ACH2147 - Desenvolvimento de Sistemas de Informação Distribuídos

Aula 02 – Definição e Objetivos dos SDs

Norton Trevisan Roman

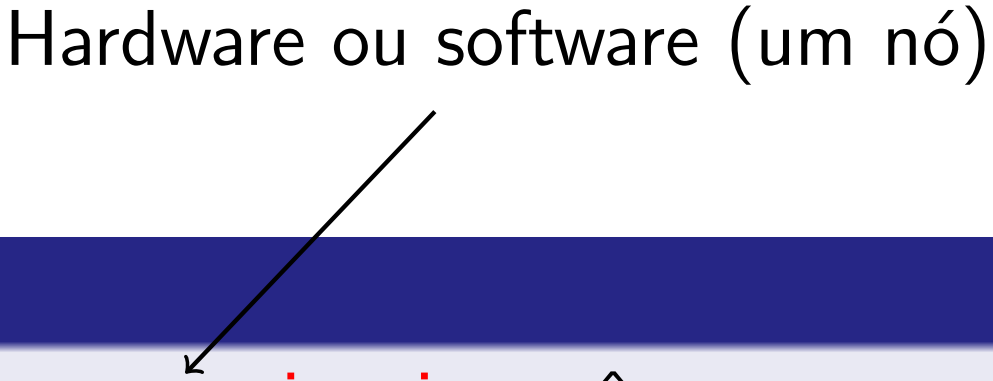
30 de março de 2022

O que é um Sistema Distribuído?

Definição

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

O que é um Sistema Distribuído?



Definição

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

O que é um Sistema Distribuído?

Hardware ou software (um nó)

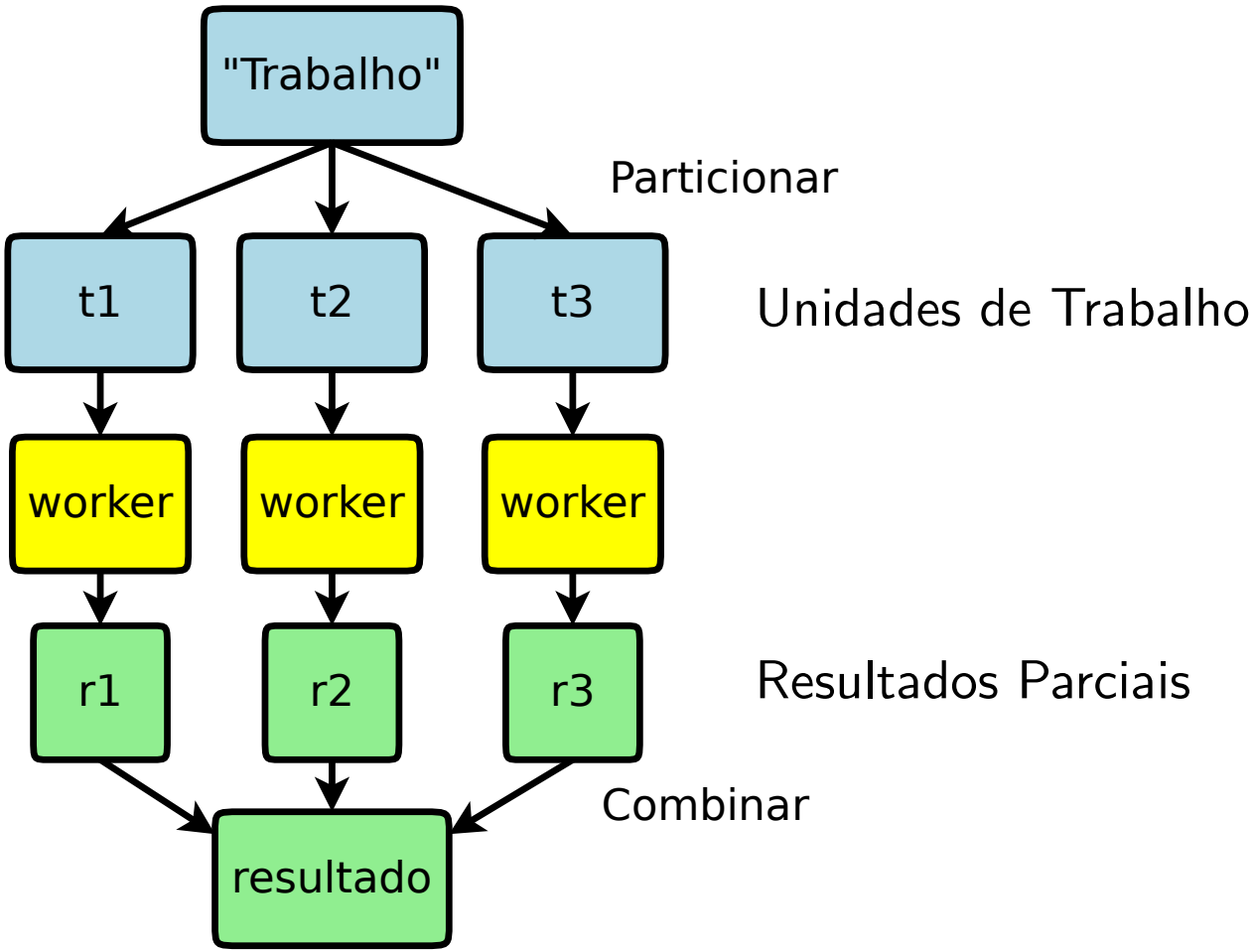
Definição

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Os usuários ou aplicações percebem um único sistema $\Rightarrow$ os nós precisam colaborar

E pra que isso?

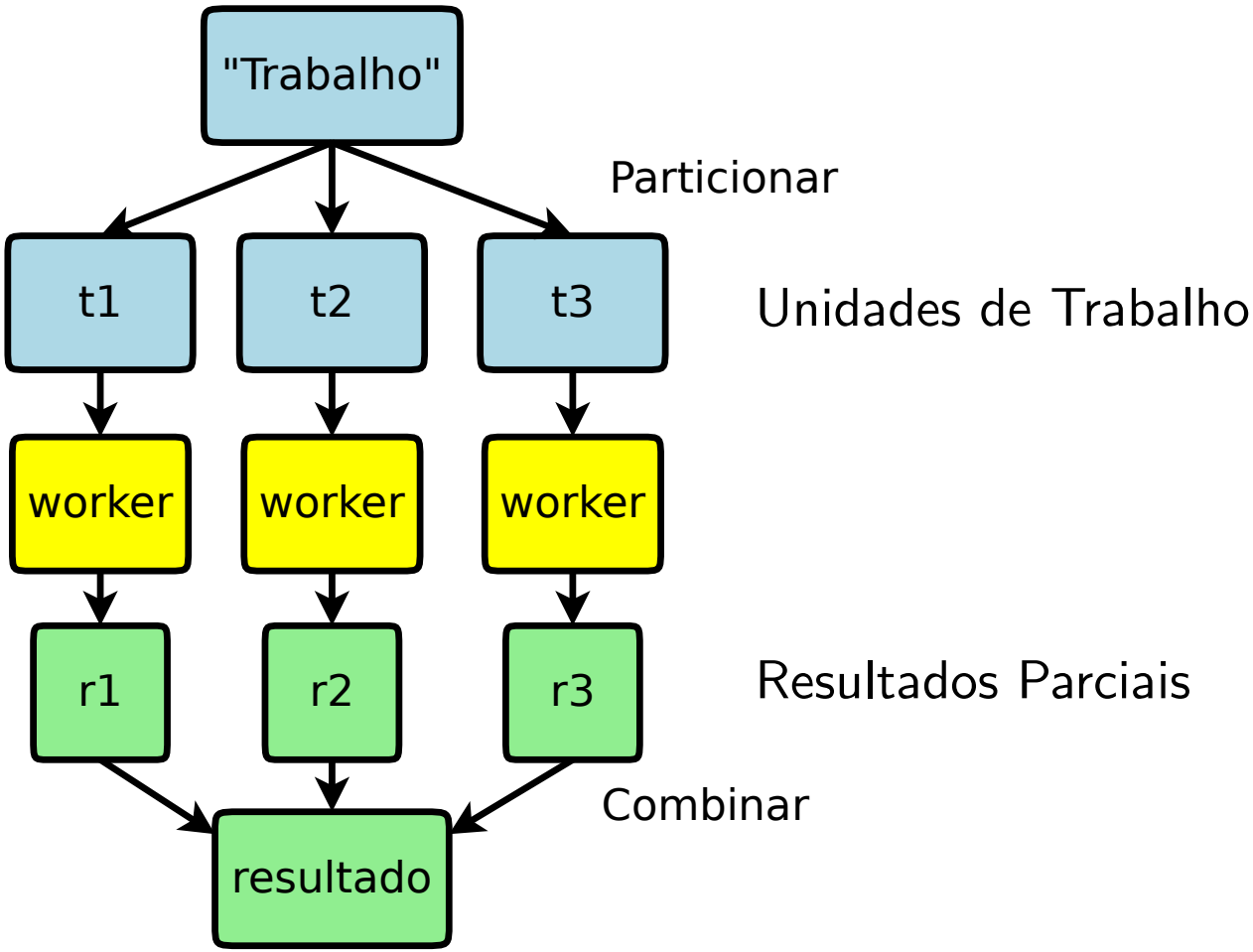
Dividir para Conquistar



E pra que isso?

