

# A Container-Based Framework to Facilitate Reproducibility in Employing Stochastic Process Algebra for Modeling Parallel Computing Systems

20-May-2019

William S. Sanders<sup>1,2</sup>, Srishti Srivastava<sup>3</sup>, and Ioana Banicescu<sup>1</sup>

<sup>1</sup> – Department of Computer Science & Engineering, Mississippi State University, MS, USA

<sup>2</sup> – The Jackson Laboratory, CT, USA

<sup>3</sup> – Department of Computer Science, University of Southern Indiana, IN, USA

# Overview

- Process Algebra
- Scientific Reproducibility
- Containerization
- Our Work
- Conclusions

# Process Algebra

CCS, CCP, PEPA, Bio-PEPA, GPEPA

# 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 1978)

# Calculus of Communicating Systems (CCS)

CCS was developed by Robin Miller in 1980.

Process algebra operators for constructing agents:

- *Action prefixing* ( $\cdot$ ):  $a \cdot P$  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  $Q$ ,  $P | Q$  denotes a system in which  $P$  and  $Q$  may operate independently or communicate complementarily.
- *Restriction* ( $\backslash$ ):  $\Sigma$  is a set of actions.  $P \backslash \Sigma$  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* ( $a \rightarrow P$ ):  $a$  is an event in the alphabet of process  $P$ , so a process performing  $a$  behaves as  $P$ .
- *Choice*
  - *Non-deterministic choice* ( $\pi$ ): 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)* ( $\backslash$ ):  $A$  is the alphabet of events of  $P$  that are not visible outside of  $P$ , denoted as  $P \backslash A$ .

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* ( $\backslash$ ): A set of components  $L$  that are unknown to process  $P$ , defined as  $P \backslash L$

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

# Bio-PEPA

Bio-PEPA is an extension to PEPA developed in 2008 by Federica Ciocchetta and the creator of PEPA, Jane Hillston, allowing the formal modeling of biochemical networks.

The components of these networks can be mapped to the components of inputs and outputs of an algorithmic pipeline with available resource limitations.

Further extensions to Bio-PEPA were developed to:

- Incorporate the occurrence of discrete events within the system
- Allow the modeling of biological compartments
- Translate Bio-PEPA models into continuous time Markov chains (CTMCs) and ordinary differential equations (ODEs)

# GPEPA

GPEPA, or Grouped PEPA, is an extension of PEPA that allows a change from CTMC time representation to instead use ODEs developed by Richard A. Hayden and Jeremy T. Bradley in 2010.

- More recent implementations of GPEPA include stochastic probes, a query mechanism for stochastic process algebras used to specify events that determine the start and end of a desired passage-time measure, where the probe specifies a set of source states and target states in the state space of the underlying stochastic process.

This change from CTMCs to ODEs, along with the requirement that GPEPA only model homogenous systems, allows an increase in complexity of the models that can be evaluated from models with  $10^{11}$  states to models with  $10^{129}$  states.

# Implementations of PEPA, Bio-PEPA, & GPEPA

- PEPA and Bio-PEPA are both implemented as plug-ins to the Eclipse Java Integrated Development Environment.
  - The PEPA Plug-in - <http://www.dcs.ed.ac.uk/pepa/tools/plugin/>  
*Recent extensions to PEPA allow ODE instead of CTMC representation with PEPA models.*
  - The Bio-PEPA Eclipse Plug-in - <http://homepages.inf.ed.ac.uk/jeh/Bio-PEPA/Tools.html>
- GPEPA is implemented through GPAnalyser, a stand-alone, CLI, Java-based execution framework.
  - GPAnalyser - <https://code.google.com/archive/p/gpanalyser/>

These tools require specific software and version dependencies to properly execute, sometimes requiring older versions of the software.

