

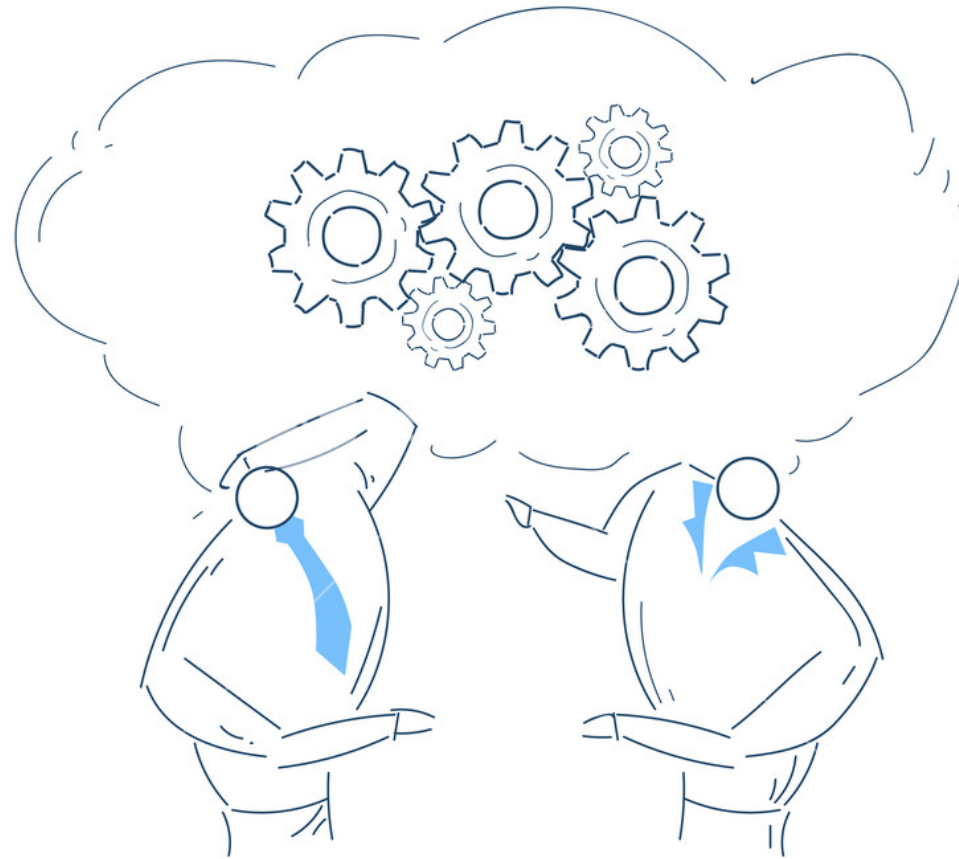


Introdução/Arquitecturas

Computação Distribuida 2018/19

dgomes@ua.pt

Processos





Processos vs Threads

- Criados pelo SO
 - Podem ter múltiplas threads
 - Maior overhead – levam mais tempo a abrir e fechar
 - Partilha de informação entre processos é lenta pois não partilham memória
- São mini-processos (dentro de um processo)
 - Partilham memória pelo que leem e escrevem eficientemente as mesma variaveis

Em Python não trazem beneficio devido ao GIL (Global Interpreter Lock)

