Prediction of workflow execution time using provenance traces: practical applications in medical data processing

Hugo Hiden
Simon Woodman
Paul Watson

e-science central

Newcastle University

How long will my program take to run?

Part of a bigger picture

Can I repeat my results?

What are the implications of errors

How long will my program take to run?

What version of the program ran?

How was a result generated?

Provenance Research

- Used to answer these questions
- Important in scientific research
- Lots of work done to capture and represent provenance
- Active research area

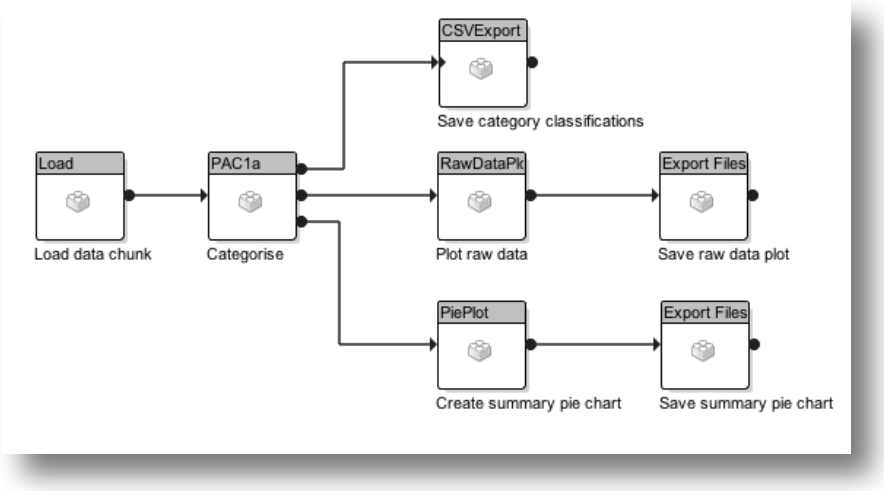
OPM ⟶ **PROV**

e-Science Central

- Source of all our provenance data
  - Platform used for many projects
- Repository of code and data
  - Users can add their own code
- Well instrumented and understood
  - Used to collect OPM
  - Now PROV
- Plenty of data sets
  - Diverse projects
  - Large applications
- Workflows for data processing

The workflow model

- Simple workflow implementation
  - Acyclic directed graph
  - Composed of connected "Blocks"
  - Deploys at reasonable scale in clouds



e-science central

Newcastle University

Modelling performance

- Execution time for a single block
  - Workflow is some combination of individual block models
- There should be some predictors:
  - The input data sizes
  - The configuration of the block
  - The machine it is running on
- The issues are:
  - What types of model are most appropriate
  - How accurate are they

Execution time of a block

time=f(input-size, block-code, block-settings, random-factors)

More data increases execution time

Each block has different characteristics, so a model is needed for each block

The configuration of the block instance can change behavior

Machine load, network traffic, hardware variations,...

A workflow is a connected pathway of blocks...

Requirements for a "real" system

- Proactively build models
  - In response to more data
  - When more blocks are added
- Select the most appropriate model
  - Pick based on best error
- Aim to always return some estimate
  - Mechanisms to return estimate if no models are available

