2022 Future Computing Summer Internship Project: (Mapping Deadlock to the SST Model)

Nicholas Schantz*, AnotherFirst AnotherLast†

July 6, 2022

Abstract

Deadlock is a common issue in distributed systems in which multiple nodes in a cycle cannot proceed with any action because they are waiting for each other to take action as well. SST will be used to model the problem and parameterized so that the problem can be simulated and metrics can be collected which identify if deadlock exist. Simulating the problem and determining if a system-level deadlock has occurred is possible, but progress needs to be made regarding if there exist metrics for detecting if a component-level deadlock exist in a distributed system.

1 Project Description

The challenge addressed by this work is to map the distributed systems problem 'Deadlock' to the SST model. The problem is simulated to identify mathematical or logical conditions that cause the problem. This information is used to develop metrics to identify that the problem has occurred in generalized systems.

2 Motivation

Determining these metrics would allow for deadlock to be more easily identifiable in distributed systems then using algorithms to detect deadlock in the system. Furthermore, having metrics that identify the existence of the problem would help in creating better architecture to prevent the problem from occurring in distributed systems. Furthermore, the SST models written are resources that other users can use to learn and utilize SST's discrete event simulator.

3 Prior work

See https://doi.org/10.1145/357360.357365 for information on deadlock detection but unrelated to metric gathering. See https://doi.org/10.1145/3005745.3005760 for similar work in this area where metrics for causing deadlock were collected for a system utilizing Remote Direct Memory Access and Priority-based flow control.

4 How to do the thing

The software developed to respond to this challenge was run on one laptop. The software is available on (https://github.com/lpsmodsim/2022HPCSummer-Deadlock)

5 Progress

The model created for the deadlock problem was a network of nodes in a ring topology passing messages along the ring in one direction. The model was built with the conditions for deadlock in mind. Each node has exclusive access to the next node's queue, is making requests to send messages to the next node's finite queue, cannot force the next node to send messages out and free up its queue, and the node's form a cyclic

connection.

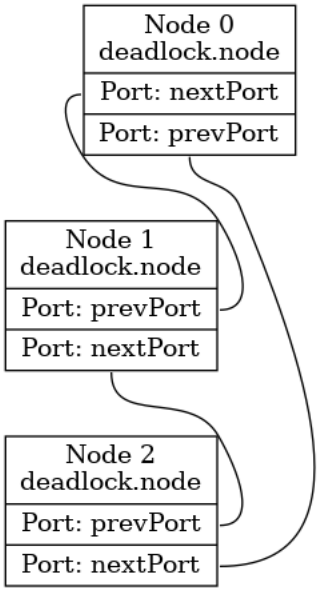

Figure 1: Topology of Nodes in the SST Model.

An issue with this model is that the conditions for deadlock occurrence are baked into the model so that deadlock will occur. A better model would be to allow the conditions to be variable to get a more generalized system to collect metrics as to when deadlock exist in a distributed system. Furthermore, this model is one specific example of deadlock in which I can collect information on mathematical conditions for why deadlock is occurring in this model, but does not determine for all cases if deadlock has occurred. For example, in a communication model of deadlock where messages are passed along, node's (what my model is based on) send rates can cause deadlocks to occur if all node's queues are filled up before messages reach their destination. Due to preemption not being built into the model, this causes a deadlock.

6 Result

The result is that for system-level deadlock, the existence of a deadlock can be detected by measuring if nodes are executing or if they are idle, measuring the number of requests for resources they are making. A global node can collect these metrics and when a user-defined metric is made then a system-level deadlock is identified to exist. If a global node is not used, nodes can detect that they are in a blocked state and then they can use a deadlock detection algorithm such as edge chasing to detect if all nodes are in a deadlocked state.
Metrics to detect component-level deadlock have not be determined.

References

[1] ADD BIB FILE

[2] ADD BIB FILE