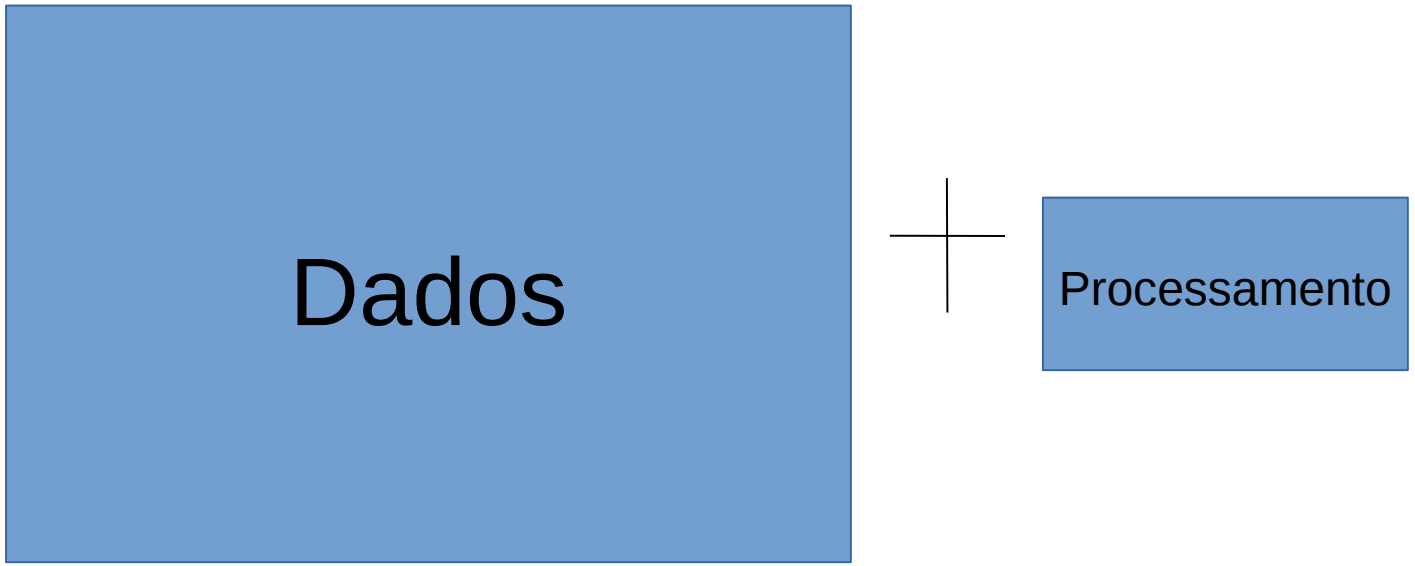
Designing Data Intensive Applications

Capítulo 1

Carmem Hara

Aplicações Atuais



Problemas

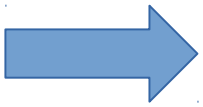
- Volume
- Complexidade
- Velocidade de atualização

Tecnologias

- SGBD: armazenamento
- Cache: resultados intermediários
- Índices: otimização de buscas
- Processamento de fluxo
- Processamento em lote

