Domande Computer Architecture

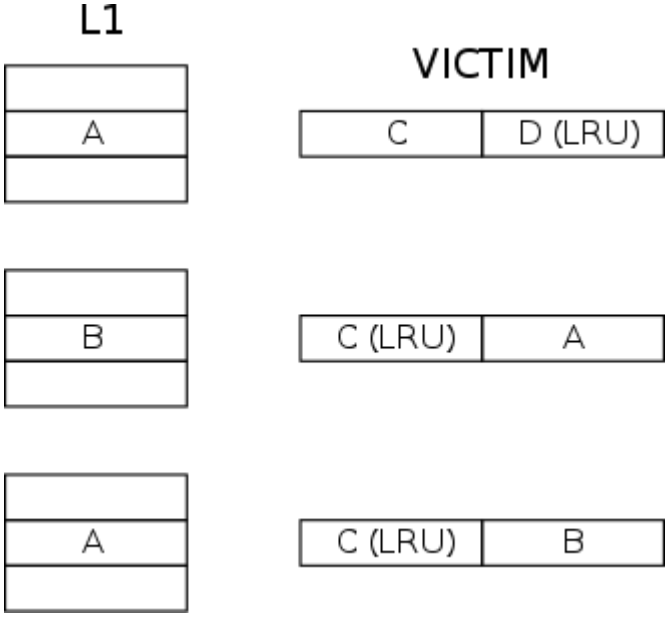
Raccolta di domande fatte durante le lezioni.

1. [presentazione sui droni] il drone lo costruisco per fare delle elaborazioni (esempio ricostruire la zona con tutte le altezze senza andare a fare misure dirette). Immaginiamo di disporre di un'applicazione in grado di costruire tutto il profilo di un palazzo, ha piú senso, dal punto di vista del consumo, utilizzare l'energia per spedire il filmato a terra oppure creare automaticamente le immagini sul drone e inviare le foto, parzialmente elaborate, a terra. (tipo watt per byte o watt per frame)

2. Devo ridurre i conflitti sulla stessa struttura offrendo due copie di tags, una per offrire il tag in caso di miss della cache di livello uno, una per rispondere alle operazioni effettuate sul bus condiviso. Se ho due copie, devo aggiornarle quando voglio cambiare queste due copie. Questo é un problema?
   L'intervallo di tempo in cui devo cambiare i due valori del tag, potrebbe causare un effetto collaterale sulla cache di primo livello o sul bus condiviso.
   Il problema é che l'attivitá tradizionale del primo livello di cache é risolvere le miss del secondo livello e per cercare quale blocco caricare bisogna cercare attraverso il tag.
   Quando devo aggiornare i tag devo eseguire un'azione atomica per evitare incosistenze.
   Questa update puó essere un problema dal punto di vista delle performance del processore?
   Quando devo aggiornare entrambe le copie del tag? In caso di miss nel secondo livello di cache, devo scrivere un nuovo valore nelle copie di tag.
   La soluzione é che la condizione per la quale devo aggiornare entrambi i tag é nel caso di miss nella cache di secondo livello, ma quando lo faccio uso il bus condiviso per chiedere una copia del blocco coinvolto nella miss, quindi il bus é occupato da un'operazione richiesta dalla cache di secondo livello, quindi non c'é un drop delle performance.

