Revisão

Introdução

Modelos de programação

Máquinas/arquiteturas paralelas

Sincronização

Passagem de mensagens

Construções e técnicas de prog //

Linguagens e bibliotecas para prog //

Hoje

Considerações de desempenho

- Métricas de desempenho
- Escalabilidade
- Eliminação de atrasos
- Escalonamento eficiente

Métricas de Desempenho

- Speedup
- Eficiência
- Redundância
- Utilização
- Qualidade

Metricas

Speedup = grau de melhora de desempenho

Eficiência = porção utilizada da capacidade

Redundância = aumento da carga qdo em p processadores

Utilização = utilização dos recursos durante computação

Qualidade = importância de utilizar processamento //

Metricas

Speedup s(p) = T(1) / T(p), onde T(1) = tempo do melhor algoritmo sequencial possível e p = número de processadores

Eficiência e(p) = s / p = T(1) / (p T(p))

Redundância r(p) = O(p) / O(1), onde O(p) = número total de ops em máquina com p processadores

Utilização u(p) = r(p) e(p) = O(p) / (p T(p))

Qualidade q(p) = (s(p) e(p)) / r(n) = $T^3(1)/(pT^2(p)O(p))$

Métricas

Outras Métricas

- MIPS
- MFLOPS
- Dhrystone = 100 comandos; CPU intensivo; ints
- Whetstone = FORTRAN; indexação de arrays, chamadas de subrotinas, etc; ints e floats
- SPEC
- Perfect Club

Escalabilidade

- Modelos de aplicacoes
- Limites algorítmicos
- Limites arquiteturais

Modelos de Aplicações

- Carga fixa = máquinas maiores para computar + rápido

- Tempo fixo = máquinas maiores para problemas maiores

- Memória fixa = máquinas maiores para problemas que precisam de + mem

Modelos de Aplicações

Limites Algorítmicos

- Falta de paralelismo
- Frequência de sincronização
- Padrão de comunicação/acesso
- Escalonamento deficiente

Limites Arquiteturais

- Latência/banda de comunicação
- Latência/banda de E/S
- Overhead de sincronização
- Overhead de coerência
- Capacidade de memória

Amdahl's Law

Speedup s = T(1)/T(p)

Trabalho total c = $T_s + T_p$ = T(1)

T(p) = $T_s + T_p/p$

s = $(T_s + T_p)$ / $(T_s + T_p/p)$ =
= c / $(T_s + T_p/p) \rightarrow c/T_s$ qdo p $\rightarrow$ inf

Amdahl's Law

Gustafson's Law

Tempo total c = $T_s + T_p$, fixo

Trabalho total = $T_s + pT_p$, assumindo tam problema aumenta linear/

scaled speedup = trabalho total / tempo total

ss = $(T_s + pT_p) / (T_s + T_p) =$
$= (T_s + pT_p)/c = (T_s + p(c\text{-}T_s))/c =$
$= p + (T_s\ (1\text{-}p))/c$, linear em T_s

Gustafson's Law

Memória Fixa

Speedup Superlinear

- Overhead reduzido (escalonamento, por exemplo)

- Mais memória/cache

- Tolerância a latência

- Randomização (problemas de otimização)

Tolerância a Overheads

Idéia é sobrepor overheads com computação ou overheads com overheads

Técnicas mais usadas

- Caches
- Prefetching
- Multithreading
- Consistência relaxada
- Protocolos baseados em updates

Prefetching

Procura trazer dados para perto do processador antecipadamente

Pode ser implementado em hardware, software (compilador ou runtime system), ou de forma híbrida (hybrid prefetching)

Algum prefetching já é feito por blocos de cache com mais de uma palavra, por page faults, etc

Ex: Numa cache miss, o hardware busca o bloco que causou a miss e o próximo bloco

Ex: Compilador insere instrução de prefetch para dado acessado no futuro

Multithreading

Procura esconder atrasos executando outra(s) thread(s)

Pode ser implementado em hardware ou software (runtime system)

Ex: Numa cache miss, o hardware pede o bloco que causou a miss, troca de contexto e executa uma outra thread

Ex: Num bloqueio por mensagem, o software passa a executar outra thread

Consistência Relaxada

Procura permitir a buferização e pipeline de escritas para sistemas baseados em memória compartilhada, sem que haja necessidade imediata de estabelecer a coerência dos dados escritos

