



SSC-0742

PROGRAMAÇÃO CONCORRENTE

**Aula 07 – Técnicas de Desenvolvimento de
Programas Paralelos – Parte 1**

Prof. Jó Ueyama

Créditos

Os slides integrantes deste material foram construídos a partir dos conteúdos relacionados às referências bibliográficas descritas neste documento

Visão Geral da Aula de Hoje

1

- Conceitos Básicos

2

- Técnicas de Decomposição

3

- Comunicação

4

- Sincronização

5

- Dependência de Dados

6

- Balanceamento de Carga

7

- Exercício e Leitura Recomendada



CONCEITOS BÁSICOS

Conceitos Básicos

- Paralelização Automática X Manual
 - Desenhar e desenvolver programas paralelos tem sido um processo essencialmente manual
 - O programador em geral é o responsável tanto pela identificação quanto pela implementação efetiva do paralelismo

Conceitos Básicos

- Paralelização Automática X Manual
 - Muitas vezes desenvolver código paralelo de forma manual consome tempo e é um processo mais complexo e suscetível a erros. Por outro lado, pode-se obter melhores *speedups* em comparação à paralelização automática de código
 - Várias ferramentas estão disponíveis para ajudar um programador converter programas seriais em programas paralelos
 - *O tipo mais comum de ferramenta para realizar esta tarefa são pré-processadores ou compiladores de paralelização*

Conceitos Básicos

- Paralelização Automática X Manual
 - Um compilador paralelizador pode ser:
 - **Totalmente automática**
 - O compilador analisa o código fonte e identifica oportunidades de paralelismo
 - A atividade envolve identificar inibidores de paralelismo e uma ponderação se o uso de paralelismo poderia ser eficiente para melhorar o desempenho
 - Loops (do, for) são os alvos mais frequentes para a atividade de paralelização automática

Conceitos Básicos

- Paralelização Automática X Manual
 - **Direcionado ao Programador**
 - Diretivas de compilação ou flags do compilador. O programador informa explicitamente ao compilador como paralelizar o código. Pode também ser utilizado com algum grau de paralelização automática

Conceitos Básicos

- Paralelização Automática X Manual
 - Alguns pontos importantes em relação à paralelização automática:
 - Erros podem ocorrer
 - O desempenho pode ser ruim
 - Menos flexível que a paralelização manual
 - Limitado a um subconjunto de código (principalmente loops)
 - Não pode paralelizar o código se a análise sugere que não há inibidores ou o código é complexo
 - Neste curso, serão abordadas técnicas para o desenvolvimento manual de programas paralelos.

Conceitos Básicos

- **Problema x Programa**

- Primeiro passo para desenvolver um programa paralelo
 - ENTENDER O PROBLEMA
- Se a ideia é a paralelização de um programa serial, o primeiro passo é entendê-lo
- NÃO PERCA TEMPO
 - Antes de desenvolver uma solução paralela para o problema, determine primeiro se o problema apresenta potencial para exploração de paralelismo

Conceitos Básicos

- Exemplo do que pode ser paralelizado:
 - Resolução de sistemas lineares
- Exemplo do que não pode ser paralelizado
 - Série de Fibonacci (1,1,2,3,5,8,13,21)
 - $F(n) = F(n-1) + F(n-2)$
 - O cálculo da sequência implica em cálculos dependentes ao invés de independentes
 - O cálculo de $F(n)$ utiliza cálculo de $F(n-1)$ e $F(n-2)$; Os três termos não podem ser calculados independentemente

Conceitos Básicos

- **Problema x Programa**

- **Identificar os pontos críticos de um programa**

- Saiba onde a maioria do trabalho está sendo feito
 - Muitos programas científicos consomem a maior parte do tempo de processamento em poucos locais
 - Ferramentas de desempenho podem auxiliar nesse sentido
 - Concentre-se no paralelismo dos pontos críticos e ignore as seções do programa que fazem pouco uso da CPU

Conceitos Básicos

- **Problema x Programa**

- **Identificar gargalos no programa**

- Há áreas de um programa que são desproporcionalmente mais lentas. Ex: E/S é algo que degrada o desempenho de um programa
 - Possibilidade de reestruturar o programa ou utilizar um algoritmo diferente para reduzir ou eliminar áreas com processamento mais demorado