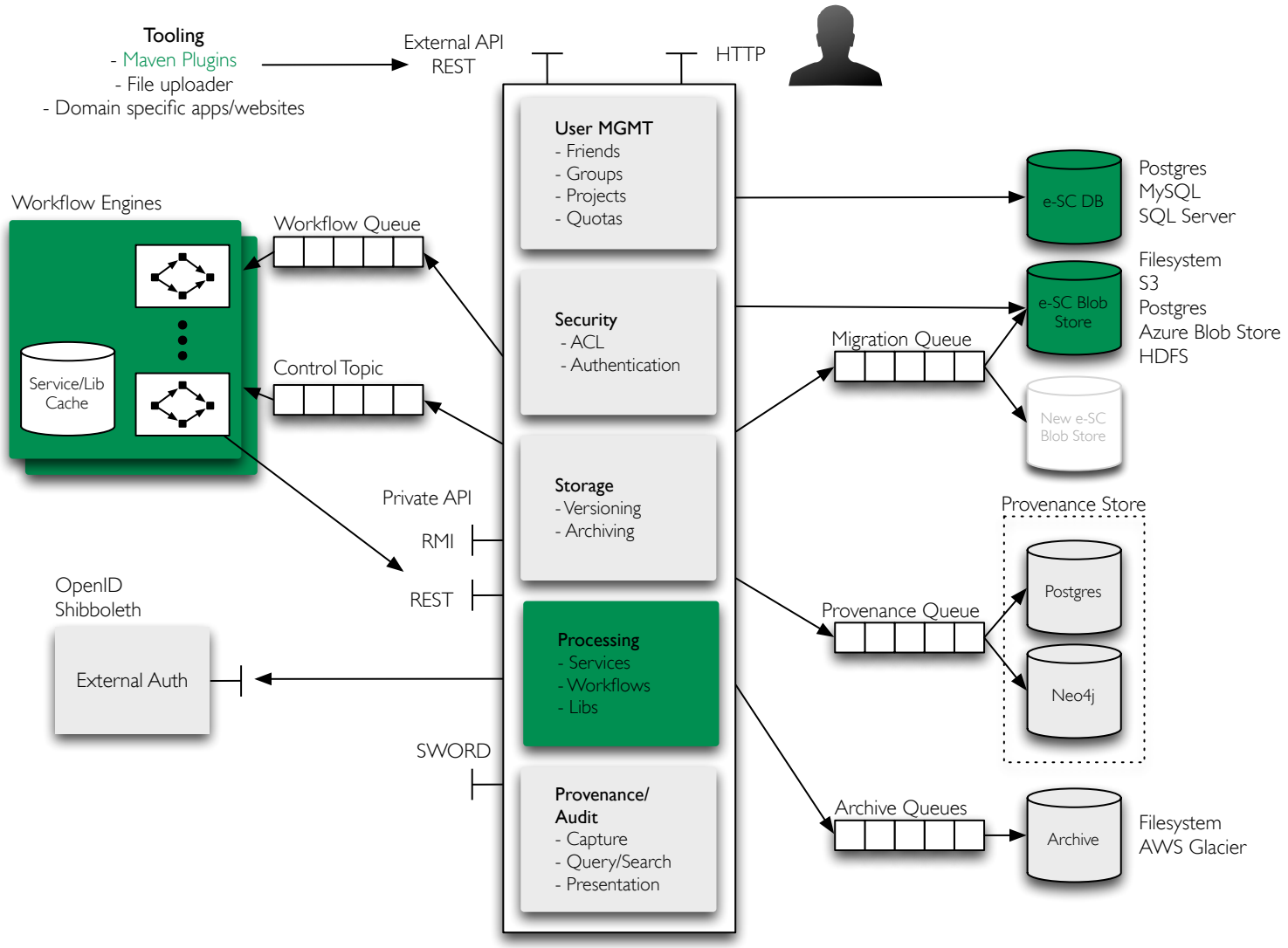
Complications

- Gathering data
  - Collect data "non-invasively"
- Model types
  - Different blocks display different characteristics
  - Different algorithms and versions
- Dynamic environment
  - New blocks being added
  - Block behaviour only becomes apparent as data is collected

Data collected via provenance

- Provenance collection already captures:
  - Data sizes
  - Code versions
  - Algorithm settings
- Extra instrumentation for
  - Block start and end times
  - Number of concurrent workflows
  - CPU / Memory usage

e-SC Architecture

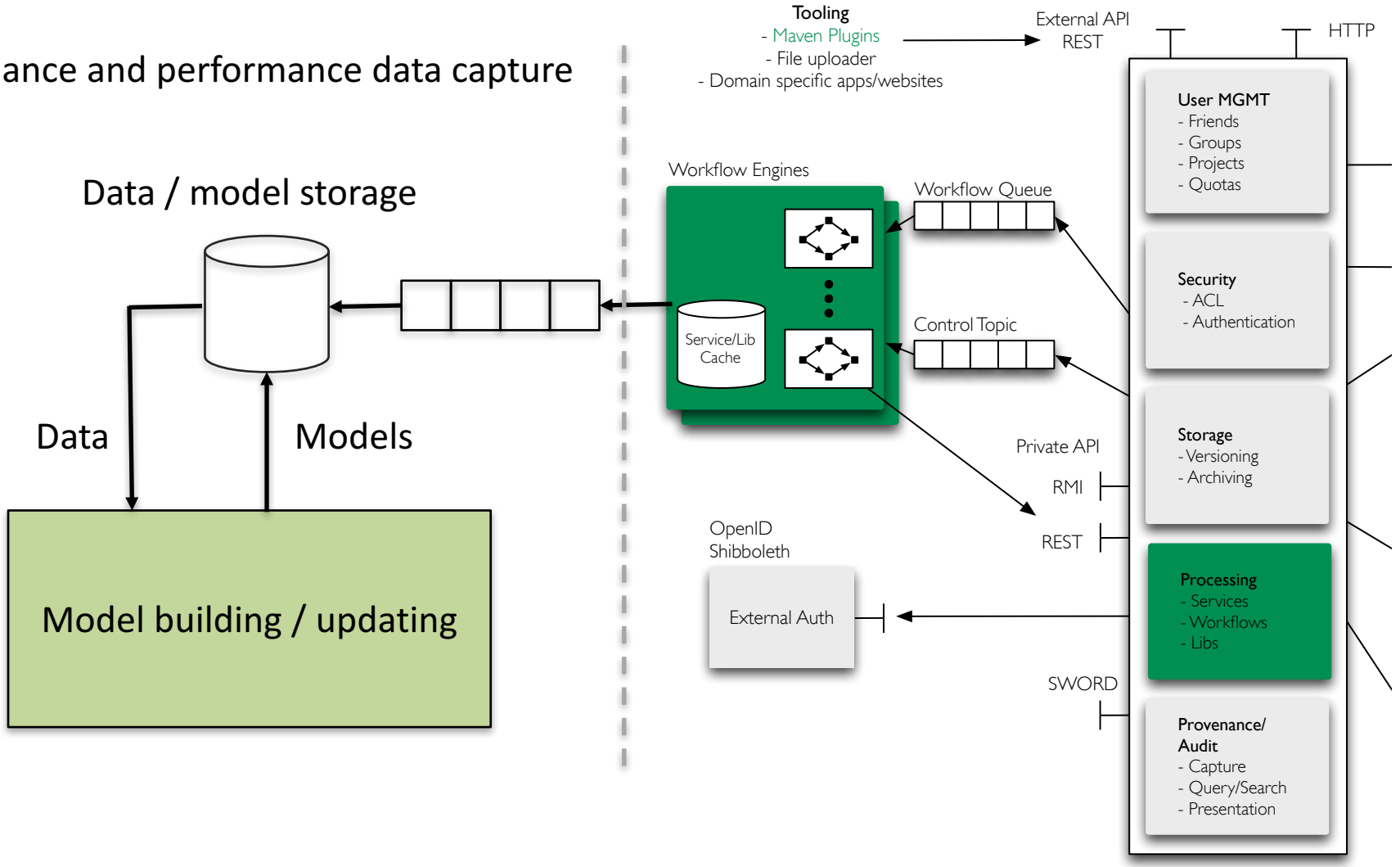


e-science central

Newcastle University

Data capture architecture

Provenance and performance data capture



e-science central

Newcastle University

Data collected

- Each execution of a block creates a single data point:

ID, Version	Setting_1, Setting_2, Memory Use, Input_size	Duration, Output_size
ID, Version	Setting_1, Setting_2, Memory Use, Input_size	Duration, Output_size
ID, Version	Setting_1, Setting_2, Memory Use, Input_size	Duration, Output_size

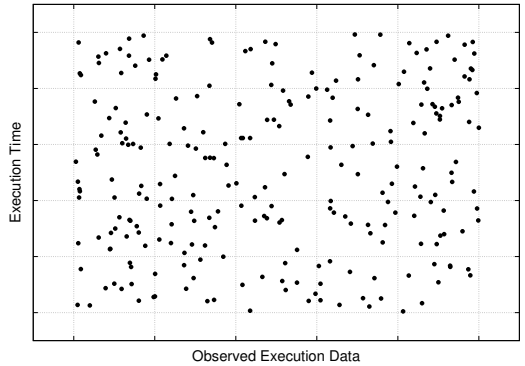
Identifying data

Model X data

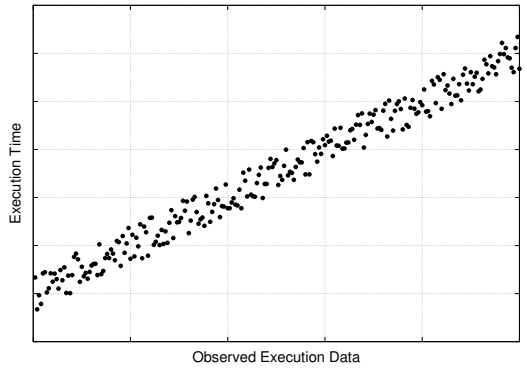
Model Y data

Block models

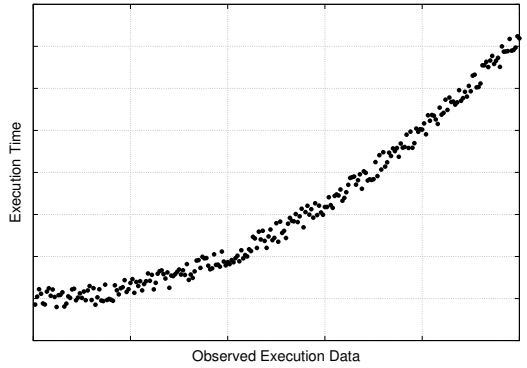
Blocks may exhibit very different behaviors depending on their implementation details / configuration



