

An Implementable Parallel Scheduler for IQ-Switches

Devavrat Shah
Stanford University

jointly

P. Giaccone and B. Prabhakar

Hot Interconnects IX, Stanford, 22nd August '01

Outline

- ◆ Scheduling problem
- ◆ Motivation
- ◆ APSARA: algorithm
- ◆ Performance
 - Throughput
 - Delay
- ◆ Conclusion

22nd August '01

Hot-IX

Scheduling

- ◆ Packet-based Input-Queued(IQ) switch with Virtual Output Queues(VOQ) at all inputs
- ◆ Cross-bar constraint requires that schedule $S = [s_{ij}]$ be such that:

$$s_{ij} \in \{0,1\}$$
$$\sum_i s_{ij} = 1 \quad \text{and} \quad \sum_j s_{ij} = 1$$

where, $s_{ij} = 1$ implies a packet will be transferred from input i to output j

22nd August '01

HotI-IX

Maximum weight matching

- ◆ Maximum weight matching is well-known good scheduling algorithm
 - delivers upto 100% of throughput for all admissible traffic
 - provides very good delay properties
- ◆ But
 - it is complex – takes $O(N^3)$ iterations in the worst case, where N is number of ports
 - not pipelineable

22nd August '01

HotI-IX

Goal

- ◆ Find a simple implementable scheduling algorithm with properties:
 - delivers up to 100% of throughput
 - approximates MWM well in terms of delay

22nd August '01

Hot-IX

Observation

- ◆ Let X be set of all possible $N!$ schedules, q_{ij} denote queue-length of VOQ(i,j)
- ◆ MWM finds schedule: $S^* = \arg \max_{S \in X} \{ \sum_{i,j} s_{ij} q_{ij} \}$
- ◆ Consider: $\tilde{S} = \arg \max_{S \in \tilde{X}} \{ \sum_{i,j} s_{ij} q_{ij} \}$
 - with $|\tilde{X}| \ll |X|$, it approximates MWM
 - if \tilde{X} represents ``good'' subset, then approximate algorithm should perform well

22nd August '01

Hot-IX

Observation

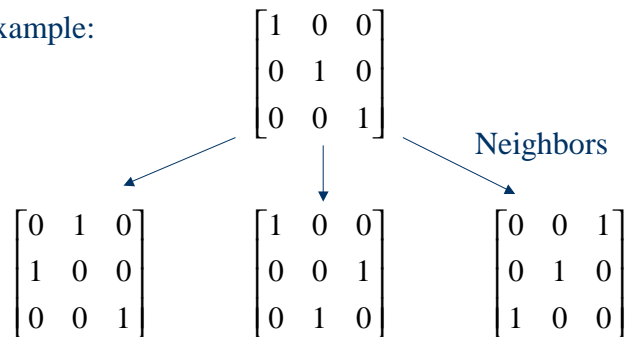
- ◆ Search in a smaller set is
 - feasible in parallel
 - quicker if easy to determine the set
- ◆ Feasible schedules are vertices of a convex set
 - Heavy schedule has heavy *neighbors*
- ◆ Memory helps

22nd August '01

Hot-I-X

Neighbors

Example:



22nd August '01

Hot-I-X

Hamiltonian Walk

- ◆ Let G be graph with all $N!$ schedules as nodes having all possible edges among these nodes
- ◆ A hamiltonian walk $H(t)$ on graph G
 - visits all schedules
 - without visiting any of them more than once

