UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

Multi-core CPU

Heterogeneous Computing

Professor: Dr. Joel Fuentes - jfuentes@ubiobio.cl

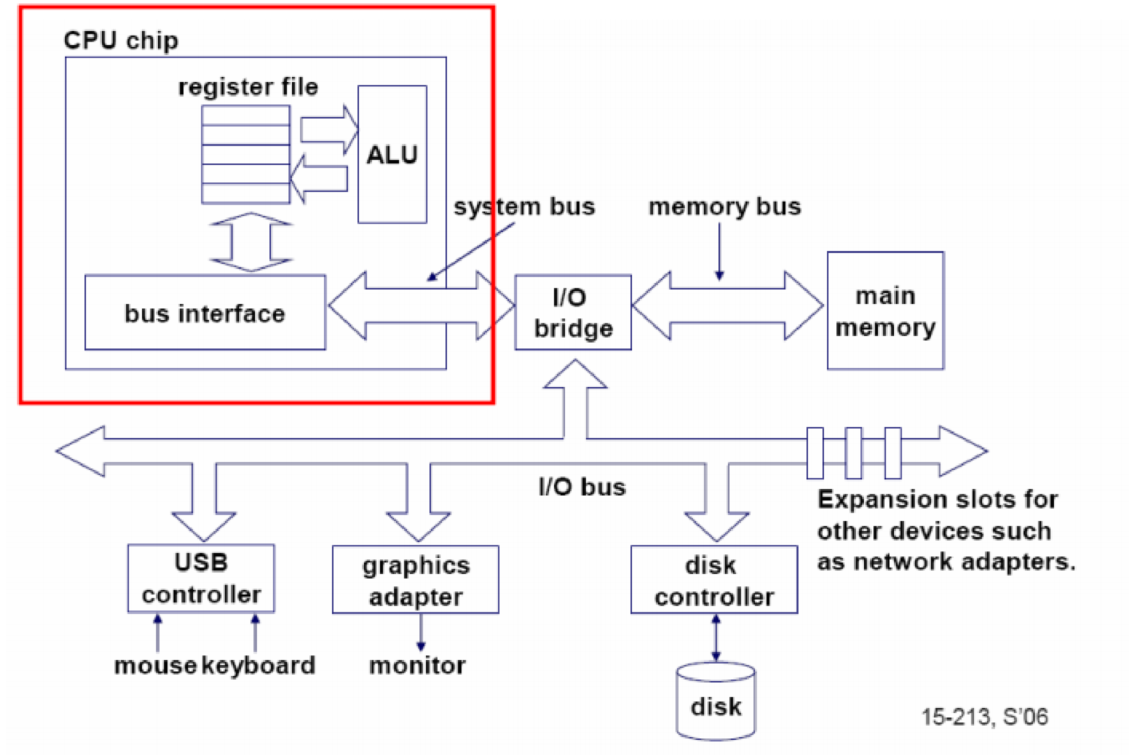
Teaching Assistants:
- Daniel López - daniel.lopez1701@alumnos.ubiobio.cl
- Sebastián González - sebastian.gonzalez1801@alumnos.ubiobio.cl

Course Website: http://www.face.ubiobio.cl/~jfuentes/classes/hc

Contents

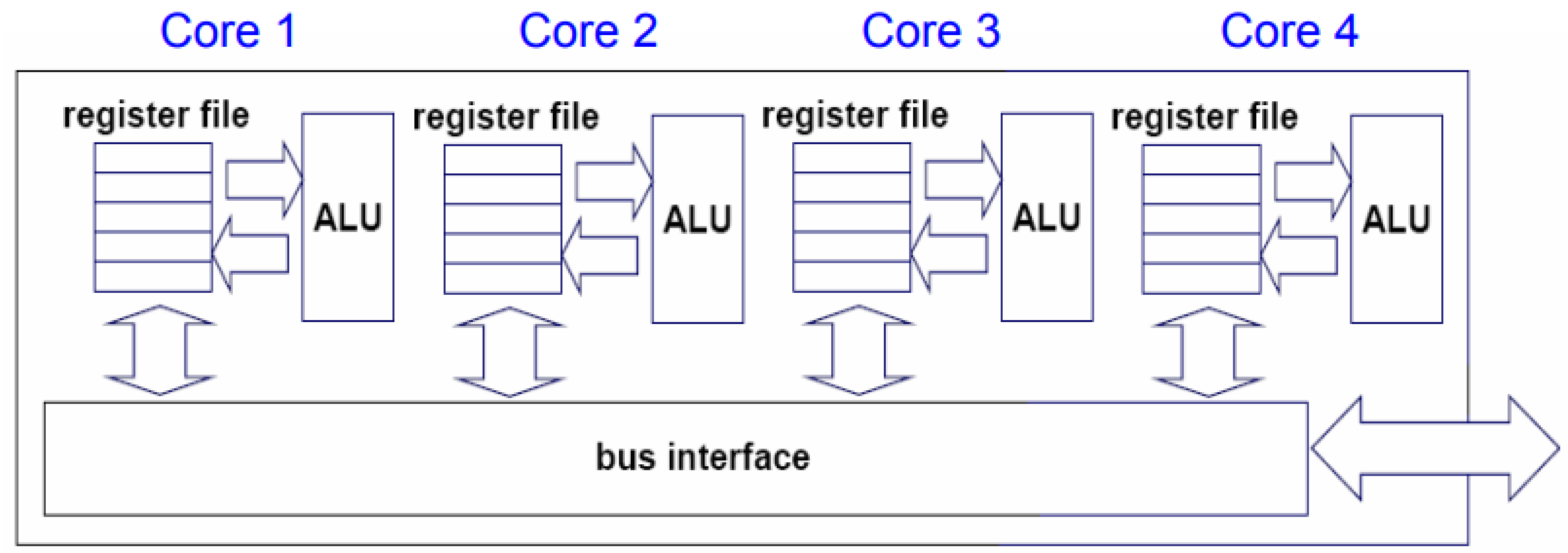
- Single-core CPU
- Multi-core CPU
- Cache coherence

Single-core CPU

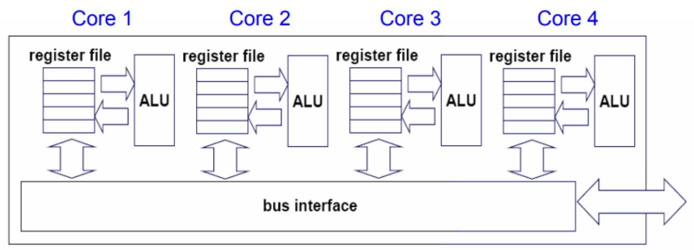


Multi-core architecture

- Main idea: duplicate multiple cores on the same processor.



