Robustness Analysis of Scaled Resource Allocation Models Using the Imperial PEPA Compiler

William S. Sanders
Computer Science & Engineering
Mississippi State University
Mississippi State, MS USA
wss2@msstate.edu

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*—The increase in scale provided by distributed computing systems has expanded scientific discovery and engineering solutions. Stochastic modeling with Performance Evaluation Process Algebra (PEPA) has been used to evaluate the robustness of static resource allocations in parallel and distributed computing systems. These evaluations have previously been performed through the PEPA Plug-In for the Eclipse Integrated Development Environment and have been limited by factors that include: i) the size and complexity of the underlying, in-use PEPA model, ii) a small number of resource allocation models available for analysis, and iii) the human interaction necessary to configure the PEPA Eclipse Plug-In, thus limiting potential automation. As the size and complexity of the underlying PEPA models increases, the number of states to be evaluated for each model also greatly increases, leading to a case of *state space explosion*. In this work, we validate the Imperial PEPA Compiler (IPC) as a replacement for the PEPA Eclipse Plug-In for the robustness analysis of resource allocations. We make available an implementation of the IPC as a Singularity container, as part of a larger online repository of PEPA resources. We then develop and test a programmatic method for generating PEPA models for resource allocations. When combined with our IPC container, this method allows automated analysis of resource allocation models at scale. The use of the IPC allows the evaluation of larger models than it is possible when using the PEPA Eclipse Plug-In. Moreover, the increases in scale in both model size and number of models, support the development of improved makespan targets for robustness metrics, including those among applications subject to perturbations at runtime, as found in typical parallel and distributed computing environments.

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

I. Introduction

The increase in scale made feasible by the arrival of distributed computing systems extends the horizons for scientific discovery and engineering solutions. However, the complexity costs of maintaining these infrastructures present new challenges. Earlier work examining the robustness of static resource allocations through performance modeling of applications mapped onto heterogeneous machines has been conducted by implementing models in the Performance Evaluation Process Algebra (PEPA), a framework for performance modeling and evaluation using stochastic process algebra [1]. Through the PEPA Plug-In for the Eclipse Integrated Development Environment (IDE) [2], the PEPA models were used to obtain the performance of static resource allocations that are obtained prior to runtime in parallel and distributed computing environments when application runtime characteristics and system runtime availability vary.

The PEPA Eclipse Plug-In is implemented in Java, and through its utilization of an IDE and its accompanying graphical user interface (GUI), it requires significant interaction with the IDE by researchers seeking to utilize PEPA for their modeling efforts. This Java-based implementation can also limit the size of PEPA models being analyzed, because the PEPA Eclipse Plug-In is itself constrained by the overall memory limits within Eclipse and Java. The PEPA Eclipse Plug-In executes as a single process within Eclipse/Java, and it is subject to a 3GB maximum address space per process due to this implementation [3]. This makes it unfeasible to simulate models with a large number of states. The increases in size and complexity of the system being modeled are accompanied by an increase in the number of states to be modeled as part of an underlying *state space explosion*.

Our previous work [4]–[6] has been limited both by the small number of published models available to be utilized for generating PEPA models, and by the aforementioned limitations of the PEPA Eclipse Plug-In itself. Additional previous work has sought to help facilitate the efforts of those seeking to employ PEPA and its derivatives by providing publicly available software containers and other resources to help overcome any issues with the installation of these tools [36].

In this work, we present efforts to overcome these limitations and provide tools for generating models of n applications mapped to k machines while also replacing the usage of the PEPA Eclipse Plug-In with a containerized version of the Imperial PEPA Compiler (IPC) [7]. The IPC is based on the Haskell programming language and is compiled before execution, unlike the Java-based implementation of the PEPA Eclipse Plug-In. The IPC was originally developed alongside the PEPA Eclipse Plug-In, but has not been as widely adopted as the PEPA Eclipse Plug-In. The substitution of the IPC for the PEPA Eclipse Plug-In allows us to automate the manual interaction that was previously necessary when using the PEPA Eclipse Plug-In. Through this automation, it becomes possible to evaluate the distributions of various models of applications with runtime perturbations mapped to machines. This allows us to gain better insight into the use of robustness and its metric for the evaluation of static resource allocations in parallel and distributed computing systems. The IPC is also adept at solving larger Markov models than the ones used by the PEPA Eclipse Plug-In, thus allowing an increase in both the size and complexity of the analyzed models.

This paper is organized as follows. In Section II of this paper, we provide a background on PEPA, the robustness of resource allocations, and the numerical challenges for modeling the mapping of applications to machines at scale. In Section III, we then describe our methodology for overcoming the limitations of the PEPA Eclipse Plug-In by utilizing the IPC. In Section IV, we present the results of our work, and in Section V our conclusions and potential future directions are discussed.

To the best of our knowledge, this work is the first effort towards using the IPC for evaluating PEPA models of static resource allocations in heterogeneous parallel computing environments to determine the robustness of mappings of applications onto machines. Moreover, it also represents the first effort to evaluate and identify more informed makespan targets for the robustness of resource allocations and its metrics, while scaling the size and complexity of the underlying models.

