

The background of the slide features a light blue gradient with a faint, semi-transparent image of classical architectural columns on the left side. The columns are white and have ornate capitals. The entire slide is framed by a thin white border.

# Computação paralela e distribuída

Computação paralela com a utilização do PVM (*Parallel Virtual Machine*)

Prof. Dr. Luciano José Senger

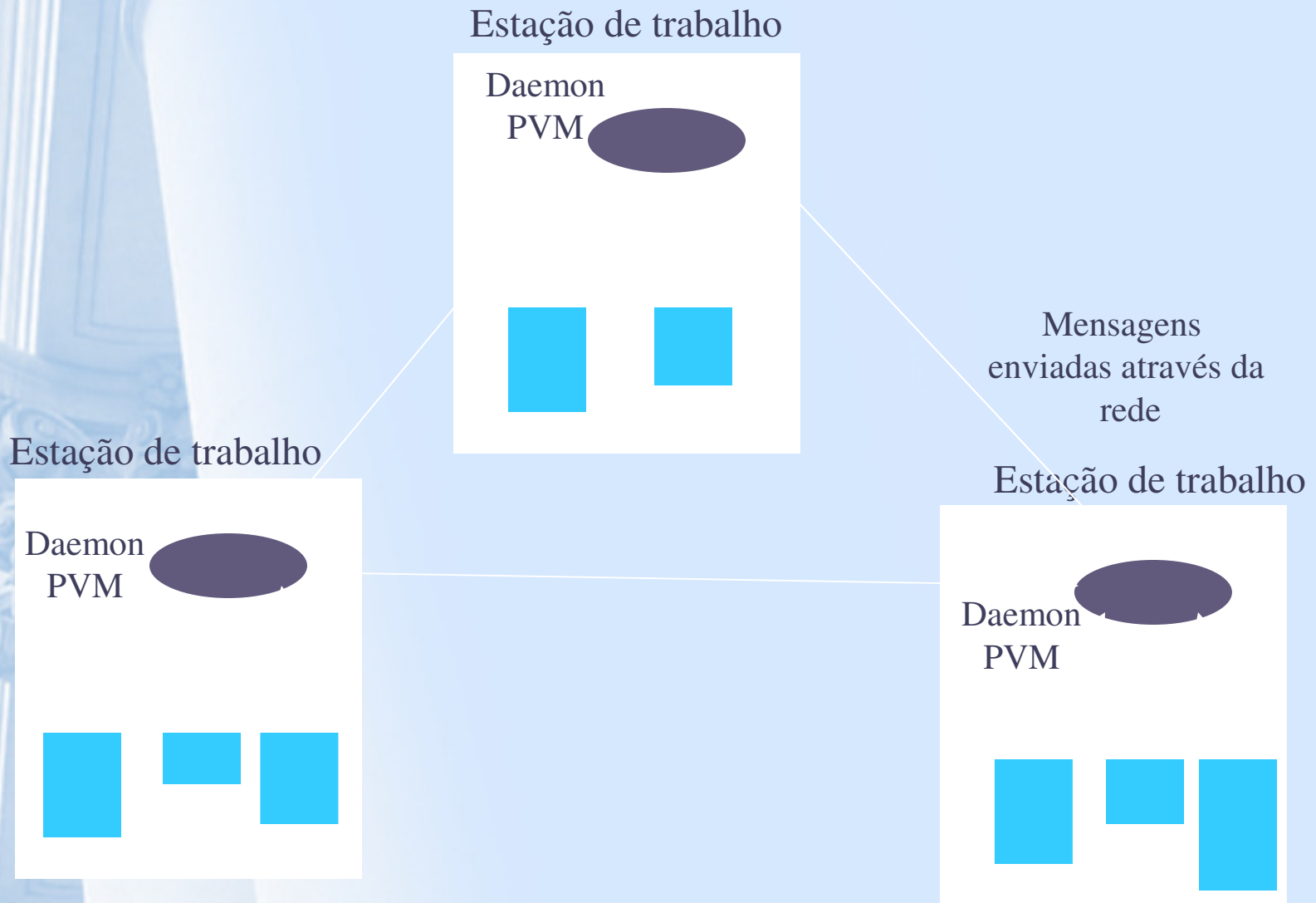
## Parallel Virtual Machine (PVM)

- O programador decompõe o programa em programas separados e cada um deles será escrito em C ou Fortran e compilado para ser executado nas diferentes máquinas da rede
- O conjunto de máquinas que será utilizado para processamento deve ser definido antes do início da execução dos programas
- Cria-se um arquivo (*hostfile*) com o nome de todas as máquinas disponíveis que será lido pelo PVM
- O roteamento de mensagens é feito pelos processos *daemon* do PVM

