Modeling Job Lifespan Delays in Volunteer Computing Projects

Trilce Estrada, Michela Taufer
University of Delaware
{estrada, taufer}@udel.edu

Kevin Reed
IBM
knreed@ibm.us.com

Abstract

Volunteer Computing (VC) projects harness the power of computers owned by volunteers across the Internet to perform hundreds of thousands of independent jobs. In VC projects, the path leading from the generation of jobs to the validation of the job results is characterized by delays hidden in the job lifespan, i.e., distribution delay, in-progress delay, and validation delay. These delays are difficult to estimate because of the dynamic behavior and heterogeneity of VC resources. A wrong estimation of these delays can cause the loss of project throughput and job latency in VC projects.

In this paper, we evaluate the accuracy of several probabilistic methods to model the upper time bounds of these delays. We show how our selected models predict up-and-down trends in traces from existing VC projects. The use of our models provides valuable insights on selecting project deadlines and taking scheduling decisions. By accurately predicting job lifespan delays, our models lead to more efficient resource use, higher project throughput, and lower job latency in VC projects.

1. Introduction

Volunteer Computing (VC) projects harness resources owned by ordinary people that volunteer processing and storage resources to computing projects. Performance in VC projects is measured in terms of throughput (number of valid results delivered to the scientists) and latency (job lifespan). BOINC (Berkeley Open Infrastructure for Network Computing) is a well-known middleware that supports the VC paradigm [1]. BOINC projects are VC projects that: (1) derive their power from heterogeneous computing resources (called hosts) with varying levels of availability and reliability [7]; (2) execute jobs with varying lengths and degrees of sensitivity to errors; and (3) have different requirements in terms of job replications (i.e., number of replicas or job instances whose results are compared for agreement before being delivered to scientists).

The path leading from a job generation to the validation of its replicated results (and delivery to scientists) is characterized by delays hidden in the job lifespan:

- Generation delay: from job generation to job instance generation. This time is normally very small, of the order of seconds, and therefore not significant for the job lifespan.
- Distribution delay: from job instance generation to job instance distribution to hosts. New jobs reside in BOINC databases and are distributed when hosts apply for computation based on scheduling criteria that can cause substantial distribution delays.
- In-progress delay: from job instance distribution to result collection. Instances are distributed in bundles. Once on the assigned host, instances are considered in-progress. In-progress instances are not necessarily in execution: they can be waiting for execution or waiting to have their results delivered to the server.
- Validation delay: from result collection to result validation. As results are reported back to the server, they are not validated immediately. A certain number of instances (quorum) have to produce matching results before considering the associated job as valid. This may require the generation of additional instances until the quorum is reached.

For performance reasons, it is vital to estimate the delays associated to the job lifespan. Unfortunately estimating these delays is difficult because of the dynamic behavior and heterogeneity of VC resources. Accurate models of the job lifespan help to estimate these delays and ultimately support higher project throughput and lower job latency in VC projects.

The attraction and retention of volunteers in VC projects is another important issue that can be affected by delays in job lifespan. Volunteers expect to be acknowledged in terms of *"credits"* or points that are assigned proportionally to the amount of computation provided. Credits are only awarded after the validation of the job results. Many volunteers place a great value on the credits and expect a prompt credit assignment. Awarding credits with some delay have a negative impact on the volunteers' attitude to donate their resources. The reduction of job lifespan helps VC projects in assigning the credits faster and retaining volunteers.

This paper's contribution to solve the problem of accurately predicting job lifespan and its delays is twofold. First, we evaluate several prediction models using different probabilistic methods. The models provide VC projects with a quantitative approach to predict upper time bounds for job lifespan delays. Then, we validate our prediction models against empirical data, i.e., traces from existing VC projects, and select the most accurate models for delay predictions. Because the selected models rigorously map trends in VC traces, they can be used (1) to support a more efficient use of volunteers' resources, (2) to increase project throughput, (3) to decrease job latency, (4) and ultimately to retain a larger number of volunteers. In particular, our models, when integrated in VC scheduling policies, help in deciding optimal deadlines (e.g., before considering a job instance timed-out) and scheduling decisions (e.g., when to generate new job instances and assign them to hosts).

The rest of this paper is organized as follows: Section 2 presents a short overview of BOINC, the VC traces, and the VC projects used in this paper. Section 3 shows through examples why it is important to model and predict delays in VC projects. Section 4 describes the probabilistic methods that we use to model VC projects. Section 5 presents our predictions for four VC projects and present their validation. Section 6 discusses existing work in this field and compares it with our approach. Finally, Section 7 concludes the paper.