II. Background and Related Work

A. Performance Evaluation Process Algebra (PEPA)

The Performance Evaluation Process Algebra (PEPA) was developed to evaluate systems best described through the use of dynamic properties, such as, modeling the number of resources available to fulfill requests in a queue-based scheduling system, or modeling the runtime performance on a cluster of parallel computing resources [8], [9], [13], [14]. PEPA utilizes stochastic Markov processes to model the state changes of the random variables that represent the evolution of dynamic systems. Markov processes are probabilistic in nature and model the probability of a component being in its current state based on inputs from its possible previous states [13], [15].

Continuous time Markov chains (CTMC) were chosen instead of discrete time Markov chains (DTMC) for describing stochastic system evolution in PEPA, because a continuous time (CT) representation more accurately addresses modeling when events are both countably finite and occur at non-specific time intervals than does a discrete time (DT) representation [14]. This is because events in models utilizing a CTMC representation are evaluated as each event occurs, while those events in models with a DTMC representation are evaluated at predefined or discrete time intervals [14]. The DTMCs do not support modeling of systems with concurrent behavior, while CTMCs allow a more accurate time representation for concurrent systems [16]. By evaluating a CTMC model at the occurrence of each event, one can more accurately model systems with many parallel agents acting independently. Markovian models have previously been utilized to perform numerical analysis of computer system behavior [14], [17].

As Markov chains grow increasingly larger, there can be a corresponding increase in the number of finite states which must be evaluated, and this leads to the *state space explosion* problem, where an increasing amount of states must be evaluated in order to generate a solution [13], [18]. This explosion in the number of states that have to be explored to generate a solution leads to an increase in the time and memory requirements, making standard, brute-forced based solutions generally unfeasible, thus limiting the scalability of the algorithm used [18]. Additional literature and background information regarding PEPA for performance modeling can be found in [1], [8], [9], [13], [15], [37].

PEPA provides an abstraction to the complex CMTC evaluation and provides the modeler with a consise set of modeling algebra. Initially used for modeling the performance of grid-based scheduling algorithms used to process National Weather Service data in Great Britain [9], the PEPA framework has been extended to provide static evaluations for a number of other concurrent systems. These include the modeling of algorithmic performance on sets of possibly limited resources [10] and online resources in a grid framework [11]. Further work extended PEPA to allow incorporation of components aware of their own state, which allowed construction of component specific feedback into the model [12].

B. Robustness

Initial work on robust scheduling was conducted utilizing frameworks for job-shop application scheduling [19]–[22]. To solve the NP-Hard robust scheduling problem (RSP) for a robust schedule of N independent jobs on a single machine, a standard branch and bound approach was used [23]. To obtain an initial robust resource allocation, optimization techniques, including both stochastic mixed integer programming [24] and iterative integer programming [25] have been used. In conjunction with the generation of these robust resource allocations and metrics [26]–[28], work has been done 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 [29]. 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 [29] [30]. 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) [31] [32].

C. Mapping Applications to Machines

The number of ways to divide a set of n labelled objects (e.g. applications) into k unlabelled subsets (e.g. machines) is a power set relationship [33] and is given by (1).

$$k^n \tag{1}$$

A Stirling set number is the number of ways to partition n labelled objects (e.g. applications) into k nonempty, unlabelled subsets (e.g. machines) (*Stirling Number of the Second Kind*) [34] and is given by (2). Special cases arise when $k = n$ (3) and when $k = 1$ (4).

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n \tag{2}$$

$$\left\{ {n \atop n} \right\} = 1 \tag{3}$$

$$\text{where } n \geq 1, \left\{ {n \atop 1} \right\} = 1 \tag{4}$$

Table I shows the rapid growth of the number of mappings in both of these cases (empty and nonempty machines), and as an example of scale, the number of possible mappings in both cases quickly exceeds the estimated number of atoms in the observable Universe ($\leq 10^{82}$) [35].

TABLE I
NUMBER OF POSSIBLE MAPPINGS OF APPLICATIONS TO MACHINES

# Applications n	**# Machines** k	**# Mappings**[a]	**# Mappings**[b]
4	1	1.00	1.00
8	2	2.56×10^2	1.27×10^2
16	4	4.29×10^9	1.72×10^8
32	8	7.92×10^{28}	1.75×10^{24}
64	16	1.16×10^{77}	4.23×10^{63}
128	32	4.56×10^{192}	9.76×10^{156}

[a] n labelled applications mapped to k unlabelled machines.
[b] n labelled applications mapped to k unlabelled, nonempty machines

III. Methodology: Performance Modeling Using PEPA for Robustness Analysis

Our previous work evaluated the robustness of static resource allocations for 20 applications subject to perturbations at runtime mapped onto 5 machines using the PEPA Eclipse Plug-In [14]. This test set was chosen to mimic the execution scenario of simulation-based experiments in earlier work [28]–[30] for developing proof of concept and validation of our PEPA-based modeling and robustness analysis. As noted, the PEPA Eclipse Plug-In limits the size and complexity of the systems that can be modeled and the ability to automate the execution of the underlying model due to the human interaction necessary to setup and configure the associated GUI. As a method to overcome these limitations, we explore the use of the IPC as an alternative. The IPC is a compiled command line interface (CLI) based program developed in the Haskell programming language, and based on these factors, we hope to use the IPC to analyze larger and more complex models. We also hope to apply automation to the process by the elimination of some of the human interaction required by the PEPA Eclipse Plug-In GUI. To validate the IPC as a drop-in replacement for the PEPA Eclipse Plug-In, we replicate our previous work conducted with the PEPA Eclipse Plug-In using IPC.

