PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

ANA CAROLINA MORELLI CHAVES 23017617

LAIS DE PAULA LEMOS 23016041

LUIZ GUSTAVO PINTO DA SILVA 23013028

DIÁRIO DO PROJETO – ATIVIDADE 2

PROGRAMAÇÃO PARALELA E DISTRIBUÍDA

CAMPINAS

2025

Diário de Desenvolvimento — Atividade de Programação Paralela e Distribuída Em Java

Início do Projeto – 17/11

Iniciamos esta nova atividade com o trio: Ana, Luiz e Lais. Depois de ler o enunciado e revisar a atividade anterior (Busca Paralela e Distribuída), percebemos que boa parte da infraestrutura já poderia ser reaproveitada, principalmente a parte de socket, comunicação e classes de apoio.

O professor também orientou que poderíamos reutilizar logicamente a base anterior desde que adotássemos o objetivo do programa. Assim, nos primeiros dias do projeto, nosso foco foi reformular a lógica de busca para transformá-la em lógica de ordenação.

Distribuição de funções:

- **Ana**: adaptação das classes de comunicação (Comunicado, Pedido, Resposta), ajustes no Distribuidor.

- **Luiz**: implementação da lógica de ordenação paralela nos Receptores e sincronização das respostas.

- **Lais**: integração, documentação, testes comparativos e revisão geral do fluxo de dados.

O trio começou o projeto alinhado e confiante, pois a estrutura do trabalho anterior trouxe segurança para seguir.

Dias de Reformulação da Atividade Anterior – 17/11 e 18/11

Nesses dois dias o objetivo foi transformar a antiga atividade de busca em uma implementação de Ordenação Paralela e Distribuída.

O que foi alterado:

- Os pedidos agora carregam vetores a serem ordenados.

- Os receptores recebem partes diferentes do vetor e ordenam localmente usando algoritmos simples como *Arrays.sort*.

- O *Distribuidor* passou a dividir o vetor em blocos iguais, enviar para máquinas/receptores diferentes e depois mesclar os blocos ordenados.

- A classe Resposta agora retorna uma parte ordenada do vetor, não mais uma contagem.

Durante essa reformulação, enfrentamos desafios principalmente na parte de intercalação dos segmentos ordenados, já que no projeto anterior não havia necessidade de juntar resultados complexos, apenas somar contagens. Também revisitamos a comunicação entre sockets para garantir que a serialização funcionasse com os novos tipos de dados.

Desenvolvimento Prático – 19/11 a 21/11

Com a reformulação pronta, começamos a implementar, testar e ajustar:

19/11 — Implementação da ordenação paralela nos Receptores

Luiz criou o método de ordenação local e garantiu que cada Receptor tratava corretamente seu subvetor.

19/11 — Distribuidor adaptado

Ana alterou todo o fluxo de:

- dividir o vetor
- enviar para os Receptores
- aguardar respostas
- ordenar/mesclar tudo em um vetor final

O maior desafio foi ajustar o método de mesclagem dos blocos.

20/11 — Testes iniciais

Lais configurou toda a estrutura de testes utilizando dois IPs diferentes pelo Radmin VPN, simulando o ambiente distribuído real.

21/11 — Testes comparativos

Foram usados vetores de:

- 10.000 elementos
- 100.000 elementos
- 1.000.000 elementos
- 100.000.000 elementos
- 1.000.000.000 elementos

Testamos:

- tempo da ordenação paralela e distribuída
- tempo da versão sem paralelismo (feito pelo Luiz para comparação)