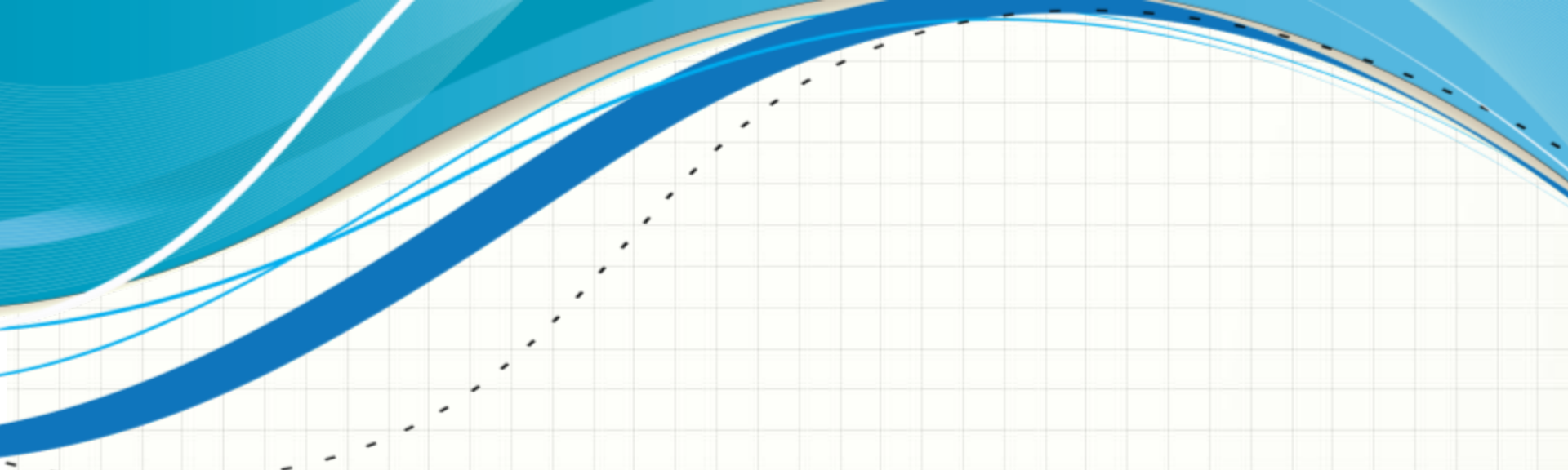
Conceitos Básicos

- **Problema x Programa**

- **Identificar inibidores de paralelismo**

- Dependência de dados (como mostrado na sequência de Fibonacci)

- Investigar outros algoritmos quando possível.
Esta pode ser a consideração mais importante ao projetar uma aplicação paralela



TÉCNICAS DE DECOMPOSIÇÃO

Técnicas de Decomposição

- Primeiro passo
- Quebre o programa em blocos de trabalho que possam ser distribuídos em várias tarefas



Decomposição/Particionamento

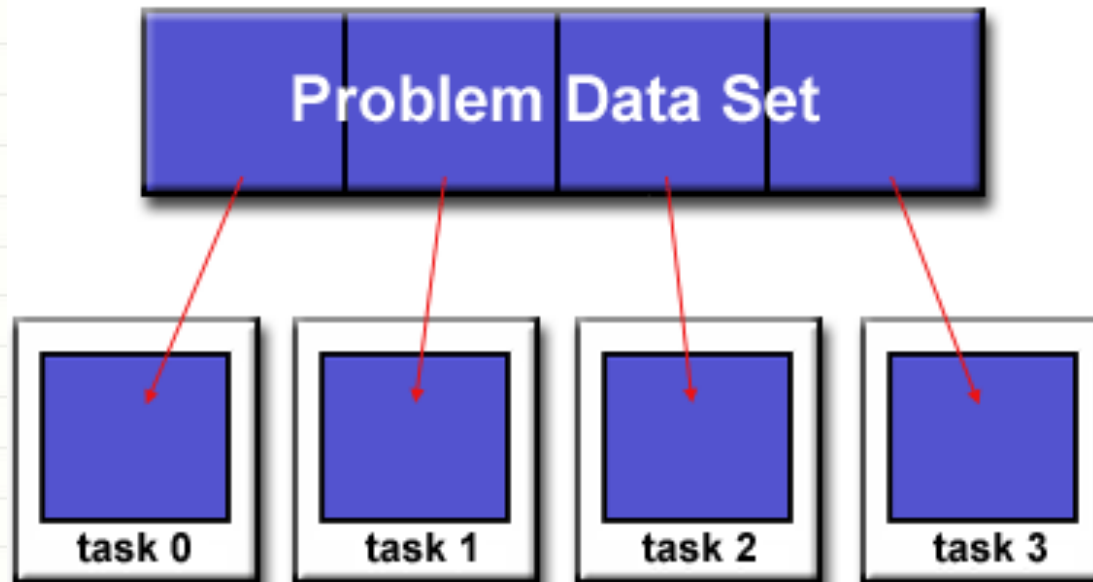
Técnicas de Decomposição

- Dois modos básicos de particionamento de um trabalho computacional em tarefas paralelas:
 - **Decomposição de domínio e Decomposição funcional**

