Sistemas Computacionais Distribuídos

Prof. Dr. Rodolfo I. Meneguette
meneguette@icmc.usp.br

Aula baseada nos slides do prof Jo Ueyama e no professor Renata Lobato

Um aparte...

- Lei de Murphy
  - Forma proverbial: Se algo pode dar errado, dará errado
  - Enunciado original: Se há mais de um modo de fazer algo, e um desses modos resultar em um desastre, então alguém o fará desse modo
- Corolários
  - Lei de Silvermoon
    - Nada é a prova de falhas para um infeliz suficientemente talentoso
  - Lei da Dinâmica Negativa de Fineagle
    - Qualquer coisa que possa sair errado, sairá -- e no pior momento possível

Partially supported by AWS in Education Grant award

Entretanto, a Lei de Murphy é grave…

- Teste de confiabilidade no sistema de proteção do reator nuclear da usina de Chernobyl
  - Objetivo: validar a atuação do sistema alternativo de refrigeração
  - 26/04/1986: sistemas principais parcialmente desativados
  - O mecanismo teste não funcionou como esperado
    - Falha de projeto, falha de treinamento…
  - Perda do controle da reação nuclear, derretimento do núcleo
  - Explosão do reator
- Maior acidente nuclear da história… até 2011

Partially supported by AWS in Education Grant award

Reator 4 de Fukushima Dai-ichi

- Tsunami interrompeu fornecimento de energia para bombas que faziam água circular nos reatores
- Contingência com coleta de água do mar
  - Demora para agir
  - Exposição do núcleo dos reatores >> calor
  - Oxidação do Zircônio deixou apenas hidrogênio
  - Explosão
- Aprendizado: prever falta de energia por muito tempo

Partially supported by AWS in Education Grant award

Tópicos

- Confiabilidade e tolerância a faltas
  - Terminologia
  - Classificação
  - Uso de redundância
  - Técnicas de recuperação
- Segurança
  - Aspectos básicos

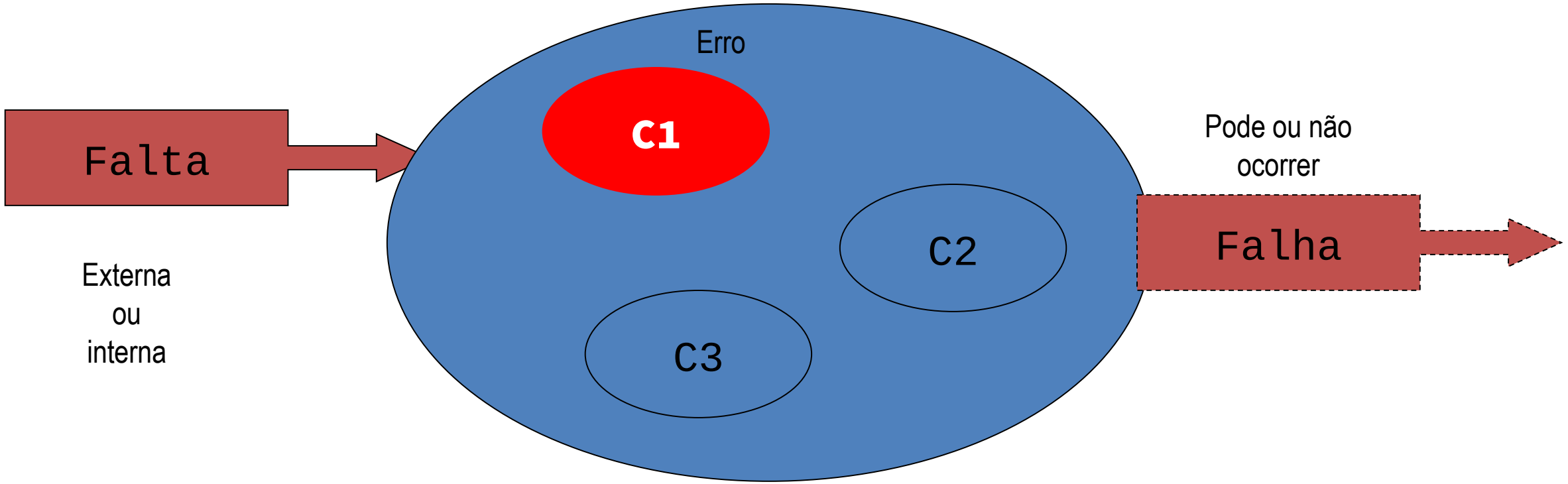
Partially supported by AWS in Education Grant award

Motivação

- Desejo primário
  - Execução correta da computação submetida
  - Resultados consistentes e coerentes
- Terminologia (autores divergem)
  - Failure (falha): efeito observado do erro. Transição do correto para o incorreto
  - Error (erro): consequência das faltas, impedindo o funcionamento. Diferentes faltas podem gerar mesmo erro
  - Fault (falta): defeitos no sistema

Partially supported by AWS in Education Grant award

Terminologia



Partially supported by AWS in Education Grant award

Falta, erro, falha

- Exemplo: HD
  - Um setor do disco pode estar com defeito (falta)
  - Um erro de leitura pode ocorrer se um programa tentar ler ou escrever neste setor
  - Pode ocorrer uma falha em um sistema que tente acessar este setor e não consiga
  - Se os dados gravados neste setor estiverem replicados em outro local, o sistema pode tolerar a falta e não apresentar falha

Partially supported by AWS in Education Grant award

Falta, erro, falha

- Nem todos os defeitos causam falhas
  - Chip com problema não está sendo usado
  - Porta de comutador com problema trocada antes de conectar servidor

Partially supported by AWS in Education Grant award

Falhas em cascata

- Falha em C1 ➔ falha em C2 ➔ ...

