

Predicting job run time of production jobs in ILCDIRAC

Erik Hidle
Stephane Poss

July 28, 2011

Content

- 1 Introduction
- 2 Background and motivation
- 3 Problem description
- 4 Methodology/Solution
- 5 Evaluation and Results
- 6 Conclusion
- 7 Further work

Introduction

- ILCDIRAC is used for Grid submission, jobs are submitted to ILCDIRAC which is responsible for executing it on the LCG, finally makes the results available for the user.
- In this project I've used Case Based Reasoning to predict job running time of job submission to ILCDIRAC.
- Job running time is useful for users and can potentially be used to increase the throughput of the system.

Background and motivation

- First: After job submission to ILCDIRAC, the user do not know when the results can be expected. For production jobs, which often contains several thousands of jobs, this can take several weeks! The system only takes care of executing the jobs (eventually) but can currently not provide any time estimate.
- Second: Currently users have to specify the amount of CPU time requested for jobs they submit to ILCDIRAC. This is a job the users tend to perform poorly (based on experience). This feature is used to make a reservation of the given resource which becomes responsible for executing the job. If this estimate is underestimated, it may result in the job being killed before the execution is completed. If it is overestimated it may result in the computing resource remaining idle after job has finished execution.

Problem description

- Provide a system that give job running time for production jobs in ILCDIRAC. The system will provide users with time estimates for when their jobs can be expected.

Project

- Started with a literature review: Many systems use job history to predict job running time for new jobs.
- Found several examples of systems that store executed jobs with inputs and their running time and use this to predict running time of new jobs.
- One of the system found employ a problem solving methodology called Case Based Reasoning. This system was used as a model and adapted to the ILCDIRAC context. I call it PREDATOR.

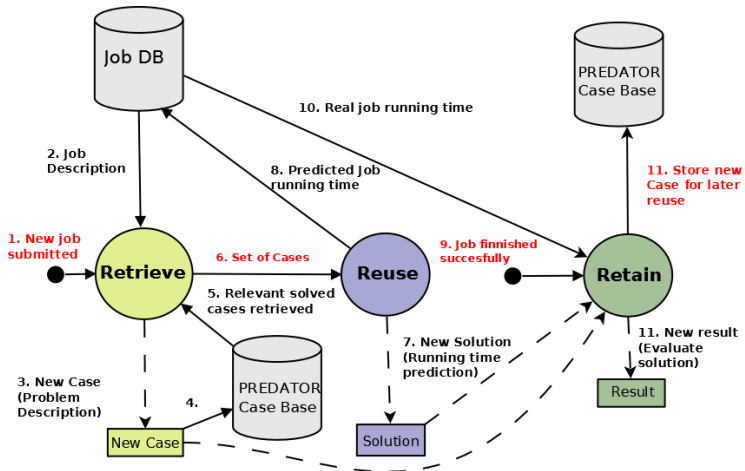
Very briefly: Case Based Reasoning

- A problem solving methodology that solves a new problem by adapting previously solved problems to the new problem.
- Three structures: A Case (problem description), Solution and Result
- The case describes the problem, the solution describes the necessary steps and possibly how the solution was created
- The result is an evaluation of the solution applied to the real world.
- The solved case with related Solution and Result is stored in a Case Base, for later reuse

Case Based Reasoning in ILCDIRAC

- The Case is constructed by retrieving job input features from the job database
 - ILCVersion
 - InputData
 - Number of events
 - etc.
- The Solution is the average of the real run time for a set of retrieved cases
- The Result is an evaluation of the Solution after the real job running time is known.

PREDATOR and ILCDIRAC



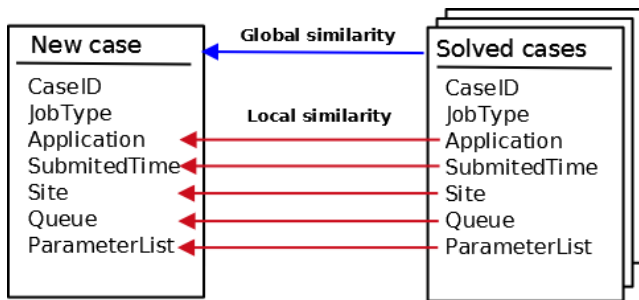
Predicting job run time in PREDATOR:

- When a new job arrives the system: get the job description from the job database and create a new case
- Compute the distance from the new case to all past solved cases.
- Select the K closest cases
- For the selected cases, retrieve the related Result object and obtain the real running time. Compute a statistical average of this values. Use the average as the predicted run time.
- When the real run time is known, retrieve it from the job database, create a Result object, evaluate the solution and insert it into the Case Base.

How is cases compared?

- Local similarity: defines similarity between two cases feature value.
- Global similarity: defines similarity between two cases by combining their local similarity.
- Example: Compare the Application name of the two cases.

Similarity between two cases



Similarity between two cases contd.

- The local similarity function depends on the feature representation.
- Symbolic: DIRAC Version, Owner, Site, Host
- Numeric: SubmissionTime, NumberOFEvents
- Returns a value between 0 and 1, 1 being most similar and 0 least similar.

Local Similarity functions

$$\text{NumericSim}(a_v, b_v) = 1 - \frac{|a_v - b_v|}{|\text{Max}(v) - \text{Min}(v)|} \quad (1)$$

$$\text{SymbolicSim}(a_v, b_v) = \begin{cases} 1 & \text{if } a_v = b_v \\ 0 & \text{if } a_v \neq b_v \end{cases} \quad (2)$$

Global similarity functions

$$GlobalSim(A, B) = \sum_{v=1}^V \omega_v LocalSim(a_v, b_v) \quad (3)$$

Evaluation

- PREDATOR have not been integrated in ILCDIRAC
- A part of PREDATOR was implemented for evaluation
- The evaluation was performed on a set of historical jobs retrieved from the job database
- Used Cross Validation on the retrieved set (training set) in order to have statistical independent measurements. (See next figure)
 - 1 Validation set
 - 2 Training set

Cross validation

Round 0



This is the initial set, partitioned by production.