No relationship



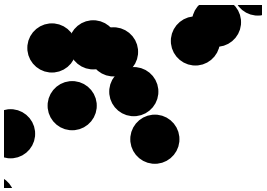
Linear relationship



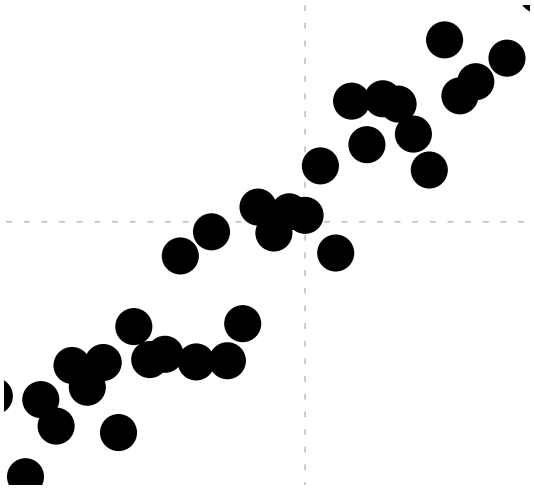
Non-linear relationship

Selecting the most appropriate model

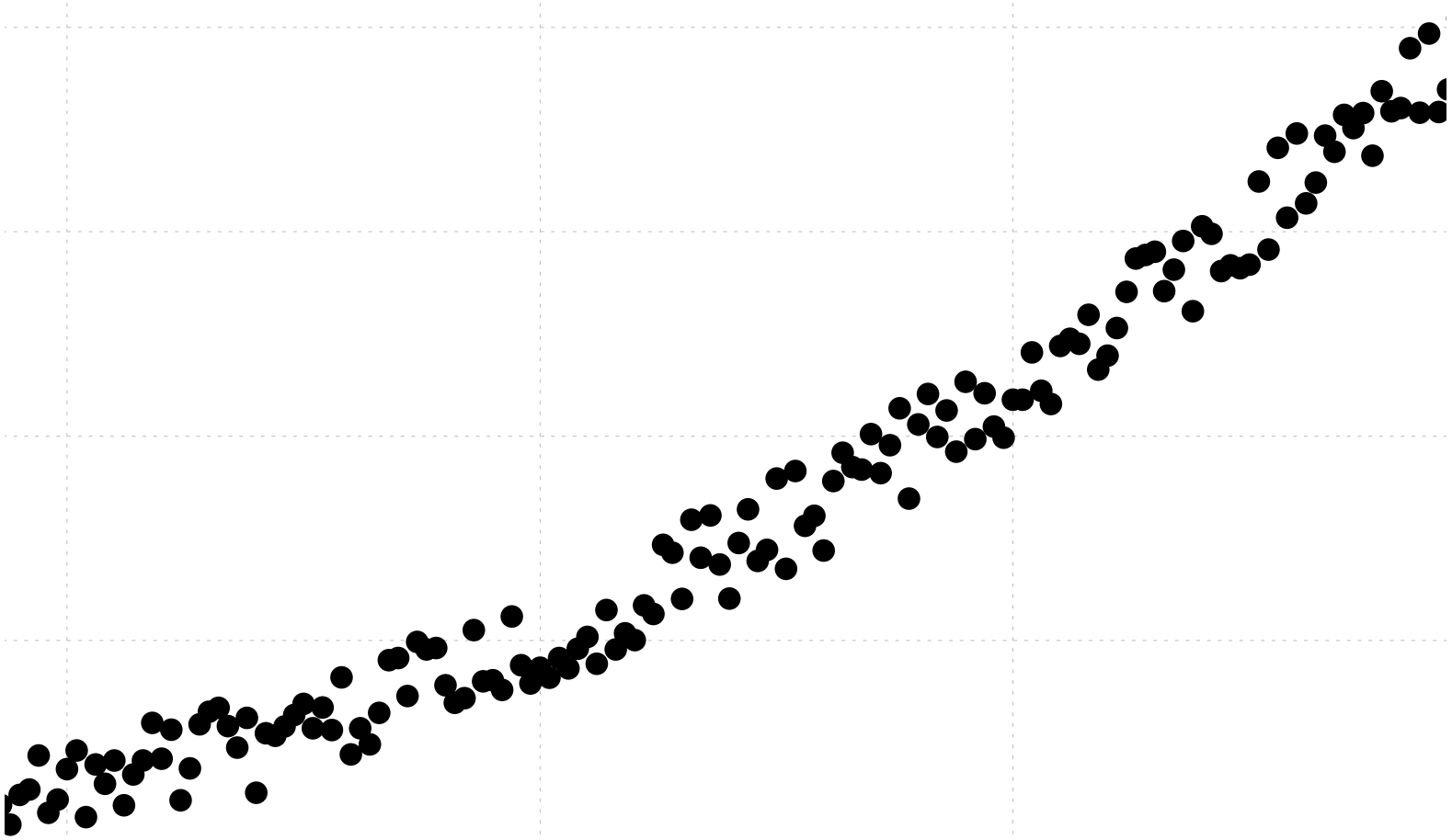
Selecting the most appropriate model



Selecting the most appropriate model



Selecting the most appropriate model



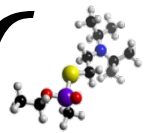
Dynamic model updating

- Impossible (difficult) to know what the best model will be
  - Gathering more data may change our view
- Need to implement model updating
  - Models can be rebuilt and replaced on the fly
- Return best available estimate at a given time
  - This may improve

"Panel of experts" pattern

- Maintain a suite of different models
  - Rebuild them all when new data arrives
  - Use the best one until the next update
- Drug modelling project:

Quantitative Structure Activity Relationship

$$Activity \approx f(\ \text{  }\ )$$

Model fallbacks

- What happens if there is no model?
  - Still want to return *something*
- We used the following logic:
  - Use version agnostic model
  - Use average execution time of block
  - Use average execution time of all blocks
- This will always return some prediction as long as a single block of any type has executed

Medical data processing

- Measure acceleration in 3-axes
  - Typically 100Hz
  - Worn for 2 weeks
  - Analyse sleep patterns, general activity levels etc
- Data collected and analysed
  - Clinicians view results and modify exercise regime
  - Collections of 100k data sets (24TB)

