PGAS: Partitioned Global Address Space

Grupo:
Jan Pierry
Júlia Nakayama
Leon Daros

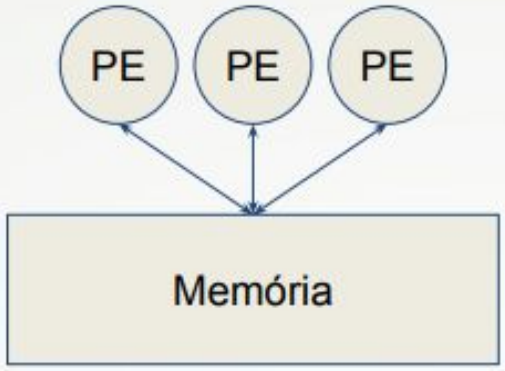
Sumário

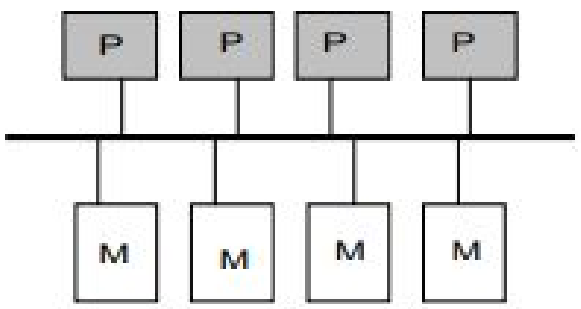
- Modelos de Programação paralela
- PGAS
- Comparações com outros modelos
- Linguagens de programação PGAS
- Conclusão

Modelos de Programação Paralela

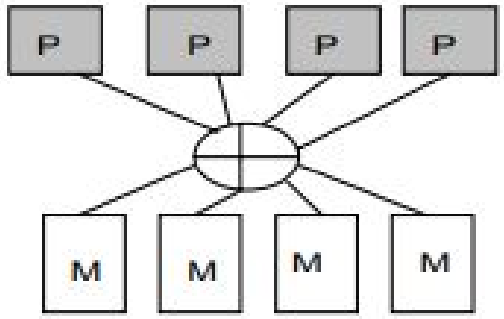
Memória Compartilhada

- Espaço de endereçamento unificado;
- Acesso direto aos dados por todas as threads;
- Sincronização entre tarefas;
- Paralelização a nível SMP (Symmetric Multi-Processing);
- Fácil de programar.





