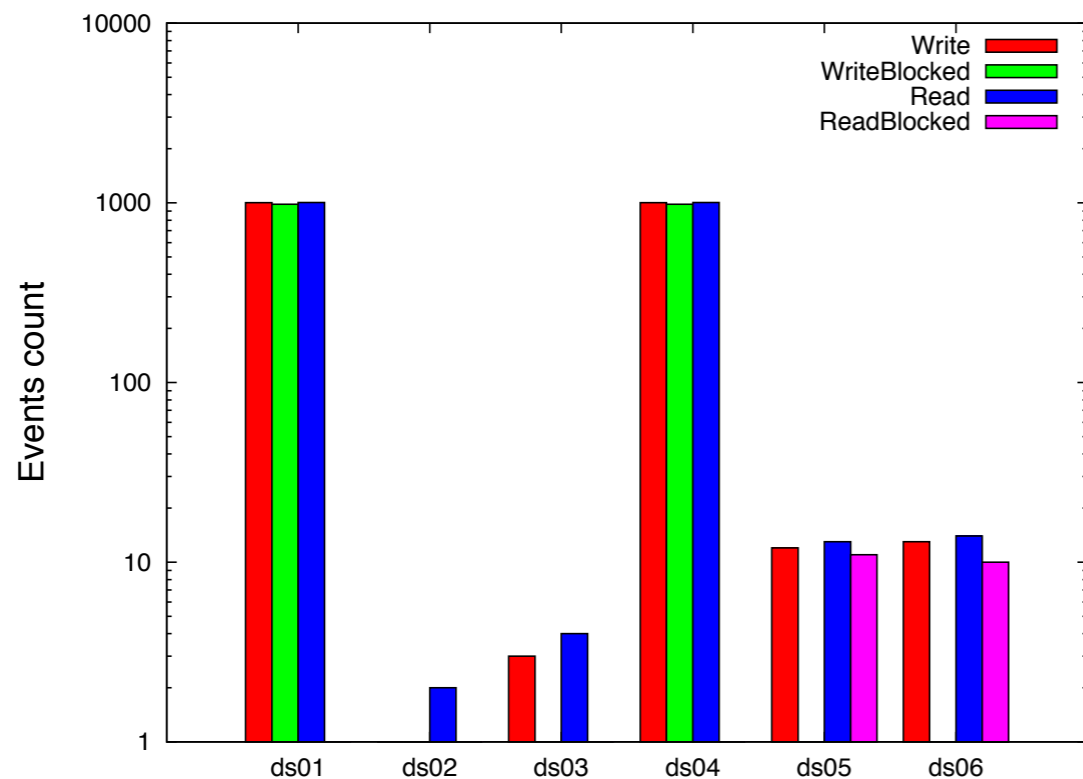
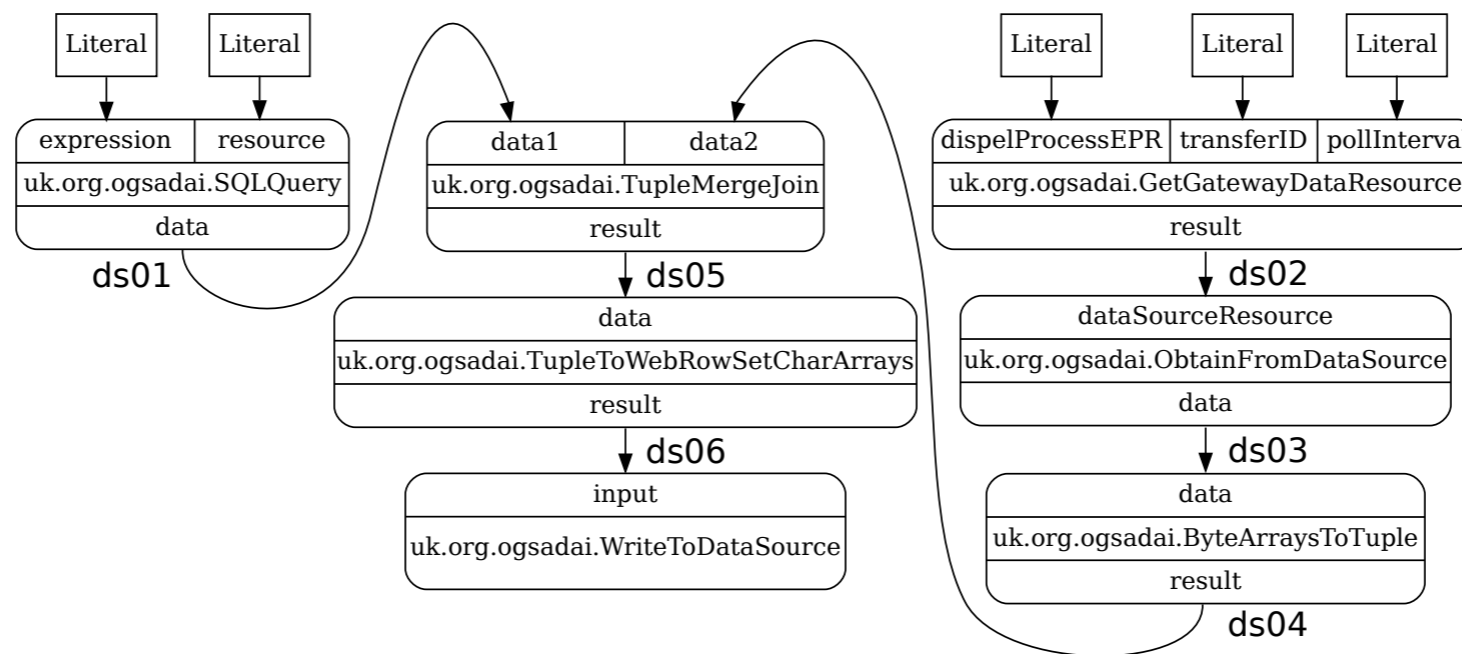
If you can not **measure** it, you can not **improve** it.