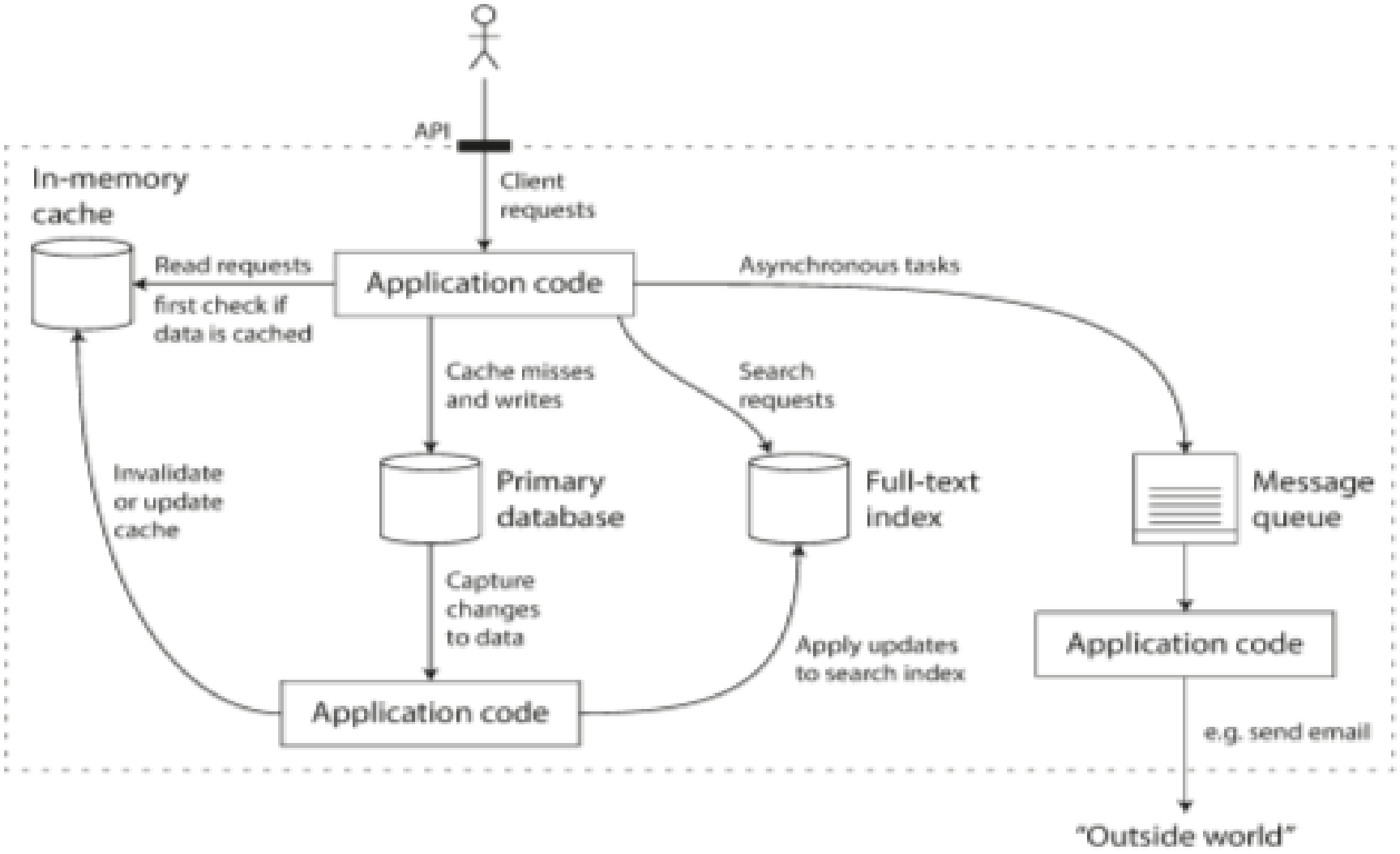
Qual o sistema de gerência de dados ideal?

