
embarcadero
Embarcadero Conference


embarcadero
Threads no meu ERP
Chegou a hora!
Embarcadero Conference

Palestrante

Dion Mai

Instrutor
Consultor
Desenvolvedor

Graduando em Eng. de Controle e Automação

O que você verá?!

- Motivos para usar Threads agora e no futuro;
- Como usar Threads;
- Principais vantagens e desvantagens;
- Demais técnicas.

Sobre a Palestra

❏ Delphi e C++ Builder

❏ Todos fontes no Git

❏ ppt no SlideShare

*Links no final da palestra

Por que usar Threads?

- Sem Threads

- Com Threads

Por que usar Threads?

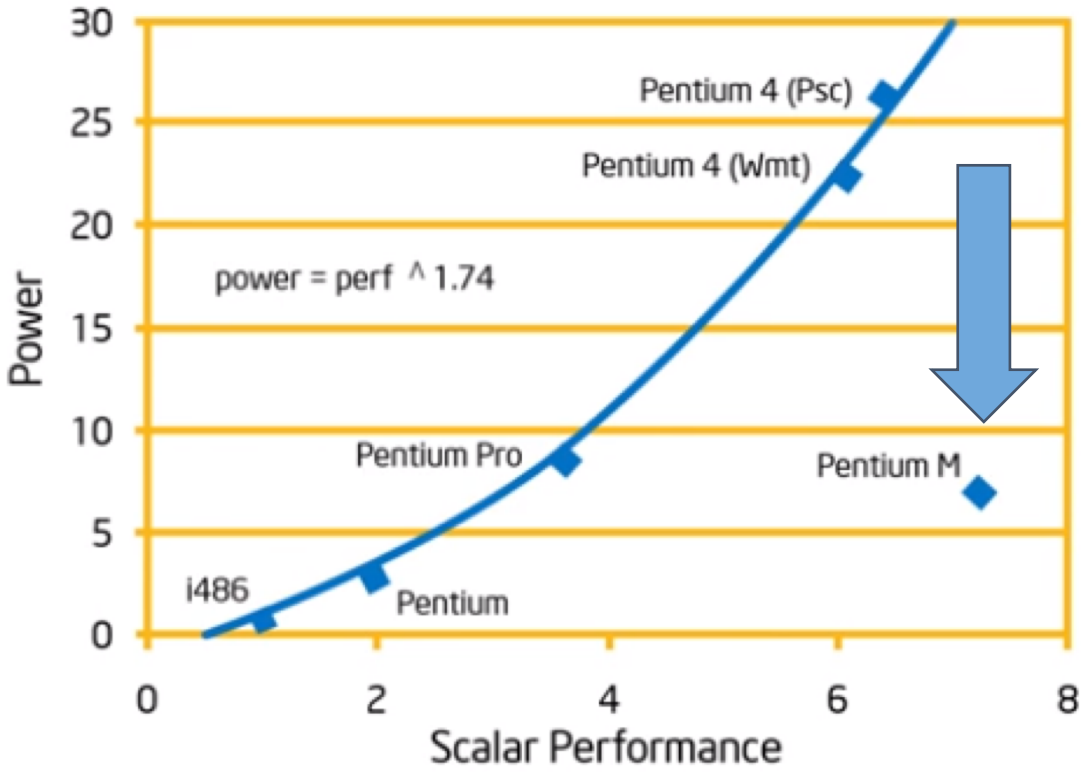
- **UX**
- Tempo processamento
- **Bateria**
- Hardware (melhor uso e integrações)

= +Resultado

Exemplo de Impacto no UX

Sobre a Bateria

Partial Solution: Simple Low Power Cores



Mobile CPUs With Shallow Pipelines Use Less Power

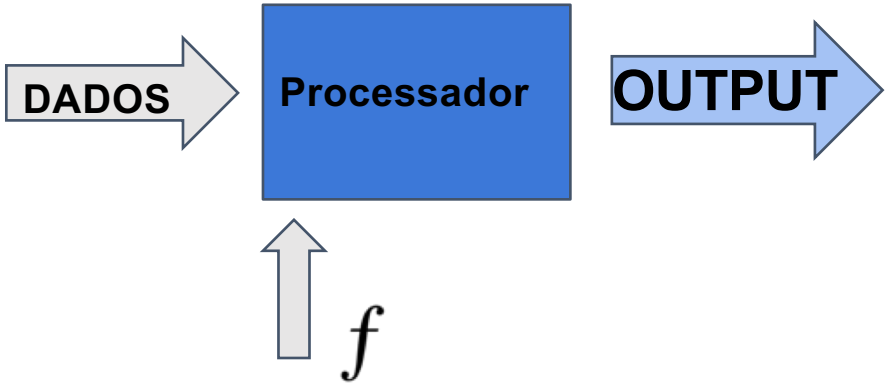
Source: E. Grochowski of Intel

Sobre a Bateria

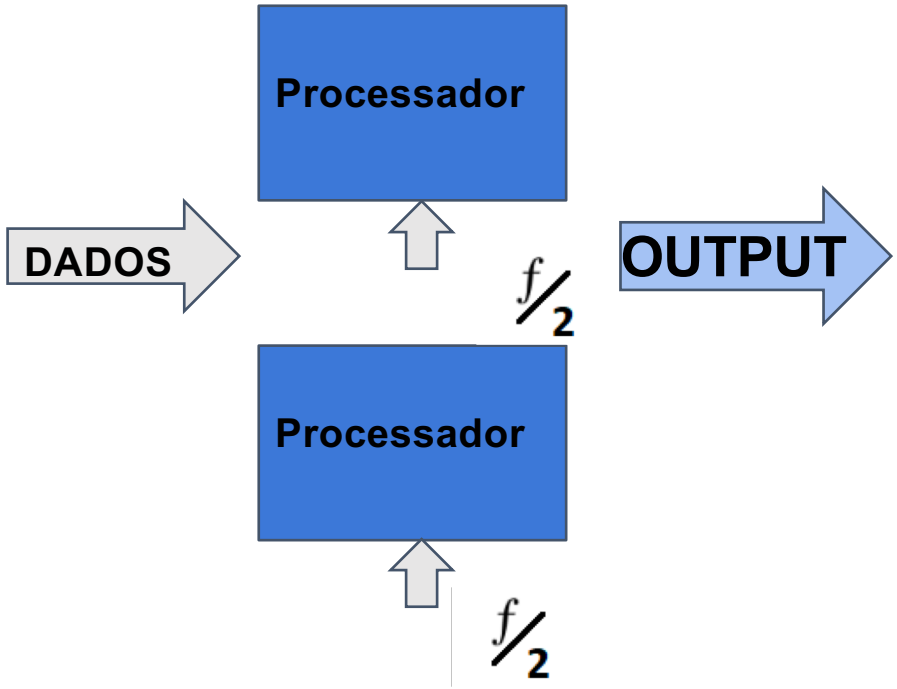
$$\text{Power} = CV^2F$$

Sobre a Bateria

Single-CORE



Multi-CORE



Sobre a Bateria

TABLE IV
RESULTS FOR THE WAVELET CHIP (1.2μm CMOS TECHNOLOGY)

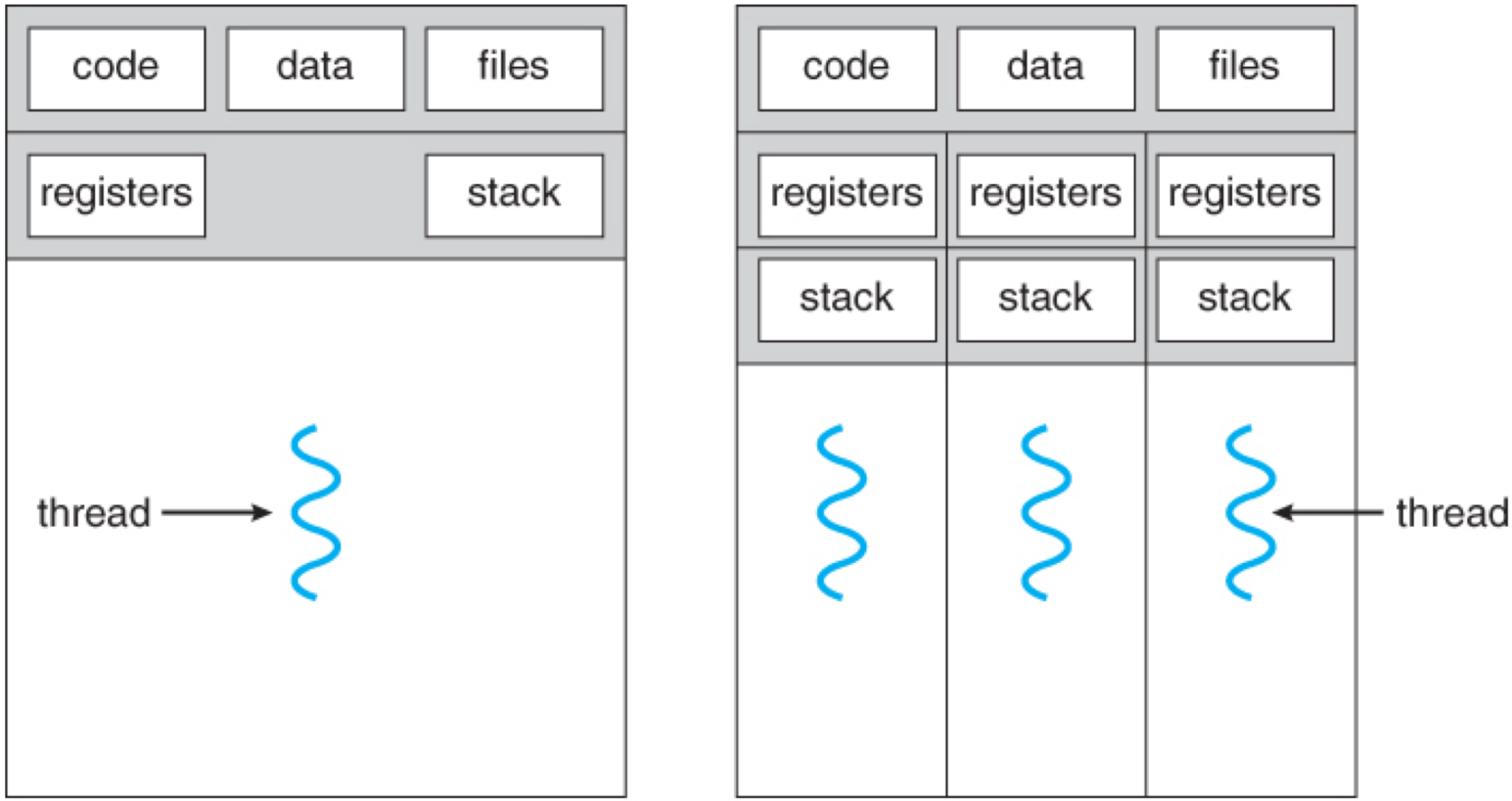
Version	$f_{samling}$	# of Control Cycles / Clock Period	Supply Voltage	Total Average Capacitance	Power, mW	Area, mm^2
Initial	1.5Mhz	22 / 30ns	5V	2870pF	107mW	8.5 mm^2
Optimized for Power	1.5Mhz	3 / 220ns	1.5V	1735pF	5.8mW	62.9mm^2

*Referências bibligráficas no final

Definição

O que é uma Thread?