Escalabilidade horizontal – há limites para processadores e memória

Dividir para Conquistar

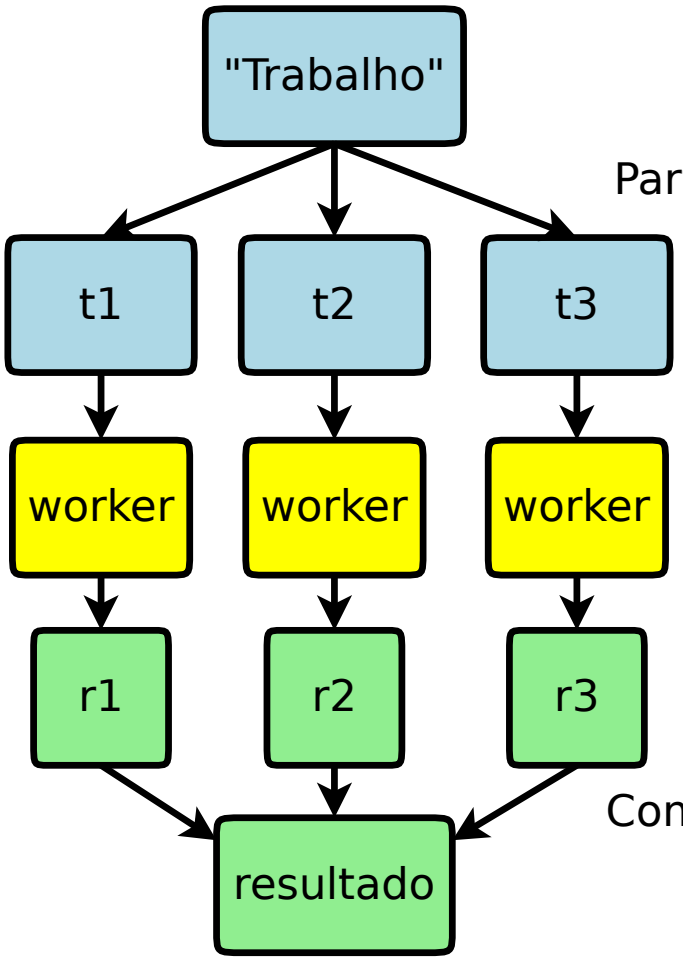


E pra que isso?

Escalabilidade horizontal – há limites para processadores e memória

Mover o processamento para perto dos dados – a banda de rede é limitada

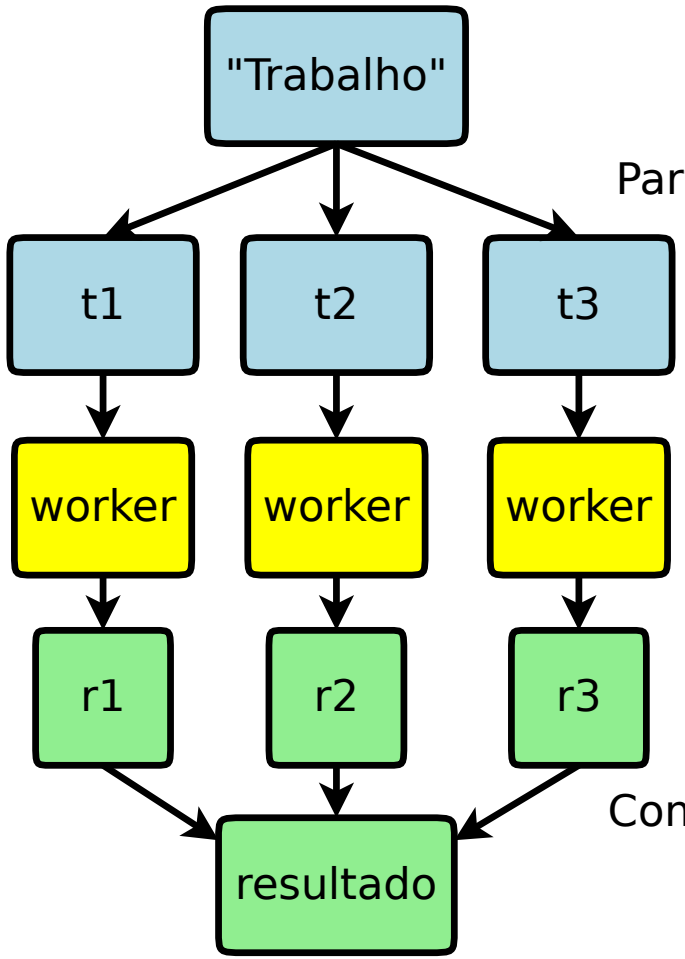
Dividir para Conquistar



Desafios da paralelização

Dividir para Conquistar

Como repartir as unidades de trabalho entre os *workers*?

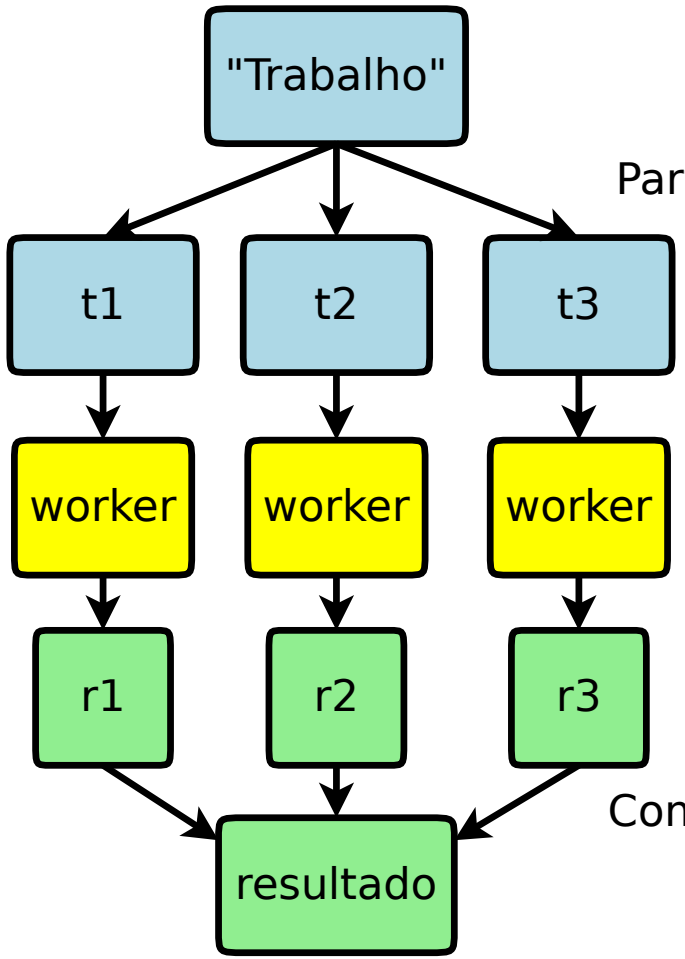


Desafios da paralelização

Dividir para Conquistar

Como repartir as unidades de trabalho entre os *workers*?

O que fazer quando temos mais trabalho do que *workers*?



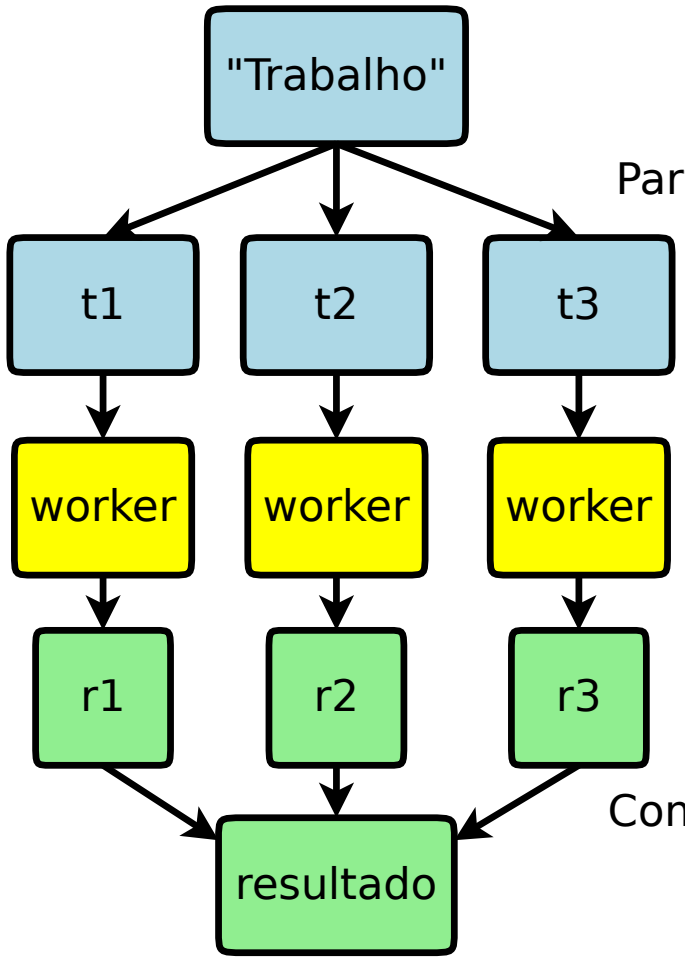
Desafios da paralelização

Dividir para Conquistar

Como repartir as unidades de trabalho entre os *workers*?

