

Código Móvel



histórico

- migração de processos em sistemas operacionais distribuídos
 - balanceamento de carga de CPU
 - otimização do uso da rede
 - ligação com transparência
- dificuldade
 - previsão do comportamento do processo

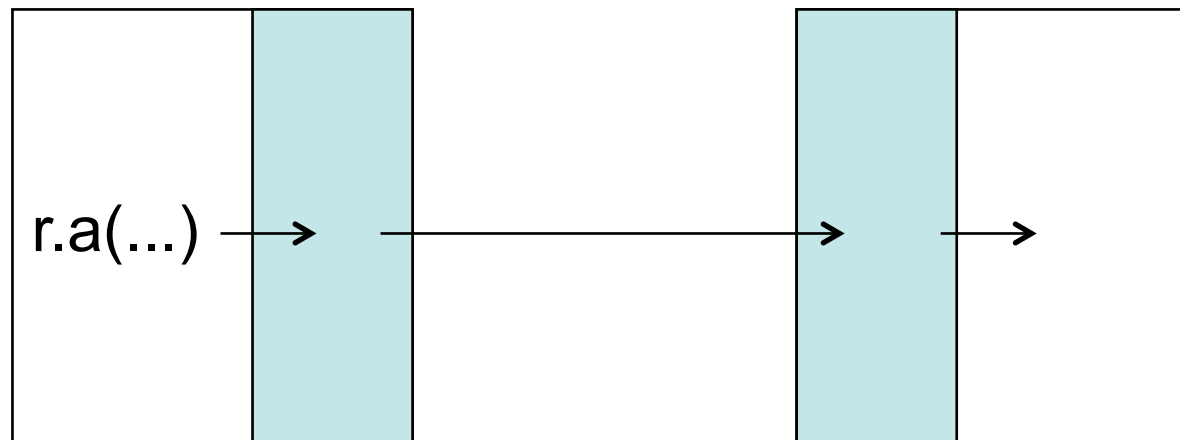


motivações mais recentes

- atualizações dinâmicas
 - “injeção de código” em programas/componentes em execução
 - adaptações a condições de execução
- customizações
 - de cliente ao servidor
 - de servidor ao cliente
- computação oportunista
- tolerância a falhas
- tratamento de conexão e desconexão



mobilidade em cliente-servidor



- redefinição dinâmica de stubs cliente e servidor
- customização para cliente/servidor específico
- relação com cliente thin/fat



classificação

- remote evaluation
- código sob demanda
- agentes móveis
 - não necessariamente inteligentes...
 - interações entre entidades A e B
 - ponto de vista cliente-servidor