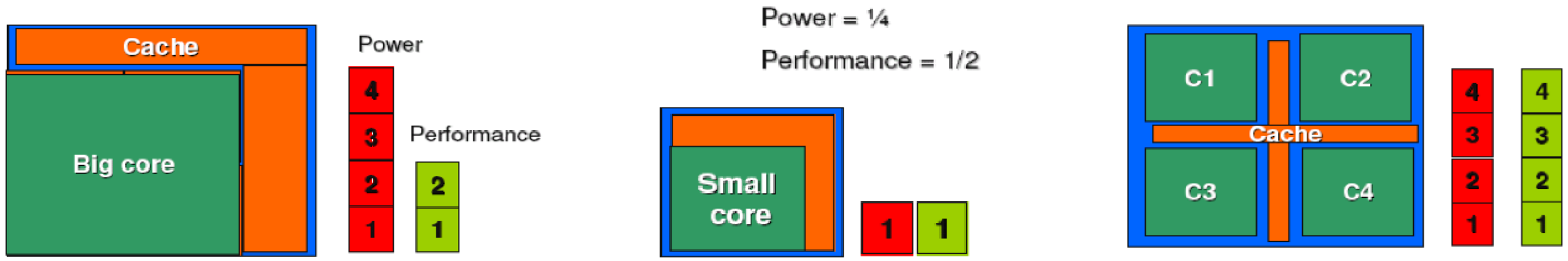
Multi-core architecture



- Why is multi-core a good idea?
- It’s very hard create single-core processors that work at high frequencies.
- Heat issues are better handled in multi-core.
- Benefits of parallelism in many apps. Better performance.
- Like single-core, multi-core also supports multi-task execution.

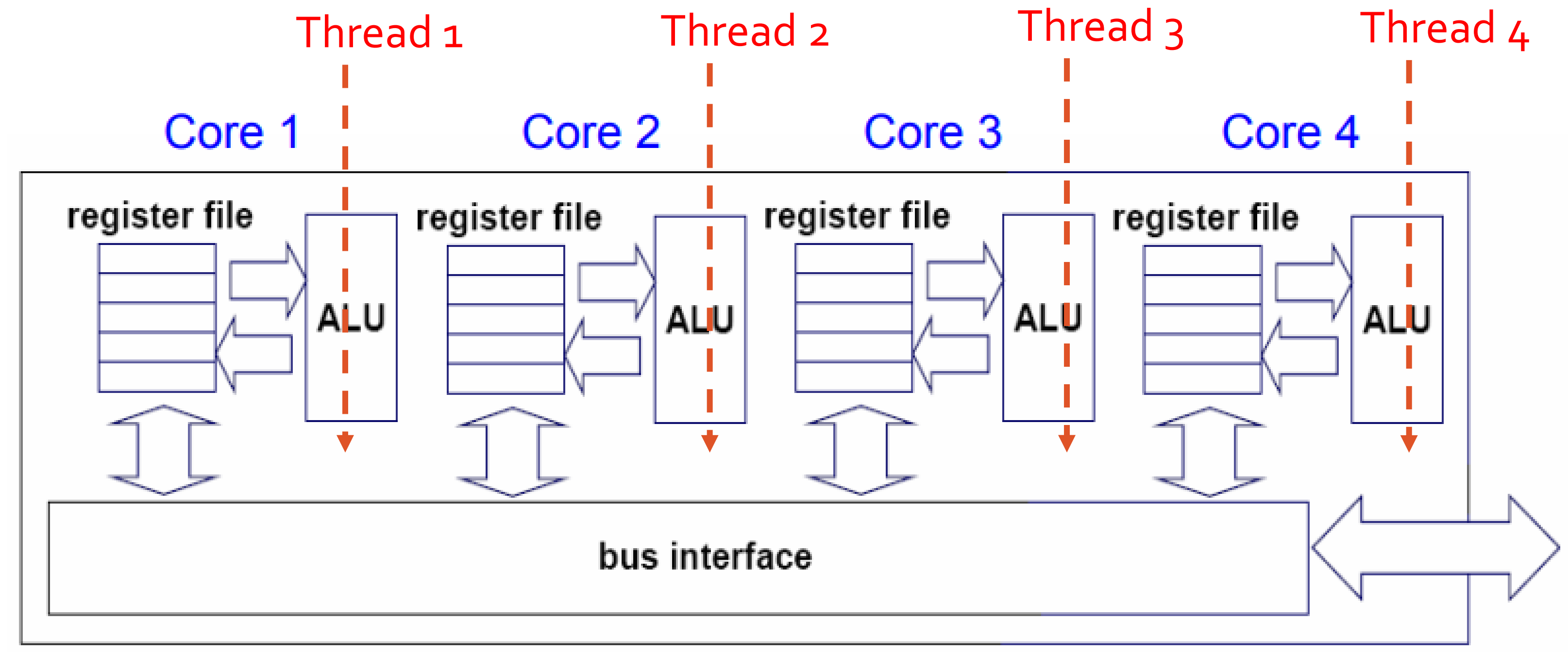
Multi-core architecture

- Why is multi-core a good idea?
- Multi-core delivers higher performance per watt



