Performance Modeling of Scalable Resource Allocations with the Imperial PEPA Compiler

William S. Sanders
Information Technology
The Jackson Laboratory
Farmington, CT USA
shane.sanders@jax.org

Srishti Srivastava
Computer Science
University of Southern Indiana
Evansville, IN USA
fsrishti@usi.edu

Ioana Banicescu
Computer Science & Engineering
Mississippi State University
Mississippi State, MS USA
ioana@cse.msstate.edu

***Abstract*—Advances in computational resources have led to corresponding increases in the scale of large parallel and distributed computer (PDC) systems. With these increases in scale, it becomes increasingly important to understand how these systems will perform as they scale when they are planned and defined, rather than post deployment. Modeling and simulation of these systems can be used to identify unexpected problems and bottlenecks, verify operational functionality, and can result in significant cost savings and avoidance if done prior to the often large capital expenditures that accompany major parallel and distributed computer system deployments. In this paper, we evaluate how PDC systems perform while they are subject to increases in both the number of applications and the number of machines. We generate 42,000 models and evaluate them with the Imperial PEPA Compiler to determine the scaling effects across both an increasing number of applications and an increasing number of machines. These results are then utilized to develop a heuristic for predicting the makespan time for sets of applications mapped onto a number of machines where the applications are subjected to perturbations at runtime. While in the current work the estimated application rates and perturbed rates considered are based on the uniform probability distribution, future work will include a wider range of probability distributions for these rates.**

***Index Terms*—Process algebra; Robustness analysis; Performance modeling; Performance evaluation; Application virtualization; Scalabilty; Stochastic processes**

I. Introduction

As high-performance computer systems designed for parallel and distributed computing (PDC) increase in both size and complexity, it becomes increasingly important to be able to evaluate their performance as they scale. Among the various methodologies, such as experimental evaluation, simulation, and numerical analyis, that are available for performance evaluation of scalable PDC systems, performance modeling and evaluation provide a lower cost, lower overhead, and better comparative analysis approach. Prior work has used performance modeling to evaluate the throughput, makespan, and robustness of static resource allocations of applications mapped onto heterogeneous machines using models constructed for the Performance Evaluation Process Algebra (PEPA) [1]–[4]. PEPA is a framework for performance modeling, and evaluation of models using stochastic process algebra. PEPA modeling is conducted using either the PEPA Plug-In for the Eclipse Integrated Development Environment (IDE) [5] or the Imperial PEPA Compiler (IPC) [6]. Several resource allocations, where the applications load and machine availability vary, have been modeled with PEPA and evaluated to obtain the performance of these systems prior to execution [2]–[4].

Both the PEPA Eclipse Plug-In and the IPC have been verified to produce identical results when evaluating PEPA models [7]. The PEPA Eclipse Plug-In is implemented as a graphical user interface (GUI), allowing users to develop and evaluate models with a robust environment and debugger. However, because of its underlying implementation, it is applicable only to the evaluation of models of limited size [7]. These model size limitations can be addressed through the use of the IPC and its command line interface (CLI). However, the IPC trades the robust IDE of the PEPA Eclipse Plug-In [7] for the capabilities of CLI-based execution and automation. A benefit of this trade-off is that the CLI-based nature of the IPC allows automation of the manual interactions required when using the PEPA Eclipse Plug-In, thus providing the capability for programmatic evaluation of PEPA models at scale [7].

In this work, the IPC is used to evaluate models of applications (a) mapped onto machines (m) where the applications are subjected to workload perturbations at runtime, as both the number of applications and the number of machines increase in scale. Prior work did not evaluate the scalability of the model as both the number of applications and the number of machines increase. We generate and evaluate 42,000 PEPA models with the IPC to determine system makespan time. Using these results, mathematical models were constructed that allow a prediction of makespan time for a given set of applications mapped onto machines. These predictions allow the construction of a heuristic that can be used to provide makespan estimates for system robustness targets, allowing a more informed determination of practical robustness metrics. The present framework uses variations in machine availability and application load to provide a foundation for early estimates of system robustness. These estimated metrics can in turn guide better robustness metric determination for more realistic predictions of broad system performance early in the acquisition process, producing cost savings across the system life cycle.

This paper is organized as follows. In Section II, a background on PEPA, robustness of resource allocations, and prior work on modeling the mapping of applications onto machines at scale are provided. In Section III, the methodology of our systematic evaluation of modeling these systems of applications subject to perturbations at runtime and mapped onto machines while both the number of applications and number of machines increase is presented, along with a methodology used to construct a heuristic function that can be utilized to provide a prediction of makespan time for a given system of applications and machines. In Section IV, the results of this work are described and discussed, and in Section V, conclusions and potential future directions are highlighted.

II. Background and Related Work

A. Performance Evaluation Process Algebra (PEPA)

To evaluate systems with dynamic properties, the Performance Evaluation Process Algebra (PEPA) was developed by Jane Hillston [8]. To represent the evolution of these dynamic systems, PEPA uses stochastic Markov processes to model the state changes of random variables in modeled systems. The probabilistic nature of Markov processes models the probability that a given component is in its current state based on the inputs of possible previous states [8], [9]. This dynamic modeling capability allows PEPA to accurately model dynamic systems such as modeling the runtime performance of a parallel computing cluster, or modeling the availability of resources to fulfill requests in a queue-based scheduling system [2], [3], [8], [10], [11].