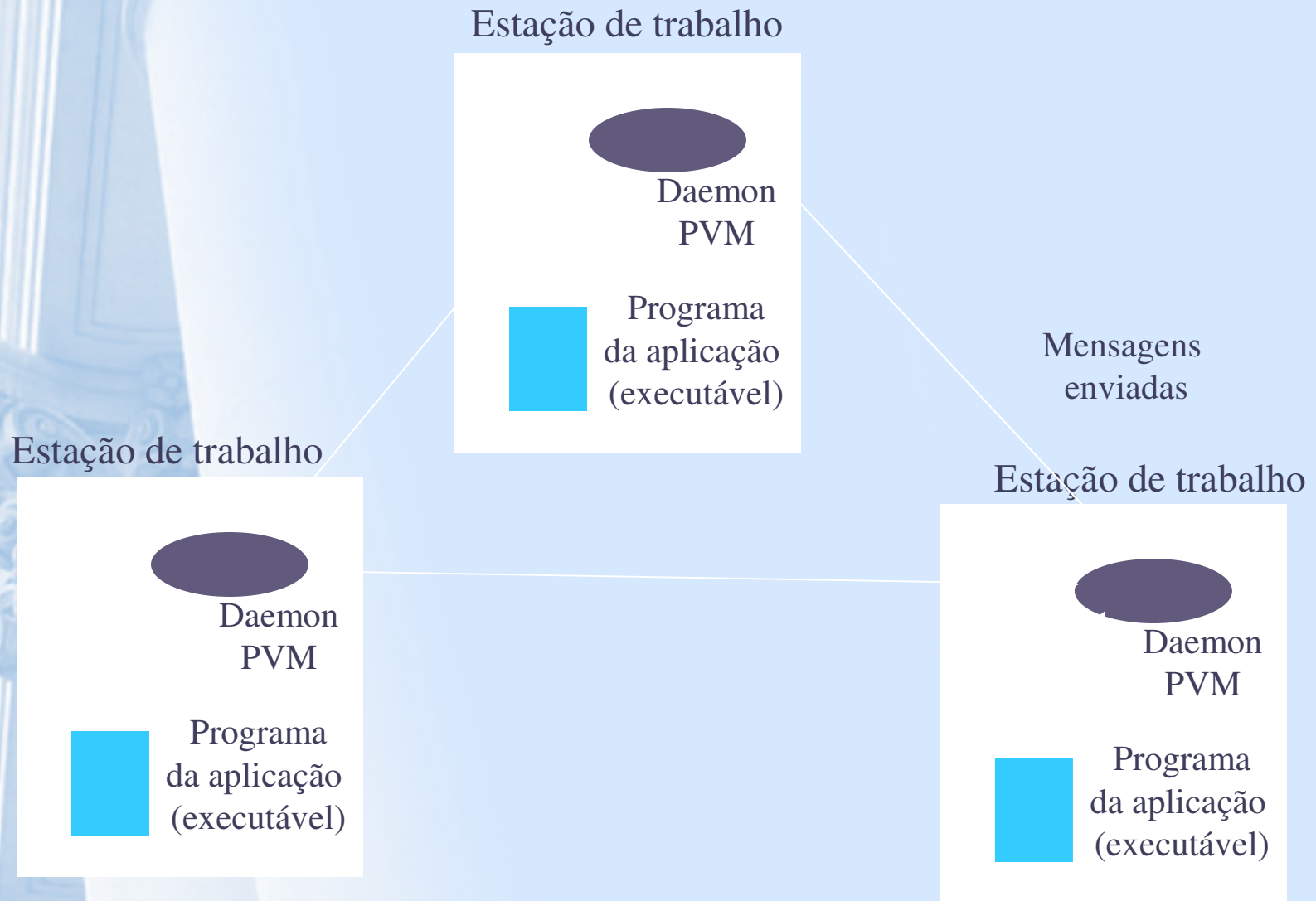
## Parallel Virtual Machine (PVM)

- Máquina virtual dinamicamente configurável
- Protocolo de transmissão de mensagens de alta-performance
- Interface extremamente simples
- Contém primitivas de alto-nível como broadcast e sincronização com barreiras

# Passagem de mensagens utilizando o PVM



# Passagem de mensagens utilizando o PVM



## Rotinas básicas do PVM

- Todas as rotinas são sem bloqueio (ou assíncronas na terminologia PVM) enquanto que rotinas de recebimento podem ser com bloqueio (síncronas) ou sem bloqueio
- Utiliza um rótulo para mensagem (*message tag*)
- Identificador inteiro para o índice ( *tid* )
- `pvm_psend()` e `pvm_precv()`
  - rotinas utilizadas quando dados sendo transmitidos são todos do mesmo tipo