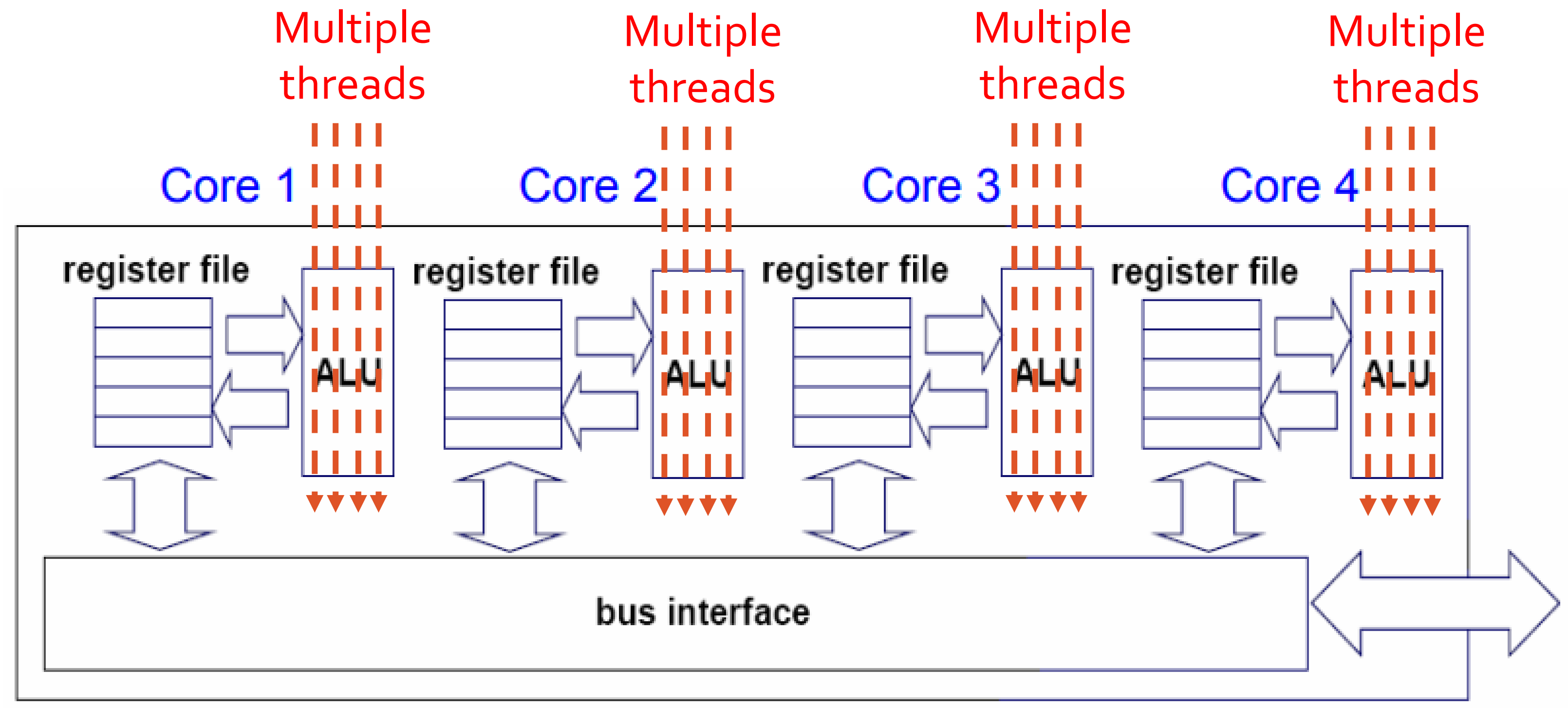
Multi-core architecture

- Cores run threads in parallel



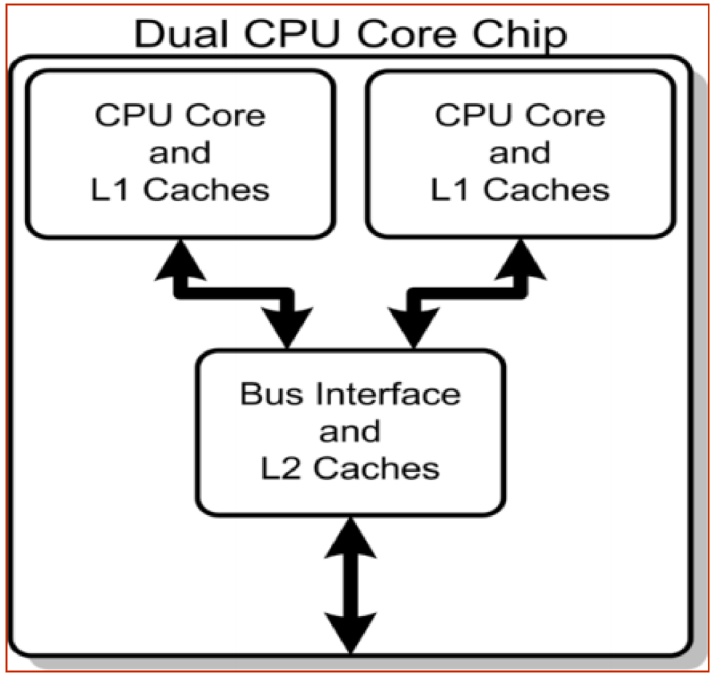
Multi-core architecture

- On each core, multiple threads are executed in an interspersed way.

