WSPPD'09

Em Direção a um Escalonador por Roubo de Tarefas na Criação de Tarefas por Divisão & Conquista com MPI-2

Stéfano **Mór** **sdkmor**@inf.ufrgs.br
Nicolas **Maillard** **nicolas**@inf.ufrgs.br

.inf INSTITUTO DE INFORMÁTICA UFRGS 20 ANOS
CNPq
GPPD

Mai/2009

Sumário

(1) Motivação

(2) Roubo de Tarefas Aleatorizado

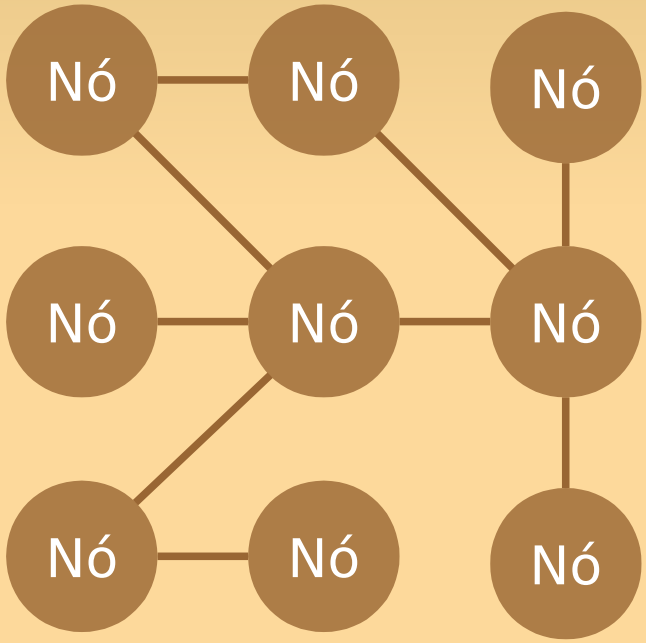
(3) Adaptando para Memória Distribuída

(4) Conclusões

Motivação

cluster

- memória **distribuída**
- troca de mensagens
- barramento rápido
- MPI
- visão lógica ”all-to-all”
- conectividade física varia



Motivação

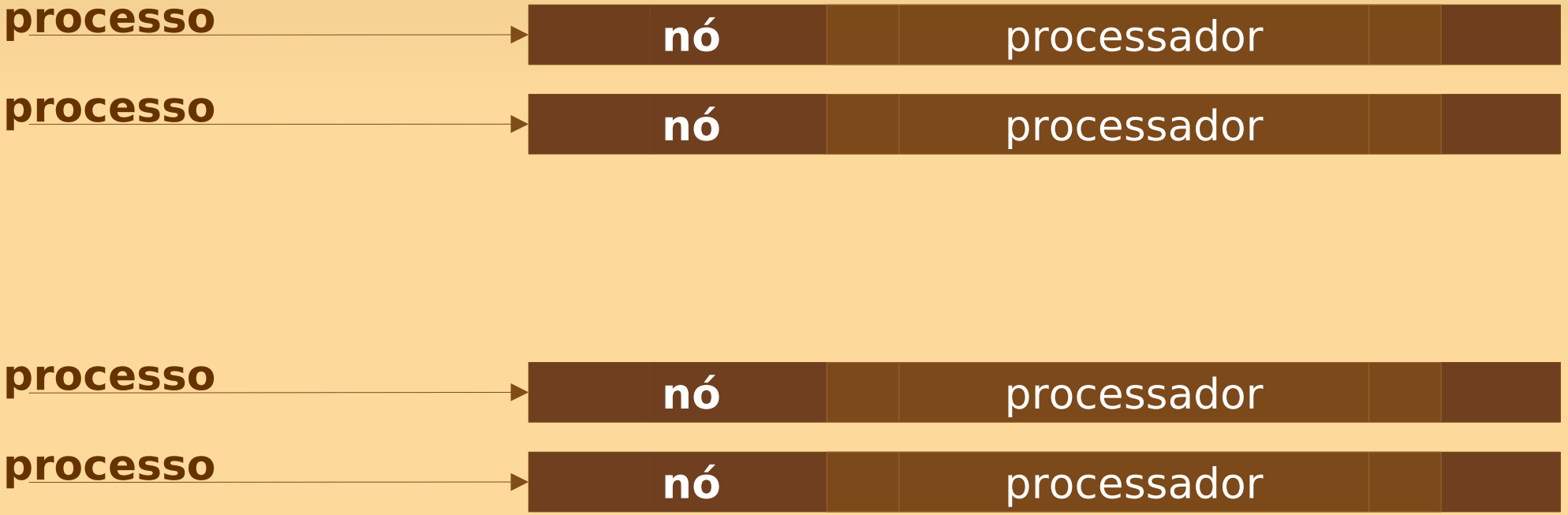
desafio: aproveitar ao máximo o hardware paralelo

foco atual: melhorar o escalonamento de processos MPI

Motivação

escalonamento MPI

MPI 1 :: estático :: tarefa=processo



Motivação

escalonamento MPI

MPI 2 :: estático & dinâmico :: processo = tarefa :: "**Task Spawning**"

Estado da Arte

Motivação

processor-aware: programador decide em qual processador alocar nova tarefa; **número de processadores** passados como **parâmetro**.