Técnicas de Decomposição

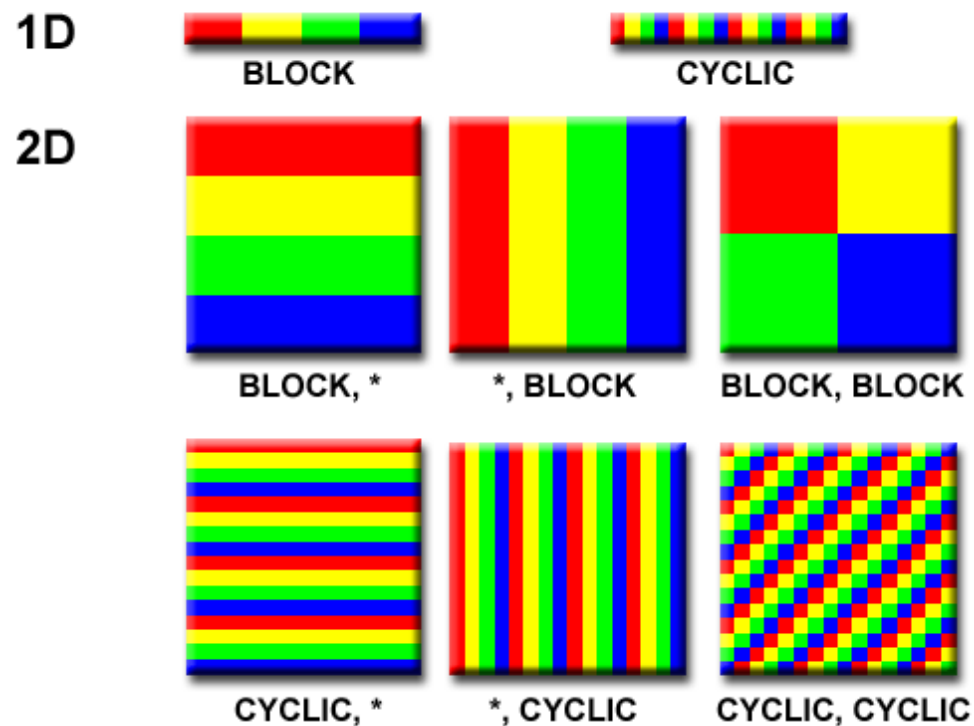
- **Decomposição de Domínio**

- O dado associado ao problema é decomposto.
 - Cada tarefa paralela trabalha sobre uma porção dos dados