Definição



single-threaded process

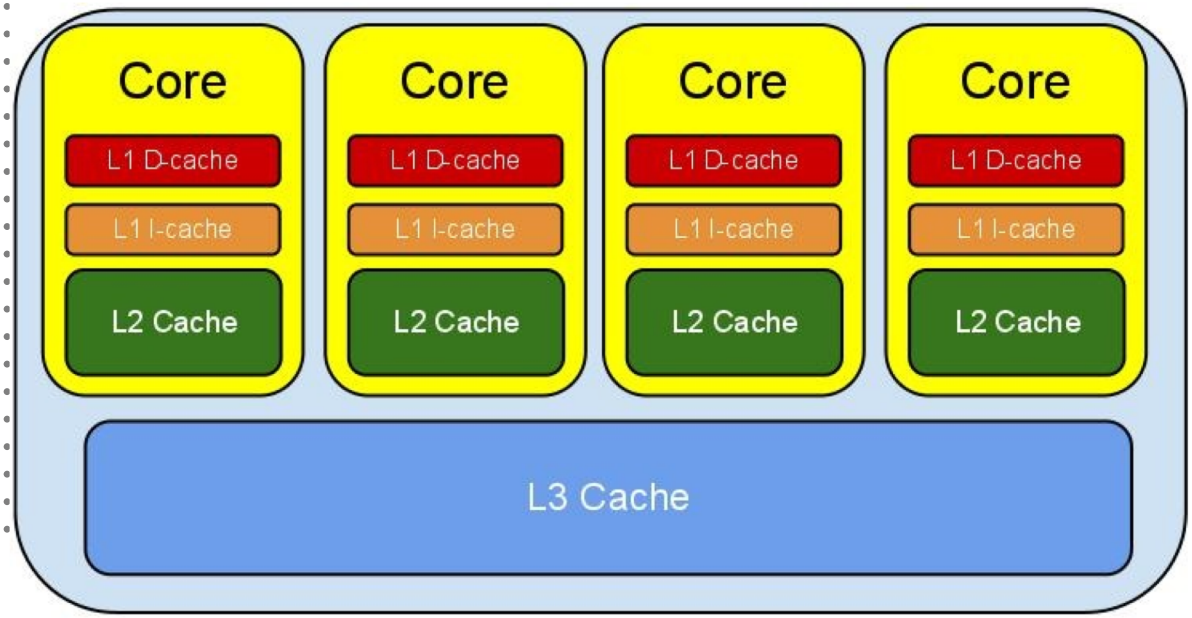
multithreaded process

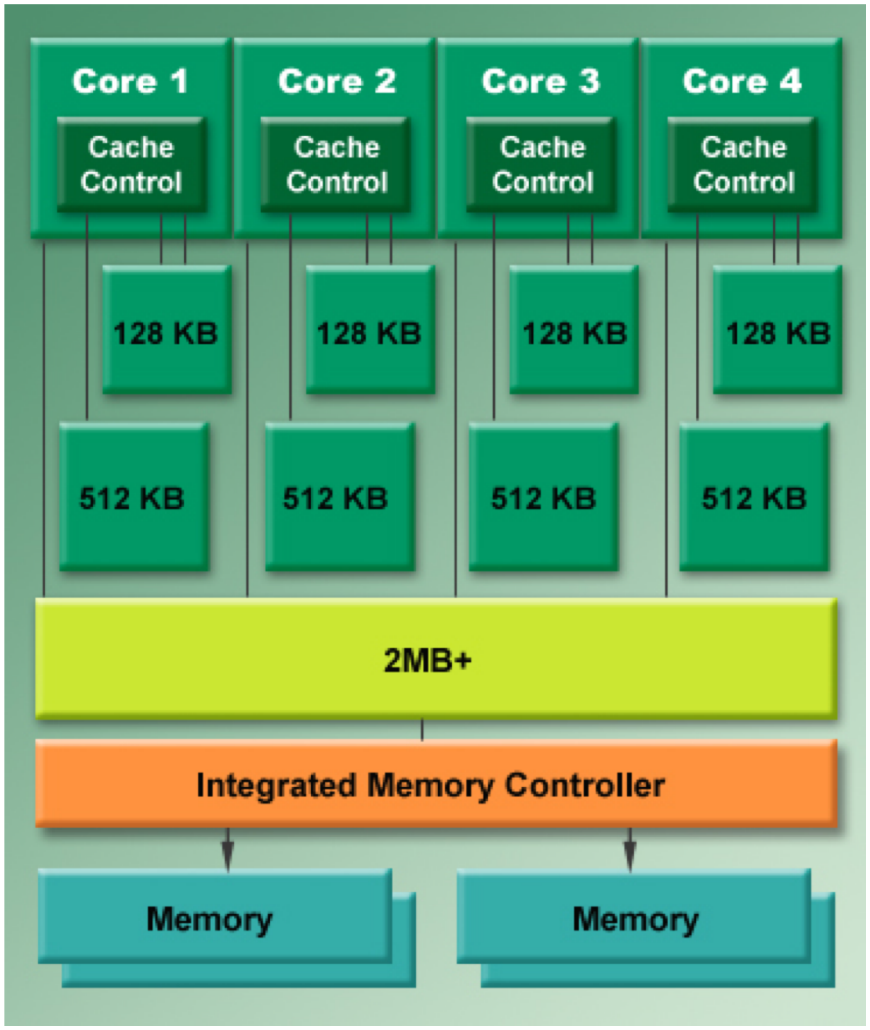
Conceito

Concurrent, non-parallel execution

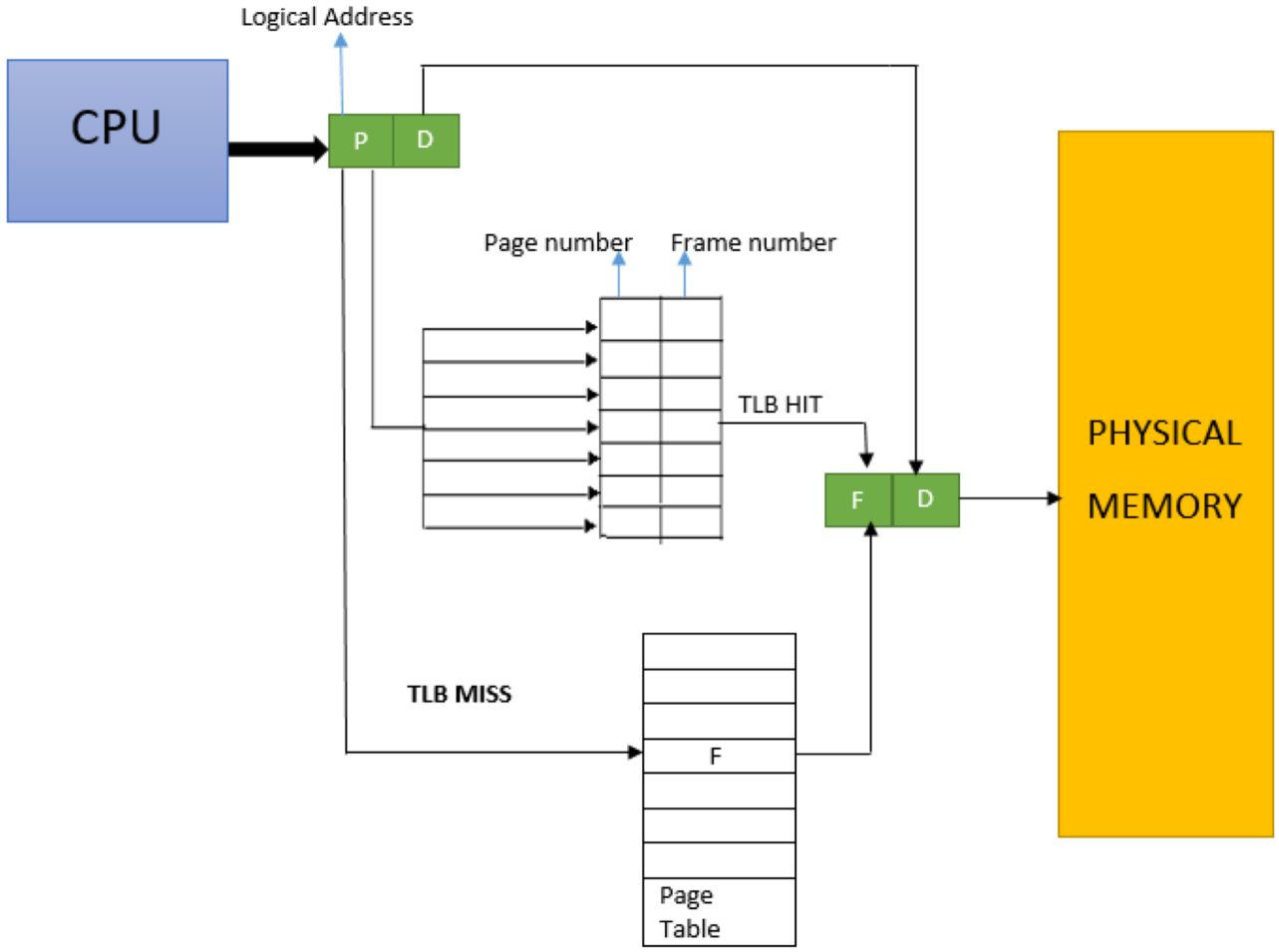
Intel, https://www.youtube.com/watch?v=6jFkNjhJ-Z4

Conceito


Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
Core
L1 D-cache
L1 I-cache
L2 Cache
L3 Cache


Core 1
Cache Control
Core 2
Cache Control
Core 3
Cache Control
Core 4
Cache Control
128 KB
128 KB
128 KB
128 KB
512 KB
512 KB
512 KB
512 KB
2MB+
Integrated Memory Controller
Memory
Memory

Conceito


Logical Address
CPU
P
D
Page number
Frame number
TLB HIT
F
D
PHYSICAL
MEMORY
TLB MISS
F
Page
Table

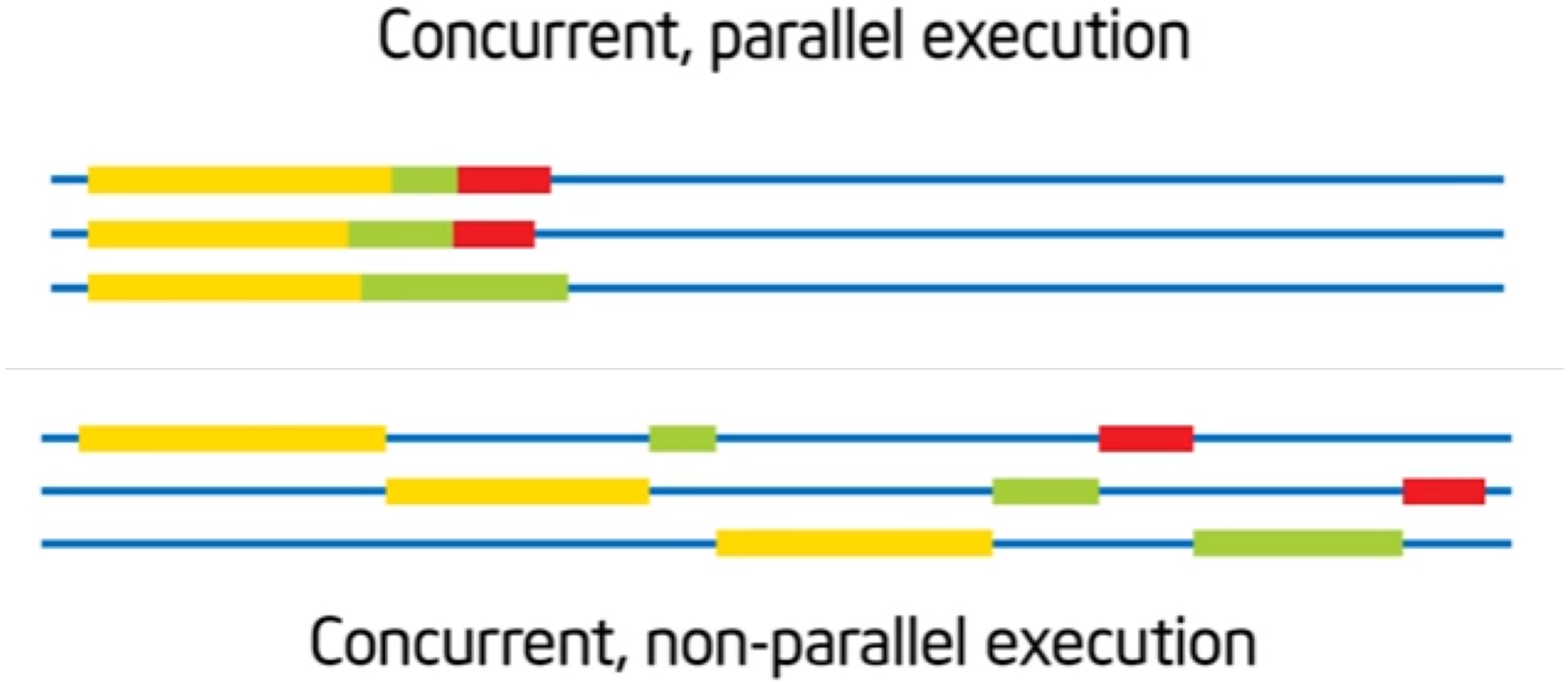
Conceito

Thread(Concurrency) Vs Parallelism

Thread <> Paralelismo

>> Threads facilitam o paralelismo.