- Uma falta no disco pode causar uma falha no sistema de arquivos
  - Os servidores Web e de e-mail, que usam o sistema de arquivos, podem falhar
  - Uma aplicação de comércio eletrônico baseada na Web pode também falhar

Partially supported by AWS in Education Grant award

Conceitos

- Previsão de Faltas
  - Estima a probabilidade de que faltas ocorram
  - Permite que se avalie os riscos de falha
- Remoção de Faltas
  - Consiste em detectar e remover as faltas antes que causem erros e falhas
  - Usar ferramentas como depuradores, verificadores de discos...
- Prevenção de Faltas
  - Elimina as condições que fazem com que faltas ocorram durante a operação do sistema
  - Usa replicação interna, técnicas de validação…

Partially supported by AWS in Education Grant award

Conceitos

- Tolerância a Faltas
  - Propriedade de sistemas que não falham necessariamente ao se deparar com uma falta
- Sistemas Tolerantes a Faltas
  - São sistemas capazes de tolerar faltas encontradas durante a sua execução
- Técnicas de Tolerância a Faltas
  - Permitem prevenir falhas contornando as faltas que os sistemas podem vir a apresentar

Partially supported by AWS in Education Grant award

Tipos de faltas

- Classificação em relação à sua origem
  - Física: causada pelo hardware
  - Arquitetural: introduzida durante a fase de projeto do sistema
  - Interacionista: ocorrida nas interfaces entre componentes do sistema ou na interação com o mundo exterior
- Classificação em relação à sua natureza
  - Acidental ou Intencional
  - Maliciosa ou Não

Partially supported by AWS in Education Grant award

Tipos de faltas

- Classificação em relação ao seu surgimento
  - Na fase de desenvolvimento do sistema
  - Na fase de operação do sistema
- Classificação em relação à sua localização
  - Interna
  - Externa
- Classificação em relação à persistência
  - Temporária (transiente ou intermitente)
  - Permanente

Partially supported by AWS in Education Grant award

… em relação à persistência

- Transientes
  - Refazer operação resolve
  - Transmissão infra-vermelho entre Palms
    - Não receber o beam → falha
    - Interromper o feixe → defeito transiente
- Intermitentes
  - Acontecem de tempos em tempos
  - Componentes aquecendo, conector com folga
- Permanentes

Partially supported by AWS in Education Grant award

Tipos de faltas

- Classificação com base no modelo de faltas
  - Faltas Omissivas
    - Crash: deixa de funcionar permanentemente
    - Omissão: sistema deixou de fazer o que deveria em um determinado instante
    - Temporal: sistema atrasou-se para executar uma determinada ação
  - Faltas Assertivas
    - Sintática: formato da saída é inadequado
    - Semântica: saída apresenta valor incorreto
  - Faltas Arbitrárias: omissivas + assertivas

Partially supported by AWS in Education Grant award

Confiabilidade

- Capacidade de atender a especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período

Partially supported by AWS in Education Grant award

Como aumentar confiabilidade?

- Aumentar confiabilidade dos componentes
  - Nem sempre é possível
- Tornar o sistema tolerante a falha
  - Falhas são mascaradas
  - Redundância ou de HW e/ou SW

Partially supported by AWS in Education Grant award

Como aumentar confiabilidade?

- Idéia básica
  - Probabilidade de falha: p em 1 segundo
  - Funcionar por k segundos consecutivos
    - $p(1 - p)^{k-1}$
  - Tempo esperado para falha
    - MTTF = 1/p
- MTTF = 10^6 ➔ probabilidade de parar 10^{-6}
  - MTBF, MTTR

$$MTTF = \sum_{k=1}^{\infty} kp(1-p)^{k-1}$$

Partially supported by AWS in Education Grant award

Tolerância a faltas

- Prevenção e remoção de faltas não são suficientes quando o sistema exige alta confiabilidade ou alta disponibilidade.
- Sistema deve ser construído usando técnicas de tolerância a faltas
  - Garantia de funcionamento correto do sistema mesmo na ocorrência de faltas
  - Baseadas em redundância, exigindo componentes adicionais ou algoritmos especiais

Partially supported by AWS in Education Grant award

Tolerância a faltas

- A tolerância a faltas não dispensa as técnicas de prevenção e remoção
- Sistemas construídos com componentes frágeis e técnicas inadequadas de projeto não conseguem ser confiáveis pela simples aplicação de tolerância a faltas

Partially supported by AWS in Education Grant award

Uso de redundância

- Solução típica para tolerância a faltas
  - Redundância de informação
    - Informação extra, para permitir recuperação
    - Códigos de Hamming, CRC, Reed-Solomon
  - Redundância de tempo
    - Executar novamente se necessário
    - Operações idempotentes, transações
  - Redundância física
    - Replicação ativa
    - Primário-backup

Partially supported by AWS in Education Grant award

Replicação

- O acesso a serviços ou dados replicados deve ser transparente para o usuário
  - Manutenção de consistência das réplicas deve ser automática
  - Mesmo que mais de uma réplica responda a uma requisição, apenas uma resposta deve ser entregue ao usuário

Partially supported by AWS in Education Grant award

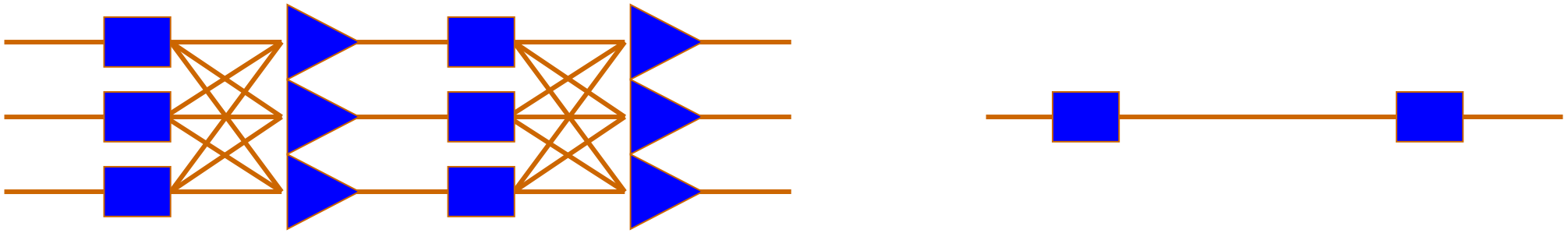
Replicação

- Definem como as réplicas se comportarão durante o funcionamento normal do sistema e sob a presença de faltas
- Principais Técnicas de Replicação
  - Replicação Passiva (Primário-Backup)
  - Replicação Ativa

Partially supported by AWS in Education Grant award

Replicação ativa

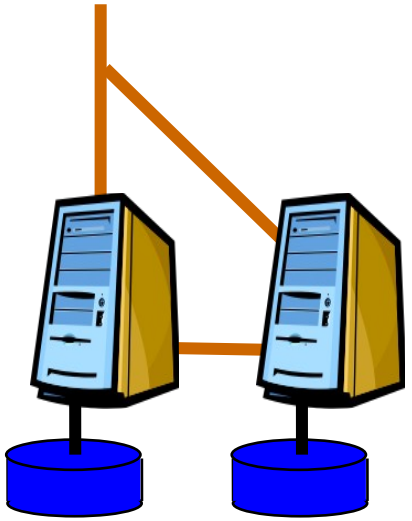
- Conceito natural
  - Dois pulmões, dois olhos, dois ouvidos...
  - Três turbinas...
- Implementação computacional
  - Triple Modular Redundancy