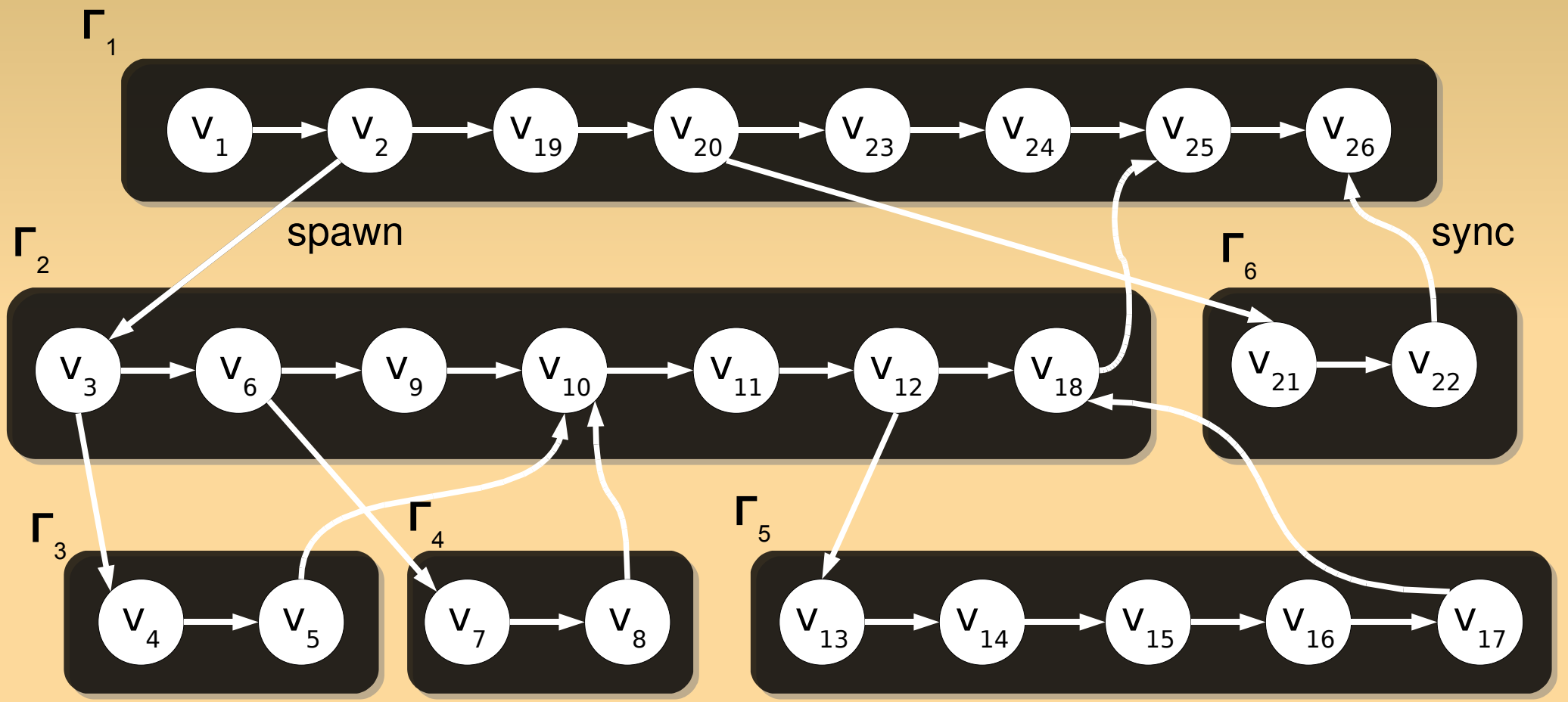
processor-oblivious: tarefa é alocada pelo middleware; **não importa** ao programador o **número de processadores**.