William Thomson

Phase 1: Measurement Framework & PDB





Chee Sun Liew, Malcolm P. Atkinson, Radosław Ostrowski, Murray Cole, Jano I. van Hemert, and Liangxiu Han. Performance database: capturing data for optimizing distributed streaming workflows, Phil. Trans. R. Soc. A, 369 (2011), pp. 3268– 3284.

Lessons learned from the first 2 phases

- There are PEIs that have to be assigned at a particular location (data source, capability restriction).
- There is a gap between the PEIs' unit processing cost (some PEIs' do not impose significant workload).
- Load balancing between the DIVMs is crucial.

Partitioning the PEIs into three subsets

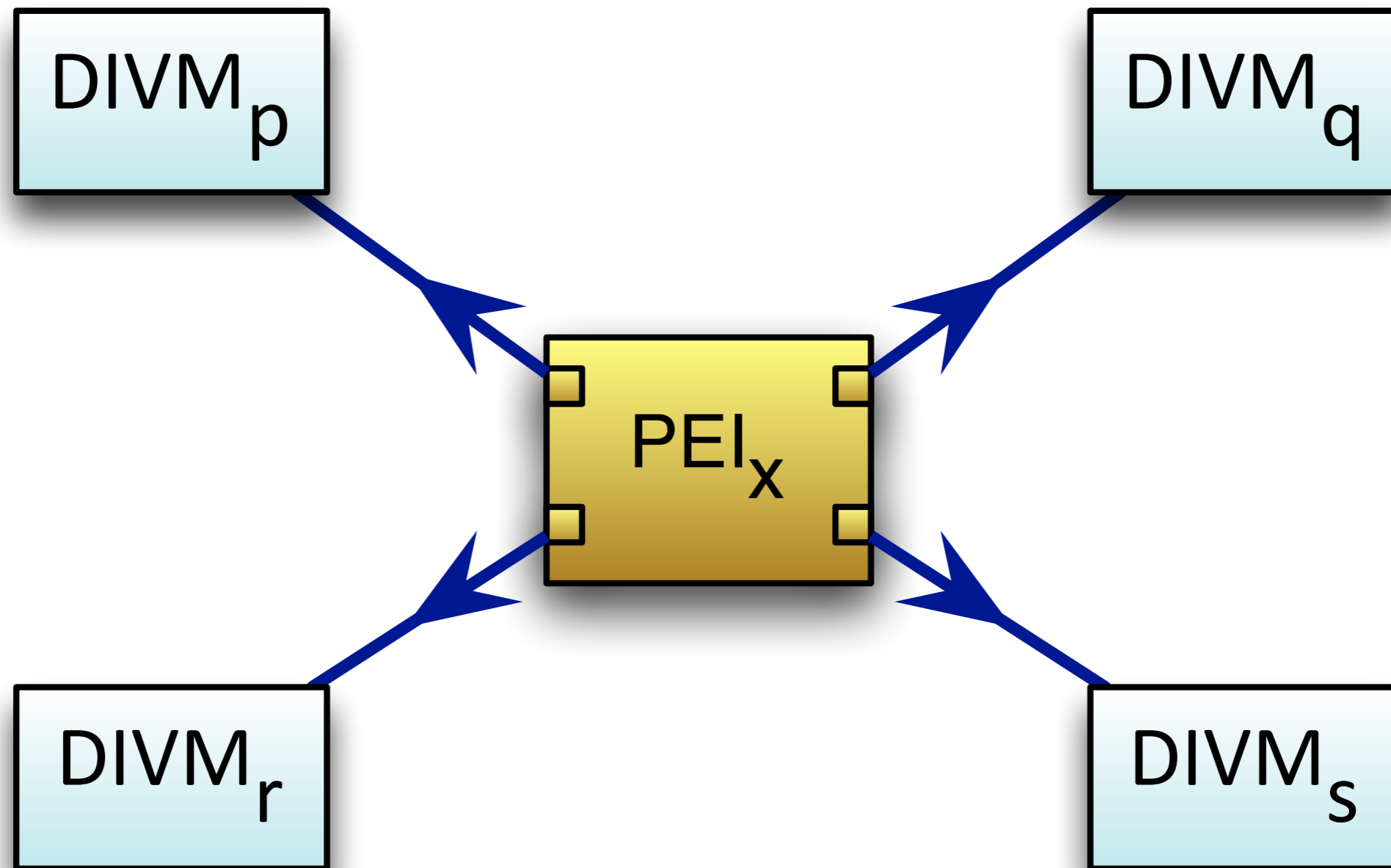
- The optimiser scans the set of PEIs and identifies two critical subsets to be allocated to DIVMs first:
 - **anchored PEI**: these are PEIs that have the **location** modifier and have been annotated with a stream literal expression defining data to which they are anchored.
 - **heavy PEI**: these are the PEI not in the above set, which have been identified as having **high unit processing costs**.
- All of the remaining PEIs are categorised as **light PEIs**.

That's been one of my mantras – **focus and simplicity**.
Simple can be harder than complex: you have to work
hard to get your thinking clean to make it simple.