2. Background

2.1 BOINC

BOINC (Berkeley Open Infrastructure for Network Computing) [2] is an open-source system that harnesses the computing power and storage capacity of thousands or millions of PCs owned by ordinary people for large-scale scientific simulations. The computing resources available to a BOINC project are highly diverse: the hosts differ by orders of magnitude in their processor speed, available RAM, disk space, and network connection speed. Some hosts connect to the Internet by modem only every few days, while others are permanently connected. Computations may have varying bounds on completion time. BOINC projects are highly attractive for large-scale simulations because they can potentially sustain a very high processing rate (tens or hundreds of TeraFLOPs).

BOINC projects are organizations (typically academic research groups) that need computing power. Projects are independent; each operates its own BOINC server. A BOINC project comprises hundreds or thousands of independent jobs. For fault-tolerance and trust reasons, a job is replicated in several job instances that are distributed to different hosts. Each time a host requests for computation, BOINC uses a scheduling policy to build a package of job instances for that machine. More and less sophisticated scheduling policies have been implemented in several BOINC projects. These policies affect project throughput and job lifespan [2, 17]. Errors, or invalid results, cause the generation and distribution of new instances for the faulty job, which may result in a loss in throughput and latency.

2.2 Volunteer Computing Traces

BOINC traces describe the overall behavior of VC hosts, jobs, and job instances. Trace data includes information on: when a job is created; when its job instances are created, distributed, returned, and validated; status of job instances (i.e., waiting for distribution, error or success, valid or invalid, and timed-out); which job instances are assigned to a given host; what are the characteristics of the several hosts (e.g., OS, speed). A host community is composed of those hosts that contribute to a given project. Host communities continuously change over time and they can overlap partially in different VC projects.

2.3 Volunteer Computing Projects

In this paper we consider four BOINC projects and their traces collected over time. The projects are: two projects from the IBM initiative, World Community Grid (http://worldcommunitygrid.org), and two projects from Predictor@Home (P@H). World Community Grid (WCG) is an initiative supported by IBM that makes VC technology available to the public and not-for-profit organizations. WCG currently supports several VC projects. In our work we consider FightAIDS@Home and Genome Comparison. FightAIDS@Home searches for drugs to disable HIV-1 Protease. Because similar genes usually have similar functions, Genome Comparison identifies already studied genes and compares similarities with those we know nothing about. Predictor@home (P@H) is a BOINC project for large-scale protein structure prediction [16]. The protein structure prediction algorithm in P@H is a multi-step pipeline that includes: (a) a conformational search using a Monte Carlo simulated-annealing approach using MFold; and (b) protein refinement, scoring, and clustering using the CHARMM Molecular Dynamics simulation package. Table 1 summarizes the main features of these four projects and their heterogeneity in terms of length of the traces (days), size of the host community (number of hosts), job instance size (flop), error rates, and number of job instances

Table 1. Main features of the performance traces of the BOINC projects used in this paper

Project	Length of traces (days)	Number of hosts	Mean and Std CPU time per job(hrs)	Error rate	Number of job
WCG FightAIDS@Home	25	35583	$\mu = 7.7, \sigma = 4.6$	2%	136779
WCG Genome Comparison	50	30540	$\mu = 1.2, \sigma = 0.9$	1%	189182
P@H MFold	15	7810	$\mu = 2.1, \sigma = 1.4$	4%	162957
P@H CHARMM	4	5093	$\mu = 1.6, \sigma = 1.0$	15%	85722

per traces considered in this work.

3 Why Predicting Delays - Case Studies

Figure 1 shows examples in which delays in job lifespan ultimately affect the overall project throughput and job latency in VC projects. Figure 1.a shows an ideal case study in which the distribution delay, in-progress delay, and validation delay complete at approximately the same time for the three instances of a given job. Unfortunately this is not the general behavior of job instances in VC projects.

Figure 1.b shows a case in which the distribution delay of at least one instance (Instance 3) is much longer than the distribution delays of the other instances (Instance 1 and Instance 2). This can happen because the server is unable to distribute the three instances at approximately the same time, e.g., in case of homogeneous redundancy mismatching [17]. In addition to the increased distribution delay, this behavior affects the validation delay and ultimately the total job lifespan. If the server can predict the upper bound distribution delay of Instance 3, it can change at run-time the number of returned results used for validation (quorum) from three to two, preventing a final loss in project throughput.

