Multi-Node Programming – Longest IP Prefix Matching: A Stream Application using Multiple Imagines in Different Configurations

Henry Fu, Harn Hua Ng, Yeow Cheng Ong
Stanford University

EE482C Project Presentation
Thursday, May 30, 2002

Outline

- Motivation
- Goals
- Application: IP routing
- Setup
- Test methods, data, metric
- Results
- Challenges
- Conclusions

Motivation

- Develop and evaluate methods to efficiently map stream programs over multiple stream processing nodes
- Examine ways to partition data and/or instructions across the nodes
- Develop methods to coordinate multiple nodes and to communicate data
- Evaluate methods for load balancing

Goals

- Multi-node programming using multiple Imagines
  - Provide more computing power and higher performance
    - Requires more memory bandwidth and higher communication overhead

➢ Investigate different configurations that give best performance with least overhead

Introduction

- IP packet routing commonly used and can be mapped as a stream application
  - Each packet is independent
    - Data Level Parallelism (DLP)
  - Multiple flows of packets in router can be mapped as different streams of data
    - Thread Level Parallelism (TLP)
  - Same instruction can be distributed to multiple ALUs to perform multiple operations in parallel
    - Instruction Level parallelism (ILP)

Overview

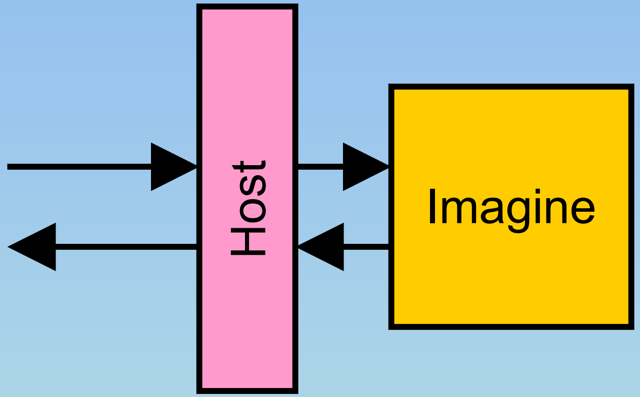
- IP Routing
  - Extract IP address information from each packet, compared against a routing table, and re-routed to appropriate nexthop address
  - IP Packet traffic modeled as data stream
  - After each lookup, each processor passes longest match result, along with current packet to a neighboring processor of another node to continue longest prefix matching

Algorithm used for IP address matching

- Within a Kernel:
  - Distribute routing table entries to all clusters
    - i.e. mask, destination address, nexthop
  - Find mask length for each routing table entries
  - Find match
    - (Packet address) AND (mask) XOR (destination address)
  - Keep track of length of longest prefix match, and corresponding next hop