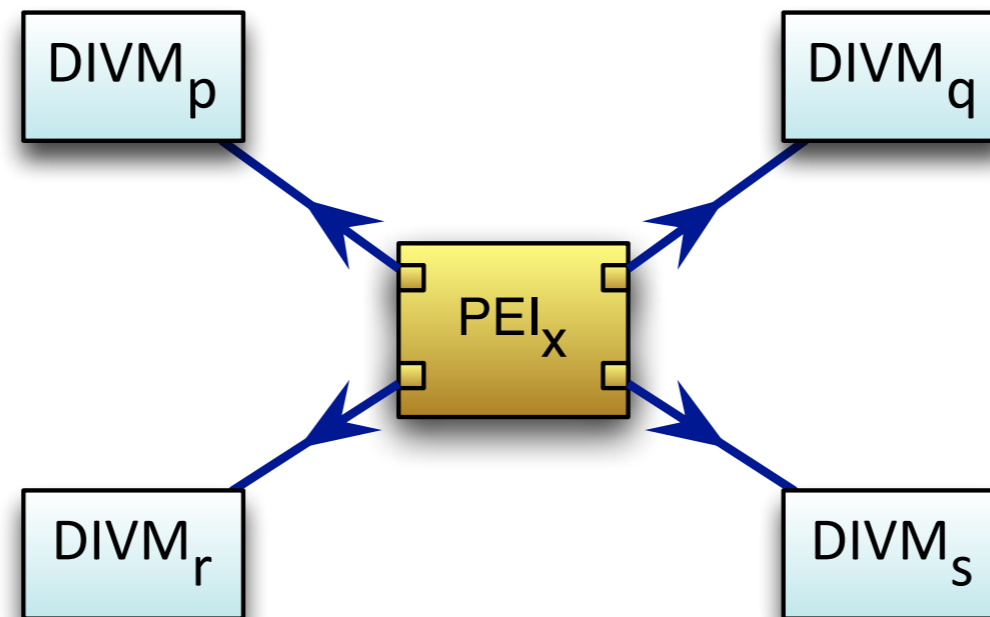


Steve Jobs

Conceptual model for assignment algorithm



Conceptual model for assignment algorithm



The force of assigning PEI_x onto $DIVM_q$, $F_{assign}(PEI_x, DIVM_q)$

stage 1

$F_{anchor}(PEI_x, DIVM_q)$

stage 3

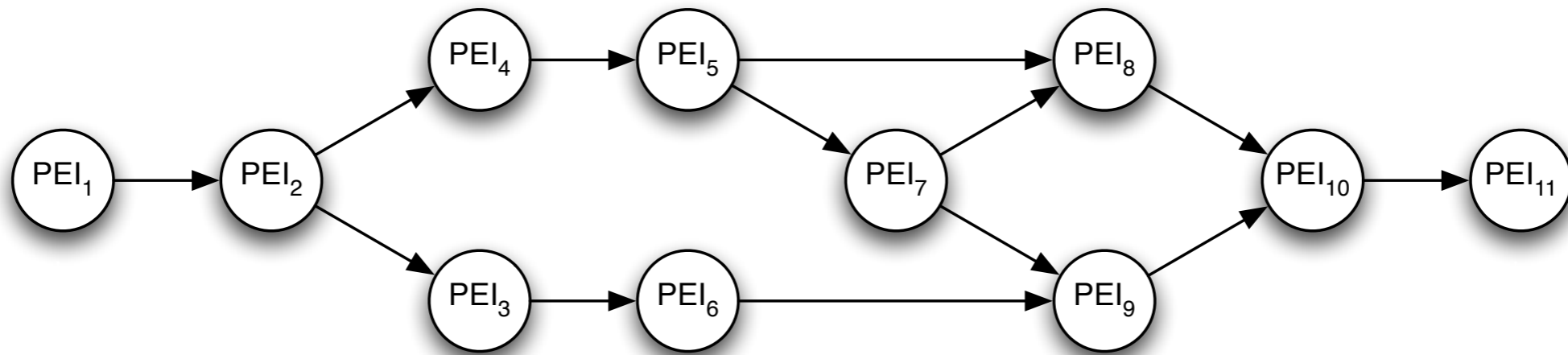