22nd August '01

Hot-IX

APSARA: Algorithm

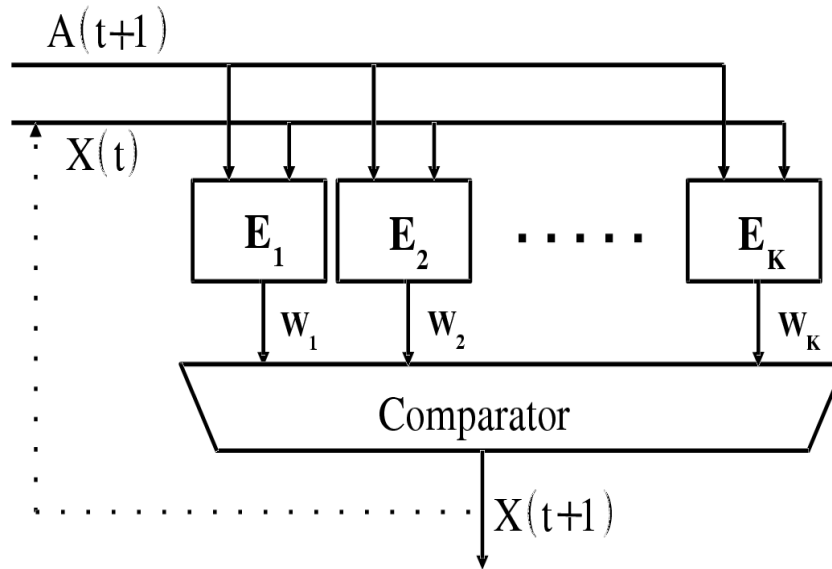
- ◆ $S(t)$ be schedule used at time t , $Q = [q_{ij}]$ be queue-length matrix at time $t+1$
- ◆ Let $N(S(t))$ be set of neighbors of $S(t)$
- ◆ Let $Z(t) = H(t \bmod N!)$
- ◆ Let $\tilde{X}(t+1) = N(S(t)) \cup Z(t) \cup S(t)$, then schedule for time $t+1$ be,

$$S(t+1) = \arg \max_{Y \in \tilde{X}(t+1)} \left\{ \sum_{i,j} y_{ij} q_{ij} \right\}$$

22nd August '01

Hot-IX

Possible Hardware



22nd August '01

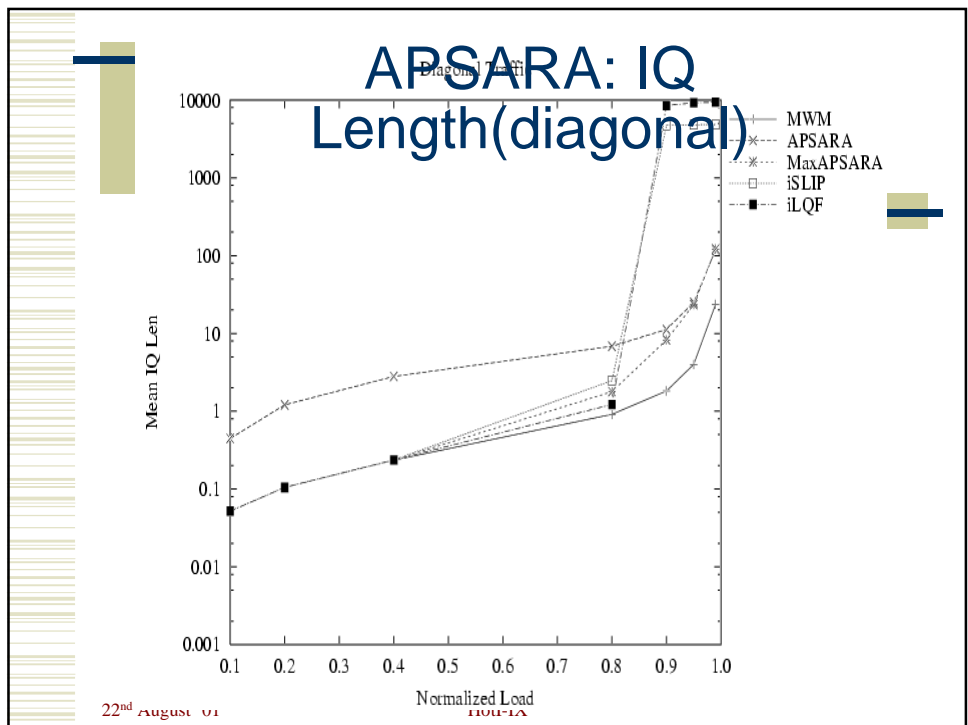
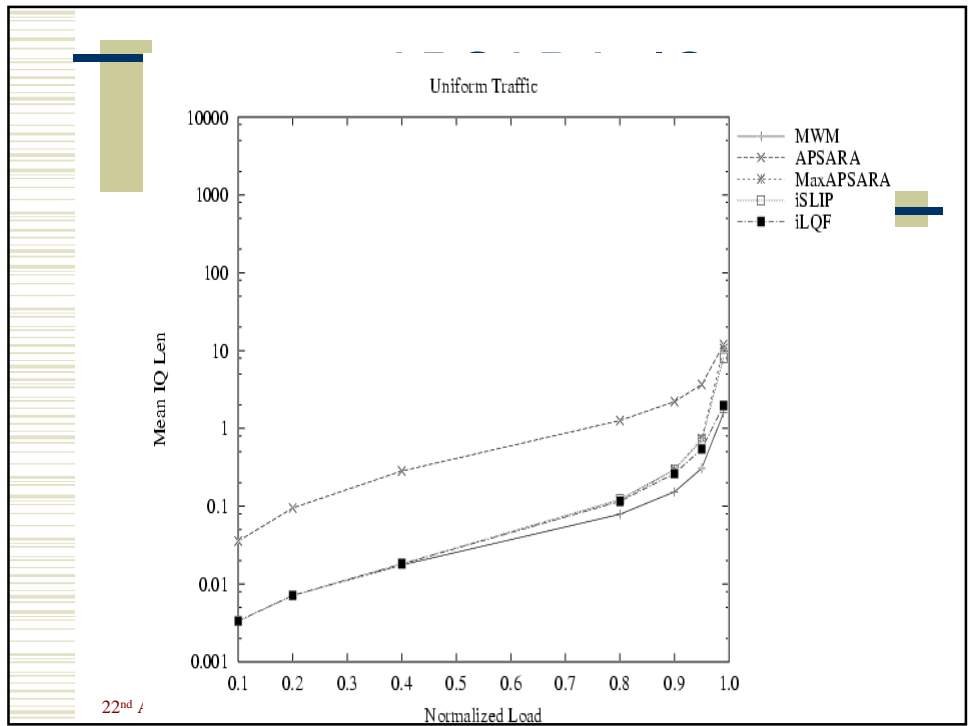
Hot-IX