- Depende da aplicação
- Uma ferramenta em geral não atende todos os requisitos

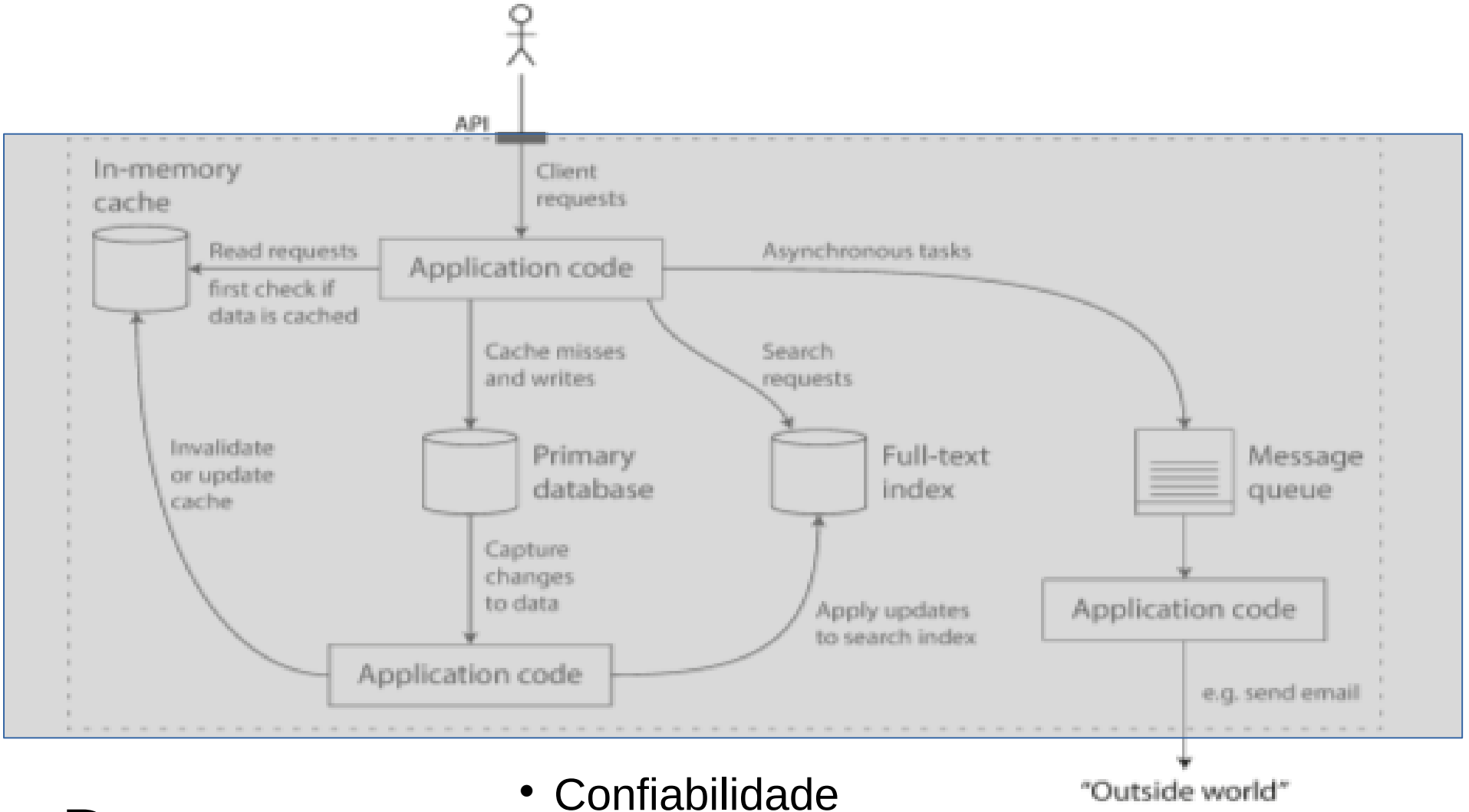


Necessidade de integrar diversas soluções

Possível Arquitetura de Integração



Novo sistema de gerência de dados



Deve prover:

- Confiabilidade
- Escalabilidade
- Manutenção

CONFIABILIDADE

Confiabilidade

Habilidade de funcionar corretamente mesmo na presença de adversidades.

- Executa as funções da forma esperada
- Tolera erros cometidos pelo usuário
- O desempenho é adequado para a carga planejada
- Não permite acessos não autorizados

Falhas

- Mecanismos
  - Tolerância a falhas
  - Prevenção de falhas
- Tipos de Falhas
  - De Hardware
  - De Software
  - Humana

Falhas de Hardware

- Falhas de RAM, de energia, ...
- Falha de disco
  - MTTF (*mean time to failure*): 10 a 50 anos
    - Cluster de 10.000 discos ⟹ 1 disco falha por dia
- Tolerância a falhas: redundância
  - RAID (*Redundant Array of Independent Disks*)
  - Baterias, geradores de energia

Falhas de Software

Exemplos

- Erros causados por entrada errada
- Sistema consome todos os recursos (CPU, memória, rede, espaço em disco)
- Um serviço utilizado pelo sistema torna-se lento ou não responde
- Falhas em cascata

Em geral ocorrem devido a uma combinação de circunstâncias não previstas.