# Scientific Reproducibility

# Reproducibility

- Rising concern about lack of repeatability, replicability and reproducibility in science and engineering
- A number of high-profile publications in a large number of domains have not been successfully replicated when efforts to reproduce the work have been attempted
- Variation in data collection methodologies, experimental environments, computational configuration, the lack of detailed and intricate documentation, and more are leading to a call for enhanced peer review and validation of experimentally produced artifacts

# Artifacts

- Digital artifacts are produced during the course of research
  - Can be either inputs or outputs of a research project
  - Examples include input data sets, raw data, software systems, scripts to run experiments or analyze results
- ACM Digital Artifacts
  - ACM has initiated formal processes for artifact review to accompany publication

Artifacts Evaluated – Functional



Artifacts Evaluated – Reusable



Artifacts Available



Results Replicated



Results Reproduced



# Artifact Repositories

## Code Repositories

- Publicly available, generally commercial solutions:
  - GitHub
  - BitBucket
  - Sourceforge



## Data Repositories

- Generally domain specific and administered:
  - Biology: GenBank, NCBI Sequence Read Archive, etc.
  - Chemistry: chEMBL, caNanoLab, STREND, etc.
  - Earth, Environmental, and Space: NASA Goddard Earth Sciences Data and Information Services Center, PANGAEA
  - Astronomy: SIMBAD, UK Solar System Data Centre
  - Ocean Sciences: SEANOE, Australian Ocean Data Network

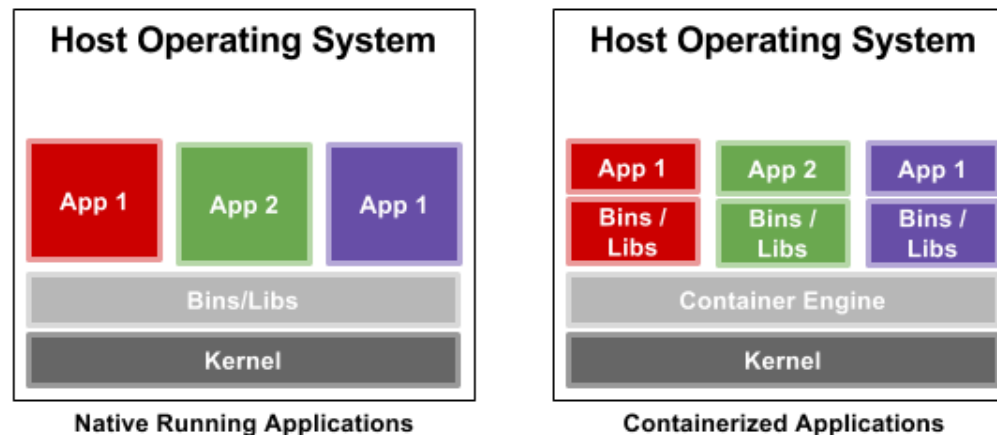
...



# Containerization

# Linux Containers

- Standard units of software that package up code and all dependencies so an application can run quickly and reliably from one computing environment to another
- Containers provide operating-system level virtualization through a virtual environment that has its own process and network space, as opposed to a full-fledged virtual machine



# Containerization

- Applications distributed as containers have the same installation overhead as a normal instance of the application, but with a slight additional overhead requiring knowledge of the containerization framework.
- Once the container is generated, the full burden of application installation and configuration is abstracted from the end users of application.
- Containers are easily migrated from system to system, and are easy to distribute and provide as digital artifacts

- Docker is the clear leader in containerization frameworks for cloud computing ecosystems



- Some concern about the security of Docker containers in non-cloud environments.
- Singularity is emerging as the containerization framework of choice in HPC environments.



# Singularity



- Developed in 2015 as an open-source project by researchers at Lawrence Berkeley National Laboratory led by Gregory Kurtzer
- Deployed at Ohio State University, Michigan State University, Texas Advanced Computing Center, San Diego Supercomputer Center, Oak Ridge National Laboratory, etc.

What makes Singularity different from other containerization solutions:

- Reproducible software stacks – container image allows easy verification
- Mobility of compute – containers can be transferred with standard tools
- Compatibility with complicated architectures – compatible with existing infrastructure and legacy systems
- Security model – security model of untrusted users running untrusted containers

# Singularity



- Enables users to have full control of their environment
- Can package entire scientific workflows, software, libraries, even data
- Empowers users, no need to ask your IT system admin to install anything
- Compatible with Docker
- Secure!

