Performance Modeling of Scalable Resource Allocations with the Imperial PEPA Compiler

William S. Sanders
The Jackson Laboratory

Srishti Srivastava
University of Southern Indiana

Ioana Banicescu
Mississippi State University

21st International Symposium on Parallel & Distributed Computing

ISPDC 2022

Basel, Switzerland

July 11-13, 2022

Presentation Overview

- Background & Motivation
  - Process Algebras
  - The Imperial PEPA Compiler
  - Prior Work with Performance Modeling of Static Resource Allocations
- Methodology & Results
  - Experimental Design
  - Scaling
  - Construction of a Heuristic
- Conclusions
- Future Work

Process Algebra

CCS, CCP, and PEPA

Process Algebra

Process algebras (or process calculi) are mathematical constructs used to model systems of concurrent processes and can be utilized to obtain qualitative and quantitative information about the modeled systems. With a set of atomic actions specified in a process algebra, more complex actions can be constructed. Process algebras can be viewed as *"the study of concurrent processes, their equational theories, transition systems, and the equivalencies between the systems.“*

Historical Process Algebras:

- Calculus of Communicating Systems (CCS) (Milner 1980)
- Communicating Sequential Processes (CSP) (Hoare 1984)

Calculus of Communicating Systems (CCS)

CCS was developed by Robin Miller in 1980.

Process algebra operators for constructing agents:

- *Action prefixing* (·): $a \cdot P$ denotes process P can only become active after action a.
- *Choice* (+): P and Q are processes, so is $P + Q$. An action from P will preempt an action from Q and vice versa.
- *Parallel composition* (|): Given P and Q, $P \mid Q$ denotes a system in which P and Q may operate independently or communicate complimentarily.
- Restriction (\): Σ is a set of actions. $P \setminus \Sigma$ denotes the set of actions P is restricted from performing.

- Relabeling: P and Q are similar, and can be mapped to each other with a transformation function. P can be relabeled as Q.

Quantitative processes can not be modeled with CCS.

Communicating Sequential Processes (CSP)

CSP evolved from CCS and was introduced by Anthony Hoare in 1984, and designed to simplify CCS.

Process algebra operators for constructing agents:

- *Prefix* (a → P): *a* is an event in the alphabet of process *P*, so a process performing *a* behaves as *P*.
- *Choice*
  - *Non-deterministic choice* (π): The choice between *P* and *Q* is decided by the system itself, and the environment has no control over the choice.
  - *Deterministic choice* (+): Similar to CCS, *P* + *Q* indicates the system can behave as either process *P* or process *Q*.
- *Parallel Composition* (|): *P* and *Q* can occur concurrently.
- *Hiding (abstraction)* (\ ): *A* is the alphabet of events of *P* that are not visible outside of *P*, denoted as *P* \ *A*.

Quantitative processes can not be modeled with CSP.

Performance Evaluation Process Algebra (PEPA)

PEPA was developed by Jane Hillston in 1991. Based on stochastic Markov processes, it addresses the lack of quantitation in CCS and CSP.

Process algebra operators for constructing agents:

- *Prefix* ($\cdot$): Activity a, and activity rate r, $(a,r) \cdot P$
- Choice (+): A choice between competing components
- Cooperation ($\bowtie$) $P \bowtie Q$ denotes concurrent activities of P and Q
- Hiding ($\setminus$): A set of components L that are unknown to process P, defined as $P \setminus L$

PEPA and various PEPA tools have been used for modeling a variety of concurrent systems

Continuous Time Markov Chains in PEPA

- Continuous time Markov chains (CTMC) were chosen as the underlying execution framework in PEPA because a continuous time (CT) representation more accurately addresses modeling parallel and distributed systems with events that are both countably finite and that occur at non-specific time intervals than does a discrete time (DT) representation.
- Events in models utilizing a CTMC representation are evaluated as each event occurs, where DTMC-based models are evaluated at predefined or discrete time intervals.
- DTMCs do not support modeling of systems with concurrent behavior, while CTMCs allow a more accurate time representation for concurrent systems.
- Evaluating a Continuous Time model at the occurrence of each event, one can more accurately model systems with many parallel agents acting independently.

Why Process Algebras?

