SISTEMAS DISTRIBUÍDOS

João Leitão, Sérgio Duarte, Pedro Camponês

(baseado nos slides de Nuno Preguiça)

OBJETIVOS DA CADEIRA

Esta é uma unidade curricular (UC) obrigatória, de introdução aos sistemas distribuídos. Esta UC introduz os **modelos**, **métodos** e **técnicas fundamentais** para o desenvolvimento de sistemas distribuídos seguros. Pretende-se que os alunos compreendam:

- Características essenciais
- Modelos e arquiteturas dos sistemas
- Técnicas de conceção

Na componente prática, os alunos colocam em prática as técnicas abordadas na disciplina desenvolvendo um sistema distribuído seguro e tolerante a falhas, com recurso a tecnologias *standard*, incluindo REST, gRPC e OAuth.

PROGRAMA DAS AULAS TEÓRICAS

1. Introdução (cap. 1)
2. Arquiteturas e modelos (cap. 2)
3. Sistemas de comunicação direta (cap. 4.1-4.3, 6.4)
4. Invocação remota (cap.5, 4.3)
5. Invocação remota na Internet (cap. 9)
6. Avaliação de sistemas distribuídos
7. Tempo e ordenação de eventos (cap. 6.1-6.3, 15.4)
8. Introdução à replicação (cap. 12.1-12.4, 15.3, 18.4)
9. Sistemas de comunicação indireta (cap. 6.1-6.3, 15.4)
10. Segurança (cap. 11.1-11.4,11.6)
11. Sistemas de Nomes (cap. 13)

PROGRAMA DAS AULAS TEÓRICAS

1. Introdução (cap. 1)
2. Arquiteturas e modelos (cap. 2)
3. Sistemas de comunicação direta (cap. 4.1-4.3, 6.4)
4. Invocação remota (cap.5, 4.3)
5. Invocação remota na Internet (cap. 9)
6. Avaliação de sistemas distribuídos
7. Tempo e
8. Introduç
9. Sistemas
10. Segurança
11. Sistemas

Tópicos mais avançados vão ser cobertos em disciplinas de mestrado:

Cloud Computing
Distributed Algorithms
Reliable Systems

PROGRAMA DAS AULAS PRÁTICAS

Pressuposto: alunos conhecem primitivas de comunicação TCP/IP (java.net)

Web services REST

Invocação remota gRPC

- Tolerância a falhas nestes sistemas

REST + gRPC sobre TLS

REST + Oauth

Soluções de replicação

PLANEAMENTO

Datas a confirmar !

Datas	Teórica-2ªfeira	Teórica-3ªfeira	Práticas	Data Importantes	Trabalho: check list
10-11 March	Apresentação & intro : 1	Introdução : 1	Docker + descoberta		
17-18 March	Comunicação direta: 3	Inv remota: 4	Web services REST	**Enunciado P1**	
24-25 March	Inv remota: 4	Inv remota: 4	DB + falhas		
31 March-1 April	Web services: 5	Web services: 5	Web services gRPC		Parte REST concluída
7-8 April	Arquiteturas: 2	Arquiteturas: 2	Suporte trabalho		
14-15 April	Arquiteturas: 2	Arquiteturas: 2	Suporte trabalho		Parte gRPC concluída
Pascoa / 22 April	Páscoa	Avaliação: 6	Suporte trabalho	**Trabalho - 26 abr**	
28-29 April	Aula de dúvidas	SPARE		**Teste 3 May (9h30/11h30) & enunciado P2**	
5-6 May	Tempo: 7	Replicação: 8	REST+Oauth		Ligação a servidor externo
12-13 May	Replicação: 8	Replicação: 8	REST / SOAP + TLS		HTTPS
19-20 May	Comunicação indireta: 9	Comunicação indireta: 9	KAFKA + replicaçao		
26-27 May	Segurança: 10	Segurança: 10	Suporte trabalho		Replicação concluída
2-3 June	Segurança: 10	Segurança: 10	Suporte trabalho	**Trabalho - 04 June**	
	Segundo Teste			**Teste 9 June**	

FUNCIONAMENTO DAS AULAS

Aulas teóricas

- Slides no CLIP.
  (documentação -> multimédia)

Aulas práticas

- Materiais no CLIP.
  (Slides: documentação -> multimédia)
  (Código base para aulas: documentação -> problemas)

OUTROS RECURSO

CLIP

- Informação académica, regras, etc.

Discord

- Usar para dúvidas gerais.
- Podem usar para formar grupos.
- Tem uma área para dicussões gerais)

  (O link para se juntarem ao Discord foi enviado por mail)

SOFTWARE

- Java
- Maven
- Docker

BIBLIOGRAFIA

Principal

- M. van Steen and A.S. Tanenbaum, Distributed Systems, 4th ed., distributed-systems.net, 2023.
  Get a PDF free here:
  https://www.distributed-systems.net/index.php/books/ds4/
- George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, "Distributed Systems - Concepts and Design,"
  Addison-Wesley, 5th Edition, 2011

AVALIAÇÃO – PARTE PRÁTICA

Frequência da parte prática. Um aluno tem frequência se a nota de frequência for igual ou superior a 9,0 valores

- Nota de frequência base = 50% * trab-fase 1 + 50% * trab-fase 2

Haverá uma discussão obrigatória para alunos selecionados para esse efeito, presencial e individual. A discussão consiste em fazer alterações ao código para que o novo programa resolva uma variante do problema original, definida no enunciado da discussão. As notas das discussões e os respetivos critérios são os seguintes:

- 20: as alterações estão globalmente certa;
- 16: as alterações estão confusas ou muito incompleta, mas o caminho poderia ser aquele;
- 12: as alterações não estão certas, havendo "algumas coisas bem e outras muito mal";
- 4: não foi feita qualquer alteração ou as alterações feitas são ínfimas ou as alterações feitas não fazem sentido.

A nota de frequência será:

- Nota de frequência = min(nota de frequência base, nota de discussão)

Os trabalhos práticos são realizados em grupos de 2 alunos (de preferência do mesmo turno prático).

AVALIAÇÃO – TRABALHOS PRÁTICOS (2)

Para o trabalho prático, será definido no enunciado em cada uma das duas fases:

- As funcionalidades mínimas que o trabalho deve implementar.
- As funcionalidades opcionais e a sua valorização na nota do trabalho.

Será fornecida uma bateria de testes para o trabalho.

Os alunos que, sem justificação, não compareçam a uma discussão para a qual foram convocados não têm frequência.

Utilização de ferramentas de IA generativo: é permitido; vamos perguntar se usaram e como; o teste serve para validar se percebem os trabalhos que entregaram.

Nota: Os testes teóricos/exame vão ter perguntas que cobrem também a parte prática.

AVALIAÇÃO

Componentes da avaliação: teste 1 (30%), teste 2 (30%), freq (40%)

- Nota: todas as notas intermédias são aproximadas às décimas.
- Média(teste 1, teste 2) >= 9,5

Nota com exame: exame (60%), freq (40%)

- exame >= 9,0

Melhorias de nota ou alunos com frequência positiva.

- A nota de frequência obtida é válida e será usada no cálculo da nota final, substituindo a nota dos trabalhos.

Os testes e exames são realizados sem consulta.

Equipa Docente e Horários de Atendimento

- João Leitão (Teóricas & 1 Turno Prático)
  - Segunda-Feira 10am-11am (P2.41 ou P2.7)
  - Terça-Feira 5pm-6pm (P2.41 ou P2.7)
  - Quinta-Feira 6pm-7pm (P2.41 ou P2.7)
- Sérgio Duarte (4 Turnos Práticos)
  - Terça-Feira 10am-11am (P3.1)
  - Sexta-Feira 6pm-8pm (zoom, link no clip)
- Pedro Componês (2 Turnos Práticos)
  - Terça-Feira 6pm-7pm (Lab 121-A)
  - Quinta-Feira 6pm-8pm (zoom: TBA)

Dúvidas sobre Funcionamento de SD

SISTEMAS DISTRIBUÍDOS

Aula 1

Introdução aos Sistemas Distribuídos

(baseado nos slides de Nuno Preguiça)

NOTA PRÉVIA

A apresentação utiliza algumas das figuras do livro de base do curso

G. Coulouris, J. Dollimore and T. Kindberg,
Distributed Systems - Concepts and Design,
Addison-Wesley, 5th Edition, 2011

ORGANIZAÇÃO DO CAPÍTULO

Definição, exemplos

Características essenciais dos sistemas distribuídos.