O que fazer quando temos mais trabalho do que *workers*?

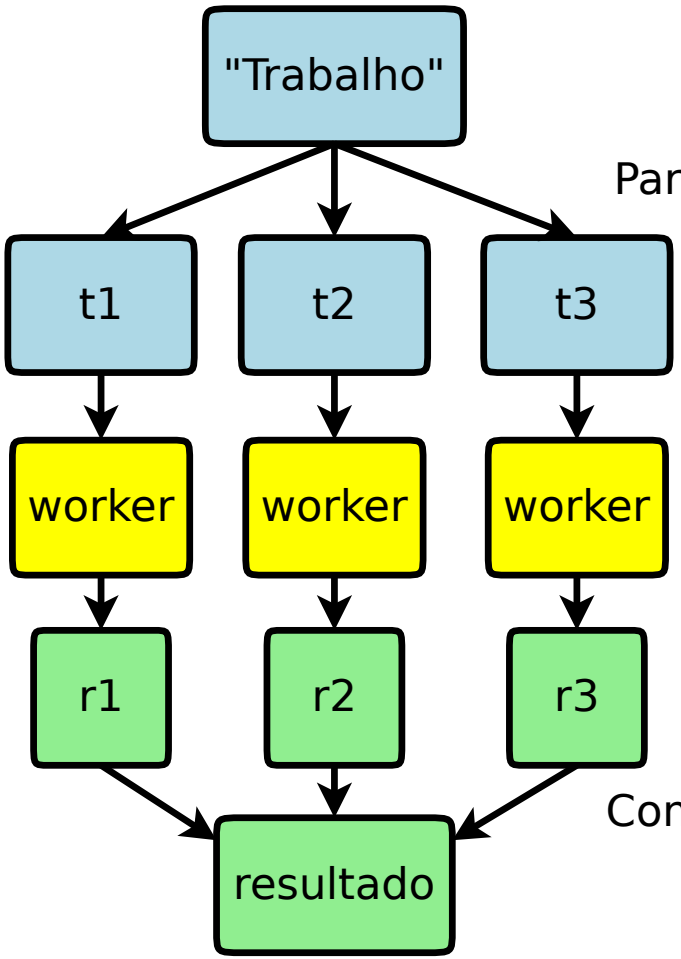
E se os *workers* precisarem compartilhar resultados intermediários entre si?



Desafios da paralelização

Dividir para Conquistar

Como agregar os resultados parciais?

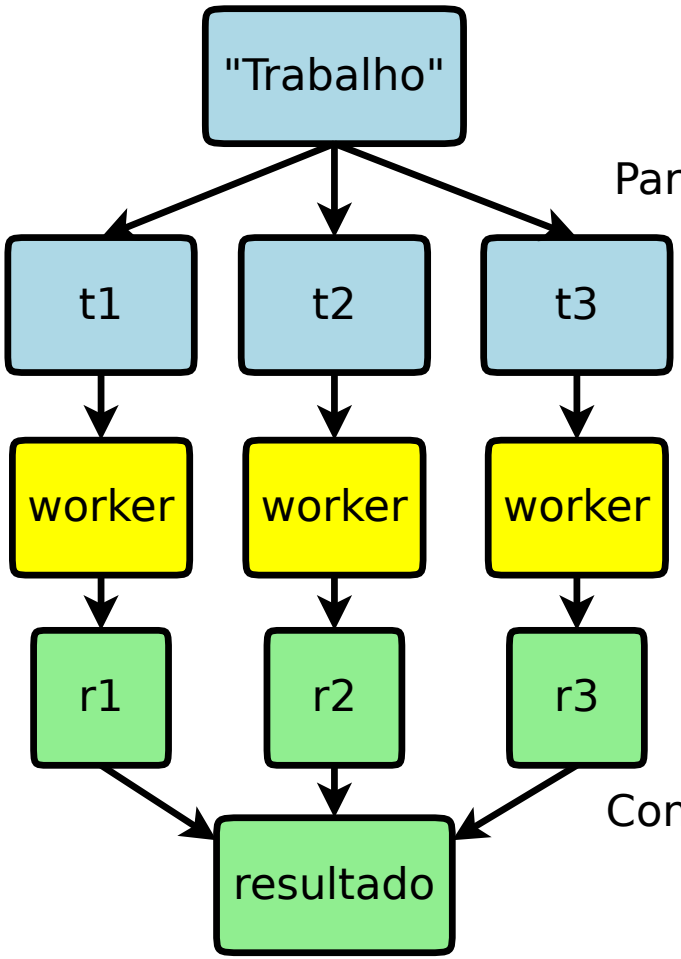


Desafios da paralelização

Dividir para Conquistar

Como agregar os resultados parciais?

O que fazer se um *worker* parar de funcionar?



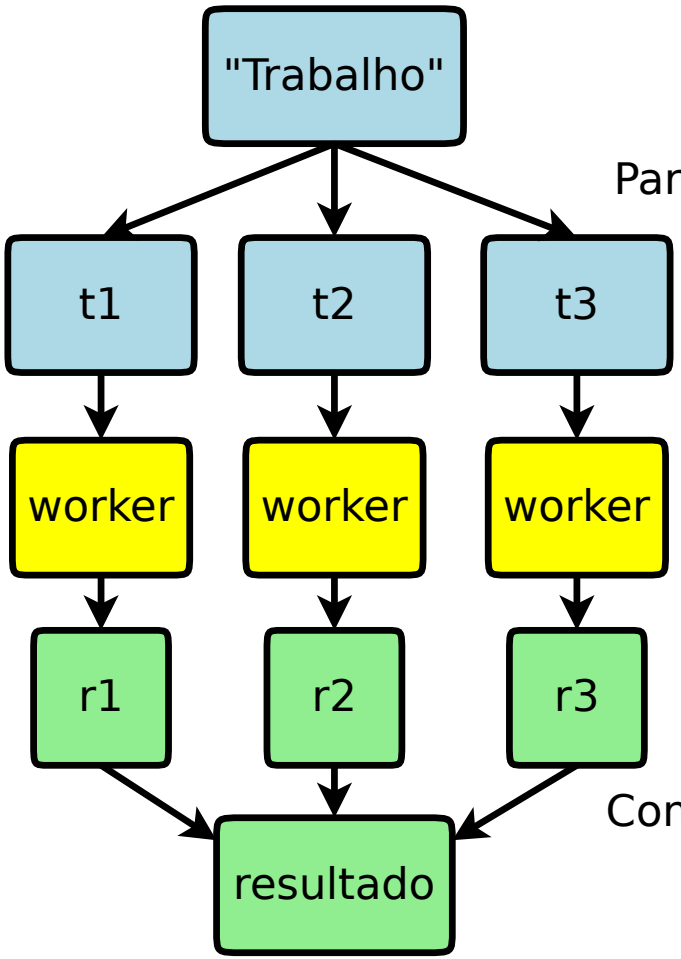
Desafios da paralelização

Dividir para Conquistar

Como agregar os resultados parciais?

