SimMatrix: SLMulator for MAny-Task computing execution fabRLe at eXascales

**Abstract**

Executeing tasks within a system can experience significant challenges such as concurrency, resilience, heterogeneity, and load balancing. In this paper, we present SimMatrix, a distributed simulator that addresses these challenges. SimMatrix is designed to be a lightweight and scalable simulator, capable of running on a wide range of workloads. It allows for the investigation of various parameters important to understanding work stealing in distributed systems.

**Contributions**

- SimMatrix is a new light-weight and scalable discrete event simulator, which enables distributed scheduling for MTC workloads.
- SimMatrix has excellent flexibility and scalability, which makes it suitable for large-scale simulations.
- SimMatrix can be used in the study of heterogeneity, different programming models, and different scheduling strategies to study various network topologies.

**Validation**

SimMatrix is validated against SimGrid and GridSim, and results show that SimMatrix can achieve 85% efficiency running on real MTC workload traces obtained from 17 months of data from a petascale supercomputer.

**Work Stealing Algorithm**

**Algorithm 1** Dynamic Multi-Random Neighbor Selection for Work Stealing

```
DNV-MUL-SEL(num_nem, num_nodes)
1. select(n) be boolean array initialized all false except node itself
2. for i = 1 to num_nodes
3.  Idx = random() % num_nodes
4.   while selected[Idx] do
5.     index = random() % num_nodes
6.   selected[index] = true
7. next node
8. end
9. return next
```

**Algorithm 2** Adaptive Work Stealing Algorithm

```
ADA-WORK-STEALING(num_neigh, num_nodes)
1. if most load node < Neigh[0] load then
2.   Neigh[0] = most load_node
3. else
4.   steal_tasks from most_load_node
5.   return
6. end
7. if most load_node.load < 0 then
8.   sleep (poll_interval)
9. else
10. poll_interval = poll_interval * 2
11. end
12. if poll_interval > 1 then
13.   return
14. end
```

**References**


**Visualization**

Visualization for 1M nodes and MTC workloads for different number of nodes, the upper left has 2 static neighbors, the upper right has 2 static neighbors, the lower left has 1 static neighbor, and the lower right has 2 expected non-static neighbors.