To ease the barrier of entry for other researchers seeking to utilize the IPC for their own work, we build it as a Singularity container and add it to our existing online repository of PEPA related resources [36]. Containers and containerization frameworks have rapidly gained adoption as part of a microservice-based movement towards developing reproducible research artifacts. Containerized applications are executable on any platform where the underlying containerization framework can be installed. We have chosen to use Singularity containers to align a growing number of academic HPC centers and to align with previous work to develop an accessible repository for PEPA related work [36].

Another limitation of our previous work is that we have a small set of available resource allocation models for robustness analysis. To overcome this limitation, we develop a programmatic approach for the automatic generation of PEPA models for n of applications subject to perturbations at runtime mapped onto k machines. To generate rates and perturbed rates for these generated applications, we use the rates and perturbed rates from applications in our previously published work to construct a statistical model to use in our programmatic generation of `*.pepa` models with Python.

Combining this programmatic approach to generating PEPA models for resource allocations with our IPC container allows us to evaluate the effectiveness of our automated approach for analyzing both a larger number and larger complexity of resource allocation models than previously performed. Through this combination, populations of increasingly larger and more complex resource allocation models are analyzed to gain insights into the variation in the model population makespan time and provide insights into robustness as a metric as model complexity scales.

IV. Results and Analysis: Robustness Analysis Via Numerical Modeling of Resource Allocations

In the following subsections, we present validation that our containerized installation of the Imperial PEPA Compiler successfully replicates previous work conducted with the PEPA Eclipse Plug-In in Section A. In Section B, we discuss the differences when analyzing a fixed set of applications with rates and perturbed rates drawn from the uniform distribution versus a population of applications with rates and perturbed rates drawn from the uniform distribution. Section C highlights the improved scalability possible when utilizing the IPC for analyzing PEPA models of increasing complexity.

A. Replication of Previous Work Using IPC

To facilitate in-depth analysis of the robustness metric at scale, we constructed a Singularity container for the IPC [7] and added it to our existing online collection of PEPA containers [36] to reduce the burden of entry for researchers seeking to utilize PEPA. The IPC was developed concurrently with the PEPA Eclipse Plug-In, and as for the analysis of a single PEPA model, the level of effort between the use of the IPC and the PEPA Eclipse Plug-In for the analysis of a single `*.pepa` file is roughly equivalent. However, because the IPC is a compiled program executed through a CLI, it is more readily incorporated into automated, CLI-based workflows and the simulation of a large number of PEPA models can be conducted.

To verify that our IPC container produces the same results as the PEPA Eclipse Plug-In, we attempt to replicate the work performed in [14] where 20 applications with pertubations expressed at runtime are mapped to and executed on 5 machines for two cases, Colorado State University (CSU) Case 1 and CSU Case 2. Both cases model a parallel computing system comprised of 20 independent applications, 5 heterogeneous machines, and a set of 3 heterogeneous sensors described in detail in [28]. The results for CSU Case 2 are not shown due to space limitation, but available online. The application rates represent the ideal rate for an application to complete an assigned workload, and the perturbed rates represent the runtime variation in the application workload with respect to the ideal workload. The application rates and perturbed rates for CSU Case 1 are shown in Table II.

The cumulative distribution function (CDF) is significant as it provides the probability that a given mapping of applications onto machines will be complete at a given time. The CDF for CSU Case 1 Mapping A and Mapping B generated through the use of the PEPA Eclipse Plug-In are shown in Fig. 1. Table III shows the mapping of applications (n = 20) to machines (k = 5) for this work for both cases, and the CDF results for CSU Case 1 Mapping A and Mapping B generated using the IPC are shown in Fig. 2. The IPC-derived finishing times for the machines in CSU Case 1 are shown in Fig. 3. These results successfully replicate the work previously presented in [14], and CSU Case 2 was also successfully replicated using the IPC. These results show that the IPC is a valid, drop-in replacement for the analysis of `*.pepa` models.

For CSU Case 1 (Fig. 3), the makespan time for the systems of applications mapped onto machines in Mapping A and Mapping B do not have significant overlap, but the robustness metrics for both mappings can be calculated by Eq. 5 [14].

$$\psi_A = \min_{\forall i,j \text{ paired as in Mapping A \& B}} \Pr\left[F_i\left(M_j, \lambda_i\right) \leq 45\right] \quad (5)$$

The applications for both Mapping A and Mapping B have similar rates with variation in their perturbed rates, and while they have similar system makespan times, the robustness values for the mappings vary significantly. This highlights the importance of developing informed makespan goals when evaluating mappings of applications onto machines for resource allocation mappings.