Alternative Methods:

- Stochastic Petri Nets (SPNs)
- Queuing Networks

Process algebras offer significant improvement over these methods because:

- Compositionality
- Quantitative Analysis
- Wide Acceptance

IPC

The Imperial PEPA Compiler

The Imperial PEPA Compiler (IPC)

- Developed in 2003, IPC is a PEPA model execution framework independent of the PEPA Eclipse Plug-In

  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

- Implemented in Haskell, and compiled before execution, unlike the PEPA Eclipse Plug-In
- IPC has not been as widely adopted as the PEPA Eclipse Plug-In, nor has it seen recent updates

Prior Work & Motivations

Prior Work

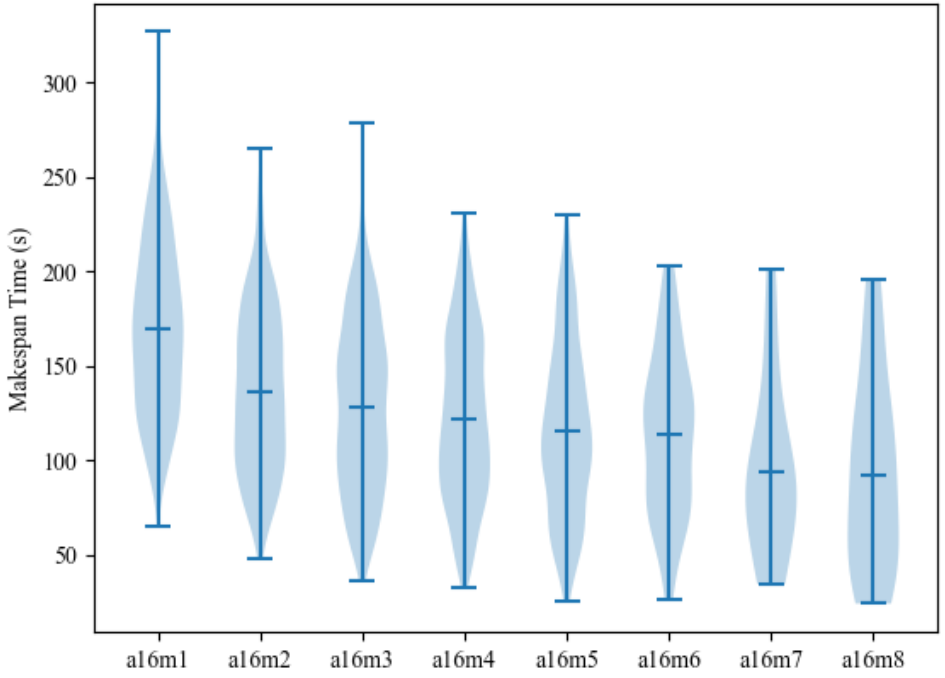
- Previous work utilized the PEPA Eclipse Plug-In to model parallel and distributed systems consisting of applications subject to perturbations at runtime mapped onto machines with varying availability to determine the robustness of a given mapping of applications onto machines
- A mapping is considered robust with respect to specific system performance features (i.e. Makespan) against perturbations if the degradation of these features is constrained when limited perturbations occur

Prior Work (continued)

- Limited Number of Models
  - Prior work was constrained by a limited number of published and available models that represent sets of applications subject to perturbations at runtime mapped onto machines
  - The small number of available models limits the insights able to be drawn into defining more informed metrics for makespan-based robustness metrics
- Model Size Limitations
  - For those models that are available, the PEPA Eclipse Plug-In is unable to execute models above a certain size, limiting our efforts to small systems that fail to capture the complexity of large parallel and distributed computing systems

Prior Work Validated IPC as a Replacement for PEPA Eclipse Plug-In

- IPC was able to replicate prior results obtained using the PEPA Eclipse Plug-In
- IPC was shown to be able to analyze larger models than previously possible with the PEPA Eclipse Plug-In



As the number of machines increases, the overall makespan time decreases

Our Motivation for This Work

- Determine Upper Limits of Scalability
  - IPC was shown to be able to simulate larger models than previously possible with the PEPA Eclipse Plug-In
  - There has not been a systematic analysis of the size limits of models where systems of applications mapped onto machines where both the number of applications and the number of machines increase
  - The small number of available models limits the insights able to be drawn into defining more informed metrics for makespan-based robustness metrics
  - These small models have limited efforts to small systems that fail to capture the complexity of large parallel and distributed computing systems

Experimental Design

$$m = 2^i \quad | \ 0 \le i \le 6$$
$$a = 2^{(i+j)} \quad | \ 1 \le j \le 6$$

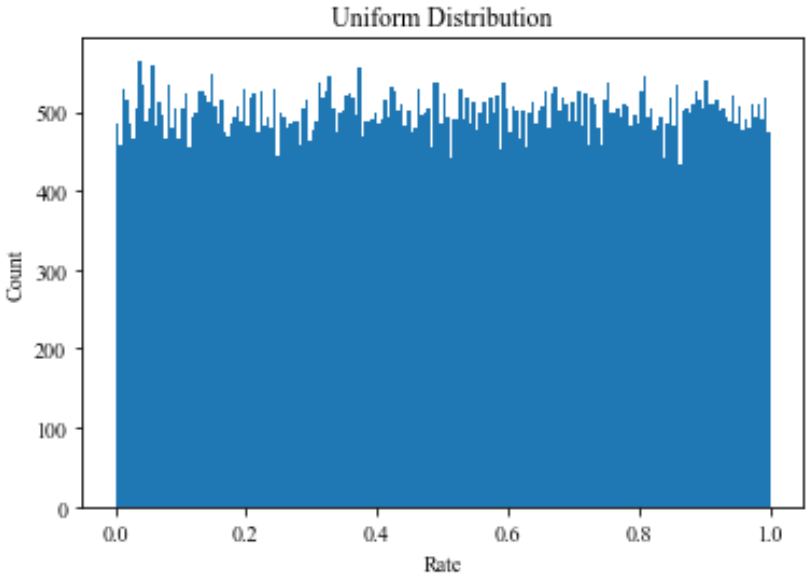
This generates 42 (m, a) pairs, ranging from (1, 2) to (64, 4,096).