Contribuição

idéia central: usar o escalonamento por roubo de tarefas (projetado para memória compartilhada) em um cenário de memória distribuída com MPI

limitação: escalonamento eficiente para computação totalmente estrita (próximo slide) e.g., Divisão & Conquista (nosso foco)

Roubo de Tarefas Aleatorizado

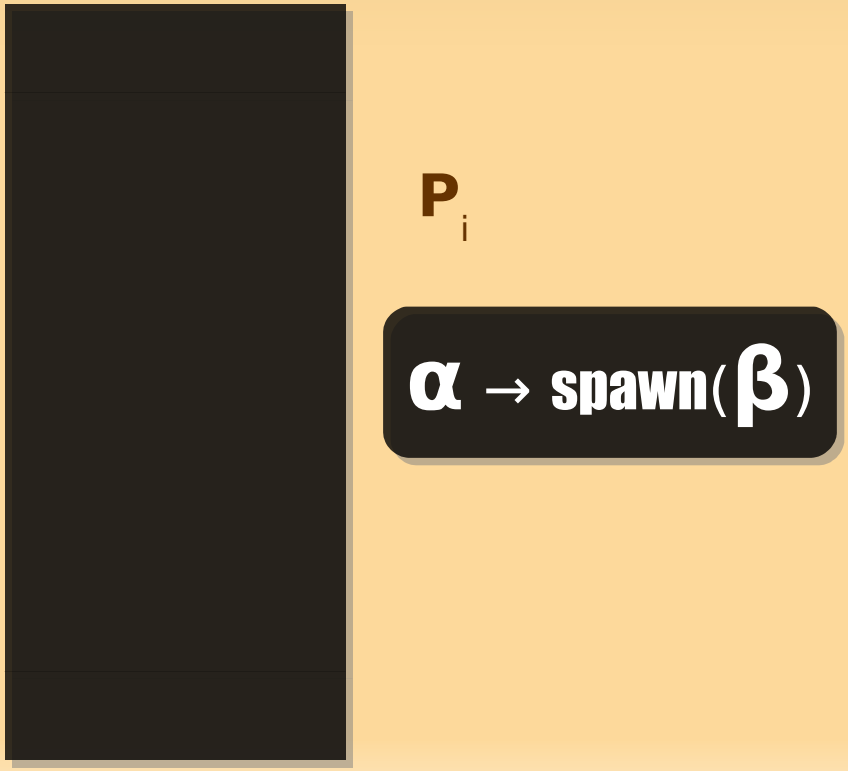


Roubo de Tarefas Aleatorizado