Técnicas de Decomposição

- **Decomposição de Domínio**
 - Dois modos diferentes para particionar dados

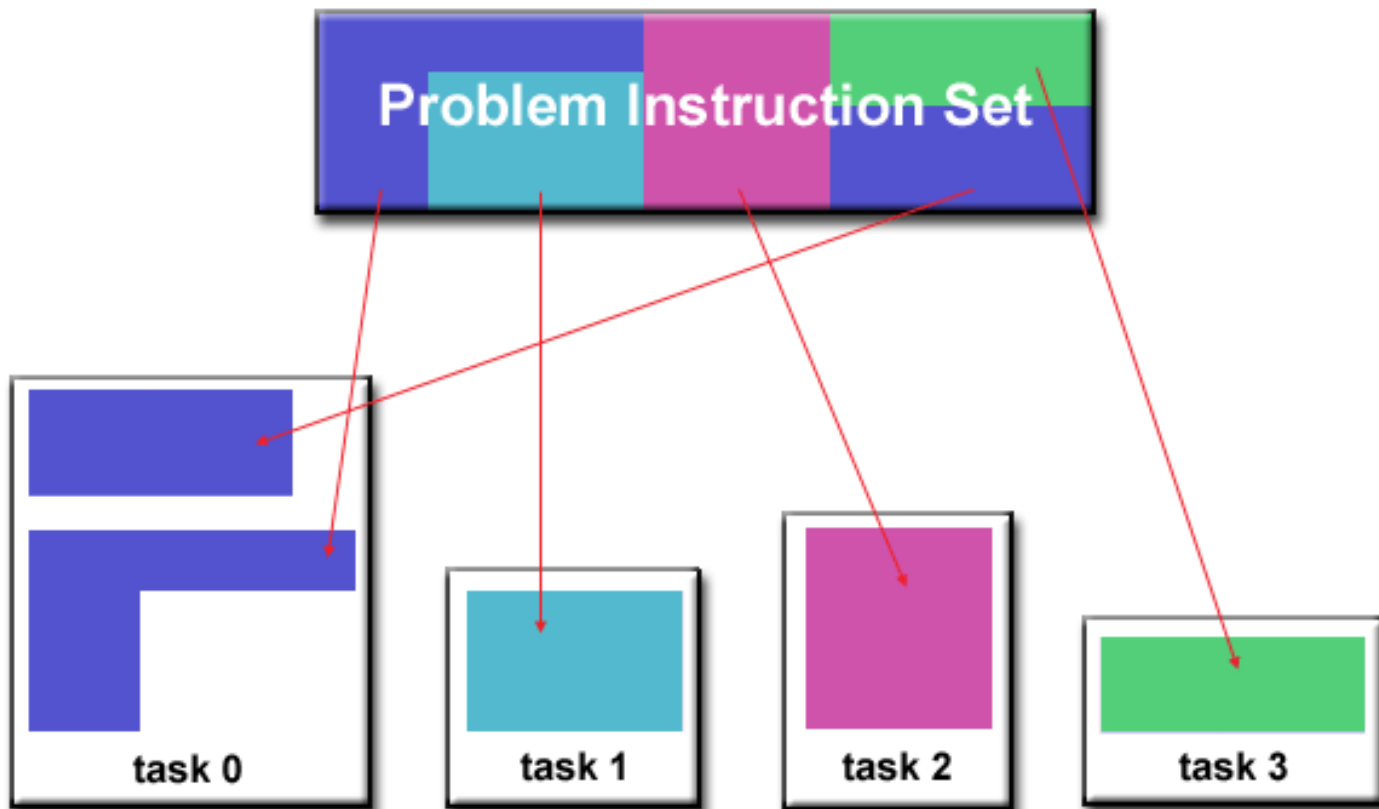


Técnicas de Decomposição

- **Decomposição Funcional**
 - Nesta abordagem o foco recai sobre o algoritmo executado e não sobre o conjunto de dados
 - O problema é decomposto de acordo com o trabalho que deve ser feito
 - Cada tarefa executa parte do trabalho global

Técnicas de Decomposição

- **Decomposição Funcional**

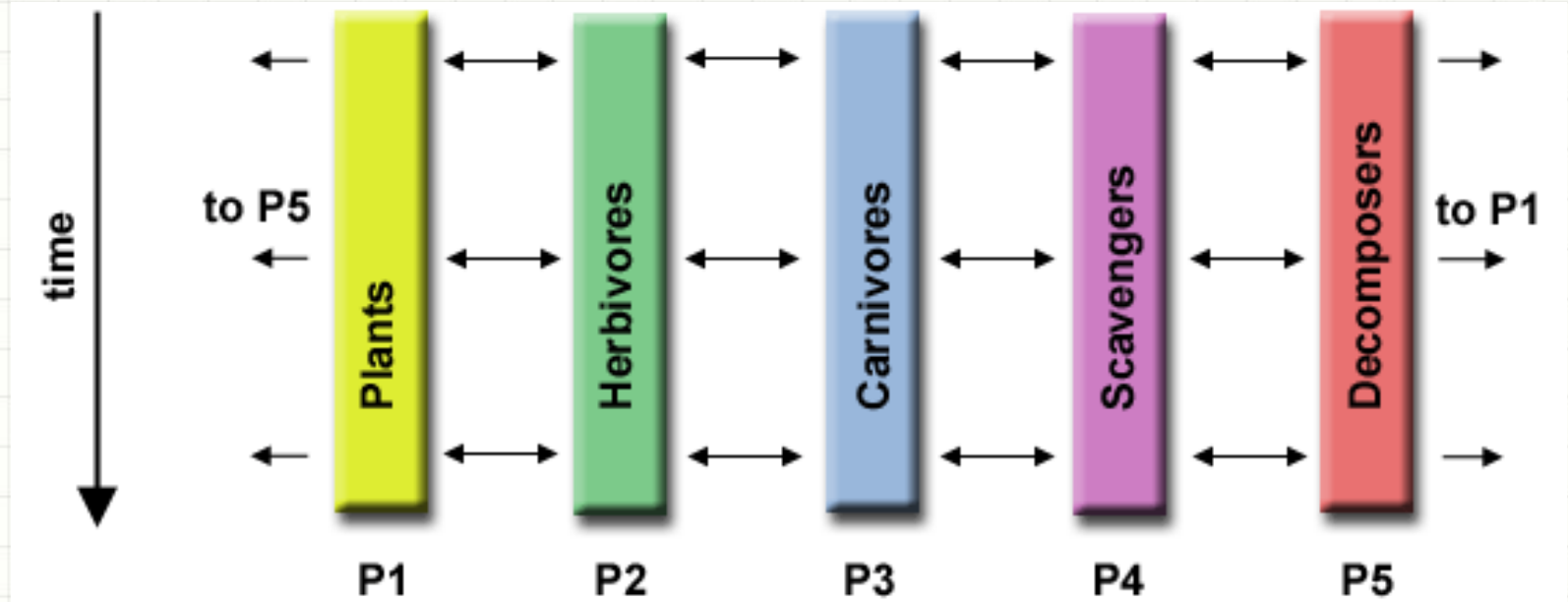


Técnicas de Decomposição

- **Decomposição Funcional**
 - Funciona bem para problemas que podem ser divididos em diferentes tarefas. Exemplo:
 - Modelagem de um ecossistema
 - Processamento de Sinais
 - Modelagem Climática