For each pair, we generate 1,000 PEPA models (N = 1,000) with the application rates and perturbed rates derived from the uniform distribution and a random mapping of applications to machines.

We evaluate each model using the IPC with the following parameters:

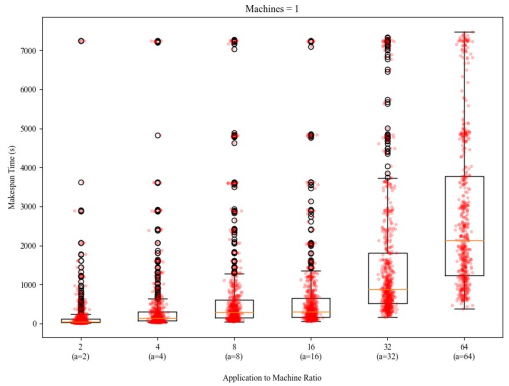
$$timeStep = 5$$
$$stopTime = 7{,}500$$

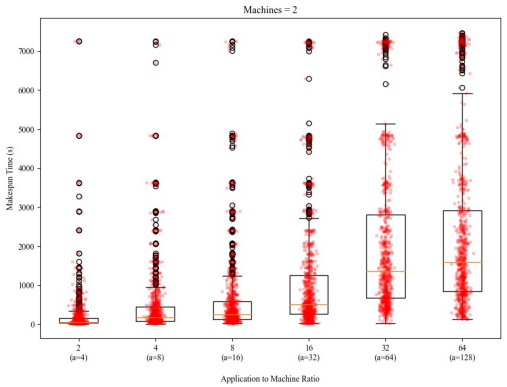
<u>42,000 PEPA Models</u>

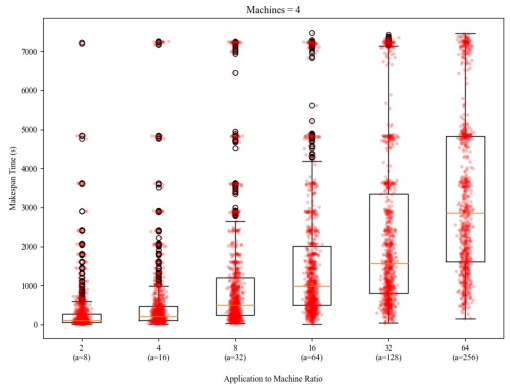


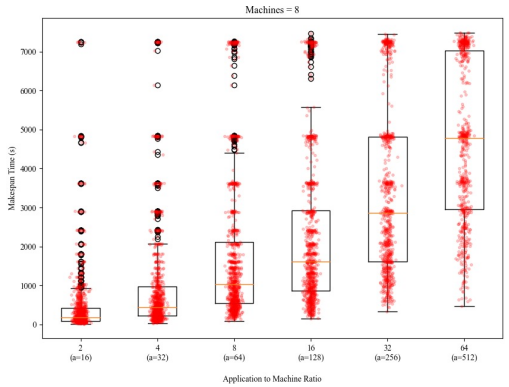
m	a	(m, a)	a / m
1	2	(1, 2)	2
1	4	(1, 4)	4
1	8	(1, 8)	8
1	16	(1, 16)	16
1	32	(1, 32)	32
1	64	(1, 64)	64
2	4	(2, 4)	2
...			
64	128	(64, 128)	2
...			
64	4096	(64, 4096)	64