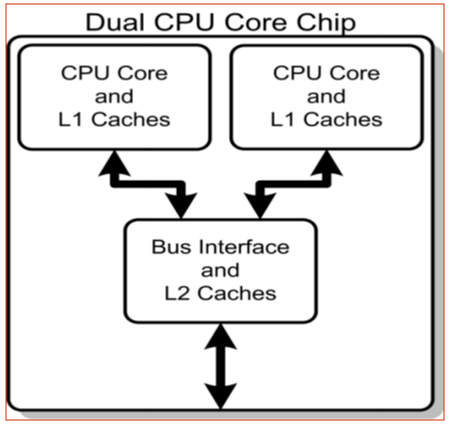


Multi-core architecture

- Conceptual diagram of a dual-core CPU
- Each core has L1 cache
- Both cores relate to L2 cache

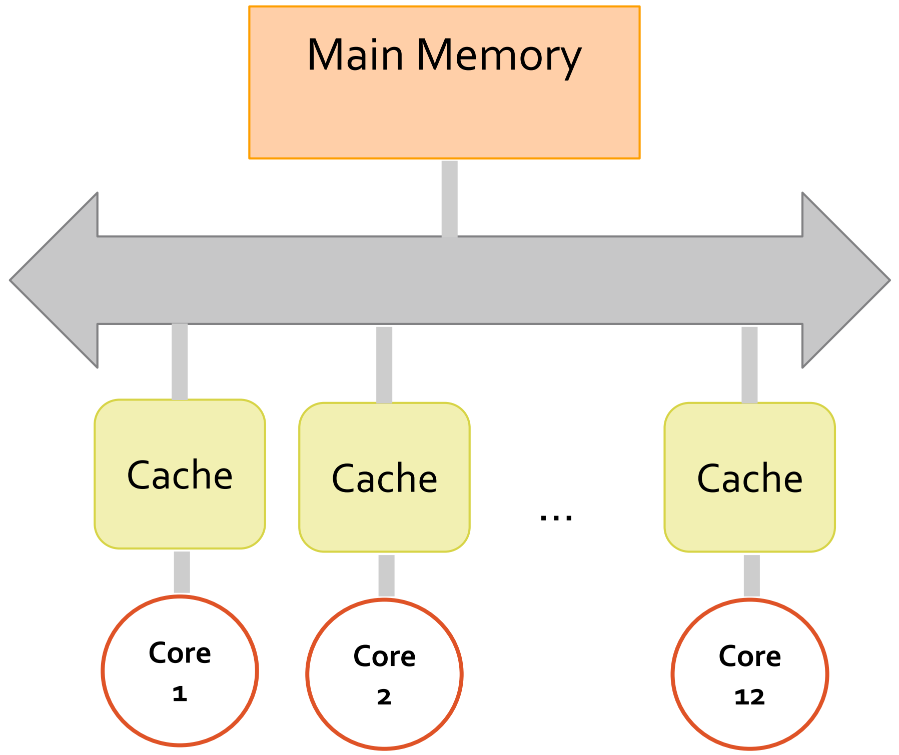


Multi-core architecture



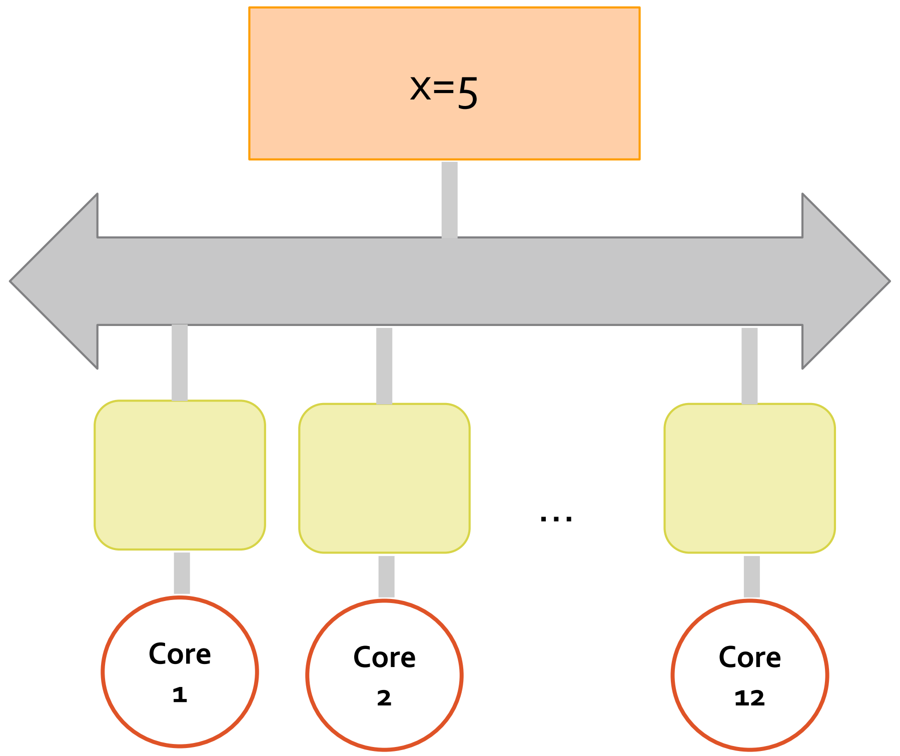
- Problems
- Memory consistency
- Operating system support
- Processor-cache affinity