Python examples

```
import threading

def count(prefix, n):
    for x in range(n):
        print(prefix, x)

if __name__ == "__main__":
    thread1 = threading.Thread(target=count,
                                args=("a", 100))
    thread2 = threading.Thread(target=count,
                                args=("b", 100))

    thread1.start()
    thread2.start()

    thread1.join()
    thread2.join()
```

```
import multiprocessing

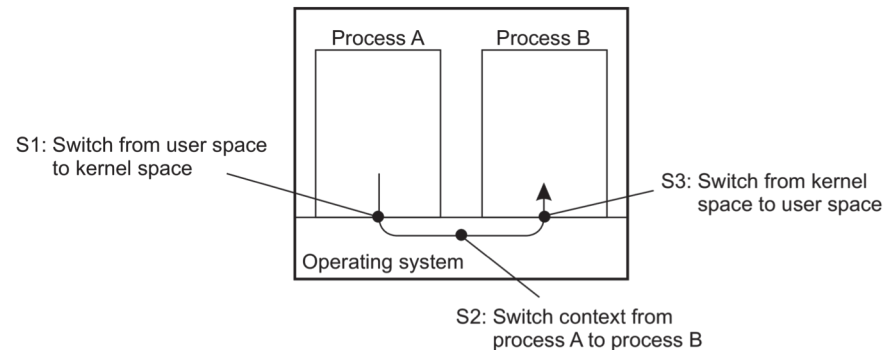
def count(prefix, n):
    for x in range(n):
        print(prefix, x)

if __name__ == "__main__":
    p1 = multiprocessing.Process(target=count,
                                  args=("a", 1000))
    p2 = multiprocessing.Process(target=count,
                                  args=("b", 1000))

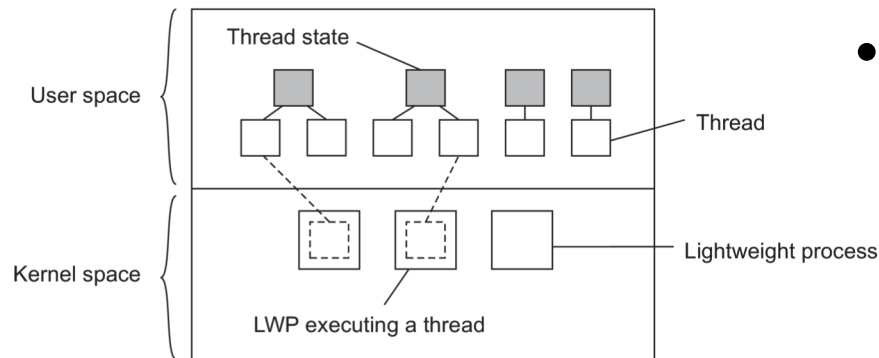
    p1.start()
    p2.start()

    p1.join()
    p2.join()
```

Threads em sistemas não distribuídos

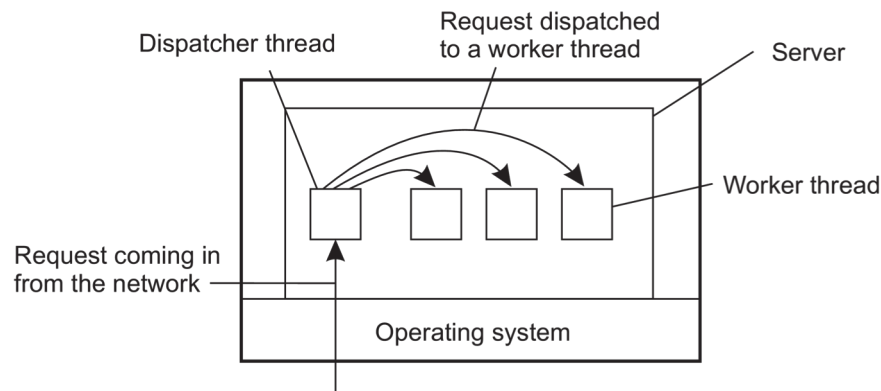


- Inter-process Communication (IPC)