Falhas de Software

Prevenção

- Planejamento: interações entre componentes, hipóteses
- Testes
- Isolamento do processo

Tolerância

- Reinicialização do sistema
- Monitoramento

Falhas Humanas

Humanos:

- Desenvolvem os sistemas
- Operam os sistemas

Erros humanos causam a maioria das falhas de sistemas computacionais*

- Erros de hardware: 10-25%

* D. Oppenheimer et al: “Why Do Internet Services Fail, and What Can Be Done About it?” Usenix Symposium, 2003

Falhas Humanas - Prevenção

- Desenvolver sistemas que minimizem o erro
- Disponibilizar ambientes de teste separados do ambiente de produção
- Realizar testes em todos os níveis: de unidade ao ambiente integrado
- Facilitar a recuperação: desfazer alterações na configuração ou computações
- Monitorar o sistema: desempenho e taxas de erro
- Fazer treinamento

ESCALABILIDADE

Escalabilidade

É a habilidade do sistema dar suporte ao aumento da carga de trabalho.

Animoto:

- 2008: 25.000 para 750.000 usuários em 4 dias
  - 5.000 servidores na AWS
- 2013: 6 milhões de usuários

Descrição de Carga

Parâmetros de carga:

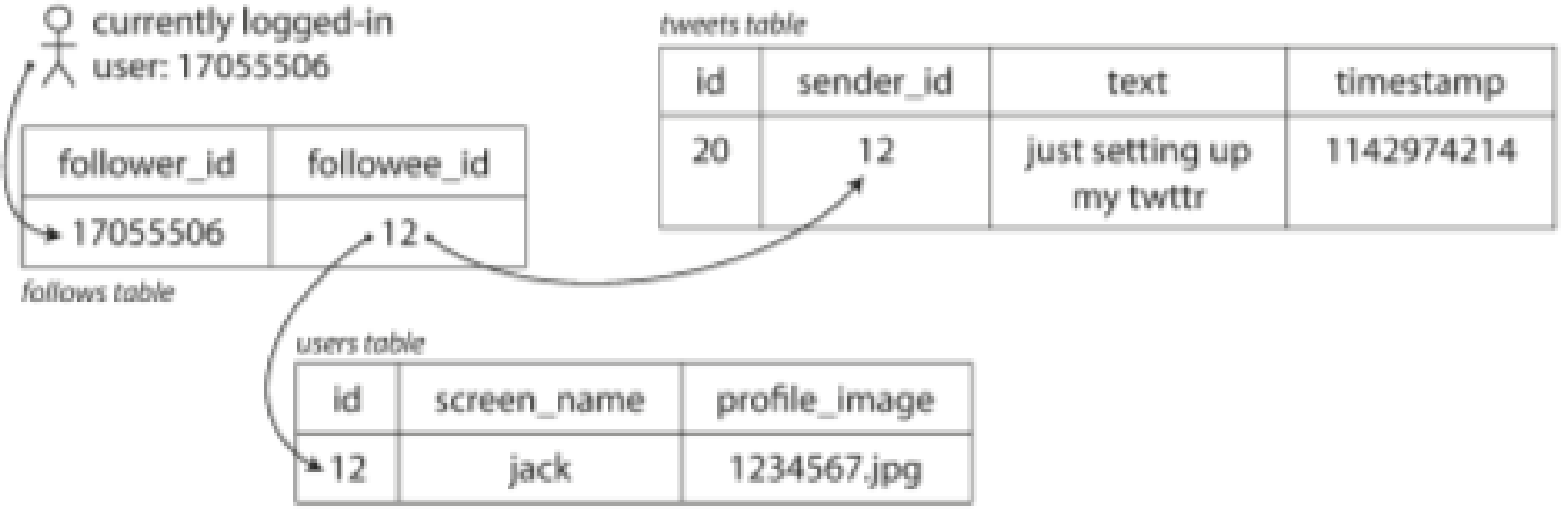
- Requisições por segundo de um servidor Web
- Porcentual de leituras e escritas em um SGBD
- Quantidade de usuários ativos em um chat
- *Hit rate* de dados em cache

Exemplo de carga: Twitter

Novembro de 2012:

- Posts:
  - Média de 4.600 posts por segundo
  - Pico de 12.000 posts por segundo
- Leitura:
  - Média de 30.000 visualizações por segundo