Conceito

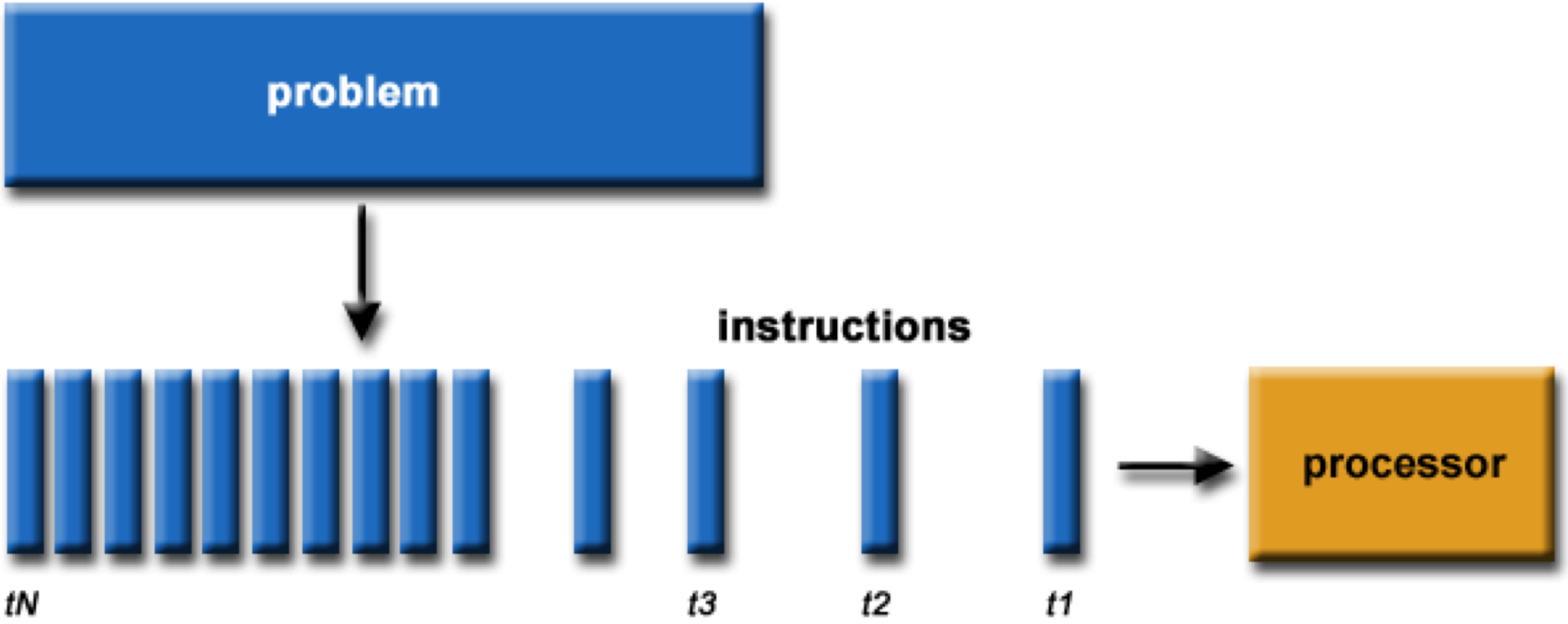


Intel, https://www.youtube.com/watch?v=6jFkNjhJ-Z4

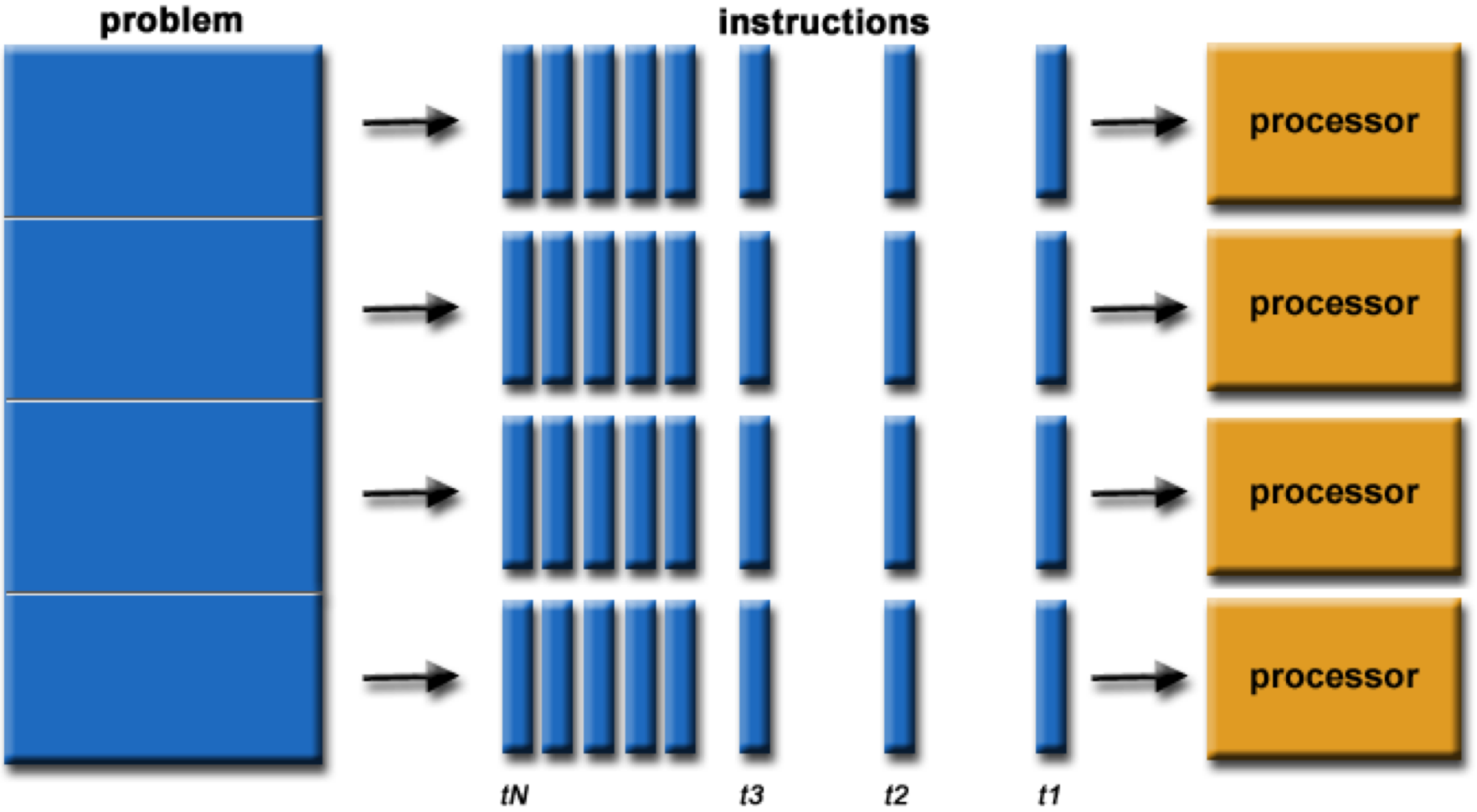
Conceito

Como utilizar?

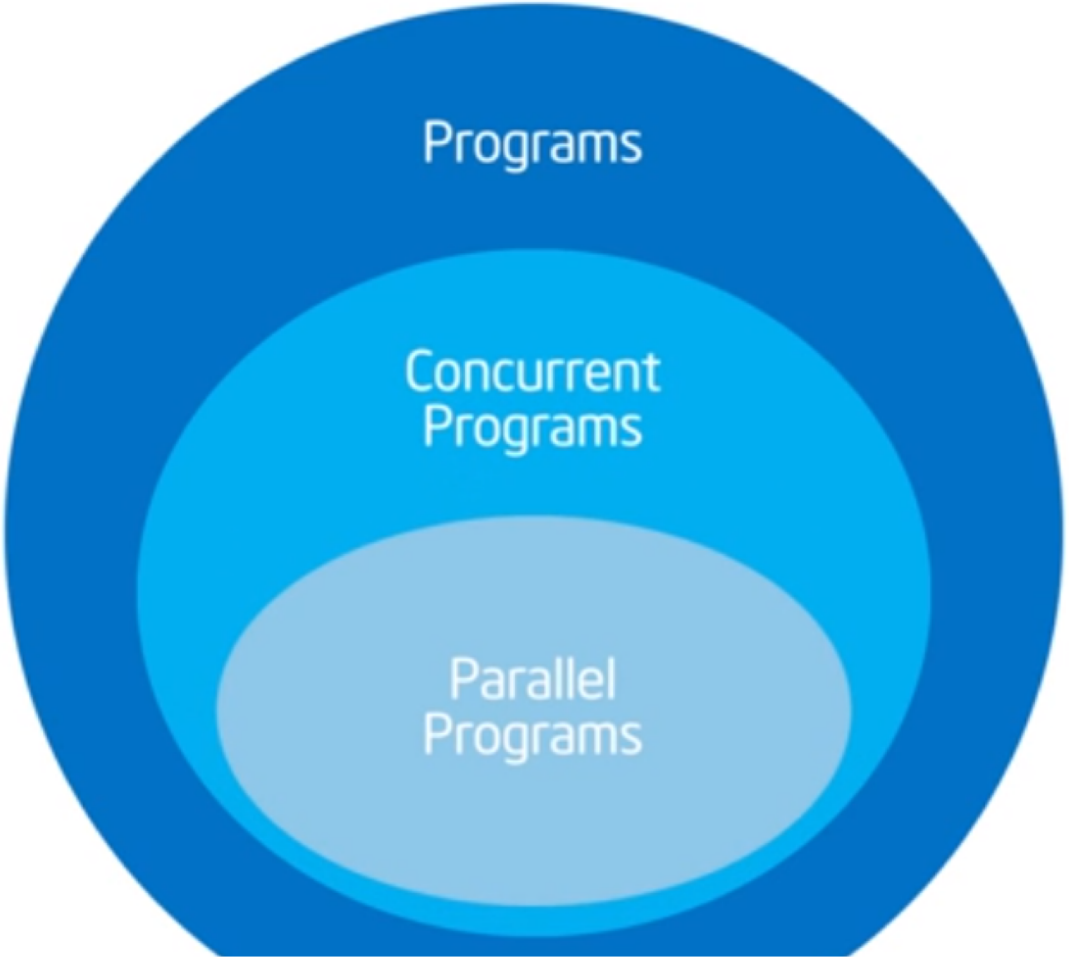
Conceito



Conceito


problem
instructions
processor
processor
processor
processor
tN
t3
t2
t1

Conceito



Propriedades/Características

- Outra linha de processamento dentro do mesmo processo
- Própria stack e registradores
- Compartilha recursos com o processo “main”

- É eliminada da memória quando processo finaliza
- Não deve acessar recursos compartilhados para escrita
- Não deve acessar recursos de renderização de tela

Base para utilizar

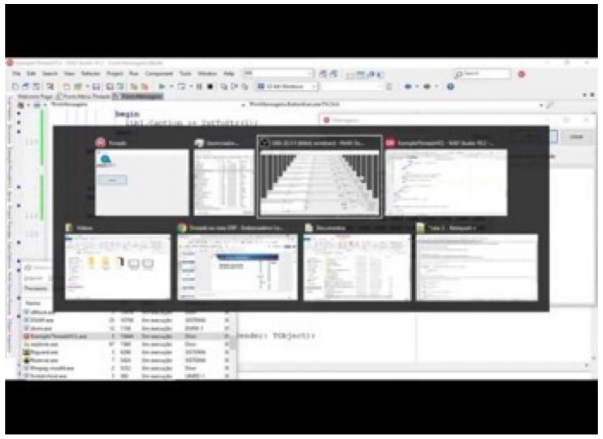
Principais técnicas e conhecimentos para usar Threads:

- O que é Syncronize()
- Como funciona a Memória?
- O que acontece com os erros?
- Programas e componentes externos
- Mensagens

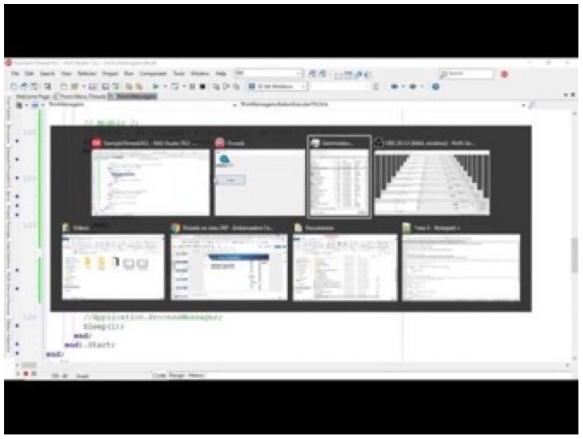
O que é Synchronize?