Partially supported by AWS in Education Grant award

Primário-backup

- Toda a computação é feita pelo primário
- No caso de falha, backup assume
  - Transição suave para os clientes
- Vantagens
  - Mais simples: mensagens apenas para o primário
  - Mais barato: 1 backup por primário
- Impossível usar para falhas bizantinas
  - Primário responde, e errado



Partially supported by AWS in Education Grant award

Detecção de falhas

- A falha em um componente pode levar todo o sistema a falhar
  - Recuperar componente para restaurar a capacidade do sistema de tolerar faltas
  - Na replicação passiva, se o único backup existente assume o lugar do primário, é preciso criar um novo backup
- É preciso detectar as falhas sofridas pelos componentes para poder recuperá-los

Partially supported by AWS in Education Grant award

Detecção local de falhas

- Podem ser usados diversos métodos
  - Rotinas de auto-verificação (self-check)
  - Guardiões: verificam constantemente as saídas geradas por um componente
  - Watchdogs : componente deve constantemente reiniciar um temporizador antes que ele se esgote, indicando uma falha
- Mesmo para um observador local, processos lentos podem parecer falhos

Partially supported by AWS in Education Grant award

Tipos de detectores de falhas

- Perfeitos
  - Determinam precisamente se um componente do sistema falhou ou não
  - Todos os componentes têm a mesma visão
- Imperfeitos
  - Detectores determinam se um processo é suspeito de falha ou não
  - Diferentes componentes podem ter visões distintas de um mesmo componente

Partially supported by AWS in Education Grant award

Tipos de detectores de falhas

- Detectores perfeitos são difíceis de obter
- Detectores quase-perfeitos podem ser obtidos usando crash controlado
  - Se um componente é suspeito de falha, ele é removido do sistema
  - O componente passa a ser ignorado por todos os demais componentes
  - Pode levar a descartar componentes que estão funcionando corretamente

Partially supported by AWS in Education Grant award

Recuperação de falhas

- Recuperação por retrocesso (backward error recovery)
  - Voltar a um estado anterior ao erro e continuar ativo
    - Reinicia a execução de um método, retransmite pacotes perdidos, etc…
  - Operações posteriores ao instante de retrocesso são perdidas, mas seu efeito pode ainda ser sentido no sistema, levando possivelmente a inconsistências

Partially supported by AWS in Education Grant award

Recuperação de falhas

- Recuperação por avanço (forward error recovery)
  - Tomar medidas que anulem ou aliviem o efeito do erro e continuar a operar normalmente
    - Descartar pacotes, substituir um valor inválido pelo valor válido anterior
  - Usada quando não há tempo para voltar para estado anterior e retomar execução, ou quando ações não podem ser desfeitas

Partially supported by AWS in Education Grant award

Motivação

- Sistema de arquivos é componente chave
- Serviço x servidor
  - Especificação de interface
  - Processo que implementa a interface
- Um serviço pode ser implementado por vários servidores
  - Problemas de replicação

Partially supported by AWS in Education Grant award

Pontos básicos

- Todas as aplicações necessitam armazenar e recuperar informações
  - Grandes volumes de informação (RAM?)
  - Persistência
  - Compartilhamento entre processos
- Armazenar os dados em arquivos
- Gerenciamento feito pelo SO através do sistema de arquivos

Partially supported by AWS in Education Grant award

Pontos básicos

- Sistema de arquivos vem desde 1960
  - Responsável pela organização, armazenagem, recuperação, nomeação, compartilhamento e proteção dos arquivos
- Oferecem uma abstração ao usuário
  - Dados → arquivo
- São projetados para gerenciar grande volume de arquivos

Partially supported by AWS in Education Grant award

Pontos básicos

- Nomeação dos arquivos é feita nos diretórios
  - Arquivo especial com mapeamento entre nomes textuais e identificadores internos de arquivos
- Geralmente es-
  trutura modular
  em camadas

Diretório	Relaciona nomes a FIDs
Arquivo	Relaciona FIDs a arquivos
Controle de acesso	Concede/nega acesso
Acesso a arquivo	Lê/escreve dados/atributos
Bloco	Recupera/aloca blocos de disco
Dispositivo	E/S no disco e buffer

Partially supported by AWS in Education Grant award

Tipos de serviços de arq...

- Modelo upload/download
  - Servidor lida apenas com arquivos inteiros
    - Cliente recebe/envia arquivo todo
  - Vantagem
    - Simplicidade conceitual
  - Desvantagens
    - Transferência do arquivo todo (tempo)
    - Cliente precisa de espaço para armazenar arquivo

Partially supported by AWS in Education Grant award

Tipos de serviços de arq...

- Modelo de acesso remoto
  - Servidor oferece várias primitivas
    - Cliente pode lidar com frações do arquivo
    - Escrever/ler um bloco específico
  - Vantagens
    - Cliente não precisa armazenar arquivo todo
    - Economia de tempo
  - Desvantagens
    - Necessidade de lidar com concorrência

Partially supported by AWS in Education Grant award

Requisitos de um sistema...

- Transparência de...
  - Acesso: clientes tratam arquivos como locais
  - Localização: espaço de nomes uniforme e sem mudança quando arquivos mudarem
  - Concorrência: operações dos clientes não devem interferir umas com as outras
  - Falha: servidores devem operar normalmente na falha dos clientes, e vice-versa
  - Desempenho: não deve variar com a carga

Partially supported by AWS in Education Grant award

Requisitos de um sistema...

- Usabilidade depende de...
  - Heterogeneidade de HW e SW
    - Interfaces definidas de forma que possam ser implementadas por vários HW e SW
  - Escalabilidade
    - Serviço deve ser extensível para acomodar mudanças de escala do SD