Desafios Principais:

- Heterogeneidade
- Abertura
- Segurança
- Escala
- Falhas
- Concorrência
- Transparência

SISTEMAS DISTRIBUÍDOS ?

Exemplos?

- Serviços web – redes sociais, e-commerce, jogo multi-utilizador, etc.
- Email
- Multibanco
- Etc.

O que é essencial num sistema distribuído?

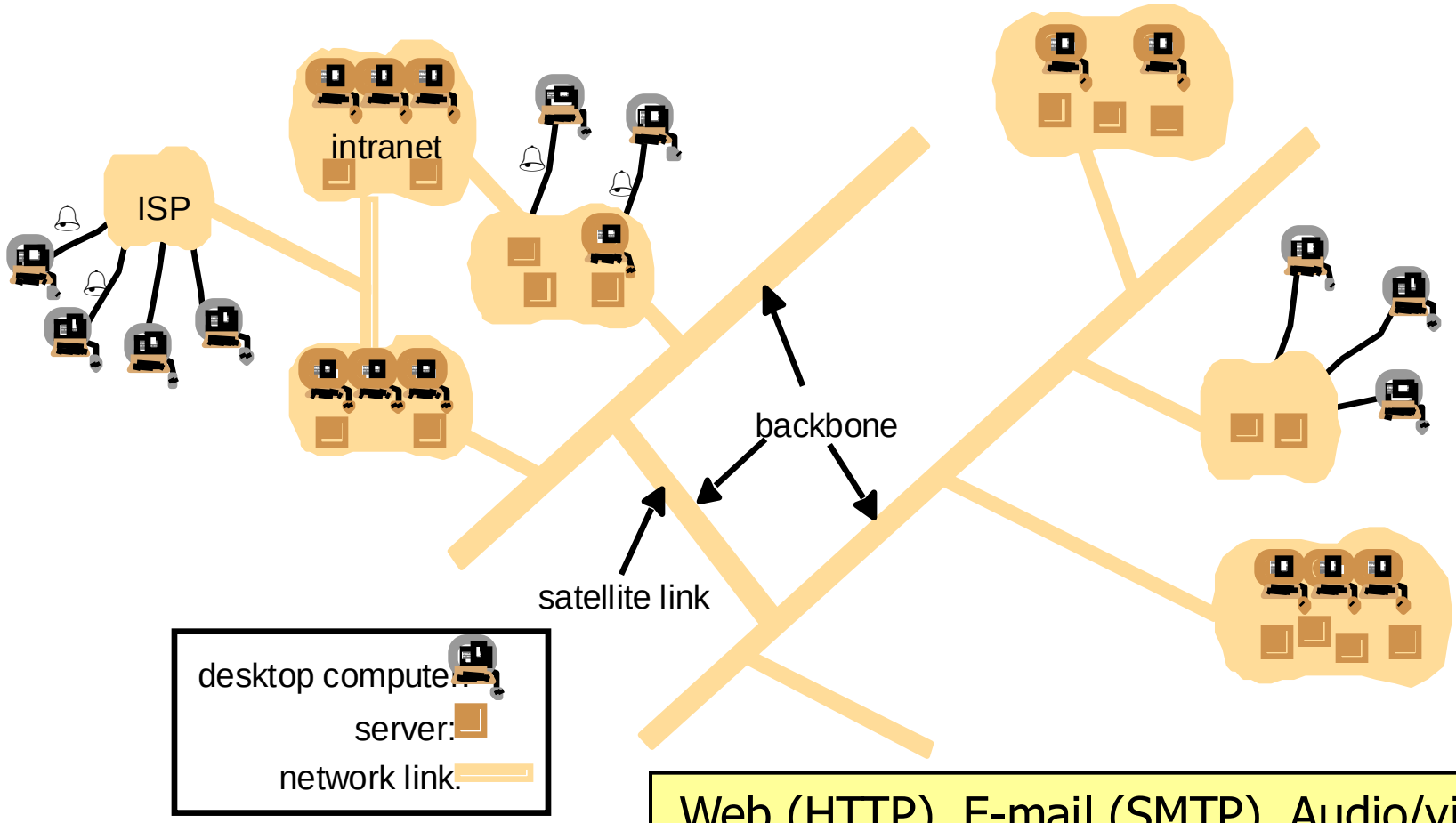
- Conjunto de nós / máquinas
- Utilização duma rede de comunicações para troca de mensagens

O QUE É UM SISTEMA DISTRIBUÍDO ?

Um sistema distribuído é um conjunto de componentes hardware e software interligados através de uma infra-estrutura de comunicações, que cooperam e se coordenam entre si apenas pela troca de mensagens, para execução de aplicações distribuídas (i.e., cooperação na realização de uma tarefa).

Nesta cadeira, não estamos interessados nos sistemas que cooperam e se coordenam pela partilha de memória física comum ou de programas que executam em GPUs (essas temáticas são abordadas inicialmente nas disciplinas de mestrado Sistemas Concorrentes e Paralelismo)