Técnicas de Decomposição

- **Decomposição Funcional**
 - **Modelagem de um Ecossistema**

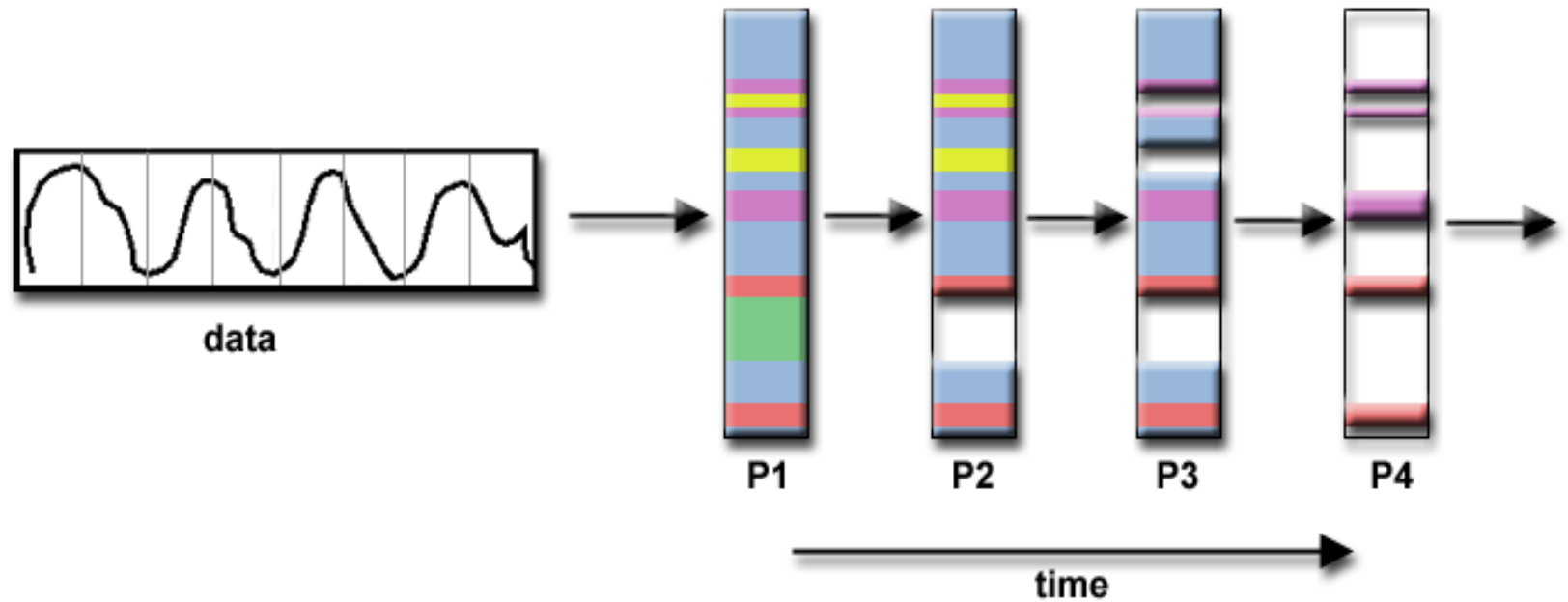


Técnicas de Decomposição

- **Decomposição Funcional**
 - **Modelagem de um Ecossistema**
 - Cada programa calcula a população de um determinado grupo. O crescimento de cada grupo depende de seus vizinhos. Conforme o tempo avança, cada processo calcula seu estado atual e em seguida troca informações com as populações vizinhas. Todas as tarefas prosseguem para calcular o estado da próxima etapa.