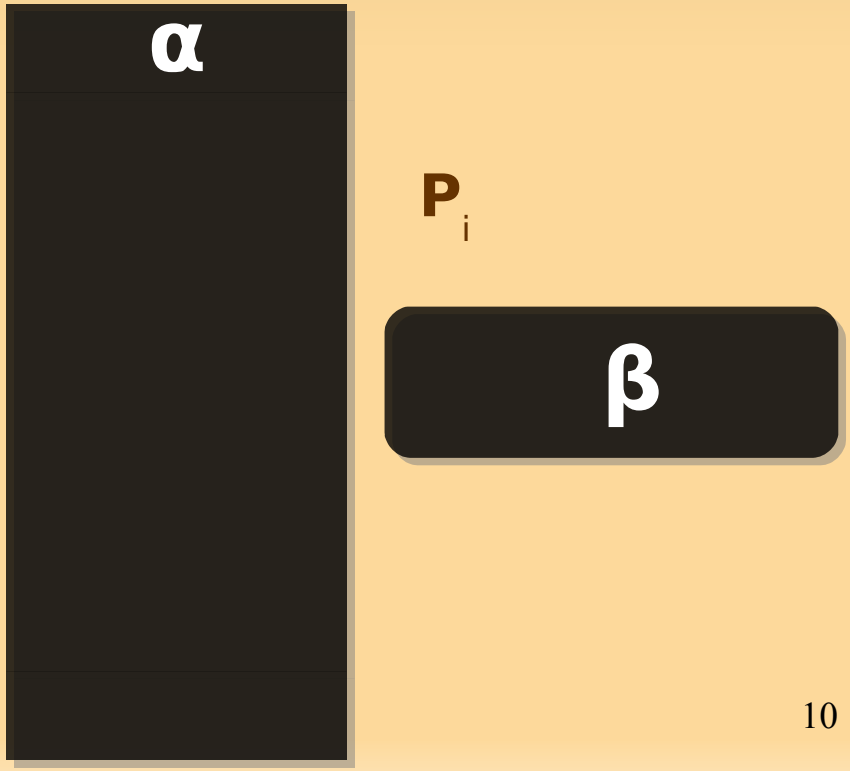
4 situações

uma *deque* por processador :: em cada processador

thread **α** faz spawn da thread **β**



α é empilhada e **β** ganha o processador



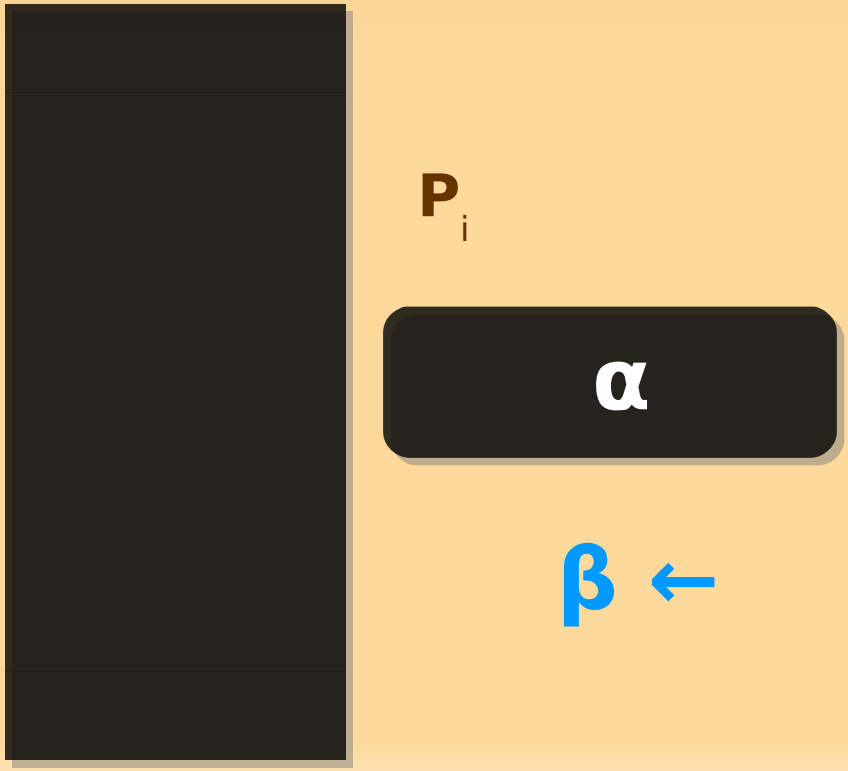
Roubo de Tarefas Aleatorizado

4 situações

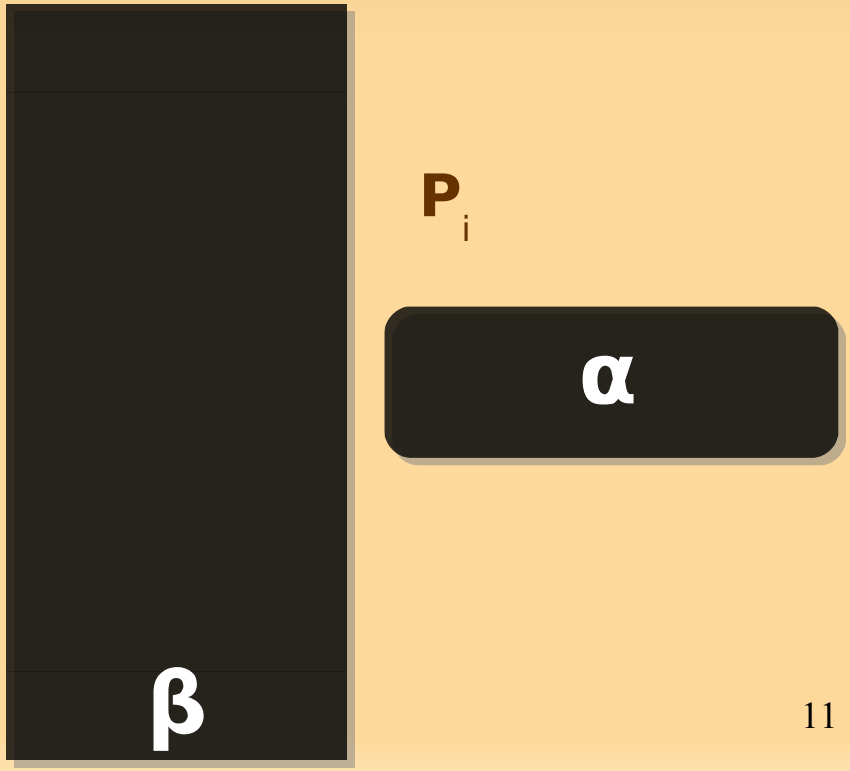
uma *deque* por processador :: em cada processador