A. Fugetta, G.P. Picco, G. Vigna. Understanding Code Mobility. IEEE Trans. Software Eng, 24(5), 1998.

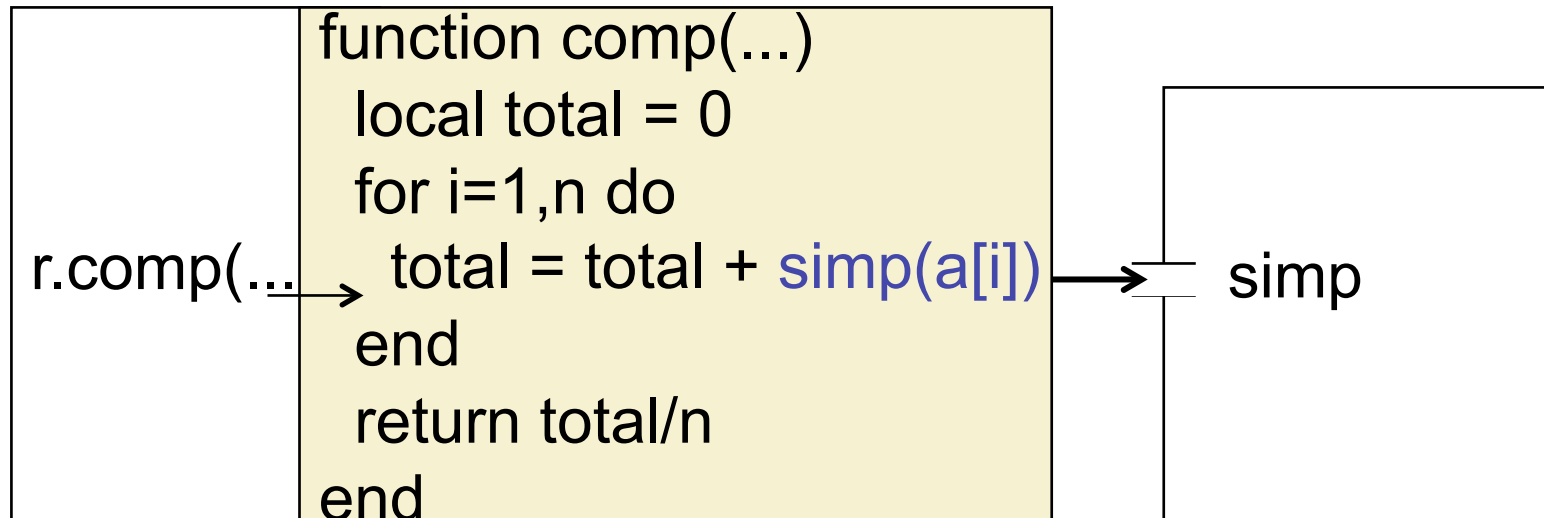


avaliação remota

- A envia código para B executar
 - filtragem de dados a serem enviados para A
 - bases de dados
 - serviços de eventos
 - ...



flexibilidade em servidores



- servidores podem oferecer API básica a partir da qual funções mais complexas podem ser construídas



servidor que aceita código

r.comp(...)

upload

```
function comp(...)
  local total = 0
  for i=1,n do
    total = total + simp(a[i])
  end
  return total/n
end
```

- novo código pode ser provisório ou permanente
 - modificações na interface
- remote evaluation
- relação com eficiência e tolerância a falhas

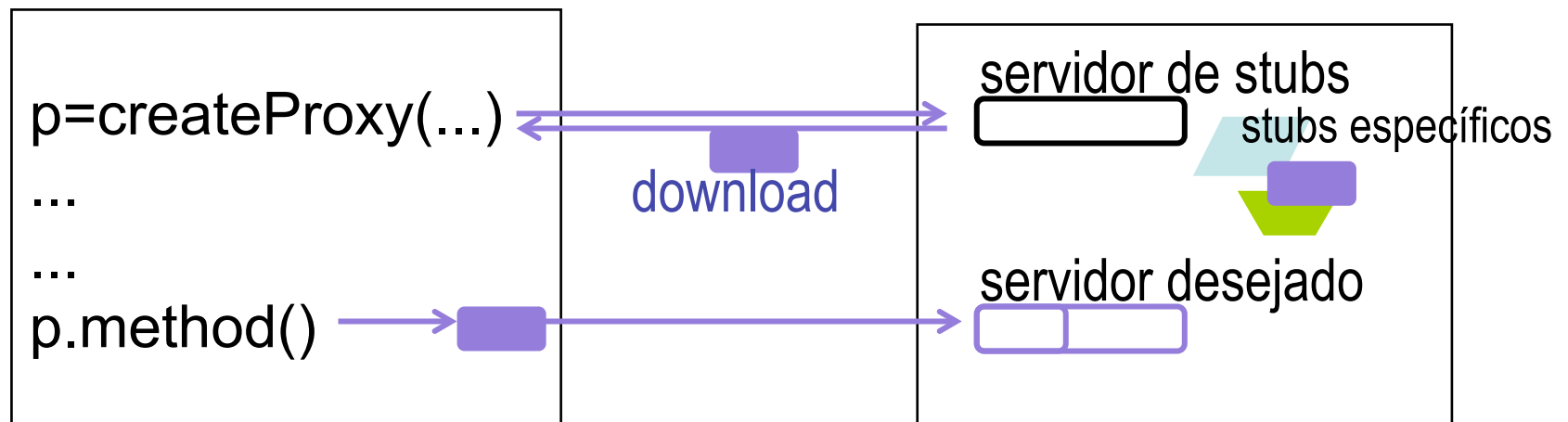


Código sob demanda

- A requisita a B, explicitamente, determinado código
 - código de comunicação com B



adaptação de clientes



- clientes baixam código apropriado para comunicação com esse servidor



adaptação de clientes

- uso de otimizações como cache
- proxies inteligentes
 - mudança no comportamento do cliente conforme estado do sistema



agentes móveis

- A migra para B levando o código que deseja executar em B



agentes móveis

- programa se transfere com estado completo
 - tipicamente primitiva "move"
- comparação com remote evaluation
 - flexibilidade maior: funcionalidade arbitrária
- redução de uso da banda passante
- possibilidade de adaptação a estado
- boa convivência com desconexões