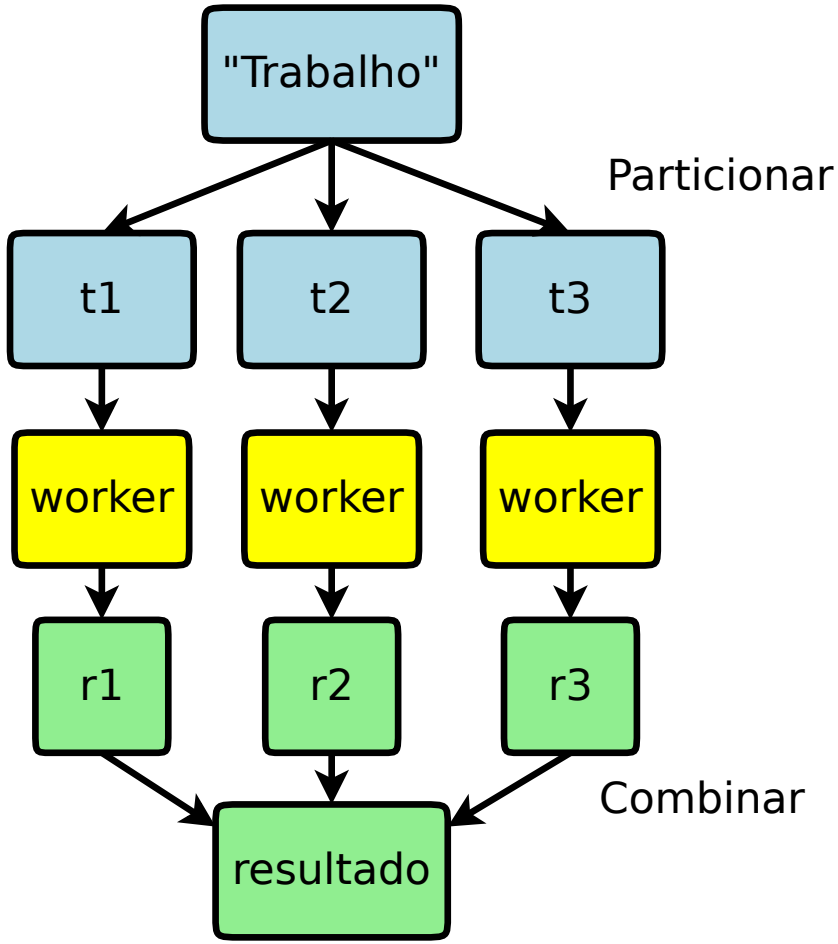
O que fazer se um *worker* parar de funcionar?

Como saber se todos os *workers* terminaram seus trabalhos?



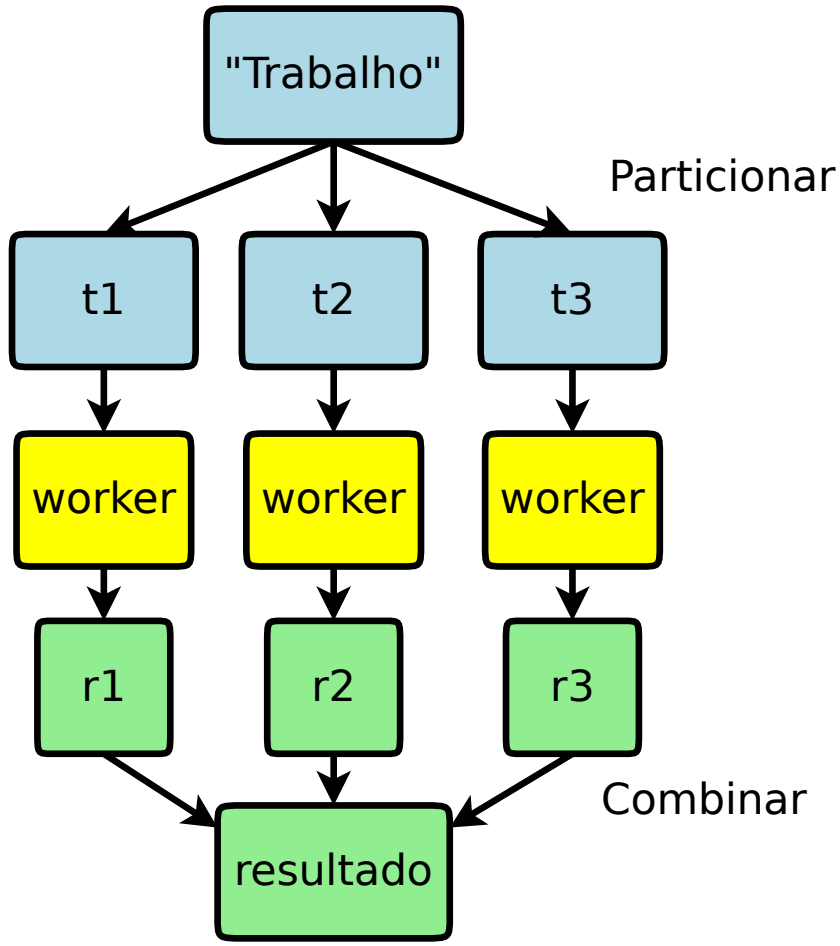
Problemas de paralelização

- Surgem por causa de:
  - Comunicação entre os *workers*
  - Acesso a recursos compartilhados (ex: dados)



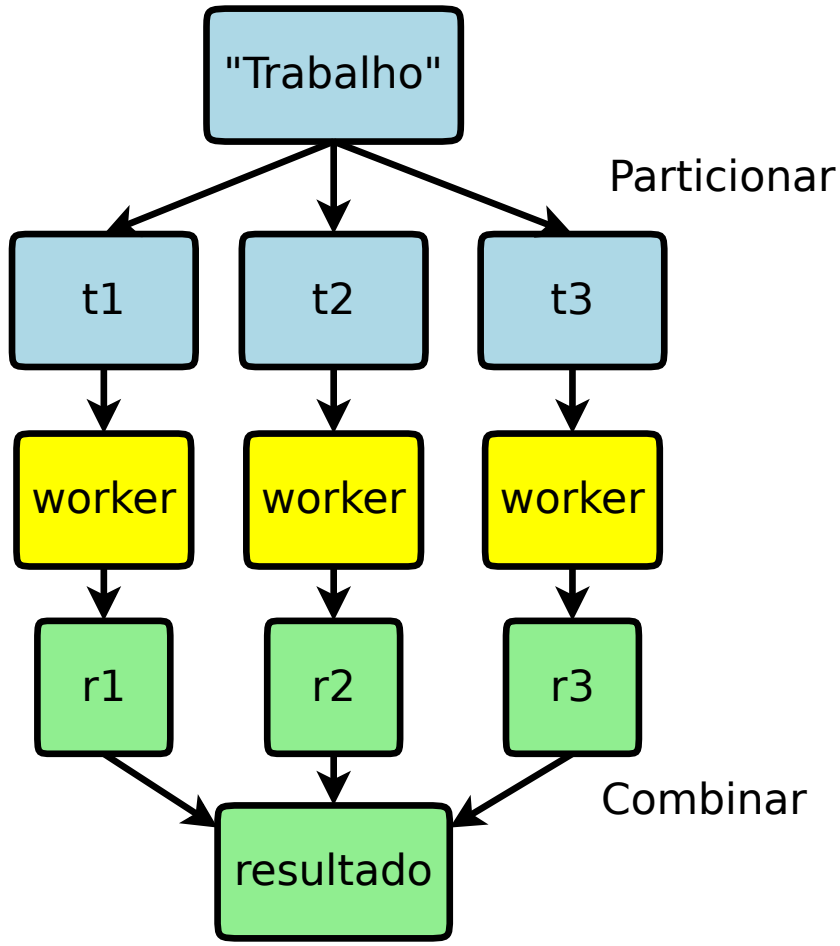
Problemas de paralelização

- Surgem por causa de:
  - Comunicação entre os *workers*
  - Acesso a recursos compartilhados (ex: dados)
- Precisamos de algum mecanismo de sincronização



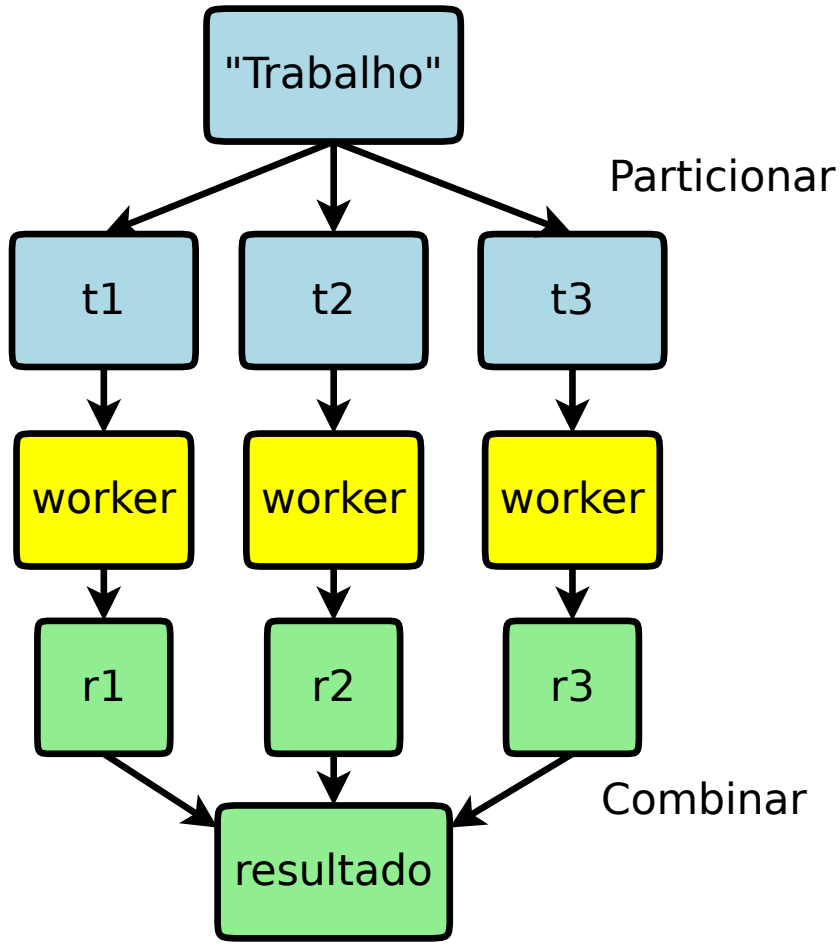
Gerenciamento dos *workers*

- Difícil:
  - Não sabemos em que ordem cada *worker* será executado
  - Não sabemos quando um *worker* irá interromper outro *worker*
  - Não sabemos em qual ordem os *workers* irão acessar os dados compartilhados



