Master Program in Computer Science and Networking

High Performance Computing Systems and Enabling Platforms

a.a. 2009-10

quarto appello, 9 September 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 the run-time support of interprocess communication primitives for multiprocessor architectures.

a) Individuate, in the code of the run-time support, the occurrences of *i)* locking critical sections, and of *ii)* indivisible sequences of memory accesses, and explain the difference between *i)* and *ii).*

b) Explain where and why there are some code sections that are equal, and some code sections that are different, in the run-time support for SMP and for NUMA architectures.

c) Assuming all-cache multiprocessor architecture, zero-copy communication, and fair locking operations, explain the differences between the run-time support for automatic and for non-automatic cache coherence implementations.

Question 2

Consider pipelined CPU architectures.

a) Give a proof of the cost model of the instruction service time for D-RISC machines.

b) Explain the impact of the Execution Unit bandwidth and of the Execution Unit latency on the instruction service time (in general, not for D-RISC machines only).

c) Consider the following parallel organizations of the Execution Unit:

1. farm, where any worker is able to execute any pipelined arithmetic operation,
2. functional partitioning, where distinct subsets of pipelined arithmetic operations are executed by distinct workers,

and compare solutions 1, 2.

Solution outline

(to be expanded properly)

Question 1

a) The required run-time support is described in details in the lecture notes. Locking critical sections are recognized in the high-level pseudo-code as *lock(sem) ... unlock(sem)* brackets (manipulation of channel descriptor and message copy, possible mutual exclusion in low-level scheduling operations), while indivisible sequences of memory accesss are recognizable inside the lock-unlock codes themselves. Both mechanisms are mutual exclusion mechanisms that imply busy waiting, while the differences depend on

- the kind of utilization: locking is used to implement *sequences of operation* in an indivisible way, *not necessarily* for mutual exclusion of shared data manipulation (though this case is frequent), while the other mechanism is just to guarantee the *mutual access to shared memory modules*;
- performance considerations: it is not convenient to implement "long" critical sections (provided that they consist in the manipulation of shared data) by means of indivisible sequence of memory accesses, while very "short" critical sections, that consist in the manipulation of shared data, are efficiently implemented by means of indivisible sequence of memory accesses, as in the indivisible implementation of lock-unlock operations.

b) All the sections that implement the communication per se (message copying) and synchronization are the same in SMP and NUMA, because both are shared memory architectures and the differences in memory organizations are not visible in the process codes that use *logical addresses*.

SMP and NUMA implementations differ in the implementation of low-level scheduling operations: see lecture notes.

c) See the send-receive implementations with automatic and non-automatic cache coherence in the lecture notes. In the automatic case the cache coherence issues are not visible in the run-time support code, while they are visible in the non automatic case. In this last case, proper and explicit operations are provided for *modification* of the main copy of shared data and for *allocation-deallocation* of cache blocks.

The answer must contain the detailed sequence of cache block accesses in both implementation and, in the non automatic case, the modification and deallocation options. Because of the *fair* locking implementation, the locking semaphore block must be *deallocated in the lock operation*, and allocated again in the unlock operation (different from an unfair implementation).

Question 2

a) The proof of the instruction service time *T* formula is contained in the lecture notes, based on a limited number of cases characterizing the D-RISC firmware support.

b) The evaluation of *logical dependencies* impact on the instruction service time requires, in general, the modeling and analysis of the system as a client-server system. In this kind of systems, the response time (in our case, the delay of logical dependencies when they occur) can be expressed as the sum of *i)* the queue waiting time W, monotonically depending on the utilization factor ρ, thus on the service time of the server (inverse of EU bandwidth), and of *ii)* the server latency (EU latency, which in general is different from the EU service time). For example, in a pipelined EU, increasing the number of pipeline stages decreases the EU service time, thus decreases W, but increases the latency.

c) According to the theory of the parallelism paradigm, when a pure function has to be parallelized, *in general* the farm solution is superior to the functional partitioning because: 1) the parallelism degree is independent of the number of distinct operations (in our case: the number of distinct arithmetic operations), 2) a partitioning-based solution is prone to load unbalance. However, in our case the two solutions are perfectly equivalent in terms of service time, i.e. the functional partitioning deficiencies have no impact on the achievable service time because of the pipeline implementation of arithmetic operations. Though a certain unbalance can occur, no worker becomes a bottleneck since its service time is the pipeline service time which is the minimum achievable service time for EU. Moreover, the functional partitioning solution cost is less than the farm solution cost, where the cost is measured in terms of chip area.

Architetture Parallele e Distribuite

Domanda 1

Vedi SPA.

Domanda 2

a) Nel sistema di equazioni occorre espandere l'espressione del tempo di servizio del servente ($T_s = T_{calc}/4$, eventualmente incrementato di T_{send} se le comunicazioni non sono sovrapponibili) e della latenza del servente ($T_{calc} + 5\ T_{send}$, tenendo conto anche delle due comunicazioni con i clienti).

b) Il tempo di servizio T_s che compare nel sistema è il tempo di servizio *ideale* del servente. Il suo tempo di servizio effettivo è uguale al tempo di interarrivo, essendo garantito che $\rho < 1$. Quindi, la banda effettiva del servente è uguale alla banda effettiva del sistema: l'andamento è crescente al valore massimo asintotico $1/T_s$ per *n* tendente a infinito. L'efficienza relativa del servente è uguale a ρ, quindi ha andamento monotono crescente tendente all'asintoto 1. L'efficienza relativa del sistema decresce e tende a zero all'aumentare di *n*, in quanto all'aumentare di *n* aumenta ρ e quindi R_Q.

c) All'aumentare del grado di parallelismo *m* del servente, diminuisce il suo tempo di servizio ed aumenta la sua latenza. L'effetto combinato su R_Q è di far aumentare la banda, pur meno che linearmente a causa della latenza. Tutti i grafici del caso *b)* si trasformano in famiglie di curve con parametro *m*, tali che, all'aumentare di *m*, la banda aumenta, l'efficienza del servente diminuisce e quella del sistema aumenta.