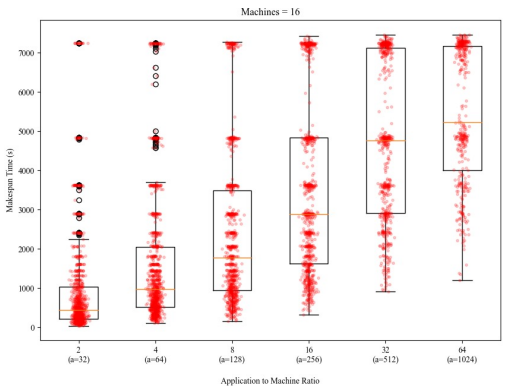
Simulation Results

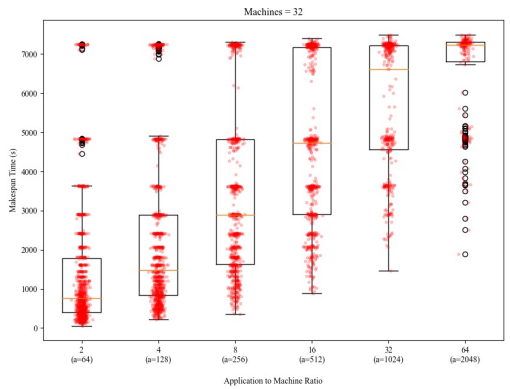


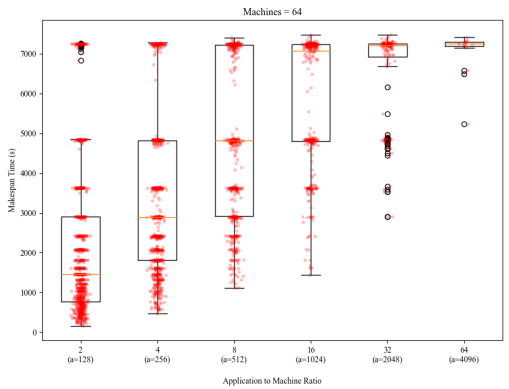






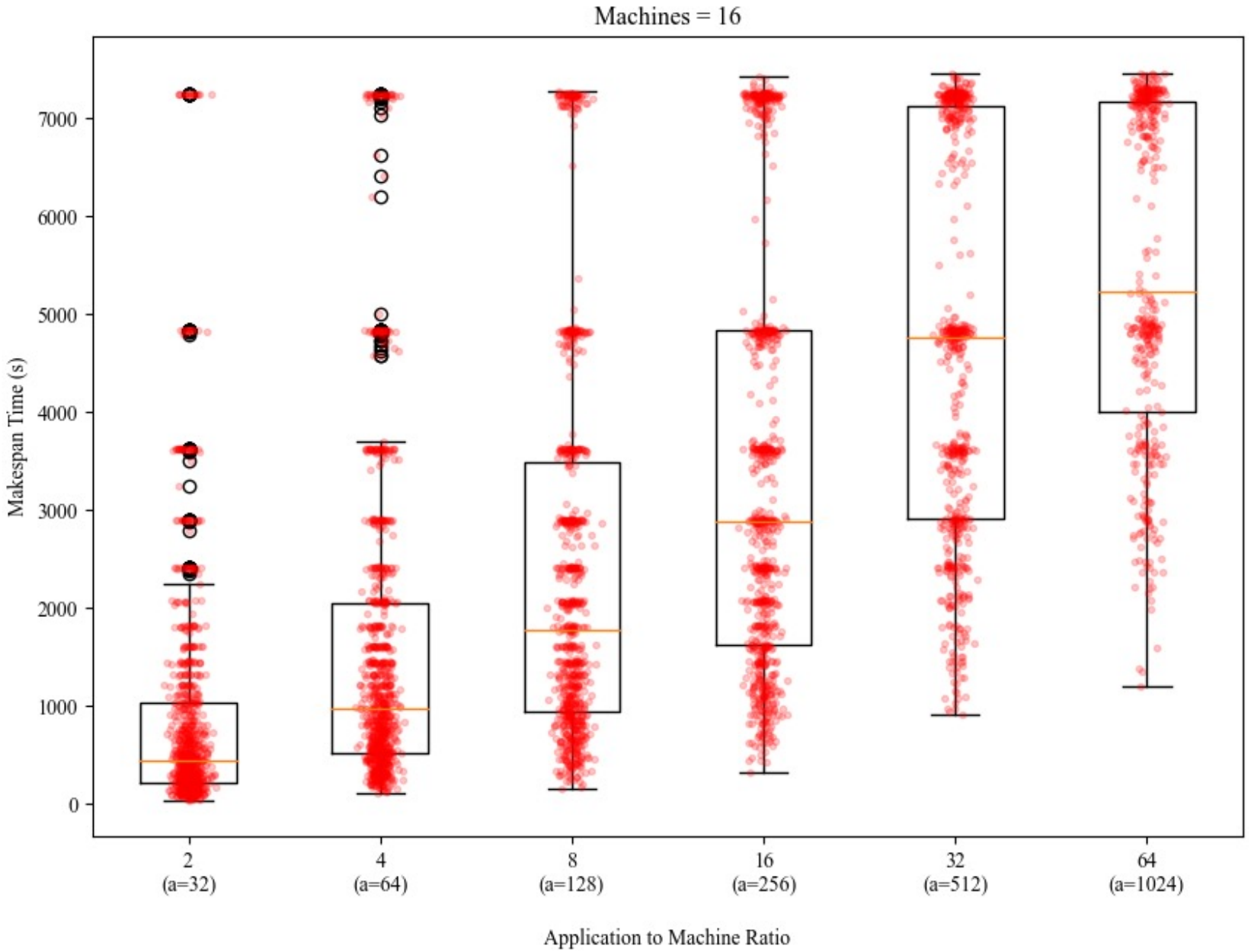




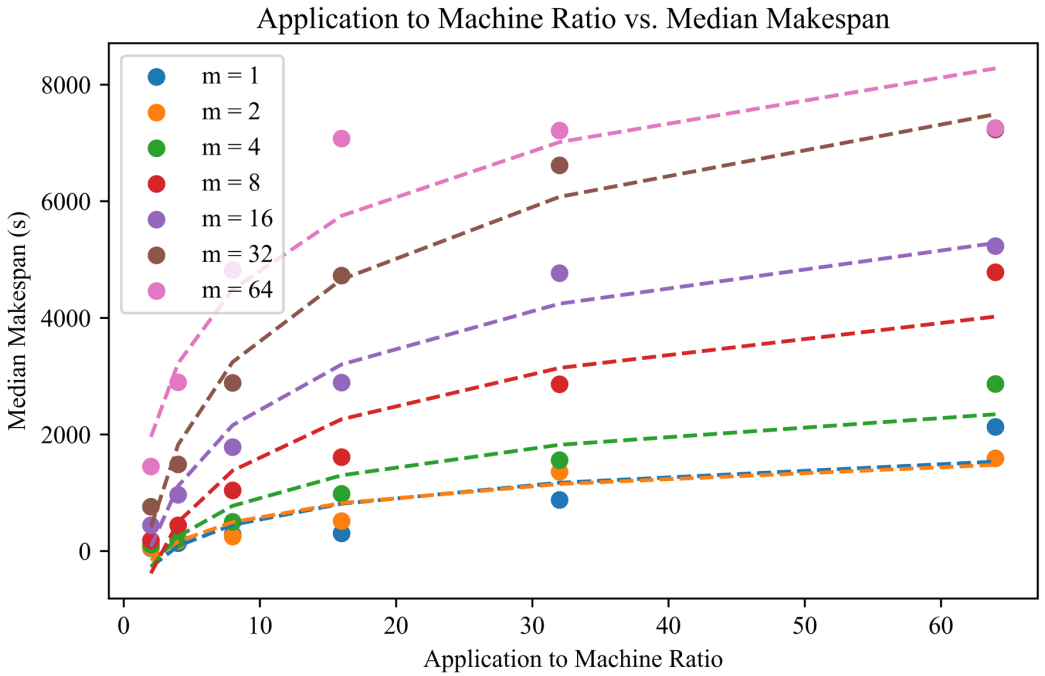