Markov models have been previously used to conduct numerical analysis of the behavior of computer systems [2], [12]. To more accurately address system models where events are both countably finite and occur at non-specific time intervals, continuous time Markov chains (CTMC) were chosen as the underlying mechanism for describing stochastic system evolution in PEPA [2]–[4]. This is because a continuous time (CT) representation more accurately captures countably finite events occurring at non-specific time intervals than the discrete time (DT) representation found when utilizing discrete time Markov chains [2]–[4]. Models using a CTMC representation are evaluated as each event occurs, whereas in DTMC representations, models are evaluated at predefined or discrete time intervals. Systems utilizing CTMCs allow a more accurate time representation for concurrent events, as DTMCs do not support modelling systems with concurrent behavior [13]. Evaluation of a CTMC model at the occurrence of each event, concurrent systems where many parallel systems acting independently can be modeled more accurately [2]–[4].

As the size of the system being evaluated increases, the underlying Markov chains grow increasingly larger. This can lead to a corresponding increase in the number of finite states to be evaluated, which is the core of the *state space explosion* problem [14]. The *state space explosion* problem occurs when an increasing number of states must be evaluated in order to generate a solution for a given problem, leading to a significant increase in time and memory requirements [8], [14]. This often makes standard, brute-force solutions infeasible, and it can limit the scalability of the algorithm used [14].

The PEPA framework provides modelers with a concise set of modeling algebra and an abstraction to complex CTMC evaluation which has led to the development of PEPA models for a number of systems. Additionally, the PEPA framework has been extended to provide static evaluations for concurrent systems, allowing additional model development. The published PEPA models include performance modeling of grid-based scheduling algorithms to process United Kingdom National Weather Service data, algorithmic performance on sets of possibly limited resources, and online resources configured in a grid-framework [11], [15], [16]. Continued advances have extended PEPA to incorporate state-aware components, allowing construction of component specific feedback into the PEPA models [17]. The combination of insights from the current and prior work with the PEPA framework indicate that this modeling framework can effectively model and predict the performance of simulated parallel and distributed computing systems as those systems scale. This framework allows system architects to evaluate the impacts of scale as systems are designed and planned, rather than post-deployment.

B. Robustness

Frameworks for job-shop application scheduling were used for initial work on robust scheduling [18]–[21]. A standard branch and bound approach was used to solve the NP-Hard robust scheduling problem (RSP) for a robust schedule of N independent jobs on a single machine [22]. Various optimization techniques, including both stochastic mixed integer programming [23] and iterative integer programming [24] have been used to obtain an initial robust resource allocation. Additional work has been done in conjunction with the generation of these robust resource allocations and metrics [25]–[27] to study the performance guarantee of available static resource allocations against possible inadequate change or variation in the computational environment parameters using a general methodology named the Feature Perturbation Impact Analysis (FePIA) procedure [28]. FePIA defines a resource allocation to be robust with respect to specific system performance features against perturbations (uncertainties), if the degradation of these features is constrained when limited perturbations occur [28] [29]. Additional research has also been conducted to evaluate the robustness of scheduling techniques at an application level through the analysis of dynamic loop scheduling algorithms, previously shown to be effective in the dynamic scheduling of applications in parallel and distributing computing systems in the presence of both varying processor loads (the flexibility metric) and varying processor failures (the resilience metric) [30] [31].

These metrics are of increasing importance as we evaluate systems where the underlying application can be subjected to perturbations at runtime. In an ideal computing environment, T_{ij} is the actual time to compute an application i on machine j with initial availability (η_j) that remains constant at runtime, and $\hat{\lambda_i}$ is calculated as a function of the initial workloads $\lambda_1, \lambda_2, \lambda_3$ [3]. This is shown in Equation 1.

$$r_i = \frac{\lambda_i}{T_{ij}} \quad \forall i, j, \text{ where } T_{ij} = \lambda_i \times \eta_j \tag{1}$$

T_{ij} is calculated as a product of the runtime workload for that application (λ_i), and the machine availability factor (η_j). Rate r_i is calculated as a ratio of the application workload λ_i and T_{ij}. In the ideal execution scenario, $\hat{\lambda_i} = \lambda_i, \forall i \in \{1, 2, 3\}$. Therefore, T_{ij} is equal to the initial expected time to compute values and consequently, the rates, $(r_1, r_2, \cdots)$, are only calculated using the initial machine availability (η_j).

Equation 2 shows the calculation for determining the perturbed rate of an application [2]. T_{ij} is the actual time to compute an application i on machine j with the varied runtime availability ($\hat{\eta_j}$), and $\hat{\lambda_i}$ is calculated as a function of the varying workloads $\hat{\lambda_1}, \hat{\lambda_2}, \hat{\lambda_3}$, and λ_i is calculated as a function of the initial sensor loads $\lambda_1, \lambda_2, \lambda_3$.

$$p_i = \frac{\lambda_i}{T_{ij}} \quad \forall i, j, \text{ where } T_{ij} = \hat{\lambda_i} \times \hat{\eta_j} \tag{2}$$

T_{ij} is calculated as a product of the estimated runtime application workload ($\hat{\lambda}$), and the runtime machine availability factor ($\hat{\eta_j}$). Henceforth, p_i is calculated as a ratio of the initial application workload λ_i and T_{ij}.