thread **α** está executando e

thread **β** é habilitada



β vai para o fundo da *deque*

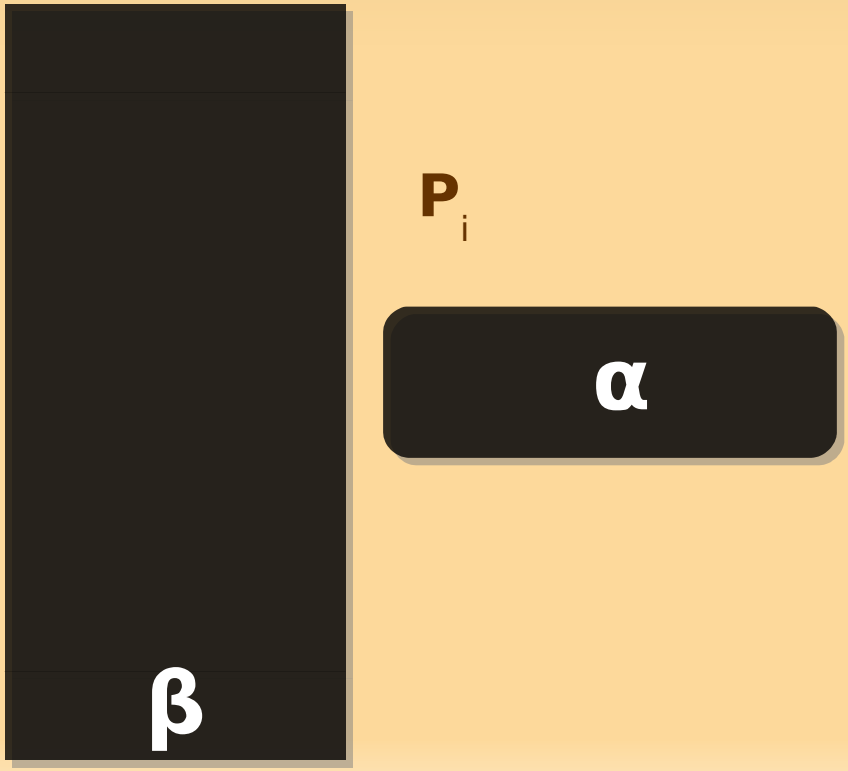


Roubo de Tarefas Aleatorizado

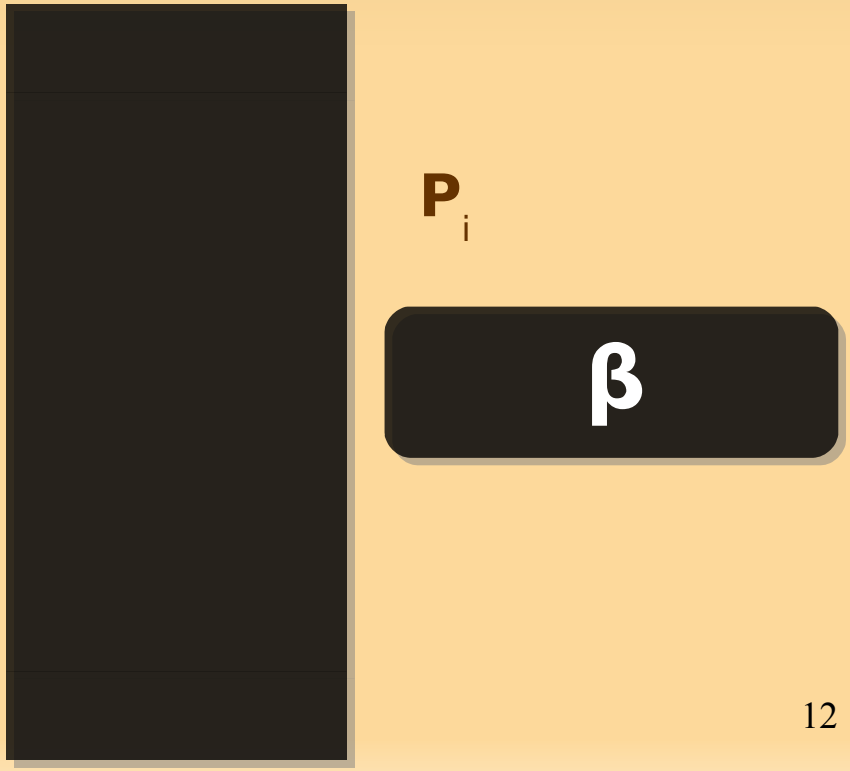
4 situações

uma *deque* por processador :: em cada processador

thread **α** está executando e morre.



se há thread **β** no fundo da *deque*, ela passa a executar.