To overcome the limitations imposed by the small number of PEPA models that currently exist for this domain, we have constructed a Python program that generates several feasible mappings of applications onto parallel heterogenous machines.

TABLE II
APPLICATION (a) RATES (r) AND PERTURBED RATES (p) OF MAPPING A AND MAPPING B FOR CSU CASE 1 [14].

	CSU Case 1			
	Mapping A		Mapping B	
Application (a)	r	p	r	p
a_1	0.256	0.104	0.256	0.256
a_2	0.128	0.128	0.192	0.047
a_3	0.256	0.256	0.128	0.128
a_4	0.128	0.128	0.128	0.085
a_5	0.154	0.138	0.128	0.128
a_6	0.385	0.156	0.192	0.192
a_7	0.256	0.256	0.256	0.063
a_8	0.192	0.192	0.192	0.047
a_9	0.154	0.062	0.256	0.256
a_{10}	0.128	0.091	0.192	0.067
a_{11}	0.192	0.167	0.192	0.142
a_{12}	0.154	0.192	0.256	0.256
a_{13}	0.128	0.128	0.256	0.199
a_{14}	0.192	0.192	0.192	0.192
a_{15}	0.128	0.112	0.256	0.132
a_{16}	0.385	0.385	0.192	0.047
a_{17}	0.154	0.107	0.128	0.128
a_{18}	0.192	0.147	0.128	0.102
a_{19}	0.128	0.128	0.192	0.082
a_{20}	0.154	0.102	0.128	0.056

TABLE III
MAPPING A AND MAPPING B OF APPLICATIONS (a_i) TO MACHINES (M_j) FROM [14]

Machine	Mapping A	Mapping B
M_1	$a_5, a_9, a_{12}, a_{17}, a_{20}$	$a_3, a_4, a_5, a_{17}, a_{18}, a_{20}$
M_2	a_6, a_{16}	$a_2, a_{11}, a_{14}, a_{19}$
M_3	a_1, a_3, a_7	a_1, a_7, a_{13}
M_4	$a_2, a_4, a_{10}, a_{13}, a_{15}, a_{19}$	a_9, a_{12}, a_{15}
M_5	$a_8, a_{11}, a_{14}, a_{18}$	a_6, a_8, a_{10}, a_{16}

To generate a diverse pool of variation in workload values, the Python function, `random.uniform`(a,b) was used to generate application rates and perturbed rates, where different values of a and b were based on the application rates and perturbed rates presented in [14]. This Python program can generate PEPA models for mapping n applications onto k machines, and the variations in application workload and machine availability can be sampled from any probability distribution model that can capture the static features of real workloads and machine characteristics at given timepoints.

The build recipe for the IPC Singularity container and the Python script to generate PEPA mappings of applications to machines is available at https://github.com/williamssanders/pepa and the container is publicly available at https://www.singularity-hub.org/collections/2351.

B. Applications with Constant versus Variable Load Rates

We used our Python application to generate a set of 20 applications with perturbations at runtime (see Table IV) and then, while holding that set of application rates and perturbed rates constant, generated 1,000 random mappings of those 20 applications onto 5 heterogeneous machines

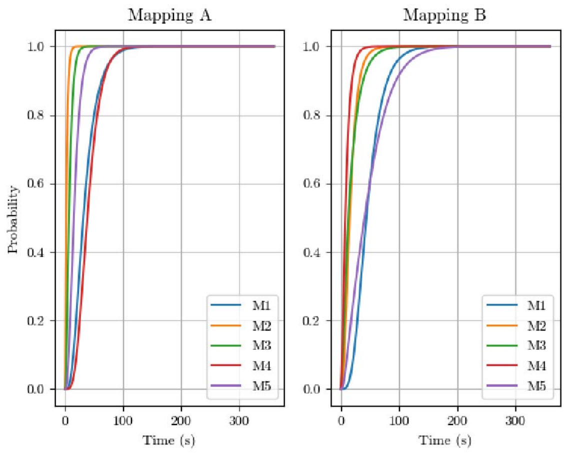

Fig. 1. PEPA Eclipse Plug-In Derived Cumulative Distribution Function of Mapping A and Mapping B for CSU Case 1.

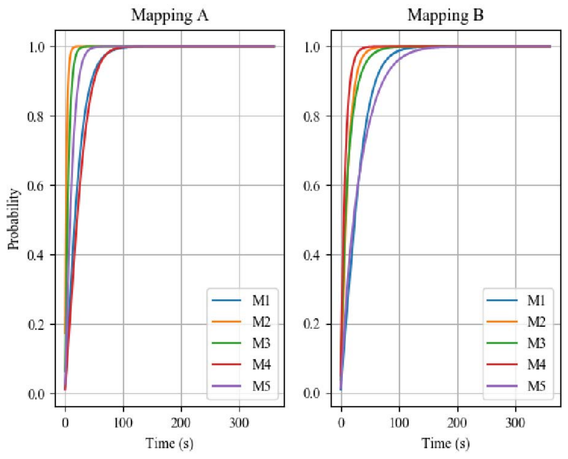

Fig. 2. Imperial PEPA Compiler Derived Cumulative Distribution Function of Mapping A and Mapping B for CSU Case 1.