For a set of parallel applications A, $a_i \in A$: a parallel application, λ_i: the workload of application a_i, β_i^{max}: a user defined makespan goal for a_i, M: a set of parallel machines, $M_j \in M$: the machine allocated to a_i, $\hat{\lambda_i}$: a perturbation parameter defined as the workload variation from the initial workload (λ_i) for an application a_i, $\hat{\eta_j}$: a perturbation parameter defined as the machine availability variation from the initial machine availability (η_j) for an application a_i, $F_i(M_j(\hat{\eta_j}), \hat{\lambda_i})$: the finishing time of application a_i on machine M_j. Then, the *robustness* (ψ) of a mapping is shown in Equation 3 [2].

$$\psi = \min_{\forall i \in A} \Pr[F_i(M_j(\hat{\eta_j}), \hat{\lambda_i}) \leq \beta_i^{max}] \tag{3}$$

The goal of the robustness analysis is to find a resource allocation that maximizes the robustness of the execution of the applications on the allocated parallel machines. A robustness metric for the makespan time of systems of applications mapped onto machines can allow system designers and architects to maximize those parallel and distributed systems towards optimal performance, risk aversion, stability of performance, or minimizing variance as they plan and design their systems [32].

C. Alternative Methods for Performance Evaluation

There are three broad approaches for performance evaluation of computer and communication systems: analytical and numerical modeling, simulation, and direct experimentation [33]. A number of factors are critical when choosing a performance evaluation technique, including the need for generality, time analysis, comparative analysis of systems, tools required, stage of the system under evaluation, and cost. Direct experimentation requires the availability of a system and control over these critical factors. Simulation provides a more cost-efficient approach for performance evaluation than direct experimentation, but it can be a time-consuming process with significant overhead and trade-offs. It requires developing an abstraction of a parallel and distributed computing system, a validation of simulations, and statistical analysis to determine soundness of the model. Analytical and numerical modeling provides a more cost-efficient approach than both direct experimentation and simulation for performance evaluation. It allows derivation of an expression for the performance feature of interest, and can provide better replication and insight into the effects of various parameters for predictive analysis of the performance of a computer system [1], [2], [10], [11].

III. Methodology

To build upon previous work on exploring the scalability of the IPC [7], an experimental design was constructed to generate models of applications (a) mapped onto machines (m) where the applications are subjected to perturbations at runtime based on Eq. 4. This design generates 42 (m, a) pairs ranging from $(2^0, 2^1)$ to $(2^6, 2^{12})$. For each (m, a) pair, 10^3 PEPA models $(N = 10^3)$ are programmatically generated using Python v3.6.7 with NumPy v1.17.4 to produce a random mapping of a applications onto m machines. The application rates and perturbed rates for these programmatically generated mappings are derived from the uniform distribution with values ranging from 0.00 to 1.00, as shown in Fig. 1. An overview of the sampling provided by this experimental design is shown in Table I, and this design allows for consistent a to m ratios as both a and m increase. This experimental design results in the creation and evaluation of 42,000 PEPA models, and each PEPA model was then evaluated using the IPC installed into a Singularity container (ipc v0.02, Singularity v3.4.1). [7]

$$(m, a) = (2^i, 2^{(i+j)}) \text{ where } \begin{matrix} 0 \leq i \leq 6 \\ 1 \leq j \leq 6 \end{matrix} \tag{4}$$

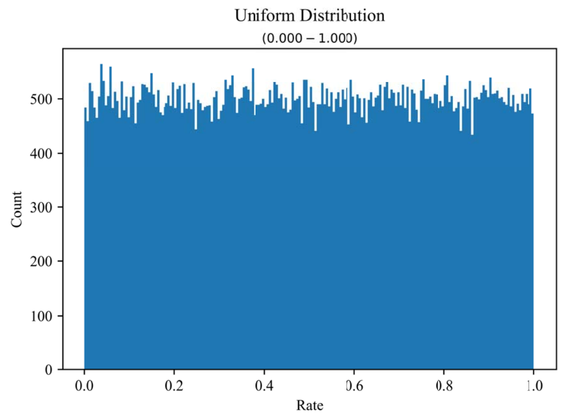

Fig. 1. Statistical Distribution Used to Derive Application Rates and Perturbed Rates. Uniform Distribution with Values Ranging from 0.000 to 1.000.