Round 1



The validation set is selected from one random production.

The training set is sampled randomly from all other productions.

Round 2



Once a set has been selected it is removed from the initial set. Thus the new validation and training set are selected from the remaining points.

Round 3



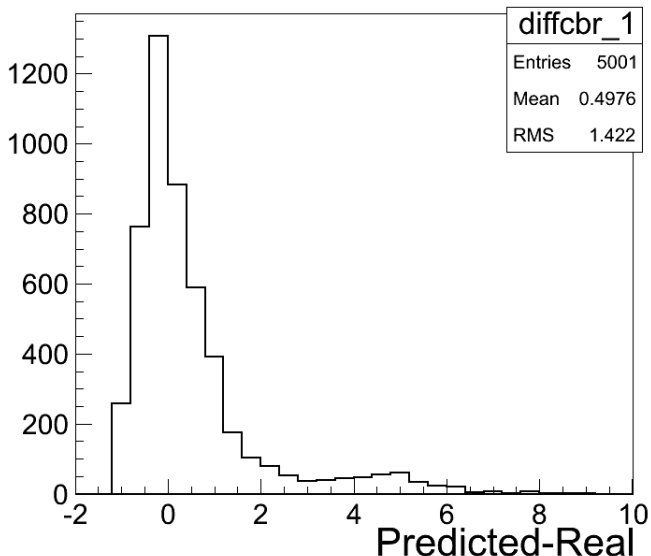
Round N



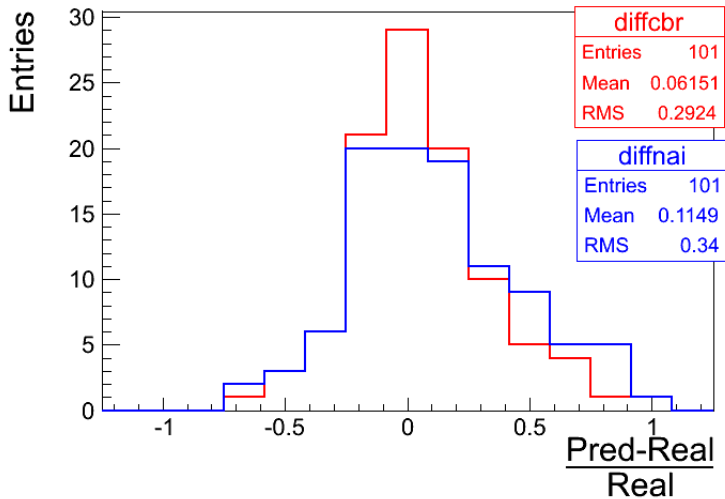
Evaluation scenarios

- 1. Predict the first job in a new production
- 2. Predict the Nth job of a given production. Given N-1 jobs from that production in the example set. Compare the results with the average of the running time of all jobs in that production (Naive).

Result 1



Result 2


















Conclusion

- 1 Job history can be used to predict job run time for production jobs in ILCDIRAC
- 2 PREDATOR produce significantly better results than a Naive approach.
- 3 We showed that PREDATOR is able to extract knowledge from known productions in order to predict job run time for new productions.

Further work

- Integrate PREDATOR with ILCDIRAC
 - Implemented part reused as a Service
 - Need a system to notify Predator of: New jobs arriving, Jobs finishing
 - Database for Case Base
 - More details in the report

Job Monitor

	JobId	Status	MinorStatus	Site ▼	Owner
<input type="checkbox"/>	3127165	 Failed	Application Finished With Errors	LCG.Weizmann.il	sposs
<input type="checkbox"/>	3127175	 Running	Job Initialization	LCG.UKI-LT2-IC-HEP.uk	jstrube
<input type="checkbox"/>	3127182	 Failed	Application Finished With Errors	LCG.RAL-LCG2.uk	jstrube
<input type="checkbox"/>	3127186	 Failed	Application Finished With Errors	LCG.RAL-LCG2.uk	jstrube
<input type="checkbox"/>	3127173	 Running	Job Initialization	LCG.RAL-LCG2.uk	jstrube
<input type="checkbox"/>	3127166	 Failed	Application Finished With Errors	LCG.QMUL.uk	sposs
<input type="checkbox"/>	3127169	 Failed	Application Finished With Errors	LCG.QMUL.uk	sposs
<input type="checkbox"/>	3127164	 Done	Execution Complete	LCG.QMUL.uk	sposs
<input type="checkbox"/>	3127179	 Waiting	Pilot Agent Submission	LCG.QMUL.uk	jstrube
<input type="checkbox"/>	3127178	 Waiting	Pilot Agent Submission	LCG.QMUL.uk	jstrube
<input type="checkbox"/>	3127180	 Waiting	Pilot Agent Submission	LCG.QMUL.uk	jstrube
<input type="checkbox"/>	3127181	 Waiting	Pilot Agent Submission	LCG.QMUL.uk	jstrube
<input type="checkbox"/>	3127163	 Done	Execution Complete	LCG.QMUL.uk	sposs
<input type="checkbox"/>	3127184	 Failed	Application Finished With Errors	LCG.QMUL.uk	jstrube
<input type="checkbox"/>	3127185	 Failed	Application Finished With Errors	LCG.QMUL.uk	jstrube