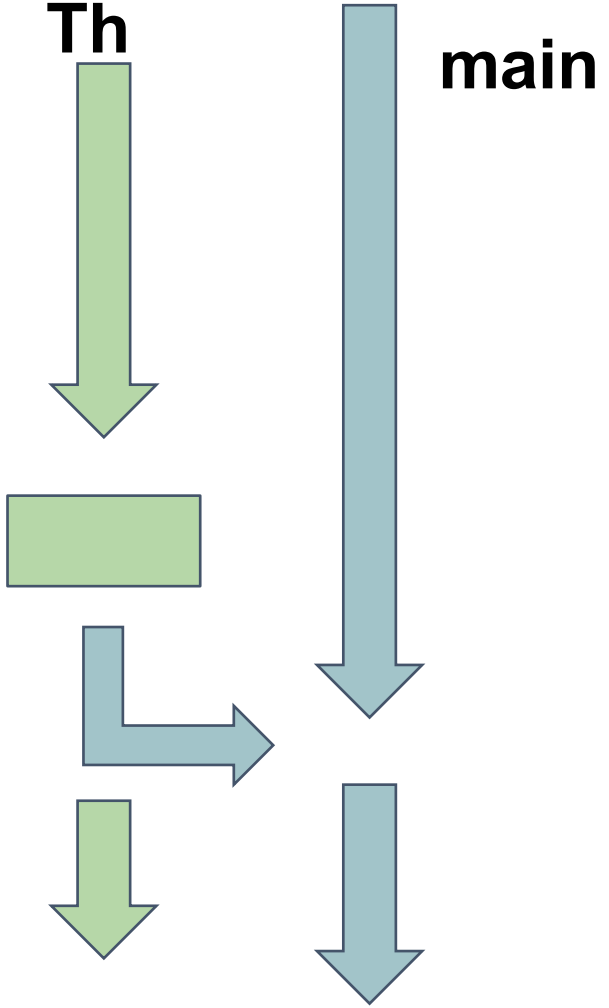
“Serealiza” execução de volta para a main Thread

Sem Sync



Com Sync





O que é Syncronize?

Quando usar?

Quando pode ser problema?

Como funciona a memória?

Vários formatos existem, porém, o mais utilizado é o

“Multiple read exclusive write”

No que isso implica?

Como funciona a memória

Como resolver?

TInterlocked

- **Mutex**
- **Semaphore**
- **Critical Section**

Synchronize

O que acontece com os erros?

Para onde eles foram?

Componentes e Bibliotecas externas

Componentes de acesso DB

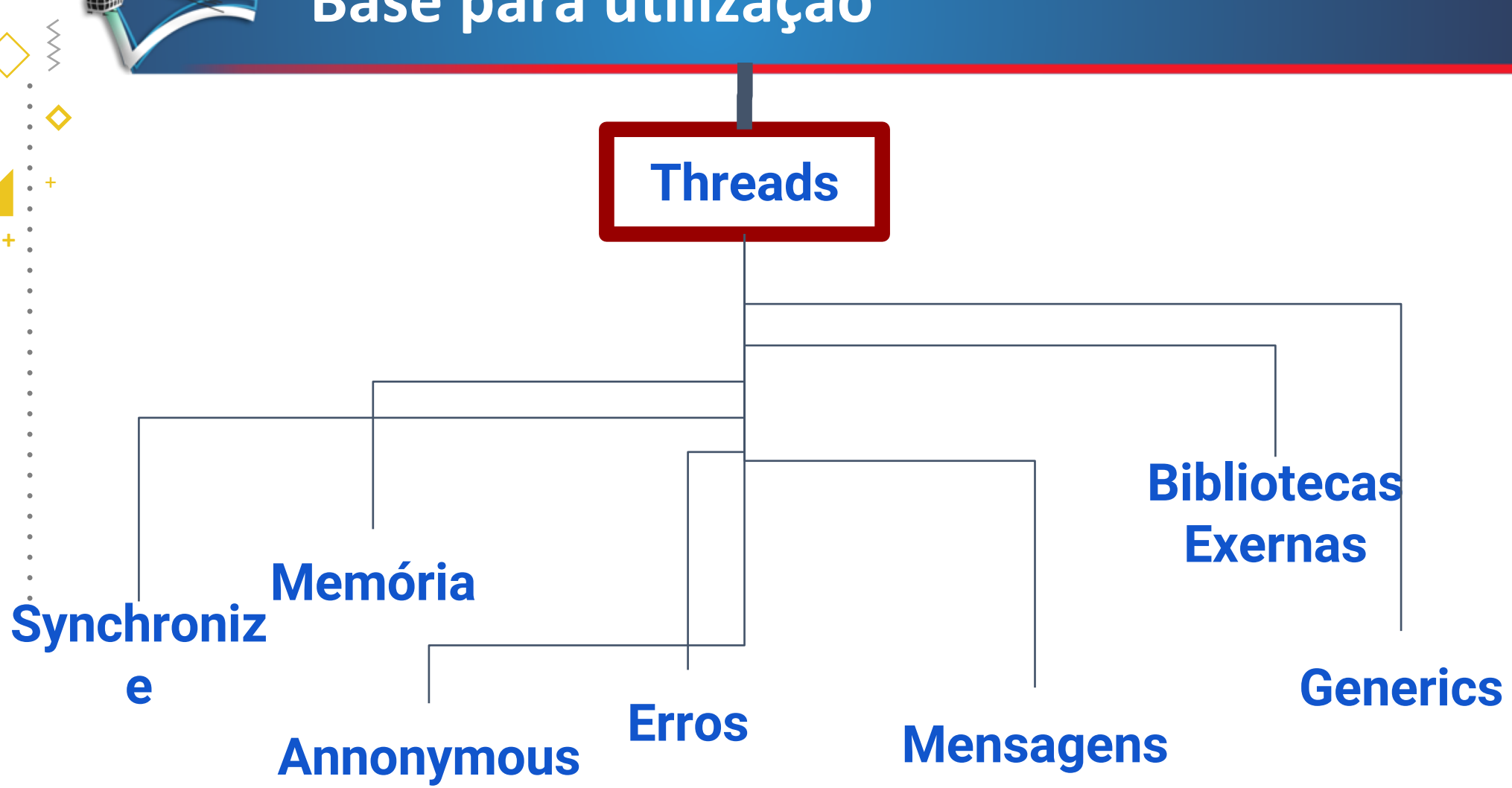
Componentes de Relatórios

Acesso a hardware

Toda biblioteca, programa, API ou componente deve ser verificado

Ou seja, não escrevi o fonte, tenho que verificar com o fabricante se é “Thread Safe”

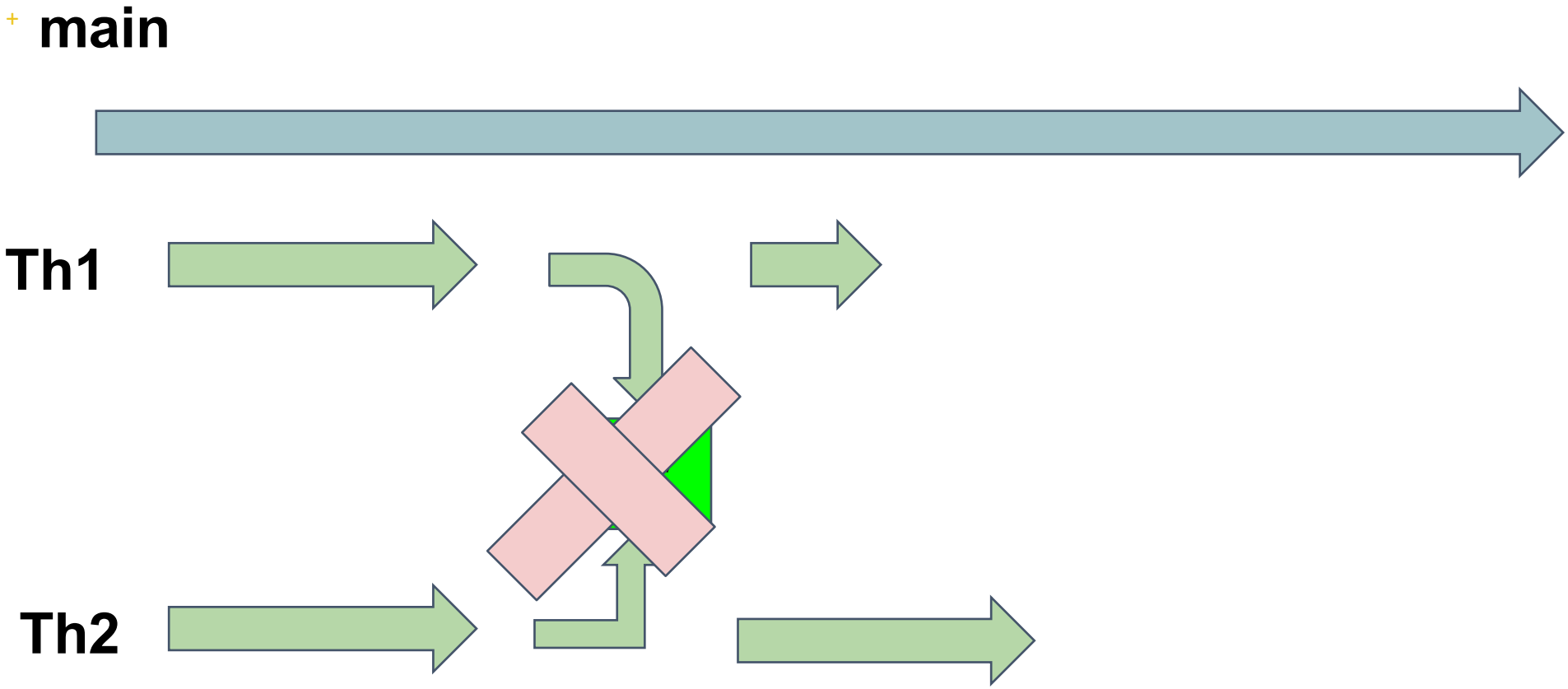
Base para utilização



Principais Problemas

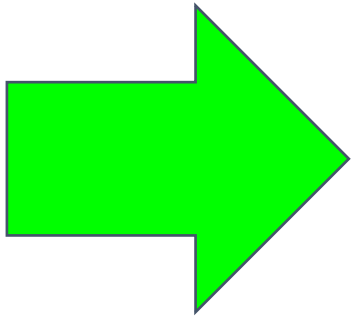
- Race condition
- AV
- Deadlock
- Bottleneck

Técnicas para contorno



Técnicas para contorno

- Race condition
- AV
- Deadlock
- Bottleneck



- Interlok (Win)
- Mutex
- Semaphore
- Critical Section

Técnicas para contorno

Tela exemplo, usando Produtor/Consumidor

Entendido!

Fácil, né?!

Agora tudo é Thread!!

Só que não...

Threads

Quando Usar:

- Processamento não envolve interface
- O tempo de cópia dos dados envolvidos no processamento não é maior do que o tempo de processamento
- A UX não será afetada prejudicialmente
- As soluções parciais do problema não invalidam o path de processamento

Threads

Como usar:

- Redesenhar o **processo** da tela(UX)/rotina afetada
- Implementar primeiro a funcionalidade, depois “paralelizar”
- Escolher a melhor técnica de paralelismo
  - Mutex, Semaphore ou Critical Section
  - Envolver Synchronize
  - Alocar memória própria para Thread ou copiar blocos?

Humm...não deu, não consegui definir qual e melhor técnica :/

Simplifica: usa um outro “processo”...