Pode ser implementada em hardware ou software (runtime system)

Ex: Release Consistency como em DASH

Ex: Lazy Release Consistency como em TreadMarks

Update Protocols

Evitam a latência de comunicação por não invalidar dados compartilhados em sistemas baseados em memória compartilhada

Problema: comunicação excessiva de atualizações (inúteis)

Alguns protocolos combinam atualizações e invalidações para melhorar a performance

Assim como outras técnicas de coerência, implementação possível em hardware ou software (compilador ou runtime system)

Ex: Munin (release consistency com updates atrasados)

Ex: Poststore na KSR

NCP_2 da COPPE/UFRJ

Memória compartilhada em SW com auxílio de HW (CP)

- Facilidade de programação
- Baixo custo
- Alto desempenho

Facilidade de programação: modelo de memória compartilhada

Baixo custo: base em SW, HW independente de plataforma, componentes off-the-shelf

Alto desempenho: tempo curto de projeto e tolêrancia a overheads de comunicação e coerência

Tolerância a Overheads

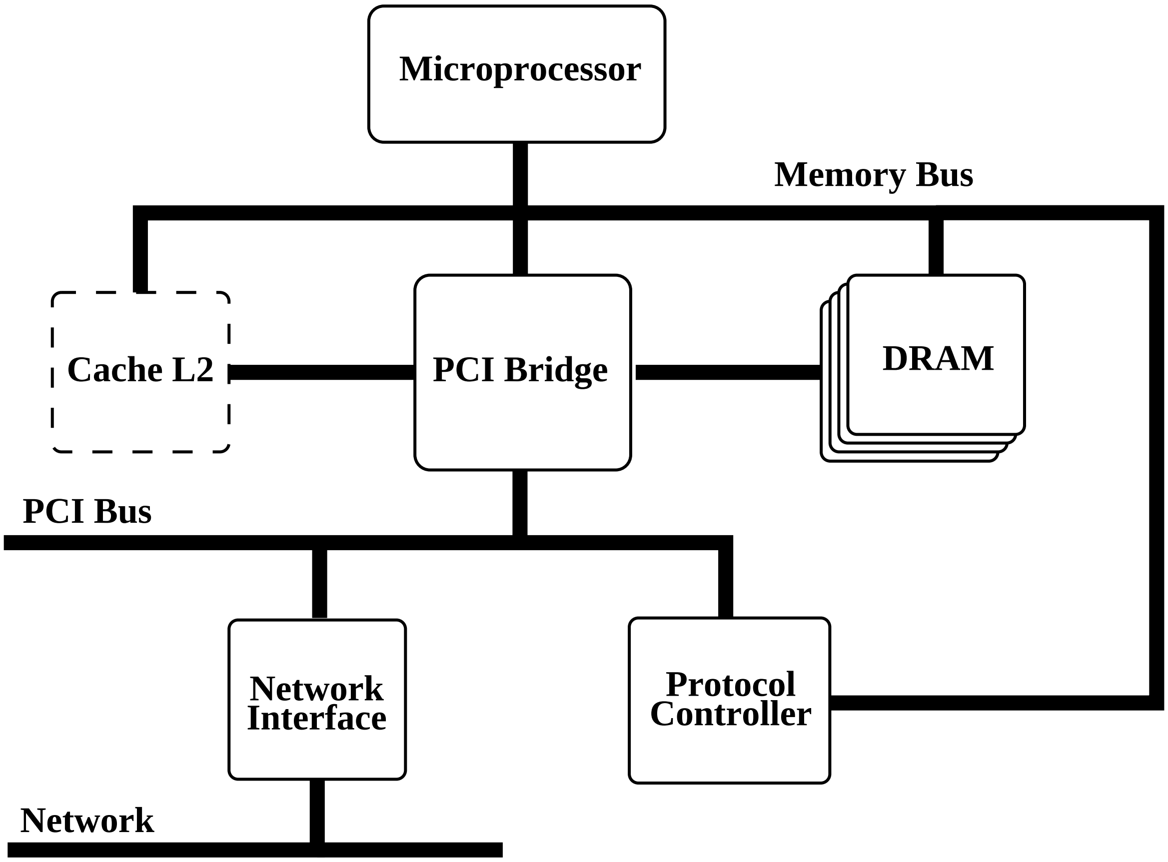
CP gera diffs antecipadamente

CP busca dados remotos antecipadamente

CP executa tarefas básicas de comunicação e coerência

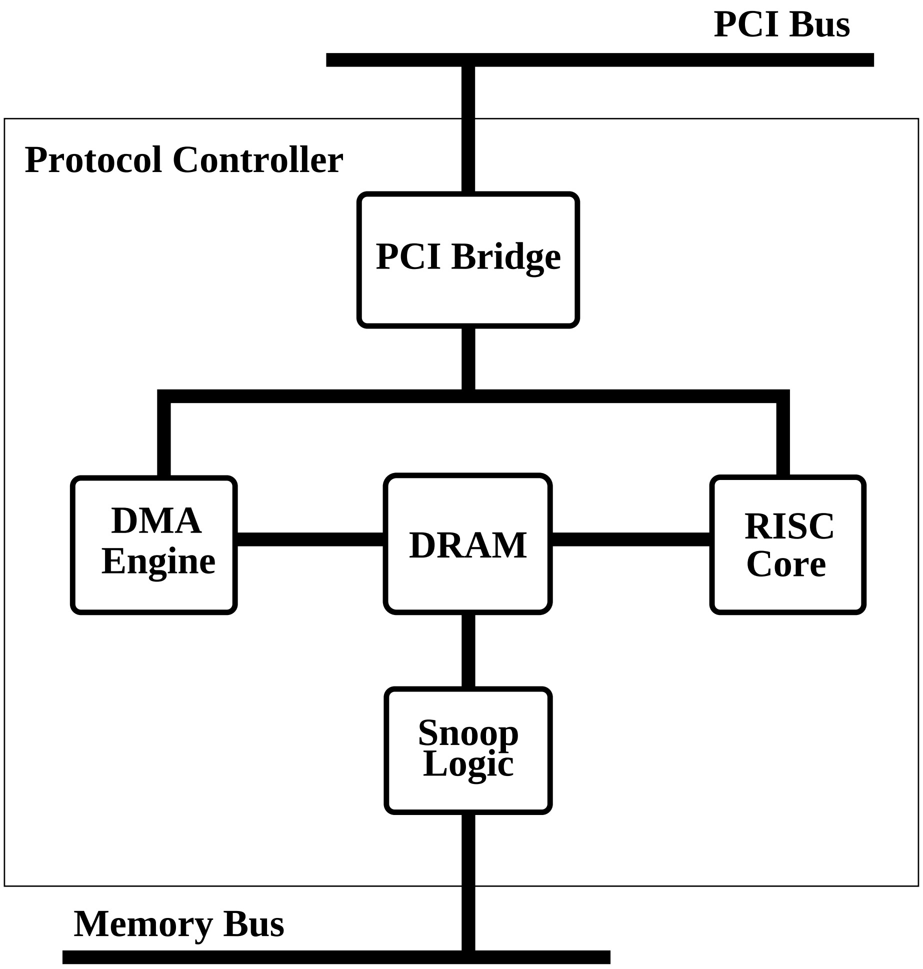
CP acumula diffs on-the-fly (geração e aplicação em HW)