- Implementação
 - Lightweight Processes (LWP)
 - Scheduler activations

Threads em sistemas distribuídos



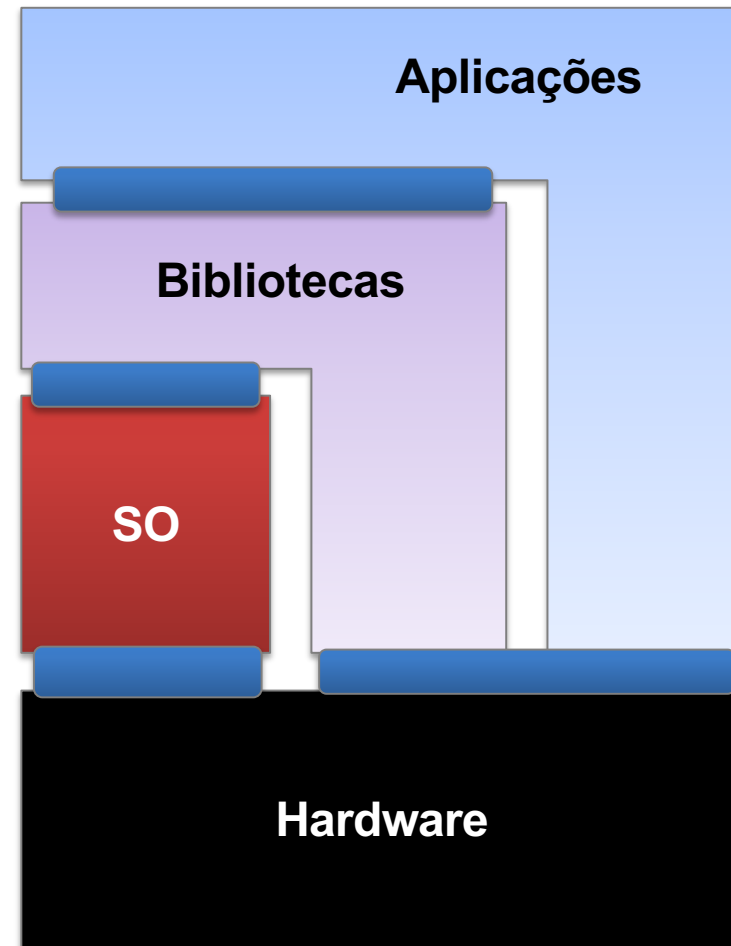
- Servidor *multithread* organizado em modelo *dispatcher/worker*

Modelo	Características
Threads	Paralelismo, Chamadas ao sistema bloqueiam
Processo Singe-thread	Sem paralelismo, Chamadas ao sistema bloqueiam
Maquina de estados finitos	Paralelismo, Chamadas ao sistema não bloqueiam

Virtualização

Virtualização pode ser conseguida a vários níveis:

- Chamadas bibliotecas
 - Mesmo SO/HW, bibliotecas individuais
- Chamadas ao sistema
 - Mesmo HW, Visão do SO individual
- Chamadas ao Hardware
 - Hardware individual





Virtualização

- Conceito muito abstracto...
 - Memória Virtual, Filesystem Virtual, Redes Virtuais (VPN's)
- Virtualização permite que um **único computador** possa desempenhar o papel de vários computadores, através da **partilha de recursos de hardware** por vários ambientes.
- Virtualização é a técnica que permite correr vários sistemas distintos no mesmo hardware, de forma completamente separada.

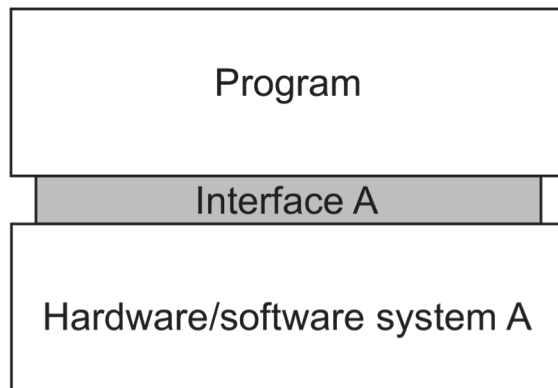
Virtualização

- Citação mais popular:
 - “*Formal Requirements for Virtualizable Third Generation Architectures*” **Gerald J. and Robert P. Goldberg**,
Communications of the ACM, Volume 17, Issue 7, Julho 1974
- “VMM satisfies efficiency, resource control, equivalence”
- “Theorem – For any conventional third generation machine, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.”