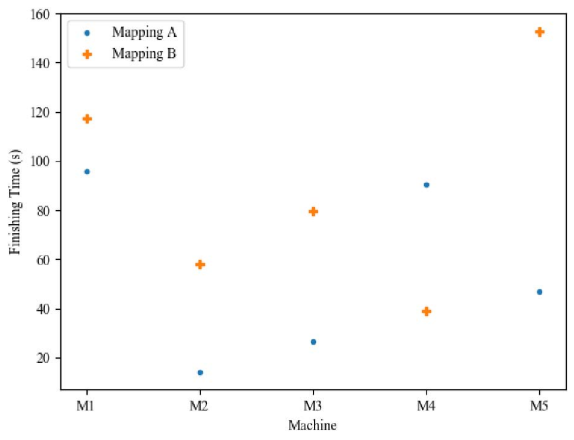

Fig. 3. Differences in Machine Finishing Times for Mapping A and Mapping B in CSU Case 1 Derived Using the Imperial PEPA Compiler.

TABLE IV
SET OF 20 APPLICATIONS WITH PERTURBATIONS AT RUNTIME USED FOR *a20m5_constant*

Application (a)	Rate (r)	Perturbed Rate (p)
a_1	0.361	0.391
a_2	0.580	0.632
a_3	0.335	0.499
a_4	0.558	0.605
a_5	0.598	0.632
a_6	0.517	0.536
a_7	0.611	0.621
a_8	0.506	0.538
a_9	0.216	0.333
a_{10}	0.570	0.587
a_{11}	0.264	0.327
a_{12}	0.232	0.284
a_{13}	0.333	0.547
a_{14}	0.086	0.541
a_{15}	0.649	0.651
a_{16}	0.311	0.611
a_{17}	0.232	0.605
a_{18}	0.459	0.570
a_{19}	0.556	0.624
a_{20}	0.622	0.633

(*a20m5_constant*). In addition, we generated 1,000 random mappings of 20 random applications with rates and perturbed rates randomly determined at runtime mapped onto 5 heterogenous machines (*a20m5_random*). In both cases, the application rates were generated based on a uniform distribution obtained from [14]. Violin plots showing the makespan distributions for these results are shown in Fig. 4. The Violin plots show the distribution of makespan values for a sampling ($N = 1,000$) models of n applications subject to perturbations at runtime mapped onto k machines. The plots show both the range of observed makespan values and the probability distribution that a given makespan value is observed. For Fig. 4, the values for n and k are $n = 20$ and $k = 5$. While the rates and perturbed rates for all the applications were derived from the same distribution, for the *a20m5_constant* case, it was observed that the overall makespan time for the distribution was significantly lower than that of the *a20m5_random* case. The *a20m5_random* case simulated a much larger number of applications exhibiting the same overall rate and perturbed rate statistical distributions.

The results shown in Fig. 4 highlight the value of examining significantly larger populations of resource allocation models, because the robustness metric is a probability that a given mapping of applications to machines will finish by a given makespan goal. As the median makespan of a population of mappings increases as more applications with rates derived from a common statistical distribution are sampled, this allows for a more informed selection of potential robustness metrics. Modeling a larger set of mappings of applications to machines shows that a robustness metric based on a makespan target for a small set of applications, by falling into a local minima or maxima, may not be an optimal metric for the population of applications that behave with similar rates.

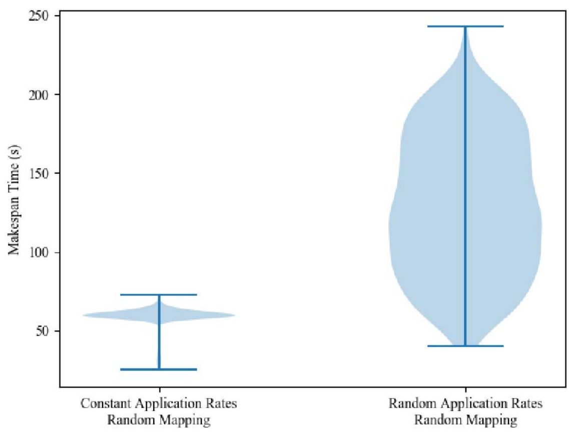

Fig. 4. Comparison of Model Makespan Time for 20 Applications (Constant vs. Random) Mapped onto 5 Machines. N = 1,000 for both cases. On the left, 20 applications with constant rates and perturbed rates generated randomly using the uniform distribution are mapped onto 5 machines, where the set of applications are held constant for all mappings (*a20m5_constant*). On the right, 20 applications with rates and perturbed rates generated randomly using the uniform distribution mapped onto 5 machines (*a20m5_random*).