EXEMPLO: SERVIÇOS NA INTERNET

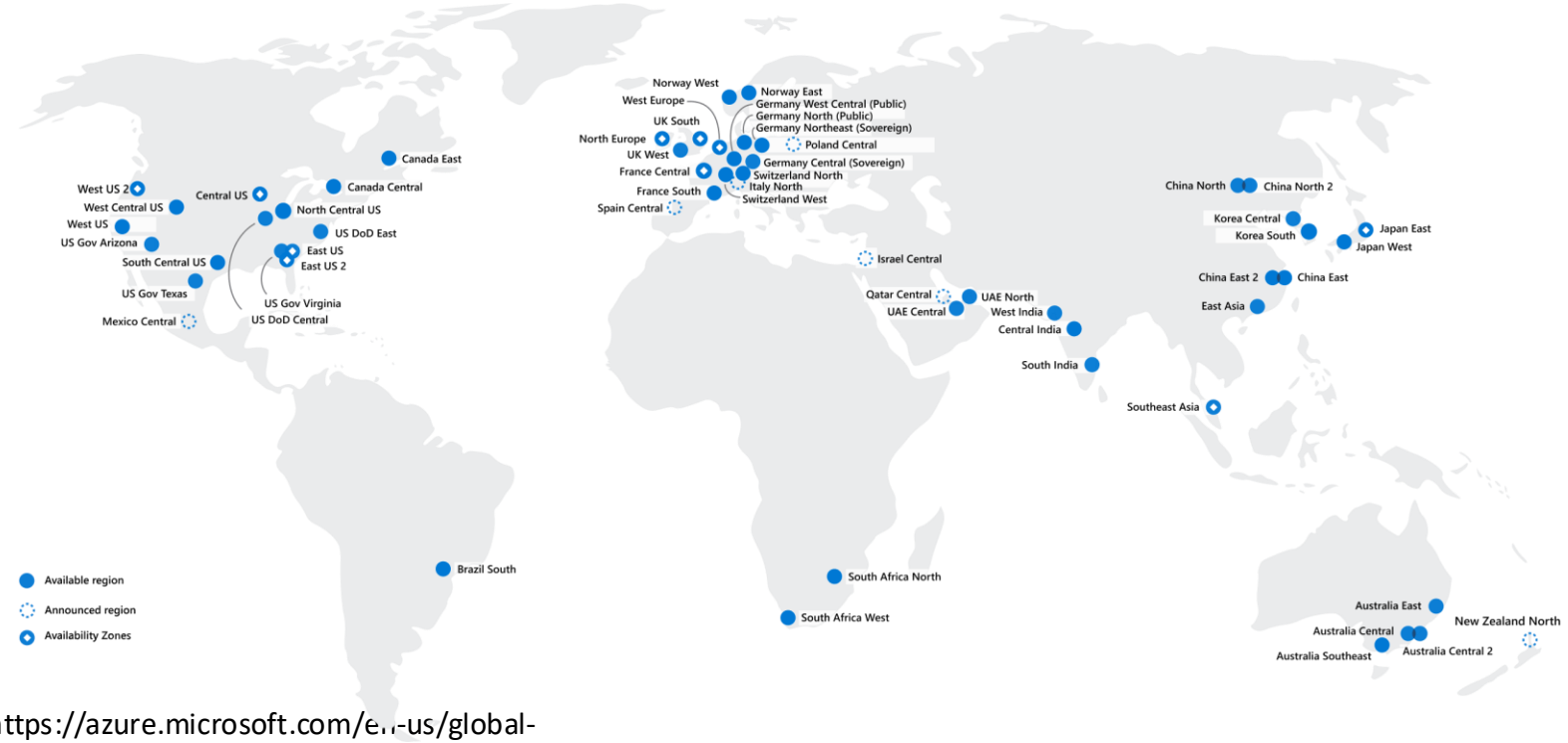


Web (HTTP), E-mail (SMTP), Audio/video-conf (e.g., Zoom, Skype), streaming (e.g., Netflix), multi-player games (e.g., Pokémon Go) etc.

EXEMPLO: CLOUD COMPUTING

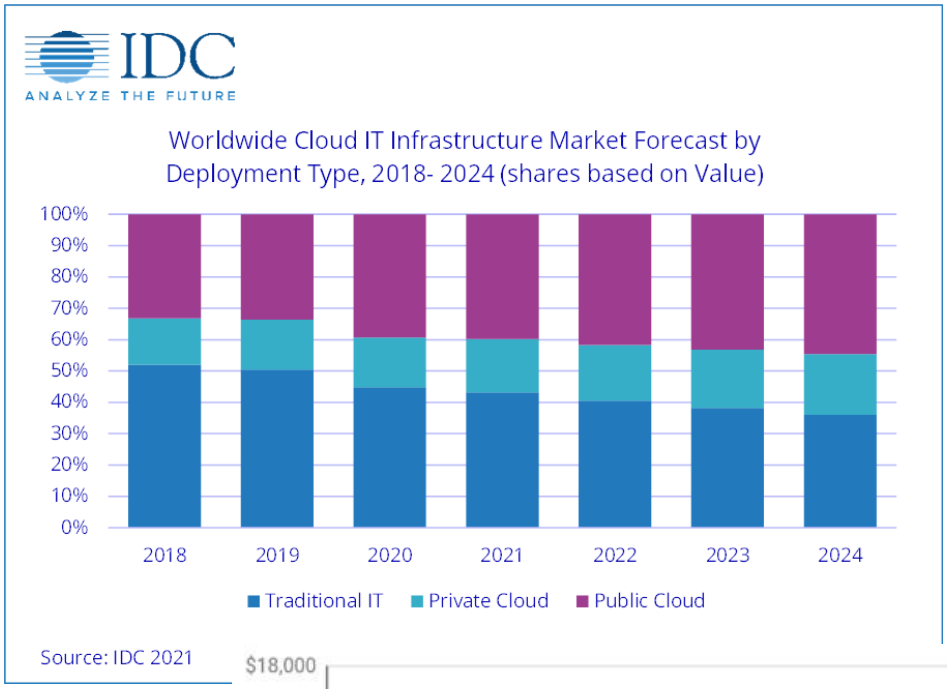
Diferentes tipos de serviço disponibilizados através duma infraestrutura distribuída globalmente.

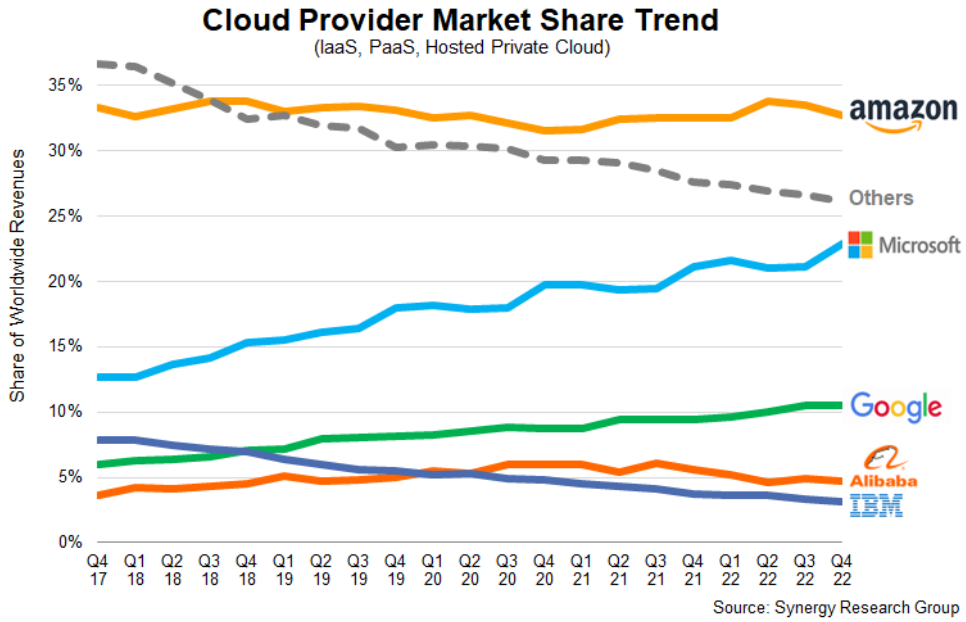
- IaaS (Infrastructure as a Service) : e.g. AWS EC2 (máquinas virtuais)
- PaaS (Platform as a Service) : e.g. Google App Engine (app service)
- SaaS (Software as a Service) : e.g. Google Docs

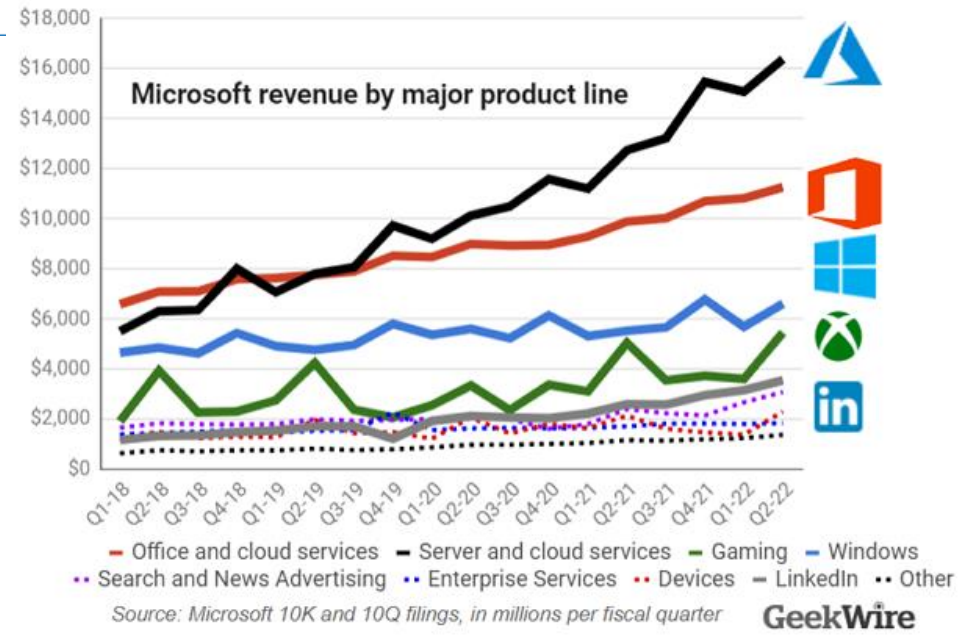


Azure locations: https://azure.microsoft.com/en-us/global-infrastructure/geographies/ (accessed: Sep, 2020)