↪ tecnologia fez menos sucesso que esperado?



paradigmas

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
<i>Client-Server</i>	A	know-how resource B	A	know-how resource B
<i>Remote Evaluation</i>	know-how A	resource B	A	<i>know-how</i> resource B
<i>Code on Demand</i>	resource A	know-how B	resource <i>know-how</i> A	B
<i>Mobile Agent</i>	know-how A	resource	—	<i>know-how</i> resource A



mobilidade de código e de estado

- mobilidade forte
 - transferência de código e do estado da execução
 - pilha
- mobilidade fraca
 - transferência de código
 - possivelmente alguns dados para inicialização
 - como esse código é incorporado no destino?
 - nova unidade de execução X incorporação



▪ distinção (forteXfraca) nem sempre óbvia



implementação de mobilidade

- código
 - relativamente simples
- estado
 - complexa, especialmente em máquinas heterogêneas
 - problema da transferência de estado externo
 - arquivos, sockets, etc



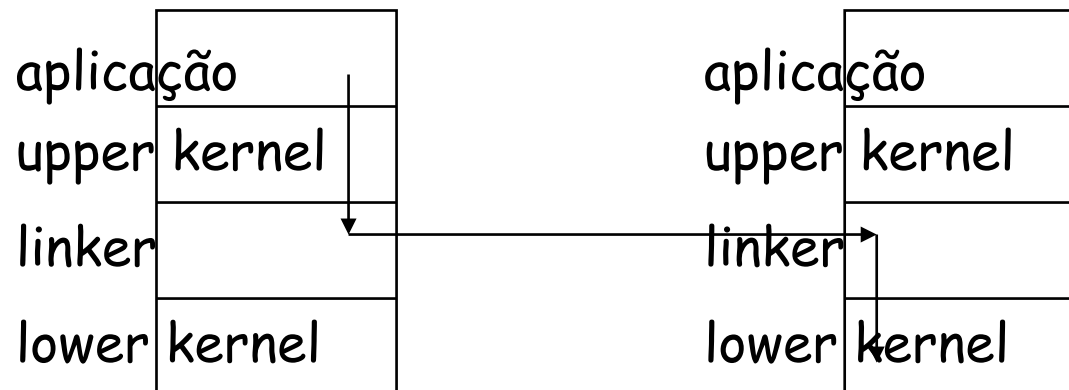
migração de processos

- assunto bastante estudado nos anos 80
- ressurgimento de interesse
 - melhores redes
 - grades e computação oportunista



MOSIX

- kernel implementa RPC
- chamadas a sistema transferidas para máquina apropriada



MOSIX: migração

- máquinas sobrecarregadas decidem destino de processos escolhidos
- carga medida por prontos e memória livre
- escolha de migração leva em conta
 - perfil (passado) do processo
 - tamanho -> tempo de migração
 - tempo de residência
 - i/o local e remoto
- objetivos:
 - maior throughput e menor tempo de resposta



migração em sistemas heterogêneos

- como transferir estado entre duas máquinas com arquiteturas diferentes?
- problemas em diversos níveis
 - representação de dados básicos
 - ...
 - estrutura da pilha
- maioria das linguagens não mantém informação sobre dados e programa em tempo de execução



captura de estado

- problema existe também na área de **persistência**
 - relação com tolerância a falhas
- importância de suporte de linguagem
 - exemplo: linguagens com call/cc (call with current continuation)
 - sobre tais primitivas pode-se construir sistemas com diferentes facilidades de migração e persistência
- há também várias implementações de sistemas monolíticos
 - mecanismos X políticas



segurança

- controle de acesso no ambiente destino
 - sandbox
 - máquinas virtuais
 - populares no mundo de grades
 - mapeamento de usuário
 - código executa como se fosse de algum usuário local
- código assinado



Bibliografia

- A. Fugetta, G.P. Picco, G. Vigna. Understanding Code Mobility. IEEE Trans. Software Eng, 24(5), 1998.
- A. Milanés, N. Rodriguez, B. Schulze. State of the art in heterogeneous strong migration of computations. Concurrency and Computation: Practice and Experience. 20(13), 2008.