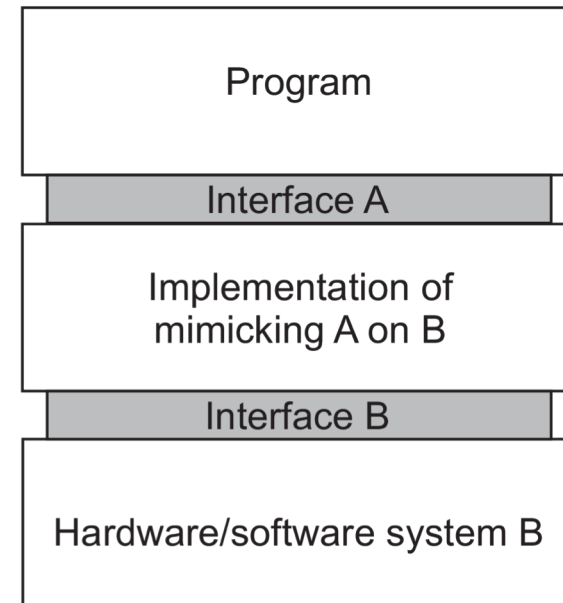
Virtualização – História

- Mainframe IBM S/360 permite o timeshare de aplicações através do particionamento de recursos (1960/70)
 - Introdução do conceito de LPAR (Logical resource Partitioning) e mais tarde DLPAR (LPAR dinamico)
 - Multiplos SO
- Bochs disponibiliza ambientes emuladores (1990)
 - Principalmente para debugging
 - Ambiente hardware complete por emulação
- VMWare (1998)
 - Fundado em Stanford
 - Parte da EMC/Dell
 - Disponibiliza hypervisors que permitem isolamento complete
- XEN (2003)
 - Fundado em Cambridge
 - Propoe para-virtualização (host assisted)

Virtualização



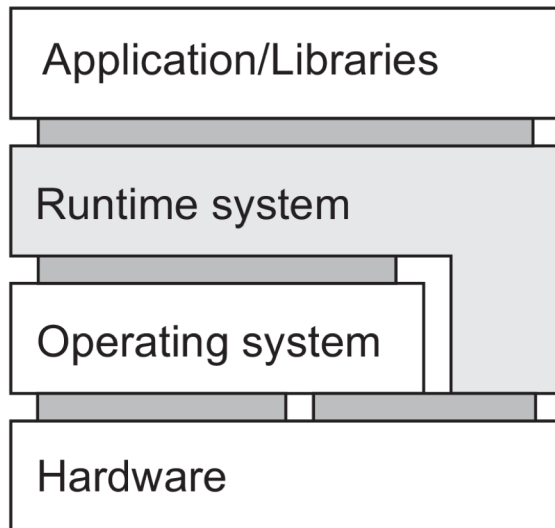
Programa, Interface e Sistema



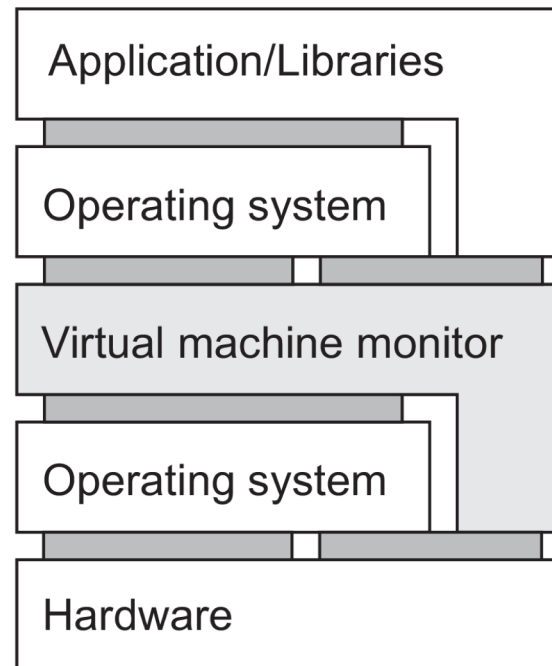
Sistema virtualizado A em cima do sistema B

Arquitecturas de VM's

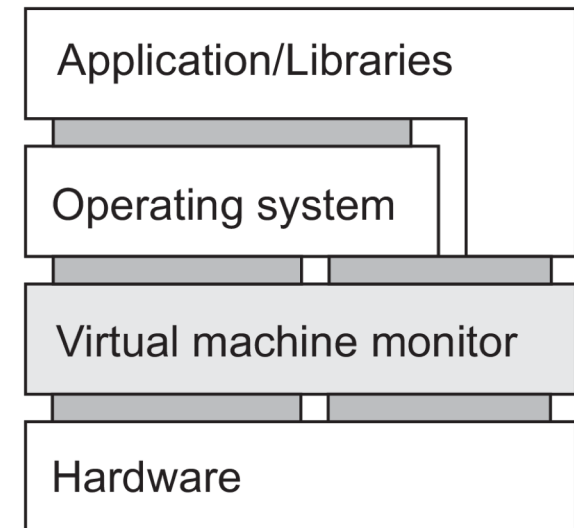
Virtual Machine
Processo
(ex. JVM)