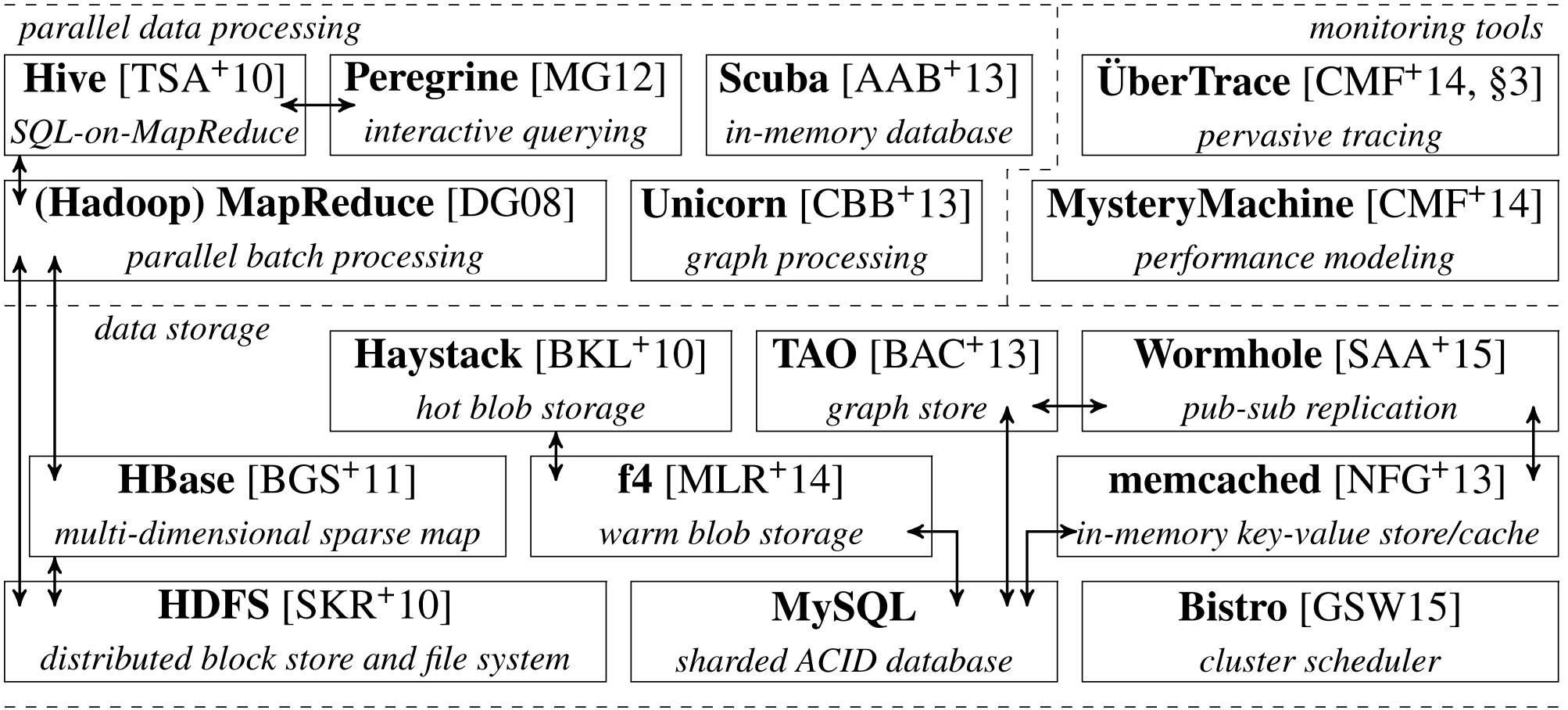
CLOUD COMPUTING: ALGUMAS TENDÊNCIAS



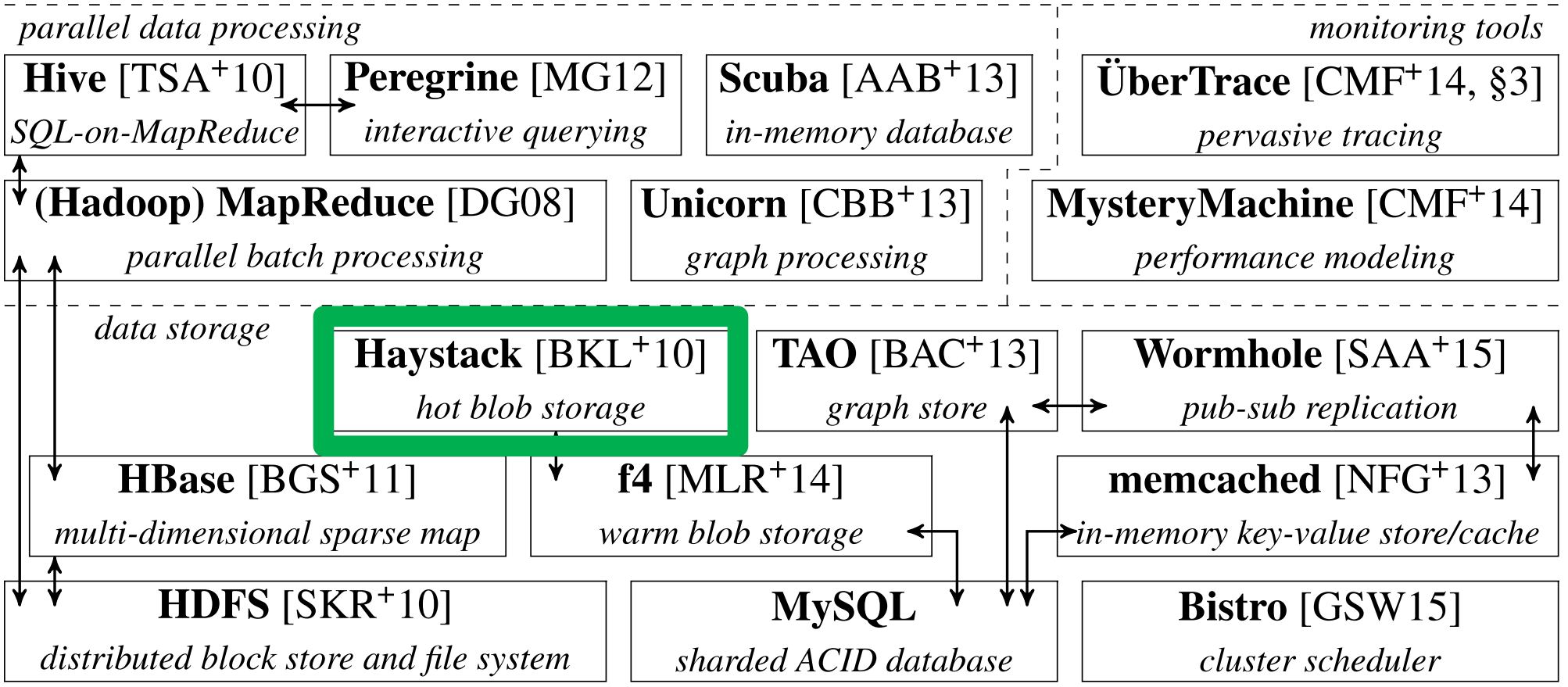




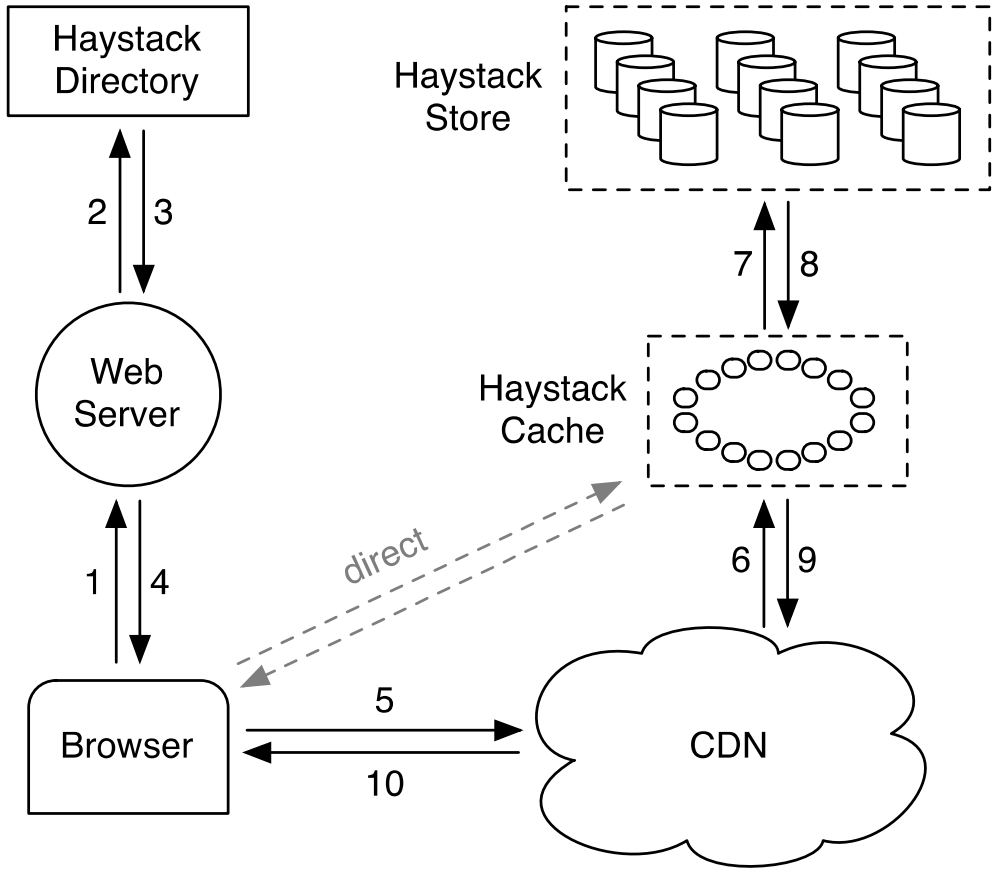
EXEMPLO: FACEBOOK / META



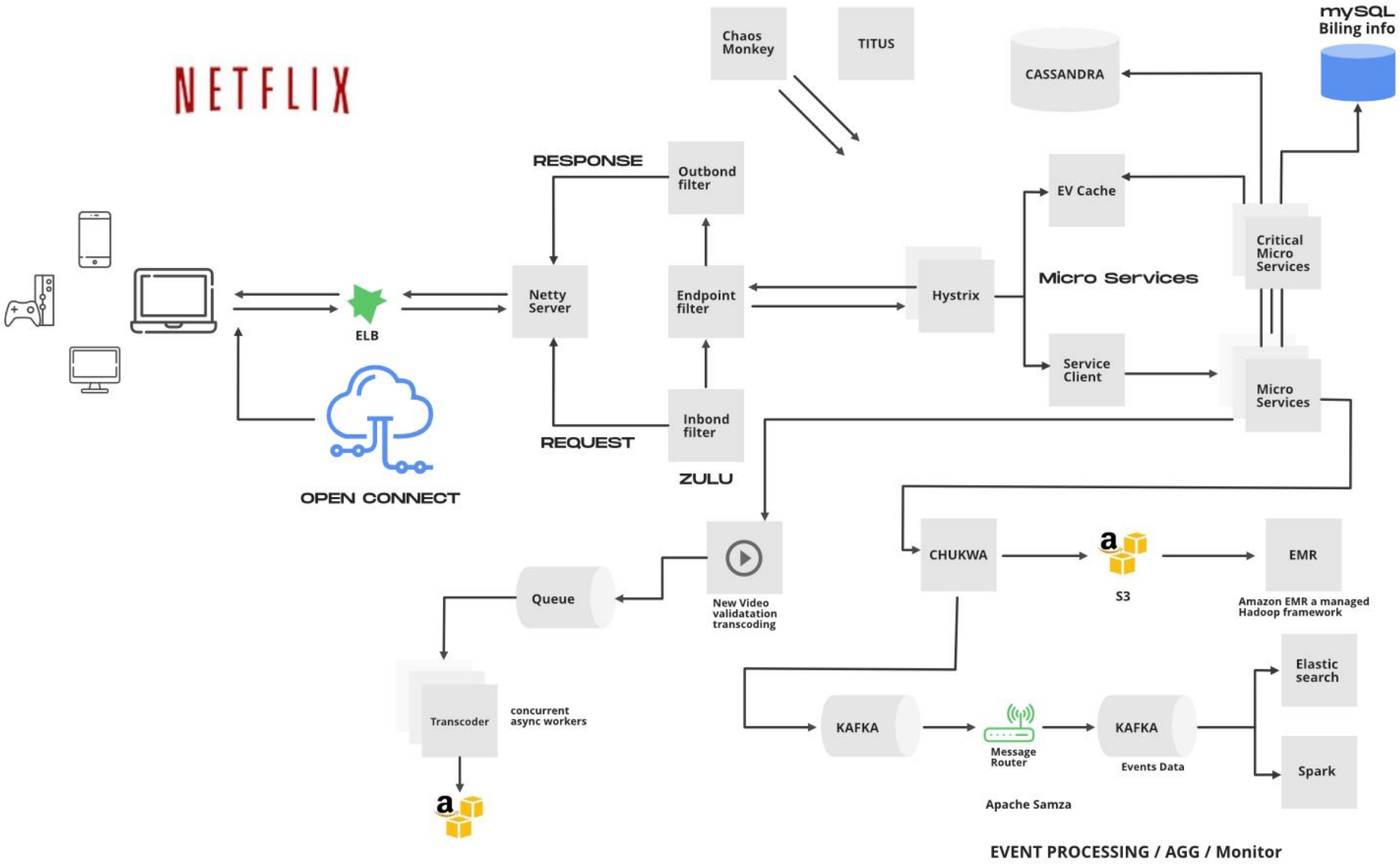
EXEMPLO: FACEBOOK / META



EXEMPLO: FACEBOOK HAYSTACK

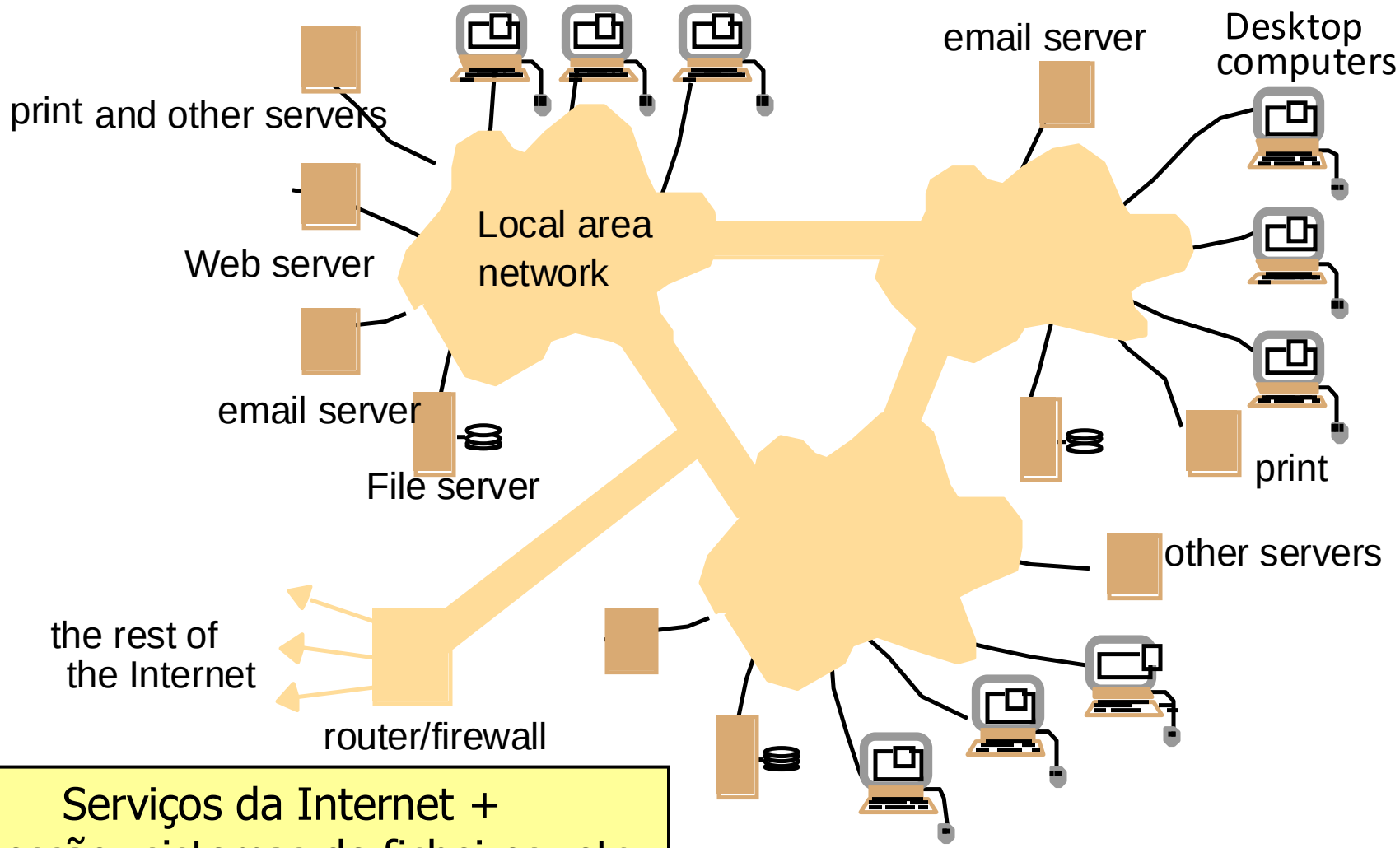


NETFLIX


NETFLIX
Chaos Monkey
TITUS
CASSANDRA
mySQL Biling info
RESPONSE
Outbond filter
EV Cache
Critical Micro Services
Micro Services
Netty Server
Endpoint filter
Hystrix
ELB
Service Client
Micro Services
Inbond filter