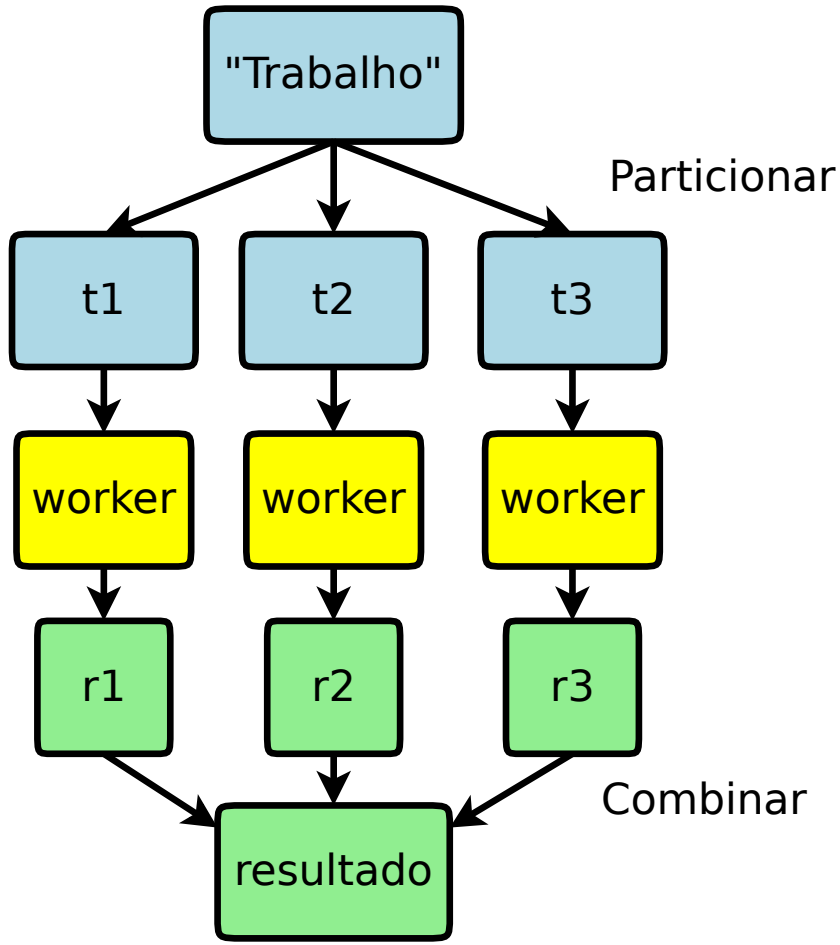
Gerenciamento dos *workers*

- Precisamos de
  - Semáforos (`up`, `down`)
  - Variáveis condicionais (`wait`, `notify`, `broadcast`)
  - Barreiras de sincronização



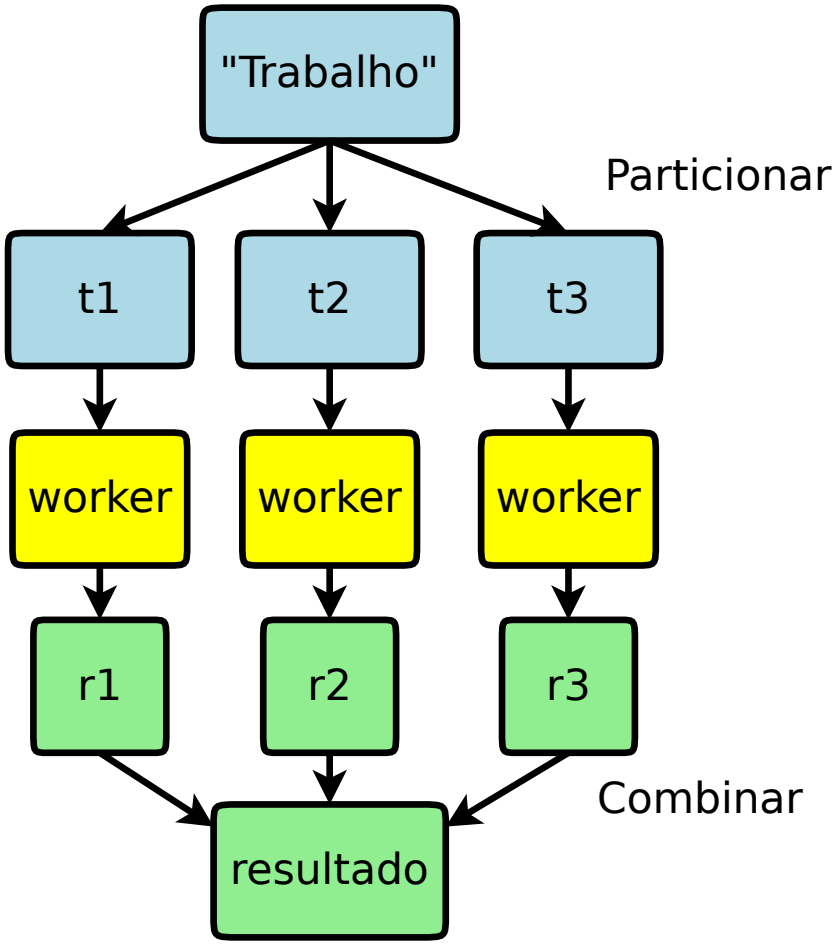
Gerenciamento dos *workers*

- Precisamos de
  - Semáforos (`up`, `down`)
  - Variáveis condicionais (`wait`, `notify`, `broadcast`)
  - Barreiras de sincronização
- Ainda assim...
  - Poderemos ter deadlocks, starvation, race coditions ...



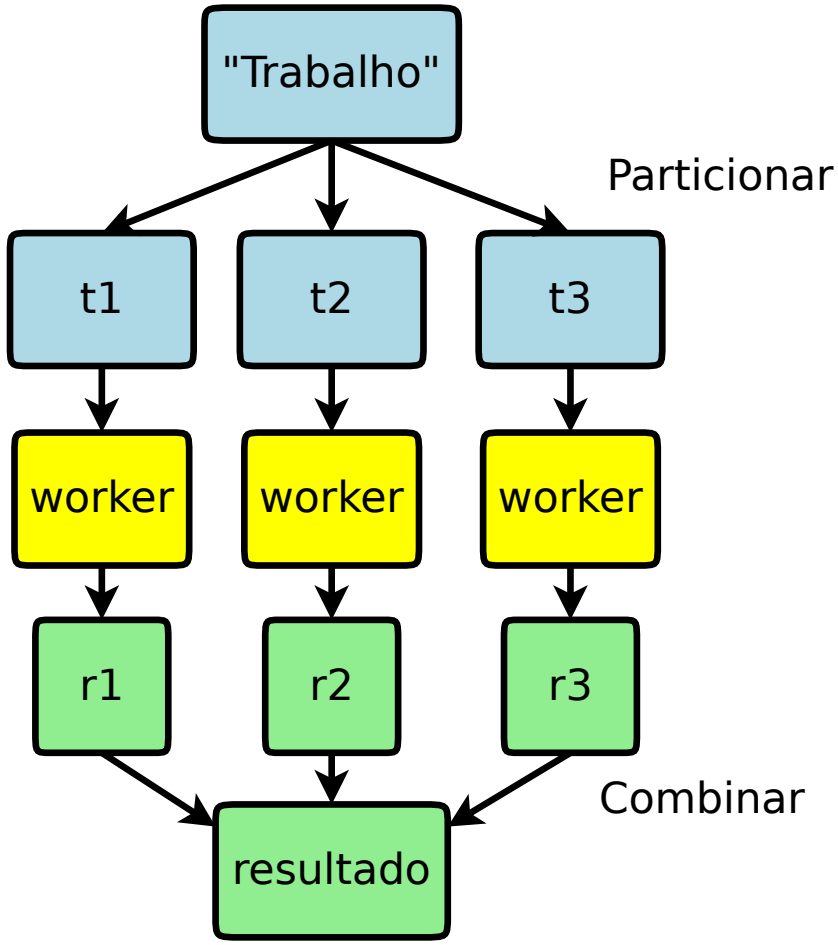
Ferramentas atuais

- Modelos de programação:
  - Memória compartilhada (*threads*)
  - Passagem de mensagens