Configuração Compartilhada

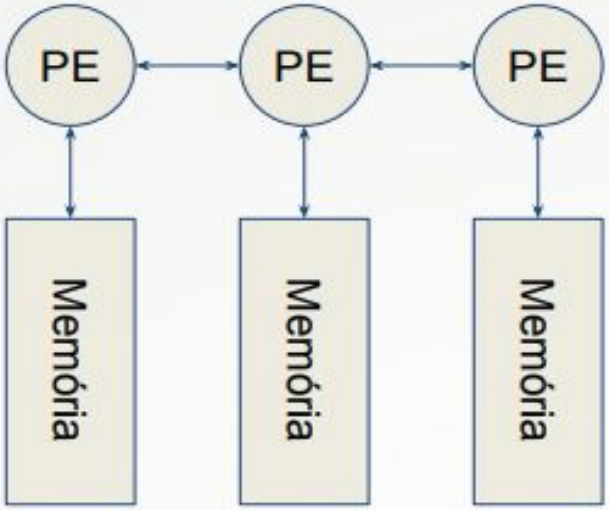


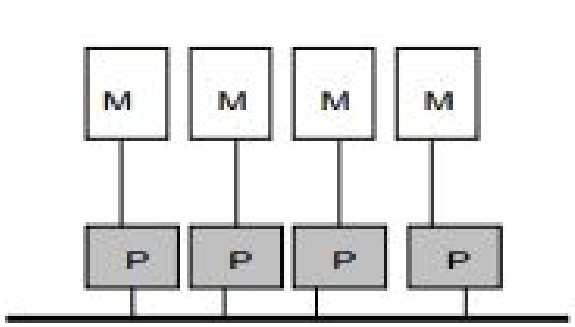
Configuração Comutada

P – Processador M - Memória

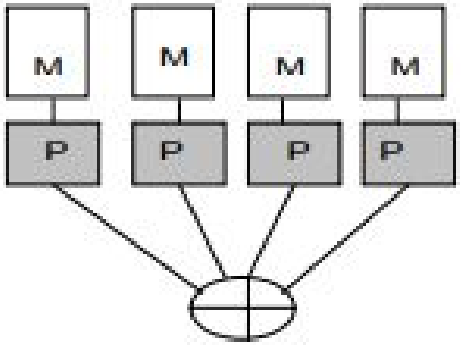
Memória Distribuída

- Espaço de endereçamento separado;
- Comunicação é feita através de troca de mensagens;
- Paralelização a nível de cluster;
- Maior escalabilidade.





Configuração Compartilhada



Configuração Comutada

P – Processador M - Memória

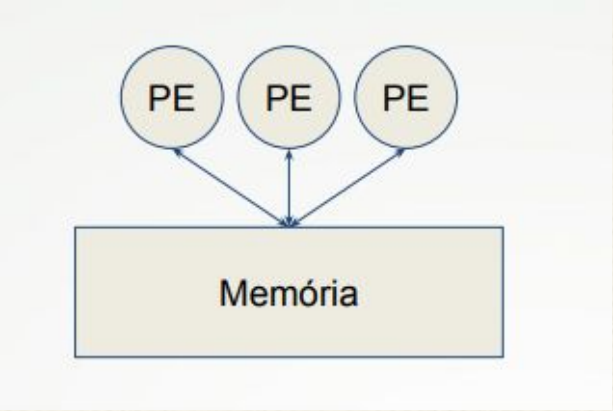
O Modelo PGAS

É um modelo onde existe uma memória global de espaço de endereçamento que está logicamente distribuída entre processos, threads, ou elementos de processamento.

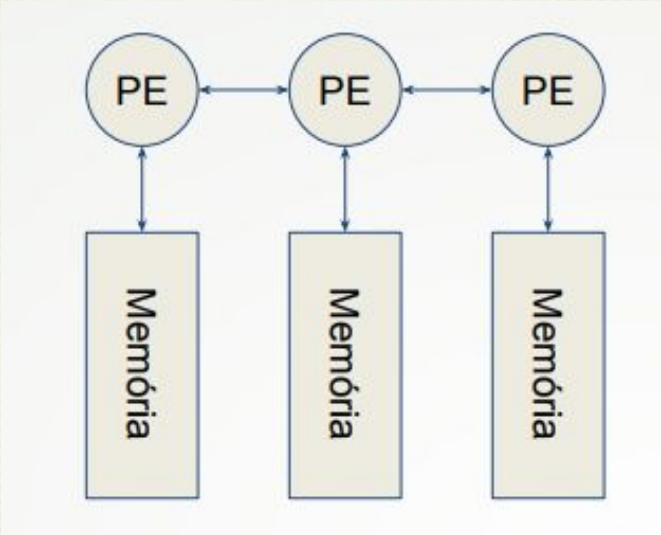
Tem o objetivo de melhorar a produtividade e, ao mesmo tempo visa a alta performance.