# Container Registries

- DockerHub (<https://hub.docker.org>)
  - Online repository of Docker container images, can be built on demand by various triggers
  - ~2,257,118 available container images
- SingularityHub (<https://singularity-hub.org>)
  - Developed and maintained by Stanford Research Computing and Stanford Medicine
  - ~4,096 containers across ~2,167 collections

# Containerization of Process Algebra Frameworks

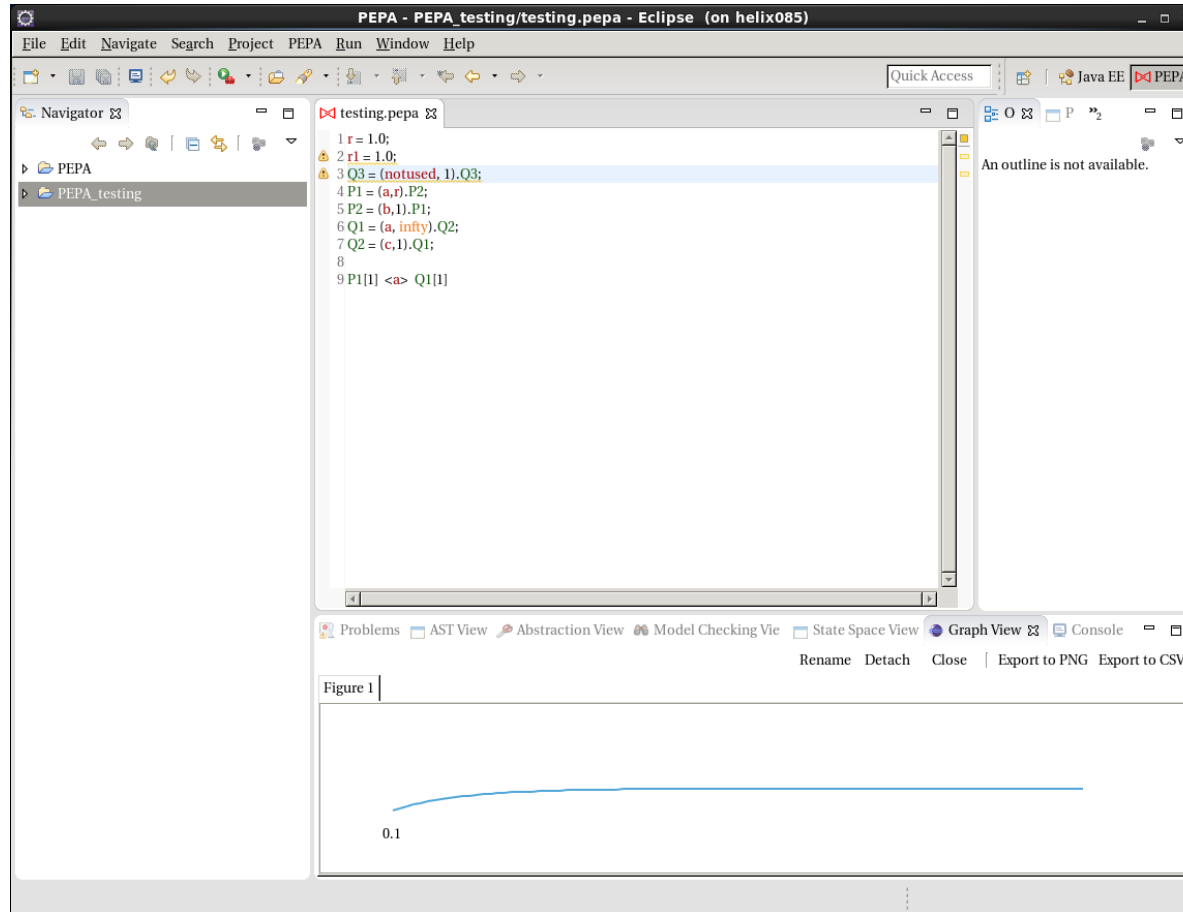
PEPA, Bio-PEPA, GPAnalyser (GPEPA)