$+ F_{transfer}(PEI_x, PEI_y)$, where $PEI_x \Leftrightarrow PEI_y$ & PEI_y in $DIVM_q$

$F_{transfer}(PEI_x, PEI_z)$, where $PEI_x \Leftrightarrow PEI_z$ & PEI_z in $DIVM_p$ & $p \neq q$

stage 2

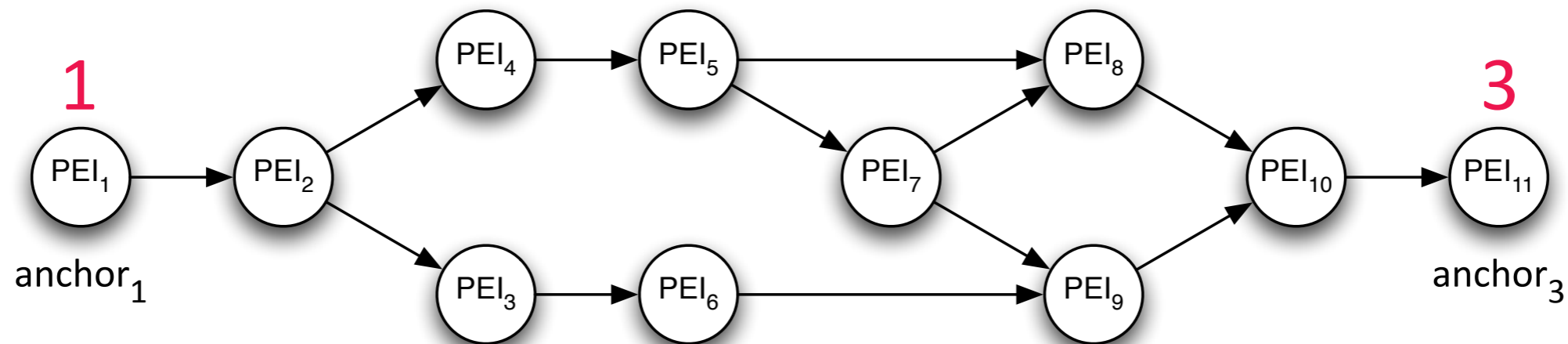
$F_{conflict}(PEI_x, PEI_y)$ where $PEI_x \Leftrightarrow PEI_y$ & PEI_y in $DIVM_q$