REQUEST
ZULU
OPEN CONNECT
CHUKWA
S3
EMR
Amazon EMR a managed Hadoop framework
Queue
New Video validatation transcoding
Transcoder
concurrent async workers
KAFKA
Message Router
KAFKA
Events Data
Elastic search
Spark
Apache Samza
EVENT PROCESSING / AGG / Monitor

EXEMPLO: SERVIÇOS EM INTRANETS

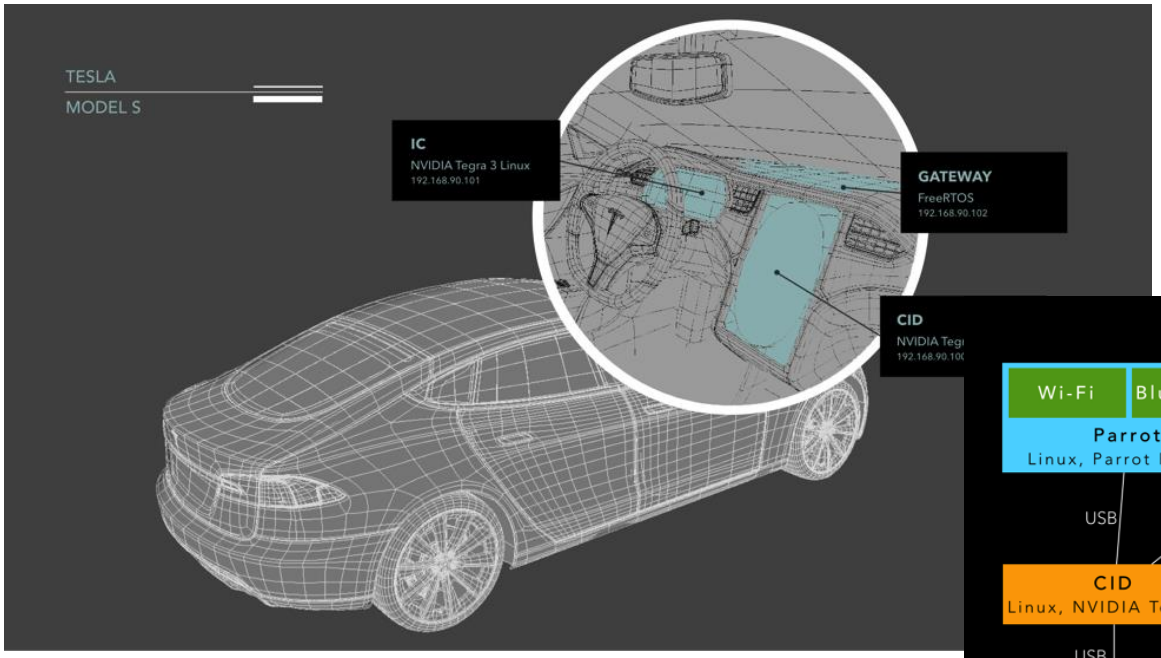
Intranets: redes isoladas fisicamente, redes isoladas logicamente (private virtual network), ligação à Internet através de Gateways protegidas por firewalls, etc.

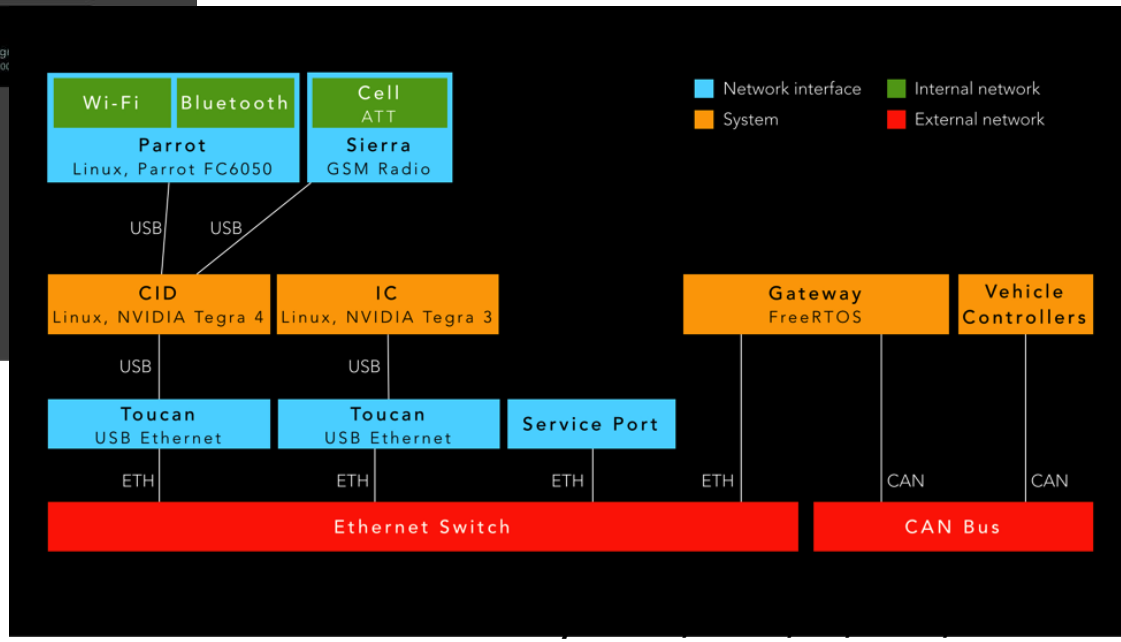


Serviços da Internet + impressão; sistemas de ficheiros; etc.

OUTROS EXEMPLOS: CARROS

Dispositivos ou máquinas especiais controlados através de conjuntos de computadores embebidos (por exemplo: um avião ou um carro, uma fábrica)





Fonte: https://blog.lookout.com/hacking-a-tesla

OUTROS EXEMPLOS

Sistemas de controlo de processos industriais em fábricas (por exemplo, linhas de montagem)

Clusters de computadores interligados através de redes de alta velocidade para cálculo paralelo

Porque é que Vocês podem querer dominar Sistemas Distribuídos?