C. Scalability

In order to emphasize the advantages of the IPC CLI-based approach, we wanted to confirm the ability to analyze large populations of mappings. We developed two sets of mappings (both with $N = 1,000$ mappings) for applications (a) where $a = 16$ onto machines (m) where $m = 1$, $m = 2$, ..., $m = 8$. For the first set of mappings, the set of applications were constant, with the application rates and perturbed rates determined from applications $a_1, a_2, ..., a_{16}$ in Table IV. The results for these mappings are shown in Fig. 5. The second set of mappings did not hold the set of applications constant, and the application rates and perturbed rates were generated through the uniform distribution for each mapping. The results for these mappings are shown in Fig. 6. Both Fig. 5 and Fig. 6 show that as the number of applications is held constant while increasing the number of machines available to process the workload increases, the median makespan time decreases as the number of machines increases.

The differences in the results of these two sets of mappings reinforces the importance of developing makespan targets for robustness metrics through analysis of populations of applications with similar rates and perturbed rates distributions as opposed to a smaller set of fixed applications which can be influenced by local minima or maxima. In Fig. 5, the population of makespan times for a fixed set of applications is higher than the population of applications with rates and perturbed rates determined randomly shown in Fig. 6. By developing makespan targets for robustness based on the population of applications with rates and perturbed rates determined randomly, we would have a better evaluation of robustness for the set of constant applications.

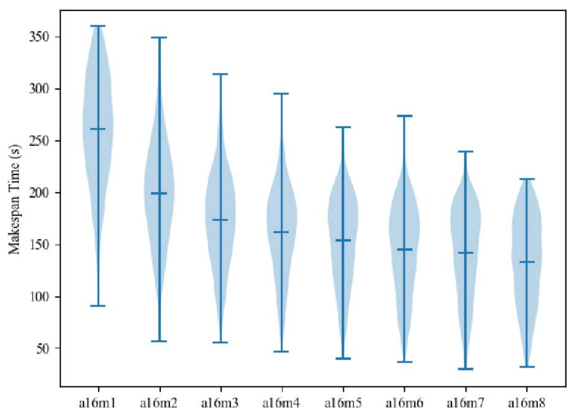

Fig. 5. Variation in Makespan Time With A Constant Number of Applications with Rates and Perturbed Rates from Table IV and an Increasing Number of Machines. As the resource allocation models being analyzed increase in the number of machines available to process a constant workload of applications, the makespan time decreases as the number of machines increases.

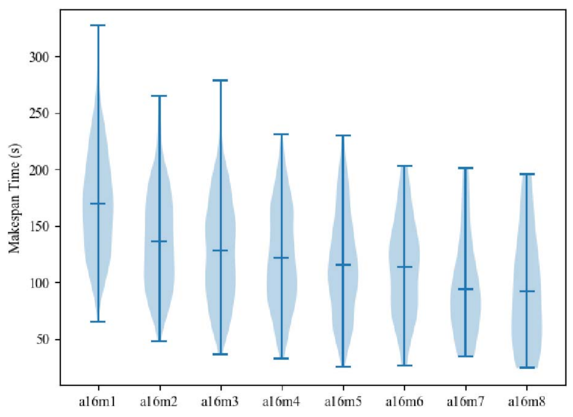

Fig. 6. Variation in Makespan Time With A Constant Number of Applications with Rates and Perturbed Rates Generated from the Uniform Distribution and an Increasing Number of Machines. As the resource allocation models being analyzed increase in the number of machines available to process a constant workload of applications, the makespan time decreases as the number of machines increases.

To evaluate the scalability of IPC for modeling `*.pepa` models of applications mapped onto machines, we developed a plan to model six (6) cases with increases in both the number of applications and the number of machines, and these cases are shown in Table V. For each of these cases, we generated 1,000 PEPA models for the mappings and analyzed the models using our IPC container. Violin plots for cases *a4m1*, *a8m2*, *a16m4*, *a32m8*, *a64m16*, and *a128m32* are shown in Fig. 7. As the size of the models increases, the underlying number of states necessary to compute the model increases correspondingly. As the number of states increases, the time required to generate the underlying CDF function for each machine being modeled also increases.

TABLE V
SCALABILITY OF MODELING THE ROBUSTNESS OF MAPPINGS OF APPLICATIONS TO MACHINES

Case	**# Applications** n	**# Machines** k	**# Possible Mappings**[a] $N = 1,000$
a4m1	4	1	1.00
a8m2	8	2	1.27×10^{2}
a16m4	16	4	1.72×10^{8}
a32m8	32	8	1.75×10^{24}
a64m16	64	16	4.23×10^{63}
a128m32	128	32	9.76×10^{156}

[a] n labelled applications mapped to k unlabelled, nonempty machines

Fig. 7 shows that as we increase the size of the resource allocation models being analyzed, the overall distribution of makespan times also increases. This reinforces the importance of examining distributions of applications with particular rate and perturbed rate behavior for establishing more informed and realistic makespan targets to use when deriving robustness metrics. As the overall size and complexity of the systems being modeled increases, the robustness metric derived from makespan goals should also increase to provide a more accurate representation of the overall system performance. It should be noted that the models for cases *a64m16* and *a128m32* were unable to be simulated using the PEPA Eclipse Plug-In, highlighting the advantage of the IPC based approach.

