Many Cores

Introduction

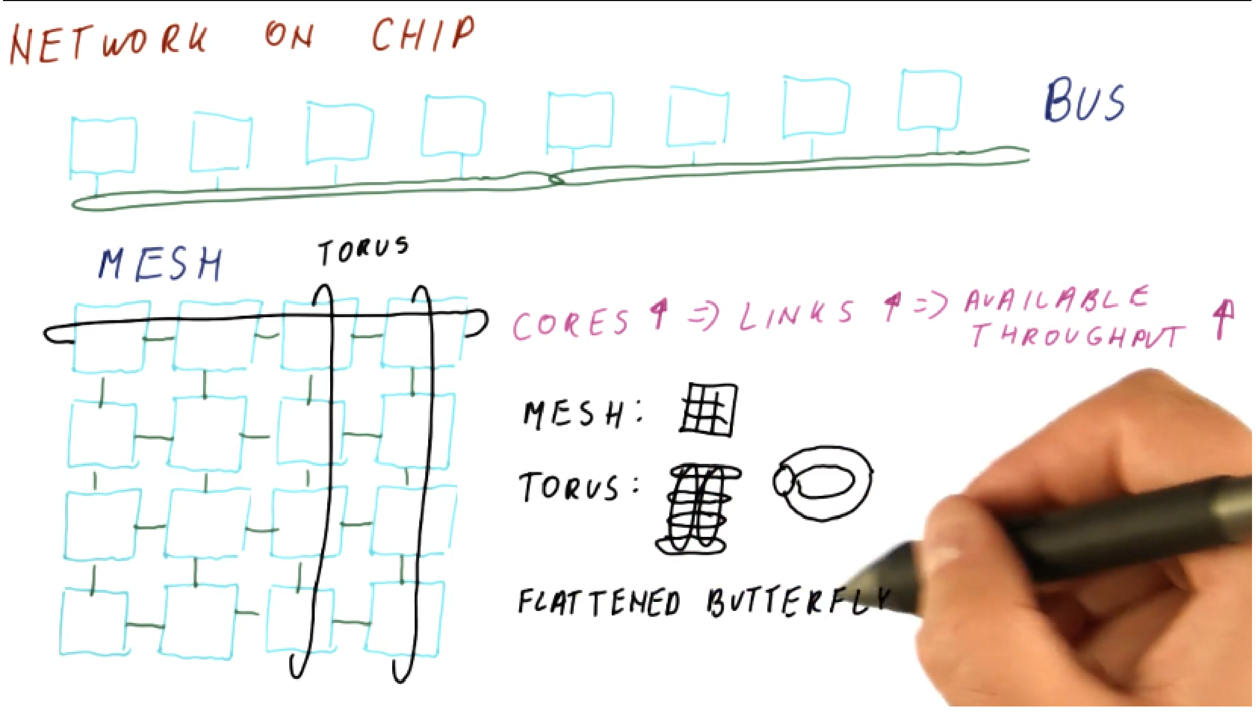
1. Apply lessons from the course to examine what happens when there are many cores on one chip

Many Core Challenges

1. As number of cores increases, coherence traffic also increases
    - Writes to shared locations result in invalidations and misses, both of which go on the shared bus
2. As number of cores increases, more writes occur per second, requiring more bus throughput until it eventually exceeds what is posssible
    - Bus: One request at a time (necessary, but becomes bottleneck)
    - Need a scalable on-chip network and directory coherence so we don't rely on the bus

Network on Chip

1. Bus will get slower as more cores are added (further distance to travel)
2. Instead, a mesh can connect all of the cores
    - A core can forward a message to other cores if two aren't connected
    - Pairs of nodes can talk totally independently of other pairs
    - As number of cores increases, number of links increases resulting in greater available throughput
    - Meshes are good for processors because the links don't cross
3. Torus: Connect end points of mesh to each other
4. Flattened butterfly
    - Advanced on-chip interconnects