Técnicas de Decomposição

- **Decomposição Funcional**
 - **Processamento de Sinais**



Técnicas de Decomposição

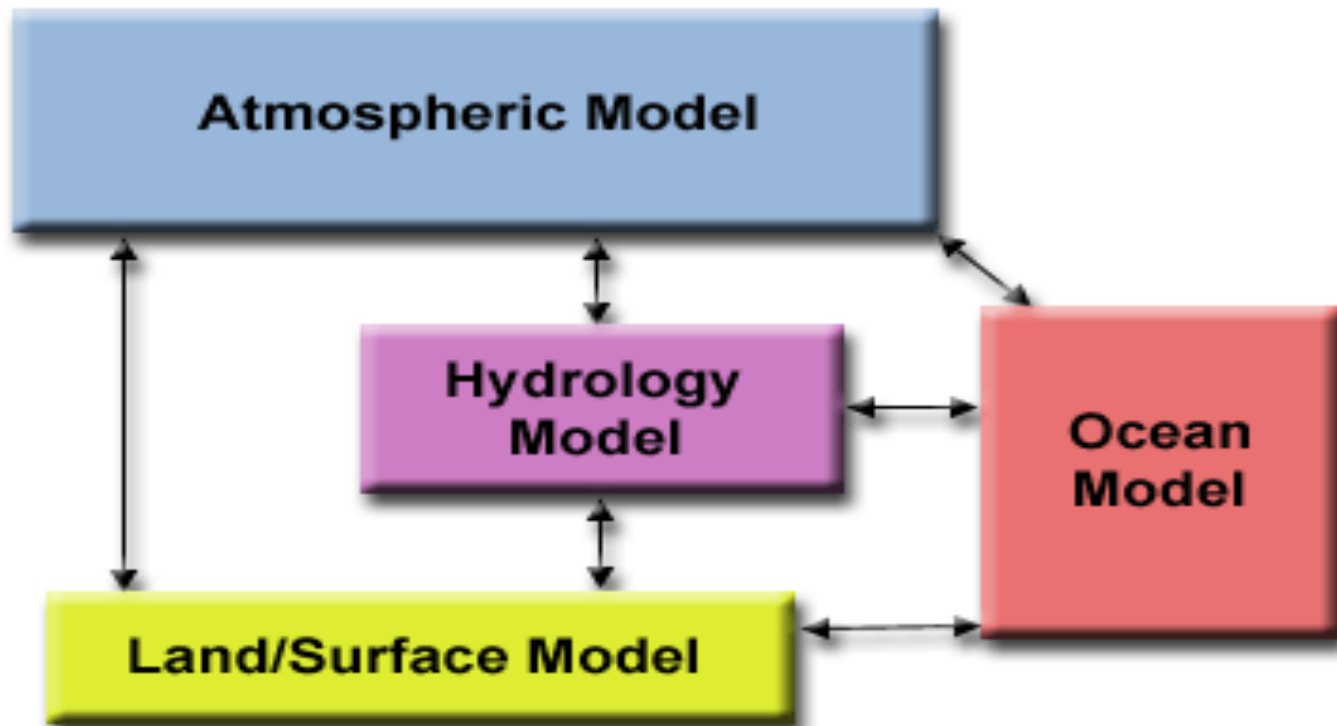
- **Decomposição Funcional**

- **Processamento de Sinais**

- Um conjunto de sinais de áudio é passado através de quatro filtros computacionais distintos. Cada filtro é um processo separado. O primeiro segmento de dados deve passar pelo primeiro filtro antes de avançar para o segundo. Quando isso ocorre, o segundo segmento de dados passa pelo primeiro filtro. Nesse instante o quarto segmento está no primeiro filtro. Todas as tarefas estão ocupadas, funcionando como um *pipeline*.

Técnicas de Decomposição

- **Decomposição Funcional**
 - **Modelagem Climática**



Técnicas de Decomposição

- **Decomposição Funcional**

- **Modelagem Climática**

- Cada componente do modelo pode ser processado por uma tarefa separada
 - As setas representam o intercâmbio de dados entre os componentes durante a computação
 - O modelo da atmosfera gera dados de velocidade do vento que são utilizados pelo modelo oceânico
 - O modelo oceânico gera dados de temperatura da superfície do mar que são usado pelo modelo de atmosfera, e assim por diante



COMUNICAÇÕES

Comunicações

- A necessidade de comunicação depende do problema
- **Necessidade de Comunicação**
 - Alguns tipos de problemas podem ser decompostos e executados em paralelo sem a necessidade das tarefas terem que compartilhar dados. Exemplo:
 - Operação de processamento de imagem onde cada pixel de uma imagem preto e branco precisa ser invertida. Os dados da imagem podem ser facilmente distribuídos para várias tarefas que então agem de forma independente uma da outra para realizar sua parte no trabalho.
 - Esse tipo de problema é chamado de embaraçosamente paralelo. Muito pouca comunicação inter-tarefa é necessária

Comunicações

- **Comunicação Necessária**

- A maioria das aplicações paralelas não são tão simples e precisam que as tarefas possam compartilhar dados umas com as outras.

Exemplo:

- Problema de difusão de calor em 3D requer que uma tarefa conheça as temperaturas calculadas pelas tarefas que tem dados vizinhos
 - As alterações nos dados vizinhos apresenta efeito direto sobre os dados dessa tarefa

Comunicações

- Fatores que precisam ser considerados
 - Há um número de fatores a considerar quando se elabora um programa com comunicações inter-tarefas. São eles:
 - Custo da comunicação
 - Latência X Largura de Banda
 - Visibilidade das Comunicações
 - Comunicação Síncrona X Comunicação Assíncrona
 - Escopo das comunicações
 - Eficiência das comunicações

Comunicações

- **Custo da Comunicação**

- A comunicação entre tarefas implica em sobrecarga
- Ciclos de CPU e outros recursos que poderiam ser utilizados para o cálculo são utilizados para estruturar e transmitir dados
- Comunicações frequentemente necessitam de algum tipo de sincronização entre as tarefas, o que faz com que tarefas fiquem esperando ao invés de fazer trabalho útil
- O tráfego pode saturar a capacidade da rede de comunicação, sinalizando para problemas de desempenho

Comunicações

- **Latência x Largura de Banda**

- Latência é o tempo que leva para enviar uma mensagem de um ponto **A** um ponto **B** e em geral é expressa em microssegundos
- Largura de Banda é quantidade de dados que podem ser transmitidos por unidade de tempo, em geral expressa em Megabits/segundo ou Gigabits/Segundo.
- Enviar muitas mensagens pequenas pode causar sobrecarga de comunicação.
 - Vale a pena empacotar pequenas mensagens em uma mensagem maior, para tornar mais efetivo o uso da banda de comunicação.