# Why containerize PEPA and its derivatives?

- Like many applications developed in a research environment, process algebra frameworks have multiple, often difficult to identify dependencies, making them challenging for new researchers to install and utilize
- Singularity containers for PEPA, Bio-PEPA, and GPAnalyser were developed
  - To facilitate update of PEPA and its derivatives
  - Lower the barriers to entry for researchers while facilitating scientific reproducibility
- Each container was initially constructed using Singularity v2.5.2 running on CentOS Linux v7.4 deployed on HP Proliant SL series servers with 2.5 GHz 20-core 64-bit Intel processors with 256GB RAM
  - Functionality was verified on CentOS v7.6, Ubuntu LTS v18.04, Ubuntu LTS v16.04, LinuxMint v19.1, Debian LTS v9.6
  - Additional testing was performed on an n1-standard-8 machine instance (8 virtual CPUs and 30GB RAM) running CentOS v7.6 and Singularity v2.5.2 on the Google Cloud Platform



# Validation of The PEPA Eclipse Plug-in Singularity Container

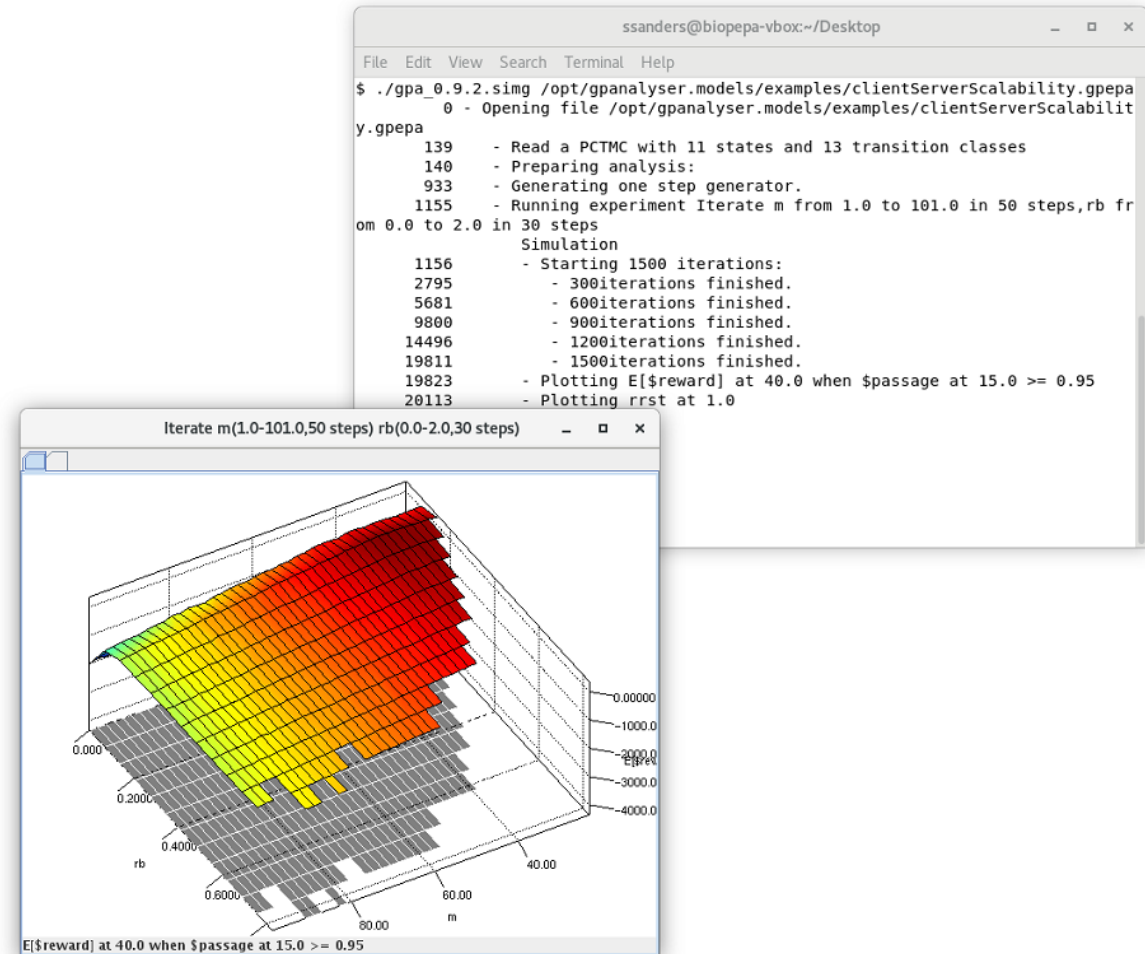