Using the simulation results from the evaluation of these 42,000 PEPA models, a heuristic is devised based on curve

TABLE I
EXPERIMENTAL DESIGN USED TO GENERATE MAPPINGS OF APPLICATIONS ONTO MACHINES.

m	a	(m, a)	a/m
1	2	(1, 2)	2
1	4	(1, 4)	4
1	8	(1, 8)	8
1	16	(1, 16)	16
1	32	(1, 32)	32
1	64	(1, 64)	64
2	4	(2, 4)	2
...			
64	128	(64, 128)	2
...			
64	4,096	(64, 4,096)	64

m - Number of Machines
a - Number of Applications
(m, a) - Machine-Application Pair
a/m - Application-to-Machine Ratio

fitting of the data in a three-dimensional space using the Python (v3.6.7) NumPy (v1.17.4) library, with number of machines (m) on the X-axis, number of applications (a) on the Y-axis, and the statistical values (lower whisker, Q1, Median, Q3, upper whisker) of the calculated Inter Quartile Range (IQR) as the Z-axis. This approach generates five equations of the form $Z = MX + NY + B$, where each equation represents the plane of the IQR values. Using this set of equations, a prediction can be made to determine the distribution of makespan times for a given (m, a) pair where a applications are mapped onto m machines, the applications are subject to perturbations at runtime, and the application rates and perturbed rates are derived from the uniform distribution.

IV. RESULTS AND ANALYSIS

A total of 42,000 PEPA models with an experimental design increasing both the number of applications and the number of machines were programmatically generated and evaluated with the IPC. Box plots for all completed PEPA models (where applications mapped onto machines and where the application rate and perturbed rate were based on the uniform distribution that completed by the end of simulation) are shown in Figures 2 to 8. It was determined that the models that had not completed by the simulation end time were representative of a significantly large number of applications mapped onto machines where the number of applications mapped onto machines is nearly unrealistic on real production systems, for example 4,096 applications mapped onto 64 machines.

These boxplot figures highlight that as an increasing number of applications with perturbations at runtime are mapped onto a constant number of machines, the amount of time for the machines to successfully complete the application workload increases as the number of applications increase. This is shown by the increasing median and IQR shown on the boxplots as the application load increases for a fixed number of machines. This is an expected result, as when an increasing number of tasks with similar complexity are given to a fixed amount of workers, in general it takes more time for the workers to

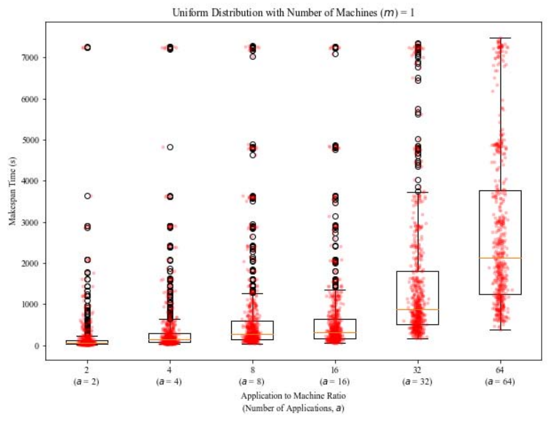

Fig. 2. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=1 and the number of applications (a) per machine (m) is determined by Eq. 4.

complete those tasks. What is notable about these results is that to the authors' best knowledge, they represent for the first time a systematic evaluation of the scalability of PEPA models obtained from simulations conducted with sets of applications mapped onto machines where the applications are subjected to perturbations at runtime, and where the number of the applications to number of machines are at the magnitude presented in these results.

A linear regression of a $\log_2$-$\log_2$ transformation showing the median makespan times mapped to application to machine ratios (a/m) for all completed PEPA models is shown in Fig. 9. For this regression, the application to machine ratio (a/m) serve as the independent variable, controlled by the experimental designer, while the median makespan time represents the dependent variable, with its value is dependent on changes to the independent variable. The R^2 values presented in both Fig. 9 and Table II represent the percentage of the dependent variable, in this case the median makespan time, that is explained by the linear model. At lower machine numbers, the R^2 value is very high, showing a strong relationship between the linear model and the makespan time. However, the correlation declines slightly when $m = 16, 32, 64$. The boxplots shown in Figures 6 to 8 and the upper bound of Fig. 9 show that for higher application to machine ratios, a number of these models are less likely to complete by the end of the simulation period. We hypothesize that given the strength of the observed linear trends, the R^2 for linear regression of these PEPA models with high numbers of applications mapped onto a given number of machines would follow the results of PEPA models with smaller application to machine pairs.

A two-dimensional linear regression of the model results fails to accurately represent the dimensionality of the experimental design utilized in this study. Separating the application to machine ratio into its components of number of applications and number of machines allows us to evaluate the results in

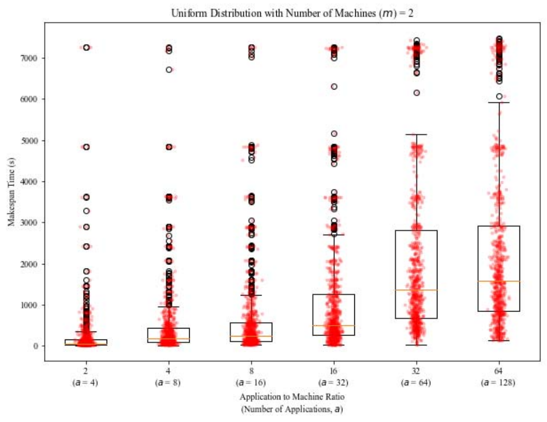

Fig. 3. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=2 and the number of applications (a) per machine (m) is determined by Eq. 4.