3. Che side effect potrebbe avere la inclusion property? Nell'ultimo microprocessore intel la inclusion property non é garantita perché puó portare a problemi dal punto di vista energetico. Il problema é che se i tre livelli di cache devono avere gli stessi blocchi, la dimensione massima della cache é data dall'ultimo livello di cache, quella piú piccola e questo puó peggiorare le performance. La soluzione é l'uso di una victim cache che permette di usare una copia locale senza richiederla attraverso il bus condiviso.
   La inclusion property é usata per ridurre la complessitá del sistema ma questo puó portare a problemi di performance.
   Come si puó risolvere? **Victim cache**, Miss caching places a fully-associative cache between cache and its re-fill path. Misses in the cache that hit in the miss cache have a one cycle penalty, as opposed to a many cycle miss penalty without the miss cache. Victim Caching is an improvement to miss caching that loads the small fully-associative cache with victim of a miss and not the requested cache line.
   A victim cache is a hardware cache designed to decrease conflict misses and improve hit latency for direct-mapped caches. It is employed at the refill path of a Level 1 cache, such that any cache-line which gets evicted from the cache is cached in the victim cache. Thus, the victim cache gets populated only when data is thrown out of Level 1 cache. In case of a miss in Level 1, the missed entry is looked up in the victim cache. If the resulting access is a hit, the

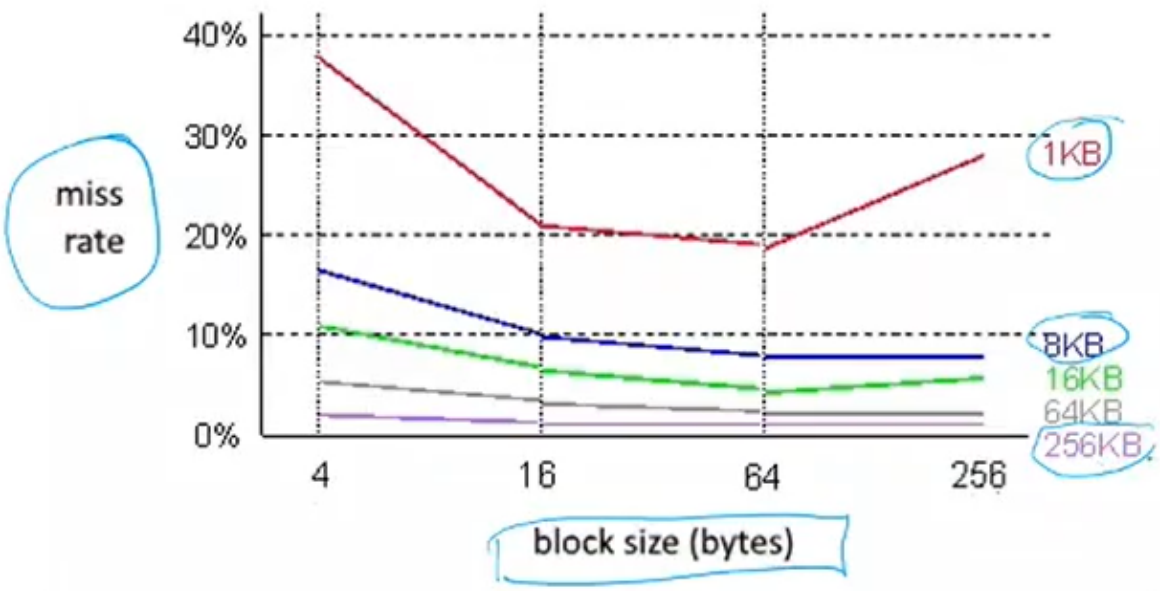
contents of the Level 1 cache-line and the matching victim cache line are swapped.



4. Perché all'interno di un multiprocessore non ho problemi di workload-balancing?

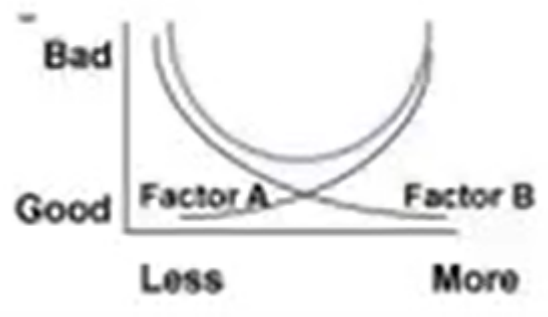
Il workload bilanciato puó essere ottenuto automaticamente tramite lo scheduler che offre un task diverso ad ogni processore. L'unico problema é riuscire a fornire un numero di processi pronti tale da fare operare il sistema alla massima utilizzazione e minimizzare la comunicazione tra processi.