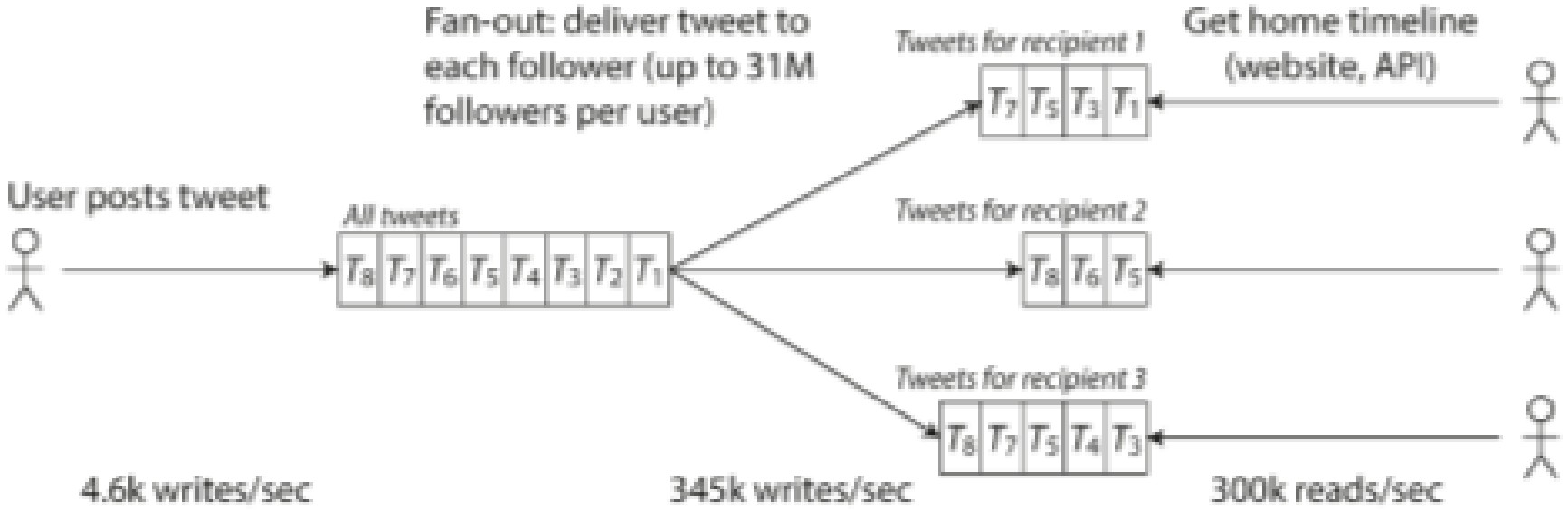
Arquitetura 1: Busca sob demanda



- O sistema busca as mensagens postadas sob demanda (no momento da leitura)
- Carga de leitura > Carga de postagem

⇒ Processamento em tempo de postagem

Arquitetura 2: Fila para cada leitor



Problema: fan-out de “celebridades”

Solução: arquitetura híbrida

Parâmetro de carga: fan-out (quantidade de seguidores)

Descrição do Desempenho

Se a carga aumentar:

1) Sem aumentar os recursos, como o desempenho será afetado?