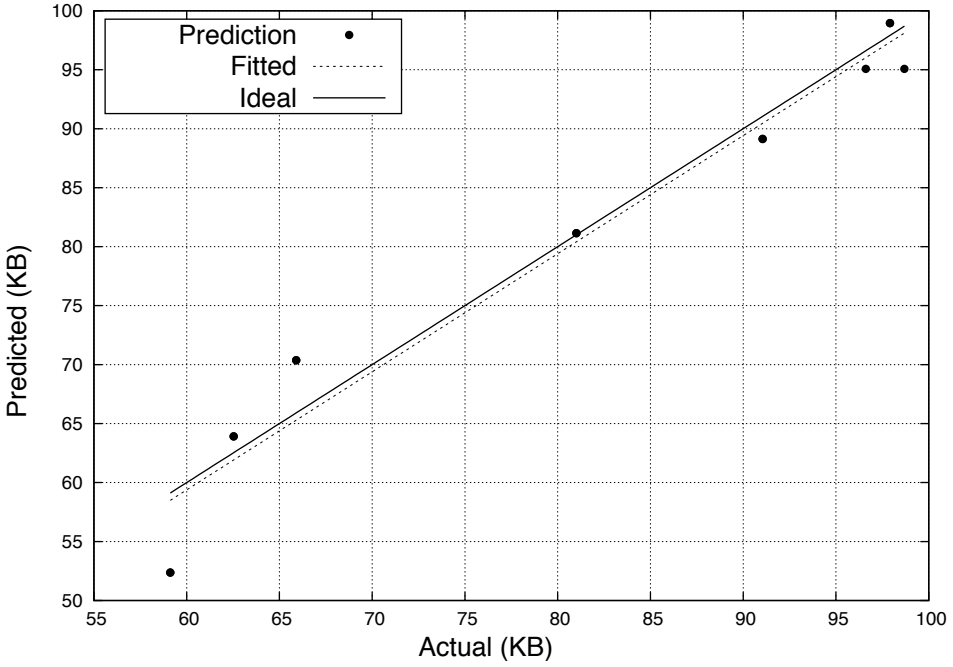
Wrist worn accelerometers

e-science central

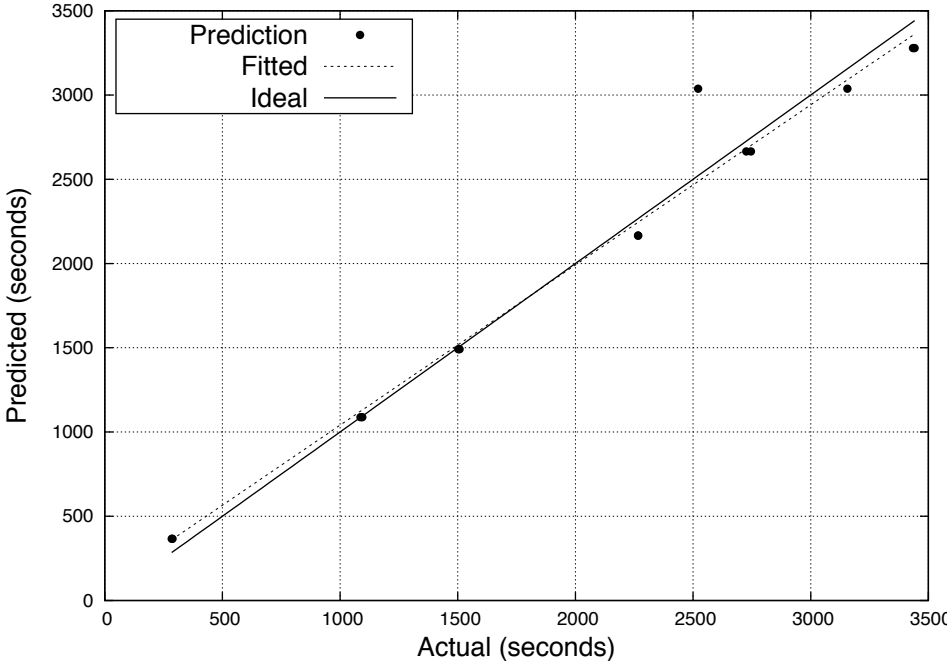
Newcastle University

Results

Physical Activity Classification (PAC1)



