A practical view of the datacenter

Miquel Garcies
University of the Balearic Islands

miquel.garcies@uib.cat

I'm sorry but my english is bad and I need to read, in order to be faster. I'm not a researcher so my presentation will be very different from yours.

My name is Miquel Garcies and I work at the IT department of the university.

This presentation covers my point of view of the evolution, last 20 years, of the IT infrastructure needed to provide resources to run the organization's applications. Ours and many others.

Our evolution of the datacenter

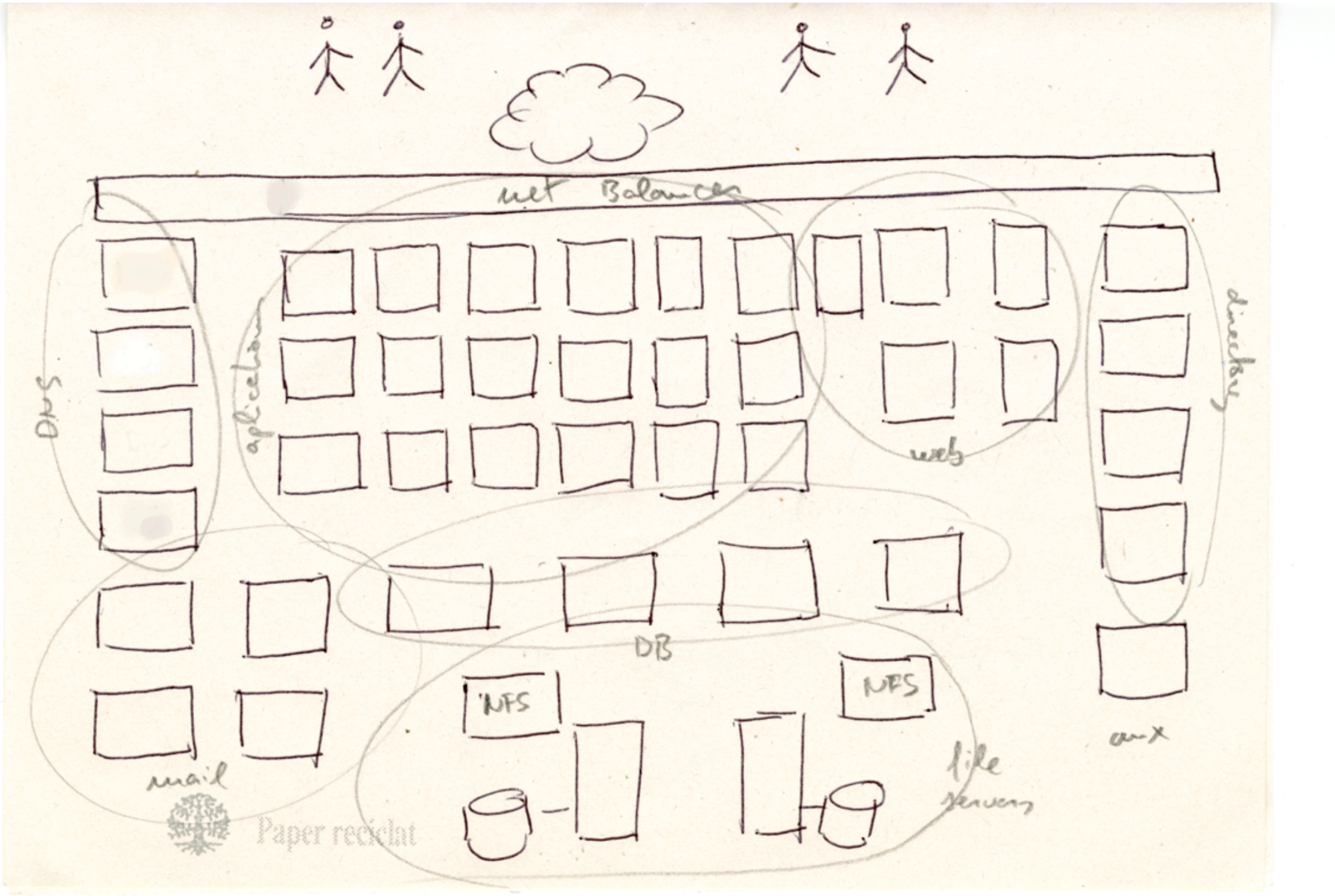
- Previous model, mainframe centered approach
  - Expensive hardware, well designed operating systems, old network architectures with robust network layer.
  - Traditional development tools: forms, compilers, RDBMS.
  - Users with terminals in the campus ran all applications.
  - Few processors, few hundreds of MB of RAM.
  - Power: 2 KW + n * 70W.
- Currently, locally distributed model
  - Cheap hardware: PCs, consumer operating systems (linux, windows and VMware), small virtualized boxes providing part of the service, TCP/IP.
  - HTML, XML, Javascript, JAVA, OAS, Tomcat, Oracle RAC.
  - Users at home with PCs run beautiful applications.
  - Hundreds of processors, TB of RAM,
  - Power: 5 KW + n * 200 W.

(READ SLIDES)

Important questions:

One of the new processors are much faster than all the older ones.
We only have some more aplications than then. Basically we are still covering the needs of the enterprise administrative processes.
A key question is that then new infrastructure consumes more power than the older (and the adquisition price maybe is not much different).

Logical view



Part of the bottom layer are real machines, all other are virtualized.
The network balancer covers all the mesh of communication between boxes.

Problems detected

- Now we have very fast processors, but the memory bandwidth didn't scaled at the same rate. They rely on the cache. This applies to HDs too (SDD are helping).
- Consumer operating systems service time degenerates rapidly as as number of processes scale up (even a small number). Open source often lacks quality (even some commercial OSs). It is needed to update all kernel scheduling algoritms and to reduce bugs.
- Virtualization is a useful tool in some cases but imposes a heavy penalty on IO and CPU overhead. To two virtualized systems comunicate, then execute all communications stack (can we bypass the socket interface with mailboxes?).
- Intra-PCs communication adds latency and increases response time. We generate a lot of traffic and need fast switches. And the PCs are nearby.
- JAVA is good programming language in an interpreted environment (with some live compilation features). It was designed to be run for clients in the network but nowadays is used in the servers!. The garbage collector algorithm has a problem with virtual memory management (due to memory access patterns).
- TCP/IP hasn't any fault tolerance capability. Is really IP a network protocol? We need to introduce the host/cluster/application concept in the communications stack. TCP doesn't work well in current worldwide heterogeneus network.
- And because of the lower reliability of the hardware, we need two CPDs. All costs are quasi duplicated.
- High level application developers usually don't think in performance (and when the applications have bad response time, someone buys more hardware).

Now I will show you some conflicting aspects with the performance of the system we have detected.

(READ SLIDES)

The actual software don't take real advantage of thousandfold increase in performance in 20 years of microelectronics evolution.

I think all the industry needs to redefine the way it develops products. Efforts have to be done finding new paradigms in software development.

You, the researchers now have more questions to think about.

Thank you