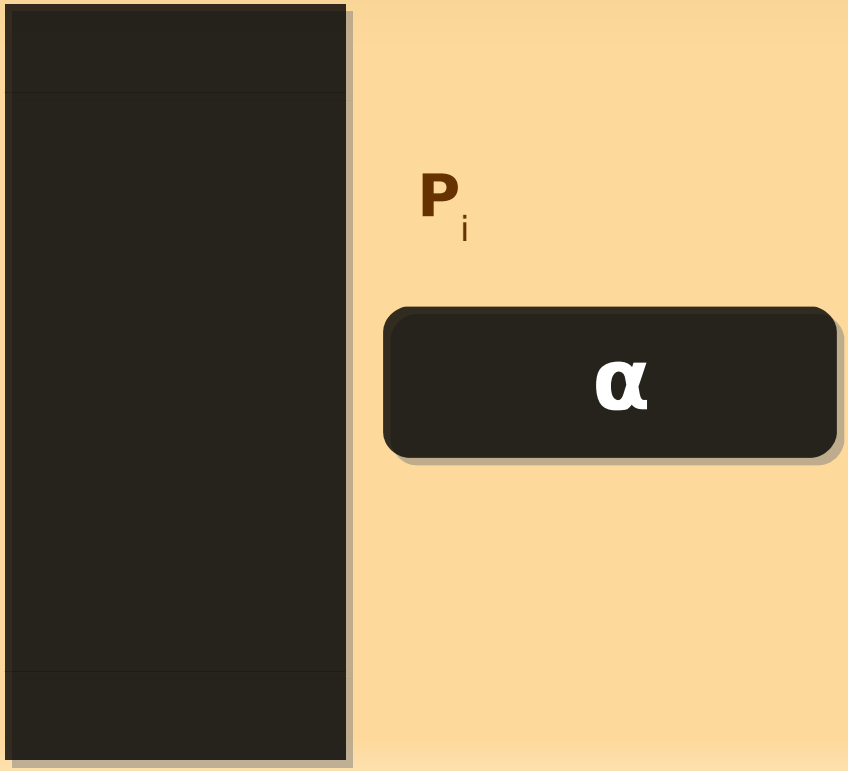


Roubo de Tarefas Aleatorizado

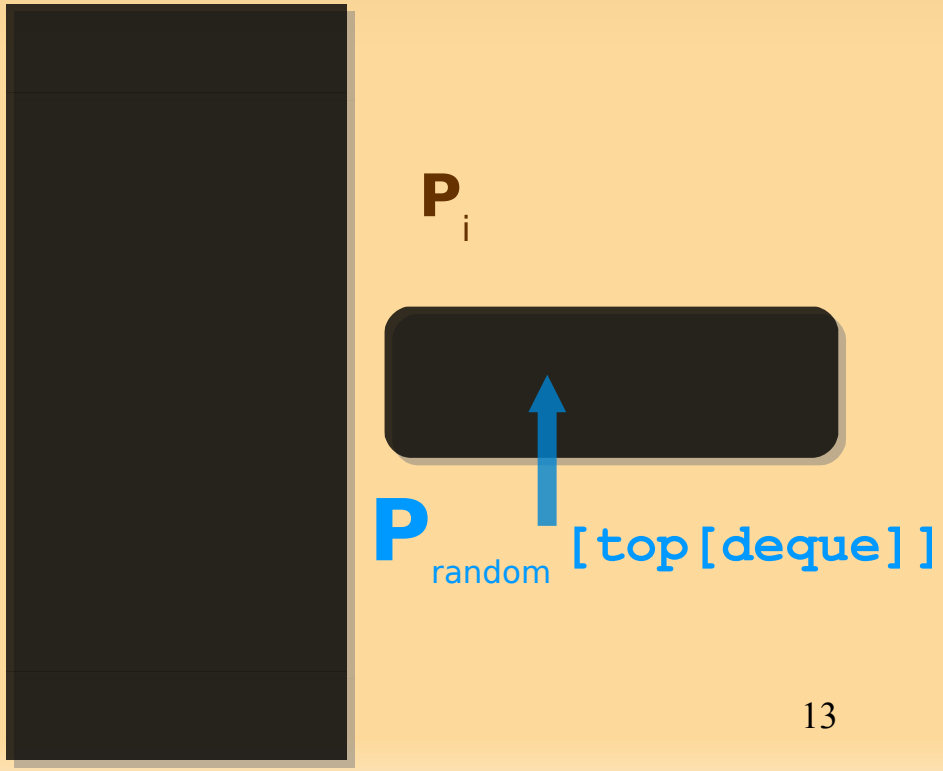
4 situações

uma *deque* por processador :: em cada processador

thread **α** está executando e morre.



se não há thread no fundo da *deque*, ela rouba tarefas do topo de outra *deque*.