Output size model



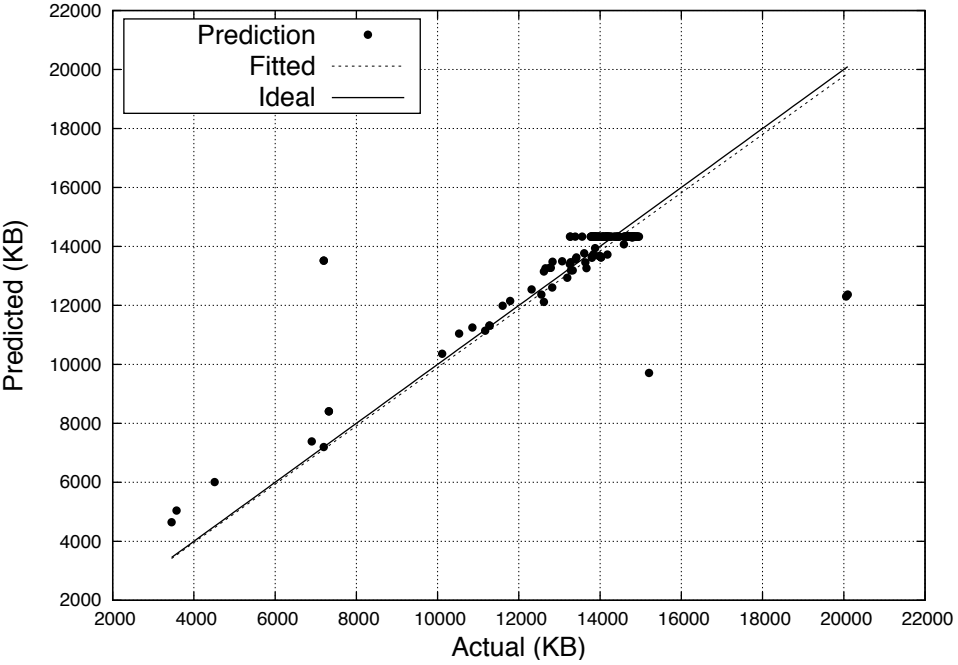
Duration model

e-science central

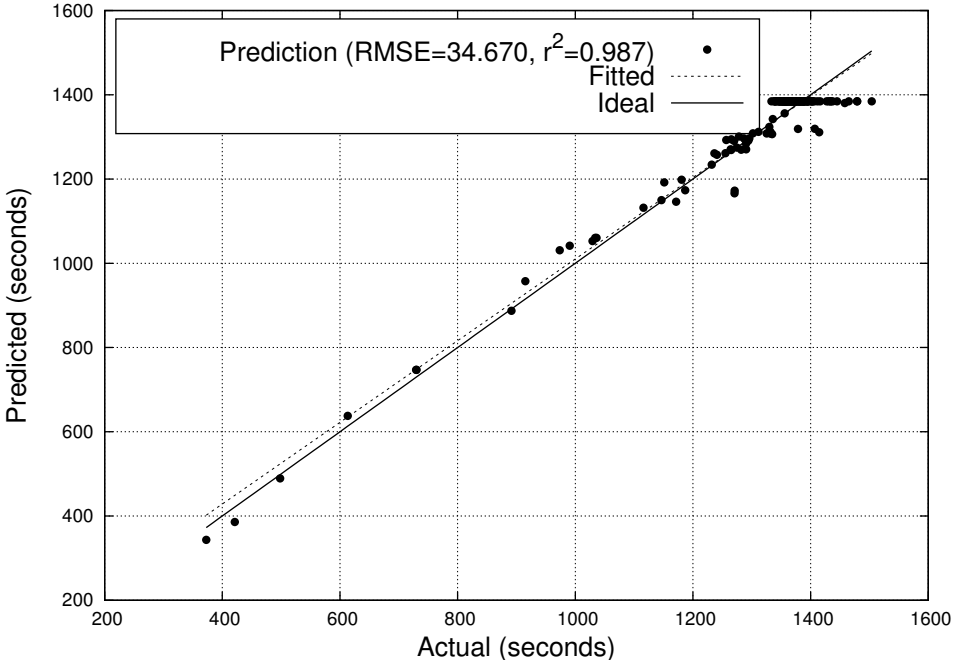
Newcastle University

Results

GGIR GENEActiv processing



Output size model

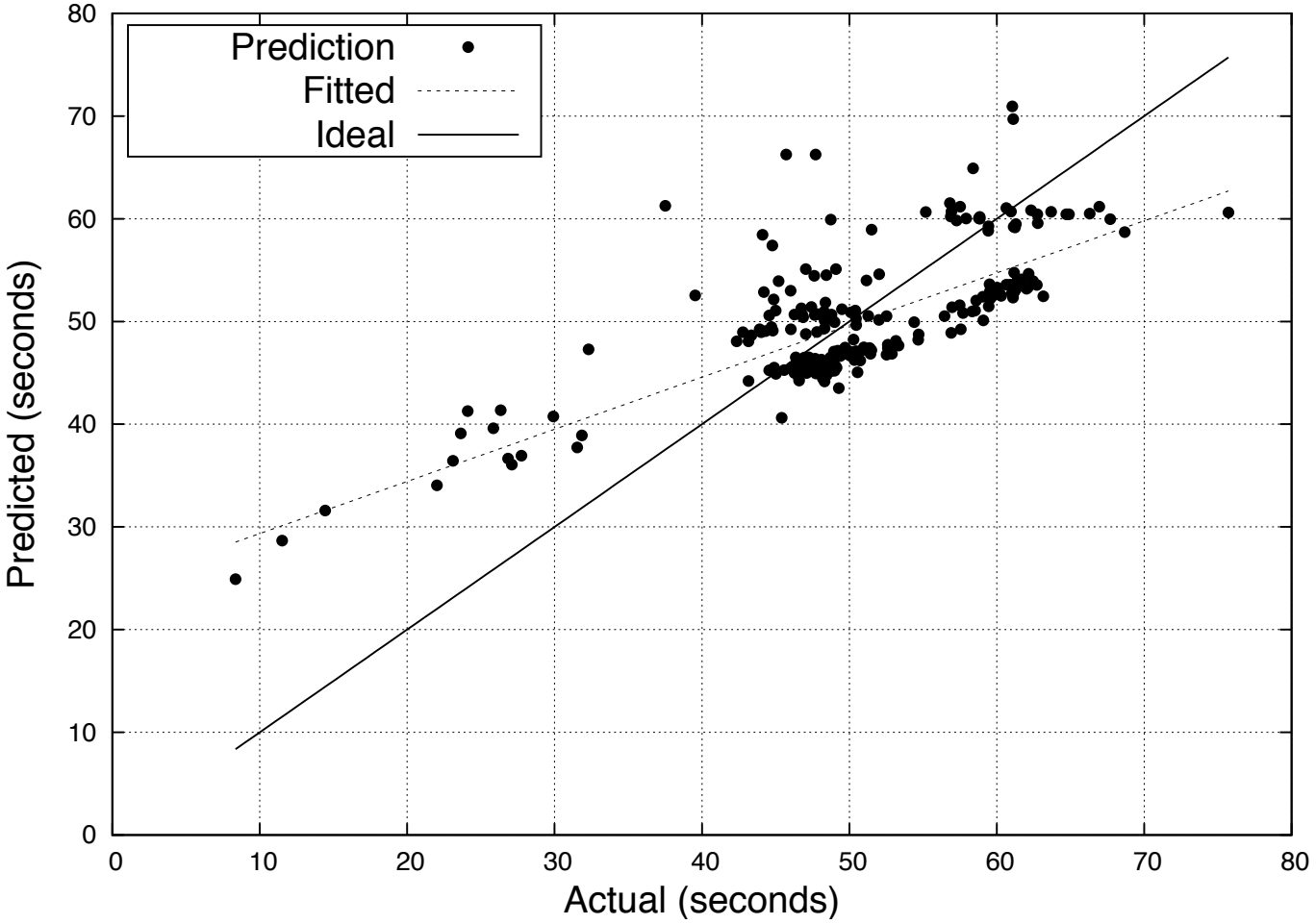


Duration model

e-science central

Newcastle University