Cache coherence



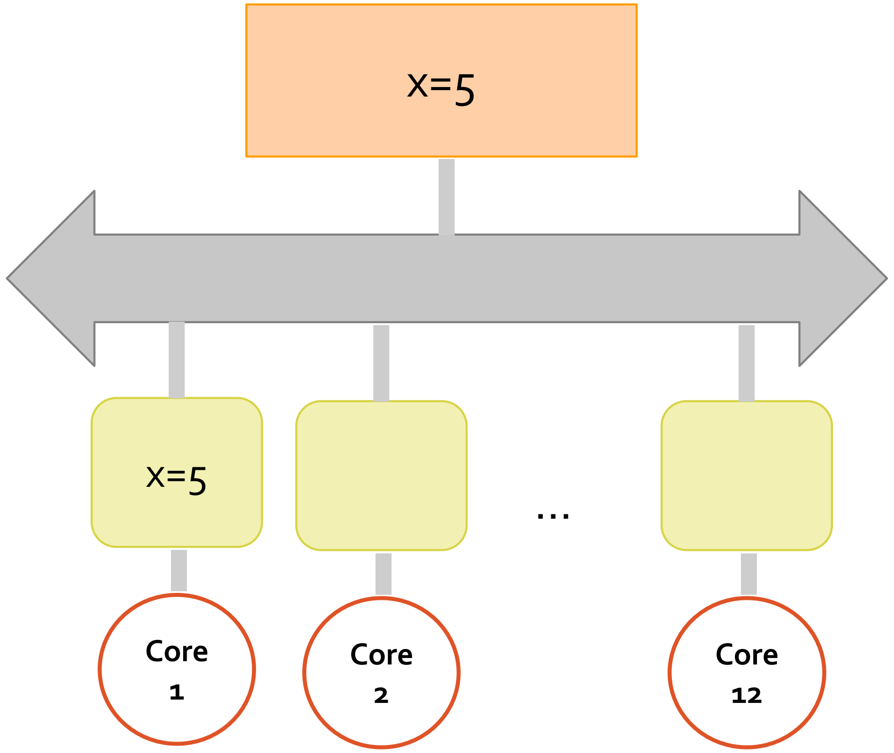
Cache coherence

- x has a value of 5 in main memory.