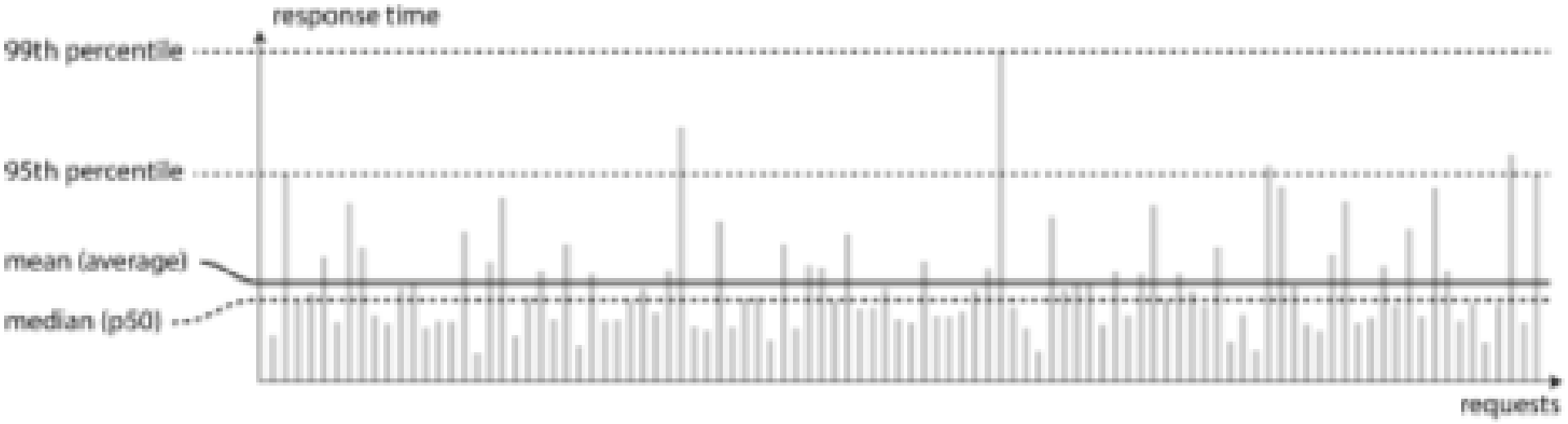
2) Qual o incremento necessário de recursos para que o desempenho seja o mesmo?

Desempenho:

- Processamento em lote: *throughput* (vazão)
- Processamento online: tempo de resposta