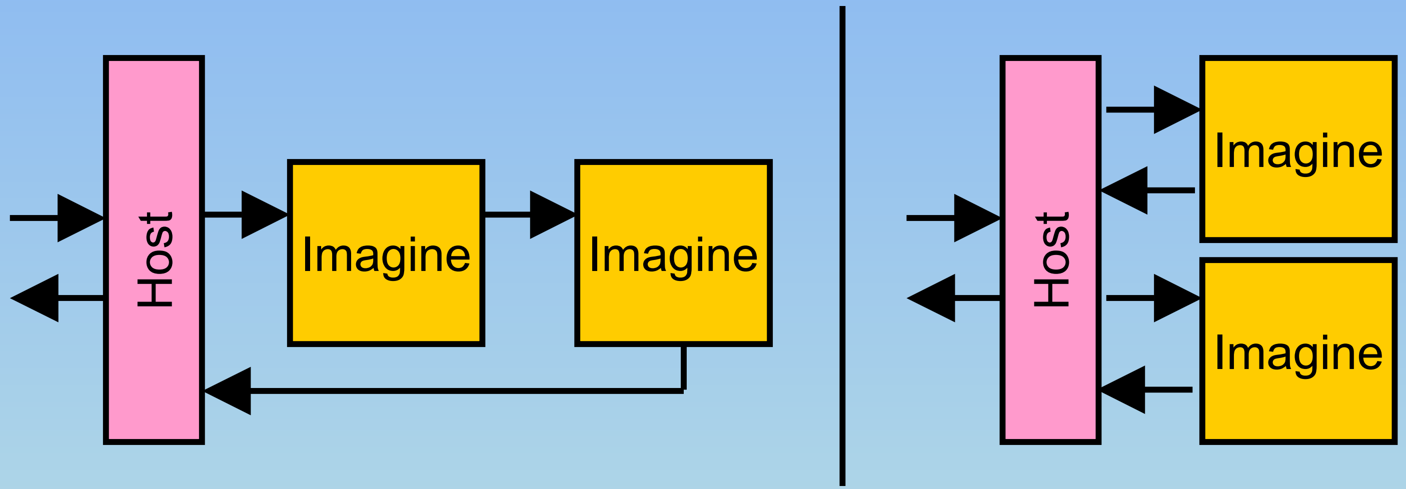
Setup

- Baseline case
  - Use 1 host processor and 1 Imagine
  - 1 parallel data lane, 1 pipeline stage
  - All results normalized according to baseline case results



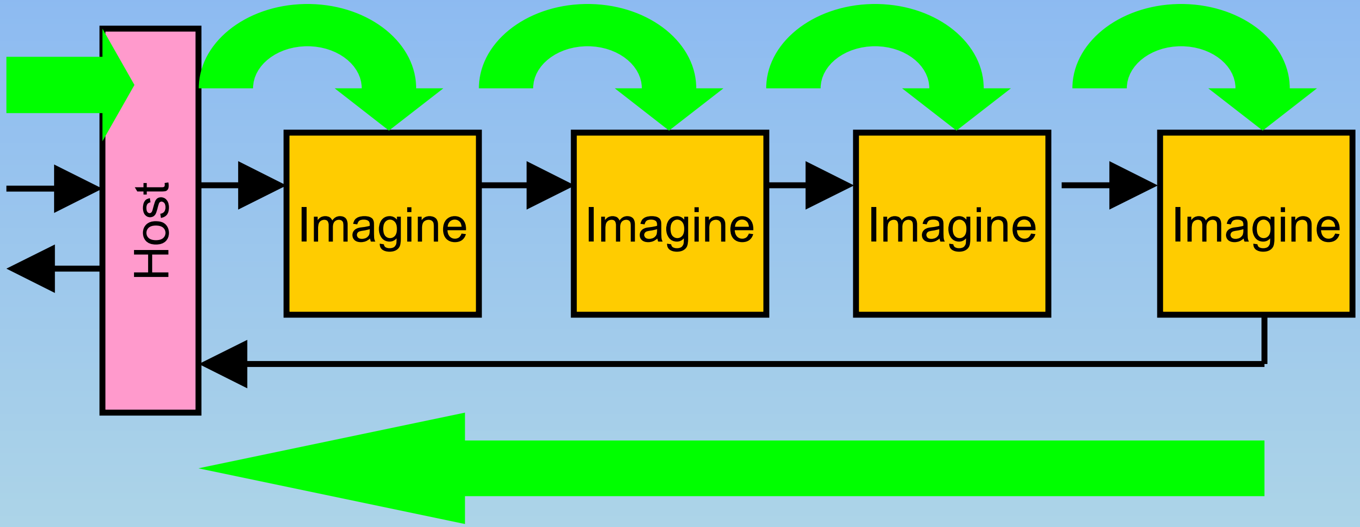
Setup (More)

- 2 Imagines
  - Use 1 host processor and 2 Imagines
  - 1 parallel data lane, 2 pipeline stages
  - 2 parallel data lanes, 1 pipeline stage



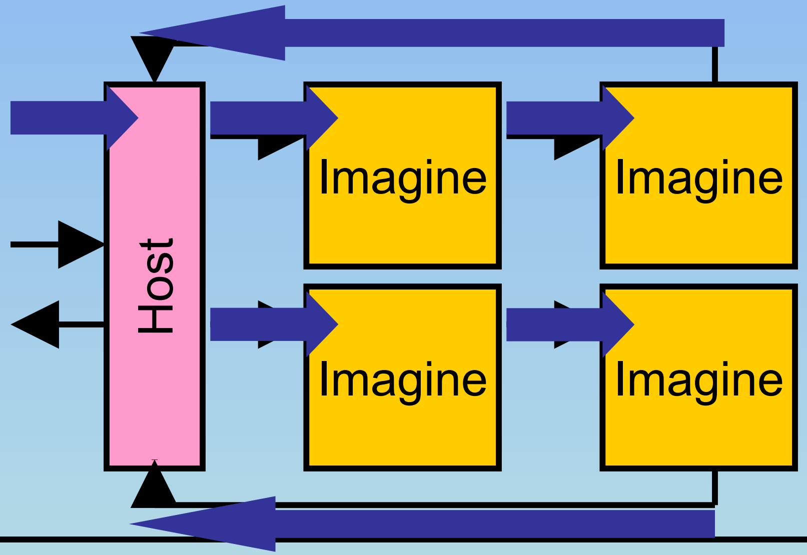
Setup (More)