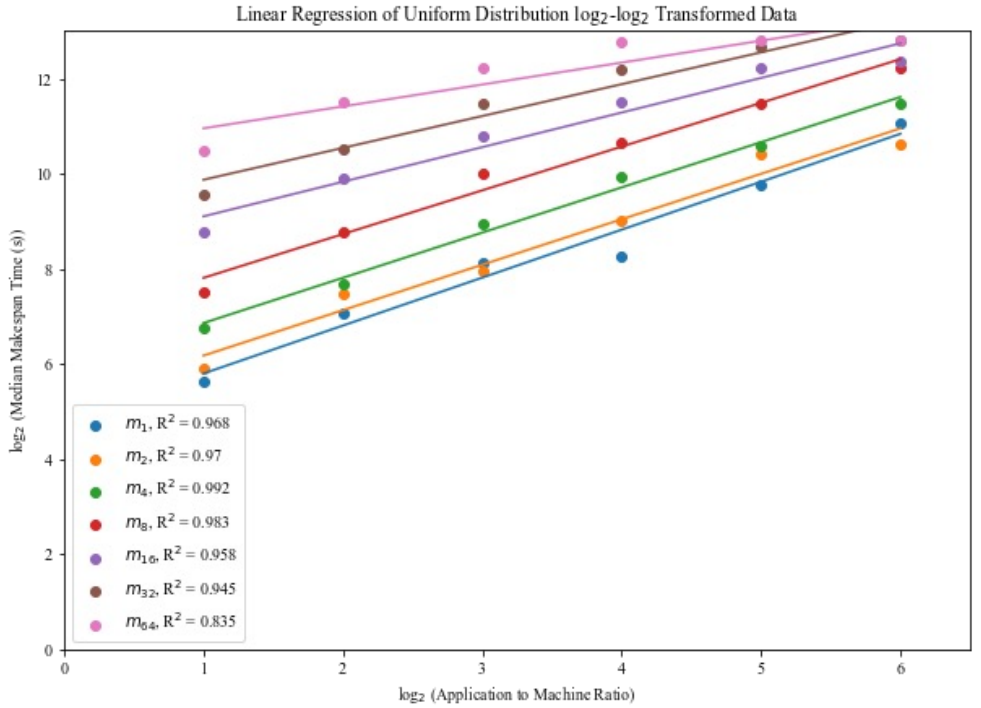
$m = 16$

- Each red point represents the makespan time for a single PEPA model
- Each box plot is comprised of 1,000 points for its number of applications mapped to machines