Partially supported by AWS in Education Grant award

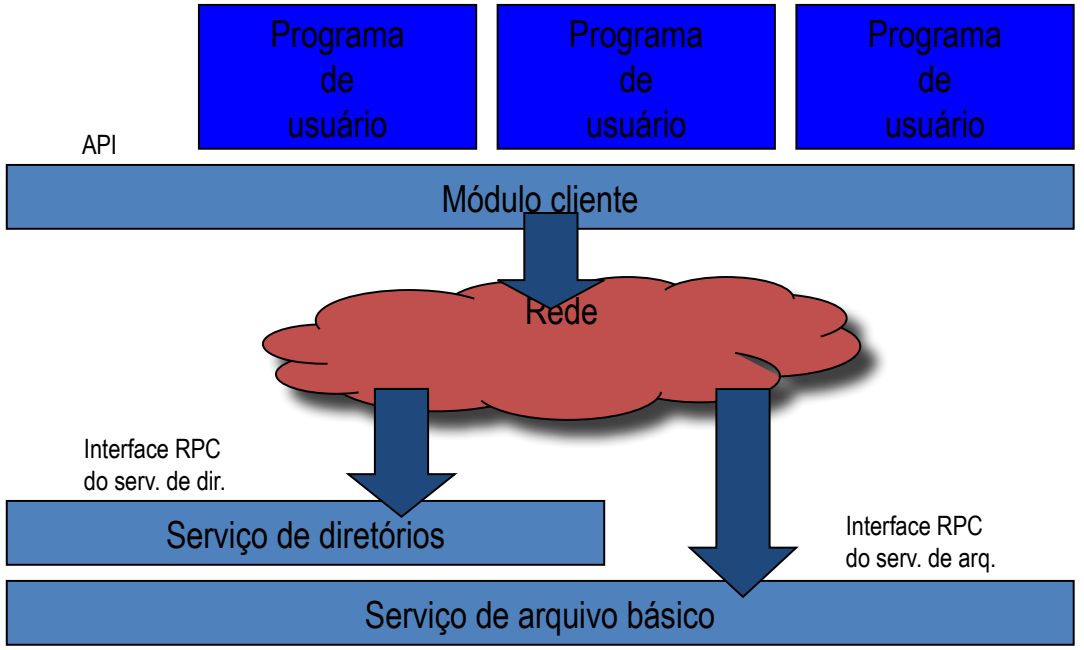
Requisitos de um sistema...

- Se escalabilidade envolver muitos hosts...
  - Transparência de replicação
    - Arquivos podem estar fisicamente replicados e cliente ignora esse fato
  - Transparência de migração
    - Arquivos podem mudar de lugar e isso não deve alterar os clientes

Partially supported by AWS in Education Grant award

Aspectos de projeto

- Para preencher os requisitos, ajuda se o SA for implementado com três componentes (Coulouris)
  - Serviço de arquivo básico
  - Serviço de diretório
  - Módulo de cliente
- Quem faz o que?



Partially supported by AWS in Education Grant award

Serviço de arquivo básico

- Implementa operações nos arquivos
- Arquivos são referenciados por seus FIDs
  - Únicos no SD
- Criação de um novo arquivo?
  - Gera um FID e devolve ao requisitante

Partially supported by AWS in Education Grant award

Serviço de diretório

- Responsável por converter nomes textuais em FIDs
- Diretório
  - Conjunto de FIDs e nomes textuais
- Cliente do serviço de arquivo básico
  - Arquivos de diretório são arquivos, e gerenciados pelo SAB
  - Hierarquia de diretórios

Partially supported by AWS in Education Grant award

Módulo de cliente

- Chamadas de sistema para manipulação de arquivos
  - Criar, ler, escrever...
- Armazena localização na redes do serviço de diretório e de arquivo básico
- Pode gerenciar cache local

Partially supported by AWS in Education Grant award

Interfaces: SAB

- **Read**(File, i, n) → (Data) – reports(BadPosition)
  Lê File, começando em i, por n blocos retornando em Data. Se i maior que tamanho do arquivo, retorna BadPosition
- **Write**(File, i, Data) – reports(BadPosition)
- **Create**() → File
  Cria um novo arquivo de tamanho 0 e retorna um novo FID
- **Truncate**(File, l)
  Reduz o tamanho do arquivo para l
- **Delete**(File)
- **GetAttributes**(File) → Attr
- **SetAttributes**(File, Attr)

Partially supported by AWS in Education Grant award

Interfaces: SAB

- Aspectos dessa interface
  - Não tem open/close; manipula usando FID
  - Com exceção de Create, operações idempotentes
    - RPC com semântica at-least-once
  - Servidor sem estado
    - Clientes são responsáveis por saber onde estão lendo/escrevendo
    - No caso de falha, depois do retorno não há procedimento especial

Partially supported by AWS in Education Grant award

Tipos de serviços de arq...

- Stateless
  - Servidor não guarda informação sobre clientes nem sobre arquivos (FIDs)
  - Responsabilidade do cliente em manter o estado
- Stateful
  - Manter informação de qual cliente abriu qual arquivo, e em que bloco está
  - Requisições são atendidas mais rápido, pois não precisa localizar o FID do arquivo

Partially supported by AWS in Education Grant award

Interfaces: diretório

- **Lookup**(Dir, Name, AccessMode, userID) → (File)
  – reports(NotFound, NoAccess)
- **AddName**(Dir, Name, File, userID) – reports(NameDup)
  Adiciona o arquivo File ao diretório Dir usando o nome Name
- **UnName**(Dir, Name) – reports(NotFound)
  Remove Name do diretório Dir
- **ReName**(Dir, OldName, NewName) – reports(NotFound)
- **GetNames**(Dir, Pattern) → NameSeq

Partially supported by AWS in Education Grant award

Interfaces: diretório

- Para montar uma estrutura hierárquica
  - Cada diretório tem o nome de diretórios e arquivos acessíveis
  - Estrutura pode ser montada no cliente
  - Raiz com FID bem conhecido (e.g., 0)

Partially supported by AWS in Education Grant award

Geração dos FIDs

- Devem ser gerados de forma única no espaço/tempo e difícil de forjar
  - Arquivo removido, FID descartado
  - Deve incorporar permissões de acesso (viram capabilities, ou capacidades)
- FID não é endereço
  - Não contém informações sobre localização do arquivo
  - Alguma forma de mapear FIDs em blocos de disco

Partially supported by AWS in Education Grant award

Generalização

- Organização geral do SA

Máquina servidora	Serviço de arquivos e/ou diretório
	Interface de cliente
Rede	Sistema de comunicação
Máquina cliente	Cliente
	Aplicação de usuário