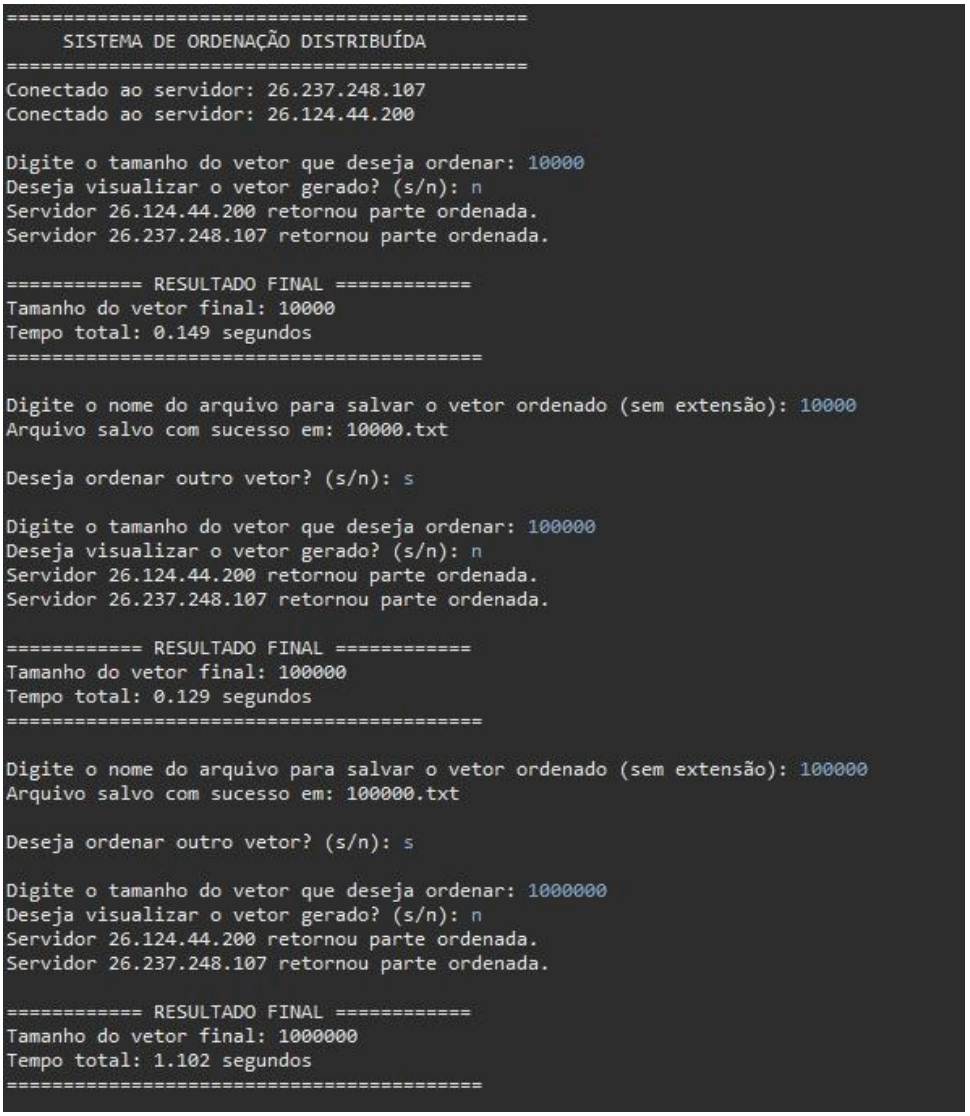
Medimos em ambos os servidores:

- IPs pelo Radmin VPN:
  - 26.237.248.107
  - 26.124.44.200

Esse processo permitiu validar ganhos e perdas da abordagem distribuída.

Logs dos teste realizados:

Distribuidor**:**



```
==========================================
     SISTEMA DE ORDENAÇÃO DISTRIBUÍDA
==========================================
Conectado ao servidor: 26.237.248.107
Conectado ao servidor: 26.124.44.200

Digite o tamanho do vetor que deseja ordenar: 10000
Deseja visualizar o vetor gerado? (s/n): n
Servidor 26.124.44.200 retornou parte ordenada.
Servidor 26.237.248.107 retornou parte ordenada.

============ RESULTADO FINAL ============
Tamanho do vetor final: 10000
Tempo total: 0.149 segundos
========================================

Digite o nome do arquivo para salvar o vetor ordenado (sem extensão): 10000
Arquivo salvo com sucesso em: 10000.txt

Deseja ordenar outro vetor? (s/n): s

Digite o tamanho do vetor que deseja ordenar: 100000
Deseja visualizar o vetor gerado? (s/n): n
Servidor 26.124.44.200 retornou parte ordenada.
Servidor 26.237.248.107 retornou parte ordenada.

============ RESULTADO FINAL ============
Tamanho do vetor final: 100000
Tempo total: 0.129 segundos
========================================

Digite o nome do arquivo para salvar o vetor ordenado (sem extensão): 100000
Arquivo salvo com sucesso em: 100000.txt

Deseja ordenar outro vetor? (s/n): s

Digite o tamanho do vetor que deseja ordenar: 1000000
Deseja visualizar o vetor gerado? (s/n): n
Servidor 26.124.44.200 retornou parte ordenada.
Servidor 26.237.248.107 retornou parte ordenada.

============ RESULTADO FINAL ============
Tamanho do vetor final: 1000000
Tempo total: 1.102 segundos
========================================
```

```
Digite o nome do arquivo para salvar o vetor ordenado (sem extensão): 1000000
Arquivo salvo com sucesso em: 1000000.txt

Deseja ordenar outro vetor? (s/n): s

Digite o tamanho do vetor que deseja ordenar: 10000000
Deseja visualizar o vetor gerado? (s/n): n
Servidor 26.124.44.200 retornou parte ordenada.
Servidor 26.237.248.107 retornou parte ordenada.

============ RESULTADO FINAL ============
Tamanho do vetor final: 10000000
Tempo total: 6.92 segundos
=========================================

Digite o nome do arquivo para salvar o vetor ordenado (sem extensão): 10000000
Arquivo salvo com sucesso em: 10000000.txt

Deseja ordenar outro vetor? (s/n): s

Digite o tamanho do vetor que deseja ordenar: 100000000
Deseja visualizar o vetor gerado? (s/n): n
Servidor 26.124.44.200 retornou parte ordenada.
Servidor 26.237.248.107 retornou parte ordenada.

============ RESULTADO FINAL ============
Tamanho do vetor final: 100000000
Tempo total: 73.034 segundos
=========================================

Digite o nome do arquivo para salvar o vetor ordenado (sem extensão): 100000000
Arquivo salvo com sucesso em: 100000000.txt

Deseja ordenar outro vetor? (s/n): n

Encerrando servidores...
Todos os servidores foram encerrados.
Programa finalizado!
```

Receptor 1:

```
[Aguardando conexão de um cliente...]
Conexão aceita de: 26.83.104.38
[R] Streams de comunicação configuradas.

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 5000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 5 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 50000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 5 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!
```

```
[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 500000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 11 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 5000000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 87 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 50000000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 905 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Cliente solicitou encerramento da conexão.
[R] Conexão encerrada com o cliente.
```

Receptor 2:

```
[Aguardando conexão de um cliente...]
Conexão aceita de: 26.83.104.38
[R] Streams de comunicação configuradas.

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 5000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 7 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 50000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 8 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!
```

```
[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 500000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 14 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 5000000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 152 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!
```

```
[R] Aguardando objeto do cliente...
[R] Pedido recebido!
 → Vetor recebido com 50000000 elementos
 → Iniciando ordenação paralela...
 → Ordenação concluída em 1505 ms
 → Enviando resposta para o cliente...
[R] Resposta enviada com sucesso!

[R] Aguardando objeto do cliente...
[R] Cliente solicitou encerramento da conexão.
[R] Conexão encerrada com o cliente.
```

Conclusão da Implementação – 22/11

Finalizamos o projeto oficialmente no dia 22/11, após validar todas as medições e documentar os resultados.

Resultados Observados

- Para vetores pequenos, o paralelismo não traz grande vantagem, porque o custo da comunicação entre Distribuidor e Receptores é maior que o ganho.

- Em vetores grandes (a partir de 10.000 elementos), já conseguimos perceber diferenças mais claras, com os receptores trabalhando em paralelo.

- O merge final ainda exige processamento, mas o desempenho geral melhorou quando vários receptores foram usados.

Impressões do trio

Ana: achou interessante observar como pequenas mudanças na estrutura de comunicação afetam todo o sistema — principalmente a necessidade de enviar e receber vetores inteiros via sockets. A ordenação paralela trouxe desafios novos e mais complexos que a busca.

Luiz: gostou de trabalhar na lógica de paralelismo e observar o ganho de desempenho. Ressaltou que o desafio maior foi o merge dos resultados, já que o paralelismo em ordenação é mais complexo que em busca.

Lais: destacou a importância dos testes comparativos, pois eles mostraram claramente como o paralelismo se comporta conforme o tamanho dos vetores cresce. A experiência ajudou a consolidar conceitos como divisão de tarefas, sincronização e análise de desempenho.

O projeto de Ordenação Paralela e Distribuída permitiu que o trio aplicasse conhecimentos mais avançados de programação distribuída e paralelismo. Ao reutilizar e reformular a base da atividade anterior, conseguimos enxergar na prática como um mesmo arcabouço pode ser adaptado para tarefas computacionais totalmente diferentes. A experiência trouxe amadurecimento técnico e mais segurança no uso de threads, sockets, divisão de tarefas e sincronização entre vários processos.