Figure 1.c and Figure 1.d show two cases in which an instance times out (Instance 3). In both cases, the timed-out instance causes the generation of a new instance (Instance 4). A timed-out instance occurs when its result is not returned to the server within a time threshold defined by the project. Typically this time-out threshold ranges from a couple of days to a couple of weeks and most of the time is selected arbitrarily. In Figure 1.c the result validation can take place only when Instance 4 finishes, resulting in a longer validation delay. If the server can predict the in-progress delay, it can also prevent the long waiting delay for validation by accordingly setting the time-out thresholds lower, which ultimately prevent a loss in throughput. In Figure 1.d the validation takes place when the considered timed-out instance (Instance 3) returns its result. In this case, the project has wasted computing resources by generating a fourth instance (Instance 4) whose results are never used. Again the prediction of the upper bound in-progress delay can prevent the generation of Instance 4 and a more efficient use of the resources.

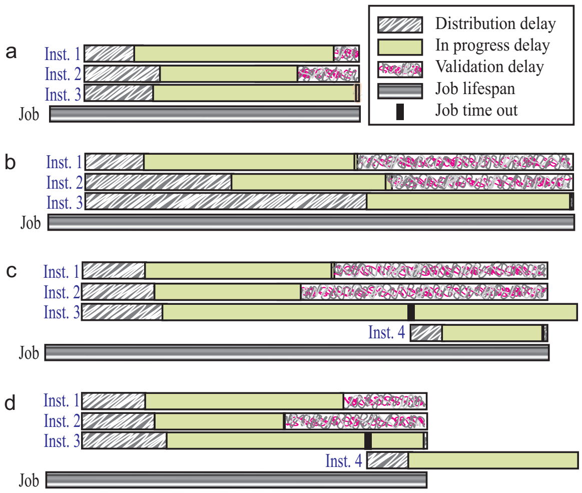


Figure 1. Characterization of job lifespans

The computing resources available in a BOINC project are heterogeneous: the hosts have different processors, memory, and network connections. Some hosts connect to a project via modem a few times per day, others are permanently connected. Every BOINC project has its own worker community, which is the set of active workers that donate computation for that specific project, with their own performance features. These features can change unpredictably and dynamically.

4. Methodology

4.1 Time Bounds

In this paper, we use statistical methods to model unknown job lifespan delays. In particular we use probabilistic density functions of the lifespan delays, i.e., time an instance is waiting for distribution, time an instance is in-progress, and time an instance is waiting for validation. We use these density functions to build cumulative distributions useful for scheduling decisions at the server level. Our approach uses statistical methods based on quantiles to estimate upper time bounds b on the q^{th} quantile with 95% confidence. This allows us to model and predict these three time bounds:

- b_{dist} or upper bound distributed delay,
- b_{prog} or upper bound in-progress delay, and
- b_{val} or upper bound validation delay.

4.2 Framework for Probabilistic Models

To build our probabilistic models, we use the traces of the four VC projects presented in Section 2. Figure 2 gives an overview of our method for building the models and predicting the bounds. First we build our models by sampling a data window from the traces of a VC project. We use the data to build a statistical model for each of the three job lifespan delays by using a distribution chosen among different types of distributions (modeling step). Then we calculate the quantiles and predict the time bounds b_{dist}, b_{prog}, and b_{val} for the given VC project and distribution (prediction step). Last but not least, we validate the models by sampling a new, consecutive data window from the VC traces (validation step). The validation phase includes the estimate of the prediction accuracy. For each VC project, the modeling, prediction, and validation phases are repeated across its traces. Figure 3 gives an overview of the sampling process for modeling and validating models for a given project. The traces are separated in time intervals; for each prediction, the two sampled windows (for modeling and validating) do not overlap. The modeling windows consist of a fixed number of job instance traces. The validation windows consist of a fixed amount time. For a new prediction, we slide both the modeling and validation windows of a fixed interval of time.