NCP$_2$: Arquitetura da 1a. versão



Arquitetura de um Nó

NCP$_2$: Arquitetura da 1a. versão



Arquitetura do CP

NCP_2: Software

Processador executa tarefas complicadas

Controlador de Protocolos

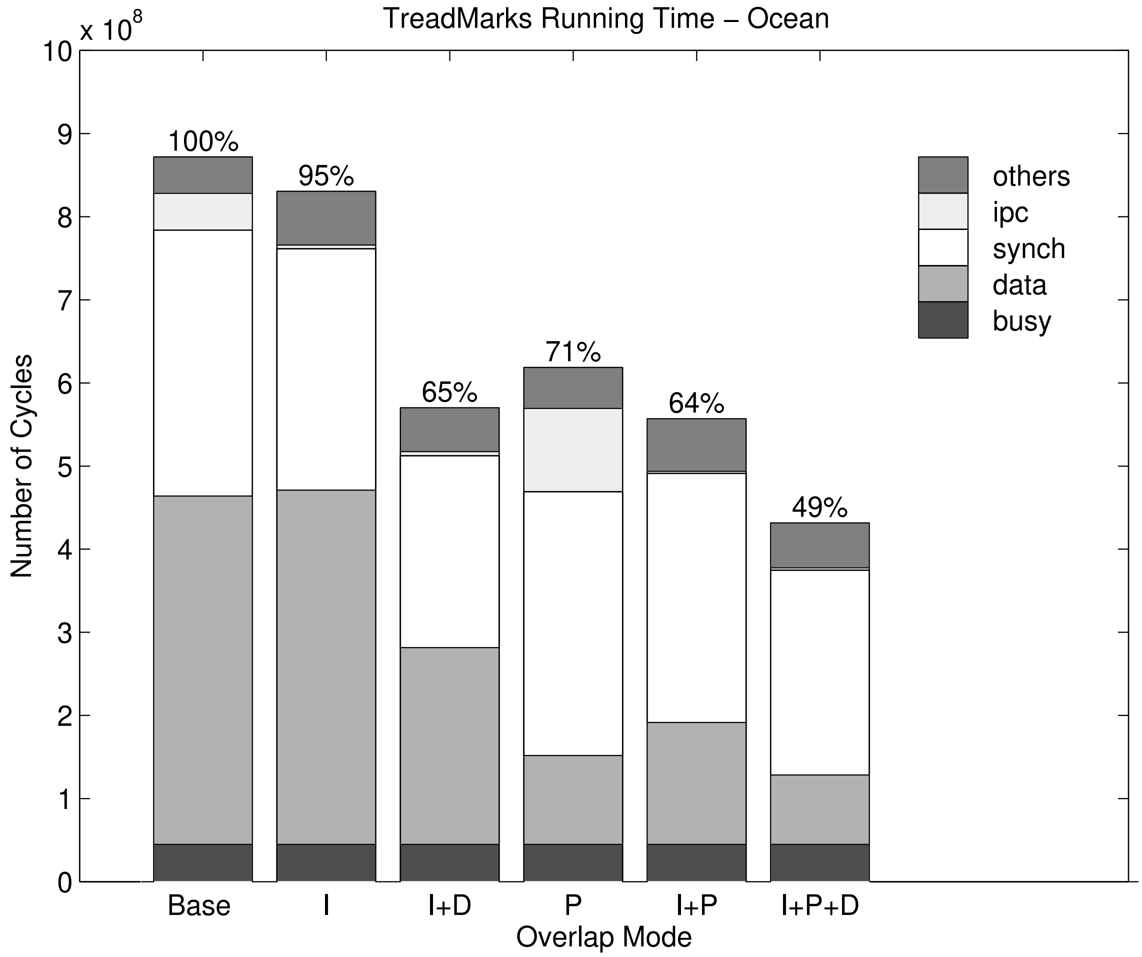
- Busca e retorno de dados remotos

- Geração e aplicação de diffs locais e remotos

- Controle de acesso a diretórios

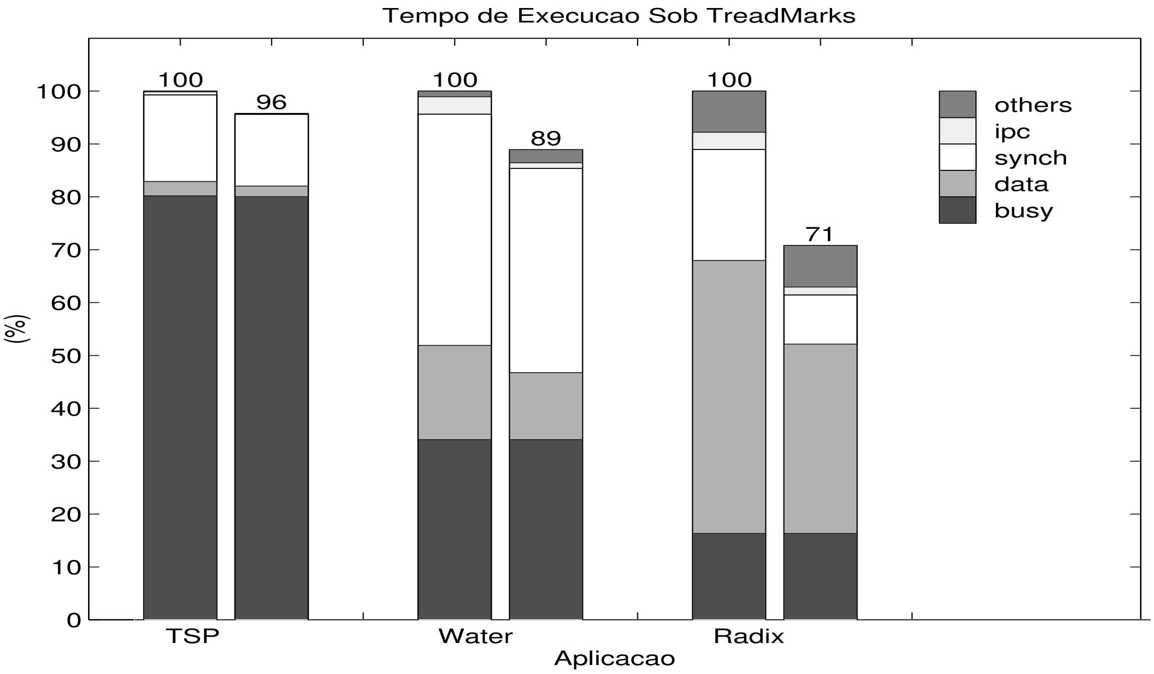
- Passagem de mensagens ativas

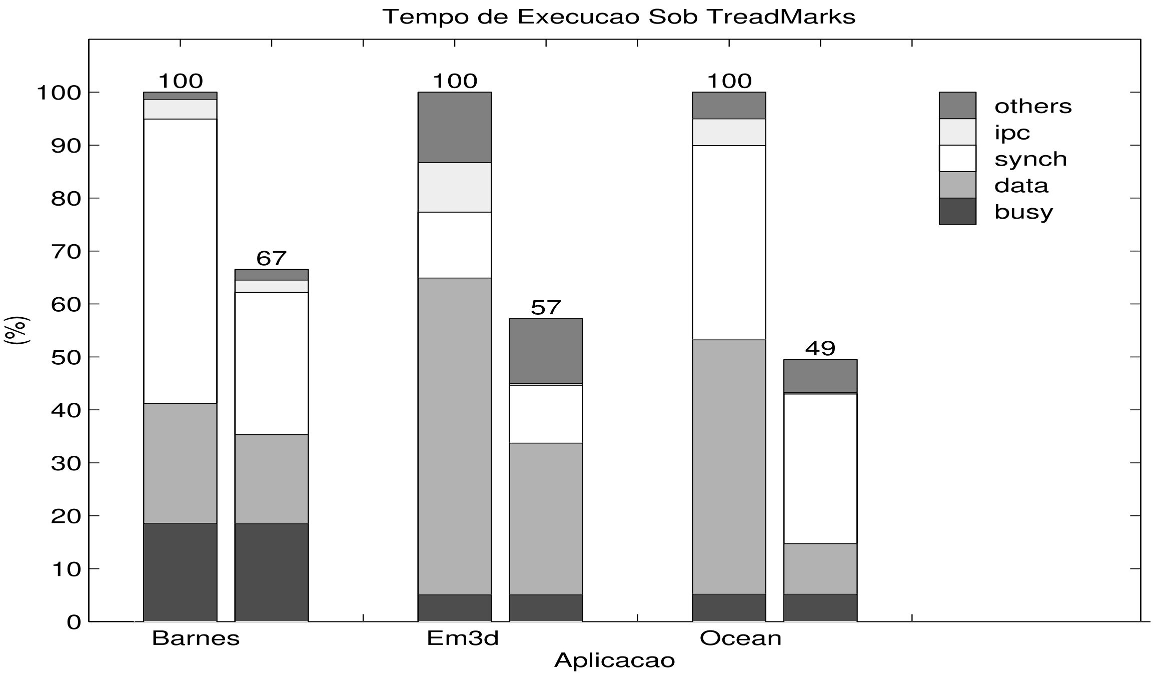
Resultados para TM Modificado



