Master Program in Computer Science and Networking

High Performance Computing Systems and Enabling Platforms

a.a. 2009-10

secondo appello, 24 June 2010

The answer can be written in English or in Italian.

Write name and surname at the top of the first page of each foolscap.

Present the answer in a legible form (writing in pencil is accepted, if preferred).

Consultation of any kind of didactic material or notes is not permitted.

Don't submit possible rough copies.

Results and date of oral exam will be published in the course page as soon as possible.

Question 1

Consider a NUMA multiprocessor architecture with 128 nodes.

Each node includes a D-RISC pipelined CPU without secondary cache. Data cache blocks are 4-word wide. Local memory locations are 128-bit wide.

The maximum capacity of main memory is 512 Giga words.

The interconnection structure is of logarithmic kind, with one-word links and wormhole flow control.

All the system processing units have the same clock cycle τ, except the local memory units having a clock cycle equal to 10τ. Any inter-chip link has a 4τ transmission latency.

a) Explain the architecture of a generic node. Show the structure of firmware messages used for local memory accesses and for remote memory accesses, and explain how and where they are built.

b) Determine the base latency of local memory accesses and of remote memory accesses.

c) Explain the qualitative impact of parameter *p* (the average number of nodes accessing the same memory module) on the memory access latency. Estimate a reliable value of *p* for a system executing a parallel program structured as a 16-stage pipeline, where each stage is a 4-worker farm.

d) Evaluate the interprocess communication latency approximately, in terms of the *base* memory access latencies only, assuming zero-copy communication and 4-word messages. Referring to the example in point *c)*, estimate the approximation degree of this evaluation compared to the evaluation in terms of under-load memory access latencies.

Question 2

Let's assume that the source code of run-time support for *send* and *receive* communication primitives is available in a uniprocessor version for a given assembler-firmware architecture S.

We try to exploit this code for other kinds of architecture at best. Explain which parts can be preserved, and which parts have to be modified/replaced *and in which way*, for

a) SMP

b) NUMA

c) Cluster

architectures exploiting the given uniprocessor architecture S as building block. Explain possible specific features required to S for the *a), b), c)* implementations.

Solution outline

(to be expanded properly)

Question 1

The solution has to be built as the composition of some course parts about multiprocessor architecture (sect. 4). Provided that the course theory is known, this requires the ability to build such composition in a synthetic, yet complete, way and to establish proper interrelations.

Moreover, several prerequisites about basic architecture concepts are necessary (e.g., system levels, node architecture, firmware communications, addressing, processes, etc.), as well as the proper approach to computer architecture and parallel paradigms.

a, b) Sect. 4.1, slides 4, 5, 6; Sect. 4.2, slides 21, 22, 23, 24, 25, 26, 27, 28.

For the local memory accesses, it has to be considered that links inside the node are not necessarily 1-word wide. The request from the CPU to the W interface units is related to a cache block transfer, and consists of (physical address, operation code and indivisibility bit) in parallel, followed by a word stream for block writing operations. The request is routed by W towards the local memory (there is no specific formatting), where 4 words are read in parallel (128-bit long word) and sent back to W and then CPU in pipeline over a 1-word link. With this premise, the local access latency is evaluated applying the same method for the remote access.

c) Sect 4.2, slides 29, 30, 31, 32, 33. In the specific example, there are 96 processes (each farm is composed by emitter, collector and 4 workers; the collector of any stage is connected to the emitter of the next stage) mapped onto the NUMA nodes, one process per node. Channel descriptors are allocated in the local memories of the respective destination processes. Thus, there are some memory modules with 4, 5, or 6 accessing nodes (including the local one), while other modules have just 2 accessing nodes (an average estimate of p may be 4). Because of the low value of p, the under-load memory access latency is very close to the base latency.

d) Sect. 4.3, slides 55-59. A more accurate evaluation distinguishes local *vs* remote memory accesses.

Question 2

The basic uniprocessor implementation is described in Sect. 3.

The multiprocessor (SMP and NUMA) version requires the proper manipulation of channel descriptor in lock state: sect. 4.3, slides 17, 18. The critical section code is preserved wrt the uniprocessor version, since we are in a shared memory architecture. The minimization of critical sections length may introduce further modification to the uniprocessor code.

Low level scheduling, and process wake-up in particular, has to be properly replaced in SMP and NUMA run-time support (slides 20-24). The uniprocessor code for ready list manipulation can be preserved (in lock state for SMP).

Proper assembler and firmware features must be provided by S for indivisible memory access sequences. Possibly, annotations for non automatic cache coherence could be provided.

In the cluster version (sect. 5), with uniprocessor nodes, the local send execution in the destination node (interrupt handler) is very similar to the uniprocessor version, except that there are no actions related to the buffer_full condition. Moreover, a proper communication to the sender instance has to be added in the receive implementation. All the actions and data structures related to the sender process have to be introduced.

Architetture Parallele e Distribuite

Domanda 1

Vedi SPA.

Domanda 2

Si consideri un programma parallelo Σ con struttura cliente-servente, con *n* clienti identici ed un servente. Come risultato visibile all'esterno, Σ produce uno stream costituito da tutti i valori calcolati dai clienti.

a) Spiegare come si valutano i parametri banda ideale, banda effettiva ed efficienza relativa:

1) per il servente preso a sé stante, *2)* per il generico cliente preso a sé stante, *3)* per Σ nel suo complesso.

b) Tanto per il servente quanto per Σ, mostrare e spiegare l'andamento grafico qualitativo della banda e dell'efficienza relativa in funzione di *n*.

c) Mostrare quanto richiesto al punto *b)* facendo variare anche il parametro grado di parallelismo *m* del servente.

Una volta spiegato il sistema di equazioni che regola il funzionamento di una computazione cliente servente, dalla comprensione qualitative della soluzione di tale sistema discendono tutte le risposte.

a, b)

- Servente: la banda ideale vale $1/T_S$, quella effettiva è uguale all'inverso del tempo di interarrivo ed è monotona crescente tendente al valore ideale. L'efficienza relativa è uguale al fattore di utilizzazione, che tende asintoticamente a uno al crescere di *n*.
- Cliente: la banda ideale vale $1/T_G$, quella effettiva è uguale a $1/(T_G + R_Q)$ ed è monotona decrescente tendente a zero al crescere di *n*, e così l'efficienza relativa.
- Σ: la banda ideale vale n/T_G, quella effettiva coincide con quella del servente. L'efficienza relativa coincide con quella del generico cliente.

c) trasformare le curve del punto *b)* in famiglie di curve con parametro *m*:

- Servente: al crescere di *m* aumenta la banda e diminuisce l'efficienza.
- Σ: al crescere di *m* aumenta la banda e aumenta l'efficienza.

Queste prestazioni sono influenzate, ma non dal punto di vista dell'andamento qualitativo generale, anche dalla forma di parallelismo del servente, che ha impatto sulla latenza, e quindi sul tempo di risposta del servente.