Quando não usar

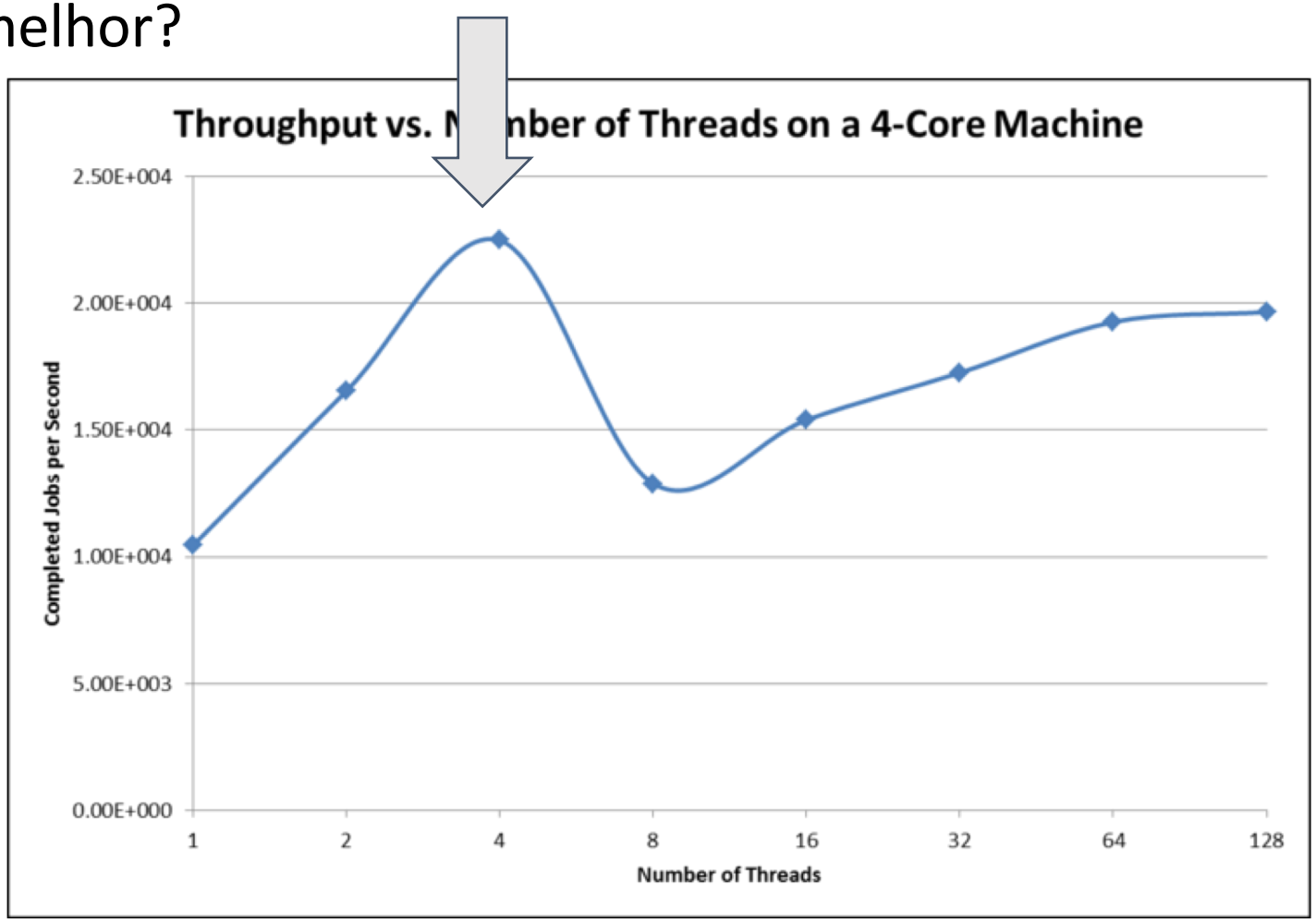
- Bloco único de memória para várias escritas
- Desconhecimento sobre componente, hardware ou biblioteca externa
- Desenho em tela
- Muitas regiões críticas ou pontos de sincronia

Threads

Threads = Performance?

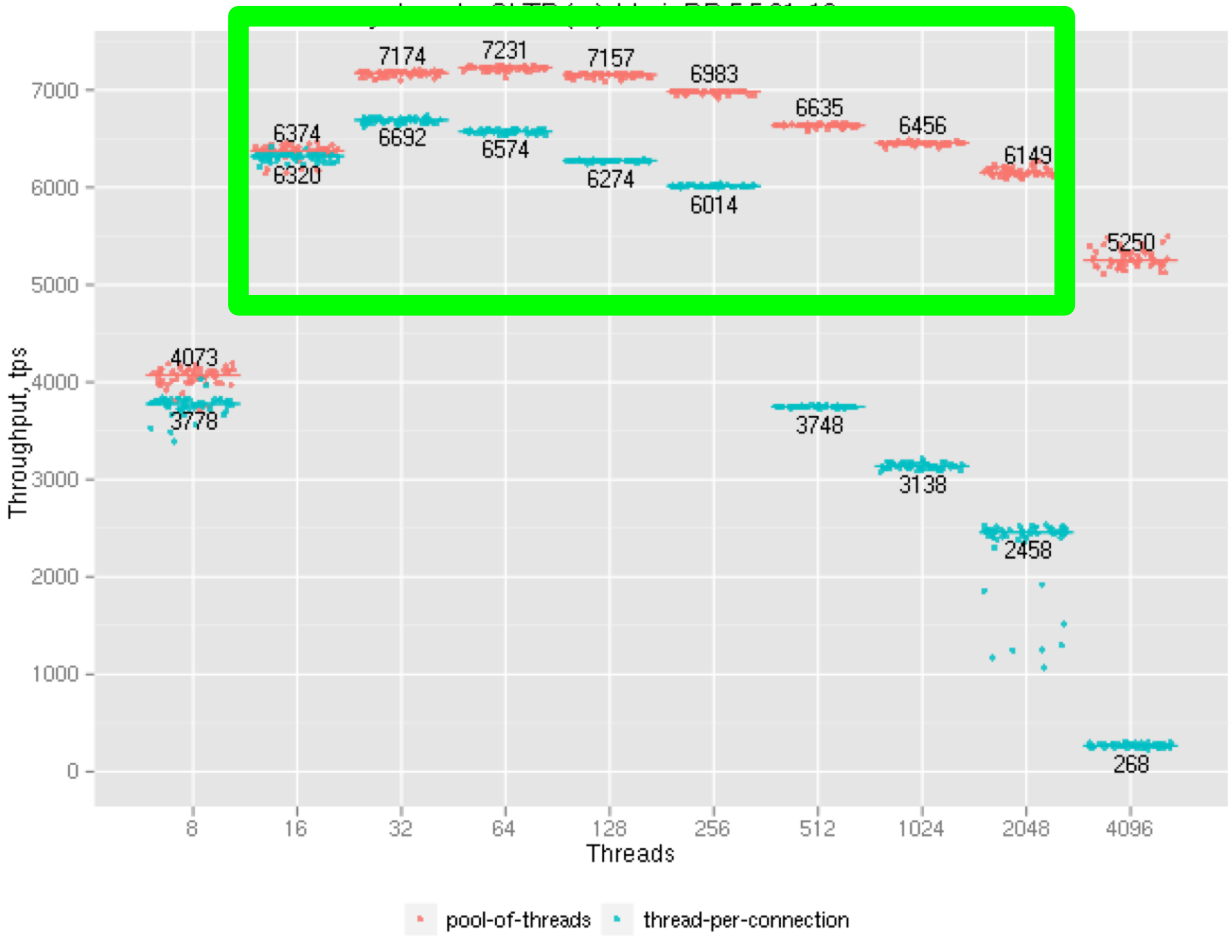
Threads

Mais é realmente melhor?



Threads

Mais é realmente melhor?



Threads

Performance =

- **Número de Threads**
- **Hardware**
- **Design do processo**

Half way to go

Querem rever algum exemplo?

Ok, mas não me bastou, quero mais recursos!

Próximos Passos

- Experiência em UX e Design de processos
- Bibliotecas externas:
  - MPI
  - OpenMP
  - pthreads

Desing e UX

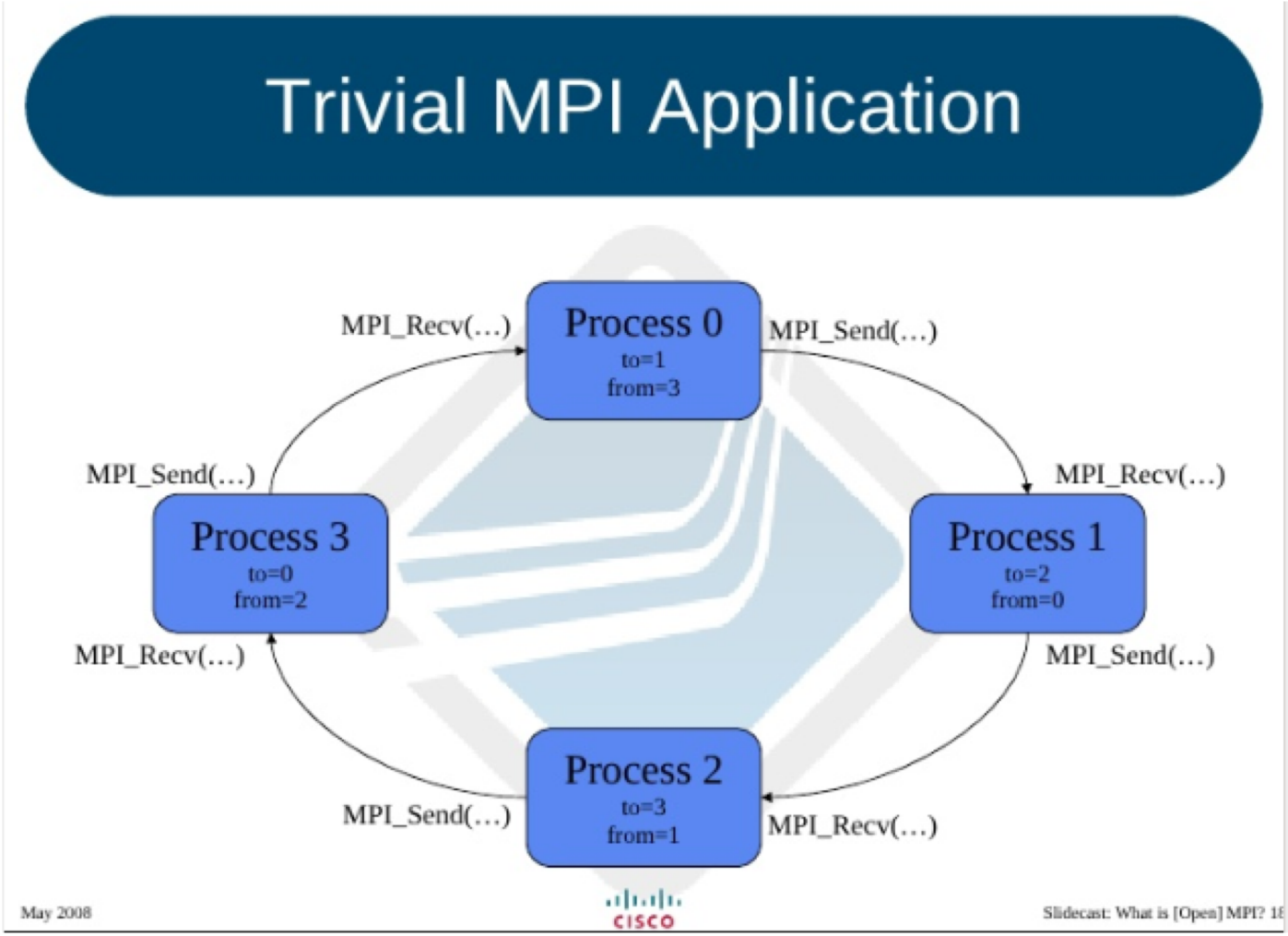
- Cursos mais práticos
- Conhecer usuário
- Conhecer processo