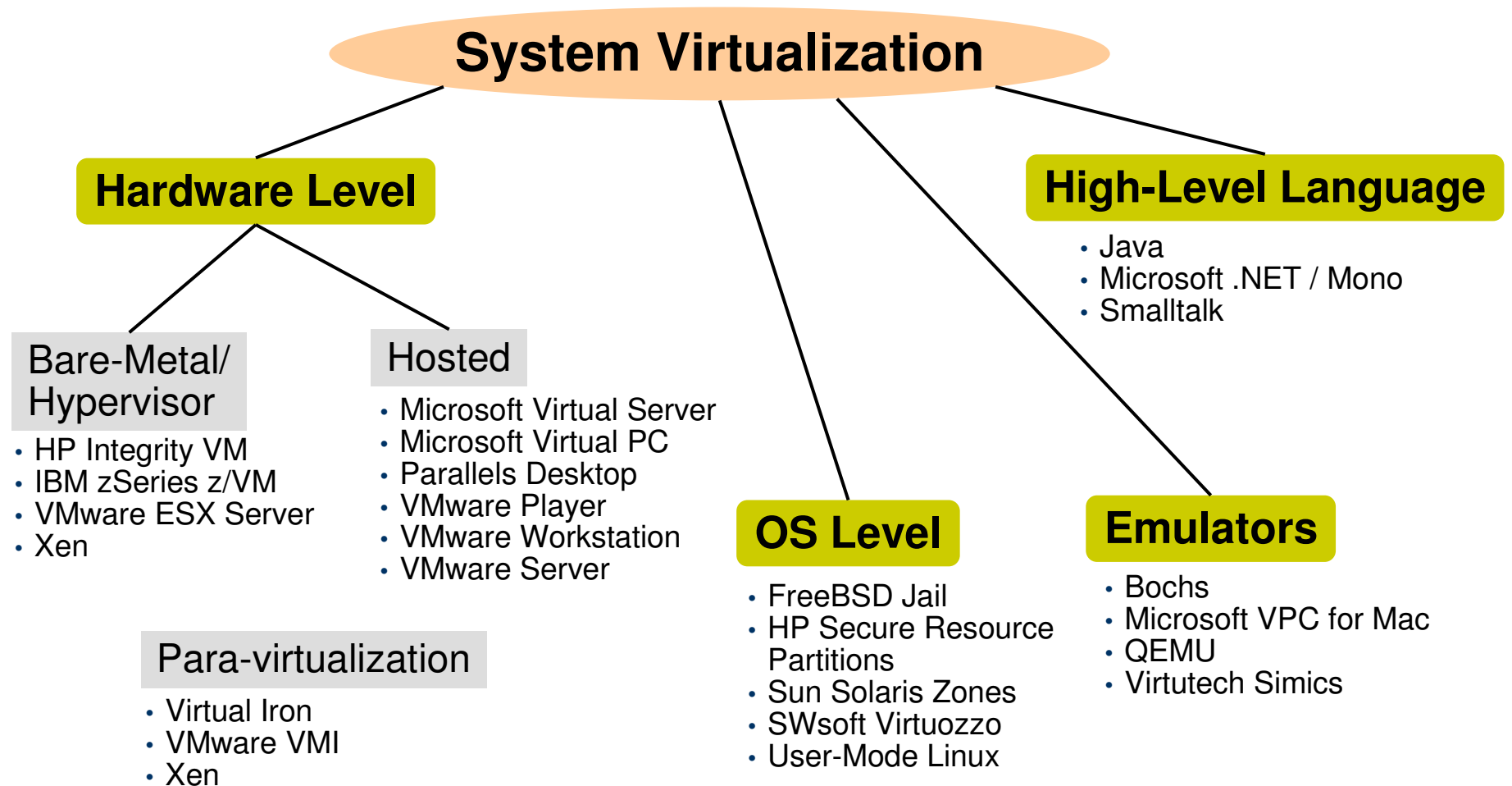
Virtual Machine
Nativa
(ex. VirtualBox)



Virtual Machine
Hosted
(ex. XEN)