Partially supported by AWS in Education Grant award

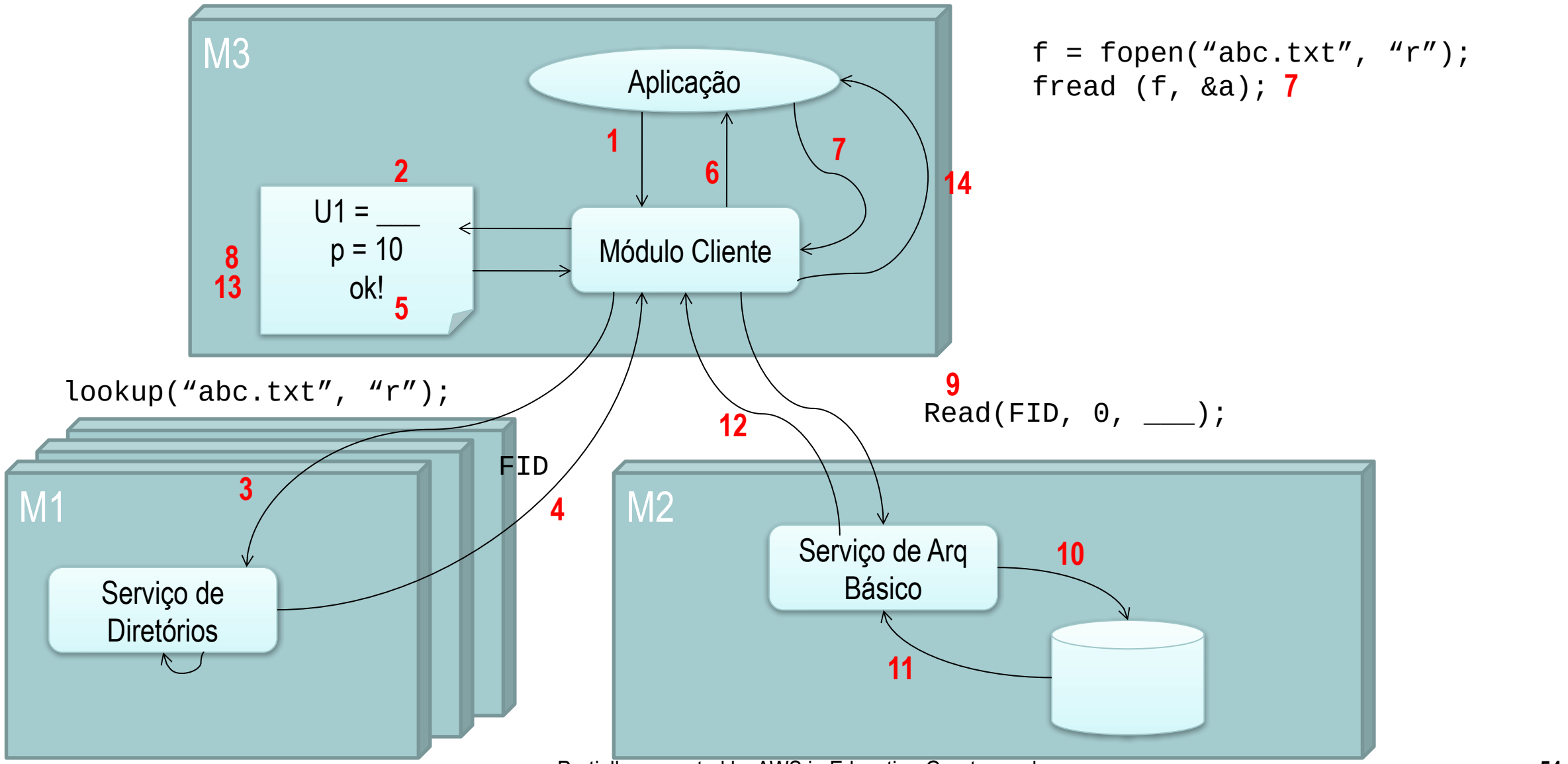
Generalização

Máquina servidora	Serviço de arquivos e/ou diretório
	Interface de cliente
Rede	Sistema de comunicação
Máquina cliente	Cliente
	Aplicação de usuário

- Serviço de arquivos e/ou diretório
  - Armazena e gerencia arquivos/diretórios
- Interface de cliente
  - Especificação das primitivas
- Cliente
  - Implementação da parte cliente da interface
- Aplicação

Partially supported by AWS in Education Grant award

Generalização



Partially supported by AWS in Education Grant award

Semânticas

- Quando há mais de um processo lendo/ escrevendo em um arquivo
  - Controle de concorrência (ok!)
  - Semântica de compartilhamento
- Quatro tipos básicos
  - Semântica UNIX
  - Semântica de sessão
  - Arquivos imutáveis
  - Transações

Partially supported by AWS in Education Grant award

Semântica UNIX

- Quando um READ segue um WRITE, o READ retorna o valor que acabou de ser escrito
  - Obrigatoriedade de ordenação total
  - Sempre retorna o valor mais atual
- Implementação trivial
  - Servidor central que processa requisições na ordem (lógica)

Partially supported by AWS in Education Grant award

Semântica de sessão

- Servidor centralizado trás problemas de gargalo e ponto central de falha
- Gargalo pode ser aliviado com uso de caches nos clientes
  - C1 lê o arquivo (coloca no cache) e altera
  - C2 lê o arquivo → dados antigos
- Solução → propagar alterações
  - Conceitualmente simples
  - Ineficiente

Partially supported by AWS in Education Grant award

Semântica de sessão

- Relaxamento na semântica UNIX
  - Mudanças em um arquivo são visíveis imediatamente para o processo, e para os demais apenas quando o arquivo for fechado
- Está errado?
  - Não
  - Amplamente implementado

Partially supported by AWS in Education Grant award

Arquivos imutáveis

- Por que se preocupar?
- Arquivos não podem ser alterados
  - Apenas READ e CREATE
- Alterar um arquivo?
  - Não pode, mas pode criar um novo de forma atômica
  - Arquivos são imutáveis, mas diretórios não

Partially supported by AWS in Education Grant award

Arquivos imutáveis

- Dois processos desejando “alterar”
  - O mais novo altera o mais velho
  - Forma não determinística
- Se for alterar com alguém lendo?
  - Criar uma cópia do antigo para continuar lendo
  - Retornar erro para o leitor

Partially supported by AWS in Education Grant award

Transações

- Tratar operações de forma atômica
  - Uma transação
- Dois processos ao mesmo tempo?
  - Transações permitem concorrência
  - Sistema faz serialização

Partially supported by AWS in Education Grant award

Semânticas

- São apenas diretrizes para construção
  - “Meu SA tem semântica Unix”
- Cabe ao projetista decidir, considerando...
  - Desempenho
  - Comportamento esperado

Partially supported by AWS in Education Grant award

Replicação

- SDs geralmente oferecem alguma forma de replicação de dados
  - Múltiplas cópias do mesmo objeto
- Por que?
  - Aumentar confiabilidade
  - Permitir acesso mesmo na caso de falha
  - Divisão de carga entre servidores
- Principal problema: transparência (!)