## Envio de mensagem com dados de vários tipos

- Os dados são empacotados em um buffer antes de enviá-los
- O receptor tem que desempacotá-los e acordo com o formato em que foram empacotados
- Rotinas específicas para empacotamento e desempacotamento para cada tipo de dados

# Mensagem PVM como uma estrutura de dados

processo\_1

```
pvm_initsend();  
  
pvm_pkint(.. &x ..);  
pvm_pkstr( ...&s ...);  
pvm_pkfloat( ... &y ...);  
pvm_send(processo_2 ...);
```

Buffer de  
envio

processo\_2

x  
s  
y

Buffer de  
recepção

```
pvm_rcv(processo_1 ...);  
pvm_upkint(.. &x ..);  
pvm_upkstr( ...&s ...);  
pvm_upkfloat( ... &y ...);
```

## Rotinas coletivas

- Rotinas coletivas são implementadas em uma biblioteca de software específica (não precisa ser compilada com a aplicação)
- Operações utilizadas com grupo de processos (`pvm_bcast()`, `pvm_scatter()`, `pvm_gather()` e `pvm_reduce()`) com exceção de multicast (`pvm_mcast`)
- Um processo se junta a um grupo através da rotina `pvm_ingroup()`
- `pvm_bcast` envia mensagem para cada membro do grupo
- `pvm_gather()` coleta valores de cada membro do grupo

# Exemplo de programa (mestre)

```
#include <stdio.h>
#include <stdlib.h>
#include <pvm3.h>
#define SLAVE ``spsum``
#define PROC 10
#define NELEM 1000
main() {
    int mytid, tids[PROC]
    int n=NELEM, nproc =PROC;
    int no, i, who, msgtype;
    int data[NELEM],result[PROC],tot=0;
    char fn[255];
    FILE *fp;
    mytid=pvm_mytid();
    no=pvm_spawn(SLAVE, (char *)0,0,`` `` ,nproc,tids);
    if (no < nproc) {
        printf(``Erro criar escravo\n``);
        for (i=0; i<no; i++) pvm_kill(tids[i]);
        pvm_exit(); exit(1);
    }
    strcpy(fn,getenv(``HOME``));
    strcat(fn,``/pvm3/src/rand_data.txt``);
    if ((fp=fopen(fn,``r``))==NULL) {
        printf(``Nao posso abrir arquivo %s \n``,fn);
        exit (1);
    }
    for (i=0; i<n; i++) fscanf(fp,"%d",&data[i]);
```

# Exemplo de programa (trabalhador)

```
#include <stdio.h>
#include <pvm3.h>
#define PROC 10
#define NELEM 1000
main()  {
    int mytid, tids[PROC]
    int n,me, i,msgtype;
    int x, nproc, master;
    int data[NELEM], sum;
    int x, low, high;

    mytid=pvm_mytid();
```

Mestre

```
pvm_initsend(PvmDataDefault);
msgtype=0;
pvm_pkint(&nproc,1,1);
pvm_pkint(tids, nproc, 1);
pvm_pkint(&n,1,1);
pvm_pkint(data, n, 1);
pvm_mcast(tids, nproc, type);

msgtype = 5;
for (i=0; i<nproc; i++) {
    pvm_rcv(-1, msgtype);
    pvm_upkint(&who, 1, 1);
    pvm_upkint(&result[who], 1, 1);
    printf(“%d de %d \n”,result[who],who);
}

for (i=0;i<nproc;i++) tot +=result[i];
printf (“O total e %d. \n \n”,tot);

pvm_exit();
return(0);
}
```

trabalhador

```
msgtype=0;
pvm_rcv(-1,msgtype);
pvm_upkint(&nproc,1,1);
pvm_upkint(tids, nproc, 1);
pvm_upkint(&n, 1,1 );
pvm_upkint(data, ,n,1);

for (i=0;i<nproc;i++)
    if (mytid ==tids[i])
        { me=i;break;}

x=n/proc;
low=me*x;
high=low+x;
for (i=low;i<high;i++)
    sum+=data[i];

pvm_initsend(PvmDataDefault);
pvm_pkint(&me, 1, 1);
pvm_pkint(&sum,1,1);
msgtype=5;
master=pvm_parent();
pvm_send(master,msgtype);

pvm_exit();
return(0);
}
```

Broadcast

Recebe resultados

## Sítios internet

- <http://www.epm.ornl.gov/pvm/>
- <http://www.netlib.org/pvm3/>
- [http://www.math.cmu.edu/Parallel\\_Cluster/pvm.html](http://www.math.cmu.edu/Parallel_Cluster/pvm.html)