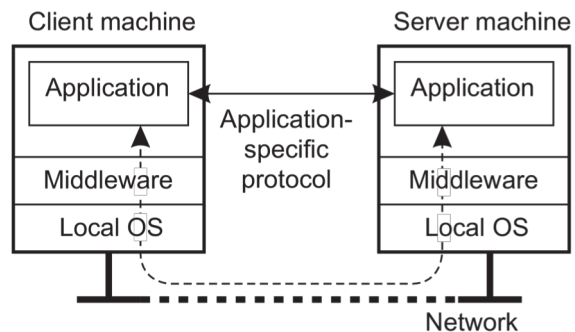


Eco-sistema

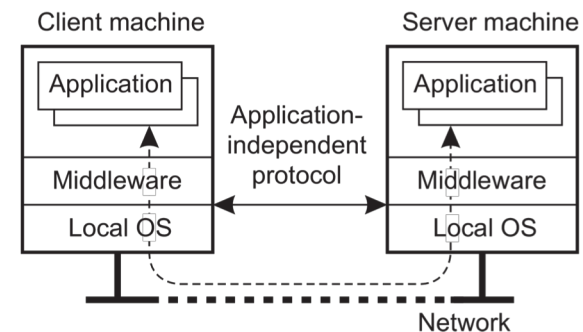


Cientes

Uma aplicação em rede com o seu próprio protocolo

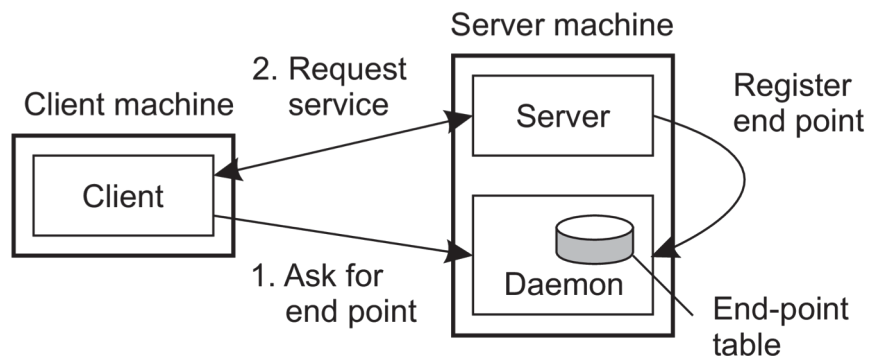


Uma solução genérica que permite o acesso remoto a aplicações