Tempo de resposta X Latência

Métricas de Desempenho

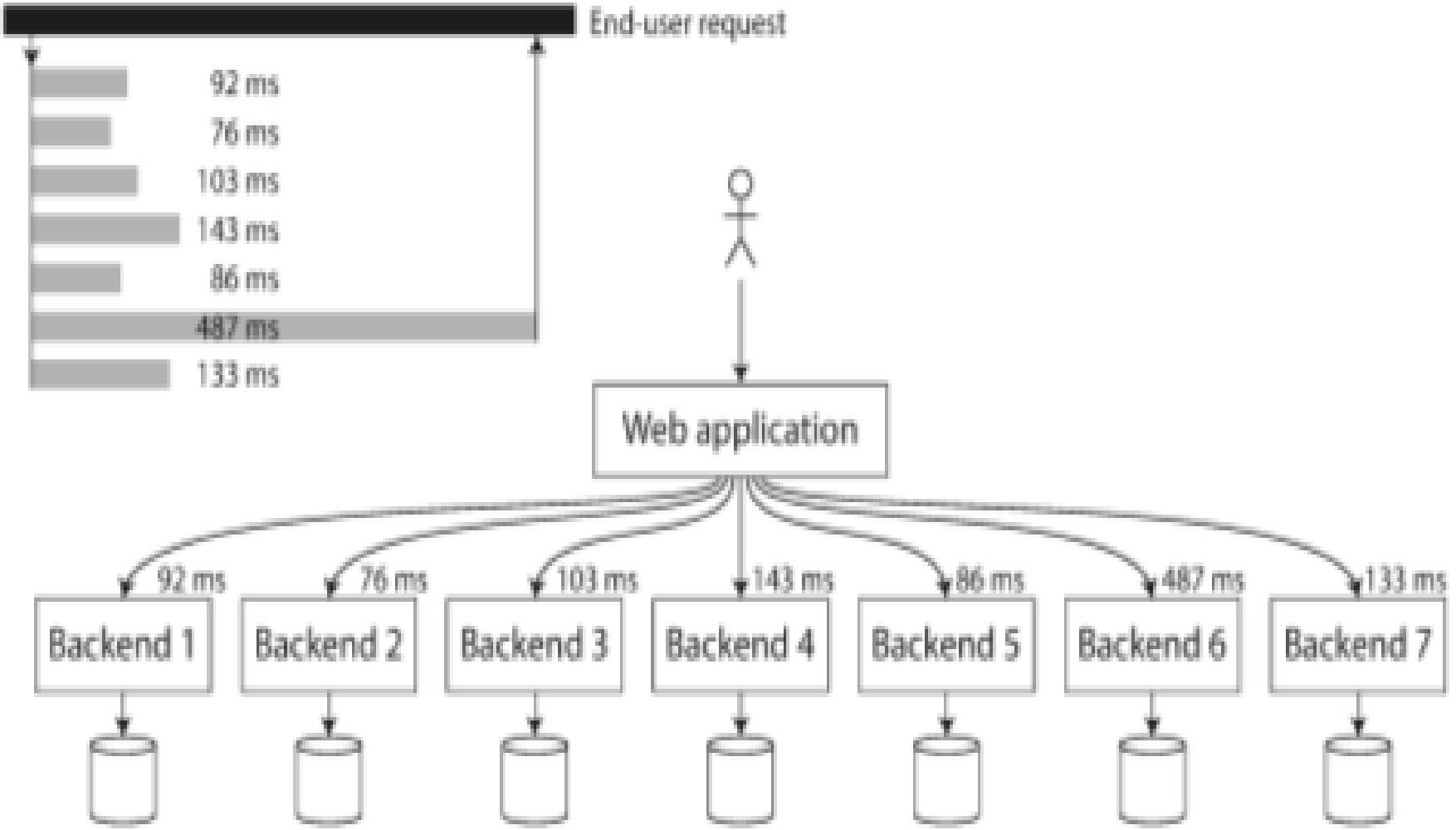


- Média: (*soma de N medições / N)*
- Percentil: ordena as medições
  - Mediana: ponto médio = p50
  - P90 = 2 segundos:

    90% das requisições em até 2 segundos

Tempo de Resposta

Aplicações que fazem chamadas a diferentes serviços



Escalabilidade

Tipos:

- Vertical (*scale-up*): trocar por uma máquina com mais recursos
  - Em geral o custo é maior
  - Sistemas mais simples
- Horizontal (*scale-out*): distribuir a carga por mais máquinas simples
  - Arquitetura shared-nothing
  - Sistemas complexos de envolvem serviços dependentes de estados (stateful)

Construção de Sistemas Escaláveis

- Utiliza padrões e técnicas/algoritmos (modelos de dados, replicação, particionamento, protocolos de transações)

Elasticidade: capacidade de automaticamente alocar e liberar recursos de acordo com a carga

MANUTENÇÃO

Manutenção

- Custo de software: maior parte é de manutenção
- 3 princípios para o desenvolvimento de sistemas:
  - Facilidade de operação
  - Simplicidade de projeto
  - Facilidade de evolução / adaptabilidade

Operação de Sistemas

"Uma boa operação pode contornar limitações de um sistema ruim ou incompleto, mas um bom sistema não funciona de forma confiável com uma má operação."

➡ Um sistema deve facilitar tarefas rotineiras

Atividades de uma Equipe de Operadores

- Monitorar o funcionamento do sistema e tomar providências para recuperá-lo
- Investigar causas de mal funcionamento
- Manter o software atualizado
- Documentar dependências entre sistemas
- Antecipar problemas (planejamento de carga)
- Executar atividade de manutenção
- Manter a segurança do sistema

Como os sistemas de gerência de dados podem dar suporte a estas atividades

- Prover monitoramento, dando visibilidade para o comportamento (interno) do sistema
- Prover suporte para automação e integração com ferramentas padronizadas
- Evitar dependência com máquinas específicas
- Disponibilizar uma boa documentação e manuais
- Prover um bom comportamento default, mas dar liberdade para alterá-lo
- Ter comportamento previsível

Complexidade do Sistema

Fontes de complexidade

- Forte acoplamento entre os módulos
- Terminologia e nomenclatura inconsistente
- Tratamento de casos especiais
- Código obscuro para melhor desempenho

A complexidade é acidental se não é inerente ao problema tratado.

A abstração é uma importante ferramenta para diminuir a complexidade.

Evolução do Sistema

Sistemas simples e fáceis de entender em geral também são faceis de manter.

Métodos ágeis de desenvolvimento:

- Fornece um framework de desenvolvimento adaptável a mudanças
- *Test-driven development* (TDD)
- Refatoramento

Desenvolvimento de Sistemas com Uso Intensivo de Dados

- Requisitos funcionais
  - Funcionalidades esperadas do sistema para solução do problema
- Requisitos não funcionais
  - Confiabilidade
  - Escalabilidade
  - Facilidade de manutenção