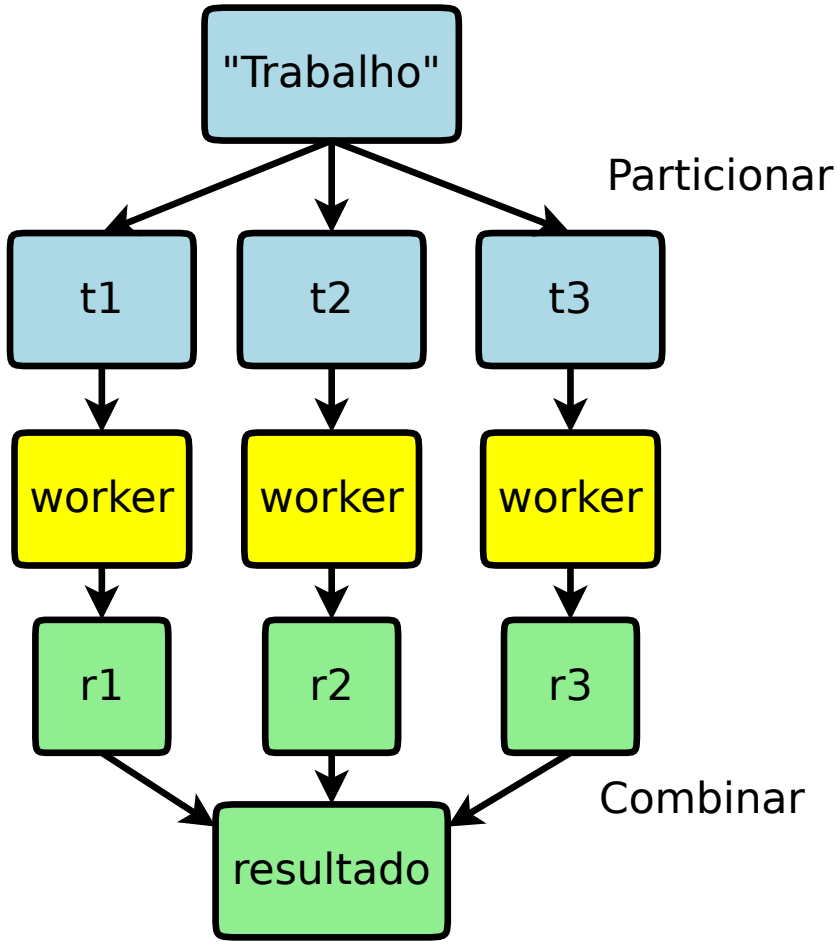
Ferramentas atuais

- Modelos de programação:
  - Memória compartilhada (*threads*)
  - Passagem de mensagens
- Padrões arquiteturais:
  - Mestre-escravo
  - Produtor-consumidor
  - Filas de trabalho compartilhadas



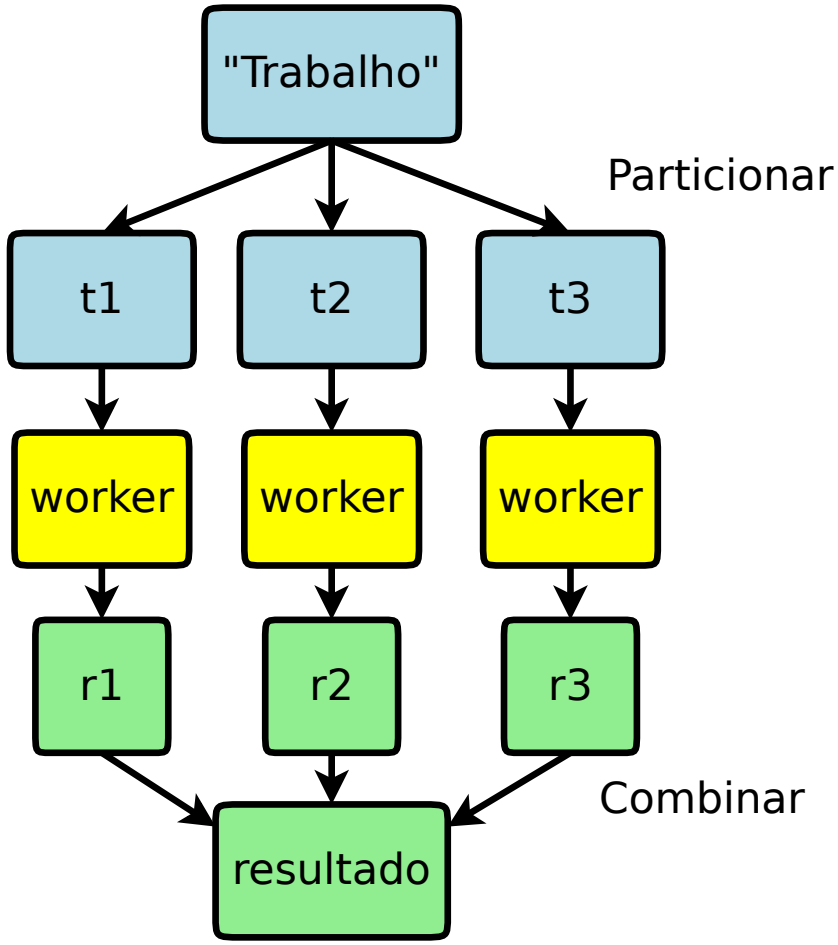
Em suma...

- Tudo se resume ao nível mais adequado de abstração



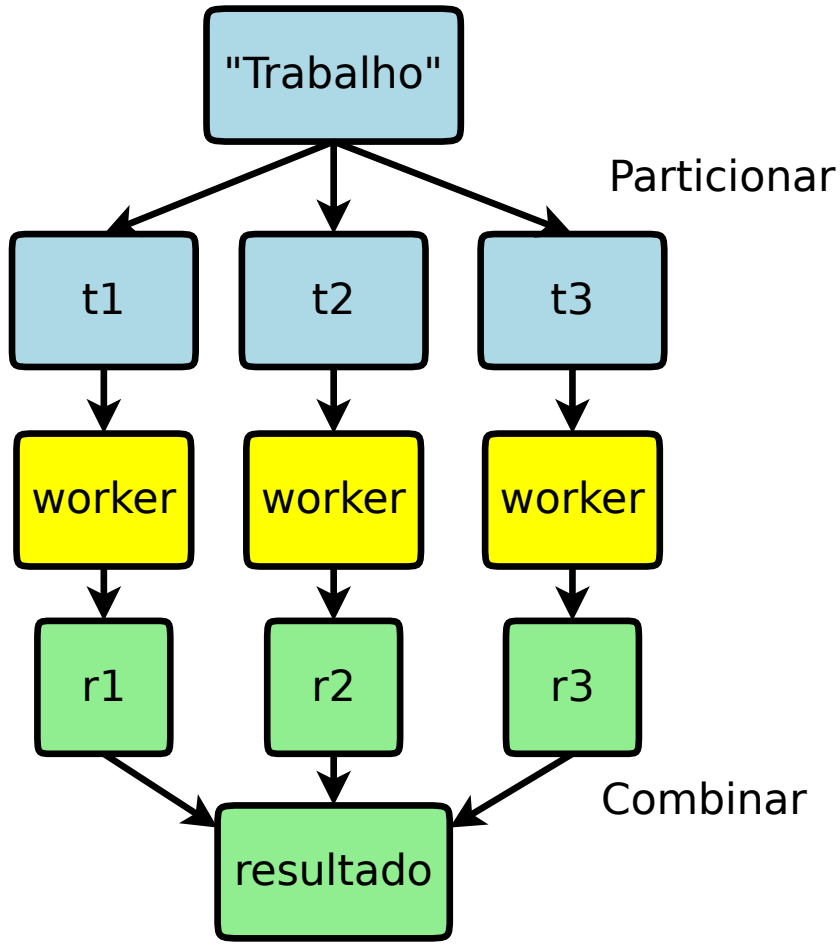
Em suma...

- Tudo se resume ao nível mais adequado de abstração
- Esconda os detalhes do sistema dos desenvolvedores
  - Evita os problemas com race conditions, trava em locks, etc.



Em suma...

- Separe o “quê” do “como”:
  - O desenvolvedor especifica apenas o que deve ser computado
  - O arcabouço deve se encarregar de como realizar a execução



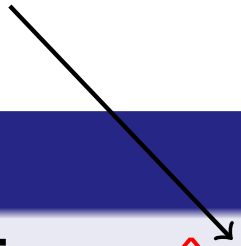
Voltemos à definição...

Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Voltemos à definição...

Comportamento independente: cada nó deve ter sua **própria noção de tempo** – Não há clock global



Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Voltemos à definição...