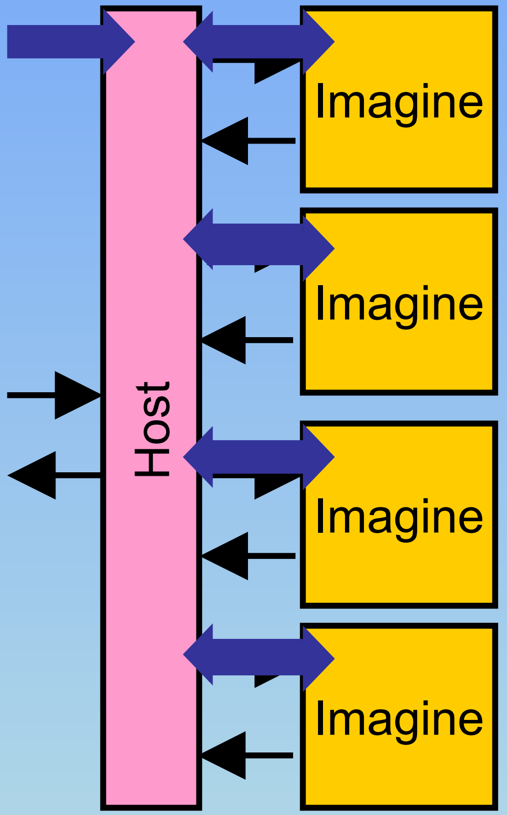
- 4 Imagines
  - Use 1 host processor and 4 Imagines
  - 1 parallel data lane, 4 pipeline stages



Setup (More)

- 4 Imagines
  - 2 parallel data lanes, 2 pipeline stages
  - 4 Parallel data lanes, 1 pipeline stage





Configurations

- Pipelined configuration: total # of routing table entries distributed evenly to all Imagine processors in each pipeline stage
  - Static load balancing
- Parallel configuration: total # of destination addresses distributed evenly to all data streams
  - Static load balancing

Test Methods

- Program written in StreamC and KernelC
- Profiling used to estimate cycle count in each kernel and total execution time

- Number of Imagines used: 1, 2, and 4
- Number of test packets used: 8, 32, 1024
- Number of routing entries used: 8, 32, 1024

Test Data

- Randomly-generated destination addresses
- Routing table entries captured from major router in ISP
  - ner-routes.bbnplanet.net
  - 119, 967 entries captured
  - Subset of total entries randomly picked for experiment
  - C program to generate correct results and to verify output of stream program