Comparação

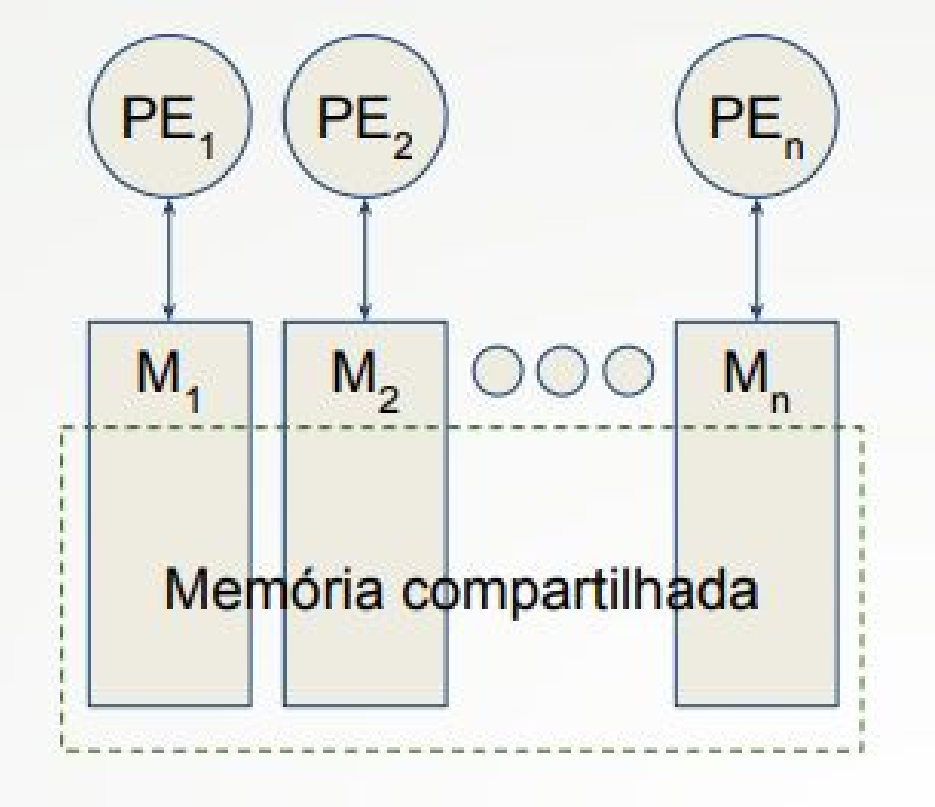
Memória Compartilhada



Memória Distribuída



Memória Compartilhada Distribuída (PGAS)



PGAS vs Outros Modelos

	UPC, X10, Chapel, CAF, Titanium	MPI	OpenMP
Memory model	PGAS	Distributed Memory	Shared Memory
Notation	Language	Library	Annotations
Global arrays?	Yes	No	No
Global pointers/ references?	Yes	No	No
Locality exploitation?	Yes	Yes, necessarily	No

Fonte: http://www.netlib.org/utk/people/JackDongarra/WEB-PAGES/SPRING-2012/Lect09-PGAS-UPC.pdf

Linguagens de Programação PGAS

- UPC
- Coarray Fortran
- Titanium

UPC

- UPC é uma extensão da linguagem C
  - Variável global MYTHREAD específica o index da thread (0..THREADS-1)
  - Número de threads pode ser modificado durante o tempo de execução
- Modelo SPMD de execução
- Comunicação de thread explícita
  - Acesso direto a variáveis compartilhadas
  - Primitivas para sincronização de threads

UPC

- variáveis podem ser private ou shared
- variáveis compartilhadas podem ser usadas por todos mas explicitam o custo de comunicação
- variáveis escalares compartilhadas: sempre na thread 0
- vetores distribuídos entre threads

Variáveis Shared

- As variáveis compartilhadas são explicitamente anotadas com a palavra-chave "shared"
- Somente permitido para variáveis de escopo global (estático, externo)
- Exemplo:

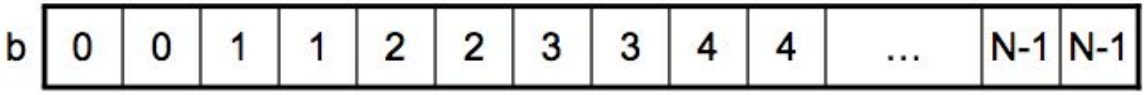
```
int A;          /* each thread keeps a separate copy of A */
shared int B;   /* single instance of B accessed by all threads
```