Mesh Network

Mesh vs Bus Throughput Quiz

1. 4 cores, each core sends 0.25 of all messages, sent round-robin to other cores, assume randomly distributed
2. Each core sends 10M messages per second
3. Bus supports 20M messages per second, mesh supports 20M messages per second per link
4. What is the speedup achieved by switching from the bus to the mesh?
    - On bus, cores slow to half their speed (2x execution time)
    - For the mesh, the two closest nodes get 1/2 of the traffic while the links to the third core get 1/6 of the traffic
        - Apply this from every node, each node sends 4/3 the number of messages sent by a single core
        - 13.3M messages per second; this means the cores don't have to slow down, so the speedup is 2

Many Core Challenges 2

1. As the number of cores increases, so does the coherence traffic
    - Scalable on-chip network and directory coherence
2. As the number of cores increases, off-chip traffic also increases
    - More cores means more caches
    - Misses per core is the same, but as more cores increases, the number of memory requests also increases
    - Number of pins increases, but not proportional to the number of cores
        - This becomes a bottleneck
    - Need to reduce the number of memory requests per core
        - Make the last level cache (L3) shared among all cores
        - Size of LLC needs to go up proportionally to the number of cores
        - One big LLC would be slow and have one entry point -> bottleneck
3. Instead of one big LLC, we make a distirbuted LLC

Distributed LLC

1. Logically one cache (block not replicated)
    - If we have a 4 MB cache, we can use all 4 MB
    - Sliced up so each tile gets part of it
        - Each core has L1, L2, and part of L3 cache
        - L3 size = N * 1 MB (N = number of cores)
    - Size grows with number of cores, but no single entry point
2. If there's a miss in an L2 core, how do we know which slice might have the data in their L3 cache?
    - Round robin by cache index
        - This distributes accesses well, but may not be good for locality
    - L3 cache with 1024 sets
        - Slice 0: Set 0, Set 8, Set 16, . . .
        - Slice 1: Set 1, Set 9, Set 17, . . .
        - . . .
        - Slice 7: Set 7, Set 15, Set 23, . . .
    - Round robin by page number
        - All blocks in the same page end up in the same slice of the L3 cache
        - OS can map pages to make accesses more local

Distributed LLC Quiz

1. 16 cores organized as a 4x4 mesh
2. Tile: Core, L1, L2, slice of L3
3. L3 cache: 8 MB, 256B block, 16-way SA, round robin by set
4. Core 0: LW 0x12345678 which misses in L1 and L2

5. Which tile receives the request from core 0?
    - 8 least significant bits are block offset
    - Next bits are the index; least significant bits tell us what tile to look at
        - 16 tiles, so next 4 bits (1 hex character)
    - This means tile 6 should have this data

Many Core Challenges 3

1. As the number of cores increases, so does the coherence traffic
    - Scalable on-chip network and directory coherence
2. As the number of cores increases, the off-chip traffic increases
    - Large, distributed shared LLC
3. Coherence directory too large to fit on the chip
    - We require directory coherence
    - Typical directory has an entry for each memory block
    - Many GB of memory -> Billions of entries -> Can't fit on chip

On Chip Directory

1. Where is the home node the contains data about the block?
    - Same as LLC slice
2. LLC only contains entries we have in the cache
    - Directory needs data for every possible memory block that could be in that slice
        - How do we handle so many entries?
    - If an entry isn't in the LLC, it can't be shared by anyone, so we don't need to maintain that information
3. Partial directory
    - Directory has limited number of entries
    - Allocate entry only for blocks that have at least 1 presence bit at 1
    - Only need to track entries in private caches (not memory or LLC)
        - We still have a limited number of entries, so we'll eventually run out

On Chip Directory Quiz

1. How do we handle the case where we run out of directory entries?
    - Replace an existing entry (e.g. LRU)

On Chip Directory 2

1. When we run out of directory entries, we pick an entry (E) to replace
    - Need to clear the presence bits for entry E
        - Send an invalidation to all tiles with P == 1
        - Once all presence bits are 0, we can remove it from the cache
        - Put new entry in E
2. This is another type of miss
    - Not due to coherence, capacity, conflict, etc.
    - Due to limited capacity in directory