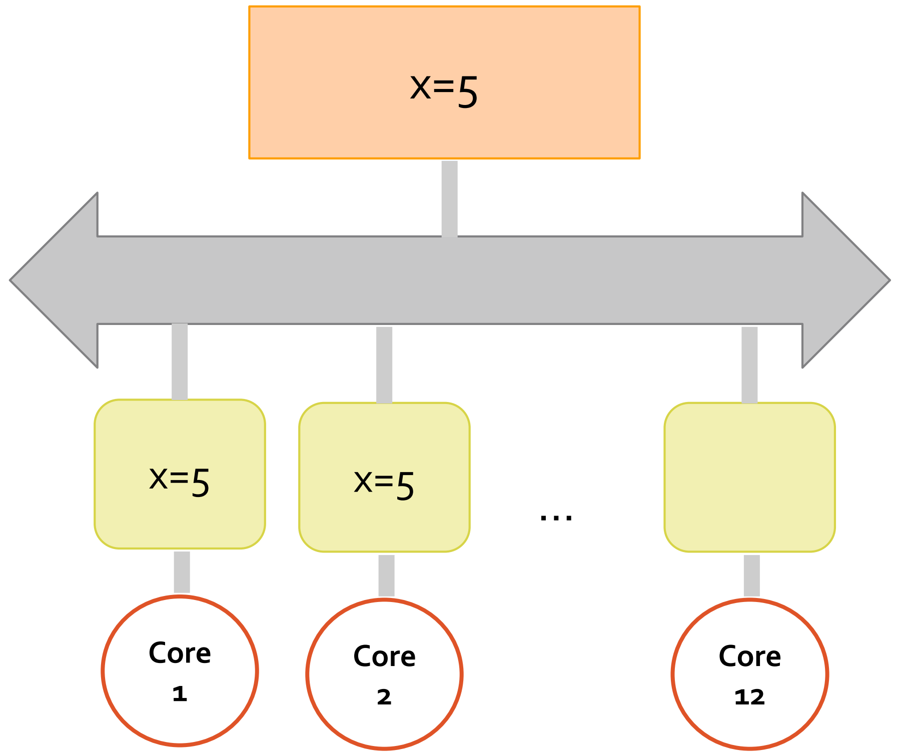
Cache coherence

- Core 1 reads x



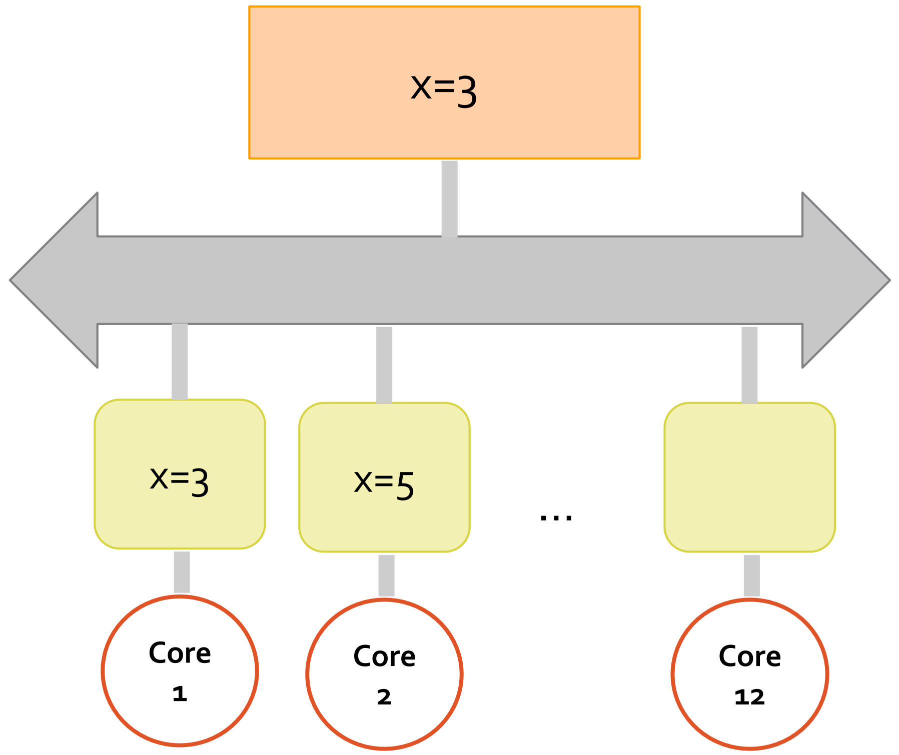
Cache coherence

- Core 2 reads x



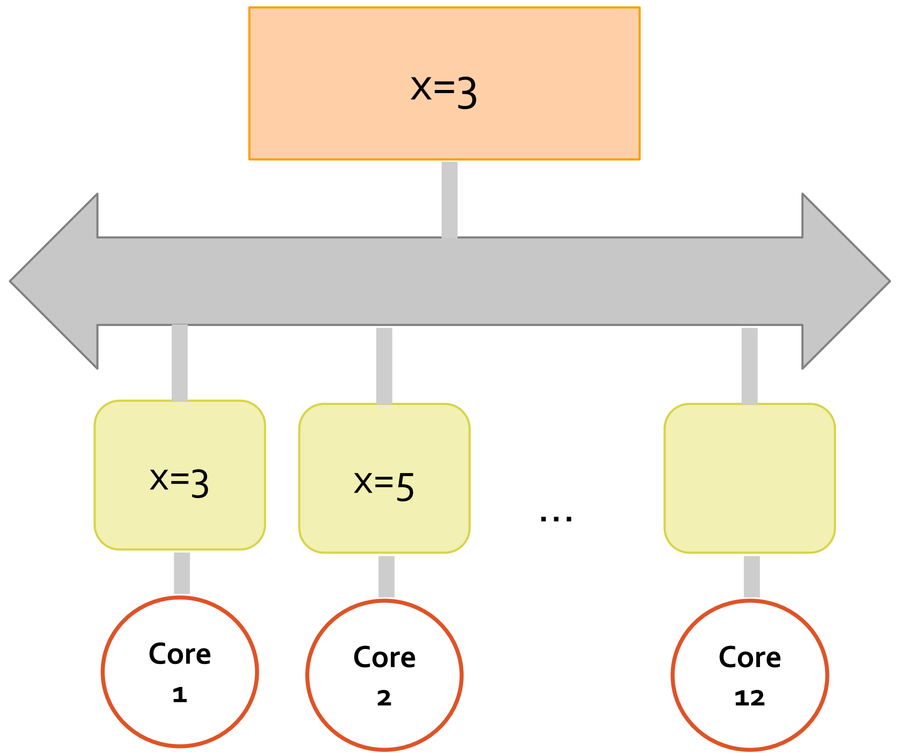
Cache coherence

- Core 1 writes x and assigns value 3
- X in main memory is updated



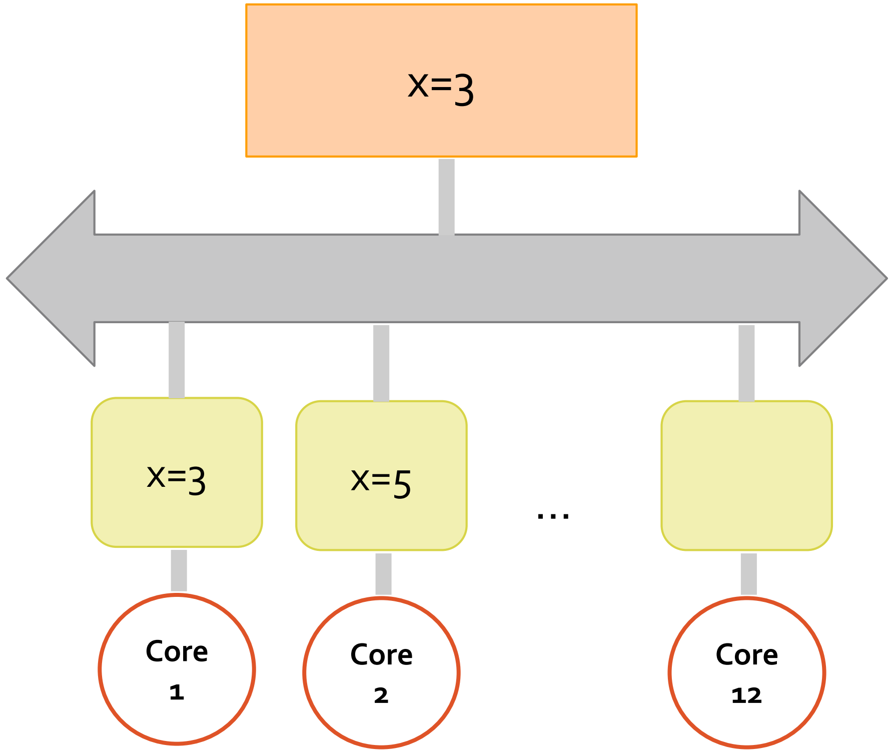
Cache coherence

- Core 2 reads x again but has an outdated copy
- Cached data consistency issue



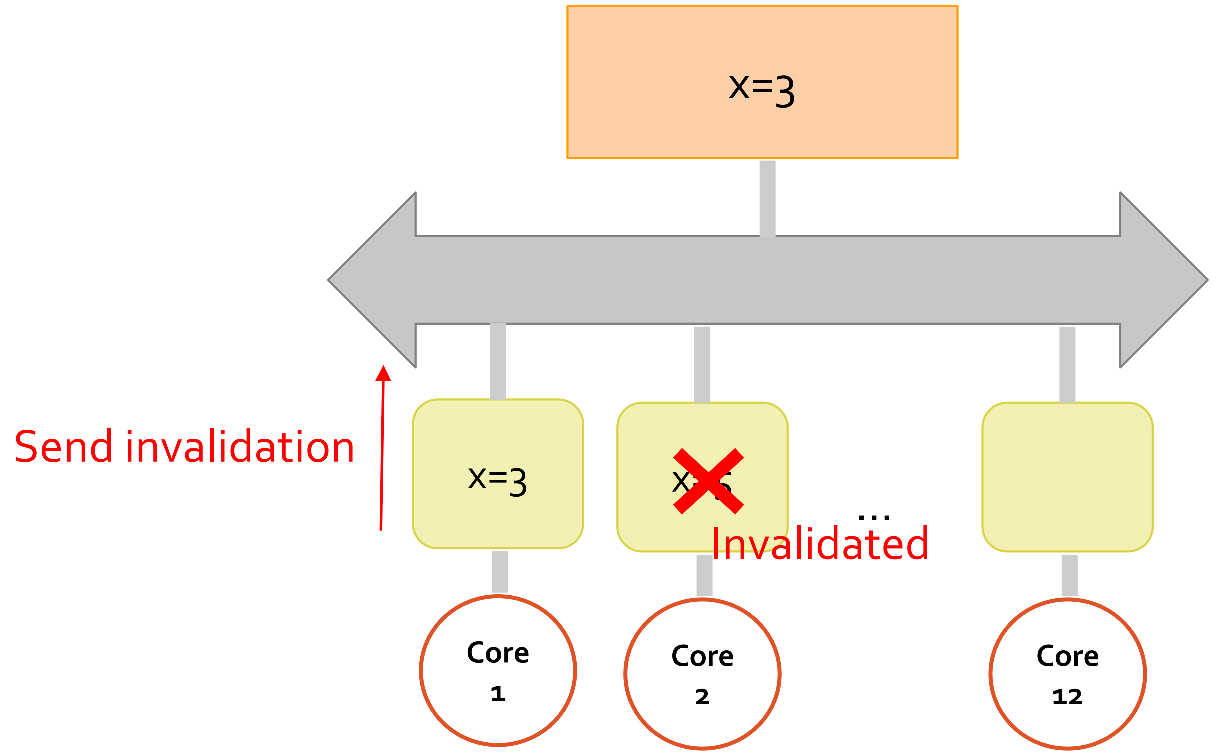
Cache coherence

- Possible solutions:
- Use consistency protocols: invalidation, snooping, update



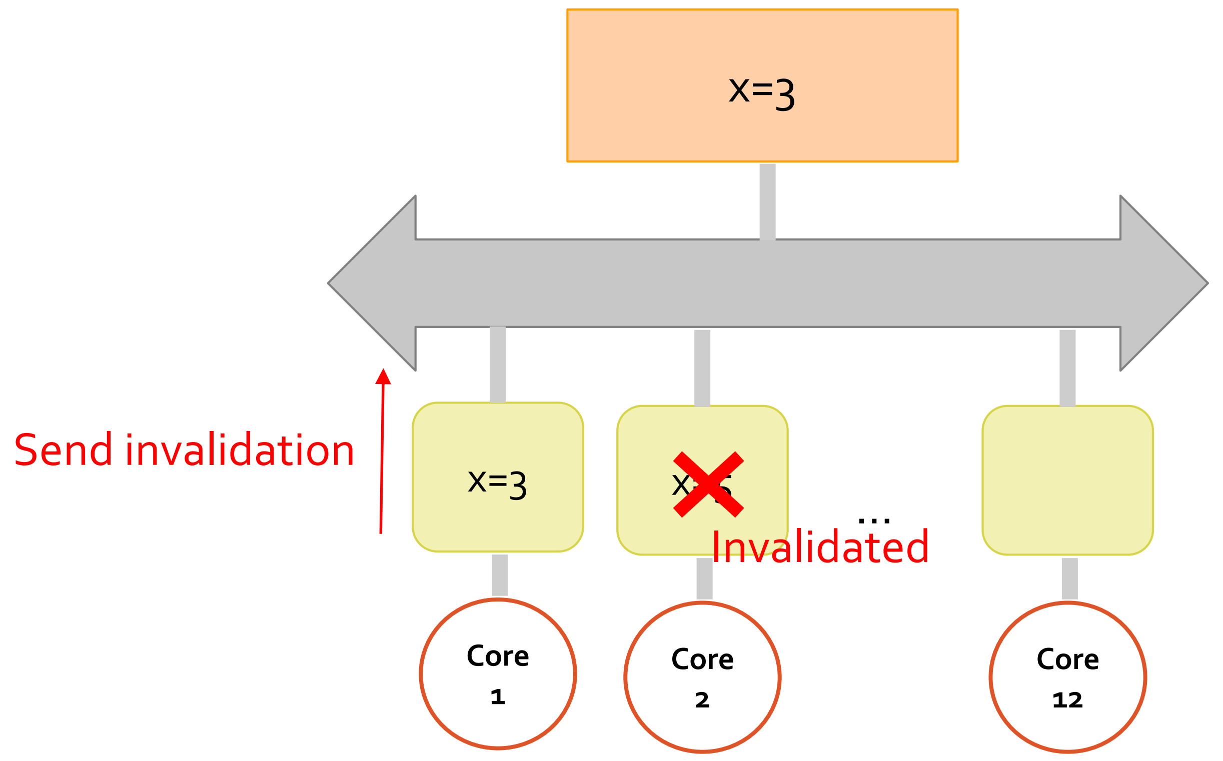
Cache coherence

- Invalidation protocol: If a core writes to an element, all other copies in other caches of the same element are invalidated



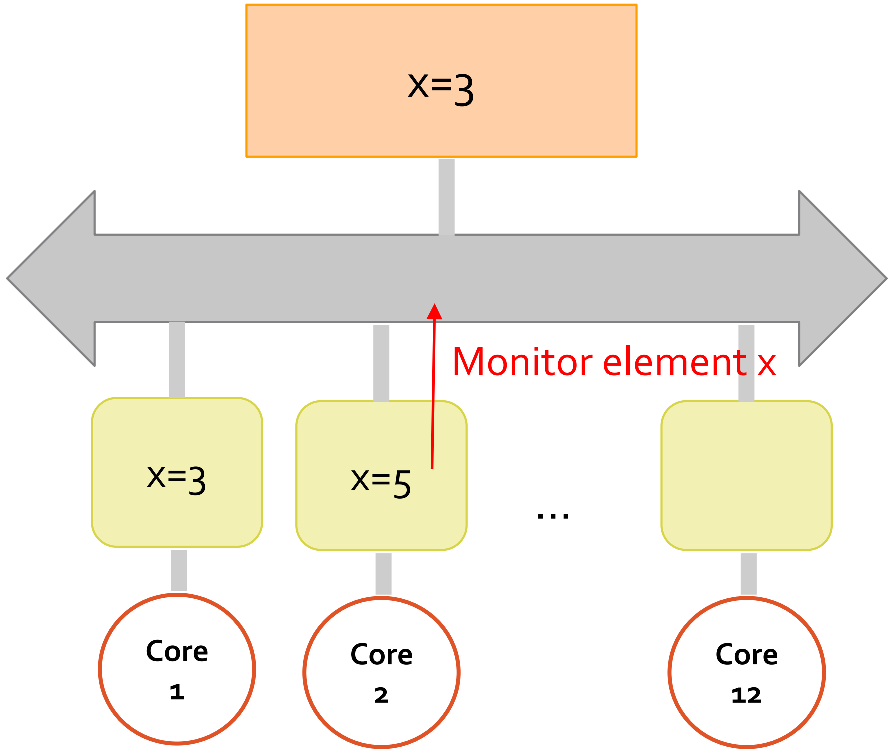
Cache coherence

- Invalidation protocol: If a core writes to an element, all other copies in other caches of the same element are invalidated.
- Invalid copies must read updated value from main memory