Comparação com TM Padrão

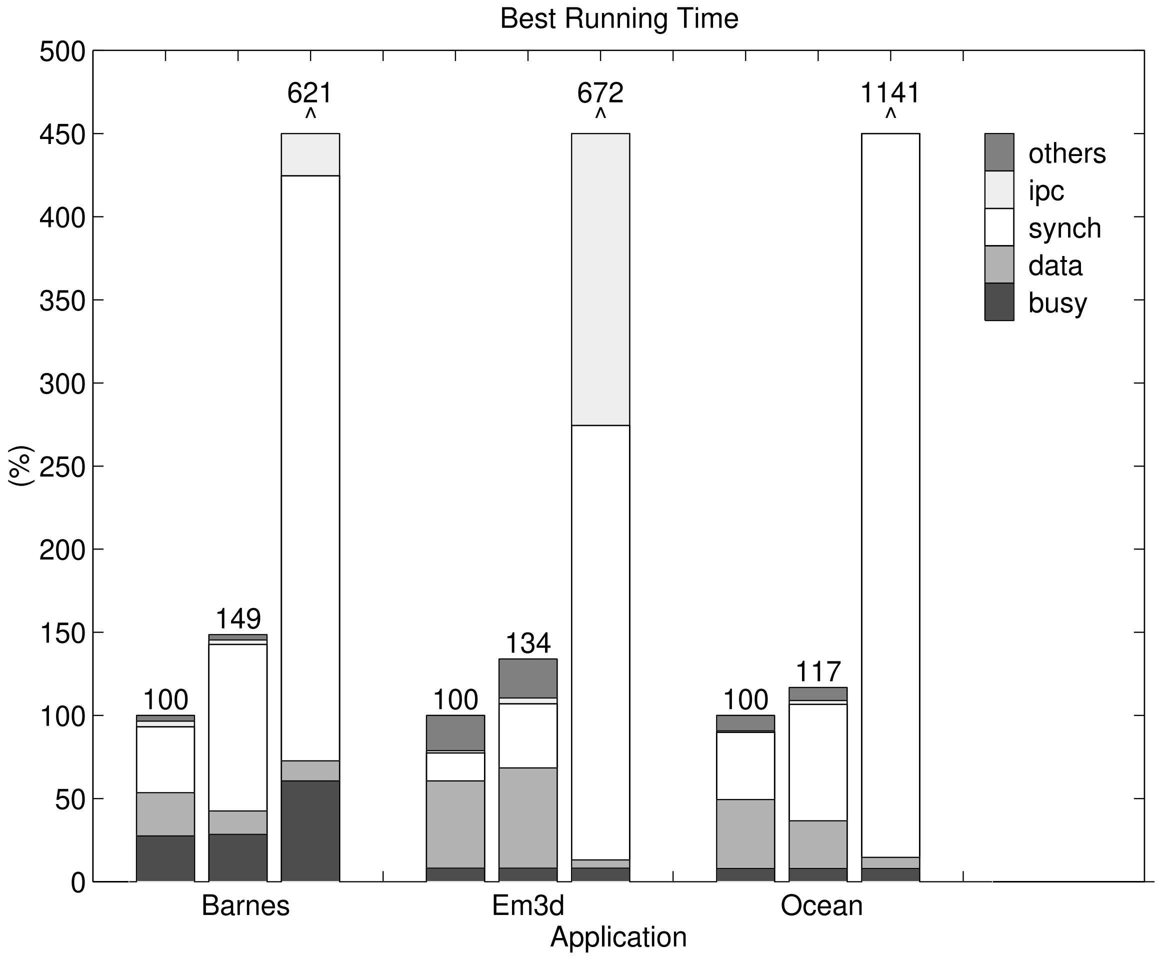
Resultados para TM Modificado





Sumário de Resultados

Resultados para TM Modificado



Comparação com AURC

NCP_2: Tecnologias

Hardware

- Placas mãe baseadas no Power PC 604
- Rede de interconexão Myrinet
- Processador i960 no CP
- Tecnologia: EPLDs

Software

- Linux
- TreadMarks modificado
- DSMs desenvolvidos na COPPE/UFRJ

Escalonamento Eficiente

Time slicing em geral tem péssimo desempenho para maquinas paralelas

Opções:

- Affinity scheduling

- Gang scheduling

- Hardware partitions

- Process control

Escalonamento Eficiente

Time slicing: Unix para multiprocs

Affinity scheduling: roda processo em processador “afim”

Gang scheduling: todos os processos de uma aplicação rodam juntos

Hardware partitions: partição reservada para cada aplicação

Process control: aplicação se adapta a partições mutantes

Próxima Aula

Capítulo 9 do livro do Foster

Aluno responsável por esclarecer dúvidas