Application output for a simple PEPA model was verified by comparing against non-containerized version of the application.

*Not-shown: The Bio-PEPA Eclipse Plug-in*

# Validation of GPAnalyser (GPEPA) Singularity Container by Execution of the `clientServerScalability.gpepa` GPEPA example file

Application output was verified by comparing against non-containerized version of the application.



# Validation by Replication

To confirm that our containers allow true replication of previously published PEPA models, we performed an in-depth replication of a recent, previously published study.

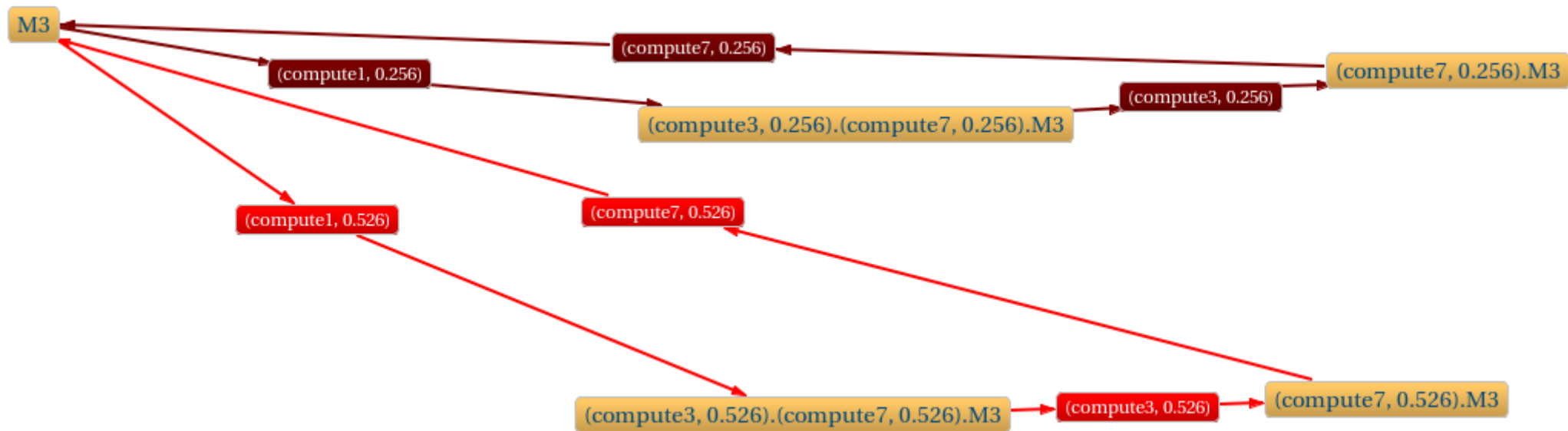
S. Srivastava & I. Banicescu. PEPA Based Performance Modeling for Robust Resource Allocations amid Varying Processor Availability. 17<sup>th</sup> International Symposium on Parallel and Distributed Computing (ISPDC). Geneva, Switzerland. IEEE Xplore. 2018.

- Models the performance of a number of static resource allocations obtained for a parallel execution environment with non-dedicated computational resources and varying availability, with a focus on studying the robustness of resource allocations.