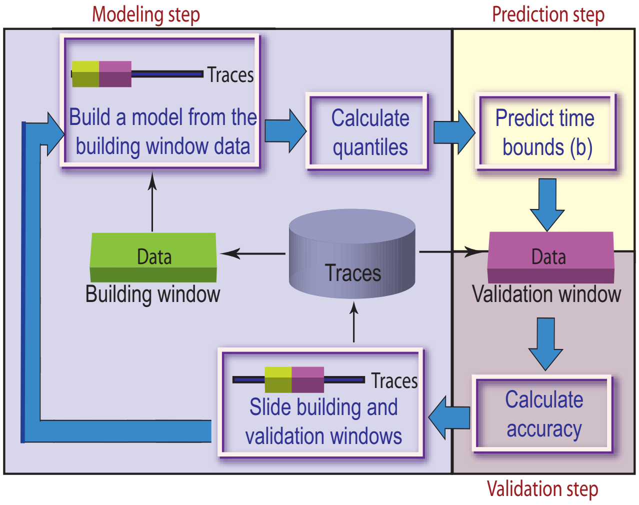


Figure 2. Our framework for probabilistic bound modeling

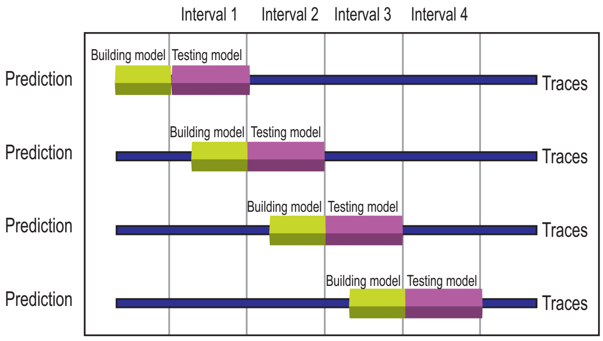


Figure 3. Time windows for modeling and validating models

4.3 Building the Model

To build our statistical models, we consider four parametric probabilistic distributions: Weibull, Levy, Gamma, and a mixture of Gaussian distributions. Table 2 presents the summary of the probabilistic distributions used. For each distribution, the table shows the probabilistic distribution function (pdf), the cumulative density function (cdf), and the list of parameters that need to be estimated for both pdf and cdf. In the table, α is a shape parameter, β is a scale parameter, $\widehat{\beta}$ is an inverse scale parameter, erf is the Gauss error function, k is the number of Gaussians, $\overrightarrow{\mu}$ is a mean vector, $\overrightarrow{\sigma}$ is a standard deviation vector, and $\overrightarrow{w}$ is a weight vector, all the vectors are of size k.

With the Weibull distribution, we change the scale of the data in the sampled window to a logarithmic scale and we adjust a straight line using the Least-squares algorithm. Once we have the line, we can compute the parameter α from the slope and the parameter β from the intercept. Figure 4 shows an example of our adjustment for the Weibull distribution for modeling the in-progress delay for the P@H - CHARMM project. For the Levy and Gamma distributions, we adjust the model using the maximum likelihood estimation (MLE) method [8]. For a mixture of Gaussian distributions [11] we use the expectation maximization algorithm (EM) [5] for the adjustment. Figure 5 shows an example of the model adjustment for the in-progress delay of the P@H - CHARMM project with the mixture of Gaussian distributions.

5. Results

As mentioned in Section 2, we consider four VC projects and their traces, i.e., FightAIDS@Home in World Community Grid (IBM), Genome Comparison in World Community Grid (IBM), MFold in Predictor@Home, and

Table 2. Summary of statistical methods

Name	Probability density function (pdf)	Cumulative distribution function (cdf)	Parameters
Weibull	$pdf(x;\alpha,\beta) = \left(\frac{\alpha}{\beta}\right)\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}$	$cdf(x;\alpha,\beta) = 1 - e^{-(x/\beta)^{\alpha}}$	α, β
Gamma	$pdf(x;\alpha,\widehat{\beta}) = x^{\alpha-1}\left(\frac{\widehat{\beta}^{\alpha} e^{-\widehat{\beta}x}}{(\alpha-1)!}\right)$	$cdf(x;\alpha,\widehat{\beta}) = \frac{\gamma(\alpha,x\widehat{\beta})}{(\alpha-1)!}$	$\alpha, \widehat{\beta}$
Levy	$pdf(x;\beta) = \sqrt{\frac{\beta}{2\Pi}}\left(\frac{e^{-\beta/2x}}{x^{3/2}}\right)$	$cdf(x;\beta) = erf(\sqrt{\frac{\beta}{2x}})$	β
Mixture of **Gaussians**	$pdf(x;\overrightarrow{\mu},\overrightarrow{\sigma},\overrightarrow{w},k) =$ $\sum_{i=1}^{k} w_i \frac{e^{-\left(\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)}}{\sigma_i\sqrt{2\|\Pi}}$	$cdf(x;\overrightarrow{\mu},\overrightarrow{\sigma},\overrightarrow{w},k) =$ $\sum_{i=1}^{k} w_i \frac{1}{2}\left(1 + erf\left(\frac{x-\mu_i}{\sigma_i\sqrt{2}}\right)\right)$	$\overrightarrow{\mu},\overrightarrow{\sigma},\overrightarrow{w},k$