5. Andamento miss rate in base alla grandezza del blocco. Vario la dimensione del blocco in bytes (4-16-64-256) e la dimensione della cache in kB (1-8-16-64-256). Perché ha quell'andamento? (**domanda tipica**)



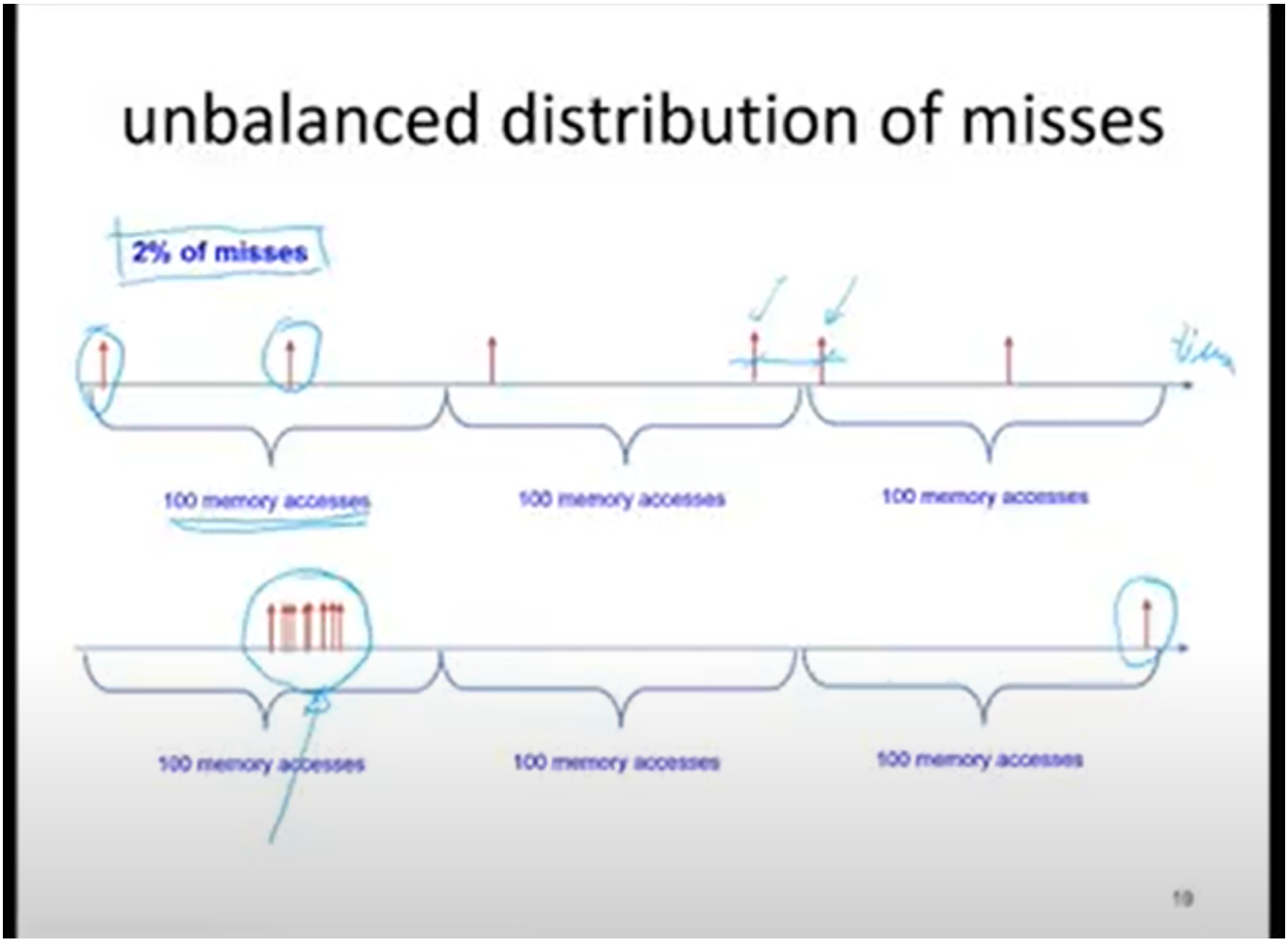
In generale la miss rate va giú quando la dimensione del blocco aumenta. Nel caso della cache da 1 kB invece la miss rate aumenta.
Una soluzione é migliore nel caso cerchiamo *spatial locality*, una nel caso cerchiamo *temporal locality*.
Nel caso di temporal locality voglio avere blocchi piccoli (4 bytes) perché voglio caricarne il piú possibile.