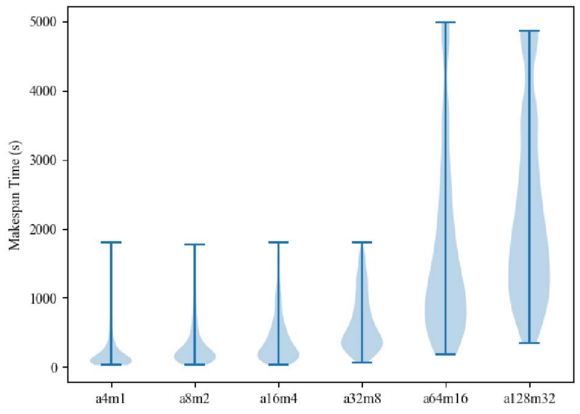

Fig. 7. Variation in Makespan Time With Increasing Numbers of Applications and Machines. As the resource allocation models being analyzed increase in size and complexity, the makespan time increases as model size and complexity increase.

V. CONCLUSIONS AND FUTURE WORK

Evaluating the robustness of heterogeneous parallel computing systems is necessary to reap the benefits of the expanded computational landscape. Through the use of analyzing populations of applications with rates and perturbed rates based on statistical distributions, we can better model and simulate the execution of scalable scientific applications in heterogeneous environments. PEPA modeling can be utilized in such a model-based framework for a predictive analysis towards selecting the most robust static resource allocation in a parallel computing system. The study performed in this work has allowed us to overcome some of the limitations in our previous work for robustness analysis for static resource allocations in parallel and distributed computing systems. Previously, an analysis was performed using a GUI-based interface (the PEPA Eclipse Plug-In), and was limited to a small set of previously published data for modeling. By changing our underlying PEPA execution framework from the PEPA Eclipse Plug-In to the Imperial PEPA Compiler, we have been able to both, overcome the limitations of model size and complexity imposed by the PEPA Eclipse Plug-In, and have also been able to achieve a larger scale of analysis through its compiled, CLI-execution. The Python program constructed to generate PEPA models of n applications mapped to k machines has allowed us to overcome the limitations imposed by a small set of existing models for better derivation and future analysis of robustness metrics based on machine makespan goal targets.

Two major outcomes are presented in this work. First, we have established that for deriving makespan goal based robustness metrics, it is critically important to base these goals on a larger analysis of applications to avoid goals biased towards local minima (or local maxima). An analysis of a larger population of applications following a given rate distribution can provide better insight into determining makespan targets for that population of applications. By evaluating populations of applications where the rates and perturbed rates follow a known statistical distribution, it is possible to derive makespan targets based on the statistical features of those populations. For example, utilization of the median makespan time for a population of applications with rates and perturbed rates generated from a known statistical distribution allows the automatic determination of an initial makespan target for robustness. This target could then be tuned based on system performance requirements and characteristics. Secondly, as the size and complexity of the parallel and distributed systems being modeled increases, there is a corresponding need to increase the makespan target used to derive a robustness metric for the system, as the overall makespan time for the systems increases with the complexity and size of the system.

Future work will include the development of additional statistical models for application rates and perturbed rates beyond the uniform distribution. This will allow us to further determine the characteristics needed for developing robustness metrics for populations of applications exhibiting the runtime behavior of those statistical distributions. Once additional populations of applications with rates are established, mixed populations of applications based on different statistical rate distributions could be constructed and analyzed, more accurately reflecting the state of shared parallel and distributed computing systems and other traditional HPC paradigms. This would allow for increasingly powerful robustness metrics to be established and utilized when evaluating a given mapping of applications to machines. In addition, real HPC data could be transformed and modeled with this framework to provide an even better insight into the development of mapping strategies.

Acknowledgment

The authors would like to acknowledge partial support of this work from the National Science Foundation (NSF) #1034897 grant and The Jackson Laboratory (JAX).

References

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

[2] 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.

[3] Oracle Corporation. "Guidelines for Java Heap Sizing." In Sun Java System Application Server 9.1 Performance Tuning Guide. Oracle Corporation. 2010. https://docs.oracle.com/cd/E19159-01/819-3681/abeij/index.html Accessed on:16-Jun-2020. [Online].

[4] I. Banicescu and S. Srivastava. "Towards Robust Resource Allocations via Performance Modeling with Stochastic Process Algebra." In Proceedings of the 18th IEEE International Conference on Computational Science and Engineering (CSE-2015), pp. 270 277, Porto, Portugal, October 2015.

[5] Srivastava, S., and Banicescu, I. "Robust resource allocations through performance modeling with stochastic process algebra." Concurrency and Computation: Practice and Experience, vol. 29(7): e3894. doi: 10.1002/cpe.3894. 2017.

[6] S. Srivastava and I. Banicescu. "PEPA Based Performance Modeling for Robust Resource Allocations Amid Varying Processor Availability." In proceedings of the 17th IEEE International Symposium on Parallel and Distributed Computing (ISPDC), Geneva, 2018, pp. 61-68.