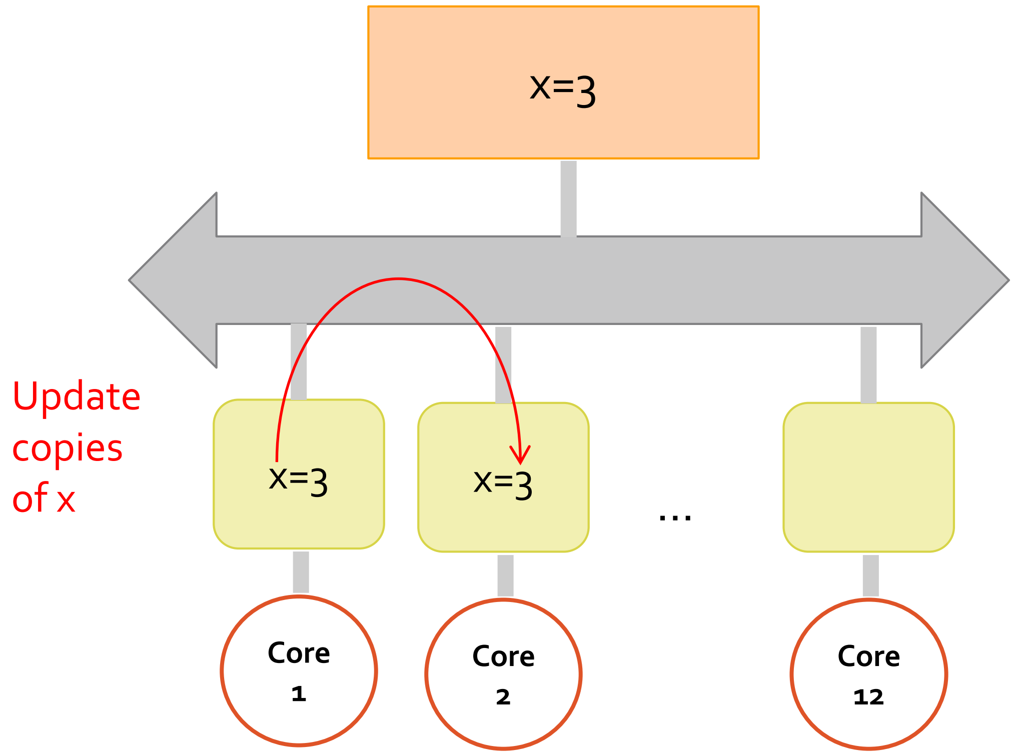
Cache coherence

- Snooping protocol: All cores continuously monitor the communication bus. If they detect that any value has changed, they update it.



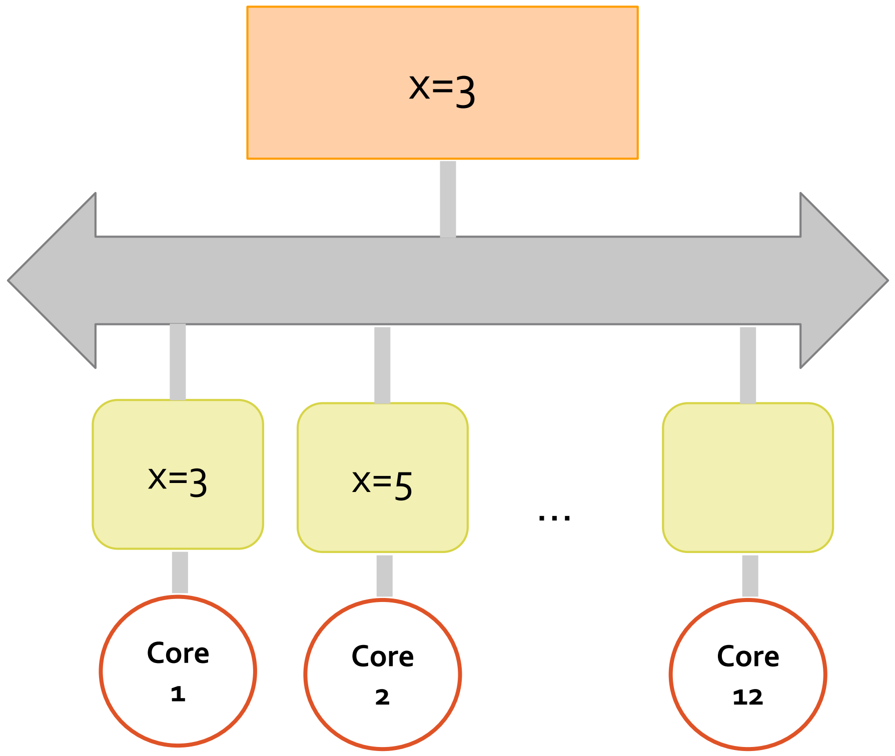
Cache coherence

- Update protocol: When a core updates a value in its cache, it must update its copies in all other core caches.



Cache coherence

- What is the best solution?
- invalidation, snooping, update?
- Invalidation: only the first time
- Update: You must transmit new writes each time
- Snooping: complex to implement



In general, invalidation has better performance, since it generates less traffic

To learn more about protocols: see MSI, MESI (Modified, Exclusive, Shared, Invalid)