Many Core Challenges 4

1. As the number of cores increases, so does the coherence traffic
    - Scalable on-chip network and directory coherence
2. As the number of cores increases, the off-chip traffic increases
    - Large, distributed shared LLC
3. Coherence directory too large to fit on the chip

    - Distributed partial directory
4. Power budget split among cores
    - As the number of cores increases, the number of Watts per core decreases because frequency and voltage also decrease
    - A single-threaded program would only get 1/N of the total power despite none of the other cores requiring the rest of the power

Multi Core Power and Performance

1. One core can spend 100W
    - Power = C * V ˆ 2 * f
    - f = 3.8 GHz
2. Two cores can spend 50W per core
    - Power = 1/2 P1 (V ~ f)
    - Power = (1/2) ˆ (1/3) = 0.8 * f = 3.0 GHz

Performance vs Number of Cores Quiz

1. Chip power is 100 W
    - 5 GHz for 1 core
    - Execution time for a program is 100s for 1 core
2. What is the clock speed for two cores?
    - (1/2) ˆ (1/3) = 0.8 * f = 5 * 0.8 = 4
3. What is the clock speed for four cores?
    - (1/4) ˆ (1/3) = 0.8 * 0.8 * f = 5 * 0.64 = 3.2
4. What is the execution speed for two cores? Remember to take into account the change in frequency
    - 20 / 1 + 80 / 2 = 60 seconds
    - 60 * 5 / 4 = 75 seconds
5. What is the execution speed for four cores? Remember to take into account the change in frequency
    - 20 / 1 + 30 / 2 + 40 / 3 + 10 / 4 = 50.8 seconds
    - 50.8 * 5 / 3.2 = 79.375 seconds

Available Parallelism	% of 1-core time
1	20%
2	30%
3	40%
4	10%

No Parallelism to Boost Frequency

1. Modern processors boost the voltage and frequency for single-threaded performance to take advantage of the power offered to the chip
2. Example: Intel's core i7-4702MQ (mobile chip)
    - Design power: 37W
    - 4 cores, "normal" clock 2.2 GHz
    - "Turbo" clock 3.2 GHz (1.45x normal -> 3x power)
        - Can't actually get 4x power
3. Can't achieve 4x power due to heat constraints
    - If we spread power around, heat is also spread around
    - When only using one core, all of the heat is concentrated
        - 3x gives similar temperatures to running all cores
4. Example: Intel's core i7-4771 (desktop chip)
    - Design power: 84W

- 4 cores, “normal” clock 3.5 GHz
- “Turbo” clock 3.9 GHz (1.11x normal -> 1.38x power)
  - Because this is a desktop chip, it is cooled much better than a mobile
  - This means it’s already operating closer to its maximum temperature, so there is little room for “turbo” boosting

Many Core Challenges 5

1. As the number of cores increases, so does the coherence traffic
   - Scalable on-chip network and directory coherence
2. As the number of cores increases, the off-chip traffic increases
   - Large, distributed shared LLC
3. Coherence directory too large to fit on the chip
   - Distributed partial directory
4. Power budget split among cores
   - “Turbo” when using only one core
5. OS confusion - multithreading, cores, chips all at the same time

SMT - Cores - Chips

1. All combined:
   - Dual socket motherboard (two chips)
   - 4 cores on each chip
   - Each core 2-way SMT
     - 16 total threads
2. Simple OS would treat these as 16 independent cores
   - This naive approach presents issues when the number of threads used is less than the number of cores
     - Distributing threads across chips or cores results in better performance
3. Windows, Linux are smart enough to efficiently utilize the hardware

Many Core Challenges 6

1. As the number of cores increases, so does the coherence traffic
   - Scalable on-chip network and directory coherence
2. As the number of cores increases, the off-chip traffic increases
   - Large, distributed shared LLC
3. Coherence directory too large to fit on the chip
   - Distributed partial directory
4. Power budget split among cores
   - “Turbo” when using only one core
5. OS confusion - multithreading, cores, chips all at the same time
   - OS is aware of hardware and distributes work intelligently

Conclusion

1. Covered what future multicore processors will likely look like