Not always successful



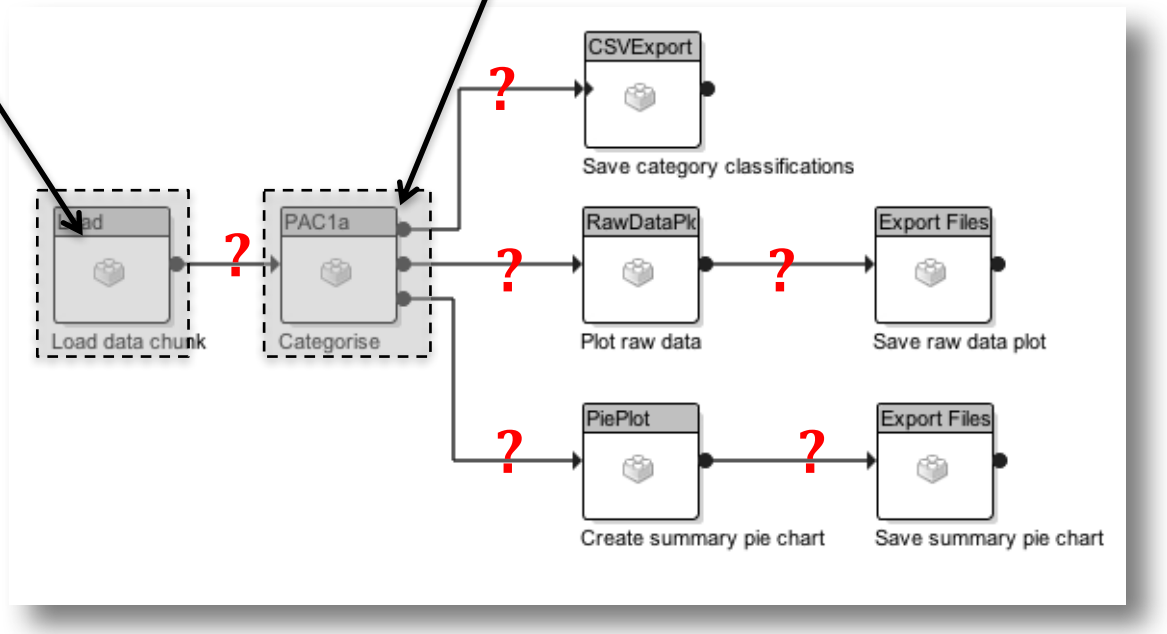
Predicting Workflow duration

Modelling is complicated by connected nature of workflow

All data for model readily available...

... not the case here

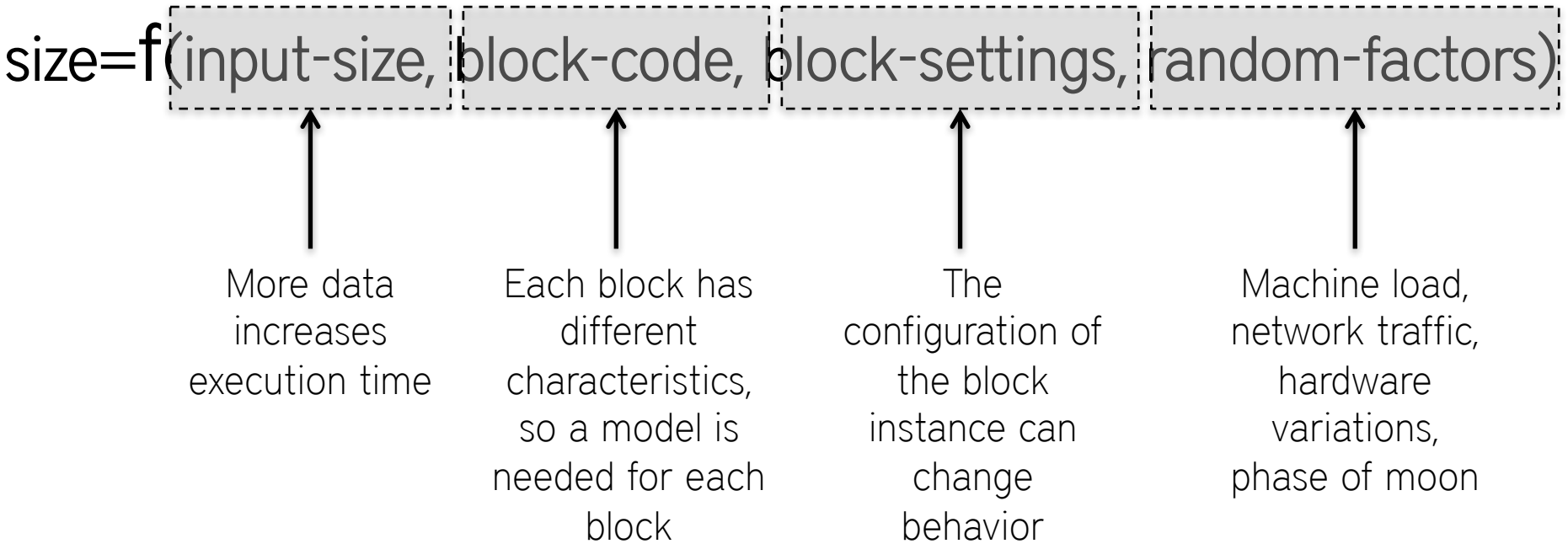
? how big are the intermediate data transfers