# Mapping A and Mapping B of Applications ( $a_i$ ) to Machines ( $M_j$ )

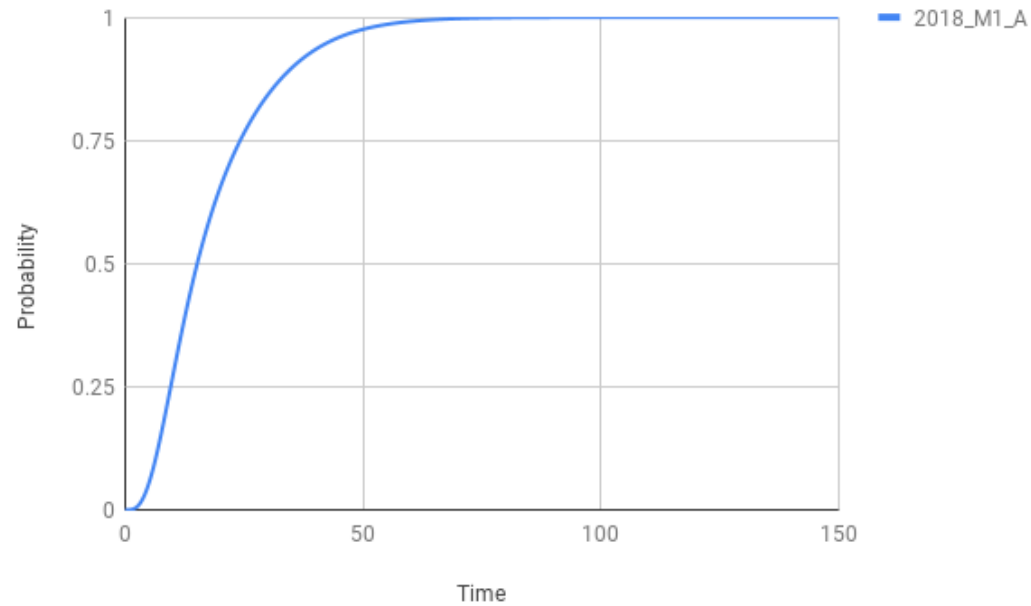
Machine	Mapping A	Mapping B
$M_1$	$a_5, a_9, a_{12}, a_{17}, a_{20}$	$a_3, a_4, a_5, a_{17}, a_{18}, a_{20}$
$M_2$	$a_6, a_{16}$	$a_2, a_{11}, a_{14}, a_{19}$
$M_3$	$a_1, a_3, a_7$	$a_1, a_7, a_{13}$
$M_4$	$a_2, a_4, a_{10}, a_{13}, a_{15}, a_{19}$	$a_9, a_{12}, a_{15}$
$M_5$	$a_8, a^{11}, a_{14}, a_{18}$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

Containerized replication of the activity diagram generated by the PEPA workbench for machine  $M_3$  component of the PEPA model for Mapping A from Srivastava 2018.