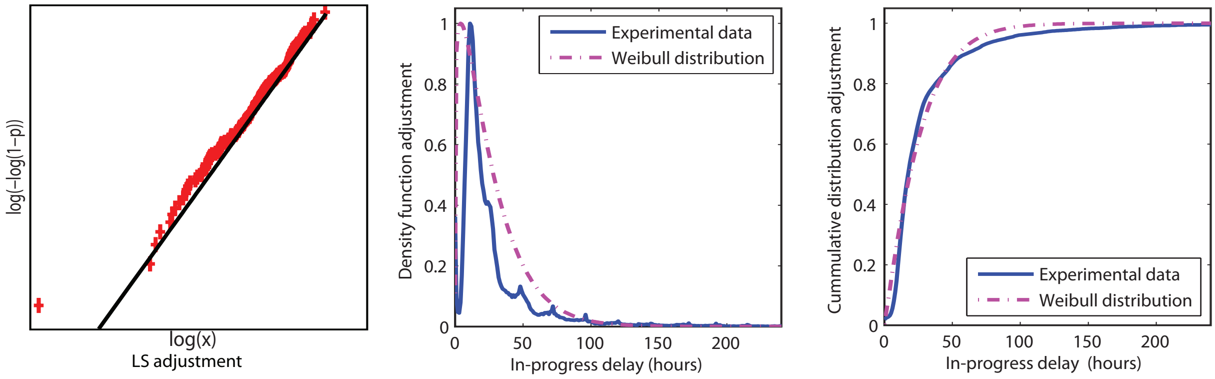


Figure 4. Model adjustment for P@H-CHARMM, in-progress delay, and Weibull distribution

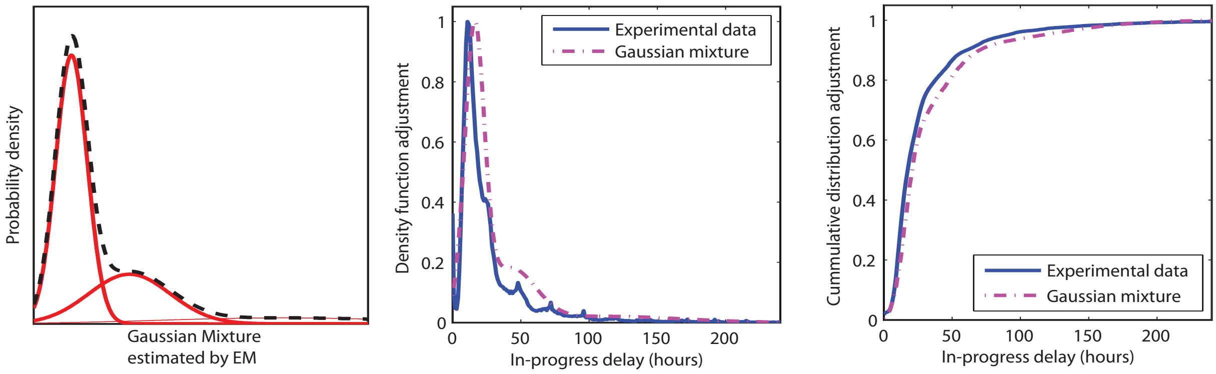


Figure 5. Model adjustment for P@H-CHARMM, in-progress delay, and a mixture of Gaussian distributions

CHARMM in Predictor@Home. Our metrics to estimate the quality of our predictions versus the empirical bounds of these projects are *accuracy* and *root-mean-square-deviation (RMSD)*. Accuracy represents the percentage of correctly predicted time bounds (below the upper time bounds) for a given project. Overall accuracy does not capture a too generous setting of the upper time bound. RMSD represents the fluctuation of the predicted time bounds and captures the divergence between predicted and empirical time bounds.

5.1 Data Analysis

Table 3 presents the best models for the different time phases across the four VC project. The table shows the best quantile, interval size used to shift the sampling windows, and probabilistic distribution. As we can see in the table, our analysis confirms results presented by Wolski in [19] indicating that the best probabilistic distribution for modeling the delay between the generation of jobs and their

Table 3. Best models across VC projects

Predicted Time	**Upper quantile** (b)	**Interval size** (hours)	**Probability Distribution**
Distribution	98	12	Weibull
In-progress	90	36	Mixture of Gaussian $k = 3$
Validation	90	36	Mixture of Gaussian $k = 3$

distribution is given by the Weibull distribution. To verify the Weibull's fitting, we performed the Anderson-Darling test [3]; our tests always succeeded with a 95% confidence. For both the prediction of in-progress delay and validation delay, we found that best predictions are obtained with a mixture of Gaussian distributions.