Comunicações

- **Visibilidade de Comunicações**
 - Com o ***Modelo de Passagem de Mensagens***, as comunicações são explícitas e em geral bastante visíveis e sobre o controle do programador
 - Como o ***Modelo Paralelo de Dados*** as comunicações frequentemente ocorrem transparentemente ao programador, particularmente em arquiteturas de memória distribuída
 - O programador pode não saber exatamente quando e como as comunicações inter-tarefas estão sendo realizadas

Comunicações

- Comunicação Síncrona x Assíncrona
 - **Comunicação Síncrona** requer algum tipo de *handshaking* entre as tarefas que estão compartilhando dados. Isso pode ser explicitamente estruturado no código pelo programador, ou pode ocorrer em baixo nível sem o conhecimento do programador
 - São bloqueantes, uma vez que outros trabalhos devem esperar até que o processo de sincronização esteja terminado

Comunicações

- Comunicação Síncrona x Assíncrona
 - ***Comunicação Assíncrona***
 - Permite que as tarefas transmitam dados de forma independente. Exemplo: Uma tarefa pode se preparar para enviar uma mensagem para a tarefa 2, e em seguida começar a fazer outras atividades. Quando a tarefa 2 vai receber os dados, não importa.
 - São não-bloqueantes, pois outros trabalhos podem ser realizados enquanto acontecem as comunicações.

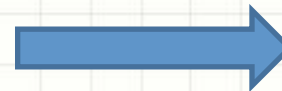
Comunicações

- **Escopo das Comunicações**

- É fundamental saber quais tarefas precisam se comunicar umas com as outras durante a concepção de um código paralelo. Há dois escopos, os quais podem ser implementados de forma síncrona ou assíncrona

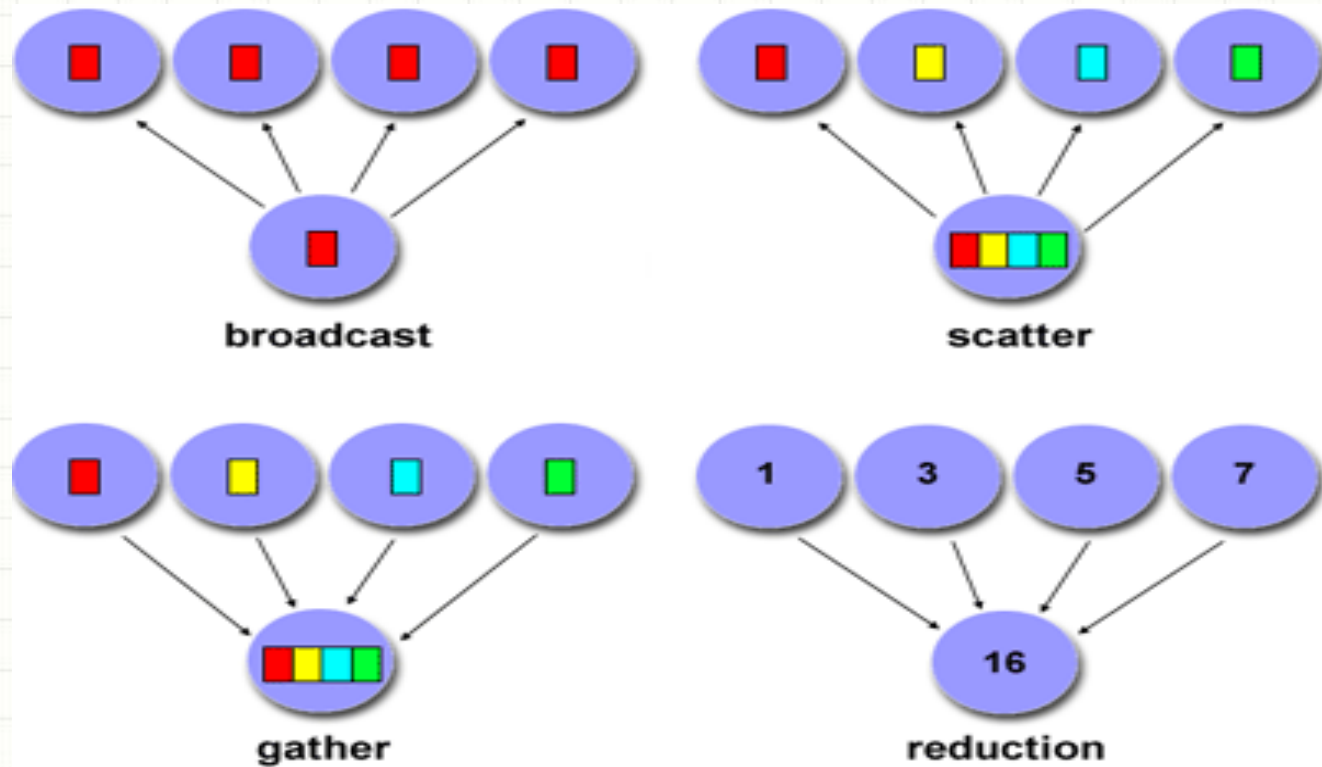
- **Ponto-a-Ponto**: Envolve duas tarefas. Uma é a emissora e outra a receptora de dados
- **Coletivo**: Envolve o compartilhamento de dados entre mais de duas tarefas, que são especificadas como sendo membros de um grupo comum, ou coletivo.