Comportamento independente: cada nó deve ter sua **própria noção de tempo** – Não há clock global

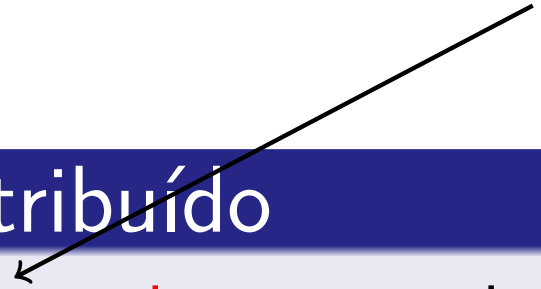
Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Leva a problemas fundamentais de sincronização e de coordenação.

Voltemos à definição...

Como gerenciar *associações em grupos*?

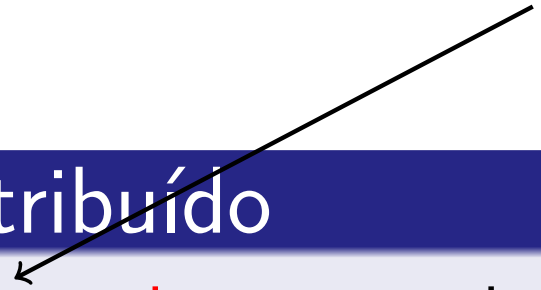


Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Voltemos à definição...

Como gerenciar *associações em grupos*?



Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Como saber se você realmente está se comunicando com um *(não-)membro autorizado do grupo*?

Coleção de nós

Organização: redes de overlay

Cada nó na coleção se comunica apenas com nós no sistema, seus vizinhos. O conjunto de vizinhos pode ser dinâmico, ou pode ser descoberto de forma implícita (ex: pode ser necessário procurá-lo)

Coleção de nós

Organização: redes de overlay

Cada nó na coleção se comunica apenas com nós no sistema, seus vizinhos. O conjunto de vizinhos pode ser dinâmico, ou pode ser descoberto de forma implícita (ex: pode ser necessário procurá-lo)

Tipos de overlay

Estruturada cada nó tem um *conjunto bem definido de vizinhos* com os quais pode comunicar (árvore, anel)

Não estruturada cada nó tem referências a *um conjunto aleatoriamente selecionado de outros nós* do sistema

Ex de rede de overlay: *sistemas peer-to-peer*

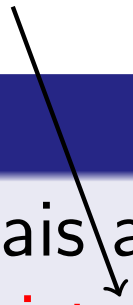
Voltemos à definição...

Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Voltemos à definição...

Essência: A coleção de nós opera sempre da mesma forma, não importando onde, quando ou como a interação entre um usuário e o sistema acontece.

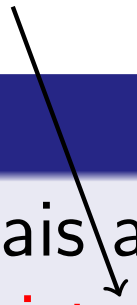


Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

Voltemos à definição...

Essência: A coleção de nós opera sempre da mesma forma, não importando onde, quando ou como a interação entre um usuário e o sistema acontece.



Sistema Distribuído

Coleção de elementos computacionais autônomos que, para o usuário, parecem um único sistema coerente.

A palavra chave é transparência de distribuição

Sistema coerente

Exemplos

- Um usuário não consegue dizer onde a computação está acontecendo
- Onde especificamente os dados estão armazenados deveria ser irrelevante para a aplicação
- O dado ser ou não replicado deveria estar completamente escondido

Sistema coerente

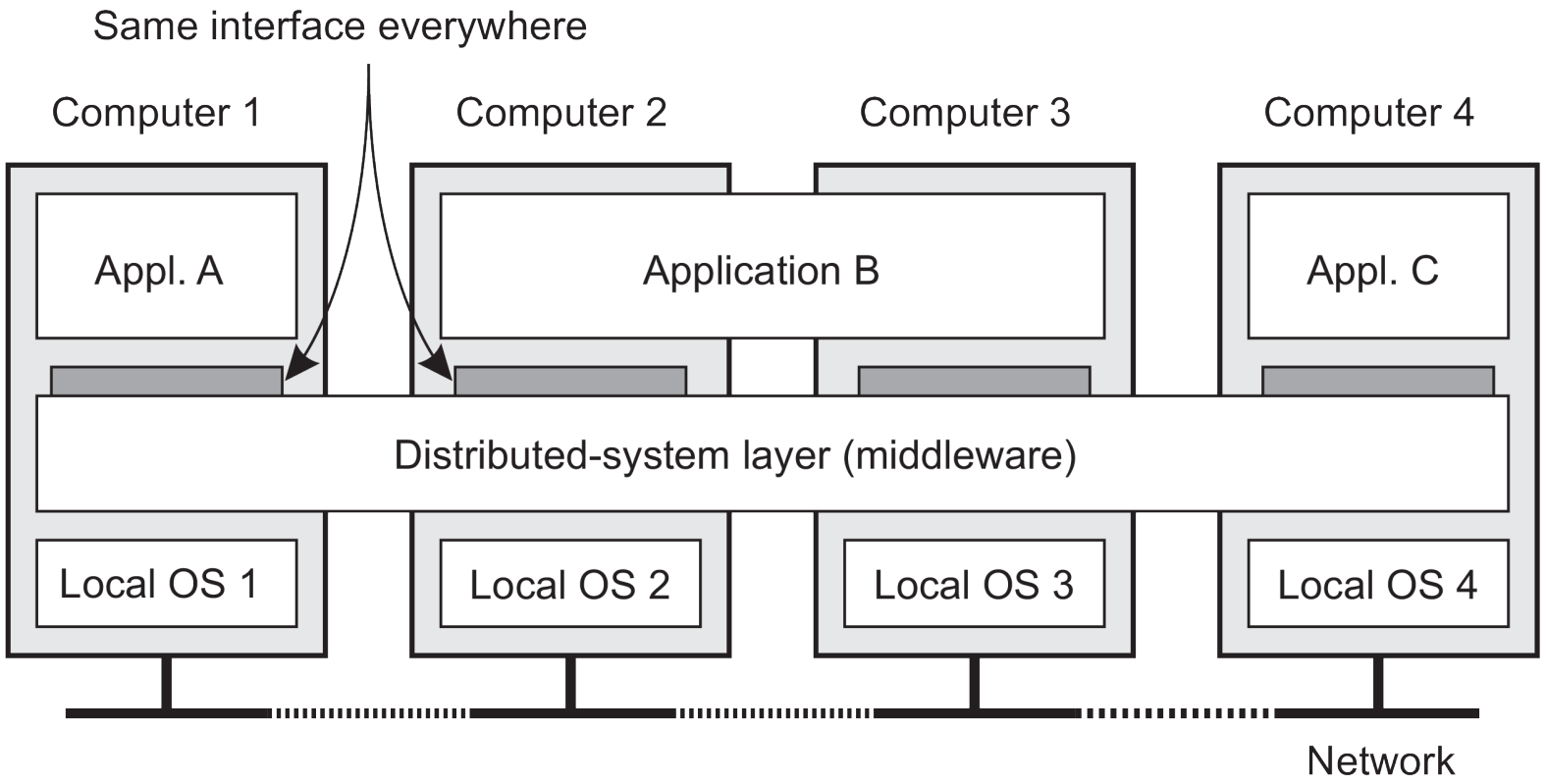
Problema: falhas parciais

- É inevitável que, a qualquer momento, um pedaço do sistema distribuído falhe
- Esconder essas falhas parciais e sua recuperação normalmente é muito difícil (em geral, impossível)

Middleware – O SO dos SDs

O que tem em um middleware?