Table 4 presents accuracy and RMSD (absolute and normalized values) for the single VC projects and for each of the job lifespan delays when we use the best models in Table 3. For WCG projects (i.e., FightAIDS@Home and Genome Comparison), our models are very accurate: the lowest accuracy is 92%. For the P@H projects (i.e., MFold and CHARMM), our models can capture accurate results for the validation delay (above 92%) and the in-progress delay (above 86%) while the accuracy of the distribution delay drops significantly (above 66%). This is due to the short traces available for these two projects (as shown in Table 1). RMSDs depend on the size of the lifespan delays that are predicted. Therefore we can compare them only for a given type of delay across projects but not across delays within a project. When sufficiently long traces are available, like for the WCG projects, the RMSD across projects is similar and in average 10% for the distribution delay, 15% for the in-progress delay, and 5.5% for the validation delay. Too short traces for highly unstable projects, such as for the CHARMM project, provide us with inconclusive results.

We also compare the empirical time bounds of the four VC projects versus our predicted time bounds. Figures 6.a, 6.b, 6.c, and 6.d show the comparisons of the empirical and predicted time bounds (b_{dist}, b_{prog}, and b_{val}) for FightAIDS@Home, Genome Comparison, MFold, and CHARMM. The dash lines represent the empirical values and the solid lines the predictions. For the in-progress delays (b_{prog}), the dotted line represent the time-out limit set by the specific project. This limit determines when a job instance is considered timed-out and thus a new instance is generated and distributed. Note that this limit becomes too high for the WCG projects as the time goes by (causing increased job latencies) and too low for the P@H projects (causing wasting of resources). In three cases (i.e., FightAIDS@Home, Genome comparison, and MFold) our adaptive method provides better bound estimates by tracking and predicting the runtime behavior. The parts of Figures 6.a and 6.b related to the in-progress delays identify the reason for the higher RMSD for FightAIDS@Home and Genome Comparison in a poor prediction of the time bounds at the very beginning of the prediction process. As the predictions evolve, our models capture the behavior of the VC projects accurately. Although the RMSD may look high in Table 4 for the three time phases in P@H - MFold, and the distribution and validation bounds in P@H - CHARMM, we observe in Figures 6.c and 6.d how our models can accurately follow the general behavior of the empirical data. In particular, note how in Figure 6.c the distribution bound of our predictions follow the up-and-down trends of the empirical data. Our worst predictions are found for the in-progress delay of the P@H - CHARMM project as shown in Figure 6.d. The CHARMM project has a very high error rate and very short traces (see Table 1) making this project a very challenging project to model and predict.

Our results can impact the way in which VC projects are designed and implemented. Overall our study shows that longer traces are vital for accurate predictions. When long VC traces are available, our framework builds models that show similar patterns as real VC projects do.