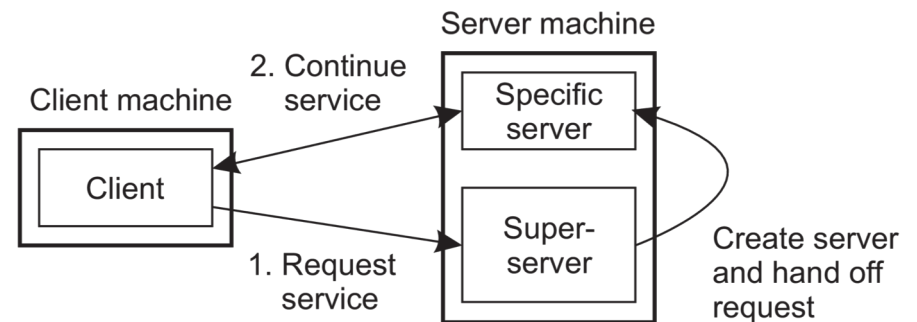


Servidores

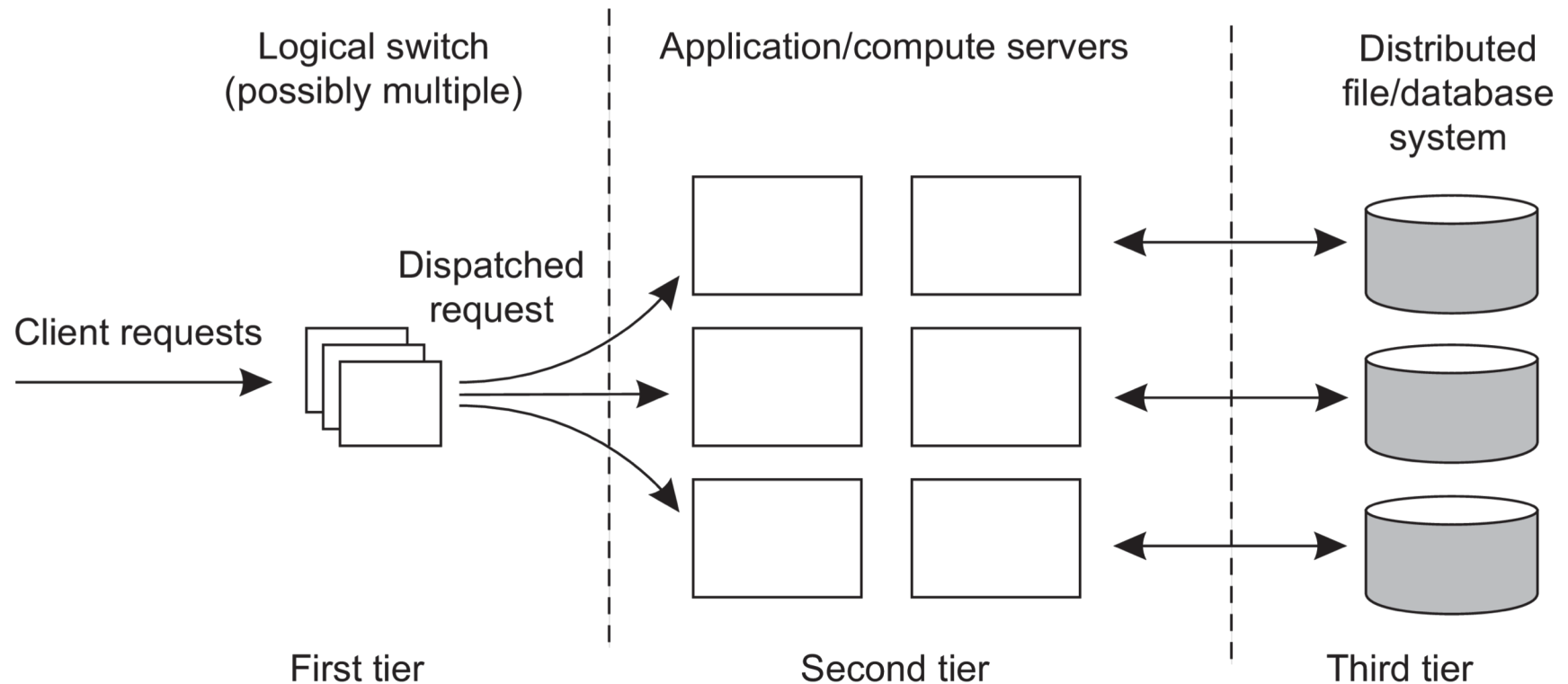
Ligação Cliente-Servidor através de um daemon.



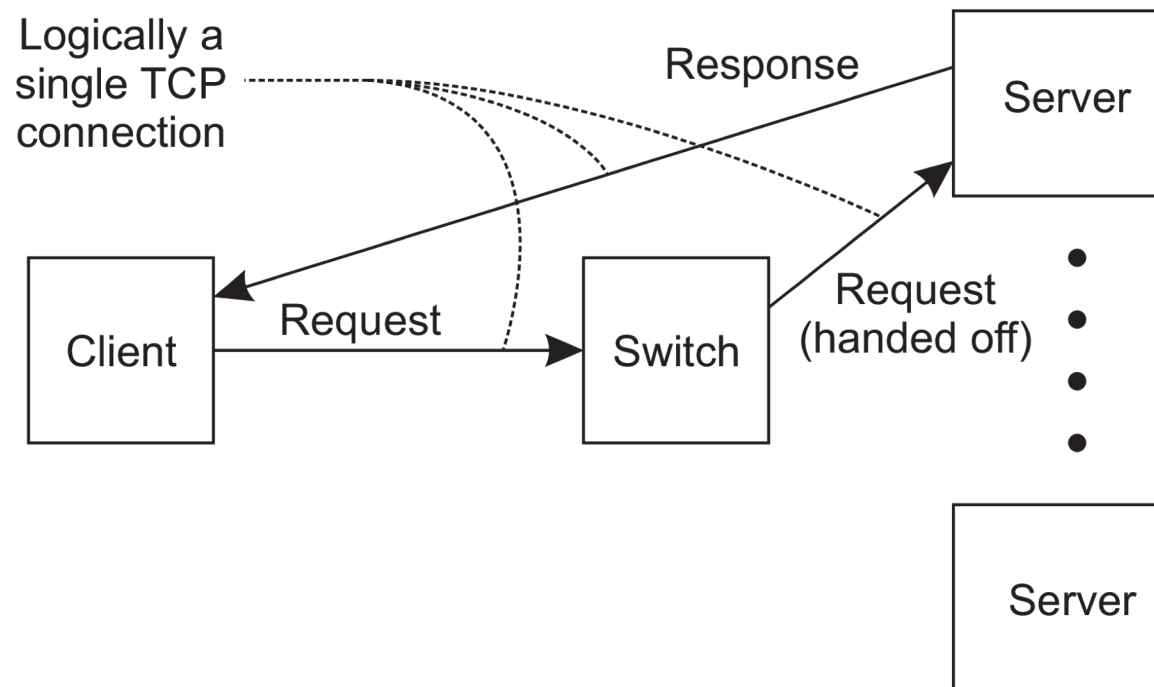
Ligação Cliente-Servidor através de um super-servidor.