Bibliotecas Externas

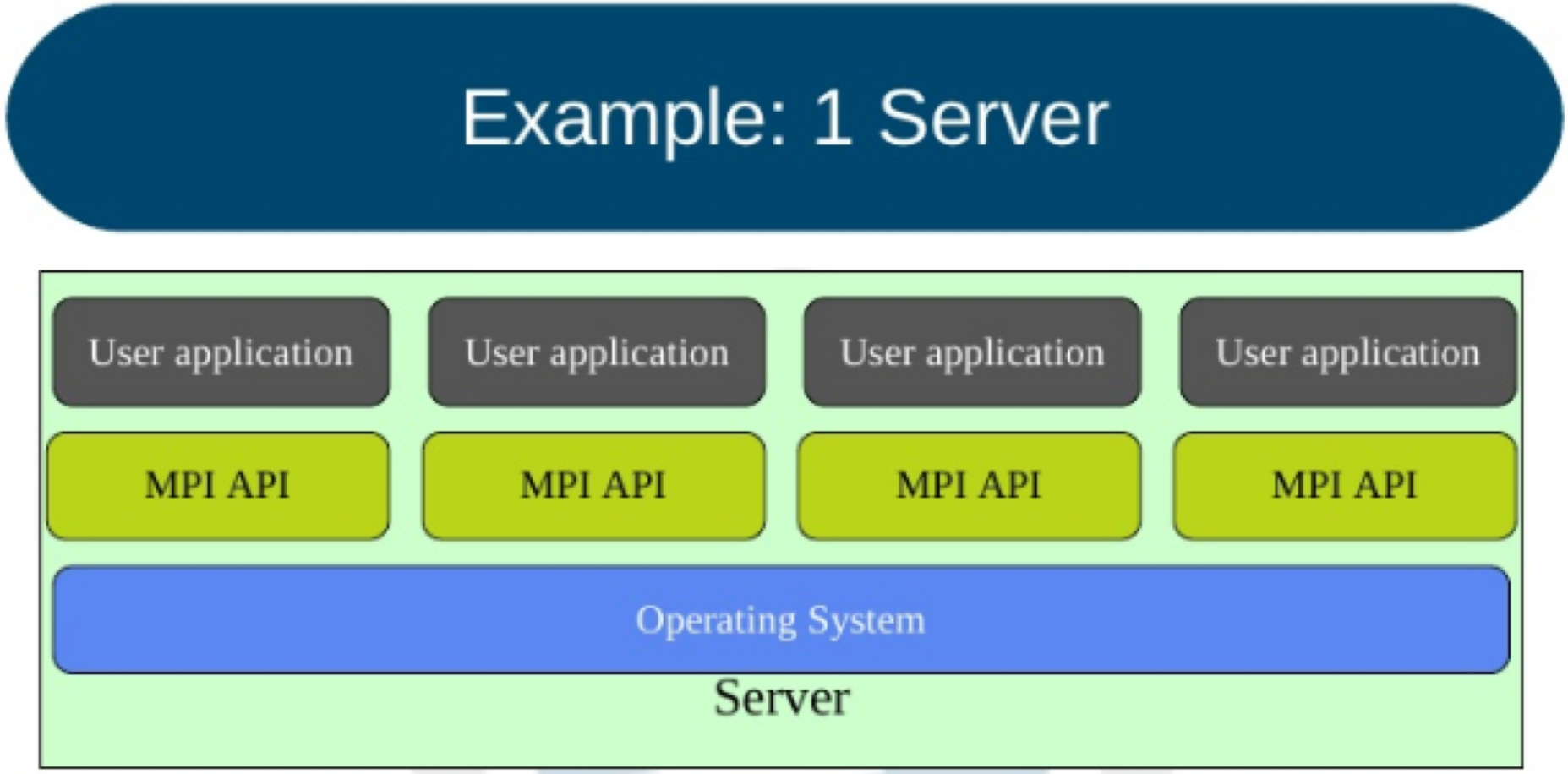
Open MPI(Message Passing Interface)

https://www.open-mpi.org/

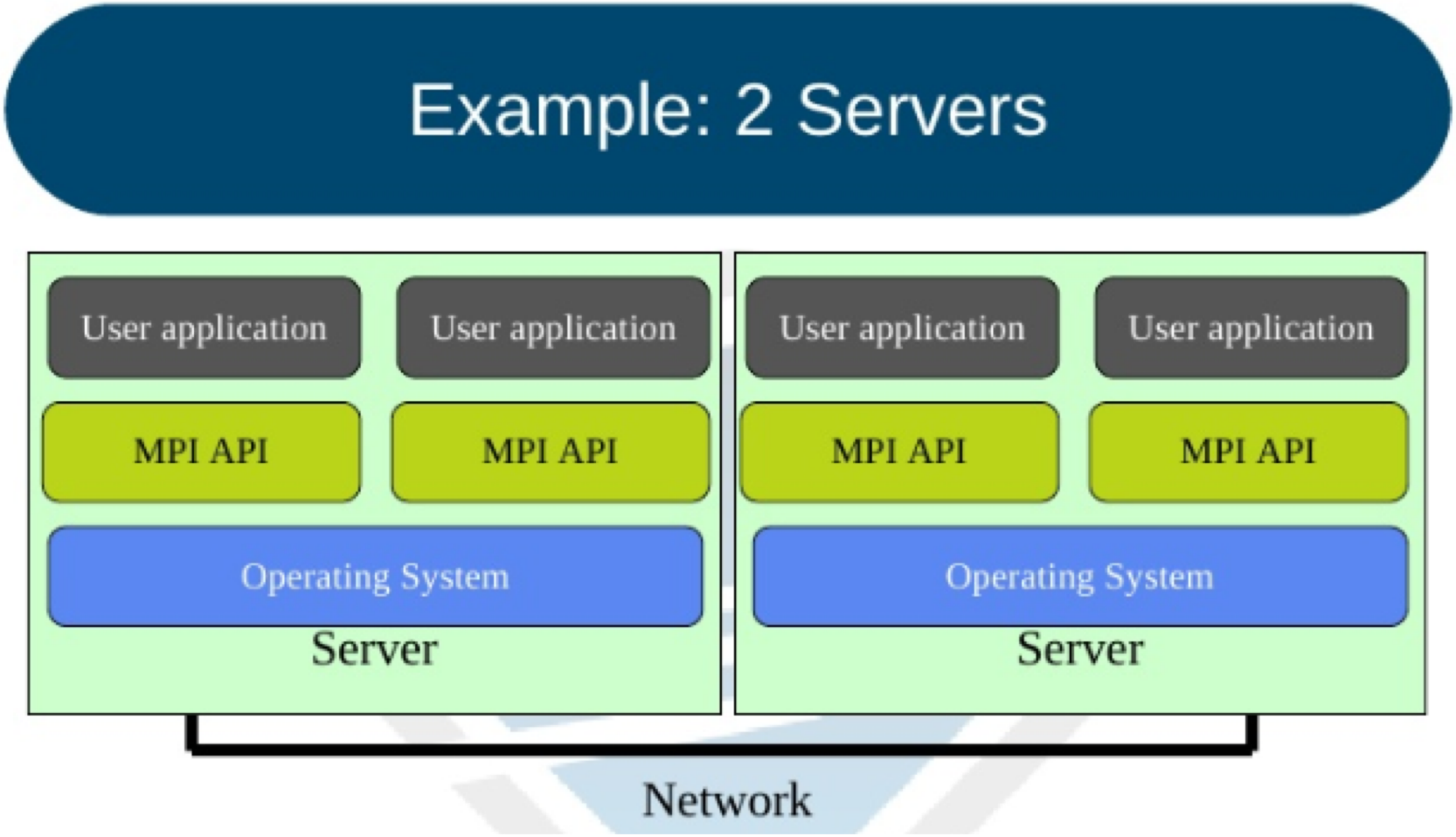
Open MPI


Trivial MPI Application
Process 0
to=1
from=3
MPI_Recv(…)
MPI_Send(…)
Process 1
to=2
from=0
MPI_Recv(…)
MPI_Send(…)
Process 2
to=3
from=1
MPI_Send(…)
MPI_Recv(…)
Process 3
to=0
from=2
MPI_Send(…)
MPI_Recv(…)
May 2008
CISCO
Slidecast: What is [Open] MPI? 18

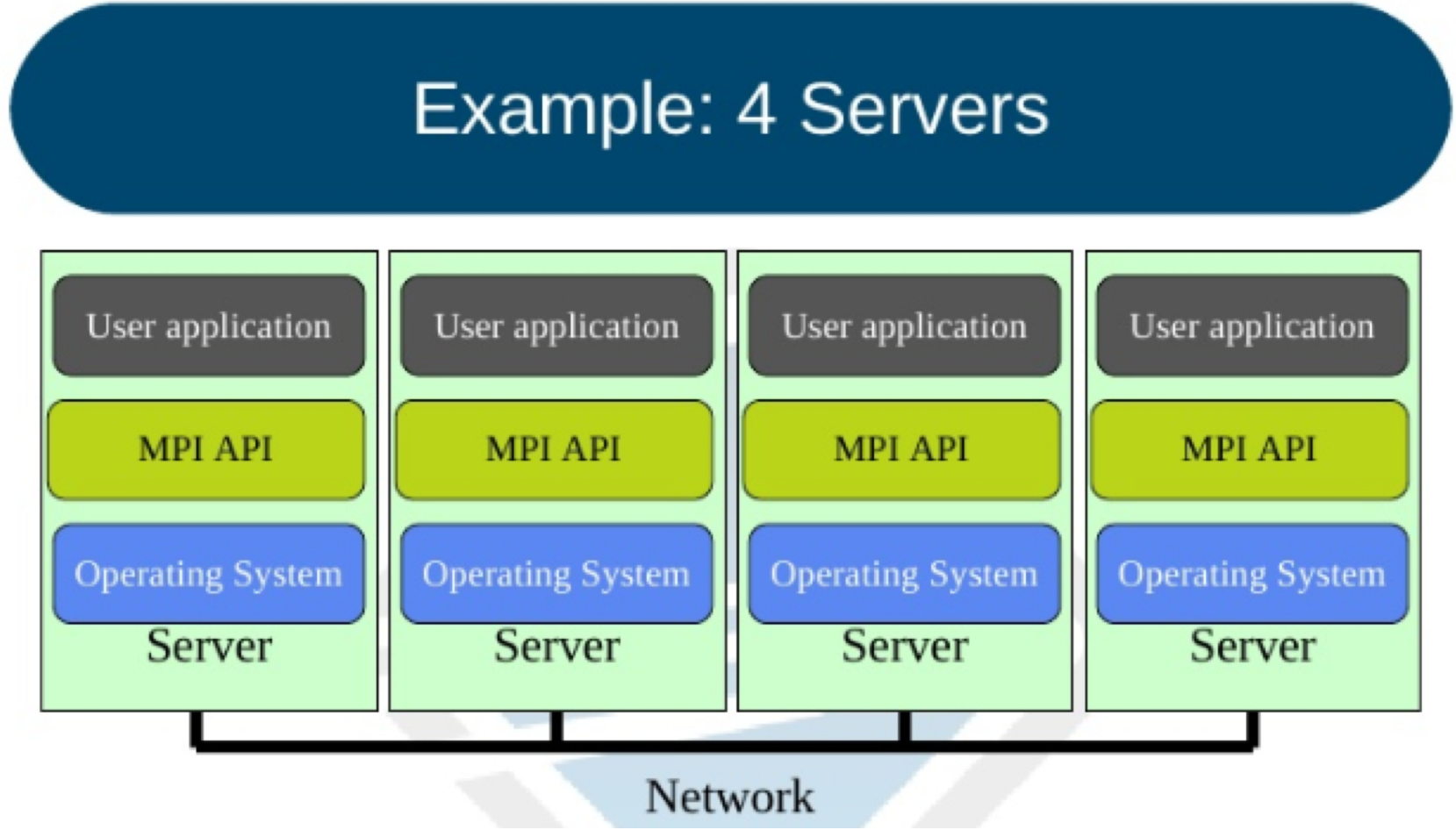
Open MPI


Example: 1 Server
User application
User application
User application
User application
MPI API
MPI API
MPI API
MPI API
Operating System
Server