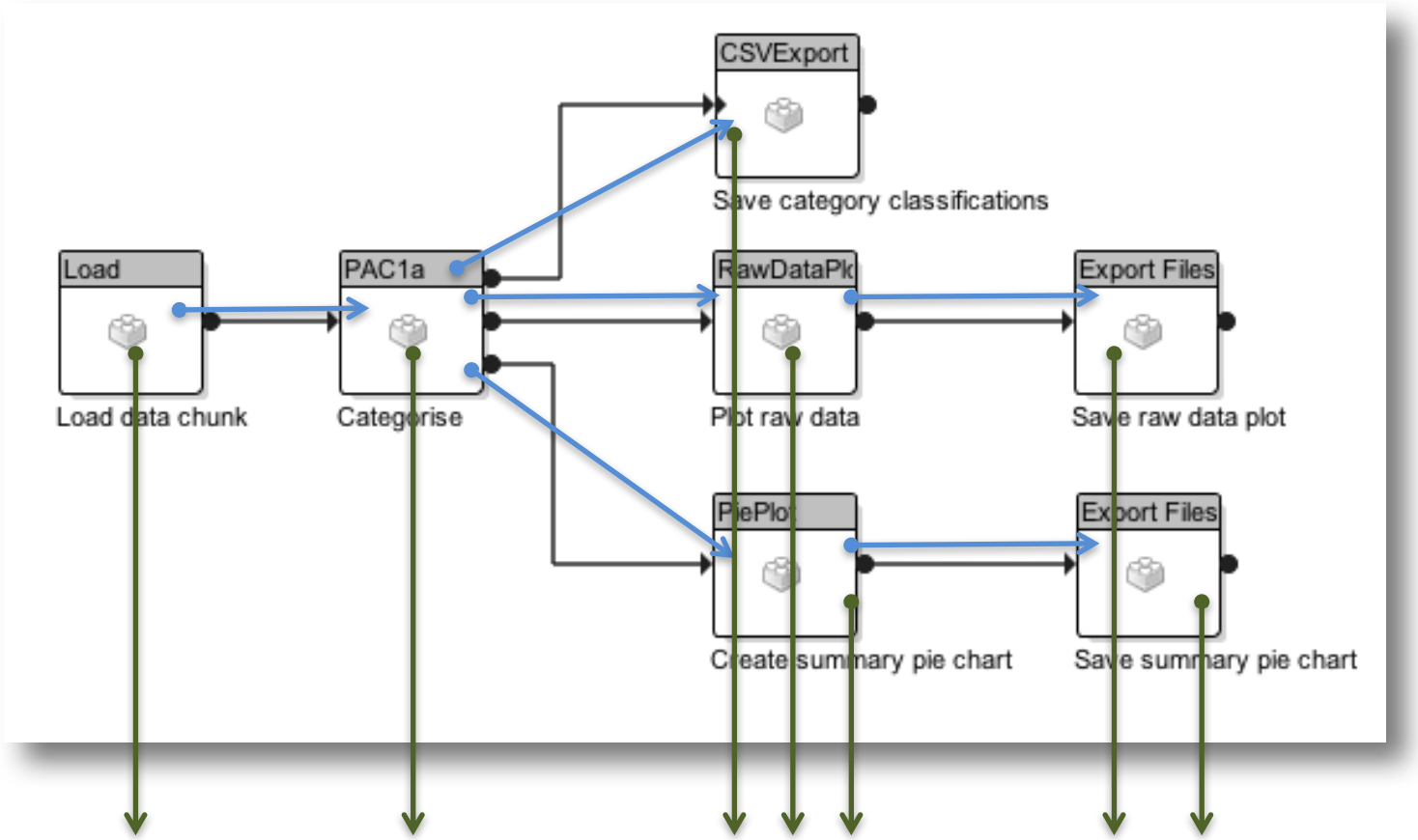
e-science central

Newcastle University

Data volume produced by a block



Modelling total execution time



Execution time = Sum(block predictions)

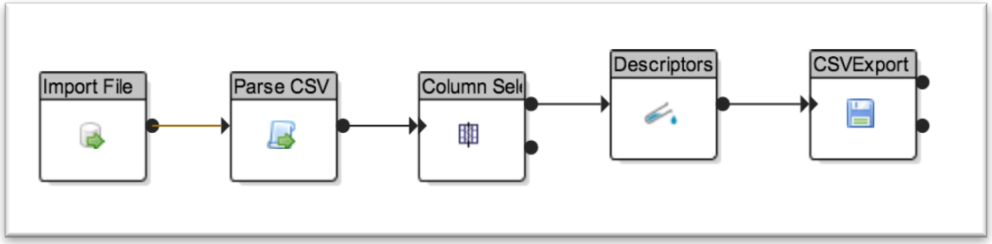
e-science central

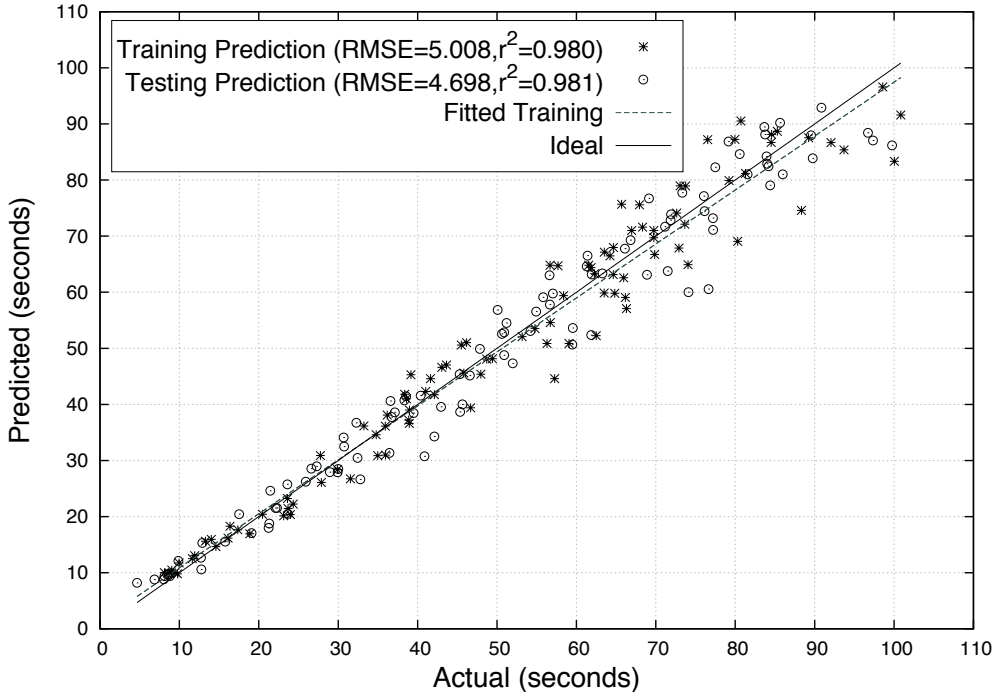
Newcastle University

Results

Chemical property modelling

- Models built for each individual block
- Prediction generated by propagating size predictions





e-science central

Newcastle University

Modelling workflows: caveats

- Much harder to model workflow duration
  - Propagation of errors
- Works for simple workflows
  - Rapidly fails for larger workflows
- Possible solutions
  - More data collection
  - Model groups of blocks
  - Build models of whole workflows

Conclusions

- Extended provenance capture to build predictive models
  - Asynchronous collection of data and model building
- Demonstrated it is possible to model block execution time
- Show it may be possible to combine predictions to estimate workflow execution time
  - Large workflows / poor block models are issues