Partially supported by AWS in Education Grant award

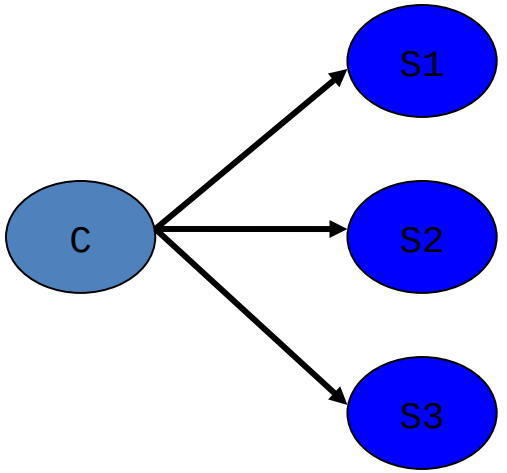
Replicação

- Quanto o usuário sabe da replicação?
  - Sabe e pode até manipular
  - Sistema faz tudo, sozinho → transparente
- Replicação pode ser feita, basicamente, de três formas
  - Replicação explícita
  - Replicação atrasada
  - Replicação usando grupo

Partially supported by AWS in Education Grant award

Replicação explícita

- Programador faz todo o trabalho sujo
- Serviço de diretório pode permitir múltiplos FIDs por arquivo
  - Recupera todos no lookup
  - Quando for manipular, tenta seqüêncialmente um por um dos FIDs, até conseguir
- Funciona, mas é muito trabalhoso, e nada transparente

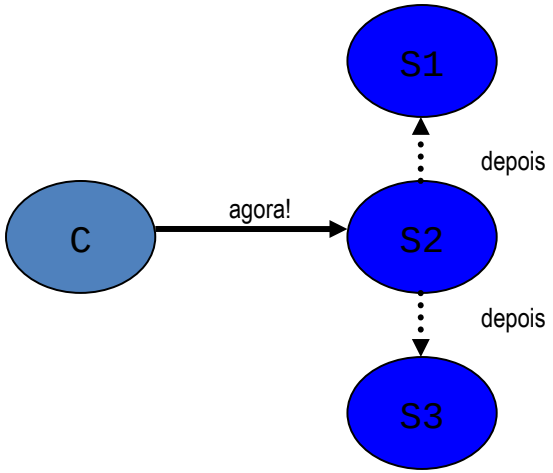


Arquivo1	1.14	2.41	3.56
Prog.c	1.32	2.56	3.23

Partially supported by AWS in Education Grant award

Replicação atrasada

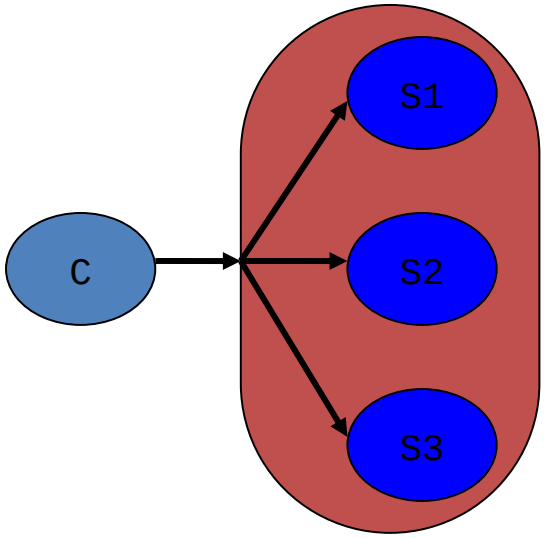
- Apenas uma cópia é feita em um servidor
- Servidor é responsável por...
  - Propagar atualizações
  - Selecionar outro servidor para atender, se ele não puder



Partially supported by AWS in Education Grant award

Replicação usando grupo

- Operação de escrita é feita ao mesmo tempo em todos
  - Multicast atômico
- Atrasada x grupo
  - Um servidor x grupo
  - Segundo plano x atômica



Partially supported by AWS in Education Grant award

Protocolos de atualização

- Até agora, só resolvemos o processo de criação
- Atualização não é tão simples
  - Mensagem de update para cada cópia
  - Se o processo coordenador falhar, cópias não alteradas → inconsistências
- Duas formas
  - Replicação de cópia primária
  - Algoritmos de votação

Partially supported by AWS in Education Grant award

Replicação de cópia primária

- Um servidor é eleito primário
  - Outros são secundários
- Atualização
  - Entregue para o primário
  - Atualizada cópia local
  - Grava em memória estável
  - Notifica secundários

Partially supported by AWS in Education Grant award

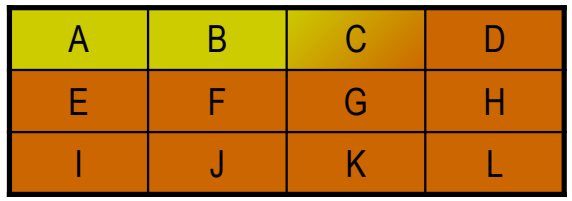
Replicação de cópia primária

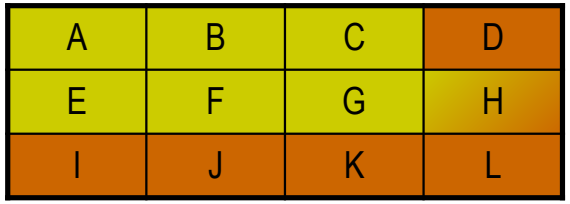
- No caso de falha
  - Primário entra no ar
  - Recupera da memória estável as atualizações
  - Continua a atualizar
- Leitura pode ser feita de qualquer um
- Problema
  - Ponto único de falha

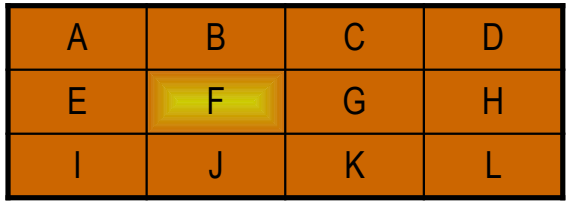
Partially supported by AWS in Education Grant award

Algoritmos de votação

- Gifford formalizou em 1979
  - Para ler, necessário quorum de leitura
    - Pelo menos N_R servidores
  - Para escrever, quorum de escrita
    - Pelo menos N_W servidores
  - Restrição: $N_R + N_W > N$