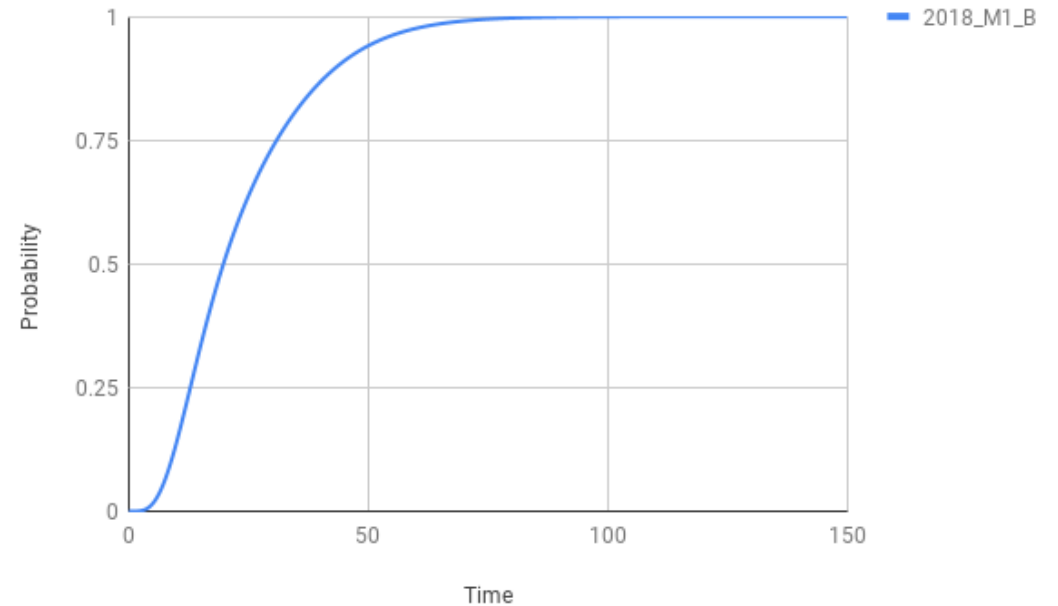


Results from the containerized replication of the cumulative distribution function (CDF) of the finishing time of Machine  $M_1$  for executing applications  $a_5, a_9, a_{12}, a_{17}, a_{20}$