6 Related Work

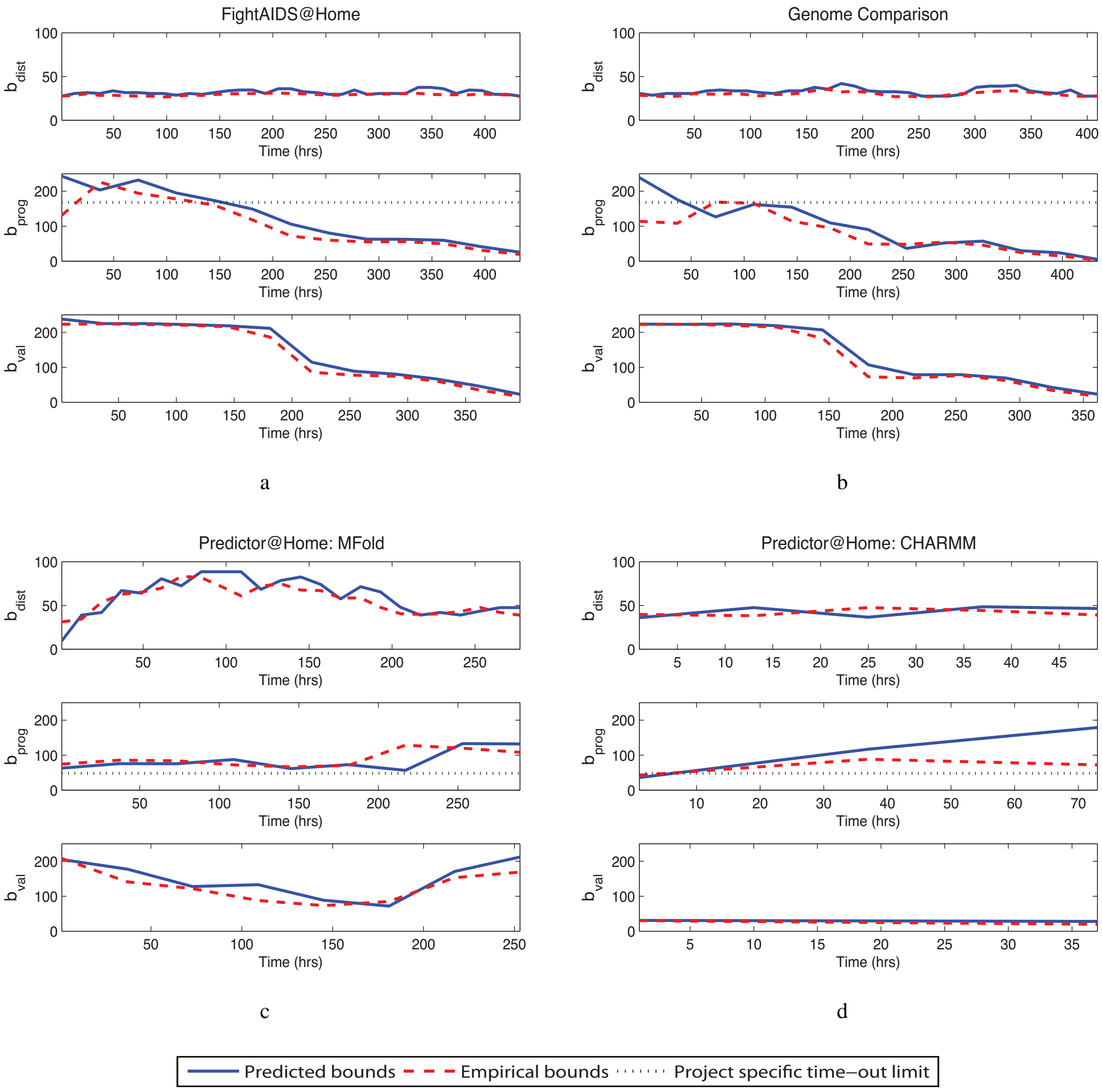
Important work has been done to predict distribution and execution times on grid and desktop grid computing systems. Predictions of distribution delays are presented in [6] where the authors use statistical techniques to predict jobs queue times submitted to a space-sharing parallel machine with the First-Come-First-Served scheduling policy. In [4, 14, 13], Wolski et al. estimate the time jobs wait in queues by using a non-parametric method and quantiles to define upper time bounds. The prediction of execution times is presented in [15] and [10]. In [15] the authors characterize and predict job execution times with templates derived from execution times of other similar applications. In [10] statistical methods are used to estimate the job execution time (modeled as a random variable) in heterogeneous environments (a large distributed network of heterogeneous machines) supported by an kNN algorithm.

Resource availability is modeled with statistical distributions in [9, 19, 18, 12]. In [9], Hein et al. model unreliable hosts with a Weibull distribution. In [19], Wolski et al. model resource capabilities, dynamic behavior, availability, failure, and connectivity for grid and global computing systems with Weibull and hyperexponential distributions. This work is extended by the authors in [18]. Two- and three-stage hyperexponentials are used to model machine availability in [12] when data shows up-and-down trends.

As in Wolski's work [4, 14, 13], we use quantiles to predict upper time bounds. As in Mutka's [12] and Wolski's [19] works, we achieve similar conclusions on the po-

Table 4. Accuracy and RMSD of VC project predictions

	Distribution		In-progress		Validation	
	b=98, update=6hrs, Weibull		b=90, rec=36hrs, mix-Gaussians		b=90, rec=36hrs, mix-Gaussians	
	Accuracy	RMSD	Accuracy	RMSD	Accuracy	RMSD
FightAIDS@Home	0.996	3.46 (0.09)	0.941	37.05 (0.14)	0.941	13.07 (0.05)
Genome Comparison	0.980	3.90 (0.11)	0.920	44.62 (0.17)	0.939	13.54 (0.06)
P@H: MFold	0.700	11.06 (0.11)	0.861	26.91 (0.21)	0.971	27.36 (0.12)
P@H: CHARMM	0.660	7.66 (0.14)	0.897	63.89 (0.33)	0.923	5.70 (0.16)



a

b

c

d

Figure 6. Predictions versus empirical results for four VC projects

tentials of combinations of distributions to model up-and-down trends. To the best of our knowledge, no past work extensively addresses predictions of distribution delays, in-progress delays, and validation delays in VC projects. Some of the studies listed above predict execution times in grid and desktop grid systems but these works do not include the time that a job spends waiting for execution on a host or waiting to have its results returned to the server. In addition, none of these studies predicts validation delays. In the case of VC projects, this time is vital for the delivery of results to scientists.