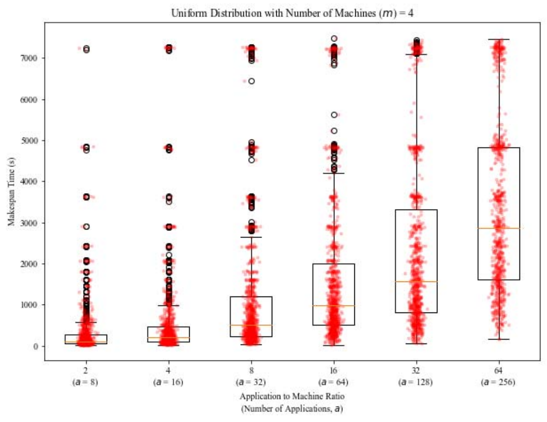

Fig. 4. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=4 and the number of applications (a) per machine (m) is determined by Eq. 4.

TABLE II
R^2 VALUES FOR THE LINEAR REGRESSION OF APPLICATION TO MACHINE RATIO AND MEDIAN MAKESPAN.

Number of Machines (m)	R^2 Value
m_1	0.968
m_2	0.970
m_4	0.992
m_8	0.983
m_{16}	0.958
m_{32}	0.945
m_{64}	0.835

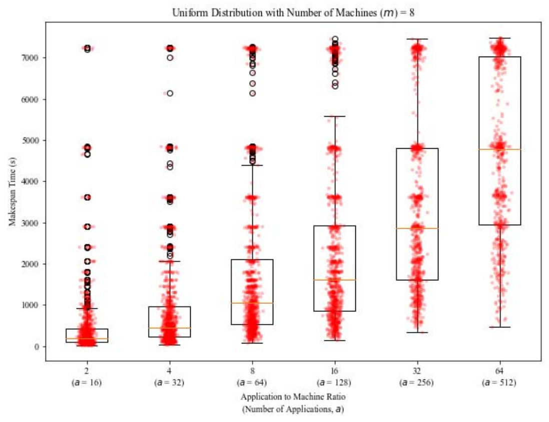

Fig. 5. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=8 and the number of applications (a) per machine (m) is determined by Eq. 4.

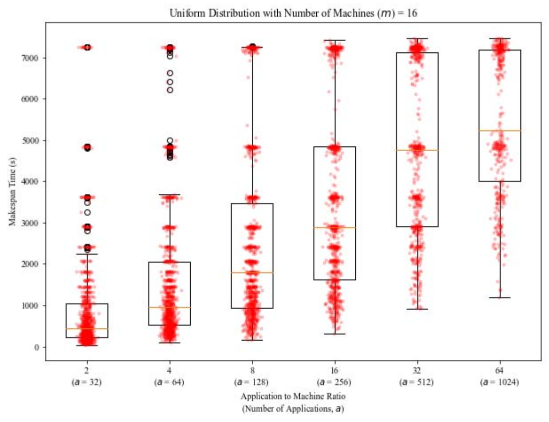

Fig. 6. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=16 and the number of applications (a) per machine (m) is determined by Eq. 4.

a higher dimensionality. This is shown in Fig. 10, where the $\log_2$ median makespan time for all completed PEPA models is plotted against the $\log_2$ applications and $\log_2$ machines. This 3-D analysis can be done for all the statistical values represented by a 2-D boxplot, resulting in plane equations represented in Table IV, which shows the values of the coefficients for these curve-fitted planes based on the plane equation $Z = MX + NY + B$ where x = $\log_2$ (number of machines), y = $\log_2$ (number of applications), and z = $\log_2$ (statistical feature), for example, the median makespan time.

The equations representing the curve-fitted planes for the makespan data from all completed PEPA models allows the construction of a heuristic function for facilitating the prediction of makespan times for a given model of applications mapped onto machines where the applications are subject to

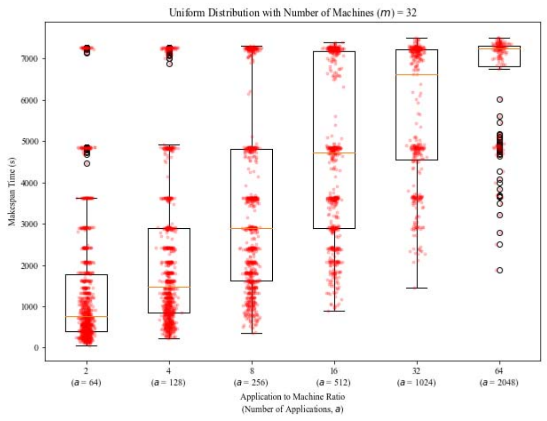

Fig. 7. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=32 and the number of applications (a) per machine (m) is determined by Eq. 4.