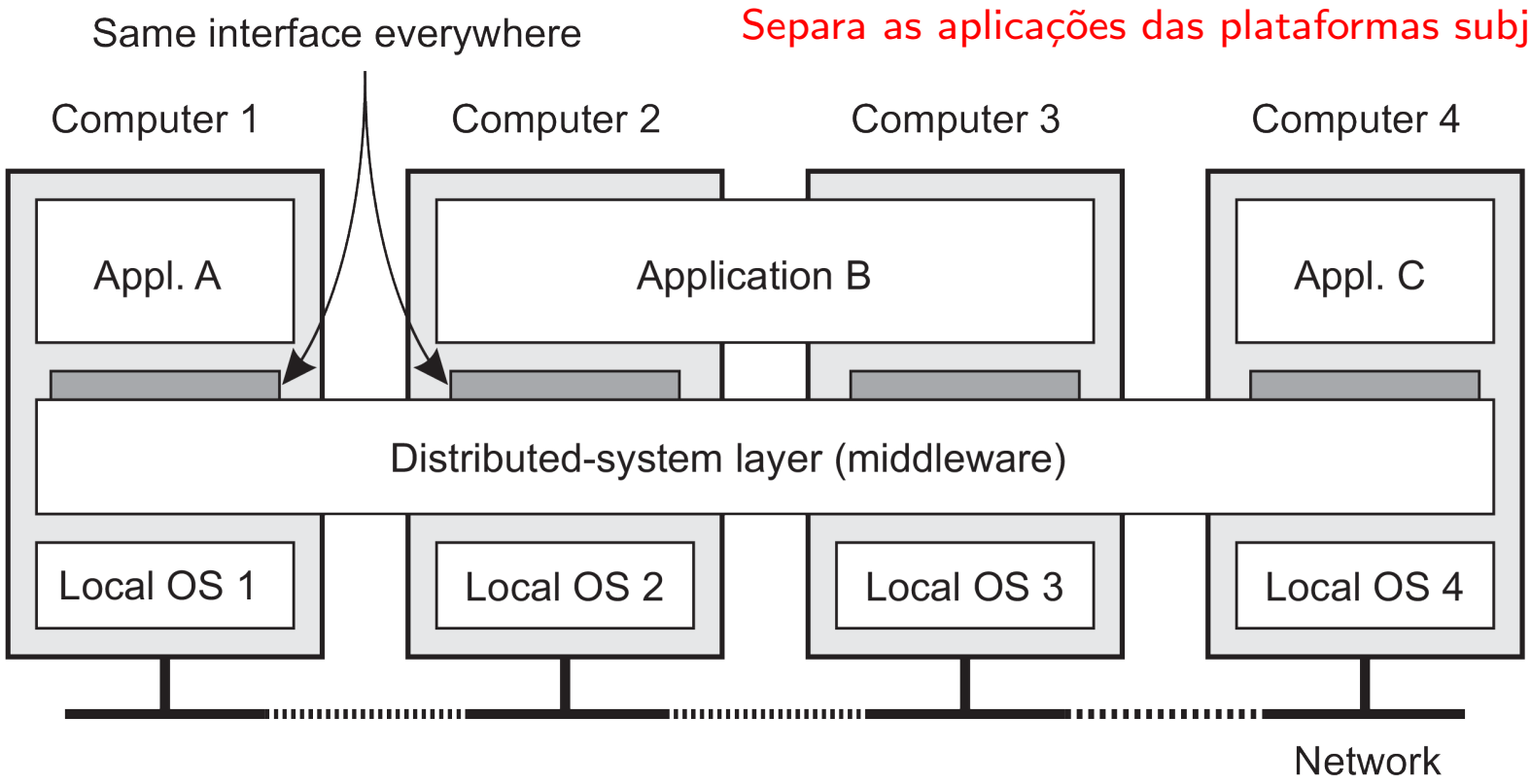
Funções e componentes comumente usados que não queremos reimplementar em cada aplicação separadamente



Middleware – O SO dos SDs

O que tem em um middleware?

Funções e componentes comumente usados que não queremos reimplementar em cada aplicação separadamente



Objetivos dos sistemas distribuídos:

- Disponibilização de recursos compartilhados
- Transparência de distribuição
- Abertura
- Escalabilidade

Objetivos dos sistemas distribuídos:

- **Disponibilização de recursos compartilhados**
- Transparência de distribuição
- Abertura
- Escalabilidade

Compartilhamento de recursos

Exemplos clássicos

- Compartilhamento de dados e arquivos na nuvem
- *Streaming* multimídia *peer-to-peer*
- Serviços de mensagens compartilhadas
- Serviços de hospedagem web compartilhados (*à la* redes de distribuição de conteúdo)

Compartilhamento de recursos

Exemplos clássicos

- Compartilhamento de dados e arquivos na nuvem
- *Streaming* multimídia *peer-to-peer*
- Serviços de mensagens compartilhadas
- Serviços de hospedagem web compartilhados (*à la* redes de distribuição de conteúdo)

A ponto de pensarmos que:

"A rede é o computador"

John Gage, Sun Microsystems (1984)

Objetivos dos sistemas distribuídos:

- Disponibilização de recursos compartilhados
- **Transparência de distribuição**
- Abertura
- Escalabilidade

Transparência de distribuição: Tipos

Transparência	Descrição
Acesso	Esconder diferenças entre as representações de dados e como um objeto é acessado
Localização	Esconder onde o objeto está localizado
Relocalização	Esconder que um objeto pode ser movido para outra localidade *enquanto está sendo utilizado*
Migração	Esconder que um objeto pode ser movido para outra localidade
Replicação	Esconder que um objeto está sendo replicado
Concorrência	Esconder que um objeto pode ser compartilhado entre diferentes usuários independentes
Falhas	Esconder falhas e a possível recuperação de um objeto

O propósito principal do middleware é prover esse tipo de transparência

Transparência de distribuição

Grau de transparência

Tentar fazer com que a distribuição seja totalmente transparente pode ser um exagero:

Transparência de distribuição

Grau de transparência

Tentar fazer com que a distribuição seja totalmente transparente pode ser um exagero:

- Há latências de comunicação que não podem ser escondidas

Transparência de distribuição

Grau de transparência

Tentar fazer com que a distribuição seja totalmente transparente pode ser um exagero:

- Há latências de comunicação que não podem ser escondidas
- Esconder completamente as falhas da rede e dos nós é (na teoria e na prática) **impossível**
  - Você não consegue distinguir um computador lento de um que está falhando
  - Você nunca consegue ter certeza de que um servidor terminou de realizar uma operação antes dele cair

Transparência de distribuição

Grau de transparência

Tentar fazer com que a distribuição seja totalmente transparente pode ser um exagero:

- Transparência completa terá um custo no desempenho, que irá expor a distribuição do sistema
  - Manter réplicas *rigorosamente* atualizadas com o original leva tempo
  - Necessário realizar *flush* das operações de escrita para garantir tolerância a falhas

Grau de transparência: Enfim...

Expor a distribuição pode ser bom

- Para usar serviços baseados em localização (ex: encontrar restaurantes no entorno)
- Quando tratar com usuários em diferentes fusos horários
- Quando isso facilita ao usuário entender o que está acontecendo
    - Ex: um servidor não responde por um longo tempo – reportar como falha

Grau de transparência: Enfim...

Expor a distribuição pode ser bom

- Para usar serviços baseados em localização (ex: encontrar restaurantes no entorno)
- Quando tratar com usuários em diferentes fusos horários
- Quando isso facilita ao usuário entender o que está acontecendo
    - Ex: um servidor não responde por um longo tempo – reportar como falha

Transparência de distribuição é um objetivo nobre, mas atingi-lo são outros quinhentos, e frequentemente não deveria nem mesmo ser almejado.

Objetivos dos sistemas distribuídos:

- Disponibilização de recursos compartilhados
- Transparência de distribuição
- **Abertura**
- Escalabilidade

Abertura de SDs

Sistemas distribuídos abertos

São capazes de interagir com outros sistemas abertos:

- devem respeitar interfaces bem definidas
- devem ser facilmente interoperáveis
- devem permitir a portabilidade de aplicações
- devem ser fáceis de estender

Abertura de SDs

Sistemas distribuídos abertos

- A abertura dos sistemas distribuídos os tornam independentes de:
  - hardware
  - plataformas
  - linguagens

Abertura de SDs: Políticas × Mecanismos

Implementando abertura: políticas

- Qual o nível de consistência necessária para os dados no cache do cliente?
- Quais operações podem ser realizadas por programas que acabamos de baixar da Internet?
- Quais os requisitos de QoS (*Quality of Service*) podem ser ajustados face a variações na banda disponível?
- Qual o nível de sigilo necessário para a comunicação?

Abertura de SDs: Políticas × Mecanismos

Implementando abertura: mecanismos

- Permitem a atribuição de políticas (dinâmicas) de cache
- Possuem diferentes níveis de confiança para código externo
- Proveem parâmetros de QoS ajustáveis por fluxo de dados
- Oferecem diferentes algoritmos de criptografia

Abertura de SDs: Políticas × Mecanismos

Implementando abertura: mecanismos

- Permitem a atribuição de políticas (dinâmicas) de cache
- Possuem diferentes níveis de confiança para código externo
- Proveem parâmetros de QoS ajustáveis por fluxo de dados
- Oferecem diferentes algoritmos de criptografia

Idealmente, sistemas distribuídos proveem apenas mecanismos.

Abertura de SDs: Políticas × Mecanismos

Observação

Quanto mais estrita for a separação entre políticas e mecanismos, mais nós precisamos garantir o uso de mecanismos apropriados, resultando potencialmente em muitos parâmetros de configuração e um gerenciamento mais complexo

Abertura de SDs: Políticas × Mecanismos

Observação

Quanto mais estrita for a separação entre políticas e mecanismos, mais nós precisamos garantir o uso de mecanismos apropriados, resultando potencialmente em muitos parâmetros de configuração e um gerenciamento mais complexo

Como encontrar um equilíbrio?

Definir políticas estritas normalmente simplifica o gerenciamento e reduz a complexidade, por outro lado isso implica em menos flexibilidade. Não há uma solução simples.