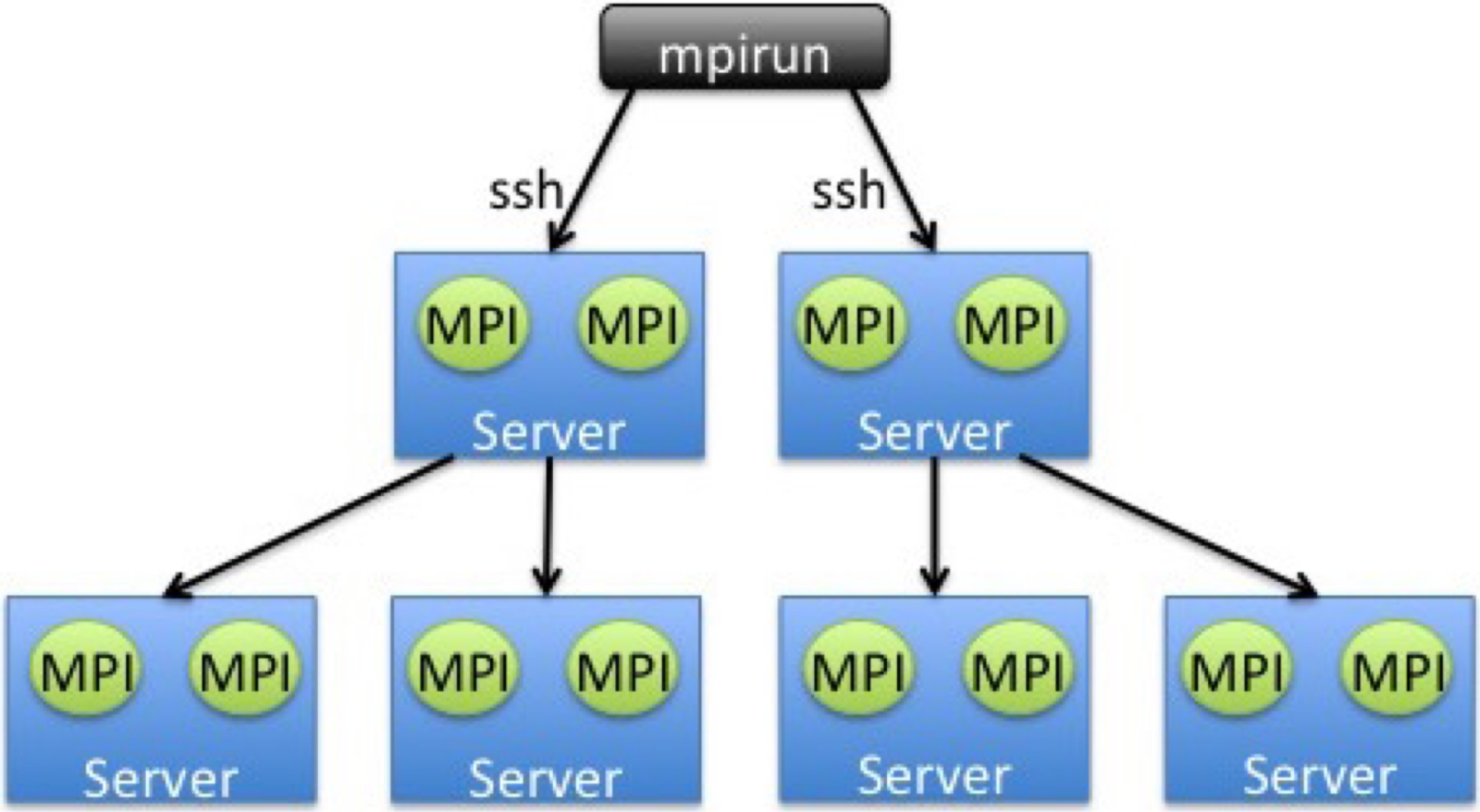
Open MPI


Example: 2 Servers
User application
User application
MPI API
MPI API
Operating System
Server
User application
User application
MPI API
MPI API
Operating System
Server
Network

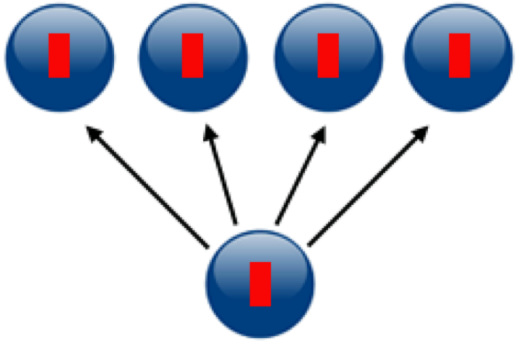
Open MPI


Example: 4 Servers
User application
MPI API
Operating System
Server
User application
MPI API
Operating System
Server
User application
MPI API
Operating System
Server
User application
MPI API
Operating System
Server
Network

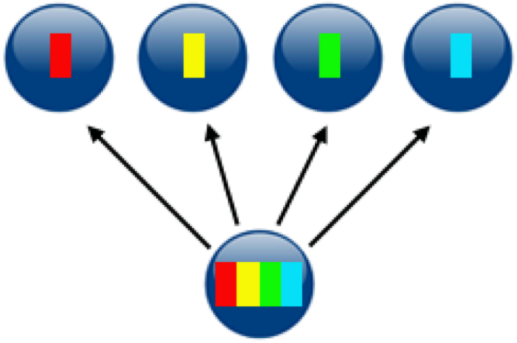
Open MPI


mpirun
ssh
ssh
MPI
MPI
Server
MPI
MPI
Server
MPI
MPI
Server
MPI
MPI
Server
MPI
MPI
Server
MPI
MPI
Server

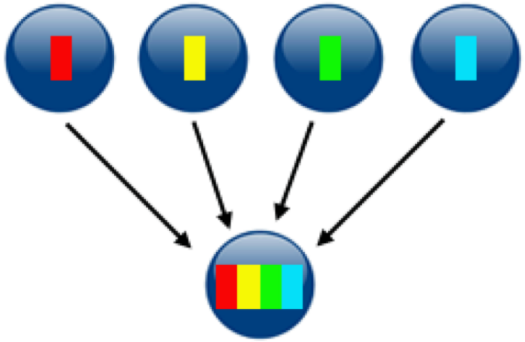
Open MPI



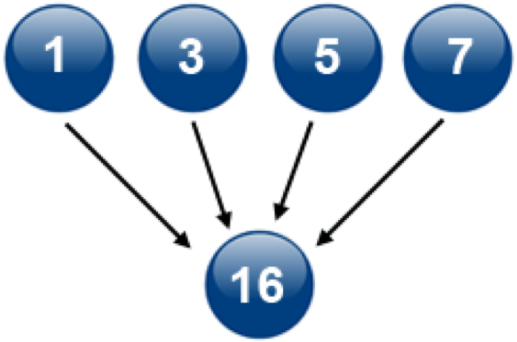
broadcast



scatter



gather

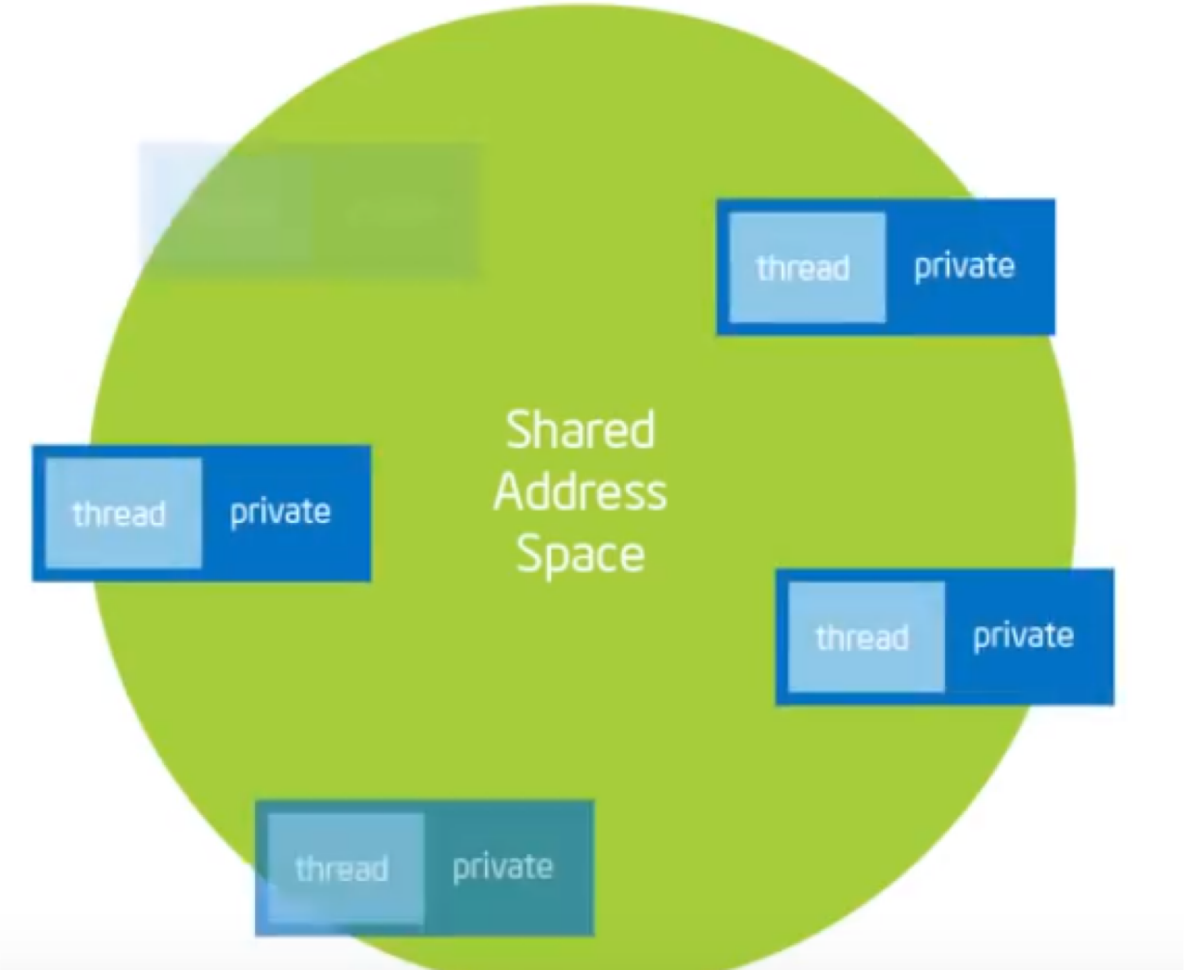


reduction

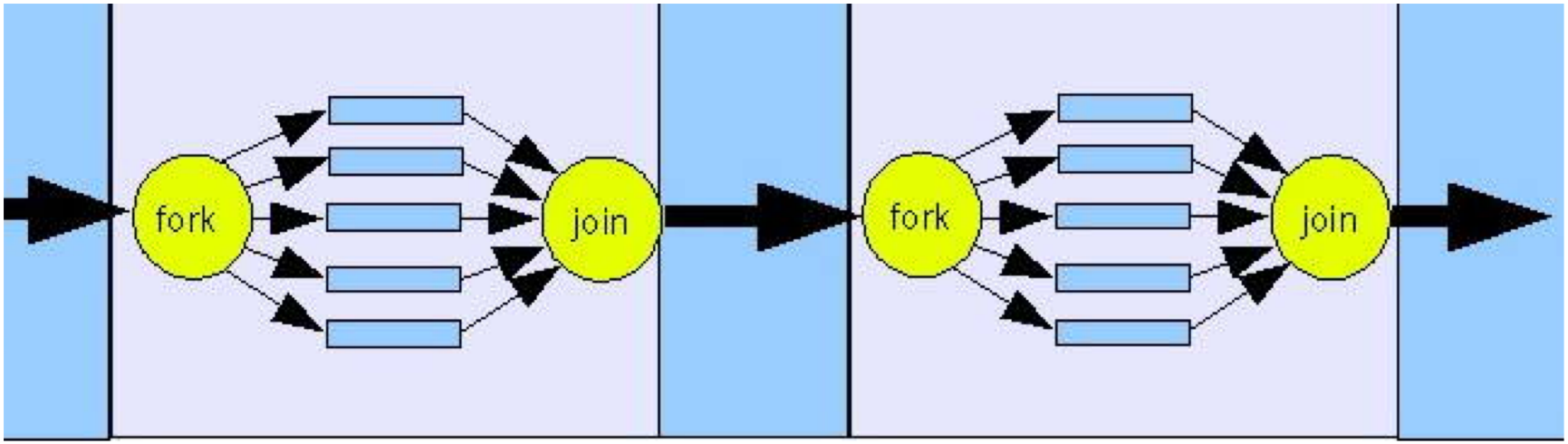
Open Multi-Processing

https://www.openmp.org/

OpenMP


thread
private
Shared
Address
Space
thread
private
thread
private
thread
private

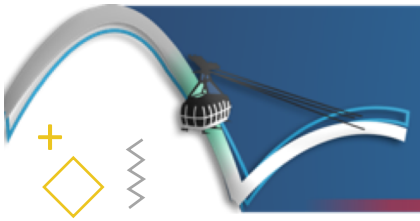
OpenMP



OpenMP

```
#include "omp.h"
 main() {
   #pragma omp parallel shared(h) {
   #pragma omp for private(x) reduction(+:sum) for  (i = 0;  i < N;  i++)
   { x = h * (i + 0.5);
     sum = sum + f(x);
    }

    printf("PI = %f\n",  sum / N);
}
```