Nel caso di spatial locality vogliamo avere la porzione piú grande di memoria possibile attorno all'indirizzo della miss. (massimizzare la dimensione del blocco)
Tornando al caso della cache da 1 KB, in caso di blocchi da 256 bytes avró solamente 4 blocchi in cache, che é la migliore situazione per la spatial locality, ma decisamente la soluzione non ottimale per la temporal locality.
Nel caso di blocchi da 4 bytes avró 256 blocchi.
Dobbiamo quindi cercare la dimensione ottimale che solitamente é vicina alla dimensione di una word in cache moltiplicata per quattro.
Soluzione = sizeof(word) * 4.



Il valore ottimo peró potrebbe dipendere da applicazione a applicazione, dato che cambia in base al tipo di accessi che fará l'applicazione in cache.

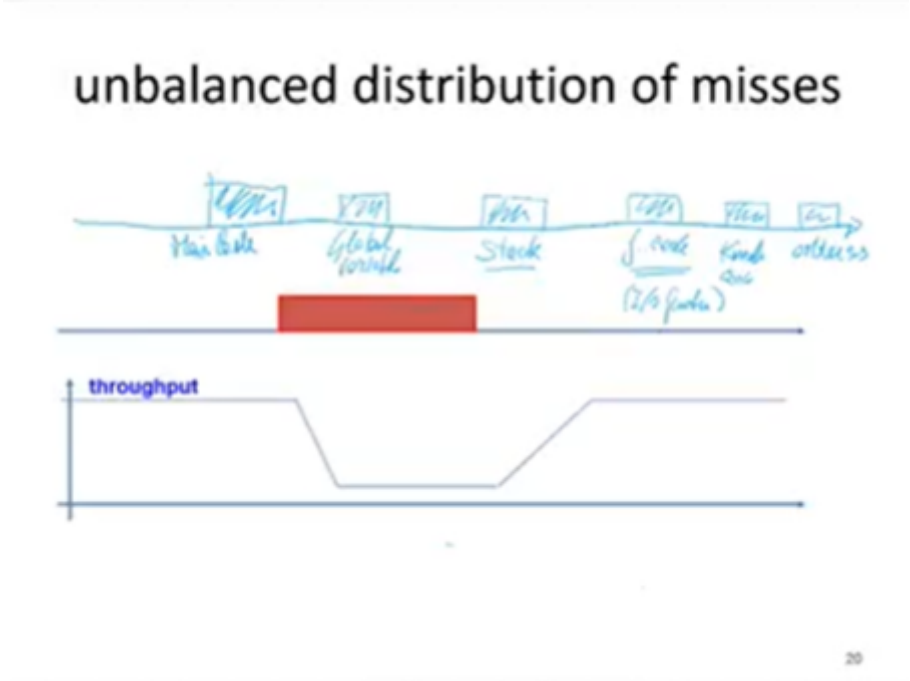
6. Perché potrebbero esserci molte miss in un intervallo molto ristretto di tempo e una molto dopo?