Three-stage assignment algorithm



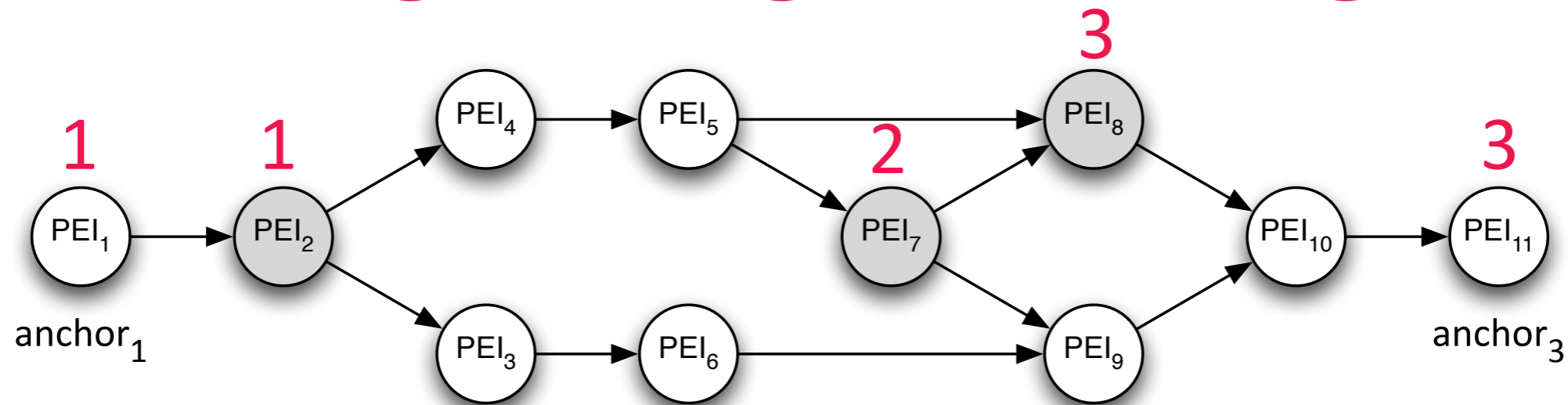
- Example:
 - number of PEIs = 11
 - number of DIVMs = 3 {DIVM₁, DIVM₂, DIVM₃}