7 Conclusions

This paper presents statistical methods for predicting upper time bounds in VC projects, including distribution delay, in-progress delay, and validation delay. In our study we consider four distributions (Weibull, Levy, Gamma, and a mixture of Gaussians) for our modeling and the traces from four existing VC projects (FightAIDS@Home, Genome Comparison, MFold, and CHARMM) for our validation. Across the VC projects, the Weibull distribution consistently outperforms when predicting the distribution delay and the mixture of Gaussians outperforms when predicting in-progress delay and validation delay. The ability of the statistical models to capture the up-and-down trends in the VC projects is empirically evaluated. We show that when sufficiently long VC traces are available, we can predict upper time bounds with an accuracy above 92% for in-progress delay and validation delay, and an accuracy of 98% for the distribution delay. With these predictions, VC projects can take better decisions about when to generate new jobs and assign their instances to hosts as well as how to estimate delay before considering a job instance timed-out.

Acknowledgment

This work was supported by the National Science Foundation, grant #SCI-0802650, “DAPLDS - a Dynamically Adaptive Protein-Ligand Docking System based on multi-scale modeling” and by the CONACyT fellowship #171595. The authors thank the BOINC community for their time, dedication, and computer resources.

References

[1] D. Anderson, E. Corpela, and R. Walton. High-performance task distribution for volunteer computing. In *Proc. of the First International Conference on e-Science and Grid Computing*, 2005.

[2] D. P. Anderson and K. Reed. Celebrating diversity in volunteer computing. In *Proc. of the Hawaii International Conference on System Sciences (HICSS)*, 2009.

[3] T. W. Anderson and D. A. Darling. Asymptotic theory of certain 'goodness of fit' criteria based on stochastic processes. *The Annals of Mathematical Statistics*, 23(2):193–212, 1952.

[4] J. Brevik, D. C. Nurmi, and R. Wolski. Predicting bounds on queuing delay in space-shared computing environments. In *Proc. of the IEEE International Symposium on Workload Characterization*, 2006.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. of Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[6] A. B. Downey. Predicting queue times on space-sharing parallel computers. In *Proc. of the 11th International Parallel and Distributed Processing Symposium*, 1997.

[7] T. Estrada, D. Flores, M. Taufer, P. Teller, A. Kerstens, and D. Anderson. The effectiveness of threshold-based scheduling policies in boinc projects. In *Proc. of e-Science'06*, 2006.

[8] J. W. Harris and H. Stocker. *Maximum Likelihood Method*. Springer-Verlag, 1998.

[9] E. M. Heien, N. Fujimoto, and K. Hagihara. Computing low latency batches with unreliable workers in volunteer computing environments. In *Proc. of the 22nd International Parallel and Distributed Processing Symposium*, 2008.

[10] M. A. Iverson, F. Ozguner, and L. Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. *IEEE Transactions on Computers*, 48(12):1374–1379, 1999.

[11] G. Mclachlan and D. Peel. *Finite Mixture Models (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2000.

[12] M. W. Mutka and M. Livny. Profiling workstations' available capacity for remote execution. In *Proc. of the 12th International Symposium on Computer Performance Modeling, Measurement and Evaluation*, 1988.

[13] D. Nurmi, A. Mandal, J. Brevik, C. Koelbel, R. Wolski, and K. Kennedy. Evaluation of a workflow scheduler using integrated performance modelling and batch queue wait time prediction. In *Proc. of SC'06*, 2006.

[14] D. C. Nurmi, J. Brevik, and R. Wolski. Qbets: queue bounds estimation from time series. In *Proc. of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2007.

[15] W. Smith, V. Taylor, and I. Foster. Using run-time predictions to estimate queue wait times and improve scheduler performance. In *Job Scheduling Strategies for Parallel Processing*, pages 202–219. Springer Verlag, 1999.

[16] M. Taufer, C. An, A. Kerstens, and C. L. Brooks III. Predictor@home: A protein structure prediction supercomputer based on global computing. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):786–796, 2006.

[17] M. Taufer, D. P. Anderson, P. Cicotti, and C. L. Brooks. Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *Proc. of the 19th International Parallel and Distributed Processing Symposium*, 2005.

[18] R. Wolski, D. Nurmi, and J. Brevik. An analysis of availability distributions in condor. In *Proc. of the 21st International Parallel and Distributed Processing Symposium*, 2007.

[19] R. Wolski, D. Nurmi, J. Brevik, H. Casanova, and A. Chien. Models and modeling infrastructures for global computational platforms. In *Proc. of the 22nd International Parallel and Distributed Processing Symposium*, 2005.