Simulation Results

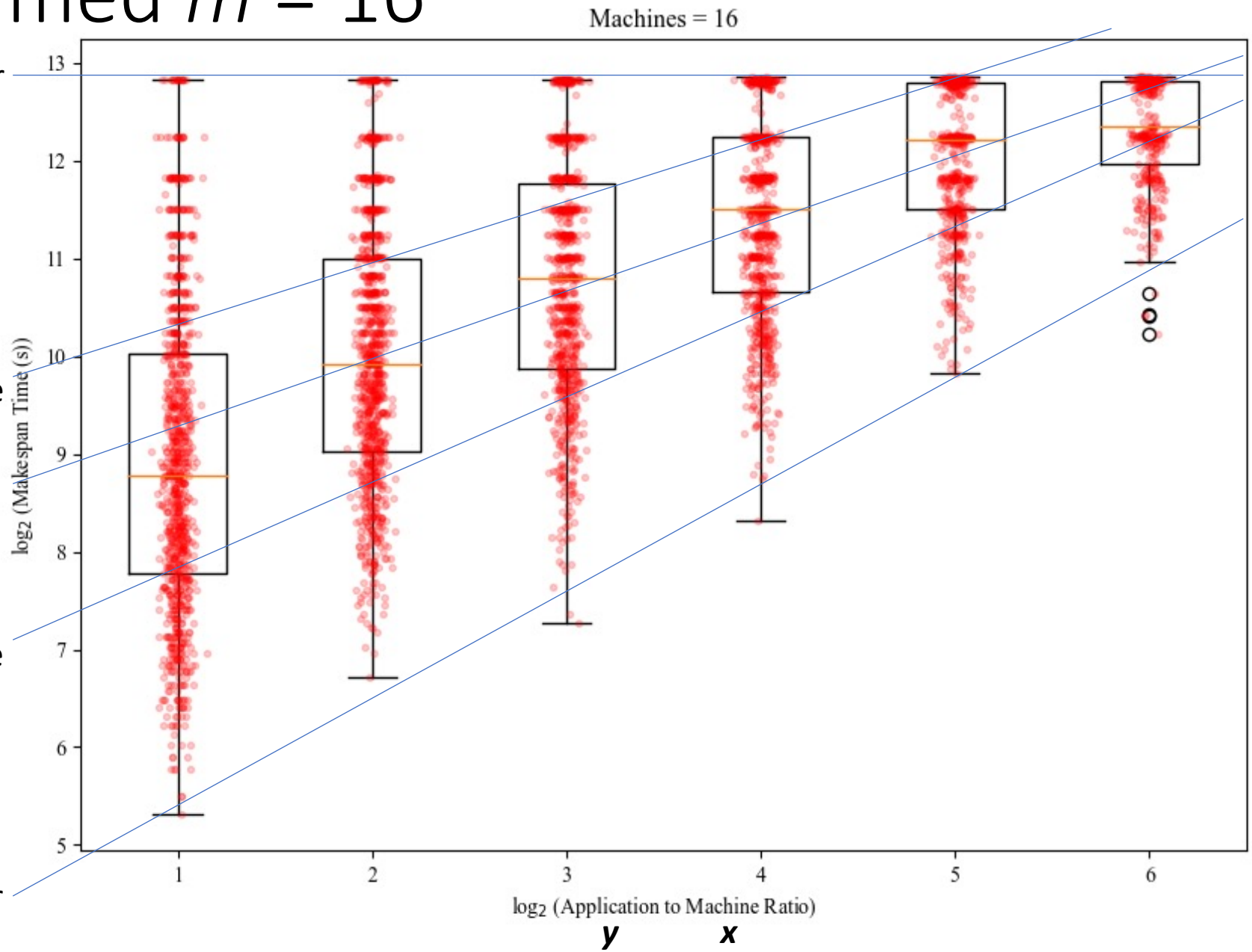




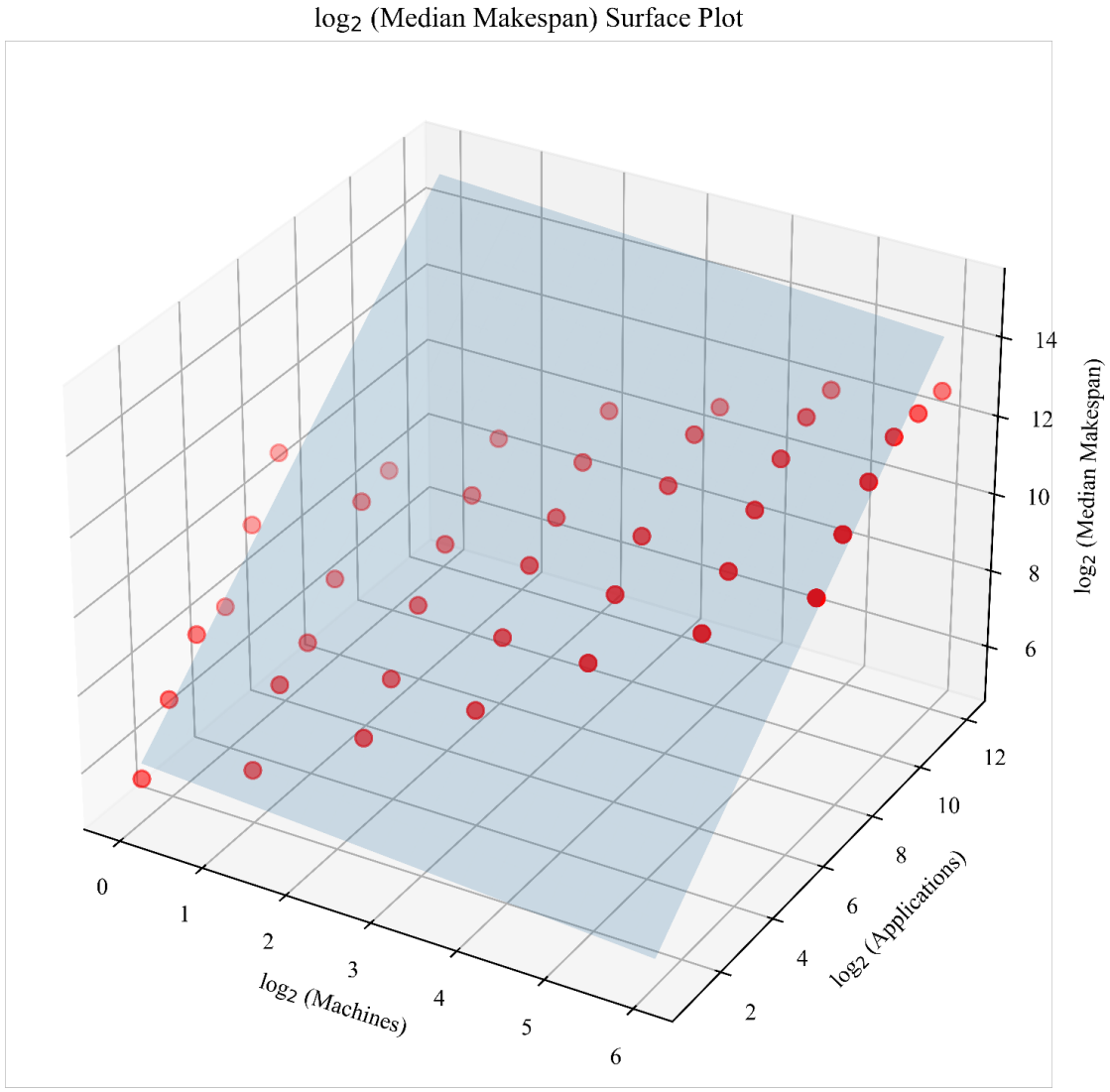
$\log_2$-$\log_2$ Transformation

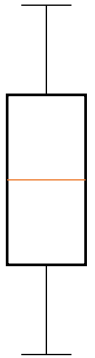
$\log_2$ Transformed m = 16

- $\log_2$-$\log_2$ transformed results are very linear
- Large number of data points allows for consistent determination of statistical features
- 2D data analysis prohibitive of full utilization of experimental design



3D Surface Fit





	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	5.238
lower_quartile (≤ 25%)	-0.144	0.887	3.989
lower_whisker (≤ 5%)	-0.007	0.890	1.404