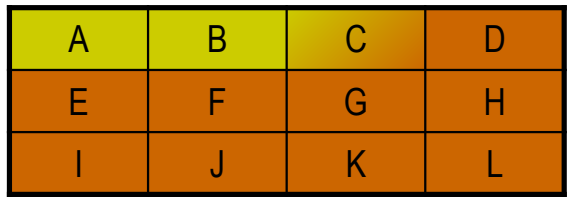
Leitura

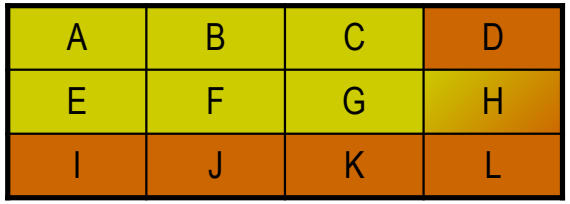
Escrita

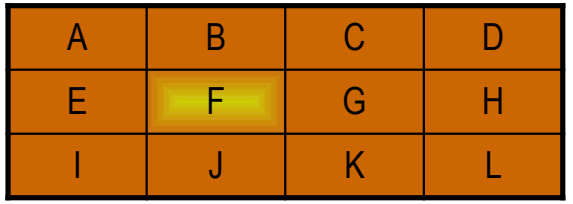
Partially supported by AWS in Education Grant award

Algoritmos de votação

- Quoruns não precisam conter os mesmos membros sempre
  - N_W = 10 servidores (C a L)
  - N_R = 3 → pelo menos 1 vai coincidir!
    - Escolhe A a C
  - Cliente recupera a versão mais nova das três
    - Participantes se sincronizam pela mais nova







Leitura

Escrita

Partially supported by AWS in Education Grant award

Estudos de caso OceanStore e Farsite

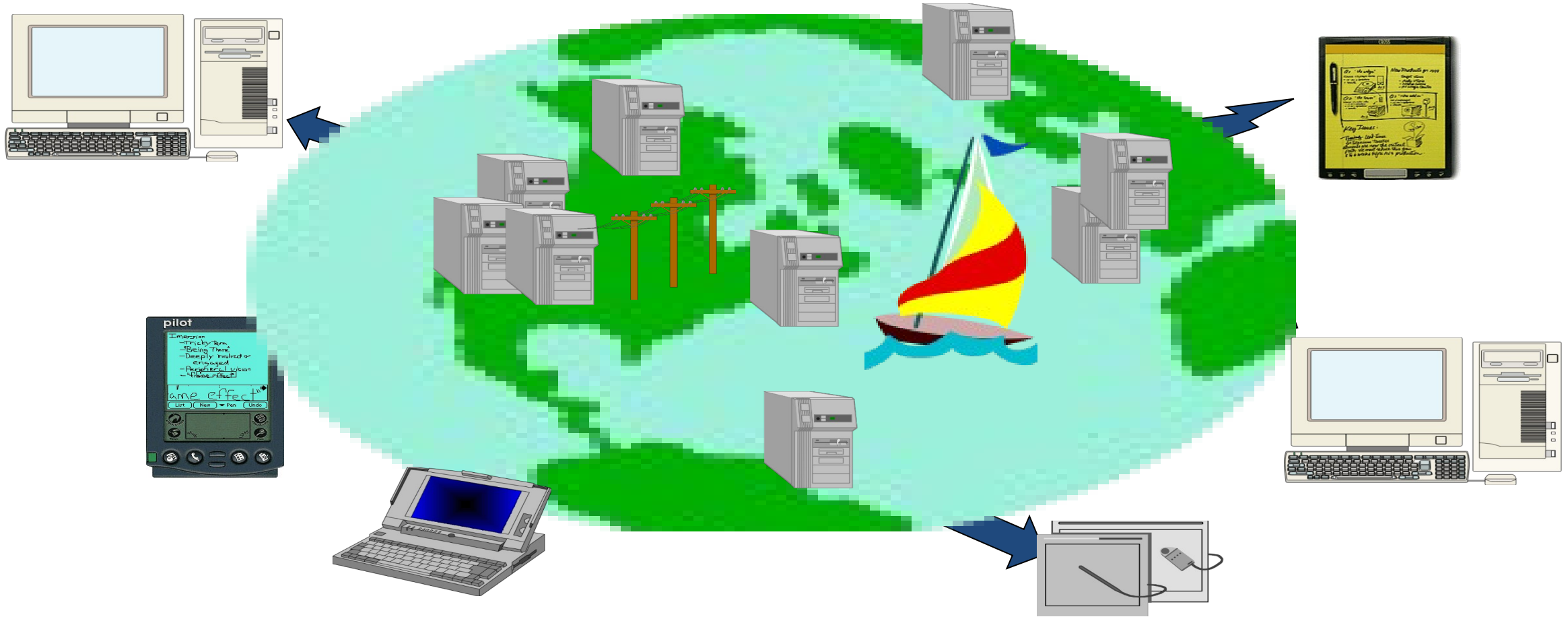
“O que eu faço com tudo isso que vi até agora?”

OceanStore

- Sistema de armazenamento persistente de escala global
  - Servidores não-confiáveis
  - Dados nômades
- Capaz de gerenciar até 10^{14} arquivos
- Usado para computação ubíqua
  - Computador em todo o lugar
  - Relógio de pulso não tem capacidade de armazenamento → OceanStore