Clusters de Servidores



TCP Handoff



Migração de Código

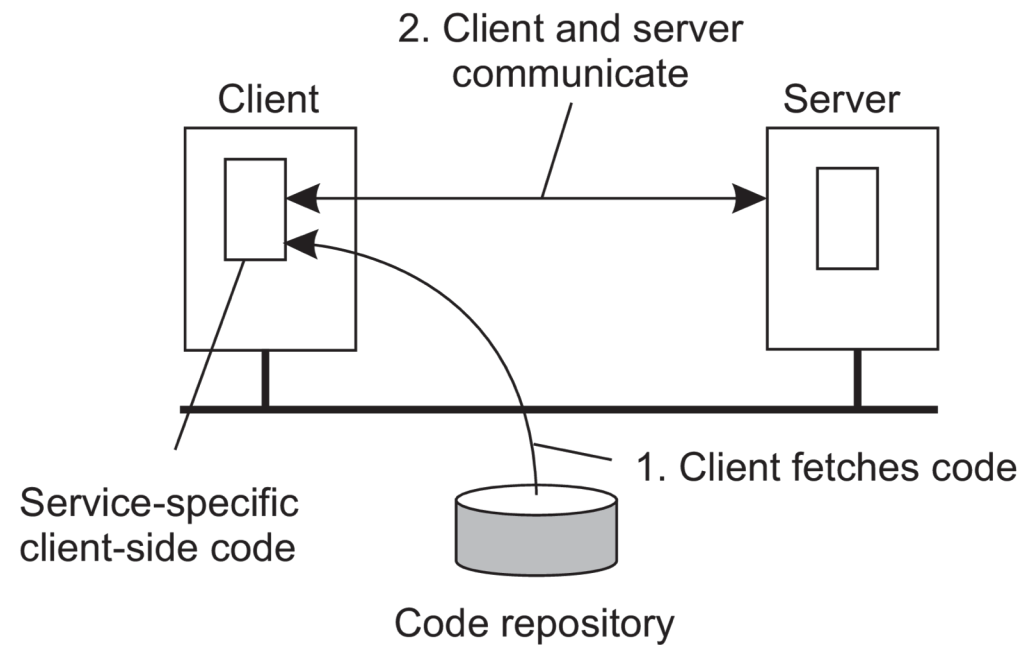
Modelos:

Mobilidade fraca

Mobilidade forte

Iniciado pelo emissor

Iniciado pelo receptor



Migração e recursos locais

- **Ligação Processo-Recurso**
 - Ligação por identificador
exemplo URL
 - Ligação por valor
exemplo C/Java
 - Ligação por tipo
exemplo referencias a tipos: monitor, impressora, etc
- **Ligação Recurso-Máquina**
 - Não ligado
exemplo ficheiros
 - Apertado
exemplo DB's
 - Fixo
exemplo dispositivos fisicos, cartões, GPU's