Artificial Deadlock Detection and Correction in Bounded Scheduling of Process Networks

by Basu Vaidyanathan

EE382C - Embedded Software Systems
Fall 1999
Goals

- Understand the bounded scheduling of process networks
- Develop an algorithm and implement to detect the artificial deadlock and to resolve it to continue the program
- Understand the existing basic PN framework implementation
- Modify the code to keep it modular and transparent to applications
Process Networks

- A networked set of Turing machines
- Models functional parallelism and simulation possible on SMP hardware
- Well-suited for signal processing systems that deal with infinite streams of data
- Termination and Boundedness are undecidable.
Process Networks

- Kahn process networks model:
  - has finite set of processes and FIFO queues
  - execution of a process suspended on read from an empty queue
  - a process cannot wait for data from one queue or another
  - a process may not test for presence or absence of data
  - Systems that follow Kahn’s model are determinate
Process Networks

- Karp and Miller Computation Graph:
  - requires a threshold number of tokens on the arc before the consumer can fire
- Number of tokens produced/consumed is known only at runtime
- Dynamic scheduling is needed. It requires:
  - 1. Non-terminating programs must execute forever
  - 2. If possible, tokens accumulation on any of the FIFO queues must be bounded
Process Networks

- Parks Scheduling policy has three rules:
  - 1. Process suspended when reading an empty queue
  - 2. Process suspended when writing to a full queue
  - 3. On artificial deadlock, increase the smallest full queue size until a producer can fire.

- Realizes program execution forever with bounded memory whenever possible.
Process Networks

- **Artificial Deadlock**
  - Occurs when at least one process is suspended on write to a full queue

- **True Deadlock**
  - If all the processes are suspended on read then the program has terminated
Basic Process Networks Framework

- Implementation details:
  - Developed by Greg Allen of ARL at UT
  - Implemented in C++, combined with POSIX Pthread library for portability
  - Threshold and PNThreshold queue layers
  - Each node as a pthread
  - FIFO queues have input and output firing thresholds
  - Threshold amount of queue data mirrored to provide address/data continuity
Basic Process Networks Framework

- Node computation time greater than thread context switch time
- POSIX condition variable used to awaken consumer once data is available and to awaken producer once space is available
- Applied in Sonar Beamforming, a real-time problem where deadlock detection is not needed
- Provides a programming model for applications
My Design and Implementation

- Details:
  - Variable queue size for each FIFO queue
  - Maintain a list of qEntry class sorted by queue size. qEntry has Queue id, iswriteblocked stored in shared memory
  - Last thread in the network before suspending itself awakens all threads suspended on write
  - Only the thread with smallest queue size expands its queue size and continues and rest of the awakened threads suspended again.
  - Never gets into artificial deadlock situation
  - deadlock detection handled in PN queue layer
Issues and Improvements

- When expanding the queue reallocation of queue buffer is not possible
- Our PN implementation must not introduce additional deadlock violating locking hierarchy
- Use of a dedicated thread to handle deadlock
- Last thread can avoid awakening all threads suspended on write
- Searching qEntry list can be improved
Any Questions?