- Porque literalmente eles estão em todo o lado:
  - Entretenimento
  - Serviços Sociais
  - Produtividade
  - Colaboração
  - AI
  - *Dating*
  - ...

- É um mercado em expansão com necessidades contínuas de engenheiros qualificados e que dominam não só as tecnologias mas também a conceção e arquitetura.
- Muita inovação hoje em dia ocorre no domínio de Sistemas Distribuídos (inúmeras e inúmeras start-ups).
- Estas competências valem muito dinheiro no mercado.

AULA 2

... NA AULA 1

O que é um sistema distribuído

- Um sistema distribuído é um conjunto de componentes hardware e software interligados através de uma infra-estrutura de comunicações, que cooperam e se coordenam entre si apenas pela troca de mensagens, para execução de aplicações distribuídas

Exemplos de sistemas distribuídos

- Serviços na Internet, cloud computing
- Sistemas de controlo (e.g. fábricas)
- Sistemas embebidos e de tempo-real (e.g. carros, aviões)

NESTA AULA

Características essenciais dos sistemas distribuídos

Desafios na conceção de sistemas distribuídos

Motivações dos Sistemas Distribuídos

Acesso generalizado sem restrições de localização

- Acessibilidade ubíqua (suporte para utilizadores fixos, móveis)

Partilha dos recursos distribuídos pelos diferentes utilizadores

- Exemplos: impressores, ficheiros

Distribuição da carga – melhoria do desempenho

Tolerância a falhas – melhoria da disponibilidade

Flexibilidade e adaptabilidade

- Decomposição de um sistema complexo num conjunto de sistemas mais simples

CARACTERÍSTICAS FUNDAMENTAIS

Componentes do sistema executam de forma concorrente – paralelismo real

- Necessidade de coordenação entre os vários componentes

Falhas independentes das componentes e das comunicações

- Impossível determinar se existe uma falha dum componente ou do sistema de comunicações
- Necessidade de tratar as falhas

Ausência de relógio global – existem limites para a precisão da sincronização dos relógios locais

- Impossível usar relógios locais para ordenar globalmente todos os eventos

IMPLICAÇÕES

Nenhum componente tem uma visão exacta instantânea do estado global de todo o sistema

Os componentes têm uma visão parcial do estado global do sistema

- Os componentes do sistemas estão distribuídos e só podem cooperar através da troca de mensagens, as quais levam um tempo não nulo a propagarem-se

Na presença de falhas, o estado global pode tornar-se incoerente, i.e., as visões parciais do estado global podem tornar-se incoerentes

- Por exemplo, réplicas de um objecto podem ficar incoerentes

DESAFIOS

Heterogeneidade

Abertura

Transparência

Segurança

Escala

Tratamento das falhas

HETEROGENEIDADE

Hardware: smartphones, tablets, portáteis, servidores, clusters, ...

- Diferentes características dos processadores, da memória, da representação dos dados, dos códigos de caracteres,...

Redes de interligação e protocolos de transporte: Redes móveis (5G, 4G, 3G, GSM), WLANs, wired LANs,..., TCP/IP, ...

Sistema de operação: Windows, MacOS, iOS, Android,...

- Diferentes interfaces para as mesmas funcionalidades

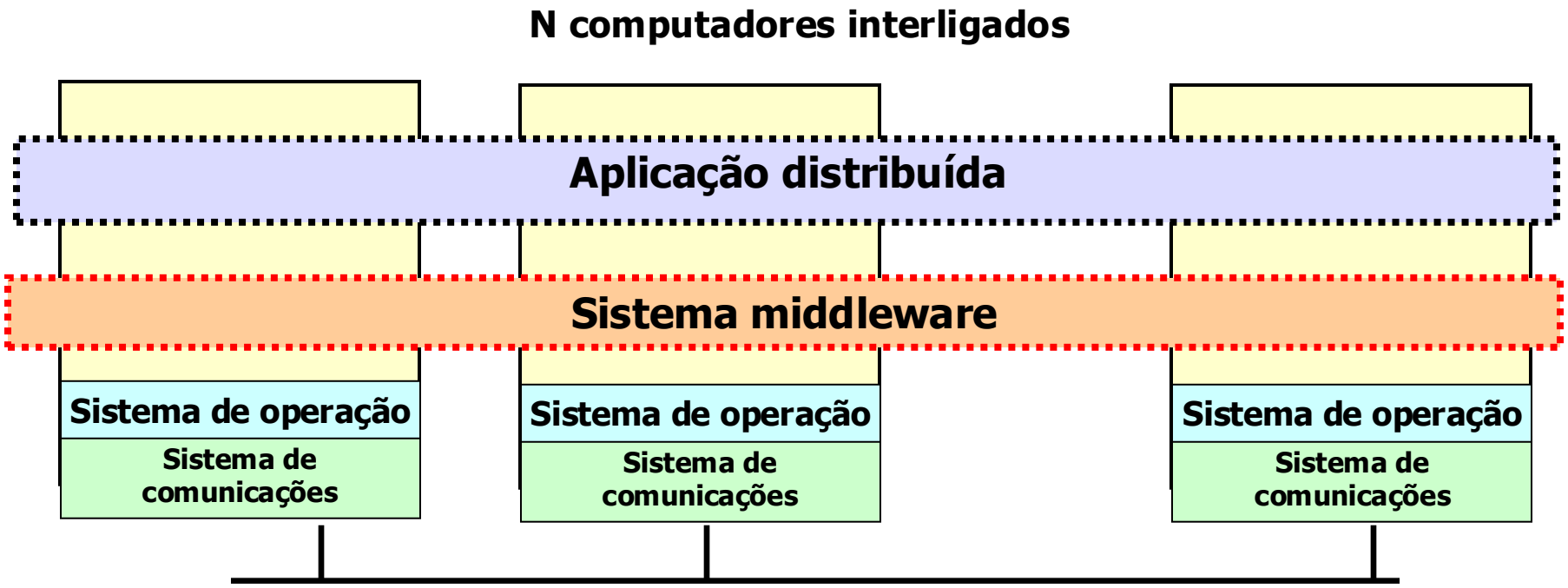
Linguagens de programação...

Lidar com esta heterogeneidade é muito complexo

As seguintes soluções podem ajudar:

- Sistemas de Middleware
- Máquinas Virtuais
- Containers

MIDDLEWARE



Sistemas operativos

- Interfaces heterogénea
- Serviços básicos

- Sistema *middleware*
  - Interface homogénea
  - Serviços mais complexos (invocação remota: Web-services; message-queue: *MQ, etc)
  - Verdadeira interoperabilidade requer idênticos interface e protocolos

MÁQUINA VIRTUAL

Objetivo: permitir executar os mesmos programas em máquinas com diferentes características

Máquina virtual aplicação: virtualiza ambiente de execução independentemente do sistema de operação

- E.g.: programas escritos numa única linguagem (JavaVM) ou em múltiplas linguagens (Microsoft CLR)

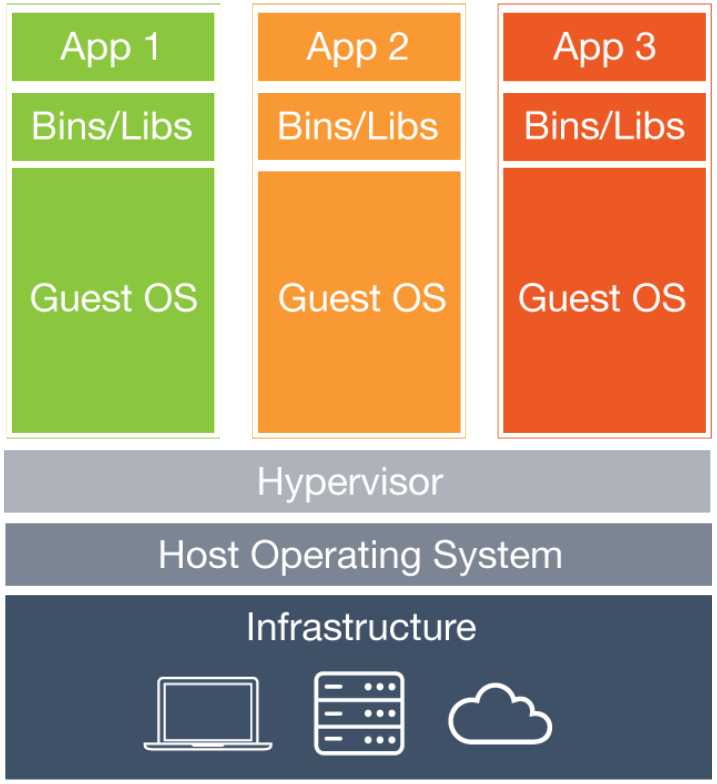
Máquina virtual sistema: virtualiza máquina física

- E.g.: VmWare, VirtualBox, etc.