Thread	0	1	2	3	...	THREADS-1
Private	A	A	A	A	...	A
Shared	B					

Particionamento de arrays compartilhados em UPC

- Arrays compartilhados são distribuídos entre todas as threads
- Os elementos são distribuídos através do escalonamento round-robin
- Número de elementos por thread determinado pelo fator de bloqueio (default = 1)

```
shared [2] int b[THREADS*2];      /* 2 elements per thread */
```



Hello Word em UPC

```
#include <upc.h>
#include <stdio.h>

int main() {
  printf("Thread %d of %d: Hello UPC world\n", MYTHREAD,
THREADS);
  return 0;
}
```

```
hello > xlupc helloWorld.upc
hello > env UPC_NTHREADS=4 ./a.out
Thread 1 of 4: Hello UPC world
Thread 0 of 4: Hello UPC world
Thread 3 of 4: Hello UPC world
Thread 2 of 4: Hello UPC world
```

Coarray Fortran

- Extensão do Fortran
- Modelo de Execução SPMD
  - Número fixo de threads, todas sendo executadas ao mesmo tempo
- Comunicação explícita de thread
  - Coarray para troca de dados entre threads
  - Funções integradas para sincronização

Coarray Fortran

- Coarrays são variáveis Fortran que são replicadas em várias threads
- A codimensão explícita determina a propriedade dos dados
  - Especificado com colchetes após declaração e acesso variável

Image	1	2	3	4	...	NUM_IMAGES()
Coarray	A[1]	A[2]	A[3]	A[4]		A[N]
Private	B	B	B	B	...	B

Sincronização de imagens em Coarray Fortran

- Nenhum armazenamento implica na ordem de acesso entre threads
  - sync memory: garante que os acessos de armazenamento sejam concluídos antes de continuar
- Sincronização de barreira
  - Sync all: Espera que todas as imagens atinjam o ponto de sincronização
- Sincronização parcial
  - sync images: espera que o grupo de imagens listado atinja o ponto de sincronização

Hello Word em Coarray Fortran

```
program abc
write (*,*) 'Hello world from thread', THIS_IMAGE(), 'of',
NUM_IMAGES()
end
```

```
$ xlf90_r -qcaf hello.f
  ** abc    === End of Compilation 1 ===
  1501-510  Compilation successful for file hello.f.
$ env CAF_NUM_IMAGES=4 ; poe a.out -hfile ~/.rhosts
  Hello world from thread 2 of 4
  Hello world from thread 3 of 4
  Hello world from thread 4 of 4
  Hello world from thread 1 of 4
```

Conclusão

- O modelo PGAS parte de uma arquitetura híbrida para obter o melhor das duas arquiteturas:
  - Menor custo de desenvolvimento (memória compartilhada)
  - Alta performance (memória distribuída)
- Suas linguagens permitem a paralelização a nível de SMP e a nível de Cluster.
- Já existem algumas linguagens que estão seguindo este paradigma, como
  - UPC
  - Coarray Fortran
  - Titanium.
- Essas linguagens possuem certos desafios como a padronização e a Integração com frameworks e ferramentas atuais.

Referências

http://www.inf.ufsc.br/~frank.siqueira/INE5418/2.5.DSM-Folhetos.pdf

http://www.netlib.org/utk/people/JackDongarra/WEB-PAGES/SPRING-2012/Lect09-PGAS-UPC.pdf

https://www.osc.edu/sites/osc.edu/files/staff_files/dhudak/pgas-tutorial.pdf

https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/99331/305016.pdf?sequence=1&isAllowed=y

https://siaiap32.univali.br/seer/index.php/acotb/article/viewFile/6230/3491

https://en.wikipedia.org/wiki/Partitioned_global_address_space

http://www.inf.puc-rio.br/~noemi/pcp-16/aula8/lpp1.pdf

http://spscicomp.org/ScicomP16/presentations/PGAS.pdf