Adaptando para Memória Distribuída

roubo de tarefas

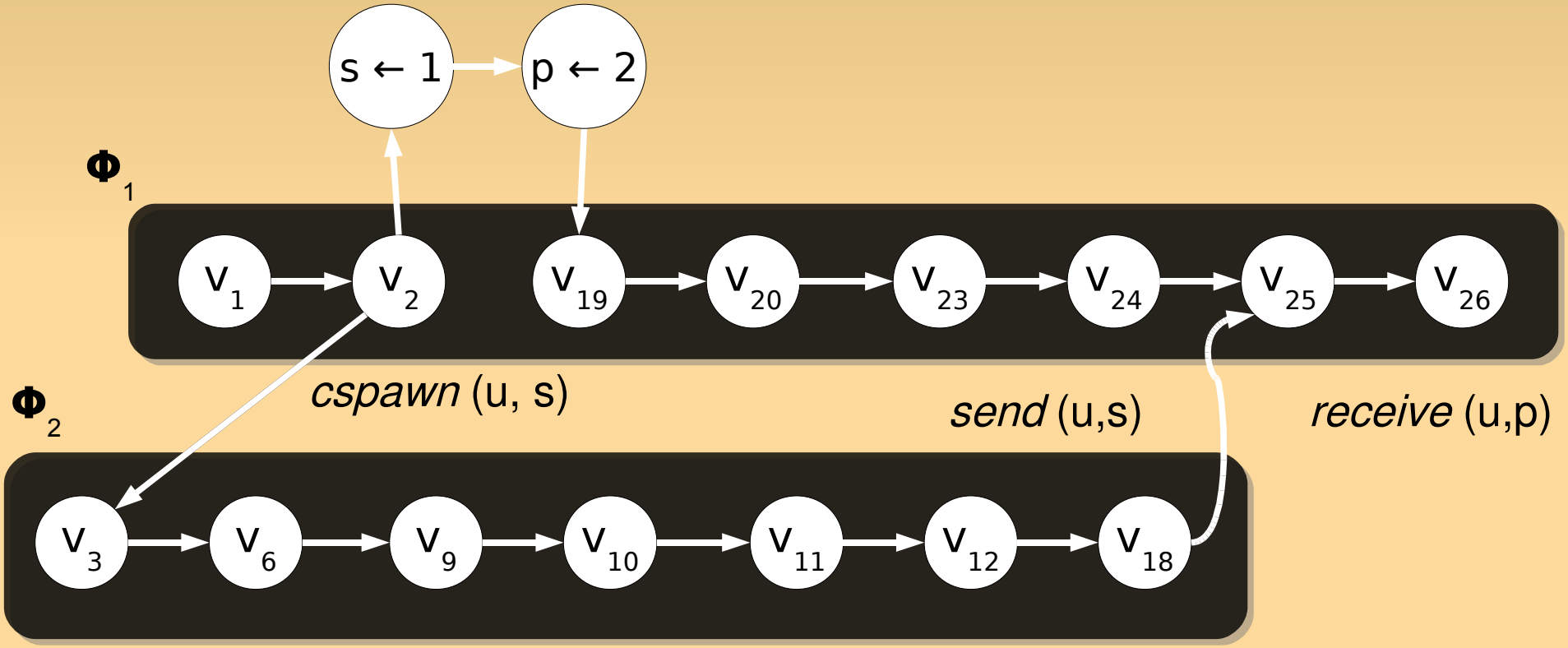
problemas com memória distribuída

- 4 problemas cruciais
  - problema do spawn (**+/-**)
  - problema da sincronização (✔)
  - problema de acesso concorrente à memória(✔)
  - problema de checkpointing(✔)

Adaptando para Memória Distribuída

roubo de tarefas

problema do spawn



Adaptando para Memória Distribuída

roubo de tarefas

problema da sincronização

- roubo de tarefas deve ser assíncrono em relação à vítima
  - nova thread que atende roubos
  - memória compartilhada com a thread que faz o processamento em si
  - não precisa parar de processar
  - apenas em máquinas SMT