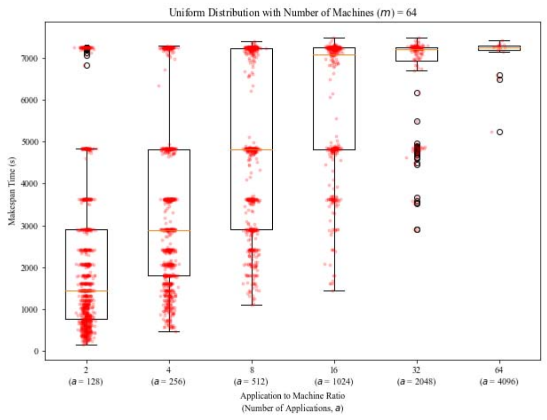

Fig. 8. Box plots of makespan time for PEPA models evaluated with the IPC where applications (a) with rates and perturbed rates based on the Uniform distribution (shown in Fig. 1) are mapped onto machines (m), where m=64 and the number of applications (a) per machine (m) is determined by Eq. 4.

perturbations at runtime and the rates and perturbed rates of the applications are derived from the uniform distribution. To evaluate these curve-fitted planes as a heuristic for makespan time, five application (a) to machine (m) pairs (m, a) that fell within the bounds of the experimental design were derived $((3, 10), (10, 200), (15, 450), (24, 800),$ and $(28, 1200))$ and 1,000 models for each (m, a) pair were generated and evaluated. The results of these evaluations are shown in Table IV. Light grey cells in Table IV represent scenarios where the projected makespan time for the relevant statistical value (median, Q3, 95%-whisker) is greater than the duration of the simulation period.

This heuristic provides a reasonable estimate of the statistical distribution of makespan times for a given mapping of applications to machines with the Q1, Median, and Q3 values

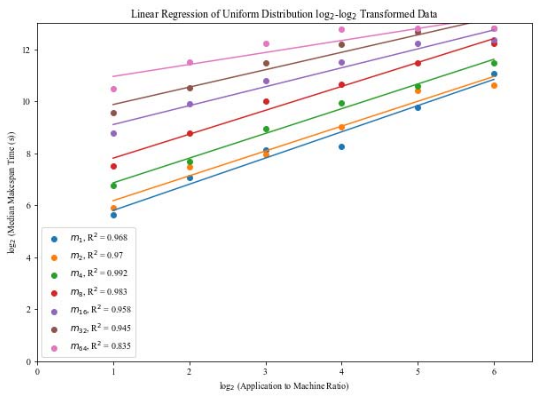

Fig. 9. Linear regression of $\log_2$-$\log_2$ transformed IPC results, where the x-axis is $\log_2(a/m)$ and the y-axis is $\log_2$(Median Makespan Time (s)) for all PEPA models that completed evaluation with the IPC by $stopTime = 7500$.

of the estimations being an average of 6.21%±3.79% from their expected values. This was calculated by calculating the mean and standard deviation for the differences between the expected result and experimental results for the Q1, median, and Q3 values. The 5% and 95% whiskers show a greater variation in their accuracy, and at the higher end, this is reflective of the simulation duration for this analysis, and is grounded in the highly correlated linear regression analysis for this data set. This heuristic provides reasonable estimates for when a given set (m, a) of applications (a) mapped onto machines (m) will complete when the applications are subject to perturbation at runtime. To the best of the authors' knowledge, this represents the first reported heuristic for these systems, and in the absence of other information, allows a baseline prediction of makespan time to be generated. This prediction can then be utilized as a potential robustness target. In the absence of other information about a given workflow, this heuristic can provide a more informed estimate of a makespan time for a robustness target over a random value. Architectural and design planning can incorporate predictions using the PEPA framework while these systems are still in development, allowing system designers additional flexibility and confidence in utilizing robustness metrics based on makespan for evaluating a given system's planned workload.

TABLE III
COEFFICIENTS FOR CURVE-FITTED 3D PLANES BASED ON $z = mx + ny + b$.

	m	n	b
Upper Whisker (95%)	-0.128	0.540	8.621
Upper Quartile (75%)	-0.112	0.669	6.983
Median (50%)	-0.134	0.813	5.328
Lower Quartile (25%)	-0.144	0.887	3.989
Lower Whisker (5%)	-0.007	0.890	1.404

TABLE IV

RESULTS OF FIVE MACHINE (m) APPLICATION (a) PAIRS EVALUATED WITH THE HEURISTIC FUNCTION CONSTRUCTED BY 3-D PLANE FITTING OF COMPLETED PEPA MODELS. FOR EACH (m, a) PAIR, 1,000 MAPPINGS OF APPLICATIONS TO MACHINES WERE GENERATED AND EVALUATED USING THE IMPERIAL PEPA COMPILER.

(m, a)	≤5%		Q1		Median		Q3		≤95%	
	Makespan Prediction	Result	Makespan Prediction	Result	Makespan Prediction	Result	Makespan Prediction	Result	Makespan Prediction	Result
(3,10)	20.368	1.371%	104.604	36.814%	211.787	60.970%	522.200	81.329%	1185.106	91.983%
(10,200)	290.199	0.356%	1255.183	23.328%	2058.418	42.399%	3386.676	63.539%	5118.572	85.867%
(15,450)	595.351	0.000%	2431.475	21.439%	3769.197	47.430%	5568.070	69.016%	7528.528	100.000%
(24,800)	990.051	0.000%	3786.572	26.172%	5649.781	53.711%	7763.766	100.000%	9670.083	100.000%
(28,1200)	1418.534	0.000%	5307.097	32.642%	7695.17	100.000%	10009.050	100.000%	11801.028	100.000%
Mean		**0.345%**		**28.079%**		**51.128%**		**71.294%**		**88.925%**