Three-stage assignment algorithm



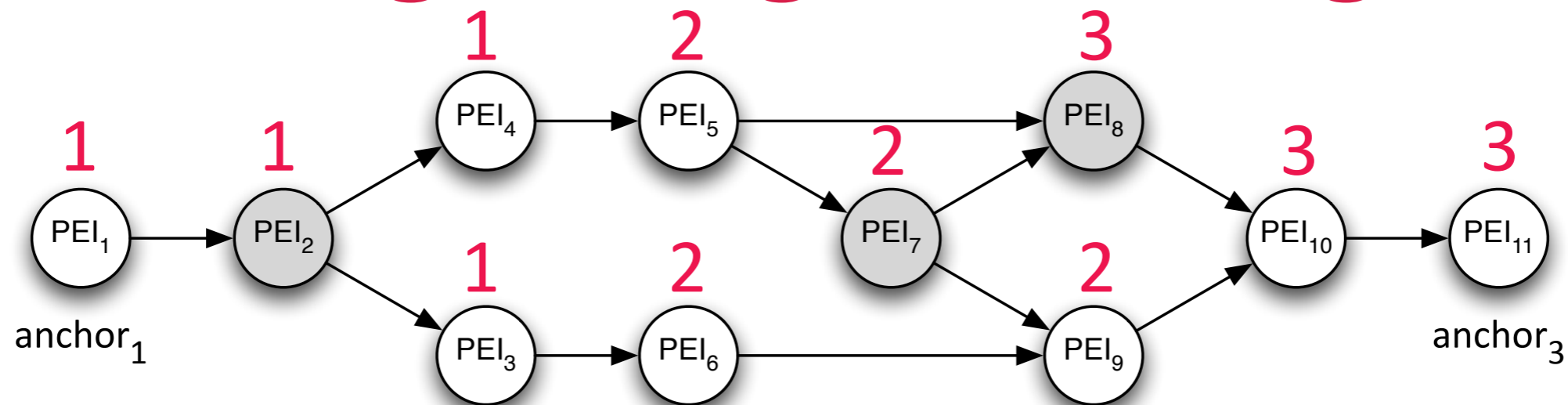
- Stage 1: Assigning anchored PEIs
 - read the anchor DIVM from the **locator** annotation
 - assign to anchor DIVM

Three-stage assignment algorithm

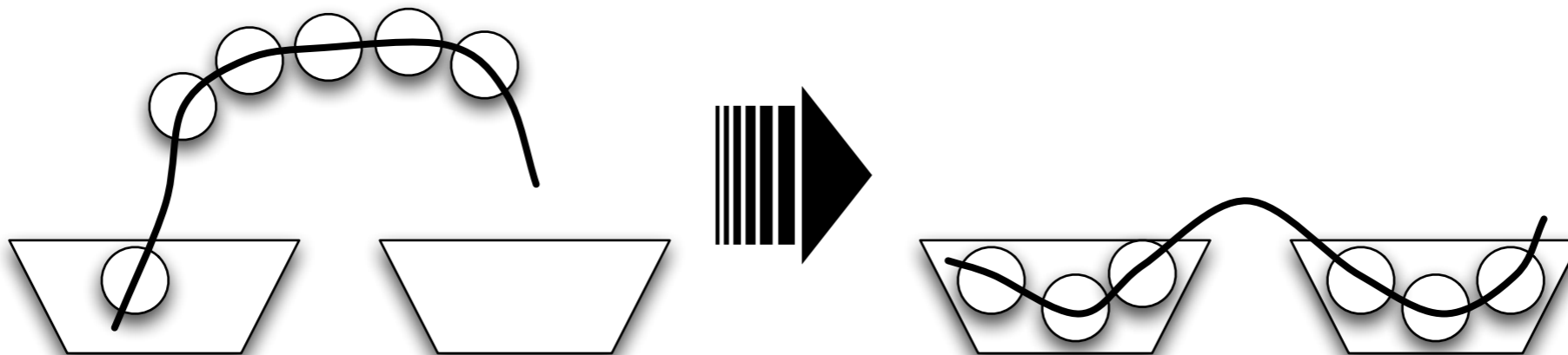


- Stage 2: Assigning heavy PEIs
 - sort unassigned PEIs by unit processing cost
 - select top- n PEIs, where n = number of DIVMs
 - assign top- n PEIs to DIVMs accordingly





Three-stage assignment algorithm



- Stage 3: Assigning light PEIs
 - think the data movement cost as the tension pulling the beads, slides beads into DIVMs
 - few methods are experimented