Test Metric

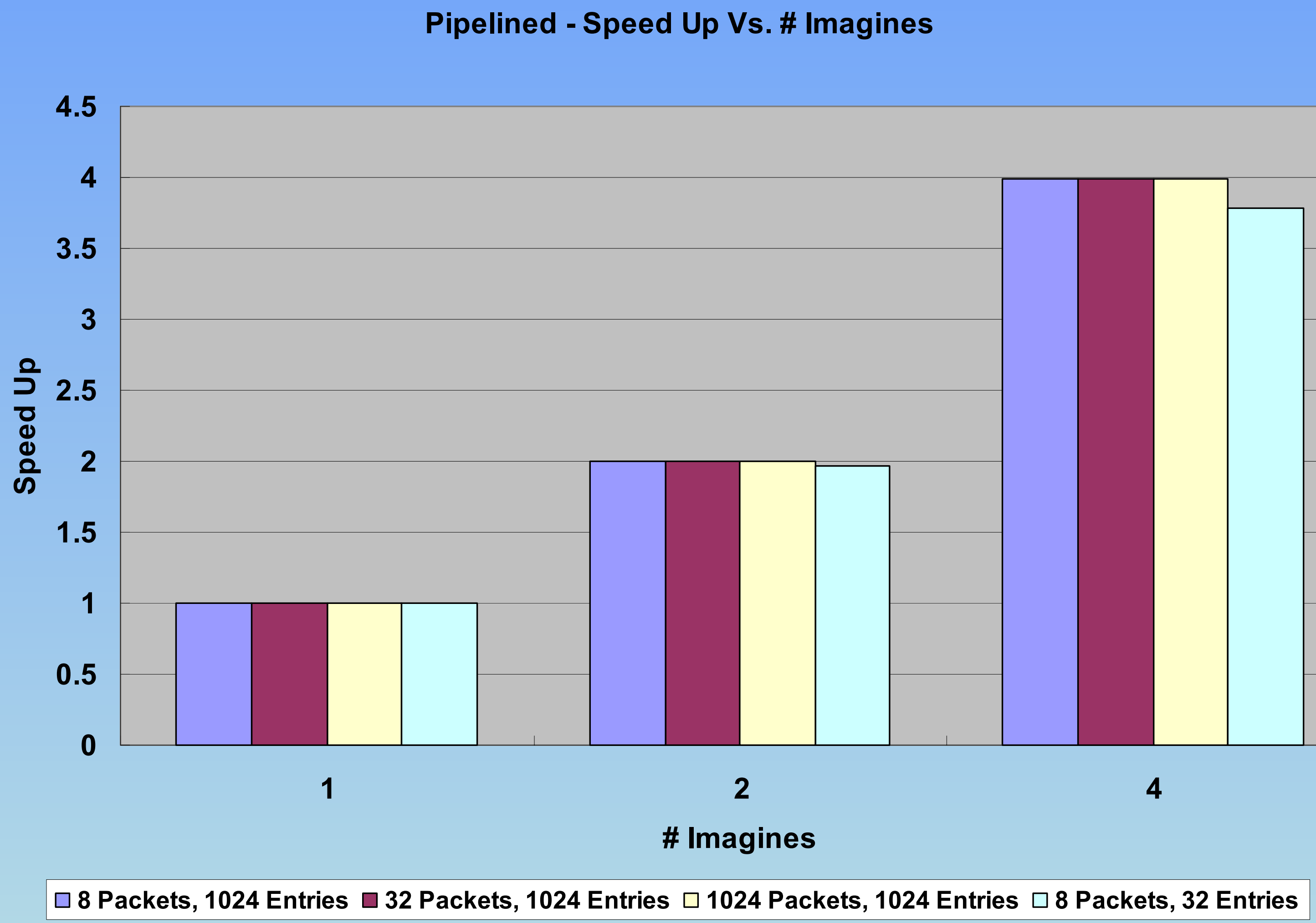
- Execution time of single Stream Processor configuration vs. that of multi-node configuration
  - 1, 2, 4 Imagines arranged in pipelined configuration vs. 1 Imagine configuration
  - 1, 2, 4 Imagines arranged in parallel configuration vs. 1 Imagine configuration
- Communication overhead examined in > 1 Imagine configuration

Test Results

- Pipelined Configuration
  - Almost ideal speed up for large data set
  - Significant overhead for small data set

Pipelined								
			Execution Time					
# Packets	# Entries	# Imagines	Imagine 0	Imagine 1	Imagine 2	Imagine 3	Avg/Img	Speed Up
8	1024	1	52325				52325	1
		2	25636	26721			26178.5	1.99877762
		4	12260	13408	13408	13345	13105.25	3.99267469
32	1024	1	209300				209300	1
		2	102544	106884			104714	1.99877762
		4	49040	53632	53632	53380	52421	3.99267469
1024	1024	1	6697600				6697600	1
		2	3281408	3420288			3350848	1.99877762
		4	1569280	1716224	1716224	1708160	1677472	3.99267469
8	32	1	1669				1669	1
		2	833	868			850.5	1.96237507
		4	415	450	450	450	441.25	3.78243626

Test Results (More)