APSARA: Throughput

Theorem: APSARA is *stable* under all admissible Bernoulli *i.i.d.* arrival traffic.

22nd August '01

Hot-IX

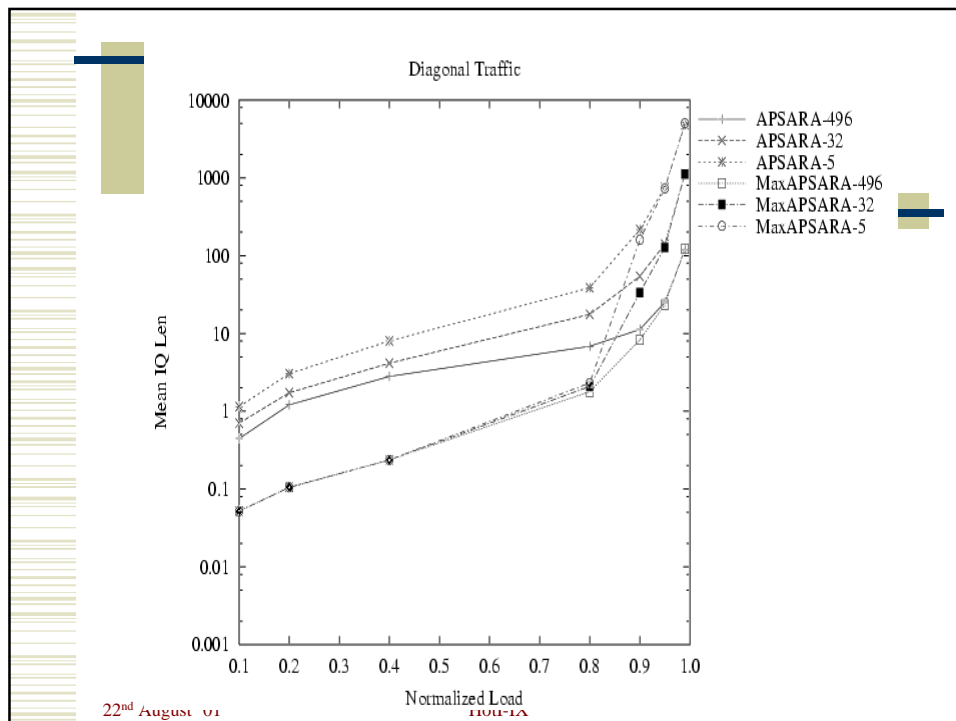


Limited Parallelism

- ◆ Basic algorithm requires $\binom{N}{2}$ modules to do search in parallel
- ◆ If space is limited to $K < \binom{N}{2}$ modules, then either
 - search over $\binom{N}{2}/K$ time slots, or
 - search over randomly chosen K neighbors from all $\binom{N}{2}$ neighbors

22nd August '01

Hot-IX



Conclusion

- ◆ APSARA: a novel algorithm
 - exploits parallelism to find good schedule in a small set of schedules
 - approximate maximum weight matching algorithm.
- ◆ With properties:
 - Throughput: 100%
 - Delay: approximates MWM very well
- ◆ Possible to implement in hardware