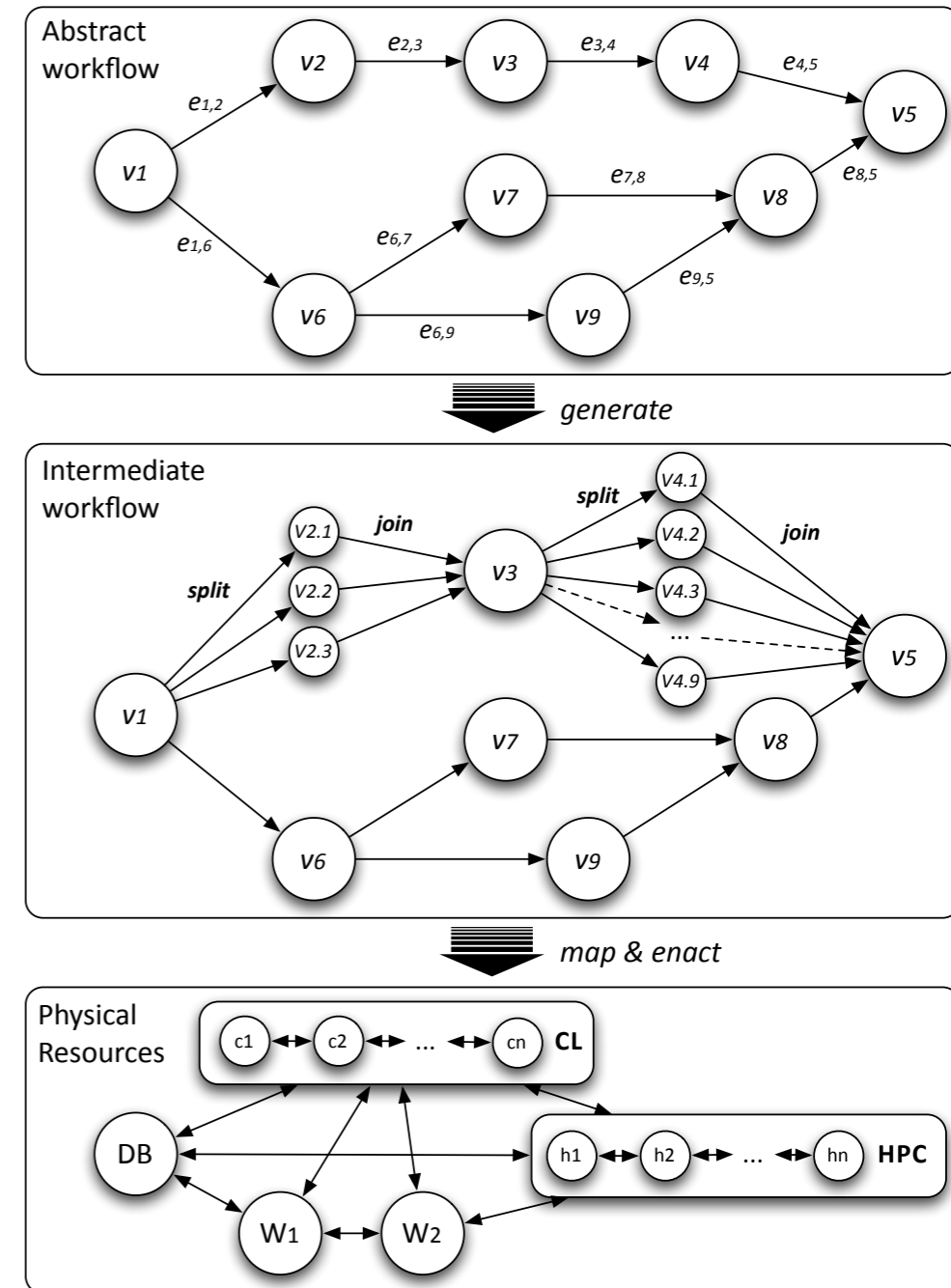


Phases in Optimisation work

- **Phase 0:** 
 - investigating the streaming behaviour of PEs
- **Phase 1:** 
 - building up measurement framework and PDB
- **Phase 2:** 
 - implementing assignment algorithm
- **Phase 3:** 
 - exploring graph transformation options

Exploring graph transformation options

- e.g. sub-graph substitution, parallelisation
- We have explored the potential of splitting pipeline workflow into parallel streams.
 - It's a “hand-crafted” solution to a life-science application.



Liangxiu Han, Chee Sun Liew, Jano I. van Hemert, and Malcolm P. Atkinson. A generic parallel processing model for facilitating data mining and integration. *Parallel Computing*, 37 (2011), pp. 157 – 171.

Ongoing work

- finish the prototype implementation
- organise test workloads from ADMIRE
 - life-science, seismology and astronomy
- run experiments and collect results
 - on EDIM1 & Eddie
- *[review and repeat the above three tasks]*
- write a paper
- write up my dissertation