Multi-Core Computer Architecture: Storage and Interconnects Week 8

Aronya Baksy

November 2021

1 Improving Quality of Service in TCMP

- L1 cache misses are handled by generating a packet to the L2 cache bank (in the appropriate tile). If the L2 misses, then L2 has to generate a packet to memory controller which replies to L2.
- **Light** applications on NoC generate less traffic in NoC (i.e. less cache misses). **Heavy** applications generate larger volume of network traffic on NoC

2 Congestion Management in NoC

- NoC is a critical resource shared by multiple applications
- Conventional switch-level scheduling policies (output port allocation for flits) are round robin and age-based priority
- Each packet has a different criticality. Packet is said to be critical if latency affects the performance of the application
- Different cache misses create packets in NoC. These packets may be towards nearby or far away routers. Cache miss Latency depends on this distance
- Packet latency = network stall time. If misses do not overlap, then all misses will stall the application
- Having OOO execution and non blocking caches allows memory requests to be sent in parallel. This, in turn, allows memory stalls to overlap and get hidden.
- This phenomenon, where packet latency and memory stall time are not same, is a result of **memory level parallelism** (MLP). Different packets have different criticality in this case

2.1 Slack Aware NoC Routing

- Slack is the number of cycles a packet can be delayed in a router without affecting application performance
- Source of slack is Memory Level Parallelism that allows a cache miss to be hidden by overlapping with other pending cache requests
- Prioritize packets with lower slack (i.e. those that cannot be delayed too many cycles)

2.2 Source Throttling

- Delaying injection of certain packets into the network in order to reduce num. of packets in network, hence reducing load
- This increases *average* network throughput and hence increases performance of TCMP
- Throttle the heavy applications and not the light ones. Throttling light apps reduces system performance because network load is not being reduced too much.
- No single throttling rate that works universally. Adjust throttling rate to avoid underutilization and congestion as well

- How to measure network intensity of applications? Use L1 cache misses per 1000 instructions (L1 MPKI) as a metric for each application and throttle the network intensive applications
- This calculation of L1 MPKI is expensive if done each cycle. Recompute these metrics at some interval (research indicates that 100k cycles is a good interval)
- During the epoch of 100k cycles, each node computes L1 MPKI and stores
- At the start of the next epoch, the measurements are sent to a **central controller** that classifies the apps as light/heavy, then adjusts throttling rate and sends this info to all the nodes

2.3 Application-Core Mapping Core Policies

- Traditional scheduling of tasks on cores is temporal (round robin, fcfs, sjf etc.)
- In TCMP, there is also the spatial aspect of which core to schedule task on based on network latencies and memory organization
- The 4 techniques listed below improve locality, reduce interference and improve resource utilization

2.3.1 Challenges

- How to minimize communication distance
- How to reduce destructive interference between apps
- How to prioritize apps to improve throughput

2.3.2 Clustering

- Divide network into clusters. Assign apps such that all the cache references it makes are in or near a single cluster only
- Using locality aware page replacement policy
- When allocating free page, give priority to pages belonging to the cluster's memory controller

2.3.3 Balancing and Isolation

- Clustering generally causes heavy apps to cluster in one region together causing uneven loads
- **Balancing** involves having right mix of light and heavy apps across clusters in the NoC
- **Isolation**: isolate sensitive applications (those that generate few misses, but need very fast service) to one cluster on the NoC, and between the remaining ones distribute the light/medium/heavy applications *fairly*
- How to estimate sensitivity:
  - High MPKI
  - High stall cycles per miss

2.3.4 Radial Mapping

- Also called intra-cluster mapping (Clustering, Balancing, Isolation are *inter-cluster* techniques)
- Heavy applications benefit from being close to the memory controllers (at the corners of the mesh)
- Map medium and light apps to the center of the cores

3 Emergent Trends in TCMPs - Scaling of Mesh Architectures

- **Multi-drop links**: special links where signals reach multiple routers in a single cycle
- These links are broken down and built by sending bypass signals to the
- **3D-NoC**: stack multiple tiles, use vertical and horizontal links.
- Benefits of 3D structure: high packing density, noise immunity, superior performance
- Vertical links can use near-field coupling schemes (capacitive uses less area, but inductive has longer transmission range) instead of physical links
- **RF-Interconnects**: instead of current & voltage, use electromagnetic waves over micro-strip lines within the metal layers of the chip
- Modulated EM signals (several GHz freq) are guided through the lines. This requires demodulation at the receiver end as well. Signals are very fast (speed of light)
- **Nanophotonics**: Microring resonators divert light of certain freq when V is applied, and let all light pass through otherwise