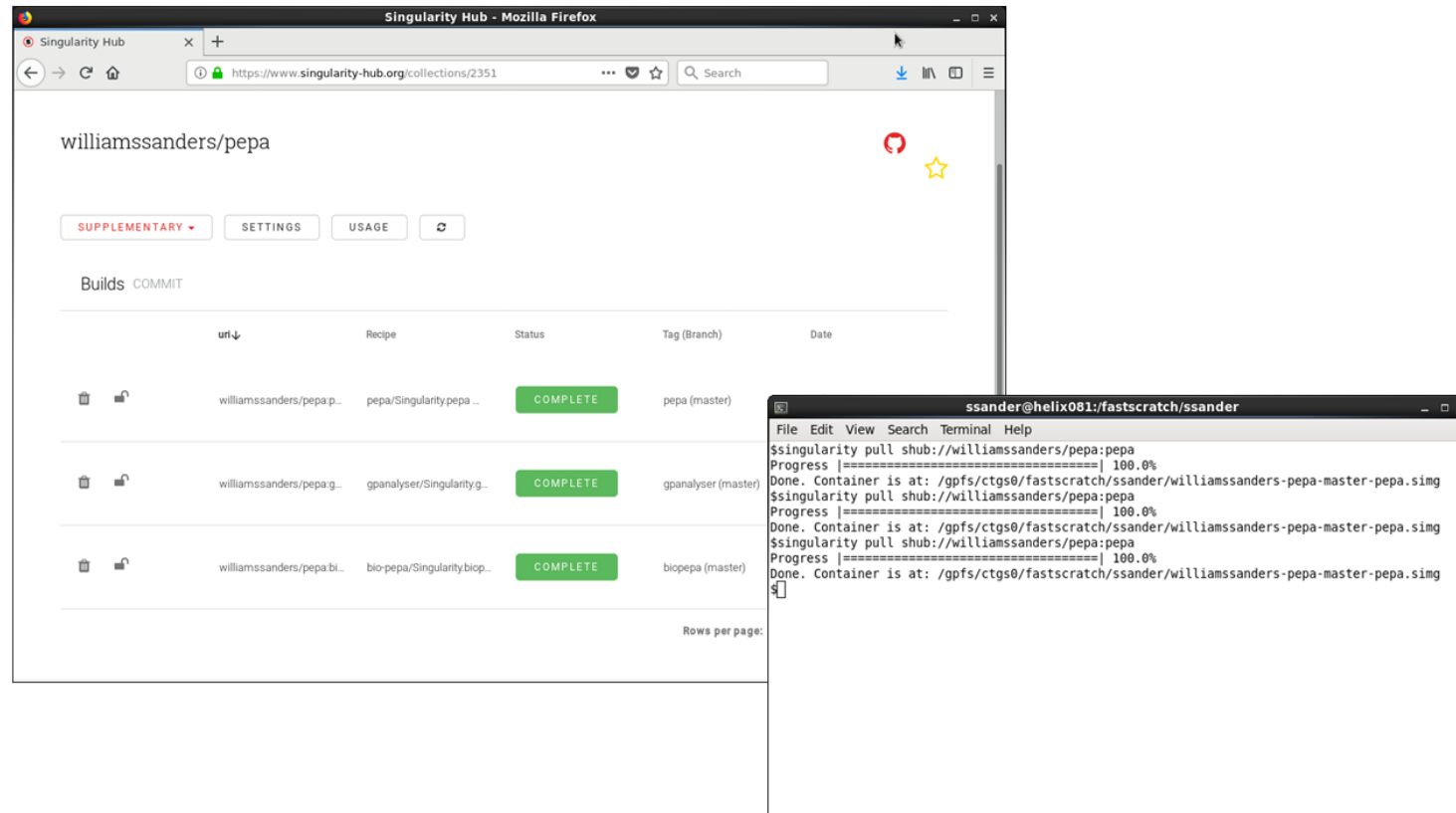
## Mapping A



## Mapping B



# Container Resources for PEPA, Bio-PEPA, and GPAnalyser



The image shows a screenshot of the Singularity Hub web interface in a Mozilla Firefox browser window. The URL is <https://www.singularity-hub.org/collections/2351>. The page displays the collection 'williamssanders/pepa' with tabs for 'SUPPLEMENTARY', 'SETTINGS', 'USAGE', and a refresh icon. Below the tabs, there is a table of builds. The table has columns for 'uri', 'Recipe', 'Status', 'Tag (Branch)', and 'Date'. Three builds are listed, all with a status of 'COMPLETE'.

uri	Recipe	Status	Tag (Branch)	Date
williamssanders/pepa.p...	pepa/Singularity.pepa...	COMPLETE	pepa (master)	
williamssanders/pepa.g...	gpanalyser/Singularity.g...	COMPLETE	gpanalyser (master)	
williamssanders/pepa.bi...	bio-pepa/Singularity.biop...	COMPLETE	biopepa (master)	

Below the table, there is a 'Rows per page:' label.

Overlaid on the bottom right of the browser window is a terminal window titled 'ssander@helix081:/fastscratch/ssander'. The terminal shows the following commands and output:

```
ssander@helix081:/fastscratch/ssander$ singularity pull shub://williamssanders/pepa:pepa
Progress |=====| 100.0%
Done. Container is at: /gpfs/ctgs0/fastscratch/ssander/williamssanders-pepa-master-pepa.sing
ssander@helix081:/fastscratch/ssander$ singularity pull shub://williamssanders/pepa:pepa
Progress |=====| 100.0%
Done. Container is at: /gpfs/ctgs0/fastscratch/ssander/williamssanders-pepa-master-pepa.sing
ssander@helix081:/fastscratch/ssander$ singularity pull shub://williamssanders/pepa:pepa
Progress |=====| 100.0%
Done. Container is at: /gpfs/ctgs0/fastscratch/ssander/williamssanders-pepa-master-pepa.sing
ssander@helix081:/fastscratch/ssander$
```

Build recipes and Singularity containers are publicly available for download from GitHub and Singularity-Hub.org.

<https://github.com/williamssanders/pepa> & <https://www.singularity-hub.org/collections/2351>

# Conclusions & Future Work

- Produced and verified versioned containers
  - The PEPA Eclipse Plug-in
  - Bio-PEPA
  - GPAnalyser (GPEPA)
- Provided Publicly Available and Reproducible Build Recipes and Singularity Containers
  - <https://github.com/williamssanders/pepa>
  - <https://www.singularity-hub.org/collections/2351>
- Shared container resource for the process calculi / process algebra communities
- Future Work
  - Identification and containerization of other PEPA derivatives and process calculi modeling tools to enhance this online resource
  - These containers will be used in future work to model resource allocations in parallel computing systems while providing reproducible artifacts to the scientific community



# Acknowledgements

- Mississippi State University:

- Ioana Banicescu
- Andy Perkins
- Donna Reese
- Daniel Peterson
- Eric Hansen



- The Jackson Laboratory:

- David McKenzie

- University of Southern Indiana:

- Srishti Srivastava



- Funding:

- JAX Tuition Reimbursement Program
- NSF #1034897



Questions?