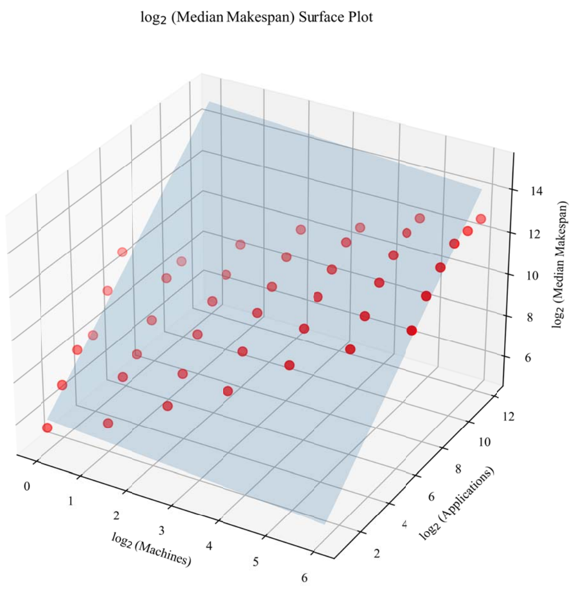

Fig. 10. 3-D Plane Fitting 3-Dimensional Median Makespan Results. Similar 3-D planes were constructed for relevant statistical features and the plane coefficients can be seen in Table IV.

A. *Threats to Validity*

The linear regression analysis and models constructed from these regressions are only valid for the ranges $((m, a) = (1, 2)...(64, 4096))$. While we can hypothesize that the model may hold for values greater than those evaluated in this study given the strong linearity evidenced by the coefficients of regression (R^2 values), that is not explored nor validated by this work.

The choice of the uniform distribution to generate application rates and perturbed rates where the rates are in the interval $(0, 1)$ normalized covers the entire range of possible application rates and perturbed rates. While it should be expected that many real-world applications will have non-uniform (ex: normal, logarithmic, exponential, etc.) application rates and perturbed rates, the choice of the uniform distribution serves as a starting point on which future work can be based. The significance of the derived heuristic may be lower if non-uniform application rates do not lend themselves to the strong linear regression analysis observed with the uniform distribution.

V. CONCLUSIONS AND FUTURE WORK

In this work, we have evaluated the scalability of resource allocations using PEPA for modeling applications mapped onto homogeneous machines where the applications are subjected to perturbations at runtime and the application rates and perturbed rates are drawn from the uniform distribution. It was found that as both the number of applications and the number of machines scale for the models evaluated, the model makespan time also increased in a linear relationship when the number of machines, the number of applications, and the makespan time are all $\log_2$ transformed. A linear regression of all available data indicates that a significant amount of variance in the model makespan time is dependent on both the number of applications and machines.

The makespan data generated by simulation of these PEPA models was used to create a three dimensional representation of the data, and plane equation coefficients were calculated for the five statistical values (5%-whisker, Q1, median, Q3, 95%-whisker) used to describe a statistical distribution of the data. These plane equations were then used to inform a heuristic for determining model makespan time for a given number of applications and machines. For the interquartile range (Q1, median, and Q3), this heuristic is accurate with an approximately 2.64% deviation from the expected prediction. This heuristic can be utilized as a predictor of makespan time for a given set of applications mapped onto a number of machines when there is little information known about the application rates or distribution of applications onto machines within a specific system that is modeled. In addition, this heuristic can help inform makespan values used to define a robustness target when little information is known about the underlying application rates and the machines can be assumed to be homogeneous in nature. Accounting for the variation in the expected system workload of both, the number of applications and the number of machines during the initial architectural planning of a new system can increase the likelihood of building a robust system, minimizing the need for post-deployment performance evaluation. Shifting this evaluation to earlier stages of the design and acquisition process can aid in the reduction of total system costs, including system downtime for component upgrades and stability improvements.

As future work, an examination of alternative statistical distributions to derive application rates other than the uniform distribution will be explored. Rates could also be modeled from application rate data from real world production high performance and parallel computing clusters. Additional work could explore if variation from the linear model is caused by machine overload, where due to load imbalance a small number of machines are executing the majority of applications.

The authors have made available application scripts, software containers, and other digital artifacts used to conduct this work on GitHub at https://github.com/williamssanders/pepa [34].

VI. Acknowledgments

The authors would like to acknowledge partial support of this work from The Jackson Laboratory (JAX) Tuition Reimbursement Program.

References

[1] J. Hillston, *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[2] S. Srivastava, "Evaluating the robustness of resource allocations obtained through performance modeling with stochastic process algebra," Dissertation, Mississippi State University, May 2015.

[3] I. Banicescu and S. Srivastava, "Towards robust resource allocations via performance modeling with stochastic process algebra," in *In Proceedings of the 18th IEEE International Conference on Computational Science and Engineering*. Porto, Portugal: IEEE, Oct. 2015, pp. 270–277.

[4] S. Srivastava and I. Banicescu, "Robust resource allocations through performance modeling with stochastic process algebra," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 7, p. e3894, 2017.