Adaptando para Memória Distribuída

roubo de tarefas

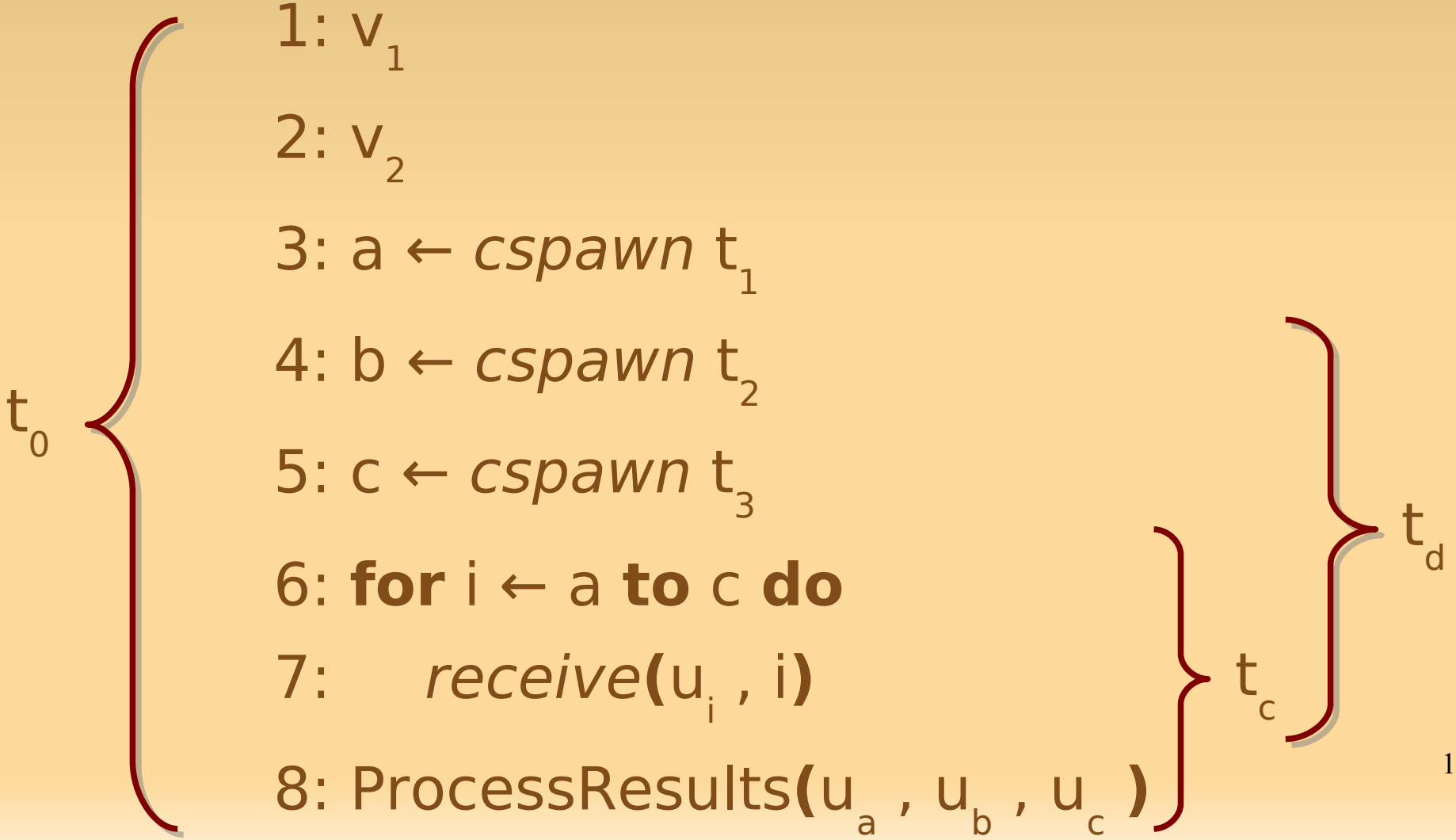
problema da acesso concorrente à memória

- thread de processamento e thread de atendimento devem ter acesso mutuamente exclusivo sobre a *deque*
  - o mesmo vale para tentativas de roubo sobre a mesma *deque*
  - implementação do protocolo THE
  - criação de uma nova thread de atendimento apra cada roubo
  - somente funciona com máquina SMT arbitrária

Adaptando para Memória Distribuída

roubo de tarefas

problema do checkpoiting



Conclusões

2 perguntas
questões em aberto

- existe equivalência entre o DAG de memória compartilhada e o DAG de memória distribuída?
  - *send/receive* bloqueante afeta a eficiência do RSWA?
- é possível implementar o RSWA de maneira eficiente em uma topologia do tipo anel (MPD)?

EOF

Agradecimentos ao CNPq

Obrigado!

Dúvidas?

* Referências bibliográficas nos anais.

SEMAC'09

Em Direção a um Escalonador por Roubo de Tarefas na Criação de Tarefas por Divisão & Conquista com MPI-2

Stéfano **Mór** **sdkmor**@inf.ufrgs.br
Nicolas **Maillard** **nicolas**@inf.ufrgs.br

INSTITUTO DE INFORMÁTICA UFRGS 20 ANOS
CNPq
GPPD

Mai/2009