Efficient barrier synchronization for 2D meshed NoC-based many-core processors

Lovic Gauthier  Farhad Mehdipour  Shinya Ueno  Koji Inoue  Hiroshi Sasaki

Kyushu University
{lovic,ueno,sasaki}@soc.ait.kyushu-u.ac.jp, {inoue}@ait.kyushu-u.ac.jp, farhad@ejust.kyushu-u.ac.jp

Context
NoC-based many-core processor
- High scalability
- High communication throughput
- But long and variable communication delays

Objective
Barrier synchronizations
- Often used in recent multi-thread applications
- Usual algorithms do not take into account NoC

Target architecture
2D-meshed NoC many-core processor
- Each core has its own scratch-pad memory
- Inter-core communication: memory-mapped

About barrier protocols
Goal: each thread must signal each other thread when at barrier point
- Protocol: set of steps
- Begin of a step: threads send messages
- End of a step: threads receive messages

State of the art protocols:
All to all
Master / Slave
Step 0
Step 0
Step 1
Step 1
Step 2
Step 2
Step 3
Step 3
Step 4
Step 4

New algorithms
For all shapes and N
Rectangle tree: minimal number of NoC hops in sequence
Divide and conquer: minimal number of messages in sequence
Combined: very good in all cases

About barrier protocols
Butterfly tree: minimal number of NoC hops in sequence

Butterfly: very good if N is power of 2
Dissemination: very good is right shape

Improved protocols
Improvements: numbering and mirroring

Mechanism for synchronizing a set of threads
- Point in each thread's code
- Can be passed through: when all the threads reached it

Barrier realized by set of messages among threads
- Generic implementation →
- Remains the commands: "protocol", our focus

Results
SW Barrier

HW Barrier