About the Course

Dobbiamo comunicare sempre in inglese, tranne alla DISCUSSIONE dell'esame, se vogliamo.

Comunicazioni sul General Forum su Virtuale. Per vedere i materiali degli scorsi anni: http://apice.unibo.it/xwiki/bin/view/Courses/Series/Ds

Non registrano a meno che qualcuno abbia un valido motivo.

Esame = progetto finale + discussione del progetto (ed eventualmente domande di teoria)

DOBBIAMO INVENTARLO NOI, NON HO CAPITO SE DOBBIAMO PRIMA FARE UN INCONTRO COL PROF PORTANDO GLI UN PO' DI IDEE DI PROGETTO E POI SI DECIDE INSIEME COSA FARE.

singolo o gruppo in quanti ci pare ma grandezza del progetto deve scalare con il num. di partecip.

STARE ATTENTI A LIMITARE GLI OBIETTIVI DI PROGETTO PER NON FARE QUALCOSA DI TROPPO GRANDE E METTERCI TROPPO TEMPO.

CREDO CHE CI SARANNO ANCHE CONTROLLI INTERMEDI DEL PROGETTO IN MODO CHE I PROBLEMI POSSANO ESSERE SISTEMATI PRIMA DELLA DISCUSSION

SPIEGARE DOVE LE COSE VISTE A LEZIONE VENGONO IMPLEMENTATE E SFRUTTATE NEL PROGETTO, SCRIVENDO NEL REPORT.

È disponibile a darci dei link per i pdf di libri di testo ma dobbiamo chiederci via e-mail credo.

Why Distributed Systems?

Computational systems are now PERVASIVE (= are everywhere) and are at the core of most artificial systems. Thus, they affect the modelling and engineering of almost all kinds of artificial systems (sometimes software, other times hardware, or even computational units and computational systems).

We see computations that can be:

- DISTRIBUTED (controlled, triggered, they wait for something to happen)
- CONCURRENT (autonomous)

Not only chips and computational units are distributed, but also physical and virtual channels.

Computational devices are always interacting with eachother, with humans or with the physical environment and its resources.

Because artificial systems are physical, they are unpredictable: something might happen in the environment they are in. We need to consider

↓

SPATIAL DISTRIBUTION

mainly → but also...

examples of things that are spatially distributed:
- computational units
- communication channels
- data / information / knowledge
- sensors / actuators

↓

TEMPORAL DISTRIBUTION

↓

examples:
- events: they are scattered, it's not always possible to have them as a totally-ordered set.

↓

Ex: database

when computations happen autonomously we can't know if we can order events or not.

At most, we might be able to create a SUBSET of ordered events.

We need a way for systems to interact without relying on having a DIRECT INTERACTION IN SPACE AND TIME.

We need distributed systems for[Ghosh, 2014]

- geographically distributed environments
- computation speedup
- resource sharing
- fault tolerance ⚠ → we will need to design our own projects keeping fault tolerance in mind.

So, Why Distributed Systems? I

Usually not distributed is cheaper than distributed

Centralised vs. Distributed Systems[Puder et al., 2005]

Criteria	Centralized system	Distributed system
Economics	low	high
Availability	low	high
Complexity	low	high
Consistency	simple	difficult
Scalability	poor	good
Technology	homogeneous	heterogeneous
Security	high	low

low = it has many problems. Any problem makes the system unavailable.

(← easy to deal with)

very complicated. Replication, consistency...

A centralised system DOESN'T SCALE ⚠ If we have many users, it causes a bottleneck.

if you have a centralised, you have only one point of access, so you only need to protect that.

In centralised, we have a single database. In distributed, we might deal with inconsistent data.

we dont have replicated database. Maybe it's in normal form, but if we have a crash while it's not a problem in centralised, it becomes one in distributed

we need to be sure that there is some model that is able to deal with ?

In distributed if you loose control of a portion you still keep everything else.

So, Why Distributed Systems? II

The CAP Theorem

FAULT TOLERANCE = the ability to HIDE faults by using the other components that are still working.

Distributed systems can be designed so that if one component of the system fails (or becomes disconnected, or partitioned) other components can replace it, thus hiding the failures or at least reduce the perceived impact of failure.
When a system can hide most of the failures, it is said to be "HIGHLY AVAILABLE". The amount of failure that a system can sustain before failure is noticed depends on how much we are willing to accept: we might be ok with a system having a 99% fault tolerance, or we might decide that it is still too low.

We expect distributed systems to:

- be consistent (behave correctly, as expected)
- be available (= be live, work)

but they can be unreliable

The CAP Theorem is one example of a more general tradeoff between SAFETY and LIVENESS IN UNRELIABLE SYSTEMS. Essentially, according to this theorem, it is only possible to simultaneously provide any 2 out of the 3 properties of distributed applications:

① Consistency (C) = Replicated data is always consistent with each other

② Availability (A) = Data is HIGHLY AVAILABLE (= the system is able to hide most of the failures)

③ Partition tolerance (P) = The system can continue to provide services to its users even when the NETWORK partitions

The 1st formulation (by Eric Brewer) states:
A distributed database potentially features 3 desirable properties:
① C
② A
③ P

According to the theorem, any shared-data distributed system can only have up to 2/3 properties.

so... we can either:

- no tolerance to network partition (~~P~~) but we have C and A.
- No ~~C~~ but then we have P and A.
- No ~~A~~ but we have C and P.

We should notice that while C and A range over a spectrum of options, P does not, and can be considered as an on/off feature. Also, we can't really do without P, so this scenario is not really acceptable. ⇒ we usually have to choose between A and C.

example: LOCATION-BASED GAMES

They use information about the position of users to evolve and progress the gameplay. Because there are millions of players worldwide, we need P, considering mobile devices are unstable in their network connectivity, and players might move in and out of network coverage.
C is needed to keep the players into the game (ex: in-game purchases)
So they usually forfeit A