[5] S. Gilmore and J. Hillston, "The pepa workbench: A tool to support a process algebra-based approach to performance modelling," in *Computer Performance Evaluation Modelling Techniques and Tools*, ser. Lecture Notes in Computer Science, G. Haring and G. Kotsis, Eds. Springer Berlin Heidelberg, 1994, vol. 794, pp. 353–368.

[6] J. Bradley, N. Dingle, S. Gilmore, and W. Knottenbelt, "Derivation of passage-time densities in pepa models using ipc: the imperial pepa compiler." Orlando, FL, USA: IEEE, 2003, pp. 344–351.

[7] W. S. Sanders, S. Srivastava, and I. Banicescu, "Robustness analysis of scaled resource allocation models using the imperial pepa compiler." Warsaw, Poland: IEEE, 2020, pp. 60–67.

[8] J. Hillston, "Tuning systems: From composition to performance," *The Computer Journal*, vol. 48, pp. 385–400, 2005.

[9] ——, "Process algebras for quantitative analysis," *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pp. 239–248, 2005.

[10] A. Benoit, M. Cole, S. Gilmore, and J. Hillston, "Evaluating the performance of skeleton-based high level parallel programs," *Computational Science*, vol. ICCS 2004, pp. 289–296, 2004.

[11] ——, "Scheduling skeleton-based grid applications using pepa and nws," *The Computer Journal*, vol. 48, pp. 369–378, 2005.

[12] V. L. Wallace and R. S. Rosenberg, "Markovian models and numerical analysis of computer system behavior," in *Spring Joint Computer Conference*. Association of Computing Machinery, 1966, pp. 141–148.

[13] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2006, ch. PRISM: A Tool for Automatic Verification of Probabilistic Systems, pp. 441–444.

[14] R. Pelánek, "Fighting state space explosion: Review and evaluation," in *International Workshop on Formal Methods for Industrial Critical Systems*, 2009, pp. 37–52.

[15] A. Benoit, M. Cole, S. Gilmore, and J. Hillston, "Enhancing the effective utilisation of grid clusters by exploiting on-line performability analysis," *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, vol. 1, pp. 317–324 Vol. 1, 2005.

[16] ——, "Evaluating the performance of pipeline-structured parallel programs with skeletons and process algebra," *Scalable Comput. Pract. Exp.*, vol. 6, no. 4, 2005. [Online]. Available: http://www.scpe.org/index.php/scpe/article/view/343

[17] A. Clark and S. Gilmore, "State-aware performance analysis with extended stochastic probes," in *Lecture Notes in Computer Science*, ser. EPEW 2008, vol. 5261, 2008, pp. 125–140.

[18] E. Coffman and J. Bruno, *Computer and job-shop scheduling theory*. Wiley, 1976.

[19] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, Nov. 1989.

[20] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *J. ACM*, vol. 24, no. 2, pp. 280–289, 1977.

[21] J. Leon, D. Wu, and R. Storer, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, vol. 26, no. 5, pp. 32–43, 1994.

[22] R. Daniels and J. Carrillo, "Beta-robust scheduling for single machine systems with uncertain processing times," *IIE Transactions*, vol. 29, no. 11, pp. 977–985, 1997.

[23] S. Gertphol and V. Prasanna, "Mip formulation for robust resource allocation in dynamic real-time systems." Nice, France: IEEE, 2003, pp. 8 pp.–.

[24] ——, "Iterative integer programming formulation for robust resource allocation in dynamic real-time systems." Santa Fe, NM, USA: IEEE, 2004, pp. 118–.

[25] A. Ghafoor and J. Yang, "A distributed heterogeneous supercomputing management system," *Computer*, vol. 26, pp. 78–86, 1993.

[26] M. 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*, vol. 48, pp. 1374–1379, 1999.

[27] S. Ali, "Robust resource allocation in dynamic distributed heterogeneous computing systems," PhD Dissertation, Purdue University, 2003.

[28] S. Ali, A. Maciejewski, H. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 630–641, 2004.

[29] S. Ali, J. Kim, H. J. Siegel, and A. A. Maciejewski, "Static heuristics for robust resource allocation of continuously executing applications," *J. Parallel Distributed Comput.*, vol. 68, no. 8, pp. 1070–1080, 2008.

[30] I. Banicescu, F. M. Ciorba, and R. L. Carino, "Towards the robustness of dynamic loop scheduling on large-scale heterogeneous distributed systems." Lisbon, Portugal: IEEE, 2009, pp. 129–132.

[31] S. Srivastava, I. Banicescu, and F. M. Ciorba, "Investigating the robustness of adaptive dynamic loop scheduling on heterogeneous computing systems." Atlanta, GA, USA: IEEE, 2010, pp. 1–8.

[32] C. McPhail, H. R. Maier, J. H. Kwakkel, M. Giuliani, A. Castelletti, and S. Westra, "Robustness metrics: How are they calculated, when should they be used and why do they give different results?" vol. 6, pp. 169–191, 2018.

[33] R. Jain, *The art of computer systems performance analysis*. New York: John Wiley & Sons., 1991.

[34] W. S. Sanders, S. Srivastava, and I. Banicescu, "A container-based framework to facilitate reproducibility in employing stochastic process algebra for modeling parallel computing systems." Rio de Janeiro, Brazil: IEEE, 2019, pp. 373–381.