- *Algumas variações:*



Comunicações

- Escopo das Comunicações



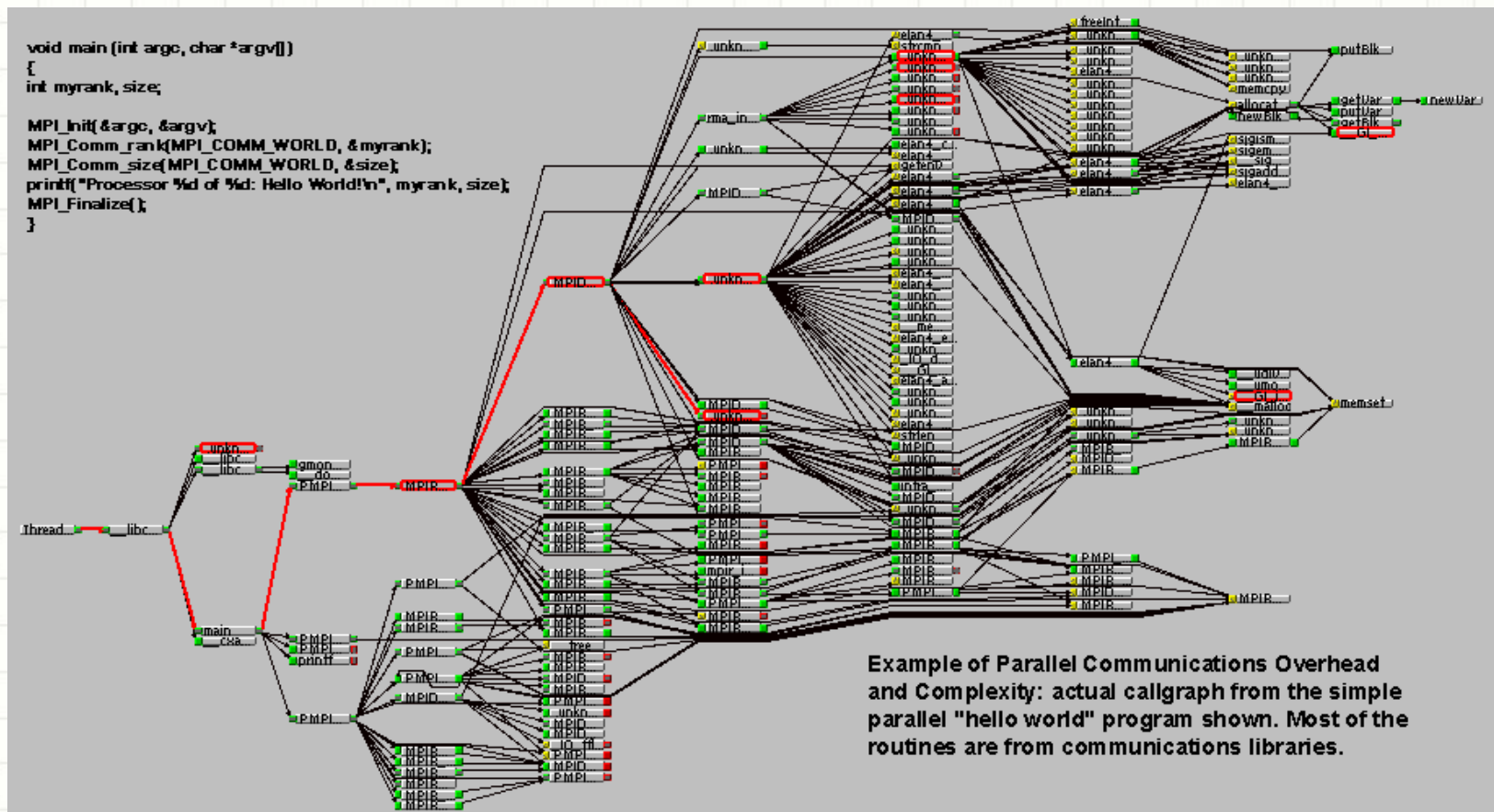
Comunicações

- **Eficiência das Comunicações**

- Muitas vezes o programador terá uma escolha no que diz respeito aos fatores que afetam o desempenho das comunicações. Exemplos:
 - **Que implementação de determinado modelo deve ser usado?**
 - Utilizar o Modelo de Passagem de Mensagens. Como exemplo, uma implementação MPI pode ser mais rápida em uma determinada plataforma de hardware do que em outra
 - **Que tipo de operações de comunicação deve ser utilizado?**
 - As operações de comunicação assíncronas podem melhorar o desempenho geral do programa
 - **Rede**
 - Algumas plataformas podem oferecer mais de uma rede de comunicação. Qual é a melhor?

Comunicações

- Sobrecarga e Complexidade





EXERCÍCIO E LEITURA RECOMENDADA

Exercício

- Acessar o Moodle

Leitura Recomendada

- Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar - 2ª ed., Addison Wesley
- Introduction to Parallel Computing
 - https://computing.llnl.gov/tutorials/parallel_comp/

Dúvidas



Próxima Aula...

- Ferramentas de Apoio ao Desenvolvimento de Aplicações Concorrentes