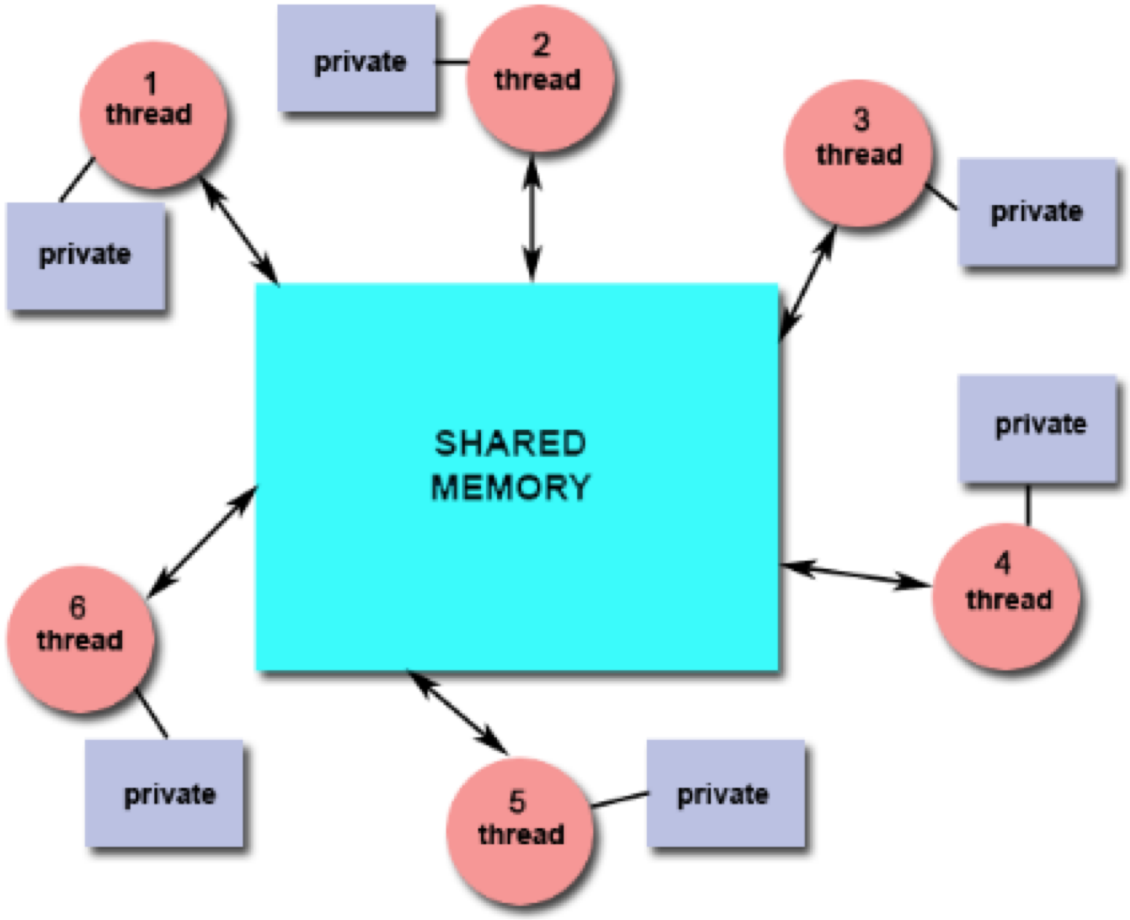


Pthread

POSIX Threads

POSIX Threads is an API defined by the standard *POSIX.1c, Threads extensions*

Pthreads


1
thread
private
private
2
thread
3
thread
private
SHARED
MEMORY
private
4
thread
6
thread
private
5
thread
private

Pthreads

Vantagens sobre Threads normais:

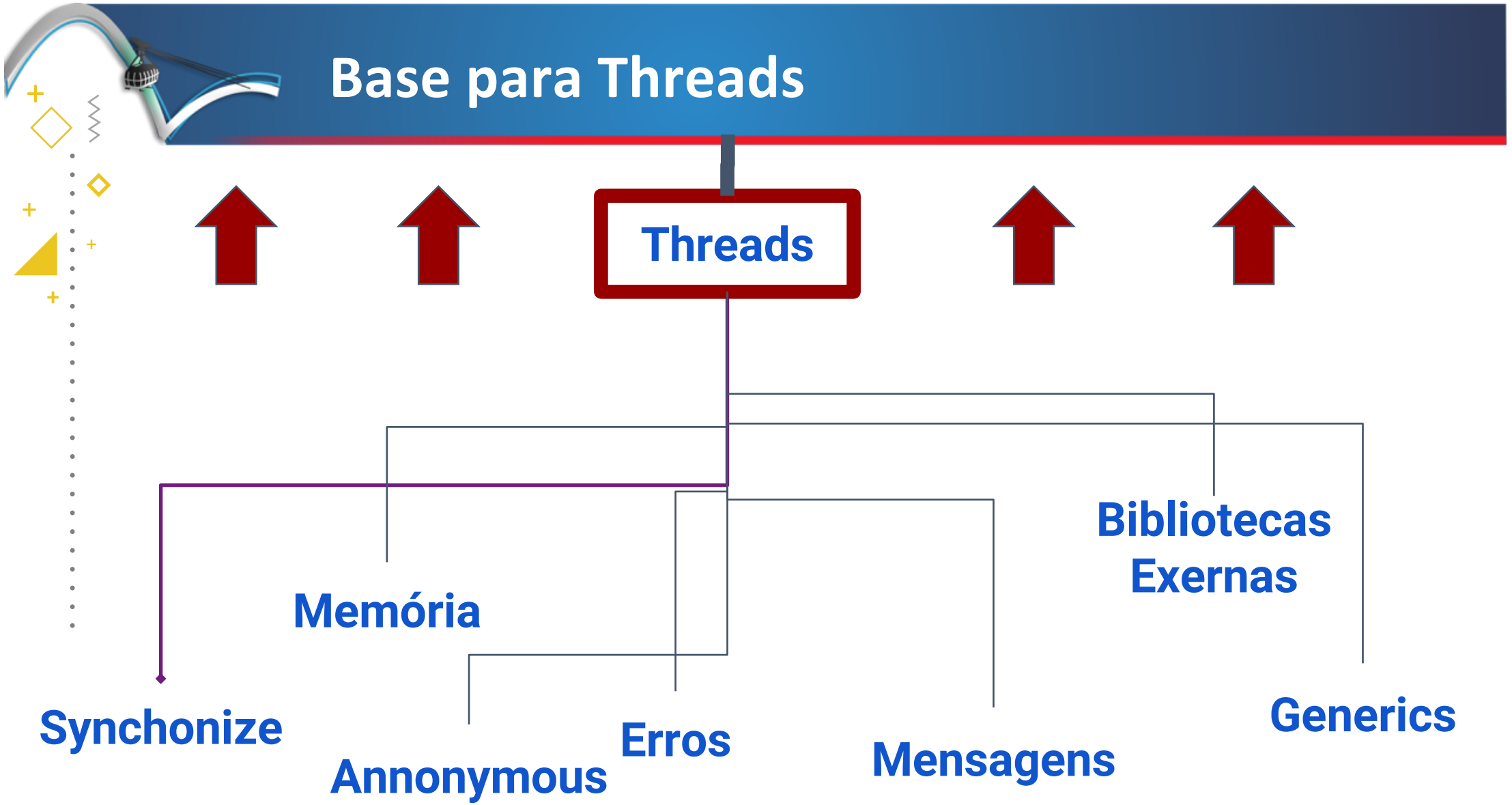
- Multiplataforma
- Recursos exclusivos
  - Mutexes por condition wait
  - TryLock de Mutex
  - Attributes

Já tá acabando a palestra

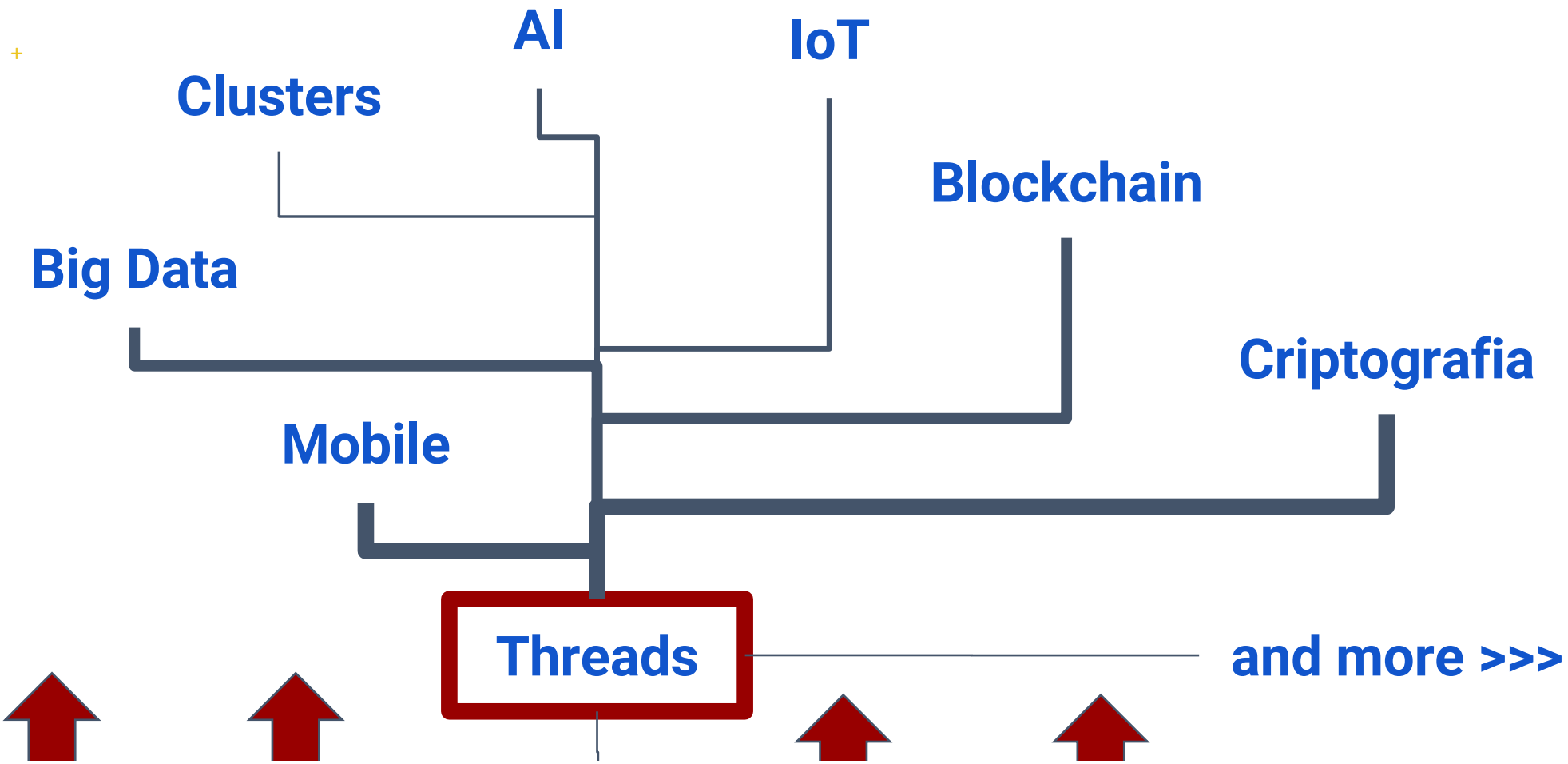
Ufa!

Mas porque tanto paralelismo?!

Base para Threads



Threads and more



embarcadero

OBRIGADO

github.com/diondcm/exemplos-delphi

linkedin.com/in/dion-mai

dion.mai@aquasoft.com.br

Embarcadero Conference

Referências

https://ieeexplore.ieee.org/document/363126

https://en.wikipedia.org/wiki/Gustafson%27s_law

https://en.wikipedia.org/wiki/Amdahl%27s_law

https://unix.stackexchange.com/questions/80424/why-using-more-threads-makes-it-slower-than-using-less-threads

http://www.danielehrman.com/blog/2014/1/29/evaluating-multithreaded-performance

https://mariadb.org/mariadb-5-5-thread-pool-performance/

https://www.cnblogs.com/jpfss/p/9560573.html

https://helloacm.com/simple-tutorial-with-openmp-how-to-use-parallel-block-in-cc-using-openmp/

https://www.slideshare.net/jsquyres/test-presentation-883454

https://www.researchgate.net/publication/311675224_POSIX_Thread_-_Introduction_to_multithreading

Referências

https://www.researchgate.net/publication/311675224_POSIX_Thread_-_Introduction_to_multithreading

https://github.com/BeRo1985/pasmp

http://man7.org/linux/man-pages/man3/pthread_create.3.html

https://www.embarcados.com.br/threads-posix/

http://blog.paolorossi.net/2017/09/04/building-a-real-linux-daemon-with-delphi-part-2/

http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Using_TParallel.For_from_the_Parallel_Programming_Library

http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Using_TTask.IFuture_from_the_Parallel_Programming_Library

http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Using_TTask_from_the_Parallel_Programming_Library

Referências

https://www.amazon.com/Introduction-Parallel-Computing-Ananth-Grama/dp/0201648652/