- chiamate di funzioni (call)
- cambio di contesto tra processi
- interrupt (int)

7. Quali sono le possibili aree nelle quali avró gli accessi? (legata a quella precedente)

- main code
- global variables
- stack
- codice di funzioni: codice usato per chiamare le librerie, funzioni I/O
- codice del kernel
- dati del kernel



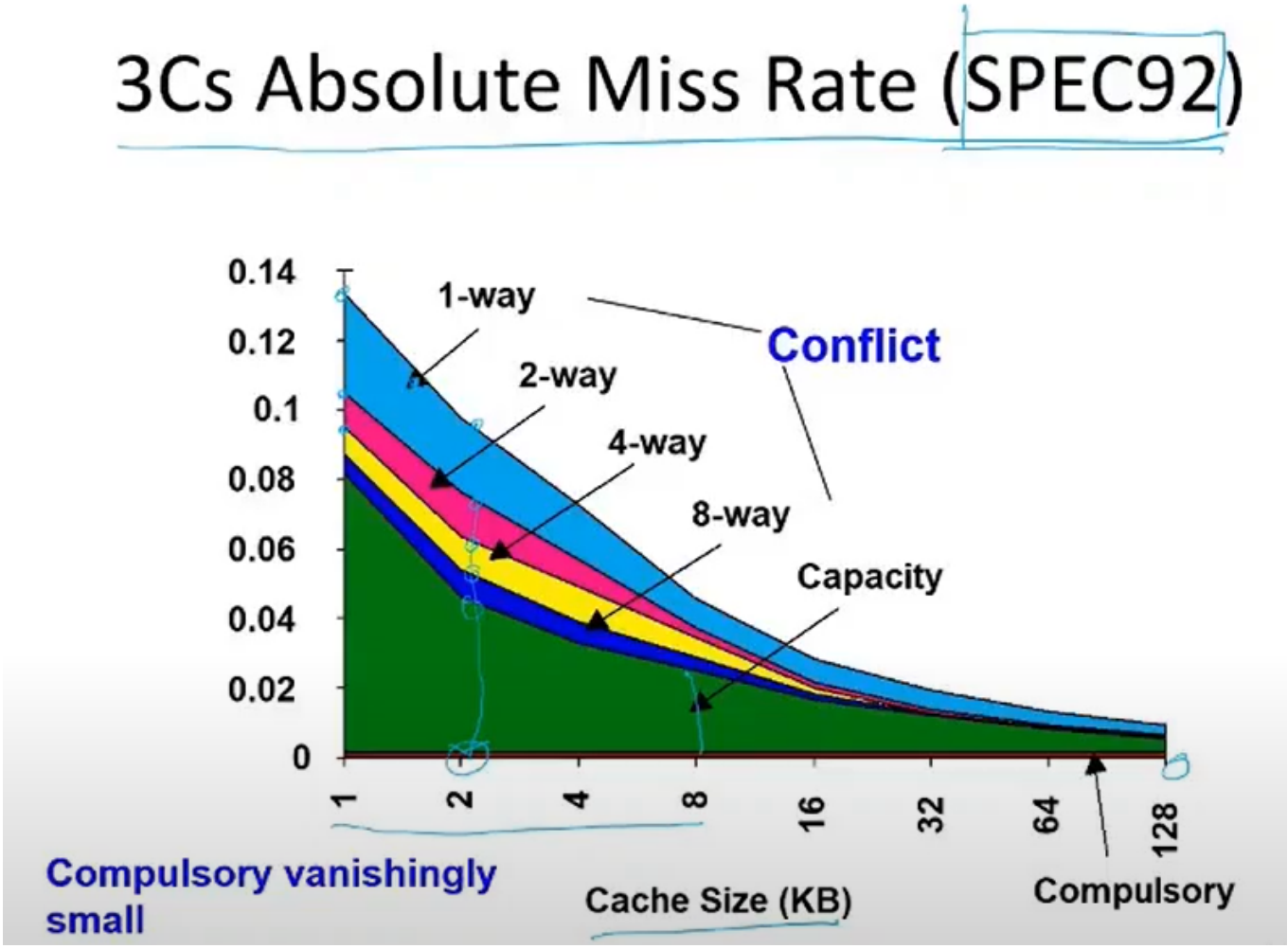
Conclusioni che possiamo trarre?
Come posso ridurre il numero di miss?