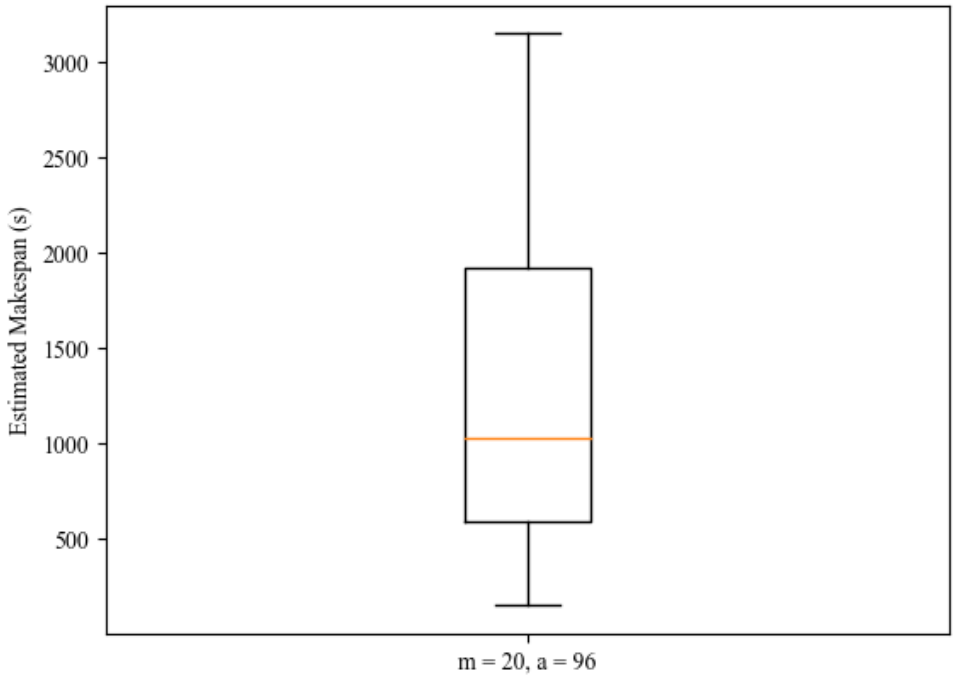
$$z = mx + ny + b$$

$x = \log_2$ (machines)
$y = \log_2$ (applications)
$z = \log_2$ (makespan)

Makespan Estimation Heuristic

```python
def uniform_makespan(m, a):
    values = []
    # Lower Whisker
    lw = (-0.00679399*math.log2(m))+ (0.88952122*math.log2(a)) + 1.40409444
    lw_makespan = 2**lw
    values.append(round(lw_makespan,3))
    # Lower Quartile
    lq = (-0.14357982*math.log2(m))+ (0.88717041*math.log2(a)) + 3.98924097
    lq_makespan = 2**lq
    values.append(round(lq_makespan,3))
    # Median
    median = (-0.13408127*math.log2(m))+ (0.81300426*math.log2(a)) + 5.23824067
    median_makespan = 2**median
    values.append(round(median_makespan,3))
    # Upper Quartile
    uq = (-0.11161406*math.log2(m))+ (0.66892942*math.log2(a)) + 6.98322805
    uq_makespan = 2**uq
    values.append(round(uq_makespan,3))
    # Upper Whisker
    uw = (-0.12839567*math.log2(m))+ (0.53997752*math.log2(a)) + 8.62053627
    uw_makespan = 2**uw
    values.append(round(uw_makespan,3))
    label_string = "m = " + str(m) + ", a = " + str(a)
    values_dict = [
        {
            'label' : label_string,
            'whislo': values[0],    # Bottom whisker position
            'q1'    : values[1],    # First quartile (25th percentile)
            'med'   : values[2],    # Median         (50th percentile)
            'q3'    : values[3],    # Third quartile (75th percentile)
            'whishi': values[4],    # Top whisker position
            'fliers': []        # Outliers
        }
    ]
    return values,values_dict
```

Given a number of machines (*m*) and a number of applications (*a*), we can generate an estimate of the expected distribution of makespan times when the application rates (*r*) and perturbed rates (*p*) follow the uniform distribution.



['lower_whisker', 'lower_quartile', 'median', 'upper_quartile', 'upper_whisker']
[150.351, 592.505, 1032.806, 1918.436, 3150.47]

Estimator Validation: $f(m,a)$

f(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	22.328%	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%

Five (m,a) pairs were generated and evaluated using the newly constructed heuristic. Cells in orange represent scenarios where the projected makespan time is grater than the duration of the simulation period.

The Q1, Median, and Q3 values are on average 6.21%±3.79% from their expected values. There is greater variation in the whiskers, and at the higher end, this is reflective of the simulation duration for the analysis.

Conclusions

- As both the number of applications and number of machines increase, the model makespan time also increases in a linear relationship when the the number of machines, the number of applications, and the model makespan time are $\log_2$ transformed
- A significant amount of variance in the model makespan time is dependent on both the number of applications and number of machines in the model being evalauated
- Constructed a heuristic that can be utilized for a given number of applications and machines to help inform makespan values to potentially help define a robustness target
- 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
  - Automatic determination of initial makespan targets for robustness
- 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

- Examination of alternative statistical models for application rates and perturbed rates beyond the uniform distribution
- Develop new datasets of application rates based on real production HPC data transformed and modeled into this framework to provide an even better insight into the development of mapping strategies
- Examination to determine if the additional variation present in the linear model is caused by machine overload, where due to load imbalance a small number of machines are executing the majority of applications

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 Pelanek. ``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].

Acknowledgements

Replication Resources:

- Source Code: https://github.com/williamssanders/pepa
- Container Resources: https://singularity-hub.org/collections/2351

Questions

shane.sanders@jax.org

https://github.com/williamssanders/pepa/