3.1 Wireless NoC

- In a CMesh (concentrated mesh), there is an $N \times N$ grid of nodes with each node itself consisting of M cores (e.g. 16×16 Cmesh with 4 cores per node is a 1024 core processor)
- The $N \times N$ Cmesh is divided into N **WCube** units. Each Wcube is labelled with a binary value
- Each Wcube can communicate wirelessly with a Wcube who's Hamming distance is 1 (edit distance)
- Address of a single node is formatted as "WCube number. node number. core number"
- Nodes arrive at a Wcube at the central router (triangle). From there they are routed to the appropriate node and core.
- In a Wcube structure every cache miss request or reply packet reaches the destination in 1-8 hops

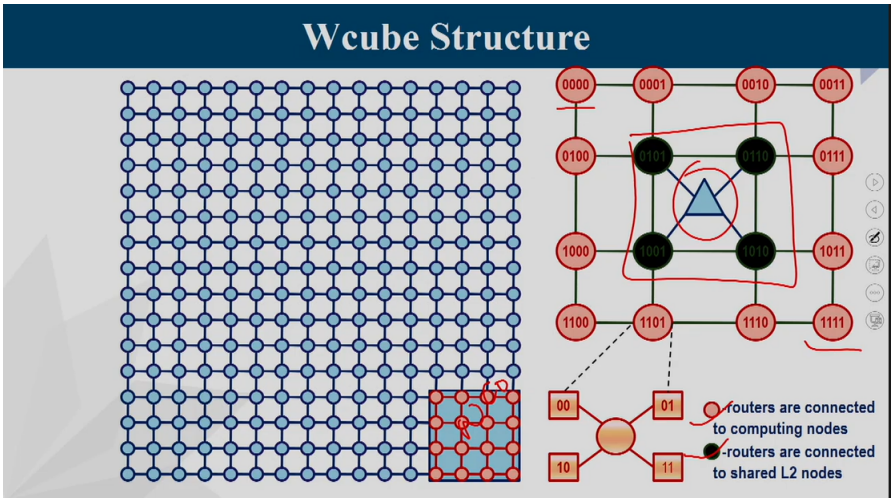

Figure 1: WCube Structure

3.2 Small-World Routing

- Split the mesh into smaller square grids. Each grid has a single wireless access point (WAP) (marked in orange in fig 2)
- A hybrid form of routing that allows both traditional XY as well as wireless routes or a mix of both

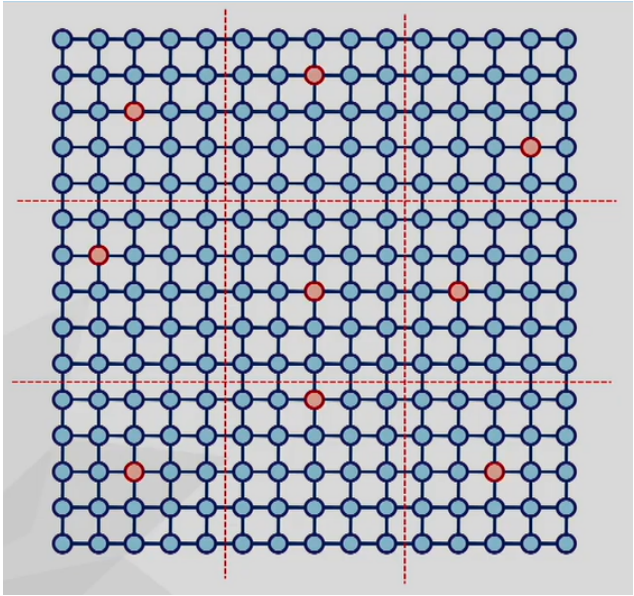

Figure 2: Small World Routing

- Source sends packet to the nearest WAP (XY route), which forwards it wirelessly to the destination WAP who forwards it to the destination (again XY route).
- Adaptive routing makes use of either the wireless or traditional XY routing depending on the distance between source and dest.
- If lots of traffic uses the wireless points then those become *isolated points of congestion.* This leads to very early saturation of network

3.3 Multicasting in Wireless NoC

- One source, multiple destinations
- Wireless infra helps this. Each router can duplicate and send packets to nearby destinations multiple times
- Used in cache coherence. One node validates/invalidates the cache blocks present in multiple other nodes at once.
- Multicasting is performed using specialized packet headers

3.4 Broadcasting in Wireless NoC

- 1 source, destination is all the nodes of the NoC
- Broadcasting is way faster in wireless NoC than in a traditional XY routing mesh NoC
- Broadcasting is needed for special commands to all routers, during boot time startup and reset procedures