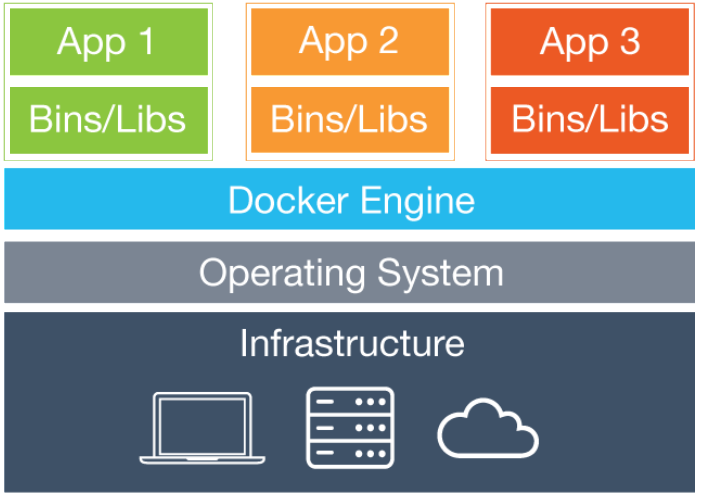
CONTAINER

Objetivo: permitir executar aplicações, incluindo todas as suas dependências

Máquina Virtual



Container



Images from docker.com

CONTAINER (CONT.)

Um *container* utiliza o sistema operativo subjacente do sistema (ou VM) onde está a ser executado…

- O sistema operativo é partilhado entre todos os *containers* em execução.

.. Mas os *containers* estão isolados uns dos outros com recursos exclusivos atribuídos, incluindo CPU, memória, rede, disco, etc.

Isto faz com que um *container* funcione como uma máquina independente, com o seu próprio IP.

ABERTURA

A abertura de um sistema determina o modo como pode ser estendido e re-implementado

Sistemas abertos

- Interfaces e modelo (incluindo protocolos de comunicação) conhecidos
- Evolução controlada por organismos de normalização independentes ou consórcios industriais
- Permite a interoperação de componentes com diferentes implementações

Sistemas proprietários

- Podem ser modificados pelo seu "dono"

Vantagens e desvantagens de cada aproximação?

ABERTURA E OPEN-SOURCE

Uma aplicação de código aberto (open source) é uma aplicação da qual o código fonte está disponível.

A utilização da aplicação é regida por uma licença que é tipicamente distribuída com o código. Existem vários tipos de licença:

Public domain – todos os direitos são transferidos.

Permissive licence (e.g. MIT, Apache) – oferece direito de utilização (incluindo relicenciar código derivado).

Copyleft (e.g. GPL) – oferece direito de utilização, proíbe tornar código proprietário (pode obrigar que todo o código que usa a aplicação tenha a mesma licença).

Noncommercial licence – oferece direito de utilização para utilização não comercial.

Proprietary licence – oferece direito de utilização em troca dum pagamento (utilização normal do copyright).

TRANSPARÊNCIA

A transparência (da distribuição) é a propriedade relativa a esconder ao utilizador e ao programador das aplicações a separação física dos elementos que compõem um sistema distribuído

- Simplicidade e flexibilidade são objectivos

Em certas circunstâncias, a transparência total é indesejável

- Que fazer em caso de falha?

SEGURANÇA

Necessidade de proteger os recursos e informação gerida num sistema distribuído

- Recursos têm valor para os seus utilizadores

Segurança tem três componentes:

- Confidencialidade: indivíduos não autorizados não podem obter informação
- Integridade: dados não podem ser alterados ou corrompidos
- Disponibilidade: acesso aos dados deve continuar disponível

Aspetos envolvidos

- Autenticação dos parceiros
- Canais seguros
- Prevenção de ataques de "negação de serviço" (denial of service attacks)

SEGURANÇA - RGPD

O **Regulamento Geral sobre a Proteção de Dados** (**RGPD**) (UE) 2016/679 regula a privacidade e proteção de dados pessoais, aplicável a todos os indivíduos na UE e Espaço Económico Europeu (EEU).

É aplicável a todas as empresas que operem no Espaço Económico Europeu, independentemente do seu país de origem.

Pretende garantir que:

- Não se podem guardar e disponibilizar dados sem consentimento explícito;
- Os dados guardados não podem ser utilizados sem que o proprietário tenha dado consentimento explícito;
- O proprietário tem o direito de revogar as permissões em qualquer momento.

ESCALA

A escala de um sistema distribuído é o âmbito que o mesmo abrange assim como o número de componentes.

A escala de um sistema tem várias facetas:

- recursos e utilizadores
- âmbito geográfico (rede local, país, mundo, ...)
- âmbito administrativo (uma organização, inter-organizações)

Um sistema capaz de escalar (escalável) é um sistema que continua eficaz quando há um aumento significativo do número de recursos e utilizadores

- i.e., em que não é necessário alterar a implementação dos componentes e da forma de interacção dos mesmos

COMO LIDAR COM A ESCALA ?

Para reduzir o número de pedidos tratados por cada componente

- Divisão de um componente em partes e sua distribuição
- Replicação e caching (problema da consistência entre réplicas e caches)

Para reduzir o tempo de acesso de clientes distribuídos geograficamente?

- Geo-replicação – replicar as aplicações (e dados) em diferentes locais geográficos
- Replicação na edge – replicar as aplicações ou dados na periferia da rede (e.g. CDN)

COMO LIDAR COM A ESCALA ?

Para reduzir dependências entre componentes

- Meios de comunicação assíncronos

Para simplificar o sistema

- Uniformidade de acesso aos recursos e dos mecanismos de cooperação, sincronização, etc.
- Meios de designação universais (independentes da localização e dos recursos)

AVARIAS, ERROS E FALHAS

Os componentes de um sistema podem falhar, i.e., comportar-se de forma não prevista e não de acordo com a especificação devido a erros (por exemplo a presença de ruído num canal de comunicação ou um erro de software) ou avarias (mecanismo que que entra em mau funcionamento)

Num sistema distribuído, as falhas são geralmente parciais (num componente do sistema) e independentes

- Um componente em falha pode induzir uma mudança de estado incorreta noutro componente, levando eventualmente o sistema a falhar, i.e., a ter um comportamento não de acordo com a sua especificação.

COMO LIDAR COM AS FALHAS

Detectar falhas

- Possível: e.g.: mensagens corrompidas através de checksums
- Pode ser impossível: Falha (crash) num computador remoto
  - Desafio: Funcionar através da suspeição das falhas

Mascarar falhas (após a sua detecção)

- Exemplos: retransmissão de mensagens, redundância

Tolerar falhas

- Definição do comportamento na presença de falhas
  - Parar até falhas serem resolvidas; recorrer a componentes redundantes para continuar a funcionar

Recuperação de falhas

- Mesmo num sistema que tolere falhas é necessário recuperar os componentes falhados. Porquê?
- Problema: recuperar estado do serviço

PARA SABER MAIS

G. Coulouris, J. Dollimore and T. Kindberg,
Distributed Systems – Concepts and Design,
Addison-Wesley, 5th Edition, 2011

Capítulo 1.