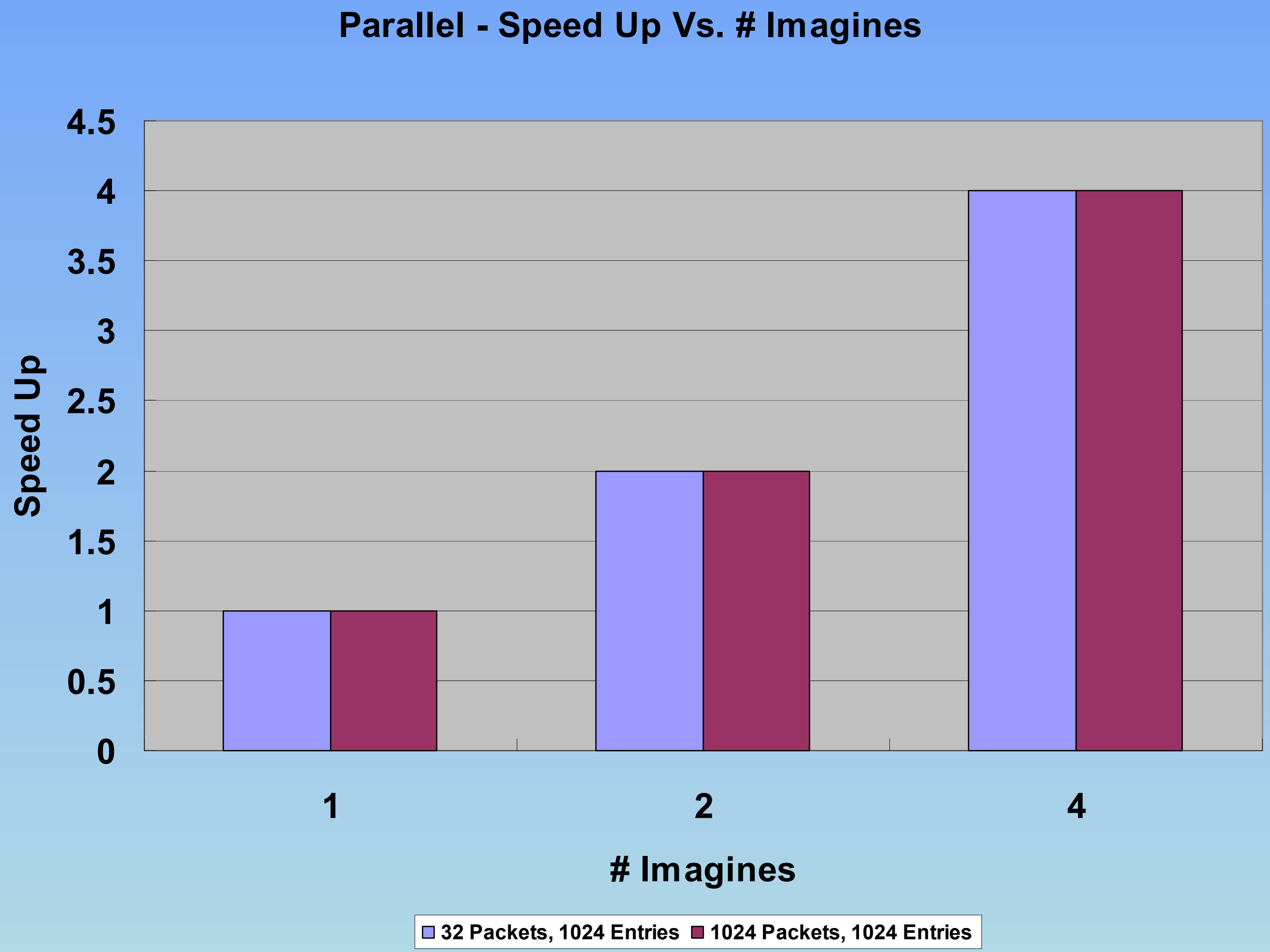


Test Results (More)

- Parallel Configuration
  - Almost ideal speed up for large data set
  - Slight overhead for large data set

Parallel								
			Execution Time					
# Packets	# Entries	# Imagines	Imagine 0	Imagine 1	Imagine 2	Imagine 3	Avg/Img	Speed Up
32	1024	1	209311				209311	1
		2	104650	104661			104655.5	2
		4	52325	52325	52325	52336	52327.75	4
1024	1024	1	6697701				6697701	1
		2	3348800	3348901			3348850.5	2
		4	1674400	1674400	1674400	1674501	1674425.25	4

Test Results (More)



Challenges

- Limitation on # of imagines (max. 4) when 1 host used
- Complexity in multiple hosts simulation
  - Out of order execution
- Profiling has restrictions
- Problems with communication and synchronization among multiple imagines

Conclusions

- Speedup increases with number of processing nodes
  - Communication and synchronization overheads
- Better to distribute data and instructions across multiple nodes to increase parallelism
- Additional configurations to be tested

Questions & Comments