Partially supported by AWS in Education Grant award

OceanStore



Partially supported by AWS in Education Grant award

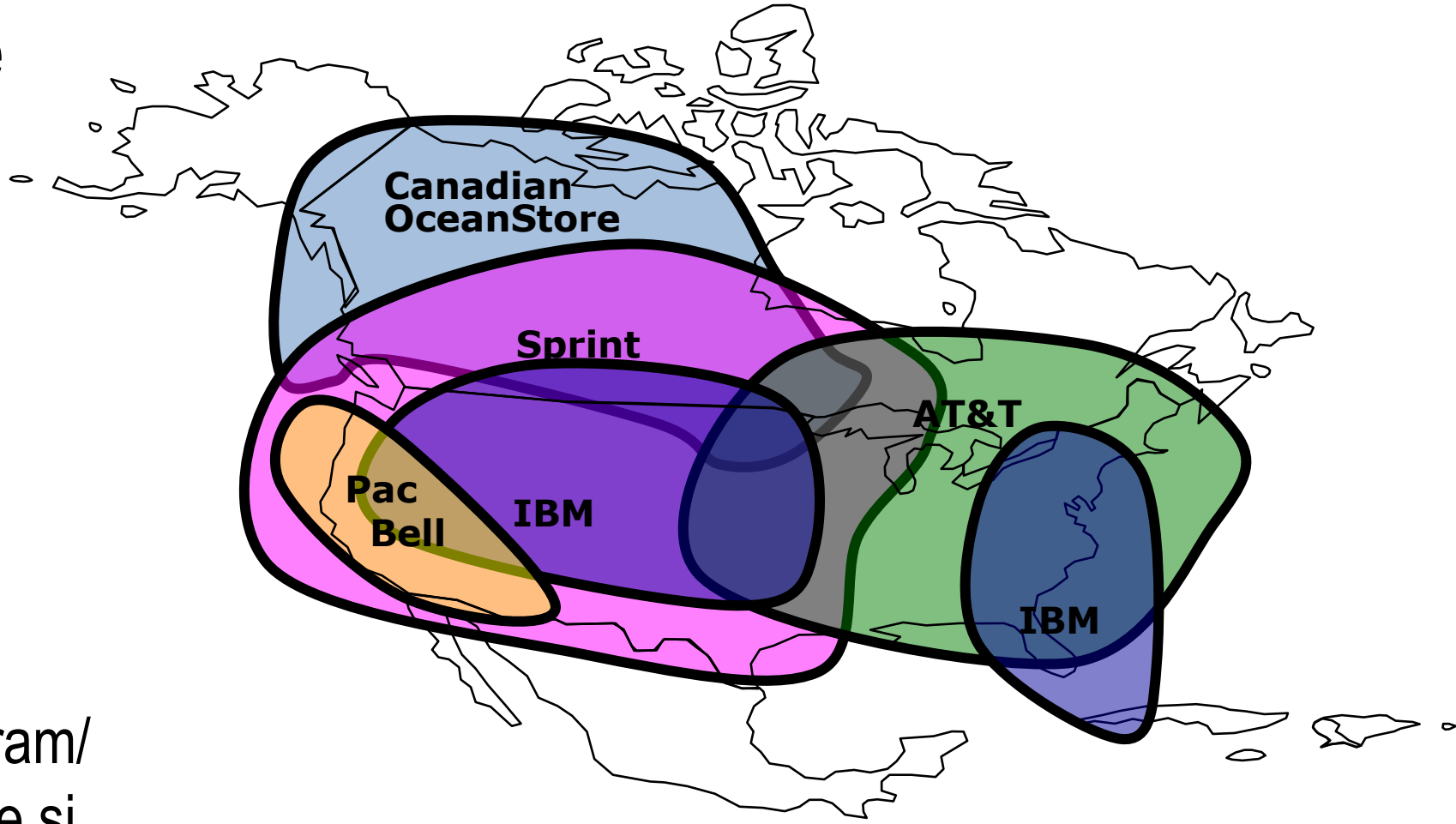
OceanStore

- Onde a informação está?
  - Geograficamente distribuída → disponibilidade
- Como está protegida?
  - Criptografia e assinatura digital
- Como ela é indestrutível
  - Redundância com redistribuição e reconstrução
- Como é gerenciada?
  - Sozinha!
- Quem oferece espaço
  - Infraestrutura de utilidade (provedores de espaço)

Partially supported by AWS in Education Grant award

OceanStore

- Infraestrutura de utilidade
  - Federações
  - Taxa mensal paga a um forne-cedor
  - Fornecedores compram/ vendem espaço entre si



Partially supported by AWS in Education Grant award

OceanStore assume que...

- Infraestrutura não confiável
  - Apenas conteúdo cifrado no OceanStore
- Alguém é responsável
  - Pela durabilidade e consistência, sem garantir privacidade
- Grande parte está bem conectada
  - Rede de alto desempenho entre produtor/ consumidor
  - Usar multicast para consistência quando puder

Partially supported by AWS in Education Grant award

OceanStore assume que...

- Cache promíscuo
  - Dados podem estar em cache em qualquer lugar, a qualquer momento
- Controle de concorrência otimista
  - Evitar lock a todo custo

Partially supported by AWS in Education Grant award

Como está o OceanStore

- “Will the pond become an ocean?”
  - Replicando
  - Localizando
  - Fazendo manutenção
  - Fragmentando
- Parte do projeto Endevour
  - Tornar a computação mais acessível
- http://oceanstore.cs.berkely.edu/
- http://bamboo-dht.org/

Partially supported by AWS in Education Grant award

Farsite

- Federated, Available, and Reliable Storage for an Incompletely Trusted Environment
  - Sistema de arquivos distribuído sem servidor
  - Todas as estações cooperam no armazenamento
- Aspecto chave
  - Participantes não são confiáveis
    - Podem ser formatados
    - Podem ser invadidos e dados corrompidos

Partially supported by AWS in Education Grant award

Farsite

- Tudo o que o Farsite oferece pode ser obtido com servidor central, mas...
  - E/S de alto desempenho
  - Matrizes de discos
  - Manutenção
  - Gargalo
  - Ponto central de falha

Partially supported by AWS in Education Grant award

Farsite

- Clientes cooperam e definem
  - Quem pode armazenar
  - Como dividir o espaço
- Várias réplicas
  - Fragmentação
  - Aumento do espaço necessário (redundância)

Partially supported by AWS in Education Grant award

Farsite

- Ainda (e ficará) sem implementação real
- Alguns estudos mostravam que...
  - Cada máquina serve 10Mb/dia em cache
  - Cada máquina recebe tráfego médio de 7Mb/hora/réplica
  - Razoável e manipulável pelos PCs da época
- Desenvolvido na Microsoft
  - http://research.microsoft.com/en-us/projects/farsite/default.aspx

Partially supported by AWS in Education Grant award

OceanStore e Farsite

- Exploram Lei de Moore
  - Capacidade de armazenamento crescente
  - Custo decrescente
  - Largura de banda da rede aumentando
  - Aumento de desempenho das estações
- Diferença básica
  - Escala (global x regional)

Partially supported by AWS in Education Grant award

Tahoe

- “Sistema de arquivos com autoridade mínima”
  - Usuários tem o mínimo de autorização necessária para fazer o que precisam
  - Arquivos mutáveis e imutáveis
  - Armazenamento/recuperação distribuídos e tolerante a faltas
- Três componentes-chave
  - Storage nodes
  - Client nodes
  - Introducers

Partially supported by AWS in Education Grant award

Tahoe e capabilities

- Arquivos imutáveis
  - Read capability
  - Verify capability
- Arquivos mutáveis
  - Read-only capability
  - Read-write capability
  - Verify capability

Partially supported by AWS in Education Grant award

Instalação e casos de uso

- Cluster local
- Acesso via FUSE, Web, FTP, SFTP e CLI

Nome	# típico de nós	Domínios	Capacidade dos nós	Disponibilidade dos nós	Churn
non-RAID	1 host, múltiplos ós	1	Diversa	Uniforme	Baixo
Friendnet	2 a 10	Vários, todos confiáveis	Variada	Variada	Baixo
Grid particular	3 a 1k servidores, até 50k clientes	1 para servidores, vários para clientes	Uniforme	Alta	Baixo
Hivecache (Farsite)	10 a 1k	1	Alguma uniformidade	Alta	Baixo
Grid global	Qualquer	Muitos	Variada	Variada	Alta