[7] J. T. Bradley, N. J. Dingle, S. T. Gilmore, and W. J. Knottenbelt. "Derivation of passage-time densities in pepa models using ipc: The imperial pepa compiler." 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, MASCOTS 2003. pp. 344 - 351.

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

[9] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. "Scheduling skeleton-based grid applications using pepa and nws." The Computer Journal, vol. 48, no. 3, pp. 369-378, 2005.

[10] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. "Evaluating the performance of pipeline-structured parallel programs with skeletons and process algebra." Scalable Computing: Practice and Experience, 6(4):1–16, 2005.

[11] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. "Enhancing the effective utilisation of grid clusters by exploiting on-line performability analysis." 2005.

[12] A. Clark and S. Gilmore. "State-aware performance analysis with extended stochastic probes." EPEW 2008, LNCS 5261, 2008.

[13] J. Hillston. "Tuning systems: From composition to performance." The Computer Journal, vol. 48, no. 4, pp. 385 - 400, May 2005.

[14] Srishti Srivastava. "Evaluating the robustness of resource allocations obtained through performance modeling with stochastic process algebra." PhD dissertation, Mississippi State University, Department of Computer Science and Engineering, May, 2015.

[15] J. Hillston. "Process algebras for quantitative analysis." In *Proceedings of the 20th Annual Symposium on Logic in Computer Science (LICS'05)*, 2005.

[16] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. "Prism: A tool for automatic verification of probabilistic systems." In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 441–444. Springer, 2006.

[17] V.L.Wallace and R.S.Rosenberg. "Markovian models and numerical analysis of computer system behavior." In Proceedings of the Spring joint computer conference. ACM, 1966, pp. 141 - 148.

[18] Radek Pelánek. "Fighting state space explosion: Review and evaluation." In *International Workshop on Formal Methods for Industrial Critical Systems*, pages 37–52. Springer, 2008.

[19] E. Coffman and J. Bruno. "Computer and job-shop scheduling theory." A Wiley-Interscience publication. Wiley, 1976.

[20] D.Fernandez-Baca. "Allocating modules to processors in a distributed system." IEEE Trans. Softw. Eng., vol. 15, no. 11, pp. 1427 - 1436, Nov. 1989.

[21] 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, Apr. 1977.

[22] J. V. Leon, D. S. Wu, and R. H. Storer. "Robustness Measures and Robust Scheduling for Job Shops." IIE Transactions, vol. 26, no. 5, pp. 32 - 43, 1994.

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

[24] S. Gertphol and V. Prasanna. "Mip formulation for robust resource allocation in dynamic real-time systems." In Proceedings of the International Parallel and Distributed Processing Symposium, 2003.

[25] S. Gertphol and V. Prasanna. "Iterative integer programming formulation for robust resource allocation in dynamic real-time systems." In Proceedings of the International Parallel and Distributed Processing Symposium, 2004.

[26] A. Ghafoor and J. Yang. "A distributed heterogeneous supercomputing management system." Computer, vol. 26, no. 6, pp. 78 - 86, June 1993.

[27] M. A. Iverson, F. Ozguner, and L. Potter. "Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment." IEEE Trans. Comput., vol. 48, no. 12, pp. 1374 - 1379, Dec. 1999.

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

[29] 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, no. 7, pp. 630 - 641, 2004.

[30] S. Ali, J.-K. Kim, H. J. Siegel, and A. A. Maciejewski. "Static heuristics for robust resource allocation of continuously executing applications." J. Parallel Distrib. Comput., vol. 68, no. 8, pp. 1070 - 1080, Aug. 2008.

[31] I.Banicescu, F.M.Ciorba, and R.L.Carino. "Towards the robustness of dynamic loop scheduling on large-scale heterogeneous distributed systems." In Proceedings of the IEEE International Symposium on Parallel and Distributed Computing (ISPDC 2009), pp. 129 - 132, 2009.

[32] S. Srivastava, I. Banicescu, and F. Ciorba. "Investigating the robustness of adaptive dynamic loop scheduling on heterogeneous computing systems." In Proceedings of The 2010 IEEE/ACM International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW-PDSEC, On CD-ROM), pp. 1 - 8, April 2010.

[33] J. H. Conway, R. K. Guy. "Choice Numbers." The Book of Numbers. New York: Springer-Verlag, pp. 67-68, 1996.

[34] M. Abramowitz, I. A. Stegun (Eds.). "Stirling Numbers of the Second Kind." Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover, pp. 824-825, 1972.

[35] Planck Collaboration et al. Cosmological parameters. Astronomy & Astrophysics. Vol 594. pp. 63, 2016.

[36] 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." In proceedings of the 2019 IEEE/ACM International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW-HIPS), Rio de Janiero, Brazil. 2019.

[37] PEPA Website. "PEPA - Performance Evaluation Process Algebra." 17-Oct-2014. [Online]. Available: http://www.dcs.ed.ac.uk/pepa/. [Accessed: 23-Mar-2020].