Una buona idea é separare le cache per i dati e le cache per il codice.
2 cache, una per il codice e una per i dati.
Da una parte posso leggere i dati, dall'altra fare il fetch di una nuova instruzione. Posso aumentare il parallelismo tra l'accesso ai dati e l'accesso al codice.
Possiamo organizzare questi due livelli di cache in modi diversi, dato che i meccanismi di localitá spaziale e temporale delle cache dei dati e del codice sono molto diversi.
Direct-mapped cache o set-associative?
Set-associative, potrebbe esserci dei conflitti dati dal fatto che potrebbero essere eseguite delle instruzioni contenute nello stesso blocco.

Quanti blocchi per set?

4-way associative cache, perché altrimenti un loop (grande) potrebbe eliminare le copie in cache appartenenti allo stesso loop.

8. Come possiamo valutare la distribuzione delle miss per una cache quando il processore sta eseguendo un particolare programma, riuscendo ad individuare la causa della miss?



Per la causa *compulsory* bisogna considerare il numero di blocchi usati per accessi in sequenza.
Per *capacity* bisogna considerare la diminuzione della capacitá della cache stessa.
Per le *way*, si usa la **2:1 cache rule**.

9. Perché i microprocessori attuali usano una cache ad indirizzi fisici al posto di una ad indirizzi virtuali?

Perché due indirizzi virtuali diversi possono mappare lo stesso indirizzo fisico (alias), che invece é univoco.
(detto meglio) Le pagine fisiche possono essere condivise tramite due indirizzi virtuali che puntano allo stesso indirizzo fisico, questo serve per permettere a programmi diversi di condividere porzioni di dati o di codice.

10. Ad un cambio di contesto posso avere molte miss nello stesso momento, come posso evitarlo?

Il problema si verifica quando due o piú processori hanno un context switch nello stesso momento. Per risolvere quindi basta non permettere di avere cambi di contesto nello stesso momento.

Domande trovate su cartella anni passati

Cache:
- Why?
- principles of locality
- Cache organization (direct, set, fully associative)
- hit time and miss penalty
- type of misses (3C)
- Replacement policy
- istruction and data cache
- write policy, cache levels
- inclusive and exclusive cache
- victim cache

GPU:
- differences with CPU
- heterogenous architectures,
- memory latency in this archtectures
- SIMT
- CUDA (thread and memory organization,structure of code)

Multiprocessor:
- Multiprocessor vs Multicomputer,
- Consistency vs Coherence,
- definitions of consistency
- UMA Architecture,
- Cache Coherence with Snooping Cache: write-throught, write-back
- (MESI, MOESI, MSI protocols)
- Crossbar interconnection
- omega network
- NUMA architecture
- Cache coherence with directory-based protocol
- Quali sono le possibili miss? Come si possono misurare le percentuali dei vari tipi di miss.
- Differenza tra throughput, latenza e banda. Cos'è più facile ottenere adesso?
- Differenza tra concorrenza e parallelismo.
- Processori superscalari. Cosa sono, cosa significa esecuzione out of order
- Dependability e scalability
- Che cos'è la scalabilità e che cos'è un collo di bottiglia? Che legame c'è tra i due.
- Quali fattori definiscono una word
- Arm Cortex A8, struttura della memoria
- Livelli di parallelismo, architetturale e applicazione
- Come deve struttura il codice il programmatore per sfruttare la parallelizzazione dell'architettura?
- Struttura cache a 2 vie (con disegno). Least Recently Used e Least Frequently Used